Lecture 11
Hadoop & Spark

Dr. Wilson Rivera

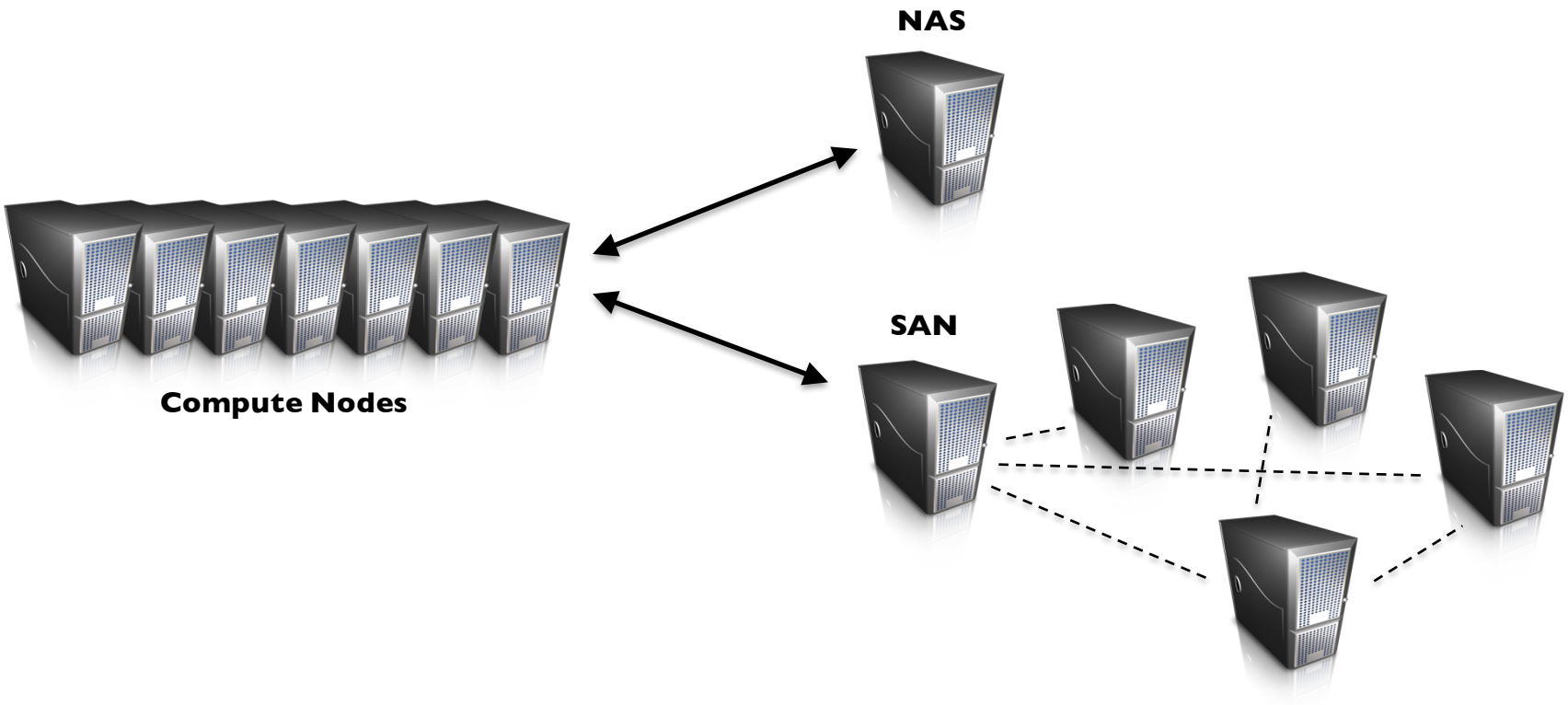
ICOM 6025: High Performance Computing

Electrical and Computer Engineering Department
University of Puerto Rico

Outline

- Distributed File Systems
- Hadoop Ecosystem
- Hadoop Architecture and Features
- Apache Spark

How do we get data to the workers?



Distributed File System

- Don't move data to workers… move workers to the data!
  - Store data on the local disks of nodes in the cluster
  - Start up the workers on the node that has the data local

- Why?
  - Not enough RAM to hold all the data in memory
  - Disk access is slow, but disk throughput is reasonable

- A distributed file system is the answer
  - GFS (Google File System)
  - HDFS (Hadoop Distributed File System)

Apache Hadoop Ecosystem

PIG

R

Hive

MapReduce

HBase

Cassandra

Hadoop Distributed File System (HDFS)

Hadoop

- Designed to reliably store data using commodity hardware
  - Redundant storage
- Designed to expect hardware failures
  - Fault tolerance mechanism
- Intended for large files
  - Not suitable for small data sets
- Not suitable for low latency data access

Hadoop

- **Files** are stored as a collection of blocks
  - Blocks are 64 MB chunks of a file (configurable)
  - Blocks are replicated on 3 nodes (configurable)

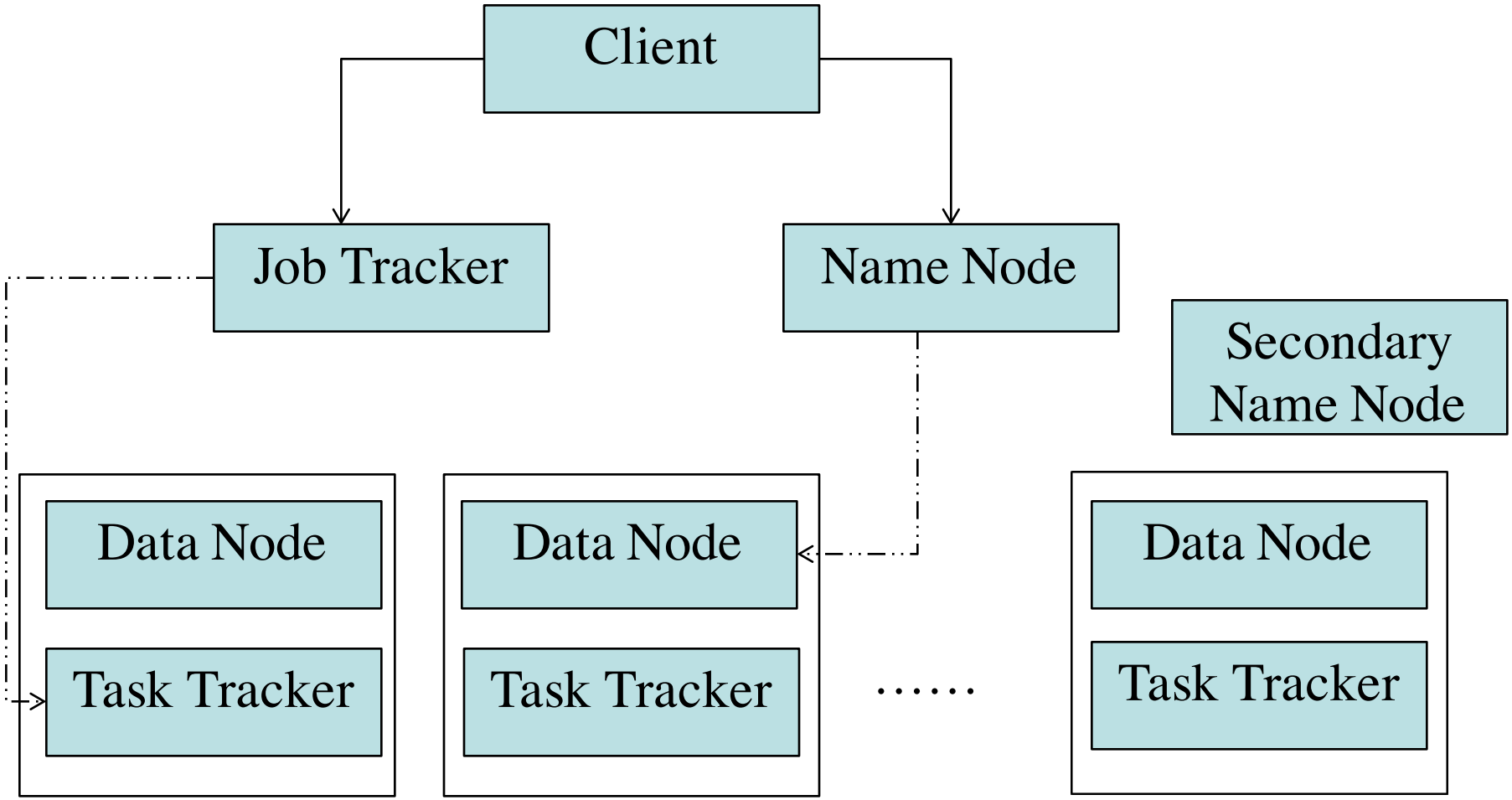
- **NameNode** (NN)
  - Manages metadata about files and blocks
- **DataNodes** (DN)
  - store and serve blocks

Jobs and Tasks in Hadoop

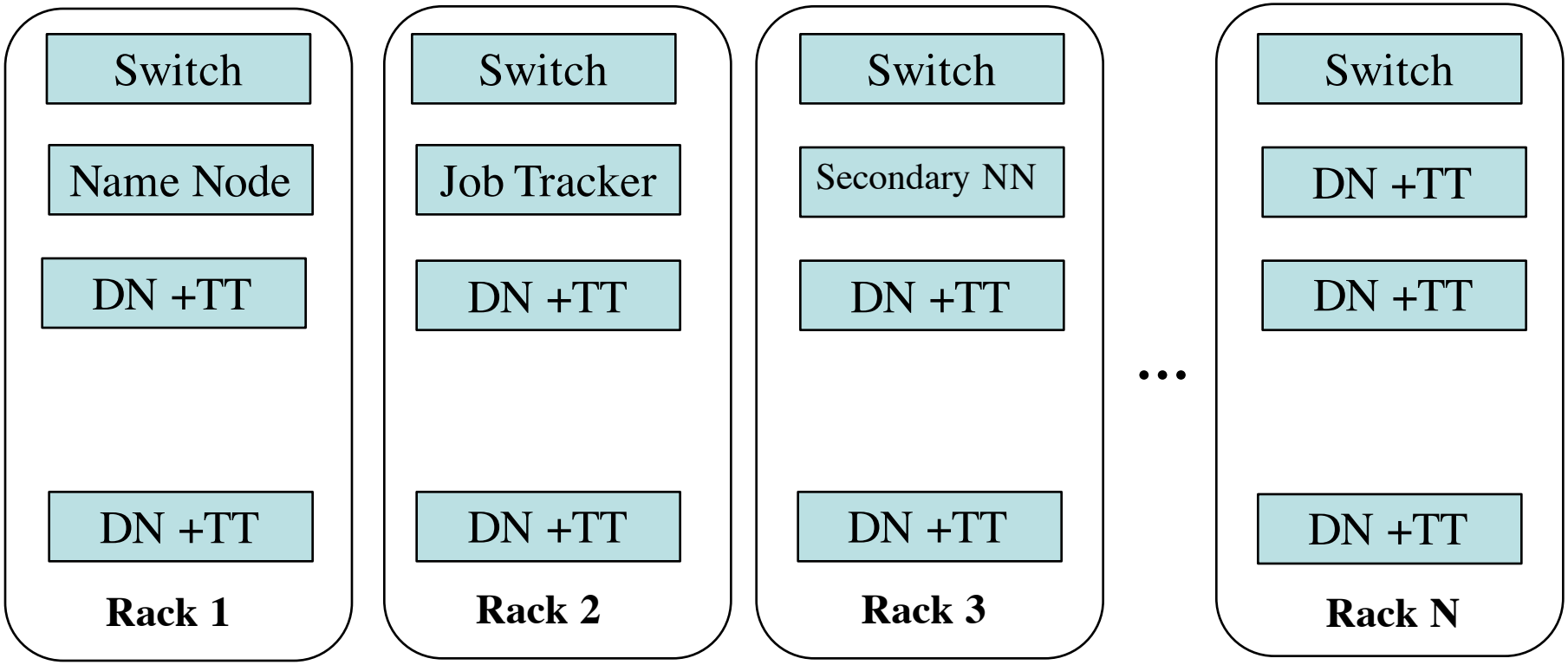
- **Job**: a user-submitted map and reduce implementation to apply to a data set
- **Task**: a single mapper or reducer task
  - Failed tasks get retried automatically
  - Tasks run local to their data, ideally

- **JobTracker** (JT) manages job submission and task delegation
- **TaskTrackers** (TT) ask for work and execute tasks

Hadoop Architecture



Typical Hadoop Cluster



Hadoop Fault Tolerance

- If a Task crashes:
  - Retry on another node:
    - OK for a map because it has no dependencies
    - OK for a reduce because map outputs are on disk
- If a node crashes:
  - Re-launch its current task on other nodes
  - Re-run any maps the node previously ran to get output data
- If a task is going slowly (straggler):
  - Launch second copy of task on another node ("speculative execution")

Hadoop Fault Tolerance

- Reactive way
  - Worker failure
    - Heartbeat, Workers are periodically pinged by master
      - NO response = failed worker
    - If the processor of a worker fails, the tasks of that worker are reassigned to another worker.
  - Master failure
    - Master writes periodic checkpoints
    - Another master can be started from the last checkpointed state
    - If eventually the master dies, the job will be aborted

Hadoop Data Locality

- Move computation to the data
  - Hadoop tries to schedule tasks on nodes with the data
  - When not possible TT has to fetch data from DN
  - Thousands of machines read input at local disk speed. Without this, rack switches limit read rate and network bandwidth becomes the bottleneck.

Hadoop Scheduling

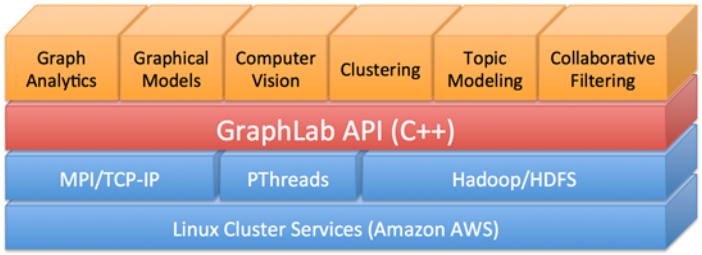
- **Fair Sharing**
  - conducts fair scheduling using greedy method to maintain data locality
- **Delay**
  - uses delay scheduling algorithm to achieve good data locality by slightly compromising fairness restriction
- **LATE** (Longest Approximate Time to End)
  - improves MapReduce application performance in heterogenous environment, like virtualized environment, through accurate speculative execution
- **Capacity**
  - Supports multiple queues for shared users and guarantees each queue a fraction of the capacity of the cluster

MPI vs. Map Reduce

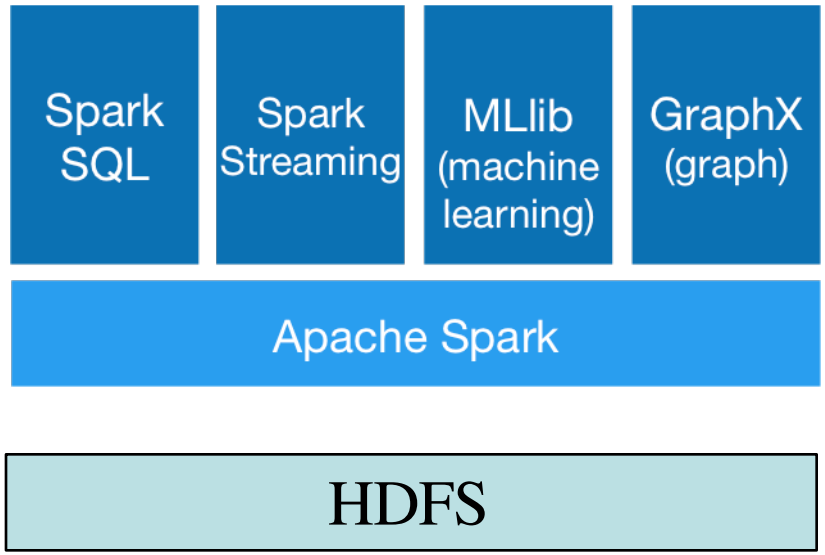
	MPI	Map Reduce
Programming in communication	Explicit	Implicit
Fault tolerance	No	Yes
Main resources	Memory	I/O
Latency	Low	High

Other Frameworks

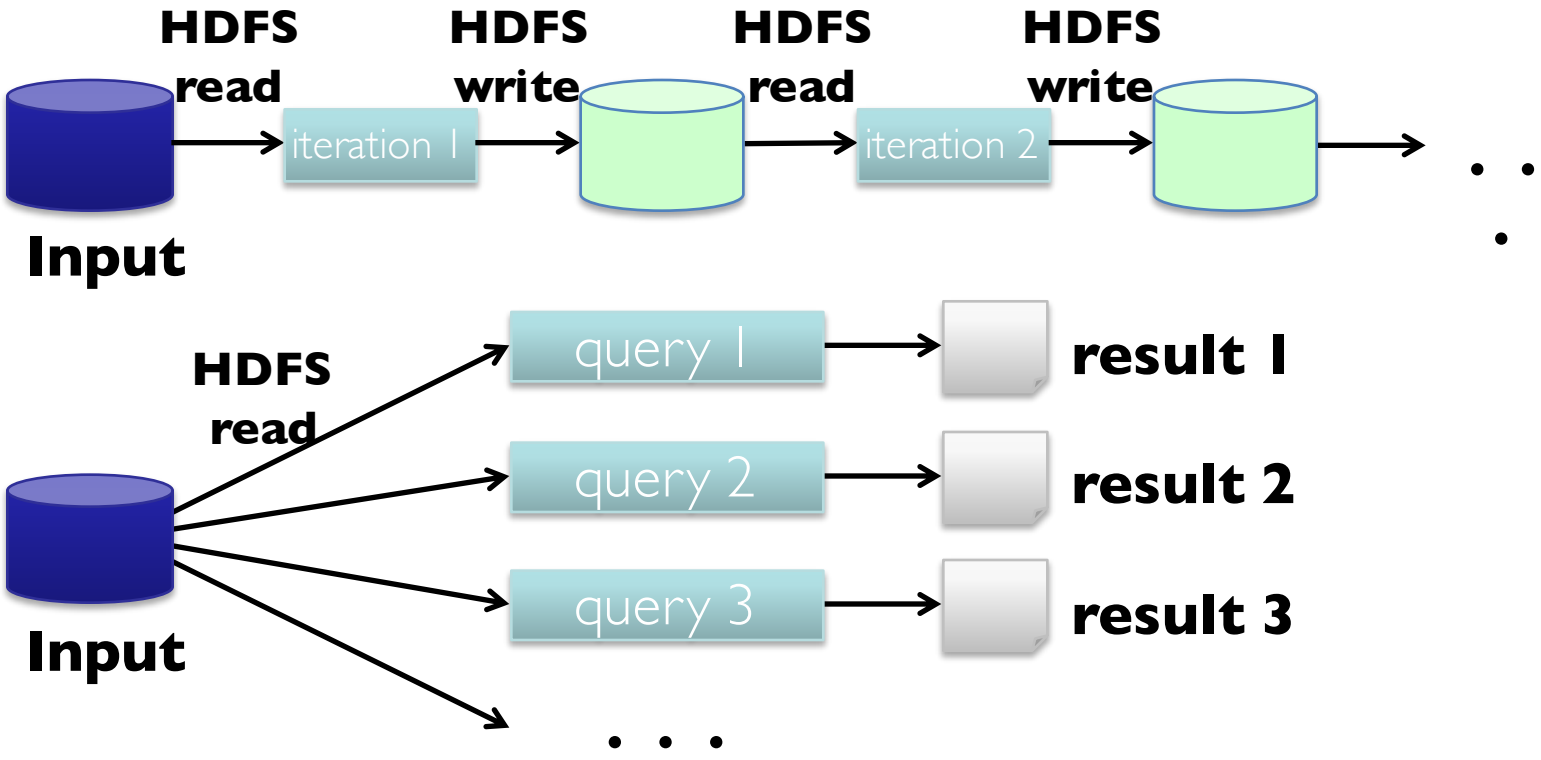
- Batch Processing
  - Hadoop
    - Independent tasks
  - GraphLab
    - Dependent tasks
- Interactive Processing
  - Drill
    - Low latency for Interactive data analysis
  - Spark
    - Resilient distributed datasets
    - Primitives for in memory computing
    - 100x faster than Hadoop!!
- Stream processing
  - Storm, Apache S4



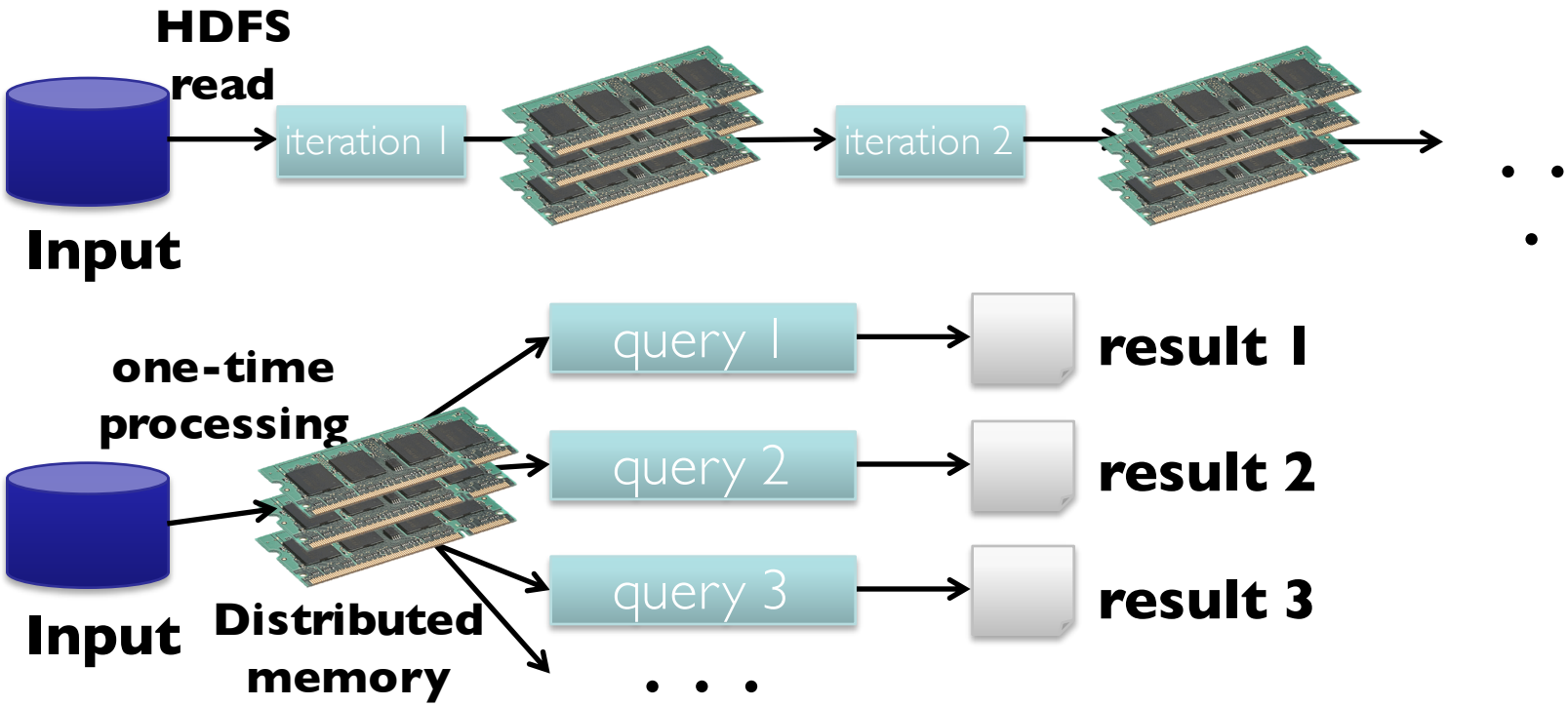
Apache Spark



Use Memory Instead of Disk



In-Memory Data Sharing



10-100x faster than network and disk

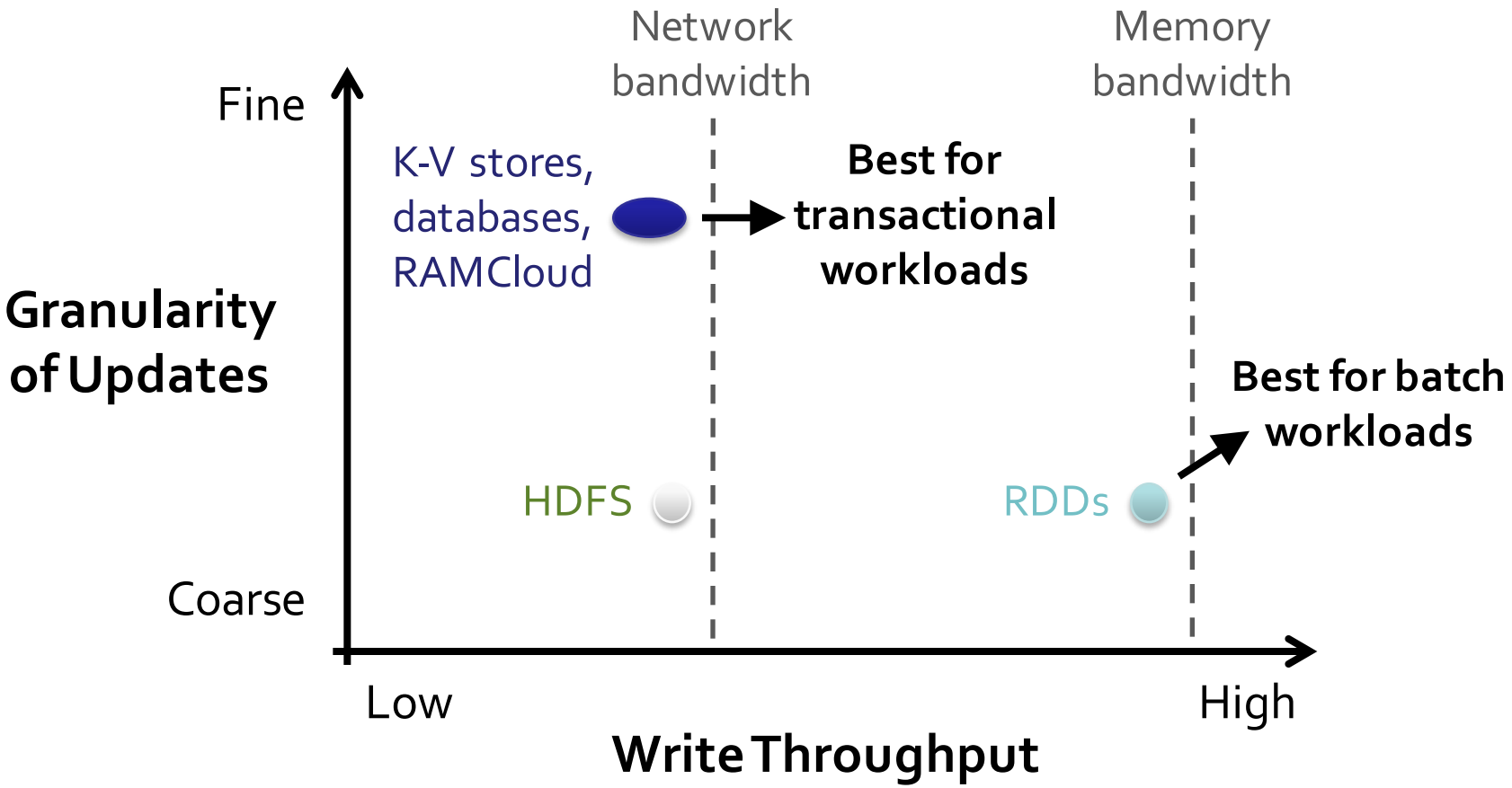
Resilient Distributed Datasets (RDDs)

- Write programs in terms of operations on distributed datasets
- Partitioned collections of objects spread across a cluster, stored in memory or on disk
- RDDs built and manipulated through a diverse set of parallel transformations (map, filter, join) and actions (count, collect, save)
- RDDs automatically rebuilt on machine failure

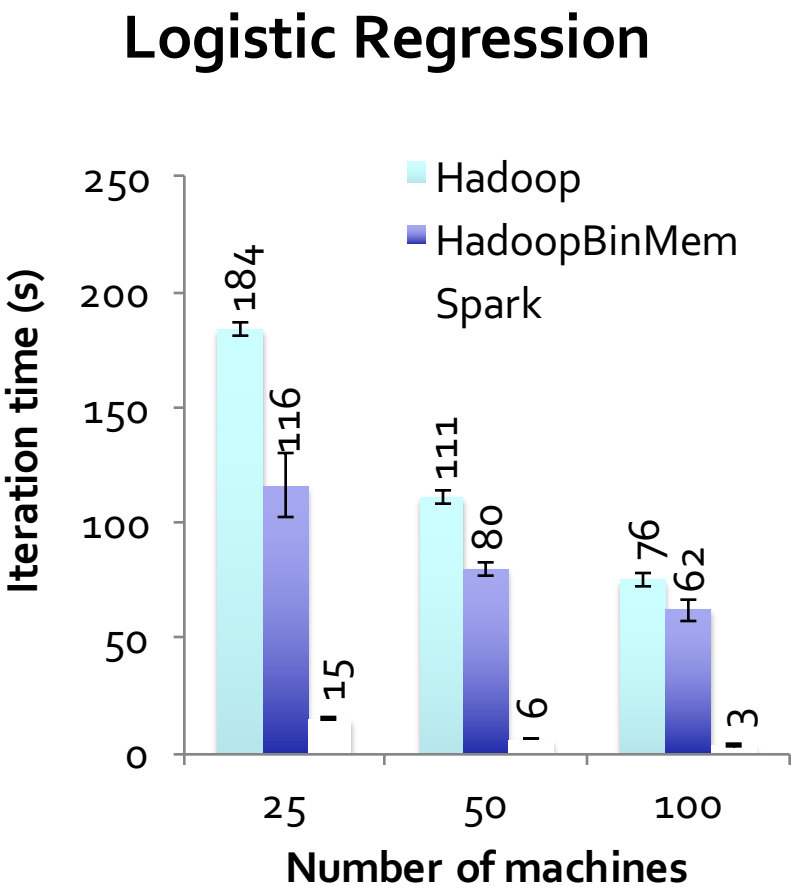
The Spark Computing Framework

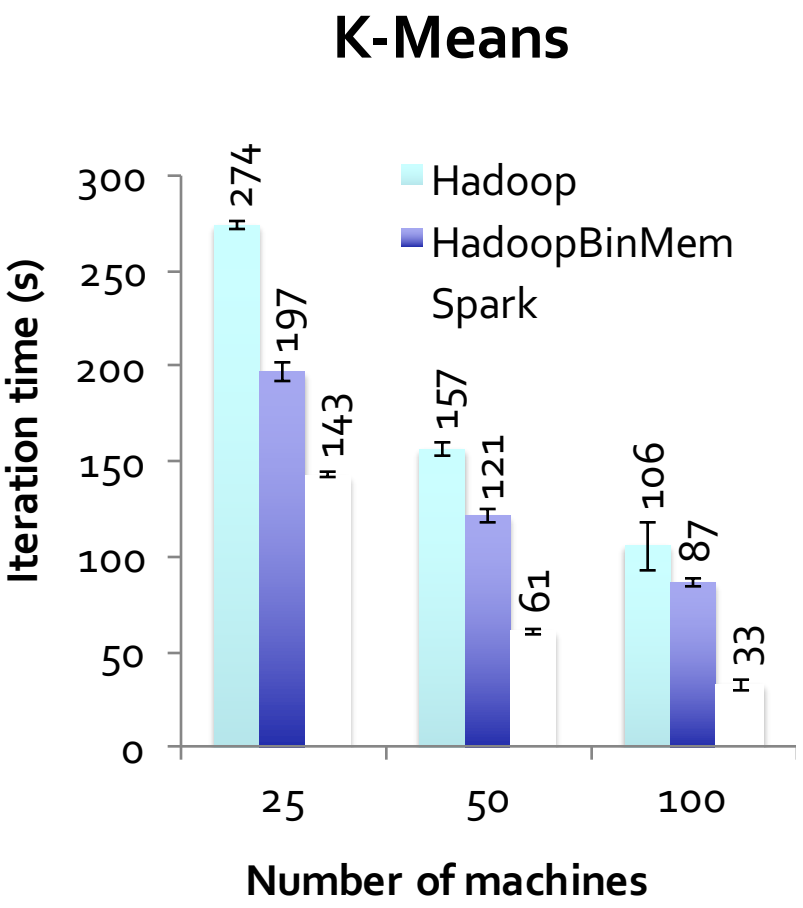
- Provides programming abstraction and parallel runtime to hide complexities of fault-tolerance and slow machines
- "Here's an operation, run it on all of the data"
  - I don't care where it runs (you schedule that)
  - In fact, feel free to run it twice on different nodes

Tradeoff Space



Scalability





Spark and Map Reduce Differences

	Hadoop Map Reduce	Spark
Storage	Disk only	In-memory or on disk
Operations	Map and Reduce	Map, Reduce, Join, Sample, etc…
Execution model	Batch	Batch, interactive, streaming
Programming environments	Java	Scala, Java, R, and Python

Summary

- Distributed File Systems
- Hadoop Ecosystem
- Hadoop Architecture and Features
- Apache Spark