WorldTravel: A Testbed for Service-Oriented Applications

Peter Budny, Srihari Govindharaj, and Karsten Schwan

Center for Experimental Research in Computer Systems
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332, USA
{peterb,srihari,schwan}@cc.gatech.edu

Abstract. This paper describes the "WorldTravel" service-oriented application and testbed. The purpose of the testbed is to provide to researchers an open source venue for experimenting with and evaluating ideas, methods, and implementation options for service-oriented architectures and applications. Built upon standard service technologies, the WorldTravel testbed offers implementations of services and service interactions specific to the WorldTravel application, comprised of (1) a substantive back-end that includes a simple airline pricing/ticketing engine, with a representative flight database, both structured similarly to those used by companies actually offering such services, (2) a front-end for travel services interacting with mid-tier request processing and routing services, and (3) load traces from the corresponding business applications that are used to drive the use of WorldTravel and its services.

We call WorldTravel a testbed rather than benchmark because its design permits extension at both the front-end, e.g., to add interesting new services like weather information about possible travel destinations, and at the back-end, e.g., to add payment services. This paper identifies the need for testbeds like WorldTravel, considers the attributes required of such testbeds, describes our current testbed in detail, and presents an initial testbed evaluation. It also describes the actual production-quality system on which WorldTravel is based.

1 Introduction

To pursue research in "service-oriented" architectures and applications (SOA), it is important to have available representative service examples and implementations, an analogous example being the well-known RUBiS eBay-like benchmark used to evaluate ideas and implementation methods for multi-tier web services [1,2]. RUBiS provides a representative back-end database, implementations of front- and mid-tier services that carry out tasks like searching for items and bidding on them, and load traces for front-ends that use these services. Since RUBiS is open source, researchers can extend or change it, using it to evaluate implementation ideas or methods for alternative ways to invoke services [1] and/or implement them, etc. In contrast, the well-known SPEC

A. Bouguettaya, I. Krueger, and T. Margaria (Eds.): ICSOC 2008, LNCS 5364, pp. 438–452, 2008.

benchmark for evaluating compiler technologies and TPC benchmarks for file systems, transactional services, and web services have as a principal goal to evaluate existing implementations and assess or compare their performance or reliability attributes.

Our goal is to enable research in service-oriented architectures, infrastructures that support their concepts, and applications based on these concepts. Toward those ends, this paper provides a novel testbed, called WorldTravel, which permits researchers to experiment with the SOA technologies of interest to them, construct their own SOA applications extending the front-end or back-end services of WorldTravel, and use WorldTravel load data to evaluate their implementations. Specific functionalities of WorldTravel highlighted in this paper are its methods for synchronous vs. asynchronous service interactions, the distinction of front-end from back-end services and the various service interactions across these different kinds of services, and the testbed's extensibility with respect to adding new services or changing existing ones. In addition, we differentiate the services-based approach used in WorldTravel from that taken in multi-tier web service testbeds like RUBiS [2], and articulate the need for testbeds like it.

In the remainder of the paper, we first define in Section 2 what we mean by "service-oriented architecture" with respect to the WorldTravel testbed, and separate it from related concepts like "Web Services". Section 3 examines other potential testbeds to determine what qualities and attributes a successful testbed should exhibit. Section 4 explains the system we have chosen to model: an airline fare pricing engine. Section 5 details the implementation of WorldTravel. Sections 6 and 7 demonstrate how WorldTravel meets the previously-defined qualities and attributes of a successful testbed. Section 8 discusses future work to be done on WorldTravel and concludes the paper.

2 Service-Oriented Architecture

Service-oriented architecture, or SOA, is an architectural design pattern in which an application is composed of loosely-coupled components that export and import services [3]. A service is a function in which the request is posed as a question, and the response is the answer to that question. In SOA, each service is specialized to only answer certain types of questions. Applications are built by composing services with each other statically or, more interestingly, at runtime.

SOA is the architectural equivalent of abstraction. Since each service provider solves a single problem, this makes it possible to build interesting applications that perform complex tasks by letting service providers at lower levels solve some of the problems, thereby freeing the upper-level from having to worry about these problems.

In the rest of this section we will refine the concept of "service-oriented architecture" as we use it with the testbed we propose. Our intent is not to constrain users to a single definition but rather to clarify the broad mindset taken as we designed the testbed.

2.1 SOA and Web Services

Since SOA applications typically run in Internet settings, the term SOA is often linked with "Web Services", the latter characterizable in multiple ways:

1. web services, *noun* – a collection of 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; [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.) <u>*or*</u> web service architecture (2.).

Perhaps most commonly, the term "web service" is used somewhat narrowly to describe "an application designed as a SOA and implemented with 'web service' standards". When used in this sense, web services are a proper subset of SOA.

2.2 Defining SOA

Having addressed web services, we next discuss the specific qualities beyond the notion of web services associated with SOA, in part to address some common misconceptions that derive from the fact that SOA implementations often use web services [4].

Concept. An application is service-oriented iff it uses certain standards.
Reality. SOA indicates the style in which components of a solution are tied together, while standards define specific implementations for doing so. Although standards encourage competition and interoperability by making it easy to switch service providers, this concept is often negated by the use of proprietary standards for the interconnection of services to promote vendor lock-in.
Concept. SOAs communicate using XML.
Reality. SOAs can be built using any format to exchange messages and data, where such alternatives are often used for reasons of improved performance (e.g., binary formats [5,6,7]) or to support legacy applications.
Concept. SOA precludes the use of RPC [8], RMI [9], or REST [10].
Reality. Each of these invocation protocols have attributes that make integrating them into a SOA difficult, yet such integration is often done. RPC and RMI are tightly-coupled compile- or link-time protocols integrated into the middleware methods used by distributed applications. This makes them most suitable for somewhat 'static' components of these applications (e.g., core back-end services like the database accesses in RUBiS), and while SOA encourages adaptability, services invoked using RPC or RMI can certainly be substituted with only moderate effort. REST, on the other hand, encourages transferring data using state, which makes ensuring idempotency difficult (see *Idempotent requests*), but is inherently well-suited for stateful services. With care, it can also be used to implement stateless services.
Concept. SOAs implement service discovery or a service registry.

Reality. Runtime service discovery and subsequent service use face difficulties in that it is not reasonable to assume that all providers of services are 'equal'. As a result, there is much recent research on dealing with service equivalence or translation, resulting in ontology-based methods for understanding degrees of similarity or equality and semantic web-based methods for doing so [11,12]. Beyond such functional equivalences, also of interest are performance or reliability differences between services, addressed in part by recent work on standards in the domain of autonomic computing [13,14].

The high level features of SOA described above are implemented using several basic techniques:

Composition. SOA is a form of distributed computing in which subtasks are distributed and treated as blackbox operations, possibly even handled by third parties. Applications are structured by abstracting and composing services vertically to provide desired higher level functionality.[1] Composition is intended to be dynamic, but issues persist concerning the viability and generalizability of dynamic composition for commercial applications with required service guarantees.

Descriptive requests. Queries in a SOA should describe the problem to be answered, not how to solve it programmatically. This means that queries in SOAs should only be interpretable as questions, not as procedures [16]. (SQL, for instance, is *not* service-oriented; every SQL command contains a verb (e.g., `SELECT`, `INSERT`, `DELETE`, etc.) and describes how to manipulate data. A hypothetical service-oriented database manipulation language would instead have 'question' words like "what" or "how many".)

Idempotent requests. Since a service answers a question, it is expected that the answer should not change between requests. However, services will have finite, dynamic resources, which means that a response involving a resource may change when the underlying resource has been modified. For stateful services, responses may differ due to changes of internal states caused by previous requests, but responses must still be idempotent for a combination of a request, the state at the time of the request, and the resources involved.

Structured responses. 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.

3 Utility of a SOA Testbed

We developed the WorldTravel testbed for multiple reasons. First, it can provide a useful basis for experimentation, both with new services and service–service interactions and with new methods that improve SOA implementations (e.g.,

[1] In contrast, horizontal composition, as exemplified by load balancing, MapReduce [15], and many other well-known techniques, is used most often as a way of enhancing performance, and is orthogonal to SOA; horizontal composition/distribution can be used both within a service and between equivalent services.

improved discovery methods, improved invocation methods, etc.). We also hope to provide researchers with an environment in which such ideas can be compared, in performance and/or usefulness, but it remains difficult to assess the performance effects of functionality like runtime service discovery and use, since 'typical' behaviors for such actions are not yet known. On the other hand, just as compiler developers agree on certain workloads to be representative of certain environments (e.g., selecting certain SPEC benchmarks), for testbeds like WorldTravel, useful request traces and request loads can be derived both from standard Internet measurements (e.g., diurnal changes in request behavior [17,18]) and in our case, from load information provided by our corporate partner, Travelport. A final purpose of the WorldTravel SOA testbed is for it to give rise to a set of testbeds (developed outside our group) that will display common, defining characteristics of SOAs.

3.1 Other Potential Testbeds

We are not aware of other SOA testbeds, but point the reader to the following related efforts that have helped shape WorldTravel and its implementation.

RUBiS is an online auction simulation modeled after eBay [2]. It is built using a standard three-tier architecture, has a sample database and offers representative load traces. Since RUBiS is focused on a single application, bidding, opportunities exist to extend its back-end (e.g., payment methods) or front-end (e.g., comparative bidding), but such extensions are difficult to perform because they must be intimately integrated into the RUBiS implementation. We note that there are other applications like RUBiS for Java-based multi-tier web service implementations, available from companies like IBM, but since they rely on IBM's Websphere middleware, they are not suited for constructing a suitable testbed. Finally, earlier versions of applications like RUBiS are even simpler (e.g., PetStore) and are also not useful building blocks for our work.

Java Adventure Builder is a sample application demonstrating web service standards on the J2EE platform [19]. We considered using it as a basis for our work, but it does not offer the abstraction necessary to be considered service-oriented; rather, it is structured using a tightly-coupled three-tier architecture. Further, because it is built solely on web service standards, exploring alternative standards in its context would imply a complete re-implementation of its functionality. Lastly, the data set provided with it is quite small and is not representative of a fully functioning SOA.

Nutch is a search engine based on Lucene Java, an indexing and search back-end [20]. Search engines certainly constitute an interesting class of service providers, but the open source Nutch implementation does not use service-based interactions with the associated web crawler. Instead, it requires the index to be produced by users based on crawling their own set of websites. It does offer some plug-ins for media-type parsing, data retrieval, querying and clustering, but those plug-ins do not use standard service interfaces or SOA methods. Finally, it does not offer other interesting interfaces, such as those concerning ad generation

and placement, etc. *Hadoop* is a related effort that enables end-users to construct their own MapReduce functions to assist in fast, scalable searching [21].

Intel Mash Maker is a tool for allowing non-expert users to create mashups, or queries on two or more related data sets [22]. 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 construed as a service provider in its own right. However, its implementation within a single web browser reduces it to being a single service consumer/provider, rather than being a complete SOA. The fact that it is not open source discourages its use research environments.

Yahoo! Pipes is a web application providing users with simple building blocks to aggregate web feeds, web pages, and other services, manipulate and combine content to create new web applications, and publish the resulting applications [23]. Like Mash Maker, it can take in structured data, and it provides services by offering them in a publicly-available fashion. However, it has a limited set of sources for structured data; data from other sources can only be fetched as unstructured strings. Further, since only basic data manipulation functions are provided, it would be difficult to build a practical business application with the available functions. It is also not open source and is hosted solely by Yahoo!, therefore limiting its use in research environments.

Apache Tuscany is an infrastructure for creating SOAs based on specifications defined by the Open SOA Collaboration [24], but is not itself a SOA or a service-oriented application.

httperf and *StreamGen* are tools focused on specific functionality useful for testbeds like WorldTravel, benchmarks like RUBiS, and streaming applications like those developed in the multimedia domain [25] and for database query-structured business monitoring or compliance codes [26,27,28]. Their role is to make it easy to generate request streams using standard loads (e.g., normal or Poisson distributions) and/or to offer non-standard loads acquired from trace files in a standard form. We use both in our research and with WorldTravel.

3.2 Criteria for a Testbed

In accordance with the issues raised in the previous section, we articulate the following criteria for our WorldTravel and other useful SOA testbeds. They should:

- be executable, complex, functional applications built from multiple composable units where each unit should be a service provider and/or consumer, and every service provider should be meaningful on its own, separately from the services built on top of it;
- be extensible, particularly because a key element of SOA is its support for composition of services to provide new functionality; thus, for our SOA testbed, it should be easy to construct new applications in its context and with its implementation;
- come with large data sets, to better represent realistic commercial applications that deal with large volumes of data;
- be open source;
- be reusable, not purposed for a single experiment or class of experiments;

- be easy to integrate with other tools, monitors, and benchmarks, and easy to change to permit experimentation with alternative composition methods, new discovery methods, etc.; and
- not rely heavily on specific standards, so they can support and be used to evaluate alternatives.

4 Reference System

As a reference domain, we use the airline travel industry; specifically, systems that price and book airline tickets [29], which are termed *global distribution systems* (GDS). A GDS provides services that include pricing and ticket sales for major airlines, independent travel agents, select airline sites (e.g., Delta Air Lines in the case of our corporate partner, Travelport), and travel websites like Expedia or Orbitz, the latter permitting customers to independently search for suitable fares and purchase airline tickets.

4.1 About Airline Fares

Prices for airline flights are not static; they are calculated dynamically from rules. These rules are collected and published several times daily. During the $\sim$ 11 months in which tickets for a flight are available, airlines modify the rules governing the pricing of that flight to reflect supply and demand. In addition, pricing rules utilize run-time input such as:

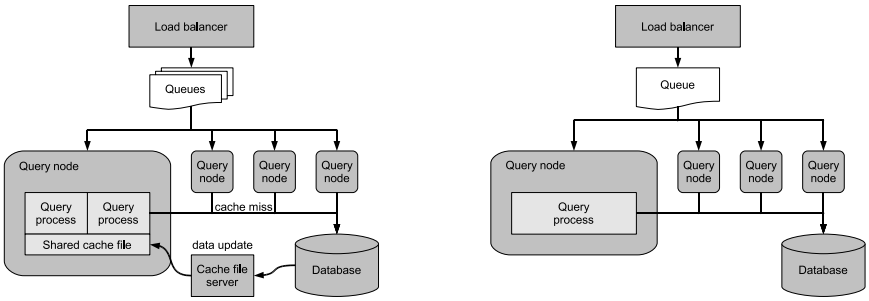
- the date and time of the desired flight;
- the travel class (i.e., first, business, or economy, which are normally broken into as many as 20 "buckets");
- the number of seats currently available; and
- various discounts and restrictions that may apply (e.g., senior citizen discounts, advance purchase requirements, blackout dates, etc.).

Pricing a flight is a matter of finding the combination of discounts and restrictions that yield the lowest price. In fact, the GDS may be liable for any differences in price from what the airline quotes as the correct price. Responses must be returned within a well-defined time and with well-defined complexity (e.g., number of fare options returned) governed by SLAs negotiated with services that use the GDS (e.g., Priceline).

4.2 About Worldspan

Our particular architecture comes from Worldspan by Travelport, a GDS whose users include Delta Air Lines, Expedia, 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 for updates to pricing rules, 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 $\sim$ 2 seconds. The data set is 3 GB for domestic (i.e., U.S.



(a) Worldspan's internal architecture (b) WorldTravel's GDS architecture

Fig. 1. WorldTravel uses a simplified version of Worldspan's architecture

and Canada) pricing data, which is updated three times daily, and 13 GB for international, updated five times daily.[2]

Worldspan's architecture is based around a 1500-node farm used to process pricing queries; each node runs two query processes. 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 read into shared memory and is used by the two query processes. Second, the databases serve requests on-line for data not contained in the cache file.

5 Implementation

The WorldTravel testbed is based on a simplified version of Worldspan and the systems with which it interacts. It has been designed to meet the aforementioned goals of a SOA testbed.

The WorldTravel design and implementation are deliberately straightforward and limited in scope and complexity. For instance, query processes do not have caches, so that data is pulled from the database on every request. This also makes it easy to experiment with alternative caching methods and implementations. The GDS also does not have the ability to process data updates or sell tickets, which makes it possible to experiment with alternative ticketing or payment services. The front-end uses a standard web server, but the mid-tier request distribution service uses open source web technologies rather than the proprietary middleware from Delta Air Lines used in our earlier work [30] or IBM's commercial MQ middleware used by Worldspan [31].

WorldTravel is currently available at `http://www.cc.gatech.edu/systems/projects/worldtravel/`.

[2] International pricing queries also require the domestic data, so the data for international queries is actually 16 GB and updated eight times daily.

5.1 Overview

As a service-oriented application, we make a distinction between a travel website, which provides an interface for users to easily search for fares, and a GDS, which performs the task of pricing fares. A GDS may provide the same information to many different websites or to other clients, and the price of a flight can be calculated by anyone with access to the data. The result is a strong incentive for use by multiple end-user services (e.g., those provided by Expedia, Priceline, and the airlines).

One of the complexities of service-oriented applications is that it may be unknown how long a request will take to process. As a result, asynchronous operations are common in SOA. Acknowledging this fact and for generality, WorldTravel implements multiple communication methods between services: (1) asynchronous with polling, (2) asynchronous with call-backs (i.e., event-based), and (3) synchronous, implemented as a wrapper around a polling interface. The services we implemented in WorldTravel all use asynchronous communication with call-backs, which frees the applications from having to keep state about ongoing queries and makes it easier to scale them. It also lets us potentially deliver responses to a different node than the one that originated the request, thus further enhancing scalability. The synchronous wrapper, on the other hand, is provided for simple applications to use, and frees them from having to handle multiple connections, instead enabling a simple call-and-response invocation style much like HTTP or other web protocols.

To protect Worldspan's intellectual property, we have applied transformations to the data they have provided so that it cannot be used to price actual flights. Ersatz prices are generated and can be used by other services (e.g., payment services or price comparison services). Unlike Worldspan's optimized implementation, which is tuned using domain knowledge about the pricing rules, the price search we have implemented is much more straightforward. This gives users the freedom to tune the engine as desired (e.g., by optimizing the search order, multithreading, etc.). This is because our primary goal is to provide a representative workload on a system constructed using SOA principles; while prices calculated by WorldTravel may be artificial due to the data transformations, the system as a whole still exhibits the same characteristics as Worldspan's functioning application.

WorldTravel is implemented as several distinct components. A minimum setup requires five nodes: three for the GDS (database server, query node, and load balancer), one for the travel website, and one for the load generator. The load balancer and travel website are implemented in Apache Geronimo; the query node in plain Java. The database server we use is MySQL, but any compatible database can be substituted.

5.2 GDS

The GDS consists of a database, one or more query nodes, and a load balancer. The load balancer acts as the front-end of the pricing service, accepting requests

and returning responses once they have been calculated. Inside the GDS, the load balancer communicates asynchronously with the query nodes via queues. The use of queuing to pass queries from the load balancer to query nodes and back is again modeled on Worldspan's architecture, which uses reliable queues to guarantee message delivery. By using multiple queues and a "partitioning" load balancer, Worldspan can also segment their query nodes for specialization (e.g., differing caches representing different markets).

5.3 Travel Website

The WorldTravel travel website is a generic clone of websites like Expedia or Orbitz. The website is also asynchronous, much like a real travel website. Upon submitting a pricing request, users are shown a page asking them to wait, which periodically refreshes to check if a response has arrived at the web server. If so, the response is processed and displayed; if not, the wait page is displayed again.

5.4 Customer

The customer is represented by a load generator, which generates requests to the travel website. The loads come from request traces provided by Worldspan, and they reflect some of the complexities particular to travel websites, such as geographical searches that vary based on time of day due to global users being in disparate time zones.

6 Testbed Analysis

By choosing a reference application and designing a system with the criteria mentioned earlier in mind, we are able to produce a system capable of serving as a SOA testbed. WorldTravel successfully emulates a complex, multi-layer system, structured using service-oriented architecture, which means the GDS and the travel website are each independent implementations, meaningful and useful without other applications using them or based on them. A multitude of interesting extensions and changes are possible, as discussed next.

WorldTravel can be expanded in many ways, and it offers rich possibilities for composition with other services. The testbed can be extended with front-end services or other well-known tools, such as Java Adventure Builder, Intel Mash Maker, Nutch, etc., or with creative new applications. One idea currently being pursued by our group is the addition of services like event ticketing and hotel booking, to create a system which can handle multiple reservations (e.g., an event ticket, a hotel, and a flight) in a transactional manner, guaranteeing that simultaneous reservations will be booked successfully. In addition, several components used at Worldspan are absent in our current testbed. They include back-end services like seat availability, ticket purchasing, and pricing rule updating. These, as well as internal components like data caching and load segmentation, can be added to expand the scope and depth of the simulation. Its composability

with other services at both the front- and back-ends is what makes WorldTravel suitable to act as a SOA testbed.

A variety of useful experiments can be built on top of WorldTravel. As a whole, WorldTravel can be used for research into the complexities of designing and managing SOA systems, such as SLA management and enforcement, service discovery and equivalency, etc. The load data provided with WorldTravel has interesting properties that are well suited to experiments with load distribution and autonomic management [32]. Finally, the services provided by WorldTravel can be composed with other services, and are suitable for research into dynamic composition, as is done by Yahoo! Pipes, for example.

WorldTravel is built with open source technologies, to enable changes and extensions to its implementation. This includes Apache Geronimo, JMS, MySQL, etc. Its transparent design encourages modification and integration with other tools and applications, such as monitoring infrastructures used to conduct experiments or enable system performance tracking and management.

In order to serve as a testbed for a broad range of SOA implementations, WorldTravel uses a generic architecture and refrains from tightly integrating with any specific platforms, tools, or standards. Specifically, we have not adopted web service standards (WS-*); the system can be integrated with these standards as validation of their applicability and utility, but does not rely on them, thereby making WorldTravel suitable for evaluating future standards, as well. A few specific standards used in WorldTravel include: XML, to encode requests and responses, which will enable future extensions that analyze and modify the semantic information contained within; SOAP, as a communications protocol; and WSDL, to identify service endpoints. Each of these standards were chosen to maximize utility and simplify the implementation of WorldTravel, but none are essential to its operation; each can be substituted in order to evaluate alternative standards or to integrate with other tools and applications which use different standards.

Perhaps most importantly, WorldTravel is distributed with a large data set generously provided by Worldspan. This includes anonymized pricing data that 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 also provide us with additional data representative of typical update feeds. Finally, Worldspan has also provided request traces, which are used to generate realistic traffic patterns on the travel website and the GDS.

7 Experimental Evaluation

In addition to describing how WorldTravel is designed to meet the abstract criteria for an effective testbed, we must also demonstrate that it represents some of the complexities of commercial applications, making it useful as a SOA testbed. This task is complicated by the fact that some behaviors do not manifest themselves without additional software and architecture, such as caching, load segmentation, and more advanced query processing. However, the data used with WorldTravel has intrinsic properties, and we can demonstrate that the WorldTravel implementation displays behaviors that relate to these properties. This

Table 1. Number of fares filed per market

Origin airport	Destination airport	Number of fares
HNL (Honolulu, HI)	SEA (Seattle, WA)	4338
HNL (Honolulu, HI)	PDX (Portland, OR)	3996
OGG (Kahului, HI)	SEA (Seattle, WA)	3840
OGG (Kahului, HI)	PDX (Portland, OR)	3782
HNL (Honolulu, HI)	SAN (San Diego, CA)	3607
OGG (Kahului, HI)	SAN (San Diego, CA)	3540
KOA (Kailua-Kona, HI)	SEA (Seattle, WA)	3538
⋮		
RNO (Reno, NV)	WRL (Worland, WY)	1
SJU (San Juan, PR)	YQK (Kenora, ON, Canada)	1
STS (Santa Rosa, CA)	TUP (Tupelo, MS)	1
STS (Santa Rosa, CA)	TYS (Knoxville, TN)	1
STS (Santa Rosa, CA)	VPS (Eglin AFB, FL)	1
STT (St. Thomas, USVI)	YQK (Kenora, ON, Canada)	1
YEV (Inuvik, NWT, Canada)	YOQ (Ottowa, ON, Canada)	1

makes WorldTravel suitable for testing solutions pertaining to client request–driven applications with varying response costs and complexities.

One such property is that not all markets (that is, pairs of origin and destination airports) have the same number of fares filed. Airlines may file short-lived fares in small markets as a way of enacting discount sales.[3] They may also file many fares in a single market to implement complex pricing rules. A quick examination of the database, shown in Table 1, shows that this distribution of fares among markets exhibits a strong geographical bias; the markets with the most fares in the database are all Hawaiian airports, while the markets with the fewest fares are mostly small airports.

We construct a simple test in which we measure the time taken by a query node to process a request, indexed by the number of matching fares available in the database. We augment WorldTravel to report the amount of time spent handling a request in the query processor. For the experiment, we randomly choose markets to request, testing both markets which have few airlines filing fares (typically 1 to 3 airlines) and which have many airlines filing fares (increasing as the number of fares increases, up to 7). After an initial request to ensure that the data has been cached by the database, we make several requests and calculate the average processing time. Table 2 on the next page shows the results of this experiment.

The results demonstrate a correlation between the number of fares filed in the requested market and the time required to process a request. Since there

[3] This is possible even when there is no direct flight between the two airports; pricing data and flight data are strictly separate. A single fare may be used to price multiple flights with connections, while a single itinerary may be priced as a combination of fares representing one or more connections.

Table 2. Time taken to process a request versus market

(a) Markets with the most airlines publishing fares

Market	Number of fares	Avg. time to process(ms)
AVL–LEX	50	40.8
GEG–IND	100	42.8
GSO–YTO	200	47.3
BOI–SJU	305	51.0
HNL–MEM	402	53.2
EWR–OGG	518	58.5
LIH–PDX	1891	105.2

(b) Markets with the fewest airlines publishing fares

Market	Number of fares	Avg. time to process(ms)
TWF–VLD	50	40.0
SJU–SLK	100	40.5
MBS–OGG	200	43.3
OGG–VPS	302	47.7
BTR–OGG	405	49.7
JHM–PDX	506	52.8
ITO–PDX	1748	99.3

is also a correlation between market geography and number of fares filed, we can conclude that the processing time for a request is affected by the market of the request. This could be exploited to provide different processing methods for requests that are expected to require more processing time based on their geography. Indeed, Worldspan sees this effect on a much greater scale: queries for international markets have up to four times as much data, and their processing times are significantly higher. In their production system, therefore, Worldspan separates international queries from domestic queries and uses faster machines to process international queries in order to avoid service timeouts.

The simple experiment discussed above demonstrates that the WorldTravel testbed exhibits behavior much like Worldspan's application. This constitutes initial evidence that the testbed is suitable for testing and analyzing techniques and solutions of use to to Worldspan and other real-world systems.

8 Conclusions and Future Work

This paper presents the WorldTravel testbed for SOA architecture investigations and for constructing and experimenting with SOA applications. The paper defines "service-oriented architecture" and its distinction from "web services", and it shows that the WorldTravel testbed can act both as a clean model for SOA and as a common ground for promoting and understanding future research and applications in this domain. The WorldTravel simulation system is based on an actual class of commercial applications in the airline travel industry, and it is constructed to exemplify the characteristics of SOA. An analysis of the current testbed demonstrates that it meets key SOA requirements.

Because we have simplified Worldspan's architecture in order to realize an easily distributed and extensible system, there are many opportunities for extending it. They include both those used in commercial settings like at Worldspan and new creative front- or back-end services that expand the testbed with new features and additional functionality. Also missing for this initial implementation is an evaluation of its composability with existing SOA-based services, includ-

ing those that are publicly available. We are currently undertaking this task. Further, we will also 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 the testbed. Finally, we will explore how the various standards for managing SOAs (e.g., web service standards, Tuscany, etc.) can be applied to the testbed.

Acknowledgments

We are most grateful to Sameh Abdelaziz and his team at Worldspan for their partnership and cooperation in allowing us access to their systems, and for providing us with documentation about the airline industry, sample data, and request traces that capture the complexity of the problems occurring in realistic enterprise applications. The authors would also like to thank Richard Bailey for his help in proofreading the paper.

References

1. Cecchet, E., Marguerite, J., Zwaenepoel, W.: Performance and scalability of EJB applications. In: Proceedings of the 17th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, pp. 246–261 (2002)
2. Rice University (RUBiS), `http://rubis.objectweb.org/`
3. Rotem-Gal-Oz, A.: What is SOA anyway? `http://rgoarchitects.com`
4. Kodali, R.R.: What is service-oriented architecture? JavaWorld.com (June 2005)
5. Bustamante, F., Eisenhauer, G., Schwan, K., Widener, P.: Efficient wire formats for high performance computing. In: Supercomputing, ACM/IEEE 2000 Conference, p. 39 (2000)
6. Chiu, K., Devadithya, T., Lu, W., Slominski, A.: A binary XML for scientific applications. In: E-SCIENCE 2005: Proceedings of the First International Conference on e-Science and Grid Computing, pp. 336–343. IEEE Computer Society, Washington (2005)
7. Seshasayee, B., Schwan, K., Widener, P.: SOAP-binQ: High-performance SOAP with continuous quality management. In: Distributed Computing Systems, Proceedings. 24th International Conference, pp. 158–165 (2004)
8. Nelson, B.J.: Remote procedure call. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA (1981)
9. Sun Microsystems (Remote method invocation), `http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp`
10. Fielding, R.T.: Representational state transfer (REST). In: Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine (2000)
11. Clerkin, P., Cunningham, P., Hayes, C.: Ontology discovery for the semnatic web using hierarchical clustering (2001)
12. Sycara, K., Paolucci, M., Ankolekar, A., Srinivasan, N.: Automated discovery, interaction and composition of Semantic Web services. In: Web Semantics: Science, Services and Agents on the World Wide Web, pp. 27–46 (2003)

13. Zhang, L., Ardagna, D.: SLA based profit optimization in autonomic computing systems. In: ICSOC 2004: Proceedings of the 2nd international conference on Service oriented computing, pp. 173–182. ACM, New York (2004)
14. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web services agreement specification (WS-Agreement)
15. Dean, J., Ghemawat, S.: MapReduce: Simplified data processing on large clusters. In: OSDI 2004: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation, p. 10. USENIX Association, Berkeley (2004)
16. He, H.: What is service-oriented architecture. XML.com (September 2003)
17. Shannon, C., Moore, D., Keys, K., Fomenkov, M., Huffaker, B., Claffy, K.: The internet measurement data catalog. SIGCOMM Comput. Commun. Rev. 35(5), 97–100 (2005)
18. Sripanidkulchai, K., Maggs, B., Zhang, H.: An analysis of live streaming workloads on the internet. In: IMC 2004: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, pp. 41–54. ACM, New York (2004)
19. Sun Microsystems (Java adventure builder reference application), `https://adventurebuilder.dev.java.net/`
20. Apache Software Foundation (Nutch), `http://lucene.apache.org/nutch/`
21. Apache Software Foundation (Hadoop), `http://hadoop.apache.org/core/`
22. Ennals, R.J., Garofalakis, M.N.: MashMaker: Mashups for the masses. In: SIGMOD 2007: Proceedings of the 2007 ACM SIGMOD international conference on Management of data, pp. 1116–1118. ACM, New York (2007)
23. Yahoo! (Yahoo! Pipes), `http://pipes.yahoo.com/`
24. Apache Software Foundation (Apache Tuscany), `http://incubator.apache.org/tuscany/`
25. Li, M.L., Sasanka, R., Adve, S., Chen, Y.K., Debes, E.: The alpbench benchmark suite for complex multimedia applications. Iiswc 0, 34–45 (2005)
26. Kumar, V., Cai, Z., Cooper, B.F., Eisenhauer, G., Schwan, K., Mansour, M., Seshasayee, B., Widener, P.: Implementing diverse messaging models with self-managing properties using IFLOW. In: IEEE International Conference on Autonomic Computing, ICAC 2006, pp. 243–252 (2006)
27. Mosberger, D., Jin, T.: Httperf—a tool for measuring web server performance. SIGMETRICS Perform. Eval. Rev. 26(3), 31–37 (1998)
28. Mansour, M., Wolf, M., Schwan, K.: StreamGen: A workload generation tool for distributed information flow applications. In: ICPP 2004: Proceedings of the 2004 International Conference on Parallel Processing, pp. 55–62. IEEE Computer Society, Washington (2004)
29. Mansour, M., Schwan, K., Abdelaziz, S.: I-Queue: Smart queues for service management. In: ICSOC 2004: Proceedings of the 2nd international conference on Service oriented computing, pp. 252–263. ACM, New York (2006)
30. Kumar, V., Cooper, B.F., Cai, Z., Eisenhauer, G., Schwan, K.: Resource-aware distributed stream management using dynamic overlays. In: ICDCS 2005: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems, pp. 783–792. IEEE Computer Society, Washington (2005)
31. IBM (WebSphere MQ), `http://www-306.ibm.com/software/integration/wmq/`
32. Kumar, V., Schwan, K., Iyer, S., Chen, Y., Sahai, A.: The state-space approach to SLA-based management. In: IEEE/IFIP Network Operation & Management Symposium, NOMS (2008)