Big Data with Hadoop

Pierre Sauvage
Big Data Consultant
pierre@adaltas.com

HBase

Low latency, columnar Database

Prerequisite

- CAP Theorem

  It is impossible for a distributed system to be

  - Consistent
  - Available
  - Partition tolerant

HBase

- **What** is HBase ?
- **How** to use HBase ? (and how it works)
- **When** to use Hbase ?

HBase

What is HBase ?

What is HBase ?

- FOSS NoSQL DB modeled after Google's Big Table
- On top of HDFS !
- Consistent, multidimentional, Sorted map (key-value database)
- (Key, column family, column, timetamp) -> value

What is **NOT** HBase ?

- Not a relationnal (SQL) database
- No joins out-of-the box
- No fancy nor sophisticated language
- No transactions

HBase cool features

- Linear scalability
- Automatic/manual sharding
- Failover support / fault tolerant
- **Strictly** consistent: R & W
- “random” r/w

Integration with Hadoop

- Integrates with Hadoop MR (and DAGs)
- Very easy Java API
- Thrift and REST API
- Bulk loading with Map Reduce
- Cross cluster replication tools
- Block cache for RT applications (fast Data) (ADVANCED usage)

Example

- Store user information and friends
- Key: username
- Column families: info and friends
- inside “info”:
  columns: properties (eg email)
  values: property values
- inside friends:
  columns: friends username
  values: age, phone number, whatever

Example: representation

key	cf	Col.	ts.	value
pierre	info	age	84078 63746 43414	24 23 22
		email	-	pierre@adaltas.com
	friends	ninon	-	1990-11-26
		jeanne		1995-03-01

http://wiki.apache.org/pig/UDFManual

HBase

How to use HBase ?

Data Model

- Untyped keys & values (byte arrays)
- Column are grouped by column family (more-or-less like Hive partitions)
- CF are defined STATICALLY at creation

Data Insertion

- **Not atomic !!!**
- HBase keeps 3 versions of your data (using timestamp as a subkey)
  get latest value by default
- No need to fill all columns nor Cfs

Hadoop integration

- HBase table as input of MR
- MR output into HBase table
- “Bulk load” HDFS files into Hbase (it uses a MR job)

Basic Operations

- Create a table:
  create 'pierrotws', 'info'
  alter 'pierrotws', 'friends'
- View table structure:
  describe 'pierrotws'
- Remove a table:
  disable + drop 'pierrotws'

Example

- Inserting

  put 'pierrotws', 'pierre', 'friends:ninon', '1990-11-26'
  put 'pierrotws', 'pierre', 'friends:jeanne', '1995-03-01'
  put 'pierrotws', 'pierre', 'info:email', 'pierrotws@adaltas.com'
  put 'pierrotws', 'pierre', 'info:age', '24'

- Getting

  scan 'pierrotws'

HBase

How does it work ?

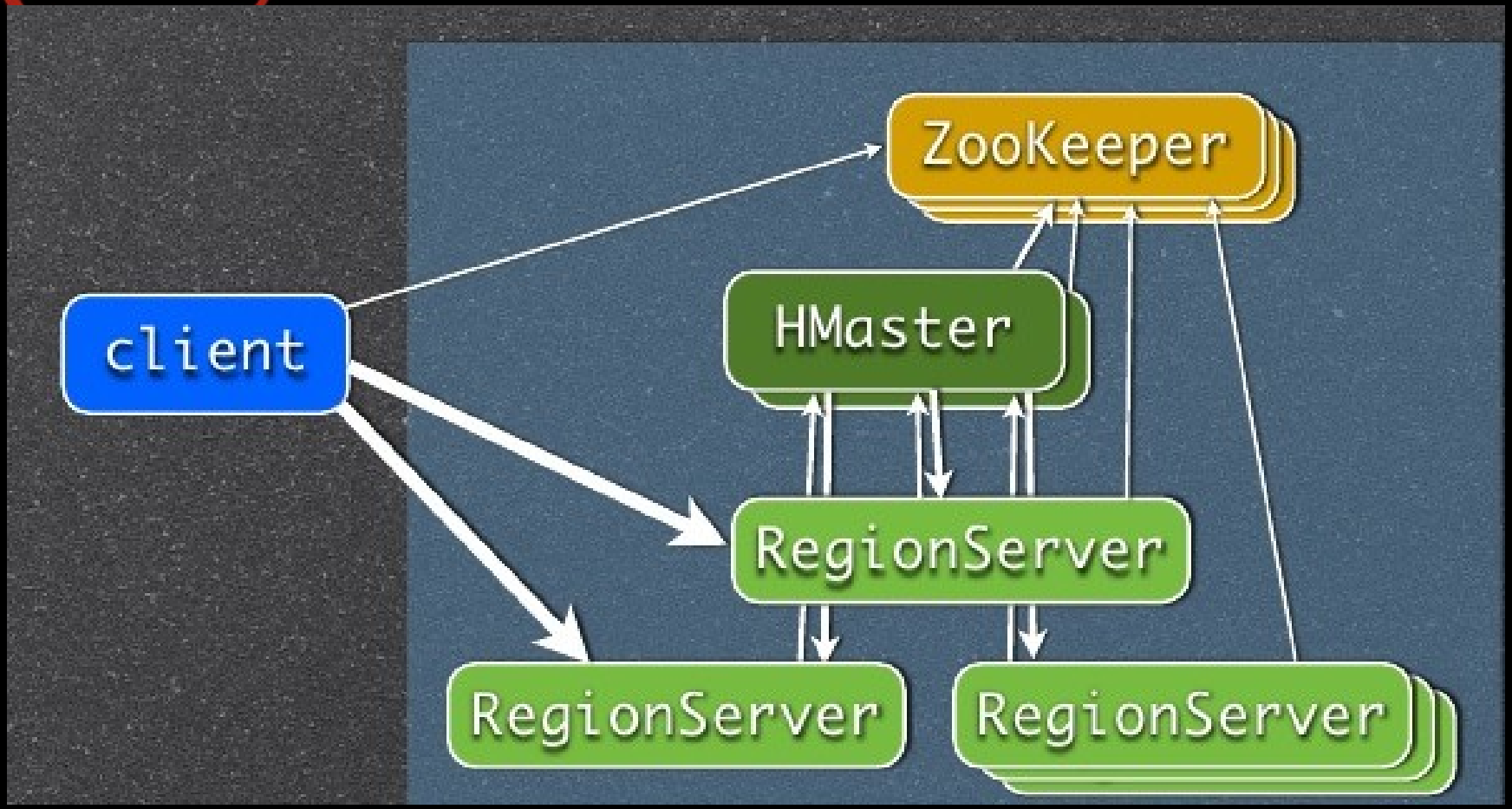
Sharding

- Tables are splitted into Regions
- Regions defined by start/end of keys
- Regions are assigned to RegionServers (workers of HBase, eq. to NodeManagers for YARN, DataNodes for HDFS)

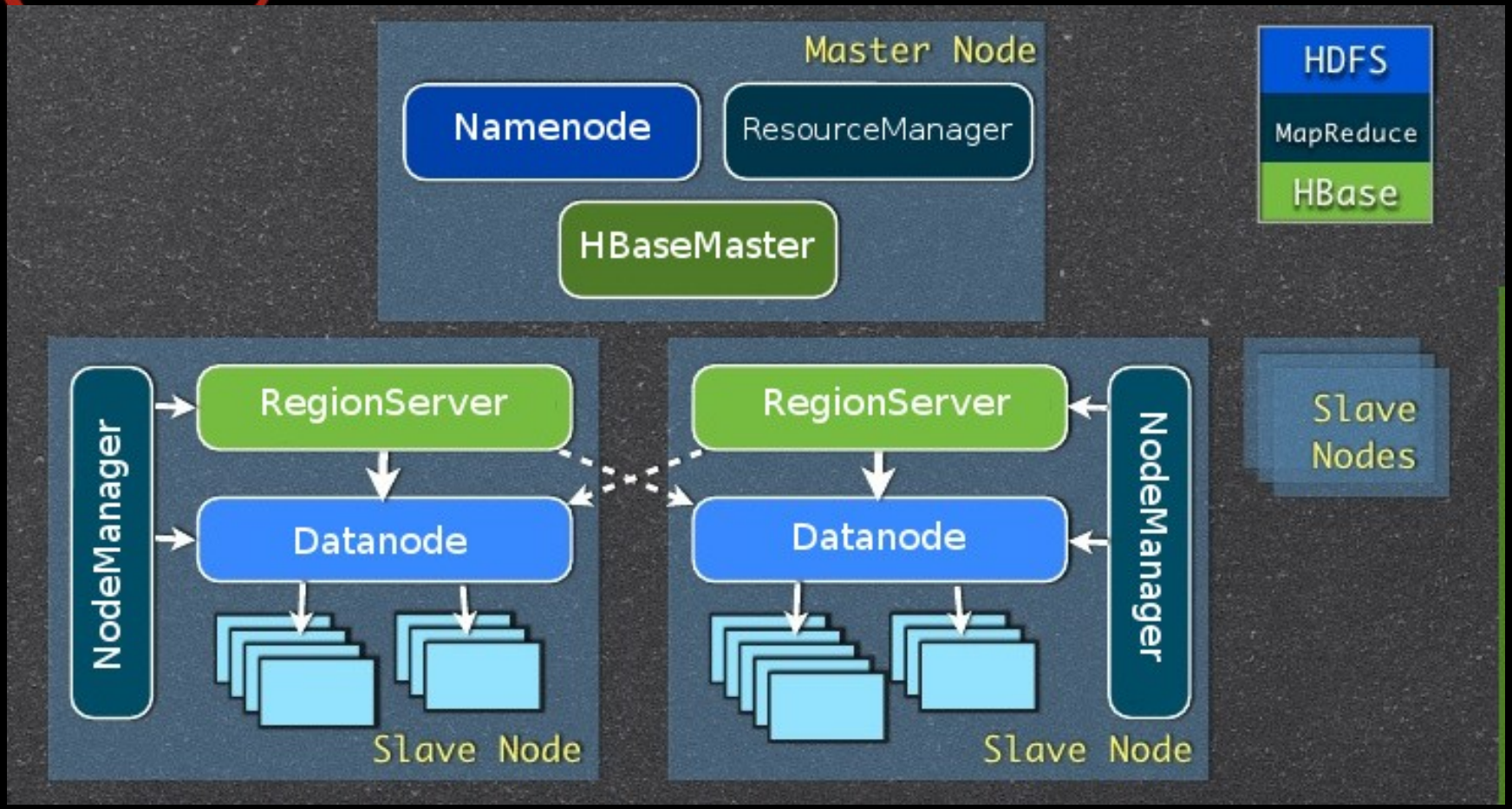
HDFS Storage

- Hfile
  - Optimised SequenceFile containing Index and data
  - Use tombstone marker for deletion (HDFS cannot update a file)
  - Post 'insert' compaction algorithm to optimize storage
- Edit logs
- In-memory cache

HBase components



HBase & Hadoop



Failover

- Data failures handled by HDFS
- RS failures handled by HbaseMaster
- HMaster failover supported

HBase

When to use HBase ?

Use Case

- Fast random access
- Large amount of data/writing
- Append-style modifications (no read-modify-write complex queries)
- Consistency over Availability

Work-In-Progress

- Stability
- Performance improvements
- RS on YARN
- Rise of the Phoenix