Lec 14 Query Execution II

→ Using multiple workers for query execution
  └ throughput ↑
  └ latency ↓
  └ hopefully lower cost of ownership ↓
→ Distributed
  └ Multiple copy for redundancy
  └ Closer to users if needed

Distributed	vs	Parallel
can be scattered far away		physically close
slower, more expensive connection		cheap, fast connection

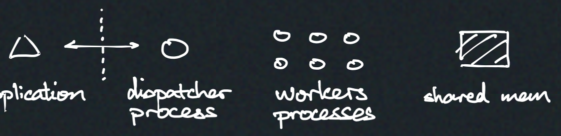
both should behave just like single thread

Process Models

How the system archi supports concurrent queries?
How workers work?

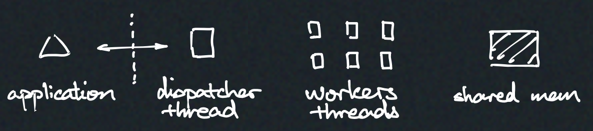
* Process model

▹ Process   IBM DB2, Oracle, Postgres



- High overhead
- OS bad at scheduling
+ Process crash doesn't take down whole system

▹ Thread   most DBMS made within last 20 years

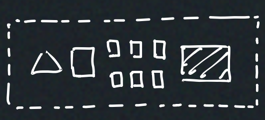


+ DBMS more control over scheduling
  └ e.g. tell OS that some threads should be on same socket
+ Lightweight
- One thread crash ⇒ whole system crash

→ SQLOS — Run SQL server but take more hardware control
  └ Queries yield every 4ms to allow scheduler to interrupt

▹ Embedded

Application embeds DB — just link the library



Execution Parallelism

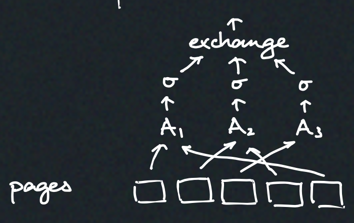
* Inter-Query

later

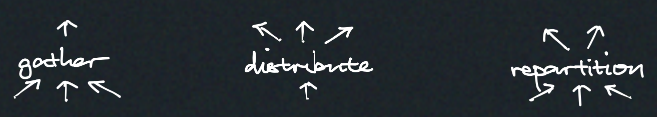
* Intra-Query

→ Parallel hash join — do all the bucket joins in parallel

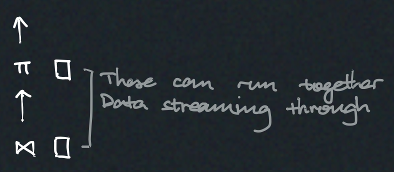
▹ Intra-operator (horizontal)



* Exchange op types



▹ Inter-operator (vertical)



▹ Bushy parallelism — do both vertical & horizontal

Deciding optimal plan is NP-hard!

IO Parallelism

Multiple disks behaves like one

→ One relation per disk?
→ Handle disk failure?

RAID — hardware controller

→ Round robin pages
→ Every disk same copy
→ In between?

Modern: erasure code

Scheduler

Quickstep

- Queue of pending work order,
- Walk dependency tree to add order
- Worker threads grab and do work