Announcement

• WQ7 due tonight
  – (was due yesterday)

• HW7 due on Wednesday

• HW8 (last!) on Spark
  – will be posted later this week

Why compute in parallel?

• Multi-cores:
  – Most processors have multiple cores
  – This trend will increase in the future

• Big data: too large to fit in main memory
  – Distributed query processing on 100-1000 servers
  – Widely available now using cloud services

Big Data

• Companies, organizations, scientists have data that is too big (and sometimes too complex) to be managed without changing tools and processes

  • Complex data processing:
    – Decision support queries (SQL w/ aggregates)
    – Machine learning (adds linear algebra and iteration)

Two Kinds to Parallel Data Processing

• Parallel databases, developed starting with the 80s (this lecture)
  – OLTP (Online Transaction Processing)
  – OLAP (Online Analytic Processing, or Decision Support)

• General purpose distributed processing: MapReduce, Spark
  – Mostly for Decision Support Queries

Performance Metrics for Parallel DBMSs

P = the number of nodes (processors, computers)

• Speedup:
  – More nodes, same data \( \Rightarrow \) higher speed

• Scaleup:
  – More nodes, more data \( \Rightarrow \) same speed

• OLTP: “Speed” = transactions per second (TPS)
• Decision Support: “Speed” = query time
Linear v.s. Non-linear Speedup

Linear v.s. Non-linear Scaleup

Challenges to Linear Speedup and Scaleup
- **Startup cost**
  - Cost of starting an operation on many nodes

- **Interference**
  - Contention for resources between nodes

- **Stragglers**
  - Slowest node becomes the bottleneck

Architectures for Parallel Databases
- **Shared memory**
- **Shared disk**
- **Shared nothing**

Shared Memory

Shared Disk
Shared Nothing

- Cluster of machines on high-speed network
- Each machine has its own memory and disk: lowest contention

NOTE: Because all machines today have many cores and many disks, then shared-nothing systems typically run many "nodes" on a single physical machine.

Characteristics:
- Today, this is the most scalable architecture.
- Most difficult to administer and tune.

We discuss only Shared Nothing in class

Shared Disk

- All nodes access the same disks
- Found in the largest "single-box" (non-cluster) multiprocessors

Oracle dominates this class of systems.

Characteristics:
- Also hard to scale past a certain point: existing deployments typically have fewer than 10 machines

Approaches to Parallel Query Evaluation

- Inter-query parallelism
  - Transaction per node
  - OLTP

- Inter-operator parallelism
  - Operator per node
  - Both OLTP and Decision Support

- Intra-operator parallelism
  - Operator on multiple nodes
  - Decision Support

We study only intra-operator parallelism: most scalable
Single Node Query Processing (Review)

Given relations R(A,B) and S(B, C), no indexes:

- **Selection**: \( \sigma_{A=123}(R) \)
  - Scan file R, select records with A=123

- **Group-by**: \( \gamma_{A,sum}(R) \)
  - Scan file R, insert into a hash table using attr A as key
  - When a new key is equal to an existing one, add B to the value

- **Join**: \( R \bowtie S \)
  - Scan file S, insert into a hash table using attr B as key
  - Scan file R, probe the hash table using attr B

Distributed Query Processing

- Data is horizontally **partitioned** across many servers

- Operators may require data reshuffling
  - not all the needed data is in one place

Horizontal Data Partitioning

<table>
<thead>
<tr>
<th>Data:</th>
<th>Servers:</th>
</tr>
</thead>
<tbody>
<tr>
<td>K A B</td>
<td>1 2 ... P</td>
</tr>
</tbody>
</table>

- **Block Partition**:
  - Partition tuples arbitrarily s.t. \( \text{size}(R_1) = \ldots = \text{size}(R_P) \)

- **Hash partitioned on attribute A**:  
  - Tuple t goes to chunk i, where \( i = h(t.A) \mod P + 1 \)

- **Range partitioned on attribute A**:  
  - Partition the range of A into \( v_0 < v_1 < \ldots < v_P = \infty \)
  - Tuple t goes to chunk i, if \( v_{i-1} < t.A < v_i \)

Parallel GroupBy

**Data**: R(K,A,B,C)  
**Query**: \( \gamma_{A,sum}(C)(R) \)

How can we compute in each case?

- R is hash-partitioned on A
- R is block-partitioned
- R is hash-partitioned on K
Parallel GroupBy

Data: R(K,A,B,C)
Query: \( \gamma_{A, \text{sum}(C)}(R) \)
• R is block-partitioned or hash-partitioned on K

Parallel Join

Data: R(K1,A,B), S(K2,B,C)
Query: R(K1,A,B) \( \bowtie \) S(K2,B,C)

Speedup and Scaleup

• Consider:
  – Query: \( \gamma_{A, \text{sum}(C)}(R) \)
  – Runtime: dominated by reading chunks from disk
  • If we double the number of nodes P, what is the new running time?
    – Half (each server holds \( \frac{1}{2} \) as many chunks)
  • If we double both P and the size of R, what is the new running time?
    – Same (each server holds the same # of chunks)

Uniform Data v.s. Skewed Data

• Let R(K,A,B,C); which of the following partition methods may result in skewed partitions?
  • Block partition
  • Hash-partition
    – On the key K
    – On the attribute A

Loading Data into a Parallel DBMS

Example using Teradata System

AMP = “Access Module Processor” = unit of parallelism
Example Parallel Query Execution

Find all orders from today, along with the items ordered:

```sql
SELECT * FROM Order o, Product p
WHERE o.pid = p.pid AND o.date = today()
```

Contains all orders and all lines where hash(pid) = 1

Contains all orders and all lines where hash(pid) = 2

Contains all orders and all lines where hash(pid) = 3