Speeding Up Data Access in SOA

Maheshi Lokumarambage[#1] and Chandana Gamage[*2]

[#] *IT Group, Sri Lanka Telecom PLC, Colombo, Sri Lanka*

[1]maheshi@slt.com.lk

[*] *Department of Computer Science & Engineering, University of Moratuwa, Sri Lanka*

[2]chandag@cse.mrt.ac.lk

***Keywords*— SOA, ESB, SOAP, Performance Benchmarking.**

Owing to the edge that IT provides to one's business over potential competition, it is important that IT infrastructure effectively leverage the business processes of an enterprise. SOA is a paradigm that can be used to build such enterprise architectures enabling those with needs (consumers) and those with capabilities (providers) to interact via services across disparate platforms, technologies, and ownership. Services are the cornerstone in service-based architectures where they act between the consumers and the providers.

ESB is a deployable system that makes enterprise application integration a reality through SOA. It plays the role of an intermediary that orchestrates the service requests for various applications for optimal service delivery. The intention of utilizing an ESB is to enable smooth operation among the diverse Enterprise Applications, but there is a common complaint of degraded performance when fulfilling the requests, unlike in the standalone silos-like applications.

The promise of agility and scalability of SOA comes at a price; the performance is impacted when an intermediary is operating in between the enterprise applications, performing various processing tasks before the requests being redirected to the actual service end-point. Degradation of performance can occur as the system scales in terms of the number of services and the number of users. The objective of the research presented in this paper is to characterize the performance of SOA in the presence of an ESB.

The research work has focused on the performance behaviour of SOA under increasing concurrent user requests, and using different SOAP payloads with different complexities. A set of benchmark Web Services have been developed which represent different use case scenarios. These benchmarking Web Services were tested against the different capabilities of the ESB. Those capabilities include direct proxy, content-based routing and caching enabled in the ESB. Another existing Web Service benchmark was then used to represent a case where the processing overhead is negligible in the web server side.

Test results have revealed that before designing an SOA for an enterprise, it is wise to have a thorough understanding of the factors that affect the performance of SOA, and their behavior under varying conditions. This research shows that various factors, like the number of concurrent user requests, the primitive data types used in the payload, SOAP payload size, cache expiry time configured in the ESB and the number of database fetches, all determine the SOA performance when used with an ESB.

The testing scenario of this research was designed considering an industrial SOA implementations for Online Transaction Processing (OLTP) environments, such as the Telecommunications Industry. The payloads are not very large since the typical OLTP systems are used for purposes such as order entry, financial transactions, Customer Relationship Management (CRM) and retail sales. Such systems have a large number of users who conduct short transactions. Database queries are of medium complexity, which returns relatively few records. In comparison to OLTP data base search complexity, the developed benchmark web service used in this experimental analysis has a complex data search, which joins several tables of a very large industrial database.

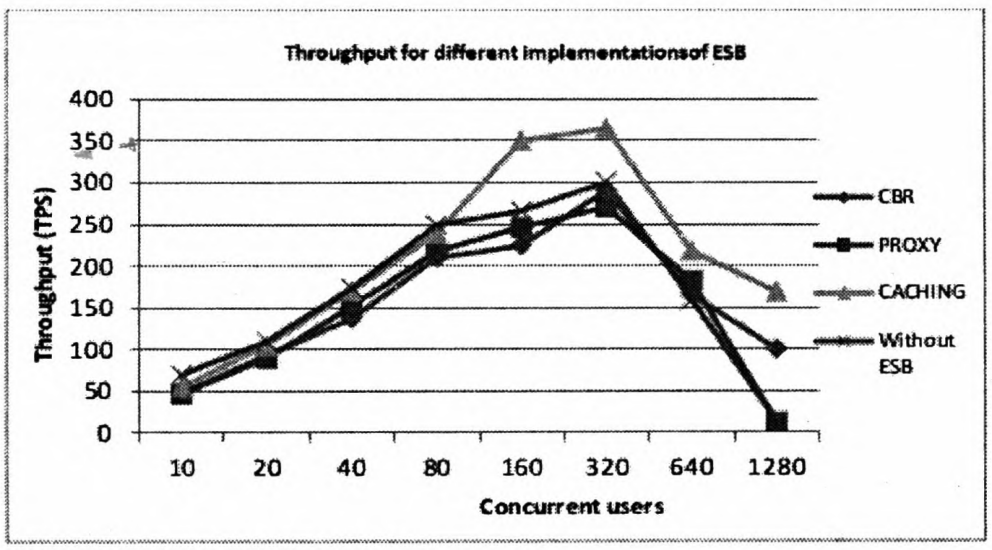

FIGURE I
INDUSTRIAL WEB SERVICE THROUGHPUT FOR DIFFERENT ESB IMPLEMENTATIONS

As a sample of test results, the throughput variation under different implementations of the ESB with increasing number of concurrent user requests is illustrated in figure 1. The most interesting observation is that for the caching implemented in the ESB scenario, the performance is much higher than the other implementations. From a theoretical perspective, the scenario of direct web service access should perform better than all the other cases but it was not the observation. The reason may be that the already retrieved responses hash value is cached and the web service processing is not needed to access the underneath database to retrieve the information. The reasoning can be further analyzed by conducting the testing for the existing benchmark suit, where the web services only echoes the provided strings, doubles, etc.