Int J Software Informatics, Vol.3, No.4, December 2009, pp. 483–499 E-mail: ijsi@iscas.ac.cn
International Journal of Software and Informatics ISSN 1673-7288 http://www.ijsi.org

Adaptive Resource Allocation for Service-Based Systems*

Stephen S. Yau and Ho G. An

(School of Computing, Informatics and Decision Systems Engineering,
Arizona State University Tempe, AZ, USA)

Abstract Due to its major advantages, service-oriented architecture (SOA) has been adopted in various distributed systems, such as web services, grid computing systems, utility computing systems and cloud computing systems. These systems are referred as service-based systems (SBS). In order to effectively use these systems in various applications, one major challenge which must be addressed is to manage the quality of services (QoS) to satisfy users' requirements. In SBS, multiple services are often hosted by the same server and compete for the limited system resources of the server, such as CPU-time, memory and network bandwidth. In addition, service compositions, resource status of servers, workflow priorities and QoS requirements are usually dynamically changing in runtime. Hence, it is necessary to have effective techniques to allocate the system resources to each service provided by a server in order to satisfy the QoS requirements of multiple workflows in SBS. In this paper, a resource allocation approach is presented to adaptively allocating the system resources of servers to their services in runtime in order to satisfy one of the most important QoS requirements, the throughput, of multiple workflows in SBS.

Key words: adaptive resource allocation; service-based systems; multiple workflows; throughput; user requirements

Yau SS, An HG. Adaptive resource allocation for service-based systems. *Int J Software Informatics*, **2009, 3(4): 483–499. http://www.ijsi.org/1673-7288/3/483.htm**

1 Introduction

The rapid growth of the Internet has enabled many applications involving many users and service providers distributed in various geographical locations. In order to facilitate the development of such applications, service-oriented architecture (SOA) has been adopted in various distributed systems, such as web service, grid computing systems, utility computing systems and cloud computing systems. Such systems are referred as service-based systems (SBS), and leads to the vision of "Internet as a supercomputer." This vision incorporates the concepts of "software as a service", "platform as a service", "infrastructure as a service" and "resources as a service," and requires software systems running on the open and dynamic Internet with the capabilities of context-awareness, self-management and autonomous adaptation for

* This work is sponsored by National Science Foundation under Grant No. CCF-0725340.
Corresponding author: Stephen S. Yau, Email: yau@asu.edu
Manuscript received 2009-08-13; revised 2009-11-08; accepted 2009-11-30; published online 2009-12-??.

changes in runtime environment. Such systems residing on the Internet are referred in general as Internetware systems.

A major challenge for the Internetware systems to satisfy various application requirements is to manage the quality of service (QoS) in runtime due to the dynamic, loosely coupled, and compositional nature of SOA. Many studies have identified important QoS features of SOA systems[1,2], such as throughput, timeliness and security, which are directly affected by the limitation of system resources. It is noted that the system resource considered here is for large-scale aggregation of distributed computing resources working together over the Internet as a tremendous virtual computer[3−5]. For example, computing grids and clouds are vast resource pools for on-demand requests over the Internet. In order to manage the QoS for such systems, an effective resource allocation technique for dynamically scalable distributed system resources is needed. In order to achieve this goal, the SBS needs the capabilities of monitoring the changing system status, analyzing and controlling system QoS features, and adapting its service configuration to satisfy the QoS requirements of multiple workflows simultaneously. Since multiple services are often hosted by the same server in SBS, and the services in the same server compete for the limited available resources of the server, such as CPU-time, memory and network bandwidth, different resource allocations will result in different QoS in runtime. In addition, the service compositions occur dynamically in runtime and with the resource status of servers dynamically changing. Thus, allocating the resources of each server to its services for successfully satisfying the QoS requirements of dynamic multiple workflows and resource status is needed to satisfy the overall QoS requirements of the workflows in SBS. In practice, a service may create multiple service instances working for different workflows[33]. Although the functionalities of the multiple service instances are exactly the same, the amount of resource consumed by the service instances may vary depending on the workflows and services they deployed in their servers. Hence, in our paper we will consider different service instances as different services.

In this paper, we will discuss the challenges of adaptive resource allocation in SBS, and the current state of the art in handling dynamic resource allocation for various computing and network systems which are useful for dynamic resource allocation in SBS. We will then present an approach to adaptively allocating the system resources of servers to their services in runtime to satisfy one of the most important QoS requirements, the throughput, of multiple workflows in SBS.

2 Challenges for Adaptive Resource Allocation for SBS

In order to develop SBS with the capability of adaptive resource allocation to satisfy the QoS requirements of multiple workflows, the following challenges need to be addressed:

C1) Identify and collect relevant contextual data for situation awareness: An SBS needs be aware of changing situations in dynamic runtime environments so that the SBS can adapt itself to continue performing satisfactorily with QoS requirements. The relevant data may include dynamic workflow compositions, service-request rate for each workflow, QoS requirements, priorities of workflows, available system resource and various relevant environmental attributes.

C2) Context analysis and QoS estimation: An SBS must be capable of analyzing

the relationship between resource allocation and various QoS of workflows efficiently. The monitored contextual data is automatically analyzed to generate good estimates of the QoS of workflows in runtime

C3) Optimal resource allocation: An SBS must be able to efficiently generate an optimal resource allocation in runtime to satisfy the QoS requirements of multiple workflows with dynamic situations. An SBS should be capable to adaptively compromise certain QoS requirements with users' preferences if not all QoS expectations can be satisfied.

C4) Implementation of resource adaptation: An SBS must be capable of adapting changes of resource allocation to their services in runtime due to C3)

C5) Efficiency and scalability: All the processes of collecting and analyzing contextual data to find an optimal resource allocation and adapting resource allocation must be efficient and scalable

3 Current State of Art of Resource Allocation for SBS

Development of QoS and resource management middleware has been studied in various areas, such as real-time systems, distributed object-oriented systems, web services, grid computing and cloud computing. A workflow-based computational resource broker[3,4] was presented for grid computing environment. The main function of the resource broker is to monitor the available resources and match workflow requirements to available resources over multiple administrative domains. The resource broker provides a uniform interface for accessing available system resources of computing grid via users credentials. An open-source toolkit Globus has emerged as a standard middleware for resource management in grid computing environment[5]. Globus provides Web Service Grid Resource Allocation and Management (WS GRAM) Protocol in which a set of web services are designed to support APIs for requesting and using grid system resources. Globus also provides Monitoring and Discovery Service (MDS) that supports a common interface for collecting contextual information on system resources, such as available processors, CPU load, network bandwidth, file system information, storage devices, and memory in computing grids.

A market-based autonomic resource management approach in cloud computing environment was developed[6], in which Service-Level Agreement Resource Allocator provides the interfaces between the cloud service providers and external users/brokers for 1) monitoring users' service requests and QoS requirements, 2) monitoring the availability of system resources, 3) examining service requests to determine whether to accept the requests according to resource availability and processing workload, 4) allocating system resources to virtual machines in cloud, 5) pricing the usage of resources and prioritizing the resource allocation, and 6) keeping track of the execution progress of the service requests and maintaining the actual usage of resources. This approach supports negotiation of QoS between users and providers to establish service-level agreements (SLA) and allocation of system resources to meet the SLAs.

Multi-layered resource management (MLRM) architecture using standard-based middleware technology[7] was developed for enterprise distributed real-time embedded (DRE) systems. This architecture supports dynamic resource management to optimize and reconfigure system resources at runtime in response to changing mission needs and resource status. With the dynamic resource allocation, a DRE system can

provide QoS for critical operations under the overloaded and resource constrained environments. In Ref.[8], a constraint-programming-based approach was presented to solve resource allocation problem in real-time systems. The problem of assigning a set of preemptive real-time tasks in a distributed system is formulated as a constraint satisfaction problem (CSP) with allocation, resource and timing constraints. First, the CSP is solved using constraint programming techniques to satisfy the allocation and resource constraints. Then, the solution is validated for timing constraints through Logic-based Benders decomposition[9].

In Ref.[10], a decentralized local greedy mechanism for dynamic resource allocation in web service applications was presented. In this mechanism, software agents are generated to buy and sell network services and resources to and from each other. In Ref.[11], a market based resource allocation for web services in a commercial environment was presented. In this approach, service providers employ a cluster of servers to host web services, and service consumers pay for service usage with QoS requirements, such as waiting time or response time. A framework was developed for evaluating the effect of particular resource allocation in terms of performance and average revenue earned per unit time. A heuristic algorithm was also developed for making resource allocation decisions in order to maximize revenue.

The QoS of networks has been investigated for providing prioritized services by efficient resource allocation techniques through labeling, scheduling and routing mechanisms. In Ref.[12], an optimal resource allocation and pricing scheme for next generation multiclass networks was presented. In this scheme, an optimization problem is formulated based on a nonlinear pricing model, whose solution ensures satisfaction of network delay constraints and efficient resource allocation in dynamic multiservice networks. In Ref.[13], an energy-efficient radio resource allocation approach based on game theory for wireless networks was presented. The study shows that the game-theoretic approach for resource allocation is useful for energy-constrained wireless networks. In Ref.[14], a resource allocation scheme for the wireless multimedia applications was presented. In this scheme, an optimization problem is formulated with fairness constraints for maximizing system capacity and resource utilization. The solution of this problem yields the optimal allocation of sub-channel, path and power. A heuristic algorithm to solve the optimization problem was also presented to ensure that the adaptive resource allocation is performed efficiently in runtime. In Ref.[15], a resource allocation approach for embedded multimedia systems using heterogeneous multiprocessors was presented, in which optimal resources are allocated to each application to meet its throughput requirement. Synchronous Dataflow Graphs (SDFG) are used to model multimedia applications with time and resource constraints and to find the optimal resource allocation.

QoS estimation according to the system activities and resource status has been studied in many ways. A QoS model of a router with feedback control that monitors the state of resource usage and adaptively adjusts parameters of traffic admission control to estimate QoS and resource utilization was presented in Ref.[16]. Batch Scheduled Admission Control (BSAC) method to predict service delay for high priority jobs in Internet-type networks was presented in Ref.[17]. A regression-based model for dynamically provisioning resource demand to deliver given QoS expectation was presented in Ref.[18]. An adaptive model for the tradeoff between service performance

and security in service-based environments was presented in Ref.[19]. This model can be used to adjust security configurations to provide sufficient protection and satisfy service performance requirements with limited system resources.

For adaptive resource allocation for QoS management in SBS, application level differentiated services[20,21] were introduced to control QoS for different classes of service consumers. When the system resources are limited, fewer resources are allocated for normal consumers, and most resources are reserved for satisfying premium consumers' expected QoS. Feedback controlled web services[22,23] were developed to adjust QoS to meet the most important performance when resources are limited and consumers' required performance cannot be fully satisfied. In Ref.[24], an integrated QoS management approach in SBS in order to satisfy users' QoS requirements by providing differentiated system resources and priority of workflows was presented. In Refs.[25, 26], a general methodology for developing SBS with QoS monitoring and adaptation was introduced. This methodology supports monitoring of the changing system status, analysis and control of tradeoffs among multiple QoS features, and adapting its service configuration to satisfy multiple QoS requirements simultaneously.

Existing resource allocation approaches cannot support dynamically changing runtime environments in SBS, such as workflow composition, QoS requirements, workflow priorities and resource status. Hence, we need a new approach to dynamic resource allocation for SBS and address the challenges discussed in Section 2.

4 Overview of Our Approach to Adaptive Resource Allocation for SBS

In this section, we will present a resource allocation approach to adaptively allocating the system resources of servers to their services in runtime in order to satisfy the throughput requirements of the multiple workflows in SBS. Our resource allocation approach addresses the challenges in Section 2.

In SOA, a service that cannot be decomposed to smaller services and serves only one type of service-request is considered as an *atomic service*. *All* other services are considered *composite services*. In order to have adaptive resource allocation capabilities in SBS, we need to analyze the relationship between resource allocation of each server to its atomic services and the throughputs in both atomic and composite services provided by the server. In our approach, we first develop the Resource-Allocation-Throughput (RAT) model for an atomic service, and extend the model for the entire SBS to analyze the relationship between resource allocation and throughputs of the multiple workflows in SBS. Based on the RAT model, we will present an algorithm to automatically formulate a linear programming optimization problem[27] to find the optimal resource allocation to serve the users' service requests. We define the optimal resource allocation as follows:

1) Optimal resource allocation will serve all the service requests of workflows in SBS until system resources are exhausted.

2) If system resources are exhausted and all the service requests cannot be served, the optimal resource allocation will selectively serve the service requests to maximize the throughput of SBS subject to the minimum throughput requirements and priorities of workflows.

The major distinction between our resource allocation approach and other re-

source allocation approaches discussed in Section 3 is that our approach can maximize utilization of limited system resources to reach maximum throughput according to dynamic workflow composition, throughput requirements, workflow priorities and resource status, rather than just serving users' service-requests in the first-in-first-serve manner by matching available system resources upon the arrival of the users' requests and requirements.

The conceptual view of our resource allocation approach is depicted in Fig.1. Each rectangle represents a discrete system component. Functional details and relationships among the components are described as follows:

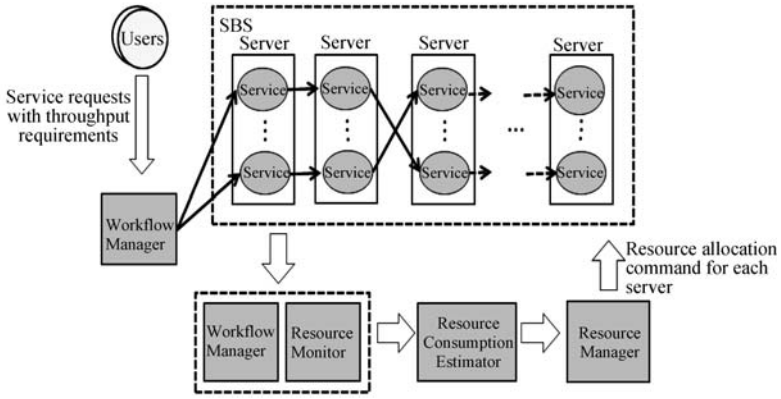

Figure 1. Our conceptual view of dynamic adaptive resource allocation for SBS

- **Workflow Manager** dynamically creates workflows based on the users' requests and available services. Automatic service discovery and composition can be done with semantic service description[28].
- **Workflow Monitor** is responsible for gathering information about users' service-request rates of each workflow from the Workflow Manager. The users' service-request rates can be dynamically changed in runtime, and the monitoring of service-request rates is done periodically.
- **Resource Consumption Estimator** is responsible for estimating the amount of resource to be consumed for executing service-requests of each workflow. The estimation is done based on the RAT model,which will be discussed in Section 5.
- **Resource Monitor** is responsible for collecting information about available resources of each server in SBS. The resource monitoring module for a large SBS can be built at the middleware layer, such as Ganglia[29] and Resource Broker[3,4].
- **Resource Manager** is responsible for finding an optimal resource allocation that maximizes the throughput of SBS with the constraints on the throughput requirements, available system resources, workflow orders, priorities, and service-request rates. In order to find the optimal resource allocation, we use linear programming (LP) for optimization of a linear objective function, subject to linear inequality constraints[27]. Detailed discussion on formulating linear

programming optimization problem will be given in Section 6. Resource Manager is also responsible for adaptively allocating system resources of servers. The resource allocation can be dynamically done in runtime at the middleware layer, such as QoS Resource Manager[26] and Globus[5].

In our approach, we address the challenges discussed in Section 2 as follows: Situation awareness is supported by Workflow Manager, Workflow Monitor and Resource Monitor. Context analysis and QoS estimation are supported by Resource Consumption Estimator. Optimal resource allocation and resource adaptation are supported by Resource Manager. Efficiency and scalability of our algorithm to find the optimal resource allocation will be discussed in Section 7.

5 RAT Models for SBS

5.1 For atomic services

In this section, we will discuss our RAT model for the relationship between system resource allocation of servers and the throughput of SBS. Before we develop the RAT model of the entire SBS, we first need to model the relationship between resource allocation of each server to its atomic services and the throughput of the atomic services.

Our RAT model for an atomic service is shown in Fig.2. Limited system resources of a server are allocated to an atomic service. As service-requests arrive at the Service Request Queue, the atomic service creates multiple threads, which utilize the system resource to process the service-requests and send out the service responses. In this model, we consider the following five factors to estimate the amount of resource to be consumed by the atomic service for executing service-requests:

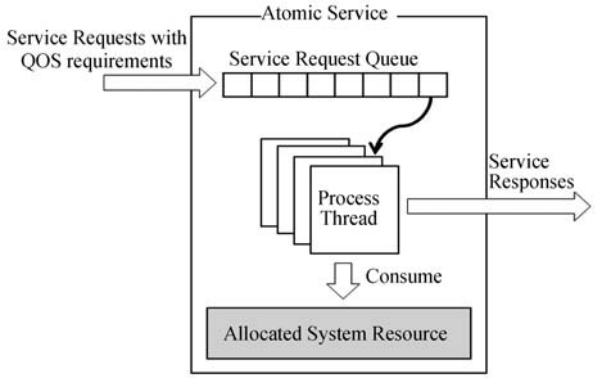

Figure 2. The RAT model for an atomic service

- **Service-request rate R of an atomic service:** R is the average number of service-requests per second arriving at an atomic service, and represents the workload of the atomic service. An atomic service can limit the number of process threads and adaptively drop some service requests or put them into a waiting-queue according to the resource status of the atomic service in order to maintain a certain level of throughput.

- **Critical resource:** Processing service-request will require various types of system resources of a server, such as CPU time, memory and network bandwidth. As R increases and more threads are created, one of the system resources of the server will become a bottleneck, and such system resource of the server is called the critical resource of the server.
- **Percentage of Allocated critical resource A:** A is the percentage of allocated critical resource of a server to an atomic service provided by the server over the total available critical resource of the server. We only need to consider the allocation of the critical resource of a server because the critical resource of the server becomes a bottleneck first when other resources are sufficient.
- **Throughput P of an atomic service:** P is defined as the average number of service responses per second of the atomic service. P is the sum of the service-responses of each thread per second in an atomic service, and determined by the R and A of the atomic service.
- **Throughput requirement of a workflow:** This is the minimum number of service responses per second required by the users for a workflow.

From this model, we notice that

1) P = R if A is not exhausted. In this case, P will be determined by R.

2) Increasing R will increase P until allocated resource is exhausted.

3) When A is exhausted, P will not increase even if R increases. Some of service-requests will be dropped or put into a waiting-queue. In this case, P will be determined by A.

Based on the above observations, we can estimate the throughput P of an atomic service as follows:

$$\begin{aligned} \mathrm{P} &= \mathrm{R} \text{ when A is not exhausted} \\ &= \alpha\ \mathrm{A} \text{ when A is exhausted} \end{aligned}$$

where α is a proportional constant, which is different for different atomic services. Given the allocated critical resource A, α A is the maximum throughput of the atomic service. We define the service-cost of an atomic service as follows:

Definition: The *service cost* of an atomic service is the percentage of critical resource required to serve a service-request over the total available critical resource.

It is noted that since $\mathrm{P} = \alpha\mathrm{A}$ when A is exhausted, the service cost of an atomic service is $(\mathrm{A}/\alpha\mathrm{A})=1/\alpha$. The critical resource type and the service cost of an atomic service can be estimated from the performance models for QoS related cause-effect dynamics in SBS, called *Activity-State-Event-QoS (ASEQ) models*[25,26]. We are currently developing a general methodology for constructing the ASEQ models for SBS by conducting well-designed experiments in SBS and collecting data related to various users' service-requests and system resource states. In this paper, we only need to consider the relationship between throughput and resource allocation since we only consider the throughput in this paper. We can estimate the service cost of an atomic service as follows: S1) Allocate an arbitrary amount of critical resource A to a target atomic service. S2) Increase the R and measure the P of the atomic service until the critical resource of the server is exhausted. S3) Repeat S1) and S2) to obtain P with various A and R. S4) Using the statistical linear regression analysis on the P obtained in S3), estimate the service cost of the target atomic service.

5.2 For composite services

SBS consists of multiple servers and workflows. Each server provides multiple atomic services, and each atomic service is represented by its service cost. Each workflow is composed of atomic services in different servers. Each server has its critical resource and ability to adaptively allocate the critical resource to its atomic services in runtime.

<SBS>::= "(" {<Workflow>}$^+$, {<Server>}$^+$ ")"
<Server>::= "("<Server ID>, <Critical resource>, <% of available
critical resource>, {<Atomic service>}$^+$ ")"
<Atomic service>::= "(" <Service ID>, <Service cost><Server ID>")"
<Workflow>::= "(" {<Service Composition>}$^+$, <Service-request rate>,
<Throughput-requirement><Priority>")"
<Starting>::= <Atomic service>
<Ending>::= <Atomic service>
<Service Composition>::= <Sequence>|<Parallel Sprit>|<Merge>|<Pick>
<Sequence>::= "(" <Starting>, <Ending>")"
<Parallel Sprit>::= "(" <Starting>, {<Ending>}$^+$ ")"
<Merge>::= "(" {<Starting>}$^+$, <Ending>")"
<Pick>::= "(" <Starting>, <Condition>, {<Ending>}+ ")"

Figure 3. BNF Representation of SBS

In practice, it may be impossible to satisfy all the users' throughput requirements simultaneously because of limited system resources of servers. In this case, we need to relax the throughput requirements for some workflows. This can be done by setting priorities for workflows and allocate more resources to high-priority workflows when system resources are exhausted. How to allocate resources according to the priority will be presented in Section 6.

Based on the relations among servers, atomic services and workflows, we describe a SBS in BNF as shown in Fig.3. From the RAT model of atomic service and the BNF of SBS, we can calculate the throughput of each workflow with given service-cost, percentage of allocated critical resource and service-request rate for each workflow as follows:

Calculating throughput of a workflow in SBS

1. S = {s | s is an atomic service in a workflow}
2. For each s $\in$ S
3. α = service-cost of s
4. R = service-request rate of s
5. A = Percentage of allocated critical resource to s
6. Let P(s) be the throughput of s
7. if (R $\leqslant$ A / α) then P(s) = R
8. else P(s) = A / α
9. Throughput of the workflow = MINIMUM s $\in$ S (P(s))

6 Resource Allocation Using RAT Models and Linear Programming

Based on the BNF of SBS and the RAT model of atomic service, we formulate a constraint-optimization problem using linear programming. The solution of this optimization problem determines the amount of critical resource required for each atomic service. The following pseudo code shows our algorithm to automatically formulate a linear programming optimization problem with given BNF of a SBS.

Formulating a linear programming problem to allocate critical resources of all servers in a SBS

```
1. W = {w | w is a workflow in SBS}
2. Sv = {sv | sv is a server in SBS}
3. For each w ∈ W
4.     Let throughput of w be TH(w)
5.     Pr(w) = Priority of w
6.     Add objective function of LP
( "Maximize Σ_{w∈W}(TH(w)× Pr(w))")
7.     TR(w) = throughput-requirement of w
8.     SR(w) = service-request rate of w
9.     Add constraint ("TH(w) ⩽ Service-request rate of w")
10.    If ( TR(w) ⩽ SR(w) )
11.    Add constraint ("TH(w) ⩾ TR(w)")
12. For each sv ∈ Sv
13.     For each w ∈ W
14.         If w uses atomic services provided by sv
15.             C = 0
16.             For each atomic service s in sv used by w
17.                 α = service-cost of s
18.                 C = C + α×TH(w)
19.         AR(sv) = % of available critical resource of sv
20.         Add constraint ("C ⩽ AR(sv)")
```

---

7 Adaptive Allocation of Critical Resources of Servers for SBS

Using the Simplex algorithm[27], we can solve the linear programming problem presented in Section 6 and find the optimal throughput of each workflow in SBS. Given n workflows and m servers, this problem will have m variables and 2m+n constraints. Thus, the complexity of solving this problem is O(m3 + nm2) in the worst case. On the average, these linear programming problems are solved in polynomial time[30]. Hence our approach addresses the challenge of efficiency and scalability discussed in Section 2. Once we find the throughput of each workflow, allocate the critical resource to each atomic service so that each workflow can produce its optimal throughput. The following pseudo code shows how to allocate the critical resources of servers to their atomic services.

An aloocation algorithm for allocating critical resources of servers to their atomic services

```
1. W = {w | w is a workflow in SBS}
2. Sv = {sv | sv is a server in SBS}
3. S = { s | s is an atomic service in Sv}
4. Solve the LP problem using the Simplex algorithm and find optimal throughputs of workflows
5. OT(w) = optimal throughput of workflow w ∈ W
6. For each s ∈ S
7.     α = service-cost of s
8.     Let A(s) be the amount of critical resource to be
9.     allocated to s
10.    A(s) = 0
11.    For each workflow w that uses s
12.        A(s) = A(s) + OT(w) ×α
```

8 Demonstration

In this section, we will demonstrate our approach using a SBS shown in Fig. 4 with four servers, nine atomic services and three workflows. The four servers were implemented as virtualized platforms using Hyper-V Manager 6.0 [31]. The configurations of the servers are as show in Table 1.

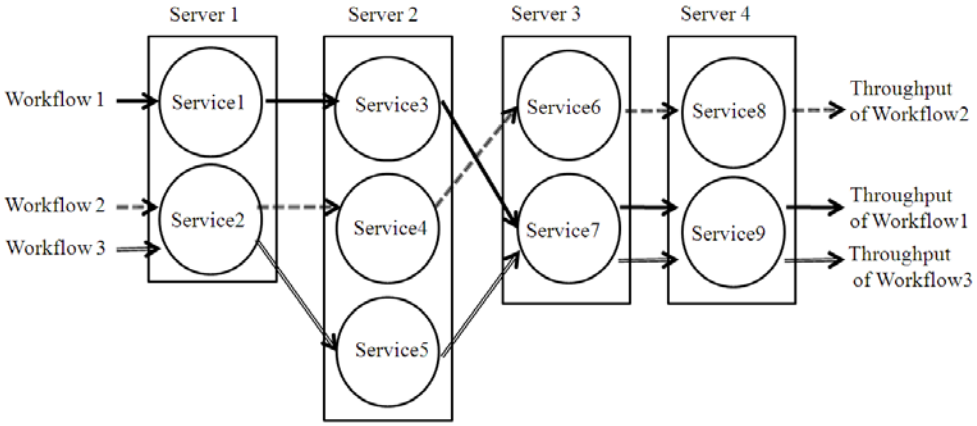

Figure 4. A SBS demonstration system for our approach

Table 1 The configurations of the four servers in our demonstration system

Server	Operating System	CPU	Memory
Server1	Windows Server 2003	0.7 GHZ	1024MB
Server2	Windows Server 2003	2.94 GHZ	256MB
Server3	Windows Server 2003	0.7 GHZ	1024MB
Server4	Windows Server 2003	0.7 GHZ	1024MB

The nine atomic services were implemented as web services using C# Microsoft .NET framework. Using the RAT model of atomic service discussed in Section 5, the

nine atomic services create threads to process runtime service requests from multiple users simultaneously. The atomic services are listed as follows:

- Service 1 (Video Encoding Service): It takes 35 KB binary stream as input and converts it to AVI video format.
- Service2 (Data Compression Service): It takes 35 KB byte array as input and compresses the data using Data Compressing library provided by C#.
- Service3 (DES Encryption Service): It takes any size of byte array as input and encrypts the data with key size 64 bits using DES Encryption library provided by C#.
- Service4 (AES128 Encryption Service): It takes any size of byte array as input and encrypts the data with key size 128 bits using AES Encryption library provided by C#.
- Service5 (AES256 Encryption Service): It takes any size of byte array as input and encrypts the data with key size 256 bits using AES Encryption library provided by C#.
- Service6 (DSA Digital Signature Service): It takes any size of byte array as input and produces digital signature with key size 1024 bits using DSA Digital Signature library provided by C#.
- Service7 (MD5 Hashing Service): It takes any size of byte array as input and generates hash of the data using MD5 Hashing library provided by C#.
- Service8 (DSA Digital Signature Service): It takes any size of byte array as input and produces digital signature with key size 2048 bits using DSA Digital Signature library provided by C#
- Service9 (SHA1 Hashing Service): It takes any size of byte array as input and generates hash of the data using SHA1 Hashing library provided by C#.

Besides these services, we also developed a throughput monitoring service with Windows Performance Objects [32] which collects the throughput of each workflow and the resource consumption of each server in runtime.

To estimate the service cost of each atomic services, we ran one set of experiments. In each set of the experiments, we invoked its atomic service and increased service request rate until the system resource of the server was exhausted. The throughput of each atomic service using the throughput monitoring service is shown in Fig. 5. The shapes of these curves are similar because the service response rates of the atomic services did not increase after the system resource of the server was exhausted even if the service request rates increased.

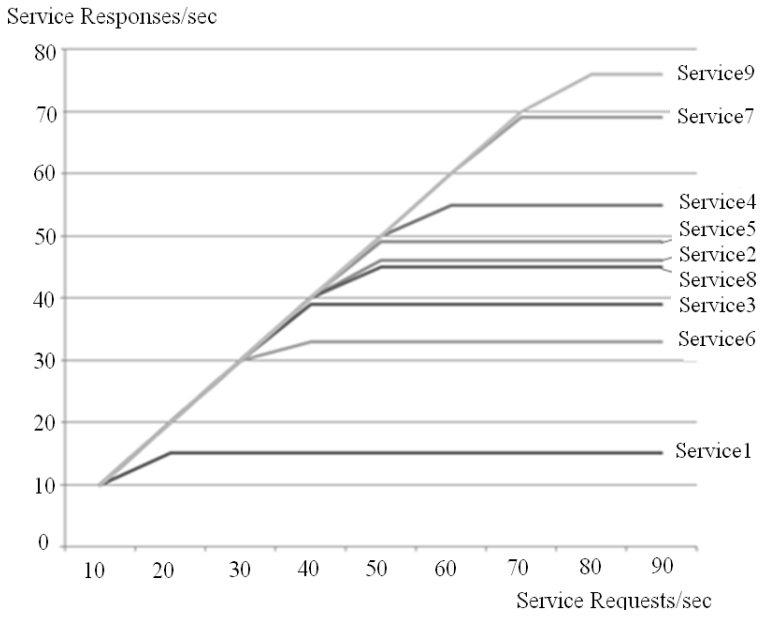

Figure 5. The throughput threshold of each of the nine atomic services in our demonstration system

The critical resource of the server of each atomic service is observed using the throughput monitoring service and shown in Table 2. Based on the experimental results and the RAT model, we found the service cost of each atomic service using the process described in Section 5.1, which is shown in Table 3.

Table 2 The critical resource of each server in our demonstration system

Server	Server 1	Server 2	Server 3	Server 4
Critical Resource	CPU	Memory	CPU	CPU

Table 3 Service cost of each atomic service

Service	Servc 1	Servc 2	Servc 3	Servc 4	Servc 5	Servc 6	Servc 7	Servc 8	Servc 9
Service Cost	6.7%	2.1%	2.6%	2%	2.1%	2.94%	1.4%	2.17%	1.3%

We have the three workflows composed by the nine atomic services as shown in Fig. 4. We ran two experiments on these workflows. In the first experiment, the workflows were executed without using our resource allocation approach. In the second experiment, we ran the workflows with our resource allocation approach. The service request rate of each workflow was set to 20 service requests/sec. The throughput requirements of workflow 1, workflow 2 and workflow 3 were set to 3, 15 and 10 service responses/sec respectively. Based on the critical resource of each server from the throughput monitoring service and the estimated service cost of each atomic service, the BNF representation of the SBS is given below:

<service1> = (<“service”> <6.7%> <“server”>)
<service2> = (<“service”> <2.1%> <“server1”>)
<service3> = (<“service3”> <2.1%> <“server2”>)
<service4> = (<“service4”> <2%> <“server2”>)
<service5> = (<“service5”> <2.1%> <“server2”>)
<service6> = (<“service6”> <2.94%> <“server3”>)
<service7> = (<“service7”> <1.4%> <“server3”>)
<service8> = (<“service8”> <2.17%> <“server4”>)
<service9> = (<“service9”> <1.3%> <“server4”>)
<Server1> = (<“server1”> <CPU> <100%> <service1> <service2>)
<Server2> = (<“server2”><Memory><100%><service3><service4><service5>)
<Server3> = (<“server3”> <CPU> <100%> <service6> <service7>)
<Server4> = (<“server3”> <CPU> <100%> <service8> <service9>)
<Sequence1> = (<service1><service3>) <Sequence2>=(<service3><service7>)
<Sequence3> = (<service2><service4>) <Sequence4>=(<service4><service6>)
<Sequence5> = (<service2><service5>) <Sequence6>=(<service5><service7>)
<Sequence7>=(<service6><service8>) <Sequence8>=(<service7><service9>)
<Workflow1>=((<Sequence1><Sequence2><Sequence8>)<30/sec><3/sec><1>)
<Workflow2>=((<Sequence3><Sequence4><Sequence7>)<30/sec><15/sec><1>)
<Workflow3>=((<Sequence5><Sequence6><Seqiemce8>)<30/sec><10/sec><1>)

From the BNF representation of the SBS, we generated the linear programming problem using the algorithm described in Section 6 shown as follows:

Maximize p = TH1 + TH2 + TH3 Subject to
TH1 <= 15, TH2 <= 15, TH3 <= 15, TH1 >= 3, TH2 >= 15, TH3 >= 10
6.7TH1 + 2.1TH2 + 2.1TH3 <= 100
2.6TH1 + 2 TH2 + 2.1 TH3 <= 100
1.4TH1 + 2.94TH2 + 1.4TH3 <= 100
1.3TH1 + 2.17TH2 + 1.3TH3 <= 100

After solving the linear programming problem, the throughputs for workflow 1, workflow 2 and workflow 3 were 3, 26.6 and 12.4 responses/sec, respectively. Then, we estimated the optimal resource allocation of the critical resource of each server using the resource allocation algorithm presented in Section 7, which is given in Table 4. The throughputs of the three workflows of the two experiments are shown in Fig. 6. It is noted that there were significant differences in throughputs of the workflows between the two experiments. It is also noted that the throughput requirements of workflow 2 and workflow 3 were not satisfied in the first experiment which was run without using our resource allocation approach, but all the throughput requirements were satisfied in the second experiment using our resource allocation approach, and that the overall throughput of the SBS, the sum of the throughputs of these three workflows increased by 28%.

Table 4 Resource allocation of each atomic service

Server	Server 1		Server 2			Server 3		Server 4	
Service	Servc 1	Servc 2	Servc 3	Servc 4	Servc 5	Servc 6	Servc 7	Servc 8	Servc 9
Optimal Resource Allocation	21%	79%	15%	55%	27%	30%	70%	20%	80%

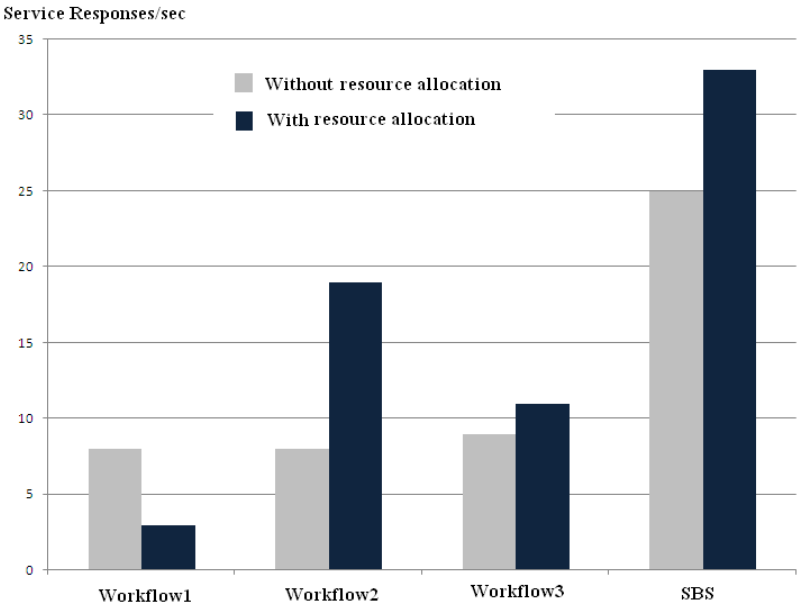

Figure 6. The throughputs of workflows in our demonstration system

9 Conclusion and Future Work

We have discussed the challenges for adaptive resource allocation to manage the QoS in SBS. We have presented a resource allocation approach to adaptively allocating the system resources of servers to their services in runtime in order to satisfy the throughput requirements of multiple workflows in SBS. To develop our approach, we have developed the RAT model for an atomic service and the BNF representation model for SBS, and an algorithm to automatically generate a linear programming optimization problem from the BNF representation model of SBS. We have also shown that our algorithm will find the optimal resource allocation in polynomial time. The demonstration results show that our dynamic resource allocation approach can substantially increase the throughput of a SBS. Future research includes the extension of our adaptive resource allocation approach to e other QoS features, such as timeliness, accuracy and security.

Acknowledgment

The authors would like to thank Dazhi Huang of Arizona State University for many valuable discussions.

References

[1] Brahnmath G, Raje RR, Olson A, Auguston M, Bryant BR, Burt CC. A Quality of Service

Catalogue for Software Components. Proc. Conf. Southeastern Software Engineering, 2002. 513–520.

[2] Ivan J, Caroline H, Stephane F. A Comprehensive Quality Model for Service-Oriented Systems. Software Quality Control, 2009, 17(1): 65–98.

[3] Yang C, Lin C, Chen S. A Workflow-based Computational Resource Broker with Information Monitoring in Grids. Proc. 5th Intl Conf. Grid and Cooperative Computing, 2006. 105–206.

[4] Venugopal S, Chu X, Buyya R. A Negotiation Mechanism for Advance Resource Reservation using the Alternate Offers Protocol. Proc. 16th Int'l Workshop on Quality of Service (IWQoS 2008), 2008.

[5] Yang CT, Shih PC, Lin CF, Chen SY. A Resource Broker with an Efficient Network Information Model on Grid Environments. The Journal of Supercomputing, 2007, 40(3): 249–267.

[6] Buyya R, Yeo CS, Venugopal S. Market Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities. Proc. 10th IEEE Int'l Conf. on High Performance Computing and Communications, 2008. 234–242.

[7] Lardieri P, Balashbramanian J, Schmidt DC. A multi-layered resource management framework for dynamic resource management in enterprise DRE systems. Journal of Systems and Software, 2007, 80(7): 984–996.

[8] Hladik PE, Cambazard H, Deplanche AM, Jussien N. Solving a real-time allocation problem with constraint programming. Journal of Systems and Software, 2008, 81(1): 132–149.

[9] Hooker JN, Ottoson G. Logic-based benders decomposition. Mathematical Programming, 2003, 96: 33–60.

[10] Stoesser J, Roessle C, Neumann D. Decentralized Online Resource Allocation for Dynamic Web Service Applications. Proc. 4th Int'l Conf. Enterprise Computing, E-Commerce, and E-Service, 2007. 425–428.

[11] Mazzucco M, Mitrani I, Palmer J, Fisher M, McKee P. Web Service Hosting and Revenue Maximization. Proc. 5th European Conf. on Web Services (ECOWS07), 2007. 92–98.

[12] Kallits M, Michailidis G, Devetsikiotis M. Pricing and Optimal Resource Allocation in Next Generation Network Services. Proc. IEEE Sarnoff Symposium, 2007. 1–5.

[13] Meshkati F, Poor HV, Schwartz SC. Energy-Efficient Resource Allocation in Wireless Networks. Signal Processing Magazine, 2007, 24(3): 58–68.

[14] Bae C, Cho D. Fairness-Aware Adaptive Resource Allocation Scheme in Multihop OFDMA Systems. IEEE Communications Letters, 2007, 11(2): 134–136.

[15] Stuijk S, Basten T, Geilen M, Corporaal H. Multiprocessor Resource Allocation for Throughput Constrained Synchronous Dataflow Graphs. Proc. Conf. 44th IEEE annual Design Automation, 2007. 777–782.

[16] Yang Z, Ye N, Lai YC. QoS model of a router with feedback control. Quality and Reliability Engineering Int'l, 2006, 22(4): 429–444.

[17] Ye N, Harish B, Li X, Farley T. Batch scheduled admission control for service dependability of computer and network resources. Information, Knowledge, Systems Management, 2006, 5(4): 211–226.

[18] Zhang Q, Cherkasova L, Smirni E. A Regression Based Analytic Model for Dynamic Resource Provisioning of Multi-Tier Applications. Proc. 4th Int'l Conf. Autonomic Computing, 2007. 27–36.

[19] Yau SS, Yin Y, An HG. An Adaptive Model for Tradeoff between Service Performance and Security in Service-based Environments. Proc. Intl Conf. Web Services (ICWS 2009), 2009. 287–294.

[20] Eggert L, Heidemann J. Application-Level Differentiated Services for Web Services. J. World-Wide Web, 1999, 2(3): 133–142.

[21] Rao G, Ramamurthy B. DiffServer: Appliction Level Differentiated Services for Webservers. Proc. IEEE Int'l Conf. on Communication, 2001, 5: 1633–1637.

[22] Lu C, Lu Y, Abdelzaher TF, Stankovic JA, Son SH. Feedback Control Architecture and Design Methodology for Service Delay Guarantees in Web Servers. IEEE Trans. on Parallel and Distributed Systems, 2006, 17(9): 1014–1027.

[23] Abdelzaher TF, Stankovic JA, Lu C, Zhang R, Lu Y. Feedback Performance Control in Software Services. IEEE Control Systems Magazine, 2003, 23(3): 74–90.

[24] Wang G, Chen A, Wang C, Fung C, Uczekaj S. Integrated quality of service (QoS) management in service-oriented enterprise architectures. Proc. 8th Int'l Conf. Enterprise Distributed Object Computing, 2004. 21–32.

[25] Yau SS, Ye N, Sarjoughian H, Huang D. Developing Service-based Software Systems with QoS Monitoring and Adaptation. Proc. 12th Int'l Workshop on Future Trends of Distributed Computing Systems, 2008. 74–80.

[26] Yau SS, Ye N, Sarjoughian H, Huang D, Roontiva A, Baydogan M, Muqsith M. Towards Development of Adaptive Service-based Software Systems. IEEE Trans. Service Computing, 2009. 247-260.

[27] Griva I, Nash SG, Sofer A. Linear and Nonlinear Optimazation (2nd ed.). Society for Industrial Mathematics, Philadelphia, PA, 2008.

[28] Kona S, Bansal A, Gupta G, Hite T. Web Service Discovery and Composition using USDL. Proc. 3rd IEEE Int'l Conf. E-Commerce Technology, 2006. 65–67.

[29] Massie ML, Chun BN, Culler DE. Ganglia Distributed Monitoring System: Design, Implementation, and Experience, Parallel Computing, 2004, 30: 817–840.

[30] Spielman D, Teng S. Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. Proc. 33th Annual ACM Symp. Theory of Computation, 2001. 296–305.

[31] http://www.microsoft.com/hyper-v-server/en/us/default.aspx

[32] http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/sag_ mpmonperf_06.mspx?mfr=true

[33] Zhang D, Xu J. Securing instance-level interactions in web services. Proc. Int'l Symposium on Autonomous Decentralized Systems, 2005. 443–450.