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 85287-8809, USA
+1 480-965-2647

{yau, ho.an}@asu.edu

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.

Categories and Subject Descriptors

K.6.2 [**Management of Computing and Information Systems**]: Installation Management – *Pricing and resource allocation*

General Terms

Algorithms, Management, Performance, Design

Keywords

Adaptive resource allocation, service-based systems, multiple workflows, throughput, and user requirements

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 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 ASBS, 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 ASBS.

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) *Situation Awareness:* An SBS must be aware of changing situations in dynamic runtime environments. An efficient and

Internetware'09, October 17-18, 2009, Beijing, China.

reliable monitoring scheme is needed for collecting the relevant contextual data of situation, such as dynamic workflow compositions, the number of service-requests for each workflow, QoS requirements, priorities of workflows, and available system resource as well as various relevant environmental attributes.

C2) *Context analysis and QoS estimation:* An SBS must be able to efficiently analyze the relationship between resource allocation and QoS of workflows. An automated analysis of the monitored context data should generate good estimates of the expected QoS of workflows.

C3) *Optimal resource allocation:* An SBS must be able to efficiently find an optimal resource allocation in runtime to satisfy the QoS requirements of multiple workflows with dynamic situations. If some of the QoS requirements of workflows cannot be satisfied, then the system must be able to adaptively sacrifice some requirements according the contextual data mentioned in C1)

C4) *Implementation of resource adaptation:* Once an optimal resource allocation is determined, SBS must be able to adaptively change the resource allocation to its services in runtime due to dynamic situation.

C5) *Efficiency and scalability:* The processes of collecting and analyzing contextual data to find an optimal resource allocation and adapt 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 research 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] is 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 provider 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 VMs 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 [8], a constraint-programming-based approach is 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 [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 [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 [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 [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 [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 [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 [16]. Batch Scheduled Admission Control (BSAC) method to predict service delay for high priority jobs in Internet-type networks was presented in [17]. A regression-based model for dynamically provisioning resource demand to deliver given QoS expectation was presented in [18]. An adaptive model for the tradeoff between service performance and security in service-based environments was presented in [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 [24], an integrated QoS management approach in SBS in order to satisfy user's QoS requirements by providing differentiated system resources and priority of workflows was presented. In [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 *atomic service*. Services including 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 and throughputs in both atomic and composite services. 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 saturated.
2. If system resources are saturated 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 resource 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 first-in-first-serve manner by matching available system resources upon the arrival of the user requests and requirements.

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

- **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. The detail discussion of the RAT model will be given 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 throughput of SBS with 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]. Detail 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.

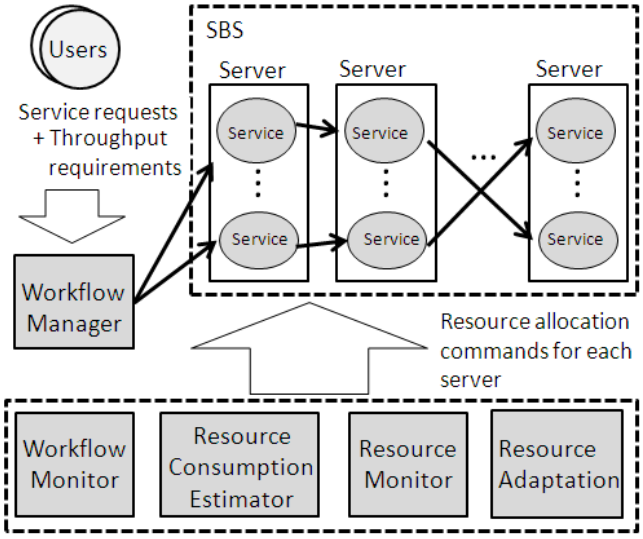

Figure 1. Our conceptual view of dynamic adaptive resource allocation for SBS

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 and throughput of an atomic service.

Our RAT model for an atomic service is shown in Figure 2. An atomic service has limited system resources allocated to the 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:

- **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 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.
- **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:

$$P = R \quad \text{when A is not saturated}$$
$$= \alpha A \quad \text{when A is saturated}$$

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

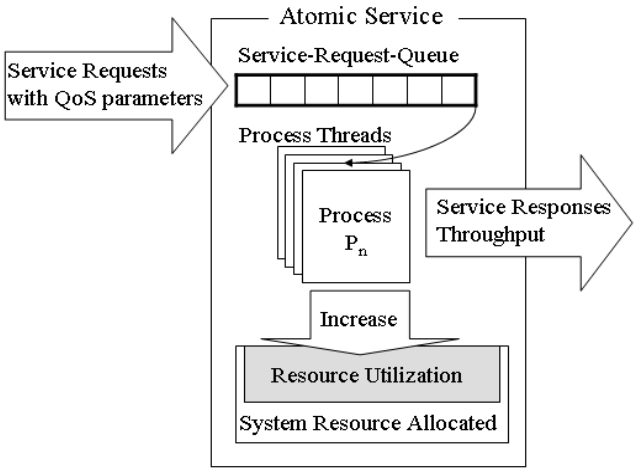

Figure 2. The RAT model for an atomic service

Definition 1: 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 $P = \alpha A$ when A is saturated, the service cost of an atomic service is $A / \alpha A = 1/\alpha$.

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. 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 throughput reaches its threshold.

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.

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 saturated. How to allocate resources according to the priority will be presented in Section 6.

```
<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

Based on the relations among servers, atomic services and workflows, we describe a SBS in BNF as shown in Figure 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 ∈ 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 ≤ 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 resources 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 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
5. Add objective function of LP
   ("Maximize $\sum_{w \in W}(TH(w) \times Pr(w))$")
6. TR(w) = throughput-requirement of w
7. SR(w) = service-request rate of w
8. Add constraint ("TH(w) ≤ Service-request rate of w")
9. If (TR(w) ≤ SR(w))
10. Add constraint ("TH(w) ≥ TR(w)")
11. For each sv ∈ Sv
12. For each w ∈ W
13. If w uses atomic services provided by sv
14. C = 0
15. For each atomic service s in sv used by w
16. α = service-cost of s
17. C = C + α×TH(w)
18. AR(sv) = % of available critical resource of sv
19. 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 $O(m^3 + nm^2)$ 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 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 allocated to s
9. A(s) = 0
10. For each workflow w that uses s
11. A(s) = A(s) + OT(w) × α

8. AN ILLUSTRATIVE EXAMPLE

In this section, we will give an example to show how our resource allocation approach works.

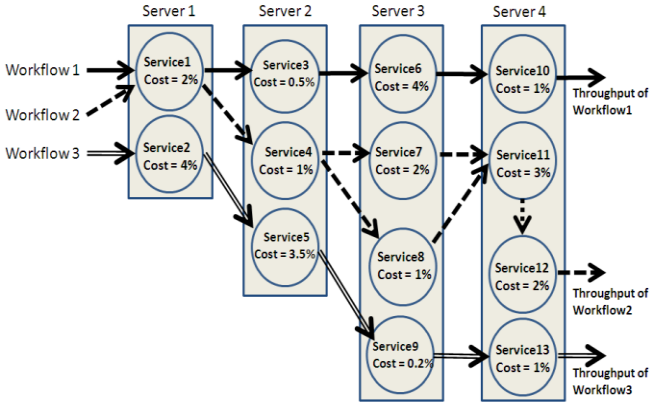

Figure 4. An example of workflows in SBS

Figure 4 shows an example of a SBS, in which there are 3 workflows composed of 13 atomic services provided by 4 servers. Rectangles represent servers and circles represent atomic services. Each atomic service has its service cost estimated from ASEQ models [25, 26] as shown in Figure 4. Let the service-request rates of workflows 1, 2 and 3 be 20, 30 and 40 requests/sec, respectively. Let the throughput requirements for workflows 1, 2 and 3 be 8 and 12 responses/sec, respectively. Let the critical resource of servers 1, 2, 3 and 4 be CPU-time, network bandwidth, CPU-time, and memory, respectively. Let the available critical resource of servers 1, 2, 3 and 4 be 100%, 90%, 80% and 90%, respectively. The priorities of workflows 1, 2 and 3 are 1, 1.5 and 1.2 respectively.

The BNF representation of the SBS is give below:

<service1> = (<"service1"> <2%> <"server1">)
…
<service13> = (<"service13"> <1%> <"server4">)
<Server1> = (<"server1"> <CPU-time> <100%>
<service1> <service2>)
<Server4> = (<"server4"> <CPU-time> <90%>
<service10> <service11> <service12> <service13>)
<Sequence1> = (<service1><service3>)
<Sequence3> = (<service6><service10>)
<Workflow1>= ((<Sequence1><Sequence2><Sequence3>)
<20 /sec> <8 /sec> <1>)
<Sequence4> = (<service1><service4>)
<Parallel Sprit1> = (<service4><service7><service8>)
<Merge1> = (<service7><service8><service11>)
<Sequence5> = (service11><service12>
<Workflow2> = ((<Sequence4><Parallel Sprit1><Merge1>
<Sequence5>) <30/sec> <12/sec> <1.5>)

From the BNF representation of the SBS, we can formulate the linear programming problem using the algorithm described in Section 6. The linear programming problem is written as follows:

Objective Function =
Maximize TH1 + 1.5×TH2 + 1.2×TH3

Subject to constrains:

$TH1 \leq 20, TH2 \leq 30, TH3 \leq 40$
$TH1 \geq 8, TH2 \geq 12, TH3 \geq 12$

$2\times TH1 + 2\times TH2 + 4\times TH3 \leq 100$
$0.5\times TH1 + TH2 + 3.5\times TH3 \leq 90$
$4\times TH1 + 2\times TH2 + TH2 + 0.2\times TH3 \leq 80$
$1\times TH1 + 3\times TH2 + 2\times TH2 + TH3 \leq 90$

After we solve the linear programming problem, the throughputs for workflows 1, 2 and 3 are 9.25, 13.4 and 13.6 responses/sec, respectively. Then, we allocate the critical resource to each atomic service so that each atomic service can provide the throughput as discussed in Section 7. For example, we allocate 45.3 % CPU-time of server 1 to service 1 and 54.4 % CPU-time of server 1 to service 2 so that each workflow can produce its optimal throughput.

9. CONCLUSION AND FUTURE WORK

We have discussed the challenges for adaptive resource allocation to manage 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 the multiple workflows in SBS. To develop our approach, we have developed the RAT model for an atomic service and 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 show that our approach will find the optimal resource allocation in polynomial time.

Future research includes investigation of the relationship among the resource allocation, throughput and other QoS features such as timeliness, accuracy, and security to expand our adaptive resource allocation approach to including other QoS requirements, in addition to the throughput.

10. ACKNOWLEDGMENT

This work is supported by National Science Foundation under grant number CCF-0725340. The authors would like to thank Dazhi Huang of Arizona State University for many valuable discussions.

11. REFERENCES

[1] G. Brahnmath, R. R. Raje, A. Olson, M. Auguston, B. R. Bryant, and C. C. Burt, "A Quality of Service Catalogue for Software Components", *Proc. Conf. Southeastern Software Engineering*, 2002, pp. 513–520

s2] J. Ivan, H. Caroline and F. Stephane, "A Comprehensive Quality Model for Service-Oriented Systems", *Software Quality Control*, vol. 17(1), 2009, pp. 65-98

[3] C. Yang, C. Lin and S. Chen, "A Workflow-based Computational Resource Broker with Information Monitoring in

Grids", *Proc. 5th Int'l Conf. Grid and Cooperative Computing*, 2006, pp. 105-206
[4] S. Venugopal, X. Chu and R. Buyya, "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] C. T. Yang, P. C. Shih, C. F. Lin and S. Y. Chen, "A Resource Broker with an Efficient Network Information Model on Grid Environments", The Journal of Supercomputing, vol. 40(3), 2007, pp. 249-267
[6] R. Buyya, C. S. Yeo, and S. Venugopal, "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, pp. 234-242
[7] P. Lardieri, J. Balashbramanian and D. C. Schmidt, "A multi-layered resource management framework for dynamic resource management in enterprise DRE systems", *Journal of Systems and Software*, vol. 80(7), 2007, pp. 984-996
[8] P. E. Hladik, H. Cambazard, A. M. Deplanche and N. Jussien, "Solving a real-time allocation problem with constraint programming", *Journal of Systems and Software*, vol. 81(1), 2008, pp. 132-149
[9] J. N. Hooker and G. Ottoson, "Logic-based benders decomposition", *Mathematical Programming*, vol. 96, 2003, pp. 33-60
[10] J. Stoesser, C. Roessle and D. Neumann, "Decentralized Online Resource Allocation for Dynamic Web Service Applications", *Proc. 4th Int'l Conf. Enterprise Computing, E-Commerce, and E-Service*, 2007, pp. 425-428
[11] M. Mazzucco, I. Mitrani, J. Palmer, M. Fisher and P. McKee, "Web Service Hosting and Revenue Maximization", *Proc. 5th European Conf. on Web Services (ECOWS'07),* 2007, pp. 92-98
[12] M. Kallits, G. Michailidis and M. Devetsikiotis, "Pricing and Optimal Resource Allocation in Next Generation Network Services", *Proc. IEEE Sarnoff Symposium*, 2007, pp. 1-5
[13] F. Meshkati, H. V. Poor and S. C. Schwartz, "Energy-Efficient Resource Allocation in Wireless Networks", *Signal Processing Magazine*, vol. 24(3), 2007, pp. 58-68
[14] C. Bae and D. Cho, "Fairness-Aware Adaptive Resource Allocation Scheme in Multihop OFDMA Systems", *IEEE Communications Letters*, vol. 11(2), 2007, pp. 134-136
[15] S. Stuijk, T. Basten, M. Geilen and H. Corporaal, "Multiprocessor Resource Allocation for Throughput Constrained Synchronous Dataflow Graphs", *Proc. Conf. 44th IEEE annual Design Automation*, 2007, pp. 777-782
[16] Z. Yang, N. Ye and Y.-C. Lai, "QoS model of a router with feedback control", *Quality and Reliability Engineering Int'l*, vol. 22(4), 2006, pp. 429-444.
[17] N. Ye, B. Harish, X. Li and T. Farley, "Batch scheduled admission control for service dependability of computer and network resources", *Information, Knowledge, Systems Management*, vol. 5(4), 2006, pp. 211-226
[18] Q. Zhang, L. Cherkasova and E. Smirni, "A Regression Based Analytic Model for Dynamic Resource Provisioning of Multi-Tier Applications", *Proc. 4th Int'l Conf. Autonomic Computing*, 2007, pp. 27-36
[19] S. S. Yau, Y. Yin and H. G. An, "An Adaptive Model for Tradeoff between Service Performance and Security in Service-based Environments", *Proc. Int'l Conf. Web Services (ICWS 2009)*, 2009, pp. 287-294
[20] L. Eggert and J. Heidemann, "Application-Level Differentiated Services for Web Services," *J. World-Wide Web*, vol. 2, no. 3, 1999, pp. 133-142.
[21]. G. Rao and B. Ramamurthy, "DiffServer: Appliction Level Differentiated Services for Webservers," *Proc. IEEE Int'l Conf. on Communication*, vol. 5, 2001, pp. 1633-1637.
[22] C. Lu, Y. Lu, T.F. Abdelzaher, J.A. Stankovic, and S.H. Son, "Feedback Control Architecture and Design Methodology for Service Delay Guarantees in Web Servers," *IEEE Trans. on Parallel and Distributed Systems*, vol. 17, no. 9, 2006, pp 1014-1027
[23] T.F. Abdelzaher, J.A. Stankovic, C. Lu, R. Zhang, and Y. Lu, "Feedback Performance Control in Software Services," *IEEE Control Systems Magazine*, vol. 23, no. 3, 2003, pp. 74–90.
[24] G. Wang, A. Chen, C. Wang, C. Fung and S. Uczekaj, "Integrated quality of service (QoS) management in service-oriented enterprise architectures", *Proc. 8th Int'l Conf. Enterprise Distributed Object Computing*, 2004, pp. 21-32
[25] S. S. Yau, N. Ye, H. Sarjoughian and D. Huang, "Developing Service-based Software Systems with QoS Monitoring and Adaptation", *Proc. 12th Int'l Workshop on Future Trends of Distributed Computing Systems*, 2008, pp. 74-80
[26] S. S. Yau, N. Ye, H. Sarjoughian, D. Huang, A. Roontiva, M. Baydogan and M. Muqsith, "Towards Development of Adaptive Service-based Software Systems", *IEEE Trans. Service Computing*, 2009
[27] I. Griva, S. G. nash and A. Sofer, "Linear and Nonlinear Optimazation (2nd ed.)", Society for Industrial Mathematics, Philadelphia, PA, 2008
[28] S. Kona, A. Bansal, G. Gupta, and T. Hite, "Web Service Discovery and Composition using USDL", *Proc. 3rd IEEE Int'l Conf. E-Commerce Technology*, 2006, pp. 65-67
[29] M.L. Massie, B.N. Chun and D.E. Culler, "Ganglia Distributed Monitoring System: Design, Implementation, and Experience", *Parallel Computing*, vol. 30, 2004, pp. 817–840
[30] D. Spielman and S. Teng, "Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time", *Proc. 33th Annual ACM Symp. Theory of Computation*, 2001, pp. 296-305