Massively parallel framework (MapReduce/Litmerin)

Speed up some computation by distributing work across servers

Manager (Pool)

Worker (miners)

Map

Filter

Shard 1

Shard 2

Aggregate

Find subset of shard that meets a criteria

Shards are not all the same size

Data imbalance

Failures/leave cluster

Slow workers

Hedging

Restart the task

Quarantine

Progress report

Redistribute proportional
Total order: all servers are able to impose the same ordering on all events.

FIFO: ordering is relative to the server which generates the event.

- Events sent, local for each event increment clock
- Max(msg.clock, local clock) +

$S_1: e_1 e_2 e_5 e_6$
$S_2: e_7 e_5 e_1 e_2$
$S_1: e_1 e_2 e_5 e_6$
$S_2: e_5 e_6 e_1 e_2$
$S_1: e_1 e_2 e_5 e_6$
$S_2: e_7 e_5 e_1 e_2$

$[0]$ $[1]$ $[2]$ $[3]$ $[4]$ $[5]$ $[6]$

$S_1$

$S_2$
Global

Local

How to get to closest? (Clarity)
How to get to a policy compliant clusters?
main objective = even utilization across servers

randomly spread clients to servers

want user to consistently go to same server

Consistency
Global LB

* purely DNS / ANYCAST
  → closest cluster (reduce latency)
  or
  some corporate policy

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>

Local LB

* Consistent hashing
  → even util
  consistently map client to server
Synchronize

Assumptions

- msgs are FIFO
- no msgs are ever dropped
- no server ever fails
- only one snapshot at a time

FIFO

Periodically

Chandy Lamport
@ initiating server

takes a checkpoint of memory

Sends out markers to every server in system

and wait for everyone else to send a marker

a queue for each server

every msg from server goes in queue until you get a marker from the server
Memory
checkpoint
FIFO \leftarrow \text{logical clock}

\rightarrow \text{Vector clock}

Total: FIFO + Tie Breaker