CS1380

Day 2: Jan 26

Theophilus Benson

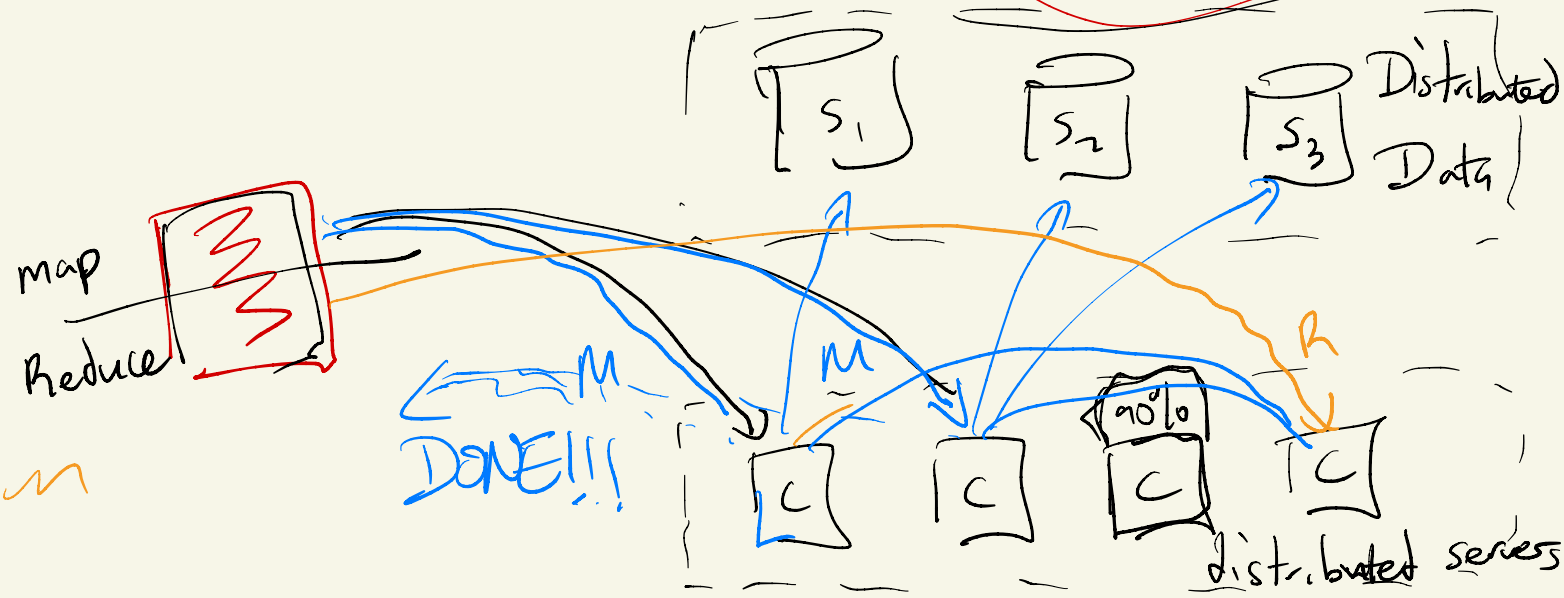
(1) Motivation for D.S. concepts
(why is D.S. Hard)

(2) Networking (How do this enable D.S.)

(3) Failures (types | practicality | overheads)

(4) "fallacies of distributed computing"

Map Reduce = large scale + easily distribute workloads + Massive data



(1) pick servers for Mapping

(2) for each mapper inform it of storage to use

m

Requirements

(a) keep track of all servers
- (i) Health
- (ii) progress
- (iii) data location,
- (iv) load / utilization

(3) pick server(s) for reduce

(4) inform mappers of reduce location

count (*) from all data where name = 'theo' & age ≥ '12'

reduce

map

> 2%

Why is the master node a bad idea?

(1) single point of failure

(2) all load is one server

(1) cheap

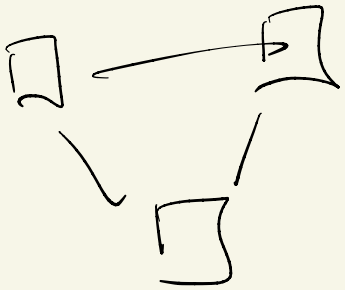
(2) easy to write

(3) easy to coordinate

(4) how much up time is good enough?

Does simplicity still matter with formal verification?

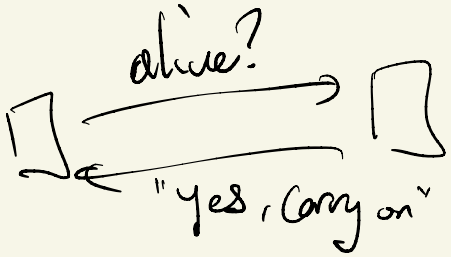
failure / performance problems



(1) failure types
   (a) network
   (b) Power
   (c) resource exhaustion (CPU/mem)
   (d) restart (state failure)
   (e) Hardware failure

easy to detect

(1) Server / power
(2) Some n/w failures



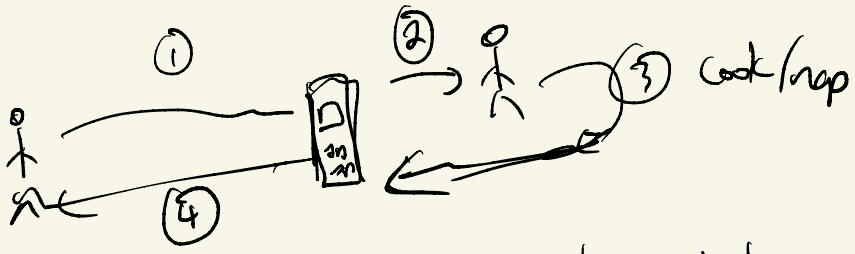
failure detection = heartbeat messages

Questions about designing
(1) frequency
(2) performance impact
(3) msg size
(4) much data to detect failure

(1) How long should you wait for a reply?

wait forever

wait one nanosecond



(1) past experience
(2) 12 hr timezone
(3) day / versus night
(4) indication of receipt

$$\frac{1s}{1} + \frac{1hr}{2} + \frac{3.5hr}{3} + \frac{1s}{4}$$

(2) How often should you send heart beat msgs?

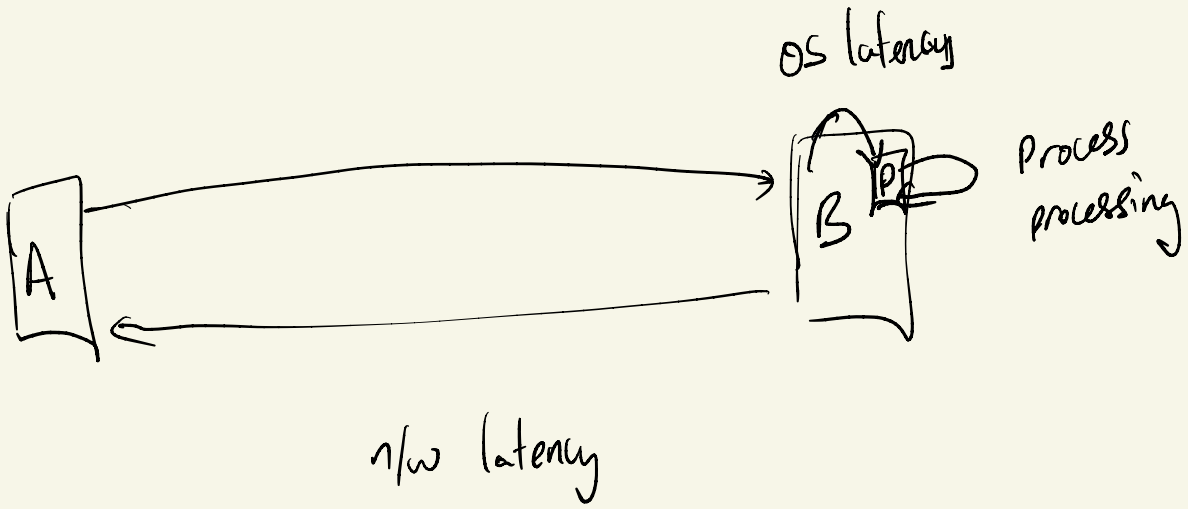
every two years

every second

(1) depends on cpu (speed / generation) !!
(2) criticality of functionality !

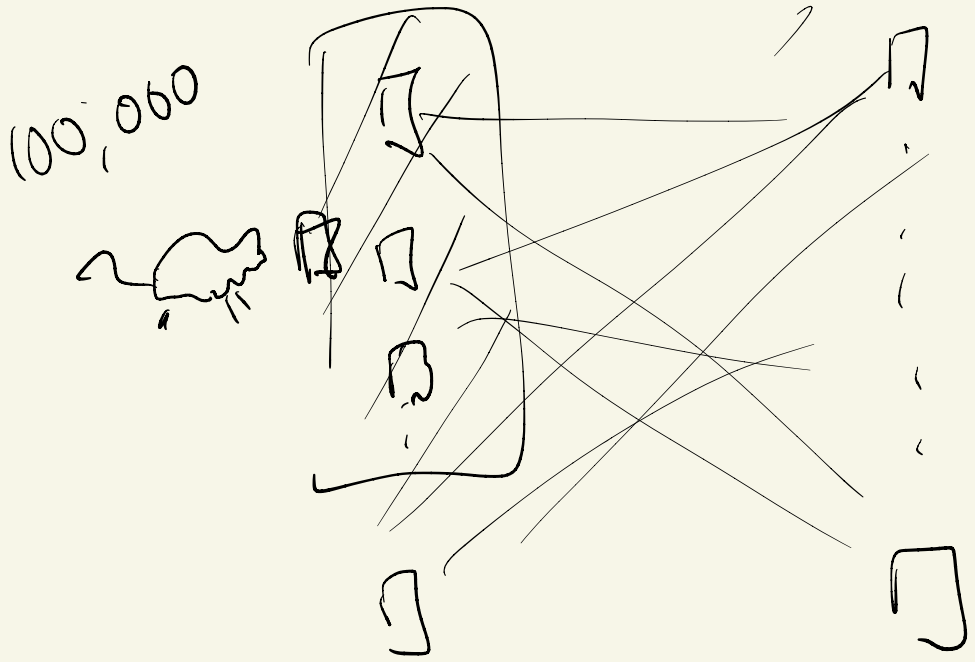
Backup

Google search



Heart beat ➔ n/w + OS + process + n/w

Heart beat time is too small → mistakes (where you think the server is down but it's up)



Theo. Com

(1) 100 k servers

(2) power outage takes out 30 k

(3) after outage all servers are online