Coarse-Grained Web Service Availability, Consistency, & Durability

Aspen Olmsted
Department of Computer Science
College of Charleston, Charleston, SC 29401

Csilla Farkas
Department of Computer Science and Engineering
University of South Carolina, Columbia, SC 29208

***Abstract*— In this paper we investigate the problem of providing consistency, availability and durability for Web Service-transactions. We consider each transaction as a black box, with only the corresponding metadata, expressed as UML specifications, as transaction semantics. We refer to these WS transactions as coarse-grained WS transactions. We propose an approach that guarantees the availability of the popular lazy replica update propagation method while increasing the durability and consistency. In this paper we extend our previous work, called the Buddy System, to handle course grained WS transactions, we are using UML stereotypes that allow scheduling semantics from the design model to support: 1.) High availability by distributing service requests across all available clusters. 2.) Consistency by performing the complete transaction on a single set of clusters. 3.) Durability by updating two clusters synchronously.**

Keywords-web services; distributed database; modeling

I. INTRODUCTION

In this work, we extend our previous results [1] that improved the lazy replica update propagation method to reduce the risk of data loss. The Buddy System executes a transaction on a primary replica like lazy-replication. However, the transaction cannot commit until a secondary replica, "the buddy", also preserves the effects of the transaction. The rest of the replicas are updated using one of the standard lazy update propagation protocols. This algorithm provides a guarantee of transactional durability (i.e., effects of the transaction are preserved even if the server, hosting the primary replica crashes before the update can be propagated to the other replicas) and efficient update propagation (i.e., our approach requires the synchronized update between two replicas only, therefore adding minimal overhead to the lazy-replication protocol).

However, the Buddy System uses an application-layer dispatcher to select the buddies based on the data items and the operations of the transactions, the data versions available, and the network characteristics of the WS farm. A limitation of the Buddy System is that the services provided were simple CRUD (Create, Read, Update, and Delete) operations. Web Services are normally coarse-grained functions with input and output that varies based on the function of the web service. In this paper we propose to use UML stereotypes to represent transactional semantics. This will allow the dispatcher to use the resource consumption semantics to distribute requests to several clusters concurrently.

Our solution provides several advantages not addressed in traditional distributed database replica update protocols. First, our approach provides the scalability required by modern n-tier applications, such as web farms, and is suitable for the architectures and technologies implementing these applications in cloud computing environments. Second, the buddy-selection algorithm supports dynamic master-slave site selection for data items and ensures correct transaction execution. Third, we show that our method can be easily extended to incorporate network specific characteristics, such as distance and bandwidth, that further reduce the latency observed by the client and to provide load-balancing among the replicas. Our empirical results support our hypothesis that in the presence of large data sets, the efficiency of our approach is comparable to the efficiency of the lazy update propagation method while also ensuring the integrity of the data.

II. EXAMPLE TRANSACTION

We demonstrate our work using a Ticket Reservation System (TRS). TRS uses web services to provide a variety of functionalities to the clients. We use the UML specification to represent the meta-data. Figure 1 shows an activity diagram for an implementation of this functionality. The Unified Modeling Language includes a set of graphic notation techniques to create visual models of object-oriented software systems [3]. The XML Metadata Interchange (XMI) is a standard used by XML to serialize the visual diagrams into an XML format that can be consumed by an application program. The following web services are used in Figure 1: GetSession, LoginAnonymous, GetZones, GetSeats, GetSeatState, GetPerformanceDetails, ReserveSeats. Our aim is to handle unknown load at deployment time. That is, during normal operations an organization may only have a few concurrent requests. When a popular event goes on sale, this number could rise to tens of thousands of requests.

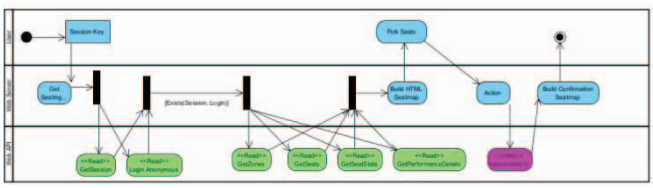

Figure 1 Activity Diagram for Self Service Seat Selection

978-0-7695-5025-1/13 $26.00 © 2013 IEEE
DOI 10.1109/ICWS.2013.91

III. BUDDY SYSTEM

In our previous work [4, 1], we proposed the *Buddy System,* that used pairs of clusters to synchronously update all transactions. The pairs of buddies are allocated for each request, allowing increased availability by fully using all server resources available. Consistency is increased over lazy-replication because all transactional elements are updated in the same cluster allowing for transaction time referential integrity and atomicity. To support the above components, an intelligent dispatcher was placed in front of all clusters. The dispatcher operated at the OSI Network level 7. This allowed the dispatcher to use application specific data for transaction distribution and buddy selection.

Coarse-grained web services are essentially distributed functions where the only information the dispatcher has at runtime is the input and output parameters of the web service. For the dispatcher to schedule the coarse-grained web services properly it needs to map the coarse-grained service to a limited set of operations on the atomic data item level.

IV. PRELIMINARIES

Semantics for the coarse-grained web services can be modeled as UML Activity and Class diagrams. A stereotype is one of three types of extensibility mechanisms in the UML that allows a designer to extend the vocabulary of UML in order to represent new model elements [3].

READ vs WRITE SEMANTICS: Figure 1 is an activity diagram with two stereotypes used to model web services that are read-only and web services that write and update data as part of the process. In the example the ReserveSeats services modifies data as part of its process and all other services just read data as part of their process.

ELEMENT UNIQUE IDENTIFIER SEMANTICS: Each Web Service in the Activity diagram has a matching UML Class diagram that shows the structure of the input and output messages. An attribute level stereotype <<PK>> is used to represent the unique identifier combination of the attributes.

PARALLEL SCHEDULING SEMANTICS: The UML Activity diagram (*Figure 1*) also provides us with the semantics required to know which services can be called in parallel. The getSession and loginAnonymous services are required to be called before the remaining services as they change required state used by the later service.

V. BUDDY SYSTEM FOR COARSE-GRAINED SERVICES

The *Dispatcher Service Request Algorithm* needs visibility into all operations of the transaction at a single point in time. To facilitate this visibility for coarse-grained services, the client sends all requests as a batch and the dispatcher sequences the calls based on the semantics from the UML.

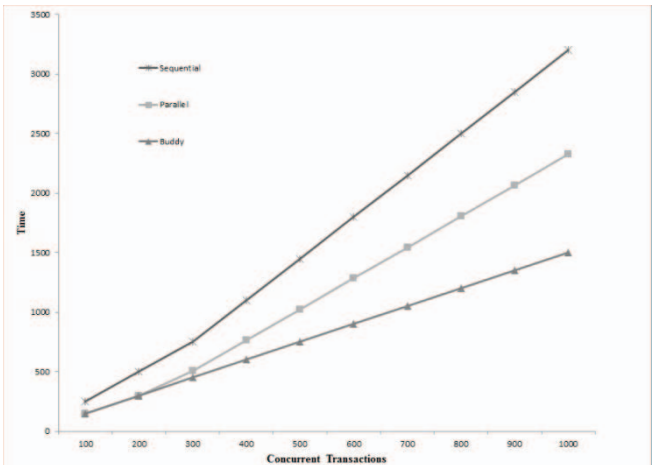

Figure 2 Empirical Results

A. Buddy Selection Algorithm

The algorithm iterates over the forks in the activity diagram to service the items that can be done in parallel. A fork is a point in the activity diagram where the flow is split and can run in parallel. The algorithm then determines eligible buddies that can service the batch of web service requests and randomly chooses two to do so.

Theorem 1: The Buddy Algorithm guarantees one-copy serializability.

VI. EMPIRICAL RESULTS

Figure 2 shows the performance results of the implementation where the additional semantics gained from the UML data allows the buddy system to almost double the availability of the original sequential schedule.

VII. CONCLUSION

In this paper we propose an extension to the buddy system to handle coarse-grained web services. Our solution is based on extending UML with stereotypes to embed CRUD, Parallel and data element semantics into the model. Each individual transaction is applied to a pair of clusters synchronously allowing enforcement of consistency guarantees and durability while increasing availability.

VIII. REFERENCES

[1] A. Olmsted and C. Farkas, "The cost of increased transactional correctness and durability in distributed databases," in *13th International Conference on Information Reuse and*, Las Vegas, NV, 2012.

[2] Object Management Group, "Unified Modeling Language: Superstructure," 05 02 2007. [Online]. Available: http://www.omg.org/spec/UML/2.1.1/. [Accessed 08 01 2013].

[3] A. Olmsted and C. Farkas, "High Volume Web Service Resource Consumption," in *Internet Technology and Secured Transactions, 2012. ICITST 2012*, London, UK, 2012.