Cloud Computing

PaaS Techniques

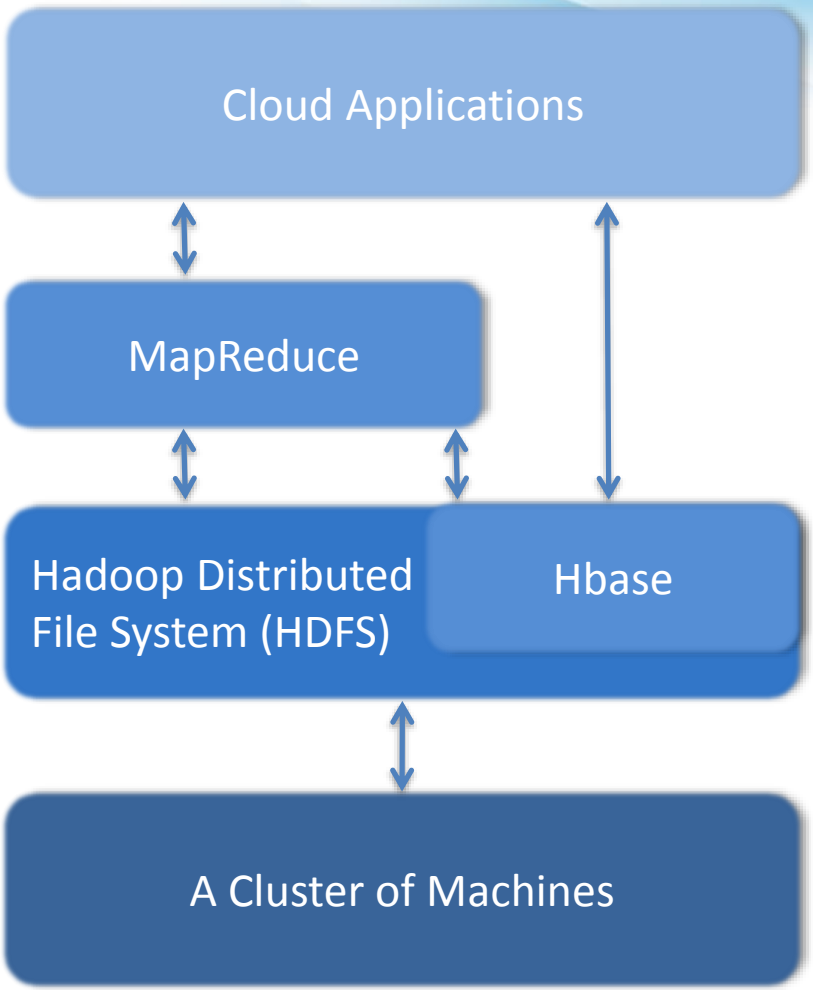
File System

Agenda

- Overview
  - Hadoop & Google
- PaaS Techniques
  - File System
    - GFS, HDFS
  - Programming Model
    - MapReduce, Pregel
  - Storage System for Structured Data
    - Bigtable, Hbase

Hadoop

- Hadoop is
  - A distributed computing platform
  - A software framework that lets one easily write and run applications that process vast amounts of data
  - Inspired from published papers by Google



Google

- Google published the designs of web-search engine
  - SOSP 2003
    - The **G**oogle **F**ile **S**ystem
  - OSDI 2004
    - **MapReduce** : Simplified Data Processing on Large Cluster
  - OSDI 2006
    - **Bigtable**: A Distributed Storage System for Structured Data

Google vs. Hadoop

Develop Group	Google	Apache
Sponsor	Google	Yahoo, Amazon
Resource	open document	open source
File System	GFS	HDFS
Programming Model	MapReduce	Hadoop MapReduce
Storage System (for structured data)	Bigtable	Hbase
Search Engine	Google	Nutch
OS	Linux	Linux / GPL

Agenda

- Overview
  - Hadoop & Google
- PaaS Techniques
  - File System
    - GFS, HDFS
  - Programming Model
    - MapReduce, Pregel
  - Storage System for Structured Data
    - Bigtable, Hbase

File System Overview

Distributed File Systems (DFS)

Hadoop Distributed File Systems (HDFS)

FILE SYSTEM

File System Overview

- System that permanently stores data
- To store data in units called “files” on disks and other media
- Files are managed by the Operating System
- The part of the Operating System that deal with files is known as the “File System”
  - A file is a collection of disk blocks
  - File System maps file names and offsets to disk blocks
- The set of valid paths form the “namespace” of the file system.

What Gets Stored

- User data itself is the bulk of the file system's contents
- Also includes meta-data on a volume-wide and per-file basis:

Volume-wide

- Available space
- Formatting info.
- Character set
- …

Per-file

- Name
- Owner
- Modification data
- …

Design Considerations

- Namespace
  - Physical mapping
  - Logical volume
- Consistency
  - What to do when more than one user reads/writes on the same file?
- Security
  - Who can do what to a file?
  - Authentication/Access Control List (ACL)
- Reliability
  - Can files not be damaged at power outage or other hardware failures?

Local FS on Unix-like Systems(1/4)

- Namespace
  - root directory “/”, followed by directories and files.
- Consistency
  - “sequential consistency”, newly written data are immediately visible to open reads
- Security
  - uid/gid, mode of files
  - kerberos: tickets
- Reliability
  - journaling, snapshot

Local FS on Unix-like Systems(2/4)

- Namespace
  - Physical mapping
    - a directory and all of its subdirectories are stored on the same physical media
      - /mnt/cdrom
      - /mnt/disk1, /mnt/disk2, … when you have multiple disks
  - Logical volume
    - a logical namespace that can contain multiple physical media or a partition of a physical media
      - still mounted like /mnt/vol1
      - dynamical resizing by adding/removing disks without reboot
      - splitting/merging volumes as long as no data spans the split

Local FS on Unix-like Systems(3/4)

- Journaling
  - Changes to the filesystem is logged in a *journal* before it is committed
    - useful if an atomic action needs two or more writes
      - e.g., appending to a file (update metadata + allocate space + write the data)
    - can play back a journal to recover data quickly in case of hardware failure.
  - What to log?
    - changes to file content: heavy overhead
    - changes to metadata: fast, but data corruption may occur
  - Implementations: xfs3, ReiserFS, IBM's JFS, etc.

Local FS on Unix-like Systems(4/4)

- Snapshot
  - A snapshot = a copy of a set of files and directories at a point in time
    - read-only snapshots, read-write snapshots
    - usually done by the filesystem itself, sometimes by LVMs
    - backing up data can be done on a read-only snapshot without worrying about consistency
  - Copy-on-write is a simple and fast way to create snapshots
    - current data is the snapshot
    - a request to write to a file creates a new copy, and work from there afterwards
  - Implementation: UFS, Sun's ZFS, etc.

File System Overview

Distributed File Systems (DFS)

Hadoop Distributed File Systems (HDFS)

FILE SYSTEM

Distributed File Systems

- Allows access to files from multiple hosts sharing via a computer network
- Must support concurrency
  - Make varying guarantees about locking, who “wins” with concurrent writes, etc...
  - Must gracefully handle dropped connections
- May include facilities for transparent replication and fault tolerance
- Different implementations sit in different places on complexity/feature scale

When is DFS Useful

- Multiple users want to share files
- The data may be much larger than the storage space of a computer
- A user wants to access his/her data from different machines at different geographic locations
- Users want a storage system
  - Backup
  - Management

Note that a “user” of a DFS may actually be a “program”

Design Considerations of DFS(1/2)

- Different systems have different designs and behaviors on the following features
  - Interface
    - file system, block I/O, custom made
  - Security
    - various authentication/authorization schemes
  - Reliability (fault-tolerance)
    - continue to function when some hardware fail (disks, nodes, power, etc.)

Design Considerations of DFS(2/2)

- Namespace (virtualization)
  - provide logical namespace that can span across physical boundaries
- Consistency
  - all clients get the same data all the time
  - related to locking, caching, and synchronization
- Parallel
  - multiple clients can have access to multiple disks at the same time
- Scope
  - local area network vs. wide area network

HDFS

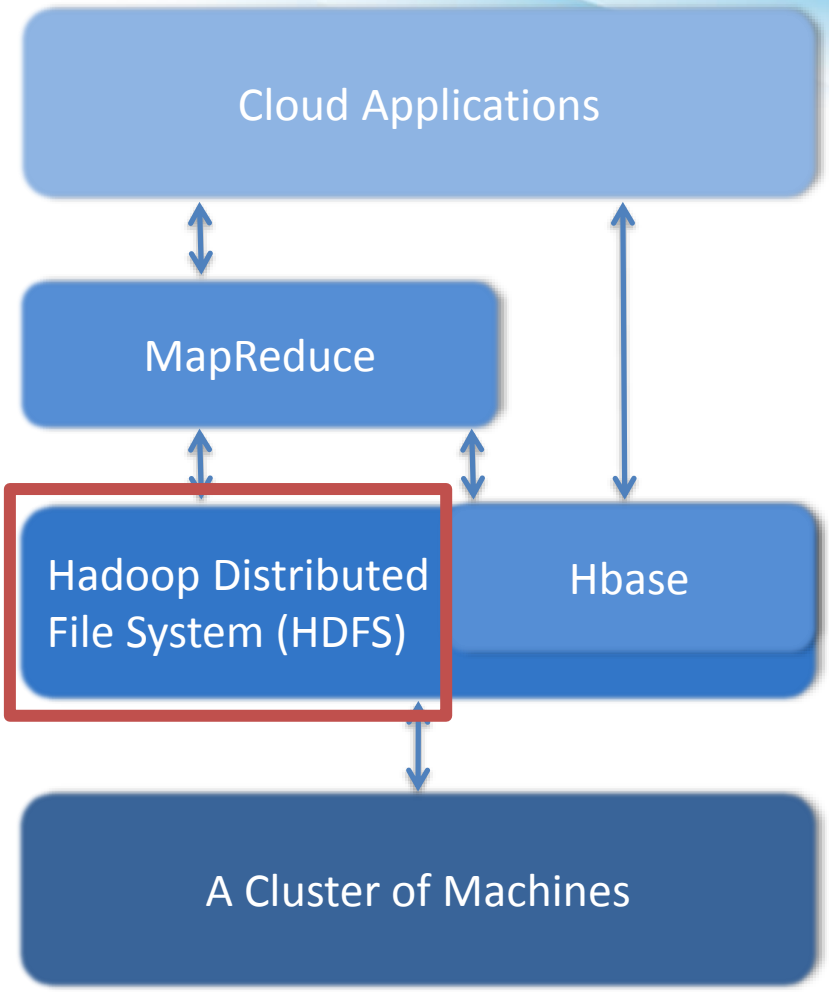
HOW ABOUT HADOOP

HDFS

- Overview
- Architecture
- Implementation
- Other Issue

What's HDFS

- Hadoop Distributed File System
  - Reference from Google File System
  - A scalable distributed file system for large data analysis
  - Based on commodity hardware with high fault-tolerance
  - The primary storage used by Hadoop applications



HDFS's Feature(1/2)

- Large data sets and files
  - Support **Petabytes** size
- Heterogeneous
  - Could be deployed on **different hardware**
- Streaming data access
  - **Batch** processing rather than interactive user access
  - Provide high aggregate data bandwidth

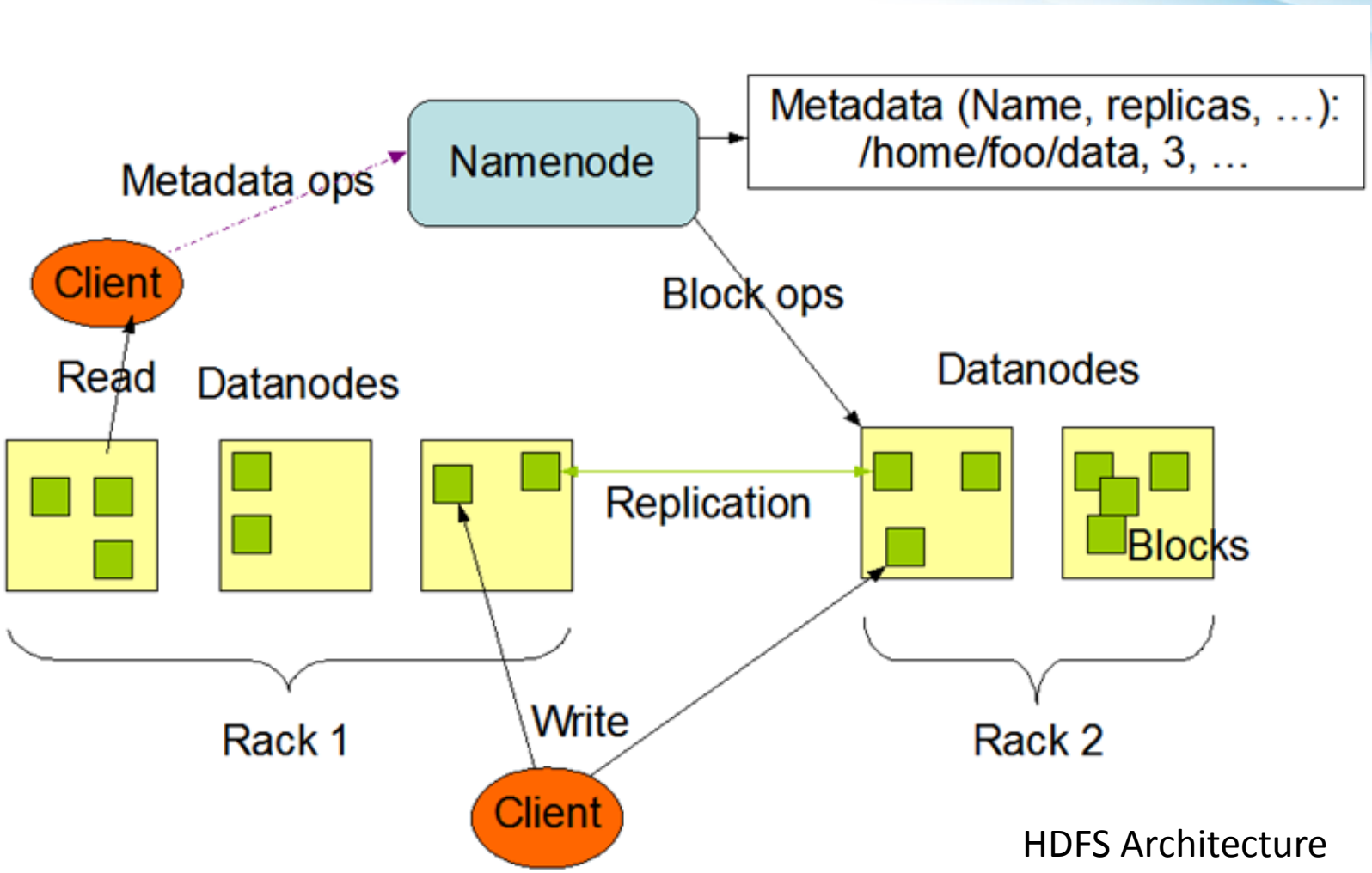
HDFS's Feature(2/2)

- Fault-Tolerance
  - The norm rather than exception
  - Automatic recovery or report failure
- Coherency Model
  - **Write-once-read-many**
  - This assumption simplifies coherency
- Data Locality
  - Move compute to data

HDFS

- Overview
- Architecture
- Implementation
- Other Issue

How to manage data



HDFS Architecture

Namenode

- Each HDFS cluster has **one Namenode**
- Manage the **file system namespace**
- Regulate **access** to files by clients
- Execute file system **namespace operations**
- Maintain a **rackid**-to-DataNode map and tries to place replicas across racks

Datanode

- **One per node** in the cluster
- **Manage storage** attached to the nodes that they run on
- Serve **read and write** requests from the file system's clients
- Perform **block creation**, **deletion**, and **replication**

File System Namespace

- Traditional **hierarchical** file organization
- Does not support hard links or soft links
- Change to the file system namespace or its properties is **recorded by the Namenode**

HDFS

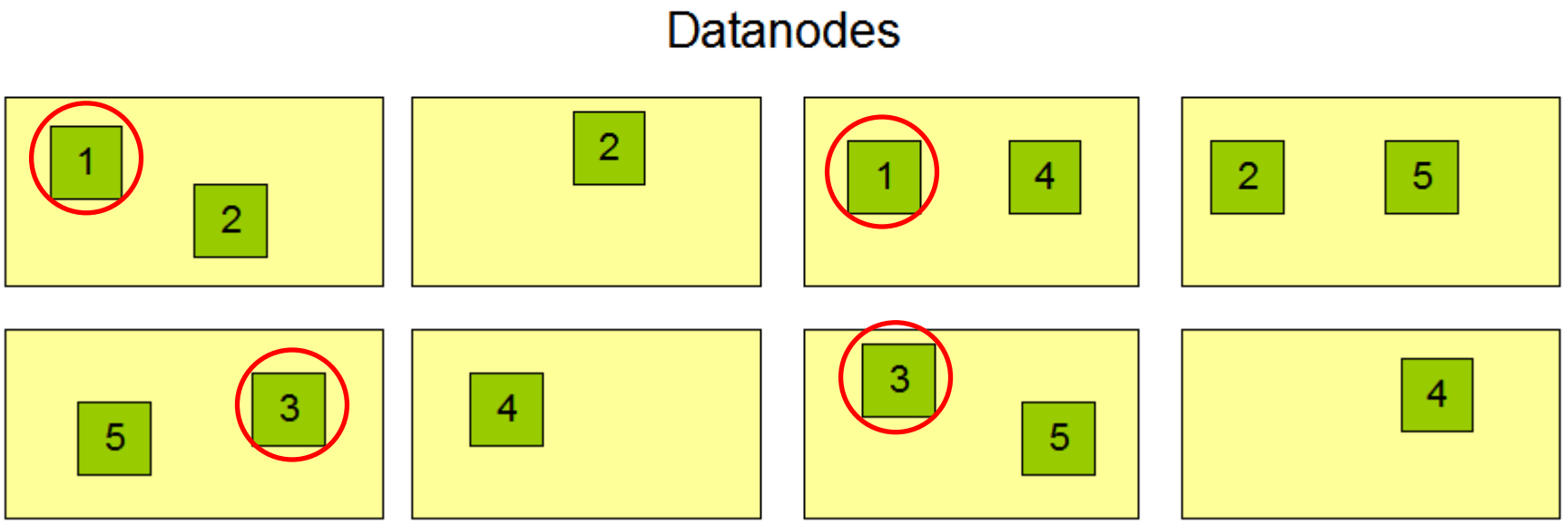
- Overview
- Architecture
- Implementation
- Other Issue

Data Replication

- Blocks of a file are replicated **for fault tolerance**
- The block size and replication factor are **configurable per file**
- Namenode makes all decisions regarding replication of blocks
  - **Heartbeat**: Datanode is functioning properly
  - **Blockreport**: a list of all blocks on a Datanode

Block Replication

Namenode (Filename, numReplicas, block-ids, …)
/users/sameerp/data/part-0, r:2, {1,3}, …
/users/sameerp/data/part-1, r:3, {2,4,5}, …



Replica Placement

- Rack-aware replica placement policy
  - data reliability
  - availability
  - network bandwidth utilization
- To validate it on production systems
  - learn more about its **behavior**
  - build a foundation to test
  - research more **sophisticated policies**

Screenshot

Number of Replicas:2

```
hadoop@ubuntu:/tmp/hadoop-hadoop/dfs/data/current$ ll -h
總計 192K
drwxr-xr-x 2 hadoop hadoop 4.0K 2010-08-12 16:47 ./
drwxr-xr-x 5 hadoop hadoop 4.0K 2010-08-11 17:46 ../
-rw-r--r-- 1 hadoop hadoop  22K 2010-08-12 16:38 blk_1339320897918224795
-rw-r--r-- 1 hadoop hadoop  179 2010-08-12 16:38 blk_1339320897918224795_1055.meta
-rw-r--r-- 1 hadoop hadoop 1.4K 2010-08-12 16:46 blk_1434033524064351855
-rw-r--r-- 1 hadoop hadoop   19 2010-08-12 16:46 blk_1434033524064351855_1056.meta
-rw-r--r-- 1 hadoop hadoop  19K 2010-08-12 16:46 blk_-2130699322832305290
-rw-r--r-- 1 hadoop hadoop  159 2010-08-12 16:46 blk_-2130699322832305290_1056.meta
-rw-r--r-- 1 hadoop hadoop    4 2010-08-11 17:50 blk_-3648396202343768262
-rw-r--r-- 1 hadoop hadoop   11 2010-08-11 17:50 blk_-3648396202343768262_1022.meta
-rw-r--r-- 1 hadoop hadoop  270 2010-08-12 16:15 blk_-412715259689855069
-rw-r--r-- 1 hadoop hadoop   11 2010-08-12 16:15 blk_-412715259689855069_1045.meta
-rw-r--r-- 1 hadoop hadoop  22K 2010-08-12 16:28 blk_6434768955339204249
-rw-r--r-- 1 hadoop hadoop  183 2010-08-12 16:28 blk_6434768955339204249_1054.meta
-rw-r--r-- 1 hadoop hadoop  17K 2010-08-12 16:18 blk_8302896404924163137
-rw-r--r-- 1 hadoop hadoop  139 2010-08-12 16:18 blk_8302896404924163137_1053.meta
-rw-r--r-- 1 hadoop hadoop  17K 2010-08-12 16:09 blk_-8974169280347947717
-rw-r--r-- 1 hadoop hadoop  139 2010-08-12 16:09 blk_-8974169280347947717_1039.meta
-rw-r--r-- 1 hadoop hadoop  17K 2010-08-12 16:09 blk_-9170609306520421532
-rw-r--r-- 1 hadoop hadoop  139 2010-08-12 16:09 blk_-9170609306520421532_1044.meta
-rw-r--r-- 1 hadoop hadoop 2.8K 2010-08-12 17:34 dncp_block_verification.log.curr
```

```
[root@localhost current]# ll -h
total 7.7M
-rw-rw-r-- 1 hadoop hadoop 1.4K Aug 12 16:46 blk_1434033524064351855
-rw-rw-r-- 1 hadoop hadoop   19 Aug 12 16:46 blk_1434033524064351855_1056.meta
-rw-rw-r-- 1 hadoop hadoop  19K Aug 12 16:47 blk_-2130699322832305290
-rw-rw-r-- 1 hadoop hadoop  159 Aug 12 16:47 blk_-2130699322832305290_1056.meta
-rw-rw-r-- 1 hadoop hadoop 385K Aug 11 17:47 blk_-2153151756941551437
-rw-rw-r-- 1 hadoop hadoop 3.1K Aug 11 17:47 blk_-2153151756941551437_1021.meta
-rw-rw-r-- 1 hadoop hadoop 1.4M Aug 11 17:47 blk_-3530601104308258899
-rw-rw-r-- 1 hadoop hadoop  11K Aug 11 17:47 blk_-3530601104308258899_1020.meta
-rw-rw-r-- 1 hadoop hadoop  266 Aug 12 16:16 blk_-3964702335439094289
-rw-rw-r-- 1 hadoop hadoop   11 Aug 12 16:16 blk_-3964702335439094289_1047.meta
-rw-rw-r-- 1 hadoop hadoop  270 Aug 12 16:15 blk_-412715259689855069
-rw-rw-r-- 1 hadoop hadoop   11 Aug 12 16:15 blk_-412715259689855069_1045.meta
-rw-rw-r-- 1 hadoop hadoop  252 Aug 12 16:16 blk_4502120706448492568
-rw-rw-r-- 1 hadoop hadoop   11 Aug 12 16:16 blk_4502120706448492568_1046.meta
-rw-rw-r-- 1 hadoop hadoop 1.6M Aug 11 17:47 blk_-4786351927176103133
-rw-rw-r-- 1 hadoop hadoop  13K Aug 11 17:47 blk_-4786351927176103133_1018.meta
-rw-rw-r-- 1 hadoop hadoop 659K Aug 11 17:47 blk_5009733536760461414
-rw-rw-r-- 1 hadoop hadoop 5.2K Aug 11 17:47 blk_5009733536760461414_1016.meta
-rw-rw-r-- 1 hadoop hadoop 336K Aug 11 17:47 blk_-6948451480699326814
-rw-rw-r-- 1 hadoop hadoop 2.7K Aug 11 17:47 blk_-6948451480699326814_1019.meta
-rw-rw-r-- 1 hadoop hadoop 1.9M Aug 11 17:47 blk_-7511688267480827381
-rw-rw-r-- 1 hadoop hadoop  15K Aug 11 17:47 blk_-7511688267480827381_1017.meta
-rw-rw-r-- 1 hadoop hadoop  17K Aug 12 16:09 blk_-8974169280347947717
-rw-rw-r-- 1 hadoop hadoop  139 Aug 12 16:09 blk_-8974169280347947717_1039.meta
-rw-rw-r-- 1 hadoop hadoop  17K Aug 12 16:10 blk_-9170609306520421532
-rw-rw-r-- 1 hadoop hadoop  139 Aug 12 16:10 blk_-9170609306520421532_1044.meta
-rw-rw-r-- 1 hadoop hadoop 1.3K Aug 12 16:17 blk_961116239961228894
-rw-rw-r-- 1 hadoop hadoop   19 Aug 12 16:17 blk_961116239961228894_1048.meta
-rw-rw-r-- 1 hadoop hadoop 1.3M Aug 12 16:38 blk_99424525126878213
-rw-rw-r-- 1 hadoop hadoop  11K Aug 12 16:38 blk_99424525126878213_1055.meta
```

Why it Fault-Tolerance

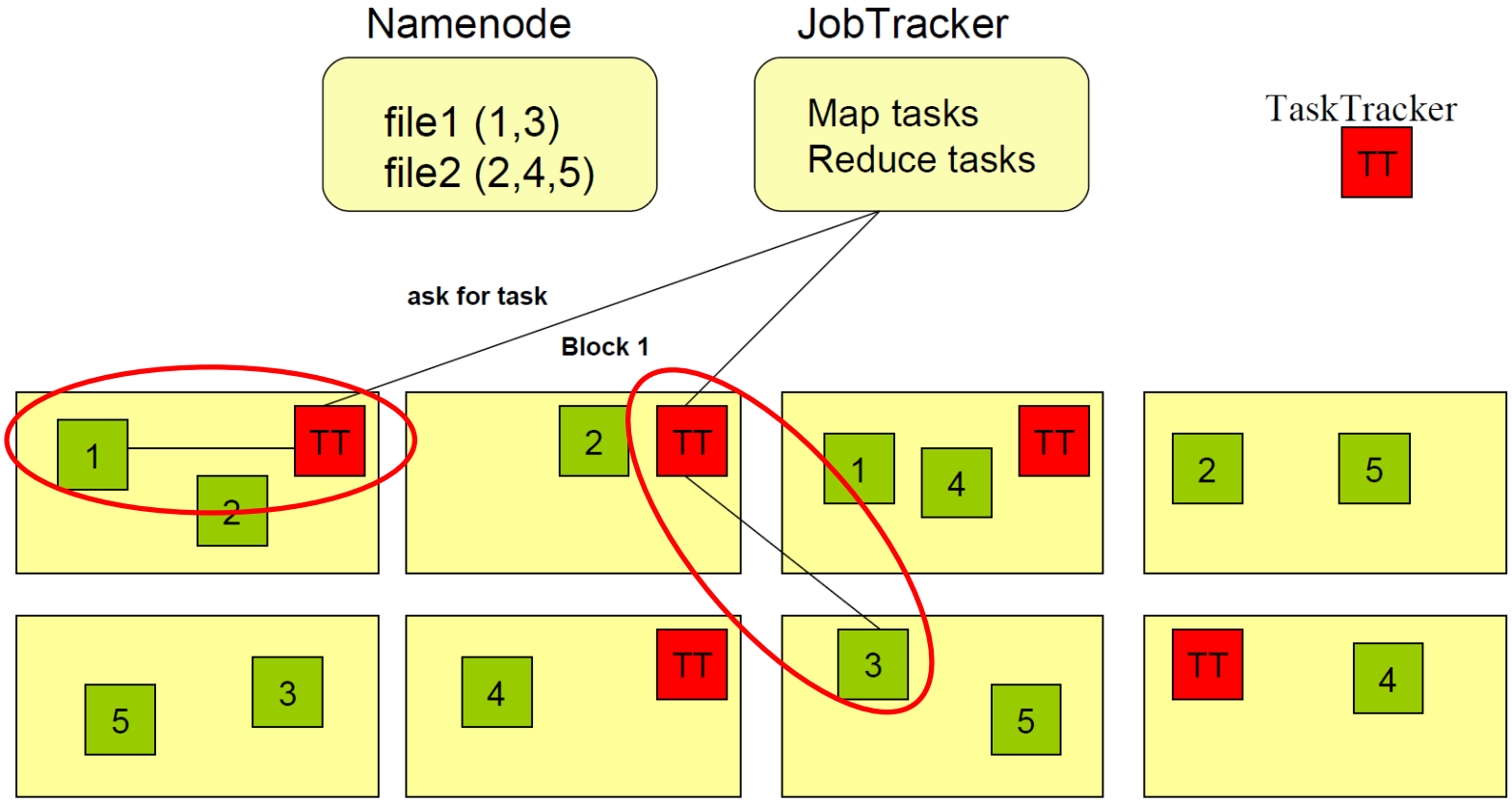
- Data Corrupt
  - Checked with **CRC32**
  - Replace corrupt block with replication one
- Network Fault & Datanode Fault
  - Datanode sends **heartbeat** to Namenode
- Namenode Fault
  - **FSImage** – **core file system** mapping image
  - **Editlog** – **transaction** log
  - **Multiple backups** of FSImage and Editlog
  - **Manually recovery** while Namenode Fault

CRC: Cyclical Redundancy Check

Coherency Model & Performance

- Coherency model of files
  - **Namenode** handles the operation of write, read and delete.
- Large Data Set and Performance
  - The default block size is **64MB**
  - Bigger block size will enhance **read performance**
  - Single file stored on HDFS might be larger than single physical disk of Datanode
  - **Fully distributed** blocks increase throughput of reading

About Data locality



HDFS

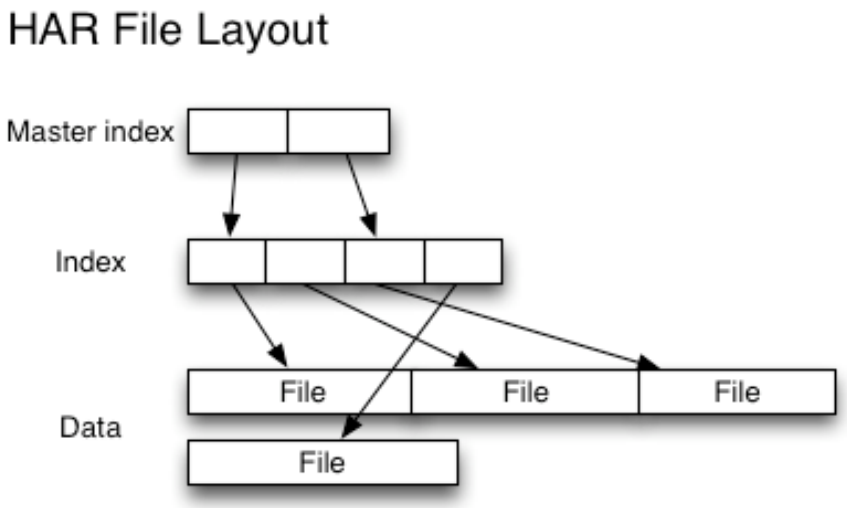
- Overview
- Architecture
- Implementation
- Other Issue

Small file problem

- Inefficiency of resource utilization
  - Significantly smaller than the HDFS block size (64MB)
- File, directory and block in HDFS is represented as **an object** in the namenode's memory, each of which occupies **150 bytes**
- HDFS is not geared up to efficiently **accessing** small files
  - Designed for streaming access of large files

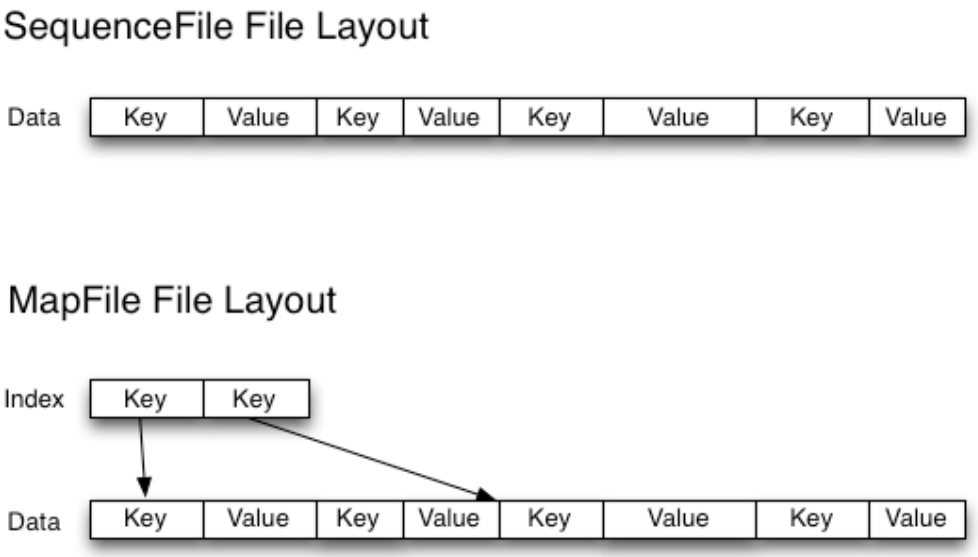
Small file solution

- Hadoop Archives (HAR)
  - Introduced to alleviate the problem of lots of files putting pressure on the namenode's memory
  - Building a **layered filesystem** on top of HDFS



Small file solution

- Sequence Files
  - Use the filename as the key and the file contents as the value



https://blog.cloudera.com/blog/2009/02/the-small-files-problem/

Summary

- Scalability
  - Provide scale-out storage capability of handling very large amounts of data
- Availability
  - Provide the ability of failure tolerance such that data would not lose on machine or disk fail
- Manageability
  - Provide mechanism for the system to automatically monitor itself and manage the massive data transparently for users
- Performance
  - High sustained bandwidth is more important than low latency

References

- S. GHEMAWAT, H. GOBIOFF, and S.-T. LEUNG, “The Google file system,” In *Proc. of the 19th ACM SOSP (Dec.* 2003)
- Hadoop.
  - http://hadoop.apache.org/
- NCHC Cloud Computing Research Group.
  - http://trac.nchc.org.tw/cloud
- NTU course- Cloud Computing and Mobile Platforms.
  - http://ntucsiecloud98.appspot.com/course_information