SNIA

Education

Big Data Storage Options for Hadoop

Sam Fineberg, HP Storage

SNIA Legal Notice

Abstract

- Big Data Storage Options for Hadoop
  - The Hadoop system was developed to enable the transformation and analysis of vast amounts of structured and unstructured information. It does this by implementing an algorithm called MapReduce across compute clusters that may consist of hundreds or even thousands of nodes. In this presentation Hadoop will be looked at from a storage perspective. The tutorial will describe the key aspects of Hadoop storage, the built-in Hadoop file system (HDFS), and some other options for Hadoop storage that exist in the commercial and open source communities.

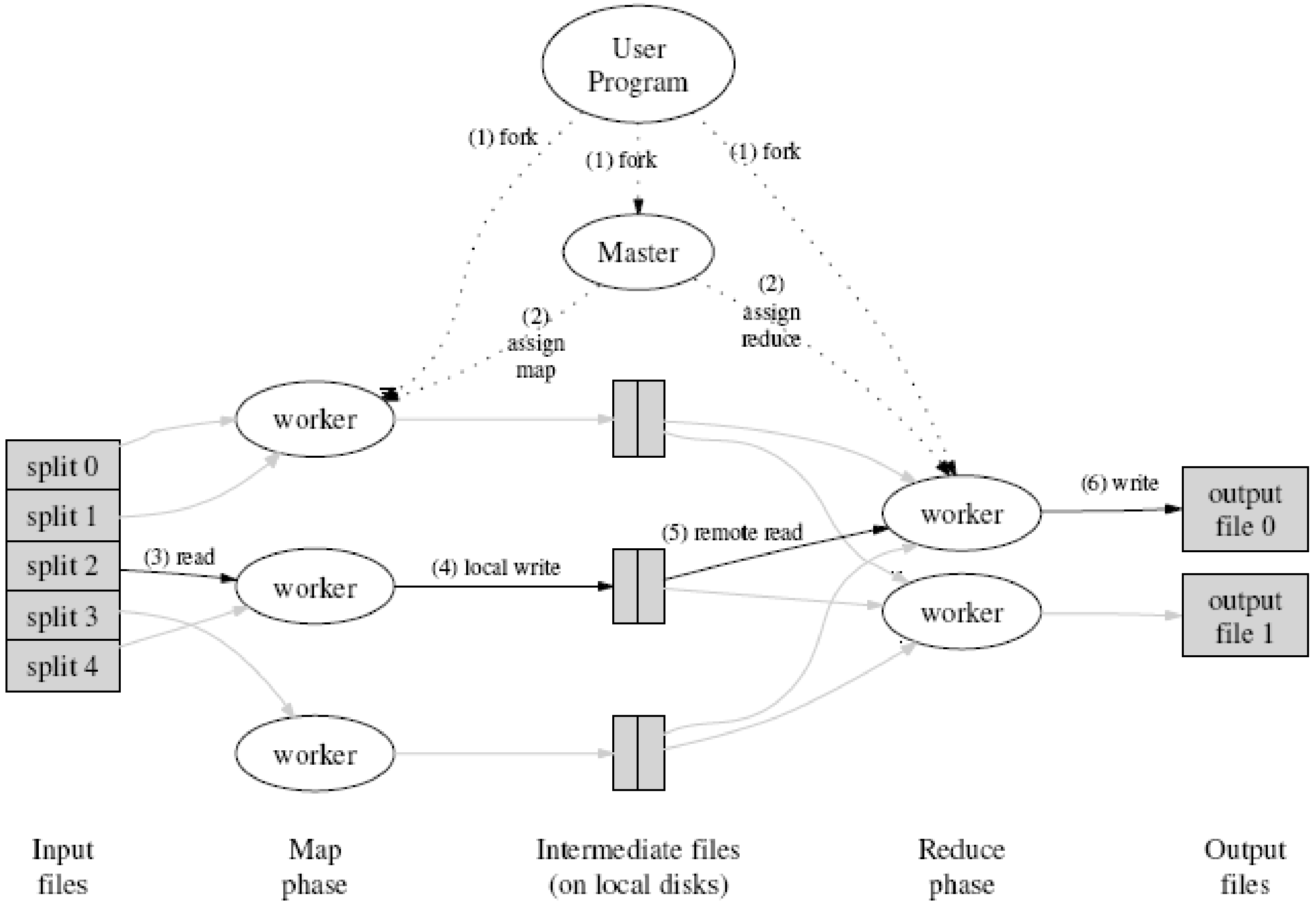
What is Hadoop?

- A scalable fault-tolerant distributed system for data storage and processing
- Core Hadoop has two main components
  - MapReduce: fault-tolerant distributed processing
    - Programming model for processing sets of data
    - Mapping inputs to outputs and reducing the output of multiple Mappers to one (or a few) answer(s)
  - Hadoop Distributed File System (HDFS): self-healing, high-bandwidth clustered storage
    - Reliable, redundant, distributed file system optimized for large files
- Operates on unstructured and structured data
- A large and active ecosystem
- Written in Java
- Open source under the friendly Apache License
  - http://wiki.apache.org/hadoop/

What is MapReduce?

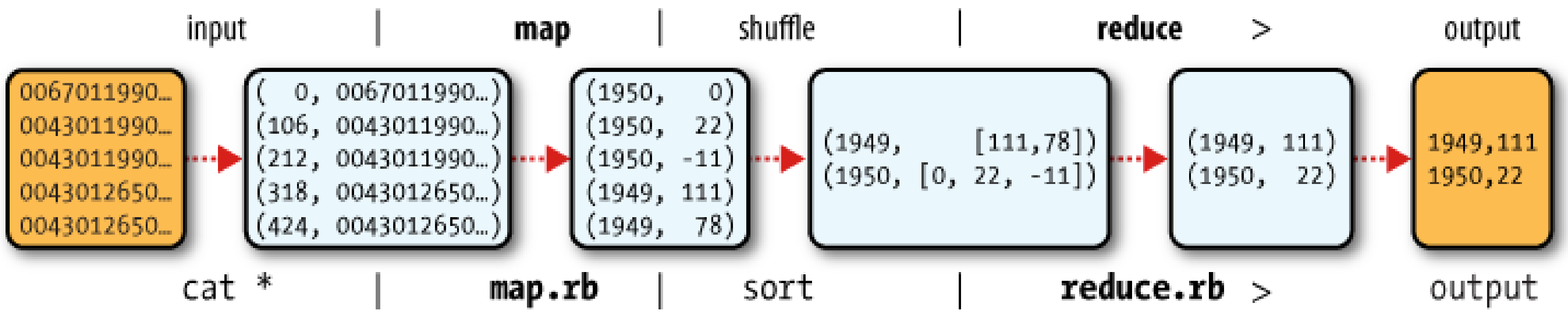
- A method for distributing a task across multiple nodes
- Each node processes data stored on that node
- Consists of two developer-created phases
  1. Map
  2. Reduce
- In between Map and Reduce is the Shuffle and Sort

MapReduce



© Google, from Google Code University, http://code.google.com/edu/parallel/mapreduce-tutorial.html

MapReduce Operation

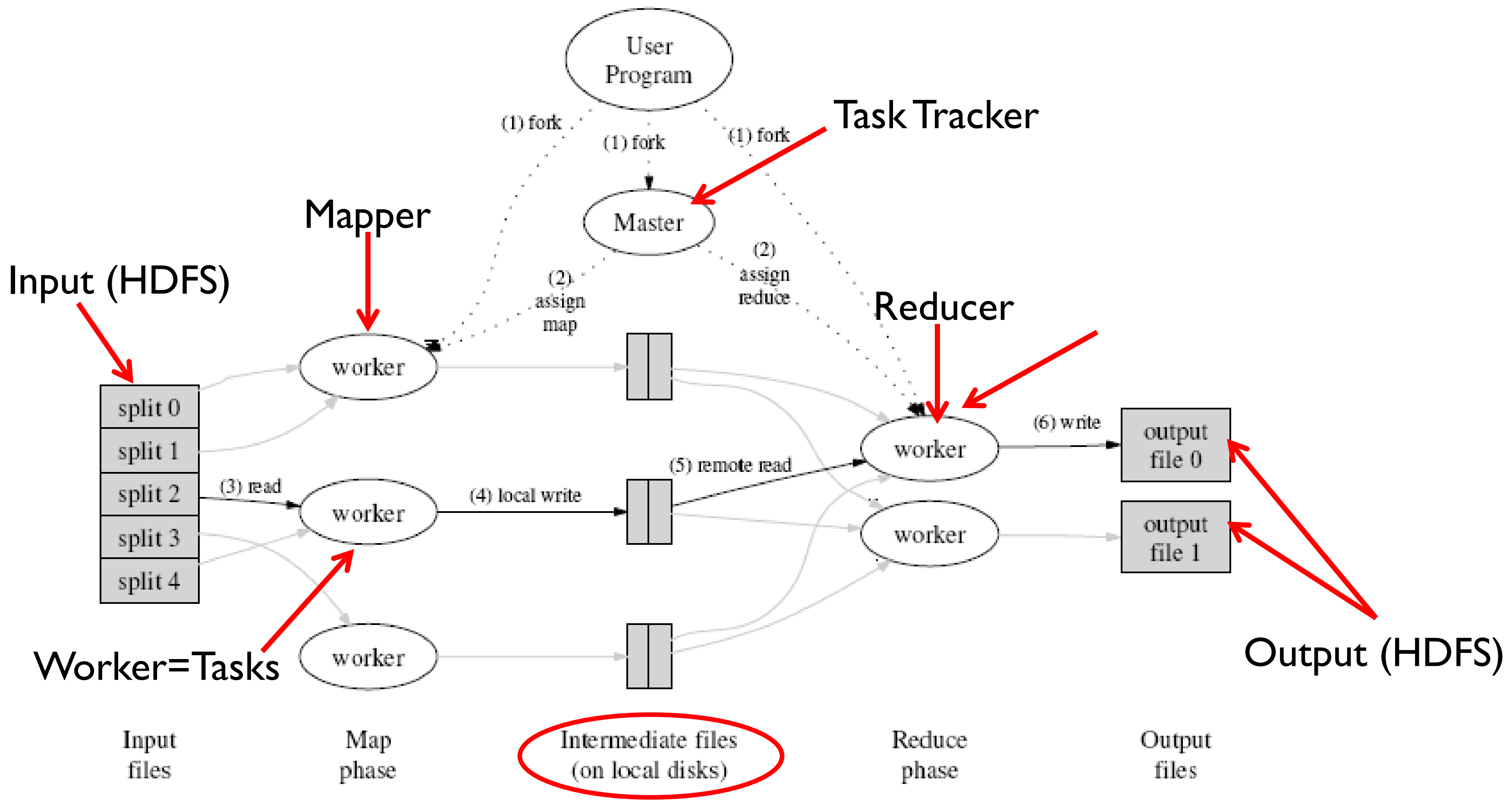


What was the max temperature for the last century?

Key MapReduce Terminology Concepts

- A user runs a client program (typically a Java application) on a client computer
- The client program submits a job to Hadoop
- The job is sent to the JobTracker process on the Master Node
- Each Slave Node runs a process called the TaskTracker
- The JobTracker instructs TaskTrackers to run and monitor tasks
- A task attempt is an instance of a task running on a slave node
- There will be at least as many task attempts as there are tasks which need to be performed

MapReduce in Hadoop



© Google, from Google Code University, http://code.google.com/edu/parallel/mapreduce-tutorial.html

MapReduce: Basic Concepts

- Each Mapper processes single input split from HDFS
- Hadoop passes one record at a time to the developer's Map code
- Each record has a key and a value
- Intermediate data written by the Mapper to local disk
- During shuffle and sort phase, all values associated with same intermediate key are transferred to same Reducer
- Reducer is passed each key and a list of all its values
- Output from Reducers is written to HDFS

What is a Distributed File System?

- A distributed file system is a file system that allows access to files from multiple hosts across a network
  - A network filesystem (NFS/CIFS) is a type of distributed file system – more tuned for file sharing than distributed computation
  - Distributed computing applications, like Hadoop, utilize a *tightly coupled distributed file system*
- Tightly coupled distributed filesystems
  - Provide a *single global namespace* across all nodes
  - Support *multiple initiators*, *multiple disk nodes*, *multiple access to files* – file parallelism
  - Examples include HDFS, pNFS, as well as many commercial and research systems

Hadoop Distributed File System - HDFS

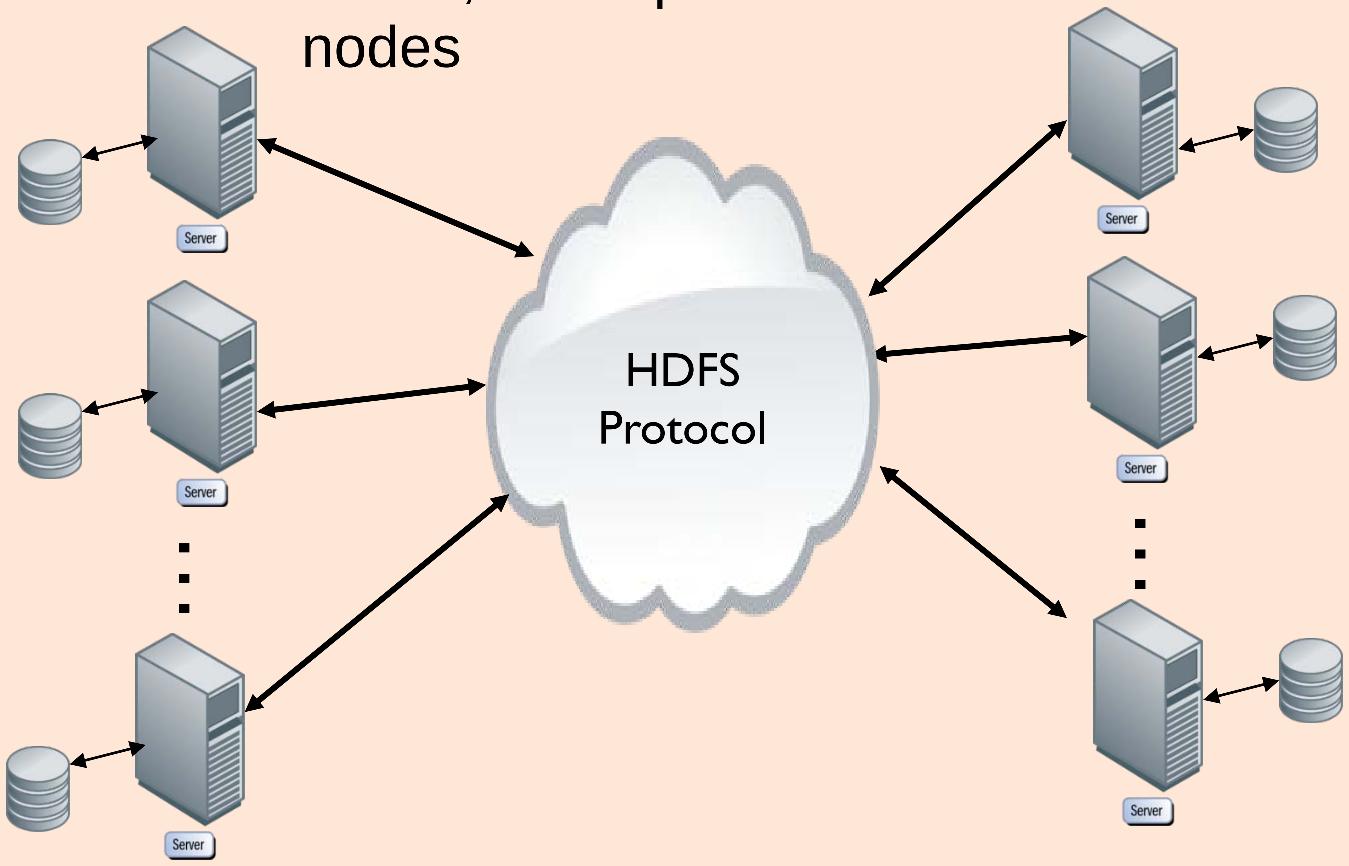
- Architecture
  - Java application, not deeply integrated with the server OS
    - Layered on top of a standard FS
    - Must use Hadoop or a special library to access HDFS files
  - Shared-nothing, all nodes have direct attached disks
  - Write once filesystem – must copy a file to modify it
- HDFS basics
  - Data is organized into files & directories
  - Files are divided into 64-128MB blocks, distributed across nodes
  - Block placement is handled by the “NameNode”
  - Placement coordinated with job tracker = writes always co-located, reads co-located with computation whenever possible
  - Blocks replicated to handle failure, replica blocks can be used by compute tasks
  - Checksums used to ensure data integrity
- Replication: one and only strategy for error handling, recovery and fault tolerance
  - Self Healing
  - Makes multiple copies (typically 3)

HDFS on local DAS

- A Hadoop cluster consisting of many nodes, each of which has local *direct attached storage* (DAS)
  - Disks are running a file system
  - HDFS blocks are stored as files in a special directory
  - Disks attached directly, for example, with SAS or SATA
  - No storage is shared, disks only attach to a single node
- The most common use case for Hadoop
  - Original design point for Hadoop/HDFS
  - Can work with cheap unreliable hardware
  - Some very large systems utilize this model

HDFS on local DAS

Compute nodes are part of HDFS, data spread across nodes



HDFS File Write Operation

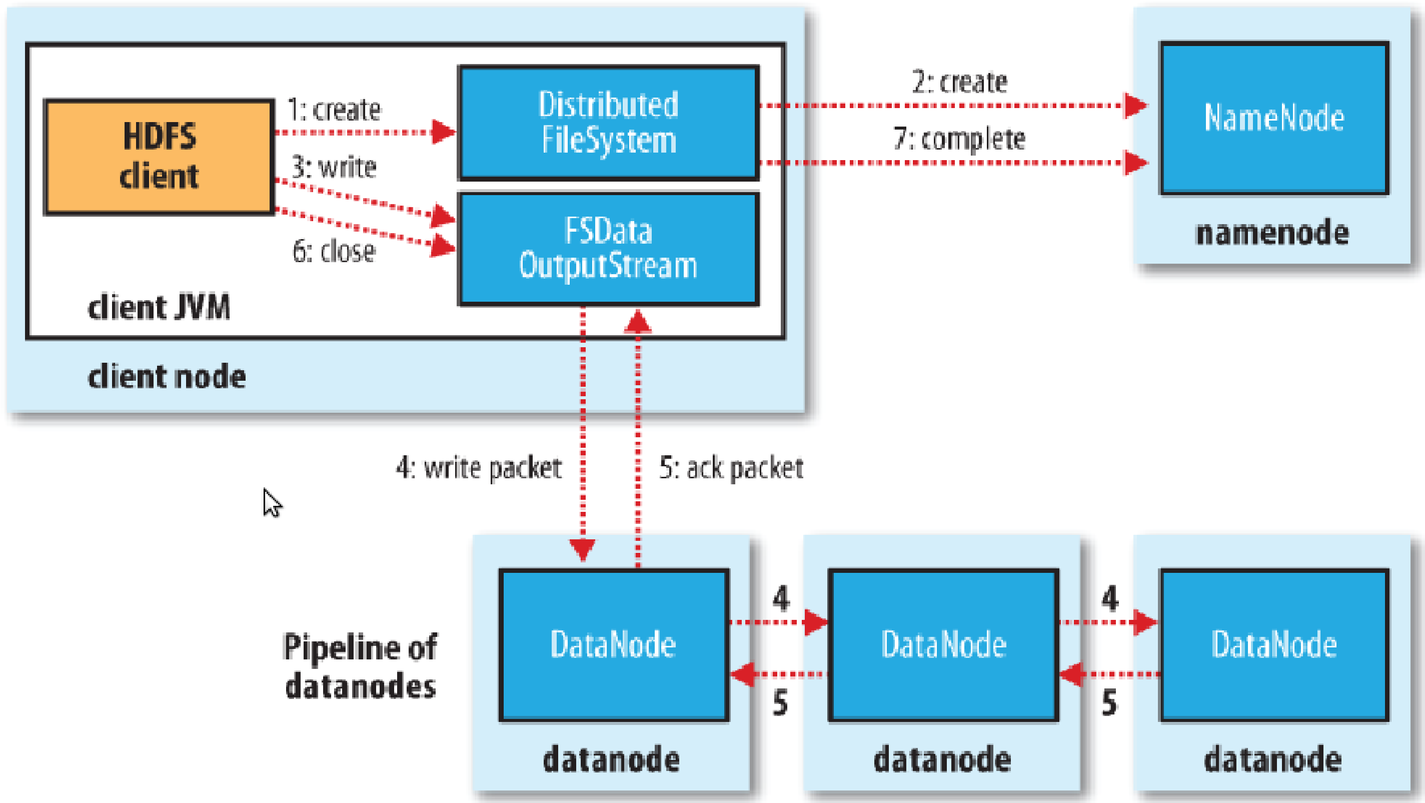


Image source: Hadoop, The Definitive Guide Tom White, O'Reilly

HDFS File Read Operation

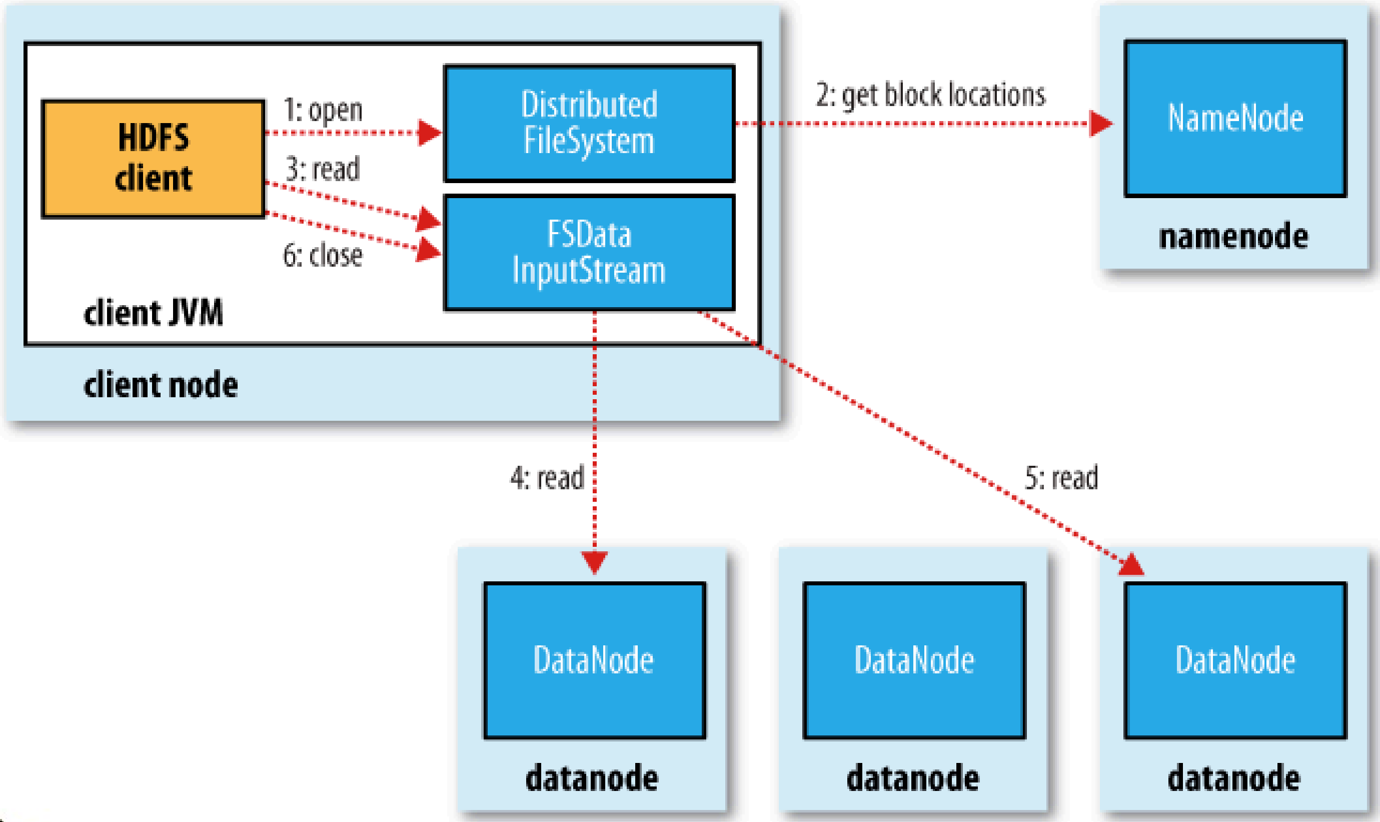


Image source: Hadoop, The Definitive Guide Tom White, O'Reilly

HDFS on local DAS - Pros and Cons

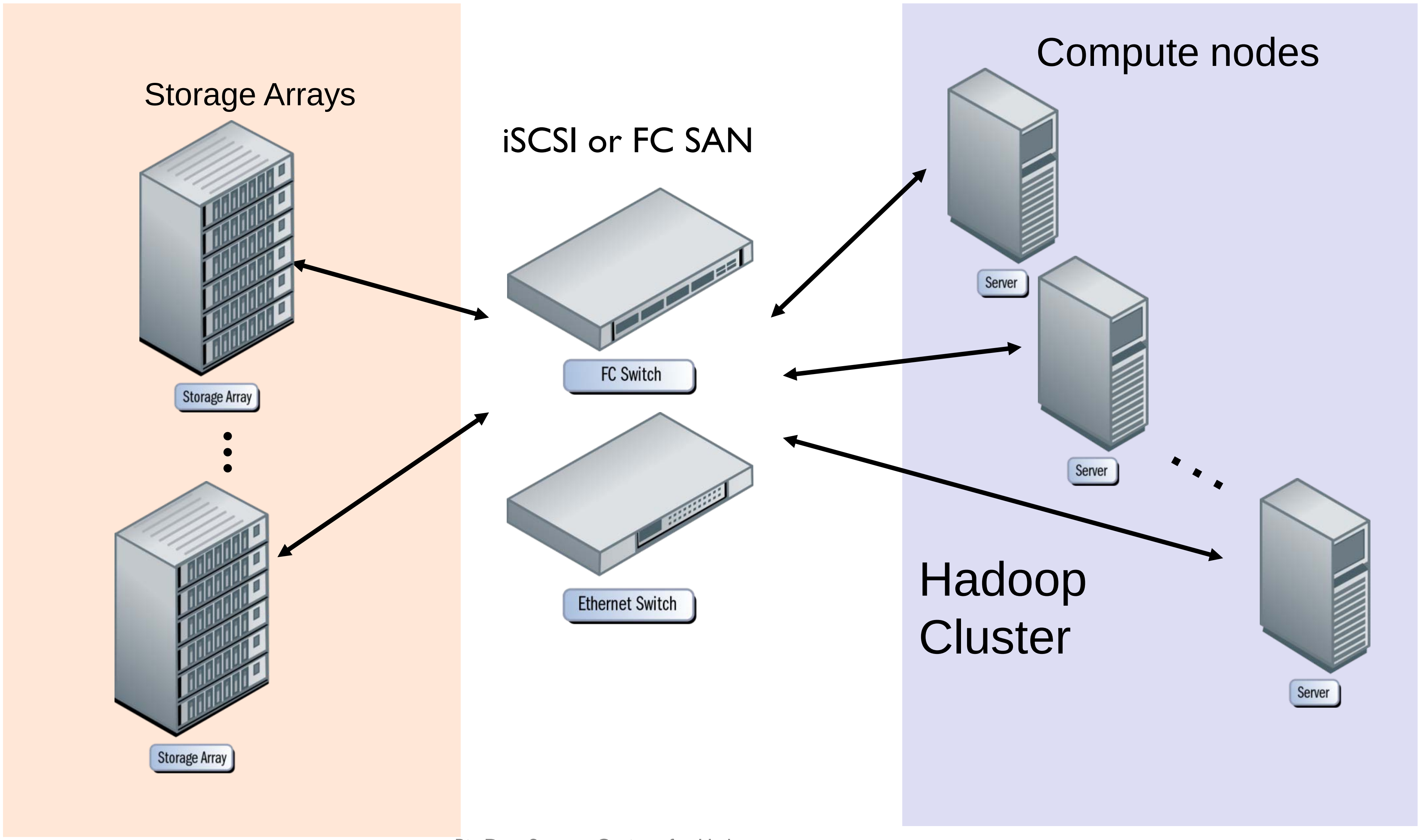
Education SNIA

- Pros
  - Writes are highly parallel
    - Large files are broken into many parts, distributed across the cluster
    - Three copies of any file block, one written local, two remote
    - Not a simple round-robin scheme, tuned for Hadoop jobs
  - Job tracker attempts to make reads local
    - If possible, tasks scheduled in same node as the needed file segment
    - Duplicate file segments are also readable, can be used for tasks too
- Cons
  - Not a kernel-based POSIX filesystem – incompatible with standard applications and utilities
  - High replication cost compared with RAID/shared disk
  - The NameNode keeps track of data location
    - SPOF (for now), location data is critical and must be protected
    - Scalability bottleneck (everything has to be in memory)

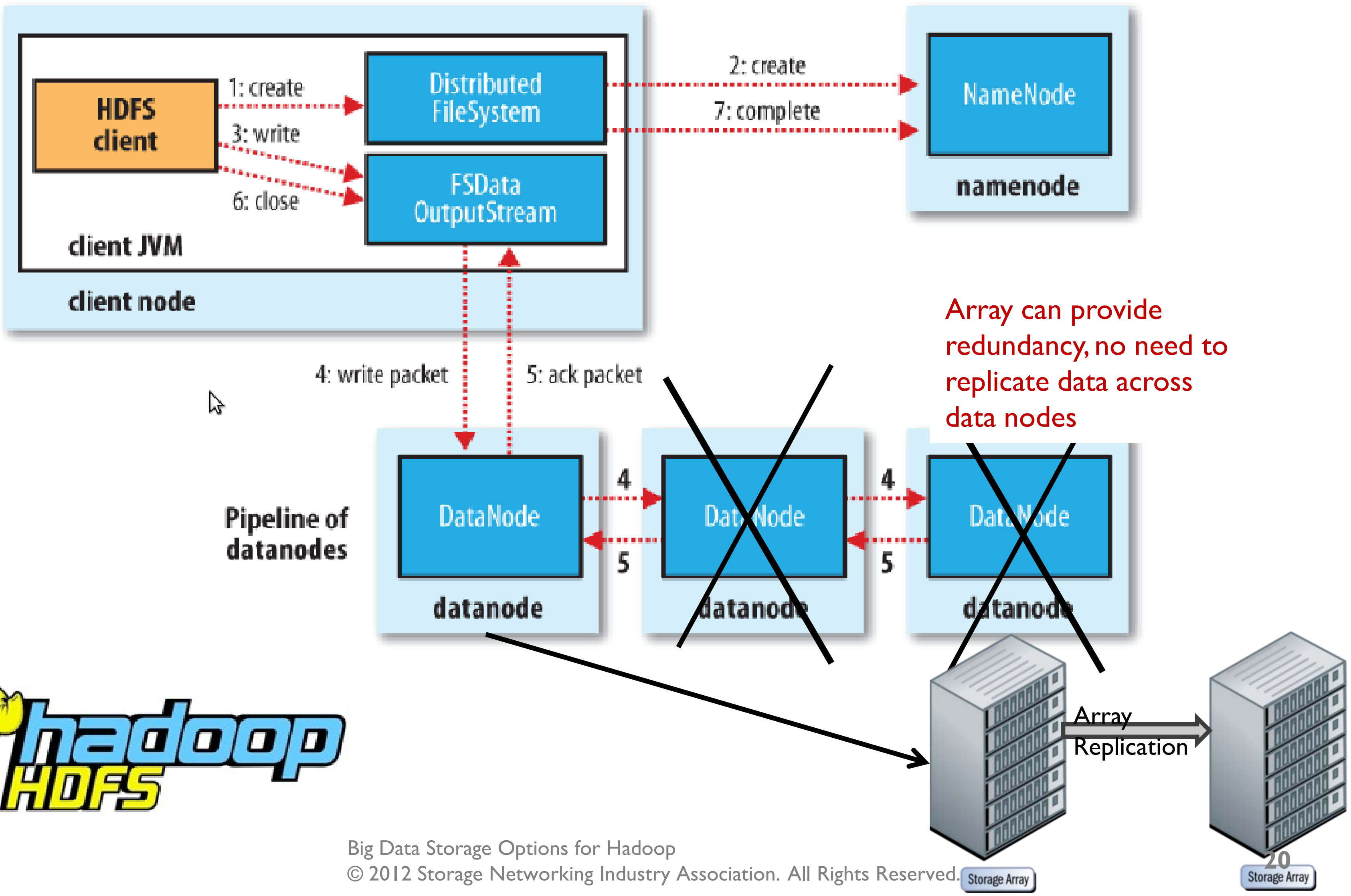
Other HDFS storage options

- HDFS on *Storage Area Network* (SAN) attached storage
  - A lot like DAS, but disks are *logical volumes* in *storage array(s)*, accessed across a SAN
  - HDFS doesn't know the difference
- SAN attached arrays aren't the same as DAS
  - Array has its own cache, redundancy, replication, etc.
  - Any node on the SAN can access any array volume
    - So a new node can be assigned to a failed node's data

HDFS with SAN Storage

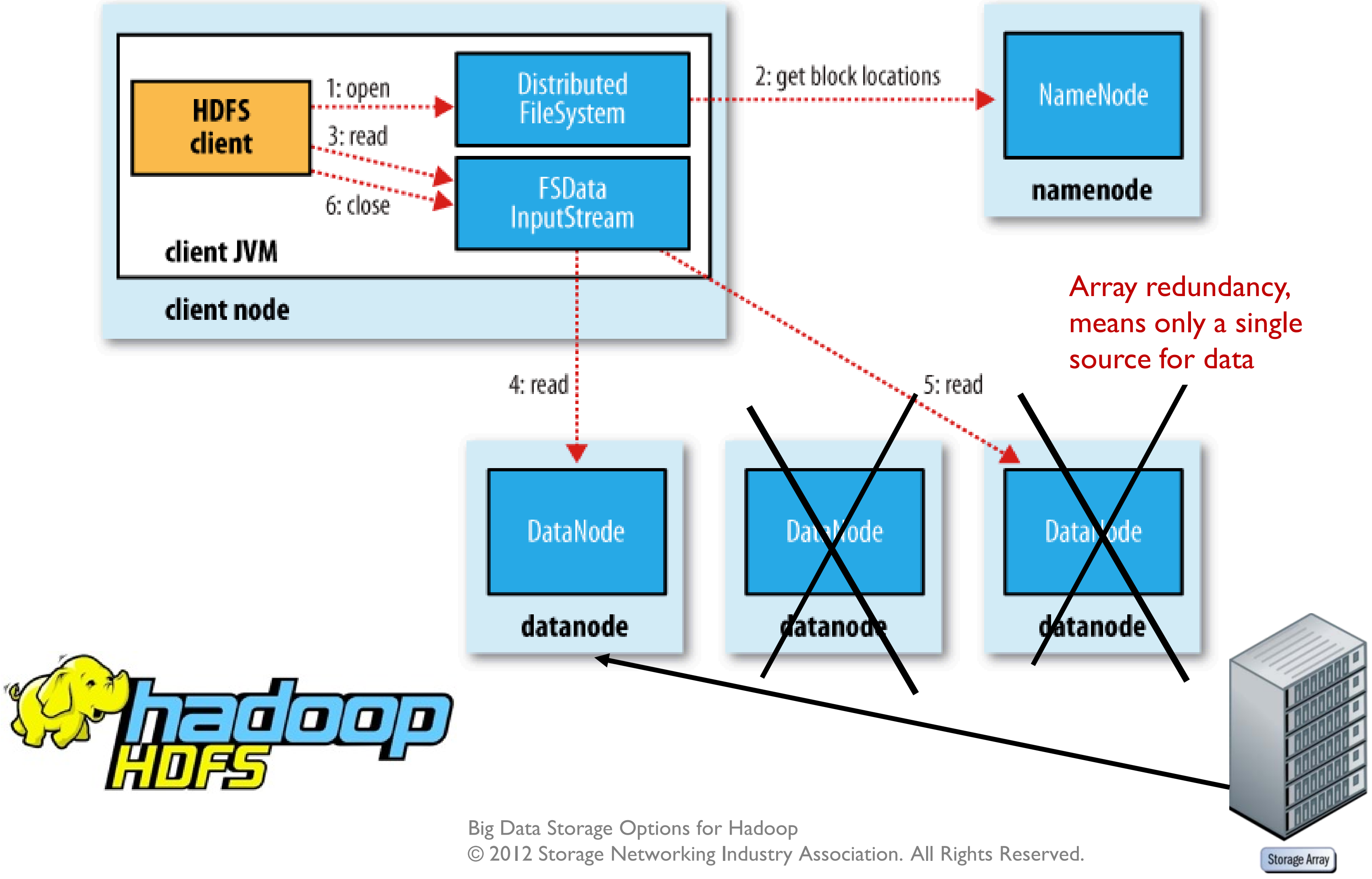


HDFS File Write Operation



Array can provide redundancy, no need to replicate data across data nodes

HDFS File Read Operation



Array redundancy, means only a single source for data

SAN for Hadoop Storage

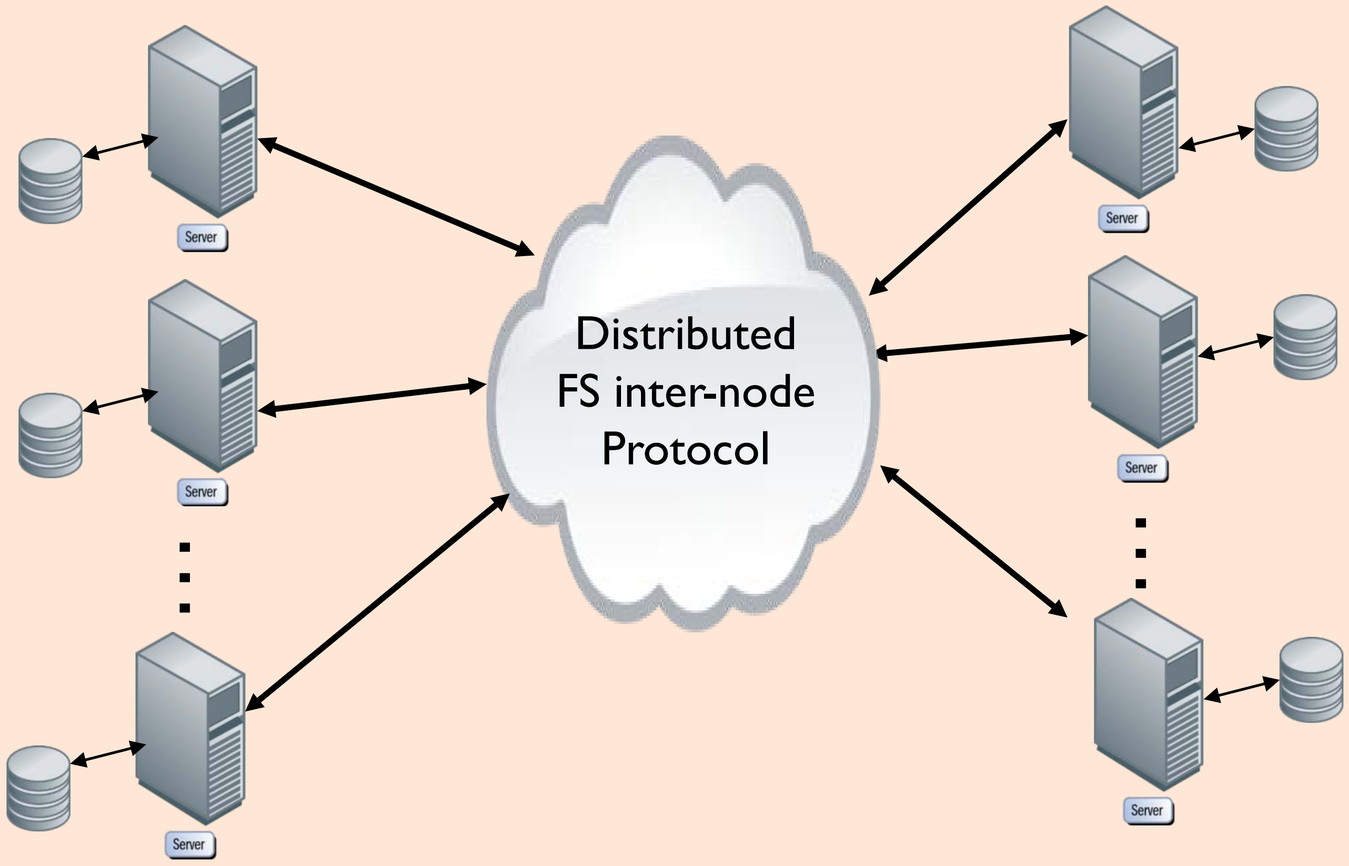
- Instead of storing data on direct attached local disks, data is in one or more arrays attached to data nodes through a SAN
  - Looks like local storage to data nodes
  - Hadoop still utilizes HDFS
- Pros
  - All the normal advantages of arrays
    - RAID, centralized caching, thin provisioning, other advanced array features
    - Centralized management, easy redistribution of storage
  - Retains advantages of HDFS (as long as array is not over-utilized)
  - Easy failover when compute node dies, can eliminate or reduce 3-way replication
- Cons
  - Cost? It depends
  - Unless if multiple arrays are used, scale is limited
    - And with multiple arrays, management and cost advantages are reduced
  - Still have HDFS complexity and manageability issues

Other distributed filesystems

- Kernel-based tightly coupled distributed file system
  - Kernel-based, i.e., no special access libraries, looks like a normal local file system
  - These filesystems have existed for years in high performance computing, scale-out NAS servers, and other scale-out computing environments
    - Many commercial and research examples
    - Not originally designed for Hadoop like HDFS
  - Location awareness is part of the file system – no NameNode
    - Works better if functionality is "exposed" to Hadoop
- Compute nodes may or may not have local storage
  - Compute nodes are "part of the storage cluster," but may be diskless – i.e., equal access to files and global namespace
    - Can tie the filesystem's location awareness into task tracker to reduce remote storage access
  - Remote storage is accessed using a filesystem specific inter-node protocol
    - Single network hop due to filesystem's location awareness

Distributed FS – local disks

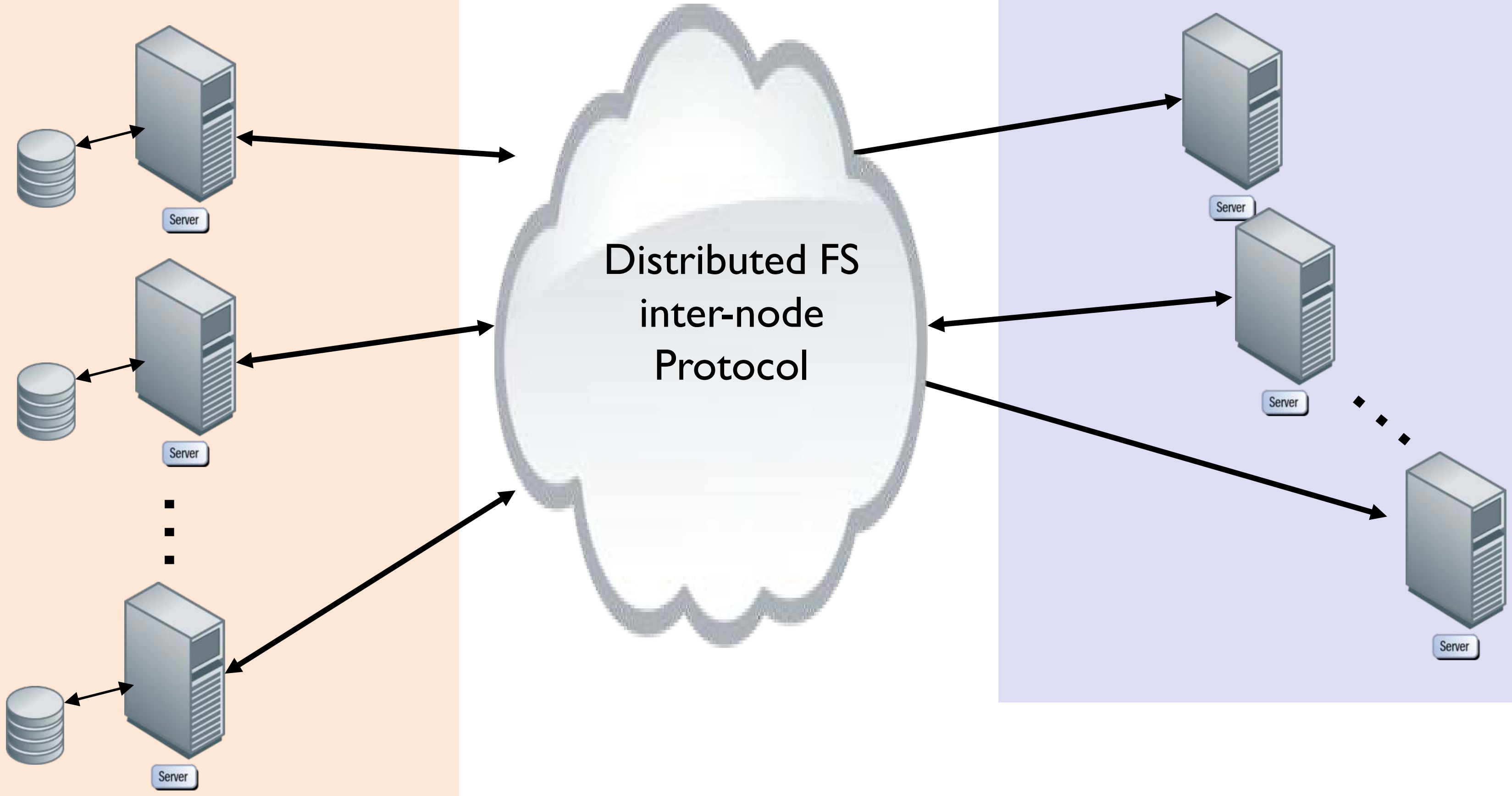
Compute nodes are part of the DFS, data spread across nodes



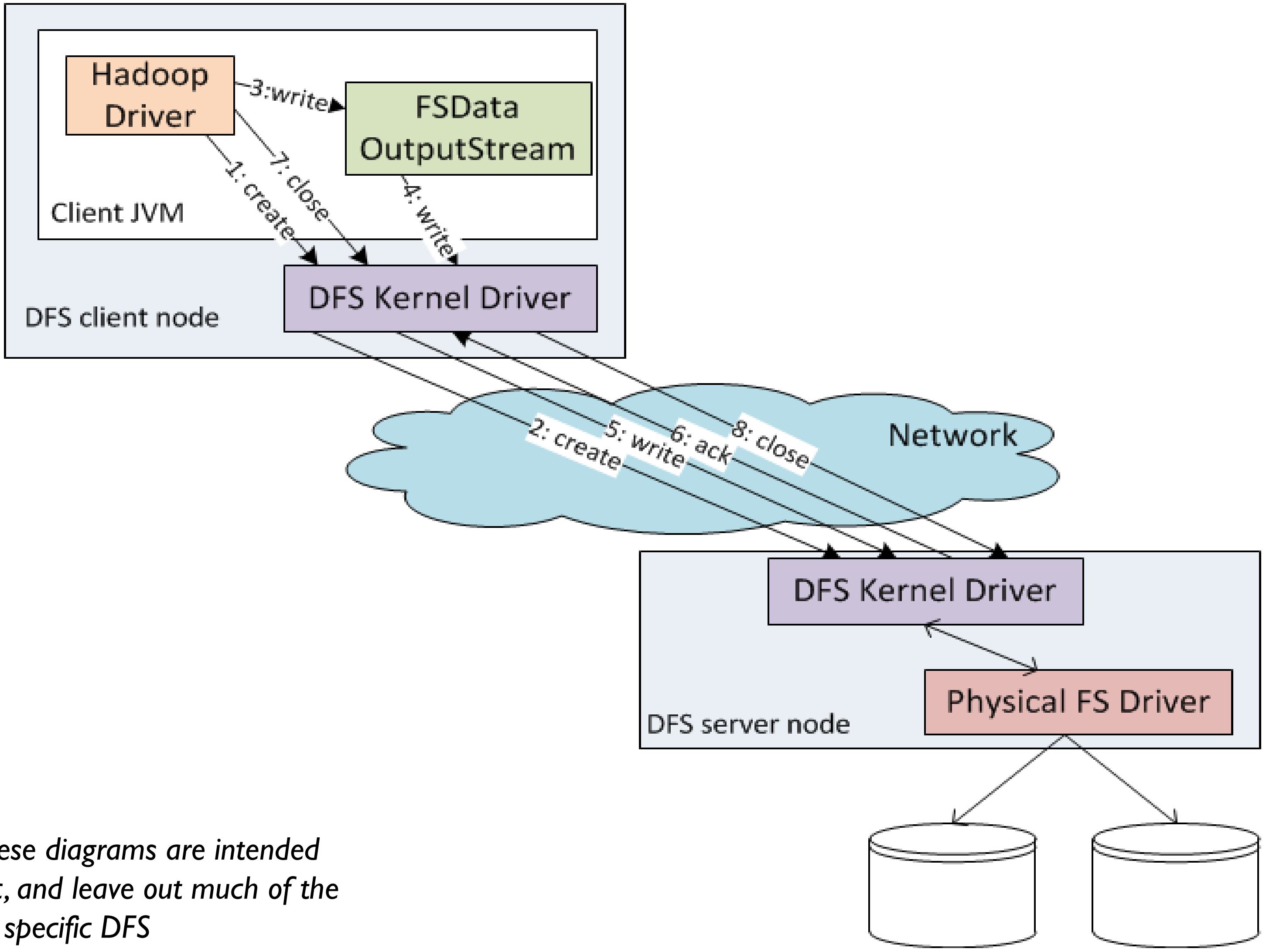
Distributed FS – remote disks

Scale out nodes are distributed FS servers

Compute nodes are distributed FS clients

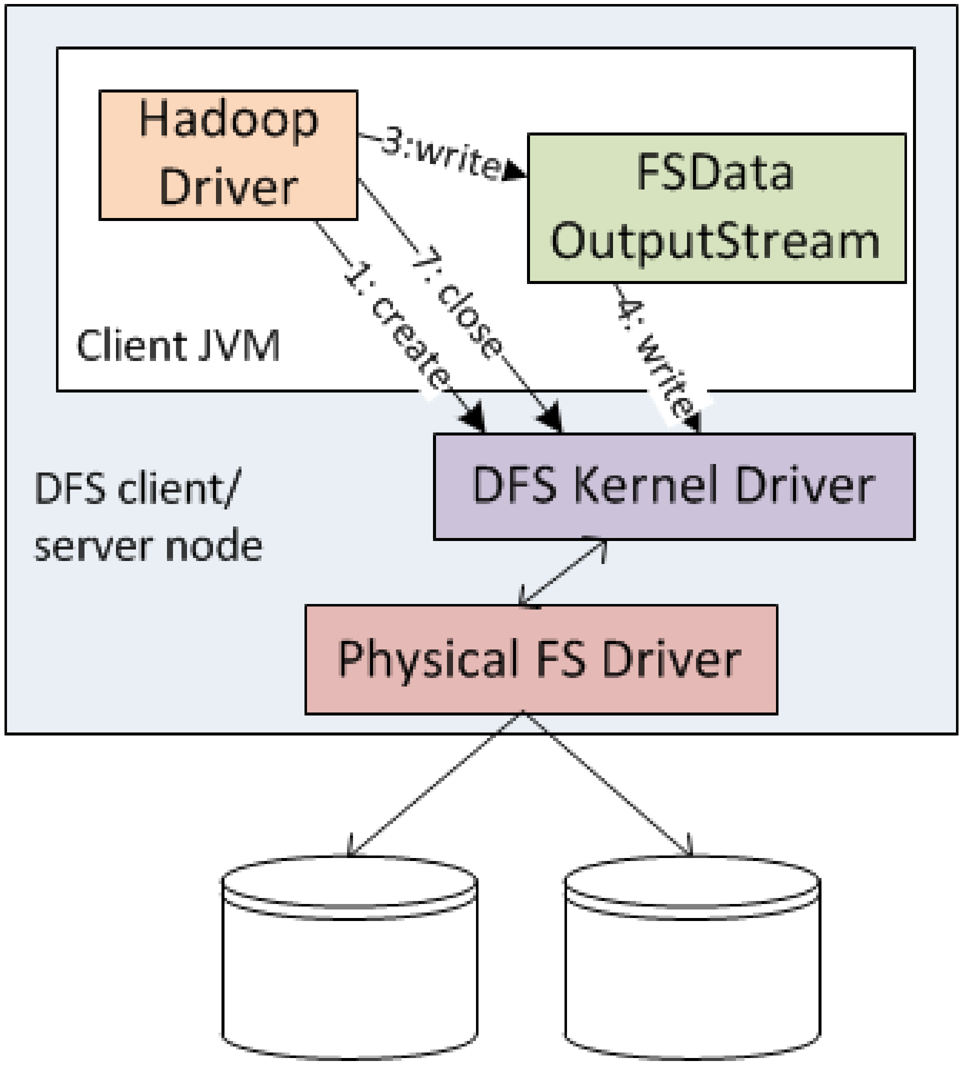


Remote DFS Write Operation



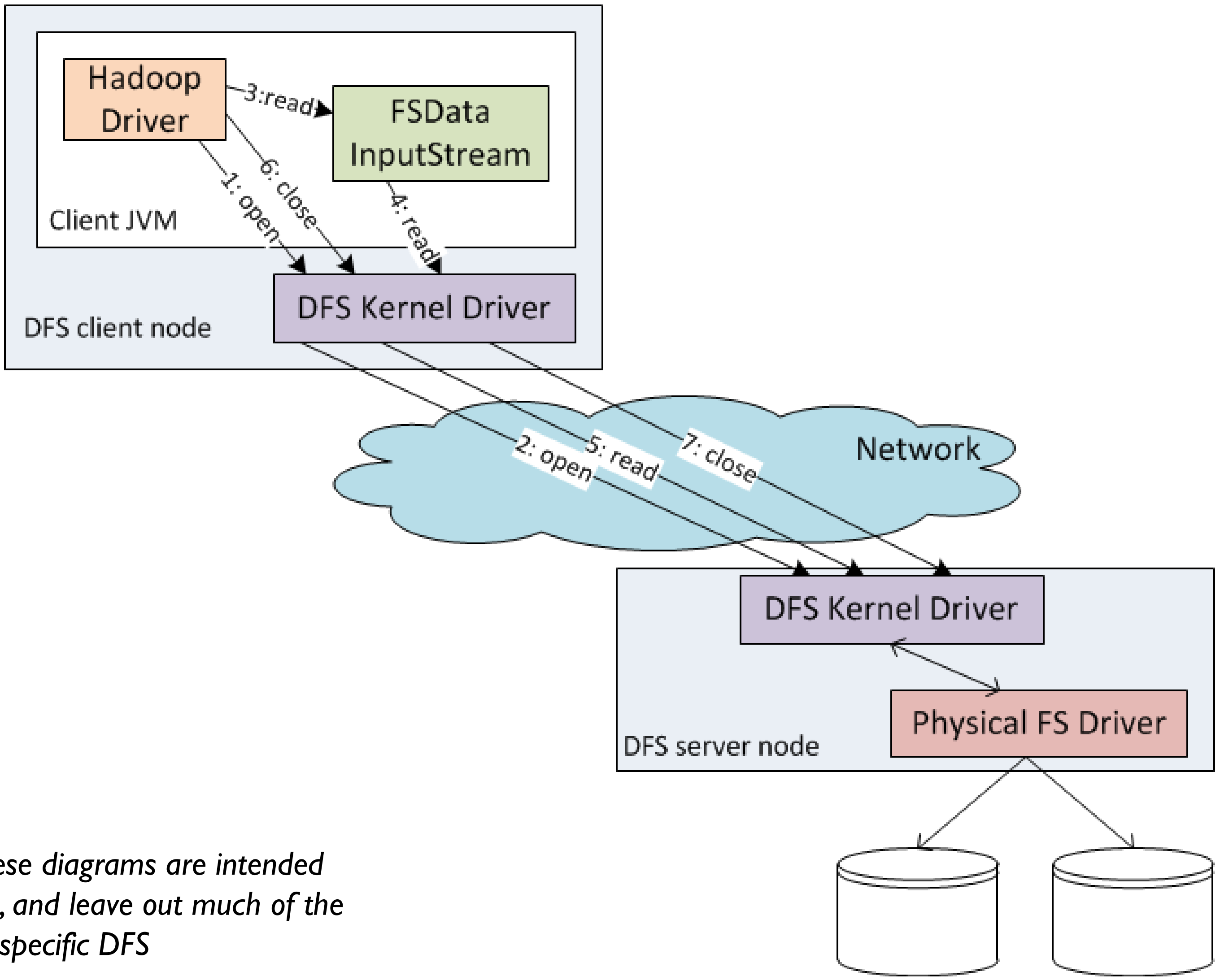
Note that these diagrams are intended to be generic, and leave out much of the detail of any specific DFS

Local DFS Write Operation



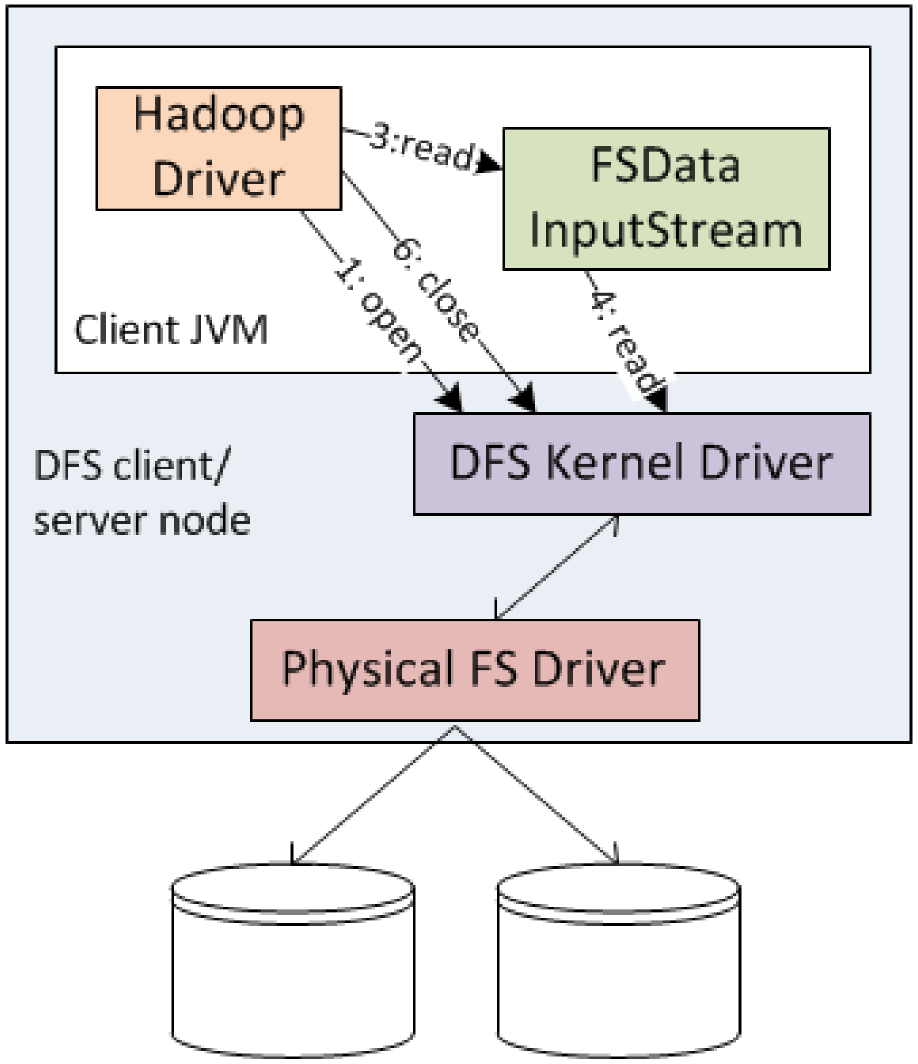
Note that these diagrams are intended to be generic, and leave out much of the detail of any specific DFS

Remote DFS Read Operation



Note that these diagrams are intended to be generic, and leave out much of the detail of any specific DFS

Local DFS Read Operation



Note that these diagrams are intended to be generic, and leave out much of the detail of any specific DFS

Tightly coupled DFS for Hadoop

- General purpose shared file system
  - Implemented in the kernel, single namespace, compatible with most applications (no special library or language)
- Data is distributed across local storage node disks
  - Architecturally like HDFS
    - Can utilize same disk options as HDFS
      - Including shared nothing DAS
      - SAN storage
    - Some can also support "shared SAN" storage where raw volumes can be accessed by multiple nodes
      - Failover model – where only one node actively uses a volume, other can take over after failure
      - Multiple initiator model – where multiple nodes actively use a volume
  - Shared nothing option has similar cost/performance to HDFS on DAS

Tightly coupled DFS for Hadoop

- Pros
  - Shared data access, any node can access any data like it is local
  - POSIX compatible, works for non-Hadoop apps just like a local file system
  - Centralized management and administration
  - No NameNode, may have a better block mapping mechanism
  - Compute in-place, same copy can be served via NFS/CIFS
  - Many of the performance benefits
- Cons
  - HDFS is highly optimized for Hadoop, unlikely to get same optimization for a general purpose DFS
    - Large file striping is not regular, based on compute distribution
    - Copies are simultaneously readable
  - Strict POSIX compliance leads to unnecessary serialization
    - Hadoop assumes multiple-access to files, however, accesses are on block boundaries and don't overlap
    - Need to relax POSIX compliance for large files, or just stick with many smaller files
  - Some DFS's have scaling limitations that are worse than HDFS, not designed for "thousands" of nodes

Summary

- Hadoop provides a scalable fault-tolerant environment for analyzing unstructured and structured information
- The default way to store data for Hadoop is HDFS on local direct attached disks
- Alternatives to this architecture include SAN array storage and tightly-coupled general purpose DFS
  - They can provide some significant advantages
  - However, they aren't without their downsides, its hard to beat a filesystem designed specifically for Hadoop
- Which one is best for you?
  - Depends on what is most important – cost, manageability, compatibility with existing infrastructure, performance, scale, …

Refer to Other Tutorials

- Use this icon to refer to other SNIA Tutorials where appropriate

Check out SNIA Tutorial:

Enter Tutorial Title Here

- Use the Hands-On Lab icon only if you've been notified that your tutorial is a cross-referenced match to the Hands-On Lab

Attribution & Feedback

The SNIA Education Committee would like to thank the following individuals for their contributions to this Tutorial.

Authorship History

Sam Fineberg, August 2012

Updates:

Additional Contributors

Rob Peglar

Please send any questions or comments regarding this SNIA Tutorial to
tracktutorials@snia.org