Course Development in Parallel and Distributed Database Systems

Jamal R. Alsabagh and Carolyn Winston

Grambling State University
Department of Math. and Computer Science
Carver Hall, Room 137
Grambling, Louisiana 71245

U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

This is work in progress. It is part of a project titled "Integrating Research Results from Parallel and Distributed Processing into the Computer Science Curriculum" being carried out in collaboration with Louisiana Tech University and is partially supported by NSF.

Approved for public release; distribution unlimited.

Parallel Database Management Systems (PDBMS) address the performance requirements of heavily loaded systems that handle very large databases. Early attempts at meeting the requirements during the 1980's concentrated on building special database hardware. That approach was useful but the building of customized hardware was expensive. In contrast, the emergence of inexpensive commodity processors and other hardware shifted the focus, during the 1990's, to one of building extensible parallel computers. Three distinct architectures have been reported in the literature; they are shared memory, shared disk, and shared nothing. This presentation, made at the Annual ADSRC Meeting on September 96, contrasts the three architectures.
COURSE DEVELOPMENT
IN
PARALLEL AND DISTRIBUTED
DATABASE MANAGEMENT SYSTEMS

Jamal Alsabbagh and Carolyn Winston

ADSRC - Grambling State University
OUTLINE

- Context and Plan for This Work
- Overview of Parallel Database Systems
CURRENT AND PROPOSED DATABASE COURSES
(AT GSU)

DB Design
(CS451)
- ER modeling
- The relational model
- Relational DB design

DBMS
(CS452)
- Query Optimization
- Concurrency/Recovery
- Security
- OODBMS

Advanced DBMS
(CS4xx)
- Parallel DBMS
- Distributed DBMS
- Dist. Objects (CORBA)

(CS3xx)
Classification of Parallel Database Systems

- Shared Memory (SM); also called Shared Everything (SE)
- Shared Disk (SD)
- Shared Nothing (SN)
Shared-Memory (SM) Parallel Database Systems

- All processors directly access, in a symmetrical fashion, all main memory and disks.

- In general, the operating system allocates processors to processes.

- Processors have local caches to reduce network traffic, but loading/flushing caches can degrade performance.

- Hardware-specific solutions are required to ensure coherence among the caches (e.g. processors continuously snoop the shared bus to see if their cached data is required elsewhere.)

- Typical Hardware: IBM 3090, IBM 370, Bull DBS8, Encore, Sequent Symmetry
Shared-Disk (SD) Parallel Database Systems

- Each node has its own memory and may itself be an SMP box.
- The nodes share the same disks logically (and may be physically too).
- System (or application) software must ensure the coherency among multiple copies of disk pages requested by different nodes.
- A query or update by a node requires it to:
  1. Transmit, to all other nodes, an intention to query/update the database.
  2. If the required page is currently being updated by any other node, then wait until it is released.
  3. Read or receive the required page.
  4. Perform the query or update.
- Typical Hardware: DEC VM Cluster, SUN Sparc 1000 cluster
Shared-Nothing (SN) Parallel Database Systems

- Each node has its own local memory and its own local disks.
- The database is partitioned across the nodes, thus allowing I/O parallelization
- Each node acts as a server for its local data.
- An update by a processor requires:
  1. Transmit a request for update to the relevant server.
  2. The server performs the update, locally.
  3. The server acknowledges the success back to the requester.
- A query by a processor requires:
  1. Transmit the query to the relevant server.
  2. The server performs the query locally.
  3. The server sends the query result to the requester.
- Typical Hardware: AT&T 3600, IBM SP2, nCUBE, VAXcluster
## Metrics for Evaluating the Three Architectures

<table>
<thead>
<tr>
<th>Metric</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Price</td>
<td>Using commodity hardware reduces system cost.</td>
</tr>
<tr>
<td>2 Throughput</td>
<td>Inter-query parallelism improves throughput.</td>
</tr>
<tr>
<td>3 Response Time</td>
<td>Intra-query parallelism improves response time.</td>
</tr>
<tr>
<td>4 Speedup</td>
<td>Ideally, twice the hardware should solve the problem in half the time.</td>
</tr>
<tr>
<td>5 Scaleup</td>
<td>Ideally, twice the hardware should solve twice the problem in the same time.</td>
</tr>
<tr>
<td>6 Startup Cost</td>
<td>Preparing a query for parallel execution is an overhead.</td>
</tr>
<tr>
<td>7 Interference</td>
<td>Processors slow each other when competing for shared resources.</td>
</tr>
<tr>
<td>8 Load Balancing</td>
<td>Ideally, all the processors should be working concurrently.</td>
</tr>
<tr>
<td>9 Comm. Overheads</td>
<td>Ideally, sub-problems of one problem should require least communication.</td>
</tr>
<tr>
<td>10 Data Availability</td>
<td>It is desirable to be able to tolerate failure of some nodes.</td>
</tr>
<tr>
<td>11 Portability</td>
<td>Porting centralized DBMS software should be relatively easy.</td>
</tr>
</tbody>
</table>