Hypercube Algorithms for Direct N-body Solvers for Different Granularities

Jean-Philippe Brunet*
Alan Edelman†
Jill P. Mesirov*

(revisions as of 10/17/91)

Abstract

We compare and contrast algorithms for the N-body problem, particularly where N is in the range for which direct methods outperform approximation methods. With fewer bodies than processors, the so-called "replicated orrery" on a three dimensional grid has been used successfully on the Connection Machine CM-2 architecture. With more bodies, the "rotated and translated Gray codes" is an ideal direct algorithm for machines such as the CM-2 in that it takes optimal advantage of the communications bandwidth of the machine.

A classical Latin square can be used to abstractly denote any direct N-body calculation. Computational windows superimposed on this square illustrate the granularity of the computation. This point of view, naturally illustrates a sequence of algorithms ranging along a granularity scale in the following order: "massively parallel," "replicated orrery", "orrery", "rotated/translated Gray codes", and "serial".

*Thinking Machines Corporation, 245 First Street, Cambridge, MA 02142, brunet@think.com, mesirov@think.com

†Department of Mathematics, University of California, Berkeley, CA 94720, edelman@math.berkeley.edu

1 Introduction

In a wide variety of applications areas, including astronomy [?, ?, ?], molecular dynamics [?, ?, ?, ?], and fluid dynamics [?, ?, ?, ?], a serious computational bottleneck occurs from the need to determine the mutual interaction of N bodies. Directly computing the N^2 pairwise interactions given by

$$\text{Force(body } i) = \sum_j \text{Force(body } i \text{ due to body } j)$$

on a conventional computer can take an unacceptably long amount of time for values of N which are exactly in the range of interest for the application.

Approximation methods are one approach towards reducing computational complexity. Examples of this are the particle/mesh [?], local correction [?], hierarchical [?, ?], and multipole [?] methods. The structure of such algorithms has been investigated by [?].

Another approach is to keep to the naive direct method, but to design algorithms that take advantage of advanced computer architectures. We have chosen to take this path and to explore only direct methods in this work. Although this approach has asymptotically higher complexity, these algorithms remain usueful for two important reasons 1) they are faster for a large range of N on certain architectures and 2) they are simpler to code and maintain thus miniminizing the important commodity of programming effort. Though the approximation algorithms must ultimately be advantagous for large values of N, for many problems N is in the range where direct body algorithms are superior. See [?] for further discussion of the advantages of a direct algorithm over an approximation algorithm.

The algorithms for direct N-body solvers that we will discuss in this paper are designed for uses on a parallel architecture with a hypercube network topology. In fact, we feel that hypercube architectures are ideally suited to the direct method. For non-direct methods on the hypercube, see [?, ?, ?].

We focus on two different algorithms, but we take a point of view that we have not yet seen in the literature: we show that the two algorithms fit along a continuous spectrum. Perhaps we can make an analogy with the notion that X-rays and radio waves properly fit along one axis. However, instead of using frequency, the important parameter here is *granularity* measured by the ratio P/N.

For the case where the number of processors, P, satisfies $N \leq P \leq N^2$ we describe an optimal mapping of a family of *replicated orreries* onto a hypercube. For the case where $P < N$, we present a multiwire all-to-all broadcast algorithm which utilizes "rotated and translated Gray codes" to make optimal use of the communication bandwidth of the hypercube. Thus, in this approach, the communication time of the algorithm is negligible for large values of N.

Both of these algorithms have been implemented on the Connection Machine CM-2. There are two programming models for the CM-2 and each algorithm lends itself more naturally to one of them. The older model (which is becoming more and more obsolete) is known as the "fieldwise model", while the more recent model is the "slicewise model." From a conceptual point of view, the value of P is 32 times greater for the fieldwise model as it is for the slicewise model. Both approaches, and thus both programming models, were used for the N-body solver in the implementation of a random vortex method for numerical simulation of fluid flow [**?**]; we will analyze the computational complexity of each approach.

Section 2 discusses N-body problems in an abstract mathematical context. In Sections 3 and 4 we present two concrete algorithms with complexity analyses: the replicated orrery and a multiwire algorithm. The CM-2 implementation of both algorithms is briefly described in Section 5. A sample application is described in Section 6 and the performance of the CM-2 implementation of both approaches is discussed.

2 Latin squares and N-body problems

Before we analyze in detail two particular approaches to solving direct N-body problems that have been successful, we would like to abstract out the important mathematical features of an N-body computation on a parallel machine. In particular, the two important questions are **where** (which processor?) and **when** (which timestep?) does each interaction occur.

Body:	1	2	3	4
	2	3	4	1
	3	4	1	2
	4	1	2	3
	1	2	3	4

Figure 1:
A Latin square of Order 4. To indicate an N-body calculation, we let the entry in row i and column j denote an interaction between that body and body j. A serial algorithm could proceed from left to right and from top to bottom along this square, one box at a time.

Implicit in the knowledge of the "where and when" of the computation is the communication that occurs. Since we know which data needs to be where at a particular moment, the only degree of freedom might be how soon the data arrives at a particular processor before a computation starts.

As a mathematical abstraction, we consider it enlightening to display the "where and when" of an N-body computation on a **Latin square**. The basic idea is quite simple, however almost any N-body problem on any parallel machine can be defined as a Latin square.

A **Latin square** is an N by N table of entries $1, 2, \ldots, N$ with the property that each number i appears exactly once in each column and once in each row. Perhaps the earliest know reference is a puzzle requiring that the 16 lettered cards (A,K,Q,J) of an ordinary 52 card deck be arranged so that no letter appears twice in the same column or row nor along a diagonal [**?**]. Classically Latin squares have had many uses including the design of experiments, displays of multiplication tables of finite groups, and other recreational puzzles (See [**?**, **?**]). Their first systematic study was performed by Euler.[**?**]

As an example, Figure 1 displays a Latin square of order 4. In the context of an N-body computation, the entry a in row i and column j can symbolize a force law interaction of body a with body j. Thus in Figure 1 above, the first column depicts the interactions of body 1 with bodies 2,3,4, and 1 respectively. We remark that in practice, self-interactions are usually omitted.

			Example		
			P	N	$O(T_{comm})$
Case I	$P = N^2$	Massively Parallel	16	4	$\log N$
Case II	$N < P < N^2$	Replicated Orrery	8	4	$N^2/P + \log(P/N)$
Case III	$P = N$	Orrery	4	4	N
Case IV	$1 < P < N$	Rot/Trans Gray Codes	2	4	$N/\log P$ [1]
Case V	$P = 1$	Serial	1	4	0

Figure 2: Possible granularities (Example for $N = 4$). The communications complixities are discussed in Sections 3 and 4.2.

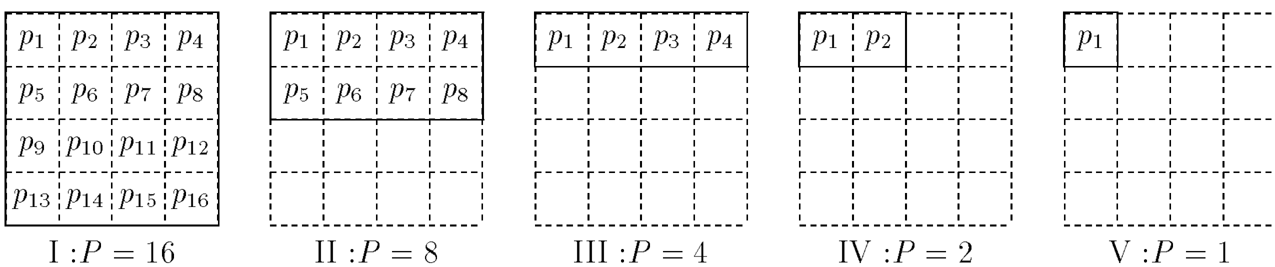


Figure 3:
Computational windows superimposed on the Latin square ($N = 4$). The entries of the square are omitted, while the processor numbers for the first timestep of the computation are included. Later timesteps are obtained by sweeping the window accross the square, so as to completely cover the square.

Also in practice, symmetric interactions may be avoided, giving a speedup of a factor at most equal to 2.

In the sections to follow, we study the five possibilities given in Figure 2. Specifically the replicated orrery is described in detail in Section 3, while the rotated and translated Gray codes is described in Section 4. Before we give the details of the algorithms, however, we can already describe some necessary features based on the granularity alone.

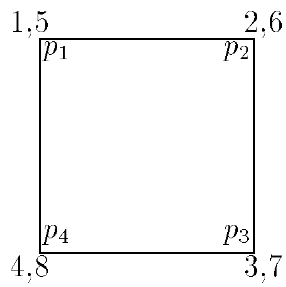
For each of the five cases in Figure 2, Figure 3 exhibits the simultaneity

[1] The actual communications complexity is the order of $\lceil \frac{N}{P \log P} \rceil P$. The approximation listed in the table breaks down for P near N.

body

interacts with

1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8
5	6	7	8	1	2	3	4
2	3	4	1	8	5	6	7
8	5	6	7	2	3	4	1
3	4	1	2	7	8	5	6
7	8	5	6	3	4	1	2
4	1	2	3	6	7	8	5
6	7	8	5	4	1	2	3



2 Dimensional Hypercube

p_1	p_2	p_3	p_4

Comp Window

Figure 4:
Latin square and computational windows for an 8-body problem with 4 processors. Each 1 by 4 rectangle denotes the four interactions of one time step, one in each processor. The entries of the Latin square are obtained by circulating a dynamic copy of the first body counterclockwise, while circulating the second body clockwise on the square. A static copy of each body remains in the processor. Communication is indicated by the vertical space between pairs of lines. After the first circulation, body 1 can interact with bodies 2 and 8, while body 5 can also interact with bodies 2 and 8 without further communication. Thus a total of four windows can pass over the square before another communication need occur.

of one parallel computation when $N = 4$. For clarity of exposition, we omit the entries of the square, but instead superimpose a computational "window" upon the square. The size of this window is exactly the number of processors. In each window, there is one intereaction for each processor. The computation proceeds by sweeping the window accross the square. Of course, if the number of bodies is held constant, then the size of the window grows with the number of processors, and the number of timesteps (equals the number of windows needed to cover the square) decreases.

The two extremes are given by Cases I , massively parallel, and V , serial, in Figure 3. In Case I , $P = N^2$ there is only one window – all N-body computations occur in parallel. On the other hand, in Case V , where $P = 1$, the algorithm is serial and only one interaction occurs in any given timestep. The processors are labeled p_i in the windows, and the computation proceeds by shifting the windows and the corresponding processors over the Latin square.

Figure 5: Digital Orrery: $P = N$. Cycle through N shifts.

We now see that once the relationship between P and N is given, all that we need to specify a direct N-body computation is 1) a choice of Latin square and 2) a mapping from the processors to the window. We explore this in further detail in later sections, but we can give an illustrative example, as we do in Figure 4. This figure illustrates an example when $N = 8$ and $P = 4$. The square on the left is readily checked to be a Latin Square. The construction of the square is quite simple. We assume that there are two bodies per node as illustrated on the 2 dimensional hypercube in the figure. One copy of the bodies remains static, while one circulates, the first body circulates counterclockwise, while the second body circulates clockwise.

3 $N \leq P \leq N^2$: Orreries and Replicated Orreries

One standard approach towards solving the N-body has N processors connected as a ring. Processor i accumulates the forces on body i. At each time step, N of the pairwise interactions are computed, then the appropriate data are passed around the ring so that in N steps all N^2 interactions have been computed and accumulated into the correct processor, see Figure 5. This particular parallel implementation of the N-body algorithm has been referred to as the "digital orrery" [?] as the "systolic loop" [?], and as "systolic loop double" [?].

As a Latin Square, this is the addition table of the Abelian cyclic group of order N: the square is circulant, and the (i, j)th entry is essentially $i - j$ modulo n. This case is illustrated in Figure 3 as Case III, which shows that one row of interactions is computed in each timestep.