A BETTER WAY FOR FINDING THE OPTIMAL NUMBER OF NODES IN A DISTRIBUTED DATABASE MANAGEMENT SYSTEM

[1]Rashed Mustafa, [1]Md. Javed Hossain and [2]Thomas Chowdhury
[1]Department of Computer Science and Telecommunication Engineering
Noakhali Science and Technology University, Bangladesh
[2]Department of Computer Science and Engineering
Chittagong University of Engineering and Technology (CUET), Bangladesh
E-mail: rashed_mustafa78@yahoo.com, javed_abc@yahoo.com and thms_chy@yahoo.com

Abstract: *Distributed Database Management System (DDBMS) is one of the prime concerns in distributed computing. The driving force of development of DDBMS is the demand of the applications that need to query very large databases (order of terabytes). Traditional Client-Server database systems are too slower to handle such applications. This paper presents a better way to find the optimal number of nodes in a distributed database management systems.*

Keywords: *DDBMS, Data Fragmentation, Linear Search, RMI.*

1 Introduction

Now a day, commercial data is increasing rapidly. Due to technological innovation, the size/price of storage devices improves briskly. For business and scientific purposes, sometimes, data size touches the thousand-terabyte limit. To process and to keep such a big databases, the demand of supercomputing has risen over twelve years. But the price of the parallel system is very high and the price of Personal Computer (PC) has fallen and its performance has been increased tremendously. So, recently, developing a Distributed system has become popular.

This paper is organized as follows: section 2 briefly discussed about distributed database management system, in section 3 proposed strategy has been analyzed, section 4 discusses its analytical result and finally section 5 conclude the paper which concentrates future work also.

2 Distributed Database Management System (DDBMS)

In a DDBMS, data are scattered in several nodes for security, faster processing or any other business purposes. For retrieving information it needs querying of data. Most applications require searching operation than inserting or updating a data to a database. To design a Distributed Database system for such applications, this research proposes to consider optimal number of nodes for ensuring the search faster.

The main disadvantages of a DDBMS are network overhead, process startup time and database connectivity time [1]. To simplify our following discussion we express these disadvantages as *Distributed System Overhead (DSO)* [2]. For small amount of data, a distributed database (DDB), which consists of hundreds of node, suffers huge network congestion. As a result, the performance will degrade. On the other hand for large amount of data (terabytes), the performance of a distributed database system, which or system that consists of few nodes, may also degrade due to heavy load in each node [4-6].

The above discussion promote that, number of nodes is a prime concern to design a DDBMS. This research identifies an optimistic number of nodes of such a system. There are two approaches to store tuples in the DDBMS [7].

2.1 Replication

Identical replicas of the whole database are stored in each node. This storage system increases the robustness of database [5]. But the main disadvantage of this storage system is that is takes a huge amount of disk space to store the data. Also, as the system needs to ensure that all replicas of a relation r must be consistent so whenever r is updated, the update must be implemented to all nodes of the system. Thus update increases overhead

in replication storage system. But it enhances the read-only operations [8].

2.2 Fragmentation

Data can be fragmented vertically and horizontally. Horizontal fragmentation splits the relation by assigning each tuple of r to one or more fragments. Whereas vertical fragmentation splits the relation vertically, i.e. one more relation will be created from a relation [9,10].

3 Theory and Analysis

In worst case, the complexity of linear search algorithm, *f(n)=N*, *N* is number of data in an array, where's the particular element is being searched.

If the comparison time of the element is t_c, then the total comparison time required for worst case in linear search is $N \times t_c$.

If *N* is very big and the data is kept in several nodes of a network, the comparison time for each node will be,

$T_d = (N / P) \times t_c$, *P* is the number of nodes.

One demerit of the distributed system is that we can't omit *DSO*. Suppose, the time taken to initialize the network for each node is t_n, the time for linear search in a distributed database is

$$T_d = ((N / P) \times t_c) + Pt_n \quad (1)$$

If P=1 i.e. the search is sequential then according to Equation (1), the comparison time will be

$$T = (N \times t_c) + t_n \quad (2)$$

There're two part of Equation (1), $(N / P) \times t_c$, this part represents the time taken for a comparison in each node. It will be decreased if we introduce more nodes without increasing the amount of data. On the other side, Pt_n is *DSO*, suffered by the application. It will be increased if we introduce more nodes without increasing the amount of data.

Now, we take a look in Equation (1), if we introduce more nodes in the system, more DSO will be introduced. But it will decrease comparison time i.e. the searching will be optimal in the DDB only when

$$(N / P) \times t_c = Pt_n \quad (3)$$

From Equation (3), we can write that the optimal number of nodes for worst case will be

$$P = \sqrt{\frac{Nt_c}{t_n}} \quad (4)$$

And for average case it will be

$$P = \sqrt{\frac{Nt_c}{2t_n}} \quad (5)$$

Here, *P* is the number of nodes, *N* is the amount of data, t_c is single compare time, t_n is network overhead. So, if we know *N*, t_c, t_n we can easily find out *P*, the number of node in the distributed system for optimum linear searching.

4 System Architecture

For practical purpose, we establish a distributed system using 16 nodes and one workstation, which act as application server. All nodes with identical configuration consist of 128 RAM, 16 GB HDD, P-II Processor with 450 MHz. A 100-based D-Link DES-1024 fast Ethernet switch with 16K memory is used for network. Operating system (OS) of all nodes and workstation is Windows 2000. MySQL and java RMI are used for distributed database and distributed processing respectively [12].

5 Proposed Strategy

Here, at first, we have taken results using 1,2,4,8,10,12,16 nodes according and, it considers 100,000,000 rows in a table where the data has been searched. The data is kept in the nodes by round robin fashion [11].

We measured the comparison time of each node is $0.81*10^{-3}$ ms/compare. The startup costs (Initialize Database + Network overhead) = 982 ms. So, from (5) theoretically, *P* = 9.09

6 Result Analysis

Analytical result shown in Table 1. From table 1, it is decided that the value of the *P* lies between 8 and 10. So, we have also taken data for 9 nodes. Table 2 shows the result. From the above results, we see that the number of nodes should be 9 for the best performance and this is equal to our theoretical result.

The following two concepts are useful in comparing sequential and distributed process [2].

The speedup, $Sp = T_s / T_d$

Which is the ratio of the time taken for the optimal sequential algorithm and the time taken required by the proposed distributed algorithm to solve the same problem.

Efficiency, $E_p = S_p / N$

Table 1 Represents the time taken (Seconds) in each node

Number of Nodes	Number of tuples per Nodes	Time Taken (Second)
1	100,000,000	81.75
2	50,000,000	42.03
4	25,000,000	27.54
8	12,500,000	18.56
10	10,000,000	19.65
12	8,333,333	22.44
16	6,250,000	25.29

Table 2 Time taken for 9 nodes

Number of nodes	Number of tuples per Nodes	Time Taken (Second)
9	11,111,111	17.83

Which measures the fraction of time that a typical node (processor) is usefully employed.

The following table 3 depicts speedup versus efficiency for corresponding nodes.

Table 3: Calculation of Speedup and Efficiency.

No of nodes	Speedup	Efficiency
1	1	1
2	1.95	0.975
4	2.968	0.742
8	4.405	0.551
9	4.585	0.509
10	4.1603	0.416
12	3.643	0.304
16	3.233	0.2021

The following figures described graphically on the basis of the results of Table 1 and 2.

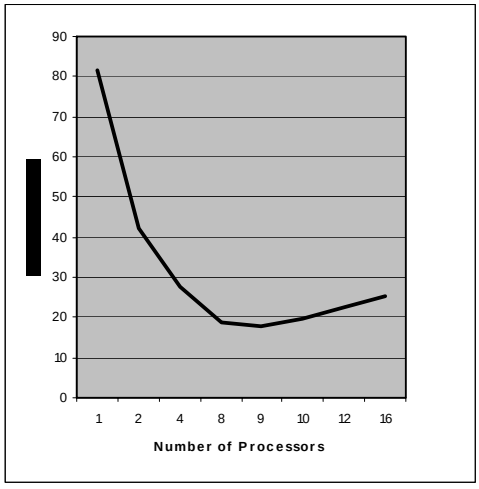

Fig. 1 Number of Processors Vs Time Taken (Second)

Figure 1 depicts that time decreases due to large number of nodes used, but the result is not always proportional to number of nodes. It is clearly shown on the graph that in the case of 9 nodes, time is minimum, which supports our experimental and also theoretical analysis.

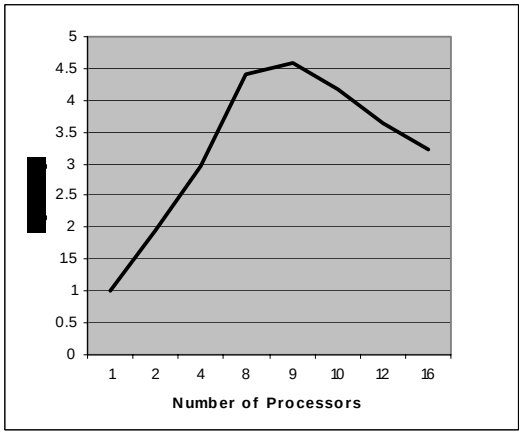

Fig. 2 Number of processors versus speedup ratio

Figure 2 described number of nodes versus speedup ratio. It is shown that speedup ratio goes upward when small number and unoptimized (excluding 8 and 9) number of nodes used.

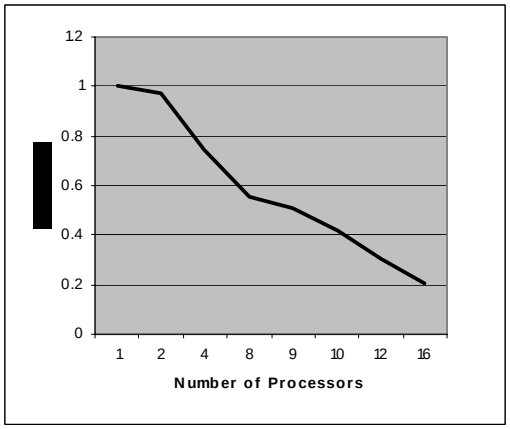

Fig. 3 Number of processors vs. efficiency

Figure 3 elucidate efficiency in accordance with number of nodes. Here also clearly shown that efficiency is inversely proportional to number of nodes.

7 Conclusion

Distributed Database is a growing technology. A huge research work has been done on it and is continuing. In this paper, theoretically and experimentally, we have shown a way to find out the optimal number of nodes for linear search in a distributed database. A related work has been done [3], which does not show the particular number of nodes for optimal searching. In this paper it is clearly identified a number of optimistic nodes for a particular searching.

References

[1] M. Adelgurefi and K.F. Wong. *"Parallel Database Techniques, IEEE-CS press june 1998 (ISBN 0-8186-8398-8)"*

[2] D. DeWitt, and J. Gray "Parallel Database Systems: The Future of High Performance Database Systems", CACM, Volume 35, No. 6, June 1992.

[3] D. DeWitt, and J. Gray "Parallel Database Systems: The Future of High Performance Database Systems", CACM, Volume 35, No. 6, June 1992.

[4] Nowshaba Durrani, Mohammed Anwer, A comparison of parallel database search algorithm on a 16 node cluster, ICCIT 2004, Brac University, Dhaka.

[5] 5. Annaratone, M., Pommerell, C., and Ruhl, R. (1989) Interprocessor communication speed and performance in distributed-memory parallel processors. IN 16th Annul Symposium on Computer Architectures, pp. 315-324, June 1989.

[6] M.B. Ibiza-Espiga and M.H. Williams. *"Data placement strategy for a parallel database system"* in proceedings of DEXA'92 Conference, pages 469-474, Valencia, Spain, September 1992.

[7] *D. Gibson, J.M Kleinberg, and P. Raghavan*. Clustering categorical data: An approach based on dynamical systems, In proc. 1998 Int. conf. Very Large Databases (VLDB'98), pages 311-323, New York, Aug, 1998

[8] *Java as a Basis for Parallel data Mining*, In proc. 7th Intl conf. On High Performance Computing and Networking HPCN Europe, April 12-14, 1999, Amsterdam, The Netherlands, LNCS 1593, Springer verlag, pp.884-884

[9] Silberschatz, Korth and Sudarshan, Database System Concepts, 4th ed., McGraw-Hill, pp.710-712, 2002.

[10] Dimitri P. Bartsekas and John N. Tsitsiklis, Parallel and Distributed Computation: Numerical Methods, Prentice-Hall, pp. 27- 28,1989.

[11] Buya R. *"High performance Cluster Computing"*, Vol-1,2, Prentice Hall PTR.

[12] Sun Microsystems: *Remote Method Invocation specification,http://java.sun.com/products/jdk/rmi*