AGST - AUTONOMIC GRID SIMULATION TOOL
A Simulator of Autonomic Functions based on the MAPE-K Model

Berto de Tácio Pereira Gomes[1] and Francisco José da Silva e Silva[2]
[1]*Electrical Engineering Graduate Program (PPGEE), Universidade Federal do Maranhão (UFMA)*
Av. dos Portugueses, s/n, Campus Universitário do Bacanga, CEP 65085-580, São Luís, MA, Brazil
[2]*Department of Computer Science (DEINF), Universidade Federal do Maranhão (UFMA), São Luís, MA, Brazil*

Keywords: Computer grids, Discrete event simulator, Autonomic computing, MAPE-K cycle.

Abstract: This paper describes AGST, a simulator tool for autonomic grids based on discrete events. The simulator provides, among others, tools for modeling grid resources and their network interconnections, grid applications and their submissions, the occurrence of resource faults, resources local workload, and the use of workload and fault traces following the SWF and FTA standards. AGST major contribution is the definition of a simulation model of the autonomic management cycle MAPE-K, which allows the simulation and evaluation of autonomic grid middleware behavior, providing support for the monitoring, analysis, planning, and dynamic execution of reconfiguration actions to be applied to the simulated grid components.

1 INTRODUCTION

A computer grid is a system that coordinates distributed resources, using standard protocols and interfaces in order to allow the integration and sharing of computing resources, such as computing power, software, data, and peripherals in cooperate networks and among institutions. The key component of a grid architecture is its middleware, which is used to hide the heterogeneous nature and the complexity generated by the distribution of its resources. The middleware provides to its users and applications an homogeneous vision of the environment, through standardized interfaces for its several services.

The high scalability and heterogeneity of modern computer grid environments and the dynamism of its applications and infrastructure makes infeasible approaches based exclusively on the intervention of human agents to perform tasks such as configuration, maintenance, and recovery in case of failures. For these reasons, it should be provided automated mechanisms that could facilitate the computer grid management. The term autonomic computing has been used to denote computer systems capable of dynamically adapting their behavior in response to variations in their execution environment according to established policies and objectives, similar to the self-regulatory behavior of biological systems.

Recent research efforts seek to apply autonomic computing techniques to grid computing, providing more autonomy and reducing the need for human intervention in the maintenance and management of these computing environments, thus creating the concept an autonomic grid. Some of the investigated subjects include the development of autonomic mechanisms for self-protection, such as detecting overloads that could potentially lead to disruption of services; self-optimization, through parameter settings and algorithms to detect performance degradations; self-healing, to overcome possible partial system failures; and self-configuration, by providing configured virtual machines on demand and their dynamic allocation to physical resources (Germain-Renaud and Rana, 2009) (Jha et al., 2009) (Collet et al., 2010) (Abraham et al., 2010).

During the development of a grid middleware, researchers often use simulation tools to validate new concepts and implementations. Simulation tools play a key role in the development of grid middleware, since: (a) researchers often do not have access to large grid environments for testing, limiting the ability to evaluate situations that require a large amount resources, (b) it is difficult to explore large-scale application scenarios that involve multiple resources and users in a repetitive and controlled way, due to the dynamic nature of grid environments, and (c) real-world grid applications are usually time consuming and can run for several weeks.

Over the last few years, several grid simulators were developed, usually based on discrete events,

de Tácio Pereira Gomes B. and José da Silva e Silva F..
AGST - AUTONOMIC GRID SIMULATION TOOL - A Simulator of Autonomic Functions based on the MAPE-K Model.
DOI: 10.5220/0003601603540359
In *Proceedings of 1st International Conference on Simulation and Modeling Methodologies, Technologies and Applications* (SIMULTECH-2011), pages 354-359
ISBN: 978-989-8425-78-2

such as: GridSim (Buyya and Murshed, 2002), SimGrid (Howell and McNab, 1998), Alea (Klusáček et al., 2008), GSSIM (Kurowski et al., 2007), DSiDE (Chtepen et al., 2009), and OGST (Cunha Filho and da Silva e Silva, 2008). None of them focused on the development of a simulation model tailored to the evaluation of autonomic grid features, such as monitoring the grid environment, analysis of context information, reconfiguration planning and the implementation of strategies that would allow the dynamic adaptation of the grid environment. Therefore, grid simulators do not have native support for the evaluation of autonomic computing techniques to be applied on computers grids.

The aim of this paper is to present AGST, an autonomic grid simulator based on discrete events. AGST major contribution is the definition of a simulation model for evaluating autonomic grid behavior based on the MAPE-K autonomic architecture (Kephart and Chess, 2003), providing support for monitoring, analysis, planning, and dynamic execution of reconfiguration actions to be applied to the simulated grid components. This article is divided as follows: Section 2 provides a brief theoretical background on MAPE-K autonomic architecture; Section 3 describes the AGST simulator, its principles, motivations and grid simulation model; Section 4 describes some related work, while in Section 5 we draw our conclusions and describe some future research directions.

2 MAPE-K ARCHITECTURE

The MAPE-K autonomic model, proposed by IBM (Kephart and Chess, 2003) and illustrated in Figure 1, is a general architecture for the development of autonomic software components. This model is being increasingly used to interrelate the architectural components of autonomic systems. It is divided into two main software components: the autonomic manager and the managed resource.

The managed resource corresponds to the system or component providing the business logic that will be dynamically adapted as the computing environment changes. The managed resource can be, for instance, a Web server, a database, a software component in a given application (e.g. the query optimizer in a database), an operating system, etc. In an autonomic grid, the task scheduler can be, for instance, a managed resource. The autonomic manager performs the functions comprising the adaptation logic of the managed resource: monitoring, analysis, planning, and adaptation execution.

The MAPE-K model requires two types of touch-

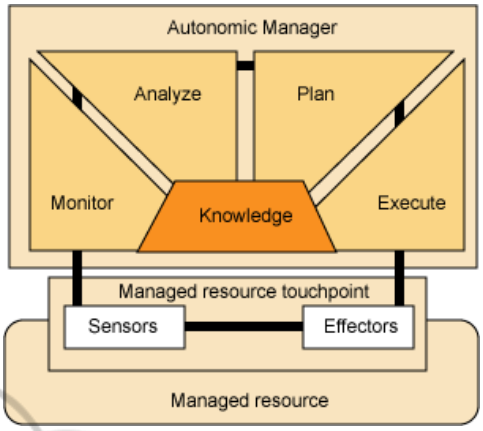

Figure 1: MAPE-K: Autonomic Management Loop.

points within and outside the managed resource: sensors and effectors. Only they have direct access to the managed resource. Sensors are responsible for collecting information from the managed resource, which can be, for instance, the customers requests response time, if the managed resource is a Web server. The information collected by the sensors are sent to the monitors where they are interpreted, preprocessed and placed in a higher level of abstraction. They are then sent to the next step of the cycle, the analysis and planning phase. This stage produces an action plan, which consists of a set of adaptation actions to be performed by the executor. The effectors are the components that allow the autonomic managers to perform adjustments in managed resources. The decision of which adaptation actions must be applied in a given situation requires knowledge representation of the computing system and its environment.

3 AGST

The AGST (Autonomic Grid Simulation Tool) main goal is to easy the development of autonomic grid middleware by providing a set of tools that allows the evaluation of new autonomic concepts and their implementations. In this way, the primary motivation for the AGST development was to provide a tool that supports simulation and evaluation of autonomic computing techniques applied to computer grids. AGST major innovation considering other well known grid simulators is the ability to simulate the functions defined in the MAPE-K autonomic management cycle, allowing the simulation of autonomic grid mechanisms.

ASGT[1] was built based on OGST (Cunha Filho

[1]www.lsd.ufma.br/~agst

and da Silva e Silva, 2008), the GridSim (Buyya and Murshed, 2002) toolkit and the SimJava framework (Howell and McNab, 1998). SimJava is a discrete event driven simulation framework that that models the system as a set of entities that communicate among themselves through events. The (discrete) simulation time steps increases based on the occurrence of these events. GridSim is a software platform that allows users to model and simulate the characteristics of grid resources and networks with different settings. All tools were developed with the Java platform.

3.1 Grid Simulation Model

The AGST allows an easy and flexible modeling of entities that comprise the grid computing environment, such as users, resources, applications, routers, and so on. Typical grid environments events can also be easily generated, such as the occurrence of resources failures and their recovery events. The entities that are part of the grid simulation model are described in the following subsections.

3.1.1 Grid Resources and Network Links

GridSim provides the necessary entities for modeling grid resources (`GridResource`), such as machinery (`Machine`) and processors (`EP`). AGST extensions to GridSim allows the automatic generation of grid resources through the use of probability distributions. For this end, users must provide some parameters, such as the number of nodes to be generated, the probability distribution that best models the heterogeneity of the required resources, and the medium or maximum and minimum processing power (defined in MIPS - Millions of Instructions Per Second), depending on the probability distribution used.

The `GridResourcesGenerator` component is responsible for the grid resources generation, while the `ResourceDataStorage` entity stores information about the resources availability. The generated resources can be of two types: dedicated (without having local users applications running on background) or non-dedicated (with a defined workload simulating the execution of local user applications).

The simulation model allows the automatic generation of a grid environment in which the resources are spread across several administrative domains (sites). The simulator user must provide parameters used for defining the sites interconnection network, such as bandwidth, latency and the number of routers (entity `Router`).

3.1.2 Failures

Resources fault events (entity `FailureEvent`) and their recovery (entity `RecoveryResource`) can be synthetically generated or extracted from fault traces databases collected from real environments using the FTA (Failure Trace Archive) standard[2] (Kondo et al., 2010). In the latter case, the user must provide the database trace URL. The synthetic generation of faults is based on probability distributions, such as exponential, Weibull and hyperexponential (Chwif and Medina, 2007). The user must provide parameters according to the distribution used. For example, when using an exponential distribution, she/he must supply the MTBF (Mean Time Between Failures) and MTTR (Mean Time To Restore) values.

In AGST, applications execution recovery due to node failures can be based on the restarting technique (tasks running on the failed node are re-executed from start on other grid nodes) or on the use of checkpoints. It is also possible to simulate the applications execution using replication, another technique commonly used in grid computing. The `ApplicationReplicationManager` entity is responsible for managing the replicas execution.

3.1.3 Applications and Task Scheduling

Extensions to GridSim provided by AGST allow the automatic generation of two types of applications: regulars (consisting of a single task) and bag-of-tasks (consisting of several independent tasks, running the same code in parallel using different input data). GridSim does not embody an explicit application model, providing only a basic entity representing a task called `Gridlet`. Thus, on GridSim, applications generation is not automatic. The arrival time of each application (entity `Application`) is set in seconds, while their size is defined in Million of Instructions.

In AGST, the automatic generation of applications may occur in three ways: (1) synthetically, through the use of probability distributions (uniform, Poisson, exponential, hyperexponetial) to generate the tasks size and the applications arrival time. The user must choose the appropriate distributions considering the desired simulation scenario; (2) from workload traces following the GWF (Grid Workload Format)[3] and SWF (Standard Workload Format)[4] standards; or (3), through the Lublin model, a detailed model for appli-

[2] http://fta.inria.fr/

[3] http://gwa.ewi.tudelft.nl/pmwiki/

[4] http://www.cs.huji.ac.il/labs/parallel/workload/swf.html

cations generation that was based on the analysis of real grid environments traces.

The GridSim simulation model does not provide grid scheduling algorithms. AGST implements a library of widely used scheduling algorithms in grids environments, such as: InteGrade, Min-Min, MCT, and WQR (Gomes et al., 2010). New strategies can also be easily added to the available library. `GridScheduler (GS)` is the entity responsible for mapping tasks to grid nodes and running the scheduling algorithm (encapsulated into a `SchedulingStrategy (SS)` entity). Each `GridResource` runs a `Resource Controller (RC)` entity, responsible for the instantiation and execution of tasks scheduled to the node, keeping a list of tasks waiting to be executed.

3.2 MAPE-K Simulation Model

In order to allow the simulation of autonomic grid mechanisms, we developed in AGST a model inspired by the MAPE-K autonomic management cycle, shown in Figure 1. The autonomic simulation model contains the following entities: `Sensor`, `Monitor`, `Analyzer`, `Executor`, and `Effector`. These entities are responsible for the autonomic management functions of monitoring, analysis and planning, adaptation control and execution. The autonomic entities communicate among themselves through events. Each entity performs a well defined function of the MAPE-K loop, forming the autonomic manager. Any grid entity can be implemented as a managed resource.

3.2.1 Monitoring

AGST monitoring entities aims to obtain data that reflect changes in the behavior of the managed resource or collect information from the simulated grid environment that are relevant to the self-adaptation process. For example, one may be interest in performing some kind of adaptation if the CPU usage of a grid node becomes less than 10%. During a simulation, the monitoring function is performed by two entities classes: `Sensor` and `Monitor`. The `Sensor` collects data from the simulated grid environment and sends them to the `Monitor`, responsible for data filtering before notifying the `Analyzer` entity of a given situation. The latter is responsible for the analysis and planning functions.

The AGST was implemented in a way that the simulated system knows about itself, its components and interrelationships, their state and behavior (self-awareness). In order to achieve this, AGST extensively uses the resources defined in the Java reflection API. Computational reflection refers to a software ability to provide structures for its own representation in a way that its behavior and the structures that describes it are causally connected. This connection determines the behavior of the system, that is controlled through the manipulation of its representation. Sensors use computational reflection to acquire knowledge concerning the structure of the managed resource (variables, methods, constructors, interfaces, relationships, hierarchy) and to get access to the variables that determine its state, even if their access are defined as protected and thus perform the sensing.

The knowledge necessary for analyzing the monitored data and for deciding the set of adaptation actions that must be performed in a given situation is expressed by rules. Each rule is represented by a `Rule` entity. The `Monitor` notifies the `Analyzer` of changes on the monitored resource or the environment by generating a monitoring event (`MonitoringEvent`). Although the simulation is oriented by discrete events, the simulated monitoring function is time-oriented, ie, the monitor is set with a simulated time interval used for periodically collecting the environment data, searching for the occurrence of significant changes.

3.2.2 Analysis and Planning

The analysis phase occurs after the monitoring and before planning. In the MAPE-K simulation model, the two phases are represented together, since in practice they are commonly implemented in a single place. The analysis and planning process is essential to achieve grid autonomy, since they decide what adaptations will be held in the managed resource. The MAPE-K simulation model provides an entity called `Analyzer` to examine the data collected by the monitors and to determine if changes should be made concerning the current policies or strategies used in the simulated grid environment. For example, if the CPU use of a `GridResource` is greater than 70%, it can be advantageous to migrate its tasks to a less busy `GridResource`, with the aim of minimizing the tasks response time.

The `Analyser` receives the monitored data already filtered out by the `Monitor` and then uses knowledge-based rules to decide whether some kind of adaptation must occur on the grid environment. The analysis and planning phase produces a set of actions (implemented by the `Action` entity), called the action plan (`ActionPlan`). These actions correspond to the reconfiguration operations that, according to the decision-making mechanism, should be implemented in the system in order to achieve a given purpose (self-optimization, self-healing, or self-protection). The action plan is sent to the `Executor`, the entity responsible for actually performing the changes in the simu-

lated system, by sending an `ActionPlanEvent`.

3.2.3 Control and Execution

The control and execution entities allow to simulate the execution of reconfiguration actions identified as necessary by the analysis and planning functions. The purpose of the reconfiguration actions is to allow the simulated grid system to evolve incrementally from one configuration to another at runtime, introducing little impact on the grid execution. In this way, the grid components do not need to be finalized and restarted in order to change them.

The self-configuring feature allows the autonomous system to automatically adjust to changing circumstances perceived on its own functioning and its execution environment. The reconfiguration process in the MAPE-K simulation model is executed by the `Executor` entities, using effectors (`Effector`). As the sensors, effectors also use computational reflection to acquire knowledge concerning the structure of the managed resource, as well as to get access and alter the variables that determine its state. The reconfiguration execution is the final stage of the MAPE-K autonomic lifecycle management, taking as input an `ActionPlan` generated by the analysis and planning process.

The AGST MAPE-K simulation model allows the execution of two adaptation approaches: parametric and compositional. Parametric adaptation concerns the modification of variables that determine the system behavior. For instance, one can adjust the amount of task replicas generated by the a scheduling algorithm. The parametric adaptation implements parameter variability, maintaining a current used strategy. On the other hand, compositional adaptation concerns the exchange of algorithms or structural parts of a managed resource, allowing it to adopt a new strategy in order to respond to a new environmental state. Examples of compositional adaptation include the addition of a new behavior to the system, or replacing parts of its components in order to circumvent eventual failures.

A component called `ExchangeComponent` is provided to support the effectors in the process of adapting a component through the use of computational reflexion. This component is responsible for performing a component or attribute value substitution. The `ExchangeComponent` public methods have the *synchronized* access modifier to ensure that once a process of replacing a component or attribute value is started, it is not interrupted by another, and thus prevent the concurrent execution of two or more adaptation processes of a given component, preserving the component state consistency.

4 RELATED WORK

There are several related work concerning the field of grid computing simulation. The majority of grid simulation tools are discrete event oriented and various of them are based on the GridSim and SimJava simulators, such as Alea (Klusáček et al., 2008), GSSIM (Kurowski et al., 2007), and OGST (Cunha Filho and da Silva e Silva, 2008). Other examples of grid simulators found in the literature are SimGrid (Casanova et al., 2008) and DSiDE (Chtepen et al., 2009).

GridSim is a simulation tool that allows the modeling and simulation of parallel and distributed computing systems entities. GSSIM and Alea are simulators that extend GridSim providing functionalities for the automatic generation of applications, computing resources, and failures in a synthetic form or through the use of trace files. SimGrid is a simulator tool that provides the basic functionality for simulating applications executions over an heterogeneous distributed environment. DSiDE is a simulator that was used in (Chtepen et al., 2009) to evaluate autonomic approaches to fault tolerance concerning grid applications execution. The aim of this work was to investigate the dynamic replacement of fault tolerance strategies according to the grid execution environment context.

Despite the great diversity of works related to grid computing simulation, for the best of our knowledge none of them address the problem of assisting the middleware developer in the design and evaluation of grid autonomic mechanisms by providing the support for the most used autonomic architecture: the MAPE-K loop. This was the main motivation for the AGST development, which has been used by our research group at the Federal University of Maranhão, Brazil to evaluate autonomic self-optimizing and self-healing strategies focused on grid environments.

5 CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

Recent research efforts seek to apply autonomic computing techniques to computer grids, providing more autonomy and reducing the need for human intervention in the maintenance and management of these computing environments, thus creating the concept of an autonomic grid. In spite of the fact that provisioning a greater degree of autonomy to grid middleware currently constitute one of the most active and challenging research theme for grid computing, current simulation tools lack an adequate support to assist developers in designing and evaluating autonomic

mechanisms for these environments.

In this work we described AGST, a new simulator tool for assisting the development and evaluation of autonomic grid mechanisms. The simulator provides, among others, tools for modeling grid resources and their network interconnections, grid applications and their submissions, the occurrence of resource faults, resources local workload, the use of workload and fault traces following the SWF and FTA standards. Nevertheless, AGST major contribution is the definition and implementation of a simulation model based on the MAPE-K autonomic architecture, that can be used to simulate the monitoring, analysis and planning, control and execution functions, allowing the simulation of an autonomic grid.

Our research group is currently using AGST to assist the development and evaluation of self-optimizing and self-healing strategies focused on grid computing environments. The AGST simulations results were consistent with the ones found on the autonomic grid literature. In AGST evaluation meetings, middleware developers reported a good effectiveness of the simulation tool in assisting the design and evaluation of the autonomic mechanisms, highlighting the simplicity of porting the developed code to a real environment, since all MAPE-K components are represented in AGST.

As future work, we intend to perform experiments in a real grid environment using the InteGrade middleware in order to compare the results with AGST simulated ones. We are also investigating the use of aspect oriented programming for simplifying crosscutting aspects of the AGST code.

ACKNOWLEDGEMENTS

This work is supported by the Brazilian Federal Research Agency, CNPq, grant No.478853/2009-2.

REFERENCES

Abraham, L., Murphy, M. A., Fenn, M., and Goasguen, S. (2010). Self-provisioned hybrid clouds. In *Proceeding of the 7th international conference on Autonomic computing*, ICAC '10, pages 161–168, New York, NY, USA. ACM.

Buyya, R. and Murshed, M. (2002). Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and Computation: Practice and Experience*, 14(13):1175–1220.

Casanova, H., Legrand, A., and Quinson, M. (2008). Simgrid: A generic framework for large-scale distributed experiments. In *Proceedings of the Tenth International Conference on Computer Modeling and Simulation*, pages 126–131, Washington, DC, USA. IEEE Computer Society.

Chtepen, M., Claeys, F. H. A., Dhoedt, B., De Turck, F., Demeester, P., and Vanrolleghem, P. A. (2009). Adaptive task checkpointing and replication: Toward efficient fault-tolerant grids. *IEEE Trans. Parallel Distrib. Syst.*, 20:180–190.

Chwif, L. and Medina, A. C. (2007). *Modelagem e Simulação de Eventos Discretos. Teoria & Aplicações. 2nd Ed.* Author's Edition, São Paulo, Brazil. ISBN 8590597822.

Collet, P., Křikava, F., Montagnat, J., Blay-Fornarino, M., and Manset, D. (2010). Issues and scenarios for self-managing grid middleware. In *Proceeding of the 2nd workshop on Grids meets autonomic computing*, GMAC '10, pages 1–10, New York, NY, USA. ACM.

Cunha Filho, G. and da Silva e Silva, F. J. (2008). Ogst: An opportunistic grid simulation tool. In *LAGrid 2008: 2nd International Workshop Latin American Grid*, Campo Grande, Mato Grosso do Sul.

Germain-Renaud, C. and Rana, O. F. (2009). The convergence of clouds , grids , and autonomics. *IEEE Internet Computing*, 13(December):9–9.

Gomes, B. d. T. P., Cunha Filho, G., Campos, I. R., Goncalves, J. F., and da Silva e Silva, F. J. (2010). Scheduling strategies evaluation for opportunistic grids. *Computing Systems, Symposium on*, 0:88–95.

Howell, F. and McNab, R. (1998). simjava: A discrete event simulation library for java. *International Conference on WebBased Modeling and Simulation*, 30:51–56.

Jha, S., Parashar, M., and Rana, O. (2009). Investigating autonomic behaviours in grid-basedcomputational science applications. In *Proceedings of the 6th international conference industry session on Grids meets autonomic computing*, GMAC '09, pages 29–38, New York, NY, USA. ACM.

Kephart, J. O. and Chess, D. M. (2003). The vision of autonomic computing. *Computer*, 36:41–50.

Klusáček, D., Matyska, L., and Rudová, H. (2008). Alea: grid scheduling simulation environment. In *Proceedings of the 7th international conference on Parallel processing and applied mathematics*, PPAM'07, pages 1029–1038, Berlin, Heidelberg. Springer-Verlag.

Kondo, D., Javadi, B., Iosup, A., and Epema, D. (2010). The failure trace archive: Enabling comparative analysis of failures in diverse distributed systems. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, CCGRID '10, pages 398–407, Washington, DC, USA. IEEE Computer Society.

Kurowski, K., Nabrzyski, J., Oleksiak, A., and Weglarz, J. (2007). Grid scheduling simulations with gssim. In *Proceedings of the 13th International Conference on Parallel and Distributed Systems - Volume 02*, pages 1–8, Washington, DC, USA. IEEE Computer Society.