Journal of Software Engineering and Applications, 2014, 7, 256-263
Published Online April 2014 in SciRes. http://www.scirp.org/journal/jsea
http://dx.doi.org/10.4236/jsea.2014.74026

A Fault Tolerance Algorithm for Resource Discovery in Semantic Grid Computing Using Task Agents

Masoud Barati[1], Soheil Lotfi[2], Azizallah Rahmati[1]

[1]Department of Computer Engineering, Kangavar Branch, Islamic Azad University, Kangavar, Iran
[2]Department of Computer Engineering, Kermanshah Science and Research Branch, Islamic Azad University, Kermanshah, Iran
Email: emsbarati@yahoo.com, soheillotfi1983@gmail.com, m_aziz_rahmati@yahoo.com

Received 29 January 2014; revised 25 February 2014; accepted 5 March 2014

Abstract

One of the interesting topics in grid computing systems is resources discovery. After the failure of a resource in a chain of resources made for a specific task in grid environment, discovering and finding a new resource that reconstructs the chain is an important topic. In this study, with defining new agent that is called task agent, and by proposing an algorithm, we will increase the fault tolerance against probable failure of a resource in the resource chain.

Keywords

Grid Computing, Resource Agent, Task Agent, Ontology, Semantic Similarity Function

1. Introduction

One of the distributed computing systems is grid computing. In these systems, resources (*i.e.*, CPU, memory, disk capacity, network bandwidth, devices, software) are shared among different organizations [1]. In these systems, available resources are heterogeneous and are distributed geographically. We are seeking to access these resources and make them integrated. In a grid computing system, resources may be added or removed [2]. In these systems, besides some important issues such as sharing, allocation and integration of resources, the discussion of resources discovery is very important. Regarding resources discovery in these systems and with these properties, several methods have been presented, such as semantic description for displaying (presenting) resources [3]-[5]. In these methods, a set of established contracts is used for presenting the resources. They help to

How to cite this paper: Barati, M., *et al*. (2014) A Fault Tolerance Algorithm for Resource Discovery in Semantic Grid Computing Using Task Agents. *Journal of Software Engineering and Applications*, **7**, 256-263.
http://dx.doi.org/10.4236/jsea.2014.74026

increase flexibility. Among the methods presented, a group in which there is not an overload of transaction and in which resource discovery is decentralized has a considerable importance. In these methods, each resource has a resource agent, each of which interacts with its neighboring agents, by semantic description and based on local knowledge, and forms a chain of resources dynamically for completing a specific task [3]. After completing the resources chain, one of the agents may encounter failure for some reasons and leave the chain. Here, we try to correct some chains with this problem by presenting a new solution. In our solution, we define a new agent named task agent. Task agent can generate sub agents that are responsible for the interaction with the resource agent to be aware of its failure or lack of its failure. When a resource agent encounters with a failure, the sub agent interacting it will calculate the connectivity components and try to find a resource agent that has the required recourse with the help of other resource agents that are in the chain. By choosing it, the connectivity of chain will be maintained. The rest of this article consists of three sections. In Section 2, some definitions and primary concepts will be presented. Section 3 will present our proposed algorithm for increasing fault tolerance against the failure of a resource agent in a resource chain, and Section 4 will analyze the results of the algorithm. Finally, the conclusion and future work are highlighted in Section 5.

2. Preliminaries

Before presenting the algorithm that increases system tolerance against failure of a resource agent, it is necessary to offer some primary definitions and concepts as follows.

2.1. Connectivity Components

Graph G is connected if for any two nodes u, v, there exists a path between them (**Figure 1**). Each connected sub-graph G is a connected component if it is not included in a wider connected sub-graph. For example, the following graph is connected.

In the above graph, if node v_2 is eliminated, two connected components will be formed as follows (**Figure 2**).

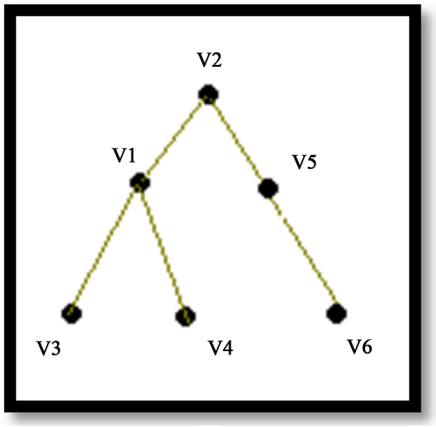

Figure 1. A connected graph.

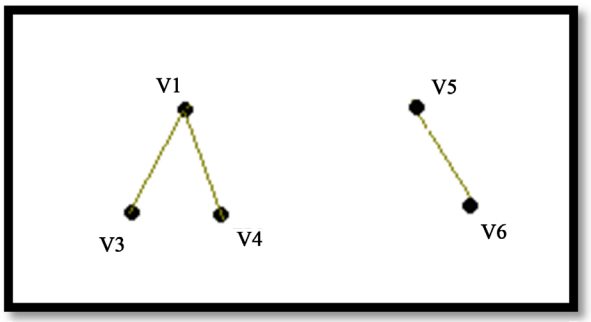

Figure 2. Two connected components.

2.2. Ontology and Semantic Similarity Function

Ontology is a formal structure which includes information about semantic description of data and a set of concepts and relations between them. It is used for retrieving information about user queries. Ontology is defined formally and in a specific domain O, as follows:

$$O = \{C, \leq c,\ R, \leq r, A\},$$

where C, R, ≤ c, and ≤ r are a set of concepts, a set of relations, an order on c, and a partial order on R, respectively. In the above definition, A is a set of axioms [6]-[9]. Semantic similarity function is used for computing similarity between two concepts. Similarity between concepts displays the degree of commonality between them. Similarity function is defined as: sim(x, y): $c \times c \rightarrow [0,1]$. The output of the function is a real number in the interval [0,1] that displays the degree of similarity between two concepts x, y, in which 0 denotes no commonality and 1 indicates complete similarity between the two concepts [10]-[14]. In this paper, we assume the threshold of semantic similarity function to be 1.

2.3. Model Description

In this paper, a grid is modeled as a graph each of the nodes of which defines a resource agent and each of the agents carries a specific resource type. We assume k different resource types existing in the system, which are displayed as $\{R_1, \ldots, R_K\}$. It is noteworthy that resource agents have the capability to find their neighboring nodes to form resource chains needed for tasks. Tasks are distributed with a certain rate μ in the network and each of them needs m different types of resources ($m \leq k$). Supposing that p tasks are distributed in the network, the tasks can be displayed as a $p \times m$ matrix, in which every row represents a task and every column the type of the required resource. As stated in [3], each node (resource agent) may be in one of the three states "inactive", "committed" and "active". Inactive state means the node is a free node and not working on any task. Committed state indicates that the node is working on a task but the resource chain has not been completed. Active state means that the node is working on a task and the resource chain has been completed. Here, we can define semantic similarity matrix as an $n \times n$ matrix whose elements represent the degree of similarity between resource type needed for the task and resource type of the resource agent. Regarding the supposition that semantic similarity threshold is 1, the elements of semantic similarity matrix are 0 and 1, in which 1 represents total similarity of resource types and 0 indicates dissimilarity of the resource type required from that of the resource agent. The above model corresponds to the model presented in [3].

2.4. Resource Agent

Resource agents interact with their neighboring agents based on their local knowledge, and are able to form a resource chain for a specific task. Each resource agent is identified by a unique code. Each of the resource agents has a field called connectivity field. In addition each of resource agents has the following information:

1) Unique code of neighboring nodes.
2) Resource type that resource agent carrying it.
3) Number of tasks that neighbors of resource agent working on.
4) The value of neighbors connectivity field.

This point should be noted that the value of connectivity field is zero at default. When a resource agent in a chain encounters failure, the chain may separate into multi connectivity components. Number of each connectivity component will register in agent's connectivity field which is stated at the component.

2.5. Task Agent

Every loaded task has many required resource types, and an agent named task agent will be create due to every loaded tasks. After comparing its resource type with other present recourse types in task agent and having necessary situations for standing in commit states, recourse agent send a message to task agent announce that it can do one of its required resources. when task agent receive this message it generate a sub agent and delete this recourse type from its task list and register it in the created sub agent. Then it makes this sub agent responsible for interact with resource agent to be aware of its failure or lack of its failure. While failure of a resource agent the interacted sub agent calculate the connectivity components to the failed agent and with help of other resource

agents that are in the chain try to find the resource agent that has the required recourse and with choosing it connectivity of chain will be maintained.

After interaction between resource agent and task sub agent, it give to the task sub agent the information of its resource agent that referred to it in Sections 4 - 2. In addition to this information, task sub agent has a field named counter that show the number of connectivity components of chain resulted from failure of resource agent. (pay attention that at default the number of this field is zero and when the resource agent encounter with failure amount of this field will change by task sub agent).

3. Proposed Algorithm

As mentioned every task sub agent with resource agent that working on it, is interacting and is aware of failure or lack of failure in recourse agent. While formation of the chain of recourses, one of the recourse agent of chain that is working on T1 may encounter with failure, first we should determine chain connectivity components that are created based on failure of considered agent. For determination of connectivity components, the following algorithm operate by task sub agent that its reference resource agent encountered with failure:

1) set the counter field with 1.
2) if among neighboring resource agents of failure node there isn't any agent working on T1, and connectivity field be equal with 1, then run the stage 5.
3) select one of the neighbors that is working on T1 and its connectivity field is equal with zero. And send this message to it "set your connectivity field with counter" then wait for a message from considered neighbor containing completion of i component formation.
4) Possibly after receiving message of completion of i component formation from considered neighbor add counter field an unit and run stage 2.
5) select one of neighbors that its connectivity field equal with counter-1 and send the required resource type to it.

After receiving a message from task sub agent or resource agent that cause change of the amount of connectivity field from zero to i or receiving a message for completion of a connectivity component in the view of resource agent of a neighbor, following algorithm can be operate by resource agents of chain:

1) If there isn't any neighbor working on T1, and connectivity field is equal with zero send to sender (that is a resource agent or task sub agent) a message containing formation completion of i connectivity component in the view of resource agent.
2) select one of the neighbors working on T1 that its connectivity field is zero and send a message containing "match your connectivity field with my connectivity field" to it.

In this section all of the connectivity components of chain that their number is counter-1 form in the view of resource agent encountered with failure and task sub agent run stage 5 of algorithm. Then each of the resource agents with receiving the required recourse type from task sub agent or resource agent of neighbor run the following stages:

1) If there is a free resource agent in the neighbors that has the required resource type, select it and send the amount of your connectivity field to it else run stage 3.
2) After receiving connectivity field from resource agent of neighbor check the neighbors connectivity field that are working on T1 and if in the value of connectivity fields of neighbors exist all connectivity components (means one to counter-1), set your state with commit state.
3) If there is a neighbor that is working on task T1, send the required recourse type to it and then run stage 1.

For example we assume there is a network with 16 nodes from A to P and 4 various resource type with numbers of 1 to 4 that are distributed uniformly in the network (for example node A2 is a resource agent that its resource type is 2). Also, we assume that task T1 that need 4 kind of resource type (1, 2, 3, 4) loaded in the network and a chain of resources like **Figure 3(a)** forms. So if node D that has resource type one encounter with failure (**Figure 3(b)**), then task sub agent from task agent T1 interact with resource agent that encountered with failure it is aware of failure and algorithm operate as follow(in the below figure CF means connectivity field and nodes that are in the commit state, are green).

- At first the task sub agent from task agent T1 that interact with node D1, set its counter with 1 and among D neighbors that are working on task T1 and the value of their connectivity field is zero means (A2, N3, B4) select a node for example A2 and send it a message containing "match your connectivity field with counter

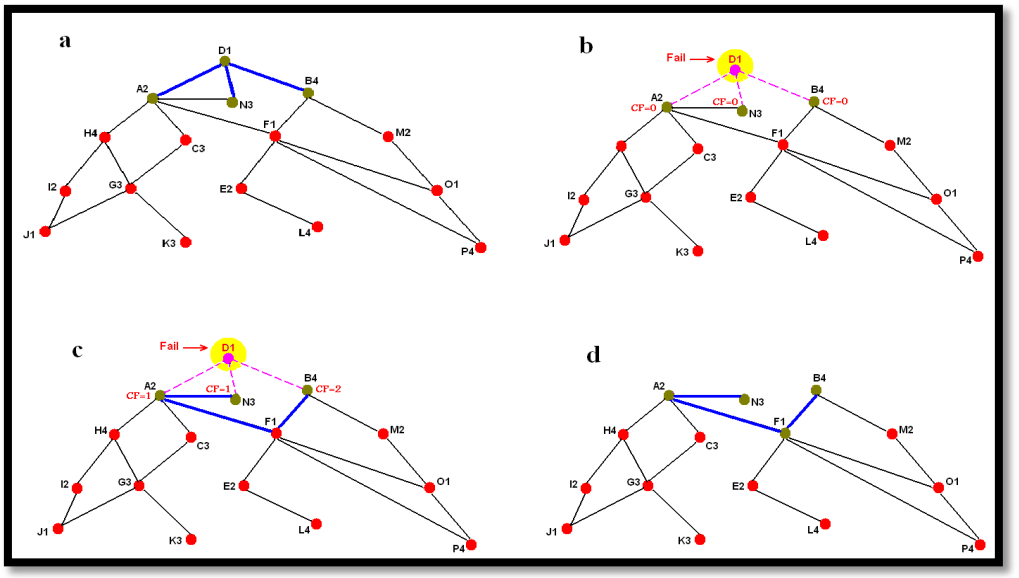

Figure 3. Scheme of the algorithm for chain reforming.

value (means 1)" and wait for receiving a message containing completion of formation first connectivity component from A2. After receiving message from task sub agent node A2 check its neighbors, and as between its neighbors node N3 is working on task T1 and value of connectivity field is equal with zero send a message containing " match your connectivity field with my field connectivity" then wait for receiving message containing completion of connectivity component from N3.

- Node N3 After receiving message and matching its connectivity field with value of connectivity field A2, check its neighbors and because it doesn't has any neighbor working on T1 and its connectivity field is zero, send a message to its neighbor containing "completion of first connectivity component formation". Node A2 After receiving message of completion of first connectivity component, send this message to the task sub agent that is interacting with node D1 that encountered with failure.
- Task sub agent that is interacting with node D1 among of the neighbors that are working on task T1 and their connectivity field are zero (here node B4 just has this condition) select one of them and after changing counter value from 1 to 2, send a message containing "match your connectivity field with amount of 2". then wait for receiving a message containing "completion of second connectivity component" from B4. node B4 after receiving the message from task sub agent and setting amount 2 in its connectivity field, check its neighbors and as there isn't any node its neighbors that is working on task T1 and its connectivity field is equal to zero, send a message containing "completion of second connectivity component" to the task sub agent, then after receiving this message by task sub agent, change its counter value to 3 (**Figure 3(c)**).
- Because there isn't any neighbor for task sub agent that working on task T1, among the neighbors of recourse agent that encountered with failure, the task sub agent that value of its connectivity fields counter-1, select B4 and send its required recourse type to it.
- After receiving the message from task sub agent, B4 check its neighbor and because node F1 is free and has the required recourse type 1, select it and send its connectivity field to it.
- Node F1 with receiving the value of connectivity field from B4 checks the connectivity fields of neighbors that working on task T1(B4, A2) and because it find in its connectivity field all of the values of set {1,2} (means all of the connectivity components) set itself in commit state.

Finally formation of connectivity chain takes place for T1 by A2, N3, B4, and F1 nodes (**Figure 3(d)**).

4. Simulation Results

The proposed algorithm can be applied to any network structure. Here, we used a small world network to run the algorithm. The small world network graph is created with 100 nodes and based on the methods in [15] [16]. In the simulation, k different resource types are uniformly distributed and then a task which requires m different

resource types ($m \leq k$) is assigned to the nodes in the network. After a chain of resources is formed for a task, we randomly cause a node to undergo failure, and then check the success or failure of re-forming the chain. In our simulation, we suppose that the values of task length (m) and the number of different resources types (k) range from 1 to 30, and for every state of task length and number of resource types, we run our simulator 30 times and calculate the average of success percentages. For example, in the case of task length 3 and number of resource types 3, success percentage is 80% and in the case of task length 5 and number of resource types 8, success percentage is 60%. By extending the network scale to 200 or 300 nodes and comparing **Figures 4-6**, it can be seen that with a fixed number of nodes and resource types and an increase in task length, the success percentage of reconstructing the resource chain reduces. Also, by increasing the number of resources types, the success percentage of chain reconstruction reduces. By comparing the following figures, another conclusion is that, with a fixed task length, when the number of nodes and resource types increases, the success percentage of resources chain reconstruction increases.

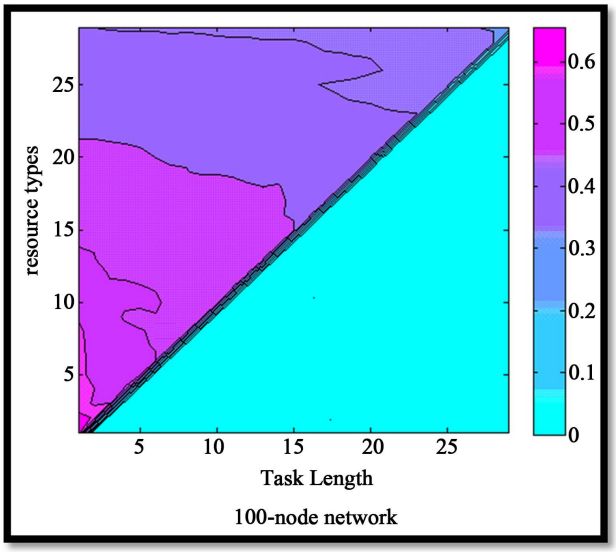

Figure 4. success probability under different resource types and task length.

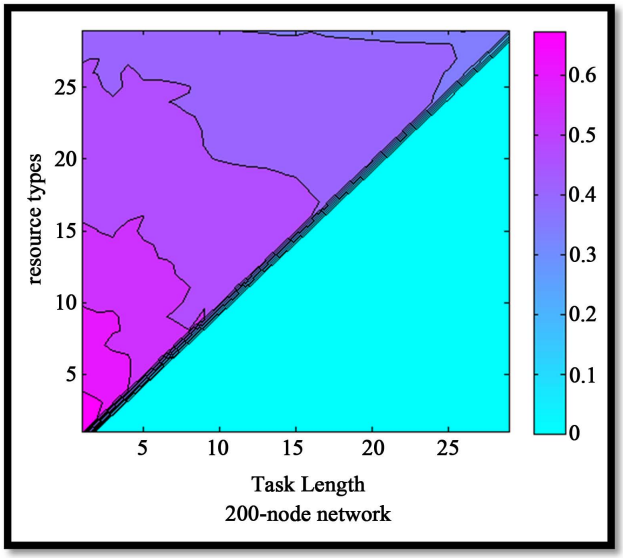

Figure 5. Success probability under different resource types and task length.

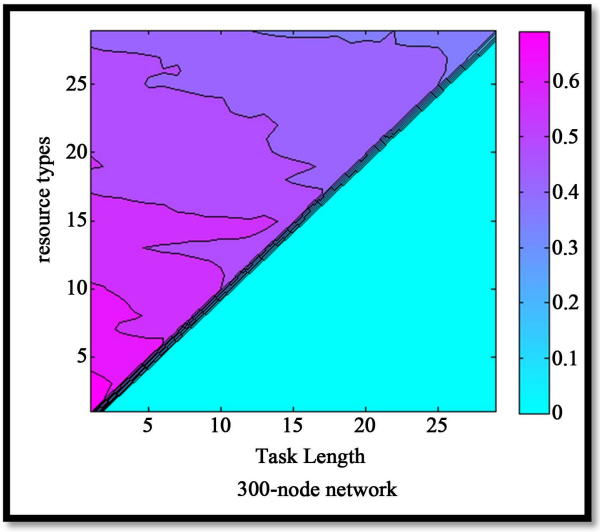

Figure 6. Success probability under different resource types and task length.

5. Conclusions and Future Work

In proposed method in [3], a resource chain for a specific task was formed. However, in it, there is no solution for fault tolerance when a resource agent in a chain encounters failure. In our paper, we tried to correct some chains with this problem by presenting a new solution. As mentioned above, in our solution when a resource agent in a chain encounters failure, the neighboring agents will be informed about the failure; by collaborating with task agent, a resource agent that has required resource type will be found.

The future work will focus on how we can apply our proposed method into one of our real e-Science projects for the resource management within grid computing.

References

[1] Foster, I. and Kesselman, C. (1999) The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, Massachusetts.

[2] Jacobi, B., Brown, M., Fukui, K. and Trivedi, N. (2005) Introduction to Grid Computing, an IBM Redbook.

[3] Liangxiu, H. and Dave, B. (2008) Semantic-Supported and Agent-Based Decentralized Grid Resource Discovery. *Future Generation Computer Systems*, **24**, 806-812. http://dx.doi.org/10.1016/j.future.2008.04.005

[4] Somasundaram, T.S., Balachandar, R.A., Kandasamy, V., Buyya, R., Raman, R., Mohanram, N. and Varun, S. (2006) Semantic-Based Grid Resource Discovery and Its Integration with the Grid Service Broker. Technical Report, GRIDS-TR-2006-10, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Melbourne, 84-89.

[5] Rahimzadeh, P., Barati, M. and alizadeh, R. (2010) A New Semantic-Supported and Agent-Based Decentralized Algorithm Forresource Discovery in Economic Grid. 3*rd International Conference on Advanced Computer Theory and Engineering*, Chengdu, 20-22 August 2010, V5-441- V5-445.

[6] Fensel, D. (2003) Ontologies: A Silver Bullet for Knowledge Management And Electronic Commerce. Springer, Berlin.

[7] Stuckenschmidt, H. (2002) Ontology-Based Information Sharing in Weakly Structured Environments. Ph.D. Thesis, AI Department, Vrije University, Amsterdam.

[8] Gruber, T.R. (2003) Toward Principles for the Design of Ontologies Used for Knowledge Sharing, KSL-93-04, Knowledge Systems Laboratory, Stanford University, Stanford.

[9] Andreasen, T., Bulskov, H. and Knappe, R. (2003) From Ontology over Similarity to Query Evaluation. In: Bernardi, R. and Moortgat, M., Eds., 2*nd CoLogNET-ElsNET Symposium*—Questions and Answers: *Theoretical and Applied Perspectives*, Amsterdam, Holland, 39-50.

[10] Resnik, O. (1999) Semantic Similarity in a Taxonomy: An Information-Based Measure and Its Application to Prob-

lems of Ambiguity and Natural Language. *Journal of Artificial Intelligence Research*, **11**, 95-130.

[11] Richardson, R. , Smeaton, A. and Murphy, J. (1994) Using WordNet as A Knowledge Base for Measuring Semantic Similarity between Words. Technical Report Working paper CA-1294, School of Computer Applications, Dublin City University, Dublin.

[12] Rodriguez, M.A. and Egenhofer, M.J. (2003) Determining Semantic Similarity among entity classes from Different Ontologies. *IEEE Transactions on Knowledge and Data Engineering*, **15**, 442-456. http://dx.doi.org/10.1109/TKDE.2003.1185844

[13] Seco, N., Veale, T. and Hayes, J. (2004) An Intrinsic Information Content Metric for Semantic Similarity in WordNet. *Tech. Report*, University College Dublin, Dublin.

[14] Tversky, A. (1977) Features of Similarity. *Psychological Review*, **84**, 327-352. http://dx.doi.org/10.1037/0033-295X.84.4.327

[15] Watts, D.J. and Strogatz, S.H. (1998) Collective Dynamics of Small-World Networks. *Nature*, **393**, 440-442. http://dx.doi.org/10.1038/30918

[16] Watts, D.J. (1999) Networks, Dynamics, and the Small-World Phenomenon, *American Journal of Sociology*, **105**, 493-527. http://dx.doi.org/10.1086/210318