SmartGRID: A fully decentralized Grid Scheduling Framework supported by Swarm Intelligence

Ye Huang, Amos Brocco, Pierre Kuonen, Michèle Courant, Béat Hirsbrunner

***Abstract*—Resource management and scheduling has proven to be one of the key topics for grid computing. Nowadays, the resource management field is subdivided into low-level and high-level approaches. While low-level resource management systems normally concern the scheduling activities within a single virtual organization, high-level schedulers focus on the large scale resources utilization with unstable resource availability, low reliability networks, multi-policies, multi-administrative domains, etc.**

In this paper, we propose a decentralized framework named SmartGRID to tackle high-level grid resource management and scheduling. Within the SmartGRID framework, swarm intelligence algorithms are used for resource discovery and monitoring, standard protocols and schemes are adopted for scheduler interoperability, and an embedded plugin mechanism is provided to utilize multi-type external scheduling strategies. With a clearly decoupled layered architecture, SmartGRID has been designed to be a generic and modular environment to support intelligent and interoperable grid resource management upon a volatile, dynamics, and heterogeneous grid computing infrastructure.

***Index Terms*—Grid, Scheduler Interoperability, High-level Scheduling, Swarm Intelligence, Ant, Grid Community.**

I. Introduction

Grid computing means coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations [1]. Grid environments contain a large number of complex services with variable functionalities. The integration of these services requires a flexible, extensible, and consensual resource management and scheduling solution, which is unavailable yet.

For a precise definition and categorization of diverse scheduling responsibilities, [2] proposed the expressions 'high-level' scheduler to denote a scheduler that is actually negotiating with another scheduler for possible job allocation, and 'low-level' resource management system for the part focusing on local job allocation and execution within each single virtual organization (VO).

Generally, resource owners of diverse VOs participate in a grid environment by providing resources with specific strategies. In this context, low-level resource management systems are responsible for assigning jobs to appropriate resources found in the same VO; on the other hand, high-level schedulers concern with scheduling events interaction, negotiation and allocation amongst multiple organizations.

Ye Huang is Ph.D student at the Department of Informatics, University of Fribourg, Switzerland. Phone: +41 26 429 6595, e-mail: ye.huang@unifr.ch

Amos Brocco is Ph.D student at the Department of Informatics, University of Fribourg, Switzerland. Phone: +41 26 300 8481, e.mail: amos.brocco@unifr.ch

Pierre Kuonen is professor at the Department of Information and communication, University of Applied Sciences of Western Switzerland, Fribourg. Phone: +41 26 429 6565, e-mail: pierre.kuonen@hefr.ch

Michèle Courant is senior researcher at the Department of Informatics, University of Fribourg, Switzerland. Phone: +41 26 300 8470, michele.courant@unifr.ch

Béat Hirsbrunner is professor at the Department of Informatics, University of Fribourg, Switzerland. Phone: +41 26 300 8467, e-mail: beat.hirsbrunner@unifr.ch

SmartGRID [3] [4] is a cooperative project aiming at bringing a decisive increase in efficiency, robustness, and reliability regarding the volatile, dynamics, and heterogeneous grid computing infrastructure. SmartGRID comprises three parts: the Smart Resource management layer (SRML), the Smart Signaling layer (SSL), and the Data Warehouse Interface (DWI).

The Smart Resource Management Layer (SRML) is an interoperable grid scheduler community composed of engaged decentralized high-level schedulers named *MaGate*[1], which are modular designed for easy extension. With the infrastructure information retrieved from the DWI, MaGates discover and connect to each other, and collaborate to construct an integrated grid community, which is used to bridge the heterogeneous grid systems with a consensual view. Furthermore, the grid community can evolve dynamically, and recover a failed grid section automatically.

Information about available resources and network status is gathered by the Smart Signaling Layer (SSL), and then put into DWI's distributed data storages. Being based on swarm intelligence algorithms, the SSL is composed by an overlay network of *Nest*s that provide the runtime environment for the execution of ant mobile agents. This approach provides an adaptive and robust signaling mechanism, supporting monitoring of the grid and resource discovery.

The remainder of the paper is organized as follows: in Section II, an overview of several existing scheduling approaches, along with the motivation of SmartGRID are introduced. Section III explains in detail the SmartGRID framework architecture, vision, and intended implementation. A typical scheduling scenario is depicted in Section IV. Conclusions and future work are presented in Section V, and acknowledgements are presented in Section VI.

II. Related Works

Efficient high-level grid scheduling solution is critical for large scale grid environments that spread over multiple administrative domains. A number of approaches on this topic have been proposed.

GridWay [5] is a high-level scheduler inside the Globus Toolkit that provides abundant scheduling policies with

[1] *MaGate* stands for *'Magnetic Gateway'* of grid system

a modular structure. In order to avoid scheduling self-competition, GridWay only allows one scheduler to manage each virtual organization.

gLite WMS [6] is the scheduler used in the EGEE [7] project. It has two very interesting policies, eager policy and lazy policy. WMS accepts jobs described in JDL [8], stores them in its own task queue, and submits jobs to a computational grid through Condor-G [9].

Meta-Scheduling Service (MSS) [10] is a high-level scheduler developed within the VIOLA project [11]. MSS is an advanced high-level scheduler with pre-defined policies, and designed to be middleware independent. MSS current version is implemented upon Unicore USite architecture.

Agent-based Scheduler Framework (ASF) [12] is a bottom-up designed grid scheduling framework that consists of a discreet meta-scheduler and a series of autonomic agents attached to each local resource manager. ASF achieves scheduling with closely local information.

In addition, new scheduling approaches composed of several sub- systems, such as D-Grid (MMS, WSS) [13] and IANOS (ISS and MSS) [14] are being proposed. Such systems are compliant with the emerging standard structure of general high-level schedulers, such as 'Teikoku' [15] and 'Scheduling Instance' [16].

Existing high-level schedulers are set up to bridge the communication gap amongst diverse existing low-level resource management systems. Different strategies are adopted to decide where the grid jobs should be sent for further disposal. However, due to the general problem of lack of grid infrastructure information, there are two great assumptions for their usage scenarios: first, the grid scope is already known; second, only one scheduler exists in each known grid system.

In order to overcome the dilemma mentioned above, with respect to existing grid scheduling systems, SmartGRID proposes a decentralized and interoperable grid scheduling framework, which integrates the local infrastructure information provided by swarm intelligence technology to construct a consensual grid community.

Swarm intelligence is a branch of artificial intelligence that focuses on distributed collaborative algorithms inspired by the behavior of swarms of insects. Bioinspired solutions have already been successfully applied to several network routing problems [17], [18], as well as for resource discovery in unstructured networks [19].

There also exist some examples of grid platforms exploiting swarm intelligence algorithms, and ant colony algorithms in particular. A first example is represented by Messor [20], a grid computation system built on the Anthill framework [21]. Messor implements a distributed load balancing algorithm inspired by the behavior of a species of ant. This algorithm is very popular, and other platforms rely on similar algorithms to execute load balancing tasks. For example, the ARMS [22] middleware is based on software agents, and uses swarm based load balancing algorithms.

In general, swarm algorithms are praised for their intrinsic distributed design, as well as for their adaptivity and robustness in failure situations. These are undoubtfully interesting aspects that can be exploited to provide a robust monitoring mechanism and optimal resource discovery services within the SmartGRID framework.

III. SmartGRID Framework

In this section, the SmartGRID Framework is introduced. Firstly, the design goal and layered architecture overview are presented. Then the SmartGRID community infrastructure and SG-Node structure are illustrated, followed by an in-depth discussion on each relevant element.

A. Design Goal

SmartGRID has been designed to be a generic and modular framework in order to support intelligent and interoperable grid resource management using swarm intelligence algorithms and multi-type grid scheduling strategies. The proposed solution is layer structured and aims at filling the gap between grid applications, which act as the resource consumers, and the grid resource low-level management systems, which behave as the resource providers. To this purpose, SmartGRID proposes an autonomic and evolutional grid community composed of SmartGRID Nodes (*SG-Node*), which will be illustrated in the following subsections.

B. Layered Architecture Overview

SmartGRID framework is structured into two layers and one internal interface. The Smart Resource Management Layer (SRML) is responsible for grid level dynamic scheduling and interoperation to serve grid applications with best use of the available computing resources. The Smart Signaling Layer (SSL) is in charge of monitoring and constituting knowledge on network and resources. Finally, the Data Warehouse Interface (DWI) is used to mediate the scheduling and signaling layers. The overall architecture of the system is shown in Figure 1.

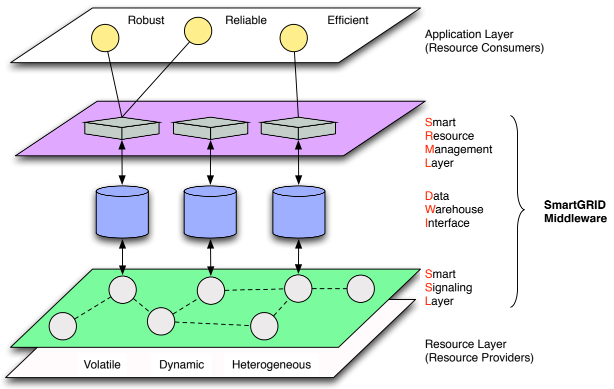

Fig. 1. SmartGRID architecture overview

C. Grid Community Infrastructure

As mentioned above, SmartGRID aims at constructing an integrated high-level grid community. The grid community is constituted of all the connected and engaged SG-Nodes. As shown in Figure 2, SG-Nodes can behave different roles depending on their positions within the grid community.

SG-Nodes can work as *executors* if they could recognize and make use of local resources to dispose the appropriate accepted scheduling requests, such as SG-Node B, C and D.

SG-Nodes connecting to other high-level schedulers are considered as *routers*, which can transport scheduling events inside the grid community, such as SG-Node E.

SG-Nodes can also work as *interfaces* when used to interact with grid users or applications, such as SG-Node A.

Moreover, SG-Nodes can behave multiple roles during their lifecycles; they can also implement all the possible functionalities and be used as a *full functional* SG-Node, such as SG-Node F.

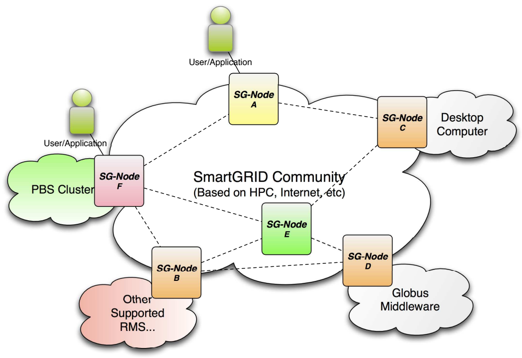

Fig. 2. SmartGRID community infrastructure

D. SG-Node Structure

To achieve our design goal with SmartGRID layered architecture, each individual SG-Node is comprised of three entities, each entity belonging to a separated layer:

- *MaGate*, is a part of SRML and acts as the gateway of the SG-Node.
- *SSL Nest*, is a part of SSL and continuously constitutes the knowledge of network and resources.
- *DataStorage*, which comes from DWI and fills the communication gap between SRML MaGate and SSL Nest.

The SG-Node internal structure is shown in Figure 3.

The MaGate of each SG-Node is responsible for collecting resource capability information via low-level resource management systems, middleware systems, or manual-based methodology, and storing the obtained data into the corresponding SG-Node DataStorage. Meanwhile, the Nest of the same SG-Node dedicates to discover remote

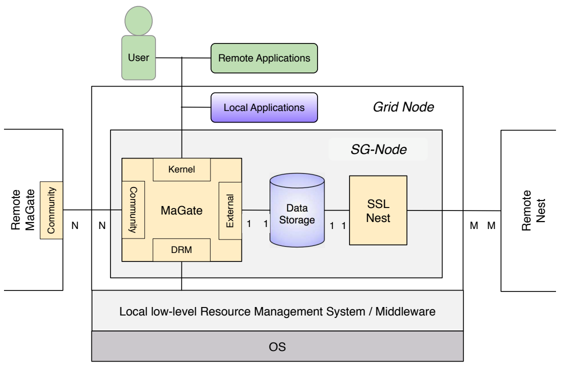

Fig. 3. The SG-Node internal structure overview

SG-Nodes by contacting remote SG-Node Nests. Once the Nest connections are constructed, resource and scheduling information stored inside the DataStorages could spread over the relevant SG-Nodes, and the MaGates could recognize the existence of their neighborhood. Finally, the grid community is established upon the negotiation and interaction amongst MaGates from diverse SG-Nodes.

E. Smart Resource Management Layer

The Smart Resource Management Layer (SRML) consists of the SmartGRID high-level schedulers named *MaGate*. MaGates are modular, and aim at being an open structure, fully decentralized, and interoperable high-level schedulers, capable of integrating diverse external grid components easily.

The core value of MaGate is *interoperability*. Beyond the high-level scheduler's traditional functionality of dispatching scheduling decisions to low-level resource management systems or middlewares, MaGates focus on the interoperability of high-level schedulers, which facilitates negotiation and transport scheduling events (requests, responses) amongst diverse virtual organizations and grid systems. In this context, MaGates empower the users to utilize resources outside the already known grid systems, with fully decentralized topology and high reliability.

The continuously updated and reliable information within DataStorages contributed via the external components, especially SSL Nests, provides the grid infrastructure information as the basis of MaGates's interaction. Furthermore, the MaGates' interoperation is based on several existing and merging protocols and standards, such as WS-Agreement [23], JSDL [24] and SDL [25].

Finally, with the interoperation capability, MaGates work together to construct an autonomic, evolutional grid community, which can span the differences amongst multi-type grids such as Enterprise Grids, HPC Grids and Global Grids [16].

Besides the interoperable community, MaGates are also designed to cover many other functionalities related to high-level scheduling, and compliant to several existing high-level scheduler general models, such as 'Scheduling Instance'. In order to be flexible and efficient, each individual MaGate can only exhibit a subset of its capabilities by launching corresponding components, depending on its specific usage scenario.

To achieve the aforementioned design purpose, the MaGate comprises five separated components: the Kernel, the Interface, the Community, the DRM, and the External, for increasing system flexibility and compatibility. The components are not monolithic and each component's involved services are also configurable. The MaGate component structure is presented in Figure 4, and an in-depth description of each component follows.

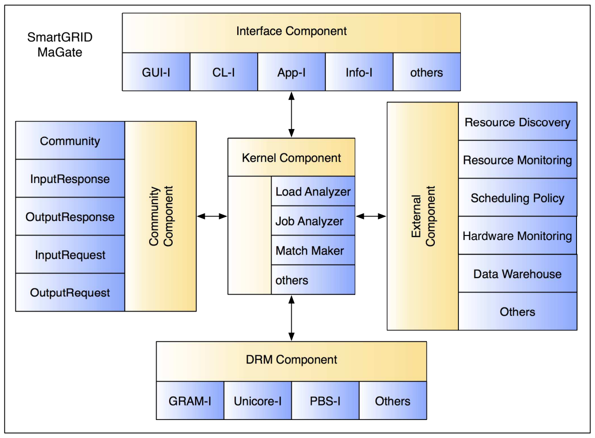

Fig. 4. *MaGate* component structure

E.1 Kernel Component

Kernel Component is responsible for MaGate self-management; it makes use of specific embedded or external components to help reducing the cost and complexity of owning a SG-Node. The *Kernel Component* covers several essential services such as:

- *LoadAnalyzer* service utilizes different approaches (embedded or external via *External* component) to monitor and analyze system workload.
- *JobAnalyzer* service is used to parse diverse grid job standards, e.g. JSDL.
- *MatchMaker* service utilizes multiple strategies (embedded or external via *External* component) to make scheduling decisions.

E.2 Interface Component

Interface Component provides interactive channels to enable SG-Node features and functionalities accessible and reusable by external grid applications and components. *Interface Component* relevant services includes:

- *GUI-I* service provides GUI (Web or Desktop client) interface for grid users.
- *CL-I* service provides command-line interface for grid users.
- *App-I* service provides an abstract interface for both local and remote high-level grid applications and programming models, e.g. POP-C++ [26].
- *Info-I* service provides the abstract information interface to interact with external data storages.

E.3 DRM Component

DRM Component interacts with distributed low-level resource management systems and middlewares located within the grid community for job local execution. Instead of replacing the existing local proprietaries and strategies, the *DRM Component* provides a series of interfaces, which support diverse low-level approaches, such as Globus, Unicore, and PBS.

E.4 Community Component

Community Component is a mandatory component for MaGate. It acts as the grid scheduling connector that transfers the scheduling requests and responses from one grid section (logically or geographically) to another. It is also strongly engaged to construct the dynamic and autonomic grid community with help of other components (e.g. SSL Nests). With the above capabilities, the *Community Component* is capable of providing an abstract and reliable grid view upon the real volatile, dynamic, and heterogeneous grid infrastructure. Considering the suggestion from 'Scheduling Instance', *Community Component* groups five services, which are the following:

- *OutputRequest* service describes and dispatches the grid application requests that can't be carried out by the local SG-Node, to other SG-Nodes within the grid community.
- *InputRequest* service accepts and analyzes the scheduling requests sent by other SG-Nodes.
- *OutputResponse* service is used to send the scheduling decisions and results made by the local SG-Node back to the scheduling request initiator.
- *InputResponse* service is used to accept scheduling decisions made by the remote SG-Nodes, to which the scheduling requests are dispatched.
- *Community* service focuses on interaction and negotiation with other SG-Nodes within the grid community. With the persistently updated grid information retrieved from local DataStorage, *CommunityService* is used to construct and maintain a consensual high-level grid view.

E.5 External Component

External Component offers the plug-in mechanism for MaGate. It could be considered as the multi- functional outlet that helps underpin the MaGate by integrating appropriate external grid services, components, algorithms and strategies. Some typical services of *External Component* are:

- *ResourceDiscovery* service focuses on grid resource (machines, network bandwidth, etc.) information discovery and collection from the infrastructure.

- *ResourceMonitoring* service is responsible for monitoring the availability and reliability of the discovered resources.
- *HardwareMonitoring* service facilitates the MaGate to utilize external hardware monitoring systems, such as PAPI [27] or Ganglia [28], to scan and retrieve resource hardware information if supported by the local systems.
- *SchedulingPolicy* service empowers the MaGate to adopt diverse external scheduling policies for different using scenarios.
- *DataWarehouse* service implements the *Interface.Info-I* interface to provide a specific data access implementation for DataStorage.

It should be emphasized that, the SmartGRID SSL strongly participates several important external services as the default option, like e.g. *ResourceDiscovery* and *ResourceMonitoring* services.

F. Smart Signaling Layer

The Smart Signaling Layer represents the interface from and to the network of the SmartGRID architecture, and provides information about the availability of other resources on other nodes, as well as their status. From the SSL point of view, a node has some partial knowledge of the underlying logical network. Nodes within this partial view are called *neighbors*. The SSL hides the complexity and instability of the underlying network by offering reliable services based on distributed ant algorithms. Ant algorithms do not require centralized control, and are known to be robust and adaptable, thus well suited for dynamic networks. Ants are defined as lightweight mobile agents traveling across the network, collecting information on each visited node. The activity of the SSL can be either reactive or proactive. Reactive behaviors are controlled by the upper layer: information is asynchronously transmitted through a dataware house interface (discussed in Section G). The same interface is used to provide feedback and results of the execution of algorithms. Continuous pro-active activities, such as network monitoring, enhance the QoS of provided services and the robustness of the whole system.

F.1 Solenopsis

Each node runs the Solenopsis middleware[29], providing an environment for the execution of ant colony algorithms. Solenopsis is a completely distributed ant platform consisting of a programming language to develop ant agents, a compiler and a virtual machine. The modular design of the framework allows the implementation of pluggable services that can be accessed by ants.

Figure 5 shows an overview of the framework. The middleware is comprised of a management daemon, service libraries, and virtual machines. Everything runs on top of the existing operating system. Services accessible by ants varies from migration, to logging, or storage. In the figure, the migration procedure is shown:

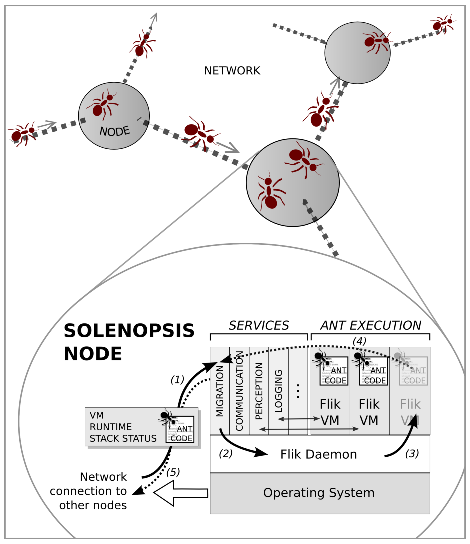

Fig. 5. Solenopsis Framework Overview

1. an ant is transferred to a node: the migration service listens for incoming transfer requests and initiates the transfer;
2. the migration services requests a new virtual machine instance to the daemon;
3. a virtual machine is created, and the ant status (execution stack) is restored;
4. an ant can request a migration by calling the migration service on the node;
5. the execution state is serialized and the ant is migrated to a target node.

Solenopsis supports strong migration of ant agents: not only is the code migrated from node to node, but also the execution state. Because algorithm data can be carried by the ant, it is possible to migrate an ant at any time during the execution, without explicitly saving data. The algorithm can check whether migration has succeeded or failed by testing the return value of the migration function.

As the platform itself does not define the ant algorithms, updates to the services offered by the SSL are just a matter of executing new species of ants. In the same spirit, different types of ant can collaborate to offer new kind of services without interfering with the existing species. Current version of the Solenopsis middleware is implemented in Java, and offers both completely decentralized execution of algorithms, as well as centralized and synchronized execution suited for analysis and evaluation.

F.2 BlåtAnt Algorithm

There exist different methods to provide resource discovery in distributed environments. Some solutions exploit particular topologies to optimize the search path [30]. Maintaining strict topologies can be difficult in highly dynamic networks, thus methods that can be implemented on unstructured networks such as flooding-like or random walk algorithms are preferred. To provide resource discovery and efficient monitoring of the SmartGRID network, we

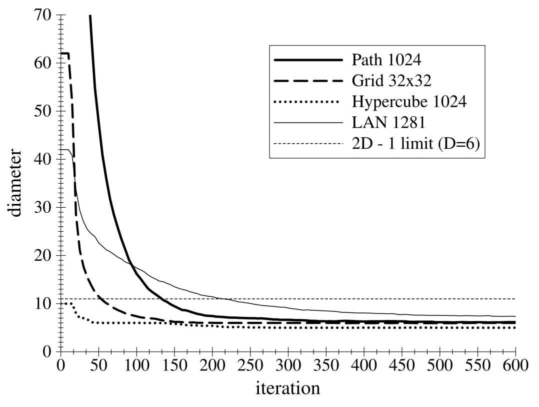

Fig. 6. Diameter Evolution

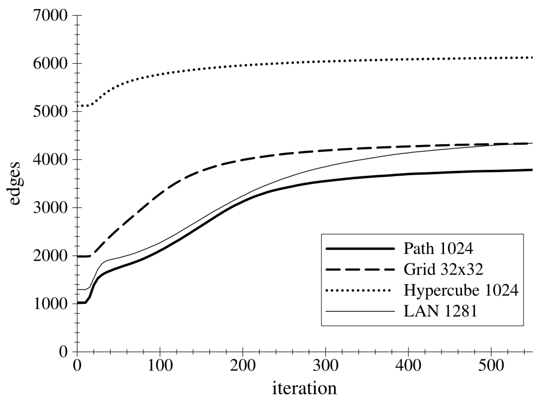

Fig. 7. Edge Count

propose to augment the existing topology with a minimal number of new logical links in order to reduce the diameter of the corresponding graph, thus minimizing the maximum distances between peers. This graph augmentation is carried out by the BlåtAnt collaborative ant algorithm [31], [32]. The rewiring method is based on two rules, tied to an user-defined parameter D.

Theorem 1 (Connection Rule) Two nodes n_i and n_j in a graph G are connected by the algorithm if:

$$d_G(n_i, n_j) \geq 2D - 1 \quad (1)$$

where $d_G(n_i, n_j)$ is the minimum distance between n_i and n_j in G.

Theorem 2 (Disconnection Rule) Two connected nodes n_i and n_j in a graph G, $i \neq j$ are disconnected if there exist another neighbor n_k of n_i such that:

$$d_{G'}(n_j, n_k) + 1 \leq D \quad (2)$$

where G' is a graph obtained from G, removing node n_i.

Using these two rules, the algorithm is able to minimize the diameter of a network to a value less than $2D - 1$. Under these assumptions, improved flooding or random walk peer discovery algorithms can be implemented.

Figure 6 and 7 show the diameter evolution, respectively the number of edges, resulting from the execution of the algorithm on several topologies. Considered scenarios include a path graph, a two-dimensional grid and an hypercube, each consisting of 1024 nodes, as well as a LAN network topology of 1281 nodes. Simulations were run with $D = 6$, and results show that BlåtAnt is able to reduce the diameter of the network to a value $\leq 2D - 1 = 11$, by adding logical links between nodes while avoiding creating a fully connected network. Actual research focuses on exploiting the characteristics of the resulting network in order to provide optimized discovery and monitoring services with minimal communication overhead.

G. Data Warehouse Interface

SmartGRID Data Warehouse Interface (DWI) is supposed to act as the loosely coupled communication channel for connecting SRML and SSL. Considering the decoupled data access approach, both SRML and SSL could benefit by ignoring the implementation technologies of the other side.

DWI is comprised of a series of distributed datawarehouses named *DataStorage* that store both persistent and cached grid information concerning network infrastructure, resource status, grid scheduling request/response, strategy parameters, SmartGRID specific events, etc. The grid information are presented with the help of existing schemes, such as GLUE [33], JSDL [24] and SDL [25].

The SSL operates in response to triggers on updates of data in the *DataStorage*s. Similarly, information retrieved by the SSL is put in the *DataStorage*s and triggers notify the SRML.

IV. Scheduling Scenario

This section describes a typical scenario of how the SG-Nodes (one local, several remotes) collaborate to schedule submitted sequential batch jobs within the SmartGRID community. The 'sequential batch' job type is chosen to avoid unnecessary complexity and depict the typical SmartGRID usage. The scenario comprises 5 phases.

Phase 1 Jobs submitting via grid users or applications.

1.1 Local SG-Node's MaGate *Interface.GUI-I* or *Interface.App-I* service receives the jobs submitted by grid users or grid applications. The jobs are normally described using JSDL.

1.2 MaGate *Kernel.JobAnalyzer* service verifies whether the jobs' description could be recognized and disposed.

1.3 MaGate *Interface.Info-I* service contacts its specific implementation, e.g. *External.DataWarehouse* service, to query local SG-Node capabilities, including resource parameters, workload, low-level resource management system type.

1.4 MaGate *Kernel.MatchMake* service utilizes specific policies (embedded or offered via *External.SchedulingPolicy* service) to determine whether the local resource is competent to carry out each individual received job.

Phase 2 If the local SG-Node is competent.

2.1 MaGate *Kernel.MatchMake* service utilizes specific policies to determine where the received job should be allocated locally.

2.2 Corresponding MaGate *DRM* service allocates the job to the targeted local low-level resource management system for execution.

2.3 Optional, MaGate *External.HardwareMonitoring* service could be launched to monitor the interested behavior data during the job execution.

2.4 Once the job execution is finished, the MaGate *Interface.Info-I* service would retrieve the results and store them into the local *DataStorage*.

2.5 The grid users and applications could access the execution status data and results via interrelated interface services, e.g. *Interface.GUI-I* or *Interface.App-I*.

Phase 3 If the local SG-Node is NOT competent.

3.1 MaGate *Kernel.MatchMake* service submits the unmatched jobs's requirement to local *External.ResourceDiscovery* service.

3.2 MaGate *External.ResourceDiscovery* service analyzes and transfers the resource parameters required by unmatched jobs to local *SSL Nest*, via local *DataStorage*.

3.3 Local *SSL Nest* receives the new requests via *DataStorage* trigger mechanism, and propagates ants to spread over the *SSL* optimized network for resource discovery.

3.4 Within pre-defined lifetime, ants sent by local *SSL Nest* bring diverse remote SG-Nodes information back, whose resource capabilities might satisfy the unmatched jobs's requirement.

3.5 Local *SSL Nest* sends the discovered remote SG-Nodes information to local *DataStorage*, where the data could be notified to MaGate *External.ResourceDiscovery* service.

3.6 MaGate *Kernel.MatchMake* service filters out the unsuitable discovered remote SG-Nodes and constructs an ordered candidate list for each unmatched job.

3.7-a If no candidate remote SG-Nodes are available, MaGate *External.ResourceDiscovery* service would be reactivated again with adjusted parameters, such as processing lifetime, discovery depth.

3.7-b If the candidate remote SG-Nodes are available, local MaGate *Community.OutputRequest* service starts to negotiate with each remote SG-Node to migrate the unmatched job, following the preference expressed in the ordered list.

3.8-a If no agreement could be achieved after all the candidate remote SG-Nodes have been negotiated, local MaGate *External.ResourceDiscovery* service would be reactivated again.

3.8-b Once a job migration agreement is achieved, local MaGate *Community.OutputRequest* service will transfer the job to the interrelated remote SG-Node MaGate *Community.InputRequest* service. Additional, local MaGate *Community.InputResponse* service could be started to monitor the following migration status data and result sent from the interrelated remote SG-Node.

Phase 4 Local SG-Node disposes the scheduling events from other remote SG-Nodes.

4.1 Once local SG-Node receives a scheduling event, e.g. remote job migration request, from its MaGate *Community.InputRequest* service, it will check the local resource capability to ensure qualification for request acceptance; the checking process is similar with phase 1.3.

4.2-a If local resource is suitable for the received scheduling event, MaGate *Community.OutputResponse* service would send the appropriate response made by local *Kernel.MatchMaker* service to the request initiator.

4.2-b If local resources don't fit for the request, MaGate *Community.OutputResponse* service will ask the request initiator, whether the request should be cancelled, or be spreaded out to somewhere else within specific lifetime.

Phase 5 Local SG-Node community knowledge self-evolution.

5.1 In some context, the local SG-Node needs to maintain the contiguous remote SG-Nodes information, which could be useful to provide a broad grid community view to the grid users and applications; meanwhile, contiguous SG-Nodes information also empowers local MaGate the collaborative capability to recover the failure community section.

5.2-a To discover new remote SG-Nodes, MaGate *Kernel.LoadAnalyzer* service sends the corresponding requests to local *External.ResourceDiscovery* service, and stores the discovered results in the local *DataStorage*.

5.2-b In order to keep the discovered remote SG-Nodes information up-to-date, local MaGate *External.ResourceMonitoring* service is periodically invoked to monitor the remote SG-Nodes's updating via local *SSL Nest*'s ant spreading, and updates the relevant data within the local *DataStorage*.

V. Conclusions and Future Work

This paper presented a brief overview of the SmartGRID framework, as well as the in-depth discussion about each layer of the solution. The proposed framework focuses on establishing an interoperable, autonomic, evolutional grid community to span the differences amongst heterogeneous grid environments, based on reactive and reliable grid infrastructure information provided by swarm intelligence technology. Currently, the design of SmartGRID framework has been completed, and an ant based signaling prototype has been implemented to verify the network shrinking and optimizing efficiency.

For this purpose, a high-level scheduling approach named SRML is designed, which comprises a series of full decentralized schedulers - *MaGate*s. The MaGates are component structured, interoperability emphasized, and empower the users to load partial scheduling functionalities and adopt external services, components, and algorithms.

Moreover, a signaling layer named SSL is introduced to utilize ant based swarm intelligence algorithms for persistent information gathering and load balancing within grid systems.

Finally, a communication channel named DWI mediates and decouples SRML and SSL so that grid users could adopt the layered functionalities separately.

Regarding the future work, we are focusing on the first prototype of MaGate, especially the *Community Component*. Meanwhile, we are working on the first development of *DataStorage*, which links the MaGate and SSL Nest to construct an integrated SG-Node prototype for validation and optimization.

Concerning the SSL, development focuses on ant algorithms to support proactive monitoring and resource discovery. Additionally, an improved version of the Solenopsis framework is being developed.

VI. Acknowledgements

SmartGRID project is designed and implemented under the collaboration between PAI group[2] from University of Fribourg, and GridGroup[3] from University of Applied Sciences of 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) (2001) 200

[2] Schwiegelshohn, U., Yahyapour, R.: Attributes for communication between scheduling instances. Global Grid Forum (GGF), December (2001)

[3] Hirsbrunner, B., Courant, M., Brocco, A., Kuonen, P.: SmartGRID: Swarm Agent-Based Dynamic Scheduling for Robust, Reliable, and Reactive Grid Computing. Technical report, Working Paper 06–13, Department of Informatics, University of Fribourg, Switzerland, October 2006

[4] Brocco, A., Hirsbrunner, B., Courant, M.: A modular middleware for high-level dynamic network management. Proceedings of the 1st workshop on Middleware-application interaction: in conjunction with Euro-Sys 2007 (2007) 21–24

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

[6] gLiteWMS: http://glite.web.cern.ch/glite/packages/r3.0/deployment/glite-wms/glite-wms.asp (2008)

[7] EGEE: http://www.eu-egee.org/ (2008)

[8] Pacini, F., Kunzt, P.: Job Description Language Attributes Specification. EGEE Project (2006)

[9] Frey, J., Tannenbaum, T., Livny, M., Foster, I., Tuecke, S.: Condor-G: A Computation Management Agent for Multi-Institutional Grids . Cluster Computing **5**(3) (2002) 237–246

[10] Waldrich, O., Wieder, P., Ziegler, W.: A Meta-scheduling Service for Co-allocating Arbitrary Types of Resources. Proc. of the Second Grid Resource Management Workshop (GRMWS05) in conjunction with the Sixth International Conference on Parallel Processing and Applied Mathematics (PPAM 2005 **3911** 782–791

[11] VIOLA: http://www.fz-juelich.de/jsc/grid/viola (2007)

[12] Kentaro, S., Hiroyuki, S.: A resource-oriented grid metascheduler based on agents. Proceedings of the 25th conference on Proceedings of the 25th IASTED International Multi-Conference: parallel and distributed computing and networks table of contents (2007) 109–114

[13] Waldrich, O., Ziegler, W., Papaspyrou, A., Wieder, P., Yahyapour, R.: Novel approaches for scheduling in d-grid towards an interoperable scheduling framework. Technical Report TR-0122, Institute on Resource Management and Scheduling, CoreGRID - Network of Excellence (December 2007)

[14] Rasheed, H., Gruber, R., Keller, V., Waldrich, O., Ziegler, W., Wieder, P., Kuonen, P., Sawley, M., Maffioletti, S., Kunszt, P.: Ianos: An intelligent application oriented scheduling middleware for a hpc grid. Technical Report TR-0110, Institute on Resource Management and Scheduling, CoreGRID - Network of Excellence (December 2007)

[15] Grimme, C., Lepping, J., Papaspyrou, A., Wieder, P., Yahyapour, R., Oleksiak, A., Supercomputing, P., Center, N., Waldrich, O., Ziegler, W.: Towards a standards-based grid scheduling architecture. Technical Report TR-0123, Institute on Resource Management and Scheduling, CoreGRID - Network of Excellence (December 2007)

[16] Tonellotto, N., Yahyapour, R., Wieder, P.: A proposal for a generic grid scheduling architecture. Proceedings of the Integrated Research in Grid Computing Workshop (2005) 337–346

[17] Schoonderwoerd, R., Holland, O., Bruten, J., Rothkrantz, L.: Ant-based load balancing in telecommunications networks. Adaptive Behavior **5**(2) (1996) 169–207

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

[19] Michlmayr, E.: Ant algorithms for search in unstructured peer-to-peer networks. In: ICDEW '06: Proceedings of the 22nd International Conference on Data Engineering Workshops, Washington, DC, USA, IEEE Computer Society (2006) 142

[20] Montresor, A., Meling, H.: Messor: Load-balancing through a swarm of autonomous agents. In: In Proceedings of the 1st Workshop on Agent and Peer-to-Peer Systems. (2002)

[21] Babaoglu, O., Meling, H., Montresor, A.: Anthill: a framework for the development of agent-based peer-to-peer systems. Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on (2002) 15–22

[22] Cao, J.: ARMS: An agent-based resource management system for grid computing. Scientific Programming **10**(2) (2002) 135–148

[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). Global Grid Forum GRAAP-WG, Draft, August (2004)

[24] Anjomshoaa, A., Brisard, F., Drescher, M., Fellows, D., Ly, A., McGough, A., Pulsipher, D., Savva, A.: Job Submission Description Language (JSDL) Specification, Version 1.0. GGF GFD **56** (2005)

[25] Ariel Oleksiak, O.W.: Scheduling attributes (initial proposal of scheduling attributes). Technical Report TR-0123, Open Grid Forum, GSA-RG (December 2007)

[26] Nguyen, T., Kuonen, P.: A Model of Dynamic Parallel Objects for Metacomputing. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications-Volume 1 table of contents (2002) 192–198

[27] Browne, S., Dongarra, J., Garner, N., Ho, G., Mucci, P.: A Portable Programming Interface for Performance Evaluation on Modern Processors. International Journal of High Performance Computing Applications **14**(3) (2000) 189

[28] Massie, M., Chun, B., Culler, D.: The ganglia distributed monitoring system: design, implementation, and experience. Parallel Computing **30**(7) (2004) 817–840

[29] Brocco, A., Hirsbrunner, B., Courant, M.: Solenopsis: A framework for the development of ant algorithms. In: Swarm Intelligence Symposium, SIS, IEEE (April 2007) 316–323

[30] Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Proceedings of the 2001 ACM SIGCOMM Conference. (2001) 149–160

[2] http://diuf.unifr.ch/pai/

[3] http://gridgroup.tic.hefr.ch/

[4] http://www.haslerstiftung.ch/

[31] Brocco, A., Frapolli, F., Hirsbrunner, B.: Shrinking the network: The blåtant algorithm. Technical Report 08-04, Department of Informatics, University of Fribourg, Fribourg (April 2008)

[32] Brocco, A., Frapolli, F., Hirsbrunner, B.: Blatant: Bounding networks' diameter with a collaborative distributed algorithm. In: Sixth International Conference on Ant Colony Optimization and Swarm Intelligence, ANTS, Springer (September 2008) Accepted.

[33] Andreozzi, S., Burke, S., Field, L., Fisher, S., Konya, B., Mambelli, M., Schopf, J., Viljoen, M., Wilson, A.: GLUE Schema Specification version 1.2. OGF GLUE-WG (2007)