Gaia

Application-Network Symbiosis in Big Data Analytics

Mosharaf Chowdhury

Big Data

The volume of data businesses want to *make sense of* is increasing

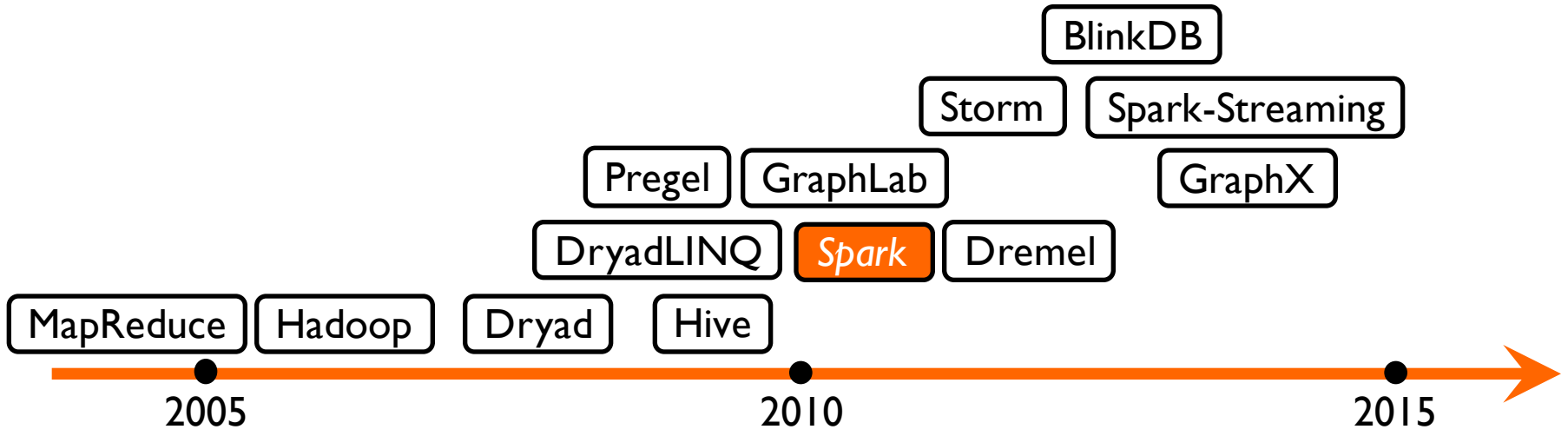
Increasing variety of sources

- Web, mobile, wearables, vehicles, scientific, …

Cheaper disks, SSDs, and memory

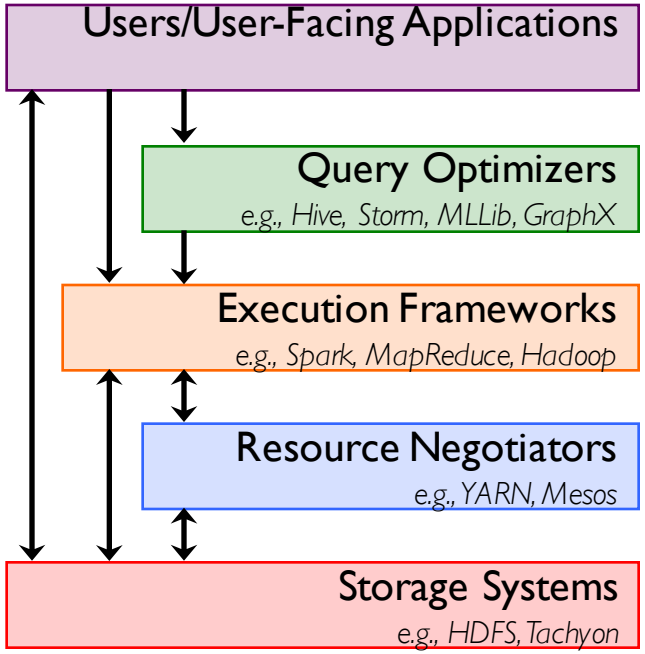
Stalling processor speeds

Big Datacenters for Massive Parallelism



1. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing, NSDI'2012.

Communication in Big Data Stacks

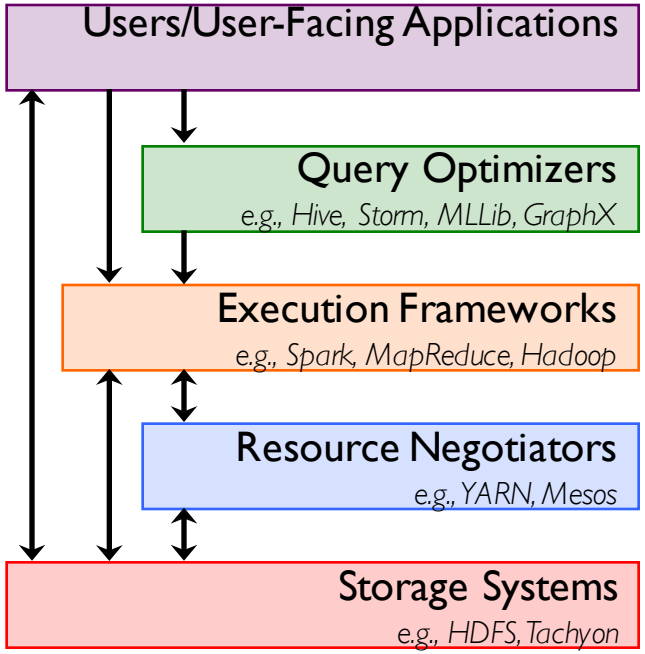


Hadoop Stack/
Spark Stack

#1 Application-Aware Networking

#2 Network-Aware Applications

Communication in Big Data Stacks



Hadoop Stack/
Spark Stack

#1 Application-Aware Networking

#2 Network-Aware Applications

Data-Parallel Applications

Multi-stage dataflow

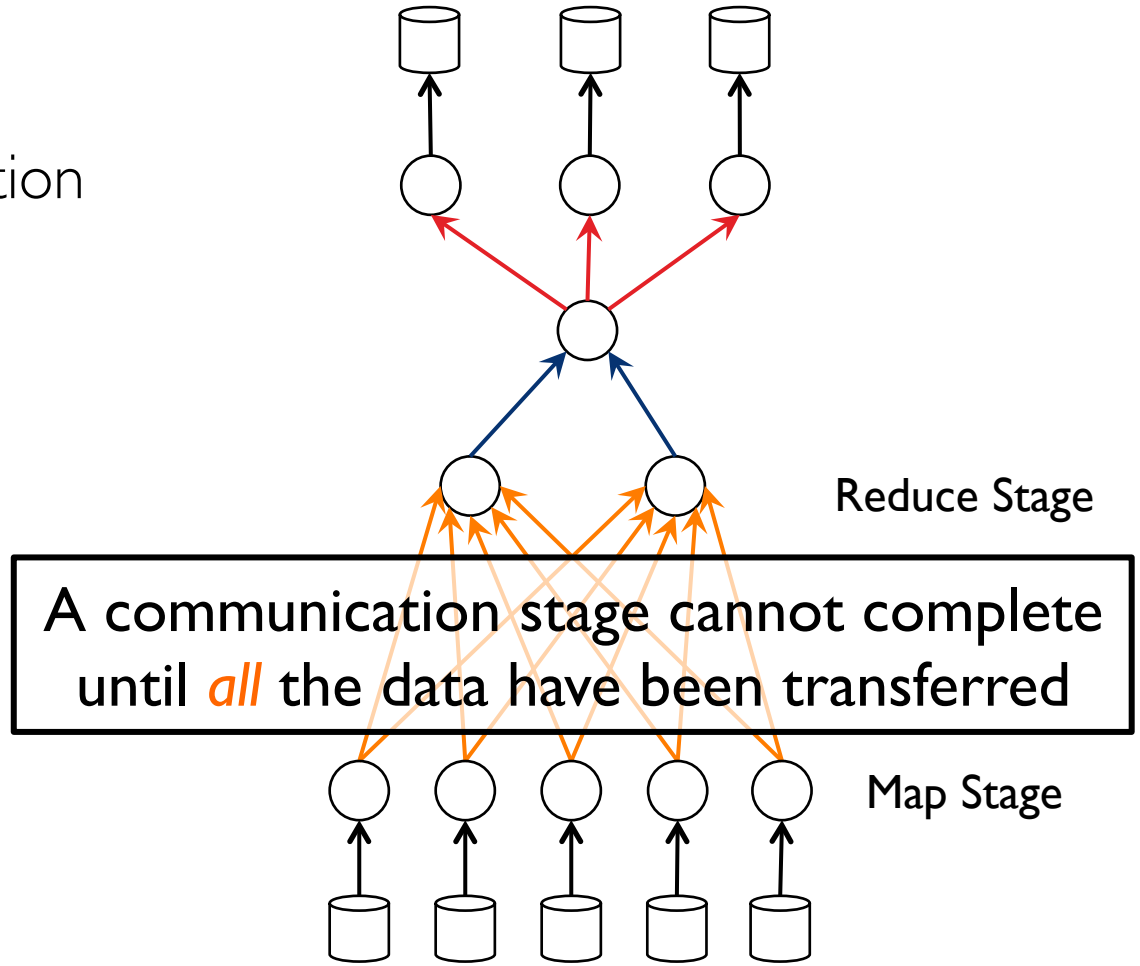
- Computation interleaved with communication

Computation Stage (e.g., Map, Reduce)

- Distributed across many machines
- Tasks run in parallel

Communication Stage (e.g., Shuffle)

- Between successive computation stages



Communication is Crucial

Performance

Facebook jobs spend ~**25%** of runtime on *average* in intermediate comm.[1]

As SSD-based and in-memory systems proliferate,
the network is likely to become the primary bottleneck

1. Based on a month-long trace with 320,000 jobs and 150 Million tasks, collected from a 3000-machine Facebook production MapReduce cluster.

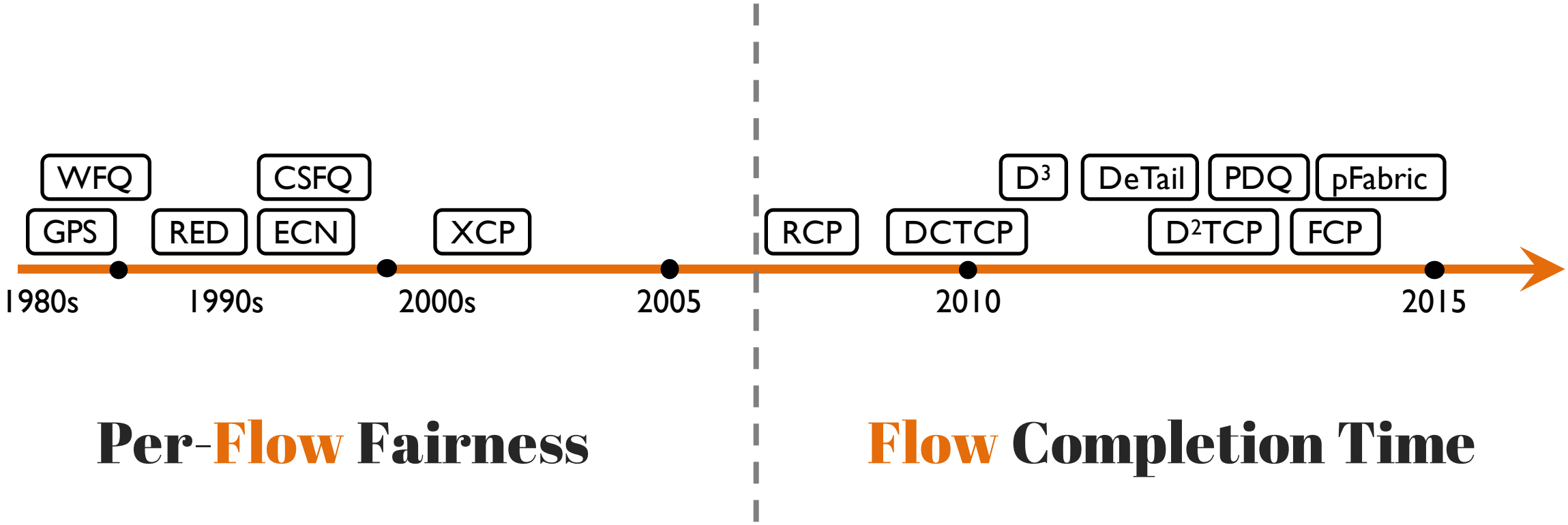
Flow

Transfers data from a source to a destination

Independent unit of allocation, sharing, load balancing, and/or prioritization

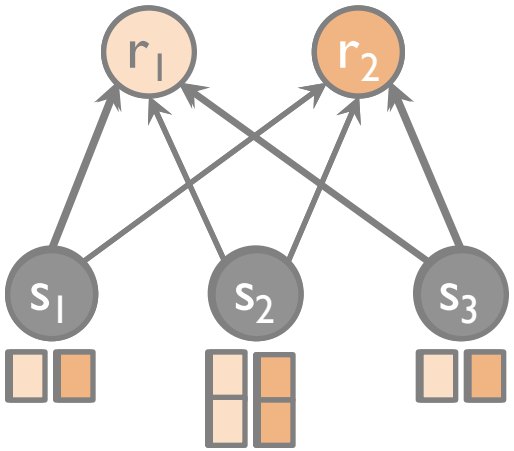
Faster Communication Stages: Traditional Networking Approach

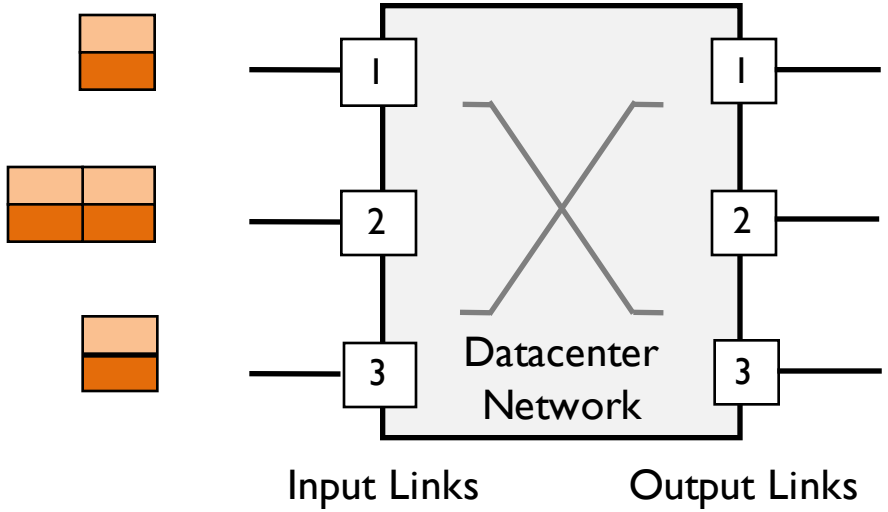
Existing Solutions



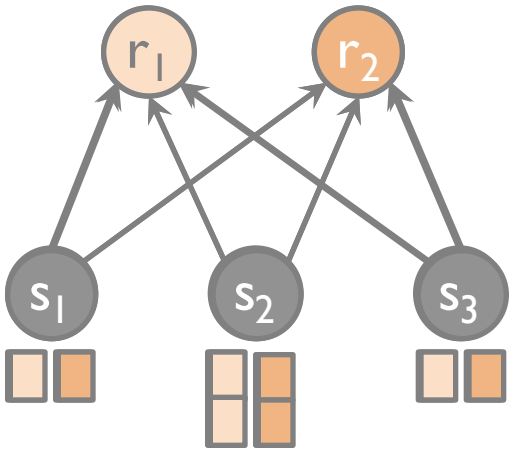
Independent flows cannot capture the collective communication behavior common in data-parallel applications

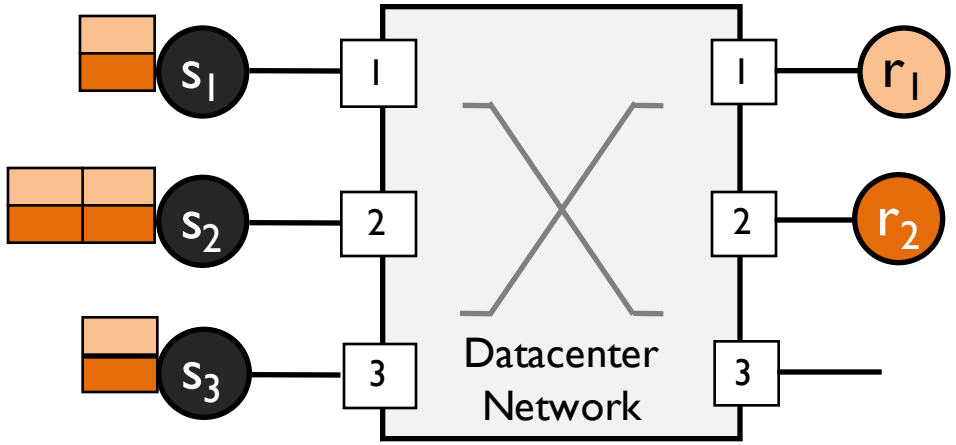
Why Do They Fall Short?



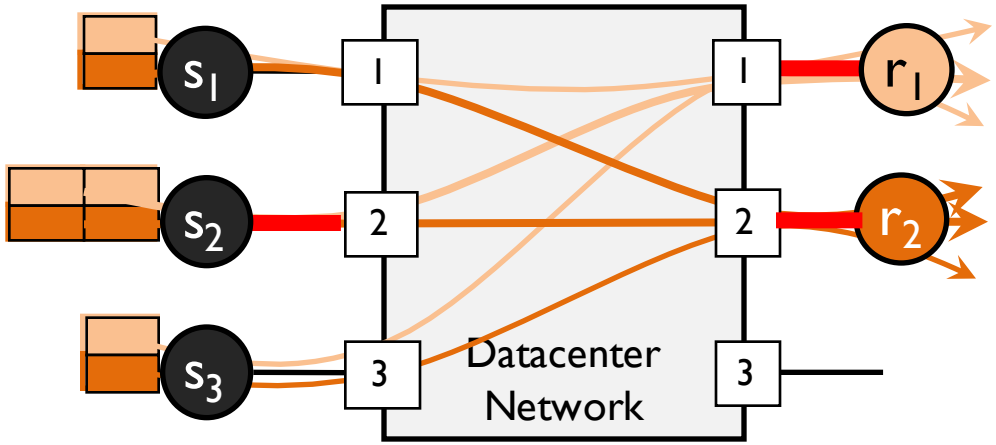


Why Do They Fall Short?

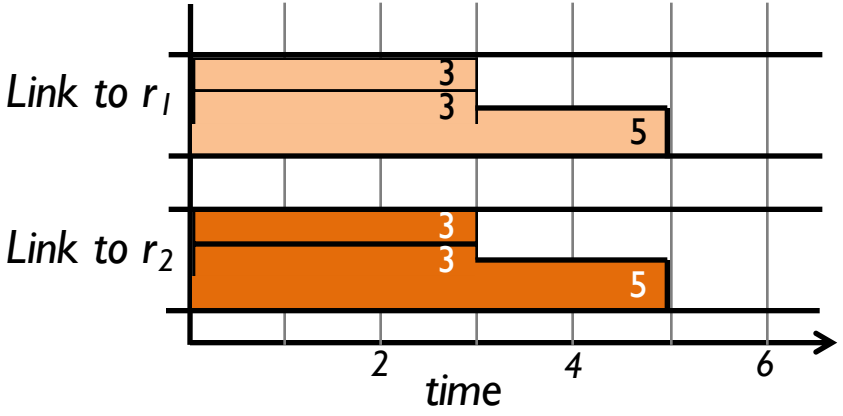




Why Do They Fall Short?



Per-Flow Fair Sharing

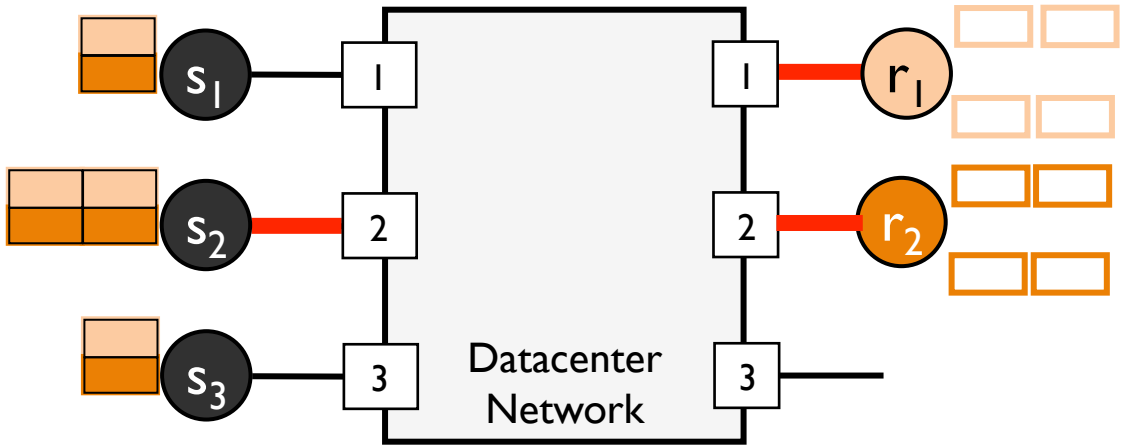


Shuffle Completion Time = **5**

Avg. Flow Completion Time = **3.66**

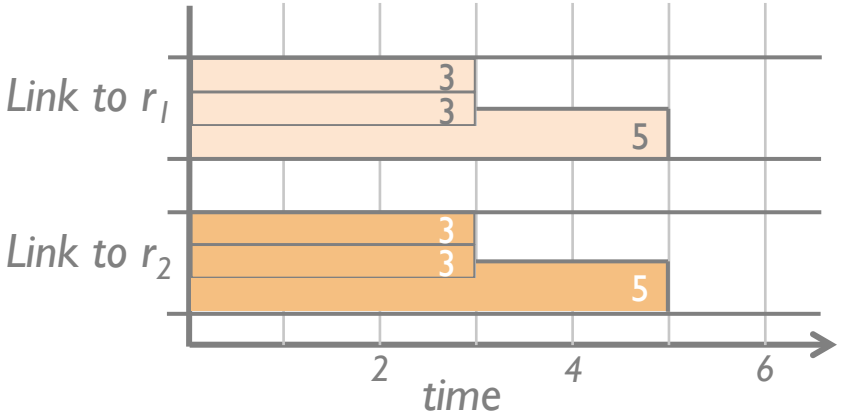
Solutions focusing on flow completion time cannot further decrease the shuffle completion time

Improve Application-Level Performance[1]



Slow down faster flows to *accelerate* slower flows

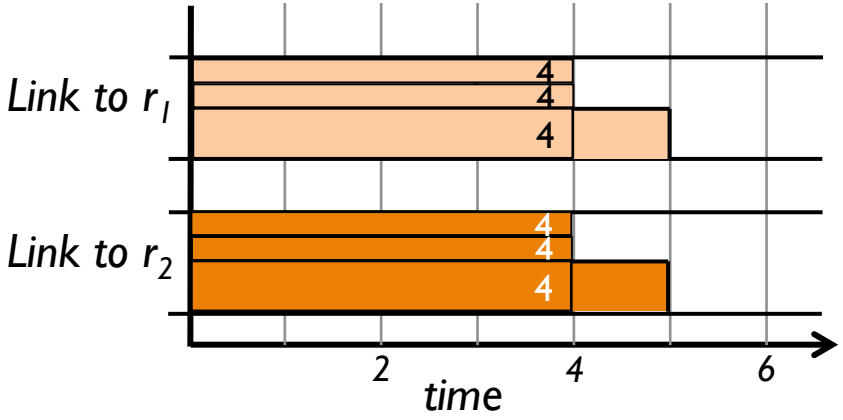
Per-Flow Fair Sharing



Shuffle Completion Time = **5**

Avg. Flow Completion Time = **3.66**

Data-Proportional Allocation



Shuffle Completion Time = **4**

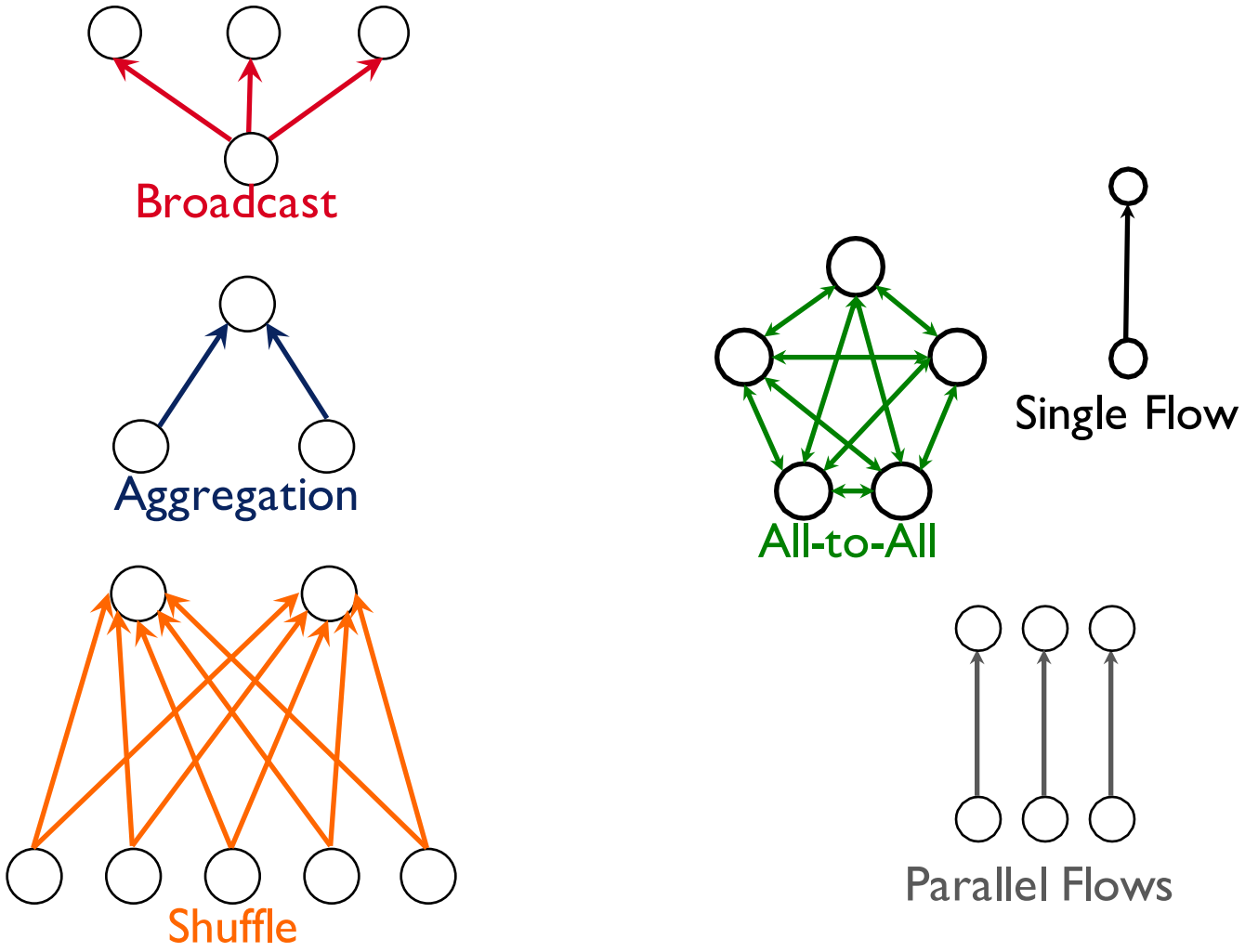
Avg. Flow Completion Time = **4**

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

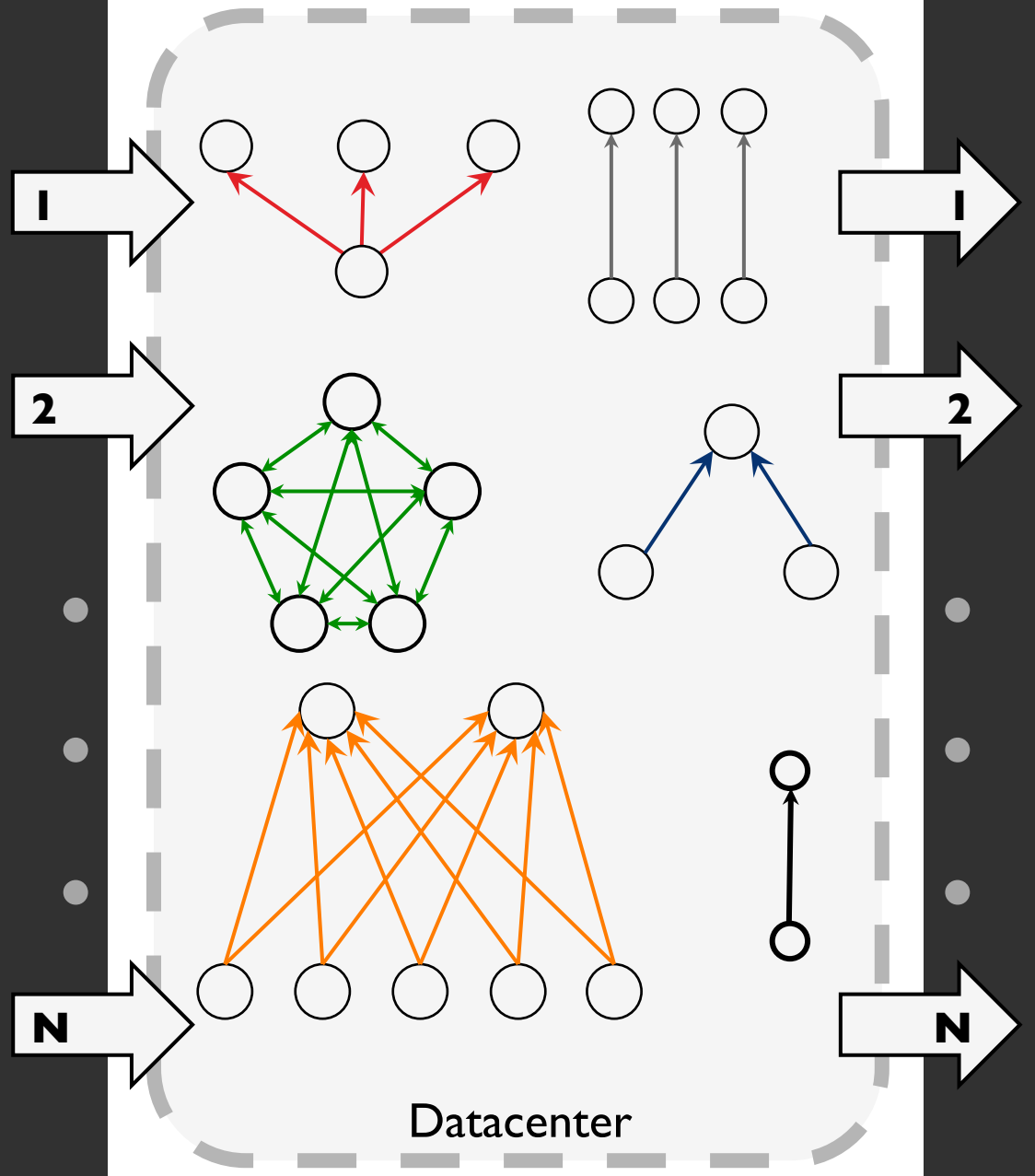
Coflow

Communication abstraction for data-parallel applications to express their performance goals

1. The size of each flow,
2. The total number of flows, and
3. The endpoints of individual flows.


Broadcast
Aggregation
Shuffle
All-to-All
Single Flow
Parallel Flows

How to schedule coflows online ...



#1 ... for faster completion of coflows?

#2 ... to meet more deadlines?

#3 ... for fair allocation of the network?

Varys[1], Aalo[2] & HUG[3]

1. Coflow Scheduler
2. Global Coordination
3. The Coflow API

Faster, application-aware data transfers throughout the network

Consistent calculation and enforcement of scheduler decisions

Decouples network optimizations from applications, relieving developers and end users

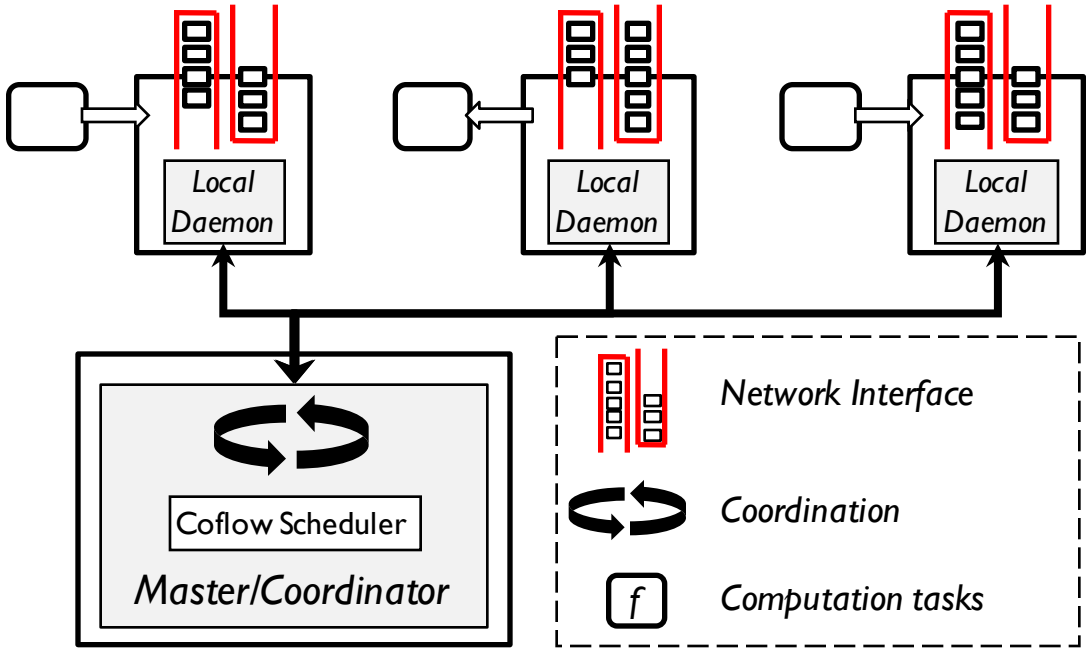
1. Efficient Coflow Scheduling with Varys, SIGCOMM'2014.
2. Efficient Coflow Scheduling Without Prior Knowledge, SIGCOMM'2015.
3. HUG: Multi-Resource Fairness for Correlated and Elastic Demands, NSDI'2016.

Coflow-Based Architecture

Centralized master-slave architecture

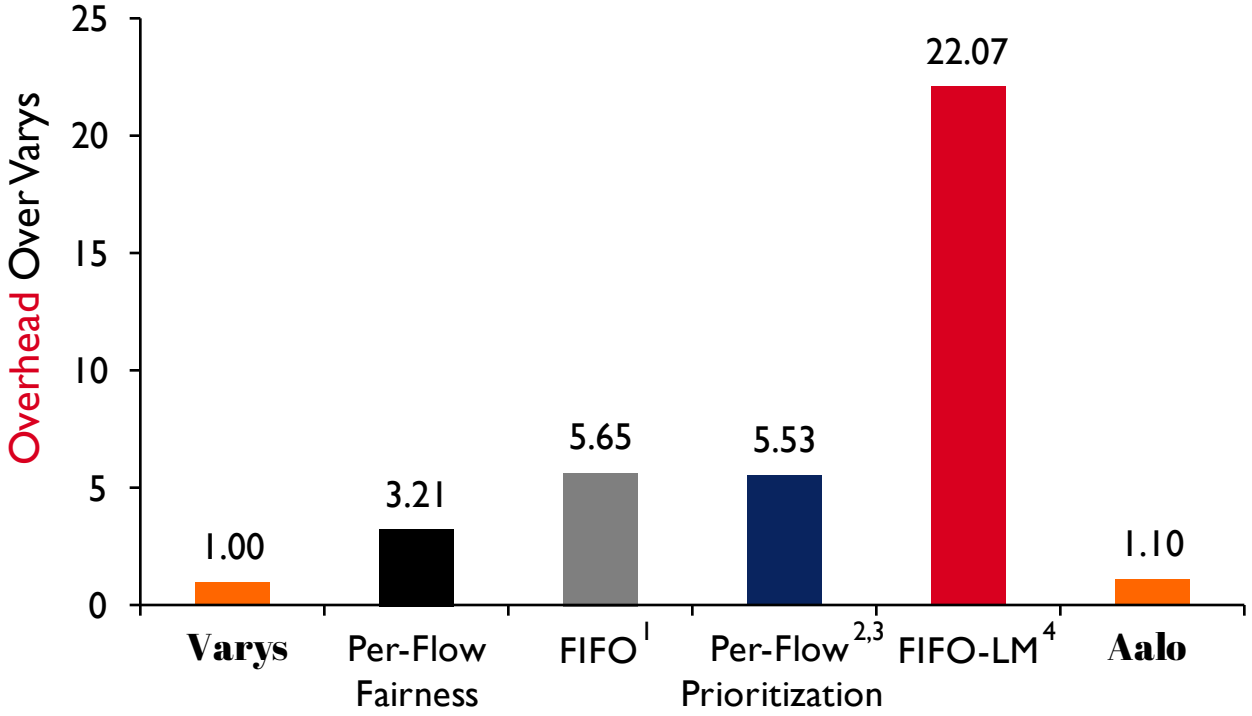
- Applications use a client library to communicate with the master

Actual *timing* and *rates are* determined by the coflow scheduler



1. Download from http://github.com/coflow

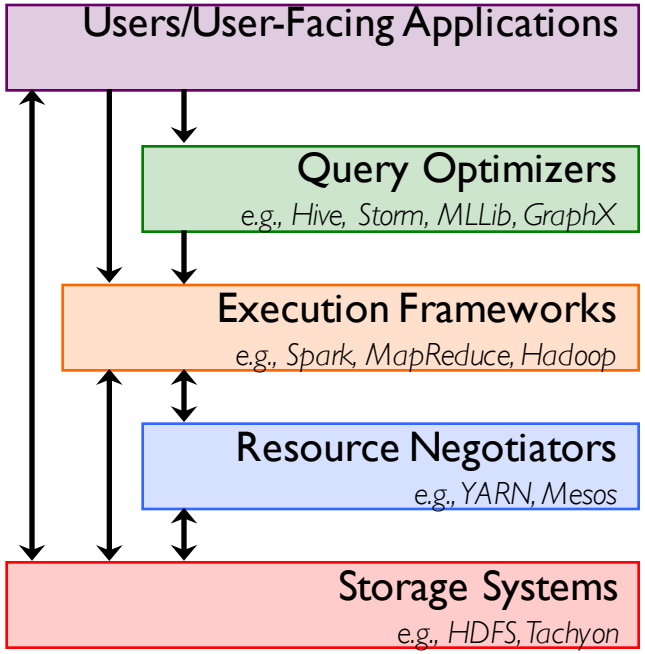
Benefits of Using **Coflows**



Lower is Better

1. *Managing Data Transfers in Computer Clusters with Orchestra, SIGCOMM'2011*
2. *Finishing Flows Quickly with Preemptive Scheduling, SIGCOMM'2012*
3. *pFabric: Minimal Near-Optimal Datacenter Transport, SIGCOMM'2013*
4. *Decentralized Task-Aware Scheduling for Data Center Networks, SIGCOMM'2014*

Communication in Big Data Stacks



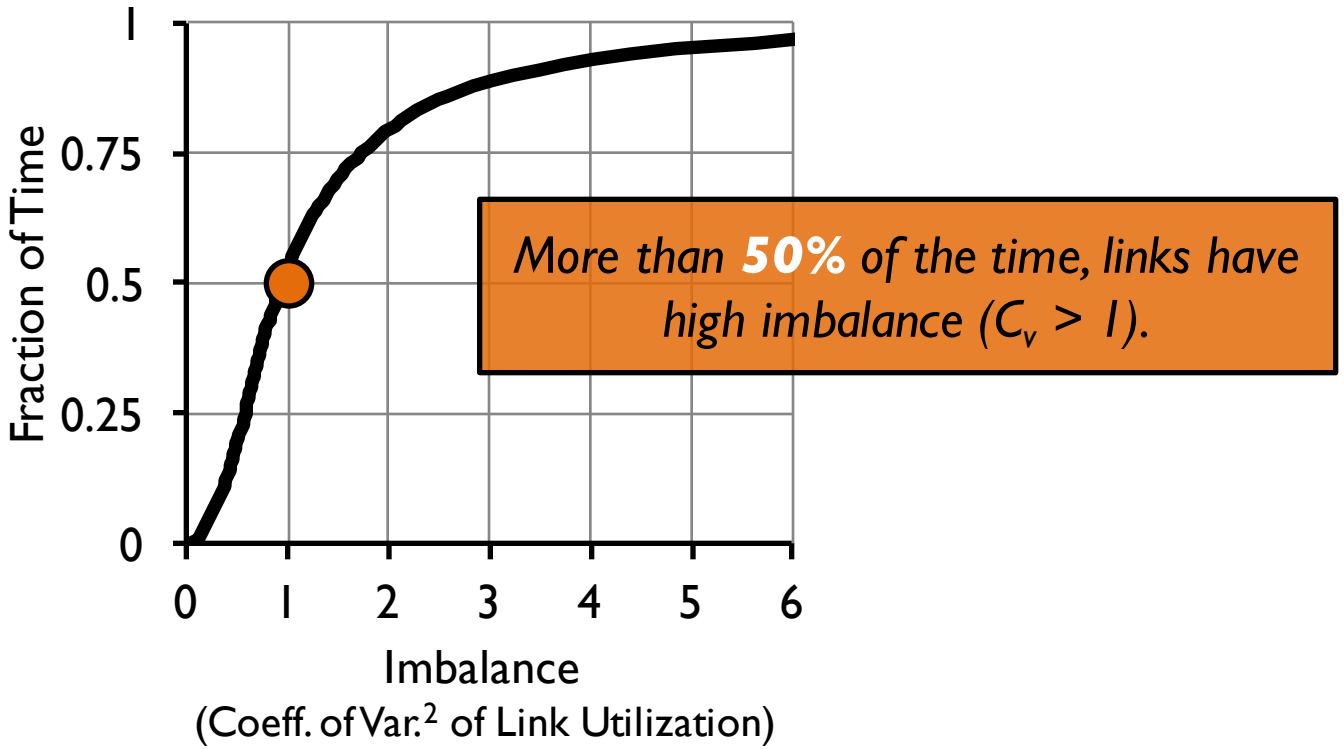
Hadoop Stack/
Spark Stack

#1 Application-Aware Networking

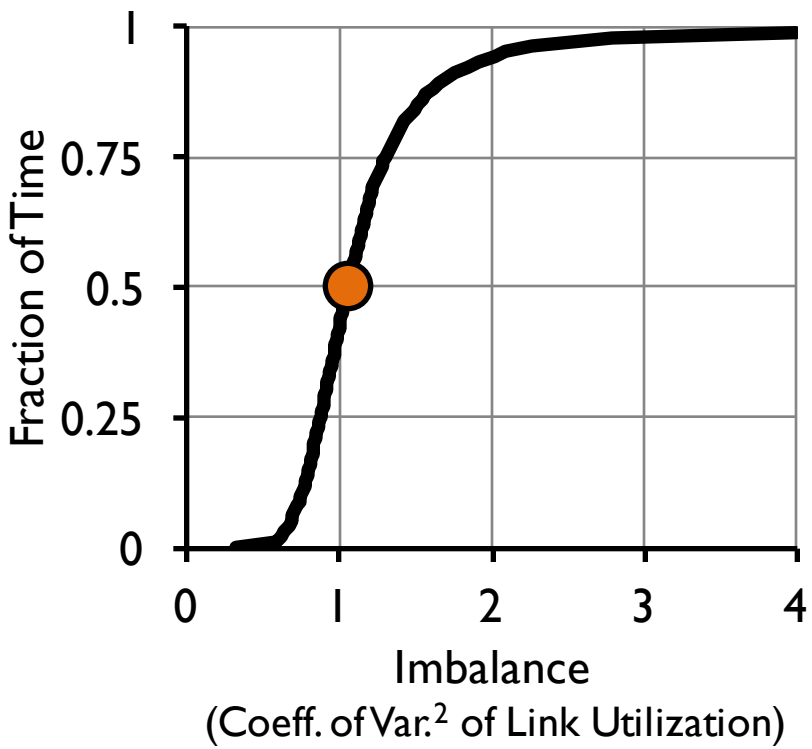
#2 Network-Aware Applications

Network Usage is Imbalanced[1]

Facebook



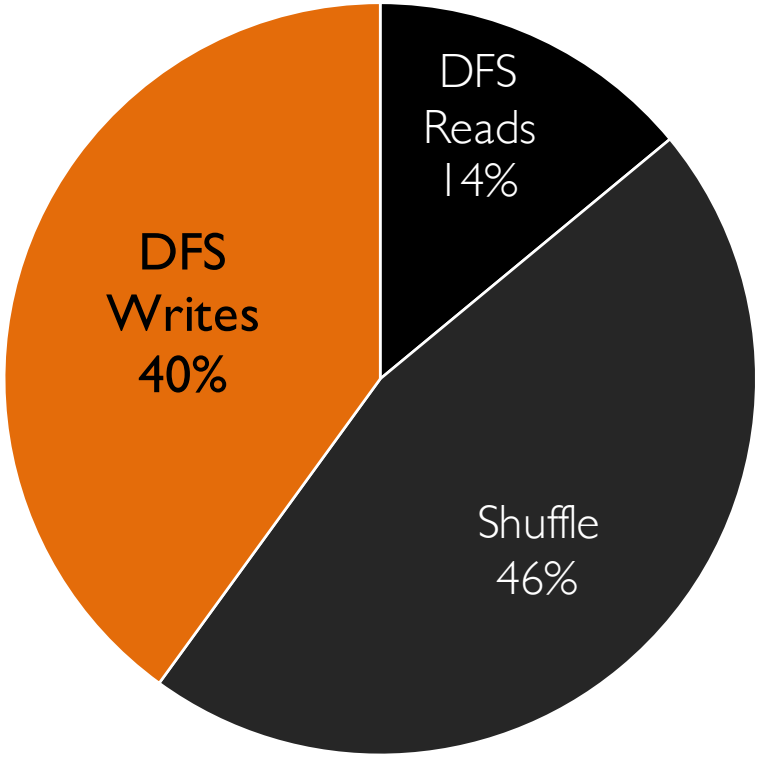
Bing



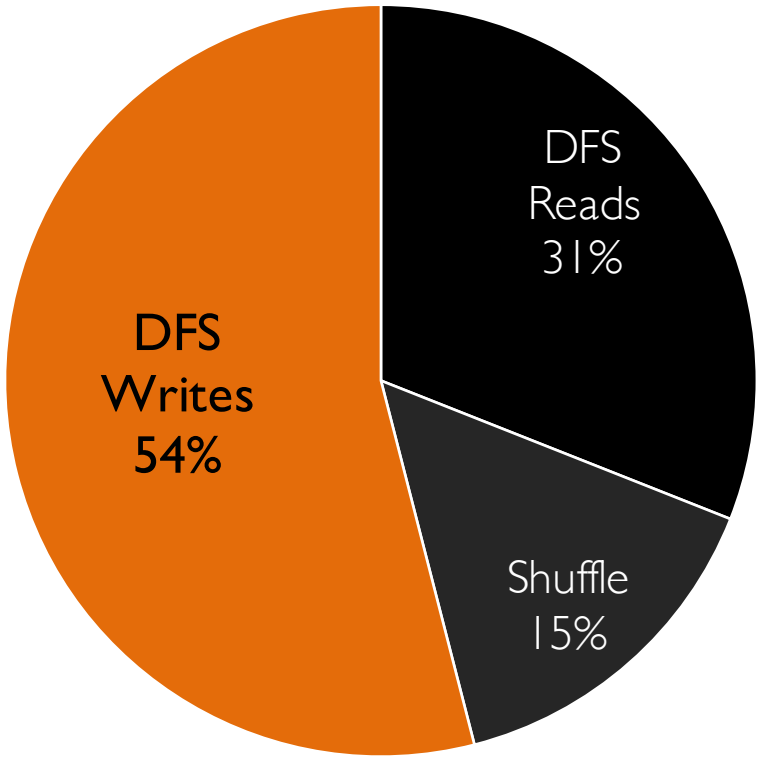
1. Imbalance considering all cross-rack bytes. Calculated in 10s bins.
2. Coefficient of variation, C_v = (stdev/mean).

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

Facebook



Bing

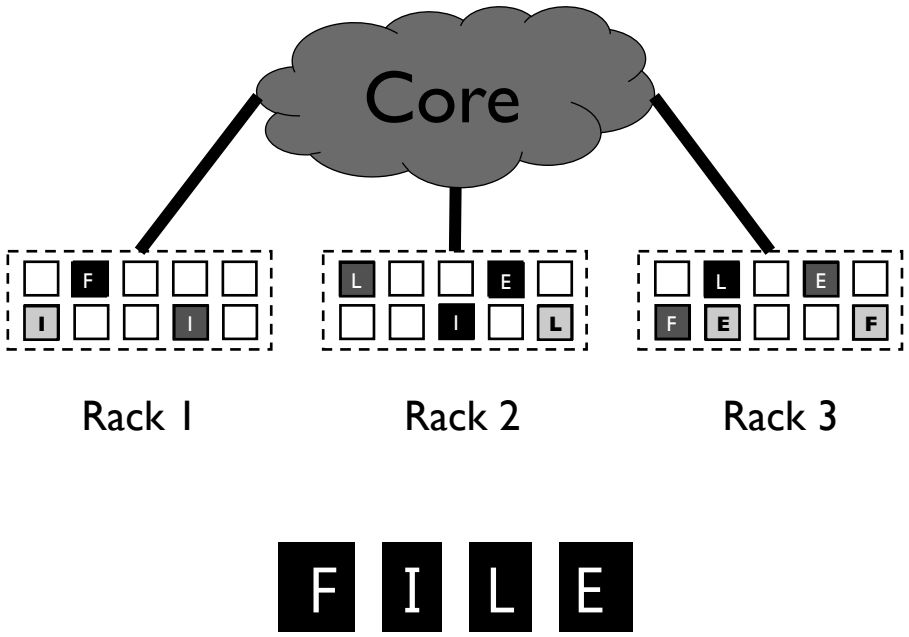


Write Sources

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

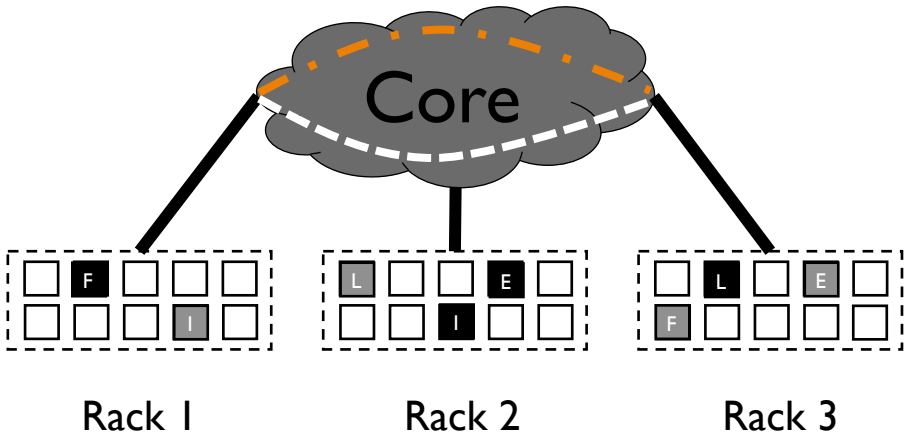
Files are divided into blocks

- 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

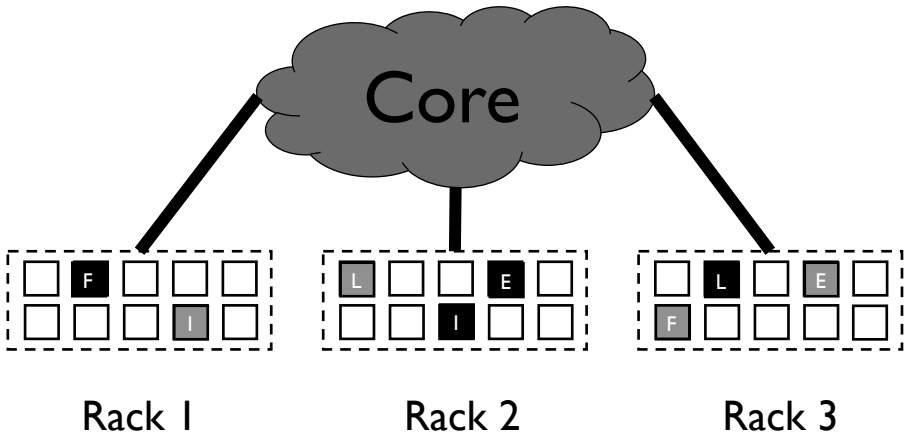
Paths
Rates

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

Flexible

- Sources
- Destinations ✔

Flexible

- Paths
- Rates

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

Sinbad[1]

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*

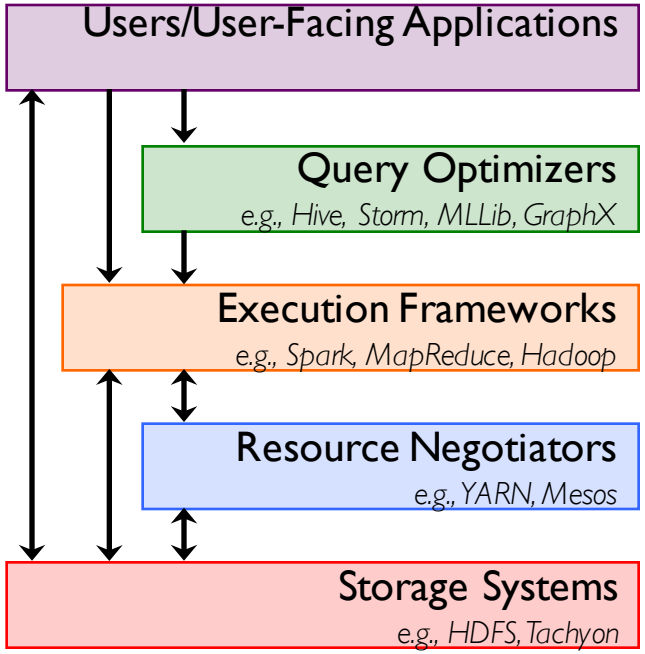
1. Leveraging Endpoint Flexibility in Data-Intensive Clusters, SIGCOMM'2013.

Faster Data Ingestion and Analytics

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

^ In-memory storage

Communication in Big Data Stacks



Hadoop Stack/
Spark Stack

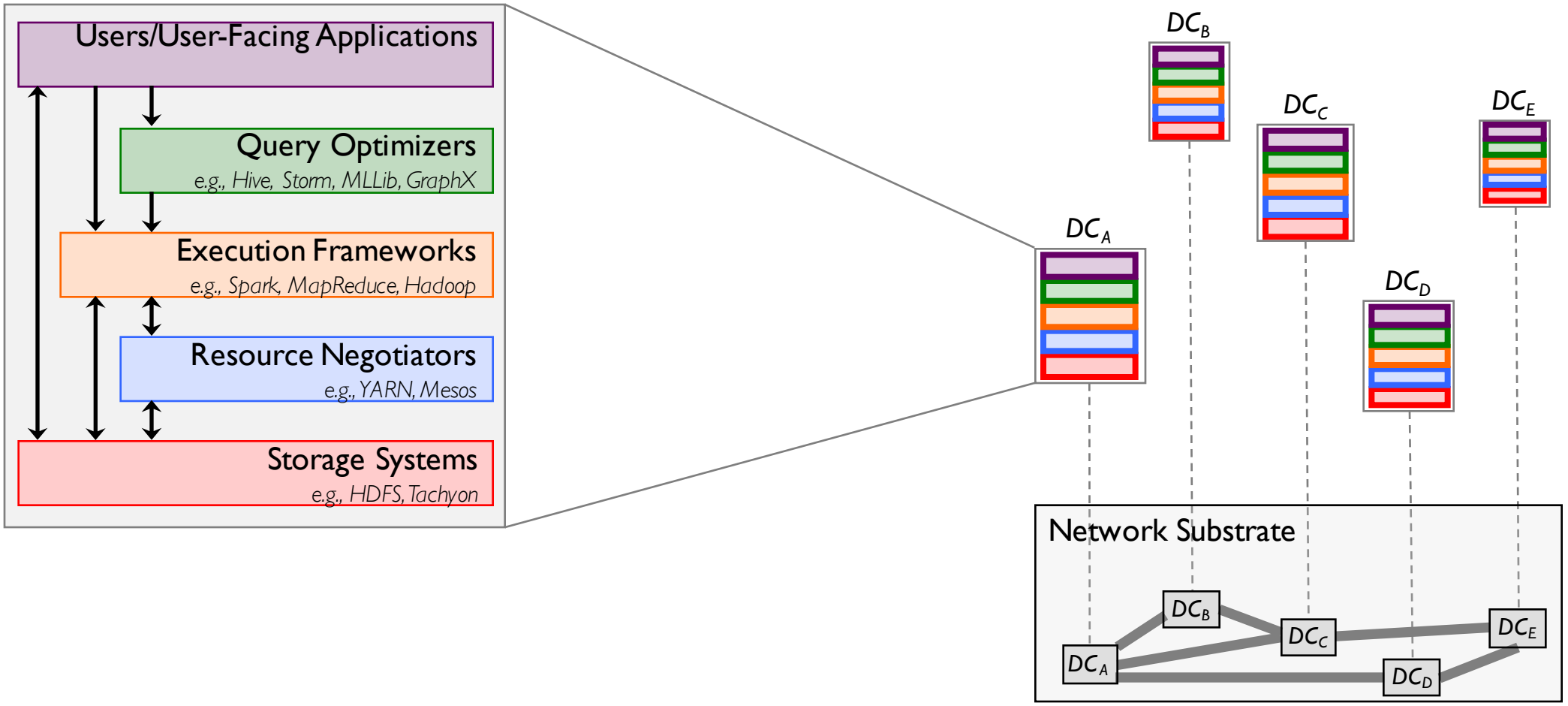
#1 Application-Aware Networking

#2 Network-Aware Applications

Geo-Distributed Clouds

1. Amazon EC2, Microsoft Azure, and Google Compute cloud regions as of April'2016.

Geo-Distributed Big Data Stacks



Network is King

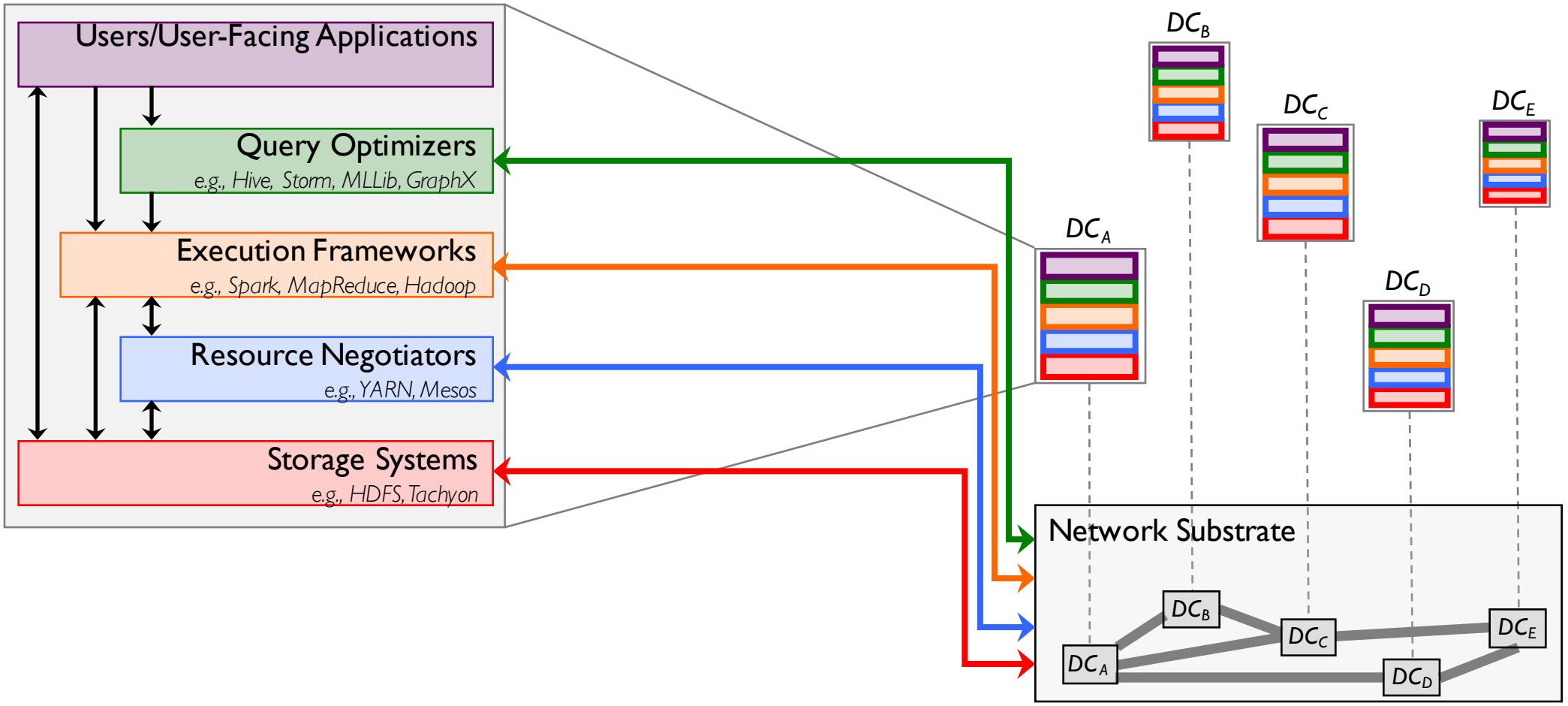
Bandwidth	*Low and expensive*
Latency	*High and variable*
Reliability	*Low and unpredictable*

Gaia

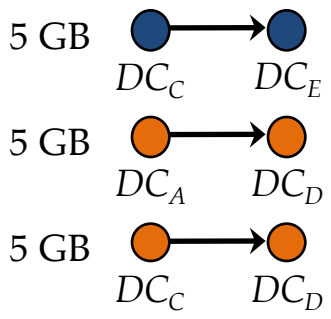
Enable tighter application-network symbiosis throughout big data stacks

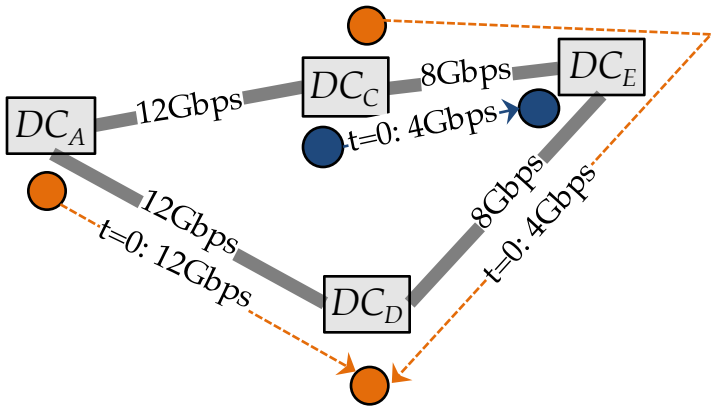
1. **Performance** — *Select faster paths and perform better load balancing*
2. **Cost** — *Minimize the $ cost of data transfers*
3. **Privacy** — *Enforce data residency constraints*

Geo-Distributed Big Data Stacks



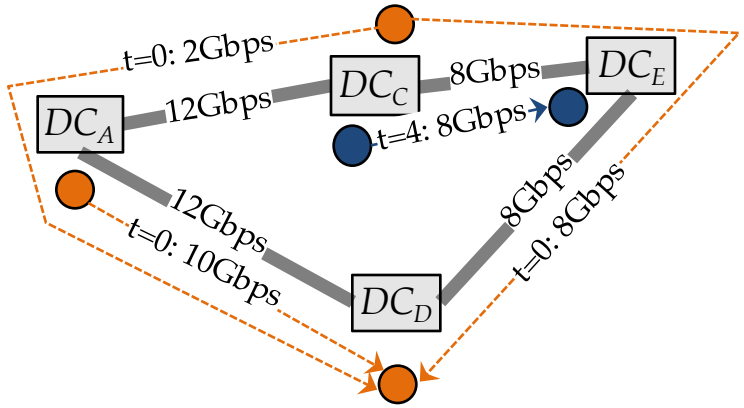
Geo-Distributed Coflows





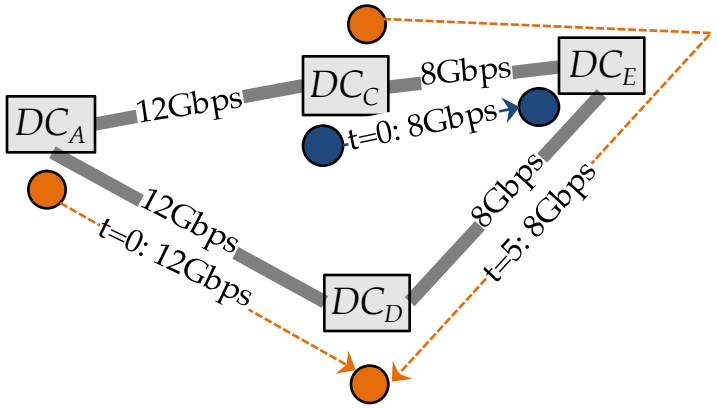
TCP

10 seconds



Geo-Distributed Coflow

4.5 seconds



Coflow

7.5 seconds

Gaia: Application-Network Symbiosis