Proffering Task oriented Grid Resource Discovery based on Learning Automata

Ali Sarhadi
Department of Computer Engineering, Malayer Branch, Islamic Azad University
Malayer Iran

Hosein Zohrevand
Department of Computer Engineering, Malayer Branch, Islamic Azad University
Malayer Iran

Ebad Zohrevandi
Department of Computer Engineering, Malayer Branch, Islamic Azad University
Malayer Iran

Rasoul Rostaei
Department of Computer Engineering, Malayer Branch, Islamic Azad University
Malayer Iran

ABSTRACT
Main challenge of existing resource discovery service is the lack of support from task oriented query. This paper puts forward a design of task-oriented grid resource discovery service based on learning automata to enable users to dynamically discover the grid resources which are suitable for their task. The core of this service is learning automata based grid resource classifier, which periodically accesses the Meta computing directory service and dynamically classifier the grid resources into task-oriented categories according to the real-time state of grid computing environment. Users can invoke this service and pass her or his task type as a parameter to discover the current most suitable grid resources. Grid resource allocation manager also can interact with this service to improve its practicability and efficiency.

Keywords
Grid, resource discovery, Learning automata, task oriented.

1. INTRODUCTION

Computational Grid [3] [9] is a new paradigm in distributed computing which aims to realize a large-scale high performance computing environment over geographically distributed resources. Computational Grid enables the sharing, selection, and aggregation of highly heterogeneous resources for solving large scale problems in science, engineering and commerce. Numerous efforts have been exerted focusing on various aspects of grid computing including resource specifications, information services, allocation, and security issues. A crucial issue to meeting the computational requirements on the grid is the resource discovery [14] [15]. Resources on the grid are typically shared and undedicated so that the contention made by various tasks results in dynamically fluctuating delays, capricious quality of services, and unpredictable behavior, which further complicates the scheduling[1,2]. On the other hand, architectures of machines available in a Grid [3] are very diverse in specification to meet different task requirements; therefore, the extent to which a given task can exploit a given architectural feature depends on how well the task's computational requirements match the machine's advanced capabilities. In brief, Grid resource management faces to two major problems; one is matching computational needs to appropriate resources, and the other is exploiting resources over highly dynamic environment. MDS (metacomputing directory service) is a key component of globus, which provides users with functions to discover, register, query, and modify the information of grid computing environment. The MDS information process is composed of the dynamic descriptions of all kinds of resources in grid computing environment. So MDS reflects the real-time state of grid computing environment. Users can discover grid resources and get their attributes by invoking MDS. For example, by invoking MDS, users can query about which resources have the specified architecture, software, and/or network bandwidth to locate their wanted resources, or query about current states of physical attributes of some resource to get their wanted dynamic information of specified resource.

In this paper, our main concern is to address the limitation of existing MDS for most users. The task-oriented queries are much more useful, such as query about which resource is suitable for massive scale data mining, and/or which resource is suitable for massive online transaction process. Unfortunately, existing MDS doesn't support the task-oriented query. It is a serious limitation, so we should design a new service for users to discover the suitable grid resources according to their task. This paper designs a learning automata based task-oriented grid resource discovery service [16] (GRDS) to solve the problem which can not be solved by existing MDS. Furthermore, GRDS could effectively cooperate with GRAM (Globus resource allocation manager) to improve the efficiency and quality of grid resource allocation.

2. RESOURCE DISCOVERY BY MDS

Resources in grid environment may be shared by organizations and individuals. Grid users have little information about the pertinent resource. Therefore, they have limited efficacious usage of the resource in question. Grid information services [6] have been devised to

Support searches, resource discovery [4] and supervision of vital grid entity. There are various architectural plans for grid information services. MDS2 is the most sophisticated one that illustrated in fig.1.

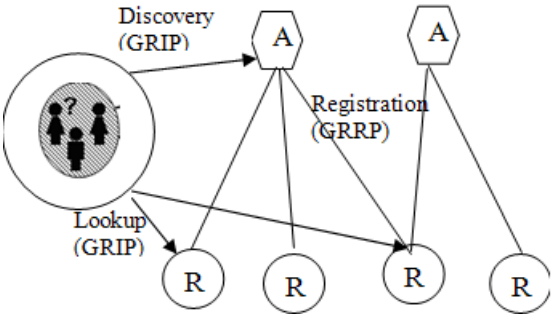

Fig. 1: Resource discovery by MDS

The functions of existing MDS include information generalization, information distribution, information storage, information search, information query and information display. There are two important components in MDS is GRIS (grid resource information service), which is a configurable information provider component.

The other one is GIIS (grid index information service), which is a configurable aggregative directory component.

The realization of existing MDS uses the LDAP (lightweight directory access protocol) as the unified interface to access grid information, meanwhile, it also support other protocols, such as SNMP (Simple Network Management Protocol) and NIS (Network Information Services), etc.

GRIS is a distributed information service, which is deployed into grid computing environment. It provides a uniform interface for clients to query about the configuration, capability and state of grid resources, such as query about static information including host computer name and version number of OS, etc., and query about dynamic information including available CPU and memory, etc.

GIIS provide away to combine all kinds of GRIS and also provides a coherent system mapping of those GRIS to facilitate the searches and queries generated by grid tasks and other clients. GIIS can differentiate resources by types, for example, GUS can list all the computing resources, and/or all the distributed storage systems of a specified virtual organization. The information which can be obtained from the existing MDS includes:

1) The information of computing resources: IP address, available software, system administrator, network connected, type and version number of OS, information of storage system, system loading, information of processes, and task queue, etc.

2) The information of network resources: network bandwidth, network protocols, network delay and network topology, etc.

3) The information of the infrastructure of Globus[5]: the information of host computers and resources administrator, etc.

According to the above description of MDS, we can find out that existing MDS supports the basic information queries very well, such as the queries about which resources have the specified architecture, software, and/or network bandwidth, etc. But do not support task oriented queries because users have to enter the specifications of the resource in query as query parameters. Therefore users themselves have to pinpoint which resource matches the tasks they have in mind. The above human recognition procedure can't be ameliorated owing to the dynamic and heterogeneous nature of the resource in grid environment. So users can't discover the suitable grid resources by passing the basic-information-based query conditions to MDS. So they need a grid resource discovery service which supports the task-oriented queries, such as the queries about which resource is suitable for page maker software, and/or which resource is suitable for massive data base transaction, and so on. Thus, users just need to pass their task types to GRDS to discover their wanted grid resources.

In this paper proposed a task oriented grid resource discovery service with learning automata at the core of this service that classifier resource with the task.

3. LEARNING AUTOMATA

Learning Automata [10] are adaptive decision-making devices operating on unknown random environments that show in fig.2. A Learning Automaton has a finite set of actions and each action has a certain probability (unknown to the automaton) of getting rewarded by the environment of the automaton. The aim is to learn to choose the optimal action (i.e. the action with the highest probability of being rewarded) through repeated interaction on the system. If the learning algorithm [13] is chosen properly, then the iterative process of interacting on the environment can be made to result in selection of the optimal action. Figure 1 illustrates how a stochastic automaton works in feedback connection with a random environment. Learning Automata can be classified into two main families: fixed structure learning automata and variable structure learning automata (VSLA) [11]. In the following, the variable structure learning automata which will be used in this paper is described.

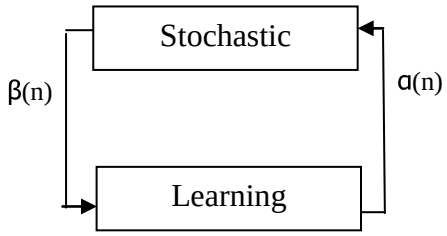

Fig. 2: The interaction between learning automata and environment

A VSLA is a quintuple < α, β, p, T(α,β,p) >, where α, β, p are an action set with s actions, an environment response set and the probability set p containing s probabilities, each being the probability of performing every action in the current internal automaton state, respectively. If the response of the environment takes binary values learning automata model is P-model and if it takes finite output set with more than two elements that take values in the interval [0, 1], such a model is referred to as Q-model, and when the output of the environment is a continuous variable in the interval [0, 1], it is refer to as S-model. The function of T is the reinforcement algorithm, which modifies the action probability vector p with respect to the performed action and received response. Assume $\beta \in [0,1]$. A general linear schema for updating action probabilities can be represented as follows. Let

action *i* be performed then:

$$p_j(n+1) = p_j(n) + \beta(n)[b/(r-1) - bp_j(n)] - [1-\beta(n)]ap_j(n) \;\; \forall j \;\; j \neq i \quad (1)$$

$$p_i(n+1) = p_i(n) - \beta(n)bp_i(n) + [1-\beta(n)]a[1-p_i(n)] \quad (2)$$

Where *a* and *b* are reward and penalty parameters. When *a=b*, the automaton is called L_{RP}. If *b=0* the automaton is called L_{RI} and if *0<b<<a<1*, the automaton is called $L_{R\varepsilon P}$. For more Information about learning automata the reader may refer to [10].

4. RESOURCE DISCOVERY SERVICE BASED ON LEARNING AUTOMATA

A new architecture based on the learning automata to pinpoint the resource discovery service in computational grid with regard to the task oriented of the resource discovery service has been proffered. The learning automaton has been opted for on account of the fact that their efficiency is much higher than other similar techniques in classification predicaments based on the obtained conclusions. The core of this service is a grid resource classifier based upon the learning automata which access Meta directory service on a periodic basis. It also classifier grid resource in terms of the diverse propounded tasks within the computational grid environment. Users can summon the aforementioned services to pass her/his task in query as the input search parameter instead of the specifications of the resource in query to select the best resource. Meta directory service reflects the situation of the computational grid environment and all the other information pertinent to the available entity in the grid. Users can determine their task as the input of learning automata. The output vector of the learning automates evinces the possibility of appropriateness of each one of the resource for the task in query. For instance the output vector is formulated based upon the ensuing relationship based upon the information obtained from a particular resource.

$T_k = (0.1, 0.1, 0.9\ldots 0.1)$

This output denotes the fact that the 3rd resource is more appropriate for this task than other ones.

5. The ARCHITECTURE OF TASK ORIENTED GRID RESOURCE DISCOVERY (GRD)

The architecture of task-oriented GRD is shown in fig.3.

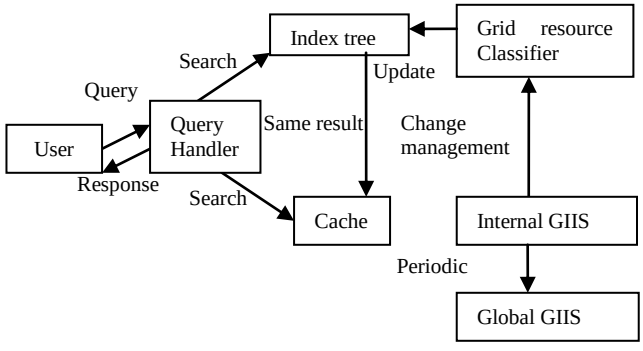

Fig.3: Architecture of Task-oriented GRD

As depicted in the above figure, the aforementioned service is composed of five components:

1. Internal GIIS

The operator of a grid resource discovery is an applied one. It summons global GIIS services on a periodical basis to obtain the situation of the grid environment and to become aware of any type of alterations in the resource status.

2. Grid resource classifier

Grid resource classifier is a chief component in task oriented grid resource discovery services. It can implement applied classifications of the resource based on the data received from grid resource through learning automata. GRC design will be expounded in the subsequent section. The above component effectuates the main task of the resource discovery service.

3. Index tree

This is a location where the conclusions produced by GRC are stored. The classification of the results from the grid resource is a pair amount. The identification key of the resource is in grid and specifies the predefined amount of the applied classification. The index tree is utilized to save such conclusions.

4. Cache

There are so many of resources in the grid environment. Thus we make a cache to save the results of the last searches to enhance the efficiency of the search. The search manipulator guarantees the matching of cache with the index tree.

5. Query handler

The task of this component is the processing of the users query. The query handler looks through cache at first. If it comes across the identifiers of the important resource of the use in cache, the query handler will delete all the counters in this resource. Then it checks the compatibility of the cache contents and the index tree. Then the identifier of the resource in question is rendered to the user, otherwise the search runner looks through the index tree and updates the cache based upon the new upshot obtained from the index tree and returns it to the user afterwards. The aforementioned amount is rewritten in cache and the pertinent counter is input as zero.

5.1 Grid resource classifier (GRC) based on learning automata

At the core of grid resource classifier learning automata are used. The reason for choosing the learning automata is their ability in choosing the best case according to specified criteria, that this action is done via repetition and comparison of acquired result with the previous ones. In this case input of learning automata is user's task and the action of automatons is choosing one of the resources for doing selected task. The criterion for choosing resource in this paper is choosing the resource that performs the proposed user's task at least time in comparison with other resource. After choosing resource by each automaton the environment gives reward or penalty to this action that the rate of reward 0.2 and the rate of penalty are 0.02. Based on experiment it is proved that with this rates regarding to final response have been quickly done.

5.2 Details of Designation

To the spreading of grid environment and number of resource that have been joined in it and dynamic property of grid environment we have consider separate pools for membership of resources. Every pools and resource members have an identifier. So that each pools can keep maximum of 5000 heterogeneous resource. After entering a new resource it is register in to a pool. If none of pool has necessary capacity for accepting the resource, grid resource management provides new pool for membership new resource. And if the resources of a pool are emptied, grid resource management will be deleting this pool. For increasing the selection speed of a resource for a proposed task, instead of using one learning automata for choosing resource, each of these pools is equipped with a learning automata and a assign queue that input tasks are placed in it. After entering and registrations a new task in assign queue of all pools, learning automata of each pool begin to discover the best existing resource in pool for performing proposed task. Each of automatons based on fig.4 chooses the best its pool resource for performing at least time in comparison with other pool resources. Then all executed time of automatons and its identifiers (for following in next state) are placed in a vector and minimum value of them is selected as chosen resource for executing proposed task.

1- Sorting the received requests in descending order in terms of the duration the pertinent Resource is required.
2- The beginning of the learning procedure in the learning automa connected to the pools
  2-1: reiterate them 5000 times
    2-1-1 Deplete the assignment queue of each pools
    2-1-2 Conduct the following stages for each input task
      2-1-2-1 Opt for the germane resource of each task and set the task in the designation queue of the pertinent pools
    2-1-3 The beginning of the fining procedure:
      Inflict financial penalty on each task who has opted for a resource that does not fulfill the Temporal limitation.
    2-1-4 The commencement of the rewarding phase:
      if a task is not fined and chooses a resource who is identical or earlier than the previous one, reward it With 0.02 rating.
  2-2 completion time of appropriate Resource of each pools(a resource that complete task at least time in comparing with other ones) with its identifier and pool's identifier(for tracking at the next stage) placed at a vector.
  2-4 select minimum value of vector and follow it with its Identifier and Pool's identifier.
3. Input task will be registered to the most appropriate resource subsequent to the convergence of results and the completion of Reiterations.

Fig.4 Resource discovery algorithm by learning automata

Environment response for learning automata associated to each pools on ideal conditions is complete time of input task by selected resource. For this propose environment uses of blew tow point for give reward or penalty to selected action by learning automata.

1- Run time of input task by selected resource shouldn't have many differences with other resources.

2- Each task must be assigned to resource that can run it earlier from other resources.

Thus the environment by using the first points will give penalty to selected action and with second point give reward to selected action by learning automata.

In fig.4, we must explain two important points: First, the above algorithm only specifies that the training process shouldn't be stopped unless meet the stop condition, but not specifies the stop condition, because in fact the training process would not be stopped for ever. The GRC always uses the current learned function to classify some grid resource in real time. According to the feedback of users, the GRC will update the current function to generate a new function by using this grid resource as a training instance to train itself.

The grid resources are highly dynamic, and their states are maybe different in any time. So, in theory, the instance space of grid resources is infinite. In order to improve the accuracy of classification, the GRC need to study continuously. So this algorithm important point would not be stopped for ever, we just continuously use the newest learned function. The result of GRC will be stored in index tree.

6. EXPERIMENTS

The goal of this experiment is to compare the performance of the task-oriented gird resource discovery to other conventional resource discovery approach such as p2p model [7], and the incoming task queries are matched with the next available resource offer which meets the task's constraints. We simulate grid environment [8] to evaluate experimentally the task-oriented gird resource discovery by means of gridsim[12].

The most suitable parameters for evaluation of this service have been propounded with regard to the effectuated securitizations. The error rate parameter expenditure improvement algorithms are regulated with regard to each selection for instance in failure cases of discovery, or in cases demanding rediscovery. Another parameter is the wait time that defines as duration of times before discovery of resource to a task and total time is complete time of a task. And finally efficiency of proffered algorithm in compared of conventional method will be shown. Reported results, is for average 20 times of simulation. Grid resource is high heterogeneous. We neglect the network topology and the communication costs associated with it. Instead, we assume that each of the users can submit applications to any of the resources. However, it is still adequate for certain situations. This experiment is to study task-oriented gird resource discovery and pricing algorithm in terms of conventional resource discovery efficiency. In this experiment, we choose respectively 100, 200, 600, and 1000 resource domains to compare resource discovery efficiency of task-oriented gird resource discovery service to the conventional approach under various the numbers of resource domains. The experiment results are shown in blew for resource discovery efficiency.

From the results in Fig. 5, when the number of resource domains is low, the difference in resource discovery efficiency between the two discovery methods is small, and both can achieve good discovery efficiency, so using conventional approach might be sufficient. When the number of resource domains increases, resource discovery efficiency of both methods will decrease. However, the resource discovery efficiency of approach might be sufficient. When the number of resource domains increases, resource discovery efficiency of both methods will decrease. However, the resource discovery efficiency of task-oriented gird resource discovery service decreases slower than that of the conventional approach. In other words, for a large number of resource domains, the conventional approach will not match the performance of the task-oriented gird resource discovery service. From above performance comparisons, we can see that the resource discovery efficiency of the task-oriented gird resource discovery method is effective improved.

As the Fig. 6 show Completion time of allocated job to the resource discovered by the proffered algorithm is less than to completion time of the same job allocated to other resources. But as shown in Fig. 7 the error rate in applying the proffered algorithm because of repeated learning automata to reach the desired resource is more. Fig. 6 shows the relation between completion time and task-oriented gird resource discovery method.

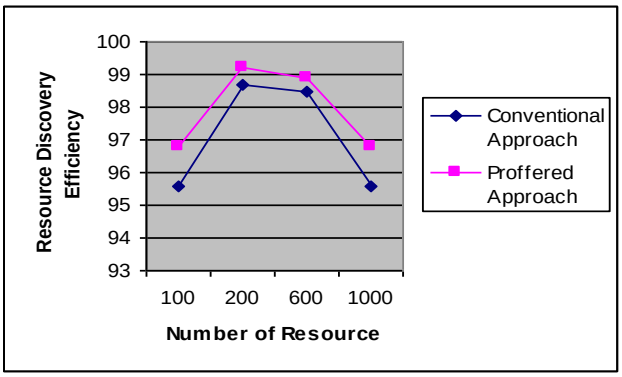

Fig. 5 Resource Discover Efficiency Comparison

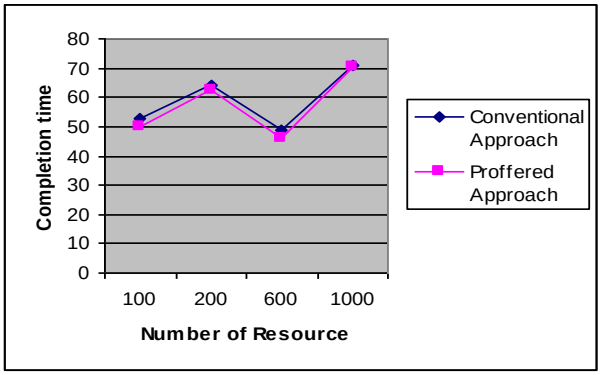

Fig. 6 Completion Time Comparison

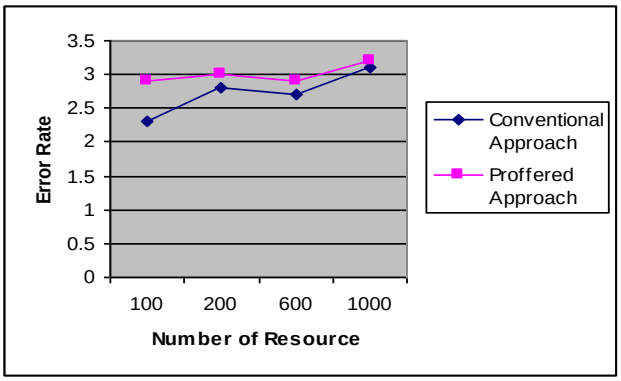

Fig. 7 Error Rate Comparison

7. CONCLUSIONS

In this paper, we put forward a design of learning automata based task-oriented gird resource discovery service to enable users to dynamically discover the grid resources which are suitable for their tasks. The core of this service is a learning automata based grid resource classifier, which periodically accesses the Meta computing directory service and dynamically classifies the grid resources into task-oriented categories according to the real-time state of grid computing environment. Users can invoke this service and pass the task type as a parameter to discover the current most suitable grid resources. Grid resource manager also can interact with this service to improve its practicability and efficiency. However, several aspects of the GRDS given in this paper still need to be researched, such as the time complexity of training process of learning automata based GRC, the space complexity of instance space of learning automata based GRC, the training algorithm, and emulation. These aspects will be the focuses of our future work.

8. ACKNOWLEDGMENTS

This paper is resulted of research project supported by Malayer Branch, Islamic Azad University Malayer Iran have been titled "Proffering application oriented architecture for resource discovery in computational grid using learning automata" under code (117). I am very grateful from Dr Majid Shams and Dr Behrooz Kord for their assistance during research.

9. REFERENCES

[1] T. L. Casavant and J. G. Kuhl, " A taxonomy of scheduling in general-purpose distributed computing systems," *IEEE Transactions on Software Engineering*, Vol. 14, No. 2, 1998, pp. 141-154.

[2] T. D. Braun, H. J. Siegel, et al.," A taxonomy for describing matching and scheduling heuristics for mixed-machine heterogeneous computing systems," *Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems*, 1998, pp. 330-335.

[3] I. Ekmecic, I. Tartalja, and V. Milutinovic, "A taxonomy of heterogeneous computing," *IEEE computer*, Vol.28, No.12, 1995, pp.68-70.

[4] Iamnitchi and I. Foster, "On Fully Decentralized Resource Discovery in Grid Environments," IEEE International Workshop on Grid Computing,Denver, CO, 2001.

[5] The GridLab Project, http://www.gridlab.org

[6] R. Al-Ali, O. Rana, D. Walker, S. Jha, and S. Sohail, "G-QoSM: Grid Service Discovery using QoS Properties", *Computing and Informatics Journal*, Special Issue on Grid Computing, vol. 21, no. 4, pp.363–382, 2002.

[7] Domenico Talia, Paolo Trunfio and Jingdi Zeng, Peer-to-Peer Models for Resource Discovery in Large-Scale Grids: A Scalable Architecture, May 2004

[8] R. Buyya, "Interactive Class Hierarchy Diagram of Economic Grid Resource Broker Simulated using the GridSim Toolkit", http://www.buyya.com/gridsim/doc/gridbroker/, Dec. 2001.

[9] 17- I. Foster, C. Kesselman and S. Tuecke, "The anatomy of the Grid: Enabling scalable virtual organizations", *International Journal of Supercomputer Applications*, 2001.

[10] 20- K. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction*, Prentice Hall, Englewood Cliffs,New Jersey, 1989.

[11] 21- R. Mirchandaney and J. A. Stankovic, "Using stochastic learning automata for Buyer scheduling in distributed processing systems", *Journal of Parallel and Distributed Computing*, pp. 527-551, 1986.

[12] 22- R. Buyya and M. Murshed, *GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing*, Technical Report, Monash University, Nov. 2001. To appear in the Journal of Concurrency and Computation: Practice and Experience (CCPE), 1-32pp, Wiley Press, May 2002

[13] 25- Kargupta, H. & Ghosh, S. (2002). Toward Machine Learning Through Genetic Codelike Transformations. *Genetic Programming and Evolvable Machines*, **3**(3), 231-258.

[14] A. Iamnitchi, and I. Foster, "On fully decentralized resource discovery in Grid environments", IEEE Int. workshop on Grid computing, Denver, 2001.

[15] Universal description discovery and integration (UDDI). http://www.uddi.org, 2001.

[16] S. Venugopal and R. Buyya, "A Market-Oriented Grid Directory Service for Publication and Discovery of Grid Service Providers and their Services", in Journal of Supercomputing, Kluwer Academic Publishers, USA, February 2004.