

CASCADE and THBREL - Implementation on the INTEL iPSC/860

C. E. Dean, M. Leslie, and S. E. Marshall

SERC Daresbury Laboratory, Daresbury, Warrington WA4 4AD

S. C. Parker

School of chemistry, University of Bath.

Introduction

CASCADE and THBREL are energy minimization programs in the CCP5 program library dealing with ionic crystals defects. The energy is minimised using a modified Newton-Raphson technique.

The programs

In both programs a second derivative matrix and gradient vector are calculated, the matrix is inverted and a step direction calculated from the product of the inverse second derivative matrix with the gradient vector. A step is then taken in this direction. This is only carried out once, as subsequent steps use a matrix update algorithm which involves only calculating the gradient vector and matrix vector multiplications. The time consuming parts of the codes are the calculation of the gradient vector and updating the second derivative matrix inverse using matrix vector multiplication.

The codes have been parallelised using the portable harness FORTNET and has run on INTEL, MEIKO(I860) and MEIKO(T800). To parallelize the gradient calculation step the existing vector code is used. This sets up a list of pairs and then calculates the interactions between them. The pairs list is set up on all nodes but the interaction calculation is only carried out on one node controlled by a master process using a global counter. A global sum is then carried out to give the gradient vector. The global counter is stored in FORTNET. On the INTEL the master process is interrupt-driven and runs on one of the nodes that performs floating point arithmetic. On the MEIKO the master process currently runs on a node which performs no other floating point arithmetic. The three-body part of the gradient calculation can be divided into sections which do the same amount of work. This part is therefore be parallelised more easily by sharing the work equally over all the nodes.

The matrix is distributed evenly over all the nodes. To do this the matrix is divided into sections consisting of a number of rows and all columns. Each section is calculated using the global counter method using all nodes storing the partial sums in a buffer. The buffer is then globally summed and copied to the elements of the distributed array on the nodes on which it is required.

A distributed matrix inverter has been written to carry out the matrix inversion. The input matrix is overwritten by the result and the work space required is of order N with only $N^2/\text{numnodes}$

of words of storage per node for the matrix. (Numnodes = number of nodes). A 2048 x 2048 matrix takes 236 seconds to invert on 8 nodes corresponding to 73 Mflops. The BLAS library rank one operation routine DGER is called to perform most of the floating point arithmetic. It has a rated performance of 11.3 Mflops per node. 1 CRAY X-MP processor takes 150 seconds running at 206 Mflops using a different algorithm to perform the same calculation.

The final step in both calculations is the distributed matrix vector multiplication for which the gradient vector will be available on all nodes following the global sum.

Timings

For “real” problems, CASCADE needs to be distributed over up to 8 ipsc/860 nodes. Timings for 1 (8) nodes on the ipsc/860, MEIKO(i860) ,MEIKO(t800) and CRAY X-MP for three problem sizes (N = number of variables) are (630 variables needs 16Mbytes for a single node calculation and could not be done on MEIKO)

MACHINE	N	INTEL I860	MEIKO I860	MEIKO T800	CRAY XMP
STEP					
Matrix Calculation	208	13.9 (3.4)	26.0 (10.0)	131.0 (55.0)	1.71
	410	34.3 (9.4)	65.0 (31.0)	320.0 (170.0)	4.24
	630	66.9 (19.4)	- (66.0)	- (372.0)	8.25
Matrix Inversion	208	1.8 (4.9)	17.0 (18.0)	105.0 (70.0)	0.28
	410	12.9 (13.6)	193.0 (58.0)	797.0 (290.0)	1.61
	630	45.3 (27.4)	- (150.0)	- (791.0)	5.88
Gradient Calculation (Per Cycle)	208	11.4 (2.0)	20.0 (4.0)	90.0 (21.0)	1.17
	410	27.5 (4.7)	48.0 (10.0)	216.0 (45.0)	2.81
	630	53.4 (8.9)	- (16.0)	- (83.0)	5.43
Iteration Step	208	122.6 (36.6)	238.0 (49.0)	923.0 (217.0)	12.24
	410	177.9 (47.3)	351.0 (64.0)	1339.0 (288.0)	17.41
	630	288.3 (59.9)	- (96.0)	- (439.0)	27.73
Total Time	208	436.1 (180.4)	685.0 (243.0)	3124.0 (979.0)	44.0
	410	709.1 (281.9)	1317.0 (438.0)	5898.0 (1872.0)	75.8
	630	1197.1 (441.1)	- (793.0)	- (3724.0)	133.0

The matrix calculation step runs 3.3 times faster on 8 nodes than on 1 node for the largest case; the timing ratio is getting worse for larger systems. This is because this step carries out $O(N^2)$ communication. However the poor scalability is not important as this step is only carried out once and only represents 5% of the total time. The matrix inversion step scales very badly for matrices of this size. The communication is $O(N)$ and the calculation is $O(N^3)$ so the scalability improves

rapidly with problem size. The poor scalability does not matter as the inversion is only done once and represents 10% of the total execution time. Larger systems will have better scalability for this step. The gradient calculation speedup on the INTEL is 5.8 for the largest system; this step improves its scalability with problem size. This step involves the bulk of the execution time.

Comparing the INTEL and MEIKO, the MEIKO(I860) is about 2 times slower and the T800 is 10 times slower. For both machines the scalability on increasing the number of nodes is worse reflecting the slower communication.