JOURNAL OF COMPUTATIONAL PHYSICS 55, 340–345 (1984)

Letter to the Editors

Solving Very Large Elliptic Problems on a Supercomputer with Solid State Disk

An experiment is described that demonstrates how secondary solid state storage devices extend effectively the capabilities of supercomputers to solve large computational problems. A linear system of size 45^3 resulting from the 27-point finite difference discretization of a three-dimensional elliptic operator is solved by a conjugate gradient algorithm on one processor of a Cray X-MP with attached 8M-word Solid State Disk (SSD). The main matrix was stored on the SSD. Although for the 8M-word SSD the problem is I/O bound, it is projected that the 32M-word SSD will support the computation at its full measured speed of 161 MFLOPS. It is concluded that the solution of a $90 \times 90 \times 90$ 27-point finite difference problem will require about 0.25 sec per iteration on the Cray X-MP, which will make three-dimensional problems of reasonable size feasible.

INTRODUCTION

The computational cost of three-dimensional models has placed most such computations beyond the reach of the most powerful supercomputers. Computational speed poses one bottleneck; however, often the real bottleneck is the limited memory size of these machines, typically up to 4 million 64-bit words of fast memory. Supercomputers are slowed down severely when computations have to be performed from disk, seriously limiting the size of problems that are feasible to solve.

The advent of affordable large MOS memories may ease this bottleneck. Although MOS memory is not quite as fast as ECL memory that has been used in supercomputers so far, memory interleaving and the combination of MOS memories with ECL memories in a hierarchical fashion can make them very useful for large computational problems.

The work described in this paper was designed to show that three-dimensional problems of reasonable size may have indeed come within our reach. Cray Research, Inc. (CRI) is currently marketing a solid state disk (SSD) as a secondary bulk storage device for use with its Cray X-MP. The SSD has up to 32 million 64-bit words and is connected to the main ECL memory of the machine through a dedicated port and high-speed I/O channel. According to the CRI hardware specifications, it is capable of transferring up to 132 Mword/sec. It is our estimate that such a device should support even linear algebra routines with major arrays stored outside of main memory at the maximum computational speed of the supercomputer.

One of the most I/O intensive problems is the solution by iterative methods of a linear system of equations that arise from a finite difference approximation of elliptic

0021-9991/84 $3.00

partial differential equations if the system is too large to be contained in main memory. With the main matrix stored on the SSD, only one add and one multiply operation are performed per transferred word. However, the hardware specifications indicate that it should be possible to perform suitable relaxation techniques at a rate approaching 200M floating-point operations per second (MFLOPS) from the SSD. Furthermore, this processing rate, a 4M-word main memory, and a 32M-word SSD should permit the relaxation of three-dimensional grids of size nearly $100 \times 100 \times 100$ using a 27-point finite difference approximation in a few tenths of a second per iteration.

To satisfy ourselves that this conjecture was correct, we designed an experiment that

(1) measures the computational rate for such an elliptic solver, and

(2) measures the I/O rate in terms of startup (access) and transfer times.

Experimental Design

The model chosen for the experiment is one for solving by iteration a linear system that arises from a 27-point discretization of a three-dimensional elliptic PDE. We have chosen for simplicity to solve Laplace's equation with Dirichlet boundary conditions. However, no advantage is taken of the fact that it is a separable problem with constant coefficients. In fact, all nonzero elements of the matrix are stored on the SSD so that the findings can be applied to a more general operator of 27-point type.

We use the conjugate gradient algorithm with a polynomial preconditioner [1] to solve $Ax = b$, where A is symmetric and positive definite of order n^3, and n is the number of points on one side of the grid cube. The dominant operation is matrix multiplication, which is used in computing the residual and in the polynomial evaluation. The matrix A is stored on the SSD, and vectors x, b, and auxiliary vectors needed for the conjugate gradient algorithm are kept in the main memory. To overlap the computation with I/O operations, we use two I/O buffers in main memory in the following way: while the CPU draws operands for its computations from one of them (the COMPUTE buffer), new data are being read from the SSD into the other one (the READ buffer). This transfer operation does not require the services of the CPU except for startup. Buffers are switched cyclically after the current READ and COMPUTE buffers have been processed.

The matrix A is stored in blocks of k rows of A containing the 27 diagonals in a $(k, 27)$ array. The matrix multiplication is carried out as multiplication by diagonals, that is

$$y = \sum_{l=-13}^{13} a_l X_l,$$

where the main diagonal is labeled a_0. The diagonals of A below (above) the main diagonal are labeled with negative (positive) indices in decreasing (increasing) order

away from the main diagonal. The X_l denote the appropriate subsets of X. The computational vector length is k and the I/O vector length is $27k$.

The matrix multiplication begins with a non-overlapped I/O operation, to fill the initial buffer, and ends with a non-overlapped computation involving the last k rows of the matrix. In between, the following set of operations is performed repetitively: an I/O startup followed by overlapped data transfer and computation on k rows. Except for small corrections for beginning and end, the total time for the matrix multiply will be the sum of I/O overhead time and the larger of the two times required for data transfer and computation.

For an 8M-word SSD the data transfer is expected to take more time than the computation. With 53 floating-point operations per row, we expect the following times for the matrix multiplication:

$$\text{Wall Clock:} \quad t_w = N \cdot t_0 + 27 \cdot n^3/R_t$$

$$\text{User CPU:} \quad t_u = 53 \cdot n^3/R_c$$

$$\text{System CPU:} \quad t_s = N \cdot t_0$$

where N is the number of I/O requests, t_0 the I/O overhead time per request, n^3 the size of the matrix, R_t the SSD transfer rate in words/sec, and R_c the computational rate in FLOPS.

For a 32M-word SSD, which contains four times as many memory banks as the 8M-word SSD and is therefore four times faster, the computations are expected to take more time than the data transfer. Therefore the wall clock time for the matrix multiply should be given by

$$t_w = N \cdot t_0 + 53 \cdot n^3/R_c,$$

with user and system CPU times the same as those for the 8M-word SSD.

An upper bound of the compute rate R_c for the matrix multiply can be obtained by considering that two loads, a multiply, and an add for a vector length of 64 can be repeated at intervals of 75 cycles if one vector register is used repeatedly in each add operation [2]. Seventy-five cycles with a cycle time of 9.5 nsec for 128 floating point operations correspond to a computational rate of 180 MFLOPS. However, the actual rate will be lower because of less than optimal scheduling by the compiler, setup times for the outer vector loops, and subroutine calls.

RESULTS

Only an 8M-word SSD was available at the time of our experiment. In addition, we were limited to 1M words of main memory on the Cray X-MP. Table I gives a portion of the raw timing data obtained by running the polynomial conjugate gradient algorithm for a matrix of size 45^3, polynomial degree 1 to an accuracy of approx-

TABLE I

Polynomial Conjugate Gradient Solver Timings (in Seconds) and Effective Execution Rates (in MFLOPS) on Cray X-MP with Matrix Stored on 8M-Word SSD[a]

I/O Buffer Size (words) Number of I/O Requests	$27 \cdot 1024$ 7832	$27 \cdot 2048$ 3960	$27 \cdot 4096$ 2024
Measured times for total algorithm			
Wall clock	12.2 sec 43 MFLOPS	10.1 sec 52 MFLOPS	9.09 sec 58 MFLOPS
User CPU	3.50 sec 151 MFLOPS	3.39 sec 156 MFLOPS	3.33 sec 159 MFLOPS
Measured times for matrix multiply only			
Wall clock	11.54 sec 37 MFLOPS	9.47 sec 45 MFLOPS	8.45 sec 50 MFLOPS
User CPU	2.70 sec 157 MFLOPS	2.66 sec 160 MFLOPS	2.64 sec 161 MFLOPS
Inferred times			
System CPU time	4.57 sec	2.31 sec	1.18 sec
I/O transfer time	6.97 sec	7.04 sec	7.20 sec

[a] Matrix size: 45^3, maximum error: 1.3×10^{-7}, number of iterations: 87, polynomial degree: 1, number of matrix multiplies: 88.

imately 10^{-7} on one processor of the Cray X-MP. These runs involved 87 iterations and 88 matrix multiplies each. Values of $k = 1024$, 2048, and 4096, respectively, were used for the I/O buffer sizes in this experiment. The program was written in Fortran. The computational rate for the matrix multiply is 161 MFLOPS for the largest buffer size, about 10% lower than the optimal rate. This was achieved by inserting parentheses to force the compiler to sequence multiplies and adds correctly.

It is evident that for the 8M-word SSD the matrix multiply is I/O bound, as expected. The wall clock time for the matrix multiply minus the time for the last non-overlapped computation is a linear function of the number of I/O requests. The slope corresponds to an I/O overhead time of 583 μsec per I/O startup, comparable to overhead times measured for disk operations.

The time needed for the actual data transfer is 7.20 sec and corresponds to an SSD transfer rate R_t of 31.1M word/sec, about 3% lower than the measured rate of 32.1M word/sec without overlapped vector computations. This reduction in the transfer rate might be due to memory bank conflicts.

Using the above data, it is not difficult to estimate the run times of the conjugate gradient algorithm for a full-size SSD of 32M words. Because the matrix multiply will be compute bound, the user will not have to give up the CPU between I/O operations. This usually implies less overhead time. We measured 410 μsec per read

TABLE II

Polynomial Conjugate Gradient Solver Timings (in Seconds) and Effective Execution Rates (in MFLOPS) on Cray X-MP with Matrix Stored on 32M-Word SSD[a]

I/O Buffer Size (words) Number of I/O Requests	27 · 1024 7832	27 · 2048 3960	27 · 4096 2024
Projected times for total algorithm			
Wall clock	6.57 sec 80 MFLOPS	4.96 sec 106 MFLOPS	4.19 sec 126 MFLOPS
User CPU	3.50 sec 151 MFLOPS	3.39 sec 156 MFLOPS	3.33 sec 159 MFLOPS
Projected times for matrix multiply only			
Wall clock	5.93 sec 72 MFLOPS	4.32 sec 98 MFLOPS	3.55 sec 119 MFLOPS
User CPU	2.70 sec 157 MFLOPS	2.66 sec 160 MFLOPS	2.64 sec 161 MFLOPS
Inferred times			
System CPU time	3.21 sec	1.62 sec	0.83 sec
I/O transfer time	1.74 sec	1.76 sec	1.80 sec

[a] Matrix size: 45^3, maximum error: 1.3×10^{-7}, number of iterations: 87, polynomial degree: 1, number of matrix multiplies: 88.

request in a separate experiment. Assuming that the data transfer proceeds at four times the measured rate of an 8M-word SSD and that overhead times are the same, we projected the execution times shown in Table II. It is evident that for the 32M-word SSD, the speed of both computation and data transfer make large I/O buffers mandatory to avoid excessive delays by I/O overhead. For the largest buffers chosen, the overall rate, including I/O, would be 126 MFLOPS.

If both processors of the Cray X-MP are used to perform the computations, an experiment we plan for the near future, the matrix multiplication will again become I/O bound.

For comparison, we ran the same experiment with the main matrix stored on disk. The total runtime of the linear solver was 453 sec, corresponding to a net computational rate of 0.9 MFLOPS for the matrix multiply. The disk-bound computation is therefore more than 50 times slower than the computation with the matrix on an 8M-word SSD.

Conclusions

Our experiment demonstrated that an 8M-word SSD attached to a Cray X-MP as a user-accessible, secondary storage device could support at a rate of 58 MFLOPS

the solution of a large linear system by a conjugate gradient method with the main matrix stored on the SSD. Although, the computation proceeded at a rate of 161 MFLOPS, I/O slowed it down by a factor of 3 to 4. Our projections show that a 32M-word SSD will support this and similar computations at close to full speed.

It would be helpful to reduce I/O overhead times for the SSD to avoid the need for very large I/O buffers.

If I/O overhead times can be reduced to a small fraction of the times required for the computations, then an estimate for an iteration on a $90 \times 90 \times 90$ grid would be $(8 \cdot 2.64/88)$ sec, or approximately one quarter of a second. This may translate into a few seconds to solve such a system to a few digits of accuracy. These results constitute a speedup by a factor of about 200 compared with computations performed from disk, and demonstrate the great usefulness of MOS memory devices for large-scale computations.

ACKNOWLEDGMENTS

We wish to thank David Slowinski of Cray Research, Inc., for writing the asynchronous I/O routines for the SSD for us and for his valuable advice during the course of this experiment.

REFERENCES

1. T. L. JORDAN, Conjugate gradient preconditioners for vector and parallel processors, *in* "Proceedings, Monterey Conference on Elliptic Problem Solvers," Academic Press, New York, 1983, in press.
2. Cray X-MP Series Mainframe Reference Manual, CRI publication HR-0032, Cray Research, Inc., 1982.

RECEIVED: June 22, 1983; REVISED: September 27, 1983

INGRID Y. BUCHER
THOMAS L. JORDAN
Los Alamos National Laboratory,
Los Alamos, New Mexico 87545