SNMP-Based Monitoring Agents and Heuristic Scheduling for Large-Scale Grids

Edgar Magaña[1,3], Laurent Lefevre[2], Masum Hasan[1], and Joan Serrat[3]

[1] Cisco Systems, Inc.
10 West Tasman Dr, San Jose, CA 95134, USA
{emagana, masum}@cisco.com
[2] Universitat Politècnica de Catalunya
Jordi Girona 1-3, Barcelona, Spain
emagana@nmg.upc.edu, serrat@tsc.upc.edu
[3] INRIA RESO / LIP Laboratory
UMR 5668 (CNRS, ENS Lyon, INRIA, UCB), France
laurent.lefevre@inria.fr

Abstract. This paper presents both, SNMP-based resource monitoring and heuristic resource scheduling systems targeted to manage large-scale Grids. This approach involves two phases: resource monitoring and resource scheduling. Resource monitoring (even discovery) phase is supported by the SNMP-based Balanced Load Monitoring Agents for Resource Scheduling (SBLOMARS). This resource monitoring and discovery approach is different from current distributed monitoring systems in three main areas. Firstly, it reaches a high level of generality by the integration of SNMP technology and thus, it is offering an alternative solution to handle heterogeneous operating platforms. Secondly, it solves the flexibility problem by the implementation of complex dynamic software structures, which are used to monitor from simple personal computers to robust multi-processor systems or clusters with even multiple hard disks and storage partitions. Finally, the scalability problem is covered by the distribution of the monitoring system into a set of sub-monitoring instances which are specific per each kind of computational resource to monitor (processor, memory, software, network and storage). Resource scheduling phase is supported by the Balanced Load Multi-Constrain Resource Scheduler (BLOMERS). This resource scheduler is implemented based on a Genetic Algorithm, as an alternative to solve the inherent NP-hard problem for resource scheduling in large-scale Grids. We show some graphical and textual snapshots of resource availability reports as well as a scheduling scenario in the Grid5000[1] platform. We have obtained a scalable scheduler with an extraordinary load balanced between all nodes participating in the Grid.

Keywords: Genetic Algorithms, Load Balancing, Monitoring Agents, Resource Monitoring, Resource Scheduling.

[1] Experiments in this article were performed on the Grid5000 platform, an initiative from the French Ministry of Research through the ACI GRID incentive action, INRIA, CNRS and RENATER and other contributing partners (https://www.grid5000.fr)

R. Meersman and Z. Tari et al. (Eds.): OTM 2007, Part II, LNCS 4804, pp. 1367–1384, 2007.

1 Introduction

Resource Management in distributed systems is a well-studied problem. There are numerous implementations available for many computing environments and which include batch schedulers, work-flow engines and operating systems. In such systems, the resource manager has complete control of resources, and thus can implement mechanisms and policies needed for effective use of those resources in isolation.

In Grid Computing [2], things are completely different, mainly because resources are heterogeneous, autonomous, and can be distributed on a large scale. Grids are dynamic environments where resource management is taking place in scenarios characterized by different administrative domains with high variability of resource availability and multiple networking issues. Moreover, in the near future, Grid systems are expected to connect large number of heterogeneous resources (desktops, data-bases, clusters, visualization tools, etc.) that are accessible by many users (in the range of millions) and able to execute a large variety of applications. They are becoming considered as large-scale Grids.

Traditional resource management research has provided many models and algorithms to tackle a wide variety of Grid resource management problems. These solutions are individually addressed to any of the three main resource management phases; namely, resource discovery, resource scheduling and resource allocation [4]. But, as the network spans and the number of users increase, the rather simple management methodologies currently used become more and more inadequate. Therefore, improving these techniques is not sufficient. New alternatives should involve better integration and synergism between the before mentioned resource management phases.

On one hand, resource monitoring problem has been solved by means of both, centralized and distributed approaches. The first ones fail when the number of resources increases or when resources maintain certain mobility in the Grid. The second ones are considered better solutions mainly for heterogeneous networks, no matter the level of resource mobility presented. The most important disadvantage is the complexity of their implementation. Thus, a level of fusion is required to improve efficiency and functionality in current approaches.

On the other hand, the efficient computational resource scheduling is recognized as a hard problem that has been tackled for many years by operation research and artificial intelligence researchers. Indeed, it is well known that the generation of optimal job-shop schedules is an NP-hard problem and hence heuristic algorithms are usually employed to find “good”, sub-optimal solutions in affordable time [8]. One of the most popular heuristic methodologies is Genetic Algorithms (GAs) [15]. These are adaptive methods that can be used to solve optimization problems, based on the genetic process of biological organisms. Over many generations, natural populations evolve according to the principles of natural selection and “survival of the fittest”. They are able to evolve solutions to real world problems, if they have been suitably encoded. Unfortunately, these heuristics algorithms require real-time and statistical information regarding resource availability in order to be successful [12].

This paper presents a new resource monitoring and scheduler systems with the capability of efficiently fulfilling multi-constrain service requirements, which are respectively: user requirements (QoS, deadlines, etc.), service necessities (memory, storage and software requirements) and resource-load balancing in the entire Grid. These requirements are expressed in service policies and they are managed by means of a Policy-based Grid Resource Management Architecture (PbGRMA) [6]. The presented approach is based on a full distributed resource monitoring system and a heuristic algorithm for resource scheduling in large-scale Grids. The first one is a SNMP-based Balanced Load Monitoring Agents for Resource Scheduling (SBLOMARS). In this approach, unlike the current monitoring and discovery systems, each targeted computational resource (memory, processor, network, storage, etc.) is monitored by an autonomous monitoring sub-agent, offering a pure decentralized monitoring system. The second one is a Balanced Load Multi-Constrain Resource Scheduler (BLOMERS). This resource scheduler is implemented based on a Genetic Algorithm, as an alternative to solving the inherent NP-hard problem for resource scheduling in large-scale Grids.

The rest of the paper is structured as follows. Section II presents related work in the area of monitoring and scheduling systems for Grid Computing. Section III presents the distributed monitoring agents and their main features to improve scheduling process. Section IV describes the general model for resource selection and its interaction with monitoring agents. Section V presents an evaluation of the monitoring system and a quantitative scheduler evaluation performed on Grid5000 platform. Conclusions and future work are described in Section VI.

2 Related Work

Many heuristic algorithms have been proposed to deal with specific cases of job scheduling but they fail, or behave inefficient, when applied to other problem domains. An important family of these heuristic solutions is based on Genetic Algorithms (GAs), which apply evolutionary strategies to allow for a faster exploration of the search space. GAs have proven to be successful in getting better distribution of the jobs through the entire network [10]. Many researchers have investigated GAs to schedule tasks in homogeneous [13] and heterogeneous [12] multi-processor systems with remarkable success.

In the area of resource monitoring, SNMP agents have been implemented by many researchers in different contexts, each one with its own strengths and weaknesses. One of the most similar approaches to SBLOMARS is GridRM [16]. It consists of a generic monitoring architecture that has been specifically designed for the Grid. It was developed integrating several technologies and standards like Java (applets, servlets and JDBC) and SQL Databases. It also follows several Open Grid Forum (OGF) [2] recommendations for resource management. SBLOMARS could be more competitive due to its lower resource consumption and its greater availability to offer at any moment reliable information regarding availability of computational resources.

Other alternatives are MonAlisa [7] and Ganglia [11], which are well known systems to monitor computational resources and clusters, respectively. On one hand

MonAlisa require complementary systems to be helpful in the Grid management area. On the other hand, Ganglia is not oriented to large-scale Grids. It is oriented to clusters and high capacity resources. NetLogger [17] is both a methodology for analyzing distributed systems, and a set of tools to help implementing the methodology. It provides tools for distributed application performance monitoring and analysis. ReMos [18] aims to allow network-aware applications to obtain relevant information about their execution environment. Condor-G [19] is a task broker designed as a front end to a computational Grid. It acts as an entry point to the grid dispatching jobs to run on the various nodes available. Also worth mentioning is, JAMM (Java Agents for Monitoring and Management) [21] a distributed set of sensors that collect and publish monitoring information of computational resources. GridLab [14] aims to enable applications to fully exploit dynamically changing computational resources.

3 SBLOMARS – Resource Monitoring Agents

Resource monitoring and discovery involves determining which resources are available to be assigned to execute a specific job, application, service, etc. The challenge here is to deal with resources not belonging to a unique centralized administrator. This is because we are assuming realistic scenarios far away from the case where a management system knows a priory the assigned resources to any node on the Grid.

A diagram showing the main classes and interfaces of SBLOMARS monitoring agents is presented in Figure 1. The **(1)PrincipalAgentDeployer**, the main class of the overall system, deploys a specific agent for each kind of resource to be monitored. It offers a generic user interface that can be used to specify the timing between every invocation of the SNMP-MIBs, as well as the number of invocations between every statistical measurement. The **(2)ResourceSubAgents** are instantiated in as many classes as different resources must be tracked (five, so far). The **(3)ResourcesDiscovery** class advises the system on the kinds of resources that are available in the nodes constituting the Grid, and stores that information in the **(8)Network-Map Database**. The **(4)RealTimeReport** generates real-time resource availability information. Finally, the **(5)HistoricalReport** generates statistical resource availability information. This information is presented in two formats: **(6)XML-based documents** containing the historical and statistical reports, and **(7)Dynamic Software Structures** consisting of real-time snapshots. The statistical reports are later used in the resource selection phase to determine, in advance and by means of a heuristic approach, which resources are more likely to be the optimal solution for the fulfillment of any user's request. Since the information thus displayed could appear to be crude or unfriendly to customers, network administrators and resource owners, a graphical interface has been integrated in order to provide user-friendly information on resource availability to any third party on the Grid.

This has no impact on the performance of SBLOMARS due to the fact that the graphical interface solely collects information that is already in the local database. A more detailed description of how the resource scheduler contacts with SBLOMARS is

left for the next section. SBLOMARS overcomes the scalability problem by splitting monitoring activities into independent frameworks (a sub-agent is deployed for every kind of resource) and distributing the monitoring phase through software agents running autonomously (agents do not depend on other systems). SBLOMARS agents have been developed based on SNMP monitoring technology because it is commonly available in any type of platform. This ensures flexibility and applicability of our approach in heterogeneous operating environments [5].

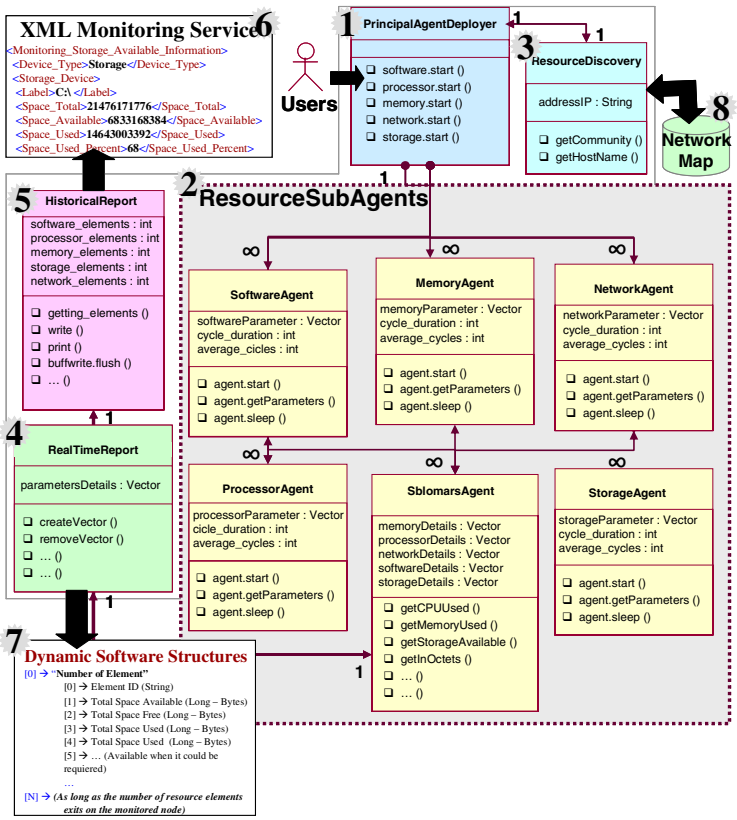

Fig. 1. SBLOMARS Architecture and Interfaces

3.1 Implementation Aspects

The monitoring agents are a set of different types of sub-agents, one per kind of resource to monitor. Our design provides real-time and historical statistical resources availability information. SBLOMARS has two main properties and advantages. Initially, it deploys a monitoring sub-agent per kind of resource to monitor (processor, memory, storage, etc.). This means that nodes forming the Grid, could share just some of their resources and not all of them. Secondly, SBLOMARS is able to monitor any amount of shared resources. This means that no matter how many types of resources

are available, the distributed agents will monitor their behavior. This is possible because SBLOMARS automatically handles its memory buffers to be as long as should be required. Therefore, it could be deployed in a wide range of nodes ranging from simple desktop computers to complex multi-processor servers.

SBLOMARS deploys a single thread per type of resource to be monitored, independently of the amount of such resources. This is worthy to mention because, many monitoring systems fail when they try to handle new "hot-plug resources" that have been added to the system. As we mentioned before, every resource is monitored by independent software threads that start again at certain lapse of time becoming an infinite cycle. The cycle-timing is defined by local or remote administrators through booting parameters at the beginning of its execution.

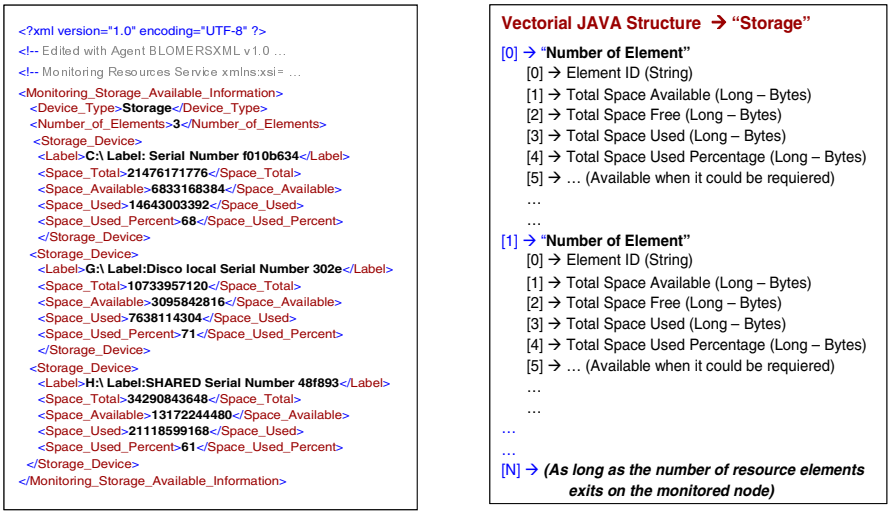

Fig. 2. (a) XML-based Reports and (b) Dynamic Software Structures

3.2 Real Time and Historical Resources Availability Reports

SBLOMARS presents real time and historical statistical resource availability information in two formats which are illustrated in Figure 2. The first one is based on XML standard [3]. These documents show real time resource availability information but SBLOMARS also produce additional XML-based documents with statistical information per resource. This statistical information is important to feed the genetic resource selection algorithm as described in the next section. Both documents are stored in an internal database that makes them accessible provided that the requesting entity has the appropriate access rights.

The second output format to present monitored information is through "Dynamic Software Structures". They are not physical documents like the previous ones. These are software structures developed to keep in memory buffer for each type of resource both, the amount of used and the amount of available resources since the last refresh. The refreshing period is also assigned by the local or remote administrators when

SBLOMARS is bootstrapping. This is another advantage of this approach; the scheduling system could get this information from the memory buffer in a faster way than accessing the XML reports because parsing activity is then avoided and then saving scheduling time.

3.3 Graphical Interface

SBLOMARS offers real-time and historical data by means of sockets' connections to any sub-agents running at any time. This information could appear quite crude or unfriendly for customers, network administrator or resources owners. Therefore, we have integrated a graphical interface to bring user friendly information regarding resource availability to any third party on the Grid. This graphical interface does not impact the performance of the SBLOMARS agents due to the fact that it collects the already available information from the local database. In figure 3, we show a snapshot of this graphical interface. In this example the graph is plotting several nodes from GRID5000 test-bed which are distributed in Nancy, Bordeaux and Lyon (France).

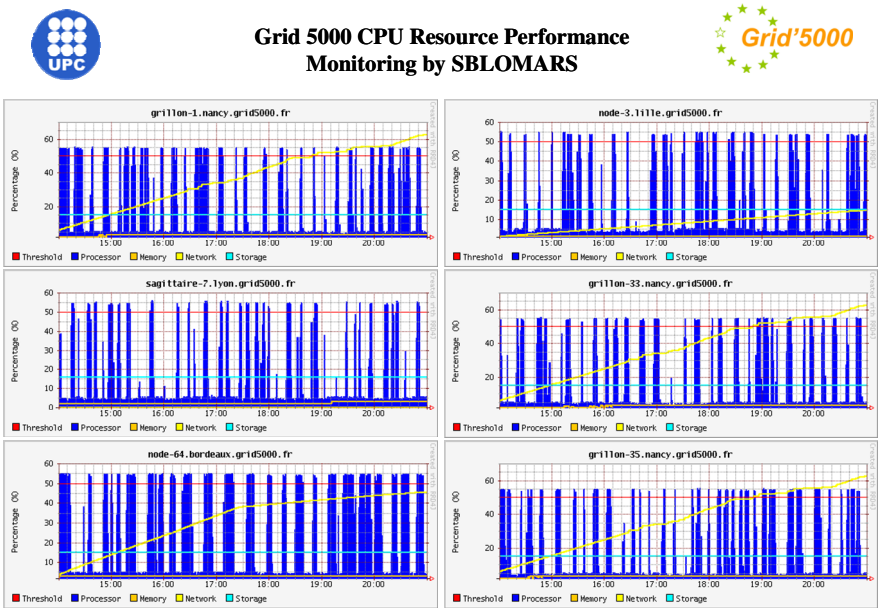

Fig. 3. SBLOMARS Graphical Interface

4 BLOMERS - Resource Scheduler

The second phase in Resource Management Systems is resource selection, which searches and matches job's requirements with resource availability. In other words, it involves determining which resources are the best ones for executing a specific job, application, service, etc. Our approach covers this phase by introducing the Balanced

Load Multi-Constrain Resource Scheduler (BLOMERS). This scheduler makes use of the statistical resource availability information generated by SBLOMARS monitoring agents. In the following sub-sections we will thoroughly explain the motivation and details of this scheduler.

4.1 Motivation of the Heuristic Resource Scheduler

The main goal of a resource scheduling system is the mapping of job requiring resources onto available computational resources in a way that satisfies the users and resource administrator policies. The scheduling problem is represented by a set of independent jobs $J=\{j_1,j_2,...,j_n\}$. Each job has an operation sequence represented by *Ci* (precedent constraints). Each job J_i consists of a set of tasks $T_i=\{t_{i1}, t_{i2},...,t_{ik}\}$ which must be performed between a starting time *(Ts)* and deadline time *(Td)*. The execution of each job requires the use of a set of computational resources $R=\{R_1,R_2,...,R_m\}$ in a local/wide area network. In practice we may assume the network constituted by a set of nodes $N=\{N_1, N_2,...,N_n\}$ sharing their resources (memory, processor, storage, network and software). The objective is to find a schedule with the shortest *makespan*. The *makespan* of a schedule is the time required for all jobs to be processed when no one job could be interrupted during its execution and each node can perform at most one operation at any time.

The scheduling search procedure is at the core of the scheduling methodology. This procedure examines the set of available resources, generates a number of candidates and evaluates the candidate resources to select a final subset to be allocated and communicates the results. The inputs of the search procedure are the set of resources as well as the scheduling policies. The number of candidate subsets *Cr* to be evaluated is given by expression (4.1), given that we have a set of *n* number of available resources and *k* number of possible assignations that fulfill the requested requirements from all the requirements' sources defined in the first section of this paper. To guarantee that the optimum *Cr* will be identified, an exhaustive search over all possible unique resource combinations would be required. However, the cost of such search is prohibitive. For an exhaustive search, all subsets from size one to the size of the entire resources set must be considered in the search.

$$\sum_{k=0}^{n}\binom{n}{k}x^n=(1+x)^n \Rightarrow \sum_{k=1}^{n}\binom{n}{k}=2^n-1 \tag{4.1}$$

4.2 Methodology Proposed and Resource Selection Algorithm

We have highlighted the search problem in large-scale Grids with uncountable number of resources. We have shown that if we have *n* resources, and if a job can be scheduled into any number of resources from *1* to *n*, the total number of possible allocations grows exponentially. Thus, it is computationally very expensive to analyze all possible allocations and solve an optimization problem. In the BLOMERS approach, we propose to find a sub-optimal solution to the problem of scheduling computational resources. This is based on a genetic algorithm, in charge of resource

selection, as a part of the resource manager system, which is embedded into a Policy-based Grid Resource Management Architecture, which has been presented in [6]. In Figure 3 we present the pseudo code of the heuristic resource scheduling algorithm.

Genetic Algorithm for Resource Selection. The genetic algorithm for resource selection has to deal with several conditions [8]. Basically, it should select a set of candidate resources from a poll, keeping individual resource performance comparatively equal in all nodes of the distributed system. This condition has been added in order to satisfy the computational resource load balancing. Finally, the resource selection algorithm needs to keep the relative operations' sequences, known as precedence constrain of the type $i \rightarrow j$. This constrain is defined to mean that data generated by task i are required to start task j.

```
Cleaning Buffer (Bk);
Initialize (k, Pk);
Evaluate (Pk);
Do (Always)
    Select_Resource_Candidates (Pk);
    Recombinate (Pk);
        Crossover (Pk);
        Mutation (Pk);
    Evaluate (Pk);
        Deliver (Solutionk);
    Return;
```

Fig. 4. BLOMERS Genetic Algorithm Pseudo Code

BLOMERS and SBLOMARS Interfaces. BLOMERS uses a collection of solutions (population) from which better solutions are produced using selective breeding and recombination strategies [15]. The first population is created randomly by means of the Initialize (k, P_k) method. Every node forming the Grid is identified by a unique ID. These IDs are stored in a configuration file (network map), which is dynamically updated when new nodes are added or removed from the network. This file is the source of information that BLOMERS uses to know which nodes could be asked about their resource availability. We have explained in Section 2 that every node in the network has a monitoring agent (SBLOMARS) running all the time. The presented genetic algorithm works with several threads in parallel, one for each resource available. SBLOMARS offers a socket connection for resource and for node. Therefore, BLOMERS needs to know which nodes are on line and which ports are been using by each node to reach the resource availability information.

BLOMERS accesses to dynamic software structures through open sockets that have been configured by the monitoring agents. Figure 5 depicts with a triangular-shaped icons the SBLOMARS monitoring agents and with star-shaped icons the BLOMERS resource scheduler. In other resource manager approaches for distributed systems, monitoring activity is controlled by the same instances of the resource manager, making thus the scalability problem a big issue. Nevertheless, in our case we have independent agents to support the growth of network nodes without compromising scalability. BLOMERS resource scheduler is always generating new populations

(solutions) in advance to be assigned when new jobs or applications request resources in the network.

Generation and Selection of Candidate Population. Once the first population has been initialized, a first simple evaluation of this population is done. Normally, the first population is never selected as a candidate solution, but it is the main entry to create new populations. The *Select Resource Candidates* (P_k) method bounds the initial populations and applies two simple genetic operators, such as *Crossover* (P_k) and *Mutation* (P_k). These methods are used to construct new solutions from pieces of old ones in such a way that the population (P_k) steadily improves. This algorithm compares faster versus other heuristic methodologies and could be better adapted to heterogeneous parameters, but the two most important advantages are that it avoids failing into a local minimum solution and that it can be running in parallel to schedule more than request at the same time. We made use of these advantages to design our genetic algorithm with many threads as different types of resources have to be controlled on the Grid. The genetic algorithm needs to be adapted according to the requirements of the application and the environment in which it will be working. The information to analyze for each search will be adaptable to resource availability and the conditions for this adaptation are completely different in each design which assures the novelty of this approach.

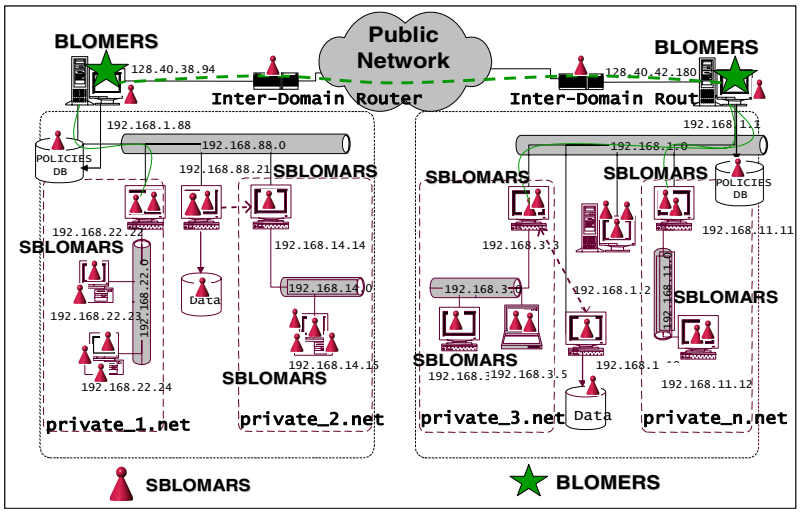

Fig. 5. BLOMERS Scheduler and SBLOMARS Monitoring Agents

5 Overall System Evaluation

We have described SBLOMARS and BLOMERS functionality and their communication workflow. In this section we are going to show the initial results for both systems and to present an ongoing test-bed for the architecture evaluation in a real large-scale Grid [9].

5.1 SBLOMARS Performance Evaluation

We deployed and executed SBLOMARS on Pentium IV system with 512MB of RAM memory and Windows XP operating system. We have analyzed processor and memory consumption impact on the system performance. We used Java Profiler to get the following graphs. In Figure 6(a) we show CPU consumption of SBLOMARS monitoring agents for a period of twenty-four hours. As it can be observed, SBLOMARS represents an insignificant impact in the system behavior. It is clear that only in very small intervals of time (30msec.), as we show in Figure 6(b), system performance could be affected but in general it does not notice any impact.

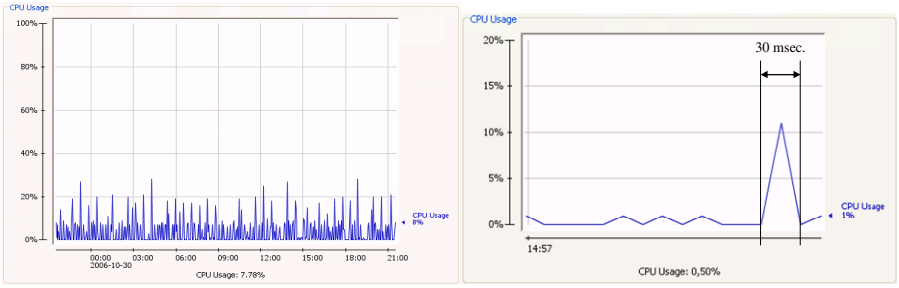

Fig. 6. (a) Twenty-four Hours and (b) Sixty Seconds Processor Overload by SBLOMARS

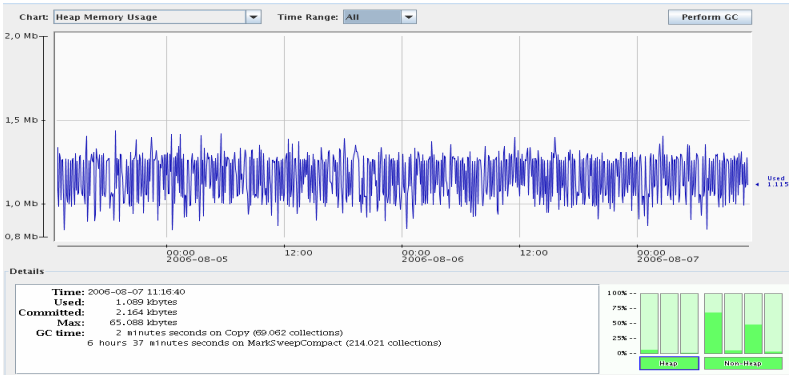

Fig. 7. Forty-eight Hours Memory Overload by SBLOMARS

As far as the memory consumption is concerned, Figure 7 reveals an increase of 0.5MB caused by the SBLOMARS monitoring agents. The important information in this test is that memory consumption remains oscillating below this maximum for whole test duration. This means that monitoring agents do not affect system performance despite their continuous resource performance sensing.

5.2 SBLOMARS Flexibility Evaluation

SBLOMARS reaches a high level of flexibility in two areas: First, it implements dynamic software structures, which have been described during the third section. In order to evaluate the reliability of these structures we have deployed SBLOMARS in a cluster storage server AthlonXP. In Figure 8, we show the total amount of devices available in this cluster and their performance (horizontal bars). We have plugged at 14:00 an external storage device (red darkness area) and as the graphs show SBLOMARS is able to automatically identify this new device and start the corresponding agent to monitor it.

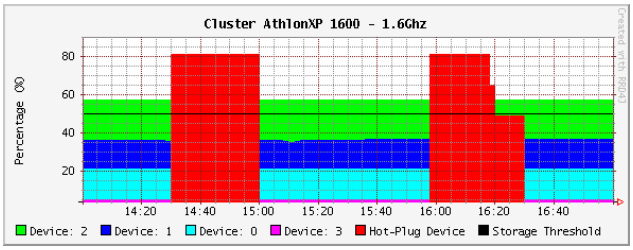

Fig. 8. SBLOMARS Flexibility Evaluation

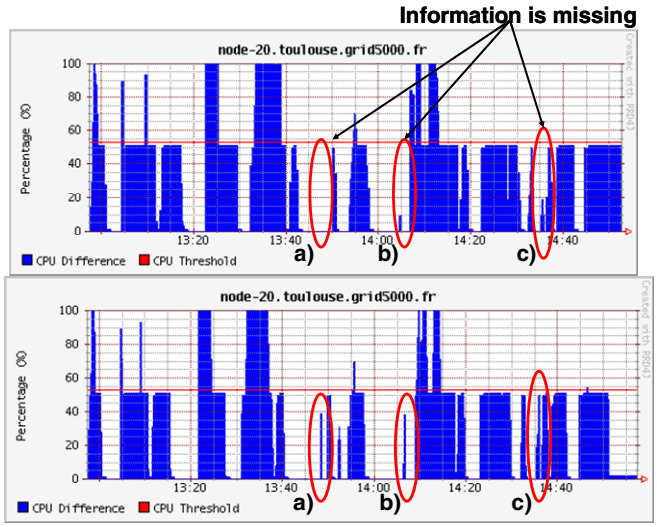

Fig. 9. (a) Fix Timing vs. (b) Auto-configuration in Monitoring Phase

Second, SBLOMARS automatically re-configures their trapping times (calls to SNMP daemon) in an autonomous way. SBLOMARS increases or decreases the interval times between every trap based on the state of the monitored devices. The following graphs show respectively the CPU performance in a Grid5000 node with

fixed times between traps 9(a) and the same node but running with auto-configuration 9(b). It is clear that in certain points the fixed configuration just does not detect certain values. (i.e. between 13:40 and 14:40).

Obviously, the fact of decreasing interval times involves an impact on the performance of the hosting node. In Figure 10 we show the processor used by SBLOMARS. It has started trapping every five seconds. Horizontal lines indicate that our approach has increased the trapping time in: 10, 20, 30, 40, 60 and 120 seconds.

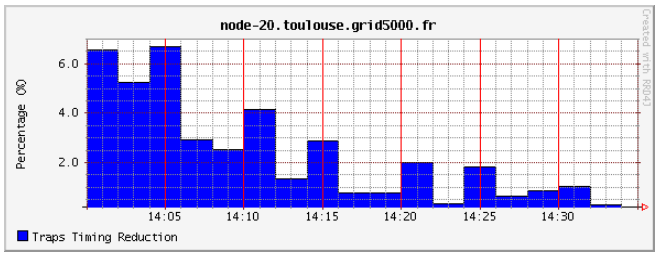

Fig. 10. Auto-configuration Overload

5.3 SBLOMARS Scalability Evaluation

The scalability evaluation of SBLOMARS monitoring agents was performed in the Grid5000 test-bed. We have used 115 nodes with heterogeneous architectures between each others. A random process generator was used to dispatch processes to the Grid Nodes in order to emulate normal "working day" conditions for all the nodes involved, so as to assure results approaching real Grid environments. Once we have our set of nodes ready to run our own experiments, we need to execute three activities to get information from SBLOMARS distributed monitoring system. The first one is the configuration of the monitoring agents. In this activity is when each node will receive the parameters to configure its environment. These parameters are initially trapping times, activation of the flexibility mechanisms and number of traps needed to generate a statistical report.

The following activity is to send the activations command to every node where SBLOMARS has been configured. The last activity is to collect some resource behavior information from different nodes. Therefore, we have also tested how good the scalability is in each one of these activities. In Figure 11(a) we have measured the time required in the Grid5000 test-bed to configure all nodes running SBLOMARS. It is the first activity of the above mentioned. We have started with just five nodes and then we were incrementing the number of nodes in five until the amount of 115 nodes. We were not able to reserve more nodes. It is because other researchers have reserved in advanced more nodes and we just were able to use these ones.

The resulting graph shows that SBLOMARS is incrementing the time in a reasonable way. This steadily increment is due to the network traffic in the test-bed. We can not control the traffic between clusters which are forming the Grid5000. Fortunately, it is not affecting our results as we show in the following graphs (Figure 11(a), Figure 11(b)

and Figure 11(c)). Regarding the activation phase, we have performed the same experiment. The presented graph shows our results. In this phase is clearer the stability that SBLOMARS performs where the number of node to activate is increasing.

Finally, we have also tested the scalability the SBLOMARS when it is offering resource behavior information. This experiment has the same structure that the previous ones. In this case, the time that SBLOMARS consumes to offer specific resource behavior information is much more less. Along this experiment, some values were longer that the expected. It is because network issues between SBLOMARS monitoring agents and the requesting entity. The requesting entity could be a user or administrator who wants to know resource behavior information in certain nodes. In Figure 11(c) is graphed our results. The time to get resource information is really short, is around twenty and thirty milliseconds. This time remains steady regardless the number of nodes in the experiment.

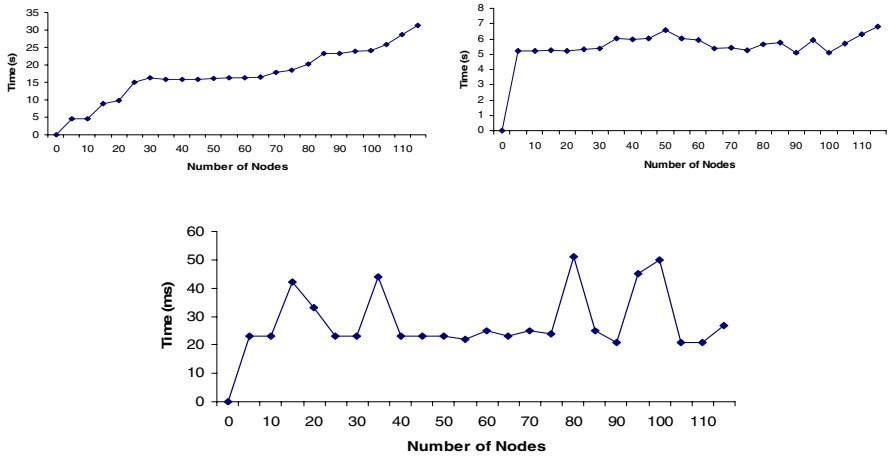

Fig. 11. SBLOMARS (a) Configuration, (b) Starting and (c) Responding

5.4 SBLOMARS Storage Evaluation

In Table 1, we present the time between each trap to MIB-OIDs values, the total amount of files generated and the total amount of disk space used. These results are reporting an interval of time for the first 24 hours due to fact that agents automatically clean memory buffers up after this period. Therefore, there we avoid the possibility to fill the system buffer and storage devices with monitoring reports.

5.5 BLOMERS Analytical Evaluation

Evaluating the BLOMERS approach corresponds to estimating the average *fitness of individuals* (group of available resources) matching a required *schem*a (conditions for

resource load balancing and minimizing makespan). Denote by *n(H, k)* the number of individuals in the population *(P_k)* matching schema *H* at generation *k*. If fitness proportional selection is used, and ignoring the effects of crossover and mutation, the expected number of individuals matching *H* at generation *k + 1* is:

$$E(n(H,k+1)) = \sum_{i \in P(k) \cap H} \frac{f(i)}{\overline{f(k)}} \tag{5.1}$$

Where *(P_k)* $\cap$ *H* denotes the individuals in P_k matching *H, f(i)* denotes the fitness of *i* and *f(k)* denotes the average fitness of the population at time *k*. If we denote by *Dc(H)* and *Dm(H)* the probability that an individual matching *H* at generation *k* will be disrupted by crossover or mutation and not match *H* at generation *k + 1*, and assume crossover and mutation to work independently of each other, a lower bound is representing by:

$$E(...) \geq \frac{u(H,k)}{\overline{f}(k)} h(H,k)(1 - Dc(H))(1 - Dm(H)) \tag{5.2}$$

Here, we are ignoring the beneficial effects of crossover and mutation. The disruption probabilities *Dc(H)* and *Dm(H)* depend on the details of the operators used, but for the classical choice of one-point crossover, *Dc(H)* will increase with the defining length of *H*. Assuming the mutation to mutate the individual bits with equal probability, *Dm(H)* will increase with the order of *H*, the number of all possible solutions in *H*. Equation (5.2) is known as the *schema theorem*. More in-depth discussions of the schema theorem as well as other theoretical approaches to genetic algorithms and evolutionary computation can be found in [15]. The estimation of the *fitness of individuals* is a classical technique to tune *fitness function* on the genetic algorithm. It is corresponding to *Evaluate (P_k)* method in Figure 4.

Table 1. Storage Space Used for The Resource Monitoring Database

Resource	Trapping Time (s)	Total Reports	Space Used (MB)
Processor	10	8640	3,520
Memory	60	1460	0,576
Network	30	2880	1,143
Storage	300	288	0,357
Software	1800	48	0,212

5.6 BLOMERS Performance Evaluation in Grid5000

We have deployed SBLOMARS and BLOMERS on the Grid5000 platform (currently 3000 nodes located in 10 different sites in France) [9]. These nodes are linked through 1 and 10Gbits networks. We evaluate our approach on different Grid scenarios (micro Grid of nodes geographically located on one site, Enterprise small scale Grids with few dozens of nodes located on a reduced number of sites (3) and large scale Grids

with 10 sites and few hundred of nodes). We experiment our tools on different scenario with different time frame experiments.

We believe that a performance comparison is unfair for others resource manager approaches. Besides, it is ponderous to deploy the current systems in order to run the same kind of monitoring and scheduling tests with equal environment in all of them. Therefore, we have decided to compare BLOMERS genetic resource selection algorithm versus trivial resource selection algorithms such as round-robin and least average used. In Figure 12 we have plotted the performance for processor scheduling with these algorithms in certain Grid5000 node. Because of space limitations we can not include all of them. The full set of graphs is available in the following reference [20]. Most of the heuristic approaches solve the selection problem in terms of reduction of the *makespan* for the entire scheduling process. BLOMERS is going a step further because the resource load in all over the Grid is quite better than other algorithms; horizontal lines show the border between every scheduling algorithm and the vertical line the threshold for BLOMERS. The perfect case should be when the number of times that this threshold is crossed tends to zero, then we show than SBLOMERS is closer to this goal than the other two.

We clearly see that SBLOMARS and BLOMERS together are quite competitive because they offer a full distributed resource management system. The main facts to highlight for our approach are: Its ability to handle different resource constrains (resource requirements sources), the wide range of computational resources to monitor, its fully distributed architecture, which allows a high level of scalability and its heuristic implementation in the resource scheduling phase, which increases its ability to minimize the makespan in every service requested.

6 Conclusions and Future Work

Scheduling computational resources in large-scale Grids is a matter which requires significant innovations. This is mainly because their behavior is time-varying, the resource availability is always unpredictable, their performance is highly unstable and the amount of resources to compute is undetermined in most of the cases. Current strategies for scheduling resources fail in fulfilling the demands of a wide variety of distributed applications. The enormous potential of Grids cannot be reached until fundamental development of new powerful scheduling algorithms has taken place.

This paper presents a novel monitoring and scheduling approach addressing critical issues such us flexibility and re-negotiation of user requests and will seek to address the problem of handling uncertainty and imprecision in both computing resources and user requirements. We have presented SBLOMARS, an open source monitoring approach, whose monitored information is used by BLOMERS scheduler, which based on a heuristic algorithm reduce the makespan for scheduling processes and maintain the load balanced in a large-scale Grid. We have tested both, the monitoring agents and the heuristic resource scheduling algorithm in a real scenario, obtaining very promising results. We have shown that the implemented algorithm performs better than a round-robin and least average used selection mechanisms. The novelty

and advantages in our approach are obtained by the synergy of these systems. We have improved machine utilization, resource scheduling time and scalability.

The presented monitoring and scheduler systems would facilitate resource owners the provisioning of facilities for turnaround-assured work. Its presents an advantage in flexibility, due to the fact that it deploys several resource monitoring agents, which work independently from the scheduler and get real-time and statistical resource availability. As future work we are planning to improve security issue. Currently we are working with version two for SNMP server configuration. We have realized that better security mechanisms should be integrated in this research. We are also planning to merge SBLOMARS and BLOMERS approaches with autonomic gateways. We expect that this conception will help Distributed Systems and Grids designers to evaluate and monitor more precisely the usage of their network resources.

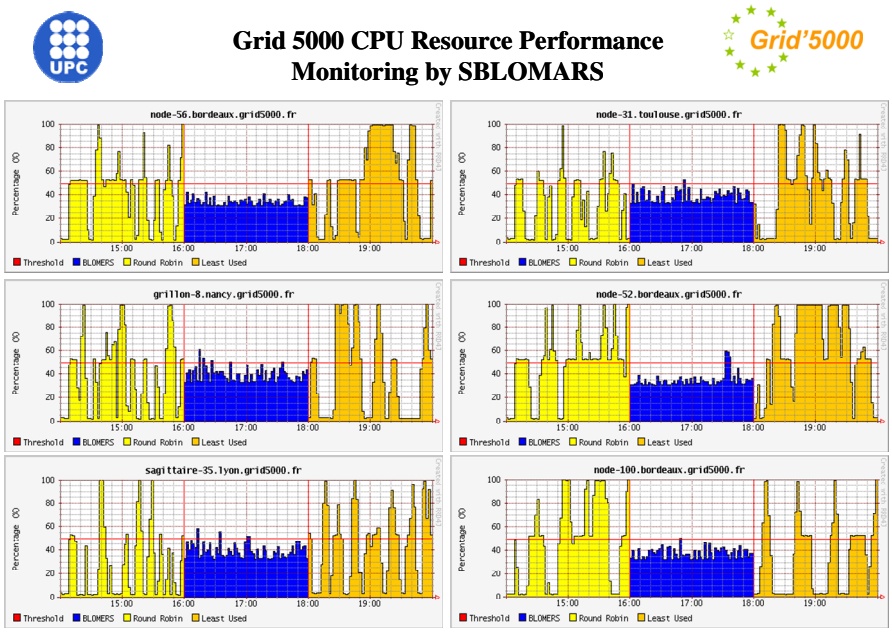

Fig. 12. BLOMERS Versus Other Resource Scheduling Algorithms

Acknowledgments. This paper is supported by the IST-EMANICS Network of Excellence (#26854). It is also supported by the Ministerio de Educación y Ciencia project TSI2005-06413.

References

1. Stallings, W.: Lawrence Berkeley National Laboratory (July 2000), SNMP, SNMPv2, SNMPv3 and RMON 1 and 2 (Third Edition). Addison-Wesley Professional, pp. 365 - 398 (1999) http://www-didc.lbl.gov/JAMM/
2. Open Grid Forum. Web Site: http://www.ogf.org

3. Klie, T., Strauβ, F.: Integrating SNMP agents with xml-based management systems. IEEE Communications 42(7), 76–83 (2004)
4. Nabrzyski, J., Schopf, J.M., Weglarz, J.: Grid Resource Management State of the Art and Future Trends. Kluwer Academic Publishers, Boston, USA (2004)
5. Subramanyan, R., Alonso, J.M., Fortes, J.: A scalable SNMP-based distributed monitoring system for heterogeneous network computing. In: Reich, S., Anderson, K.M. (eds.) Open Hypermedia Systems and Structural Computing. LNCS, vol. 1903, pp. 4–10. Springer, Heidelberg (2000)
6. Magaña, E., Lefevre, L., Serrat, J.: Autonomic Management Architecture for Flexible Grid Services Deployment Based on Policies. In: ARCS 2007, Zurich, Switzerland (2007)
7. Legrand, I., Newman, H., et al.: MonALISA: An Agent based, Dynamic Service System to Monitor, Control and Optimize Grid based Applications. In: CHEP 2004, Interlaken, Switzerland (September 2004)
8. Garrido, A., Salido, M.A., Barber, F.: Heuristic Methods for Solving Job-Shop Scheduling Problems. In: ECAI-2000 Workshop on New Results in Planning, Scheduling and Design, Berlín, pp. 36–43 (2000)
9. Cappello, F., et al.: Grid'5000: A Large Scale, Reconfigurable, Controlable and Monitorable Grid Platform. In: Grid 2005. 6th IEEE/ACM Grid Computing, Seattle, Washington, USA, November 13–14 (2005)
10. Zomaya, A., The, Y.H.: Observations on using genetic algorithms for dynamic load-balancing. IEEE Transactions on Parallel and Distributed Systems 12(9), 899–911 (2001)
11. Massie, M., Chun, B., Culler, D.: The Ganglia Distributed Monitoring System: Design, Implementation and Experience. Parallel Computing 30(7) (July 2004)
12. Page, A., Naughton, T.: Dynamic task scheduling using genetic algorithms for heterogeneous distributed computing. In: 19th IEEE IPDPS 2005, Denver, Colorado, USA (April 3-8, 2005)
13. Ahmad, I., Kwok, Y.K., Dhodhi, M.: Scheduling parallel programs using genetic algorithms. John Wiley and Sons, New York, USA (2001)
14. GridLab. A Grid Application Toolkit and Testbed, www.gridlab.org/
15. Reeves, C.: Modern Heuristic Techniques for Combinatorial Problems. McGraw-Hill Book Company, UK (1995)
16. Baker, M., Smith, G.: GridRM: A Resource Monitoring Architecture for the Grid. In: 3rd International Workshop on Grid Computing, Baltimore, Maryland, USA (November 2002)
17. Tierney, B., Gunter, D.: NetLogger: A Toolkit for Distributed System Performance Tuning and Debugging. LBNL Tech Report LBNL-51276 (2002)
18. DeWitt, A., Gross, T., Lowekamp, B., et al.: ReMoS: A Resource Monitoring System for Network-Aware Applications. Carnegie Mellon School of Computer Science
19. Thain, D., et al.: Distributed Computing in Practice: The Condor Experience. Concurrency and Computation 17(2-4), 323–356 (2005)
20. BLOMERS Performance Web Page: http://nmg.upc.es/ emagana/sblomars/grid5000.html
21. JAMM Project. Java Agents for Monitoring and Management. Lawrence Berkeley National Laboratory (July 2000), http://www-didc.lbl.gov/JAMM/