Replicate and Bundle: A Mechanism for Increasing Efficiency and Scalability of Data Centers

Amruta T. Paul
Computer Science and Engineering
Priyadarshini Institute of Engineering andTechnology
Nagpur, India

Prof. Sonali Bodkhe
Computer science and engineering
Priyadarshini Institute of Engineering andTechnology
Nagpur, India

***Abstract*— This paper addresses the issue of maximizing the efficiency and scalability of RAM- Based storage systems where in multiple objects must be retrieved per user request. The focus is on per server transaction, not per requested item. By introducing RnB, a innovative mechanism to minimize the number of servers accessed per user request, it increases the scalability and efficiency of RAM-Based storage systems.**

In this paper, We present "Replicate and Bundle" (RnB), a method for reducing the number of transactions required to process an end user request. This method enables increasing the maximum system throughput without adding CPUs. RnB entails 1) data replication and 2) bundling of items requested from the same server into a single transaction. We use a pseudo-random object-to-server mapping for each object's different replicas, placing the replicas on different servers for each object. During data fetch, we choose which replica to access in order to reduce the number of servers that need to be accessed for any given request. Finding a minimal set of servers is the well known minimum set cover problem, which is NP-complete . Therefore, we use heuristic low complexity approaches. Considerable benefits are obtained even with sub-optimal server selection.

For increasing the efficiency of system, LRU based caching system is used. For bundling Ranged consistent hashing RCH, which allows selecting, for each item stored, a group of servers that has the copy of the requested item.

Keywords: LRU, consistent hashing, replication n bundling, efficiency, scalability

I. INTRODUCTION

A. Background

In this work, we consider the scalability and efficiency of RAM-based read-mostly [6] storage caching systems in Web data centers (e.g., Facebook, Twitter, Gmail). In these data centers, a large number of web servers, nearly stateless present behind the load balancer. These stateless web servers are present to run the application code for the web. The original copies of the (read-mostly) data for the application are present in the large disk based databases such as MySQL, MS-SQL, Oracle, Cassandra, etc. As the database access is slow, a special caching layer is present between the web servers and the database of the application. Memcached is a RAM based key-value storage/caching service. This caching service uses the simple network access protocol. The Memcached server stores the recent database access results. Here the Memcached servers are served as RAM based storage not as the caches.

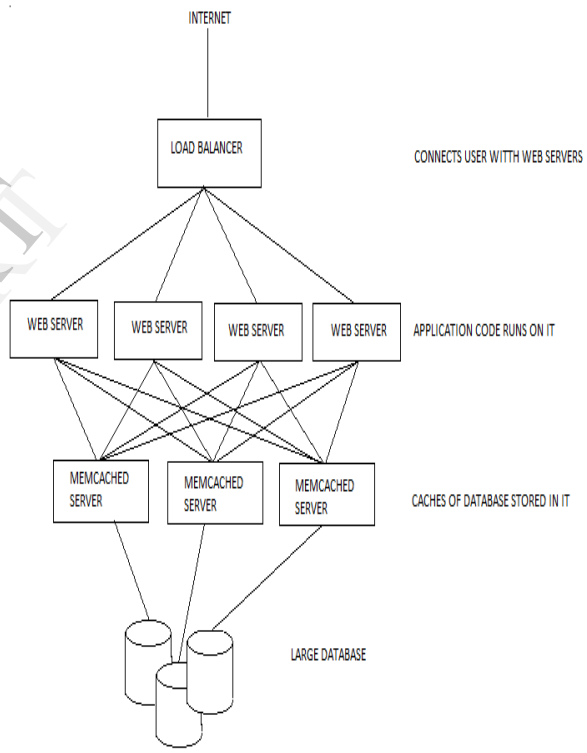

Fig. 1. TYPICAL WEB APPLICATION

For increasing performance and scalability, it is needed to identify the servers which are storing the copies of the requested items. This identification usually, done without communication. Therefore, Memcached servers use consistent hashing to store the items to the servers. As a result, in an *N*-server system, a client request for *M* specific items will require sending requests to $N(1-(1-1/N)^M)$ servers on an average[1]. When there is a request for the items from the servers and the request set is larger than the available number of servers the that request needs to access almost all servers to fulfill the user request, so adding servers commensurately increases the number of transactions per user

request. If the maximum amount of server CPU work is per server transaction not the per server item, then this provides no relief to the server CPU's. this goes into the problem of Multi-Get Hole [2].

B. Terminologies used

Every transaction in our assumption is the collection of requested items from the client which we called as the request set.

The end user sends the set of requested item/ items to the web service. The request size is the number of items in the request. The web server gets the request from the end user, these web servers are called as the clients. The client translates the request into number of transactions, which contains the requested items list. These transactions are for the access of items from the database, is send to the different Memcached servers. If the item is not found in the Memcached server it will be accessed from the database, and the copy is written back to the relevant Memcached server.

Finally, we define several metrics used in this work :

•**Transactions Per Request (TPR)** – the average number of transactions that are needed to fulfill the single user request.

• **Transactions Per Request Per Server (TPRPS)** – The average number of transaction for single user request divided by the number of servers available in the system.

•**Maximum System Throughput ("Throughput")** – the maximum time required for completing the transactions for the entire system.

•**TPRPS Scaling Factor** – It is the ratio of Transactions Per Request Per Server between two systems.

•**Throughput Scaling Factor** – It is the throughput ratio between two systems.

For reader convenience, we provide here definitions of terms that are used in a later part of this work:

•**Overbooking** – providing less physical memory than implied by the declared number of replicas.

C. Contribution of replicate and bundle

Replicate and Bundle is a mechanism to reduce the number of transactions that are required to satisfy the end user request. It increases the throughput of the entire system by reducing the server access for satisfying the number of requests from the number of clients or end users. It will give maximum throughput without adding CPUs to the system. RnB entails: 1) data Replication: Replication enables you to have identical copies of information on multiple servers and across more than one data center and 2) Bundling of requested items. In data replication the data which are not available in the Memcached servers are duplicated into it, for increasing the efficiency of the system. In Bundling the requested items from the same server are combined in single transaction.

For placing the replicas on different servers for each object, we use pseudo-random object-to-server mapping for each object's different replicas. Whenever the we need to access the data, from multiple replicas, we choose which replica to access, so that the number of servers accessed should be minimum, for any given request. Finding a minimal set of servers is the well known minimum set cover problem, which is NP-complete [3]. We can get the Considerable benefits even with sub-optimal server selection, but we use we use heuristic low complexity approaches.

The actual data is always available in the databases like in MySql etc. The database access is very slow, so we need the cache layer in between the application program and the database which are called as the memcached servers. But which we need the data to be replicated on these memcached servers for availability. So we need the mechanism which replicates the data on an appropriate server. But again the problem is which replica to select from which sever so that the number of transactions required to fulfill the end user request. For that here we are using the Bundling mechanism in combination with replication.

RnB is a distributed, stateless algorithm. The algorithm requires almost same amount of configuration information as required in consistent hashing and it does not require any additional communication for this. RnB can be beneficially applied to other similar workloads like in the social network data sets, as our results are in the context of it. RnB can be nearly free as the data are replicated in systems for other reasons like for availability, fault tolerance, etc.

With this basic RnB we are going to use the extensions as LRU for reducing the number of replicas on Memcached servers. So that the time required for accessing the data gets reduced. If the data in servers will be less it will require minimum time to search the requested object. Another approach is the extension of consistent hashing, which we call as the Ranged consistent hashing (RCH). This extension allows, for requested items, finding the set of severs, that have the copies of it. While achieving a balanced and uniform distribution of the replicas, The approach preserves the good attributes of consistent hashing.

II. REPLICATE AND BUNDLE (RNB)

A. The RnB Solution

Replication: Replication is making copies of available data from database to the Memcached servers. Each requested item is written to a preconfigured set of servers. The unit for our system is the number of transaction per user request.

To reduce the number of transactions on server the data is replicated on multiple caching servers. So that access speed of requested item will get reduced. For this consistent hashing is used.

For replication multiple hashing techniques can be used. The further studies can include comparative study of multiple hashing which can reduce the time required for replication.

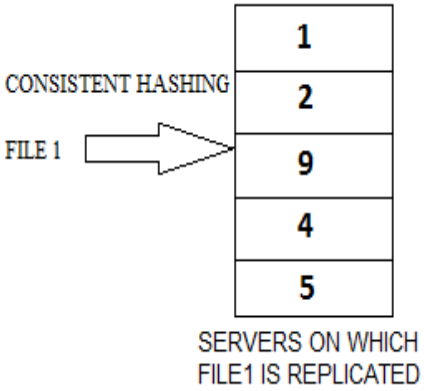

Fig. 2. Each item is written to a preconfigured set of servers, chosen using consistent hashing.

Bundling: In this, the locations of all the data item's replicas are calculated and set of servers which jointly possess all the replicas of the requested items are computed. Whenever there is a request for the data access from the web server the request doesn't go to the database directly, it is first found in the memcached servers. Bundling finds the set of servers which are jointly possessing the replicas of the requested items, from these replicas which replica to access is found. The problem of finding the minimum group is NP-complete [7].

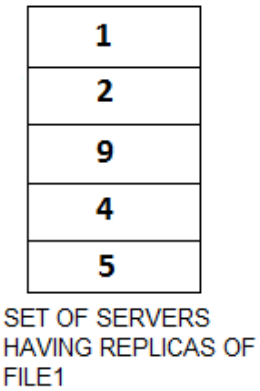

Fig. 3. A group of servers that jointly possess all requested items is computed.

Here for evaluation we assumed that system can handles each user request individually and bundles only items in the same request.

B. Enhancements to RnB

There are some enhancements that we are going to implement in this are as follows:

Distinguished copy overbooking:

This enhancement is aimed at exploiting the fact that there are some replicas on the Memcached servers which are used very less frequently in the access of data. These copies are called as "cold" copies of the data.

Our mechanism combines the properties of Memcached servers and a property of the replica selection algorithm. Each Memcached servers keeps a local Least Recently Used (LRU) list of the items which are stored on the different Memcached server. From this list it selects the items which are used very less frequently and drops these unused copies from the server when running out of space. The result is that the number of replicas of the object and the locations of these replicas are determined in fully distributed manner and implicitly. But there must be at least one copy of the object replicated on any of the Memcached server, we mark one of the copy from multiple replica as the distinguished copy. This can be done by selecting one of the one of the hash functions as the "distinguished" hash function.

The algorithm which we are using for selecting the servers to satisfy a request is the greedy set cover algorithm. See figure4.This algorithm is has the property as, if there are two requests which require same item sets, the replicas which are used to fulfill the request are mostly the same for maximum request. This property allows to "automatically" benefit from the spatial locality in the requests. This will make some of the replicas of each object as the "cold" copies. The local LRUs on the memcached servers will drop these cold replicas.

Consider the figure, if there are two requests, consider request 1- for the file 1,file 2 and file 3 and request 2- for for the file 1,file 2 and file 4. The figure gives the possible placements of data items on multiple servers. Here both the requests will fetch the data from server A, even though there is a virtual copy of F1 is present on server C and F2 is present on B. so maximum access of F1 and F2 will be done from server A and the copies on server B and C are not used at all, so the servers will eventually discard the replicas through their LRU mechanism.

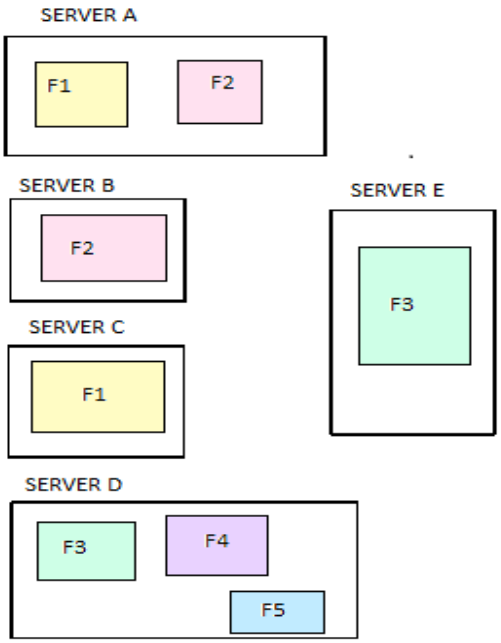

Fig. 4. An example of request locality reducing the needed memory.

C. Merging Multiple Requests

Several ([9], [10]) real world implementations of Memcached support merging multiple end user requests, thereby reducing the number of transactions performed with the servers.

The simulator is designed to collect a predefined number of requests, and these requests are handled as a single request. The consecutive requests are combined which is called as bundling of item into a single transaction. This will reduce the number of transactions per user request.

D. When RnB is not Effective

If the request is for write data: RnB cannot satisfy the write requests, as it will require updating multiple replicas of the items on different servers.

If the request is for individual data i.e. for single item per request: here if the request are for single data item at a time the basic RnB can do nothing for it.

If the request set is very large: here if the request sets are large then it performs like per item access not as the per transaction access, so RnB cannot help here much.

III. IMPLEMENTATION OF RNB

In this paper we have defined the main elements that are required for the main implementation of the RnB in the Memcached servers. We have implemented the paper partially; the full implementation is not yet done. So the results are not present in the paper. Some elements which we have defined and implemented partially are given below:

Basic RnB : The main requirement here is the replication of data on multiple servers, which is already implemented in most of the systems for reliability and availability of the system. The Bundling of the multiple requests is done, thereby reducing number of transactions per user request.

Ranged consistent hashing : It is an enhancement to the basic consistent hashing. This extension allows, for requested items, finding the set of severs, that have the copies of it. While achieving a balanced and uniform distribution of the replicas, The approach preserves the good attributes of consistent hashing. It improves the runtime efficiency for finding the set servers having copies of requested item set.

Atomic operations are supported : with the use of LRU in local Memcached servers keeping only the distinguished copies of the item, and discarding the cold copies is done automatically in the system. If the copies are present in the server then loads them on demand.

IV. RELATED WORK

FAWN [14] is a distributed key–value storage system with a memcached interface, this work focuses on power efficiency. In [11] it is compared with disk based systems. It makes no use of redundancy.

CRAQ [12] uses redundant copies of the data. That is it avails multiple copies of the data on multiple servers as replication. This allows better read performance. But the work in it uses for single-item requests. Bundling of multiple items is not proposed in this work.

The replication and bundling concept, similar to the RnB is also used in [8], for storage system. The goal of this is improving system performance by accessing the data faster as it is available on multiple servers and the requests are combined. However, the focus in [8] is on data arrangement within a disk to reduce seek work.

For the RAM based storage, Ongaro et al.[13] consider replication. The focus of their work is on fault recovery. As such, it assumes that only one replica is memory residentand the other or the secondary replicas written to mechanical disks.

In [14], Mike Mitzenmacher proposed the use of a choice between two options for load balancing. While the utilization of choice is common between this work and [14], Mitzenmacher's work focused on achieving a better load balancing for achieving better performance of system, while this work focuses on achieving a better bundling, which reduces the total amount of work that the system performs.

CONCLUSIONS

Here we considered the D-RAM based storage for our results. In this paper, it is discussed that efficiency and scalability of RAM- Based storage systems can be maximized by reducing the number of transactions per user request.

RnB is a combination of object replication on different servers and the bundling of requested items into a single transaction, which reduces the total amount of work that the system performs.

In addition to the basic RnB scheme various enhancements, such as declaring a larger number of replicas than can actually be stored in memory have been proposed and evaluated. We have developed the efficient technique such as ranged consistent hashing, which is for finding the number of servers that jointly possess the replicas of the requested item.

RnB does not create any extra work for the front-end servers. The object replicas are already present for other reasons in the system, like for availability, fault tolerance etc. so we don't need to put extra effort for replication of data on multiple servers. So the main cost element of RnB comes almost for free. RnB also supports smooth scalability and is relatively easy to incorporate in.

Summarization: The existing systems have the technologies which are used in this mechanism but there are some limitations of these systems. Replication is done in many systems for other reasons, here it is used for increasing the efficiency of the system, and jointly it provides availability to the system. Cache misses are less in this case. So the replication in this system is nearly free.

Bundling of the consecutive item requests from the end user is done, so that it reduces the number of transactions per user request. This is the main issue of the mechanism.

LRU is the mechanism used in addition to the basic RnB for reducing the unnecessary use of the memory used for the system. If there are replicas which are vey "cold" are removed from the server so that the actual needed data can occupy the server memory space.

RCH is Ranged Consistent Hashing which is an extension to the Consistent Hashing. It provides the set of servers that jointly possess the requested item set.

REFRENCES

[1] Shachar Raindel and Yitzhak Birk "Replicate and Bundle (RnB) – A Mechanism for Relieving Bottlenecks in Data Centers" 2013 IEEE 27th International Symposium on Parallel & Distributed Processing

[2] J. Rothschild, "High Performance at Massive Scale – Lessons learned at Facebook." http://cns.ucsd.edu/lecturearchive09.shtml#Roth, October 2009.

[3] R. M. Karp, "Reducibility Among Combinatorial Problems," in *Complexity of Computer Computations* (R. E. Miller and J. W. Thatcher, eds.), pp. 85–103, Plenum Press, 1972.

[4] Tonglin, Xiaobing Zhou, Kevin Brandstatter, Dongfang Zhao "ZHT: A Light-weight Reliable Persistent Dynamic Scalable Zero-hop Distributed Hash Table" 2013 IEEE 27th International Symposium on Parallel & Distributed Processing

[5] Ahmad Waqas, Nahadia Majeed, "Dynamic Object Replica Placement using Underlying Routing Protocol: Ensuring the Reliability", 2013 5th International Coference on Information and Communication Technology for the Muslim World.

[6] "Memcached Overview Page." http://code.google.com/p/memcached/wiki/NewOverview, Feb. 2013.

[7] S. Raindel, "Replicate and Bundle (RnB) - A Mechanism for Relieving Bottlenecks in Data Centers," M.Sc. thesis.

[8] I. Hoque and I. Gupta, "Social Network-Aware Disk Management," tech. rep., University of Illinois at Urbana-Champaign, Dec. 2010.

[9] "Feature List of Moxi." http://code.google.com/p/moxi/, Feb. 2013.

[10] "Spymemcached Optimizations Description." http://code.google.com/p/spymemcached/wiki/Optimizations, Feb. 2013.

[11] D. G. Andersen, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasudevan, "FAWN: A Fast Array of Wimpy Nodes," in *ACM Symposium on Operating Sys. Principles (SOSP)*, (Big Sky, MT), Oct. 2009.

[12] J. Terrace and M. J. Freedman, "Object storage on CRAQ: high-throughput chain replication for read-mostly workloads," in *USENIX*, pp. 11–11, USENIX, 2009.

[13] D. Ongaro, S. M. Rumble, R. Stutsman, J. K. Ousterhout, and M. Rosenblum, "Fast crash recovery in RAMCloud," in *ACM Symposium on Operating Systems Principles (SOSP)* (T. Wobber and P. Druschel, eds.), pp. 29–41, ACM, 2011.

[14] M. Mitzenmacher, "The power of two choices in randomized load balancing," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 12, pp. 1094 –1104, Oct. 2001.