Modeling an Automated System,
that manages the mapping and splitting of parallel databases for better efficiency of access of distributed systems[#].

Saleem G. Zougbi
Bethlehem, Palestine (*saleem@bethlehem.edu*)

Abstract

Modeling an automated system, that manages the mapping and splitting of parallel databases for better efficiency of access of distributed systems. The problem of splitting this large database into clusters and shipping them from one place to another across the nodes of the network would yield to a very reasonable and agreeable performance for large distributed DB systems. Selecting the splitting method and implementing it according to a distributive plan could be programmed to run automatically. This paper is based on a research that focuses on the idea of using adaptive digital filtering technique in order to monitor, decide and implement parallelism in distributed heavy data base applications.

Keywords

Parallel databases, distributed processing, adaptive information management, database structures, adaptive filtering applications.

1. The problem

The centrality of large information systems that use huge databases is no more realistic. Online applications have explicitly stressed the need for technical ability to access large data components. This access could be in form of simple look-up or highly volatile in terms of updates and/ or processing. The design of data bases naturally started to involve making decision on the splitting, fragmentation and positioning of data, and to a high degree programs too, across different nodes and sites of a computer network. In this research, a new priority approach was addressed. It is the frequency of accessing database information, whether in horizontal or vertical structures, with the dynamic nature is represented through frequency analysis. The higher order magnitude of the database components imply heavier communications tasks, and hence poor and in efficient database information access. As an example, consider clinical information systems that house information about patients. Assuming that large numbers of patients and the type of information is mainly object oriented with a lot of higher resolution medical images for example, the aggregate of data records pertaining to a particular patient becomes of vital importance to access whenever medical examination is carried out for this patient. If this large database resides on a central server, and having medical staff consulting this database from remote sites, the access efficiency becomes a serious impediment to users. Serious and urgent action becomes mandatory if access becomes more permanent, i.e. more frequent. An example is when the patient moved to the site of the other remote node. In such a situation, shared-disks databases become inefficient and shared-nothing approach becomes a very realistic approach to use. The critical issues become *when to split*, and *what factors* the splitting decision is to be based on, to achieve best efficiency possible in terms of optimizing access and SQL requests over the larger network.

As a matter of fact, the presence of a parallel object database server would add an important task besides its normal SQL-oriented queries. In general, the classical central DB system is characterized by a unified information structure. However well designed and elaborate this system is, the fact remains that there are user requests and queries from other systems online and that the information aggregates are relatively large and require heavy access time and storage.

Normally the source of requests comes from a quasi-static client. This means that certain parts of the information aggregates are requested from a client that does not change. Imagine that this DB is a huge patient's information aggregate. This patient's information is requested normally in a certain city and a certain clinic, and very rarely it would be queried from other places.

The other characteristic is that this information aggregate would not concern other users or requests. Therefore we would look for a place to host this aggregate that is the closest (in communication routing and least overhead). *Therefore the problem is to devise an algorithm where decision is taken as to where to host (split it from the central DB and transfer it to that place), and how this process can be automated.*

2. The proposed approach

Examine figure 1. The classical central system is shown in a four-tier structure: the users, the database user interface, the DB controls and the actual DB system. The users can be different clients over the network, but the DB system is present on a server, that

[#] This research was started while the author was a research visitor at the International Institute of Software Technology, United Nations University, Macao, SAR China in 2004, and completed in 2005.

is serving other clients on line. In contrast, if the fourth tier is split into n chunks, and physically stored in different nodes, the optimal performance criteria described above can be achieved. Notice that a shared-nothing parallel database is the main concept of this splitting. Figure 2 shows such a solution.

In this research, I propose an approach that is based designing a software system that would perform the task of splitting in an adaptive automatic way depending on user SQL requests. The approach is based on the analysis and study of the procedural performance specification, with application of digital filtering techniques. This is done in the following manner:

- Modeling the SQL's (type, frequency, and overhead costs (such as time, response, volatility)
- Decision-making support in terms of the model of the SQL's
- Ability to adapt to changing facts and characteristics of the model

3. Current Research

Currently, the state-of-the-art in this field is to focus on research into parallelism, rather than on the ODBMS interfaces.

Although there has been little work on the usage of parallelism to provide scalable performance in Object Database Management Systems (ODBMS), the dynamism of this parallelism is the critical issue of this suggested solution.

One could argue that the very expensive solution of using machines like the IBM z9-109 model for example, where multiple I/O architecture permits fast and efficient I/O access and management, a much more pragmatic and cheaper solution is highly desired specially in applications of users which have limited technology and support.

Previous research on the issue of the design of distributed databases involved making decisions on the fragmentation and placement of data and programs across the sites of a computer network. This process was a systematic analysis and design that required proper studies and then working on to indicate the most adequate fragmentation technique to be applied in each class of the database schema. It also includes finding the proper algorithms for horizontal or vertical splitting.

Some consideration was given to the idea of using techniques based on stochastic Petri nets (SPN) to analyze the statistical behavior of node-perceived dependability and performance of the splitting, but adaptive filtering techniques seemed to be a much simpler solution and can yield efficient results as well.

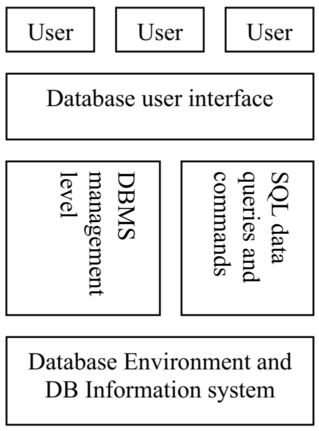
User
User
User
Database user interface
DBMS management level
SQL data queries and commands
Database Environment and DB Information system

Fig 1: A central disk-shared (and/or memory-shared) DB server

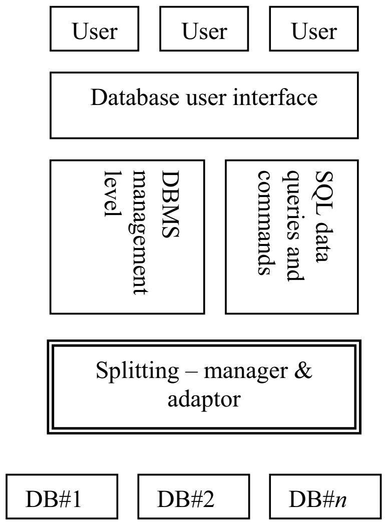

Fig 2: A shared-nothing parallel DB system on many DB servers

The literature survey examined several current approaches to this problem. Among these were how to balance dynamic loads on inter-transaction and intra-transactions [Rahm & Marek]. Some suggestions included skew handling of parallel joins [Dewitt et al.]or even to use buffer management based on priorities [Jauhari et al.], implementing transient versioning [Mohan et al.]. The research even also considered the symmetric & peer-to-peer storage relations: Samsara [Cox & Noble]. In all these algorithms and work, the concept of static situation was the dominant factor, even for short periods of time for user nodes. The problem emphasizes here is that such algorithm would have to be flexible and self-changing to fit unexpected patters of change of the incoming SQL's.

4. Assumptions

There are certain assumptions of the database that this research is based on. These are:

1. A central database with large number of records (high DB dimension), going vertically in different levels (depending on data type and content) and horizontally(depending on heterogeneity of the data components)
2. Sizes of data components differ widely in sizes, ranging from simple fields and chunks to huge aggregates of data.
3. Data records are to be accessed from anywhere on the wider network.
4. Likelihood of shared data records or aggregates is next to zero, i.e. needs are basically pier-to-pier

5. The decision of splitting

The concept behind the splitting process is the following:

Maintain the separation of the Database starting as a central and shared-disk system. A software agent, the suggested solution, referred to in this paper as the "smart splitter software (SSS)" is residing quietly interception the queries, and recording them.

The SSS is a software system that would automatically monitor, decide and implement the algorithms for the operation of this parallel shared-nothing object data base management system. It would depend on using computed figures of merit indicating the frequencies of access and SQL operations for each record (or cluster of records). Once these figures are computed, it processes these figures of merit in a real-time method into a digital filter that has response *redesigned automatically* to adapt to the changing needs of incoming SQLs. It also has the task to supervise, decide on and implement the motion that optimizes the overall performance of the DB. Notice that the motion is not only from the server to the node in question, it could be also in the reverse direction, i.e. instead of splitting, it could be joining, if the queries have changed source, and have become critical to justify return of these aggregates back to the server, or to another node, passing by the server.

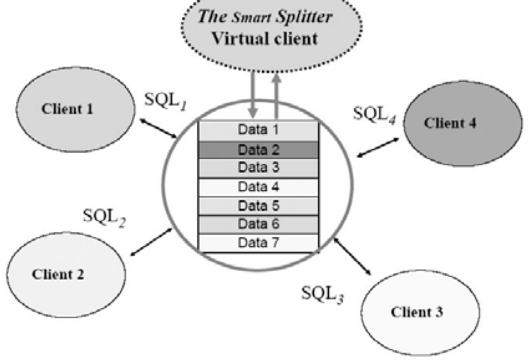

Fig. 3: The Central initial structure

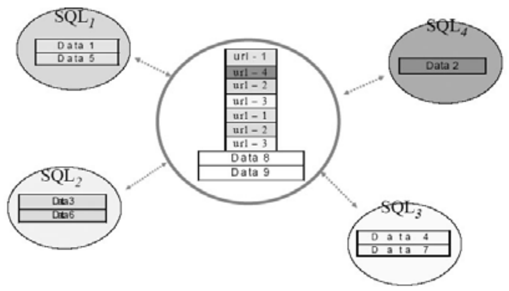

Fig. 4: The parallel shared-nothing data base

6. The Filtering process

As queries repeat, and according to sufficient reason, the SSS collects these requests and process them in a specially designed digital low-pass filter, of which the output is decided as low or high. If high, then a decision to split is taken and the data aggregate of which its queries has generated this data and decision, will be split and sent to the node that is the source of the most of these queries.

The issues here for optimal "filtering" and decision making are what kind of filter, and what are the parameters for it. In fact the filter coefficients are recalculated periodically, since queries and requests may change periodically, even from the main node that generates them. This is the "Adaptive" nature of the filter.

The requests are counts of frequency over a period of time. Every time a querying SQL comes for a specific data aggregate is kept as a discrete function

$$\overline{X}_j = \{x_{jk} : k = 0,1,\cdots M-1$$

where M is the number of time periods examined, and j is the j^{th} node.

In general, it is modeled as

$$X = \begin{vmatrix} x_{0,0} & x_{0,1} & \cdots & x_{0,M-1} \\ x_{1,0} & x_{1,1} & \cdots & x_{1,M-1} \\ \vdots & \vdots & \vdots & \vdots \\ x_{N-1,0} & x_{N-1,1} & \cdots & x_{N-1,M-1} \end{vmatrix}$$

The columns are the frequency counts of SQL's from each node in one time period, and the rows are the frequency counts of one signal node in all time periods.

The node that is most involved is the one that has a high enough total of counts in that period, selecting the node *j* as

$$X_j = \sum_{k=0}^{M-1} x_{j,k} \forall j = 0,1,\cdots N-1$$

By examining all values of X_j one will be selected as the active node. This could be as simply as selecting the j for which a maximum value of X_j is found, or perhaps selecting the one that has the largest positive standard deviation form the mean of the all the sums. In either case, the value of j that corresponds to the selected value represents the node that is involved in this process.

Now add all the frequency counts of that node, that is find the vector *V* corresponding to the node *j*

$$V_j = \{x_{j,0} \quad x_{j,1} \quad \cdots \quad x_{j,M-1}\}^T$$

Already an IIR filter has been designed with a pre-calculated cut-off frequency, as

$$Y_n = \sum_{k=0}^{N-1} a_k V^j{}_k - \sum_{k=0}^{N-2} b_k Y_k$$

Notice that the sequence *Y* and *V* change for *N* signals with shifting in time as SQL's keep coming in. This is like having memory only for the last N time periods.

The *a's* and *b's* are constants of the filter coefficients. Selecting the size of the filter, i.e. the value of *N* and these coefficients will determine how accurate and smooth the filter can work. The result of the filter is the output *Y* which can be sued to determine if it is high enough or not. If high enough, then a decision to split the data aggregate of which its frequency data has been used, to be split and transferred to node *j*.

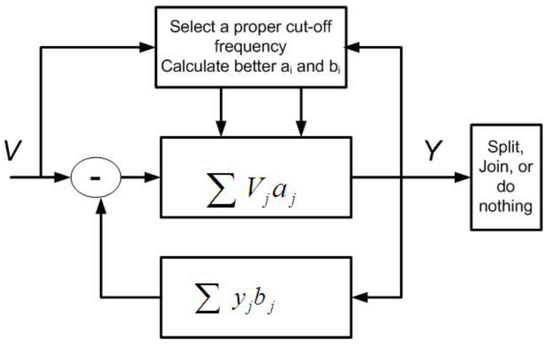
Select a proper cut-off frequency
Calculate better a_i and b_i
V
-
$\sum V_j a_j$
Y
Split, Join, or do nothing
$\sum y_j b_j$

Fig. 5: The Adaptive filter

Some important considerations are:

- In fact an IIR (Infinite Impulse Response filter) is picked up as the design to reflect "infinite duration difference calculus applied, i.e. a sense of permanent operation.
- The coefficients can be DFT, Z-transform, Fermat, Hadamaard or other suitable frequency domain discrete transform
- The Cut-off frequency (f_c) is not to be kept FIXED, it is reselected depending on historical statistical record of how many times a motion is done etc.
- After selecting it, use an algorithm to redesign the filter again (i.e. in a form of iteration: use existing values to recalculate a_i and b_j for new values such as *Butterworth or Chebyscheff)*

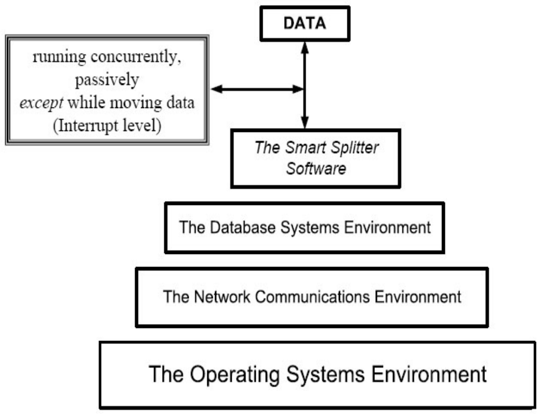
DATA
running concurrently, passively *except* while moving data (Interrupt level)
The Smart Splitter Software
The Database Systems Environment
The Network Communications Environment
The Operating Systems Environment

Fig. 6: The Block diagram of operation

In other words, SSS is designed to operate as follows:

- It (SSS) resides on the Server quietly
- Its main task is to do the switching records from *data* to *url* mode and vice versa (*data motion*)

- Monitoring access history, calculating access parameters
- Adapting itself to new facts and statistics according to access history of incoming *SQL*s
- Predicting types of motion on clusters of data records such as *Brownian, periodic, systematic, or immigration*, & adapt DB accordingly
- Finding an optimal pattern for the SQL machine to access the parallel DB for administrative purposes without resorting to motion again *(i.e. separate client from administrator requests)*

7. Conclusion

Basic simulation has been done using randomly generated sequences of queries and requests and algorithmic design of a *Chebyscheff filter gave a model of software structure to base its design on.*

Currently work is going on this model. Future work will include programming of this SSS and testing it on a server-based small network in the lab.

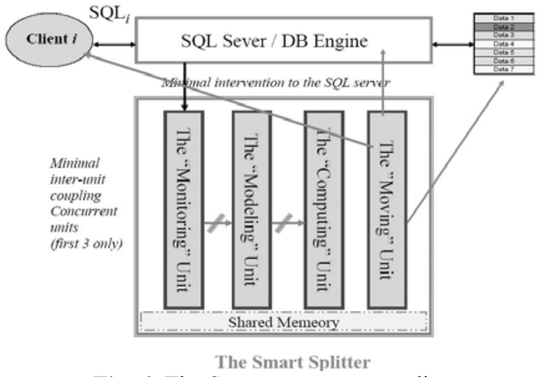

Fig. 6: The Systems component diagram

References

[1] Kaladhar Voruganti, M. Tamer Özsu, Ronald C. Unrau, Journal of Distributed and Parallel Databases: "An Adaptive Data-Shipping Architecture for Client Caching Data Management Systems", Vol. 15 (2): pp. 137-177, March 2004

[2] Fernanda Baião, Marta Mattoso, Gerson Zaverucha, Journal of Distributed and Parallel Databases: "A Distribution Design Methodology for Object DBMS", Vol. 16 (1): pp. 45-90, July 2004

[3] Lawrence Mutenda, Masaru Kitsuregawa, Parallel and Distributed Systems: "Parallel R-Tree Spatial Join for a Shared-Nothing Architecture", Institute of Industrial Science, Tokyo

[4] Jim Smith, Sandra Sampaio, Paul Watson, Norman W. Paton, ACM 0-89791—88-6/97/05: "Polar: An Architecture for a Parallel ODMG Compliant Object Database", CIKM' 2000 Washington D.C., USA

[5] V. V. Khodorovskii, Programming and Computer Software: "On Normalization of Relations in Relational Databases", Vo. 28 (1): pp. 41-52, January - February, 2002

[6] Ing-Ray Chen, Ding-Chau Wang, Chih-Ping Chu, Journal of Distributed and Parallel Databases: "Analyzing User-Perceived Dependability and Performance Characteristics of Voting Algorithms for Managing Replicated Data", Vol. 14 (3): pp. 199-219, November 2003

[7] Cyrus Shahabi, Yi-Shin Chen,Journal of Distributed and Parallel Databases: "An Adaptive Recommendation System without Explicit Acquisition of User Relevance Feedback", Vol. 14 (2): pp. 173-192, September 2003

[8] E.G. Hoel, Proceedings of the 23rd Intl. Conf. on Parallel Processing: "Data-Parallel Spatial Join Algorithms", pp. 227-234, 1994

[9] M. Olson, W. Hong, M. Ubell, M. Stonebraker, Data Engineering Bulletin: "Query Processing in a Parallel Object-Relational Databse System", vol. 19(4) pp. 3-10, 1996

[10] Nicolas Perrin, Bonnie Heck Ferri, "Digital Filters with Adaptive Length for Real-Time Applications", Georgia Institute of Technology, Georgia, USA

[11] Samir Khuller, Yoo-Ah Kim, Yung-Chun Wan, Society for Industrial and Applied Mathematics (SIAM): "Algorithms for Data Migration with Cloning", vol 33, No. 2 pp. 448-461, 2004.

[12] Peter Buneman, Sanjeev Khanna, Keishi Tajima, Wang-Chiew Tan, "ACM Transactions on Database Systems: "Archiving Scientific Data", vol. 29 Nr. 1 pp. 2-42, March 2004

[13] Fernando Pedone, Rachid Guerraoui, André Schiper, Journal of Distributed and Parallel Databases: "The Database State Machine Approach", Vol. 14 (1): pp. 71-98, July 2003

[14] Shashi Shekhar, Sivakumar Ravada, Vipin Kumar, Douglas Chubb, and Greg Turner, "Load-Balancing in High Performance GIS: Partitioning Polygonal Maps", publication of GIS database applications proceedings