A Fault Tolerance Approach Based on Reinforcement Learning in the Context of Autonomic Opportunistic Grids

Alcilene Dalília de Sousa and Luciano Reis Coutinho
Departamento de Informática
Universidade Federal do Maranhão, UFMA
São Luis, Maranhão - Brasil
e-mail: alcileneluzsousa@gmail.com / lrc@deinf.ufma.br

***Abstract* — Fault tolerance is a longstanding problem. Two basic solutions are replication and checkpointing, both with their pros and cons. In this paper, we put forward an approach to balance replication and checkpointing in order to provide fault tolerance in opportunistic grid computing systems. We try to retain the benefits of both techniques, while avoiding their downsides. The approach combines reinforcement learning with the MAPE-K architecture for autonomic computing. To validate our proposal, we have performed experiments based simulation using the Autonomic Grid Simulator Tool (AGST). We report promising results. We show that the proposed approach is able to learn suitable switching thresholds between checkpointing and replication. The suitability is verified by comparing the average completion time and the success rate of applications of our proposal against the values from other approaches in the literature.**

Keywords - ***fault tolerance; grid computing; opportunistic grids; autonomic computing; reinforcement learning.***

I. INTRODUCTION

Achieving a high processing rate by dividing computational tasks among several geographically distributed machines is, in essence, the core idea behind the computational model called *Grid Computing* [9]. In this context, it was introduced the concept of *opportunistic grids* that, potentially, gather thousands of resources, services and applications to provide greater computational power at a lower cost [12]. On the one hand, this type of computational grid promotes the use of non-dedicated resources, such as desktop workstations situated in different administrative domains, by using their idle processing power. On the other hand, there is the challenge of providing services in a dynamic execution environment, where heterogeneous nodes can enter and leave the grid at any time. In this case, it is important to be able of effectively monitoring the grid composition in order to detect and react to these events in a timely manner.

Grid computing encompasses various technical challenges. One of them, especially in the context of opportunistic grids, is how to provide *fault tolerance* in an inherently dynamical environment, an environment in which computing nodes are heterogeneous and can become unavailable at any time. Fault tolerance is the ability of a system to continue to work even in the presence of faults [7]. We say that a fault has occurred when one of system components fails or malfunction, leading to a behavior not in accordance with the system specifications. Concerned with this problem, researchers have been seeking for solutions. Among these, we found some approaches based on the idea of *autonomic computing*. Broadly, the idea of autonomic computing consists in modeling and building computing systems that have the ability of self-management and self-adaptation to unpredictable changes [6][8][10]. Applied to the problem of fault tolerance in opportunistic grids, autonomic computing has given rise to approaches where the grid middleware tries to automatically adjust parameters or to dynamically combine traditional fault tolerance techniques such as *checkpointing* and *replication* [2][15][16][18].

In general, these autonomic approaches provide important gains w.r.t. the traditional fault tolerance techniques. Despite these gains, there are some opportunities for improvement. In this paper, we focus on the following issue regarding the approach presented in [15] [16] (and based on [2]). When should we switch from checkpointing to replication, and vice-versa, given the current workload of an opportunistic grid system? Viana [15] fixes a threshold below which the grid performs replication and above which it performs checkpoint. Our hypothesis is that this threshold can vary, and that this variation is beneficial to the fault tolerance strategy of the grid system. It can lead to fewer delays in the execution time of the application due to fault tolerance concerns.

To test this hypothesis, we put forward an adaptive approach to the problem of balancing checkpointing and replication in the context of opportunistic grids. Our approach is based on the use of *reinforcement learning*. Reinforcement learning is a paradigm of machine learning based on trial-and-error and delayed reward [1][13]. A typical reinforcement learning problem consists in finding a policy (a decision function) that maps environmental states to actions. Such a policy is optimal when it can be used to guide our behavior through the environment in such a way that we obtain maximum cumulative reward over time. In the case of the problem of balancing checkpointing and replication, we want an optimal (or near optimal) policy that help us to decide, directly or indirectly, when it is the right time to switch from checkpointing to replication, and vice-versa. By choosing reinforcement learning we are following the path of several researchers in the areas of Grid, Cloud and Autonomic Computing [3][4][14][17][19].

 ISBN: 978-1-61208-331-5

The paper is organized as follows. In Section II, we further discuss the concepts of *grid* and *autonomic computing*, and briefly present AGST, the *Autonomic Grid Simulation Tool* we have used in our experiments. In Section III, we characterize the need and challenge of *fault tolerance* in grid computing environments, and report on the state of the art. In Section IV, we present the basic ideas of *reinforcement learning* and related algorithms. In Section V, we describe our approach to the problem of fault tolerance in opportunistic grid which merges ideas from autonomic computing and reinforcement learning. Section VI reports some experimental results we have obtained when evaluating the approach. Finally, in Section VII, we draw our conclusion and discuss future work.

II. GRID AND AUTONOMIC COMPUTING

Grid computing is a particular ideal case of distributed and parallel computing [9]. It seeks to extend the potential of computer networks to enable the sharing, selection and aggregation of geographically distributed and possibly heterogeneous computing resources (e.g., processors, data and applications) in a pervasive and transparent way. The idea is that individual users (client applications) can access computing resources as needed with a minimum knowledge of localization, underlying technologies of hardware and software, etc.

A. Opportunistic Grids

Opportunistic grids are grid computing systems that promote the dynamical integration of non dedicated workstations, possibly distributed along several administrative domains (e.g., organizations, academic laboratories, home PCs, etc.), by using their idle computing time to the execution of parallel applications [12]. This way, opportunistic grids are highly heterogeneous and dynamic. They aggregate regular personal computers, with their particular hardware and software configurations, to execute distributed application in large scale. And these machines can enter and leave the grid at different times, using network connections with different capabilities with regard to properties such as bandwidth, error rate and communication latency.

From the point of view of the user, the grid computing system should be viewed as a single integrated resource and should still be easy to use. These are some of the challenges that an opportunistic grid middleware face. The *grid middleware* is a software layer between the operating systems running on the computing nodes and the user applications submitted to the grid. The focus of an opportunistic grid middleware is not the integration of dedicated computer clusters [11], or to provide supercomputing resources, but to promote a better use of existing computing resources and the execution of computationally intensive parallel applications.

B. Autonomic Computing

Autonomic computing, as a research field, aims at developing computational systems able to manage themselves with minimal human intervention [6][8][10]. The term autonomic comes from biology and is inspired from the human nervous system. Like the nervous system, an autonomic computing system must possess some characteristics or properties such as *self-awareness*, *self-configuration*, *self-protection*, *self-optimization* and *self-healing*, among others [6]. In sum, these characteristics relate autonomic computing to the design of complex systems; systems that need constant adjustments to various dynamical circumstances, as is the case of opportunistic grid systems.

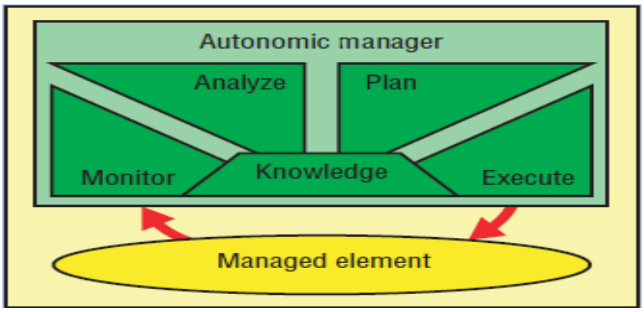

Figure 1. Architecture of an Autonomic Element [8].

In general, an autonomic computing system is conceived as one or more *autonomic elements* composed of an *autonomic manager* associated to a *managed element* (see Figure 1). The managed element represents a resource or device (computer, printers, databases, etc.) composing a computing system. The autonomic manager is an active component that encapsulates a managed element to turn it into an autonomic element. The internal working of an autonomic manager is organized in cycles, each one divided in four distinct phases: *monitoring*, in which data is collected by means of sensors; *analysis*, in which possible needs and problems are detected; *planning*, in which strategies are drawn to make necessary changes and adaptations; and *execution*, in which the planned strategies are effectively implemented. This general architecture is known as the MAPE-K model.

C. Autonomic Grid Simulator Tool

In our research, we have used the *Autonomic Grid Simulator Tool* (AGST)[15], a tool that allows the modeling and simulation of autonomic approaches to self-management problems in the context of Opportunistic Grids [5]. AGST is based on the MAPE-K model, providing support for all phases of the autonomic manager execution cycle. By using AGST, one can simulate autonomic approaches based on two types of dynamic adaptation: *parametric* and *compositional*. Parametric adaptation consists in the continual modification of variables that determine the behavior of algorithms used by the grid middleware. And compositional adaptation is the exchange of algorithms or components of the grid middleware, enabling the adoption of new strategies to handle new situations and to react to changes in the grid environment.

 ISBN: 978-1-61208-331-5

III. FAULT TOLERANCE

In opportunistic grid systems, and distributed systems in general, it is important to have mechanisms that allow the system to continue functioning despite the presence of faults [7]. Faults occur every time the system behavior does not comply with its intended operation due to some failure or malfunctioning of one or more of its components. What happens after a fault determines the degree of fault tolerance of the system. At one extreme, we have full fault tolerant systems in which fault does not decrease their quality of service. At other, there are zero fault tolerant systems in which any fault causes total system breakdown. Between these extremes, we found systems that present a graceful degradation characterized by a reduction in the quality of their operations proportional to the severity of the fault. Given that failures are inevitable in dynamic and heterogeneous systems, and that the cost of zero fault tolerance can be very high, fault tolerance becomes a characteristic of paramount importance.

A. Basic Techniques

A basic source of faults in opportunistic grid systems is the loss of running tasks due to problems on a grid node. These can be caused by many factors, among them the computer be turned off by its owner. In general, this is the kind of fault tolerance we deal in our work, and that will be considered in the remainder of the paper.

Ultimately, the less sophisticated way to deal with the fact that nodes may become unavailable is to detect the node failure and restart, in another available node, the tasks that were running on it. To avoid this restart, the researchers have devised two basic techniques: *replication* and *checkpointing* [7]. Replication consists in executing several replicas of the same task in different nodes at the same time. On the one hand, with several replicas, the change of having to restart a task due to a node failure is minimized. On the other hand, the grid middleware has to manage and synchronize several running replicas that consume computational resources and leave less space for scheduling new incoming tasks. Looking for a better use of resources, checkpointing is a technique that promotes fault tolerance by periodically saving the state of the running tasks so that they can be resumed on a different machine in the case of a node failure. However, the downside of this technique is the time overhead imposed upon the tasks that need to be constantly interrupted by the grid middleware to record their running states.

Replication and checkpointing have pros and cons depending on different conditions of the grid environment. If there are plenty of computing nodes in comparison to number of tasks to be run, then replication is a better option than checkpointing. But, to the extent that the number of available computing nodes decreases, replication becomes less and less attractive until we reach a point where checkpointing becomes a better option than replication. With these pros and cons in mind, in the last few years, some researchers have proposed adaptive fault tolerance mechanisms that try to autonomically balance the use of replication and checkpointing depending on the current grid condition [2][15][18].

B. State of the Art

Wu et al. [18] propose a mechanism based on the number of times a task is resumed due to node failures. Initially, for each new task, the grid middleware performs checkpointing. If a node failure occurs, the task is restarted from the last saved state in the same computing node. It is considered that it was only a transient fault and that the restarting on the same node is sufficient to solve the problem. If a second fault occurs, it is considered that probably the node in which the task is running is not stable and, therefore, the task is restarted from the last checkpoint on another node. If the task fails a third time, it is considered that the grid environment has a high fault rate and, therefore, the middleware starts multiple replicas of the task to be executed simultaneously.

Chtepen et al. [2] present heuristics for the adaptive use of checkpointing, replication and a combination of them. The goal is to improve resource utilization and reduce the execution time of tasks. In the case of checkpointing, one heuristics consists in increasing or decreasing the interval between checkpoints for each task according to the mean time between failures (MTBF) of the computing nodes. Regarding replication, another heuristics is to limit the use of replication according to the system workload (grid occupancy). A third heuristics is to dynamically switch from checkpointing to replication, and vice-versa, based on workload (if occupancy is high use checkpointing, otherwise use replication).

Based on the work by Chtepen et al. [2] and the MAPE-K model, Viana et al. [15][16] propose an autonomic fault tolerance mechanism for opportunistic grids. The basic idea is to make each computing node a managed element controlled by an autonomic manager. Thus, the autonomic manager continually adjusts the parameters of the fault tolerance technique currently in use for each node of the grid. It also makes a structural reconfiguration, replacing the fault tolerance technique in use by another one, when the system workload reaches a given fixed threshold.

Despite being adaptive (by combining replication and checkpointing taking into account the current state of the grid environment), these state of art mechanisms still depend upon certain parameters that need to be adjusted empirically by the system administrator. One example is the fixed threshold to switch between checkpointing and replication found in [15]. Another limitation perceived is that [2] and [15] rely only on a measure of grid occupancy to switch between checkpointing and replication. As can be seen in the work [18] other factors such as rate mean time between failures (MTBF) can also have a decisive influence in this decision.

Motivated by these shortcomings, we put forward an extension of the work by Viana et al. [15]. Our proposal,

 ISBN: 978-1-61208-331-5

presented in Section V, is inspired by some studies that use *Reinforcement Learning* for resource allocation in computing grids, such as [3][4][14][17][19].

IV. REINFORCEMENT LEARNING

Reinforcement learning is a *Machine Learning* paradigm [1][13] that addresses the issue of how an *agent* (i.e., an autonomous entity that perceive and act in an *environment*) can interactively learn the right *policy* to achieve a given purpose (see Figure 2).

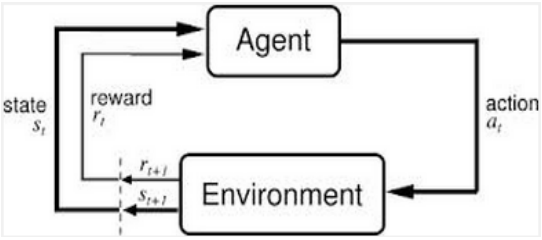

Figure 2. The agent-environment interaction [13].

A. Markov Decision Process

Formally, the problem faced by the agent in a reinforcement learning scenario is rendered as a *Markov Decision Process* (MDP)[13]. A MDP is characterized by a finite set of environmental *states S*; a finite set of possible *actions A*; a *state transition function* $T: S \times A \rightarrow \Pr(S)$ that gives, for each state and action pair, a probability distribution over the set of states (where $T(s, a)(s')$ is the probability of the environment transit to state s' when the agent perform action a in state s); and a (expected) *reward function* $R: S \times A \rightarrow \mathbb{R}$ that maps each state and action pair to a real value representing the (expected) immediate reward after performing action a in state s. In this setting, the functions T and R abstract the dynamics of the environment and are not necessarily known to the agent.

To solve a MDP is to come up with an optimal *policy* $\pi: S \rightarrow A$, i.e., a decision function that maps each possible state to an action in such a way to produce, over time, the greatest possible cumulative reward to the agent.

B. Temporal Difference Learning

In general, reinforcement learning algorithms are based on estimating value functions that characterize optimal policies. One of these functions maps state action pairs (s, a) to real values $Q(s, a)$ that are estimations of the cumulative reward that the agent is expected to receive in the long run if it performs the action a in the state s.

Two popular algorithms for learning Q value functions are Q-learning and SARSA [13]. Both are *Temporal Difference Learning* (TD Learning) algorithms. This means that they work by using the difference between the current and previous estimates to incrementally update Q values. Specifically, the update rule in SARSA is $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$; and in Q-learning is $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$; where α is a step size parameter, γ is the discount factor for future rewards, $r = R(s,a)$ and s' the observed next state when the action a was performed in the current state s.

The basic difference between Q-learning and SARSA lies in the action used to update the Q values. In Q-learning the update is done with the optimal action obtained from a greedy choice. Regarding SARSA, it is used the next action obtained when the agent follows the policy derived from the current Q values. In practice, this difference is reflected in the optimality and in the safeness of the learned policy. Q-learning tends to find policies with expected cumulative rewards higher than SARSA. However the polices found by SARSA are safer than those by Q-learning, in the sense of obtaining less negative rewards when the agent deviates from the policy to explore new possibilities.

V. PROPOSED APPROACH

As discussed in Section III, recent autonomic mechanisms for fault tolerance in grid systems have used heuristic rules that require empirical adjustments of some parameters. Discovering these parameters is not always an easy task. Thus, we propose a fault tolerance approach that extends the work by Viana et al. [15][16] by using Reinforcement Learning to automatically adjust the threshold used to switch between checkpointing and replication. Instead of relying only in the occupancy level of the grid, the idea is to make this switching also dependent on the amount and reliability of the computing nodes composing the grid system.

A. Adaptive Switching Threshold

The autonomic fault tolerance mechanism in [15], based on [2], deals with two levels of adaptation: parametric and structural adaptation. Regarding parametric adaptation, some parameters such as number of replicas or frequency of checkpointing are dynamically adjusted depending on the grid statistics (e.g., MTBF and grid workload). With respect to structural adaptation, what occurs is the switching between checkpointing and replication based on the current grid occupancy.

We retain these basic ideas from [15]. However, we add to the parametric adaptation a further item: the *switching threshold between checkpointing and replication* (measured in terms of grid occupancy percentage). To do this parametric adaption, we develop an approach in which the autonomic manager learns by reinforcement how to increment or decrement the switching threshold in order to minimize the execution time of successfully completed applications. In this way, the grid middleware initially adopts a switching policy reflecting the threshold proposed in [15]. Over time, to the extent that applications complete, the middleware try to increment or decrement the threshold guided by the amount of delayed or restarted applications. At the end, the switching threshold is modified to a value below or above the default value reflecting the particular

characteristic of the grid environment (e.g., MTBF, grid workload and number of computing nodes).

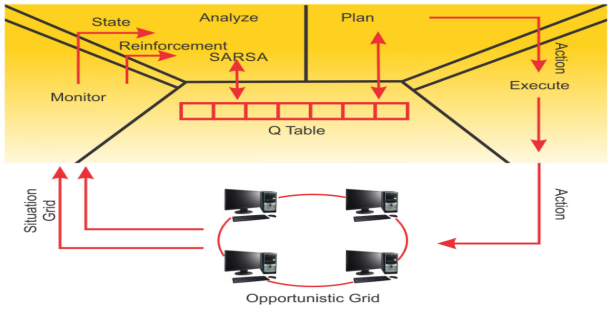

Figure 3. Proposed approach.

In Figure 3, we depict the proposed approach. It is an adaptation of the MAPE-K model shown in Figure 1. The managed elements are the nodes of an opportunistic grid. The autonomic manager, in addition to dealing with the parametric and structural adaptations laid down in [15], is supposed to record and update the policy for changing the switching threshold. This policy is represented as a Q-table, a table that holds a Q value function. During the analysis phase, the Q-table is updated given the current state and reinforcement coming from the grid environment. The update is performed by using the SARSA algorithm (here we have opted for the safeness at the expense of optimality). In the planning phase, the policy coded in the Q-table is used to update the switching threshold and to decide if a structural adaptation (change to checkpointing or replication) is needed or not.

B. Markov Decision Process

In sum, the problem of adapting the switching threshold, formalized as a MDP, consists in: states $s \in \{$ *threshold values in terms of grid occupancy percentage* $\}$; actions $a \in \{$ *increment, decrement, maintain* $\}$; state transition function $T(s,a)$ is deterministic and previously known since increment and decrement assume their mathematical meaning of addition and subtraction, and maintain means leave the threshold value unaltered; immediate reward function $R(s,a)$ is unknown, but delayed negative reward values are calculated from the amount of delayed and restarted tasks (i.e., for each autonomic cycle, the number of delayed and restarted tasks is counted and aggregated as a negative reward; it is used a weighted sum in which a restart is 10 times worse than a delayed task).

VI. EVALUATION OF THE APPROACH

To evaluate the proposed approach, we have conducted several simulation experiments using AGST (Section II-c). To put into perspective the results obtained we compare our approach with the traditional techniques of checkpointing, replication, and the autonomic approach reported in [15].

A. Scenarios

Here, we consider two basic scenarios: 1 – many resources, many faults, 100 applications; and 2 – few resources, many faults, 200 applications.

In the first scenario, we created a simulation model with 1400 computing nodes; in the second, 700 nodes. In both, the nodes were interconnected by a network of 100 Mbps. The medium processing power was equivalent to a Pentium IV 1.6 GHz (1,858 MIPS, based on the TSCP 5 benchmark); to simulate heterogeneity, this medium varies according to a uniform distribution U (938; 2,779) MIPS, where the processing power of the faster machine is approximately three times greater than the processing power of the slower machine. Regarding the faults, AGST was configured to generate synthetic failures with an exponential distribution with MTBF equal to 500 seconds. The duration of failure (downtime) was determined by an exponential distribution with variable mean, whose minimum and maximum values were respectively 300 and 600 seconds (faults with fast recovery, typical of opportunistic grids environments where frequent failures are due to restarting of machines by the users, or electrical current fluctuations, instead of long terms failures such as hardware failures). At last, concerning the grid workload, the applications consist in *bag-of-tasks* applications with three tasks each, resulting in a total of 300 tasks in the first scenario and 600 tasks in the second. These applications were generated with a variation in size (in terms of millions of instructions) according to a uniform distribution U (53,510; 321,062) MI (considering the medium processing power of 1,858 MIPS, each application would take approximately from 8 to 48 hours to complete). All these settings are similar to the settings found in [15] to easy the comparison.

B. Simulations

By combining the four fault tolerance strategies (checkpointing, replication, autonomic [15] and our approach denoted in the sequel as RLearning) and the two scenarios we reach at eight different simulations. In the simulation involving checkpointing, the technique was configured to perform the checkpoint of the tasks on a fixed interval of 30 minutes. With regard to replication, it was configured to statically create three replicas for each task. The autonomic approach was configured as described in [15]. Specifically, the threshold adopted for switching between checkpoint and replication is 30% (i.e., when the grid workload is < 30% use replication, otherwise use checkpointing). Finally, the RLearning approach follows the same configurations of the autonomic approach, with the difference that the switching threshold is variable.

All the eight different simulations were repeated 40 times, resulting in a total of 320 experiments. The metrics used to compare the fault tolerance strategies were the average completion time (in hours) and the success rate of the applications (percentage of application that concluded execution without restarting).

 ISBN: 978-1-61208-331-5

C. Results

The simulation results are shown in Figures 4 and 5.

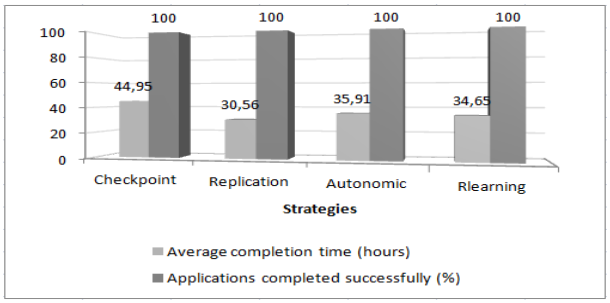

Figure 4. Scenario 1 – 1,400 nodes, 300 tasks and many faults.

Analyzing Figure 4, it is possible to notice that when there plenty of resources the best strategy is to use replication only. It leads to the smallest average completion time (30.56) with 100% of successfully completed applications. The worst is checkpointing only. In between these extremes we have the autonomic and the RLearning approaches. They both approximate the results of replication only by using replication when there are many idle nodes. As applications are submitted and replicas are created, the grid occupancy begins to increase, forcing a switch to checkpointing when the workload reaches the predefined threshold value. This explains why the autonomic and RLearning approaches are beaten by replication in a scenario with many resources and few applications.

When we focus on the autonomic and the RLearning approaches, we see that RLearning was a better approximation to replication than the autonomic approach was. We attribute this difference to the fact that the RLearning dynamically changes the switching threshold between checkpointing and replication. Broadly, the RLearning approach tries other threshold values; if these values do not produce negative rewards, then they became adopted by the system. Thus, the tendency is the threshold to converge to a higher value that prevents the system to incur in an earlier than needed use of checkpointing.

Analyzing Figure 5, we observe how the fault tolerance strategies fare when the amount of resources decreased and the number of application increase. In this case, the checkpointing only remains as the worst approach. However, replication only is not the best approach if we deem the loss of running tasks as an undesirable event. In comparison to checkpointing and the autonomic approach, replication has lost 5% of the running applications (this loss occurs when all replicas are killed due to node failures).

Looking specifically to the autonomic versus the RLearning, we realize that the RLearning approach pursue the average completion time of the replication only strategy (which is the lowest of all approaches), while trying to avoid application loss. In our experiments this loss was less than 0,03%, a small value compared to the 5% obtained by replication. In this way, we judge that the RLearning approach arrived at a good tradeoff between average completion time and application successful completion, w.r.t. the other approaches.

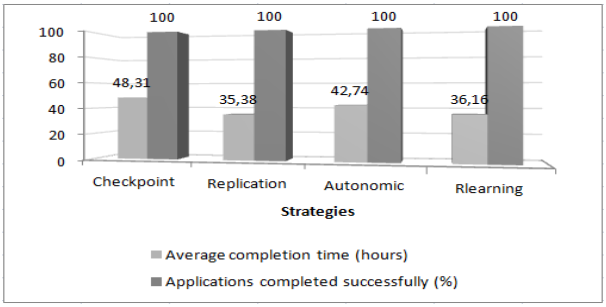

Figure 5. Scenario 2 – 700 nodes, 600 tasks and many faults.

Finally, we call the reader's attention to the fact that we have performed other simulation experiments than these that were reported. We have varied number of resources, applications and fault rates. In general, the results have shown that we obtain better levels of adaptation to the greed characteristics by using the RLearning over the autonomic approach.

VII. CONCLUSION AND FUTURE WORK

In this paper, we have dealt with the problem of providing fault tolerance in opportunistic grid environment, by balancing the use of checkpointing and replication. Building upon the state of art, we have proposed the use of the MAPE-K model together with reinforcement learning as a viable approach to decide the exact point when checkpointing should be used instead of replication, and vice-versa. Our reinforcement learning approach was evaluated by means of simulation models developed by using AGST. The obtained results have corroborated our initial hypothesis that the switching threshold between checkpointing and replication should not be a fixed value, but may dependent on the amount of resources per applications and the reliability of the computing nodes composing the grid system.

Fault tolerance is a challenging problem. Currently, we are exploring the aspects of convergence versus continual policy modification lying at the heart of our approach. For this we are performing further experimental evaluation by means of simulation. As future work we plan to extend the approach to deal with other parameters discussed in [15]. For example, we can try to learn the number of replicas, or the interval between checkpoints. Finally, at long run, we also plan to experiment the approach in a real grid middleware. In this regard, we are thinking about the InteGrade middleware [12].

ACKNOWLEDGMENT

Thanks to the Graduate Program in Computer Science from the Federal University of Maranhão and the Foundation for Research Support of Maranhão (FAPEMA) for the financial support, Proc. BM-01440/13.

REFERENCES

[1] E. Alpaydin, Introduction to Machine Learning (2nd Ed). Cambridge, MA: MIT Press, 2010.

[2] M. Chtepen, F. H. A. Claeys, B. Dhoedt, F. Turck, P. Demeester, and P. A. Vanrolleghem, "Adaptive Task Checkpointing and Replication: Toward Efficient fault-tolerant grids," IEEE transactions on parallel and distributed systems, vol. 20, no. 2, Feb. 2009.

[3] X. Dutreilh, S. Kirgizov, O. Melekhova, J. Malenfant, N. Rivierre, and I. Truck, "Using Reinforcement Learning for Autonomic Resource Allocation in Clouds: Towards a Fully Automated Workflow," The Seventh International Conference on Autonomic and Autonomous Systems (ICAS 2011) IARIA, May 2011.

[4] A. Galstyan, K. Czajkowski, and K. Lerman, "Resource allocation in the grid using reinforcement learning," Third International Joint Conference on Autonomous Agents and Multiagent Systems – Vol. 3 (AAMAS 2004), IEEE Computer Society, Jul. 2004, pp. 1314-1315, doi:10.1109/AAMAS.2004.232

[5] B. T. Gomes, and F. J. Silva e Silva, "AGST - Autonomic Grid Simulation Tool. A Simulator of Autonomic Functions Based on the MAPE-K Model," First International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2011), Jul. 2011, pp. 354–359.

[6] S. Hariri, B. Khargharia, H. Chen, J. Yang and Y. Zhang, "The Autonomic Computing Paradigm," Cluster Computing, vol. 9, pp. 5–17, 2006.

[7] P. Jalote, Fault Tolerance in Distribuited Systems. Englewoods Cliffs, NJ: Prentice Hall, 1994.

[8] J. O. Kephart, and D. M. Chess, "The vision of autonomic computing," IEEE Computer, vol. 36, n. 1, pp. 41–50, Jan. 2003.

[9] F. Magoules, J. Pan, K. Tan, and A. Kumar, Introduction to Grid Computing. CRC Press, 2009.

[10] M. Parashar, Z. Li, H. Liu, V. Matossian, and C. Schmidt, "Enabling Autonomic Grid Applications: Requirements, Models and Infrastructure," In: Self-star Properties in Complex Information Systems (SELF-STAR 2004), Springer LNCS, vol. 3460, pp. 273-290, 2005.

[11] P. R. Prins, "Teaching parallel computing using beowulf clusters: a laboratory approach," J. Comput. Sci. Coll., vol. 20, n. 2, pp. 55–61, Dec. 2004.

[12] F. J. da Silva e Silva, F. Kon, A. Goldman, M. Finger, R. Y. Camargo, F. F. Castor, and F. M. Costa, "Application Execution Management on the InteGrade Opportunistic Grid Middleware," Journal of Parallel and Distributed Computing, vol. 70, n. 5, pp. 573–583, May 2010.

[13] R. S. Sutton, and A. G. Barto, Reinforcement Learning: An Introduction. Cambridge, MA: MIT Press, 1998.

[14] G. Tesauro, "Reinforcement Learning in Autonomic Computing: A Manifesto and Case Studies," IEEE Internet Computing, vol. 11, n. 1, pp. 22-30, 2007.

[15] A. E. Viana, An Autonomic Approach to Fault Tolerance in Running Applications on Desktop Grids. Master's thesis, Universidade Federal do Maranhão, São Luís, MA, Brasil, 2011.

[16] A. E. Viana, B. Gomes, J. Gonçalves, L. R. Coutinho, and F. J. Silva e Silva, "Design and Evalutation of Autonomic Fault Tolerance Strategies Using the AGST Autonomic Grid Simulator," Latin American Conference On High Performance Computing (CLCAR 2011), Sep. 2011.

[17] Z. Wang, X. Qiu, and T. Wang, "A Hybrid Reinforcement Learning Algorithm for Policy-based Autonomic Management," 9th International Conference on Service Systems and Service Management (ICSSSM 2012) IEEE, Jul. 2012, pp. 533–536, doi:10.1109/ICSSSM.2012.6252294

[18] Y. Wu, Y. Yuan, G. Yang, and W. Zheng, "An Adaptive Task-Level Fault-Tolerant Approach to Grid," Journal of Supercomputing, vol. 51, n. 2, pp. 97–114, Feb. 2010.

[19] Z. Zhai, "Grid Resource Selection Based on Reinforcement Learning," Int. Conference on Computer Application and System Modeling – Vol. 12 (ICCASM 2010) IEEE, Oct. 2010, pp. 644 – 647, doi:10.1109/ICCASM.2010.5622441

 ISBN: 978-1-61208-331-5