Big Data (DS-GA 1004) – Lecture 4 Finals Preparation Notes

Fully based on Week 4 Slides + Lecture Transcript + Expanded Knowledge

Spring 2025

1 Distributed Storage: Introduction to HDFS

1.1 Class Culture Reminder

- ”I don’t believe it unless I implement it in code and understand the algorithm”: Computer Science (CS) view.
- ”I don’t believe it unless I see proof and derivation”: Statistics (IDS) view.
- ”I need to see experimental design and data”: Data Science (DS) view.

2 Last Week vs. This Week

- Last week: Introduction to distributed computation and MapReduce.
- This week:
  - CDS
  - Data storage concepts
  - Distributed data storage
  - The Hadoop Distributed File System (HDFS)

3 Confusion, Doubt, & Struggle: Hash Functions

- Take input (message) → output a fixed-size string.
- Properties:
  - Deterministic
  - Fast computation
  - Small input changes → Large output changes
  - Pre-image resistant
  - Collision resistant

3.1 Examples

- Function $f(x) = x \mod 10$
  - $f(7) = 7$, $f(42) = 2$, $f(420) = 0$, $f(2777) = 7$
  - Not collision resistant (e.g., 7 and 2777 map to 7).
- Function $g(x) = ([ax] + b) \mod p$, with constants $a = 3$, $b = 5$, prime $p = 97$
  - More robust; harder to invert.

3.2 Hashing Use Cases

- Error detection (checksum).
- Password storage (cryptographic hashes).
- Hash tables (for fast lookups).

4 Data Storage Systems

4.1 File Systems and Hard Disks

- Files → broken into **blocks** → mapped onto **sectors**.
- Sectors: Smallest unit (512 to 4096 bytes).
- Moving read head limits throughput → Files can fragment.

4.2 RAID (Redundant Array of Inexpensive Disks)

- Multiple disks appear as one to OS.
- Goals:
  - Capacity
  - Reliability
  - Throughput
- Different RAID levels trade off these goals differently.

4.3 Common RAID Levels

- Striping only (no redundancy, capacity scales linearly).
- Full redundancy (RAID 1: mirrored disks).
- RAID 5: Distributed parity (balance between reliability and capacity).

4.4 RAID 5 and Parity Bits

- XOR based parity: Recover lost data blocks.
- Example: Disk0 $\oplus$ Disk1 = Parity.

5 Distributed Data Storage

5.1 Why Not Just RAID?

- RAID is for single machines.
- For **distributed computation**, we need distributed storage.
- Communication over network $\rightarrow$ Bottleneck.

5.2 Simple (Bad) MapReduce Implementation

- Central node stores all data.
- Data transfer becomes network bottleneck.

5.3 Extreme Local Storage

- Replicate all data on all nodes $\rightarrow$ Wasteful and expensive.

5.4 Distributed File Systems

- Reasonable trade-off between redundancy and communication.
- Key Design Factors:
  - Minimize communication.
  - Redundancy control.
  - Data locality.
  - Access patterns (small programs, large datasets).

6 HDFS (Hadoop Distributed File System)

6.1 Key Features

- Distributed, redundant storage.
- Optimized for write-once, read-many patterns.

6.2 Hadoop Framework

- **MapReduce**: Processing engine.
- **YARN**: Resource manager.
- **HDFS**: Storage layer.

6.3 Using HDFS

- Files stored through HDFS commands.
- Sits *above* the native file system.
- Accessed using commands like `hadoopfs -command`.

6.4 HDFS Node Types

- **Name Node**: Metadata manager (file → blocks → data nodes).
- **Data Nodes**: Actually store the blocks.

6.5 Name Node Details

- Maintains the namespace.
- Stores metadata only.
- Failure of name node → Catastrophic.

6.6 Data Node Details

- Store blocks as files.
- Maintain checksum and generation stamp for each block.
- Send periodic heartbeats to name node.

6.7 Writing to HDFS: Block Addition

1. Client asks name node for block allocation.
2. Name node returns list of data nodes.
3. Client sends block to first data node (DN1).
4. DN1 stores and forwards to DN2.
5. DN2 stores and forwards to DN3.
6. DN3 stores and acknowledges → Client finalizes.

6.8 Fault Recovery

- Checkpoints capture name node's state.
- Data node failures tolerated up to replication factor.

6.9 POSIX Non-Compliance

- Append-only writes.
- Simplifies replication and consistency.
- Not designed for small frequent updates.

7 HDFS and CAP Theorem

- Consistency: Name node maintains global consistency.
- Availability: Guaranteed if name node is alive.
- Partition Tolerance: Depends on replication factor.

8 Wrap-up on HDFS

- Distributed, redundant file system optimized for large, immutable datasets.
- Critical component of scalable data infrastructure.

9 Next Week

- Big data infrastructure: General principles and NYU specifics.