Sinbad

Leveraging Endpoint Flexibility in Data-Intensive Clusters

Mosharaf Chowdhury,
Srikanth Kandula, Ion Stoica

amplab UC Berkeley
Microsoft Research

Communication is Crucial for Analytics at Scale

Performance

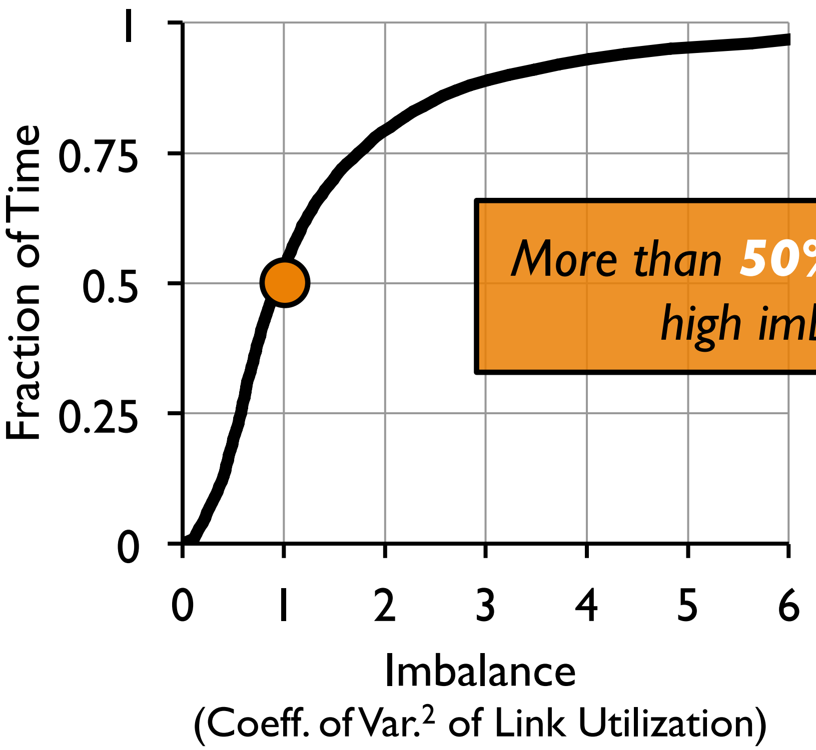
Facebook analytics jobs spend **33%** of their runtime in communication[1]

As in-memory systems proliferate,
the network is likely to become the ***primary bottleneck***

1. *Managing Data Transfers in Computer Clusters with Orchestra, SIGCOMM'2011*

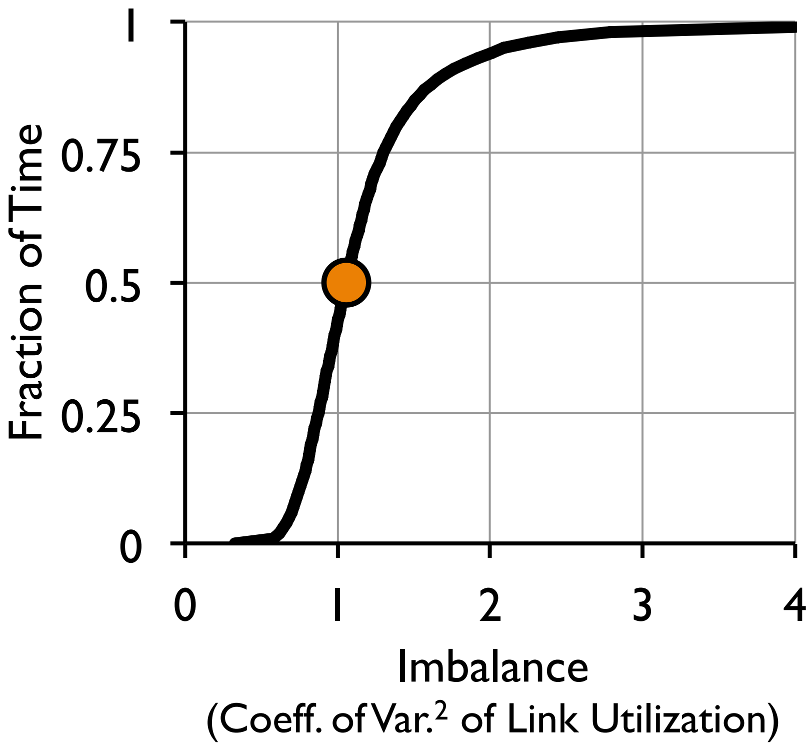
Network Usage is Imbalanced[1]

Facebook



*More than **50%** of the time, links have high imbalance ($C_v > 1$).*

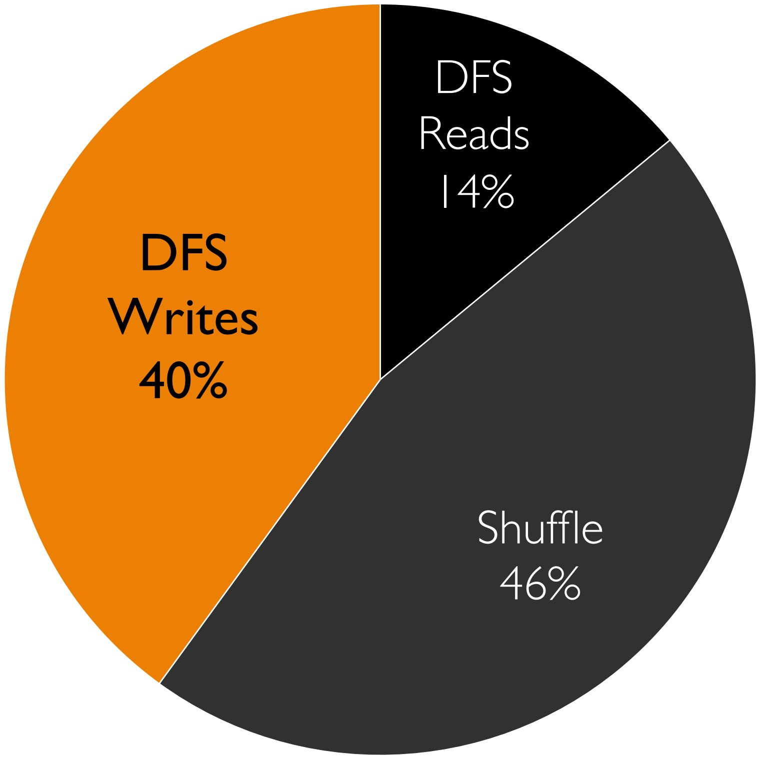
Bing



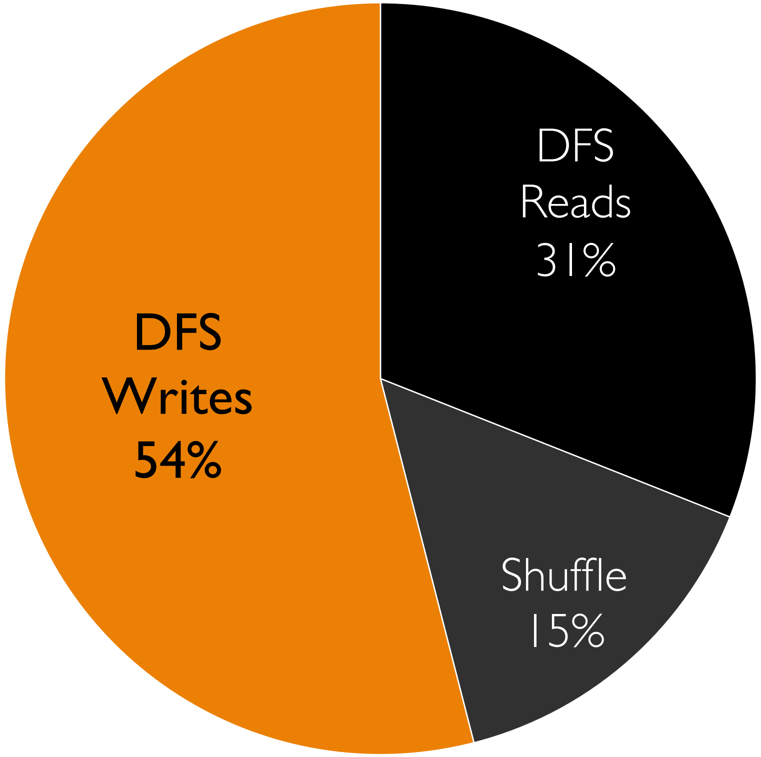
1. *Imbalance considering all cross-rack bytes. Calculated in 10s bins.*
2. *Coefficient of variation, C_v = (stdev/mean).*

What Are the Sources of *Cross-Rack* Traffic?

Facebook



Bing

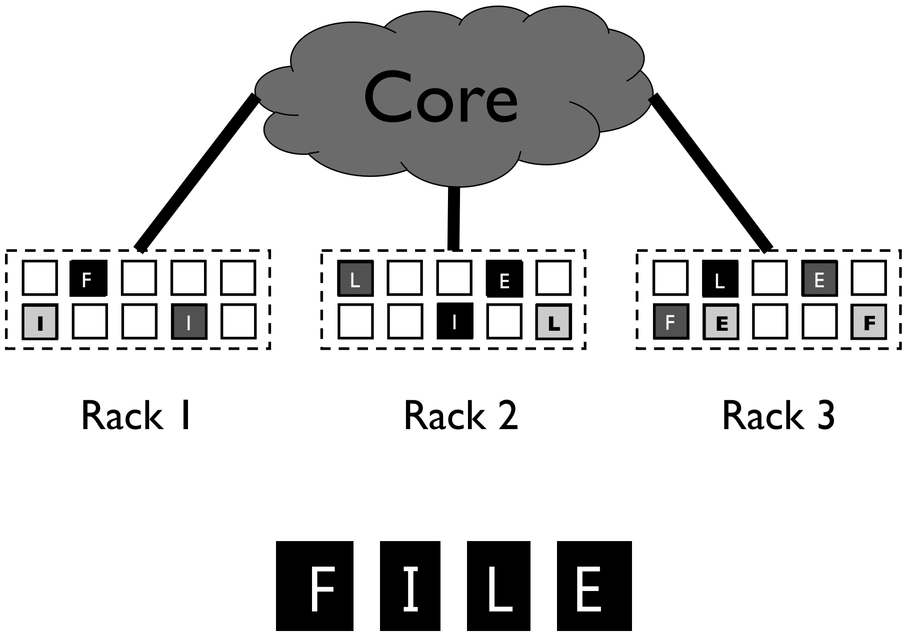


Write Sources

1. Ingestion
2. Pre-processing
3. Job outputs

1. DFS = Distributed File System

Distributed File Systems (DFS)



Pervasive in BigData clusters

- E.g., GFS, HDFS, Cosmos
- Many frameworks interact w/ the same DFS

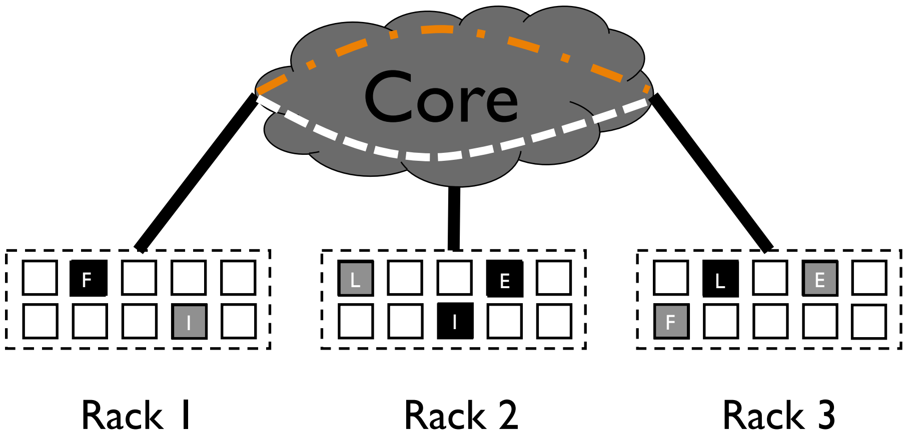
Files are divided into blocks

- 64MB to 1GB in size

Each block is replicated

- To **3** machines for ***fault tolerance***
- In **2** fault domains for ***partition tolerance.***
- **Uniformly** placed for a **balanced storage**

Synchronous operations



Fixed

- Sources
- Destinations

Flexible

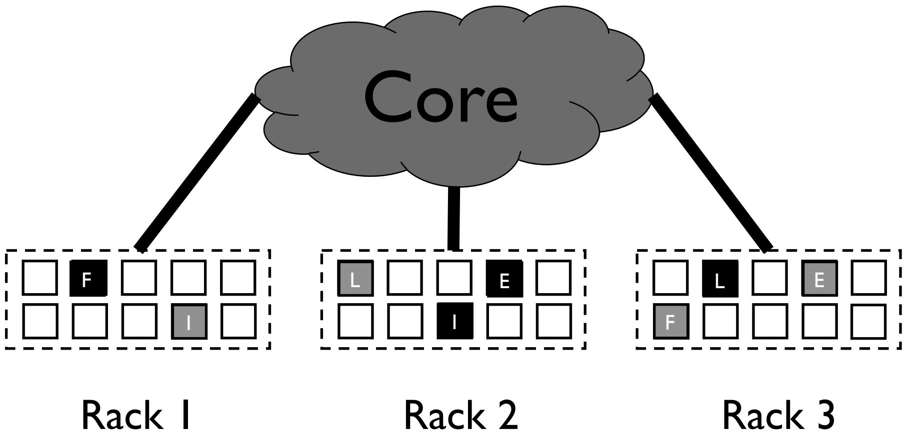
Paths
Rates

How to handle DFS flows?

A few seconds long

Hedera, VLB,
Orchestra, Coflow,
MicroTE, DevoFlow, …

Distributed File Systems (DFS)



Pervasive in BigData clusters

- Many frameworks interact w/ the same DFS

Files are divided into blocks

- 64MB to 1GB in size

Each block is replicated

- To **3** machines for ***fault tolerance***
- In **2** fault domains for ***partition tolerance.***
- **Uniformly** placed for a **balanced storage**

Replica locations do not matter
as long as constraints are met

Flexible

■ Sources
■ Destinations ✔

Flexible

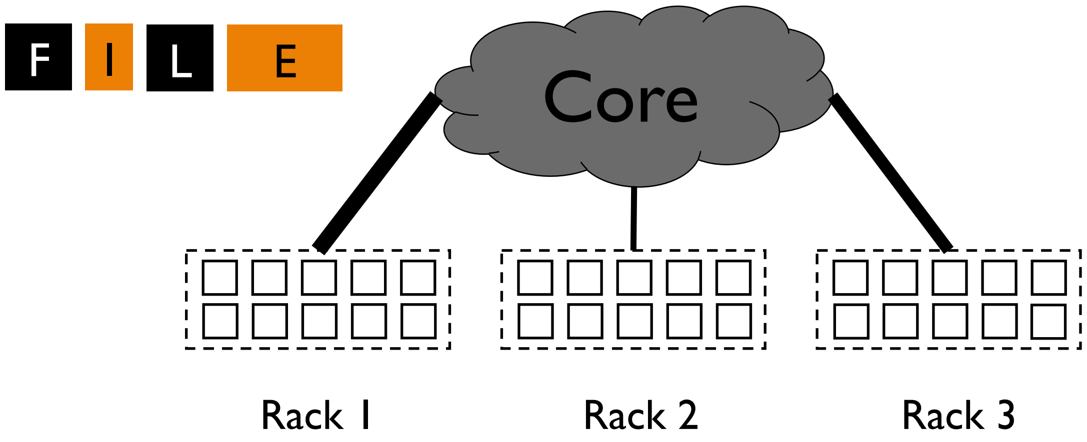
Paths
Rates

Sinbad

Steers flexible replication traffic away from hotspots

1. **Faster Writes** — *By avoiding contention during replication*
2. **Faster Transfers** — *Due to more balanced network usage closer to edges*

The *Distributed Writing* Problem is **NP-Hard**

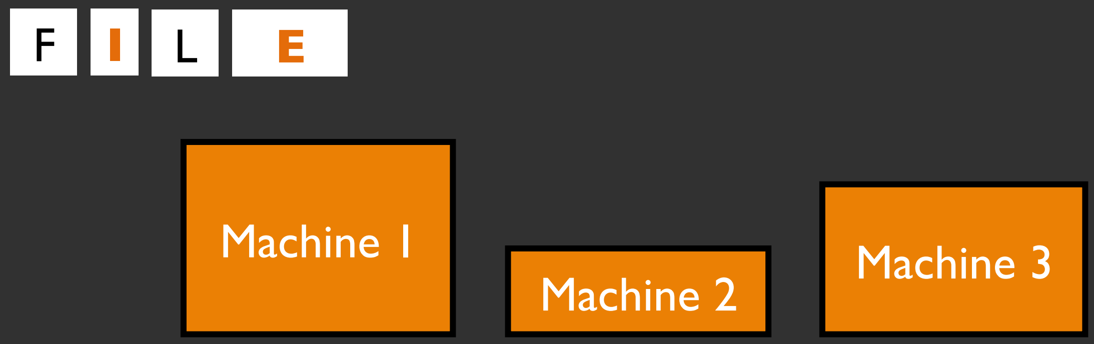


Given

- Blocks of different size, and
- Links of different capacities,

Place blocks to minimize

- The average block write time
- The average file write time



Job Shop Scheduling

Given

- *Jobs* of different *length*, and
- *Machines* of different *speed*,

Schedule jobs to minimize

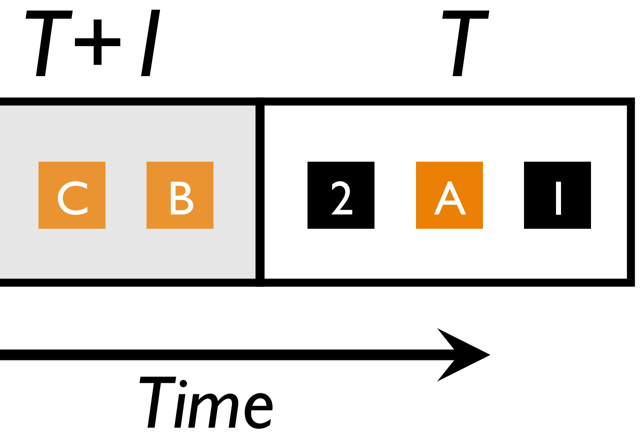
- The average *job* completion time

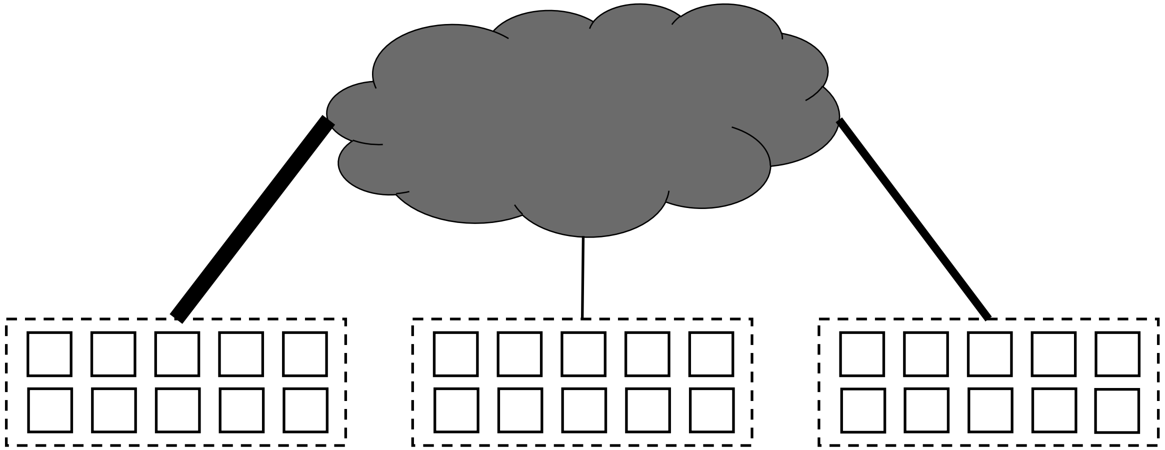
The Online Distributed Writing Problem is NP-Hard

Lack of future knowledge about the

- Locations and durations of network hotspots,
- Size and arrival times of new replica placement requests

How to Make it Easy?





Assumptions

1. Link utilizations are **stable**
2. All blocks have the **same size**

Theorem:

Greedy placement minimizes average block/file write times

How to Make it Easy? – In Practice

Reality

1. Average link utilizations are *temporarily* stable[1,2]
2. Fixed-size large blocks write **93%** of all bytes

Assumptions

1. Link utilizations are **stable**
2. All blocks have the **same size**

1. Utilization is considered stable if its average over next x seconds remains within ±5% of the initial value
2. Typically, x ranges from 5 to 10 seconds. Time to write a 256MB block assuming 50MBps write throughput is 5 seconds

Sinbad

Performs two-step ***greedy*** *replica placement*

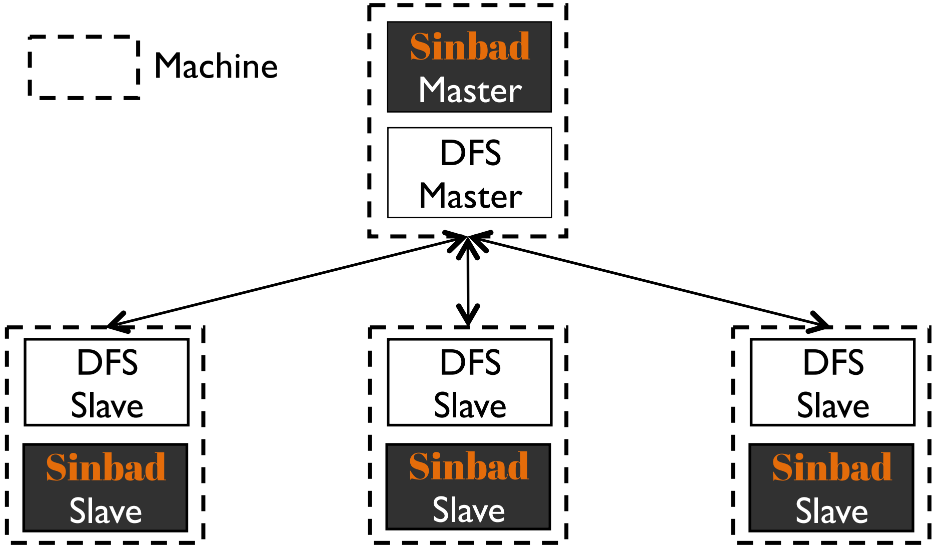
1. Pick the ***least-loaded*** link
2. Send a block from the file with the ***least-remaining*** blocks through the selected link

Sinbad Overview

Centralized master-slave architecture

- Agents collocated with DFS agents

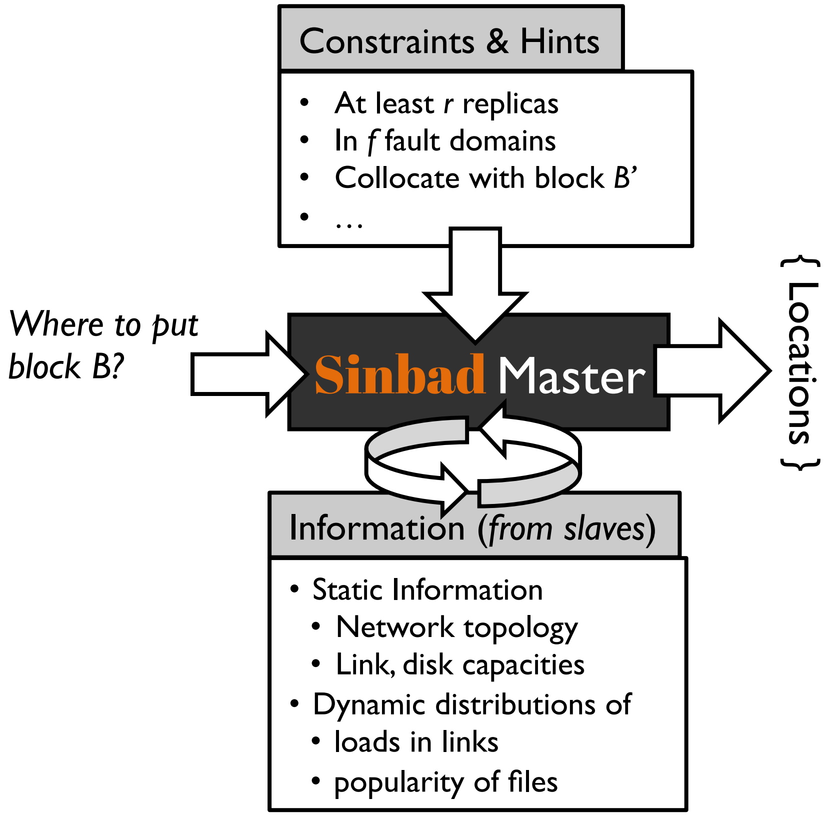
Slaves periodically report information



Sinbad Master

Performs network-aware replica placement for large blocks

- Periodically estimates network hotspots
- Takes greedy online decisions
- Adds hysteresis until next measurement



Evaluation

A 3000-node trace-driven simulation matched against a 100-node EC2 deployment

1. Does it improve performance?
2. Does it balance the network?
3. Does the storage remain balanced?

YES

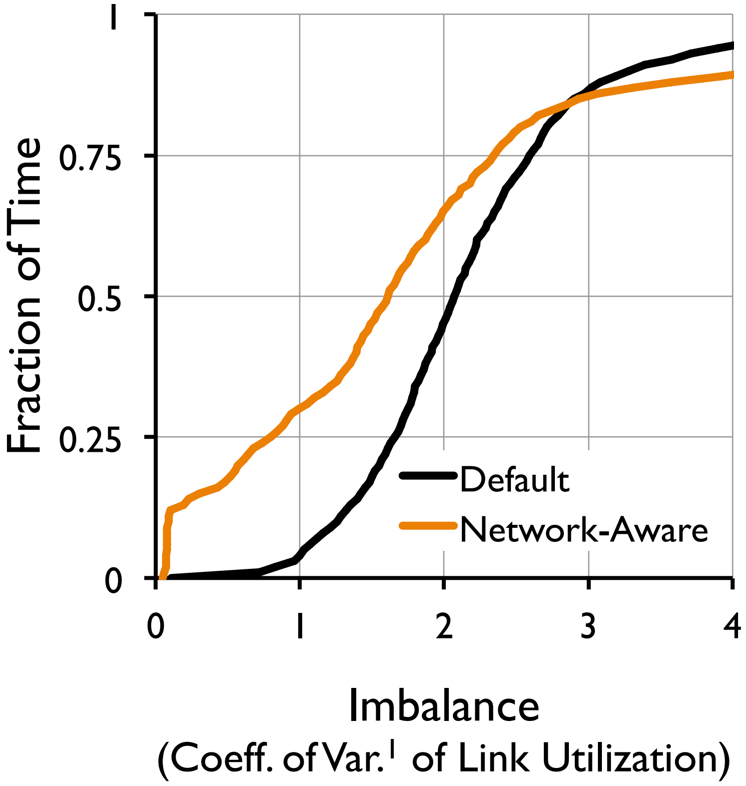
Faster

	Job Improv.	DFS Improv.
Sim	**1.39X**	**1.79X**
Exp	**1.26X**	**1.60X**

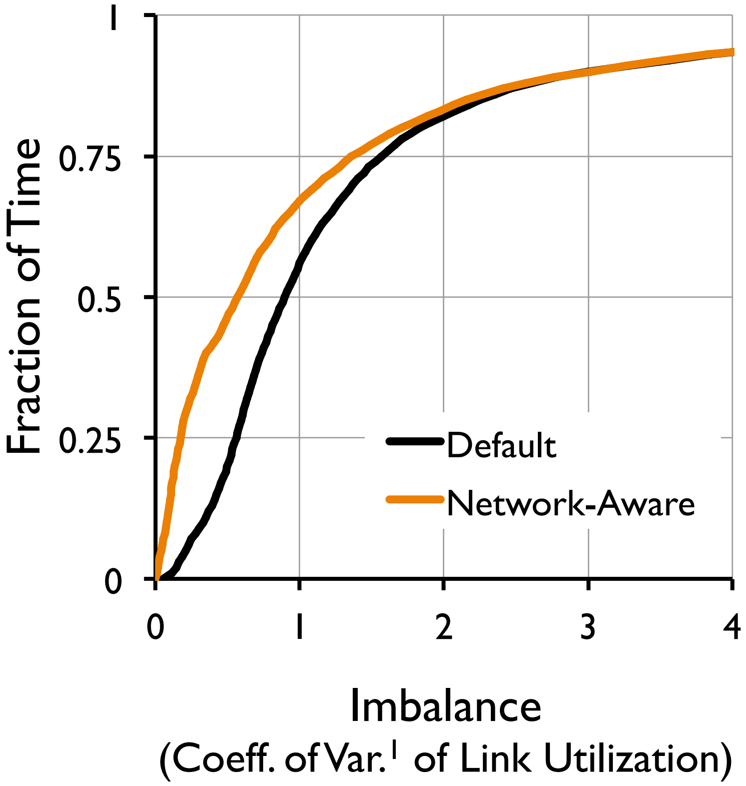
In-memory storage

More Balanced

EC2 Deployment



Facebook Trace Simulation



What About Storage Balance?

Network is imbalanced in the short term;
but, in the long term,
hotspots are uniformly distributed

#1

Increase Capacity

Fatter links/interfaces
Increase Bisection B/W

Fat tree, VL2, DCell, BCube, F10, …

#2

Decrease Load

Data locality
Static optimization

Fair scheduling, Delay scheduling, Mantri, Quincy, PeriSCOPE, RoPE, Rhea, …

#3

Balance Usage

Manage elephant flows
Optimize intermediate comm.

Valiant load balancing (VLB), Hedera, Orchestra, *Coflow, MicroTE, DevoFlow, …*

Sinbad

Greedily steers replication traffic away from hotspots

- Improves job performance by making the network more balanced
- Improves DFS write performance while keeping the storage balanced
- Sinbad will become increasingly more important as storage becomes faster

We are planning to deploy **Sinbad** at facebook

Trace Details

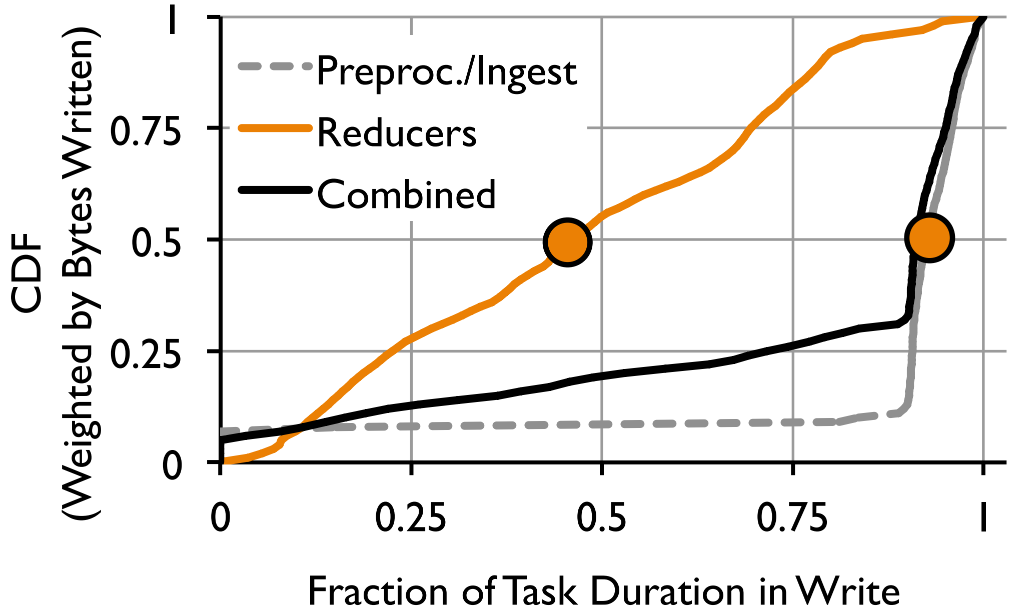
	Facebook	Microsoft Bing
Period	Oct 2010	Mar – Apr 2012
Duration	1 Week	1 Month
Framework	Hadoop MapReduce	SCOPE
Jobs	175,000	O(10,000)
Tasks	30 Millions	O(10 Millions)
File System	HDFS	Cosmos
Block Size	256MB	256MB
Number of Machines	3,000	O(1,000)
Number of Racks	150	O(100)
Core : Rack Oversubscription	10 : 1	Better / Less Oversubscribed

Writer Characteristics

***42%** of the reducers and **91%** of other writers spend at least **50%** of run time*

37% of all tasks write to the DFS

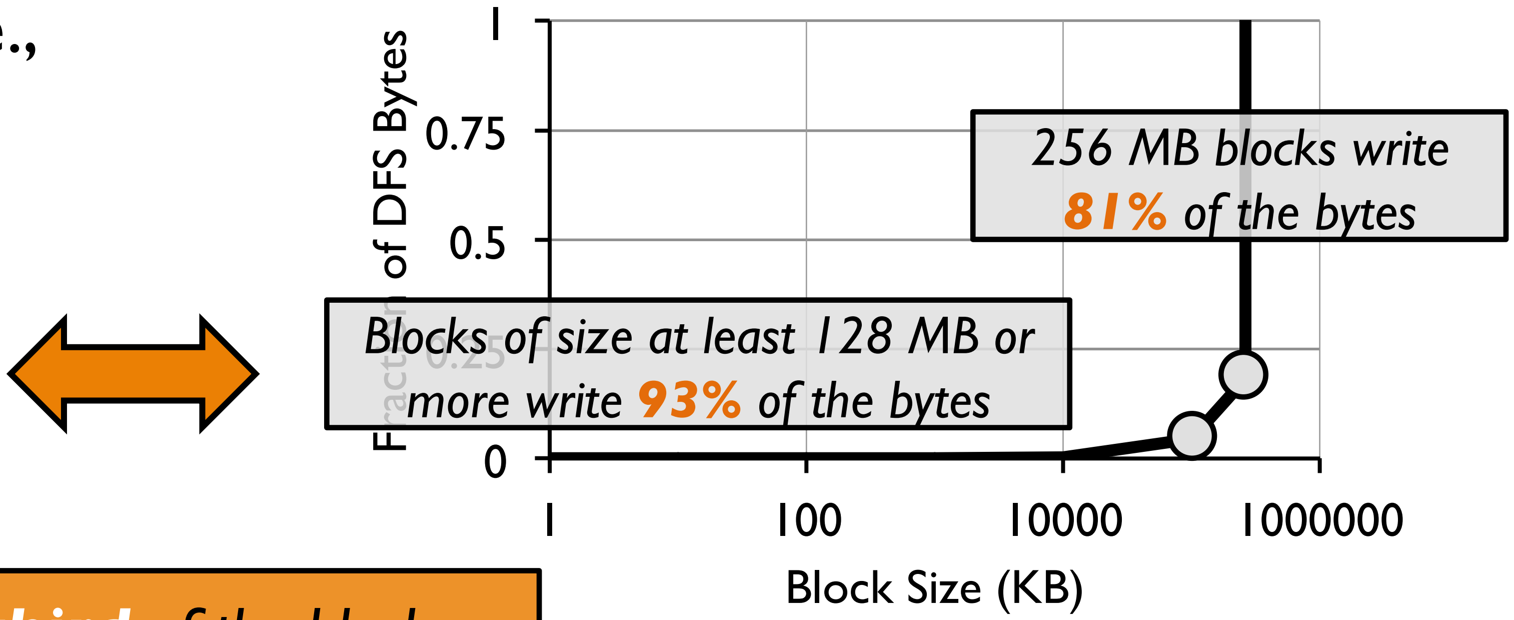
Writers spend large fractions of runtime in writing



Big Blocks Write Most Bytes

30% blocks are full sized, i.e., they are capped at 256MB

35% blocks are of at least 128 MB or more in size



One third *of the blocks generate* ***almost all*** *replication bytes*

Big Blocks Write Most Bytes

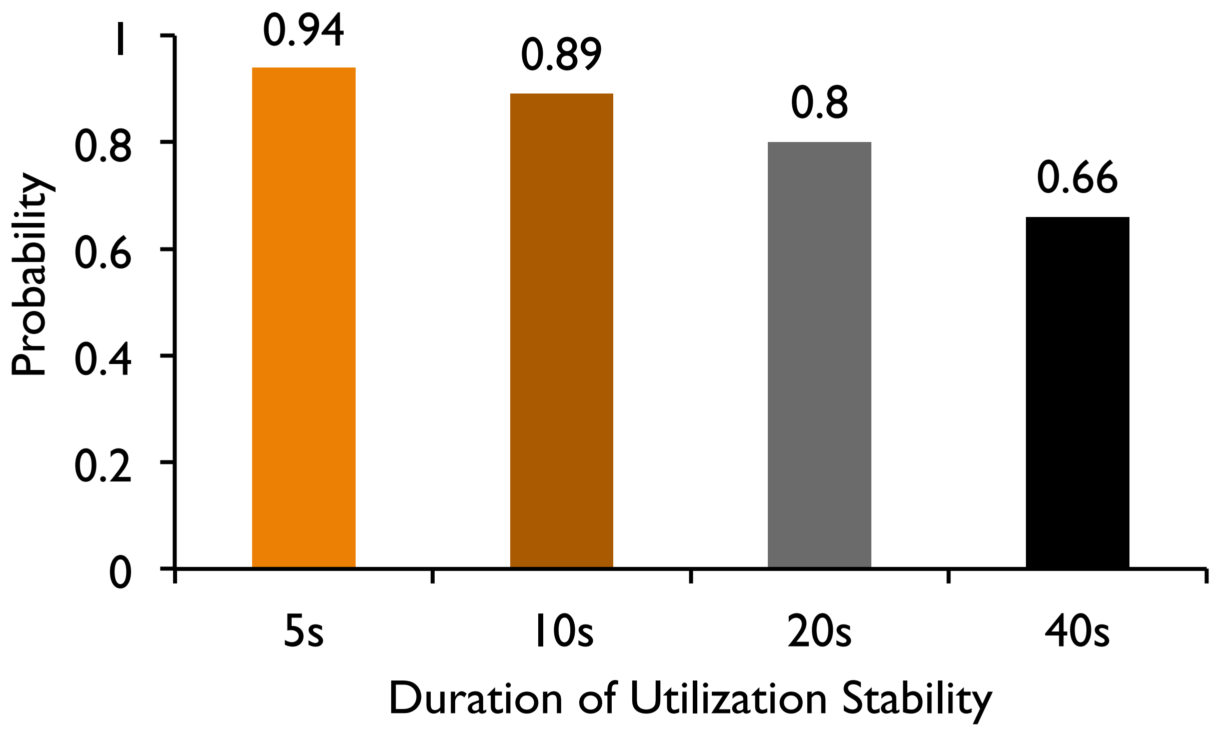
30%

Blocks are large
(256 MB)

81%

Bytes are written
by large blocks

Hotspots are Stable[1] *in the Short Term*



Long enough[2] to write a block even if disk is the bottleneck

1. Utilization is considered stable if its average over next x seconds remains within ±5% of the initial value

- If the block size is fixed, and
- hotspots are temporarily stable,

the solution is …

Greedy Placement

Thm.

Greedy assignment of blocks to the least-loaded link in the least-remaining-blocks-first order is optimal for minimizing the average write times

Utilization Estimator

Utilizations updated using EWMA at Δ intervals

- $v_{new} = \alpha * v_{measured} + (1-\alpha) * v_{old}$
- We use $\alpha = 0.2$

Update interval (Δ)

- Too small a Δ creates overhead, but too large gives stale data
- We use $\Delta = 1$ second right now
- Missing updates are treated *conservatively*, as if the link is fully loaded

Utilization Estimator

Hysteresis after each placement decision

- *Temporarily* bump up estimates to avoid putting too many blocks in the same location
- Once the next measurement update arrives, hysteresis is removed and the actual estimation is used

Hysteresis function

- Proportional to the size of block just placed
- Inversely proportional to the time remaining till next update

Implementation

Implemented and integrated with HDFS

- Pluggable replica placement policy on *https://github.com/facebook/hadoop-20*
- Slaves are integrated into DataNode and the master into NameNode
- Update comes over the Heartbeat messages
- Few hundred lines of Java

Methodology

HDFS deployment in EC2

- Focus on large (in terms of network bytes) jobs only
- 100 m1.xlarge nodes with 4x400GB disks
- 55MBps/disk maximum write throughput
- 700+Mbps/node during all-to-all communication

Trace-driven simulation

- Detailed replay of a day-long Facebook trace (circa October 2010)
- 3000-node,150-rack cluster with 10:1 oversubscription

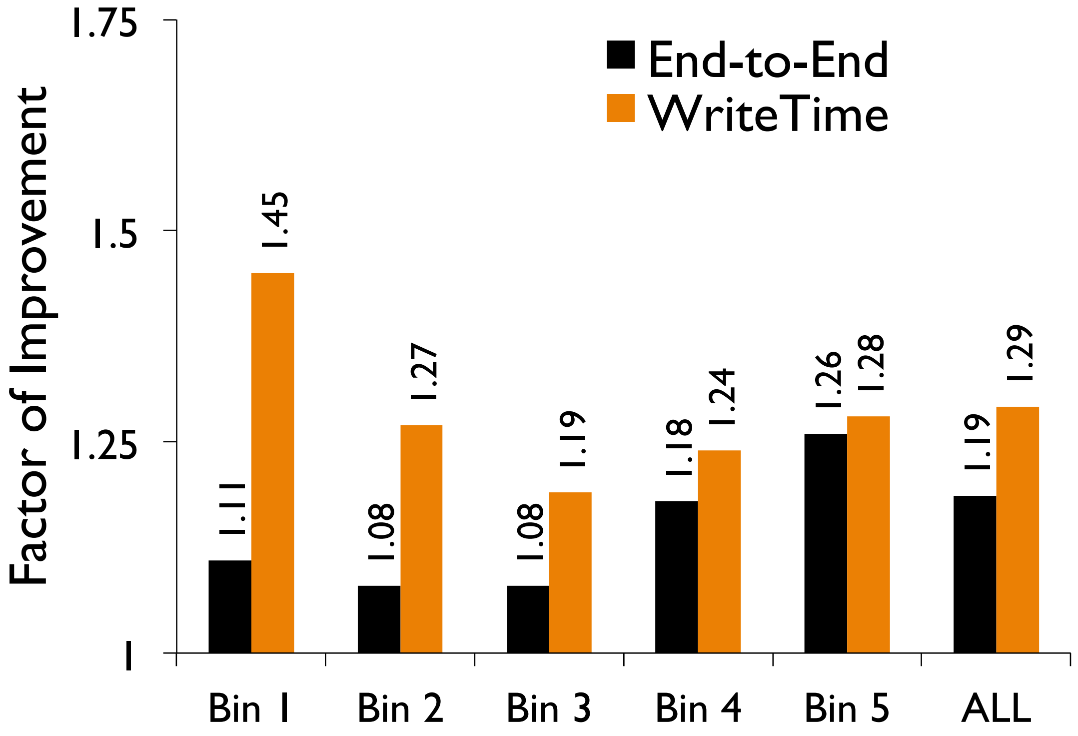
Breakdown [Time Spent]

Jobs spend varying amounts of time in writing

- Jobs in Bin-1 the least and jobs in Bin-5 the most

No clear correlation

- We improve block writes w/o considering job characteristics



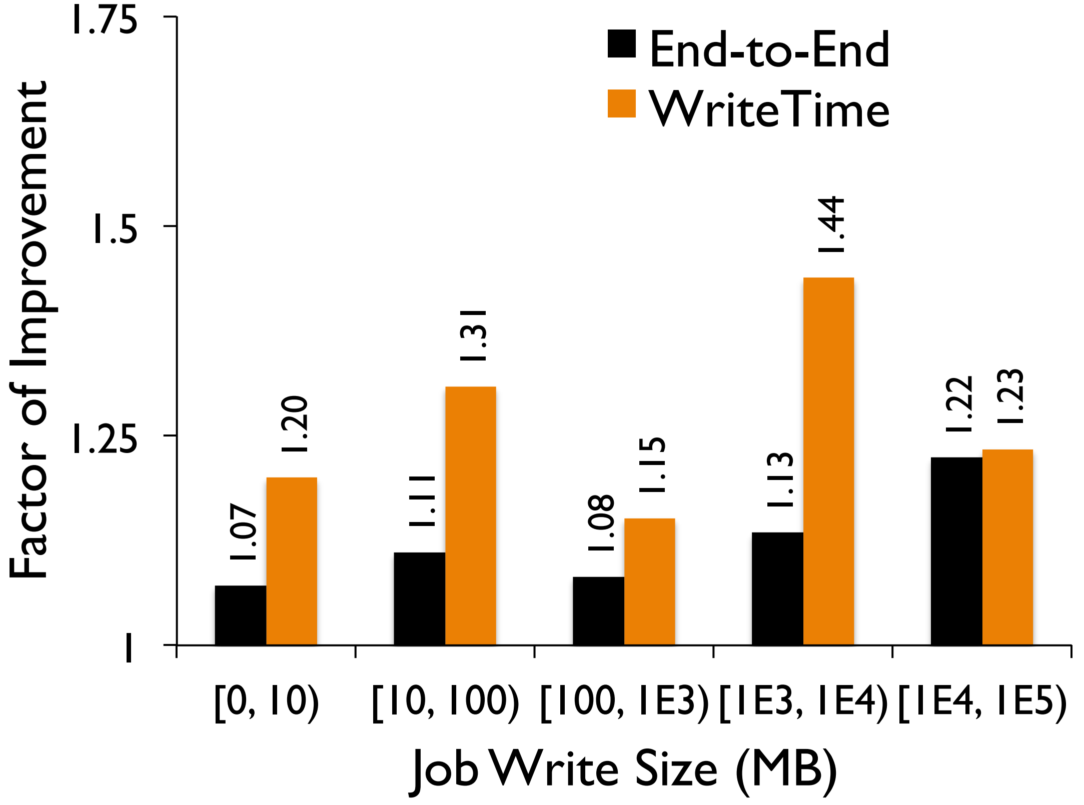
Breakdown [Bytes Written]

Jobs write varying amounts to HDFS as well

- Ingestion and pre-processing jobs write the most

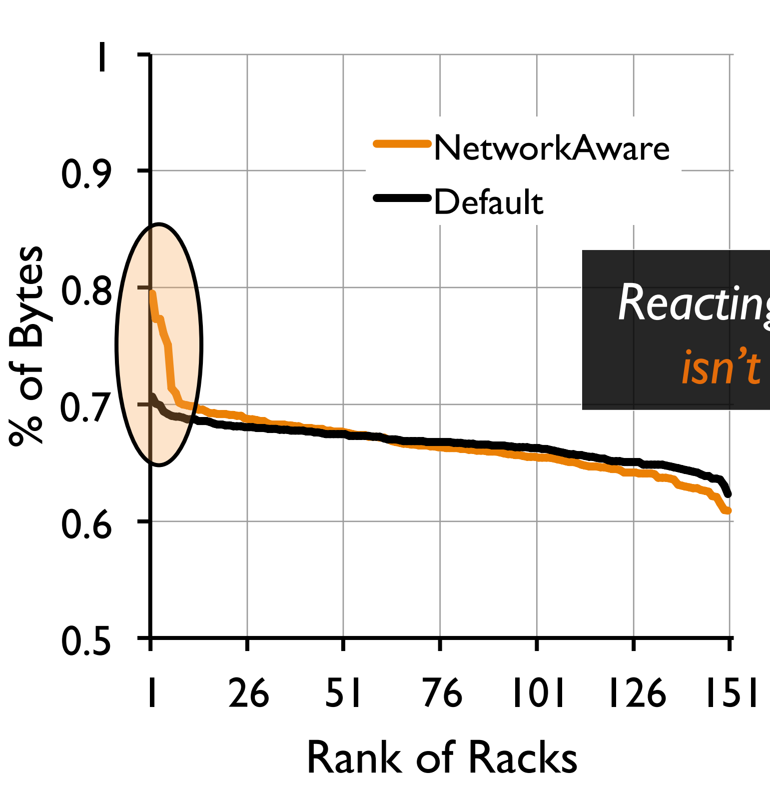
No clear correlation

- We do not use file characteristics while selecting destinations



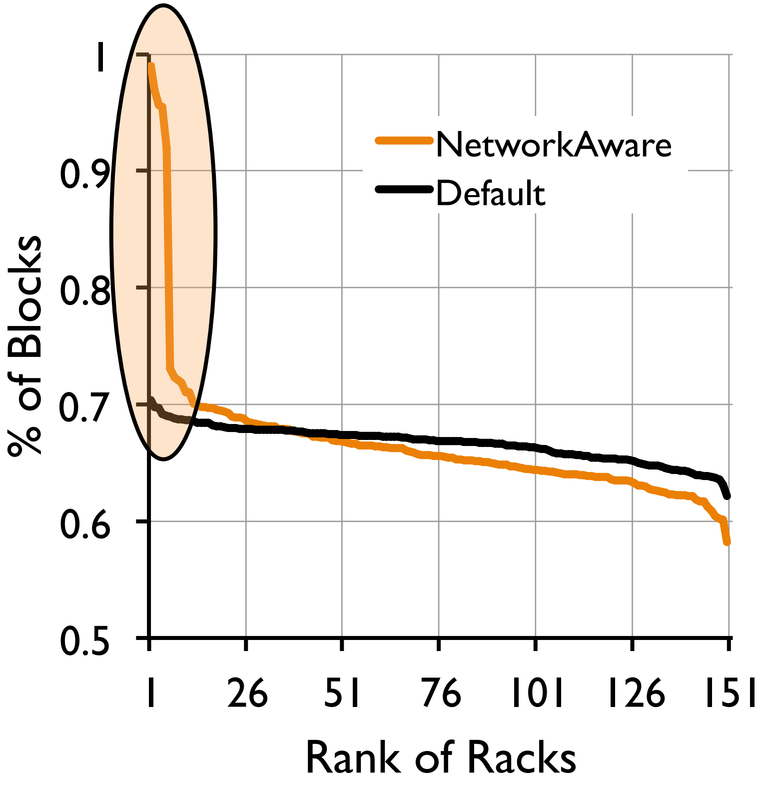
Balanced Storage [Simulation]

Byte Distribution



Reacting to the imbalance isn't always perfect!

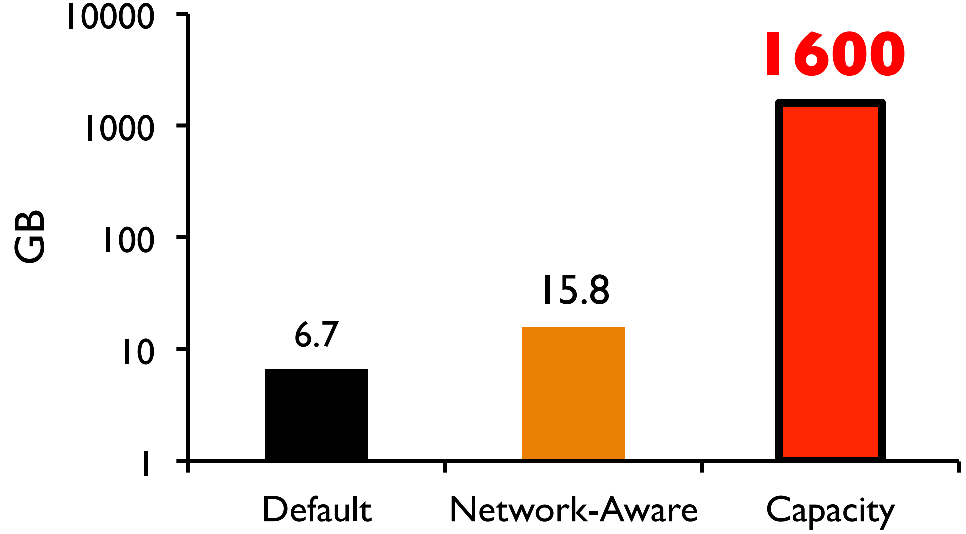
Block Distribution



Balanced Storage [EC2]

An hour-long 100-node EC2 experiment

- Wrote ~10TB of data
- Calculated standard deviation of disk usage across all machines



Imbalance is less than 1% of the storage capacity of each machine