Scalable Computing: Practice and Experience

Scientific International Journal
for Parallel and Distributed Computing

ISSN: 1895-1767

Universitatea de Vest
din Timişoara

Volume 22(4) December 2021

Scalable Computing: Practice and Experience

Volume 22, Number 4, December 2021

TABLE OF CONTENTS

Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2021 SCPE. Volume 22, Issues 4, pp. 387–400, DOI 10.12694/scpe.v22i4.1870

A METHOD TO IMPROVE EXACT MATCHING RESULTS IN COMPRESSED TEXT USING PARALLEL WAVELET TREE

SHASHANK SRIVASTAV[*], PRADEEP KUMAR SINGH[†], AND DIVAKAR YADAV[‡]

Abstract. The process of searching on the World Wide Web (WWW) is increasing regularly, and users around the world also use it regularly. In WWW the size of the text corpus is constantly increasing at an exponential rate, so we need an efficient indexing algorithm that reduces both space and time during the search process. This paper proposes a new technique that utilizes Word-Based Tagging Coding compression which is implemented using Parallel Wavelet Tree, called WBTC_PWT. WBTC_PWT uses the word-based tagging coding encoding technique to reduce the space complexity of the index and uses a parallel wavelet tree which reduces the time it takes to construct indexes. This technique utilizes the features of compressed pattern matching to minimize search time complexity. In this technique, all the unique words present in the text corpus are divided into different levels according to the word frequency table and a different wavelet tree is made for each level in parallel. Compared to other existing search algorithms based on compressed text, the proposed WBTC_PWT search method is significantly faster and it reduces the chances of getting the false matching result.

Key words: Parallel computing, Wavelet Tree, Compressed Text Matching, Text Searching, Word-Based Tagged Coding, Compressed Indexing.

AMS subject classifications. 68W10

1. Introduction.

In recent years, the production of a wide variety of devices such as powerful laptops, tablets, Wi-Fi TVs, various electronic gadgets and other mobile devices has been growing rapidly in the field of information technology. All these devices can connect directly to the Internet and create large amounts of data and search within the produced data [14][35]. This is an example of how we are living in an era of information explosion. If the size of the data is too large, it becomes a costly affair to store, process, retrieve and communicate data on such a large scale. Therefore, we need to do data compression to manage those huge data. Compression of data brings stability to the data and renders it in the minimum number of bit-space. The compression of data involves the process of data-encoding and data-decoding. This article uses the term data packing to refer to encoding and the term data un-packing to refer to decoding.

String matching is a method of identifying all possible pattern (string/substring) events from a large text corpus. String matching features are used in various applications for information retrieval, big data, text mining, plagiarism checking, DNA matching, and more. Compressed pattern matching (CPM) is known as a process of matching string in compressed data. CPM [2][20][27] is a process in which string matching is performed directly on compressed text without the need for decompression. Compared to other algorithms, CPM supported compression algorithms are considered more effective. During the decompression process, CPM saves time wastage and reduces search time. To save disk space, CPM algorithms are used and is also used to transfer a vast amount of information over a data network. CPM is introduced using the Lempel – Ziv – Welch (LZW) compression technique in [1] by Aamir et al. The CPM problem is to observe and reveal each event of P in T, in O(u + m) time, using only P and Z, the material T, where u is the length of compressed material Z and m is the length of an example pattern P. The upside of CPM is that instead of matching it to unpacked records, it directly matches the instance in the packed document and thus optimizes the search time. Later CPM is developed by M. Farch et al. [9] using the LZ77 compression process. For large text databases, compression based on the Huffman technique is not considered efficient, as one achieves a lower compression ratio using

[*]Department of Computer Science and Engineering, MMMUT Gorakhpur, India. (shashank07oct@gmail.com).
[†]Department of Computer Science and Engineering, MMMUT Gorakhpur, India.(topksingh@gmail.com).
[‡]Department of Computer Science and Engineering, NIT Hamirpur, India.(divakaryadav@nith.ac.in) .

the Huffman compression technique. On the other hand, LZW family compression techniques (LZW77, LZ78, etc.) produce a very good compression ratio, but the problem is that they cannot be considered efficient when searching for patterns directly in the packed content. Various techniques can be used to solve this problem. In [8][24], CPM is performed using straight-line software (SLP). The SLP uses a plot based on the structure of the sentence. The run-length encoding (RLE) approach suggested in [7] is used as an example for matching, where template matching was designed using Boyer Moore [3] and Knuth Morris Pratt (KMP) [22].

In [36], the authors introduce procedures for searching in packed data for Huffman's text. To fit the example material inside the compressed material, they used KMP measurements, but the correct match is not reliably distributed. One of the problems with this packing technique is false matching, as stated in [36] that the Huffman character packing technique is modified to handle words, and example patterns to execute CPMs. There are more problems of false matching in this situation. The word-based Huffman coding is said to be using bytes instead of bits [32]. In this approach, each specific word in a sample pattern is packed with a combination of bytes instead of bits. Words are packaged with either 128 bits ("tagging-Huffman packaging") or 256 bits ("plain-Huffman packaging"). For the first byte of each word-code of the tagged Huffman package, the 7 least significant bits are used for the Huffman packaging, and the most significant bit is used as the guard bit. Each guard bit used in the word-code is to distinguish its code from other word-codes and to mark the beginning of the code for each word. Thus, the use of this technique easily detects mismatch cases, and the use of bytes does not affect the efficiency of the packaging technique. This technique allows un-packaging of the content at any time and a pattern can also be searched effectively. The word-based Huffman packaging technique treats word-models without spaces.

The work depicted in [16][18] has implemented a new packaging technique, known as word-based tagging coding (WBTC), which enables to pack the contents partially and from any subjective position using the marked bit, enables the material to be easily unpacked. It also supports CPM and can quickly detect false matches. WBTC is a packaging technology that views the word as its basic unit of compression. At each level, each word present in the material has a fixed number of bits. As with other common packing techniques it often gives us false matches in the CPM process. WBTC can also suffer from false matching problems. In [17], the matched strategy uses a linear search on the packaged material but if the material is extensive then this process becomes an expensive one. WBTC codes are generally longer than most methods for packing.

An advanced data structure, the wavelet tree (WT), is used to represent and react to sequences. This data structure is space-efficient and can be used to construct indexes. It supports the rank operation, which detects the occurrence of the word and the select operation, which detects the position of the word and executes these operations within O (1) time. WT has been used previously in [15] and is known as a data structure that can be used as self-indexed, which can be constructed for characters or words, respectively. Various symbols are available on a WT leaf, which is either the character or the word. As seen in [25][28], WT can be used for text indexing as well as spatial search. The workings of WT indexing are explained in [34] using a set of documents on WWW.

1.1. Motivation, Contribution and Organization of paper. In WWW the size of the text corpus is increasing at an exponential rate, so an efficient algorithm is needed that can reduce the time taken during the search process. Due to the large data size, it uses a lot of memory space in the device. So, we need to do data compression to solve the space problem. CPM is one of the ways to reduce data space and provide search options. Many research articles have been studied to address the CPM problem, but both compression and search methods can be further improved. Several CPM solution algorithms have been developed and are given in [2] [12] [15] [18] [17] [16] [19] [20] [21] [8] [36]. The algorithms are given in [12] [15] [21] [20] work efficiently in search but do not provide a good compression ratio. On the other hand, algorithms given in [18] [16] [17] [19] [36] provides a good compression ratio but fail to provide correct matches. Thus, an algorithm should be proposed that provides a decent compression ratio and an accurate search procedure without getting incorrect results.

This paper proposes a word-based compression as well as word-based indexing of text content. As regards the number of bits, the word compression technique provides much better results than the compression technique for characters given in [32], so we choose word-based compression and create indexes using WT. The main contributions of this paper are as follows:

Table 2.1: Huffman word codes for content T

S. No.	**Words**	**Frequencies**	**Huffman word-Codes**
0	Indian	3	1
1	a	2	011
2	good	2	010
3	is	1	0011
4	for	1	0010
5	all	1	0001
6	always	1	0000

- Study of existing word-based compression techniques with examples and study about the possibility of mismatch results in CPM.
- An indexing approach is proposed with the help of WBTC and WT which efficiently solves the CPM problem with no mismatch results.
- Compares the results of the proposed method with the other word-based compression techniques used to solve the problem of CPM.
- The proposed approach can handle both single text patterns and multiple text patterns very efficiently.

Other parts of the paper are organized as follows. The basics and related functions are described in Section 2. Section 3 contains a description of the WBTC_PWT technique. Experimental analysis and demonstration are described in Section 4. Finally, Section 5 describes the conclusions of the proposed algorithm and its future work.

2. Basics and Related Work. In this section, we only focus on the word-based packing technique that supports CPM so that we can search for any query text directly in the packed file. CPM supported word-based techniques include Huffman Packing, WBTC Packing and WT. Here we describe the Huffman packing and WBTC packing methods used exclusively for words and illustrate with examples of how codes are assigned to words in these techniques. Here we also understand WT and how to use it to efficiently search for words in packaged content.

2.1. Huffman Technique of Word-Packing. The Huffman technique uses a clever method to generate packing variable length codes. It is naive to the number of words that are most visible in corpus material. If the frequency of a word increases, we use the least number of bits in the code of that word and vice versa. So, we use this packaging method to reduce the size of the content. In [32], insight into the design of the Huffman word coding is presented. Precedent 1 illustrates the strategy used in Huffman word coding.

Precedent 1. Let's take any material like T = "a good Indian is always a good Indian for all Indian". The above sentence is formed from a collection of 'a', 'always', 'all', 'for', 'good', 'Indian', 'is'. We use the Huffman word-based compression technique to derive the code for each word as presented in Table 2.1. For packing purposes, we only expect a space-less content format. When the word ends with space in a space-less content format, no changes are made to the word's code and for another word that ends with a separator (such as colon, semicolon, comma, etc.), then the word and the separator both are coded separately. Compressed content T' of content T is encoded using Table 1 and is represented as T' = 011 010 1 0011 0000 011 010 1 0010 0001 1, that requires 31 bits. If the same material is compressed using Huffman character packing, so it needs 139 bits to represent T in compressed form. This precedent reflects a huge improvement in bit requirement by using the Huffman word packing on the Huffman character packing.

Depending on the number of bits used for packing, the Huffman packing is divided into two parts - binary and plane. Each word is packaged using 128 bits or 256 bits according to the byte-oriented Huffman packaging technique. When using 128 bits, it is called binary Huffman and when using 256 bits, it is called plain Huffman. As [32] suggested, byte-oriented Huffman packaging technology uses bytes instead of bits without violating the efficiency of packing and promotes unpacking faster than binary Huffman code. We consider only seven lower bits for each byte in a packing made by binary Huffman-tagging, and these 7 bits are used for packing. In each byte, code the most significant bit (MSB) with the following rule: MSB should be 1 for the first byte

Table 2.2: WBTC Code for content T

Indexes	Unique-Words	Frequency	Word-Codes
0	Indian	3	01
1	a	2	10
2	good	2	0001
3	is	1	0010
4	for	1	1101
5	all	1	1110
6	always	1	000001

of the word-code and 0 for the remaining bytes. In this method by this rule, we mark the beginning of each word in the packaged content, which encourages the example text to be properly searched within the packaged material. The plane Huffman-tagging packing is also like the binary Huffman-tagging packing, except that it allocates 256 bits to a word-code. Tagged-Huffman's packing may lose some of its data due to the extra bit used to identify the beginning of a word, so we prefer to use plain Huffman packing because it has a greater number of bits.

2.2. Word-Based Tagging Code (WBTC). WBTC is an efficient tool for compression developed by [18][16]. They have built compression techniques used for dynamic datasets. In this technique, the term has been considered an effective compressed unit rather than a character. This preserves each highlight of sub-optimal code with optimal compression ratios. The risk of mismatch is also low, and it is possible to see the pattern directly in the compressed text. Precedent 2 indicates the coding methodology. Here are the steps that have been taken during the coding process:

Step 1: For m=1, the first 2m unique terms of a text corpus are given a pair of bits as ‘10’ and ‘01’. (level-1)
Step 2: In each code created in the previous step, we add prefix pairs ‘11’ and ‘00’ and code the next group of 2m words. (m=2) (level-2)
Step 3: Utilizing steps above one can normalize the encoding method by adding prefix pairs 11 and 00 of each code created in the last level, we code the 2m words in the next level. (level-m).
Step 4: Steps 1–3 is used repeatedly until the encoding of all the words are performed.

Precedent 2. We again use content T = ”a good Indian is always a good Indian for all Indian”. We perform the coding of all unique words in the above material T using the WBTC packing technique, and the assigned codes are shown in Table 2.2. Now compressed content T’ is given as T’ = 10 0001 01 0010 000001 10 0001 01 1101 1110 01. Here we see that only 36 bits are required to represent the contents of T by the WBTC technique. As seen earlier in precedent 1, the Huffman-word packaging requires 31 bits to represent this content T. WBTC entropy is higher than Huffman word packing for smaller corpus, but the chances of false matching are higher in Huffman word packing compared to WBTC. So, WBTC is an efficient and powerful packaging technique for compressing large volumes of text data.

Searching stage: Inside the compressed content, the search request for the word ‘W’ is completed with the steps below:

Step 1: We first see the word-code C of W that the WBTC coding method assigns.
Step 2: Word-code C is now used to execute CPM by the linear search.

The bit-pairs ’01’ and ’10’ are used as a marker to detect the end of the word-code. For the rest of the words in the next level, the concatenation of bits ’11’ or ’00’ is used as a prefix. WBTC codes are prefix-free, so we can easily identify the beginning and end of each unique word-code. Therefore, the chance of false matching is very low. Assume that sub-string g is postfix of the following string h, e.g. h = fg in which $f\epsilon \sum * \; and \; |g| \leq |h|$. For instance, assume h is given in space less model as h = bcadcaabcacba and we pick g = abcacba from h. Since g is postfix of string h, thus it can be shown as g!h. Assume now that x and y are the word-codes provided to h and g given as x = 1100110001 and y = 0001. Here, y is the postfix of x. Thus, it can be deduced that

Table 2.3: Words with their corresponding codes

Words	**Word-Codes**
good	45 41
Indian	23 25 18
always	25 18
is	41 23 25

the WBTC packing codes are not free of prefixes and thus it is very important to verify whether the match is correct or not. We must confirm whether the match is correct or not. Suppose the word is located at the 'i' position in the packed material. To check this, we must scan the bit pairs before this location 'i'. The match is true if we find 10/01 for the last two bits. The same if we find that the last two bits are 11/00, then the match is not correct. In this way, total match results and fake matches can be obtained quickly. For example, consider searching for the query word Q = 'Indian' in the content T above, in the packaged content T', first we can find that the WBTC word-code for Q is '01'. After that, in the compressed material T', we are looking for the word-code 01 directly in T' = 10 0001 01 0010 000001 10 0001 01 1101 1110 01

We find that the word 'Indian' (word-code = 01) is mixed in packed content T' at three places. The other word-code '01' matches like '0001', '000001', '0001' and '1101' are invalid matches because they do not consist of either '01' or '10' as consecutive bit-pairs before matching positions. Here we see that by using WBTC compression we can directly search into packaged content without the need for decompression and WBTC provided codes have less chance of a faulty matching, as shown in [26].

2.3. False Matching in CPM. The codes provided by the different packaging methods are free of prefixes. The compression ratio may be okay, but the main objective of these approaches is not to perform a search directly within the packaged content, as it may result in an invalid match. Precedent 3 addresses how to obtain false matching [32] [12] [19] in the CPM method.

Precedent 3. Suppose the content consists of words that are 'Indian', 'good', 'always' and 'is' and Table 2.3 shows the codes assigned to these words using some packing technique like Huffman. We have used numbers in the place of bits to better understand the problem. Now suppose a query content T and its compressed content T' is given as follows:

T: 'good Indian'
T': 45 41 23 25 18

We examine that the words 'always' and 'is' do not appear in the content T yet their word-code (25 18) and (41 23 25) are in T', indicating that 'always' and 'is' might match in T', but in reality, they do not appear in T, and when that happens, we conclude that this is an invalid match.

2.4. Wavelet Tree (WT). WT [15], a data structure with space efficiency which represents a series of queries and answers them properly. For representing sequences, rearranging elements, a point grid, and others, a WT can be utilized. We can retrieve any content at any time by retaining the content's index [5]. Authors implemented different WT forms in [4][28] and they completed their work and performed different types of tasks on WTs. The function of WT can be understood by performing the following functions: the number of events (to count the number of a symbol present in the source material), position (in the source material, find the correct position of any symbol), and Show (in the source material, show the symbol's status). All important tasks are implemented with these simple bit-map operations such as rank and select. In [6], we briefly addressed the efficiency of rank and select operations. For any given bitmap sequence M, $rank_c(M, i)$ = the frequency of symbol c up to i^{th} location in M[1..n] and $select_c(M, i)$ = index of the i^{th} case of symbol c in M[1...n]. For e.g. bitmap is M = 1010100101011, then $rank_0(M, 10) = 5$ and $select_1(M, 3) = 5$. The compressibility of the content and its advantages are established in [10][29]. We can also create WTs with compressed content. You will find various letter/word symbols on the WT leaf. Insights are given in [15] on the creation of WT. A new indexing model with the properties of the WT has been developed. The greatest advantage of constructing

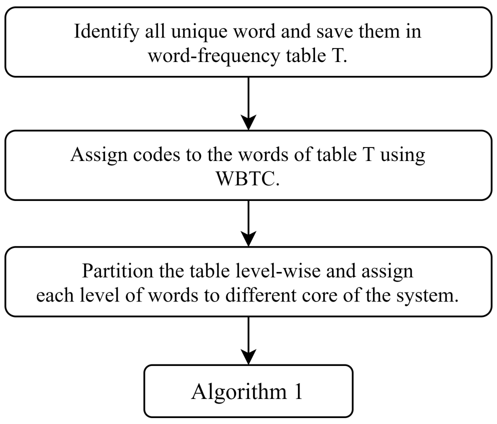

Identify all unique word and save them in word-frequency table T.
Assign codes to the words of table T using WBTC.
Partition the table level-wise and assign each level of words to different core of the system.
Algorithm 1

Fig. 2.1: Pre-processing before Construction of PWT.

indexes using the data structure of the WT is that it is much less complex than other data structures, such as the B tree and the B^+ tree. The downside to WT is that it takes a lot of time to create it. Various studies have recommended several construction paradigms. The WT is constructed sequentially or parallelly, and each has its problems. We choose a parallel WT structure to reduce the development time of WT.

There are bunches of hypothetical work on the parallel construction of the WT. [13] performed the creation of a parallel WT for the first time. The authors suggest the use of a maximum of $log\ \sigma$ processors to construct WT using two linear-time algorithms of $O(n)$ in parallel. The complexities of the solution are $O(n)$ for depth and $O(\sigma\ log\ n)$ for work. In 2015, A new algorithm for the creation of parallel WT has been introduced by [30]. In this work the construction of the WT is done level by level, requiring $O(log\ n)$ time for depth per level and $O(n)$ time for work. If there is σ number of levels in the WT, the complexities of the solution are given as $O(log\ n\ log\ \sigma)$ time for depth and $O(n\ log\ \sigma)$ time for work. [23] recommended a different strategy, which is a more space-efficient algorithm than the work of (Shun, 2015) but achieved the same complexity and limitation. The only difference is that Shun has used a 40-core processor to implement his algorithm, while J. Labeit has used a 64-core processor for his algorithm. An additional improvement is achieved by the recursive implementation of the algorithm rather than done using level by level. In [31], the author used parallel integer sorting methods to improve the work [30] proposed by himself and reduced his work up to $O(\sigma + log\ n)$ for depth and $O(n \left\lceil log\ \sigma/\sqrt{(log\ \sigma)} \right\rceil)$ for work. The fastest sequential and parallel construction of the WT has been introduced in [11]. The text size n has been divided into $\theta(n/p)$ and it has been allocated to each p-core of the system. However, it takes $O(n\ log\ \sigma)$ work and $O(n)$ time to construct the WT parallelly and it also needs $4\sigma\ \lceil log\ n \rceil$ in bits separately for input and output operation.

This segment reflects on how data packing is achieved and what are the issues with compression. The value of using the word as the essential compressive element in place of characters has been shown. Here we see that the word packing of Huffman is fast, and it requires less memory in the number of bits. WBTC packing is also a good method and more efficient than the Huffman methods of packing. Both Huffman and WBTC packing may give false matching results when we use it with the CPM quick-search mechanism. We also see the different WT construction approaches as we use WT to execute our WBTC packing and create the tree with multi-core computer architectures in parallel.

3. The WBTC_PWT Approach. We already understand that the data size is growing nearly every day, so it is very challenging to find any text faster. Therefore we need a data-set size reduction algorithm, which will take up less space for storing data, and will also allow a quick search. For this, we parallelize WT using the multi-core architecture of the computer and we use the WBTC packing technique with the help of WT. The idea is to divide our data-set into several unrelated sets by dividing the word-frequency table into several levels. Each level of words is assigned to a specific core of the multiprocessor computer, and each core forms a level of parallel WT. At the end of the algorithm, we logically add each small WT to create the final WT of the entire content.

Algorithm 1: Fast creation of WT

```
Input: Suppose there is an ‘X’ level in word-frequency tables.
  In each small partition table T,
    Unique-words $\longrightarrow$ $uw_1$, $uw_2$, $uw_3$ ………… $uw_n$.
    Word-codes $\longrightarrow$ $wc_1$, $wc_2$, $wc_3$ ………… $wc_n$.
Result: ‘X’ number of WTs.
Method:
 1: Par-For r $\longleftarrow$ 0 to X-1 do (assign each small table partitions to a core)
 2: For s $\longleftarrow$ 1 to n do (at each core)
 3: $P_{s,r}$ $\longleftarrow$ first prefix pair of word-code $wc_s$;
 4: Insert $A_{s,r}$ in Root($T_r$) of WT;
 5: Present_Node $\longleftarrow$ Root($T_r$);
 6: t $\longleftarrow$ 2;
 7: While (until word-code $wc_s$ is not empty) do
 8: $P_{s,t}$ $\longleftarrow$ next prefix pair bits of $wc_s$ ;
 9: If ($P_{s,t-1}$ == “00” ) then
10: Present_Node $\longleftarrow$ L-child (Present_Node);
11: Elseif ($P_{s,t-1}$ == “11” ) then
12: Present_Node $\longleftarrow$ R-child (Present_Node);
13: Else
14: Insert $P_{s,t}$ into Present_Node;
15: End If
16: t $\longleftarrow$ t+1;
17: End While
18: End For
19: End Par-For
```

Every WT is combined level by level starting from the root node in such a way that all the root nodes of small WTs have to be included in the final WT such as bitmaps of all the root nodes are combined to form the root node’s final bitmap and based on the prefix pair of bits, we then determine left and right subtree of the root of the final WT. This process is repeated for all the other nodes as well. To construct a WT S, the entire corpus is divided by first dividing the word-frequency table level-wise and then assigning each partition to a single core using a parallel for loop. The fastest sequential WT creation method given in [11] is applied to each core, and different WTs are formed for each level of words parallelly. We perform pre-processing on the text corpus, as shown in Fig 2.1, and use Algorithm 1 to build WT faster.

After the completion of Algorithm 1, each small WTs are combined in such a way that roots of all small WTs are combined using their prefix pair code-words to form the root of final WT, next L-child of root for all the small WTs are combined to form the L-child of final WT, and the same process is repeated for the R-child too. This process is followed for all the levels present in the small WT until each node of all small WTs does not merge into the final WT.

Pattern search: Whenever a search request comes for a pattern, then we perform the pre-processing for finding the WBTC code of the query word or pattern. If the word is not found in the word frequency table, we can conclude that it does not appear in the source content T, and there is no need for further processing. Once we find the WBTC code of all the words present in the query pattern from the word-frequency table, we load the final WT into the computer’s primary memory. Now we use Algorithm 2 to find the position of the query pattern in the compressed text corpus.

To better understand the proposed strategy, we consider the same previously used textual content T given as ”a good Indian is always a good Indian for all Indian”. We now perform pre-processing according to the steps shown in Fig 2.1 and apply Algorithm 1. Each unique word that is in T is stored in the word-frequency table according to their frequency and divides the word-frequency table into levels. We now provide the WBTC code

Algorithm 2: Fast Matching

Input: Word-code ‘wc’ of each query words.
Result: The exact position of each word and its occurrence in the packed text corpus.
Method:

```
1: For each unique wc do
2: $B_0$ ⟵ Root (T); (T is the root of Final WT)
3: $P_0$ ⟵ find the first prefix pair from word-code wc;
4: r ⟵ 0;
5: While ($P_r$ != “01” and $P_r$ != “10”) do
6: If ($P_r$ == “00”) then
7: $B_{r+1}$ ⟵ L-child ($B_r$);
8: Else
9: $B_{r+1}$ ⟵ R-child ($B_r$);
10: End-If
11: r ⟵ r+1;
12: $P_r$ ⟵ find the $r^{th}$ prefix pair bits from wc;
13: End While
14: $N_{occ}$ ⟵ $Rank_{P_r}$ (Br, $|Br|$);
15: For k ⟵ 1 to $N_{occ}$ do
16: pos ⟵ $Select_{P_r}$ ($B_r$, k);
17: level ⟵ r;
18: While ($B_{level}$ != Root(T)) do
19: level ⟵ level – 1;
20: $B_{level}$ ⟵ Parent ($B_{level+1}$);
21: If ($B_{level+1}$ == L-child ($B_{level}$)) then
22: $P_{level}$ ⟵ 00;
23: Else
24: $P_{level}$ ⟵ 11;
25: End If
26: pos ⟵ $Select_{P_{level}}$ ($B_{level}$, pos);
27: End While
28: display $k^{th}$ occurrence of the word at pos;
29: End For
```

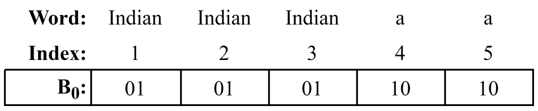
Table 3.1: WBTC codes for all the words level by level

Levels	Words in each level	Indexes	Words	Frequencies	WBTC Codes
Level 1	2^1 words	1	Indian	3	01
		2	a	2	10
Level 2	2^2 words	3	good	2	0001
		4	is	1	0010
		5	for	1	1101
		6	all	1	1110
Level 3	2^3 words	7	always	1	000001

for the words present at all levels, as shown in Table 3.1. The example pattern has three levels, and all three levels are assigned to multi-core systems for constructing three different WTs, as shown in Fig 3.1, Fig 3.2 and Fig 3.3.

All three WTs are combined to form the Final-WT because this Final-WT is used for matching a query pattern directly on the compressed text. After the WBTC packing, the example content T may look like T’ = 10 0001 01 0010 000001 10 0001 01 1101 1110 01, and its Final WT is shown in Fig 3.4.

Matching and searching: Using the WBTC_PWT approach we search if the word ‘always’ is found or not



Word:	Indian	Indian	Indian	a	a
Index:	1	2	3	4	5
B_0:	01	01	01	10	10

Fig. 3.1: WT for Level 1.

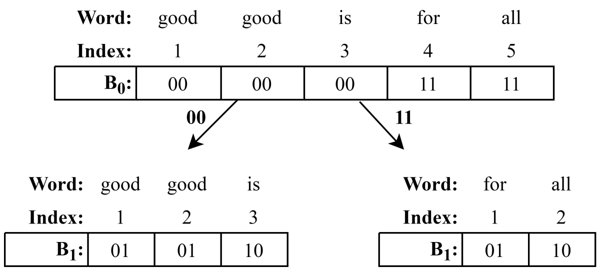


Fig. 3.2: WT for Level 2.

in the textual material T, and if it is found in T then how often and what is its correct position. To search for the word 'always' one must first get its code from the WBTC frequency table and then implement Algorithm 2. The WBTC code of the word 'always' is '000001', so we start searching from the root bitmap B_0 of the final WT. The first prefix pair of bits is '00', so we should move towards the left subtree of the final WT and reach node B_1. The next pair of bits is '00', so we next move towards the left subtree of bitmap B_1 and reach node B_2. The next pair of bits is '01' that is the flag bit, so we search bitmap B_2 for the number of occurrences of the word 'always'. If the code '01' is not found in B_2 then, we say the word 'always' is not present in the compressed content, but since it is found so we perform a rank operation at B_2 to know the frequency of the word 'always' as $Rank_{01}(B_2, |B_2|) = 1$, that implies the word 'always' appears only one time in the content T.

To give the exact position of the word 'always' we perform the select operation and traverse the final WT in reverse order. Since the frequency of occurrence of the word is 1, so there should be only one position where the word 'always' is to be found in T. The select operation is performed to give the exact position of the word 'always'. So, we calculate $Select_{01}(B_2, 1) = 1$. Now we move to the inverse direction towards its parent node and again run the select operation with the outputs of the previous select operation such as $Select_{00}(B_1, 1) = 3$. Now again we move to the inverse direction towards its parent node and again run the select operation with the outputs of the previous select operation such as $Select_{00}(B_0, 3) = 5$. Since B_0 is the root node of the final WT thus, we can correctly say that the query word 'always' is found exactly at position 5 from the starting position in the source content T.

4. Experimental Setup and Results. To do this experiment, the Intel(R) Xeon(R) CPU E3-1245-v3 with 12 Gigabytes of RAM has been selected and all our algorithms have been implemented and demonstrated with Ubuntu 18.04 LTS. The programming language used is C++ with OpenMP, and the GCC compiler is used to construct all the codes. The research is tested on a small, self-made word document and consequences are compared with existing algorithms like Huffman word packing, WT, and WBTC as these algorithms have been used for word-based compressed pattern matching. The steps for different sample words of different lengths have been executed periodically, and we have followed the average time value of the algorithms to be represented in the result. There are two algorithms for our solution, the first is for wavelet building, and the second algorithm is sample word matching. Thus, both the build-time and the matching time are the cumulative time of the methodology. With $O(n \ log \ \sigma)$ time for work and an additional $4\sigma \lceil log \ n \rceil$ bits of space, the construction of parallel WT is achieved in $O(n)$ time. WBTC encoding takes $O(n \ log_2 \ n)$ if the source content includes n number of words. It takes $O(n)$ time to fulfil the word matching request using a WT. Therefore, our solution takes the cumulative of $O(n \ log_2 \ n)$ time to design the index and matching sample query.

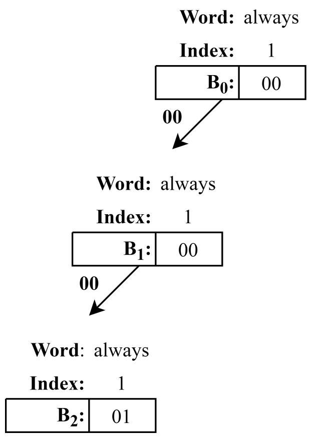


Fig. 3.3: WT for Level 3.

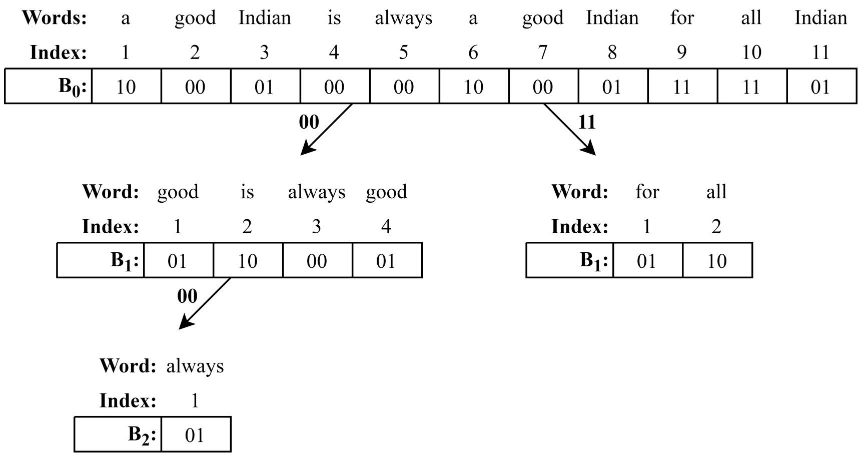

Words: a good Indian is always a good Indian for all Indian
Index: 1 2 3 4 5 6 7 8 9 10 11
B0: 10 00 01 00 00 10 00 01 11 11 01
00
11
Word: good is always good
Index: 1 2 3 4
B1: 01 10 00 01
Word: for all
Index: 1 2
B1: 01 10
00
Word: always
Index: 1
B2: 01

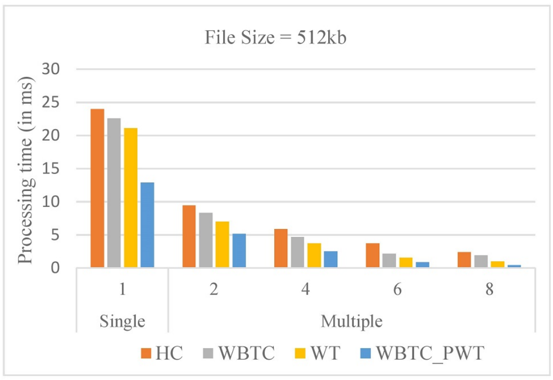
Fig. 3.4: Final WT for the content T.

In our approach, some additional memory is required in parallel construction and in creating the final index by merging all the smaller WTs into the final WT. The indexes created by our algorithm are more space-efficient than the indexes created by previous existing algorithms. The main advantage of using a WT data structure to create indexes is that its space complexity is much lower than other data structures such as B tree and B^+ tree. The other advantage of WT is that each node stores a pair of bits, so this will minimize the overall height of the WT compared to the Huffman tree for words. The rank and select operations of the WT always help to avoid mismatches and advantageously, all these operations are completed in $O(1)$ time. The build time is the only disadvantage of a WT, so we try to reduce it by using a parallel construction approach. Our proposed method matches the words better than other algorithms by comparing the processing time of the proposed method with different prevalent compressed matching algorithms for words like Huffman Word-Coding (HC), WBTC and WT. Table 4.1 and Fig 4.1display the processing time of all algorithms by varying the file size with a fixed alphabet size. Fig 4.1 displays the processing time of HC, WBTC, WT, and WBTC_PWT with a fixed alphabet length of 256. Furthermore, the processing time is also increasing as the file size increases as displayed in Table 4.1.

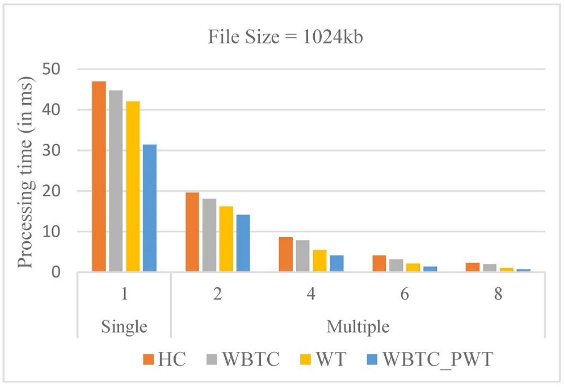
From Table 4.1 we see, for all word patterns and the file size 512 KB, the HC algorithm runs for about 45.353 seconds, the WBTC algorithm runs for about 39.584 seconds, the WT algorithm runs for about 34.320 seconds,

Table 4.1: Processing time (in seconds) of algorithms (for fixed alphabet size = 256)

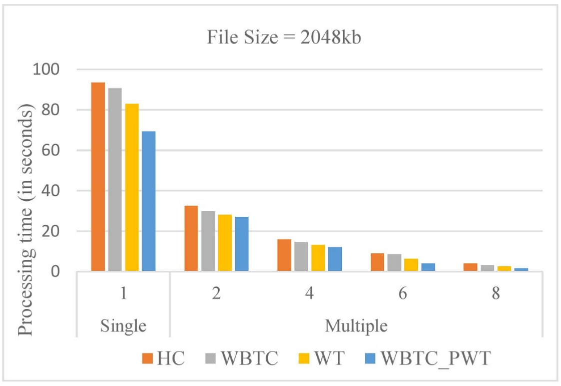
Words in Pattern		File size = 512KB				File size = 1024KB				File size = 2048KB			
		HC	**WBTC**	**WT**	**WBTC_PWT**	**HC**	**WBTC**	**WT**	**WBTC_PWT**	**HC**	**WBTC**	**WT**	**WBTC_PWT**
Single	1	23.965	22.546	21.089	12.876	46.947	44.763	41.984	31.380	93.531	90.735	82.967	69.386
Multiple	2	9.432	8.321	6.978	5.129	19.621	18.057	16.245	14.096	32.607	29.841	28.073	27.007
	4	5.854	4.675	3.723	2.531	8.639	7.847	5.397	4.062	15.832	14.546	13.094	11.998
	6	3.729	2.145	1.541	0.859	4.067	3.185	2.074	1.354	9.045	8.598	6.332	4.098
	8	2.373	1.897	0.989	0.431	2.273	1.945	1.023	0.687	4.023	3.105	2.639	1.576



(a)



(b)



(c)

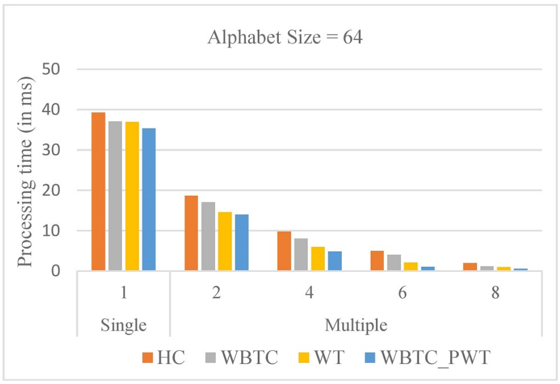
Fig. 4.1: (a), (b) and (c) shows the processing time of algorithms (for fixed alphabet size of 256)

and our suggested WBTC_PWT algorithm runs for about 21.826 seconds. Thus, our proposed algorithm shows an average increase of up to 51%, 44% and 36% relative to the HC algorithm, the WBTC algorithm and the WT algorithm, respectively. According to our estimates, the average improvement of our proposed WBTC_PWT algorithm over HC algorithm, WBTC algorithm and WT algorithm is 36%, 31% and 22% respectively for file size 1024Kb and 26%, 22% and 16% respectively for file size 2048Kb. Fig 4.1 also demonstrate that processing time decreases as the number of words in pattern increases as shown in Table 4.1.

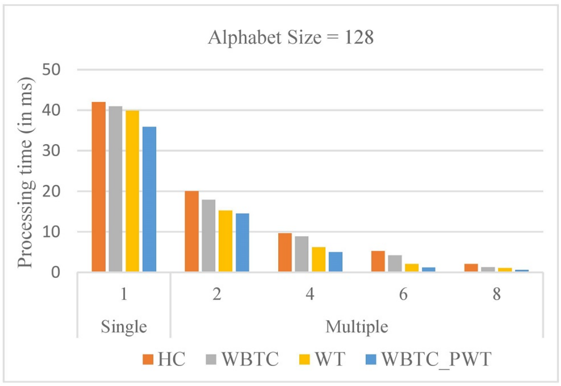
Table 4.2 and Fig 4.2 shows algorithm processing time while alphabet length varies, and file size is constant as 1024KB. Table 4.2 display that the algorithms processing time is decreasing as the alphabet size increases.

Table 4.2: Processing time (in seconds) of algorithms (for fixed file size = 1024KB)

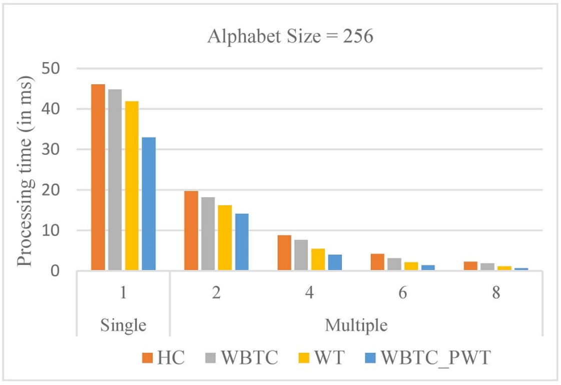
Words in Pattern		**Alphabet size = 64**				**Alphabet size = 128**				**Alphabet size = 256**			
		HC	**WBTC**	**WT**	**WBTC_PWT**	**HC**	**WBTC**	**WT**	**WBTC_PWT**	**HC**	**WBTC**	**WT**	**WBTC_PWT**
Single		39.280	37.084	36.932	35.371	42.042	40.993	39.897	35.902	46.140	44.823	41.871	32.984
Multiple	2	18.635	17.035	14.581	14.005	20.048	17.941	15.261	14.523	19.729	18.173	16.187	14.106
	4	9.734	8.047	5.932	4.842	9.642	8.903	6.186	4.943	8.833	7.683	5.431	4.007
	6	4.981	4.002	2.094	0.989	5.258	4.174	2.068	1.238	4.168	3.186	2.109	1.399
	8	1.994	1.179	0.928	0.599	2.043	1.278	1.048	0.641	2.275	1.890	1.098	0.698



(a)



(b)



(c)

Fig. 4.2: (a), (b) and (c) shows the processing time of algorithms (for fix file-size of 1024KB)

From Table 4.2 we see, for all word patterns and the alphabet size of 64, the HC algorithm runs for about 74.624 seconds, the WBTC algorithm runs for about 67.347 seconds, the WT algorithm runs for about 60.467 seconds, and our suggested WBTC_PWT algorithm runs for about 55.806 seconds. Thus, our proposed algorithm shows an average increase of up to 25%, 17% and 7% relative to the HC algorithm, the WBTC algorithm and the WT algorithm, respectively. According to our estimates, the average improvement of our proposed WBTC_PWT algorithm over HC algorithm, WBTC algorithm and WT algorithm is 27%, 21% and 12% respectively for the alphabet size 128 and 34%, 29% and 20% respectively for the alphabet size 256. Fig 4.2 demonstrate that

Table 4.3: Compression-Ratio of the Approaches

S.No.	Approaches	Compression Ratio
1.	HC	37.93
2.	WBTC	40.06
3.	WT	32.48
4.	WBTC_PWT	30.29

runtime increases as the size of the alphabet increases. The results show differences in processing time when we process a single pattern and multiple patterns in both cases. The proposed approach outperforms the other existing algorithms.

4.1. Compression Ratio. In our proposed approach WBTC_PWT, we need to store the packaged file as well as the word frequency table. This word frequency table is used to store all the unique words in the corpus along with their corresponding WBTC codes. For a large text file, the word frequency table size is also large which can directly impact the compression ratio. As stated by the heap rule [33], a text file of size 'w' in words will have the word frequency table size s $= O(w^{\eta})$ for $0.3 < \eta < 0.8$. Therefore, for a large text file, storing the word frequency table must take the minimum size in memory. For the file size of 1024KB and alphabet size of 256, Table 4.3 shows the compression ratio of the approaches discussed here and clearly shows that on average WBTC_PWT takes a lower space than the other approaches.

5. Conclusion and Future work. This paper presents a method WBTC_PWT, for exact text matching in a compressed text corpus. We also tried to minimize the size of the data by performing data packing, so less memory is needed in the processing of data. This paper presents an improvement in matching time compared to other algorithms. As we know, whenever the size of the indexes is larger than the size of the main memory and a search request arrives for any query, these indexes must be loaded into the main memory. Since the size of the indexes is larger than the size of the main memory, it will cause a high number of page faults and thus affect the overall system throughput. We used the WBTC packing technique to minimize the size of the data at the initial stage and then create the indexes of the data with the help of the WT. The main advantage of creating indexes using a WT is that it takes up less memory than other indexing methods and it is free from getting a false match. The main drawback of the WT is its construction time. So, to minimize the construction time of the WT, we used parallel processing using the computer's multi-core architecture. Furthermore, we minimize the matching and search time by matching the query text directly to the compressed text content, without the need for decompression. This approach improves the overall throughput of the system.

In the future, we will attempt to create a WT using machine learning, by using our algorithm with various size of text datasets. We can also use computer classification models to find and measure the frequency of specific terms to allocate WBTC codes to all words. Using machine learning we can reduce the additional memory needs in constructing WT in the WBTC_PWT algorithm.

REFERENCES

[1] Amihood Amir, Gary Benson, and Martin Farach. Let sleeping files lie: Pattern matching in z-compressed files. *Journal of Computer and System Sciences*, 52(2):299–307, 1996.

[2] Richard Beal and Donald Adjeroh. Compressed parameterized pattern matching. *Theoretical Computer Science*, 609:129–142, 2016.

[3] Robert S Boyer and J Strother Moore. A fast string searching algorithm. *Communications of the ACM*, 20(10):762–772, 1977.

[4] Nieves R Brisaboa, Yolanda Cillero, Antonio Farina, Susana Ladra, and Oscar Pedreira. A new approach for document indexing usingwavelet trees. In *18th International Workshop on Database and Expert Systems Applications (DEXA 2007)*, pages 69–73. IEEE, 2007.

[5] Nieves R Brisaboa, Antonio Farina, Susana Ladra, and Gonzalo Navarro. Implicit indexing of natural language text by reorganizing bytecodes. *Information Retrieval*, 15(6):527–557, 2012.

[6] Francisco Claude and Gonzalo Navarro. Practical rank/select queries over arbitrary sequences. In *International Symposium on String Processing and Information Retrieval*, pages 176–187. Springer, 2008.

[7] Edleno Silva De Moura, Gonzalo Navarro, Nivio Ziviani, and Ricardo Baeza-Yates. Direct pattern matching on compressed text. In *Proceedings. String Processing and Information Retrieval: A South American Symposium (Cat. No. 98EX207)*, pages 90–95. IEEE, 1998.

[8] Edleno Silva De Moura, Gonzalo Navarro, Nivio Ziviani, and Ricardo Baeza-Yates. Fast searching on compressed text allowing errors. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 298–306, 1998.

[9] Martin Farach and Mikkel Thorup. String matching in lempel—ziv compressed strings. *Algorithmica*, 20(4):388–404, 1998.

[10] Paolo Ferragina, Giovanni Manzini, Veli Mäkinen, and Gonzalo Navarro. Compressed representations of sequences and full-text indexes. *ACM Transactions on Algorithms (TALG)*, 3(2):20–es, 2007.

[11] Johannes Fischer, Florian Kurpicz, and Marvin Löbel. Simple, fast and lightweight parallel wavelet tree construction. In *2018 Proceedings of the Twentieth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 9–20. SIAM, 2018.

[12] Kimmo Fredriksson and Maxim Mozgovoy. Efficient parameterized string matching. *Information Processing Letters*, 100(3):91–96, 2006.

[13] José Fuentes-Sepúlveda, Erick Elejalde, Leo Ferres, and Diego Seco. Efficient wavelet tree construction and querying for multicore architectures. In *International Symposium on Experimental Algorithms*, pages 150–161. Springer, 2014.

[14] Lars Gleim and Stefan Decker. Open challenges for the management and preservation of evolving data on the web. *MEPDaW@ ISWC*, 2020.

[15] Roberto Grossi, Ankur Gupta, and Jeffrey Scott Vitter. High-order entropy-compressed text indexes. 2003.

[16] A Gupta and S Agarwal. A scheme that facilitates searching and partial decompression of textual documents. *Int. J. Adv. Comput. Eng*, 1(2), 2008.

[17] Ashutosh Gupta and Suneeta Agarwal. A fast dynamic compression scheme for natural language texts. *Computers & Mathematics with Applications*, 60(12):3139–3151, 2010.

[18] Rahul Gupta, Ashutosh Gupta, and Suneeta Agarwal. A novel approach of data compression for dynamic data. In *2008 IEEE International Conference on System of Systems Engineering*, pages 1–6. IEEE, 2008.

[19] David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[20] Radhika Khetan, Suneeta Agarwal, and Rajesh Prasad. An efficient approach towards compressed parameterized word matching using wavelet tree. *Journal of Information and Optimization Sciences*, 37(2):285–301, 2016.

[21] Shmuel T Klein and Dana Shapira. Compressed matching in dictionaries. *Algorithms*, 4(1):61–74, 2011.

[22] Donald E Knuth, James H Morris, Jr, and Vaughan R Pratt. Fast pattern matching in strings. *SIAM journal on computing*, 6(2):323–350, 1977.

[23] Julian Labeit, Julian Shun, and Guy E Blelloch. Parallel lightweight wavelet tree, suffix array and fm-index construction. *Journal of Discrete Algorithms*, 43:2–17, 2017.

[24] Yury Lifshits. Processing compressed texts: A tractability border. In *Annual Symposium on Combinatorial Pattern Matching*, pages 228–240. Springer, 2007.

[25] Christos Makris. Wavelet trees: A survey. *Computer Science and Information Systems*, 9(2):585–625, 2012.

[26] Surya Prakash Mishra, Rajesh Prasad, and Gurmit Singh. Fast pattern matching in compressed text using wavelet tree. *IETE Journal of Research*, 64(1):87–99, 2018.

[27] Surya Prakash Mishra, Col Gurmit Singh, and Rajesh Prasad. A review on compressed pattern matching. *Perspectives in Science*, 8:727–729, 2016.

[28] Gonzalo Navarro. Wavelet trees for all. *Journal of Discrete Algorithms*, 25:2–20, 2014.

[29] Gonzalo Navarro and Veli Mäkinen. Compressed full-text indexes. *ACM Computing Surveys (CSUR)*, 39(1):2–es, 2007.

[30] Julian Shun. Parallel wavelet tree construction. In *2015 Data Compression Conference*, pages 63–72. IEEE, 2015.

[31] Julian Shun. Improved parallel construction of wavelet trees and rank/select structures. *Information and Computation*, 273:104516, 2020.

[32] Edleno Silva de Moura, Gonzalo Navarro, Nivio Ziviani, and Ricardo Baeza-Yates. Fast and flexible word searching on compressed text. *ACM Transactions on Information Systems (TOIS)*, 18(2):113–139, 2000.

[33] Dick C van Leijenhorst and Th P Van der Weide. A formal derivation of heaps' law. *Information Sciences*, 170(2-4):263–272, 2005.

[34] Arun Kumar Yadav, Divakar Yadav, and Rajesh Prasad. Efficient textual web retrieval using wavelet tree. *International Journal of Information Retrieval Research (IJIRR)*, 6(4):16–29, 2016.

[35] Divakar Yadav, Apeksha Singh, and Vinita Jain. Search results optimization. In *International Conference on Contemporary Computing*, pages 325–334. Springer, 2011.

[36] Nivio Ziviani, E Silva De Moura, Gonzalo Navarro, and Ricardo Baeza-Yates. Compression: A key for next-generation text retrieval systems. *Computer*, 33(11):37–44, 2000.

Edited by: Dana Petcu
Received: Apr 26, 2021
Accepted: Nov 16, 2021

Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2021 SCPE. Volume 22, Issues 4, pp. 401–411, DOI 10.12694/scpe.v22i4.1945

PERFORMANCE-EFFICIENT RECOMMENDATION AND PREDICTION SERVICE FOR BIG DATA FRAMEWORKS FOCUSING ON DATA COMPRESSION AND IN-MEMORY DATA STORAGE INDICATORS

HRACHYA ASTSATRYAN[*], ARTHUR LALAYAN[†], ARAM KOCHARYAN[‡], AND DANIEL HAGIMONT[§]

Abstract. The MapReduce framework manages Big Data sets by splitting the large datasets into a set of distributed blocks and processes them in parallel. Data compression and in-memory file systems are widely used methods in Big Data processing to reduce resource-intensive I/O operations and improve I/O rate correspondingly. The article presents a performance-efficient modular and configurable decision-making robust service relying on data compression and in-memory data storage indicators. The service consists of Recommendation and Prediction modules, predicts the execution time of a given job based on metrics, and recommends the best configuration parameters to improve Hadoop and Spark frameworks' performance. Several CPU and data-intensive applications and micro-benchmarks have been evaluated to improve the performance, including Log Analyzer, WordCount, and K-Means.

Key words: Hadoop; Spark; MapReduce; data compression; in-memory file system

AMS subject classifications. 68T09

1. Introduction. Big Data has become a critical research area due to the massive data generation [1]. Managing such data is a resource-intensive operation demanding parallel processing. The rapid development of Big Data frameworks addresses the distribution, communication, and processing of a vast number of data. For instance, Hadoop and Spark popular frameworks [2] may handle massive amounts of data relying on the MapReduce paradigm [3] to process and generate extensive data sets [4]. The data sets are stored across distributed clusters to run a distributed processing scheme in each cluster. The Hadoop Distributed File System (HDFS) [5] is a storage layer and a back-end file system for the MapReduce model, providing a scalable, fault-tolerant, and portable architecture for MapReduce jobs.

The MapReduce framework splits large data sets into a set of distributed blocks, executes Map tasks in parallel on these blocks, and finally reduces tasks for the aggregation of results to process the data. HDFS performance mainly depends on data access and data movement, where memory data access bandwidth is higher than the disk access bandwidth. Therefore, an influential organization of an intensive I/O application to disk is a challenge. Data compression is a critical approach to reduce I/O operations, whereas the in-memory file systems using HDFS lazy persist strategy [6] improves the I/O rate. From one side, data compression reduces the input data size, HDFS storage usage, and network traffic. Map and Reduce tasks are executed on different machines in a parallel fashion in the MapReduce programming model. The final result is achieved only after the completion of the Reduce tasks. The compression at various stages of MapReduce jobs or minimization of mapper output saves the disk space and minimizes spilling. From another side, HDFS Data nodes flush in-memory data to disk asynchronously to remove checksum computations and expensive disk I/O from the performance-sensitive I/O path. HDFS offers best-effort persistence guarantees for Lazy Persist Writes, where data are written in a faster off-heap memory located in a RAM than writing on a hard drive. Therefore, it may decrease the waiting time for the written data to the disk and improve I/O rate.

[*]Institute for Informatics and Automation Problems National Academy of Sciences of Armenia 1, Paruyr Sevak str. 0014 Yerevan, Armenia (hrach@sci.am).
[†]National Polytechnic University of Armenia 105, Teryan str. 0009 Yerevan, Armenia (arthurlalayan97@gmail.com).
[‡]Université Fédérale Toulouse Midi-Pyrénées Toulouse Cedex 7, 31000 Toulouse, France (ar.kocharyan@gmail.com).
[§]Université Fédérale Toulouse Midi-Pyrénées Toulouse Cedex 7, 31000 Toulouse, France (daniel.hagimont@irit.fr).

Hadoop and Spark frameworks support different data compression methods to reduce I/O and improve performance. The methods require configuring the frameworks to recognize compression codecs by specifying the codec implementation paths. Then to compress the input data and save it to HDFS. Before processing data, the framework decompresses the data if the input file is compressed in HDFS. Our recent studies show [7, 8, 9] that the average memory usage for selected scientific workflows is 13-17% for Hadoop and 20-40% for Spark jobs, which neglect the full utilization of the RAM of HDFS nodes. Therefore, the usage of RAM-free space may boost the performance of HDFS processing. Our primary approach is to benefit from combining RAM and disk space by setting up HDFS on top of the unused space by implementing virtual RAMDisks in all data nodes. That approach allows saving the input data in RAM before processing it. The article presents a codeless performance-efficient decision-making robust service relying on data compression techniques and in-memory file systems. The service, which consists of Recommendation and Prediction modules, has been evaluated for several MapReduce workflows.

The content of this paper is organized as follows. The related work is presented in Section 2. The Recommendation and Prediction service can be found in Section 3. The evaluation results are given in Section 4. Finally, the conclusion and directives for future research are drawn in Section 5.

2. Related work. The performance efficiency, memory caching techniques, and in-memory computing models are widely used to improve the performance of Hadoop and Spark frameworks. The Intra-Node Combiner approach is proposed by [10, 11] using in-node and in-mapper combiners that rely on memory caching technologies. This approach changes the MapReduce job mapping methods in the Map and Reduce phases using in-node and in-map design patterns to combine the data of a node in the Map phase and then send it to the Reduce phase. The in-memory Redis data cache improves job execution time by 23% over the standard approach. Notably, the in-map combiner reduces execution time by 25%, and the in-node combiner reduces execution time by 20%. Besides, the in-memory cache observed that the average Map completion time is reduced by 14.8%. The combined design pattern improves performance but depends on the codebase memory caching techniques like Redis and doesn't provide a codeless performance efficiency. Therefore, it is necessary to redesign the job in such an approach, read input from the cache, and develop mechanisms for managing the memory cache to get a fully customized environment.

The impact of the usage of high throughput, high bandwidth, and low latency networks over RDMA is studied by to reduce I/O and improve performance focusing on the InfiniBand network [12, 13]. The studies show that the RDMA-based approach enhances the performance of some MapReduce workflows (RandomWriter, PageRank, Sort, TeraSort, TestDFSIO, etc.) by 1.2-1.8 times. It was observed 42%-49% and 46%-50% latency reduction compared to default Hadoop RPC over 10GigE and IPoIB QDR (32 Gbps). Besides, a 10% performance improvement for CloudBurst application and 26% cache improvement of a put operation for HBase. The studies show that RDMA Hadoop reduces I/O to affect the performance optimizations using several compression techniques with an overhead to the uncompressed process. The essential advantage of this approach is the performance-boosting of using network interface cards, which may have high throughput limitations. The network card supports the shuffle phase to exchange intermediate outputs from the Map phase over the network.

Another perspective is a distributed multi-tier caching system on top of HDFS. The experiments on such a multi-tier caching system show that for a 10MB file, the new cache was 56% faster than Redis, and 26- and 60-times fasters for GET and SET operations [14]. Performance and power analysis models estimate the performance per watt for various cloud benchmarks with an error of less than 10% based on comparative studies of performance, the power consumption of multiple clusters with heterogeneous processor configurations (with and without cache memory). Their focus is on performance studies using heterogeneous processor architectures instead of developing or integrating new caching techniques.

RAMCloud storage system is suggested by [15] to integrate the available memory resources in an HDFS cluster to form a cloud storage system. The authors [16] suggest a memory-based Hadoop distributed file system to improve I/O rate because disks have poor performance. In addition to the new memory cache, an in-memory data replacement strategy is proposed to allocate memory space. Such an approach cannot consider compression techniques reducing I/O.

Instead of performance-boosting, the execution time of different scientific workflows in Hadoop or Spark

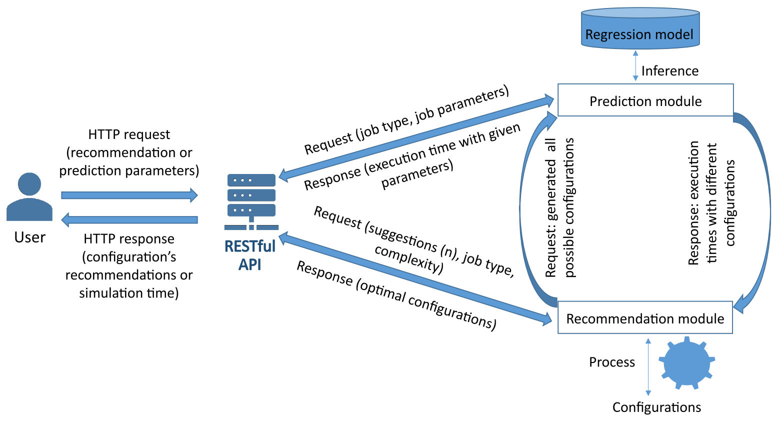

Regression model
Inference
Prediction module
HTTP request (recommendation or prediction parameters)
Request (job type, job parameters)
Response (execution time with given parameters)
User
HTTP response (configuration's recommendations or simulation time)
RESTful API
Request (suggestions (n), job type, complexity)
Response (optimal configurations)
Request: generated all possible configurations
Response: execution times with different configurations
Recommendation module
Process
Configurations

Fig. 3.1: The structure of the performance-efficient Recommendation and Prediction service

is studied by [17, 18]. The authors suggest a weighting system to estimate job execution times in Hadoop using the historical simulation data or machine learning regression models to predict the execution time of various machine learning jobs and SQL queries in Spark. Our studies focus on input data compression and saving in-memory approaches for both Hadoop and Spark frameworks. Besides the Prediction module, a Recommendation module is suggested to advise users to get optimal parameters to improve the performance. The article presents a codeless performance-efficient decision-making system mixing the approaches mentioned above to enhance our previous studies [8].

3. Recommendation and Prediction service for Big Data frameworks. The suggested Recommendation and Prediction service for Big Data frameworks (see fig. 3.1) consists of the following three modules[1]:

- RESTful (Representational State Transfer) - to provide an interface to the MapReduce service [19];
- Recommendation - to recommend a configuration parameters according to the user request;
- Prediction - to analyze and predict suitable configurations satisfying the user requests.

3.1. RESTful module. The RESTful module fully utilizes the architectural principles of REST API to develop a modular and configurable service delivering two different endpoints for the suggested Prediction and Recommendation modules. The Prediction module forecasts the simulation time considering various configuration parameters in regression models. In comparison, the Recommendation module depends on configuration files and the Prediction module to recommend optimal configuration parameters to improve performance. The service modules are abstracted from the system, enabling the implementation of a custom module (using static values, linear or non-linear ML models) and loading the module to the system. The user's requests to the REST API are addressed and transformed via HTTP protocol. The Post method executes the query and fetches the suitable configuration. The JSON (JavaScript Object Notation) lightweight, text-based, language-independent data interchange format empowered the request and response bodies based on simple key/value pairs. The input parameters of the service are the JSON attributes of the request body (see Table 3.1).

The specific metrics are identified per each application or micro-benchmark having its local dataset on the top of the general metrics. The replication and block sizes are specified as static metrics, considering the direct impacts on all the collected metrics. TestDFSIO has an additional option parameter (with values for reading and writing). At the same time, TeraSort consists of three different tasks (TeraGen, TeraSort, and TeraValidate), and K-Means has the number of clusters to find parameters. The input parameters are different

[1]https://github.com/ArmHPC/MapReduce-Optimization-Solutions

Table 3.1: Service input parameters

Attribute name	**Value**
Environment	Hadoop or Spark
Compression method	Uncompressed, gzip, bzip2, lzo, lz4, snappy, zstandard
Data nodes count	number
Complexity	Input data size
Additional	Application specific parameters

Table 3.2: List of metrics

Metric name	**Value**
Environment name	Hadoop or Spark
RAM over HDFS	True/False
Number of nodes	Number
Application	Name
Input Data Size	Number
Input Data Type	Method
Execution time	Value in seconds
CPU	%
Memory	%
I/O Read	Number
I/O Write	Number
Network received	Number
Network sent	Number

for Prediction and Recommendation modules since the Prediction module requires all the described parameters. Still, the Recommendation module requires only job type and complexity (other parameters are optional).

The following types of workflows have been studied:

- data/CPU intensive - WordCount, Log Analyzer;
- ML - K-Means;
- micro-benchmarks - TestDFSIO, TeraSort.

The WordCount is a simple data and CPU-intensive benchmark to count the number of occurrences of each word in a given input set. At the same time, the Log Analyzer is a generic log analyzer benchmark using MapReduce framework [20]. The K-Means focuses on the MapReduce implementation of standard K-means, as clustering algorithms require scalable solutions to manage large datasets [21]. TestDFSIO micro-benchmark is for stressing HDFS with the options of reading and writing, whereas the TeraSort is for generating, sorting, and validating data.

3.2. Prediction module. As the input metrics of each MapReduce application are dissimilar, the Prediction module relies on regression models to predict the job execution time. Different linear and polynomial regression models were trained using key features from the simulation dataset (see Table 3.2).

The regression model requires encoded inputs as there are qualitative variables such as environment or compression method. The qualitative data of the simulation dataset is encoded using the one-hot encoding schema, and the training process is performed on the encoded input data. The REST call encodes the parameters specified by the user based on identical encoders to pass the encoded data to the regression model. The encoders of each MapReduce application are unique as the models, and input parameters are different. As soon as the Prediction module receives the request, the module finds a regression model using the job type and returns the model output for a given input. All the trained models and encoders are accessible through the .pkl format.

The service abstraction easily extends the system for new MapReduce workloads. The trained regression model, the encoder, and the configuration file are provided per each new application. The configuration file corresponds to the trained regression model and contains information about the trained regression model features [22]. As a result of the training series, the polynomial regression models with degree 3 are provided for the studied applications with the following loss function:

$$L = (\ln y - X\beta)^T(\ln y - X\beta) + \lambda\beta^T\beta \tag{3.1}$$

The β weights are determined during the training phase by minimizing the loss function using the gradient descent method. While X is the dataset consisting of different features and all polynomial combinations with degrees less than or equal to 3, y is execution times with given X features, λ is the regularization parameter, I is an identical matrix. The degree of polynomial and regularization hyperparameters are determined using the cross-validation technique. After the training, the Prediction is made by the following formula:

$$\hat{y} = e^{X\beta} \tag{3.2}$$

where $\hat{y}$ is the predicted execution time.

3.3. Recommendation module. The Recommendation module aims to provide an optimal recommendation according to the user request relying on the Prediction module. The module generates all possible configurations and sends the configurations to the Prediction module to get the execution times of the given possible configurations. Finally, the Recommendation module sorts the results according to the execution time and returns top-n configuration parameters, the first of which is the optimal configuration. Before using the Recommendation module, all configurations per application are uploaded to indicate a set of significant metrics, such as the input type, size, or environment. The configuration files are described in JSON format.

Moreover, the configuration must not be loaded every time, as the system caches every successfully loaded configuration. When the user first requests the Recommendation module, it packs a set of configurations using the application name and stores them in the cache. The metrics of the configuration files depend on the key features from the simulation datasets (see Table 2) and correspond to the features of the trained regression models.

The user may specify some fixed metrics. In that case, the Recommendation module will not generate possible values for that metric. For instance, if the user is interested in the Spark environment, the module will not consider the Hadoop environment. The maximum number of configurations is equal to:

$$N_{conf} = N_{env} \cdot N_{datatype} \cdot N_{nodes} \tag{3.3}$$

where N_{env} is the number of environments, $N_{datatype}$ is the number of input data types, and N_{nodes} is the number of nodes in the cluster.

In the current version of the module, the number of possible configurations is equal to 48 per application, as two environments (Hadoop and Spark), eighth input data types (uncompressed, compressed with 7 possible compression methods and using the RAM over HDFS approach), and three cluster configurations (4, 8, and 16 nodes) are used.

4. Evaluation. Various workloads as input data have been evaluated based on diverse data node configurations and compression methods using CPU, memory, I/O, network, and job execution time metrics. Computational and storage resources of the IaaS (Infrastructure as a Service) cloud service of the Armenian hybrid research computing platform have been used for the experiments [23, 24]. The hardware and software configurations of the experimental environment is described in Table 4.1.

With the RAM over HDFS approach, 2GB of RAM was allocated from each data node for HDFS. Different input data sizes, compression methods (gzip, bzip, lzo, snappy, lz4, and zstandart), RAM over HDFS approach, data nodes (up to 16), and additional application-specific parameters have been evaluated for selected MapReduce applications. The outputs of the experiments have been used to train the regression models. 20% of the collected data was selected as a test dataset and the remaining 80% as a training dataset. The k-Fold Cross-Validation was used to find the optimal regression model's hyperparameters for all types of applications [25].

Table 4.2 shows the experimental results of R^2 and RMSE per job, where R^2 score and root mean square error evaluate the regression models linked with the service. R^2 measures the percentage of variation, while the

Table 4.1: List of the software and hardware configurations

Configuration name	**Description**
Operating system	Ubuntu 18.04
Master nodes count	1
Data nodes count	16
RAM	4 GB per node
Hard disk	120 GB SATA per node
Hadoop version	3.2.1
Spark version	2.4.5
Java JDK version	1.8
HDFS block size	128 MB
HDFS block replication	2

Table 4.2: Prediction errors of the regression model

Workflow	**Train**		**Test**	
	R^2	**RMSE**	R^2	**RMSE**
Log Analyzer	0.99	11.5	0.96	25.3
WordCount	0.99	65.2	0.99	158.8
K-Means	0.97	71.4	0.96	87.4
TestDFSIO	0.96	13.6	0.92	22.6
TeraSort	0.98	48.4	0.94	58.2

RMSE measures of the size of the error in the regression model. For each type of application, one polynomial regression model was trained, which at the input receives the necessary metrics described in Table 3.1. The output gives the execution time of the application with the given configuration parameters.

The best R^2 score on the train and test datasets has the regression model of the WordCount application. In contrast, the lowest RMSE on the training dataset has the regression model of the LogAnalyzer application and the test dataset TestDFSIO's regression model. The highest RMSE on the training dataset has K-means's model and on the test dataset WordCount's model. WordCount is expected to have a stable execution time as it is a CPU-intensive application, but the RMSE of the regression model is highest on the test dataset. The reason is that the execution times of WordCount in different configuration parameters are pretty diverse (e.g. the difference between the execution time of WordCount if the input data is compressed with gzip and is uncompressed is high compared with the other applications with the same scenario). This means that the WordCount application execution times with different configurations have higher variance than the other applications.

4.1. Applications. The experiments aim to evaluate the boosting of real applications by considering compression technique to reduce I/O and RAM over the HDFS approach to improve I/O rate compered to the default approach. The optimal compression method provides minimal execution time compared to the other compression methods. The Recommendation module offers the optimal performance considering compression or RAM over HDFS approaches. The evaluation results are given on the figures for 16 data node configurations.

4.1.1. LogAnalyzer. The real performance boosts of using compression methods, RAM over HDFS approach and the predicted performance boost of LogAnalyzer job are shown in Fig. 4.1.

The best performance boosts for 4GB, 8GB, and 16GB are correspondingly 2%, 7.5%, and 3% for Hadoop and 49%, 53% and 83% for Spark. Fig. 4.1 shows that the best performance boost for 4 GB and 16 GB inputs in the Hadoop environment gives compression techniques, while for 8 GB RAM over HDFS approach. Besides, the predicted performance boost compared with real ones has less than 3% of relative error. For the Spark environment, the scenario is a bit different, as the best actual performance boosts compression techniques for

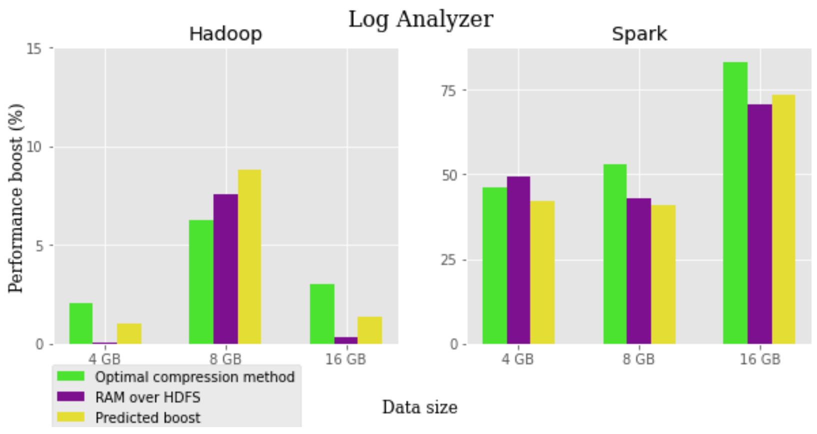


Fig. 4.1: Log analyzer performance boosts for 4GB, 8GB, and 16GB data on Hadoop and Spark

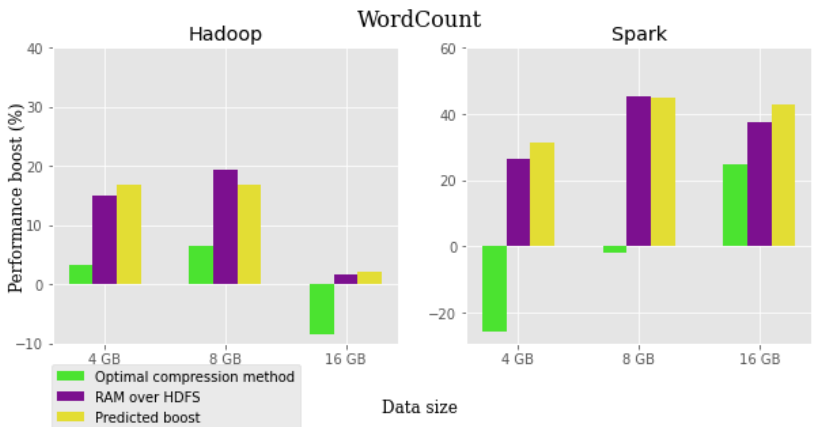


Fig. 4.2: Wordcount performance boosts for 4GB, 8GB, and 16GB data on Hadoop and Spark

8 GB and 16 GB inputs, while for 4 GB input data RAM over HDFS approach gives more performance boost. The predicted performance boost for Spark environment is also close to real ones and has less than 10%.

4.1.2. WordCount. Compared with LogAnalyzer, the real and predicted performance boosts of a CPU and data-intensive WordCount application are shown in Fig. 4.2.

The best performance boosts for 4GB, 8GB and 16GB are correspondingly 15%, 19% and 1.5% for Hadoop and 26%, 45.5% and 37.5% for Spark. As the figure shows, compression methods in the Hadoop environment give performance boost only in 4 GB and 8 GB of input data. When the input data size is 16 GB, compression methods add an average of 10% to the job execution time. RAM over HDFS approach for Hadoop environment compared with compression techniques shows more accurate performance boosts and in three sizes of input data gives performance boost. When the input data is 4 GB and 8GB, the performance boosts are big (15-20%), but for 16 GB of input data, the performance boost is approximately 2%. In this case, the predicted performance boost is also closed to real ones and has less than 3.5% relative error than real performance boosts. The Spark environment is quite different from the Hadoop, as Spark delivers fast performance, while Hadoop has an advantage of linear processing. For 4 GB and 8GB of input data, compression techniques don't give a performance boost, while 16GB of input compression improves the performance. As in Hadoop, the RAM over HDFS approach gives more performance boost against compression techniques in a Spark environment. In this

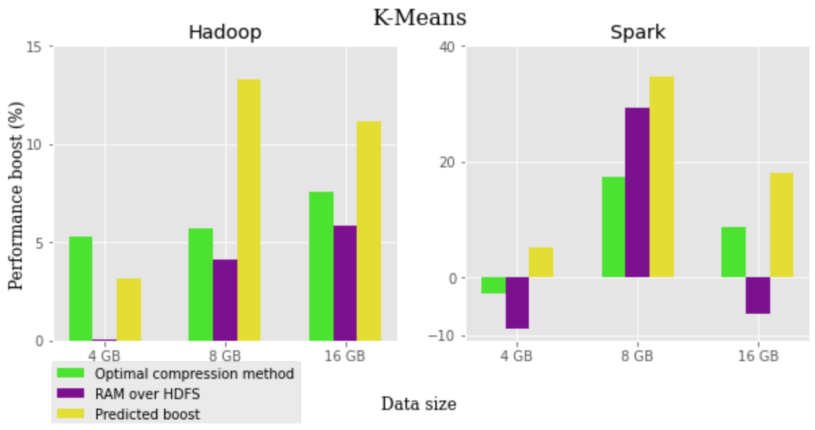


Fig. 4.3: K-Means performance boosts for 4GB, 8GB, and 16GB data on Hadoop and Spark

case, the predicted performance boost is also close to real ones and has less than 6% relative error compared with the best real performance boost.

4.1.3. K-Means. The following real application is K-Means, in which performance boosts are shown in Fig. 4.3.

The best performance boosts for 4GB, 8GB, and 16GB are correspondingly 5.3%, 5.7%, and 7.6% for Hadoop and Spark if the input data is 4 GB, there is not positive performance boost. In contrast, for other sizes, the best performance boost is correspondingly 29.3% and 8.7% for Spark. The figure shows that both approaches improve the performance in the Hadoop environment. Still, compression techniques for the K-Means job help reach a better performance boost than RAM over the HDFS approach. In the Spark environment, compression techniques help get a performance boost for 8GB and 16GB input data, while RAM over the HDFS approach improves performance only for 8GB input. Compared with real ones for Hadoop, the predicted performance boost has approximately less than 7.3% and for Spark less than 10%.

Considering 4GB, 8GB, and 16GB are respectively light, medium, and heavy workload, the average CPU and memory usages across all nodes for real-world applications is shown in a RAM over HDFS approach and using the optimal compression method in Table 4.3.

For all types of real applications, the performance boost of Spark is significantly better than Hadoop because Spark utilizes more CPU and RAM than Hadoop.

4.2. Micro benchmarks. Real application examples show that using compression techniques or the RAM over HDFS approach can achieve different performance gains for various applications. The scenario for micro-benchmarks compared with real applications is the opposite. On the one hand, using compression for micro-benchmarks is inconvenient because the generated TestDFSIO or TeraGen data has many repetitions, and compression methods may show fake and colossal performance boosts. On the other hand, some micro-benchmarks, such as the TeraSort, do not support compression. Therefore, for micro-benchmarks, the performances is compared without considering compression techniques.

4.2.1. TestDFSIO. The execution time of the TestDFSIO application for reading and writing options compared to RAM over the HDFS approach is shown in Fig. 4.4.

The figure shows that not always RAM over HDFS helps to improve performance. TestDFSIO with the read option is faster in the original case, while the write option is faster in the RAM over HDFS approach. Hence, the RAM is much faster than the hard disk; the simulation time boost is considerable for write-option, as it depends on faster RAM than the hard disk.

In the Hadoop environment original approach for 4GB, 8GB, and 16GB data sizes for the read option is correspondingly 1.25, 2.5, and 2.3 times faster than RAM over HDFS approach, while the write option is

Table 4.3: Average CPU and memory usage across cluster nodes

Workflow				**LogAnalyzer**		**WordCount**		**K-Means**	
Framework				Hadoop	Spark	Hadoop	Spark	Hadoop	Spark
Complexity	**Light**	Original approach	CPU	6.47	24.54	5.78	47.94	7.16	39.89
			RAM	15.82	16.47	13.24	34.39	18.09	29.88
		Best compression	CPU	6.51	35.48	6.11	36.55	7.05	31.82
			RAM	16.51	18.77	16.57	40.47	18.14	26.69
		RAM over HDFS	CPU	6.87	55.49	6.87	55.49	6.97	40.23
			RAM	23.31	29.83	23.31	29.83	18.07	29.64
	Medium	Original approach	CPU	5.81	14.70	5.28	21.58	6.46	50.46
			RAM	14.40	17.16	15.92	37.91	16.54	30.49
		Best compression	CPU	5.87	53.42	5.57	41.21	6.32	43.68
			RAM	1.39	22.12	16.45	66.99	16.83	29.64
		RAM over HDFS	CPU	6.97	61.27	6.97	61.27	7.01	70.14
			RAM	29.44	29.49	29.44	29.49	18.58	31.83
	Heavy	Original approach	CPU	6.37	15.53	6.72	37.61	6.69	37.65
			RAM	13.99	23.56	17.36	42.69	18.92	44.37
		Best compression	CPU	6.39	71.19	6.29	51.53	6.68	44.39
			RAM	16.23	20.44	16.55	67.69	17.16	35.62
		RAM over HDFS	CPU	6.92	49.17	6.92	49.17	7.36	58.02
			RAM	41.14	47.45	41.14	47.45	23.37	43.71

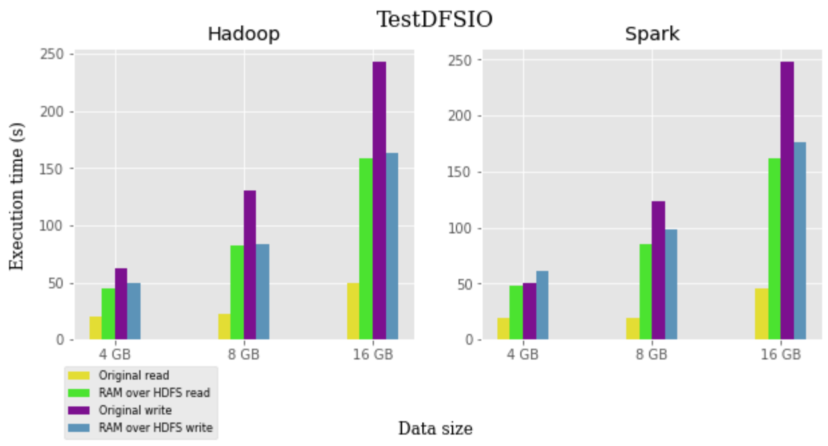


Fig. 4.4: Evaluation of TestDFSIO for read and write options compared to RAM over HDFS approach

correspondingly 1.21, 1.35, and 1.33 times faster than the original approach. In the Spark environment, the scenario is similar. The only difference is that the 4 GB data write option is also 1.2 times faster than the original approach. For 8GB and 16 GB input data for the read option original approach is 3.47 and 2.52 times faster than RAM over HDFS, while for the write option RAM over HDFS approach shows correspondingly 1.2 and 1.29 times fast processing time.

4.2.2. TeraSort. The results of the TeraSort micro-benchmark, which is similar to TestDFSIO, show that in most cases, the RAM over HDFS improves the performance (TestDFSIO with write option and TeraGen job of the TeraSort package). In contrast, the scenario is different in some cases (TestDFSIO read option and the remaining two jobs of the TeraSort package). The increase and decrease of performance mainly depend on the overloading and underloading memory usage. Hence, the underloading memory amount can be effectively used, as RAM over HDFS, to boost the performance. The logarithmic scale evaluation of the TeraSort micro-

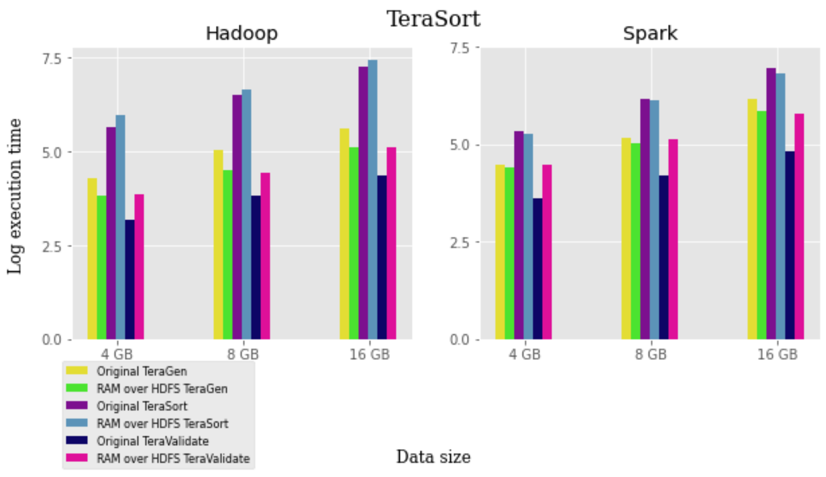


Fig. 4.5: Evaluation of TeraSort compared to RAM over HDFS approach

benchmark for TeraGen, TeraSort, and TeraValidate jobs compared to RAM over the HDFS approach is shown in Fig. 4.5. The logarithmic scale improves the graph's appearance because the execution time difference between TeraSort micro-benchmark jobs (TeraGen, TeraSort, TeraValidate) is significant.

The figure shows that the Teragen job works faster in Hadoop and Spark environments with RAM over the HDFS approach. In contrast, the other two TeraSort and TeraGen jobs are faster in the original approach.

In the Hadoop environment for 4GB, 8GB, and 16 GB data TeraGen works 1.36, 1.32, and 1.4 times faster than the original approach. TeraGen and TeraSort are faster in RAM over the HDFS approach for the Spark environment, while TeraValidate is faster in the original one. The TeraGen for 4 GB, 8GB, and 16 GB data works correspondingly 1.07, 1.11, and 1.26 times, and TeraSort 1.04, 1.15, and 1.12 times faster than the original approach. At the same time, TeraValidate is faster, correspondingly 2.32, 2.56, and 2.59 times than RAM over the HDFS approach.

The experimental results show that for micro-benchmarks using RAM over the HDFS approach, not always provide a performance boost, while for real applications mostly there is a way to reach performance improvement using either compression techniques or RAM over the HDFS approach.

5. Conclusion. The article presented performance-efficient Recommendation and Prediction service for Hadoop and Spark Big Data frameworks based on data compression and in-memory file system. The suggested service consists of Prediction and Recommendation modules to predict an application's execution time with given metrics and provide optimal configurations considering the simulation time. The codeless service can be easily expanded for new scientific workflows by adding configuration files describing all the applications' features and train regression model and encoder.

The service has been evaluated using diverse configuration parameters, like the environment, several data nodes, or the compression method. Some applications and micro-benchmarks have been studied to improve the performance, such as Log Analyzer, WordCount, and K-Means. Each approach has a different effect on performance for different types of applications. The relative error of the predicted boost for Log Analyzer and WordCount in Hadoop is about 3–3.5% and in Spark is about 6-10%, while for K-Means, it is less than 7.3% and 10%, respectively.

It is planned to continue the experiments and studies to improve regression models' accuracy and find similar patterns in different applications considering the energy efficiency alongside the simulation time.

Acknowledgement. The European Commission through the NI4OS, EaPConnect Projects and the COST Action IC1305, 'Network for Sustainable Ultrascale Computing (NESUS)', and State Thematic Project entitled "Towards the Green e-Infrastructures" supported the work presented in this paper.

REFERENCES

[1] A. Mohamed, M. K. Najafabadi, Y. B. Wah, E. A. K. Zaman, R. Maskat, *The state of the art and taxonomy of big data analytics*, Artificial Intelligence, Vol. 53, 2020, No. 2, pp. 989–1037, doi:10.1007/s10462-019-09685-9.

[2] U. R. Pol, *Big data analysis: comparision of Hadoop MapReduce and apache spark*, International Journal of Engineering Science, Vol. 6389, 2016, doi: 10.4010/2016.1535.

[3] J. Dean, S. Ghemawat, *MapReduce: a flexible data processing tool*, Communications of the ACM, Vol. 53, 2010, No. 1, pp. 72–77, doi: 10.1145/1629175.1629198.

[4] H. Won, M.C. Nguyen, M.S. Gil, Y.S. Moon, K.Y. Whang, *Moving metadata from ad hoc files to database tables for robust, highly available, and scalable HDFS*, The Journal of Supercomputing, Vol 73, 2017, No. 6, pp. 2657–2681, doi: 10.1007/s11227-016-1949-7.

[5] K. Shvachko, H. Kuang, S. Radia, R. Chansler, *The hadoop distributed file system*, IEEE 26th symposium on mass storage systems and technologies (MSST), 2010, pp. 1–10, doi: 10.1109/MSST.2010.5496972.

[6] Al-Laham, M.— El Emary, I.M., *Comparative study between various algorithms of data compression techniques*, IJCSNS International Journal of Computer Science and Network Security, Vol. 7, 2007, No. 4, pp. 281–291.

[7] A. Kocharyan, B. Ekane, B. Teabe, G.S. Tran, H. Astsatryan, D. Hagimont, *A remote memory sharing system for virtualized computing infrastructures*, IEEE Transactions on Cloud Computing, 2020, doi: 10.1109/TCC.2020.3018089.

[8] H. Astsatryan, A. Kocharyan, D. Hagimont, A. Lalayan, *Performance Optimization System for Hadoop and Spark Frameworks*, Cybernetics and Information Technologies: Vol. 20, 2020, No. 6, pp. 5–17, doi: 10.2478/cait-2020-0056.

[9] A. Kocharyan, B. Teabe, V. Nitu, A. Tchana, D. Hagimont, H. Astsatryan, H. Kocharyan, *Intra-node Cooperative Memory Management System for Virtualized Environments*, IEEE Ivannikov Memorial Workshop, 2018, pp. 56–60, doi: 10.1109/IVMEM.2018.00018.

[10] D.C. Vinutha, G. Raju, *In-Memory Cache and Intra-Node Combiner Approaches for Optimizing Execution Time in High-Performance Computing*, SN Computer Science, Vol. 1, 2020, No. 2, pp. 1–7, doi: 10.1007/s42979-020-0089-6.

[11] Lee, Woo-Hyun— Jun, Hee-Gook— Kim, Hyoung-Joo, *Mapreduce performance enhancement using in-node combiners*, International Journal of Computer Science and Information Technology, Vol. 7, 2015, No. 5, pp. 1–17, doi: 10.5121/ijcsit.2015.7501.

[12] N.S. Islam, M.W. Rahman, R. Jose, H. Wang, H. Subramoni, C. Murthy, D.K. Panda, *High performance RDMA-based design of HDFS over InfiniBand*, Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, 2012, pp. 1–12, doi: 10.1109/SC.2012.65.

[13] L. Xiaoyi, S. I. Nusrat, Md. Wasi-Ur-Rahman, J. Jithin, S. Hari, W. Hao, K. P. Dhabaleswa, *High-performance design of Hadoop RPC with RDMA over InfiniBand*, 42nd international conference on parallel processing: 2013, pp. 641–650, doi: 10.1109/ICPP.2013.78.

[14] Z. Jing, W. Gongqing, X. Hu, W. Xindong, *A distributed cache for hadoop distributed file system in real-time cloud services*, ACM/IEEE 13th International Conference on Grid Computing: 2012, pp. 12–21, doi: 10.1109/Grid.2012.17.

[15] Y. Luo, S. Lou, J. Guan, S. Zhou, *A RAMCloud Storage System based on HDFS*, Architecture, implementation and evaluation: Journal of Systems and Software, Vol. 86, 2013, No. 3, pp. 744–750, doi: 10.1016/j.jss.2012.11.025.

[16] S. Aibo, M. Zhao, X. Yingying, L. Junzhou, *MHDFS: A memory-based hadoop framework for large data storage*, Scientific Programming: 2016, pp. 1–12, doi: 10.1155/2016/1808396.

[17] P. Narges, M. Ali, *Estimating runtime of a job in Hadoop MapReduce*, Journal of Big Data, Vol. 7, 2020, No. 44, pp. 1–18, doi: 10.21203/rs.2.20701/v1.

[18] M. Sara, E. Iman, A. I. Mohamed, *A Machine Learning Approach for Predicting Execution Time of Spark Jobs*, Alexandria Engineering Journal, Vol. 57, 2018, No. 4, pp. 3767–3778, doi: 10.1016/j.aej.2018.03.006.

[19] M. L. Alberto, S. Sergio, R. C. Antonio, *RESTest: Black-Box Constraint-Based Testing of RESTful Web APIs*, International Conference on Service-Oriented Computing: 2020, pp. 459-475, doi: 10.1007/978-3-030-65310-1_33.

[20] N. Sayalee, B. Tripti, *HMR log analyzer: Analyze web application logs over Hadoop MapReduce*, International Journal of UbiComp: Vol. 4, 2013, No. 3, pp. 41-51, doi: 10.5121/iju.2013.4304.

[21] K. Krishna, N. M. Murty, *Genetic K-means algorithm*, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics): Vol. 29, 1999, No. 3, pp. 433–439, doi: 10.1109/3477.764879.

[22] E. Ostertagová, *Modelling using polynomial regression*, Procedia Engineering, Vol. 48, 2012, pp. 500–506, doi: 10.1016/j.proeng.2012.09.545.

[23] H. Astsatryan, V. Sahakyan, Y. Shoukourian, J. Dongarra, P.H. Cros, M. Dayde, P. Oster, *Strengthening compute and data intensive capacities of Armenia*, 14th RoEduNet International Conference-Networking in Education and Research: 2015, No. 3, pp. 28–33, doi: 10.1109/RoEduNet.2015.7311823.

[24] Yu. Shoukourian, V. Sahakyan, H. Astsatryan, *E-Infrastructures in Armenia: Virtual research environments*, Ninth International Conference on Computer Science and Information Technologies Revised Selected Papers, 2013, pp. 1–7, doi: 10.1109/CSITechnol.2013.6710360.

[25] T. Fushiki, *Estimation of prediction error by using K-fold cross-validation*, Statistics and Computing, vol. 21, 2011, No. 2, pp. 137–146, doi: 10.1007/s11222-009-9153-8.

Edited by: Dana Petcu
Received: Nov 1, 2021
Accepted: Dec 1, 2021

Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2021 SCPE. Volume 22, Issues 4, pp. 413–423, DOI 10.12694/scpe.v22i4.1948

K-WAY BALANCED GRAPH PARTITIONING FOR PARALLEL COMPUTING

SIDDHESHWAR V. PATIL[*] AND DINESH B. KULKARNI[†]

Abstract. In modern computing, high-performance computing (HPC) and parallel computing require most of the decision-making in terms of distributing the payloads (input) uniformly across the available set of resources, majorly processors; the former deals with the hardware and its better utilization. In parallel computing, a larger, complex problem is broken down into multiple smaller calculations and executed simultaneously on several processors. The efficient use of resources (processors) plays a vital role in achieving the maximum throughput which necessitates uniform load distribution across available processors, i.e. load balancing.

The load balancing in parallel computing is modeled as a graph partitioning problem. In the graph partitioning problem, the weighted nodes represent the computing cost at each node, and the weighted edges represent the communication cost between the connected nodes. The goal is to partition the graph G into k partitions such that - I) the sum of weights on the nodes is approximately equal for each partition, and, II) the sum of weights on the edges across different partitions is minimum. In this paper, a novel node-weighted and edge-weighted k-way balanced graph partitioning (NWEWBGP) algorithm of $O(n^2)$ is proposed. The algorithm works for all relevant values of k, meets or improves on earlier algorithms in terms of balanced partitioning and lowest edge-cut. For evaluation and validation, the outcome is compared with the ground truth benchmarks.

Key words: Parallel Computing, Load Balancing, Graph Partitioning

AMS subject classifications. 68W10, 05C85

1. Introduction.

In computer science, graphs are commonly used to depict various real-world problems, including but not limited to social, biological, and information networks. The scale of such networks continues to grow with each passing day, especially in the domains like social networks, task scheduling, cloud computing, data centers, etc. In such examples, graph-based computations are central to the core functions of the applications.

In parallel computing, the problem of graph partitioning is more relevant [3]. There is a need to assign data/code equally among available processors while minimizing the communication overhead to exploit the parallelization. The goal is to balance the complete workload on available processors and reduce inter-process communication to achieve optimum efficiency and throughput. The operating costs of all of these systems are high, it is necessary to utilize all of the processing capacity optimally. So, to solve any problem on a parallel system, it should be divided into sub-problems. Each sub-problem has some computational load. There are dependencies between sub-problems due to communication. The effectiveness of parallel calculation is determined by the effective distribution of computational load across all available processors. Graph partitioning is a technique for simulating load balancing issues. The graph should be partitioned taking two measures into account. Firstly, the total weights on the number of nodes (computational load) of each part should be approximately equal. Secondly, to guarantee minimum communication overhead, the weights on the number of edges incident on the nodes of different partitions should be minimum [14].

It's worth noting that the majority of existing research focuses on graphs with either unit weighted nodes/edges or weighted graphs (only edge weighted). Such graphs cannot appropriately model the variety of partitioning problems. Few researchers use greedy optimization techniques [9, 12] while others use approximate optimization techniques, like spectral clustering [11], some use recursive algorithms like multilevel graph partitioning [10] to solve the problem. There is little work on node-weighted graphs as compared to edge-weighted graph partitioning. Also many algorithm has running time of $O(n^3)$ and more. To address these

[*]Research Scholar, Department of CSE, Walchand College of Engineering, Sangli (Assistant Professor, ADCET, Ashta) (siddheshwar.patil@walchandsangli.ac.in).

[†]Professor, Department of Information Technology, Walchand College of Engineering, Sangli (dinesh.kulkarni@walchandsangli.ac.in).

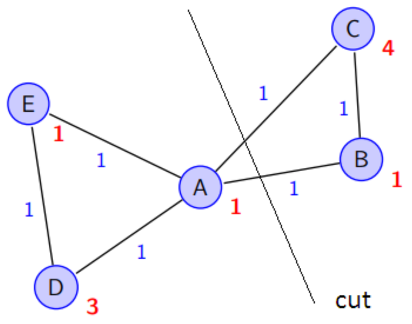


Fig. 2.1: Node-weighted balanced partitioning

issues, the multi-objective k-way graph partitioning (node-weighted and edge-weighted balanced graph partitioning) is proposed in this paper. This work is not similar to state-of-art approaches which use collapsing nodes and edges like multilevel graph partitioning approaches. It employs a heuristic approximation-based approach.

The NWEWBGP provides following main contributions:

1. Literature is critically reviewed and problems are identified.
2. The best partitioning initialization is selection of balanced factor that helps to improve the accuracy and decrease the computations.
3. The partitioning algorithm, NWEWBGP that address the node balance and edge-cut problem.
4. Performance evaluation of the proposed algorithm and comparison with standard benchmark tools such as Metis.

The remaining sections of the paper are organised as follows.
Section 2 introduces the graph partitioning problem. Section 3 provides state-of-the-art algorithms related to the scope of our work. The implementation, results of the proposed algorithm are discussed in Section 4. Concluding remarks are finally stated in Section 5.

2. Graph Partitioning Preliminaries. A k-way partitioning of G divides V into k disjoint parts V_0, V_1, ...,V_k. The total node weights for each subset should be approximately equal, and the total weights of the edges linking the subsets are minimal. The edges cut (cut-cost) are the edges that connect V_0 , V_1, ...V_k and it is denoted by E_c. Cut-weight is the overall weight on E_c. The weight of a subset V_i denoted by $w(V_i)$ is the total weight of the nodes/vertices in V_i i.e., $w(V_i) = \sum_{v \subset V_i} w(v)$ for $i = 0, 1, \ldots k$. Figure 2.1 shows example of balanced partitioning. There are two equal partitions ($[A, D, E]$ and $[B, C]$) with total node-weight as 5 per partition and cut-cost is 2.

3. Related Work. Graph partitioning has been the subject of significant research over many years, which was less attempted to be comprehensively reviewed. The k-way balanced graph partitioning is practically important to any application which requires large graph data. So, even though it is NP-hard and available approximation s are too expensive, various practical solutions have been proposed using heuristics [18, 1, 2]. A survey of partitioning techniques is presented in the following subsections.

3.1. Local Graph Partitioners. The category of local graph partitioners includes any partitioner which makes use of local search, which is also known as iterative vertex swapping or local refinement. Simply put, local search aims to improve an existing graph partitioning by swapping vertices between partitions to minimize some objection function, usually, minimum edge-cut. Local graph partitioners vary in how they select which vertices to swap and which partitions they will consider sending vertices between (e.g. adjacent partitions, or highly imbalanced partitions). Although the global partitioning method already produces a balanced partition, the local method tries to further improve it. The potential for improvement depends upon the difference between current cut-cost and minimum cut-cost.

3.1.1. Random Vertex Distribution. To solve the graph partitioning problem, one simple way is to randomly distribute V vertices among p parts [17]. This method starts with all parts being empty. The vertices are one after another assigned to a part by randomly choosing one part which has less than V/p vertices. The random partitioning method produces partitions with the cut size (edge-cut) of approximately $(1 - 1/p) * E$. This cut size is usually much higher than the lowest possible one. Although this method is not a good approach for graph partitioning, it may be used as starting point for efficient local partitioning methods. Furthermore, this method can be used to compare the solution calculated by the weak method to the solution calculated by a sophisticated one.

3.1.2. Greedy Methods. Greedy methods [13] are commonly used for graph partitioning. Unlike random methods, the greedy method considers adjacency information. There are many different variations of this type of approach. They all start with empty partitions and according to a certain order, they handle one vertex after the another and assign it to a part of the partition. Once an assignment for a part is done, the decision will not be changed later. A common variation is the breadth-first search which is a greedy approach based on breadth-first strategy. It starts with assigning a vertex with the minimum degree to V_0. The method proceeds in a breadth-first manner. It considers all the vertices which are not already assigned to any part. Among them, it adds all vertices to V_0 which are adjacent to any vertex already in V_0. Thus, it proceeds in levels of vertices with the same distance to the initial vertex. The method assigns vertices to V_0 until V_0 of size V/p. Another vertex from the remaining graph which is adjacent to a vertex of V_0 is taken as a new seed for V_1. V_1 and all following parts are filled in the same manner as V_0. The time required for this method is linear to the graph size. The solutions are partitions with compact parts and fairly low cuts. The disadvantage is that the last part consists of all the leftover vertices. They may form elongated or even disconnected parts. To overcome this issue, start growing the parts simultaneously at several vertices, one for each part. The calculation of the two vertices with maximum distance is very time-consuming. Therefore, several heuristics are known to calculate two vertices with high distances.

3.1.3. Kernighan-Lin. Kernighan and Lin [8] proposed a classic example of a local search . Indeed perhaps the first example of a graph partitioner in general. Their key intuition was as follows: given two partitions forming halves of a balanced graph bisection $P_2(G) = \{V_1, V_2\}$, there exist subsets of vertices $A \subset V_1$, $B \subset V_2$ which may be swapped between partitions to generate the arrangement which is globally optimal for some objective function (minimum edge-cut). The Kernighan-Lin (KL) operates in iterations, selecting vertex sets to swap which will result in the greatest improvement (which is objective function gain). Within each iteration, the considers every vertex v_i from a partition (say V_1), then calculates the potential gain when swapping v_i with each vertex v_j from partition V_2. The pair (v_i, v_j) with the highest gain is marked for swapping (i.e. v_i, v_j are added to A, B respectively) and the next vertex in V_1 is considered with each vertex in V_2. Note that when considering swapping the neighbors of vertices already marked in this iteration, the potential objective gain for those neighbors is calculated as if marked vertices have already been swapped. When every vertex has been considered, sets A; B are swapped between partitions and the next iteration may begin. Iterations continue until the total gain for all suggested swaps is ≤ 0.

There are two main issues with the KL as a graph partitioner. The first is that it is limited to improving the partitioning quality of graph bisections, rather than k-way partitioning. The second is that it is highly expensive, with a single iteration of the having complexity $O(n^3)$ with $n =\mid V \mid$.

3.1.4. Fiduccia-Mattheyses. To address critical issues with the KL , several improvements have been proposed. Perhaps most significantly, Fiduccia and Mattheyses [16] present a modified (KL/FM) which is significantly less expensive. The iteration complexity for KL/FM is $O(m)$ with $m =\mid E \mid$ and upper bound for m is $(n-1)^2$. There are two major differences between the KL/FM and KL s. Firstly, the vertex swapping between bisections is asymmetric, i.e. vertices are marked for transfer individually, rather than as a pair with a vertex from the other partition. This means that for each vertex considered for swapping, it is not necessary to consider every vertex from the other partition. Secondly, Fiduccia and Mattheyses use a data structure called a bucket queue for efficiently updating neighbor objective function gain after marking vertices for swapping. Despite the improvement that the KL/FM represents, it shares KL's original limitation of being applied only to graph bisections. However, there are extensions of local search techniques which generalize for improving

k-way partitioning.

3.1.5. Simulated Annealing. The optimization technique, Simulated Annealing (SA) [5] is a local search statistical technique used to get the answer by local arrangements. The description of a solution and neighborhood relation between solutions is important. A cost function assigns a specific cost value to each solution and the cost function should be minimized by changing from one solution to another according to the neighborhood relation. SA iteratively proposes a new solution and evaluates its cost value. The new answer is preserved if the new value is lower than the previous one. Otherwise, a new solution is kept with some probability. This process is repeated till the desired solution can be found or a maximum number of iterations are exceeded.

3.2. Global Graph Partitioners. In general, global graph partitioners refer to those partitioners which, unlike their local counterparts, take an entire unpartitioned graph as input. By default, such partitioners are also executed within the confines of a single machine. They are the most commonly used family of techniques due to their effectiveness. Indeed Karypis and Kumar's Metis [7] is considered the de-facto standard for partitioning quality. However, it is also likely that the popularity of such techniques is at least partially due to their being relatively simple to use and available in several robust software packages.

3.2.1. Spectral Techniques. Donath and Hoffman [4] suggested spectral graph partitioning for computing bisections of a graph G . Firstly, the Laplacian matrix L is computed by subtraction of $G's$ adjacency matrix A from its degree matrix D, $L = D - A$. The second step is to compute the eigenvector associated with $G's$ second smallest eigenvalue. This relies upon the intuition of eigenvector, called the Fiedler vector, which contains an integer value for each vertex, which corresponds to its connectedness in the graph. Using this value as an ordering, the then divides the vertices of G around the median, into two sets of equal size. These sets represent a bisection that is good for minimum edge-cut. Note that, this generalizes to producing k-way partitioning of graphs through recursively bisecting generated partitions. All other spectral partitioning techniques extend this core algorithm. The k-way spectral graph partitioning is achieved in the following way - It finds first k eigenvectors of a Laplacian matrix L and uses k-means algorithm for the final partitioning. i. e. it partitions data set into clusters based on grouping of eigenvectors.

3.2.2. Multilevel Approach. Multilevel technique [15] is effective for effectively partitioning graphs. The tactics for coarsening and local improvement dominate the efficiency of the multilevel method. The vertices' weights are the sum of the vertices' weights in the coarsened graph's. As a result, for both the initial and coarsened graphs, the sum of all vertices' weights is equal. Furthermore, the coarsened graph's partitioning corresponds to the initial graph's partitioning. The edges that were connected to the vertices before coarsening do not appear in the coarsened network since they connect the corresponding vertices. The multi-edges are merged into a single edge which has an edge-weight equal to total weight of all merged edges' weight. The cut-size for partitioning before coarsening and partitioning after coarsening should be the same.

The best example of multilevel techniques is Metis [6]. In the coarsening stage, Metis compresses a graph G by computing maximal edge matching. An edge matching is defined as a set of edges E_M from G, such that no two edges in E_M are incident upon the same vertex. Given such matching, a single level of the compressed graph is computed by combining vertices connected by an edge $e \in E_M$, treating each pair as a single "multi" vertex. Compression is performed using these matching rather than, for example, arbitrarily combining vertices as it prevents anyone "multi" vertex form containing many more elements than another. As a result, a balanced partitioning of the compressed graph will match the original graph's balanced partitioning. Next, when computing an initial partitioning for the compressed graph G_M, Metis uses a technique based on spectral recursive bisection. Finally, in the uncoarsening stage of partitioning, Metis uses the Greedy Refinement local algorithm to move "multi" vertices between partitions, improving the partitioning after each step of match/combine compression has been reversed. Table 3.1 shows the comparative study of prominent algorithms with NWEWBGP.

Table 3.1: Comparison of different graph partitioning techniques

Technique	**Weighted Nodes**	**Weighted Edges**	**Method used**	**Best used for**	**Key points**
KL	No	No	Kernighan Lin algorithm	Bipartitioning	1) Handles only unit edge weights 2) Difficult to extend to k-way graph partitioning 3) Can't partition for varying node and edge weights 4) Need dummy nodes in imbalance situation 5) A pass's time complexity is high, $O(n^3)$ 6) Execution time is longer
FM	No	No	Fiduccia Mattheyses algorithm	Bipartitioning	1) Handles only unit edge weights 2) A pass's time complexity is high, $O(n^3)$ 3) Execution time is longer
MGP	Yes	Yes	Multilevel graph partitioning	k-way partitioning	1) Recursive in nature 2) Edge weighted centric 3) A pass's time complexity is high, $O(n^3)$
Spectral	No	Yes	Laplacian, Eigen Computations	k-way partitioning	1) Can't handle node-weighted graph 2) Time complexity is high, $O(n^3)$ 3) Computationally expensive
Proposed Algorithm	Yes	Yes	Sub-graph computations	k-way partitioning	1) Handles both node and edge weighted graph 2) Flexible, Good quality solutions 3) Time complexity is low, $O(n^2)$

Table 4.1: NWEWBGP Algorithm Conventions

Variable	**Definition**
G	Graph
V	Set of Nodes (Vertices)
E	Set of Edges
ϵ	Maximum ratio
Max_weight	Maximum weight
Min_weight	Minimum weight
k	Number of partitions
w	Sum of weights of all nodes
c	Sum of weights of all edges
Nbr	Neighbor node

4. Node Weighted and Edge Weighted k-way Balanced Graph Partitioning (NWEWBGP).

This work presents the NWEWBGP algorithm (Algorithm 3). The objective is to generate graph partitioning with approximately balanced node weights. The other objective function is to minimize cut-size (edge-cut). Table 4.1 shows the conventions used for the NWEWBGP algorithm.

4.1. Problem Formulation. Let undirected, connected graph $G = (V, E)$ has non-negative weights on nodes w_v, non-negative weights on edges w_e and partitioning criteria, $k >= 2$. A balance requirement necessitates that all blocks be around the same size. In this work, it's worth mentioning that obtaining a uniform balanced graph division isn't always attainable. As a result, the ϵ option to evaluate a partition's balancing factor is introduced. The term completely balanced will be achieved if $\epsilon = 0$. In our work, this factor varies from 0 to 1 with step of 0.1. The step is incremented by 0.1 if the following requirement will not fulfill.

The $(k, 1 + \epsilon)$-balanced partitioning is the problem of solving G into k partitions such that:

1. Maximum weight of $(1 + \epsilon) * w/k$ in each part
2. Minimum weight of $w/(1 + \epsilon) * k$ in each part

4.2. NWEWBGP Algorithm Pseudocode. A summary of the findings of our NWEWBGP research is presented in this section. Various graph partitioning cases are evaluated and compared with the Metis, a standard benchmark tool. We assess the proposed algorithm's performance on the weighted graphs.

The number of partitions needed, k and ratio factor ϵ is taken as an input. The minimum and maximum weights are calculated as shown in step 1 of Algorithm 3. i.e. Node weight of each subgraph may differ from the average by no more than a factor of max_ratio. In step 2 of the algorithm, all partitions of the graph into subgraphs within weight limits are enumerated. It will return the list of partitions, each partition a list of subgraphs, each subgraph a list of nodes. Firstly it finds the node with the highest weight. It will find all subgraphs containing the heaviest node, within weight limits. For subgraph in subgraphs, check if the subgraph splits the graph into disconnected parts. If so, check parts for min_weight and discard subgraph if any are underweight. While doing so, the remainder graph is checked, if it is empty, then add a 1-part partition(subgraph) otherwise add subgraph to each subpartition.

4.3. Experimental Results and Discussion. The proposed algorithm was implemented in python. The Metis library of python (pmetis) was also used for comparison and validation of the proposed algorithm. This work was carried out on the system having 24GB RAM and P100 NVIDIA's GPGPU (3600 CUDA Cores). Consider the graph examples shown in Figure 4.1 and respective results are shown in Table 4.2. The input graphs used in this study have been generated using a graph generator function. The number of nodes, number of edges, weights on the nodes/edges are randomly generated to test the performance of the heuristics under different conditions. The input was a graph with node-weights, edge-weights, and partitioning criteria. In all the graph examples shown in Figure 4.1 , the values present in the circle are node-weight and the values present outside the circle are node indices. The values at every edge represent edge-weight. The partitioning criteria allows the user to set the number of approximately balanced sub-graphs to be made.

Algorithm 3: NWEWBGP Algorithm Pseudocode

```
Step 1: Estimate the minimum and maximum weights for partitioning graph into $k$ number of parts
        Based on $\epsilon$ ranging from 0 to 1 with step of 0.1, the maximum ratio (threshold) of maximum
    weight to the minimum weight is considered.
        A partitioning of $G$ into $k$ sections at a low cost, such that
            $Min\_weight \leq \sum v \in V_i \quad w(V)/k \leq Max\_weight$
            $Min\_weight = (w/(k*(1+\epsilon)))$
            $Max\_weight = (1+\epsilon)(w/k)$
Step 2: Enumerate all partitions of graph into sub-graphs within weight limits
    While (Remainder Graph $! = 0$)
        Step a): Find a node with highest weight, $\{v_i \mid v_i \in V \ \& \ v_i^w$ is maximum$\}$
        Step b): Find all sub-graphs containing heaviest node within weight limits,
                 $\{G' \mid G' \in G \ \& \ min\_wt \leq G' \leq max\_wt\}$
                 Check if sub-graph splits the graph into disconnected parts?
                 If so, check parts for $Min\_weight$,
                 Discard sub-graph if any are underweight.
Step 3: Find all sub-graphs of graph which contain sub-graph, within limits.
        Sub-graphs must have weight (sum of node weights) within limits. Ignore any nodes in ignore.
        Find all neighbors of sub-graph,excluding nodes in ignore.Use neighbors of the forward node.
        Try adding each neighbor node to sub-graph
        if($G'(w) > max\_wt$) ignore neighbor (nbr too heavy to add to subgraph, skip it later)
        else, New Subgraph, $G'' = G'(w) + Nbgr(w)$
Step 4: Min-cut function: $W(G_1, G_2) = \sum_{x_i \in G_1, x_j \in G_2} w_{ij}$ and $\bar{G}_1$ as the complement of $G_1$.
        $cut(G_1, \ldots, G_k) = \frac{1}{2} \sum_{i=1}^{k} W(G_i, \bar{G}_i)$.
```

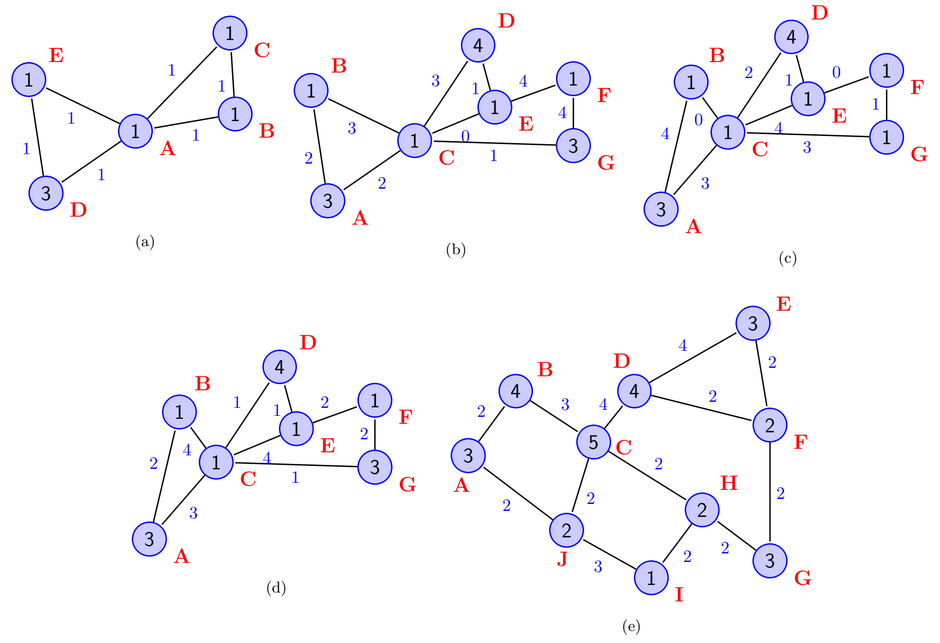


Fig. 4.1: Graph Examples for Experimentation

Table 4.2: NWEWBGP Results and Comparison with Metis

Fig.2	Max. Ratio	Max. Weight	Min. Weight	No. of Parts	NWEWBGP Result		Metis Result	
					Partitions	Cut-cost	Partitions	Cut-cost
(a)	1.2	4.2	2.9	2	[D,E], [A,B,C]	2	[A,B,C,D,E]	0
(b)	1.2	8.4	5.83	2	[A,B,C,G], [D,E,F]	7	[A,B,C,D,E,F,G]	0
(c)	1.4	5.6	2.85	3	[A,B], [C,D], [E,F,G]	11	[], [D,F,G], [A,B,C,E]	6
(d)	1.2	5.6	3.88	3	[A,B], [C,D], [E,F,G]	13	[], [D,G], [A,B,C,E,F]	5
(e)	1.1	15.95	13.18	2	[A,B,C,I,J], [D,E,F,G,H]	8	[A,B,C,I,J], [D,E,F,G,H]	8
(e)	1.2	11.6	8.05	3	[A,G,H,I,J], [B,C], [D,E,F]	12	[A,G,I,J], [B,C,H], [D,E,F]	14
(e)	1.3	9.425	5.58	4	[A,I,J], [D,E], [F,G,H], [B,C]	16	[A,I,J], [D,E], [F,G,H], [B,C]	16

Table 4.3: Selection of balanced factor on graph (e) of Figure 4.1

Balanced Factor (ϵ)	**No. of Parts**	**Max. Wt.**	**Min. Wt.**	**Sub-graphs formed**	**Partitions**	**Cut-cost**
0.1	2	15.95	13.18	14	[3,4,5,6,7], [0,1,2,8,9]	8
0.2	2	17.4	12.08	27	[0,4,5,6,7,8,9], [1,2,3]	12
0.3	2	18.85	11.15	61	[3,4,5,6,7,8,9], [0,1,2]	10
0.1	3	10.63	8.78	4	-	-
0.2	3	11.6	8.05	5	[1,2], [3,4,5], [0,6,7,8,9]	12
0.3	3	12.56	7.43	15	[0,1,2], [3,4,5], [6,7,8,9]	12
0.4	3	13.53	6.9	38	[1,2], [3,4], [0,5,6,7,8,9]	14

The output provides a summary of the graph partitions which includes the total weight of each partition and the cut-cost on the set of examples. 3 levels of partitions were tested in this work, particularly $k = 2, 3, 4$. The results further compare and evaluate the performance of Metis and NWEWBGP.

As explained in Algorithm 3, based on the balanced factor, maximum weight and minimum weight is calculated. The results show that choosing the best-balanced factor helps to divide the graph into more appropriate balanced partitioning. Based on the number of partitions (k) to be made, the proposed algorithm shows an approximate balanced partitioning with cut-cost. For the same examples, the Metis results are also calculated. The NWEWBGP either gives the results same as Metis or improves in some cases. In few of the scenarios, Metis doesn't give the partitions as expected while the NWEWBGP works well by providing an appropriate balance of both node-weighted balance constraint and minimum edge-cut constraint. The NWEWBGP running time is ($O(n)^2$), which is better than Metis running time of ($O(n)^3$).

4.4. Selection of appropriate balanced ratio. The Algorithm 3 presented above is able to compute the partitions of high quality in a reasonable amount of time with balanced ratio (balanced factor), ϵ close to 0. The Table 4.3 shows the 2-partitions and 3-partitions obtained for various values of ϵ on the graph example (e) of Figure 4.1. It is seen that, as we increase the value of ϵ, the number of sub-graphs computed will also increases and the quality of partitions will decreases (Outcome: imbalanced partitions). So, selection of ϵ is more important for graph partitioning problem. From the Table 4.3, in case of $(2, 1 + \epsilon)$-balanced partitioning, the best balanced factor is 0.1. In case of $(3, 1 + \epsilon)$-balanced partitioning, the best balanced factor is 0.2.

4.5. Output Balance Metric. Different partitioning tools provide different output metrics. For instance, the output metrics provided by some graph partitioning are the set size, edge cuts etc. The output metrics provided by our approach are 'Balance' and 'Cut edges'. 'Cut edges' or 'Cut-cost' is similar to the edge cuts as measured by Metis. 'Largest' is the total weight of nodes present in the largest set and similarly 'Smallest' is the total weight in the smallest set. The metric 'Balance' is important for understanding the load balance obtained from the partition. In this work, 'Balance' metric is calculated using the following formula.

$$Balance = S_{max}/S_{opt}$$

where S_{max} is the weight of the largest subgraph and S_{min} is the optimum subgraph (ideal) size given by:

$$S_{opt} = G/k$$

where G is the total weight of nodes in the graph and k is the number of partitions. Table 4.4 shows 'Balance' and 'Cut-cost' metric comparison between proposed work and Metis. The Cut-cost comparison is also shown.

The Figures 4.2 and 4.3 shows the comparison for the 'Balance' metric and 'Cut-cost' metric, respectively. It is seen that, the proposed work on NWEWBGP gives better results than the Metis.

Table 4.4: Output balance metric and cut-cost

Sr. No.	No. of Nodes	No. of Edges	Parts	Proposed method		Metis result	
				Balance Metric	Cut-cost	Balance Metric	Cut-cost
1	2000	2000	3	1.23	341	1.44	366
2			4	1.14	447	1.39	482
3	4000	4500	3	1.76	589	1.97	634
4			4	1.23	665	1.51	738

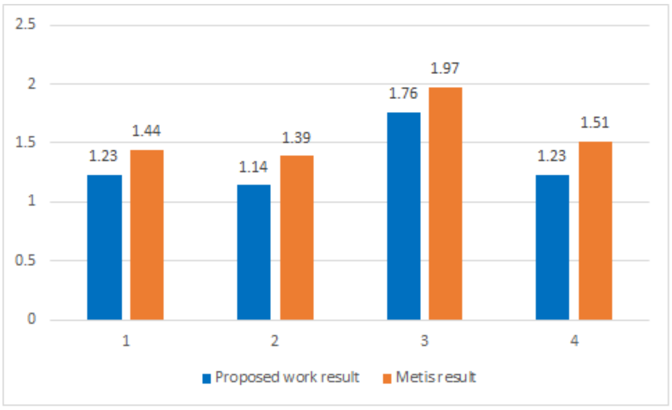


Fig. 4.2: Output Balance Metric

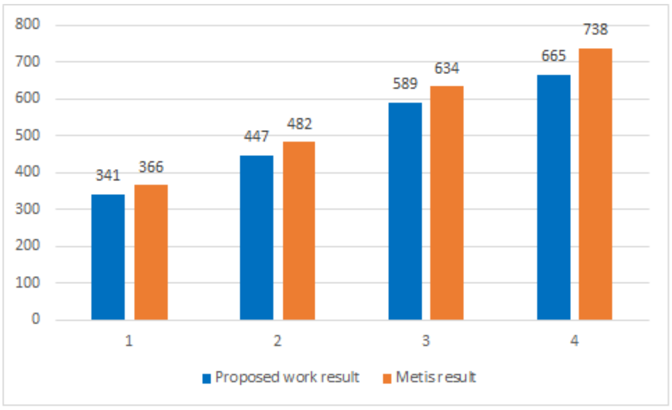


Fig. 4.3: Cut-cost

5. Conclusion. In this paper, the existing graph partitioning techniques are studied and the necessity of proposed graph partitioning in the context of parallel computing is discussed. The graph partitioning problem can be defined with several objective functions, however, the objective discussed in this study emphasizes the need in parallel computing domain - the balanced partitioning with minimum cut-cost.

The proposed node-weighted and edge-weighted k-way balanced graph partitioning algorithm shows an approximately balanced partitioning and can aid in achieving better utilization of processors in parallel computing. The results demonstrate that the algorithm succeeds in effectively balancing the node weights across partitions and minimizing the cut-cost (weighted edge-cut) as compared to the results of Metis, a benchmark tool for graph partitioning. The proposed NWEWBGP method also outperforms the existing state-of-the-art algorithms in terms of running time.

REFERENCES

[1] K. Andreev and H. Racke, *Balanced graph partitioning*, Theory of Computing Systems, 39 (2006), pp. 929–939.

[2] A. Buluç, H. Meyerhenke, I. Safro, P. Sanders, and C. Schulz, *Recent advances in graph partitioning*, Algorithm engineering, (2016), pp. 117–158.

[3] J. Doe, *Load balancing strategies in parallel computing: Short survey*, (2015).

[4] W. E. Donath and A. J. Hoffman, *Lower bounds for the partitioning of graphs*, in Selected Papers Of Alan J Hoffman: With Commentary, World Scientific, 2003, pp. 437–442.

[5] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon, *Optimization by simulated annealing: An experimental evaluation; part i, graph partitioning*, Operations research, 37 (1989), pp. 865–892.

[6] G. Karypis and V. Kumar, *Metis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices*, (1997).

[7] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM Journal on scientific Computing, 20 (1998), pp. 359–392.

[8] B. W. Kernighan and S. Lin, *An efficient heuristic procedure for partitioning graphs*, The Bell system technical journal, 49 (1970), pp. 291–307.

[9] M. Kushwaha and S. Gupta, *Various schemes of load balancing in distributed systems-a review*, International Journal of Scientific Research Engineering & Technology (IJSRET), 4 (2015), pp. 741–748.

[10] M. Leng, S. Yu, and Y. Chen, *An effective refinement algorithm based on multilevel paradigm for graph bipartitioning*, in International Conference on Programming Languages for Manufacturing, Springer, 2006, pp. 294–303.

[11] R. Peng, H. Sun, and L. Zanetti, *Partitioning well-clustered graphs: Spectral clustering works!*, in Conference on Learning Theory, PMLR, 2015, pp. 1423–1455.

[12] V. Prasad, *Load balancing and scheduling of tasks in parallel processing environment*, Int. J. Inf. Comput. Technol, 4 (2014), pp. 1727–1732.

[13] M. Predari and A. Esnard, *A k-way greedy graph partitioning with initial fixed vertices for parallel applications*, in 2016 24^{th} Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP), IEEE, 2016, pp. 280–287.

[14] C. Sakouhi, A. Khaldi, and H. B. Ghezal, *An overview of recent graph partitioning algorithms*, in Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), The Steering Committee of The World Congress in Computer Science, Computer ..., 2018, pp. 408–414.

[15] P. Sanders and C. Schulz, *Engineering multilevel graph partitioning algorithms*, in European Symposium on Algorithms, Springer, 2011, pp. 469–480.

[16] M. Sheblaev and A. Sheblaeva, *A method of improving initial partition of fiduccia–mattheyses algorithm*, Lobachevskii Journal of Mathematics, 39 (2018), pp. 1270–1276.

[17] I. Stanton and G. Kliot, *Streaming graph partitioning for large distributed graphs*, in Proceedings of the 18^{th} ACM SIGKDD international conference on Knowledge discovery and data mining, 2012, pp. 1222–1230.

[18] J. Sun, H. Vandierendonck, and D. S. Nikolopoulos, *Graphgrind: Addressing load imbalance of graph partitioning*, in Proceedings of the International Conference on Supercomputing, 2017, pp. 1–10.

Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2021 SCPE. Volume 22, Issues 4, pp. 425–444, DOI 10.12694/scpe.v22i4.1952

TRUST MANAGEMENT IN THE WORLD OF CLOUD COMPUTING. PAST TRENDS AND SOME NEW DIRECTIONS

MAHREEN SALEEM, M.R WARSI, SAIFUL ISLAM, AREESHA ANJUM AND NADIA SIDDIQUII *

Abstract. Over the past years, Cloud computing has become one of the most influential information technologies to combat computing needs because of its unprecedented advantages. In spite of all the social and economic benefits it provides, it has its own fair share of issues. These include privacy, security, virtualization, storage, and trust. The underlying issues of privacy, security, and trust are the major barriers to the adoption of cloud by individuals and organizations as a whole. Trust has been the least looked into since it includes both subjective and objective characteristics. There is a lack of review on trust models in this research domain. This paper focuses on getting insight into the nomenclature of trust, its classifications, trust dimensions and throws an insight into various trust models that exist in the current knowledge stack. Also, various trust evaluation measures are highlighted in this work. We also draw a comparative analysis of various trust evaluation models and metrics to better understand the notion of trust in cloud environments. Furthermore, this work brings into light some of the gaps and areas that need to be tackled toward solving the trust issues in cloud environments so as to provide a trustworthy cloud ecosystem. Lastly, we proposed a Machine Learning backed Rich model based solution for trust verification in Cloud Computing. We proposed an approach for verifying whether the right software is running for the correct services in a trusted manner by analyzing features generated from the output cloud processed data. The proposed scheme can be utilized for verifying the cloud trust in delivering services as expected that can be perceived as an initiative towards trust evaluation in cloud services employing Machine learning techniques. The experimental results prove that the proposed method verifies the service utilized with an accuracy of 99%.

Key words: Cloud SLA, Trust, Trust Evaluation, Trust Management, SRM, Ensemble classifier

AMS subject classifications. 65M25, 68M14

1. Introduction.

Cloud Computing as a paradigm shift, has enabled convenient and on-demand /immediate accessibility to the available pool of shared resources. With the unstoppable advancements in the networking technologies and rapidly increasing demand for computing resources, cloud adoption has become as inevitable and convenient as the daily-life utilities like gas, water, and electricity. Cloud service providers (CSP) commonly utilize Virtual Machine (VM) [1] technologies backed up by the contemporary data centers to dynamically provide computing services like ubiquitous network access, computing-on-demand, rapid resource elasticity on a pay-as-you-use basis [2]. The different types of services that Cloud computing offers consist of 'Infrastructure as a Service (IaaS)', 'Platform as a Service (PaaS)', and 'Software as a Service (SaaS)'. With IaaS, a CSP offers computing, storage or networking infrastructure to its customers for use. With Platform as a Service (PaaS), a CSP allows customers to leverage the shared location-independent cloud resources from the resource pool of CSP to operate custom applications. Software as a Service (SaaS), allows consumers to employ softwares running on CSP's infrastructure. For most customers, the facility of pay-as-you-go is a great motivation to migrate to Cloud, as it relieves the user of the planning and maintenance of the underlying architecture.

Yet, in spite of the swift acceleration of cloud services adoption, most IT administratives still are skeptical to commend a "cloud-first" approach. Even worse, some desist to utilize any of the cloud services, quoting privacy and security issues, functional challenges and most prominently the diminishing of control over the data after it goes beyond the boundary [3]. Undoubtedly, the issue of trust management is one of the most complex in cloud computing systems where a collection of services, applications, and nodes function collectively to serve each other [4] [5] [6]. This calls for the deployment of urgent means that uphold awareness for transparency, accountability, and, governance of the CSPs. To improve adoption of cloud, trust is a must desired criteria that

*Department of Computer Engineering, Aligarh Muslim University, India (mehkhan27@gmail.com, warsimr@yahoo.com, saifulislam@zhcet.ac.in, aressha.anjum@zhcet.ac.in, nadiasid2134@gmail.com)

if guaranteed, would motivate individuals and organizations at whole to migrate to the cloud [7] [8]. The solution is to build a holistic cloud trust scheme — one that involves potential business and IT stakeholders to deliver accurate checks and balances to create secure cloud environment that allows a controlled and cost-effective cloud investment.

By building a cloud trust framework to evaluate and monitor, upgrade and enhance their cloud environment, and to certify and comply with it, IT experts can transform cloud fear into an possibility to tackle increasingly complicated security and privacy issues [3] [9]. However, despite of the significance of cloud trust evaluation in trust management, there is a lack of relevant literature addressing this challenge comprehensively. Considering trust to play a pivot role in Cloud systems the main objective of this paper is to survey the existent trust models and to highlight the significant contemporary trust management challenges faced in Cloud Computing and finally to propose a solution for verifying trust in Cloud service utilization.

Briefly, the outlines of this paper are as follows:

- Providing the basic semantics, taxonomy, and notion of trust specific to Cloud Computing.
- Offering the overview of Trust evaluation mechanism in Cloud environment.
- Systematically discussing the significant Trust Models in Cloud Computing.
- Comparing the reviewed Trust models and mechanisms, and outlining their major features.
- Highlighting the open issues and suggestions to establish the trust in cloud computing systems.
- Proposing a Machine Learning backed Rich model based solution for trust verification in Cloud Computing.

The rest of this paper is structured as follows. Section 2 introduces the trust semantics and terminologies. Section 3 offers the overview of trust in Cloud context and gives insight into challenges in trust management. Section 4 describes various trust dimensions and trust evaluation measures in cloud systems. Section 5 presents an overview of a few proposed Trust cloud models in the literature. Section 6 presents the discussion and outlines some open issues. In section 7 we propose a Machine Learning backed Rich model based solution for trust verification in Cloud Computing. Finally, section 8 ends this paper.

2. Semantics Of Trust. Trust is a complex notion; therefore a multidisciplinary approach is required to describe it. Trust is, for the most part, characterized as "the certainty levels in something or somebody" [10]. In IT environments, this implies that your counterpart functions in accordance with the defined protocols. Trust is to be expanded by alleviating technical and psychological boundaries to utilizing cloud services.

According to Leonard.G et al. [11], trust in something can be described as the level of certainty that an object will perform in an acceptable fashion. This assurance can be related to the 'quality of service' or the 'security and privacy policies' that the object follows. Trust management in the system can assist in establishing collaborations among new devices that have joined the network. Nodes may be reported as safe or unsafe, subject to the trust level [12]. Rousseau and colleagues [13], describe trust as a psychological state of accepting to being intentionally vulnerable on the basis of positive expectations of someone else's behavior. In computer science, trust is considered to be quantifiable, and object X's trust in object Y for any service S implies that X believes that the behavior of Y will be satisfactory for a specified time period within a defined context for the service S.

Reputation and trust are recurrently used exchangeably; though the conception of both is quite very much alike but not the same [11]. Reputation is the belief someone has regarding something. Trust information can be collected from the reputation; however trust formation takes other components as well. Reputation is established on the basis of past behavior of an entity while trust predicts its future behavior [14].

Often times, security is associated with trust. Security guards systems against the malicious entities. Nonetheless, there is a need of essential pre-configured and established information well equipped to protect network entities. In trust context, security measures could be viewed as transporters of trust from the source where it is established to the point where it is required [15]. If the two interacting entities share their key, they have officially established trust with one another.

2.1. Trust Terminology. Alhanahnah et al. [16] presented a taxonomy of trust factors in their article and they define trust-related terms to avoid ambiguity as they often have varying connotations in the literature. Those are defined as follows:

Trust factors (TFs): Trust factors are the criteria that are considered for trust evaluation in CSPs. Security, privacy, and data management are some instances of high-level TFs.

Trust indicators (TIs): These are the indicators that represent trust factors. Score (e.g. 90 percent), or rating (4 out of 5 stars) are TIs that represent the trust factor reputation.

Trustor: An agent that trusts some other entity is termed as trustor. In cloud context, a trustor corresponds to a consumer or user.

Trustee: An entity trusted by the trustor is the trustee. In cloud context, a CSP corresponds to the trustee entity. Two entities are involved in building trust –trustor and trustee – both have two different goals. The essential role of the trustor is to assess the accurate trustworthiness, that is to predict the future behavior, of the trustee. However, the goal of the trustee is to acquire the creditably best trust values.

Trustworthiness: Trustworthiness is perceived as the collection of all significant trust factors.

Trust decision: A trustor decides to connect with the trustee only if it's trust value is perceived to be sufficiently high at that instant. This decision is termed as the trust decision.

3. Understanding Trust in Cloud Context. First off, we sketch the perception of trust in cloud computing and feature notable concerns that highlight the need for constructing and managing the trust in cloud context. Then we put forward the taxonomy and outline how it can be enforced to practical scenarios.

Trust is widely described in generic literature, however, it is still in its infancy in the cloud computing setting. In IT environments, the notion of trust pertains to further than security. It encompasses reliability, dependability, integrity, and capability to execute a service. Trust additionally propels collaboration among groups. In online communities having a majority of unknowns, the trust metric would predict the degree to which a user can trust another user in the network [17]. Trust in cloud computing setting has three important referents: trust in the Cloud provider; trust in the Cloud services that CSP has to offer; and trust in the Cloud as a technology itself. Among the three potential referents the most difficult one seems the trust in the Cloud itself as it involves motivating users to adopt cloud computing as a technology.

Fujitsu Research Institute conducted a 2010 survey [18] that derived 88% of potential cloud end users are concerned mainly about who else has access to their information and necessitated more transparency of what happens in the physical servers at the backend. Surveys suchlike demonstrate the pressing need for IT executives and researchers to address the obstacles to trust urgently. While there exist preventive measures like encryption, ID profiling based access control etc. to mitigate privacy and security risks, they are not sufficient. Due to insufficient confidences in the Cloud, many users are reluctant to migrate their data to the Cloud or to utilize the shared computing resources provided by it. Security, privacy and trust issues have become one of the major barriers to the adoption of Cloud services by many individuals and organizations. Cloud Service providers do not provide guaranteed service performance. In terms of dependability, some initial approaches exist in levels of service availability and reliability [19] [20] [21].

Due to the very complex and distributed nature of cloud computing, cloud consumers lose control over their information, as the data resides on the distributed cloud data centers located across different geographical locations. Concerns over the security of remotely stored information and the consequent ownership and diminishing transparency issues are aggravated by the fact that most customers may not be conscious of the underlying security and privacy measures enforced by the CSP [22]. Building trust plays a significant role to combat these challenges [1] [6] [16] [23]. Undeniably, trust can remunerate the absence of control and build the confidence of the consumers in the security and privacy measures executed by the CSP. As such, a CSP 's trustworthiness is a denotation that the CSP conforms to the security and privacy benchmarks, and meets the safety prerequisites of its consumers [16].

Challenges of Trust Management in Cloud Computing. We discuss in what follows the trust management challenges associated with establishing trust among consumers and CSPs. It categorizes trust management challenges into various levels based on different approaches as outlined in this section.

The Cloud Security Alliance listed some top threats in cloud computing in their report 'Top Threats to Cloud Computing V1.0' [22]. There are many challenges in Cloud Computing in addition to internal and external threats, software bugs, hardware failures, server misconfigurations, etc. [24] [25] [26]; the most significant ones are being highlighted below:

- Security and Privacy
- Integrity
- Interoperability and Portability
- Reputation
- Virtualization
- SLA
- Standardization
- Query and access
- Service quality, and
- Trust management

Trust management facilitates the establishment of trust between entities and specifies the structure and mechanisms that enable the trust to be manifested in a system. For the first time, trust management conception was introduced by 'M. Blaze, J. Feigenbaum, and J. Lacy' [27]. They outline the definition of trust management as "the problem of figuring based on formulated security policies and security credentials if a set of security credentials of an entity satisfies the security policies". To guarantee the trustworthiness of an entity, trust management describes the two aspects of the system trust: 1) What information to gather, and 2) How to gather that information. Trust management encompasses three major components: i) Trust establishment ii) Trust update and iii) Trust revoke [12].

In service oriented IoT, the goal of trust management is to guarantee whether connecting to the service provider is safe in terms of reliability, security, and availability [28] [29] [30]. This trust-related information is either stored centrally or in a distributed setup prior to being delivered to the network.

Trust management of cloud services is a challenging issue. This is accredited to the unique characteristics of cloud services wherein millions of nodes, services, and applications are deployed to serve each other under a single umbrella. The nature of cloud computing is very dynamic where newer cloud services and nodes keep joining the network. Moreover, it is difficult to access the legitimacy of the user trust feedbacks [31]. Also, it is challenging to assign credibility to experienced users and track bad-mouthing in order to identify malicious trust feedback [32].

Navimpour et al. [33] in their survey paper categorized trust issues of cloud systems into four sub-classes, that comprise:

(a) How to define and access trust related to dynamic cloud systems.
(b) How to handle recommended trust information from the malicious entities.
(c) How to provide varying levels of service in accordance to the expected trust degree, and
(d) How to monitor trust values as they update with variation in time and context, and to update the trust information and adapt the system to the dynamic changes as and when they occur in time.

Practically, autonomic trust management is difficult to realize due to the ever-expanding scale of deployment of the cloud of things and the very dynamic nature of the cloud systems further complicates the task [2]. Ryan Ko et al. [10] mentioned the primary issues and challenges in building a trusted cloud environment through the establishment of detective controls and presented the 'TrustCloud' framework, that attends to the accountability issues in cloud computing through technical and policy-based techniques. The authors quote that in spite of auditability contributing a significant role in building trust, the present day leading CSPs (e.g. Amazon EC2/S3, Microsoft Azure etc.) still do not grant complete transparency and facilities to track and audit record access history and the provenience of the server (physical and virtual) usage [18]. At present, clients can at best have transparency to track performance metrics of the virtual hardware and monitor service event logs [34] [35].

4. Trust Management Overview.

4.1. Trust Dimensions. Trust computing is a subcategory of trust management that describes how trust information is collected, what trust features are used, and how the gathered trust values are aggregated to generate the final concluding trust values that are again broadcasted over the network. J. Guo et al. [36] classified trust computing methods in service-oriented IoT into five dimensions: i) Trust composition ii) Trust propagation iii) Trust update iv) Trust formation v) Trust aggregating.

Trust Composition: Trust composition is a major component of trust management and it defines the components that are contemplated in the computation of trust. The set of trust components can be categorized into two components: i) Quality of Service (QoS) component and ii) Social trust component. QoS trust is defined by the level of assurance in the node to deliver the requested service [37]. Social trust component is reflected from social affiliations of entities, their social networking etc and it could be employed to contribute to overall trust derivation.

Trust Propagation: This dimension of trust computation represents how to store the composed trust values in the network. In the literature, two major schemes are outlined; centralized and distributed. In the centralized trust propagation scheme a centralized entity, like the cloud, is responsible for acting as a datastore of trust information for all network entities. A centralized datastore of trust information makes message propagation simple. Also, the centralized server can be responsible for the processing of trust values. Every entity in the network has access to the same central information; hence limited storage issue in the resource-constrained entities is resolved. However, this causes the central node to eventually be a single point of breakdown. In a distributed approach, instead of a central entity, every object in the network has the obligation to compute and store the required trust values, and render recommendations to other objects. However, the distributed approach can suffer from attacks such as bad-mouthing.

Trust Update: Trust information once collected and propagated either centrally or in a distributed manner needs to be updated. When these trust values are updated, is described by the trust update dimension. Trust update can be either event-driven or time-driven. In the event-driven approach, trust update takes place after the occurrence of an event e.g. after completion of a transaction. In the time-driven approach, there is a periodic update in trust values at regular time intervals. In a distributed scenario time-driven approach is considered to be more suitable by virtue of it being energy persevering in contrast to an event-driven scheme [36]. Though, over the course of time the trust value of an entity may waver; hence, making the time-driven approach questionable and less accurate. To maximize accuracy and minimize energy consumption, an optimal time interval needs to be estimated.

Trust formation: Once the trust information is collected from multiple features, trust properties need to be weighted corresponding to their relevance. Thus, trust formation specifies how trust constitution takes place from composite trust features. Trust formation can take place either by single-trust, wherein one single trust property like QoS is considered, or by multi-trust where multiple properties are considered.

Trust aggregation: Once the trust attributes are finalized, trust information is to be collected from the network entities. Trust aggregation defines how these trust feature values are collected from other members in the network. In the current literature, the most common method for combining these trust experiences into a single value is the weighted sum method [36] [38]. Other techniques for aggregating the trust values include Bayesian inference, belief theory, fuzzy logic, and regression analysis [36]. Subjective logic, a type of probabilistic logic, is the most appropriate aggregation technique for fog computing. It takes source trust and its uncertainty into account.

4.2. Components of Trust Evaluation in Cloud Computing. To build confidences in cloud by mitigating challenges in cloud trust management, we primarily require the understanding of the key components impacting cloud trust. The criteria that are considered while evaluating cloud trust are termed as Trust factors (TFs) [16] . Security, privacy, and data management are the instances of high-level (in terms of significance) trust factors. Ryan KO et al. [10] specify the following significant components that affect trust viz:

1. Security - Techniques like encryption that makes it challenging for an unauthorized individual to gain access over some information [39].
2. Privacy – Protecting confidential data from exposure or leakage.
3. Accountability – Obligation of an individual or organization to be answerable and liable for delivering agreed-upon services.
4. Auditability – The relative simplicity of inspecting a framework or a domain. Poor auditability implies inadequately-maintained (or non-existent) evidences or records that empower proficient reviewing of procedures inside the cloud.

In the paper [40] M. Alhamad et al., developed a model for trust evaluation based on the five most significant components of cloud trust; which include security, availability, scalability, and usability parameters. The authors developed a trust evaluation scheme, for IaaS model with an e-learning application as a case study, using fuzzy-set theory since each of these trust properties is characterized by fuzzy aspects.

Parameters of Trust Evaluation. Specifically, trust should be quantifiable, measured for a particular context, and necessitated to be updated. In order to draw a comparative outline of the trust frameworks, Alhanahnah et al. [16] identified five criteria, based on the available literature.

- Standardization effort- whether or not trust frameworks contemplate cloud standards (such as 'Cloud Security Alliance's Cloud Controls Matrix' [41] and 'European Commission initiatives' [42]) in their design.
- Multifaceted criteria - measuring trustworthiness from diverse features, instead of relying only on a single percept for evaluation.
- Consumer perception of trust- determines whether the consumers are allowed to convey their orientation towards specific trust factors through the trust framework.
- Extensibility – ensures that the trust frameworks must be having the capacity to accommodate changes owing to the dynamic and constantly evolving nature of the cloud.
- Context awareness- whether the trust framework can comply to diverse application contexts with varying consumer's trust requirements.

4.3. Trust Value Computation and Evaluation Strategies. A simple method to compute reputation scores in a rating based system, as was used in eBay's reputation forum [43] [44] [45]. However, due to simplicity of this method the results obtained are not much effective. Most commercial websites like Epinions [46], eLance and Amazon use sophisticated approaches that calculate the weighted average over all ratings depending upon the user's credibility. Pan et al. [47] utilize Jaccard's Similarity Coefficient and Pearson Correlation Coefficient to seect trustorthy cloud services. Guo et al. [36] discuss several trust computation techniques that aggregate the trust values obtained from various sources. The authors reported weighted sum to be the most commonly used method in the literature. In reputations based systems [14], the most reputed users have the heaviest impact on the final trust value evaluated from the weighted sum method [48]. Other approaches described by Guo et al. [36] include Bayesian Systems [49] [50] [51], Regression Analysis [52], Belief Models [53], Fuzzy Models [54] [55] [56] [57] [40] [58] [29], and Flow Models [59] [60] [61]. Bayesian systems and Fuzzy models are the two most widespread trust evaluation strategies. Subjective logic, a special type of belief theory and regression analysis, has been mentioned in various studies [36]. Adopting logistic Regression analysis for trust estimation is a recent approach. It being more computation heavier than subjective logic provides better results. Subjective logic is based on the foundation of belief model. Figure 4.1 depicts the relation between the three variables defined by Josang [53], where the trust degree for an object i corresponds to a point in the triangle defined by the tuple

$$w_i = (b_i, d_i, u_i)$$

where *'belief' b* of an observer represents the belief in the object to be in trust state, *'disbelief' d* depicts the likelihood for the object not being in trust state and *'uncertainty' u* satiates the gap between *belief* and *disbelief*, as defined by Josang [53], such that

$$b + d + u = 1$$

Josang and Knapskog [62] gave a metric to evaluate the values of *b,d,u,* as depicted in Equations 4.1 to 4.3, where p and n represent the count of positive and negative experiences respectively.

$$b = \frac{p}{p+n+1} \tag{4.1}$$

$$d = \frac{n}{p+n+1} \tag{4.2}$$

$$u = \frac{1}{p+n+1} \tag{4.3}$$

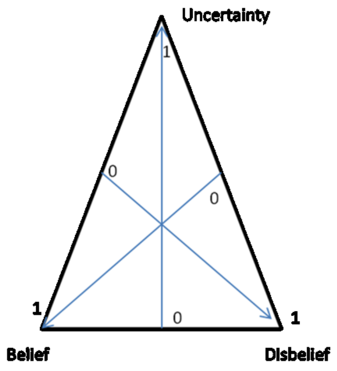


Fig. 4.1: Relationship between trust variables

Mei et al. [63] presented a 'Trusted Bytecode Virtual Machine Module' (TBVMM) as a technique of dynamic remote verification in cloud computing. To solve the dynamics of trust relationships in a distributed environment they used Bayesian model and Kalman filter. Wang et al. [64] investigated the existing dynamic trust level scheduling (DLS) and presented a cognitive model inspired novel Bayesian method for cloud computing. The proposed framework offered low dependability, integrity and safety confidentiality. Barsoum and Hasan [65] have proposed a cloud-based storage strategy that permits indirect reciprocal trust among the CSP and the data owner. Table 4.1 presents a detailed description of various trust computation techniques.

5. Cloud Trust Models. Trust is widely discussed in the generic computing literature, nonetheless, trust still is an emerging concept in the cloud computing environment. The challenges and issues in building trusted Cloud ecosystems has been talked about from various perspectives and there is a lack of any standard model for evaluating cloud trust. Trust is a standout amongst the most significant pointers for benefit determination and suggestion in Cloud adoption. There are three most common categories of trust models 1) Service Level Agreement (SLA) based trust model, 2) Recommendation based trust model and 3) Reputation based trust model. In this section we highlight some of the Cloud Trust models existing in the literature:

I. *TrustCloud Model*
Ryan K Lo and colleagues at Cloud & Security Lab of Hewlett-Packard Laboratories (HP), Singapore and Bristol presented the TrustCloud framework [10] that handles accountability issues in cloud computing through some technological and policy-based mechanisms. In their paper, they highlight that there is a pressing need for research in the domain of accountability in cloud systems. They classified trust components into two categories: Preventive Controls and Detective Controls. Preventive approach mitigates the happening of an event at all (like firewalls). Detective approach identifies the risks that are against the security and privacy policies of the system (like security audit trails, intrusion detection systems (IDS), and logs). Furthermore, there are also corrective controls, which happen to fix any unwanted events that occur.
For the cloud framework, the authors have focused on detective controls since they are non-invasive, and not only investigate the external risks alone, but the risks arising from inside the CSP as well. Although measures to directly stop the occurrence of irregularities are scarce, in cloud computing detective controls serve as some kind of a psychological barrier to policy breaches and even serve as a forensic record should there be any case of non-compliance. Detective controls act in a manner similar to speed cameras for traffic control. Generally, the combination of both the approaches is needed to complement each other for appropriate protection.

II. *EY Cloud Trust Model*
In the report "Building Trust in Cloud" Ernst & Young Global Ltd. (EY) [3] presented the 'EY Cloud Trust lifecycle Model' as a foundation that could be utilized by the organizations to build trusted cloud

Table 4.1: Description of various trust computation techniques

Technique	Description	Features	Source
Rating based system	Users provide a star rating as a feedback for products or services consumed.	Simplicity of method leading to ineffective results. Absence of feedback validity results in unfair ratings.	[43] [44] [45]
Bayesian Systems	Bayesian systems are built around prior beliefs for estimating the mean of a population.	This is useful for rankings so there is a need for a backdrop of a scale of all competing products' number of ratings. It is relevant when dataset is small.	[49] [50] [63] [56] [51]
Regression Analysis	A set of statistical processes for relationship estimation among variables. Mostly used for prediction and forecasting. Specifically it may be used to estimate continuous response variables.	Computation heavier than subjective logic, hence provides better results. Requires complex statistical analysis.	[42]
Belief Models /Subjective logic	It is a framework for dealing with uncertainty. It operates on opinions and beliefs about the world. A special type of belief theory and regression analysis is subjective logic.	It employs components from the 'Dempster-Shafer' belief theory. Uncertain probabilities are based on belief model.	[36] [53]
Fuzzy Models	Compute trust by fuzzy logic rules.	Need domain experts for doing parameter tuning and defining fuzzy rules	[54] [55] [56] [57] [40] [58]
Flow Models	Trust evaluation task is handled using network flow model.	Model is more compact and general and provides integral optimum solutions.	[59] [60] [61]
ACO (Ant Colony Optimization)	It introduces the pheromone concept and transition probability for dynamic trust representation.	Suitable for distributed feedback systems like cloud computing. Takes change of time and interaction frequency into account for direct trust evaluation.	[61] [62]

ecosystems. As shown in Fig. 5.1, the framework provides the following broad functionalities:

- Monitor and evaluate the risk profile of the organization and then developing strategies addressing the key areas of vulnerability.
- Improvise by performing remediation activities to enhance the developed strategies.
- Obtain certification and compliance from the third-party for assuring the security, trustworthiness, and auditability of the organization's cloud ecosystem.

The authors describe the six key dimensions that serve as a blueprint for the making of the trusted cloud systems [3]. The key dimensions also line up with the CSA's 'Cloud Control Matrix' and help in the understanding of the trust cloud characteristics. The six dimensions, as shown in Fig.5.1, are described below:

(a) Organizational: Organizations' internal users and CSP staff can introduce risk in the cloud ecosystem if the CSPs fail to manage organizational roles and human resources that deal with the challenges that emerge in a cloud ecosystem.
(b) Technology: With the right technical configurations of CSPs in place, like encryption, key management, access management, underlying infrastructure, vulnerability management, virtualization management, API security etc. can make a huge difference in building a trusted cloud ecosystem.
(c) Data: Maintaining the organization's data assets at different geographic locations puts challenges on

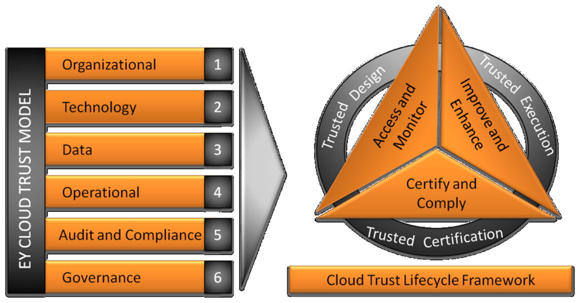


Fig. 5.1: EY Cloud Trust Model and Lifecycle Framework (after [3])

data protection. For adequate protection of data assets, organizations are required to know about their owned assets and the value that these assets hold.

(d) Operational: Moving IT operations to the CSP has an impact on those operations. There is a need of IT operation management system that controls the physical and environmental risks, provides backup and recovery plans, improved efficiency, and data security etc.

(e) Audit and compliance: CSPs need to have robust compliance procedures, audit plans and a third party assurance report at the minimum.

(f) Governance: A well-established governance model should be in place to design scalable programs, manage compliance and risks and monitor performance.

The authors conclude that by utilizing the 'EY Cloud Trust model' build upon six key cloud dimensions, trusted cloud ecosystems can be built. There has to be a balance between the risks and the value that the CSPs impart to the business. This trust model involves both preventive and detective measures for building trust.

III. *Hierarchical Trust Model*

In the paper [66], the authors presented a 'Trust Model for Measuring the Security Strength of Cloud Computing Services' that evaluates the cloud trust value based on the identified parameters relevant to security aspects. Both the static trust and the dynamic trust is evaluated to determine cloud service security. The evaluated trust value represents the overall strength of the cloud service security. Trust values identified include Identity Management System (IDM), Authentication, Authorization, Data Protection, Confidentiality, Communication, Isolation, Virtualization, and Compliance. The detailed description of the parameters is given in [66]. The trust model is indicated in the form of a hierarchical tree structure, where the root indicates the trust value and the child nodes represent the parameters with varying weights at different levels. The root node or the final trust value is the weighted sum of all the vector parameters. This model is based on the preventive approach of establishing trust.

Trust Factor Taxonomy:

In the article [16], the authors Alhanahnah et al. present a taxonomy of trust factors for trust evaluation that aims at helping cloud customers to choose trustworthy cloud service providers. This paper describes the notion of trust in cloud computing as discussed in Sec. 2. and also highlights some of the significant cloud trust concerns that demand rapid solutions for establishing and maintaining trust. Authors describe the two conceptual trust phases: 1) Establishing trust and 2) Maintaining trust. The establishing trust phase identifies the trust factors that evaluate the CSP trustworthiness. The trust factors are categorized as: 1) 'SLA' Trust Factors (TFs) and 2) 'Non-SLA' Trust Factors; as shown in Table.5.1. Briefly stated, SLA TFs are the trust properties extracted from the Service Level Agreements (SLA) [67] [68] [69] [70], whereas the non-SLA TFs are the trust properties derived from various other sources. The establishing trust phase generates a pretrust value which is then re-evaluated iteratively by the maintaining trust phase that follows; to keep the trust value updated. The maintaining trust phase monitors allegiance of service

Table 5.1: Taxonomy of Trust Factors (TFs)

SLA TFs	Non SLA TFs
Performance Security Data Management Personal Data Protection(PDP) Cost	CSCs Feedback Experts Opinion Financial Status

provider with the SLA which is significant for dynamic trust evaluation.

IV. *Trust Model in Cloud of Things Environment*
In the cloud of things environment, an autonomic system for trust management is difficult to be realized because there is a lack of control due to large scalability of deployment, node mobility, limited computing capacity [10] [71]. Therefore, in such scenarios, the trust manager must be flexible and adapt to the dynamic circumstances posed by the network. Namal et al. [4] present an autonomic trust framework for managing trust in Cloud-based IOT Applications called MAPE-K. MAPE-K is the feedback loop which is based upon 'Monitor', 'Analyse', 'Plan', 'Execute' and, 'Knowledge' for evaluation of trust levels in IoT-cloud systems. The distributed trust agents in the framework filter the required trust information out of rest of the information to support autonomic trust decision-making. The monitor component filters and aggregates the information to detect any symptom which requires analysis. The analyze component carries out data analysis on the pre-detected symptoms from the monitor component. If any modifications are needed to achieve some targeted objectives, a change requisition is sent to the plan component that plans the actions and formulates the procedures needed to achieve the desired objectives. Execution phase actually brings the necessary modifications in the system behavior using effectors, as recommended by the plan component. The knowledge (K) includes the data associated with all the MAPE components that is centrally shared among all the trust agents in the network employing cloud infrastructure to serve decision-making. The knowledge data includes trust values, symptoms, context information, metrics, topology information, historical logs, and policies etc.

V. *Other Models*
M.Alhamad et.al. [40], presented a fuzzy logic based scheme for evaluating trust in cloud applications enabling cloud users to evaluate the trustworthiness of CSPs when migrating their operations to cloud data centers. The evaluation method uses four input factors: security, scalability, usability, and availability. The input parameters are fed to the fuzzy inference system employing Sugeno fuzzy technique with gbell membership function. Neural networks are employed to lessen the count of fuzzy rules in the training process to select only the most significant IF-THEN rules. M. Balasubramanian et al. [54] presented a framework to evaluate the trustworthiness of cloud servers and trust authority based on fuzzy approach by evaluating conformity on QoS parameters. Emeakaroha et.al [72] presented a trust-label system developed to communicate trust values in order to boost consumer confidences in Cloud services. Q. Chenhao and Rajkumar Buyya [56] presented a 'Hierarchical Fuzzy Inference based System' for evaluating Cloud Trust for Service Selection that evaluates trust of IaaS cloud based upon the user requirements. The model offers linguistic descriptors for both naïve and expert consumers to submit their vague demands or uncertain resources. The authors generated benchmark results for the 11 dynamic attributes that include CPU Speed, Memory Read, Memory Write, Disk Read, Disk Write, Network In, Network Out, Availability, Failure Rate, VM Startup, and VM Shutdown.
Sherchan et al. [73], proposed a Hidden Markov Model (HMM) based model for predicting trust in service web. It utilizes a time sensitive dynamic model for training pool. Xiaonian Wua et al. [74] proposed a 'D-S evidence theory' based model with sliding windows for evaluating trust in cloud computing. They calculated the direct trust of entities as the first-hand evidence by employing DS combination rules. Trust assesment relies upon the interaction evidence between CSP and CU. Since, the significance and legitimacy of evidence eventually would decay with time, sliding window is employed to delineate the timeliness of

Table 6.1: Comparison of different trust models

Source	Method / Model	Parameters	Observations
R.Shaikh [66]	Hierarchical tree structure based trust model for evaluating the overall strength of security in Cloud Computing Services .	Identified trust values include IDM, Authentication, Authorization, Data Protection, Confidentiality, Communication, Isolation, Virtualization, and Compliance.	Calculates both static and dynamic trust values. Uses weighted sum aggregation technique.
Ryan K. Lo [10]	TrustCloud framework	Preventive and Detective Controls	The model handles accountability issues of cloud systems through policy-based mechanisms.
Namal et al. [4]	An autonomic trust framework for managing trust in Cloud-based IOT Applications called MAPE-K.	The feedback loop which is based on ‘Monitor’, ‘Analyse’, ‘Plan’, ‘Execute’ and, ’Knowledge’ to evaluate trust.	MAPE-K evaluates the trust levels in an IoT cloud ecosystem.
M. Balasubramanian et al. [54]	Fuzzy logic approach	Evaluation based on conformity to QoS parameters.	The cloud server and trust authority parameter values are collected from cloud benchmark service.
M.Alhamad et al. [40]	Fuzzy logic based trust evaluation scheme for cloud applications.	Evaluation parameters include: security, scalability, usability, and availability.	Neural networks are employed to reduce the number of fuzzy rules in the training process to select only the most significant IF-THEN rule.
Xiaonian Wua et al. [74]	D-S evidence theory and sliding windows based trust evaluation model.	Direct trust of entities calculated by DS combination rules. Evaluation is based on interaction evidence between CSP and CU.	Describes the timeliness of the evidence information and takes care of the dynamisim of interaction evidences.
Lin et al. [75], Bedi et al. [76] [77]	Ant colony optimization (ACO) technique	Use ‘pheromone’ to model transition possibility for representing behavior trust.	ACO based trust-recommender system takes change of time and interaction frequency into account for trust evaluation.

the evidence information and it also takes care of the dynamism of interaction evidences.

Lin et al. [75] proposed a cloud trust model that is based on behavior of entities in cloud environment. Since the trust associations among corresponding entities are difficult, dynamic, and mostly unknown; the authors presented an Ant colony optimization (ACO) technique which is based on the conception of ‘pheromone’ to model transition possibility for representing behavior trust. Bedi et al. [76] also proposed an ant colony based trust-recommender system.

6. Discussion and Open Issues. The central aspect of trust management in Cloud environment is to determine how to assess the corresponding trust values of an entity. Various trust components have been specified however, not all possible trust properties should be gathered and stored. Determining the most relevant trust components is significant to obtain the accurate trust predictions. Machine learning approaches could be used to determine the set of properties that provide the most accurate trust predictions. Table 6.1 draws a comparison of different trust models and highlights the trust parameters identified in various studies.

As we explore some open issues in cloud trust management the central problem that has been identified is how to collect and verify trust information. The authenticity of trust information is a major concern and requires further research. Furthermore, different deployment models like public and private cloud models might need different trust management schemes. Trust information can be used as a metric to analyze system state in a particular duration or service context. Trust information can perform other tasks in addition to facilitating service selection. Lowered trust value could be an indication that the CSP is misbehaving due to resource-depletion at peak hours of the service usage. In the following subsection we provide a summary of the identified open issues that are potentially significant roadblocks that hinder cloud trust.

6.1. Open Issues. There is an absence of a reliable trust and reputation model that is standard and specific to the cloud architecture that assists the customers in choosing the trustworthy service providers [40]. Despite the fact that the development of trust and reputation management systems has been widespread and popularly implemented for several online services, there does not exist any such model implemented for cloud computing. Here we highlight some of the major open issues in Cloud Trust Management:

- As described in section 2, while we defined semantics of trust, trust is context-dependent. Hence, it may present some inaccurate information in some intense contextual conditions. There has to be a standard definition for describing trust in cloud scenario.
- The literature has addressed a broad range of SLA-based variables. However, there is no agreement defining and choosing a correct set of trust variables, as most methods are not in line with standardization bodies' norms and guidelines.
- Trust management system needs a uniform mechanism for accumulating multiple trust attributes irrespective of the evaluation procedures employed to evaluate the subjective trust parameters.
- In the complex IoT-based cloud environments, SLAs are inadequate. Often times, the ambiguous clauses and vague technical specifications of SLAs can prevent service consumers from identifying trustworthiness of cloud services.
- In the cloud of things environment, having an autonomic trust management system is difficult to be realized due to node mobility, limited computing capacity, and lack of control [10] [71].

6.2. Datasets for Trust Evaluation. We have identified some datasets that could be employed to evaluate trust related computations applicable to cloud computing setting:

- *Cloud Armor dataset* [78]: It is the real time data containing the trust information obtained from various cloud service providers derived from consumer feedbacks [32].
  [Cloud Armor [79] is a research project aiming to develop a scalable Trust Management System for cloud services, at the University of Adelaide, Australia.]
- *Trust Feedback Dataset* [78]: This dataset is a collection of consumers' feedback of cloud services; based on QoS attributes. The dataset is a collection of 10,000+ feedbacks received from nearly 7,000 service consumers for 113 real-world cloud services.
- *Epinion Dataset* [80]: Epinion is a review website where people post product reviews. A trust network of users is created by adding other users to their ”Web of Trust” whose reviews and ratings they find consistently valuable” and conversely adding users to their ”Block list” whose reviews are found consistently not valuable or inaccurate. The Epinion dataset collected from the website Epinions.com contains about 664,824 reviews and 487,181 issued trust statements from 49,290 users for 139,738 different items.
- *Extended Epinion Dataset* [80]: This dataset also contains distrust lists for items. The dataset consists of 841,372 reviews (717,667 trusts and 123,705 distrusts) received from about 132,000 users.

The datasets described above are merely based on user feedbacks and rely solely on user experiences and preferences. These consumer feedbacks act as Trust Indicators that lead to Trust Formation and assist cloud users to choose a trustworthy cloud service among a pool of available service providers; and also gives cloud users an idea of the services they can trust. However, there is a lack of available data for evaluating cloud services based on their functionalities rather than the user experiences or ratings. Evaluating services for their functionalities provides more transparent, accurate and direct trust metrics for system evaluation. We derived a service evaluation dataset from a standard BOSSBase dataset for trust verification of cloud services, the details of which are introduced in the next section.

7. Proposed Rich Model based Machine Learning solution for Trust Verification. As the advancements in the field of Machine Learning continue to expand, several approaches to solving cloud computing challenges have utilized the opportunities of problem solving using Machine Leaning algorithms [81] [82]. Gulen et al. [83], [84] utilized machine learning algorithms to solve the issue of anomaly detection in Clouds. Khilar et al. [85] put forth an access control model for cloud computing based on trust by employing Machine Learning. Wang et al. [86] designed a machine learning techniques based privacy-preserving framework using feature extraction.

In this study we identified one major direction that would serve as a roadmap for creating trusted cloud environment. As we have seen that cloud-based services have increasingly gained much popularity, yet there is a lack in the tools that allow cloud consumers to verify that these services perform as expected [87] [88] [89]. Dykstra et al. [90] for the first time provided an evaluation model for some cloud-forensic acquisition tools that aids in providing confidences in the acquired evidences. In addition to promising security guarantees these tools should verify functional correctness and performance along with the service availability and reliability. Users would be immensely benefited from knowing as to what extent the CSP delivered the services as promised. More precisely, the agreement between the cloud consumer and the service provider should be verified and not merely be reliant on cloud provider's report. It is quiet important to capture the misbehavior of CSP in terms of functional correctness of the service promised so as to detect failures. As cloud software is delivered by a third party as a Software-as-a-Service, the actual code of implementation is expected to be unavailable to the service users. Thus the utilization of white-box technique strategies (like symbolic execution [91] [92] [93]) cannot be utilized.

In our work we mainly focus on two key areas: A) Trusted Software: To verify whether the right software is running for the correct services. B) Functional correctness: To verify whether the running service is working as it is expected to perform. Here we consider cloud customers falling into two categories, one is the service providers that deploy their services or softwares in the cloud (PaaS or IaaS), and the other is cloud users that utilize a software or cloud storage service.

7.1. Approach for verifying the trusted software. If the software deployed by the cloud service provider is tampered with or replaced by some other low cost version, the requested service being invoked by the user would deviate from the expected behavior and result in trust violation. Hence, for the service providers in order to guarantee that trusted software is running, verifying the trusted software running on cloud nodes on the basis of output generated from the service utilization be facilitated at the consumer end. We propose an approach to verify functional properties of a cloud services based on the validation of results or the output generated after utilizing a particular cloud service; without involving a third party for certification or without the need of accessing the implementation code of the service utilized. As image processing jobs are computation and storage costly services, due to their large scale processing needs, many such requests are processed over the cloud to reduce computational and implementation overheads at the client site.

As a use-case, we consider a scenario where one such service is being invoked by a cloud user. The user requests the service over a set of input images submitted to a cloudlet. The service provider performs the computation over a given set of input data as requested by the user. At this point the service provider may either satisfy the user request by faithfully processing the data utilizing the legitimate software or maliciously employ a low-cost, or some tampered/ unlicensed version of the software. The users would find it difficult to capture the violation in the services delivered unless they have the verification mechanisms at their disposal to verify the legitimacy of the output produced. We have proposed a Machine Learning backed software verification approach based on Spatial Rich Models (SRM) [94] inspired by its wide applicability in steganographic detectors in the domain of digital forensics [95] [96] [97] [98].

Our approach relies on the hypothesis that any data processing service would leave some digital footprints that would enable verification of the software utilized for processing user requests. The goal is to extract distinct features from the processed output returned to the service consumer in order to capture the slight differences in the output generated from the distinct softwares utilized by the service provider. We extract spatial rich features for the output data and investigate the residual noise distributions for capturing any violations in the usage of software-as-a-service. We evaluate these features by the Machine learning model called Ensemble Classifier [99] which constitutes an array of base learners that predicts the output class based on majority voting. The effectiveness of the classifier is evaluated using ROC curve. The detailed algorithms for generating features for evaluation and model training are explained in the Algorithm 1 and 2 respectively.

Algorithm 4: Generate Features for verification (D_n, F)

```
Input: $\boldsymbol{D}_n$ Set of Input data
Output: $\boldsymbol{F}$ Feature Set
    1. User submits $\boldsymbol{n}$ service requests to cloud server:
        I. Cloud server receives $\boldsymbol{D}_I = (\boldsymbol{D}_{i1}, \boldsymbol{D}_{i2}, \boldsymbol{D}_{i3} \dots \boldsymbol{D}_{in})$ input n files to be processed.
    2. Cloud service provider invokes the requested service to process input queries.
        I. Apply the required data processing operations on the input data received $\boldsymbol{D}_I$.
           (At this point the service provider may ideally utilize the genuine software to process the requests
           and faithfully return the output data to the user or may maliciously utilize a low cost version of
           the software for the task.)
    3. Return the output data to the customer after applying the required data processing.
        I. Customer receives the resulting processed data $\boldsymbol{D}_R = (\boldsymbol{D}_{r1}, \boldsymbol{D}_{r2}, \boldsymbol{D}_{r3} \dots \boldsymbol{D}_{rn})$, from the Cloud
           Server.
        II. This $\boldsymbol{D}_R = (\boldsymbol{D}_{r1}, \boldsymbol{D}_{r2}, \boldsymbol{D}_{r3} \dots \boldsymbol{D}_{rn})$, will be used to verify the cloud service utillized.
    4. Extract spatial rich features in the output data $\mathbf{D}_R$ by employing Spatial Rich Models (SRM).
        I. Extract the discrete rich models that are constructed from the neighbouring noise residual samples
           in the spatial domain.
        II. Extract all the 106 rich spatial submodels from the output data returned by the service provider;
            for each data item in $\boldsymbol{D}_R$.
        III. The extracted submodels ($\boldsymbol{F}$) have the dimensionality of 34671, for each $\boldsymbol{R}$ in $\boldsymbol{D}_R$.
        IV. Spatial rich features $\boldsymbol{F} = (\boldsymbol{F}_1, \boldsymbol{F}_2, \boldsymbol{F}_3 \dots \boldsymbol{F}_n)$, are analyzed to verify the legitimacy of the cloud
            service utilized.
```

Algorithm 5: Evaluating Features for Software Verification (F, C_p)

```
Input: $\boldsymbol{F}$ Feature Set
Output: $\boldsymbol{C}_p$ Predicted Class
    1. Train the Machine learning model for Feature Evaluation.
        I. Train the verification model on random samples $(\boldsymbol{F})$ of the features obtained from pre-processed
           data of licensed and unlicensed versions, using cross validation method.
        II. Decision is carried out by majority voting by employing Ensemble classifier of $\boldsymbol{L}$ binary base
            learners.
    2. Verify the cloud service utilized by the customer
        I. Customer verifies the data $\boldsymbol{D}_R$ received from the Cloud Server for its legitimacy using the trained
           machine learning classification model.
        II. The area-under-curve (AUC) in the ROC curve gives the classification prediction accuracy of the
            model.
```

7.2. Experimental Setup. For the usecase described in Section 7, we utilized SRM for feature extraction as it has been widely employed to extract spatial features from neighboring pixels to capture slight variations in the spatial domain. The process involves assembling many submodels derived from the noise distributions of neighboring samples of image residuals. The assembling of submodels is made a part of the machine learning training process that is driven by corresponding Matlab and Octave processed samples. By capturing the variations in the spatial domain, after the data undergoes processing using different softwares, we were able to utilize the power of rich models for verifying trust in cloud services. Another important aspect of employing SRM as a feature extractor is its ability to be used as a general-purpose model for digital forensics as it is independent of the data content [94].

As a feature classifier, and for assembling individual models, we utilized Ensemble classifier. As SRM features have a high dimentionality and we are processing large training datasets, the random forest of Ensemble classifiers is more suitable due to its low computational complexity and its efficiency for high-dimensional features involving big training datasets. Using Ensemble as a classifier also solves the problem of over-fitting when dealing with large training samples; which could have been a case if we used SVM classifier instead. The Ensemble classifier is based on an array of individual base learners that arrive at a consensus by majority voting. Every individual learner is trained on a random feature-set and acts as an independent classifier. By evaluating how various spatial submodels involve in trust verification of cloud services, trust violation is detected.

Our proposed model works in two steps: A) Extracting features for model training and service verification using SRM. B) Evaluating rich features for service verification using Ensemble classifier. First off we generate service utilization datasets that act as samples to pre-train our proposed model. The datasets were created by performing image processing operations on a standard BOSSBase dataset. To verify the effectiveness of our model we invoked image processing services using a legitimate software(Matlab) and also using is low-cost version (Octave) for capturing variations in spatial domain. After deriving the required datasets for Matlab and Octave, we extracted rich features for the derived datasets using SRM. A total of 106 sub-models are extracted, each having an average dimentionality of 327, thereby making a total dimentionality of 34671 for a single feature set against each data file. Then, we trained random forest of Ensemble classifiers using ten-fold cross validation on the extracted rich feature-sets.

Once our model is trained on sufficient feature-sets, it is utilized for trust evaluation of services utilized. When a cloud user submits a request along with the input data to a cloud server invoking a service, the service provider responds with the requested output. Whether the service provider has utilized the legitimate software for processing the user request, is verified by evaluating the rich features of the output produced.

Due to the lack of availability of Trust-datasets that evaluate direct trust values in cloud services, we derived a new trust dataset for direct trust evaluation. The datasets described in section 6.2 include user feedbacks and reviews, hence these can be utilized to evaluate indirect trust that does not add to increased transparency in cloud service monitoring.

7.3. Results and Discussion. As a usecase we consider an image processing task being invoked by the customer wherein the user requests for Edge Detection algorithm operated over a set of input images using Matlab software. We extract two streams of SRM features, one corresponding to a set of images processed by Matlab and another using an unlicensed low cost version Octave. We have explained the details of the experiments in [100] and [101]. We utilized the standard BOSSBase dataset having 10,000 images, for our experimentation. The prediction accuracy of the trained Ensemble classifier is calculated by out-of-bag (OOB) error. Figure 7.1. a) shows the histogram of base learners and Fig 7.1. b) shows the variation in OOB estimate with variation in the count of base learners. The count of base learners, at which the OOB saturates, is fixed. Fig 7.2. shows the area-under-curve plot of ROC curve that gives the true positive rate (recall) vs. false positive rate (fall-out). The results suggest that spatial rich features give a prediction accuracy of 99% with an average testing error of 0.0040 (+/- 0.0009) calculated over 10 splits. This high prediction accuracy is accredited to the large number of diverse submodels involved in extracting rich features using SRM, along with all the varying combinations of submodels and their quantization levels. These individual models capture large number of relationships among neighboring pixels in the data and thus contribute to high prediction accuracy.

Although, Rich Models have not been utilized for evaluating cloud service trust verification, but we have compared its prediction accuracy with related domains that utilize SRM. Jian et.al [102] utilized local residual descriptors for predicting recapture effect in images and achieved an accuracy of 96.72% with three submodels and an accuracy of 98.43% with ten submodels of SRM. Han et.al [95] utilized Rich models for detecting image manipulations. They adopted two-stream R-CNN network to predict if the image has been manipulated or not and achieved a prediction accuracy of 93.7% on NIST16 dataset. In another similar work Cozzolino et.al. [103] utilized SRM for image forgery detection. They extracted a single model SRM feature-set using CNN and employed SVM classifier for forgery detection. The prediction accuracy recorded was 84% to 99% for different image manipulations.

These results suggest that Rich Models are powerful feature extractors that have a great potential in the domain of image forensics and we have successfully utilized its potential for verifying cloud services for functional correctness. The proposed approach of feature generation from the output data for verifying the cloud trust

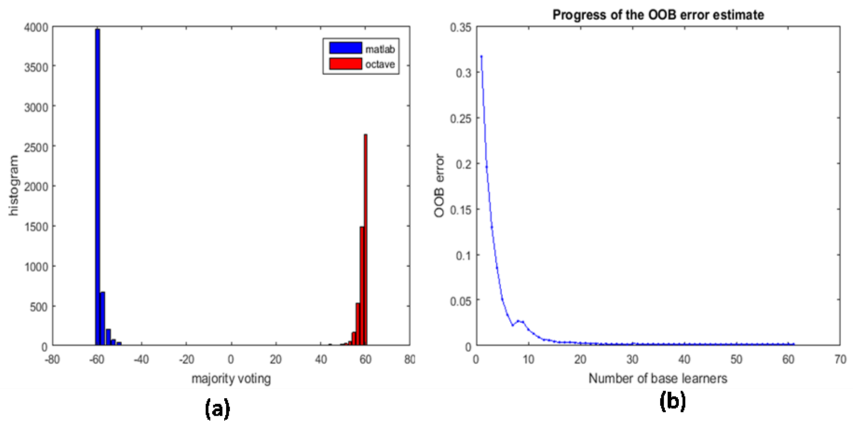


Fig. 7.1: a) Majority voting histogram b) Change of OOB error with change in base learner count.

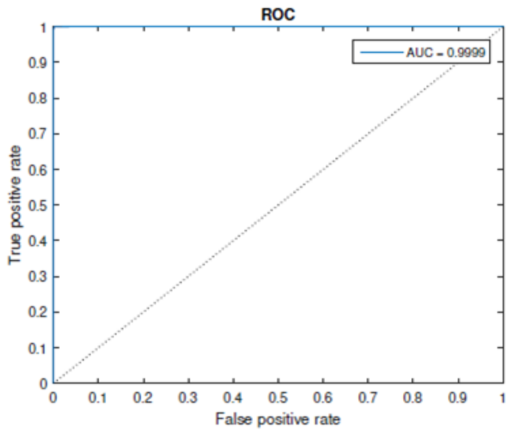


Fig. 7.2: Classification accuracy ROC estimate.

in delivering services as expected can be perceived as an initiative towards trust evaluation in cloud services employing Machine learning techniques.

8. Conclusion. Cloud computing being an opportune technology, but inadequate trust management is hindering its progress. Despite significant efforts to mitigate trust issues, the lack of control and fear of change inhibits individuals and organizations to adopt cloud computing. Even though the legal approaches have been laid down for cloud trust assurance, they continue to be insufficient on their own. Cloud service users still have to rely on cloud providers' promises to provide the desired services. Rather than obliging consumers to rely on providers' genuine behavior, cloud services should employ a standard trust management system so that the users could access and predict accurate trust information. In the literature, the detailed review discussing the issue of trust management is very rare. We presented a review of trust management in the cloud context and highlighted some open issues and possible research domains to uncover the trust issues in cloud computing scenario. A few open questions have been identified. The central issue is to determine the most relevant trust properties to predict accurate trust values and to identify the most effective method to aggregate multiple trust variables to obtain the final trust value. One open issue is to reach for a trade-off between a centralized and distributed trust propagation technique. A potential solution is to implement a hierarchical approach by combining both the approaches. We proposed a Rich model based Machine Learning backed solution to verify the software-as-a-service utilized by the cloud consumer. We performed the experimentation on the standard dataset to evaluate the effectiveness of our scheme to verify the utilization of spatial noise residuals as features for classifying trusted services. Nevertheless, there are several open questions pertinent to trust management in cloud computing that need to be explored further.

REFERENCES

[1] N. Santos, R. Rodrigues, K. P. Gummadi, S. Saroiu, Policy-Sealed Data: A New Abstraction for Building Trusted Cloud Services, Tech. rep.

[2] P. Mell, T. Grance, The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology, Nist Spec. Publ. 145 (2011) 7. arXiv:2305-0543, doi:10.1136/emj.2010.096966.

[3] EY, Building trust in the cloud Creating confidence in your cloud ecosystem, Insights governance, risk compliance (June) (2014).

[4] S. Namal, H. Gamaarachchi, G. M. Lee, T. W. Um, Autonomic trust management in cloud-based and highly dynamic IoT applications, in: Proc. 2015 ITU Kaleidosc. Trust Inf. Soc. K-2015 - Acad. Conf., IEEE, 2016, pp. 1–8. doi:10.1109/Kaleidoscope.2015.7383635.

[5] K. M. Khan, Q. Malluhi, Establishing Trust in Cloud Computing, IT Prof. 12 (5) (2010) 20–27. doi:10.1109/MITP.2010.128.

[6] M. S. Khan, M. R. Warsi, S. Islam, Trust Management Issues in Cloud Computing Ecosystems, in: Elsevier SSRN Ser., Elsevier, 2019, pp. 2233–2238.

[7] T. Lynn, P. Healy, R. McClatchey, J. Morrison, C. Pahl, B. Lee, The Case for Cloud Service Trustmarks and Assurance-as-a-Service, arXiv Prepr. (feb 2014). arXiv:1402.5770.

[8] M. Tang, X. Dai, J. Liu, J. Chen, Towards a trust evaluation middleware for cloud service selection, Futur. Gener. Comput. Syst. 74 (2017) 302–312. doi:10.1016/j.future.2016.01.009.

[9] N. Gonzalez, C. Miers, F. Red, M. Simpl, Open Access A quantitative analysis of current security concerns and solutions for cloud computing (2012) 1–18.

[10] R. K. Ko, P. Jagadpramana, M. Mowbray, S. Pearson, M. Kirchberg, Q. Liang, B. S. Lee, TrustCloud: A framework for accountability and trust in cloud computing, in: Proc. - 2011 IEEE World Congr. Serv. Serv. 2011, 2011, pp. 584–588. doi:10.1109/SERVICES.2011.91.

[11] J. B. Michael, L. T. Gaines, Trust Management in Distributed Databases, in: Computer (Long. Beach. Calif)., Vol. 40, IEEE Computer Society Press, 2005, pp. 329–337. doi:10.1007/0-306-47008-x_29.

[12] T. S. Dybedokken, Trust Management in Fog Computing, Futur. Gener. Comput. Syst. 5 (June) (2017) 15619–15629. doi:10.1109/ACCESS.2017.2733225.

[13] D. M. Rousseau, S. B. Sitkin, R. S. Burt, C. Camerer, Not so different after all: A cross-discipline view of trust, Acad. Manag. Rev. 23 (3) (1998) 393–404. doi:10.5465/AMR.1998.926617.

[14] A. Jøsang, R. Ismail, The beta reputation system, 15th Bled Electron. Commer. Conf. (2002) 2502–2511doi:10.1.1.60.5461.

[15] A. Jøsang, C. Keser, T. Dimitrakos, Can we manage trust?, in: Lect. Notes Comput. Sci., Vol. 3477, Springer, Berlin, Heidelberg, 2005, pp. 93–107. doi:10.1007/11429760_7.

[16] M. Alhanahnah, P. Bertok, Z. Tari, Trusting cloud service providers: Trust phases and a taxonomy of trust factors, IEEE Cloud Comput. 4 (1) (2017) 44–54. doi:10.1109/MCC.2017.20.

[17] P. Massa, K. Souren, Trustlet, open research on trust metrics, in: CEUR Workshop Proc., Vol. 333, 2008, pp. 31–44.

[18] Fujitsu, Personal data in the cloud: (2010).
URL `https://www.fujitsu.com/ie/imagesgig5/fujitsu`$_{p}ersonal-data-in-the-cloud.pdf$

[19] Service Level Agreement - Amazon Simple Storage Service (S3).pdf.
URL `https://aws.amazon.com/s3/sla/`

[20] Amazon EC2 Service Level Agreement – Amazon Web Services.
URL `https://aws.amazon.com/ec2/sla/historical/`

[21] Service Level Agreements - Home | Microsoft Azure.
URL `https://azure.microsoft.com/en-in/support/legal/sla/`

[22] Cloud Security Alliance, CSA - Top Threaths to Cloud Computing v1.0, Security (March) (2010) 1–14.
URL `https://ioactive.com/wp-content/uploads/2018/05/csathreats.v1.0-1.pdf`

[23] J. Schiffman, T. Moyer, H. Vijayakumar, T. Jaeger, P. McDaniel, Seeding clouds with trust anchors, in: Proc. ACM Conf. Comput. Commun. Secur., ACM Press, New York, New York, USA, 2010, pp. 43–48. doi:10.1145/1866835.1866843.

[24] M. Abomhara, G. M. Kien, Cyber Security and the Internet of Things: Vulnerabilities, Threats, Intruders and Attacks, J. Cyber Secur. Mobil. 4 (1) (2015) 65–88. doi:10.13052/jcsm2245-1439.414.

[25] J. Harauz, B. P. Lori M. Kaufman, Data Security in the World of Cloud Computing, IEEE Secur. Priv. (2009) 61–64.

[26] T. H. Noor, Q. Z. Sheng, Z. Maamar, S. Zeadally, Managing Trust in the Cloud: State of the Art and Research Challenges, IEEE Comput. 49 (2) (2016) 34–45. doi:10.1109/MC.2016.57.

[27] M. Blaze, J. Feigenbaum, J. Lacy, Decentralized trust management, in: Proc. 1996 IEEE Symp. Secur. Priv., 1996, pp. 164–173. doi:10.1109/SECPRI.1996.502679.

[28] C. V. L. Mendoza, J. H. Kleinschmidt, Mitigating on-off attacks in the internet of things using a distributed trust management scheme, Int. J. Distrib. Sens. Networks 2015 (2015). doi:10.1155/2015/859731.

[29] L. Gu, J. Wang, B. Sun, Trust management mechanism for Internet of Things, China Commun. 11 (2) (2014) 148–156. doi:10.1109/CC.2014.6821746.

[30] I.-R. Chen, F. Bao, J. Guo, Trust-Based Service Management for Social Internet of Things Systems, IEEE Trans. Dependable Secur. Comput. 13 (6) (2016) 684–696. doi:10.1109/TDSC.2015.2420552.
[31] M. Alruwaythi, K. Kambhampaty, K. E. Nygard, User Behavior and Trust Evaluation in Cloud Computing 58 (2019) 378–368. doi:10.29007/q5bd.
[32] Cloud Armor Project Website - About.
URL `https://cs.adelaide.edu.au/ cloudarmor/research.html`
[33] M. Chiregi, N. J. Navimipour, A comprehensive study of the trust evaluation mechanisms in the cloud computing, J. Serv. Sci. Res. 9 (1) (2017) 1–30. doi:10.1007/s12927-017-0001-7.
[34] R. Marty, Cloud application logging for forensics, 2011, p. 178. doi:10.1145/1982185.1982226.
[35] J. R. Jain, A. Asaduzzaman, A Novel Data Logging Framework to Enhance Security of Cloud Computing (2016).
[36] J. Guo, I. R. Chen, J. J. Tsai, A survey of trust computation models for service management in internet of things systems, Comput. Commun. 97 (2017) 1–14. doi:10.1016/j.comcom.2016.10.012.
[37] H. Kim, H. Lee, W. Kim, Y. Kim, A trust evaluation model for QoS guarantee in cloud systems, Int. J. Grid Distrib. ... 3 (1) (2010) 1–10.
URL `http://www.sersc.org/journals/IJGDC/vol3_no1/1.pdf`
[38] Y. Wang, J. Wen, X. Wang, B. Tao, W. Zhou, A cloud service trust evaluation model based on combining weights and gray correlation analysis, Secur. Commun. Networks 2019 (2019). doi:10.1155/2019/2437062.
[39] Y. Chen, R. H. Katz, What ' s New About Cloud Computing Security ? (2010).
[40] M. Alhamad, T. Dillon, E. Chang, A Trust-Evaluation Metric for Cloud applications, Int. J. Mach. Learn. Comput. 1 (4) (2013) 416–421. doi:10.7763/ijmlc.2011.v1.62.
[41] Cloud Controls Matrix | Cloud Security Alliance.
URL `https://cloudsecurityalliance.org/working-groups/cloud-controls-matrix/#_overview`
[42] E. Commission, The European Cloud Initiative | Digital Single Market (2018).
URL `https://ec.europa.eu/digital-single-market/en/ european-cloud-initiative`
[43] Ebay, Feedback.
URL `https://pages.ebay.com/services/forum/feedback.html`
[44] B. Rietjens, Trust and reputation on eBay: Towards a legal framework for feedback intermediaries, Inf. Commun. Technol. Law 15 (1) (2006) 55–78. doi:10.1080/13600830600557935.
URL `http://www.tandfonline.com/action/journalInformation?journalCode=cict20`
[45] P. Y. Sun, Building Trust in Online Rating Systems through Signal Modeling Online Feedback - based Rating Systems, Strategy.
[46] Shopping, Shopping Online at Shopping.com | Price Comparison Site.
URL `http://epinion.com/?sb=1`
[47] Y. Pan, S. Ding, W. Fan, J. Li, S. Yang, Trust-enhanced cloud service selection model based on QoS analysis, PLoS One 10 (11) (2015) 1–14. doi:10.1371/journal.pone.0143448.
[48] V. Shmatikov, C. Talcott, Reputation-based trust management, J. Comput. Secur. 13 (1) (2005) 167–190. doi:10.3233/JCS-2005-13107.
[49] D. Marudhadevi, V. N. Dhatchayani, V. S. S. Sriram, A Trust Evaluation Model for Cloud Computing Using Service Level Agreement, Comput. J. 58 (10) (2014) 2225–2232. doi:10.1093/comjnl/bxu129.
[50] H. T. Nguyen, W. Zhao, J. Yang, A trust and reputation model based on Bayesian network for web services, in: ICWS 2010 - 2010 IEEE 8th Int. Conf. Web Serv., IEEE, 2010, pp. 251–258. doi:10.1109/ICWS.2010.36.
[51] L. Wang, X. Li, X. Yan, S. Qing, Y. Chen, Service Dynamic Trust Evaluation Model based on Bayesian Network in Distributed Computing Environment 9 (5) (2015) 31–42.
[52] Y. Wang, Y.-c. Lu, I.-r. Chen, J.-h. Cho, A. Swami, C.-t. Lu, LogitTrust : A Logit Regression-based Trust Model for Mobile Ad Hoc Networks State of the Art, Proc. 6th ASE Int. Conf. Privacy, Secur. Risk Trust (PASSAT '14) (2014).
[53] A. Jøsang, A Logic for Uncertain Probabilities, Int. J. Uncertainty, Fuzziness Knowledge-Based Syst. 09 (03) (2001) 279–311.
[54] M. Balasubramanian, H. Kim, Trust evaluation scheme for cloud data security using fuzzy based approach, Int. J. Appl. Eng. Res. 12 (13) (2017) 3908–3913.
[55] X. Anita, M. A. Bhagyaveni, J. M. L. Manickam, Fuzzy-Based trust prediction model for routing in WSNs, Sci. World J. 2014 (iii) (2014). doi:10.1155/2014/480202.
[56] C. Qu, R. Buyya, A cloud trust evaluation system using hierarchical fuzzy inference system for service selection, in: Proc. - Int. Conf. Adv. Inf. Netw. Appl. AINA, no. May, 2014, pp. 850–857. doi:10.1109/AINA.2014.104.
[57] P. N. Mahalle, P. A. Thakre, N. R. Prasad, R. Prasad, A fuzzy approach to trust based access control in internet of things, in: 2013 3rd Int. Conf. Wirel. Commun. Veh. Technol. Inf. Theory Aerosp. Electron. Syst. VITAE 2013 - Co-located with Glob. Wirel. Summit 2013, IEEE, 2013, pp. 1–5. doi:10.1109/VITAE.2013.6617083.
[58] J. Du, X. Li, Adaptive and attribute-based trust model for service-level agreement guarantee in cloud computing, IET Inf. Secur. 7 (1) (2013) 39–50. doi:10.1049/iet-ifs.2012.0232.
[59] C. N. Ziegler, G. Lausen, Spreading activation models for trust propagation, in: Proc. - 2004 IEEE Int. Conf. e-Technology, e-Commerce e-Service, EEE 2004, IEEE, 2004, pp. 83–97. doi:10.1109/EEE.2004.1287293.
[60] S. Brin, L. Page, The PageRank Citation Ranking: Bringing Order to the Web, BMC Syst. Biol. 4 Suppl 2 (1999-66) (2010) S13. arXiv:1111.4503,
[61] W. Jiang, J. Wu, F. Li, G. Wang, H. Zheng, Trust evaluation in online social networks using generalized network flow, IEEE Trans. Comput. 65 (3) (2016) 952–963. doi:10.1109/TC.2015.2435785.
[62] A. Jøsang, A. Jøsang, S. J. Knapskog, A Metric for Trusted Systems, Proc. 21ST Natl. Secur. Conf. NSA (1998).
URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.38.8631`

[63] S. Mei, Z. Wang, Y. Cheng, J. Ren, J. Wu, J. Zhou, Trusted Bytecode Virtual Machine Module: A Novel Method for Dynamic Remote Attestation in Cloud Computing, Int. J. Comput. Intell. Syst. 5 (5) (2012) 924–932. doi:10.1080/18756891.2012.733231.

[64] W. Wang, G. Zeng, D. Tang, J. Yao, Cloud-DLS: Dynamic trusted scheduling for Cloud computing, Expert Syst. Appl. 39 (3) (2012) 2321–2329. doi:10.1016/j.eswa.2011.08.048.

[65] A. Barsoum, A. Hasan, Enabling Dynamic Data and Indirect Mutual Trust for Cloud Computing Storage Systems, IEEE Trans. Parallel Distrib. Syst. 24 (12) (2013) 2375–2385. doi:10.1109/TPDS.2012.337.

[66] R. Shaikh, M. Sasikumar, Trust model for measuring security strength of cloud computing service, in: Procedia Comput. Sci., Vol. 45, Elsevier Masson SAS, 2015, pp. 380–389. doi:10.1016/j.procs.2015.03.165.

[67] M. B. Chhetri, Q. B. Vo, R. Kowalczyk, Policy-based automation of SLA establishment for cloud computing services, in: Proc. - 12th IEEE/ACM Int. Symp. Clust. Cloud Grid Comput. CCGrid 2012, IEEE, 2012, pp. 164–171. doi:10.1109/CCGrid.2012.116.

[68] M. Macías, J. Guitart, Client classification policies for SLA enforcement in shared cloud datacenters, in: Proc. - 12th IEEE/ACM Int. Symp. Clust. Cloud Grid Comput. CCGrid 2012, 2012, pp. 156–163. doi:10.1109/CCGrid.2012.15.

[69] L. Malrait, S. Bouchenak, N. Marchand, Experience with CONSER: A system for server control through fluid modeling, IEEE Trans. Comput. 60 (7) (2011) 951–963. doi:10.1109/TC.2010.164.

[70] V. C. Emeakaroha, M. A. Netto, R. N. Calheiros, I. Brandic, R. Buyya, C. A. De Rose, Towards autonomic detection of SLA violations in Cloud infrastructures, Futur. Gener. Comput. Syst. 28 (7) (2012) 1017–1029. doi:10.1016/j.future.2011.08.018.

[71] S. Pearson, A. Benameur, Privacy, security and trust issues arising from cloud computing, in: Proc. - 2nd IEEE Int. Conf. Cloud Comput. Technol. Sci. CloudCom 2010, IEEE, Pearson2010, 2010, pp. 693–702. doi:10.1109/CloudCom.2010.66.

[72] V. C. Emeakaroha, K. Fatema, L. Van Der Werff, P. Healy, T. Lynn, J. P. Morrison, A Trust Label System for Communicating Trust in Cloud Services, IEEE Trans. Serv. Comput. 10 (5) (2017) 689–700. doi:10.1109/TSC.2016.2553036.

[73] W. Sherchan, S. Nepal, A. Bouguettaya, A trust prediction model for service web, Proc. 10th IEEE Int. Conf. Trust. Secur. Priv. Comput. Commun. Trust. 2011, 8th IEEE Int. Conf. Embed. Softw. Syst. ICESS 2011, 6th Int. Conf. FCST 2011 (2011) 258–265doi:10.1109/TrustCom.2011.35.

[74] X. Wu, R. Zhang, B. Zeng, S. Zhou, A trust evaluation model for cloud computing, in: Procedia Comput. Sci., Vol. 17, 2013, pp. 1170–1177. doi:10.1016/j.procs.2013.05.149.

[75] G. Lin, Y. Bie, M. Lei, K. Zheng, ACO-BTM: A Behavior Trust Model in Cloud Computing Environment, Int. J. Comput. Intell. Syst. 7 (4) (2014) 785–795. doi:10.1080/18756891.2013.864479.

[76] P. Bedi, R. Sharma, Trust based recommender system using ant colony for trust computation, Expert Syst. Appl. 39 (1) (2012) 1183–1190. doi:10.1016/j.eswa.2011.07.124.

[77] J. Bharath, V. S. S. Sriram, Genetically Modified Ant Colony Optimization based Trust Evaluation in Cloud Computing 9 (December) (2016). doi:10.17485/ijst/2016/v9i48/107967.

[78] Cloud Armor Project Website - Dataset.
URL `https://cs.adelaide.edu.au/ cloudarmor/ds.html`

[79] T. H. NOOR, S. QUAN Z., S. ZEADALLY, Y. U. JIAN, Trust Management of Services in Cloud Environments: Obstacles and Solutions., ACM Comput. Surv. 46 (1) (2013) 12 – 12:30. doi:10.1145/2522968.2522980.

[80] Epinions trust network dataset – {KONECT} (2017).
URL `http://www.trustlet.org/epinions.html http://konect.uni-koblenz.de/networks/epinions`

[81] J. Fiala, A Survey of Machine Learning Applications to Cloud Computing (2015) 1–26.
URL `http://www.cse.wustl.edu/ jain/cse570-15/ftp/cld_ml.pdf`

[82] D. Pop, Machine Learning and Cloud Computing: Survey of Distributed and SaaS Solutions (2016). arXiv:1603.08767.
URL `http://arxiv.org/abs/1603.08767`

[83] A. Gulenko, M. Wallschlager, F. Schmidt, O. Kao, F. Liu, Evaluating machine learning algorithms for anomaly detection in clouds, Proc. - 2016 IEEE Int. Conf. Big Data, Big Data 2016 (2016) 2716–2721doi:10.1109/BigData.2016.7840917.

[84] T. Salman, D. Bhamare, A. Erbad, R. Jain, M. Samaka, Machine Learning for Anomaly Detection and Categorization in Multi-Cloud Environments, Proc. - 4th IEEE Int. Conf. Cyber Secur. Cloud Comput. CSCloud 2017 3rd IEEE Int. Conf. Scalable Smart Cloud, SSC 2017 (August) (2017) 97–103. doi:10.1109/CSCloud.2017.15.

[85] P. Khilar, V. Chaudhari, R. Swain, Trust-Based Access Control in Cloud Computing Using Machine Learning: Intelligent Edge, Fog and Mist Computing, 2019, pp. 55–79. doi:10.1007/978-3-030-03359-0_3.

[86] J. Wang, S. Hu, Q. Wang, Y. Ma, Privacy-preserving outsourced feature extractions in the cloud: A survey, IEEE Netw. 31 (5) (2017) 36–41. doi:10.1109/MNET.2017.1600240.

[87] S. Bouchenak, G. Chockler, H. Chockler, G. Gheorghe, N. Santos, A. Shraer, Verifying cloud services, ACM SIGOPS Oper. Syst. Rev. 47 (2) (2013) 6–19. doi:10.1145/2506164.2506167.

[88] G. Grispos, T. Storer, W. B. Glisson, Calm Before the Storm: The Challenges of Cloud Computing in Digital Forensics, Int. J. Digit. Crime Forensics 4 (2) (2014) 28–48. arXiv:1410.2123, doi:10.4018/jdcf.2012040103.

[89] A. Aminnezhad, A. Dehghantanha, M. Taufik Abdullah, M. Damshenas, Cloud Forensics Issues and Opportunities, Int. J. Inf. Process. Manag. 4 (4) (2013) 76–85. doi:10.4156/ijipm.vol4.issue4.9.

[90] J. Dykstra, A. T. Sherman, Acquiring forensic evidence from infrastructure-as-a-service cloud computing: Exploring and evaluating tools, trust, and techniques, Digit. Investig. 9 (SUPPL.) (2012). doi:10.1016/j.diin.2012.05.001.

[91] C. Cadar, D. Dunbar, D. Engler, Klee: Unassisted and automatic generation of high-coverage tests for complex systems programs, in: Proc. 8th USENIX Symp. Oper. Syst. Des. Implementation, OSDI 2008, 2019, pp. 209–224.

[92] J. C. King, A new approach to program testing, in: Proc. 1975 Int. Conf. Reliab. Softw., Association for Computing Machinery, Inc, New York, New York, USA, 1975, pp. 228–233. doi:10.1145/800027.808444.

[93] V. Chipounov, V. Kuznetsov, G. Candea, The S2E platform: Design, implementation, and applications, in: ACM Trans. Comput. Syst., Vol. 30, 2012, pp. 1–49. doi:10.1145/2110356.2110358.

[94] J. Fridrich, J. Kodovsky, Rich models for steganalysis of digital images, IEEE Trans. Inf. Forensics Secur. 7 (3) (2012) 868–882. doi:10.1109/TIFS.2012.2190402.

[95] X. Han, L. S. Davis, Learning Rich Features for Image Manipulation Detection RPN layer RGB stream input RGB RoI features Bilinear Noise stream input Noise Conv Layers Noise RoI features, Cvpr (2018) 1313–1328

[96] D. Cozzolino, G. Poggi, L. Verdoliva, Splicebuster: A new blind image splicing detector, in: 2015 IEEE Int. Work. Inf. Forensics Secur. WIFS 2015 - Proc., IEEE, 2015, pp. 1–6. doi:10.1109/WIFS.2015.7368565.

[97] Y. Rao, J. Ni, A deep learning approach to detection of splicing and copy-move forgeries in images, in: 8th IEEE Int. Work. Inf. Forensics Secur. WIFS 2016, IEEE, 2017, pp. 1–6. doi:10.1109/WIFS.2016.7823911.

[98] R. Zhang, F. Zhu, J. Liu, G. Liu, Efficient feature learning and multi-size image steganalysis based on CNN, Preprint (2) (2018) 1–10. arXiv:1807.11428.
URL `http://arxiv.org/abs/1807.11428`

[99] J. Kodovský, J. Fridrich, V. Holub, Ensemble classifiers for steganalysis of digital media, in: IEEE Trans. Inf. Forensics Secur., Vol. 7, 2012, pp. 432–444. doi:10.1109/TIFS.2011.2175919.

[100] M. Saleem, M. R. Warsi, S. Islam, Learning rich features from software-as-a-service cloud computing for detecting trust violations, in: Lect. Notes Networks Syst., Vol. 116, 2020, pp. 445–452. doi:10.1007/978-981-15-3020-3_38.

[101] M. Saleem, M. R. Warsi, S. Islam, Feature Evaluation for Learning Underlying Data-Processing to Enhance Cloud Trust Through Rich Models, in: ICT Compet. Strateg., 2020, pp. 539–544. doi:10.1201/9781003052098-56.

[102] J. Li, G. Wu, Image recapture detection through residual-based local descriptors and machine learning, Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 10603 LNCS (2017) 653–660. doi:10.1007/978-3-319-68542-7_56.

[103] D. Cozzolino, G. Poggi, L. Verdoliva, Recasting residual-based local descriptors as convolutional neural networks: An application to image forgery detection, in: IH MMSec 2017 - Proc. 2017 ACM Work. Inf. Hiding Multimed. Secur., 2017, pp. 159–164. arXiv:1703.04615, doi:10.1145/3082031.3083247.

Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2021 SCPE. Volume 22, Issues 4, pp. 445–462, DOI 10.12694/scpe.v22i4.1878

ENABLING INTERNET OF THINGS THROUGH SENSOR CLOUD: A REVIEW

JYOTSNA VERMA [*]

Abstract. With the inception of the Internet of Things (IoT), wireless technology found a new outlook where the physical objects can interact with each other and can sense the environment. The IoT has found its way in the real world and has connected billions of devices throughout the world. However, its limitations, such as limited processing capability, storage capability, security and privacy issues, and energy constraints prevent the IoT system to be properly utilized by the real-world applications. Hence, the integration of IoT with various emerging technologies like big data, software defined networks, machine learning, fog computing, sensor cloud, etc., will make the IoT system a more powerful technology. The sensor cloud provides an open, secure, flexible, large storage and a computational capable infrastructure which makes the ensemble architecture of IoT and sensor cloud more efficient. An extensive review of the IoT system enabled sensor cloud is presented in the paper, and with this context, the paper attempts to summarize the sensor cloud infrastructure along with its challenges. In addition, the paper presents the possible integrated architecture of the IoT and the sensor cloud which enables the network to be properly utilized. Further, the importance of integrating these two promising technologies and research challenges associated with it is also identified. Finally, the paper analyses and discusses the motivation behind the ensemble system along with future research direction.

Key words: Sensor cloud architecture, Internet of Things, Cloud computing, WSN.

AMS subject classifications. 68M14

1. Introduction.

The world apparently has been focused on connectivity and convergence since the late 1980s; way back in the twentieth century, machines had no senses; they only had brains capable of understanding what we asked them to do. But this is not the case with the ”Internet of Things,” yet machines and other physical things must sense for themselves [1]. With the proliferation of the IoT we have reached an era where we envision the objects or physical things around us connected to the internet allowing them to send, receive, and exchange data. The advent of the term “Internet of Things” was coined by Kevin Atshon in 1999 and is considered as the world’s third wave of the information industry after the invention of computers and the internet [2]. According to IETF [3] definition “The Internet of Things is the network of physical objects or ”things” embedded with electronics, software, sensors, and connectivity to enable objects to exchange data with the manufacturer, operator, and/or other connected devices”. The physical objects in the Internet of Things embedded with sophisticated sensors for sensing the environment of the real-world scenarios, self-configuring nodes (things), actuators to communicate with other smart devices or nodes, and RFID (Radio Frequency Identification) chips for unique digital identification of the things with which they are connected and are integrated into the network. Each physical object is automatically identified by their unique digital identities which makes them share and exchange data with each other.

By the rapid and increasing growth of IoT, it has marked its place and has made ground in the world of wireless networking. IoT systems are supposed to provide connectivity and intelligence to billions of physical objects and are being widely deployed in different application domains like smart home applications, health care sectors, smart cities, agriculture, industrial automation, wearable, etc. Probably, in the coming years say by 2025 the cumulative installed base of linked Internet of Things (IoT) devices is expected to be 75.44 billion worldwide, a five-fold increase in ten years [4]. The IoT, allowed by the omnipresent internet technology, is the next important step in fulfilling the promise of the internet to make the world connected and hence opens the door for emerging technologies like Big Data, real-time analytics, Software Defined Networks (SDN), machine learning, sensor cloud and many more to come up and integrate with the IoT to maximize its potentials. IoT generally deals with various concerns like limited storage capacity, limited processing capabilities and energy

[*]Department of Computer Science, The ICFAI University, Jaipur, Rajasthan, India. (jyotsna.verma@iujaipur.edu.in).

constraints etc., which ultimately affects the security, reliability, efficiency, and privacy of the physical objects. By integrating with these emerging technologies, the IoT can reach a next level milestone in the near future.

Cloud computing, which is another network paradigm characterized to have virtualized storage, computation, and network resources which makes the cloud infrastructure simple, cheap, scalable, and manageable than physical devices. According to the definition provided by the NIST [5] "Cloud computing is a technological and operational model for ubiquitous, on-demand network access to a shared pool of configurable infrastructure, processing, storage and application services that can be provisioned and released for use with minimal system management effort or service provider interaction". The cloud computing provides three types of services: Software as a service (SaaS) which is the top layer of the cloud infrastructure and it offers applications like Google App Engine, HEROKU, IBM Bluemix, and Microsoft Azure running on the cloud environment, Platform as a service (PaaS) is the middle layer and offers platform layer resources, and the lowest layer of the cloud infrastructure is Infrastructure as a service (IaaS) which offers the computing, storage, and network resources. The economic and technical benefits of cloud computing like virtually unlimited storage, low cost computing capabilities, provision of providing leased virtualized resources on an on-demand basis have gained popularity and attention from the research community and the paradigm has been widely adopted by large companies like Google, IBM, Amazon, Oracle, Microsoft, Facebook etc. The convergence of IoT and cloud computing, the two relatively thriving networking paradigm helps in efficiently collecting and analyzing real-time data [6-9].

The integration of Cloud Computing and IoT solves various issues of IoT constrained devices such as data analysis, computation, data access, storage and is called as Cloud of Things [10] or as CloudIoT [11]. The CloudIoT offers various services such as Things as a service, SenaaS (Sensor as a Service) [12]. Merely integrating IoT with the cloud is not enough to solve the technical limitations of the IoT rather IoT should be integrated with various other technologies like big data, software defined networks, fog computing, sensor cloud, etc., to provide efficient real-world applications. IoT is highly dependent upon sensor based data acquisition and processing of the data. Hence, enabling IoT through cloud sensors helps in the efficient data acquisition, storage, and processing of data in real time. Cloud sensor or sensor cloud is another networking paradigm that gathers all the data or information from the physical sensors deployed in a particular application domain and transfers them to the cloud infrastructure. The ensembling of IoT with sensor cloud opens up new horizons for the data aggregation, data storage, real-time processing of data, and scalability of nodes.

1.1. Contributions. The contributions of this paper are as listed below:

1. Presents a comprehensive literature review of IoT and sensor cloud with their applications, challenges, and architectures.
2. Presents the possible ensemble architecture of the IoT and the sensor cloud.
3. Analysed and discussed the integration of IoT with the sensor cloud and presents the significance and research challenges of two integrated networking paradigms.
4. Finally, the effect of mobility models on network performance with respect to the mobility under Dynamic Source Routing (DSR) protocol is evaluated.
5. Presents the future research directions for the integrated system of IoT and sensor cloud.

1.2. Organization of the paper. The paper primarily focuses on the review of the integration of IoT with sensor cloud and research challenges associated with it. The rest of the paper is organized as follows: Section 2 discusses the sensor network and the cloud. Section 3 presents the architecture of the sensor cloud and Section 4 discusses the sensor virtualization. Section 5 summarizes the challenges and constraints associated with the sensor cloud infrastructure. Section 6 presents the integration of IoT with sensor cloud and Section 7 presents the possible ensemble architecture of the IoT and the sensor cloud. The research challenges associated with the integration of IoT and sensor cloud is presented in Section 8. Section 9 analyses and discusses the integrated architecture of IoT and sensor cloud. Further, future research directions for the integrated system are identified and are presented in Section 10. Finally, Section 11 concludes the paper.

2. Sensor network and the cloud. With the advancements in wireless networking, smart sensing becomes a reality. Sensor network is an infrastructure-less, distributed network of a set of large number of sensor nodes that are deployed in a particular domain. Sensor nodes are light weighted, low power multifunctional tiny devices that have the capability of data computation, communication over the network and sensing the physical parameters. It senses the physical parameters (temperature, humidity, speed etc.) and converts them to electrical or optical signals. This signal helped the sensor network to measure the physical parameters electrically to monitor and control sensors which eventually provides various services like weather forecasting, environment monitoring, healthcare services, agricultural services, military services, government services etc., by using various types of sensors like thermal sensor, body sensor, a seismic sensor, visual sensor and environmental sensors etc. The mobility of a sensor node in the sensor network is not a mandatory requirement and the network is much more scalable and fault tolerant than the ad hoc wireless network.

The activity of sensing a particular application domain in a sensor network can be periodic or sporadic [13]. A periodic sensing senses the application scenario periodically; for instance, environmental factors such as humidity, temperature, pollution, and nuclear radiation etc., can be measured periodically whereas sporadic sensing, senses the particular application domain sporadically, such as border intrusion detection, threshold detection of a furnace, pressure, motion, stress measurement of a building or machinery etc. [13]. The sensor network has two important functions: data dissemination and data aggregation. Data dissemination disseminates or propagate the data throughout the sensor network i.e., the collected information from the sensor nodes is communicated to the base station and the nodes which are interested in seeking that data or information, whereas the data aggregation function aggregates or gathers the sensed data from each sensor node to a sink node. The node from which the data or information is gathered is called as source node and the node that seeks data or information is called as sink node. The endless applications of the sensor network make the sensing reality and are a key component of the Internet of Things. They can coordinate with RFID system for monitoring the status of the physical objects or things, to get the information about the position, temperature, movement of the objects etc. However, there are various limitations pertaining to the wireless sensor network like security, privacy, limited energy and power constraints, mobility, short communication range, processing capabilities, storage capacity, bandwidth availability etc., that is needed to be addressed [14].

The research communities are trying to deal several limitations of sensor networks like energy efficiency, reliability, scalability, robustness etc., at different layers [15]. Moreover the research is more focused on the physical sensor localization [16, 17], sensor node programming [18-20], power consumption [21], sensor data processing [22] and the management and deployment of physical sensors; as the sensor resides in its local domain and it is hard to access them from external servers. The users of the sensor network should also need to know about the status of the sensor nodes, whether the node is faulty or the node disconnects the network, so that the alternative functioning nodes should be used for the proper functioning of the network [23]. These limitations along with the heterogeneity of the sensor nodes create complexity for the engineers. For establishing the communication between the network they have to examine the interface of the sensors and study their protocols; this eventually leads to the integration of sensor networks and the cloud as cloud sensors.

According to the IntelliSys Sensor Cloud is an "infrastructure that allows truly pervasive computation using sensors as an interface between physical and cyber worlds, the data-compute clusters, as the cyber backbone and the internet as the communication medium" [24, 25]. The sensor network and the cloud infrastructure are linked with their respective gateways i.e., sensor gateway and cloud gateway. The sensor gateway collects all the data from the sensor nodes in a compressed form and transfers the collected data to the cloud gateway. The cloud gateway then decompresses the data and stores the data into cloud storage which has sufficiently large storage. The integration of cloud and sensor network solves the problem of data processing and storage capacity of wireless sensor network as cloud infrastructure has huge data storage capacity and data processing capabilities. The cloud sensor also frees the engineer to bother about the heterogeneity of the sensors and offers them to fully focus on the applications and services provided to the users. Traditional sensor network deployed the sensors into the application domain and be able to provide the data for one purpose only and hence, does lots of wastage of resources; as the collected data from the sensor network can be used for a variety of applications. For example, if we consider the application where the sensors deployed for the traffic monitoring has data about the traffic flow in the specific area. Now, this collected data from the sensor can be used for the purpose of satellite navigation, possible route suggestions for the travellers to avoid the congested traffic area etc. The same sensor network can thus be utilized for various applications and be able to provide various services through cloud sensors. This is just one instance where we saw how sensor cloud help to reduce the resource wastages, there are a variety of other applications also where sensor cloud proves their applicability in an efficient manner. Some applications like environmental monitoring [26], health monitoring [27], telematics [25], transportation and vehicular applications [28] can be benefited with such infrastructure.

3. Architecture of Sensor cloud. The infrastructure of sensor cloud has been evolved in recent years. Sensor cloud allows user to easily collect, process, archive, access, share and visualize the sensor data collected from various application domains by using the computational and storage resources of the cloud computing [29]. The infrastructure of cloud computing has been extended, to support and manage the sensors deployed in various application domains. It basically integrates several networks with sensing applications and cloud infrastructure to provide cross disciplinary application support which can scale up to multiple organizations [25]. The framework of cloud computing services does not support the implementation details of the services provided to the user. They make use of services provided to them when they make a request for the corresponding services. The user cannot make use physical sensors when needed as the physical sensors are bound to provide services pertaining to specific applications. Hence there is a need and the requirement to manage these sensors so that the resources are properly and efficiently utilized by the users. Literature showcases the studies on physical sensors focused only on power management [30], localization [31], data processing [32], routing [33], and clock synchronization [34].

There is a significant need of providing schemes for managing the sensors to make use of sensor resources in an efficient manner. There are no generalized application scenarios which uses all types of physical sensors all the time. In [35] a mechanism is proposed to appropriately choose the physical sensors for the specific application scenario. However, sensor services can support a variety of applications if it moved to the cloud [36, 37]. The cloud sensor architecture must satisfy two major requirements: (1) The network should be scalable and energy efficient and (2) It should be able to provide open ended application development system [38].

Broadly, three layered architecture can be visualized for the sensor cloud [14]. The physical sensors are present at the lowest layer of the architecture of sensor cloud. The multiple sensors which are deployed in the application scenarios collect and upload the data to the sensor cloud which is present in the second layer of the sensor cloud architecture in a standardized format. The data uploaded in a standardized format allow users to use the services of the sensor cloud without having concern over protocols and formatting of the uploaded data. The sensor cloud is present at the second layer of the architecture which procreates the virtual sensors defined by the sensor owners based upon the service templates. Finally, at the top layer application developers use these virtual sensors into their respective applications. The three layer architecture is shown in the figure 3.1. The transmission of sensed data to the sensor cloud is done through physical entities or logical entities. The logical entities use the concept of virtualization, which is nothing but the logical abstraction of the physical entities for efficient utilization of resources. For the logical abstraction or to cover the virtualization aspect, a channel is required for the communication between the physical resources and virtual resources.

The two key components of sensor cloud infrastructure are: (1) Resource host (2) Sensor cloud platform.

1. **Resource Host:** Resource host covers the lowest layer of the sensor cloud infrastructure which consists of [39]:
   (a) **Physical sensors:**Physical sensors are the devices that are distributed to the various application domains and have the capability of sensing the data from the deployed region. On the basis of their sensor readings and the actual distance from the events, the physical sensors are ranked [39]. The authors proposed a FIND technique [39] to find the faulty node in the system by finding the mismatch between the actual distance rank and sensor rank; if there is a mismatch, the node is faulty. Each physical sensor deployed in application scenarios has a unique resource id and shares a common power unit hosted by the sensor coordinators. The physical sensors are XML encoded so that the services provided by these sensors can be utilized on heterogeneous platforms without having the concern over the conversion of these services on various platforms [36].
   (b) **Sensor Coordinators:** The physical sensors do not have the capability to transmit the data; rather it only senses the environment where it is deployed. It is the responsibility of the sensor co-ordinators to transmit the sensed data to the sensor cloud. Based on the transmission capabilities the sensor coordinators which have unique resource id, can be locally or globally identified. The locally identifiable coordinators transmit the sensed data from the physical sensors to the sensor cloud through a sensor gateway, whereas the globally identifiable coordinators transmit the sensed

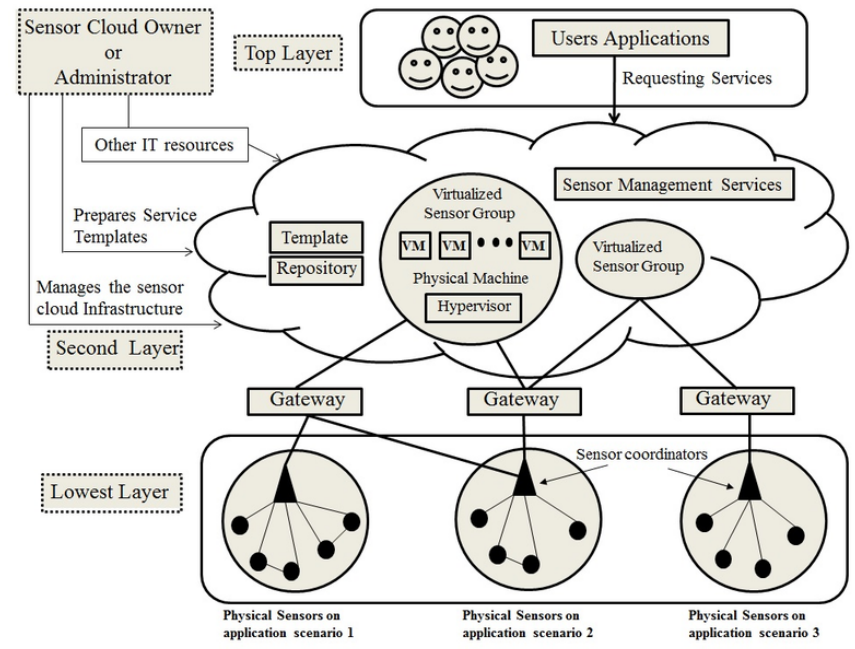

Sensor Cloud Owner or Administrator
Top Layer
Users Applications
Requesting Services
Other IT resources
Sensor Management Services
Prepares Service Templates
Template
Repository
Virtualized Sensor Group
VM
VM
VM
Physical Machine
Hypervisor
Virtualized Sensor Group
Manages the sensor cloud Infrastructure
Second Layer
Gateway
Gateway
Gateway
Sensor coordinators
Lowest Layer
Physical Sensors on application scenario 1
Physical Sensors on application scenario 2
Physical Sensors on application scenario 3

Fig. 3.1: Sensor Cloud Architecture.

data directly to the sensor cloud platform. There are three different topologies a physical sensor can attach to the sensor coordinators [39]:

i. **Fixed:** In this topology, the sensor coordinator is attached to a fixed set of sensors and they cannot be plugged in/out in any way.
ii. **Mobile:** In this topology, sensor coordinators are attached with a variable set of physical sensors that can be plugged in/out in any case. This type of topology supports the mobility of both physical sensor and sensor coordinator as the connection of the physical sensor with the sensor coordinator can either be wired or wireless.
iii. **Variable:** In this type of topology, the sensor coordinator has always been fixed and is attached with a fixed set of physical sensors in a timely slotted manner.

(c) **Sensor Gateway:** The sensed and aggregated data are sent from physical sensors to the cloud platform through sensor gateways. The locally identifiable coordinators sent the data to the cloud platform through the sensor gateway which acts as an interface between the sensor cloud platform and locally identifiable coordinators. The sensor coordinators and various intermediate devices between the coordinator and sensor gateway act intelligently to route the data packets to the sensor gateway. The sensor gateways then take these data traffic and send it to the cloud platform for further processing.

2. **Sensor Cloud Platform:** The sensor cloud platform is the second layer of the sensor cloud architecture which consists of following major components:
   (a) **Templates:** The sensor cloud infrastructure procreates the virtual sensors and provides it to the user applications when required in such a way that the end user has the illusion that the service instances are the part of resources, such as disk storage, CPU, memory, etc. [41]. The service instance creation is done by using the appropriate templates defined by the cloud owners; the end users also make requests of these service instances via the interface by selecting appropriate service templates.
   (b) **Sensor management services:** This manages the services provided by the cloud owner. The

service provider manages the sensor cloud templates and can modify the services of the template as per the requirement of the user application and services [37]. The delivery time of the services to the end users is improved by the automation of services which is an important factor in providing the cloud computing services to the user applications [42]. As human intervention in providing the services has an adverse effect on the system; hence, automation of services proves to improve the flexibility and efficiency of the system [14].

(c) **Virtual sensor:** The virtual sensor abstracts the physical sensor and the concept of virtualization, which will be discussed in subsequent sections of this paper in detail, virtualizes the physical sensors and frees the user from the concern of knowing the status of the connected physical sensors with the sensor cloud infrastructure; it only concerns the status of the virtual sensor. However, the user of the sensor cloud infrastructure should also concern the status of the physical sensors along with the status of the virtual sensor for getting accurate results [14].

The sensor cloud infrastructure is a three layered architecture in which sensor owners are free to register or unregister their physical sensors, and can join the sensor cloud infrastructure. After the registration of the physical sensors, the corresponding IT resources will become operational and templates for the virtual sensors and virtualized sensor groups are created. The service templates are created as a catalogue menu service and the users can create new services for the existing sensors service instances [14]. Once the templates for the service instances are created, the virtual sensors can share the sensor data and user application can request virtual sensors for their use by appropriately selecting the service templates. Further, the service templates are discarded once they become useless to reduce the utilization charges for the corresponding resources [37].

4. Sensor virtualization.

The most pertinent technology for cloud computing is virtualization; which hides the heterogeneous platforms, infrastructure, and data from the user applications without giving the details about the underlying hardware implementation for seamless operation. The key idea behind virtualization is resource sharing and collaboration. The virtualization basically partitions the physical server into multiple logical servers which in turn behaves like an independent physical server that are able to run the operating system and applications. The virtual server encapsulates the virtualization software called a hypervisor, which runs multiple server instances or virtual machines on a single host by the encapsulating guest version of the operating system. It emulates the physical hardware resources and increases the utilization of resources by reducing the need of physical hardware systems. There are two types of hypervisors that virtualization software uses, to manage multiple virtual machines on a single host machine:

Type 1 or bare-metal hypervisors: The type 1 hypervisors are installed directly on to the server to control the physical hardware and hence, provide higher efficiency, stability and are ideal for larger operations. The following are the Type 1 hypervisors; Nutanix AHV, AntsleOs, Xen, XCP-ng, Oracle VM Server for SPARC, Microsoft Hyper-V, Oracle VM Server for x86, , Xbox One system software, and VMware ESXi.

Type 2 or hosted hypervisors: The type 2 hypervisors are not installed directly onto the server; instead, it is installed on top of the server's operating system. It is easier to install and are ideal for small operations. Following are the examples of type-2 hypervisors: Microsoft virtual PC, VMware Workstation, VMware Player, VirtualBox, Oracle Solaris Zones.

Many companies like VMWare, Microsoft, IBM, Citrix, RedHat, and Oracle, etc., provide virtualization services for storage and computations to the user requesting for services. The virtual server indeed proved to be cost-effective and less time consuming when compared to traditional methods for storing and computing the data. Hence, the benefits of hypervisors made the sensor cloud environment to incorporate the virtual sensors.

The virtual sensor is software that abstracts the physical sensors deployed in the application domain. The physical sensors convert the collected data, from electromagnetic signals into digital form in a standardized format and send it to the virtual sensors.The need of a virtual sensor is not to replace the physical sensor; instead it facilitates, adds further functionalities and can create the additional module if required for the physical sensors at the software-computing level for efficient resource utilization [39]. The virtual sensor can be placed and accessed from anywhere which improves the coverage parameter of physical sensors and hence enhance the performance of physical sensors with limited resources [39]. The group of virtual sensors has unique identities that are mapped to an IP. APIs at the IaaS level are used for virtual sensor allocation and enable the user application to establish the connection, uploading data to the cloud, register and deregister with the virtual sensor, etc. After the allocation of virtual sensors to a particular virtual machine, it can communicate with the physical sensors present at the lowest layer of the sensor cloud architecture. The virtualization technique helps to manage the physical sensor management services through the hypervisor. The physical sensors have the ability to sense, process the data, store, and communicate with the system. These abilities of the physical sensors are the four modules of the system which are distributed at various levels (physical sensor level, sensor gateway, or cloud) of the sensor cloud architecture [39]. The virtual sensor interacts with the system which consists of four modules of the physical sensor interacting with each other. The sensor module is placed at the sensor coordinator or sensor gateway which is the lowest level of the sensor cloud architecture. The sensor module is closest to the physical devices where data is being aggregated [39]. The processing module processes the data aggregated from the physical devices and can be further subdivided to handle various other individual tasks. It is basically placed at the cloud platform and works at the IaaS or PaaS level [39]. Another important module of the system attached to the virtual sensor, is a storage module that stores the aggregated and processed data.

The main limitation with the sensor device is storage as it has memory constraints. Hence, the storage module of the system solves the issue by mapping the sensor memory to cloud storage. A virtual sensor can also represent temporal data [39]. The communication module of the system helps in the transmission and reception of the data. The source port of the virtual sensor aggregates the data from the physical device coordinator through sensor gateway and destination port of the virtual sensor transmit this data to the cloud or forward it to the other virtual sensor for further processing or transmit to the user applications if required [39]. An error-free data must be provided by the virtual sensor; for that, the virtualization aspect of the sensor should monitor the system. The APIs of virtual sensor management services, monitor network connectivity and bandwidth availability, control the events, report data redundancy etc., for efficient and error-free data availability [39].

5. Challenges and Constraints. There are various challenges and constraints associated with the sensor cloud infrastructure which make sensor cloud to be limited in certain scenario of applications; removal of such challenges and constraints will make the sensor cloud infrastructure into a different level, which allows the user to be more flexible regardless of the security and other concerns. Following are the challenges that need to be considered before wide acceptability of sensor cloud infrastructure [14]:

1. **Design issues:** The continuous transfer of data must be guaranteed between the sensor devices and the server for reliable and fault-tolerant communication. The designer needs to focus on these issues to avoid accumulation errors which can be occurred in applications like medical health care monitoring, hospitals, etc. Such types of applications require continuous transfer of data as the patient or the person concerned moves frequently, in and out from the coverage area of the smart device gateways [43, 44].
2. **Storage issues:** Storage of data is done at the server side and there is lots of data processing that creates bursty data processing and needs to be avoided because of the simultaneous connection of multiple clients with the system. To deal with the problem, a predictive storage concept has been introduced [45] in which data at some remote sensors of the sensor cloud infrastructure are archived and predictive caching is used at the proxies of the systems.
3. **Authorization issues:** Privacy issues can be dealt with by giving authorization to different entities that communicate with the system and authenticate the entities via a web interface.
4. **Power issues:** Power issue is a critical issue that should be dealt with when the mobile phone gateway is connected with the sensor cloud infrastructure [46]. The continuous transfer of data and data processing drains out the battery of mobile phones.
5. **Event management:** The data come from heterogeneous sensors and the sensor cloud infrastructure needs to deal with different data abstraction models from different vendors. The infrastructure should develop the APIs, standard format and design new database mechanisms to deal with the event query coming from the users, for the large real-time data.
6. **Service level agreement violation:** The infrastructure should be able to provide quality services

to the users and if it fails to do so the cloud provider violates the service level agreement. Opting the best possible cloud provider in terms of cost, QoS, time is the biggest challenge [47].

7. **Efficient information dissemination:** Efficient dissemination of information is a critical challenge as the data sets and their respective access services are geographically distributed and it is difficult to provide the data and services to the appropriate user applications.
8. **Security and privacy issues:** Security and privacy is an important concern for any network. With a sensor cloud platform the data comes from a diverse range of physical devices and it is the responsibility of the infrastructure to secure the sensitive data and create better privacy policies to maintain privacy [47, 48].
9. **Real time multimedia content processing issue:** The availability of real time multimedia data from the large data sources in the cloud and to classify real time multimedia and contents to provide appropriate services to users at their location is another challenge for the sensor cloud infrastructure [24].
10. **Energy efficiency issue:** The constant transmission of data from the sensor devices to the cloud consumes lots of energy as the sensor nodes have power constraints and also they have less storage and processing capabilities. Therefore, there is an ultimate need for the transmission of the sensed data to the cloud for further processing and hence the nodes consume power in sensing and transmission. So, one solution to this problem is to add a smart gateway to the middleware [49] that can do some pre-processing and compression of data and can reduce the transmission load. The reduction in transmission load and compression of data can ultimately reduce energy consumption and can improve memory usage.
11. **Bandwidth limitation and network access management:** The number of connected devices in a sensor network and the cloud users is large. So, to manage these devices in terms of bandwidth utilization is a big challenge for the sensor cloud infrastructure [50]. The efficient access management of the number of the connected network in the sensor cloud infrastructure helps in bandwidth utilization [51].

6. Integration of IoT with sensor cloud. The Internet of Things (IoT) and sensor cloud are two broad and distinct technologies. The integration of sensor cloud and IoT has opened the door for future internet and benefits the real world problems. The seamless integration of sensor cloud and the Internet of things make the sensor cloud platform reachable and also solve the storage and computing limitation of IoT. The applications of both technologies are becoming viable, low cost, approachable, and robust and are extremely pervasive. There are two components involved in the integration of sensor cloud and IoT: (1) Sensor cloud infrastructure and (2) IoT system. The sensor cloud infrastructure provides the platform for the IoT system and mitigates its storage and computational limitations whereas the IoT system provides the interoperable and lightweight procedures for exchanging the data and information with the sensor cloud infrastructure. The architecture of IoT is still under construction enabling the future technologies, but is mainly divided into three layers: (1) Perpetual layer (2) Network layer (3) Application layer. Further, in some literature [52] [53] researchers have added two more layers: (1) Middleware layer and (2) Business layer. The five layer architecture is described in Fig. 6.1.

The lowest layer of the IoT architecture is the perception layer consisting of data sensors, RFID tags, actuators, GPS, Bluetooth, camera, barcode labels, etc. The perception layer is the physical layer that identifies the unique things and perceives the data from the environment. The layer deals with the gathered and sensed data from the environment and sends it to the network layer. The network layer of the IoT architecture consists of one or more gateway with interfaces through which the gathered and sensed data are sent to the internet. The layer is responsible for connecting to the other network devices, servers, smart things, and performs the network management of the architecture. The next layer of the IoT architecture is the middleware layer also called as processing layer; it receives the information from the network layer. The middleware layer is responsible for storing, analysing, and processing the huge amount of information and takes decisions on the processed information. It provides and manages various services to underlying layers by employing various technologies like databases, big data processing modules, cloud computing etc. [54]. The processed information is then sent to the application layer which provides various applications and services to the users. The information received from the application layer is then utilized for creating various services for the business. The business layer deals with processing or moulding the information for providing various services to users.

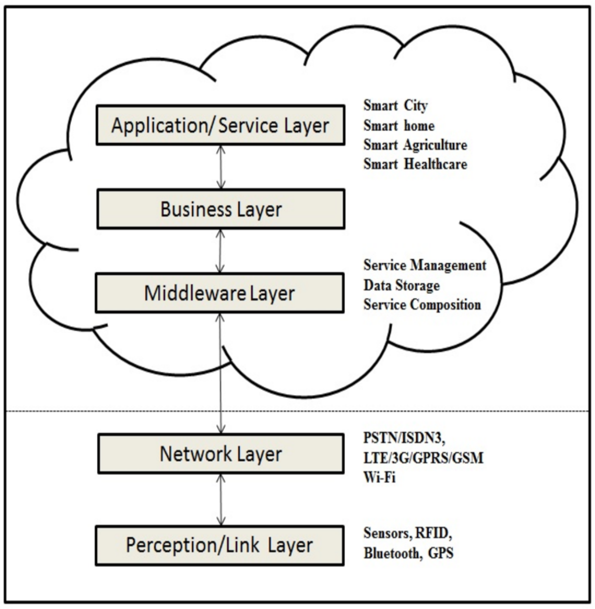


Fig. 6.1: Layers of Internet of Things.

6.1. Significance of integrating IoT with sensor cloud. The ensembling of IoT and sensor cloud can be seen as the potential solution to the limitations of the IoT. The sensor cloud provides an open, flexible, scalable and reconfigurable environment for various monitoring and controlling applications which makes it easier to enable with IoT. Following are the significance of integrating the IoT with sensor cloud:

1. **Physical sensors:** The physical sensors which are used to sense the environment has difficulty in tracking the objects. Hence, the integration of unique identifiers with the physical sensors provides a reasonable addition to the network and the objects of the network become trackable with the sensing capabilities [55]. Moreover, this integrated architecture will be able to support heterogeneous platforms as the physical sensors are XML encoded to support several services on various platforms [36].
2. **Mobility:** The sensor nodes in sensor networks are usually stationary whereas in IoT the nodes are mobile with unique identities which help to identify the nodes. By integrating the two networks, the integrated system can be able to support applications which require mobility.
3. **Communications:** The physical sensors in the sensor network provide multi-hop communication and the unique identifiers like barcodes, serial numbers, RFID helps to identify and track the objects in the integrated system [56] which will offer an effective solution to track, connect and manage the things from anywhere and anytime.
4. **Cost:** By replacing some of the physical sensors of the sensor network with small and cheap RFID tags, the integrated system becomes more economical [55].
5. **Storage:** The IoT has a large amount of structured and unstructured data from various information sources [57], but has limited storage capability; hence integrating IoT with sensor cloud improves the storage capability of the integrated system.
6. **Computation:** The IoT nodes deal with limited data processing and energy resources and need another more powerful node for data processing and aggregation of data. The sensor cloud solves this issue by offering unlimited processing capabilities. Hence, the integrated system offers large processing capabilities and saves time and energy.
7. **Security and Privacy:** The sensor network provides multi-hop communication. In multi-hop communication the sensed data are sent to the sensor cloud platform through the sensor coordinator. The

attached physical devices with the sensor coordinator are seen as a black box to the outer layer and provide more security and privacy to the IoT network [10].

8. **Scalability:** The integrated architecture of IoT with sensor networks allows the IoT system to be more scalable, as only cloud is not enough to support various physical devices; the cloud requires various data centres which are comparatively costly.
9. **Visualization:** The visualization API of sensor cloud infrastructure helps the users to predict future trends through visualization tools with the stored sensed data from various physical devices [48].
10. **Resource Optimization:** As the sensor-cloud provides resource optimization which enables the system to share the resources for various applications [48]; integrating sensor-cloud with IoT makes the integrated system utilize the shared resources which reduce the resource cost of the system and provide wider range of applications and services.

7. Possible ensemble architecture of the IoT and sensor cloud. The ensembling of IoT and sensor cloud is guaranteed to serve a variety of real-world applications as they make use of their respective advantages to provide better services to the users. In essence, the ensemble architecture of IoT and sensor cloud contains numerous elements, such as IoT sensors and actuators, sensor coordinators, protocols, gateways, sensor cloud, and data center. The ensemble architecture of IoT and sensor cloud as shown in Fig. 7.1 which consists of two main components: (1) Unique identifiers, IoT sensor, and actuators and (2) Sensor cloud platform.

The IoT devices are present in the lowest layer of the architecture which consists of unique identifiers to identify the physical objects and sensors to sense the physical environment. The identifier is a pattern of characters and numbers that is used to identify physical or virtual entities within a single context. In IoT standards, identifiers are divided into distinct categories [58]: (1) object identifiers (Object Ids) (2) communication identifiers (Communication IDs) and (3) application identifiers (Application IDs). The object identifiers are used to identify physical or virtual objects [58], like barcodes, RFIDs, Uniform Resource Locator (URL) and Uniform Resource Identifiers (URI), MAC, serial numbers, etc. The communication identifiers are used when the devices or nodes in the network needed to be uniquely identified [58]. IPV4, IPV6, and (Domain Name System) DNS etc., comes under communication IDs. The application IDs are used to identify the services provided by the application layer like URL and URI are application layer identifiers [58]. Besides these, there are several universal identification schemes, such as Electronic Product Code (EPC), International Mobile Equipment Identity (IMEI), Universal Unique Identifier (UUID), Universal Product Code (UPC), OID (Object identifier), etc [59, 60]. After unique identification of the physical entities, the IoT sensors and actuators are used to sense the physical environment. The IoT sensor changes the physical parameter into electrical signals and reliable, accurate sensors are utilized in various applications in miniaturized packages, health care, packages for harsh environments, multi sensor modules and led the foundation for engineers to apprehend the diverse properties of applications. As against IoT sensors, actuators are used to control or alter the physical changes; they convert the electrical signals into physical output, like heat or motion. For example, actuators can be utilized in laser, LED, loudspeaker, solenoid, motor controllers, etc. The sensed information from the sensors and actuators are then sent to the sensor coordinators which is locally or globally identified in the network; for the transmission of the data to the sensor cloud through sensor gateways or directly to the sensor if sensor coordinators are globally identified, for further processing as discussed in Section 3.

The sensor gateways are smart physical device or software program that acts as a bridge between the sensor cloud and sensor coordinators, intelligent devices, etc., and are provided with little extra computational functionality; so that they can decide when to upload the data to the sensor cloud depending upon the application scenarios. The physical devices generate the sensed data when they are connected to the network, but at some point in time, the sensed data is no longer required. The uploading of sense data to the sensor cloud and synchronization of devices becomes unnecessary depending upon the application scenarios; and needs to be stopped for a while for preserving the energy, sensor cloud, and network resources [61]. Apart from uploading the necessary data to the cloud infrastructure, the smart gateways also perform various other tasks, like pre-processing of sensed data from the IoT enabled devices, filtering and restructuring the data into a useful form, keep checks on the physical device and energy constraints of the devices, provide security and privacy to the IoT network and physical devices, etc [10]. The multiple IoT devices connected to a network can directly send their data to the smart gateway which is possible in single hop communication or they can send the data

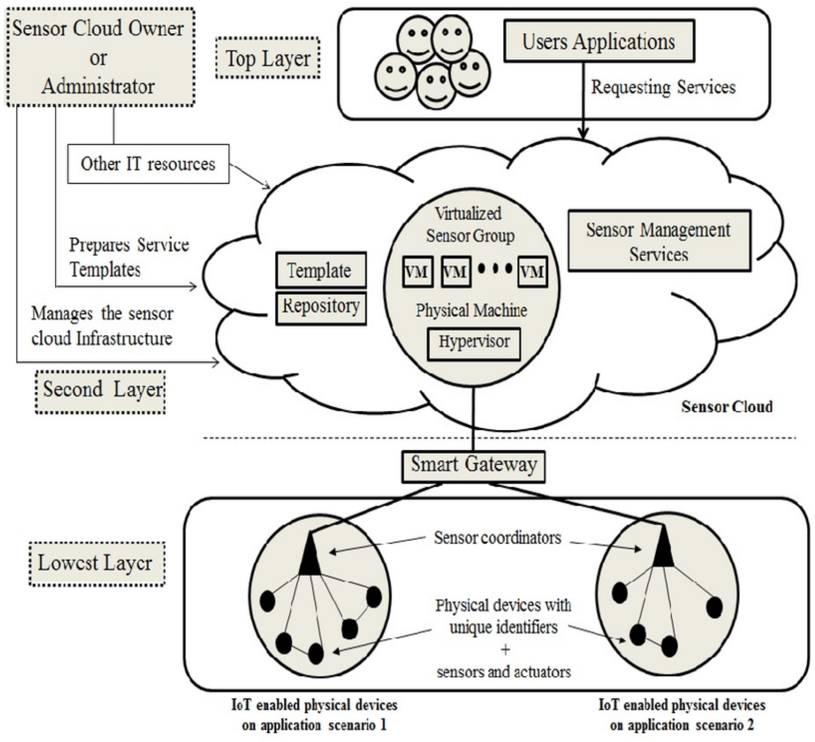

Sensor Cloud Owner or Administrator
Top Layer
Users Applications
Requesting Services
Other IT resources
Prepares Service Templates
Manages the sensor cloud Infrastructure
Second Layer
Template
Repository
Virtualized Sensor Group
VM
VM
VM
Physical Machine
Hypervisor
Sensor Management Services
Sensor Cloud
Smart Gateway
Lowest Layer
Sensor coordinators
Physical devices with unique identifiers + sensors and actuators
IoT enabled physical devices on application scenario 1
IoT enabled physical devices on application scenario 2

Fig. 7.1: Ensemble architecture of IoT and sensor cloud.

through their sensor coordinators in multi-hop communication. The multi-hop communication scenario allows the IoT network to be more scalable, with a diverse range of physical devices and heterogeneous data which requires extensive data analysis and pre-processing from the smart gateways [10]. In multi-hop communication with the smart gateways, the sensor coordinators make their underlying physical devices as a black box to the outer layer; which in turn adds additional security to the IoT network, and hence security can be customized according to the sensor and IoT network [10]. Once the sensed data is pre-processed the restructured data from the smart gateways are then sent to the sensor cloud for further processing as discussed in Section 3. The data from the sensor cloud are then utilized for making various applications and services for users. The application layer deals with the utilization of services and applications for users. The IoT and sensor cloud has a diverse range of applications and integration of these two promising network paradigms promised to offer a variety of real world applications. The applications of the IoT system are based on network availability, coverage, scalability, heterogeneity, repeatability, and user involvement [62]. The most common application of these types of infrastructure can be in health monitoring, battlefield monitoring, telematics, agricultural and irrigation control, earth observation, wildlife monitoring.

8. Research challenges and constraints for ensembling IoT through sensor cloud. Following are the challenges that need to be addressed properly for the proliferation of the efficient enabling of IoT through sensor cloud:

1. **Unique identifiers:** Despite the availability of various identification schemes discussed in Section 7, there is no universal unique identification scheme for the heterogeneous platform for IoT applications. Various IoT platforms use globally unique identification schemes [63, 64] or have their own unique identification schemes to identify the IoT platforms, but the diverse nature of physical devices and the heterogeneous IoT platforms with the absence of de facto standards for naming and addressing, pose

a key challenge over interoperability of different identifiers [65, 66] and need to be addressed. The commercial and the political difference between the standard bodies; which provide the specifications, and framework for the IoT platforms, create the problems of having unique identification schemes [67]. A new universal identification mechanism for the heterogeneous IoT platforms should be created for the interoperability of the different IoT platforms, but moving IoT platforms supporting a diverse range of IoT applications to a whole new universal identification scheme is also quite challenging and poses several issues.

2. **Infrastructure difference:** The presence of infrastructural differences between the sensor cloud and IoT system makes the difficulty in balancing and managing the sensor cloud environment and IoT requirements. Hence, while integrating these two technologies, balancing, and management of the two platforms should be taken care of.
3. **Smart gateway:** Integration of IoT and sensor cloud requires a smart gateway which pre-processes, filters, and restructures the collected data from the IoT devices and transfers the useful data to the sensor cloud; which is not possible with the light IoT devices and sensors. The gateway should be smart and intelligent enough to do the tasks, such as data management, device management. Different types of gateway works at different levels in the protocol layers and the seamless integration of enabling technologies with the smart gateway makes the IoT system smarter to deal with things.
4. **Mobility:** As IoT nodes are mobile; there is a constant disruption of services when the nodes move from one gateway to another depending upon the application scenarios. Hence, it is another issue and needs to be taken into consideration while integrating the IoT with the sensor cloud. There is an ultimate need of designing an efficient mobility management scheme for these kinds of networks where they are a requirement of the constant interaction of IoT nodes for providing the services to the users. Researchers have proposed various mobility models for the various types of network so that the network can be utilized for various real world applications. A resource mobility scheme [68] was proposed which operates in two modes: caching and tunneling; for providing services in continuity to users and allow the application to use the sensed data when the resources are temporarily unavailable during mobility of the nodes. The scheme shows a 30% reduction in service loss in the mobility scenarios. An efficient mobility management scheme should be designed to support heterogeneous devices and platforms. A feasible group mobility management scheme [69] was proposed, in which the nodes with similar mobility behavior stored at their location database are grouped and the leader of that group will manage the mobility on behalf of other nodes of the group which in turn try to mitigate the signalling congestion problem. One more approach [70] exists in the literature that maintains the continuity of services of the migrating sensor nodes in a framework which is based on the concept of a Web of Things. Several other group mobility models for mobile ad hoc networks [71-73] were proposed that manage the mobility pattern of the nodes, inspired by the social behavior of the natural systems.
5. **Standardization:** Standardization of cloud infrastructure is also a key challenge when integrating the IoT with sensor cloud. The cloud interoperates with various cloud vendors where each cloud vendor supports their standardized formats. The integration of IoT with sensor cloud has to deal with the diverse standardized format for the cloud interfaces, resources, and for managing the networks. The interoperability of cloud infrastructure to enable the underlying technologies is very important and needs to be addressed [74, 75].
6. **Performance:** The performance evaluation of the IoT network is an open issue that needs to be addressed when integrating it with other technologies. The performance evaluation of the IoT network is dependent on the performance of various underlying technologies and several other factors, as the network is built from various elements. So thorough evaluation of these networks is a tedious task and is not much reported in the literature, but needs to be addressed for the efficient and wide deployment of the integrated IoT system in real world scenarios.
7. **Energy management:** The sensor nodes consume lots of energy while communicating with the cloud infrastructure. The sensor nodes consist of sensing, transmission, processing, and power unit. The presence of sensors in physical devices to sense the physical environment consumes lots of energy; the temporary power unit, like batteries in a network of billions of physical devices may not be sufficient and

Table 9.1: Characteristics of sensor cloud, IoT and IoT enabled sensor cloud

Characteristics	Sensor cloud	IoT	IoT enabled sensor cloud
Storage capability	Virtually Unlimited	Limited	Virtually Unlimited
Computational Power	Virtually Unlimited	Limited	Virtually Unlimited
Displacement	Centralized	Ubiquitous	Centralized
Data	Generate and manages the unlimited data	Generates lots of data	Generate and manages the unlimited data
Reachability	Pervasive	Limited	Extensive
Mobility of physical devices	Mostly stationary	Mobile	Mobile
Resource Optimization	High	Low	High
Implementation cost	High	Moderate	Low

needs permanent energy supplies from the environment, like mechanical motion, solar energy, radiation, thermal gradient, and light [76, 77]. These renewable energy sources can be used in the sensor nodes [78, 79] to curb the energy limitations of the network. An efficient mechanism should be addressed for the utilization of energy. Integrating fog computing with the IoT system can bring cloud resources locally [80] and can save lots of energy.

8. **Resource allocation:** Resource allocation to IoT devices by the sensor cloud is another challenge that needs to be addressed when enabling IoT with sensor cloud. Resource allocation to the physical entity is a challenging issue as it is difficult to decide how much or whom to give resources when resources are in demand by several physical entities. Depending upon the application scenarios the requirement of resources is decided. One solution to this problem is to bring for computing as middleware to manage resources to the underlying physical devices [10].
9. **Bandwidth usage and network access management:** The drastic increase in physical devices and sensor-cloud users pose great difficulty in the allocation of proper bandwidth to each device and sensor-cloud users despite having the presence of various bandwidth allocation methods in the literature. Providing the proper bandwidth to such a huge infrastructure which is using various networks to deal with its applications is a difficult task, but with proper network access management schemes, the link performance can be improved and can have optimized bandwidth usage [51].

9. Analysis and discussion:. This section analyses the ensemble system of IoT and sensor cloud. There are various literature which showcases the integration of IoT system with other technologies, such as fog computing [81, 82], big data [83, 84], software defined networks [85, 86], machine learning [87], and many more. The adaptability of these integrated IoT systems with other technologies in real world scenarios provides interoperability with a diverse range of systems and is proved to be economically viable. In particular, the seamless integration of IoT and sensor cloud is very significant in terms of their applicability to the diverse range of applications. The characteristics of the sensor cloud, IoT and IoT enabled sensor cloud is presented in Table 9.1. As can be seen, the characteristics of sensor cloud and IoT are complementary to each other and these are some reasons why researcher are showing more interest in integrating these two promising technologies. The integration of the IoT and the sensor cloud will fill the gaps of each other; the IoT will have the benefit of virtually large storage, improved processing capabilities, communication, and resource optimization from the sensor cloud to limit its technological constraints. On the other hand, the sensor-cloud can extend its scope to deal with various physical devices in a dynamic and distributed manner, and become approachable to a diverse range of real world scenarios for providing new services to the users. The sensor cloud in the integrated system provide the intermediary layer between the physical devices and the user applications to hide the complexities and functionalities which will later benefit the implementation of the technologies and will impact the future application development.

The motivation towards the integration of the IoT and sensor cloud lies in the storage, communication, and computation. The properties of IoT, such as interconnectivity, heterogeneity, dynamism, data privacy and security, etc., demand the requirement of connecting heterogeneous things or objects in a dynamic manner with a secured ecosystem containing physical devices, internet, and the end users. The sensor network is identified as main enabler of IoT [88]. Technical advances made low powered, efficient, low cost devices, to be used in large scale applications. The physical devices are enabled with a variety of sensors and are deployed in variety of application domains despite being suffering from various technical constraints, such as energy, processing power, reliability, mobility, etc. With this context, the timely processing of the huge amount of collected data along with making sensor energy efficient, small and reliable; the ensemble system poses several challenges [89]. Integrating sensor data with the cloud provides new opportunities in large coverage, and relevancy, but affects the data privacy and security [89]. Cross platform of IoT is another challenging issue. The manufacturers have to set the specifications of the sensors, actuators, and controllers and are essential for them to incorporate cross-platform solutions to customize requirements for diverse range of applications [90]. Collaborating with other organizations for building the IoT products will reduce the system complexity of the solutions and be more energy efficient, cost effective and reliable, as it is difficult to provide and meet all the requirements of the IoT product by a single organization or company [91].

The integration of IoT and sensor cloud is considered a vital footprint for mitigating various limitations and challenges of both the integrated drivers. These drivers contribute to the ensemble architecture by (1) publishing/subscribing the sensed information: The sensed information from the multiple sensor networks are published and is subscribed by the applications based on their requirements and on-demand basis [87], (2) collaborating with different manufacturers: The IoT devices require cross-platform design and integration of heterogeneous sensors in terms of hardware, communication and sensing range, communication protocols, and standards [90]. Hence, collaboration with different manufacturers and organizations will give ease in the manufacturing the IoT devices, (3) on-demand resource scalability at application runtime and providing customized query processing to the end users and (4) visualizing sensor: The sensor cloud platform provides an API to the end-users for managing the sensors on their own [49]. The end user can perform variously customized operations like addition or deletion of a sensor, selection of sensors based on their current services etc. The sensor visualization is very important for IoT platform as it provides interfaces to customers for issuing the queries and it can be present anywhere like, in smartphones, laptops, and other digital assistants [90]. The ensemble system of IoT and sensor-cloud is making significant improvements and contributing immensely to the proliferation of the IoT system.

10. Future Directions. This section presents the future research direction for the ensemble system of IoT and sensor cloud. Following are the scope where there is a requirement of putting additional research efforts to make use of the whole capacity of the integrated system:

1. **Network virtualization:** Network virtualization is an important aspect of research and many applications of integrated system of IoT and sensor cloud will take benefit from this field. The efficient utilization of network and resources can be done by logically isolating the network partition from the globally distributed network infrastructure.
2. **Common open API and standard formats:** The common API and standard format will make the integrated system more flexible and efficient. It also provides various business opportunities for providing a common open service platform environment for the integrated IoT and sensor cloud system.
3. **Multi-networking:** The ensemble system should support handover and location management. As the mobility is high in IoT enabled sensor cloud and is dynamic in nature; it is necessary to maintain and improve the network reliability, QoS, and fault tolerance.
4. Enabling the ensemble system with other technologies like fog computing and software defined networking will also serve a great benefit to the ensemble system.
5. **Identification:** A unique identification, addressing and naming of the physical device in the integrated system is very important for supporting diverse devices and to handle their mobility.
6. **Optimal gateway selection:** The selection of optimal gateway to serve the user request in IoT enabled sensor cloud environment is important for real time monitoring systems. The mapping of service request to the optimal gateway support variety of applications and make an efficient and reliable integrated system.

11. Conclusion. The internet technology has flourished with the arrival of the IoT and has witnessed a revolution in enabling IoT with the emerging technologies. The limitations of the IoT system can be efficiently handled by seamlessly integrating IoT with other technologies which in turn offer a variety of services and applications. This paper summarizes the concept of IoT and sensor cloud with their challenges and architectures.

Further, the paper reviews the possible ensemble architecture of the IoT and the sensor cloud. Enabling IoT with sensor cloud makes the integrated system cost-effective, flexible, scalable, and provides better computation power and storage. The significance and research challenges of the ensemble system are also being reviewed. The paper analyses and discusses the motivation, and need behind the integration of the IoT and sensor cloud. Moreover, future research directions are also identified in order to exploit the full potential of the integrated system. The integrated system provides a wide variety of advantages and applications which will continue to evolve in years to come.

REFERENCES

[1] J. Verma, *Internet of Things*,Everyman's Science,12 (2018), pp. 12-14.
[2] C. Zhu , V.C Leung, J. J. Rodrigues, L. Shu, L. Wang, and H. Zhou, *Social sensor cloud: framework, greenness, issues, and outlook*, IEEE Network, 32(5) (2018), pp. 100-105, http://www.disca.upv.es/misan/mobmodel.htm, page accessed on October 15th 2017.
[3] https://ietf.org/topics/iot/, Accessed 30 March 2020.
[4] https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/, Accessed 2 April 2020.
[5] P. Mell, and T. Grance,*The NIST definition of cloud computing*, (2011).
[6] A. R. Biswas, and R. Giaffreda, *IoT and cloud convergence: Opportunities and challenges*, in IEEE World Forum on Internet of Things (WF-IoT) , IEEE, 2014.
[7] M.P. Andersen,G. Fierro, and D. E. Culler, *Enabling synergy in iot: Platform to service and beyond*, Journal of Network and Computer Applications, 81 (2017), pp. 96-110.
[8] Q. Zhang, L. Cheng, and R. Boutaba, *Cloud computing: state-of-the-art and research challenges*, Journal of internet services and applications, 1(1) (2010), pp. 7-18.
[9] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, *Fog computing and its role in the internet of things*, in Proceedings of the first edition of the MCC workshop on Mobile cloud computing, ACM, August 2012, pp. 13-16.
[10] M. Aazam, and E. N. Huh, *Fog computing and smart gateway based communication for cloud of things*, in 2014 International Conference on Future Internet of Things and Cloud, IEEE, August 2014, pp. 464-470.
[11] A. Botta, W. De Donato, V. Persico, and A. Pescapé, *Integration of cloud computing and internet of things: a survey*, Future generation computer systems,56 (2016), pp. 684-700.
[12] S. K. Dash, S. Mohapatra, and P. K. Pattnaik, *A survey on applications of wireless sensor network using cloud computing*, International Journal of Computer Science & Emerging Technologies,1(4) (2010), pp. 50-55.
[13] C. S. R. Murthy, and B. S. Manoj, *Ad hoc wireless networks: Architectures and protocols*, Pearson Education, India, 2004.
[14] A. Alamri, W. S. Ansari, M. M. Hassan, M. S. Hossain, A. Alelaiwi, and M. A. Hossain, *A survey on sensor-cloud: architecture, applications, and approaches*, International Journal of Distributed Sensor Networks, 9(2) (2013), pp. 917-923.
[15] L. Atzori, A. Iera, and G. Morabito, *The internet of things: A survey*, Computer networks, 54(15) (2010) , pp. 2787-2805.
[16] J. Minlan, L. Jingyuan, and Z. Xiaokang, *Research on algorithm of three-dimensional wireless sensor networks node localization*,Journal of Sensors (2016).
[17] M. Khelifi, I. Benyahia, S. Moussaoui, F. Naït-Abdesselam, *An overview of localization algorithms in mobile wireless sensor networks*, in 2015 International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS), IEEE, July 2015, pp. 1-6.
[18] J. S. Miller, P. A. Dinda, and R. P. Dick, *Evaluating a basic approach to sensor network node programming*, in Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, ACM, November 2009, pp. 155-168.
[19] G. Fortino, R. Giannantonio, R. Gravina, P. Kuryloski, and R. Jafari, *Enabling effective programming and flexible management of efficient body sensor network applications*, IEEE Transactions on Human-Machine Systems, 43(1) (2012), pp. 115-133.
[20] L. Mottola, and G. P. Picco, *Programming wireless sensor networks: Fundamental concepts and state of the art*, ACM Computing Surveys (CSUR), 43(3) (2011), pp. 19.
[21] S. Biswas, R. Das, and P. Chatterjee, *Energy-efficient connected target coverage in multi-hop wireless sensor networks*, in Industry interactive innovations in science, engineering and technology, Springer, Singapore, 2018, pp. 411-421.
[22] Y. Ma, L. Wang, P. Liu, R. Ranjan, *Towards building a data-intensive index for big data computing–A case study of Remote Sensing data processing*, Information Sciences, 319 (2015), pp. 171-188.
[23] L. Hang, L., W. Jin, H. Yoon, Y. Hong, and D. Kim, *Design and Implementation of a Sensor-Cloud Platform for Physical Sensor Management on CoT Environments*, Electronics, 7(8) (2018), pp. 140.
[24] http://www.ntu.edu.sg/intellisys, Accessed 4 April 2020.
[25] K. T. Lan, *"What's next? Sensor + Cloud?"*, in Proceedings of the 7th International Workshop on Data Management for Sensor Networks, ACM Digital Library, 2010, pp. 978-971.
[26] N. Kurata, M. Suzuki, S. Saruwatari, and H. Morikawa, *Actual application of ubiquitous structural monitoring system using wireless sensor networks*, in Proceedings of the 14th World Conference on Earthquake Engineering (14WCEE), October 2008, pp. 1-9.
[27] G. Demiris, B. K. Hensel, M. Skubic, and M. Rantz, *Senior residents' perceived need of and preferences for "smart home" sensor technologies*, International journal of technology assessment in health care, 24(1) (2008), pp. 120-124.

[28] A. Alexe, and R. Ezhilarasie., *Cloud computing based vehicle tracking information systems*, International journal of technology assessment in health care, IJCST, 2(1 (2011)), pp. 49-52.

[29] B. P. Rao,P. Saluia, N. Sharma, A. Mittal, S. V. Sharma, *Cloud computing for Internet of Things & sensing based applications*, in 2012 Sixth International Conference on Sensing Technology (ICST), IEEE, December 2012, pp. 374-380.

[30] R. Katsuma,Y. Murata, N. Shibata, K. Yasumoto, K., and M. Ito, *Extending k-coverage lifetime of wireless sensor networks using mobile sensor nodes*, in 2009 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, IEEE, December 2012, pp. 48-54.

[31] K. Matsumoto, R. Katsuma, N. Shibata, K. Yasumoto, and M. Ito, *Minimizing localization cost with mobile anchor in underwater sensor networks*, in Proceedings of the Fourth ACM International Workshop on UnderWater Networks, ACM, November 2009, pp. 14.

[32] S. Madden, and M. J. Franklin, *Fjording the stream: An architecture for queries over streaming sensor data*, icde, 2 (2002), pp. 555.

[33] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, *Collection tree protocol*, in Proceedings of the 7th ACM conference on embedded networked sensor systems, ACM, Novemberr 2009, pp. 1-14.

[34] A. Rowe, V. Gupta, and R. R. Rajkumar, *Low-power clock synchronization using electromagnetic energy radiating from ac power lines*, in Proceedings of the 7th ACM conference on embedded networked sensor systems, ACM, November 2009, pp. 211-224.

[35] M. Gaynor, S. L. Moulton, M. Welsh, E. La Combe, A. Rowan, and J. Wynne, *Integrating wireless sensor networks with the grid*, IEEE Internet Computing, 8(4) (2004), pp. 32-39 .

[36] M. Yuriyama, T. Kushida, and M. Itakura , *A new model of accelerating service innovation with sensor-cloud infrastructure*, in 2011 Annual SRII Global Conference, IEEE, March 2011, pp. 308-314.

[37] M. Yuriyama, T. Kushida, *Sensor-cloud infrastructure-physical sensor management with virtualized sensors on cloud computing*, in 2010 13th International Conference on Network-Based Information Systems, IEEE, September 2010, pp. 1-8.

[38] Y. Xu, and A.Helal, *Scalable cloud–sensor architecture for the Internet of Things*, IEEE Internet of Things Journal, 3(3) (2015), pp. 285-298

[39] S. Bose, A. Gupta, S. Adhikary, and N. Mukherjee, *Towards a sensor-cloud infrastructure with sensor virtualization*, in Proceedings of the Second Workshop on Mobile Sensing, Computing and Communication, June 2015, pp. 25-3).

[40] S.Guo, Z, Zhong, and T. He *FIND: faulty node detection for wireless sensor networks*, in Proceedings of the 7th ACM conference on embedded networked sensor systems, November 2009, pp. 253-266.

[41] R. S. Ponmagal, and J. Raja, *An extensible cloud architecture model for heterogeneous sensor services*, International Journal of Computer Science and Information Security, 9(1) (2011), pp. 147-155.

[42] C. O.Rolim, F. L. Koch, C. B. Westphall, J. Werner, A. Fracalossi, and G. S. Salvador, *A cloud computing solution for patient's data collection in health care institutions*, in 2010 Second International Conference on eHealth, Telemedicine, and Social Medicine, IEEE, Februrary 2010, pp. 95-99.

[43] J. Biswas, J. Maniyeri, K. Gopalakrishnan, L. Shue, J. E. Phua, H. N. Palit, and X. Li, *Processing of wearable sensor data on the cloud-a step towards scaling of continuous monitoring of health and well-being*, in 2010 annual international conference of the IEEE engineering in medicine and biology, IEEE, August 2010, pp. 3860-3863.

[44] G. Singh, J. O'Donoghue, and C. K. Soon, *Telemedicine: issues and implications*, Technology and Health Care, 10(1) (2002), pp. 1-10.

[45] M.Islam, M. M. Hassan, G. W. Lee, and E. N. Huh, *A survey on virtualization of wireless sensor networks*, Sensors, 12(2) (2012), pp.2175-2207.

[46] K. Lee, D. Murray, D. Hughes, and W. Joosen, *Extending sensor networks into the cloud using amazon web services*, in 2010 IEEE International Conference on Networked Embedded Systems for Enterprise Applications, IEEE, November 2010, pp. 1-7.

[47] X. H. Le, S. Lee, P. T. H. Truc, A. M. Khattak, M. Han, D. V. Hung, and E. N. Huh, *Secured WSN-integrated cloud computing for u-life care*, in 2010 7th IEEE Consumer Communications and Networking Conference, IEEE, January 2010, pp. 1-2.

[48] C.Doukas, and I. Maglogiannis, *Managing wearable sensor data through cloud computing*, in 2011 IEEE Third International Conference on Cloud Computing Technology and Science, IEEE, Novemeber 2011, pp. 440-445.

[49] L. D. Kumar, S. S. Grace, A. Krishnan, V. M. Manikandan, R. Chinraj, and M. R. Sumalatha ,*Data filtering in wireless sensor networks using neural networks for storage in cloud*, in 2012 International Conference on Recent Trends in Information Technology, IEEE, April 2012, pp. 202-205.

[50] Y.Xu, S. Helal, M. Thai, M. Scmalz, *Optimizing push/pull envelopes for energy-efficient cloud-sensor systems*, in Proceedings of the 14th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems, ACM, October 20111, pp. 17-26.

[51] F.Ge, H. Lin, A. Khajeh, C. J. Chiang, M. E. Ahmed, W. B. Charles, and R. Chadha, *Cognitive radio rides on the cloud*, in 2010-MILCOM 2010 MILITARY COMMUNICATIONS CONFERENCE, IEEE, October 2010, pp. 1448-1453.

[52] , W. Qun, *Research on architecture of Internet of Things and construction of its simulation experiment platform [J]*, Experimental Technology and Management, 10 (2010).

[53] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, *Future internet: the internet of things architecture, possible applications and key challenges*, in 2012 10th international conference on frontiers of information technology, IEEE, December 2012, pp. 257-260.

[54] P.Sethi, and S. R. Sarangi, *Internet of things: architectures, protocols, and applications*, Journal of Electrical and Computer Engineering, 2017.

[55] H. Liu, M. Bolic, A. Nayak, and I. Stojmenovic, *Taxonomy and challenges of the integration of RFID and wireless sensor networks*, IEEE network, 22(6) (2008), pp. 26-35.

[56] C. Englund, and H. Wallin, *RFID in wireless sensor network*, Doctoral dissertation, Chalmers tekniska högsk, Doctoral dissertation, Chalmers tekniska högsk.

[57] *EC–European Commission, Definition of a Research and Innovation Policy Leveraging Cloud Computing and IoT Combination*, Final report, (2014).

[58] C.Pastrone, D. Rotondi, A. Skarmeta, H. Sundmaeker, O. Vermesan, S. Ziegler, and L. Yang, *Internet of things*, eu-china joint white paper on internet-of-things identification. Tech. Rep., European Research Cluster on the Internet of Things.

[59] A.Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, *Internet of things: A survey on enabling technologies, protocols, and applications*, IEEE communications surveys & tutorials, 17(4) (2015), pp. 2347-2376.

[60] Z.Yan, N. Kong, Y. Tian, and Y. J. Park, *A universal object name resolution scheme for IoT*, in 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, IEEE, August 2013, pp. 1120-1124.

[61] M. Aazam, I. Khan, A. A. Alsaffar, and E. N. Huh, *Cloud of Things: Integrating Internet of Things and cloud computing and the issues involved*, in Proceedings of 2014 11th International Bhurban Conference on Applied Sciences & Technology (IBCAST) Islamabad, Pakistan, IEEE, January 2014, pp. 414-419.

[62] A.Gluhak, S. Krco, M. Nati, D. Pfisterer, N. Mitton, and T. Razafindralambo, *A survey on facilities for experimental internet of things research*, IEEE Communications Magazine, 49(11) (2011), pp. 58-67.

[63] R.Ma, Y. Liu, C. Shan, X. L. Zhao, and X. A. Wang, *Research on Identification and Addressing of the Internet of Things*, in 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), IEEE, November2015, pp. 810-814.

[64] G.Roussos, and P. Chartier, *Scalable id/locator resolution for the iot*, in 2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing, IEEE, October 2011, pp. 58-66.

[65] C. H. Liu, B. Yang, and T. Liu, *Efficient naming, addressing and profile services in Internet-of-Things sensory environments*, Ad Hoc Networks, 18 (2014), pp. 85-101.

[66] L.Roalter, M. Kranz, and A. Möller, *A middleware for intelligent environments and the internet of things*, in International Conference on Ubiquitous Intelligence and Computingpp, Springer, Berlin, Heidelberg, October 2010, pp. 267-281.

[67] H. Aftab, K. Gilani, J. Lee, L. Nkenyereye, S. Jeong, and J. Song, *Analysis of identifiers on IoT platforms*, Digital Communications and Networks, (2019).

[68] F. Ganz, R. Li, P. Barnaghi, and H. Harai, H, *A resource mobility scheme for service-continuity in the Internet of Things*, in 2012 IEEE International Conference on Green Computing and Communications, IEEE, November 2012, pp. 261-264.

[69] H. L. Fu, P. Lin, H. Yue, G. M. Huang, and C. P. Lee, *Group mobility management for large-scale machine-to-machine mobile networking*, IEEE Transactions on Vehicular Technology, 63(3) (2013), pp. 1296-1305.

[70] T.Elsaleh, A. Gluhak, and K. Moessner, *Service continuity for subscribers of the mobile real world Internet*, in 2011 IEEE International Conference on Communications Workshops (ICC), IEEE, June 2011, pp. 1-5.

[71] S. Misra, and P. Agarwal, *Bio-Inspired Group Mobility Model for Mobile Ad hoc Networks based on Bird-Flocking Behavior*, Soft Computing, 16(3) (2012), pp. 437-450.

[72] J. Verma, and N. Kesswani, *BIGM: A Biogeography Inspired Group Mobility Model for Mobile Ad Hoc Networks*, International Journal of Wireless Information Networks, 25(4) (2018), pp. 488-505.

[73] J. Verma, and N. Kesswani, *AMIGM: Animal Migration Inspired Group Mobility Model for Mobile Ad hoc Networks*, Scalable Computing: Practice and Experience, 20(3) (2019), pp. 577-590.

[74] E. B. Gürsel, and A. Tarek, *Analysis of Interoperability In Cloud Computing*, in Proceedings of the 2019 5th International Conference on Computer and Technology Applications, April 2019, pp. 189-192.

[75] , G. Lewis, *The Role of Standards in Cloud-Computing Interoperability*, CMU/SEI-2012-TN-012, Software Engineering Institute, Carnegie Mellon University, (2012).

[76] S. Kim, R. Vyas, J. Bito, K. Niotaki, A. Collado, A. Georgiadis, M. M. Tentzeris, *Ambient RF energy-harvesting technologies for self-sustainable standalone wireless sensor platforms*, in Proceedings of the IEEE, 102(11) (2014), pp. 1649-1666.

[77] P. D.Mitcheson, E. M. Yeatman, G. K. Rao, A. S. Holmes, and T. C. Green, *Energy harvesting from human and machine motion for wireless electronic devices*, in Proceedings of the IEEE, 96(9) (2008), pp. 1457-1486.

[78] , C.Botteron, D. Briand, B. Mishra, G. Tasselli, P. Janphuang, F. Haug, and P. A. Farine, *A low-cost UWB sensor node powered by a piezoelectric harvester or solar cells*, Sensors and Actuators A: Physical, 239 (2016), pp. 127-136.

[79] B.Gusarov, E. Gusarova, B. Viala, L. Gimeno, S. Boisseau, O. Cugat, and B. Louison, *Thermal energy harvesting by piezoelectric PVDF polymer coupled with shape memory alloy*, Sensors and Actuators A: Physical, 243 (2016), pp. 175-181.

[80] D.Evans, *The internet of things: How the next evolution of the internet is changing everything*, CISCO white paper, 1(2011), pp. 1-11.

[81] R. Z. Naeem, S. Bashir, M. F. Amjad, H. Abbas, and H. Afzal, *Fog computing in internet of things: Practical applications and future directions*, Peer-to-Peer Networking and Applications, 12(5) (2019), pp. 1236-1262.

[82] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, and O. Rana, *Fog computing for the internet of things: A Survey*, ACM Transactions on Internet Technology (TOIT), 19(2) (2019), pp. 1-41.

[83] B. P. Sutjiatmo, A. Erwinsyah, E. Laxmi Lydia, K. Shankar, P. T. Nguyen, W. Hashim, and A. Maseleno *Empowering internet of things (IoT) through big data*, (2019).

[84] X. Xu, Q. Liu, Y. Luo, K. Peng, X. Zhang, S. Meng, L. Qi, *A computation offloading method over big data for IoT-enabled*

cloud-edge computing, Future Generation Computer Systems, 95 (2019), pp. 522-533.

[85] P. Throrat, S. Singh, A. Bhat, V. L. Narasimhan, G. Jain, *SDN-Enabled IoT: Ensuring Reliability in IoT Networks Through Software Defined Networks*, in Towards Cognitive IoT Networks, Springer, Cham, (2020), pp. 33-53.

[86] K. Sood, S. R. Pokhrel, K. Karmakar, V. Vardharajan, and S. Yu, *SDN-Capable IoT Last-Miles: Design Challenges*, in 2019 IEEE Global Communications Conference (GLOBECOM), IEEE, December 2019, pp. 1-6.

[87] F. Hussain, R. Hussain, S. A. Hassan, and E. Hossain, *Machine learning in IoT security: current solutions and future challenges*, arXiv preprint arXiv:1904.05735,(2019).

[88] A.Zaslavsky, C. Perera, and D. Georgakopoulos, *Sensing as a service and big data*, arXiv preprint arXiv:1301.0159, (2013).

[89] F. Zhao, *Sensors meet the cloud: Planetary-scale distributed sensing and decision making*, in 9th IEEE International Conference on Cognitive Informatics (ICCI'10), IEEE, July 2010, pp. 998-998.

[90] S. Misra, S. Sarkar, and S. Chatterjee, *Sensors, Cloud, and Fog: The Enabling Technologies for the Internet of Things*, CRC Press (2019).

Edited by: Dana Petcu
Received: May 15, 2021
Accepted: Nov 16, 2021

Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2021 SCPE. Volume 22, Issues 4, pp. 463–468, DOI 10.12694/scpe.v22i4.1947

A REVIEW OF BLOCKCHAIN-ENABLED FOG COMPUTING IN THE CLOUD CONTINUUM CONTEXT

ADRIAN SPĂTARU*

Abstract. This article surveys the literature in search of systems and components that use Blockchain or Smart Contracts to manage computational resources, store data, and execute services using the Cloud paradigm. This paradigm has extended from warehouse-scale data centres to the edge of the network and in between, giving rise to the domains of Edge and Fog Computing. The Cloud Continuum encompasses the three fields and focuses on the management of applications composed of connected services that span from one end to the other of the computational spectrum. Several components that are commanded by Smart Contracts are identified and compared concerning their functionality. Two important research directions are the experimental evaluation of the identified platforms and the identification of standards that can accelerate the adoption of Blockchain-based Fog platforms.

Key words: Blockchain, Decentralized Cloud, Service Orchestration

AMS subject classifications. 68M14,68M15

1. Introduction. New challenges concerning data transfer have been introduced in the scope of the Internet of Things (IoT) advancements. Previous paradigms requiring all data to be moved in the Cloud for processing have been replaced by the Edge and Fog Computing paradigms. Edge Computing tries to push the computation on the end-devices, while Fog Computing tackles the processing of data at an intermediate level, between the data originator and the Cloud. This reduces network congestion, but constructing proprietary Fog networks implies a high cost and both in terms of investment and maintenance.

The emergence of Blockchain technologies (Bitcoin [20], Ethereum [4]) has proved that if economic incentives are high, then the crowd will invest in the hardware required for running the technology. In the case of blockchain mining, the resources are mostly wasted and the probability to mine the next block is decreasing as more nodes join the network. Instead, some of the high-end personal computers and small private Clouds are excellent candidates for exposing their resources to the Fog fabric. In this manner, the nodes will be reimbursed for executing applications, storing data, or monitoring other nodes.

There has been a surge in the number of platforms developed to take advantage of security, transparency, and traceability offered by the Blockchain. Nevertheless, these properties can only be offered for data stored on the Blockchain. Generally, there is a need for further proofs related to objects managed externally, proofs which can be checked using Smart Contracts. FileCoin [2] builds upon the IPFS peer-to-peer file system, using Ethereum Smart Contracts for handling access management, payments, and reward allocation. A Proof of Replication [11] algorithm has been proposed to validate the amount of space a node has dedicated for storing file blocks under different replication standards.

The biggest downside of peer to peer systems constructed using personal computers is the unpredictability of the nodes' availability. The Blockchain use case does not suffer from this because nodes do not participate actively in synchronization procedures. Rather, when a block is mined, the block is sent to other peers which will validate it and append it to the history of blocks. If a peer is not online at the moment, the peer will ask for all new blocks when coming back online. In the case of file systems and cloud services, there is a need to ensure that nodes that hold files or execute applications will be available for a given period, or at least have backup plans in case they become absent. This should also be ensured for the components that manage the platform, as they are also external to the Blockchain and need to maintain a state in a distributed environment.

*Department of Computer Science, West University of Timişoara, Romania. (adrian.spataru@e-uvt.ro).

Table 2.1: Current research directions

Research direction	**Approach**	**Research Items**
File Storage, Content Distribution Networks	Some use Smart Contracts to check an externally generated proof of storage. Others use Smart Contracts to retais a mapping of data blocks to locations and to manage access control. Data is stored on nodes running Ethereum and IPFS.	[2, 11], [7, 18, 26]
Resource Management / Task scheduling	Smart Contract maintains list of resources. Some rely on the Smart Contract to make the matchmaking, but this approach is expensive in terms of gas. Instead, the parties can negotiate off-chain and the Smart Contract can *authorize* their agreement.	[13, 27, 32, 33, 34]
Service Orchestration	Several Smart Contracts are used to manage the resources across multiple clusters, and to instantiate applications composed of one or multiple services which may depend on each other. Several Orchestration components assure the fault tolerance of the deployment and monitoring processes.	[28, 29]
Monitoring and Quality of Service	One approach is to use Oracles (trusted parties) to monitor the services and assess their quality. Instead, a peer to peer network with a robust protocol can ensure the correctness of the monitoring process.	[14, 29, 31]
Result Verification	Credibility-based fault tolerance protocols and zero-knowledge proofs can be used to determine if a node has executed the program as expected. These strategies are encoded in Smart Contracts which verify the proofs. A different approach is to use Trusted Execution Environments which cryptographically ensure that the program is executed as planned.	[8, 15, 23]

The subject of peer to peer networks has been investigated thoroughly with respect to distributed consensus [21, 16, 25, 5, 6], file sharing protocols like Kademlia [17], BitTorrent [22], IPFS [3], and protocols for volunteer computing (BOINC [1], XtremWeb[10]) and achieving correct results in the presence of saboteurs [23].

This report surveys the literature in search of platforms or components that use Smart Contracts for the management of computational resources and deployment of applications. Two surveys complement our work. The survey presented in [30] inspects three Blockchain-based Cloud platforms. It provides an abstraction based on the three architectures and presents standards that can be used to ease the integration between Cloud Components. An in-depth investigation is pursued in [9], which focuses on Blockchain protocols with built-in logic for specific delivery paradigms (IaaS, PaaS, SaaS, and more).

The contributions of this paper are the identification of Fog platforms and components that can be used in the context of Cloud Continuum. Several components are identified and categorized based on their functionality, and components within the same category are compared. Finally, future research directions required for the integration of the Cloud, Fog, and Edge devices are presented.

2. Algorithms and Components.

Several research papers tackle the decentralization of the Cloud by proposing solutions for specific Cloud processes such as resource management, data storage, or service orchestration, quality of service and result verification. The directions are summarized in Table 2.1.

Resource management and task scheduling have been investigated in [27]. The authors investigate the operational constraints and costs for managing resources and applications through a Smart Contract and analyze the impact in terms of latency and gas usage of three scheduling methods. The authors conclude that the best approach is to negotiate the resource selection off-chain and use the Smart Contract to seal the agreement. The other option is for the Smart Contract to handle the matchmaking algorithm, which proves to be expensive both in terms of gas and in terms of time until the transaction is mined. A different paper, [13] approached the problem via the Serverless computing paradigm, but by using Hyperledger Fabric the team did not focus on the economic cost of deploying the solution to a public blockchain. have been proposed in [34, 33, 32], but do not provide the same level of fine-grained evaluation analysis.

Several platforms making use of the Blockchain to verify data storage and transfers have been proposed, making use of known protocols like BitTorrent or IPFS. FileCoin [2] and others [18, 7, 26], use IPFS as the backbone for file storage and transfer and apply diverse logic applicable for different purposes. FileCoin makes use of a Proof-of-Replication mechanism [11] to decide that a node has spent some space for a given amount of time. The solutions presented in [18] and [26] propose a secure document communication protocols based on IPFS that uses the blockchain for establishing communication channel properties. In [7], the authors distinguish between hot and cold data and apply different replication mechanisms depending on their type. The blockchain is used for data access management and to keep track of the nodes holding the data. We further refer the reader to [12], for a comparison between centralized and blockchain-based decentralized storage solutions.

Service Orchestration intermediated by the blockchain has been investigated in a few papers. In [29], a Component Administration Network stores Orchestrator checkpoints which ensure the continuity of Application Deployment in the presence of Orchestrator failures. A decentralized, modular platform has been presented in [28] and investigated in detail. The aforementioned work has focused on the management and fault tolerance of the Fog platform, which can later ensure the fault tolerance of the running applications. A series of Smart Contracts are used to manage resource selection and service deployment. A Registry Contract maintains a catalogue of resource managers (local Clouds, ad-hoc clusters) that a customer can query for specific resources. The resource manager can create an Application Contract which is used by an Orchestration component to deploy and monitor the status of the services composing the application. Orchestration components periodically update the Application Contract regarding the status of the services, thus ensuring a fair payment. Some nodes part of the Blockchain peer-to-peer network will form the administration network instead of mining. These nodes check on each other's availability and on any components that are running (e.g. Orchestrator). Components are run by a subset of the administration nodes and store checkpoints on this network using a distributed file system. All nodes that contribute to the running and monitoring of an application are reimbursed for their contribution using the checkpoints (checkpoint metadata is stored in the application Smart Contract).

Several works have tackled the Quality of Service (QoS) aspect of the deployed applications. Some of the solutions (e.g. [14]) depend on external oracles which are trusted parties for monitoring the QoS. Alternately, the monitoring job can be delivered by the crowd. The concept of "Crowd-based Oracle-as-a-service For Consensus On Qos Monitoring" presented in [31] is similar to the Component Administration Networks presented in [29], making use of a Smart Contract to manage a network of trusted peers. The advantages of the platform presented in [31] and [14] are represented by the usage Service Level Agreements (SLA) and Service Level Objectives (SLO), while the solution presented in [29] only accounts for the service being responsive for a given amount of time.

In the case of batch tasks, several papers investigate the correctness of the result. Sarmenta's approach to deal with saboteur nodes using credibility-based fault tolerance [23] is currently the base of the Proof of Computation protocol (PoCo) at the heart of the iExec platform (described further below). Another approach is to use zero-knowledge proofs to verify the results of computation by providing a small proof to a Smart Contract [15]. The introduction of the Intel SGX technology [8] paved the way for Trusted Execution Environments to become popular. A hardware key is used for ensuring no corruptions have been made to the code, while memory is encrypted to prevent a root user of the machine to access the computation data.

3. Platforms. Several platforms that offer service deployment through the means of Blockchain exist. Ethereum itself is taught as the *world computer*, though the capabilities of storing data and execution are drastically limited by the price of smart contract operations. Thus, platforms rely on using the Ethereum Blockchain in order to create tokens for their platform, and raise investment funds through *Initial Coin Offerings* (ICOs).

Golem[1] uses IPFS [3] as the means to distribute file blocks in a network of worker nodes which process data at the block level and later collect and merge the results computed in parallel. The platform intended to offer Software as a Service, but starting in 2020 the focus was shifted to Platform as a Service, providing an SDK for creating Golem applications.

SONM[2] uses Docker for executing Container Images and achieves a higher level of abstraction, getting close to a generic Cloud platform. An Ethereum side chain is used to manage the orders. *Suppliers* must

[1]https://golem.network/
[2]https://docs.sonm.com/concepts/main-entities

Table 3.1: Blockchain based Cloud Platforms

Name	**Blockchain**	**Market**	**Storage**	**Services**
Golem	Ethereum	No	No	PaaS
SONM	Ethereum	Yes	Yes	IaaS
iExec	Ethereum	Yes	Yes	PaaS
Decenter	Ethereum	Yes	External	IaaS

Table 3.2: iExec entities

iExec Hub and Marketplace	an auditable smart contract used to manage the stakes and keep track of the history of the actors.
Dataset providers	individuals which will sell access to their data on the platform.
Application providers	individuals which will deploy applications on the platform; applications can be free or ask for a price.
Workers	individuals or companies which expose their resources on the marketplace.
Worker pools	smart contracts to which workers can subscribe; the smart contract will take care of workers contributions and will receive fees for managing the underlying infrastructure. This contract stores scheduler settings, required in order to handle the stake and payments.

interact with this chain to instantiate worker nodes that will act on their behalf. Resources (CPU, RAM, storage, bandwidth) are exposed using benchmark identifiers such as GFLOPS, IOPS, etc. The platform considers two delivery models: fixed-time or pay-as-you-go. There are no peer-reviewed experiments presenting the performance of the platform, and the limited amount of documentation does not present implementation details.

iExec makes use of Xtremweb middleware [10, 19] to handle task placement and the validity of results. The platform logic is implemented using a side chain and the public Ethereum Blockchain is used to create and handle the platform tokens. The platform defines several entities that are shown in Table 3.2.

A *Proof of Contribution* (*PoCo*) protocol is used for acknowledging the correct result of an Application, using the sabotage tolerance introduced in [23]. The *PoCo* links two entities: the iExec marketplace (where deals are made) and the computing infrastructure (based on XtremWeb-HEP middleware [10]).

DECENTER is a Horizon 2020 financed project aiming at providing a federated brokering platform for fog resources [24]. Their proposed architecture is centred around the Resource Exchange Broker (REB) Smart Contract. Resource providers deploy an REB Contract which manages the selection of resources and signals Orchestration Components to deploy applications. A resource provider is required to have installed a full Cloud management software stack: infrastructure management and provisioning, together with service orchestration components.

Recent publications present an architecture that allows for the definition of Service Level Agreements (SLAs) using Smart Contracts [14]. Quality of Service (QoS) parameters such as network throughput, or the latency between different tiers of the same Application are then used by a decision-making layer, which is composed of the monitoring and orchestration components.

4. Conclusion. This report has investigated the current efforts in the direction of Blockchain-based platforms offering SaaS, PaaS, IaaS. The most difficult problem is the verification of work, generally implying one of two options: work replication, or workload monitoring. The first option is employed by iExec and uses the results computed by multiple replicas to decide on the validity of a result. This, however, implies a batch model for the Application in order to decide and this is a requirement that cannot be satisfied by today's Cloud Services, which are generally interactive.

Golem is using monitoring, but the monitoring metrics are self-reported by the node participating in the network. This raises security concerns as the metrics can be fabricated by the node. SONM and DECENTER choose to dedicate some nodes for running the monitoring agents that inspect the worker nodes and applications running on top of them, yet they consider these to be trusted by default. Instead, research efforts are pushing the boundaries by focusing on fault-tolerant mechanisms for ensuring the quality of work execution and ensuring the fair payment of all entities taking part in managing the platform.

There is still a limited amount of experimental research that compares the performance and scalability of Blockchain-based systems and components that expose computational resources. This is mostly due to the infancy of the domain, researchers focusing on publishing architectures and protocols as the first materials. New simulation platforms need to be implemented to tackle the vast distribution of future services and their interconnectivity.

An important future direction for Blockchain-mediated Fog service delivery should focus on new standards for the deployment and monitoring of applications spanning the Cloud Continuum. Future applications will be represented by workflows that start at the Edge of the network (processing raw data obtained from sensors), flow through the Fog (aggregating data from multiple Edge devices) and reach the Cloud (for archival, analytics, visualisation). Adaptation of existing standards can increase the chances for a blockchain-based Fog network to be integrated with the Cloud Continuum.

Acknowledgment. This work was supported by a grant of the Romanian Ministry of Education and Research, CNCS - UEFISCDI, project number PN-III-P4-ID-PCE-2020-0407, within PNCDI III.

REFERENCES

[1] David P Anderson. Boinc: A system for public-resource computing and storage. In *proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, pages 4–10. IEEE Computer Society, 2004.

[2] J Benet and N Greco. Filecoin: A decentralized storage network. *Protoc. Labs*, pages 1–36, 2018.

[3] Juan Benet. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*, 2014.

[4] Vitalik Buterin et al. Ethereum white paper, 2014. *URL https://github. com/ ethereum/wiki/wiki/White-Paper*, 2013.

[5] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems (TOCS)*, 20(4):398–461, 2002.

[6] Bernadette Charron-Bost, Fernando Pedone, and André Schiper. Replication. *LNCS*, 5959:19–40, 2010.

[7] Yongle Chen, Hui Li, Kejiao Li, and Jiyang Zhang. An improved p2p file system scheme based on ipfs and blockchain. In *Big Data (Big Data), 2017 IEEE International Conference on*, pages 2652–2657. IEEE, 2017.

[8] Victor Costan and Srinivas Devadas. Intel sgx explained. *IACR Cryptol. ePrint Arch.*, 2016(86):1–118, 2016.

[9] MR Dorsala, VN Sastry, and S Chapram. Blockchain-based solutions for cloud computing: A survey. *Journal of Network and Computer*, 2021. Query date: 2021-12-03 19:28:36.

[10] Gilles Fedak, Cecile Germain, Vincent Neri, and Franck Cappello. Xtremweb: A generic global computing system. In *Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on*, pages 582–587. IEEE, 2001.

[11] Ben Fisch, Joseph Bonneau, Nicola Greco, and Juan Benet. Scaling proof-of-replication for filecoin mining. *Benet//Technical report, Stanford University*, 2018.

[12] T. Gabriel, A. Cornel-Cristian, M. Arhip-Calin, and A. Zamfirescu. Cloud storage. a comparison between centralized solutions versus decentralized cloud storage solutions using blockchain technology. In *2019 54th International Universities Power Engineering Conference (UPEC)*, pages 1–5, 2019.

[13] Sara Ghaemi, Hamzeh Khazaei, and Petr Musilek. Chainfaas: An open blockchain-based serverless platform. *IEEE Access*, 8:131760–131778, 2020.

[14] Petar Kochovski, Vlado Stankovski, Sandi Gec, Francescomaria Faticanti, Marco Savi, Domenico Siracusa, and Seungwoo Kum. Smart contracts for service-level agreements in edge-to-cloud computing. *Journal of Grid Computing*, pages 1–18, 2020.

[15] Benjamin Körbel, Marten Sigwart, Philip Frauenthaler, Michael Sober, and Stefan Schulte. Blockchain-based result verification for computation offloading. In Hakim Hacid, Odej Kao, Massimo Mecella, Naouel Moha, and Hye-young Paik, editors, *Service-Oriented Computing*, pages 99–115, Cham, 2021. Springer International Publishing.

[16] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[17] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *International Workshop on Peer-to-Peer Systems*, pages 53–65. Springer, 2002.

[18] Mruthyunjaya Mendu, B Krishna, Sallauddin Mohmmad, Y Sharvani, and Ch Vinay Kumar Reddy. Secure deployment of decentralized cloud in blockchain environment using inter-planetary file system. In *IOP Conference Series: Materials Science and Engineering*, volume 981, page 022037. IOP Publishing, 2020.

[19] Mircea Moca, Cristian Litan, Gheorghe Cosmin Silaghi, and Gilles Fedak. Multi-criteria and satisfaction oriented scheduling for hybrid distributed computing infrastructures. *Future Generation Computer Systems*, 55:428–443, 2016.

[20] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *URL https:// bitcoin.com/bitcoin.pdf*, 2008.

[21] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM (JACM)*, 27(2):228–234, 1980.

[22] Johan Pouwelse, Paweł Garbacki, Dick Epema, and Henk Sips. The bittorrent p2p file-sharing system: Measurements and analysis. In *International Workshop on Peer-to-Peer Systems*, pages 205–216. Springer, 2005.

[23] Luis FG Sarmenta. Sabotage-tolerance mechanisms for volunteer computing systems. *Future Generation Computer Systems*, 18(4):561–572, 2002.

[24] Marco Savi, Daniele Santoro, Katarzyna Di Meo, Daniele Pizzolli, Miguel Pincheira, Raffaele Giaffreda, Silvio Cretti, Seung-woo Kum, and Domenico Siracusa. A blockchain-based brokerage platform for fog computing resource federation. In *2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 147–149. IEEE, 2020.

[25] Fred B Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys (CSUR)*, 22(4):299–319, 1990.

[26] Meet Shah, Mohammedhasan Shaikh, Vishwajeet Mishra, and Grinal Tuscano. Decentralized cloud storage using blockchain. In *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184)*, pages 384–389. IEEE, 2020.

[27] Adrian Spataru, Laura Ricci, Dana Petcu, and Barbara Guidi. Decentralized cloud scheduling via smart contracts. operational constraints and costs. In *The International Symposium on Blockchain Computing and Applications (BCCA2019)*, 2019.

[28] Adrian Spătaru. *Decentralized Cloud Computing*. PhD thesis, 2021.

[29] Adrian Spătaru. Decentralized and fault tolerant cloud service orchestration. *Scalable Computing: Practice and Experience*, 21(4):709–725, 2020.

[30] Rafael Brundo Uriarte and Rocco De Nicola. Blockchain-based decentralized cloud/fog solutions: Challenges, opportunities, and standards. *IEEE Communications Standards Magazine*, 2(3):22–28, 2018.

[31] Rafael Brundo Uriarte, Huan Zhou, Kyriakos Kritikos, Zeshun Shi, Zhiming Zhao, and Rocco De Nicola. Distributed service-level agreement management with smart contracts and blockchain. *Concurrency and Computation: Practice and Experience*, 33(14):e5800, 2021.

[32] Zehui Xiong, Shaohan Feng, Wenbo Wang, Dusit Niyato, Ping Wang, and Zhu Han. Cloud/fog computing resource management and pricing for blockchain networks. *IEEE Internet of Things Journal*, 6(3):4585–4600, 2018.

[33] Chenhan Xu, Kun Wang, and Mingyi Guo. Intelligent resource management in blockchain-based cloud datacenters. *IEEE Cloud Computing*, 4(6):50–59, 2017.

[34] H. Zhu, Y. Wang, X. Hei, W. Ji, and L. Zhang. A blockchain-based decentralized cloud resource scheduling architecture. In *2018 International Conference on Networking and Network Applications (NaNA)*, pages 324–329, 2018.

AIMS AND SCOPE

The area of scalable computing has matured and reached a point where new issues and trends require a professional forum. SCPE will provide this avenue by publishing original refereed papers that address the present as well as the future of parallel and distributed computing. The journal will focus on algorithm development, implementation and execution on real-world parallel architectures, and application of parallel and distributed computing to the solution of real-life problems. Of particular interest are:

Expressiveness:
- high level languages,
- object oriented techniques,
- compiler technology for parallel computing,
- implementation techniques and their efficiency.

System engineering:
- programming environments,
- debugging tools,
- software libraries.

Performance:
- performance measurement: metrics, evaluation, visualization,
- performance improvement: resource allocation and scheduling, I/O, network throughput.

Applications:
- database,
- control systems,
- embedded systems,
- fault tolerance,
- industrial and business,
- real-time,
- scientific computing,
- visualization.

Future:
- limitations of current approaches,
- engineering trends and their consequences,
- novel parallel architectures.

Taking into account the extremely rapid pace of changes in the field SCPE is committed to fast turnaround of papers and a short publication time of accepted papers.

INSTRUCTIONS FOR CONTRIBUTORS

Proposals of Special Issues should be submitted to the editor-in-chief.

The language of the journal is English. SCPE publishes three categories of papers: overview papers, research papers and short communications. Electronic submissions are preferred. Overview papers and short communications should be submitted to the editor-in-chief. Research papers should be submitted to the editor whose research interests match the subject of the paper most closely. The list of editors' research interests can be found at the journal WWW site (`http://www.scpe.org`). Each paper appropriate to the journal will be refereed by a minimum of two referees.

There is no a priori limit on the length of overview papers. Research papers should be limited to approximately 20 pages, while short communications should not exceed 5 pages. A 50–100 word abstract should be included.

Upon acceptance the authors will be asked to transfer copyright of the article to the publisher. The authors will be required to prepare the text in LaTeX 2ε using the journal document class file (based on the SIAM's `siamltex.clo` document class, available at the journal WWW site). Figures must be prepared in encapsulated PostScript and appropriately incorporated into the text. The bibliography should be formatted using the SIAM convention. Detailed instructions for the Authors are available on the SCPE WWW site at `http://www.scpe.org`.

Contributions are accepted for review on the understanding that the same work has not been published and that it is not being considered for publication elsewhere. Technical reports can be submitted. Substantially revised versions of papers published in not easily accessible conference proceedings can also be submitted. The editor-in-chief should be notified at the time of submission and the author is responsible for obtaining the necessary copyright releases for all copyrighted material.