The Hadoop Distributed File System

Ruoxi Zhang

Introduction - The Apache Hadoop

- **Open source project**
- **Characteristic**
  - Distribute data and computation across clusters
  - Execute application jobs in parallel
- **Major contributors**
  - Yahoo!, Microsoft, Facebook, Cloudera
- **Hadoop-related projects at Apache**
  - MapReduce, HBase, Pig, Hive, ZooKeeper, Chukwa, Avro, and etc.

https://hadoop.apache.org/
https://hbase.apache.org/
https://pig.apache.org/
https://hive.apache.org/
https://zookeeper.apache.org/
http://chukwa.apache.org/
https://avro.apache.org/

The Hadoop Distributed File System (HDFS)

- Distributed file system
- MapReduce framework
- High I/O bandwidth for large datasets
- Data durability – replication
- Fault-tolerant

Architecture

- **Cluster**
  - **DataNodes** – application data
    - *block* & *replica* – file segments
    - *block's metadata* – checksums & generation stamp
  - **NameNode** – namespace tree & mapping of blocks to DataNodes
    - *inodes* – representation of files & directories; record attributes
    - *image* (in RAM) – file system's metadata
    - *checkpoint* (on disk) – a persistent record of the image
    - *journal* (on disk) – modification log of the image

Architecture

- **Cluster**
  - **CheckpointNode** – create a checkpoint
    - Downloads the current checkpoint & journal files from the active NameNode
  - **BackupNode** – create a checkpoint
    - Maintain the latest namespace state in its own storage

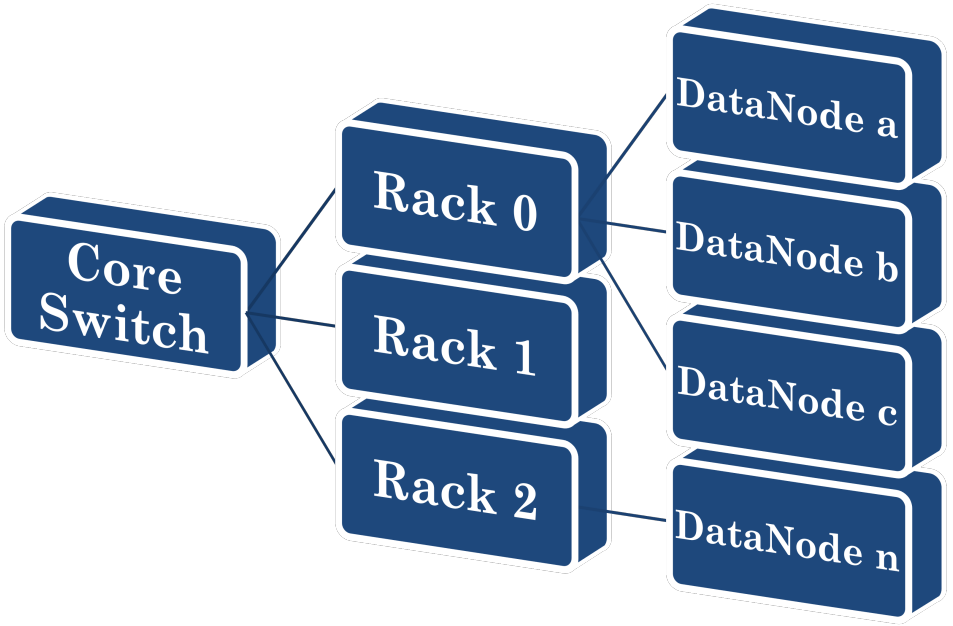
Architecture

- **Cluster**
  - **Applications** – users
    - Perform HDFS operations
      - read, write and delete files
      - create and delete directories
    - Use HDFS provided API
      - Reveal block locations
      - User defined replication factor
  - **HDFS Client** – interface
    - A black box to users

Startup

- **System Startup/Restart**
  - DataNodes → NameNode
    - *handshake* – verify namespace ID & software version
    - *register* – verify storage ID
    - *block report* – send block information periodically/on-demand
    - *heartbeat* – periodically (shorter interval) ping
  - NameNode → DateNodes
    - *heartbeat reply* – instructions
  - *layout version* – data representation formats
  - Read the checkpoint & apply journals

Block Placement



- Nodes are spread across multiple *racks*
  - Nodes of a rack share a *switch*
  - Switches are connected to *core switch(es)*
- Bandwidth within a rack > bandwidth across racks
- NameNode assigns and resolves rack locations of DataNodes

Replicate for Durability

- *replication factor* – 3 or user-defined
- The NameNode allocates new blocks and replicates them based on a block/replica placement policy.
  - At most 1 replica of a block at a node.
  - At most 2 replicas of a block in a rack.
- Factors: write/read cost, reliability, availability, and aggregation bandwidth

Replication Management

- **Over-replicated Block** – the NameNode removes replicas based on available storage and rack location
- **Under-replicated Block** – a dedicated thread replicates blocks from a priority queue
- **Balancer** [admin] – balances disk storage usage of DataNodes
- **Decommissioning** [admin] – safely removes a DataNode
- **Block Scanner** [DataNode] – scans replicas and verifies checksums to detect corruptions

Interactions

- Single writer & multiple readers – lease
- Write – push data to the DataNodes pipeline
  - *hflush* – ensure content visibility before file closed
  - TCP based protocols
    - Client-side buffer
    - Three stages: setup, transfer, and close
    - Window size and acknowledgment

Fault-tolerance

- **System Corruption**
  - Reasons: software upgrades, software bugs, or human mistakes
  - Solution: roll back the entire HDFS cluster to the snapshot state
  - **Snapshot** [exactly one] – NameNode
    - Save the current namespace and storage state
  - **Local Snapshot** – DataNodes
    - Save a copy of the storage directory and hard links to blocks
  - **Fault-tolerant** – Rack or Core Switch
    - Replication & deliberately restart
- **Data Corruption** – the client verifies checksums of blocks when read

Implementation

- Yahoo!'s large HDFS cluster [1]
  - ~3500 nodes
  - 9.8 PB storage
  - 60 million files
  - 63 million blocks
  - The probability of losing a block < 0.005 per year
  - Linear total bandwidth

Implementation

- Testing the I/O operations
  - **DFSIO Benchmark** – average throughput
    - the same application, job, and data
  - **Metric Collection System** – average throughput
    - Many application, multiple jobs, and different data
  - **Gray Sort Competition** – the best throughput
  - **NNThroughput Benchmark** – NameNode throughput
    - Many clients, the same job, on a single NameNode

Other Implementation Details

- Separated **permissions** for files and directories
- **Identity**
  - Weak – query the local OS for user identity
  - Strong – endpoint authentication to verify user identity and system identity
- Application level **fairness**
- Control the **size of the namespace**
  - *quota* for directories' space allocation
  - HAR to achieve a large number of small files under a common directory

Future Work

- Automated **failover** solution – Zookeeper
- **Scalability** of the NameNode – archive tool, partially stored namespaces in RAM, distributed NameNodes
- Larger clusters and **cooperation** between clusters

HDFS as A Hadoop Project

- Big data storage
- Parallel processing
- High-throughput access
- Multiple copies of data
- Restart and backup

Reference

- K. Shvachko, H. Kuang, S. Radia, R. Chansler, The Hadoop Distributed File System, *IEEE 26th Symposium on Mass Storage Systems and Technologies*, 2010.

Q&A