ISSN No. 0976-5697

Volume 2, No. 3, May-June 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Architecture of shared-nothing cluster to fast accessing and ensure the availability of data in distributed database environment

Deepak Sukheja*
Priytam institute of tech and managment,
Indore, India
d_sukheja@rediffmail.com

Umesh K. Singh
Institute of Computer Science,
Vikaram University, Ujjain, INDIA
umeshsingh@rediffmail.com

Abstract: Database systems have been essential for all forms of data processing for a long time. In recent years, the amount of processed data has been growing dramatically, even in small projects. At the other hand, database management systems tend to be static in terms of size and performance, which makes scaling a difficult and expensive task. Enterprises may have multiple database systems spread across the organization for redundancy or for serving different applications. In such systems, query workloads can be distributed across different servers for better performance. In this paper, we focus on complex queries whose evaluation tends to be time-consuming and design the secured share nothing clustering architecture to improve the performance of application and also assure to the user to availability of the data. The proposed architecture is very helpful towards a two phase query optimizer. In the first phase, the synchroniz and decomposes a query into subqueries and tranfer them to appropriate cluster nodes. In the second phase, each cluster node optimizes and evaluates its subquery locally.

Key word: cluster technology, database technology, shared-nothing cluster, query processing, query optimization.

I. INTRODUCTION

Database technology has become a common requirement in almost application and due to high performance and cost effectiveness, cluster of workstations has recognized in recent years. It is cheap and simply available to all kind of database based applications. Consequently, *database clusters* likewise are becoming a reality. A database cluster is a network of workstations (PCs) and each node (workstations) runs the shelf database. Due to high performance, cost effectiveness and simply availability of clustering technology have gained popularity. A cluster can be built either from asymmetric or symmetric processors,

But generally it is built from Symmetric Multi-Processors (SMP). Standard distributed physical design schemes are mentioned in [1] to data distribution and determines the query evaluation. The main design alternatives are full replication, vertically partitioning and clustering for improving intra-query parallelism. In the ideal case, a database cluster allows to scale out, i.e., it allows to add more nodes in order to meet a given performance goal, rather than tuning the nodes.
The main motivation behind parallel processing application is that we need to solve bigger problems with minimum resource requirements beyond current limits. Parallel computing is the way because it performs work in lesser time, solve large problem easily, saves cost and provides concurrency [2]. Data intensive applications that require huge databases waste a lot of time in scanning and searching. The optimal way to run these applications is to use the computational power of more than one system by distributing the workload among the nodes in a cluster.

Traditional serial computation has many serious limitations like memory size and speed, limited instruction level parallelism, power usage, heat problem etc. With the wide availability of parallel computing platforms like HPC centers, local Linux clusters, multiple CPU's and GPU's(graphics processing unit) the above mentioned limitations can be overcome.

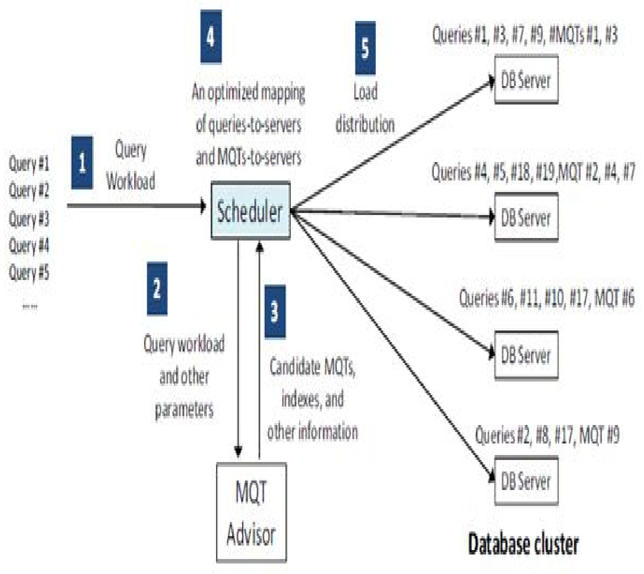

Query #1
Query #2
Query #3
Query #4
Query #5
1
Query Workload
2
Query workload and other parameters
3
Candidate MQTs, indexes, and other information
4
An optimized mapping of queries-to-servers and MQTs-to-servers
5
Load distribution
Scheduler
MQT Advisor
Queries #1, #3, #7, #9, #MQTs #1, #3
DB Server
Queries #4, #5, #18, #19,MQT #2, #4, #7
DB Server
Queries #6, #11, #10, #17, MQT #6
DB Server
Queries #2, #8, #17, MQT #9
DB Server
Database cluster

Figure1: Traditional System

According to Thomas Phan, Wen-Syan Li in [6], traditional DBMS work into 5 steps: (1) A query workload is submitted to DDBMS scheduler. (2) The scheduler gives the queries to an MQT(Materialized Query Tables) Advisor product, which (3) returns a set of candidate MQTs and associated indexes. (4) The scheduler runs a search heuristic against possible combinations and produces query-to-server and MQT-to-server mappings. (5) The mappings are used to distribute the queries and MQTs onto the servers. The load distribution is to divide the workload into multiple sub-workloads and assign each sub-workload to a database server in some greedy manner, such as a round robin distribution. This simple solution will not work for several reasons:

- Queries routed to a database server may not be collocated with their needed MQTs;
- Some MQTs may not fit in the data server that has a limited disk space;
- Some sub-workloads may be more expensive to execute than others, so some server may be idle while other servers are still busy;
- Some servers may be more powerful than others.

II. OBJECTIVES

The major objective of this paper is to optimize query processing time by providing parallelization with the use of distributed database systems to store data rather than overloading a single DB Server machine. In this paper, we propose a framework for coordinating and optimizing execution of complex query workloads across a cluster of database servers with shared-nothing architecture. For a database cluster, such an optimization is achieved when the maximum completion time of the workloads across all database servers is minimized. The completion time at each database server includes MQT and index building time and query workload execution time.

III. DESIGN AND METHODOLOGY

This section provides a detailed illustration of proposed architecture. The architectural goal is to develop share-nothing distributed system architecture for fast query processing and highly availability of the data. The basic terminologies used to develop the architecture are: Cluster is a group of Server (A, B, & C) configured to work together to serve clients in a similar fashion[5]. Each server participating in a cluster is called as node i.e. A, B & C is nodes. Clusters can be configured in two modes:

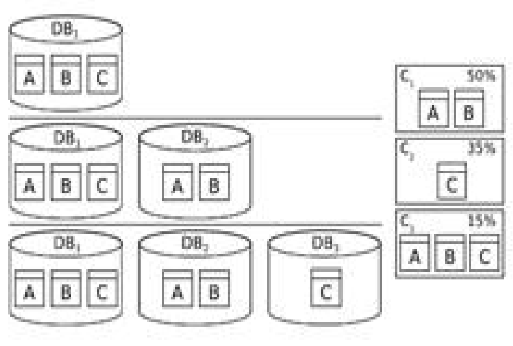


Figure 2: Cluster of servers

Active/Passive Mode, In the Active/Passive mode only the active server serves the client requests. One instance of DB Server is installed on both the server systems. The second server in the cluster is configured as passive node. Second becomes active node only in cases where the first server fails. The Active/Passive cluster Mode is providing the highly availability and fast accessing of database.

Active/Active mode, In Active/Active mode both the servers serve the client requests. The MSCS arbitrarily chooses one of the servers to serve the client requests. If any of the active nodes fails, its resources are moved to another active node.

Network Load balanced Cluster: Uses a load balancing architecture, which means that a resource can be active on ALL servers in the cluster at any one time. Because of this, it is well suited to applications that do not maintain state

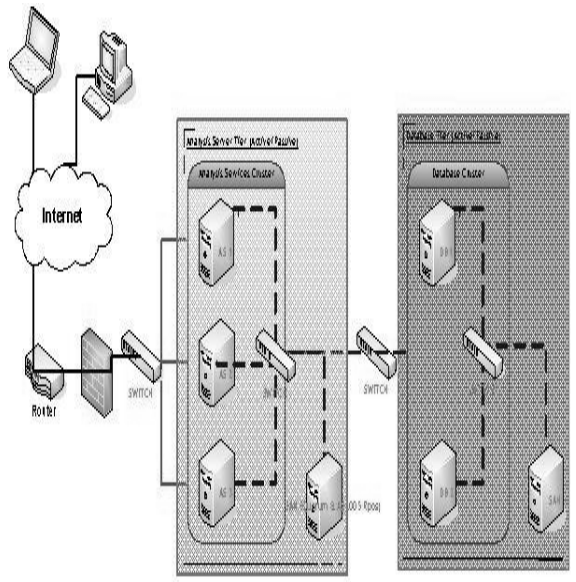


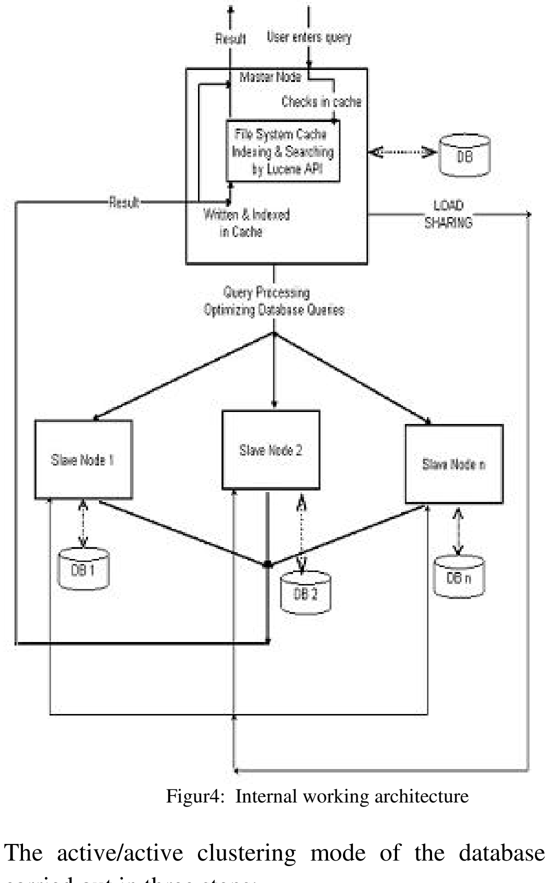
Figure3: open architecture of Shared-Nothing Cluster

IV. WORKING

Shared-nothing architecture works on the principle of ownership, that each node in a cluster has individual ownership of the data on that node. Every node shares no data with the other nodes of the cluster, hence the term shared-nothing. When you move from a single server to multiple servers in a shared-nothing cluster, you must divide the data across the servers.

In a heterogeneous cluster many factors have to be taken into account [3] to decide how much task is to be allotted to each node/server. These factors include processor speed, disk storage, input/output and network latency. While in case of homogeneous cluster we have to only take network latency into account. Distribution of database has to be done according to the execution time taken by a particular node/server. Less the execution time more the data given to that node. It can present load sharing in a mathematical model as follows:

Let D be the database size and S be a set of heterogeneous machines; S = {N1, N2,..,Nn}, where N denotes a machine and n be the number of nodes. Ni(Ci,Si,Li) represents a machine having computational power Ci, disk storage Si and estimated network bandwidth Li; Li={aij, i ≠ j}, where aij is the average latency between nodes Ni and Nj. Let Ei be the execution time taken by the node Ni.
This can be written as: Ei = Ci + Ii, where Ci is estimated computational time and Ii is estimated I/O time.


Result
User enters query
Master Node
Checks in cache
File System Cache Indexing & Searching by Lucene API
DB
Result
Written & Indexed in Cache
LOAD SHARING
Query Processing Optimizing Database Queries
Slave Node 1
Slave Node 2
Slave Node n
DB 1
DB 2
DB n

Figur4: Internal working architecture

The active/active clustering mode of the database can be carried out in three steps:

1. Estimating the number of nodes in which the database has to be partitioned.
2. Finding the rank of each node.
3. Partitioning database.

V. IMPLEMENTATION

This has been implemented through the following modules:
1) Load Sharing: The module is to perform load sharing through database distribution. The flow is shown in figure 5. To avoid direct access to the slave node and individual entering of records into the data base, the data can be directly entered via master node. The records are retrieved from the master node and then are partitioned depending on the number of nodes present in the cluster.

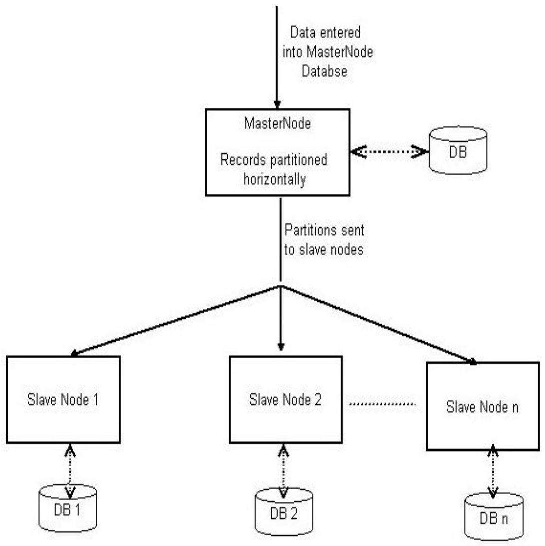

Data entered into MasterNode Databse
MasterNode
Records partitioned horizontally
DB
Partitions sent to slave nodes
Slave Node 1
Slave Node 2
Slave Node n
DB 1
DB 2
DB n

Figure 5: Load Sharing

2) MPI (Message Passing Interface): This module establishes the communication interface between the remote nodes. To establishes the remote communication use TCP/IP connection. Once the connection is established, the slave nodes are ready to accept the query from the master node and also reply to master nodes.
3) Indexing Database Queries: A database index is a data structure that can be created using one or more columns of a database table and its main advantage is that it improves the speed of operations on a database table, providing the basis for both rapid random look ups and efficient access of ordered records. Since indexes do not consist of the details that are present in the table and contains only the key fields according to which the table is arranged, it occupies much lesser space than the table and thus provides the chance of storing the index in memory of those tables which are too large. Whenever the user enters a complex query, an index containing the attribute as columns present in the query is created. Now the next time when the user enters a query containing similar attributes, the index is first accessed rather than the table and thus leading to time optimization.

VI. CONCLUSION

In this paper, we have designed and implemented a high performance and scalable inter node communication within a cluster for fast query processing in a heterogeneous database environment with replication scheme. In future works will perform both vertical and horizontal partitioning of the database.

VII. REFERENCES

[1]. Philip Hatcher, Mathew Reno, Gabriel Antoniu, Luc Boug?, "Cluster Computing with Java," Computing in Science and Engineering, vol. 7, no. 2, pp. 34-39, Mar./Apr. 2005.

[2]. Sanan Srakaew, Nikitas A. Alexandridis, Punpiti Piamsa-nga and George Blankenship, “Content-based Multimedia Data Retrieval on Heterogeneous System Environment,” in International Conference on Intelligent Systems (ICIS-99) , Denver, Colorado, June 24-26, 1999.

[3]. Röhm U., Böhm K., Schek H.-J., “*Cache-Aware Query Routing in a Cluster of Databases”,* In Proc. 17th Int. Conf. on Data Engineering, ICDE2001.

[4]. Oracle Corporation, “*Clustering Database Applications to Lower IT Cost*”, White Paper, Microsoft Corporation, octumber 2010.

[5]. Lorinda Visnick , “ Clustering Techniques” , A Technical Whitepaper, www.objectstore.net.

[6]. Thomas Phan , Wen-Syan Li, A request-routing framework for SOA-based enterprise computing, Proceedings of the VLDB Endowment, v.1 n.1, August 2008.

[7]. T. Rabl, M. Pfeffer, H. Kosch,” Dynamic Allocation in a Self-Scaling Cluster Database”, VLDB DMG, Volume 20, Issue 17 Pages 1977–2088.