Building a Service-Oriented Architecture Benchmark

Peter Budny
peterb@cc.gatech.edu

Abstract

In order to facilitate future research on service-oriented architectures, we create a simulation system which can be used as an experimental platform and a benchmark. This simulator is based on a real-world system: an airline fare pricing/ticketing website and the necessary services to support it. We describe the system our simulator is based on and the modifications we made to make the simulation more amenable to supporting research.

1 Introduction

The term "service-oriented architecture" is something of a buzzword. There are a multitude of definitions, some narrow, some wide; it is surrounded by confusion, and is still a very abstract concept. There is also no one system people can point to as an example of a service-oriented architecture or a point of reference for understanding research regarding service-oriented architectures.

In this paper we define what a service-oriented architecture is, and separate it from other similar concepts like Web services. We show that there is a need for a tool or software package which can suitably act as a benchmark for research on service-oriented architectures, and that no such system currently exists. We then introduce our simulation system, based on a functioning business application, and show how it meets the criteria for becoming a research benchmark for service-oriented architectures.

2 What is service-oriented architecture?

Service-oriented architecture, or SOA, is an architectural design pattern in which an application is composed of loosely-coupled components which export and import services.[8] A service is a function in which the request is posed as a question, and the response is the answer to that question. In a SOA, each service is specialized to only answer certain types of questions. Applications are built by composing services upon each other.

SOA is the architectural equivalent of abstraction. Each service provider solves a single problem. This makes it possible to build interesting applications that do hard things by letting service providers at lower levels solve the hard problems.

2.1 SOA vs. Web services

SOA tends to be confused or conflated with "Web services". Unfortunately, "Web services" is perhaps more poorly-defined than "service-oriented architecture", and suffers from multiple meanings.

1. Web services, *noun* – standards including SOAP, XML, WSDL, UDDI, and WS-*.
2. Web service, *noun* – an architectural design pattern in which programmatic interfaces allow two applications to communicate directly to each other.

   The Web service standards (1) are being created to support Web service architectures (2).
3. Web service, *noun, adj.* – an application constructed using Web service standards (1) *or* Web service architecture (2).

Making matters worse, "Web services" is often used narrowly but without defining which sense is meant. Most commonly, it is used to mean "an architecture in which a SOA is built using Web services standards". Used in this sense, Web services are a proper subset of SOA. In any other meaning, Web services are either orthogonal or outright unrelated to SOAs.

2.2 Defining SOA

Having addressed Web services, we can dispel some common misconceptions about SOAs, many of which derive from confusion with Web services.[7]

Myth SOAs use standardized interfaces.

Reality Standardized interfaces encourage competition; many commercial service providers define proprietary interfaces to lock users in. (SOAs do have "standard" interfaces in the sense of "well-defined".)

Myth SOAs communicate using XML.

Reality SOAs can be built using any format to exchange messages and data.

Myth SOA precludes the use of RPC or REST.

Reality Although each have drawbacks that make integrating them into a SOA difficult (RPC is rigid and tightly-coupled, REST encourages transferring data using state), both have their uses in SOA.

Myth SOAs implement service discovery or a service registry.

Reality The idea of service discovery implies that every provider of a service is equal. While this may be true in an ideal world, in reality some service providers may give results faster, better, or cheaper than others.

The above are all superficial features, which would not tell us anything about what SOA is even if they were true. To truly define SOA, we must look deeper at what sets it apart from other architectures.

Dynamic, not static Although a SOA may operate with static dependencies on particular service providers, it must still be feasible (though not necessarily easy) to replace one service provider with another.

Descriptive requests, not instructive A service exists to transform data from one form to another, or to answer questions. Therefore, queries in a SOA should describe the problem to be answered, not how to solve it programmatically. (In other words, queries in SOAs should only be interpretable as questions, not procedures.[6])

If a query can be phrased procedurally, then the interaction is data-oriented, not service-oriented.

Idempotent requests Since a service answers a question, it is expected that the answer should not change between requests. However, services may be built on top of finite resources, in which case a response involving a resource may change when the resource has been exhausted.

Structured responses, not arbitrary data A service's response must be structured data which answers the question posed by a request. Arbitrary, unstructured data cannot make up the entire response.

An architecture, not a standard SOA is a design pattern, independent of any standards.

3 The need for a SOA benchmark

The purpose of a benchmark is two-fold: it solidifies people's ideas about the concept embodied by the benchmark, and it acts as a measuring stick for understanding new ideas on a common ground.

3.1 Other potential benchmarks

There are many ongoing projects in the domain of SOAs. However, individually none of them meet the needs of a successful SOA benchmark.

RUBiS is an online auction simulator modelled after eBay.[4] Although it is a good simulation of a production system, it is a single application which does not consume any services, and does not have any obvious paths for expansion.

Java Adventure Builder is a sample application demonstrating Web service standards on the J2EE platform.[2] However, it is a single application, and does not have the service abstraction necessary to be considered service-oriented. Further, because it is built solely on Web service standards, it is difficult to use for exploring alternative standards. Lastly, the data set provided with it is very small, and not representative of a functioning application.

Nutch is a search engine based on Lucene Java, an indexing and search backend.[3] Search engines meet the criteria to be considered service providers, but not service consumers, since they do not take in structured data. Nutch has a modular architecture supporting plug-ins for media-type parsing, data retrieval, querying and clustering; however, this structure is too rigid and closed to be considered service-oriented. Furthermore, users must provide their own data for Nutch to index.

Intel Mash Maker is a tool for allowing non-expert users to create mashups, or queries on two or more related data sets.[5] Although it is primarily designed to take in semi-structured data, it could also take in fully-structured data (i.e., consume services). It could also be considered a service provider in its own right. However, by itself it is only a single service consumer/provider, not a complete SOA. It is also not currently open source, which discourages use in a research environment, and does not come with a data set or a load generator.

Apache Tuscany is an infrastructure for creating SOAs based on specifications defined by the Open SOA Collaboration, but is not itself a SOA or a service-oriented application.[1]

3.2 Criteria for a benchmark

Based on the issues with each of these systems, we propose that a successful SOA benchmark must...

- ...be a complex, functional application built from multiple composable units. Each unit should be a service provider and/or consumer, and every service provider should be meaningful on its own, apart from the services built on top of it.
- ...have possibilities for expansion. Since SOA is about the composition of services to provide new functionality, a SOA benchmark must have obvious ways in which new applications can be created on top of it.
- ...come with large data set. Commercial applications deal with large volumes of data; a believable benchmark should also be provided with a large, realistic data set.
- ...be open source.
- ...be reusable, not purposed for a single experiment or class of experiments.
- ...be extensible and easy to integrate with other tools, monitors, and benchmarks.
- ...not rely heavily on standards. A benchmark should be able to support and test any standard; tying its design or implementation to a single set of standards closes this opportunity.

4 A reference system

As a reference system, we use the airline travel industry; specifically, systems which price and book airline tickets. Such a system is called a *global distribution system* (GDS). These systems provide pricing and ticket sales for major airlines, independent travel agents, and travel websites like Expedia or Orbitz which allow end-users to independently search for and purchase airline tickets.

4.1 About airline fares

Prices for airline flights are not static; they are built dynamically from rules. These rules are collected and published several times daily. During the ∼10 months in which tickets for a flight are available, the rules governing the pricing of that flight may undergo frequent modification. In addition, prices are calculated based on run-time input such as:

- the date and time of the flight
- the travel class (i.e., first, business, or economy)
- various discounts and restrictions that may apply (e.g., senior citizen discounts, advance purchase requirements, blackout dates, etc.)

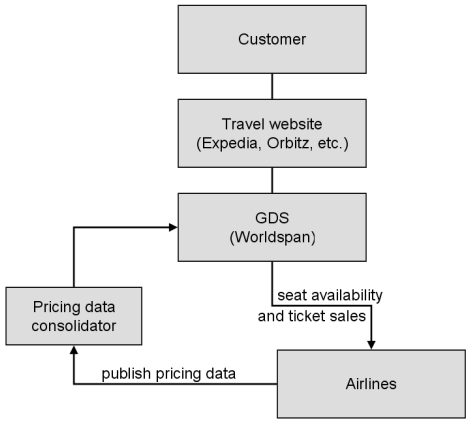

Figure 1: Worldspan's interactions with other systems

Pricing a flight is a matter of finding the combination of discounts and restrictions that yield the lowest price. The GDS may be liable for any differences in price from what the airline quotes as the correct price. They must also return a response within a time allotted by their SLA.

4.2 About Worldspan

Our particular architecture comes from Worldspan, a GDS whose users include Delta Air Lines, Northwest Airlines, Orbitz, Hotwire, and Priceline. These retailers use Worldspan to calculate ticket prices, check seat availability, and book and purchase tickets. Worldspan in turn relies on airline fare consolidators to provide a data feed, and also communicates directly with airlines to check seat availability and book tickets.

Worldspan serves an average of 11.6 million requests per day, with an average response time of ∼2 seconds. The data set is 3 GB for domestic pricing data, which is updated three times daily, and 13 GB for international, updated five times daily.[1]

Worldspan's architecture is based around a 1500-node farm used to price queries; each node runs two query processes. The pricing data is stored in four load-balanced SQL database servers, which are accessed in two different ways. First, after updating the database with new prices, data for frequently-used markets is loaded into a 1.5 GB cache file which is then pushed to the nodes in the farm. This cache file is loaded into shared memory and used by the two query processes. Second, the databases serve requests on-line for data not contained in the cache file.

[1]International pricing queries may also require the domestic data, so the combined set is actually 16 GB and updated eight times daily.

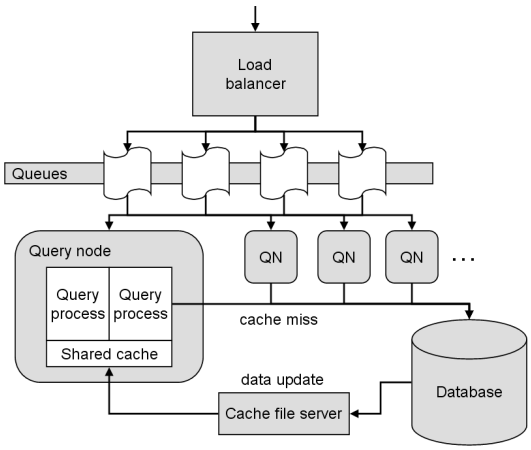

Figure 2: Worldspan's internal architecture

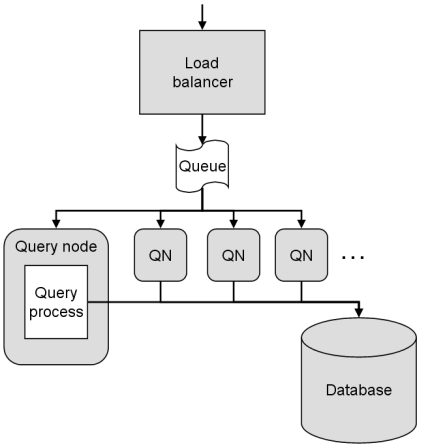

Figure 3: Our SOA benchmark is a simplified version of Worldspan's architecture

5 Our implementation

Our simulator is based on a simplified version of Worldspan and the systems it interacts with. It has been designed to meet the aforementioned goals of a SOA benchmark.

Our initial implementation has three basic components:

Customer The customer is represented by a load generator, which generates requests to the travel website.

Travel website The travel website is a generic clone of Expedia or Orbitz.

GDS The GDS consists of a database, one or more query nodes, and a load balancer.

The load generator communicates with the travel website over HTTP. The travel website uses synchronous SOAP calls to request pricing data from the GDS.

Inside the GDS, the load balancer communicates asynchronously with the query nodes via queues. This simplifies the load balancer, and also makes it easier to have the load balancer re-send a query to the nodes if it does not receive a timely response for any reason.

We have deliberately kept the scope small and the design simple so that we can release our benchmark quickly. The query nodes, for instance, do not have caches; data is pulled from the database on every request. The GDS also does not have the ability to process data updates or sell tickets. Since we are uncertain which features will be most useful, we intend to start with a small version and implement additional features based on demand and need.

6 A successful benchmark

By choosing a reference application and designing a system with the criteria mentioned earlier in mind, we are able to produce a system which is capable of serving as a SOA benchmark.

It is a non-trivial application which successfully simulates an complex, multi-layer system, structured as a service-oriented architecture. The GDS and the travel website are each independent implementations, meaningful and useful without the applications above them.

It can be expanded in many directions. There are plenty of components and capabilities from Worldspan's architecture which are absent in our initial implementation—data updating, seat availability and ticket purchasing, data caching on query nodes, etc.—which could be used to expand the scope and depth of the simulation. In addition, it could be combined with other well-known tools (e.g. Java Adventure Builder, Intel Mash Maker, Nutch, etc.) or with creative new applications.

It is built using open technologies: JMS, Tomcat, MySQL, SOAP, etc. Its clean, transparent design encourages modification and integration with other tools and applications.

It is a general-purpose simulation which does not comprise an experiment of its own. Many different experiments can be built on top of it.

It relies on a minimum of standards. Web service standards (WS-*) in particular have been excluded; the system should be able to integrate with these standards as verification of their applicability, but it should not be tied to them to the exclusion of other standards. The few standards it does use (SOAP) can easily be substituted.

Perhaps most importantly, the system is distributed with a large data set. Worldspan has generously provided us with data for use in our simulations. This includes real

pricing data, which covers several major airlines, regional airlines, and small competitive low-cost airlines. The total size of this data is >1 GB. When we implement data updates, Worldspan will provide us with additional data representative of real update feeds. Worldspan has also provided us with request traces, which are used to generate realistic traffic patterns on the travel website and the GDS.

7 Future work

Because we have simplified Worldspan's architecture in order to quickly implement a system and distribute it, there is much more work to be done in expanding it to have more of the features found in the reference application. Future expansion of the application will be guided by user demand, both external and internal.

We will attempt to integrate other applications often cited as being service-oriented such as Java Adventure Builder or Intel Mash Maker to demonstrate that many service-oriented applications can be integrated into our benchmark successfully. We will also explore how the various standards for managing SOAs (e.g., Web service standards, Tuscany, etc.) can be applied to our system.

8 Conclusion

In this paper we have defined "service-oriented architecture", and shown the need for a benchmark to act as a clear model of a SOA and a common ground for promoting and understanding future research. We have introduced our simulation system, based on a real application in the airline travel industry, which meets the requirements for being a SOA. Finally, we showed how our simulation system meets the requirements for it to serve as a benchmark of SOAs in future research.

References

[1] Apache tuscany. Online.

[2] Java adventure builder reference application. Online.

[3] Nutch. Online.

[4] Cecchet, E., Marguerite, J., and Zwaenepoel, W. Performance and scalability of EJB applications. In *Proceedings of the 17th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications* (2002), pp. 246–261.

[5] Ennals, R. J., and Garofalakis, M. N. MashMaker: Mashups for the masses. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data* (New York, NY, USA, 2007), ACM, pp. 1116–1118.

[6] He, H. What is service-oriented architecture. *XML.com* (Sept. 2003).

[7] Kodali, R. R. What is service-oriented architecture? *JavaWorld.com* (June 2005).

[8] Rotem-Gal-Oz, A. What is SOA anyway? *rgoarchitects.com*.