Distributed Database: Concepts and Applications

Priyanka Singh
Department of Computer Engineering
CPC, Chandrapur, MS (India)

ABSTRACT
The author wants to submit the endeavor about the Distributed Database storage concepts and usefulness in large enterprises. One of the most useful storage technique for Distributed Database is sharding and the concept of database sharding has gained popularity over the past several years due to the enormous growth in transaction volume and size of business-application databases and database security also. Database sharding can be simply defined as a "shared-nothing" partitioning scheme for large databases across a number of servers, enabling new levels of database performance and scalability. If you think of broken glass, you can get the concept of sharding—breaking your database down into smaller chunks called "shards" and spreading them across a number of distributed servers.

High performance web applications often reach the limits of one database server. Such systems require a smart distribution of data. Sharding is a mechanism that helps the application to scale horizontal and gain responsibility by splitting information across multiple servers. The paper will give an introduction on sharding and possible implementations as well as covering problems with this approach.

1. INTRODUCTION
Sharding is a database technique where you break up a big database into many smaller ones. In other words, Sharding, or horizontal partitioning, is the process of splitting a database into separate instances (or shards) based on row data (as opposed to columns, which is the process of normalization). This process makes huge amounts of data more manageable and consequently, can increase system performance (i.e. queries are faster due to smaller table data).

Fragmentation has a very important role in distributed databases. Fragmentation means partitioning of data .when the data is very large partitioning is done to improve the performance. This partitioning can be done horizontally or vertically. Vertical partition can be performed easily using normalization. As we normalize the data vertical fragmentation is done. Horizontal fragmentation means to store the different tuples at different places. This horizontal fragmentation means Data Sharding.

Each partitioned forms part of a shard, which may be located on a separate database server or physical location. The advantage is the number of rows in each table is reduced (this reduces index size, thus improves search performance). dbShards is the industry's first software product that allows database sharding to be applied to existing applications and databases with little or no modification to existing code. Database sharding is a simple concept - instead of storing application data in a single database on a single server with shared CPU, memory and disk, the database is divided into a number of smaller "shards", each of which can be hosted on independent servers, with dedicated CPU, memory and disk, therefore greatly reducing resource contention. Because each shard is small, the database server can do a much better job of storing indexes and query caches in memory, resulting in significantly improved performance. Just as databases slow down exponentially as they grow beyond the limits of a single server, sharding a database can result in better-than-linear performance gains.

2. THE SHARDING PROCESS
The sharding process includes a number of steps. The data is divided into number of shards and the different shards are stored on different servers. But here comes various problems like joining .The sharding process does not includes replication. Converting an existing application to work with the sharded database is simply a matter of replacing the database driver with the dbShards database driver as well as installing and configuring dbShards replication and query agents on each shard server. dbShards currently supports Java, PHP and Ruby clients.

Using data sharding load of different servers can be distributed. In data sharding different fragments are stored on different nodes. each node maintains its copy of fragment.

For example if we want to store the data of bank branch wise each branch will have the same fields but they will have different rows.

Branch name	Branch Address

Customer ID	Customer Name	Cust Address	Phone no

Acc no	Acc type	Balance

In the above example each branch will have the tables but will have different rows.

Accounts

Acc no	**Acc type**	**Balance**
1010010	SAVINGS	25000
1025639	SAVINGS	15000
1010010	SAVINGS	25000
1025639	SAVINGS	15000
2056987	CURRENT	32000
2056987	SAVINGS	50000

Shards are as follows:

Accounts with Branch 1

Acc no	Acc type	Balance
1010010	SAVINGS	25000
1025639	SAVINGS	15000

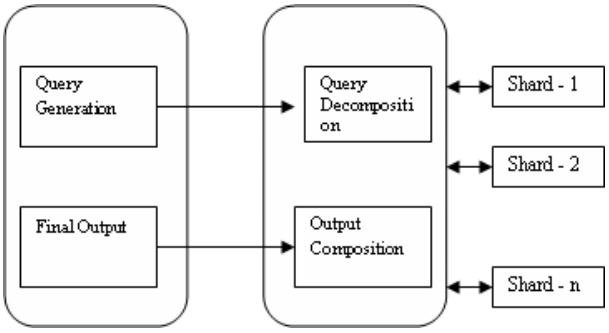
Accounts with Branch 2

Acc no	Acc type	Balance
3012562	CURRENT	300000
3256014	SAVINGS	12500

Accounts with Branch 3

Acc no	Acc type	Balance
2056987	CURRENT	32000
2056987	SAVINGS	50000

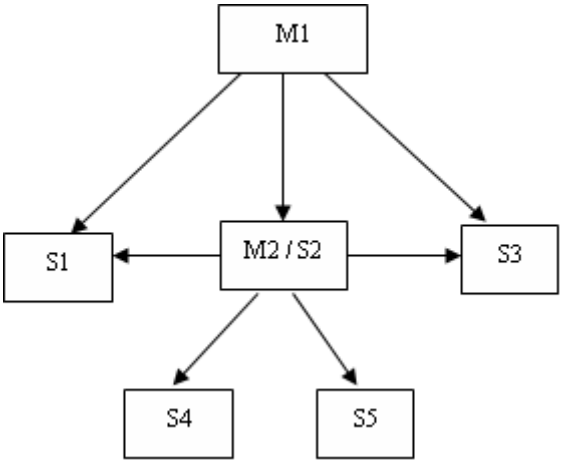
To keep the servers in the sharding system balanced we are monitoring several parameters such as number of users, file size of tables and databases, amount of read and write queries, cpu load, etc. Based on those stats we can make decisions to move shards to new or different servers, or even to move users from one shard to another. Move operations of single users can be done completely transparently and online without that user experiencing downtime. We do this by monitoring write queries. If we start a move operation for a user, we start copying his data to the destination shard. When a write query is executed for that user, we abort the move process, clean up and try again later.



The Sharding Process is based on partitioning. If we want to partition the data it can be done using σ operator. The selection (σ) operator is used to partition the rows according to the condition specified. Let us consider the complete database as the master database and the result obtained after the operation is called the slaves. The slave can again be divided into slaves. The Account table as mention above is a master table

And the accounts at branch 1, 2, 3 are the slaves. If each branch have many account it can again be sharded.

Let us call master as M and Slaves as S.The master M2 will act as M1 if the master M1 fails.



3. THE ADVANTAGES

- *Easily available.* If one box goes down the others still operate.
- *Speed Increases.* As the amount of data is less thev queries access is fast..
- *More write bandwidth*. With no master database serializing writes you can write in parallel which increases your write throughput. Writing is major bottleneck for many websites.
- *Extra work can be done*. As the data is stored at various sites Parallel processing can be done .

4. THE PROBLEMS

Along with the advantages.Sharding has some disadvantages also which are as follows:

- *Rebalancing data*. Sometimes when the data is very large.It has to be rebalanced because shards may be smaller as compared to the data.
- *Shards have to be rejoined*. Data is stored at different places which has to be rejoined for use.
- *How do you partition your data in shards?* What data do you put in which shard? Where do comments go? Should all user data really go together, or just their profile data? Should a user's media, IMs, friends lists, etc go somewhere else? Unfortunately there are no easy answer to these questions.
- *Less leverage*. People have experience with traditional RDBMS tools so there is a lot of help out there. You have books, experts, tool chains, and discussion forums when something goes wrong or you are wondering how to implement a new feature. Eclipse won't have a shard view and you won't find any automated backup and restore programs for your shard. With sharding you are on your own.
- *Implementing shards is not well supported.* It is not properly supported.
- *Refrential integrity cannot be maintained* **.** As the data is stored on different database servers therefore it is difficult to maintain the foreign key for different servers because many database management server does not support it. denormalization and lack of referential integrity can

become a significant development cost to the service.

- *Rebalancing* **.** As there may be difference be difference in the data present in various shards.Some may contain very large data and some may not therefore it is necessary to rebalance the data.

5. SOLUTION OF THE PROBLEMS

The difficulties of sharding are partially tackled by implementations with these 3 technologies: Caching the memory, parallel processing and full text search engine.

5.1 Caching the memory

It is a memory caching system for distributed database it increases the speed of the database. This system is very fast. This technique is called "memcached". By putting a memory caching layer in between our application logic and the SQL-queries to our shard database we are able to get results much, much faster. This caching layer also allows us to do some of the cross-shard data fetching, previously thought impossible on SQL-level.

5.2 PARALLEL PROCESSING

Parallel processing increases the speed of execution. In parallel processing several servers performs the task and each server performs a small task. This in results increases the performance.

It is not strange to fetch data stored on different shards in one go, because most data is probably available from memory The amount of actual database queries needed for this will be small, and even so, the queries are simple and super fast .Problem arises when a large amount of data is to be retrieved. For this we've implemented a system for splitting up certain big tasks into several smaller ones we can process in parallel. It is actually faster to process 10 smaller tasks simultaneously than to do the whole thing at once. The overhead of the extra web requests and cpu cycles it takes to split up the task and combine the results, are irrelevant compared to the gain.

5.3 Full text search engine(Sphinx)

Other typical queries that become impossible for sharded data are overview queries. Say you'd like a page of all the latest photos uploaded by all users If we have data distributed over a hundred of databases, you'd have to query each, and then process all of those results. Doing that for several features would not be justifiable, so most of our "Explore" pages are served from a different system. Sphinx is a free and open source SQL full-text search engine. In fact a list of most viewed videos of the day can also be a query result from Sphinx. For most of the data on these overview pages it's not a problem if the data isn't real time. So it's possible to retrieve those results from indexes that are regularly built from the data on each shard

6. DATA SHARDING IS UNIQUE WHY ?

Sharding is different than traditional database architecture in several important ways:

- *Data are denormalized*. Traditionally we normalize data. Data are splayed out into anomaly-less tables and then joined back together again when they need to be used. In sharding the data are denormalized. You store together data that are used together.
- *Data are parallelized across many physical instances*. Historically database servers are scaled up. You buy bigger machines to get more power. With sharding the data are parallelized and you scale by scaling out. Using this approach you can get massively more work done because it can be done in parallel.
- *Data are kept small*. The larger a set of data a server handles the harder it is to cash intelligently because you have such a wide diversity of data being accessed. You need huge gobs of RAM that may not even be enough to cache the data when you need it. By isolating data into smaller shards the data you are accessing is more likely to stay in cache.

Smaller sets of data are also easier to backup, restore, and manage.

- *Data are more highly available*. Since the shards are independent a failure in one doesn't cause a failure in another. Keeping multiple data copies within a shard also helps with redundancy and making the data more parallelized so more work can be done on the data. You can also setup a shard to have a master-slave or dual master relationship within the shard to avoid a single point of failure within the shard. If one server goes down the other can take over.

6.1 Methods of Sharding

Sharding can be done in various ways .If we want to save large amount of data on various servers it has to be divived into many data bases .Some of the sharding schemes defined as follows.

- *Vertical Partitioning:* In Vertical Partitioning the data is stored to the related server.But there is a problem that some server may have very large data in that case again sharding is done.
- *Range Based Partitioning*: In range based sharding a particular field is selected according to the value of that field shard are made.It is necssary to look after the fiel dwhich is selected because if the field is not properly selected then There may be imbalance in the shards.
- *Key or Hash Based Partitioning:* This is often a synonym for user based partitioning for Web 2.0 sites. With this approach, each entity has a value that can be used as input into a hash function whose output is used to determine which database server to use.
- *Directory Based Partitioning:* A loosely couples approach to this problem is to create a lookup service which knows your current partitioning scheme and abstracts it away from the database access code.

6.2 Partitioning vs. Sharding

A partition is a structure that divides a space into two parts. Multiple partitions can break up that space into an arbitrary number of parts. In computer operating systems, this even has a more specific definition referring to the division of resources into portions. As a verb it means to divide something (typically a space) into small pieces.

Partitioning is a more general concept and it can be applied to databases at many levels. One common use is taking a single large table and splitting it into parts in order to place those parts that are accessed more frequently on faster (more

expensive) storage. However, partitioning isn't limited to a single machine. That partitioning schema was to allow use of more than one (and even a different type/cost) disk spindle. It can also be applied to multiple database instances; it is a loose term. However, partitioning does not imply a logical separation. Some partitioning schemes require mapping questions across many nodes and some partitioning schemes provide a priori knowledge about which components hold what data allowing more targeted questioning.

A shard is a piece of broken ceramic, glass, rock (or some other hard material) and is often sharp and dangerous. Sharding is the act of creating shards. Sharding really is different than database partitioning. With partitioning, you traditionally split up your logical data model into parts and put each of the data partitions on different servers. One logical record in a partitioned database usually has its information stored across each of the partitions. Sharding differs from data partitioning in two ways. Firstly, each database server is identical, having the same table structure. Secondly, the data records are logically split up in a sharded database. Unlike the partitioned database, each complete data record exists in only one shard in that database. You may not like the terminology used, but this does represent a different way of organizing a logical database into smaller parts.

As we understand, the difference between partitioning and sharding is that sharding applies specifically to the technique of horizontal partitioning, whereas partitioning itself could be either horizontal or vertical. The term sharding is slightly more specific.

7. CONCLUSION

OUR PERSONAL INFORMATION IS IMPORTANT TO ALL OF US. WE WANT TO BE ABLE TO SHOP ONLINE AND HAVE GOODS DELIVERED TO OUR DOOR. WE WANT RESPONSIVE, ACCESSIBLE AND INDIVIDUALIZED PUBLIC SERVICES. THEREFORE, THE authors want to bring into light the concept of data sharding and the problems Faced by data sharding. Data sharding makes the query processing faster. Extra work can be done in less time .The data sharding approach needs data security which can be explored more widely

8. REFERENCES

[1] www.mongodb.org/display/DOCS/Sharding+Introduction

[2] www.datacenterknowledge.com

[3] www.genomel.org

[4] www.icpsr.umich.edu

[5] Sharing Data from Large-scale Biological Research Projects: A System of Tripartite Responsibility (Wellcome Trust, 2003); available at http://www.wellcome.ac.uk/stellent/groups/corporatesite/@policy_communications/documents/web_document

[6] http://lifescaler.com/2008/04/database-sharding-unraveled-part-i/

[7] Kraitchik, M. "The Unfinished Game." §6.1 in Mathematical Recreations. New York: W. W. Norton, pp. 117-118, 1942.

[8] www.wow.com/2009/

[9] 08/19/wow-rookie-sharding-etiquette/