

Grid Trading Services in Linear Algebra^{*}

Aurélie Hurault¹, Marc Pantel¹, Eddy Caron², and Frédéric Desprez²

¹ IRIT - ENSEEIHT, 2 rue Camichel, B.P. 7122, F-31071 TOULOUSE CEDEX 7
{Aurelie.Hurault,Marc.Pantel}@enseeiht.fr

² LIP - ENS Lyon, 6 allée d'Italie, F-69364 LYON CEDEX 7
{Eddy.Caron,Frederic.Desprez}@ens-lyon.fr

Abstract. Trading service is a difficult problem on a grid due to the high number of similar services and the rapid change in performance and availability. It must rely both on the services descriptions and on their availability and performance. These aspects are usually managed in two separate parts of the trading process. We advocate in this paper that an accurate description of services that allow to combine several services in order to produce a more sophisticated one should cooperate with the scheduling part during the trading process in order to improve the performance of the produced composite service. This paper first show how an accurate semantics in the descriptions of the services allows to combine several available services to execute a more complex one. To have a meaningful description, we have to consider applications in a specific domain. This paper deals with the linear algebra domain (however, the approach remains general and can be applied to any domain). In most case, many composition can answer to a client service request.

This paper then propose an interaction with a grid scheduling middleware, which provides insights about the availability of services, to improve the effectiveness of the trading algorithm. These insights are useful to direct research and obtain, more quickly, a pertinent service offer. Then, this multiple offer is given back to the middleware, which chooses the best one to execute depending on the load of computers and network. The composition of services introduces constraints, due to the amount of exchanged data, to choose the place where the service will be executed.

1 Introduction

The main purpose of software trading is to find the right piece of software according to the user need. Distributed service trading combines the search for a piece of software and for a computer able to run it (usually looking for the best one). Several aspects must then be taken into account :

- the function provided by softwares,
- the availability of softwares on computers,
- the cost of running each software on a given computer,

^{*} This work has been partially supported by the ACI GRID TLSE project from the french department of research.

- the cost of communicating parameters and results from the client to the computer running the software,
- ...

A grid provides a lot of different computers with different communication links. A grid service trader should then take into account the following specific points :

- a computer may be running several softwares concurrently,
- many different softwares are available that may provide the same function,
- many different computers may be running these softwares,
- some services may require the use of various software running on different computers,
- several versions of the same software may be available on different computers, on the grid, the parameters and results should then be exchanged between the various parts of the global service,
- the cost of parameters and results exchange may be very different from computers to computers,
- the cost of software run may be very different from computers to computers.

Most of the available grid traders either focus on accurate service description or on sophisticated scheduling strategies in order to lower the costs. These two points of view are usually combined by trading first the best software according to the functional requirements then scheduling the best computer and communication link for the chosen software. Usually, on a grid, the best software according to several requirements is not the best for each of the requirement considered separately. This weakness becomes significant when the required service relies on the run of many different services available on different computers. Several service combinations are usually available for a given service. Choosing one of these combination without taking into account the communication and running cost can lead to costly composite solutions.

To satisfy these constraints, we propose to combine an accurate software trader with a grid middleware in order to choose the best composite service. The main advantage is that the software trader will be able to choose the best available service combination in order to satisfy the user request and the middleware will reduce the running and communication costs.

1.1 The trader

To this end a framework for accurate services trading [HP05] is developed by the two first authors. Services descriptions must be as precise and natural as possible. To have an appropriate description, without losing the possibility of using these descriptions in a decidable algorithm, the target domain is restricted to a specific one (this paper will focus on linear algebra). The Monet project³ treats similar problems. This project aim *«is to develop a framework for the description*

³ <http://monet.nag.co.uk/>

and provision of web-based mathematical services». The main difference between Monet and our framework is the kind of semantics used. This point is discussed later. Our framework has the advantage to be adaptable to any domain. In particular, it will be integrated to the GRID-TLSE project [BDPP04] whose goal is to design an expert site for sparse matrices solvers. In this specific project, an interaction, with a middleware which will be in charge of service localization and computational requests scheduling among available servers, will be possible.

1.2 The scheduling middleware DIET

A scheduler (or a set of distributed schedulers), on a grid, allows us to use the most adequate service, depending on information about the machines load and the network state, and on historical information about previous executions. In the GRID-TLSE project, the middleware used is DIET [CDF⁺02]. Both teams collaborate to integrate our framework to DIET. This integration done, this framework will not only be used by the GRID-TLSE project but can be integrated in other PSE. It can, for example, be useful for a Scilab user who is looking for a service in his libraries or over the grid.

The rest of this paper is organized as follow. The next section describe the problem we propose to solve. Then, an example is given to illustrate the problem. Finally, the grid middleware DIET and its interactions with the framework to improve the research is exposed.

2 Problem description

Service description is a key aspect in the search for services. More or less formal kinds of semantics can be used to describe services. Research based on informal semantics are usually based on keywords or meta-data. It is currently the case for the GRID-TLSE project [PPA05,Pan04]. The disadvantage of this method is the requirement of a preliminary agreement to define the keywords and their meaning (i.e definition of an ontology). The Monet project uses ontologies.

Another method consist in using a formal semantic describing more precisely the mathematical function computed by the service. In the case of a general application domain, this semantics will be complex or unexpressive. This is the reason why, our framework must be used in some specific application domain. The semantics can then be expressive and simple. The formal semantics advocated in our framework is similar to algebraic data type specification. Indeed, the required information are:

- the types (or sorts) used (*Scalar, Matrix, ...*);
- the main operators of the specific domain (*add, mult, ...*);
- the operators signatures ($ar(add) = Matrix \times Matrix \rightarrow Matrix$);
- the operators properties (like commutativity and associativity), which will have an impact on the parameters permutation ($add(a, b) = add(b, a)$) and

equations which link operators (like distributivity). The Monet project is based on OpenMath to describe services. In OpenMath, operators are described in *Content Dictionaries*, by a natural language description and an OpenMath operators properties description. These descriptions could be used to fill our system. However, to our knowledge, Monet does not use these properties for service trading.

Then, services trading consist in comparing the user request and the available services using those equations. This approach is related to E-unification. Our algorithm is based on the work of J.H. Gallier and W. Snyder [GS89] and of C. Lynch and B. Morawska [LM01]. These proposals defines sets of transformations which are complete but undecidable. Using these results, we implement decidable heuristics based on an amount of energy given to the algorithm (number of equations which can be applied in order to produce a given service using the available ones). Our approach is complete if we give an infinite amount of energy. Therefore, our algorithm may take an infinite time trying to find a solution. To release this constraint our algorithm is based on a breath first traversal of the search tree which will produce finite solutions before looking for infinite ones. This constraint is therefore not relevant as search can either be interactive or controlled by another program managing to access the finite solution without looking for the infinite one. All the more, concluding that a service is not available, even if a very complex combination of existing ones could be used is not too much of a problem. If a user really need a service, he will provide more energy and wait longer.

3 Description of a particular problem

In a restricted domain, like linear algebra, a simple and accurate service description is possible. In this purpose, the types and operators, used in the services description, must be defined. In the case of linear algebra, used types will be matrices, vectors and scalars, the operators will be addition, multiplication or transposition. In order to provide an easy to read example, the number of operators in our example is restricted.

Operators description For each operator used to describe services, its name, its signature and its properties must be specified. The constants of the domain must also be known. Here is a subset of the description for linear algebra:

| | | |
|----------------------------|-----------|---|
| infix commutative operator | + | : Matrix \times Matrix \rightarrow Matrix . |
| infix operator | * | : Matrix \times Matrix \rightarrow Matrix . |
| postfix operator | \hat{T} | : Matrix \rightarrow Matrix . |
| prefix operator | op | : char \times Matrix \rightarrow Matrix . |
| infix operator | * | : Scalar \times Matrix \rightarrow Matrix . |
| constant | I, O | : Matrix . |
| constant | 1, 0 | : Scalar . |
| constant | 'n', 't' | : char . |

Services description Once the basic operators have been defined, the available services can be described with these operators. Currently, the services can be functions or procedures. That means, in particular, that signature must take into account parameters transmitted by references. An example of services description is the following:

1. $add(Matrix\ a, Matrix\ b) : Matrix$
 $= (a + b)$
2. $mult(Matrix\ a, Matrix\ b) : Matrix$
 $= (a * b)$
3. $sgemm(char\ transa, char\ transb, Scalar\ m, Scalar\ n, Scalar\ k,$
 $Scalar\ \alpha, Matrix\ a, Matrix\ b, Scalar\ \beta, Matrix\ c) : void$
 $= c \leftarrow ((\alpha * (op(transa, a) * op(transb, b))) + (\beta * c))$

In the level three BLAS [BLA02] routine *sgemm*, parameters *transa* (respectively *transb*) are used to indicate if the service apply on a matrix or its transpose, that is $op('n', x) = x$ or $op('t', x) = x^T$. The parameters *m*, *n* and *k* are link to the matrices sizes.

Request and results Once these elements have been described, the user can submit his request. A comparison with all the available services is done and the set of solutions found is given back. For example among the results for the request: «*x, y, z : Matrix (x * y) + z*», will be:

- the ones requiring a single service:
 - $sgemm('n', 'n', m?, n?, k?, 1, x, y, 1, p1 = z); p1;$
 When the result is obtain by modifying a parameter transmitted by references, the result *p1* specifies which parameter contains the computation. When the value of a parameter does not matter, it will be assigned to *Any* value. The parameters followed by a «?» will be determined by an other tool. As *m*, *n* and *k* are link to the sizes of the matrices. The matrix *a* is a $m \times k$ matrix, *b* a $k \times n$ matrix and *c* a $m \times n$ matrix. We currently only use very simple types however we plan to integrate more sophisticated constraints (for example on the size of matrices). We will also rely on the use of metadata. This part of our work will rely on the TLSE project (see [BDPP04,PPA05]).
- the ones requiring several services:
 - $p2 = mult(x, y); p1 = add(p2, z); p1;$
 - $p2 = mult(x, y); p1 = add(z, p2); p1;$
 We first execute the multiplication, its result is then an input to the addition, and the result of this addition is returned to the client.

Operators properties Determining the compatibility of a service request and an available service requires to compare their descriptions. So, the operators properties (commutativity, associativity, ...) and the properties which link the various operators (distributivity, neutral element, absorbant element,...) must be known to be able to transform a problem into an equivalent one, and eventually transform the requested service into one of the available ones (or a combination of available ones). In our example, to obtain the precedent results, the following properties must be known:

$$\begin{array}{l}
(O^T) = O \\
\forall a : Matrix \quad (a + O) = a \\
\forall a : Matrix \quad (a * I) = a \\
\forall a : Matrix \quad (a * O) = O \\
\forall a : Matrix \quad (0 * a) = 0 \\
\forall a : Matrix \quad (1 * a) = a \\
\forall a : Matrix \quad op('n', a) = a
\end{array}
\left|
\begin{array}{l}
\forall a : Matrix \\
\forall \alpha : Scalar; \forall a, b : Matrix \\
\forall \alpha : Scalar; \forall a, b : Matrix \\
\forall a, b, c : Matrix \\
\forall a, b, c : Matrix \\
\forall a, b : Matrix \\
\forall a, b : Matrix
\end{array}
\right.
\begin{array}{l}
op('t', a) = (a^T) \\
(\alpha * (a * b)) = ((\alpha * a) * b) \\
(\alpha * (a * b)) = (a * (\alpha * b)) \\
(a * (b + c)) = ((a * b) + (a * c)) \\
((a + b) + c) = (a + (b + c)) \\
((a + b)^T) = ((a^T) + (b^T)) \\
((a * b)^T) = ((b^T) * (a^T))
\end{array}$$

For example to assign values to the parameters β and c of *sgemm*, the services $\beta * c$ and z will be compared. The equation $\langle \forall a : Matrix (1 * a) = a \rangle$ is then needed to be able to answer that $\beta = 1$ and $c = z$ is a solution.

4 Interaction with a grid middleware

As the trader gives back all the possible solutions, the set of results can be very large and a non negligible time can be necessary to obtain them. With the precedent example, if we allow both a service combination depth of two and to apply two equations, the trader gives back 195 results using less than one second. Moreover, all the solutions are not as interesting as the others from a performance point of view (in the precedent set, we can find : $\langle sgemm('n', 'n', m?, n?, k?, 1, x, y, 1, p3 = O); p2 = add(p3, O); p4 = mult(O, I); p5 = mult(z, I); p6 = add(p4, p5); p1 = add(p2, p6); p1 \rangle$). So, it is necessary to improve the research using weighting cost functions to sort the results and obtain the most relevant results as quickly as possible.

The idea is to use a scheduling middleware which provides information specific to the target application. These information will be used to control the research and propose the most relevant solutions. The middleware will then schedule the chosen composite service.

4.1 The DIET scheduling middleware

As we said before, this work will be integrated to the TLSE project which uses the DIET middleware in order to schedule the computations on the grid. On the client side, DIET implements the GridRPC [SLD⁺04] API. This paradigm is close to the classical RPC (*Remote Procedure Call*) model. The environments following this paradigm are usually called Network Enabled Servers (NES). Several environments provide this feature like NetSolve [AAB⁺01], NINF [TNS⁺03], DIET [DIE], NEOS [NEO], or RCS [AGM97].

In [MNSS00], authors give a survey of NES based environments that allow the access to computational servers through the Internet. These environments are usually based on five different kind of components: **clients** that submit computation requests to servers, **servers** that solve requests on behalf of clients, **monitors** that gather information about the current state of resources and store them in a **database**, and finally a **scheduler** (also called **agent** in our architecture) that choose an appropriate server depending of the submitted problem and the information stored in the database.

NetSolve and Ninf projects follow this approach. Unfortunately, only one scheduling agent can be started which is in charge of load-balancing for a given set of servers⁴. This can lead to a performance bottleneck that forbids the use of large number of clients and servers. This also can lead to fault-tolerance problems. Indeed, the crash of the agent breaks the whole platform.

To overcome these problems, DIET distribute the work of the agent using a new organization. The agent is replaced by a set of agents connected in two different ways: a hierarchical connection improving the scheduling efficiency (Figure 1) and a peer-to-peer approach improving the robustness of the system (Figure 2). The distribution of the agent's work has several advantages. First, a better load-balancing of its work among the different agents and a better stability of the system (if one element crashes, other elements can reorganize their connections and replace it). Finally, we obtain a simpler management of large scale networks of servers (the administration of each group of server and their associated agent can be made local).

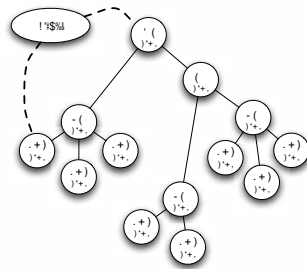


Fig. 1. DIET Architecture.

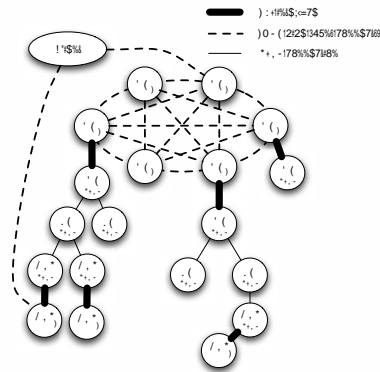


Fig. 2. DIET_J Architecture using P2P Connections.

A **client** is an application that use DIET to solve computational problems. Several kind of clients should be able to connect to DIET. A problem can be submitted from a web page, a problem solving environment like Matlab or Scilab [CCCV⁺01] or from a compiled code (in C, C++, Fortran, ...).

A **Master Agent (MA)** is directly connected to clients. It receives computational requests and choose one (or several) SeD(s) which are able to solve the problem in a small amount of time. A MA has the same informations has a LA but it has a global (and high level) view of problems that can be solved and data already in sub-trees.

⁴ One version of Ninf allows the use of several agent (Metaserver) but each of them has a global knowledge of the whole environment using information broadcast.

A **Leader Agent (LA)** sets a hierarchical level within DIET agents. It can be the link between a Master Agent and a SeD, between an other LA and a SeD, or between two LAs. It aims at broadcasting requests and information between MAs and SeDs. It maintains a list of working requests, and for each of its subtrees, the number of servers able to solve a given problem as well as information about persistent data.

A **Server Daemon (SeD)** is the entry point of a computational server. It is directly connected to a LA. It maintains a list of available data on a server (maybe with their data distribution and the way to access them), a list of problem that can be solved, and all the information about its load (CPU, memory, disk, ...). On a parallel machine, a SeD will be started on its front-end.

The current target platform of DIET is the fast network connecting clusters and parallel machine of research centers involved in the Grid5000 project ⁵. This architecture is highly hierarchical as a WAN connects each LAN of research center, as well as the networks of the local clusters.

DIET behaves in the following way. A new client must first contact a master agent (the most appropriate, for example by minimizing a network distance). Then it can submit a request to it. The problem asked within the request can also be solved by a succession of different services. To choose the most appropriate server to solve a given problem, the Master Agent broadcast the request within its sub-trees ⁶ to be able to find at the same time data involved (which may already on the network due to previous computation) and servers able to solve the problem asked. When the request reaches the SeDs, they compute an estimation of the execution time (if possible) as well as the communication time to move data to a given server. This done through our performance evaluation tool called FAST [Qui02]. If the server owns a data needed for the request sent, it sends this information back to the agent hierarchy.

Scheduling decision can then be taken at each level of the hierarchy up to the MA. At each level, a timer is started to be able to wait until some LA have answered. When the answer reaches the Master Agent, it takes a final scheduling decision and sends the address of the chosen server back to the client. (A list of "best" servers can also be sent to the client). The client then send the computation request to the chosen server, together with the data needed. After the completion of the computation, output data are sent back to the client. Data persistence can also be applied to avoid extra communication overhead [DFLNP05].

The DIET architecture will thus allow to communicate to the trading framework important information. Among them, we will get the availability of services, the machine load, their performance, ... The trading framework will provide DIET with information about the possibility of using one service or the combination of several services to solve a more sophisticated computational request on behalf of one client.

⁵ <http://www.grid5000.org>

⁶ The multi-MA extension works as well and other MAs connected to the MA called act as LAs.

4.2 Relevant solutions

To obtain an exploitable set of solutions, only the most relevant ones should be kept among the ones given back by the trader. To this end, the trader needs some information about the service running cost. Some of these information can be statically described, like the ones which express that an equation is more expensive than an other (factorization is cheaper than development because the result term has less operations) or that an operation is cheaper than an other (multiplication of matrices needs more floating point operations than addition). Other information, which are dynamics, depend on the application and will be provide by DIET.

Piloting research To improve the search of relevant results, we can direct the trading process by first considering the most interesting services and equations. To know if a service is interesting, we will consider its complexity (static information) and its availability (dynamic information). For the equations, there are only static information. Using at the beginning of the trading process these services and equations, the first solutions found will be the most relevant ones. Static information are not sufficient. Indeed, we will prefer to compare a request with a service located on a server which has a strong availability than a request with a service located on a busy server. The service might then need more time to be executed, but we must wait less time before running it. DIET will provide these dynamic information.

Sorting results The results thus obtained will be sorted. Currently, this sort is done by considering the complexity of the operators (and thus of the services) and by considering the parameters. With the same complexity, we will prefer a service whose parameters are the simplest ones. Suppose that $f(x, y, x, O)$, $f(x, y, O, O)$ and $f(x, y, Any, O)$ solve the problem, the most interesting result is the last one ($f(x, y, Any, O)$) because it is the most general. $f(x, y, O, O)$ is more interesting than $f(x, y, x, O)$, because the null matrix is less complex than the user matrix, so the computation will be faster. In the general case, services with same complexity will be sorted in function of the numbers of *Any*, constants and parameters given by the user, among their parameters.

Execution The most relevant results will be transmitted to DIET which will evaluate them. In the case of simple services (without combination), the state of the machine where the service is located, its capacity, its availability . . . will be considered. In the case where services are combined, in addition to these information, the data dependencies must be also take into account to evaluate the costs in term of communication between the computers running the different services. Indeed, the local execution (even on a less powerful server) might be quicker than the remote execution because of the extra overhead due to data movements.

If none of the services seems satisfying to DIET, DIET can ask for more results until obtaining satisfaction. More complex searches might be started. When DIET will obtain a valid solution, it will execute the request (or the sequence of requests).

5 Conclusion

To improve the trading of services on a grid, we propose to combine two aspects of services description. The first one is the service semantics, which represents its functionalities, and the second one is the service availability and performance (both in runtime and communication). So we propose to mix functional and non-functional (in connection with the execution) aspects for a better services choice.

This integration of our framework and DIET is under development, it will be used to validate all the possible interactions between the trader and the scheduler, at the various possible levels. The service semantics used in this paper only take into account the mathematical function provided by the service. The type used for parameters are very simple. We will integrate in our future work more elaborate types and metadata about the parameters and the services. Constraints between the various types and metadata will improve much further the quality of the result provided by the trading process.

References

- [AAB⁺01] D. Arnold, S. Agrawal, S. Blackford, J. Dongarra, M. Miller, K. Sagi, Z. Shi, and S. Vadhiyar. Users' Guide to NetSolve V1.4. Computer Science Dept. Technical Report CS-01-467, University of Tennessee, Knoxville, TN, July 2001.
- [AGM97] P. Arbenz, W. Gander, and J. Mori. The Remote Computational System. *Parallel Computing*, 23(10):1421–1428, 1997.
- [BDPP04] M. Buvry, M. Daydé, M. Pantel, and C. Puglisi. TLSE Project: A Grid-Based Expertise Site for Sparse Matrix Solvers. In *AcrossGrids 2004, Nicosia, Cyprus*, 28-30 janvier 2004.
- [BLA02] An Updated Set of Basic Linear Algebra Subprograms (BLAS). *ACM Trans. Math. Softw.*, 28(2):135–151, 2002.
- [CCCV⁺01] E. Caron, S. Chaumette, S. Contassot-Vivier, F. Desprez, E. Fleury, C. Gomez, M. Goursat, E. Jeannot, D. Lazure, F. Lombard, J.-M. Nicod, L. Philippe, M. Quinson, P. Ramet, J. Roman, F. Rubi, S. Steer, F. Suter, and G. Utard. Scilab to Scilab//, the OURAGAN Project. *Parallel Computing*, 11(27):1497–1519, OCT 2001.
- [CDF⁺02] E. Caron, F. Desprez, E. Fleury, F. Lombard, J.-M. Nicod, M. Quinson, and F. Suter. Une approche hiérarchique des serveurs de calculs. In Françoise Baude, editor, *Calcul réparti à grande échelle*. Hermès Science Paris, 2002. ISBN 2-7462-0472-X.
- [DFLNP05] B. Del Fabbro, D. Laiymani, J.-M. Nicod, and L. Philippe. Data management in grid applications providers. In *IEEE International Conference DFMA'05*, Besançon, France, February 2005. to appear.

- [DIE] DIET. <http://graal.ens-lyon.fr/DIET>.
- [GS89] J. H. Gallier and W. Snyder. Complete Sets of Transformations for General E-Unification. *Theor. Comput. Sci.*, 67(2-3):203–260, 1989.
- [HP05] Aurélie Hurault and Marc Pantel. Algebraic approach to semantic service trading. In *Submission to FMOODS 2005*, 2005.
- [LM01] C. Lynch and B. Morawska. Goal-Directed E-Unification. In A. Middeldorp, editor, *Rewriting Techniques and Applications, 12th International Conference, RTA-01*, LNCS 2051, pages 231–245, Utrecht, The Netherlands, May 22–24, 2001. Springer.
- [MNSS00] S. Matsuoka, H. Nakada, M. Sato, and S. Sekiguchi. Design Issues of Network Enabled Server Systems for the Grid. <http://www.eece.unm.edu/dbader/grid/WhitePapers/satoshi.pdf>, 2000. Grid Forum, Advanced Programming Models Working Group whitepaper.
- [NEO] NEOS - Server for Optimization. <http://www-neos.mcs.anl.gov/neos/>.
- [Pan04] M. Pantel. Test of Large Systems of Equations on the Grid: Meta-Data for Matrices, Computers, and Solvers. In *PMAA'04*, 2004.
- [PPA05] Marc Pantel, Chiara Puglisi, and Patrick Amestoy. Grid, Components and Scientific computing. In *Submission to Euro-Par 2005*, 2005.
- [Qui02] M. Quinson. Dynamic performance forecasting for network-enabled servers in a metacomputing environment. In *International Workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems (PMEO-PDS'02), in conjunction with IPDPS'02*, April 15-19 2002.
- [SLD⁺04] K. Seymour, C. Lee, F. Desprez, H. Nakada, and Y. Tanaka. The End-User and Middleware APIs for GridRPC. In *Workshop on Grid Application Programming Interfaces, In conjunction with GGF12*, Brussels, Belgium, September 2004.
- [TNS⁺03] Yoshio Tanaka, Hidemoto Nakada, Satoshi Sekiguchi, Toyotaro Suzumura, and Satoshi Matsuoka. Ninf-G: A Reference Implementation of RPC-based Programming Middleware for Grid Computing. *Journal of Grid Computing*, 1(1):41–51, 2003.