http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 7 (6): 859-867, 2008
ISSN 1812-5638

A Scalable Grid Information Service Framework for Engineering and Scientific Computation

[1]G.Y. Wei, [1]G.X. Wu, [1]Y.Y. Gu and [2]Y. Zheng
[1]Zhejiang Gongshang University, Hangzhou, 310018, People's Republic of China
[2]Zhejiang University, Hangzhou, 310027, People's Republic of China

Abstract: This study proposes an adaptive information discovery framework for computational grid, called PIVOT. With an active information discovery mechanism, PIVOT can extract and provide explicit information of distributed grid resources for specific scheduling algorithm. By introducing a tunable α-hops flooding method for distributed information query and collection, PIVOT supports customized resources information retrieval to fulfill requirements of applications. The scalable and adaptive information discovering mechanism of PIVOT is better than traditional pre-configured information services. PIVOT is implemented in the grid environment MASSIVE and is evaluated with an actual scheduling algorithm. Experiments demonstrate that PIVOT improves the effectiveness of resources scheduling and lessen the executing time of grid tasks.

Key words: Computational grid, information discovery, scalability, resources management

INTRODUCTION

Grid computing is distinguished from conventional distributed computing by its focus on large-scale resource sharing for scientific and engineering problem solving, in some cases high-performance orientation. The problems underlie the grid is coordinated resources sharing in dynamic, multi-institutional organizations and individuals. As grid becoming a viable high performance computing many experiment platforms of scientific and engineering domains are transferring to computational grid environment because it is cheaper than supercomputers. Usually, grid computing system can not provide enough performance as supercomputing system. The key reason is that network of grid can not achieve high speed as its compute power. This reason extremely limits its application in the scientific and engineering domains. The large-scale numerical simulation of complicated engineering problem is a type of communication-bounded, IO-bounded and computation-bounded task along with amount of computation. Also, volumes of data and special visualization devices are involved when parallel solving program running upon the grid. In such computing environment, those grid resources scheduling algorithms only concerning computing power can not work perfectly. How to improve grid performance for scientific and engineering applications? Facing the challenge, we need more effective scheduling algorithms. Novel grid resources management methods should be issued to improve computing performance. Effective algorithms should base on sufficient resources information. But, how many information is needed?. No one can answer. It lies on the resources demands of applications and on the design schemes of resources scheduling algorithms. The computation intensive applications, such as Monte-Carlo simulations (Li and Mascagni, 2003) and SETI@Home (Anderson *et al*., 2002), need high speed computing power. The I/O intensive applications, such as distributed data sharing system (Chervenak *et al*., 2002) need large capacity storage. The communication intensive applications, such as computational fluid dynamics simulations (Dijkstra and Steen, 2005) need large bandwidth and low latency. So, we need an adaptive scalable grid resources information collector that can discover and obtain demanded resources information. It not passively collects information pre-stored by resources themselves, but works in an active mode, deploys itself and extracts resources information as scheduling algorithm needed. As we known, information services of current grid computing middle ware, such as Globus (Foster and Kesselman, 1997), Legion (Andrew *et al*., 1997), UNICORE (Almond and Snelling, 1999), Sun Grid Engine, Condor (Frey *et al*., 2002), GLORIAD, PRAGMA, CROWN (Huai *et al*., 2007), works in a passive mode, the information content is pre-configured by resources' owner and retrieval method usually adopts grid service or

Corresponding Author: G.Y. Wei, College of Computer Science and Information Engineering, Zhejiang Gongshang University, Xuezheng Street No. 18, Hangzhou, Zhejiang, People's Republic of China
Tel: +86-571-2800-8318 Fax: +86-571-2800-8303

web service technology. The type, volume and content of obtained information are limited by diverse resources sharing schemes. Resources demander can only run its scheduling algorithm based on intersection of all resources information that is the minimized valid information set.

In this study, we issue an adaptive information discovery framework for computational grid (PIVOT) to resolve above problem. PIVOT is a part of our MASSIVE project. It consists of three layers, the registration and deployment layer, normalization and organization layer and information collection layer. Unlike Web Services, PIVOT provides an active information discovery scheme. Users can acquire customized grid resources information that fit into their special resources schedule algorithms. PIVOT is implemented in our grid environment MASSIVE (Wei and Zheng, 2004). It utilizes grid security infrastructure (Butler *et al*., 2000) for resources autonomous secure control. Present experiments demonstrate PIVOT improves the scalability of grid information services and the efficiency of resources utilization.

MASSIVE PROJECT OVERVIEW

The grid created for the MASSIVE project utilizes resources located at the CESC and relevant departments of Zhejiang University, as shown in Fig. 1. We plan to add resources at other institutions of high performance computing. The grid currently uses Globus Toolkit as the middleware to enable the resources to be accessed after authentication and authorization.

Using OGSA architecture, the MASSIVE project enables grid-based Computational Fluid Dynamics (CFD) and Computational Solid Mechanics (CSM) by some specified services. Kernel services of MASSIVE include: geometry pre-processing and mesh generation service, migration and execution service, collaborative visualization service and data analysis service. A typical usage would be the generation of a mesh using a mesh generation service on an IRIX platform, the solution of CFD and CSM problems with the meshes previously created on a PC cluster and the collaborative visualization of the numerical results with equipments such as a stereo tiled display wall and a BARCO stereo projection system at the CESC.

The architecture of MASSIVE is constructed with multiple layers, as given in Fig. 2. The bottom layer is hardware that includes varied distributed heterogenous resources. The second is security layer, GSI. The third layer is basic grid services provide by grid middle-ware. The fourth layer contains services that are visually re-capsulated services of the third layer. PIVOT is resided in this layer to provide scalable information services. The fifth layer is composite services that integrate application-oriented libraries. The top layer contains three environments that respond to the steps of multidisciplinary problem solving.

MASSIVE supports the composition of applications from service-based components, execution and monitoring of such applications on remote resources and collaborative visualization, exploration and analysis of the numerical results. In addition, the MASSIVE environment

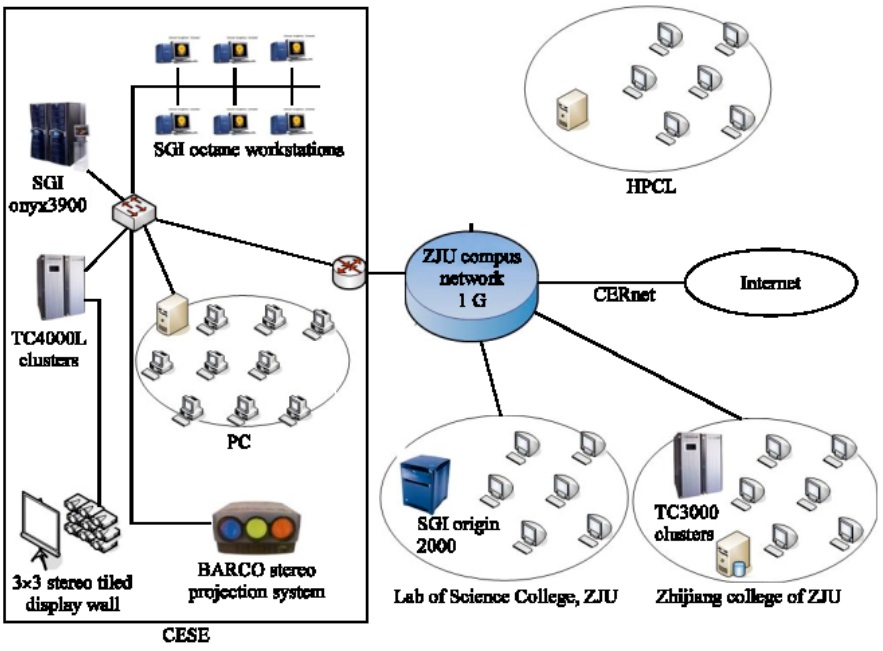

Fig. 1: The fabric layer of MASSIVE Grid

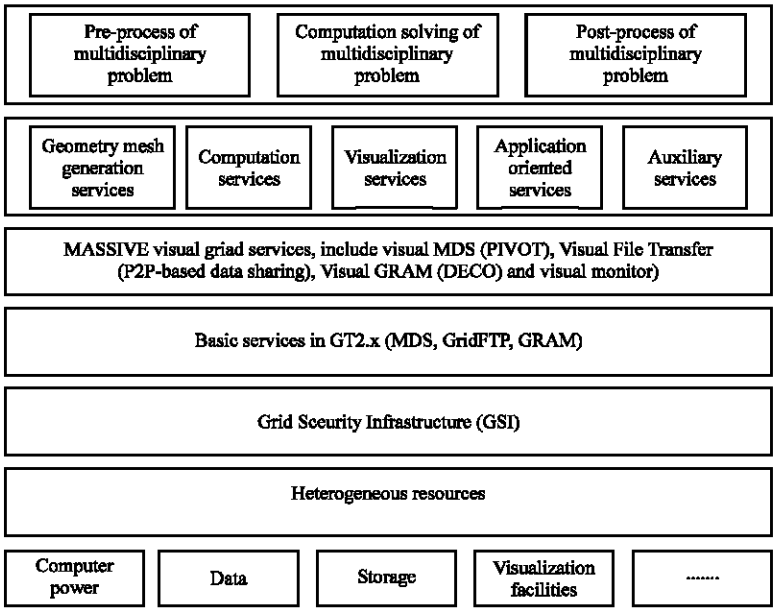

Fig. 2: The layered structure of MASSIVE

also provides an interface to meshing service and supports the visualization capability of meshes created. To minimize the risks of remote execution of simulations, the MASSIVE makes it possible for the user to keep the codes and data at a local machine and to transfer them to a remote machine only when it is to be executed and it is then erased from the remote site after execution. Services will be provided to support the secure migration, remote execution, deletion of the application program and the secure return of the application results to the user (Maria and David, 2008).

PIVOT FRAMEWORK

Topology of resources: Most wide-area distributed systems contain a range of networked devices ranging from computer components (CPU, memory and disk), to network components (hubs, routers, gateways) and specialized data sources (embedded devices, sensors, data-feeds). The topology of the grid fabric layer is important for any resources management scheme and scheduling algorithm. It also affects information retrieval methods. In order to describe information query and collecting process, we use a hypothetic grid resource topology (Fig. 3).

In Fig. 3, the resources A, B and C constitute organization one. In the same way, resources C, D and E constitute organization two. There, resource C is overlapped in two organizations. Any organization is complete graph. The resources F, G and H are shared by

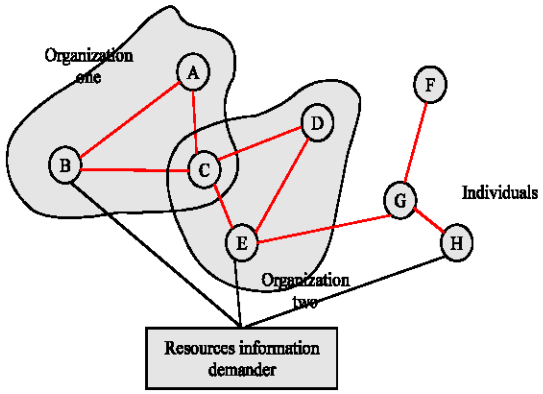

Fig. 3: Topology of a hypothetic grid includes two virtual organizations and several individual sharing resources

individuals. We assume all resources adopt the same security mechanism provided by Grid Security Infrastructure (GSI).

Based network topology given in Fig. 3, we propose a tunable α-hops flooding method for information collection.

α-hops information collection: The process of resources information collection includes three sequential steps. Firstly, information demander deploys its specialized information collector. Actually, the collector is a program that can be run in multiple platforms. When disseminated to objective host, it registers as a local routine with

security consideration and periodically extracts specialized resource information with privacy guarantee mechanism. In this deployment scheme, the collector is distributed within α-hops from demander host. To describe the collector disseminating process, we depict a 2-hops forwarding in detail using topology given in Fig. 3 (shown as Fig. 4). The collector reaches node C and G twice and does not reach node F. Secondly, the deployed collectors exchange their location. All collectors form an overlay network logically that look like a peer to peer information sharing system. But, only neighbored collectors periodically communicate each other to exchange dynamic information their freshly discovered. Finally, all collectors send back their mined information by the reversed deployment way (Fig. 5). Because the collector is not deployed in node F, it does not send any resource information to the node G.

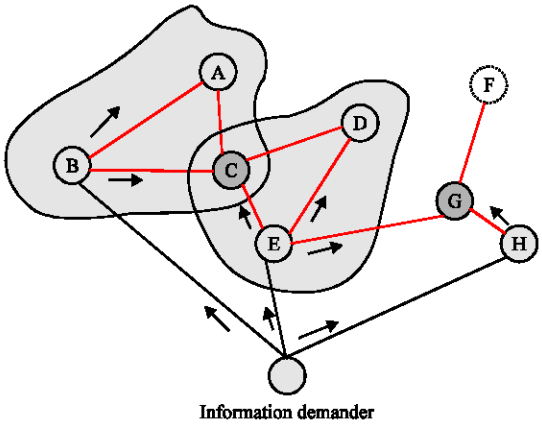

Fig. 4: Two-hops information collector dissemination process: issued by information demander, disseminates as arrows directions, arrives node C and G twice from different ways, does not arrive node F

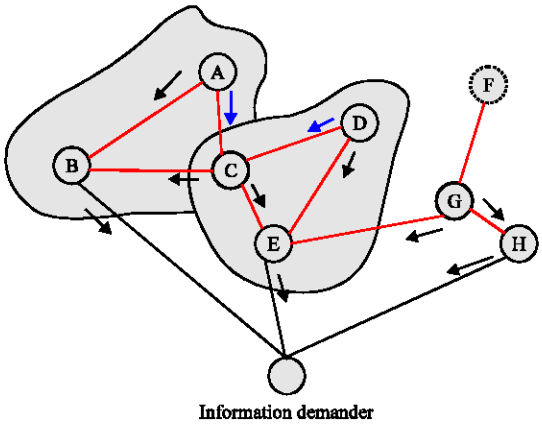

Fig. 5: Two-hops information collecting process: node information are collected and sent back in the reversed way of collector forwarding. Node A and D does not send information to node C because node C has not sent collector forward to them

Information collector

Definitions: In the PIVOT, a deployment of information collector is defined as formula (1).

Deployment = {collector, result, sules, hops, ttl} (1)

Deployment is the name of an information collector deployment. It is string measured up a fixed format <domain : sequenceNumber>. For example, the string cesc.zju.edu.cn:3201 denotes it is an information collector deployment issued by the host cesc.zju.edu.cn with identified number 3201.

The parameter collector in the formula (1) represents the name of information collector. A grid information demander may have more than one information collector for their multiple resources scheduling objectives. Generally, we name P with its executable file name. collector includes the executable program, runtime environment (such as OS type, OS version and necessary library) and program version, denoted as formula (2). collector is represented in XML style.

Collector = {executable, env, ver} (2)

The parameter result in the formula (1) represents the set of output raw data extracted by a collector. In result set, member data includes four fields as defined in formula (3).

Rawdata = {type, resources, attribute, value, timestamp} (3)

There is two data type, single and dual. Data of singe type indicate status of an independent resource. For example, {single, 10.21.202.2, FreeMemory, 54.6, 03202008142310}. Dual data indicate connection status between two resources, such as bandwidth of two hosts and communication latency between them. For example, {dual, (10.21.202.2, 10.21.3.7), Bandwidth, 95.3, 03202008142310}.

The parameter rules in the formula (1) represents the collection of semantics, knowledge and transformation rules.

Rules = {Ont, R} (4)

where, ont is ontology set used to capsulate transformation knowledge. R is used to represent relations among ontologies in Ont.

The parameter hops in the formula (1) represents the deployment distance. It is an integer. Because PIVOT adopts flooding deployment policy, hops is not so big usually. ttl is the time-to-live for collector.

Process of information collection: After deployed, information collector should be authenticated by the owner of remote grid resource for privacy. Then, it is registered as a legal native program. The collecting process includes five steps (Fig. 6).

The normalization process includes the following four steps:

Step 1: Discover raw dada of the resources (hardware, such as computer power, memory, storage, network and other peripheral equipments; software, such as operation system, compiler, DBMS, essential libraries, supported programs and data, such as database and related documents)

Step 2: Encapsulate raw dada into class style. The class is constructed according to information semantics delete that defined by collector. Standard semantics are applied to raw data. Common entity and attribute names are defined and used, regardless of the native terminology employed. The encapsulation adapts an extended LDAP namespace for identification, uses knowledge for filter, extraction and grouping and conforms to rules of the data

Step 3: Transform information in class style to XML format and merge them into a single XML document. The transformation and migration should use namespace, knowledge and rules of the data too. Furthermore some XML schemas and templates are needed. The schemas define information-oriented instantiation. While a naming schema provides a conceptual semantic description of a class of resource, the translation schema is an implementation of a naming schema for a specific translator

Step 4: Save the merged information into reservations and cache it. As we know, some information is static. The dynamic information depicted varied aspect of resource has varied changing periods. It is not necessary for all pre-configured information to update every time. We adopt an incremental saving policy

Step 5: Exchange information with direct connected neighbors

RESULTS AND DISCUSSION

Problem: As an example, a structure analysis of a crank using CSM is processed upon MASSIVE. After geometrical modeling, the crank is decomposed into 16 domains using Metis (Schloegel *et al*., 2002) and delivered to different processors for parallel execution and at last the data is visualized. The simulation is to show the model with displacement contour after mechanics problem solving computing and results merging. Figure 7 shows the corresponding geometrical model after discretizedation. The model is decomposed into 142070 meshes (elements). The simulation is a computation-bounded and communication-bounded task. Sixteen parallel processes will utilize networked three grid hosts to perform the simulation. The parallel processes must communicate with large-volume data periodically. Figure 8 shows the collaborative visualization of the crank simulation.

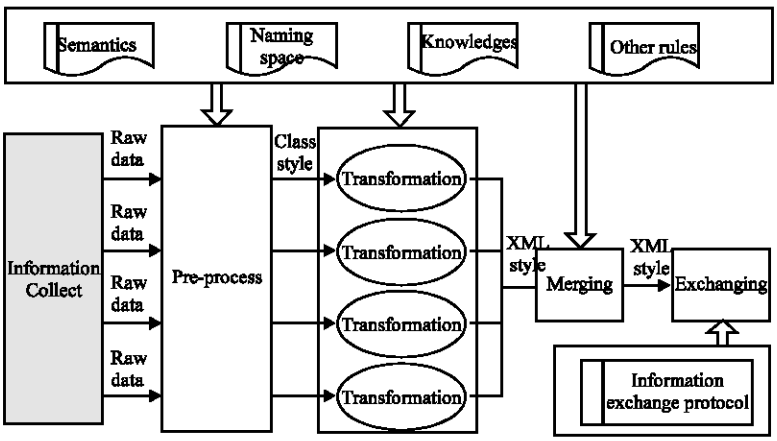

Fig. 6: The information collecting process

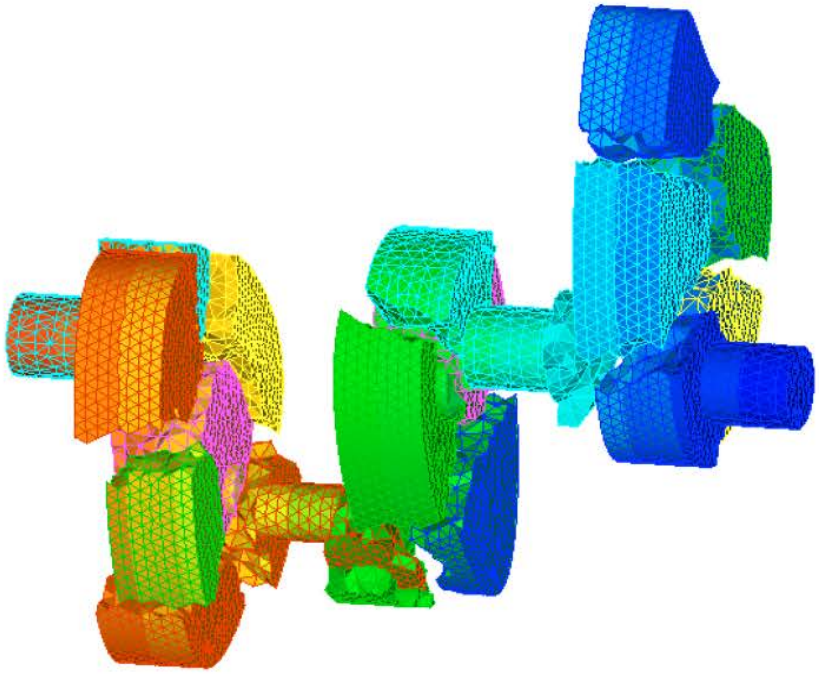

Fig. 7: Geometrical model after discretizedation

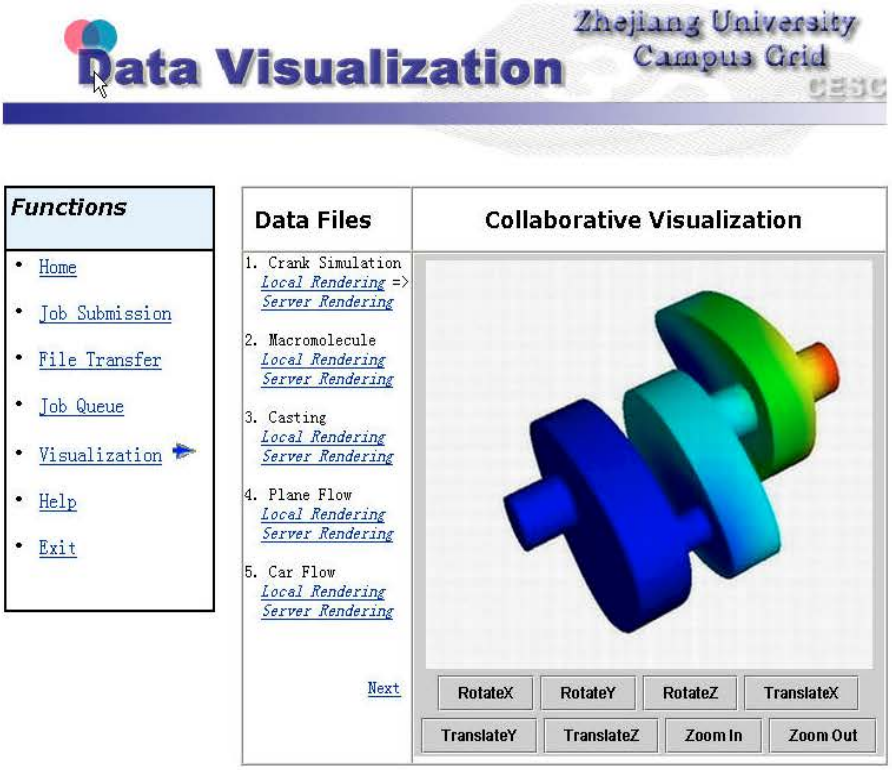

Fig. 8: Collaborative visualization of crank simulation

In this case, the information of resources network is very important for scheduling algorithm. But traditional grid information services can not provide information of resources network. A scalable information discovery service is need. We conduct this simulation on MASSIVE with PIVOT. An information collector integrated Network Weather Services (NWS) (Wolski *et al*., 1999) is developed to collect network information.

Resources: In order to test effectiveness of PIVOT, we construct a grid resources environment in MASSIVE grid. After deployment, there are eight valid hosts discovered for our crank simulation task (Fig. 9). In the Fig. 9, the edges are labeled with network bandwidth and resource nodes are labeled with computing power. For example, resource A can provide 5Gflops computing power, 150 M sec^{-1} bandwidth communicating with resource B and 200 MB sec^{-1} bandwidth communicating with resource C. The resource information is discovered by PIVOT dynamically according to application demands.

Scheduling algorithm

Definition of the object function: Now we define a proper function to measure the performance. Defining the function as a weighed sum of the communication cost and the job completion time is a good idea. It is necessary to give our clients the right to specify the weights for the communication cost and the completion time, respectively. For example, a certain user considers one unit of time as valuable as one hundred units of communication cost, then we set $w_m : w_t = 100: 1$ and $w_m : w_t = 1$. Here w_m and w_t stand for the weights of cost and time, respectively. The users can also set the deadline (denoted as T_0) and the maximum cost that can be afforded (denoted as T_0). Based on this logic, we define the object function in this way:

$$\min Z = w_m \cdot \sum_{i=1}^{N}\sum_{j=1}^{M}(C_j \cdot T_{ij} \cdot R_{ij}) + w_t \cdot T \quad (5)$$

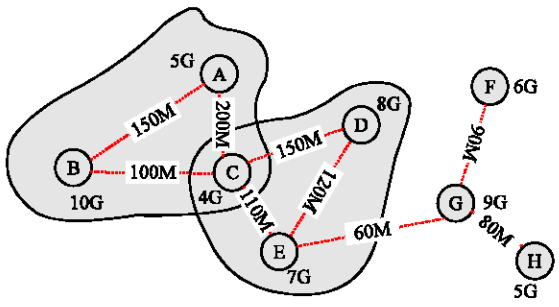

Fig. 9: Resource information discovered by PIVOT

Here, T_{ij} is the time spent on task X_i given that X_i is assigned to resource S_j. For each pair of i and j, as we have assumed, T_{ij} is known. C_j is the cost of resource S_j per unit CPU time. If we really assign task X_i to S_j, then the value of R_{ij} will be 1, otherwise be 0. T stands for the duration from the beginning of the first sub-task to the end of the last sub-task.

Constrained condition of the problem:

$$R_{ij} = 0 \text{ or } 1 \ (i = 1, 2, ..., N; j = 1, 2, ..., M) \quad (6)$$

$$T \le T_0 \quad (7)$$

$$\sum_{i=1}^{N}\sum_{j=1}^{M}(C_j \cdot T_{ij} \cdot R_{ij}) \le M_0 \quad (8)$$

$$\sum_{j=1}^{M} R_{ij} = 1 (i = 1, 2, ..., N) \quad (9)$$

$$\sum_{i=1}^{N} Rij \le 1 (j = 1, 2, ..., M) \quad (10)$$

$$T_k = \max_{ij} \{R_{ij} \cdot T_{ij} \cdot n_k\} + \max_{ij} \{R_{ij} \cdot T_k^{ij}\}, (k = 1, 2, ..., q) \quad (11)$$

$$T = \sum_{k=1}^{q} T_k + \max_{ij} \{R_{ij} \cdot T_{ij} \cdot (1 - \sum_{k=1}^{q} n_k)\} \quad (12)$$

Here, q stands for the times of communications of sub-tasks.

The meaning of R_{ij} in Eq. 6 is the same with that of Eq. 5. In Eq. 7 and 8 mean constrained. Equation 9 means that each single task should be assigned to one and only one computational resource. In Eq. 10 means that each single computational resource should process at most one of those sub-tasks. That is because communications cannot happen until every one of those sub-tasks has been finished by the same percentage. Equation 11 gives the value of the duration from the end of the (k-1)th communication to the end of the kth communication and n_k is the percentage amount of work completed during these two communications. For each k, t_k^{ij} stands for the time for the kth communication for task X_i assigned to machine S_j (The value of R_{ij} determines whether or not X_i is assigned to S_j). Equation 11 means that, another n_k (percentage amount) of every task has been finished and then the kth communication will happen. The reason why we use the first max is that the communication will not happen until all the tasks are ready for communication. $\max_{ij}(R_{ij} \cdot t_k^{ij})$ stands for the time spent on the kth communication. Here, we use max because of the

following reason: the quality of communications varies in different paths of network and it is the slowest path that determines the time for communications of the whole task. Equation 12 has given the value of the job completion time of N tasks. We use the term $1-\sum_{k=1}^{q} n_k$ because after the last communication, $1-\sum_{k=1}^{q} n_k$ of each task is left.

Except that R_{ij} should be binary integers, all the constrained conditions in our model are linear. Thus, the model is straightforward and is reduced to a classical Binary Integer Programming problem, for which a lot of methods are available.

In reality, the user may have some other requirements for some tasks, for example, reliability. In such cases, not all the resources in a grid are suitable for the tasks. If a certain machine is not suitable for a certain task, we set the corresponding term T_{ij} to be greater than T_0. In this way, our algorithm can avoid allocating task to those machines. Therefore, our algorithm can schedule tasks with QoS requirement.

Evaluation: Firstly, we schedule the computation task is scheduled with information services MDS provided by Globus Toolkit. MDS can only obtain independent hosts information, such as processor, memory and disk. Network topology and information related to communication are not found. So, scheduling algorithm runs and selects resources according to compute power only in the case. The scheduling result is that parallel processes are dispatched to node B, D and G. Finally, the simulation task completes with approximate 87 sec wasted.

Secondly, the same simulation is done with above algorithm. Because PIVOT is utilized to discover dynamic network status, the information is sufficient for the scheduling algorithm. The scheduling result shows that parallel processes are dispatched to node C, D and E. The simulation task completes with approximate 37 sec wasted.

CONCLUSION

Resource information service is an important part of any grid middle ware. It affects the effectiveness of grid resource scheduling. The task characters and resources requirements are varied largely in different engineering domains. A good scheduling algorithm is supported with adequate information. So, a scalable information service is important to computational grid. This study describes an adaptive information discovery framework (PIVOT) that operates over the wide area. It implements dynamic deployments of information collector for special information search with scalability and security. PIVOT provides an active information discovery scheme. Users can acquire customized grid resources information that fit into their special resources schedule algorithms. Present experiments demonstrate PIVOT improves the effectiveness of resources scheduling and the efficiency of resources utilization.

ACKNOWLEDGMENTS

This research was supported in part by the National Natural Science Foundation of China under Grants 60673179 and 60225009, Natural Science Foundation of Zhejiang Province of China under Grant No. Z106727, Grand Science Project of Zhejiang Province of China under Grant No. 2007C13068 and Science Fund for Distinguished Young Scholars of Zhejiang Gongshang University under Grant No. Q07-03.

REFERENCES

Almond, J. and D. Snelling, 1999. UNICORE: Uniform access to supercomputing as an element of electronic commerce. Future Gener. Comp. Syst., 15: 539-548.

Anderson, D.P., J. Cobb, E. Korpela, M. Lebofsky and D. Werthimer, 2002. SETI@home: An experiment in public-resource computing. Commun. ACM., 45: 56-61.

Andrew, S.G., W.A. Wulf and Corporate The Legion Team, 1997. The legion vision of a worldwide virtual computer. Commun. ACM., 40: 39-45.

Butler, R., V. Welch, D. Engert, I. Foster and S. Tuecke *et al*., 2000. A national-scale authentication infrastructure. Computer, 33: 60-66.

Chervenak, A., I. Foster, C. Kesselman, C. Salisbury and S. Tuecke, 2000. The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets. J. Network Comput. Appl., 23: 187-200.

Dijkstra, F. and A.J. Steen, 2005. Integration of two ocean models within cactus. Concurr. Comput. Pract. Experience, 18: 193-202.

Foster, I. and C. Kesselman, 1997. Globus: A metacomputing infrastructure toolkit. Int. J. High Performance Comput. Appl., 11: 115-128.

Frey, J., T. Tannenbaum, M. Livny, I. Foster and S. Tuecke, 2002. Condor-G: A computation management agent for multi-institutional grids. Cluster Comput., 3: 237-246.

Huai, J.P., H.L. Sun, C.M. Hu, Y.M. Zhu, Y.H. Liu and J.X. Li, 2007. ROST: Remote and hot service deployment with trustworthiness in CROWN Grid. Future Genera. Comput. Syst., 6: 825-835.

Li, Y. and M. Mascagni, 2003. Analysis of large-scale grid-based monte carlo applications. Int. J. High Perform. Comput. Appl., 17: 369-382.

Maria, L. and D.W. Walker, 2008. GECEM: Grid-enabled computational electromagnetics. Future Genera. Comput. Syst., 1: 66-72.

Schloegel, K., G. Karypis and V. Kumar, 2002. Parallel static and dynamic multi-constraint graph partitioning. Concurr. Comp. Pract. Experience, 14: 219-240.

Wei, G.Y. and Y. Zheng, 2004. MASSIVE: A multidisciplinary applications-oriented simulation and visualization environment. Proceedings of the IEEE International Conference on Services Computing, September 15-18, IEEE Computer Society Press, Shanghai, China, pp: 583-587.

Wolski, R., N. Spring and J. Hayes, 1999. The network weather service: A distributed resource performance forecasting service for metacomputing. J. Future Genera. Comput. Syst., 5: 757-768.