Scalable Computing: Practice and Experience
Volume 11, Number 1, pp. 53–62. http://www.scpe.org
ISSN 1895-1767

EXPLORING CARRIER AGENTS IN SWARM-ARRAY COMPUTING

BLESSON VARGHESE* AND GERARD MCKEE†

Abstract. How can a bridge be built between autonomic computing approaches and parallel computing systems? The work reported in this paper is motivated towards bridging this gap by proposing a swarm-array computing approach based on 'Intelligent Agents' to achieve autonomy for distributed parallel computing systems. In the proposed approach, a task to be executed on parallel computing cores is carried onto a computing core by carrier agents that can seamlessly transfer between processing cores in the event of a predicted failure. The cognitive capabilities of the carrier agents on a parallel processing core serves in achieving the self-ware objectives of autonomic computing, hence applying autonomic computing concepts for the benefit of parallel computing systems. The feasibility of the proposed approach is validated by simulation studies using a multi-agent simulator on an FPGA (Field-Programmable Gate Array) and experimental studies using MPI (Message Passing Interface) on a computer cluster. Preliminary results confirm that applying autonomic computing principles to parallel computing systems is beneficial.

Key words: swarm-array computing, intelligent agents, carrier agents, autonomic computing

1. Introduction. Inspirations from nature have led computing scientists to focus on biologically inspired computing paradigms. Amorphous computing [1], evolutionary computing [2] and organic computing [3] are such areas that focus on abstracting designs from nature. Lately, autonomic computing inspired by the autonomic human nervous system [4] is the emphasis of distributed computing researchers which is considered in this paper.

With the aim of building large scale systems [5], reducing cost of ownership [6] [7] and reallocating management responsibilities from administrators to the computing system itself [8][9][10], autonomic computing principles have paved necessary foundations towards self-managing systems. Self-managing systems are characterized by four objectives, namely self-configuration, self-healing, self-optimizing and self-protecting and four attributes, namely self-awareness, self-situated, self-monitoring and self-adjusting [4][11][12].

Autonomic computing researchers have adopted six different approaches, namely emergence-based, component/service-based, control theoretic based, artificial intelligence, swarm intelligence and agent based approaches to achieve self-managing systems.

The emergence based approach for distributed systems considers complex behaviours of simple entities with simple behaviours without global knowledge [13]. Autonomic computing research on emergence based approaches is reported in [13] and [14].

The component/service based approach for distributed systems employ service-oriented architectures, and is implemented in many web based services. These approaches are being developed for large scale networked systems including grids. Autonomic computing research on component/service based approaches is reported in [15], [16] and [17].

The control theoretic based approach aims to apply control theory for developing autonomic computing systems. The building blocks of control theory are used to model computing systems and further used to study system properties. Using a defined set of control theory methodologies, the objectives of a control system can be achieved. Research on control theoretic based approaches applied to autonomic computing is reported in [18] and [19].

The artificial intelligence based approaches aim for automated decision making and the design of rational agents. The concept of autonomy is realized by maximizing an agent's objective based on perception and action in the agent's environment with the aid of information from sensors and in-built knowledge. Work on artificial intelligence approaches for autonomic computing is reported in [20] and [21].

The swarm intelligence based approaches focus on designing algorithms based on collective behaviour of swarm units that arise from local interactions with their environment. The algorithms considered are population-based stochastic methods executed on distributed processors. Autonomic computing research on swarm intelligence approaches is reported in [22] and [23].

The agent based approaches for distributed systems is a generic technique adopted to implement emergence, component/service, artificial intelligence or swarm intelligence based approaches. The agents act as

*Active Robotics Laboratory, School of Systems Engineering, University of Reading, Whiteknights Campus, Reading, Berkshire, United Kingdom, RG6 6AY (b.varghese@student.reading.ac.uk).

†School of Systems Engineering, University of Reading, Whiteknights Campus, Reading, Berkshire, United Kingdom, RG6 6AY (g.t.mckee@reading.ac.uk).

autonomic elements or entities that perform distributed task. The domain of software engineering considers agents to facilitate autonomy and hence have a profound impact on achieving the objectives of autonomic computing. Research work based on multi-agents supporting autonomic computing are reported in [9], [24] and [25].

However, though all of the autonomic computing approaches above aim towards the objectives of autonomic computing, few researchers have applied autonomic computing concepts to parallel computing systems. This is surprising since most distributed computing systems are closely associated with the parallel computing paradigm. The benefits of autonomy in computing systems, namely reducing cost of ownership and reallocating management responsibilities to the system itself are also relevant to parallel computing systems.

How can a bridge be built between autonomic computing approaches and parallel computing system? The work reported in this paper is motivated towards bridging this gap by proposing a swarm-array computing approach, that aims to achieve autonomy for distributed parallel computing systems.

Swarm-array computing is biologically inspired by the theory of autonomous agents in natural swarms that are abstracted and implemented in parallel computing systems. This technique considers the computational resource as a swarm of resources and the task to be executed as a swarm of sub-tasks. Hence, the approach considers complex interactions between swarms of sub-tasks and swarms of resources. These interactions bring about the notion of intelligent agents or swarm agents carrying the sub-tasks and intelligent cores or swarm of cores executing the sub-task.

In this paper, a swarm-array computing approach is proposed as a solution that aims to apply autonomic concepts to parallel computing systems and in effect achieve the objectives and attributes of self-managing systems. Unlike another swarm-array computing approach reported in [26], the approach proposed in this paper considers the task to be executed on parallel computing cores as a swarm of autonomous agents.

The remainder of the paper is organized as follows. Section 2 considers the proposed swarm-array computing approach. The second approach is of focus in this paper. Section 3 investigates the feasibility of the proposed approach by considering simulations. Section 4 presents the implementation of the proposed approach on a computer cluster. Section 5 concludes the paper by considering future work.

2. Swarm-Array Computing Approach. Swarm-array computing is a swarm robotics inspired approach that is proposed as a path to achieve autonomy in parallel computing systems. The foundations of swarm-array computing are the existing computing paradigms of parallel and autonomic computing. There are three approaches based on intelligent cores, intelligent agents and a hybrid approach based on both intelligent cores and intelligent agents that bind the swarm-array computing constituents together [26].

In this paper, the focus is on the second approach based on intelligent agents. The aim of the 'Intelligent Agent based' approach is to demonstrate that the cognitive capabilities of an agent complementing its intelligence can be used to achieve the objectives and attributes of autonomic computing.

In the intelligent agent based approach, a task to be executed on a parallel computing system is decomposed into sub-tasks and mapped onto agents that carry these tasks onto nodes or cores for execution. The agent and the sub-problem are independent of each other; in other words, the agents only carry the sub-tasks or act as a wrapper around the sub-task independent of the operations performed by the task.

In the proposed approach, an agent has capabilities similar to the capabilities of a natural agent presented above. Intelligence of an agent in the computing environment is demonstrated in four different ways. Firstly, an agent is aware of its environment, that is the nodes or cores on which it can carry a task onto, other agents in its vicinity and agents with which it interacts or shares information. Secondly, an agent can situate itself on a node or core that may not fail soon and can provide necessary and sufficient consistency in executing the task. Thirdly, an agent can predict core failures by consistent monitoring (for example, power consumption and heat dissipation of the cores can be used to predict failures). Fourthly, an agent is capable of shifting gracefully from one core to another, without causing interruption to the state of execution, and notifying other interacting agents in the system when a core on which a sub-task being executed is predicted to fail.

Hence, objectives such as self-configuration, self-healing and self-optimizing and attributes such as self-awareness, self-situated, self-monitoring and self-adjusting are inherently obtained in the proposed approach by the cognitive capabilities demonstrated by the agent.

3. Simulation Studies. Simulation studies were pursued to validate and visualize the proposed intelligent agent based approach. Various simulation platforms were considered, namely network simulators, which could predict behaviours of data packets in networks, and multi-agent simulators, that could model agents and their

behaviours in an environment. Since FPGA cores are considered in this paper, network simulators were not an appropriate choice. The approach proposed in this paper considers executing cores as agents; hence a multi-agent simulator is employed. This section is organized into describing the experimental environment, modelling the experiment and experimental results.

With the objective of exploring swarm-array computing, FPGAs are selected as an experimental platform for simulating the proposed approaches. FPGAs are a technology under investigation in which the cores of the computing system are not geographically distributed. The cores in close proximity can be configured to achieve a regular grid or a two dimensional lattice structure. Another reason of choice to look into FPGAs is its flexibility for implementing reconfigurable computing.

The feasibility of the proposed swarm-array computing approach was validated on the SeSAm (Shell for Simulated Agent Systems) simulator. The SeSAm simulator environment supports the modelling of complex agent-based models and their visualization [27][28].

The environment has provisions for modelling agents, the world and simulation runs. Agents are characterized implemented in the form of an activity diagram by a reasoning engine and a set of state variables. The state variables of the agent specify the state of an agent. The world provides knowledge about the surroundings in which the agent is situated. A world is also characterized by variables and behaviours and defines the external influences that can affect the global behaviour of the agent. Simulation runs are defined by simulation elements, namely situations, analysis lists, simulations and experiments that contribute to the agent-based model being constructed.

As considered in Section 2, the swarm-array computing approach considers the computing platform and the problem/task. An FPGA is modelled in the SeSAm environment such that the hardware cores are arranged in a 5 X 5 regular grid structure. The model assumes serial bus connectivity between individual cores. Hence, a task scheduled on a core can be transferred onto any other core in the regular grid.

The breakdown of any given task into subtasks is not considered within the problem domain of swarm-array computing. The simulation is initialized with sub-tasks scheduled to a few cores in the FPGA grid. Each sub-task carrying agent consistently monitors the hardware cores. This is possible by sensory information (in our model, temperature is sensed consistently) passed onto the carrier agent. In the event of a predicted failure, the carrier agent displaces itself to another core in the computing system. The behaviour of the individual cores vary randomly in the simulation. For example, the temperature of the FPGA core changes during simulation. If the temperature of a core exceeds a predefined threshold, the subtask being executed on the core is transferred by the carrier agent onto another available core that is not predicted to fail. During the event of a transfer or reassignment, a record of the status of execution of the subtask maintained by the carrier agent also gets transferred to the new core. If more than one sub-task is executed on a core predicted to fail, each sub-task may be transferred to different cores.

Figure 3.1 is a series of screenshots of a random simulation run developed on SeSAm for nine consecutive time steps from initialization. The figure shows the executing cores as rectangular blocks in pale yellow colour. When a core is predicted to fail, i. e., temperature increases beyond a threshold, the core is displayed in red. The subtasks wrapped by the carrier agents are shown as blue filled circles that occupy a random position on a core. As discussed above, when a core is predicted to fail, the subtask executing on the core predicted to fail gets seamlessly transferred to a core capable of processing at that instant.

The simulation studies are in accordance with the expectation and hence are a preliminary confirmation of the feasibility of the proposed approach in swarm-array computing. Though some assumptions and minor approximations are made, the approach is an opening for applying autonomic concepts to parallel computing platforms.

4. Implementation. In this section, a cluster-based implementation of the intelligent agent approach is considered. The cluster used for the research reported in this paper is one among the high performance computing resources available at the Centre for Advanced Computing and Emerging Technologies (ACET), University of Reading, United Kingdom [29] [30]. The cluster is primarily used for the purpose of teaching and performing multi-disciplinary research. The cluster consists of a head node and 33 compute nodes. All nodes are connected via a Gigabit ethernet switch and communicate via the standard TCP protocol.

The cluster-based implementations reported in this paper are based on the Message Passing Interface (MPI) [31], a standardized application programming interface (API) used for parallel and/or distributed computing. Open MPI [32] [33] version 1.3.3, an open source implementation of MPI 2.0 is employed on the cluster. An

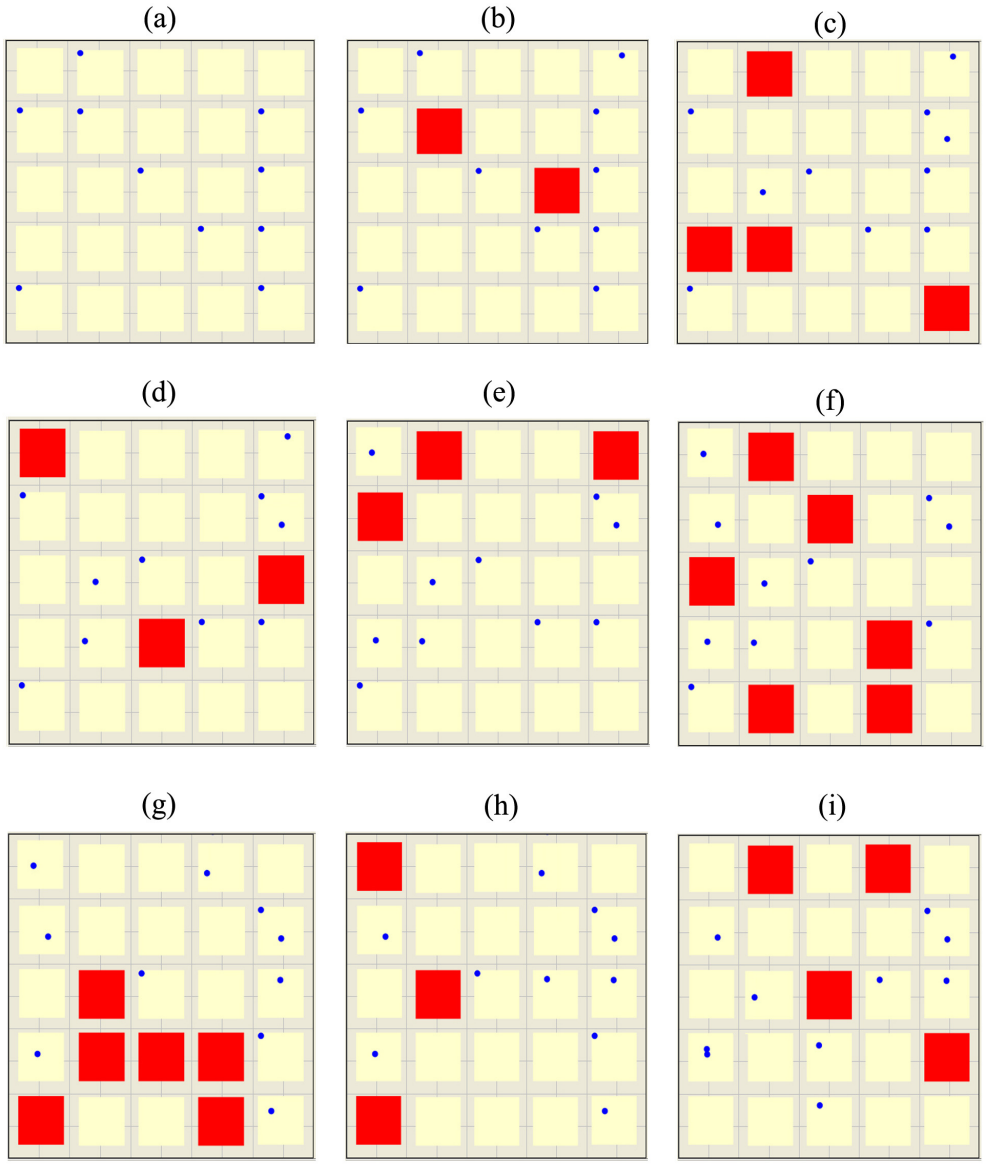

FIG. 3.1. *Sequence of nine simulation screenshots (a) - (i) of a simulation run from initialization on the SeSAm multi-agent simulator. Figure shows how the carrier agents carrying sub-tasks are seamlessly transferred to a new core when executing cores fail.*

important feature of MPI 2.0, dynamic process creation and management, is of potential for exploration in the context of swarm-array computing.

The MPI dynamic process model permits the creation and management of a set of processes both when an MPI application begins and after the application has started. The management of newly created processes include cooperative termination of a process, communication between newly created processes and existing MPI application, and establishing communication between two independent processes. MPI routines such as MPI_COMM_SPAWN is used to create a new MPI process and establish communication from an existing MPI application. On the other hand, MPI routines such as MPI MPI_COMM_ACCEPT and MPI_COMM_CONNECT can be used to establish communication between two independent processes. More MPI specific details on dynamic process model can be obtained from [31] [34].

To analyse the proposed intelligent agent based approach a parallel reduction algorithm is considered. Parallel reduction algorithms implement the bottom-up approach of binary trees [35], and are of interest in the

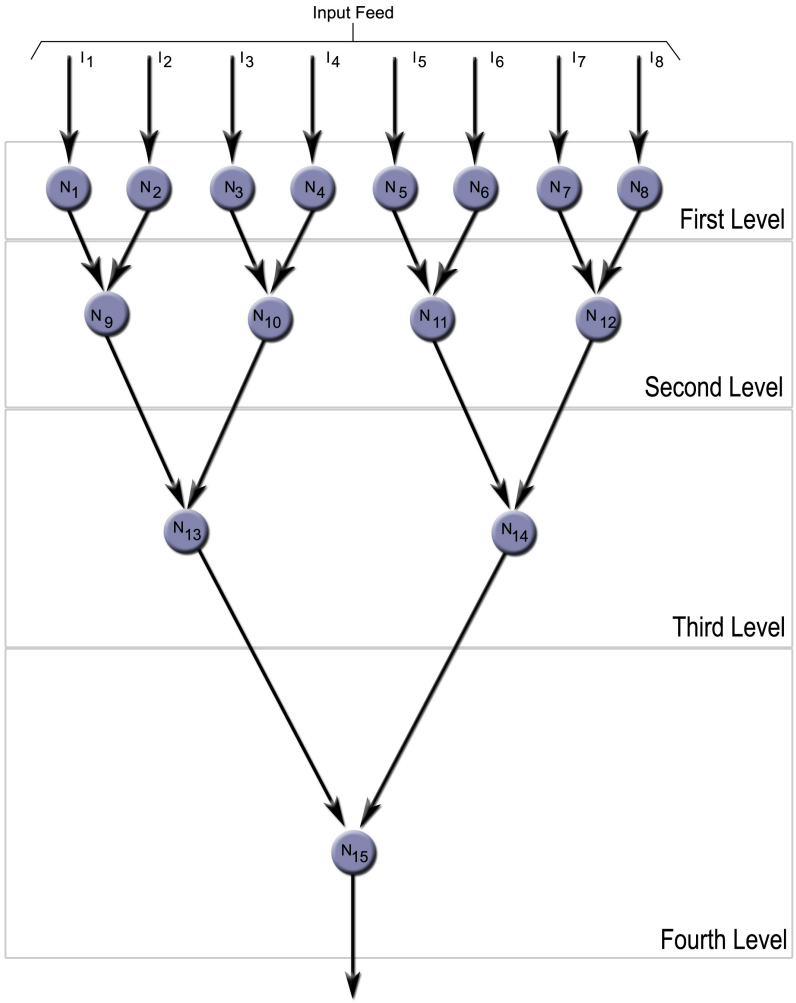

FIG. 4.1. *Illustration of the parallel summation algorithm.*

context of applying autonomic computing concepts to parallel computing systems due to two reasons. Firstly, the computing nodes of a parallel reduction algorithm tend to be critical. The execution of the algorithm stalls or produces an incorrect solution if any node information is lost. Secondly, parallel reduction algorithms are employed in critical applications such as space applications. Such applications require autonomic distributed systems.

Parallel summation is an exemplar of parallel reduction algorithm considered in this paper. If this class of algorithms do not incorporate fault tolerance concepts, then if a computing node fails due to an unpredictable event, the execution of the algorithm would stall.

The general concept of the algorithm is illustrated in figure 4.1. The algorithm works in four sequential levels. The first level comprising nodes $N_1 - N_8$ receives a live input feed of data $I_1 - I_8$. The second level comprising nodes $N_9 - N_{12}$ receives data from the first level, adds the data received and yields the result to the third level nodes N_{13} and N_{14}. The fourth level, adds data received from the third level nodes and produces the final result.

For a given time step, every node in a level operates in parallel. Each node is characterized by input dependencies (process or processor a node is dependent on for receiving an input), output dependencies (process or processor a node yields data to as output) and data contained in the node. The first level nodes have one input dependency and one output dependency. For instance, node N_1 has one input dependency I_1 and node N_9 as its output dependency. However, the second, third and fourth levels have two input dependencies and one

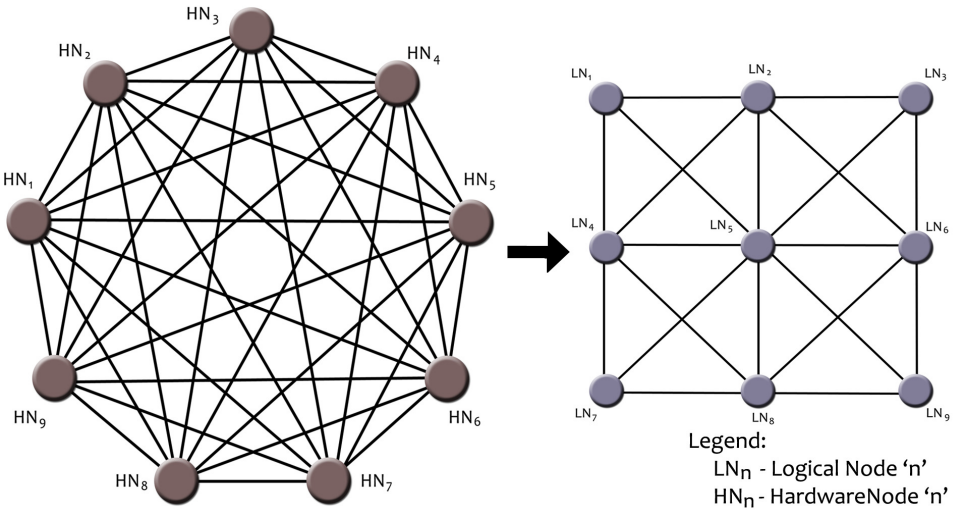

FIG. 4.2. *Mapping hardware nodes of the cluster to logical nodes in the abstracted layer.*

output dependency. For instance, node N_{13} of the third level has nodes N_9 and N_{10} as input dependencies and node N_{15} as output dependency. The data contained in a node is either the input data for the first level nodes or a calculated value (sum of two value in the case of a parallel summation algorithm) stored within a node.

A layer that abstracted the hardware resource layer, otherwise referred to as the abstracted layer had to be implemented. The hardware resource layer comprises physical nodes of the cluster and is connected via a switch, thereby forming a fully connected mesh topology. However, the abstracted layer is obtained when the physical nodes are abstracted as logical nodes. This is possible by implementing rules/policies. The policies are such that a process can only communicate with a vertically, horizontally or diagonally adjacent process, effectively leading to a grid topology on the abstracted layer. For example, nine nodes forming a fully connected mesh topology in figure 4.2 is abstracted to a grid topology in the abstraction layer.

The intelligent agents implemented in the parallel summation algorithm are with respect to the cognitive capabilities of agents considered in Section 2. The agents on the abstracted layer are created such that they carry input and output dependencies and data. Since, parallel summation is relatively less complex when compared to other computational algorithms, the agents carry little information and have only few dependencies.

Each process executing on a node also gathers some sensory information to predict whether a node is likely to fail. The sensory information enables an agent to know its own surroundings on the computational environment, hence demonstrating the first cognitive capability considered in Section 2.

In the implementation presented in this paper node temperatures are simulated. When the temperature of a node rises beyond a threshold, the process executing on that node predicts a failure and hence spawns a process on an adjacent core in the abstracted layer. In this case, an agent gathers sensory information on rising temperature than can likely impair or deteriorate its functioning, thereby demonstrating the third cognitive capability considered in Section 2. When rising temperature is detected, an agent has the potential to identify a node in the computational environment on which a new process can be spawned, thereby demonstrating the second cognitive capability considered in Section 2.

The agent on the abstracted core expected to fail shifts to the adjacent core on which the new process was spawned. An agent is capable of passing from one node to another, thereby demonstrating the fourth cognitive capability considered in Section 2.

The dependency information carried by the agent that was shifted to the new core is employed to reinstate the state of execution of the algorithm. The data for summation contained in the agent, either obtained from a previous level or a calculated value to be yielded to the next level, ensures that information is not lost and does not affect the final solution in critical applications.

Since the agents possess cognitive capabilities autonomic computing objectives such as self-configuration, self-healing and self-optimizing and autonomic computing attributes such as self-awareness, self-situated, self-monitoring and self-adjusting are inherently achieved. Hence, the approach implemented above incorporates

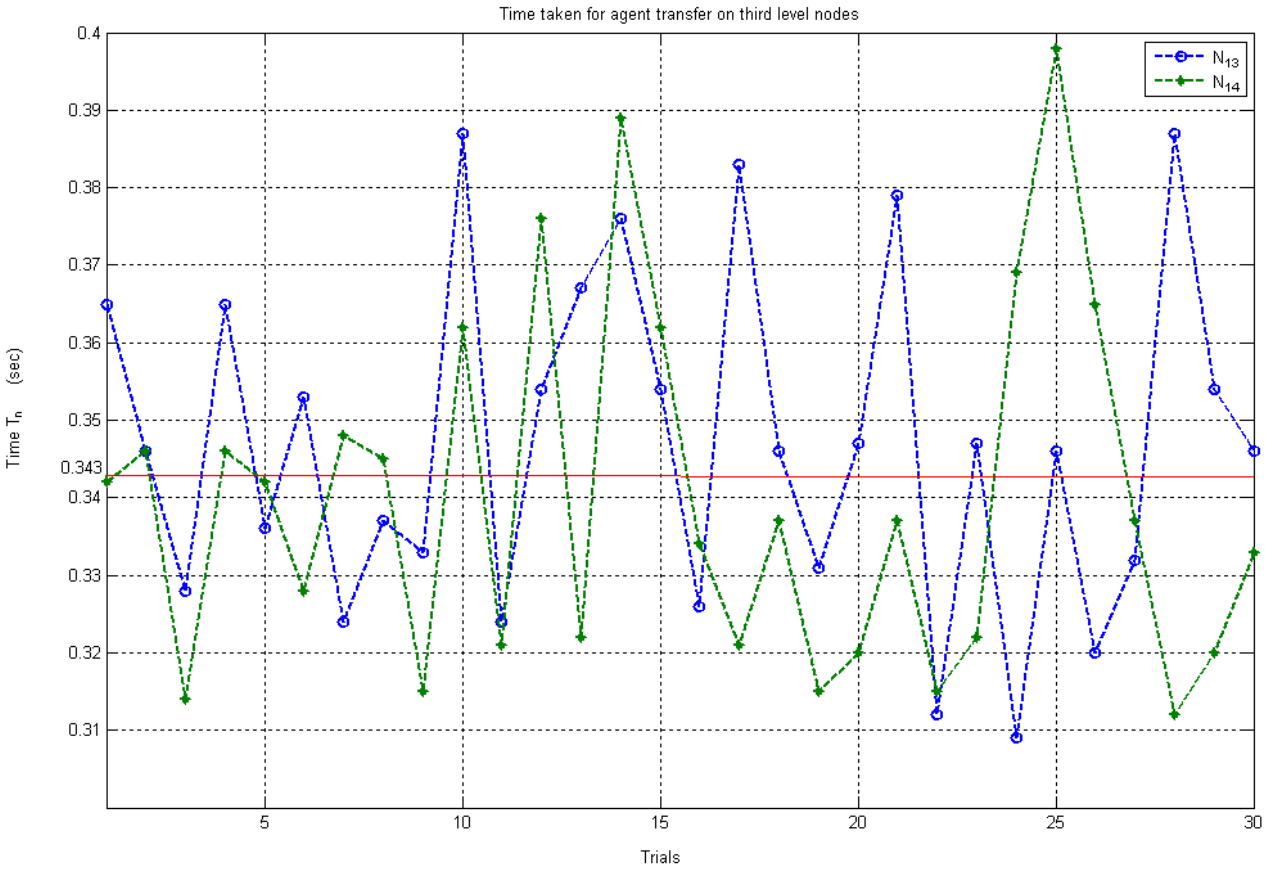

FIG. 5.1. *Time taken for an agent transfer from a computational node in the third level to an adjacent node. Mean time for agent transfer in third level nodes* $MT_{L_3} = 0.343$ *sec.*

concepts of the intelligent agent approach in swarm-array computing and is a preliminary step towards applying autonomic computing concepts to parallel computing systems.

The method proposed is an ample demonstration, though not highly sophisticated, that accommodates the concepts of intelligent agents leading towards achieving a few autonomic computing objectives and attributes.

5. Results. T_{N_n}, the time taken by an agent to transfer from a node N_n predicted to fail onto an adjacent node in the abstracted layer and re-establish all process dependencies for seamless execution was noted. Nodes N_{13} and N_{14} as shown in figure 4.1 are the third level computational nodes of the parallel summation algorithm, and hence are the only nodes considered for calculating T_{N_n} in this paper. Thirty different trial runs were performed to gather the statistic.

Figure 5.1 is the plot that shows T_{N_n} for the third level nodes N_{13} and N_{14} for 30 different trials.

Further, MT_{N_n}, the mean time of T_{N_n} for a particular node was calculated. This metric yields information on the mean time taken by an agent to transfer from a node N_n predicted to fail onto an adjacent node in the abstracted layer and re-establish all process dependencies for seamless execution. MT_{N_n} is calculated as follows:

$$MT_{N_n} = \left(\sum_{TR=1}^{30} T_n \right) \Big/ 30, n = 13, 14 \tag{5.1}$$

and TR being independent trials.

$MT_{L_p}, p = 3$, the mean time taken for an agent transfer from all nodes predicted to fail in the third level of the parallel summation algorithm onto an adjacent node in the abstracted layer is calculated as follows:

$$MT_{L_3} = \frac{1}{2} \sum_{n=13}^{14} MT_{N_n} \tag{5.2}$$

The mean time for an agent transfer from a computational node in the third level to an adjacent node in the abstracted layer is obtained as $MT_{L_3} = 0.343sec$, indicated by a red axis line in figure 5.1.

This statistic is the time taken for reinstating execution after a predicted third level node failure. If other approaches such as traditional checkpointing with human adminstration was employed, reinstating execution would be atleast in the order of minutes. This brief comparison reveals that the intelligent based approach is effective than traditional methods. Hence, applying autonomic computing concepts to parallel computing systems is beneficial.

In short, though preliminary results obtained through simple experiments are presented, the intelligent agent based approach of swarm-array computing proposed in this paper is promising and paves a path for bridging autonomic computing concepts and parallel computing systems.

6. Discussion & Conclusion. The impact that swarm-array computing can bring about can be foreseen by taking into account the industrial or business perspective and research perspective. From the industrial viewpoint, achieving autonomy in parallel computing systems is productive. The path towards autonomy can be equated to increasing reliability of geographically dispersed systems and hence reduction in total cost for maintenance. From the research perspective, achieving mobility of swarm agents in a heterogeneous parallel computing environment opens a new avenue to be explored. Moreover, swarm-array computing can be proposed as a new approach for closer examination and investigation.

From an application oriented point of view, swarm-array computing can be more assuring for applications that demand reliability. One potential application that can be influenced includes space applications. Space crafts employ FPGAs, a special purpose parallel computing system that are subject to malfunctioning or failures of hardware due to 'Single Event Upsets' (SEUs), caused by radiation on moving out of the protection of the atmosphere [36]–[38]. One solution to over-come this problem is to employ reconfigurable FPGAs. However, there are many overheads in using such technology and hardware reconfiguration is challenging in space environments. In other words, replacement or servicing of hardware is an extremely limited option in space environments. On the other hand software changes can be accomplished. In such cases, the swarm-array computing approach can provide solutions based on agent mobility and minimize overheads in software uploading and exclude requirement to reconfigure hardware.

In this paper, a swarm-array computing approach based on intelligent agents that act as carriers of decomposed tasks has been explored. Foundational concepts of a swarm-array computing approach namely intelligent agent based approach is considered. The feasibility of the proposed approach is validated on a multi-agent simulator. Experimental results obtained from a cluster based implementation of a parallel summation algorithm that implements concepts of intelligent agents is presented. Though only preliminary results are presented in this paper, the approach gives ground for expectation that autonomic computing concepts can be applied to parallel computing systems and hence open a new avenue of research in the scientific community.

Future work will aim to study the other swarm-array computing approaches considered in section 2. Efforts will also be made towards implementing the approaches in real time on other parallel computing systems using other existing middleware for parallel computing systems.

REFERENCES

[1] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. Knight, R. Nagpal, E. Rauch, G. Sussman and R. Weiss, *Amorphous computing*, Communications of the ACM, 43(5) (2000).

[2] S. R. Hedberg, *Evolutionary Computing: the spawning of a new generation*, IEEE Intelligent Systems and their Applications, Vol. 13, Issue 3 (2008), pp. 79–81.

[3] H. Schmeck, *Organic Computing - A New Vision for Distributed Embedded Systems*, in the Proceedings of the 8th IEEE Symposium on Object-Oriented Real-Time Distributed Computing, 2005, pp. 201–203.

[4] M. G. Hinchey and R. Sterritt, *99% (Biological) Inspiration*, in the Proceedings of the 4th IEEE International Workshop on Engineering of Autonomic and Autonomous Systems, 2007, pp. 187–195.

[5] P. Lin, A. MacArthur and J. Leaney, *Defining Autonomic Computing: A Software Engineering Perspective*, in the Proceedings of the Australian Software Engineering Conference, 2005, pp. 88–97.

[6] R. Sterritt and M. Hinchey, *Autonomic Computing - Panacea or Poppy-cock?*, in the Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, 2005, pp. 535–539.
[7] R. Sterritt and D. Bustard, *Autonomic Computing - a Means of Achieving Dependability?*, in the Proceedings of the 10th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, 2003, pp. 247–251.
[8] M. R. Nami and M. Sharifi, *Autonomic Computing a New Approach*, in the Proceedings of the First Asia International Conference on Modelling and Simulation, 2007, pp. 352–357.
[9] M. Jarrett and R. Seviora, *Constructing an Autonomic Computing Infrastructure using Cougaar*, in the Proceedings of the 3rd IEEE International Workshop on Engineering of Autonomic and Autonomous Systems, 2006, pp. 119–128.
[10] S. Lightstone, *Foundations of Autonomic Computing Development*, in the Proceedings of the 4th IEEE Workshop on Engineering of Autonomic and Autonomous Systems, 2007.
[11] T. Marshall and Y. S. Dai, *Reliability Improvement and Models in Autonomic Computing*, in the Proceedings of the 11th International Conference on Parallel and Distributed Systems, 2005, pp. 468–472.
[12] T. M. King, D. Babich, J. Alava, P. J. Clarke and R. Stevens, *Towards Self-Testing in Autonomic Computing Systems*, in the Proceedings of the 8th International Symposium on Autonomous Decentralized Systems, 2007, pp. 51–58.
[13] R. J. Anthony, *Emergence: a Paradigm for Robust and Scalable distributed applications*, in the Proceedings of the International Conference on Autonomic Computing, 2004, pp. 132–139.
[14] F. Saffre, J. Halloy, M. Shackleton and J. L. Deneubourg, *Self-Organized Service Orchestration Through Collective Differentiation*, IEEE Transactions on Systems, Man and Cybernetics, Part B (2006), pp. 1237–1246.
[15] A. Zeid and S. Gurguis, *Towards Autonomic Web Services*, in the Proceedings of the 3rd ACS/IEEE International Conference on Computer Systems and Applications, 2005.
[16] J. Almeida, V. Almeida, D. Ardagna, C. Francalanci and M. Trubian, *Resource Management in the Autonomic Service-Oriented Architecture*, in the Proceedings of the IEEE International Conference on Autonomic Computing, 2006, pp. 84–92.
[17] M. Parashar, Z. Li, H. Liu, V. Matossian and C. Schmidt, *Enabling Autonomic Grid Applications: Requirements, Models and Infrastructure*, Lecture Notes in Computer Science, Self-Star Properties in Complex Information Systems, Springer Verlag. Vol. 3460, 2005, pp. 273–290.
[18] Y. Diao, J. L. Hellerstein, S. Parekh, R. Griffith, G. Kaiser and D. Phung, *Self-Managing Systems: A Control Theory Foundation*, in the Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, 2005, pp. 441–448.
[19] Q. Zhu, L. Lin, H. M. Kienle and H. A. Muller, *Characterizing Maintainability concerns in Autonomic Element Design*, in the Proceedings of the IEEE International Conference on Software Maintenance, 2008, pp. 197–206.
[20] J. O. Kephart and W. E. Walsh, *An Artificial Intelligence Perspective on Autonomic Computing Policies*, in the Proceedings of the 5th IEEE International Workshop on Policies for Distributed Systems and Networks, 2004, pp. 3–12.
[21] A. Peddemors, I. Niemegeers, H. Eertink and J. de Heer, *A System Perspective on Cognition for Autonomic Computing and Communication*, in the Proceedings of the 16th International Workshop on Database and Expert Systems Application, 2005, pp. 181–185.
[22] M. Hinchey, Y. S. Dai, C. A. Rouff, J. L. Rash and M. Qi, *Modeling for NASA Autonomous Nano-Technology Swarm Missions and Model-Driven Autonomic Computing*, in the Proceedings of the 21st International Conference on Advanced Information Net-working and Applications, 2007, pp. 250–257.
[23] L. M. F.-Carrasco, H. T.-Marin and M. V.-Rendon, *On the Path Towards Autonomic Com-puting: Combining Swarm Intelligence and Excitable Media Models*, in the Proceedings of the 7th Mexican International Conference on Artificial Intelligence, 2008, pp. 192–198.
[24] H. Guo, J. Gao, P. Zhu and F. Zhang, *A Self-Organized Model of Agent-Enabling Autonomic Computing for Grid Environment*, in the Proceedings of the 6th World Congress on Intelligent Control and Automation, 2006, pp. 2623–2627.
[25] J. Hu, J. Gao, B.-S. Liao and J.-J. Chen, *Multi-Agent System based Autonomic Computing Environment*, in the Proceedings of the International Conference on Machine Learning and Cybernetics, 2004, pp. 105–110.
[26] B. Varghese and G. T. McKee, *Towards Self-ware via Swarm-Array Computing*, in the Proceedings of the International Conference on Computational Intelligence and Cognitive Informatics, Paris, France, 2009, pp. 178–184.
[27] F. Klugl, R. Herrler and M. Fehler, *SeSAm: Implementation of Agent-Based Simulation Using Visual Programming*, in the Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems, Japan, 2006, pp. 1439–1440.
[28] SeSAm website: `http://www.simsesam.de`
[29] Center for Advanced Computing and Emerging Technologies (ACET) website: www.acet.reading.ac.uk
[30] High Performance Computing at ACET website: `http://hpc.acet.rdg.ac.uk/`
[31] W. Gropp, E. Lusk and A. Skjullum, *Using MPI-2: Advanced Features of the Message Passing Interface*, MIT Press (1999).
[32] OpenMPI website: `http://www.open-mpi.org/`
[33] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham and T. S. Woodall, *Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation*, in the Proceedings of the 11th European PVM/MPI Users' Group Meeting, Budapest, Hungary, 2004, pp. 97–104.
[34] MPI Tutorial: `http://www.mpi-forum.org/docs/mpi-20-html/mpi2-report.html`
[35] M. J. Quinn, *Parallel Computing Theory and Practice*, McGraw-Hill Inc. (1994).
[36] M. V. O'Bryan, C. Poivey, S. D. Kniffin, S. P. Buchner, R. L. Ladbury, T. R. Oldham, J. W. Howard Jr., K. A. LaBel, A. B. Sanders, M. Berg, C. J. Marshall, P. W. Marshall, H. S. Km, A. M. Dung-Phan, D. K. Hawkins, M. A. Carts, J. D. Forney, T. Irwin, .C. M. Seidleck, S. R. Cox, M. Friendlich, R. J. Flanigan, D. Petrick, W. Powell, J. Karsh and M. Baze, *Compendium of Single Event Effects Results for Candidate Spacecraft Electronics for NASA* in the Proceedings of the IEEE Radiation Effects Data Workshop, 2006, pp. 19–25.

[37] E. Johnson, M. J. Wirthlin and M. Caffrey, *Single-Event Upset Simulation on an FPGA*, in the Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms, USA, 2002.
[38] S. Habinc, *Suitability of Reprogrammable FPGAs in Space Applications*. Report for the European Space Agency by Gaisler Research under ESA con-tract No. 15102/01/NL/FM(SC) CCN-3, 2002.

Edited by: Costin Bădică, Viorel Negru
Received: March 6, 2010
Accepted: March 31, 2010