NORMAL DISTRIBUTION AS A METHOD FOR DATA REPLICATION IN A PARALLEL DATA SERVER

By

Sanhita Sarkar

and

Shyam Sundar Sarkar

IMA Preprint Series # 1437

November 1996

INSTITUTE FOR MATHEMATICS AND ITS APPLICATIONS

UNIVERSITY OF MINNESOTA

514 Vincent Hall
206 Church Street S.E.
Minneapolis, Minnesota 55455

Normal Distribution as a Method for Data Replication in a Parallel Data Server

Sanhita Sarkar[1]
Institute for Mathematics and its Applications (IMA),
University of Minnesota,
514 Vincent Hall, 206 Church Street, S.E.,
Minneapolis, MN 55455
sarkar@ima.umn.edu

Shyam Sundar Sarkar
Centura Software Corporation,
1060 Marsh Road,
Menlo Park, CA 94025

Keywords.- Decision Support Systems, Parallel Databases, Simulation, High Performance Computing.

ABSTRACT

Run-time load balancing is always a problem in scalable parallel data servers because of initial data distribution. In practice, data distribution is expressed by users or database designers at the time of cluster definitions. A data warehouse is a scalable parallel server to operate on high volumes of data for analytical processing. Relational model implementations for analytical processing should incorporate very fast operations like join, scan, sort, etc. Currently, for shared-nothing MPP architecture, data is partitioned at load time in a shared-nothing way. The task distribution in a join, scan and sort process, is determined by the query optimizer in a relational engine and there is no possibility of load balancing at run-time because of the high network and I/O cost. However, for a large number of nodes in a MPP architecture, there may be a need for load sharing at run time, unlike the plan generated by the optimizer in relational engines. We have developed a theory for normal distribution of data from a node to its immediate neighbouring nodes. This normal distribution is implemented as a replication technique for small fractions of data from a node to its adjacent nodes at load time. Since a data warehouse is a read-only data service system, fractional replications can be distributed without worrying about audit or update. During run time, based on mutual agreements, task load can be transferred from a node to its adjacent nodes, for minimizing the overall time involved in relational operations. This normal distribution of fractional replicated data will lead to a unique probabilistic run-time solution for load balancing, not available in current parallel data servers.

1. INTRODUCTION

A data warehouse is a subject-oriented, integrated, time variant, nonvolatile collection of data in support of management's decision making process. Data warehouse brings together large volumes of business information obtained from operational systems and other data sources. The data is combined, transformed and stored in a consistent format. The data collected is retained over time so that changes and trends can be identified. Critical business functions like market planning and analyses provide an enterprise a better understanding of existing business and discovery of new opportunities. The single most important component of a data warehouse system is the Relational Database Management System (RDBMS) which stores the vast amount of information from which answers to critical business questions are derived. The major requirements of a data warehouse database server are: (i) scalability to handle tens to hundreds of gigabytes of data well, and yet capable of handling terabytes of data; (ii) fast query

[1] To whom correspondence should be addressed.

performance for interactive ad hoc queries as well as extremely complex queries that go through a large amount of detailed data, for example, computationally intensive aspects like joins, sorting and grouping are often part of the applications requiring complex queries; (iii) fast data load and updates; (iv) high availability of the warehouse for mission critical decision-support applications for users all over the world and last of all (v) data warehouse administrative capabilities to manage a large-scale relational database.

A parallel relational database with MPP (Massively Parallel Processing) architecture is referred to as a shared-nothing architecture consisting of nothing more than a fast network of independent uniprocessors or nodes. By connecting multiple nodes together through a high speed, point-to-point interconnected network, MPP architectures can scale indefinitely from a hardware perspective. This is because each node is an independent entity and is not bound by a common system bus or shared operating software. As a result, these architectures can support an unlimited number of processor nodes and deliver the performance necessary for running complex and analytical decision-support queries. This shared-nothing approach is used in managing data in current data servers to minimize operating system overhead and reduce network cost. To achieve this level of independence, each node runs its own instance of the database which consists of services for managing its own logging, recovery, locking and buffer management. This instance of the database is called a *co-server*. Each co-server owns a set of disks and the partitions of the database that reside on these disks (Figure 1). A co-server may have physical accessibility to other disks owned by other co-server(s) for failover purposes, but in normal operation, each co-server accesses only those disks that it owns.

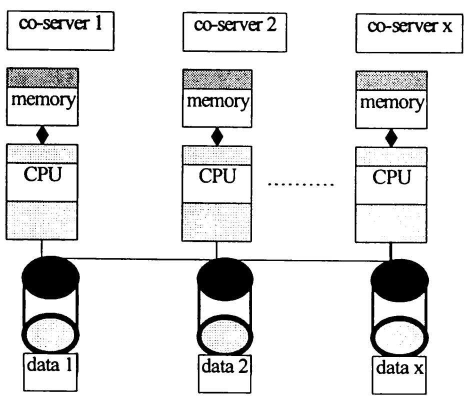

Figure 1: A typical shared-nothing parallel server architecture.

A query engine is designed based on *three* important aspects in order to maximize scale up and speed up. They include: optimal *partitioning of data*, *partitioning of control* and *partitioning of execution*. Data partitioning enables the physical division of a database to make it appear as if it is a group of small databases. Current parallel servers with a shared-nothing architecture maintains scalability by partition ownership. That is, a data partition is read and written *only* by the co-server that owns it. This is to minimize expensive network cost for any network traffic. The optimal data partitioning is done by the users and the database designers through cluster creation syntax. The partitioning of control as well as that of execution are carried out according to the optimal plan generated by a query optimizer. Each co-server may interact and coordinate activities with other co-servers in order to perform scan, join and sort tasks in order to speed up a query. This level of coordination is achieved by making intelligent decisions about how to divide the query and where to send the component operations to be performed on different nodes. These decisions include the (i) request manager which makes decisions about how a query should be divided and distributed while ensuring the workload is balanced across the nodes; (ii) query optimizer which determines the lowest-cost plan to perform a query; (iii) metadata manager which resides on each co-server and cooperate with others to provide accessibility to the metadata of all the databases stored in the system and the (iv) scheduler which distributes execution tasks by activating a plan such that the proper resources are locally available (Figure 2).

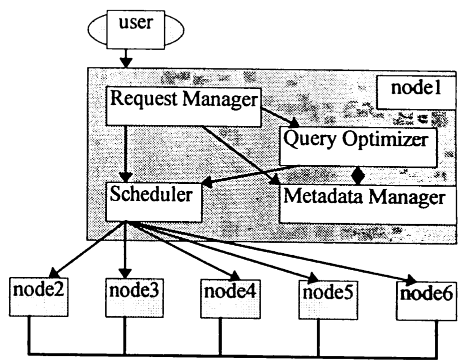

Figure 2: Components of a typical parallel server for optimal decision making at compile time.

It is important to mention at this point that the *decisions regarding the optimal data partitioning and those regarding task partitioning* as described above, are all done *at load time and compile time, respectively*. Run-time decision making regarding data partitioning and task partitioning is avoided by the current parallel data servers due to high cost one has to pay for the network and I/O.

As a solution to the unaffordability of dynamic load balancing, we have first developed a theory for normal distribution of data from a node to its neighbouring nodes as a replication technique of small fragments of data during *load time*. After we initially load the data to the processors, we decide on a data packet size and let each such packet of data be distributed to other processors for replication, using a Binormal distribution whose corresponding density function $g(x,y)$ with mean zero and standard deviation σ, is given by,

$$g(x,y) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{x^2+y^2}{2\sigma^2}\right\} \qquad [1.1]$$

The graph of such a density function is a bell-shaped curve that is symmetric around the origin (Figure 3).

We actually use Box-Muller transformation to generate two normal deviates from a normal distribution,

$g(y) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{y^2}{2\sigma^2}\right\}$ and consider transformation between two uniform deviates on (0,1), x_1, x_2 and two quantities y_1, y_2 as,

$$y_1 = \sqrt{-2\sigma \ln x_1}\cos(2\pi x_2),$$
$$y_2 = \sqrt{-2\sigma \ln x_1}\sin(2\pi x_2) \qquad [1.2].$$

Instead of picking uniform deviates x_1, x_2 in the unit square, we instead pick v_1, v_2 as the ordinate and abscissa of a random point inside the unit circle around the origin. Then the sum of their squares $R \equiv v_1^2 + v_2^2$ is a uniform deviate, which can be used for x_1, while the angle that (v_1, v_2) defines with respect to the v_1 axis can serve as the random angle $2\pi x_2$. Then the sine and cosine in [1.2] can be written as $\frac{v_1}{\sqrt{R}}$ and $\frac{v_2}{\sqrt{R}}$, obviating the trigonometric function calls.

We are using such a distribution function to ensure a uniform spread of the data in the x and y directions of 2-d network of processors (extensible to three and higher dimensions). The probability of a data packet to fall within a range along x and y axes, can be evaluated by obtaining the area under this density function between the range [Hiller and Lieberman, 1974]. We generate two uniform random numbers in the interval [0,1] from a normal distribution by [1.2] using a standard routine [William, 1994]. The generated pair of normally distributed independent random numbers, will give the random coordinates of the processor to which the data packet has been distributed for replication. In this way, we carry out several steps of Monte-Carlo simulation for each processor on packets of data, and draw a pair of random deviates from a Binormal distribution with mean zero and standard deviation (S.D) σ. This pair of random deviates as discussed above, determines the location of the replicated data packet in the network of processors. Here, σ is a processor-specific parameter.

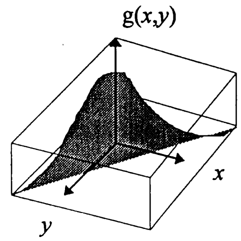

Figure 3: The normal distribution of data from a single node to the neighbouring nodes in the 2-d space for replication.

Example:

Let us consider an example of a data warehouse application using data from a national grocery retailer. The retailer wants to know more about the various buying patterns of customers and which products were commonly purchased at that time. A star schema was selected as the data model. We consider a part of the star schema to explain the data replication technique (Figure 4). We consider 3 tables: *Products* of size 83K, *Purchases* of size 170M and *Store* of size 900. There is a foreign-key relationship from *Products* to *Purchases* and also from *Store* to *Purchases*. For uniform distribution over 100 processors, say, in a 2-d network, we shall first distribute in each node, 830 *Products* tuples or records, the corresponding *Purchases* records carrying foreign keys from those 830 *Products* records, as well as records from the *Store* table whose keys are also inherited by those *Purchases* records. In this specific example, the *Store* table can be replicated at every node because it is small in size. Having done this initial distribution, we shall consider *Products* table as the primary table for normal distribution and will assume that the number of iterations of Monte-Carlo simulation, N say is 1000 and the data packet size, m in each iteration is 83. In each such packet, we shall consider 83 tuples of *Products* records, the corresponding tuples from *Purchases* table carrying foreign keys from the *Products* records selected. Thus, we have given a formal picture of an approach to deal with data distribution for replication.

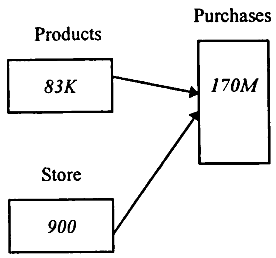

Figure 4: A part of a market analysis star schema for a grocery retailer

The theory we developed next is for *run-time* load balancing after the data has already been replicated at load time, following the above distribution rule. Each node maintains its local cost (sum of I/O CPU and communication cost due to messages) below a certain specified threshold so that a global cost function is minimized. At each time step, the nodes coordinate with each other asynchronously in a way to minimize this global cost function as the sum total of I/O, CPU and communication cost across all nodes at that time step. In case of an unbalanced task load by a node, a message may be sent asynchronously to a node containing its partial replicated data, so that this other node may take over its task only if its own cost fraction is less than a specified threshold. The above mentioned global cost as the sum of the local contributions from all nodes is thus minimized following a gradient descent method of optimization.

2. THE THEORY

2.1. Replication of Data by Normal Distribution

The equations in the ensuing paragraphs refer to a processor node in location (i,j), considering a 2-d network of nodes.

The amount of data as a variable at a particular node (i,j) at the initial *load time* is given by :

$$x(i,j) = x'(i,j) + \sum_{k,l} D(k,l) \qquad [2.1.1]$$

where, $x(i,j)$ is the total amount of data a node has after replication, $x'(i,j)$ is the initial amount of data loaded and $D(k,l)$ is the random input of data to the node (i,j) from all nodes for replication.

A constant m typifies the average size of a data packet. A data packet is a collection of data that "*migrate*" to a specific node for replication, following normal distribution. Thus, the total number of data packets present in a particular node (i,j) after initial load, can be written as:

$$N(i,j) = \frac{1}{m} x'(i,j) \qquad [2.1.2].$$

Repeating a Monte-Carlo simulation N times for each node (i,j), we draw a pair of random deviates from a Binormal distribution with mean zero and SD σ. The parameter σ is node-specific, and describes its characteristics related to the amount of data it holds. The pair of random deviates determine the location of each of the N packets of data in the 2-d network, and thus contributes to $D(k,l)$.

2.2. Run-Time Load Balancing

Run-time load balancing being very expensive in current parallel data servers, is avoided. However, the replication technique described above can solve this problem. After data loading is over, a node has its initial data as well as portions of replicated data from other nodes. At run-time, based on agreements between the nodes, task load may be transferred in a way so that the global cost of the whole relational operation at a particular time step is minimized. The global cost here, is the sum total of the local contributions to the I/O, CPU and communication costs, from all the nodes in the network. Each node minimizes its local cost function (function of local I/O, CPU and communication costs) by keeping the task load below a certain specified threshold. It also takes into consideration the cost it incurred in the previous time step, in making this decision. Thus, with the cooperation of all the nodes in the network, the global cost function follows a gradient descent towards minimization [Smolensky and Riley, 1984].

Whenever a node faces a situation of load imbalance, it *asynchronously* sends a message to other nodes. All nodes check for any pending messages for receipt. The node having *the* portion of the replicated data the sender needs to operate on, will check the contents of the message for the satisfaction of all the prescribed hypotheses regarding the message size, the processor-id, etc. It will also check on the task load fraction it will have to operate on, against a threshold. If less than the threshold, it will agree to operate on the replicated data and will send an asynchronous message back regarding such a mutual agreement. This mutual agreement follows all the conditions for the local minimization of cost tending to the global minimization, as discussed earlier.

2.2.1. The Global Cost Function

In this section, we provide a complete mathematical formulation for the dynamic load balancing situation and how such a situation is handled with an optimal cost.

Dynamics are defined as changes over time. Since the total number of computations in the data partitions can be discretized and this number changes with load transfer between the nodes, we describe a discrete, dynamical model for the local and global cost incurred in the network at a particular time step.

Example:

Let us first explain our approach with an example, where a join needs to be taken between two tables T_1 and T_2. Let each node out of say, 100 nodes initially contain 100 tuples of T_1 and 10,000 tuples of T_2. The total computation involved in this join, may be discretized into 25 time steps each involving tasks of joining 4 tuples of T_1 with T_2. Each task is done in a step. We call it a discrete computational step. Assuming it is a nested loop join, a node at each computational step, may do a cost analysis over its previous computations and decide to perform a join for 2 tuples of T_1 and delegate 2 other join computations to the neighbouring nodes by mutual agreements. This decision is made in a way such that the overall cost of the join computation decreases with time. In the following paragraphs, we will present a formal framework for such decision making.

The probability of the node (i,j) to become active is proportional to the total cost it contributes to the network [Hopfield, 1982]. By saying that a node is active, we mean that the node is actively taking part in load sharing. A node will probabilistically decide on task load sharing, by comparing the estimated cost at the current step, with the cost, it incurred in the previous time step. In the following lines, we describe how such a decision is made. By assumption, the cost function is additive under network decomposition. In such a network what is required of the probability assigned to a state of the network, is nothing but the product of the probabilities assigned to the states of the component networks. Thus adding the costs of the components' states should correspond to multiplying the probabilities of the components' states. It is a mathematical fact that the only continuous functions f that map addition into multiplication, i.e., $f(x+y) = f(x)f(y)$ are the exponential functions $f(x) = a^x$ for some positive number a. Equivalently, these functions can be written $f(x) = e^x$.

In the present case, the proportionality between the probability of the node (i,j) to become active and the total cost (cost of decision making and communication + computational cost) it contributes to the network at time t, can be written as,

$$\text{prob} \propto \exp(-cost) \qquad [2.2.1.1]$$

The negative sign indicates that lower the cost of its state in the network, higher is the probability of (i,j) to activate in sharing task load from other nodes.

Now, a node at (i,j) has an earlier estimated cost for every step in the computation. It will activate depending on the probability whether this estimated cost at the current step t is lower than the cost incurred by it at the previous step.

Every node (i,j) must thus compute the local likelihood ratio

$$\frac{\text{prob(to activate)}}{\text{prob(not to activate)}} = \exp\{\text{cost }(t-1) - \text{estimated cost }(t)\} \qquad [2.2.1.2].$$

If the exponential term is positive, i.e., if the previously estimated cost at time t is lower than the cost the node (i,j) incurred at time $(t-1)$ (taking into account the cost of decision making and computation, in the previous step), the probability for the node (i,j) to activate will be greater, i.e., the node (i,j) will more likely activate in response to the net input of the task loads.

Thus, at the cost of replication at static level, the task load from one node can be transferred to another node at run-time, on the basis of mutual agreements. This does not need true data migration at run-time, which is very expensive, as explained earlier. The communication cost will be very low as it deals with only short messages carrying information regarding the processor-id, status, data range and the type of computation involved. Thus, we get a true dynamic load balancing scenario at an optimal cost with some extra cost involved for initial data replication.

In the following paragraphs, we develop the theory for the local and global cost functions. The local cost contributed by a node (i,j) is the sum of the cost due to agreements involving communication and the cost due to computation on the data. The global cost is the sum of the contributions from all the nodes (i,j).

Let us consider a time step t, when a load balancing situation arises. The nodes with unbalanced task load send messages in the form of vectors whose elements contain all information regarding its processor-id; its current status as whether it may be a receiver or sender of an extra task load; the exact range of data to work on; a small query execution plan for the computations to be done on the data; etc. These vectors are mapped onto a so-called mathematically termed observation space. An observation space is a N-dimensional vector space where any N-dimensional vector if mapped onto, is considered to be point. Let $r_0(t), r_1(t), r_2(t), \dots, r_{k-1}(t)$ be the k message vectors that are being mapped onto the observation space at time step t. These vectors may be represented as a block message vector $\mathbf{r}$, for our future notation.

Thus,

$\mathbf{r}(t) = [r_0(t), r_1(t), r_2(t), \dots, r_{k-1}(t)]'$, is the block message vector. Each node (i,j) in a 2-d network of nodes, is assigned a block feature vector $\mathbf{f}(i,j)$ whose vector elements convey information regarding its own processor-id, what range of data it has, what computations it is supposed to do, its current status, etc. The list of all task loads $\{q_p(i,j, r_p(t),t)\}$, messaged to (i,j) via all possible $r_p(t)$, $p = 0,\dots,(k-1)$, comprises the input vector $\mathbf{q}(i,j,\mathbf{r}(t),t)$.

A decision rule is applied by the node (i,j), by associating its own features with the elements of the incoming message vectors $r_0(t), r_1(t), r_2(t), \dots, r_{k-1}(t)$. Each $f_p(i,j)$ is compared with each of the message vectors $r_p(t)$, $p = 0,\dots,(k-1)$. It checks whether the incoming message in each of the message vector matches the features in its own feature vector or not regarding send or receive of task loads. It labels 1,-1 or 0 to the association of its feature vector with the message vectors. "0" implies no match at all, "1" implies that there is a match and a task load is possibly to be received from the source associated with the message vector. Similarly, "-1" implies that there is a possibility of sending a task load to the receiver specified by the message vector.

Thus, for $p = 0,\dots,(k-1)$,

$$label(f_p(i,j), r_p(t)) = 0,\ 1,\ \text{or } -1.$$

An agreement occurs between a node (i,j) and another node in the network, if $f_p(i,j)$ matches any $r_p(t)$, $p = 0,\dots,(k-1)$. The node with which this agreement takes place, is the one specified in $r_p(t)$ and the communication for this agreement involves a message of size m, say. It is denoted by $m(i,j,r_p(t),t)$. *It is important to mention here, that the communications will only deal with messages regarding the task load transfer and no true data transfer will actually take place.* These communications are random, depending on the run-time situation of the system. If there are several iterations l of such random communications between different nodes in load balancing situations at time step t, then the true state of the whole system at time t is determined by the net effect of all the activities taking place in all the l steps at that particular time t. The number of times l a node will come across such a load sharing scenario and the random message vectors r_p with load sharing information, will all be consistent with the initial Binormal distribution of data. Thus, we may say that the run-time sharing of tasks will follow the trend of the Binormal distribution.

Let us denote the state of the network by the pair $(\mathbf{r}(t),t)$ consisting of the incoming block message vector $\mathbf{r}$ mapped onto the mathematically so-called observation space, and the time t. The cost function assigns a real number to each state. The cost function has as parameters the set of the feature vectors $\{\mathbf{f}(i,j)\}$ associated with each (i,j) and we call this the feature base $\mathbf{F}$.

The cost function considered should be additive under the decomposition of the system, i.e., if the network is partitioned into two unconnected networks, the total cost of the network will be the sum of the costs of the two subnetworks.

Let the global cost function as a contribution from all (i,j) at time t, be given by

$$\begin{aligned} C_{\mathbf{F}}(\mathbf{r}(t),t) = \sum_{i,j} \{ & C_v(\mathbf{r}(t), \mathbf{f}(i,j), \mathbf{m}(i,j,\mathbf{r}(t),t)) \\ & + C'(\mathbf{r}(t), \mathbf{f}(i,j), \mathbf{q}(i,j,\mathbf{r}(t),t))\} \end{aligned} \qquad [2.2.1.3]$$

where $C_v(\mathbf{r}(t), \mathbf{f}(i,j), \mathbf{m}(i,j,\mathbf{r}(t),t))$ is the cost contributed by the node (i,j) due to agreements with adjacent nodes through communication of messages of size m. $C'(\mathbf{r}(t), \mathbf{f}(i,j), \mathbf{q}(i,j,\mathbf{r}(t)$ is the computational cost on the changing task load, during load balancing.

2.2.2. *Calculation of the Cost due to Agreements through Communication:*

Now the total contribution to the communication cost by (i,j) due to agreements is given by,

$$C_v(\mathbf{r}(t), \mathbf{f}(i,j), \mathbf{m}(i,j,\mathbf{r}(t),t))$$

$$= \alpha \sum_{l} \sum_{p=0}^{k-1} \left| label(r_p(t), f_p(i,j)) \right| m_p(i,j,r_p(t),t)) \quad [2.2.2.1]$$

where l is the total number of iterations of a set of randomized communication operations, m is the message size communicated during every task load transfer and α is the constant of proportionality. The "mod" value implies that cost will always be additive, whether the label is 1 or -1 for a receive or send respectively.

2.2.3. *Calculation of the Computational Cost due to Agreements:*

The *net* input of task load to (i,j) is given by

$$I_v(\mathbf{r}(t), \mathbf{f}(i,j), \mathbf{q}(i,j,\mathbf{r}(t),t))$$

$$= \sum_{l} \sum_{p=0}^{k-1} label(r_p(t), f_p(i,j)) q_p(i,j,r_p(t),t) \quad [2.2.3.1]$$

due to agreements of $r_p(t)$ and $(f(i,j))_p$ on transferring the critical fraction of task loads, $p = 0,\ldots,(k\text{-}1)$.

Now, the *initial* task load on a node (i,j) was proportional to $x'(i,j)$ excluding the replicated data on which it operates, if necessary. The task load at time t is proportional to the data to work on and is given by,

$$y(i,j) = \frac{x'(i,j)}{T} \quad [2.2.3.2]$$

where, T is the total number of time steps to complete the relational operation. Task loads are added and subtracted from it on the basis of agreements. The net input of task load at time step t, on basis of agreements is $I_v(\mathbf{r}(t), \mathbf{f}(i,j), \mathbf{q}(i,j,\mathbf{r}(t),t))$. Thus the amount of data it operates on, at time t is given by:

$$y(i,j) + I_v(\mathbf{r}(t), \mathbf{f}(i,j), \mathbf{q}(i,j,\mathbf{r}(t),t))$$

The computational cost $C'(\mathbf{r}(t), \mathbf{f}(i,j), \mathbf{q}(i,j,\mathbf{r}(t)$ is proportional to the amount of the data operated on. Let this constant of proportionality be β.

Thus, we may write,

$$C'(\mathbf{r}(t), \mathbf{f}(i,j), \mathbf{q}(i,j,\mathbf{r}(t)$$

$$= \beta \{ y(i,j) + I_v(\mathbf{r}(t), \mathbf{f}(i,j), \mathbf{q}(i,j,\mathbf{r}(t),t)) \} \quad [2.2.3.3]$$

Since the term $I_v(\cdot)$ above, can be positive, negative or zero, the computational cost at each node at each time step may vary according to the necessity of task load balancing.

The cost terms in [2.2.2.1] and [2.2.3.3] from all (i,j) contribute to the global cost $C_F(\mathbf{r}(t),t)$ in [2.2.1.3]. The global cost will be minimized if the total contribution to the communication and computation cost from each active (i,j) is minimized.

2.2.4. *The Threshold for Decision Making*

For each iteration of a set of l randomized communications, a node calculates the fraction of the total weighted cost incurred with respect to the previously estimated cost for each such computational step. Let this cost be denoted by E. This cost is incurred on the basis of *agreements* of $f_p(i,j)$ with $r_p(t)$, as discussed earlier. A node (i,j) thus calculates

wtd-cost-fraction

$$= \frac{\left| \sum_{p=0}^{k-1} label(r_p(t), f_p(i,j)) q_p(i,j,r_p(t),t) \right|}{estimated\ cost} \quad [2.2.4.1]$$

We use the "mod" value of the sum of all weighted agreements, to ensure that the cost fraction is always positive. This fraction should be less than a threshold for minimization. So, we update the fraction as,

critical-fraction

$$= \nu - \frac{\left| \sum_{p=0}^{k-1} label(r_p(t), f_p(i,j)) q_p(i,j,r_p(t),t) \right|}{estimated\ cost} \quad [2.2.4.2]$$

where v is a threshold. Each node decides to receive or send based on agreements, the critical fraction of the task load [Sarkar, June 1996]. It orients its own feature vector f(*i,j*) with the information that it can send or receive the task loads.

We may express the best threshold range (Figure 5) in terms of the number of processors, *P*. The theory has been developed in the thesis [Sarkar, June 1996] and the best threshold range for an optimal dynamic load balancing situation, for a network of *P* processors should lie within the range,

$$1 > v > 1 - \frac{2}{P} \qquad [2.2.4.3].$$

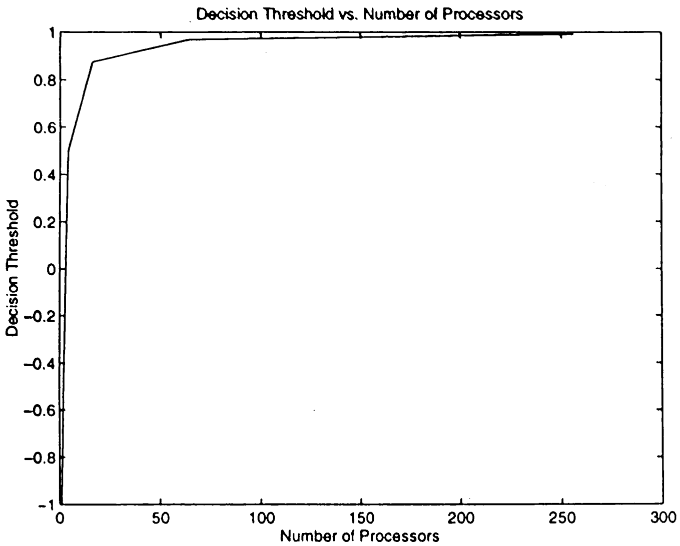

Figure 5: The best threshold as a function of the number of processors, *P*. as *P* increases, the threshold value limits to 1.

3. THE IMPLEMENTATION

The motivation of this section is to present some of the results of the simulation of a parallel data server using the data replication by normal distribution and dynamic load balancing in order to minimize the global cost. The implementation has been carried out on a parallel computer CM-5 (Connection Machine) which simulates a parallel data server. Conceptually, each processor simulates a co-server in a true parallel data server. The normal distribution as a replication technique as well as the asynchronous communication between the nodes for dynamic load balancing have been implemented on CM-5 using C++.

In our implementation, we are considering a range of data to lie in a 2-d space. This 2-d space has then been subdivided into small 2-d partitions and the computations in a particular data partition, are entirely local to that partition. We are now using a 2-d array of *P* processors consisting of $\sqrt{P}$ processors along one dimension and again $\sqrt{P}$ processors along the orthogonal dimension. Let *d0* and *d1* be the coordinates of a processor respectively in the X and Y dimensions. The value of *d0* is specified as CMMD_self_address()/ $\sqrt{P}$ and the value of *d1* is specified as CMMD_self_address() % *P*, where the call CMMD_self_address() returns the processor identifier for the processor passed as an argument. The data partitions are then evenly loaded onto these *P* processors arranged in the above mentioned checkerboard fashion.

A Monte-Carlo simulation of the normal distribution of several data packets is then carried out by each processor in order to replicate the data, it owns.

A computationally intensive program is then made to run in parallel with all processors working asynchronously on different partitions of data for several iterations of time steps. At times, when a processor's task load is such that its cost fraction rises above a certain threshold, it asynchronously sends a message containing the information regarding its processor-id, its status, the exact range of data it needs someone else to operate on, etc. Any processor containing that part of its replicated data will respond to the message and will agree or disagree to receive the task load at that instant. This decision is based on the truth of a few hypotheses and also on the fact that the incoming task load will not cause its local cost fraction exceed the threshold. On agreement, it will pursue the prescribed operations on the replicated data. Figure 7 shows that the best minima of the local costs along the 2-d space of processors are got using the threshold value within the range as described in equation [2.2.4.3]. In this way, the processors carry on with the computations on its local data partitions taking part in the load balancing situations when necessary and come out of the loop with a minimized global cost after all time steps are over (Figure 6). Tests were also made with increasing data partitions to simulate a Very Large Database (VLDB). We used up to 256 processors and were able to bring about a good speedup. The graph in Figure 8 ensures the scalability of our implementation.

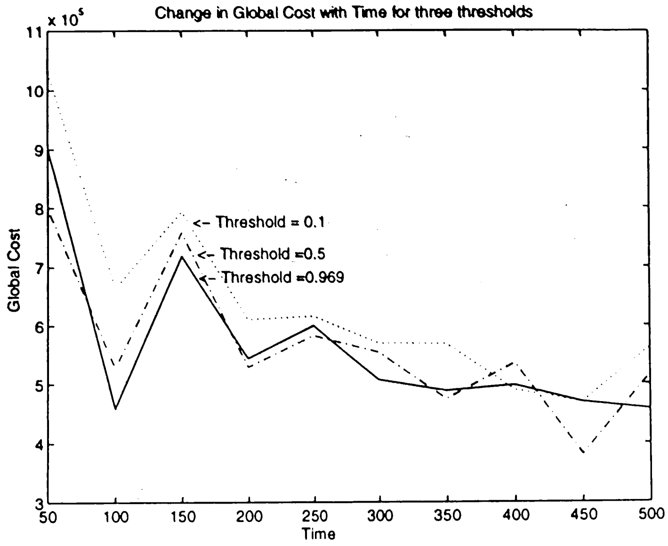

Figure 6: Change in the global cost with time for three different thresholds with $P = 64$, where P is the number of processors in the 2-d space.

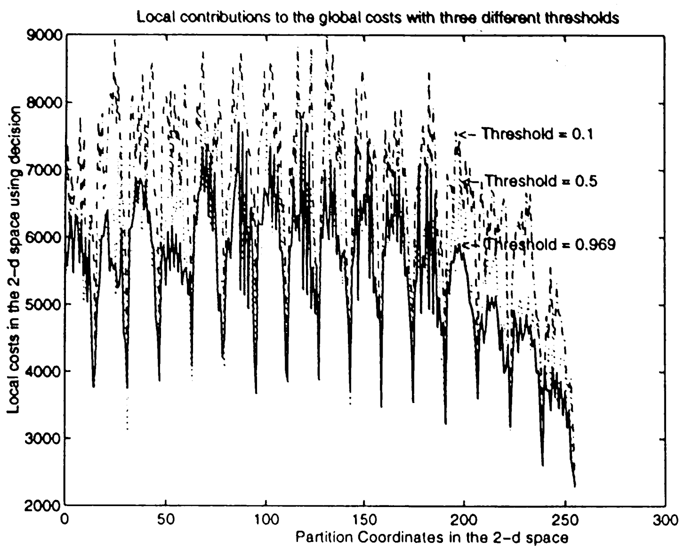

Figure 7: Local contributions (costs) along the 2-d space of processors for three different thresholds. Here, $P = 64$, where P is the number of processors in the 2-d space.

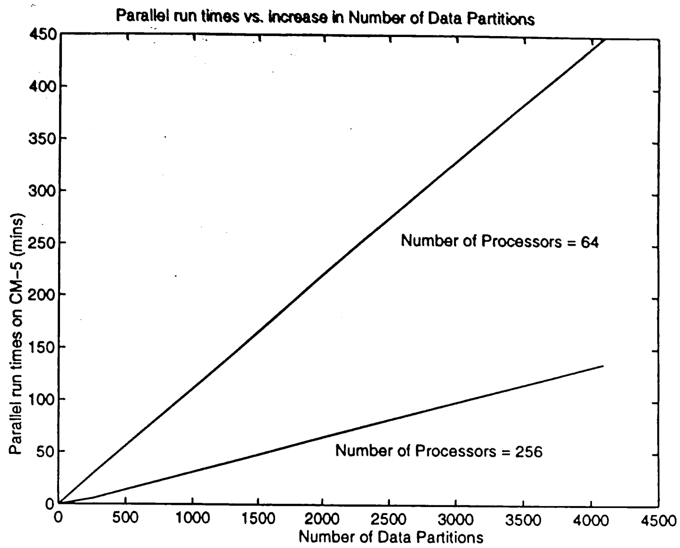

Figure 8: The change in the parallel run-times with the increase in the number of data partitions in the 2-d space.

4. CONCLUSIONS

Our main concentration in this research has been in finding out a better way to deal with dynamic load balancing issues, in a parallel data server. Since the current problem lies in the huge amount of data transaction involved for dynamic load balancing, we thought of a technique for data replication during load time, using normal distribution function. This will leave intact all the advantages of static data partitioning, like balanced I/O operations across all nodes and disks, accomodation for VLDBs, etc. In addition, it will be able to speed up the query operations by a cost optimal, decision-based, dynamic load balancing mechanism in a scalable manner. We have simulated a parallel data server here, only to get an initial idea of the possible results. As a future direction, we are also looking into distributed OLTP (On Line Transaction Processing) databases, where updates and audit trail synchronization will be a considerable issue. We believe that run-time load balancing as opposed to compile-time plan generation for executions, can work well for both OLTP (On Line Transaction Processing) and OLAP (On Line Analytical Processing) parallel data servers.

5. ACKNOWLEDGEMENTS

Minnesota Supercomputer Institute, for the machine time allocations on CM-5 in the primary stage;

Army High Performance Computing Research Center, Minnesota, for funding the CM-5 hours needed for the project for the rest of the period;

Institute for Mathematics and its Applications (IMA), University of Minnesota, for supporting this research with all their resources while hosting the High Performance Computing Program 1996-1997.

6. REFERENCES

Box, George E.P., and George C. Tiao. 1973. Bayesian Inference in Statistical Analysis. Philippines: Addison-Wesley Publishing Company, Inc.

Chacko, George K. 1971. Applied Statistics in Decision Making. New York: American Elsevier Publishing Company, Inc.

Davenport, W.B., and W.L. Root. 1958. Random Signals and Noise. New York: McGraw-Hill Book Company.

Feller, W. 1950, 1957. An Introduction to Probability Theory and Its Applications. Volume 1. New York: John Wiley and Sons, Inc.

Fisher, R.A, 1922. On the Mathematical Foundations of Theoritical Statistics. Phil. Trans. Roy. Soc., London. 222. 309.

Hiller, Fredrick.S and G.J. Lieberman. 1974. Operations Research.: San Francisco, California: Holden-Day, Inc.

Hopfield, J.J. 1982. Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences, USA. 79. 2554-2558.

Kumar,V., A. Grama, A. Gupta, and G. Karypis. 1994. Introduction to Parallel Computing: Design and Analysis of Algorithms. California: Benjamin/Cummings Publishing Company, Inc.

McClelland, J.L., D.E. Rumelhart and the PDP Research Group. 1986. Parallel Distributed Processing. Volume 2. Cambridge: The MIT Press.

Neyman, J., and E.S. Pearson. 1933. On the Problem of the Most Efficient Tests of Statistical Hypotheses. Phil. Trans. Roy. Soc. London. A231. 289.

Rumelhart, D.E., J.L. McClelland and the PDP Research Group. 1988. Parallel Distributed Processing. Volume 1. Cambridge: The MIT Press.

Sarkar, S. June 1996. Parallel Implementation of a spatial-temporal control system with decision based communication. Ph.D thesis, University of Minnesota.

Sarkar, S., and S.S. Sarkar. 1996. Data Replication in a Read-Only Parallel Data Server for Dynamic Load Balancing. Conference in High Performance Computing, 1997, Atlanta, Georgia. Paper accepted.

Smolensky, P. 1983. Schema selection and stochastic inference in modular environments. Proceedings of the National Conference on Artificial Intelligence. AAAI-83. 109-113.

Smolensky, P. 1984. The mathematical role of self-consistency in parallel computation. Proceedings of the Sixth Annual Conference of the Cognitive Science Society.

Smolensky, P., and M.S. Riley. 1984. Harmony theory: Problem solving, parallel cognitive models, and thermal physics. Tech. Rep. No. 8404. La Jolla: University of California, San Diego, Institute for Cognitive Science.

Thinking Machines Corporation. August 1992. The Connection Machine System: CMMD Reference Manual.

Ullman, Jeffrey.D. 1982. Principles of Database Systems. Computer Science Press.

William, H. 1994. Numerical Recipes in C: the art of scientific computing. 1994. Cambridge University Press.

Recent IMA Preprints

#	Author/s	Title

1348 **A.V. Fursikov**, Certain optimal control problems for Navier-Stokes system with distributed control function

1349 **F. Gesztesy, R. Nowell & W. Pötz**, One-dimensional scattering theory for quantum systems with nontrivial spatial asymptotics

1350 **F. Gesztesy & H. Holden**, On trace formulas for Schrödinger-type operators

1351 **X. Chen**, Global asymptotic limit of solutions of the Cahn-Hilliard equation

1352 **X. Chen**, Lorenz equations, Part I: Existence and nonexistence of homoclinic orbits

1353 **X. Chen**, Lorenz equations Part II: "Randomly" rotated homoclinic orbits and chaotic trajectories

1354 **X. Chen**, Lorenz equations, Part III: Existence of hyperbolic sets

1355 **R. Abeyaratne, C. Chu & R.D. James**, Kinetics of materials with wiggly energies: Theory and application to the evolution of twinning microstructures in a Cu-Al-Ni shape memory alloy

1356 **C. Liu**, The Helmholtz equation on Lipschitz domains

1357 **G. Avalos & I. Lasiecka**, Exponential stability of a thermoelastic system without mechanical dissipation

1358 **R. Lipton**, Heat conduction in fine scale mixtures with interfacial contact resistance

1359 **V. Odisharia & J. Peradze**, Solvability of a nonlinear problem of Kirchhoff shell

1360 **P.J. Olver, G. Sapiro & A. Tannenbaum**, Affine invariant edge maps and active contours

1361 **R.D. James**, Hysteresis in phase transformations

1362 **A. Sei & W. Symes**, A note on consistency and adjointness for numerical schemes

1363 **A. Friedman & B. Hu**, Head-media interaction in magnetic recording

1364 **A. Friedman & J.J.L. Velázquez**, Time-dependent coating flows in a strip, part I: The linearized problem

1365 **X. Ren & M. Winter**, Young measures in a nonlocal phase transition problem

1366 **K. Bhattacharya & R.V. Kohn**, Elastic energy minimization and the recoverable strains of polycrystalline shape-memory materials

1367 **G.A. Chechkin**, Operator pencil and homogenization in the problem of vibration of fluid in a vessel with a fine net on the surface

1368 **M. Carme Calderer & B. Mukherjee**, On Poiseuille flow of liquid crystals

1369 **M.A. Pinsky & M.E. Taylor**, Pointwise Fourier inversion: A wave equation approach

1370 **D. Brandon & R.C. Rogers**, Order parameter models of elastic bars and precursor oscillations

1371 **H.A. Levine & B.D. Sleeman**, A system of reaction diffusion equations arising in the theory of reinforced random walks

1372 **B. Cockburn & P.-A. Gremaud**, A priori error estimates for numerical methods for scalar conservation laws. Part II: Flux-splitting monotone schemes on irregular Cartesian grids

1373 **B. Li & M. Luskin**, Finite element analysis of microstructure for the cubic to tetragonal transformation

1374 **M. Luskin**, On the computation of crystalline microstructure

1375 **J.P. Matos**, On gradient young measures supported on a point and a well

1376 **M. Nitsche**, Scaling properties of vortex ring formation at a circular tube opening

1377 **J.L. Bona & Y.A. Li**, Decay and analyticity of solitary waves

1378 **V. Isakov**, On uniqueness in a lateral cauchy problem with multiple characteristics

1379 **M.A. Kouritzin**, Averaging for fundamental solutions of parabolic equations

1380 **T. Aktosun, M. Klaus & C. van der Mee**, Integral equation methods for the inverse problem with discontinuous wavespeed

1381 **P. Morin & R.D. Spies**, Convergent spectral approximations for the thermomechanical processes in shape memory allows

1382 **D.N. Arnold & X. Liu**, Interior estimates for a low order finite element method for the Reissner-Mindlin plate model

1383 **D.N. Arnold & R.S. Falk**, Analysis of a linear-linear finite element for the Reissner-Mindlin plate model

1384 **D.N. Arnold, R.S. Falk & R. Winther**, Preconditioning in $H(\mathrm{div})$ and applications

1385 **M. Lavrentiev**, Nonlinear parabolic problems possessing solutions with unbounded gradients

1386 **O.P. Bruno & P. Laurence**, Existence of three-dimensional toroidal MHD equilibria with nonconstant pressure

1387 **O.P. Bruno, F. Reitich, & P.H. Leo**, The overall elastic energy of polycrystallin martensitic solids

1388 **M. Fila & H.A. Levine**, On critical exponents for a semilinear parabolic system coupled in an equation and a boundary condition

1389 **J.M. Berg, W.G. Frazier, A. Chaudhary, & S.S. Banda**, Optimal open-loop ram velocity profiles for isothermal forging: A variational approach

1390 **J.M. Berg & H.G. Kwatny**, Unfolding the zero structure of a linear control system

1391 **A. Sei**, High order finite-difference approximations of the wave equation with absorbing boundary conditions: A stability analysis

1392 **A.V. Coward & Y.Y. Renardy**, Small amplitude oscillatory forcing on two-layer plane channel flow

1393 **V.A. Pliss & G.R. Sell**, Approximation dynamics and the stability of invariant sets

1394 **J.G. Cao & P. Roblin**, A new computational model for heterojunction resonant tunneling diode
1395 **C. Liu**, Inverse obstacle problem: Local uniqueness for rougher obstacles and the identification of a ball
1396 **K.A. Pericak-Spector & S.J. Spector**, Dynamic cavitation with shocks in nonlinear elasticity
1397 **G. Avalos & I. Lasiecka**, Exponential stability of a thermoelastic system without mechanical dissipation II: The case of simply supported boundary conditions
1398 **B. Brighi & M. Chipot**, Approximation of infima in the calculus of variations
1399 **G. Avalos**, Concerning the well-posedness of a nonlinearly coupled semilinear wave and beam-like equation
1400 **R. Lipton**, Variational methods, bounds and size effects for composites with highly conducting interface
1401 **B.T. Hayes & P.G. LeFloch**, Non-classical shock waves in scalar conservation laws
1402 **K.T. Joseph & P.G. LeFloch**, Boundary layers in weak solutions to hyperbolic conservation laws
1403 **Y. Diao, C. Ernst, & E.J.J. Van Rensburg**, Energies of knots
1404 **Xiaofeng Ren**, Multi-layer local minimum solutions of the bistable equation in an infinite tube
1405 **Vlastimil Pták**, Krylov sequences and orthogonal polynomials
1406 **T. Aktosun, M. Klaus, & C. van der Mee**, Factorization of scattering matrices due to aprtitioning of potentials in one-dimensional Schrödinger-type equations
1407 **C.-S. Man & R. Paroni**, On the separation of stress-induced and texture-induced birefringence in acoustoelasticity
1408 **D.N. Arnold, R.S. Falk, & R. Winther**, Preconditioning discrete approximations of the Reissner-Mindlin plate model
1409 **M.A. Kouritzin**, On exact filters for continuous signals with discrete observations
1410 **R. Lipton**, The second Stekloff eigenvalue and energy dissipation inequalities for functionals with surface energy
1411 **R. Lipton**, The second Stekloff eigenvalue of an inclusion and new size effects for composites with imperfect interface
1412 **W. Littman & B. Liu**, The regularity and singularity of solutions of certain elliptic problems on polygonal domains
1413 **C.R. Collins**, Spurious oscillations are not fatal in computing microstructures
1414 **M.A. Horn**, Sharp trace regularity for the solutions of the equations of dynamic elasticity
1415 **A. Friedman, B. Hu & Y. Liu**, A boundary value problem for the Poisson equation with multi-scale oscillating boundary
1416 **P. Baumann, D. Phillips & Q. Tang**, Stable nucleation for the Ginzburg-Landau system with an applied magnetic field
1417 **J.M. Berg**, A strain profile for robust control of microstructure using dynamic recrystallization
1418 **P. Klouček**, Toward the computational modeling of nonequilibrium thermodynamics of the Martensitic transformations
1419 **S. Chawla & S.M. Lenhart**, Application of optimal control theory to in Situ bioremediation
1420 **B. Li & M. Luskin**, Nonconforming finite element approximation of crystalline microstructure
1421 **H. Kang & J.K. Seo**, Inverse conductivity problem with one measurement: Uniqueness of balls in $\mathbb{R}^3$
1422 **Avner Friedman & Robert Gulliver, Organizers**, Mathematical modeling for instructors, July 29 – August 16, 1996
1423 **G. Friesecke**, Pair correlations and exchange phenomena in the free electron gas
1424 **Y.A. Li & P.J. Olver**, Convergence of solitary-wave solutions in a perturbed Bi-Hamiltonian dynamical system I. Compactons and Peakons II. Complex Analytic Behavior III. Convergence to Non-Analytic Solutions
1425 **C. Huang**, On boundary regularity of vortex patches for 3D incompressible euler systems
1426 **C. Huang**, A free boundary problem with nonlinear jump and kinetics on the free boundaries
1427 **X. Chen, C. Huang & J. Zhao**, A nonlinear parabolic equation modeling surfactant diffusion
1428 **A. Friedman & B. Hu**, Optimal control of chemical vapor deposition reactor
1429 **A. Friedman & B. Hu**, A non-stationary multi-scale oscillating free boundary for the Laplace and heat equations
1430 **X. Chen**, Existence, uniqueness, and asymptotic stability of traveling waves in nonlocal evolution equations
1431 **J. Yong**, Finding adapted solutions of forward-backward stochastic differential equations – Methods of continuation
1432 **J. Yong**, Linear forward-backward stochastic differential equations
1433 **D.A. Dawson & M.A. Kouritzin**, Invariance principles for parabolic equations with random coefficients
1434 **R. Lipton**, Energy minimizing configurations for mixtures of two imperfectly bonded conductors
1435 **D.C. Dobson & F. Santosa**, Nondestructive evaluation of plates using Eddy current methods
1436 **W. Littman & B. Liu**, On the spectral properties and stabilization of acoustic flow
1437 **S. Sarkar & S. Sundar Sarkar**, Normal distribution as a method for data replication in a parallel data server
1438 **S. Sarkar & S. Sundar Sarkar**, Parallel view materialization with dynamic load balancing: A graph theoretic approach
1439 **S. Sarkar & S. Sundar Sarkar**, Internet and relational databases in a multi-tier client/server model
1440 **J. Liang & S. Subramaniam**, Numerical computing of molecular electrostatics through boundary integral equations
1441 **J. Wu**, Inviscid limits and regularity estimates for the solutions of the 2-D dissipative quasi-geostrophic equations
1442 **P. Constantin & J. Wu**, Statistical solutions of the Navier-Stokes equations on the phase space of vorticity and the inviscid limits
1443 **M.A. Kouritzin**, Stochastic processes and perturbation problems defined by parabolic equations with a small parameter
1444 **M.A. Kouritzin**, Approximations for singularly perturbed parabolic equations of arbitrary order
1445 **A. Novick-Cohen** Triple junction motion for Allen-Cahn/Cahn-Hilliard systems