Performance of a Collaborative Framework for Federating Distributed Digital Entities

Ahmet Fatih Mustacoglu
TUBITAK-National Research Institute of Electronics and Cryptology (UEKAE), Marmara Research Center, Gebze, Kocaeli, 41470 TURKEY
ahmet.fatih@uekae.tubitak.gov.tr

Geoffrey C. Fox
Community Grids Lab, Indiana University, Bloomington, IN, 47404, USA
School of Informatics and Computing, Indiana University, Bloomington, IN, 47405, USA
gcf@indiana.edu

ABSTRACT

We investigate the performance and the scalability metrics of the Event-based Infrastructure and consistency model of the IDIOM (Internet Documentation and Integration of Metadata) framework that is for federating online digital entities. The IDIOM consists of tools and services for supporting Cyberinfrastructure based scientific research. This system supports a number of existing online Web 2.0 research tools (social bookmarking, academic search, scientific databases, journal and conference content management systems) and aims to develop added-value community building tools that leverage the management and the federation of digital entities and their metadata obtained from multiple services. We provide the performance and the scalability experiment results and conclude with a discussion of further research opportunities in this proposed research.

KEYWORDS: Collaboration, Web 2.0, Annotation Tools, Distributed Digital Entities, Federation.

1. INTRODUCTION

One of the major challenges that people are facing with is to remember and access information that they have found earlier and thought could be useful for them later. Probably the most common approach to re-finding information on the web is to use personal bookmarks provided by several web browsers. For instance, Mozilla Firefox browser supports the creation of collections of URLs. Furthermore, URLs can be annotated by using keywords or free-form text. These collections can also be sorted based on a various things such as keyword, last visited, location or time. People create bookmarks depend on their personal interests in the information and quality of the resource, possibility of future use, current necessities as explained in [1].

Another challenge is to find and share information that is spread all over the Web in various locations including centralized repositories, web servers and user desktops. Centralized repositories represent the old fashion techniques for resource sharing, whereas completely decentralized systems such as P2P systems allow users to share information without depending on a third party repository. The necessities to find and share information led to development of emergent Web 2.0 applications. These new Web 2.0 applications such as social bookmarking tools introduce a new way of sharing information rather than the old fashion and P2P systems do. Social bookmarking tools address the challenging problems of finding and sharing information among small groups, teams and communities. Various types of social bookmarking tools developed their own systems to support different kind of resources. Flickr [2], for example, allows the tagging and the sharing of photos, del.icio.us [3] the tagging and the sharing of bookmarks, BibSonomy [4], CiteULike [5] and Connotea [6] the tagging and the sharing of scholarly publications, YouTube [7] the tagging and the sharing of video, and 43Things [8] the tagging and the sharing of goals in private life.

There are several common features for social bookmarking systems. First of all, these tools provide their users with ability to create personal bookmarks and share them with other users instantly. Data is stored centrally in these social bookmarking tools and it is available from any computer that is connected to the internet. Second, these systems enable entering personal keywords called tags explicitly by the user for each bookmark. Using tags for the resources allows users to organize and display their collections in a meaningful way. Furthermore, assigning multiple keywords for a bookmark makes it belong to multiple categories. The

final common feature of social bookmarking tools is the social way of their use. The collection of bookmarks created by users is also visible to other users. For instance, when a user name is clicked on, then the collection of bookmarks for that user is viewable to other users. Similar transparency is also valid for tags. So, one can retrieve similar resources that fall into same interest of other users by clicking on an interested tag.

As the web-based social bookmarking services have gained popularity, an emerging need has appeared for methodologies to retrieve, represent, share and manage information that are stored in these annotation tools for scholarly publications. As these services enable storing, tagging and sharing documents, another emerging need has also appeared for supporting these tools by using their existing services via Web Service wrappers with added capabilities. In our Internet Documentation and Integration of Metadata (IDIOM) framework [9-14], we are able to manage, share and reconcile scholarly publications that are stored in several social bookmarking tools in a Service Oriented Architecture where communications are provided through the Web Service technology [15]. The IDIOM software is available from [16]. Figure 1 illustrates the architecture of the IDIOM system.

This paper investigates the performance and the scalability results of the event-based infrastructure and consistency maintenance parts of the IDIOM system [9-14] that federates online digital entities. The rest of this paper is organized as follows: Section 2 provides background information on event-based systems and consistency mechanisms. Section 3 provides a discussion of the underlying event-based and the consistency structure of the IDIOM framework. Section 4 presents the performance and the scalability test results for the IDIOM system. Finally, Section 5 summarizes the work and describes further research opportunities.

2. BACKGROUND

We overview the event systems and the consistency maintenance issues for distributed systems that are crucial for the IDIOM framework in the following sub-sections respectively.

2.1. Event Systems

In recent years, there has been an increasing amount of research focused on event based systems. Their main objective is to notify the necessary entities about the changes that occurred in the domain of interest. Today, event systems are needed and used in several areas such as graphical user interfaces, databases, web based applications, networking applications, distributed applications, publish-subscribe paradigm etc. Several tools have been developed for each of these areas to satisfy their needs, and NaradaBrokering [17-21], which is an open-source messaging infrastructure, is an example for publish-subscribe paradigm. NaradaBrokering was developed in Community Grids Lab at Indiana University [22]. There are two different approaches to the event definition. The first approach defines an event as it is an instantaneous atomic occurrence, so it is represented as a point in time [23-25]. Based on this approach, timestamps of event occurrences can be categorized in three different ways:

- Absolute time point: It consists of date and time
- Relative time points: It is defined relative to a particular position
- Virtual Clocks are explained in detail in [26], and unique timestamp values are assigned automatically to each event by the system.

The second approach defines an event as occurrence as an interval in time [27-30]. Based on this approach, a state change of an event can be specified within a specific interval and the interval can be represented in two ways:

- As relative, absolute, or virtual time points represent starting and ending point of an interval
- Event occurrences that represent the initial and ending points of an interval

So, first approach defines events as having no duration while the second approach defines events by having them particular duration. Most of the previous event system related works use the first approach in their event-based modeling and design.

2.2. Consistency Maintenance

Consistency is an important issue in distributed systems. Consistency means that all copies of a same document meant to be the same. When one copy is updated, then it must be ensured that all copies are updated as well [31].

According to [31], consistency models can be classified into two groups: (a) Data-Centric Consistency Models; (b) Client-Centric Consistency Models. Details about these two models are given in the following sections respectively.

2.2.1. Data-Centric Consistency Models

A consistency model is an agreement between processes and hosting environment, where data is stored. As long as processes obey the rules, the hosting environment promises to work correctly. A process that executes a read operation on a data item expects to get a value that is a result of the last write operation on the data item. However, in the absence of a global clock, it is difficult to

say which write operation is the last one. So to maintain consistency in different ways, there are other data-centric consistency model definitions. Each data-centric consistency model has different restrictions on what a read operation can return on a data item. It is easy to implement and use consistency models with minor restrictions whereas it requires lots of effort to use consistency models with major restrictions. But the gain is different in each model since the one with major restrictions provide better results than the one with minor restrictions do [31]. More information on consistency models can be found in [32, 33]. Tanenbaum classifies data-centric consistency models into seven sub-categories: (a) Strict Consistency; (b) Linearizability and Sequential Consistency; (c) Casual Consistency; (d) FIFO Consistency; (e) Weak Consistency; (f) Release Consistency; and (g) Entry Consistency [31].

2.2.2. Client-Centric Consistency Models

In the previous section, we have overviewed and summarized data-centric consistency models that are all about providing a system wide consistent view on a shared data. On the other hand, client-centric consistency models ensure the consistent view of data from a client's perspective. They allow copies of a data to be inconsistent with each other as long as the consistency is maintained from a single client's point of view. Tanenbaum classifies client-centric consistency models into five sub-categories: (a) Eventual Consistency; (b) Monotonic Reads; (c) Monotonic Writes; (d) Read Your Writes; and (e) Writes Follow Reads.

3. THE EVENT-BASED INFRASTRUCURE AND CONSISTENCY MODEL OF THE IDIOM FRAMEWORK

We describe the novel event-based infrastructure and consistency maintenance of the IDIOM framework that is designed to provide an ideal approach to unify and federate major annotation tools, support collaboration, represent and manage content of scientific documents coming from various sources in a flexible fashion. The event-based infrastructure and consistency maintenance constitute the core underlying architectural concepts of the IDIOM framework. Both concepts are briefly overviewed in this section but the detailed information can be found in [9].

The content of scientific documents is represented as a Digital Entity (DE) in the IDIOM framework. Another word is that a DE consists of several metadata fields that hold the related metadata of a scholarly publication. In the Event-based Infrastructure of the IDIOM framework, data and metadata coming from various sources are represented as events. The initial metadata of a scientific document is represented by a major event, while further updates to any metadata field of an existing DE are represented by minor events. Both major and minor events, which form DE and contain the metadata fields of the document, are stored into a MySQL system database [9]. General architectural design for the Event-based Infrastructure (EBI) and Consistency Framework for Distributed Annotation Records (CFDAR) of the IDIOM system appears in Figure 1.

The consistency maintenance issue has to do with ensuring that all copies of the same data to be the same at a given time. Some approaches to maintain consistency are discussed in detail in [31, 34-38]. Tanenbaum [31] differentiates consistency under two main categories: (1) data-centric; and (2) client-centric. In data-centric approach, all copies of data are updated whether some clients is aware of those updates or not. In client-centric approach, consistency is maintained from a client's perspective. Client-centric consistency model allows copies of data to be inconsistent with each other as long as the consistency is ensured from a single client's point of view. The implementation of the consistency models can be categorized as primary-based protocols (primary-copy approach) and replicated-write protocols [31]. In primary copy approach, updates are executed on a single location, and propagated replicas from there, while in the replicated-write approach; updates can be originated from multiple locations.

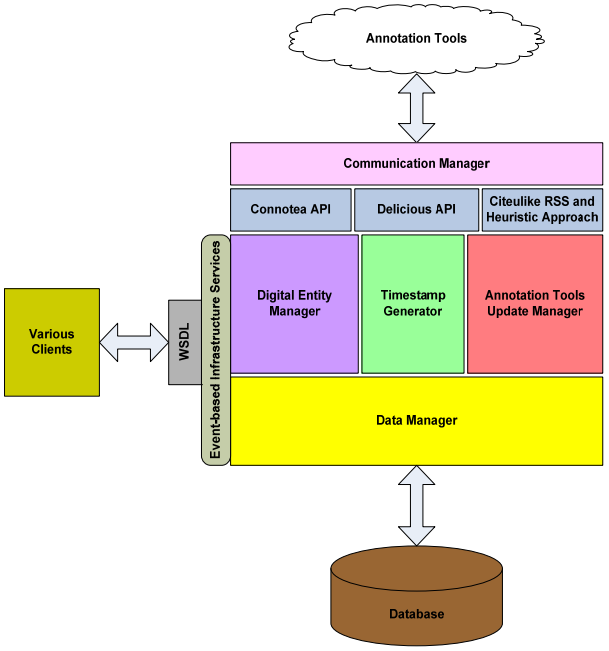

Figure 1. The Architectural Design for the Event-based Infrastructure and Consistency Framework for Distributed Annotation Records

Our proposed framework CFDAR supports collaboration among DARs, which are replicas of the same document, kept at various web-based annotation tools. CFDAR adopts optimistic replication approach to ensure eventual consistency between annotation tools and the system database. An overview of the proposed architecture design appears in Figure 1.

4. THE PERFORMANCE AND THE SCALABILITY EXPERIMENTS RESULTS

We performed extensive series of measurements to evaluate the prototype implementation of the proposed architecture and investigate its practical usefulness in real life applications.

4.1. Testing Environment

We tested our Event-based Infrastructure and Consistency Framework implementation by using gf12-15 and gf16 Linux machines that are part of a cluster located at Community Grids Laboratory at Indiana University [22]. We have run our client programs on gf12-gf15 Linux machines, we have deployed our service-based Event-based Infrastructure and Consistency Framework system on gf16 Linux machine, and we have installed our database on gf16 Linux machine. Summary of these machine configurations are given in Table 1.

In our general experiments methodology, we have used single-threaded and multi-threaded client programs. Our Event-based Infrastructure and Consistency Framework is also a multi-threaded service-enabled system running on cluster node gf16.ucs.indiana.edu. We have sent various requests from the client programs to our proposed system implementation to test the performance, and the scalability of our proposed system.

Table 1. Summary of Cluster Nodes

	Cluster Nodes	
	gf12-15.ucs.indiana.edu	**gf16.ucs.indiana.edu**
Processor	Intel® Xeon™ CPU (E5345 2.33GHz)	Intel® Xeon™ CPU (E5345 2.33GHz)
RAM	8 GB (each node)	8 GB Total
OS	GNU/Linux (kernel release 2.6.9-5.ELsmp)	GNU/Linux (kernel release 2.6.9-5.ELsmp

We have implemented our service-enabled Event-based Infrastructure and Consistency Framework in Java Language, using Java 2 Standard Edition compiler with version 1.5.0_12. In our experiments with the prototype implementation, we used Apache Tomcat Server as a container with version 5.0.28 and Apache Axis technology for Web Service technology with version 1.2. We set the maximum heap size of Java Virtual Machine (JVM) to1024MB by using the option –Xmx1024m. In our experiments, we also increased the maximum number of threads from default value to 1000 in Apache Tomcat Server to be able to test the system behavior for the huge numbers of concurrent clients.

4.2. System Responsiveness Experiments

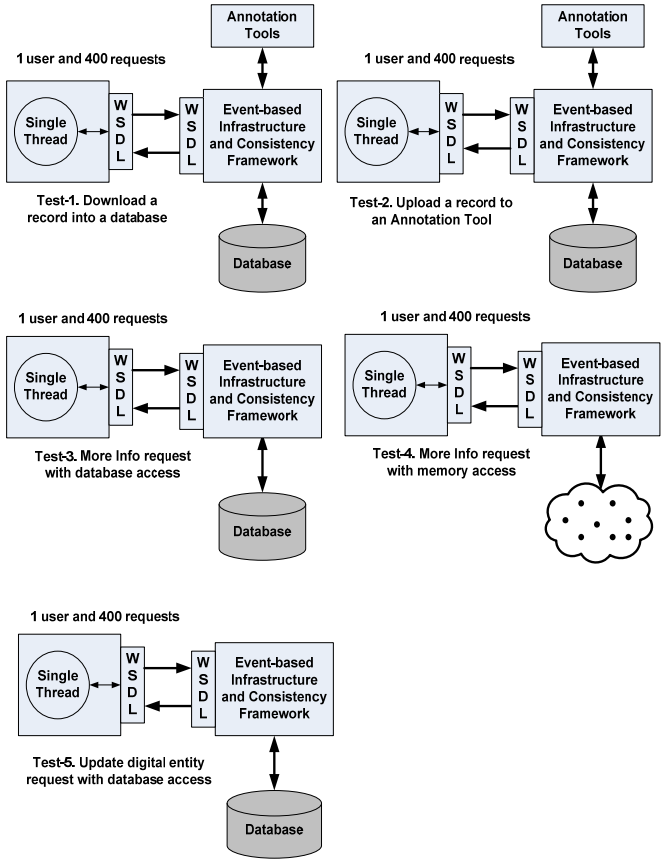

Figure 2. Testing Cases for System Responsiveness Experiment

Our main goal in doing this experiment is to measure the baseline performance of our Event-based Infrastructure and Consistency Framework implementation. We have tested the performance of our proposed system by measuring the times necessary to download a record from an annotation tool into a repository, and to upload a new record from a repository to an annotation tool (forms a DAR). Furthermore, we have investigated latency values for More Info functionality with DB access and memory utilization, and Update DE functionality. The performance evaluation is done when there is no additional traffic in the system. The primary interest for doing system responsiveness experiment was to investigate the optimum performance of the system for download, upload, more info and update digital entity primary operations for the proposed system. The client

programs were running on a cluster nodes gf12-gf15, while service-enabled Event-based Infrastructure and Consistency mechanism was running on a cluster node gf16. In this experiment, we were exploring the performance of our methodology for download, upload, more info and update digital entity operations of the proposed system. We have conducted the following test cases: a) A single client sends a request to download a DAR from an annotation tool as a major event required to access to the DB; b) A single client sends a request to make a new DAR required to access to an annotation tool; c) A single client sends a request to get a more info on a digital entity from a repository required to access to the DB; d) A single client sends a request to get a more info on a digital entity from the cache required to access to the memory; and e) A single client sends a request to update a digital entity existed in a repository. In our each testing case, the clients send 400 sequential requests for download, upload, more info and update digital entity standard operations. We recorded the average execution time, and this experiment was repeated 5 times. Figure 2 shows the design of these experiments.

4.2.1. System Responsiveness Experiment Results

We conduct experiments where we investigate the base performance of the proposed system. Depicted in Figure 3, Figure 4, and listed in Table 2, Table 3 represents basic responsiveness result of our system. In this experiment we first recorded execution times for: a) calling the download service to measure the processing time of our implemented service; b) calling the upload service to measure the processing time of our implemented service. Next, we recorded round trip times for: a) calling the More Info service with database access to measure the latency of our implemented service; b) calling More Info service with memory utilization to measure the latency of our implemented service; c) calling Update DE service to measure the latency of our implemented service. Downloading a new entry requires to store this entry as a major event in the database and it is one of the major services provided by our Event-based Infrastructure and Consistency Framework system. Furthermore, our Event-based Infrastructure and Consistency Framework system propagates the updates via push mechanism by using upload service of the system in order to maintain consistency. This experiment shows the necessary time requirements for these major services to download or to upload a digital entity between the database and annotation tools (replicas).

4.3. Scalability Experiment

The primary interest in doing this experiment was to investigate the scalability of Event-based Infrastructure and Consistency Framework implementation. We conducted three testing cases and tried to answer the following research questions: a) how well does the system performs when the message rate per second is increased for More Info standard operation request on a DE with DB access?; b) how well does the system performs when the message rate per second is increased for More Info standard operation request on a DE with memory utilization?; c) how well does the system performs when the message rate per second is increased for Update DE standard operation request?

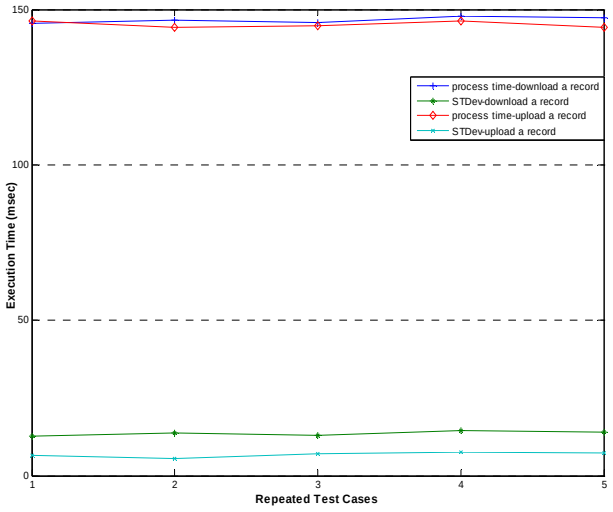

Figure 3. Download and Upload a Record

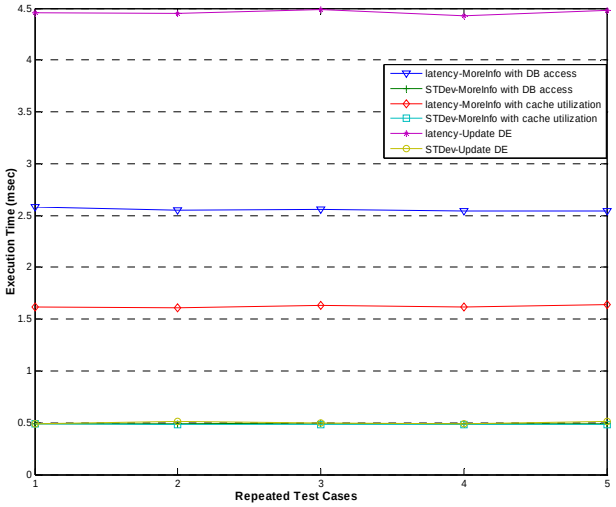

Figure 4. Latency and STDev Values for Update DE and More Info Standard Operation (with DB and Memory Utilization)

In first experiment, our main goal is to identify the number of concurrent requests requiring DB access that can be handled by the proposed system when message rate per second are increased in the Event-based Infrastructure and Consistency Framework. We have completed this test case by increasing the message rate/sec until the response time degrades. In this testing

case, we recorded round trip time at each MoreInfo request on a DE with DB access. In the second testing case, we have applied the same technique as previous experiment except that each request is responded by using memory utilization. In the third experiment, we have investigated the concurrent requests for an Update DE main operation that can be serviced by the Event-based Infrastructure and Consistency Framework while message rate per second are increased. The designs of these testing cases are depicted in Figure 5.

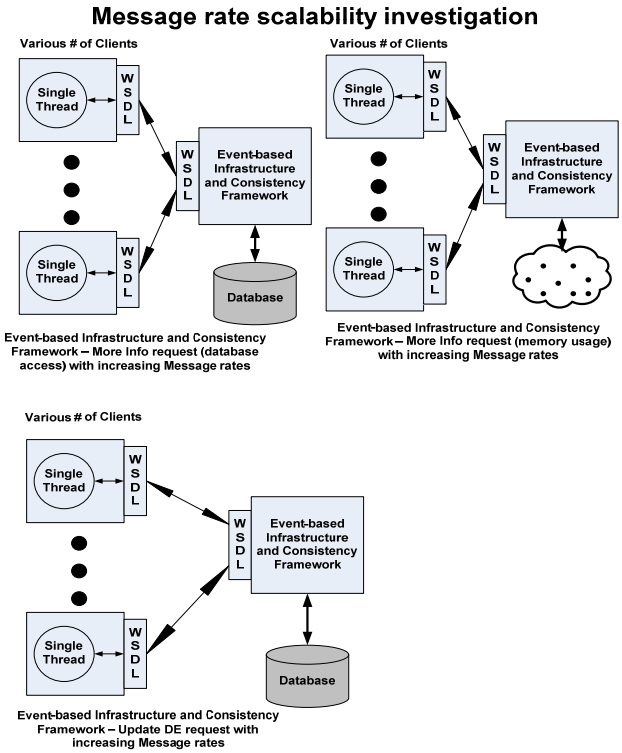

Figure 5. Testing Cases of Scalability Experiment for More Info and Update DE Requests

4.3.1. Scalability Experiment Results

Based on the results depicted in Figure 6, we determined that concurrent inquiry requests may be well responded by Event-based Infrastructure and Consistency Framework without any error. According to the experiment result, we identified that Event-Based Infrastructure and Consistency Framework's major operations performed well for the increased message rate. However, after a certain number of messages per second, performance starts to degrade due to high message rate. We observe that after around 1060 inquiry messages per second for More Info with DB access, after around 2068 inquiry messages per second for More Info with memory utilization, after around 533 inquiry messages per second for Update DE, the system performance degrades due to high message rate. This threshold is mainly due to Apache Tomcat (thread scheduling and context switches). Experiment results are depicted in Figure 6.

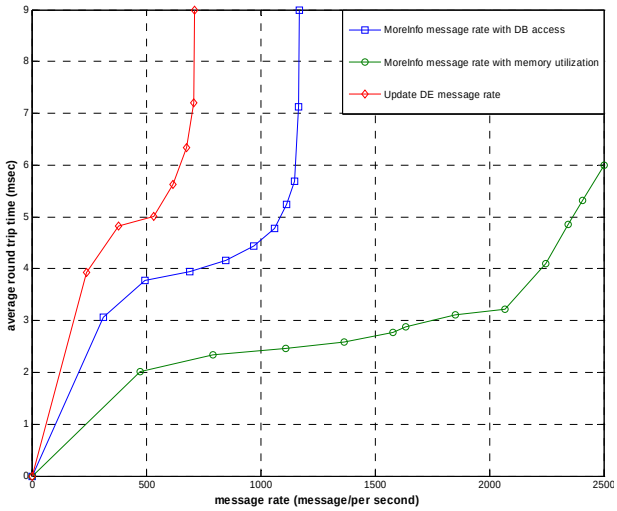

Figure 6. Update DE and MoreInfo Message Rate with DB and Memory Access

Table 2. Statistics of the Experiment Depicted in Figure 3

Repeated Test Cases	1	2	3	4	5
Download Process time (msec)	145.44	146.49	145.72	147.77	147.37
Download STDev	12.74	13.64	13.09	14.54	13.94
Upload Process time (msec)	146.24	144.23	144.75	146.33	144.3
Upload STDev	6.61	5.52	7.11	7.6	7.24

Table 3. Statistics of the Experiment Depicted in Figure

Repeated Test Cases	1	2	3	4	5
Latency-MoreInfo with DB access	2.58	2.55	2.56	2.54	2.54
STDev-MoreInfo with DB access	0.49	0.49	0.50	0.49	0.49
Latency-MoreInfo with cache utilization	1.62	1.61	1.63	1.62	1.64
STDev-MoreInfo with cache utilization	0.49	0.48	0.48	0.48	0.48
Latency-Update DE	4.46	4.45	4.49	4.43	4.48
STDev-Update DE	0.49	0.51	0.50	0.49	0.51

5. FUTURE WORK

The IDIOM framework deploys an Event-based Infrastructure and adopts a consistency technique for distributed systems to maintain consistency among distributed annotation records and their primary copies stored at a central repository. It introduces an Event-based Infrastructure and utilizes optimistic replication approach to ensure eventual consistency between distributed annotation records representing scholarly publications. We plan to expand on this approach to be able to apply other application domains such as video collaboration domain (YouTube etc.) and social networking domain (Facebook etc.). We will further research machine

learning techniques to identify typing errors within the documents. An additional area that we intend to research is to migrate from centralized structure to decentralized structure.

6. CONCLUSION

In this paper we presented the performance and the scalability experiment results of our proposed IDIOM framework. We have also mentioned the event-based infrastructure and consistency model of the IDIOM system briefly. Furthermore, we described the architecture components of the IDIOM and outlined some directions for future work.

REFERENCES

[1] A. David, B. Ron, and C. Mark, "Information archiving with bookmarks: personal Web space construction and organization," in *SIGCHI conference on Human factors in computing systems*, Los Angeles, California, United States, 1998.

[2] Flickr web site. http://www.flickr.com/

[3] Delicious web site. http://de.icio.us

[4] Bibsonomy web site. http://www.bibsonomy.org

[5] CiteULike web site. http://www.citeulike.org

[6] Connotea web site. http://www.connotea.org

[7] YouTube web site. http://www.youtube.com/

[8] 43things web site. http://www.43things.com/

[9] A. F. Mustacoglu, "Event-Based Infrastructure for Reconciling Distributed Annotation Records," PhD Thesis in *Computer Science*, 2008, p-184, Indiana University: Bloomington, IN.

[10] A. F. Mustacoglu and G. Fox, "Hybrid Consistency Framework for Distributed Annotation Records in a Collaborative Environment " in *The 2008 International Symposium on Collaborative Technologies and Systems (CTS 2008)* Irvine,CA: IEEE Computer Society, ACM, 2008, pp. 267-274.

[11] A. E. Topcu, A. F. Mustacoglu, G. Fox, and A. Cami, "Integration of Collaborative Information Systems in Web 2.0," in *3rd International Conference on Semantics, Knowledge and Grid* vol. 0 Xian,China: IEEE Computer Society, 2007, p. 523.

[12] A. F. Mustacoglu, A. E. Topcu, A. Cami, and G. Fox, "A Novel Event-Based Consistency Model for Supporting Collaborative Cyberinfrastructure Based Scientific Research," in *Collaborative Technologies and Systems CTS 2007 in Technical Cooperation with The IEEE Computer Society* Orlando, FL, USA: IEEE Computer Society, 2007.

[13] G. Fox, A. F. Mustacoglu, A. E. Topcu, and A. Cami, "SRG: A Digital Document-Enhanced Service Oriented Research Grid," in *Information Reuse and Integration IRI-07* Las Vegas, NV, USA: IEEE Computer Society, 2007, pp. 61-66.

[14] G. C. Fox, M. E. Pierce, A. F. Mustacoglu, and A. E. Topcu, "Web 2.0 for E-Science Environments," in *3rd International Conference on Semantics, Knowledge and Grid*. vol. 0 Xian,China: IEEE Computer Society, 2007, p. 1.

[15] H. Kreger, "Web Services Conceptual Architecture (WSCA 1.0)," 2001. Available from: http://www.cs.uoi.gr/~pvassil/courses/diplomatikes/miscellaneous/WebServicesConceptualArchitecture.pdf

[16] "Internet Documentation and Integration of Metadata (IDIOM) Project web site. http://gf16.ucs.indiana.edu:54571/IDIOM/login.jsp

[17] G. Fox and S. Pallickara, "Deploying the NaradaBrokering Substrate in Aiding Efficient Web and Grid Service Interactions," *Grid Computing*, vol. 93, pp. 564-577, 2005.

[18] S. Pallickara, M. Pierce, H. Gadgil, G. Fox, Y. Yan, and H. Yi, "A Framework for Secure End-to-End Delivery of Messages in Publish/Subscribe Systems," vol. 0, p. 215, 2006.

[19] S. Pallickara and G. Fox, "NaradaBrokering: A Distributed Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids," *Middleware 2003*, pp. 998-999, 2003.

[20] G. Fox, S. Pallickara, and X. Rao, "A scaleable event infrastructure for peer to peer grids," in *the 2002 joint ACM-ISCOPE conference on Java Grande*, Seattle, Washington, USA, 2002.

[21] S. Pallickara, H. Bulut, P. Burnap, G. Fox, A. Uyar, and D. Walker, "Support for High Performance Real-time Collaboration within the NaradaBrokering Substrate," 2005.

[22] Community Grids Lab at Indiana University web site. Available from: http://communitygrids.iu.edu/index.php

[23] S. Gatziu, *Events in an active, object-oriented database system*. Hamburg: Verlag Dr. Kovac, 1995.

[24] K. R. Dittrich and S. Gatziu, "Time Issues in Active Database Systems," in *International Workshop on an Infrastructure for Temporal Databases*, Arlington, Texas, 1993.

[25] G. Liu, A. Mok, and P. Konana, "A Unified Approach for Specifying Timing Constraints and Composite Events in Active Real-Time Database Systems." vol. 00, 1998, p. 199.

[26] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Commun. ACM*, vol. 21, pp. 558-565, 1978.

[27] J. F. Allen and G. Ferguson, "Actions and Events in Interval Temporal Logic," *Journal of Logic and Computation*, vol. 4, pp. 531-579, 1994.

[28] P.-s. Kam and A. W.-c. Fu, "Discovering temporal patterns for interval-based events," *Lecture Notes in Computer Science*, vol. 1874/2000, pp. 317-326, 2000 2000.

[29] C. Liebig, M. Cilia, and A. Buchmann, "Event Composition in Time-Dependent Distributed Systems." vol. 00, 1999, p. 70.

[30] Peter R., Pietzuch R., Shand B., and B. J., "A Framework for Event Composition in Distributed Systems," in *4th International Conference on Middleware (MW'03)* Rio de Janeiro, Brazil: Springer, 2003, pp. 62-82.

[31] A. S. Tanenbaum and M. V. Steen, *Distributed Ssytems Principles and Paradigms*, 2002.

[32] S. V. Adve and K. Gharachorloo, "Shared Memory Consistency Models: A Tutorial." vol. 29, 1996, pp. 66-76.

[33] D. Mosberger, "Memory consistency models," *SIGOPS Oper. Syst. Rev.*, vol. 27, pp. 18-26, 1993.

[34] S. Chengzheng and C. David, "Consistency maintenance in real-time collaborative graphics editing systems," *ACM Trans. Comput.-Hum. Interact.*, vol. 9, pp. 1-41, 2002.

[35] L. Jiang, L. Xiaotao, S. Prashant, and R. Krithi, "Consistency Maintenance In Peer-to-Peer File Sharing Networks," in *Proceedings of the The Third IEEE Workshop on Internet Applications*: IEEE Computer Society, 2003.

[36] R. Jonathan, F. Sarah, and V. Sankar, "Consistency management for distributed collaboration," *ACM Comput. Surv.*, vol. 31, p. 13, 1999.

[37] V. Jurgen, V. JiRgen, G. Werner, C. Li-Te, and M. Michael, "Consistency Control for Synchronous and Asynchronous Collaboration Based on Shared Objects and Activities," *Comput. Supported Coop. Work*, vol. 13, pp. 573-602, 2004.

[38] G. Werner, V. Jurgen, C. Li-Te, and M. Michael, "Supporting activity-centric collaboration through peer-to-peer shared objects," in *Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work* Sanibel Island, Florida, USA: ACM Press, 2003.