Final Presentation

Team Red

How to Run

- Master and worker are distributed as .jar files.

```
java -jar master.jar 2
```

```
java -jar worker.jar 141.223.91.80:5000 -I /data1/input /data2/input -O /home/red/data
```

Results: Small Settings

- Our implementation works well with `small` dataset
  - with 5 machines:
    - Without fault: around 60s
    - With fault: +10~20s
  - with 20 machines:
    - around 20 minutes
    - log shows one or two machines takes unusually long time to sort data (most likely disk I/O issue)

Does the master print a sequence of workers? - ✅

Is the output sorted in each workers? - ✅

# of input records == # of records in the output? - ✅

Results: Large Settings

- However, it crashes with `large` dataset.
  - postmortem analysis will follow after system architecture.

- We couldn’t test `big` dataset due to limited cluster usage time.

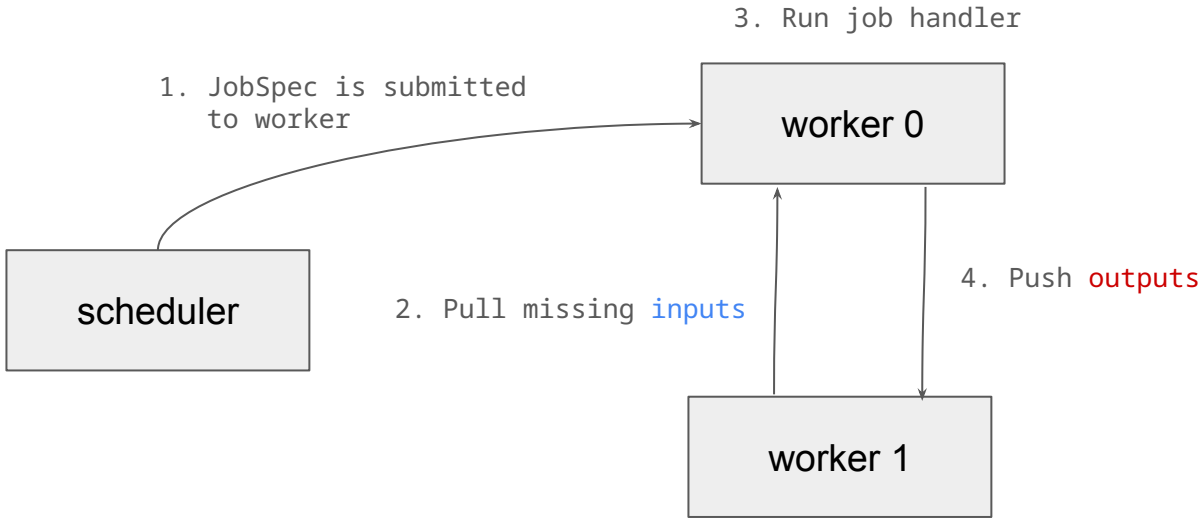
System Architecture

Three subprojects:

- `jobs` - job running infrastructure library, provides:
  - `Scheduler` for dispatching jobs to workers
  - `Worker` for running job when requested by `Scheduler`

- `master` and `worker` - builds on `jobs` library

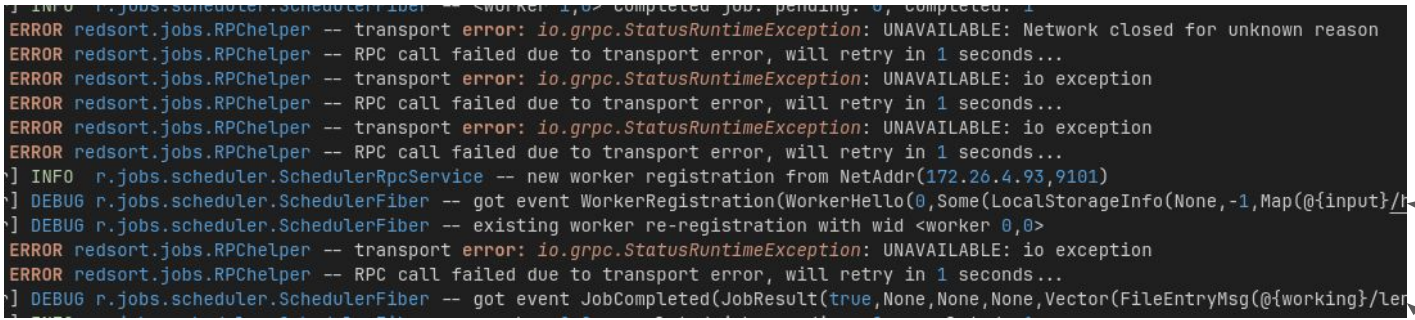
System Architecture: `jobs` package




```
JobSpec(
  name = “sort”,
  args = [],
  inputs = [
    FileEntry(
      path = “@{input}/part1”,
      replicas = [0]
    )
  ],
  outputs = [
    FileEntry(
      path = “@{working}/sorted.0”,
      replicas = [0]
    )
  ]
)
```

System Architecture: Fault Tolerance

- gRPC keepalive for fault detection
  - will raise transport error when keepalive timeout fires
- retry job for fault tolerance - input must be available somewhere!



Why unsuccessful?

We underestimated **network congestion**.

Assumptions on Network

We assume there is a reliable and ordered network channel between any pairs of machines available unless one of machine is down.

Wrong assumption in design phase

```
.keepAliveTime(10, TimeUnit.SECONDS)
.keepAliveTimeout(5, TimeUnit.SECONDS)
```

Misconfiguration of keepalive

network congestion due to replication

insufficient testing on large-scale settings

Why unsuccessful?

What actually happened:

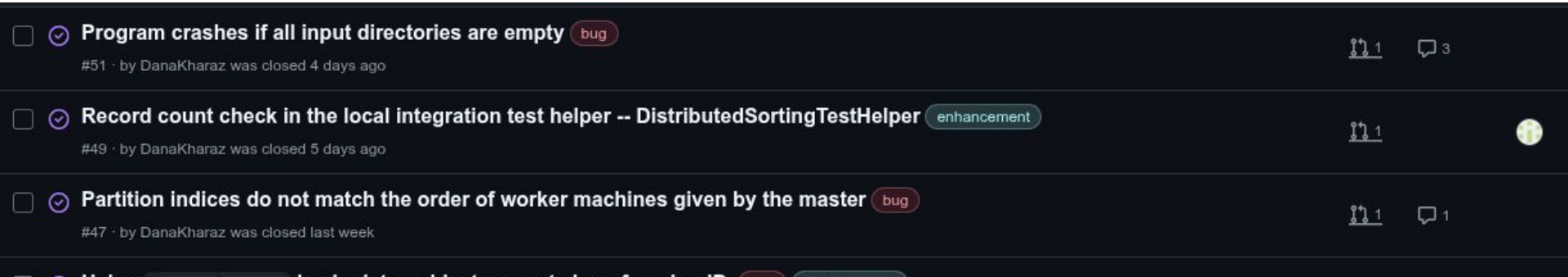
- Our program builds on assumption that network congestion does not happen.
- But with large number of workers, this assumption breaks.

  replicating large files across workers
  → network congestion
  → gRPC keepalive packet drops
  → keepalive timeout fires (false fault detection)
  → job resubmission
  → *exception*

Lessons Learned: Good Parts

1. **Design more, suffer less**
   : Thanks to detailed design, implementation phase sent smoother than we expected.

2. **Collaboration works quite well once all team members get used to the workflow.**



Lessons Learned: Bitter Parts

Always test your program on actual environment.

- The bug we described earlier was not reproducible on local settings.
- Indeed, the bug can be fixed by modifying just a few lines of codes.