Chapter 6
Performance Management of Composite Applications in Service Oriented Architectures

Vinod K. Dubey
Booz Allen Hamilton, USA

Daniel A. Menascé
George Mason University, USA

ABSTRACT

The use of Service Oriented Architectures (SOA) enables the existence of a market of service providers delivering functionally equivalent services at different Quality of Service (QoS) and cost levels. The QoS of composite applications can typically be described in terms of metrics such as response time, availability, and throughput of the services that compose the application. A global utility function of the various QoS metrics is the objective function used to determine a near-optimal selection of service providers that support the composite application. This chapter describes the architecture of a QoS Broker that manages the performance of composite applications. The broker continually monitors the utility of the applications and triggers a new service selection when the utility falls below a pre-established threshold or when a service provider fails. A proof-of-concept prototype of the QoS broker demonstrates how it maintains the average utility of the composite application above the threshold in spite of service provider failures and performance degradation.

INTRODUCTION

Service Oriented Architecture (SOA) enables a market of service providers delivering functionally equivalent services at different Quality of Service (QoS) and cost levels. This presents a unique opportunity for consumers to pick and choose services that meet their business and QoS needs. The selected services can be orchestrated in a process flow to optimize the execution of business processes in a cost-effective manner. We assume that service providers publish their QoS levels and offer resource reservation to guarantee them within a certain range. We also assume that...
the characteristics of QoS metrics, more specifically QoS levels offered by service providers, may change over time. For instance, the performance of a service may degrade due to heavy workload conditions of the associated service provider or due to some unforeseen unavailability of the service. As a result, this may affect the end-to-end QoS of a business process that may depend on such services. This requires monitoring the performance of services and business processes at runtime and, if needed, taking corrective measures to ensure that the QoS levels of running business processes were not compromised.

This chapter describes the architecture of a QoS Broker that manages the performance of composite applications. The broker facilitates near-optimal service selection for the composite application, continually monitors the utility of the applications, and triggers a new service selection when the utility falls below a pre-established threshold or when a service provider fails. A proof-of-concept prototype of the QoS broker demonstrates how it maintains the average utility of the composite application above the threshold in spite of service provider failures and performance degradation.

**Background**

We assume there will be a market of service providers delivering services at different QoS and cost levels. The service providers specify their QoS metrics in terms of response time, availability, and throughput. Response time (R) refers to the time it takes for a service to respond to a user’s request. It is measured in time units such as sec or msec. Throughput (X) represents the number of requests or transactions completed per unit of time. Availability (A) refers to the fraction of time a system is up and available for use.

A composite application in this chapter refers to a system composed of various services that support the execution of a business process. A business process is defined as a collection of activities connected to one another in a certain workflow to address business needs of an enterprise. The execution time, availability, and throughput of composite applications are computed as a function of the QoS metrics of the individual service providers selected for the business process. This computation of the end-to-end QoS for a business process must take into consideration the constructs used in the business process workflow. For example, the end-to-end execution time of a business process with sequential activities is additive in nature, while for parallel constructs (e.g., fork-and-join), it is the maximum of the response times of the service providers that support the parallel activities. The throughput of composite applications with sequence and flow constructs is the minimum of the throughputs of all service providers chosen to support the business process, while the end-to-end availability is the product of the availabilities of the individual service providers.

We consider a global utility function of these metrics as the objective function used to determine the near optimal service selection to support the execution of a business process. A utility function measures the usefulness of a service or business process to a consumer in terms of various QoS metrics. The utility is typically represented as a scalar with no units. The utility increases with the decrease in response time and increases with throughput, availability, and security. Typically, utility functions are monotonically decreasing for response time and monotonically increasing for throughput, and availability. The monotonicity assumption corresponds to rational user expectations. For example, one would expect a user to see less utility in a system as its response time increases than the other way around.

Utility functions have been used for achieving self-optimization in distributed autonomic systems (Kephart and Das, 2007; Tesauro et. al., 2005). Bennani & Menascé (2005) used utility functions combined with analytical queuing network models to dynamically allocate servers to applications being hosted by an Internet data center. Menascé