Improving performance of Long-distance Geographically Distributed Dedicated Clusters

Adhvan Furtado[1], Andre Rebouças[1], Josemar Souza[1], Eduardo Argollo[2], Dolores Rechaxs[2], Emílio Luque[2]

[1]CEBACAD – Centro Baiano de Computação de Alto Desempenho, Curso de Informática, Universidade Católica do Salvador (UCSal), Campus da Federação, nº 205
40.220-140 – Salvador – BA – Brazil

[2]Unidad de Arquitectura de Ordenadores y Sistemas Operativos, Departamento de Informatica, Universitat Autonoma de Barcelona (UAB), Edifici Q
08193 – Bellaterra - Spain

`{adhvannf,andrecm,josemar}@ucsal.br, eargollo@aows10.uab.es,`
`{Dolores.Rexachs, Emilio.Luque}@uab.es`

Abstract. *A collection of geographically distributed dedicated Heterogeneous Network of Workstations interconnected by a long distance network (LDN) could be an efficient and cheap solution for large-scale problems or data intensive computation. This paper presents a hierarchical architecture combined with a dynamic data distribution politic designed under a layered scheme and a model that improves the LDN usage. An adaptive Master/Worker model is used where a sub-master groups' HNOWs workers. A separate machine is used to manage communications over the LDN. The chosen benchmark algorithm was the Matrix Multiplication. We use a library based in the MPI standard and Sockets to improve the communication over the LDN. The testbed system used is composed of two dedicated HNOWs, one in Spain and the other in Brazil, interconnected by a non-dedicated LDN.*

Resumo. *Um conjunto de HNOWs distribuídos, interconectados por uma rede de longa distância (LDN) pode ser considerado uma solução eficiente e barata para processamento de dados em grande escala. Este artigo apresenta uma arquitetura hierárquica combinada com uma política de distribuição de dados dinâmica, desenvolvida sobre um esquema de camadas, e um modelo que melhora a comunicação através de uma LDN. Um modelo Mestre/Trabalhador flexível é usado onde um Sub-mestre agrupa os trabalhadores de um HNOW. Uma maquina separada é usada para gerir a comunicação através da LDN. O* benchmark *escolhido foi a Multiplicação de Matrizes. Nós usamos uma biblioteca de comunicação baseada no padrão MPI combinado com* Sockets*, justamente para melhorar a comunicação através da LDN. O sistema de teste é composto por dois HNOWs dedicados, um localizado na Espanha e outro no Brasil, conectados por uma LDN não dedicada.*

1. Introduction

With the spread of Internet use and its increasing bandwidth and reliability enhancement, the possibility to interconnect geographically scattered groups of low cost parallel machines became real; thus, a collection of these machines (CoHNOW), interconnected by a long distance network can cooperate in solving a common large-scale problem [1]. This environment is named Long-distance Geographically Distributed Dedicated Clusters (LDGD n-Clusters). This kind of machine can be an efficient and cheap solution for intensive and large computational problems; however, its efficient management is no trivial matter.

Defining efficient policies related to the workload distribution and management is essential to achieve high performance in this particular system. Computational and communication overlapping techniques must be implemented in order to improve the system efficiency. Also the idea of redistributing the remote workload among the local workers, once they are available, is fundamental to deal with sudden Internet performance decay. From a certain point of view, the idea of grouping HNOWs scattered around the world is tied to the Grid concept of a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities [2]. This paper main goal is to propose an architecture combined with a dynamic data distribution policy designed under a layered conceptual model that reduces influence of the LDN in the system overall performance. Also this on going study defines some communication drawbacks related to the interconnection under the LDN and presents some equations that can predict the cluster beahavior.

The Matrix Multiplication (MM) algorithm is the benchmark application. It is a highly scalable algorithm and the amount of work can be modified with no great efforts. According to the LIP Group [3], the MM problem with different-speed processors turns out to be surprisingly difficult, in fact the LIP group proved its NP-completeness over heterogeneous platforms. A new parallel MM algorithm was proposed by [4] in order to take advantage of the specific characteristics of HNOW.

Two clusters geographically separated compose our testbed system. Each cluster is one dedicated HNOW, one of them located in Salvador, Bahia, Brazil and the other one located in Bellaterra, Barcelona, Spain. The idea here is to group them into one LDGD Clusters, interconnected by a LDN (Internet), cooperating together to solve large-scale problems (Figure 1.1). In fact, you will see in this document that, in order to gather those scattered resources, we have performed our application benchmark under the Master/Worker paradigm.

The Spanish HNOW is located in UAB (*Universitat Autònoma de Barcelona*), in the CAOS (Computer Architecture and Operating Systems) Department. The HNOW from Brazil is located in UCSal (*Universidade Católica do Salvador*), *Federação* Campus, at the CEBACAD's (*Centro Baiano de Computação de Alto Desempenho*) lab. Nowadays, CEBACAD has a team with twelve researchers (students and teachers) and two HNOWs, one production cluster with sixteen hosts, and another for practices with four hosts.

An important issue of dealing with such a heterogeneous environment is the interconnection network. As we can see in the Figure 1.1, we can identify two levels of interconnection, a local interconnection network (LAN – responsible for interconnecting the local machines) and a remote interconnection network (a public WAN – responsible for interconnecting each group of machines). As we now, the Internet is a public network, and its latency and bandwidth can be extremely unpredictable. Therefore, the network latency unpredictability, forces us to use policies that should allow our application to deal with the possible variations of this value parameter.

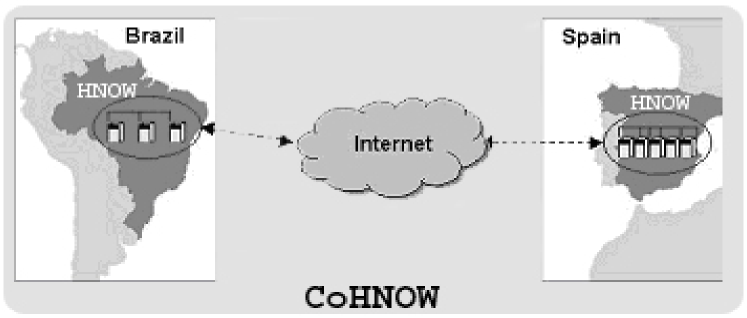

Figure 1.1 – Testbed system

The system architecture evolved in the way that the CoHNOW has been organized as a hierarchical Master/Worker based collection of HNOW's using MPI [5] for intra-cluster communication. To optimize inter-cluster communication performance, guarantee transparency and isolate and solve Internet connections failures the proposed architecture dedicates a workstation per HNOW for the long-distance communication task [6] [7]. This task is done by special developed software, the Communication Manager (CM). Pipelines strategies are implemented in each communication level in order to overlap communication and computation.

The following sections present our study in further detail. Section 2 describes the system's proposed architecture, the conceptual model and the communication problems involved. Some experiments are described in Section 3 and finally, conclusions and further work are presented in Section 4.

2. Proposed Architecture

Make distributed heterogeneous clusters collaborate is not an easy effort, there are many variables that impact the execution time. The algorithm should balance the load among workers so all machines can collaborate equally. Each machine and each cluster have different performances and communication time represents a major bottleneck specially when using a public WAN with unpredictable behavior. In order to optimize the execution of parallel applications, we propose a hierarchical architecture and a dynamic distribution policy, this architecture fits in a layered conceptual model.

2.1 Conceptual Model

Dividing the system in layers we can better separate problems boundaries, analyze and propose solutions for each level without interfering the others, and gain in modularity and clearness in the program design.

This Conceptual Model shown in Figure 2.1 [7] can be fit in five different layers, divided in two levels, the upper Software oriented level, and the lower and Structure oriented one.

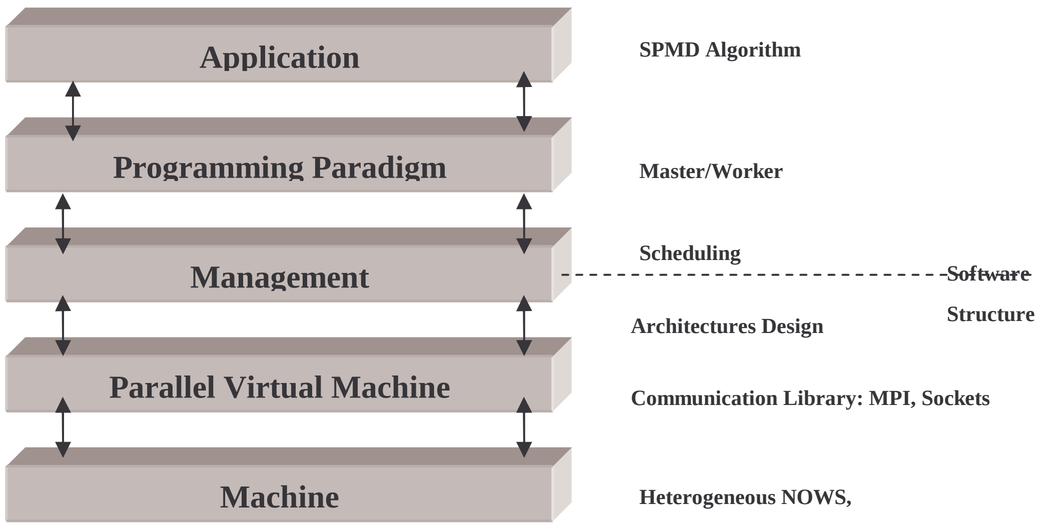

Figure 2.1 –System Levels

In the upper half is the **Application** plane where the parallel application and its parallelization are defined. The parallel algorithm should be isolated from other auxiliary code for portability and clearness.

The **Programming Paradigm Level** defines the underlying model in which the algorithm will be parallelized, among all parallel programming paradigms, we have chosen the Master Worker paradigm. Our situation corresponds to a very unpredictable environment, heterogeneous machines with distinct computing capabilities dedicated to solve a common problem. Inside each cluster we can determine a known throughput behavior with a previous characterization. On the other side we use a public interconnection network to link different clusters, and this link has a unpredictable behavior, leading to a complex environment where the computing time should be analyzed as dependent of the computing power and the communication time.

The M/W paradigm itself facilitates centralization of the system management, with the master providing a central point of control. In a communication environment with a time variable latency, it is important to dynamic measure and control network usability, a controller aware of any changes is easily adapted to a M/W paradigm; also this paradigm can be applied to hierarchical architectures.

A hierarchical Master Worker model is an extension of the common M/W paradigm where we can have specialized workers acting as Sub Masters capable of gathering data from and distributing data to workers also called sub Workers, as can be seen in figure 2.2.

This hierarchical architecture is well suited to distributed clusters, each cluster node acts in an independent master-worker paradigm inside itself, and also communicate to a central master node with the same paradigm. It is important to remark that this M/W architecture provides a number of hierarchical models in order to gather data and optimize distribution over sub-clusters. The possibilities of expanding this model are unlimited. The connection between a master and a sub-master usually can be different (in latency for example) from the one that interconnect a master to a worker. The master and workers could be in the same LAN, the sub-master and its sub-workers in other LAN, and the connection between them could be a public WAN.

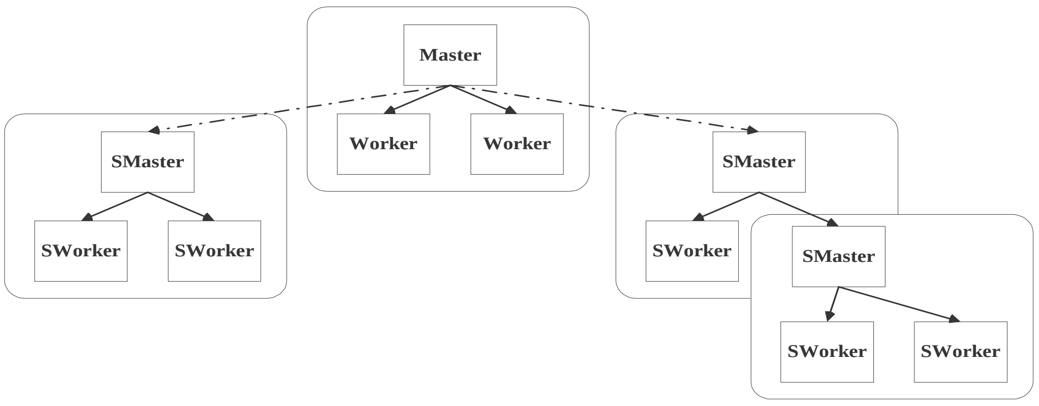

Figure 2.2 – Hierarchical Master-Worker

The **Management** layer is the interface between the Application Logic and the Physical Structure views, there are both software and hardware components in it, logically this layer defines the data distribution and scheduling politics, and physically settle on the task allocation scheme where machines are specialized to receive tasks. It is important to separate the application-paralleling problem from the structural environment problem. This layer is responsible of hiding communication latency and machines heterogeneity from the Program Paradigm level. Different architectures can be set up to provide efficient Allocation, Scheduling and Data Distribution polices for the parallel algorithm.

For clearness we will define this parameters as:

Allocation- Determines which machine fits each task, based on their computational characteristics, so a machine with a better network card would be better used in managing network communication while an extreme powerful machine should be better used computing.

Scheduling- it will provide a time driven politic to send data to workers, indicating when the data should be sent, statically at the beginning of the execution or dynamically while workers finish their jobs

Data Distribution- This model defines the data granularity sent to each process, defining if they are coarse grained or fine grained depending on the network and computational capabilities. We realized that we have two levels of granularities one for communication and other for computation

Well adjusting these parameters, the workload would be balanced among machines, not neglecting the time lost with communications.

The **Parallel Virtual Machine System** implements a software layer upon the operating system which provides the illusion of working over a single machine even when using a collection of clusters and heterogeneous machines. At this level lays the communications libraries: MPICH based in the MPI standard, PVM, and/or any other program to communicate machines, in our case a socket based extension to the MPI library is also used.

At the **Machine** Level are the physical machines, the hardware of the clusters: machine's cpu, memory, I/O devices; the operating system with its communication buffers and internal schedulers, the network adapter and the interconnection network: LAN or WAN.

This layered model separates the Cluster Machine complexity in small blocks, our intention is to clear down what are the details behind the execution of algorithm over a machine with this characteristics, and more, better situate our contribution in the Cluster programming and architecture field.

2.2 LDGD nClusters Architecture

After analyzing different architectures we chose an architecture that can obtain high performance minimizing communication effects by overlapping computation with communication [7]. In this architecture the final Cluster can be seen as an organized collection of HNOW, therefore called CoHNOW. There is a main cluster, which contains the master, on which all data resides and from which the application execution starts. All other remote clusters are also called sub-clusters, its masters, sub-masters and its workers, sub-workers. (Figure 2.3)

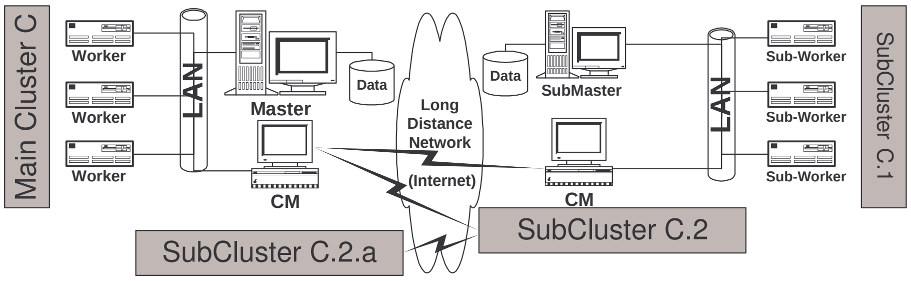

Figure 2.3 – LDGD Clusters Architecture

The M/W paradigm itself facilitates centralization of the system management by using a central point that should function as a global master, managing all communications with others clusters, controlling task execution and also reallocating data, when necessary. One advantage of this architecture is that the master does not have to be always tied to one specific Cluster. We realized that adding machine organization we could improve performance; first we detected the necessity of uncoupling the slow link from the Master and Sub-Master, in the meantime their were sending data through the long distance network (LDN) the answer to local workers work requests was delayed. Another problem was the instability of the communication library while communicating in the LDN. To solve these problems different strategies were implemented for LAN and WAN communication. Inside the local network, low latency and high throughput permits the utilization of a standard MPI based library. For the outside communication at each cluster one machine was targeted to act as the point of intercommunication and special software was developed: the Communication Manager (CM). CM's isolate the local network from the public one, manage the public network failures, guarantee the inter-cluster communication and maintain it continuously, exploiting as much as possible the communication link unpredictable throughput.

Communication Managers proved to be a necessary resource not just to efficiently provide reliable communication over the Internet link, but also providing transparency on this process.

2.3 Communication Problems

We have two communications levels, one inside each cluster, where we use a dedicated Ethernet or Fast Ethernet LAN with a predictable bandwidth and low latency, and other between geographically distributed clusters where a common Internet link is used for communication where we could expect low bandwidth and high latencies. Inside each cluster we communicate the machines using an implementation of the MPI standard called MPICH library [8], and it worked fine to our communications purposes. Nevertheless, a stable inter cluster communication could not be achieved with the MPICH library, the program just hang up.

Some tests were made to better characterize the network behavior. Figure 2.4a and b shows some of these results.

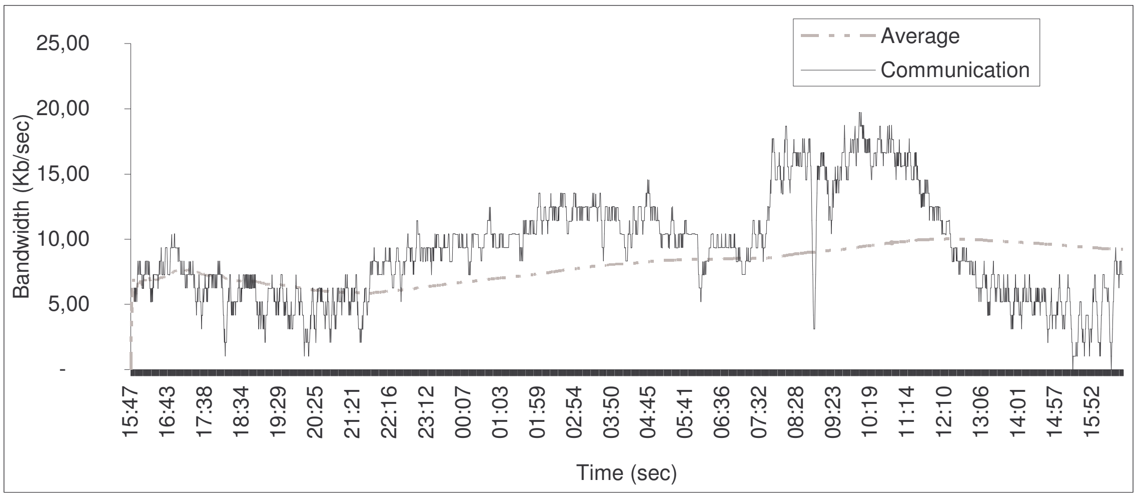

Figure 2.4 (a) – Bandwidth in the execution of a Ping Pong program with a packet size of 320KB during 24 hours.

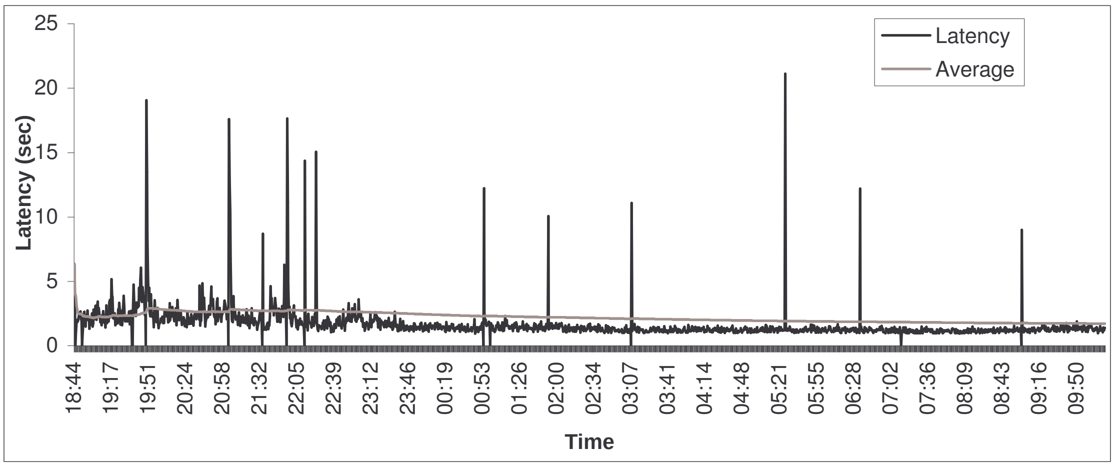

Figure 2.4 (b) - Latency in the execution of a ping-pong program in a different day.

In figure 2.4(a) we can observe an average bandwidth of 9.25 KBps and in the second one, an average latency of 1.39 seconds, normal values for an Internet connection between Brazil and Spain. All the programs were based in TCP sockets with a 40 seconds forced time out, and while testing we observed a high number of TCP disconnections. In the first example in 24 hours there were 152 disconnections, more than one each ten minutes. These disconnections led to problems in the communications between the two clusters using MPICH. We were unable to sustain an MPICH cluster and the program hang up; it is a reported failure and according to the MPICH group it happens due weak Linux TCP connections, especially under heavy traffic while communicating. There are a number of MPI communication libraries implementations dedicated to long distance communication, our group is studying this implementation and how they can be adapted to our environment.

In order to avoid disconnections and maintain the communication between clusters another element was introduced to the architecture: the Communication Manager (CM), figure 2.5. It considers long periods of lack of communication as a disconnection and opens other socket connection.

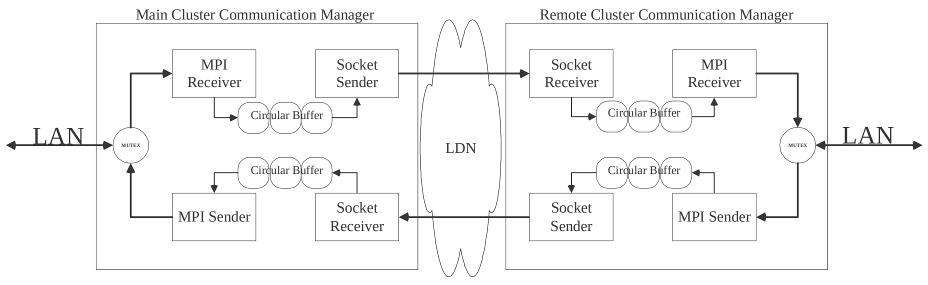

Figure 2.5 – Communication Manager architecture

The main concept of the CM is the idea of achieving the best use of the LDN link. In order to do that, the CM has to differentiate intra and inter-cluster communication granularity and has to be always prepared to exploit bandwidth peaks and manage communication fails. To isolate the networks, CMs machines have two net cards (one for the LAN and other for LDN) and they act as the only way for accessing the remote cluster.

The Communication Manager architecture is organized around four threads and two circular buffers. The threads permit the independent and concurrent transmission and reception of data. Buffers will get full when low bandwidth are reached and can be rapidly used on bandwidth peaks. The CM manages different packet sizes for LAN and LDN communication, dividing and joining then, so that the best Internet throughput can be obtained. In case of Internet disconnection, the CMs will reconnect when possible and resend lost data transparently. A mutex semaphore is necessary in order not to happen two simultaneous MPI calls, avoiding MPICH failures. Although other alternatives are being tested, the inter-network communication is made through the use of two sockets TCP connections, each one in one way.

2.4 Load Balance and allocation scheme

To achieve high performance we must yield all computing power available. In LDGD clusters, communication time can be very time consuming. To masquerade long distance communications effects we should overlap local computation with communication just like a pipeline. The first requirement for this task is fully use the network when the link between clusters is available, and when some failures such as TCP disconnections occurs, we should be able to reconnect and reestablish the far cluster maximum computation in the less possible time.

Notice that, to achieve these goals we must collaborate architecture design with load balance politics.

Within the architecture previously defined we have a specialized machine dedicated to communications, the communication manager, responsible of managing the network link. It is important to remark that the master does not notice the communication manager, and the CM is able to divide the received message in small blocks, re-send then in any specified size and, in the other side, reorganize the message in its original size. Communication and computation blocks do not have to be of the same size.

The load distribution should be dynamic; the workload is divided in blocks and sent to workers, as soon as worker machine finishes its job, another block is assigned to it. To overlap computation and communication, the workers should not wait for blocks. There are buffers in each worker, as soon as a worker finishes a block, a message is sent to the master requesting for more data, and the buffer is computed.

The better execution of a specific application requires a fine-tuning in the load balance and allocation scheme. We can observe some parameters useful for defining politics such as data block and communication block granularity, total number of messages, network throughput and cluster performance. A formulation can be inferred from the relation of these elements, shown in the next section.

We must achieve each cluster higher performance and sustain this performance as long as possible. If the necessary time to achieve the higher performance in the remote cluster is longer than the time it spends working, the contribution to the overall execution is low. This relation can be observed in figure 2.6.

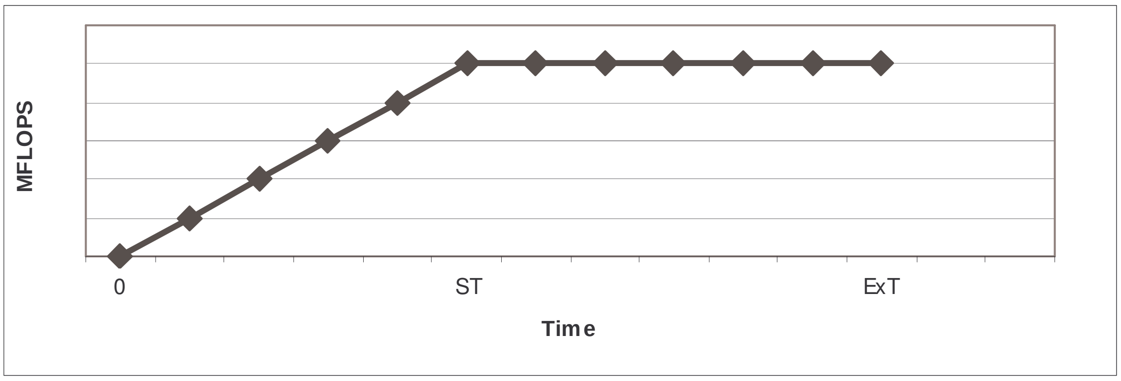

Figure 2.6 – Expected MFLOPS/Time relation in the remote cluster

The stabilization time (ST) is the necessary time to fill the pipeline by sending enough work to the remote cluster, making it LDN independent. This time is dependent on the parallel algorithm ratio between computation complexity and communication complexity to define how computation data is divided to fill in the pipeline, and it is also dependent on network latency. ExT represents the total execution time in the graph. The more the workload increase the less representative will be the amount of time spent until the stabilization is reached and the more will be the obtained average performance over the execution.

When applying real experiments, we cannot expect a linear progress until the stabilization time, neither a constant performance of the cluster after it. It will vary with the ratio communication/computation of the algorithm and the network throughput behavior.

Our effort is to develop a system capable of adapt itself to any parallel algorithm, by now we are proving the execution of a specific problem, Matrix Multiplication, and trying to achieve the higher possible performance.

3. Experimentation

In order to validate our architecture some experimentations were done. We try to prove that adjusting some parameters the best possible performance on LDGD Clusters can be achieved. Also the contribution of each cluster can be predicted.

As a benchmark algorithm we use the Matrix Multiplication (MM) problem. The MM algorithm is an important Linear Algebra Kernel [9] and a well-known benchmark. By now we do not use any compiler enhancement or pre built blocks of functions, we just use a blocking optimization technique for improving the effectiveness of memory hierarchies. Blocked algorithms operate on sub matrices or blocks, so that data loaded into the faster levels of the memory hierarchy are reused [10].

Pairs of blocks are distributed dynamically among workers, so the system adjusts itself. Each local worker has data buffers, when a block is computed a data request is sent to the master and the worker computes the blocks in the buffer, overlapping communication with computation. The maximum intra cluster performance is achieved when there is no worker idle time and this happens whenever computation time is greater or equal to communication time. The computation time for one block is a relation between the number of its floating-point operations and the sum of the cluster performance for this block size. The cluster performance is the sum of the serial MM block algorithm execution on each worker. The communication time is dependent of the data block size, the algorithm logic and also of the LAN throughput. Analyzing these characteristics some parameters can be identified: Matrix size (M), number of elements in a block (B), floating point data size (α) and LAN Throughout (LanTPut).

With these parameters, an equation of cluster performance limit (ClusterPerfLimit) for a block size and network can be obtained:

$$ClusterPerfLimit \leq \frac{(2 * B - 1) * LanTPut}{\partial}$$

The effective cluster execution time for a block size can be achieved as the minimum between the local cluster performance and the cluster performance limit:

$$ClusterExpectedPerf(B) = \min(LocalClusterPerf(B), ClusterPerfLimit)$$

Once modeled the available performance that could be reached by the different clusters, the next target is to consider the inter-cluster communication and the workload distribution and management between clusters in order to determine the percentage of the remote cluster performance that can be attained to obtain the best performance on the whole CoHNOW.

To optimize the execution in the remote cluster, the sub-master should receive enough data to distribute to all workers. Due to the MM algorithm data locality, the sub-master can reuse data already received with new arrived ones, generating new sets of multiplications to the sub workers.

In the MM application this can be obtained through the distribution of complete operands sets of rows/columns. Sending one entire line, column one block can be computed, sending another line, column more three blocks can be computed, as seen in figure 3.1. Keeping the same data packet granularity (one column, line) more computation can be done at each new delivery.

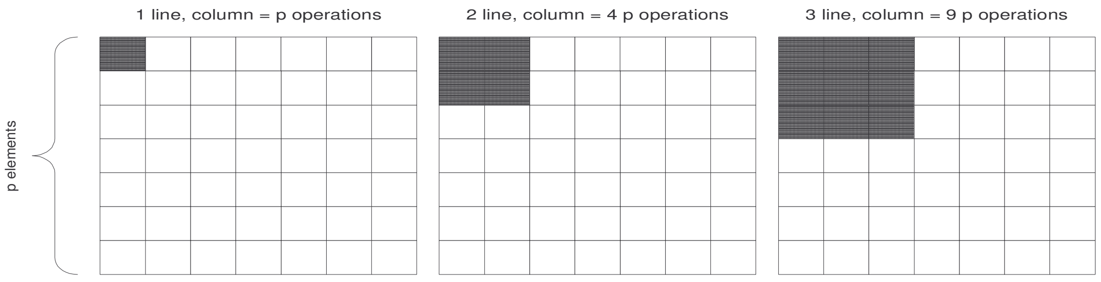
1 line, column = p operations
2 line, column = 4 p operations
3 line, column = 9 p operations
p elements

Figure 3.1 – For each new line, column L sent more block operations can be executed.

Each new row/column (r/c) pair that reaches the sub-master can be computed with the previous received columns/rows. Supposing a matrix with N blocks per r/c, when the r/c L is received by the sub-master a total of L^2*N block operations are possible. The new possible operations for the received r/c L are the subtraction between the new total possible operations L^2 by the already available $(L-1)^2$ operations.

It can be concluded that, with this strategy, for a new r/c L sent (2*L - 1)*N new block operations are possible. With a study similar to the one used for the intracluster performance limit, the following formula was achieved:

$$ContribPerfLimit \leq \frac{(2*M-1)*LdnResTPut}{\alpha}$$

The limit of the remote cluster contribution (ContribPerfLim) is bounded by the Matrix size and the throughput of the response blocks. Meaning that the total remote cluster expected performance could always be reached once the computational problem is sufficiently large. Once again the effective cluster contribution is limited by the minimum between the contribution performance limit and the cluster expected performance.

Two tests were done to confirm the methodology and the architecture usefulness, the tests prediction values are shown at Table 1 considering a throughput of 10KBps, the Brazil cluster as local and Spain cluster as remote.

For the first test a matrix of 10,000 x 10,000 and a block of 100 x 100 elements was selected. At this case both clusters will have their own performance limited by the LAN and the stabilization point will be reached at 450 minutes when 35 columns are sent.

This test had 19 hours duration and its performance execution behavior through time, for each cluster on the CoHNOW, is shown at Fig. 3.2 and the experiment throughput evolution at Fig. 3.3 The system was first stabilized on the minute 316 due to the throughput improvement between the minute 206 and 400. A gap on the remote collaboration is seen on the minute 498 as a consequence of sudden decrease of the links throughput. The second stabilization point is reached at 512 when the column 35 is sent.

M	B	Brazil (Mflops)	Brazil/Expectec (Mflops)	Spain (Mflops)	Contrib Perf Limit (Mflops)	Expected Contrib Perf (Mflops)	MinLC	Stabilization Time(Min)	Stabilization Speedup
10.000	100	44,27	17,39	95,15	30,72	17,39	34	448,75	2,00
20.000	400	28,97	28,97	58,10	61,44	58,10	29	3.007,35	3,01

Table 1. Execution prediction to different workloads for 100 x 100 and 400 x 400 elements block.

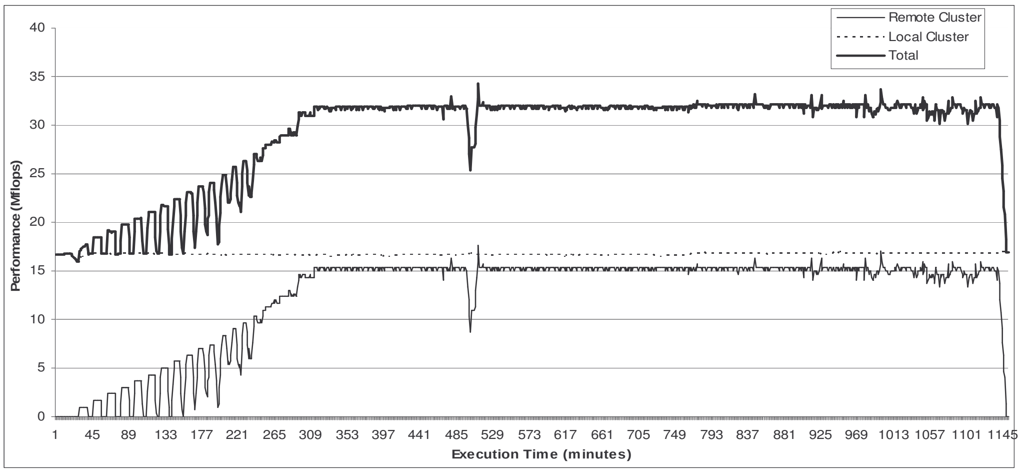

Figure 3.2 - Two cluster execution behavior.

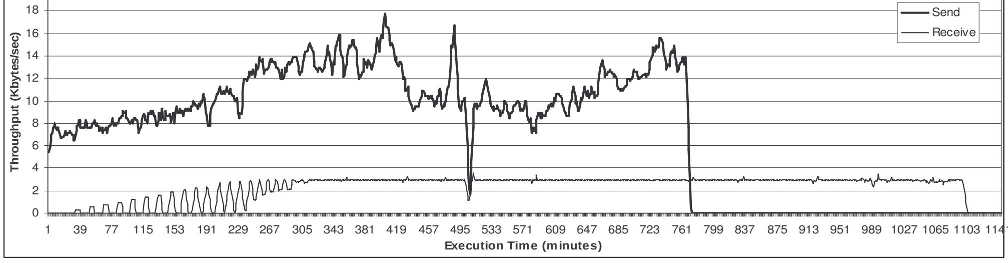

Figure 3.3 - Throughput on the CoHNOW execution.

The second experiment was as an attempt to get close to the maximum possible CoHNOW performance and the 400 x 400 block size was chosen with a 20,000 x 20,000 matrix. The experiment lasted for 89 hours and the obtained stabilized performance was 90% the maximum CoHNOW performance (Spain + Brazil performance).

It is important to remark that in both experiments each cluster contributed closer to its maximun performance; thus contributing to reduce the overall execution time.

4. Conclusion and Further Work

With the support of all knowledge provided by the experimentations performed, we can conclude that effective contribution can be achieved in LDGD Clusters if some machine organization is issued. This was accomplished by the system architecture in figure 2.3 with a communication manager responsible of the better use of the network link. Overlapping computation and communication each cluster could contribute to complete the task; some formulas were defined to predict the limit of each cluster collaboration.

Using the proposed model, the parameters values on the second experiment were set to get a performance close to the maximum possible for the CoHNOW. To do this two matrixes of 20,000 x 20,000 elements with 400x400 blocks were multiplied in 89 hours. The prediction precision for the stabilization performance was 92%.

From this analysis and model a methodology is being elaborated for adjusting the design parameters in order to get the maximum performance of CoHNOWS.

In order to increase LDGD Cluster reliability and efficiency, further work is required. It is part of this ongoing study increase the whole system's adaptability. To improve adaptability, is essential to adjust the whole methodology involved with LDGD Clusters, extending the methodology and model for n-clusters. Our idea is to maintain the user's application good performance no matter the chosen application. Also, it won't be necessary to the user become an expert in the application. Another idea is to extend the system to certain level that we could deal with more than two clusters. The network interconnection is an important aspect in any parallel machine; in a LDGD n-Cluster is not different. It is necessary need to develop more efficient techniques to improve the LDN usage. Therefore, multiple sockets combined with a multi-thread environment are necessary. The idea is to use the maximum capacity of the communication channel, in order to enhance the communication throughput. Fault tolerance is also a point to be attacked in this ongoing study.

5. Bibliography

[1] R. Buyya, High Performance Cluster Computing: Architectures and Systems, volume 1. Prentice Hall PTR. 1999.

[2] I. Foster, C. Kesselman, The Grid, Blueprint for a New Computing Infrastructure, pp. 15-51, Morgan-Kaufmann, 1999.

[3] O. Beaumont, F. Rastello and Y. Robert. "Matrix Multiplication on Heterogeneous Platforms", IEEE Trans. On Parallel and Distributed Systems, vol. 12, No. 10, October 2001.

[4] F. Tinetti, A. Quijano, A. Giusti, E. Luque, "Heterogeneous Network of Work8stations and the Parallel Matrix Multiplication", Euro PVM/MPI 2001, Y. Cotronis and J. Dongarra, eds., pp. 296-303, 2001.

[5] W. Gropp, E. Lusk, R. Thakur, Using MPI-2: Advanced Features of the Message-Passing Interface, Scientific and Engineering Computation Series, Massachusetts Intitute of Technology, 1999.

[6] A. Furtado, J. Souza, A. Rebouças, D. Rexachs, E. Luque, Architectures for an Efficient Application Execution in a Collection of HNOWS. In: D. Kranzlmüller et al. (Eds.):Euro PVM/MPI 2002, LNCS 2474, pp.450-460, 2002.

[7] A. Furtado, A. Rebouças, J. Souza, D. Rexachs, E. Luque, E. Argollo, Application Execution Over a CoHNOWS, International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications. ISBN: 0-9742059-0-7, 2003.

[8] W. Gropp, E. Lusk,N. Doss,A. Skjellum, A high-performance, portable implementation of the MPI message passing interface standard, Parallel Computing volume 22 number 6, pages 789-828, sep 1996

[9] Dongarra J., D. Walker, “Libraries for Linear Algebra”, in Sabot G. W. (Ed.), High Performance Computing: Problem Solving with Parallel and Vector Architectures, Addison-Wesley Publishing Company, Inc., pp. 93-134, 1995.

[10] M. S. Lam, E. Rothberg, M. E. Wolf, “The Cache Performance and Optimizations of Blocked Algorithms”, Fourth Intern. Conference on Architectural Support for Programming Languages and Operating Systems, Palo Alto CA, April 1999.