Network Capacity for Data Intensive Applications on Clusters of Workstations*

Guy Pardon **Gustavo Alonso**
Information and Communication Systems Research Group
Institute of Information Systems, Swiss Federal Institute of Technology (ETH)
ETH Zentrum, CH-8092 Zürich, Switzerland
{pardon,alonso}@inf.ethz.ch

February 19, 1998

Abstract

Component software, distribution, and the use of clusters of workstations are all key trends in today's technology. Little attention has been paid, however, to the network bandwidth required for data intensive applications. In the context of databases, much work has been done in parallelization strategies for monolithic architectures with dedicated, specialized networks or over disk arrays. We envision, however, data intensive applications over clusters of PCs or workstations connected by a commodity LAN. For such scenarios, this paper evaluates and analyzes the network requirements of a data processing application built from stand-alone components connected by a LAN. Using Amdahl's law in the context of a serial run of complex data-manipulating tasks (TPC-D like), we derive an estimate of the required network bandwidth as a function of the number of nodes involved, disk I/O bandwidth, amount of main memory vs. data volume size, and the nature of the applications. The analytical results are validated with experiments running the TPC-D benchmark over a prototype implementation of such high-level parallel data management system. The results prove the feasibility of the data intensive cluster approach and may serve as a guideline for other systems based on clusters of workstations and stand-alone components.

1 Introduction

Clusters of workstations are a pervasive and versatile computing platform. In most cases they are already in place and use commodity hardware and software, which makes them inexpensive and readily available [Gra95]. Clusters offer several advantages over centralized architectures, among these, and at least in theory, clusters have higher scalability and a greater degree of inherent parallelism. This idea is being pursued in numerous industrial and research projects [Bla96, BSS+95]. The two main constraints associated with clusters are that it may not be possible to use the network as efficiently as in a multi-processor or parallel machine, and that the participating nodes are not specifically designed for parallel optimization strategies [Duq97]. Moreover, several studies

*Part of this work has been funded by ETH Zürich within the DRAGON project(Reg-Nr 41-2642.5)

have concluded that there is significant overhead involved when using standard networks for cluster applications [vEBBV95, vEBB95, GBD+94]. Such results seem to point out significant difficulties in taking advantage of commercial off-the-shelf networks for clustered architectures. In view of this, the first question that arises when designing a cluster of data servers is whether a standard LAN can support the traffic generated by query execution. According to our own observations, the effective bandwidth for database applications is fairly low due to software overheads: 120 KB/s over an FDDI network with 100 Mb/s of raw capacity. This immediately poses the question of how to architect the system so as to provide sufficient performance gains in spite of the low bandwidth. The paper analyzes this question in detail. In particular, we express, in the context of a data intensive application, the bandwidth as a function of important system parameters such as disk I/O bandwidth, database size, or query (update) selectivity. This allows to estimate the required bandwidth as well as to understand under what conditions the cluster approach is a feasible solution. Contrary to the case of parallel cluster relational databases (so-called PRDBMS), for which quantitative studies have shown there are no communication bottlenecks [DG92, MD97], we expect this not to be the case in general clustered data architectures. In order to ascertain this point, we first investigate analytically the possible performance gain as a function of the disk bandwidth, the available network bandwidth at the application level, the number of nodes in the cluster, the amount of main memory versus disk space, and the nature of the application. We then validate the analytical study by performing several measurements over a prototype implementation. The results obtained show the feasibility and current limitations of clusters and, at the same time, offer helpful insights on the architectural details that must be taken into account in order to provide sufficient data handling capacity in a clustered architecture.

It must be pointed out that existing studies do not help to address these questions. For instance, [LD93] analyzes the TPC-C benchmark but TPC-C is meant for transaction processing and not so much for large data requests, and there is no explicit network involved in the analysis. Among the many studies on data placement [MR95, HF82, Ape88], sensitivity is often an unknown factor. Our work differs from previous studies because we specifically concentrate on the network and how the necessary bandwidth depends on other system parameters. This allows us to use our results for the assessment of performance on a cluster and thereby also present design guidelines. Moreover, we are also able to predict when it is advisable not to use a cluster for a particular application.

Among other contributions, our study shows that, if the application and the disk I/O bandwidth remain constant, the required network bandwidth is independent of the database size, an observation that can have a significant impact in terms of the scalability of the system. Moreover, we have been able to establish a precise relation between the network and disk I/O bandwidths. The overall efficiency of the system depends heavily on this relation and points at interesting evolution paths for cluster architectures. A third important issue uncovered by our study is the actual network bandwidth required for data intensive applications and how it is affected by software overheads. Based on these ideas, we provide some basic guidelines for network provisioning in a clustered architecture.

The rest of the paper is organized as follows: Section 2 introduces the system model and the notation. Section 3 describes the analytical study whereas Section 4 describes its implications. Validation with experimental data is presented in Section 5. Section 6 concludes the paper.

2 System Model and Notation

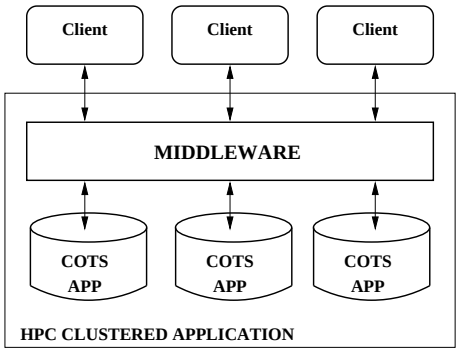

Figure 1: Architecture of high-level parallel data cluster

Throughout the rest of the paper we will consider a system similar to that shown in Figure 1. More precisely, we will use a set of Oracle7 servers interconnected through an FDDI network with each server containing a fragment of a TPC-D database ([Cou96]). The client application launches the requests as defined in the benchmark specification and the middleware component then divides the tasks into subtasks which are given to the appropriate COTS (Commercial Off-The-Shelf) servers. Both the analytical part and the experimental part are based on a set of complex requests that are executed serially, first on one machine and later on a parallel cluster system. We deliberately chose fairly simple fragmentation and execution optimization assumptions in order to be able to generalize the results. Similarly, we do not assume that the component systems offer anything more than a high level interface not necessarily designed for optimizing parallel queries or updates (the component servers may not have been developed with parallelism in mind). The notation used is shown in Table 1 for the known parameters (measured or given) and in Table 2 for derived values. In Table 1, the selectivity parameters indicate the amount of data associated with each query. The input selectivity is the fraction of the database read during the execution of a particular q_i. The output selectivity is the fraction of the input data read given as output (the filtering factor of q_i). The network selectivity is the fraction of the total database sent over the network during the execution of q_i (this is relevant for queries which are executed on one node but need data from other nodes).

We assume that every node has the same amount of central memory and the same fraction of the original database. Furthermore, we assume that the input and output selectivities on the different nodes are the same as for the centralized case. Also, disk and system characteristics such as page size and BLT are assumed to be the same on every node. As in other performance studies, we consider only queries. Updates do not generally result in high network bandwidth and are therefore omitted from this study. Queries are divided into two classes: P and NP. P is the set of requests (expressed as a fraction of the total number of requests) that can be executed in parallel with the only communication necessary being that involved in gathering the results. That is, no input data or intermediate data has to be interchanged. In practice, this is the case for simple selects or joins on tables which have been partitioned using hashing on primary key and foreign key. Throughout the paper, P will be used both as a set and a fractional value whenever appropriate. The P set is

q_i	request number i
L_i	output selectivity of q_i
L_i'	input selectivity for q_i
L_i''	network selectivity
S	total (centralized) database size in bytes
PS	individual size of a buffer page in bytes
BLT	time to replace one block in buffer
M	amount of buffer memory per node (in bytes)
N	number of nodes in cluster
P	fraction of requests whose parallel execution involves no communication
NP	the complementary set of P (P+NP=1)
T	average time for a run of Q different q_i in centralized case
T_i	centralized execution time for q_i
Q	number of distinct requests

Table 1: known parameters

processed on a cluster in the following way:

1. Each of the nodes executes its part of the task on its local data fragment of size S/N. This stage requires no communication by definition of P. Also, because the P set is inherently parallelizable, the total time for this step on a cluster is just the time needed by each node.

2. After the processing step, the results obtained by each node are gathered in one collecting node. This node might do some final ordering or grouping, but this step is obviated here (we have strong experimental support for making this assumption).

NP (not to be confused with Nondeterministic Polynomial) is the complementary fraction of the request set. The NP set represents the non-parallelizable requests, their execution is performed by fetching the necessary input data from N-1 nodes and doing the entire processing in a single node. For the NP set, processing on a cluster involves the following:

1. One node, the processing node, collects the data which is distributed equally over the N nodes. This step may significantly increase network traffic and, depending on the situation, it may or may not be possible to have the N nodes read their disks in parallel.

2. The processing of the data is done on one node only, and is assumed to take no longer than the corresponding one-node centralized implementation would take. Because the cluster nodes are assumed to be identical to the centralized node, this is a reasonable assumption.

To validate the analytical study, practical tests were conducted using the TPC-D benchmark on a centralized one-node configuration and later on a distributed three-node architecture implemented with the same simple optimization and fragmentation strategies. We chose TPC-D in order to have a realistically large example [Cou96].

BHR	buffer hit ratio
C	application-level network capacity
$Comm(q_i)$	communication overhead for the parallel execution of q_i
$DiskIOtime(q_i)$	disk I/O overhead for q_i
T'	total averaged time after parallelization
T'_i	parallel time for q_i
$NoPages_i$	number of pages used by q_i

Table 2: derived parameters

3 Analytical study of query performance

The use of a cluster can result in a gain or loss of performance. In here, we are not so much interested in the absolute gain but rather in under what conditions such gain exists and how it can be quantified. We compare a serial and centralized run of a set of queries with the same run over a clustered architecture. On the cluster, the individual tasks are executed in parallel whenever possible (intra-task parallelism), but the order of execution is still serial (i.e., a task is started only after the previous one has ended). The original data set is assumed to be equally distributed over the nodes. Each node has the same hardware configuration as the original centralized node. The rationale behind serial execution is to study the isolated behaviour of a task, much like traditional query optimization studies that do not consider the influence of concurrent queries. In practice, the concept of a serial run is similar to the TPC-D benchmark, which will be used later on in the evaluation. The outline of the rest of this section is as follows: we start with a general formulation of the relation between a (given) centralized execution time and the corresponding execution time in a distributed setup. In this relation we will identify two major contributions: Amdahl's law and an extra IO term, which is of interest to us, since it represents both the gain in disk transfer by using many disks in parallel (whenever possible) and the overhead of communication due to network transfers. The rest of the discussion will then focus on this IO term and how it can be quantified. With that knowledge, we can derive a reasonable lower limit for the network capacity.

3.1 Basic Analysis

For analysis purposes, following Amdahl's law [Amd67], the total time for the parallel computation is given by the time required to execute those queries which cannot be parallelized (the NP set) plus the time required to execute the parallelizable queries (the P set).

$$Time_{parallel} = Time_{serial} + Time_{parallelizable}/N$$

Intuitively, the gain will be quantified by the disk and network overhead. Centralized execution does not incur any network overhead; on the other hand parallel execution reduces the disk overhead while introducing the network cost. The aim is to arrive at an expression showing how these factors can be adequately balanced. More precisely, assuming that the cluster executes each of the Q queries q_i in time T'_i, then after X repetitions of the experiment (with X sufficiently large), we can expect the following average cluster execution time T':

$$T' = \frac{X/Q.\sum_{i \in P} T'_i + X/Q.\sum_{j \notin P} T'_j}{X} = 1/Q.\sum_{i \in P} T'_i + 1/Q.\sum_{j \notin P} T'_j \quad (1)$$

Note that we use the index "i" for the P set queries and the index "j" for the NP set. In general, every execution time T_k and corresponding T'_k (on the cluster platform) is composed of three components: the CPU time, the disk I/O time and the network I/O time (if relevant). This way, we can relate T'_k to T_k as shown below. For the P set, the relationship between the parallel execution time T'_i and the centralized time T_i is:

$$T'_i = \frac{T_i - oldDiskIOtime(q_i)}{N} + Comm(q_i) + newDiskIOtime(q_i)$$

The first term represents the CPU-time, which is divided by N in parallel execution. We assume the majority of the operations have a complexity linear in the input data size (therefore, a query that only does an index retrieval will not fit into this context). The second term is the extra time that is needed for the communication (which includes the time that the processor spends copying messages for the operating system) and the third term represents the disk I/O in a parallel environment. The definition of the P set includes parallel disk I/O, which means that $newDiskIOtime(q_i)$ is lower than $oldDiskIOtime(q_i)$ for every query in the P set.
For the NP queries, we find:

$$T'_j = T_j - oldDiskIOtime(q_j) + Comm(q_j) + newDiskIOtime(q_j)$$

The first two terms represent the CPU time of the reference case of centralized execution. As stated in our definition of the NP set, this processing time is assumed to be the same on the cluster. Depending on the situation, $newDiskIOtime(q_j)$ may or may not be smaller than $oldDiskIOtime(q_j)$. That is, if the data scan can be parallelized we will have a smaller value. Next we can substitute the expressions for T'_i and T'_j in (1). We obtain:

$$\begin{aligned} T' &= 1/Q.\sum_{i \in P}[\frac{T_i - oldDiskIOtime(q_i)}{N} + Comm(q_i) + newDiskIOtime(q_i)] \\ &\quad +1/Q.\sum_{j \notin P}[T_j - oldDiskIOtime(q_j) + Comm(q_j) + newDiskIOtime(q_j)] \\ &= 1/Q.\sum_{i \in P}\frac{T_i}{N} + 1/Q.\sum_{j \notin P} T_j + IOterm \end{aligned} \quad (2)$$

For simplicity and clarity of the next equation, we will assume that $\sum_{i \in P} T_i = P.\sum_{all} T_k$ and $\sum_{j \in NP} T_j = (1-P).\sum_{all} T_k$; to avoid this assumption we could have defined an extra factor as the fraction of time spent for executing P set queries, but then the expression would have been more complex without providing any additional information. Since in turn $\sum_{all} T_k$ equals $Q.T$ (by definition of the average time T), we can now relate the old (centralized) average execution time T to the new T':

$$T' = \frac{P.T}{N} + (1-P).T + IOterm \quad (3)$$

The first two terms express Amdahl's law, the last term is the I/O contribution. In the remaining part of this section, we will first derive a more detailed expression for $IOterm$ in terms of the different system characteristics (like the network bandwidth C and the disk transfer speed). Note that the goal is to minimize T'. Since we can not avoid the effects of Amdahl's law (which essentially implies that there is no linear speedup for $P < 1$), the only factor to modify is $IOterm$. If this term is positive, it will make the system's behaviour deviate even more from linear speedup. Thus, we will define the ideal network bandwidth as the lowest value of C for which $IOterm$ becomes negative.

3.2 Disk I/O overhead

A first contribution to $IOterm$ is the disk overhead, which can be calculated as follows. The number of pages that a query, q_i, needs during its execution in the centralized case is given by:

$$NoPages_i = \frac{amount\ of\ data\ for\ q_i}{pagesize} = \frac{L_i'.S}{PS}$$

This simply states that the number of pages needed is the number of bytes needed (expressed by the product of the input selectivity L_i' and database size S) divided by the number of bytes that makes up one page (PS). Next, let BHR_0 be the buffer hit ratio for the centralized case. The number of pages that q_i effectively causes to be replaced is then the number of pages needed (expressed by $L_i'.S/PS$) multiplied by the probability that a page is not found in the buffer ($1-BHR_0$):

$$\frac{(1 - BHR_0).L_i'.S}{PS}$$

from which it follows that the expected disk I/O time for q_i is given by:

$$oldDiskIOtime(q_i) = \frac{(1 - BHR_0).L_i'.S}{PS}.(BLT) \tag{4}$$

BLT represents the block load time: the time it takes to get the requested block (page) into the buffer so it can be accessed.

For the same query in the parallel cluster execution, the same derivation is possible, only now the database size per node is N times smaller and the buffer hit ratio may be different. If the query is from the P set, then parallelization is by definition easy and I/O is done in parallel too. Therefore, the overhead is just the time for each node to load the input data from the smaller local database (note that the P set requires remote accesses only for result shipping, for the rest of the execution the N nodes are independent in their execution):

$$newDiskIOtime(q_i) = \frac{(1 - BHR_1).L_i'.S/N}{PS}.(BLT) \tag{5}$$

In the alternate case of the NP set, the disk I/O can be either parallel or serial depending on the particular situation: it might be possible to do I/O in parallel whereas the processing itself has to be done serially (e.g., because of internode dependencies concerning intermediate results). In such cases, we also consider the task in question to be in the NP set. In general, we can write for the NP tasks:

$$newDiskIOtime(q_j) = \frac{(1 - BHR_1).L_j'.S/N}{PS}.N_{optional}.(BLT) \tag{6}$$

The factor $N_{optional}$ is present only in the case of serial I/O (the pessimistic case). For parallel disk I/O, the expression becomes the same as for the P set queries.

3.3 Network I/O overhead

A second factor in $IOterm$ is the network overhead. We only discuss the cluster case because there is no communication in the centralized execution. As a consequence, we always use the new database sizes S/N for each node. The selectivities are again assumed to be independent of the number of nodes involved. For queries in the P set, by definition only the results have to be sent over the network towards some gathering node. In this case, the size of the result data set on each node separately is $L'_i.L_i.S/N$ (each of N nodes is supposed to contain an equal fraction of the original S bytes of data and the selectivities are assumed to be the same on each node). To gather the results, N-1 nodes send their final data to one node, and hence the total amount of network overhead on the application level is given by:

$$Comm(q_i) = \frac{amount\ of\ data\ to\ be\ transmitted}{network\ capacity(=C)} = (N-1).\frac{S.L'_i.L_i}{N.C} \tag{7}$$

The processor overhead for copying messages is included in C, meaning that C is to be interpreted as the application-level network bandwidth.
For the NP queries q_j which cannot be easily executed in parallel on N nodes, but whose input data has to be transmitted entirely to one node we will use $L''_j.S/N$ as the amount of data fetched from each node (this is a straightforward consequence of the definition of L''_j and the fact that each node has an assumed database size of S/N). Note that we assume here that these queries are executed on one node only, by collecting the necessary input data on the other nodes and transferring it before processing. The reason not to use L'_j (input selectivity) is because the total number of pages read is probably larger than the number that is actually sent as pure input data: we may have local index pages that are read, but not sent over the network. Hence the special network factor for this case. Note that for queries in the P set, $L'' = L'.L$. From here:

$$Comm(q_j) = (N-1).\frac{S.L''_j}{N.C} \tag{8}$$

3.4 Parallelization cost

The value of $IOterm$ can be derived from expressions (4),(5),(6),(7) and (8). For notational simplicity, the product $L'_i.L_i$ for the P set is replaced by its conceptual equivalent L''_i (for the P set, only the output data is sent through the network).

$$
\begin{aligned}
IOterm \;=\; & \frac{(N-1).S}{N.C}.1/Q.\sum_{all} L''_k \\
& -\frac{(1-BHR_0).S}{PS}.(BLT)[1/Q.\sum_{P}\frac{L'_i}{N} + 1/Q.\sum_{NP} L'_j] \\
& +\alpha\frac{(1-BHR_1).S}{PS.N}.(BLT)[1/Q.\sum_{P} L'_i + 1/Q.\sum_{NP} N_{optional}.L'_j] \quad (9)
\end{aligned}
$$

The expression $1/Q.\sum_{all} L''_k$ is by definition the average L''. For simplicity, we will now assume that $\sum_{i\in P} L'_i = P.\sum_{all} L'_k$ and $\sum_{j\in NP} L'_j = (1-P).\sum_{all} L'_k$. By definition, $1/Q.\sum_{all} L'_k$ is the average L'. Thus, equation (9) becomes:

$$
\begin{aligned}
IOterm \;=\; & \frac{(N-1).L''.S}{N.C} \\
& -\frac{(1-BHR_0).L'.S}{PS}.(BLT)[(1-P)+P/N] \\
& +\alpha\frac{(1-BHR_1).L'.S}{PS}.(BLT)[P/N + N_{optional}.(1-P)/N] \quad (10)
\end{aligned}
$$

Once $IOterm$ has been calculated, T' can be properly characterized. A visual representation of T' as a function of N and C (with the other parameters having the values found in the experiments, see Section 5) is shown in Figure 2. As can be seen, for increasing values of C the overall time decreases quickly.

As an alternative, the NP queries might have a dominant L' (as we will observe in the experiments in Section 5). Then we can ignore the sums over P in equation (9). Furthermore, we then have that $1/Q.\sum_{NP} L'_j \approx 1/Q.\sum_{all} L'_j$ which is the average L'. Instead of having the previous equation, we can now say that:

$$
\begin{aligned}
IOterm \;=\; & \frac{(N-1).L''.S}{N.C} \\
& -\frac{(1-BHR_0).L'.S}{PS}.(BLT) \\
& +\alpha\frac{(1-BHR_1).L'.S}{N.PS}.(BLT).N_{optional} \quad (11)
\end{aligned}
$$

3.5 Lower bound for the network capacity

The (ideal, optimal) lower bound for the network capacity is defined as the lowest value of C that makes $IOterm$ in (10) negative. In order to determine the required network capacity, a more detailed expression for the BHR is needed. We will use the relation suggested in [TPK97] which gives BHR as a constant term "a" and a proportional contribution "b" of the logarithm of (buffer

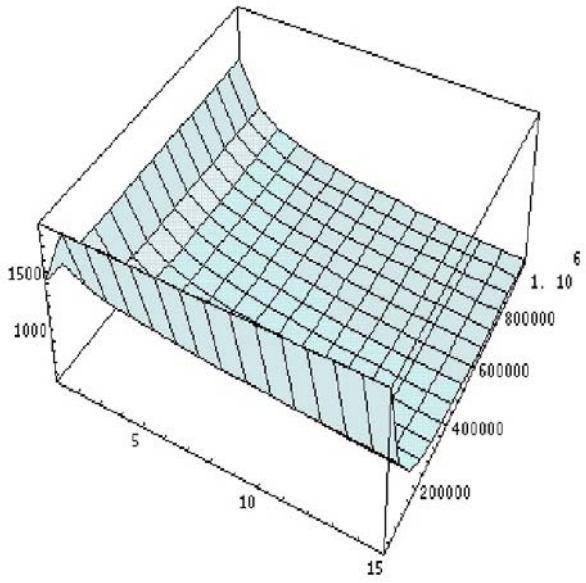

Figure 2: Theoretical prediction of T' (vertically, in seconds) as more nodes are added (front axis) and the network capacity C (in B/s) increases(right axis)

size/database size)[1] and assume pessimistically that the disk I/O for NP set queries has to be done serially:

$$BHR_0 = a + b.Ln[M/S]$$

$$BHR_1 = a + b.Ln[M.N/S]$$

For BHR_1, we have database size S/N, hence the appearance of N. With this, $IOterm$ becomes negative if:

$$C \geq \frac{N-1}{N} \cdot \frac{PS.L''}{L'.BLT.b.Ln[N]} [\frac{1}{P/N + 1 - P}] \quad (12)$$

4 On the network requirements for cluster architectures

From expression (12), a number of important conclusions can be derived. A crucial point to note about (12) is that, for a given application and the same number of nodes, it depends on roughly constant values. This is a very important point since it implies that given a high enough initial value of C, it will remain that way *independently of the database size.*

[1] Actually, the real relationship includes a small linear contribution of S in the range of valid values, but since it is limited to about 1% of the value of BHR, we will ignore it here.

4.1 Network capacity and disk I/O bandwidth

An additional important aspect of expression (12) is that the necessary network capacity is directly proportional to PS/BLT, which is roughly the average disk I/O bandwidth. Therefore, as disk speeds increase, the required network capacity must also increase. We consider this a very relevant factor pointed out by our study. In a cluster of workstations, the possible gain is determined by the disk I/O efficiency. With the same network but higher disk throughput, the cluster loses some of its attractivity. This can be intuitively explained in terms of where bottlenecks may arise. As each node in the cluster is able to pump more data into the network as the disk I/O bandwidth increases, the network capacity must increase to accommodate the resulting traffic. With a network capacity sufficiently high the system is bound only by the disk I/O bandwidth. Note that the result is independent of software overheads. The value calculated for C can be seen as the total effective bandwidth, independently of the raw capacity of the network. These calculations seem to show that the idea of a cluster parallel architecture becomes feasible as soon as the available network capacity becomes some multiple of the disk transfer speed, which intuitively corresponds to the situation where network traffic becomes more or less as fast as disk I/O [OV91], including some extra communication caused by the distributed setup.

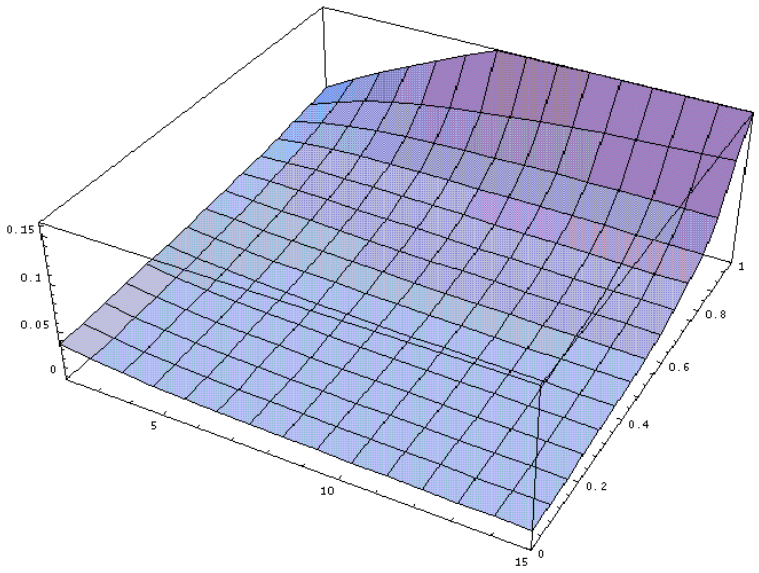

Figure 3: Theoretical value of factor F, as N varies (front axis) and P varies between 0 and 1 (right axis)

To explore this idea further, C can be expressed as a function of the disk bandwidth multiplied by a factor F that represents the other parameters:

$$C = F.PS/BLT$$

For a fixed disk speed, C varies with F. Figure 3 shows how F (and therefore also C) changes with P and N. The counterintuitive increase of F as P increases is caused by the lower value of the disk I/O terms in expression (10) for P queries. Since the value is lower, trying to compensate it with

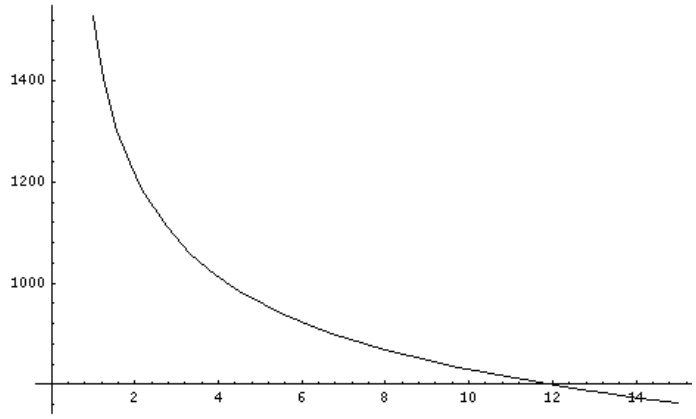

Figure 4: Theoretical prediction of T' (vertically, in seconds) as nodes are added (front axis) for C=120KB/s (as in our experiments)

the network cost will require a higher value of C than for the NP queries. Also note that the three overhead terms in (10) have a different physical meaning in the case of P and NP queries. For NP queries they are the pure difference in I/O, whereas for the P set they merely reflect that there is a difference without giving its exact value. This is also responsible for the paradoxical shape of the surface in Figure 3.

4.2 Absolute network capacity

For the concrete example of our TPCD benchmark query set (more information about the experiments is presented in the next section), the NP queries account for most of the average L', so we use expression 11 to derive the ideal C value, again as the lowest value of C that makes the last three terms negative. The formula now looks like:

$$C \geq \frac{N-1}{N} . \frac{PS.L''}{L'.BLT.b.Ln[N]} \qquad (13)$$

We use (13) to find a quantitative estimate of C. The values for the parameters are based on experimental findings concerning realistic configuration parameter values (Section 5). Thus P=8/17, with L" about 1/75, L' about 5. BLT was found to have a value of about 0,6 ms, the page size being 8KB. Furthermore, b is about 0.08, N equals 3 in our case, so the required capacity C as predicted by our analysis is:

$$C \geq 2,7.10^5 B/s \qquad (14)$$

Note that this is the optimal case: from this value on, the I/O term has a decreasing effect on the average execution time. This is an encouraging fact. For C being 120KB/s (which was the value obtained in our experiments), the time varies with N as shown in Figure 4.

Care should be taken when comparing this figure with our experiments, since as we will see, the assumption of equal distribution over the nodes will not entirely be satisfied there.
The impact of higher C depends on the nature of the application. In general, it can be said that increasing the value of C will always decrease the value of T'. The careful reader will probably have noticed that 0,27 MB/s has to be the pure data transmission bandwidth, meaning that the real channel has to have a higher capacity. In practice, the available bandwidth is much less than the nominal transmission rate of the medium due to different overheads (buffering, copies, protocols, etc.). Within the high performance computing community, it is generally accepted that communication mechanisms over standard networks, such as PVM message passing over TCP/IP, introduce unacceptable overheads with latencies as high as several milliseconds, much larger than in multiprocessor machines. This also seems to be the case for high speed networks: reports on the performance of BSD TCP/IP sockets on an ATM based workstation cluster show that only a small fraction of the raw network capacity is used due to software overhead[vEBBV95, vEBB95, GBD+94]. But even taking this into account, the resulting value seems to be reachable: 100 Mb/s FDDI is already at hand, Ethernet of that capacity as well and ATM networks promise even higher rates.

4.3 Alternative buffer management policies

To test whether the suggested model for the BHR can be applied in our case, we have also derived a bounding capacity value with the more naive assumption that the hit ratio is directly proportional to the ratio of the buffersize and the DBsize: $BHR = M/S$. Then, starting from (10) we obtained the following expression:

$$C \geq \frac{L''}{L'} . \frac{PS}{BLT} . \frac{S}{M} . \frac{1}{N.(1-P)+P} \qquad (15)$$

The first factor in (15) is smaller than one as argued in the definitions, the second one is the disk I/O bandwidth; the third one is the ratio of database size over buffer memory. We expect this last value to be about 10 since [TPK97] suggests 10 as a rule-of-thumb, [Gra95] foresees a ratio of exactly this value for the next decennium and our own platform had a similar value. Based on this and our previous values for the selectivities and the disk BW, we find that C must be at least (or rather optimally) 1MB/s which is of the same order of magnitude as before. Because the ratio S/M is probably going to be around 10 and remain so for at least some years as argued above, an initial satisfaction of the optimal C requirement can be expected to be sufficient for several years. Thus, the behaviour is the same as in the previous model. Note however that the observed effective bandwidth for database applications is rather low (120 KB/s) which seems to indicate that the software overhead may need to be significantly reduced.

4.4 Cluster design guidelines

Based on these ideas, we propose some general design guidelines for similar systems. Doing so, we assume that parameters such as selectivities are known or can be estimated. The heuristics can be formulated as follows:

1. Based on expression (3) we suggest to construct a fragmentation scheme that allows for a favorable (maximal) fraction P of parallelizable tasks, without worrying about bandwidth limitations. Note that it may be necessary to iterate in this step. For instance, a possibly maximal fraction P might be found by looking for queries (and updates) that are equivalent to the union of N different subqueries (-updates) ranging over N disjunct data sets. If the fragments on which the subtasks operate are allocated to different nodes, then the cluster can easily compute the results in parallel. Note that this may introduce high network cost for the remaining set of operations, a problem that can be addressed with an appropriate data placement strategy.

2. With the resulting fragmentation and the knowledge of the selectivities it is possible to make an estimate of the ideal value of C as given by the corresponding expressions in our analysis. If this value corresponds to what is at hand (taking the software overhead into account), then a 'good' architecture has been found. If not, continue with the next step.

3. If the necessary bandwidth is not available then there are several alternatives that can be considered. First, one could start with a different fragmentation scheme in the first step. Keeping the obviously limited communication infrastructure in mind, it might be wise to look for those allocations for which the input selectivity of resulting NP tasks is as low as possible. Alternatively, one could try to decrease the network selectivity factor (probably the only selectivity factor that can be changed easily). A way of doing so is using replication in order to avoid remote accesses ([RNS96]). In the extreme situation of total replication, all network traffic for NP input data gathering is eliminated. Care must be taken in the case of updates, though: if no suitable replication protocols are used then the deadlock rate might drastically increase. The use of a central co-ordinator for lock management is a possible solution, one that sacrifices fault tolerance for performance. The initial fragmentation chosen also has its influence on L''. Lastly, optimizations of the query execution can also reduce L''.

5 Validation of the analytical results

In order to validate the analytical results, a number of experiments have been performed based on the TPC-D benchmark over Oracle servers. The hardware platform is a cluster of Sun workstations running Solaris, with the interconnection network between the nodes being of the FDDI type (100 Mb/s). The validation consists of measuring the execution times and comparing them to the times that can be calculated with the analysis.

5.1 Experimental setting

The TPC-D benchmark has the schema represented in Figure 5. SF is meant to be a scaling factor with a minimal value of one (yielding a database size of 1GB). The numbers represent the number of rows in each table. The benchmark consists of 17 queries of the complex, decision support type (e.g. 'who are the customers who have ordered products with suppliers that are located in the same nation?'). Also defined are two update transactions, but those were not used in the experiments. The queries are referred to by their numbers [Cou96].

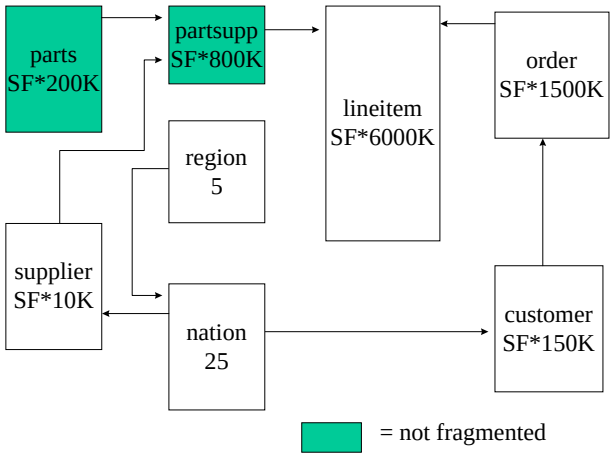

Figure 5: Fragmentation of TPC-D schema

Two sets of experiments were performed, one centralized and one distributed. In the centralized setting there was about 1GB of data (not including indexes) and a total database size of about 2,5GB. For the distributed experiment the database was fragmented over three nodes, resulting in a somewhat unbalanced distribution. The sizes after distribution were: 1,5GB on the heaviest node and 0,5GB on the others. The nation table was distributed equally over the nodes, associated with the corresponding tuples of the suppliers and customer table (by taking the corresponding foreign key values) and so on (by following the one-to-many directions of the relationships). With this fragmentation, we found that the TPC-D queries that fall into the P set are queries 1,3,4,5,6,10,12 and 13 (8 out of 17 queries). Two tables were not fragmented because they did not fit into this simple strategy. This does not significantly change the results but causes one node to be heavier than the others. In practice, the queries in the P set are such that the formulas we derived can be applied to them, in a slightly modified way to account for the size imbalance.

5.2 Experimental parameters

In the parallel experiment, the 17 TPC-D queries were implemented using PL/SQL scripts run in parallel in a shell environment. Results were only locally sorted (that is, on each node). The P and NP queries were executed as assumed in the analytical model. More precisely: for the NP set, a query is executed in one node after gathering all necessary data, and I/O is done serially.
The actual SQL-Net (Oracle's middleware) network capacity was measured using a test program and the diagnostic tools for Oracle. The value obtained=120 KB/s or about 50 percent of the lower bound determined in the analysis above. This low utilization value is consistent with results regarding the wasted bandwidth due to software overhead [HKL97].
The BLT is estimated using a test program (value obtained=0.6 ms to even 0.2 ms according to Oracle statistics. Note that we worked with queries only, so no block writes were needed on page replacement, furthermore we used striped disks). Oracle seems to be using a prefetch strategy for, e.g., sequential scans through a table, which of course leads to a low BLT as well.
For different buffer sizes, we have measured the BHR after the execution of a fixed set of queries (that is, with the database resulting in the same final buffer state by going through the same series

of intermediate states). These results are shown in Figure 6.
Since we measured the network I/O and the number of pages read directly, there is no need to determine the selectivities. In the experiments we were able to measure all page accesses and network traffic, so the factors with selectivities themselves are replaced by actual figures.

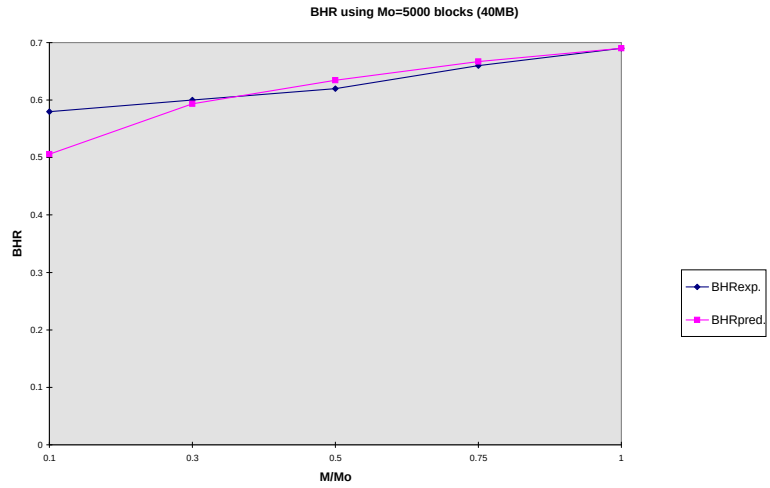

Figure 6: BHR predicted and measured for different memory sizes w.r.t. Mo=40MB

5.3 Execution times

Figure 7 compares the execution times of the queries of the TPC-D benchmark in the centralized case, the distributed system, and the analytical value calculated earlier. In comparing these results, it must be taken into account that some tables were not fragmented since they did not fit into that decomposition (we do not claim to have used the best fragmentation technique possible, the idea was just to investigate the possibilities of the general approach). Nevertheless, we observed good speedup for the parallelizable set and that the overall query mix can be executed faster in the parallel environment than in the centralized setup as shown in Figure 8. The actual speedup is less than three (our number of nodes) because the fragmentation we chose left one node with more data than the others. The actual TPC-D metrics were not that important to us, since we did not envisage a comparison with commercial solutions but rather a study of the behaviour of two configurations relative to each other.
The network capacity needed seems to be theoretically and practically available, and more or less stable over time (unless the disk I/O rate increases drastically which would probably mean that any parallel distributed approach will lose compared to a centralized implementation). As one can see from our results, the available capacity is not as high as desired, but still the gain of having the P set queries execute faster outweighs the slowing down of NP set queries. This is mainly due to the percentage of parallelizable tasks: had everything been inherently serial then, of course, it would have been worse, since we did not have the ideal bandwidth at our disposal. But for at

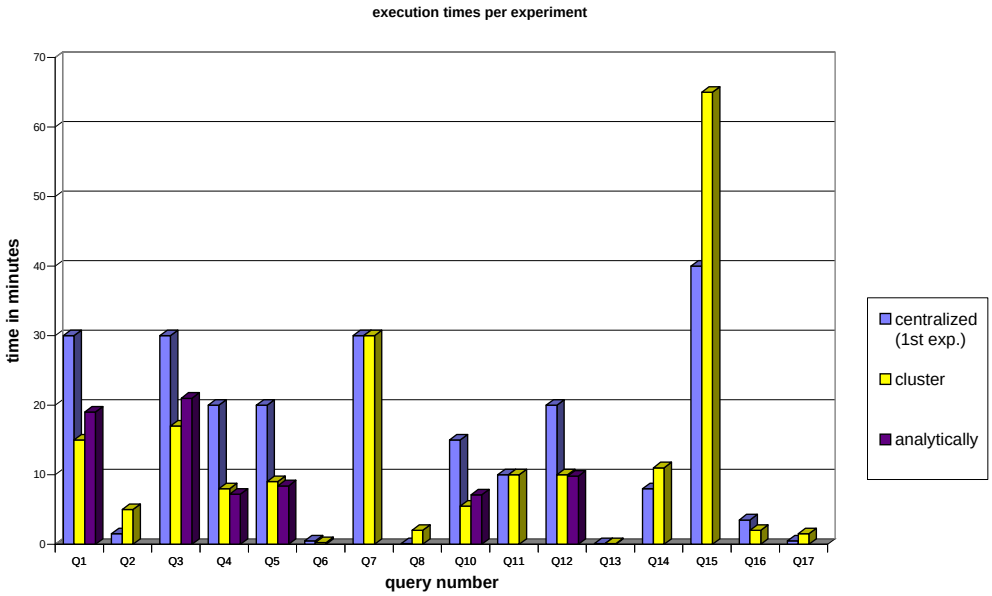

Figure 7: querytimes

least one concrete and practically relevant situation we have demonstrated that there is an inherent parallelizable part of approximately 50 percent.

5.4 NP set validation

For the NP set, we have run the entire set at once (individual validation measurements have been limited to the P set). The total number of bytes sent over the network, the actual network capacity and the total number of logical page reads give us the information we need for expressing the expected T'_{NP} using the results of the first series in Figure 7 (assuming the logarithmic relationship for BHR). Not shown is q_9 since it takes about 200 minutes to execute in all cases. For the NP set, the total time for execution in the centralized case was 290 minutes. In the distributed experiment, these same queries took a total time of 360 minutes. Additional tests have been performed with global sorting and more sophisticated optimization (e.g. keeping intermediate results in a local table if they are used later), which improves performance but does not add to the discussion in this paper.
The results for the NP-measurements are the following:

- For the NP set, the amount of data sent over the network is 592 MB.
- The total number of logical page reads (pages that are requested from the buffer whether they are already there or not) was 23.10^6 for these 9 queries. Of this, 9.10^6 were done at the 'heavier' node (the one that contains more data than the other two).

Taking into account the different database size at each node, we can estimate the difference in disk

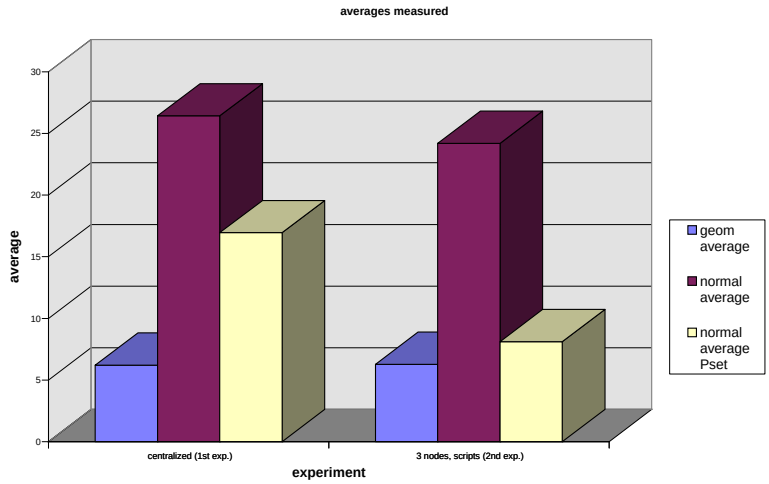

Figure 8: Average (total set) execution times per experiment

overhead for this entire set of queries (we do not have actual measurements of the BHR for the first setup, but we calculate one using the [TPK97]-like formula–in this we assume that the total number of pages that a query's execution needs in both the centralized and the distributed case is the same):
Let $Pages$ denote the intermediate factor representing the product of BLT and the total number of pages read for the NP set:

$$Pages = 23.10^6.0,6.10^{-3} = 13,8.10^3$$

Then the difference in disk I/O, Δ, taking into account that after distribution 9/23 of the total page reads is done at the heavy node and 7/23 at each other node, is:

$$\Delta = -(1 - BHR_0).Pages + (1 - BHR_{1@heavynode}).9/23.Pages$$

$$+(1 - BHR_{1@others}).7/23.Pages + (1 - BHR_{1@others}).7/23.Pages$$

Which can be reduced to (with M=4MB; S=2,5GB; b=0.08 and using the logarithmic expression for BHR):

$$\Delta = b.Pages.(\ln(\frac{4.10^6}{2,5.10^9}) - 9/23.\ln(\frac{4.10^6}{1,5.10^9}) - 14/23.\ln(\frac{4.10^6}{0,5.10^9})) = -1300sec = -22min$$

Therefore, the new time for the NP query set can be expected to be the sum of the times in the centralized case, plus the (negative) difference in disk overhead and plus the network overhead:

$$NPnewtime = 290minutes - 22minutes + \frac{592.10^6}{0,12.10^6.60}minutes = 350minutes$$

The value obtained experimentally was 360 minutes, fairly similar to the analytical result.

Differences can be accounted for by the difficulties in measuring the appropriate values accurately (e.g., the average time it takes to load one page) and the fact that distributed queries might yield different execution plans (which could also lead to different input selectivities). The network capacity used can be measured rather well, the block read time on the other hand depends largely on the file system cache. Since the queries only read, block replacement only involves reading the block again – an operation which can be reduced to maximally 1.5 ms per page (this is the time of NP query set execution in the one-node case divided by the total time for these queries). Experimentally, reading a 40 MB file in the file system varies from 81 seconds in the first execution (not cached) to 3 seconds in the later executions (cached version) yielding respectively 16 ms per page and 0.6 ms per page. The Oracle statistics observed seemed to be more like the last case.

5.5 P set validation

For the P set, we ran the queries that take a sufficiently long time to execute separately and used essentially the same kind of measurements as above, the only difference being that the reads and network transmission were now measured on a per-query basis. We have measured the queries number 1,3,4,5,10 and 12. The others are too short for a meaningful comparison. The results found are as follows:

query	*bytes sent*	*reads@node1*	*reads@2*	*reads@3*	*time(predicted)*	*time(experimental)*
1	80KB	60000	17500	18000	19	15
3	402KB	118500	26600	26500	21	17
4	100KB	120000	106000	107000	7,2	8
5	80KB	223000	146000	164000	8,4	9
10	4,4MB	99000	54000	55000	7,1	5
12	124KB	84000	44000	43000	9,8	10

The times in the last two columns are expressed in minutes. For all these queries we can calculate the difference in disk I/O in a similar way as before. However, with the numbers given, the difference is rather negligible. The same applies in the case of the network overhead (the difference is less than one minute). Therefore the result is mainly dependent on the time spent in processing, not on I/O. Note that the number of page reads on the 'heavy' node is significantly higher than on the other ones. Therefore, more processing will have to be done there and, consequently, it will take longer than the other nodes to finish (as indeed we see experimentally). The new execution time will be given by the old time on this node (which was measured in the centralized case) multiplied by the fraction of the total pages that have to be read here (the network and disk overhead differences are small as argued earlier). The results of these calculations are included in Figure 7.
We believe that the theoretical and experimental numbers are sufficiently close to each other to validate the expression obtained above. These data together with those for the NP query set allow for an estimate of the overall L': approximately 24 million pages (8KB each) or about 192 GB of data is read. Then, $L' = 1/17.\sum_{all} \frac{dataread}{databasesize} = \frac{1}{17.databasesize}.\sum_{all}(dataread) = \frac{192GB}{17.2{,}5GB} \approx 5$. In

a similar way, we find $L'' \approx 1/75$, which are the figures used for calculating the lower bound on the bandwidth in the previous section.

6 Conclusions and future work

The idea of using clusters of workstations as the basic hardware configuration for high performance computing is pervasive: many applications and systems are being developed exploiting the advantages of clusters using commercial, off-the-shelf components. Within our projects at ETH Zürich, we are interested in using clusters as the basis for developing an alternative form of parallel database. As in many other efforts, one of the main problems to be faced is the reduction in effective network bandwidth due to software overheads. Thus, a first step in the project has been to calculate the network bandwidth required to support data intensive applications. The paper has presented this study following an analytical approach later validated by experimental results acquired using the TPC-D benchmark. The formulation obtained for the required effective network bandwidth leads to several important conclusions. First, that the ideal bandwidth is stable, i.e., it can be expected to remain roughly the same irrespective of future data volume growth. Second, and based on the experimental data, an evaluation of the expression yields a required network bandwidth of 270 KB/s, which provides us with a good estimate of the specifications must meet. It is interesting to note that, as has already been pointed out in many other studies, the effective network bandwidth is significantly lower than the raw network capacity. This is a combination of the software overhead and, in our case, since a commodity LAN is being used, due to contention for the network resources (one of the design assumptions is that the underlying network should not be exclusively dedicated to the the system but shared with other applications). Third, the relationship between the required bandwidth and the disk transfer speed is studied in detail. If the disk I/O bandwidth increases, then the network has to be faster in order not to become a bottleneck. The experiments performed provide sufficient evidence to validate the analytical result. Moreover, the same experiments show a reasonable increase in performance for the fraction of the workload which was easily parallelizable. Finally, we outlined a design strategy that can be used to build similar systems. Future work involves further development of a prototype and a more detailed analysis of other application types with different network characteristics.

References

[Amd67] G.M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *AFIPS Conference, Volume 30*, 1967.

[Ape88] Peter Apers. Data allocation in distributed database systems. *ACM Transactions on Database Systems*, 13(3), 1988.

[Bla96] J.A. Blakeley. OLE DB: A component DBMS architecture. In *12th Intl Conference on Data Engineering*, 1996.

[BSS+95] Donald J. Becker, Thomas Sterling, Daniel Savarese, Bruce Fryxell, and Kevin Olson. Communication overhead for space science applications on the beowulf parallel workstation. In *High Performance and Distributed Computing 95*, 1995.

[Cou96] Transaction Processing Performance Council. TPC benchmark D standard specification. Technical report, Transaction Processing Performance Council, 1996.

[DG92] David Dewitt and Jim Gray. Parallel database systems: The future of high-performance database systems. *Communications of the ACM*, 35(6), 92.

[Duq97] Wayne Duquaine. Why PC servers won't overtake mainframe servers anytime soon. In *Seventh Intl. Workshop on High Performance Transaction Systems (HPTS)*, 1997.

[GBD+94] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V.Sunderam. *PVM: Parallel Virtual Machine. A User's Guide and Tutorial for Networked Parallel Computing.* MIT Press, 1994.

[Gra95] Jim Gray. A survey of parallel database techniques and systems. In *Tutorial Handouts of 21st VLDB*, 1995.

[HF82] L.W. Howdy and D.V. Foster. Comparative models of the file assignment problem. *ACM Computing Surveys*, 14(2), 1982.

[HKL97] Hermann Hellwagner, Wolfgang Karl, and Markus Leberecht. Enabling a PC cluster for high-performance computing. In *Proc. of 21st Speedup Workshop*, 1997.

[LD93] Scott T. Leutenegger and Daniel Dias. A modeling study of the TPC-C benchmark. *ACM SIGMOD*, 1993.

[MD97] Manish Mehta and David DeWitt. Data placement in shared-nothing parallel database systems. *The VLDB Journal*, 6(1), 97.

[MR95] Salvatore March and Sangkyu Rho. Allocating data and operations ot nodes in distributed database design. *IEEE Transactions on Knowledge and Data Engineering*, 7(2), 1995.

[OV91] M. Özsu and P. Valduriez. *Principles of Distributed Database Systems.* Prentice Hall, 1991.

[RNS96] Michael Rys, Moira Norrie, and Hans-Joerg Schek. Intra-transaction parallelism in the mapping of an object model to a relational multiprocessor system. In *VLDB96*, 1996.

[TPK97] Thin-Fong Tsuei, Allan Packer, and Keng-Tai Ko. Database buffer size investigation for OLTP workloads. In *Proceedings of ACM SIGMOD*, 1997.

[vEBB95] T. von Eicken, A. Basu, and V. Buch. Low latency communication over ATM networks using active messages. *IEEE Micro*, pages 46–53, February 1995.

[vEBBV95] T. von Eicken, A. Basu, V. Buch, and W. Vogels. U-net: A user-level network interface for parallel and distributed computing. In *Proc. 15th ACM Symposium on Operating System Principles*, 1995.

Appendix A: Sensitivity analysis: identification of the main parameters in the expression 10

In the experiments, a number of parameters have been measured and used in our predictions. We will of course have uncertainties on those values. We can get an idea of which parameters are most important by differentiating the expression for the new time found in the analytical section, and using that to calculate the sensitivities of T'. The sensitivity for a parameter x, denoted S(x), is defined here as the percentage change of T' due to a one percent error (change) for x, which can be calculated by multiplication of the partial derivative (for x) and one percent of x's value, and dividing this by T'.

Partial derivative for x	$S(x) = \frac{\partial T'}{T'.\partial x}.x$ in percent
$\frac{\partial T'}{\partial BHR_0} = \frac{L'.S.BLT}{PS}.[1 - P.(N-1)/N]$	$\approx 10^{-1}\%$
$\frac{\partial T'}{\partial BHR_1} = -\frac{\alpha.L'.S.BLT}{N.PS}.[P + N_{optional}.(1-P)]$	$\approx 10^{-1}\%$
$\frac{\partial T'}{\partial L'} = -\frac{S.BLT}{PS}.(1 - BHR_0).(1 - P + P/N) + \alpha.(1 - BHR_1)\frac{S.BLT}{N.PS}[P + N_{optional}.(1-P)]$	$\approx 10^{-1}\%$
$\frac{\partial T'}{\partial L''} = \frac{(1-P).(N-1).S}{N.C}$	$\approx 10^{-1}\%$
$\frac{\partial T'}{\partial BLT} = -(1 - BHR_0)\frac{L'.S}{PS}[(1-P) + P/N] + \alpha.(1 - BHR_1)\frac{L'.S}{N.PS}[N_{optional}.(1-P) + P]$	$\approx 1\ \%$
$\frac{\partial T'}{\partial C} = -\frac{(N-1).S.L''}{C^2.N}$	$\approx 10^{-1}\%$

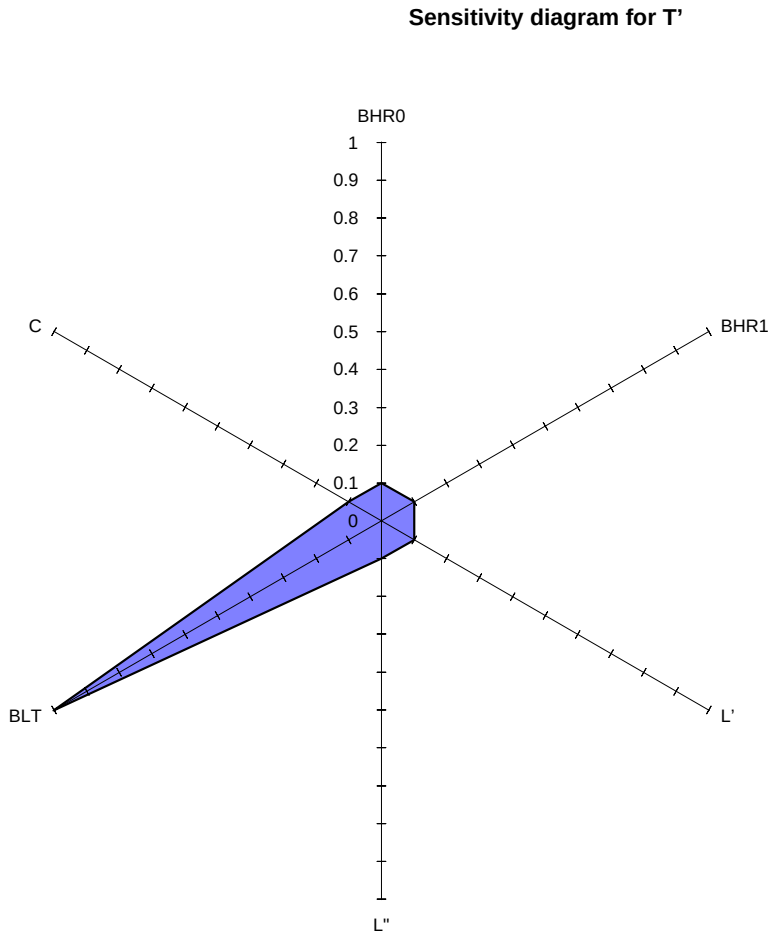

Figure 9: Sensitivity analysis of T'

Appendix B:Sensitivity analysis of expression 12

Similar to appendix A, we can calculate the selectivities for the optimal C:

$$C = \frac{N-1}{N}.\frac{L''}{L'}.\frac{PS}{BLT}.\frac{1}{(BHR_1 - BHR_0).(1 - P + P/N)}$$

Partial derivative for x	$S(x) = \frac{\partial C}{C.\partial x}.x$ in percent
$\frac{\partial C}{\partial N} = \frac{L''.PS}{L'.BLT.(BHR_1-BHR_0)[P+N.(1-P)]^2}$	$\approx 1\%$
$\frac{\partial C}{\partial L''} = \frac{N-1}{N} \cdot \frac{1}{L'} \cdot \frac{PS}{BLT} \cdot \frac{1}{(BHR_1-BHR_0).(1-P+P/N)}$	$\approx 1\%$
$\frac{\partial C}{\partial L'} = \frac{N-1}{N} \cdot \frac{L''}{L'} \cdot \frac{PS}{BLT} \cdot \frac{1}{(BHR_1-BHR_0).(1-P+P/N)} \cdot \frac{-1}{L'}$	$\approx 1\%$
$\frac{\partial C}{\partial BLT} = \frac{N-1}{N} \cdot \frac{L''}{L'} \cdot \frac{PS}{BLT} \cdot \frac{1}{(BHR_1-BHR_0).(1-P+P/N)} \cdot \frac{-1}{BLT}$	$\approx 1\%$
$\frac{\partial C}{\partial PS} = \frac{N-1}{N} \cdot \frac{L''}{L'} \cdot \frac{1}{BLT} \cdot \frac{1}{(BHR_1-BHR_0).(1-P+P/N)}$	$\approx 1\%$
$\frac{\partial C}{\partial BHR_1} = \frac{N-1}{N} \cdot \frac{L''}{L'} \cdot \frac{PS}{BLT} \cdot \frac{1}{(BHR_1-BHR_0).(1-P+P/N)} \cdot \frac{-1}{BHR_1-BHR_0}$	$\approx 10\%$
$\frac{\partial C}{\partial BHR_0} = \frac{N-1}{N} \cdot \frac{L''}{L'} \cdot \frac{PS}{BLT} \cdot \frac{1}{(BHR_1-BHR_0).(1-P+P/N)} \cdot \frac{1}{BHR_1-BHR_0}$	$\approx 10\%$
$\frac{\partial C}{\partial P} = \frac{N-1}{N} \cdot \frac{L''}{L'} \cdot \frac{PS}{BLT} \cdot \frac{1}{(BHR_1-BHR_0).(1-P+P/N)} \cdot \frac{N-1}{P+N.(1-P)}$	$\approx 1\%$

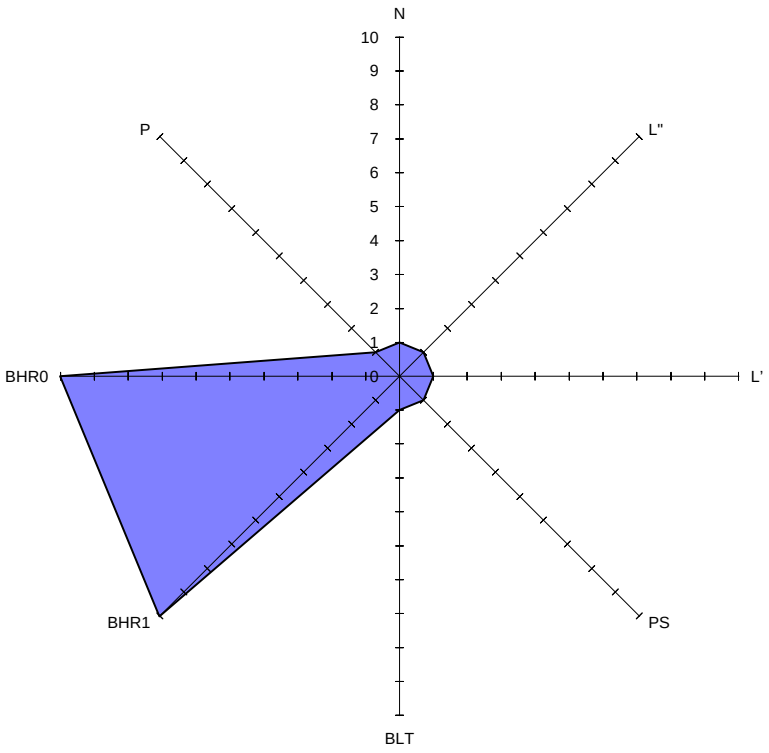

Figure 10: Sensitivity analysis of C

From this, it is clear that the most important parameters are the buffer hit ratios.