JOURNAL OF COMPUTATIONAL PHYSICS 96, 224–228 (1991)

Note

On the Systolic Calculation of All-Pairs Interactions Using Transputer Arrays

The parallelization methods of the molecular dynamics (MD) algorithms depend on the interprocessor connection topology of the multiprocessor array as well as on the characteristics of the simulated physical system, and in particular on the particle–particle interactions. Furthermore, the parallelism grain to be adopted is determined by the ratio between the size of the simulation box and the range of the interparticle potentials. Thus, if the latter is comparable with the size of the simulation box, a fine grain approach is most suitable, where all-pair interactions have to be calculated. This was the case in previous concurrent MD program implementations concerning Lennard–Jones particles and water molecules on transputer based systems [1, 2]. Additional alternative schemes have been described in Refs. [3, 4]. The present note describes a conceptually simpler formulation and an improved implementation of the systolic approach to the same problem of computing all-pairs interactions.

Consider a system of $\mathbf{N}$ molecules whose pairwise interactions have to be evaluated. For the sake of clarity, we start examining the case of an odd number of processors $\mathbf{P}$ and a particle number $\mathbf{N}$ which is a multiple of $\mathbf{P}$, i.e., $\mathbf{N} = \mathbf{nP}$. These assumptions will be relaxed in the following. In our implementation, $\mathbf{P}$ identical processes run concurrently on $\mathbf{P}$ transputers which are connected by communication channels to form a circular pipeline. Each process receives data from the neighboring process on one side and sends data to the neighboring process on the other side. Thus, $\mathbf{P}$ data packets consisting of coordinates and forces corresponding to groups of $\mathbf{n}$ particles, are circulated round the ring in a synchronized manner which is called systolic [5].

Each process starts each MD integration step by making a copy of its data packet. One packet will remain in its "home" process, while the copy will be systolically passed along the ring, coming back to its home process in $\mathbf{P}$ pulses. Initially—that is, in the zeroth pulse—the $\mathbf{n}(\mathbf{n}-1)/2$ interactions within the resident $\mathbf{n}$-particle groups are evaluated. Then, $\mathbf{P}-1$ pulses of systolic calculations will follow in which interactions among fixed and circulating packets are evaluated. In each of the first $(\mathbf{P}-1)/2$ pulses, $\mathbf{n}(\mathbf{n}+1)/2$ interactions are evaluated, whereas in each of the remaining $(\mathbf{P}-1)/2$ pulses, $\mathbf{n}(\mathbf{n}-1)/2$ interactions are calculated. To bring each circulating data packet back to its home process a further pulse of pure communication is needed. In the home process the fixed and circulating force vectors are added to yield the total force acting on each particle. At this point, each

0021-9991/91 $3.00

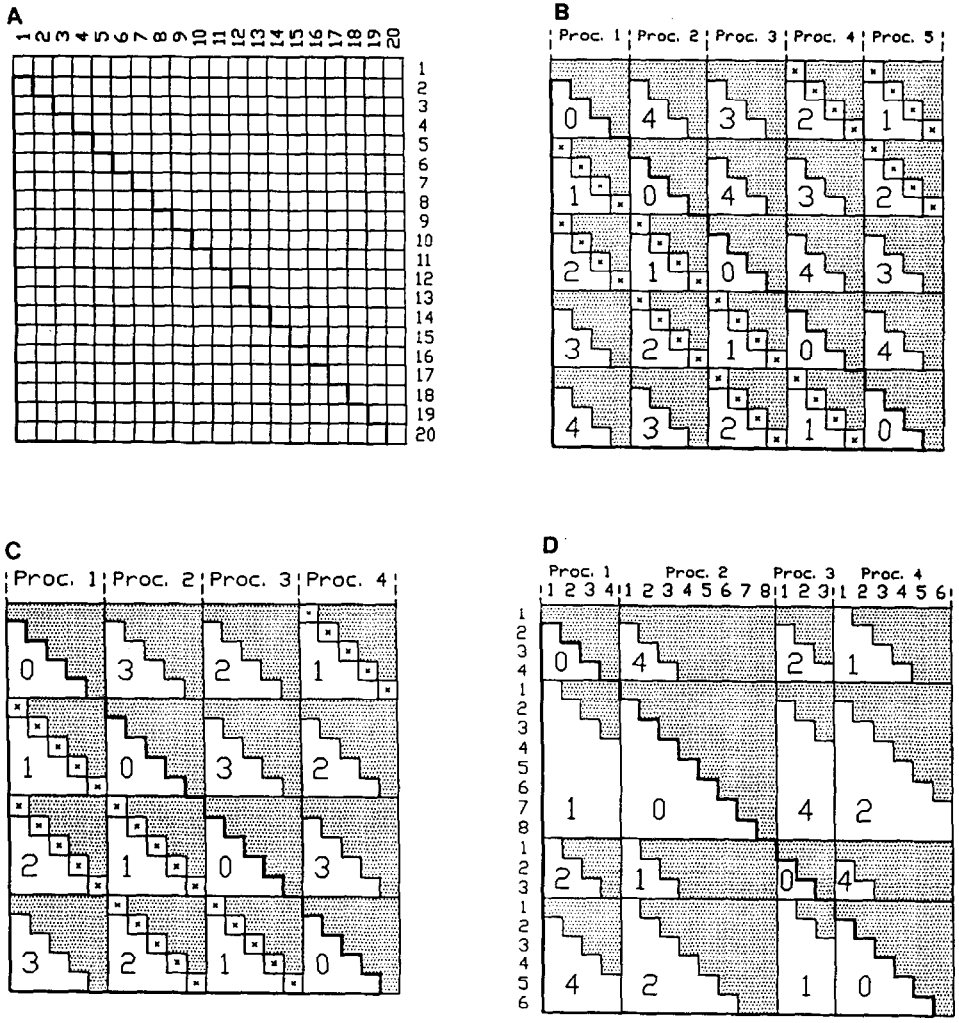

FIG. 1. (A) Interaction matrix **A** for a system of 20 particles. Since **A** is symmetric, only the evidenced strictly lower triangular part has to be evaluated. (B) Decomposition of the interaction matrix **A** into square submatrices for the case of particle number $\mathbf{N} = 20$ and processor number $\mathbf{P} = 5$. Each column of **P** submatrices is assigned to the indicated processor. The dotted regions are *not* calculated. The strictly lower triangular parts of the submatrices are labeled by the systolic pulse number in which they are evaluated. The subdiagonals of **A** (labeled by **x**) are calculated in the "extended" (see text) systolic pulses 1 and 2. (C) The same as in (B) for the case of an even number of processors ($\mathbf{P} = 4$). The systolic pulse 2 is "extended" only for processors 1 and 2. (D) Unbalanced processor loading for the case of $\mathbf{N} = 21$ and $\mathbf{P} = 4$. The particle indexing within each submatrix is indicated. As in (C) the systolic pulse 2 is "extended" only for processors 1 and 2.

process updates the coordinates of its own particles by integration of Newton's equation using some numerical method—e.g., Verlet's formulas [6].

To show that the systolic loop method outlined above enables one to evaluate all pair interactions without any duplication, we will refer to the $\mathbf{N} \times \mathbf{N}$ symmetric matrix **A** of interactions shown in Fig. 1A for the $\mathbf{N} = 20$ case. This matrix can be decomposed into $\mathbf{P}^2$ square disjoint submatrices of order **n**. The diagonal elements of **P** of them coincide with diagonal elements of **A**. We shall refer to these submatrices as DGL. The remaining $\mathbf{P}(\mathbf{P}-1)$ are two by two transposed. Moreover, the strictly lower triangular part (SLTP) of **A** is made up of $\mathbf{P}(\mathbf{P}-1)/2$ submatrices which will be called "internal," and by the SLTP of the **P** DGL matrices. Each internal matrix has its transposed in the upper triangular part of **A**.

In the zeroth systolic pulse, each processor evaluates the SLTP of the DGL matrix that corresponds to the interactions among its own particles. To understand how the remaining SLTP of **A** can be evaluated, maintaining virtually perfect the load balance among processors, it is convenient to think that a column of **P** square submatrices is assigned to each processor, as shown in Fig. 1B. In the ith pulse of the systolic loop, each processor evaluates the SLTP of that of its submatrices, which is labeled by i. In this way, during each pulse all processors calculate the same number of interactions. Further, no calculation is duplicated. Indeed, examining the submatrices attributed to any processor pair, one can see that they are completely different, except for two of them which are transposed. In the latter case, the evaluation by both processors of the SLTP of their own matrices is equivalent to the evaluation of the whole submatrix except for its diagonal elements. Thus, if in every pulse only the SLTP of each submatrix is evaluated, after **P** pulses $\mathbf{P}-1$ subdiagonals of the SLTP of the **A** matrix would still remain to be calculated, i.e., those subdiagonals whose elements (labeled by **x** in Fig. 1B) coincide with the diagonals of the internal submatrices. The natural synchronization of the submatrix evaluation by concurrent processes, which is imposed by the systolic data flow described above, helps to avoid this difficulty. Indeed, during each systolic pulse—different from the zeroth one—the evaluation by each processor of the diagonal elements of its submatrix results in the evaluation of two subdiagonals of **A**. Therefore, all $\mathbf{P}-1$ subdiagonals of **A** can be evaluated if in the first $(\mathbf{P}-1)/2$ pulses the whole lower triangular part of the submatrices is evaluated. In the following we shall refer to these calculation pulses as "extended pulses."

This method does not show any idiosyncrasy concerning the number of processors or the particle to processor number ratio. Indeed, it is applicable also if the processor number is even, as shown in Fig. 1C. The only difference with respect to the above discussed case of an odd number of processors concerns the evaluation of the $\mathbf{P}-1$ subdiagonals of the **A** matrix. Since their number is odd in this case, the first $\mathbf{P}-2$ of them are evaluated in $(\mathbf{P}-2)/2$ extended pulses, and the last is evaluated in a further pulse which is extended only for the first half of the processors and normal for the others. This causes a slight unbalance of the processor computational load which is of the order of $1/\mathbf{N}$.

The present method can also be applied when the particles are not evenly distributed onto processors (Fig. 1D). This feature is useful, e.g., in solid-state simulations when **N**, being restricted by symmetry requirements of the basic cell, is not a multiple of the processor number. In this case, the **P** submatrices assigned to each processor are not all square matrices. Therefore, it will generally be impossible to speak about SLTP or diagonals of submatrices. Nevertheless, a strict analogy still holds true between an "unbalanced" case (e.g., Fig. 1D) and the corresponding "balanced" one (Fig. 1C). Thus, instead of evaluating the SLTP of a square submatrix (as in the balanced case), each processor calculates the interactions among its home particles and the particles labeled by higher indexes within the relevant submatrix. Furthermore, in the extended pulses the interactions among particles with the same index are also included.

The neighbor list acceleration device can be easily added to this method, since at each MD time step a given processor evaluates the interactions involving its home particles. Thus, it can set up a neighbor list in its local memory and use it on subsequent time steps. Likewise, a spherical cutoff can be used, though some degree of workload unbalance can result [4]. A concurrent Occam [7] implementation of this method has been carried out for MD simulation of water molecules interacting via ST2 potentials [8], which can be parametrically adapted to any particle and processor numbers. To perform our tests, various transputer arrays were used, each of them being organized according to a ring topology using two serial links per transputer. In Table I, results are reported concerning execution time values (for a time step of $5 * 10^{-14}$ s) that have been obtained by simulating a system of 216 water molecules on transputer arrays and on a VAX-11/750 computer equipped with floating point accelerator. Values are also reported of the system efficiency E, which are obtained by the expression

$$E = (T_1/P)/T_P,$$

TABLE I

Execution Time (in s) per Molecular Dynamics Time-Step ($5 \cdot 10^{-14}$ s) and Efficiency for a System Consisting of 216 Water Molecules Interacting via ST2 Potentials

Algorithm type	Computer	Exec. time (s)	Efficiency
Sequen. (Fortran)	VAX-11/750	47	
	1 T800-20	20	
Sequen. (Occam)	1 T800-20	12.7	1.00
Concur. (Occam)	3 T800-20	4.2	1.00
	4 T800-20	3.2	0.99
	6 T800-20	2.2	0.96
	8 T800-20	1.7	0.93

Note. Simulated on different computers using sequential (Sequen.) or concurrent (Concur.) algorithms.

where T_1 and T_P are the execution time values for one- and **P**-transputer systems, respectively. A spherical cutoff of 7.8 Å has been used in water–water interaction evaluation.

ACKNOWLEDGMENTS

General indirect support from Italian MPI-60% and C.R.R.N.S.M. is acknowledged.

REFERENCES

1. F. BRUGÈ, V. MARTORANA, AND S. L. FORNILI, *Mol. Simul.* **1**, 309 (1988).
2. F. BRUGÈ, V. MARTORANA, AND S. L. FORNILI, in *Proceedings of CONPAR88, Manchester, U.K., 1988*, edited by C. R. Jesshope, and K. D. Reinartz (Cambridge Univ. Press, Cambridge, UK, 1989), p. 474.
3. D. FINCHAM, *Mol. Simul.* **1**, 1 (1987).
4. A. R. C. RAINE, D. FINCHAM, AND W. SMITH, Daresbury Laboratory Report No. DL/SCI/P625T, 1989 (unpublished).
5. J. A. B. FORTES AND B. W. WAH, *IEEE Comput.* **20**, 12 (July 1987).
6. L. VERLET, *Phys. Rev.* **159**, 98 (1967).
7. INMOS, *Occam2 Reference Manual* (Prentice-Hall, New York, 1988).
8. F. H. STILLINGER AND A. RAHMAN, *J. Chem. Phys.* **60**, 1545 (1974).

RECEIVED: March 13, 1989; REVISED: September 6, 1989

FILIPPO BRUGÈ
Physics Department
University of Palermo
Palermo, Italy

SANDRO L. FORNILI
Physics Department
University of Palermo and C.N.R.-I.A.I.F.
Via Archirafi 36
I-90123 Palermo, Italy