



Hibari: A Whitepaper

Table of Contents

1	Introduction	3
2	What is Hibari?	4
2.1	We are surrounded by “Big Data”	4
2.2	Hibari	4
2.3	Brief History of Hibari.....	5
2.4	NoSQL versus Relational Databases.....	5
3	Hibari Applications.....	7
3.1	Large-scale (GB) Mail System	7
3.2	Mail Archive	7
3.3	Log Processing for Business Intelligence.....	7
3.4	Large Scale (GB) Messaging Systems	8
3.5	Social Network data	8
3.6	Mobile DB.....	8
4	Hibari Technical Characteristics	9
4.1	Data Model	9
4.2	Hibari Key Concept: Consistency and Availability.....	9
4.3	Chain Replication	9
4.4	Client Application (Development Language and API)	10
4.5	High Performance	11
4.6	High Scalability.....	11
4.7	High Availability	11
4.8	High Reliability	12
4.9	Economical	12
5	Other Key Value Database	13
5.1	A partial list.....	13
5.1.1	Distributed key-value stores	13
5.1.2	Distributed Column stores	13
5.1.3	Document Oriented.....	14
5.2	Comparison.....	14
6	Appendix: The Press Release.....	15
	Open Source Release of 'Hibari,' A Database for Big Data	15

1 Introduction

Gemini Mobile Technologies, a worldwide leader in developing and deploying large scale messaging and storage system, has just released “Hibari” to the Open Source community. Hibari is a key value database, and is the result of more than five years work across several engineers from Gemini.

This whitepaper briefly describes Hibari and how Hibari fits in the increasingly crowded world of “Big Data”. Specifically, we will be discussing the specific “problems and pains” that Hibari addresses and solves.

Finally, we will also be discussing some interesting Hibari based applications that are currently deployed, as well as potential Hibari based applications that may be developed and deployed.

2 What is Hibari?

2.1 We are surrounded by “Big Data”

Believe it or not, we live in the world of “Big Data”. We are using Facebook daily, we use cloud-based mail systems, and the reality is that we are surrounded (and quite often overwhelmed) by huge data structures, quite often in the order of Terabytes and Petabytes.

The term “Big Data” describes datasets that grow so large that they become awkward to work with using standard, on-hand database management tools. Difficulties include capture, storage, search, sharing, analytics, and visualizing. This trend continues because of the benefits of working with larger and larger datasets allowing all of us to live and thrive in the modern “social world” (think Facebook, LinkedIn, Skype, Google Mail, etc), and allowing analysts to “spot business trends, prevent diseases, combat crime.” Though a moving target, current limits are on the order of terabytes, exa-bytes and zetta-bytes of data. Scientists regularly encounter this problem in internet search, big systems monitoring, meteorology, biological research, finance and business informatics. Data sets also grow in size because of the “multimedia” approach (text, audio, video, pictures, etc), since the root information is gathered by ubiquitous information-sensing mobile devices, software logs, cameras, microphones, RFID readers, wireless sensor networks and so on.



According to Wikipedia, “one current feature of Big data is the difficulty working with it using relational databases and desktop statistics/visualization packages, requiring instead “massively parallel software running on tens, hundreds, or even thousands of servers.”[10] The size of “Big data” varies depending on the capabilities of the organization managing the set. “For some organizations, facing hundreds of gigabytes of data for the first time may trigger a need to reconsider data management options. For others, it may take tens or hundreds of terabytes before data size becomes a significant consideration”.

2.2 Hibari

Hibari is an Open Source Key Value Data-Base, developed by Gemini Mobile Technologies, that solves or improves on the handling of problems created by “Big Data”.

Hibari main characteristics are:

1. Hibari is based on “key values”, therefore it is not a Relational Database (RDBMS), and belongs to the “noSQL” community (not only SQL);

2. Hibari runs on multiple nodes, and indeed it is a distributed database. These nodes can be located in different geographical locations. Read / write operations are distributed among multiples nodes;
3. Hibari is “Open Source” software. Open Source means “free”, everybody can download Hibari source code, and everybody can modify and improve open source code;
4. Hibari runs on commodity hardware, standard PCs blades, does not require expensive SAN architecture; typically 2 quad-core CPUs, 8-16 GB RAM, 4 250GB disks;
5. Hibari data is replicated across the different nodes, therefore providing much-needed fault tolerance without a single point of failure (SPOF).

2.3 Brief History of Hibari

Since 2001, Gemini Mobile Technologies has been a leader in providing complex solutions for huge carriers and other big Customers in the area of transaction processing and messaging processing. Gemini’s platform solution leverages Gemini’s HyperScale-technology. This powerful, modular architecture was designed from the ground up to provide mobile carriers, content providers and aggregators with solutions that set new standards for flexibility and scalability. HyperScale is the foundation of Gemini messaging platform: the HyperScale Messaging Center.

Major HyperScale deployments include the Messaging system at Softbank in Japan and the Messaging system at Nextel International. More recently, a large scale web mail system has been launched by a major Asian carrier, and this system is also based on the HyperScale technology.

In the course of developing these “big” architectures, there was an inherent need to develop an efficient, fast, scalable and flexible database. In 2005, the choice was taken at Gemini to develop an internal database, given the absence (at that time) of available systems that met the stringent Gemini requirements. So, the first incarnation of Hibari was born. Now, five years later, Gemini has decided to release Hibari to the open source community, so that everybody has a chance to further improve it, and every big project has a chance to use it directly.

So, while several incarnations of Hibari have been released commercially, Gemini believes that the software has reached a deep level of quality and stability, which again prompted the introduction to the Open Source arena. The open community is indeed very well exposed to plenty of real-life requirements (in terms of scalability, reliability, and especially in terms of pluralities of applications taking advantage of this software), and therefore the open community is the most logical venue for this technology to perfect and to grow.

2.4 NoSQL versus Relational Databases

Typical modern relational databases have shown poor performance on data-intensive applications including indexing a large number of documents, serving pages on high-traffic websites and delivering streaming media. They can be efficient only when they are tuned either for small but frequent read/write transactions or for large batch transactions with rare write accesses, while there are

demands for data stores capable of heavy workloads with frequent updates. Real-world examples include Digg's 3 TB for green badges, Facebook 's 50 TB for inbox search and eBay's 2 PB overall data.

NoSQL architectures often provide weak consistency guarantees such as eventual consistency and/or transactions restricted to single data items. Some systems, however, provide full ACID (*atomicity, consistency, isolation, durability*) guarantees, in some instances by adding a supplementary middleware layer (e.g. CloudTPS).

Several NoSQL systems employ a distributed architecture, with the data held in a redundant manner on several servers, often using a distributed hash table. In this way the system can readily scale up by adding more servers, and failure of a server can be tolerated.

3 Hibari Applications

This section is only a brief appetizer for all the many applications that can be built using Hibari: indeed, new Applications are indeed discussed daily. This section attempts to provide a list of applications that have already been commercially launched, and at the same time proved an (incomplete) list of some proposed (but yet to be implemented or developed) applications for Hibari.

3.1 Large-scale (GB) Mail System

This is the application that is currently live at a major carrier in Asia. This installation today has in excess of 3M users, and Hibari is used in storing all the email generated by those users. Clearly, the size of every single email can be very small, or extremely huge.

The system itself runs on 56 independent nodes. As of June 2010, there were approximately 1.3 billion emails stored in the production system.

3.2 Mail Archive

Mail Archiving has become more and more critical after the latest regulations (not only in the United States) about storing all corporate email for an indefinite number of years, with fast access provided to a restricted set of officials and people. The challenge is the classical one:

- Huge amount of data, and huge amount of records
- Size of record is not consistent, it may be very small or very huge
- Immediate access is a key requirement
- The “key value” to be used in this application may very well be:
 - A date range
 - The sender
 - The recipient
 - The subject
 - A specific keyword

3.3 Log Processing for Business Intelligence

In any voice or data network (typically IP these days), fixed line or wireless, the data gathered by Log and Statistics is growing quickly. A rise in data and messaging traffic, corresponds to a parallel rise for Logs and Statistics. Storage of Log and Statistics data is expensive (more hardware, larger data centers, more power consumption, etc), and some information must be processed in real time (for example, the transitioning of CDNs based on HTTP sessions).

The advantages that a Hibari-based application will bring are:

- A much better optimized network, due to the ability to process logs and statistics data;

- No more loss of important data (such as subscriber use information) because transaction data cannot be stored and cannot be processed prior to deletion of data;
- The ability to retain data long enough for useful data mining
- The ability to extract useful statistics from log data even if it is stored

3.4 Large Scale (GB) Messaging Systems

This application is similar to the first one presented here (Large Scale Mail System), with the difference that specific records are not limited to emails. They may indeed include:

- SMS or Text Messages (from SMSC)
- MMS – Multimedia Messages (from MMSC)
- Voice Records (from Voice Messaging Systems)
- IM Messages (from Instant Messaging Systems)

3.5 Social Network data

A modern Social Network needs to contain all sorts of information (from reputation-based recommendations, all the way to friend and block lists; and of course gigabytes of pictures, video clips, audio clips, etc); and this information requires exactly the database characteristics offered by Hibari.

3.6 Mobile DB

A mobile Database will encompass all the information related to every specific user of the mobile system. This may include not only the specific user profile information, but also critical information on the user behavior and usage of the system.

4 Hibari Technical Characteristics

As discussed earlier, Hibari inherits important characteristics of both the distributed non-RDBMS world, and of Gemini HyperScale Platform.

Hibari has the following characteristics to flexibly support growing volumes of data in the cloud computing space.

- Economical system build-up using commodity hardware
- Flexible, high scalability
- High performance and availability by consistent hashing across multiple machines
- High fault-tolerance by chain replication across multiple machines

For instance, when applied to large-scale email systems, Hibari can provide economic efficiency and thus avoid a typical situation: employees/customers occasionally must delete messages and files to free up space in limited capacities of mail systems and file servers.

Also, when "Hibari" is applied to a large-scale archive system, a storage system can be economically developed and operated to perpetually store and manage vast amounts of information automatically generated by a wide range of systems, e.g., logs and telemetric measurements (for instance, location of cars and other moving devices, and usage of utility services).

4.1 Data Model

Hibari provides a simple data model that has five (5) attributes: keys to uniquely identify data items, values that correspond to keys, timestamps to detect race conditions for accessing, expiry to manage expiration of data, and flags to store meta data.

4.2 Hibari Key Concept: Consistency and Availability

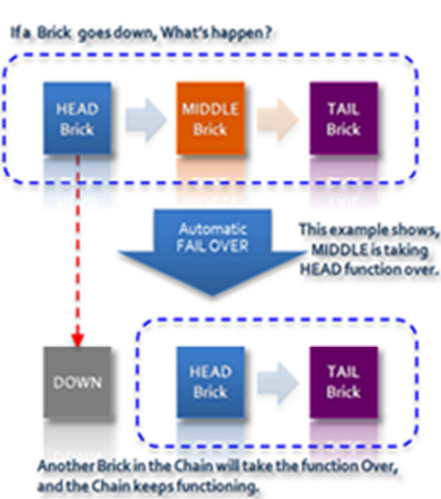
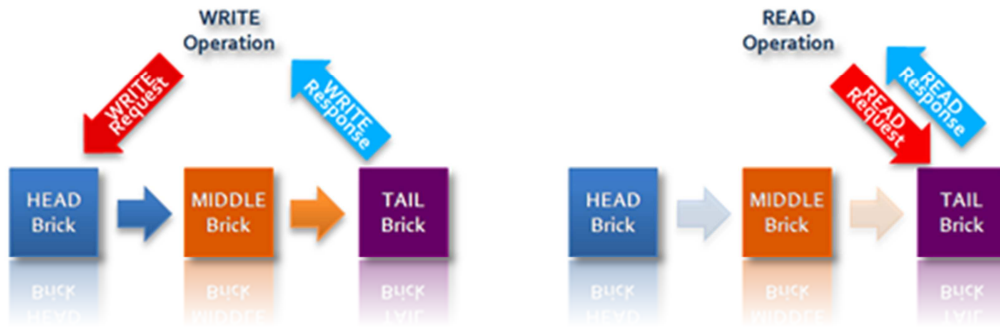
In its architecture, Hibari incorporates a concept of storage bricks to build a large storage system by using a number of commodity computers connected to a network and a concept of chain replication to provide high throughput and availability without sacrificing data consistency in a large storage system.



4.3 Chain Replication

In Hibari, a database cluster consists of multiple chains, and each chain is a set of multiple groups of brick storages (of computers connected to a network). It means that a database cluster of Hibari consists of a number of brick storages that are grouped into chains.

For chain replication, each chain is basically structured with three types of brick storages; one head brick, one tail brick, and zero or more middle bricks. Data is replicated in all brick storages within the chain.



When there is a request for data write to a chain in a database cluster, the request is sent to the head brick of a chain. The head brick receiving the data write request updates data and sends it via a middle brick to a tail brick to execute replication within the chain. A reply to the write request is returned from the tail brick. Chains of length two have no middle brick. In chain of length one, the role of head brick and tail brick are performed by the same brick (Chains of length one have no data redundancy). When there is a request for data read in a chain, the request is sent to a tail brick of the chain. The tail brick returns a reply.

If a failure happens in a brick storage (node) within a chain, the chain is automatically shortened to sustain normal chain

operations. For instance, when a chain configuration has three brick storages, even if two of them fail simultaneously, access to data in the chain is maintained. When a brick store is restarted, its data will be repaired automatically. When a brick's repair is finished, the brick will be automatically re-added to the chain.

4.4 Client Application (Development Language and API)

Hibari provides various development options to flexibly support different needs of development of client applications.

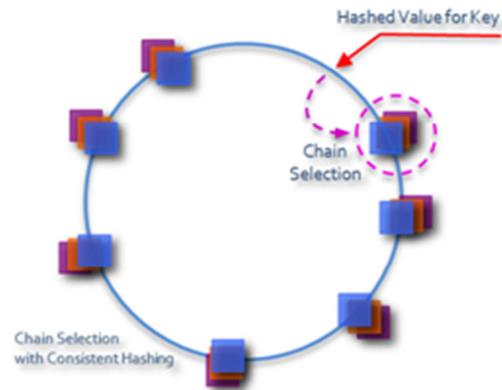
Hibari provides highly-versatile APIs including Amazon S3, JSON-RPC-RFC4627, Universal Binary Protocol (UBF), soon to be followed by Thrift, Avro, and Google's Protocol Buffers. Hibari supports Java, C/C++, Python, Ruby, Erlang.

4.5 High Performance

Erlang, a programming language which excels at parallel processing, is the driver for high performance of Hibari. Erlang shares many of the design principles as HyperScale® Technology and simplifies integration with other Gemini HyperScale® software components. The following two approaches also help to build a high-performance distributed system.

The first approach is data distribution based on consistent hashing algorithm. The algorithm is used to allocate data stored in Hibari in multiple chains within a database cluster. Data allocated in multiple chains enables parallel execution of simultaneous access to data distributed to individual chains.

The second approach is storage options. Hibari can provide options of disk-based storage and RAM (on-memory)-based storage. For instance, when high speed is particularly required in a system, RAM-based storage can be selected. In both cases, data is written and flushed to disk to protect against data center power failures. A "batch commit" technique is used to minimize disk I/O.



4.6 High Scalability

Hibari adopts an architecture that allows dynamic addition of brick storages (nodes) that construct a database cluster. The architecture enables the expansion of database clusters, such as upgrading storage capacities and access performance, without service outage.

Also, Hibari supports the following features for easy system expansion:

- Automatic replication
- Automatic balancing of data allocation upon a change of cluster configuration, e.g, addition or reduction of nodes

Hibari thus has an architecture that enables dynamic resource addition to clusters and provides automatic data management functions.

4.7 High Availability

Hibari delivers high availability of data access based on consistent hashing algorithm and chain replication.

Consistent hashing algorithm of Hibari is the basis for partitioning of key space of database to allow distribution of data in multiple chains. Efficient distribution of data to database clusters drives parallel processing of access to individual chains, improving the availability of data access.

Also, each chain of the Hibari database cluster can be redundantly configured by using multiple (two or more) bricks in each chain. As long as at least one brick storage (node) within a chain is running, the chain is fully functional. This includes retaining capacity, throughput, as well as ability to read and write. For instance, when a chain configuration has three bricks even if two of them fail simultaneously, functionality of the chain is maintained.

In Hibari, all other components of a system can be also redundantly configured to totally eliminate a single point of failure (SPOF) within the system. Each brick store machine is an independent unit of failure. Failures of disk, RAM, CPU, power supply, network interface, or any other component are all considered the same by the chain replication implementation. This architecture is frequently called a "shared nothing" storage architecture.

4.8 High Reliability

Generally speaking, maintaining consistency of data items that are replicated for redundancy is an important challenge for distributed databases.

Hibari guarantees strong consistency of multiply-replicated data items distributed in a database. In reply to a request for data read from a client, the system always provides the latest data for which a write operation is completed.

Also, when a RAM-based storage option is selected, Hibari records all data updates on disks to ensure data durability.

4.9 Economical

Hibari can realize economical system build-up, using commodity hardware, regardless of the required size of a system. Hibari's system expansion method is basically "scale-out." The size of clusters (extension or reduction) can be flexibly changed, without any service outage, to cope with a change of required processing performance or storage size of the system.

In addition, hardware of differing capacities can be mixed as nodes in a cluster. It allows the user to select the latest, cost-effective hardware whenever it is necessary to extend a cluster's size.

Also, reallocation of data upon a change of cluster size or recovery after a failure is automatically executed by automatic balancing of data allocation, simplifying operations and reducing overall operation cost.

5 Other Key Value Database

5.1 A partial list

We distinguish three fairly distinct types of non-relational database:

- a. Distributed key-value stores
- b. Distributed column stores
- c. Document oriented

5.1.1 Distributed key-value stores

Examples of distributed key-value stores are:

- Dynamite
<http://github.com/cliffmoon/dynomite>
- Riak
<http://www.basho.com>
- Voldemort
<http://project-voldemort.com/>
- Redis
<http://programmingzen.com/2009/03/11/introducing-redis-a-key-value-database/>
- Tokyo Cabinet
<http://fallabs.com/tokyocabinet/>
- Scalaris
<http://code.google.com/p/scalaris/>

5.1.2 Distributed Column stores

- Cassandra
<http://wiki.apache.org/cassandra/DataModel>
- HBase
<http://hbase.apache.org/docs/current/api/overview-summary.html>
- Hypertable
<http://www.hypertable.org/>

- Amazon Dynamo
http://www.readwriteweb.com/archives/amazon_dynamo.php
- Google BigData
<http://labs.google.com/papers/bigtable.html>

5.1.3 Document Oriented

- Couch DB
<http://couchdb.apache.org/>
- Mongo DB
<http://www.mongodb.org/>

5.2 Comparison

As typically is the case, real life comparisons under the same circumstances are very difficult to perform. Quite often, one ends up comparing apples to oranges, and oranges to mangos.

Regardless, it is worth noting some initial indications from preliminary testing (that for sure will be followed by more robust testing, and this is a key advantage of being in the Open Source space).

When running parallel testing with HBase (version 0.20.3) and Cassandra (version 0.5), Hibari performance for random and sequential keys is practically the same as Cassandra and HBase. The interesting note is that Hibari read performance appears to be superior with large value (above 200KB) entries.

6 Appendix: The Press Release

Open Source Release of 'Hibari,' A Database for Big Data



SAN MATEO, Calif., July 27 /PRNewswire/ -- Gemini Mobile Technologies (Gemini) announced at Wireless Japan 2010 in Tokyo that it will release Hibari (meaning Cloud Bird in Japanese) as open source. Hibari is a database optimized for the highly reliable, highly available storage of massive data, so-called Big Data. Hibari can be used in Cloud Computing Applications such as web mail, Social Networking Services (SNS), and other services requiring storage of tera-bytes and peta-bytes of new daily data.

Hibari, developed by Gemini, is based on distributed non-relational database technologies of key value store and chain replication. These technologies bring benefits of low cost and high reliability by enabling data storage on tens or hundreds of PC servers, instead of costly special-purpose storage appliances such as SANs. Development started in 2005, and has been deployed and commercially proven in a number of large telecom operators, storing everything from SNS digital goods to Cloud Mail for millions of users.

Big Data applications are growing rapidly, fueled by tremendous growth in digital content, social media, automatically-generated data such as logs, histories, and telemetric statistics (electricity utilization, vehicle location information, etc.). By releasing Hibari to open sourcing, Gemini expects its commercially-proven, non-relational database technology to be used in a variety of fields, including enterprise private cloud computing, digital entertainment, e-commerce, financials, and telemetries.

Hibari is developed in Erlang, and is released under the Apache license. Hibari provides highly-versatile APIs including Amazon S3, JSON-RPC-RFC4627, Universal Binary Protocol, and soon-to-be-released Thrift, Avro and **Google's** Protocol Buffers; Hibari supports [Java](#), C/C++, Python, Ruby, and Erlang. Gemini plans to provide Hadoop Map-Reduce integration as well as a commercial license which includes updates and support.

-- Hibari download site - <http://sourceforge.net/projects/hibari/> -- About Gemini Mobile Technologies - <http://www.geminimobile.com/> -- About Hibari - <http://www.geminimobile.com/technologies/technologiesDB.html> -- Contact us - <http://www.geminimobile.com/contact/contactus.html>

Authors:

Gemini Mobile Technologies

Feedback:

gpropersi@geminimobile.com

United States

2600 Campus Drive, Suite 280
San Mateo, CA 94403
Phone: +1.650.227.2380
Fax: +1.650.227.0299

Japan

1-10-5 Dogenzaka, Shibuya-ku
Shibuya Place 7F
Tokyo 150-0043
Phone + 81.3.5456.9486
Fax + 81.3.5456.9481

China

Star City, C-1108
10 Jiu Xian Qiao Rd
Chao Yang District
Beijing 100016
Phone: +86.10.5827.9090
Fax: +86.10.5827.9010

