Lec 22 Database, Distributed

Way too much data → need distributed
but hard to connect them together

Parallel vs Distributed Configs

▷ Parallel
- Multiple nodes, close to each other (e.g. same rack)
- Fast to access things on another machine

▷ Distributed
- Multiple nodes, far away
- Slow to communicate with another node
- Usually for OLTP

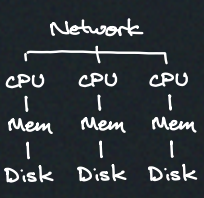
(But when distinction not needed we just say Distributed)

System Architecture

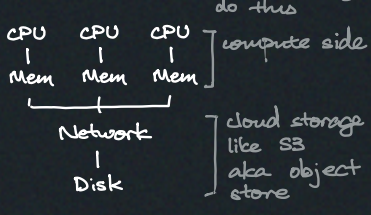
Specifies what shared resources are accessible to the CPU

Shared everything

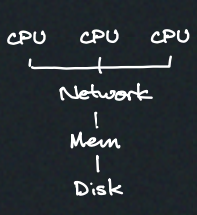
Shared nothing ← many dist. DBs do this



Shared disk ← modern cloud system usually do this



Shared memory ← High performance compute people sometimes do this



Query coordination

▷ Shared nothing

Partition: e.g. by record id ranges
(track ranges in catalog as metadata)

└ For single record access, just use the node with it
└ For aggregate, ask for it from another node

Elastic ... need to shuffle data around :(

▷ Shared Disk

No more shuffling data

Or even serverless: provider hosts stuff and runs queries as a service
provider takes care of scaling

Homogenous vs Heterogenous

Each node does equal thing

Nodes specialise, e.g. optimisation node, ...

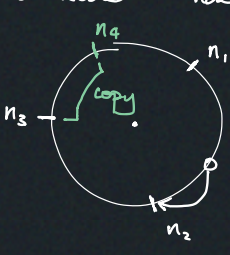
Partitioning

Physical (usually for shared nothing)
▷ Naïve — by table
- Bad for cross-table operations
- Some tables can be bottleneck
- Table can still be huge

▷ Vertical — part. by column
▷ Horizontal — by key + hash
or ranges
or something like that
or round robin
▷ ...

Logical (usually for shared disk)

Elasticity?
→ Consistent Hashing
└ Put hash into circular space
Find node: hash and go clockwise



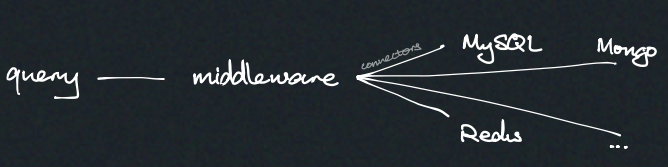
New node: add in node on circle, only grab a small range

Txns

▷ Centralised
└ TP monitor grants partition access
└ Middleware
▷ Decentralised
└ select leader to coordinate per txn
protocol next lecture
▷ Hybrid

Federated DB

Bring together data from different DBs



Misc

- Distributed 2PL
- Use CRDT