Hibari/Erlang/ NOSQL for BIGDATA

November 2010

Gemini Mobile Technologies

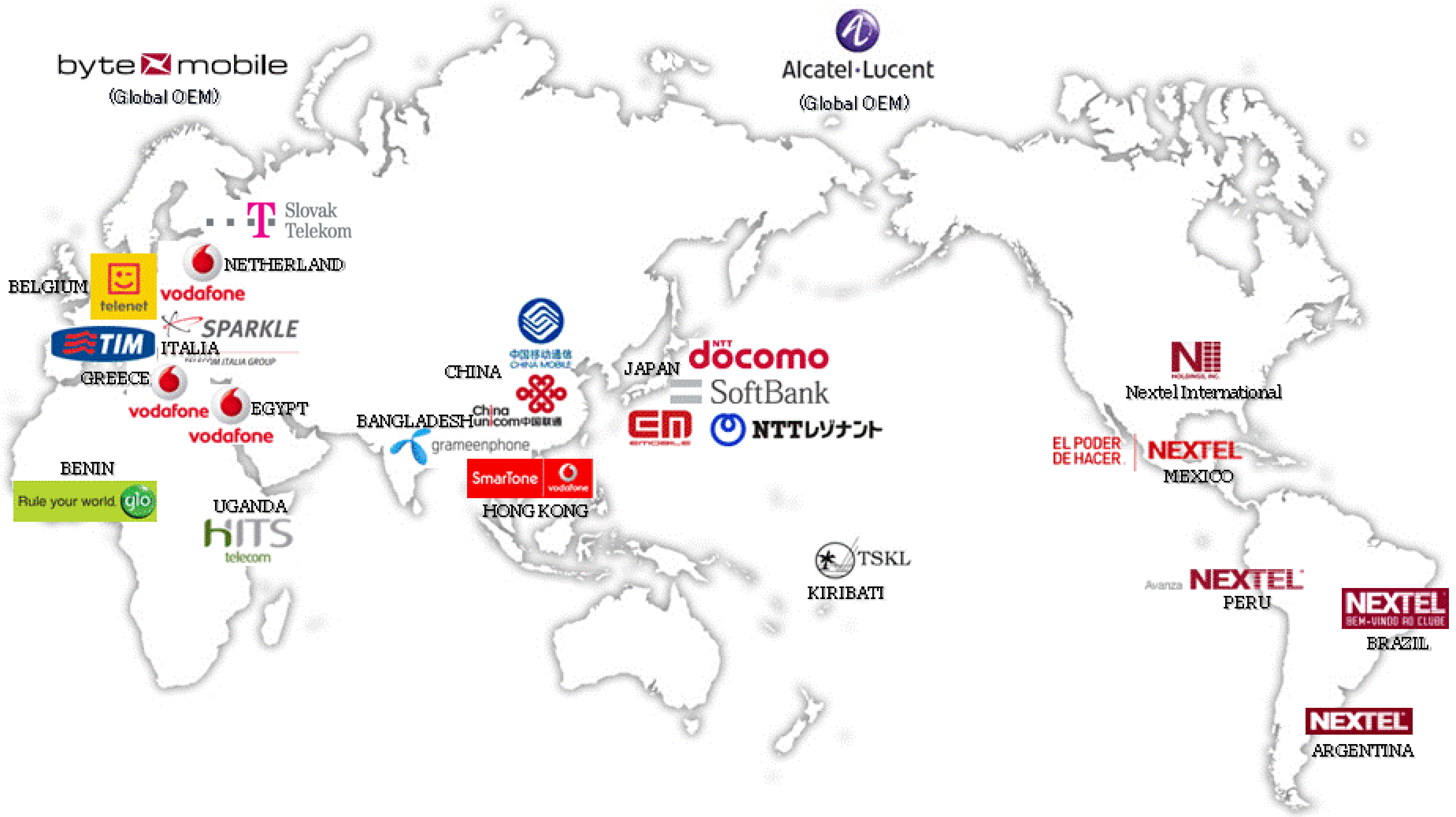
Hibari Open Source project: http://sourceforge.net/projects/hibari/

Introduction

GEMINI
Mobile Technologies

- Founded: July, 2001
- Offices: San Francisco, Tokyo, Beijing
- Investors:
  - Goldman Sachs, Mitsubishi-UFJ, Mizuho, Nomura, Ignite, Access, Aplix
- Accomplishments:
  - Messaging Products
    - Provide MMSC to 3 out of 4 Carriers in Japan (DoCoMo, Softbank, eMobile)
    - Largest MMSC in the world (Softbank Japan)
    - OEM to Alcatel-Lucent and ByteMobile
  - NOSQL / Big Data
    - 2006: First Mobile 3D SNS (Softbank, China Unicom, iPhone App)
    - 4/2010: WebMail, Japanese Mobile Carrier & Internet Provider
    - 7/2010: Hibari Open Source

Customers



GEMINI

Hibari (= Cloud Birds)

What is Hibari?

- Hibari is a production-ready, distributed, key-value, big data store.
  - China Mobile and China Unicom - SNS
  - Japanese internet provider - GB mailbox webmail
  - Japanese mobile carrier - GB mailbox webmail
- Hibari uses chain replication for strong consistency, high-availability, and durability.
- Hibari has excellent performance especially for read and large value operations.
- Hibari is open-source software under the Apache 2.0 license.

Environments

- Hibari runs on commodity, heterogeneous servers.
- Hibari supports Red Hat, CentOS, and Fedora Linux distributions.
  - Debian, Ubuntu, Gentoo, Mac OS X, and Free BSD are coming soon.
- Hibari supports Erlang/OTP R13B04.
  - R14B is coming soon.
- Hibari supports Amazon S3, JSON-RPC-RFC4627, UBF/EBF/JSF and native Erlang client APIs.
  - Thrift API was open sourced last week.

Why NOSQL?

We needed to build a scalable, high performance web mail system

- Big Data
  - Several million end users from the start
  - Several billion messages in a few months
  - Hundreds of TB data

- Low Cost requirements
  - Customer's business model (Freemium)
  - Distributed >50 PC servers
  - No need for rich and expensive functions of SQL
- Continuous growth of data in the storage
  - Elasticity to expand capacity due to increasing data

What were the customer’s needs?

- Durability
  - Data loss (e.g., messages, metadata) is not acceptable
- Strong Consistency
  - Because of interactive sessions, consistency is required
- Low Latency
  - <1 sec response time for end user transactions
- High Availability
  - As a branded service to the end user, service must always be available.
- Read Heavy
  - Many more read than write operations
- Big Data and Data size highly variable
  - Large GB mail box as service differentiator was required
  - Mail messages range from a few bytes to many MB with attachments

How does Hibari address these needs?

- Durable updates

  Every update is written and flushed to stable storage (fsync() system call) before sending acknowledgments to the client.

- Consistent updates

  After an update is acknowledged, no client can see an older version. ”Chain Replication” is used to maintain consistency across all replicas.

- High Availability

  Each key can be replicated multiple times. As long as one copy of the key survives, all operations on that key are permitted.

- Lockless API

  Locks are not required for all client operations. Optionally, Hibari supports “test-and-set” of each key-value pair via an increasing (enforced by the server) timestamp value.

- Micro-transactions

  Under limited circumstances, operations on multiple keys can be given transactional commit/abort semantics.

Why Erlang?

- Concurrency and Distribution
- Robustness
- Efficient garbage collection
- Hot code and incremental upgrade
- Online tracing
- Efficiency and Productivity
  - **Small teams make big impact**
- Ericsson's support of Erlang/OTP is wonderful

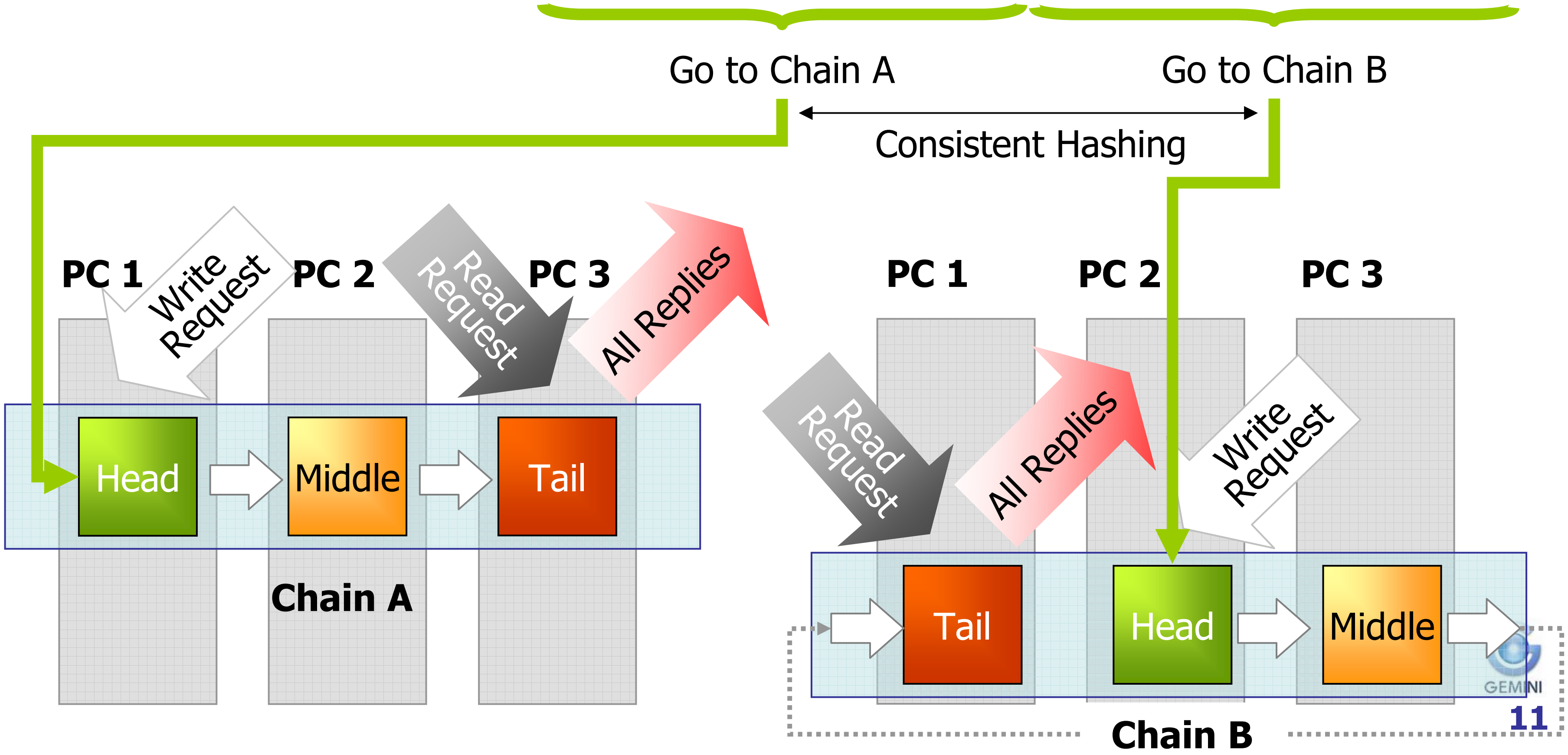
Everything you need to build robust, high performance distributed systems

Chain Replication for Strong Consistency

	Key	Values
Data Model	0…………n	BLOB (Binary Large OBject)

Key Value Table

Key 1	Key 2	Key 3	·	·	·	·	Key n
Value	Value	Value	·	·	·	·	Value



Go to Chain A

Go to Chain B

Consistent Hashing

PC 1

Write Request

PC 2

Read Request

PC 3

All Replies

Head

Middle

Tail

Chain A

PC 1

PC 2

PC 3

Read Request

All Replies

Write Request

Tail

Head

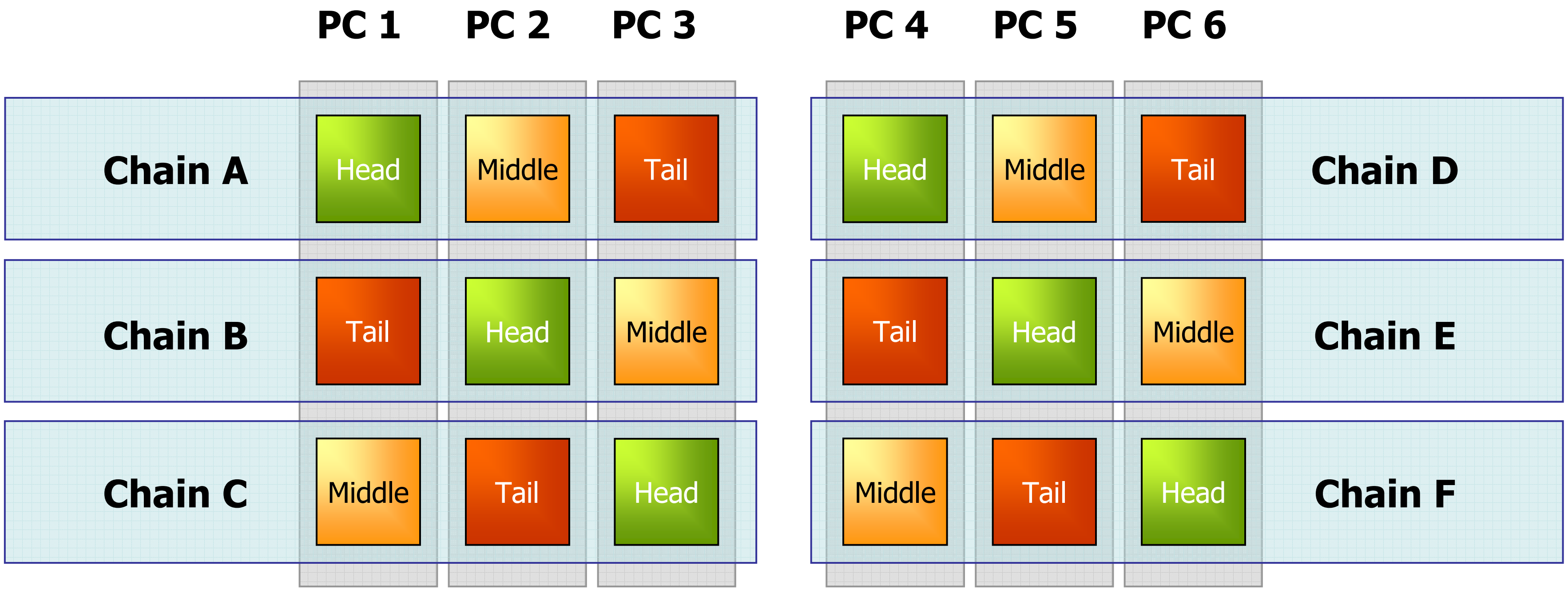
Middle

Chain B

Concept of “Chain”

Evenly distributed load in multiple nodes

(Case of 3 replications/6 chains)

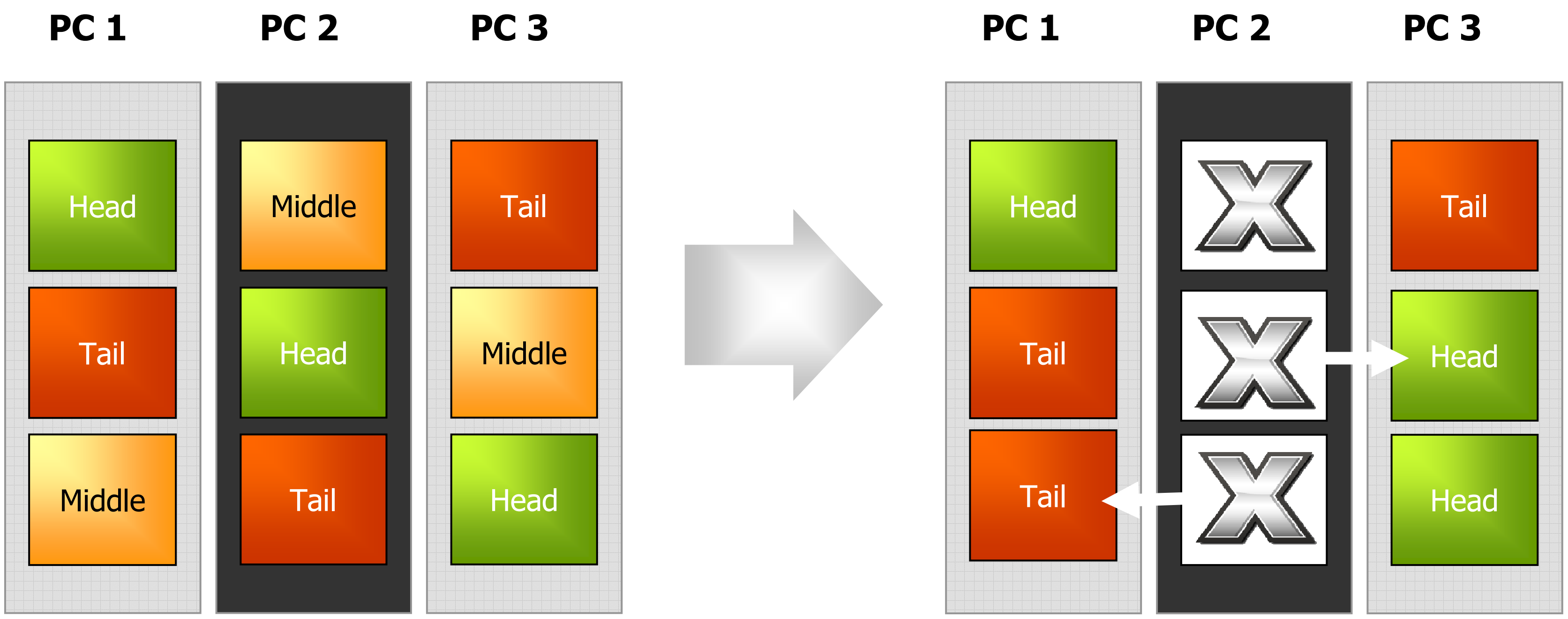


	PC 1	PC 2	PC 3
Chain A	Head	Middle	Tail
Chain B	Tail	Head	Middle
Chain C	Middle	Tail	Head

PC 4	PC 5	PC 6	
Head	Middle	Tail	Chain D
Tail	Head	Middle	Chain E
Middle	Tail	Head	Chain F

Chain Replication for High Availability and Fault Tolerance

Failover mechanism



PC 1	PC 2	PC 3
Head	Middle	Tail
Tail	Head	Middle
Middle	Tail	Head

Node down

PC 1	PC 2	PC 3
Head	X	Tail
Tail	X	Head
Tail	X	Head

Service continuation

Hibari Benchmarking

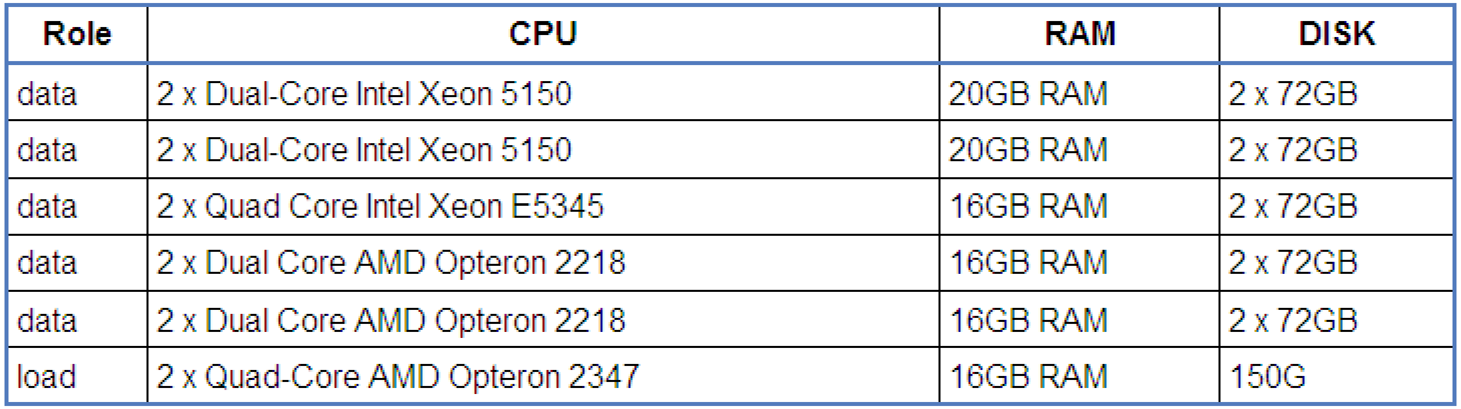
Hibari's benchmarking has been done with primarily 3 approaches

In house tools	• Micro benchmarks, standalone load client, and end-to-end as part of larger systems • C/C++ and Erlang based implementations • Good for Gemini's internal use but not ideal for the open source community
Yahoo's Cloud Storage Benchmark (YCSB)	• New, easy to use Java-based load tool • Java implementation for Cassandra db driver and *new* Hibari db driver • **But …** investigating latency issues with Hibari's YCSB db driver
Basho Bench (BB)	• Simple, easy to use Erlang-based load tool • Erlang implementation for Cassandra db driver and *new* Hibari db driver • **But …** traffic scenarios are not as full featured as YCSB and investigating stability issues with Cassandra's BB db driver

Test conditions

- Tool: Basho Bench - Hibari db driver and Cassandra db driver
- Keys: 1 … 1,000,000 Truncated Pareto (20% of keys, 80% of time)
- Values: 10K, 50k, 100K, and 150K
- Operations:
  - 20 minutes Random Put
  - 40 minutes Random Get/Put (50%/50%)
- Replication factor: 2
- Disk sync of commit log: enabled
- Storage: Keys RAM+DISK, Values DISK only

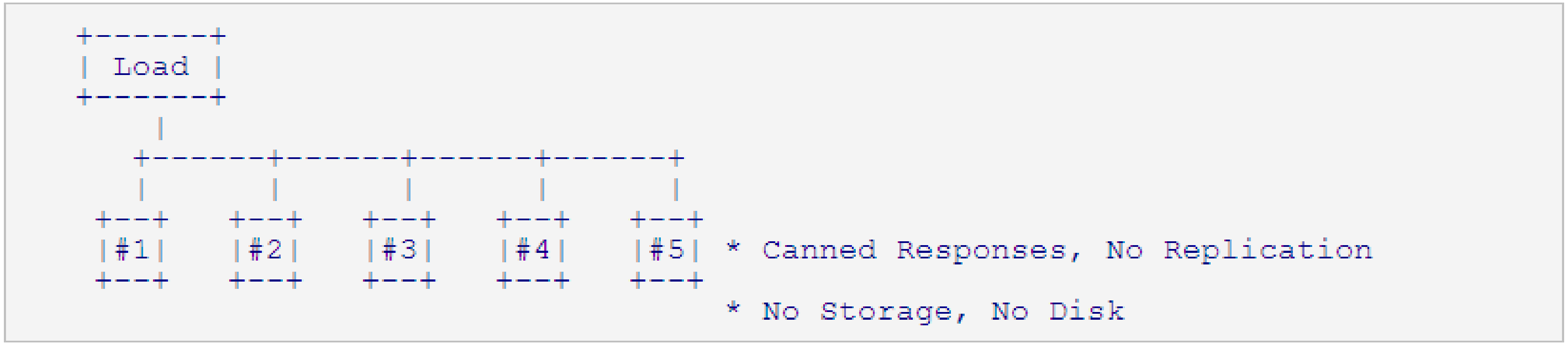
Hardware



Role	CPU	RAM	DISK
data	2 x Dual-Core Intel Xeon 5150	20GB RAM	2 x 72GB
data	2 x Dual-Core Intel Xeon 5150	20GB RAM	2 x 72GB
data	2 x Quad Core Intel Xeon E5345	16GB RAM	2 x 72GB
data	2 x Dual Core AMD Opteron 2218	16GB RAM	2 x 72GB
data	2 x Dual Core AMD Opteron 2218	16GB RAM	2 x 72GB
load	2 x Quad-Core AMD Opteron 2347	16GB RAM	150G

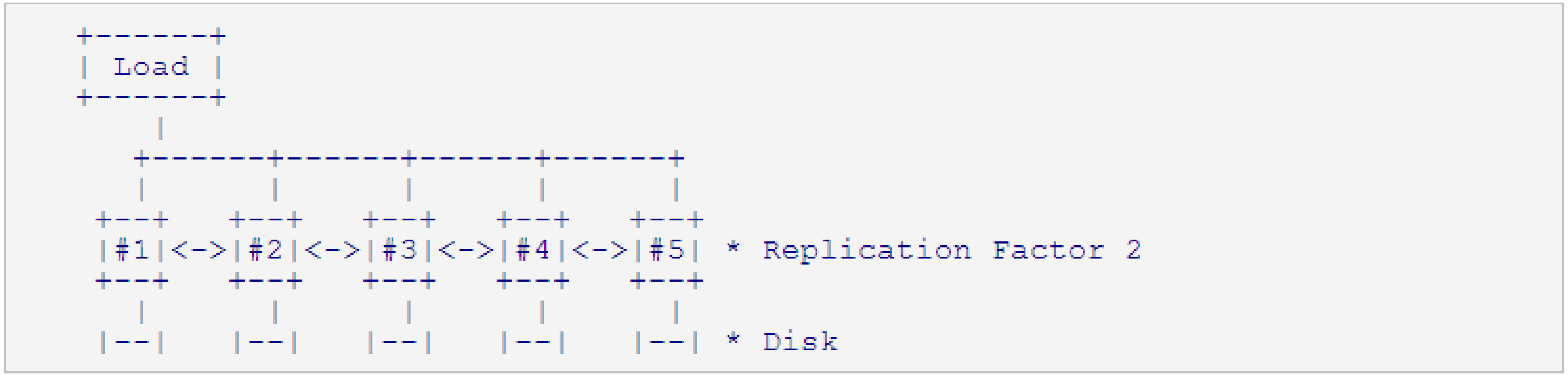
Test scenario

1. "Stub" Hibari Only Test Scenario



```
+-------+
| Load  |
+-------+
    |
   +-------+-------+-------+-------+
   |       |       |       |       |
 +---+   +---+   +---+   +---+   +---+
 |#1|    |#2|    |#3|    |#4|    |#5| * Canned Responses, No Replication
 +---+   +---+   +---+   +---+   +---+
                                      * No Storage, No Disk
```

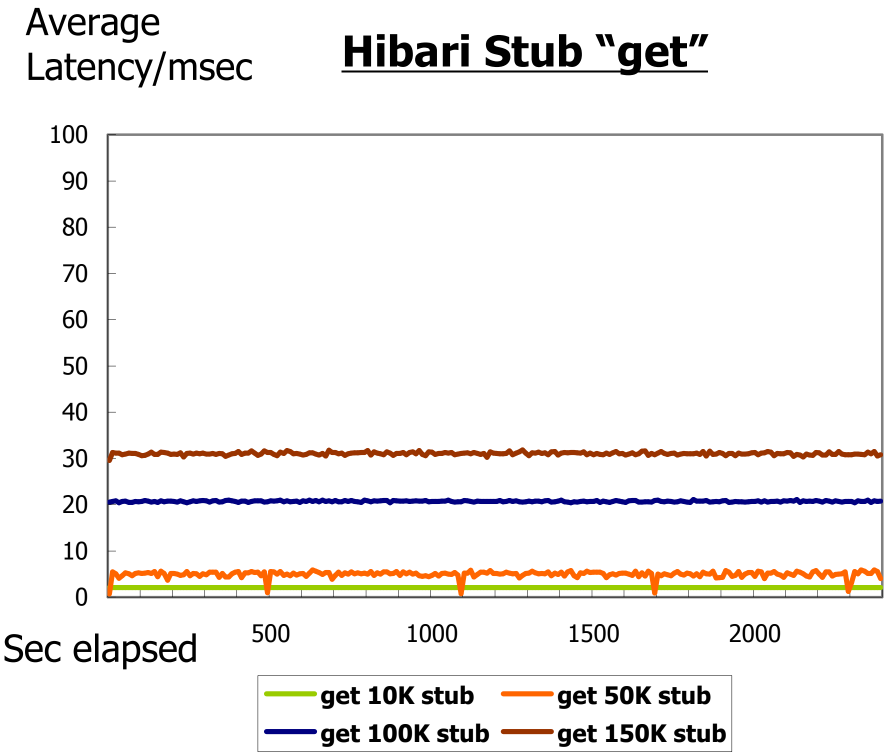
2. Hibari and Cassandra Test Scenario

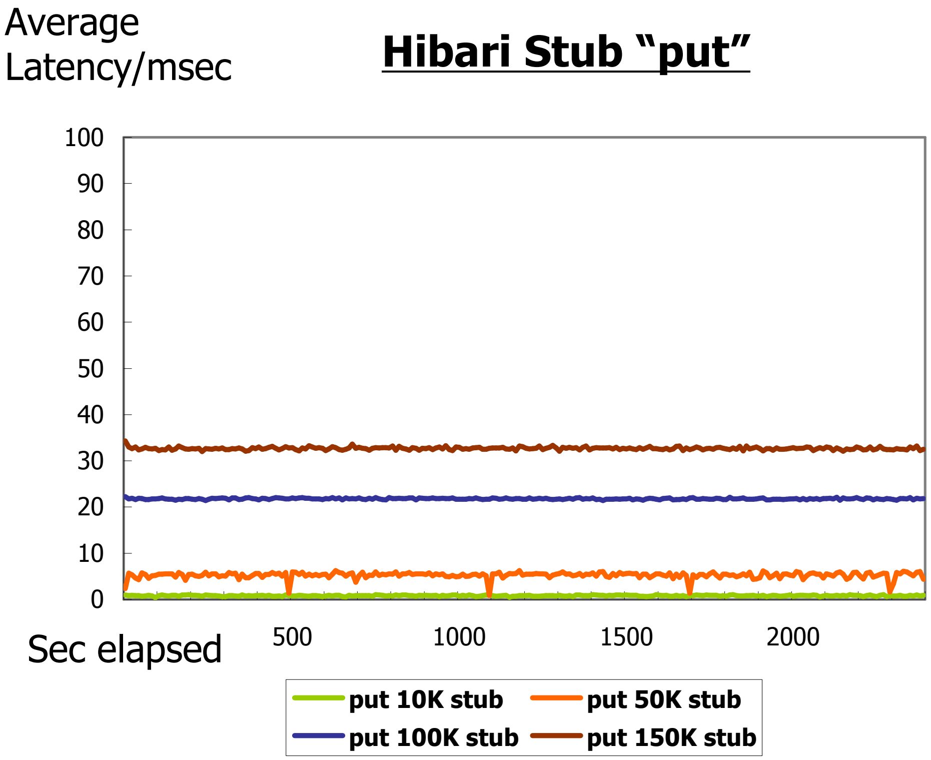


```
+-------+
| Load  |
+-------+
    |
   +-------+-------+-------+-------+
   |       |       |       |       |
 +---+   +---+   +---+   +---+   +---+
 |#1|<->|#2|<->|#3|<->|#4|<->|#5| * Replication Factor 2
 +---+   +---+   +---+   +---+   +---+
   |       |       |       |       |
 |--|    |--|    |--|    |--|    |--| * Disk
```

GEMINI

“Hibari Stub” results

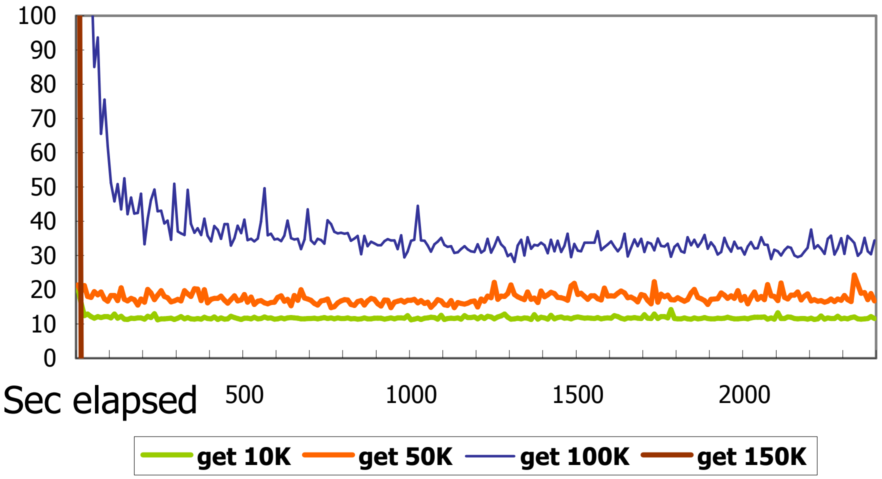




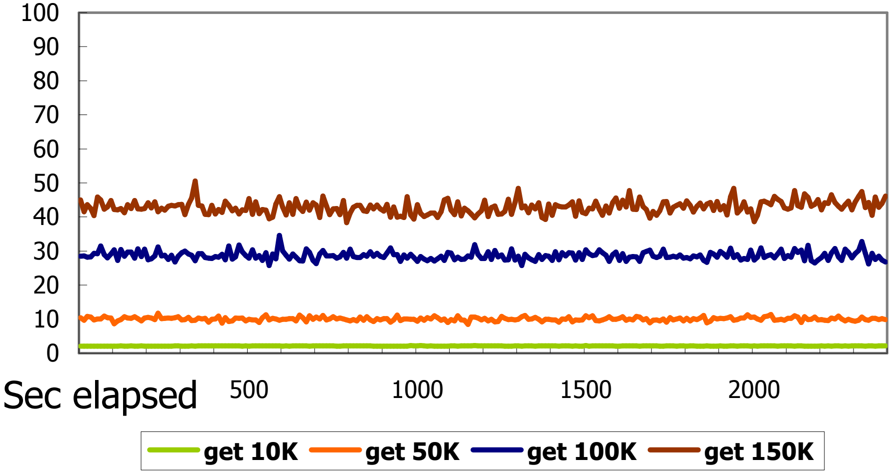
“Hibari and Cassandra” results by Basho Bench tool

Apache-Cassandra 0.6.5

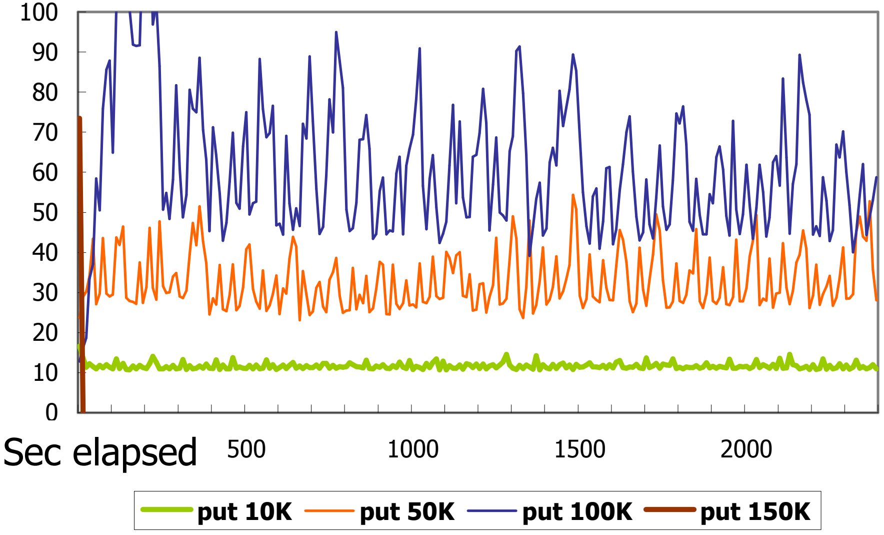
Cassandra “get”



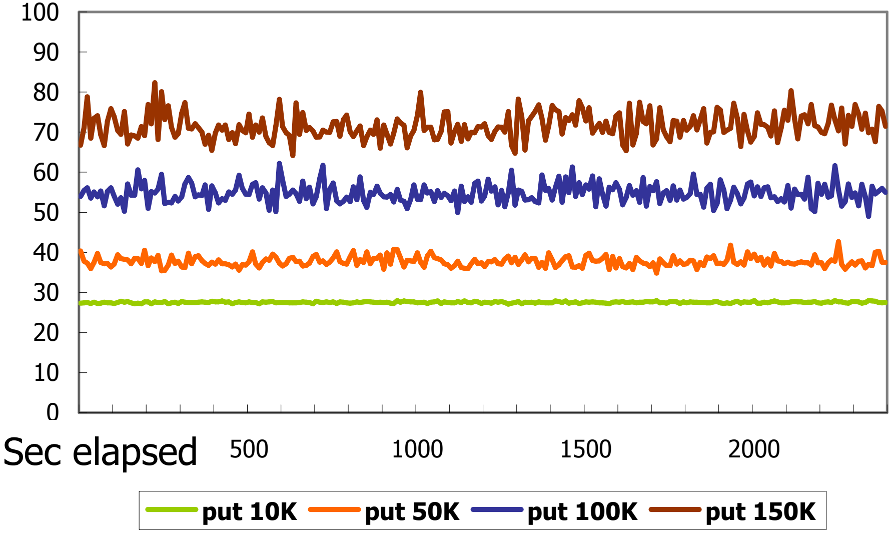
Hibari “get”



Cassandra “put”



Hibari “put”



Language (& Priority) Barriers

- Latency issues with Java load tool speaking with an Erlang NOSQL server

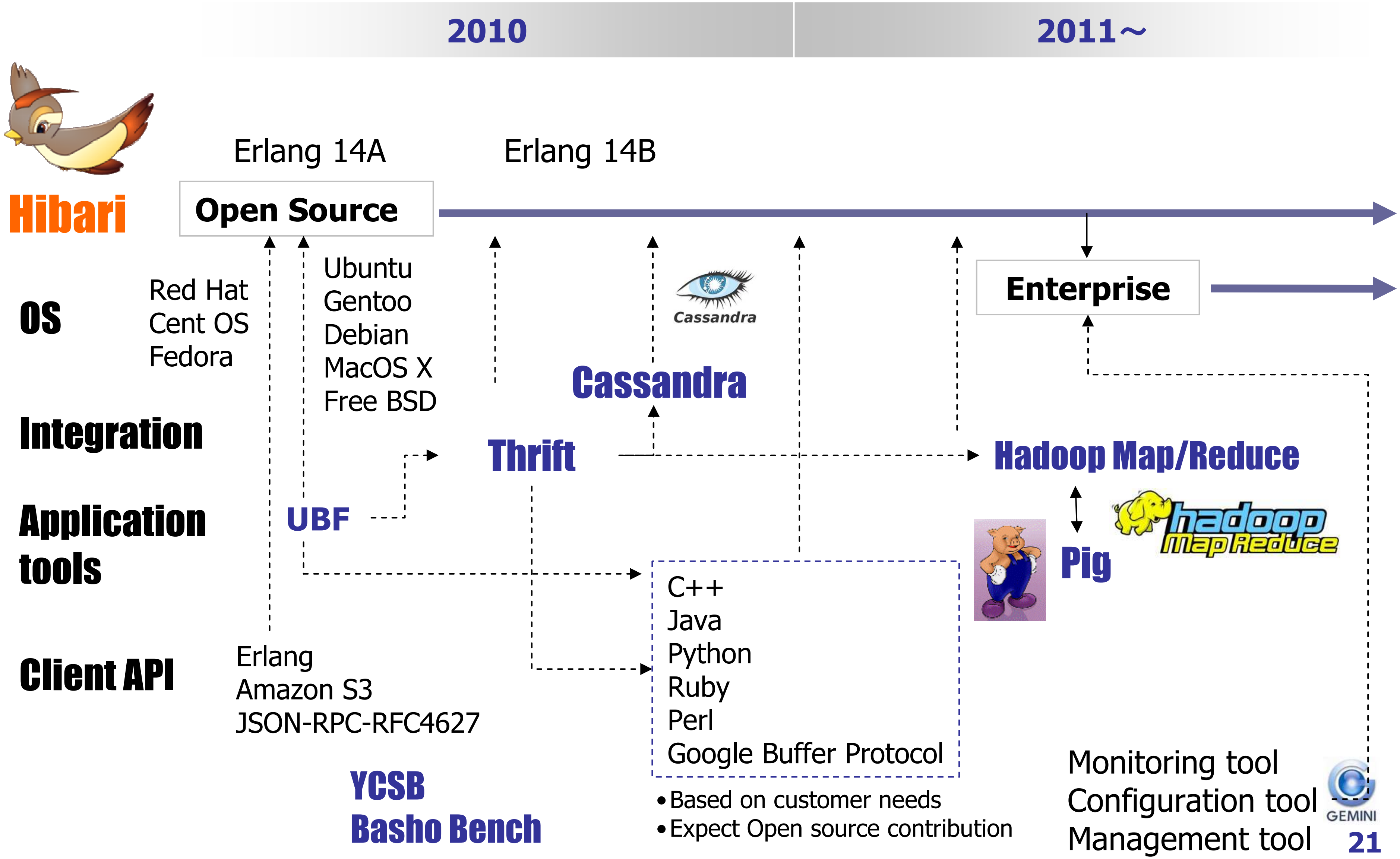
- Stability issues with Erlang load tool speaking with a Java NOSQL server

Further investigation is required to identify issues and areas of improvement. Our primary targets are Hibari, Cassandra, and the YCSB tool.

“Applications / Customer first” approach leads to mutual complements

FEATURE	(e.g.) Cassandra	Hibari
Data model	**Column-oriented**	**Key-value**
Data partitioning	Consistent hashing	Consistent hashing
Data consistency	Configurable	Yes
Data replication	**Preference lists**	**Chain replication**
Elasticity	Admin operations, Gossip, Data redistribution	Chain migration
Node health detection	Peer-to-peer monitor, gossip	Admin server monitors
O&M	nodetool, JMX	Admin UI with brick, chain health, statistics
Performance	**write-optimized**	**read-optimized**
Implementation	Java	Erlang
API	get/put/delete, scan, map/reduce, atomic row ops	get/put/delete, micro-transactions

“Not Only Hibari” Roadmap



2010

2011～

Hibari

Erlang 14A

Erlang 14B

Open Source

Enterprise

OS

Red Hat
Cent OS
Fedora

Ubuntu
Gentoo
Debian
MacOS X
Free BSD

Integration

Thrift

Cassandra

Hadoop Map/Reduce

Pig

Application tools

UBF

Client API

Erlang
Amazon S3
JSON-RPC-RFC4627

C++
Java
Python
Ruby
Perl
Google Buffer Protocol

YCSB
Basho Bench

- Based on customer needs
- Expect Open source contribution

Monitoring tool
Configuration tool
Management tool

“NOSQL Hands-on Training” will be started from December. Please follow @geminimobile

Thank you

Hibari Open Source project: http://sourceforge.net/projects/hibari/

Thrift: http://hibari.sourceforge.net/hibari-developer-guide.en.html#client-api-tbf

Hibari Twitter: @hibaridb Hashtag: #hibaridb

Gemini Twitter: @geminimobile

Big Data blog: http://gemini-bigdata.com/

Slideshare: http://www.slideshare.net/geminimobile