technical memorandum

Daresbury Laborator

DL/SCI/TM90T

A BENCHMARK OF CRYSTAL ON WORKSTATIONS / 1

by

N.M. HARRISON and V.R. SAUNDERS, SERC Daresbury Laboratory

DECEMBER, 1992

G92/249

Science and Engineering Research Council

DARESBURY LABORATORY

Daresbury, Warrington WA4 4AD

ISSN 0144-5677

A BENCHMARK OF CRYSTAL ON WORKSTATIONS / I

N.M. Harrison and V.R. Saunders

SERC Daresbury Laboratory, Warrington, WA4 4AD, UK

December 2, 1992

---

1. Introduction

CRYSTAL is a program for the computation of the electronic structure of crystals, slabs, polymers and molecules using Hartree-Fock theory and a Gaussian basis set. It is broken up into two parts. Part I 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 covered IBM RS/6000 models 530 (25MHz), 530H (33.3MHz) and 550 (41.7MHz), DEC/5000 model 200, SGI 4D/220 (25MHz R3000 chip) and Crimson (50 MHz R4000 chip), SUN SS10 model 30 (36 MHz), and HP/9000 model 750 (66.7 MHz). 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). All data was collected between April and August 1992 the release date of the present document being 2 December 1992. It is intended to release updates of the document as more data becomes available.

2. Problems Encountered

(a) 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 (and similarly for WRITE statements). Efficiency was 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, by using the second syntax form. 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.

(b) The SGI Fortran compiler version 3.4.1 miscompiled one routine when the full optimization level (-O2) was used (SUBROUTINE RCALXY). Accordingly this routine was compiled at level –O1 without problems. The 'bug' appeared in what seems to us to be a relatively simple section of scalar code, and has been reported to SGI. We have determined that the problem remains in more recent versions of the compiler..

(c) 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.

(d) 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. The Results

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 the table below we report the sum of system and user time (=Cpu-time) for part 1 (integral generation) and part 2 (energy minimization iterations).

Computer System	Cpu-time (Part 1) seconds	Cpu-time (Part 2) seconds
Convex C220	5816	1117
Cray XMP 4/16	760	123
IBM RS/6000 model 530	2006	1020
IBM RS/6000 model 530H	1499	765
IBM RS/6000 model 550	1205	612
HP/9000 model 750	1356	690
DEC/5000 model 200	8564	4849
SGI 4D/220	6104	2175
SGI Crimson	2363	829
SUN SS10 model 30	2904	1439