PERFORMANCE ANALYSIS OF THE CONNECTION MACHINE

E GELENBE

Rapporteur: M Packzad

Performance Analysis of the Connection Machine *

Erol Gelenbe

Ecole des Hautes Etudes en Informatique
Université René Descartes
45 rue des Saints-Pères
Paris, France

Abstract

This paper presents an analysis of the performance of the Connection Machine, with special emphasis on estimating the effect of its interprocessor communication architecture. A queueing model of the network architecture, including the NEWS and ROUTER networks, is used to compute the slow-down induced by message exchange between processors. Locality of the message exchanges is modelled by message sending probabilities which depend on whether a message is sent by a processor to another processor placed on the same NEWS network, or on the same ROUTER, or at a "remote" location which is only accessible via the ROUTER network. The specific slotted TDMA structure of the ROUTER Network communications is taken into account. The performance degradation of the Connection Machine as a function of the communication and architecture parameters is derived.

*Work supported in part by a grant from CNRS-C3

1 Introduction

A highly innovative and massively parallel computer system known as the Connection Machine [1] [1, 2] was introduced in the mid-eighties as a new tool for very rapid symbolic processing. Recent experience has shown that this architecture is not only of interest for symbolic or artificial intelligence based applications [14, 15], but that when it is equipped with a sufficiently large number of floating-point processors it is a highly effective tool for large scale numerical computations.

A simplified representation of the Connection Machine (which we designate by CM) is given in Fig. 1. The system presented is composed of 64 K (K stands for 1028) or 65 536 processors (P), with 16 K floating–point processors (FP). The system is partitioned into four subsystems of equal size, each of which is controlled separately via a Micro-Controller. The user interface of the system is a set of four host machines.

The CM architecture is perhaps the most novel massively parallel architecture among parallel systems which have been recently developed or proposed [3, 4, 5, 6, 7]. It is a 'logic-in-memory' architecture which does not physically separate the processors from the main memory. Thus, the usual memory to processor communication problem which is of crucial importance to conventional multiprocessor architectures, and which is dealt with by using high performance interconnection networks, does not exist here. On the other hand intense communication needs to take place between the processors, so that the CM is equipped with a sophisticated interprocessor network architecture.

In [1-9] various highly parallel architectures, including the Connection Machine [1,2,8] are described. Communication problems related to such architectures are discussed in [10], while [11-15] present various applications which have been run on the Connection Machine. In [16] the effect of communication on the performance of the Connection Machine is discussed and a queueing network model approach is suggested.

A simplified CM processor interconnection structure is shown in Fig. 2.

The key to understanding the architecture of the CM is the communica-

[1]The term Connection Machine is a registered trade-mark of Thinking Machines Corporation.

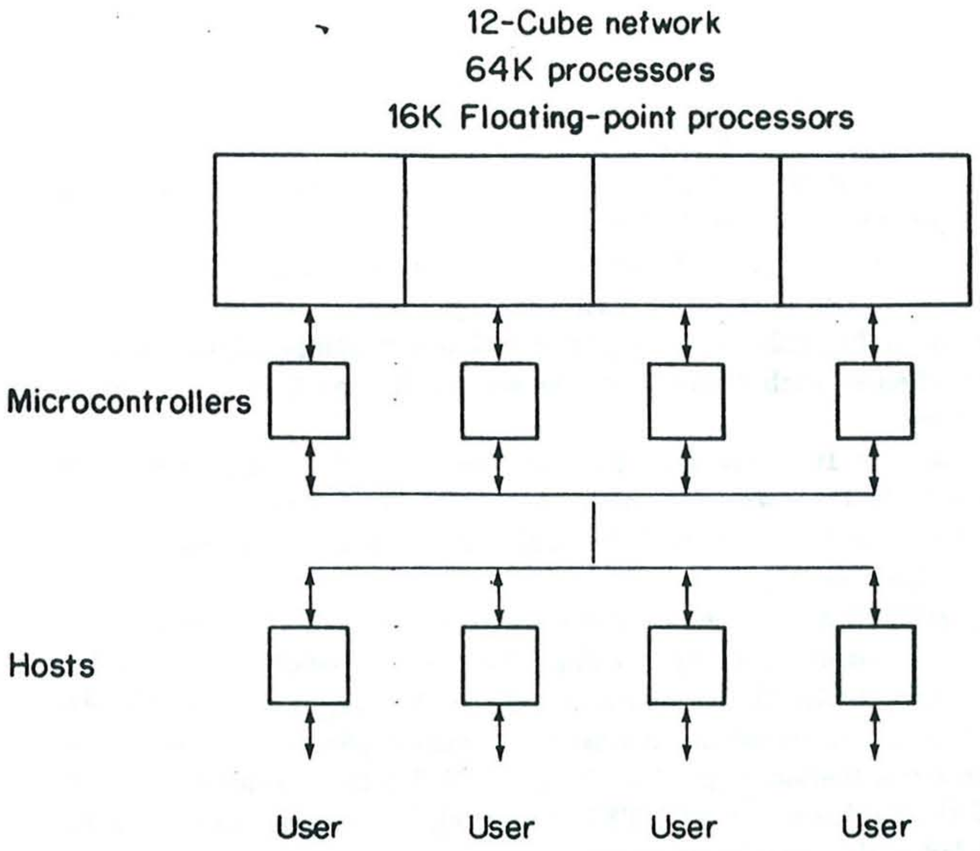

Figure 1: General structure of the Connection Machine with four hosts and 64 K processors

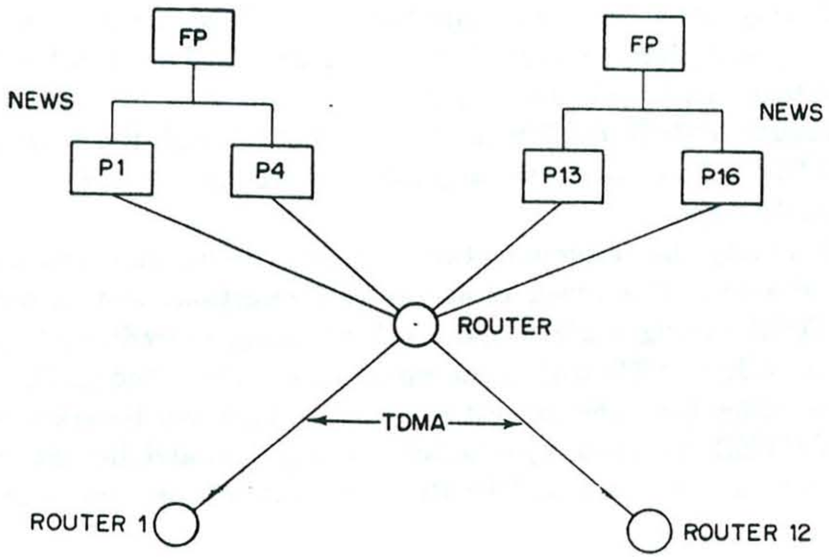

Figure 2: Connection Machine processor interconnection structure

tion mechanism used to interconnect the processors. Processors are interconnected in groups of four (P1,P2,P3,P4, or P13,P14,P15,P16, etc.) in a nearest-neighbour network called the NEWS (North–East–West–South) Network. A more complex network called the ROUTER Network is then used to provide a communication path between any pair of processors.

In Fig. 2 we have shown a system in which a floating-point processor (FP) is attached to each NEWS network so that it may be shared among four processors.

Each group of 16 processors, for instance P1 to P16, are connected to the same ROUTER which links them to the rest of the system, and in particular to any other ROUTER, via a 12-Cube network which implements the ROUTER Network.

Each ROUTER is connected to twelve other ROUTERs via bi-directional links. Communication from any ROUTER to these twelve other ROUTERs is carried out in TDMA (Time-Division-Multiple-Access) mode. The TDMA ROUTER Network operates in communication cycles, and during any communication cycle (called a 'petit cycle'), a ROUTER can send a message to *each* of the twelve other ROUTERs to which it is connected, using 12 slots included within one 'petit cycle'.

For a 64 K (65 536) processor machine there will be 4 K (4096) ROUTERs, since 64 K $= 16 \times 4$ K and each ROUTER is connected to 16 processors. Notice that $4096 = 2^{12}$, so that the 12-cube interconnection network allows the connection of all the 4 K ROUTERs [9].

During one single communication cycle, if free time slots are available in the appropriate position of the TDMA algorithm used by the network, a message can travel from any ROUTER to any other ROUTER, passing via intermediate ROUTERs as it progresses through the network. Thus all 4096 ROUTERs are potentially fully interconnected during a single communication cycle.

In reality the communication time may be larger because of traffic in the network. If a given dimension (or direction) slot is occupied at a ROUTER during a given cycle, other messages needing the same direction at a ROUTER will queue up and be transmitted in first-come-first-served order (i.e. the longest waiting message will be transmitted first). A ROUTER may also reject a new message just arriving into the network because its buffer is full. The ROUTER network is guaranteed to deliver messages once they have entered the network using a store-and-forward procedure.

On the other hand, the NEWS network handles local communication between neighbouring processors; it thus reduces the load on the ROUTER network and provides fast local communications between four neighbour processors.

Each elementary processor P is a variable-length operand ALU (arithmetic and logical unit), though this may change in future versions of the machine, and it has a 64 K bit memory. In fact each processor may support up to four resident processes; when this is done throughout the machine, the CM appears as if it is partitioned among four users. Each processor's memory contains a data area and a stack of 1 K bits. Floating-point operations emanating from any one of a set of four processors will be forwarded to the FP for execution without travelling through the ROUTER Network (see Fig. 2). Clearly, each of ROUTER 1 to ROUTER 12 of Fig. 2 will have the same interconnection structure as the ROUTER placed in the middle of the figure.

Each of the processors executes 'nano-instructions' which are broadcast to all processors via the micro-controllers (see Fig. 1). These nano-instructions are generated from macro-instructions which are received by the hosts. Individual processors can be instructed to mask out certain instructions so that it is possible to assign certain computations to certain processors.

Nevertheless, due to the difficulty of addressing a wide variety of instructions to individual processors, the CM architecture is most appropriate for handling SIMD (single-instruction-multiple-data) type parallelism. Thus a stream of instructions is sent from the hosts to a large number of processors simultaneously. Data can be transmitted initially from the hosts to the processors, and intermediate results will then flow between processors via the NEWS and ROUTER networks as the computation progresses.

Clearly, the CM appears to be well adapted to situations in which a small number of distinct instructions have to be executed on a very large data set. Typical examples of this type of computation include operations on very large matrices, image processing, computer vision, etc.

The purpose of this paper is to examine the effect of communications between processors on the effective processing power of the Connection

Machine. The nominal processing power is the number of instructions executed per unit time assuming that all processors are simultaneously busy processing instructions. On the other hand, the effective processing power is the rate of instruction execution when communication slowdowns due to the exchange of messages is included. We assume that the instructions executed, if any, for sending and receiving messages are part of the 'normal' work of a processor, and that therefore these do not contribute to the communications slowdown. However, we obviously consider that the time spent by the processors waiting for messages to arrive from other processors constitutes a source of reductions in the processing power of the machine. Therefore in this paper we introduce an analytical methdod to evaluate this reduction. We then present numerical examples to illustrate this effect. Of course, these numerical examples are based on certain parameter choices, such as the nominal instruction execution time of a CM processor, the number of instructions executed between two successive message exchanges, the characteristics of the network, the use of local communications (via the same ROUTER on the network, or within the same NEWS network), etc. Thus these parameters may vary from one application environment to the other. The methodlogy we propose can be used with other parameter values than those which we introduce in our examples, and our numerical examples are proposed essentially as an illustration of the results.

Since the key element of the CM is the ROUTER Network, in Section 2 we present a simple analytical method for evaluating the delay it introduces. The method will be based on an analysis of its performance using existing analytical results in queueing theory. In Section 3 we will present a theoretical evaluation of the CM processing capacity using the results of Section 2, and a workload model which takes into account the aspects such as the locality of communication patterns among processors.

2 ROUTER Network Performance

The structure of the ROUTER Network (RN) has been presented in the previous section. In this section we discuss its performance using a queueing model based on the 'server with vacation times' [17,18].

A message to be sent through the RN will be considered to be a data element which can enter the RN via some ROUTER i and leave it from some ROUTER j in at most 12 communication cycles, or in just one cycle if none of the slots which the message requires as it progresses through the network are already occupied. Notice that because we are dealing with a hypercube, there may be several paths between i and j which are of the same length. We shall denote by k(i,j) the shortest distance, in number of hops, between ROUTER i and ROUTER j and by π the average distance traveled by a message between any two ROUTER's in the ROUTER Network.

Let T be the duration of a communication cycle and by S the length (in time units) of a slot, so that $T = 12S$. A message arriving at a ROUTER and wishing to proceed to some other ROUTER to which it is directly connected (by a single link) observes the following service structure. If the message arrives at an instant when no other message which has arrived before it is directed to the same next ROUTER, it will wait for some time, which will vary between 0 and T, until the appropriate slot comes up; it will then be transmitted during a duration of length S to the appropriate ROUTER. If on the other hand the message finds other messages already waiting for the *same* slot when it arrives, it will be queued until it arrives to the head of the queue and will then be transmitted.

We neglect in our analysis the case where the ROUTER buffer is full so that the message cannot enter its queue. Though this situation is possible in theory, we assume that it is sufficiently unlikely to occur so that it may be neglected. Therefore it will be assumed that all queues considered are of infinite capacity, and blocking at queues is neglected.

The behaviour of the queue of messages at a ROUTER which are directed towards the same neighboring ROUTER (hence using the same slot) is a special case of the queue with autonomous service or of walking type (also known as the queue with server vacations), which is known in the queueing theory literature [17,18]. We shall use a formula derived in [18] to analyse it.

According to these results, and under the only assumption that the arrival of messages to a ROUTER whose destination is a neighbour ROUTER constitute a renewal process (i.e. interarrival times are independent and identically distributed random variables), we can write the following formula for the steady-state *response time* W experienced by the messages (where W denotes the random variable):

$$W = V + T^{*} + S \tag{1}$$

where T^{*} is a uniformly distributed random variable in the interval $[0, T]$, V is the waiting time which will be detailed below, and V and T^{*} are independent random variables.

In this formula, V denotes the *waiting time* in a simple queueing system with the same arrival process as the system considered, but with constant service time T; note that the waiting time of a message is defined as the time it waits in queue before receiving service, while its response time is its waiting time plus the transmission time of the message once it has arrived to the head of the queue. As a consequence of (1), the *average response time* of a message is obtained as:

$$E[W] = E[V] + \frac{1}{2}T + S \tag{2}$$

Since any given ROUTER receives messages from its twelve neighbouring ROUTERs the arriving message traffic can be expected to be quite random in nature. We shall therefore assume that it is Poisson, since it is the superposition of 12 different and independent arrival processes (it is well known that the Poisson arrival process can be obtained by the superposition of a large number of independent arrival processes). V is then the waiting time of an $M/D/1$ queue (Poisson arrivals and constant service times) and $E[V]$ is obtained from the well-known formula [17]:

$$E[V] = \frac{\lambda T^{2}}{2(1 - \lambda T)} \tag{3}$$

where λ is the number of messages arriving per time unit from the ROUTER Network (RN) which are directed to the particular ROUTER being considered. Finally we have the average response time at the ROUTER:

$$E[W] = \frac{T}{2}\left[1 + \frac{\lambda T}{(1 - \lambda T)}\right] + S \tag{4}$$

In order to obtain a measure of the performance of the RN as a whole, we have to know what the average path length Π is for messages entering the network. Clearly we may have a path of length one, if the message leaves the network directly after the first neighbour of the ROUTER from which it entered, or it may be as large as twelve.

For the time being we shall not consider the messages exchanged locally via the ROUTER being considered, between the processors to which it is directly connected. These, and the effect of the NEWS network, will be considered later.

Now assuming that all ROUTER nodes are equivalent with respect to the traffic they carry, we have the RN average response time R to a message which enters it:

$$R = \frac{\Pi T}{2}\left[1 + \frac{\lambda T}{(1-\lambda T)}\right] + \Pi S \qquad (5)$$

Of course, the *total* traffic coming into ROUTER from the RN will be 12λ on the average if all links carry the same load.

Let us now turn to the traffic *offered* by the processors. Let λ_N be the traffic (always in messages per unit time) which *one* processor sends to the other processors on the same NEWS network. Clearly λ_N will not enter the RN. Let λ_n be the traffic which a processor sends to the twelve other processors which are connected to the same NEWS network (see Fig. 2); this traffic may cause congestion at the ROUTER, but it will also not enter the RN and therefore it does not contribute to λ. Finally, we consider the traffic λ_r emanating from a processor and directed to any one of $16 \times (4096 - 1)$ other processors and which can travel from a given ROUTER to a final destination at any one of the remaining 4095 other ROUTERs. Thus a ROUTER will receive on the average, from the processors to which it is directly connected, a traffic of $16\lambda_r$ messages per unit time for transmission over the RN.

Let us examine the relative importance of the quantities involved. In order to optimize the performance of the CM, it is reasonable to assume that a user would attempt to organize his application so thatfor a given value of $\lambda_N + \lambda_n + \lambda_r$:

- λ_N (the local traffic on the NEWS network) is as large as possible
- then $\lambda_N + \lambda_n$ is as large as possible

- λ_r is as small as possible since it is the traffic which will have the greatest delay
- Π is as small as possible

Furthermore an attempt will be made in the application to share the load equally among ROUTERs, in order to reduce average response times, by using equitable routing through the RN; thus it is reasonable to assume that the traffic λ carried on all neighbouring links (i.e. 'slot' traffic) is the same. The assumptions may not hold in each particular case but they are valid on the average for a carefully organised application.

We can now relate the link traffic λ to the preceding parameters. Since all traffic emanates from the processors, each ROUTER will receive on the average $16\lambda_r\Pi$ messages per unit time, including messages in transit and fresh incoming traffic from the processors. Assuming that all 12 links are equally loaded we shall have:

$$\lambda = 4\lambda_r\Pi/3 \tag{6}$$

As a consequence and after some simplifications the average response time to messages entering the RN becomes

$$R = \Pi T\left[\frac{7}{12} + \frac{2\lambda_r\Pi T}{(3 - 4\lambda_r\Pi T)}\right]$$

where we have used the fact that $S = T/12$. We see that in order to avoid saturation we must have

$$\lambda_r < \frac{3}{4\Pi T}$$

as the stability (or non-saturation) condition on the average number of messages per time unit that a processor sends into the RN.

The approach we propose here yields, rapidly and in closed form, the average response time or average network traversal time R for the RN. If we choose the time unit to be the duration of the communication (or 'petit') cycle T, i.e. $T = 1$, the formula for R becomes

$$R = \Pi\left[\frac{7}{12} + \frac{2\lambda_r\Pi}{(3 - 4\lambda_r\Pi)}\right] \tag{7}$$

This formula will be used in the sequel for the evaluation of the performance of the CM.

If we denote by Λ the total message traffic emanating from a single processor

$$\Lambda = \lambda_N + \lambda_n + \lambda_r$$

then the proportion $f_r = \lambda_r/\Lambda$ of messages sent to other processors via the RN will be an important factor in the performance of the CM. Similarly, the parameter Π will also play an important role.

Indeed let ν be the average message delay over the NEWS network, and γ the average message delay for the local interprocessor messages which do not enter the NEWS network or the RN, then the overall average delay D or response time for a message from a processor to its destination will be

$$D = f_N \nu + f_r R + \gamma(1 - f_N) \tag{8}$$

where $f_N = \lambda_N/\Lambda$, because *all* messages which *do not* travel through the NEWS network must proceed to the initial ROUTER and then from there to another processor incurring a delay γ; this is not the case for messages which remain on the NEWS network.

The parameters f_N and $f_n = 1 - f_N - f_r$ characterize the computations which only require communications among neighbouring processors; this provides a quantitative characterization of the concept of 'local sphere of computation' introduced in [16].

3 Highly Balanced Computations

Intuitively speaking one can expect that the CM performance will be maximized for computations in which each processor carries out the same computational step synchronously with as little communication as possible. The small amount of communication should be carried out among processors which are very close. An example of such a computation, represented by a computation graph, is shown on Fig. 3.

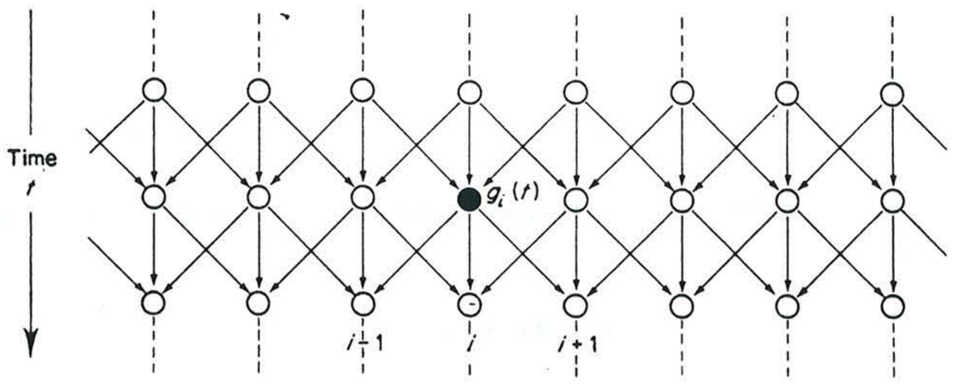

Figure 3: A simple highly balanced computation graph

In this example, which can represent, for instance, the numerical solution of the heat equation in one-dimensional space and in time :

$$\frac{\partial g}{\partial t} = a\frac{\partial g}{\partial x} + b\frac{\partial^2 g}{\partial x^2}\ , \qquad g \equiv g(x,t)$$

each processor handles the computation of $g_i(t)$ for some i, where h is the discretization step of x, and

$$g_i(t) \approx g(ih,t)$$

Thus in the example of Fig. 3 processor i computes $g_i(t)$ for discretized time; it receives messages from processors $i-1$ and $i+1$, and sends messages to processors $i-1$ and $i+1$ after each computational step. Other more complex examples of the same general form are frequent in numerical algorithms.

More generally, we shall consider a computational scheme in which that each processor computes for a time C, after which it sends messages for some time m, and then waits for messages for some time M. We shall assume that at each step it sends L messages so that

$$\Lambda = L/(C+m+M)$$

Even for the very simple example of Fig. 3 we see that all messages cannot be sent on the NEWS network; indeed each neighbouring group of four processors will send eight messages (we do not consider the information which the processor needs from *itself* and which is represented by the vertical arrow) of which only six can remain on the NEWS network; we thus have $f_N = 0.75$. Similarly we can see that $f_r = 2/32 = 0.0625$.

Assuming perfect synchronization between processors, we shall have M given by the formula for D in (8):

$$M = \gamma + f_N(\nu - \gamma) + f_r\Pi \left[0.583 + \frac{2\Lambda f_r}{(3 - 4\Lambda f_r\Pi)}\right] \tag{9}$$

This equation in fact has the variable M on both sides because Λ is a function of M; it is quadratic in M so that we can solve it easily. Once this is done, we can obtain the processing power η defined as 'Number of Instructions Executed per Unit Time' for the CM on this application as follows. If I instructions are executed in time C by each processor, and if P processors are used by the application (e.g. $P = 16$ K or 64 K), then the processing power η of the CM for this application is given by the formula:

$$\eta = \frac{I.P}{C + m + M} \tag{10}$$

where C will of course increase with I, though it will depend on the type of instructions which are being executed in the application. We can use M obtained by substituting (9) in (10) as a function of the parameters of the architecture, namely P, γ and ν, or of those which depend on the architecture of the CM and on the properties of the application such as I, C, m, L, f_r, f_N, Π.

3.1 The lightly loaded ROUTER network

For the case when the RN is lightly loaded, i.e. $\Lambda f_r \approx 0$, we have from (9):

$$M = \gamma + f_N(\nu - \gamma) + 0.583\Pi f_r$$

For the sake of simplicity we shall assume that the message delay on the NEWS network is the same as that for messages which go through the ROUTER without passing through the RN: $\gamma = \nu$, so that

$$M = \gamma + 0.583\Pi f_r$$

We then have from (10):

$$\eta = \frac{I.P}{\alpha I + m + \gamma + 0.583\Pi f_r} \tag{11}$$

where we have taken $C = \alpha I$ where α is the nominal time needed to execute an instruction. Recall that this formula includes the delay at the RN but assumes that no queues form at the ROUTERs.

In [8] it is indicated that the time necessary to execute a 32 bit add instruction on the CM-2 is 21 μs; therefore we shall choose $\alpha = 21$ if the unit time T is taken to be 1 microsecond.

Let us first assume that $f_r = 0$, so that no messages are being sent in the RN and it has no effect on performance. we then have that the processing power is given by

$$\eta' = \frac{P}{\alpha + (m + \gamma)/I} \tag{12}$$

Therefore for P=16 K and $\alpha = 21$, the processing power η' varies between 780.2 MOPS when $(m + \gamma)/I \ll 0.1$ and 712.34 MOPS when $(m + \gamma)/I$ is equal to 2. We see that though this peak performance is sensitive to local communications, the effect is relatively moderate.

In Fig. 4 we plot η' against $(m + \gamma)/I$ for $\alpha = 21$.

Let us now consider the effect of the RN. To do so we take the ratio η/η' to examine the reduction in performance due to the RN:

$$\frac{\eta}{\eta'} = \left[1 + \frac{0.583\Pi f_r}{\alpha I + m + \gamma}\right]^{-1} \tag{13}$$

The above relation allows us to correct the curves given in Fig. 4 in order to take into account the slow-down introduced by the RN.

On Fig. 5 we plot η/η' as a percentage, for $I = 1$, $I = 10$ and $I = 100$ instructions, $\alpha = 21$, as well as for $\alpha = 1$ and $\alpha = 0.1$, against Πf_r which varies from 0 to 12 (its largest possible value when Π=12 and f_r=1).

These two last values of α are of no practical interest today, but serve to indicate the degradation of the RN on a hypothetical future very fast CM.

We see that for $\alpha = 21$, which corresponds to the 21 microsecond instruction execution time, the RN has no practical effect on performance

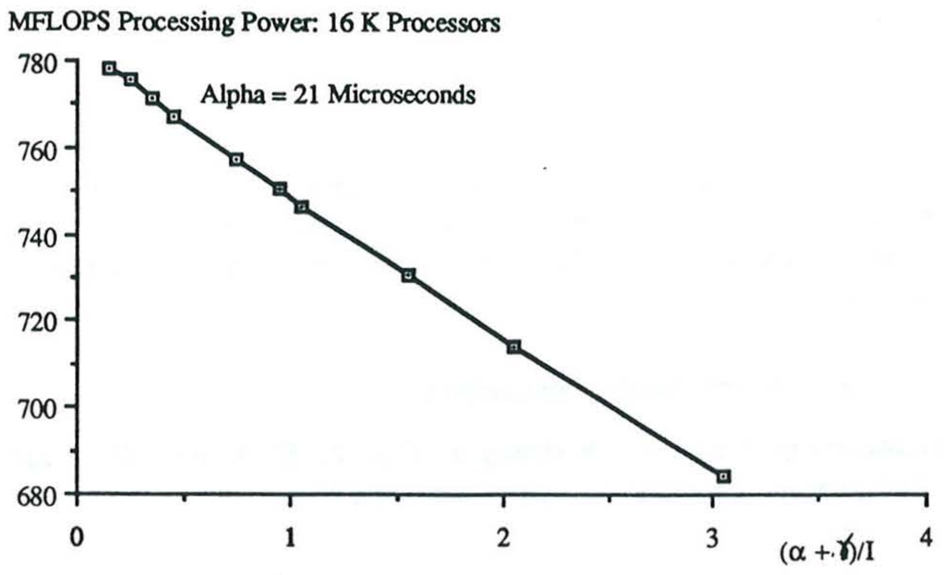

Figure 4: η' processing power in MOPS of the Connection Machine without ROUTER network access

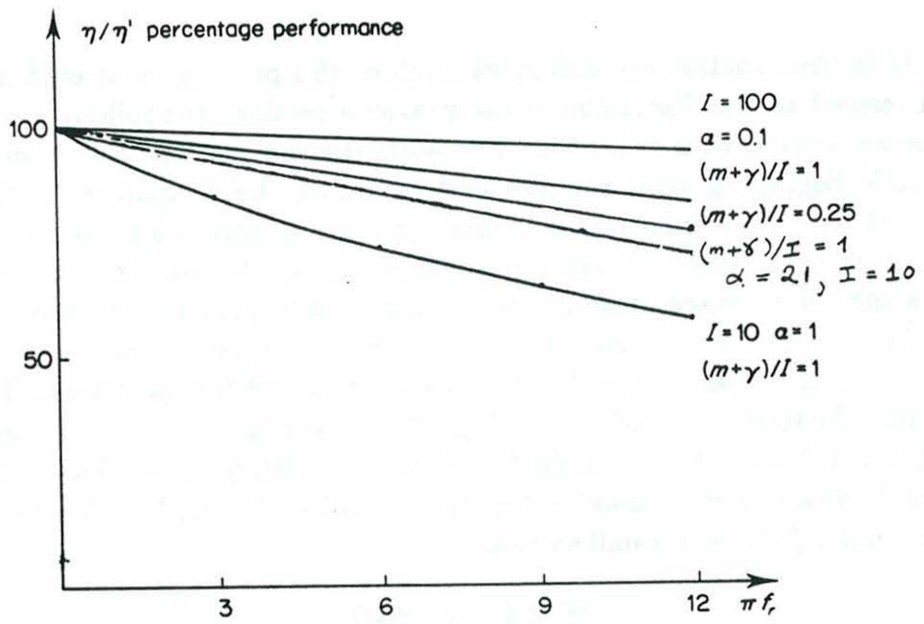

Figure 5: Percentage reduction in processing power of the CM due to the RN slow-down without queueing at the ROUTER

degradation since η'/η is reduced at most by 3 percent if I is larger than ten instructions. The degradation only becomes significant for a small value of I if we had a much faster machine with 1 microsecond ($\alpha = 1$) instruction execution times.

3.2 A simple concrete example

Consider the computation graph shown on Fig. 3. Each individual task, executed on processor i at some time t, takes the form:

```
begin repeat indefinitely
wait for messages from $i-1$ and $i+1$;
        receive messages
        $g_i(t) \leftarrow G[g_{i-1}(t)$, messages];
        send messages to $i-1$ and $i+1$;
end.
```

Here G is the function computed by each of the processors at each step. With respect to the discussion of the previous section, the following times can be associated with this code. The **wait for** instruction takes time M. The code beginning with **receive** and including the assignment $g_i(t) \leftarrow G[.]$, will execute I instructions taking (on the average) αI time units . The **send** instruction will take time m, including the time necessary for processing all message sending but excluding the principal transmission time; indeed, we assume that all messages sent are guaranteed to arrive so that a processor does not need to wait to be informed that a message it has sent has effectively arrived. From Fig. 3 we have$f_N = 0.75$, $f_r = 0.0625$ and $L = 2$ Taking unit time to be $T = 1$ microsecond, $\gamma = 1$, and $\alpha = 21$ (so that we have a 21 microsecond instruction execution time), $I = 100$, $m = 1$, we see that Λf_r is very small so that

$$M \cong 1 + 0.0364\Pi$$

For this application with some care it should be easy to implement the application so as to obtain $\Pi = 1.5$; indeed for each message entering the

RN, it seems possible to address it either to a ROUTER at a single hop distance or a two hop distance. We then have $M \approx 1.05$. From (11) we have $\eta \approx 761.1$ MOPS for a 16 K processor CM.

4 Conclusions

In this paper we have considered the performance of the Connection Machine architecture. Special emphasis has been placed on the performance degradation which may be expected due to communication delays between processors. Hence special attention has been placed on the architecture of the Connection Machine communication network including the NEWS and ROUTER networks. In particular, the TDMA protocol of the ROUTER Network has been modelled using a queueing model with vacation times. Both local communication, among processors connected to the same NEWS network, and remote communication via the ROUTER network has been considered. This has lead us to present a quantitative characterization of the concept of "local sphere of computation" introduced for the CM in [16].

The communication delay between processors has been computed from a queueing model and introduced into the model of the global architecture. This has yielded a formula for the performance degradation of the CM processing power as a function of communication locality and as a function of parameters of the architecture. A simple example of a SIMD computation has been provided in order to illustrate the methodlogy we have have developed.

Our example shows that with some care, and assuming parameters currently annouced for the CM, this degradation can be quite small as long as the number of arithmetic operations I executed between successive communication steps exceeds 10. For lower values of I the degradation can become quite significant important.

5 References

[1] Hillis, W.D., 'The Connection Machine', MIT Press, Cambridge, Mass., (1985).

[2] Hillis, W.D., 'The Connection Machine: A Computer Architecture Based on Cellular Automata', *Physica*, 10, 213–228, (1984).

[3] Frenkel, K.A., 'Evaluating Two Massively Parallel Machines', *Communications of the ACM*, Vol. 29, no. 8, (August 1986).

[4] Batcher, K.E., 'Architecture of a Massively Paralllel Processor', *Proceedings of the 7th Annual Int. Symp. on Computer Architecture*, La Baule, France,(May 1980).

[5] BBN Laboratories, 'Butterfly Parallel Processor Overview', *BBN*, Cambridge, Massachusetts, (December 1985).

[6] Intel Corporation, *iPSC System Overview*, (October 1985).

[7] Flanders, P.M ., Hunt S.F., Reddaway, S.F. and Parkinson D., 'Efficient High Speed Computing with the Distributed Array Processor', *Proceedings of the Symp. on High Speed Computer and Algorithm Organization*, University of Illinois, Academic Press, (1977).

[8] 'Connection Machine Model CM-2 Technical Summary', *Thinking Machines Technical Report HA87-4*, Thinking Machines Corporation, Cambridge, Mass., (April 1987).

[9] Broomell, G. and Heath, J.R., 'Classification Categories and Historical Developement of Circuit Switching Topologies', *Computing Surveys*, Vol. 15, (June 1983).

[10] Levitan, S.P., 'Measuring Communication Structures in Parallel Architectures and Algorithms', in *The Characteristics of Parallel Algorithms*, pp. 101–137, Jamieson, L.H., Gannon, D.B. and Douglas, R.J. (eds), MIT Press, Cambridge, Mass.,(1987).

[11] Little, J.J., 'Parallel algorithms for Computer Vision on the Connection Machine', *AI Memo 928*, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, Cambridge, Massachusetts, (November 1986).

[12] Flynn, A.M. and Harris, J.G., 'Recognition Algorithms for the Connection Machine', *Proceedings of the 9th Int. Joint Conf. on Artificial Intelligence*, Los Angeles, CA, 57–60, (August 1985).

[13] Harris, J.G. and Flynn, A.M., 'Object Recognition Using the Connection Machine's Router', *Proceedings IEEE 1986 Conf. Computer Vision and Pattern Recognition*, 134–139, (May 1986).

[14] Stanfill, C. and Kahle, B., 'Parallel Free Text Search on the Connection Machine System', *Communications of the ACM*, Vol. 29, No. 12, (December 1986).

[15] Stanfill, C. and Waltz, D., 'Toward Memory-based Reasoning', *Communications of the ACM*, Vol. 29, No. 12, (December 1986).

[16] Upton, R.A. and Tripathi, S.K., 'On the Performance Evaluation of Fine-Grained SIMD Computer Architectures: an Analysis of the Connection Machine', *High Performance Computer Systems*, Gelenbe,E. (ed.), Elsevier Science Publishers, North-Holland, Amsterdam, (1988).

[17] Gelenbe, E. and Mitrani, I., *Analysis and Synthesis of Computer Systems*, Academic Press, London and New York,(1980).

[18] Gelenbe, E. and Iasnogorodsky, R. 'A Queue with Server of Walking Type', *Annals de l'Institut Henri Poincaré, Série B (Probabilité et Statistiques)*, Vol. XVI, No. 1, 63–73, (1980).

DISCUSSION

Rapporteur: Mustapha Packzad

After the talk Professor Gelenbe was asked by Professor Randell about the gains made by this particular analysis of the Connection Machine which differ from other similar analyses carried out by others. Professor Gelenbe replied that there have been no previous analyses of the Connection Machine: experiments have been carried out on the machine, but they have not been analysed. There have been many benchmarks but this is the first analysis He added that "it is very hard to match figures with figures"because of the large numbers of parameters involved in the measurements such as machine parameters. Some of these parameters are hard to discover. What the analysis does confirm is that performance reduction is substantial with respect to the number of components.

Another participant asked how the results of the analysis compare with experimental results and whether the results are optimistic. Professor Gelenbe replied that the order of magnitudes shown by the results of the analysis is comparable with those obtained by experiments. The answer to the second part of this question was that the results are optimistic because of the assumptions of perfect SIMD and that communication is not synchronised. The results would be much worse if the communications were synchronised. This is consequently a best-case analysis rather than a worst-case analysis.

A participant commented on the fact that some manufacturers present performance curves which are near-linear with increasing numbers of processors, but do not mention that there would be a fall in the performance if the curve was to be extended beyond the number of processors shown. Professor Gelenbe replied that with 4K processors and interconnection networking the performance is really quite good.