DL/SCI/TM101T

technical memorandum

Daresbury Laboratory

DL/SCI/TM101

A BENCHMARK OF CRYSTAL ON WORKSTATIONS / 2

by

N. M. HARRISON & V. R. SAUNDERS, EPSRC Daresbury Laboratory

April, 1994

94/300

DRAL

Daresbury Laboratory
Rutherford Appleton Laboratory

DRAL is part of the Engineering and Physical Sciences Research Council

ISSN 0144-5677

A Benchmark Of CRYSTAL On Workstations / 2

N.M. Harrison[1] and V.R. Saunders
SERC Daresbury Laboratory, Warrington, WA4 4AD, UK

Abstract

This memorandum extends previous benchmarks [1] of CRYSTAL92. Timings for a replicated data parallel implementation and several new workstation platforms are presented.

1 Introduction

CRYSTAL is a program for the computation of the electronic structure in crystals, slabs, polymers and molecules using Hartree-Fock theory and a Gaussian basis set [2]. The calculation is performed in two steps (parts). Part 1 is responsible for computing a file of Gaussian integrals, performs little input/output and is not highly vectorizable (typically a compute rate of 20 Mflops is observed on a Cray XMP). Part 2 peforms the energy minimization, is input/output intensive and well vectorised (typically a compute rate of 80 Mflops is observed on a Cray XMP).

The results of benchmarking CRYSTAL on a variety of workstations are reported. Twelve test cases, including crystals, slabs, polymers and molecules were used, and the timings reported are summed over these cases. The range of equipment benchmarked is given in table 1. Comparable results for a Convex C220 (1 processor) and a Cray XMP 4/16 (1 processor) are presented for reference. In all cases the Fortran compiler was used at the maximum level of optimization declared to be safe by the supplier (except as noted below for SGI and HP). All data was collected prior to March 1994.

[1] n.m.harrison@dl.ac.uk

2 Problems Encountered

- It was noted that when performing unformatted input/output the syntax forms

```
READ(IUNIT)(A(I),I=1,N)
or
DIMENSION A(N)
READ(IUNIT)A
```

  usually gave very different performance levels, with the second syntax form being more efficient (similar behaviour was observed for WRITE statements). Using the second syntax CPU-times were improved by approximately a factor of ten for the IBM and SGI systems, four for SUN systems, two for DEC systems and not at all for the HP, Convex and Cray systems. CRYSTAL is a code which performs extensive input/output; accordingly all the input/output was modified to use the second syntax form, and it is this modified form of the code which was benchmarked. On a number of machines we have determined that coding the input/output by direct calls to the C library causes a further improvement by a factor of 2 to 3, although such an implementatation does not form part of the present benchmark.

- The SGI Fortran compiler version 3.4.1 miscompiled one routine when the full optimization level (-O2) was used (SUBROUTINE RCALXY). This problem is fixed in compiler version 3.16 which was supplied with the SGI Challenge/XL.

- The HP Fortran compiler version 9.00 miscompiles a routine at optimisation levels O3 and O2 (SUBROUTINE VIC5J). Accordingly this routine was compiled at level O1 without problems.

- On the HP, implicit opening of files worked except if a file was REWOUND before being first used in a READ or WRITE statement. In this case the file became incorrectly positioned causing ensuing input/output errors. The solution was to explicitly open all files before use. This "problem" has been reported to HP.

- The code must be run in 64-bit floating point precision for satisfactory results. The default precision on all the workstations and the Convex is 32-bit whilst on the Cray it is 64-bit. Accordingly a code (AUTO) which reads single-precision Fortran source and outputs the double-precision counterpart was prepared and used for all runs on workstations and the Convex. AUTO appears to be completely portable and is freely available from the present authors. We have also confirmed that use of the manufacturer supplied AUTODBL option of the Fortran compiler on IBM RS/6000 systems instead of AUTO also gives rise to satisfactory results.

3 Results; sequential code

All numerical results obtained from the various computer systems tested agreed to within the tolerances expected from rounding error considerations (agreement to at least 10 significant figures in all cases), thereby confirming that the code ran correctly on all machines. In table 1 we report the sum of system and user time (=CPU-time) for part 1 (integral generation) and part 2 (energy minimization iterations). For a given processor type the CPU-times scale roughly with the clock speed: this is not true for the HP 750-755 where the compiler version 9.00 on the HP 755 appears to compile part 1 significantly better than did version 8.05 on the HP 750. Clock speed is not a reliable indicator of performance when different processor classes are compared.

4 Results; parallel code

Parts 1 and 2 of CRYSTAL have been modified to take advantage of a multi-processor environment. A strategy suitable for distributed memory/disk systems is adopted in that we imagine that each node has local memory and disk which may be accessed considerably faster than that on remote nodes. The implementation is thus targetted at loosely coupled machines. A replicated data strategy is used in which computationally intensive *tasks* are defined and distributed dynamically using a synchronous global index.

In part 1 each processor calculates a subset of the bi-electronic integrals

Computer System	clock(Mhz)	Part 1	Part 2
Convex C220	25	5816	1117
Cray XMP 4/16	118	760	123
IBM RS/6000 530	25	2006	1020
IBM RS/6000 530H	33.3	1499	765
IBM RS/6000 550	41.7	1205	612
IBM RS/6000 370	62.5	771	391
HP/9000 750	66.7	1356	690
HP/9000 755	99	633	482
DECstation 5200	-	8561	4849
DEC AXP 3600	175	760	361
SGI 4D/220	25	6104	2175
SGI Crimson	50	2363	829
SGI Challenge/XL	150	1183	392
SUN SS10 model 30	36	2901	1439
SUN SS10 model 11	10	2018	1337

Table 1: Sequential version CPU-time for Parts 1 and 2 in seconds

which may be labelled (i^0,j^G,k^H,l^{N+H}) where the (i,j,k,l) indices refer to AO's within a cell and the (0,G,H,N) are cell labels. A compromise between task size and management overheads is reached if one defines a *task* as the set of all integrals involving (i^0,j^G). In part 2 each processor reads the subset of the integrals stored on its local disk and computes a partial Fock matrix in direct space. The work distribution of this step is determined by the distribution of the integral calculation in part1. The time limiting step is disk-access rates and we do not attempt to access disks attached to remote nodes.

The Fock matrices are then accumulated on each node by *global summation* and transformed to a Bloch function representation at a finite number of sampling points ($\mathbf{k}_I$). The diagonalisation of the Fock matrix at each k-point is treated as a *task*; these tasks are distributed using a global index. The resultant eigenvalues are accumulated on each node by *global summation*.

This simple strategy was chosen as it is achieved with a small number of changes to the sequential code but successfully distributes the major computational and IO tasks. A limitation of this strategy is that copies of all main memory resident data structures are stored on each processor, so that the maximum problem size is limited by the node with the smallest memory.

A distributed data algorithm is currently under development.

In order to benchmark the performance of CRYSTAL on clusters of workstations we calculated the electronic structure of a slab of corundum (Al_2O_3). There are 10 atoms in a unit cell with 100 electrons described by 86 atomic orbitals; 8 irreducible k-points were used. We report the *average* CPU-time per node and the total elapsed time for various types of network and workstation. The elapsed time contains the overhead associated with loading the program from an NFS file system within the cluster.

4.1 Homogeneous Clusters

Two seperate clusters were considered. Firstly a cluster of 3 IBM RS/6000 530H workstations linked using ethernet and serial optical comunications channels (SOCC). Secondly a cluster of 4 HP/9000 model 720 machines linked using ethernet and FDDI channels. On both clusters local SCSI disks running at approximately 2 MB/sec were used. The average CPU-time per node and elapsed times are reported in table 2.

For part1 the elapsed times scale with cluster size rather well, with elapsed time reduced by a factor of 3.7 on 4-nodes (HP-FDDI). The communication overhead in part1 does not increase with system size and ideal scaling will be achieved in larger calculations. In part2 elapsed times are reduced by a factor of 3.3 on 4-nodes (HP-FDDI). Two factors limit the efficiency of part2. As noted above the load balancing of the Fock matrix construction is determined by part1. This difficulty is most pronounced on heterogeneous hardware and is examined in greater detail below. The other factor is the communication of the partial Fock matrices to all nodes; a step which scales as N^2 where N is the number of AO's in the system. The CPU-time required to diagonalise the Fock matrix scales as N^3 and thus this overhead will become less significant in studies of larger systems. The effect of different network connections on the elapsed times is slight.

4.2 Heterogenous Clusters

In order to examine the effects of *load balancing* on heterogeneous hardware we performed the same benchmark calculation on a cluster consisting of a Sun sparc2, and SGI 4D35 and an IBM 320 workstation. Here, for comparison purposes, we also introduce *static load balancing* in which each node is

	Part1		Part 2		
	CPU/node	elapsed	CPU/node	elapsed	
IBM 530H					
1	275	294	178	424	
2	138	149	96	222	Ethernet
3	94	100	66	158	
1	275	292	179	424	
2	138	148	95	217	SOCC
3	92	100	64	151	
HP 720					
1	349	360	198	556	
2	176	186	111	324	Ethernet
3	119	124	83	254	
4	90	97	66	188	
1	349	358	196	557	
2	176	182	111	313	FDDI
3	119	124	82	233	
4	90	98	66	170	

Table 2: Parallel version timings (seconds) for an homogeneous cluster

required to perfom the same number of integral evaluation *tasks*. The results are reported in table 3.

The IBM workstation is a factor of 5 faster at generating integrals than the SUN workstation and almost 2 times faster than the SGI. The introduction of dynamic load balancing thus produces a dramatic decrease in the elapsed time. In part2 the elapsed time is slightly increased as the IBM now has many more integrals to process than the SUN or SGI machines and its relative performance advantage in performing this task is not so great. The overall effect of dynamic load balancing is a 30% decrease in elapsed time.

	Static		Dynamic	
	elapsed	CPU	elapsed	CPU
Part1				
sun		612		229
sgi	671	231	263	242
ibm		121		219
Part2				
	wall	CPU	wall	CPU
sun		319		84
sgi	406	90	468	85
ibm		69		162

Table 3: Total elapsed and CPU-time on each node (seconds) for static and dynamic load balancing on an heterogeneous cluster

References

[1] N. M. Harrison and V. R. Saunders *A Benchmark Of CRYSTAL On Workstations / 1*, DL/SCI/TM90T, (1993)

[2] C. Pisani, R. Dovesi and C. Roetti *Hartree Fock Ab Initio Treatment of Crystalline Systems*, Lecture Notes In Chemistry, **48**, (Heidelberg: Springer-Verlag) (1988); R. Dovesi, V. R. Saunders and C. Roetti *CRYSTAL92: An ab initio LCAO Hartree Fock Program for Periodic Systems*, User Manual (1992)