PIERS Timings on Various Parallel Supercomputers

Philip T. Keenan
Jon Flower

CRPC-TR93327
August 1993

Center for Research on Parallel Computation
Rice University
P.O. Box 1892
Houston, TX 77251-1892

PIERS Timings on various Parallel Supercomputers

Philip T. Keenan* Jon Flower†

August 2, 1993

Abstract

PIERS is a Parallel Implicit Research Reservoir Simulator. It was made available to us as a test program, designed for experimentation rather than production, yet incorporating many features of realistic petroleum reservoir simulators. This note reports on the results of timing PIERS on five parallel supercomputers including the iNTEL Touchstone DELTA and the Connection Machine 5 (CM-5). Speedup graphs illustrate performance as a function of the number of processors for a variety of problem sizes. The results indicate that parallel computation can provide substantial speedup in reservoir simulator codes. Moreover the raw timings provide an interesting comparison of machine performance — though one must remember that they apply directly only to PIERS, and may vary with different simulators, compilers, operating systems and hardware configurations.

*Department of Computational and Applied Mathematics, Rice University
†ParaSoft Corp.

1 Introduction

PIERS is a Parallel Implicit Research Reservoir Simulator originally developed by John Wheeler and Richard Smith at Exxon Production Research Company[2]. It was made available to us as a test program, designed for experimentation rather than production, yet incorporating many features of realistic petroleum reservoir simulators. This note reports on the results of timing PIERS on five parallel supercomputers including the iNTEL Touchstone DELTA and the Connection Machine 5 (CM-5). Speedup graphs illustrate performance as a function of the number of processors for a variety of problem sizes. The results indicate that parallel computation can provide substantial speedup in reservoir simulator codes. Moreover the raw timings provide an interesting comparison of machine performance — though one must remember that they apply directly only to PIERS, and may vary with different simulators, compilers, operating systems and hardware configurations.

2 PIERS

PIERS was originally developed for the iNTEL iPSC/2 Hypercube, a 16 node machine with 1 megabyte of data storage per node, and relatively fast communications relative to its computational speed[2]. One should note that the time and space constraints of the iPSC/2 forced some design decisions which might not be optimal on other machines. We converted it to run on a variety of parallel computers and timed it on each. However, we did not attempt to rewrite the code to optimize it for any particular target machine.

PIERS solves the two phase (oil and water) equations for slightly compressible flow in a three dimensional porous medium. These equations form a coupled system of nonlinear advection-diffusion type partial differential equations. PIERS uses a fully implicit numerical method rather than the frequently used IMPES (implicit pressure, explicit saturation) formulation. It accepts very general problem descriptions. Porosity and permeability can be functions of position; relative permeabilities and capillary pressure curves can be provided from empirical data; and multiple vertical injection and production wells can be used, with specified time varying bottom hole pressure boundary conditions or total flow rate conditions. The reservoir shape is specified by a possibly irregular areal cross section, which is duplicated at each vertical layer. PIERS uses standard rectangular finite differences and a data decomposition approach to parallelization.

3 Methodology

If a computation on an N processor parallel computer takes T_N time units, one might hope that

$$T_N = T_1/N.$$

The ratio T_1/T_N is called the *speedup* from N processors. Unfortunately, communication costs and other overheads preclude linear speedups in practice. However, in partial differential equation solvers these overheads often scale more slowly than the computational requirements, leading to improved speedups for larger problem sizes.

PIERS assigns processors to approximately equally sized subsets of a horizontal cross section of the reservoir. Each processor is then said to own the vertical cylinder-like subset of grid blocks above and below its portion of the cross section. Thus on average each processor communicates with four neighbors, though depending on the geometry a given processor might communicate with fewer. Each processor computes based on the $O(n^3)$ grid blocks it owns, but only needs to exchange $O(n^2)$ pieces of boundary information with its neighboring processors. Thus, for a fixed number of processors, communication effects should become relatively less important at a rate of $O(1/n)$ as the problem size (which is $O(n^3)$) increases. Conversely, for a fixed size problem communication effects become increasingly dominant as the number of processors increases.

To study these effects we tested each machine by running PIERS on 5 different size problems as shown in Table 1.

Data set	Layout	Grid Blocks
ptk1	4 x 18 x 13	756
test7	6 x 24 x 24	3,456
test7m	12 x 36 x 36	15,552
test7l	18 x 48 x 48	41,472
t100	30 x 100 x 100	300,000

Table 1: Problem Sizes

For each problem size we varied the number of processors p. Typically p was a perfect square and processors were assigned in a square layout, in order to keep

the symmetry of the problem constant as the number of processors increased. In some cases additional values of p were used. The timing graphs illustrate the often substantial effect of changing aspect ratio. This appears to be due in part to the nature of the linear solver and in part to cache effects on each microprocessor.

We compared PIERS on the 5 parallel supercomputers listed in Table 2.

Vendor	Machine	Processors
iNTEL	iPSC/i860 hypercube	64 i860's
iNTEL	Touchstone DELTA	512 i860's
IBM	Power Visualization System	32 i860's
nCUBE	nCUBE/2	64 custom chips
TMC	Connection Machine CM-5	64 Sparc's

Table 2: Machines

All of these machines except the IBM use distributed memory. The IBM uses shared memory. The CM-5 lacked vector boards. Non-vectorizing optimization levels (`-O2`) were used in all cases; timings should improve if vectorization was selected on those machines which support it.

PIERS uses blocking sends and receives rather than the sometimes more efficient non-blocking versions. To port PIERS to a variety of parallel machines we converted it to use Express[1], a commercially available portable communications library from ParaSoft. Timings on the iNTEL hypercube with and without Express were identical.

4 Timings and Speedups

The following graphs present the timings and speedups obtained for each machine and data set, as a function of the number of processors.

For instance, Figure 1 gives the step time on the CM-5 in wall clock seconds. On the CM-5 nodes are time-shared among applications. However, the timing functions on the CM-5 automatically compensate for this, yielding the same timings as would be obtained on an otherwise empty machine. The error bars indicate the range of times observed during multiple runs.

Figure 2 gives the speedups obtained on the CM-5. Since PIERS was not designed to run on fewer than 4 processors, speedups are based on an estimate of the

best sequential time obtained by extrapolation from internal timings of the communication and computation portions of the code.

Figure 3 and Figure 4 present the analogous results on the Delta machine. Graphs for the other machines are similar except for the scale. Scale comparisons will be explored below.

> **Disclaimer:** While the timings were run from shell scripts to ensure uniformity across machines, it is possible that there are incorrect timings in the data, due to human and/or machine error. Moreover, timings were observed to vary in response to many factors beyond our control, particularly hardware and software upgrades. Factors such as operating system changes, compiler optimization level and capability changes, and changes in the message passing hardware and clock rate can all impact timings substantially. Thus the timings, speedups and machine comparisons presented here should be taken as a rough approximation only. Moreover, the results are specific to PIERS and may be different for other programs. In particular, cache effects depend strongly on problem size and the locality of the data access patterns in the program.

4.1 Machine Comparisons

Subject to the warnings given in the disclaimer above, we think it is interesting to compare the performance of the various machines. We selected the largest test case, `qt100`, for this comparison. On smaller problems, the smaller and slower machines are likely to do better, since there is not enough computational work for large numbers of fast processors.

Figure 5 presents a log-log view of the timings on each machine. An ideal machine with no communication overhead running a perfectly parallel algorithm would produce a line with slope -1. All of the machines are reasonably close to this.

Among the three `i860` based machines, the Delta and the iPSC/i860 hypercube were very close in speed. The Delta is seen to have slightly faster communications and so can efficiently apply more processors to the fixed size problem. The shared memory IBM is seen to be slower than the two other distributed memory machines, despite using the same micro-processors. This may be due to bus contention, since all the processors compute and communicate more or less in sync with each other.

The CM-5 was slower than the Delta machine, but its timings should improve when vector boards are available.

It appears that the `i860` processors ran about 3 times faster than the nCube custom processors, but since the nCube is significantly less expensive than the iNTEL machines, this is not surprising.

We are in the progress of porting PIERS to the iNTEL Paragon, which is the commercial successor to the DELTA machine; timings for it should be available shortly.

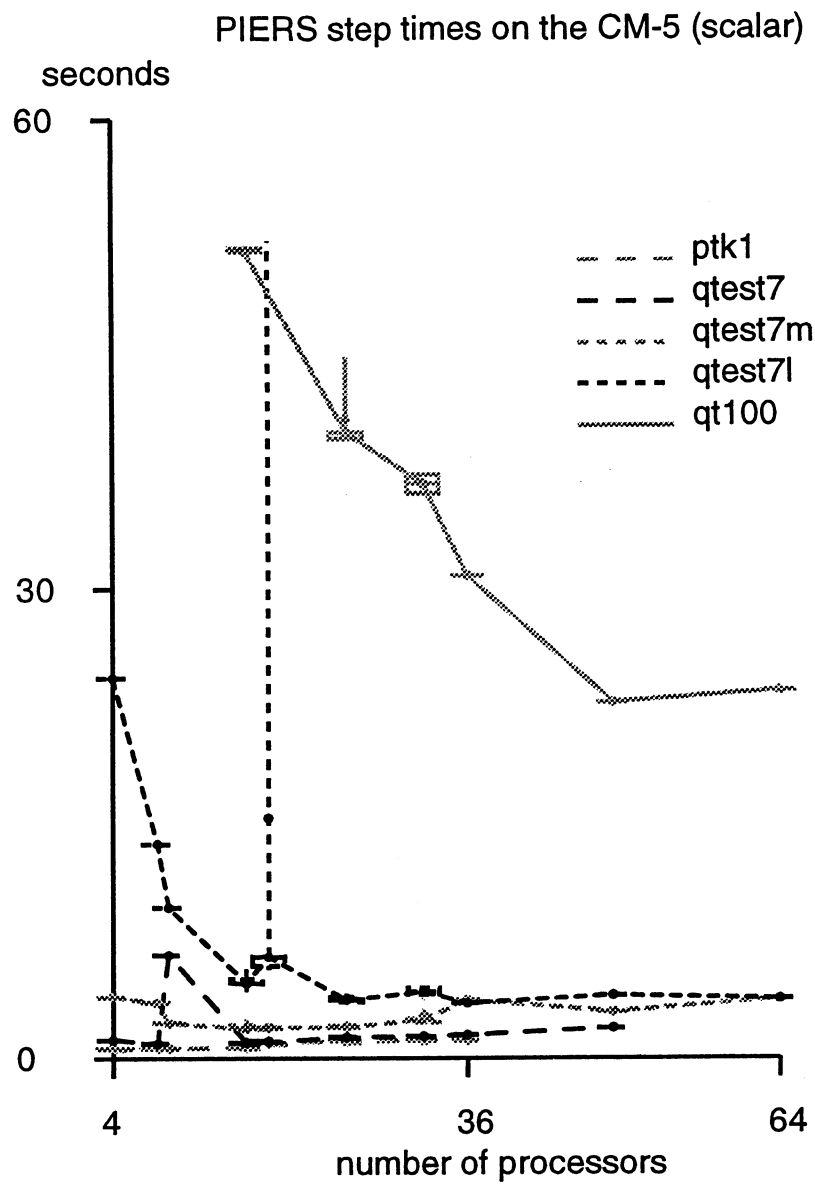

Figure 1: CM-5 Timings

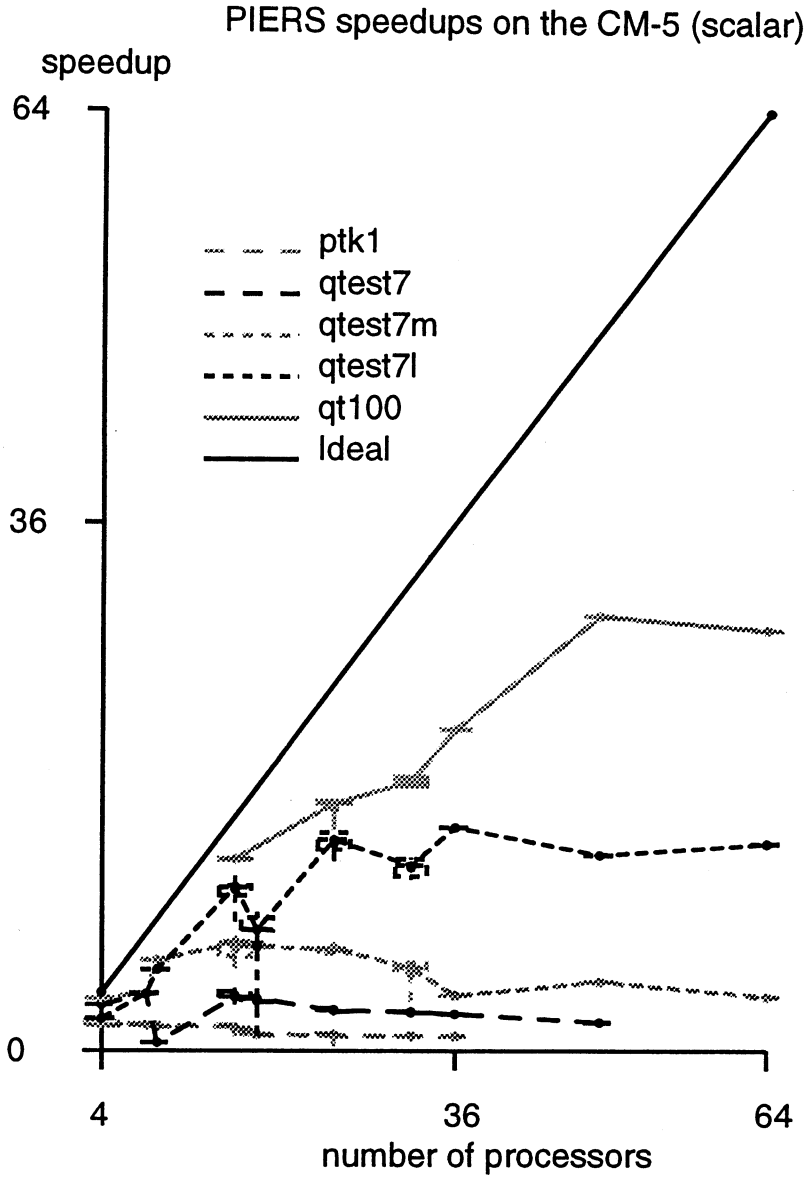

Figure 2: CM-5 Speedups

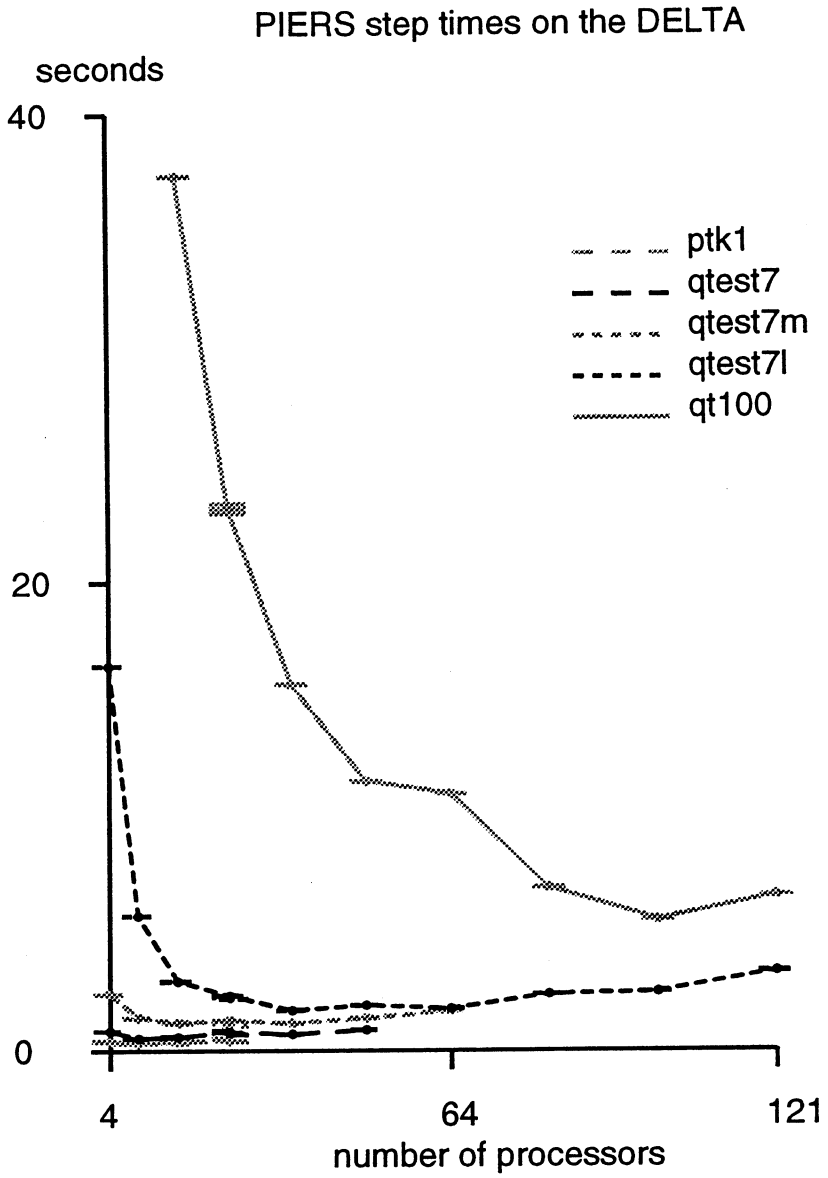

Figure 3: Delta Timings

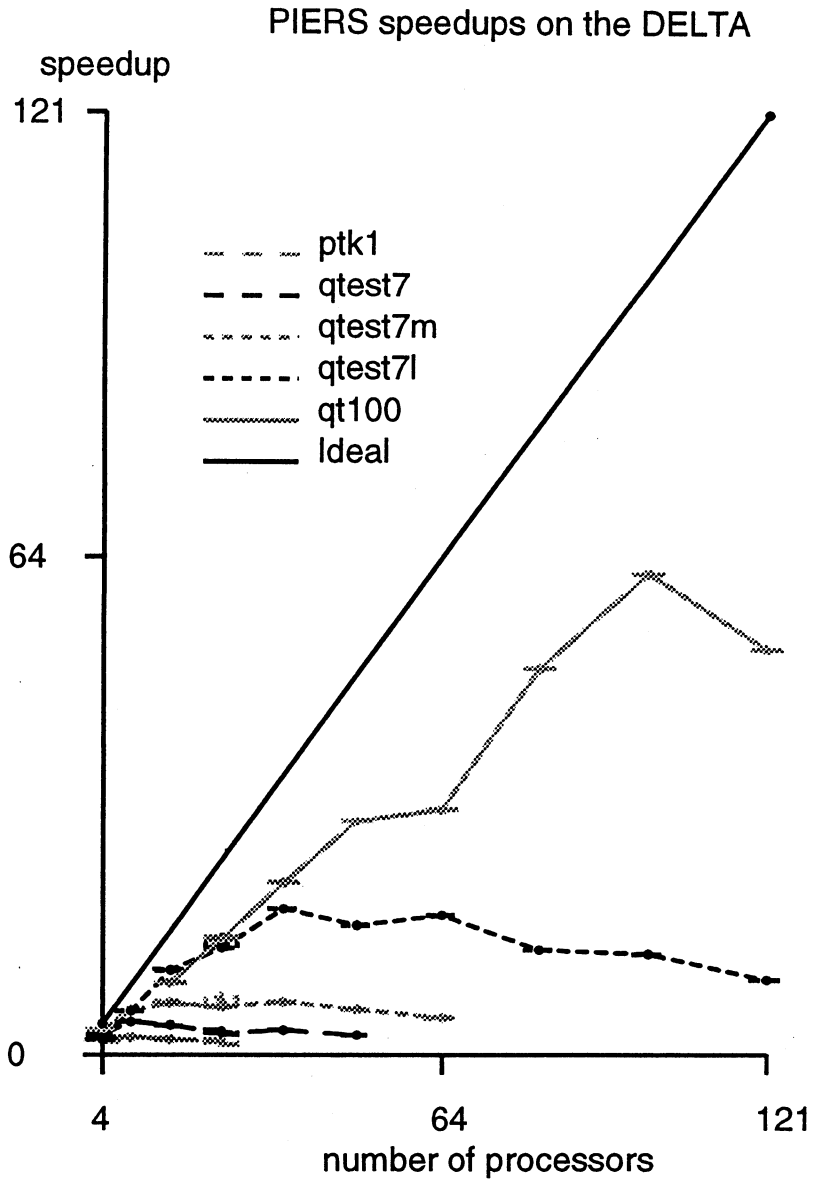

Figure 4: Delta Speedups

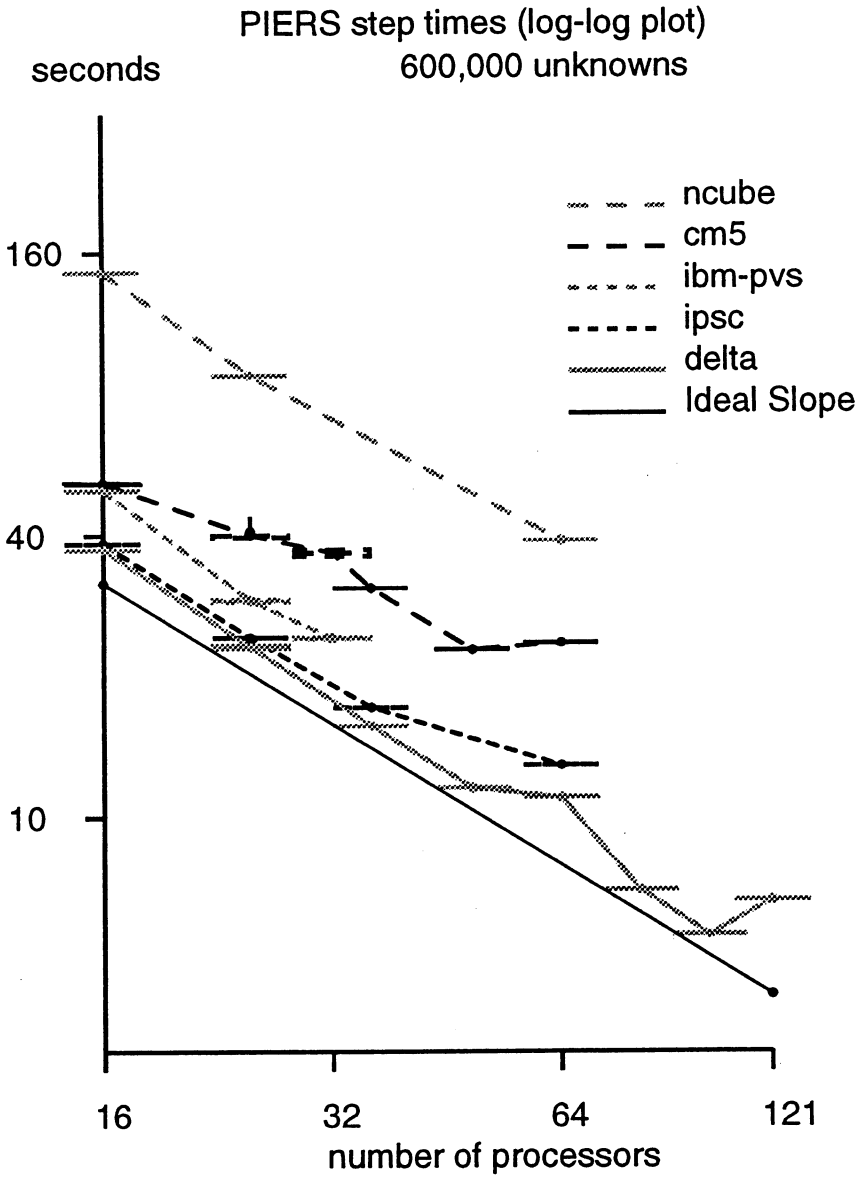

Figure 5: Machine Comparison: Log-Log Plot of Step Times

5 Conclusions

The timing results generally exhibit the surface to volume ratio effect, which is typical of PDE codes on MIMD machines. The specific timings depend as much on problem size and number of processors as on the machine, with cache effects and other oddities complicating the smooth shape of the curves. Thus buying decisions should be based on test cases of the same size as production runs, since timings often do not scale linearly. Moreover the results may be very sensitive to the particular simulator used. Despite these complications, it is clear that parallel computation can produce significant speedups.

References

[1] Flower, Jon and Kolawa, Adam, *A Packet History of Message Passing Systems*, Physics Reports 207, 291, 1991.

[2] Wheeler, John. A. and Smith, Richard A., *Reservoir Simulation on a Hypercube*, SPE Reservoir Engineering, Nov. 1990, pp. 554–548.