Distributed Database

Terry Yin

November 16, 2014

On this Tuesday (November 11, 2014), in one day, the biggest online shopping store in China, Taobao, recorded 9.3 billion USD sales. In the first minute of this "Single Day sales", it recorded 19 million USD sales. What kind of database can process data flow like this?

Before we analyze why certain distributed database is desired in some stituation, it's important to remember the following numbers Boner (2014).

Latency Comparison Numbers

Main memory reference	100	ns		20x L2 cache, 200x L1 cache
Send 1K bytes over 1 Gbps network	10,000	ns	0.01 ms	
Read 1 MB sequentially from memory	250,000	ns	0.25 ms	
Round trip within same datacenter	500,000	ns	0.5 ms	
Read 1 MB sequentially from SSD	1,000,000	ns	1 ms	4X memory
Disk seek	10,000,000	ns	10 ms	20x datacenter roundtrip
Read 1 MB sequentially from disk	20,000,000	ns	20 ms	80x memory, 20X SSD
Send packet CA->Netherlands->CA	150,000,000	ns	150 ms	

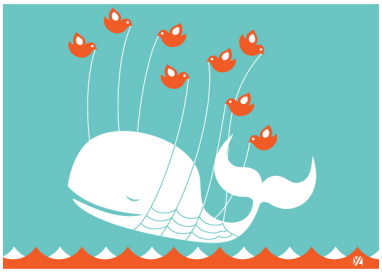
Notes

- 1 ns = 10-9 seconds
- 1 ms = 10-3 seconds

1 Motivations of Using Distributed Database

1.1 Capacity

The most basic database structure of Twitter could be as simple as only two tables, one for the users, the other for their twits. If you are a frequent twitter user, you must have seen this picture before. So what makes it so complicated to deal with only two tables?



The biggest challenge is the tables are too big, and keep growing. In the beginning probably a plain text file is enough to keep all the twits. Next it need a centralized database. When the table grew too big, they

might need a separate database for each table. But when the table grew too big, they would need to split the table. There are basically two ways of splitting a table into a distributed database Wikipedia (2014*b*):

- **Horizontal fragmentation** is to split the columns of the table into different databases.
- **Vertical fragmentation** is to split the records of the table into different databases.

From the table of latency at the beginning of this paper we can see, if we only want capacity, the databases need to be inside one building or at least within the same data center. Otherwise the latency would make the database access on the same table too slow.

1.2 Low Latency and High Availability

Traditionally, a database need to have the ACID properties Wikipedia (2014*a*)

- A-atomicity, the transaction with either be fully successful, or leave no influence.
- C-consistency, maintain the data integrity all the time.
- I-isolation, during the transaction, the related data are not changing.
- D-durability, the data is safe.

The challenge of ACID is when database scales up it becomes very hard to maintain them all. So sometimes people use a different CAP theorem for distributed database Brewer (2012). The CAP theorem says in a distributed DBMS, if you cannot have all the ACID, you can have two of the three interesting properties of a DBMS. These three properties are:

- **Consistency** your data is correct all the time. Everyone accessing the database gets the same result.
- **Availability** you can read and write your data all the time.
- **Partition Tolerance** if one or more nodes fails the system still works and becomes consistent when the system comes on-line.

Put it in another word, you can get the other two properties by giving up one of the three. For example, if we don't ask for consistency all the time, then it's easier to get high availability and partition tolerance. We can replay consistency with something called **eventual consistency**. For example, it is OK if each person reviewing twitter from a different side of the planet might see slightly different twits even if the are reviewing the same thing for a short period like several minutes.

Because we no longer require all the ACID properties, it is OK if our database is distributed across the whole country or even worldwide. We want the first level database get closer to the users, so it is common to have one or multiple databases in every country or city that we want the users having a better experience from there.

1.3 Protecting Data

Another motivation of using a distributed database is to protect the data. In engineering, the most common (if not the only) way of increasing reliability is by increasing the redundancy Wikipedia (2014*c*). By replicating the data in different geographical locations, we can protect the data from corruption by multiple reasons, e.g. a disaster.

Big companies usually have multiple data centers worldwide. One of the motivation is to protect their data. One interesting finding is to mirror the data between data centers is very costly. As the founding CEO of Docker Solomon Hykes said:

> "I feel it's embarrassing, personally, that on average it will take more time and energy to get a collection of software from one data center to the next than it is to ship physical goods from on side of the planet to another. Hykes (n.d.)"

2 Conclusion

In the age of big data, we often need a distributed database rather than a local database. The reasons we need distributed database includes capacity, low latency, high availability and protecting data. Based on the different need we may choose different geographical distribution. Given the current trend of the IT industry, it more about which data center providers to choose, and what kind of server distribution plan we want from these data center providers.

References

Boner, J. (2014), 'Latency numbers every programmer should know', https://gist.github.com/jboner/2841832.

Brewer, E. (2012), 'Pushing the cap: Strategies for consistency and availability', *Computer* **45**(2), 23–29.

Hykes, S. (n.d.), 'What is docker?', https://www.youtube.com/watch?v=ZzQfxoMFH0U#t=245. Accessed: 2014-11-17.

Wikipedia (2014*a*), 'Acid — wikipedia, the free encyclopedia'. [Online; accessed 16-November-2014].
URL: *http://en.wikipedia.org/w/index.php?title=ACID&oldid=633228986*

Wikipedia (2014*b*), 'Distributed database — wikipedia, the free encyclopedia'. [Online; accessed 16-November-2014].
URL: *http://en.wikipedia.org/w/index.php?title=Distributed_database&oldid=630771581*

Wikipedia (2014*c*), 'Redundancy (engineering) — wikipedia, the free encyclopedia'. [Online; accessed 16-November-2014].
URL: *http://en.wikipedia.org/w/index.php?title=Redundancy_(engineering)&oldid=632965998*