SUB-CONTRACTS: DELEGATING CONTRACTS FOR RESOURCE DISCOVERY

Javier Bustos-Jimenez and Cristian Varas
Escuela de Ingenieria Informatica
Universidad Diego Portales
Santiago de Chile
javier.bustos@inf.udp.cl
cristian.varas2@al.udp.cl

Jose Piquer
Departamento de Ciencias de la Computacion
Universidad de Chile
Santiago de Chile
jpiquer@nic.cl

Abstract Grid Computing promised to present a large number of resources distributed on a world-area network, ready to be used by a single user: that promise is true. Now, the problem has moved to the user side, because a normal user normally knows at most only his organization's resources, and those numbers of resources are often not enough for his purposes. Defining a Virtual Organization (VO) as a set of scientific resources, processors, clusters and Grids which are available to the user, we study the problem of resource discovery for VOs in a distributed approach. Viewing a VO-to-VO network as a Peer-to-Peer network, we present our solution based on the use of contracts to perform the query and assignment, delegating contracts if the query cannot be fully handled. We first present a blind scheme of delegation (that is, without knowing of neighbors' resource availability), evaluating it by simulations, and showing that it is not necessary to delegate the query to all neighbors to handle it. Then, a scheme knowing only direct neighbors which uses the blind scheme is presented. Finally, we will give recommendations and extensions of our scheme to improve the resource discovery process.

Keywords: Grid computing, resource discovery, contracts, delegation.

1. Introduction

In large multi-cluster grids the resources suffer, most of the time, of low utilization. If, under some events, the demand exceeds the capacity of a single system, it is possible to take actions like making the system grow by adding more resources, or enqueuing the additional demand until the system can serve it, but none of them seems to be appropriate in the context of Grid Computing.

A better solution lies in making the grids inter-operate in order to drive their collective demand to achieve a stable utilization of the combined system.

For making grids to inter-operate, some design choices involving resource selection and performance must be taken. First, a meta-scheduler should be used to redirect the jobs to the appropriate grid; otherwise, the users would be forced to submit their jobs directly to the grid system arising new problems such as to know where are located the suitable resources, or availability of alien grids.

Considering that a centralized scheduler can be useful in the context of serving sets of processors, cluster or grids belonging to the same institution (also known as a *Virtual Organization* or *VO*) to perform intra-organization scheduling, to extend this approach to inter-organization scheduling (that is, scheduling into VO-to-VO networks) could turn easily into a bottleneck, and a decentralized approach is needed.

Completely decentralized solutions exist but they still have not been able to achieve enough benefits under typical grid workloads. These solutions include the Koala scheduler[1], the AliEn Resource Brokers[2]and OurGrid[3].

Our work is focused in exploiting structural properties of VO-to-VO networks (kind of natural networks), to perform inter-VO resource discovery through the use of *contracts* and delegation. A contract is used to exchange information between parties and then it can be used to transform a resource query into a resource agreement using the same infrastructure. If a VO can handle only part of the resource request, it will delegate the search for the remaining resources. Once all the resources are found and claimed by a virtual organization, the contract could be audited to generate complaints in case of error.

Our solution is partially related with the work of Baraglia et al. [4]which introduces the concept of Grid Awareness, providing to virtual organizations useful information such as network topology and QoS through query/response messages formatted by XML schemes.

This article is organized as follows: Section 2 introduces Contracts for coupling variables, which are used into Section 3 to define delegation schemes for resource discovery. Section 4 presents the behavior of the scheme in simulations for parameter tunning. Future work is presented in Section 5 followed by Conclusions in Section 6.

2. Contracts

Contracts for coupling variables were firstly defined in the work of Leyton et al. [5]. In a nutshell, a contract corresponds to the interaction between two interfaces of different parties, providing the means to define *how* information between these two parties can be shared (in Figure 1, C is the Contract between A and B). Coupling variables with contracts is a scheme similar to Condor Matchmaking [6], which allows finding suitable matches but specifying only *what* information is exchanged. Another related approach to contracts is the Web Services Agreement (WSAgreement) Specification [7], which specifies that "*an agreement defines a dynamically-established and dynamically-managed relationship between parties*".

Contracts were studied in the context of coupling generic interfaces between two parties, and in the work of Leyton et al. contracts were studied specifically to couple distributed application with Grid deployment descriptors. In this work, we use contracts in two contexts:

1 To generate a *resource query* between parties.

2 To generate *resource delegation* between parties.

In the first case, an application with resource (or set of resources) needs sends the contract as a resource query to its parties. In the second case, if a Virtual Organization is only capable to provide a subset of the resources, it delegates the complement set of resources to its own parties.

A contract is defined as a set of *typed clauses*. A coupling between two parties is valid if and only if all clauses on the contract have the same type for both parties and all of them are valid.

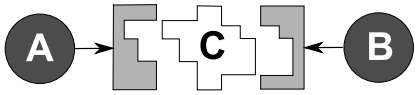


Figure 1. Contracts as an interface between parties

3. Contracts for Resource Discovery

A basic contract for resource discovery has to declare four main clauses:

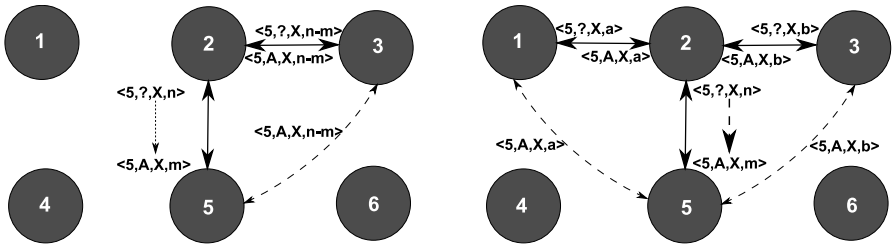
1 `SOURCE` = Reference to the VO which owns the resource query.

2 `RESOURCE` = Name of the resource

3 `NUMBER` = Number of resources wanted

4 `STATUS` = {`QUERY`, `ACCEPTED`, `REJECTED`}

The process of resource discovery begins when a user asks inside of its virtual organization (VO) for a given resource, if the VO is not able to handle the resource request, the query is delegated to other VOs. We study the behavior for two types of delegation: one-to-one and one-to-many (Figure 2).



(a) one-to-one

(b) one-to-many

Figure 2. Delegation schemes

3.1 Delegation one-to-one

A contract-query for `n` resources of type `X` is sent to another VO, if the receiver can handle the query, a contract is made between parties. Nevertheless, if the receiver can only partially handle the request (lets say, it has only `m` available resources of type `X`), it will make a reservation of its `m` resources for the contract and it will choose another VO to delegate the query (now requesting `n-m` resources). In the example of Figure 2(a) we show that the delegated query of `n-m` resources can be handled. Therefore, the original contract `<5,QUERY,X,n>` is set up as `<5,ACCEPTED,X,m>` and a new contract `<5,ACCEPTED,X,n-m>` is made between VOs 3 and 5. Note that the contract between VO 3 and VO 5 is made through VO 2 to inform that reserved resources will be used.

To avoid infinite queries inside the network (for instance, demanding more resources than the number available in the system), the delegation has a `TTL` parameter which is the delegation maximal depth. If after `TTL` delegations the last VO is not capable of handle the request, the contract status is set to `REJECTED`, and therefore its reserved resources are freed. For simplicity reasons, if a delegated contract is rejected, all the contracts belonging to the delegation will be set to `REJECTED` and therefore all the reserved resources will be free.

3.2 Delegation one-to-many

A contract-query for `n` resources of type `X` is sent to another VO, if the receiver can handle the query, a contract is made between parties. Nevertheless, if the receiver can only partially handle the request (lets say, it has only `m` available resources of type `X`, it will reserve its `m` resources for the contract and it will choose other VOs to delegate the query. In Figure 2(b) we show that all delegated parties are capable to handle their request, therefore the original contract `<5,QUERY,X,n>` is set up as `<5,ACCEPTED,X,m>`, and two new contracts (`<5,ACCEPTED,X,a>` and `<5,ACCEPTED,X,b>` having $a + b = n - m$) are made between the VO 5 and VOs 1 and 3 through VO 2.

As in the one-to-one scheme, a `TTL` parameter is set to avoid infinite queries. Moreover, using the `SOURCE` clause the scheme refuses multiples request of resources from the same source, noting that in this scheme is highly probable of receive two times the same contract-query. Again, if a delegated contract is rejected, all delegated contracts will be set to `REJECTED` and therefore all reserved resources will be freed.

3.3 The next step: sub-contracts

Placing us in the context of *VO-to-VO* networks, we can exploit structural properties of this (natural) network storing the resource information of neighbors: sending to neighbors a notification of resource utilization each time a VO make (or finish) a contract.

Therefore, if a VO (A) asks to another VO (B) for `n` resources of type `X`, B will looks into the neighboring information if all together (not including A) can handle the request. If true, a contract `<X,n>` is made between A and B and contracts `<X,ni>` (where $n = \sum ni$) are made between B and its neighbors (this process is known as *sub-contracting*). If no sub-contract can be made, B delegates the query to its neighbors as we defined above.

Due to the dynamic behavior of the VO-to-VO network in terms of resources reserved and used, a restrictive verification such as "*we can handle the query if we have* ***at least*** n *resources of type X*" could produce a fault of resources on contract claiming time, generating complaints between A and B (because finally B was not able to accomplish his contract) and complaints between B and some of its neighbors. Therefore, a ponderer $s > 1$ can be used to sub-contracting only the whole group can handle a request of $s \times n$ resources. This scheme can be used, for instance, if B manages fault-tolerance and A does not, therefore B asks for more resources than A requested to maintain always a set of n resources alive.

4. Contracts on a simulated testbed

Considering that a network made of Virtual Organizations has similar structural properties than a Peer-to-Peer network, we tested our resource discovery scheme on a VO-to-VO network simulated using PlanetSim [8]. In our simulation, we interconnected $1,000$ VOs and randomly placed $1,000$ resources on the network (values were selected considering $1,000$ a medium-size for a network). Then, we ran our scheme measuring the % of succesful requests; that is, given a randomly chosen VO needing n items of a resource, the % of tries that the VO was capable of find the resources and a delegated contract was made.

For the one-to-one delegation scheme, we measured the % using as parameters:

- `TTL` = $1, 2, 3, 4, 5$.
- n = $1,000$; 500; 250; 125 and 63.

The results of our experience are shown in Figure 3. As we expected, a use of `TTL=1` produces a poor performance of the scheme, and the performance increases with `TTL`. The importance of this result is given by the fact that in this scheme `TTL` means also the number of contracts sent between parties (i.e: number of messages in the network). Therefore, if the resource wanted is popular and a VO needs only a low number of items, an one-to-one scheme could be useful.

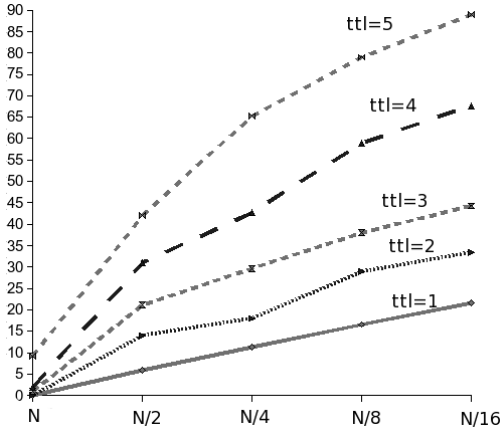


Figure 3. % of succesful request by total number of resources (N) = $1,000$ for a one-to-one delegation scheme

For the one-to-many delegation scheme, defining v as the number of neighbors selected to perform the delegation, and m the number of resources to

search in delegation. We evenly distributed the search sending to each neighbor a contract by $\lceil \frac{m}{v} \rceil$ resources. Then, we measured the % of succesful requests using the following parameters:

- Defining V as the total number of neighbors for a given VO, we used v = V, $\left\lceil \frac{V}{2} \right\rceil$, $\left\lceil \frac{V}{4} \right\rceil$ and $\left\lceil \frac{V}{8} \right\rceil$. The idea of dividing the query in a number less than the total number of neighbors comes from a previous work which performs efficient load-balancing on Peer-to-Peer networks [9].

- `TTL` = $1, 2, 3, 4, 5$.

- n = $1,000$; 500; 250; 125 and 63.

Results are presented in Figure 4, showing that a reduction in the number of delegated contracts does not produce a great reduction on the performance of the delegation scheme and, as we expected, the % of succesful queries is exponential to the search depth.

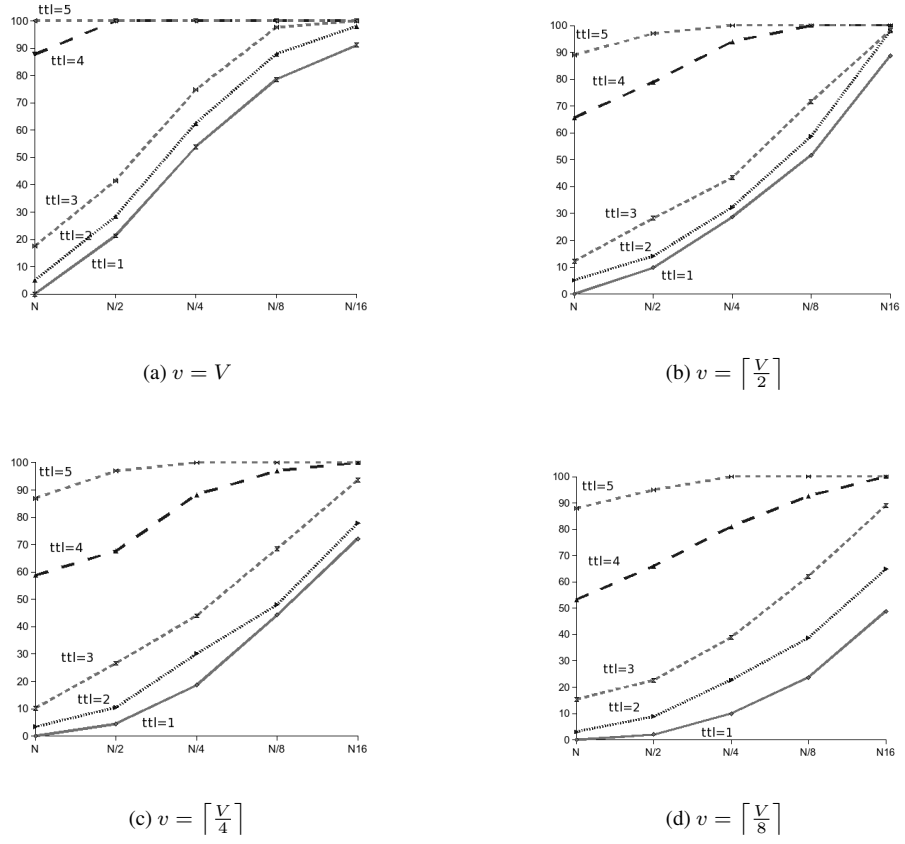
5. Future Work

In this work we presented how to use coupling contracts for resource discovery, validating our scheme through simulations using PlanetSim. We plan to implement our scheme in real systems such as XtreemOS [10]or ProActive [11]to provide them with a full distributed resource discovery scheme.

On the other hand, we plan to extend our scheme in the use of useful information for delegation, for instance to use the information of neighbors (as sub-contracting) to perform *smart* delegations. Also, we plan to add to contracts the number of complaints and to perform composition of resources (or services).

5.1 Complaints: Do you deserve to be part of us?

Some systems as Koala maintain an error counter [1](equivalent to a complaint counter) and, if the number of consecutive errors by an entity reaches a given threshold, the entity is marked unusable (similar to be removed from the network). We do not agree with that scheme because we are aligned with the statement: “let the user decides”. A complaint counter (or complaint-per month CPM ratio) is a useful information for some testbed and it may be added to the contract. If a VO needs a reliable resource the obviously step is to add a clause limiting the CPM ratio. In the other hand, for some experimental testbeds (to test adaptive algorithms such as fault tolerance or distributed garbage collectors) will be useful to have *unstable* networks, that is, with a high CPM ratio.



(a) $v = V$

(b) $v = \left\lceil \frac{V}{2} \right\rceil$

(c) $v = \left\lceil \frac{V}{4} \right\rceil$

(d) $v = \left\lceil \frac{V}{8} \right\rceil$

Figure 4. % of succesful request by total number of resources (N) = $1,000$ for a one-to-many delegation scheme

Note that CPM and the ponderer s are highly related: if a VO wants to looks like very reliable, a good selection of s has to be performed. In fact, the study of s is very important to reduce the number of messages traversing the network.

5.2 The following step: Composition

Imagine now that a VO does not have a given resource, but it knows hot to compose it using other resources. For instance, in Figure 5 VO 1 asks to its neighbors VO 2 for n resources of type `X`, and VO 2 does not have the resource `X` itself but it knows that resource `X` can be composed by resources `A`,`B` and `C`. Moreover, VO 2 knows that its neighbors VO 4, VO 5 and VO 6 have availability of resources `A`, `B` and `C` respectively. Therefore, contracts

between VO 2 and its parties can be made to acquire the needed resources and the contract status between VO 1 and VO 2 can be set up as ACCEPTED.

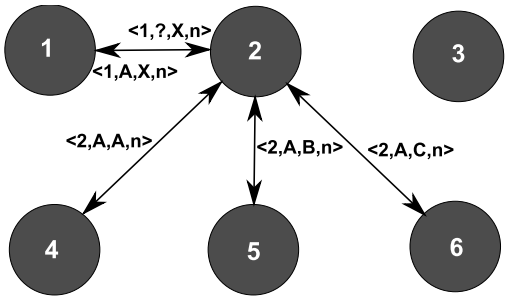


Figure 5. Composition of resources

6. Conclusions

In this work, our solution to the distributed resource discovery problem was presented. This solution, called "sub-contracting" is based on the use of contracts to perform the resource query and assignment, delegating contracts if the query cannot be fully handled.

We first presented a blind scheme of delegation in two flavors: one-to-one and one-to-many, evaluating them by simulations, and showing that the first scheme is usefully for popular resources if the requirement is useful for low number of resources. For the second scheme, we show that it is not necessary to delegate the query to all neighbors, finding on the 85% of the experiences all the needed resources for a mid-size set of resources even using $\frac{1}{4}$ of the neighbors to delegate the query.

Following the blind schemes, a scheme knowing only direct neighbors was presented, discussing the usefulness of a strict delegation due to the dynamic nature of the network.

Finally, recommendations and extensions of our scheme to improve the resource discovery process have been discussed.

Acknowledgments

This work was partially funded by CoreGrid NoE and NIC Labs.

References

[1] H. Mohamed and D. Epema, "Experiences with the KOALA co-allocating scheduler in multiclusters," *Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on*, vol. 2, 2005.

[2] P. Saiz, P. Buncic, and A. J. Peters, "Alien resource brokers," *CoRR*, vol. cs.DC/0306068, 2003.

[3] N. Andrade, W. Cirne, F. Brasileiro, and P. Roisenberg, "OurGrid: An Approach to Easily Assemble Grids with Equitable Resource Sharing," *Proceedings of the 9th Workshop on Job Scheduling Strategies for Parallel Processing*, pp. 61–68, 2003.

[4] R. Baraglia, D. Laforenza, R. Ferrini, N. Tonellotto, D. Adami, S. Giordano, and R. Yahapour, "A study on network resources management," in *Proceedings of the 2nd Integrated Research in Grid Computing Workshop* (S. Gorlatch, M. Bubak, and T. Priol, eds.), (Krakow, Poland), pp. 213–224, CoreGRID, IST, Academic Computer Centre CYFRONET AGH, October 2006.

[5] J. Bustos-Jiménez, D. Caromel, M. Leyton, and J. M. Piquer, "Coupling contracts for deployment on alien grids," in *CoreGRID Workshop on Grid Middleware (in conjunction with EuroPar)*, Lecture Notes in Computer Science, Springer Berlin, September 2006.

[6] R. Raman, M. Livny, and M. Solomon, "Matchmaking: Distributed resource management for high throughput computing," in *HPDC '98: Proceedings of the The Seventh IEEE International Symposium on High Performance Distributed Computing*, (Washington, DC, USA), pp. 140–146, IEEE Computer Society, 1998.

[7] A. Andrieux, karl Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu, "Web Services Agreement Specification (WS-Agreement)," *Grid Resource Allocation and Agreement Protocol WG, GGF*, vol. 1, 2006.

[8] P. Garc"a, C. Pairot, R. Mondéjar, J. Pujol, H. Tejedor, and R. Rallo, "PlanetSim: A New Overlay Network Simulation Framework," *Lecture Notes in Computer Science (LNCS), Software Engineering and Middleware (SEM)*, vol. 3437, pp. 123–137, 2005.

[9] J. Bustos-Jiménez, D. Caromel, and J. M. Piquer, "Toward the infinite network, and beyond," in *Proceedings of 12th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP)*, vol. 4376 of *Lecture Notes in Computer Science*, pp. 176–191, Springer Berlin Heidelberg, June 2006.

[10] C. Morin, "Xtreemos: A grid operating system making your computer ready for participating in virtual organizations," *isorc*, vol. 00, pp. 393–402, 2007.

[11] Oasis Group at INRIA Sohpia-Antipolis, "Proactive, the java library for parallel, distributed, concurrent computing with security and mobility." `http://proactive.objectweb.org`, 2002.