Automation of Rolling Upgrade for Hadoop Cluster without Data Loss and Job Failures

May 17, 2017

Hiroshi Yamaguchi & Hiroyuki Adachi

YAHOO! JAPAN

About Us

Hiroshi Yamaguchi

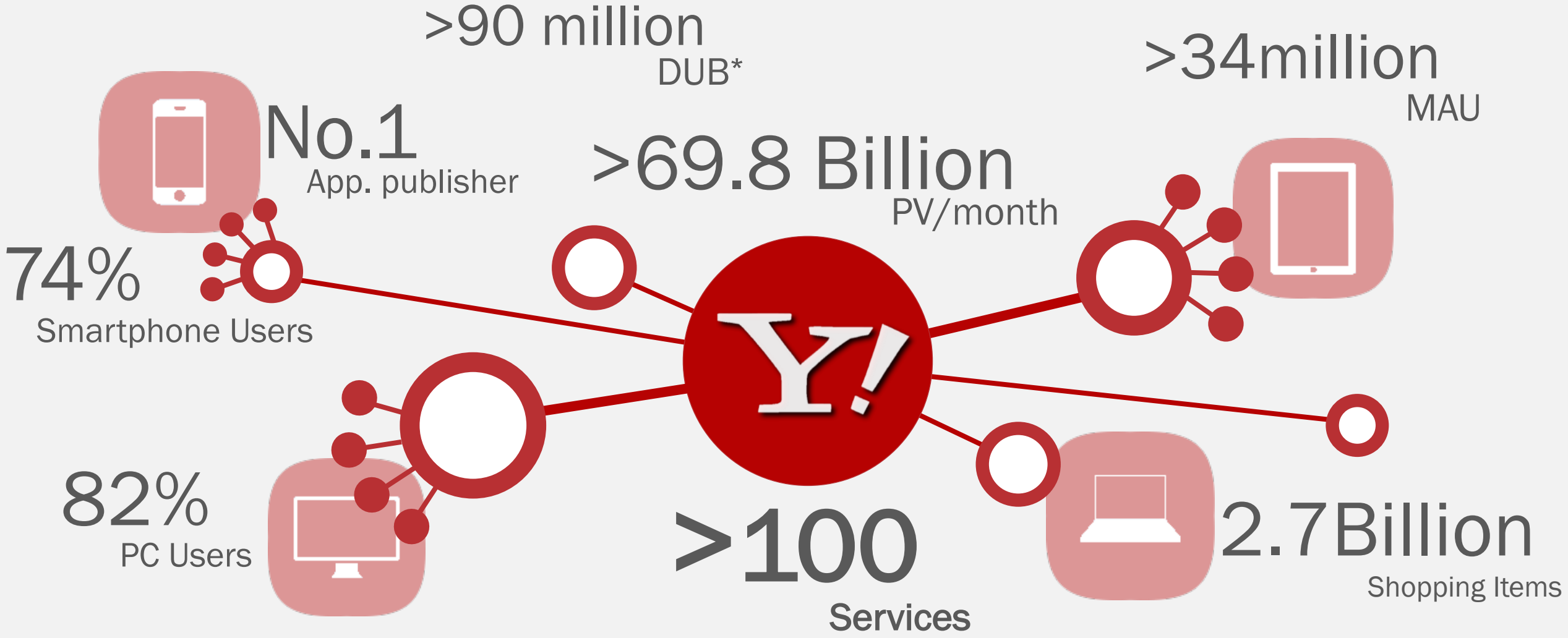
- Hadoop DevOps Engineer

Hiroyuki Adachi

- Hadoop Engineer

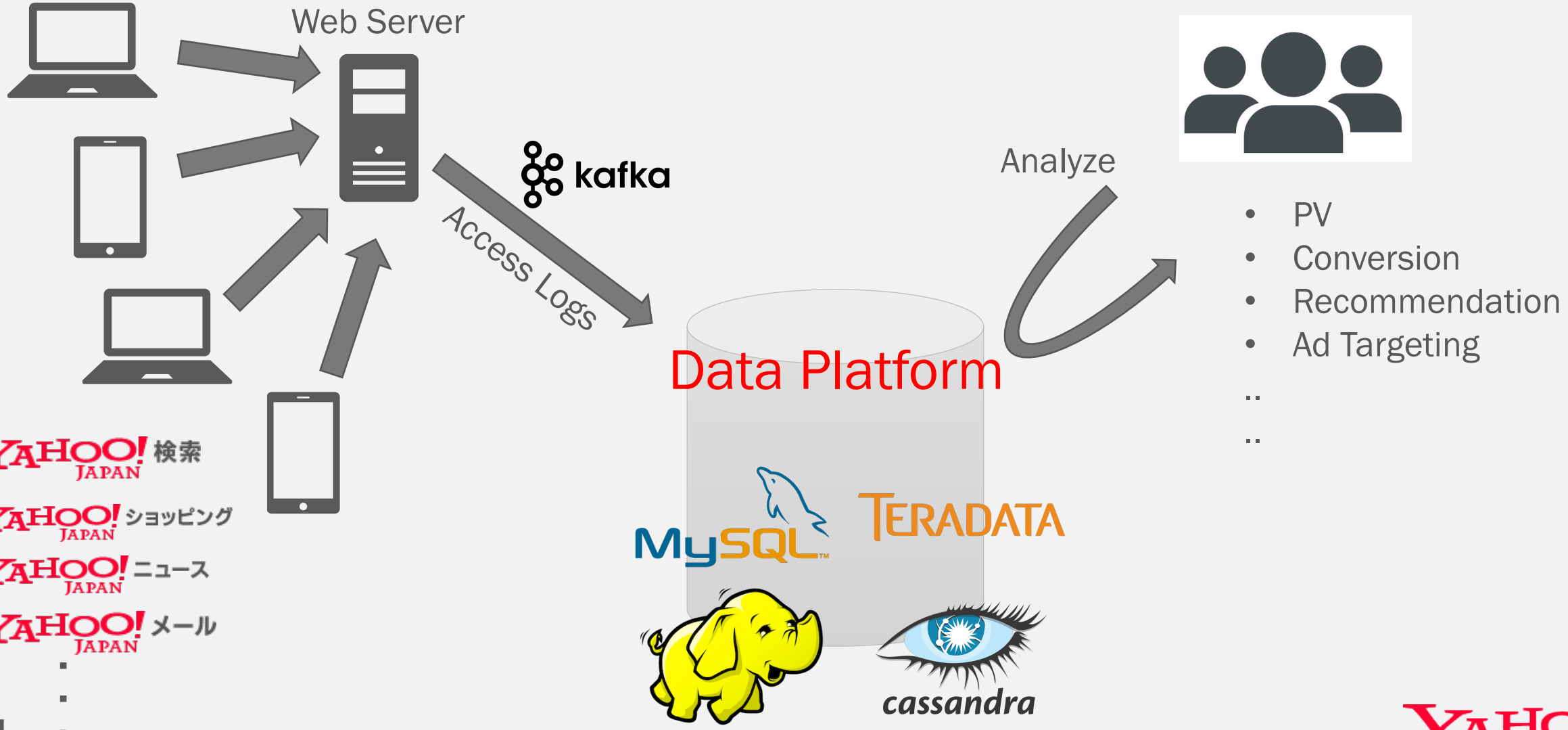
YAHOO! JAPAN

Largest Portal Site in Japan



* Daily Unique Browsers

Overview of Our Data Platform



Agenda

1. Our Hadoop Cluster
2. Issues Involved in Previous Upgrade
3. Upgrade Approach
4. How to Upgrade
5. Results
6. Conclusion

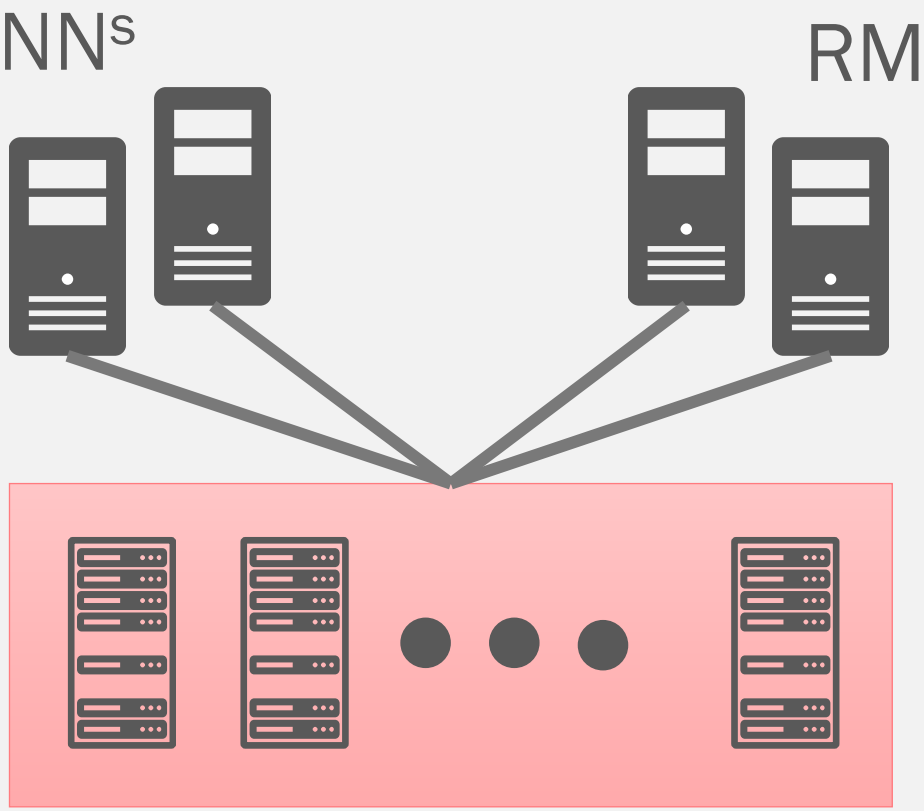
YAHOO! JAPAN

Our Hadoop Cluster

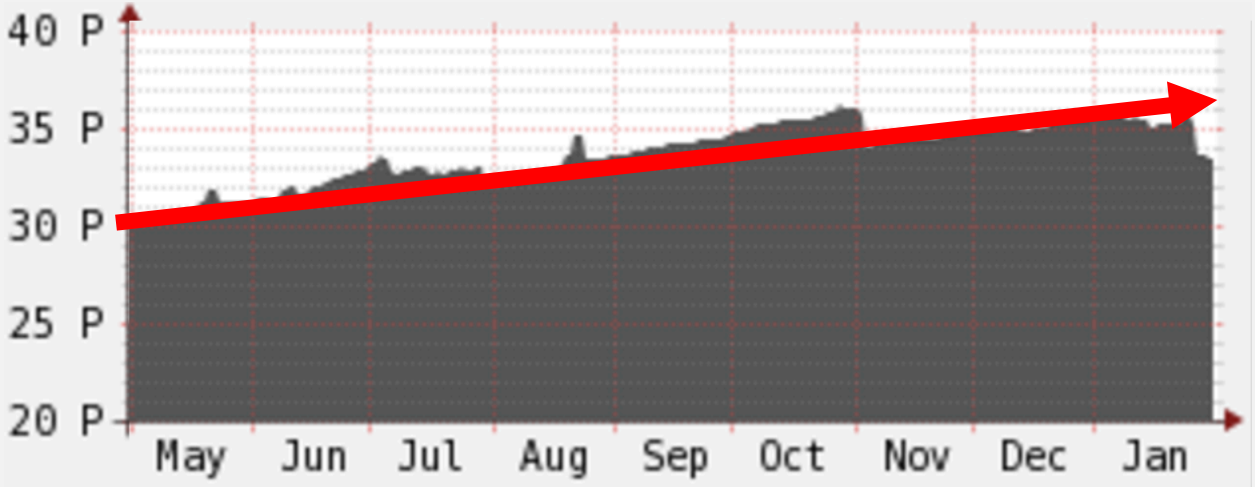
YAHOO! JAPAN

Overview

- HDP 2.3 + Ambari 2.2
- Almost all core components are configured with HA
  - HA: NameNode, ResourceManager, JournalNode, Zookeeper
  - Non HA: Application Timeline Server
- Secured with Kerberos
- 800 slave nodes
  - DataNode/NodeManager
- Other components
  - HiveServer2, Oozie, HttpFS



Data Volume

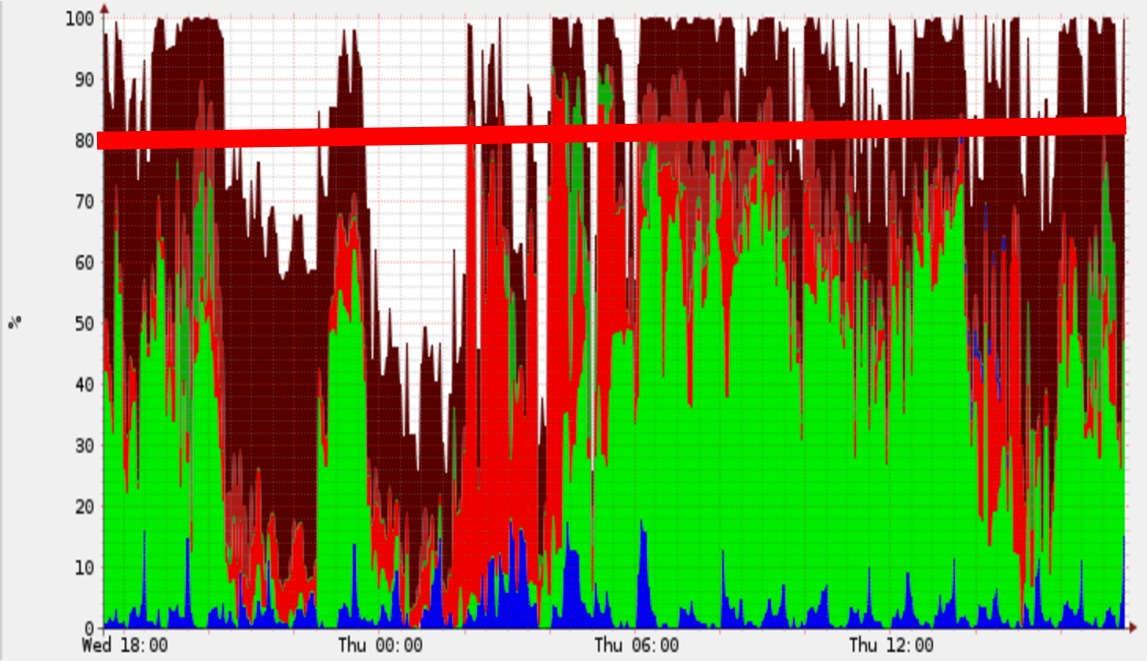


Increase in data volume

- Total stored data 37 PB
- Increases about
  50 ~ 60 TB/day

YAHOO! JAPAN

Work Load



YARN work load of the day

- Multi-tenant cluster
  - ETL / E-Commerce / Advertising / Search ...
  - 20K ~ 30K jobs/day
- Average resource utilization is over 80%
- Data processing needs are growing

YAHOO! JAPAN

Issues Involved in Previous Upgrade

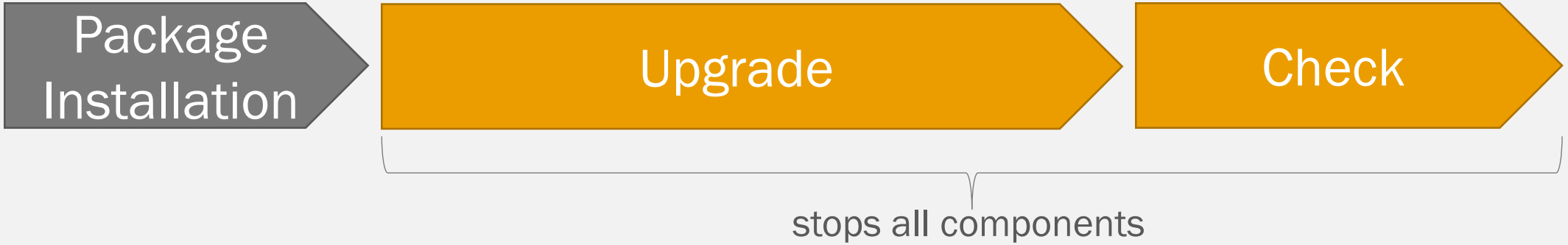
YAHOO! JAPAN

Previous Upgrade

- Performed in Q4 2015
- Without using Ambari
- Manual Express Upgrade including above mentioned steps

YAHOO! JAPAN

Previous Upgrade



- Upgrade
  - Restarting all components for updates to come into effect
- Check
  - Component's normality checks
  - Wait till missing blocks to be fixed

It took 12 hours

YAHOO! JAPAN

Issue 1 : 12 Hours of Cluster Down

- Why took so long ?
  - Large number of slave nodes
  - Multiple nodes failed to start after upgrade
  - Job failed due to data loss
  - Must recover missing blocks

YAHOO! JAPAN

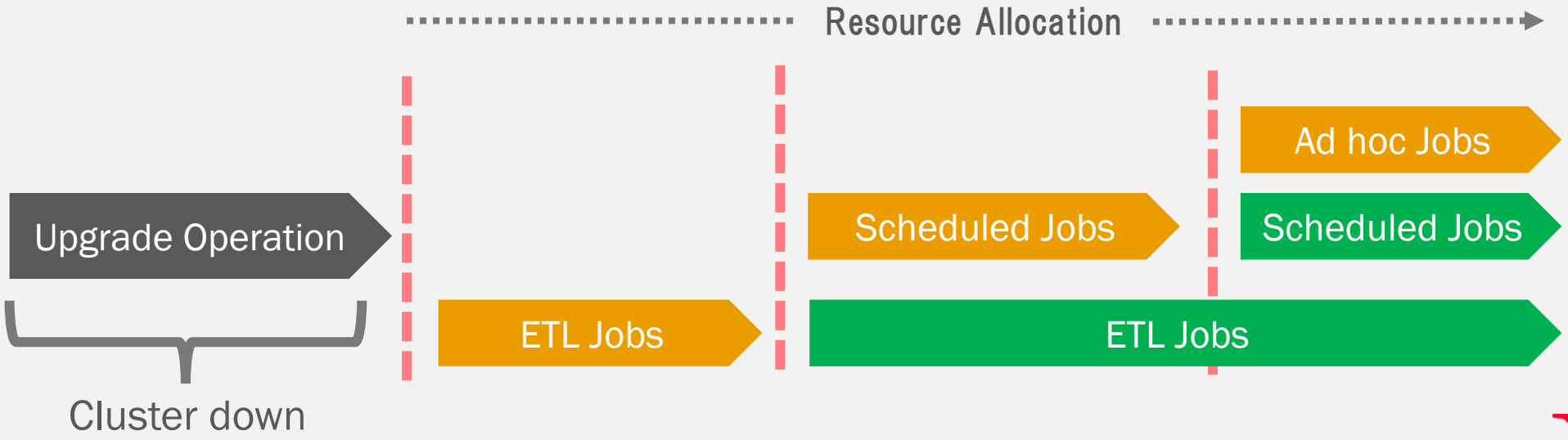
Issue 2 : Finding Right Schedule

- Challenging to find a right schedule
  - Cluster is multi-tenant, shared by hundreds of services
  - Picking a day with minimal impact
    e.g. A day without weekly or monthly running jobs

Photo by: Aflo

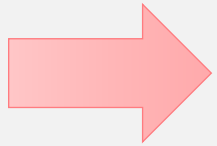
Issue 3 : Coordinating Resource Allocation

- Post-Upgrade : Restarting all jobs simultaneously caused resource exhaustion
- Jobs were recovered by precedence



Issues to be Addressed

- Storage
  - Should not affect HDFS Read/Write
  - Should prevent missing blocks
- Processing
  - To keep Components (Hive, Oozie ...) working



Impact-less upgrade operation

YAHOO! JAPAN

Upgrade Approach

Possible Upgrading Methods

1. Ambari Provided Express Upgrade
2. Ambari Provided Rolling Upgrade
3. Manual Express Upgrade
4. Manual Rolling Upgrade

YAHOO! JAPAN

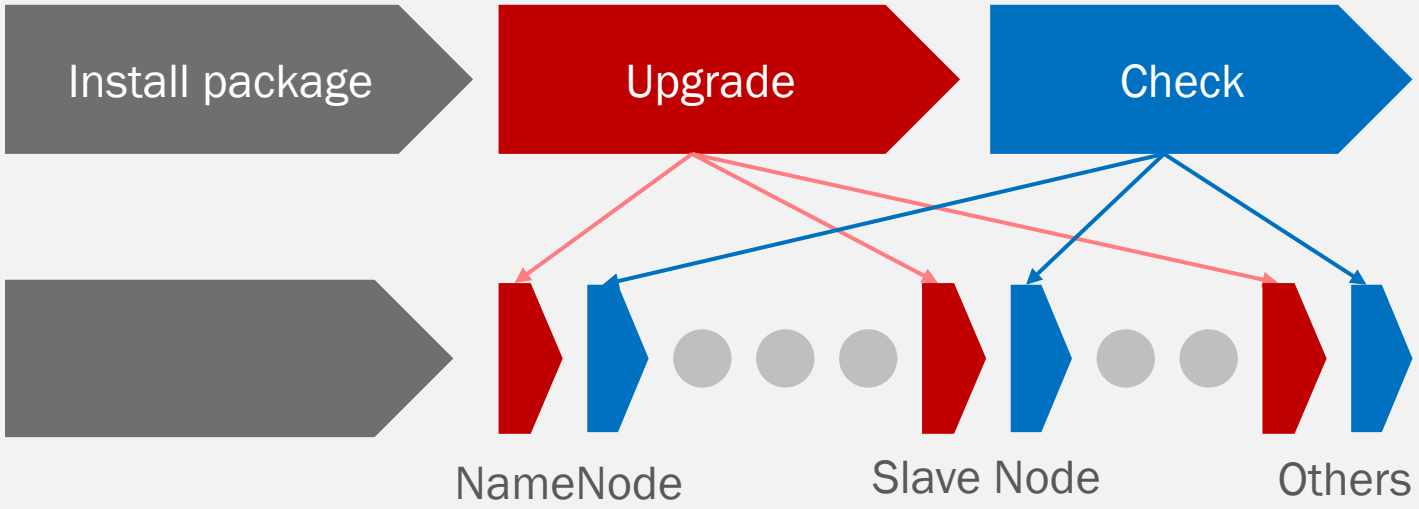
Possible Upgrading Methods

1. Ambari Provided Express Upgrade
2. Ambari Provided Rolling Upgrade
3. Manual Express Upgrade
4. Manual Rolling Upgrade

1’st and 2’nd are not suitable for our environment
3’rd has several issues as explained earlier

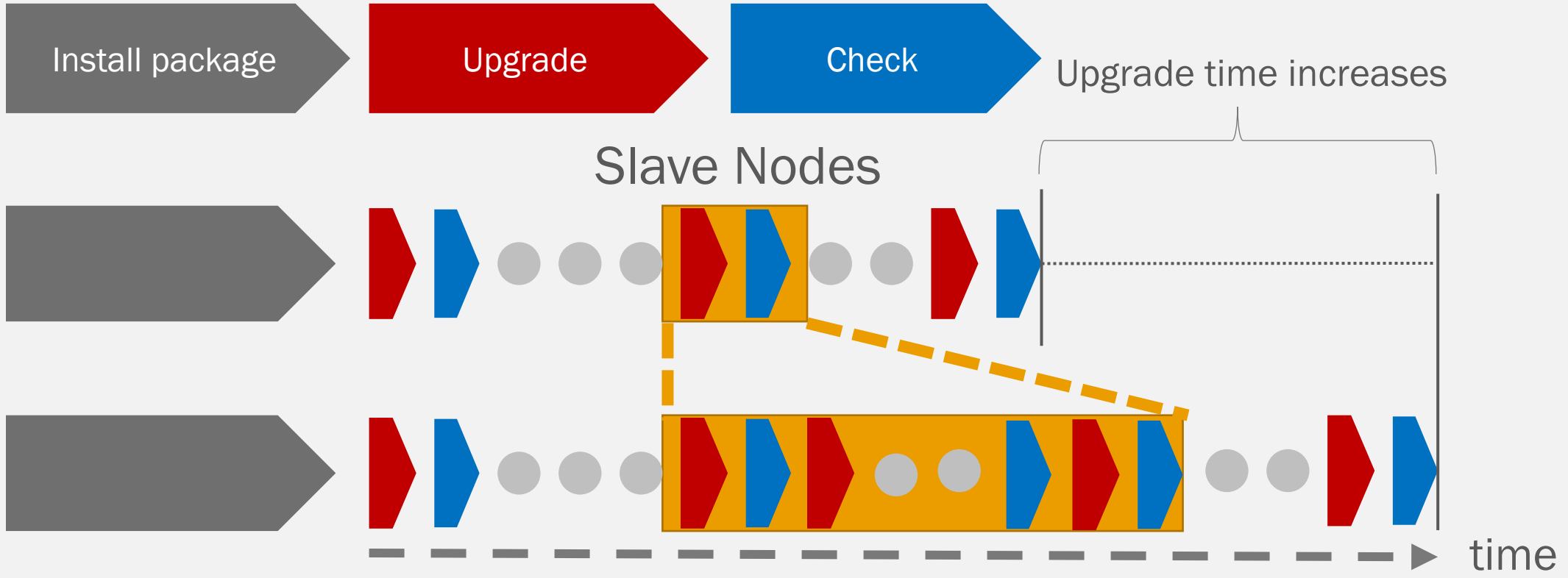
YAHOO! JAPAN

Approaches for Upgrading



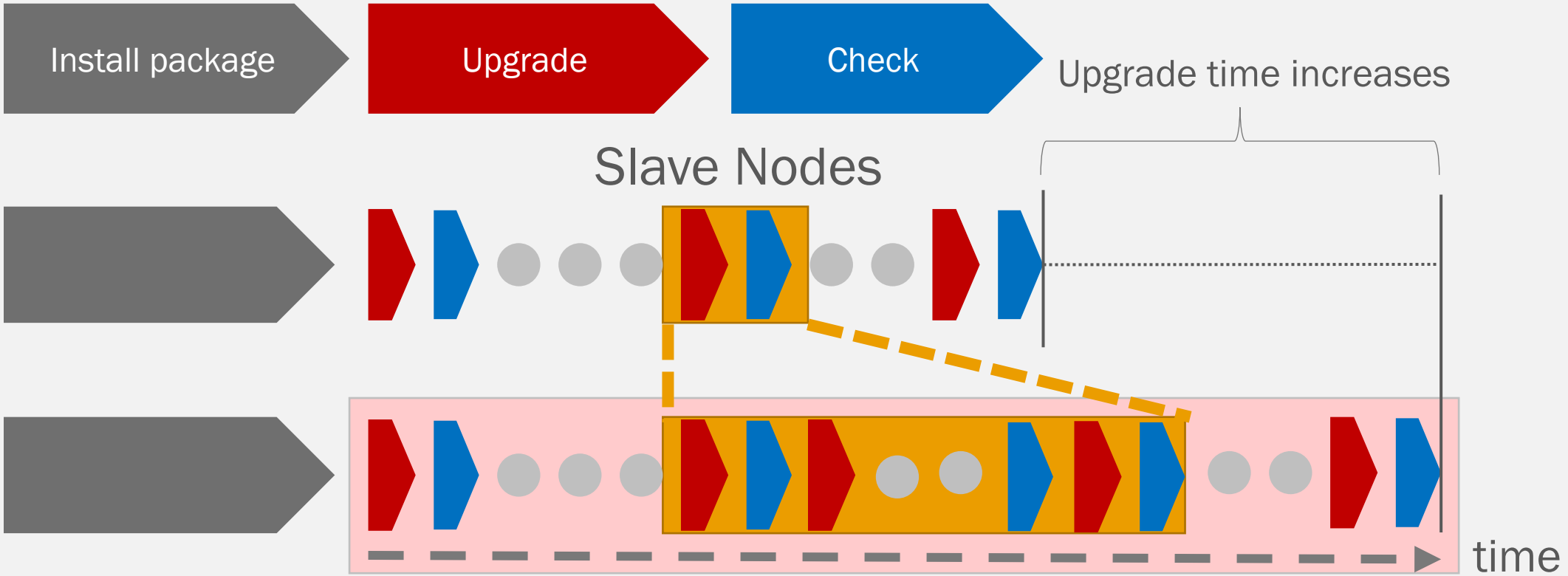
- Impact-less Rolling Upgrade
  - Grouping components e.g. NameNode
  - Upgrading & Checking each component group

Approaches for Upgrading



- Total upgrade time increases as slave nodes are upgraded one-by-one for safety

Approaches for Upgrading



- Automating upgrade and check process to reduce operation cost

Summary

- Impact-less rolling upgrade is possible
- Automating upgrade process reduces operation cost

YAHOO! JAPAN

How to Upgrade

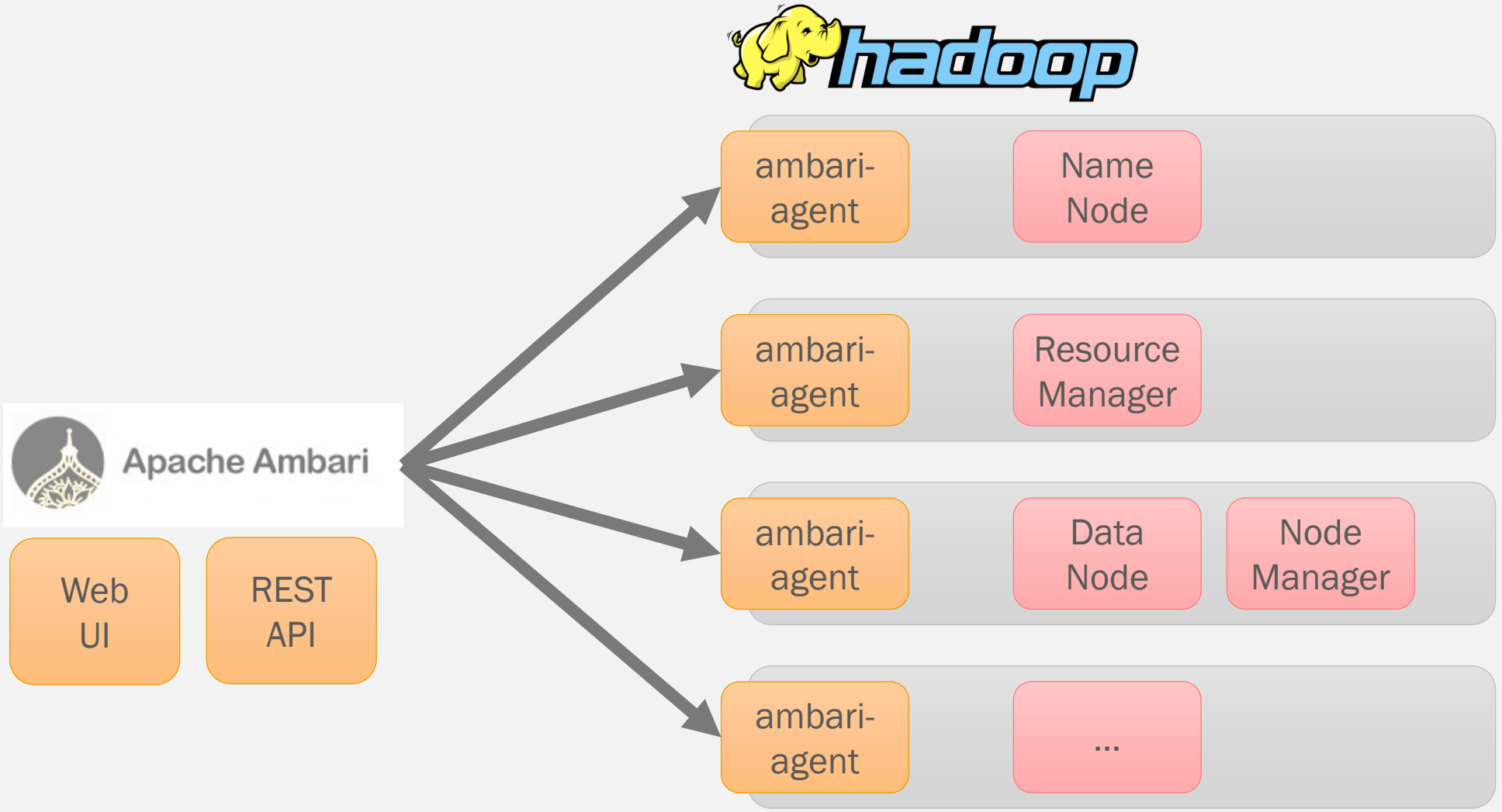
YAHOO! JAPAN

Target Cluster

- HDP 2.3.x + Ambari 2.2.0
- Master nodes
  - HA: NameNode, ResourceManager, JournalNode, Zookeeper
  - Non-HA: Application Timeline Server
- Slave nodes
  - 800 DataNode/NodeManager
- Others
  - HiveServer2, Oozie, HttpFS

YAHOO! JAPAN

Apache Ambari



YAHOO! JAPAN

Ambari Provided HDP Upgrade Methods

- Express Upgrade
  - Brings down entire cluster
- Rolling Upgrade
  - Cannot control
    - Load balanced HiveServer2 restart
    - Collective DataNode restart

But we can’t adopt either of these methods

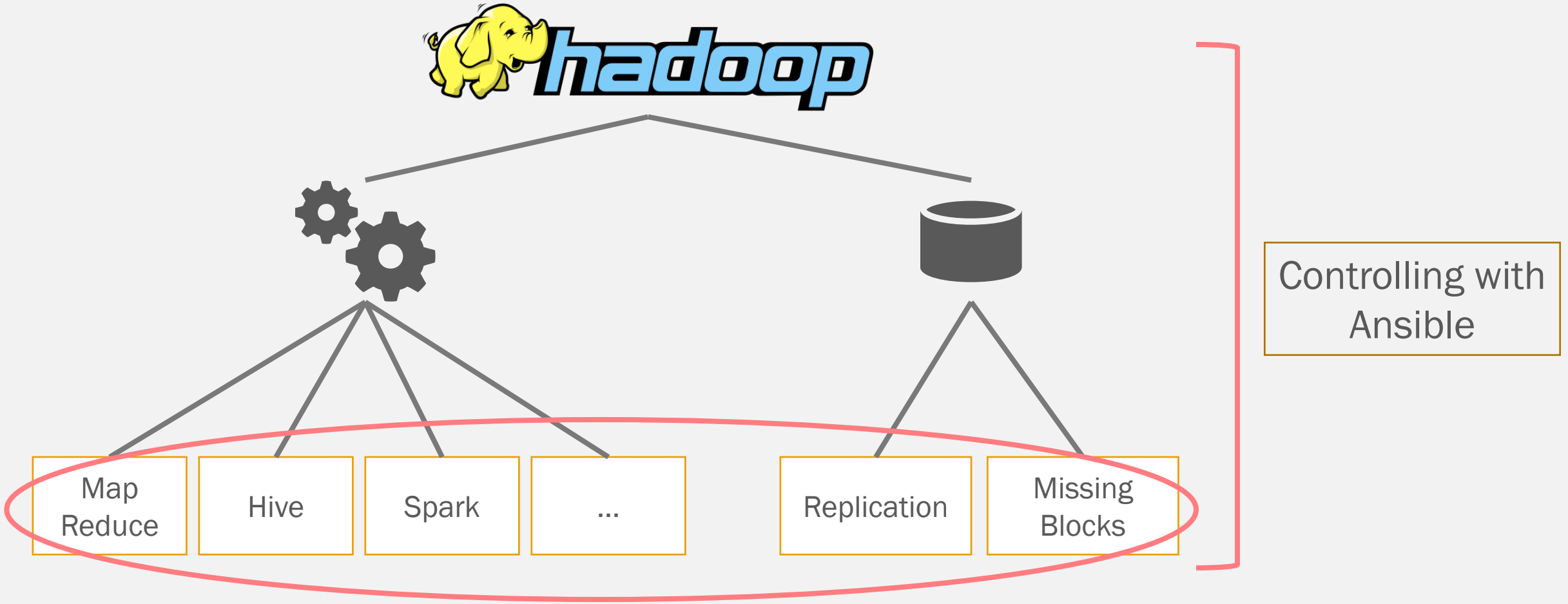
YAHOO! JAPAN

Extending Ambari's Operations

- Safety restart and our environment specific operations
  - Ambari Custom Service
    - Additional operations such as NN failover
  - API wrapper scripts
    - Additional confirmation of service normality
    - Precise control

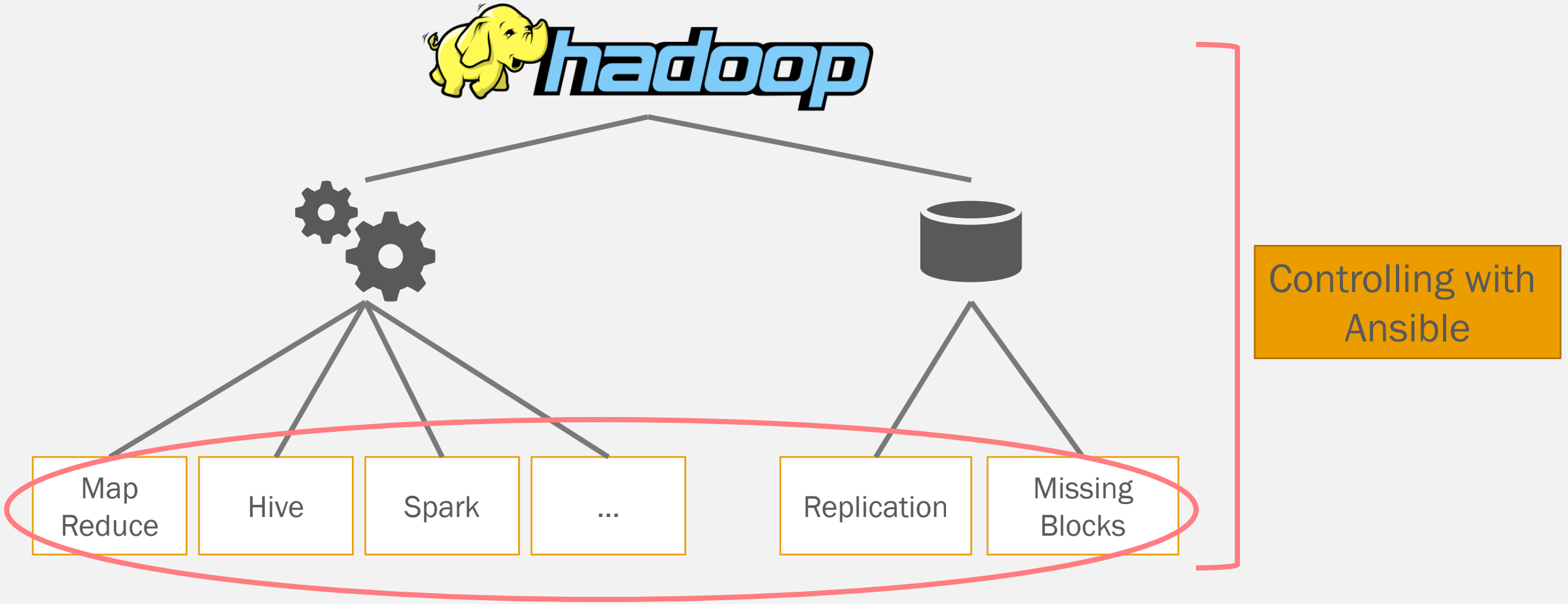
YAHOO! JAPAN

For Non-stop Upgrade



YAHOO! JAPAN

Upgrade Flow Control

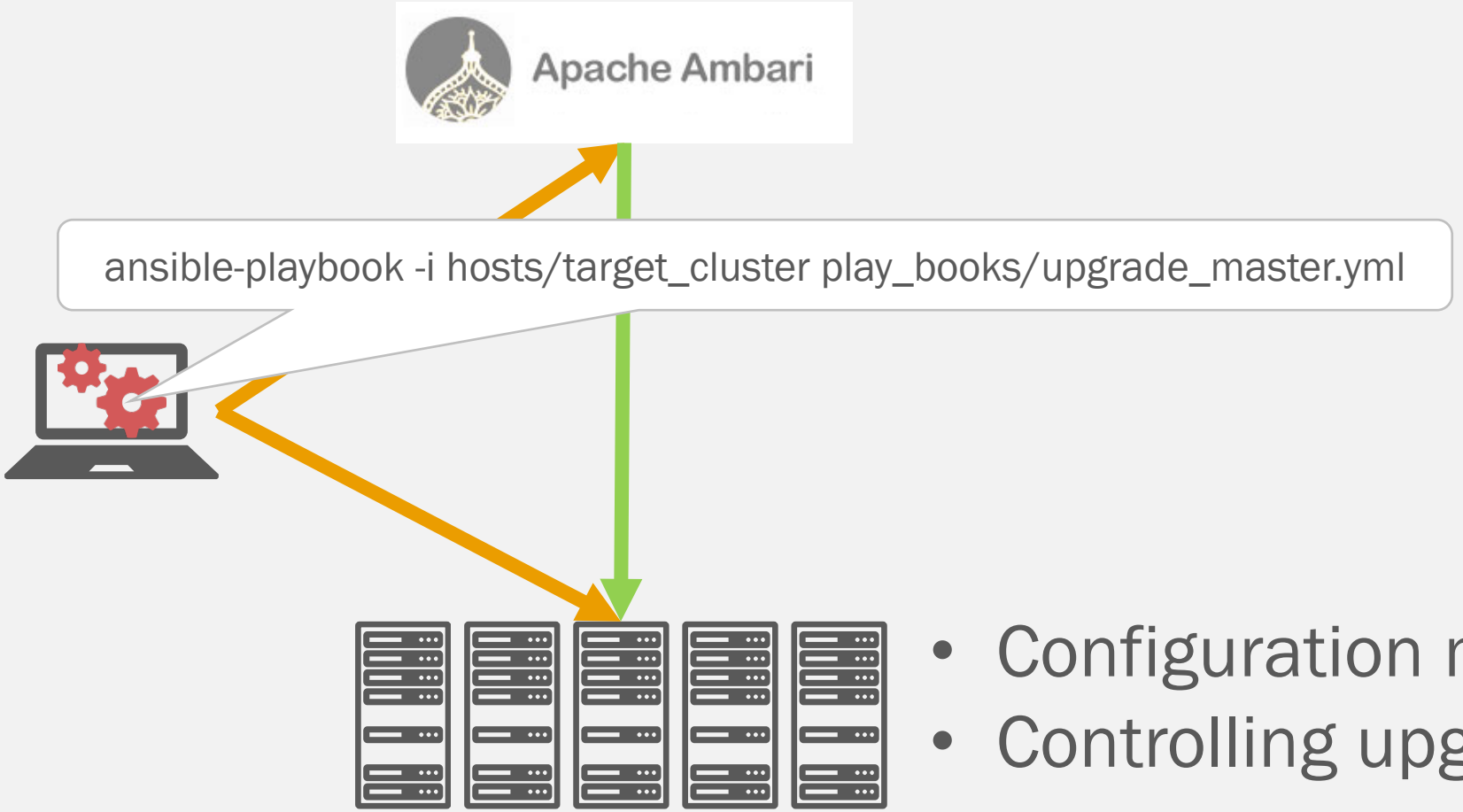


YAHOO! JAPAN

Ansible

- Configuration management tool
- Why we chose?
  - Easy to learn
  - Agent less, push based system
  - Can control upgrading workflow

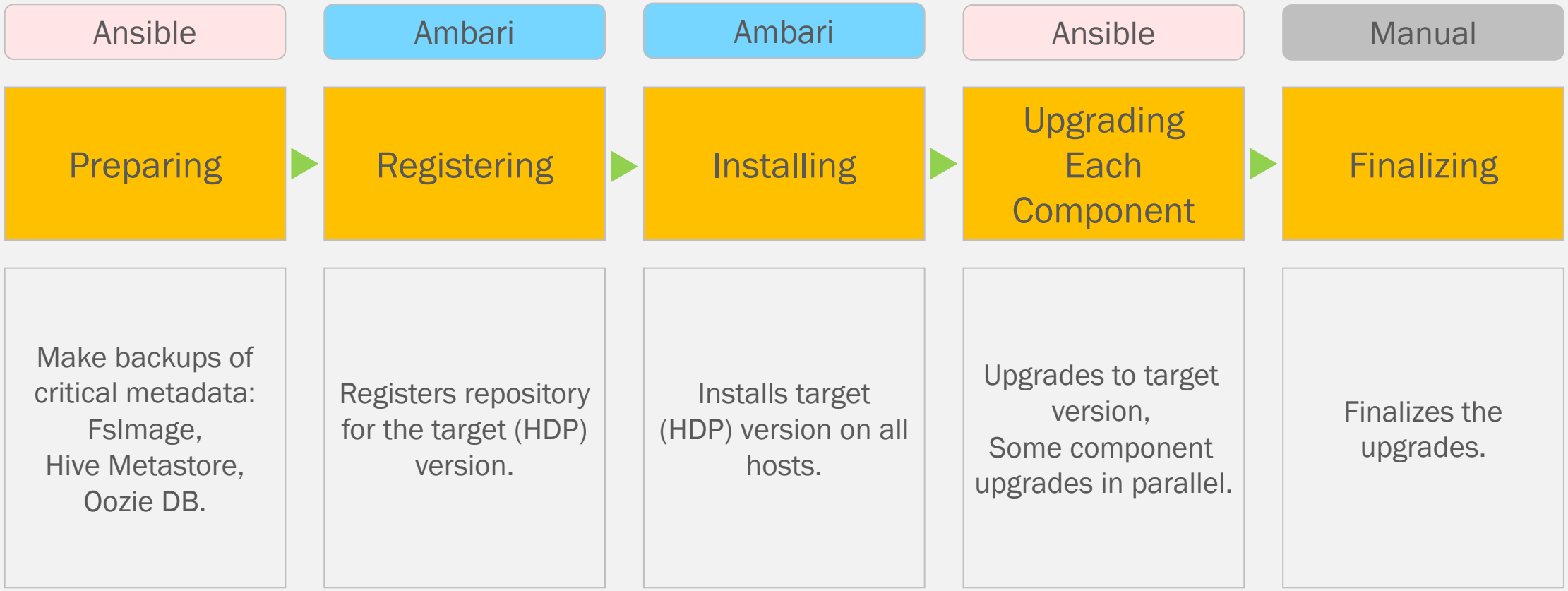
Overview of Upgrading with Ansible



- Configuration management
- Controlling upgrade sequence

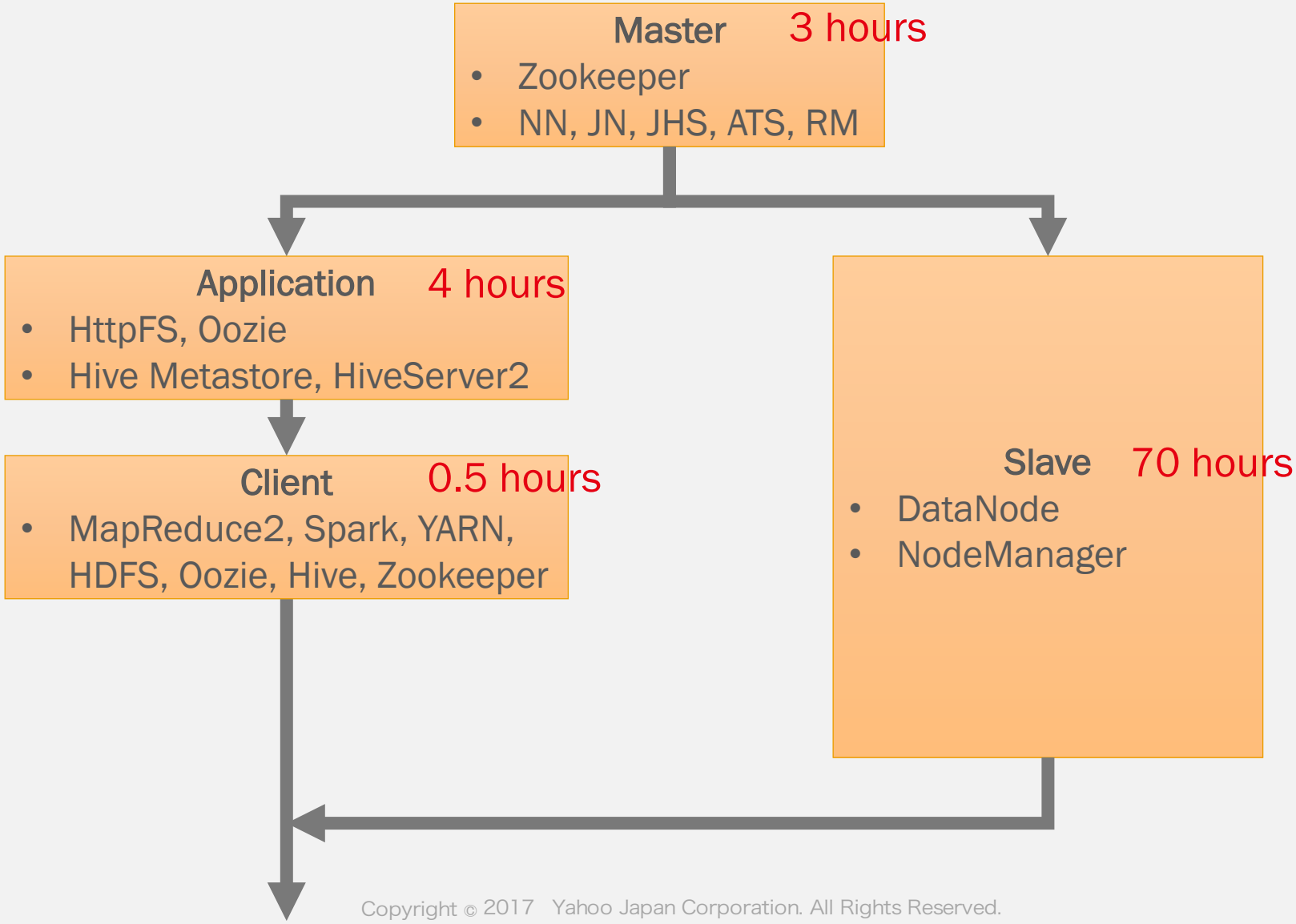
YAHOO! JAPAN

Upgrading Flow

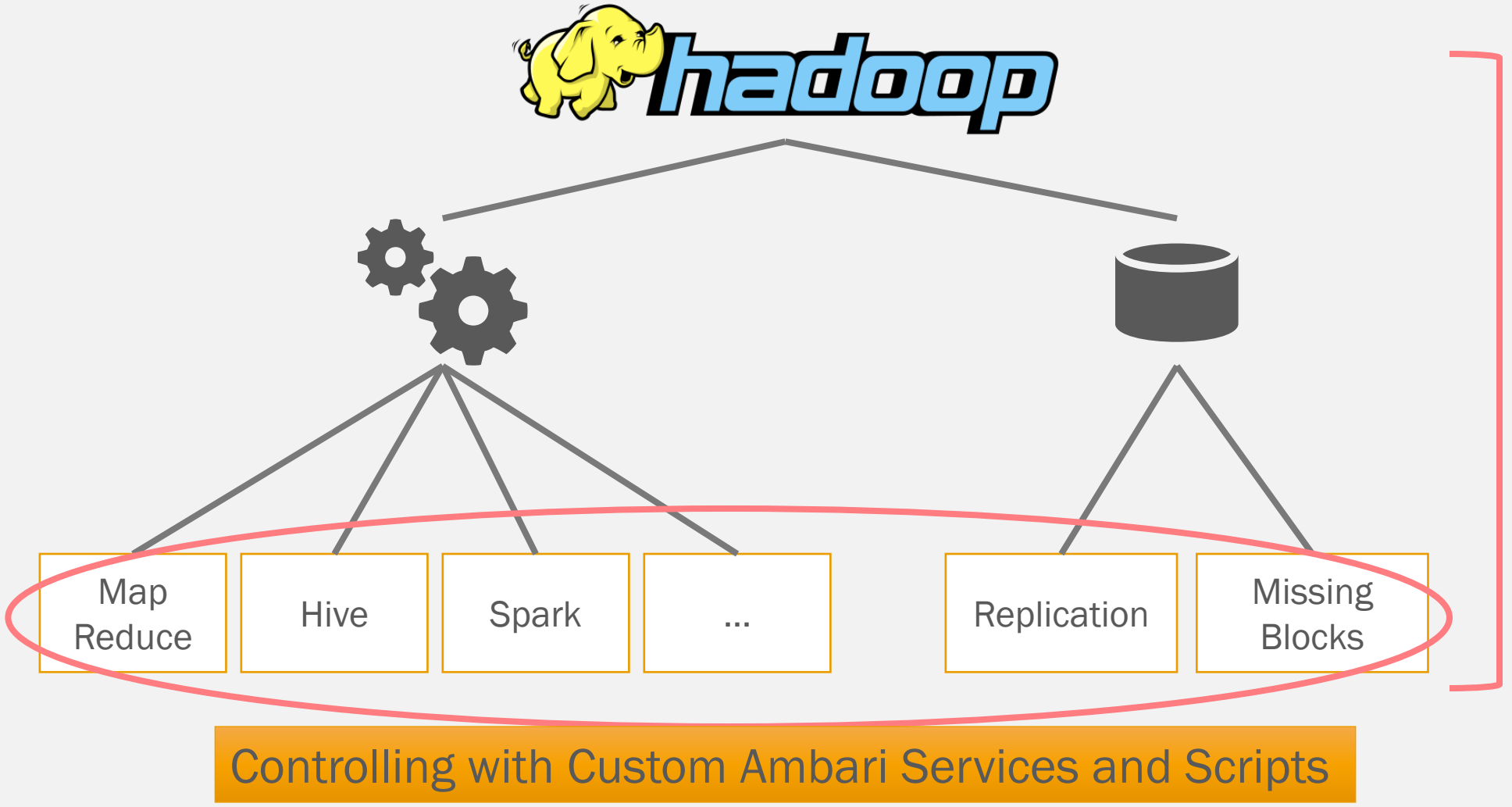


https://docs.hortonworks.com/HDPDocuments/Ambari-2.1.1.0/bk_upgrading_Ambari/content/_manual_minor_upgrade.html

Upgrading Each Component



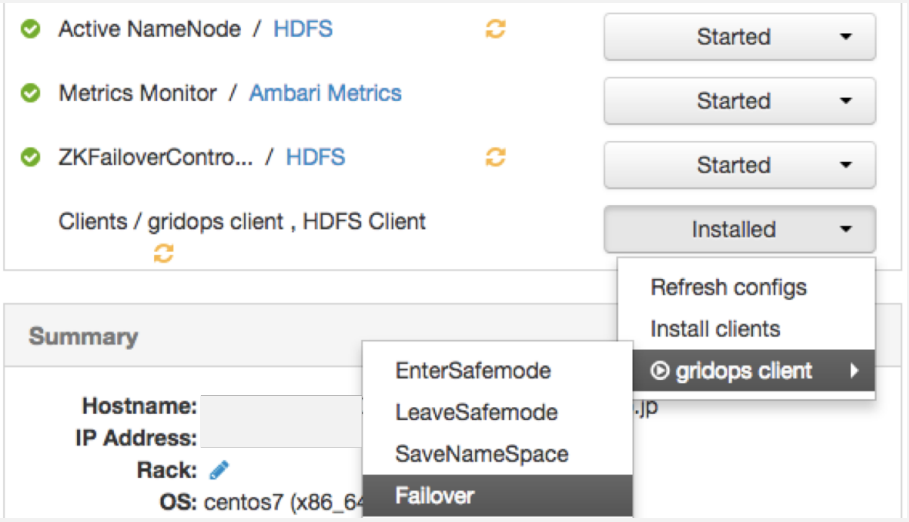
Preventing Job Failures



YAHOO! JAPAN

Ambari Custom Service

- Ambari can implement custom service
- Operational commands
  - NameNode F/O
  - Load balancer pool add/remove
- Can operate as existing Ambari services



Ambari Custom Service

- Add service definition xml and python scripts
- No need of manually executing commands on a server
- Prevents operation miss

```
<customCommand>
    <name>Failover</name>
    <commandScript>
        <script>scripts/gridops_client.py</script>
        <scriptType>PYTHON</scriptType>
        <timeout>600</timeout>
    </commandScript>
</customCommand>
```

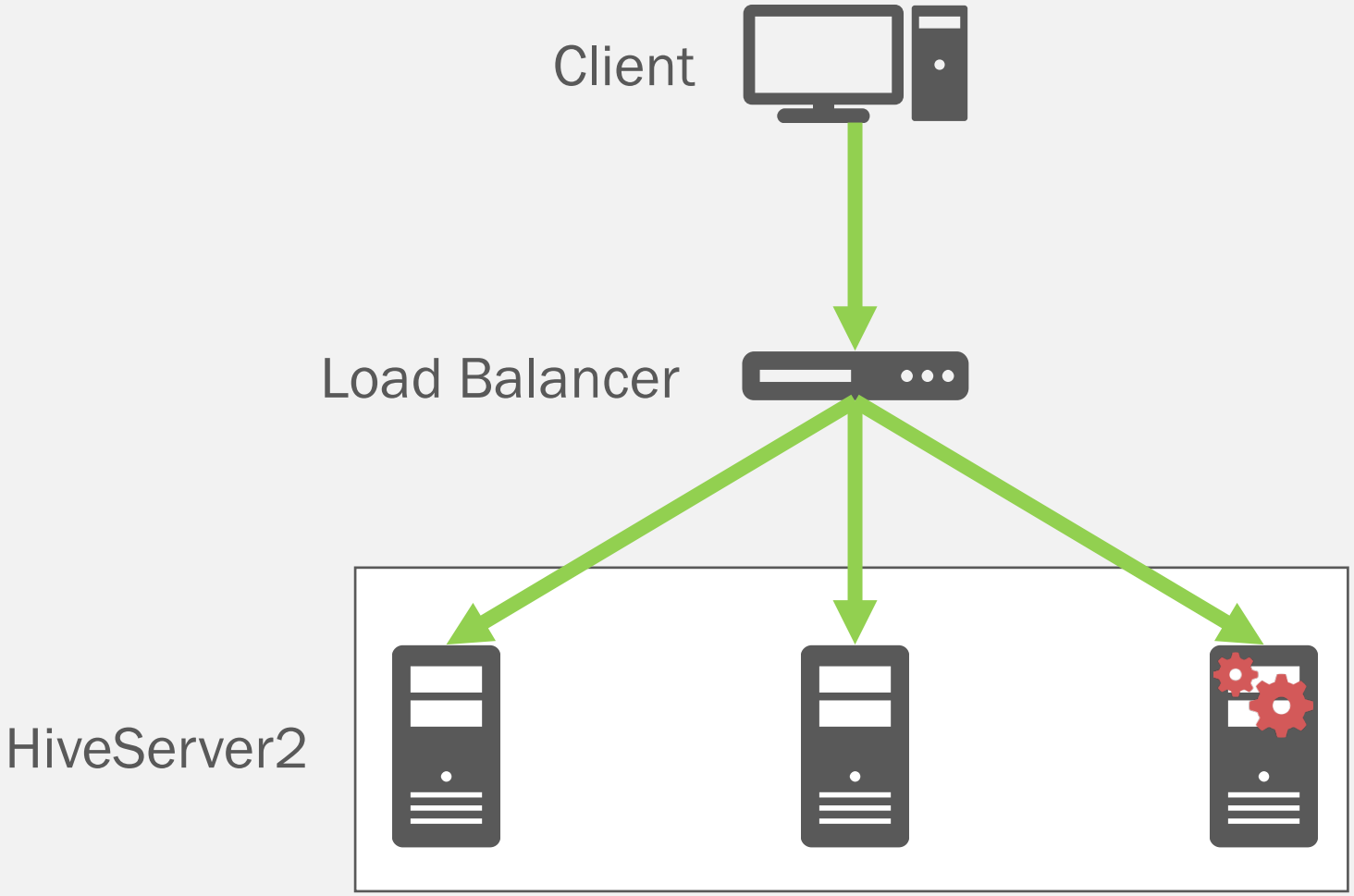
```
def failover(self, env):
  '''
  Tries to failover from active NameNode to standby NameNode
  '''
  self._kinit_superuser(env)
  ha_state = {'nn1': None, 'nn2': None}
  ns = Script.get_config()['configurations']['gridops-config
  _, ha_state['nn1'], stderr = shell.checked_call('hdfs haad
  _, ha_state['nn2'], stderr = shell.checked_call('hdfs haad
  service_id = self._get_service_id(env)
  if service_id not in ha_state:
    failover_from_to = 'nn1 nn2'
  elif ha_state['nn1'] == 'standby' and ha_state['nn2'] == '
    failover_from_to = 'nn2 nn1'
  else:
    raise Fail('Faild to get HA state of NameNodes')
  Execute('hdfs haadmin -ns %s -failover %s' % (ns, failover
```


YAHOO! JAPAN

Ambari CLI

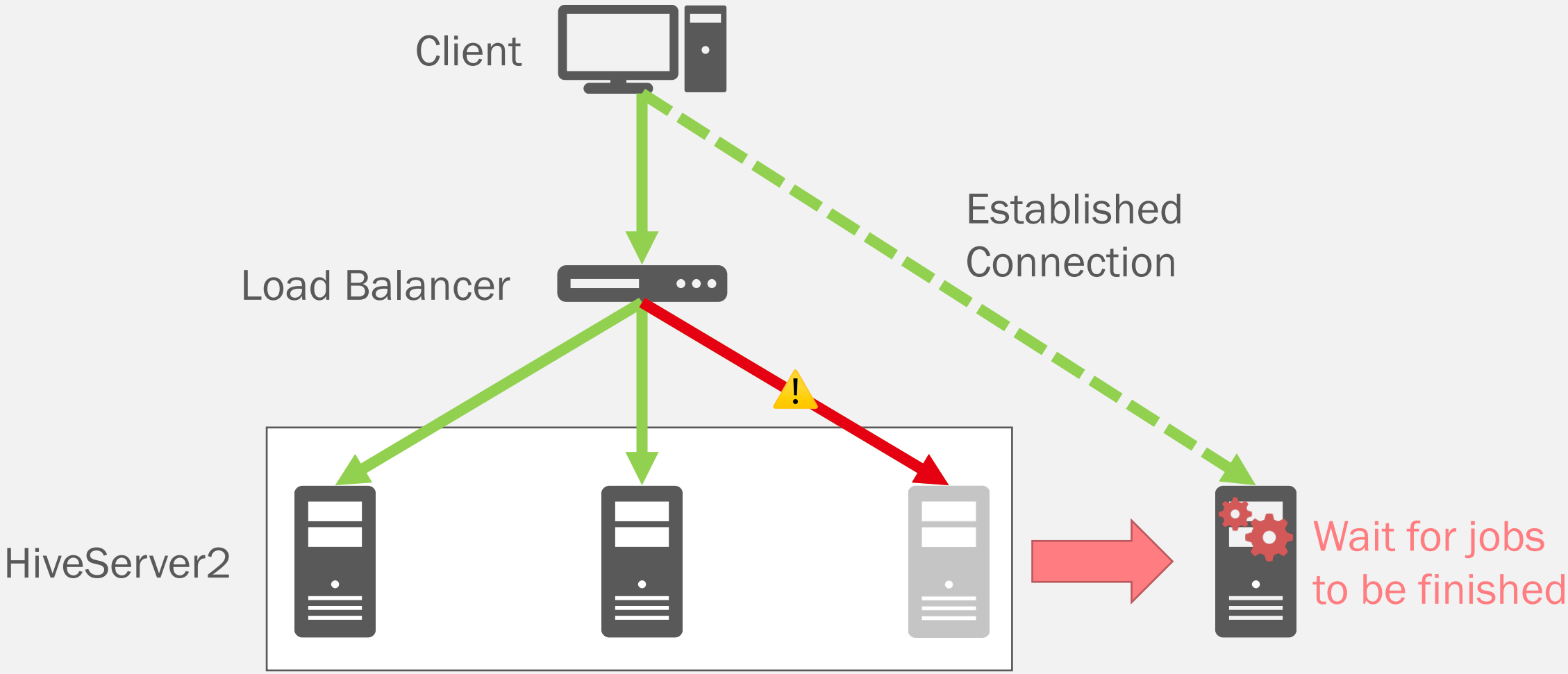
- In-house script for cluster admins
- A wrapper script using API's of Ambari, NameNode, ResourceManager etc.
- Provides safe operations
  - Pre and post restart check for each component
  - Preventing data loss

Safety Restart for HiveServer2

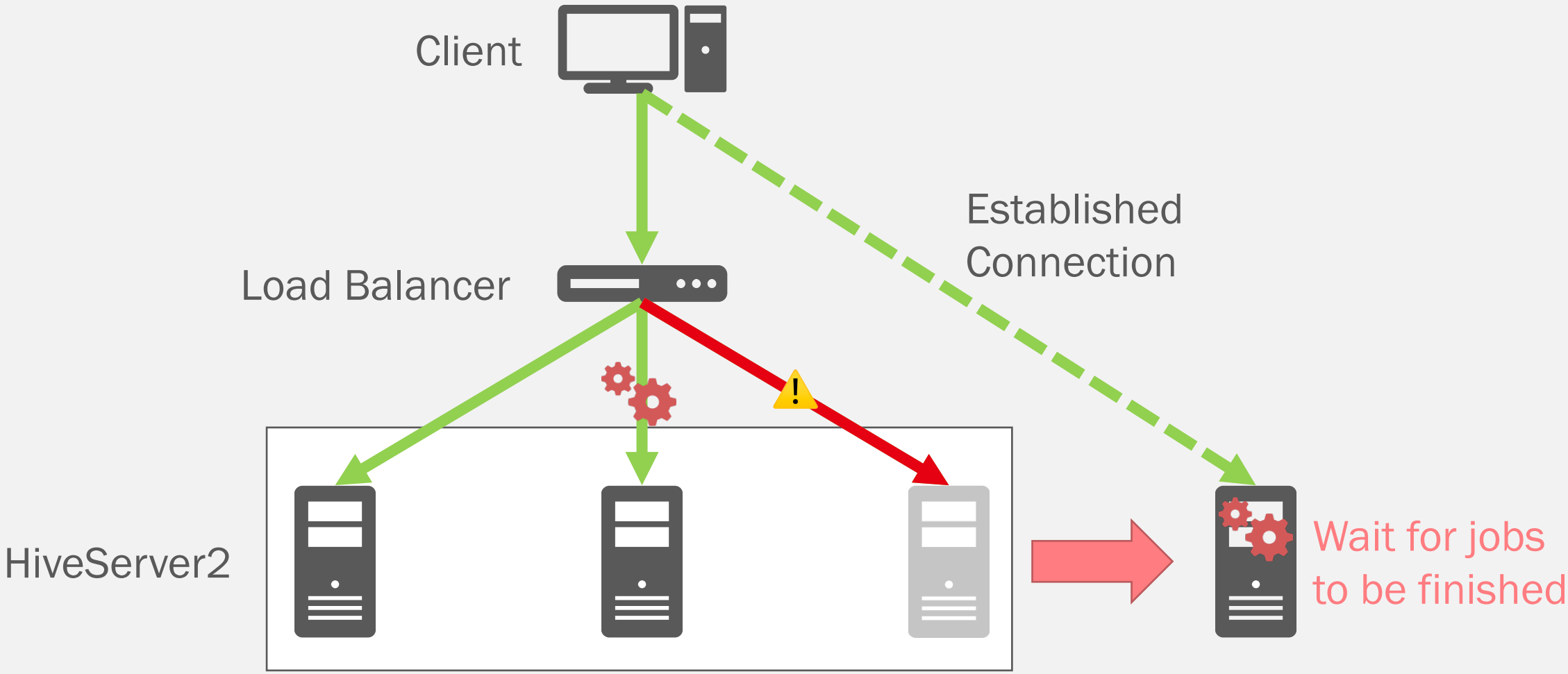


YAHOO! JAPAN

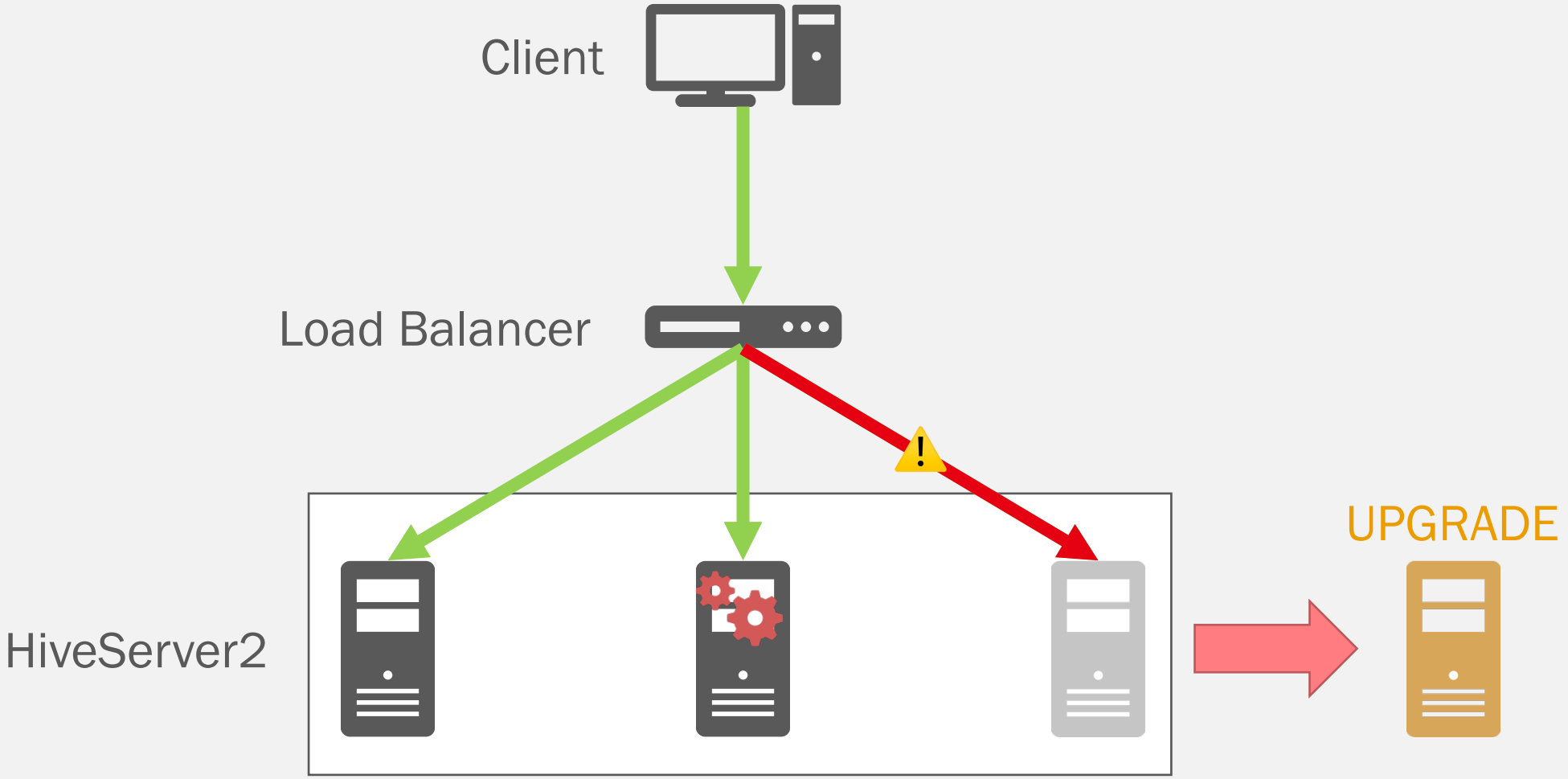
Safety Restart for HiveServer2



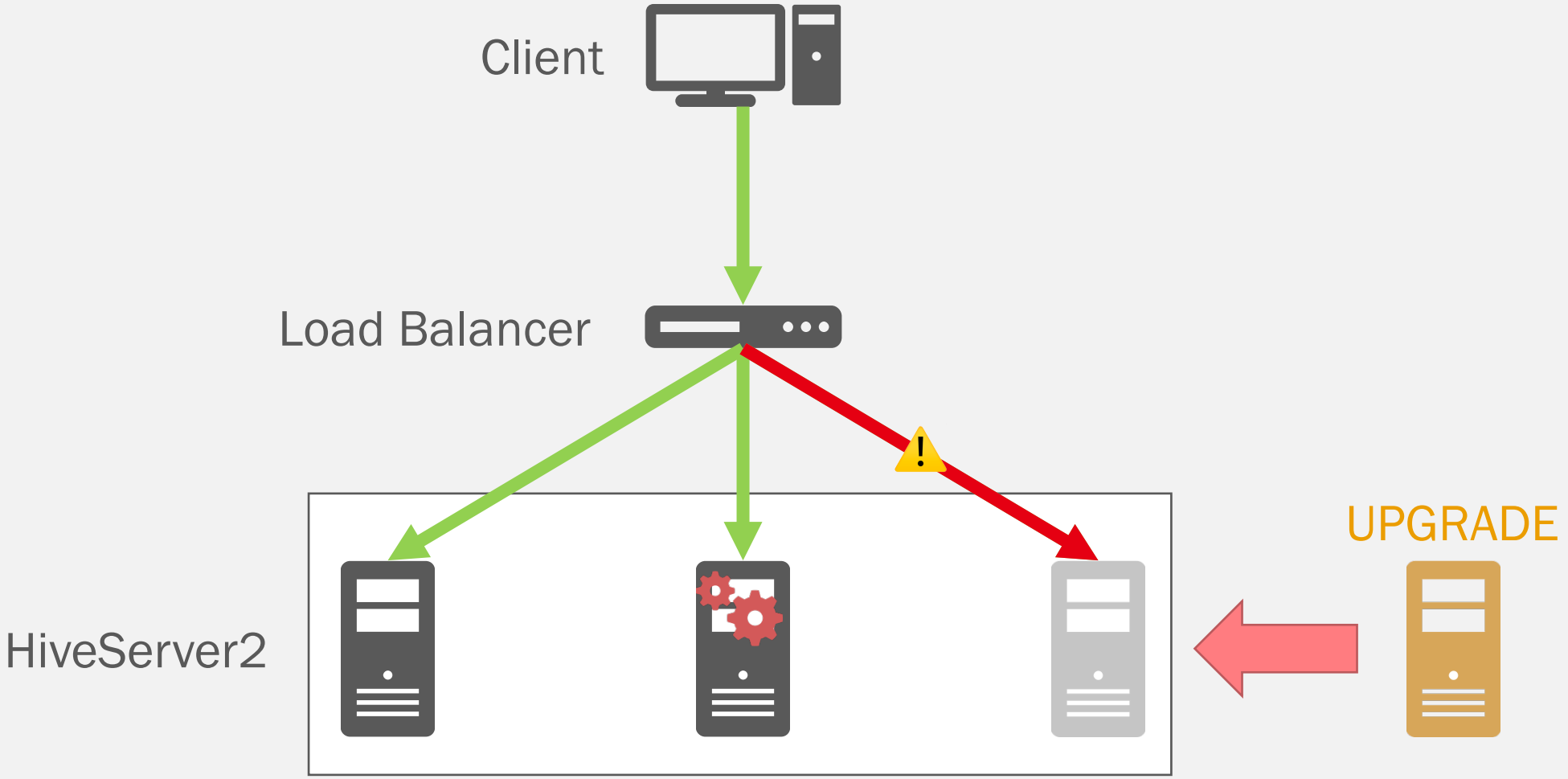
Safety Restart for HiveServer2



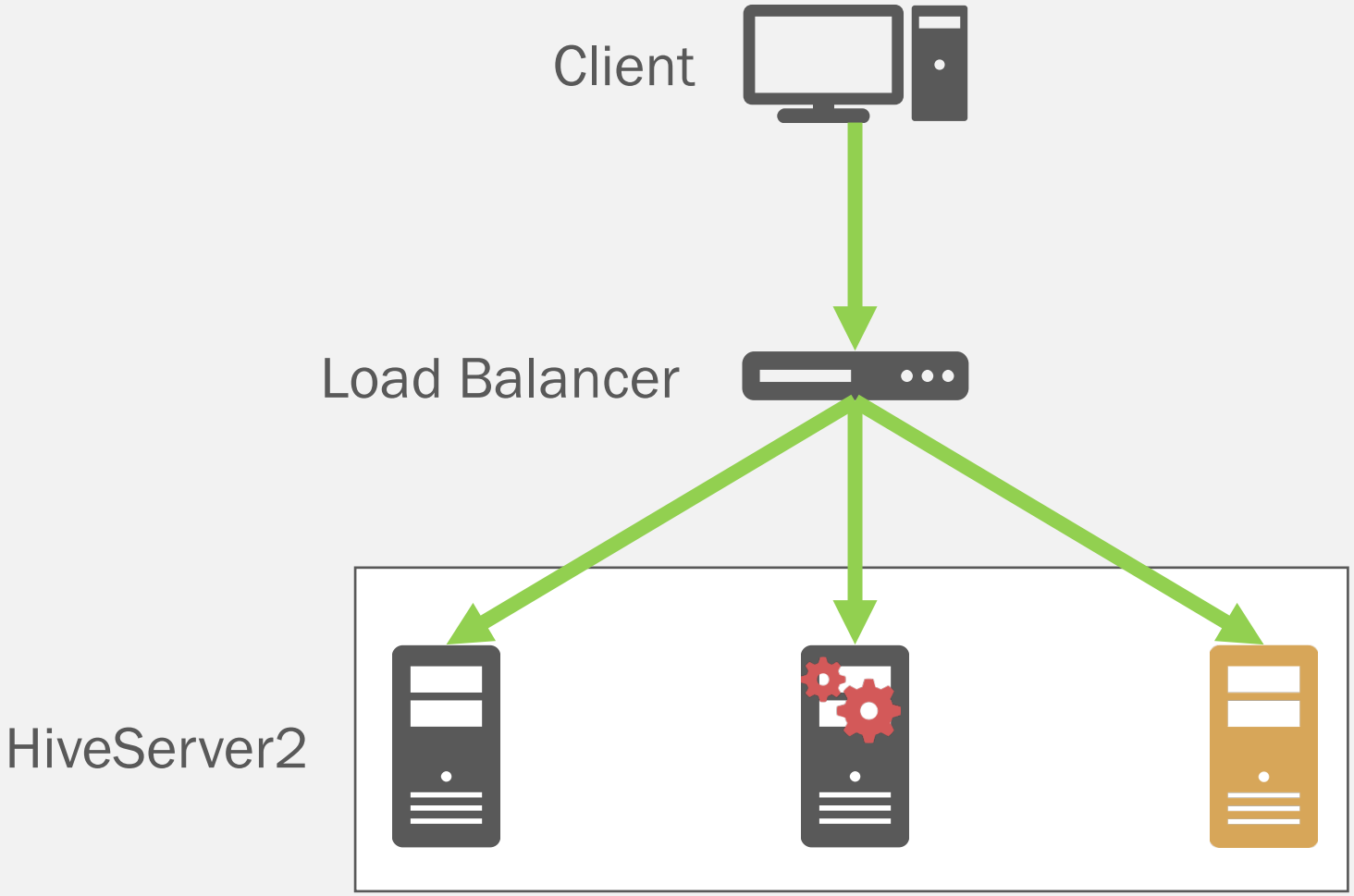
Safety Restart for HiveServer2



Safety Restart for HiveServer2

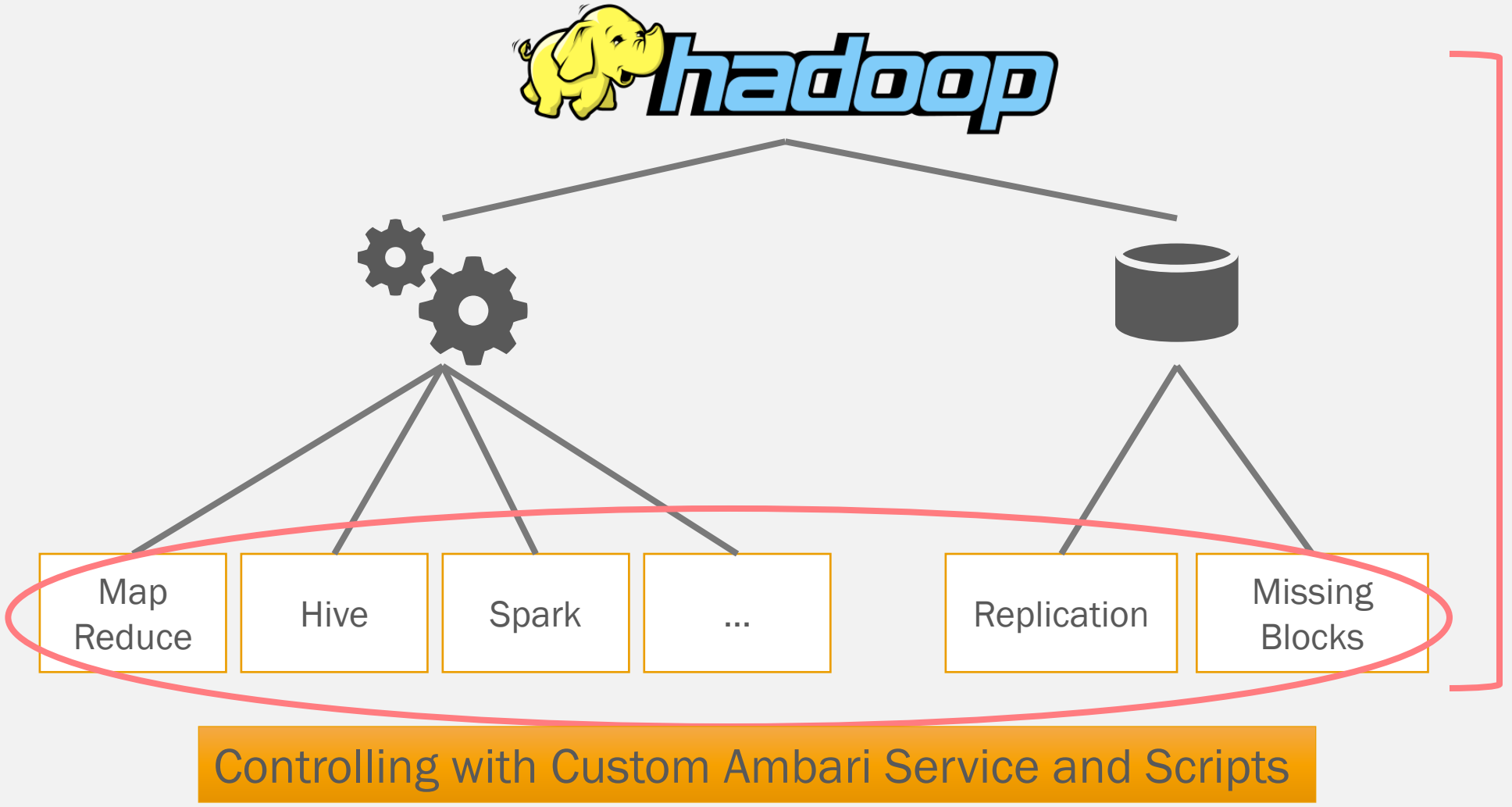


Safety Restart for HiveServer2



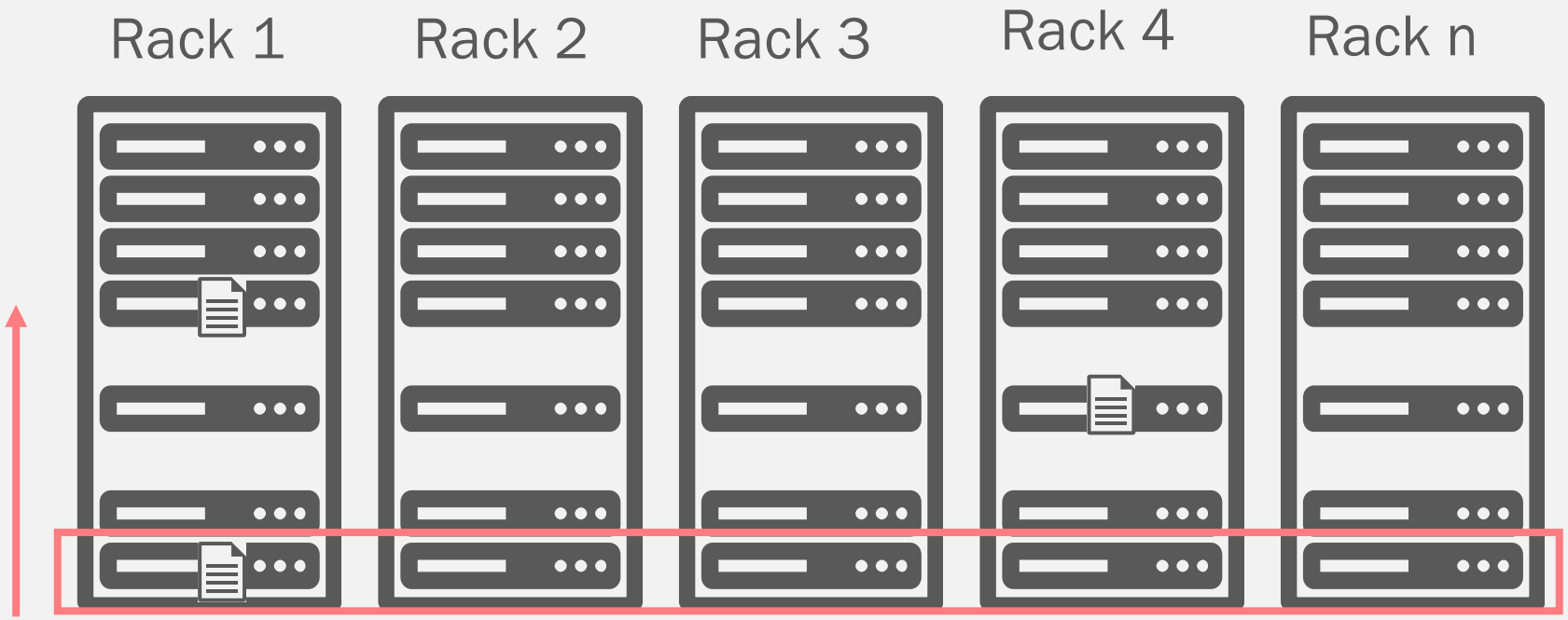
YAHOO! JAPAN

Preventing Data Loss



YAHOO! JAPAN

Safety Restart for DataNode



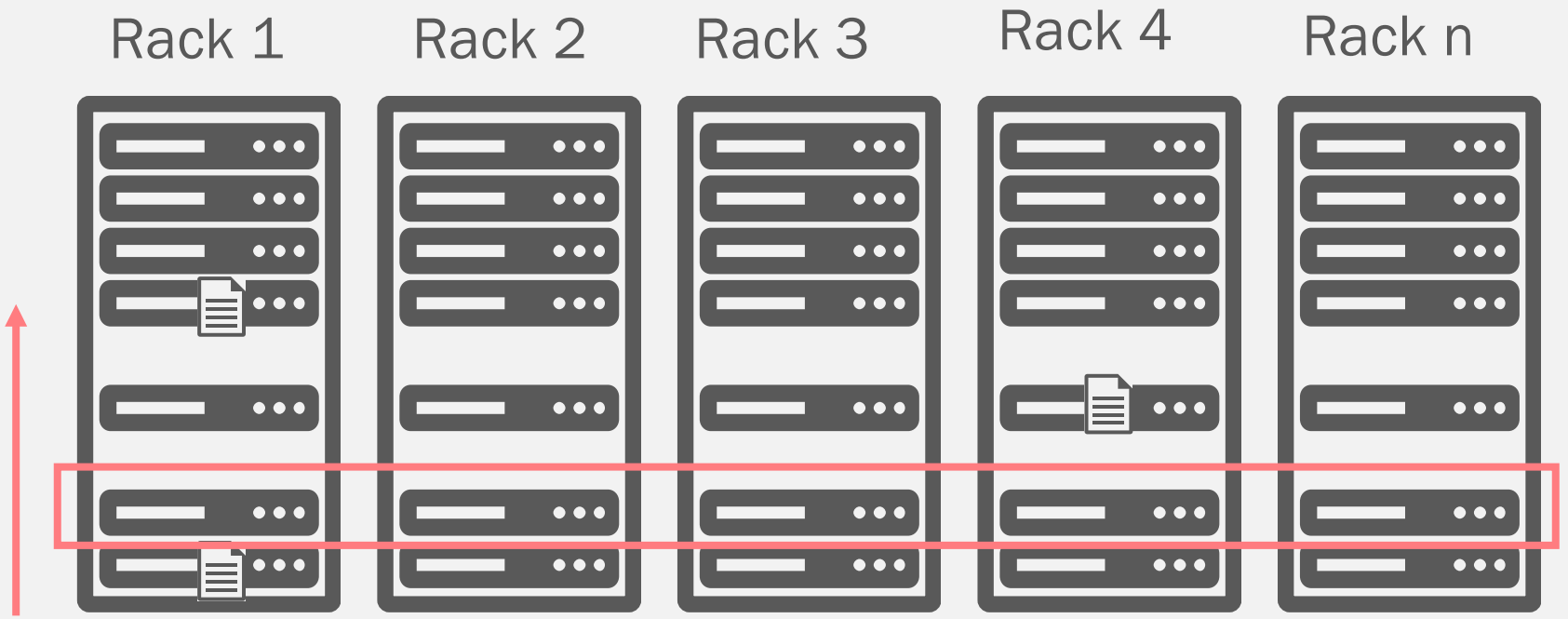
Missing Blocks: 0
Under Replicated Blocks: 0
Corrupt Blocks: 0

Group of DataNodes
at a time

Upgrade, restart and wait for Missing Blocks to be 0

YAHOO! JAPAN

Safety Restart for DataNode



Missing Blocks: 0
Under Replicated Blocks: 0
Corrupt Blocks: 0

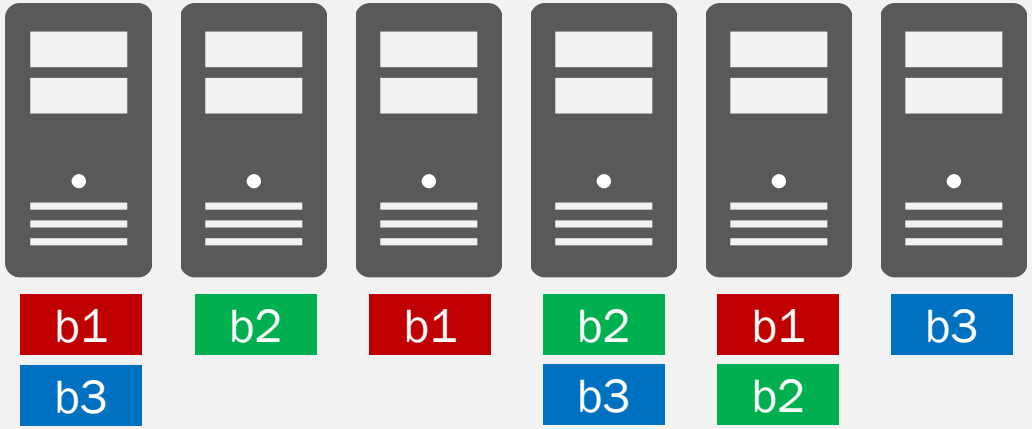
Group of DataNodes
at a time

Upgrade, restart and wait for Missing Blocks to be 0

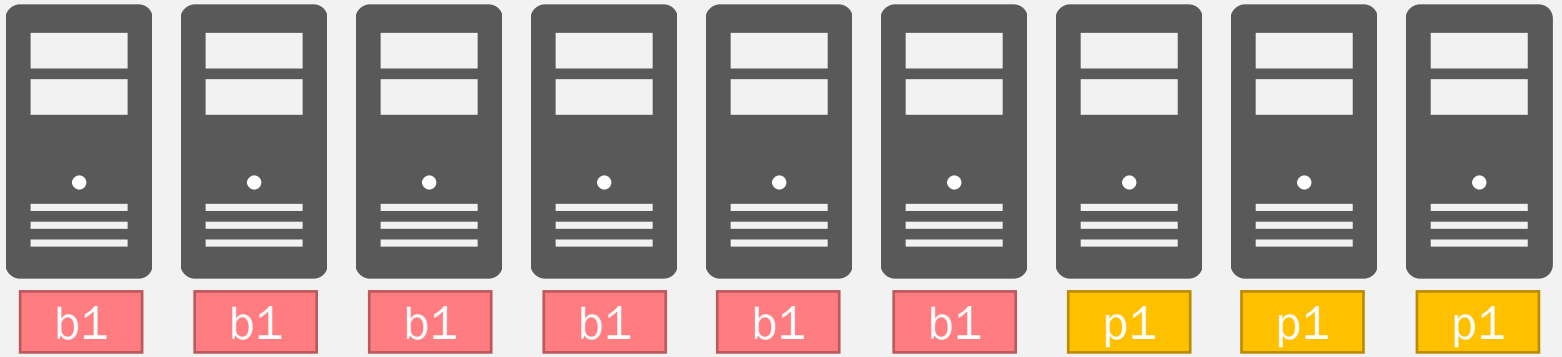
YAHOO! JAPAN

Replication Vs Erasure Coding

Replication

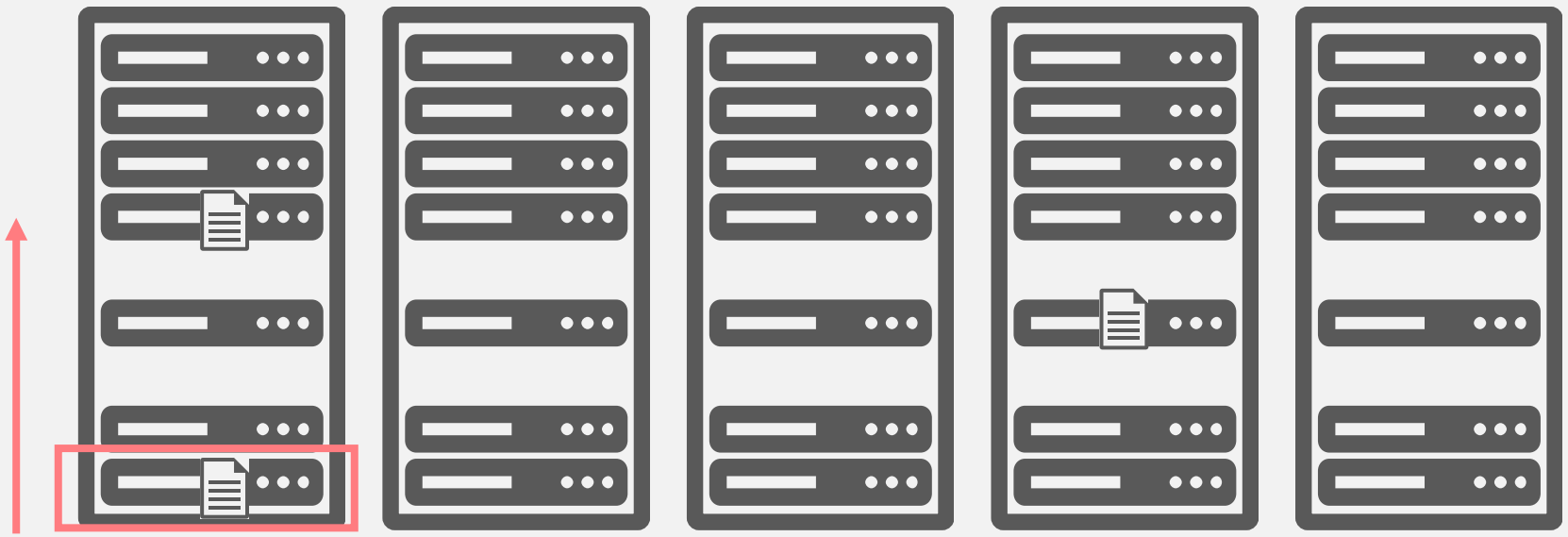


Erasure Coding (EC)



YAHOO! JAPAN

Safety Restart for DataNode (EC)



Missing Blocks: 0
Under Replicated Blocks: 0
Corrupt Blocks: 0

One DataNode at a time

Upgrade, restart and wait for Missing Blocks to be 0

YAHOO! JAPAN

Test Jobs for Each Component

- HDFS Read/Write
- MapReduce
- Hive, Hive on Tez
- Pig, Spark
- HttpFS, Oozie

```
hdfs dfs -put /tmp/test
```

```
yarn jar hadoop-mapreduce-examples.jar pi 5 10
```

```
hive -e “select x from default.dual group by x”
```

```
curl --negotiate -u : “https://.../webhdfs/v1/?op=liststatus”
```


YAHOO! JAPAN

Results

Results of Non-stop Upgrade

- Hadoop 2.3.x to 2.3.y upgrade
- 7 minutes of downtime
- 0.3% of job failure
- 0% data loss

YAHOO! JAPAN

Results of Non-stop Upgrade

Component	Non-stop	Description
HDFS	✅	
HttpFS	✅	
MapReduce	✅	
Hive	✅	
Pig	✅	
Spark	△	Disconnected from Hive Metastore, few jobs failed
HiveServer2	△	Checksum error occurred for some jobs
Oozie	△	Non-HA component

✅ successful upgrade without any job failures
△ Upgrade was affected up to some extent

YAHOO! JAPAN

Problems During the Upgrade

- NameNode restart
- DataNode restart
- Spark and Hive job failure

YAHOO! JAPAN

NameNode Restart

- NN metadata was huge and it took long to free memory
- As a workaround, stop NN → wait → start NN

DataNode Restart

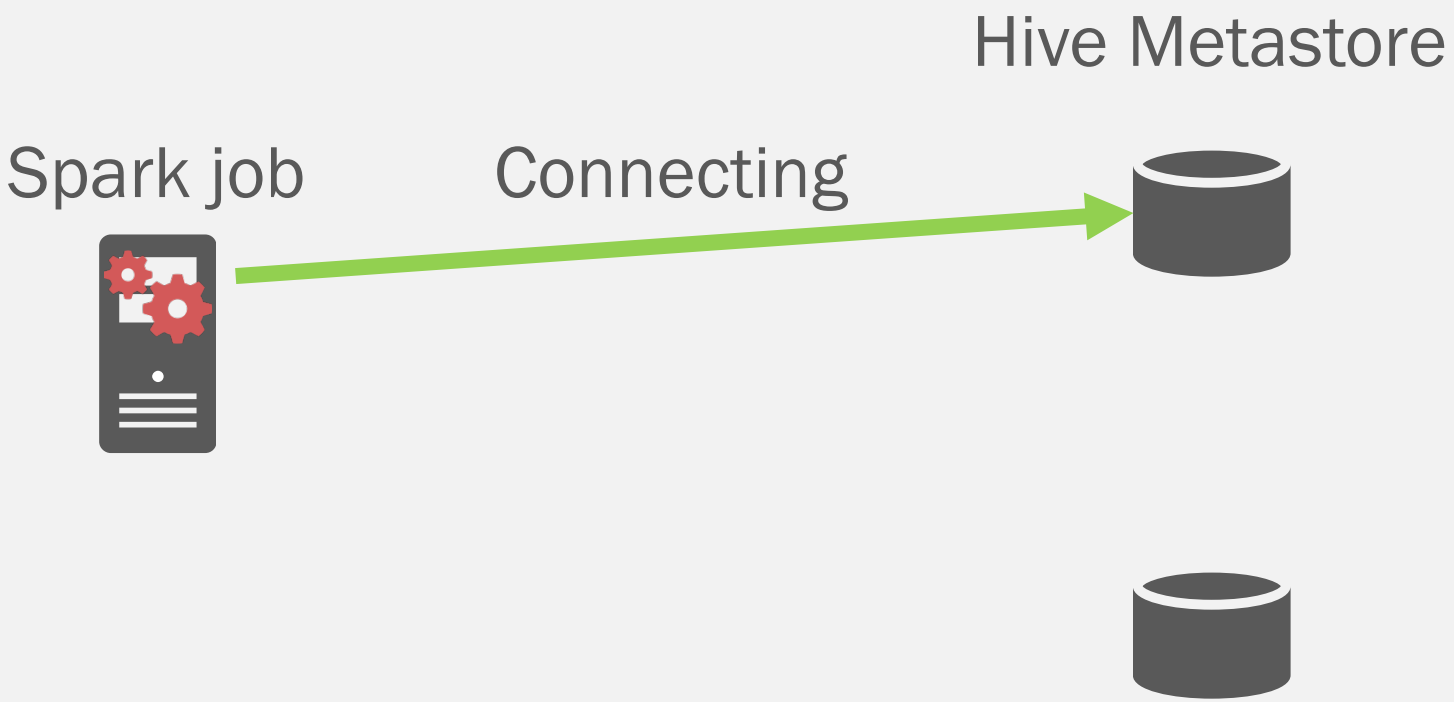
- Restart failed due to existence of old pid file
- 2 DataNode Processes
  - Parent: /var/run/hdfs/hadoop-hdfs-datanode.pid
  - Child: /var/run/hdfs/hadoop_secure_dn.pid

Usually gets deleted automatically, but

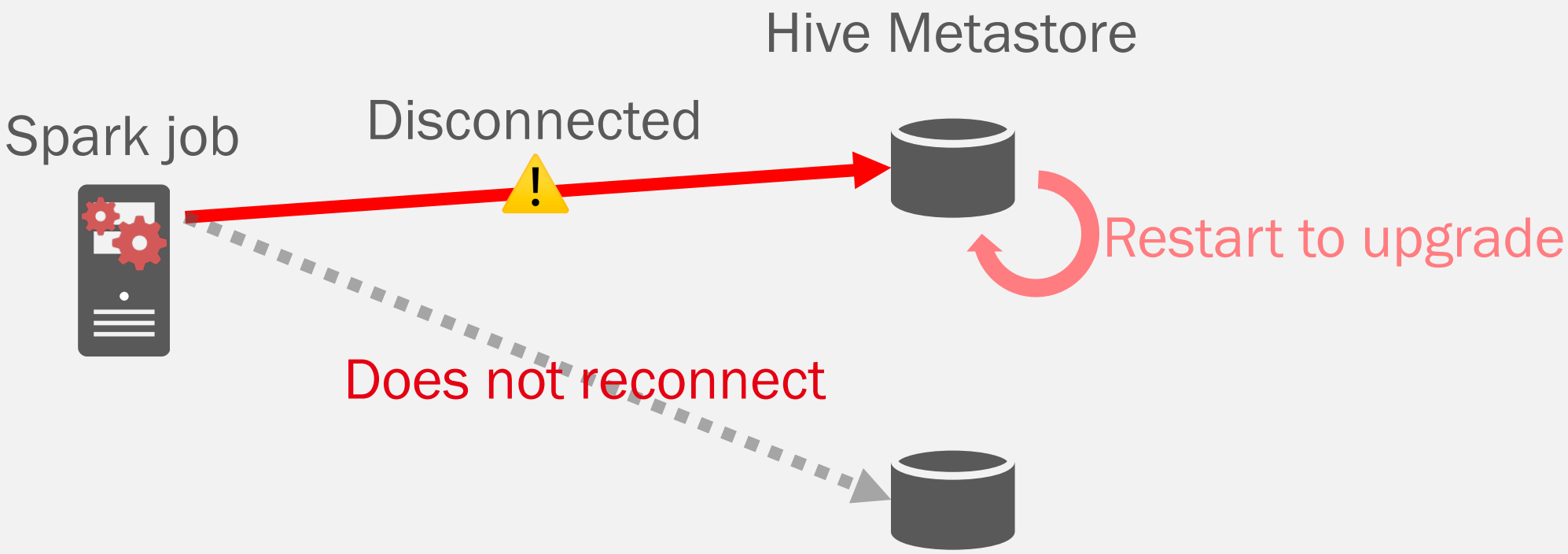
Starting regular datanode initialization
Still running according to PID file /var/run/hdfs/hadoop_secure_dn.pid

YAHOO! JAPAN

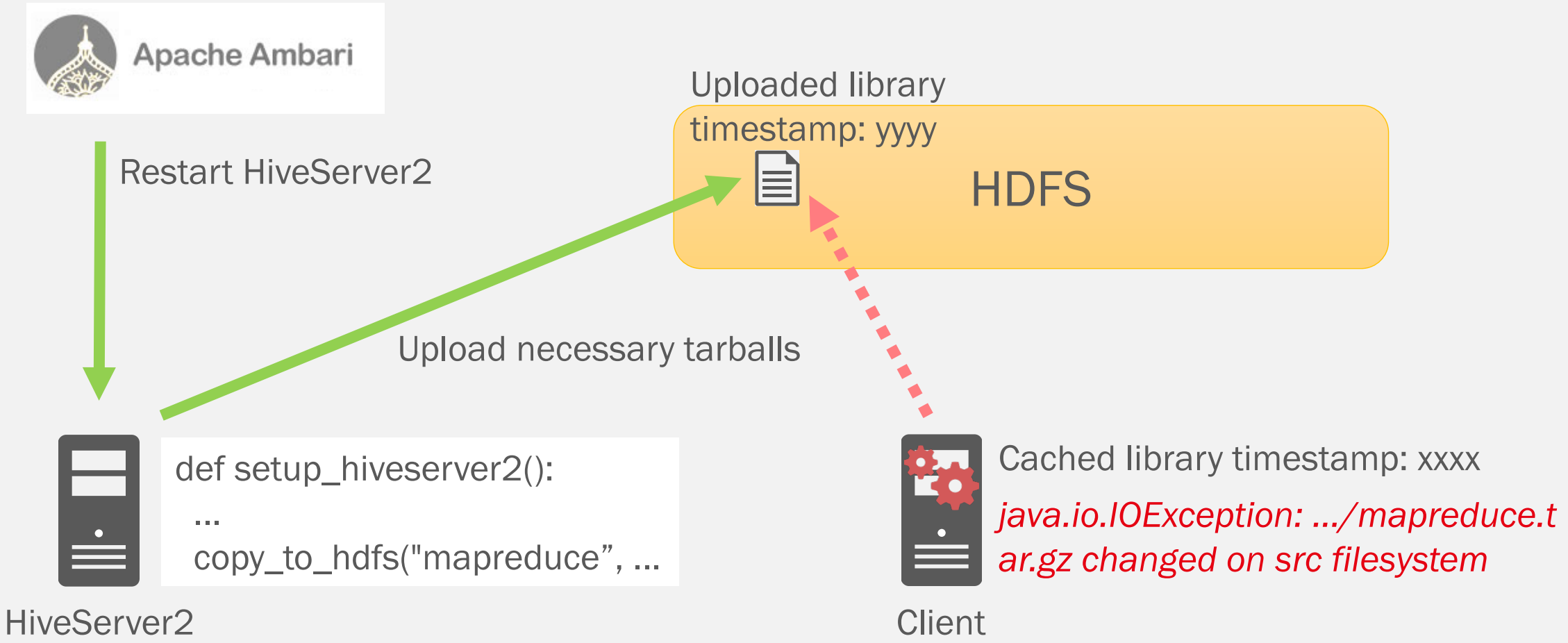
Spark (v1.5) Job Failure



Spark (v1.5) Job Failure



Hive Job Failure



Conclusion

Non-stop Upgrade

- Hadoop 2.3.x to 2.3.y upgrade
- 7 minutes of downtime
- 0.3% of job failure
- 0% data loss
- No need of separate resource scheduling

YAHOO! JAPAN

Comparison of Upgrade Results

	Previous	Proposed (without automation)	Proposed (with automation)	
Cluster Downtime	12h	0min	7min	User impact 
Maintenance Time	12h	72h	72h	—
Man-hour	108h	648h	21h	Operating cost 

YAHOO! JAPAN

Future Work

- Improving non-stop upgrade by bringing down
  - Cluster downtime to 0
  - Job failures to 0
- To be able to perform major upgrades (upgrades involving NameNode metadata changes)
- Automating of preparation stage and handle failures with recovery operations
- Contributing to OSS

YAHOO! JAPAN