RW-OGS: an Optimized Random Walk Protocol for Resource Discovery in Large Scale Dynamic Grids

Emmanuel Jeanvoine[‡], Christine Morin[†]
[‡] INRIA Saclay - Île-de-France research centre - SED
[†] INRIA Rennes - Bretagne Atlantique research centre - PARIS project-team
{*emmanuel.jeanvoine,christine.morin*}*@inria.fr*

Abstract

The information service is an important component of any Grid middleware as it helps users to find resources suitable for the execution of their applications. Traditional approaches based on data bases that have been designed for Grids of moderate size do not fit the requirements of wide-area large scale Grids composed of dozens of thousands highly dynamic nodes. In this paper, we propose a fully distributed information service relying on an unstructured overlay network. The core of this information service is the RW-OGS cached-based optimized random walk protocol. We describe experimental results showing that RW-OGS outperforms state of the art protocols in terms of bandwidth consumption and resource discovery latency. Moreover, RW-OGS is optimized to increase the probability to discover nodes with free resources and to enable fast discovery of rare resources. The RW-OGS protocol has been implemented as part of the Vigne middleware aiming at easing the use of large scale desktop Grids for executing a wide range of distributed applications.

1. Introduction

Since several years, Grid computing has become very attractive for the execution of a wide range of time-consuming applications in various scientific fields. However, harnessing the huge power of aggregated resources in a Grid is still challenging. Many studies have investigated the different issues to be managed in Grid as security, file transfers, scheduling, etc. resulting in Grid middleware such as Globus [6], XtremWeb [4] or Vishwa [15]. We have developed Vigne, a Grid middleware targeting large scale Grids composed of several thousands of dynamic nodes widely distributed [10, 11, 14].

The information service is an important component of any Grid middleware as it helps users to find resources suitable for the execution of their applications. Traditional approaches based on databases that have been designed for Grids of moderate size do not fit the requirements of wide-area large scale Grids composed of dozens of thousands highly dynamic nodes like the Internet desktop Grids. In this paper, we focus on Vigne fully distributed information service that exhibits the scalability and high availability desirable properties. We present the RW-OGS protocol enabling efficient resource discovery in large scale dynamic Grids. RW-OGS is a cached-based optimized random walk protocol over an unstructured overlay. It has been designed to reduce the network bandwidth consumption and the resource discovery latency. It is also optimized to increase the probability to discover nodes with free resources and to enable fast discovery of rare resources.

The paper is organized as follows. In Section 2, we present the requirements for a large scale Grid information service and related background. In Section 3 we describe the proposed RW-OGS resource discovery protocol. Experimental results are analyzed in Section 4. Finally we conclude in Section 5.

2. Grid Information Service

We define what is a Grid information service, give the requirements for an information service suitable for large scale dynamic Grids and present several existing approaches discussing their pros and cons in the context of large scale Grids.

2.1. Definition

The purpose of a Grid information service is to find resources to execute an application according to a user request. For instance, such a request may be: *"find 45 available nodes that have Opteron processors, at least 4Gb of memory, a 64 bits Linux based operating system and at least a 2.6.24 Linux kernel"*. An available node is defined as a

node having free resources to execute an application when the query is issued. In this paper, we only consider Grid nodes used in interactive mode such as computers directly accessible through their operating system interface. Reservation mechanisms will be considered in future works.

We can see that a user request may contain several attributes. An attribute may be static, like the processor type, or dynamic, like the availability. Furthermore, an attribute can have a scalar value, like the kind of operating system, or it can be defined by a range of values, as in the expression *"at least a 2.6.24 Linux kernel"*.

2.2. Requirements in a Large Scale Grid

In a large scale Grid, the information service should be able to record information concerning dozens of thousands heterogeneous resources. Furthermore, it should be able to provide up to date information despite the node volatility inherent to large scale wide area Grids. In fact, due to the application of system administration policies, maintenance operations or failures, a node may join and leave a Grid with an unpredictable frequency. However, users are only interested in resources able to execute their applications at the time they issue their queries. Finally, in a large scale Grid, some resources with specific characteristics may be rare. So, the information service must be able to find these resources among a large number of other resources if they are required by users.

2.3. Background

Several architectures have been proposed for building a Grid information service. We distinguish two main approaches: the server based approach and the fully distributed approach.

Server based approaches The simplest approach to create an information service is to use a centralized server. It is for instance the case in Condor [13] or BOINC [1] projects. The main advantage of this approach is the simplicity to find information since the complete information is on the server. However, this approach is neither scalable and nor fault tolerant.

To overcome the fault tolerance issue, some projects like PUNCH [12] propose to use replicated servers. However, the cost of replication may become discouraging if the scale of the Grid increases.

Finally, Globus [6] follows the hierarchical servers approach to provide a scalable information service called MDS. However, this kind of approach suffers from fault tolerance issues. Indeed, the loss of a node in the tree induces the loss of an entire subtree.

Fully distributed approaches To provide a high degree of scalability and fault tolerance required in large scale Grids, works like Iamnitchi et al. [8] propose to use the results of the peer-to-peer researches in Grids. Indeed, these fully distributed systems have intrinsic properties that are desirable to build large scale and fault tolerant distributed systems.

The first kind of approach consists in using unstructured overlay networks and a flooding based query propagation. This is the case for instance in Zorilla [5] and Vishwa [15] systems. One of the advantages of this approach is the ability to perform resource discovery with a high expressiveness. However, the searches are not exhaustive, their cost and duration may be highly depending on the resource discovery protocol used. So that may be an issue if rare resources are requested.

Another kind of approach relies on structured overlay networks, as in NodeWiz [2] system for instance. Searches can be exhaustive, especially by using distributed hash tables, that is suitable to find rare resources. However, structured overlays suffer from a lack of expressiveness for the queries. Indeed, queries with an unknown number of attributes and with attributes defined on a range of values cannot be easily handled. However, some projects in the peer-to-peer community are currently working on this issue [3].

3. Vigne Information Service Design Principles

In this section, we focus on the Vigne information service. First, we give an overview of application management in Vigne middleware. Then, we draw the concepts used to build the Vigne information service and we describe the RW-OGS resource discovery protocol.

3.1. Vigne Overview

Vigne is a Grid middleware designed to federate re-

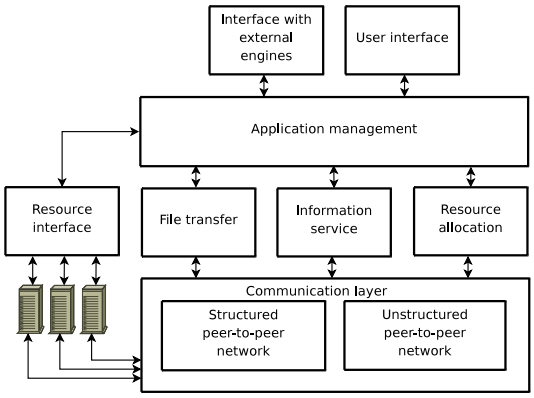


Figure 1. Architecture of Vigne Middleware

sources of large scale Grids [11], typically like the desktop Grids. Vigne architecture is shown in Figure 1. By using structured and unstructured overlays as a bottom layer in the system, several services can be built on top of it with unique properties like scalability and self-healing [14]. This is the case for instance for the communication service or the information service.

The Vigne application manager service offers advanced features for the execution of legacy distributed applications [10]. In particular it embeds a simplified workflow engine and co-allocation mechanisms that allow the allocation of close resources for communicating tasks. Thanks to a life-cycle manager, Vigne ensures that all submitted applications are reliably executed to the end despite possible failures in the Grid if enough resources are available.

In the Vigne system, the information service is used by the resource allocation process. Figure 2 shows the two steps process. First, the user provides the information service with a description of the resources needed. Then the information service has to find suitable resources in the Grid. Potentially, the information service finds more resources than required. The objective is to apply an allocation policy after the discovery. This is the aim of the resource allocation service. For instance, a user can specify a request like: *find 2 nodes with at least 2 Ghz clocked processors*. Then, an allocation policy can choose the nodes with the most powerful processors if more nodes than required have been found.

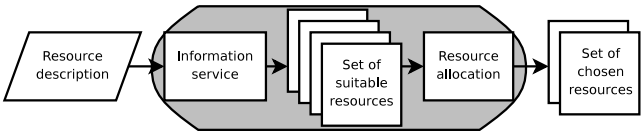


Figure 2. Resource allocation workflow process in Vigne

As far as the purpose is to allocate available nodes, the information service should return a set of suitable but also available nodes. Thus, for a given number of returned results (or discovered nodes that fulfill the requirements), the allocation will be more efficient if the number of interesting results (or available nodes) is high. Indeed, the allocation service is able to apply its policy among a larger choice.

In contrast to other traditional Grid middlewares, Vigne is based on a fully distributed scheduler and does not rely on any centralized part. The resource discovery requests are performed from any Grid node and each request is performed without paying attention to the other ones. Thus, the choice of a resource for a job is done by taking into account the information gathered on the nodes during a resource discovery. This best-effort approach is particularly suited for large scale Grids since it is impossible to design an optimal scheduler in a context where there are a lot of resources that have a dynamic behavior and where there are a lot of users distributed over all the Grid. Since we have presented the place of the information service in Vigne, the mechanisms used in the allocation service are out of the scope of this paper.

3.2. Random Walk on Top of an Unstructured Overlay Network

In order to record the static and dynamic information about Grid resources, it seems obvious to choose a distributed architecture for the information service, in particular if large scale Grids are targeted.

As stated in Section 2.3, the peer-to-peer based approaches have intrinsic properties that are desirable in a large scale context. Given our requirements, we choose to use an unstructured peer-to-peer network. Indeed, the unstructured overlays enable to make very expressive resource discovery queries that contain an undefined number of attributes and attributes defined on a range of values.

The use of a peer-to-peer overlay removes the constraints of having a central server that contains the values of the static and dynamic attributes for each node. Indeed, with this model every update of the node characteristics is reduced to a local operation and does not require extra communication towards a central server.

To construct the overlay, we choose to use a scalable and lightweight protocol. Scamp [7] has these characteristics and can dynamically construct an overlay where all the nodes have in average the same number of neighbors ($log(n)$ neighbors if there are n nodes). Furthermore, Scamp is easy to implement, that is why we choose it. Using this overlay, several methods can be used to perform resource discovery. To avoid wasting bandwidth with the traditional flooding of requests, random walks can be used.

With a random walk protocol, when a node receives a query, it propagates the query to one neighbor, randomly chosen, in contrast to flooding where a query is propagated to all the neighbors. As with flooding protocols, a query, in a random walk, is propagated from nodes to nodes until a given depth. To initialize the protocol, the number of desired results must be stated. If after a random walk, the number of results is lower than the number of desired results, another random walk is performed, and so on for a duration defined at the beginning. The main advantage of this protocol, with respect to flooding, is its lightness since the number of propagated messages is reduced. Furthermore, the bottleneck risk is limited for the receiver because during a random walk, the maximum number of responses that can be obtained in a short time is the depth of a random walk whereas with flooding the number of received responses is exponential.

The Scamp based overlay and the random walk protocol can be used as a keystone to build a Grid information service. Indeed, each resource in the Grid provide its own description and the resource discovery queries can be used to find suitable resources. In the Vigne system, the resource discovery queries are initiated by the application manager service. As far as this service is fully distributed (see [10]), the resource discovery queries are initiated from any Grid node and the mechanism used ensures that the distribution of the queries is regular over all the Grid nodes. This feature ensures scalability of the information service. Indeed, if the queries were launched by a centralized entity, a bottleneck would be observed in case of burst of query submission.

3.3. Improvement of a Cache-based Resource Discovery Protocol

To minimize the bandwidth cost of the resource discovery protocol, Iamnitchi et al. [8] have proposed an optimization of the random walk protocol by using caches. Thus, when a resource discovery is performed and when the results are obtained, resources references are put in a cache. If another query is submitted on the same node, the cached information can be used to reduce the number of random walks required to find enough resources. This optimization allows to reduce the bandwidth and the time used for resource discoveries.

However, this method suffers from three issues in our context. As far as the queries are initiated from any node in the Grid, it is quite uncommon that two queries are initiated from the same node, especially in a large scale Grid. Thus, the information stored in the caches may be useless. Furthermore, the information stored must be frequently refreshed to take into account volatility of Grid resources. Cache refreshment is also highly desirable if the information cached contains dynamic attributes such as node availability. Finally, the approach used in [8] does not offer any feature to improve the discovery of rare resources. This is a lack with the use of unstructured networks.

To deal with these issues we propose a new optimization of the cache based protocol for use in a large scale Grid context. We call RW-OGS (Random Walk Optimized for Grid Scheduling) this new protocol. Our contribution is based on three ideas.

First, we add a cooperation mechanism between the caches. This is based on a scattering mechanism that aims to improve the speed of cache filling. The cooperation also improves the freshness of the information cached since a node can inform other nodes that a resource has been allocated. Thus, dynamic attributes, like the date of allocation, in the cached data can be updated. To fit our requirements, each cache entry contains the description of node resources (ie. the static attributes) and the date of its last allocation.

Second, we use a cache policy that favors the discovery of available nodes. Thus, when a cache is full, the information about the resources that have been allocated the most recently is removed.

Third, we propose a mechanism to improve the discovery of rare resources. The idea is to provide a node with the capability to evaluate its rarity and to allow its self promotion in a pro-active way. To evaluate its rarity, a node periodically performs long random walks to retrieve a sample of the whole Grid nodes and compares its characteristics with the sample. If the rarity is detected, some long links are created in the overlay towards a set of randomly chosen nodes in order to improve the probability of discovering rare resources in random walks.

3.4. RW-OGS in Action

In this section we detail RW-OGS for a large scale highly dynamic Grid. RW-OGS improves a random walk based resource discovery protocol with cooperative caches. Our description is illustrated with an example depicted in Figure 3 (all the algorithms of this protocol omitted for space reasons can be found in [9]).

Figure 3.a shows an unstructured overlay network composed of 10 nodes represented by a circle. The name of each node is given in the upper part of the circles. Some lines between the nodes represent the connections in the overlay network. In the example, each connection is bidirectional but this is not required.

In Figure 3.b, node 3 initiates a resource discovery query. The purpose of this query is to find at least two nodes of a given type in order to choose the best one at the resource allocation step for the execution of a task. In the example, we assume that the depth of a random walk is 3. In this step, node 3 initiates a first random walk that is propagated to nodes 2, 1 and 4. On the walk, only node 1 is able to fulfill the query, and as a consequence it is the only one to respond to node 3. Thus, another random walk is performed through nodes 5, 7 and 9. On the walk, node 9 responds to the query. At this stage, enough answers have been received.

In Figure 3.c, we assume that the resource allocation service chooses node 1 to execute the task. We can see that node 3 records the information about nodes 1 and 9 in its cache (whose content is represented in the lower part of the circles). We can also notice that the allocation date of node 1 is recorded in the cache and illustrated by a rectangle around number 1 in the cache of node 3. Finally, the protocol broadcasts the information of the unused nodes (among discovered nodes) to the direct neighborhood of the query's initiator. So, node 3 broadcasts the information about node 9 to nodes 2, 4 and 5. We can see that the caches of nodes 2, 4 and 5 contain some information about node 9.

In Figure 3.d, node 5 initiates another resource discov-

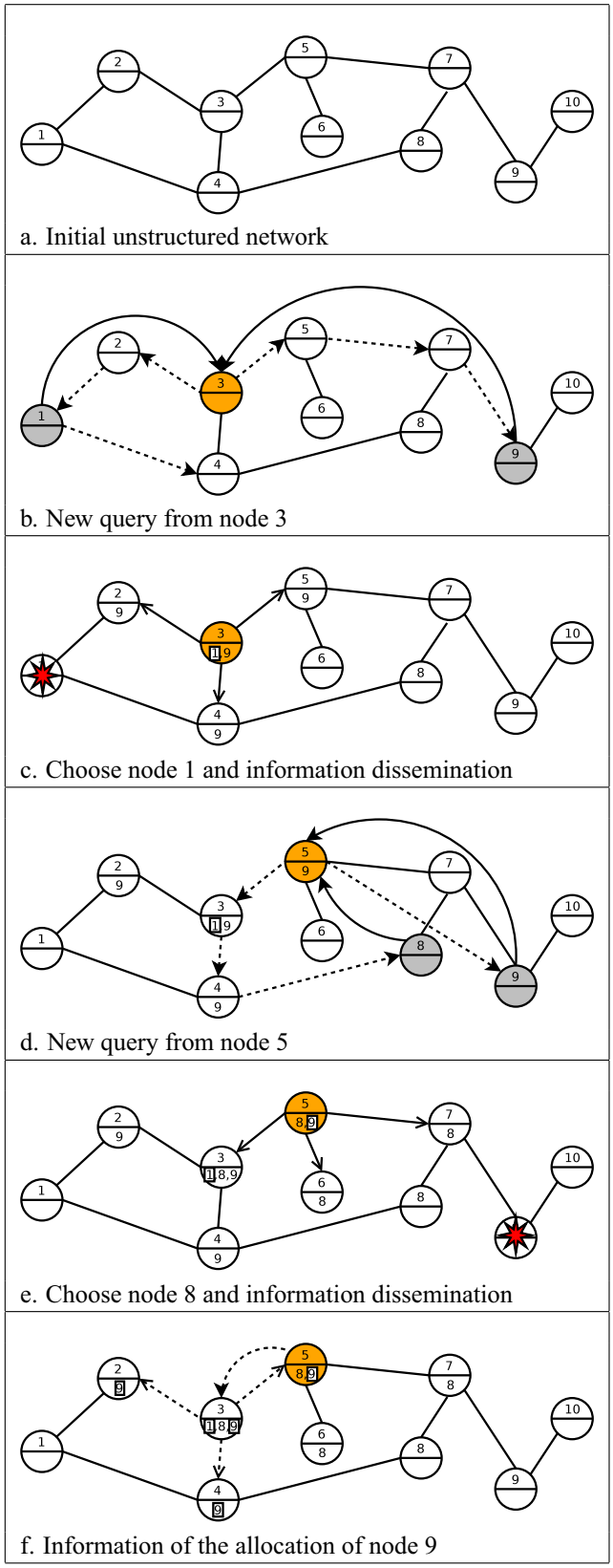


Figure 3. Exemplification of the RW-OGS protocol

ery, similar to the one previously initiated by node 3. First of all, it checks in its cache if it contains information about nodes that could fulfill the query. In the example, node 9 seems to fit. So, node 5 directly contacts node 9. Node 9 responds to 5 but at least two nodes are required. Thus, a random walk is initiated to complete the results obtained. The random walk is propagated through nodes 3, 4 and 8. Finally, node 8 also answers to the query and enough nodes are discovered.

In Figure 3.e, the resource allocation service chooses node 9. The information about the discovery is stored in the cache of node 5 and the information about the unused nodes (here, the information about node 8) are broadcasted to the neighbors of node 5.

Figure 3.f shows the last step of the protocol. As far as the node allocated by node 5 has been previously known on the bounce of a broadcast initiated by node 3 (see Figure 3.c), node 5 informs node 3 of the allocation of node 9. Then, node 3 records this allocation date in its cache and informs its neighbors. Thus, if a query is initiated from a node that contains information about allocated and non allocated nodes, it will firstly query the most recently allocated nodes. Furthermore, if any cache becomes full, the cache policy removes the most recently allocated nodes. As far as several resource discovery can be performed simultaneously (since the scheduling is fully distributed), we use a locking mechanism that prevents a node from answering several queries in a given time for the same resources.

3.5. Improving the Discovery of Rare Resources

Finding rare resources (ie. resources with rare static characteristics like an exotic CPU) in a Grid is simple with a centralized information service. However it is more difficult with a fully distributed one. Our proposition divides the problem in two parts: evaluation of the rarity of a resource and improvement of the knowledge of rare resources.

To evaluate the rarity of a resource we assume that a kind of resource that cannot be easily found is rare. We also suppose that a resource that can frequently respond to the resource discovery queries without being often allocated is not considered as a rare resource. So we propose that each node performs periodically a long random walk in the network to find resources with its own static properties. If the proportion of identical resources is very low, the resource can be considered as rare. The frequency of evaluation can be tuned based on a rarity coefficient. For instance, if a resource can often respond to queries, this resource knows that it is not rare, thus it is not necessary to evaluate its rarity using random walks because its rarity coefficient is low. However, this coefficient is modified if the resource has not been able to respond to queries for a long time considering

only its static characteristics.

Once a coefficient of rarity is known on each node, the nodes with a low coefficient make pro-active publicity towards other nodes in the network. We propose to use again long random walks in order to inform some other nodes in the overlay. The idea is to add long links from randomly chosen nodes towards the rare nodes in order to increase the degree of connectivity of the rare nodes. As a consequence, these rare nodes can be found easily with random walks.

4. Evaluation

In this section we briefly present the prototype used to evaluate our information service based on the RW-OGS protocol and we show the early results obtained.

4.1. Prototype

A working prototype that implements the architecture presented in Figure 1 has been written in C language (about 30,000 lines of code) and runs on top of Linux based system.

The prototype can be used in real execution mode where a daemon is executed on each node of the Grid or in simulation mode where all the nodes are simulated on a single computer. All the prototype code is identical for both execution modes except the low level communication layer. In real execution mode, the low communication layer uses sockets whereas it uses a discrete event simulator in simulation mode.

The simulation mode allows to perform reproducible experiments. This is very useful to thoroughly compare several protocols because the random failures of the Grid are discarded. As a consequence, we choose this execution mode for the evaluation of the Vigne information service. Vigne application management service has also been experimented in real execution mode with a large number of nodes as reported in [9, 10, 11].

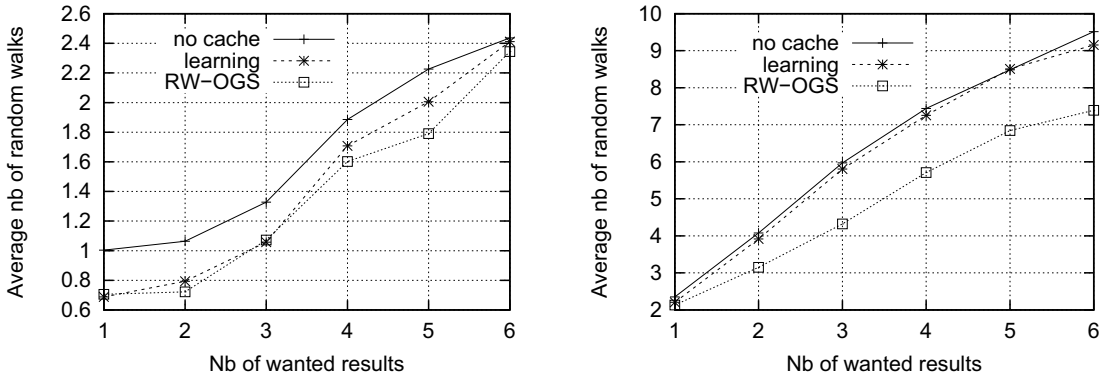
4.2. Results

In the following experiments, we compare our optimized protocol with the basic random walk protocol that does not use cache and with a protocol presented in [8] that uses random walks with a learning cache policy. The first protocol has been chosen since it is the simplest one and since it already provides better performance, in terms of bandwidth consumption and time of resource discovery, than the traditional flooding protocol. The second protocol has been chosen since it is a well known protocol in the state of the art and since it also uses cache to reduce the bandwidth consumed. To perform the comparison, these two protocols have been implemented in Vigne.

In all experiments, we consider a network of 500 nodes. Several parameters of the protocols have been fixed like: the depth of the random walks is 6, the maximum number of entries in a cache for the protocols based on a cache is 200, the maximum duration for a resource discovery is 60 VTU (Virtual Time Unit in our simulator). In Figure 4, we evaluate the impact of the resource rarity on the bandwidth cost of each protocol. We have executed 500 tasks of a given load, each task requiring one node. A task is submitted to the system every 2 VTU. We vary the number of desired results for each resource discovery query from 1 to 6. We also vary the number of resource types. In Figure 4.a, all nodes have the same characteristics whereas in Figure 4.b, there are 8 node types. In the experiment with 8 types of resources, there are 8 types of query and about $\frac{500}{8}$ queries per type. In all these experiments we measure the average number of random walks required to find the desired number of resources. This number gives an approximate idea of the bandwidth consumed. We can observe in Figure 4.a and 4.b that RW-OGS is consistently equal or better than the two other ones. When the discovery is easy (ie. when there are a lot of available resources), the learning protocol obtains good results since it can save about 30% bandwidth comparing to the traditional protocol. However, RW-OGS protocol is more efficient than the two other ones when the resources are rare (see Figure 4.b). It can save more than 20% bandwidth compared to the learning protocol.

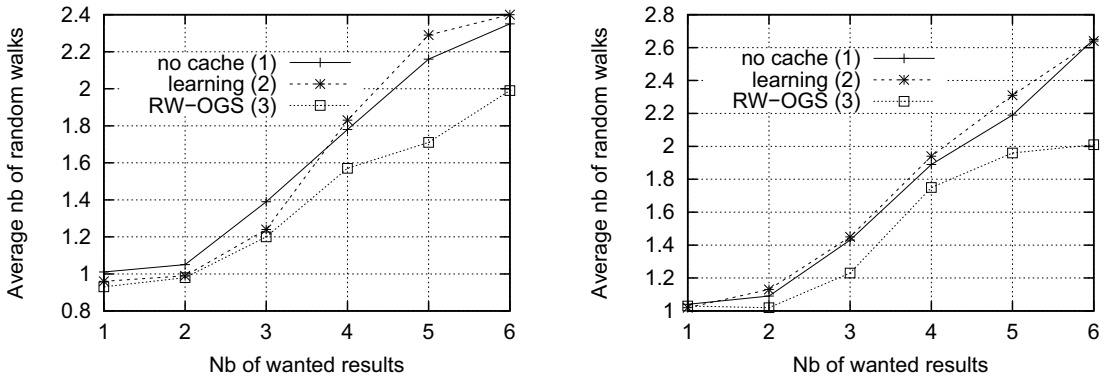
In the second experiment, we vary the task load and we measure the bandwidth consumed. In Figure 5.a, each task contains 100 instructions and it contains 500 instructions in Figure 5.b. By increasing the load of each task, we decrease the number of free resources in the Grid, thus the discovery of available nodes is harder. We can see in Figure 5 that RW-OGS is again equal or better than the two other ones. In particular, its efficiency can be observed in hard conditions, when the system must find 6 resources each time whereas the number of available resources is low. Indeed, in these conditions, RW-OGS saves about 25% more bandwidth than the two other protocols.

In the last experiment, we vary the number of tasks submitted to the system and we measure the average time for the resource discoveries. In Figure 6.a, we execute 50 tasks and in Figure 6.b, we execute 500 tasks. Like in the previous experiment, the idea is to increase the load on the Grid and to see the impact on the three protocols. When only 50 tasks are launched, the three protocols obtain similar results. In all cases, the discovery of available resources is easy since only 10% of the nodes are busy. However, when we execute 500 tasks and more, when the conditions become harder, RW-OGS allows to perform resource discovery about 20% faster than the two other protocols. Furthermore, the resource discovery latency with RW-OGS is lower when we execute 500 tasks than when we execute only 50



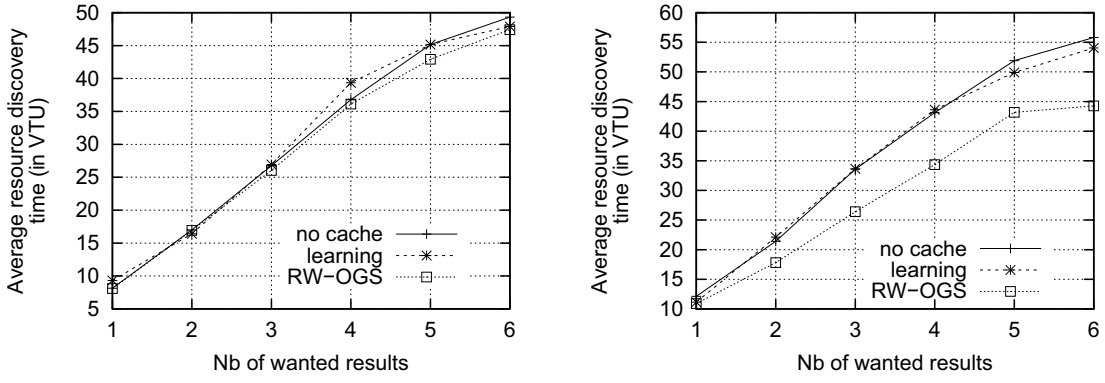
a. 1 type of resource b. 8 types of resource

Figure 4. Impact of the resource rarity on the bandwidth cost



a. Workload of 100 instructions per task b. Workload of 500 instructions per task

Figure 5. Impact of the task's load on the bandwidth cost



a. Execution of 50 tasks b. Execution of 500 tasks

Figure 6. Impact of the number of tasks in the Grid on resource discovery latency

tasks. This is due to the fact that the caches are statistically less efficient when a small number of queries are executed in the system.

As a conclusion, we can see through these experiments that our information system based on RW-OGS provides better results in terms of bandwidth saving and latency of resource discovery than the two other protocols used for the comparison. In particular, RW-OGS obtains particularly good results when the conditions of discovery are hard (small number of available resources). Thus, this protocol and our architecture based on an unstructured peer-to-peer overlay constitute an interesting solution to build a Grid information service that has several valuable properties like scalability, self-healing and self-organizing that cannot be obtained with centralized solutions.

5. Conclusion and Future Work

The contribution of this paper is the design, implementation and experimental evaluation of the RW-OGS resource discovery protocol designed for large scale dynamic Grids. RW-OGS that is part of Vigne middleware is an optimized random walk protocol over an unstructured overlay. RW-OGS relies on the use of cooperative caches that apply a policy favoring the discovery of non allocated resources. This novel approach improves some existing work (see [8]) that attempts to take advantage in the Grid community of results from the peer-to-peer community. Hence, RW-OGS protocol contributes to the architecture of an innovative fully distributed middleware that does not use any centralized scheduler. The choice of peer-to-peer technologies is deemed to be appropriate to deal with the increasing scale of Grids and their node volatility.

We have implemented a fully operational prototype that has been used to evaluate the RW-OGS protocol in a large scale Grid. The early results obtained show that RW-OGS can save significant bandwidth and reduce the network bandwidth consumption in comparison with other existing protocols. Furthermore, the experiments show that the RW-OGS protocol behaves well in hard conditions, in particular when there are only few available resources and when rare resources are queried.

Future works will be dedicated to the improvement of the protocol by taking into account reservation mechanisms on Grid nodes.

References

[1] D. P. Anderson. Boinc: A system for public-resource computing and storage. In *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*, pages 4–10. IEEE Computer Society, 2004.

[2] S. Basu, S. Banerjee, P. Sharma, and S.-J. Lee. Nodewiz: peer-to-peer resource discovery for grids. In *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid 2005 (CCGrid 2005)*, pages 213–220, Cardiff, UK, May 2005.

[3] O. Beaumont, A.-M. Kermarrec, L. Marchal, and E. Rivière. Voronet: a scalable object network based on voronoi tessellations. In *Proceedings of 21st IEEE International Parallel & Distributed Processing Symposium (IPDPS 2007)*, pages 1–10, Long Beach (CA), USA, 2007.

[4] F. Cappello, S. Djilali, G. Fedak, T. Herault, F. Magniette, V. Nri, and O. Lodygensky. Computing on large-scale distributed systems: Xtremweb architecture, programming models, security, tests and convergence with grid. *Future Generation Computer Systems*, 21:417–437, 2005.

[5] N. Drost, R. V. van NieuwPoort, and H. E. Bal. Simple locality-aware co-allocation in peer-to-peer supercomputing. In *Proceedings of the Sixth International Workshop on Global and Peer-2-Peer Computing (GP2P)*, volume 2, page 14, Singapore, May 2006.

[6] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, an Francisco, CA, USA, 1st edition, July 1998.

[7] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulié. Peer-to-peer membership management for gossip-based protocols. *IEEE Transactions on Computers*, 52(2):139–149, February 2003.

[8] A. Iamnitchi and I. T. Foster. On fully decentralized resource discovery in grid environments. In *Proceedings of the Second International Workshop on Grid Computing (Grid'01)*, pages 51–62, London, UK, 2001. Springer-Verlag.

[9] E. Jeanvoine. *Intergiciel pour l'exécution efficace et fiable d'applications distribuées dans des grilles dynamiques de très grande taille*. Phd thesis, University of Rennes 1, IRISA, Rennes, France, November 2007. In french.

[10] E. Jeanvoine, C. Morin, and D. Leprince. Vigne: Executing easily and efficiently a wide range of distributed applications in grids. In *Proceedings of Euro-Par 2007*, volume 4641 of *Lecture Notes in Computer Science*, pages 384–393, Rennes, France, August 2007. Springer.

[11] E. Jeanvoine, L. Rilling, C. Morin, and D. Leprince. Using overlay networks to build operating system services for large scale grids. *Scalable Computing: Practice and Experience*, 8(3):229–239, 2007.

[12] N. H. Kapadia and J. A. B. Fortes. Punch: An architecture for web-enabled wide-area network-computing. *Cluster Computing*, 2(2):153–164, 1999.

[13] M. Litzkow, M. Livny, and M. Mutka. Condor - a hunter of idle workstations. In *8th International Conference of Distributed Computing Systems*, June 1988.

[14] L. Rilling. Vigne: Towards a self-healing grid operating system. In *Proceedings of Euro-Par 2006*, volume 4128 of *Lecture Notes in Computer Science*, pages 437–447, Dresden, Germany, August 2006. Springer.

[15] G. T. Venkateswara Reddy M., Vijay Srinivas A. and J. D. Vishwa: A reconfigurable peer-to-peer middleware for grid computations. In *Proceedings of the 35th International Conference on Parallel Processing*, pages 381–390, Ohio, USA, August 2006. IEEE Computer Society.