IDEA: An Infrastructure for Detection-based Adaptive Consistency Control in Replicated Services

Yijun Lu Ying Lu Hong Jiang
Dept. of Computer Science and Engineering, University of Nebraska-Lincoln, Lincoln, NE
{yijlu, ylu, jiang}@cse.unl.edu

ABSTRACT

In Internet-scale distributed and replicated services, poor consistency results in poor QoS or even monetary loss. Recent research focuses on enforcing a certain consistency level, instead of perfect consistency, to strike a balance between consistency guarantee and system's scalability. In this paper, we argue that it is equally, if not more, important to achieve adaptability. *I.e.*, the system adjusts its consistency level on the fly to suit applications' ongoing need. This paper presents IDEA (an Infrastructure for DEtection-based Adaptive consistency control), a protocol that adaptively controls consistency in replicated services by detecting inconsistency among nodes in a timely manner via an inconsistency detection framework and resolving the detected inconsistencies efficiently when necessary. Through experimentation on Planet-Lab, IDEA is evaluated from two aspects: its adaptive interface and its performance of inconsistency resolution. Results show that IDEA achieves adaptability by adjusting the consistency level according to users' preference on-demand, and it achieves low inconsistency resolution delay and incurs minimal communication cost.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; C.4 [**Performance of Systems**]: Reliability, Availability, and Serviceability; H.3.5 [**Online Information Services**]: Data Sharing

General Terms

Performance, Design, Experimentation, Human Factors

Keywords

Inconsistency Detection, Consistency Control, Adaptability, Replicated Services

1. INTRODUCTION

Replicating data and services is an attractive strategy to increase availability and performance in distributed systems; in an Internet-scale system, such as large-scale Grid, replication-based schemes may indeed be the only way to provide continuous service and prevent data loss in the presence of unreliable Internet connections. Within this context, consistency control among replicas has become an important research area because poor consistency in replication-based systems results in poor QoS or even monetary loss (in e-business applications). Realizing that a large collection of applications, such as e-business, are willing to sacrifice a certain degree of consistency in order to scale their services, recent research has concentrated on striking a balance between consistency guarantee and system's scalability by enforcing a certain level of, rather than perfect, consistency.

In this paper, we argue that it is equally, if not more, important to achieve adaptability in consistency control. Here, adaptability means that the system should be able to adjust its consistency level on the fly, as opposed to a predefined consistency level. This is important because multiple applications with different consistency requirements can run simultaneously in a modern distributed computer system and even one application's consistency requirement can change from time to time.

To this end, we present IDEA (an Infrastructure for DEtection-based Adaptive consistency control) that achieves this adaptability. Besides adaptability, IDEA achieves high performance as well by detecting inconsistencies among participants in a timely manner via an efficient Inconsistency Detection mechanism proposed in [1, 2] by the authors, and resolving the detected inconsistencies efficiently when necessary.

HPDC'07, June 25–29, 2007, Monterey, California, USA.
ACM 978-1-59593-673-8/07/0006.

2. DESIGN OF IDEA

IDEA is assumed to work with a general distributed file system that handles the ordinary read/write operations. IDEA is deployed in the middleware level and applications on different nodes consult IDEA when they access files.

IDEA utilizes a two-layer (top/bottom layer) infrastructure to detect and resolve inconsistency for each shared file or object. The top layer for a given file includes nodes that update this file sufficiently frequently and hence is where most inconsistencies come from. The remaining nodes form the bottom layer.

2.1 A Scenario of Using IDEA

We illustrate a scenario in which IDEA is used. First, a user predefines or hints on his or her acceptable consistency level, denoted as L_1, when initiating an application. Upon a read or write request, IDEA retrieves a copy of the file from the underlying replication-based system and returns it to the application. At the same time, IDEA derives a consistency level for the returned replica and then checks whether the consistency level is acceptable based on either the user's predefined tolerance level or the interaction with the user. If the consistency level is acceptable, IDEA does nothing; otherwise, it resolves the inconsistency. IDEA also adapts the predefined hint level on the fly. When a user is not satisfied with the result based on consistency level of L_1, IDEA will increase the consistency level by Δ and $L_1 + \Delta$ will then become the new desired consistency level for the user. IDEA will keep the application's consistency above this new level to avoid annoying the user again. Users can communicate with IDEA about why the current consistency level is not sufficient.

2.2 The IDEA Protocol

The workflow of the IDEA protocol is depicted in Figure 1. From the figure, we can see that the IDEA protocol is triggered by two operations: write and certain read operations. The write operation, such as issuing an update in a white board, triggers the IDEA protocol because it will surely cause inconsistency among replicas. For read operations, IDEA is triggered when a reader tries to retrieve a new file (such as a new snapshot of a white board) because, in this case, the user needs to make sure that the file retrieved is sufficiently consistent for the user's purpose. For other reads, IDEA is triggered according to the context: if the file is locally updated frequently, the read will not trigger IDEA; if the file hasn't been locally updated for a long time and the user is afraid that the file may be inconsistent, IDEA will be triggered.

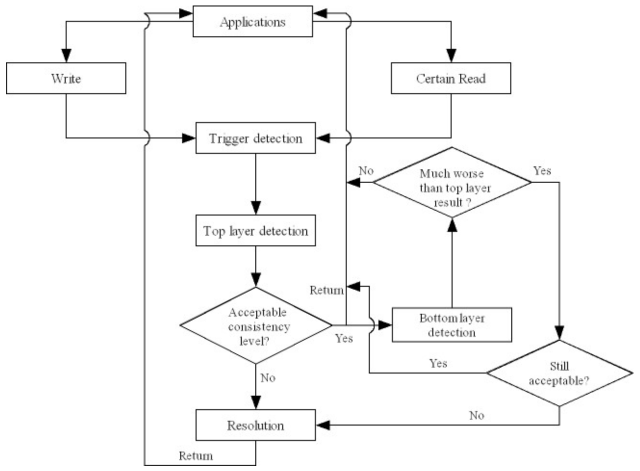

Figure 1. Workflow of the IDEA protocol

After IDEA is triggered, it will use a detection-based mechanism to check the consistency level, represented by a single percentage number, such as 90%. This number is obtained appropriately either from interpreting users' view of QoS or by (analysis of) the nature of an application; the mechanism to properly quantify this parameter is omitted here due to space limit and can be found in [3]. After the inconsistency level is returned, IDEA checks whether the consistency level is acceptable either based on the user's predefined tolerance level or through the interaction with the user. If the consistency level is acceptable, IDEA does nothing; otherwise, it resolves the inconsistency. IDEA will also learn from previous interactions and adapt the acceptable consistency levels to prevent annoying users again.

To improve responsiveness, the calculation of the consistency level is based on detection among the top-layer nodes only. This value may not be accurate because the nodes in the bottom layer can cause inconsistencies too, albeit rather infrequently. We deploy a rollback mechanism, through which IDEA lets users continue their work when the initially returned consistency level (from top layer nodes) is acceptable. In the background, though, IDEA continues to detect inconsistency in the bottom layer and returns a new value afterward. If the new value is not sufficiently close to the previous one obtained from the top layer, IDEA alerts the user about the discrepancy and resolves the inconsistency if the users demand so.

2.3 Adaptive Consistency Control

We realize that different applications have different meanings of adaptability and the scenario we described in Section 2.1 cannot possibly support all of them. Here we list three possible application types that can benefit from IDEA. Other schemes certainly exist and we plan to explore them in the future. (1) **On-demand**. In this scheme, users explicitly request inconsistency resolution when they are not satisfied with the current consistency level. Otherwise, they depend on the periodic background consistency resolution. (2) **Hint-based**. This scheme asks users to give hints about their approximate consistency requirements. IDEA only triggers the consistency resolution when the consistency level drops below that hinted by the user. (3) **Fully automatic**. This scheme improves consistency with best effort where background resolution is invoked as frequently as possible within a certain overhead level.

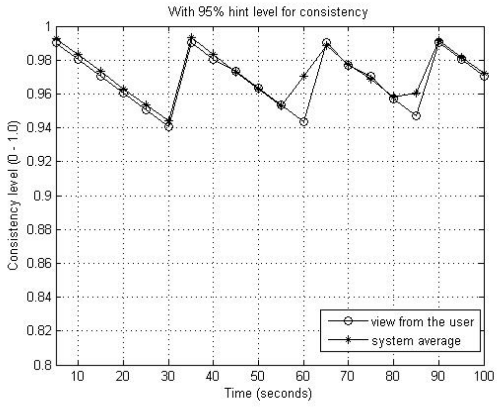

Figure 2. Adaptive consistency control with 95% hint level

3. EVALUATION

We emulate a distributed white board application on Planet-Lab test-bed. In this application, we assume that users indicate the consistency tolerance levels (the *hints*) to be 95%. As shown in Figure 2, the consistency level is improved right after IDEA kicks in and evokes the inconsistency resolution scheme. With further investigation, we find that IDEA brings the consistency level back in less than one second with up to ten simultaneous writers and this speed can be further improved with simple parallelism [3]. IDEA also incurs minimal communication cost even for dial-up connections [3].

4. REFERENCES

[1] Lu, Y., and Jiang, H. A framework for efficient inconsistency detection in a grid and Internet-scale distributed environment, In *Proc. of HPDC-14*. Research Triangle Park, NC, July 2005, pp. 318-319.

[2] Lu, Y., Jiang, H., and Feng, D. An efficient, low-cost inconsistency detection framework for data and service sharing in an Internet-scale system. In *Proc. of IEEE ICEBE 2005*, Beijing, China, Oct. 2005. pp. 373-380.

[3] Lu, Y., Lu, Y., and Jiang, H. IDEA: An infrastructure for detection-based adaptive consistency control in replicated services, *Technical Report TR-UNL-CSE-2007-0001,* University of Nebraska-Lincoln, Jan. 2007.