Overview on ZHT

Introduction to NoSQL databases and CS554 projects based on ZHT
Outlines

- General terms
- Overview to NoSQL databases and key-value stores
- Introduction to ZHT
- CS554 projects
Databases/datastores

- Relational databases
  - Query with SQL
  - DB2, MySQL, Oracle, SQL Server
  - CS 425, 525
- NoSQL databases
  - Loose consistency model
  - Simpler design
  - High performance
  - Distributed design
Categories in NoSQL

- Key-Value store
  - ZHT, Dynamo, Memcached, Cassandra, Chord
- Document Oriented Databases
  - MongoDB, Couchbase
- Graph databases
  - Neo4J, Allegro, Virtuoso
Key-value Stores

- Another name for Distributed Hash Table

Data

- Fox
  - Hash function
  - DFCD3454

- The red fox runs across the ice
  - Hash function
  - 52ED879E

- The red fox walks across the ice
  - Hash function
  - 46042841

Distributed Network

Peers
Zero-hop hash mapping
2-layer hashing

- **Server 2**
  - Replica 1
  - Replica 3
  - Member list
  - Persistent Hash table

- **Server 3**
  - Replica 2
  - Replica 4
  - Member list
  - Persistent Hash table

- **Server 4**
  - Replica 3
  - Replica 5
  - Member list
  - Persistent Hash table

**Client**

- Key
- Value

**Hash function**

**MOD num_nodes**
Consistency

- Updating membership tables
  - Planned nodes join and leave: strong consistency
  - Nodes fail: eventual consistency

- Updating replicas
  - Configurable
  - Strong consistency: consistent, reliable
  - Eventual consistency: fast, availability
Related work: Distributed Hash Tables

- Many DHTs: Chord, Kademlia, Pastry, Cassandra, C-MPI, Memcached, Dynamo ...
- Why another?

<table>
<thead>
<tr>
<th>Name</th>
<th>Impl.</th>
<th>Routing Time</th>
<th>Persistence</th>
<th>Dynamic membership</th>
<th>Append Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cassandra</td>
<td>Java</td>
<td>Log(N)</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>C-MPI</td>
<td>C</td>
<td>Log(N)</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Dynamo</td>
<td>Java</td>
<td>o to Log(N)</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Memcachted</td>
<td>C</td>
<td>0</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ZHT</td>
<td>C++</td>
<td>0 to 2</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Related projects

- ZHT Bench: Benchmarking mainstream NoSQL databases
- ZHT Cons: Eventual consistency support for ZHT
- ZHT DMHDFS: Distributed Metadata Management for the Hadoop File System
- ZHT Graph: Design and implement a graph database on ZHT
- ZHT OHT: Hierarchical Distributed Hash Tables
- ZHT ZST: Enhance ZHT through Range Queries and Iterators
Evaluation: test beds

- IBM Blue Gene/P supercomputer
  - Up to 8192 nodes
  - 32768 instance deployed
- Commodity Cluster
  - Up to 64 node
- Amazon EC2
  - M1.medium and Cc2.8xlarge
  - 96 VMs, 768 ZHT instances deployed
Genera requirements

- Familiar with Linux and its command line
- Shell scripting language (e.g. Bash, zsh...)
- Programming skills in C++/C (except benchmark)
- GCC compiler
- No object oriented skill needed
Goal: Extensively benchmarking NoSQL databases and analysis performance data.
- ZHT, MongoDB, Cassandra
- Neo4J (experiment for Graph)
- And others...

Metrics
- Latency and its distribution, throughput

Parameters
- Message size
- Scales
- Key Distributions
Goal 1: allow replicas serve read operation
Goal 2: maintain eventual consistency between replicas
Goal 3: make it scale (pretty hard!)

Optional goal: allow replicas serve write requests and maintain consistency (applying Paxos protocol, even harder)
What is metadata?

Goal: improve HDFS performance by adding distributed metadata service

Requirement: experience with Hadoop and HDFS; strong programming skill in both Java and C++
ZHT Graph: Design and implement a graph database on ZHT

- Goal: build a graph databases on top of ZHT
- How: construct a mapping from key-value store interface to graph interface
Goal: adding a proxy level to ZHT architecture so to reduce concurrency stress to each server
- Easy: make it work and scale
- Hard: handle failures
Goal: design and implement new interface methods to ZHT

- Iterator: next/previous operation
- Range get/put: given a range of key, return a series of results in one request loop

How?

- Sorted map
- B+ tree (bold!)
What do I expect?

- Communication: come and talk to me (by appointment)
- Make good use of Google
- Fail quick, fail early, fail cheap.
- Fast iteration: very small but frequent progress

- Why bother? 80% points from projects!
Welcome abroad and enjoy!

Tonglin Li
tli13@hawk.iit.edu
http://datasys.cs.iit.edu/projects/ZHT/