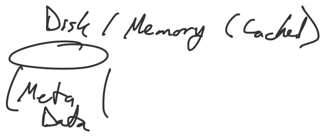
# HDFS Architecture

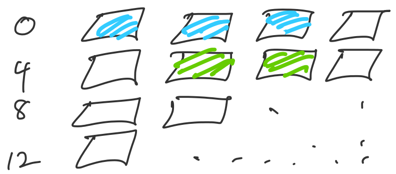
Tuesday, January 10, 2023 7:09 AM

- Distributed File Systems (DFS)
  - Overview
    - Storage partitioned across multiple machines
    - Distributed systems
      - CAP
        - Consistency: read() followed after write()
        - Availability: nodes allowed for read/write access
        - Partitioning: node/net split & connectivity

- HDFS
  - DFS
  - Filesystem Abstraction /API over Blob/object Storage
    - ↳ S3, Azure ... (HTTP/web API)
  - Use Cases
    - Large Files (very!) → GB/TB/PB+
      - ↳ Not good for too many (small) files
    - Fault Tolerance → Replication of files/blocks across nodes
      - ↳ K-safety $k \geq 2$
    - Read-Only/Read-Heavy Workloads (WORM)
      - ↳ No edits, overwrites only, appends allowed
    - High Throughput / Not Low-Latency
      - ↳ Batch jobs/tasks → OLAP

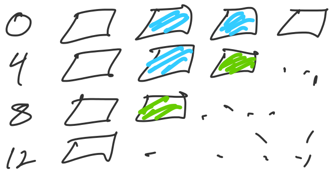
- Blocks
  - Traditional FS *Blocks ~ KB

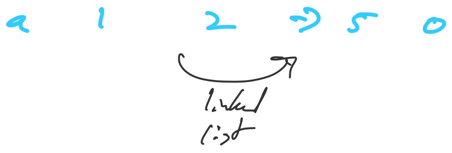




| file | start | length |
|---|---|---|
| a | 0 | 3 |
| b | 5 | 2 |

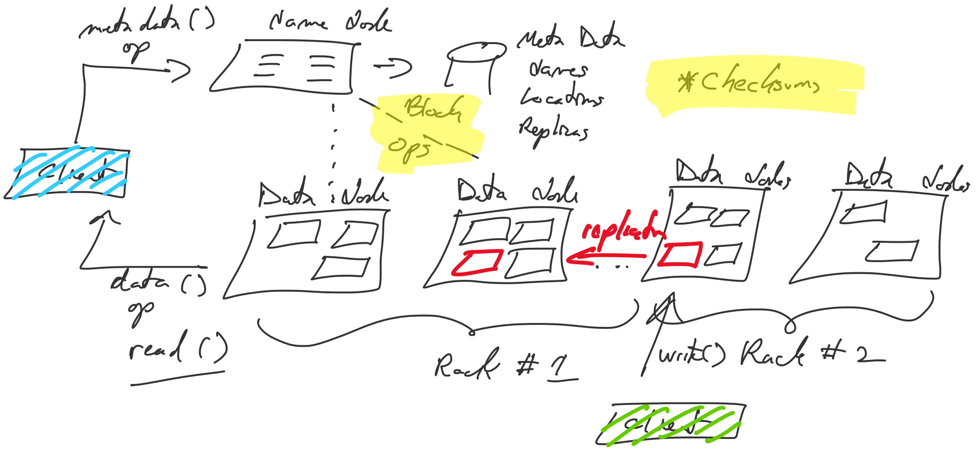
* Fragmentation





- DFS (HDFS)
  - Blocks *Large ~ 64MB
  - Block-Level Operations
    - read / write
    - copy / replicate (k-safety)

*Replication
- k = 3 (default) : original + 2copies per block
- replicas stored different nodes / racks
- Block size & Replication Factor <—> File Size



- Nodes
  - Two types of data: Data, Meta Data
    - ↳ 2 node types
  - Name Node (1+) : Meta Data
    - FS
      - Directory Tree
      - Permissions
      - Block Locations
      - checksums...

      } Cache & Persist
    - Client
      - Block Locations / Data Node Addresses
      - Alternative Locations / Failover
- Data Node (N+) : Data
  - Files consisting of Blocks
  - Blocks are unaware of file-level-membership
  - Block ops: new blocks, replication, notifications (delete, etc...)
  - Heartbeats to Name Node!

* write new file:

1. add block to Name Node (Meta Data)
2. write block to local Data Node
3. background block operations → replications
    - orchestrated by Name Node w/ local Data Node as source