MaGate: an interoperable, decentralized and modular high-level grid scheduler

*Ye Huang, Amos Brocco, Michele Courant and Beat Hirsbrunner
Department of Informatics
University of Fribourg, Switzerland

Pierre Kuonen
Department of Information and Communication
University of Applied Sciences of Western Switzerland

This work presents the design and architecture of a decentralized grid scheduler named MaGate, which is developed within the SmartGRID project and focuses on grid scheduler interoperation. The MaGate scheduler is modular structured, and emphasizes the functionality, procedure and policy of delegating local unsuited jobs to appropriate remote MaGates within the same grid system. To avoid just another isolated solution, web services and several existing and emerging grid standards are adopted, as well as a series of interfaces to both publish MaGate capabilities and integrate functionalities from external grid components. Meanwhile, a specific swarm intelligence solution is employed as a critical complementary service for MaGate, to maintain an optimized peer-to-peer overlay that supports efficient resource discovery.
Regarding evaluation, the effectiveness brought by job sharing within a physically connected grid community with the use of the MaGate has been illustrated by means of experiments on communities of different scale, and under various scenarios.

keywords: Grid Computing, Scheduling, Meta-scheduling, SmartGRID, MaGate scheduler

Introduction

The grid scheduling service, also known as superscheduling (Schopf, 2003), is defined as "*scheduling job across grid resources such as computational clusters, parallel supercomputers, desktop machines that belong to different administrative domains*". It is a crucial component for grid computing infrastructures because it determines the effectiveness and efficiency of a grid system by identifying, characterizing, discovering, selecting, and allocating the resources that are best suited for a particular job.

Grid scheduling is a critical but complex task. The heterogeneous and distributed nature of grid systems imposes additional constraints on scheduling services, such as lack of remote resource control, or incomplete overall knowledge of the grid system.

Besides the theoretical issues, the realities of grid scheduler design and implementation have made things even more complicated. Existing grid schedulers typically depend on (or are completely integrated in) some particular grid middleware. Therefore, it is a non-trivial task to migrate a grid scheduler from one middleware to another, or to exchange messages between schedulers, or to delegate jobs between different types of scheduler. Grid schedulers designed upon various middlewares respectively can be regarded as a set of *heterogeneous grid schedulers*.

The contribution of this paper is the design of a decentralized modular high-level grid scheduler named MaGate. The MaGate scheduler dedicates to improve the rate of successfully executed jobs submitted to the same grid community, by means of interacting with each other and delegating jobs amongst all participating nodes of the community. In other words, the MaGate schedulers are driven to cooperated 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 individually.

To achieve the purpose mentioned above, the MaGate scheduler emphasizes on several relevant issues: (i) the approach of discovering remote resources dynamically and efficiently; (ii) the community policy of determining jobs to delegate remotely, and acceptation of arrived remote jobs; (iii) the platform independent communication protocol to facilitate the interaction between different MaGate schedulers on heterogenous nodes; (iv) the negotiation procedure to tackle various job delegation scenarios flexibly, i.e. job delegation accept/reject/conditional reject, job delegation proxy and forwarding, etc.

The MaGate is being developed within the SmartGRID project (Huang, Brocco, Kuonen, Courant, & Hirsbrunner, 2008), which aims at improving the efficiency of existing grids through a modular, layered architecture: the *Smart Resource Management Layer (SRML)* to support grid scheduling, and the *Smart Signaling Layer (SSL)* to provide resource discovery. Furthermore, communication between layers is mediated by means of the *Datawarehouse Interface (DWI)*.

MaGate is developed within SmartGRID project, a collaborative work led by PAI group from University of Fribourg, and GridGroup from University of Applied Sciences of Western Switzerland. This work is supported by the Swiss Hasler Foundation, in the framework of the ManCom Initiative (ManCom for Managing Complexity of Information and Communication Systems), project Nr. 2122.
Ye Huang could be contacted at email:ye.huang@unifr.ch)

The Smart Resource Management Layer (SRML) is comprised of a set of MaGates. Each MaGate is composed of a set of loosely coupled modules, in order to tackle several critical issues raised by grid scheduling, such as:

- **Standard-compliant interaction between different grid schedulers.** In order to guarantee the interoperability, extensibility and reusability of MaGates, all input and output communication protocols and data formats are designed to be based on existing and emerging standards, especially for job representation, resource modeling, resource capabilities advertisement, and negotiation agreement management.
- **Dynamic resource discovery.** It is fundamentally important to be able to efficiently discover resources in a dynamic network. Our work tackles this issue by using a self-structured peer-to-peer overlay network, constructed and maintained using ant colony algorithms, whose intrinsic design, adaptiveness and robustness provide an optimal platform for resource discovery and monitoring mechanisms.
- **Infrastructure independent job allocation and management.** Infrastructure independency is a non-trivial issue, and the main difficulty lies on the semantics. To overcome such a problem requires either to find a common denominator to hide the infrastructure differences, or to develop separate adaptors for each diverse infrastructure respectively. In order to minimize the work related with this issue, and to provide interoperability and reusability, MaGate relies on the unified interfaces provided by standardized specifications, to achieve infrastructure independent job allocation and management.
- **Platform independent interface to external grid services.** Presently, the grid community has realized the importance of standardizing grid solutions, and developed many relevant specifications and libraries, to facilitate grid development with web services technologies. MaGate follows this philosophy, and provides a series of web services based interfaces both to obtain external functionalities from other grid services, and to advertise its own reusable capabilities to external users also.

The remainder of the paper is organized as follows: related work on grid schedulers, the grid standards and resource discovery is introduced in next section. Derivation and purpose of the MaGate is illustrated in section *SmartGRID*, followed by a detailed MaGate description in section *MaGate Architecture*. Section *Reference Experimental Results* discusses the experiment configuration and corresponding results. Finally, section *Conclusions and future work* presents some insights to future development.

Related Work

Grid Schedulers

Considering the important role of grid scheduling, many approaches on this topic have been proposed. Between the most known works, the Meta-Scheduling Service (MSS) (Waldrich, Wieder, & Ziegler, 2006) is a middleware-independent grid scheduler designed with pre-defined policies and currently implemented on the Unicore USite architecture. The GridWay (Huedo, Montero, & Llorente, 2005) is also a well known high-level scheduler from the Globus Toolkit (Foster & Kesselman, 1997) that provides abundant features, such as adaptive scheduling and adaptive execution, within a modular structure. In order to avoid scheduling self-competition, GridWay only allows one scheduler to manage each virtual organization. Additionally, other grid scheduling solutions, such as Moab Grid Suite (*Moab Grid Suite*, 2009) and Community Scheduler Framework (CSF) (Xiaohui, Zhaohui, Shutao, Chang, & Huizhen, 2006), have been developed in collaboration with the industry.

Besides the existing implementations, general scheduler structures, such as the Scheduling Instance (Tonellotto, Wieder, & Yahyapour, 2005), have also been proposed to give a design cornerstone for future grid schedulers.

Current grid schedulers are set up to bridge the gap between grid applications and various pre-existing local resource management systems. Combined with a general lack of grid infrastructure information, two constraints have emerged regarding grid schedulers: (a) the scope of grid system is assumed to be known a-priori, (b) no horizontal interaction between grid schedulers is considered.

To overcome the dilemmas mentioned above, with respect to existing grid scheduling systems, MaGate is designed to be a decentralized grid schedule that emphasizes on grid scheduler interoperation, and complemented by a dynamic resource discovery approach on decentralized network. In order to share the jobs submitted from a local MaGate to other MaGates within the same grid community, a set of community scheduling relevant parameters are evaluated and discussed to address various job delegation scenarios between different MaGates.

Existing and Emerging Standard Specifications

The experiences of the grid community so far have shown that the realization of an ideally single interconnected, interoperating computation ecosystem is difficult. Instead, many different grids for specific usage scenarios have appeared. In order to achieve the promised unified computation environment and being widely adopted by the e-science and industry community, standardized technologies in many fields are being developed, and some of them have established their importance through time.

In particular, the Job Submission Description Language (JSDL) (Anjomshoaa et al., 2005) is known as a XML-based language specifically for describing computational grid jobs submission and their resource requirement. WS-Agreement (Andrieux et al., 2004) works as a platform independent protocol for advertising capabilities of services, and making agreement between service providers and consumers.

Meanwhile, both Simple API For Grid Application (SAGA) (Goodale et al., 2006) and Distributed Resource Management Application API (DRMAA) (Troger, Rajic, Haas, & Domagalski, 2007) dedicate to provide API specification to cover the functionalities of submitting, controlling, and monitoring jobs on local resource management systems.

The aforementioned specifications facilitate either the in-

teroperation amongst grid components, or the interaction between grid components and local resource management systems. All such scenarios are critical for MaGate.

Resource Discovery in Distributed Systems

Resource discovery mechanisms are a vital fundamental part of grid computing, not only because they affect the efficiency of discovering appropriate resources for job execution, but also because their architecture influence the logical topology of grid resources. Concerning the ecosystem of MaGate, resources distributed on a decentralized peer-to-peer (P2P) based network have to discovered dynamically.

At present, proposals of discovering resources on P2P topology have gained significant momentum and being generally categorized into structured and unstructured systems. Structured systems, such as Distributed Hash Tables (DHTs), offer deterministic query search results within logarithmic bounds on network complexity, which means that a look-up operation will be successful within a predefined time bound. Unstructured systems (Ripeanu & Foster, 2001) don't put any constraints on the structure of network and data distribution. Normally, a query flooding protocol is adopted to process look-up requests: this might have series drawbacks, such as high communication overhead and non scalability.

In an attempt to remedy the issues of unstructured overlay, self-structured solutions have been proposed. In contrast to structured approaches, self-structured systems reorganize existing unstructured topologies by adding and removing logical links between nodes (Ripeanu, Iamnitchi, Foster, & Rogers, 2007; Shen, 2004; Schmid & Wattenhofer, 2007). Our work addresses the problem of decentralized resource discovery by using a self-structured overlay topology maintained with help of a bioinspired algorithm that borrows ideas from the swarm intelligence and ant colony optimization. Swarm intelligence (Bonabeau, Dorigo, & Theraulaz, 1999) is a branch of artificial intelligence that focuses on algorithms inspired by the collaborative behavior of swarms of insects. Such bioinspired solutions have already been successfully applied to several network routing problems (Schoonderwoerd, Holland, Bruten, & Rothkrantz, 1997; Di Caro & Dorigo, 1998), as well as for resource discovery in unstructured networks (Michlmayr, 2006). More generally, swarm algorithms exhibit an inherent decentralized design, which helps their implementation in fully distributed systems.

SmartGRID

SmartGRID is a generic and modular framework that supports intelligent and interoperable grid resource management using swarm intelligence algorithms. SmartGRID is structured as a loosely coupled architecture, which is comprised of two layers and one internal interface, as shown in Figure 1.

Smart Resource Management Layer (SRML). SRML is responsible for grid level dynamic scheduling and interoperation serving grid applications with dynamically discovered

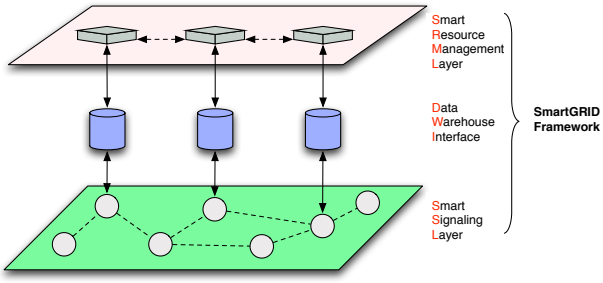

Figure 1. SmartGRID architecture overview

computing resources. SRML is composed of all the engaged *MaGates* schedulers. Each participating MaGate is expected to delegate jobs that are submitted through local MaGate but can not fit the local resource, titled *local unsuited jobs*, to discovered remote MaGates; inversely, each MaGate is also expected to accept job delegation requests from other MaGates within the same community, if the delegation requirements match local MaGate's community policy and current working status. A detailed description of MaGate is presented in section *MaGate Architecture*.

The Smart Signaling Layer (SSL). SSL represents the interface from and to the network of the SmartGRID framework, and provides information about the availability of resources on other nodes, as well as their status. The SSL hides the complexity and instability of the underlying network by offering reliable services based on distributed ant algorithms. Ants are defined as lightweight mobile agents traveling across the network, collecting information on each visited node. A middleware named *Solenopsis* (Brocco, Hirsbrunner, & Courant, 2007) is developed to run each ant node, providing an environment for the execution of ant colony algorithms, specially the BlåtAnt collaborative ant algorithm (Brocco, Frapolli, & Hirsbrunner, 2008, 2009).

The Data Warehouse Interface (DWI). The DWI acts as a loosely coupled communication channel, which is used to mediate the data exchange between SRML and SSL without exposing technical implementation details of either layer. The DWI is comprised of a series of distributed data storages that store both persistent and cached grid information concerning network infrastructure, resource status, grid scheduling request/response, strategy parameters, SmartGRID specific events, etc.

MaGate Architecture

As mentioned before, the MaGate scheduler dedicates to improve the rate of successfully executed jobs submitted to the scope of entire grid community, by means of interacting with each other and delegating jobs amongst all participating MaGates of the same community, using various community policies. Besides, a set of other relevant issues are also targeted, such as utilizing dynamic resource discovery service

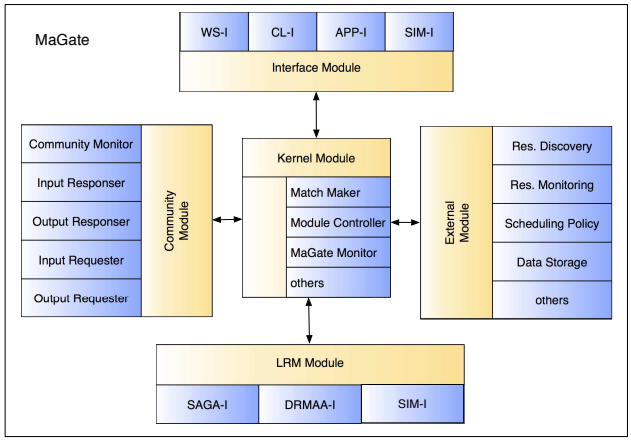

Figure 2. MaGate modular architecture

and open structured for cooperating with external grid components. In order to address different purposes within an uniform and loosely coupled architecture, the MaGate is modular designed, as illustrated in Figure 2: (a) the *Kernel Module* is responsible for MaGate self-management and addressing of internal events; (b) the *Community Module* tackles the interoperation with external schedulers; (c) the *LRM Module* performs job allocation and management on Local Resource Management systems (LRM); (d) the *External Module* interacts with external grid services for additional functionalities; (e) the *Interface Module* manages the interface for accepting job submission from various local invokers, and delivers the results back.

A detailed description of each individual module of MaGate is given in the following subsections.

Modules

Kernel Module. The Kernel Module is responsible for MaGate self-management, which disposes internal events, provides local scheduling decisions, and connects other modules to work as a whole. Additionally, the Kernel Module is also in charge of local behavior logging and analysis.

The *Module Controller* accepts jobs from the Interface Module and the Community Module, validates job format, and transfers the retrieved job requirements to the *MatchMaker*. The *MatchMaker* checks local resource capabilities and evaluates its policy to decide whether the job could be executed locally. If the job can be fulfilled by local resources, the *MatchMaker* allocates the job to an interface instance of the LRM Module; otherwise, the *MatchMaker* propagates a discovery query to the External Module, filters out unsuitable returned results, and invokes the Community Module to execute job delegation to proper remote resources. The *MaGate Monitor* is used to record MaGate behavior and scheduling history for statistic purposes.

Interface Module. The Interface Module is responsible for accepting job submissions from MaGate local invokers, including grid users and high level grid applications. MaGate's local available functionalities are also published through the Interface Module.

The *CL-I* provides a command line based interface for the interacting with the MaGate, which accepts parameter based job local submissions, and delivers the results back. Besides, both *APP-I* and *WS-I* offer the alternative approaches to grid applications and web services based invokers respectively.

Meanwhile, in order to validate the MaGate prototype within a simulation environment, the *SIM-I* is provided to accept submission of simulated grid jobs.

Community Module. The Community Module is a mandatory component of MaGate. It acts as a connector that both accepts jobs from remote MaGates for local execution, and delegates local unsuited jobs to other MaGates for remote execution. With the help of the Community Module, physically connected MaGates can collaborate to construct a dynamic and interoperable grid scheduler community, namely the *Smart Resource Management Layer*. The design of the Community Module follows the suggestion of the Scheduling Instance (Tonellotto et al., 2005).

The *Output Requester* prepares local unsuited jobs and contacts discovered remote MaGates for job delegation. Inversely, the *Input Requester* monitors job delegation requests from other MaGates, validates delegation requirements and local MaGate's community acceptance policy, and transfers the accepted delegated jobs to the Kernel Module for local execution.

Once delegated remote jobs are approved and executed results are collected, the *Output Responser* is used to construct corresponding responses and send them back to the delegation initiators. Inversely, the *Input Responser* is used to monitor the incoming delegation response messages from other MaGates.

The *Community Monitor* maintains a known neighborhood list, and periodically contacts each remote MaGate from the list to obtain a replica of their node status, including node workload, node neighborhood list, etc. Furthermore, more remote MaGates could be proactively discovered and complemented by the External Module. In this case, each MaGate has a partial view of the entire grid scheduler community, titled the *MaGate Community*, which helps to achieve exchanging of work, load balancing and failure recovery within the scope of this known community.

LRM Module. The LRM Module empowers MaGate to utilize grid infrastructure, such as local resources management systems, to allocate the accepted jobs for local execution, monitor the execution status, and retrieve results back. As mentioned before, instead of direct support to too many existing facilities, the LRM Module provides several interfaces to support local resource management systems through standardized API-based specifications, such as *SAGA-I* (Goodale et al., 2006) for "Simple API For Grid Application (SAGA)", and *DRMAA-I* (Troger et al., 2007) for "Distributed Resource Management Application API (DRMAA)".

Meanwhile, in order to validate MaGate simulation based prototype, the *SIM-I*, which simulates resource management systems, is also provided to execute accepted simulated jobs.

External Module. The External Module offers a plug-in mechanism for MaGate. It works as a multi-functional outlet that helps to strengthen the MaGate by integrating appropriate external grid services, components, algorithms and strategies. Since developing grid services using web services has gained significant momentum recently, the service interfaces exposed by the External Module are web services compliant.

The *Resource Discovery* connects MaGate to an external grid resource discovery service for obtaining information of remote resources. It is a critical component for MaGate to validate the idea of scheduler community. The *Resource Monitoring* empowers MaGate to monitor the change of remote resource status. The *Scheduling Policy* offers a parameter based approach for adopting external scheduling algorithms, which follow the uniform I/O parameter schema and developed by other organizations. The *Data Storage* facilitates MaGate to preserve its data into external storage facilities.

Reference Scenario

To make the MaGate scheduler fulfill the purpose of serving grid community as a whole, each newly established MaGate must be connected to a resource discovery service, which is able to discovery remote MaGates from an existing community. Meanwhile, each MaGate of the community is required to publish their public capability profile using a specific key-value format, which is supposed to be discovered and monitored by resource discovery services from other MaGates during the lifecycle. Additionally, if an individual MaGate wishes to contribute its local resources, the LRM Module must be utilized to mediate the communication between the MaGate and the local resource management system.

The Interface Module receives job submissions from the exposed interfaces, and transfers the validated jobs to the Kernel Module. Scheduling algorithms are launched by the Kernel Module to evaluate the job requirements. If the local resource could satisfy the job requirements, the jobs are transfered to the LRM Module for local execution; if not, the Community Module is invoked, to either looks up appropriate remote resources from its local cached neighbors list, or propagates resource searching queries based on job requirements, and transfers such queries to the interconnected external resource discovery services through the External Module. At a later stage, the discovered information regarding remote resources is used by the Community Module to initialize job delegation requests respectively. As soon as one delegation request is accepted by a remote MaGate, the Community Module delegates the corresponding job, leaving a callback address for getting results back. If all delegation requests of an individual job have been rejected, it is then the responsibility of the Community Module to decide whether a re-negotiation iteration should be issued later, with modified delegation parameters. Such decisions are made regarding the utilized community policies by different MaGates.

Inversely, once the Community Module of a MaGate has received job delegation requests from other remote MaGates, acceptance decisions are also made depending on the adopted community policies, such as the length limit of MaGate's *Community Input Queue*, which is used to preserve the accepted but unprocessed delegated remote jobs.

Noteworthy that the Kernel Module is able to addresses the job requests both from local users, and from other connected remote MaGates.

Reference Implementation

The current reference implementation of MaGate is simulation based (Huang, Brocco, Courant, Hirsbrunner, & Kuonen, 2009). The implemented MaGate simulator is based on GridSim (Buyya & Murshed, 2002) and Alea (Klusacek, Matyska, & Rudova, 2008).

For both the Interface Module and the LRM Module, the *SIM-I* interfaces have been implemented to allocate simulated jobs to simulated resources. The interaction between the MaGate and existing local schedulers/middlewares is not mentioned at current stage. Regarding the job allocation on local resource has been out of the main interest of the MaGate, external grid components, such as SAGA (Goodale et al., 2006) and DRMAA (Troger et al., 2007), will be evaluated to facilitate this work in our future implementation.

For the External Module, according to the ecosystem of MaGate, Smart Signaling Layer (SSL) is adopted as the default external service for both *Resource Discovery* and *Resource Monitoring*.

For the Community Module, a socket based implementation has shown that the scheduler interaction is functionally ready. Meanwhile, different job delegation related factors, such as resource discovery policies and community scheduling policies (e.g. delegation negotiation/re-negotiation policy, delegation acceptance policy), have been evaluated respectively. Regarding the future work, a web services based communication service is being developed to achieve the infrastructure independent scheduler interoperation; additional, an algorithm to integrate diverse community policies and automate the negotiation/re-negotation procedure is under the development.

For the Kernel Module, the *Module Controller* is implemented to dispose MaGate internal events, and interact with simulation environment. The *MaGate Monitor* is used to record event history from simulated infrastructure and produce logged data for statistic analysis. A benchmarked "First Come First Service" algorithm is adopted by the *Match Maker* for making local scheduling decision. Meanwhile, algorithm "Easy Backfilling", which is used for comparison purpose, is under the development.

The reference implementation is used to do the reference experiment, which is discussed as follows:

Reference Experimental Results

The reference experiment is evaluated to prove a functional ready MaGate prototype, which is able to address different scheduling related events using an uniform modular architecture. Specially, the capability of scheduler interoperation and work sharing within the interconnected community is emphasized to facilitate the improvement of our criterion: the *Rate of successfully executed Jobs from the entire grid Community (RJC)*. The *RJC* is adopted as the judgement of experiment results to prove the functional effectiveness brought by the design of job delegation on an interoperable MaGate community. We try to maximize this value because it presents the capability and effectiveness of disposing *local unsuited jobs* on remote nodes from the scope of grid community. The disadvantages of using the *RJC* as the only criterion are that both the overall resource throughput and the network load of transferring resource discovery requests are missing currently, which will be considered and measured in our future work.

Reference Models

Although grid systems vary widely depending on the usage scenarios, one of the typical example of a computational grid is still the execution of computational intensive batch jobs on collaborative computers. Several models retrieved from this scenario are used in our experiment, and represented as follows:

Machine Model. The machine refers to the Massive Parallel Processor Systems (MPP), which are comprised of several Process Elements (PE) connected via fast interconnections. Each process element is a single processing system with local CPU and memory, using space-sharing policy and running jobs exclusively. All process elements of the same MPP share the same operating system.

Site Model. The site stands for the grid participators who contribute their computational resources and share their jobs in a grid system. Each site is comprised of several machines, has its own resource management system and local policies. The resources between different sites are heterogeneous. Cluster(s) of a single affiliation is a typical site.

Node Model. The node is a group of sites, managed by a single MaGate scheduler. The grid community is comprised of different nodes, each participating node has the possibility to discover another node, and interacts with each other.

Job Model. The job concerns computationally intensive batch jobs submitted by users continuously through time. Each job is comprised of several parameters, including requested run time, requested number of PE, requested type of operating system, etc. Both sequential and parallel jobs are simulated for execution upon a single machine with sufficient number of PEs, job migration and preemption are not supported currently.

Experimental Scenarios

Once a MaGate with local resource exists and being connected to grid community, it is assumed to be discoverable by resource discovery services from other MaGates. The interesting thing is that various parameters can be utilized to generate community scheduling policies with different cost and benefit. Considering each individual user might have his/her own judgement on the cost and benefit, an automatic mechanism that is capable of generating user customized community scheduling policies dynamically will bring great flexibility and adaptability in our future work.

In current experiment, various scenario parameters have been demonstrated as follows, and utilized to compose diverse policies for determining job delegation across grid communities with different size.

- The *Local* means that each MaGate is configured to work alone, no job delegation to remote MaGates is allowed. In this case, all the local unsuited jobs submitted on each MaGate will be simply suppressed and considered as failed.
- The *Neighbor* means that each MaGate is allowed to look up appropriate remote MaGates from its direct neighborhood list, and delegate the local unsuited jobs to the discovered remote MaGates. The direct neighborhood list of each MaGate is kept up-to-date by its resource discovery service, depending on the network connection between the local MaGate and the remote MaGate.
- The *Search* means that each MaGate is able to propagate and submit job requirement based queries to the grid community, in order to discover appropriate remote MaGates for accepting the local unsuited jobs. The interval time between the query submission and result obtention plays an important role because it represents user's endurable delay to get the discovered results back. In our experiment, for example, the *Search100* illustrates once a query has been submitted to the grid community, 100 milliseconds are allowed to wait and get the discovered results back.
- The *Nego* represents the maximal number of negotiation allowed to achieve a single job delegation. For example, the *Nego1* means that each to-delegate job is allowed to be negotiated with a set of appropriate remote MaGates for one time; while the *Nego10* means that the host MaGate is able to retry a single job delegation for maximally ten times, with same or different parameters.
- 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.

Simulation Configuration

Both job and machine parameters are either constants, or randomly generated according to a uniform distribution.

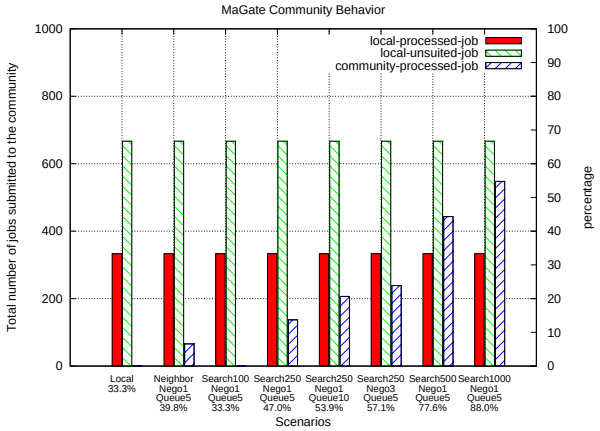

Figure 3. Community of 10 MaGates

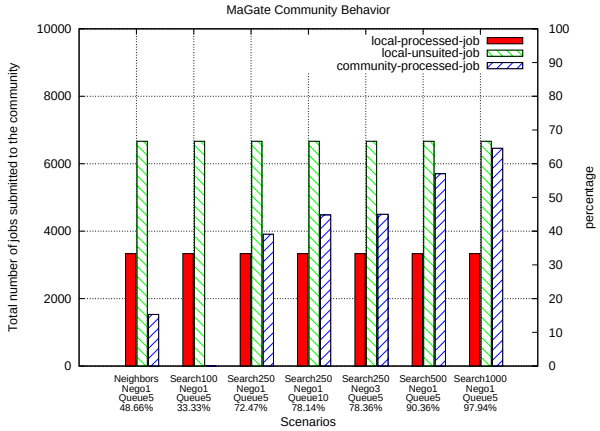

Figure 4. Community of 100 MaGates

- Number of locally submitted jobs on each MaGate: 100.
- Job arrival time: [0-12 hours].
- Job estimated execution time: 1000s.
- Job estimated MIPS: 1000.
- Number of PEs required by each individual job: [1-5].
- Number of sites per MaGate: 1.
- Number of machines per site: 1.
- Number of PEs per machine: [64-128].
- MIPS of each PE: 1000.
- Size of the *Direct Neighborhood List*: 6.
- Number of times for negotiation/re-negotiation: [1, 3].
- Length of the *Community Input Queue*: [5, 10].
- Types of operating system required by job: [Linux, Windows, Mac].
- Types of machine operating system: [Linux, Windows, Mac].

In order to obtain stable values, each scenario results were averaged from 10 repeated iterations. The experiments are performed upon an Intel Core Duo 2.2GHz machine, with 2GB RAM.

Discussion

All the scenarios tested in the reference experiment are comprised of several parameters mentioned above. For example, the scenario *Search250-Nego3-Queue5-57.1%* stands for that the host MaGate is using community search policy, with 250 millisecond interval waiting time, to discover remote MaGates for job delegation; the maximal times of negotiation allowed for each single delegation is three, remote MaGates's length limit of the *Community Input Queue* is five, and the obtained *RJC* has reached 57.1%.

The *RJC* of a 10-MaGate community is shown in Figure 3. As expected, for scenario *Local*, because no job delegation to community is allowed, all submitted local unsuited jobs are suppressed till end of the simulation. Regarding each MaGate manages one site with a single operating system, and the jobs submitted by its local users vary their operating system

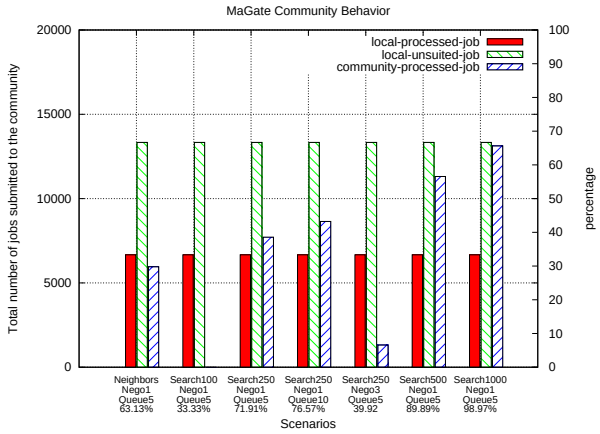

Figure 5. Community of 200 MaGates

requirements from an uniform three-option distribution, only 1/3 of the locally submitted jobs can be satisfied by the local resource. Simultaneously, considering the choices of operating system, which are owned by all sites within the grid community, fall into the same distribution as job requirement, it is expected that for each individual local unsuited job, 1/3 MaGates of the entire grid community have, in average, the expected capabilities to accept them.

Furthermore, illustrated by scenario *Neighbor-Nego1-Queue5* from the 10-MaGate community, involved MaGates are allowed to delegate local unsuited jobs to the grid community by looking up proper remote MaGates from their direct neighborhood list, which are constructed and kept up-to-date by the interconnected resource discovery services. It is evident that, from the point of view of the grid community, some MaGates have found appropriate neighbors, and additional 6.55% jobs were accomplished due to the achieved delegations.

An alternative approach of seeking remote MaGates for job delegation is sending resource discovery queries to the

grid community. It is to be expected that if appropriate remote MaGates exist, being connected within the same community, represented properly and approved to be public available by their community policies, the corresponding queries will be matched within enough interval waiting time. As presented in Figure 3, a short waiting time (*Search100-Nego1-Queue5* allows 100ms) for getting the results from the community search leads to no delegations achieved because the adopted resource discovery solution could not find expected resources from the community within such a limited duration. However, if more time is allowed, as shown by scenario *Search250-Nego1-Queue5* and *Search500-Nego1-Queue5*, useful discovered remote MaGates start to appear, and the *RJC* benefited from job remote delegation can be improved by 13.7% and 44.3% respectively. The more interval time between query submission and result obtention is allowed, the better *RJC* becomes.

Meanwhile, results illustrated by scenario *Search250-Nego1-Queue10* and *Search250-Nego3-Queue5* has demonstrated that even within the same interval waiting time, the *RJC* benefited from job remote delegation can be still improved by utilizing different community cooperative policies, such as 23.8% by increased times for re-negotiation, and 20.6% by expanded length limit of the *Community Input Queue*. It is also noteworthy that enough interval time allowed for community search, such as scenario *Search1000-Nego1-Queue5*, is the necessary but insufficient condition to make *RJC* reach the 100%, because candidate remote MaGates may already reached their length limit of the *Community Input Queue*, and not released during the delegation waiting period.

Besides the results mentioned in a 10-MaGate community, for horizontal comparison purpose, Figure 4 and Figure 5 have shown different behaviors of the same scenarios in communities of different size, namely the community with 100 MaGates, and the community with 200 MaGates.

The *RJC* of scenario *Neighbors-Nego1-Queue5* in the 100-MaGate community (48.66%) and 200-MaGate community (63.13%) has shown that the successful probability of getting appropriate remote MaGates from the direct neighborhood list improves within a grid community of larger size, because more remote MaGates with good connection with the local MaGate can be discovered and considered as direct neighbors. Similarly, although still no remote MaGate could be found within a limited community search duration, scenario *Search250-** have demonstrated that more appropriate remote MaGates can be discovered using the same increased interval time. For example, the *RJC* of scenario *Search250-Nego1-Queue5* in a 100-MaGate community is 25.47% improved compared to his behavior in a 10-MaGate community.

However, the improvement gained by search in a larger community with more interval time is not simply incremental, because the overload distributed resource discovery service has limited the number of discovered remote MaGates. Specially, as shown in scenario *Search250-Nego3-Queue5* in a 200-MaGate community, the *RJC* benefited from job remote delegation has dropped to 6.62% because too many resource discovery queries almost halt the adopted ant-based resource discovery service. In this case, how to balance the cooperation between different factors to achieve an effectiveness and efficient job delegation procedure, will be an interesting issue in our future work.

Conclusion and Future Work

This paper presented the design of an interoperable, decentralized and modular high-level grid scheduler named *MaGate*. The MaGate scheduler dedicates to improve the rate of successfully executed jobs submitted to the same grid community, by means of interacting with each other and delegating jobs amongst all participating nodes of the community. In other words, the MaGate schedulers are driven to cooperated 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.

Currently, both design and the first prototype of MaGate have been completed. Together with the adopted resource discovery service, the reference experiment results have proven a functional ready MaGate scheduler, which is able to address different scheduling related events using an uniformed modular architecture. Specially, the capability of scheduler interoperation and work sharing within the interconnected community is emphasized, and various community scheduling related parameters have been evaluated to illustrate the effectiveness brought by sharing *local unsuited jobs* within an interoperable and collaborative grid community.

Regarding the future work, the second MaGate prototype is under the development. Firstly, an advanced *Community Component* will be re-implemented based on WS-* protocols. Secondly, more local scheduling algorithms are planned to be supported to evaluate their behaviors within the environment of community collaboration. Finally, a community algorithm is to be proposed to integrate various community scheduling related parameters flexibly, and facilitate the negotiation/re-negotiation between different MaGates automatically.

MaGate is supported by an efficient resource discovery service on fully decentralized grid infrastructure, which is also developed within the SmartGRID project and named as Smart Signaling Layer (SSL). Concerning the SSL, the next development focuses on ant algorithms to support proactive monitoring and resource discovery. Additionally, an improved version of the Solenopsis framework is being developed, and will be integrated with the next prototype of MaGate.

References

Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Pruyne, J., et al. (2004). *Web Services Agreement Specification (WS-Agreement)* (Tech. Rep.). USA: Open Grid Forum.

Anjomshoaa, A., Brisard, F., Drescher, M., Fellows, D., Ly, A., McGough, S., et al. (2005). *Job submission description language (JSDL) specification* (Tech. Rep.). USA: Open Grid Forum.

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm intelligence: from natural to artificial systems*. New York, NY, USA: Oxford University Press, Inc.

Brocco, A., Frapolli, F., & Hirsbrunner, B. (2008, September). BlatAnt: Bounding Networks' Diameter with a Collaborative Distributed Algorithm. In *Sixth International Conference on Ant Colony Optimization and Swarm Intelligence.* Bruxelles: Springer.

Brocco, A., Frapolli, F., & Hirsbrunner, B. (2009, April). Bounded Diameter Overlay Construction: A Self Organized Approach. In *IEEE Swarm Intelligence Symposium.* Nashville, Tennessee, USA: IEEE.

Brocco, A., Hirsbrunner, B., & Courant, M. (2007, April). Solenopsis: A Framework for the Development of Ant Algorithms. In *Swarm Intelligence Symposium* (pp. 316–323). Honolulu, Hawaii: IEEE.

Buyya, R., & 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*(13-15), 1175–1220.

Di Caro, G., & Dorigo, M. (1998). AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, *9*(2), 317–365.

Foster, I., & Kesselman, C. (1997). Globus: a Metacomputing Infrastructure Toolkit. *International Journal of High Performance Computing Applications*, *11*(2), 115.

Goodale, T., Jha, S., Kaiser, H., Kielmann, T., Kleijer, P., Laszewski, G. von, et al. (2006). SAGA: A Simple API for Grid Applications. High-level application programming on the Grid. *Computational Methods in Science and Technology*, *12*(1), 7–20.

Huang, Y., Brocco, A., Courant, M., Hirsbrunner, B., & Kuonen, P. (2009). MaGate Simulator: a simulation environment for a decentralized grid scheduler. In *International Conference on Advanced Parallel Processing Technologies (APPT'09).* Rapperswil, Switzerland: Springer.

Huang, Y., Brocco, A., Kuonen, P., Courant, M., & Hirsbrunner, B. (2008). SmartGRID: A Fully Decentralized Grid Scheduling Framework Supported by Swarm Intelligence. In *Seventh International Conference on Grid and Cooperative Computing, 2008. GCC '08* (pp. 160–168). China: IEEE Computer Society.

Huedo, E., Montero, R., & Llorente, I. (2005). The GridWay framework for adaptive scheduling and execution on grids. *Scalable Computing: Practice and Experience*, *6*(3), 1–8.

Klusacek, D., Matyska, L., & Rudova, H. (2008). Alea-Grid Scheduling Simulation Environment. *Lecture Notes in Computer Science*, *4967*, 1029.

Michlmayr, E. (2006). Ant Algorithms for Search in Unstructured Peer-to-Peer Networks. In *the 22nd International Conference on Data Engineering Workshops (ICDE2006)* (p. 142). Washington, DC, USA: IEEE Computer Society.

Moab Grid Suite. (2009). `http://www.clusterresources.com/pages/products/moab-grid-suite.php`. Cluster Resource Inc.

Ripeanu, M., & Foster, I. (2001). Peer-to-peer architecture case study: Gnutella network. In *First Conference on Peer-to-peer Computing* (pp. 99–100). Sweden: IEEE Computer Press.

Ripeanu, M., Iamnitchi, A., Foster, I., & Rogers, A. (2007). In Search of Simplicity: A Self-Organizing Group Communication Overlay. *University of British Columbia TR-2007-05*.

Schmid, S., & Wattenhofer, R. (2007). Structuring Unstructured Peer-to-Peer Networks. In *14th Annual IEEE International Conference on High Performance Computing (HiPC).* Goa, India: IEEE Press.

Schoonderwoerd, R., Holland, O., Bruten, J., & Rothkrantz, L. (1997). Ant-based load balancing in telecommunications networks. *Adaptive behavior*, *5*(2), 169.

Schopf, J. (2003). Ten actions when superscheduling: A grid scheduling architecture. In *Workshop on Scheduling Architecture.* Tokyo: Global Grid Forum.

Shen, K. (2004). Structure management for scalable overlay service construction. In *First Symposium on Networked Systems Design and Implementation (NSDI'04)* (pp. 21–21). San Francisco, California: USENIX Association.

Tonellotto, N., Wieder, P., & Yahyapour, R. (2005). A proposal for a generic grid scheduling architecture. In *Integrated Research in Grid Computing Workshop* (pp. 337–346). Greece: Springer.

Troger, P., Rajic, H., Haas, A., & Domagalski, P. (2007). Standardization of an API for Distributed Resource Management Systems. In *CCGRID '07: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid* (pp. 619–626). Washington, DC, USA: IEEE Computer Society.

Waldrich, O., Wieder, P., & Ziegler, W. (2006). A meta-scheduling service for co-allocating arbitrary types of resources. *Lecture Notes in Computer Science*, *3911*, 782.

Xiaohui, W., Zhaohui, D., Shutao, Y., Chang, H., & Huizhen, L. (2006). CSF4: A WSRF compliant meta-scheduler. In *The 2006 World Congress in Computer Science, Computer Engineering, and Applied Computing, GCA* (Vol. 6, pp. 61–67). Las Vegas, Nevada, USA: Bentham Science.