EC/CS 528: Cloud Computing

Distributed Systems and Resource Management

Instructor: Alan Liu

BU Department of Electrical & Computer Engineering

BOSTON UNIVERSITY

Ceph: A Scalable, High-Performance Distributed File System

- Sage A. Weil, Scott A. Brandt, Ethan L. Miller, Darrell D. E. Long, and Carlos Maltzahn
- OSDI 2006

BOSTON UNIVERSITY

File System Metadata

Metadata

Owner: abc
Size: 18M
last-modified: 28 Dec, 2014
Permissions: read/write

Data blocks:
Address1
Address2
Address3
….

Ceph Key Components

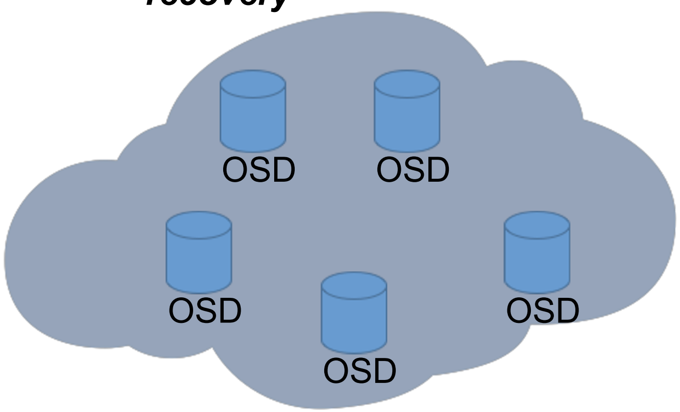
- ***A cluster of metadata servers which has dynamic load balancing feature and fault tolerant***

MDS Cluster

Client

- ***A program that runs in userspace and exposes file system interface to the host***

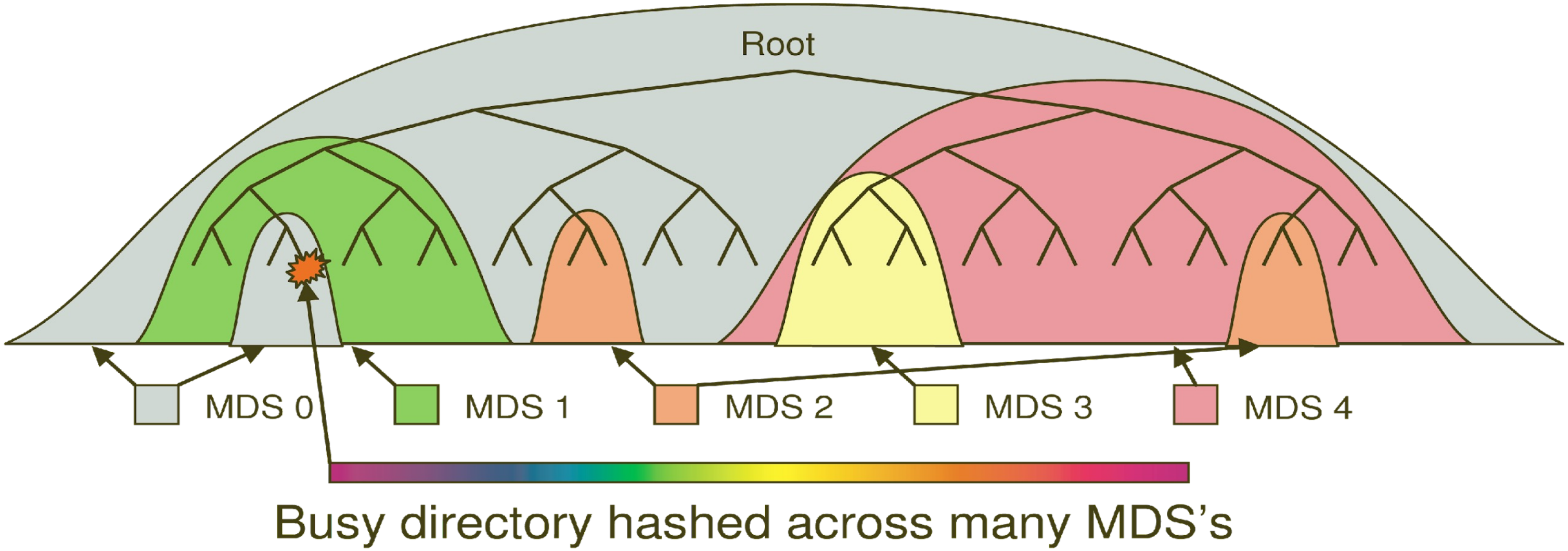
- ***Intelligent Storage device that handle data replication, failure detection and recovery***



Object Storage Cluster

Metadata Servers (MDS)

- Adaptively distributes cached metadata across a set of MDS w.r.t popularity

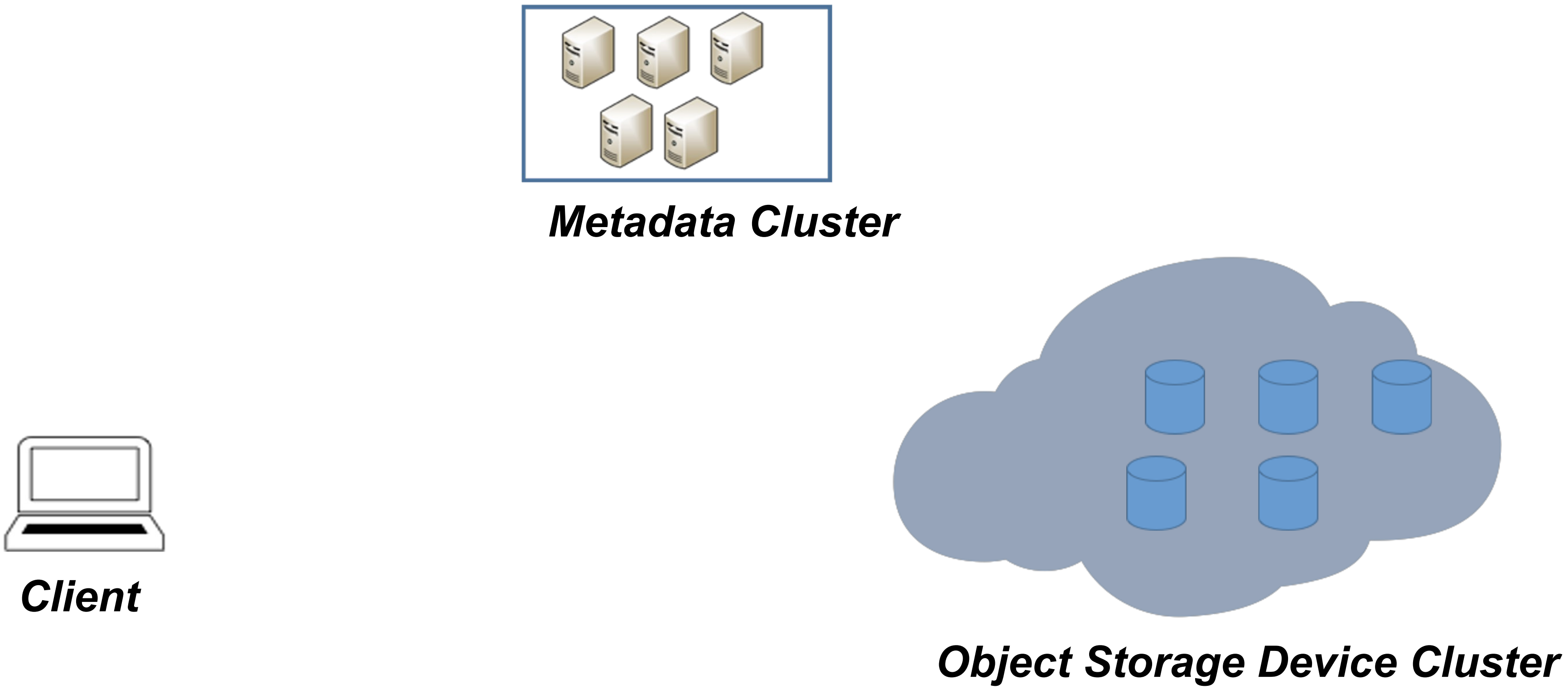


BOSTON UNIVERSITY

Object Storage Device (OSD)

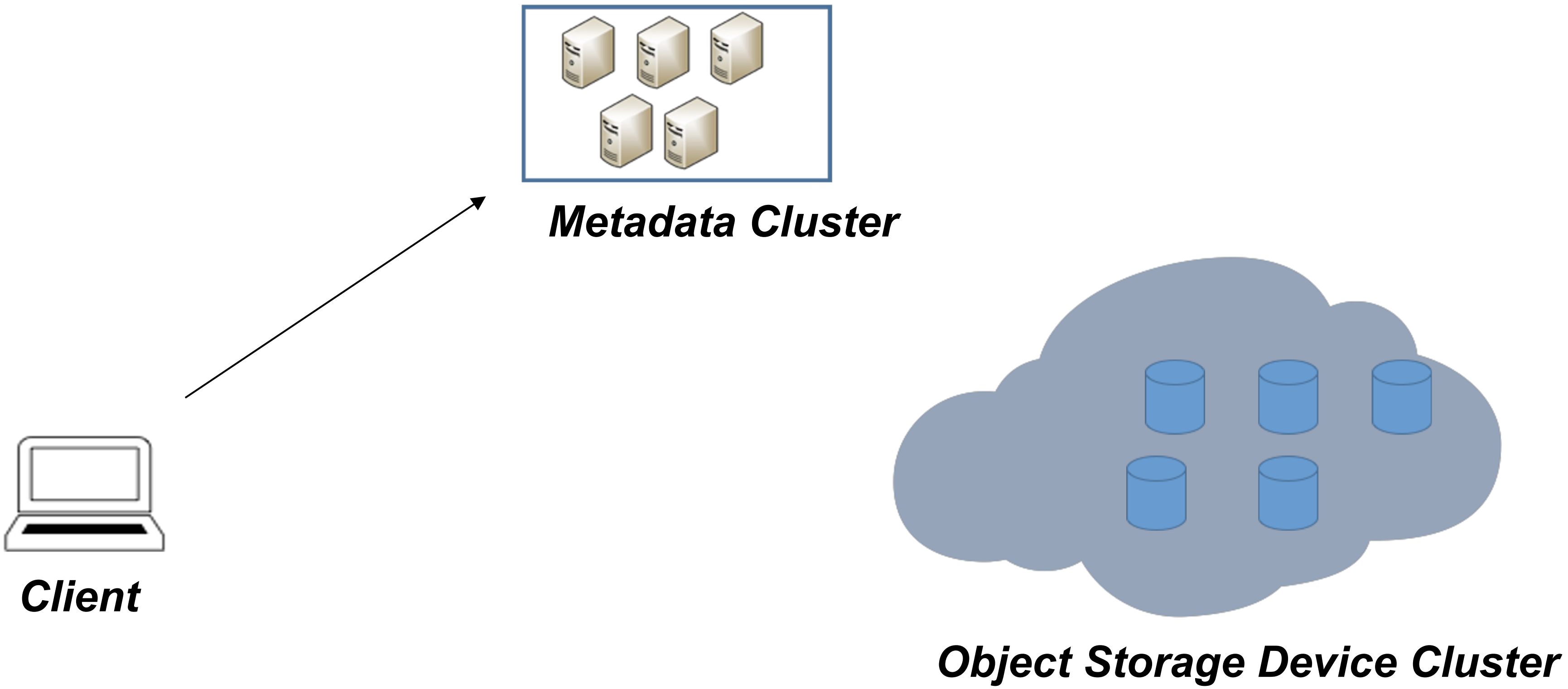
- Commodity server
  - with CPU, memory and storage
- Can make the replication and recovery decision
- Large blocks of data (e.g., 4MB objects)

Ceph Key Components



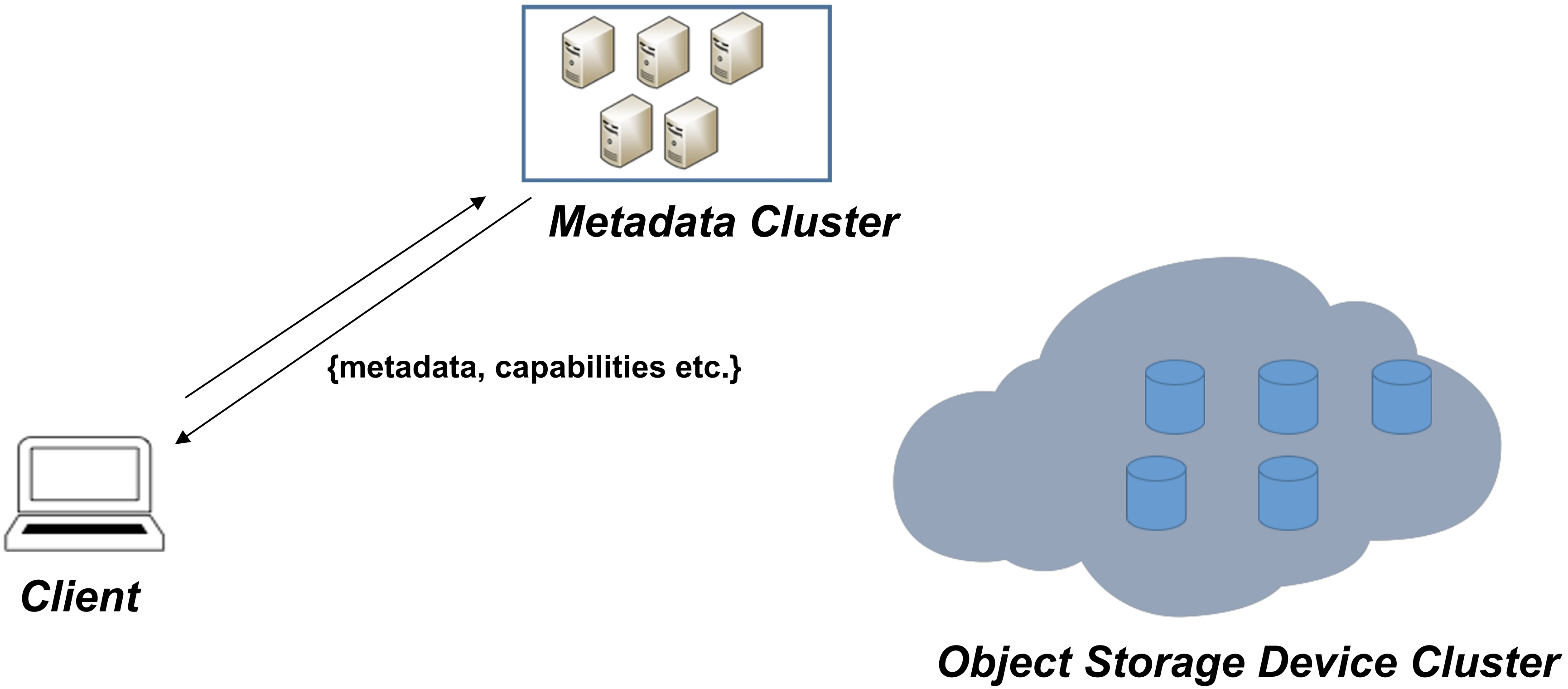
BOSTON UNIVERSITY

Ceph Key Components

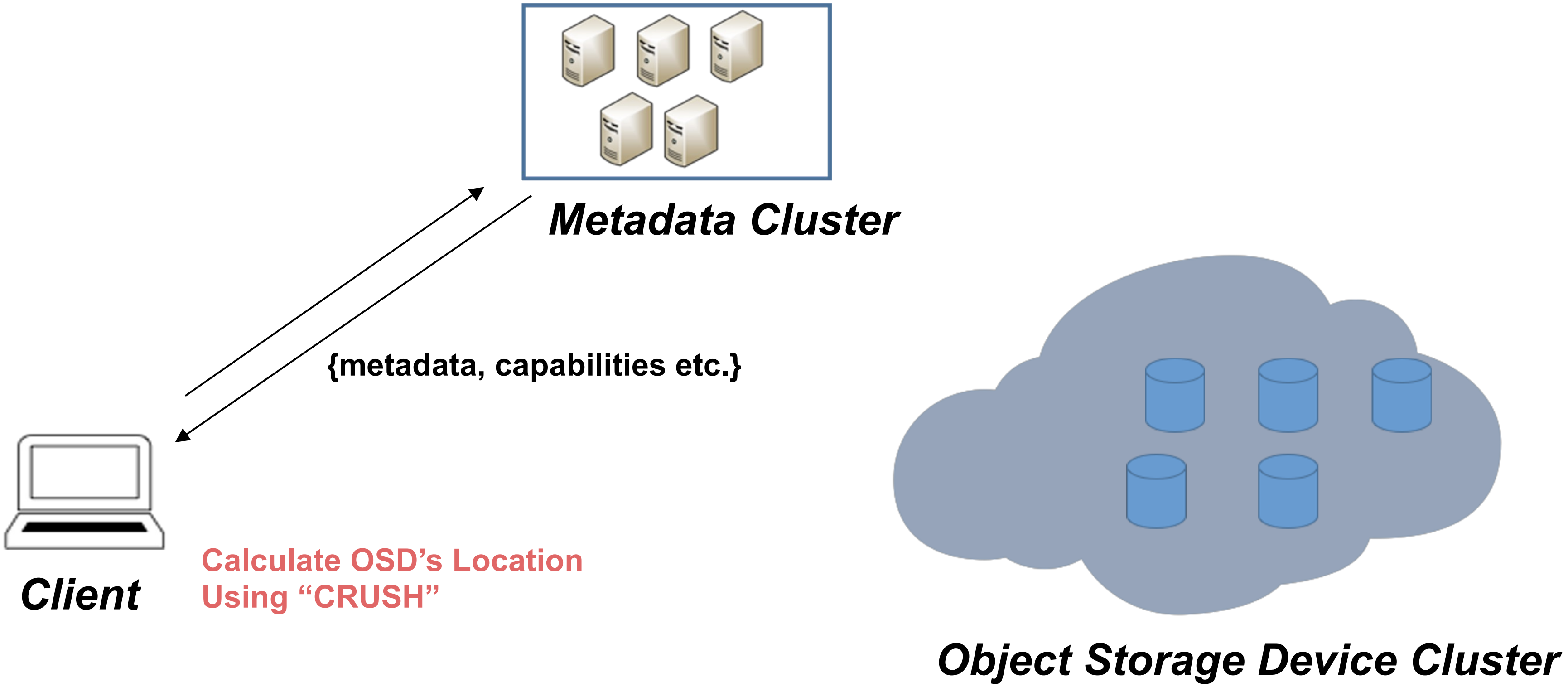


BOSTON UNIVERSITY

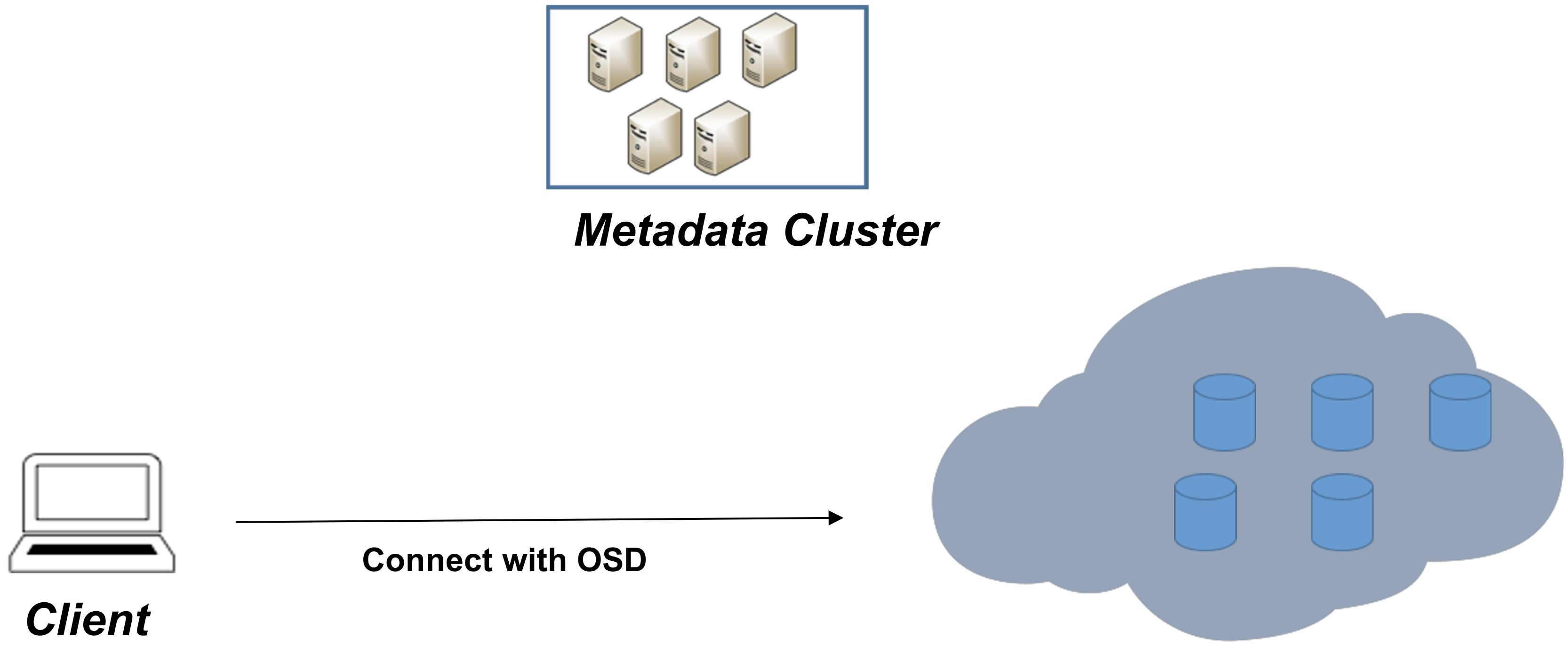
Ceph Key Components



Ceph Key Components



Ceph Key Components



BOSTON UNIVERSITY

Placement

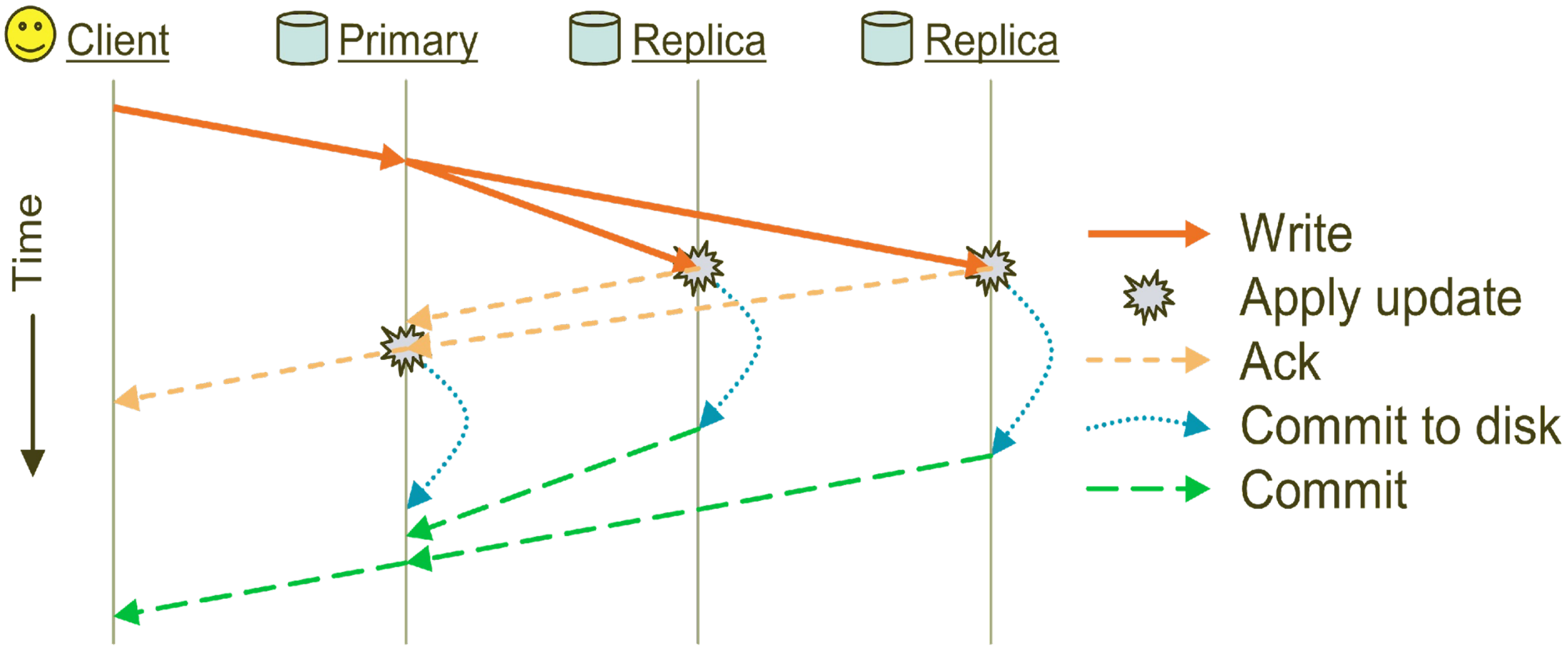
- Two levels of hashing:
  a. object ID -> placement group (PG)
  b. PG -> OSD
     - CRUSH: consistent hash that minimizes movement

- Client requires hash algorithm, PG, OSD cluster map
  – a compact, hierarchical description of the devices comprising the storage cluster.

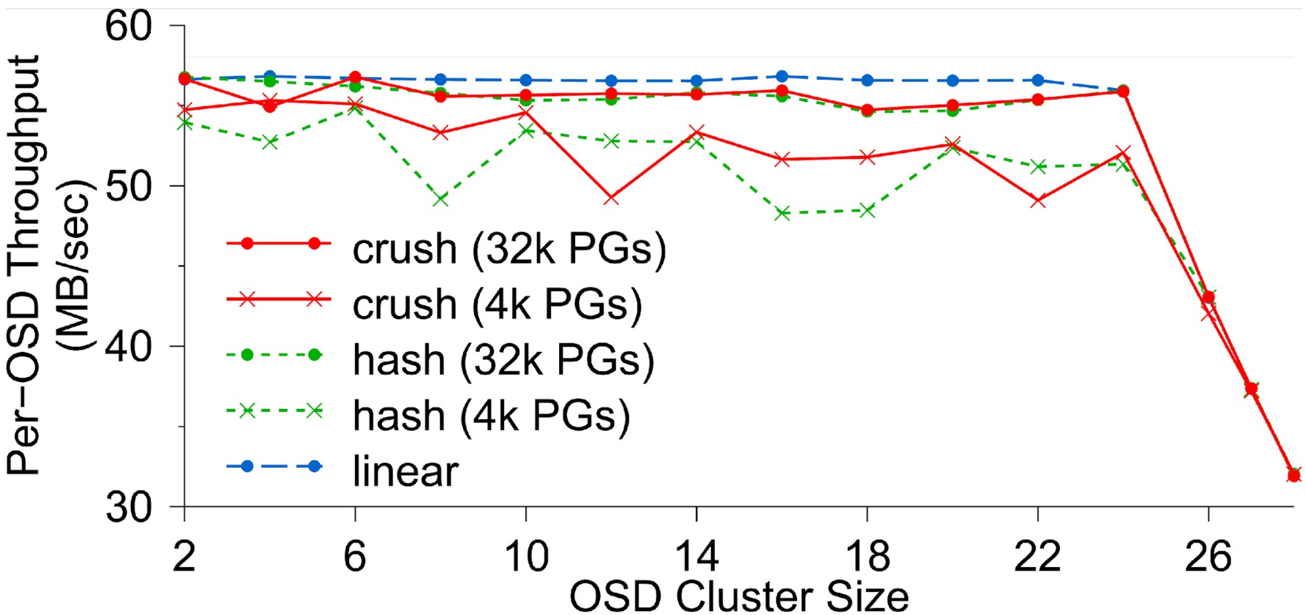
- Around 100 PG per OSD; when an OSD fails, different OSDs take over all the PG

BOSTON UNIVERSITY

Replication



Ceph Scalability



- ❑ Achieves nearly linear scaling up to 24 OSDs
- ❑ CRUSH and hash performance improves when more PGs lower variance in OSD utilization
- ❑ Linear striping & hash function fail to cope with device failures or other OSD cluster changes

BOSTON UNIVERSITY

Ceph - Conclusion

- Demonstrated key value of consistent hashing for locating data
- Only used meta-data for implementing traditional file system semantics
- Achieved excellent performance and scalability

Ceph - Conclusion

- From a PhD thesis to $175 Million company (Inktank) purchased by Red Hat
  - part of $34B acquisition by IBM
  - displaced Gluster...
- The file system was irrelevant; key value:
  - volume storage RBD
  - Object Storage with RGW
- Today Dominant file system in private clouds

BOSTON UNIVERSITY

Mesos

- A Platform for Fine-Grained Resource Sharing in the Data Center

Benjamin Hindman, Andy Konwinski, **Matei Zaharia**,
Ali Ghodsi, Anthony Joseph, Randy Katz, Scott Shenker, Ion Stoica

University of California, Berkeley

amplab

BOSTON UNIVERSITY

Background

- Rapid innovation in cluster computing frameworks

hadoop MapReduce

Google Pregel

Pig

RAILS

CIEL

S4 distributed stream computing platform

APACHE Spark

Google Percolator

MPI-2

BOSTON UNIVERSITY

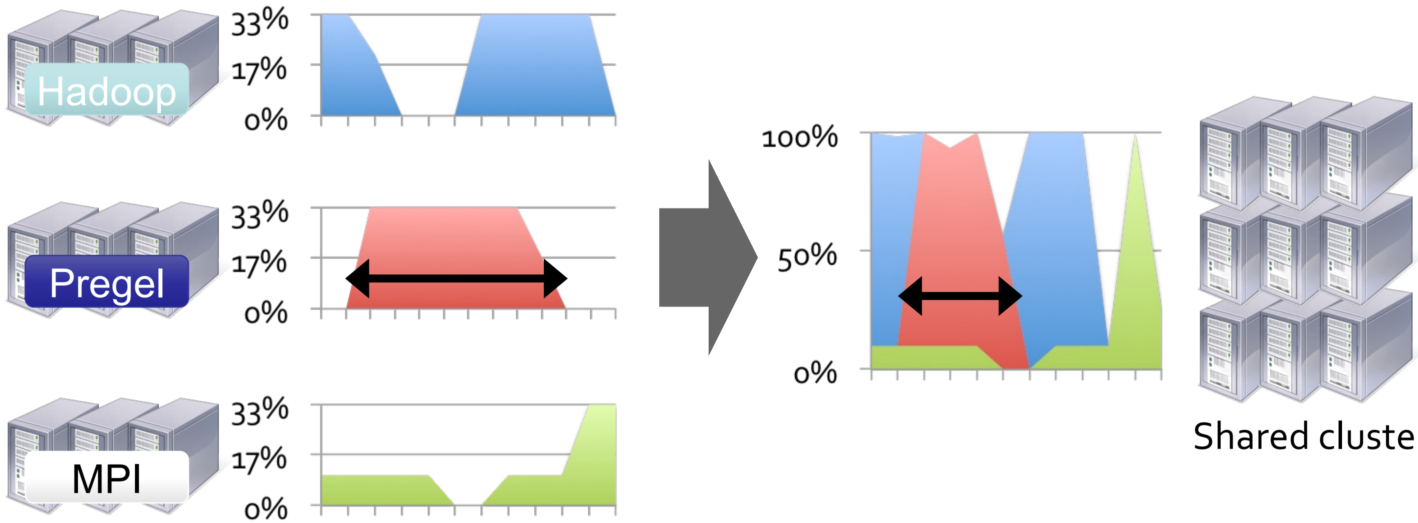
Problem

- Rapid innovation in cluster computing frameworks
- **No single framework optimal for all applications**
- Want to run multiple frameworks in a single cluster
  - …to *maximize utilization*
  - …to *share data* between frameworks

Where We Want to Go?

Today: static partitioning

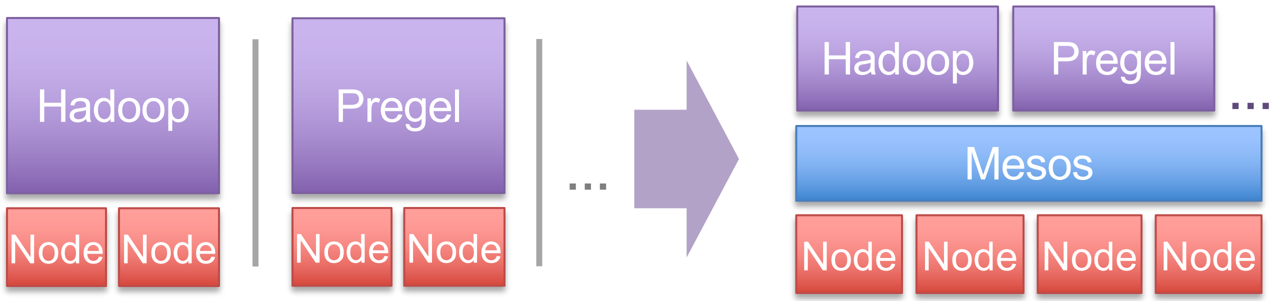
Mesos: dynamic sharing



BOSTON UNIVERSITY

Solution

- Mesos is a common resource sharing layer over which diverse frameworks can run



Other benefits of Mesos

- Run multiple instances of the *same* framework
  - Isolate production and experimental jobs
  - Run multiple versions of the framework concurrently
- Build *specialized frameworks* targeting particular problem domains
  - Better performance than general-purpose abstractions

BOSTON UNIVERSITY

Outline

- Mesos Goals and Architecture
- Implementation
- Results
- Related Work

What are the key goals of Mesos?

Mesos Goals

- **High utilization** of resources
- **Support diverse frameworks** (current & future)
- **Scalability** to 10,000's of nodes
- **Reliability** in face of failures

Resulting design: Small microkernel-like core that pushes scheduling logic to frameworks

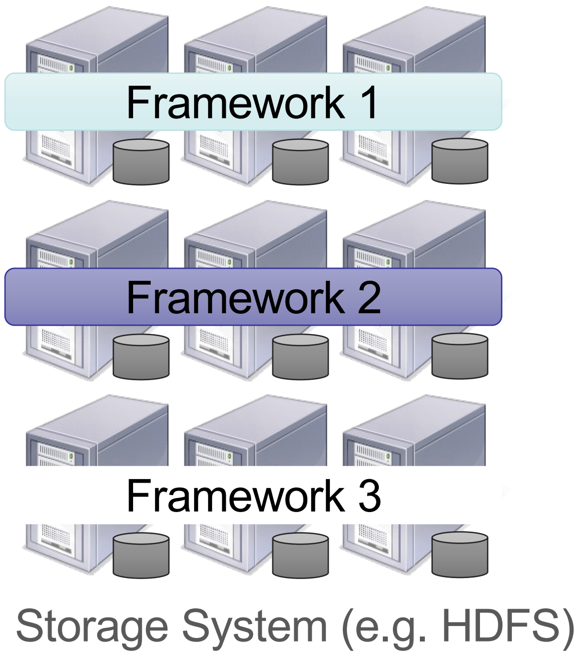
What are the two key design elements?

Design Elements

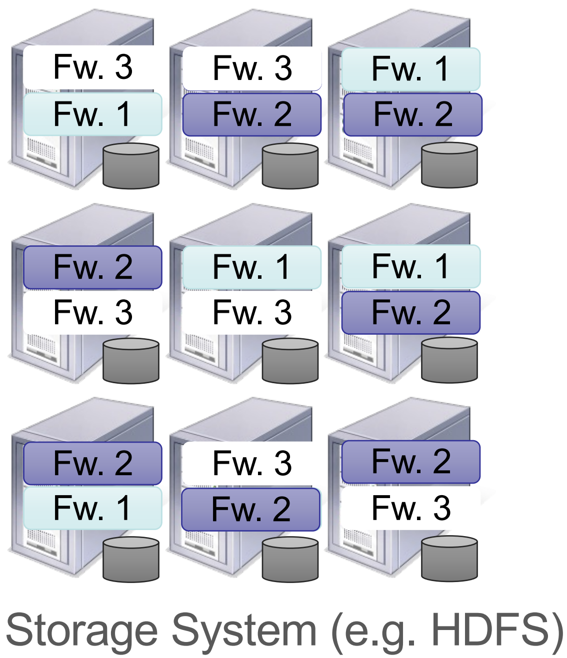
- Fine-grained sharing:
  - Allocation at the level of *tasks* within a job
  - Improves utilization, latency, and data locality
- Resource offers:
  - Simple, scalable application-controlled scheduling mechanism

Element 1: Fine-Grained Sharing

Coarse-Grained Sharing (HPC):



Fine-Grained Sharing (Mesos):



+ Improved utilization, responsiveness, data locality

Element 2: Resource Offers

- Option: Global scheduler
  - Frameworks express needs in a specification language, global scheduler matches them to resources

+ Can make optimal decisions

- – Complex: language must support all framework needs

– Difficult to scale and to make robust

– Future frameworks may have unanticipated needs

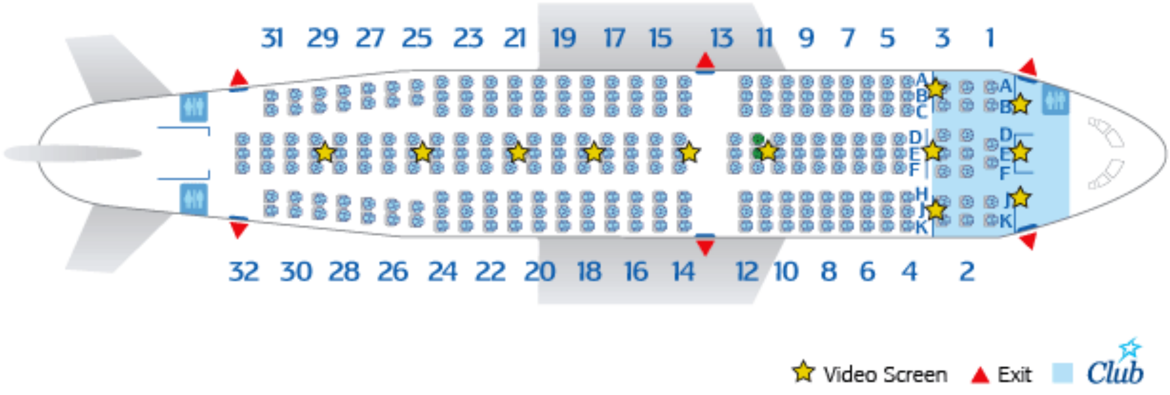
BOSTON UNIVERSITY

Element 2: Resource Offers

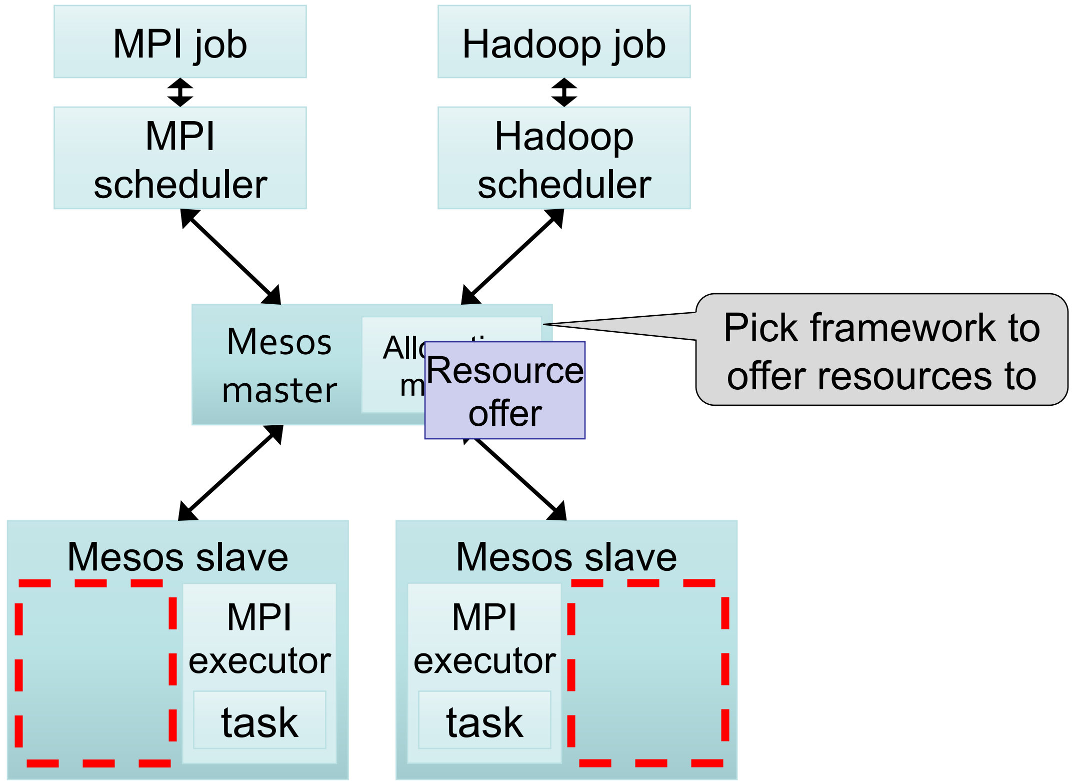
- Mesos: Resource offers
  - Offer available resources to frameworks, let them pick which resources to use and which tasks to launch

+ Keeps Mesos simple, lets it support future frameworks

– Decentralized decisions might not be optimal

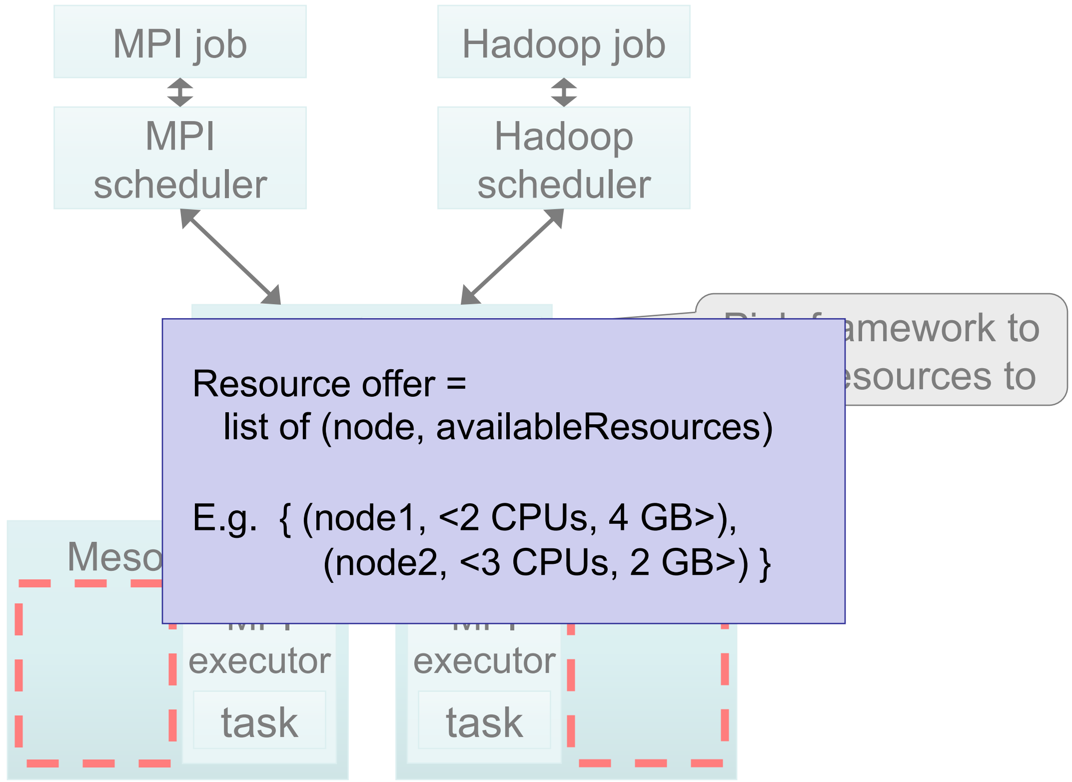


Mesos Architecture

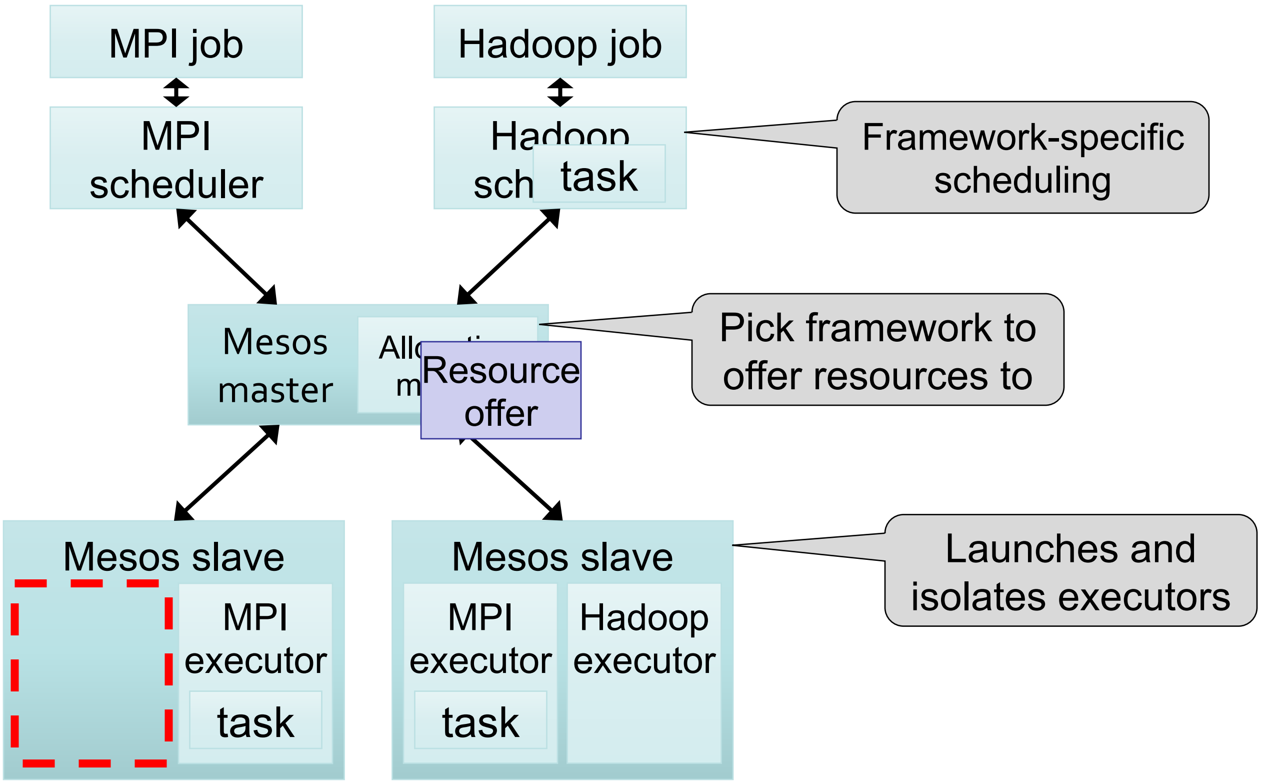


BOSTON UNIVERSITY

Mesos Architecture



Mesos Architecture



Optimization: Filters

- Let frameworks short-circuit rejection by providing a predicate on resources to be offered
  - E.g. "nodes from list L" or "nodes with > 8 GB RAM"
  - Could generalize to other hints as well
- Ability to reject still ensures *correctness* when needs cannot be expressed using filters

BOSTON UNIVERSITY

Implementation Stats

- 10,000 lines of C++
- Master failover using ZooKeeper
- Frameworks ported: Hadoop, MPI, Torque
- New specialized framework: Spark, for iterative jobs (up to 20 × faster than Hadoop)

- Open source in Apache Incubator

Apache incubator

BOSTON UNIVERSITY

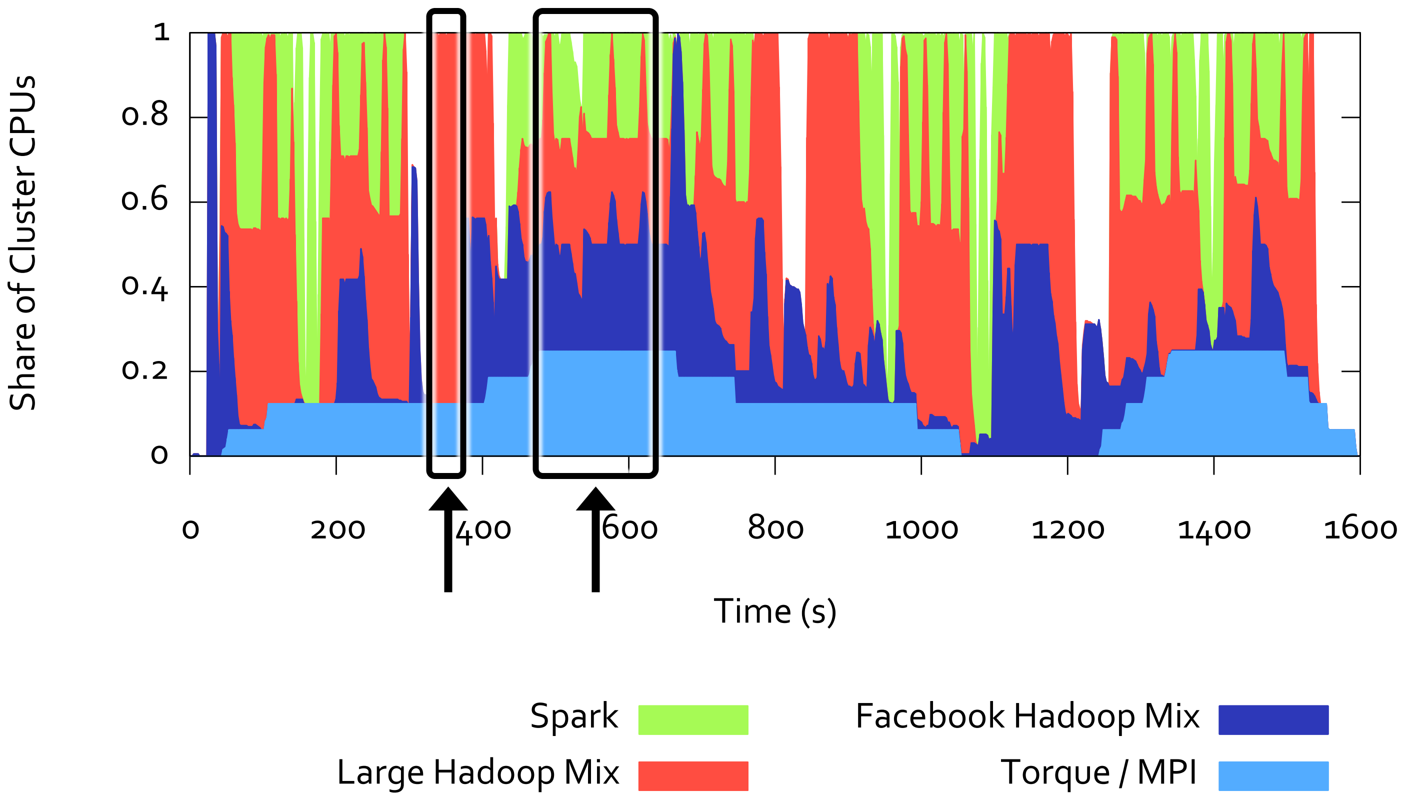
Users

- **Twitter** uses Mesos on > 100 nodes to run ~12 production services (mostly stream processing)
- **Berkeley** machine learning researchers are running several algorithms at scale on Spark
- **Conviva** is using Spark for data analytics
- **UCSF** medical researchers are using Mesos to run Hadoop and eventually non-Hadoop apps

Results

- Utilization and performance vs static partitioning
- Framework placement goals: data locality
- Scalability
- Fault recovery

Dynamic Resource Sharing



Mesos vs Static Partitioning

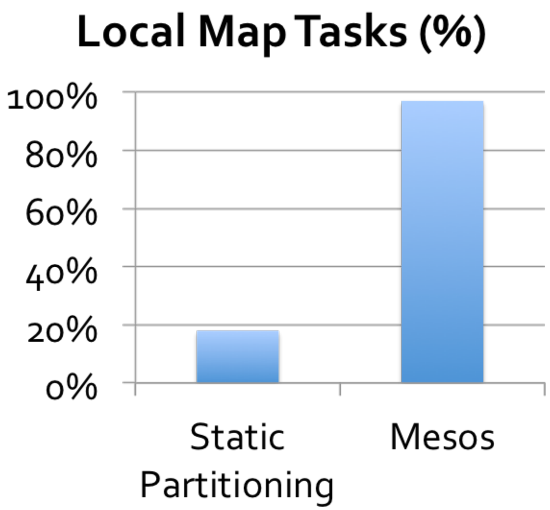
- Compared performance with statically partitioned cluster where each framework gets 25% of nodes

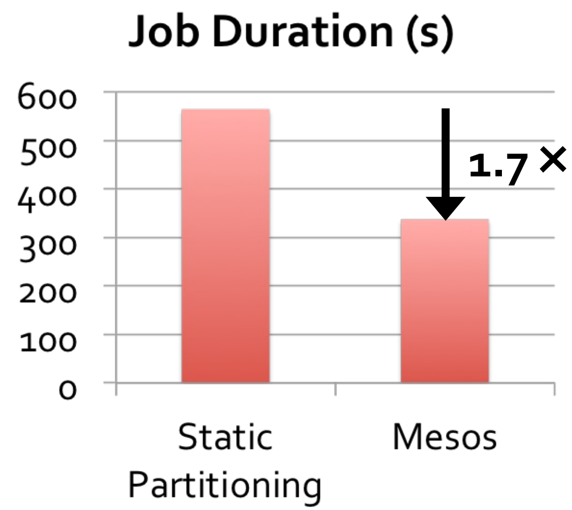
Framework	Speedup on Mesos
Facebook Hadoop Mix	1.14×
Large Hadoop Mix	2.10×
Spark	1.26×
Torque / MPI	0.96×

BOSTON UNIVERSITY

Data Locality with Resource Offers

- Ran 16 instances of Hadoop on a shared HDFS cluster
- Used delay scheduling [EuroSys '10] in Hadoop to get locality (wait a short time to acquire data-local nodes)

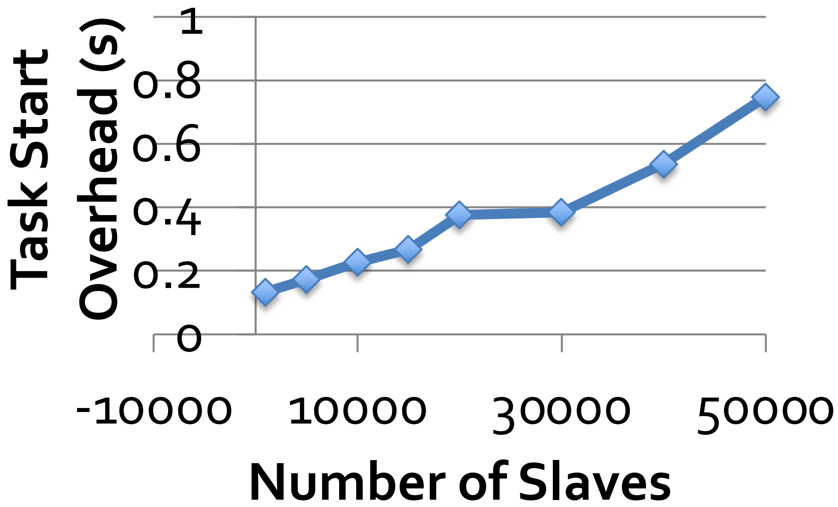




Scalability

- Mesos only performs *inter-framework* scheduling (e.g. fair sharing), which is easier than intra-framework scheduling

Result:
Scaled to 50,000 emulated slaves, 200 frameworks, 100K tasks (30s len)



BOSTON UNIVERSITY

Fault Tolerance

- Mesos master has only *soft state*: list of currently running frameworks and tasks
- Rebuild when frameworks and slaves re-register with new master after a failure
- **Result:** fault detection and recovery in ~10 sec

Conclusion

- Mesos shares clusters efficiently among diverse frameworks thanks to two design elements:
  - **Fine-grained sharing** at the level of tasks
  - **Resource offers**, a scalable mechanism for application-controlled scheduling

- Enables co-existence of current frameworks and development of new specialized ones

- In use at Twitter, UC Berkeley, Conviva and UCSF

BOSTON UNIVERSITY

Framework Isolation

- Mesos uses OS isolation mechanisms, such as Linux containers and Solaris projects
- Containers currently support CPU, memory, IO and network bandwidth isolation
- Not perfect, but much better than no isolation

BOSTON UNIVERSITY

Q&A