CS 344 Databases

Lecture: 28 October 2011

Submitted by: Nikita Garg (08012309)
Tanvi Rai (08012326)

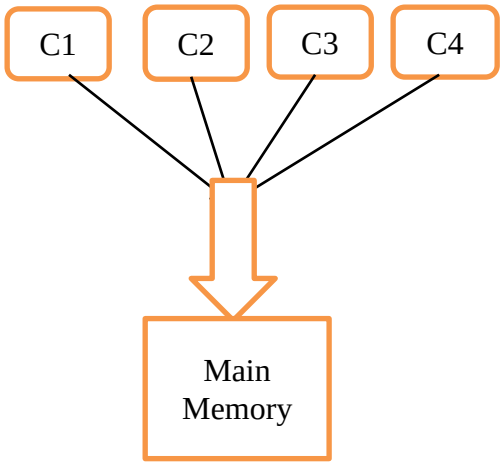
In the earlier lectures we have been dealing with the DBMS implementation for the basic form of hardware that has a single core CPU. With the improvements in hardware like multiple cores and multiple processors, DBMS implementation should be accordingly changed to take advantage of the improved hardware.

The improvement in hardware can be categorized as:

Multi Core -----> Multi Processor -----> Cluster Computing -----> Distributed Computing

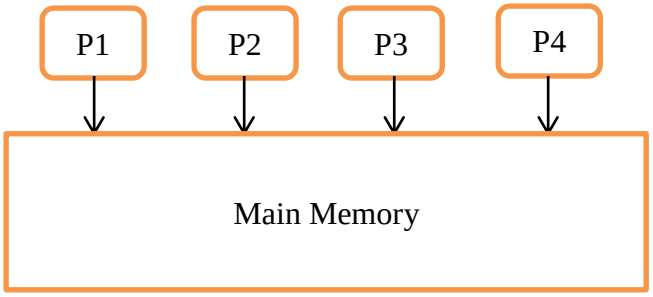
Multi Core Processor:

- These have multiple processing units called cores, which share a common path to the main memory (RAM).
- The bottleneck here is memory access.
- Applications which do not retrieve much data from RAM work faster on multi-core processors.



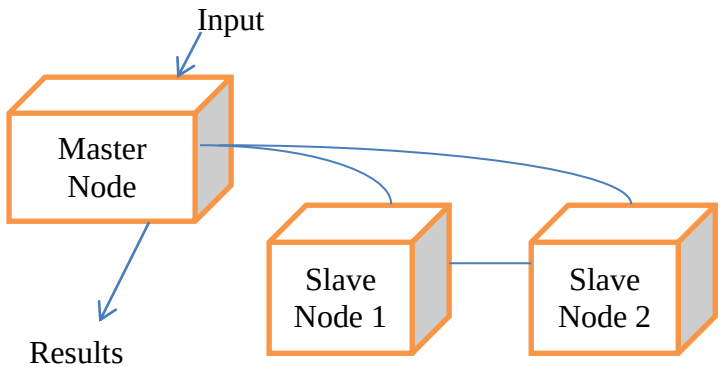
Multi Processors:

- These have multiple processing units which interact independently with the main memory (RAM) unlike multi core processors.



Cluster Computing:

- Multiple machines are set up in a single location connected by a high speed local area network working as a single machine.
- The multiple nodes are controlled by a single entity called the master node.
- The bottleneck here is communication between different nodes within the network.



Distributed Computing:

- Multiple autonomous computers communicate through a computer network.
- The nodes can be at very different locations and the network speed is usually slow as compared to cluster computing.
- Each node has its own memory and communication takes place by message passing unlike parallel computing where nodes share memory.
- Different entities control different nodes, so there is no single point of failure.

Now we go from a serial implementation of DBMS to a parallel implementation taking advantage of the improved hardware. We will discuss the impact of the enhanced hardware on some basic operations related to databases.

Multi-Core:

Selection (σ):

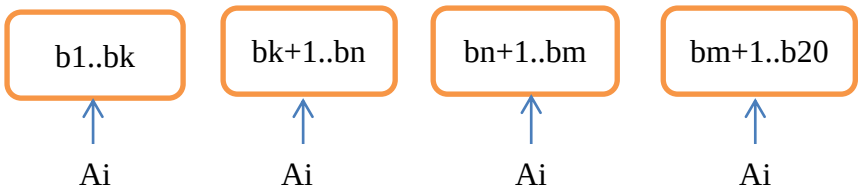
- The load can be balanced among different cores and the respective outputs can then be merged by a DBMS query manager in a sequential manner.
- The load division is also taken care of by the DBMS.

Projection (Π): Same implementation as described for selection opeartion.

Cross Product (x): Here, the load should be divided as per the size of the records and the cache size of each core. For example:

Consider two files A with 4 pages and B with 20 pages each. If we have to evaluate the cross-product of these two files, one method could be to divide the file B among the cores and then each core could read each page of file A one-by-one.

A (pages): a1, a2, a3, a4
B (pages) : b1, b2, …, b20



Natural Join (⋈): Assuming that the data is sorted based on the common attribute, there is a considerable improvement in evaluating the natural join expression as compared to the cross product because only a fraction of data needs to be processed for natural join.

Set Operations (union, intersection etc.): The implementation is similar to that for natural join expressions.

Modification: Operations like deletion of records, changing schema etc are not affected much with the improvement in hardware. But then, the frequency of these operations being carried out is very less as compared to the other operations described above

Multi-Processor:

Projection (Π), selection (σ) and cross-product (x) operations are better implemented in the muti-processor environment as compared to multi-core because there is no memory access bottleneck in the former.

The Natural Join operation is a little better but there is no significant improvement.

Thus, a faster DBMS implementation is achieved with multi-processors than just multi-cores but the cost is high in the former.

Cluster Computing:

The most important aspect when it comes to using cluster computing for DBMS implementation is how is the data stored.

Some general approaches to data division are:

1. Horizontal Partitioning (Row based division)
2. Column based division
3. Store each table in a different machine

One of the above approaches is chosen depending upon the workload.

Assuming horizontal partitioning:

Selection (σ): The selection operation implementation in a clustered environment is similar to the implementation in a multiprocessor environment.

Projection(Π): Same as for the selection operation.

Cross-Product (x):

- Here a part of the product can be done locally quite easily but for the rest, a lot of communication is required between nodes.
- In this case, it would have been much faster if there had been a copy of the table in each machine.

Assuming column based division:

Selection (σ):

- A lot of communication is required between the nodes.
- First the primary key needs to be acquired corresponding to the selection criteria and then all the attributes need to the selected corresponding to the primary key from each machine.

Projection (Π): Much faster as there is no need to read the whole table to get a particular attribute.

Distributed System:

With distributed systems, the advantage of having a fast communication network as is the case with cluster computing, is lost. Also the messages may get lost.
There exists the same basic problem of how to distribute data across the nodes like cluster computing. Another factor that should be taken into consideration is that the speed of the nodes could differ widely and hence data should be distributed accordingly.

Selection (σ) and Projection(Π): Same scenario as cluster computing.

Cross Product (x): The problems of cluster computing gets amplified here as the communication between nodes is slower.