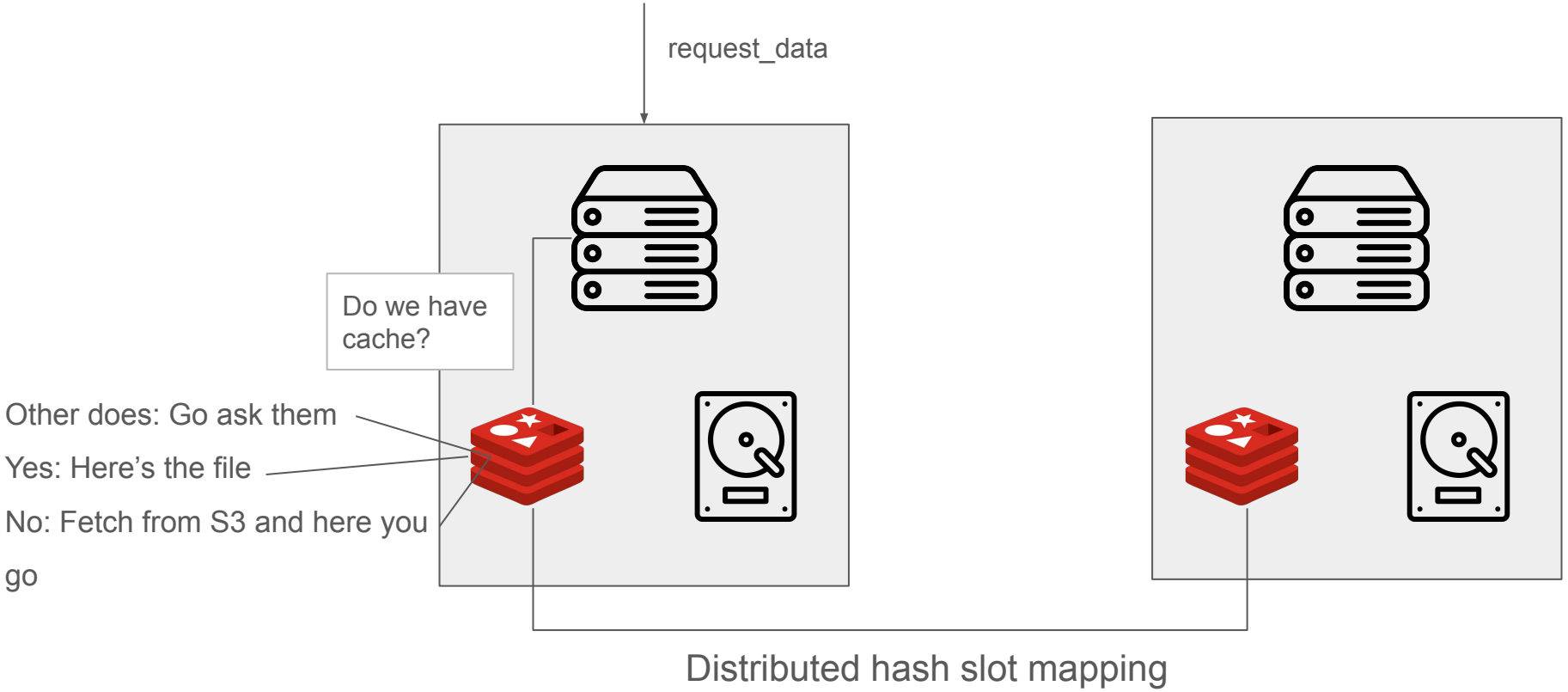
ISTZIIO

Final Update

Core Idea


request_data
Do we have cache?
Other does: Go ask them
Yes: Here's the file
No: Fetch from S3 and here you go
Distributed hash slot mapping

Goals

Phase 1 (75% goal): Implement the single-node version with correct functionalities.

Phase 2 (100% goal): Distributed cache, Benchmark [Finished]

Phase 3 (125% goal) : Predicate pushdown, advanced caching, advanced load-balancing, more APIs, etc

Testing Correctness

- Unit test
- CI/CD setup
- Have ran benchmarks a lot of times

Code Quality Discussion

Strong:

- Core client logic (from request to record batches)
- Trust on redis and tokio
- Simple and correct distributed logic

More work to do:

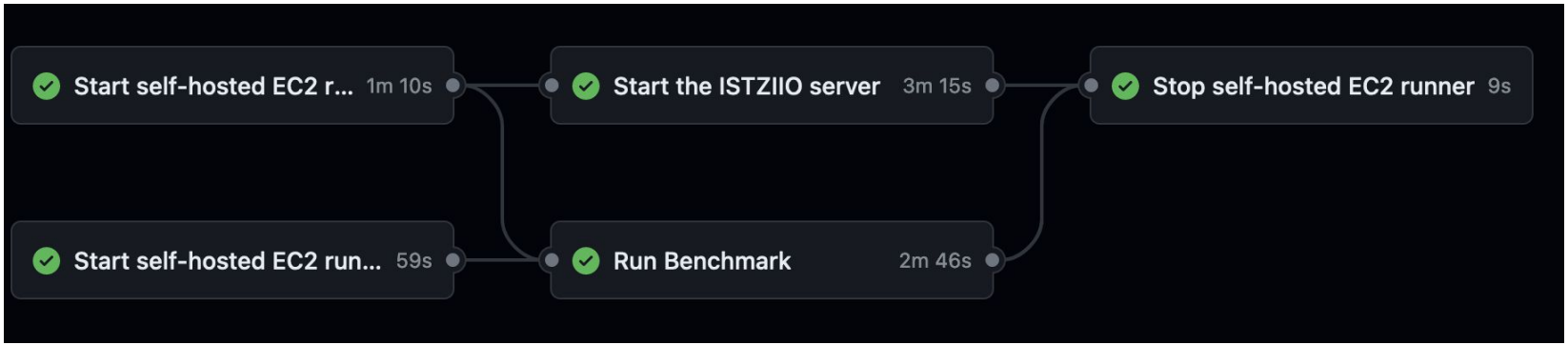
- Local benchmark
- Profiling

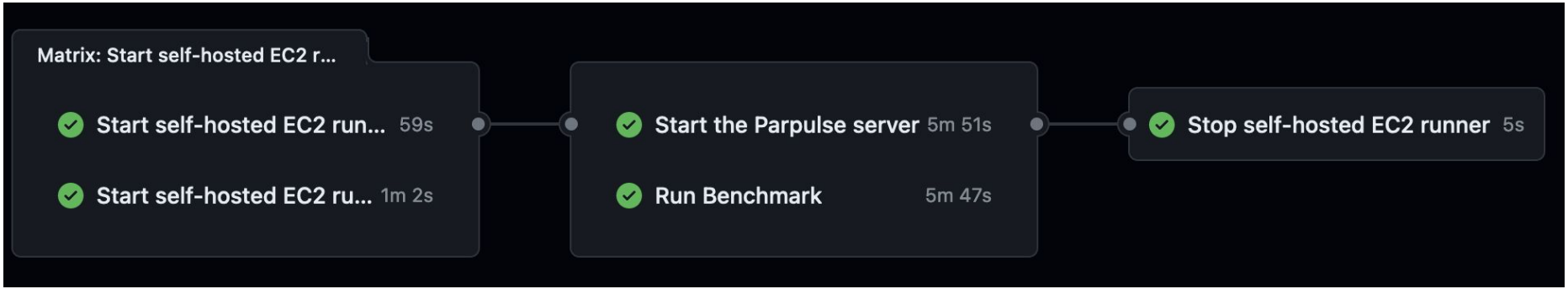
Code Quality Discussion

```
error: failed to publish to registry at https://crates.io

Caused by:
  the remote server responded with an error (status 429 Too Many Requests): You have publi
shed too many versions of this crate in the last 24 hours
```

Benchmark Design - setting



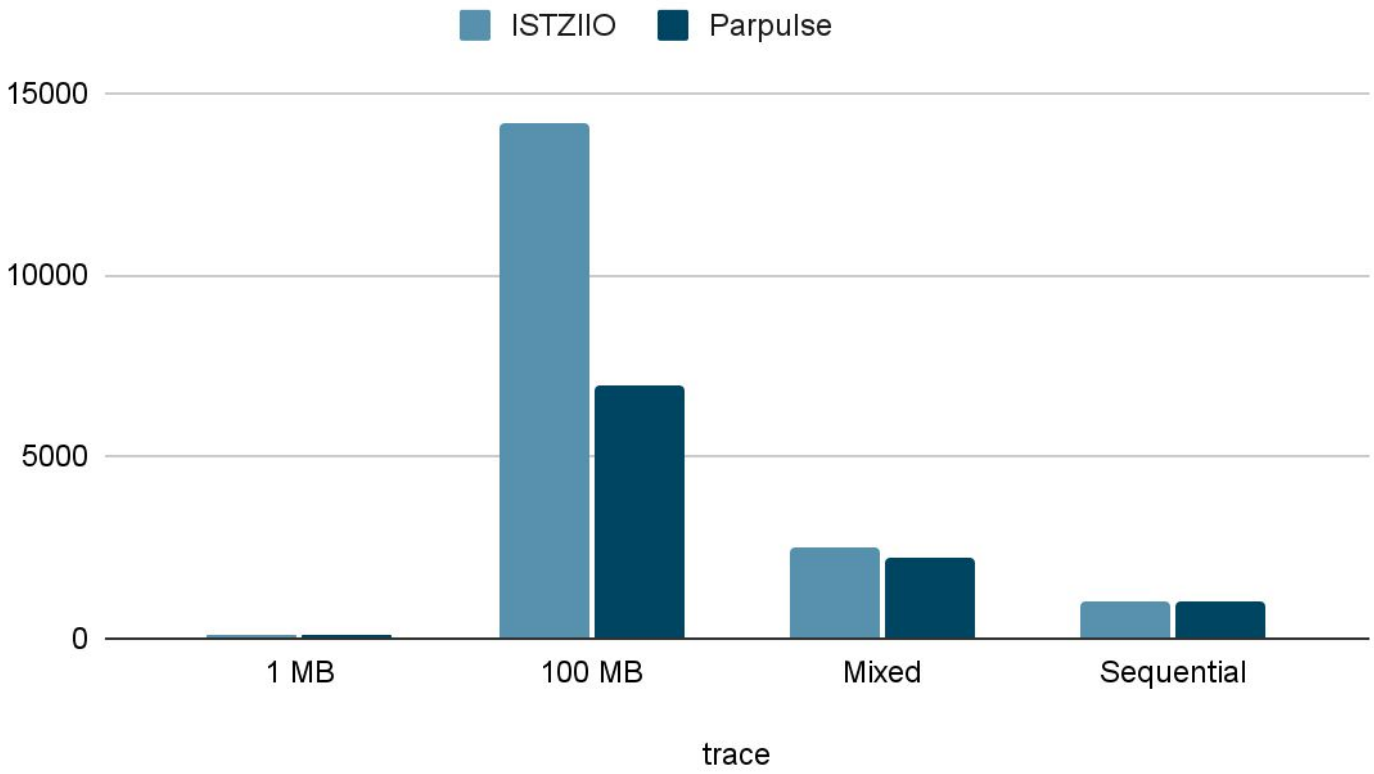


Benchmark Design - setting

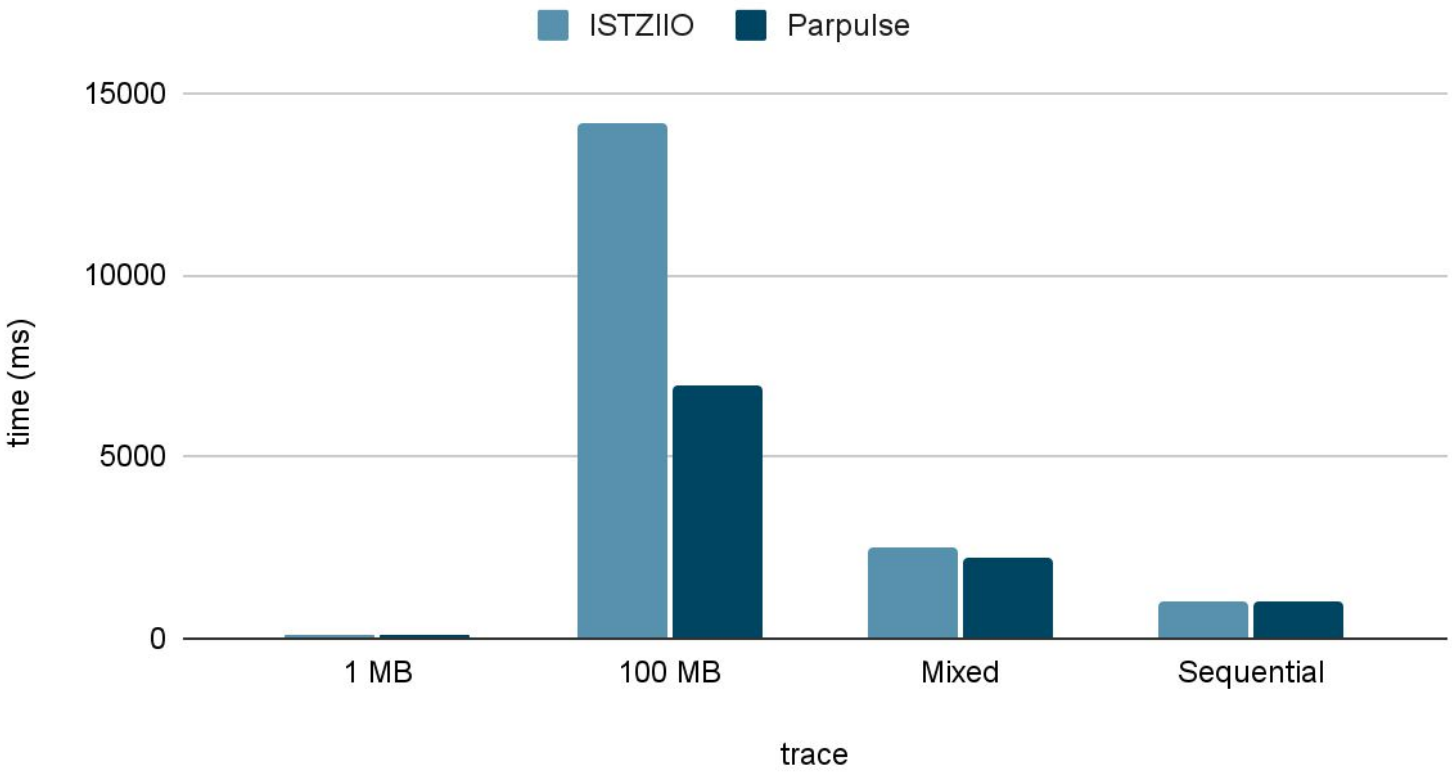
Benchmark environment

- Instance: **C5.xlarge** (both server and client)
- Cache capacity: **1GB**
- Avg. ping: **124 ms**
- Bandwidth: **2.5 Gbps**

Benchmark Design - workload

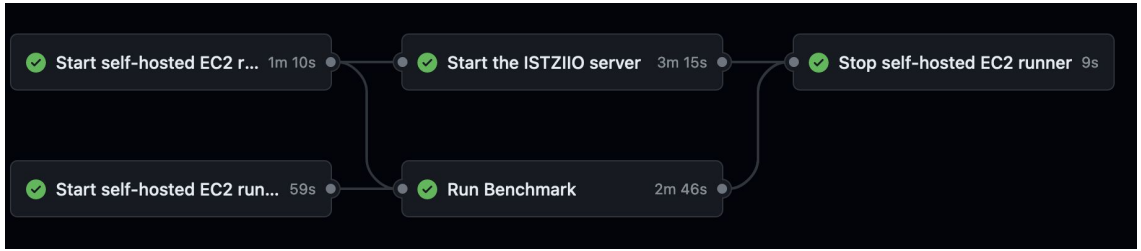

ISTZIIO
Parpulse
15000
10000
5000
0
1 MB
100 MB
Mixed
Sequential
trace

I/O time


ISTZIIO
Parpulse
15000
10000
5000
0
time (ms)
1 MB
100 MB
Mixed
Sequential
trace

Result - single server instance

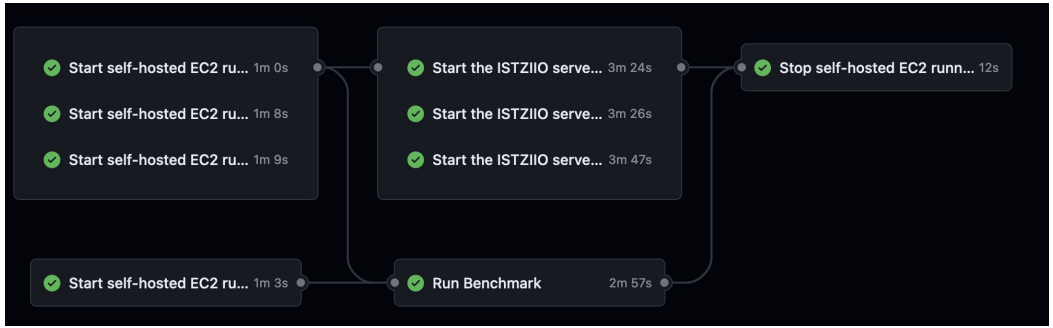
Trace	Avg. Response	File Fetch	Decoding	Server Time*
1MB	116.6 ms +8%	94.9 ms	20.5 ms	30.25ms
100MB	14203.6 ms +104%	10944.1 ms	1919.8 ms	532.475ms
Parallel	2508.5 ms +13%	2301.5 ms	191.1 ms	60.3 ms
Serial	1055.1 ms +2%	556.1 ms	427.5 ms	452ms



*approximated

Result - three server instance

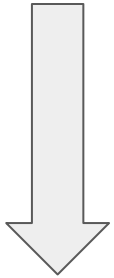
Trace	Avg. Response	File Fetch	Decoding	Server time*
1MB	121.5 ms	94.9 ms	20.2 ms	27.25ms
100MB	9572.5 ms	7962.5 ms	1950 ms	502.475ms
Parallel	2648.6 ms	2056.7 ms	188.1ms	56.3 ms
Serial	1021.7 ms	904.9 ms	667 ms	407ms



*approximated

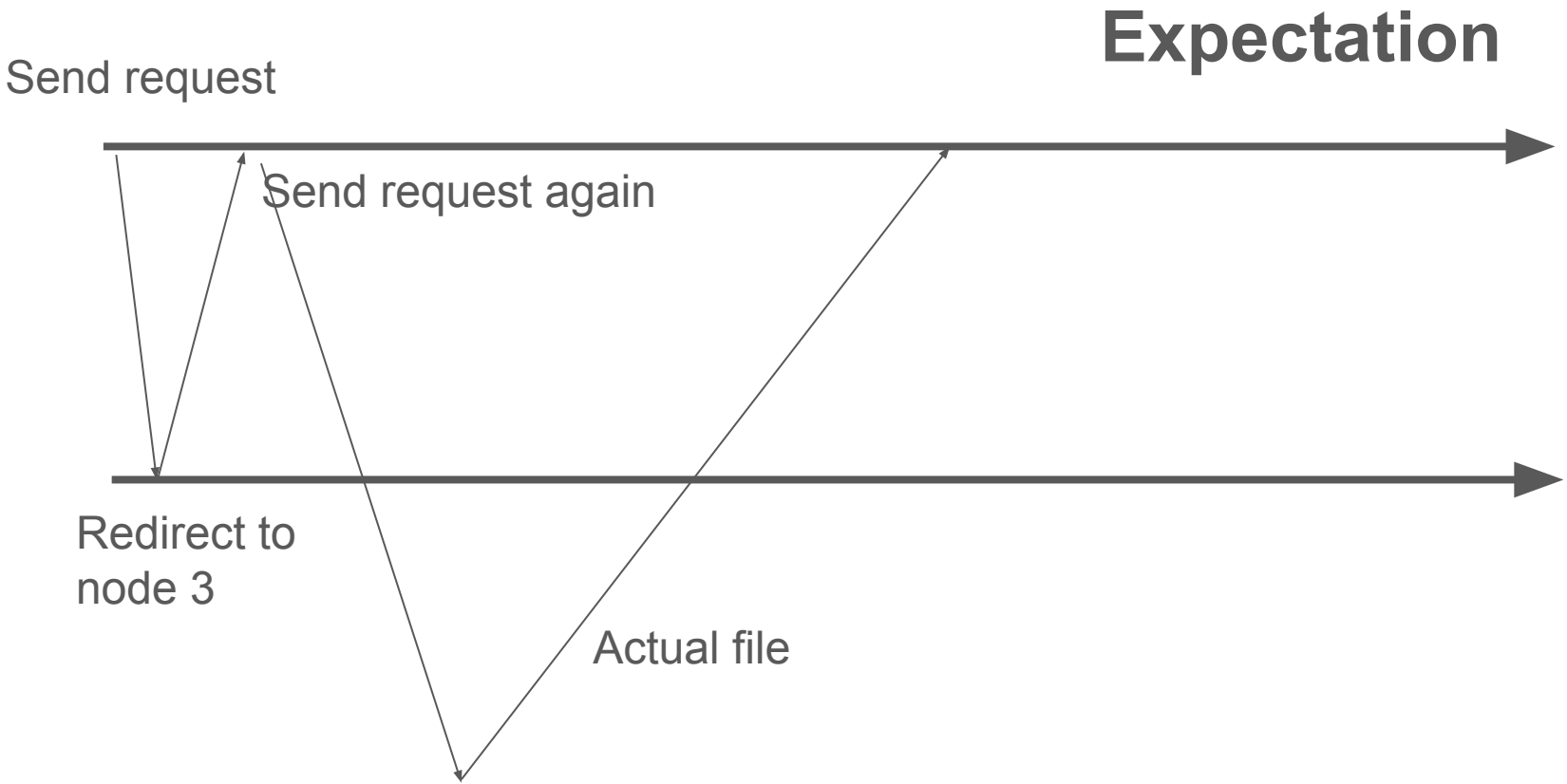
Finding–Client side network bottleneck

~13 s 1 server machine & 1 client machine

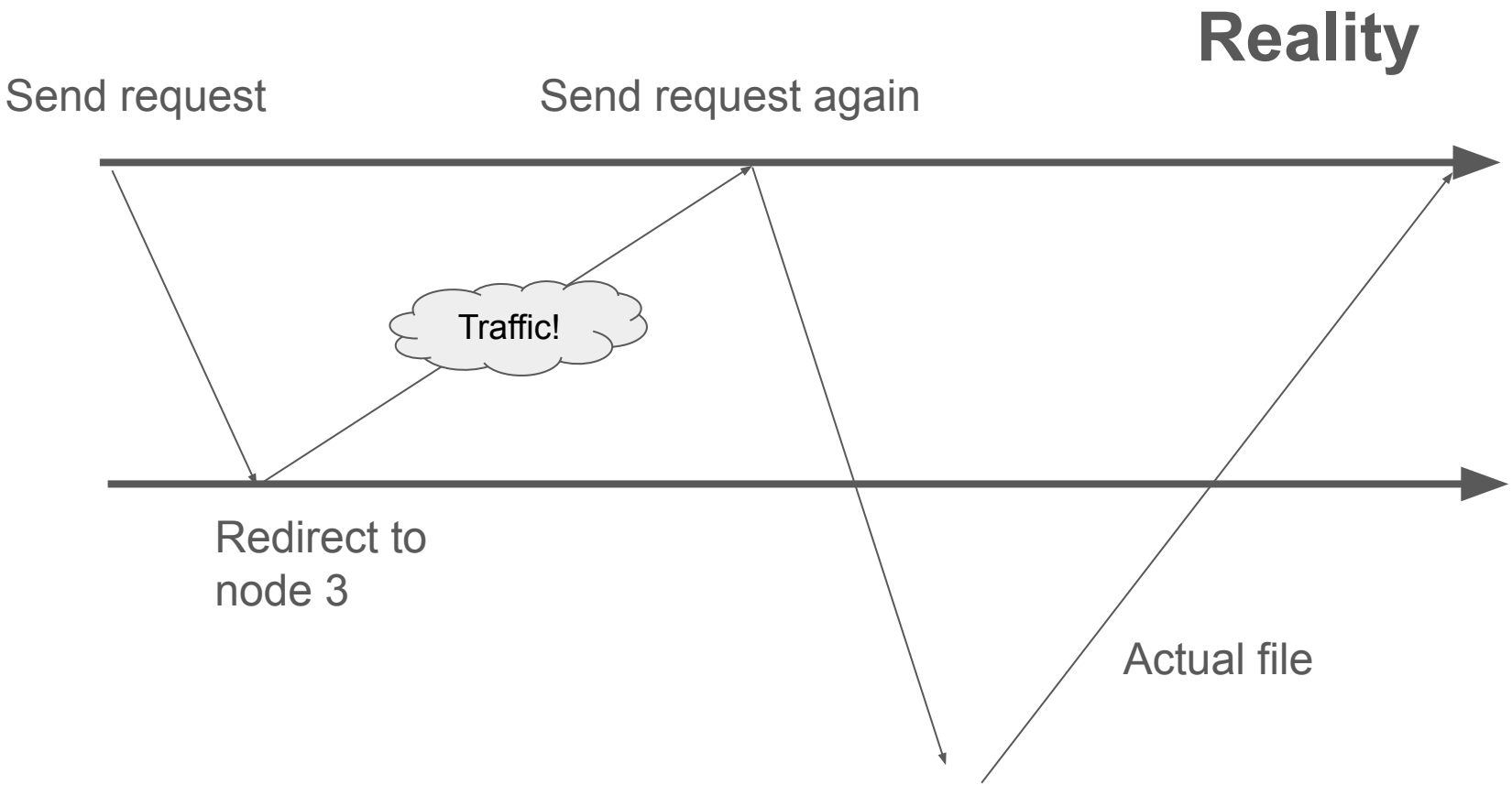


~5 s 1 server machine & 3 client machine

Finding–HTTP Redirection pitfall



Finding–HTTP Redirection pitfall



Future Direction

- More efficient architecture: client side cache for routing(almost complete!)
- Streaming between client and server
- lockfree lru on server side or dynamic tuning hash bucket size or steal from other bucket
- More control over the scheduling of the request threads other than relying on tokio
- Bring back auto scaling