



第六讲：NoSQL综述和Apache HBase基础

Will Du



IT21
LEARNING

IT21 Learning Inc.

本课目标 (Objective)

- ◆ 了解NoSQL的基本概念
- ◆ 了解HBase的基本概念
- ◆ 了解HBase的设计
- ◆ Shell查询HBase



最积极的回答问题 & 最精准的翻译关键字



NoSQL基本概念

History of the Database

- ◆ 1960' s – Hierarchical and Network type
- ◆ 1970' s – Beginnings of theory behind relational model.
- ◆ 1980' s – Rise of the relational model. SQL. E/R Model
- ◆ 1990' s – Access/Excel and MySQL. Online Data MS appears
- ◆ 2000' s – Two forces; large enterprise and open source. Google and Amazon. CAP Theorem (more on that to come...)
- ◆ 2010' s – Immergence of NoSQL as an industry player and viable alternative for IoT

What is NoSQL?

- ◆ A NoSQL (originally referring to "non SQL", "non relational" or "not only SQL")
- ◆ NoSQL is a generic term used to refer to any data store that does not follow the traditional RDBMS model—specifically, the data is non-relational and it does not use SQL as the main query language. It is used to refer to the databases that attempt to solve the problems of scalability and availability against that of atomicity or consistency.

Not **O**nly **SQL**

Why are the NoSQL Needed?

Internet Scale

- ◆ Millions of users
- ◆ Low cost of storage
- ◆ Increased processing power
- ◆ Ability to capture (and need) of millions of events. Caching solves it to an extent but brings other complexities
- ◆ Real-time response
- ◆ Need to scale out and not up. (add infinite number of low cost machines vs. replace with a more powerful machine).

Cost

- ◆ Let' s not forget for enterprise DB' s Internet scale can become expensive
- ◆ Open source DB' s may solve license cost, but don' t ignore operational costs

Relational vs. NoSQL

◆ Relational

Divide into tables, relate into foreign keys, DB constraints, normalized data, the Interface is SQL

◆ NoSQL

Store in schemaless format, redundancy encouraged, application access determines the storage format (your queries). Interface varies and is optimized for the implementation, no forced DB constraints.

What Are Tradeoffs from NoSQL?

- ◆ **Eventual consistency**
- ◆ **Application has increased responsibility such as maintain consistency & handle transactions**
- ◆ **Store redundant data**

NoSQL is NOT Big Data

- ◆ NoSQL != Big Data
- ◆ NoSQL products were created to help solve the big data problem.
- ◆ Big data is a much larger problem than just storage.
 - Hadoop
 - Kafka
 - Spark, etc.



HBase基础

HBase 基础

- ◆ What is Apache HBase
- ◆ HBase History, Use Cases, and User Group
- ◆ HBase Architecture

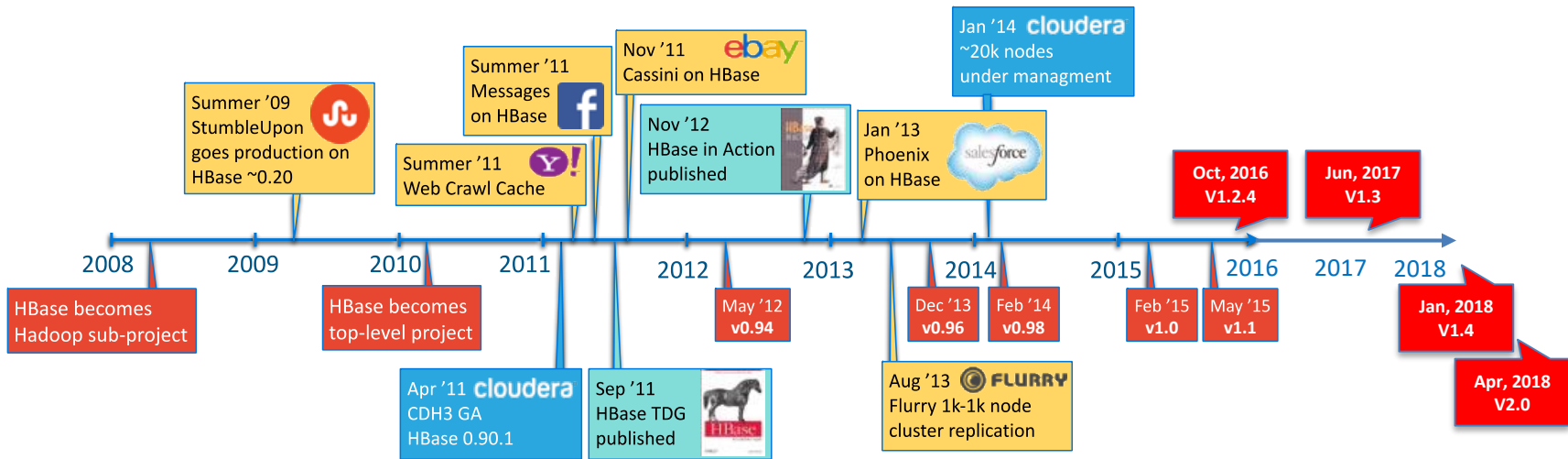


What is Apache HBase

- ◆ HBase is a leading No-SQL database which stores data on HDFS.
- ◆ HBase is column-oriented database.
- ◆ HBase is a distributed hash map.
- ◆ HBase is based on the Google Big Table paper.
- ◆ HBase uses HDFS as storage and leverage its reliability.
- ◆ Data can be accessed quickly, ~2-20 millisecond response time.
- ◆ Great support random read & write 20k to 100k+ ops/s per node.
- ◆ Scales to 20,000+ nodes.



Apache HBase History



Year	Event
Nov 2006	Google released the paper on BigTable.
Feb 2007	Initial HBase prototype was created as a Hadoop contribution.
Oct 2007	The first usable HBase along with Hadoop 0.15.0 was released.

Apache HBase User Groups



Apache HBase Use Case

◆ Capturing Incremental Data – Time Series Data

- ❖ High volume, high velocity writes

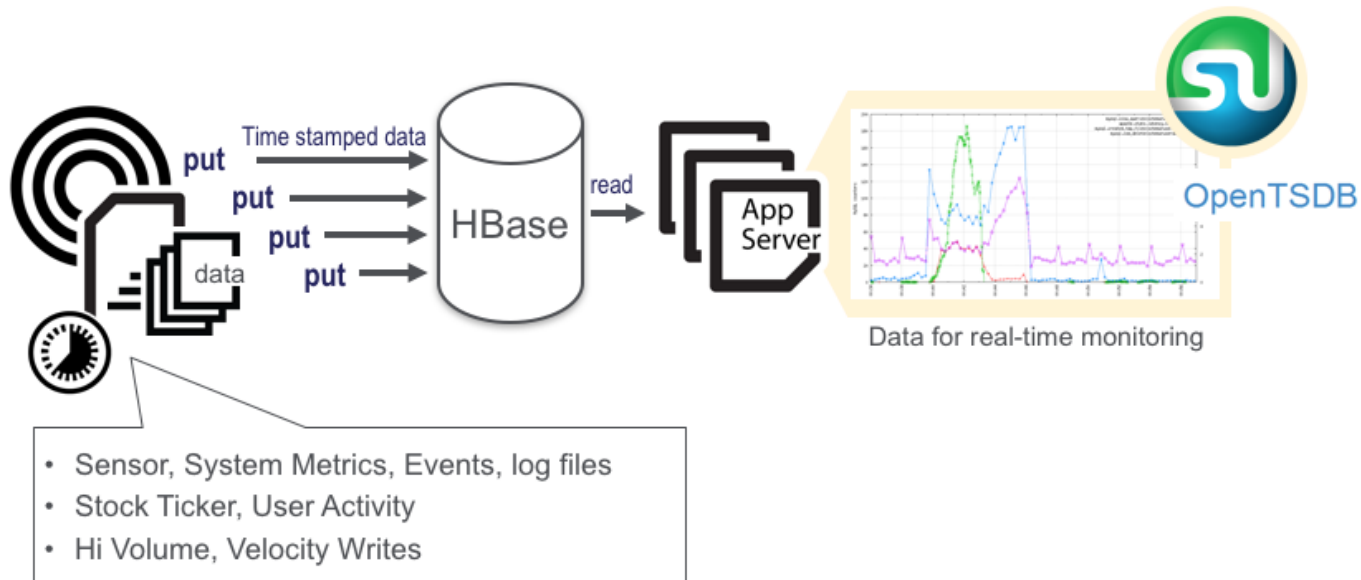
◆ Information Exchange – Messaging

- ❖ High volume, high velocity write/read

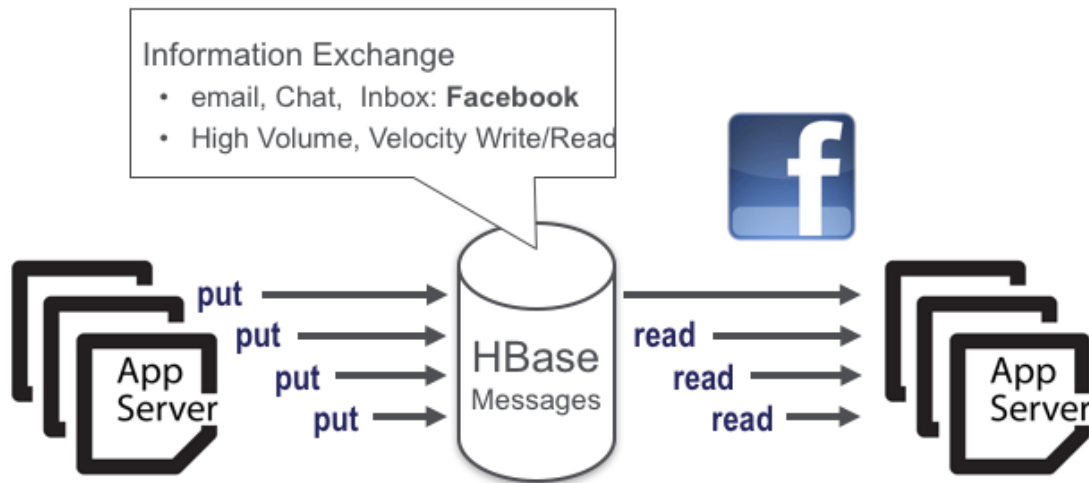
◆ Content Serving – Web Application Backend

- ❖ High volume, high velocity reads

Time Series Data



Information Exchange



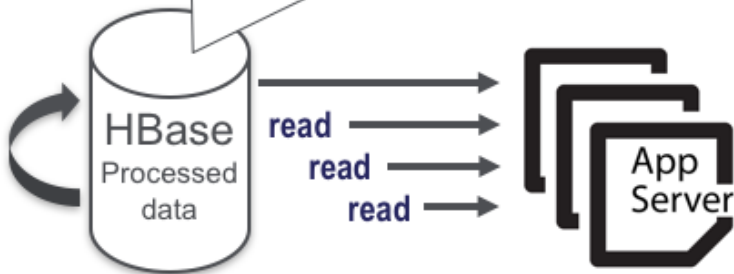
Content Serving

Content Serving, Web Application Backend

- Online Catalog: Gap, World Library Catalog.
- Search Index: ebay
- Online Pre-Computed View: Groupon, Pinterest
- High Volume, Velocity Reads



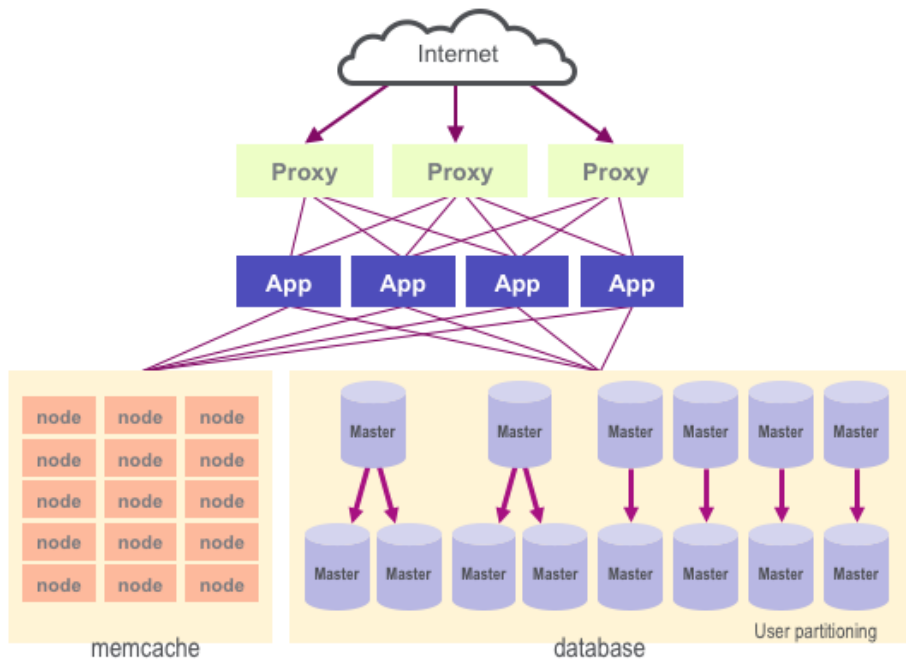
Bulk Import
Pre-Computed
Materialized View



Use Case Example - HBase in Facebook

- ◆ 9000 memcached instances
- ◆ 4000 shards mysql
- ◆ Moves to Hbase since 2011

facebook®



Use Case Example - HBase in Alibaba Search

◆ HBase is a core storage in Alibaba search system since 2010

◆ History Version

- 2010 ~ 2014: 0.20 -> 0.94
- 2014 ~ 2015: 0.94 -> 0.98
- 2016: 0.98 -> 1.1.2

◆ Current Scale

- ❖ 3 cluster each with 1000+ nodes
- ❖ Share with Flink on Yarn
- ❖ Serving over 10M+/s operations every day

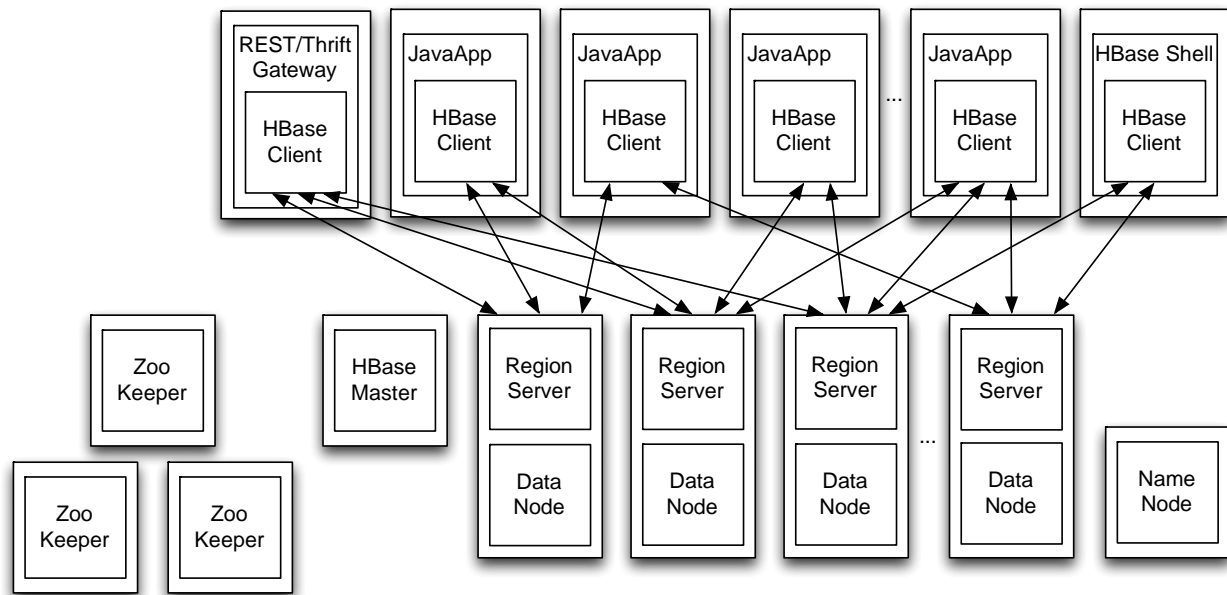


Apache HBase Ecosystem

- ◆ Lily – CRM on HBase
- ◆ OpenTSDB – HBase for time-series data
- ◆ Kylin – OLAP on HBase
- ◆ Phoenix – SQL over HBase
- ◆ Splice Machine – OLTP with HBase
- ◆ Apache Tephra – HBase transaction
- ◆ TiDB – Distribute SQL DB
- ◆ Apache Omid - Optimistically transaction Management
- ◆ Yarn application timeline server v.2 is moving to HBase
- ◆ Hive metadata store is moving to HBase
- ◆ Ambari Metrics Server will use HBase as data store

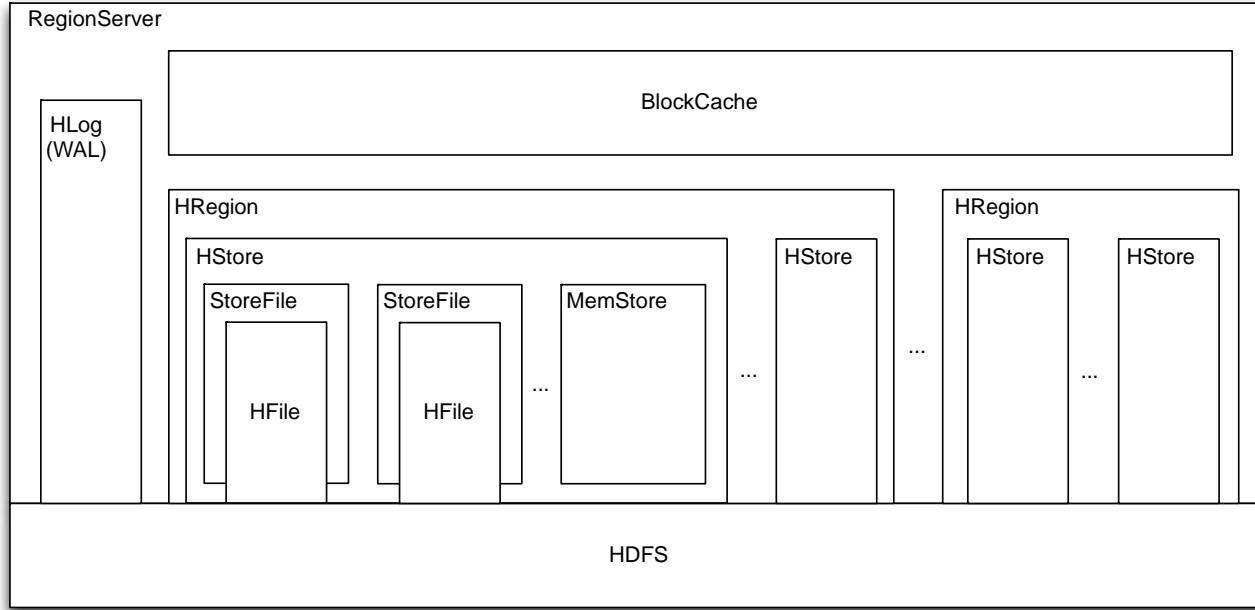


HBase Overall Physical Architecture



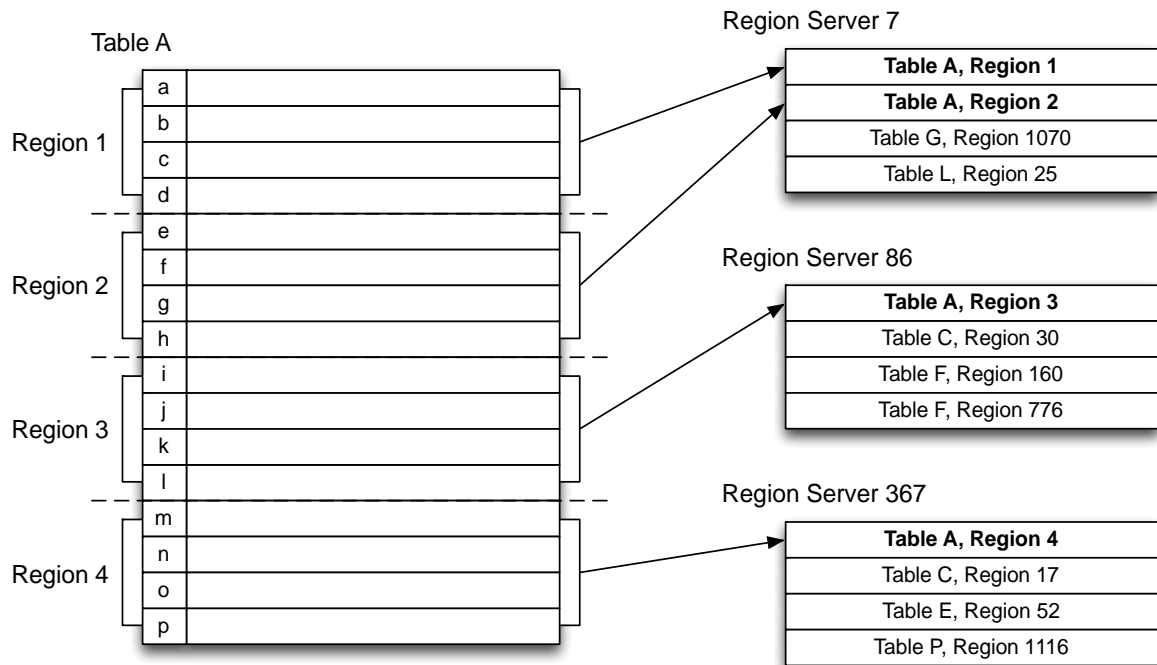
- An HBase RegionServer is collocated with an HDFS DataNode.
- HBase clients communicate directly with Region Servers for sending and receiving data.
- HMaster manages Region assignment and handles DDL operations.
- Online configuration state is maintained in ZooKeeper.
- HMaster and ZooKeeper are NOT involved in data path.

Physical Architecture – Region Server



- A RegionServer contains a single WAL, single BlockCache (read cache), and multiple Regions. - A Region contains multiple Stores, one for each Column Family.
- A Store consists of multiple StoreFiles and a MemStore (write/edit cache).
- A StoreFile corresponds to a single HFile and Column Family.
- HFiles and WAL are persisted on HDFS as sequence file.

Logical Architecture – Region和Table



- A single table is partitioned into Regions of roughly equal size.
- Regions are assigned to the Region servers across the cluster.
- A region is assigned to only one server
- Region servers host roughly the same number of region.
- Column families split within regions to group setup columns together
- Client directly talks to the region server

Logical Architecture – Rows

rowkey	column family	column qualifier	timestamp	value
a	cf1	"bar"	1368394583	7
			1368394261	"hello"
		"foo"	1368394583	22
			1368394925	13.6
	cf2	"2011-07-04"	1368396302	"fourth of July"
		"number"	1368387684	"almost the loneliest number"
b	cf2	"thumb"	1368387247	[3.6 kb png data]

- Row key are unique and sorted.
- Schema can define when inserts the records
- Each Row can define its own columns, even if other rows do not use them.
- Related Columns grouped into Column Families (prefer<3) which are saved into different files.
- Multiple row versions maintained with unique timestamps.
- Value types can change between versions.
- HBase only knows bytes and clients must impart meaning.

Physical coordinates for a cell:

Region directory => CF Directory => Row Key => Column Family => Column Qualifier => Version

Data Management

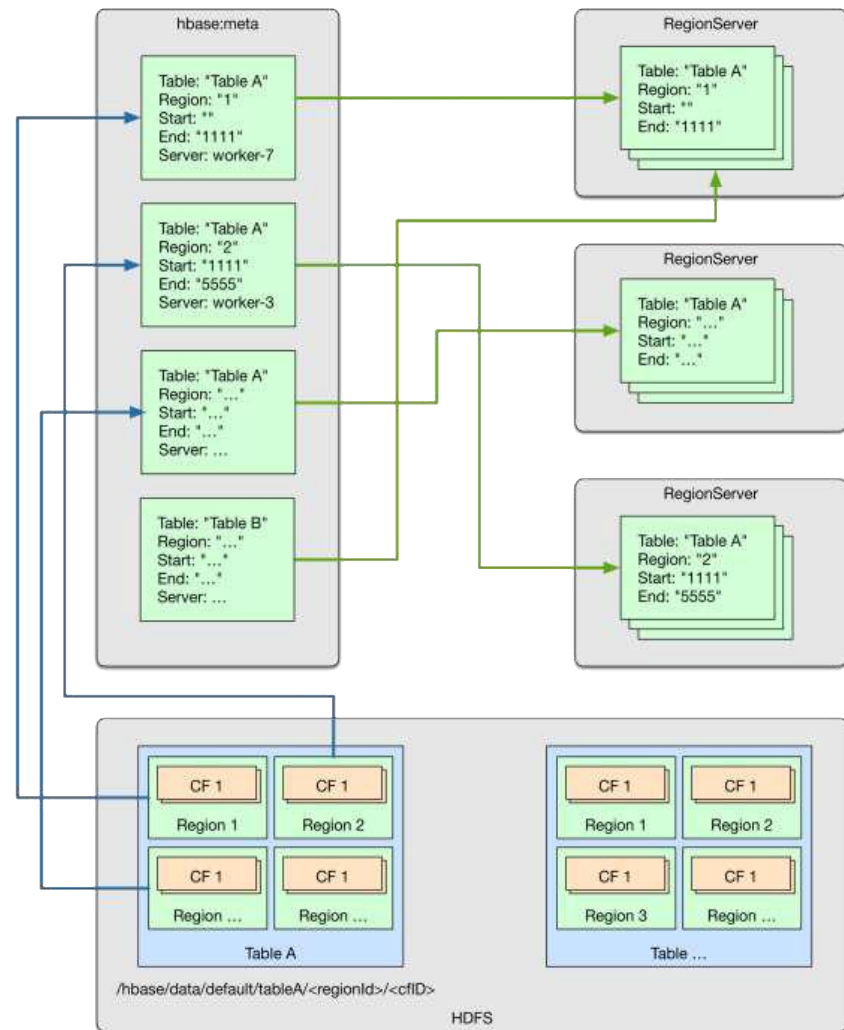
What is available is tracked in three locations

- System catalog table hbase:meta
- Files in HDFS directories
- Open region instances on servers

System aligns these locations

- Sometimes (very rarely) a repair may be needed using HBase Fsck

Redundant information is useful to repair corrupt tables



总结: HBase Architecture Advantages

HBase provides the following benefits

◆ Strong consistency model

- when a write returns, all readers will see same value

◆ Scales automatically

- Splits when regions become too large
- Uses HDFS to spread data and manage space

◆ Built-in recovery

- Using a Write Ahead Log, similar to journaling on file system

◆ Integrated with Hadoop

- MapReduce on HBase is straightforward

How to Access HBase – HBase shell

- ◆ HBase shell is an interactive mode of using hbase
- ◆ It support full set of hbase commands

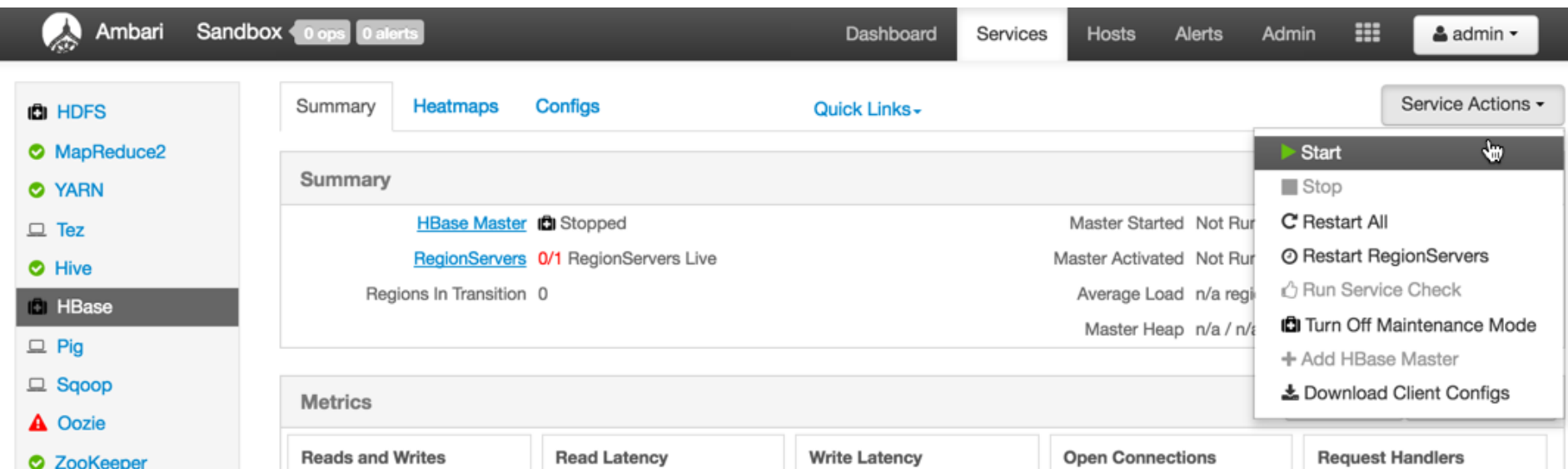
CMD Category	CMDs
<u>General</u>	version, status, whoami, help
<u>DDL</u>	alter, create, describe, disable, drop, enable, exists, is_disabled, is_enabled, list
<u>DML</u>	count, delete, deleteall, get, get_counter, incr, put, scan, truncate
Tools	assign, balance_switch, balancer, close_region, compact, flush, major_compact, move, split, unassign, zk_dump
Replication	add_peer, disable_peer, enable_peer, remove_peer, start_replication, stop_replication



HBase Shell 实战

HBase服务开启

Sandbox



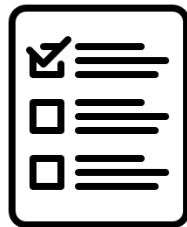
The screenshot shows the Ambari Sandbox interface. The top navigation bar includes 'Ambari', 'Sandbox' (with 0 ops and 0 alerts), 'Dashboard', 'Services', 'Hosts', 'Alerts', 'Admin', and a user profile 'admin'. The left sidebar lists services: HDFS, MapReduce2, YARN, Tez, Hive, HBase (selected), Pig, Sqoop, Oozie, and ZooKeeper. The main content area has tabs for 'Summary', 'Heatmaps', 'Configs', and 'Quick Links'. The 'Summary' tab shows the HBase service status: 'HBase Master' is 'Stopped', 'RegionServers' are '0/1 RegionServers Live', and 'Regions In Transition' is '0'. A 'Service Actions' dropdown menu is open, showing options: 'Start' (highlighted), 'Stop', 'Restart All', 'Restart RegionServers', 'Run Service Check', 'Turn Off Maintenance Mode', 'Add HBase Master', and 'Download Client Configs'. Below the summary, there is a 'Metrics' section with a table showing various performance metrics.

Metrics	
Reads and Writes	Read Latency
Write Latency	Open Connections
Request Handlers	

Lab machine: ops start

总结 (Summary)

- ◆ 了解NoSQL的基本概念
- ◆ 了解HBase的基本概念
- ◆ 掌握常用HBase命令



课后作业 (Assignment)

◆ 自己实现HBase课上演示



