HEIDELBERG LAUREATE FORUM

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY Dharwad ज्ञानेन विकासः

acm India Council

Pingala Interactions In Computing

Distributed Computing Systems

Dr. Animesh Chaturvedi

Assistant Professor: IIIT Dharwad

Young Researcher: Pingala Interaction in Computing

Young Researcher: Heidelberg Laureate Forum

Postdoc: King's College London & The Alan Turing Institute

PhD: IIT Indore MTech: IIITDM Jabalpur

KING'S College LONDON

The Alan Turing Institute

Indian Institute of Technology Indore
भारतीय प्रौद्योगिकी संस्थान इंदौर

PDPM
Indian Institute of Information Technology, Design and Manufacturing, Jabalpur

Distributed File System

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?
  - Network bisection bandwidth is limited
  - 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) for Google's MapReduce
  - HDFS (Hadoop Distributed File System) for Hadoop

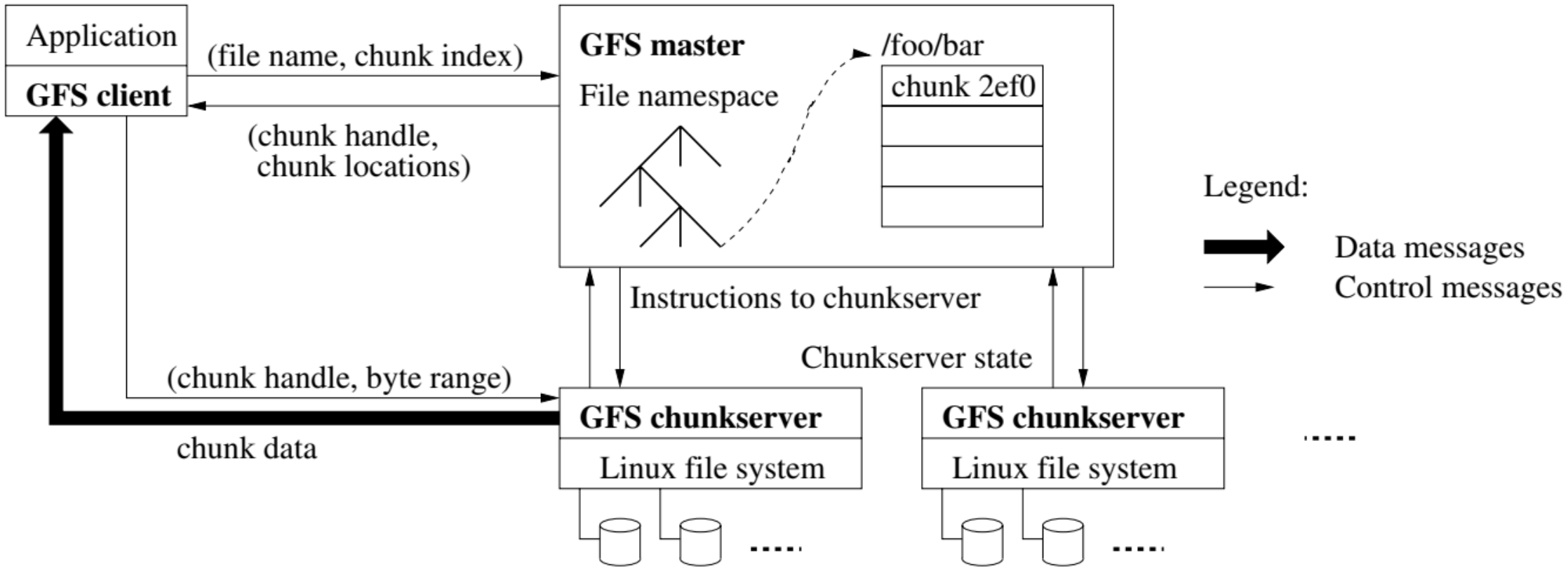
Big Files to Google File System (GFS)

- Earlier Google effort, "BigFiles", developed by Larry Page and Sergey Brin.
  - Supervisors: Hector Garcia-Molina, Rajeev Motwani, Jeff Ullman, and Terry Winograd
- "Big File" was regenerated as "Google File System" by Sanjay Ghemawat, et al.
- Google File System (GFS)
  - "It is widely deployed within Google as the storage platform for the generation and processing of data used by Google service as well as research and development efforts that require large data sets." 2003
  - "The largest cluster to date provides hundreds of terabytes of storage across thousands of disks on over a thousand machines, and it is concurrently accessed by hundreds of clients." 2003

http://infolab.stanford.edu/~backrub/google.html

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. "The Google file system." Proceedings of the nineteenth ACM symposium on Operating systems principles. 2003.

Google File System (GFS)



https://en.wikipedia.org/wiki/Google_File_System

https://sites.google.com/site/gfsassignmentwiki/home

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. "The Google file system." Proceedings of the nineteenth ACM symposium on Operating systems principles. 2003.

GFS: Assumptions

- Choose commodity hardware over "exotic" hardware
  - Scale "out", not "up"
- High component failure rates
  - Inexpensive commodity components fail all the time
- "Modest" number of huge files
  - Multi-gigabyte files are common, if not encouraged
- Files are write-once, mostly appended to
  - Perhaps concurrently
- Large streaming reads over random access
  - High sustained throughput over low latency

https://en.wikipedia.org/wiki/Google_File_System
https://sites.google.com/site/gfsassignmentwiki/home
Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. "The Google file system." Proceedings of the nineteenth ACM symposium on Operating systems principles. 2003.

GFS: Design Decisions

- Files stored as chunks
  - Fixed size (64MB)
- Reliability through replication
  - Each chunk replicated across 3+ chunkservers
- Single master to coordinate access, keep metadata
  - Simple centralized management
- No data caching
  - Little benefit due to large datasets, streaming reads
- Simplify the API
  - Push some of the issues onto the client (e.g., data layout)

HDFS = GFS clone (same basic ideas implemented in Java)

GFS to HDFS

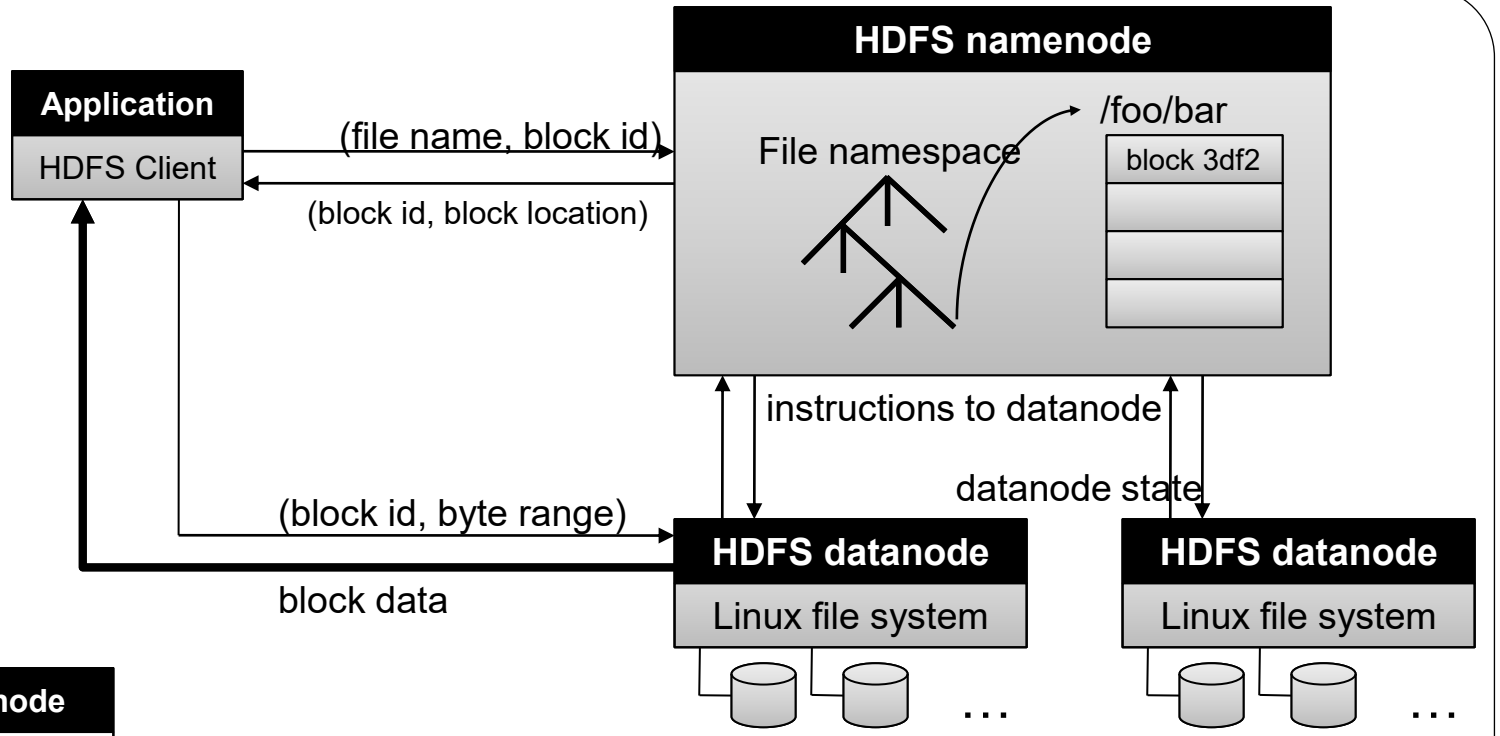
- Google File System (GFS) has similar open-source
  - "Hadoop Distributed File System (HDFS)"
- GFS and HDFS are distributed computing environment to process "Big Data".
- GFS and HDFS are not implemented in the kernel of an operating system, but they are instead provided as a userspace library.
- GFS and HDFS properties
  - Files are divided into fixed-size chunks of 64 megabytes.
  - Scalable distributed file system for large distributed data intensive applications.
  - Provides fault tolerance.
  - High aggregate performance to a large number of clients.

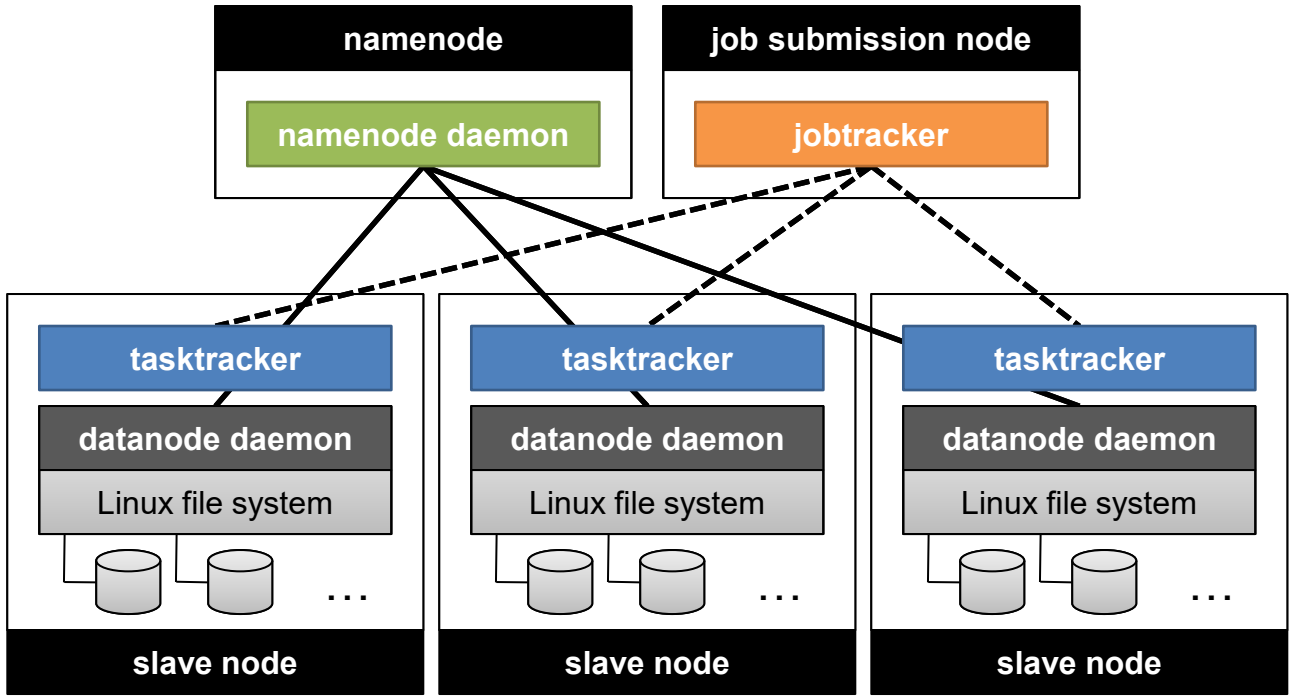
https://en.wikipedia.org/wiki/Google_File_System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. "The Google file system." Proceedings of the nineteenth ACM symposium on Operating systems principles. 2003.

GFS to HDFS

- GFS master
  - Hadoop namenode
- GFS chunkservers
  - Hadoop datanodes

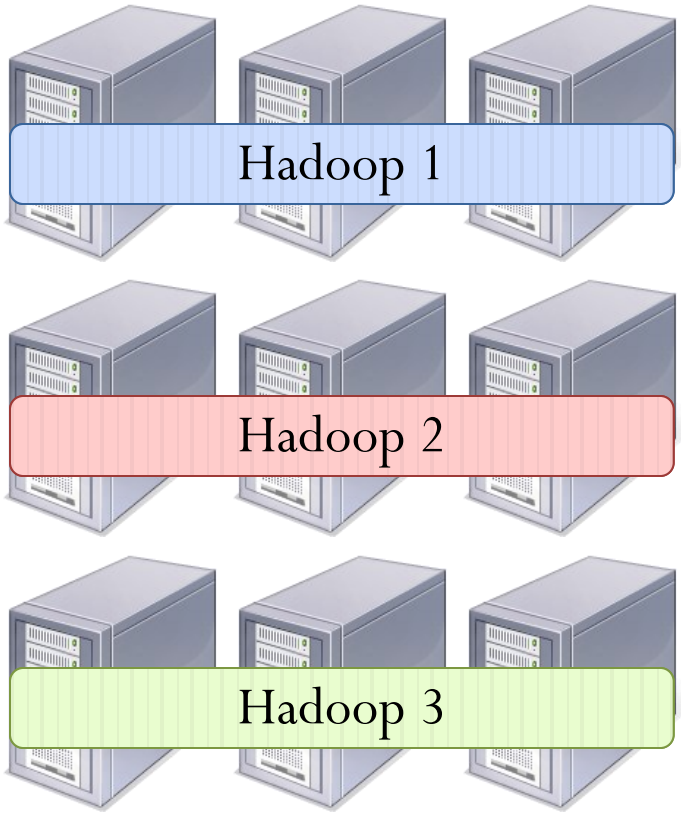




Coarse-grained sharing

Option: Coarse-grained sharing

- Give framework a (slice of) machine for its entire duration



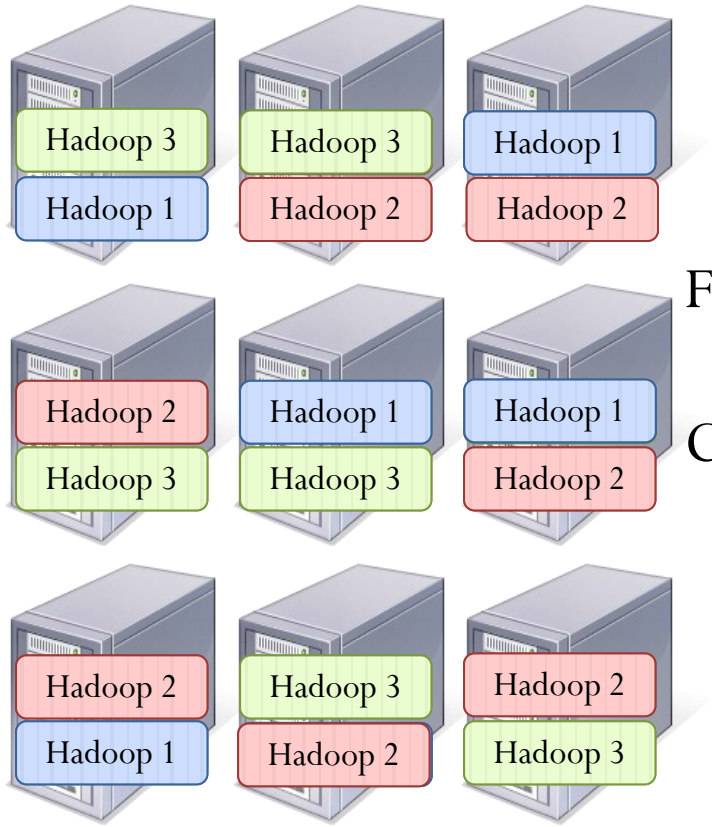
Data locality compromised if machine held for long time

Hard to account for new frameworks and changing demands

→ **hurts utilization and interactivity**

Fine-grained sharing

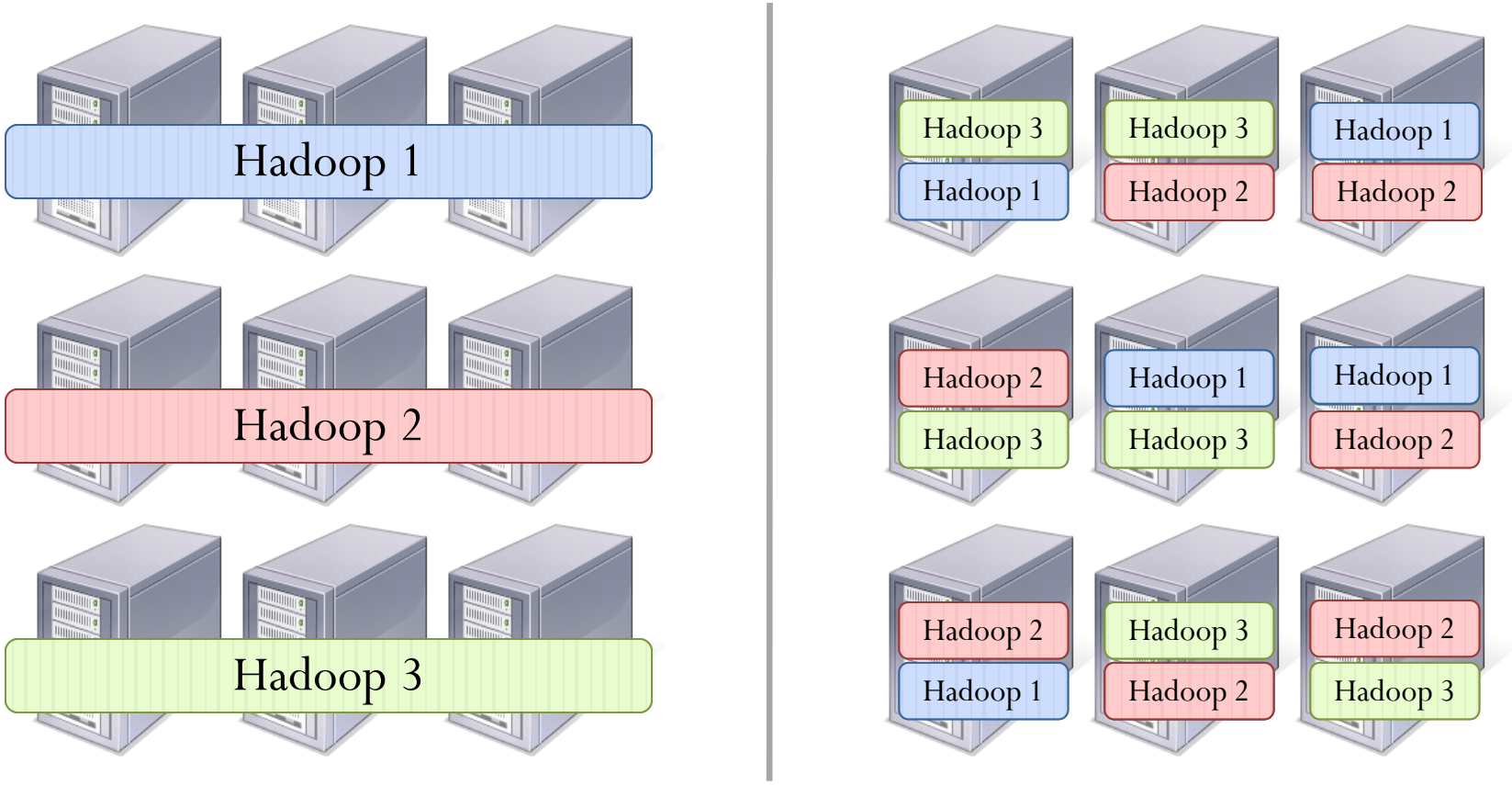
- Support frameworks that use smaller *tasks* (in time and space) by multiplexing them across all available resources



Frameworks can take turns accessing data on each node

Can resize frameworks shares to get utilization & interactivity

Multiple Hadoops Experiment



Big Data

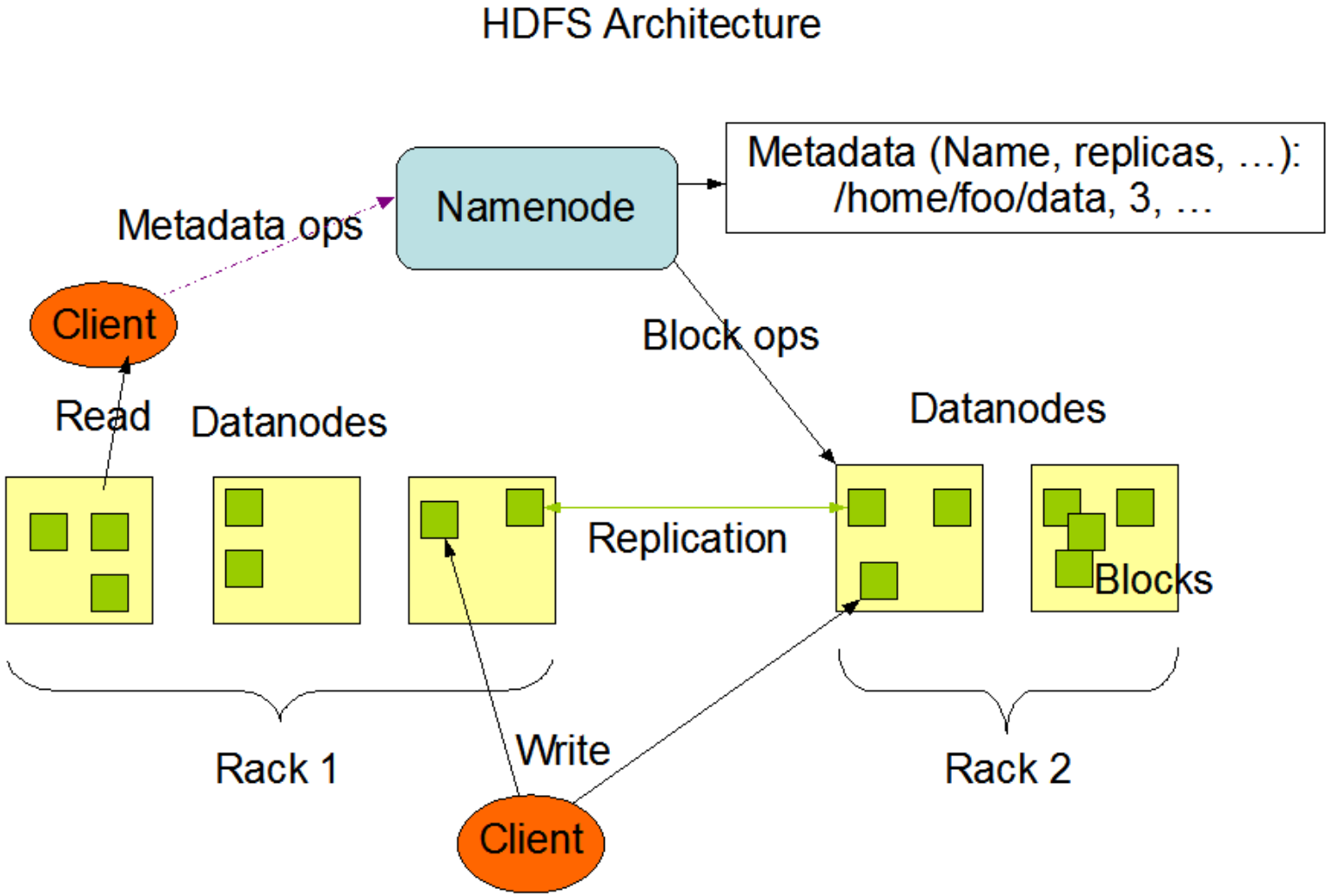
- Big data can be described by the following characteristics:
  - Volume: size large than terabytes and petabytes
  - Variety: type and nature, structured, semi-structured or unstructured
  - Velocity: speed of generation and processing to meet the demands
  - Veracity: the data quality and the data value
  - Value: Useful or not useful
- The main components and ecosystem of Big Data
  - Data Analytics: data mining, machine learning and natural language processing etc.
  - Technologies: Business Intelligence, Cloud computing & Databases etc.
  - Visualization: Charts, Graphs etc.

Hadoop

- “The Apache Hadoop project develops open source software for reliable, scalable, distributed computing.” Software library and a framework.
- Created by Doug Cutting Named on his son's stuffed elephant
- For distributed processing of large data sets across clusters of computers using simple programming models.
- Locality of reference
- Scalability: Scale up from single servers to thousands of machines,
  - Each offering local computation and storage
  - Program remains same for 10, 100, 1000,… nodes
  - Corresponding performance improvement
- Fault-tolerant file system:
  - Detect and handle failures and Delivering a highly-available.
- Hadoop Distributed File System (HDFS) Modeled on Google File system
- MapReduce for Parallel computation using
- Components – Pig, Hbase, HIVE, ZooKeeper

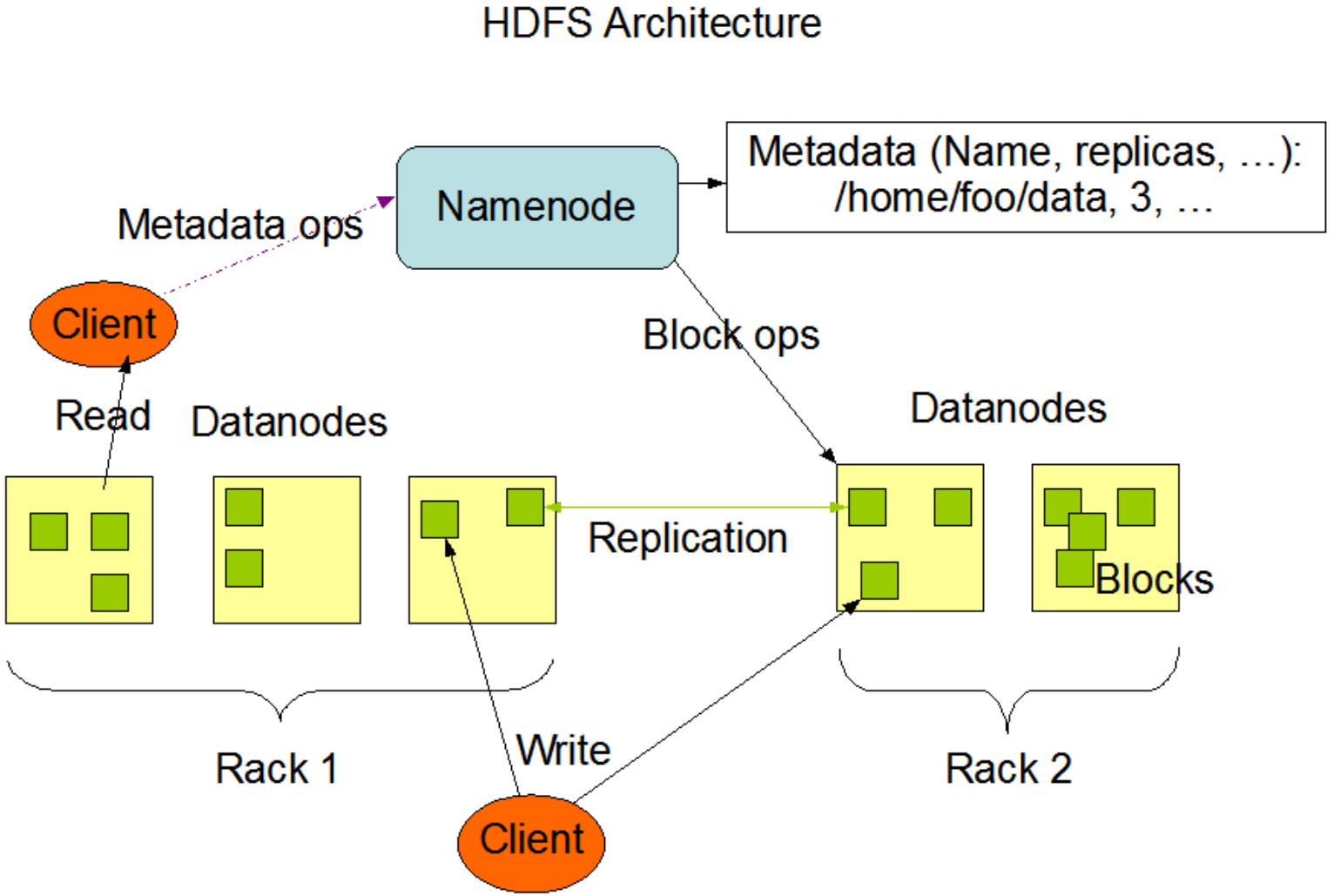
Hadoop Distributed File System (HDFS)

- HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients.
- In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on.
- HDFS exposes a file system namespace and allows user data to be stored in files.



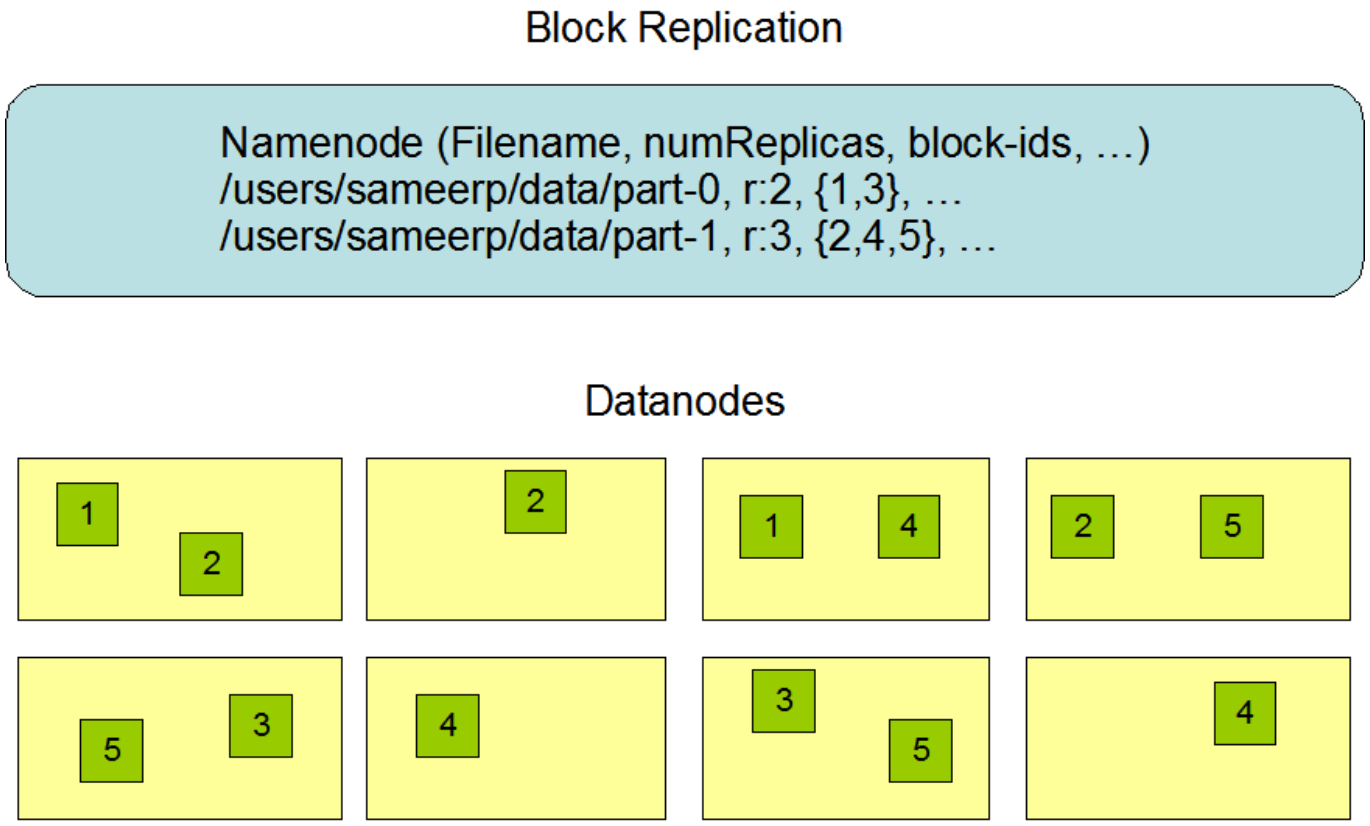
Hadoop Distributed File System (HDFS)

- Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes.
- The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes.
- The DataNodes are responsible for serving read and write requests from the file system's clients.
- The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode



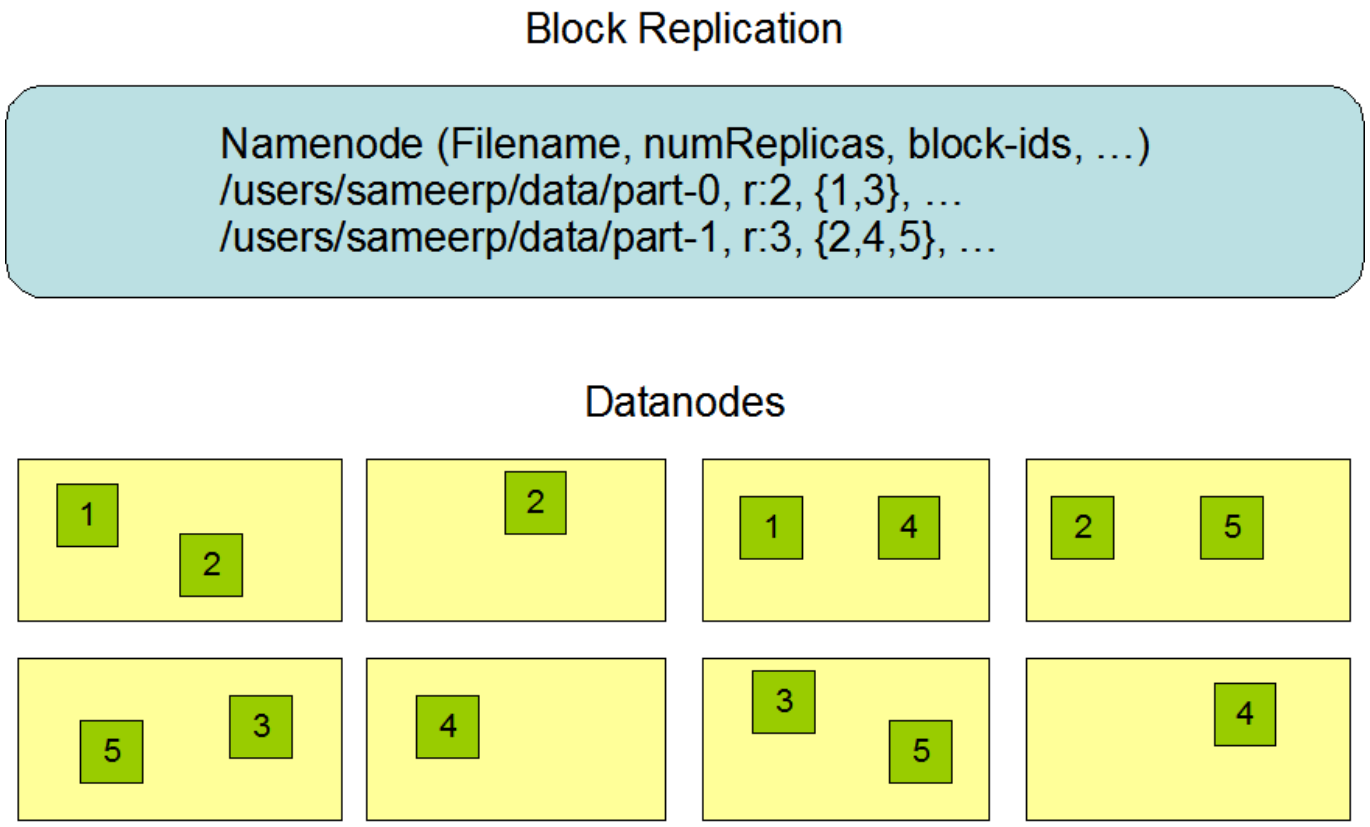
Namenode Responsibilities

- Manages File System namespace
  - mapping files to blocks and blocks to data nodes.
  - Maintains status of data nodes
  - holds file/directory structure, metadata, file-to-block mapping, access permissions, etc.
- Maintaining overall health:
  - Periodic communication with the Datanodes
  - Block re-replication and rebalancing
  - Garbage collection
- Heartbeat: Datanode sends heartbeat at regular intervals, if heartbeat is not received, Datanode is declared to be dead
- Coordinating file operations:
  - Directs clients to Datanodes for reads and writes
  - No data is moved through the Namenode
- Blockreport: Datanode sends list of blocks on it. Used to check health of HDFS

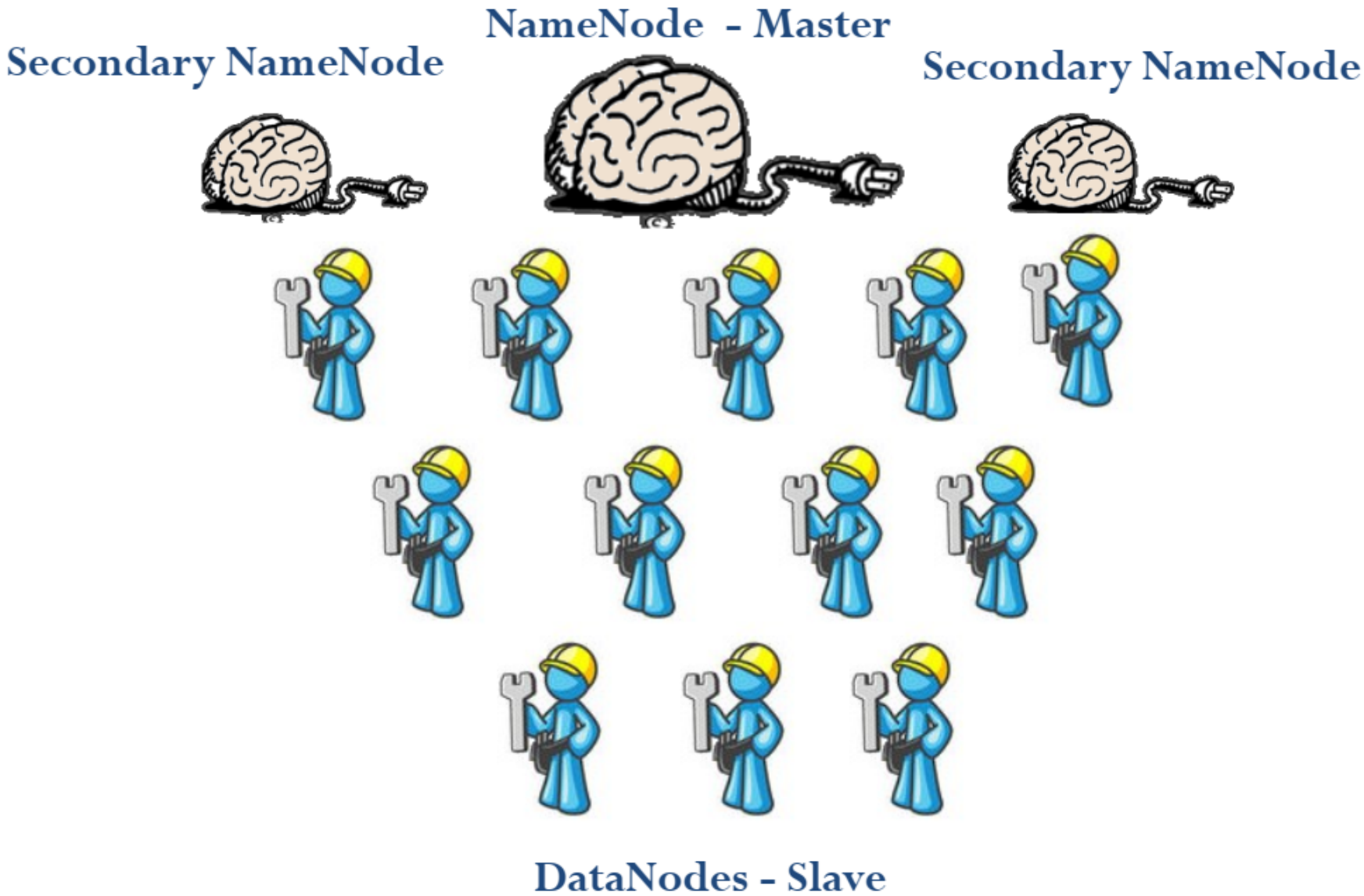


Datanodes: Responsibilities

- Replicates
  - On Datanode failure,
  - On Disk failure,
  - On Block corruption
- Data integrity
  - Checksum for each block,
  - Stored in hidden file
- Rebalancing - balancer tool
  - Provisioning: addition of new nodes,
  - Decommissioning: remove node,
  - Deletion of some files



HDFS Force


NameNode - Master
Secondary NameNode
Secondary NameNode
DataNodes - Slave

Map Reduce

MapReduce is a programming model

- An implementation for processing and generating large data sets.
- Many real world tasks are expressible in this model.
- Users specify
  - a map function that processes a key/value pair to generate a set of intermediate key/value pairs,
  - a reduce function that merges all intermediate values associated with the same intermediate key.
- MapReduce computation processes
  - many terabytes of data
  - on thousands of machines.
- Runs on a large cluster of commodity machines and is highly scalable.

Jeffrey Dean, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." *Communications of the ACM* 51.1 (2008): 107-113.

MapReduce for programmer

- Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines.
- Programmers
  - find it easy to use,
  - easily utilize the resources of a large distributed system,
  - without any experience with parallel and distributed systems
- Hundreds of MapReduce programs have been implemented
- Thousands of MapReduce jobs are executed on Google's clusters every day.
- Takes care of
  - partitioning the input data,
  - scheduling the program's execution across a set of machines,
  - handling machine failures, and
  - managing the required inter-machine communication.

Jeffrey Dean, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." *Communications of the ACM* 51.1 (2008): 107-113.

Typical Large-Data Problem

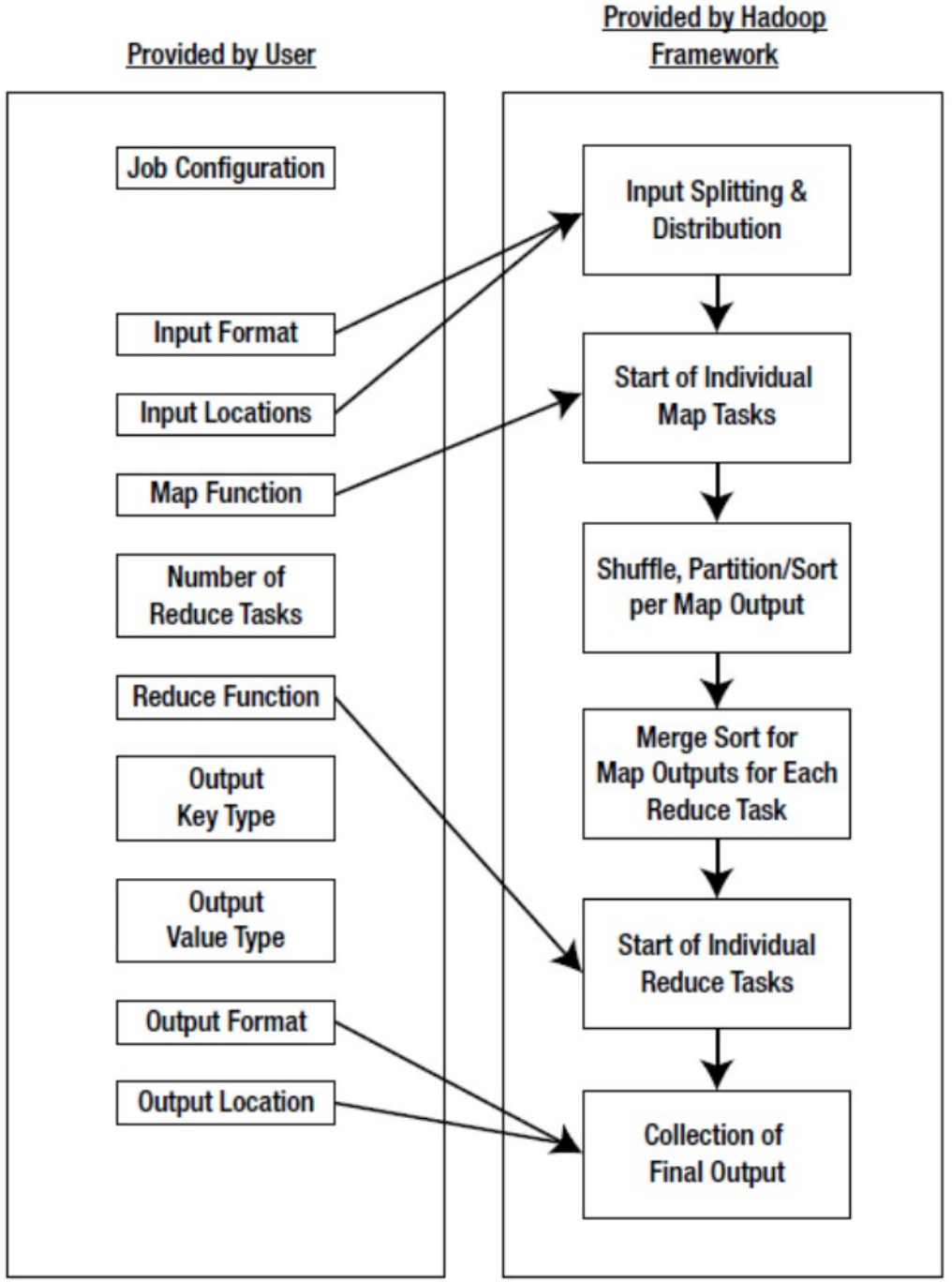
- Iterate over a large number of records **Map**
- Extract something of interest from each
- Shuffle and sort intermediate results
- Aggregate intermediate results **Reduce**
- Generate final output

Key idea: provide a functional abstraction for these two operations – MapReduce

Jeffrey Dean, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." *Communications of the ACM* 51.1 (2008): 107-113.

Map Reduce

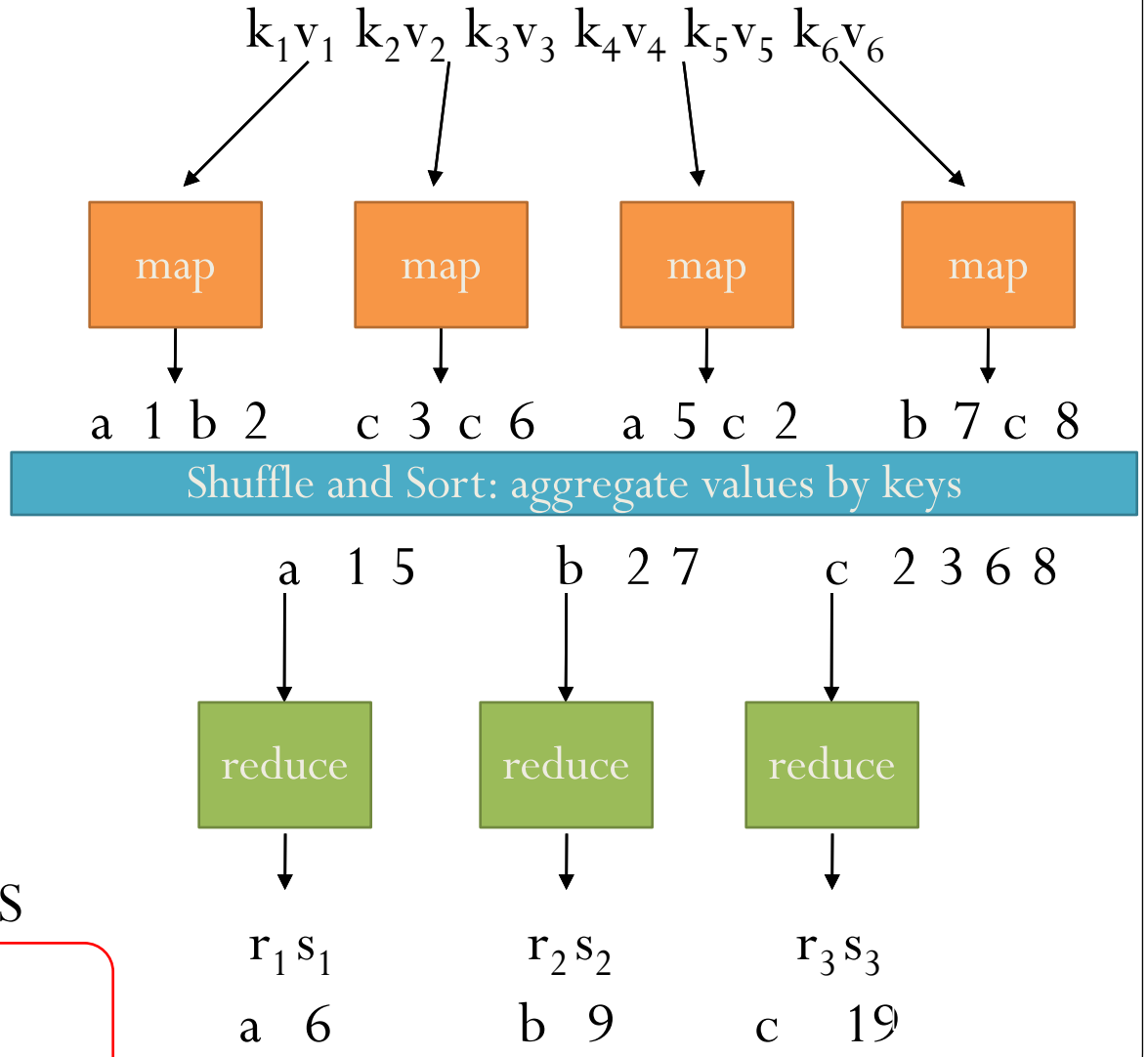
- Format of input- output (key, value)
  - Map: (k1, v1) → list (k2, v2)
  - Reduce: (k2, list v2) → list (k3, v3)
  - All values with the same key are sent to the same reducer
- The execution framework handles everything else
- MapReduce will not work for
  - Inter-process communication
  - Data sharing required
  - Example: Recursive functions



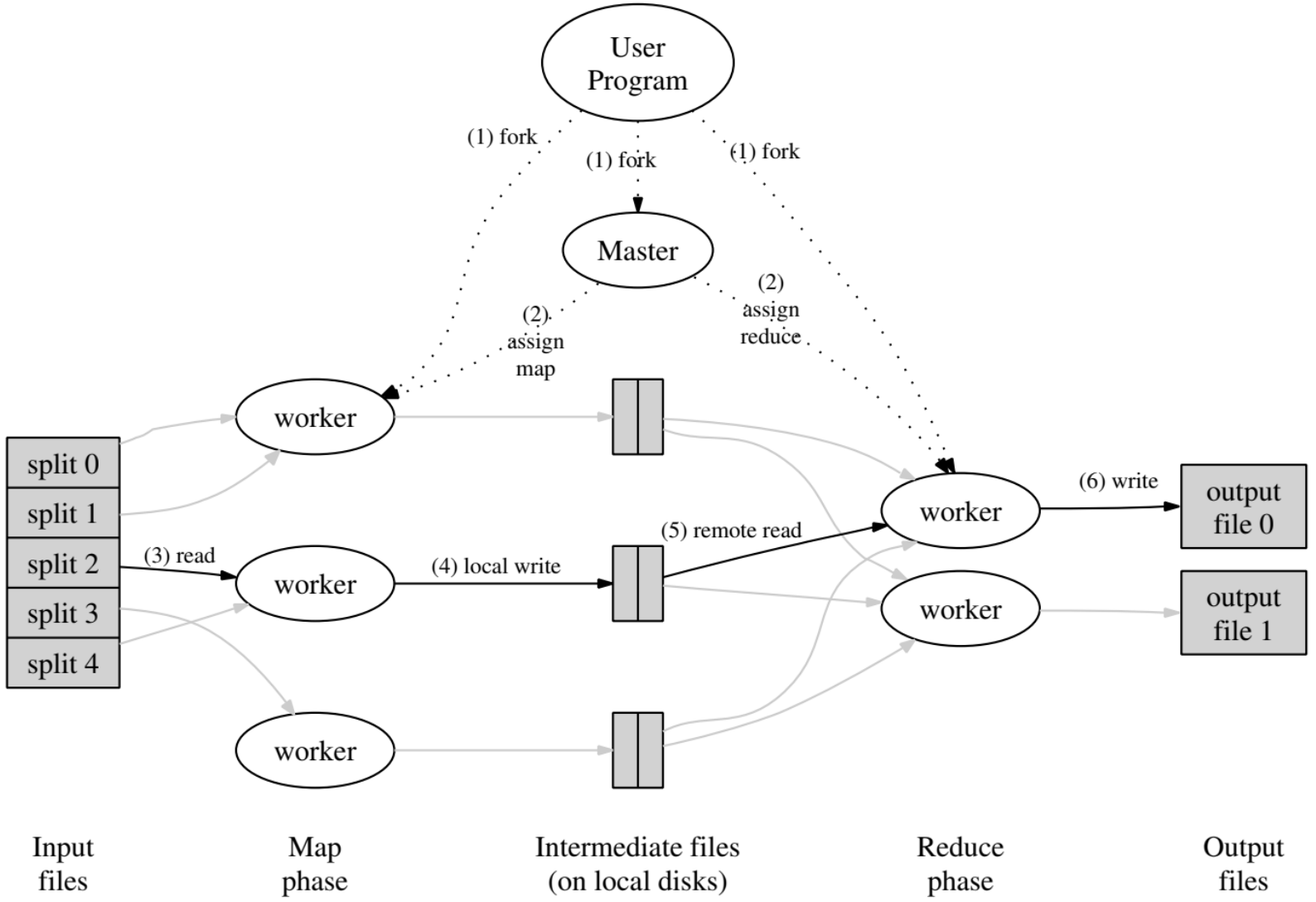
MapReduce Runtime

- Handles scheduling
  - Assigns workers to map and reduce tasks
- Handles "data distribution"
  - Moves processes to data
- Handles synchronization
  - Gathers, sorts, and shuffles intermediate data
- Handles errors and faults
  - Detects worker failures and automatically restarts
- Handles speculative execution
  - Detects "slow" workers and re-executes work
- Everything happens on top of a Distributed FS

Sounds simple, but many challenges!



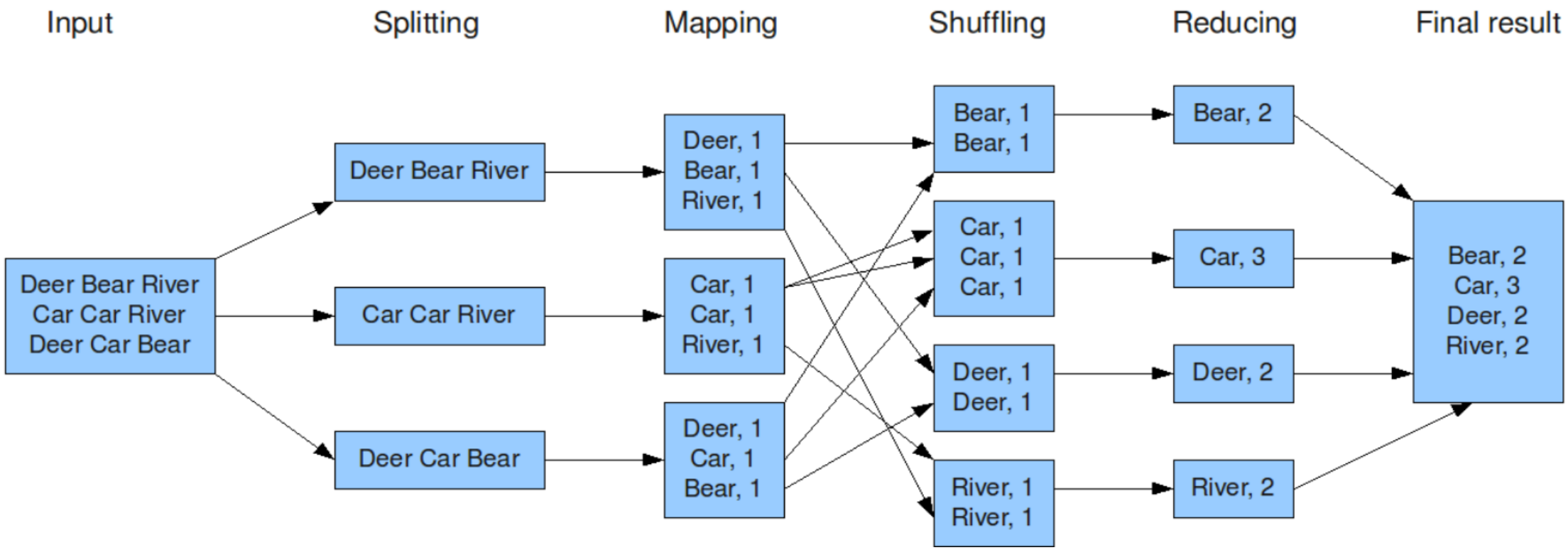
MapReduce Overall Architecture



Jeffrey Dean, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." *Communications of the ACM* 51.1 (2008): 107-113.

Word Count

- **Map:** Input lines of text to breaks them into words gives outputs for each word <key = word, value =1 >
- **Reduce:** Input <word, 1> output <word, + value>



“Hello World” Example: Word Count

```
map(String key, String value):
  // key: document name
  // value: document contents
  for each word w in value:
    EmitIntermediate(w, "1");

reduce(String key, Iterator values):
  // key: a word
  // values: a list of counts
  int result = 0;
  for each v in values:
    result += ParseInt(v);
  Emit(AsString(result));
```

Jeffrey Dean, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." *Communications of the ACM* 51.1 (2008): 107-113.

MapReduce Implementations

- Google has a proprietary implementation in C++
  - Bindings in Java, Python
- Hadoop is an open-source implementation in Java
  - Development led by Yahoo, used in production
  - Now an Apache project
  - Rapidly expanding software ecosystem, but still lots of room for improvement
- Lots of custom research implementations issues

Map Reduce/GFS Summary

- Simple, but powerful programming model
- Scales to handle petabyte+ workloads
  - Google: six hours and two minutes to sort 1PB (10 trillion 100-byte records) on 4,000 computers
  - Yahoo!: 16.25 hours to sort 1PB on 3,800 computers
- Incremental performance improvement with more nodes
- Seamlessly handles failures, but possibly with performance penalties

References

- Cloud Computing: Past, Present, and Future, Professor Anthony D. Joseph, UC Berkeley Reliable Adaptive Distributed systems Lab (RAD lab) UC Berkley http://abovetheclouds.cs.berkeley.edu/
- https://en.wikipedia.org/wiki/Google_File_System
- https://sites.google.com/site/gfsassignmentwiki/home
- Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. "The Google file system." Proceedings of the nineteenth ACM symposium on Operating systems principles. 2003.
- Jeffrey Dean, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." *Communications of the ACM* 51.1 (2008): 107-113.

תודה רבה
Hebrew

Ευχαριστώ
Greek

Спасибо
Russian

Danke
German

धन्यवादः
Sanskrit

Merci
French

ধন্যবাদ
Bangla

நன்றி
Tamil

شكراً
Arabic

ಧನ್ಯವಾದಗಳು
Kannada

Thank You
English

നന്ദി
Malayalam

多謝
Traditional Chinese

Grazie
Italian

ధన్యవాదాలు
Telugu

આભાર
Gujarati

ਧੰਨਵਾਦ
Punjabi

धन्यवाद
Hindi & Marathi

多谢
Simplified Chinese

Gracias
Spanish

https://sites.google.com/site/animeshchaturvedi07

Obrigado
Portuguese

ありがとうございました
Japanese

ขอบคุณ
Thai

감사합니다
Korean