Hadoop Distributed File System

Big Data Management

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

DTIM
www.essi.upc.edu/dtim

Knowledge objectives

1. Recognize the need of persistent storage
2. Enumerate the design goals of GFS
3. Explain the structural components of HDFS
4. Name three file formats in HDFS and explain their differences
5. Recognize the importance of choosing the file format depending on workload
6. Explain the actions of the coordinator node in front of chunkserver failure
7. Explain a mechanism to avoid overloading the master node in HDFS
8. Explain how data is partitioned and replicated in HDFS
9. Recognize the relevance of sequential read

Understanding objectives

1. Choose the format for an HDFS file based on heuristics
2. Estimate the data retrieved by scan, projection and selection operations in SequenceFile, Avro and Parquet

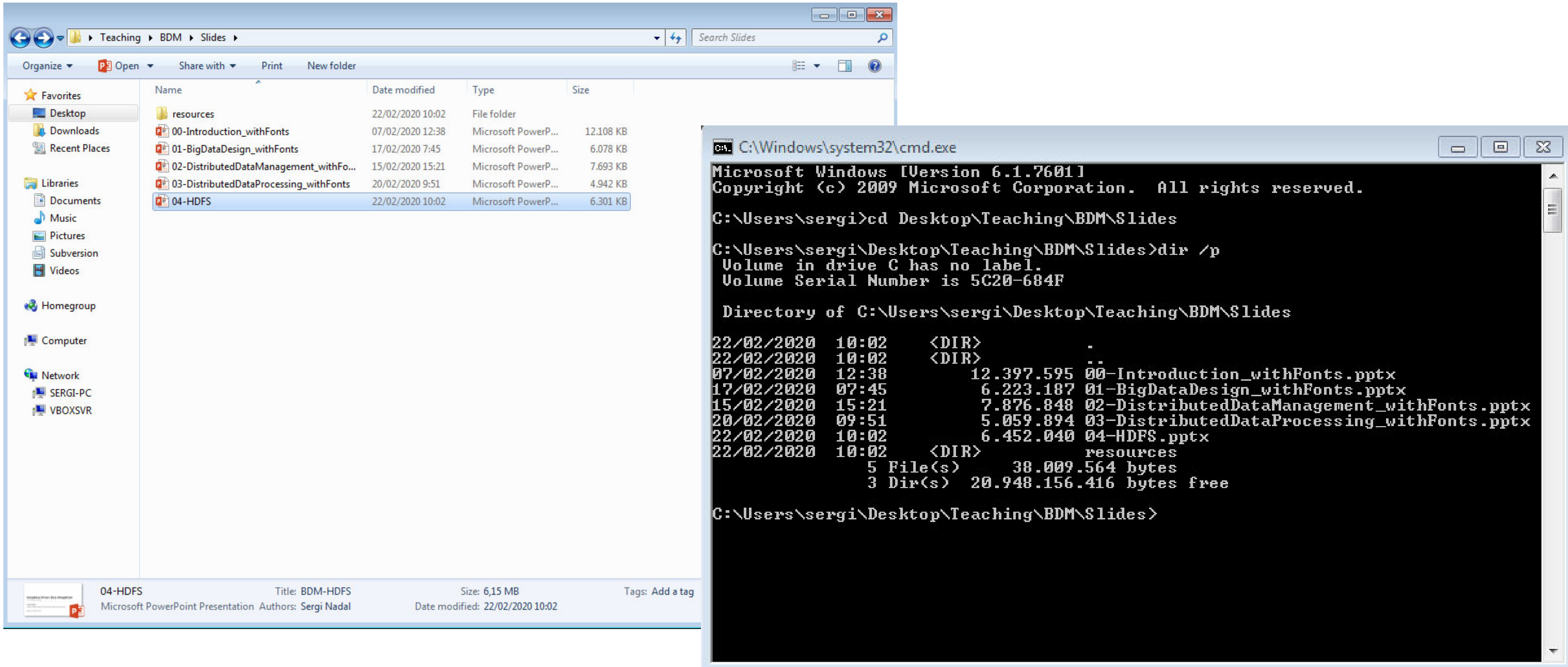
(Distributed) File Systems

Google File System

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

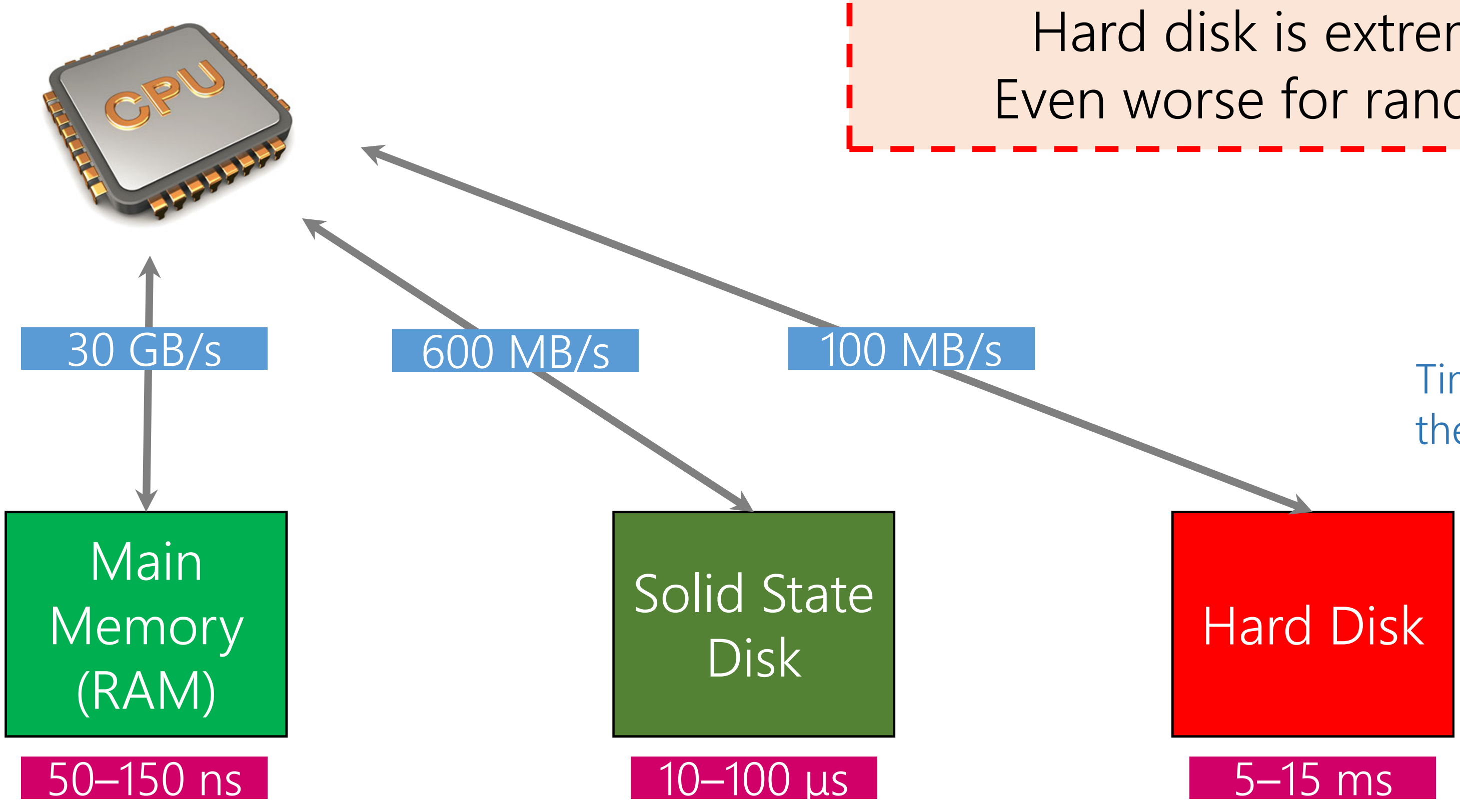
DTIM
www.essi.upc.edu/dtim

What is a file system?



Time to bring data (approximations)

Hard disk is extremely slow!
Even worse for random access.



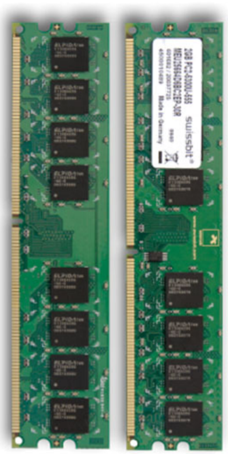
Data transfer

Time to move the data from the disk to the processor

Latency

Time to start an I/O (i.e., access a block)

Reasons to keep using HDDs



Main Memory

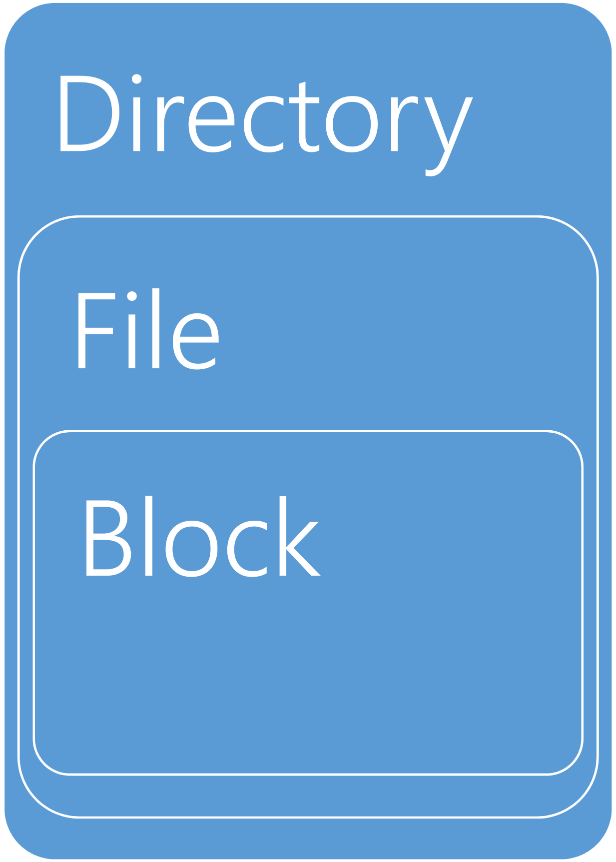
Faster ✓

✓ Cheaper
✓ Persistant

Hard Drive Disk

Functionalities provided by a FS

- Creates a hierarchical structure of data
- Splits and stores data into files and blocks
- Maintains directories/files metadata
  - Size, date of creation, permissions, …
- Provides interfaces to read/write/delete



Distributed File Systems

- Same requirements, different setting
  1. Files are huge for traditional standards
  2. Most files are updated by appending data rather than overwriting
     - Write once and read many times
  3. Component failures are the norm rather than the exception
- Google File System (GFS)
  - The first large-scale distributed file system
  - Capacity of a GFS cluster

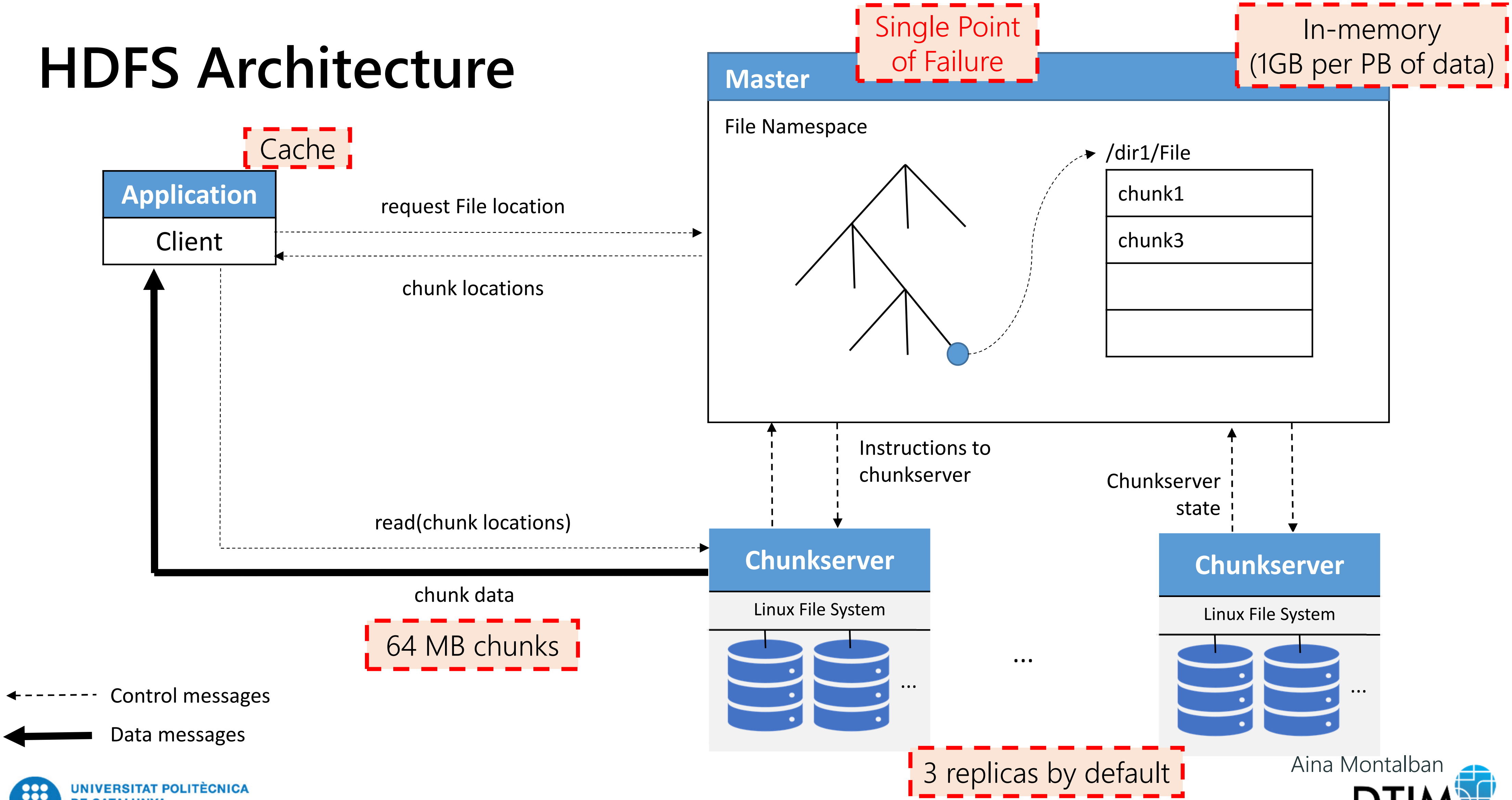
Capacity	Nodes	Clients	File
10 PB	10.000	100.000	100.000.000

Design goals of GFS

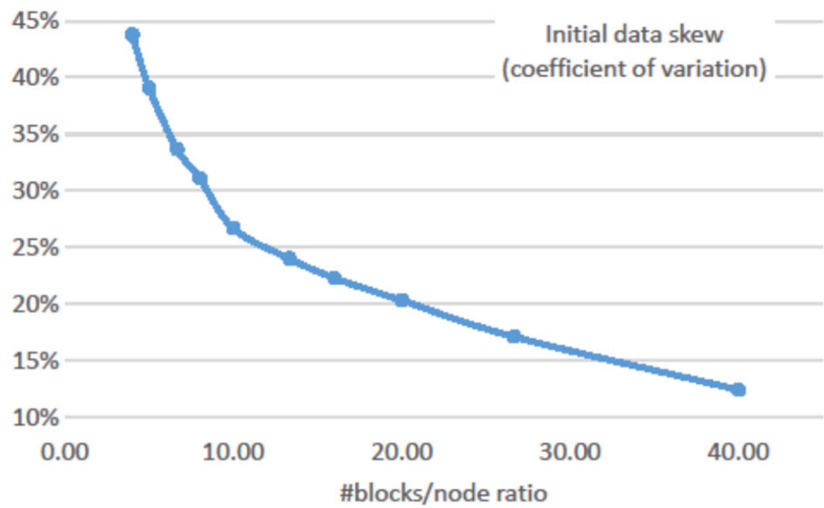
- Efficient management of files
  - Optimized for very large files (GBs to TBs)
- Efficiently append data to the end of files
  - Allow concurrency
- Tolerance to failures
  - Clusters are composed of many inexpensive machines that fail often
    - Failure probability (2-3/1.000 per day)
- Sequential scans optimized
  - Overcome the high latency of HDDs (5-15ms) compared to main memory (50-150ns)

GFS Architecture

HDFS Architecture



Other features



- Rebalance
  - Avoids skewness in the distribution of chunks
- Deletion
  - Moves a file to the trash (hidden)
    - Kept for 6h
  - *expunge* to force the trash to be emptied
- Management of stale replicas
  - Coordinator maintains versioning information about all chunks

Data Design

Challenge I

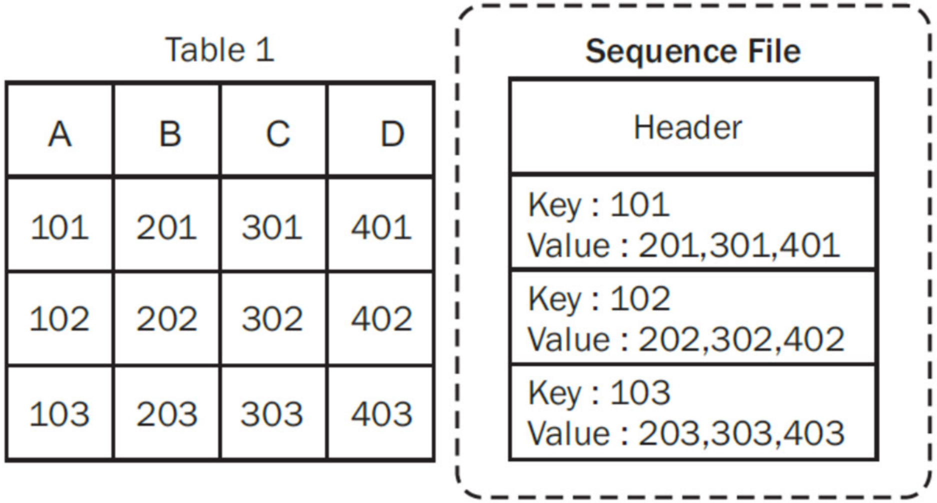
Storage layouts

"Jack of all trades, master of none"

- Different workloads require different layout
  - Horizontal
    - For scan-based workloads
  - Vertical
    - For projection-based workloads (reads a subset of columns)
  - Hybrid
    - For projection- and predicate-based workloads (reads a subset of columns or rows)

Horizontal layout – Sequence File

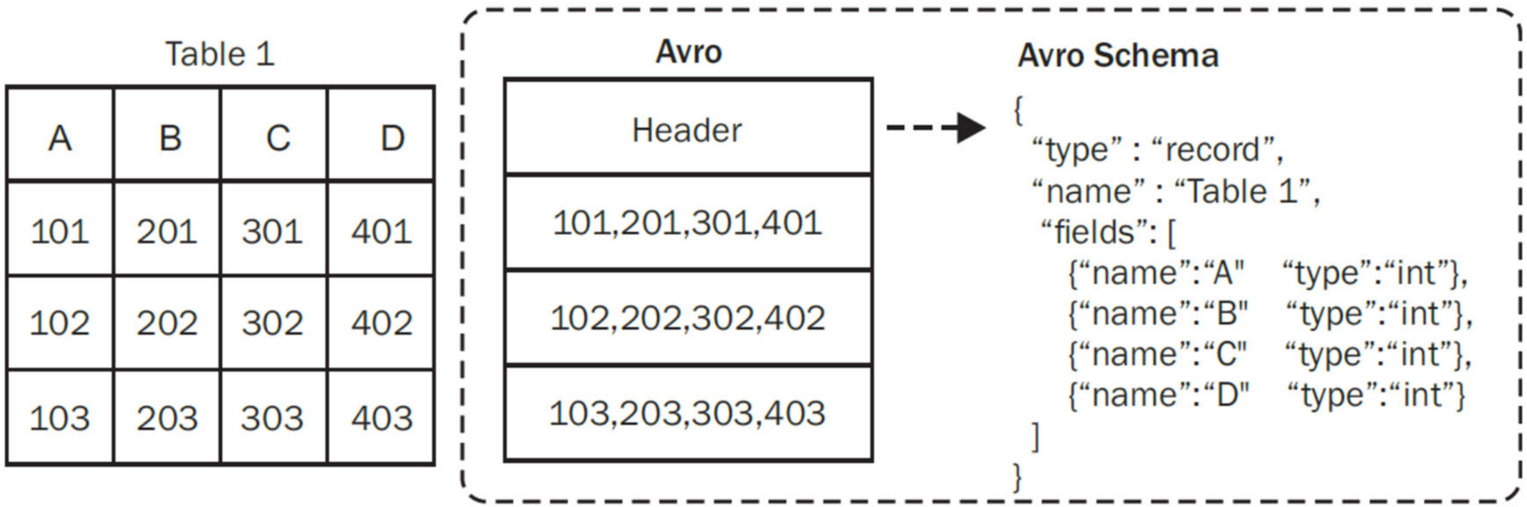
- Records of binary key-value pairs



- Compression
  - Uncompressed
  - Record-compressed
  - Block-compressed
    - “block” is the compression unit – a block of records (not a chunk)
      - 1 MB default

Horizontal layout - Avro

- Binary encoding of (compressed) rows
- The header contains a schema encoded in JSON

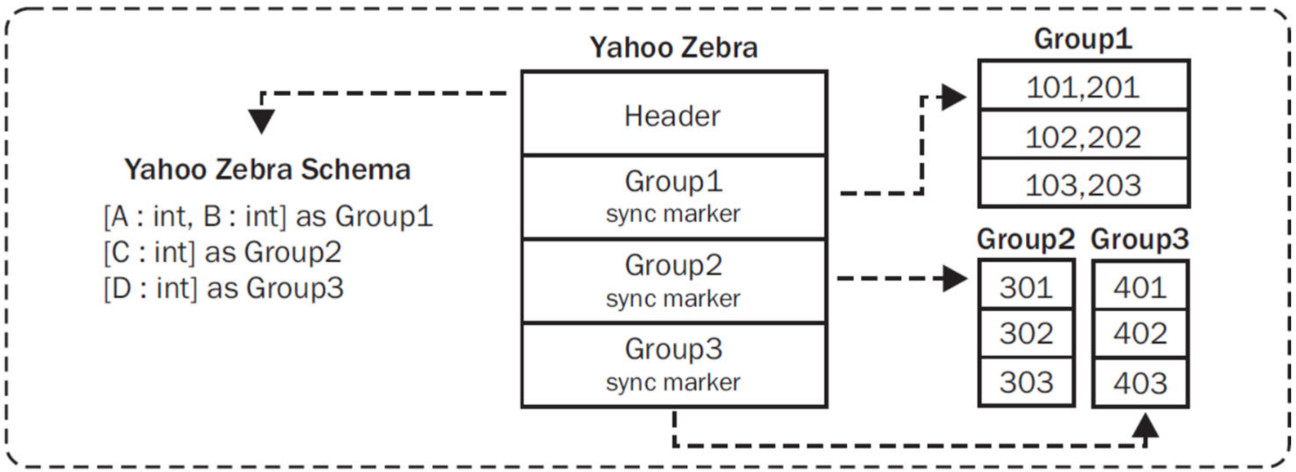


Vertical layout - Zebra

- The header contains the definition of groups
  - Each group contains a set of columns
  - Widely benefits from compression
- Not really used in practice

Table 1

A	B	C	D
101	201	301	401
102	202	302	402
103	203	303	403

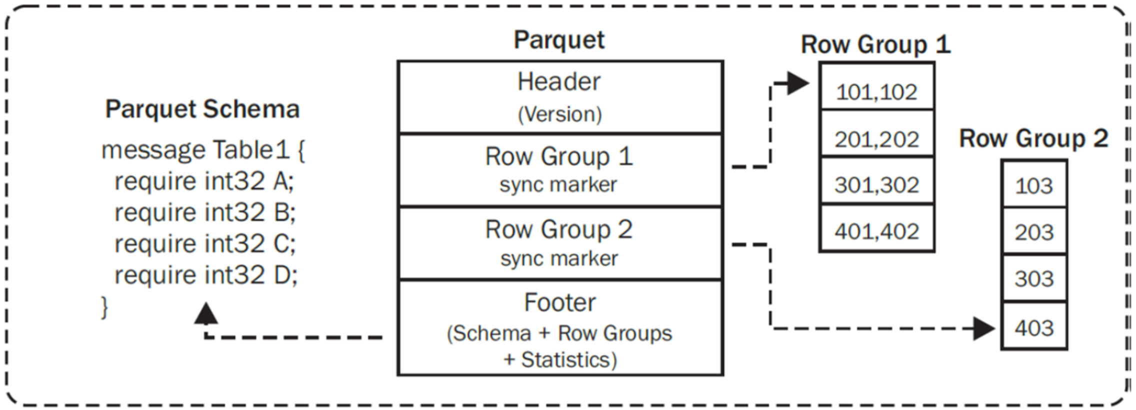


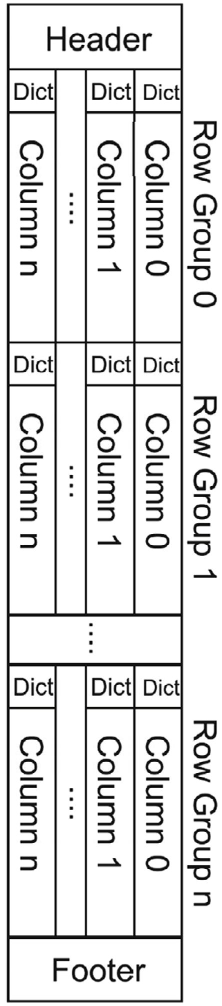
Hybrid layout - Parquet

- Row groups (RG) - horizontal partitions
  - Data vertically partitioned within RGs
- Statistics per row group (aid filtering)
  - E.g., min-max

Table 1

A	B	C	D
101	201	301	401
102	202	302	402
103	203	303	403





Comparison of data formats

Features	Horizontal		Vertical	Hybrid	
	Sequence Files	Avro	Yahoo Zebra	ORC	Parquet
Schema	No	Yes	Yes	Yes	Yes
Column Pruning	No	No	Yes	Yes	Yes
Predicate Pushdown	No	No	No	Yes	Yes
Indexing Information	No	No	No	Yes	Yes
Statistics Information	No	No	No	Yes	Yes
Nested Records	No	No	Yes	Yes	Yes

Rule-based choice (heuristic)

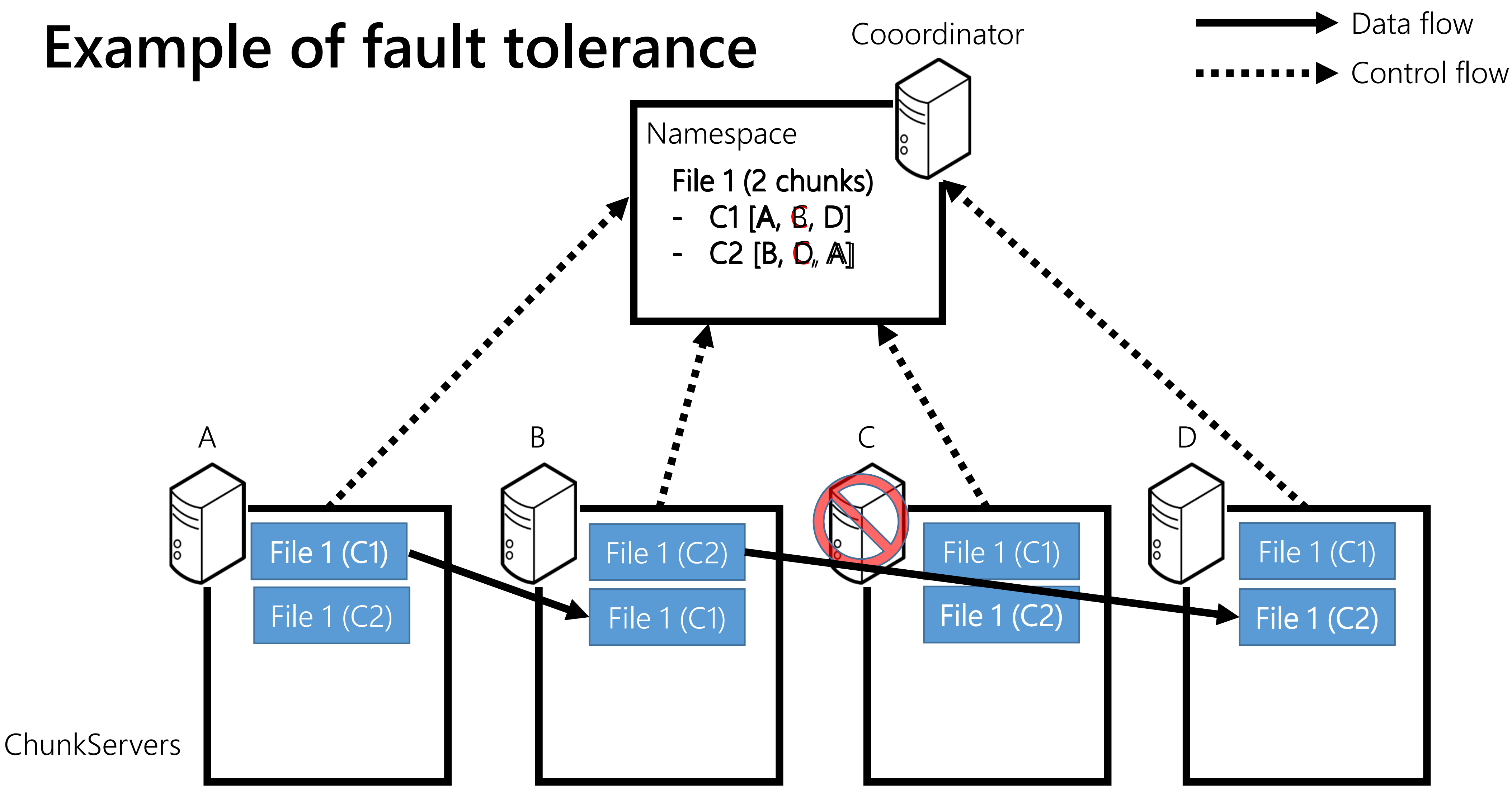
Given a flow represented as DAG(V, E)

- SequenceFile
  - $size(getCol(v)) = 2$
- Parquet
  - $\exists e \in O(v), getType(e) = \{AggregationOps\}$
  - $\exists e \in O(v), getCol(getOP(e)) \subset getCol(v)$
- Avro
  - $\forall e \in O(v), getCol(getOP(e)) = getCol(v)$
  - $\exists e \in O(v), getType(e) \in \{Join, CartesianProduct, GroupALL, Distinct\}$

Fault tolerance

- Managed from the coordinator
  - It expects to receive every 3 seconds a *heartbeat* message from chunkservers
- If a chunkserver does not send a heartbeat for 60 seconds a fault is declared
- Corrective actions
  - Update the namespace
  - Copy one of the replicas to a new chunkserver
    - Potentially electing a new primary replica

Example of fault tolerance



Catalog Management

Challenge II

Client caching

Cache miss

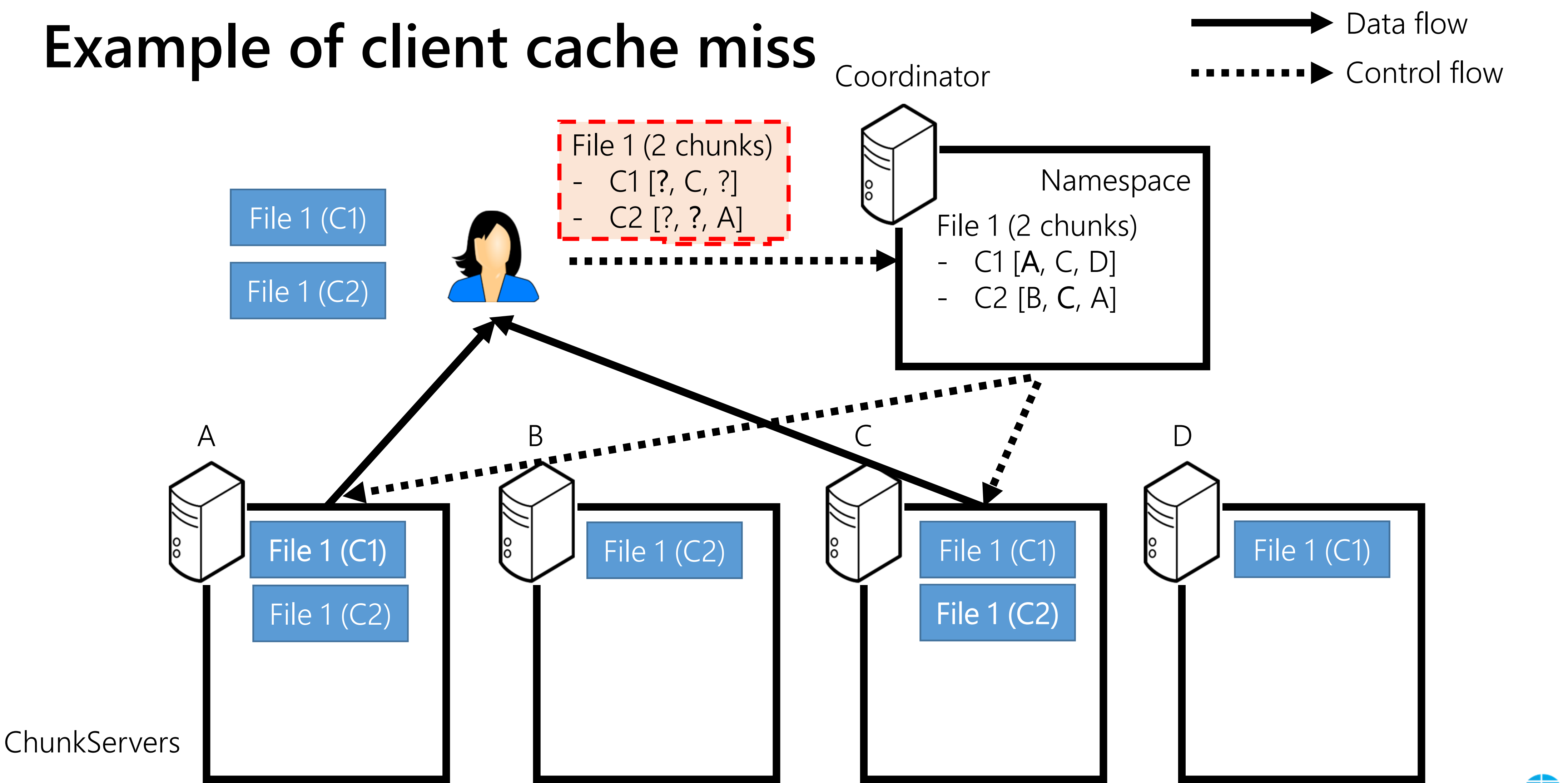
1. The client sends a READ command to the coordinator
2. The coordinator requests chunkservers to send the chunks to the client
   - Ranked according to the closeness in the network
3. The list is cached in the client
   - Not a complete view of all partitions

Cache hit

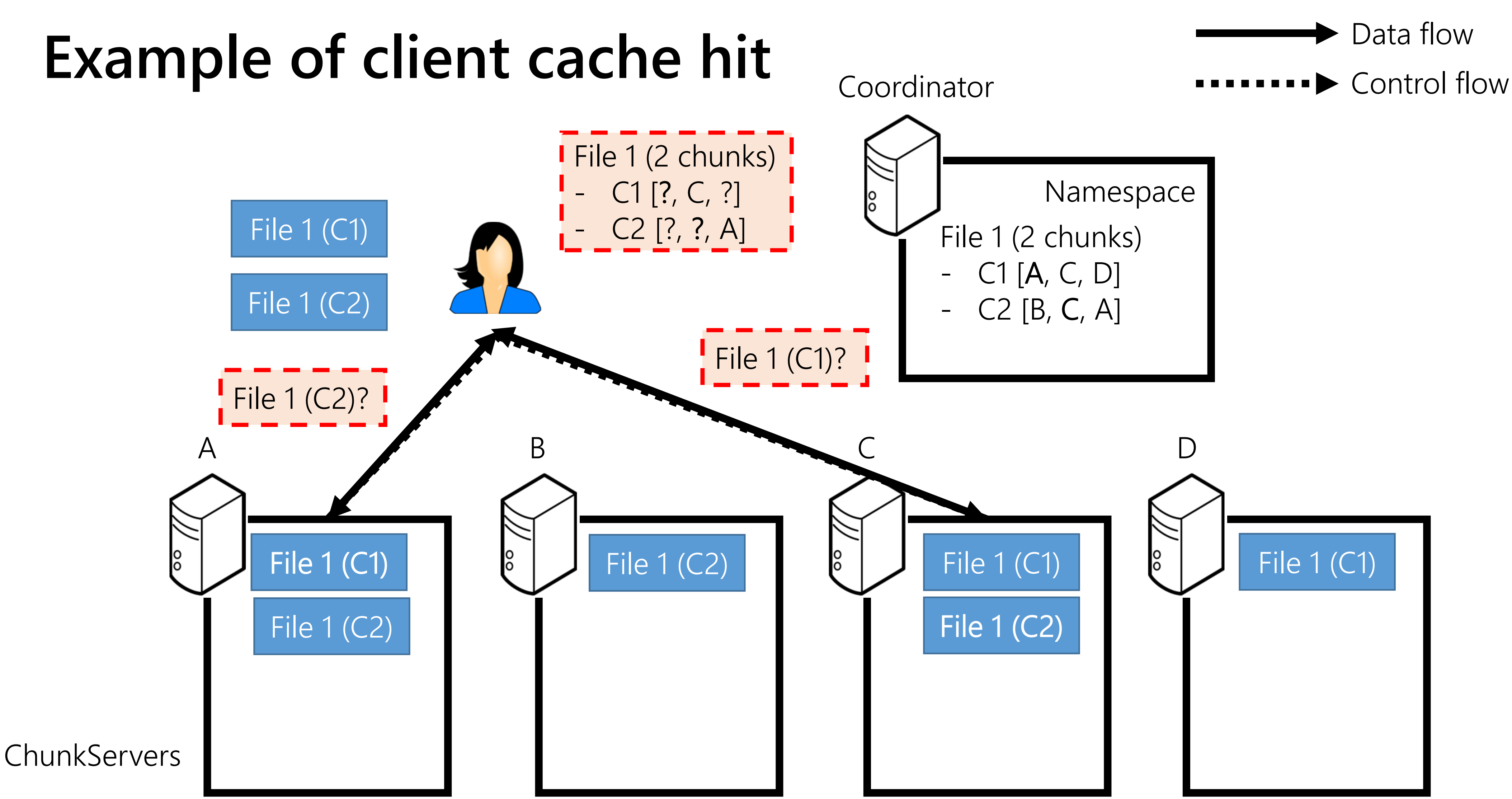
1. The client reads the cache and requests the chunkservers to send the chunks

Avoid coordinator bottleneck
+
One communication step is saved

Example of client cache miss



Example of client cache hit

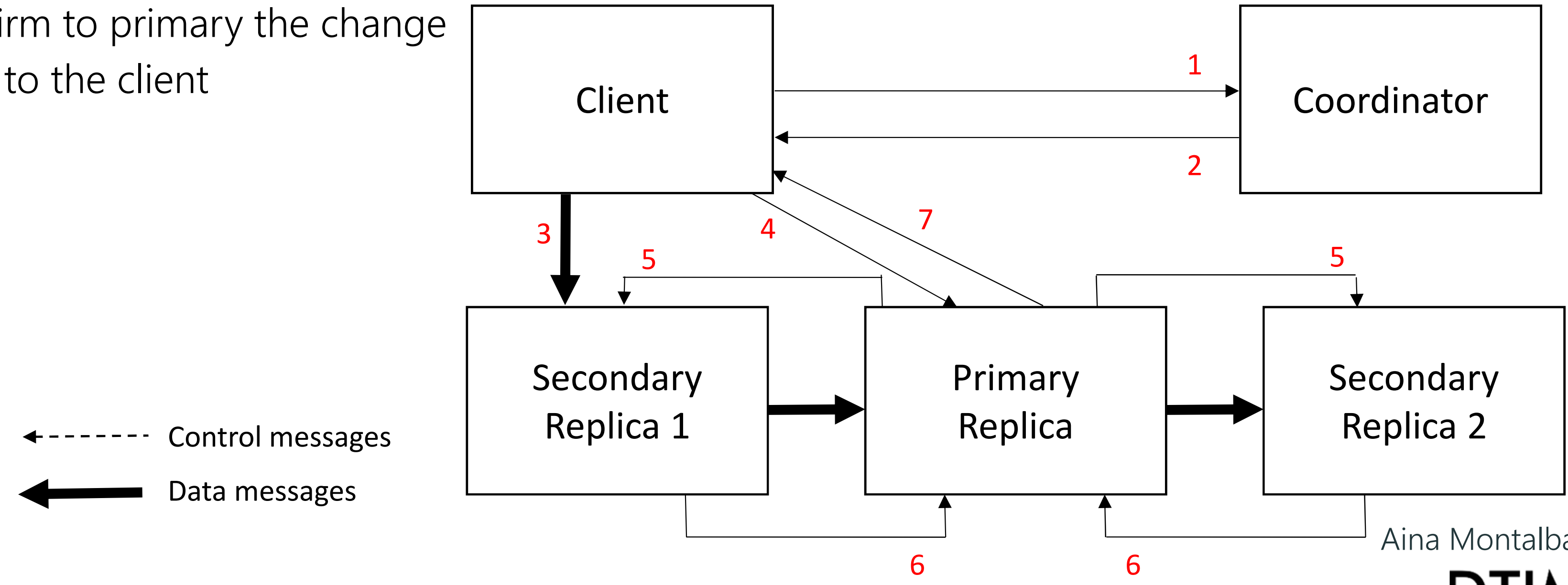


Transaction Management

Challenge III

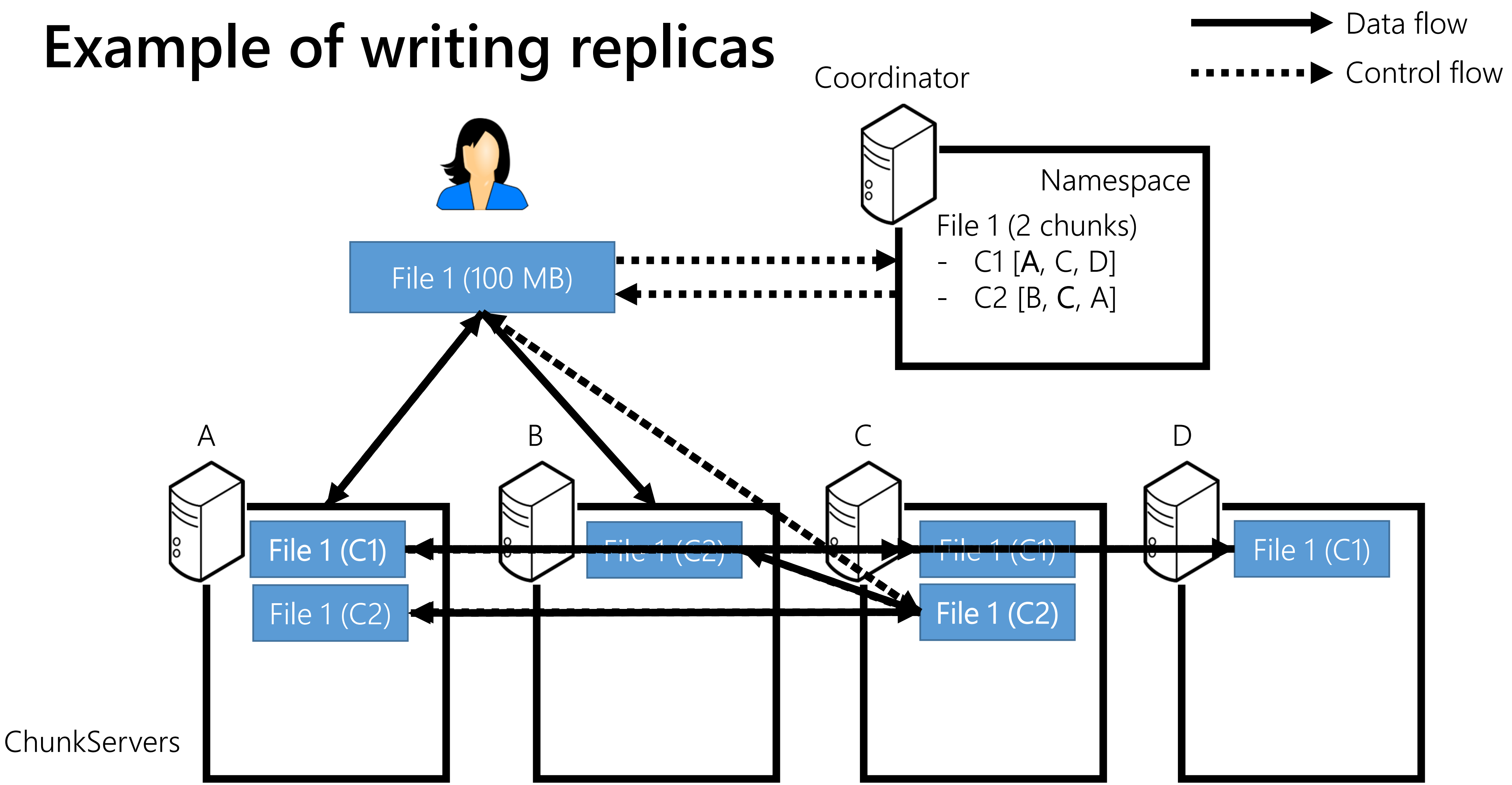
Writing replicas

1. The client requests the list of the replicas of a file
2. Coordinator returns metadata
3. Client sends a chunk to the closest chunkserver in the network
   - This chunk is pipelined to the other chunkservers in the order defined by the master (leases)
4. Client sends WRITE order to primary replica
5. Primary replica sends WRITE order to secondary replicas
6. Secondaries confirm to primary the change
7. Primary confirms to the client



Aina Montalban

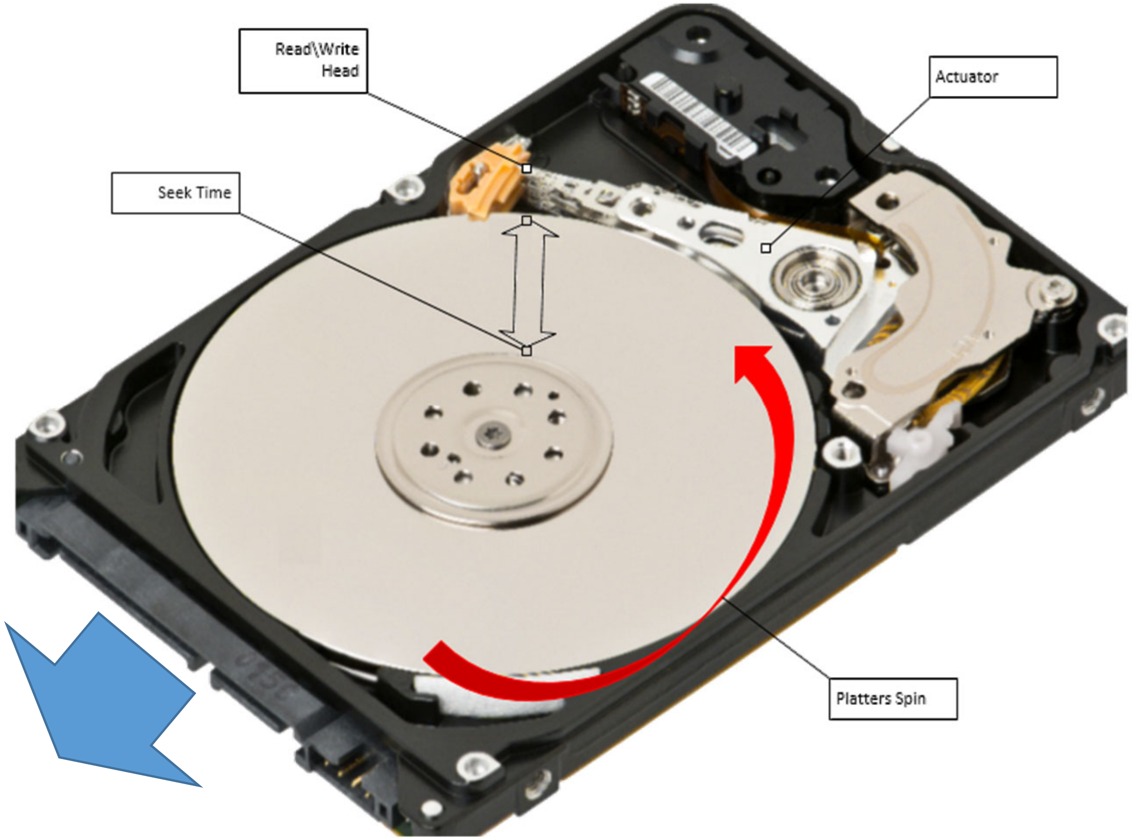
Example of writing replicas



Query processing

Challenge IV

HDDs costs



Rotational Latency

The amount of time it take for the platters to spin the data under the head (measured in RPM)

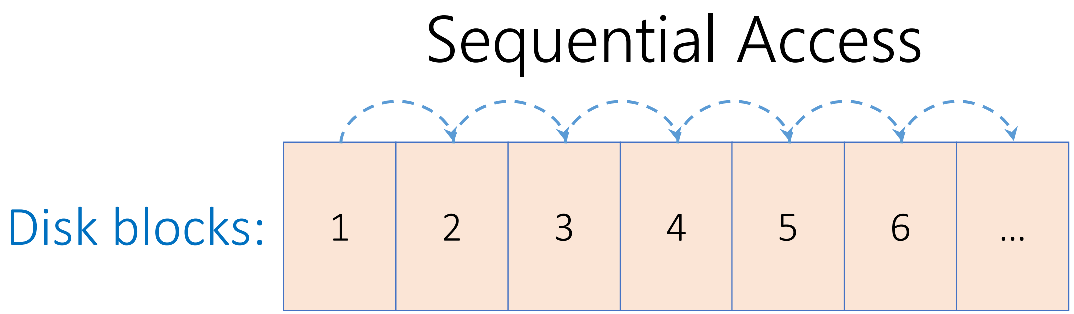
Seek Time

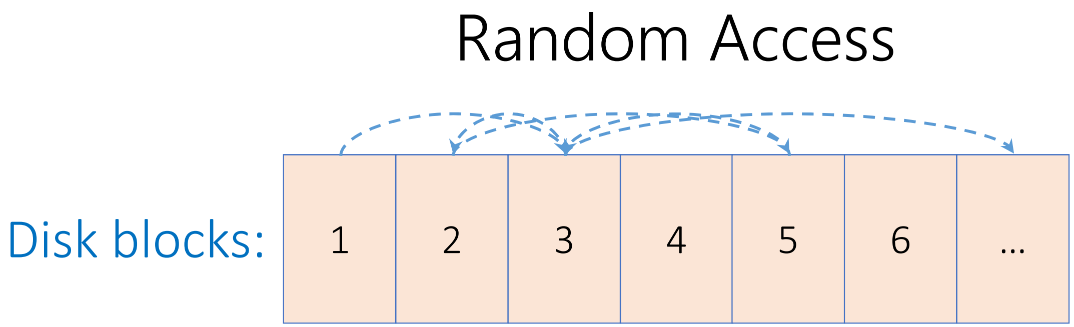
Time it takes for the ReadWrite head (mechanical arm) to move between sectors on the disk

Transfer Time

Time it takes for requests to get from the system to the disk (depends on the block size, e.g., 4KB)

Sequential vs. Random access





Cost of accessing data (approximations)

- Sequential reads
  - Option to maximize the effective read ratio
    - Depends on DB design
  - Enables pre-fetching

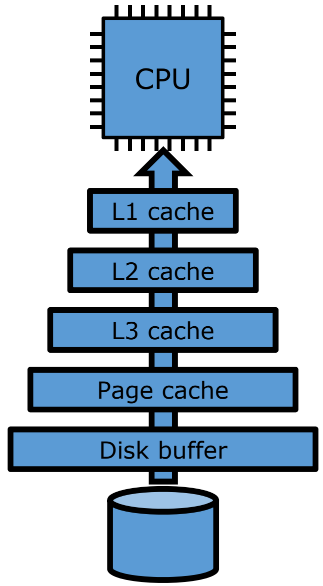
  Cost = seek+rotation+n*transfer

- Random Access
  - Requires indexing structures
  - Ignores data locality

  $\text{Cost}_{\text{single cylinder files}}$ = seek+n*(rotation+transfer)

  $\text{Cost}_{\text{multi-cylinder files}}$ = n*(seek+rotation+transfer)

seek	~12ms
rotation	~3ms
transfer (8KB)	~0.03ms



Closing

Summary

- GFS architecture and components
- GFS main operations
  - Fault tolerance
  - Writing files and maintenance of replicas
  - Reading files
- HDFS file formats
  - Horizontal
  - Vertical
  - Hybrid

References

- S. Ghemawat et al. *The Google File System*. OSDI'03
- K. V. Shvachko. *HDFS scalability: the limits to growth*. 2010
- S. Abiteboul et al. *Web data management*. Cambridge University Press, 2011
- A. Jindal et al. *Trojan data layouts: right shoes for a running elephant.* SOCC, 2011
- F. Färber et al. *SAP HANA database: data management for modern business applications.* SIGMOD, 2011
- V. Raman et al. *DB2 with BLU Acceleration: So Much More than Just a Column Store*. VLDB, 2013
- D. Abadi, et al. *Column-stores vs. row-stores: how different are they really?* SIGMOD Conference, 2008
- M. Stonebraker et al. *C-Store: A Column-oriented DBMS*. VLDB, 2005
- G. Copeland and S. Khoshafian. *A Decomposition Storage Model*. SIGMOD Conference, 1985
- F. Munir. *Storage Format Selection and Optimization of Materialized Intermediate Results in Data-Intensive Flows*. PhD Thesis (UPC), 2019
- A. Hogan. *Procesado de Datos Masivos* (U. Chile)