

Storage Systems For Scalable systems

Haytham ElFadeel

Researcher in Computer Sciences

Agenda

- Introduction
 - Glance at the Scalable systems.
 - What the available storage solution.
 - The problem with the current solutions.
 - The problem with the Database
- The Next-Generation of Storage System
 - Key-Value store systems.
 - Performance comparison.
- How it's works
- Discussions, Q/A

Glance at Scalable Systems

- Scalable systems
 - Scalability is the ability to provide better performance when you add more computing power.
 - This performance gained should be relevant to the added computing power.
 - Examples: Google, Yahoo, Facebook, Amazon, eBay, Orkut, Google App Engine, etc.

Glance at Scalable Systems

- Scalable types
 - **Vertical Scalability:** Adding resource within the same logical unit to increase the capacity. For example: Add more CPUs, or expanding the storage or the memory.
 - **Horizontal Scalability:** Add multiple logical units of resources and make them together work as a single unit. You can think about it like: Clustering, Distributed, and Load-Balancing.

Vertical Scalability vs. Horizontal Scalability

Vertical Scaling

Limited

Hardware only

Horizontal Scaling

Not limited

Software and Hardware

Vertical Scalability vs. Horizontal Scalability

Haytham ElFadeel Quote:

If you need scalability, urgently, going to vertical scaling is probably will to be the easiest, but be sure that Vertical scaling, gets more and more expensive as you grow, and While infinite horizontal linear scalability is difficult to achieve, infinite vertical scalability is impossible.

Vertical Scalability vs. Horizontal Scalability

Haytham ElFadeel Quote:

On the other hand Horizontal scalability doesn't require you to buy more and more expensive hardware. It's meant to be scaled using commodity storage and server solutions. But Horizontal scalability isn't cheap either. The application has to be built ground up to run on multiple servers as a single application.

Glance at Scalable Systems

- Facebook
 - More than 200,000,000 active user.
 - 50,000 photo uploaded per minute.
 - The most active social-network in the Web.
- Facebook chat
 - The main challenge is maintain the users status.
 - Distribute the load should depend on the users, and they friends to avoid the traveling.
 - Building a system that should scale from that start to serve 100,000,000 user is really hard.

Glance at Scalable Systems

- Amazon
 - More than 10,000,000 transition in every holidays.
 - The Reliability of the user shopping cart is not option.
- Google, Yahoo, Microsoft, Kngine, etc
 - Processing huge amount of data, more than 1TB.
 - Sorting the index by the rank value. Which means, sort more than 1TB of data.
 - Save the Crawled Web pages.

The Available Storage Solutions

- Memory:
 - Just a Data Structure :)
- Disk:
 - Text File: { XML, Protocol Buffer, Json }
 - Binary File: { Serialized, custom format }
 - Database: { MySQL, SQL Server, SQLite, Oracle }

The Available Storage Solutions

- Memory:

What about capacity

- Just a Data Structure :)

- Disk:

Bad performance

- Text File: { XML, Protocol Buffer, Json }

- Binary File: { Serialized, custom format }

Not portable, questions about performance

- Database: { MySQL, SQL Server, SQLite, Oracle }

Bad performance,
Complex, huge latency.

The Problem with the Database

- **Causes**

- Old and Very complex system.
- Many wasted features.
- Many steps to process the SQL query.
- Need administration, and others.

The Problem with the Database

- **Causes**

- Old and Very complex system.

- The RDMS is very complex system, just like Operating System:

- Thread Scheduling, Deadlock monitor, Resource manager.

- I/O Manager, Pages Manager, Execution Plan Manager.

- Case Manager, Memory Manager, Transaction Manager, etc.

- Most of DBMS architecture, designs, algorithms came up around 1970s:

- Different hardware, platform properties.

- Old architecture, design, and algorithms.

Please review resource #1

The Problem with the Database

- **Causes**

- Many wasted features.

- Today systems have very rich features, simply because they think that ‘one size fits all’:

- CLR Types, CLR Integration, Replication, Functions.

- Policy, Relations, Transaction, Stored procedure, ACID, etc.

- You can even call a Web Service from SQL Server! All this mess, make the database appear like a platform and development environment.

The problem with the Database

- **Causes**

- Many Steps to process the query.
 - Parse the Query.
 - Build the expression tree, and resolve the relational algebra expression.
 - Optimize the expression tree.
 - Choice the execution plan.
 - Start execute.

Please review resource #2, #3

The problem with the Database

- **Effects**

- Bad Performance: Throughput, Resource usage, Latency.
- Not Scalable.

The problem with the Database

- **Effects**

- Bad Performance: Throughput, Resource usage, Latency:

- Even the faster DBMS 'MySQL' can't provide more than 5,000 query per second*.
 - Add to this the consumed resource, and the big latency.

* Depend on the configuration

The problem with the Database

- **Effects**

- Not Scale:

- The Database is not designed to scale.
 - Even if you get a new PC and partition the Database you will never get (accepted) good performance improvement.

The problem with the Database

The Database give us ACID:

- **Atomicity:** A transaction is all or nothing.
- **Consistency:** Only valid data is written to the database.
- **Isolation:** pretend all transactions are happening serially and the data is correct.
- **Durability:** What you write is what you get.

The problem with the Database

The problem with ACID is that it gives you too much, it trips you up when you are trying to scale a system across multiple nodes.

Down time is unacceptable. So your system needs to be reliable. Reliability requires multiple nodes to handle machine failures.

To make a scalable systems that can handle lots and lots of reads and writes you need many more nodes.

The problem with the Database

Once you try to scale ACID across many machines you hit problems with network failures and delays. The algorithms don't work in a distributed environment at any acceptable speed.

It's a dead end

The Next generation of Storage Systems

From long time ago many researches teams and companies discovered that the database is main bottleneck.

Many wasted features, bad performance, and not designed for scale systems.

The Next generation of Storage Systems

Building large systems on top of a traditional RDBMS data storage layer is no longer good enough.

This talk explores the landscape of new technologies available today to augment your data layer to improve performance and reliability.

Please review resource #4

Key-Value Storage Systems

- Simple data-model, just key-value pairs.
- Every Value Assigned to Key.
- No complex stuff, such as: Relations, ACID, or SQL quires.
- Simple interface:
 - Get(key)
 - Put(key, value)
 - Delete(key) < Optional

Key-Value Storage Systems

- Designed from the start to scale to hundreds of machines.
- Designed to be reliable, even if 50% of the machines crashed.
- No extra work require to add new machine, just plug the machine and it will work in harmony.
- Many open source projects (C++, Java, Lisp).

Key-Value Storage Systems

- Who use such systems:
 - Facebook.
 - Google Orkut, Analysis.
 - Google Web Crawling.
 - Amazon.
 - Powerset.
 - eBay.
 - Kngine.
 - Yahoo.
 - General using.
 - Storing, and huge data analysis.
 - Transactions, and huge data analysis.

Key-Value Storage Systems

You may wonder, can we really live without Relations, ACID ?!

- The short answer: Absolutely Yes.
- The long answer: Absolutely Yes, But nothing for free.

Key-Value Storage Systems

Now

You should make your decide

**Take the blue pill
And see the truth**

**Or, Take the red pill
And stay in
wonderland**

Key-Value Storage Systems

Key-Value Storage System, and other systems built around CAP concept:

Consistency: your data is correct all the time. What you write is what you read.

Availability: you can read and write 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.

Key-Value Storage Systems

One Node - Performance Comparison (Web)

- **MySql**
 - 3,030 sets/second.
 - 4,670 gets/second.
- **Redis**
 - 11,200 sets/second. (3.7x MySQL)
 - 9,840 gets/second. (2.1x MySQL)
- **Tokyo Tyrant**
 - 9,030 sets/second. (3.0x MySQL)
 - 9,250 gets/second. (2.0x MySQL)

Please review resource #5

Key-Value Storage Systems

Two High-End Nodes - Performance Comparison
(Web)

- **Redis**
 - 89,230 sets/second.
 - 85,840 gets/second.

Key-Value Storage Systems

One Node - Performance Comparison

- **SQL Server**
 - 2,900 sets/second.
 - 3,500 gets/second.
- **Vina***
 - 10,100 sets/second. (3.4x SQL Server)
 - 9,970 gets/second. (2.8x SQL Server)

* Vina : Key-Value Storage System used inside [Kngine](#).

How it's Works

Any Key-Value storage system, consist of two primary layers:

- **Aggregation Layer**
- **Storing Layer**

How it's Works

Any Key-Value storage system, consist of two primary layers:

- **Aggregation Layer**

- Manage the instances, replication and distribution.

- **Storing Layer**

- One or many Disk-based Hash-Table.

How it's Works (Storing Layer)

On the board

How it's Works (Aggregation Layer)

- Received the requests.
- Route it to the target node.
- Manage Partitioning, and Replicas.
- The Partitioning, Replication done by Consistence Hashing algorithm.

On the board

Please review resource #6

Key-Value Storage Systems

- Amazon Dynamo. < Paper
- Facebook Cassandra. < Open source
- Tokyo Cabinet/Tyrant. < Open source
- Redis < Open source
- MongoDB < Open source

Q / A



References

1. The End of an Architectural Era (It's Time for a Complete Rewrite). Paper.
2. Database Systems - Paul Beynon-Davies. Book.
3. Inside SQL Server engine - MS Press. Book.
4. Drop ACID and Think About Data. Highscalability.com.
5. Redis vs MySQL vs Tokyo Tyrant. Colin Howe's Blog.
6. Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web. Paper.
7. Dynamo: Amazon's Highly Available Key-value Store. Paper.
8. Redis, Tokyo Tyrant project.
9. Consistent Hashing. Tom white Blog.

Resources

1. High Scalability blog.
Highscalability.com
1. It's all about innovation blog.
Hfadeel.com/blog.
2. All Things Distributed.
Allthingsdistributed.com
3. Tom White blog
lexemetech.com



Thanks...

Dear all,

All of my presentation content it's open source.

Please feel free to use, copy, and re-distribute it.