Building Online HBase Cluster of Zhihu Based on Kubernetes

知乎 白志勇

Agenda

- HBase at Zhihu
- Using Kubernetes
- HBase Online Platform

- **HBase at Zhihu**
- Using Kubernetes
- HBase Online Platform

知乎
www.zhihu.com

HBase at Zhihu

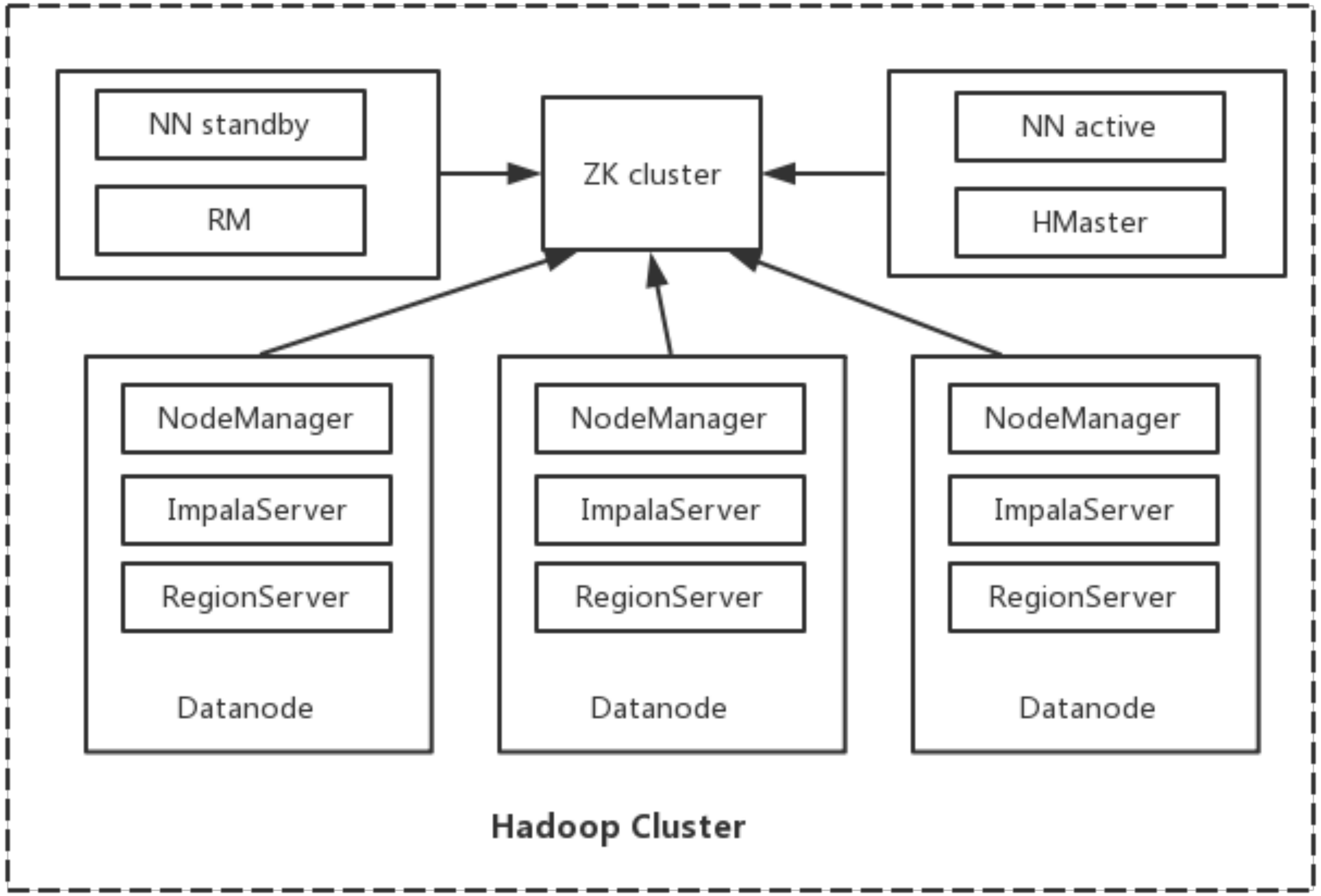
- Offline
  - Physical machine, more than 200 nodes.
  - Working with Spark/Hadoop.
- Online
  - Based on Kubernetes, more than 300 containers.

Our online storage

- **mysql**
  - used in most business
  - some need scale, some need transform
  - all SSD， expensive
- **Redis**
  - cache and partial storage
  - no shard
  - expensive
- **HBase / Cassandra / Rocksdb etc. ?**

At the beginning

- All business at one big cluster
- Also runs Nodemanager and ImpalaServer
- Basically operation
- Physical node level monitor



What we want

- **From Business Sight**
  - environment isolation
  - SLA definition
  - business level monition
- **From Operation Sight**
  - balance resource (CPU, I/O, RAM)
  - friendly api
  - controllable costs

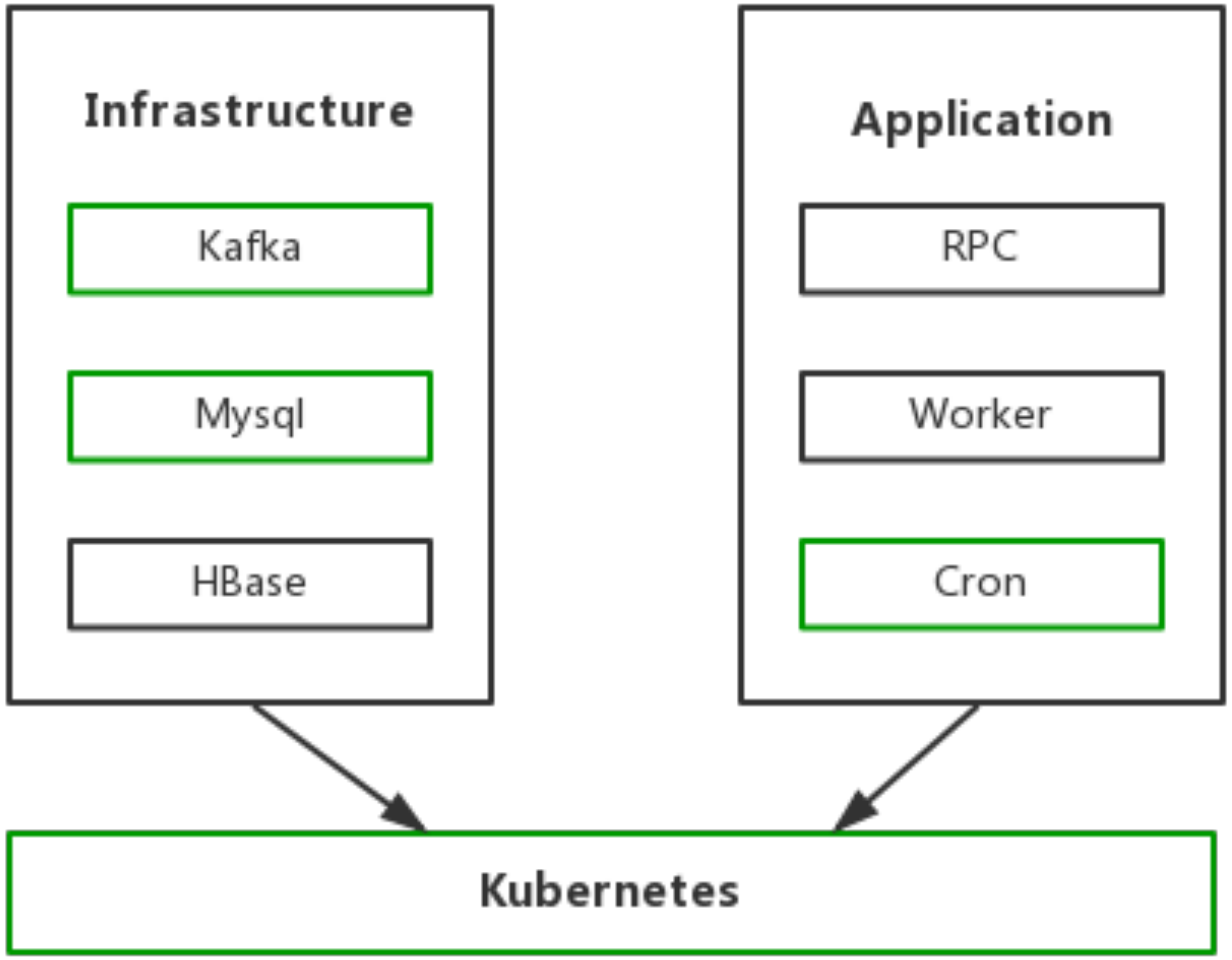
In one word:

Make HBase as a Service.

- HBase at Zhihu
- Using Kubernetes
- HBase Online Platform

知乎
www.zhihu.com

Zhihu’s Unified Cluster Manage Platfom



Kubernetes

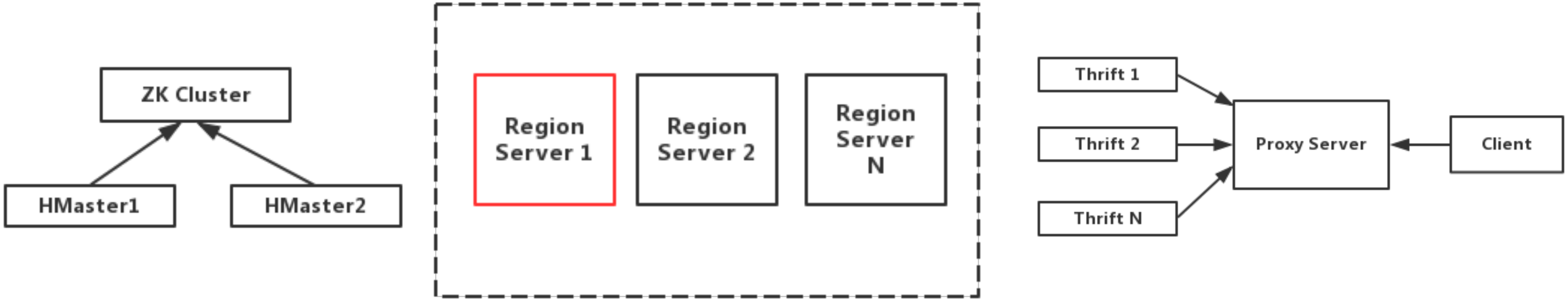
- Cluster resource manager and scheduler
- Using container to isolate resource
- Application management
- Perfect API and active community

Failover Design

- Component Level
- Cluster Level
- Data Replication

Component Level

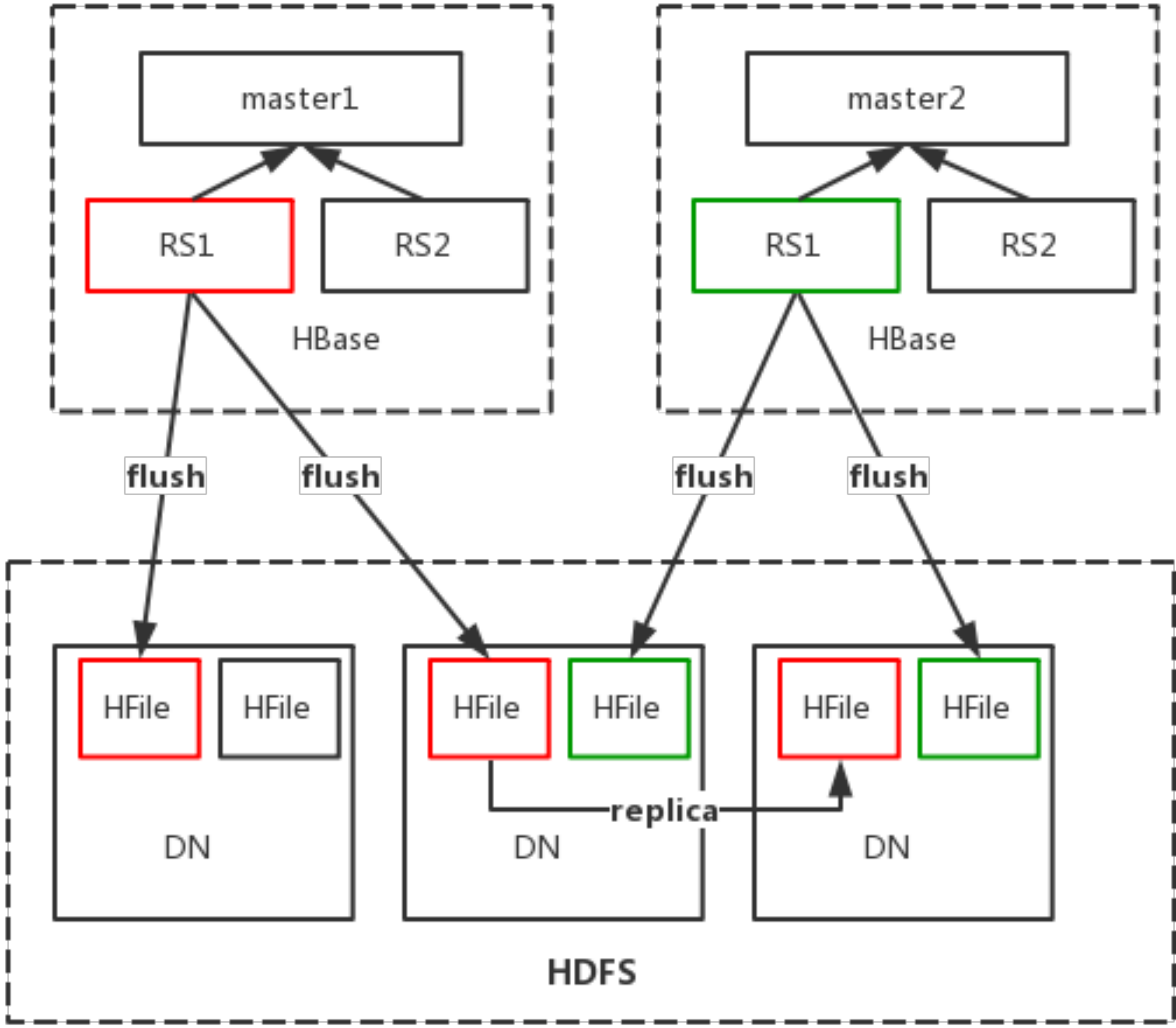
- HMaster -> use ZooKeeper
- RegionServer -> Stateless designed
- ThriftServer -> use proxy
- HFile -> ???



Component Level - HFile

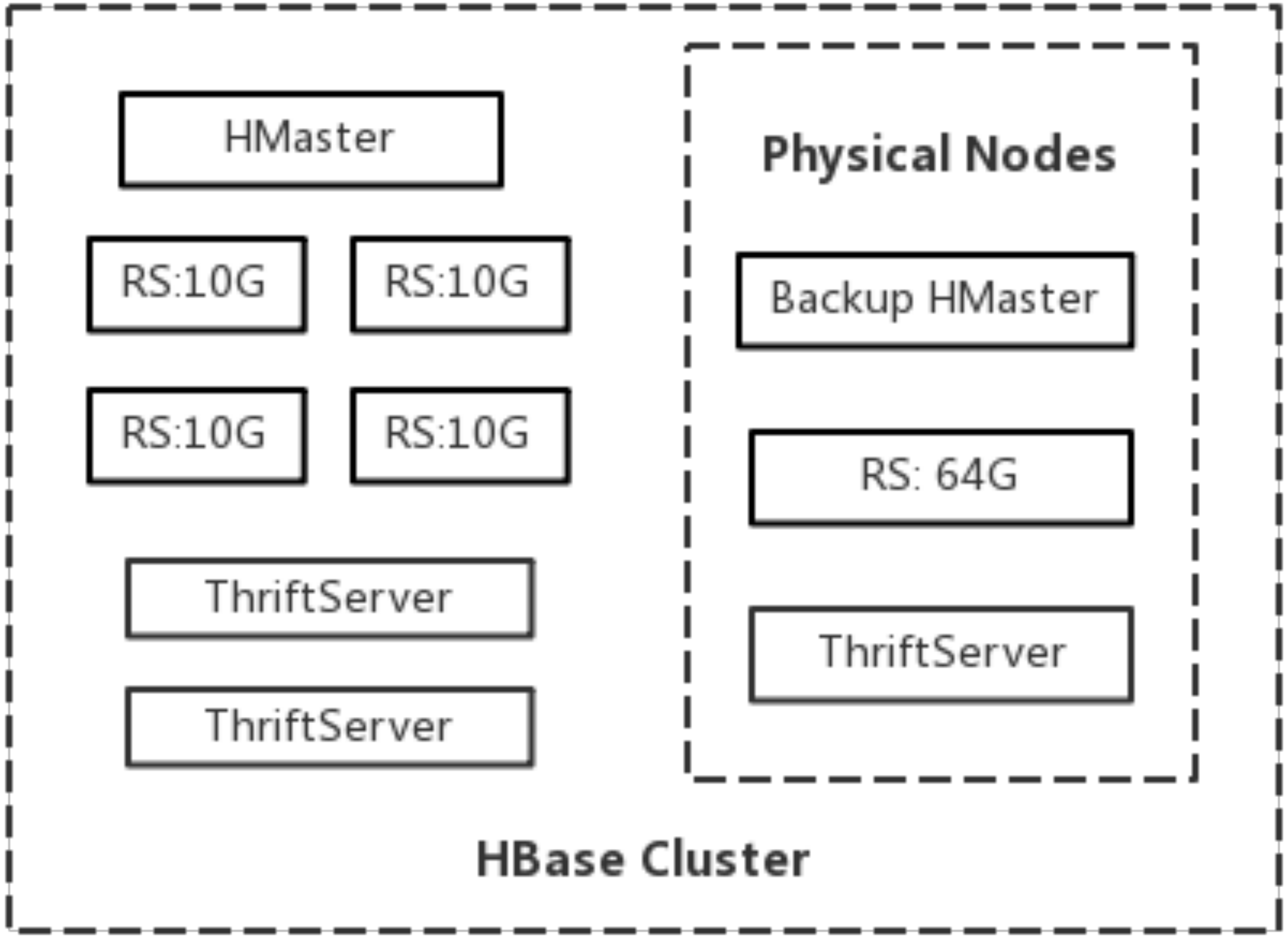
- Shared HDFS Cluster

- Keep the whole cluster stateless



Cluster Level

- What if cluster is down ?
    - Component -> Kubernetes ReplicationSet
- What if Kubernetes is down ?
    - Mixed deployment
    - Few physical nodes with high CPU && RAM



Data Replication

- **Replication in cluster**
  - HDFS built in (3 replicas)
- **Replication between clusters**
  - snapshot + bulk load
  - HBase replication
  - Offline cluster doing MR / Spark

- HBase at Zhihu
- Using Kubernetes
- **HBase Online Platform**

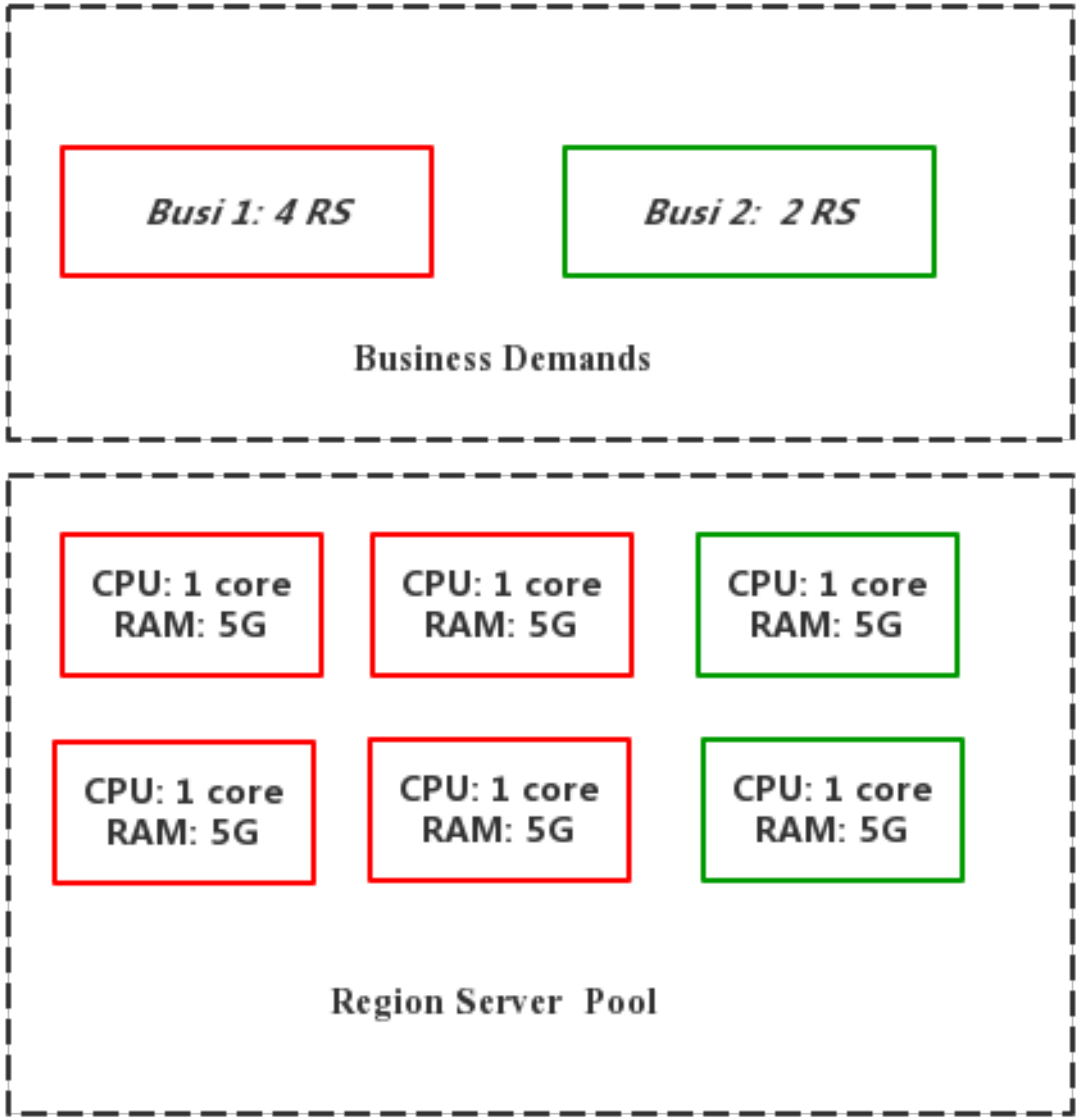
知乎
www.zhihu.com

Physical Node Resource

- CPU: 2 * 12 core
- Memory: 128 G
- Disk: 4 T

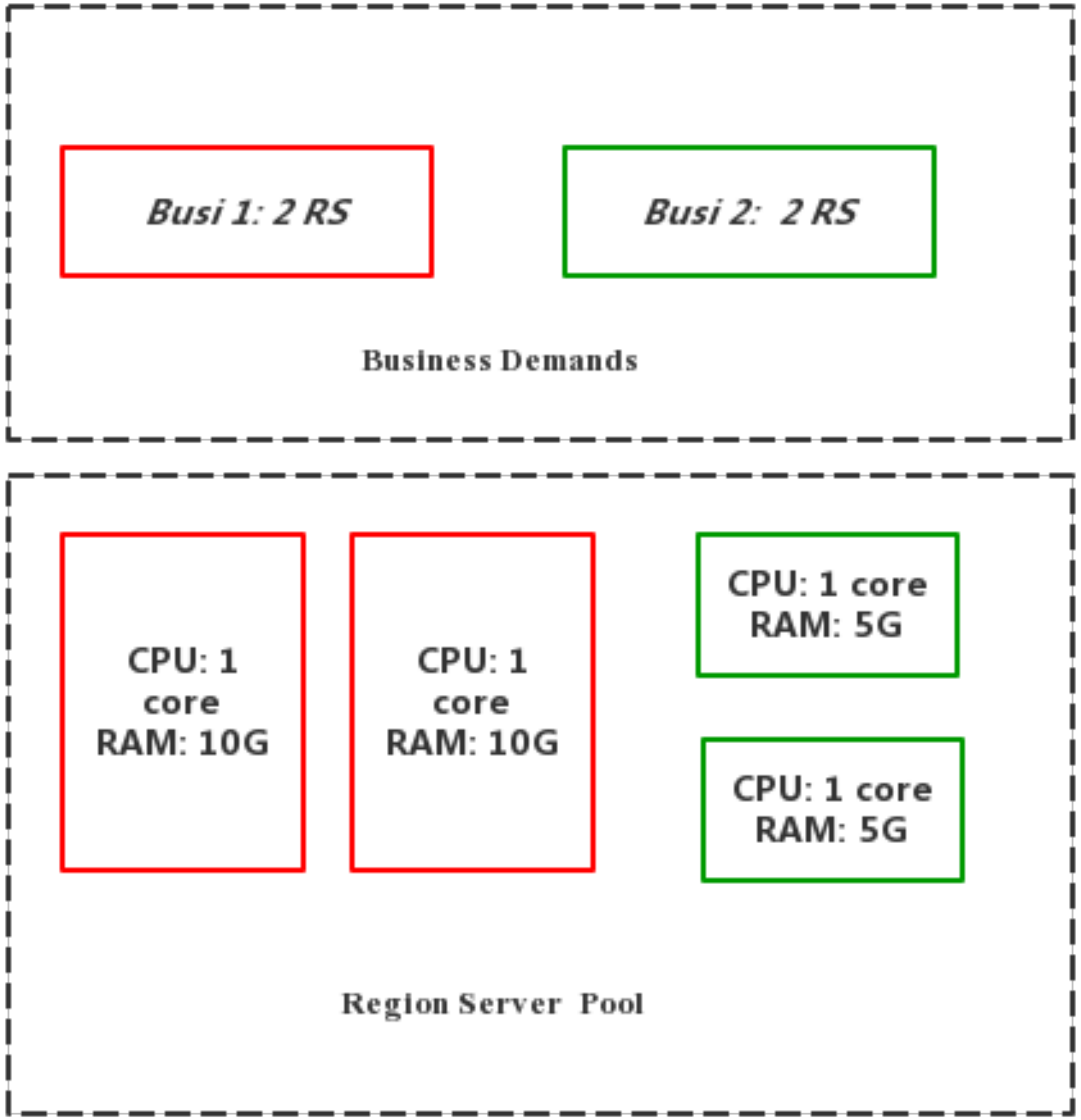
Resource Definition (1)

- Minimize the resource
- Business scaled by number of containers
- Pros
  - reduce resource wasted per node
  - simplified debug
- Cons
  - minimum resource not easy to define by business
  - hardly tune params for RAMs and GC



Resource Definition (2)

- Customize container resource by business
- Business scaled by number of containers
- Pros
  - flexible RAM config and tuning (especially non-heap size)
  - used in production



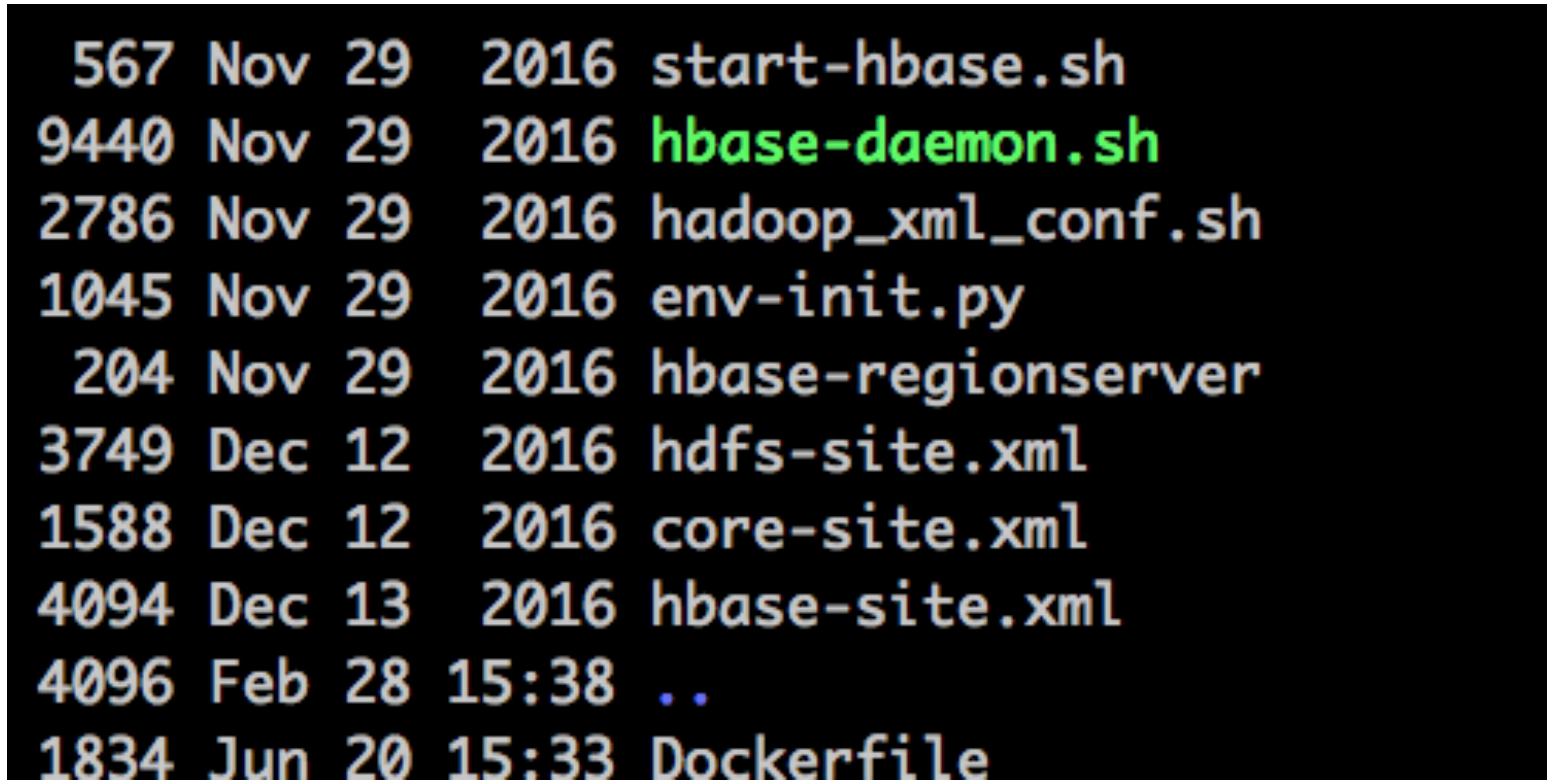
Container Configuration

- Params inject to container via ENV

- Add xml config to container

- Use start-env.sh to init configuration

- Modify params during cluster running is permitted



```
 567 Nov 29  2016 start-hbase.sh
9440 Nov 29  2016 hbase-daemon.sh
2786 Nov 29  2016 hadoop_xml_conf.sh
1045 Nov 29  2016 env-init.py
 204 Nov 29  2016 hbase-regionserver
3749 Dec 12  2016 hdfs-site.xml
1588 Dec 12  2016 core-site.xml
4094 Dec 13  2016 hbase-site.xml
4096 Feb 28 15:38 ..
1834 Jun 20 15:33 Dockerfile
```

知乎
www.zhihu.com

RegionServer G1GC (thanks Xiaomi)

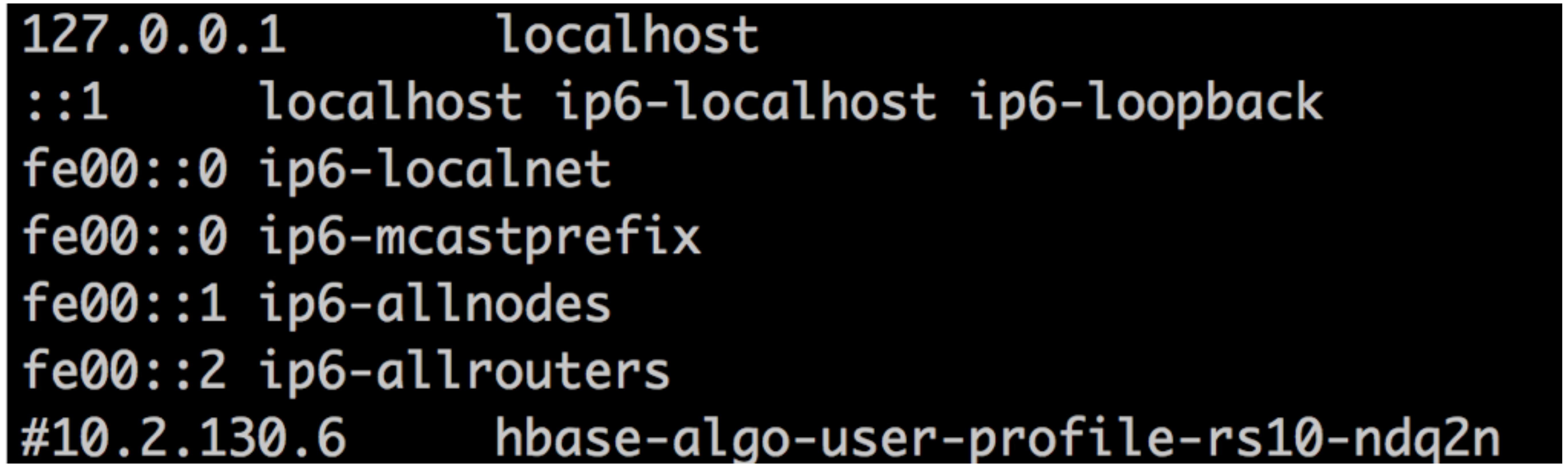
```
-XX:+UnlockExperimentalVMOptions

-XX:MaxGCPauseMillis=50

-XX:G1NewSizePercent=5

-XX:InitiatingHeapOccupancyPercent=45

-XX:+ParallelRefProcEnabled

-XX:ConcGCThreads=2

-XX:ParallelGCThreads=8

-XX:MaxTenuringThreshold=15

-XX:G1OldCSetRegionThresholdPercent=10

-XX:G1MixedGCCountTarget=16

-XX:MaxDirectMemorySize=256M
```

Network

- Dedicated ip per container

- DNS register/deregister automatically

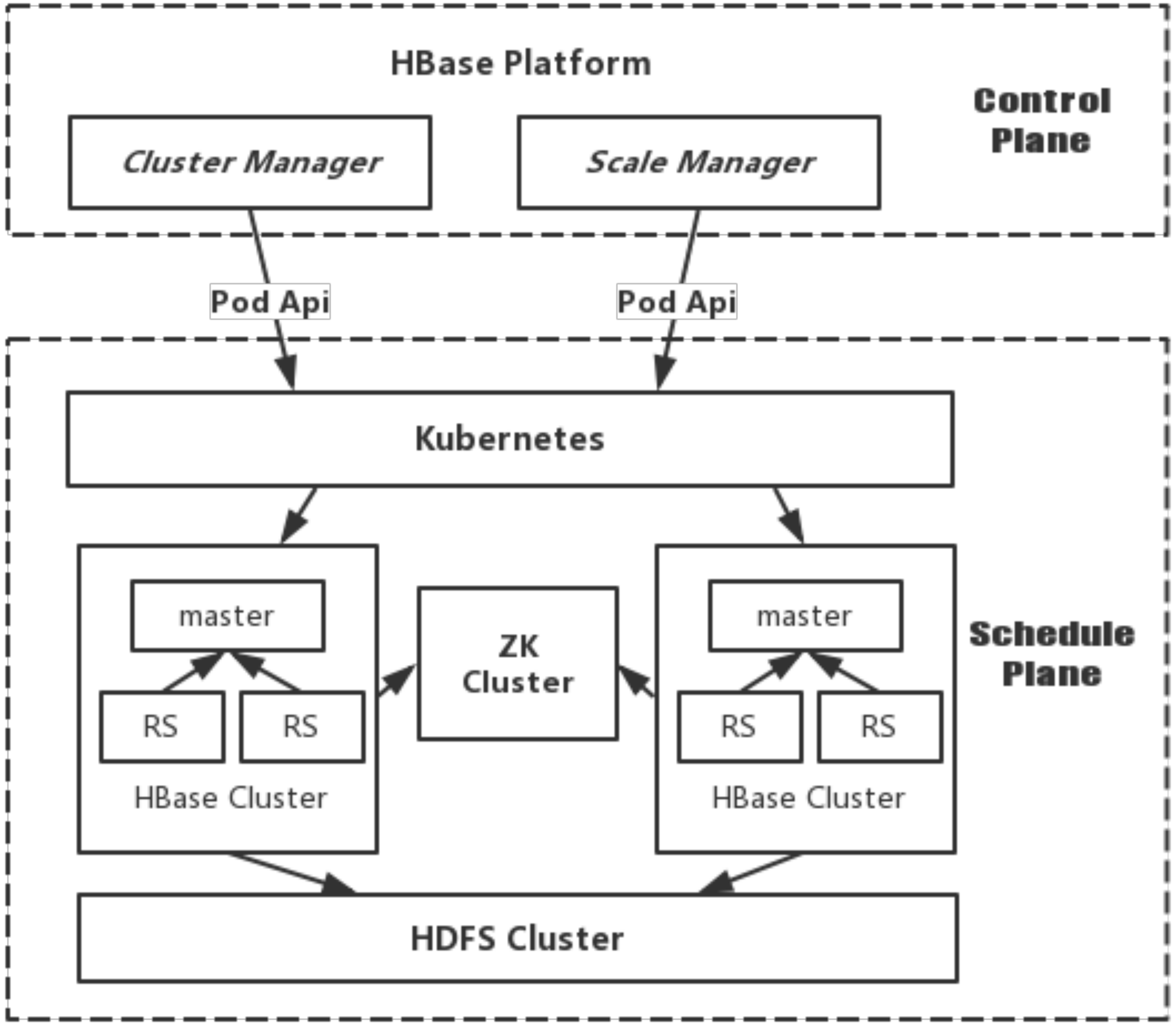
- Modified /etc/hosts for pod



```
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
fe00::0 ip6-mcastprefix
fe00::1 ip6-allnodes
fe00::2 ip6-allrouters
#10.2.130.6     hbase-algo-user-profile-rs10-ndq2n
```

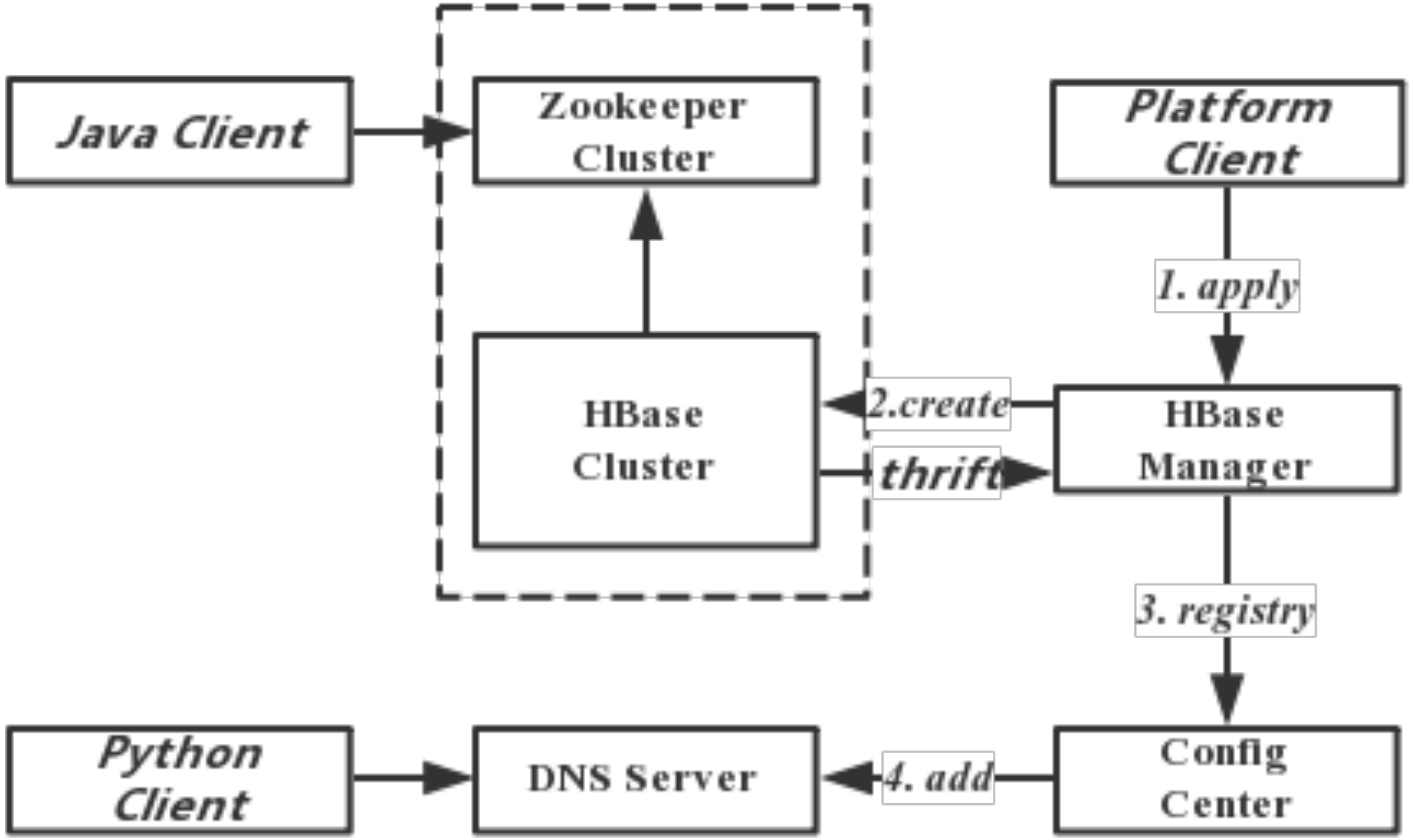
Manage Cluster

- Platform controls cluster
- Kubernetes schedule resources
- Shared HDFS and ZK cluster
- Cons:
  - fully scan still impact whole cluster
  - no locality && short circuit holly



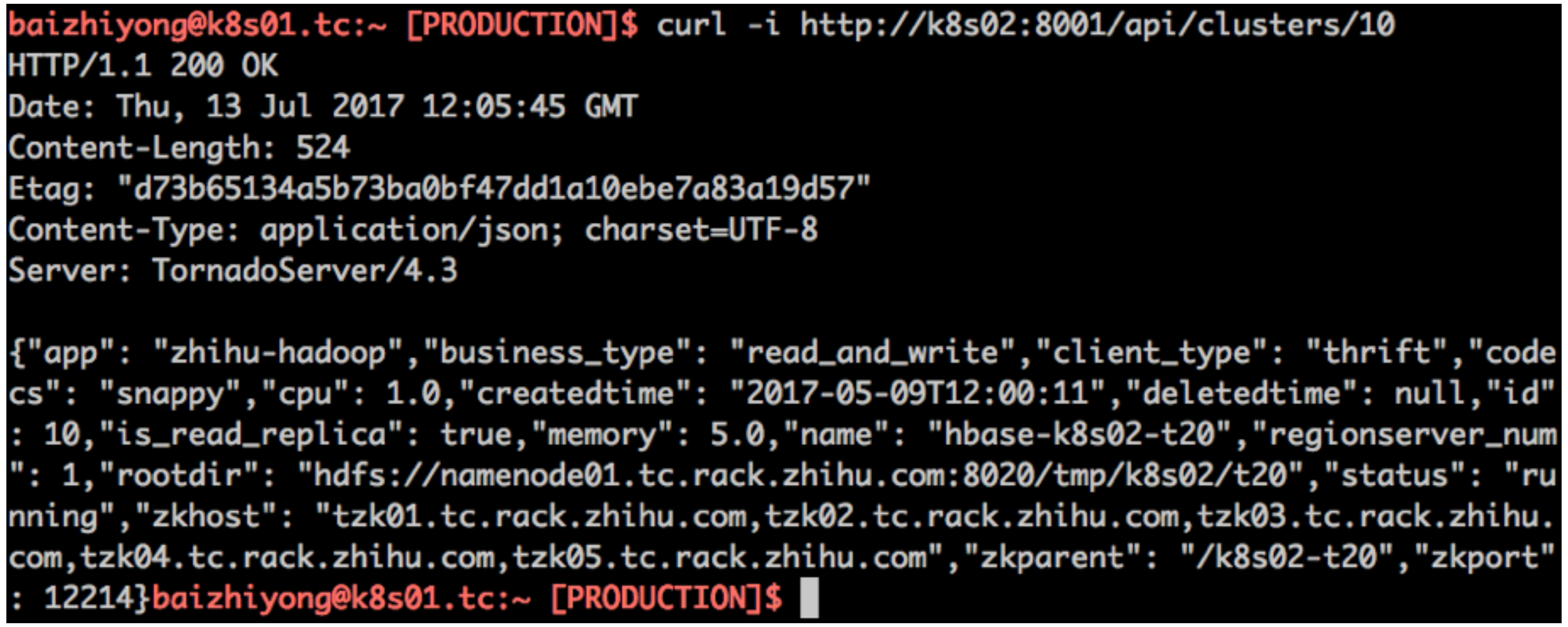
Client Design

- For Java/Scala
    - native HBase client
    - only offer ZK address to business
- For Python
    - happybase
    - client proxy
    - service discovery



API Server

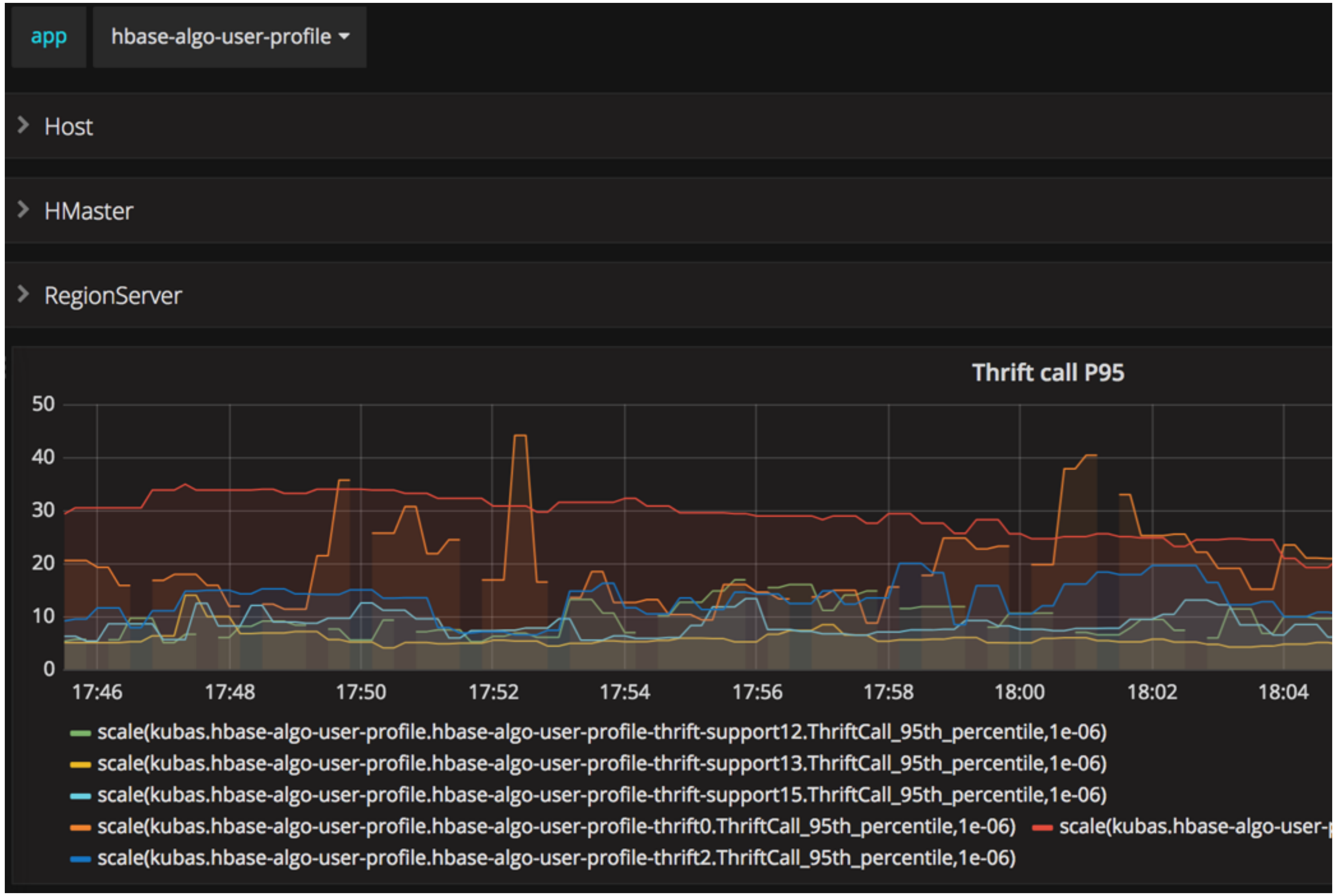
- Bridge between Kubernetes and business user
- Encapsulate component of a HBase cluster
- Restful API
- Friendly interface



```
baizhiyong@k8s01.tc:~ [PRODUCTION]$ curl -i http://k8s02:8001/api/clusters/10
HTTP/1.1 200 OK
Date: Thu, 13 Jul 2017 12:05:45 GMT
Content-Length: 524
Etag: "d73b65134a5b73ba0bf47dd1a10ebe7a83a19d57"
Content-Type: application/json; charset=UTF-8
Server: TornadoServer/4.3

{"app": "zhihu-hadoop","business_type": "read_and_write","client_type": "thrift","code
cs": "snappy","cpu": 1.0,"createdtime": "2017-05-09T12:00:11","deletedtime": null,"id"
: 10,"is_read_replica": true,"memory": 5.0,"name": "hbase-k8s02-t20","regionserver_num
": 1,"rootdir": "hdfs://namenode01.tc.rack.zhihu.com:8020/tmp/k8s02/t20","status": "ru
nning","zkhost": "tzk01.tc.rack.zhihu.com,tzk02.tc.rack.zhihu.com,tzk03.tc.rack.zhihu.
com,tzk04.tc.rack.zhihu.com,tzk05.tc.rack.zhihu.com","zkparent": "/k8s02-t20","zkport"
: 12214}baizhiyong@k8s01.tc:~ [PRODUCTION]$ 
```

Monitor Cluster

- **Physical nodes Level**
  - nodes cpu loads && usage (via IT)
- **Cluster Level**
  - pods cpu loads (via Kubernetes)
  - read && write rate , P95, cacheHit (via JMX)
- **Table Level**
  - client write speed && read latency (via tracing)
  - thrift server (via JMX)
  - proxy concurrency (via DNS/haproxy monitor)


app
hbase-algo-user-profile
Host
HMaster
RegionServer
Thrift call P95
50
40
30
20
10
0
17:46
17:48
17:50
17:52
17:54
17:56
17:58
18:00
18:02
18:04
scale(kubas.hbase-algo-user-profile.hbase-algo-user-profile-thrift-support12.ThriftCall_95th_percentile,1e-06)
scale(kubas.hbase-algo-user-profile.hbase-algo-user-profile-thrift-support13.ThriftCall_95th_percentile,1e-06)
scale(kubas.hbase-algo-user-profile.hbase-algo-user-profile-thrift-support15.ThriftCall_95th_percentile,1e-06)
scale(kubas.hbase-algo-user-profile.hbase-algo-user-profile-thrift0.ThriftCall_95th_percentile,1e-06)
scale(kubas.hbase-algo-user-
scale(kubas.hbase-algo-user-profile.hbase-algo-user-profile-thrift2.ThriftCall_95th_percentile,1e-06)

Current Situation

- 10 online business on platform
- More than 300 containers
- 100% SLA

Benefits

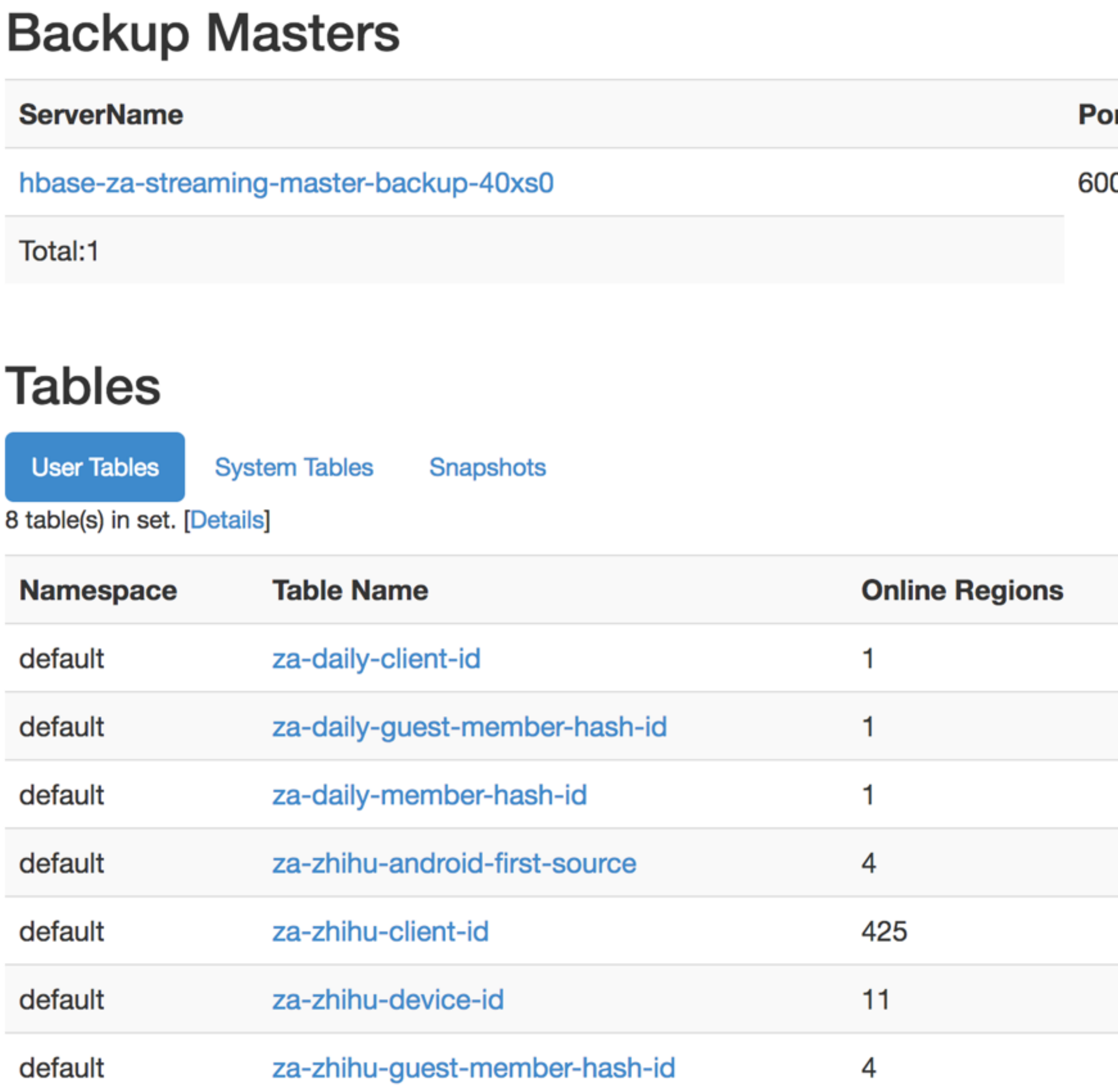
- Easy
- Isolate
- Flexible

Easy

- No code needed
- HBase container publish independently
- Deployment and orchestration straight forward
- Decoupled from physical nodes

Isolate

- table
- thrift
- monitor



Backup Masters

ServerName	Por
hbase-za-streaming-master-backup-40xs0	600
Total:1	

Tables

8 table(s) in set. [Details]

Namespace	Table Name	Online Regions
default	za-daily-client-id	1
default	za-daily-guest-member-hash-id	1
default	za-daily-member-hash-id	1
default	za-zhihu-android-first-source	4
default	za-zhihu-client-id	425
default	za-zhihu-device-id	11
default	za-zhihu-guest-member-hash-id	4

Flexible

- **Muti version**
  - mostly cdh5.5.0-hbase1.0.0
  - one upgrade to 1.2 (HBASE-14283)
  - customize version easily
- **Configuration motivated by business**
  - low latency -> replica read
  - high random read -> closed block cache
  - etc.

Next

- **Enhance performance**
  - Use Netty on ThriftServer
  - Python HBase Client
  - SSD for Datanode
- **Auto scale**
  - by RegionServer number
  - by JVM heap
  - etc.

Thanks!

知乎

www.zhihu.com