Client-Server Systems

- Database functionality can be divided into:
  - **Back-end**: manages access structures, query evaluation and optimization, concurrency control and recovery.
  - **Front-end**: consists of tools such as forms, report-writers, and graphical user interface facilities.
- The interface between the front-end and the back-end is through SQL or through an application program interface.
Parallel Databases

- **Why?**
  - More transactions per second, or less time per query
  - Throughput vs. Response Time
  - Speedup vs. Scaleup

- **Database operations are embarrassingly parallel**
  - E.g. Consider a join between R and S on R.b = S.b

- **But, perfect speedup doesn’t happen**
  - Start-up costs
  - Interference
  - Skew

Parallel Systems

- Parallel database systems consist of multiple processors and multiple disks connected by a fast interconnection network.

- A **coarse-grain parallel** machine consists of a small number of powerful processors

- A **massively parallel** or **fine grain parallel** machine utilizes thousands of smaller processors.

- Two main performance measures:
  - **throughput** --- the number of tasks that can be completed in a given time interval
  - **response time** --- the amount of time it takes to complete a single task from the time it is submitted
Speed-Up and Scale-Up

- **Speedup**: a fixed-sized problem executing on a small system is given to a system which is $N$-times larger.
  - Measured by:
    \[
    \text{speedup} = \frac{\text{small system elapsed time}}{\text{large system elapsed time}}
    \]
  - Speedup is **linear** if equation equals $N$.

- **Scaleup**: increase the size of both the problem and the system
  - $N$-times larger system used to perform $N$-times larger job
  - Measured by:
    \[
    \text{scaleup} = \frac{\text{small system small problem elapsed time}}{\text{big system big problem elapsed time}}
    \]
  - Scale up is **linear** if equation equals 1.

---

**Speedup**

- linear speedup
- sublinear speedup

---

**Diagram**

- Speed vs. Resources
- Linear speedup
- Sublinear speedup
Factors Limiting Speedup and Scaleup

Speedup and scaleup are often sublinear due to:

- **Startup costs:**
  - Cost of starting up multiple processes may dominate computation time, if the degree of parallelism is high.

- **Interference:**
  - Processes accessing shared resources (e.g., system bus, disks, or locks) compete with each other, thus spending time waiting on other processes, rather than performing useful work.

- **Skew:**
  - Increasing the degree of parallelism increases the variance in service times of executing tasks in parallel.
  - Overall execution time determined by **slowest** of parallelly executing tasks.
Parallel Databases

- Shared-nothing vs. shared-memory vs. shared-disk

<table>
<thead>
<tr>
<th></th>
<th>Shared Memory</th>
<th>Shared Disk</th>
<th>Shared Nothing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication between processors</td>
<td>Extremely fast</td>
<td>Disk interconnect is very fast</td>
<td>Over a LAN, so slowest</td>
</tr>
<tr>
<td>Scalability ?</td>
<td>Not beyond 32 or 64 or so (memory bus is the bottleneck)</td>
<td>Not very scalable (disk interconnect is the bottleneck)</td>
<td>Very very scalable</td>
</tr>
<tr>
<td>Notes</td>
<td>Cache-coherency an issue</td>
<td>Transactions complicated; natural fault-tolerance.</td>
<td>Distributed transactions are complicated (deadlock detection etc);</td>
</tr>
<tr>
<td>Main use</td>
<td>Low degrees of parallelism</td>
<td>Not used very often</td>
<td>Everywhere</td>
</tr>
</tbody>
</table>
Distributed Systems

- Over a wide area network
- Typically not motivated by performance reasons
  - For that, use a parallel system
- Done because of necessity
  - Imagine a large corporation with offices all over the world
  - Also, for redundancy and for disaster recovery reasons (geo-replication)
- Lot of headaches
  - Especially if trying to execute transactions that involve data from multiple sites
    - Keeping the databases in sync
      - 2-phase commit for transactions uniformly hated
    - Autonomy issues
      - Even within an organization, people tend to be protective of their unit/department
  - Locks/Deadlock management
  - Works better for query processing
    - Since we are only reading the data

Distributed Systems: Issues

- Consistency
  - Two-phase commit across geo-replicated stores

- Scaling

- Replication

- Linearizability / externalizability / “strict consistency”

Discussing Calvin and Spanner
Exam Questions

- serializability:
  - conflict, view,
  - Thomas’s write rule, phantom problem
- locking:
  - 2-phase, strict, rigorous
  - issues: dirty reads, recoverability, cascading aborts
  - lock granularity, acquisition order
- other CC:
  - timestamp consistency, optimistic concurrency control, snapshot isolation
- rollback recovery:
  - force/no_force, steal/no_steal
  - log record / db write ordering
- calvin / spanner ............... won’t be on it

Grades

- Grade statistics (through 5/5):
  - average: 83%
  - stdev: 9.3%
- Grade cutoffs:
  - A: 90%
  - B: 80%
  - C: 70%
  - unless Exam3 results are very much different that prior results