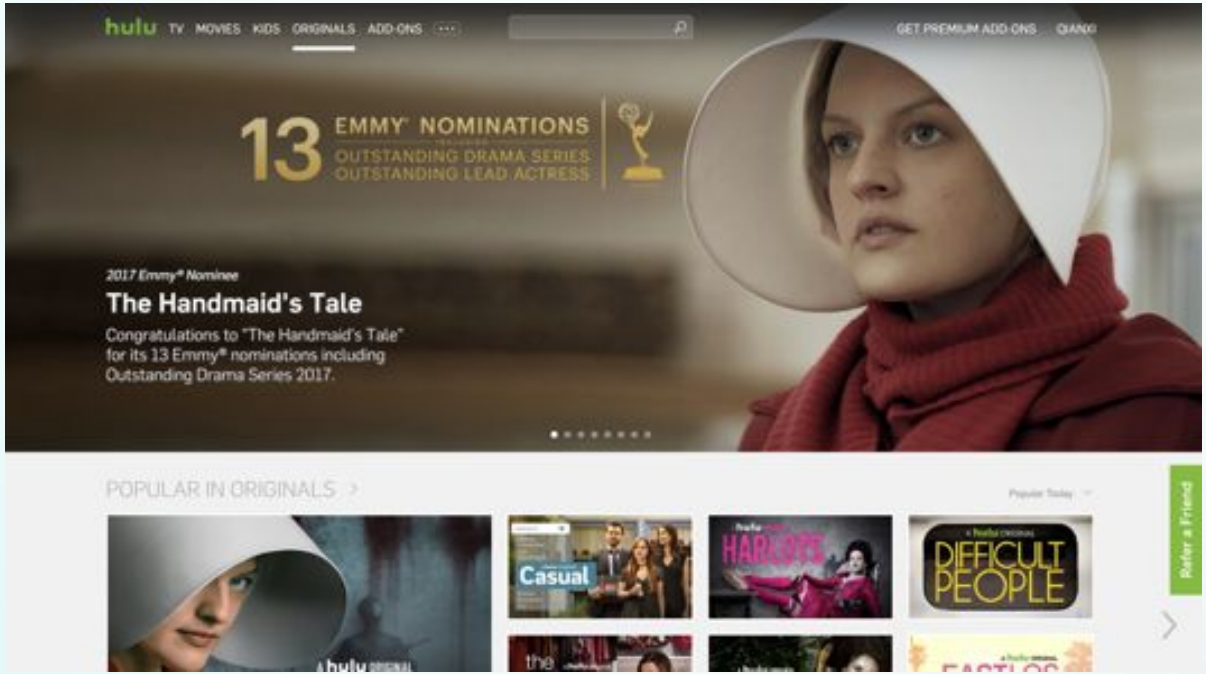
hulu

PB级Hadoop集群跨机房迁移实战

董西成 @ Hulu

About Hulu


hulu
TV
MOVIES
KIDS
ORIGINALS
ADD-ONS
GET PREMIUM ADD-ONS
13 EMMY® NOMINATIONS
OUTSTANDING DRAMA SERIES
OUTSTANDING LEAD ACTRESS
2017 Emmy® Nominee
The Handmaid's Tale
Congratulations to "The Handmaid's Tale" for its 13 Emmy® nominations including Outstanding Drama Series 2017.
POPULAR IN ORIGINALS
Casual
HARLOTS
DIFFICULT PEOPLE
Refer a Friend

hulu

Agenda

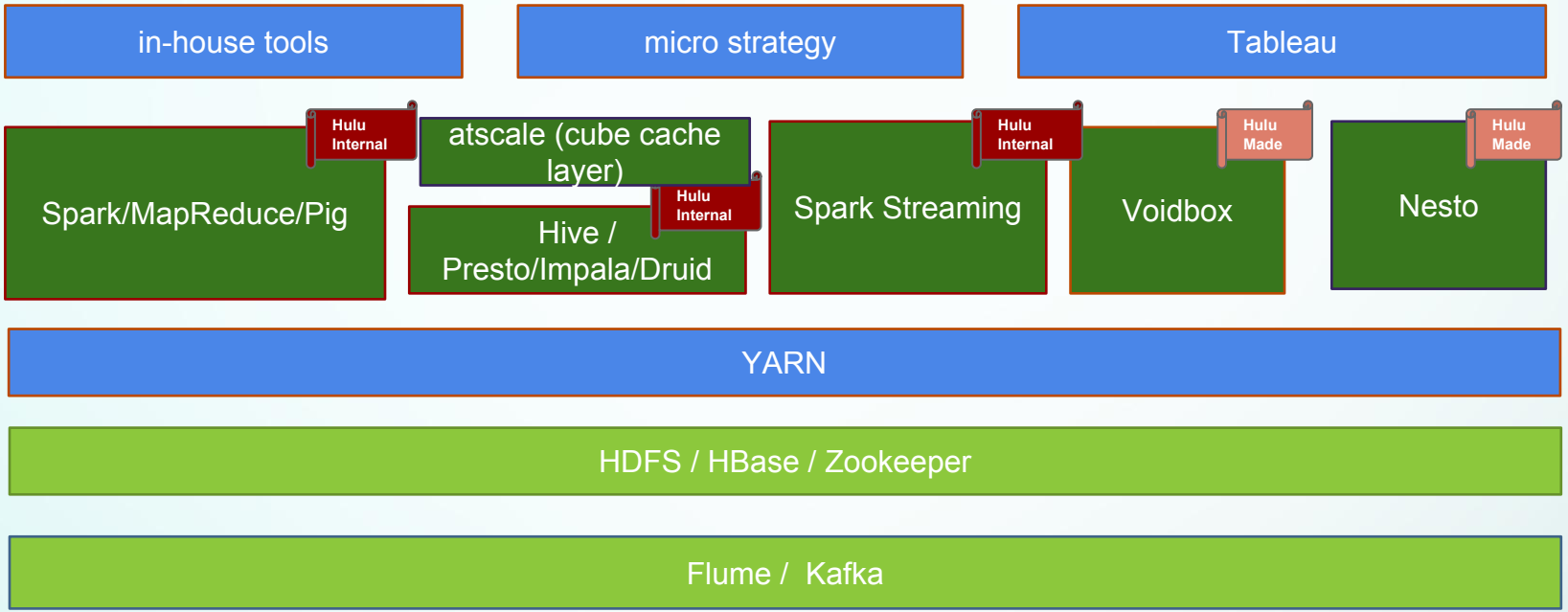
- Background
- Multi-DC HDFS
- Experience
- Summary

hulu

Background

basic background of hadoop migration

Hulu Big Data Infrastructure



hulu

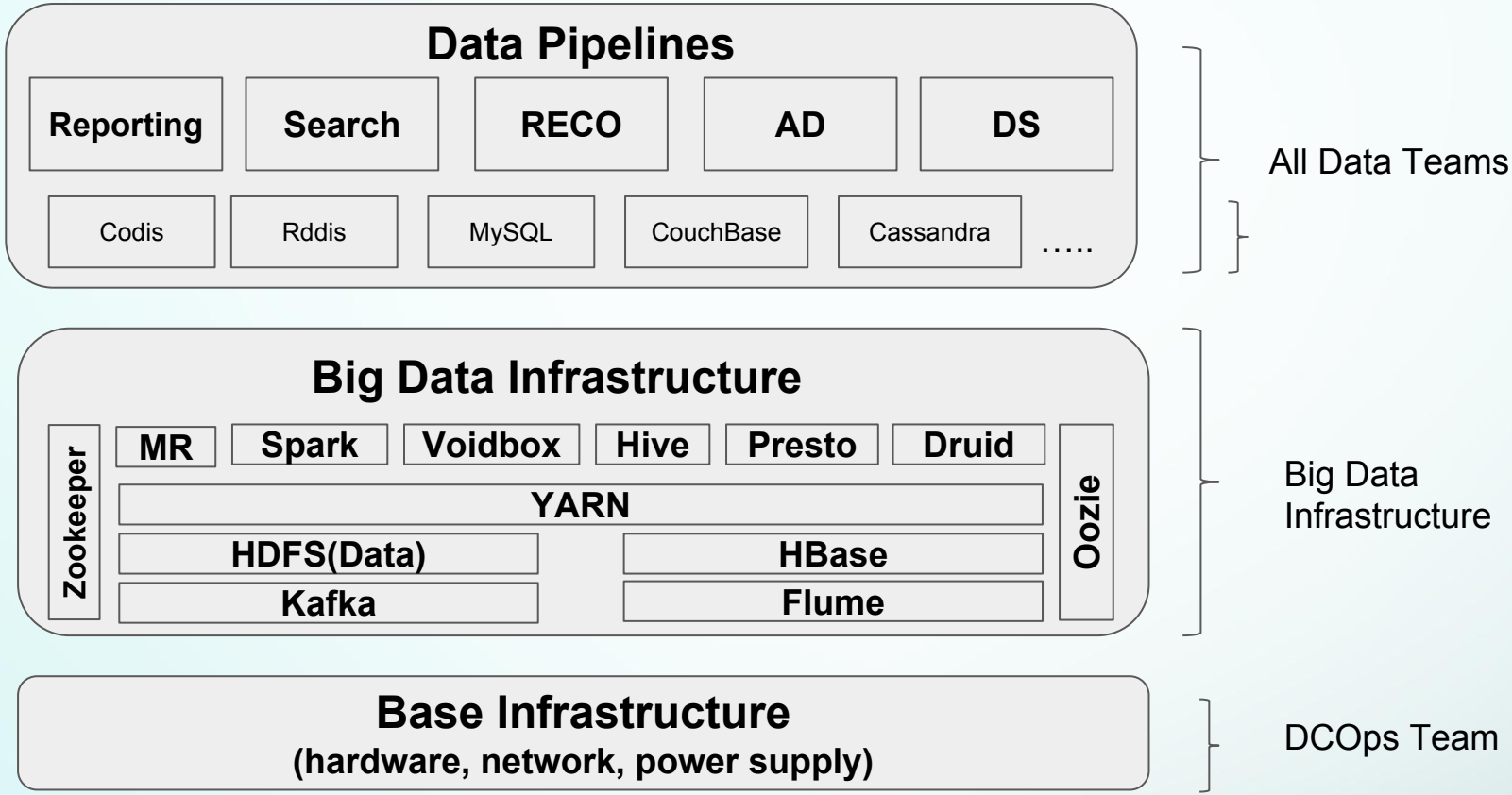
Why Hadoop Migration?

hulu

Challenge

- **Big data & complex applications**
  - Data volume: tens of PB
  - Applications: 20k ~30k per day
  - Mixed types of applications(MapReduce, Hive, Spark, docker...)
  - “all or nothing”
- **hour-level downtime**
  - Decrease business impact
- **Economical way**
  - Order as less new machine in LAS as possible
- **Across-team efforts**
  - DCOps/Devops/Data Teams in BJ/Seatle/SM
- **Customize infrastructure (code-level) to guarantee migration smoothly**
  - simplify application-level migration by enhancing infrastructure

Hulu Big Data Landscape



hulu

The Key For Hadoop Migration: Data Migration

- **Stateless system is easy**
  - YARN, Presto, Impala
- **Stateful system is harder**
  - **With small meta data**
    - Hive (MySQL), Zookeeper(Disk)
  - **With windowed data**
    - Kafka
  - **With huge data**
    - HDFS(metadata & files), HBase(storing data in HDFS)

hulu

Multi-DC HDFS

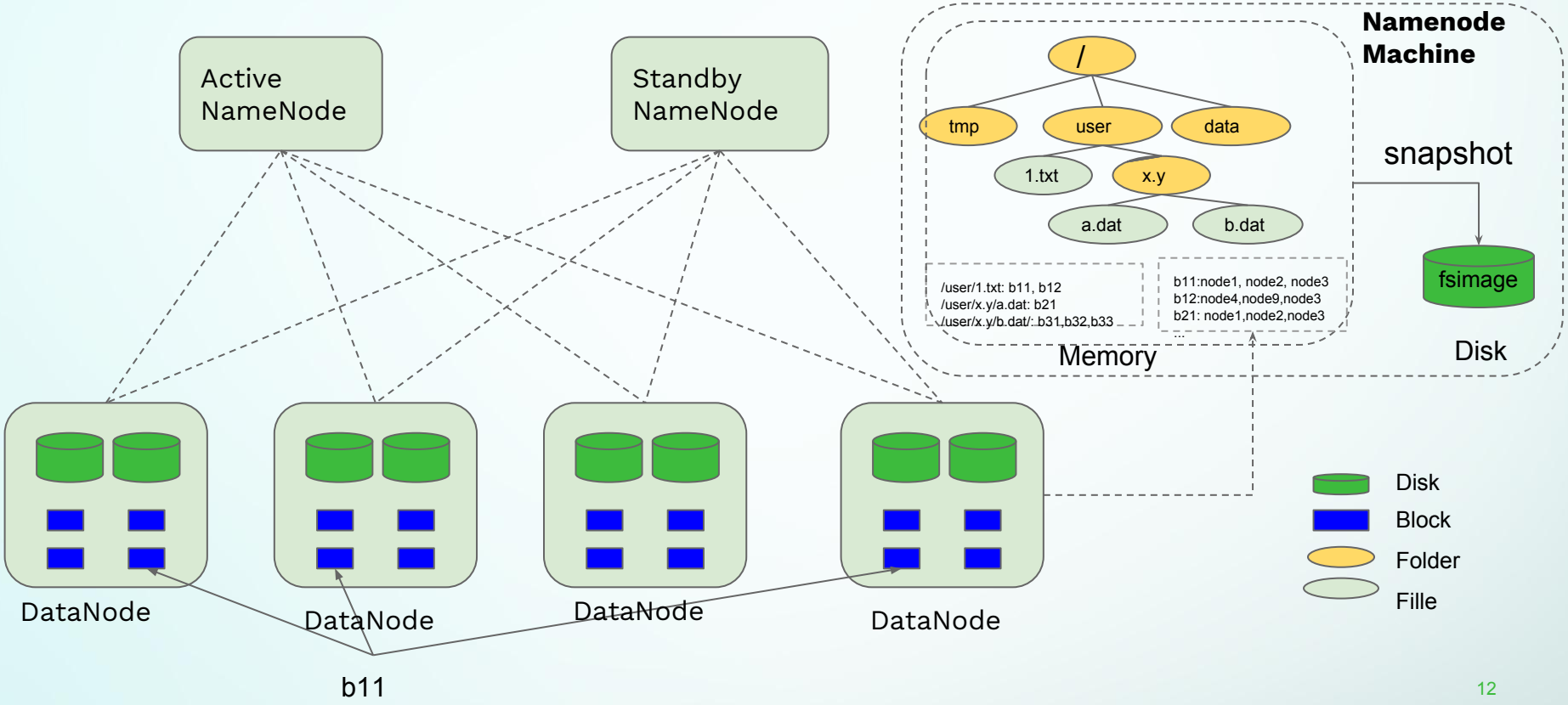
how to extend HDFS to support several data centers

HDFS: Introduction

- **HDFS: Hadoop Distributed File System**
  - Almost all hulu big data are stored on HDFS
- **HDFS namespaces**
  - HDFS federation
  - Three namespaces stored different kinds of data

hulu

HDFS: Introduction

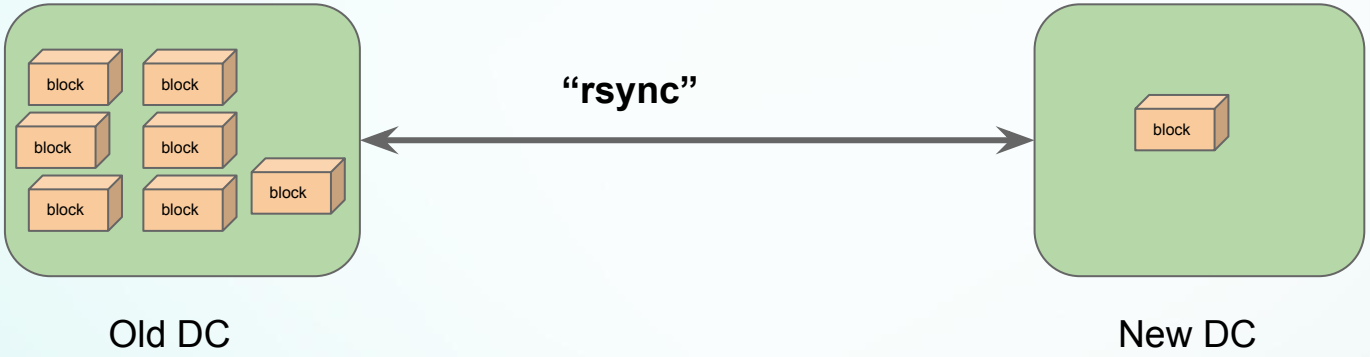


HDFS: Brief Introduction

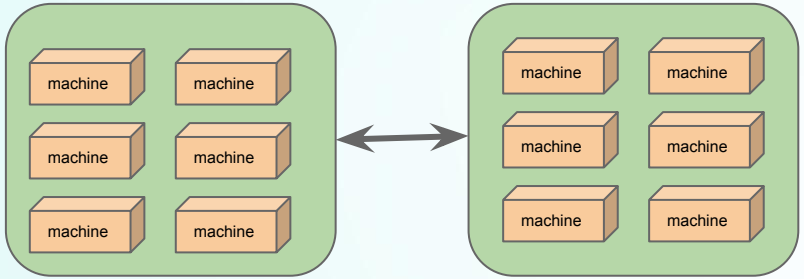
- **HDFS Namenode (meta data)**
  - directory tree
  - file-blocks mapping
  - block-locations mapping (report from every datanodes)
- **HDFS datanode(real data)**
  - blocks

hulu

Extended HDFS: Implement HDFS-level “rsync”



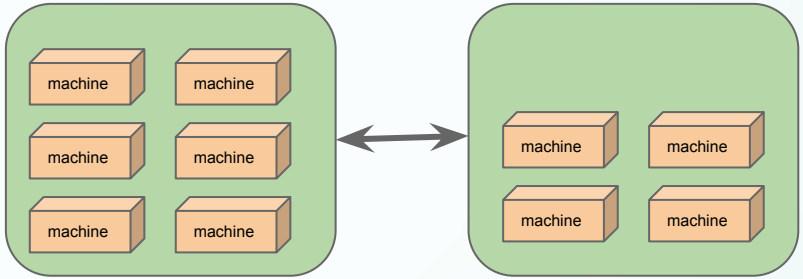
Extended HDFS: Solutions Comparison



Hadoop In DC1 Hadoop In DC2

Solution 1: Set up a mirror hadoop cluster in new DC

- ➔ **pros**
  - non-invasive to HDFS kernel
  - different hadoop versions
- ➔ **cons**
  - keep data consistent is hard
  - not transparent to users (address is changed)
  - order same number of machines
- ➔ Use in one small hadoop cluster(HDP → CDH)

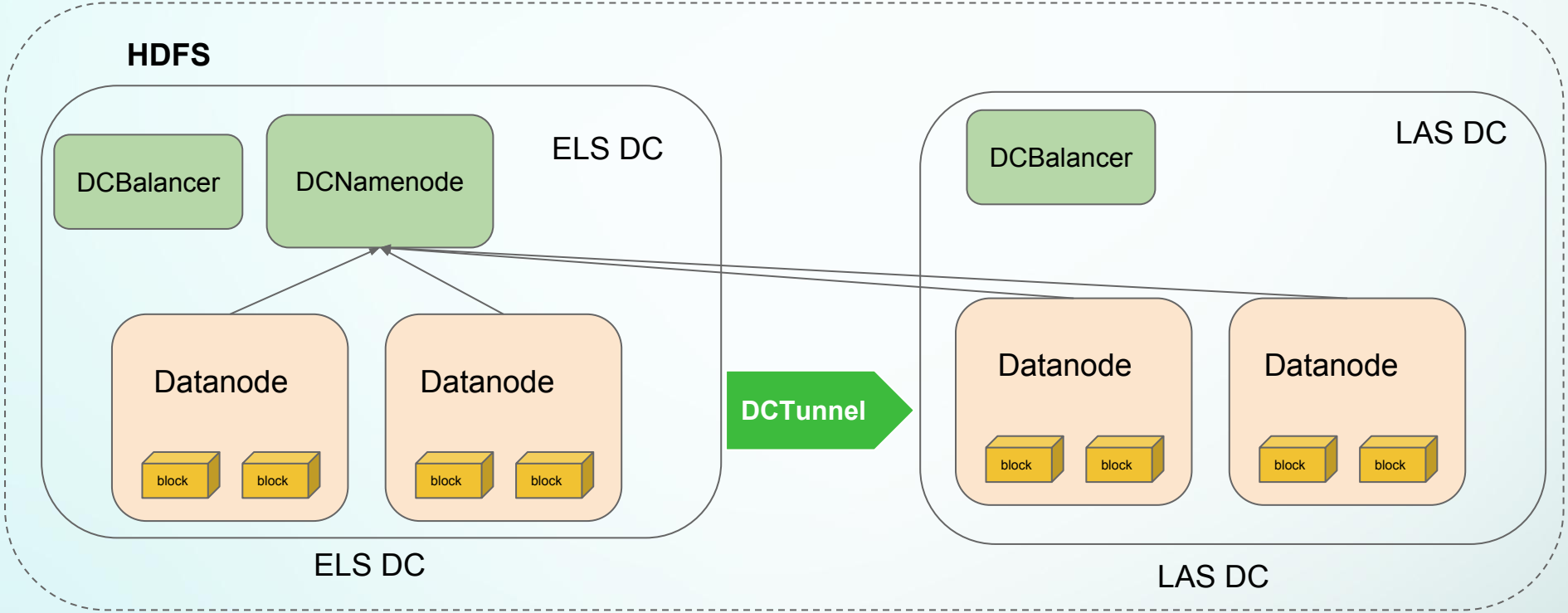


One single Hadoop In DC1 & DC2

Solution 2: Extend HDFS to support multiple DC

- ➔ **pros**
  - transparent
  - economical
- ➔ **cons**
  - invasive
  - risky
- ➔ Use in our biggest hadoop cluster(today only covers this part)

Architecture



hulu

DCNameNode: Add Data Center Level Topology

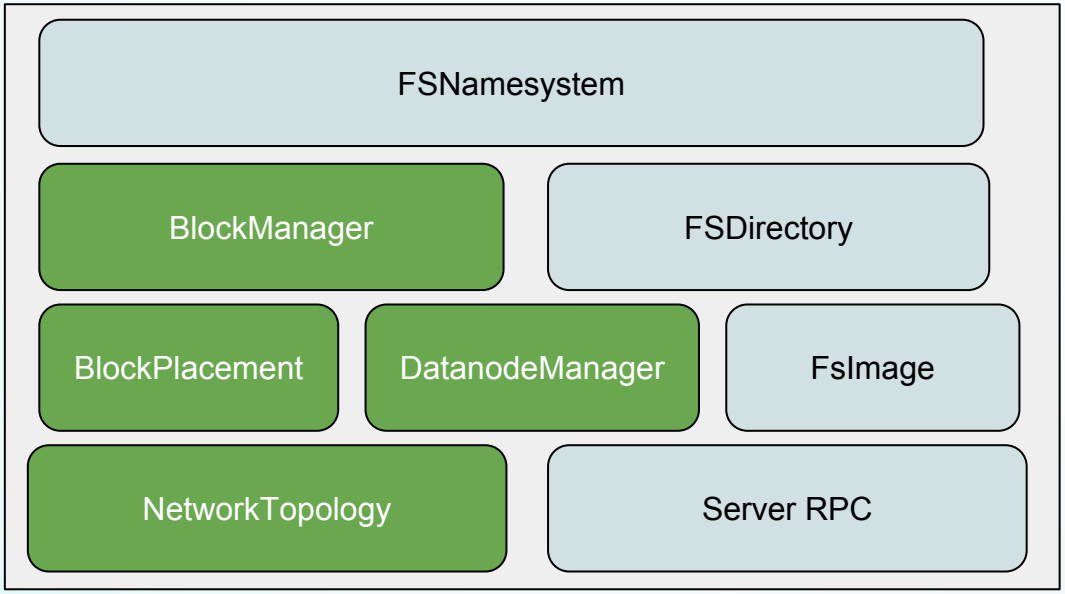
- Topology Configuration
  - Format “/datacenter/rack/node”, e.g
    - /datacenter1/rack1/node1
    - /datacenter2/rack2/node1
  - One of the data centers is “primary”, which is set by admin
- Read/Write Strategy
  - Read local data center first, and then the other
  - Write to primary data center only

hulu

DCNameNode: Data Center Replica Control

- Each file has a file-level replication factor
  - 3 by default
- Control global file replica across different data centers
  - e.g DC1:DC2 = 3 : 2
- Fine-grained replica control
  - each DC has a minimum and maximum replication factor(RF)
  - minimum file replica = min {file-level RF, DC minimum RF}
  - maximum file replica = min {file-level RF DC maximum RF}

DCNamenode: Modified From Namenode



hulu

DCBalancer: Balance Data In One Single DC

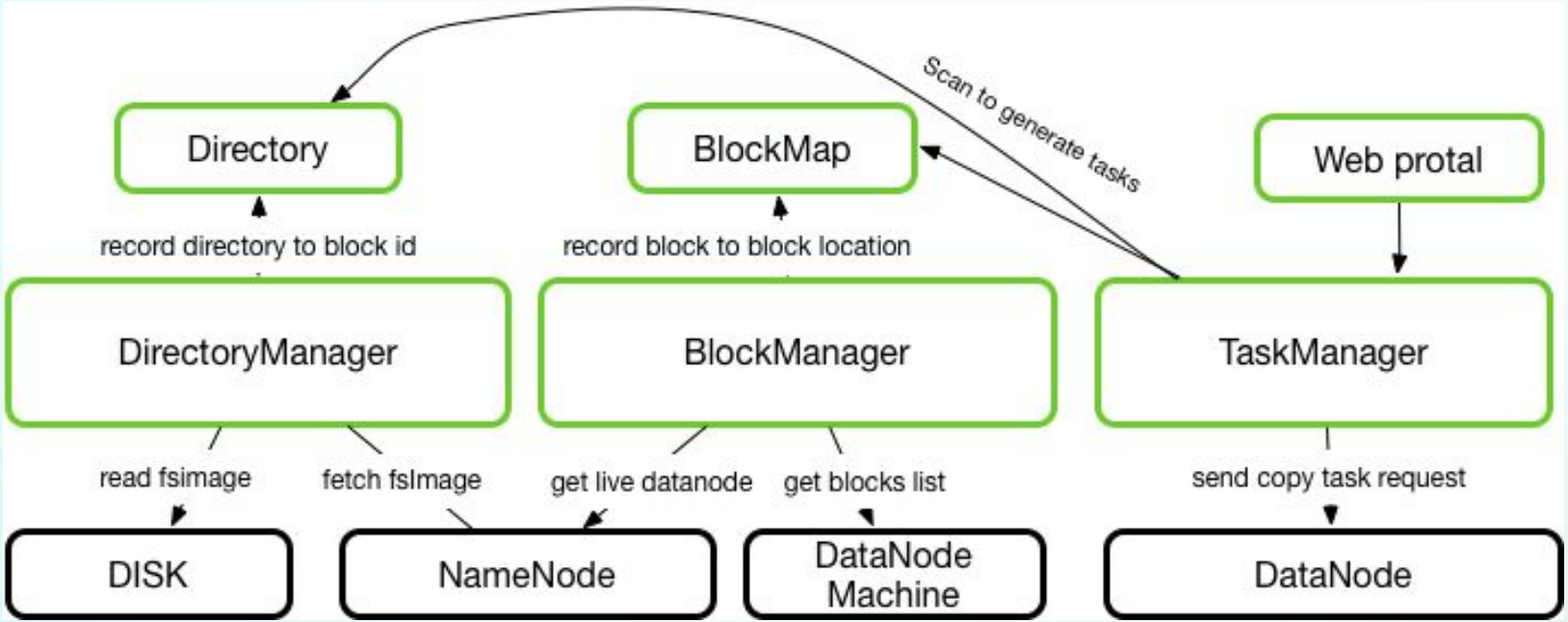
- Balancer
  - A HDFS component to balance data among different machines
- DCBalancer
  - Modified from HDFS Balancer
  - Only balance data among one specific data center

hulu

DCTunnel: Distributed Block Replication Scheduler

- Transfer data block across datacenters in a controllable way
- Features
  - Sync blocks according to folder-level whitelist & blacklist
  - Bandwidth limitation
  - Priority-based block replication
    - Missing blocks will be replicated quickly from the other datacenter
  - Quote adjustment
    - E.g. DC1: DC2=3:2
  - Web portal to display progress

DCTunnel: Architecture



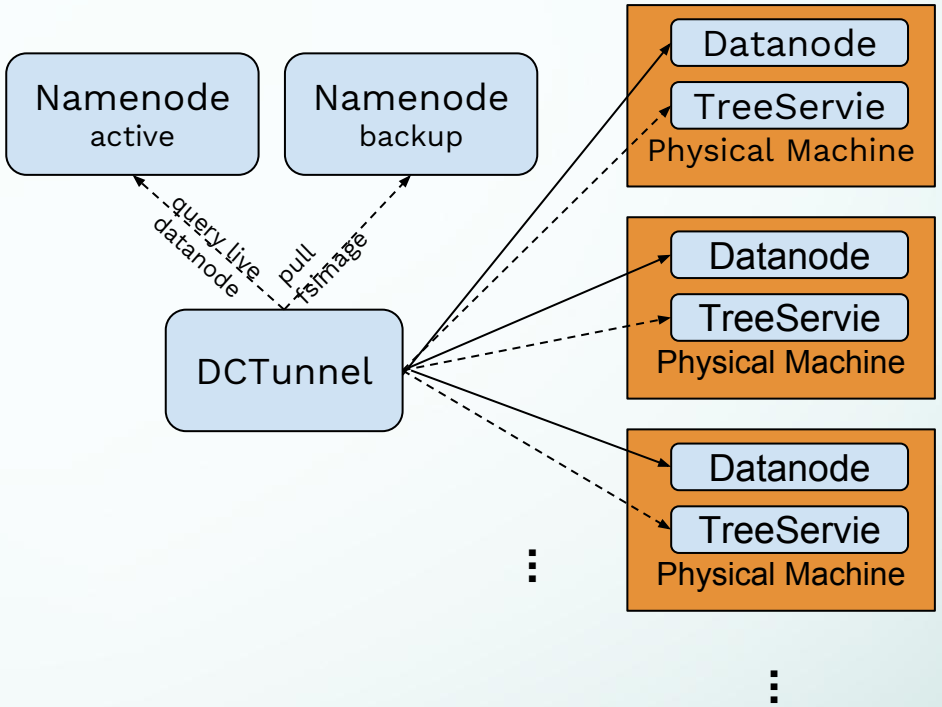
hulu

DCTunnel: Optimization

- Minimize impact to online namenode and datanode
- MVCC-based block location management
- Memory-friendly structure to store block-location mapping
- PID-based automatic bandwidth controller
- Optimized for “the long tail”

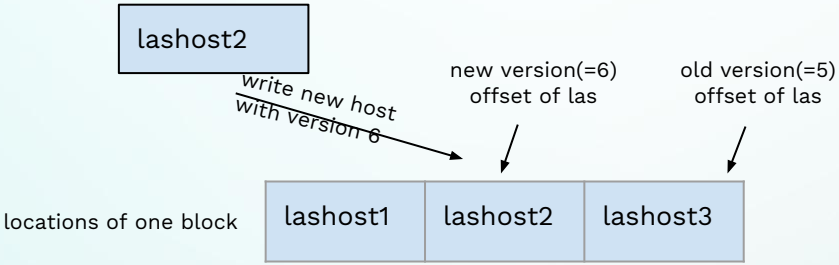
DCTunnel: Minimize Impact To Online NN & DN

- fetch fsimage from backup namenode
- fetch block location from tree service on every datanode host
- fetch live node list from active namenode
- replication task will send to datanode without touching namenode



DCTunnel: MVCC-based Block Location Management

- Challenge in block management
  - 0.3 billion blocks * 5 replica(3 replicas in DC1 and 2 replicas in DC2)
  - update every two hours(fetch metadata from fsimage, block-locations mapping from every datanodes)
  - replica changes frequently(moved, deleted or created)
- Solution:
  - MVCC(*Multiversion Concurrency Control*) based block location management

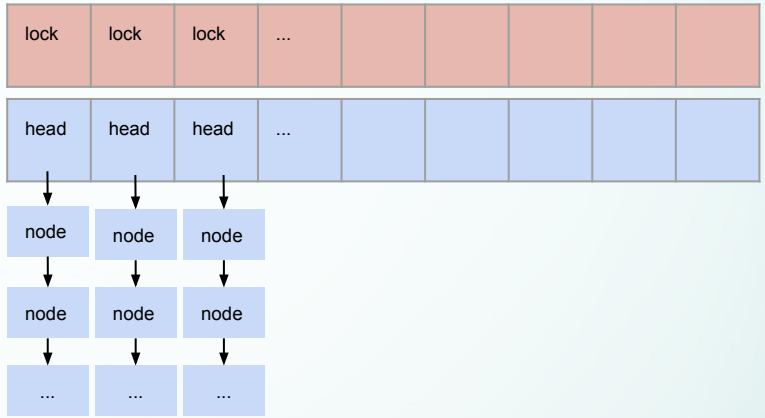


Read with version 5: lashost1, lashost2, lashost3
Read with version 6: lashost1, lashost2

hulu

DCTunnel: Memory-friendly Structure Storing Block-location

- Memory-friendly map structure
  - low overhead for each element
  - mark deleted and obsolete block collection to avoid huge GC
  - modified from HDFS internal structure LightWeightGSet(add slot-level fined-grained lock)



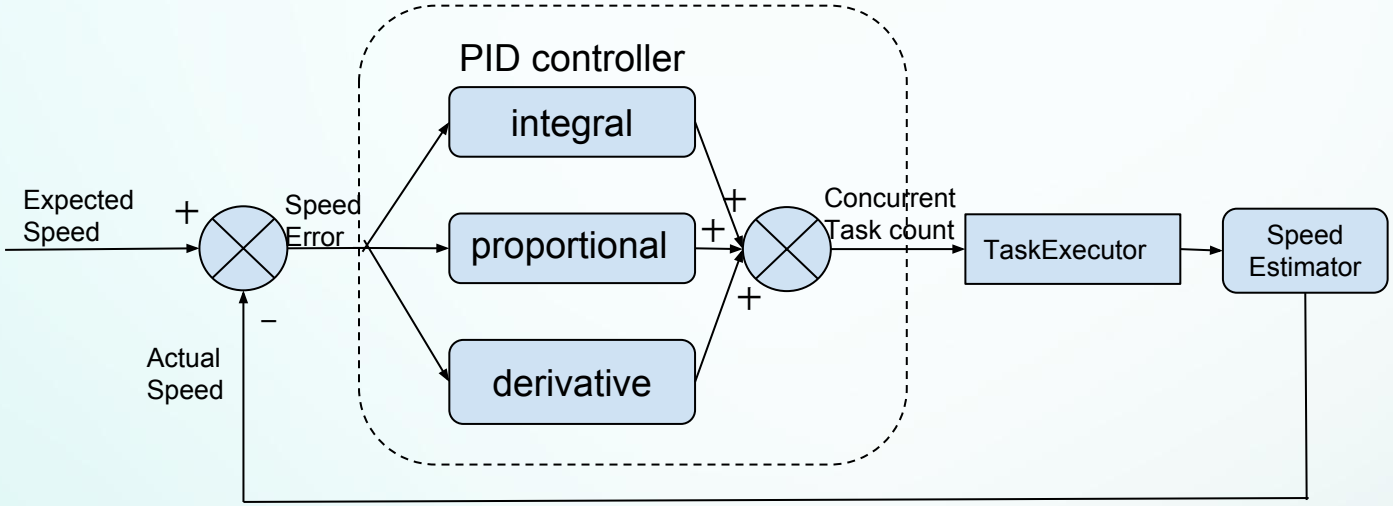
16M slot for 200M blocks, total consumed memory is 40GB(One machine is enough).

- OverHead for ~0.2B blocks

ConcurrentHashMap	ConcurrentSkipList	LightWeightGSet
40GB+	20GB+	4GB+

DCTunnel: PID-based Automatic Bandwidth Controller

- PID controller -- based on speed error



FRI: https://en.wikipedia.org/wiki/PID_controller

TaskExecutor: control how many migration task execute concurrently. Every task migrate a block from DC1 to DC2.

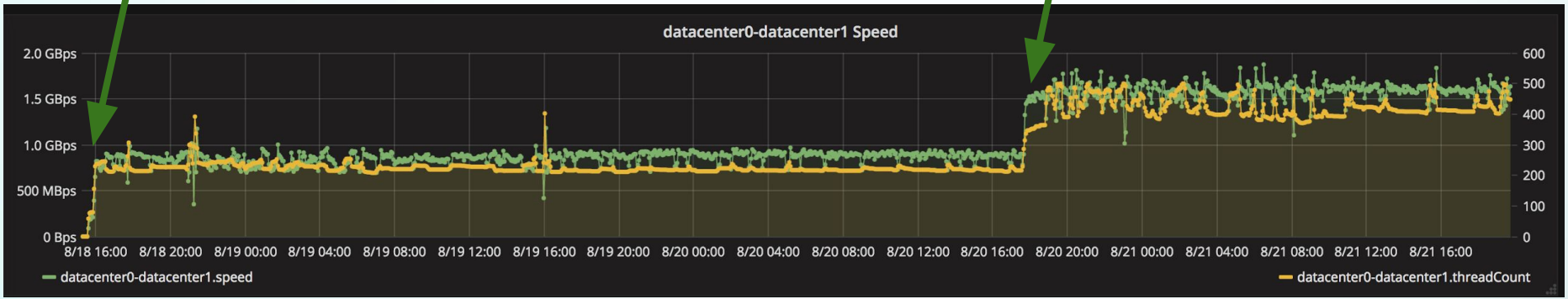
hulu

DCTunnel: PID-based Automatic Bandwidth Controller

- PID controller example

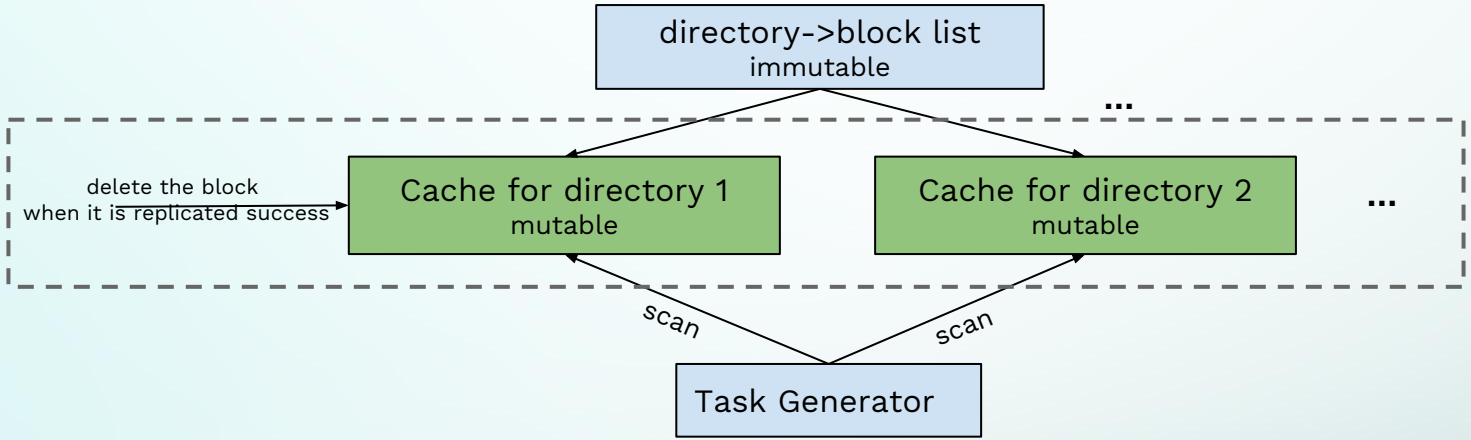
start with expected speed 800MB/s

change expected speed to 1.5GB/s

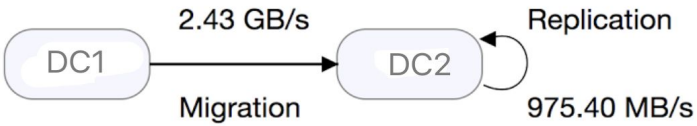


DCTunnel: Optimize For “The Long Tail”

- The Long Tail Problem
  - The meta data is updated every 2 hours
  - When most replicas(e.g. 99.9%) are migrated to new DC, finding out remaining blocks(e.g. 0.1%) becomes harder and harder
- Optimization: build *directory-blocks* cache for each folder to store only not-migrated blocks

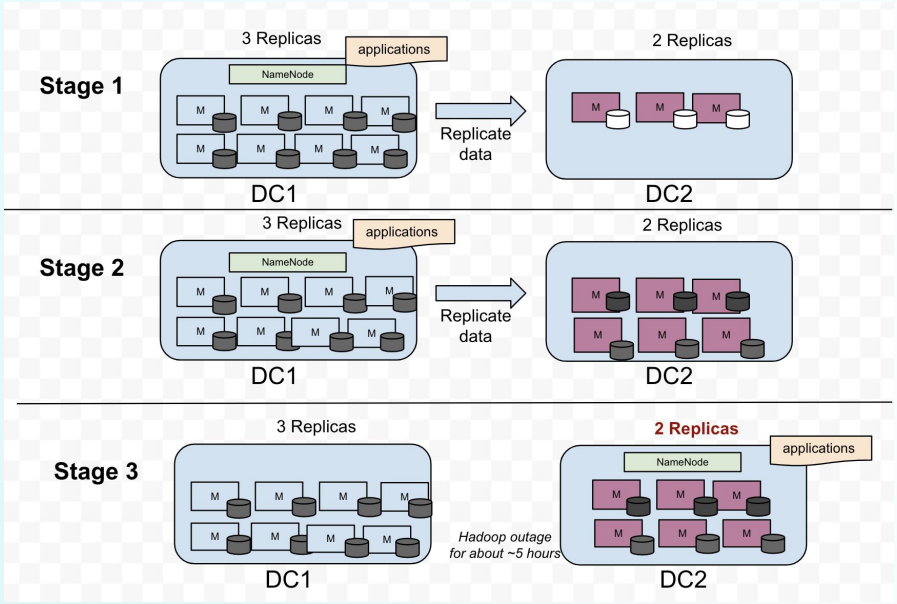


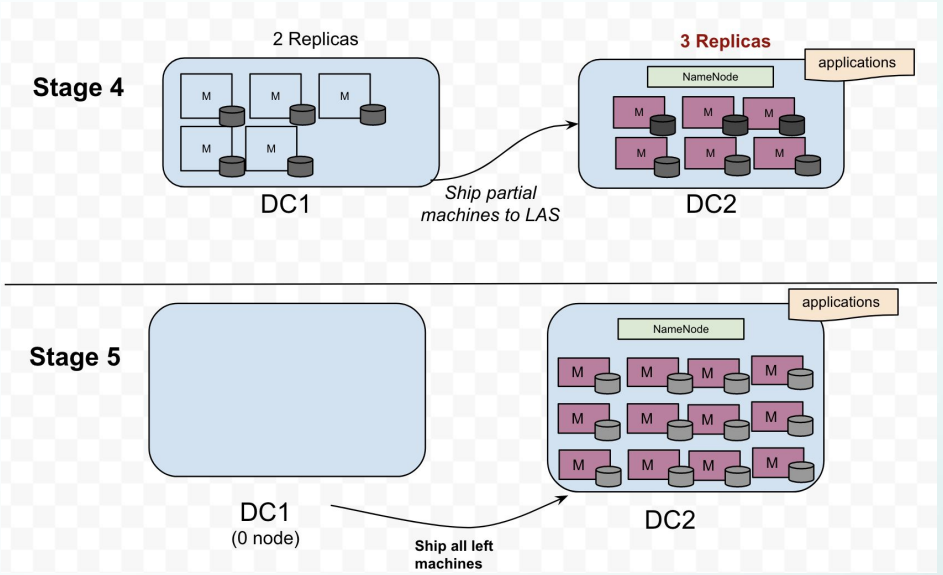
Demo



	[redacted] -> [redacted] (Migration)				[redacted] -> [redacted] (Replication)			
Namespace	Migration Speed	Block Progress	Migrated Blocks Ratio	Size Ratio	Replication Speed	Replica Progress	Migrated Replica Ratio	Size Ratio
apstore	217.26 MB/s	99%	[redacted]	[redacted]	58.07 MB/s	99%	[redacted]	[redacted]
beaconstore	734.31 MB/s	81%	[redacted]	[redacted]	0 B/s	81%	[redacted]	[redacted]
warehousestore	1.77 GB/s	54%	[redacted]	[redacted]	248.23 MB/s	52%	[redacted]	[redacted]

Execution


Stage 1
3 Replicas
applications
NameNode
M
Replicate data
2 Replicas
DC1
DC2
Stage 2
3 Replicas
applications
NameNode
Replicate data
2 Replicas
DC1
DC2
Stage 3
3 Replicas
2 Replicas
applications
NameNode
Hadoop outage for about ~5 hours
DC1
DC2


Stage 4
2 Replicas
3 Replicas
applications
NameNode
M
DC1
Ship partial machines to LAS
DC2
Stage 5
applications
NameNode
DC1
(0 node)
Ship all left machines
DC2

hulu

Experience Gained

sharing experiences

hulu

Experience

- Network
  - Fully control network utilization between two DC
  - Avoid machine hotspot
- Validation
  - validate both metadata and data
  - Ensure data is copied accurately
- Monitoring
  - Measure everything to know your bottleneck(can’t fail behind)
- Fully test and rehearsal before actual migration

hulu

THANK YOU