MaGate Simulator: A Simulation Environment for a Decentralized Grid Scheduler

Ye Huang, Amos Brocco, Michele Courant,
Beat Hirsbrunner, and Pierre Kuonen

Department of Informatics, University of Fribourg, Switzerland
Department of Information and Communication Technologies, University of Applied Sciences Western Switzerland (Fribourg)
{ye.huang,amos.brocco,michele.courant,beat.hirsbrunner}@unifr.ch,
pierre.kuonen@hefr.ch

Abstract. This paper presents a simulator for of a decentralized modular grid scheduler named MaGate. MaGate's design emphasizes scheduler interoperability by providing intelligent scheduling serving the grid community as a whole. Each MaGate scheduler instance is able to deal with dynamic scheduling conditions, with continuously arriving grid jobs. Received jobs are either allocated on local resources, or delegated to other MaGates for remote execution. The proposed MaGate simulator is based on GridSim toolkit and Alea simulator, and abstracts the features and behaviors of complex fundamental grid elements, such as grid jobs, grid resources, and grid users. Simulation of scheduling tasks is supported by a grid network overlay simulator executing distributed ant-based swarm intelligence algorithms to provide services such as group communication and resource discovery. For evaluation, a comparison of behaviors of different collaborative policies among a community of MaGates is provided. Results support the use of the proposed approach as a functional ready grid scheduler simulator.

Keywords: Grid Scheduling, SmartGRID, MaGate Simulator, Simulation.

1 Introduction

Distributed heterogeneous systems under decentralized control are conventionally understood as grid computing [1], pervasive computing [2] or peer-to-peer (P2P) computing [3] systems. Grid nodes are organized as decentralized virtual organizations (VO) with each member sharing its resources with the community. The goal of a grid is thus to construct and manage a powerful shared pool of resource that enables large scale usage and better resource throughput.

Grid scheduling services, also known as 'high level' scheduling [4], are considered as a crucial component for grid computing because they determine the effectiveness and efficiency of the grid. Scheduling services are in charge of identifying, characterizing, discovering, selecting, and allocating the resources best suited for a particular job.

Y. Dou, R. Gruber, and J. Joller (Eds.): APPT 2009, LNCS 5737, pp. 273–287, 2009.

The contribution of this paper is a simulation based implementation of a decentralized modular grid scheduler named MaGate. As a grid scheduler, the MaGate enables the scheduling of a job across a variety of grid resources such as computational clusters, parallel supercomputers, desktop machines that belong to different VOs. More precisely, submitted job may not be executed only on nodes within same VO, but also on appropriate remote resource from other VO with independent scheduling systems and polices. By allocating user's jobs to a proper resource, selected from the entire grid community, improvement of the rate of successfully job execution can be expected. In other words, the MaGate schedulers are designed to cooperated with each other, in order to provide intelligent scheduling for the scope of serving the grid community as a whole, not just for a individual grid nodes.

The MaGate simulator is implemented on GridSim [5] and Alea [6], which together provide the modeling of different kinds of essential grid components, such as grid jobs with various parameters, heterogeneous grid resources, and grid users. Based on this simulated grid ecosystem, each MaGate scheduler receives locally submitted jobs throughout its lifecycle, and matches job requirements with local resource characteristic using the adopted scheduling policies. Jobs suited for local execution are kept, whereas for each unsuited job, a resource search query is propagated to other grid nodes in order to discover remote MaGates accepting remote execution of the job.

This work is implemented within the SmartGRID[7] project, which aims at developing a flexible grid middleware supported by fully decentralized bio-inspired algorithms. Accordingly, in order to support scheduling activities, the MaGate simulator relies on a grid overlay simulator that provides services such as group communication and resource discovery. Due to the requirements of the SmartGRID project, actual implementation only considers fully decentralized peer-to-peer systems where nodes are connected over an unstructured topologies. Communication between the MaGate and overlay simulators is achieved by means of asynchronous message passing: the scheduler simulator can control the grid overlay one by requesting connection of new nodes, disconnection or crash of existing nodes, as well as by starting resource discovery queries. Currently the overlay simulator supports static overlays, as well as dynamic unstructured overlays managed by different algorithms such as BlatAnt [8], Gnutella[9], and Newscast [10].

Both the MaGate and the overlay simulator are designed and developed within the SmartGRID project, which aims at bringing a decisive increase in efficiency, robustness, and reliability regarding the volatile, dynamics, and heterogeneous grid computing infrastructure. The SmartGRID is comprised of two layers and one internal interface: the *Smart Resource Management Layer (SRML)* to support grid scheduling; the *Smart Signaling Layer (SSL)* to provide reactive resource discovery, and the *Datawarehouse Interface (DWI)* to facilitate data exchanging between SRML and SSL. Detailed description of SmartGRID can be found at [7] [11].

The remainder of the paper is organized as follows: in section 2, an overview of related work is introduced. Section 3 and Section 4 details the framework of MaGate simulator and Overlay simulator respectively, followed by a discussion on the experimental results as an illustration of its usage in Section 5. Conclusions and future work are presented in section 6.

2 Related Work

This section provides an overview of related work concerning both the scheduling simulator, as well as the grid overlay simulator.

2.1 Grid Simulator

The management of a real grid system has shown its complexity, which limits researchers' capability to test and explore ideas at the investigating stage. In order to remedy the unnecessary pain at an early phase, a simulation system is quite necessary.

GridSim. GridSim [5] is a toolkit implemented in Java, which allows parallel modeling and simulation of different grid entities, such as distributed grid users, applications, resources, schedulers, and resource brokers. It provides the facility for creating different classes of heterogeneous resources that can be aggregated by resource brokers, and mapped to job requirements.

GridSim supports modeling of uni-processor and/or multi-processors machines with time-shared and/or space-shared scheduling policies. Furthermore GridSim lets users define their own application behaviors, and supports various types of jobs, which are known as *gridlets* and are parameterized by information like MIPS, I/O, etc. A range of protocols enable the *gridlets* to be mapped on different kinds of resources.

Other salient features of GridSim toolkit include: resource time zone, special time slots for resources (weekend, holidays, etc), advance reservation, market-driven economic models, network speed specification, statistic and analyzing of GridSim actions, etc.

Alea. Alea [6] is a simulator based on GridSim, developed for the purpose of dealing with common scheduling problems in grid environment, such as heterogeneous resources and jobs, dynamic job arriving flow, etc.

Alea is a strong addition to GridSim because it brings many important and useful features including an experimental centralized grid scheduler with advanced scheduling techniques for schedule generation, support of jobs requesting single-processor and/or multi-processor, a set of separated profiles for describing job requirement and resource capability, various implemented queue based algorithms (FCS, EDF, Easy Backfilling, EDF-Backfilling), improved simulation determinism, and support of Grid Workload Format (GWF) [12].

GSSIM. GSSIM [13] is another simulation framework based on GridSim, which provides an easy-to-use grid scheduling framework for enabling simulations of a wide range of scheduling algorithms in multi-level heterogeneous grid infrastructures. GSSIM is structured with a set of flexible and replaceable plugin components, such as grid scheduling plugins, local scheduling plugins, and runtime calculation plugins. Moreover by means of a specific developed network manager, GSSIM improves the speed of grid simulations by avoiding the need to packetize large network transfers, and by providing a network-aware scheduling simulation in distributed environment.

2.2 Overlay Simulator

To simulate the underlying network connecting grid nodes, a variety of networks simulators are available [14] [15]. Although these tools provide precise discrete simulation and evaluation of network protocols, they are too low-level for the purpose of evaluating the SmartGRID approach. Accordingly, a more high-level network simulator geared toward peer-to-peer protocols was considered. In this respect, there exist different simulators that allow rapid prototyping and evaluation of peer-to-peer algorithms, both for simple membership management as well as for resource discovery.

PeerSim. PeerSim [16] is a Java simulator that provides a set of classes to ease the implementation of peer-to-peer algorithms. In order to keep the simulation process simple, PeerSim is not concerned with the transport layer. Two simulation models are provided: an event-based and a cycle-based model. As concurrency is not supported, nodes operate in a sequential order.

PlanetSim. PlanetSim [17] provides a simulation environment for overlay networks and services. The platform enables the implementation and evaluation of network services on top of different overlay algorithms, as well as the implementation of new network management protocols. Furthermore, PlanetSim allows seamless deployment on the PlanetLab[1] network for real world experiments.

AntHill. In contrast to the previously described simulators, AntHill focuses on multi-agent systems and distributed bio-inspired ant colony algorithms: software agents can migrate between peers and collaborate to solve complex tasks.

The simulator implemented within SmartGRID borrows ideas from these projects and provides an environment for the evaluation of fully distributed algorithms based both on swarm intelligence, as well as on traditional peer-to-peer protocols.

3 MaGate Simulator

In this section we introduce the MaGate simulator framework. First the design goals are presented, followed by an overview of the framework, and an in-depth discussion of its components.

[1] http://www.planet-lab.org/

3.1 Design Goals

The MaGate simulator is developed with goal of providing a set of easy-to-use simulated decentralized grid schedulers, which are able to interact with each other for job exchanging, collaborate with external grid services and/or simulations, and help researchers to evaluate different scheduling relevant algorithms/models under various using scenarios.

3.2 MaGate Simulator Modules

As the adopted grid scheduler by SmartGRID, the MaGate scheduler is dedicated to tackle different grid scheduling relevant events within an uniform and loosely coupled architecture, including delegating jobs with appropriate remote nodes, using dynamic resource discovery service, open structured for cooperating with external grid components, etc. The MaGate simulator addresses such goals with a modular architecture that corresponds to the real world, as illustrated in Figure 1.

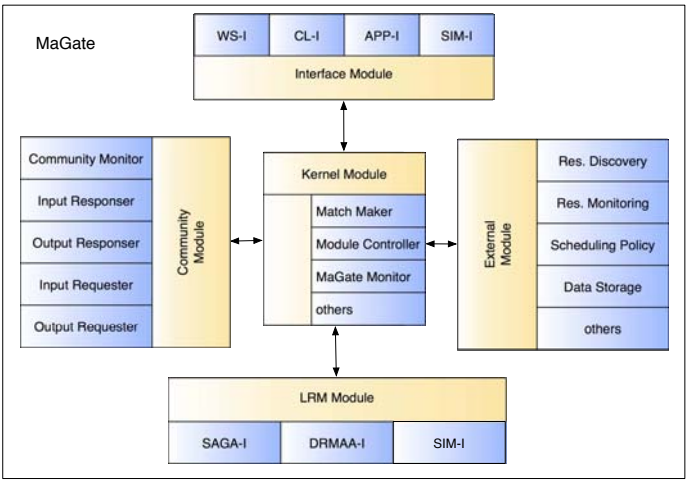

Fig. 1. MaGate Modular Architecture

Kernel Module. The Kernel Module is responsible for MaGate's self-management, which addresses various MaGate internal events, provides local scheduling decisions, and interacts with other modules to make the MaGate work as a whole. Meanwhile, the Kernel Module is also in charge of system logging and analysis.

The *ModuleController* plays an important role because it is in charge of processing the continuously incoming internal simulation events during the scheduler lifecycle, including: job submission/scheduled/completion events, community knowledge updates, system self-inspection requests. In response to these

```
...
Sim_event ev = new Sim_event();
while (Sim_system.running()) {
  super.sim_get_next(ev);
  <routine code for system/community checking>
  if (ev.get_tag() == Message.JobToMatchMaker) {
    <code to process job submission>
    continue;
  } if (ev.get_tag() == Message.ScheduleMadeByMatchMaker) {
    <code to process schedules made due to local scheduling policy>
    continue;
  } ...
  if (ev.get_tag() == GridSimTags.END_OF_SIMULATION) {
    <code to process signal of end of simulation>
    break;
  }
}
<code to finalize the simulation>
```

Fig. 2. *ModuleController* checking the continuously arriving internal events

events, the *ModuleController* determines its future actions. Figure 2 shows how self-management is achieved.

The *MatchMaker* receives jobs transferred from the *ModuleController*, evaluates the adopted policy with knowledge of local resource capabilities, and decides whether the job could be executed locally. If a job can be fulfilled by local resources, the *MatchMaker* allocates the job to the implemented *SIM-I* interface of the LRM Module; otherwise, the *MatchMaker* either looks up appropriate remote nodes from the local cached *direct neighbors list*, or sends the propagated queries to the External Module, in order to discover potential suitable remote nodes from the grid community directly. At a later stage, the *MatchMaker* filters discovered results, and invokes the Community Module to delegate corresponding jobs to the selected remote resources. The *MaGateMonitor* is used to record MaGate behavior and scheduling history for statistical purposes.

Interface Module. The Interface Module manages the interfaces for accepting job submission from multi-type local invokers, such as grid users, high level grid applications, and simulation based instances, and for giving the responses back.

The *CL-I* provides a command line based interface to receive parametrized job submission. Similarly, both the *APP-I* and the *WS-I* offer alternative approaches for receiving job submission from specific grid applications and web service based invokers respectively.

Furthermore, for the purpose of MaGate simulator validation, the *SIM-I* also accepts submission of simulated grid jobs.

Community Module. The Community Module is a vital component of the MaGate scheduler, because it mediates the interaction between different schedulers, and facilitates the work (job) exchange among the interconnected grid community. With help from the Community Module, jobs that cannot be satisfied by local resources are allowed to be delegated for remote execution. In other words, connected schedulers collaborate to construct a dynamic and interoperable grid scheduler community, namely a *Smart Resource Management Layer*. The design of the Community Module follows the suggestion of the Scheduling Instance [18].

As illustrated in Figure 3, the *OutputRequester* firstly checks whether jobs need to be delegated to remote nodes. In that case, the *OutputRequester* is responsible for searching appropriate remote MaGate schedulers based on adopted resource searching policy, and tries job delegation to each discovered remote nodes, until the delegation request is accepted. Inversely, the *InputRequester* is responsible for incoming job delegation requests, has to determine job delegation acceptance depending on the utilized community collaborative policy, and manages transfer of accepted delegated jobs to the Kernel Module for local execution.

After delegated jobs are processed, the *OutputResponser* is used to construct corresponding responses, and deliver them back to the delegation initiators. Similarly, the *InputResponser* monitors the incoming delegation response information from other grid schedulers.

The *CommunityMonitor* keeps a cached *direct neighbors list*, which is established and maintained by the interconnected resource discovery service through

```
... JobInfo fetchedJobInfo = null;
while(!this.maGate.getStorage().isEmpty_localUnsuitedJob()) {
  <code to fetch well presented jobs for delegation>
  boolean status = false;
  if(RDProtocol.equals(Message.RDFromDirectNeighbors)) {
    remoteNodes = this.maGate.getStorage().getNeighborList();
  } else if (RDProtocol.equals(Message.RDFromCommunitySearch)) {
    remoteNodes = searchRemoteMaGate(fetchedJobInfo.getJobProfile());
  } else {
    <code of other resource discovery approaches>
  }
  for(Node rNode : remoteNodes) {
    status = inputRequestToRemoteMaGate(rNode, fetchedJobInfo);
    if(status) { break; }
  }
  <code to process job delegation success/failure result>
}
```

Fig. 3. *OutputRequester* delegate local unsuited jobs

the External Module. Each member of such list mainly contains the information of remote scheduler's published profile, as well as the recent node status. In this case, each MaGate has a partial knowledge of its grid community, enabling fast resource discovery, work exchanging, load balancing and failure recovery of the entire grid.

LRM Module. The LRM Module bridges the interaction between the MaGate and existing grid infrastructure, e.g., local resource management systems. Accepted local suited jobs, from both local users and the grid community, are allocated to the interconnected local resources, and retrieved back once the process is completed. Instead of supporting all of the existing facilities, the LRM Module prefers to make use of emerging standardized API-based specifications, such as *SAGA-I* (Simple API For Grid Application) [19] and *DRMAA-I* (Distributed Resource Management Application API) [20], to support various heterogenous resources.

At current stage, in order to validate the MaGate simulator, implementation of the *SIM-I* is focused on processing simulated jobs on simulated resources.

External Module. The External Module offers a plug-in mechanism, which strengthen each MaGate scheduler instance by integrating available external grid components/services/algorithms, and makes the grid scheduler fit various usage scenarios.

The *ResourceDiscovery* is a critical component because it connects the MaGate scheduler to an existing grid community, which makes the job delegation

```
HashMap<String,Object> maGateProfile = new HashMap<String,Object>();
maGateProfile.put("os", this.maGate.getLRM().getOsType());
<code to put other resource characteristic>
this.maGate.setMaGateProfile(maGateProfile);
this.maGate.getMaGateInfra().updateProfile(maGateId, maGateProfile);
```

Fig. 4. Resource community profile publish

```
getMaGateInfrastructure().startQuery(this.maGateId, queryId, queryProfile);
this.results.put(queryId, queryProfile);
Thread.sleep(MaGateParam.timeAllowedForCommunitySearch);
SearchResult result = this.results.get(queryId);
this.results.remove(queryId);
```

Fig. 5. Resource discovery from grid community

on remote nodes to be possible. As shown in Figure 4, once a MaGate scheduler applies to join an existing grid community, its community available capability information has to be published in the meantime. Afterward, once the *ResourceDiscovery* is invoked to discover remote resources with expected characteristic, the published community profile of each arrived remote scheduler will be used for mapping and resource selection (shown in Figure 5). The *ResourceMonitoring* component works in a similar way as the *ResourceDiscovery* component, by monitoring changes in the published community profile of already contacted remote MaGate schedulers.

Besides that, the *SchedulingPolicy* offers a parameter based approach for adopting external scheduling algorithms, which follow the uniform I/O parameter schema and may be developed by other organizations. Finally, the *DataStorage* component enables the storage of MaGate's data into external storage facilities.

4 Overlay Simulator

The overlay simulator provides both membership management for grid nodes, as well as resource discovery and group communication services. The topology maintained within the simulator is fully controllable by the MaGate simulator: nodes can be connected to the network, disconnected, or forced to crash (i.e. disconnect abruptly from the network).

Although the implementation focuses on BlatAnt [8] as the main overlay management algorithm, several other are also available (for example, Gnutella[9] and Newscast [10]). BlatAnt constructs and maintains a self-structured overlay using a collaborative approach inspired by the behavior of ant colonies. The overlay is resilient to node failures, and exhibits low path distances between nodes, as well as a small number of connections between nodes.

Communication between the overlay simulator and MaGates is based on an asynchronous message passing protocol: this ensures independence between the simulators, and permits them to be executed on different computers. Each MaGate interact with a corresponding overlay node to publish its resource profile (used to match resource discovery queries), request new connections to other nodes or disconnect from a node. The interface with the MaGate simulator only exposes these high-level services, and is not tied to the actual algorithm used to manage the overlay.

Resource discovery is currently achieved using a restricted flooding algorithm. When a MaGate issues a resource discovery query the overlay node propagates the query to all of its neighbors. Each receiving node will forward the query up to a determined distance. Forwarding is stopped when a node with a profile matching the query is found: in this case a notification is sent to the requesting MaGate for each matching node found. As queries are tagged with a unique identifier, nodes will not forward queries that have already been received in the past.

5 Case Study

As an example of usage of the MaGate simulator, this section presents a reference experiment made by using the simulator, and the adopted community collaborative policies. In Subsection 5.1, the internal interaction workflow of the MaGate simulator is given. In Subsection 5.2 and 5.3, the adopted community collaborative policies are discussed as reference for future work. The configuration of the experiment is illustrated in Subsection 5.4, followed by the results discussion in Subsection 5.5.

5.1 Interaction Scenarios

Once a new MaGate scheduler instance is established within the simulation, an external resource discovery service must be interconnected for future community collaboration; meanwhile, a profile with regards to the MaGate's community capability contribution has to be published.

Afterward, the newly established MaGate receives job submission from its local users, and decides whether the job requirement could be satisfied by the local resources. If yes, the job is accepted and allocated to the local resource management system for local execution; if not, the MaGate tries to discover an appropriate remote node which matches the job requirement, and delegates the local unsuited job for remote execution.

Once a MaGate scheduler instance receives a job delegation request from the grid community, acceptance decision is made according to the adopted community policies. If such a request is acceptable, the delegated job will be preserved locally until the process is completed; if not, the reject response is delivered back to the request initiator, with optional reject reasons. Then it is the responsibility of the request initiator to decide whether another re-negoation process should be issued later, depending its the adopted community policy again.

Noteworthy that once a delegated remote job is accepted by the local MaGate instance, there is no difference between jobs submitted locally, and jobs delegated from the grid community.

5.2 Resource Discovery Policies

To delegate local unsuited jobs to appropriate remote resources, such resources have to be discovered firstly. Concerning the ecosystem of MaGate simulator, we address the problem of decentralized resource discovery by using flooding based protocols on a self-structured overlay topology maintained with the help of a bioinspired algorithm that borrows ideas from the swarm intelligence and ant colony optimization.

Two alternative approaches are evaluated in the reference experiment, and illustrated as follows:

Neighbors look-up policy. The *Neighbors* means that each MaGate scheduler instance is supposed to discover remote MaGates from a local cached *direct*

neighbors list, which is maintained and kept up-to-date by the adopted overlay resource discovery service, regarding to the network connection status with the local MaGate.

Community search policy. The *Search* stands for each MaGate scheduler instance is responsible for propagating resource discovery queries according to the job requirement, submitting such queries to the grid community, and obtaining the return results after a certain period of waiting time. For example, the *Search100* presents the waiting time between query submission and result obtaining is 100 milliseconds.

5.3 Job Delegation (Re)Negotiation and Acceptance Rules

Many rules can be applied to determine whether a job delegation request should be accepted, as well as the subsequent behaviors. During experiments two simple rules to be the reference benchmark for the future work has been used.

Job delegation (re)negotiation rule. The *Nego* defines the maximal allowed number of times of negotiation for each individual job delegation. For example, the *Nego1* stands implies that if a job delegation is rejected by a remote MaGate, the same request should not be resent to the same remote MaGate anymore; inversely, the *Nego10* implies that a rejected delegation request is allowed to be retried with the same remote MaGate for ten times, with same or different parameters.

Job delegation acceptance rule. The *Queue* stands for the length limit of the *Community Input Queue*. Each time the host MaGate approves a job delegation request, the accepted but unprocessed remote job will be preserved in the *Community Input Queue* until the job is processed and sent back to the delegation initiator. In our experiment, for example, the *Queue5* presents that the host MaGate is able to manage at most five accepted but unprocessed remote jobs, as long as the length limit is reached, the subsequent delegation requests to the host MaGate will be rejected.

5.4 Simulation Configuration

The experiment were performed on an Intel Core Duo 2.2GHz physical machine, with 2GB RAM. In order to obtain stable values, the results were averaged from 10 repeated iterations. The experiment is done on a grid with 100 MaGates, each MaGate manages a Massive Parallel Processor System (MPP) with 64 or 128 processors. Each MaGate is supposed to receive 100 jobs submitted from the local user during 12 hours, each job may require 1 to 5 processors. the choices of operating system owned by all MPPs fall into the same distribution as the job requirement: [Linux, Windows, Mac]; similarly, the processors MIPS owned by all MPPs is configured as same as the job requirement.

Additionally, size of the *direct neighbors list* of each MaGate is 6, the number of times allowed for (re)negotiation is either 1 or 3, and length of the *Community Input Queue* is either 5 or 10.

5.5 Results and Discussion

The benefit of large scale grid computing has been verified by many researchers [21] [22]. In our work, a new criterion titled *RJC (Rate of successfully executed Jobs from the entire grid Community)* is proposed to demonstrate the functionalities of the MaGate simulator. The reference experiment aim at increasing the value of *RJC*, as it represents the effectiveness of allowing local unsuited jobs to be shared amongst different grid schedulers, from the grid's point of view. Other criterions such as resource throughput and network workload are not yet considered.

The behaviors of different scenarios in a 100-MaGate community illustrated in Figure 6.

The *Local* represents a reference scenario where no jobs sharing between MaGate scheduler instances is allowed. If a locally submitted job cannot be fulfilled by the local resource, it is considered as a *local unsuited job* and marked as failure. Considering that each MaGate manages one MPP machine with a single operating system, and the submitted jobs vary their operating system requirements from an uniform three-option distribution, in average each MaGate could only process 1/3 locally submitted jobs on its local resource. Conversely, considering that the choices of operating system owned by all MPPs within the grid community fall into the same distribution as job requirements, it is expected that for each individual local unsuited job an average of 1/3 of the MaGates of the entire grid community has the expected capabilities to accept them.

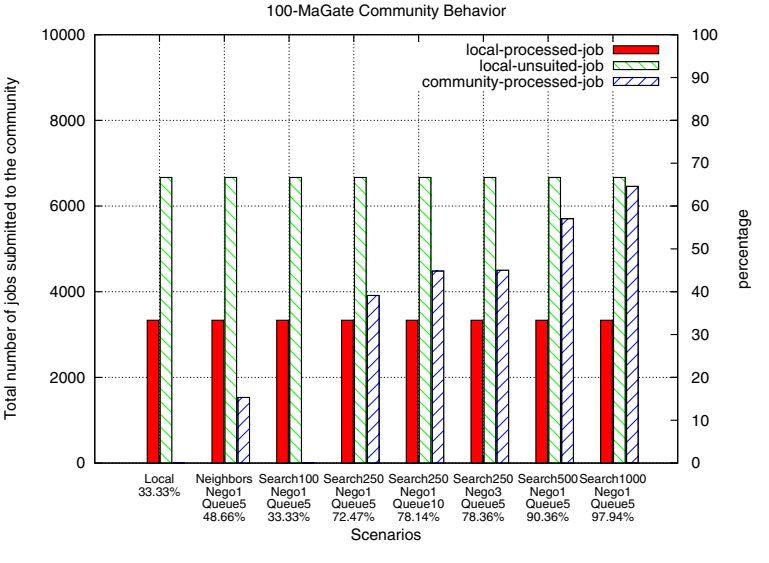

Fig. 6. Grid community of 100 MaGates

Next, as presented by scenario *Neighbor-Nego1-Queue5*, if the *Neighbors Look-up Policy* is adopted as the resource discovery approach, it is evident that useful remote MaGates can be discovered sometimes when needed by job delegation requests. In this case, the *RJC* has been improved by approximate 15%.

An alternative way of seeking remote MaGates for job delegation is the *Community Search Policy*. It is to be expected that if appropriate remote MaGates exist, are connected within the same community, are represented properly, and are pproved to be publicly available by their community policies, the corresponding resource discovery queries will be matched within a reasonable time delay. However, as shown in scenario *Search100-Nego1-Queue5*, if the waiting time is too limited, for example 100 milliseconds, no remote MaGates can be discovered, which makes no difference in the obtained *RJC*.

If the waiting time allowed for *Community Search Policy* is increased a little, such as 250 mili-seconds by scenario *Search250-Nego1-Queue5*, and 500 milliseconds by scenario *Search500-Nego1-Queue5*, discovery becomes more successful, and the *RJC* is improved by 39.14% and 57.03% respectively.

Additionally, results illustrated in scenario *Search250-Nego1-Queue10* and *Search250-Nego3-Queue5* demonstrate that even within the same waiting time for the *Community Search Policy*, the *RJC* can be still improved by utilizing various job delegation related rules, such as increased times for (re)negotiation and expanded length limit of the *Community Input Queue*.

Finally, it is noteworthy that allowing enough waiting time for resource discovery using the *Community Search Policy*, as shown by scenario *Search1000-Nego1-Queue5*, is a necessary condition to achieve an *RJC* of 100%. Nonetheless, failure to obtain results might still be possible either because of limits of the restricted flooding algorithm used by the Overlay simulator, or because candidate remote MaGates that may already reached their length limit of the *Community Input Queue* and have not been released during the delegation waiting period.

6 Conclusion and Future Work

In this paper, we presented the MaGate simulator as a grid simulation environment. The MaGate simulator is composed of different modules, and aims at providing a simulation based implementation for the MaGate scheduler, an interoperable decentralized grid scheduler used within the SmartGRID project [7], and dedicates to cooperate with each other to provide intelligent scheduling for the scope of serving the grid community as a whole, not just for a single grid node. Moreover the simulator itself can be easily extended, and adopted for evaluating newly developed decentralized scheduling algorithms, models, or workflows. An overlay simulator is employed by the MaGate simulator, to provide services such as group communication and resource discovery on fully decentralized peer-to-peer network.

As an example, two resource discovery polices, along with another two job delegation (re)negotiation and acceptance rules have been used as reference

scenarios, and the validated results have shown the use of a functionally ready grid scheduler simulator.

Future work will focus on the the extension of the simulator with an advanced Community Module supporting web services technology (especially the WS-Agreement specification [23]), as well as better support to other existing local scheduling algorithms. Furthermore, a study of an automatic mechanism to dynamically generate user customized community collaborative policies will be carried out, in order to evaluate all the different parameters that can be used to generate various community collaborative policies. This work our may bring increased flexibility and adaptability in grid scheduling.

Acknowledgements

MaGate simulator is developed within SmartGRID project, a collaborative work led by PAI group[2] from University of Fribourg, and GridGroup[3] fromUniversity of Applied Sciences Western Switzerland (Fribourg). This work is supported by the Swiss Hasler Foundation[4], in the framework of the ManCom Initiative (ManCom for Managing Complexity of Information and Communication Systems), project Nr. 2122.

References

1. Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of High Performance Computing Applications 15(3), 200 (2001)
2. Satyanarayanan, M.: Pervasive computing: vision and challenges. IEEE Personal Communications, [see also IEEE Wireless Communications] 8(4), 10–17 (2001)
3. Milojicic, D., Kalogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard, B., Rollins, S., Xu, Z.: Peer-to-Peer Computing. HP Laboratories Palo Alto (March 2002)
4. Schwiegelshohn, U., Yahyapour, R.: Attributes for communication between scheduling instances. Global Grid Forum, GGF (December 2001)
5. Buyya, R., Murshed, M.: GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing. Concurrency and Computation: Practice and Experience 14(13-15), 1175–1220 (2002)
6. Klusacek, D., Matyska, L., Rudova, H.: Alea-Grid Scheduling Simulation Environment. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Wasniewski, J. (eds.) PPAM 2007. LNCS, vol. 4967, pp. 1029–1038. Springer, Heidelberg (2008)
7. Huang, Y., Brocco, A., Kuonen, P., Courant, M., Hirsbrunner, B.: SmartGRID: A Fully Decentralized Grid Scheduling Framework Supported by Swarm Intelligence. In: Seventh International Conference on Grid and Cooperative Computing, 2008. GCC 2008, China, pp. 160–168. IEEE Computer Society, Los Alamitos (2008)

[2] http://diuf.unifr.ch/pai/
[3] http://gridgroup.tic.hefr.ch/
[4] http://www.haslerstiftung.ch/

8. Brocco, A., Frapolli, F., Hirsbrunner, B.: Bounded diameter overlay construction: A self organized approach. In: IEEE Swarm Intelligence Symposium, SIS 2009. IEEE, Los Alamitos (2009)
9. Ripeanu, M., Foster, I.: Peer-to-peer architecture case study: Gnutella network. In: First Conference on Peer-to-peer Computing, Sweden, pp. 99–100. IEEE Computer Press, Los Alamitos (2001)
10. Jelasity, M., van Steen, M.: Large-scale newscast computing on the internet. Technical Report IR-503, Vrije Universiteit Amsterdam, Department of Computer Science, Amsterdam, The Netherlands (October 2002)
11. Brocco, A., Hirsbrunner, B., Courant, M.: A modular middleware for high-level dynamic network management. In: Proceedings of the 1st workshop on Middleware-application interaction: in conjunction with Euro-Sys 2007, pp. 21–24. ACM Press, New York (2007)
12. GridWorkloadsArchive: `http://gwa.ewi.tudelft.nl/pmwiki/`
13. Kurowski, K., Nabrzyski, J., Oleksiak, A., Weglarz, J.: Grid scheduling simulations with GSSIM. In: 3rd Workshop on Scheduling and Resource Management for Parallel and Distributed Systems, Proceedings of the 13th International Conference on Parallel and Distributed Systems, Hsinchu, Taiwan (2007)
14. Henderson, T., Lacage, M., Riley, G.: Network simulations with the ns-3 simulator. Demo paper at ACM SIGCOMM 2008 (2008)
15. Mathieu Lacage, T.R.H.: Yet another network simulator. In: WNS2 2006: Proceeding from the 2006 workshop on ns-2: the IP network simulator, p. 12. ACM, New York (2006)
16. Jelasity, M., Montresor, A., Jesi, G.P., Voulgaris, S.: The Peersim simulator, `http://peersim.sf.net`
17. García, P., Pairot, C., Mondéjar, R., Pujol, J., Tejedor, H., Rallo, R.: Planetsim: A new overlay network simulation framework. Software Engineering and Middleware, 123–136 (2005)
18. Tonellotto, N., Wieder, P., Yahyapour, R.: A proposal for a generic grid scheduling architecture. In: Integrated Research in Grid Computing Workshop, Greece, pp. 337–346. Springer, Heidelberg (2005)
19. Goodale, T., Jha, S., Kaiser, H., Kielmann, T., Kleijer, P., von Laszewski, G., Lee, C., Merzky, A., Rajic, H., Shalf, J.: SAGA: A Simple API for Grid Applications. High-level application programming on the Grid. Computational Methods in Science and Technology 12(1), 7–20 (2006)
20. Troger, P., Rajic, H., Haas, A., Domagalski, P.: Standardization of an API for Distributed Resource Management Systems. In: CCGRID 2007: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid, pp. 619–626. IEEE Computer Society, Washington (2007)
21. Ernemann, C., Hamscher, V., Yahyapour, R.: Benefits of global grid computing for job scheduling. In: Fifth IEEE/ACM International Workshop on Grid Computing, Pittsburgh, USA, pp. 374–379. IEEE Press, Los Alamitos (2004)
22. Ernemann, C., Hamscher, V., Schwiegelshohn, U., Yahyapour, R., Streit, A.: On Advantages of Grid Computing for Parallel Job Scheduling. In: 2nd IEEE International Symposium on Cluster Computing and the Grid (CC-GRID 2002), Berlin, Germany, pp. 39–46. IEEE Press, Los Alamitos (2002)
23. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web Services Agreement Specification (WS-Agreement). Technical report, Open Grid Forum, USA (2004)