

Towards the Parallel Computing Based on Quality of Service

**Regiane Y. S. Kawasaki¹, Luiz Affonso H. G. de Oliveira²,
Carlos Renato L. Francês¹, Mauro Margalho Coutinho³,
Diego L. Cardoso¹, Ádamo Santana¹**

¹Laboratório de Computação Aplicada (LACA) - Universidade Federal do Pará (UFPA)
Caixa Postal 479 - 66.075-110 - Belém - PA – Brazil
Phone +55 91 2111302 - Fax +55 91 2111634

²Departamento de Engenharia de Computação e Automação, Centro de Tecnologia,
Universidade Federal do Rio Grande do Norte (UFRN),
59.072-970, Natal – RN - Brazil
Phone +55 84 2153771 - Fax +55 84 2153738

²Laboratório de Eletromagnetismo Aplicado (LEA)
Universidade Federal do Pará (UFPA)
Caixa Postal 479 - 66.075-110 - Belém - PA – Brazil
Phone +55 91 2111302 - Fax +55 91 2111634

**kawasaki@ufpa.br, affonso@dca.ufrn.br, rfrances@ufpa.br,
margalho@ufpa.br, diego@nautilus.com.br, alwkynew@hotmail.com**

Abstract. *Despite the existence of a series of applications that demand high performance computing, there are two strong restrictions to the effective use of parallel computing applications: (1) the high costs in the environments of parallel computing (typically, clusters and parallel machines) and (2) the dedicated character of these architectures (fact that generates a great degree of idleness of these platforms). This paper presents an architecture to provide parallel computing in traditional environments (non-dedicated), MPI_QoS calls, which is based on the concepts of quality of service (QoS) to assure low time responses necessary to this kind of computation.*

Keywords: *Parallel Computing, QoS, MPI, Real-time Linux.*

1. Introduction

In the current state of the art many areas of the human knowledge have demand for a great processing power. The range of these demands is sufficiently varied, such as in contemporary situations [1] like: studies of meteorological phenomena, analysis of finite elements, aerodynamics, modeling of oil reserves, genetic engineering, high performance web servers, among others.

However, many of the above cited activities already exist since a long time, however there was no computational power available to treat such problems. During the course of computers evolution, some factors can be considered determinant to generate the basis necessary to the increase of current processing power, among which there can be cited, for example, the evolution of microelectronics and the inclusion of deviation forecast heuristics in the normal sequence of the instructions flow [2].

Based on the first factor, the advance of microelectronics, many characteristics had been added even to the most traditional architectures (in the trivial sense). Today in personal computers there is a considerable number of registers, many levels of cache, great amount of main memory, high performance bars, pipeline with specialized units and input/output devices quicker and more reliable than its predecessors [3]. Due to all these improvements, personal computers have a processing power near to the one of the current RISC stations, being tenuous the very own delimitation of the RISC/CISC border.

Therefore, due to the cited evolution, it could be expected that the current computational power of isolated machines would already be enough for the solution of the great majority of problems initially pointed. Even so, there is a great number of specialized problems whose power of individual processing available is not enough. For these problems basically two possible solutions exist: (1) breaking the physical restriction of the light speed, through the use of "quantum computation", whose research still is in the threshold of its implementations; (2) paralleling the algorithm of the problem solution and running it simultaneously, what, ideally, should diminish the total processing time when compared with the sequential time [1]. For viability reasons, the second strategy is usually adopted for the solution of problems that demand high performance processing.

This paper presents an architecture proposed to allow high performance computation in traditional networks (non-dedicated), extending the possibility of parallel computing use and allowing a more favorable cost/performance relation. For

that, a feasibility study of the architecture is carried out by means of discrete simulation, on which the used data is based on the use of the real system, implemented in a laboratory of general use.

2. Traditional Architectures to Provide Parallel Computing

Parallel computing, in accordance with [3], consists on the division of one determined application in a way that it can be executed by many elements of processing that cooperate between themselves, aiming at the improvement of its efficiency through the break of the sequential execution paradigm of the instructions flow, dictated by the Von Neumann philosophy.

Parallel computing, in the current patterns, can be made in diverse environments. These environments usually are chosen in accordance with the nature of the parallel application to be executed. However, the determinant reason for choosing the type of parallel architecture is its cost. Many agencies, whether public or private, that need a powerful processing environment opt to not acquiring real parallel machines, called Multiprocessors or MIMD with shared memory. The high cost for attainment and maintenance of these machines is a determinant factor for the non-proliferation of this type of architecture.

An alternative to the high cost presented by the multiprocessors is the use of clusters of computers, that are formed by groups of computers that work linked in a network way, whose objective is to present a performance equivalent to the one of a real parallel machine [4]. Even being economically more viable and with a processing power close to the one of parallel machines, clusters of computers are still for specific intentions, used in dedicated way generating usually idleness of resources, what in the current state of the art is extremely undesirable.

A possible alternative to the reduction of the idleness, without harming the final cost of the architecture, is the use of computer networks of general purpose, usually installed in most of the organizations. However, some current technological restrictions make impracticable the use of this desirable solution. Some existing restrictions are: (1) the scheduling politics of the current operating systems that destine quantum for processes, independent of its priorities; (2) the used protocols do not give any support to differentiated services, and (3) the proper environments of message passing (as Virtual Parallel Machine - VPM and Message Passing Interface - MPI) do not generate its processes with quality of service standards.

Thus, to make parallel computing of non-dedicated form in networks of general purpose without interferences in its usual functioning, it is necessary to implement the philosophy of quality of service, QoS, as a concept end-the-end, from the generation of the parallel processes for the environment of message passing, followed by the treatment given to the processes by the local operating system, the differentiated transmission by the communication channel and finishing with the treatment in the operating system of the machine destination [5].

Given this overview, making parallel computing in networks of general purpose possible requires the proposal of a computational architecture involving differentiated concepts of message passing environment, real time operating systems and communication protocols that make possible the implementation of differentiated services. Despite all these peculiarities, proposed solutions should always be established in international standards.

This way, the proposal of this article is based on the insertion of QoS concept in the high performance computation, through the use of programming languages, operating systems and well-known communication protocols.

3. An Overview of Quality of Service

The Quality of Service (QoS) concept can be understood as a combination of demands from a network intending to achieve a better way to share the available bandwidth and guarantees of data transmission regarding four items: latency, jitter, bandwidth and reliability [6]. Latency is the delay that an application can tolerate in delivering a data package through the network; jitter is the latency variation in an end-to-end transmission; bandwidth is the maximum transmission rate at which an application's traffic must be carried by the network; and reliability is related to the routing, responsible to delay the packages transmission, modify its order or even discard them when the queues are full.

A network that handles QoS must provide mechanisms for traffic differentiation, prioritizing certain packages or data flows rather than others specially when an overflow in the network is evidenced. Those processes defined with higher priorities must continue their work perfectly, even though harming the performance of lower priority processes. Certain applications such as videoconferences and telemedicine are more susceptible to such parameters than others, like e-mail and file transfers.

The use and implementation of QoS is based on the models provided by the IETF (Internet Engineering Task Force), among them the Intserv and the Diffserv are the most known [7].

The Intserv assumes that the architecture of the current Internet does not need to be modified, but can be extended to offer some new services. It was projected to establish QoS "end-to-end", guaranteeing that the total quality will be offered exactly as was established in the original configuration, between two nodes that are connected through this system for reserving resources of the net for this connection. Some virtual connections are established as in Frame-Relay and ATM, and the routers stores in tables the state of each connection. The disadvantage of this architecture is the lack of scalability; in a big network the reserve of these resources can generate big traffic due to the great exchange of messages, making the architecture impracticable.

The Diffserv comes detaching due to its scalability and low cost. All its operation is based on marking a field of the heading of IP packages that was not used until then. This field of 8 bits, today called TOS (Type of Service) in the IPv4, and TCF (Traffic Class Field) in the IPv6 receives a numerical value that the routers use to know which treatment to give to this package. Its marking can be made using as a rule the address of origin or destination, origin door or destination door, schedule, MAC address, and others, besides being able to combine several of them.

4. MPI_QoS – A Proposal of Architecture for Parallel Computing with QoS

In this section, the structure currently in implementation is presented, whose tests are being carried in a production environment. The architecture, called MPI_QoS, is based on a set of open source software, making possible the use of a wider range basically for two reasons: (1) the used packages are of public domain, widely tested and with a good know-how available, (2) the gratuitousness of the involved packages must encourage in a more effective way the adoption of the proposed architecture.

It is known that usually programmers of parallel programs assume that all the necessary resources to the efficient execution of their programs are dedicated only and exclusively to their processes. However, in local networks as well as in WANs the dispute for shared resources, for example processors and I/O devices, can result in significant variations in the availability of these resources, causing adverse consequences to the global performance of the program.

An alternative for solving this problem is to provide parallel programs with hybrid routines that exchange messages in sets with parameters of QoS. These routines must guarantee the efficient execution of parallel processes, causing a similar performance to the execution of the same parallel process in a dedicated parallel architecture.

The concept of QoS is considered an end-to-end concept, therefore in this archetype, the quality of service is not provided only by the MPI (Message Passing Interface), but over all by the operating system, in this case Linux (more specifically the RT-Linux) that after having its kernel modified for provision of QoS, manages efficiently the parallel processes that request minors response times, without a bigger harm in the execution of the other conventional processes.

According to Figure 1, the QoS parameters supplied by the routines `Mpi_Send_QoS()` and `Mpi_Receive_QoS()` are received by the process scheduler of the RT-Linux. This scheduler analyzes the QoS parameters given and then sets a mechanism of appropriate scheduling such that parallel processes have a higher execution priority compared to other conventional processes.

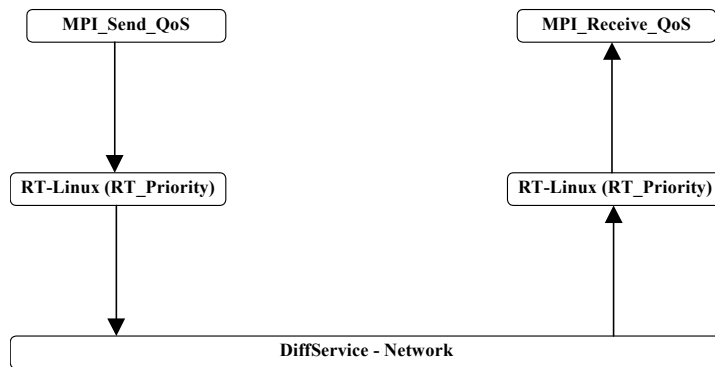


Figure 1. MPI_QoS Architecture

In general, when the process manager of the RT-Linux receives a service request, it must be capable to determine the resources necessary to reach the requirements of QoS of the applications and deciding to which scheduling class the application must belong. It is the process manager that makes the admission control, verifying the viability of the acceptance of the flow in a way that the requested QoS is guaranteed. Parallel processes must be privileged and scheduled to be executed for the CPU. Conventional processes must be scheduled also such that the average time of response and the reached outflow are acceptable to the user. Figure 2 presents the functioning of the process manager, which is based on multiple queues with priorities.

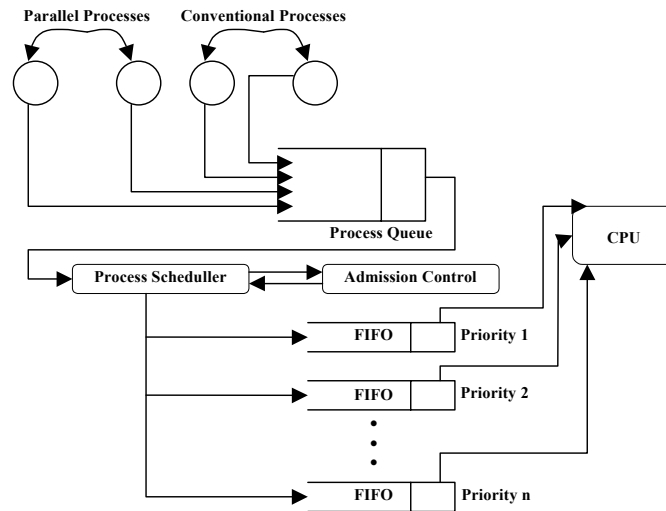


Figure 2. Process Scheduler of the MPI_QoS Architecture

When one of the parts (operating system and process) finishes using and/or providing the service, or the timeout specified at the moment of admission is expired, the service contract-closing phase begins.

5. Study of Feasibility Based on Discrete-Event Simulation.

This section describes a feasibility study carried out using NS (Network Simulator) [8], considering three situations described as follows:

- Situation 1: a dedicated network (without additional traffic), having only the processes of the parallel computation - as a parallel machine or a cluster;
- Situation 2: a network with typical traffic (non-dedicated), but with no strategy of QoS;
- Situation 3: a network with typical traffic (non-dedicated), but with an implemented strategy of QoS, based on the approach adopted in the MPI_QoS.

The inserted data in the simulation is based on the typical use of the Laboratory of Applied Computation, UFPA-Brazil, collected throughout a learning period (six months) adopting the average values. The performance measurement considered was the delay, due to the nature of parallel computing that requires that minimum response times.

- Dedicated Network (behavior of a cluster): Figure 3 presents three TCP flows started without traffic and QoS (considering typical applications of text and audio/video transferences). Due to the presented delay, this is the best possible situation. The only observed delay refers to the packing time of the TCP/IP protocol. Actually all the flows of generated data have the same behavior and the graph reflects this once it overlaps all three.

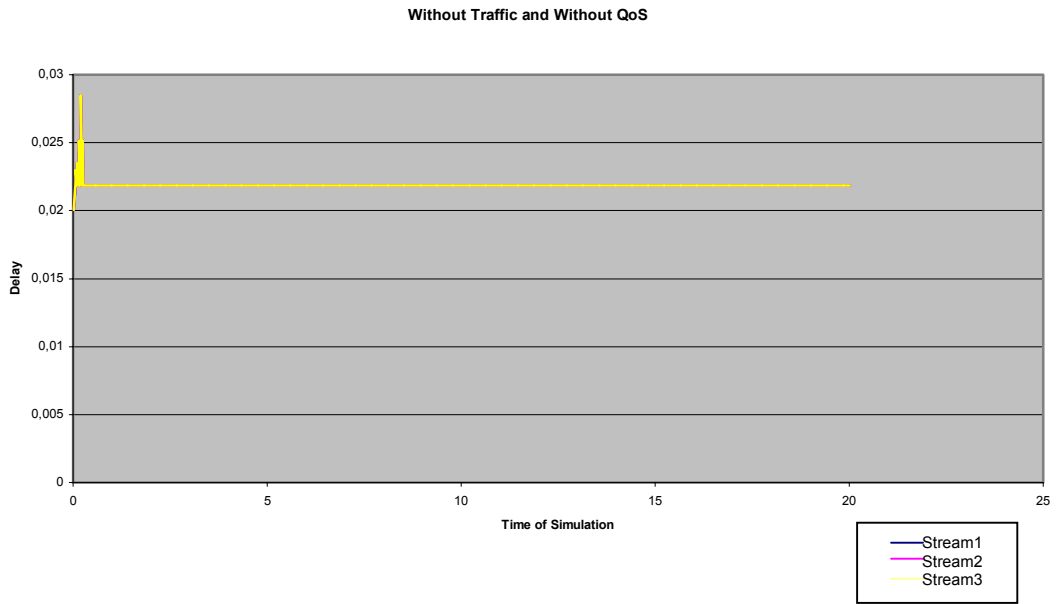


Figure 3. Delay obtained for the Situation 1.

- Non-dedicated net without QoS: Figure 4 reflects the traffic with congestion of background and without QoS. Due to the strategies of scheduling adopted by the operating systems and the lack of QoS mechanisms in the used protocols, the process of parallel computing is treated as an ordinary procedure, reflecting in the average delay attained for such applications. Adopting this strategy the response time in average duplicates for the case in study.

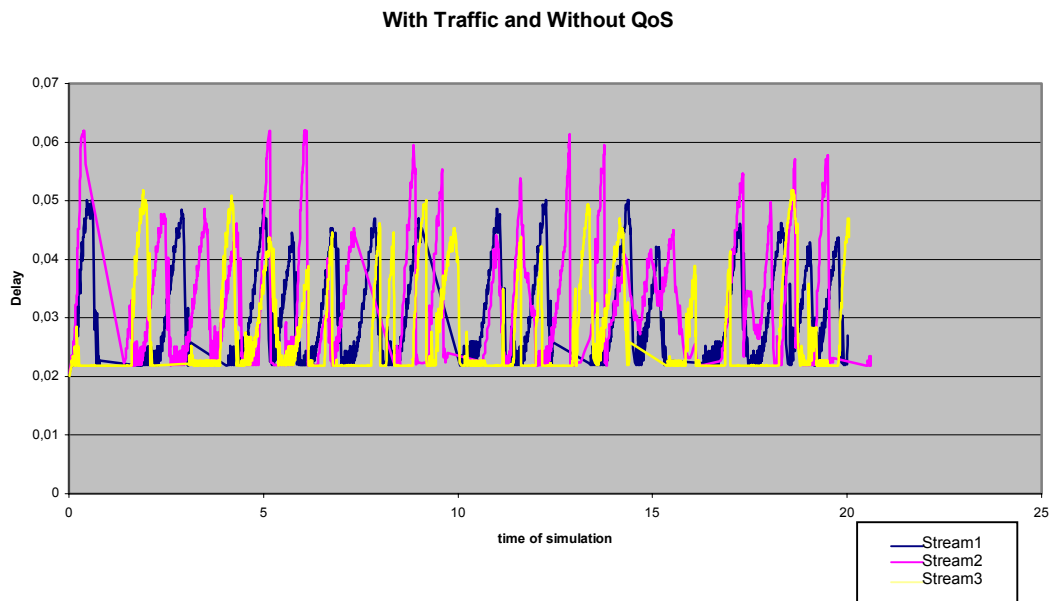


Figure 4. Delay obtained for Situation 2.

- Non-dedicated network with QoS: Figure 5 shows the situation with the same congestion applied in the second case, however here the QoS scheme adopted in the MPI_QoS architecture is used. The result is a delay almost similar to the situation that simulates the behavior of a cluster or a parallel machine, once the delays obtained for the processes of parallel computing using QoS is in the same order of the processes using the network in a dedicated way. This fact presents in the average a favorable cost/performance relation for the third situation.

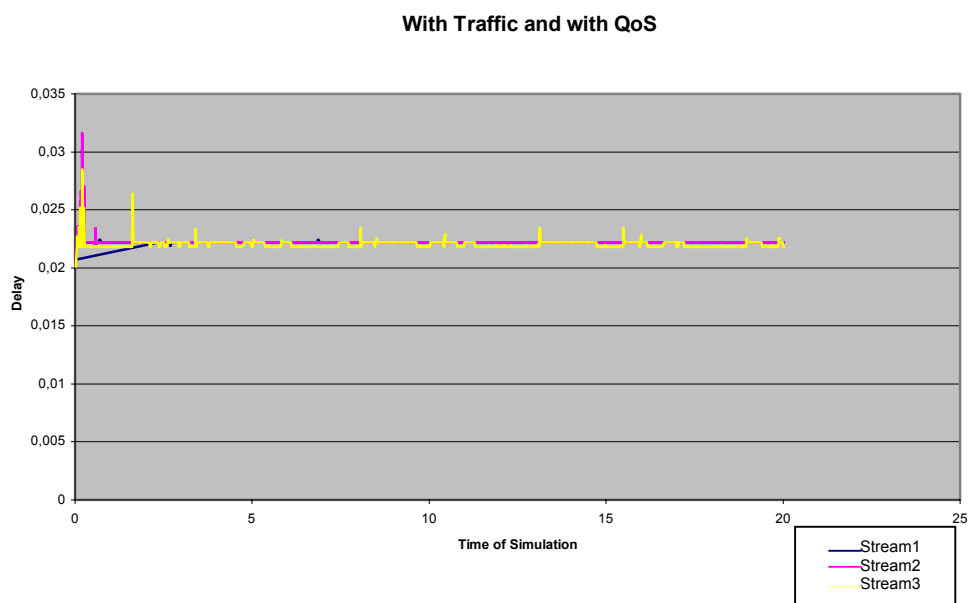


Figure 5. Delay obtained for the Situation 3.

All the simulations were executed, at the very least, thirty times, admitting themselves a confidence interval of 95%.

6. Related Works

Currently, there is a certain number of works directed to the insertion of QoS in diverse types and levels of computational systems. Some of these works are directed to the use of QoS in the message-passing environment MPI, such as MPICH-GQ [9] and MPI/RT [10]. The MPICH-GQ describes an architecture of message passing that uses mechanisms of QoS in MPI applications. It is part of the General-purpose Architecture for Reservation and Allocation (GARA) System. The GARA integrates different mechanisms of QoS, allowing the implementation of different types of services with QoS. Another work that introduces QoS in the MPI is the MPI/RT. It is based on real-time extensions to provide mechanisms with QoS in parallel applications.

The philosophy of the MPI_QoS is to join concepts of QoS in many levels considering an end-to-end approach: in the application (through the MPI), in the operating system (RT-Linux) and in the communication protocol (TCP/IP). Thus, the MPI-QoS, besides implementing a system directed to the QoS, also uses the easiness supplied for the operating system provided with QoS, using the figure of a process manager to guarantee a privileged access of parallel processes to the CPU.

Other works related to the provision of QoS in operational systems range from proposals of extension for operating systems of general purpose to projects of new operating systems, all focusing on mechanisms of resources management.

[11] presents the QoSOS architecture that provides adaptable QoS in the Linux operational system. However, this work only aims at supplying QoS in Linux, not considering the execution of parallel processes with QoS.

7. Final Remarks

This paper presents a feasibility study of the MPI_QoS architecture to provide parallel computing in environments with general purpose. The proposal brings some contributions, such as the ones highlighted below:

- Possibility of a larger use of parallel computing applications in networks of general purpose;
- Use of standards throughout the architecture (MPI, TCP/IP and Linux);
- Reduction of the computational platform implementation costs, with the use of gratuitous tools and open source code.

Recently new classes of applications have been defined and associated to available priorities in the RT-Linux. The idea is that the presence of parallel computing processes does not inflict any damage on the applications that are being executed in the network.

8. References

- [1] Foster, I., Kennedy, K., Dongarra, J., Fox, G. *Sourcebook of Parallel Computing*, Morgan Kaufmman Pub, 2002.
- [2] Andrews, G. R. *Foundations of Multithreaded, Parallel and Distributed Programming*. Addison Wesley, 2000.
- [3] Almasi, G. S., Gottlieb, A. *Highly Parallel Computing*. The Benjamin Cummings Publishing Company, 2nd. ed., 1994.
- [4] Spector, D. *Building Linux Clusters*. O'Reilly & Associates, USA, 2000.
- [5] Roy, A. et all. MPICH-GQ: Quality-of-Service for Message Passing Programs. Proceedings of the IEEE/ACM SC2000 Conference, November, 2000.
- [6] Xiao, X., Lionel, M. *Internet QoS: A Big Picture*, IEEE Networks, March/April, 1999.
- [7] ISO/IEC DIS 13236, *Information Technology – Quality of Service – Framework*, ISO/OSI/ODP, July 1995.
- [8] Fall, K.; Varadhan, K.; "The NS Manual"; Network Simulator 2.1b9a, VINT Project; 2002

- [9] Roy, A.; Foster, I.; Gropp, W.; Karonis, N.; Sander, V.; Toonen, B. "MPICH-GQ: Quality-of-Service for Message Passing Programs"; in the Proceedings of the IEEE/ACM SC2000 Conference; November, 2000.
- [10] MPI/RT Forum. <http://www.mpirt.org> (July 2003).
- [11] Moreno, M.; Gomes, A.; Soares, L.; *Provisão de QoS Adaptável em Sistemas Operacionais: O Subsistema de Rede*. XXI Simpósio Brasileiro de Redes de Computadores - SBRC2003. Natal, Brazil, Maio de 2003.