Towards Self-ware via Swarm-Array Computing

Blesson Varghese and Gerard McKee

Abstract— The work reported in this paper proposes Swarm-Array computing, a novel technique inspired by swarm robotics, and built on the foundations of autonomic and parallel computing. The approach aims to apply autonomic computing constructs to parallel computing systems and in effect achieve the self-ware objectives that describe self-managing systems. The constitution of swarm-array computing comprising four constituents, namely the computing system, the problem/task, the swarm and the landscape is considered. Approaches that bind these constituents together are proposed. Space applications employing FPGAs are identified as a potential area for applying swarm-array computing for building reliable systems. The feasibility of a proposed approach is validated on the SeSAm multi-agent simulator and landscapes are generated using the MATLAB toolkit.

Keywords—Swarm-Array computing, Autonomic computing, landscapes.

I. INTRODUCTION

RESEARCHERS in the field of computing are often inspired by ideas from nature. The approach of abstracting good design from nature is referred to as biomimetics [1]. In the context of computing, biomimetics has resulted in the emergence of relatively new computing paradigms, cited as biologically-inspired computing [2]. For example, amorphous computing is inspired from a colony of cells operating to form a multi-cellular organism based on a genetic program shared by the members of the colony [3]. Evolutionary computing is inspired from biological evolution mechanisms on a population of individuals [4]. Autonomic computing is also one such biologically-inspired computing paradigm based on human autonomic nervous system [2] that will be the focus of this paper.

Autonomic computing is a visionary paradigm for developing large scale distributed systems [11]. 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]. Autonomic computing solutions hence aims 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.

The aim of autonomic computing paradigm is to achieve autonomy in computing systems. The word autonomy has broad meanings in a philosophical and physiological context. In general, the autonomy of an entity can be referred to as the integrated ability of cognition and mobility [12]. Perception and learning are closely associated with cognition resulting in planning, inference and decision making. For example, gaining information from the environment is one such cognitive capability of an entity resulting in some inference about its environment. On the other hand, mobility refers to the capability of an entity to move around in an environment. It is worthwhile to note that cognition and mobility are complementing abilities. An entity can achieve higher degree of mobility in an environment due to its complementing cognitive capability, and higher degree of cognitive capabilities due to its complementing mobility.

Autonomy from the research perspective of autonomic computing is defined by self-management [10], and is characterized by four objectives and four attributes. The objectives and attributes that contribute to self-management are not independent functions. The objectives considered are [13]: (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 [15]. Self-configuring systems are expected to increase resource availability [14]. (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 [14]. (c) Self-optimizing – the

Blesson Varghese is a research student with the Active Robotics Laboratory, School of Systems Engineering, University of Reading, Reading, Berkshire, United Kingdom, RG6 6AY.(e-mail: hx019035@ reading.ac.uk).

Gerard T. McKee is Senior Lecturer in Networked Robotics, School of Systems Engineering, University of Reading, Reading, Berkshire, United Kingdom, RG6 6AY. (e-mail:g.t.mckee@reading.ac.uk).

capability of a computing system to automatically tune resources and balance workloads to improve operational efficiency [15]. (d) Self-protecting – the capability of a computing system to protect itself from malicious attacks originating from within and without the system [14]. Self-protection safeguards the system from damages due to uncorrected cascading failures [15].

The attributes considered are [13]: (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.

The benefits of autonomy in computing systems are also apparent for parallel computing systems, namely reducing cost of ownership and reallocating management responsibilities to the system itself. The parallel computing paradigm employs the concurrent utilization of multiple processing elements to solve a problem [16]. Wide ranges of problems have found quicker solutions by utilizing parallel computational power, since the processor-memory bottleneck is addressed. For example, parallel computing is useful for developing problem solvers that engage in computationally intensive operations and voluminous data processing requiring high processing rate.

The work reported in this paper focuses on the research oriented perspective of autonomic computing applied to parallel computing and proposes Swarm-Array computing as a path to achieve autonomy. The approach is biomimetically inspired by the theory of autonomous agents in natural swarms, abstracted and implemented in swarm robotics. FPGA cores are considered as the parallel computing system and an approach to implement swarm-array computing is proposed. The cores of the FPGA are considered to be autonomous agents with a high degree of self-managing capabilities. Subtasks to be executed reside upon a landscape of intelligent cores. The feasibility of the proposed approach is validated using SeSAm simulator and two dimensional landscapes are generated on the MATLAB toolkit.

A review of swarm computing literature reveals the existence of swarm computing and swarm intelligence research. Swarm computing [17, 18] considers a large number of small independent devices that communicate with each other to perform an assigned task. The approach mainly targets the realization of distributed miniature computing devices as swarm units executing similar swarm programs based on primitives.

Swarm intelligence research focuses on designing algorithms and distributed problem solving devices inspired by collective behaviour of swarm units that arise from local interactions with their environment [19, 20]. The algorithms considered are population-based stochastic methods executed on distributed processors.

On the contrary, swarm-array computing is an approach that not only considers the computational resource as a swarm of resources, but also 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. The interactions between swarm agents 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. The interactions between different swarms give rise to the notion of landscapes. In other words, the approach can be viewed as a computational approach emerging from the interaction of multi-dimensional arrays of swarm agents.

The remainder of the paper is organised as follows. Section II considers the swarm-array computing constitution and the various self-ware properties that are achieved in effect in the approach. Section III proposes three different approaches to tie together the constituents of swarm-array computing. Section IV highlights the impact that swarm-array computing can bring about. Experimental studies are described in Section V. Section VI concludes and presents future work.

II. SWARM-ARRAY COMPUTING

As discussed in the section above, parallel computing can also benefit from the application of the autonomic computing paradigm. However, which path should be adopted to achieve this autonomy in parallel computing systems? In this context, swarm-array computing, a swarm robotics inspired approach is proposed as a path to achieve autonomy. The development of the swarm-array computing approach from the foundations of parallel and autonomic computing is shown in Figure 1. The constitution of the swarm-array computing approach can be separated into four different constituents, namely the computing system, the problem / task, the swarms and the landscape as shown in Figure 1. Each constituent is considered in the following sub sections.

A. The Computing System

The computing systems 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 implementing 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 a parallel 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
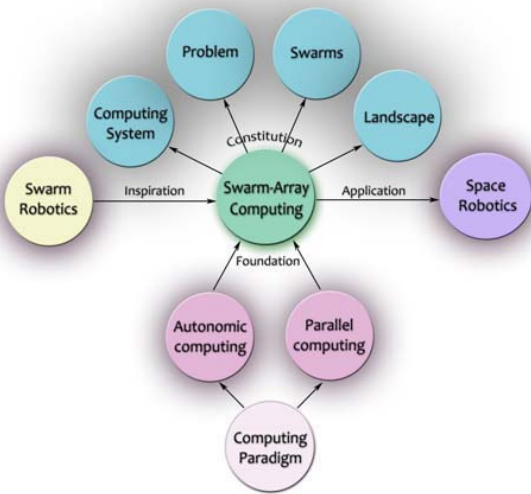
Fig. 1. The development of Swarm-Array Computing

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.

B. The Problem / Task

The task to be executed on the parallel computing cores 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 or 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 to a goal 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.

C. The Swarms

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 considered in the next sub-section.

D. The landscape

The landscape 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.

III. 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 II A to transfer tasks from one core to another at the event of a hardware failure. Figure 2 describes the approach diagrammatically.

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 II B to achieve mobility and successful execution of a task.

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 first approach and is only considered for experimental studies.

IV. IMPACT

Will the emergence of swarm-array computing have an impact over parallel and autonomic computing? This question can be answered 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. Decentralized checkpointing accomplished by the aid of swarm carrier agents is another interesting area of research. With these objectives, swarm-array computing can hence be proposed as a new approach for closer examination and investigation.

Swarm-array computing can be more assuring for applications that demand reliability. Potential applications that can be influenced include space applications and cloud computing. 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 [21] - [23]. 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
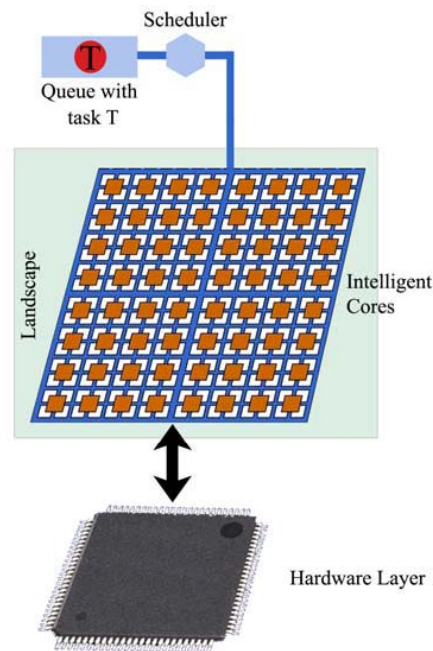

Fig. 2. First Approach in swarm-array computing

software changes can be accomplished. In such cases, the swarm-array computing approach can provide solutions based on agent mobility within the abstracted landscape and hence minimize overheads in software uploading and exclude requirement to reconfigure hardware.

V. EXPERIMENTAL STUDIES

Simulation studies were pursued to validate and visualize the proposed approach in Swarm-Array Computing. 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 first approach proposed in this paper considers executing cores as agents; hence a multi-agent simulator is employed. The remainder of this section is organized into describing the experimental environment, modelling the experiment and generating landscapes of intelligent cores.

A. Experimental 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 [24] [25].

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
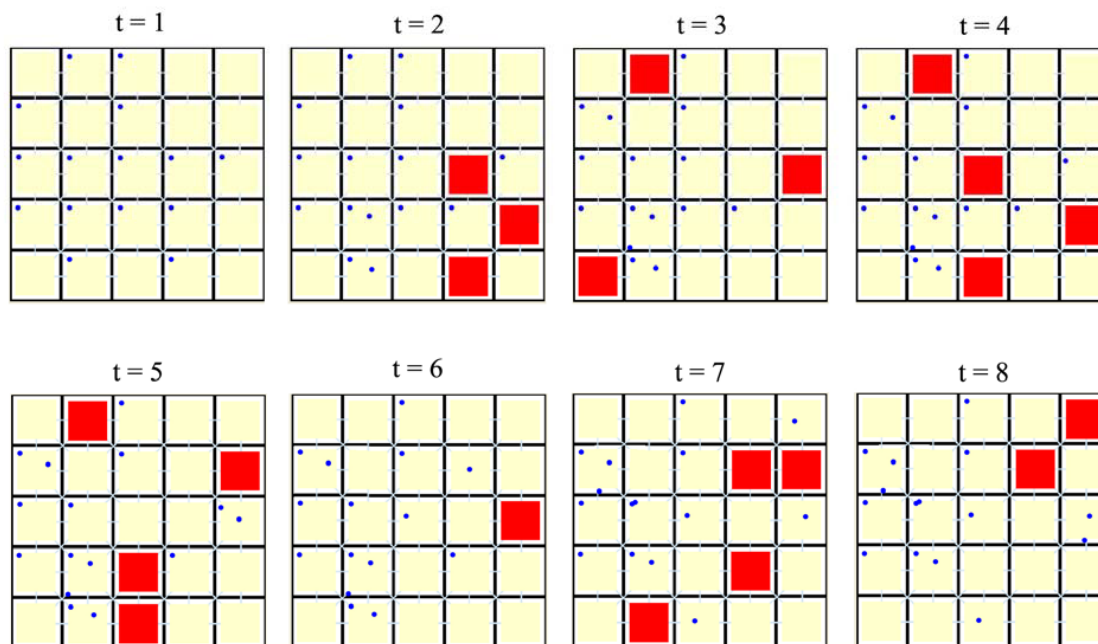
Fig.3. Screenshots of the simulation on SeSAm multi-agent simulator for eight consecutive time steps

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 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.

B. Modelling

As considered in Section II, 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. The cores of the FPGA are modelled as agents in SeSAm, in accordance with the swarm-array computing approach reported in this paper. The intelligent cores are an abstraction of the hardware cores 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 abstraction.

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 core maintains a record of the subtasks it is executing and can monitor cores in the regular grid to which the subtasks can be assigned in the event of a predicted failure. 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 executed on the core requires reassignment to another available core that is not predicted to fail. During the event of a transfer or reassignment, the record of the subtask maintained by the core is also transferred to the new core. If more than one sub-task is executed on a core predicted to fail, each sub-task may be reassigned to different cores.

C. Generating Landscapes

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 yellow colour. When a core is predicted to fail, i.e., temperature increases beyond a threshold, the core is displayed in red. The subtasks 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.
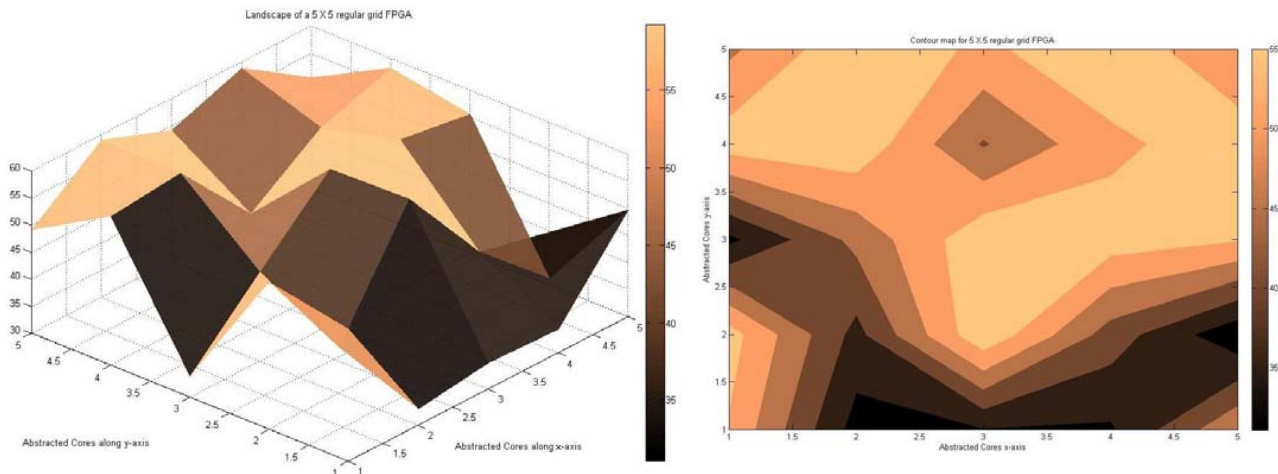
Fig. 4. 3D surface plot (left) and contour map (right) of landscape for a 5 X 5 regular grid FPGA
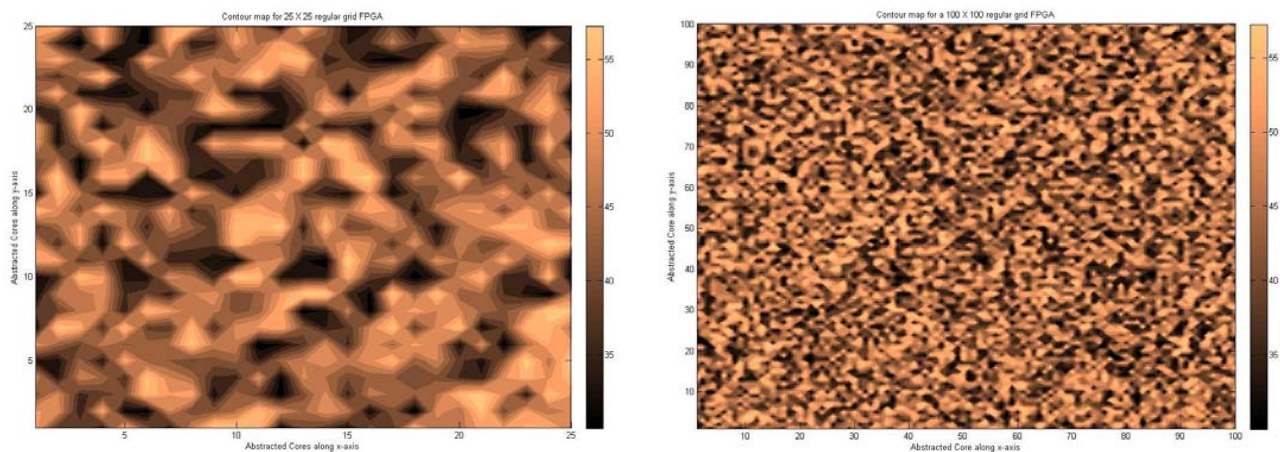

Fig. 5. Contour map of landscape for a 25 X 25 regular grid FPGA (left) and 100 X 100 regular grid FPGA (right)

A log was recorded of the temperature of individual cores of the FPGA during a simulation. A 3D plot of the multi-core landscape during an arbitrary time step, where the z-axis represents the temperature was generated on the MATLAB toolkit. Figure 4 (left) shows the 3D representation of the landscape. Since the landscape is uneven, a contour plot was also generated for better visualization. Figure 4 (right) shows the contour plot of the landscape. Since figure 4 is the representation of the landscape of a 5 X 5 regular grid, the landscape has less resolution. Hence higher resolution contour plots were generated. Figure 5 (left) shows the contour plot for a 25X 25 regular grid during an arbitrary time step of the simulation run and figure 5 (right) shows the contour plot for a 100X100 regular grid during an arbitrary time step of the simulation run.

The generated landscape provides a pictorial view of the computing space and the problem. According to the colour bar provided in the legend of the plots, a light shade on the contour plot represents cores of higher temperature while the dark shaded regions represent safe cores of the FPGA. From the 3D plot it is understood that the peaks of the landscape are cores predicted to fail while the valleys and lower elevations on the

landscape represent safe cores of the FPGA. In the approach considered in this paper, the sub task always resides on the safe cores of the FPGA. In other words, within the landscape sub tasks thrive in valleys and lower elevations.

From a different perspective, the peaks in the landscape can be considered as obstacles for the executing sub tasks. Hence, the sub tasks avoid the peaks by being autonomously and seamlessly transferred to safer regions within the landscape. The notion of obstacles in a landscape is also seen within the domain of swarm robotics, where a swarm of mobile autonomous agents on a landscape avoids obstacles to reach a goal. The landscape in the approach presented in this paper is dynamic (subject to change every time step) and hence makes the computing space and problem much more complex than a static landscape usually considered in swarm robotics. Eventually, the problem of applying autonomic computing constructs to parallel computing platforms is funnelled to a classic, yet a complex swarm robotics problem of obstacle avoidance.

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 constructs to parallel computing platforms.

VI. CONCLUSIONS

In this paper, swarm-array computing, a novel technique to apply autonomic computing constructs in parallel computing is proposed. The foundation and inspiration of the approach is introduced. The constitution of swarm-array computing and how the constituents can be utilized to achieve autonomic properties are described. Three approaches that bind the constitution of swarm-array computing are proposed. The impact of swarm-array computing on emerging as a new avenue for research is pointed out. Space applications employing FPGAs are identified as an area that can be influenced by swarm-array computing. The feasibility of one among the three proposed approaches is presented in this paper.

Future work will include identifying the major challenges that need to be addressed in swarm-array computing. Emphasis will be laid on how the swarm-computing approaches can be steered into real time implementation. Further effort will be made to explore the concepts of abstraction, decentralized checkpointing, intelligent cores and intelligent agents.

REFERENCES

[1] M. K. Habib, K. Watanabe, and K. Izumi, "Biomimetics Robots from Bio-inspiration to Implementation" in the Proceedings of the 33rd Annual Conference of the IEEE Industrial Electronics Society, 2007.

[2] 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.

[3] 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), May 2000.

[4] S. R. Hedberg, "Evolutionary Computing: the spawning of a new generation" in the IEEE Intelligent Systems and their Applications, May – June 2008, Vol. 13, Issue 3, pp. 79 – 81.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] R. Sterritt and D. F. Bantz, "Personal Autonomic Computing reflex reactions and healing" in the IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews, 2006, pp. 304 – 314.

[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] 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.

[12] 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 Applications, 2005, pp. 181 – 185.

[13] 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.

[14] 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.

[15] 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.

[16] G. S. Almasi and A. Gottlieb, "Highly Parallel Computing," Benjamin-Cummings Publishers, 1989.

[17] R. K. Persaud, "Investigating the Fundamentals of Swarm Computing," a Bachelor of Science in Computer Science thesis, University of Virginia, 2001.

[18] A. Seth, "Scalability and Communication within Swarm Computing," a Bachelor of Science in Computer Science thesis, University of Virginia, 2003.

[19] 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.

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

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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