CLOUD COMPUTING APPLICATIONS

HDFS Introduction

Roy Campbell & Reza Farivar

Large-Scale Data Processing

- Many tasks
  - Processing lots of data in parallel to produce lots of other data
- Large-scale data processing
  - Want to use 1000s of CPUs
    - But don't want overhead of **managing** storage
    - Storage devices fail 1.7% year 1 – 8.6% year 3 (Google, 2007)
    - 10,000 nodes, 7 disks per node, year 1, 1190 failures/yr or 3.3 failures/day
- MapReduce provides:
  - User-defined functions
  - Automatic parallelization and distribution
  - Fault tolerance
  - I/O scheduling
  - Status and monitoring

HDFS

- Synergistic with Hadoop
  - Apache Project inspired by Google Map/Reduce and GFS
- Massive throughput
- Throughput scales with attached HDs
- Have seen VERY LARGE production clusters
  - Facebook, Yahoo… Nebraska
- Doesn’t even pretend to be POSIX compliant
- Optimized for reads, sequential writes and appends

References

- The Google File System
  - SOSP 2003
- The Hadoop Distributed File System
  - 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)

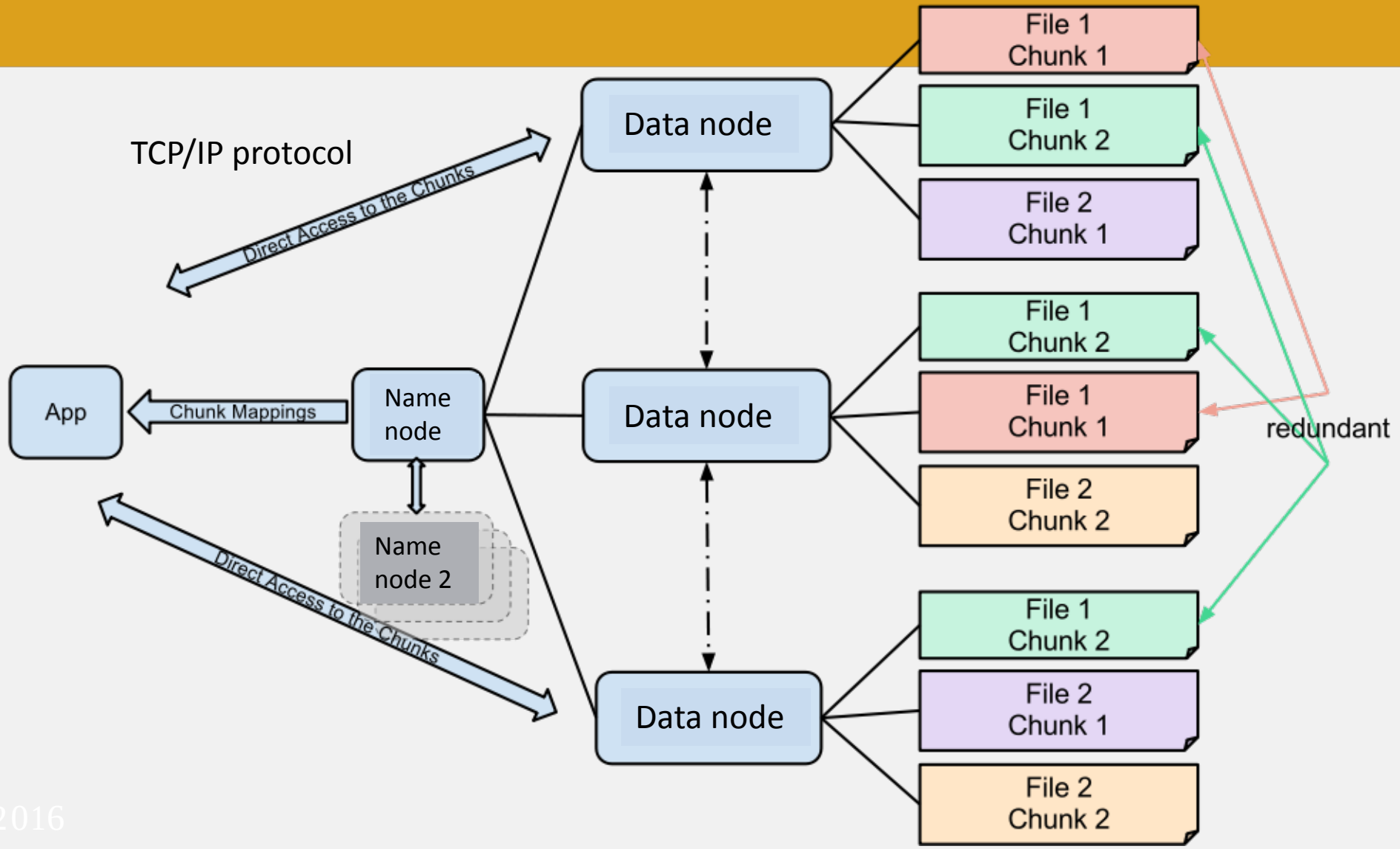
Stable Storage

- If nodes failure is the norm and not the exception, how can we store data persistently?
- **Answer:** Distributed File System replicates files
  - Provides global file namespace
  - Google GFS, Hadoop HDFS
  - Kosmix KFS
- Typical usage pattern
  - Huge files (100s of GB to TB)
  - Data is rarely updated in place
  - Multiple copies improves availability
  - Reads and appends are common

Distributed File System

- Datanode Servers
  - A file is split into contiguous chunks
  - Typically each chunk is 16-64MB
  - Each chunk replicated (usually 2x or 3x)
  - Sends heartbeat and BlockReport to namenode
- Replicas are placed: one on a node in a local rack, one on a different node in the local rack, and one on a node in a different rack

HDFS Architecture



Distributed File System

- Master node
  - a.k.a. NameNode in HDFS
  - Stores metadata
  - Might be replicated
- Client library for file access
  - Talks to master to find data node chunk
  - Connects directly to data node servers to access data

Replication Pipelining

- When the client receives response from NameNode, it flushes its block in small pieces (4K) to the first replica, that in turn copies it to the next replica and so on
- Thus data is pipelined from data node to the next

Staging

- A client request to create a file does not reach NameNode immediately
- HDFS client caches the data into a temporary file. When the data reaches an HDFS block size, the client contacts the NameNode
- NameNode inserts the filename into its hierarchy and allocates a data block for it
- The NameNode responds to the client with the identity of the data node and the destination of the replicas (data nodes) for the block
- Then the client flushes it from its local memory

Application Programming Interface

- HDFS provides Java API for application to use
- Python access is also used in many applications
- A C language wrapper for Java API is also available
- A command line interface provided in Hadoop
- The syntax of the commands is similar to bash
- Example: to create a directory /foodir
- /bin/hadoop dfs –mkdir /foodir
- An HTTP browser can be used to browse the files of an HDFS instance