CS8803 BDS / CS4365

In-Memory Data Management

Prof. Dr. Ling Liu
School of CS
Georgia Institute of Technology

Slides are adapted from Internet sources: Makoto Shimazu@TauraLab2014, tim_ellison@uk.ibm.com, JordanCao&UddhavGupta@SAP HANA2013, Operating System Concepts, 7th Edition, Silberschatz, Galvin and Gagne ©2005

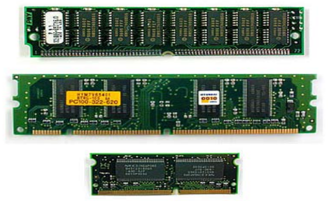
1

Course Administravia

- Workshop
  - Nov. 17, Nov. 19, Nov. 24, Dec. 1, Dec. 3
- Demo
  - Dec. 3 in the small conf room across KACB 3340, Prof. Liu's office.
- Technology Review
  - Dec. 9

2

Outline



- Summary of last week
  - In Memory Computing (SSD, Persistent Memory)
  - Inter-VM communication optimization using shared memory
- **This week**
  - In Memory Data Management + Memory Overcommitment in Virtualized Cloud
  - Big Data Computing: Outlook

3

IMDB - Introduction

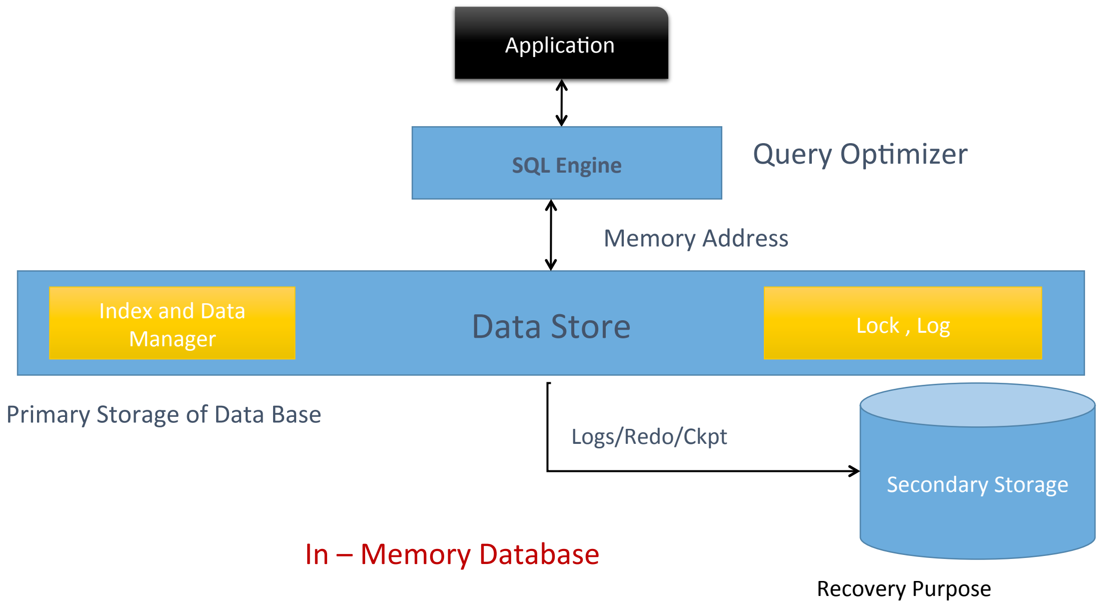
- What is In-Memory Data Base(IMDB)?

An **IMDB** also called **Main memory Database(MMDB)** is a database whose primary data store is main memory.

Why IMDB? 4 Factors:

- **LOWERING COSTS & GROWING SIZE (RAM):**
  - In early 2000, the cost of 64 MB RAM @ $71
  - But now , 8GB DDR3 @ $69.99
- **MULTICORE PROCESSORS**
  - parallel and faster computation
- **64 bit Computing**
  - multiple GB of main memory
- **Faster responses to queries**

Architecture



IMDB vs. DRDB(Disk Resident DB)

Disk Resident Data Base	**In-Memory Data Base**
Carries File I/O burden	No file I/O burden
Extra memory For Cache	No extra memory
Algorithm optimized for disk	Algorithms optimized for memory
More CPU cycles	Less CPU cycles
Assumes Memory is abundant	Uses memory more efficiently

Practical Application

- Applications that demand very fast data access, storage and manipulation
- In real-time embedded systems
- Music databases in MP3 players
- Programming data in set-top boxes
- e-commerce and social networking sites
- financial services and many more…

Redis: Main Memory NoSQL data store

NoSQL introduction

- NoSQL is a **non-relational** database management system, different from traditional RDBMS in some significant ways
- The NoSQL term should be used as in the **Not-Only-SQL**
    - not as **No to SQL** or **Never SQL**

Databases

- Classical variant - store data in a **relational** database and guarantee ACID properties
  - MySQL
  - PostgreSQL
  - H2
  - SQLite
  - HSQLDB
  - and many more...
- Modern trend in a Web programming: store data in **NoSQL** databases

ACID

- Atomicity
  - "all or nothing": if one part of the transaction fails, the entire transaction fails, and the database state is left unchanged
- Consistency
  - Ensures that any transaction will bring the database from one valid state to another. Any data written to the database must be valid according to all defined rules (constraints, cascades, triggers etc).

ACID

- Isolation
  - Ensures that the concurrent execution of transactions results in a system state that could have been obtained if transactions are executed serially
- Durability
  - Means that once a transaction has been committed, it will remain so, even in the event of power loss, crashes, or errors

NoSQL introduction

- Two trends:
  - The **exponential growth** of the volume of data generated by users and systems
  - The **increasing interdependency** and complexity of data, accelerated by the Internet, Web 2.0, social networks
- NoSQL databases are useful when working with a huge quantity of data and the data's nature does not require a relational model for the data structure

NoSQL characteristics

- Does not use SQL as its query language
  - NoSQL databases are not primarily built on tables, and generally do not use SQL for data manipulation
- May not give full ACID guarantees
  - Usually only **eventual consistency** is guaranteed or transactions limited to single data items
- Distributed, fault-tolerant architecture
  - Data are partitioned and held on large number of servers, and is replicated among these machines: **horizontal scaling**

Eventual consistency

- Given a sufficiently long period of time over which no changes are sent, all updates can be expected to **propagate eventually** through the system and all the replicas will be consistent.
- Conflict resolution:
  - **Read repair**: The correction is done when a read finds an inconsistency. This slows down the read operation.
  - **Write repair**: The correction takes place during a write operation, if an inconsistency has been found, slowing down the write operation.
  - **Asynchronous repair**: The correction is not part of a read or write operation.

BASE vs ACID

- BASE is an alternative to ACID
  - **B**asically **A**vailable
  - **S**oft state
  - **E**ventual consistency
- Weak consistency
- Availability first
- Approximate answers
- Faster

Scalability

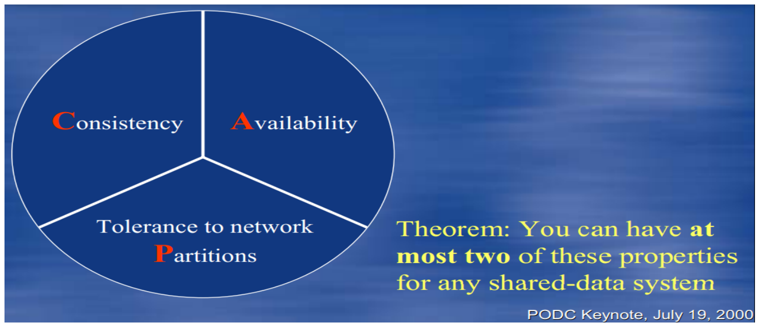
- **Scalability** is the ability of a system to handle a growing amount of work in an efficient manner
  - the ability to be enlarged to accommodate the system growth
- Scale horizontally (scale out)
  - Add more nodes to a system, such as adding a new computer to a distributed software application
- Scale vertically (scale up)
  - Add resources to a single node in a system, typically CPUs or memory

CAP theorem

- States that it is impossible for a distributed computer system to simultaneously provide all three of the following guarantees:
  - **Consistency**
    - All nodes see the same data at the same time
  - **Availability**
    - A guarantee that every request receives a response about whether it was successful or failed
  - **Partition tolerance**
    - The system continues to operate despite arbitrary message loss or failure of part of the system

Brewer's Conjecture

- In 2000, a conjecture was proposed in the keynote speech by Eric Brewer at the ACM Symposium on the Principles of Distributed Computing
- Slides: http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf

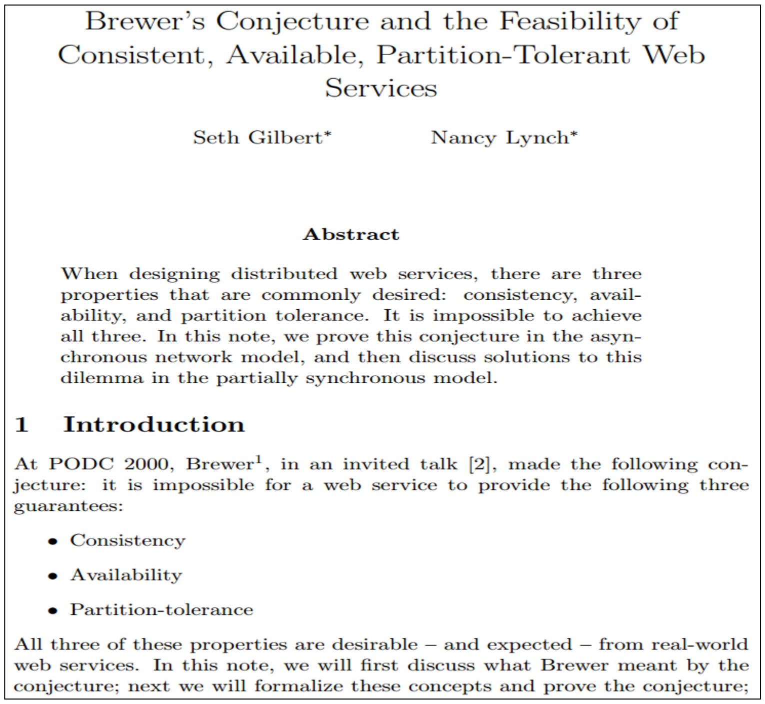


Theorem: You can have **at most two** of these properties for any shared-data system

PODC Keynote, July 19, 2000

Formal proof

- In 2002, Seth Gilbert and Nancy Lynch of MIT, formally proved Brewer to be correct
- http://lpd.epfl.ch/sgilbert/pubs/BrewersConjecture-SigAct.pdf
- "We have shown that it is impossible to reliably provide atomic, consistent data when there are partitions in the network."
- "It is feasible, however, to achieve any two of the three properties: consistency, availability, and partition tolerance."
- "In particular, most real-world systems today are forced to settle with returning "**most of the data, most of the time**.""



Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services

Seth Gilbert* Nancy Lynch*

Abstract

When designing distributed web services, there are three properties that are commonly desired: consistency, availability, and partition tolerance. It is impossible to achieve all three. In this note, we prove this conjecture in the asynchronous network model, and then discuss solutions to this dilemma in the partially synchronous model.

1 Introduction

At PODC 2000, Brewer[1], in an invited talk [2], made the following conjecture: it is impossible for a web service to provide the following three guarantees:

- Consistency
- Availability
- Partition-tolerance

All three of these properties are desirable – and expected – from real-world web services. In this note, we will first discuss what Brewer meant by the conjecture; next we will formalize these concepts and prove the conjecture;

Categories of NoSQL storages

- Key-Value
  - memcached
  - Redis
- Column Family
  - Cassandra
- Document
  - MongoDB
- Tabular
  - BigTable, HBase
- Graph, XML, Object, Multivalued,…

Redis

- Redis is an open source, advanced **key-value data store**
- Often referred to as a **data structure server** since keys can contain strings, hashes, lists, sets and sorted sets
- Redis works with an **in-memory** dataset
- It is possible to **persist** dataset either by
  - dumping the dataset to disk every once in a while
  - or by appending each command to a log

Who is using Redis?

twitter

Instagram

ask.fm

stackoverflow

BLIZZARD ENTERTAINMENT

Installation

Linux: http://redis.io/download

Windows

1. Clone from Git repo: https://github.com/MSOpenTech/redis
2. Unzip file from **`/redis/bin/release`** (e.g. **`redisbin64.zip`**) to **`/redis`**
3. Important files:
   - **`/redis/redis-server.exe`**
   - **`/redis/redis-cli.exe`**

Configuration

- Configuration file: **`/redis/redis.conf`**
- It is possible to change a port (if you wish):

```
port 6379
```

- For development environment it is useful to change data persisting policy

```
save 900 1
save 300 10
save 60 10000
```

$\longrightarrow$

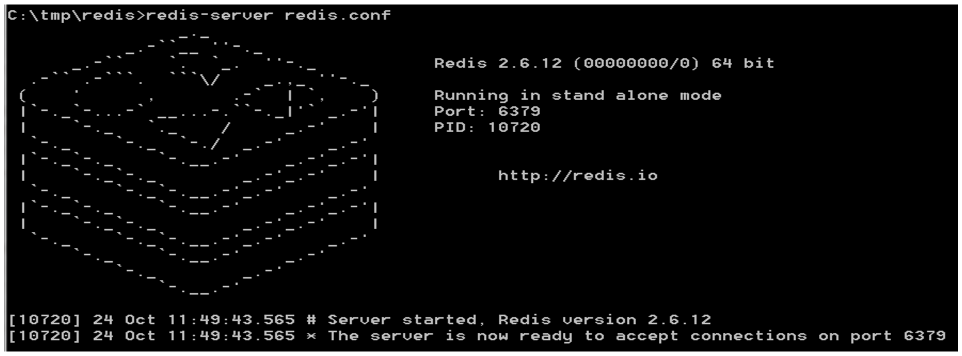
```
save 10 1
```

save after 10 sec if at least 1 key changed

Running Redis Server

- Run **`/redis/bin/redis-server.exe`** and specify configuration file to use

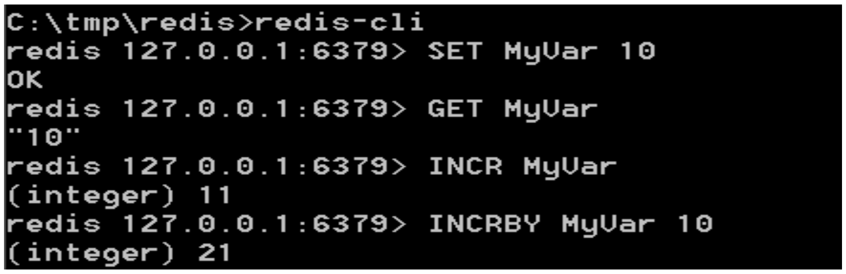
```
redis>redis-server redis.conf
```



```
C:\tmp\redis>redis-server redis.conf

                                   Redis 2.6.12 (00000000/0) 64 bit

                                   Running in stand alone mode
                                   Port: 6379
                                   PID: 10720

                                         http://redis.io

[10720] 24 Oct 11:49:43.565 # Server started, Redis version 2.6.12
[10720] 24 Oct 11:49:43.565 * The server is now ready to accept connections on port 6379
```

Running Redis Client

- Run **`/redis/bin/redis-cli.exe`**
- Now you can play with Redis a little bit



```
C:\tmp\redis>redis-cli
redis 127.0.0.1:6379> SET MyVar 10
OK
redis 127.0.0.1:6379> GET MyVar
"10"
redis 127.0.0.1:6379> INCR MyVar
(integer) 11
redis 127.0.0.1:6379> INCRBY MyVar 10
(integer) 21
```

Redis data types

Redis is often referred to as a **data structure server** since keys can contain:

- Strings
- Lists
- Sets
- Hashes
- Sorted Sets

Comparing Redis with other NoSQL stores

- using the YCSB - Yahoo! Cloud Serving Benchmark
- Default workloads are:
  - A (50% read and 50% update)
  - B (95% read and 5% update)
  - C (100% read)
  - D (95% read and 5% insert)
  - E (95% scan and 5% insert)
  - F (50% read and 50% read-modify-write)
  - H (1000 updates over 600,000 records)

Comparing Redis with other NoSQL stores

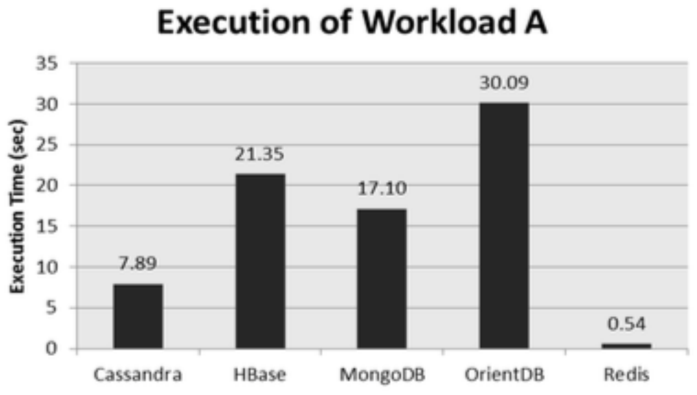


Figure 1: Execution time of workload A (50% reads and 50% updates over 600.000 records)

Comparing Redis with other NoSQL stores

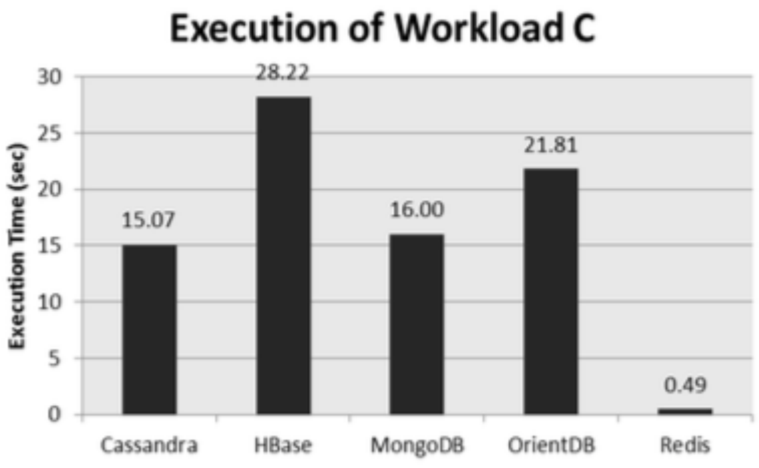


Figure 2: Execution time of workload C (100% reads over 600.000 records)

Comparing Redis with other NoSQL stores

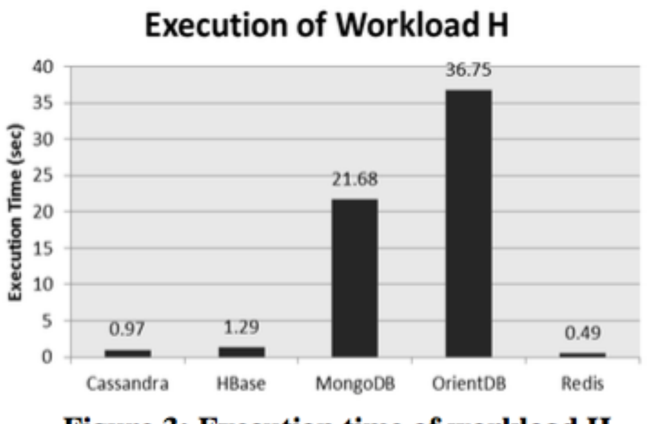


Figure 3: Execution time of workload H (100% update over 600.000 records)

Comparing Redis with other NoSQL stores

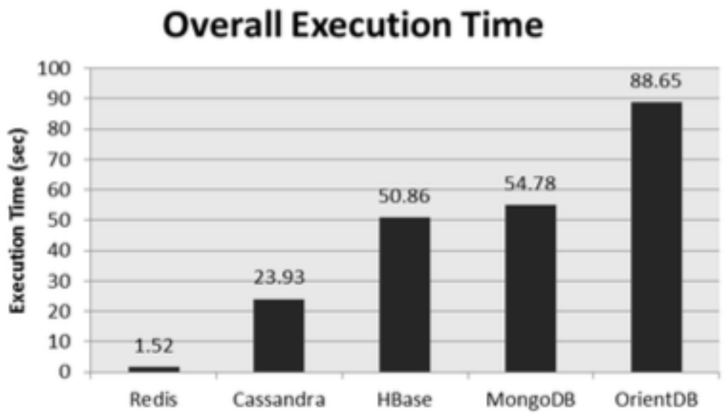


Figure 4: Overall execution time of workloads A+C+H

Using Redis in Java

- **JRedis** - Java Client for Redis
- **Jedis** - a blazingly small and sane Redis Java client
- **Spring Data Redis**

Web References

- "NoSQL -- Your Ultimate Guide to the Non - Relational Universe!"
  http://nosql-database.org/links.html
- "NoSQL (RDBMS)"
  http://en.wikipedia.org/wiki/NoSQL
- PODC Keynote, July 19, 2000. *Towards Robust. Distributed Systems*. Dr. Eric A. *Brewer*. Professor, UC Berkeley. Co-Founder & Chief Scientist, Inktomi .
  www.eecs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf
- "Brewer's CAP Theorem" posted by Julian Browne, January 11, 2009.
  http://www.julianbrowne.com/article/viewer/brewers-cap-theorem
- "How to write a CV" Geek & Poke Cartoon
  http://geekandpoke.typepad.com/geekandpoke/2011/01/nosql.html

37

Web References

- "Exploring CouchDB: A document-oriented database for Web applications", Joe Lennon, Software developer, Core International.
  http://www.ibm.com/developerworks/opensource/library/os-couchdb/index.html
- "Graph Databases, NOSQL and Neo4j" Posted by Peter Neubauer on May 12, 2010 at: http://www.infoq.com/articles/graph-nosql-neo4j
- "Cassandra vs MongoDB vs CouchDB vs Redis vs Riak vs HBase comparison", Kristóf Kovács. http://kkovacs.eu/cassandra-vs-mongodb-vs-couchdb-vs-redis
- "Distinguishing Two Major Types of Column-Stores" Posted by Daniel Abadi onMarch 29, 2010
  http://dbmsmusings.blogspot.com/2010/03/distinguishing-two-major-types-of_29.html

38

Web References

- "MapReduce: Simplified Data Processing on Large Clusters", Jeffrey Dean and Sanjay Ghemawat, December 2004. http://labs.google.com/papers/mapreduce.html
- "Scalable SQL", ACM Queue, Michael Rys, April 19, 2011 http://queue.acm.org/detail.cfm?id=1971597
- "a practical guide to noSQL", Posted by Denise Miura on March 17, 2011 at http://blogs.marklogic.com/2011/03/17/a-practical-guide-to-nosql/

39

Books

- "CouchDB *The Definitive Guide*", J. Chris Anderson, Jan Lehnardt and Noah Slater. O'Reilly Media Inc., Sebastopool, CA, USA. 2010
- "Hadoop *The Definitive Guide*", Tom White. O'Reilly Media Inc., Sebastopool, CA, USA. 2011
- "MongoDB *The Definitive Guide*", Kristina Chodorow and Michael Dirolf. O'Reilly Media Inc., Sebastopool, CA, USA. 2010

40