Database Choices

Introduction

There are mainly two categories of database management system, which are relational database (RDBMS) and non-relational database (NoSQL), and dozens or even hundreds of products for each category. Obviously, each one has its own advantages and disadvantages, and how to choose a suitable database could be a problem. This article briefly introduces four distinctive databases, analyzing their pros/cons, as well as recommending proper use scenarios.

Relational Database

The first category is relational database, or relational database management system (RDBMS). RDBMS has been in use since 1980s and was almost the only option for data storage for a long time.

The most distinguishing features of RDBMS are that it adopts the relational model, which is considering data as entities and relations, to store data and uses a standard query language, SQL, to access data.

There are several well-known RDBMS, both open source and commercial, like Microsoft SQL Server, Oracle, MySQL, etc. All of them are versatile, mature and popular, and well analyzed by others, and won't be introduced in this article.

The one will be introduced here is SQLite.

SQLite

Advantages

- SQLite is a self-contained, file-based relational database.
  It has very few dependencies, and could run on any operating systems. It stores data as file format, and when an application query data, the integration works with functional and direct calls made to a file holding the data, making the access fast and efficient.
- SQLite implements standard SQL syntax.

Not like some other RDBMS, SQLite exactly follows the SQL standard, letting the migration between SQLite and another standard SQL database easier.

Disadvantages

- It is lack of some features such as user management.
  By design, SQLite is focusing on simplicity and portability, thus it scarifies many features, like user management and concurrency control.
- It has bad performance in terms of scalability and concurrency.
  Again, by design, it is not used for complex application or dealing with high concurrent, high throughput scenarios.

When to use

- One use scenario may be single user light weight application.
  In its website, it claims that SQLite could handle 400K to 500K http requests per day, which is enough for small to medium level website.
- The other use scenario could be local test for web application.
  Since it is easy to configure and set up, SQLite is very suitable for local testing of web application. It won't take too long to populate some fake data locally and connect front end app or even backend app to it.

Non-relational Database

Another category of database is non-relational database, which is also called NoSQL, meaning "Not only SQL". Though many products are called NoSQL, they are quite different from each other. Basically, NoSQL refers to a set of schema free database that eliminate the use of the standardized SQL. Behind those databases, there are various data models, such as document model, key/value model, and column model.

Each data model has its own merits as well as shortcomings, leading to specific sweeting zones for use. Thus, the best way to choose a correct database is to learn what each database is good at and what is not. Since there are too many products in market to cover, two representative cases, Cassandra and Couchbase, are chosen here.

Cassandra

Advantages

- Cassandra is a highly scalable, high-performance database.
  Cassandra is designed to handle large amounts of data across many commodity servers. Thanks to the column data model, its performance is excellent even under petabyte level data environment. Also, it claims that Cassandra could be linearly scaling-up. In contrast, most of RDBMS could only scale-up exponentially.
- Cassandra uses distributed design, making it single points of failure tolerant.
  Because of distributed architecture design, there are no single points of failure as well as no network bottlenecks.

Disadvantages

- It is lack of data consistency.
  As a trade-off, it scarifies the data consistency, meaning that after one user update some data, another user is not guaranteed to see this update immediately. Thus, it is not suitable for online transaction/purchase application.
- The learning curve is high.
  Due to the complicated architecture, Cassandra is hard to configure and maintain. It requires expert knowledge to maximize the performance, result in a technique barrier to new coming users.

When to use

- Cassandra is good for heavy data access applications with low consistency requirement.
  Cassandra emphasizes scalability and performance. So, if the application is not large enough, or the performance requirement is low, it's better to avoid to use it. It should be applied only if there is a concern of dealing with extremely high volume or huge throughput of data access. In fact, it was first developed for Facebook inbox application.

Couchbase

Advantages

- Couchbase implements a query language called N1QL

N1QL is a declarative query language that extends SQL for JSON. It is similar as SQL, so it is very easy to leverage knowledge of SQL to easily create query script with N1QL. By doing that, it significantly saves time for experienced SQL developer, improving their efficiency.

- Couchbase has built-in in-memory cache functionality
  The second advantage is that it has built-in in-memory key-value store engine, like Redis, which could be used as cache or high read/write performance storage. This feature is attractive since cache is widely used everywhere. By integrating two data model together, it reduces the workload for switching between different databases.

Disadvantages

- Its data consistency is also not guaranteed.
  Comparing to relational database, it also can't guarantee data consistency. That's because of the nature of NoSQL database.
- It is relatively new.
  Though it is growing fast since its first release at 2011, Couchbase is still a noob in database area. Comparing to MangoDB, its primary competitor, Couchbase could be less stable and has more bugs. Moreover, the smaller community is also less supportive.

When to use

- If developers are familiar with SQL but want high performance or deal with unstructured data, then Couchbase is a good fit.
  Its N1QL makes development much more productive if all developers are familiar with SQL, and its nature to handle unstructured data provides a shortcut achieve better overall performance when majority of data are unstructured.
- If the application needs in-memory cache together with data storage.
  It's better to use one Couchbase instead of using MangoDB+Redis since it is easy to manage and maintain.

Conclusion

There is no one database could handle all cases, and choosing the right one for the application finally come down to questions:

1. What are the requirements for the database?
2. Which data model is best for my data?
3. What does the typical request look like?
4. What knowledge do I have?

By answering those questions, it would be easy to figure out what database should be used.

In summary, the best places to use NoSQL database is where the data model is simple, where flexibility is more important than strict control over defined data structures, where high performance is a must, and strict data consistency is not required. Moreover, to deal with very large sets of data, consistently scaling is easier to achieve with many of the NoSQL databases, like Cassandra.

On the flip side, if the application need consistent data, durable writes, transactions and the data has complex structures and relationships, a relational database is still the best choice.

Reference

https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems#database-management-systems

https://www.digitalocean.com/community/tutorials/a-comparison-of-nosql-database-management-systems-and-models

http://db-engines.com/en/ranking

http://cassandra.apache.org/

https://www.tutorialspoint.com/cassandra/cassandra_introduction.htm

https://sqlite.org/features.html

http://www.javaworld.com/article/2078750/open-source-tools/nosql-showdown--mongodb-vs--couchbase.html

http://www.couchbase.com/