How to Ensure Reasonable Serializability for Web Caches

Jinho Ahn

Dept. of Comp. Sci., Kyonggi Univ., Iuidong, Yeongtong,
Suwon 443-760 Gyeonggi, Republic of Korea
jhahn@kgu.ac.kr

Abstract. Many Internet-based services require not a few of data updates on databases and caches, incurring high update latency to their clients. In this paper, we introduce a cache coherent scheme to adjust the tradeoff between scalability and consistency according to the system requirements. The scheme is capable of ensuring the data consistency within a cache cluster while incurring the system performance degradation much more lightweight than the transactional consistency applied on the database servers. It can significantly lower the possibility web clients read staled data from incoherent cache servers.

Keywords: In-memory Caches, Consistency, Scalability, Serializability.

1 Introduction

Most of Internet-based services like e-Bay, Alibaba and Amazon require not a few of data updates on databases and caches like in figure 1 [6]. However, if every data update operation should be done on the back-end database, large-scale web-based systems cannot accept the performance penalty resulting from this limitation [1, 2, 4]. On the other hand, in order to maximize scalability of the system, it may be designed to be very vulnerable to the staleness of cached data on reading them, making the usage of the system unattractive to web users and raising big loss of earnings. In this paper, we introduce a cache coherent scheme to adjust the tradeoff between scalability and consistency according to the system requirements. The scheme is capable of ensuring the data consistency within a cache cluster while incurring the system performance degradation much more lightweight than the transactional consistency applied on the database servers. It can significantly lower the possibility web clients read staled data from incoherent cache servers.

2 The Cluster-Wide Serializability Guaranteed Scheme

The two-layered cache server architecture is divided into a group of cache clusters like in figure 2. Each cluster consists of a number of cache servers and is controlled by a cluster coordinator elected among them, which manages a directory having the information about which cache server holds on its cache which data items or shards. The set of cluster
coordinators form a virtual ring to effectively locate a particular item or shard on read request by using consistent hashing \cite{3, 5}. The proposed scheme is performed based on the architecture stated above to ensure the cluster-wide serializability.

![Diagram](image)

**Fig. 1.** General architecture for updating the requested data between web servers and back-end DBs via cache servers.

When a web client $c$ attempts to update a particular data item or shard in cache server $p$, $p$ retrieves and invalidates an element for it from its own cache. Then, $p$ should transmit the update request to its local cluster coordinator $\alpha$. As the request arrives at the coordinator from cache server $q$, $\alpha$ searches for the list of cache servers within the cluster holding the data on their respective caches using its own directory. Then, it forces every live cache server on the list to update its cache accordingly. After it has received all the update completions from them, $\alpha$ informs server $q$ of the cluster-wide update completion. When $q$ receives the completion from $\alpha$, it updates the element for the data on its own cache and notifies the client of the update operation completion. The detailed algorithms for cache servers and their cluster coordinators are given in figures 3 and 4.

![Diagram](image)

**Fig. 2.** Two-layered cache server architecture.
Fig. 3. Cached data update procedure for cache server $p$.

Fig. 4. Cached data update procedure for cluster coordinator $\alpha$ with server $q$. 
3 Conclusions

This paper proposed a cache coherent scheme to be capable of ensuring the data consistency within a cache cluster while incurring the system performance degradation much more lightweight than the transactional consistency applied on the database servers. The probability the staleness of cached data is shown to web servers may be very low. If the scheme is combined with the epidemic style multicasting for asynchronously updating data on cache servers outside the local cluster, it may support higher level of transactional consistency virtually close to full atomicity without giving up the high cache read performance.

Acknowledgments. This work was supported by the Gyeonggi Regional Research Center (GRRC) and Contents Convergence Software (CCS) research center in Korea (Project No.: GRRC Kyonggi 2015-B04).

References