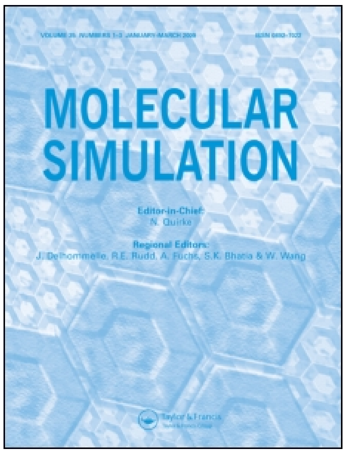

Molecular Simulation
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713644482

The Use of Supercomputers and Microcomputers in Computational Chemistry — Job Turnaround Time as a Criterion for the Choice of System

Stephen J. Zara[a]; David Nicholson[b]; James Barber[a]
[a] Department of Pure & Applied, Biology Imperial College of Science and Technology Prince, London, UK [b] Department of Chemistry, Imperial College of Science and Technology University of London, London, UK

To cite this Article Zara, Stephen J. , Nicholson, David and Barber, James(1988) 'The Use of Supercomputers and Microcomputers in Computational Chemistry — Job Turnaround Time as a Criterion for the Choice of System', Molecular Simulation, 1: 5, 299 — 307
To link to this Article: DOI: 10.1080/08927028808080951
URL: http://dx.doi.org/10.1080/08927028808080951

Molecular Simulation, 1988, Vol. 1, pp. 299–307
Photocopying permitted by license only
Reprints available directly from the publisher

THE USE OF SUPERCOMPUTERS AND MICROCOMPUTERS IN COMPUTATIONAL CHEMISTRY — JOB TURNAROUND TIME AS A CRITERION FOR THE CHOICE OF SYSTEM

STEPHEN J ZARA*, DAVID NICHOLSON** and JAMES BARBER*

**Department of Pure & Applied Biology Imperial College of Science and Technology Prince Consort Road London SW7 2BB, UK*
***Department of Chemistry Imperial College of Science and Technology University of London Exhibition Road London SW7, UK*

(*Received November 1987; in final form January 1988*)

The performance of a supercomputer installation (Cray-1S with Amdahl front-end) is compared with a 'desk-top' computer (IBM PC-AT with 80287 Numeric Data Processor) using both execution time and total job turnaround time for a set of benchmark programs which includes a Monte Carlo simulation. The effect of compiler efficiency and optimisation for system-specific hardware on execution time are discusssed. We propose the use of 'turn-around time' as a criterion for computer system performance, and show that judged by this criterion, 'desk-top' computing can provide a significant fraction of the power of a networked supercomputer installation.

KEY WORDS: Personal computers, supercomputers, simulation.

INTRODUCTION

Much current research in Physics and Chemistry involves intensive numerical work and demand from simulationists for computer power is seemingly insatiable. This requirement has traditionally been met by the provision of networked supercomputer facilities, using machines such as the Cray series. However, the effective power available to each user of such a facility may be only a small fraction of the total processor power, thus making smaller and considerably cheaper sources of computing competitive. In a typical networked installation, a supercomputer is connected to a host machine which interacts with users. Job entry is performed either directly, through the host machine or via a network. Such installations are heavily used, with limited access for each user.

At the same time as supercomputer facilities have been established and expanded, another source of processor power has arisen; the 'micro'. From humble origins in the late 1970s these devices have grown in power dramatically; the term 'micro' may be appropriate as a description of the size of these devices, but no longer applies to their processor power. With the establishment of the 'Industry Standard PC' by IBM in the early 1980's, it is possible to describe the level of computing power which is widespread; the AT ('Advanced Technology'). These machines contain a powerful

All correspondence to: Dr S J Zara

processor of the 8086 family [2], the 16-bit Intel 80286, and where extensive numerical work is required the performance of the AT can be greatly enhanced by the installation of a floating-point co-processor, the 80287 "maths chip" [3]. The AT machines usually run under the MS-DOS operating system, but a better invironment for scientific work, including software development, is provided by the 'XENIX' system a version of UNIX, as more memory is available for programs and data (up to 3 Mbytes), and extensive batch processing and time-share facilities are provided.

At first sight, access to a typical supercomputer facility would seem to be vastly superior to possession of a desk-top micro. However, a comparison between maximum processor speeds can be misleading for two reasons:

(i) The quoted peak MFLOP rates of supercomputers usually bear little relation to the calculation rates obtained for typical software, these being frequently an order of magnitude lower. The simpler, purely sequential, operation of the 80286 and 80287 microprocessors in AT machines means that optimum processor speeds can be obtained without sophisticated programming techniques and compilers, and a calculation rate of the order of 100 KFLOP is available continuously.

(ii) machine accessibility also imposes a limitation on effective supercomputer performance for the majority of users, who will typically enjoy no more than a small percentage of the machine's total operating time each. Such considerations make a comparison between a supercomputer and an enhanced desk-top micro realistic.

In this paper we compare the processing power of a supercomputer, the Cray-1S [1], and a desk-top microcomputer, the IBM AT. Techniques of software optimisation appropriate to each machine are discussed, and the processing speed of the machines are evaluated using a set of benchmark programs and Monte Carlo simulation software. Processor speed is measured in terms of both the execution time when access to the computer is available, and the total time taken for the task to be completed (turnaround time). The former unit is the more conventional measure, whereas the latter takes into account waiting times in queues, job transfers etc.

METHOD

2.1 Microcomputer Hardware

The desk-top computer system consisted of an IBM AT with 8 MHz clock. The processor of this machine is the Intel 80286, which can achieve a mean rate of 10^6 instructions per second for integer addition, integer subtraction and data movement operations. Integer multiplication and division can be performed at the rate of 1000 per second. The rate at which instructions are executed depends on the structure of a program. The sections of the processor which fetch instructions from memory works independently from the section which executes the instructions, and most of the time (with the exception of branch instructions) the two parts of the 80286 can be running in parallel. With the addition of a Numeric Data Processor (the Intel 80287) the system gains the power to perform operations on floating point values with a mean rate of the order of 100,000 FLOPs. The 80287 is particularly efficient at square root calculation; a benefit when coulombic potentials are evaluated. To a large extent NDP and main processor execute in parallel [3], so that while time-consuming floating point operations are being performed by the NDP, the main processor can be occupied in other work. The speed of the floating point processor was increased above that in the standard IBM AT by the addition of a Turbo-287 unit purchased from Microway

Inc. The memory of the PC AT was expanded to the maximum allowed under XENIX-286 – 3 M Bytes, using two AST Rampage expansion boards. On-line permanent storage consists of a 20 Mbyte hard disk. This provided ample capacity for restricted primitive simulation work (as in the program GCEMC2 used in benchmarking – section 2.5), but additional capacity for this type of micro is available through additional hard disk storage and external tape drives.

2.2 Microcomputer Software

The system included the XENIX-286 operating system (largely compatible with UNIX), which, unlike the standard IBM PC operating system (MS-DOS), gives access to several Mbytes of memory. Fortran77 programs were compiled using the Microsoft Fortran compiler, which accepted the ANSI X3.9-1978 Subset FORTRAN standard [4] (largely equivalent to Fortran77). The compiler can be directed to recognise the presence of the maths processor (80287) and include instructions recognised by this chip, or to code floating point operations using the integer arithmetic of the main processor (80286). Various degrees of optimisation can be performed by the compiler, to minimise either execution time and/or program size.

2.3 Supercomputer Hardware

The supercomputer used in this work was a Cray-1S sited at the University of London Computer Centre. This machine consists of a Cray processor with an 80 MHz clock, and 1 million 64-bit words (equivalent to 8 Mbytes). The Cray is equipped with eight 64-word vector registers which allow rapid execution of operations on arrays, the importance of which will be discussed later.

2.4 Supercomputer Software

The benchmark programs were compiled and run under the COS operating system [1]. The Cray-1S is accessible via remote job entry, either from the Amdahl front-end, or via the JANET U.K. academic network. Tasks to run the benchmarks were submitted through this network from a VAX 8600 at Imperial College Computer Centre, using the JTMP protocol. The Fortran77 compiler available was CFT 1.2 [1], which includes facilities for compiler-performed optimisation, particularly the use of the vector registers (see below).

2.5 The Benchmark Software

Three indices of computation power were measured for each computer:

T_{unopt} - the unoptimised execution time. For the IBM AT, the use of the floating point processor is disabled. For the Cray 1S, the use of vectorisation is disallowed. The purpose of this index is to reduce the use of hardward specific to each machine.

T_{opt} - the optimised execution time. The compilers are instructed to maximise the use of machine-specific hardware: the 80287 processor in the AT and the vector processors in the Cray 1S.

T_{total} - the total job turnaround time. The compilations are performed as for T_{opt}, and the total time taken from the initiation of the task to the completion, including time spent during transmission and waiting in queues. The ratio T_{opt}/T_{total} gives the fraction of the maximum computer power available to the task.

There are endless criteria that can be specified for software to test processor power. The benchmark software used in this work have been designed to incorporate the processes that are commonplace in our primary area of research: Monte Carlo Simulation of fluid systems, so the tests will be weighted towards array accessing (as in the storage and retrieval of particle coordinates) and the calculation of powers (as in force and distance determinations).

Five small Fortran77 programs have been written:

TRIG	evaluates trigonometric functions.
ROOTS	evaluates square roots.
NEWTON	uses Newton-Raphson iteration for square root calculation.
TABLE	Sums the elements of an array, first in row order, then in column order.
MATMUL	multiplies two matrices.

The processing power of the computers under test was also investigated using a Grand Canonical Ensemble Monte Carlo simulation program [5], which had been developed to model restricted primitive unsymmetrical electrolytes. The Fortran77 source code of the five benchmark programs are given in the Appendix. The program used 32-bit precision for floating point number storage on the IBM-PC, although the accuracy of the 80287 processor during calculations is 80 bits. 64-bit precision is available on the PC, with a consequent increase in calculation time. However, very long simulations have been run on the IBM-PC using 32-bit storage, and have yielded indentical results when run on the Cray with 64-bit precision, so the use of lower precision on the PC to improve program speed does not, for our software, result in rounding errors.

3. RESULTS

Table 1 gives T_{unopt}, T_{opt} and T_{total} for each of the five benchmark programs running on the IBM AT and on the Cray 1S, together with the ratio of T_{total} for the IBM AT to that for the Cray-1S is shown for each of the benchmark programs and for the simulation program. This ratio is a measure of the proportion of the available Cray power yielded by the IBM AT. The T_{total} values for the Cray in Table 1 are approximate, as the available power of a mainframe installation varies considerably with the number of users accessing the facility. The available performance of the Cray-1S at night and at weekends can be orders of magnitude greater than during peak usage times. Nevertheless, the performance at peak times is, by definition, that experienced by most users and is therefore a valid basis for comparison with other sources of computing power.

The values of T_{unopt} can be compared with T_{opt} to give an indication of how much speed improvement is obtained when the software is optimised by the compiler for the hardware specific to each computer. It is apparent that the use of the maths co-processor in the IBM AT can speed up software manyfold; the reasons for this will be discussed below. For the Cray, T_{unopt} was obtained by disabling use of the vector registers.

The data in Table 1 shows that the IBM desk-top machine can yield a significant

Table 1

Program	*Machine*	T_{unopt}	T_{opt}	T_{total}	T_{opt}/T_{total}	T_{total} *(Cray)*/ T_{total} *(PC)*
TRIG	IBM PC	19	5.1	5.1	1	
	Cray	0.073	0.073	33 (29)	0.010	6
ROOTS	IBM PC	29	19	19	1	
	Cray	0.26	0.058	6.6 (3.0)	0.021	0.4
NEWTON	IBM PC	4.7	2.7	2.7	1	
	Cray	0.021	0.020	2.7 (1.3)	0.011	1
TABLE	IBM PC	3.6	1.4	1.4	1	
	Cray	0.024	0.017	2.3 (0.6)	0.0021	2
MATMUL	IBM PC	260	99	99	1	
	Cray	0.50	0.11	16 (13)	0.013	0.2
GCEMC2	IBM PC	3000	1100	1100	1	
	Cray		43	220	0.2	0.2

GCEMC2 consisted of 1400 lines of FORTRAN 77 code for the Grand Canonical Ensemble simulation of restricted primitive electrolyte solutions confined by a model membrane. Vectorisable IF statements and similar CFT-specific coding was not employed in the Cray version for this study. Likewise, no machine specific "fine tuning" was applied in this study when running the program on the IBM machine.

Execution times (with and without optimisation) and total job times for five runs of each of the five benchmark programs, and for a 25,000 configuration run of the simulation program GCEMC2 are shown, in seconds, together with the proportion of total job time spent in execution (T_{opt}/T_{total}) and the proportion of Cray power provided by the IBM machine (T_{total} for the IBM machine / T_{total} for the Cray). The 95% confidence limits of the means are shown where the uncertainty effects the first two significant digits. The values in the final column are given to one significant digit only, as they were subject to very large variation.

fraction of the available power of a Cray-1S installation. It is at its most effective in comparison with the Cray for both very short and very long tasks. This is a consequence of the method of scheduling used on the Cray, where the user has to indicate the maximum execution time required by a task, and this time is used to determine the queue priority. Very long tasks (up to 20 minutes of computing time) may wait up to a day before commencing. However, even for the shortest tasks, which are started with minimal delay, the time spent waiting for, peripheral activities to occur (e.g. accessing of data files, printing of results) will can exceed the running time for the program. For 'medium-size' tasks, the time spent in queues may be only a few hours, and so the Cray gains a significant advantage over desk-top machines in such circumstances.

DISCUSSION

4.1 Optimisation

There are many ways in which the speed of software can be improved. The simplest method is to alter the source code to eliminate invariant expressions in loops, repeated array indexing etc. Methods for coding time-efficient software are well documented. Most modern compilers are able to re-organise source code in order to improve efficiency, and such optimisation can be selected when a compiler is started. Program speed can sometimes be improved by re-writing sections of the source code which are expensive on time in machine code. This is a specialised job, and the program loses portability, but substantial gains in speed are sometimes possible.

4.1.1 The IBM PC-AT

As previously described, the power of IBM AT machines can be considerably enhanced by the addition of a maths co-processor, the 80287. This device acts as an extension to the original processing unit, so that the combined processor can perform operations on floating-point numbers. Speed enhancement arises in two ways: firstly, calculations using the 80287, in which floating point operations are performed by hardware, are many times faster than with the 80286, in which floating point operations must be programmed. Secondly, the 80286 and 80287 can execute instructions in parallel, they are virtually autonomous processors sharing the same instruction stream. For example, in a matrix calculation, while the 80287 is performing a numerical operation on one matrix element, the 80286 can be locating and accessing the next element. Careful programming at the machine code level, or the use of optimising compilers can take advantage of this parallel capacity.

The performance of software can also be improved by careful organisation of data structures: the memory of PCs is grouped into 64 kbyte segments, and switching between segments takes time. However, for the benchmarks, no such optimisations were attempted.

4.1.2 The Cray-1S

The Cray series of computers achieves high performance through two processing techniques: vectorisation and chaining. With vectorisation, a single instruction can initiate a repeated operation on a collection of data (the vector). In chaining, a series of operations on a set of data are organised so that much of the work is overlapped. The presence of multiple sets of vector registers in the Cray permits data paths between these registers to be used simultaneously. The combination of vector registers and chained operations is the key to the efficient use of the Cray's architecture.

4.2 The Importance of Hardware and Software Support

There are disadvantages to the use of stand-alone microcomputers as substitutes for centralised supercomputers. A supercomputer facility usually consists of more than just the computer; libraries of software are provided together with advisory services. Such libraries (NAG, for example) are becoming available for micros, albeit slowly, but the programmer will have to be self-reliant in the face of software development problems. Where on-line access to either the supercomputer or front-end machine is possible, the initial phases of software development — compilation and debugging — may be performed at a rate that far exceeds the capacity of a PC: Such tasks may require individually only a few seconds of processor time, and consequently have a rapid throughput on the supercomputer. If such access is feasible, then an immense amount of tracing and debugging information can be rapidly acquired.

4.3 Future Developments in Microcomputers

With advancing technology the power of both super- and micro-computers is increasing; as large machines become able yield gigaFLOP rates (maximum power), the newer micros routinely yield more than 500 kFLOP (continuous rate), and the new generation of Reduced Instruction Set Chip (RISC) technology machines, typified by the Floating Point Transputer (T800) and the Acorn RISC Machine (the processor in the 'Archimedes' — the latest BBC micro), promise processor rates exceeding the MFLOP rate by a wide margin.

Note added in proof: Recent experience with a single (T800) Transputer incorporated as a plug-in card in our IBM (The Microway Monoputer) has shown a fifteen-fold improvement in speed over the PC rates quoted in Table 1. It seems likely, therefore, that the gap in power between microcomputers and supercomputers will narrow.

5. CONCLUSIONS

In conclusion, the processor speed available for a computeration may be more usefully defined as the mean calculation rate over the total job turnaround time T_{total}, rather than the maximum power for a sole user T_{opt}. Such a definition, which is after all an accurate measure of the computing facility available to each user, reveals that 'Personal Computing' as typified by the desk-top machines can be surprisingly powerful for numerical computation, and can often prove more effective than the purchase of supercomputer time.

APPENDIX

Fortran77 source for the five benchmark programs:

TRIG

```
      PROGRAM TRIG
      REAL CX,SX,TX,X
      INTEGER I
      DO 10 I=1, 360
        X=FLOAT(I)/180.*3.1415926
        CX=COS(X)
        SX=SIN(X)
        TX=TAN(X)
10    CONTINUE
      END
```

ROOTS

```
      PROGRAM ROOTS
      REAL SUM
      INTEGER I
      SUM=0.
      DO 10 I=1,100000
        SUM=SUM+SQRT(FLOAT(I))
10    CONTINUE
      END
```

NEWTON

```
      PROGRAM NEWTON
      REAL R
      INTEGER I
      DO 10 I=1,1000
        R=ROOT(FLOAT(I))
10    CONTINUE
      END

      FUNCTION ROOT(X)
      REAL OLDVAL, NEWVAL
      OLDVAL=1
1     NEWVAL=(X/OLDVAL+OLDVAL)/2.
      IF(ABS(OLDVAL-NEWVAL).GE.1E-6)THEN
        ·OLDVAL=NEWVAL
      GOTO 1
      ENDIF
      ROOT=NEWVAL
      RETURN
      END
```

MATMUL

```
        PROGRAM MATMUL
        INTEGER I,J,K
        REAL SUM, MAT1,MAT2,MAT3
        COMMON/MATRIX/ MAT1(100,100),MAT2(100,100),
      + MAT3(100,100)
        DO 10 I=1,100
          DO 20 J=1,100
            MAT1(I,J)=I+J
            MAT2(I,J)=I-J
20        CONTINUE
10      CONTINUE
        DO 30 I=1,100
          DO 40 J=1,100
              SUM=0.
              DO 50 K=1,100
                SUM=SUM+MAT1(I,K)*MAT2(K,J)
50            CONTINUE
      MAT3(I,J)=SUM
40        CONTINUE
30      CONTINUE
      END
```

TABLE

```
      PROGRAM TABLE
      REAL T(100,100)
      INTEGER I,J
      REAL SUM1,SUM2

      DO 10 I=1,100
        DO 20 J=1,100
            T(I,J)=I+J
```

```
20      CONTINUE
10    CONTINUE

      SUM1 = 0.

      DO 30 I = 1,100
        DO 40 J = 1,100
          SUM1 = SUM1 + T(I,J)
40      CONTINUE
30    CONTINUE

      SUM2 = 0.

      DO 50 J = 1,100
        DO 60 I = 1,100
          SUM2 = SUM2 + T(I,J)
60      CONTINUE
50    CONTINUE
      END
```

Acknowledgements

This work was funded by a joint SERC/Unilever Grant. We wish to thank Unilever for funding the purchase of computing equipment.

References

[1] Lazou, C. (1983) Workshop notes on effective use of the Cray-1S/1000 at ULCC. University of London Computer Centre.
[2] Thorne, M. (1986) in: "Programming the 8086/8088 for the IBM PC & Compatibles". The Benjamin/Cummings Publishing Company Inc., California.
[3] Palmer, J.F. and Morse, S.P. (1984) in "The 8087 Primer", Wiley Press, New York.
[4] Katzan, H. (1978) "Fortran 77". van Nostrand Reinhold Co., New York.
[5] Emshoff, J.R. and Sisson, R.L. (1970) in "Design and Use of Computer Simulation Models". Macmillan, New York.