Fault Tolerant Dataflow Platform

Distributed Systems Project - A.Y. 2023/24
Luca Lain & Lorenzo Morelli

Content:

- Assumptions
- Design Choices
- Network Topology
- Fault Tolerance Mechanism
- Demo

Assumptions

- Coordinator is reliable, while workers may crash at any time.
- Links and channels are reliable.
- HDFS is reliable.
- *Reduce* operation is always the last one.
- Coordinator receives the input files.
- Input files are well formatted.

Design Choices

- **Batch** approach.
- Distributed File System (using HDFS) with network TCP channels.
- Input files are divided by key.
- At the beginning, # workers = # partitions.
- Every partition is a file containing a unique set of keys.
- Local checkpointing is based on the amount of processed data.
- Coordinator may accept multiple programs (sets of data and operations), which are concurrently processed.
- One-Phase / Two-Phases:
    a. if changeKey && reduce -> Two-Phases
    b. else -> One-Phase

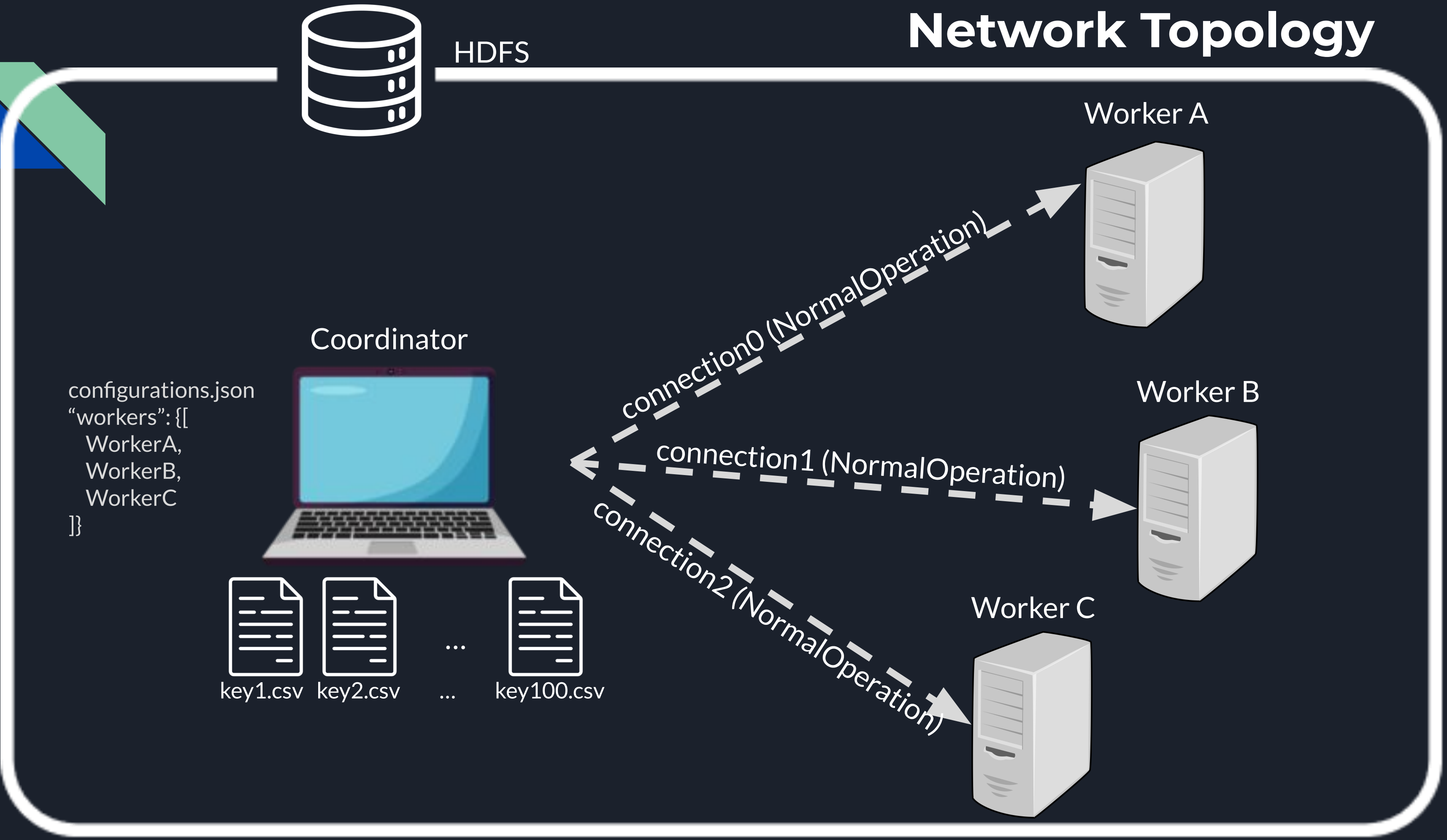
One-phase

1. Coordinator sends a "NormalOperations" message to each worker, which includes the path of the file to be processed.
2. Workers concurrently compute the same operation(s) on different data.
3. Each worker sends a “EndComputation” message to the coordinator.
4. Coordinator reads the result of each worker from HDFS and merges them into a file.

Two-phases

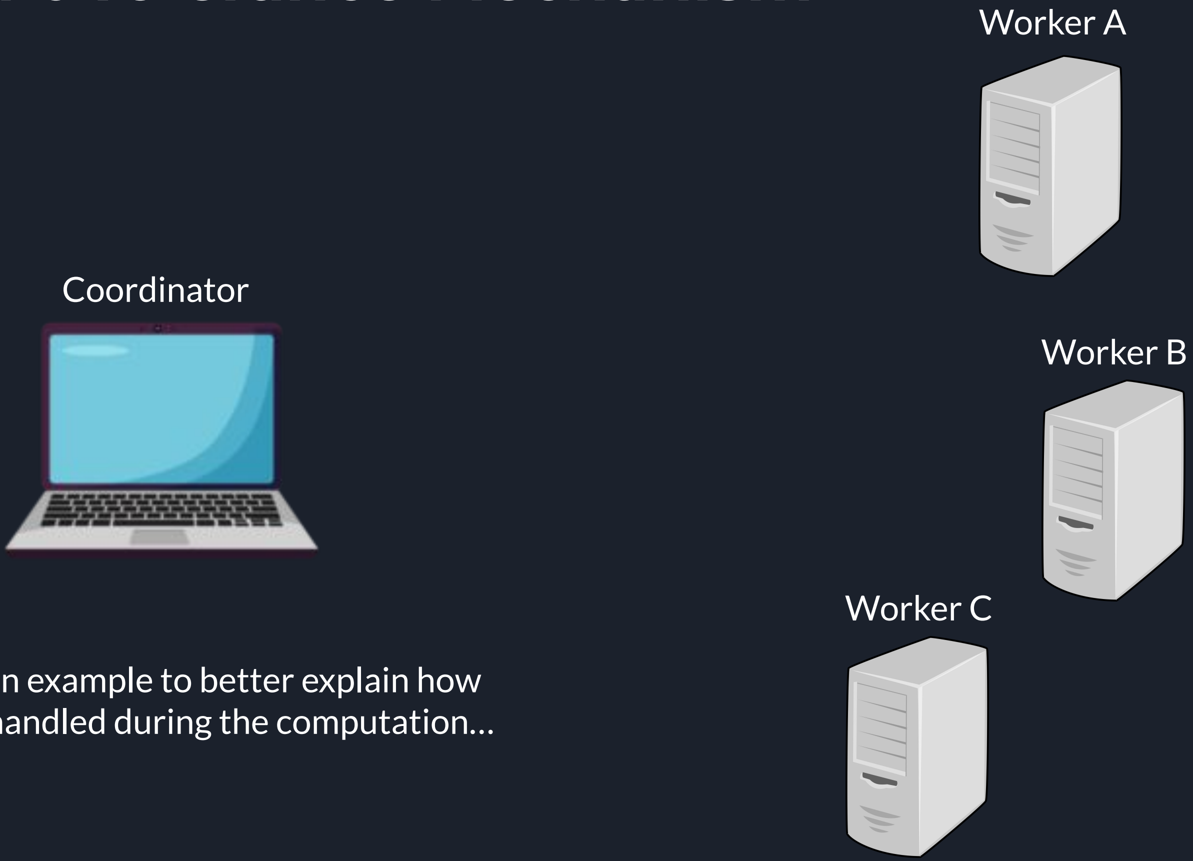
1. Coordinator sends a "NormalOperations" message to each worker, which includes the path of the file to be processed.
2. Workers concurrently compute the same operation(s) on different data.
3. Each worker sends a “EndComputation” message to the coordinator.
4. Coordinator sends a “ReduceOperation” message to each worker.
5. Each worker computes the reduce operation on the given set of keys, specified in the “ReduceOperation” message.
6. Each worker sends a “EndComputation” message to the coordinator.
7. Coordinator reads the result of each worker from HDFS and merges them into a file.

Network Topology


HDFS
Worker A
Worker B
Worker C
Coordinator
connection0 (NormalOperation)
connection1 (NormalOperation)
connection2 (NormalOperation)
key1.csv
key2.csv
...
key100.csv

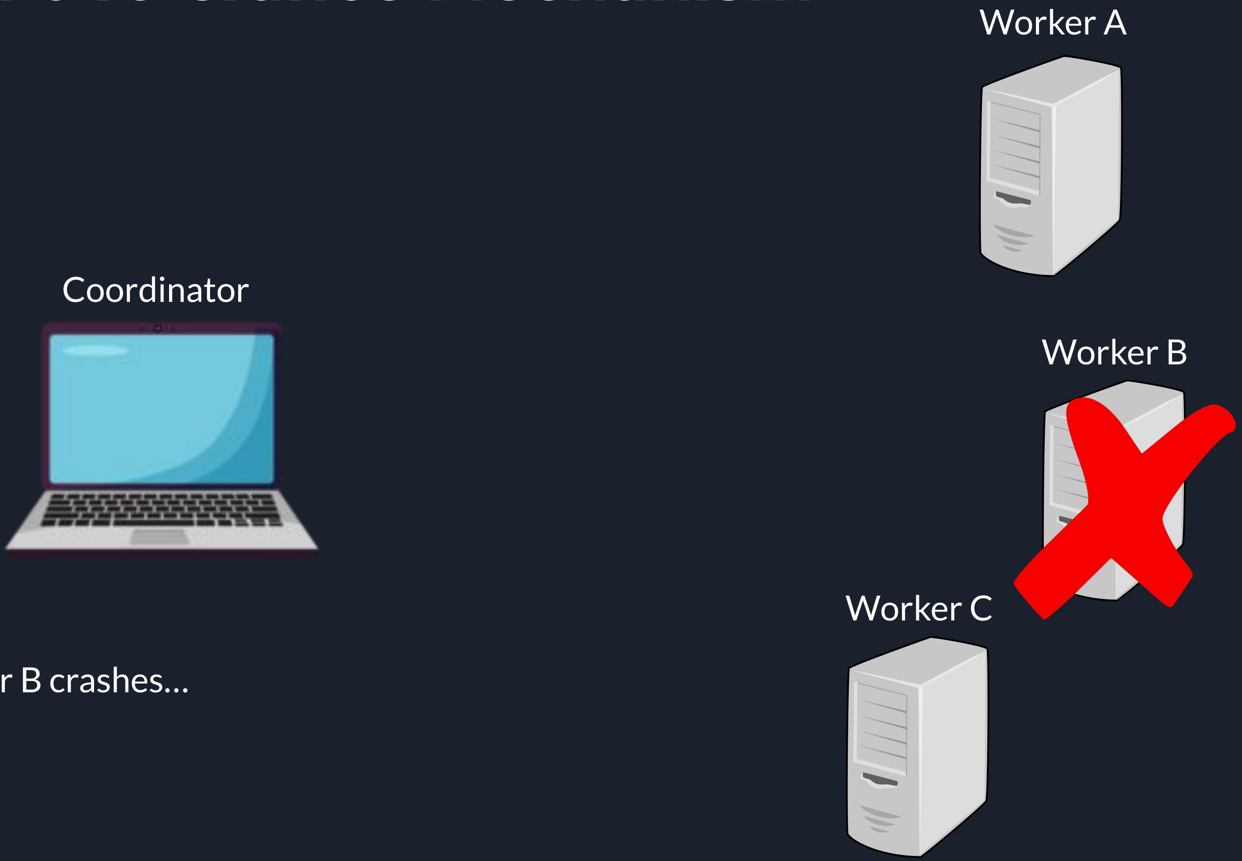
```
configurations.json
"workers": {[
   WorkerA,
   WorkerB,
   WorkerC
]}
```

Fault Tolerance Mechanism



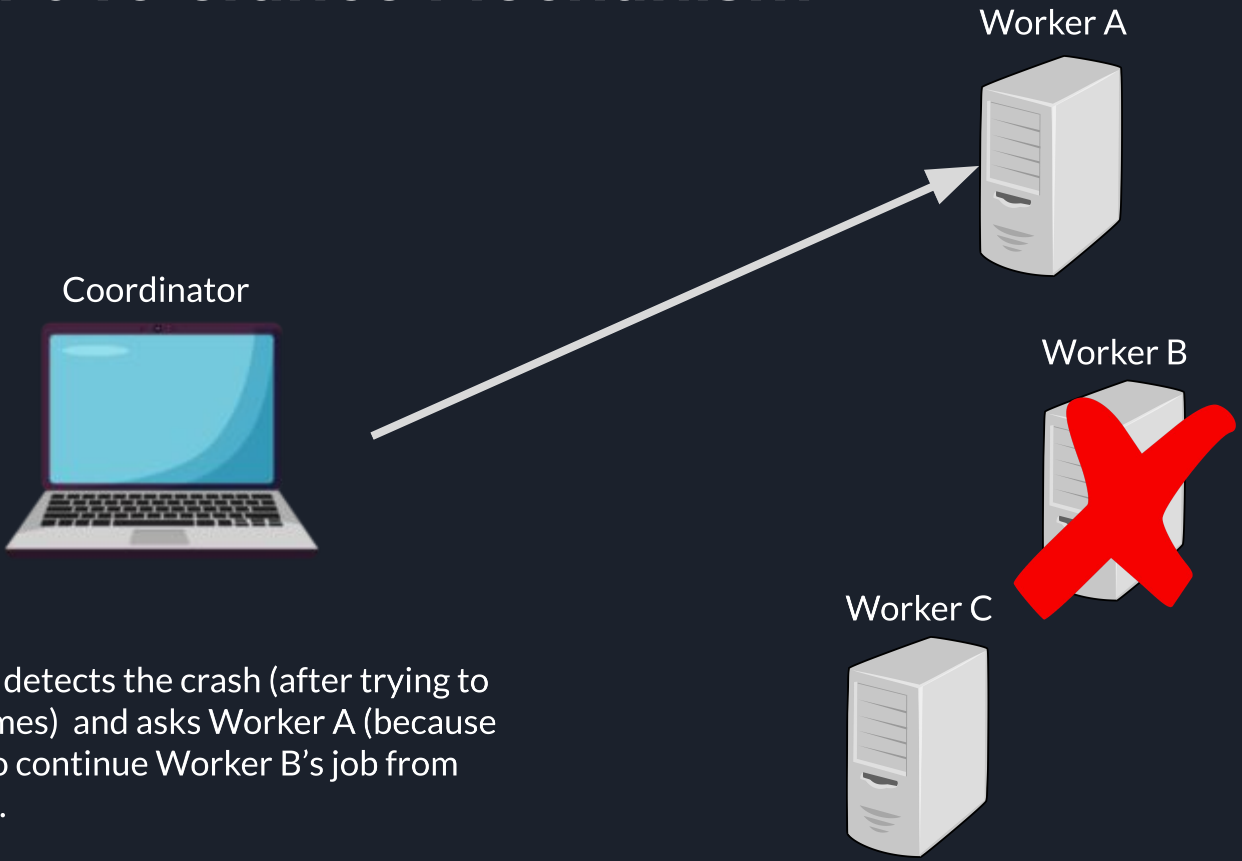
We can consider an example to better explain how fault tolerance is handled during the computation…

Fault Tolerance Mechanism



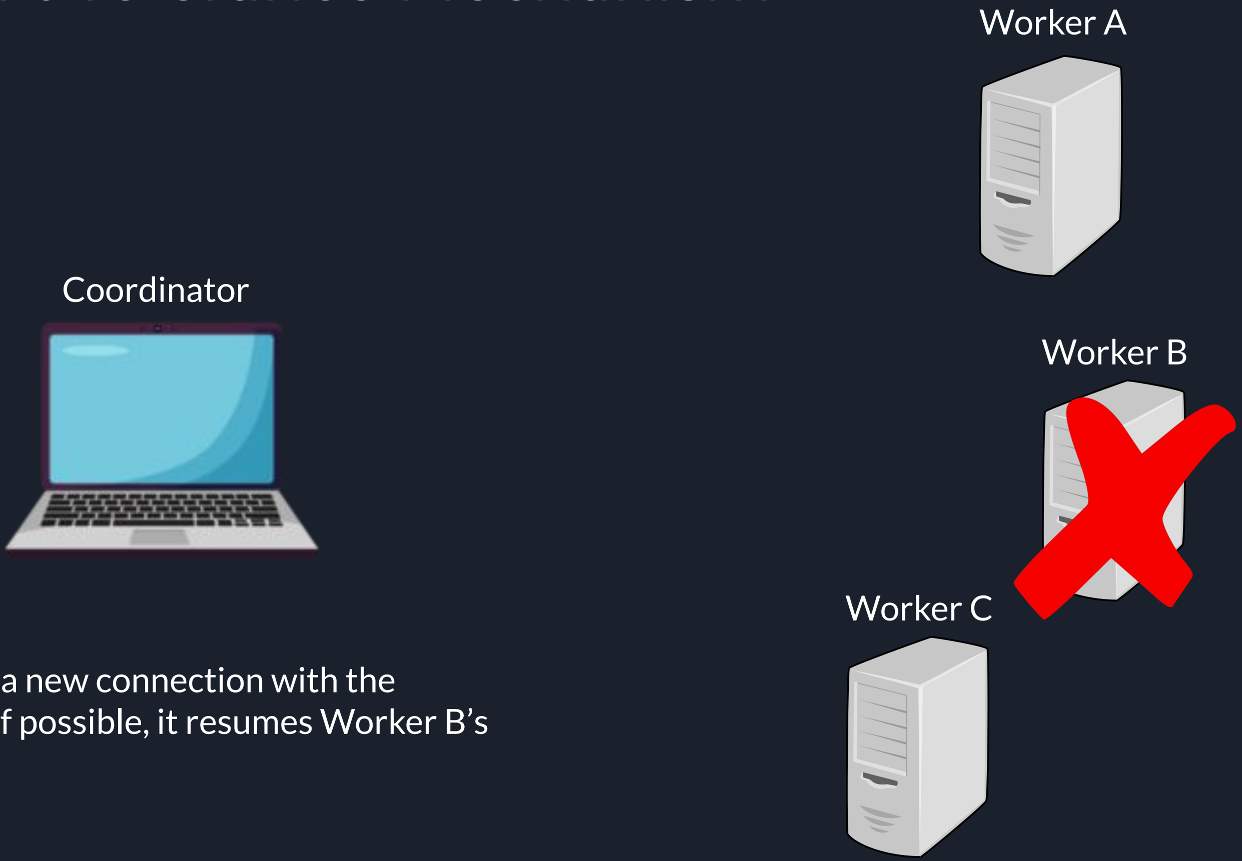
…suddenly Worker B crashes…

Fault Tolerance Mechanism



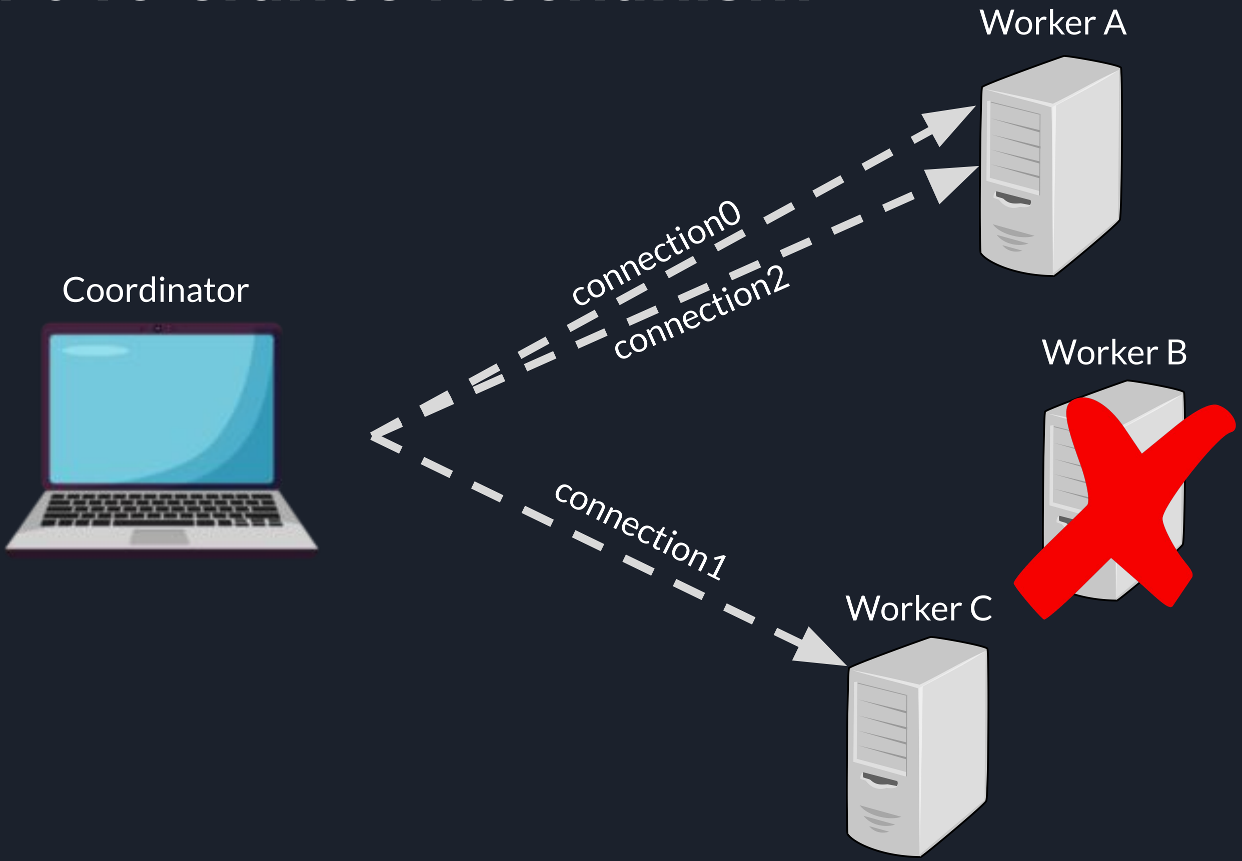
…the Coordinator detects the crash (after trying to reconnect for 3 times) and asks Worker A (because it’s the first one) to continue Worker B’s job from where it stopped…

Fault Tolerance Mechanism



…Worker A starts a new connection with the Coordinator and, if possible, it resumes Worker B's computation.

Fault Tolerance Mechanism



Let’s see the Demo!