Large scale data near-line loading method and architecture

FiberHome Telecommunication
2017-7-19

/usr/bin/whoami

Shuaifeng Zhou(周帅锋):

- Big data research and development director (Fiberhome 2013-)
- Software engineer (Huawei 2007-2013)
- Use and contribute to HBase since 2009
- sfzhou1791@fiberhome.com

Contents

1 Motivation

2 Solution

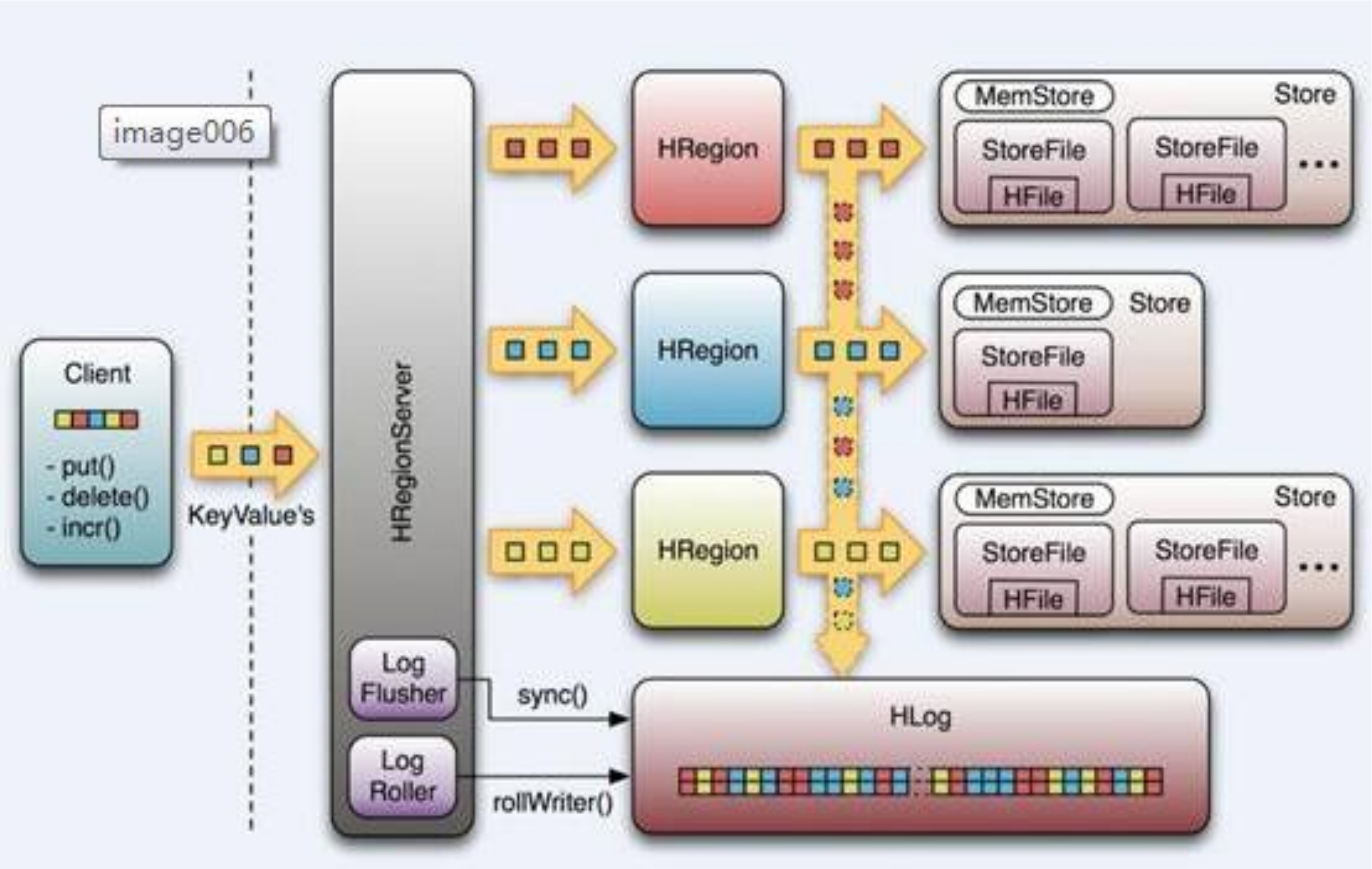
3 Optimization

4 Tests

5 Summarize

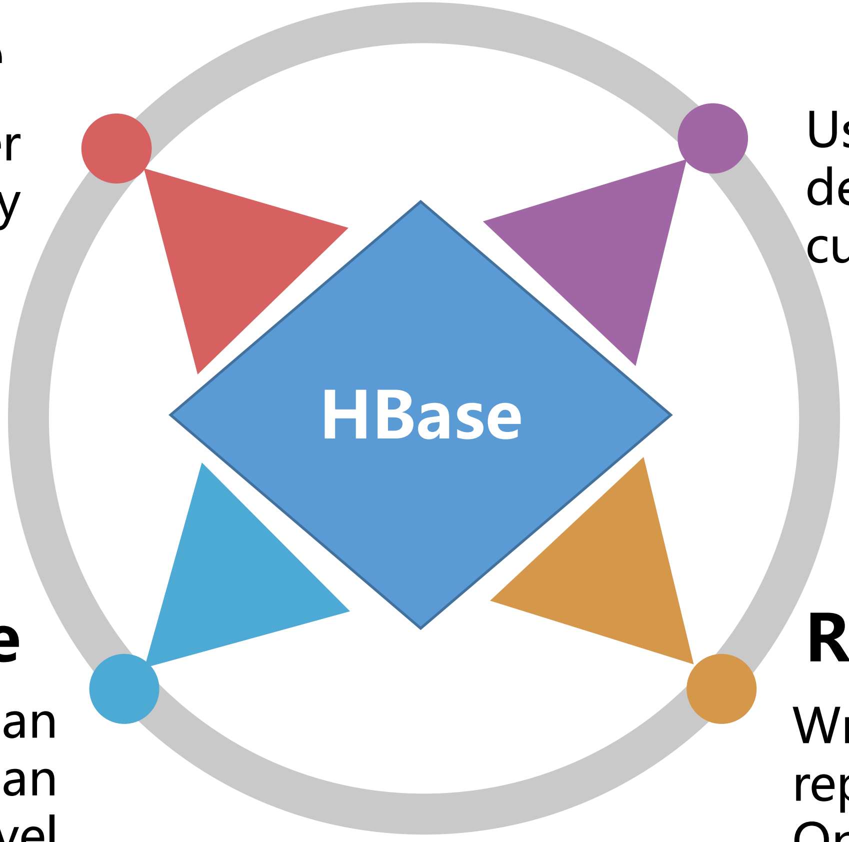
HBase Realtime Data Loading

FiberHome



- WAL/Flush/Compact Triple IO pressure
- Read/Write operations share resource:
  - Cpu
  - Network
  - Disk IO
  - Handler
- Read performance decrease too much when write load is heavy

Why near-line data loading?



Scale

Billions write ops per region server one day

Delay

Usually, several minutes delay is acceptable for customers

Resource

Resource occupied can be limited under an acceptable level

Reliable

Write op can be repeated
Optimistic failure handling

Large scale data loading reliably with acceptable time delay and resource occupation

Contents

1 Motivation

2 Solution

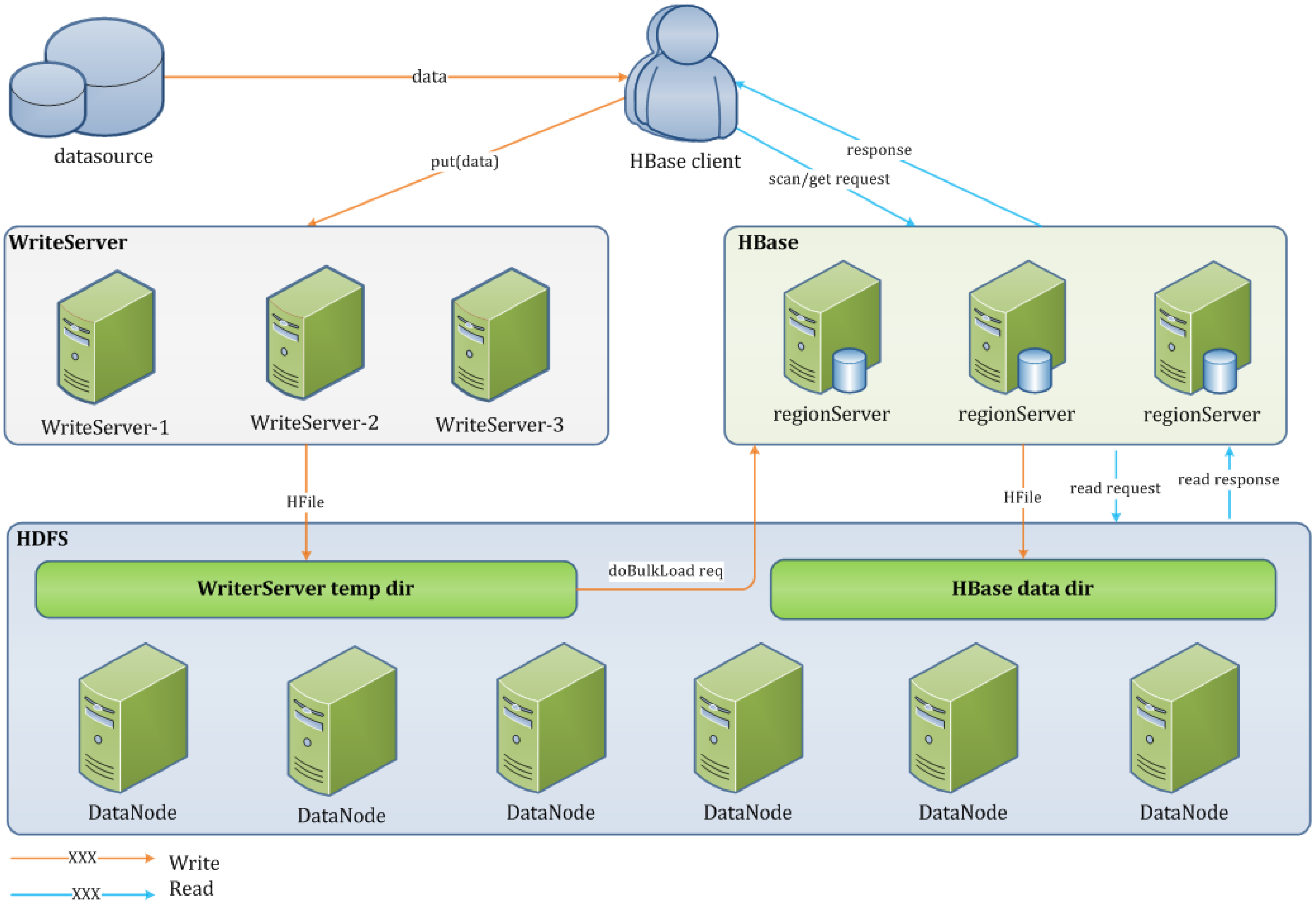
3 Optimization

4 Tests

5 Summarize

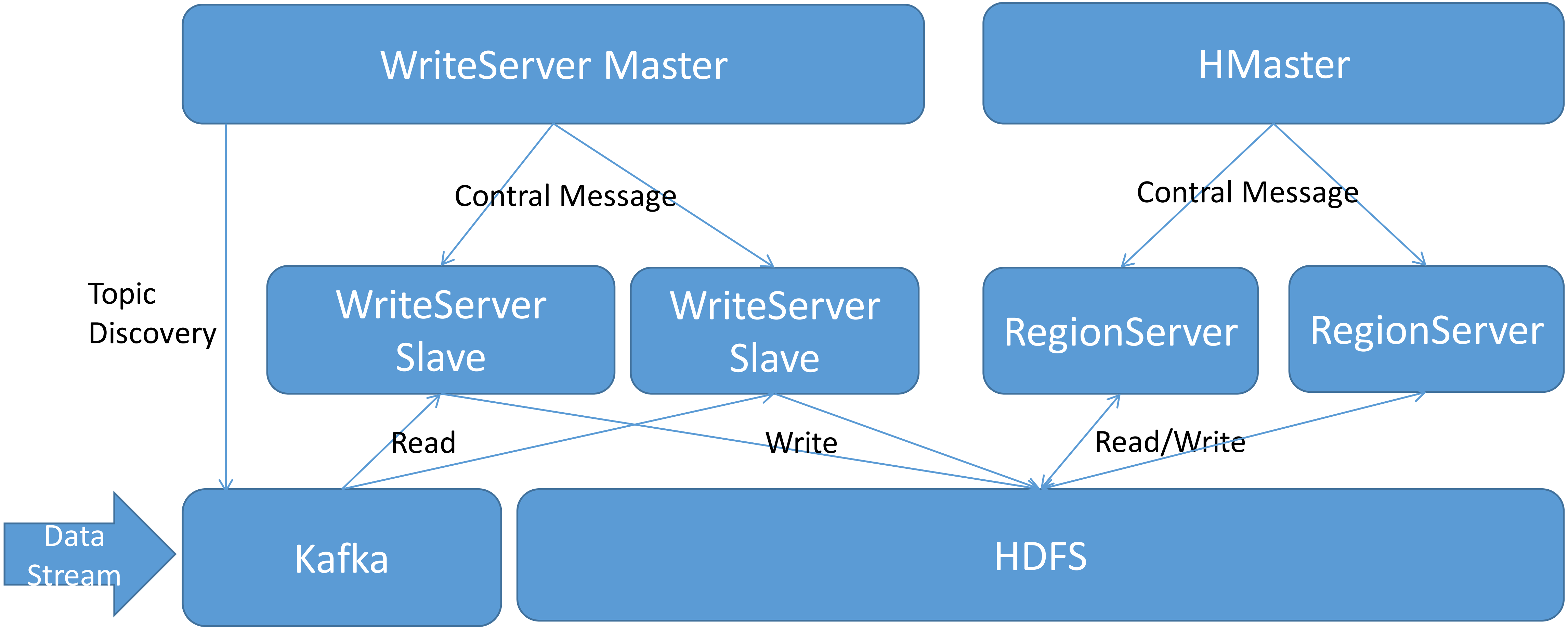
Read-Write split data loading

FiberHome



- Independent WriterServer to handle put request
- RegionServer only handle read request
- WriteServer write HFile on HDFS, send do-bulkload operation.
- Several minutes delay between put and data readable.

Architecture



WriteServer Master

Topic Management

- Discover new kafka topics
- Receive loading request
- Loading records statistic

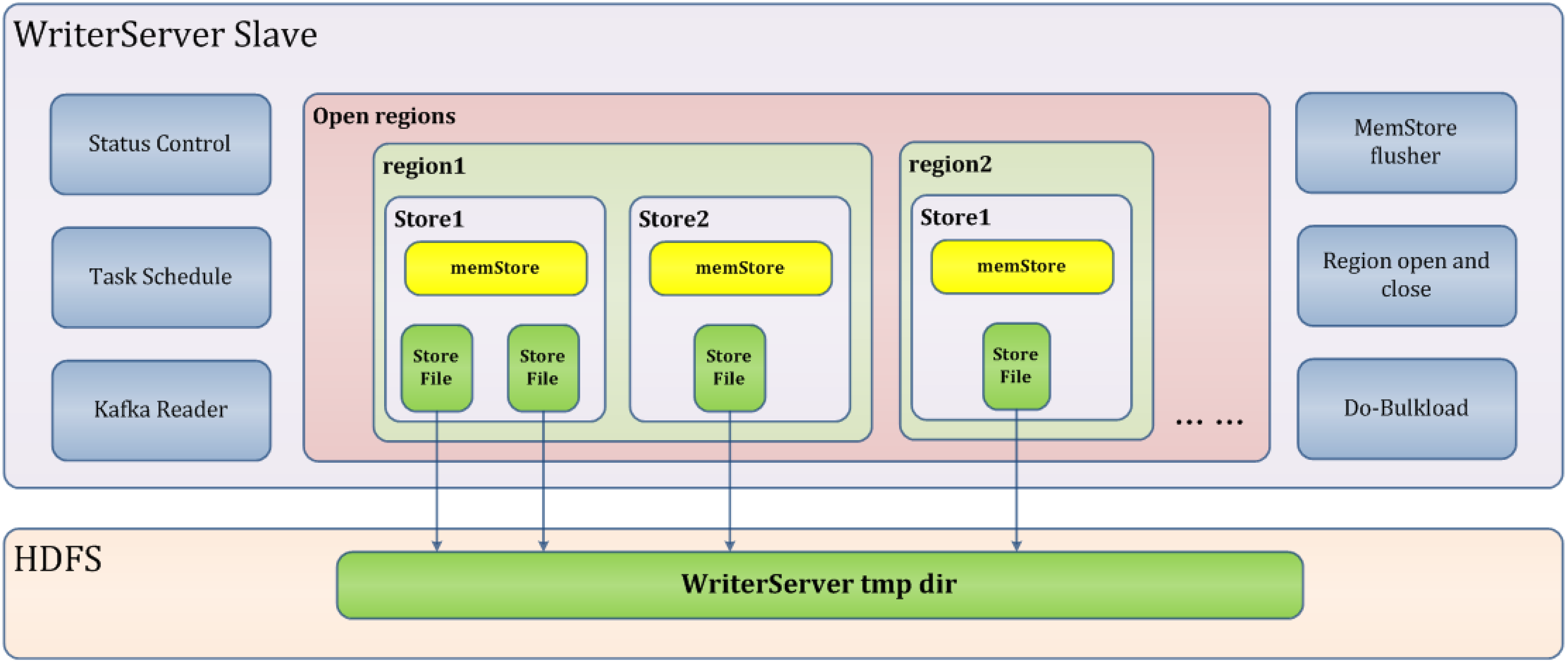
Task Management

- Create new loading tasks every five minutes or every 10,000 records
- Find a slave to load the task
- Task status control

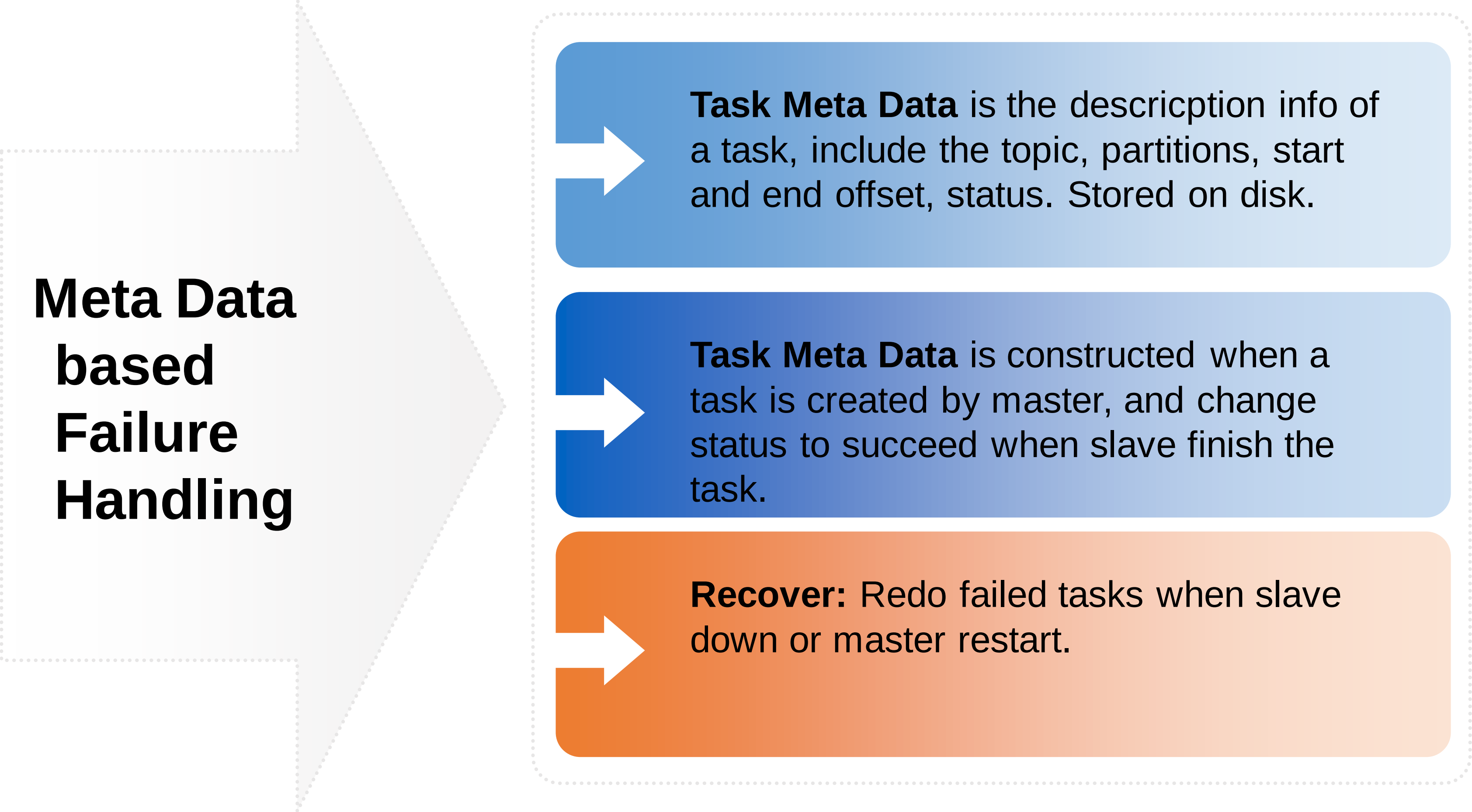
Slaves Management

- Slave status report to master
- Balance
- failover

WriteServer Slave



Failure Handling



Meta Data based Failure Handling

Task Meta Data is the descricption info of a task, include the topic, partitions, start and end offset, status. Stored on disk.

Task Meta Data is constructed when a task is created by master, and change status to succeed when slave finish the task.

Recover: Redo failed tasks when slave down or master restart.

Contents

1 Motivation

2 Solution

3 Optimization

4 Tests

5 Summarize

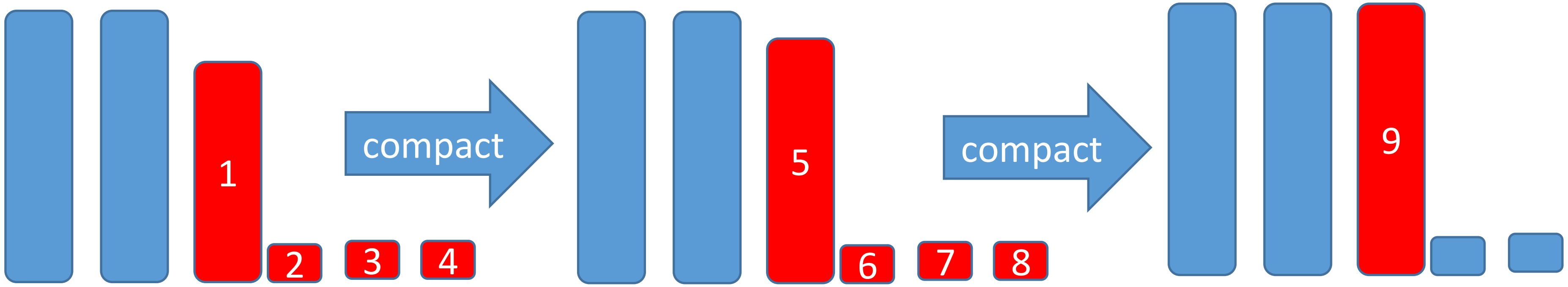
Balance

Load balance according tasks:

- Send new tasks to slaves with less tasks on handling
- Try to send tasks of one topic to a few fixed slaves
  - avoid one region open everywhere
  - Less region open, less small files
- Keep region opened for a while, even there are no tasks
  - avoid region open/close too frequently

Compact

- Small files with higher priority
- Avoid one large file together with many small files compact again and again



StoreEngine

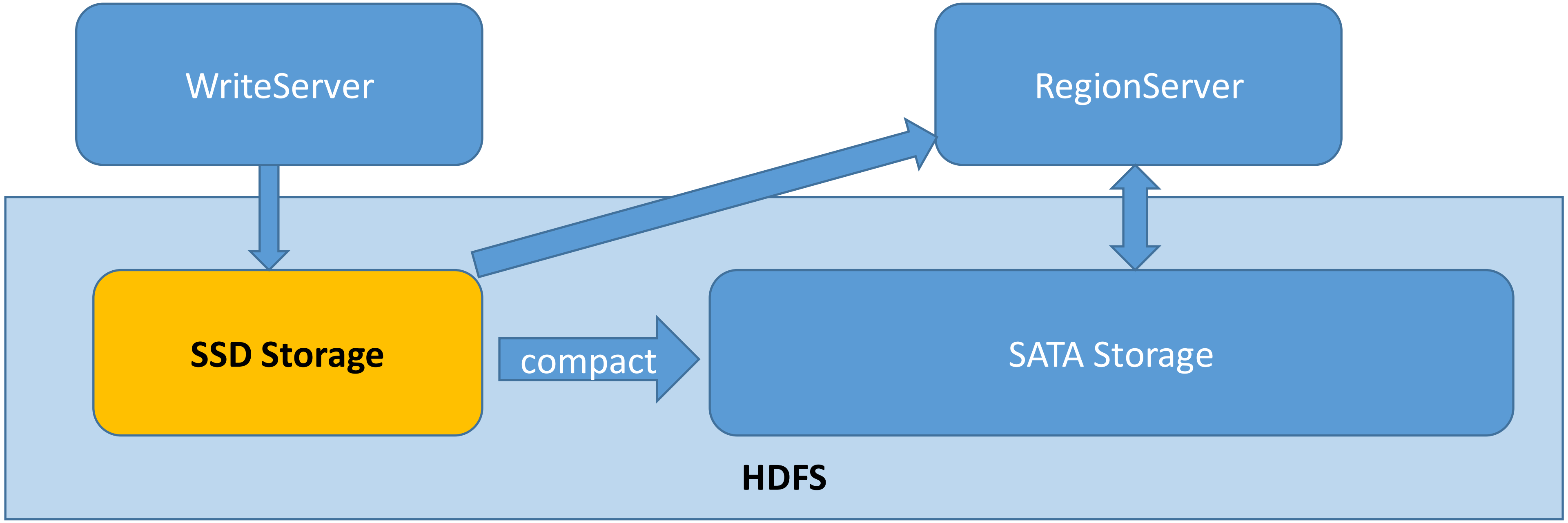
Customized store engine:

- organize store files in two queues
  - one can be read and compact
  - The other can only be compact
  - If there are too many files, new file will not be readable until they are compact
- **Some new files discovered later better than all files can not be read before time out**
  - Occasionally data explosion can be handled
  - Region need split
  - “Hot key” should be handled

HDFS Heterogeneous Storage Usage

FiberHome

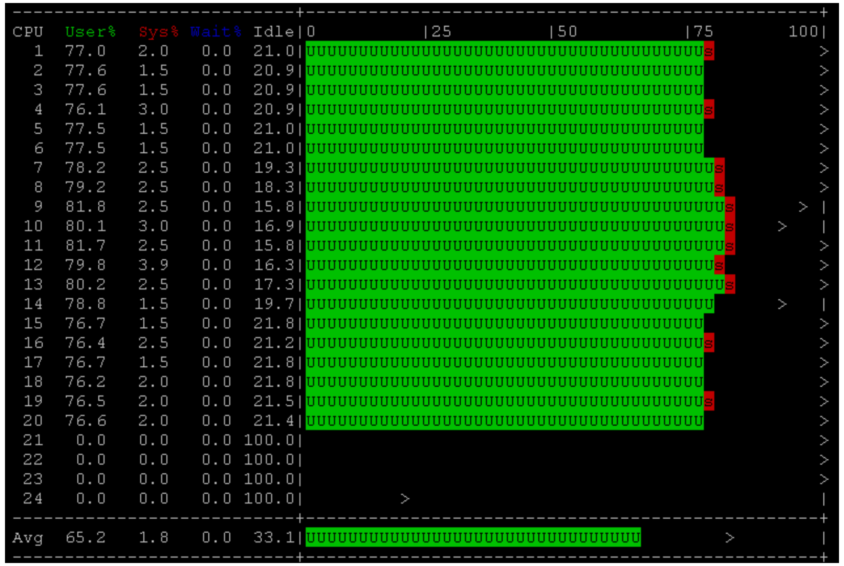
- **Use SSD storage as WriteServer tmp dir**
- **Use SATA as HBase data dir storage**
  - WriteServer write HFile on SSD
  - Load HFile to HBase(Only move)
  - Change to SATA storage after compact by regionServer



Resource Control

Resource used by WriteServer should be controllable:

- Memory:
  - JVM parameters 30~50GB memory
  - Large Memory Store will avoid small files
  - Too Large memory store will cause gc problems
- **CPU:**
  - Slave can use 80% cpu cores at most
  - Compare to real-time data load, a big optimize is we can control the cpu occupation by write operations.



```
CPU  User%   Sys%  Wait%  Idle|0           |25          |50          |75         100|
  1  77.0    2.0    0.0   21.0|
  2  77.6    1.5    0.0   20.9|
  3  77.6    1.5    0.0   20.9|
  4  76.1    3.0    0.0   20.9|
  5  77.5    1.5    0.0   21.0|
  6  77.5    1.5    0.0   21.0|
  7  78.2    2.5    0.0   19.3|
  8  79.2    2.5    0.0   18.3|
  9  81.8    2.5    0.0   15.8|
 10  80.1    3.0    0.0   16.9|
 11  81.7    2.5    0.0   15.8|
 12  79.8    3.9    0.0   16.3|
 13  80.2    2.5    0.0   17.3|
 14  78.8    1.5    0.0   19.7|
 15  76.7    1.5    0.0   21.8|
 16  76.4    2.5    0.0   21.2|
 17  76.7    1.5    0.0   21.8|
 18  76.2    2.0    0.0   21.8|
 19  76.5    2.0    0.0   21.5|
 20  76.6    2.0    0.0   21.4|
 21   0.0    0.0    0.0  100.0|
 22   0.0    0.0    0.0  100.0|
 23   0.0    0.0    0.0  100.0|
 24   0.0    0.0    0.0  100.0|
------------------------------+
Avg  65.2    1.8    0.0   33.1|
```

Contents

1 Motivation

2 Solution

3 Optimization

4 Tests

5 Summarize

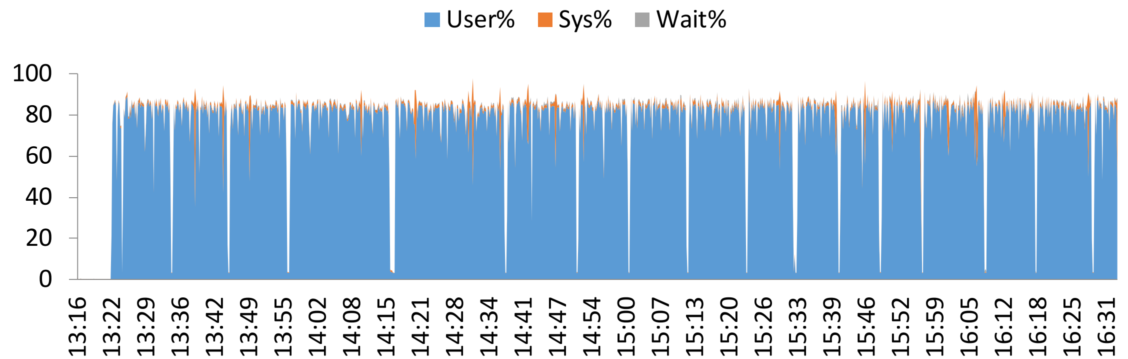
Loading Performance

WriterServer Slave	
CPU	Intel(R) Xeon(R) CPU E5-2640 v2 @ 2.00GHz
Memory	128G
Disk	1TB SSD * 4
Network	10GE
Record size	1KB
Compress	Snappy
Performance	**300,000 records/s**

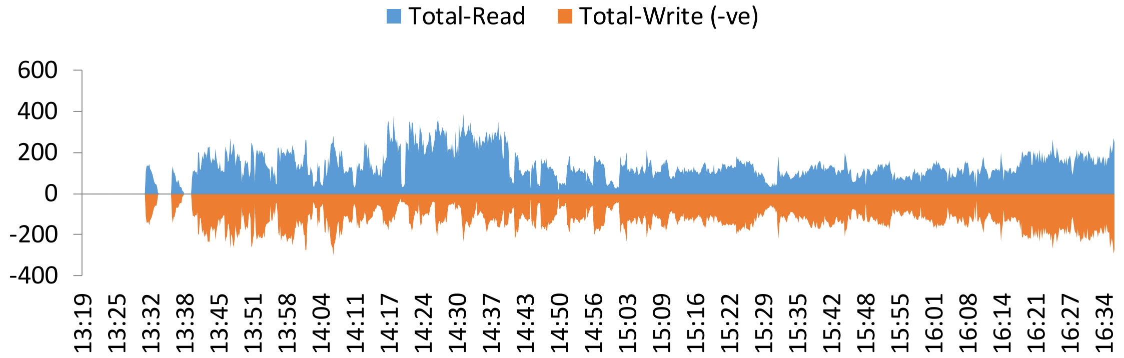
One WriteServer slave can match 5 RegionServer’s loading requirements before RegionServer reach compact limitation.

Resource Performance

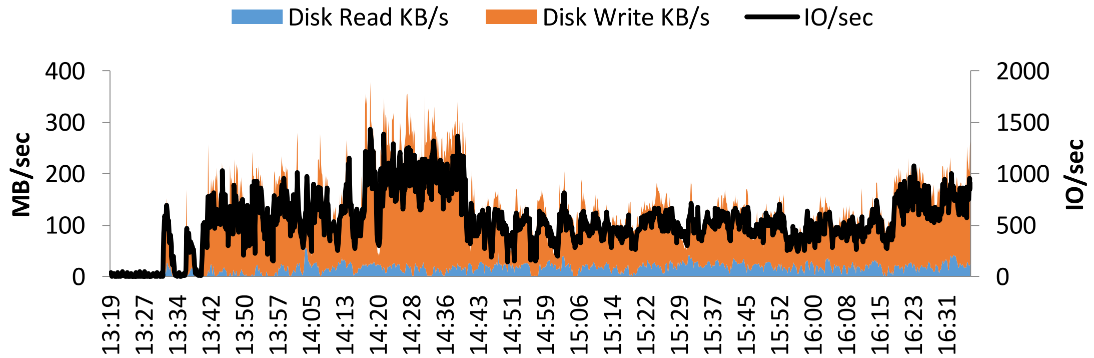
CPU Total WS-Slave5 – 2017/2/17



Network I/O WS-Slave5 (MB/s) - 2017/2/17



Disk total MB/s WS-Slave5- 2017/2/17



Memory

JVM: aways use memory as much as assigned

GC: config gc policy to avoid full gc.

Contents

1 Motivation

2 Solution

3 Optimization

4 Tests

5 Summarize

Summarize

FiberHome

We proposed an read-write split near-line loading method and architecture:

- Increase loading performance
- Control resource used by write operation, make sure read operation can not be starved
- Provide an architecture corresponding with kafka and hdfs
- Provide some optimize method, eg: compact, balance, etc.
- Provide test result

FiberHome

Questions ?

Thanks