N87-26555

EXPERIENCE IN HIGHLY PARALLEL PROCESSING USING DAP

D. Parkinson
DAP Support Unit
Queen Mary College
University of London
London, England

ABSTRACT

DAP systems have been in day to day use for 10 years and a large amount of user experience has been gained. The profile of user applications is similar to that of the MPP working group.

Experience has shown that contrary to expectations, highly parallel systems provide excellent performance on so-called "dirty problems" such as the 'physics' part of meteorological codes. The reasons for this observation are discussed. The arguments against replacing bit processors with floating point processors are also discussed.

Keywords: DAP, bit serial processors, FORTRAN, balance.

INTRODUCTION

The DAP [2] is generally similar in concept to MPP but differs in some important ways. Probably the most ways in which the DAP differs from MPP are:

a) Memory size : The system view of DAP has always been that of a memory module with processing capability and so emphasis has always been placed upon having the largest memory it was practical to use when the systems were designed. Thus the 64x64 DAP when delivered to QMC had 4K bits per PE, but this was enlarged to 16K bits per PE when the appropriate static RAMs became available at reasonable prices. User experience has shown that many applications are very memory hungry and there is little limit to the memory which users can exploit. Plans for DAP-3 have a minimum memory of 64K bits per PE rising through 256K bits per PE to a current maximum of 1Mbit/PE giving a potential total of 128M bytes of a 32x32 DAP-3 or ½G bytes on a 64x64 system.

b) Interconnection Network : The DAP DAP has a 2D NEWS network similar to that of the MPP but additionally has a second layer network of row and column highways. These highways provide facilities to rapidly fetch and broadcast data. The highways are connected to edge registers in the master control unit and permit data to be selected from any set of processors, one per row/column or broadcast to any of the processors in the row/column. As the same edge registers are used for both row and column highways, 'corner turning' is also supported. The highways are not 'buses' but are essentially multi-way OR gates and so give the DAP the properties of an 'associative processor' [3].

c) Software Systems : An extended FORTRAN dialect is the principal programming language for DAP use [4]. This language supports array processing in data sets with either N or N^2 items on an N*N DAP. Reals with 3, 4, 5, 6 or 7 and 8 byte precisions are fully supported, also available are 1-8 byte integers and bit variables (logicals). The flexibility and efficiency of the DAP-FORTRAN language is such that users very rarely need to use the assembler language APAL. Indeed, the use of APAL is actively discouraged and is only justified for specialist bit manipulation algorithms arising in such areas as low precision image processing and customised-format arithmetic routines - floating point users never need APAL.

Above the high level system language DAP-FORTRAN there exists a subroutine library structured on the NAG line containing a wide variety of useful routines. The most widely used routines include Random Number generators, FFT's, tridiagonal equation solvers, sorting routines and a number of general utilities for data re-arrangements. The performance of the random number routines is particularly spectacular [5].

USER EXPERIENCE

The DAP has been used for a wide range of applications in the scientific field. The range of topics is almost identical to that covered at this conference, so there is little point in detailing it here. The application area most represented is Monte Carlo Simulations in such areas as QCD, Ising models, molecular dynamics etc., much to the surprise of many so-called experts who felt that SIMD architectures were unsuitable for Monte Carlo work. In addition to scientific (floating point) work, the DAP has been used for a number of non-numeric problems including:
- searching for large primes
- textual data abstraction
- information retrieval, and
- graphics - in particular, molecular graphics.

The bit serial nature of the processing elements has helped greatly in providing the flexibility to support more than just specialist number crunching.

"DIRTY PROBLEMS ARE NICE!"

The reaction of users to the DAP has been very favourable especially to reprogramming exercise. The expressive power of DAP-FORTRAN has allowed great reductions in program size and complexity, and much higher performance than expected for so-called "dirty problems", i.e. problems requiring many data dependent decisions.

There is a natural tendency to assume that problems with many decisions are not optimum for the DAP/MPP type of computers as the natural way to implement them is by parallel conditional expressions of the type

```
where (temperature<freezing point)
    Heat:=heat + latent_heat
```

In DAP-FORTRAN this would be coded as

```
HEAT(TEMPERATURE.LT.FREEZING POINT)
= HEAT + LATENT_HEAT
```

the array indexing type is called MASK-INDEXING.

As only some processors will update their local values of the variables, the natural tendency is to assume that such code will demonstrate a lower speed-up over 'conventional' computers than, say, unconditional code. Our experience is that the reverse is the case and tasks of the above nature demonstrate better speed-up than the clean unconditional code!

The somewhat paradoxical result is explained when one realises that one is comparing the performance of two machines rather than measuring some absolute computing performance, and what one is really observing is that arrays of bit-serial processors are better suited to "dirty code" than are 'conventional' word based architectures. The explanation is simple but difficult for many people to grasp. Bit-serial systems are primarily optimal for dealing with arrays of logical operations, as these are essential single bit operations. Arithmetic is software produced out of lots of logical primitives, hence the worst performance of the system comes from pure floating point code! The greater the proportion of a problem that is purely logical, the greater the potential is for an array of bit-serial processors to outshine architectures based on floating point pipelines - which try to avoid conditionals.

SYSTEM BALANCE

Some users of DAP have argued that the system would be improved if the bit-serial processors were replaced by floating point units and it is not surprising to me to find a number of the MPP group making the same observation. Care should be taken in evaluating what is really being stated. There are three arguments which can be made against such changes.

1) 'Dirty problems' will really become dirty as the arguments in the previous section become invalid.
2) The parallelism in systems of equal cost becomes much less. The cost (in say mm^2 of silicon) of a 64 bit floating point processor is equivalent to the cost of more than 128 single bit PE's, hence one should realise that the option one is really comparing is that of say, 16K, 1 bit processors and 128 - 64 bit processors.
3) The inter-processor communication rate should be somehow in "balance" with the processor computational capabilities. I have attempted to put some flesh on this concept of balance [6], which I summarise here.

For many parallel algorithms - especially those with results producing global properties of arrays, the computation is carried out in a loop structure of the form:

```
K:=1;
WHILE K<N begin
  move some data a "distance" K;
  perform a computation;
  K:=K*2
  END WHILE
```

typical of such loops are - SUM, Product, Maximum, Minimum, FFT, Sorting etc.

The loop is traversed $\log_2 N$ times

so the cost is:

$(\log_2 N)$ cost of a computational step +

cost of shifts 'distance' 1+2+4+8.... On a mesh connected processor with edge size n and dimensionality d and

$N=n^d$ (for MPP n = 128, d = 2), the cost of such shifts become $d(n-1)*$ cost of moving a data item from one processor to its neighbout.

The Balance Factor can be defined by:

$$B = \frac{\text{time for computational component}}{\text{time for data movement component}}$$

if B = 1 the two costs are equal, i.e. the system is in balance. If $B>>1$, the computational time is dominating and we would gain in performance if more powerful processors were used. If $B<<1$, data movement costs dominate and little benefit would appear if the processors were speeded up without improving the data transmission speed.

G = 1/B could be called the Granuality factor of the system as it gives an idea of the minimum recommended number of computational operations that must be performed per data transmission step.

Computation of balance factors for systems such as MPP and DAP suggests that for floating point operations the systems are in approximate balance and no benefit would be gained by using floating point PE's unless the interprocessor network were improved by multibit highways and/or higher dimensionality meshes.

The bit serial processors approach seems to be an optimal route to providing very highly parallel systems which can support a large range of application areas.

Although one could try and produce the computer with the worlds largest MFLOP rating by building arrays of

2^{16} floating point chips, there is little evidence that such machines would produce even 0.1% of their theoretical performance on anything other than highly artificial problems such as the Mandelbrot Set.

REFERENCES

1. Flanders, P.M., Hunt, D.J., Reddaway, S.F. and Parkinson, D, "Efficient High Speed Computing with the Distributed Array Processor", (High Speed Computer and Algorithm Organisation) Academic Press 1977.

2. Parkinson, D. "The Distributed Array Processor" (Computer Physics Comm. Vol.28, pp. 335,) 1983.

3. Foster, C.C. "Content Addressable Parallel Processors", (Van Nostrand), 1976.

4. International Computers Ltd., Technical Manual TP 6918,"DAP-FORTRAN" (see Ref. 2 for brief description)

5. Smith, K.A., Reddaway, S.F., Scott, D.M., "Very High Performance Pseudo-Random Number Generation on DAP", (Vector and Parallel Processors), North Holland, 1985.

6. Parkinson. D, "Organisational Aspects of Using Parallel Computers" (International Conference on Vector and Parallel Computing, Loen, Norway, June 2-6, 1986). To be published.