Introduction Parallel Databases

Objectives:

1) Sources of Parallelism
2) Hardware/Software Architectures
3) Effectiveness

Slide thanks: many sources, ;including Suciu

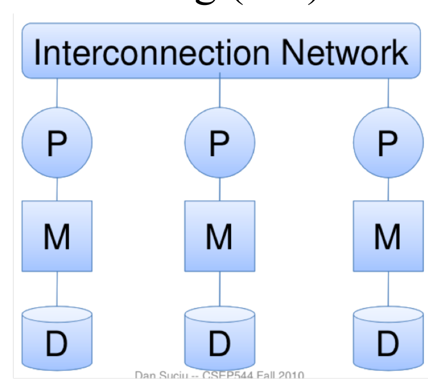
DBs Provide Many Opportunities for Parallelism

- I/O Parallelism // RAID or simply many disks
- Interquery Parallelism // Transactions
- Intraquery Parallelism // operations in a plan
  - pipelined,
  - simply independent, Interoperation Parallelism
- Intraoperation Parallelism

[Silberschatz et. al.]

Hardware Configurations

Shared Nothing (DB) Clusters



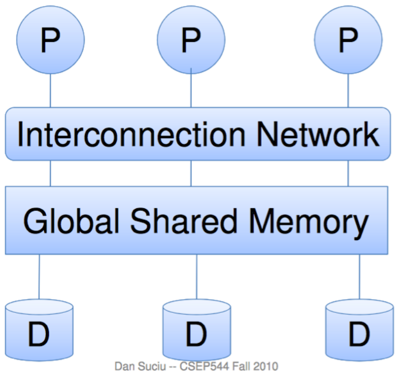
Shared Nothing – Advantages

- Most scalable
  - Minimizes interference by minimizing resource sharing
  - Memory and I/O bandwidth and capacity grow with the number of compute nodes.
- Can use commodity hardware

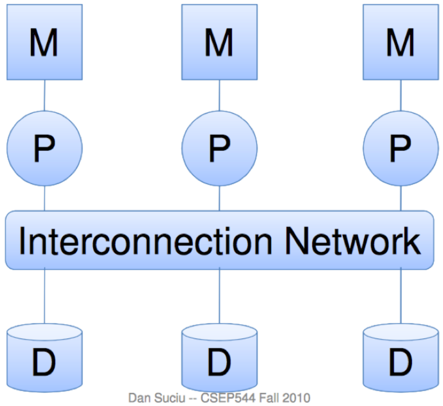
Other Hardware Organizations

- Shared memory
- Shared disk
- Logic per disk [head] and other weird computers.

Shared Memory



Shared Disk



Weird Machines

- Logic, i.e. a CPU per disk [head]
  - push select and project into disk controller

    // reduces I/O bandwidth
- "**Database machines: an idea whose time has passed? A critique of the future of database machines**"
  - Haran Boral and David Dewitt, 1989
  - Core point: must use commodity hardware
- "Query Processing on Smart SSDs"
  - Park et.al. and Dewitt
  - http://sites.computer.org/debull/A14june/p19.pdf

Parallel DBMSs

- Goal
  - Improve performance by executing multiple operations in parallel
- Key benefit
  - Cheaper to scale than relying on a single increasingly more powerful processor
- Key challenge
  - Ensure overhead and contention do not kill performance

Dan Suciu – CSEP544 Fall 2011

Performance Metrics for Parallel DBMSs

- Speedup
  - More processors ➔ higher speed
  - Individual queries should run faster
  - Should do more transactions per second (TPS)
- Scaleup // sequential execution time
  - More processors ➔ can process more data
  - Batch scaleup
    - Same query on larger input data should take the same time
  - Transaction scaleup
    - N-times as many TPS on N-times larger database
    - But each transaction typically remains small

Dan Suciu – CSEP544 Fall 2011

Speed-up and Amdahl's law

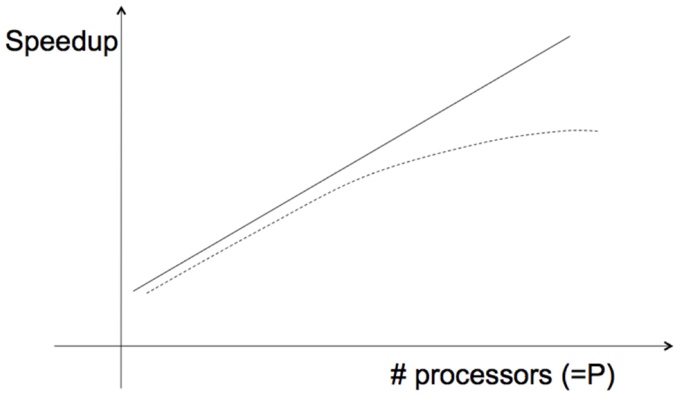
- Speed up, S

$$S = \frac{Tseq}{Tpar}$$

// sequential execution time

// parallel execution time

Linear v.s. Non-linear Speedup



Challenges to Linear Speedup and Scaleup

- Startup cost
  - Cost of starting an operation on many processors
- Interference
  - Contention for resources between processors
- Skew
  - Slowest processor becomes the bottleneck

Speed-up and Amdahl's law

- Speed up, S

$$S = \frac{Tseq}{Tpar}$$

- Suppose,

$Tpar = (1 - \alpha)\ Tseq + \alpha\ Tseq/P$ // α proportion of sequential execution // time that can be parallelized

- Then

$S = 1 / ((1 - \alpha) + \alpha/P)$

- Suppose $\alpha = 0.9$, number of processors $P \rightarrow \infty$, then $S \rightarrow 10$

How does this relate to HW5a?

How does this relate to HW5a?

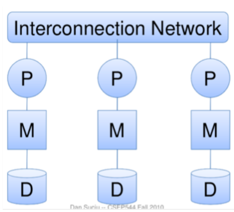
- Unless processing starts with
  data distributed/enabled for parallel
access witnessing parallel speed-up is hard.

How is this so ignored/unknown?

Current Business as Usual

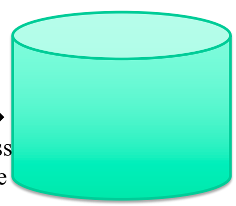
Cloud storage charges ($):

Cluster



$ here →
much less
← $ here

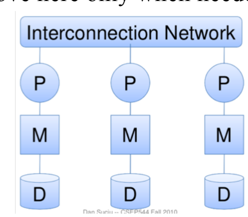
Mass Storage



Result: Common Practice

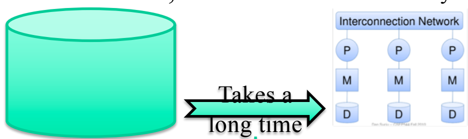
Store Here,

move here only when needed



Further:

Store Here, move here only when needed



Data mining, Machine learning, ad-hoc*

- Do a particular analysis
- Train a model

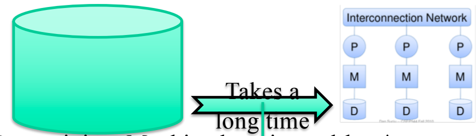
* do it once when need arises

Move data night before,
Spend one day, (and associated charges),
iterating on your DM/ML task.

100% certain

- how to work on machines at TACC.
- Automatic consequence of $.

Store Here, move here only when needed



Data mining, Machine learning, ad-hoc*

- Do a particular analysis
- Train a model

* do it once when need arises

Move data night before,
Spend one day, (and associated charges),
iterating on your DM/ML task.