Scalable Computing: Practice and Experience
Volume 12, Number 2, pp. 193–208. http://www.scpe.org
ISSN 1895-1767

OPTIMIZING DATA DISTRIBUTION IN VOLUNTEER COMPUTING SYSTEMS USING RESOURCES OF PARTICIPANTS

ABDELHAMID ELWAER*, IAN J. TAYLOR, AND OMER RANA

Abstract. Many scientific projects use BOINC middleware to build a volunteer computing project. BOINC uses centralized data servers to distribute data to its users and some projects can attract thousands of participants. Such large numbers of users coupled with large datasets can cause a bottleneck for the centralized organization of the BOINC data servers, which has a knock-on effect on the performance of the project as a whole by limiting the throughput of jobs. Alternative methods have been proposed, such as the Attic file system, which decentralize data distribution to BOINC participants. This has been shown to scale but does not attempt to optimize the use of the various distributed data centres being used. We describe performance techniques based on trust algorithms that when layered on the Attic file system can significantly improve data availability and access time through intelligent selection of the data center for each user, based on the optimization of three parameters: trust, current connection speed and availability.

Key words: volunteer computing , P2P, data distribution, trust

AMS subject classifications. 15A15, 15A09, 15A23

1. Introduction. Volunteer computing systems provide a useful infrastructure for supporting scientific discovery, examplified through systems such as SETI@Home [16], Einstein@Home [18] , Entropia [17], PlanetLab and recently the Berkeley Open Infrastructure for Network Computing (BOINC) [1]. Each of these systems provide middleware to enable inclusion of donated resources into a particular scientific project. Such distributed computing middleware makes use of idle CPU time of donated volunteered resources within the system. BOINC, for instance, is based on a client/server architecture, where the server generated jobs are pulled onto the clients for local execution and the results are uploaded back to the server upon completion. Although job execution may be supported on remote clients in BOINC (with each job being replicated multiple times to improve reliability), only limited data management support is provided within such systems. The BOINC middleware uses a fixed set of centralized servers to provide data to each client. As some volunteer computing projects may attract thousands of participants (table 1.1), access to a single data server becomes a bottleneck. Additionally, the BOINC data server cannot manage data distribution to all of these participants concurrently, especially when their bandwidth is limited. The BOINC middleware is therefore primarily designed for CPU-intensive jobs, by using the idle CPU time of its participants, but it does not take account of the bandwidth and storages capabilities of the contributing participants.

Projects	Users
SETI@Home	1,174,317
World Community Grid	337,456
Roseta@Home	314,822
Einstein@Home	292,175
Climate Prediction	245,355
MilyWay@home	95,492
ABC@Home	45,734
QMC@Home	45,433
PrimeGrid	40,241
SIMAP	36,693

Table 1.1: The top BOINC projects (from [14])

To overcome this bottleneck and improve data distribution, alternative approaches using Peer-to-Peer (P2P) networks have been proposed. A P2P approach, being far more decentralized in nature, employs a distributed data layer for the participants of the network and can provide far better data-throughput for the project. This

*School of Computer Sciences & Informatics, Cardiff University, Cardiff ,UK, Corresponding Author (a.elwaer@cs.cardiff.ac.uk).

data layer can be extended dynamically, as each volunteer can also now play the role of a data storage server (referred to as a data centre), and thereby also limit the static nature of BOINC servers. Data centres are therefore interim distributed storage facilities that provide a buffer between the data serving application and the client applications. The availability of multiple such data centres reduces access time and improves resilience of the system (at the cost of ensuring that all copies of the data are consistent).

The participants of volunteer computing projects have limited bandwidth and access speed. Furthermore, they are unlikely to overload their internet connection by serving a large number of data requests, due to the availability of limited bandwidth and also due to other applications that they may wish to run concurrently. We propose a framework built on top of the Attic File System (AtticFS) [13] that can offer file-sharing facilities that can control bandwidth and limit the number of connections a participant can serve, as well as the *keep alive* time of these connections. In addition, the framework supports data provisioning and deals with various distributed network conditions that can occur during the distributed allocation of multiple data centres. For example, it is possible for data to become corrupted (with or without the intent of the volunteered resource owner); data centres can also have different upload speeds and their availability can change over time; servers may also become unresponsive or unavailable during the operation. Our framework addresses these issues and allows a client to decide which data centre to download data from (based on preferences identified by the client) using a trust model.

The rest of this paper is organized as follows. Section 2 provides a background to the problem. Section 3 covers related work in P2P data management and trust and Section 4 contains a discussion about the requirements for a system that meets the objectives outlined in the introduction above. Section 5 describes the system architecture and Section 6 describes the integration of the AtticFS system with BOINC. Section 7 contains a description of the associated trust framework and Section 8 contains a number of experiments used to evaluate the framework, by comparing it with a system that does not support trust using the core AtticFS [13] system. We demonstrate that the use of the trust framework makes the resulting volunteer computing environment more predictable and reliable.

2. Background. The context for the approach discussed in subsequent sections is presented. A brief overview of volunteer computing and AtticFS is provided in subsections 2.1 and 2.2 respectively.

2.1. Volunteer Computing. Volunteer computing is a relatively new distributed computing paradigm; it uses computers which are connected to the Internet and donated by their owners for distributed computing application, using the computing and storage capability of each computer. Volunteer computing is currently used in many scientific projects. SETI@home [16] is a scientific project for analyzing radio-telescopic signals to attempt to detect extra-terrestrial activity. Einstein@home [18] is a project to search for spinning neutron stars using data from the LIGO and GEO gravitational wave detectors. Climatprediction.net [19] is a scientific project which aims to produce forecasts of the climate in the 21st century. It makes use of a functional programming approach to analyse large volumes of data. In April 2011, these and other scientific projects using BOINC middleware have recorded 5,620 Teraflops through 2,167,503 users, and the SETI@home project has more than 1,174,317 participants, with a sustained performance of about 533 TeraFlops.

2.2. Attic File System. AtticFS is a decentralized, P2P data sharing software architecture for accessing distributed storage resources available over a network in a similar way to BitTorrent. The AtticFS consists of four main elements: (i) a data serving application that replicates data on the network; (ii) data centres that cache data, providing the distributed data source overlay; (iii) a look up service that keeps track of which data centres have individual data items, and (iv) client applications that download data from data centres on the network. The primary differences between AtticFS and BitTorrent are the concept of data centres and the use of HTTP. Since AtticFS is a distributed P2P file serving system, some peers can play the role of data centres to distribute data to other peers in the network.

Data centres are interim storage facilities that provide a buffer between the data serving application and client applications. This buffer is particularly important for volunteer computing environments because it ensures that the data sources can be trusted by clients. Trust plays a crucial role in volunteer computing environments (and the number of volunteers involved in a project can determine its success or failure), as a client may often need to interact with or download data from a peer that may exist for a short period of time and may be operated by an unknown provider. If trust is broken (i.e. the providing peer does not behave as expected) then the project will fail to attract volunteers and become unsuccessful. Verifying the identity of a peer is also an important concern in this context, hence in *controlled/closed* environments, data centres are

typically issued with certificates signed by a trusted certificate authority (e.g. the project itself) allowing client applications to verify the identity of the data centre when downloading. Alternatively, clients can be configured with known data centre hosts. However, the AtticFS architecture allows any client to also serve data (i.e., become a data centre) and thus the use of trusted data centres is primarily a deployment choice.

Attic is particularly suitable because it has been designed with volunteer computing in mind, and does not require all participants to share data, or share equally, therefore differing it from other types of P2P file-sharing systems e.g. BitTorrent, etc. Attic provides not only a level of security for the data sharing hosts and a managed lookup facility to coordinate data mirrors and results while minimizing stale information, but also, unlike most P2P systems, Attic allows client nodes to forgo "tit-for-tat" algorithms to receive data. This is useful in volunteer computing systems, because many data receiving hosts will not be actively sharing input data with the network (as is required in many P2P systems in order to continue receiving data) for various security reasons. Instead, these clients are acting solely as data recipients and are using the P2P system only to receive data from (potentially) multiple locations. This scenario is of course perfectly valid for BOINC-like projects because these clients are contributing to the overall project by providing CPU cycles, and therefore are not required to also share data to be useful. Further, the use of HTTP for all AtticFS transactions allows easy integration with existing software and firewall configurations and client applications that choose not to server data require only an out-going HTTP connection.

3. Related Work. Two aspects are discussed in this section: data management in P2P systems and the general notion of trust in distributed systems. In section 3.1 we outline related work with reference to the BOINC system also covering related commercial deployments based on BitTorrent and Amazon S3. In section 3.2 we identify how trust has been used to support service provisioning and how the various views on trust can be incorporated into a data management system.

3.1. Desktop Grids, Data Management and P2P Systems. BOINC contains a scheduling server and a client program that is installed on user machines. The client software periodically contacts the scheduling server to report its hardware and availability, and in response receives a *work unit* for downloading and executing a job. Once the client completes the work, it uploads resulting output files to the scheduling server and requests more work. For data distribution, BOINC projects generally use a single centralized server or, in the case of more popular projects, a set of server mirrors, which all have to be configured with the BOINC stack. The centralized nature of this system incurs additional costs on BOINC projects, in both hardware cost and administration and can quickly become inefficient, especially as replication factors increase and many tasks begin to share the same input files.

XtremWeb [2], like BOINC, is a software system that gathers unused resources on donated computers and makes them available to scientific projects. Unlike BOINC, which has centralized servers and users subscribe to a relatively static set of projects for which they will donate their CPU time, XtremWeb allows for multiple-users and multiple (changing) applications to run concurrently on the system. XtremWeb provides a platform on which scientists and volunteers can share their respective resources with one another, and it is often the case that providers of resources are also the users of the system. If one were to think of BOINC as a large, stable, managed, and centralized volunteer computing system for distributing workloads, XtremWeb would be its smaller P2P counterpart that organizes resources in an ad-hoc manner and allows the running of arbitrary code provided it has been signed and verified by a trusted party. Similar to BOINC, work units in XtremWeb are provided with the URIs of input files, and these are downloaded as a preprocessing step when a client job is launched.

There are many popular and proven P2P technologies such as BitTorrent [5], or commercial solutions like Amazon S3 or Google GFS [3], that could be promising alternatives to the current data handling in BOINC and XtremWeb. However, in the case of commercial products, there is a direct monetary cost involved that is often difficult to justify and fund for many scientific projects. Likewise, in P2P systems like BitTorrent, KaZaa, or FastReplica [4], the facility to easily secure or limit who is able to receive, cache, or propagate different pieces of data within the same network is generally limited or non-existent. Like most other P2P systems, these systems discussed above have focused on ensuring conventional file-sharing features that appeal to the broader user-base, such as efficient transfers, equal sharing to promote network stability and availability, and file integrity.

Desktop Grids, and in particular, volunteer computing environments, have significantly different requirements to more conventional P2P file-sharing because security is more complex. In Desktop Grids, it can be a prerequisite that only certain amounts of data are shared with an individual peer, or that strict bandwidth

caps are enforced to guarantee that data-sharing volunteers' Internet connections do not become saturated. It is important to support both data integrity and reliability, whilst also providing safeguards that can limit a peer nodes' exposure to malicious attacks. Furthermore, certain project also require peers that share data to be restricted to certain trusted entities to ensure the distribution can be relied upon. In this paper, we attempt to automate this process and provide safeguards and trust mechanisms that can reliably assess each data server and provide strong cues to the client for choosing a consistent data server at each period of time.

3.2. Trust. The general notion of trust is excessively complex and appears to have many different meanings depending on how it is used in electronic service provisioning. There is also no consensus in the computer and information sciences literature on a common definition of trust, although its importance has been widely recognized and the literature available on trust is substantial. Generally, trust may be used as a metric to guide an agent in deciding how, when and who to interact with. An agent in this context refers to either a service user or a provider. Such a metric takes into account the subjective probability with which an agent views its interaction partners, taking into account local state and external recommendations made by other agents. To establish a trust mode, agents must gather data about their counterparts. This has been achieved in three ways in the literature: (i) using prior interaction experience: in this context, trust is computed as a rating of the level of performance of the trustee using historical data. The trustee's performance is assessed over multiple interactions to check how good and consistent it is at doing what it says it does. Interactions that have taken place recently are treated preferentially to those that have taken place in the distant past. Witkowski et al. [6] propose a model whereby the trust in an agent is calculated based on its performance in past interactions.

Similar to Witkowski et al., Sabater et al. [7] (using the REGRET system) propose a similar model but do not just limit the overall performance to the agent's direct perception, but they also evaluate its behavior with other agents in the system; (ii) information gathered from other agents: trust in this approach is drawn indirectly from recommendations provided by others. As the recommendations could be unreliable, the agent must be able to reason about the recommendations gathered from other agents. The latter is achieved in different ways: (1) deploying rules to enable an agent to decide which other agents' recommendation they give greater preference, as introduced by Abdul-Rahman et al. [8]; (2) weighting the recommendation by the trust the agent has in the recommender, EigenTrust [9] and PageRank [10] are examples of this approach. In both of these approaches, the connectivity graph between recommenders is used to infer trust. Generally, an agent that has successfully delivered its advertised capability and recommends another agent will cause some of its trust to be transferred to its recommended agent.

Both PageRank and EigenTrust are therefore based on the assumption that a general user, searching over a set of possible service providing peers, will eventually end up finding a more trustworthy peer if they follow the recommendation chain from any point in the network. The PowerTrust [11] model works in a similar way to EigenTrust, focusing on creating overlay hashing functions to assign score managers (i.e. peers who calculate trust values for other peers, thereby preventing peers from maliciously changing their own trust values) for peers in the system and for combining trust values to create a global reputation score. Both of these approaches have limited benefit when considering multiple criteria (as considered in this work and as outlined in section 3) when calculating trust, i.e., both EigenTrust and PowerTrust are focused on searching for objects using a single keyword, such as a file name; (iii) socio-cognitive trust: in this context, trust is drawn by characterizing the known motivations of the other agents. This involves forming coherent beliefs about different characteristics of these agents and reasoning about these beliefs in order to decide how much trust should be put in them. An example of this is work by Castelfranchi [12]. Our focus in this work is primarily on characteristics (i) and (ii) defined above. We used historical data to establish a trust model for a given data centre, and use three metrics (honesty, availability and speed) to evaluate the level of trust that one can place in a data centre. We consider reputation to be an aggregated community view about a data provider, i.e. the greater the number of participants who trust a data centre, the greater the reputation the data centre holds.

4. System Requirements. We identify a set of requirements for a framework that makes use of resources provided by participants within a volunteer computing project. The core aim is to subsequently provide a tool based on this framework that is usable by BOINC project participants, allowing them to contribute as a data centre (data distributor) in the decentralized data centre layer, a data worker, or both. Various subsections below identify the components of the framework.

4.1. Data Caching. In BOINC projects, clients contact the scheduler server to get jobs to execute on their local resources, then request input data files from a data server to process the downloaded job, and finally

upload the results. The input data files on a BOINC client can be obtained by using dynamic caching. This increases data availability and improves fault tolerance. Moreover, dynamic caching avoids the bottleneck in the BOINC data server, because data can now be downloade from different places rather than just the data server. Furthermore, it reduces data download time as client can concurrently download data from these different sources. To form such a decentralized data centre layer, AtticFS has been used to add caching functionality to the BOINC client, enabling a BOINC client previously capable of only processing jobs to also be able to cache data and provide it to other clients.

4.2. Trust. The participants of BOINC projects are ordinary internet users, who have different behaviours and connection capability. Therefore, to utilize their resources effectively for data distribution, some optimization or trust mechanisms are needed. In our work we use a trust mechanism that makes use of particular properties of a data centre, such as it's bandwidth, connection speed, availability and other preferences that may be chosen by a client (provided that data associated with these preferences is recorded). The trust mechanism is subsequently used to select one or more data centres to download data from, based on preferences identified by a client.

4.3. Data Management. The formation and use of a decentralized data centre layer requires the consideration of two key issues:

1. **Data Source** For a BOINC client to become a data centre, it has to cache data that is initially provided by the main BOINC data server. When the BOINC client downloads data to process its job, it will cache this data to be available for other clients, who process the same job, thereby propagating the dataset on demand. Here, the BOINC data server is made the primary source of data when data is not available on the data centre layer. When data becomes available on other clients this will extend the source of data and the BOINC client can use them to download data and cache it.
2. **Data Downloading** As the data centres are ordinary internet users (who may be connected to the network using a variety of connection types, e.g. dial-up, DSL, wireless, etc), they can frequently become unavailable to provide data to other BOINC clients. This transient connectivity therefore needs to be addressed in the download algorithm and available data centres need to be dynamically updated as the network evolves. It must also deal with the case that no data centres are available, in which case, the BOINC client should switch to the main source e.g. BOINC data server to get data.

The general scheme for downloading data is outlined in algorithm 4.3.1, which shows that the trust algorithm provides the intelligence for each client to determine the best data centre at that point in time. Each client updates its empirically gathered parameters to feed back into the trust algorithm for the next iteration. In this way, the system learns dynamically to deal with the changing network conditions.

Algorithm 4.3.1 Download Algorithm

```
1: if data available in data centre layer then
2:     Identify trusted data centres using trust algorithm
3:     Download data
4:     Generate feedback for server
5: else
6:     Use BOINC data server for data download
7: end if
8: Cache data locally
9: Register data location with server
```

4.4. Bandwidth Throttling. Many internet users have limited bandwidth and may not be interested in offering this to distribute data – as this can slow their own use of the internet. This framework therefore must offer a throttle capability to a client on its bandwidth use e.g., if it has 1 MB/s connection it can offer 256KB/s to other clients for downloading data, thereby enabling a user to better plan how their capacity will be shared between other users. This enables a volunteered resources to continue to participate in the framework, whilst also enabling a resource owner to continue their own work. We believe such a mechanisms for bandwidth sharing is likely to increase contribution of resources to a project.

5. System Architecture. Our trust system has four general layers, which are provided in Figure 5.1. The bottom layer represents the participants of scientific projects who provide their resources to volunteer

computing projects. The next layer provides the P2P network capability – here we make use of AtticFS. This layer provides the core capability for volunteers to share data with other participants.

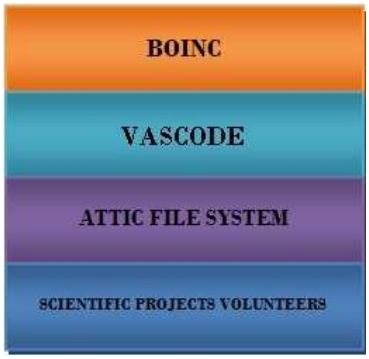

Fig. 5.1: FRAMEWORK LAYER

The third layer provides the *Volunteered Atomic Servers for Data Collection and Optimization in Distributed Environments* (VASCODE) trust framework for choosing he location of data download at each time step. In layer three therefore we integrate a data collection layer, which provides peers the necessary tools to select from which data server to download data, which peers to trust, and the throttling capabilities to manage their bandwidth. This layer is described in detail in the section 5.1.

Finally the last layer represents BOINC, which is the programming and Web interface that a project interacts with in order to the use the system. Since both AtticFS and VASCODE deal with HTTP endpoints, a simple `attic` URL scheme plug-in can be used to switch out the general BOINC URL endpoint with a dynamic AtticFS one to provide multiple possible endpoints for each dataset. Therefore, a project does not need to be aware of the use of VASCODE and AtticFS in order to have it enabled as the data distribution mechanism. The integration of AtticFS and VASCODE into existing projects is possible through a plug in that proxies the HTTP connections and resolves multiple endpoints available in AtticFS.

5.1. VASCODE. VASCODE is a layer built upon the capability provided by AtticFS to add the necessary functionality to its peers (e.g. worker or data centre) when they download data or distribute it. Figure 5.2 provides an overview of this functionality. Essentially, VASCODE provides the trust-based mechanisms within AtticFS, enabling the better integration of various components that make up the AtticFS system into BOINC. VASCODE enables user defined preferences to be taken into account when selecting data centres to download from, based on previous usage data that has been recorded about these data centers.

The three user defined preferences employed in this work to demonstrate the concept include: availability, data integrity (i.e. establishment of trust in that the data has not been altered) and connection speed of each data centre. Each peer in the system provides feedback on each interaction they have had with a data centre. This data is collected (through the data lookup nodes) and used in the trust algorithm to feedback into a selection criteria a client uses, to determine the most appropriate sets of data centres to use. By using this mechanism, the system continuously (i.e. on the completion of each interaction with a data centre) updates the aggregate statistics for each data centre. We can modify whether the aggregation must be undertaken after each interaction, or only after a pre-defined number of interactions, in order to reduce the computational overhead of calculating the aggregate value.

Fig. 5.2: VASCODE Layer on top of AtticFS

Figure 5.2 shows the three possible roles that a peer can perform in the BOINC-VASCODE integration: a conventional data worker, a data centre or a data lookup server. Further, each of these roles has various properties. For example, when the BOINC client is a data worker, it must also be capable of accessing the data centre layer. It also needs to make use of VASCODE to calculate the trustworthy of these data centres and select which data centres to get data from. Furthermore, it needs to then provide feedback to help other participants determine which is the best data centre at this point in time.

A BOINC client can use the VASCODE framework to interact with these functions. If a BOINC client wants to perform data centre capabilities, then they need to provides attributes such as, (i) how much bandwidth it wants to offer for data distribution, (ii) the maximum number of connections it will serve, and (iii) the keep alive time for these connections, for the general configuration of the data centre. Finally, when the BOINC client plays the role of a data lookup server, it is important to have the ability to collect feedback from clients and distribute these to other clients for use by the network as a whole. These capabilities are also offered through our VASCODE layer.

6. Integrating AtticFS with BOINC Projects. AtticFS uses a unique *ID* to identify data instead of using a file name. Data is first published using AtticFS to obtain the new ID and then the resolved `attic` URL using this ID is deployed in the generated BOINC work unit. A snapshot of a `DataDescription` of data published in the AtticFS is shown below:

```
<DataDescription xmlns="http://Attic.org">
    <id>
        f050a833-2fac-44c9-9b08-7c
    </id>
    <name>
        file10MB.dat
    </name>
            ......
            ......
</DataDescription>
```

The download URL in a work unit is changed to use an `attic` URL scheme instead of `http`. This indicates to the BOINC client when it parses the workunit description to use AtticFS to resolve the endpoint that will be used to download the actual data. This is illustrated in Figure 6.1.

1. The BOINC client contacts the Task Server to get a workunit.
2. Then it parses the work unit and resolves the data identifier by contacting the data look up server to get a list of data HTTP endpoint from data centres.
3. The BOINC client then uses VACSODE to determine the best endpoint to use at this point in time and contacts the data centres to start downloading the data.

A snapshot of a `workunit` using AtticFS is shown below

```
<file_info>
  <name>f050a833-2fac-44c9-9b08-7c</name>
  <url>
  attic://host/cplan/download/f050a833...9b08-7c
  </url>
  <md5_cksum>92d8c8...b0250e5</md5_cksum>
  <nbytes>10485760</nbytes>
</file_info>
<workunit>
  <file_ref>
    <file_name>f050a833...9b08-7c</file_name>
    <open_name>in</open_name>
</file_ref>
  .......
  .......
</workunit>
```

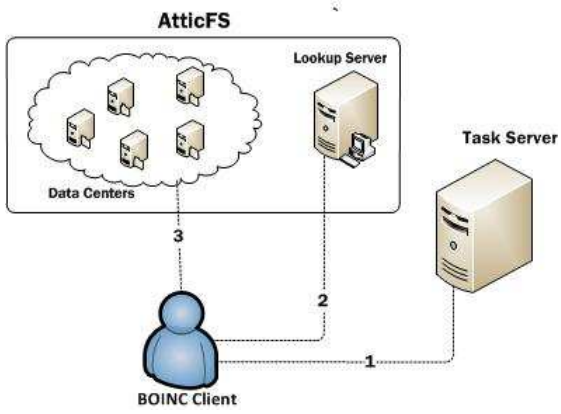

Fig. 6.1: Integrating AtticFS in BOINC Project

6.1. Baseline Comparison with BOINC. In this section, we make a comparative evaluation of BOINC with AtticFS to determine a baseline for the further experiments described later in the paper. We undertake two experiments to carry out this comparison and the result of this comparison are provide in Figure 6.2.

In the first experiment, we employ the use of a BOINC data server. A different number of data clients (1, 3 and 9) are used to download a 10 MB file. The network speed is set to 10 Mbps for the server and 100Mbps for the clients. We analysed the performance of the BOINC server as a baseline, the result of this part of the experiment shows that the average download time needed by all clients to download the 10 MB file increases as the number of clients increase.

In the second experiment, AtticFS is used to download a 10 MB file, published using a 1 MB chunk size. The number of data centres is set to 1, 3 and 9. The network speed in this experiment was 10 Mbps for the servers and 100 Mbps for the clients. We ran the experiment three times using a different number of clients (1, 3 and 9). The results of this part of the experiment show the download time for each client using a different number of data centres (1, 3 and 9).

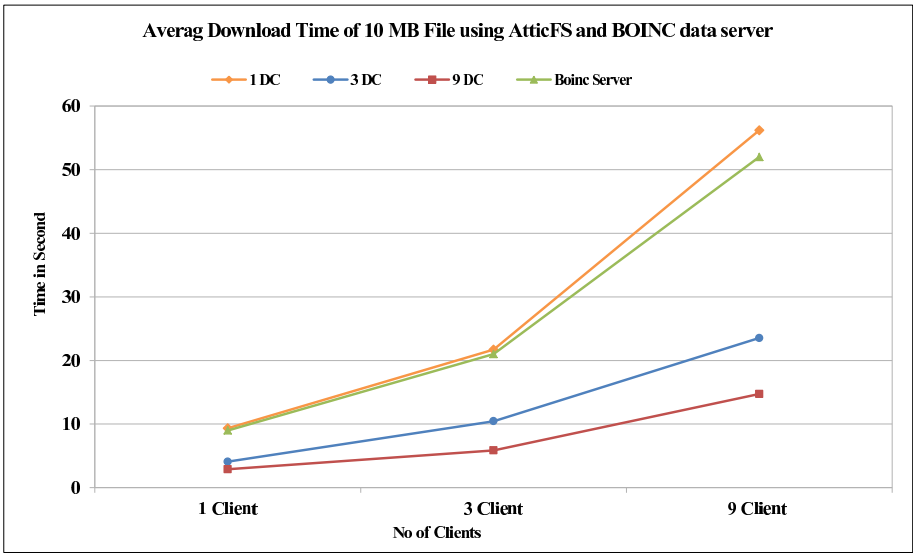

Fig. 6.2: Average Download Time 10 MB File Using AtticFS and BOINC Data Server

By using one data centre, the clients download time increases as more clients are added to the network, because of the limited network bandwidth of the server. The download time increases as more clients attempt

to download from the same resource creating a queue. Increasing the number of data centres by a factor of three enables clients to concurrently download chunks. Since the three data centres are shared between the clients this means the waiting time will decrease. For the nine data centres case, we see the largest improvements and the most consistency between the clients. By comparing the results of this experiment in Figure 6.2, we notice that AtticFS will give almost the same results as a BOINC data server when one data centre is used. In fact, BOINC is marginally more efficient, as it does not have the message overhead that AtticFS has, in terms of querying and prioritising endpoints before downloading commences. However, as AtticFS adds data centres, the download time decreases.

7. The VACSCODE Trust Framework. In comparison with the general BOINC system (which uses a pre-defined data server), a client in our system is required to identify a data centre prior to commencing data download. The selection of data centres is based on their reputation in the system, with reference to one or more metrics. The metrics are:

- the upload speed that a client obtained through a connection with a data server
- the availability of a particular data centre
- and the integrity of data supplied by the data centre

These metric are named in the framework as **DCspeed**, **DCAvailability** and **DCHonesty**, respectively. Each metric is independently calculated using feedback from multiple clients using our specialist tools. We have chosen these particular metrics because of their dominance in P2P literature, and in supporting job execution in volunteer computing systems.

Speed primarily relates to performance issues such as access time, latency and effective bandwidth. Availability relates to uptime and resilience, covering aspects such as downtime, and failure rate. Honesty covers aspects such as data integrity and quality, storage reliability and any malicious modification to the data. The trust framework makes use of the AtticFS (discussed in section 3.1) to support concurrent data downloads from multiple data centres. It utilizes the communication between the clients and the data lookup server to send feedback and receive data on the associated trust metrics.

The trust framework (Figure 7.1) involves components that operate on both the client and the server; the client generates feedback, processes trust values and selects data centres based on its preferences; the server collects client feedback, updates the reputation database and provides reputation data to the clients.

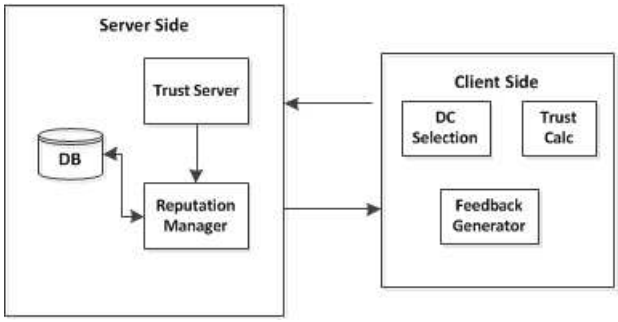

Fig. 7.1: Trust Framework extending AtticFS

In our framework trust is calculated by using feedback from the multiple clients that interact with a data centre using a Beta distribution [15], as outlined in equations 7.1 and 7.2, specifying whether the client was satisfied (r) or not satisfied (s) with the download from a data centre. The Beta distribution has been used to take into account this combined assessment (i.e. by considering both satisfied and not satisfied) from multiple clients, rather than only consider the positive outcomes (i.e. the number of times that a client has been satisfied with the download). After a client completes downloading data, it provides a subjective assessment of each of the three metrics (availability, honest and speed) for each data centre that has been used by this client. This public feedback can then subsequently be used by other clients, to support their decision about which data centres to trust, using equations 7.1 and 7.2.

$$x = \frac{r+1}{r+s+2} \tag{7.1}$$

$$T = a.TAvailability + b.THonesty + c.TSpeed,\ where\ a+b+c=1 \tag{7.2}$$

When a client calculates the total trust value of each data centre it uses algorithm 7.0.1. As outlined in the algorithm, a threshold referred to as `trustThreshold` is used to limit the number of data centres that have been returned as an outcome of a search. The client can either modify this parameter themselves or set the minimum number of data centres (referred to as `minDC` in algorithm 7.0.1) they would prefer to obtain to download from (i.e., the total number of data centres that match their particular trust criteria). In algorithm 7.0.1, the threshold value is set by a client to be 0.9. If an automated approach is used, i.e. where a client does not specify the threshold but instead identifies the minimum number of data centres they would prefer to download from, this threshold value is automatically adjusted.

Algorithm 7.0.1 Data Centre Selection

```
1: selectedDataCentres = 0;
2: trustThreshold = 1.0;
3: decrement = 0.1;
4: minDC=3;
5: loop
6: for each DataCentre[i] do
7:    if TotalTrustValue[i] ≥ trustThreshold then
8:       selectedDataCentres = selectedDataCentres + 1;
9:       return [i] ;
10:   end if
11: end for
12: if selectedDataCentres ≤ minDC then
13:   trustThreshold = trustThreshold - decrement;
14:   goto loop
15:   else exit();
16: end if
```

8. Experiments.

8.1. Availability Experiments. In a volunteer computing environment peers can enter and exist the network at any time. In AtticFS, as peers can also play the role of a data centre, the availability of these data centres can change over time. With the distributed servers appearing and disappearing, a mechanism is required to limit this variability in the network so that a clients download efficiency is maximised. We conducted a series of experiments to show how the download bandwidth is improved when the trust framework is utilised.

In the first experiment, Attic FS consists of a lookup server and 10 data centres. The 10 data centres have a 10Mb/s connection and are all deemed honest peers (i.e. no malicious behaviour) for this experiment. 20 modified AtticFS clients (running on 20 Linux machines) were used to mimic data requests, which were setup to request a 10MB file at periodic intervals. The clients report their download experience to the lookup server when they finish downloading data. This feedback to the server contains the status of each data centre and information about whether they were available, whether the download speed was satisfactory and whether they were honest. A Poisson distribution was used to simulate the availability of data centres – as this represents a realistic scenario from many existing P2P systems. Figure 8.1 shows the distribution of the data centres and when they are online and offline over a four-hour period. The total duration of the experiment was eight hours, only the first four hours have been shown here to demonstrate the overall availability trend.

We compare the behaviour of the AtticFS client with and without the trust framework. One instance of each type of client are used, each requesting data periodically every five minutes during the experiment. The modified

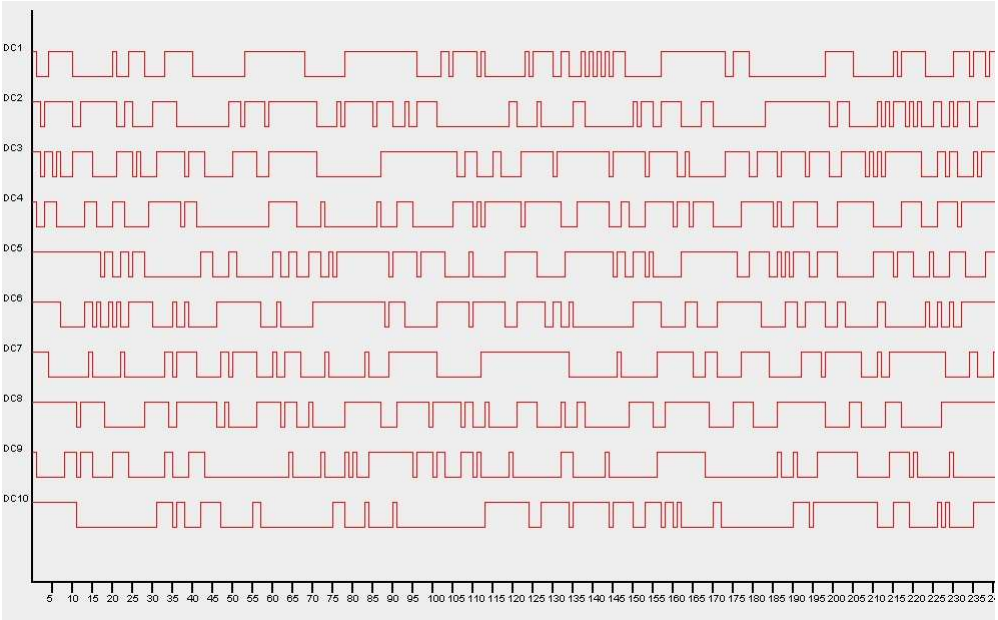

Fig. 8.1: Data centres Availability

AtticFS client is configured with the following parameters: Availability weight factor (AWF)=1, Honesty weight factor (HWF) = 0, and Speed weight factor (SWF)=0, as we are only interested in the availability of the 10 data centres, the other weight factors are set to zero.

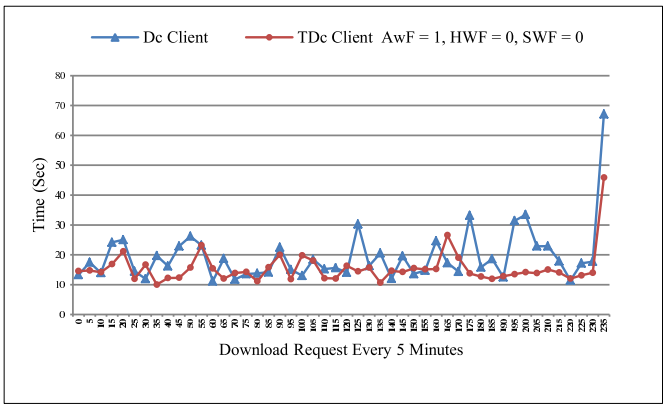

Fig. 8.2: Data Download using AWF in the first four hour period

This experiment had a duration of eight hours, the data centre availability in the first four hours is the same as in the second four hours as shown in (Figure 8.2), and we found that during the first four hours of the experiment, the download time of both clients is similar see (Figure 8.3). However, in the next four hours, our trusted data centre has an improved download time as the trust algorithm converges and learns the state of the network i.e. our trusted data centres learn to avoid the unavailable data centres as the experiment progresses. In Figure 8.3, it can be observed that the behaviour of nodes employing the use of the trust algorithm becomes smoother and more predictable during the last four hours of the experiment. We believe this convergence shows promise as the algorithm can adapt to network conditions and variability in node availability, which is a requirement of the volunteer computing environment as a whole.

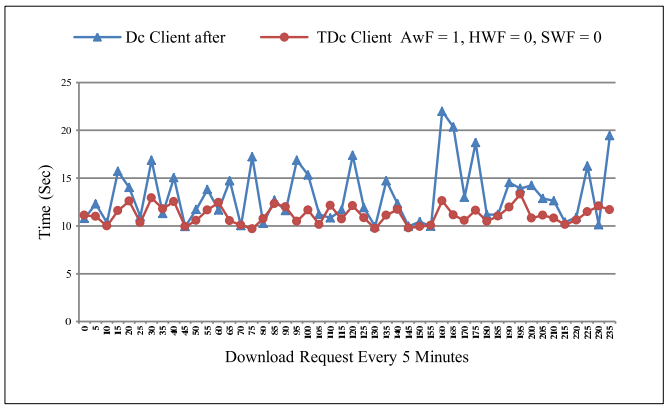

Fig. 8.3: Data Download using AWF in the second four hour period

8.2. Honesty Experiments. In this experiment, malicious data centres are injected into the network in order to disrupt the system. These nodes intentionally provide corrupted data to their clients in order to attack the system. We designed an experiment to show how our trust framework can become fault tolerant to this malicious behaviour and avoid the use of malicious data centres to recover and repair the corrupted network. This experiment focuses on the honesty of data centres and how this affects the download time. It aims to show how the client which uses our trust framework offers better stability and hence increased download efficiency than the ordinary AtticFS client. Note that since an MD5 hash is taken of the data, malicious peers can only slow down the network because if the hash of the downloaded file does not match the original hash of the data, it will be discarded and downloaded again. Our framework detects these malicious peers and effectively removes them from a clients download list thereby making significant gains overall.

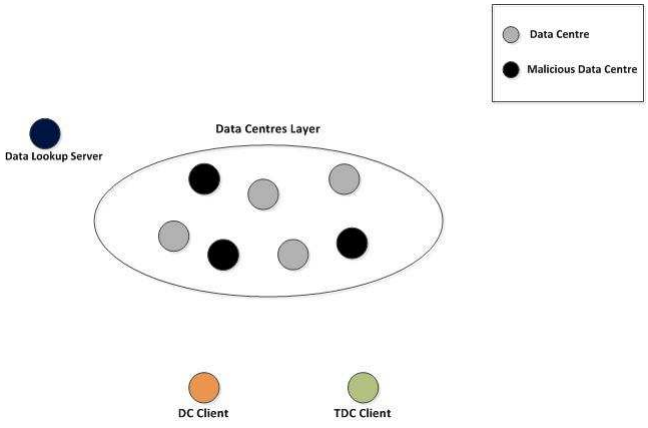

Fig. 8.4: Attic File System using Different Data centres Behaviour

Seven data centres are used in this experiment (Figure 8.4). Three of them are honest data centres and the other three are malicious and send corrupted data to their clients. Because we are interested in the honesty of these data centres, they are available throughout the duration of the experiment, this experiment is repeated three times every time the data centres uses the throttle functionality to configure their speed connection to (128 KB, 256 KB and 512 KB). The modified AtticFS client is configured with the following parameters (AWS = 0, HWF = 1, SWF = 0). As identified in Experiment 1 for availability, only the honesty weight factor is set to 1 in this experiment.

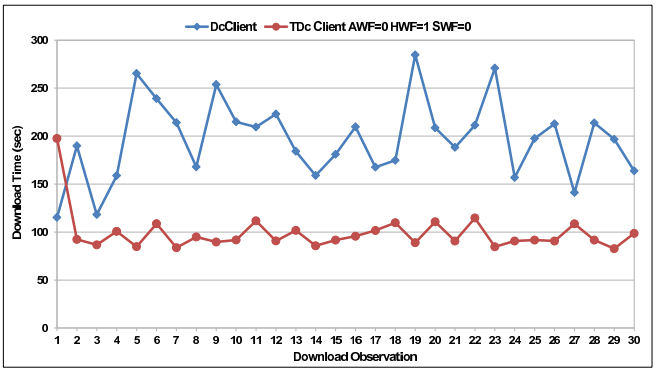

Fig. 8.5: 128 KB upload speed

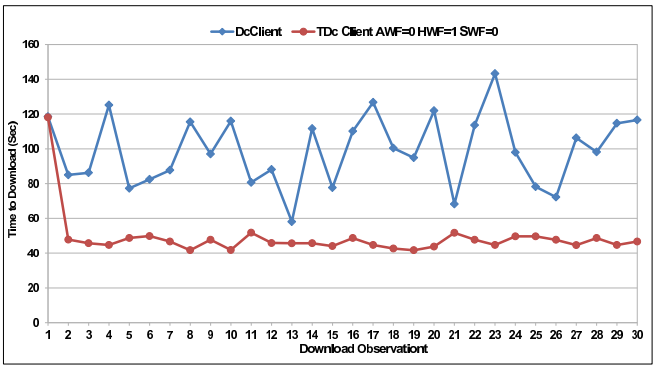

Fig. 8.6: 256 KB upload speed

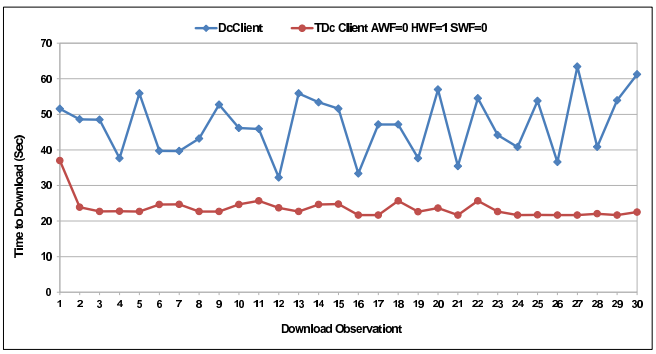

Fig. 8.7: 512 KB upload speed

Figures (8.5, 8.6, 8.7) show the result of this experiment. It can be observed that our trusted client has significantly better download time because it avoids the malicious data centres, while the ordinary AtticFS client uses all the data centres and therefore downloads a number of unnecessary corrupted chunks, which need to be reloaded – thereby incurring a download overhead. After a short period of convergence, our system performs on average 3 times better than the standard AtticFS approach. We believe that this shows a huge potential as it addresses one of the fundamental issues in volunteers distributing data, which is the ability to trust third party peers. This experiment indicates that our system can learn to avoid malicious peers and dynamically select more trusted peers in the network, which opens up the possibility for such a dynamic data distribution approach.

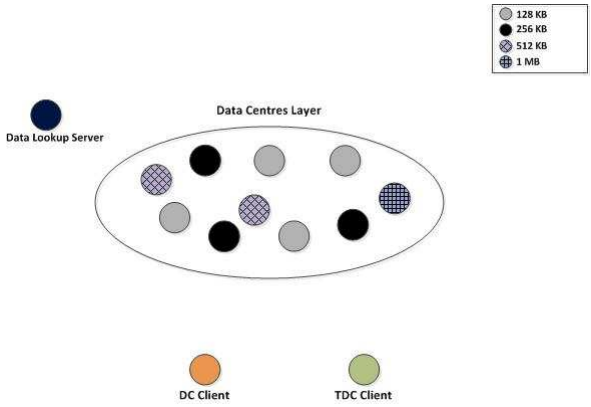

Fig. 8.8: Different upload speed

8.3. Speed Experiments. The data centres used in AtticFS can choose different upload speeds using the throttle functionality and this obviously affects the download time of each peer. Since clients are interested in getting data in the fastest possible way, they should obviously choose those data centres with high-speed connections. However, if all peers download from the same fastest peers then there will be a bottleneck, so any algorithm must dynamically optimise the distribution of clients connected to a data centre at each time-step during the operation of the system. Our trust framework provides such a mechanism by choosing data centres with high bandwidth at that point in time in order to optimise the throughput of the distributed system as a whole. In this experiment therefore we are interested in how the trust framework is used to choose the data centres which have the highest bandwidth connections at any point in time. For this experiment, 10 data centres are used (Figure 8.8). The upload speed of these data centres are configured using throttli ng functionality to configure the upload speed of each data centre as mentioned in (Table 8.1). The data centres have continuous availability and they all act honestly. The modified AtticFS client is configured with the following parameters (AWS = 0, HWF = 0, SWF = 1) to configure the system to only focus on the speed of the data centres.

No of Data centres	Bandwidth
1	1 MB
2	512 KB
3	256 KB
4	128 KB

Table 8.1: Different upload Speed

The results in Figure 8.9 show significant improvements by clients making use of the trust framework, compared to the conventional AtticFS approach, achieving download speeds several times faster overall. Even at this small-scale proof-of-concept testbed, these results demonstrate the benefit of the approach.

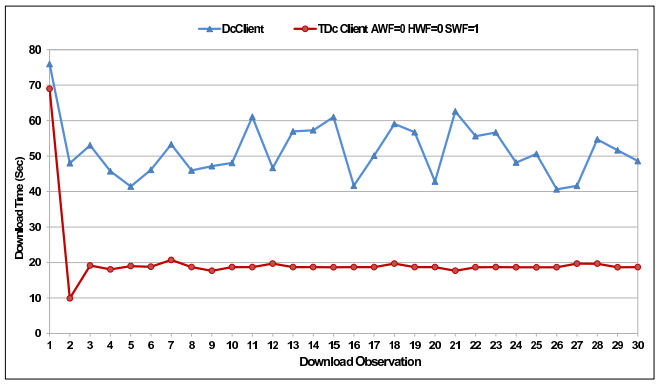

Fig. 8.9: Different upload speed

9. Conclusion and Future Work. The experiments described in this paper show how to extend and improve data distribution in volunteer computing projects using volunteers resources . Furthermore, they show significant improvements with respect to a clients download time when the trust model is used compared to the conventional AtticFS scheme. By comparing against AtticFS which makes use of multiple concurrent downloads using file chunks, rather than a conventional downloading mechanism based on a complete file, we are able to evaluate our approach against an already optimized system. Although the testbed we used is small, it employs the use of machines on a real network rather than taking a simulated approach where it is typically not possible to record a direct observation of the system. We believe these experiments show extremely encouraging results that can be investigated further. We are planning to perform experiments using this system using higher numbers of network sizes and also to investigate different combinations of the trust algorithms parameters in order to achieve an optimal balance of network behavior and performance.

REFERENCES

[1] D. P. Anderson, *BOINC: A System for Public-Resource Computing and Storage*. In: 5th IEEE/ACM International Workshop on Grid Computing, pp. 365–372. Pittsburgh, USA (2004)
[2] F. Cappello et al., *Computing on Large-Scale Distributed Systems: XtremWeb Architecture, Programming Models, Security, Tests and Convergence with Grid*. Future Generation Computer Systems 21(3), 417–437 (2005)
[3] S. Ghemawat, H. Gobioff, S. T. Leung ,*The Google File System*. SIGOPS Oper. Syst. Rev. 37(5), 29–43 (2003). DOI http://doi.acm.org/10.1145/1165389.945450
[4] L. Cherkasova, J. Lee, *Fastreplica: Efficient large file distribution within content delivery networks*. In: Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems (USITS). Seattle, WA, USA (2003)
[5] B. Cohen, *Incentives build robustness in BitTorrent*. In P2PEcon (2003).
[6] M. Witkowski, A. Aritikis, and J. Pitt, *Experiments in building experiential trust in a society of objective-trust based agents*. Trust in Cyper-societies, pages 111-132, 2001
[7] J. Sabater and C.Sierra,*Regret: a reputation model for gregarious societies*. Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agents Systems, 2002.
[8] A. Abdul-Rahman and S. Hailes. *Using Recommendations for managing trust in distributed systems*. Proceedings IEEE Malaysia International Conference on Communication, 1997.
[9] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. *The Eigentrust algorithm for reputation management in P2P networks*. Proceedings of the Twelfth International World Wide Web Conference, 2003.
[10] L. Page, S. Brin, R. Motwani, and T. Winograd, *The Pagerank citation ranking: Bringing order to the Web*. Stanford Digital Library Technologies Project, 1998.
[11] R. Zhou and K. Hwang, *PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing*, IEEE Transactions on Parallel and Distributed Systems, Vol. 18, No. 4, pp 460–473, 2007.
[12] R. F. C. Castelfranchi, *Principles of Trust for Multi-Agent Systems: Cognitive anatomy*, social importance and quantification. Proceedings of the International Conference on Multi-Agent Systems, 1998.
[13] Ian Kelley and Ian Taylor, *Bridging the Data Management Gap Between Service and Desktop Grids*. In: Distributed and Parallel Systems In Focus: Desktop Grid Computing, Peter Kacsuk, Robert Lovas and Zsolt Nemeth (Editors), Springer, 2008.
[14] BOINC statistics, `http://www.boincstats.com`. Last accessed: April 2011.
[15] A. Jsang, R. Ismail , *The beta reputation system*, in: Proceedings of the 15th Bled Electronic Commerce Conference, Bled, Slovenia, 2002.
[16] The SETI@Home project, `http://setiathome.berkeley.edu`. Last accessed: March 2011.

[17] Entropia project, http://www.entropia.com. Last accessed: March 2011.
[18] Einstein@Home project, http://einstein.phys.uwm.edu. Last accessed: March 2011.
[19] The Climateprediction.net project, http://climateprediction.net. Last accessed: April 2011.

Edited by: Dana Petcu and Marcin Paprzycki
Received: May 1, 2011
Accepted: May 31, 2011