Hardware Capacity Evaluation in Shared-Nothing Data Warehouses

Ricardo Antunes
Department of Informatics Engineering
Faculty of Science and Technology
University of Coimbra
rantunes@dei.uc.pt

Pedro Furtado*
Department of Informatics Engineering
Faculty of Science and Technology
University of Coimbra
pnf@dei.uc.pt

Abstract

Parallel data warehouses have mostly been seen as dedicated decision support systems, but as the need for cost effective multi-application business solutions grows, non-dedication and adaptation to existing environments become evermore tempting.

Data placement in such systems is a major problem seeing as existing environments may have heterogeneous nodes, which process data at different rates, and non-dedicated environments can not promise full resource commitment to execute a query. Therefore it is critical to distribute a data warehouse's information in such a way that no node be overloaded or underestimated.

This paper presents the Capacity Evaluator (CE), an application capable of measuring the system's ability to process information, which in turn helps the Data Warehouse Parallel Architecture's (DWPA) automatic data placer determine exactly how much data should be allocated to each node.

1. Introduction

Application areas, such as data mining, digital libraries, multimedia services like video on demand, geographic information systems, etc… require that computers be capable of coping with the ever-growing need for speed and space [1].

Data warehouses are no exception to the rule. Their innate nature insists that they handle massive quantities of data in the least amount of time possible.

Presently data warehouses can be found residing within a mainframe or a parallel server. Mainframes are highly specialized monolithic systems that require large investments. Parallel servers, on the other hand, are an economically viable alternative, composed of tens to hundreds of fast standard microprocessors interconnected by a scalable network, with an aggregate memory that can reach to thousands of gigabytes.

In today's competitive and profit driven world, companies seek to find decision support systems (data warehouses) that are not only effective but also cheap. This makes parallelization very attractive.

There essentially exist three architectures for designing parallel database systems [2], being shared-nothing (SN) the most popular, due to its high scalability.

The SN architecture can basically be described as an agglomerate of autonomous processing nodes (PN), each owning a private physical memory and running a separate copy of the Database Management System (DBMS). In implementing this environment the data warehouse has to be partitioned among the PNs so that each DBMS instance can directly access data from the local partition. Access to non-local data requires that the PNs interchange messages with each other through a network.

Various SN systems have been implemented throughout the last couple of decades, ex: Bubba [3], GAMMA [4], Tandem [5] and Volcano [6]. These all contemplated a homogeneous environment, which meant that a company who did not posses a homogenous cluster of PNs would most likely have to purchase one.

Allied to the previous fact is that these systems were designed to exclusively dedicate themselves to processing the tasks at hand.

Such a system would only be justifiable if its mission was critical and resource consumption by foreign services was highly undesirable.

Recently researchers have begun to focus on parallel solutions that are cost-effective and yet very efficient, ex: Data Warehouse Striping (DWS) [7] and DWPA [8].

* [1] This work was partially supported by the Auto-DWPA project – FCT POSC/EIA/57974/2004

By introducing non-dedication and heterogeneity, various problems arise to question the traditional view of parallel processing. Probably one of the most pertinent is how should data be distributed between the PN in a non-dedicated and heterogeneous environment.

This paper provides a solution to this quandary by introducing the CE, an application capable of determining each PN's availability to sift through data.

The remainder of this paper is organized as follows. Section 2 introduces the DWPA architecture, for which the CE is intended. Section 3 describes what exactly the capacity evaluator does. This is followed by a short result demonstration. Finally, conclusions and future work are presented in Section 5.

2. DWPA

In general, DWPA functions in much the same way a generic SN system would. Figure 1 illustrates this aspect in a very simplistic manner.

A user sends a query to the data warehouse, without knowing that it is distributed.

A submitter node receives the query (Phase A) and if necessary rewrites it so as to allow intraquery parallelism, i.e. some aggregate functions should not be locally processed, requiring that they be derived into simpler functions, ex: AVG() can be derived into SUM() and COUNT(), before being shipped to the executer nodes, i.e. rest of the nodes (Phase B).

After the query has been locally executed, the PNs send the partial query results to a receptor node (Phase C). The receptor's goal is to merge the intermediate results, and if necessary apply global aggregate operations that were unable to be executed locally, ex: using the AVG example given earlier, one could obtain an average by summing the locally calculated SUM()'s and dividing that result by the sum of the local COUNT()'s.

Finally the receptor node sends the result set to the user. One should note that the submitter node usually assumes the task of global receptor.

Although DWPA is based on a SN architecture its main objective is to provide a data warehousing framework capable of surpassing common SN flaws. DWPA does this by focusing mainly on availability, processing and partitioning aspects [9].

Availability in SN systems is always of great concern because the data is spread through various nodes. If one or a couple of nodes should happen to malfunction or be temporarily put offline, the data warehouse becomes incomplete. When queried, the system will return a partial result to the end user. This is of course unacceptable if one seeks exact answers. Furtado [10], offers three replication strategies to avoid such problems: Full Replicas, Fully Partitioned Replicas and Partitioned Replicas.

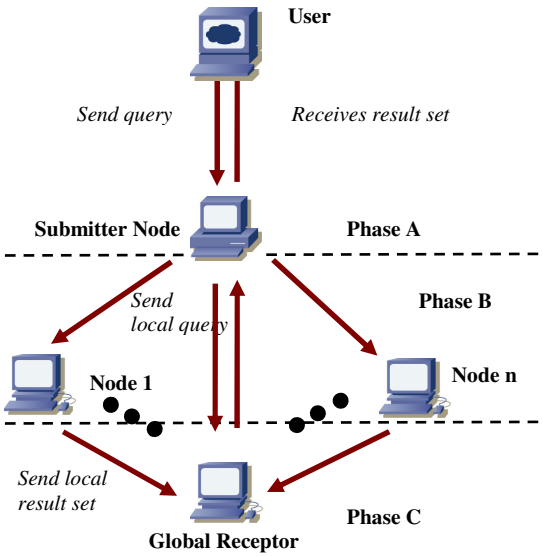

Figure 1. Generic illustration of the DWPA query processing mechanism.

Rapid and effective processing is essential to data warehouses. Distribution and parallelization are in themselves processing performance enhancements. However one could optimize such features by using more efficient algorithms. In [11], the author introduces the Parallel Hash-Join and the Parallel Associative Join to DWPA, both algorithms focus on decreasing the time needed to repartition data between nodes.

An important resource management issue in SN parallel database systems is data placement. Studies have shown that performance and scalability are reliant on the physical layout of the data across the nodes. A poor placement strategy could result in a non-uniform distribution of the load and the formation of bottlenecks [12]. DWPA is aware of this problem and tries to resolve it by placing data intelligently. The idea is to replicate all small tables and hash partition the larger ones, thus the architecture is able to avoid data imbalance and ultimately speed up query responses [13].

One of DWPA's forthcoming ventures is the introduction of an automatic data placer. This feature will be capable of deciding how data should be partitioned and how much of it is designated to a certain node.

In a non-dedicated environment possibly containing heterogeneous PNs, the data placer will need information pertaining to each PN's resource capacity as well as its "processing habits".

3. Capacity Evaluator

Although a good number of a PN's resources are essential to execute a query, of all, the most time-consuming are the hard disk [14] and the network [15]. One could say that the time used by the nuclear devices (CPU, RAM, Buses), to process a query, is practically discardable compared to those aforementioned.

Due to the above observation, the CE uses a set of tests to measure how each PN's hard disk and network connection hold up under simulated scenarios.

These scenarios are devised to mimic actions any PN would have to carry out while executing a parallel query.

3.1. Disk Benchmarks

The following enumeration states which actions are carried out by the disk benchmarks as well as a possible data warehouse scenario that it tries imitate:

- *sequential reads* – ex: sequence scans;
- *sequential writes* - used in table replication and for writing intermediate results to disk, i.e. for sorting purposes;
- *random reads* - common practice when the DBMS retrieves tuples from a relation being accessed through an unclustered index;
- *random writes* - indispensable for updating a series of non-sequential tuples.

Note that other data warehousing operations could also have been mentioned above.

Having the disk benchmark results of each PN, DWPA's automatic data placer could find out which hard disks are fastest, and decide to place a greater information load to that specific set of PNs.

Although the disk performance of a PN is only a small part of a larger problem (how to best place data within a parallel data warehouse), it is a crucial one.

3.2. Network Benchmarks

A network's performance is usually disregarded when designing an SN architecture. The reason for this is that people usually assume that the PN's will be interconnected by some sort of high speed LAN topology.

The automatic data placer can not afford to assume such a feature, especially if DWPA aspires to function in any environment no matter how extreme. For example: DWPA could hypothetically be run on clusters of PNs geographically separated by thousands of kilometers, or even in environments were large occurrences of network congestions are verified.

These scenarios require that careful ponderation be applied before deciding if and when network communication should be applied.

Once again the CE tries to simulate such possibilities by periodically testing how the parallel data warehouse's network connections cope with typical message passing between PNs.

The CE has contemplated a set of possible situations where network communication is necessary.

These situations are as follows:

- *1 to 1 communication* - usually necessary when a PN is replicating a relation, or part of it, to another PN [10];
- *1 to N communication* – applied in two different situations:
    a. Data repartitioning between a "producer" PN and various "consumer" PNs;
    b. Partial data replication of one PN to various PNs, as described in [10];
- *N to 1 communication* – used in intermediate result set merges, i.e. occurs when various PNs have finished executing their local query and the results need to be shipped to the global receptor node;
- *M to N communication* – similar to *1 – N a. communication*, except for the fact data repartitioning occurs with various "producer" PNs sending parcels of data to various "consumer" PNs.

As was mentioned with the disk benchmarks, so too are the network benchmarks a small piece of a larger puzzle. But nevertheless it could determine the data placer's decision on how much of the data warehouse's information should be stored within a specific PN.

3.3. PN Profiles

The previous sub-sections have briefly described the tests the CE uses to determine a node's resource capacity. However these results refer only to a PN's current state. A snapshot of a system's resources should never be used to determine how much information it could process, especially in a non-dedicated environment, because that snapshot only reveals a momentary capacity that could very well change the following second, ex: due to the execution of a foreign task.

Aware of this problem, the CE is programmed to periodically re-launch the tests executed on the PNs. This periodicity is dictated by the parallel data warehouse's administrator.

The CE maintains a resource histogram for each PN, which he updates with every received benchmark result. In doing so, an individualized profile of each PN is maintained.

By resorting to a much more substantiated method of measurement, the CE is now able to decide on how data should be placed, with a greater degree of confidence.

4. Experimental Results

After the implementation phase, CE was tested on a small cluster of PNs.

The experimental environment was composed of 5 PNs with the following characteristics:

- Nodes 1,2,4 and 5 – Each had a Pentium IV at 3.0 gigahertz (GHz) processor, with 2 gigabytes (GB) of RAM and two hard disks containing 200 GB of capacity each, running at 7200 rotations per minute (RPM). The operating system run on all four nodes was Windows XP Professional Service Pack 2 (SP2). Nodes 1 and 2 had gigabit per second (Gbps) network cards, while nodes 4 and 5 ran with 100 megabits per second (Mbps) network cards;
- Node 3 – Had a Pentium Centrino at 1.6 GHz, with 512 megabytes (MB) of RAM and one 60 GB hard disk running at 4200 RPM. The PN's operating system was a Windows XP Home Edition SP2. The network card had a nominal capacity of 100 megabits per second (Mbps);
- The interconnecting network topology for all five PNs was a Gigabit Ethernet.

The objective of this section is to demonstrate that the CE, as a concept, works. Only a few results will be discussed due to writing space constraints.

4.1. Test 1 – 1 to 1 Throughput

This test aims at calculating the real network throughput between nodes 3 and 1. Node 3 is going to transfer 300 MB of memory resident data to node 1. For illustrative purposes this test will be run three times.

Fig. 1 displays a bar chart with the results obtained from each run. Each result is represented as kilobytes per second (KBps). One must note that this is only a captured moment in the nodes daily routine, and that no conclusions should be withdrawn from its analysis, seeing as non-dedicated environments may vary their performance.

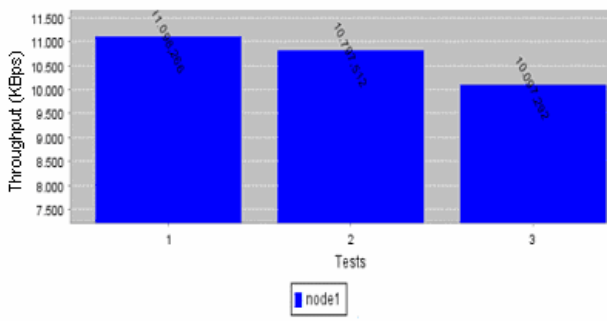

Figure 2. 1 to 1 network throughput test using three runs between PN 3 and PN 1.

Several more tests were executed with the same parameters, so as to introduce more information for the profiles. The result is displayed in fig 3.

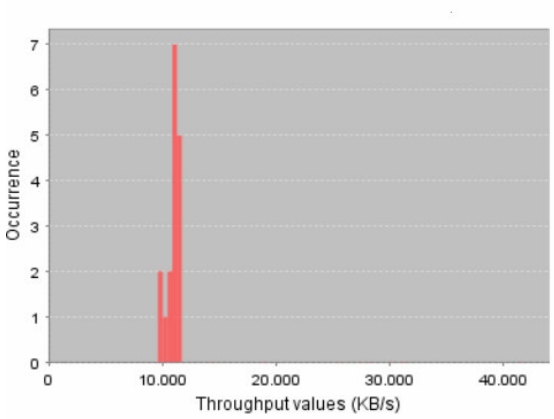

Figure 3. A 1 to 1 communication network profile between PN 3 and PN 1.

The histogram categorizes the run results into bins. Every time a result falls within a bin's interval, that bin is incremented by one. The above histogram shows us that the througput between both nodes never exceeds 12,5 MBps. Knowing that the gigabit ethernet network has a throughput of 125 MBps we can conclude that node 3's network card is slowing the data transfer between both PNs. Another conclusion that can also be withdrawn, is that the communication throughput is fairly constant, meaning that very little of node 3's network bandwidth is used by foreign applications.

The DWPA auto data placer could use this information to allocate less data on node 3, or in the case of an intensive information exchange enviroment it could opt to replicate relations so as to avoid network exchange.

4.2. Test 2 – N to M Throughput

This test tries to simulate data repartitioning from N PNs to M destination PNs. Nodes 3 and 5 are going to

reparation 300 MB of disk generated data to the rest of the cluster's PNs. Fig. 4 illustrates the data interchange between the nodes.

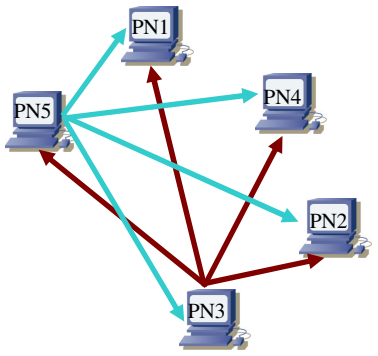

Figure 4. PNs 3 and 5 repartition their data between the remaining PNs in the cluster.

The operation is complex because it involves swapping sections of data sets between PNs. As was said earlier (Section 1), the SN architecture divides the data warehouse throughout the nodes. In some queries PNs need to process tuples that are not locally situated on their hard drives, meaning that these PNs will have to import the missing tuples from their peers. The amount of data transferred could be voluminous.

Both network and hard disk devices are challenged in this operation because quasi simultaneous disk reads and writes occur while network receptions and emissions are being carried out.

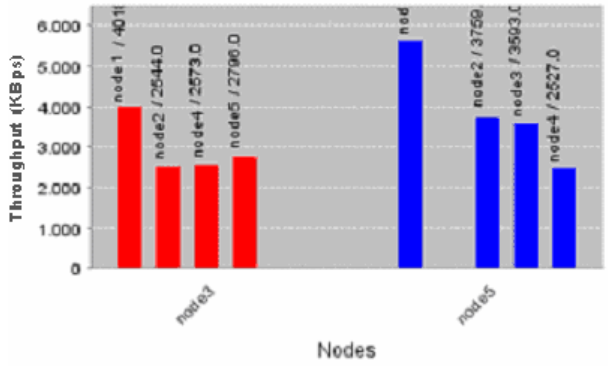

Figure 5. Network throughput results of PNs 3 and 5 repartitioning effort.

Fig. 5 is a graphic generated result of node 3 and 5's repartitioning efforts. Both nodes have a 100 Mbps bandwidth, which they split between each destination PN.

Analysing the aforementioned figure, one can claim various conclusions. The most obvious would be that the sum of the throughputs of each destination PN is no larger than the source node's nominal bandwidth 100 Mbps (12,5MBps). Another assertion is that the disk transfer rates are not influential in data repartitioning when the available bandwidth is relatively low. This could however have a different outcome if the PN outputting the data were to have a 1Gbps network card, seeing as few hard disks can reach a transfer rate of 125 MBps (≈ 1Gbps).

4.3. Test 3 – Disk Transfer Rate

This test tries to measure a local PN's disk transfer rate for various types of actions (see section 3), commonly executed within a data warehouse.

The test results shown below are withdrawn from node 3's profile. All four disk actions were performed using 300 MB of data as their threshold.

Table 1. Disk performance results on PN 3.

Action	Result (KBps)
Write Throughput	5346.59
Read Throughput	5247.61
Random write throughput	553.10
Random read throughput	378.49

The results presented here can in some ways contradict the analyses made earlier in test 2: "Another assertion is that the disk transfer rates are not ...". It is easily seen that if a large amount of data were to be read randomly, the bottleneck would be found on the hard disk device, instead of the network's throughput.

Depending on the type of query processing being carried out by each PN, some devices will be more affected than others, ultimately dictating how the PNs perform.

5. Conclusions and Future Work

Today's companies hold a panoply of computer programs and services that all require sufficient resources to function properly, but does this mean that each program or service should reside in an individual computer or cluster of computers? The ideal answer would probably be yes, but to invest in so much hardware when existing systems can probably handle the load, is a misuse of a business's funds.

People tend to be cost aware when they have already acquired computer systems in a no so distant past, and are many times reluctant to buy new hardware just to accommodate a new service.

DWPA is an architecture that tries to use the existing environment to its advantage, seeking to constantly adapt in order to perform well. Adaptation requires that data be placed as best possible, and this is impossible if the

architecture is unaware of what each PN is capable of doing.

The CE was introduced to inform DWPA of how fast the slowest hardware devices, network and disk, of a PN could perform. With the collected results, the automatic data placer could now base its placement strategy on real capacity values instead of nominal or factory based values.

Non-dedication adds complexity to an already complicated problem resolution. The CE tries to diminish the difficulty by creating profiles that reflect the PN's daily routine, which would then be used to decide how the DWPA architecture should distribute the query's processing.

The next step in the CE's lifecycle is to integrate it with the DWPA architecture. At the present moment the CE is an autonomous program that does not feed its managed profiles to DWPA.

REFERENCES

[1] E. Rahm, "Dynamic Load Balancing in Parallel Database Systems", in Proceedings EURO-PAR, 1996, pp. 37-52

[2] D.J. DeWitt and J. Gray, "Parallel Database Systems: The Future of High Performance Database Systems", in Communications of the ACM, vol. 35, number 6, 1992, pp. 85-98.

[3] Brown, K., Carey, M., Dewitt, D., Mehta, M. and Naughton, J., "Resource Allocation and Scheduling for Mixed Database Workloads", Computer Sciences Technical Report #1095, Department of Computer Sciences, University of Wisconsin, Madison, July 1992.

[4] DeWitt, D. et al., "The Gamma Database Machine Project", IEEE Transactions on Knowledge and Data Engineering, 2(1), March 1990.

[5] Tandem Performance Group, "A benchmark of non-stop SQL on the debit credit transaction", Proc. ACM SIGMOD Conf., Chicago, Ill, June 1988.

[6] Graefe, G., "Volcano: An extensible and parallel dataflow query processing system", Computer Science Technical Report, Oregon Graduate Center, Beaverton, Ore, June 1989.

[7] Bernardino, J., "Técnicas para o Aumento do Desempenho e da Disponibilidade em Data Warehouses", Ph.D thesis, University of Coimbra, Coimbra, Portugal, December 2000.

[8] Furtado, P. , "Algorithms for Efficient Processing of Complex Queries in Node-Partitioned Data Warehouses", in Proc. of the IEEE Computer Society Press in the proceedings of the Eighth International Database Engineering and Applications Symposium (IDEAS04), Eighth International Database Engineering & Applications Symposium, Coimbra, July 2004

[9] P. Furtado, "Efficient and Robust Node-Partitioned Data Warehouses", in Data Warehouses and OLAP: Concepts, Architectures and Solutions, Eds. Idea Group, 2007

[10] P. Furtado, "Replication in Node Partitioned Data Warehouses", in Proceedings of the LNCS, VLDB Ws. on Design, Implementation, and Deployment of Database Replication, 2005.

[11] P. Furtado, "Algorithms for Efficient Processing of Complex Queries in Node-Partitioned Data Warehouses", in IDEAS '04: Proceedings of the International Database Engineering and Applications Symposium, 2004, pp. 117-122

[12] M. Mehta, D. DeWitt, "Data placement in shared-nothing parallel database systems", VLDB Journal: Very Large Databases, vol. 6(1), pp. 53-72, 1997

[13] P. Furtado, "Experimental Evidence on Partitioning in Parallel Data Warehouses", in DOLAP 04 - WORKSHOP of the International Conference on Information and Knowledge Management (CIKM), 2004, pp. 23-30

[14] H. Garcia Molina, *Database Systems: The Complete Book*, Prentice Hall, 2002

[15] A. Leon-Garcia, I. Widjaja, *Communication Networks: Fundamental Concepts and Key Architectures*, McGraw-Hill, 2001.