RC24936 (W1001-073) January 25, 2010
Computer Science

IBM Research Report

Conditions for Scalability of Mesh-connected Massively Parallel Processors

José E. Moreira
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Conditions for Scalability of Mesh-connected Massively Parallel Processors

José E. Moreira
IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598
`jmoreira@us.ibm.com`

Abstract

Several metrics are used when comparing processing elements of different massively parallel processors. The most common being computational performance (flops/second), memory capacity (bytes) and communication performance (bytes/second). We show that for an important class of problems, namely three-dimensional simulation in real space performed on mesh-connected massively parallel processors, it is the relationship between the metrics, rather than their individual values, that determine scalability. We show that processing elements with different metrics can result in the same scalability behavior at the machine level. In particular, we show how one can trade memory capacity for communication performance and vice-verse. We establish a new metric, that we call the "quality" of a processing element that allow us to compare the scalability behavior of different systems.

1 Introduction

Mesh-connected (and variants like torus-connected) massively parallel processors are among the most successful supercomputers. Examples include the older Cray T3D [4] and T3E [5] machines as well was the more recent Cray XT (including the XT3, XT4 and XT5) [2] and Blue Gene (including Blue Gene/L and Blue Gene/P) [3, 1] family of supercomputers.

When analyzing and comparing the performance of such machines, it is common to focus on metrics that characterize the basic processing elements (also called *nodes*) of these machines. The typical metrics include the computational performance (usually measured in floating-point operations per second), the memory capacity (measured in bytes) and the communication performance (measured in bytes/second) of the processing elements. It is also common to use certain figures of merit that express the ratio between metrics.

One of the most commonly used figures of merit for parallel machines is the "bytes-to-flop" ratio. That is computed as the ratio of communication bandwidth (B), typically measured in bytes/second, to the floating-point computation rate (R), typically measured in flops (floating-point operations)/second. Processing elements with higher B/R are considered superior to processing elements with a lower B/R.

Let us consider a three-dimensional mesh-interconnected parallel machine, consisting of processing elements with communication bandwidth B and floating-point rate R. Then, the byte-to-flop ratio of the processing element is simply B/R. Now consider a submesh of dimensions $q \times q \times q$ processing elements. The communication bandwidth in and out of the submesh is q^2B (proportional to the surface of the submesh) while the floating-point rate is q^3R (proportional to the volume of the submesh). Therefore, the byte-to-flop ratio of the submesh is $B/(qR)$. It is clear that the "byte-to-flop" figure of merit is not constant throughout the machine.

In this paper we derive expressions for the performance of a mesh-connected massively parallel processor for a limited but important class of computations. We consider computations that mimic physical simulations in real space. (As opposed to, for example, reciprocal space computations like FFT.)

We show the conditions for which machines built of completely different processing elements can deliver the same performance behavior for the computation. We introduce a new metric, that we call the *quality* of a processing element, that helps us compare processing elements of different characteristics. In particular, we show that one can trade off the "size" of the processing element (the calculation rate and memory capacity) for the communication performance of that processing element.

The rest of this paper is organized as follows. Section 2 introduces the model for the computation that we execute on a massively parallel machine. Section 3 introduces the model for the machine itself and derives the expressions that characterize the behavior for the machine when performing that computations. Section 4 discusses the implications of those expressions. Finally, Section 5 presents our conclusions.

2 Computation model

In this section we present a computation model that is representative of a limited by important class of applications. That class consists of simulations with nearest-neighbor interactions performed on a physical three-dimensional space. Examples of important applications that fall into that category includes sPPM and Sweep3D [6].

Consider a computation performed over a three-dimensional grid of dimensions $N \times N \times N$. For convenience, let us call N the *size* of the grid. Each grid point can be identified by its (x, y, z) coordinates, $0 \leq x < N$, $0 \leq y < N$, $0 \leq z < N$. Associated with each grid point are k bytes of data. That is, the total problem size in kN^3 bytes. For each grid point, f floating-point operations are performed in the computation. That is, the total amount of calculations is fN^3 flops. The operations are independent across grid points.

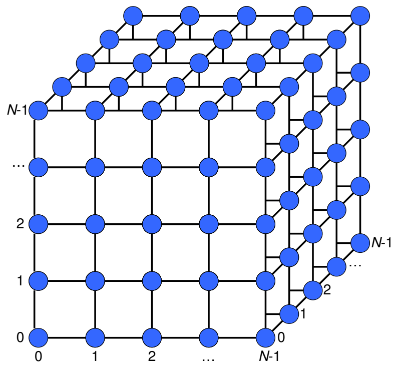

Figure 1: Computation model. The computation is performed over a three-dimensional grid of dimension $N \times N \times N$. For each grid point there are k bytes of data and f floating-point operations are performed in the computation.

Let the calculations in each grid point require data from its neighbors along each axis. That is, the calculations in grid point (x, y, z) requires data from grid points

$$\begin{array}{c}(x-1,y,z),(x-2,y,z),\ldots,(x-d_x,y,z)\\(x+1,y,z),(x+2,y,z),\ldots,(x+d_x,y,z)\\(x,y-1,z),(x,y-2,z),\ldots,(x,y-d_y,z)\\(x,y+1,z),(x,y+2,z),\ldots,(x,y+d_y,z)\\(x,y,z-1),(x,y,z-2),\ldots,(x,y,z-d_z)\\(x,y,z+1),(x,y,z+2),\ldots,(x,y,z+d_z)\end{array}$$

where $\mathbf{d} = (d_x, d_y, d_z)$ is called the *depth vector* of the computation. For simplicity, let $d_x = d_y = d_z = d$. In other words, performing the calculations for a particular grid point requires data from grid points up to d steps away in each direction.

In this paper we consider the case in which the problem is non-periodic. For example, grid points with $x = 0$ do not have any neighbors in the $-x$ direction. In our derivations we will ignore these boundary effects as they do not affect the conclusions. Our analysis is also valid for periodic problems. (For example, the $-x$ neighbor of grid point $(0, y, z)$ is grid point $(N-1, y, z)$.) In those cases, our derivations are the same if the machine is also periodic (*i.e.*, torus-connected as opposed to mesh-connected).

3 Machine model

We perform the computation on a machine that is a three-dimensional mesh of processing elements of dimension $P \times P \times P$. For convenience, let us call P the *size* of the mesh. A processing element

e can be identified by its coordinates:

$$e(x,y,z)\begin{cases} x = 0,1,\ldots,P-1 \\ y = 0,1,\ldots,P-1 \\ z = 0,1,\ldots,P-1 \end{cases}$$

Each processing element $e(x,y,z)$ in the interior of the mesh is directly connected by bidirectional links to six other processing elements – one each in the plus and minus directions of the x, y and z axes. That is, processing element $e(x,y,z)$ is directly connected to $e(x-1,y,z)$, $e(x+1,y,z)$, $e(x,y-1,z)$, $e(x,y+1,z)$, $e(x,y,z-1)$, $e(x,y,z+1)$, $\forall x,y,z : 0 \leq x,y,z < P$. Elements on the surface, edges and corners of the mesh are connected to fewer other elements, but that does not affect our analysis. Communication can be performed concurrently along all links.

We want to decompose and map the previous computation (presented in Section 2 on this machine. We adopt a straightforward decomposition and mapping as follows. Let $n = N/P$ be an integer. That is, let P divide N evenly. Then partition each of the axes of the problem grid into chunks of n points and number each chunk along an axes from 0 to $P-1$. After this partitioning, the problem space consists of $P \times P \times P$ chunks $c(x,y,z)$ of size $n \times n \times n$, $\forall x,y,z : 0 \leq x,y,z < P$. We map each chunk $c(x,y,z)$ to processing element $e(x,y,z)$. Just like N represents the global size of the computation, n is the size of the local grid in each processing element.

The processing elements are homogeneous and characterized by the following set of parameters:

1. $M(P)$: The size of the local memory of the processing element.
2. $L(P)$: The latency to initiate a communication over a link connecting two processing elements.
3. $B(P)$: The communication bandwidth along a link connecting two processing elements.
4. $R(P)$: The floating-point rate achieved by the processing element for this problem.

We make those parameters dependent on the mesh size (P) because we want to investigate how they impact the scalability of the machine.

For the computation to be feasible, each chunk of the problem grid must fit inside a processing element. Ignoring instruction memory and considering that all memory can be used for the computation data, that requirement can be expressed as

$$M(P) \geq kn^3 = k\left(\frac{N}{P}\right)^3, \tag{1}$$

which can also be expressed as

$$N \leq \left(\frac{1}{k}\right)^{\frac{1}{3}} P\sqrt[3]{M(P)}. \tag{2}$$

That is, the maximum problem size $N_{\max}$ that can be computed grows with the cube root of the processing element memory (for a given P).

The execution of the computation in the machine requires both communication (to exchange data between neighboring processing elements) and calculation of new values for each grid point. We will first derive expressions for the time to perform each of these separately and then we will combine them to obtain a total computation time.

The communication time, for the case in which all links can be active simultaneously and data only comes from the directly connected processing elements (*i.e.*, $d \leq n$), can be expressed as

$$T_{\text{comm}}(P) = L(P) + \frac{dkn^2}{B(P)} = L(P) + \frac{dk}{B(P) \cdot P^2} N^2. \tag{3}$$

Equation (3) includes the latency $L(P)$ to start communication in a mesh of size P plus the time to transfer d planes of size $n \times n$ grid points, each with k bytes of data, over a link of bandwidth $B(P)$.

The calculation time can be expressed as

$$T_{\text{calc}}(P) = \frac{fn^3}{R(P)} = \frac{f}{R(P) \cdot P^3} N^3. \tag{4}$$

where $R(P)$ is the floating-point rate achieved by the processing element when operating over a chunk of size $n \times n \times n$. We note that, in general, $R(P)$ is a function both of the calculation being performed (algorithm) and the local problem size n. In most machines, the rate decreases with the problem size, as data has to reside in progressively larger and slower levels of the memory hierarchy. We will focus on those cases for which the data always resides in main memory (as opposed to being small enough to fit entirely in cache) and therefore $R(P)$ is independent of the local problem size.

We can now compute the total computation time in a mesh of size P by combining the communication and calculation time. We first consider the case in which communication and calculation do not overlap. The cause of the nonoverlap is not important. It can be either forced by the algorithm or a characteristic of the machine. The total computation time for the nonoverlapping case can be expressed as

$$\begin{aligned} T_{\text{noov}}(P) &= T_{\text{comm}}(P) + T_{\text{calc}}(P) \\ &= L(P) + \tfrac{dk}{B(P)} n^2 + \tfrac{f}{R(P)} n^3 \\ &= L(P) + \tfrac{dk}{B(P) \cdot P^2} N^2 + \tfrac{f}{R(P) \cdot P^3} N^3. \end{aligned} \tag{5}$$

In general, some overlap is possible between computation and communication. Let us consider the case in which a complete overlap is possible. Then the total communication time is the maximum between the calculation and communication times.

$$\begin{aligned} T_{\text{ovlp}}(P) &= \max(T_{\text{comm}}(P), T_{\text{calc}}(P)) \\ &= \begin{cases} L(P) + \frac{dk}{B(P)} n^2 = L(P) + \frac{dk}{B(P) \cdot P^2} N^2, & \text{if } T_{\text{calc}}(P) \leq T_{\text{comm}}(P) \\ \frac{f}{R(P)} n^3 = \frac{f}{R(P) \cdot P^3} N^3, & \text{if } T_{\text{calc}}(P) > T_{\text{comm}}(P) \end{cases} \end{aligned} \tag{6}$$

4 Implications

Let us consider two scalability scenarios: *weak scaling* and *strong scaling*. In weak scaling, we keep the local problem size n constant and we grow the total problem size N as we grow P. Given a particular processing element of characteristics L, B and R, the computation time for a mesh of any size P will be the same, whether we fall in the overlapping or nonoverlapping case. The computation time is still sensitive to the parameters of the processing element (we want lower L and higher B and R), but any mesh scales perfectly and indefinitely under weak scaling for the problem we are considering.

In strong scaling, the total problem size N is constant and the local problem size n decreases with an increasing P. In this case, we compute the speedup $S(P)$ and efficiency $E(P)$ for a mesh of size P and processing element parameters L, B and R.

$$T(1) = \frac{f}{R}N^3 \tag{7}$$

$$S(P) = \frac{T(1)}{T(P)} = \frac{\frac{f}{R}N^3}{L + \frac{dk}{BP^2}N^2 + \frac{f}{RP^3}N^3} \tag{8}$$

$$E(P) = \frac{S(P)}{P^3} = \frac{\frac{f}{R}N^3}{P^3 L + P\frac{dk}{B}N^2 + \frac{f}{R}N^3} \tag{9}$$

The expressions are for the case of nonoverlapping calculation and communication. The efficiency is < 1 for all $P > 1$ and furthermore the efficiency decreases with increasing P. These conclusions also hold for the overlapping case, unless the communication can always be completely overlapped under the calculation even as we make n smaller.

Now, let us consider a submesh of dimension $q \times q \times q$, where $P = qQ$. (Both q and Q are integers.) The mesh of size P is equivalent to a smaller mesh of size Q in which the processing elements are $q \times q \times q$ submeshes of the processing elements in the larger mesh. That is, a mesh of size Q is equivalent to a mesh of size $P = qQ$ if the processing elements of the two meshes have the following characteristics:

$$M(Q) = q^3 M(P) \tag{10}$$

$$L(Q) = L(P) \tag{11}$$

$$B(Q) = q^2 B(P) \tag{12}$$

$$R(Q) = q^3 R(P) \tag{13}$$

The requirements above follow from construction of the mesh of size Q, since a submesh of size $q \times q \times q$ of processing elements has a memory capacity q^3 times larger than the memory of one processing element, the same latency as one element, q^2 times the communication bandwidth (there are q^2 links on each face of the submesh) and q^3 times the floating-point rate.

We note that the same requirements can be obtained by using Equation (5) to express the computation time for a mesh of size Q and forcing it to be the same as the computation time for a mesh of size P. (The same results would be obtained by using Equation (6).)

$$\begin{aligned}
T(Q) &= L(Q) + \frac{dk}{B(Q)\cdot Q^2}N^2 + \frac{f}{R(Q)\cdot Q^3}N^3 && (14)\\
&= L(Q) + \frac{dk}{B(Q)\cdot \left(\frac{P}{q}\right)^2}N^2 + \frac{f}{R(Q)\cdot \left(\frac{P}{q}\right)^3}N^3 && (15)\\
&= L(Q) + \frac{dkq^2}{B(Q)\cdot P^2}N^2 + \frac{fq^3}{R(Q)\cdot P^3}N^3 && (16)\\
&= L(P) + \frac{dk}{B(P)\cdot P^2}N^2 + \frac{f}{R(P)\cdot P^3}N^3. && (17)
\end{aligned}$$

We essentially want polynomials (16) and (17) to be equivalent for all N. Therefore, we need

$$\begin{aligned}
L(Q) &= L(P) && (18)\\
\frac{dkq^2}{B(Q)\cdot P^2} &= \frac{dk}{B(P)\cdot P^2} \Rightarrow B(Q) = q^2B(P) && (19)\\
\frac{fq^3}{R(Q)\cdot P^3} &= \frac{f}{R(P)\cdot P^3} \Rightarrow R(Q) = q^3R(P) && (20)
\end{aligned}$$

Which are the same requirements obtained previously by construction. The requirement of $M(Q) = q^3M(P)$ comes from having to fit the same data set into a smaller number of processing elements.

We note that

$$\frac{B(Q)}{R(Q)} = \frac{q^2B(P)}{q^3R(P)} = \frac{1}{q}\frac{B(P)}{R(P)}. \qquad (21)$$

That is, the byte-to-flop ratio for the bigger (more memory, higher floating-point rate) processing element (or of a $q \times q \times q$ submesh) is q times smaller than the ratio for the smaller processing element. Yet, meshes of the same total performance (same execution time for a computation) built from these two different processing elements perform identically for the computation model being considered.

At this point, we introduce a different figure of merit for a processing element, that we call the *quality* of the element:

$$\Phi_3(P) = \frac{B(P)}{R(P)}\sqrt[3]{M(P)} \qquad (22)$$

where the subscript 3 is used to indicate that this is applicable to the three-dimensional computation and machine models being considered. We note that

$$\Phi_3(Q) = \frac{B(Q)}{R(Q)}\sqrt[3]{M(Q)} = \frac{q^2B(P)}{q^3R(P)}\sqrt[3]{q^3M(P)} = \frac{B(P)}{R(P)}\sqrt[3]{M(P)} = \Phi_3(P). \qquad (23)$$

That is, the Φ_3 metric is the same for processing elements of meshes of equivalent performance. Furthermore, the Φ_3 metric for any $q \times q \times q$ submesh of processing elements is the same as the Φ_3 metric for the individual element.

These equivalences imply a flexibility in choosing the characteristics of the processing element for a three-dimensional mesh. First, one has to choose a certain memory-to-flop (M/R) ratio, as this will dictate how fast the calculations can be performed for a certain problem size. With that value chosen, larger processing elements (those with larger M and R) can tolerate lower B/R ratios. That is, the bandwidth in and out of the processing element does not have to scale linearly with R (or M). Correspondingly, nodes with larger B/R ratios can tolerate smaller memory (M) and computation rate (R).

In particular, Figure 2 shows some of the design space for the processing elements of a mesh. The reference point has a byte-to-flop ratio of B/R and certain memory size M. If we use a processing element with 1/2 or 1/4 of that ratio, then we correspondingly have to make this processing element 8 or 64 times bigger (in memory and floating-point rate) in order to build meshes of the same behavior.

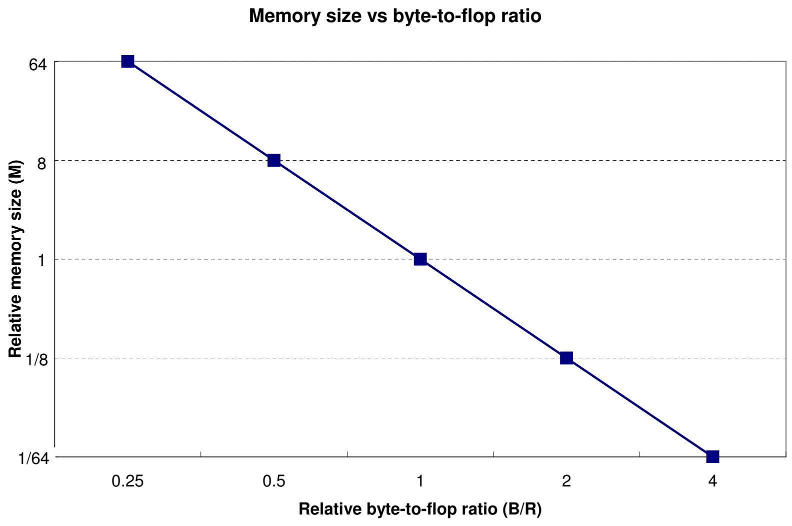

Figure 2: The memory size of a processing element must change with its byte-to-flop ratio if we want to keep the performance and scalability characteristics of the mesh constant. As we increase the byte-to-flop ratio by a factor a , the memory size can decrease by a factor a^3. The floating-point operation rate is proportional to the memory size.

If we can build processing elements with a byte-to-flop ratio that is 2 or 4 times the reference, then we can make those processing elements 8 or 64 times smaller (in memory and floating-point rate) and still build meshes with the same performance behavior. Obviously more of the smaller elements are needed to achieve the same performance.

5 Conclusions

We have shown how basic characteristics of a processing element (memory size, communication latency and bandwidth, floating-point rate) determine the performance and scalability characteristics of a massively parallel processor consisting of a three-dimensional mesh of those processing elements.

We show that meshes of equivalent performance for the same problem can be built of different processing elements, as long as the elements have the same memory-to-flop ratio and the same quality metric. We show that processing elements with larger memory can tolerate lower byte-to-flop ratios and, correspondingly, processing elements with higher byte-to-flop ratios can have smaller memory.

We believe that these results are particularly important to designers of highly integrated systems. One of the difficulties of integrating an entire processing element on a chip is that the memory is typically too big to fit on a single chip. By increasing the communication bandwidth we can design smaller processing elements (in memory and floating-point rate) until they fit on a single chip. Maybe even multiple processing elements can fit on a single-chip. Of course, this would increase the total number of processing elements in the machine (for the same global performance), but it is one more design option to consider.

Finally, we should mention that the analysis performed here can be extended for other computation and machine models. For example, one could compute a different metric called Φ_{FFT} that would give the condition for equivalence when computing a fast Fourier transform on different machines, or Φ_{LINPACK} when computing Linpack.

Acknowledgment We want to thank Prof. Geoffrey Fox for suggesting expanding the concept of quality of a processing element to other types of computations, beyond the strict three-dimensional models considered in this paper.

References

[1] S. Alam, R. Barrett, M. Bast, M. R. Fahey, J. Kuehn, C. McCurdy, J. Rogers, P. Roth, R. Sankaran, J. S. Vetter, P. Worley, and W. Yu. Early evaluation of ibm bluegene/p. In *SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, pages 1–12, Piscataway, NJ, USA, 2008. IEEE Press.

[2] Sadaf R. Alam, Jeffery A. Kuehn, Richard F. Barrett, Jeff M. Larkin, Mark R. Fahey, Ramanan Sankaran, and Patrick H. Worley. Cray xt4: an early evaluation for petascale scientific simulation. In *SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, pages 1–12, New York, NY, USA, 2007. ACM.

[3] A. Gara et al. Overview of the Blue Gene/L system architecture. *IBM Journal of Research and Development*, 49(2/3), 2005.

[4] Wilfried Oed and Martin Walker. An overview of cray research computers including the y-mp/c90 and the new mpp t3d. In *SPAA '93: Proceedings of the fifth annual ACM symposium on Parallel algorithms and architectures*, pages 271–272, New York, NY, USA, 1993. ACM.

[5] Steven L. Scott. Synchronization and communication in the t3e multiprocessor. In *ASPLOS-VII: Proceedings of the seventh international conference on Architectural support for programming languages and operating systems*, pages 26–36, New York, NY, USA, 1996. ACM.

[6] Jeffrey S. Vetter and Andy Yoo. An empirical performance evaluation of scalable scientific applications. In *Supercomputing '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–18, Los Alamitos, CA, USA, 2002. IEEE Computer Society Press.