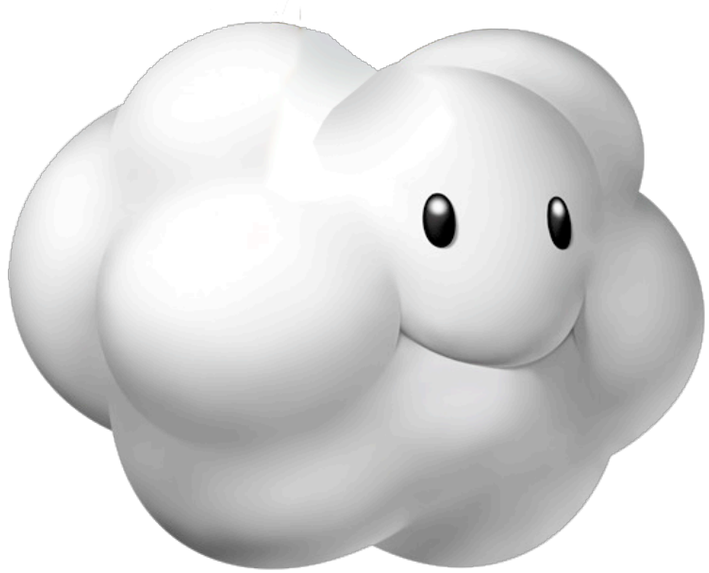


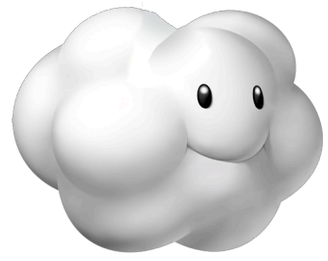
Accelerating NoSQL

Running Voldemort on HailDB

Sunny Gleason
March 11, 2011

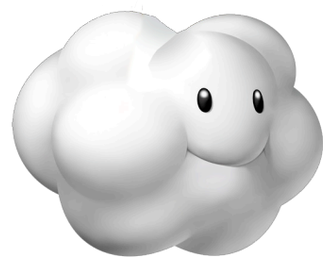
whoami

- Sunny Gleason, human
- passion: distributed systems engineering
- previous...
  Ning : custom social networks
  Amazon.com : infra & web services
- now...
  building cloud infrastructure



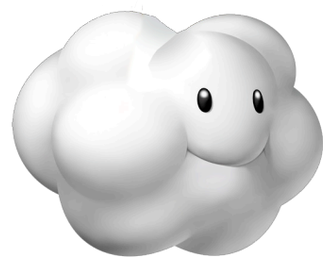
whereami

- twitter : twitter.com/sunnygleason
- github : github.com/sunnygleason
- linkedin : linkedin.com/in/sunnygleason



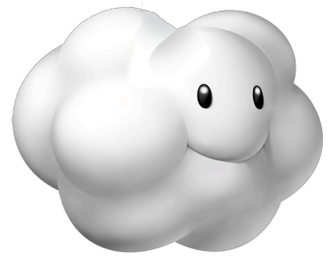
what's in this presentation?

- NoSQL Roundup
- Voldemort who?
- HailDB wha?
- Results & Next Steps
- Special Bonus Material



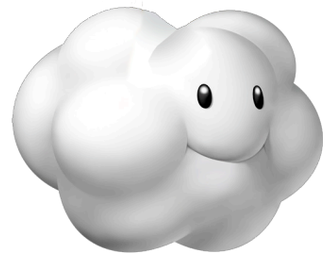
NoSQL

- "Not Only" SQL
- What's the point?
- Proponent: "reaching next level of scale"
- Cynic: "cloud is hype, ops nightmare"



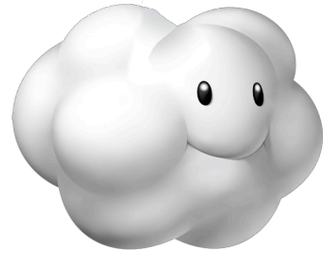
what does it gain?

- Higher performance, scalability, availability
- More robust fault-tolerance
- Simplified systems design
- Easier operations

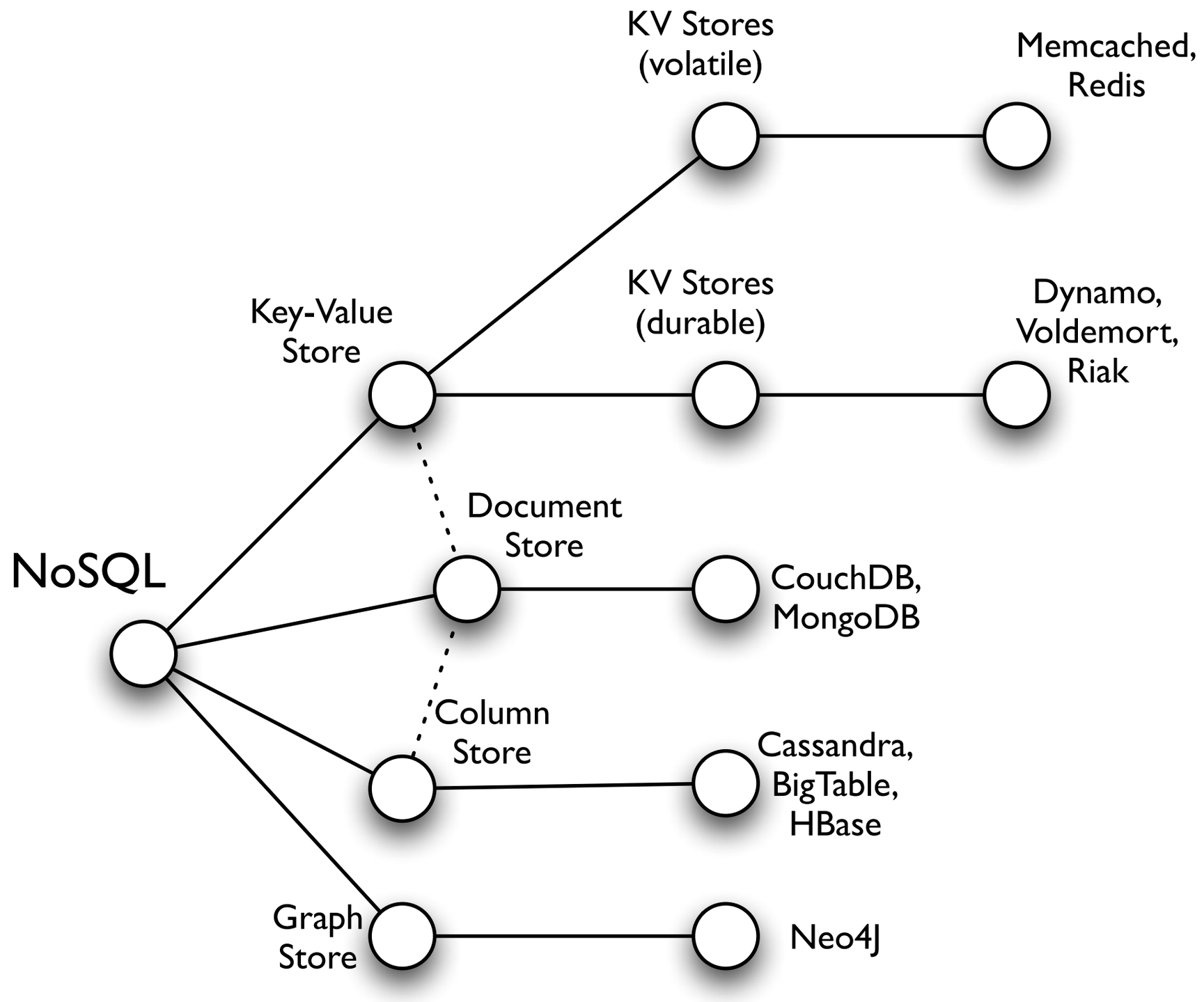


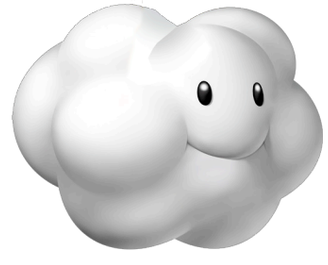
what does it lose?

- Reduced / simplified programming model
- No ad-hoc queries, no joins, no txns
- Not ACID: Atomicity / Consistency / Isolation / Durability
- Operations / management is still evolving
- Challenging to quantify health of system
- Fewer domain experts

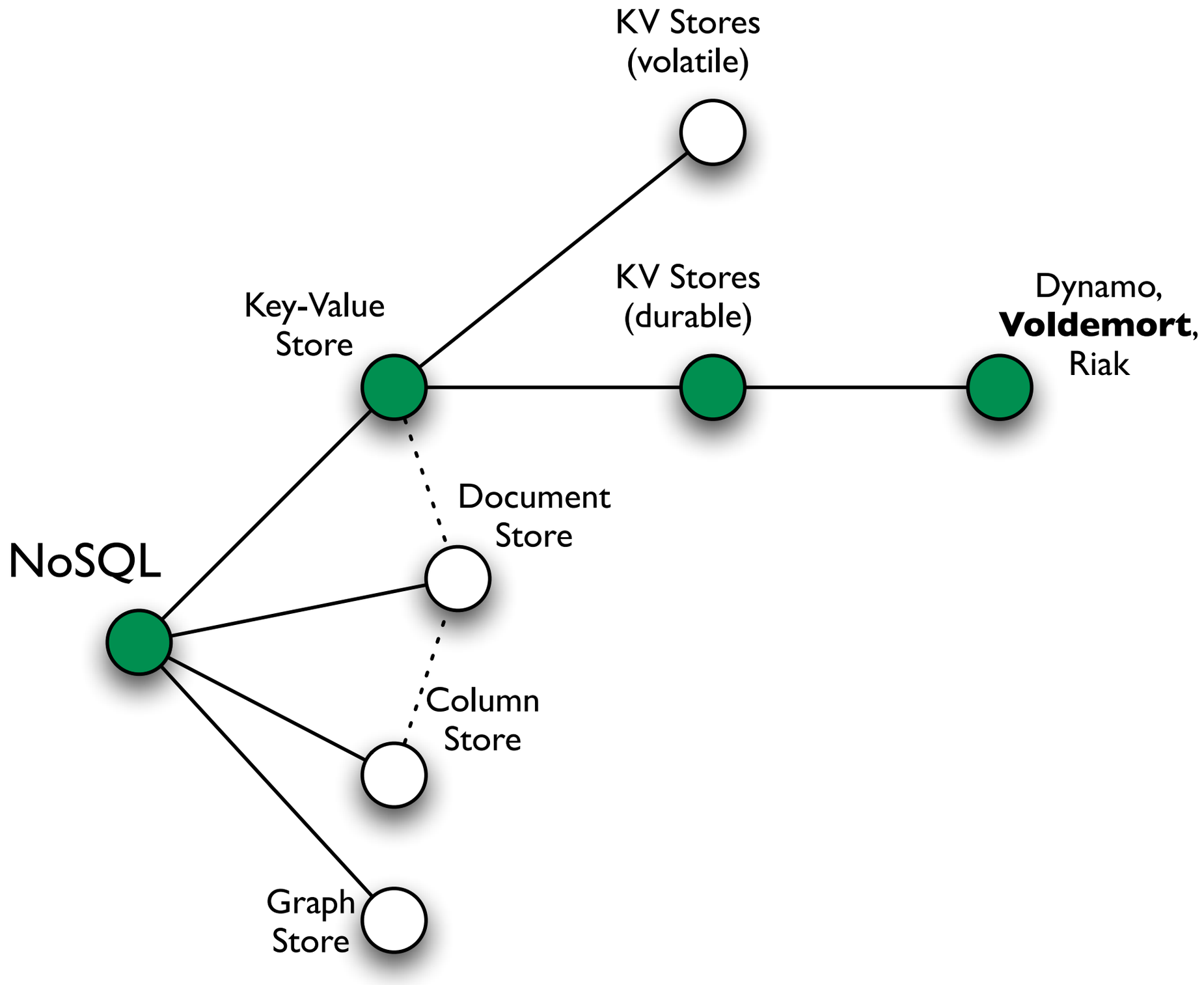


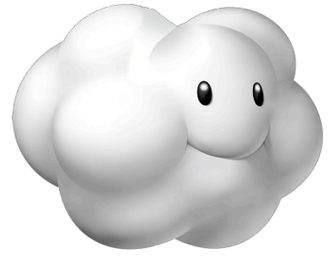
NoSQL Map


KV Stores
(volatile)
Memcached,
Redis
Key-Value
Store
KV Stores
(durable)
Dynamo,
Voldemort,
Riak
Document
Store
NoSQL
CouchDB,
MongoDB
Column
Store
Cassandra,
BigTable,
HBase
Graph
Store
Neo4J



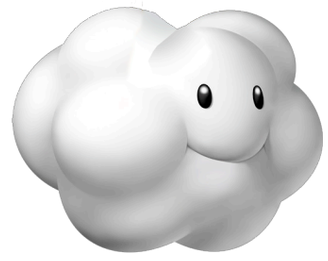
NoSQL Map


KV Stores
(volatile)
KV Stores
(durable)
Key-Value
Store
Dynamo,
Voldemort,
Riak
Document
Store
NoSQL
Column
Store
Graph
Store

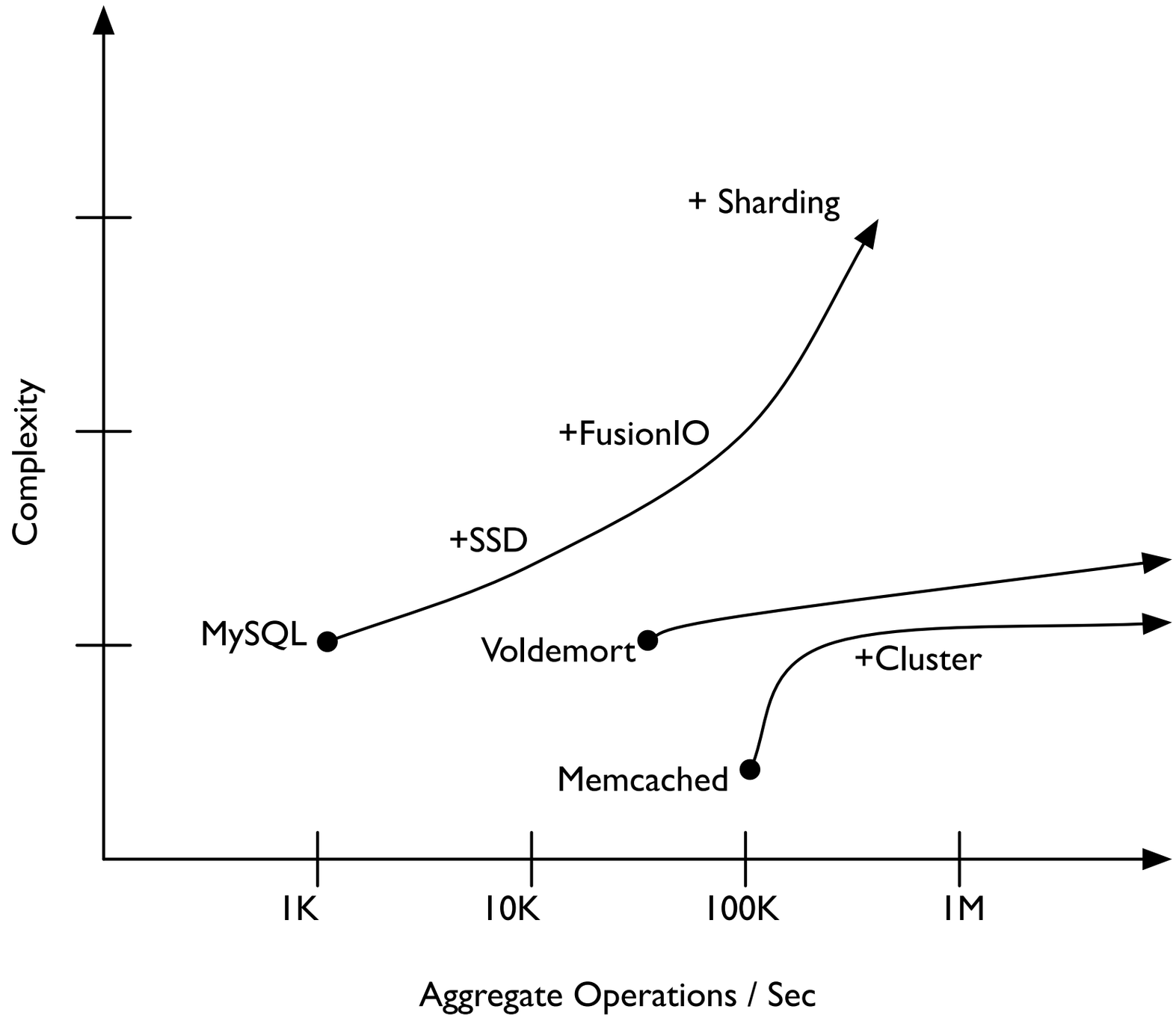


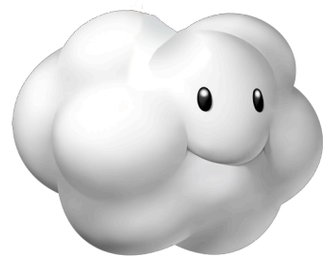
motivation

- database on 1 box : ok
- database with master/slave replication : ok
- database on cluster : tricky
- database on SAN : time bomb



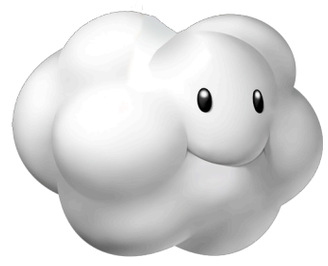
performance





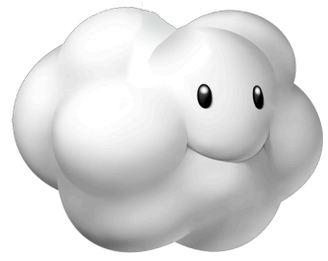
dynamo case study

- Amazon : high read throughput, always-accessible writes
- Shopping cart application
- ‘Glitches’ ok, duplicate or missing item
- Data loss or unavailability is unacceptable
- Solution: K-V schema plus smart routing & data placement



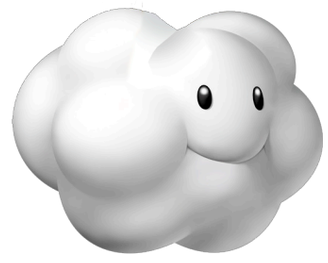
key-value storage

- Essentially, a gigantic hash table
- Typically assign byte[] values to byte[] keys
- Plus versioning mixed in to handle failures and conflicts
- Yes, you *can* do range partitioning; in practice, avoid it because of hot spots



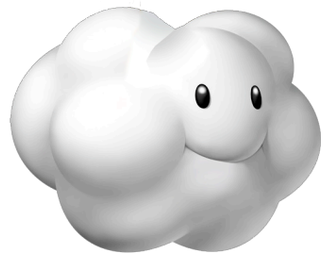
k-v: durable vs. volatile

- RAM is ridiculous speed (ns), not durable
- Disk is persistent and slow (3-7ms)
- RAID eases the pain a bit (4-8x throughput)
- SSD is providing good promise (100-300us)
- FusionIO is redefining the space (30-100us)



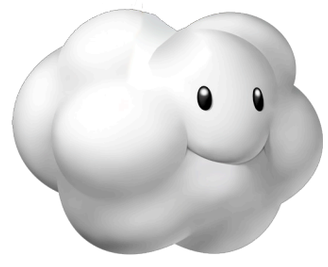
dynamo clones

- Voldemort : from LinkedIn, dynamo implementation in Java (default: BDB-JE)
- Riak : from Basho, dynamo implementation in Erlang (default: embedded InnoDB)



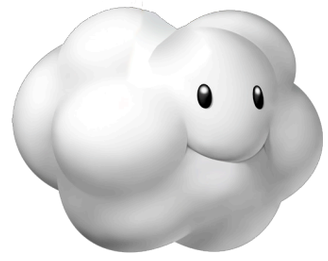
Voldemort

- Developed at LinkedIn
- Scalable Key-Value Storage
- Based on Amazon Dynamo model
- High Read Throughput
- Always Writable

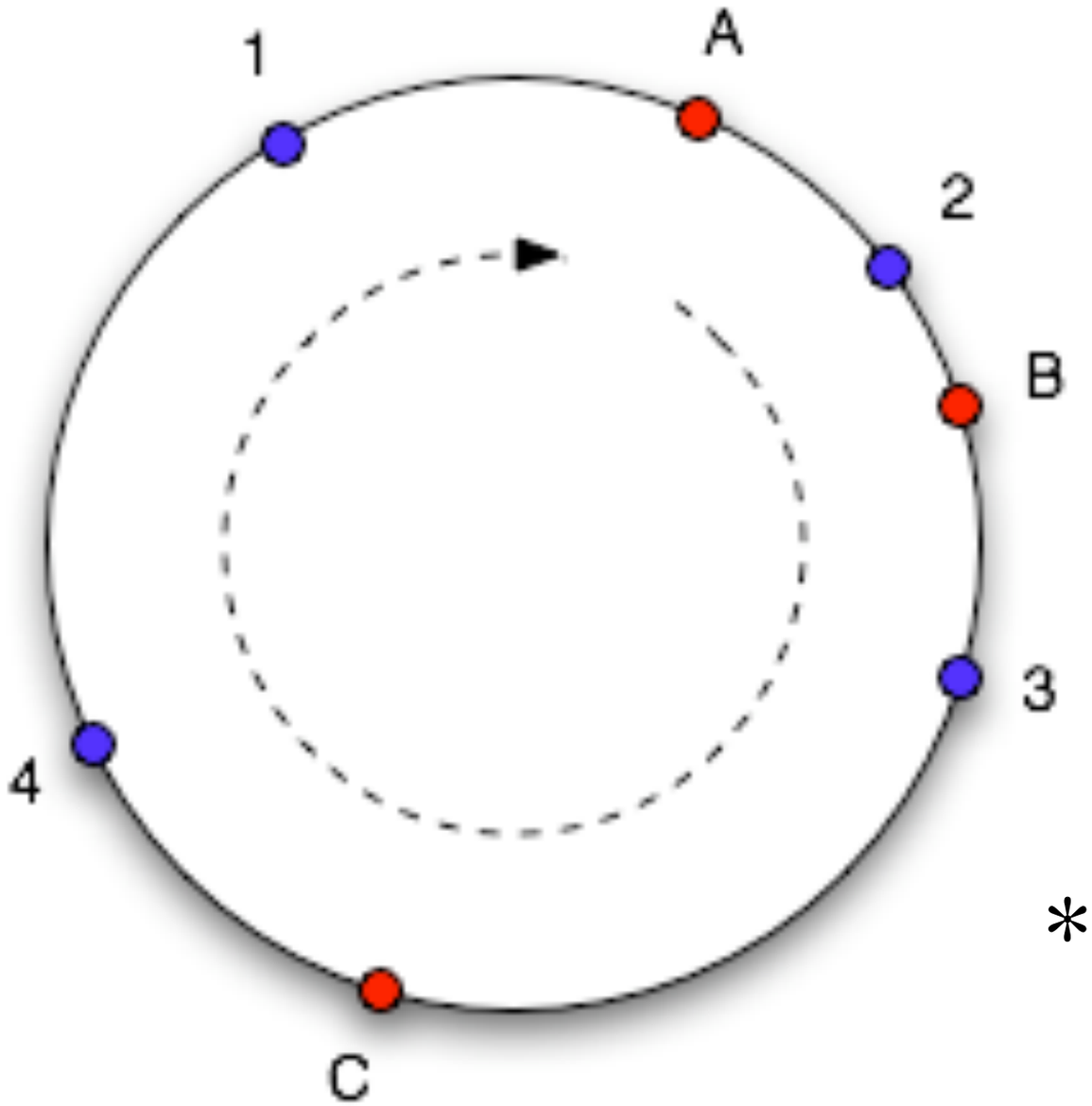


Voldemort features

- Consistent Hashing
- Quorum settings : R, W, N
- Auto-sharding & rebalancing
- Pluggable storage engines



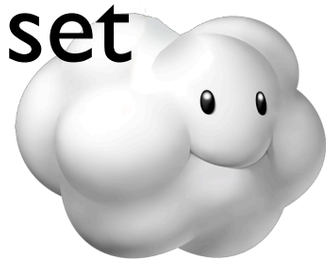
Consistent Hashing



* Arrange keys around ring

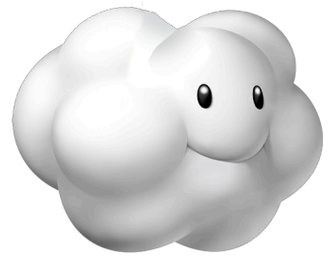
* Compute token in ring using hash function

* Determine nodes responsible for token using live set



R/W/N

- N : maximum number of nodes to query for an operation
- R : read quorum
- W : write quorum
- Can adjust ‘quorum’ to balance throughput and fault-tolerance



setting up Voldemort !

Step 1: Download the code

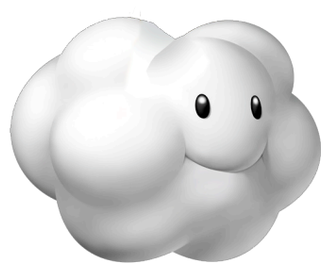
Download either a recent stable release or, for those who like to live more dangerously, the up-to-the-minute build from the build server.

Step 2: Start single node cluster

```
> bin/voldemort-server.sh config/single_node_cluster > /tmp/voldemort.log &
```

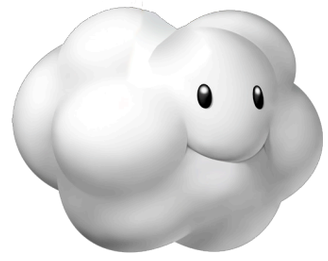
Step 3: Start commandline test client and do some operations

```
> bin/voldemort-shell.sh test tcp://localhost:6666
Established connection to test via tcp://localhost:6666
> put "hello" "world"
> get "hello"
version(0:1): "world"
> delete "hello"
> get "hello"
null
> exit
k k thx bye.
```



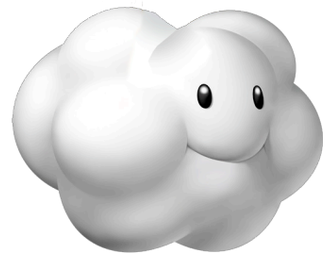
setting up Voldemort 2

- For a cluster, use cloud startup scripts
- Works with Amazon EC2
- See https://github.com/voldemort/voldemort/wiki/EC2-Testing-Infrastructure



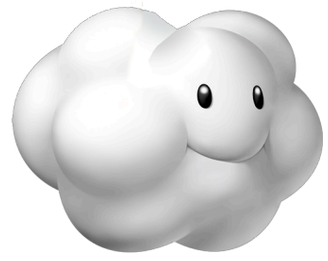
Voldemort client libraries

- Java, Scala, Clojure
- Ruby
- Python
- C++



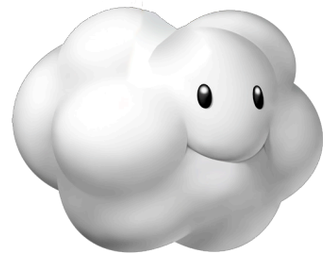
storage engines

- BDB-JE (Oracle Sleepycat, the original)
- Krati (LinkedIn, pretty new)
- HailDB (new!)
- MySQL (old / dated)



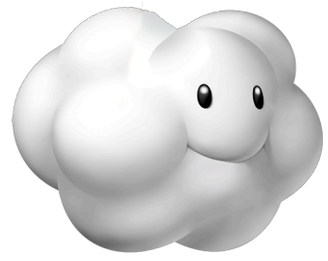
BDB-JE

- Log-Structured B-Tree
- Fast Storage When Mostly Cached
- Configured without fsync() by default - writes are batched and flushed periodically



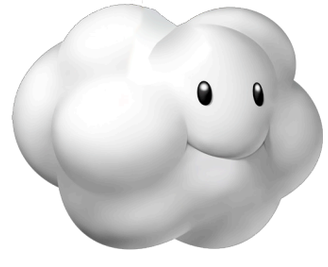
Krati

- Fast Hash-Oriented Storage
- Uses memory-mapped files for speed
- Configured without fsync() by default - writes are batched and flushed periodically

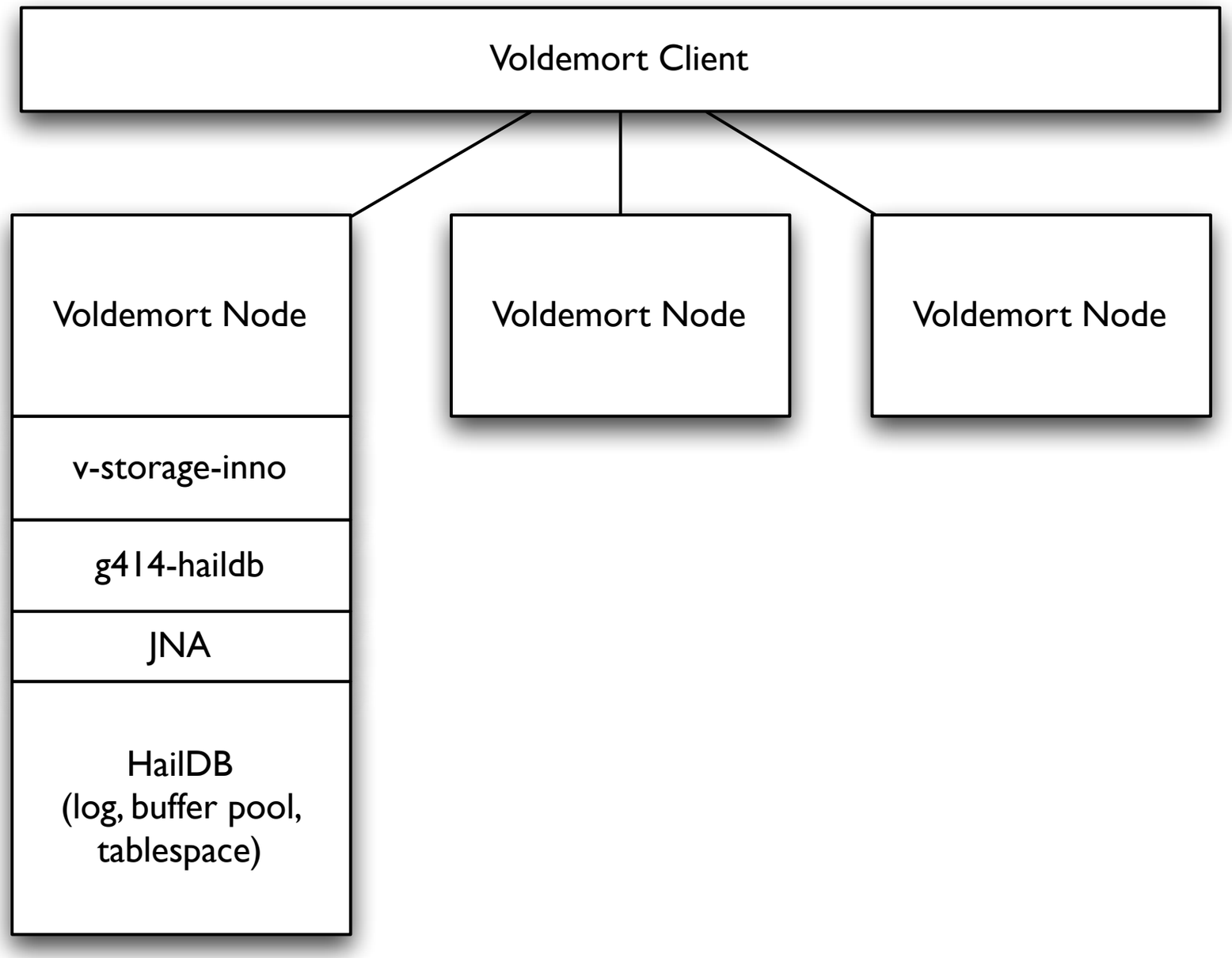


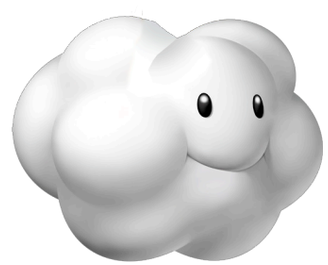
HailDB

- Fork of MySQL InnoDB plugin (contributors : Oracle, Google, Facebook, Percona)
- Higher stability for large data sets
- Fast crash recovery
- External from Java heap (ease GC pain)
- apt-get install haildb (from launchpad PPA)
- Use “flush-once-per-second” mode



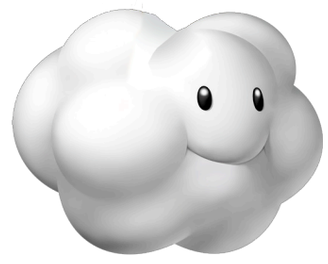
HailDB, Java & Voldemort





HailDB & Java

- g414-haildb : where the magic happens
- uses JNA: Java Native Access
- dynamic binding to libhaildb shared library
- auto-generated from .h file (w/ JNAerator)
- Pointer classes & other shenanigans



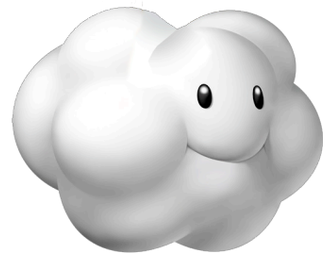
HailDB schema

_key VARBINARY(200)

_version VARBINARY(200)

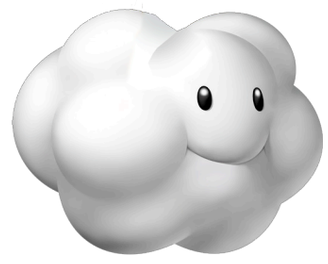
_value BLOB

PRIMARY KEY(_key, _version)



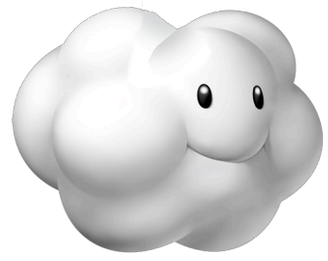
implementation gotchas

- InnoDB API-level usage is unclear
- Synchronization & locking is unclear
- Therefore... I learned to love reading C
- Error handling is *nasty*
- Installation a bit of a pain

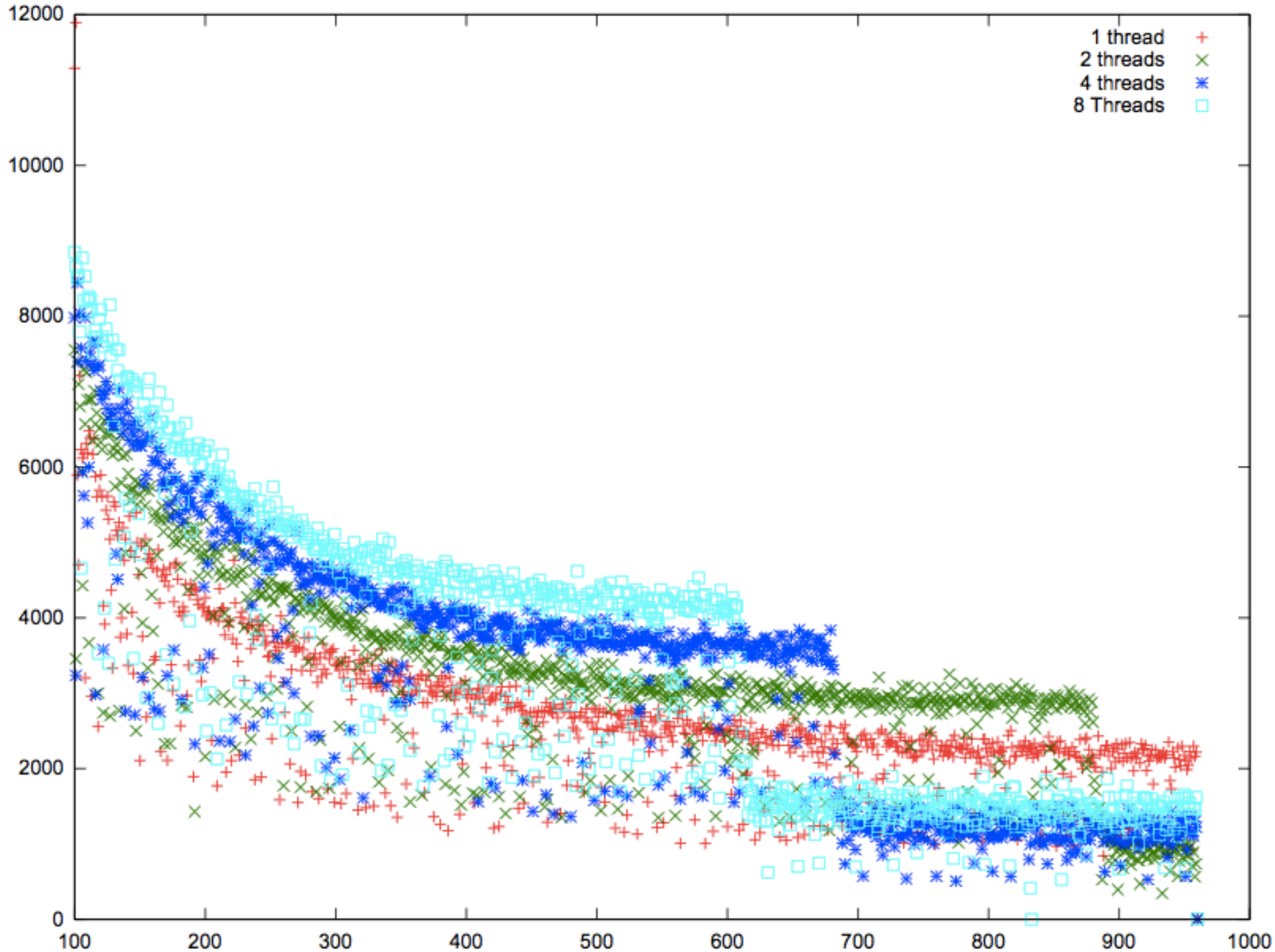


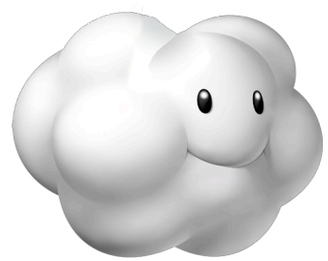
experimental setup

- OS X: 8-Core Xeon, 32GB RAM, 200GB OWC SSD
- Faban Benchmark : PUT 64-byte key, 1024-byte value
- Scenarios: 1, 2, 4, 8 threads
- 512M Java Heap

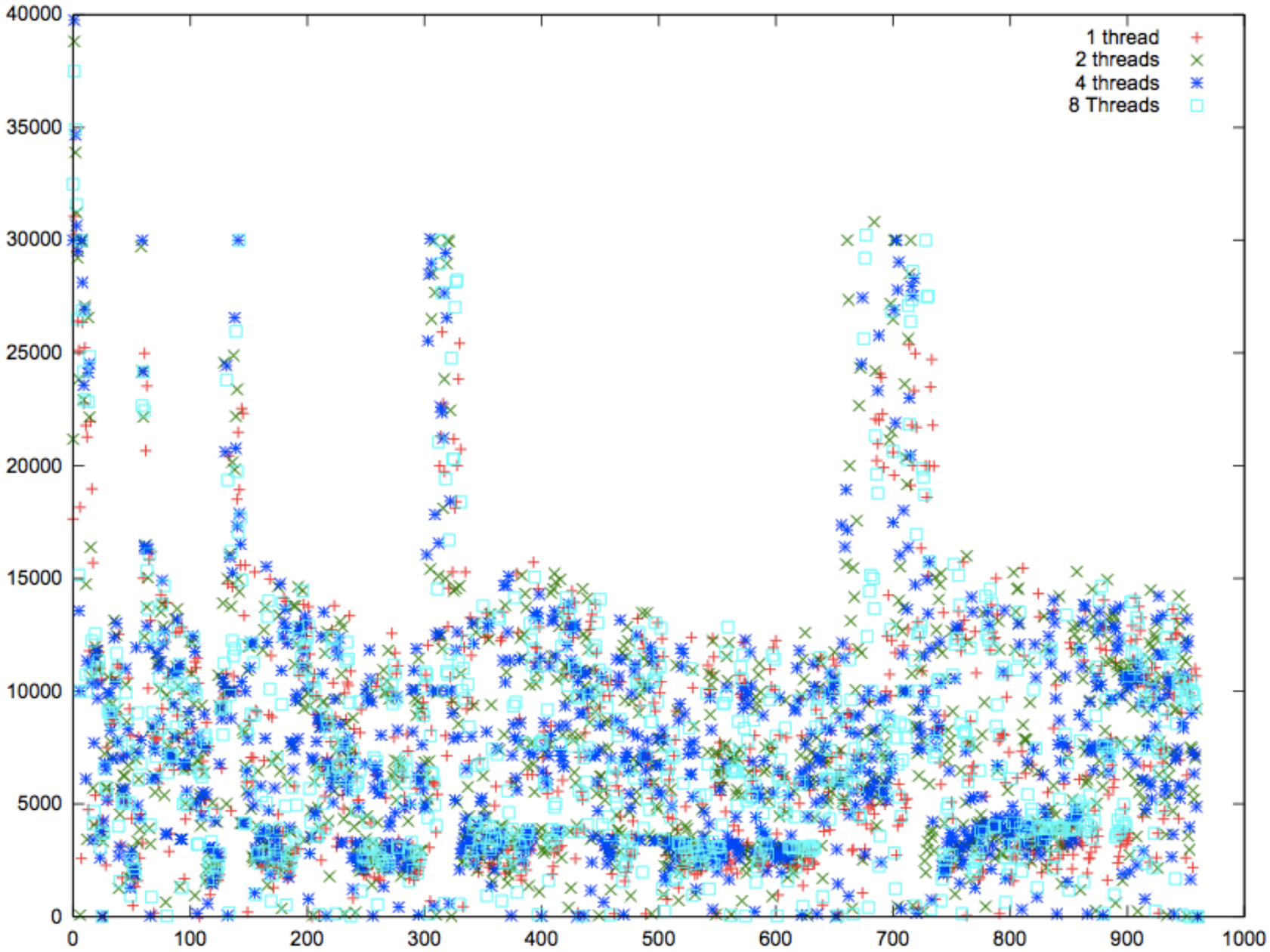


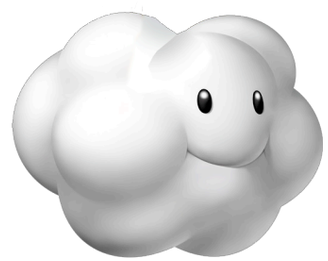
Perf: BDB Put


12000
10000
8000
6000
4000
2000
0
100
200
300
400
500
600
700
800
900
1000
1 thread
2 threads
4 threads
8 Threads

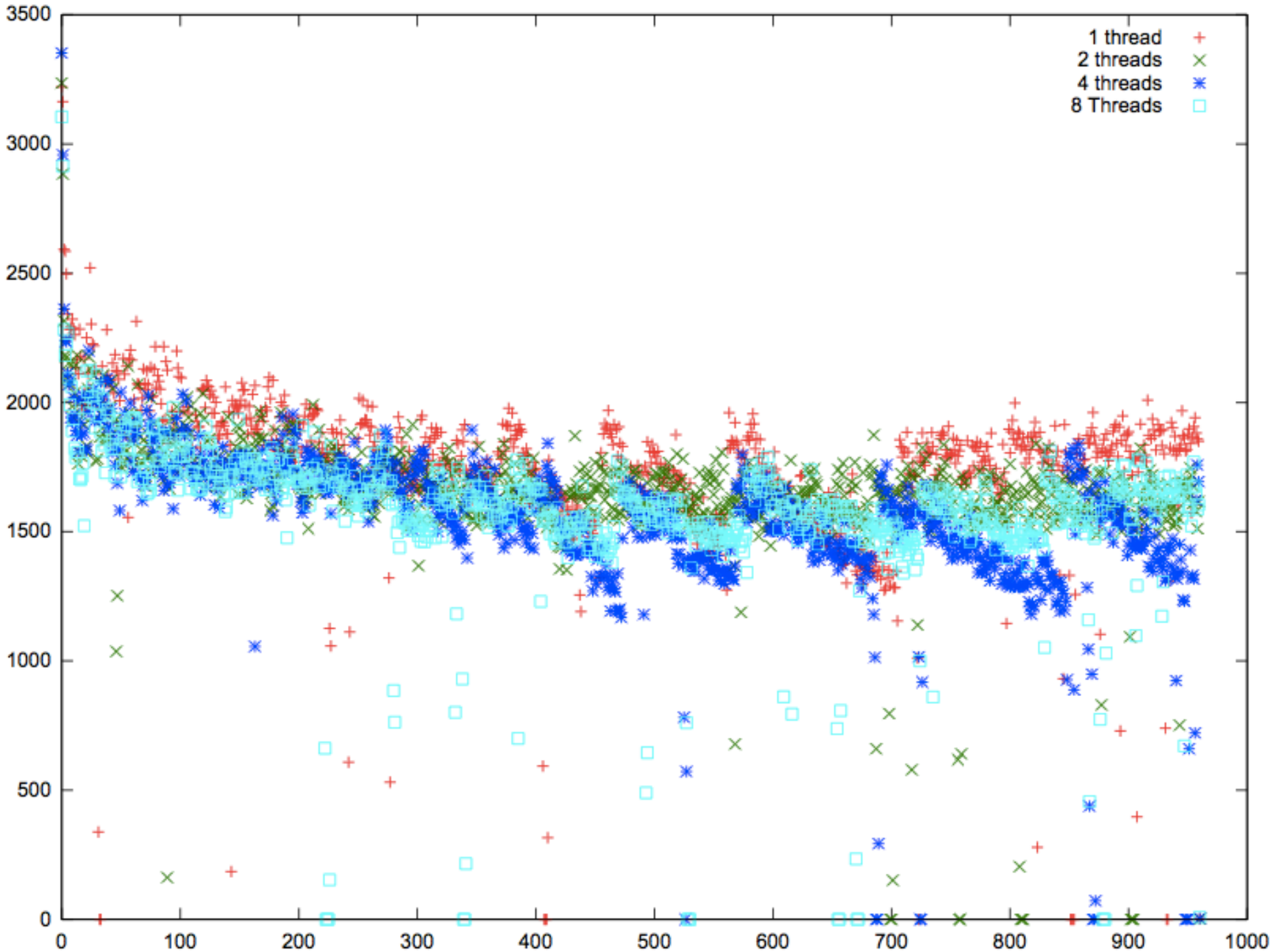


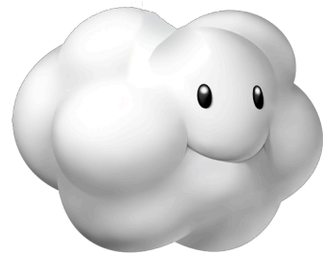
Perf: Krati Put





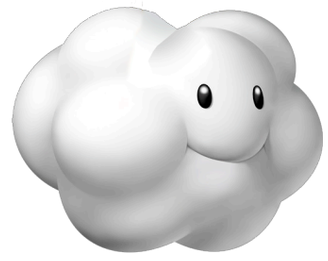
Perf: HailDB Put





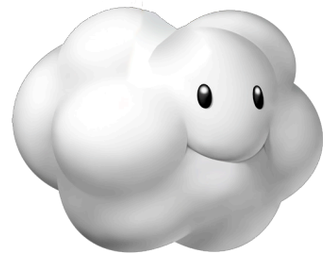
future work

- Improve Packaging / Installation
- Schema refinements & perf enhancements
- Online backup/export with XtraBackup
- JNI Bindings



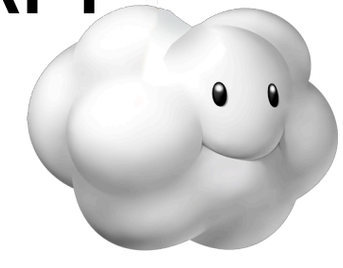
schema refinements

- Build upon Nokia work on fast k-v schema
- 8-byte 'long' key hash vs. full key bytes
- Smart use of secondary indexes
- Native representation of vector clocks
- Delayed / soft deletion
- Expect 40-50% performance boost



InnoDB tuning

- Skinny columns, skinny rows! (esp. Primary Key)
  - Varchar enum ‘bad’, int or smallint ‘good’
  - fixed-width rows allows in-place updates
- Use covering indexes strategically
- More data per page means faster index scans, more efficient buffer pool utilization
- You only get so many trx’s on given CPU/RAM configuration - benchmark this!



refined schema

_id BIGINT (auto increment)

_key_hash BIGINT

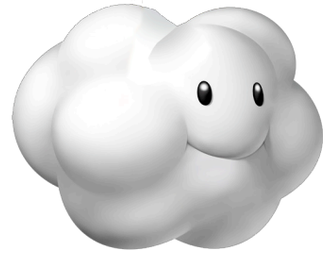
_key VARBINARY(200)

_version VARBINARY(200)

_value BLOB

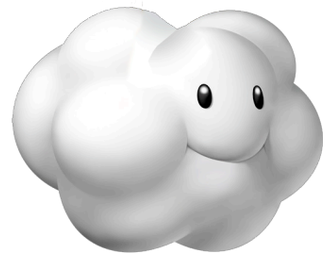
PRIMARY KEY(_id)

KEY(_key_hash)



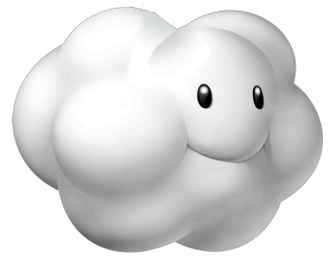
online backup

- hot backup of data to other machine / destination
- test Percona Xtrabackup with HailDB
- next step: backup/export to Hadoop/HDFS (similar to Cloudera Sqoop tool)



JNI bindings

- JNI can get 2-5x perf boost vs. JNA
- ... at the expense of nasty code
- Will go for schema optimizations and InnoDB tuning tips *first*

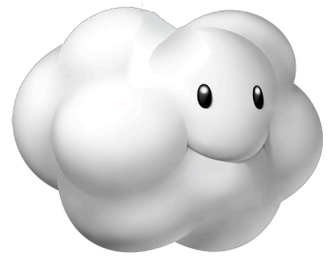


resources

- github.com/voldemort/voldemort
  freenode #voldemort

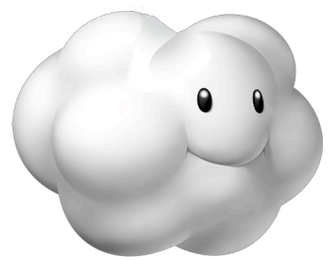
- github.com/sunnygleason/v-storage-haildb
  github.com/sunnygleason/v-storage-bench
  github.com/sunnygleason/g414-haildb

- jna.dev.java.net



more resources

- Amazon Dynamo
- Faban / XFaban
- HailDB
- Drizzle
- PBXT



Thank You!