Introduction to Grid Computing via Map-Reduce & Hadoop

Jothi Padmanabhan
jothipn@yahoo-inc.com
Yahoo! Bangalore
Dec 5, 2010 @ New Delhi

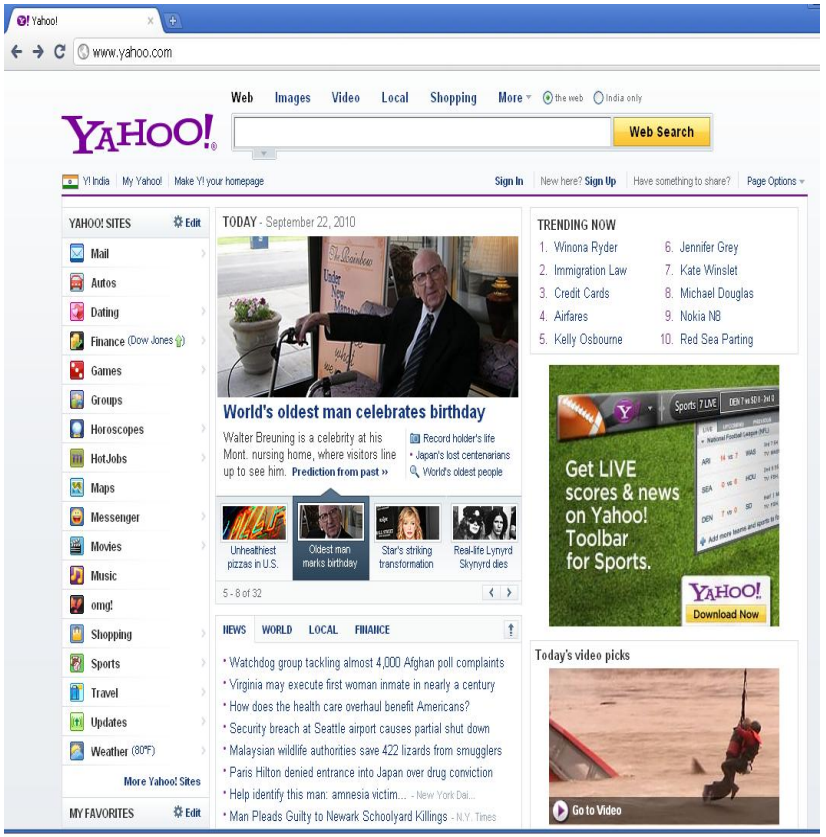
Outline

- Large Scale Computing Needs @ Yahoo!
- Grid Computing as a solution
- Map-Reduce paradigm for large-scale computing
- Apache Hadoop (Map/Reduce Framework)
- Q & A

The Challenge:
Very large scale data processing

Internet Scale Generates Big Data

- Yahoo is the most Visited Site on the Internet
  - 600M+ Unique Visitors per Month
  - Billions of Page Views per Day
  - Billions of Searches per Month
  - Billions of Emails per Month
  - IMs, Address Books, Buddy Lists, …
  - **Terabytes of Data per Day**
- And we crawl the Web
  - 150+ Billion Pages
  - 5+ Trillion Links
  - **Petabytes of data**



…terabytes, 100s of terabytes, petabytes, 10s of petabytes…

Processing big data

- Search
  - Crawl Internet, Build “Webmap & Index.
  - Serve Search results
- Ad and Content Serving
  - Select targeted Ads to display, based on content, context & user.
  - Personalize content (millions of unique views of the same page!)
- Analytics
  - Model Ad performance using click patterns
  - Study user behavior (based on clicks, page views, time spent, …)

How to Process Big Data?

- Processing of the order of ~100s of terabytes / day.
- Storage of the order of 10s of PBs.
- Just reading 100 terabytes of data can be overwhelming
  - On a standard computer (100 MBPS):
    - ~11 days
  - Across a 10Gbit link (high end storage solution):
    - 1 day
  - On 1000 standard computers:
    - 15 minutes!

Outline

- Large Scale Computing @ Yahoo!
- Grid Computing as a solution
- Map-Reduce paradigm for large-scale computing
- Apache Hadoop (Map-Reduce Framework)
- Q & A

Traditional Parallel Computing vs. Grid (Utility) Computing

	Traditional Parallel Computing	Grid/Utility Computing
Problem Class	•Computation Intensive •Floating Point oriented. •Generally parallelizable, as bounded by Amdahl's law. • E.g., Computational Fluid Dynamics, Molecular Dynamics, Graphics & Viz.	•Data Intensive •Bulk-processing oriented. •Typically embarrassingly parallel •E.g., Data Mining, User Data Analysis, "Webmap"
Hardware	Traditionally Specialized Big-Iron. Commodity clusters with High-Speed networks now common.	Commodity (Intel Serverboards!)
Communication & Synchronization Patterns	•Shared or distributed-shared memory based. •Iterative. •Point-to-point & collective.	•Distributed memory (message / RPC) based. •Infrequent communication.
Technology Examples	•Message Passing Interface (MPI) •OpenMP •Parallel Virtual Machines (PVM)	Map-Reduce

Some Grids

- SETI@home (Search for Extra Terrestrial Intelligence)
- PlanetLab (mostly Universities)
- Grids based on Globus
- Hadoop (that's us)
- Google
- Amazon (EC2, S3)
- GARUDA (connecting IITs and C-DAC centers)
- many more

Grid Computing @ Yahoo!

- Hardware
  - 10,000s of commodity machines.
  - Sample Configs: Harpertown, 16 GB RAM, 4 SATA disks / node.
  - No special hardware, storage
- Software: Hadoop Stack
  - Hadoop (Map-Reduce Framework)
  - Hadoop DFS (Distributed File System)
  - PIG, Hive, Oozie – High level abstractions
- 1000s of applications
  - Search, Advertizing, User Data Analytics, …
- Multiple Hadoop clusters
  - Largest cluster: ~4000 nodes (32K cores)

What is hard in Grid Computing

- Insulating Applications from Reliability problems
  - In large clusters, computers fail every day, and in different ways
  - For a single machine: MTBF is ~3 years
  - On 1000 machines, MTBF is ~1 day
  - Data is corrupted or lost (disk, memory, network)
  - Computations are disrupted
- Providing a framework to make programming on clusters easy
  - Need simple programming paradigms or languages
  - Traditional debugging and performance tools don’t apply

What is hard in Grid Computing

- Storage
  - How do I ensure reliable storage of PetaBytes of data?
  - How do I provide efficient data read/write path-ways on all cluster machines?
- Resource management & Scheduling
  - How will concurrent jobs share resources?
  - How do I make sure all resources are utilized?
- Others
  - Operability
  - Uptime (24x7)
  - Security
  - Monitoring

Recap

- Our challenge is large scale data processing
- We use grids/clusters of thousands of networked commodity machines
- Hadoop is the software that ties this together (more later)
- Hadoop abstracts away from programmers the hard problems that come with scale

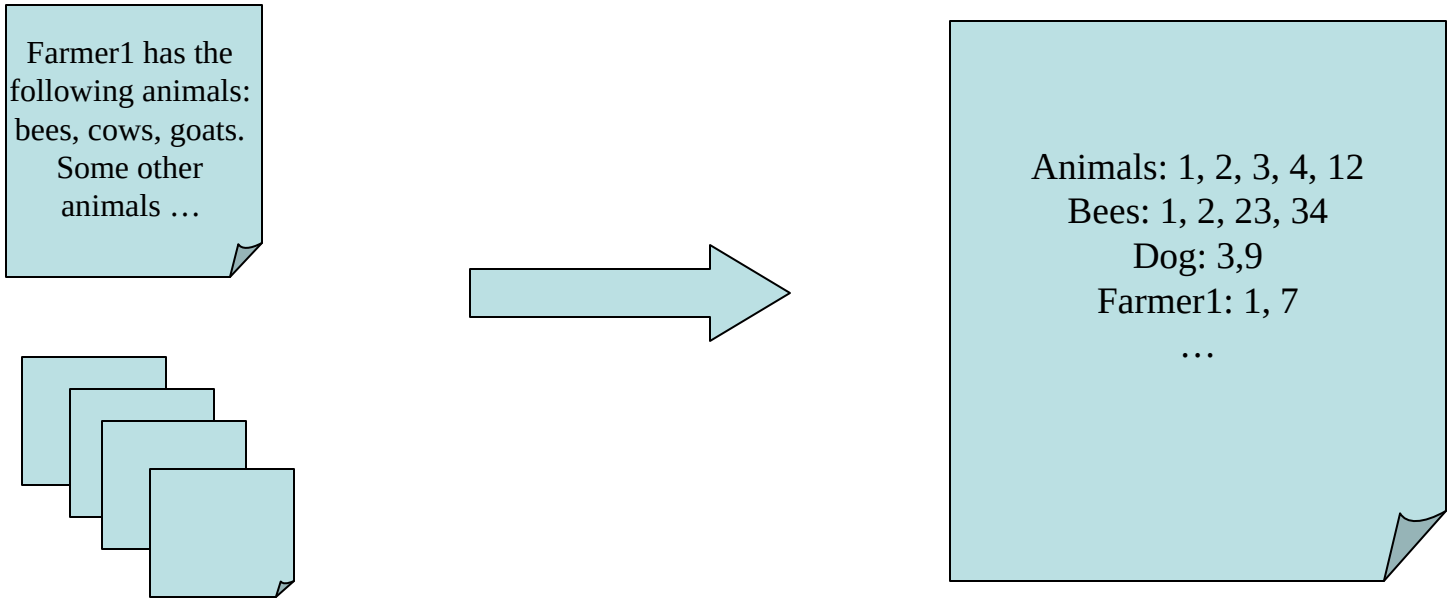
Outline

- Large Scale Computing @ Yahoo!
- Grid Computing as a solution
- Map-Reduce paradigm for large-scale computing
- Apache Hadoop (Map-Reduce Framework)
- Q & A

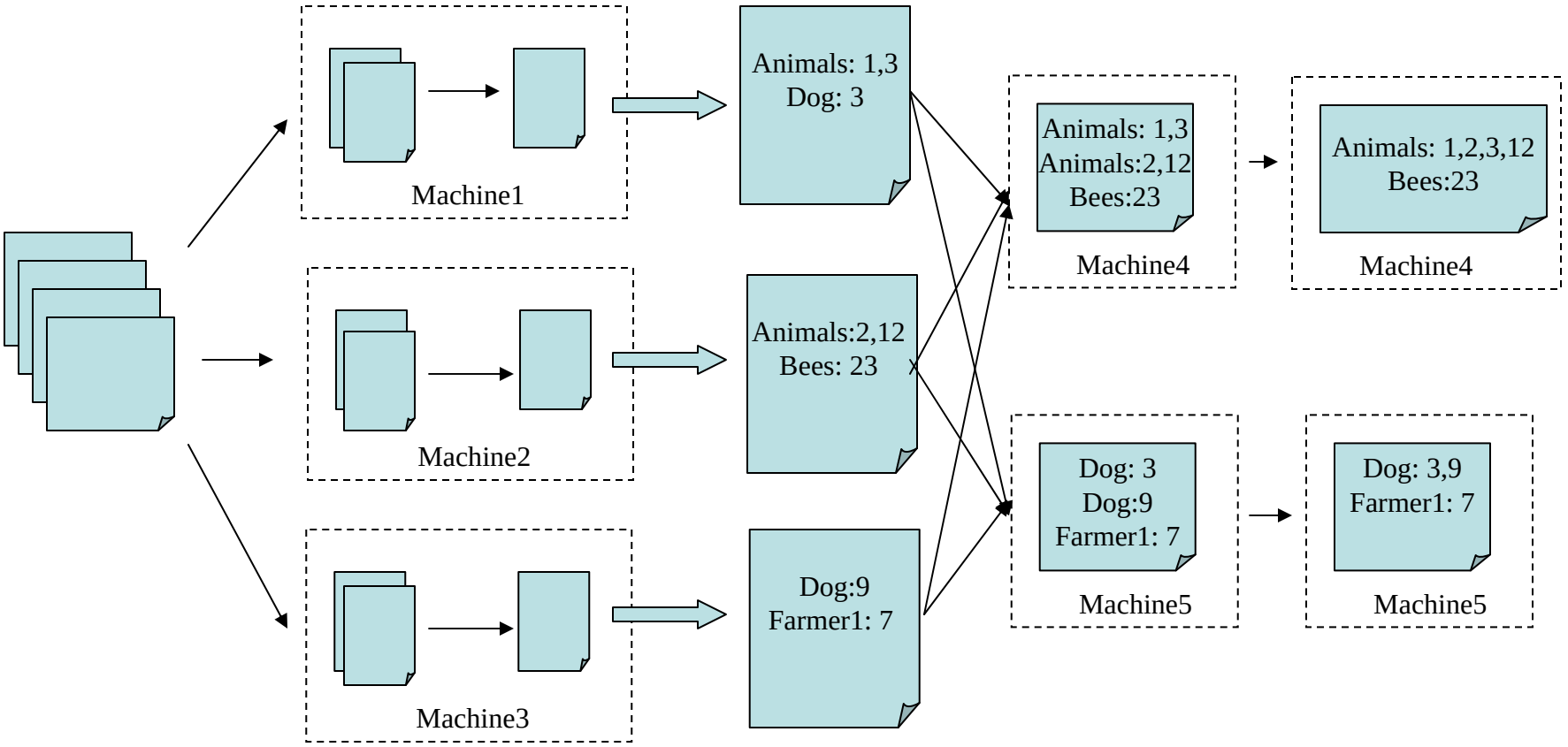
Quick Intro to Map-Reduce

- A Programming Paradigm that abstracts your problem into *Map* and *Reduce* operations.
- Suitable to batch-process large amounts of data on multiple machines.
- As an example, consider the problem of creating an index for search
  - You have hundreds of documents
  - You want to build an index so you can search on them (a map of words to their source docs)
  - You have multiple machines

The problem: inverted index



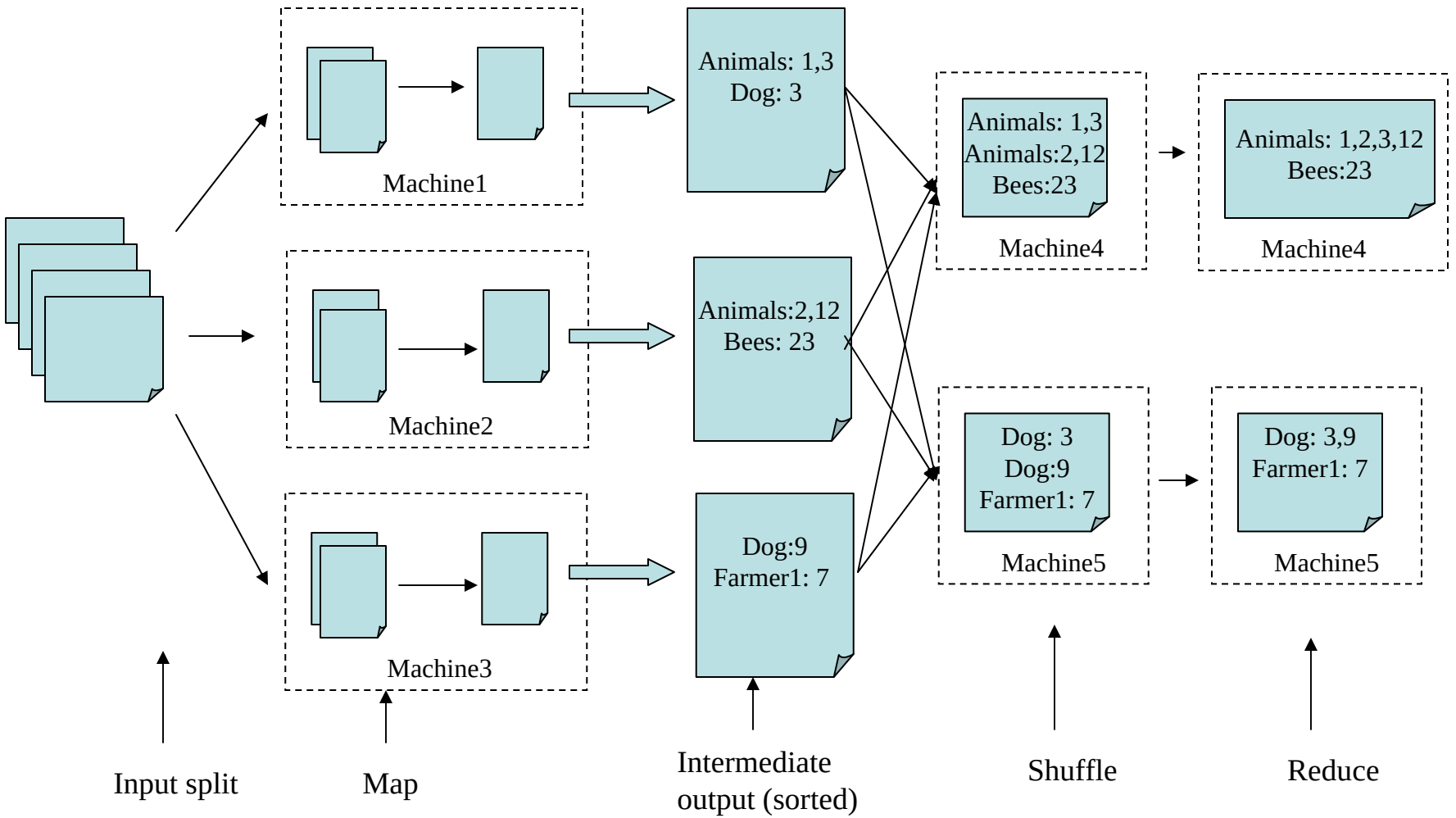
Building an inverted index



This is Map-Reduce

- General form:
  - Map: (K1, V1) → list(K2, V2)
  - Reduce: (K2, list(V2)) → list(K3, V3)
- In our example
  - Map: (line#, word) → [(word, doc-num)]
  - Reduce: (word, [doc1, doc3, ...]) → [(word, “doc1, doc3, …”)]
- Invented by two engineers in Google
  - Gained immense popularity because of applicability to large class of problems
  - Has been observed as a functional programming extension to scale.

Mapping our example to Map-Reduce



Why Map-Reduce?

- May seem like an arbitrary model...
- Generalizes common patterns
  - for very large data collections
- Natural for:
  - statistics, e.g., counting
  - offline databases (index creation, update)
  - extraction, refinement, exploration, etc.
- Permits optimizations
  - scalable, distributed, data-locality, etc.

Map-Reduce on a larger scale

- Take the previous example and expand it
  - billions of web pages
  - index can reach a few petabytes
  - Thousands of machines
  - Run multiple jobs/programs
- We need a platform that can let us do this. What should the platform support?

Grid Components

- Storage (file system)
  - highly available (machines can fail)
  - replicated (large amounts of data, can't store copies on every machine)
  - data consistency (due to replication)
- Framework (Map-Reduce)
  - user should just say where the data is, how to split it, and what to do with each data set
  - Framework should start/stop tasks, move data/computation, manage execution

Grid Components

- Communication / data exchange
  - Serialize/deserialize data (into files, across networks) in a generic manner
  - Language/OS neutral
  - What transport mechanism? Sockets?
- Provisioning
  - To utilize machines, we need to run many applications. How do we share resources across many apps?
- Monitoring
  - error detection, logging, etc

Recap

- We saw an example of Map-Reduce
- What do we need in a Grid:
  - Storage
  - Programming model (Map-Reduce)
  - Communication
  - Provisioning
  - Monitoring, error handling

Outline

- Large Scale Computing @ Yahoo!
- Grid Computing as a solution
- Map-Reduce paradigm for large-scale computing
- Apache Hadoop (Map-Reduce Framework)
- Q & A

Hadoop

- Framework for running applications on large clusters built of commodity hardware
- Lets one easily write and run applications that process vast amounts of data (petabytes).
- Distributed File System
  - Modeled on GFS
- Distributed Processing Framework
  - Using Map-Reduce metaphor
- Scheduler/Resource Management

Hadoop is

- Completely Open Source
  - Top level Apache project
- Written in Java
  - Runs on Linux, Mac OS/X, Windows, and Solaris
  - Commodity hardware
  - Client apps can be written in various languages

Apache Hadoop – Open Source

- Multi-Organization Development Community
  - Yahoo! is primary contributor
  - Also IBM, Amazon, Facebook, Cloudera, …
  - And various independent programmers
- Anyone can use Hadoop, for free!
  - Many organization have registered their Hadoop usage / clusters
  - Hadoop is used in Universities on several continents
- Anyone can enhance Hadoop!
  - Fix a bug, submit a test case, write some docs, starting is easy
  - Publish a new Hadoop application or two
  - Hadoop's direction is determined by those who invest in it
- http://hadoop.apache.org/

Hadoop – Core Components

- HDFS
- Map-Reduce Framework
- Resource Manager/Scheduler

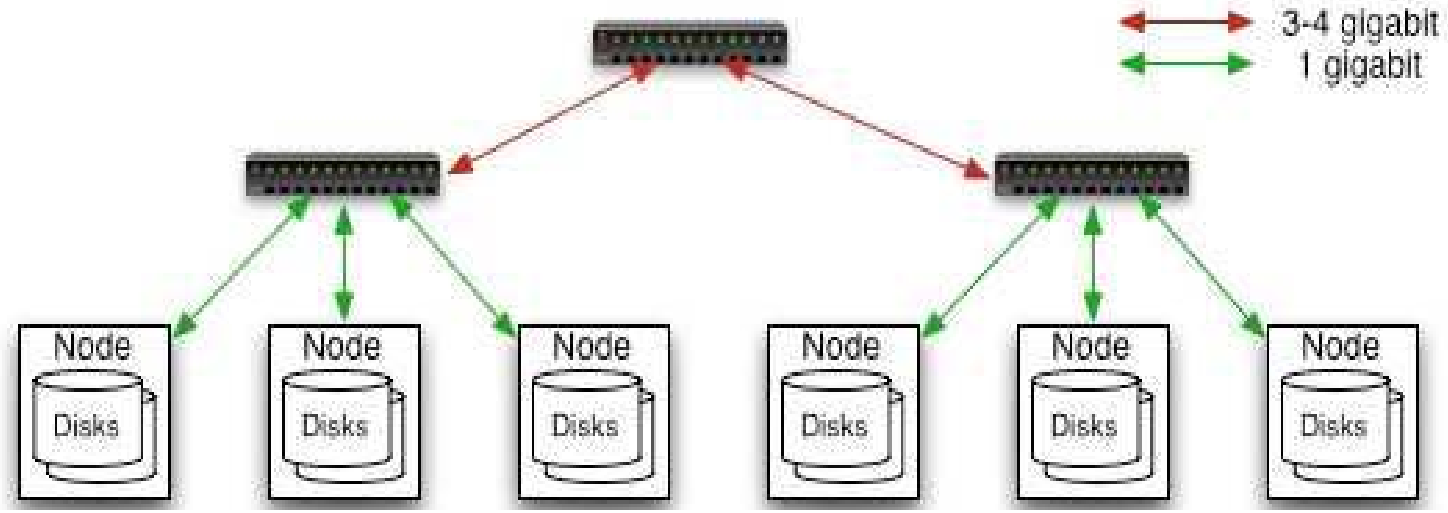
HDFS Goals

- Very Large Distributed File System
  - 10K nodes, 100 million files, 10 PB
- Assumes Commodity Hardware
  - Failure is expected, rather than exceptional
  - The number of nodes in a cluster is not constant
- Streaming Data Access
  - Write-Once, Read-Many pattern
- Batch processing

HDFS is not for

- Low latency data access
- Lots of small files
- Multiple writers, arbitrary file modifications

Commodity Hardware



- Nodes are commodity PCs
- 30-40 nodes/rack
- Uplink from rack is 3-4 gigabit
- Rack-internal is 1 gigabit

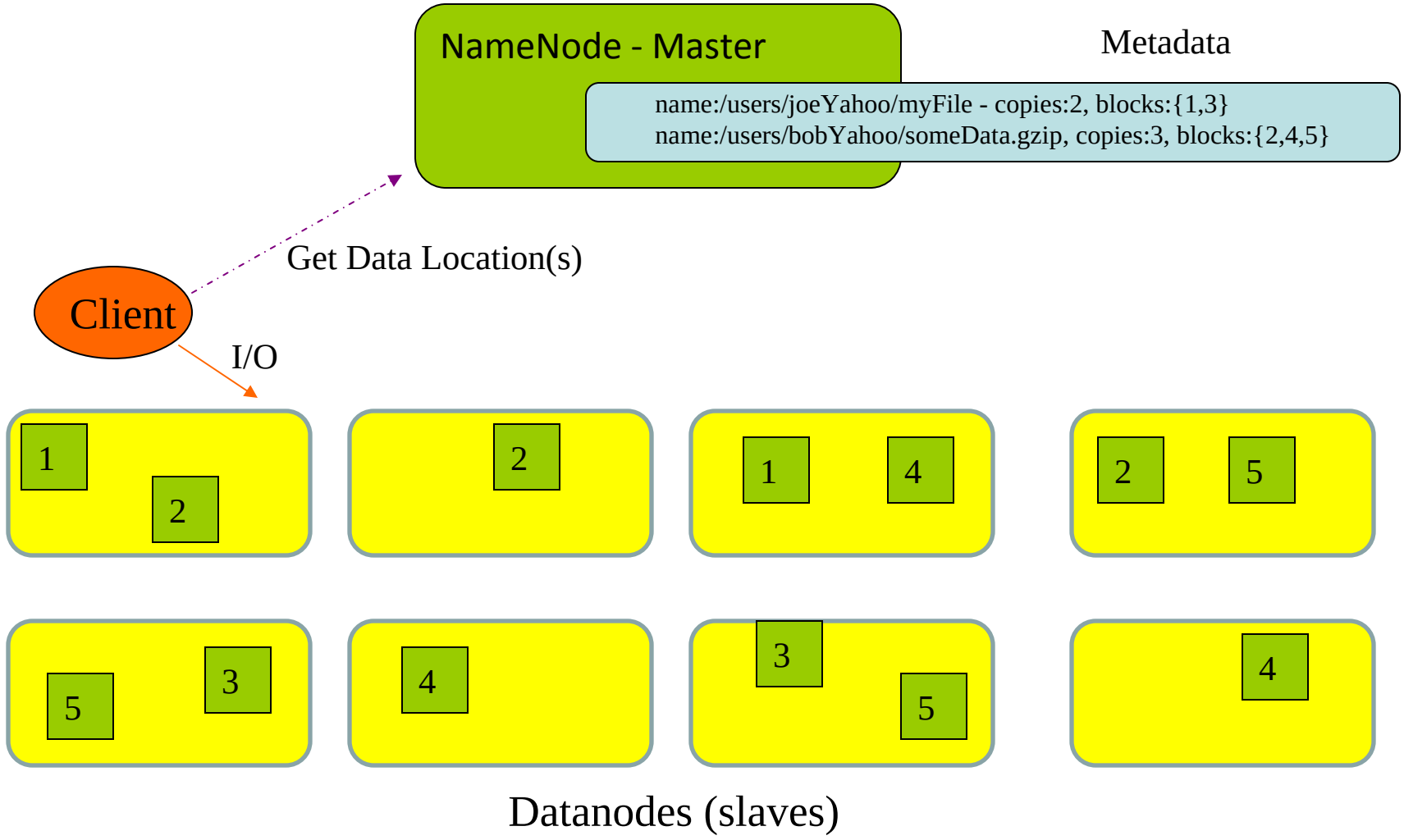
Hadoop Distributed File System

- Fault tolerant, scalable, distributed storage system
- Designed to reliably store very large files across machines in a large cluster
- Data Model
  - Data is organized into files and directories
  - Files are divided into uniform sized blocks and distributed across cluster nodes
  - Blocks are replicated to handle hardware failure
  - Filesystem keeps checksums of data for corruption detection and recovery
  - HDFS exposes block placement so that computes can be migrated to data

HDFS Architecture

- Master-Worker architecture
- HDFS Master “Namenode”
  - Manages the filesystem namespace
  - Controls read/write access to files
  - Manages block replication
  - Checkpoints namespace and journals namespace changes for reliability
- HDFS Workers “Datanodes”
  - Serve read/write requests from clients
  - Perform replication tasks upon instruction by Namenode

HDFS architecture



Data Correctness

- Data is checked with CRC32
- File Creation
  - Client computes checksum per 512 byte
  - DataNode stores the checksum
- File access
  - Client retrieves the data and checksum from DataNode
  - If Validation fails, Client tries other replicas

Namenode

- Stores metadata
  - List of files
  - List of Blocks for each file
  - List of DataNodes for each block
  - File attributes, e.g creation time, replication factor
- Where is metadata stored?
  - The entire metadata is in main memory
  - No demand paging of meta-data
- Single Point of Failure
  - Transaction Log stored in multiple directories
  - Secondary namenode merges transaction log with data in memory, uploads back to namenode.

Map-Reduce Framework in Hadoop

- Abstracts functionality common to all Map-Reduce applications
  - Distribute tasks to multiple machines
  - Sorts, transfers and merges intermediate data from all machines from the Map phase to the Reduce phase
  - Monitors task progress
  - Handles faulty machines, faulty tasks transparently
- Provides pluggable APIs and configuration mechanisms for writing applications
  - Map and Reduce functions
  - Input formats and splits
  - Number of tasks, data types, etc…
- Provides status about jobs to users

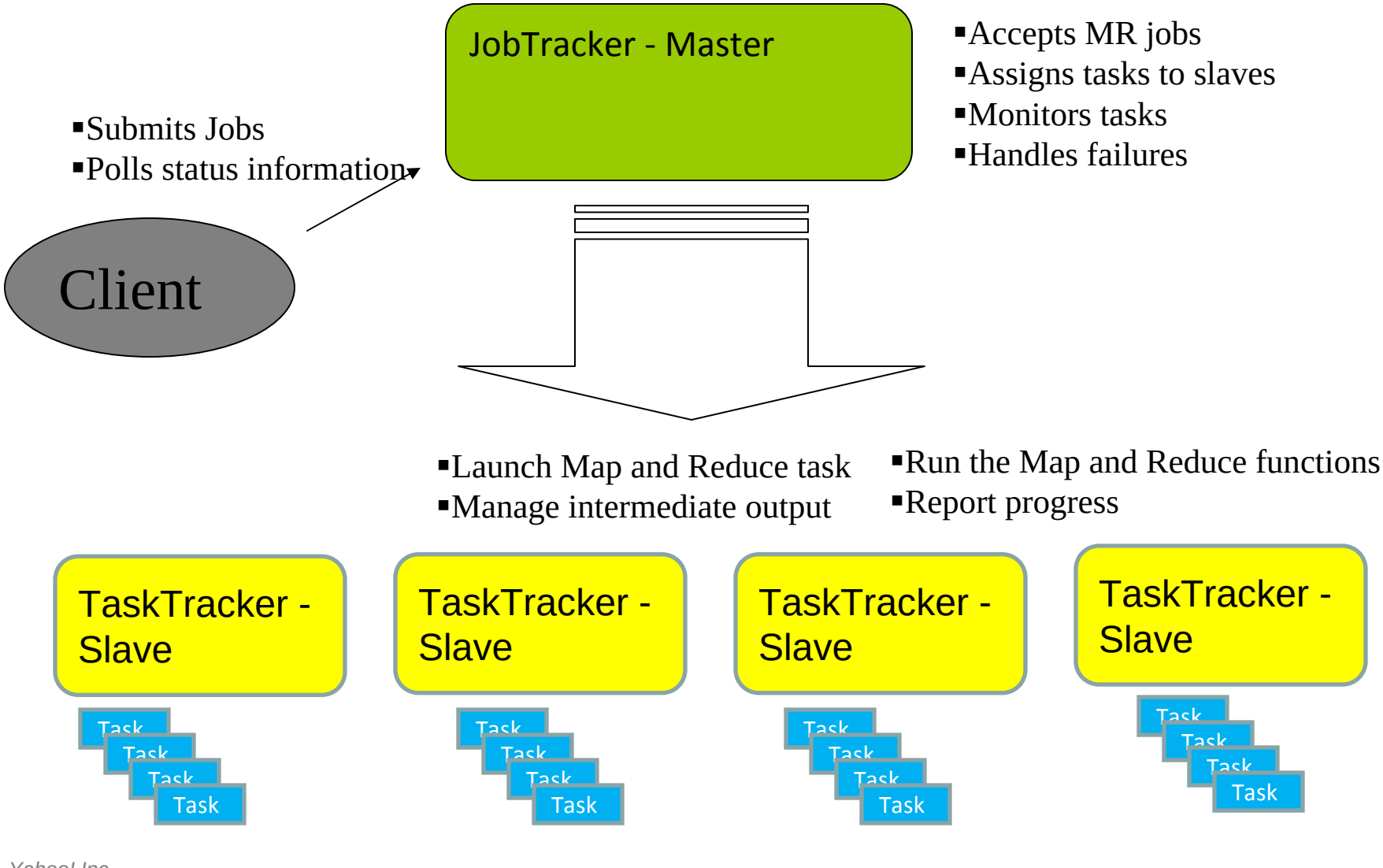
Map-Reduce features

- Fine grained Map and Reduce tasks
  - Improved load balancing
  - Faster recovery from failed tasks
- Automatic re-execution on failure
  - In a large cluster, some nodes are always slow or flaky
  - Introduces long tails or failures in computation
  - Framework re-executes failed tasks
- Locality optimizations
  - With big data, bandwidth to data is a problem
  - Map-Reduce + HDFS is a very effective solution
  - Map-Reduce queries HDFS for locations of input data
  - Map tasks are scheduled local to the inputs when possible

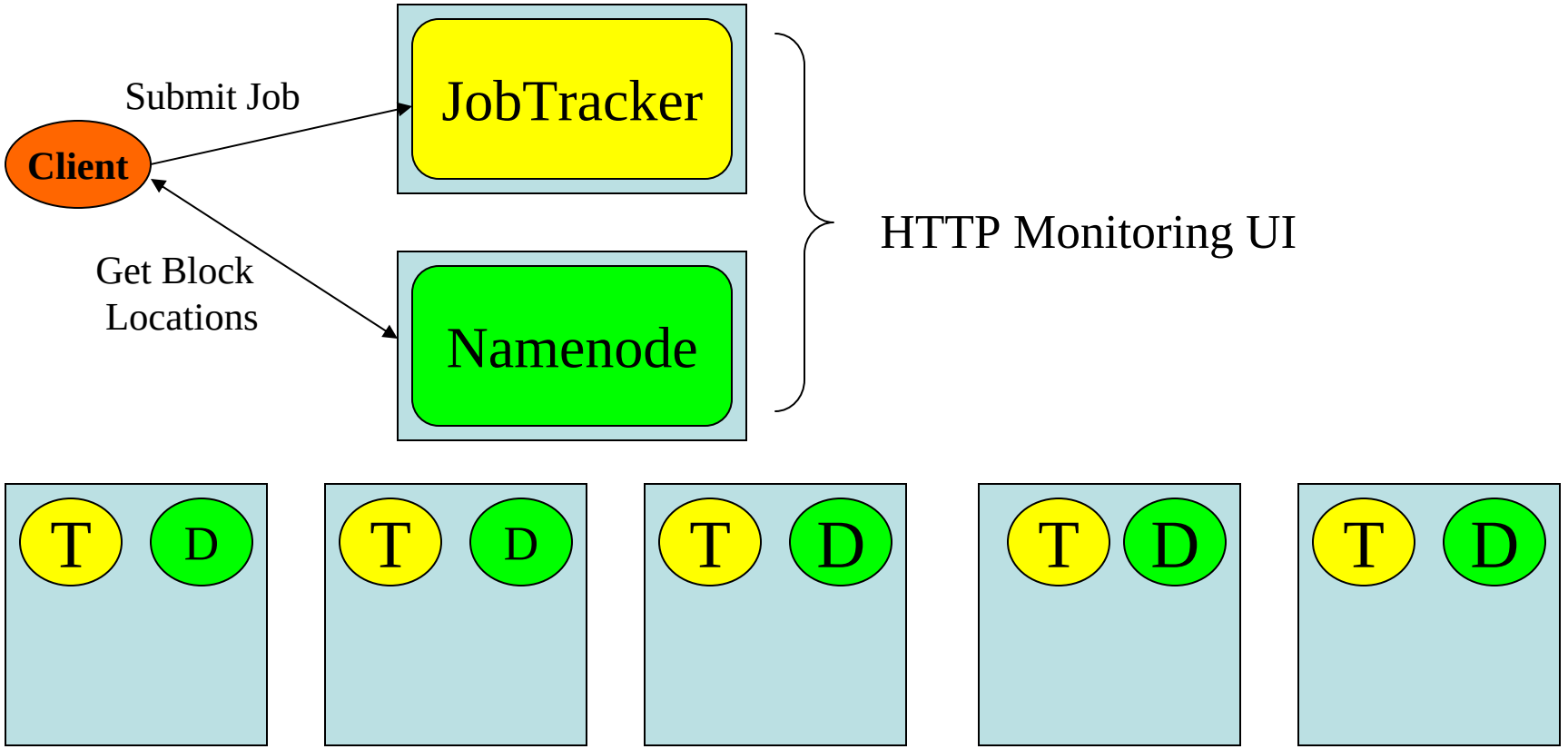
Map-Reduce Framework Architecture

- Master-Slave architecture
- Master: JobTracker
  - Accepts MR jobs submitted by users
  - Assigns Map and Reduce tasks to Tasktrackers
  - Monitors task and tasktracker status, re-executes tasks upon failure
- Slaves: Tasktrackers
  - Run Map and Reduce tasks upon instruction from the Jobtracker
  - Manage storage and transmission of intermediate output

Map-Reduce Architecture



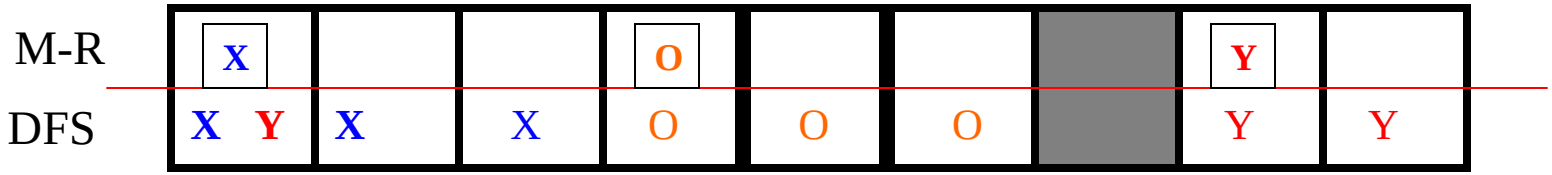
Hadoop HDFS + MR cluster



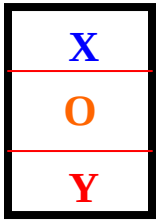
Machines with Datanodes and Tasktrackers

Hadoop: Two Services in One

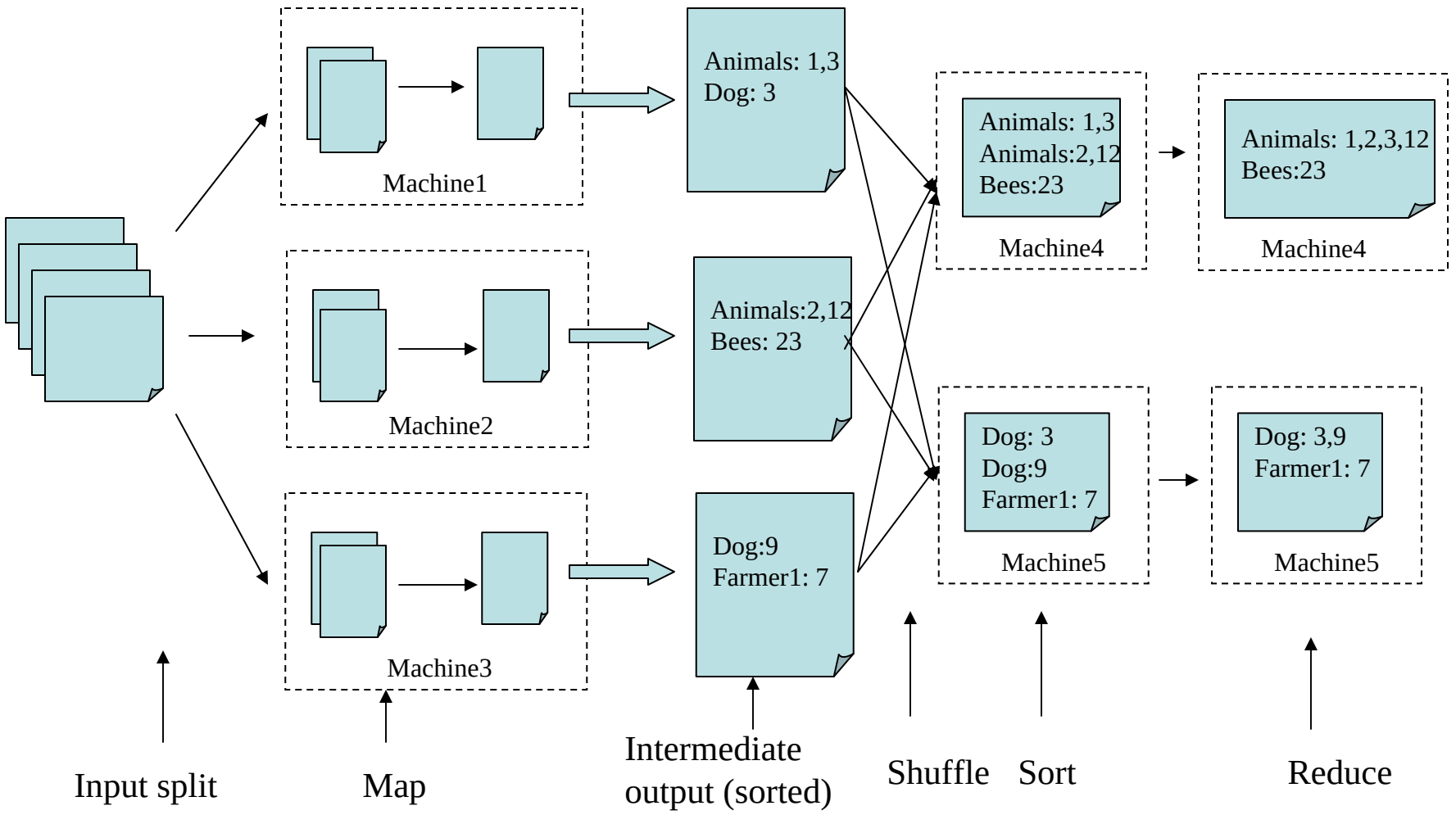
Cluster Nodes run both DFS and M-R



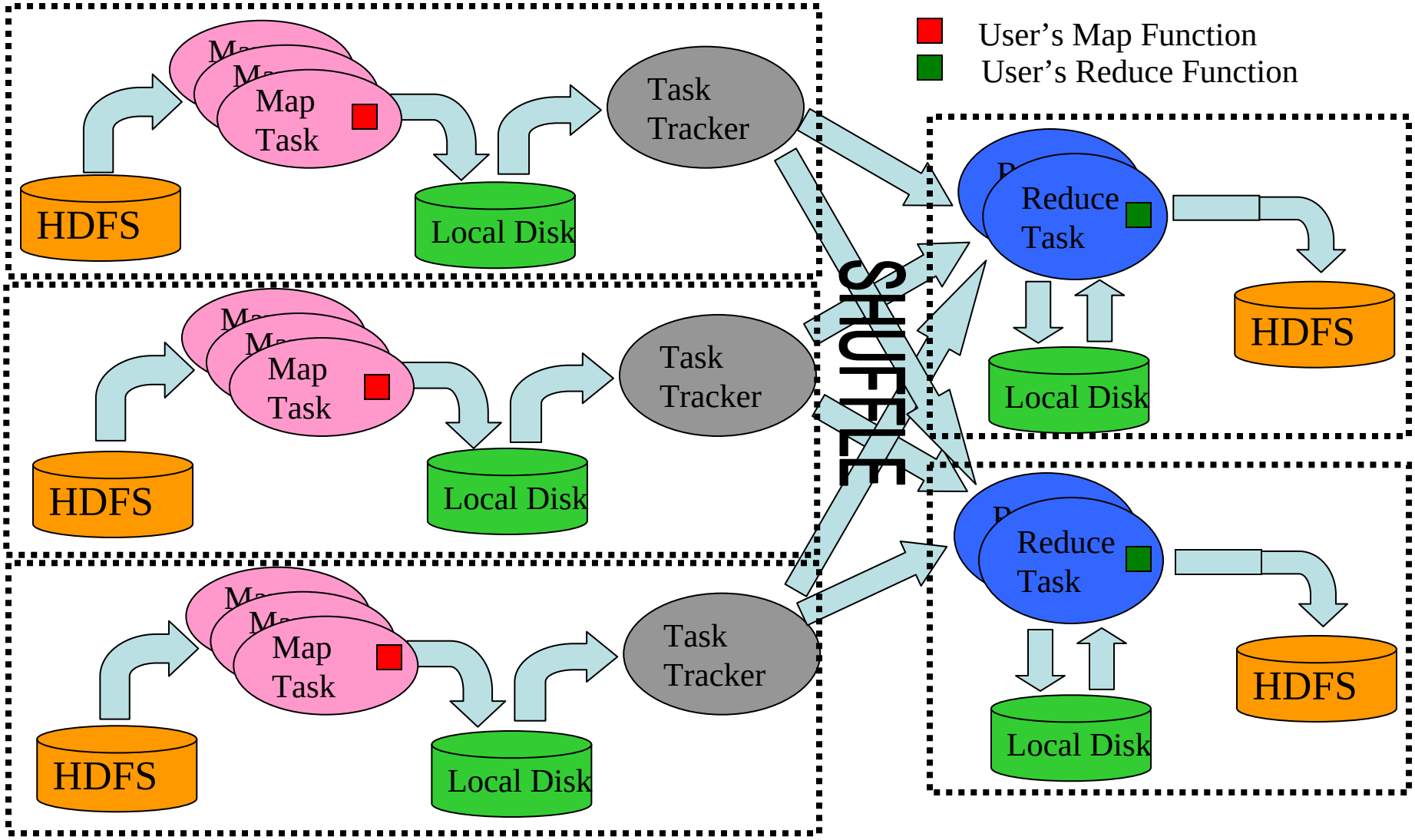
Input File
(128MB blocks)



Our Map-Reduce example and…



… its flow as a Hadoop Job



Reference links

- http://hadoop.apache.org/ - The main Apache site
  - Mailing lists, the code, documentation and more
- http://developer.yahoo.com/blogs/hadoop - Our blog
  - Reports, videos,… More on the way
- http://wiki.apache.org/hadoop/PoweredBy
  - A list of users, please add yourself!
- http://wiki.apache.org/hadoop/ProjectSuggestions
  - Ideas for folks who want to get started

Interesting news article on Hadoop

- http://www.bbc.co.uk/news/technology-11313194
  - 16 September 2010. Pi record smashed as team finds two-quadrillionth digit

Nicholas Sze, of tech firm Yahoo, said that when pi is expressed in binary, the two quadrillionth "bit" is 0. Mr. Sze used Yahoo's Hadoop cloud computing technology to more than double the previous record. It took 23 days on 1,000 of Yahoo's computers - on a standard PC, the calculation would have taken 500 years…

Outline

- Large Scale Computing @ Yahoo!
- Grid Computing as a solution
- Map-Reduce paradigm for large-scale computing
- Apache Hadoop (Map-Reduce Framework)
- Q & A