Towards Autonomous Service Composition in A Grid Environment

William K. Cheung[+], Jiming Liu[+], Kevin H. Tsang[+], Raymond K. Wong[++]

Department of Computer Science[+]
Hong Kong Baptist University
Hong Kong
{william,jiming,hhtsang}@comp.hkbu.edu.hk

School of Computer Science & Engineering[++]
University of New South Wales
Australia
wong@cse.unsw.edu.au

Abstract

Web services are becoming important in applications from electronic commerce to application interoperation. While numerous efforts have focused on service composition, service selection among similar services from multiple providers has not been addressed. Such issue is more serious when services are embraced in Grid platforms, which are usually resource-conscious. Experimental results show that our considerations are valid and our preliminary solution works well in our Globus grid network.

Keywords: Autonomous services composition, Bidding, Web services, Grid computing

1. Introduction

Web services are becoming the prominent paradigm for electronic business and interoperable applications across heterogeneous systems. However, Web services standards such as WSDL [1], UDDI [2], and SOAP [3] do not address the issues of service re-use and composition, especially dynamic composition of existing services from multiple sources. Various efforts on addressing this issue including the recent initiative of BPEL4WS [4] focus on representing compositions, whereas, the actual issues involved in composing the services, e.g., the selection process and composition considerations such as run-time costs, etc., have not been considered.

Another technology that is getting increasing popularity is Grid [5]. Grid is a distributed environment that enables flexible, secure, coordinated resource sharing, among dynamic collections of individuals, institutions and resources. The benefit of embracing Web services on Grid have been recently realized in the Open Grid Services Architecture (OGSA) of Globus (GT3) – the de-facto standard of Grid middleware [6], and shown in various projects (e.g., Geodise - www.geodise.org, MyGrid - www.mygrid.org.uk). However, Grid platform is in general more conscious regarding the utilization and reliability of resources, and services composed in Grid need to be planned in an optimized way. Along this line and different from previous works, this paper attempts to investigate the underlying criteria in practice, propose an initial solution using a bidding-like mechanism, and finally realize its significance by implementing the solution (called BU-Grid) and running series of experiments. Experimental results are encouraging and further improvements shall be obtained from our ongoing effort.

1.1 Related Works

Due to the increasing attention to Web services from the research and industry communities, there have been lots of recent works addressing various issues of Web services (e.g., [7]). To name a few, for example, in [8], the issue of service composition is addressed in the context of Web components, as a way for creating composite Web Services by re-using, specializing and extending existing ones. McIlarith and Son [9] proposed an approach to building agent technology based on the notion of generic procedures and customizing user constraints. They argue that an augmented version of the logic programming language Golog provides a natural formalism for programming Web services. Prototypes that guide a user in composing Web services in a semi-automatic manner have been proposed in [10,11]. The semi-automatic process is facilitated by presenting matching services to the user at each step of a composition and filtering the possibilities by using semantic descriptions of the services. While there are numerous papers describing specifications and methods for service composition, seldom of them have addressed the issues of choosing services based on its costs and resources (which is an important issue in utilizing resources in a Grid

environment). For instance, [12] mentioned a simple scoring service based on the summation of the services' weighted scores. However, the details of estimating the scores and evaluating criteria (which are crucial in the actual implementation and system evaluation, again, especially in Grid) have been left out. Blythe *et al*., in [13], used limited state information (the current data storage of the distributed hosts) for optimizing services compositions for e-Science applications. The work closest to ours is due to Sample *et al.* [14] that incorporated services uncertainty (e.g., costs, performance, reliability) via probabilistic modeling in the composition process.

1.2 Paper Organization

The remaining of the paper is as follow. Section 2 gives a typical environment for autonomous service composition. Section 3 describes in detail the overall system architecture of BU-Grid. Section 4 provides in detail a bidding mechanism for service selection in a dynamic Grid environment. Experimental results and the lessons learnt are found in Section 5 and 6, respectively. Section 7 concludes the paper with a number of future research directions.

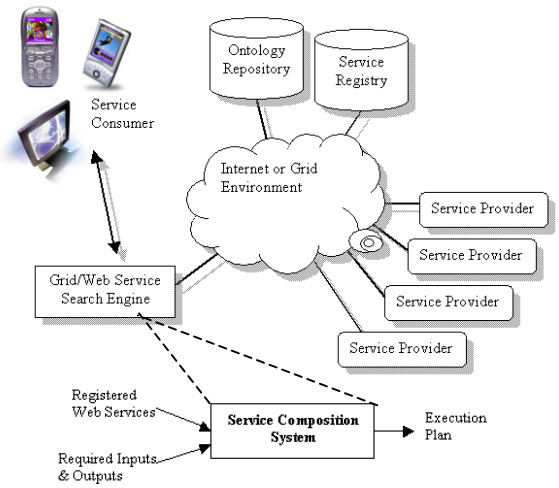


Figure 1. A typical service composition environment.

2. Autonomous Service Composition

A typical environment for supporting Grid/Web service composition is illustrated in Figure 1. A collection of *service providers* expose, via the Internet, the services they support as Web services. The services are *registered* at a *service registry* (e.g. UDDI) for *service discovery*. The semantics of the available Web services (e.g., the semantics of the input/output parameters) are described by some machine understandable semantic Web language (e.g. OWL-S). Relationships and concepts of the vocabularies used to enable semantic matching of services are shared in a *ontology repository*. A *service consumer* is a client program which sends service requests (e.g., in terms of desired input/output relationships) to the Grid/Web service broker which bears the duty of selecting suitable primitive services, composing them as well as monitoring their execution.

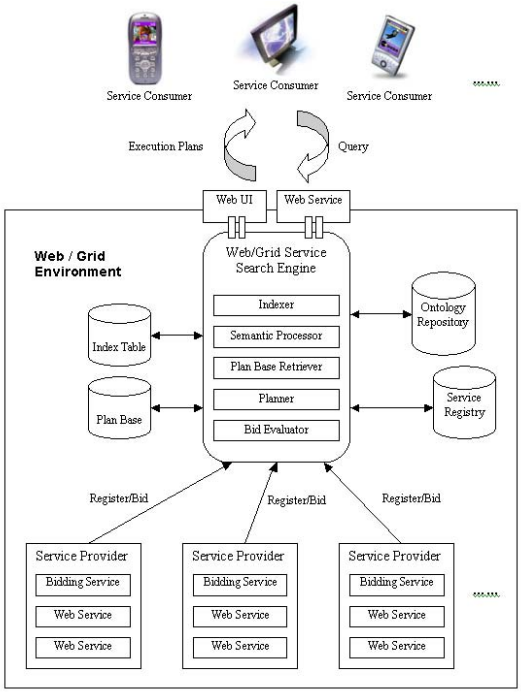


Figure 2. The system architecture of BU-Grid.

3. BU-Grid System Architecture

The architectural design of the proposed BU-Grid, to be further described in the following (also see Figure 2 for an overview), contains components that are common in most of the service composition systems. In addition, it is featured by the incorporation of a) bidding services and bid evaluation components for dynamic service selection, as well as b) a plan base and a plan retriever for plan re-use support. While the focus of this paper is to study in detail how the state information can be used to form the selection criteria and optimize the overall system utilization via a bidding mechanism, details about the planning part and its relationship with the proposed service bidding mechanism will also be included for completeness.

3.1 Service Registration and Indexing

Semantic descriptions of Grid services are stored at the Service Registry, which may include:

- High-level services descriptors: E.g., for e-business applications, they can be company name, business nature/categories, contact person, phone number, email address, etc.
- Low-level services interface descriptors: E.g., service name, functional description, URL of the WSDL file or Grid Service Handle (GSH), semantics of the input/output parameters, etc.

To support efficient access of GSHs from the Service Registry and efficient update of the services' state information, both the high-level and low-level service semantics are indexed. Furthermore, to extend the service discovery capability to go beyond simple keyword search, different domain-specific ontologies are maintained in Ontology Repository to support semantic matching.

3.2 Task Specification & Service Composition

In BU-Grid, a task is represented by specifying the required input[1] and desired output. To plan for the task (or to satisfy the specification), a meta-level service will be composed on-demand using the primitive services available in the Service Registry.

By treating the input as the initial state, the desired output as the goal, and the available services as the operators, service composition can readily be formulated as an AI planning problem [14]. Under the Grid context, one challenge is that the planning has to be performed in a dynamic environment, containing multiple functionally equivalent operators (services) but with possibly different implementations as well as time-varying resources. Besides, services matchmaking based on semantics is also a non-trivial task.

3.2.1 Services Matchmaking

To enable correct matchmaking between Grid services, we need to well-define services compatibility. There exist at least two types of compatibility measures, namely data type compatibility as well as semantic compatibility. Eq.(1) and Eq.(2) give two possible forms of compatibility in terms of data type and semantics between an output of a service and an input of a matching service.

a) Data Type Compatibility

$$Compatibility_t(type_{output}, type_{input}) = \begin{cases} 1 & same / upcast \\ 0.5 & downcast \\ 0 & otherwise \end{cases} \quad (1)$$

where "upcast" means the output has to be upcasted (e.g., from `int` to `float`) so as to be fed into the next input, and similarly for "downcast".

b) Semantics Compatibility

$$Compatability_s(semantic_{input}, semantic_{output}) = \begin{cases} 1 & equivalent \\ 0.8 & subclass \\ 0 & otherwise \end{cases} \quad (2)$$

where "subclass" means that the output is a subclass of the input and the need of ontology is explicitly implied.

3.2.2 Planning

Based on the services compatibility measures defined, service composition can be proceeded using different planning paradigms. One example is regression planning which is based on backward chaining. Starting from the output of the specified task as the ultimate goal, the planner can search the Service Registry for services with their outputs compatible with that of the specified task. It is possible that the set of compatible services can be categorized into several distinct *service interfaces,* each contains a unique input/output pair. One can then use those distinct service interfaces as sub-goals and continue to search for the best plan. Sometimes, for efficiency purpose, one may want to use a local search strategy by choosing one of the interfaces and continuing the search. The selection can be done based on a local performance estimation of the interfaces. See Figure 4 for an overview and refer to Section 6.2 for more discussion on dynamic plan optimization.

As one service interface is in fact representing a group of functionally equivalent services, its performance estimation should be characterized by the best service under the same interface. So, under this scenario, the remaining question is how to select the best service under the dynamic environment.

3.2.3 Service Selection

Services with equivalent input/output interfaces can have different implementations and have transient performance due to time-varying system load, data cached, etc. A mechanism for making a wise choice for better Grid resource utilization is needed. We believe that bidding based on a dynamic scoring scheme can be adopted for the service selection task, as detailed in Section 4.

[1] Sometimes, a task can be fully specified by only desired output, for example, accessing some processed e-Science data from the grid.

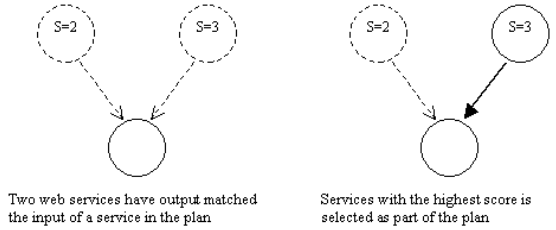


Figure 3. Service selection.

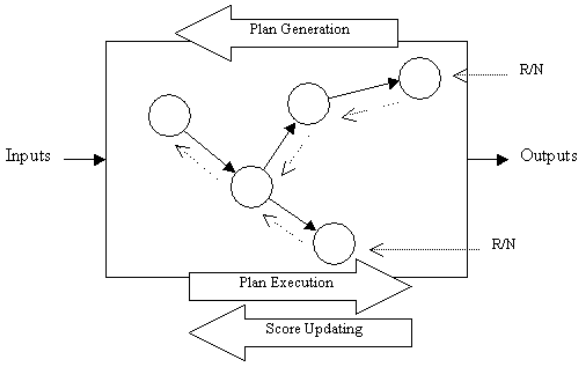


Figure 4. An overview of service composition and execution process.

4. Service Selection Via Bidding

Here we propose a bidding-like mechanism for the aforementioned service selection problem with the hope of balancing the load among a set of Grid nodes in a virtual organization.

4.1 Notations

Let I denote a particular service interface, $E_i(I)$ denote the estimated service time of the i^{th} implementation for the service interface I, $B_i(I)$ denote the value sent to the broker by the i^{th} implementation for bidding the interface I to be performed.

4.2 Bidding Process

The broker (search engine) first notifies each of the service providers that host the required service implementations. Being notified, each service implementation will make use of the current estimated service time $E_i(I)$ (track record) as well as the current system load (current resource) to compute a bid value as in Eq.(3) and send the bid back to the broker:

$$B_i(I) = (1 - L_i) \times \frac{1}{E_i(I)} \tag{3}$$

where L_i is the **system load** of the node hosting the i^{th} service implementation.

The broker then selects a service implementation according to the probability distribution:

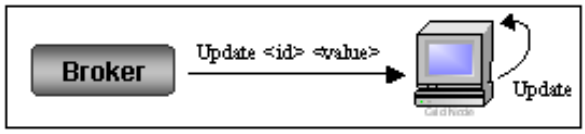
$$P(i) = \frac{B_i(I)}{\sum_i B_i(I)} \tag{4}$$

4.3 Estimation of Service Performance

After the selected implementation finished the assigned job, it will notify the broker the result. The broker will then return the actual service time A_i, and the estimated service time of the i^{th} service implementation will be updated as

$$E_i^{t+1}(I) = (1 - \alpha) \times E_i^t(I) + \alpha \times A_i \tag{5}$$

where α is the updating rate. In our experiment, its value is set to 0.8.



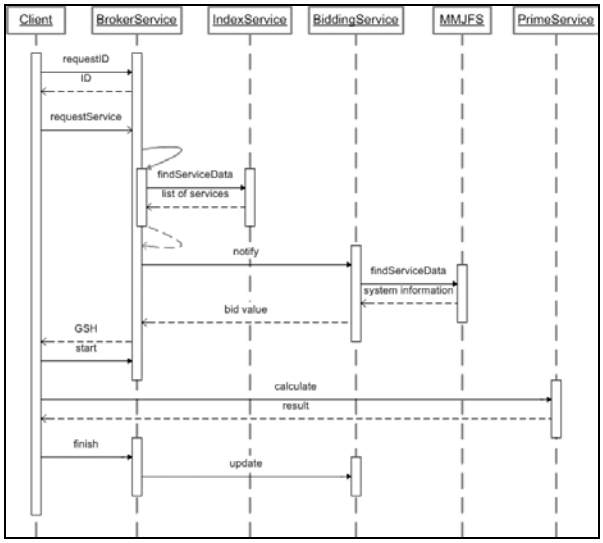


Figure 5. The sequence diagram of the bidding process.

5. Experiments

In order to study in detail the effectiveness of the proposed bidding process on the Globus platform and the behavior at each grid node, we have set up a small grid environment with four grid nodes, one being the Service Broker and the other three being the Service

Providers. Figure 5 shows the sequence diagram of the overall bidding process. All the Grid services are running in the service container provided by GT3. The BrokerService queries the IndexService of each Grid node to get the list of available service implementations. BiddingService consults MasterManagedJobFactoryService (MMJFS) of its own node to get the current system information. Three experiments have been conducted for evaluating three different virtual organization scenarios on the grid platform:

Experiment 1 assumes that the available service implementations (Service A) in all the nodes are homogeneous, and all the incoming service requests can be served by Service A.

Experiment 2 assumes that the available service implementations are heterogeneous, including Service A, B and C. The implementations of Service A and Service C are 3 and 2 times less efficient than that of Service B. Also, all the incoming service requests can be served by either the implementations of Service A, B or C.

Experiment 3 assumes that each node contains one composite service implementation and one primitive service implementation needed as part of the composite service. The service request stream is of homogeneous type and requests the Service Broker for the composite service A+2B regularly. The composite service A+2B means that it has to perform subtask A first before two subtasks B can be performed in parallel. A composite service request is said to be fulfilled only if all its subtasks are finished. Thus, there are in fact two levels of bidding as illustrated in Figure 6.

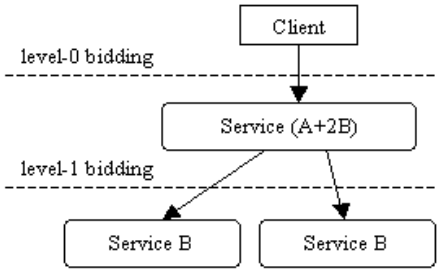


Figure 6. An illustration of a multi-level bidding needed by composite services.

The inter-arrival time for the service request was 20 seconds throughout the experiments. For performance evaluation, information like job start time, end time, system load, and service time are collected during the experiments and the results are shown in Figure 7-15

Given: service requests arrive at a 20 sec. interval			
	node-1 (CPU 2.6GHz)	**node-2** (CPU 0.65GHz)	**node-3** (CPU 0.7GHz)
Expt. 1 (homo.)	Service A	Service A	Service A
Expt. 2 (hetero.)	Service A	Service B	Service C
Expt. 3 (composite)	Service A+2B, Service B	Service A+2B, Service B	Service A+2B, Service B

Table 1. Experiment setups for performance evaluation.

Observation 1: While the three experiments were designed to correspond to three different virtual organization scenarios on the grid platform, our proposed bidding mechanism managed to distribute the service request streams to the three Service Providers for improving system utilization, as shown in Figure 8, 11 and 14.

Observation 2: In Experiment 1, as all the service implementations were homogeneous, node-1, being the most powerful machine, naturally shouldered more jobs via the bidding mechanism, when compared with the other two nodes.

Observation 3: By comparing Figure 7 (Expt. 1) and Figure 10 (Expt. 2), it is noted that the service implementations of both Service A and C being less efficient than that of Service B resulted in more jobs being assigned to node-2 which is hosting the more efficient service implementation Service B in Expt. 2, even though node-1 is the fastest machine. This reinforces the design of the proposed bidding mechanism that, other than the computing power, it should (implicitly) take into the consideration of the efficiency of the service implementation and react accordingly.

Observation 4: As we moved from Experiment 1 to 3, the overall load of the set of requested jobs was increasing (see Figure 8, 11, 14). We observed that all the grid nodes were moving closer to be full loaded at most of the time, which we believe to be an indicator of good resource utilization. However, the service time per job fluctuated quite seriously as the overall load increases (see Figure 14). We believe that the fluctuation is caused by the time dependency requirement of the composite services. We are still investigating the conditions and bidding strategies for reducing the fluctuation, and thus improving the service reliability.

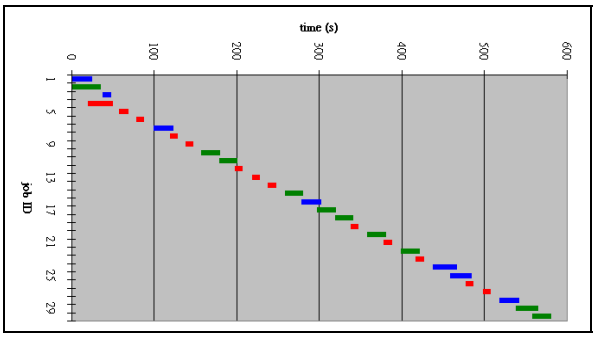
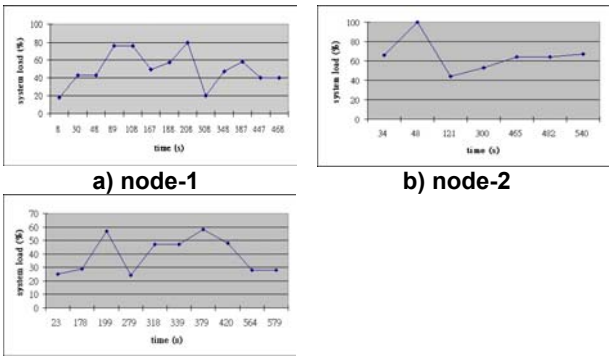


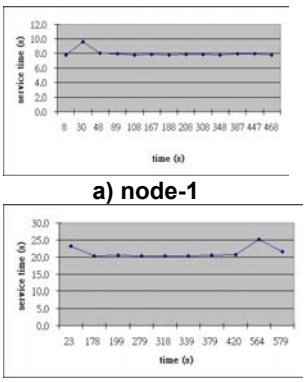
Figure 7. Job schedules under homogeneous services scenario (red for node-1, blue for node-2, green for node-3).



a) node-1 **b) node-2**

c) node-3

Figure 8. System load of each grid nodes under homogeneous services scenario.



a) node-1 **b) node-2**

c) node-3

Figure 9. Job service time of each grid nodes under homogeneous services scenario.

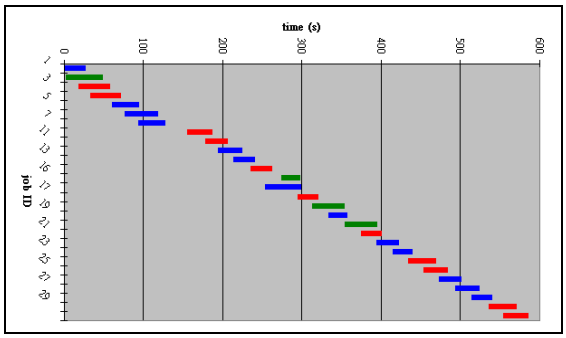
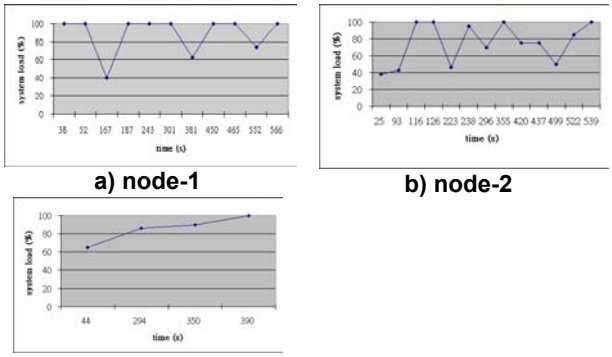


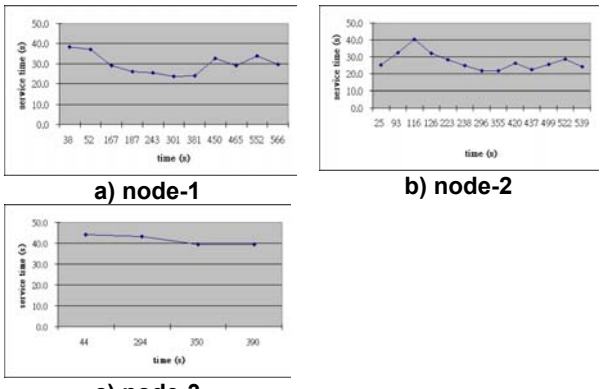
Figure 10. Job schedules under heterogeneous services scenario (red for node-1, blue for node-2, green for node-3).



a) node-1 **b) node-2**

c) node-3

Figure 11. System load of each grid nodes under heterogeneous services scenario.



a) node-1 **b) node-2**

c) node-3

Figure 12. Job service time for each grid node under heterogeneous services scenario.

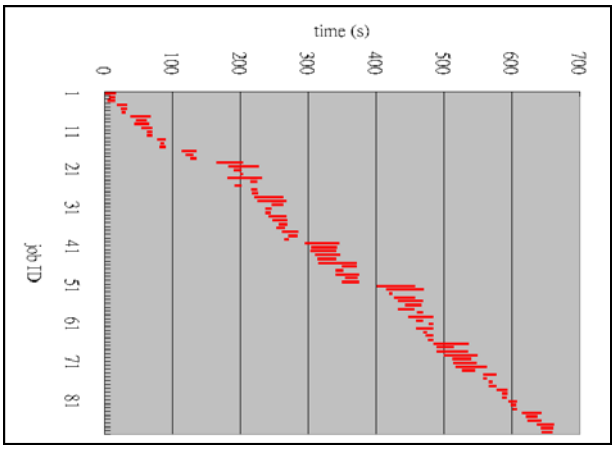
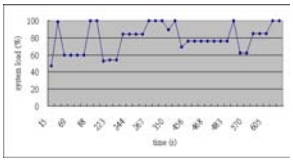
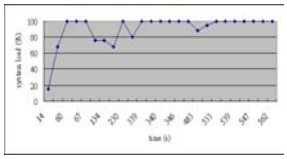


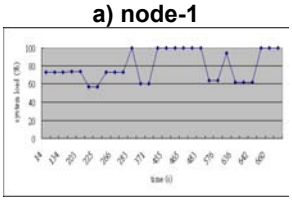
Figure 13. Job schedules under composite services scenario (all the jobs are mixed).



a) node-1

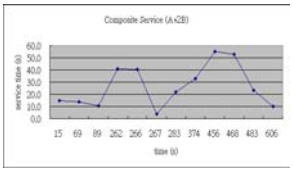


b) node-2

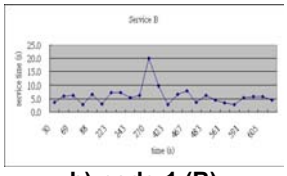


c) node-3

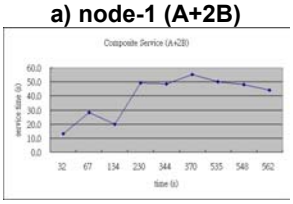
Figure 14. System load of each grid nodes under composite services scenario.



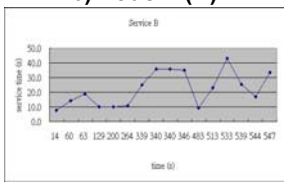
a) node-1 (A+2B)



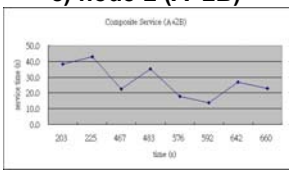
b) node-1 (B)



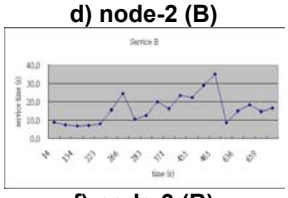
c) node-2 (A+2B)



d) node-2 (B)



e) node-3 (A+2B)



f) node-3 (B)

Figure 15. Job service time for each grid node under composite services scenario.

6. Discussion and Future Works

6.1 Accuracy of The Provided Load Estimation

The current implementation of the GT3 can only provide up-to-minute state information, where we encountered some difficulties in more fine-grained load balancing. The effect will be especially important if the execution time per job is short and the quantity of them is huge. It seems that a Grid service for supporting on-demand real-time system load reporting could be needed in the Grid middleware.

6.2 Dynamic Plan Optimization

The next obvious step of this work is to integrate the bidding mechanism one step upward to the planning step. By assuming that each Grid service interface keeps a table of scores S to indicate its desirability to use some other services, where the scores can be some statistics computed during the bidding for services selection (Section 4). Then, the setup will be similar to that of the PageRank algorithm [15] used by Google search engine for indicating Web page importance.

For example (see Figure 4), let R denote the reward for a selected plan (can be a constant equal to, say, 1), N denote the number of the outputs of the specified task, n denotes the current updating service interface, m denotes the service interfaces that use the output of current service interface n, and α denotes the updating rate (can be a constant equal to some value less than 1).

For service interfaces with their outputs form the outputs of the specified task (i.e., the ultimate goal),

$$S_n^{t+1} = (1-\alpha)\cdot S_n^t + \alpha \cdot \frac{R}{N}$$

Then, for the subsequent planning steps,

$$S_n^{t+1} = (1-\alpha)\cdot S_n^t + \alpha \cdot \sum_m S_m^t$$

Such a scoring scheme implies implicitly that frequently selected (good track records) service interfaces will be updated more frequently. Also, those interfaces often appear near to the final output of the selected plans (bringing you faster to the goal) will have higher scores. Also, those interfaces provide more outputs (more resourceful) will have a higher score. We are currently studying the effectiveness of such a scoring scheme.

6.3 Plan Base

Performing service composition from scratch can be a time-consuming process for time-critical applications. One can use a plan base for storing plans that have

been executed. A similar idea has been echoed in [13]. The archived plans (as some options of pre-composed services) can then be used for the construction of new plans. The reuse of plans should be can increase the efficiency of plan construction. For better use of the storage resource, there can also be some related policies for deleting plans that appear obsolete.

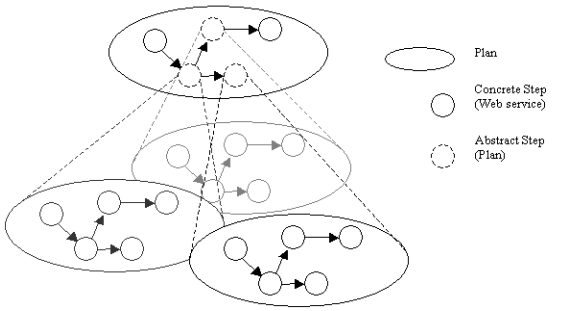


Figure 16. Plan Generation.

7. Conclusion

This paper focuses on service selection, which is usually disregarded by previous works in Web service composition. While Web services embraced in Grid platforms is getting popular, we demonstrated that service selection could make significant performance and resource utilization differences during service composition. In particular, the service bidding mechanism proposed here ensures the performance of the service to be performed and also the fairness to the service providers/bidders. Although the experimental results were encouraging, we believe that further investigation on selecting services for large scale service composition will encourage more Web service usages, especially for Grid environments where resource utilization and service performance are concerned.

Acknowledgement

This work is supported by Centre for E-Transformation Research, Hong Kong Baptist University under the RGC Group Research Grant (HKBU 2/03/C).

References

1. WSDL, http://www.w3.org/TR/wsdl
2. UDDI, http://www.uddi.org
3. SOAP, http://www.w3.org/TR/SOAP/
4. BPEL4WS, http://www-106.ibm.com/developerworks/library/ws-bpel/
5. I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," International Journal of High Performance Computing Applications, Vol. 15, pp. 200-222, 2001
6. I. Foster, C. Kesselman, J.M. Nick and S. Tuecke, "Grid Services for Distributed System Integration," IEEE Computer, June, 2002
7. IEEE Internet Computing, Special issue: Middleware for Web services, 2003.
8. J. Yang and M. Papazoglou, "Web components: A substrate for web service reuse and composition," Advanced Information Systems Engineering, Proceedings of the 14th International Conference, CAiSE 2002 Toronto, Canada, May 27-31, 2002.
9. S. McIlraith and T. Son, "Adapting golog for composition of semantic Web services", Proceedings of the 8th International Conference on Principles of Knowledge Representation and Reasoning, 2002
10. Evren Sirin, James Hendler, and Bijan Parsia, "Semi-automatic composition of Web services using semantic descriptions," Proceeding of Web Services: Modeling, Architecture and Infrastructure workshop in ICEIS, April, 2003
11. L. Chen, N.R. Shadbolt, C. Goble, F. Tao, S.J. Cox, C. Puleston, P.R. Smart, "Towards a Knowledge-based Approach to Semantic Service Composition," 2nd International Semantic Web Conference (ISWC2003), 20-23 October 2003, Florida, USA, Lecture Notes in Computer Science, LNCS 2870, pp 319-334
12. B. Benatallah, Q. Sheng, and M. Dumas, The Self-Serv environment for Web services composition, in *IEEE Internet Computing*, pages 40--48, 7(1), 2003.
13. J. Blythe, E. Deelman, Y. Gil, C. Kesselman, A. Agarwal, G. Mehta, K. Vahi, "The Role of Planning in Grid Computing," Proceedings of the 13th International Conference on Automated Planning and Scheduling (ICAPS), June 9-13, 2003, Trento, Italy
14. N. Sample, Pedram Keyani, Gio Wiederhold, "Scheduling Under Uncertainty: Planning for the Ubiquitous Grid," Proceedings of the Fifth International Conference on Coordination Models and Languages (Coord2002)
15. R. Motwani, S. Brin, L. Page, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," Stanford Digital Libraries Working Paper, 1998.