Achieving Intelligent Agents and its feasibility in Swarm-Array Computing?

BLESSON VARGHESE AND GERARD MCKEE
Active Robotics Laboratory
School of Systems Engineering
University of Reading, Whiteknights Campus
Reading, Berkshire
UNITED KINGDOM, RG6 6AY
E-mail : b.varghese@student.reading.ac.uk, g.t.mckee@reading.ac.uk

Abstract: - The work reported in this paper proposes 'Intelligent Agents', a Swarm-Array computing approach focused to apply autonomic computing concepts to parallel computing systems and build reliable systems for space applications. Swarm-array computing is a robotics a swarm robotics inspired novel computing approach considered as a path to achieve autonomy in parallel computing systems. In the intelligent agent approach, a task to be executed on parallel computing cores is considered as a swarm of autonomous agents. A task is carried to a computing core by carrier agents and can be seamlessly transferred between cores in the event of a predicted failure, thereby achieving self-* objectives of autonomic computing. The approach is validated on a multi-agent simulator.

Key-Words: - Autonomic computing, Swarm-array computing, Intelligent agents, Carrier agents.

1 Introduction

Autonomic computing has recently emerged as a domain of interest to computing researchers worldwide. What is autonomic computing, and what are its inspiration and vision? What are its distinct perspectives? What are the autonomic approaches? What needs to be focused ahead? These are the few questions answered in this section, before commencing discussions on Intelligent Agents and their feasibility in Swarm-Array Computing, the primary focus of this paper.

What is autonomic computing, and what are its inspiration and vision? With the advancements of computing techniques, biologically inspired computing has emerged as a major domain in computing. Many computing paradigms, namely amorphous computing, evolutionary computing and organic computing have emerged as a result of being inspired from natural phenomenon. Autonomic computing is one such biologically inspired computing paradigm based on the autonomic human nervous system [1].

Autonomic computing is a visionary paradigm for developing large scale distributed systems, reducing cost of ownership and reallocating management responsibilities from administrators to the computing system itself [2] – [9]. Autonomic computing paves the necessary foundation autonomic computing principles have paved necessary foundations towards self-managing systems.

Self-managing [10] systems are characterized by four objectives and four attributes. The objectives and attributes that contribute to self-management are not independent functions. The objectives considered are [1] [11] [12]: (a) Self-configuration – the capability of a computing system to automatically adapt to changes in the existing physical topology and software environment. The system must be also capable to seamlessly integrate new system components. Self-configuring systems are expected to increase resource availability. (b) Self-healing - the capability of a computing system to recuperate from faults and loss. Constant and consistent monitoring of the computing system is required to detect faults and loss. (c) Self-optimizing – the capability of a computing system to automatically tune resources and balance workloads to improve operational efficiency. (d) Self-protecting – the capability of a computing system to protect itself from malicious attacks originating from within and without the system. Self-protection safeguards the system from damages due to uncorrected cascading failures.

The attributes considered are [1] [11] [12]: (a) Self-awareness – the capability of a computing system to be aware of its internal state and knowledge of the possible states the system can transform to from the current state. (b) Self-situated – the capability of a computing system to be aware of the external operating conditions. (c) Self-monitoring – the capability of a computing system to detect the change of internal and external circumstances consistently. (d) Self-adjusting – the capability of a computing system to adapt to internal and external changes reflexively.

What are the perspectives of autonomic computing? There are mainly two perspectives, namely business and research oriented perspectives that provide a bird's-eye view of the paradigm. Firstly, from a business oriented perspective, autonomic computing was proposed by IBM for better management of increasingly complex computing systems and reduce the total cost of ownership of systems today [5] [6], hence aiming to reallocate

management responsibilities from administrators to the computing systems itself based on high-level policies [7] [8]. With the aim to implement autonomic principles in personal computing environments, personal autonomic computing a subset of autonomic computing has also emerged [9].

Secondly, the research oriented perspective primarily focuses on the worms-eye view, laying necessary foundations for the newly emerging computing paradigm. There are two categories of ongoing research in the area of autonomic computing. Firstly, research describing approaches and technologies related to autonomic computing [10]. The aim of the approaches is to achieve autonomy without specifying the technology to be implemented [11]. Any existing technology capable of achieving autonomy (in any degree) can be used in the approaches. Secondly, research attempting to develop autonomic computing as a unified project [10]. The research lays emphasis on the means to achieve autonomy and initiatives are taken to define a set of standard practices and methods as the path towards autonomy.

What are the autonomic computing approaches? 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 [12]. Intelligent behaviour is thus repercussions of interactions and coordination between entities. One major challenge in emergence based approaches is on how to achieve global coherent behaviour [13]. Autonomic computing research on emergence based approaches is reported in [12] - [16].

The component/service based approach for distributed systems employ service-oriented architectures. With advancements in software engineering practices, component/service based approaches are also implemented in many web based services. The autonomic element of the autonomic system is a component whose interfaces, behaviours and design patterns aim to achieve self-management. These approaches are being developed for large scale networked systems including grids. Autonomic computing research on component/service based approaches is reported in [17] - [21].

The control theoretic based approach aims to apply control theory for developing autonomic computing systems. The building blocks of control theory such as reference input, control input, control error, controller, disturbance input, measured output, noise input, target system and transducer are used to model computing systems and further used to study properties like stability, short settling times, and accurate regulation. Using a defined set of control theory methodologies, the objectives of a control system namely regulatory control, disturbance rejection and optimization can be achieved. These objectives are closely associated with the objectives of autonomic computing. Research on control theoretic based approaches applied to autonomic computing is reported in [22] - [24].

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 [25] [26].

The swarm intelligence based approaches focuses on designing algorithms and distributed problem solving devices inspired by collective behaviour of swarm units that arise from local interactions with their environment [27] [28]. The algorithms considered are population-based stochastic methods executed on distributed processors. Autonomic computing research on swarm intelligence approaches is reported in [29] - [32].

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 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 [5] [33] - [39].

What needs to be focused ahead? The focus of researchers in autonomic computing should be towards two directions. Firstly, researchers ought to aim towards applying autonomic computing concepts to parallel computing systems. This focus is essential 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. It is surprising that only few researchers have applied autonomic computing concepts to parallel computing systems in the approaches above.

Secondly, autonomic computing researchers ought to focus towards implementing the approaches for building reliable systems. One potential area of application that demands reliable systems is space applications. Space crafts employ FPGAs, a special purpose parallel computing system that are subject to malfunctioning or

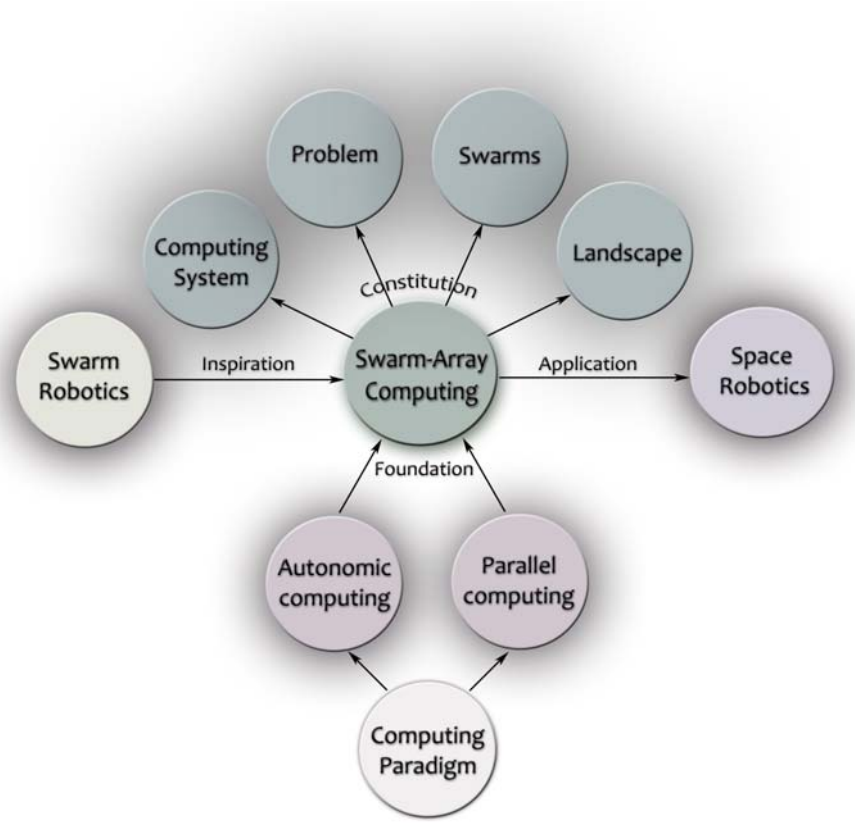

Figure 1. The development of Swarm-Array Computing

failures of hardware due to 'Single Event Upsets' (SEUs), caused by radiation on moving out of the protection of the atmosphere [40] - [42]. One solution to overcome 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, autonomic computing approaches can come to play.

How can a bridge be built between autonomic computing approaches and parallel computing systems? How can autonomic computing approaches be extended towards building reliable systems for space applications? The work reported in this paper is motivated towards bridging this gap by proposing swarm-array computing, a novel technique to achieve autonomy for distributed parallel computing systems and experimenting the feasibility of a proposed approach on FPGAs that can be useful for space applications.

The remainder of the paper is organized as follows. Section 2 introduces swarm-array computing. Section 3 investigates the feasibility of the proposed approach by simulations. Section 4 presents the real-time implementation detail to be considered in future work. Section 5 concludes the paper.

2 Swarm-Array Computing

Swarm-array computing is a swarm robotics inspired approach and is proposed as a path to achieve autonomy. The development of the swarm-array computing is shown in figure 1. The foundations of swarm-array computing are based on parallel and autonomic computing paradigms. The constitution of the swarm-array computing approach can be separated into four constituents. Three approaches are proposed that bind the swarm-array computing constituents together. The four constituents and the three approaches are considered in the following sub sections.

2.1 Constituents

There are four prime constituents that make up the constitution of swarm-array computing. They are the computing system, the problem/task, the swarms and the landscape considered in this section.

Firstly, the computing systems which are available for parallel computing are multi-core processors, clusters,

grids, field programmable gate arrays (FPGA), general purpose graphics processing units (GPGPU), application-specific integrated circuit (ASIC) and vector processors. With the objective of exploring swarm-array computing, FPGAs are selected as an experimental platform for 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 cores of the computing system can be considered as a set of autonomous agents, interacting with each other and coordinating the execution of tasks. In this case, a processing core is similar to an organism whose function is to execute a task. The focus towards autonomy is laid on the parallel computing cores abstracted onto intelligent cores. The set of intelligent cores hence transform the parallel computing system into an intelligent swarm. The intelligent cores hence form a swarm-array. A parallel task to be executed resides within a queue and is scheduled onto different cores by the scheduler. The swarm of cores collectively executes the task.

The intelligent cores described above are an abstract view of the hardware cores. But then the question on what intelligence can be achieved on the set of cores needs to be addressed. Intelligence of the cores is achieved in two different ways. Firstly, by monitoring local neighbours. Independent of what the cores are executing, the cores can monitor each other. Each core can ask the question of 'are you alive' to its neighbours and gain information. Secondly, by adjusting to core failures. If a core fails, the process which was executed on the core needs to be shifted to another core where resources previously accessed can be utilized. Once a process has been shifted, all data dependencies need to be re-established.

To shift a process from one core to another, there is a requirement of storing data associated and state of the executing process, referred to as checkpointing. This can be achieved by a process monitoring each core or by swarm carrier agents that can store the state of an executing process. The checkpointing method suggested is decentralized and distributed across the computing system. Hence, though a core failure may occur, a process can seamlessly be transferred onto another core. In effect, awareness and optimizing features of the self-ware properties are achieved.

Secondly, the problem/task to be executed on the parallel computing cores that can be considered as a swarm of autonomous agents. To achieve this, a single task needs to be decomposed and the sub tasks need to be mapped onto swarm agents. The agent and the sub-problems are independent of each other; in other words, the swarm agents are only carriers of the sub-tasks or are a wrapper around the sub-tasks.

The swarm displaces itself across the parallel computing cores or the environment. The goal would be to find an area accessible to resources required for executing the sub tasks within the environment. In this case, a swarm agent is similar to an organism whose function is to execute on a core. The focus towards autonomy is laid on the executing task abstracted onto intelligent agents. The intelligent agents hence form a swarm-array.

The intelligent agents described above are an abstract view of the sub-tasks to be executed on the hardware cores. Intelligence of the carrier agents is demonstrated in two ways. Firstly, the capabilities of the carrier swarm agents to identify and move to the right location to execute a task. In this case, the agents need to be aware of their environments and which cores can execute the task. Secondly, the prediction of some type of core failures can be inferred by consistent monitoring of power consumption and heat dissipation of the cores. If the core on which a sub-task being executed is predicted to fail, then the carrier agents shift from one core to another gracefully without causing an interruption to execution, hence making the system more fault-tolerant and reliable. An agent can shift from one core to another by being aware of which cores in the nearest vicinity of the currently executing core are available.

Thirdly, a combination of the intelligent cores and intelligent swarm agents leads to intelligent swarms. The intelligent cores and intelligent agents form a multi-dimensional swarm-array. The arena in which the swarms interact with each other is termed as landscape.

Fourthly, the landscape that is a representation of the arena of cores and agents that are interacting with each other in the parallel computing system. At any given instance, the landscape can define the current state of the computing system. Computing cores that have failed and are predicted to fail are holes in the environment and obstacles to be avoided by the swarms.

A landscape is modelled from three different perspectives which is the basis for the swarm-array computing approaches discussed in the next section. Firstly, a landscape comprising dynamic cores (are autonomous) and static agents (are not autonomous) can be considered. In this case, the landscape is affected by the intelligent cores. Secondly, a landscape comprising of static cores and dynamic agents can be considered. In this case, the landscape is affected by the mobility of the intelligent agents. Thirdly, a landscape comprising of dynamic cores and dynamic agents can be considered. In this case, the landscape is affected by the intelligent cores and mobility of the carrier agents.

2.2 Approaches

At this point it is appropriate to consider how the constitution of swarm-array computing fits together? To answer this question, three approaches that combine the constituents of swarm-array computing are proposed.

In the first approach, only the intelligent cores are considered to be autonomous swarm agents and form the landscape. A parallel task to be executed resides within a queue and is scheduled onto the cores by a scheduler. The intelligent cores interact with each other as considered in section 2.1 to transfer tasks from one core to another at the event of a hardware failure.

In the second approach, only the intelligent swarm agents are considered to be autonomous and form the landscape. A parallel task to be executed resides in a queue, which is mapped onto carrier swarm agents by the scheduler. The carrier swarm displace through the cores to find an appropriate area to cluster and execute the task. The intelligent agents interact with each other as considered in Section 2.1 to achieve mobility and successful execution of a task. Figure 2 describes the approach diagrammatically.

In the third approach, both the intelligent cores and intelligent agents are considered to form the landscape. Hence, the approach is called a combinative approach. A parallel task to be executed resides in a queue, which is mapped onto swarm agents by a scheduler. The swarm agents can shift through the landscape utilizing their own intelligence, or the swarm of cores could transfer tasks from core to core in the landscape. The landscape is affected by the mobility of intelligent agents on the cores and intelligent cores collectively executing a task by accommodating the intelligent agent.

However, in this paper the major focus is the second approach and is only considered for experimental studies. The experimental results of the first method is reported in [43].

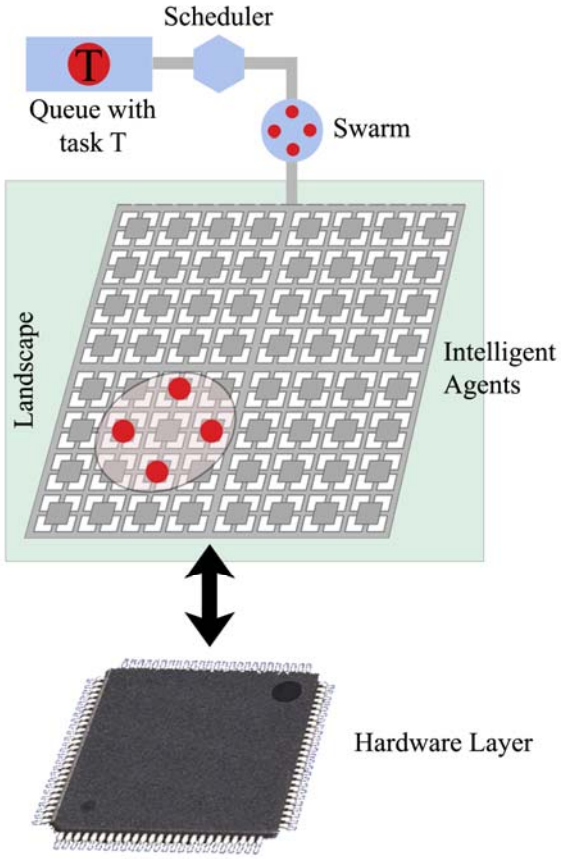

Figure 2. Second Approach in swarm-array computing

3 Simulation Studies

Simulation studies were pursued to validate and visualize the proposed approach in swarm-array Computing. Since FPGA cores are considered in this paper and 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 simulation environment, experimental platform and model and simulation results.

3.1 Simulation Environment

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 [44] [45].

The environment has provisions for modelling agents, the world and simulation runs. Agents are characterized by a reasoning engine and a set of state variables. The reasoning engine defines the behaviour of the agent, and is implemented in the form of an activity diagram, similar to a UML-based activity diagram. The state variables of the agent specify the state of an agent. Rules that define activities and conditions can be visually modelled without the knowledge of a programming language. The building block of such rules is primitives that are pre-defined. Complex constructs such as functions and data-types can be user-defined.

The world provides knowledge about the surroundings the agent is thriving. A world is also characterized by variables and behaviours. The modelling of the world defines the external influences that can affect the agent Hence, variables associated with a world class can be used as parameters that define global behaviours. This in turn leads to the control over agent generation, distribution and destruction.

Simulation runs are defined by simulation elements that contribute to the agent-based model being

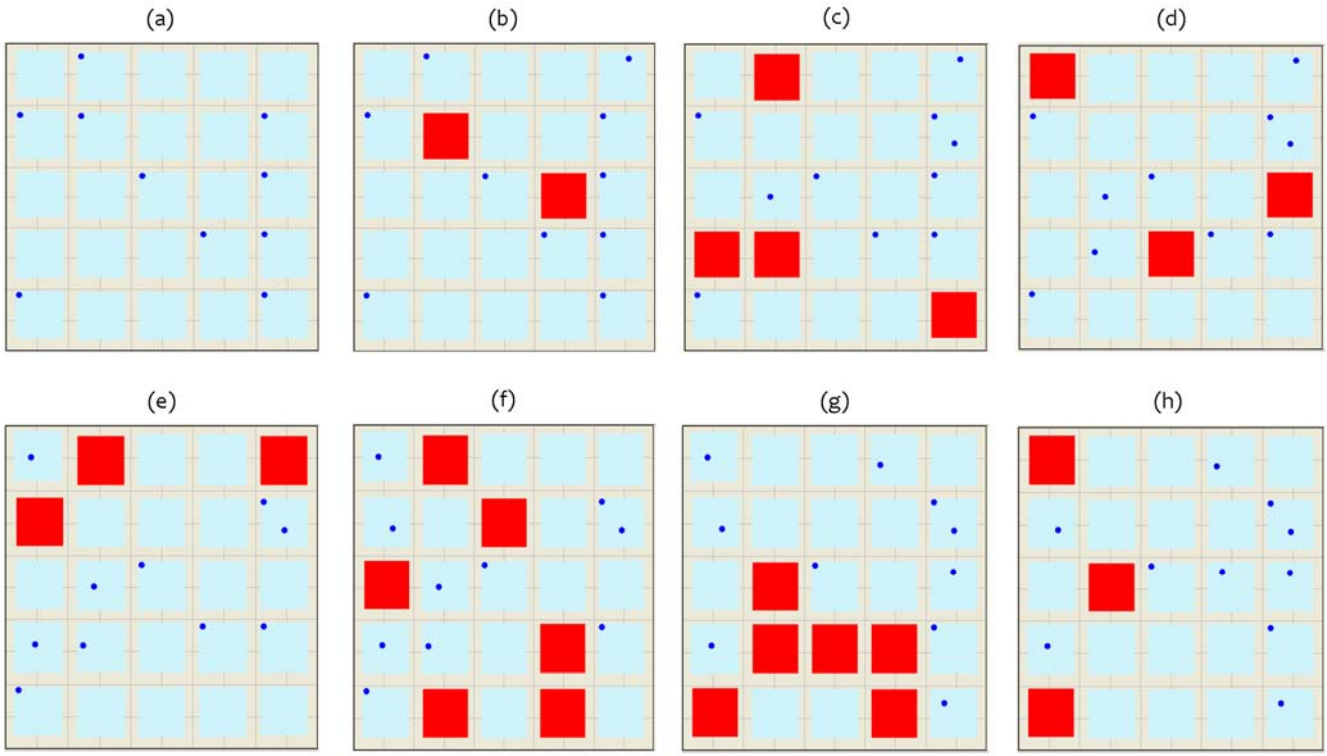

Figure 3. Sequence of eight simulation screenshots (a) – (h) of a simulation run from initialization on the SeSAm multi-agent simulator. Figure shows how the carrier agents carrying sub-tasks are seamlessly transferred to a new core when executing cores fail.

constructed. The simulation elements include situations, analysis lists, simulations and experiments. Situations are configurations of the world with pre-positioned agents to start a simulation run. Analysis lists define means to study agents and their behaviour with respect to time. Simulations are combinations of a situation, a set of analysis items and a simulation run; or in other words a complete definition of a single simulation run. Experiments are used when a combination of single simulation runs are required to be defined.

3.2 Experimental Platform & Model

As considered in section 2.1, the swarm-array computing approach needs to consider the computing platform, the problem/task and the landscapes. The parallel computing platform considered in the studies reported in this paper is FPGAs and is modelled in SeSAm. 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 to 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 grid. Each subtask 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 varies 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 to 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.

3.3 Simulation Results

Figure 3 is a series of screenshots of a random simulation run developed on SeSAm for eight consecutive time steps from initialization. The figure shows the executing cores as rectangular blocks in pale blue 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 Real time Implementation

To implement the second approach considered in Section 2.2, the authors of this paper would like to opt for 'Cluster-based implementation'. Clusters are parallel computing systems, which have nodes linked together. Three basic elements that define a cluster are the collection of individual nodes, a network connecting the nodes, and software that enables the nodes to communicate [46].

The approach will be implemented on the cluster computing systems owned by the Centre for Advanced Computing and Emerging Technologies (ACET) [47] [48] at the University of Reading. The cluster intended to be utilized comprises 16 Linux nodes.

Immediate efforts will be made to implement classical algorithms on the cluster. One such type of algorithms is the parallel reduction algorithms [49] onto which the swarm-array computing approach will be adopted. Parallel reduction algorithms are chosen due to two reasons. Firstly, the criticality of the executing nodes. Each node of the cluster is a critical node while executing a parallel reduction algorithm. This is due to the fact that each node executes and maintains a piece of data important to other nodes. However, the information contained is not replicated and diffused to adjacent nodes. If one node fails, it is most likely that the executing algorithm stalls and requires to be reinstated by a restart operation. Secondly, parallel reduction algorithms are used for critical applications including space applications. Self-management is an inevitable issue in such applications. If there exists any point of failure, it is most likely that the entire application or mission fails.

To implement the approach, Message Passing Interface (MPI) [50] [51] and Parallel Virtual Machine (PVM) [52] [53] will be utilized. MPI provides control over the executing process and hence will be useful to implement intelligent agents. On the other hand, PVM provides better control over the processor on which a process executes (MPI does not offer control over processor) and hence will be useful to implement the intelligent core approach.

Future work will also aim to study the third proposed approach or the combinative approach in swarm-array computing. Efforts will be made towards implementing the approaches in real time as discussed above and hence explore in depth the fundamental concepts associated with the constituents of swarm-array computing.

5 Conclusions

In this paper, a swarm-array computing approach based on intelligent agents that act as carriers of tasks has been explored. Foundational concepts that define swarm-array computing are introduced. The feasibility of the proposed approach is validated on a multi-agent simulator. Though only preliminary results are produced in this paper, the approach gives ground for expectation that autonomic computing concepts can be applied to parallel computing systems and build reliable systems for space applications. Real-time implementation details necessary to achieve the concepts of intelligent cores and intelligent agents are also presented briefly.

References:

[1] 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.

[2] 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.

[3] R. Sterritt and M. Hinchey, "Autonomic Computing – Panacea or Poppycock?" in the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, 2005, pp. 535 – 539.

[4] 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.

[5] M. R. Nami and M. Sharifi, "Autonomic Computing a New Approach" in the First Asia International Conference on Modelling and Simulation, 2007, pp. 352 – 357.

[6] 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.

[7] S. Lightstone, "Foundations of Autonomic Computing Development" in the Proceedings of the 4th IEEE Workshop on Engineering of Autonomic and Autonomous Systems, 2007.

[8] W. Gentsch, K. Iano, D. J.-Watt, M. A. Minhas and M. Yousif, "Self-Adaptable Autonomic Computing Systems: An Industry View" in the Proceedings of the 16th IEEE International Workshop on Database and Expert Systems Applications, 2005.

[9] G. Cybenko, V. H. Berk, I. D. G.-DeSouza and C. Behre, "Practical Autonomic Computing" in the Proceedings of the 30th IEEE Annual International Computer Software and Applications Conference, 2006.

[10] M. R. Nami and K. Bertels, "A Survey of Autonomic Computing Systems" in the Third International Conference on Autonomic and Autonomous Systems, 2007, pp. 26 – 30.

[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] T. De Wolf and T. Holvoet, "Emergence as a general architecture for distributed autonomic computing," K. U. Leuven, Department of Computer Science, Report CW 384, January, 2004.

[15] F. Saffre, J. Halloy, M. Shackleton and J. L. Deneubourg, "Self-Organized Service Orchestration Through Collective Differentiation" in the IEEE Transactions on Systems, Man and Cybernetics, Part B, Dec 2006, pp. 1237 – 1246.

[16]. Privosnik and M. Marolt, "The Development of Emergent Properties in Massive Multi-Agent Systems," in the Proceedings of the WSES International Conference on Multimedia, Internet, Video Technologies, Malta, 2001.

[17] P. Champrasert, C. Lee and J. Suzuki, "SymbioticSphere: Towards an Autonomic Grid Network System" in the Proceedings of the IEEE International Conference on Cluster Computing, 2005, pp. 1 – 2.

[18] 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.

[19] J. Almeida, V. Almeida, D. Ardagna, C. Francalanci and M. Trubian, "Resource Management in the Service Oriented Architecture" in the Proceedings of the IEEE International Conference on Autonomic Computing, 2006, pp. 84 – 92.

[20] S. R. White, J. E. Hanson, I. Whalley, D. M. Chess and J. O. Kephart, "An Architectural Approach to Autonomic Computing" in the Proceedings of the IEEE International Conference on Autonomic Computing, 2004.

[21] A. Moga, J. Soos, I. Salomie and M. Dinsoreanu, "Adding Self-healing Behaviour to Dynamic Web Service Compostion," in the Proceedings of the 5th WSEAS International Conference on Data Networks, Communications & Computers, Bucharest, Romania, 2006.

[22] M. Parashar, .Z. Li, H. Liu, V. Matossian and C. Schmidt, "Enabling Autonomic Grid Applications: Requirements, Models and Infrastructure" in the Lecture Notes in Computer Science, Self-Star Properties in Complex Information Systems, Springer Verlag. Vol. 3460, 2005, pp. 273-290.

[23] 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.

[24] S. Abdelwahed, N. Kandasamy, and S. Neema, "Online Control for Self-Management in Computing Systems," in the Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium, Toronto, Canada, 2004.

[25] 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.

[26] 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.

[27] 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.

[28]M. G. Hinchey, R. Sterritt and C. Rouff, "Swarms and Swarm Intelligence " in IEEE Computer, Vol. 40, No. 4, IEEE Computer Society, April 2007, pp. 111-113.

[29]J. Kennedy, R. C. Eberhart and Y. Shi, "Swarm Intelligence", Morgan Kaufmann Publishers, 2001.

[30]J. Wang, B. J. d'Auriol, Y.-K. Lee and S. Lee, "A Swarm Intelligence inspired Autonomic Routing Scenario in Ubiquitous Sensor Networks" in the Proceedings of the International Conference on Multimedia and Ubiquitous Engineering, 2007, pp. 745 – 750.

[31]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 Networking and Applications, 2007, pp. 250 – 257.

[32]L. M. F.-Carrasco, H. T.-Marin and M. V.-Rendon, "On the Path Towards Autonomic Computing: Combining Swarm Intelligence and Excitable Media Models" in the Proceedings of the 7th Mexican International Conference on Artificial Intelligence, 2008, pp. 192 – 198.

[33]T. De Wolf and T. Holovet, "Towards Autonomic Computing: Agent-Based Modelling, Dynamical Systems Analysis, and Decentralised Control" in the Proceedings of the IEEE International Conference on Industrial Informatics, 2003, pp. 470 – 479.

[34]D. Bonino, A. Bosca and F. Corno, "An Agent based Autonomic Semantic Platform" in the Proceedings of the International Conference on Autonomic Computing, 2004, pp. 189 – 196.

[35]H. Tianfield, "Multi-agent Autonomic Architecture and its Application in e-Medicine" in the Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology, 2003.

[36]G. Pour, "Prospects for Expanding Telehealth: Multi-Agent Autonomic Architectures" in the Proceedings of the International Conference on Computational Intelligence for Modelling and Automation, and International Conference on Intelligent Agents, Web Technologies and Internet Commerce 2006.

[37]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.

[38]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..

[39]A.Schiaffino M. De Franceschi, K. S. Borges, R. Moraes, F. Vasques, "Autonomic Computing Systems: Using AI Techniques for the Development of Agents in the Network Management Domain," in the WSEAS Transactions on Communications, Issue 8, Volume 5, August 2006.

[40]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.

[41]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.

[42]S. Habinc, "Suitability of Reprogrammable FPGAs in Space Applications" a feasibility Report for the European Space Agency by Gaisler Research under ESA contract No. 15102/01/NL/FM(SC) CCN-3, September 2002.

[43]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.

[44]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.

[45]SeSAm website: http://www.simsesam.de

[46]J. D. Sloan, "High Performance Linux Clusters with OSCAR, Rocks, openMosix & MPI," O'Reilly Media, Inc. 2005.

[47] Centre for Advanced Computing and Emerging Technologies (ACET) website: http://www.acet.reading.ac.uk

[48] High Performance Computing website of the ACET Centre: http://hpc.acet.rdg.ac.uk

[49] M. J. Quinn, "Parallel Computing Theory and Practice," McGraw-Hill, Inc, Second Edition, 1994.

[50] S. Vetter, Y. Aoyama and J. Nakano, "RS/6000 SP: Practical MPI Programming," IBM Redbooks, 1999.

[51] W. Gropp, E. Lusk and A. Skjullum, "Using MPI-2: Advanced Features of the Message Passing Interface," MIT Press, 1999.

[52] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek and V. Sunderam, "PVM: Parallel Virtual Machine A Users' Guide and Tutorial for Networked Parallel Computing," MIT Press, 1994.

[53] G. A. Geist, J. A. Kohl and P. M. Papadopoulos, "PVM and MPI: A Comparison of Features," in Calculateurs Paralleles Vol. 8 No. 2, 1996.