Navier-Stokes Computations on Commodity Computers

By

Veer N. Vatsa
NASA Langley Research Center, Hampton, VA
v.n.vatsa@larc.nasa.gov

And

Thomas R. Faulkner
MRJ Technology Solutions, Moffett Field, CA
faulkner@nas.nasa.gov

Abstract

In this paper we discuss and demonstrate the feasibility of solving high-fidelity, nonlinear computational fluid dynamics (CFD) problems of practical interest on commodity machines, namely Pentium Pro PC's. Such calculations have now become possible due to the progress in computational power and memory of the off-the-shelf commodity computers, along with the growth in bandwidth and communication speeds of networks. A widely used CFD code known as TLNS3D, which was developed originally on large shared memory computers was selected for this effort. This code has recently been ported to massively parallel processor (MPP) type machines, where natural partitioning along grid blocks is adopted in which one or more blocks are distributed to each of the available processors. In this paper, a similar approach is adapted to port this code to a cluster of Pentium Pro computers. The message passing among the processors is accomplished through the use of standard message passing interface (MPI) libraries. Scaling studies indicate fairly high level of parallelism on such clusters of commodity machines, thus making solutions to Navier-Stokes equations for practical problems more affordable.

Introduction

Most of the effort in the field of computational fluid dynamics (CFD) in past three decades has been focussed on developing codes for the largest and most powerful computers or the so called super-computers. This trend in computing started with the advent of the CDC 7600 computer in 1969. The next stage in development focussed on vector processors, and first in this series were the ILLIAC-IV and CDC STAR 100 computers of the early seventies. Programming on these computers was extremely tedious, especially for obtaining optimal vector performance for complex implicit algorithms. Advent of Cray computers, starting with Cray 1 in 1976, was a boon to researchers in the CFD applications, because of its large memory, fast speed (12.5 Mflops), and easier programming environment. The trend for using the most powerful shared memory computers (e.g. Cray C-90) for scientific computing, especially large scale computations, is still popular with the CFD research community, due to ease of programming and availability of large memories.

However, current trends in computer hardware are dictating a gradual shift toward the use of massively parallel processing (MPP) machines and workstation clusters for scientific computing. This is evident from the sudden growth in technical meetings and workshops devoted to code parallelization related activities. For example, a quick glance at the proceedings of one such workshop (ref. 1) clearly indicates that significant effort is being devoted to porting existing CFD codes to distributed computing environments. Most of these efforts can be classified into two broad categories: the first one concentrating on modifying (or seeking) algorithms and software more suited for parallelization (refs. 2-4); the second one focussing on porting existing, well established large computer codes to the distributed computing environment (refs. 5-8). We are pursuing the latter approach to benchmark the performance of an existing Navier-Stokes code on the MPP's and clusters of commodity computers.

Computation on commodity computers: A new Paradigm

The development of parallel computer codes and the availability of the new class of powerful commodity (personal) computers with larger memories is creating a paradigm shift in the field of CFD. In the past, only large and well-funded organizations could afford to purchase the Cray class Vector super-computers or the high-end MPP computers, such as the distributed memory IBM SP-2 systems, and the shared memory multiprocessor (SMP) SGI/Cray O2000 systems, which can easily cost several million dollars. However, it is now possible to assemble a reasonable size cluster of powerful PC's complete with a fast and dedicated network under $100,000. Such developments open the door for performing non-linear, high-fidelity CFD computations at smaller research groups in universities and small businesses, as well as in countries where super-computers are unavailable due to variety of reasons. We plan to demonstrate the application of the widely used Navier-Stokes code, TLNS3D (ref. 9), on such a cluster of PC's. In this paper, we would discuss the pertinent issues involved in the parallelization of this code for computing on clusters of machines.

Parallelization Implementation Strategies

An engineering approach outlined by Vatsa and Wedan, and Vatsa and Hammond (refs. 10-11) is used here for parallelizing the TLNS3D code so that significant benefits of parallelization could be realized with a minimum of code changes. We take advantage of the existing block structure of the code to partition the problem for achieving good load balance and parallelism. A coarse-grain parallelization approach is used where multiple blocks can be assigned to a compute node but a given block cannot be split across nodes. Each compute node can have from one block (maximum parallelization) to all blocks (no parallelization) assigned to it. Thus, the maximum number of compute nodes is equal to the total number of grid blocks for a given problem. Each compute node stores the complete data for the blocks assigned to it. If the grid blocks that share a common interface are assigned to different nodes, then the interface data must be transferred by using 'SEND' and 'RECEIVE' calls to the underlying message passing libraries.

A master node serves as the host for sending and receiving global data to and from the compute nodes. After the input data have been read by the master node and a rank for each compute node

has been determined, the global input data and block-specific geometric data are sent to the appropriate compute nodes. The compute nodes perform identical instructions on the assigned blocks. Whenever the logic calls for an exchange of data from other nodes, this exchange is accomplished through the use of 'SEND' and 'RECEIVE' calls by the appropriate nodes via node-to-node communications, thus bypassing the master node completely for such exchanges. We have implemented both synchronous (blocking) and asynchronous (non-blocking) modes of communications among processors supported by the MPI libraries (ref. 12), with asynchronous communication mode being preferred as the number of compute nodes is increased.

Load Balancing

Load balancing is critical to obtain good performance for parallel computing. Although perfect load balance is very difficult to achieve for practical problems with different dimensions for various grid blocks, it is still possible to achieve a reasonable level of load balance for multiblock codes. We make use of a simple, static load balancing algorithm that attempts to equalize the total number of grid points assigned to each compute node without further subdivision of the grid blocks, as described in refs. 10-11.

In real-life CFD problems, a wide variation exists in grid sizes across blocks, which can limit the benefits of distributed computing because of load-balancing considerations for computer codes that employ only coarse-grain parallelism at the block level similar to the strategy employed here. Before running a test case, we analyze the suitability of a problem for distributed computing from the load-balancing viewpoint. If it is determined that a good load balance is not feasible, we split the larger blocks into smaller ones as a pre-processor step. The usefulness of this approach was demonstrated in ref. 11, where a 19-block grid suitable for running only on 3 compute nodes was split into a 26-block grid, which produced a good load balance on 16 compute nodes. We can get even higher level of parallelism for this problem by splitting the (larger) blocks further. The estimated ideal speedups for this case (neglecting communications overhead) for 19, 26 and 71 blocks are compared in Fig. 1. Note that starting from a maximum of 3 speedup (for 19 blocks), one can expect to get a speedup of almost 48 for the equivalent 71 block problem. Such an approach makes it feasible to improve the parallelizability of uneven-sized grid problems with only minor rearrangement of grids, through judicious use of grid splitting. The actual computational results for this problem are presented in a later section of the paper.

Computing Platforms

We have chosen two test-beds in this paper to compare the performance of the TLNS3D code for the emerging class of parallel computers. The first one is a state-of-art shared memory, multiprocessor (SMP) machine, namely a 64 CPU SGI/Cray Origin 2000 computer consisting of 195 MHz R10000 MIPS processors with a built-in MPI 3.0 library for message passing, costing approximately $45,000 per processor. The main reason for selecting this platform was to provide baseline computational results, since an MPI version of the TLNS3D Navier-Stokes code was already operational on this system. These calculations and timings could be used for assessing the performance of this code on other systems. The second computer platform chosen for this work is a 36-node cluster of 128 MB (per processor) Pentium Pro PC's connected to a fast dedicated Ethernet

network. The freely available MPICH 1.1.0 library is used for message passing among nodes. The cost per processor (including network interconnect) is approximately $2,000 for this setup. In order to quantify the communication overheads and its effect on parallelizability, we instrumented the code to generate statistics on message passing activities. Only minor code modifications were required to run it on the PC cluster. Basically, these changes were related to optimization levels, since some of the subroutines in the original code produced erroneous results with higher levels of optimization; these offending subroutines were compiled with lowest optimization levels. This type of glitches are not too uncommon in the early stages of compiler development, and our feeling is that the Fortran compilers for PC's are lagging behind the compilers available for scientific workstations and super-computers. Thus we can expect even better performance from PC's in future from the compiler development side.

Results and Discussion

The distributed (parallel) version of the Navier-Stokes code TLNS3D was used to compute the viscous flow over an isolated wing, known as the ONERA M6 wing as the first test case for this study. A 48-block grid obtained by splitting a single-block 289x65x49 grid was used for this case. The block sizes are identical for this test case, so that load-balance is not an issue, as long as the number of compute nodes is selected from factors of 48. The total wall time to compute this case are shown in Fig. 2 for both systems. On an average, the Pentiums take approximately 5 times longer to complete the calculations compared to O2000 system, and the total computation time is under 2 hours for 12 or more nodes.

We now examine the statistics on message passing activities in order to get a quantitative measure of communication overhead. These results are shown in Fig. 3. Note that for the SGI O2000 system, the communication overhead peaks at about 6%, whereas for the Pentium Pro PC's, it is steadily increasing with the number of nodes, and is about 28% for 24 compute nodes, and therefore is expected to have an adverse affect on the speedups for the Pentiums. It might be worth mentioning that communication overheads can make parallel computing impractical on standard (non-dedicated) general purpose networks. We conducted an in-house exercise, using 6 SGI R10000 workstations, and found dismal levels of performance due to the communication bottleneck on the network. Based on our experience, a fast dedicated node-to-node network is of utmost importance for achieving high levels of parallelism from computer clusters.

The actual speedups in wall time (over single node computations) are shown in Fig. 4 along with the estimated linear speedups. The single node computational time used for determining these speedups was obtained by running the identical problem on a single node with the serial (non-parallel) version of the code to eliminate any communication and parallelization related overheads. The ideal speedup based solely on the grid point distribution on the nodes is expected to be linear. The present results indicate that approximately 72% parallel efficiency has been achieved on 24 Pentium compute nodes, which is somewhat lower than the efficiency on O2000 system, but is considered quite good considering the high communication overhead (see Fig. 3) associated with the Pentium cluster. The more important observation drawn from this study is the demonstration of the feasibility of performing Navier-Stokes computations on commodity computers (Pentium Pro PC's) for aerodynamic problems of interest.

The final test case chosen for this study is a generic high speed civil transport (HSCT) configuration, discussed earlier in the load balancing section, and shown schematically in Fig. 5. The computations were performed on the 71-block grid due to its potential for high degree of parallelism. The convergence history for lift coefficient for this case is shown in Fig. 6. The computations were performed by employing the Full Multigrid with grid sequencing, with 50-50-300 iterations on the coarse, medium and fine grids. Based on this figure, it is clear that the lift coefficient is well converged. The speedup in wall time on the SGI O2000 and Pentium Pro computers with increasing number of compute nodes for these computations are shown in Fig. 7. The estimated ideal speedup based solely on computational loads is also plotted in this figure for comparison purposes. The actual speedup for the O2000 system traces the ideal speedup very closely. Although the speedup for Pentium cluster is somewhat lower for larger number of compute nodes, it is still quite good and results in a monotonic decrease in wall time with increasing number of nodes. The actual wall times to obtain the converged solutions on this system are shown in the next figure (Fig. 8). It is observed from this figure that one can obtain a converged solution to the Navier-Stokes equations for a complex and realistic configurations on a cluster of 32 Pentium Pros in approximately 2.5 hours of wall time, which is quite remarkable

The significance of such computations becomes even more apparent when one compares the relative costs of these systems. Using published data for the cost of purchasing these systems, it is estimated that the average cost/performance ratio, measured as dollars per million floating point operations (MFLOPS) for the SGI O2000 system for this case is $1000, and for the Pentium Pro PC-based system the cost is $273. Even more important is the fact that the total cost of the O2000 system used in this study was $2.9 million, whereas the cost of the PC cluster was $72,000. As mentioned earlier in the paper, the PC cluster offers a more affordable compute platform, albeit somewhat slower.

Concluding Remarks

A widely used Navier-Stokes code has been ported to run efficiently on a cluster of commodity Pentium Pro computers. The code has been applied to solve viscous turbulent flows over a configuration of practical interest. Good scalability has been demonstrated for fixed sized problems for up to 24 nodes. The results presented in the paper demonstrate the feasibility of solving Navier-Stokes equations for practical CFD problems on clusters of commodity computers, thus opening up the doors to smaller research groups for non-linear, high-fidelity computations with limited budgets. The final paper would include results for essentially a full aircraft configuration consisting of 71 unequal sized blocks to demonstrate this methodology for solving viscous, turbulent flows over complex aircraft configurations.

REFERENCES

1. Feiereisen, W.J. (Editor): NASA CD Conference Publication 20011, proceedings of Computational Aerosciences Workshop 96, NASA Ames Research Center, May 1997.
2. DiNucci, D.C.: “Cooperative Data Sharing: An Architecture-Independent Interface for Implementing Parallel CFD Applications.” NASA CP 20011, May 1997, pp. 111-116.
3. Roe, K., and Mehrotra, P.: “Evaluation of High Performance Fortran for CAS Applications.” NASA CP 20011, May 1997, pp. 131-132.
4. Naik, V.K.; Coleman, D.W.; Konura, R.B.; and Moreira, J.E.: “DRMS: An Application Oriented Management Environment for Dynamic Control and Scheduling of Resources on IBM-SP2.” NASA CP 20011, May 1997, pp. 157-162.
5. Krishnan, R.K.: “An Efficient Parallelization Procedure for Multi-Block CFD Codes.” NASA CP 20011, May 1997, pp. 29-34.
6. Van der Wijngaart; and Yarrow, M.: “RANS-MP: Portable Parallel Navier-Stokes Solver.” NASA CP 20011, May 1997, pp. 145-150.
7. Reuther, J.; Rimlinger, M.J.; Alonso, J.J.; and Jameson, A.: “Rapid Cycle Aerodynamic Shape Optimization of Complete Aircraft Configurations via an Adjoint Formulation and Parallel Computing.” NASA CP 20011, May 1997, pp. 167-172.
8. Bhat, M.K.: “Development of an unstructured Grid Fluids Module for ENSAERO in a Parallel Environment.” NASA CP 20011, May 1997, pp. 179-184.
9. Vatsa, V. N.; Sanetrik, M. D.; and Parlette, E. B.: “Development of a Flexible and Efficient Multigrid-Based Multiblock Flow Solver.” AIAA Paper No. 93-0677, January 1993.
10. Vatsa, V.N.; and Wedan, B.W.: “Parallelization of a Multiblock Flow Code: An Engineering Approach.” To appear in Computer and Fluids.
11. Vatsa, V.N.; and Hammond, D.P.: “Viscous Flow Computations for Complex Geometries on Parallel Computers.” Paper presented at the “4th NASA National Symposium on Large-Scale Analysis and Design on High-Performance Computers and Workstations,” Williamsburg, Virginia, Oct. 1997.
12. “MPI: A Message-Passing Interface Standard.” MPI Forum, University of Tennessee, Knoxville, TN, May 1994.

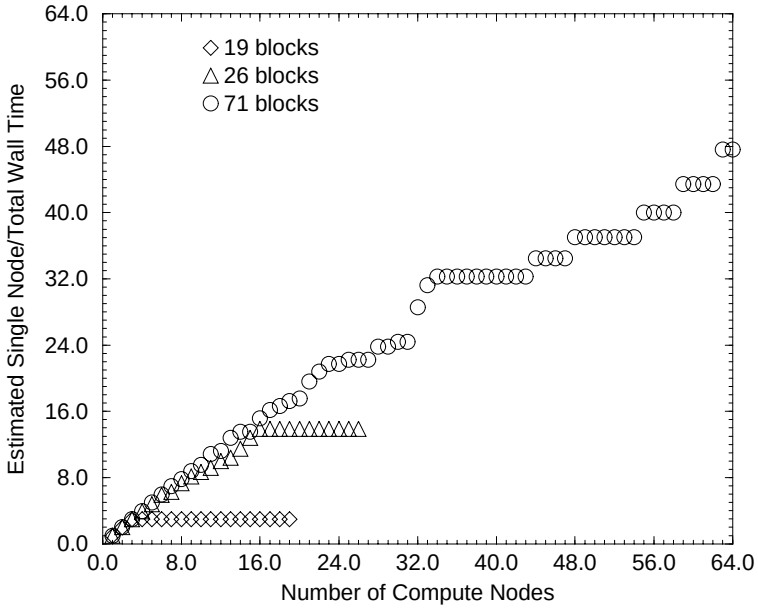

Fig. 1: Estimated ideal speed gains for unequal size block grid configuration

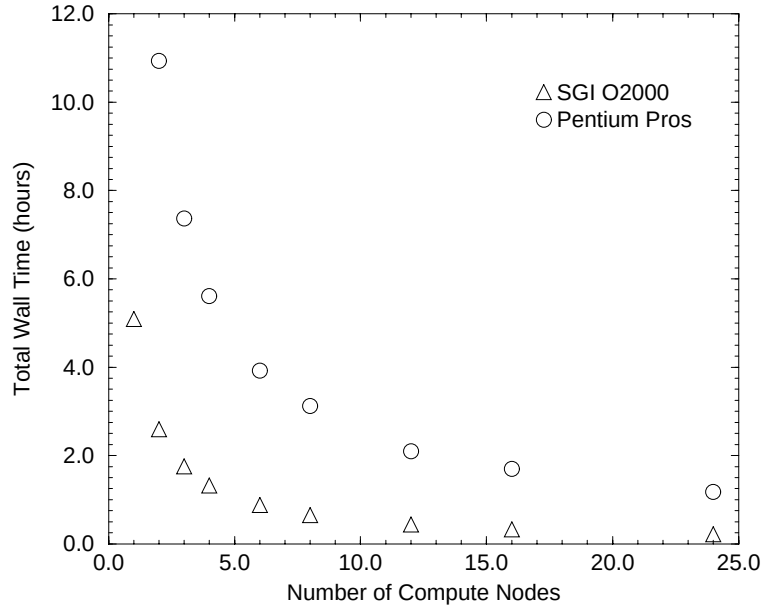

Fig. 2: Comparison of total computational time for ONERA M6 wing

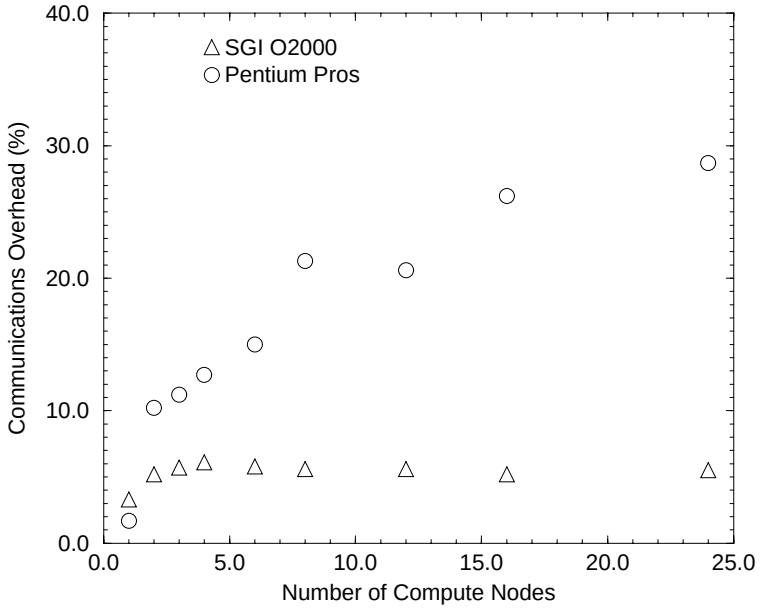

Fig. 3: Communication overhead for the ONERA M6 wing computations

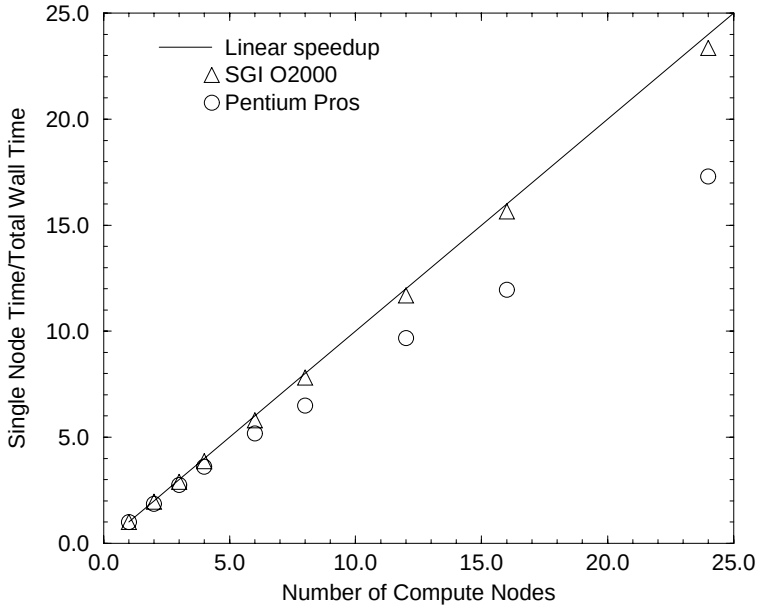

Fig. 4: Speedup comparisons for parallel computing for ONERA M6 wing

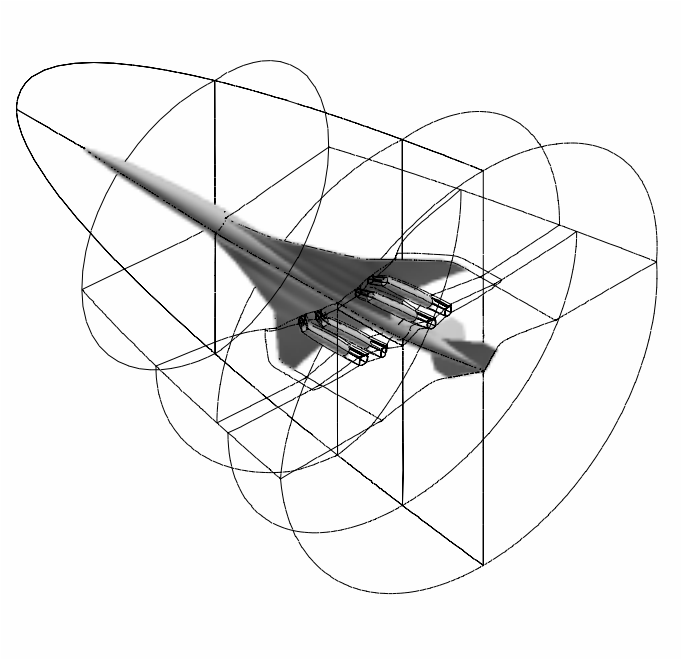

Fig. 5: Schematic of a generic High Speed Civil Transport (HSCT)

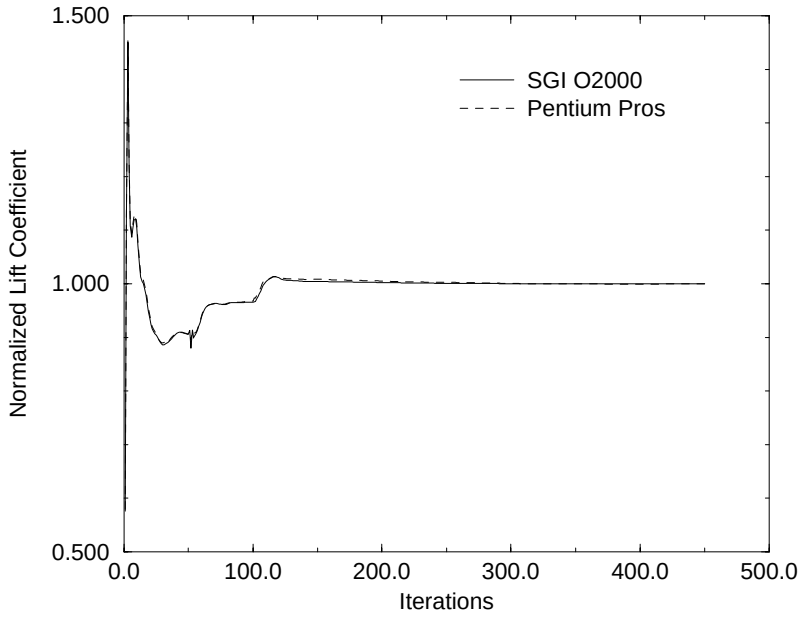

Fig. 6: Lift convergence history for High Speed Civil Transport

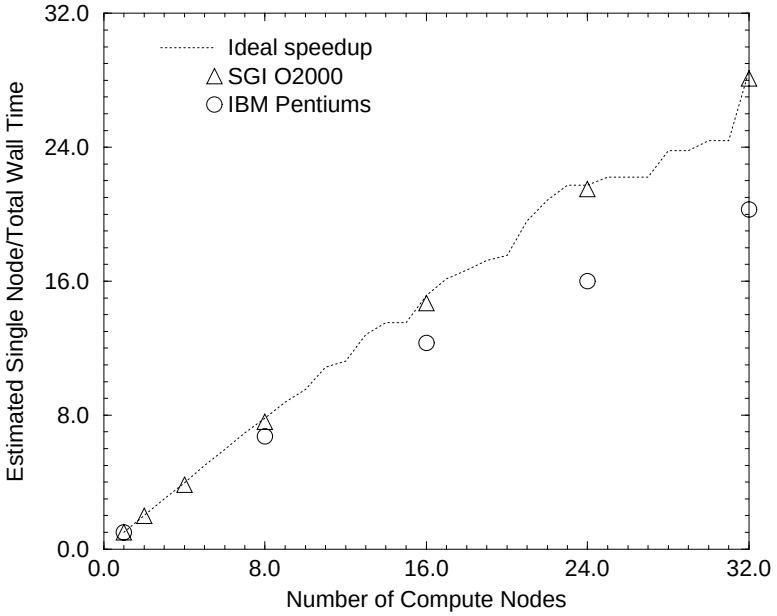

Fig. 7: Wall time speedups for HSCT computations

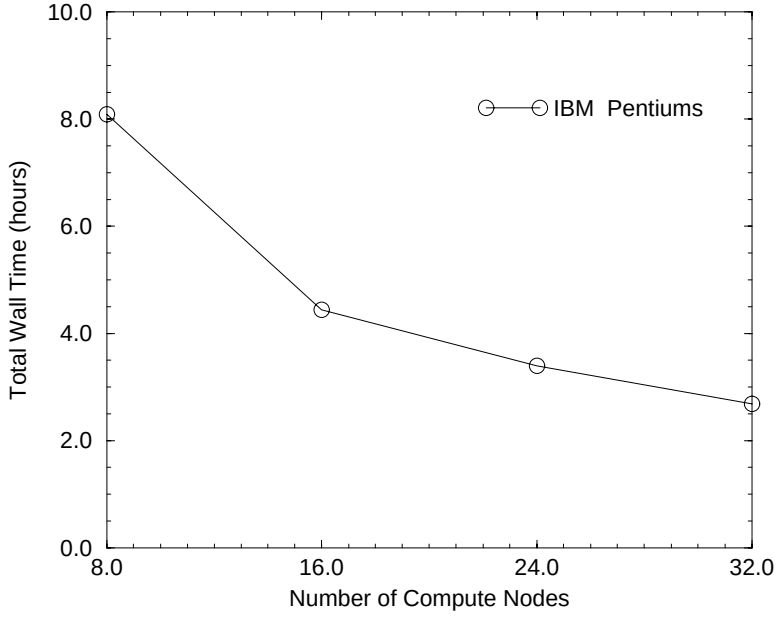

Fig. 8: Physical wall times for HSCT computations