Using ontology for resources matchmaking in grid middleware.

Aurélie Hurault and Kento Aida

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
hurault@grid.nii.ac.jp and aida@nii.ac.jp

Abstract. The globalization of resources on grids introduces the need of matchmaking mechanisms. A matchmaker finds the set of resources that match the user requirements and gives them back to the user. In this paper, we propose an ontology-based matchmaker. The user requirements and the resources are described thanks to the ontology. The user requirements are expressed as a complex workflow (composed with jobs in sequence, parallel, co-allocated, ...). Each single job requirements can specify the number of CPU, the amount of memory, the kind of operating system, CPU architecture, ...wanted and some time constraints. The resource's description allows to specify the physical characteristics of the system (number of CPU, memory, network properties, ...) but also the applications available and the access policies. The matchmaker takes this information into account to find the systems that match the user requirements and that the user can use. The conditions of use are also given back.

1 Introduction

The globalization of resources (data, systems or services) thanks to the Grid [FK98] introduces new needs. Some mechanisms to allow finding the resources that suit user requirements are needed. In the case of computational grid, the grid allows users to run their applications on different systems in different geographic places, with different owners and policies, different characteristics (CPU architecture, operating system, memory, the number of CPU, ...), ... The user may need a specific kind of resource for his/her application. The operation to find resources that match the user requirements is called *matchmaking*.

To find the resources that match the user requirements, both resources and user requirements have to be described and to be compared. Traditional resource matching mechanisms are based on an attribute description as in the Condor matchmaker [RLS98]. This mechanism imposes that resources owners and users agree on attributes and values. This makes an extension of the description difficult.

Ontologies allow defining more meaningful and flexible descriptions. They are mature enough to be used in application domains.

Some works have been done to use ontologies for matchmaking. Tangmunarunkit and al. [TDK03] propose a description based on ontologies (OWL) and rules (TRIPLE [SD02]). While matchmaking on the Grid needs multiple ontologies, e.g. that for resources, requests and policies, the work implements only one policy to authorize a user to use a system. The rules are used to add some information about the domain, for example, «a Linux system is compatible with a SunOs system». TRIPLE/XSD is used as a deductive database system. While the work shows interesting results, it has some problem for scalability and practicality. For example, the user needs to be familiar with TRIPLE to add a new OS type. The representation of policies is poor (only one policy), and it assumes that the user and the owner agree on some points, such as the unit of the values.

The work presented in [SBK+06] uses only ontology for matchmaking. The work uses Algernon[1] as an inference engine. However, Algernon is suitable for small to medium-size expert systems, which can be a problem for the scalability. Moreover, Algernon does not seem to be maintained and will not take into account the new features on ontology that may be interesting for some later evolutions. The ontology they define is also poor and does not take into account all the possibilities of the standard like CIM for resource description. They do not have the notions of user, group and policy.

More recently, the work in [KPPN07] proposes an extension of [TDK03] to define systems, storages and networks; however, the work does not discuss requirements using ontology and does not propose a matchmaking algorithm. Some other works also use ontology on the Grid, but for other purposes. For example, the work in [LvS02] discusses to use ontology for service discovery on the Grid.

In this paper, we discuss matchmaking using a description based on semantic web technologies and in particular ontologies. Resources on the Grid will be diverse and user requirements will be more complex on practical grids, or production grids. The existing work described above still has problems in scalability and practicality. We propose a scalable and practical system to use ontology for matchmaking. The proposed system enables complex and diverse descriptions of user requirements and resources on the production grid, e.g. complex workflow and co-allocation. Extension of description, e.g. OS types, is easily done.

In the proposed system, the ontology is developed using Protégé [2] [KFNM04] and the OWL standard[3]. The system is implemented in the NAREGI Grid middleware[4] [MIU06]. While the current description rules follow the expressions in the NAREGI Grid middleware, it can be integrated to other Grid middleware. The matchmaking algorithm is also developed and the demonstrations are presented in this paper.

The rest of the paper is organized as follows: Section 2 and 3 present the ontologies for user requirements and resources, respectively. Section 4 describes

[1] http://algernon-j.sourceforge.net/
[2] `http://protege.stanford.edu/`
[3] `http://www.w3.org/TR/owl-features/`
[4] `http://www.naregi.org/`

the matchmaking algorithm and Section 5 shows the demonstrations. Section 6 concludes the work and outlines the future work.

2 User requirements

In the NAREGI project, the jobs are submitted using the workflow tool, that allows designing complex workflows (sequence, parallel, co-allocated, conditional, loop, ...) as request. The simplest workflow is a single job described using JSDL [ABD+05]. In the ontology, we find a class «Request» which is composed of a workflow and a user and represents the user request. This request will then be compared with the resources to find the ones which match the user requirements.

The figure 1 shows the ontology classes for the request. They will be detailed in the following.

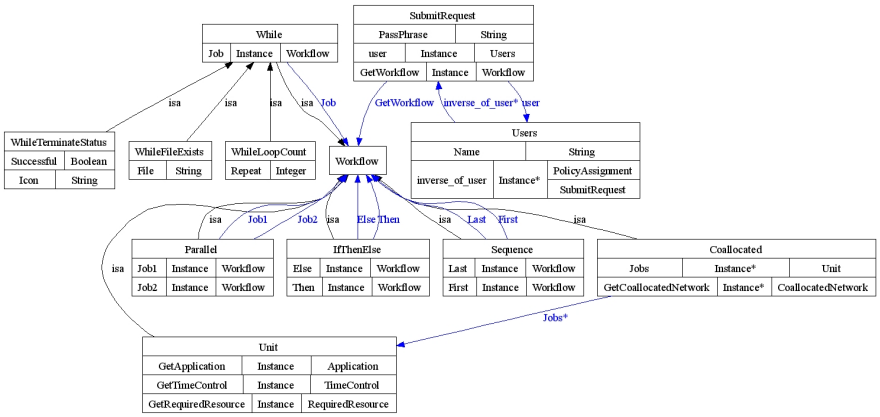

Fig. 1. Request

2.1 Complex workflow

The workflow in the request can be of any kind of the ones allowed on the NAREGI project. We will find one class for parallel jobs, sequential jobs, a conditional branch, a loop and co-allocated jobs.

For co-allocated jobs, it is possible to specify n single jobs and m networks required between the resources allocated for some jobs. The latter is not currently supported in the NAREGI Grid middleware and allows specifying some jobs as shown in the figure 2.

2.2 Single job

In the NAREGI Grid middleware, the simplest jobs are expressed using the JSDL standard. So, the ontology can describe all information that can be found in a JSDL description:

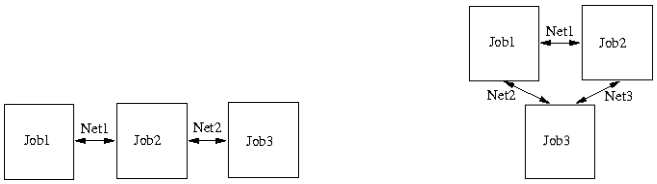

Fig. 2. Co-allocated jobs, with network specifications

- the application part: executable
- the resource part: candidate hosts (list of the hosts that can be allocated), operating system, CPU architecture, individual CPU count, total CPU count, total physical memory, total virtual memory, ...

It is also possible to specify the network, defined by its bandwidth and its latency, required for a cluster (new functionality).

Some time information can also be given (new functionality):

- BeginTime: the date after which the job must start;
- EndTime: the date before which the job must finish;
- Duration: the time the resource is reserved.

The time information are represented using the OWL Time ontology[5] [HP04].

Constraints and units In the JSDL standard, the properties with «amount requirement» are given with the type «RangeValue_Type», that allows expressing complex constraints (lower bound, upper bound, range, set of constraints, ...). In the ontology there is a class «Constraint» that has the same expressiveness.

The problem is that the user and the owner of the resource have to agree on the units the values are given in. If one uses Gbps and the other Bps, the results will not be the ones expected. To solve this problem, some units descriptions are proposed and should be specified both in the requests and resources. So the constraints are combined with a unit (see fig. 3). The ontology can describe the classes as follows:

- «UnitDescrition»: super class of all the unit classes;
- «BDescription»: for bytes;
- «BpsDescription»: for bits per second.

Request parser Once a job is submitted, via the NAREGI workflow tool, the request is «translated» into an XML file that respects the NAREGI-WFML schema. A parser is available to convert an XML document that respects the NAREGI-WFML schema into an instance of the class «Request» of the ontology. The NAREGI-WFLM schema has been extended to take into account the new functionalities.

[5] `http://www.w3.org/TR/owl-time/`

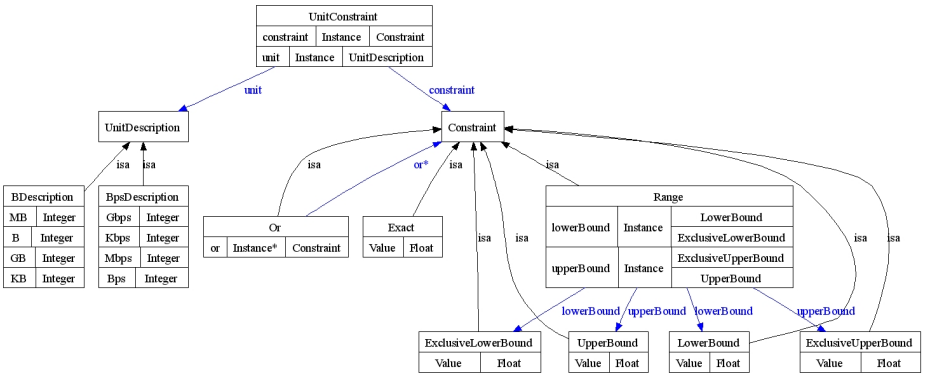

Fig. 3. Unit description

3 Resource

In the Naregi Grid middleware, resources are described using a schema based on the CIM schema[6] [DMT99] with some extensions [Ltd07]. The ontology is defined based on this description with some improvements.

3.1 Mapping the NAREGI schema into an OWL ontology

The technique used to construct the ontology is the one explained in [QAW+04], for an OWL ontology.

Most of NAREGI schema classes are mapped into classes in the ontology, except for association and aggregation classes, which are mapped into object properties. Most of the NAREGI schema properties are mapped into data type properties.

The ontology created contains some classes that are not used in the matchmaker algorithm. But they are included if needed later for some other purpose.

Some properties are not mapped exactly as described, to take advantage of ontology specificities, or to be more precise, meaningful and flexible. They will be discussed in the following sections.

3.2 Cluster

It is possible to give a complete description of a cluster. The cluster is described by nodes groups («NodeGroup»), with same properties («NodeType») (see fig. 4). The properties of the groups are the OS, the processors installed and the number of CPU. The number of nodes of each node group is also given. So, it is possible to define heterogeneous clusters with different kinds of nodes groups.

[6] `http://www.dmtf.org/standards/cim/`

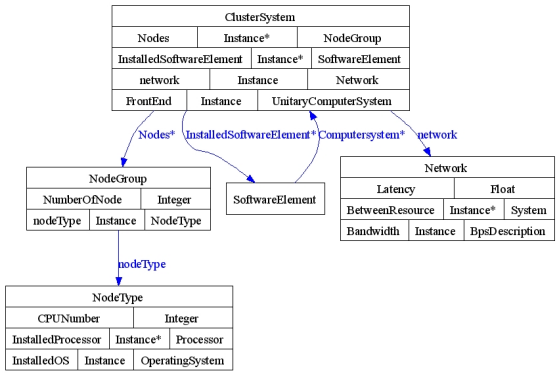

Fig. 4. Cluster

3.3 OS type and processor family

In the NAREGI schema, the OS types and the processor families are represented using an integer, and each integer represents a specific OS or family. This representation is not meaningful and is not really flexible, since it is difficult to add new OS or processors; however they often have new products.

So the representation has been changed. A class «OStype» (resp. «ProcessorFamily») is created to represent the OS types (resp. the processor families). For each type (resp. family) there is a subclass of «OStype» (resp. «ProcessorFamily»). The name of the class is more relevant and meaningful than an integer and to add a new product, a class is added in the ontology.

Moreover, it is possible to have a hierarchy in the OS types (resp. processor families), this allows to create families: Unix (with subclasses: Linux, SunOS, ...), Windows, ... (resp. Intel (with subclasses: Celeron, pentium IV, ...), IBM, Power PC, ...). One consequence is that it is possible to specify that we want an OS of type Unix and the matchmaker will consider the resources with Linux, Sun OS, ... This was not possible easily before. No change is needed when a new product is added. It just needs to be well positioned in the hierarchy.

3.4 Default software

In the ontology, it is possible to define software installed on a system (comes from CIM schema). This is a way to check if the application that the user wants to run is installed on the system. But, there are some applications that are installed by default on a given OS type, e.g. ls, hostname, ... on Unix systems.

In the ontology, it is possible to add a list of software installed by default on systems that have a given OS. This is done thanks to a property at the class level. Currently, some are added for Unix and Windows, but it is easy to add some more, and no change will be needed in the matchmaker algorithm.

3.5 Planning

For each system a planning of reservations is given. This is needed to check the time constraints of the request. This will allow giving the time intervals in which the job must start to valid the time constraints.

3.6 Users and policies

On a system, there often are different users with different permissions. Some will be limited in time (trial version for example), some in the resources used (number of CPU, memory ...). These policies, which define the use of a system for users, may depend of the group of the user. For example in a university a student may not have the same permissions as a professor, but all the students will have the same.

To represent this functionality, a class «Users» has been added which can be a «Group» or a «User» and each «Users» can be part of one or several «Group». A class «Policy» is defined to represent the access constraints on the target system. Some subclasses («AuthorizedMemberPolicy», «CPUPerHourPolicy», ...) are added to define particular policies. For each system there is a set of «PolicyAssignment» that assign a «Users» to a «Policy». These classes are presented in the figure 5 (there are more policies implemented than shown in the figure). It is possible to have several «PolicyAssignment» for a «Users» on the same system (one for memory, one for CPU and one for time, for example). In this case all the constraints must be satisfied.

A user can be part of different groups that have different policies on a system. For example a professor «M. X» is part of the group «professor» which is part of the group «university personal». If members of «university personal» are allowed to use 1GB memory; members of «professor» are allowed to use 16 CPU/hour and «M. X» is allowed to use 32 CPU/hour; «M. X» will have to use less than 1GB of memory and 32 CPU/hour. So he has to valid all the constraints of his groups except the ones that are replaced by another of a subgroup.

So, we have to know if a policy can replace others. At the class level, a property «canReplace» (symmetric) has been added, to express that all instances of a policy can replace one of others.

If a user is part of several groups, with no subgroup relation, the user will have to valid one of the sets of policies.

3.7 Units

The properties which need values with units are described using the unit description and not an integer or float. For example the property «Bandwidth» of the class «Network» is expressed with an instance of the class «BpsDescription». The results are given back with the units in which there are computed.

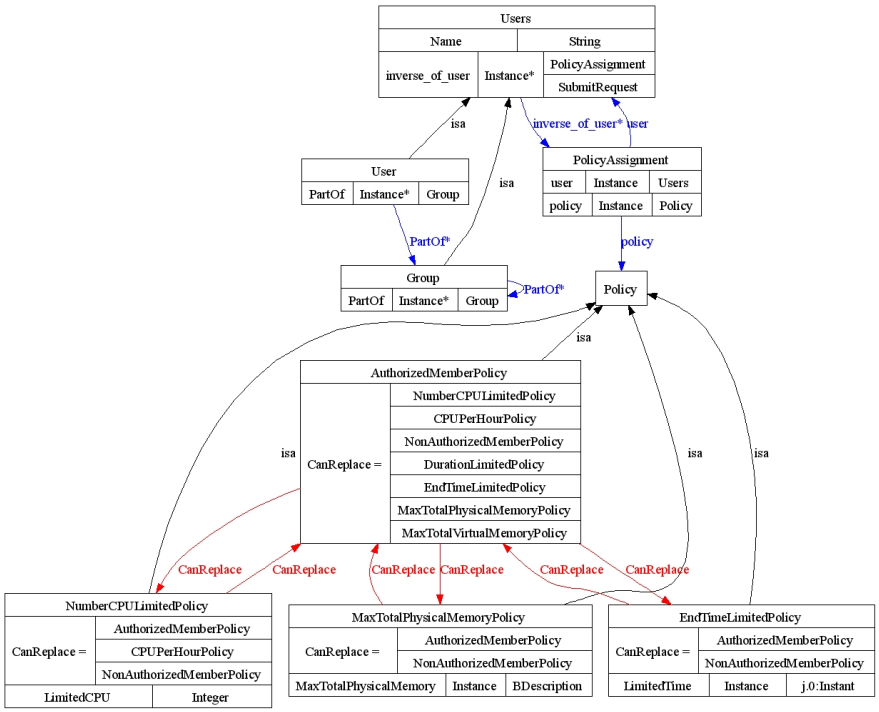

Fig. 5. Users and policies

3.8 Data extraction from a database

It is possible to extract some data from an existing database that respects the NAREGI resource description schema specification [Ltd07] and to convert them into an instance of a class of the ontology.

Currently only a part of the database is extracted, the one that is used in the matchmaker algorithm. If more is needed, this part should be extended. As the ontology is «mapped» on the database, it can be done easily.

4 Matchmaking algorithm

The matchmaking algorithm compares the user constraints (application, resource requirements, time, ...) to all the available resources to find the ones he/she is allowed to use and that match his/her needs. The time intervals in which the single jobs must start, to respect the time constraints, are also given back.

4.1 Complex workflow

If the workflow associated to the request is:

- composed of two workflows in parallel (or condition), the solutions are the combinations of the solutions of both workflows;
- composed of two workflows in sequence, the solutions are the combinations of the solutions of both workflows, with a new time constraint: the second workflow must start after the end of the first one;
- a loop, the solutions are the solutions for the workflow in the loop with the time intervals for each loop (when the number of loops is known);
- a co-allocated workflow, the solutions are the combinations of the solutions for each jobs. Only the solutions that respect the networks requirements and with a non-null intersection of time intervals are kept.
- a single job, the solutions are the systems that match the policies, application, resources and time constraints.

4.2 Single job

As just said, for single jobs, four points are checked:

- the non-violation, by the user, of the system policies;
- the installation of the application on the system;
- the match of the user resource requirements;
- the time constraints.

User and policies First the set of policies that specifies the conditions of use of the resource by the user is computed. To compute this set, the specific policies for the user are added, and then ones of the direct super-group, but only ones that are not overwritten by one specific to the user. And the set of policies is constructed this way until there are no more groups the user belongs to. If the user belongs to different groups with no subclass relation between them, different sets of policies are generated and the user must validate at least one of them.

These policies are then checked. There is two possibilities:

- it is possible to know if the user is allowed to use the resource or not (AuthorizedMemberPolicy or NonAuthorizedMemberPolicy for example) and the resource is selected for the next step or is rejected;
- the policy imposes more constraints in the use of the resources. A set of new constraints is generated, and will be compared, in the next steps, with the resource properties and the user requirements.

Application The matchmaker checks if the user application is installed on the system (see 3.4), if not, the system does not match the request and will not be part of the solution.
Some users may want to allocate some resources and deploy their own environment before running a job. In this case, matchmaking omits to check availability of applications installed on a system.

Resource requirements Matchmaking for resource requirements of «UnitaryComputerSystem» is different from that of «ClusterSystem». It is possible to partially allocate nodes on a cluster; however, on a unitary computer, whole resources (CPUs) should be allocated[7].

Matchmaking is performed as follows: for a cluster, information of the nodes is checked in the first step and only the ones that match are kept. Second, the number of available nodes requested in the requirements and the total information are checked. For unitary computer system, matchmaking is performed in more simple way, that is, information of the system is checked.

First the **candidate hosts** are checked. If the name of the resource is not member of the non-empty list of wanted hosts, the resource is rejected.

Then the **«first category» properties** are checked. Among these properties we can find requirements about *operating system*. The version of the operating system requested and those on the system are compared. The OS types are also compared. If the class of OS requested is the same as OS on the system or a super-class, the system matches the requirements. The same mechanism is used for the *CPU architecture* and the processor family.

The amount of *individual free physical memory* is then computed and converted into the unit the constraints are expressed. Once the conversion done, the value on the system is compared to the range of constraints to determine if the system matches or not. The same mechanism is used for *individual free virtual memory*.

Finally the **«second category» properties** are checked to select or reject the system. The *total number of CPU*, *total free physical memory* and *total free virtual memory* are computed, with unit conversion if needed. They are then compared to the range of constraints to determine if the system matches or not. At the end, the *network* of the cluster is compared to the bandwidth and the latency expected by the user. This comparison is done after unit conversion.

Time The last step to check resources is time constraints or availability of resources for requested time duration. The following will concern the unitary computer system, some points have to be modified for clusters. If there is no begin time specified by the user, we suppose that it is as soon as possible. If there is no end time, we suppose that it is not critical and fix it to the default end time[8]. The duration of the reservation has to be given. The time intervals in which the job must start to respect the time constraints are given, taking into account the reservations already done on the resource. It is also possible to run the algorithm without taking time into consideration.

[7] An example of a unitary computer system is a SMP computer or node. We assume that all CPUs in a unitary computer are allocated to a single application to improve performance.

[8] In the current implementation, we set the default end time to ten years, but it can be modified.

5 Demonstration

5.1 OS type and processor family

We assume the following «UnitaryComputerSystem»:

Name	OS	Processor
simpleLinuxPentium4	Linux	Intel Pentium 4
simpleLinuxCeleron	Linux	Intel Celeron
simpleSolarisPentium4	Solaris	Intel Pentium 4
simpleSolarisSparc	Solaris	Sparc
simpleWindowsPentium4	Windows	Intel Pentium 4

When the user asks for a system with:

- a Unix OS, the algorithm finds simpleLinuxPentium4, simpleLinuxCeleron, simpleSolarisPentium4 and SolarisSparc.
- a processor architecture Intel, the algorithm finds simpleLinuxPentium4, simpleLinuxCeleron, simpleSolarisPentium4 and simpleWindowsPentium4.
- a Unix OS and a processor architecture Intel, the algorithm finds simpleLinuxPentium4, simpleLinuxCeleron and simpleSolarisPentium4.

In the ontology, the classes «Solaris» and «Linux» are subclasses of «Unix», but «Windows» is not. So, when the matchmaker looks for the Unix like OS, Solaris and Linux will match, but Windows does not.

It is the same for processor family. In the ontology «Celeron» and «Pentium_4» are subclasses of «Intel», but «Sparc» is not.

5.2 Time

We assume the following «UnitaryComputerSystem»:

- Name: vcp01.naregi.org
- Planning: Jobs reservations on
  - [Oct 1, 2007 12:00:00 AM ; Oct 2, 2007 12:00:00 AM]
  - [Oct 5, 2007 10:00:00 AM ; Oct 5, 2007 3:00:00 PM]
  - [Oct 6, 2007 12:00:00 PM ; Oct 7, 2007 12:00:00 AM]

When the user asks for two systems for running two jobs in sequence, half an hour each, the algorithm finds

- Job1: vcp01.naregi.org with the time intervals:
  - [Oct 2, 2007 12:00:00 AM ; Oct 5, 2007 9:00:00 AM]
  - [Oct 5, 2007 3:00:00 PM, ; Oct 6, 2007 11:00:00 AM]
  - [Oct 7, 2007 12:00:00 AM ; Aug 26, 2017 3:47:37 PM]
- Job2: vcp01.naregi.org with the time intervals:
  - [Oct 2, 2007 1:00:00 AM ; Oct 5, 2007 9:00:00 AM]
  - [Oct 5, 2007 3:00:00 PM ; Oct 6, 2007 11:00:00 AM]
  - [Oct 7, 2007 12:00:00 AM ; Aug 26, 2017 3:47:41 PM]

Time intervals for the second job are given assuming that the first job will begin as soon as possible.

5.3 Co-allocated jobs

We assume that there are one station in Tokyo, one in Nagoya, one in Osaka and one in Kobe. The SuperSINET network (10 Gbps) links Tokyo, Osaka and Nagoya. Kobe is linked to other towns by the SINET network (1 Gbps).

When the user asks for three resources for co-allocated jobs and a fast network between them (8 Gbps), the algorithm finds all the combinations of the stations in Tokyo, Nagoya and Osaka for the jobs.

When a slow network (500 Mbps) is required, the algorithm finds all the combinations of the stations in Tokyo, Nagoya, Osaka and Kobe for the jobs. The demonstration also illustrates the unit functionality. In the request, we find 500 Mbps and for the networks descriptions: 10 Gbps and 1 Gbps. The values compared are not 500, 10 and 1 (in this case, no solution is found), but the values and their units. So before the comparison, 10 Gbps and 1Gbps are converted into their values in Mbps.

When a slow network is required between job#3 and the other jobs and a fast one between the two others, the algorithm finds all the combinations of the stations in Tokyo, Nagoya and Osaka for the job#1 and job#2, and all the stations for the job#3.

5.4 Groups and policies

We assume the following cluster

- Myrinet network
- 50 nodes (bi-processor Opteron and 2GB of RAM)

We assume that we are in the context of a «commercial» cluster and there are three groups of users:

- «TrialVersionUsers» allow to use 4 CPU and 2 GB of memory for one month;
- «SimplePackUsers» allow to use 16 CPU and 10 GB of memory;
- «DoublePackUsers» allow to use 32 CPU and 20 GB of memory.

There also are some users registered:

- a user «TrialVersionUser1» that purchases the «trial version» the 01/09/2007;
- a user «TrialVersionUser2» that purchases the «trial version» the 01/10/2007;
- a user «SimplePackUser1» that purchases the «simple pack»;
- a user «DoublePackUser1» that purchases the «double pack».

Suppose that we are the 02/10/2007 and all users want to reserve a part of the cluster for half an hour. The algorithm finds:

User	Matchmaker response
«TrialVersionUser1»	No resource found
«TrialVersionUser2»	Total CPU: [0;4] and Total Physical Memory: [0;2]GB
«SimplePackUser1»	Total CPU: [0;16] and Total Physical Memory: [0;10]GB
«DoublePackUser1»	Total CPU: [0;32] and Total Physical Memory: [0;20]GB

In the ontology, we find the group «TrialVersionUsers» associated to:

- an instance of «NumberCPULimitedPolicy» with the property «limitedCPU» set to 4;
- an instance of «MaxTotalPhysicalMemoryPolicy» with the property «MaxTotalPhysicalMemory» set to 2GB.

and each particular user got an instance of «EndTimeLimitedPolicy» with the property «LimitedTime» set to his expiration date.

A user of trial version has to satisfy the three policies. This is the reason why «TrialVersionUser1» find no resource (the last policy is not satisfied) and «TrialVersionUser2» find some.

6 Conclusion

In this paper, we propose a description of resources and user requirements based on ontology. This allows a meaningful description that can be completed with new products without any problem and change. This also allows complex access policies and the possibility to easily add new ones. The load of an existing database allows working with existing data with no change for the user.

Our future work includes continuing to develop the ontology to have a more precise description, e.g. network. Some works are currently done on the NAREGI project to measure and store some information about networks. It could be interesting to take this information into account in the ontology and have a more complex description of networks.

The other work will be to interact with the trader mechanism described in [DHP06,Hur06]. This trader mechanism allows finding the services and combination of services that match the user functional requirements. This will allow obtaining a matcher on both physical and functional aspects. The physical aspect is the work expose in this paper about system physical characteristics: memory, the number of CPU, ... The functional aspect will be an extension of the application part of this paper using the trader. The user will not specify the name of the executable but a mathematical description of the service and the services or combination of services that can solve the problem will be given back. The workflow will be automatically generated and the resources that suit both physical and functional aspects returned.

Acknowledgment

The work of the first author was supported by a postdoctoral grant in the NAREGI project. We want to thank professor Miura for this opportunity.

References

[ABD+05] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva. Job submission description language (jsdl) specification, version 1.0. Technical report, Global Grid Forum, 2005.

[DHP06] M. Daydé, A. Hurault, and M. Pantel. Semantic-based service trading: Application to linear algebra. In M. Daydé, J.Palma, A. Coutinho, E. Pacitti, and J. Correia Lopes, editors, *VECPAR*, volume 4395 of *Lecture Notes in Computer Science*, pages 622–633. Springer, 2006.

[DMT99] Distributed Management Task Force. *Common Information Model (CIM) Specification*, Jun 1999.

[FK98] I. Foster and C. Kesselman. *The grid: blueprint for a new computing infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, November 1998.

[HP04] J. R. Hobbs and F. Pan. An ontology of time for the semantic web. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(1):66–85, 2004.

[Hur06] A. Hurault. *Courtage sémantique de services de calcul*. PhD thesis, INPT, Toulouse, DÈcembre 2006.

[KFNM04] H. Knublauch, R. W. Fergerson, N. F. Noy, and M. A. Musen. The protégé owl plugin: An open development environment for semantic web applications. In *Third International Semantic Web Conference - ISWC 2004*, Hiroshima, Japan, 2004.

[KPPN07] S. Kailash, P. Prasanna, V. Prabha, and V. Neela Narayanan. Semantic resource description for grid. In *AMS '07: Proceedings of the First Asia International Conference on Modelling & Simulation*, pages 112–115, Washington, DC, USA, 2007. IEEE Computer Society.

[Ltd07] Hitachi Ltd. Naregi beta2 resource description schema specification and relational data model. Technical report, NII, NAREGI project, March 2007.

[LvS02] S. Ludwig and P. van Santen. A grid service discovery matchmaker based on ontology description, 2002.

[MIU06] K. MIURA. Overview of japanese science grid project naregi. *Progress in Informatics*, 3:67–75, 2006.

[QAW+04] S. Quirolgico, P. Assis, A. Westerinen, M. Baskey, and E. Stokes. *Toward a Formal Common Information Model Ontology*, volume 3307, pages 11–21. Na, 2004.

[RLS98] R. Raman, M. Livny, and M. H. Solomon. Matchmaking: Distributed resource management for high throughput computing. In *HPDC*, pages 140–, 1998.

[SBK+06] T. S. Somasundaram, R.A. Balachandar, V. Kandasamy, R. Buyya, R. Raman, N. Mohanram, and S.Varun. Semantic-based grid resource discovery and its integration with the grid service broker. In *Proceedings of the 14th International Conference on Advanced Computing and Communications*, pages 84–89. IEEE Press„ 2006.

[SD02] M. Sintek and S. Decker. Triple - a query, inference, and transformation language for the semantic web. In *ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web*, pages 364–378, London, UK, 2002. Springer-Verlag.

[TDK03] H. Tangmunarunkit, S. Decker, and C. Kesselman. Ontology-based resource matching in the grid—the grid meets the semantic web, 2003.