ARMY RESEARCH LABORATORY

ARL Zonal Navier-Stokes Solvers, CHSSI CFD-6 Project Annual Report, 1 April–30 September 1996

by James P. Collins, Daniel M. Pressel, Charles J. Nietubicz, Jubaraj Sahu, Karen Heavey, Paul Weinacht, Harris Edge, Marek Behr, and Jerry Clarke

ARL-MR-364 August 1997

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5067

ARL-MR-364 **August 1997**

ARL Zonal Navier-Stokes Solvers, CHSSI CFD-6 Project Annual Report, 1 April–30 September 1996

James P. Collins, Daniel M. Pressel, Charles J. Nietubicz
Corporate Information and Computing Center, ARL

Jubaraj Sahu, Karen Heavey, Paul Weinacht, Harris Edge
Weapons and Materials Research Directorate, ARL

Marek Behr
Army High Performance Computing Research Center

Jerry Clarke
Raytheon E-Systems

Abstract

The goal of the Common High-Performance Computing (HPC) Software Support Initiative (CHSSI) Computational Fluid Dynamics (CFD)-6 is to develop scalable versions of two Army Navier-Stokes solvers that can efficiently utilize the Department of Defense's (DOD) HPC resources. At the completion of this project, these codes will be made available to the DOD Research, Development, Test, and Engineering (RDT&E) community for analysis and design of weapons systems. This report documents the first 6 mo of this effort.

Table of Contents

INTENTIONALLY LEFT BLANK.

List of Figures

List of Tables

INTENTIONALLY LEFT BLANK.

1. Introduction

The primary goal of the Zonal Navier-Stokes (ZNS) Common High-Performance Computing (HPC) Software Support Initiative (CHSSI) Computational Fluid Dynamics (CFD)-6 project is to develop scalable versions of two Navier-Stokes solvers used by the Army. The two codes chosen for this project are used by designers of projectiles and missiles to understand flow characteristics of their particular designs. By producing scalable versions of these codes, more accurate flow simulations will be achievable in less time, thus reducing the cost of the design process. In addition to the primary goal, other features currently not in the codes will be added. These include a user interface within a complete CFD environment, including grid generation and flow visualization; more general boundary conditions; and an optional high-resolution upwind scheme. Also, a triservice team of software developers, users, and support personnel for these codes will be established. At the completion of this project, training courses will be offered on the use of this software.

Sections 1.1 and 1.2 give a description of the two Navier-Stokes codes and the Distributed Interactive Flow Field System (DIFFS). The remainder of this report documents our progress after the first 6 mo of work.

1.1 ZNS Codes. The implicit version of the F3D code and the U.S. Army Research Laboratory's (ARL) explicit ZNS solver have been chosen for this work. The F3D code was originally developed by Dr. Joseph Steger, while at both NASA at Ames and the University of California at Davis. The F3D code solves the unsteady Reynolds averaged Navier-Stokes equations. It currently uses a flux-split upwind finite difference scheme in the streamwise direction and central differencing in the other directions — an option for upwinding in all three directions will be added. The F3D code uses the Baldwin-Lomax algebraic turbulence model. There are many papers in the literature [1–6] reporting results from this code and favorably comparing the code's results to experiments for various projectile configurations (nonspinning and spinning).

The ARL explicit ZNS solver has been under development at ARL for the last decade and has been used in numerous mission and customer CFD studies [7–12]. It solves the three-dimensional Reynolds averaged Navier-Stokes equations using the McCormick scheme — an option for a more modern high-resolution upwind scheme will be added. The ARL ZNS code was developed specifically for implementation on parallel computers. Past efforts aimed at porting this code to the Denelcor Heterogenous Element Processor (HEP), Cray X-MP, Cray 2, Cray Y-MP, CM-200, CM-5, and Intel Hypercube computers have demonstrated that it is well structured to run on a wide range of parallel platforms, including the current generation of parallel architectures. DZonal is the distributed memory version of this code, modified to run over the Network Distributed Global Memory (NDGM) system (see appendix).

1.2 DIFFS. The DIFFS is a complete CFD environment developed at ARL. It tightly couples grid generation, flow field solution, and visualization and provides a user-friendly graphical user interface. It uses NDGM, a locally developed software library for implementing a virtual shared memory environment. This complete computing environment will provide the means to transition quickly other research codes to the user.

2. Scalable Performance and Validation Results

Our goal is to achieve software scalability, portability, and reusability among the Department of Defense (DOD) scalable high-performance computing platforms by both scalar and parallel optimizations. Over the last 6 mo, we have achieved much of this objective with our implicit ZNS code. Section 2.1 addresses the implicit ZNS code, and section 2.2 addresses the explicit code. A description of the benchmark and validation cases used for each code is given first. For comparison purposes, a hardware overview of some of DOD's scalable high-performance computing platforms is given in Figure 1.

Hardware Overview

CRAY Research C90	**C90**
- 1-8 CPUs	
- 1 GFLOPS / CPU peak	8 GFLOPS
- 4 GB of memory / 8 CPU	4 GBytes
CRAY Research T3D	**T3D**
- 1-512 CPUs	
- 150 MFLOPS / CPU peak	76 GFLOPS
- 64 MB of memory / 1 CPU	32 GBytes
Silicon Graphics Power Challenge Array	**SGI**
- 1-96 CPUs	
- 300 MFLOPS / CPU peak	28 GFLOPS
- 2 GB of memory / 12 CPU	16 GBytes
IBM SP2	**SP2**
- 1-400 CPUs	
- 266 MFLOPS / CPU peak	106 GFLOPS
- 64-1024 MB of memory / 1 CPU	57 GBytes

- *ignores communication costs
- *ignores percentage of peak achievable

Figure 1. HPC Hardware Overview.

2.1 Implicit ZNS Code.

2.1.1 Description of the Benchmark Case. Three-dimensional flow over a generic missile configuration is our primary benchmark case for the implicit ZNS code. Calculations were made for this missile at Mach number 2.5 and 14° angle of attack. The model consists of a 3-caliber ogive and a 10-caliber cylindrical afterbody. The computational domain contains only half of the missile—appropriate symmetry boundary conditions are imposed at the two symmetry planes in the circumferential direction. Numerical results were obtained on a Cray C-90 supercomputer (PK) using the original code, a Silicon Graphics (SGI) power challenge array (PCA), and a Convex Exemplar using the new scalar and parallel optimized version of the code and a Cray T3D using the new message-passing version of the code.

The computational grid consists of three overlapping zones: one upstream of the nose and two on the missile body. The full grid consists of 189 points in the streamwise direction: 75 points in the circumferential direction and 70 points in the normal direction for a total of 992,250 grid points. Each zone is a structured mesh constructed using an algebraic grid generator. The grid outer boundary is two diameters away from the surface of the projectile. Figure 2 shows an expanded view of the grid near the ogive-cylinder junction region, highlighting a longitudinal and a circumferential grid plane. Grid points are clustered near the afterbody surface to capture the viscous effects in the turbulent boundary layer. The grid spacing used for the first point away from the body surface was 0.000005 calibers. The same grid was used for all cases considered here. In each case, the solution was marched from free stream conditions everywhere, until the final converged solution was obtained. The computed solutions are checked for agreement and then compared with the Defense Research Agency (DRA) experimental data.*

Throughout this report, we will refer to the previous case as BM32. A second test case is also used to monitor performance of this code. This case, referred to as BM32A, is identical to BM32 with the exception that the grid was resized to 191 × 225 × 70. This case has over 3 million grid points.

---

* Comparisons to the DRA data were part of the Technical Cooperation Program (TTCP) Weapons Technical Panel-2 (WTP-2) Key Technical Area (KTA) 2-12 collaborative exercise. The data were provided by DRA.

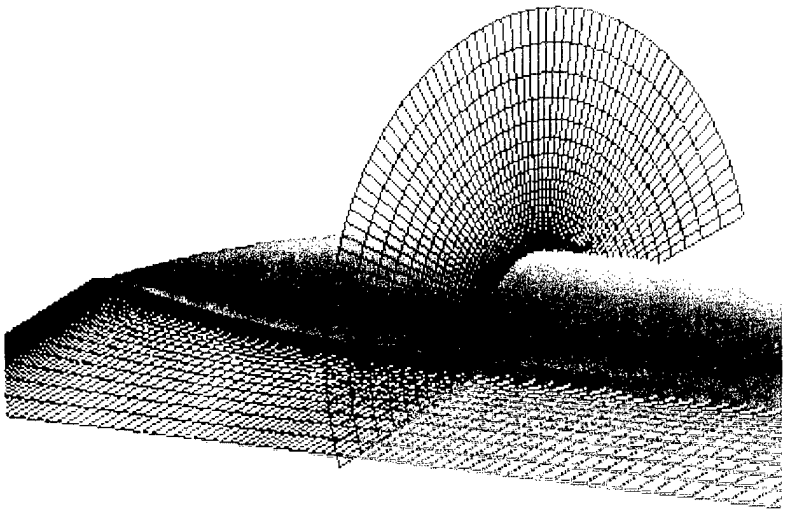


Figure 2. Computational Grid for BM32: Zonal Grid 189 × 75 × 70.

2.1.2 Performance and Validation Results on Shared Memory Architectures. The results presented in this section were obtained using the new reduced instruction set computer (RISC) and parallel optimized version of implicit ZNS solver. Results were generated using ARL's SGI PCA and NRAD's Convex Exemplar. Results were also generated on the C90 at the Corps of Engineers Waterways Experiment Station (CEWES) (PK) for comparison purposes. All C90 results presented were obtained with the C90 version of the code.

Figure 3 compares the pressure computed using the PCA and Cray C90. Both the wind side and lee side pressure contours are shown after 1,800 time steps. The solutions are identical. As expected, a strong shock wave is seen to emanate from the nose of the missile in the wind side. It is much weaker on the lee side. Also, notice the asymmetry in the shock wave angle. Generally, the pressure on the wind side is higher, and the pressure on the lee side is lower. One can also observe the flow expansion at the ogive-cylinder corner. The shock and expansion waves are seen to pass through the outer boundary rather smoothly, which results from the use of the nonreflective boundary condition at the outer boundary. Figure 4 shows longitudinal Mach contours for the PCA solutions at various time steps. They show that the solution has converged at 1,800 time steps.

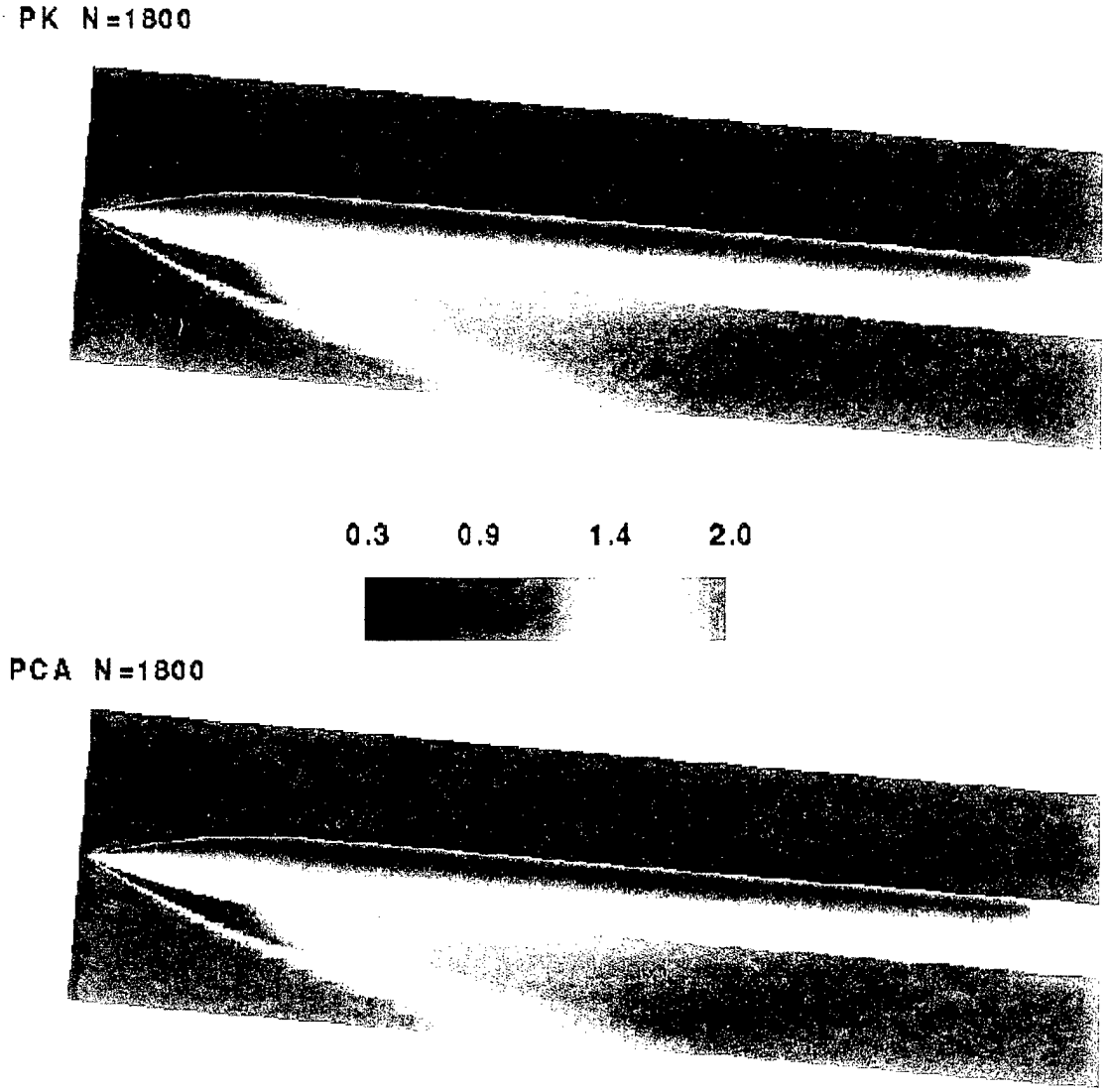


Figure 3. Pressure Contours for BM32 Using the SGI PCA and the Cray C90.

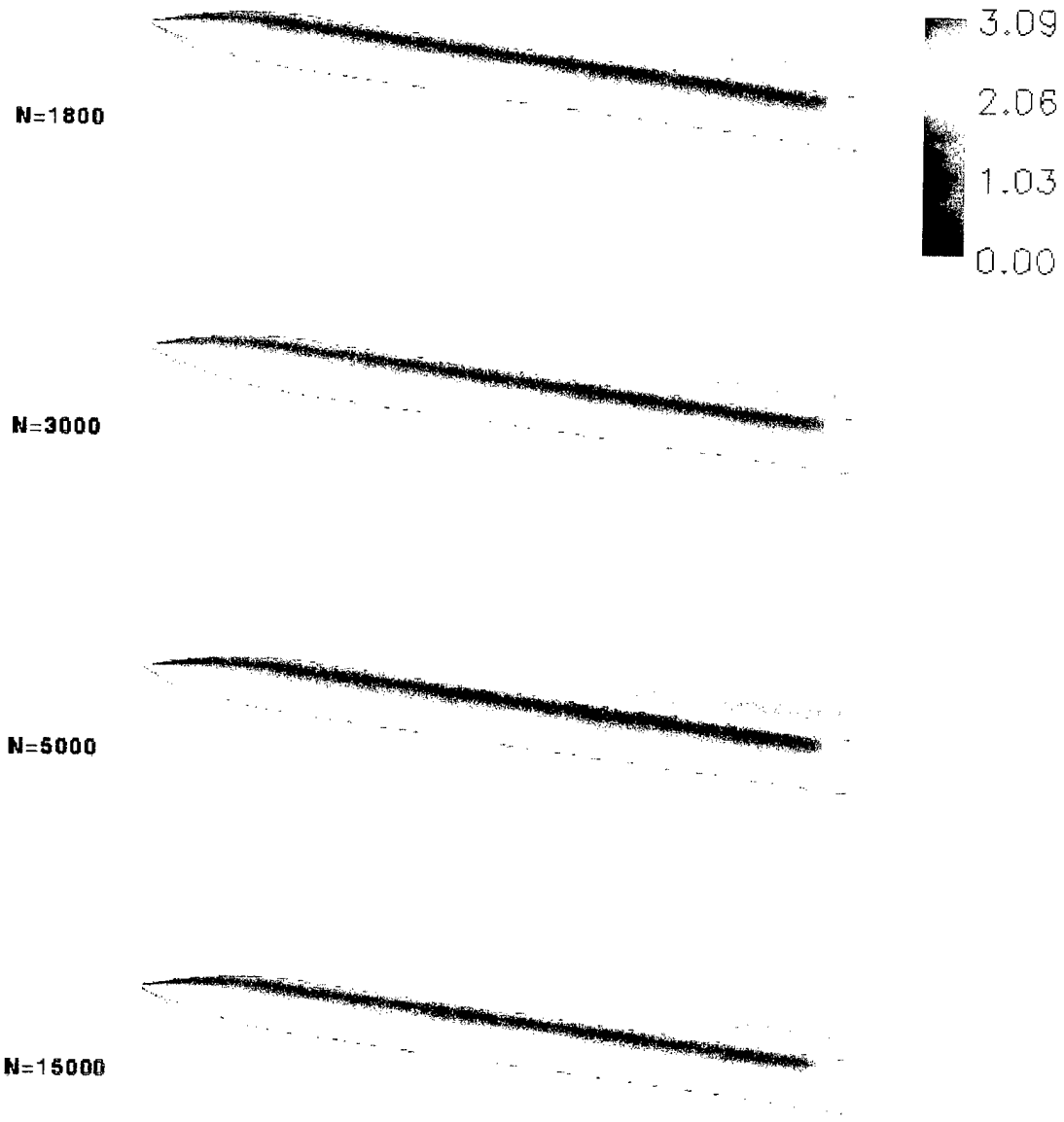


Figure 4. Mach Number Contours for BM32 Using the SGI PCA.

Figure 5 shows the circumferential surface pressure distributions on the missile for various longitudinal stations. One of the longitudinal stations is on the ogive section of the missile (X/D = 2.4) and the others (X/D = 3.5, 8.5, and 11.5) are on the cylindrical section. Phi = 0° corresponds to the wind side and Phi = 180° corresponds to the lee side. Computed results after 1,800 time steps are presented using the Cray C90 (PK) and the PCA, and are compared with the experimental data. Both computed and experimental results show the same trends (i.e., higher surface pressure on the wind side and low pressure on the lee side). Again, agreement between the Cray and PCA solutions is excellent.

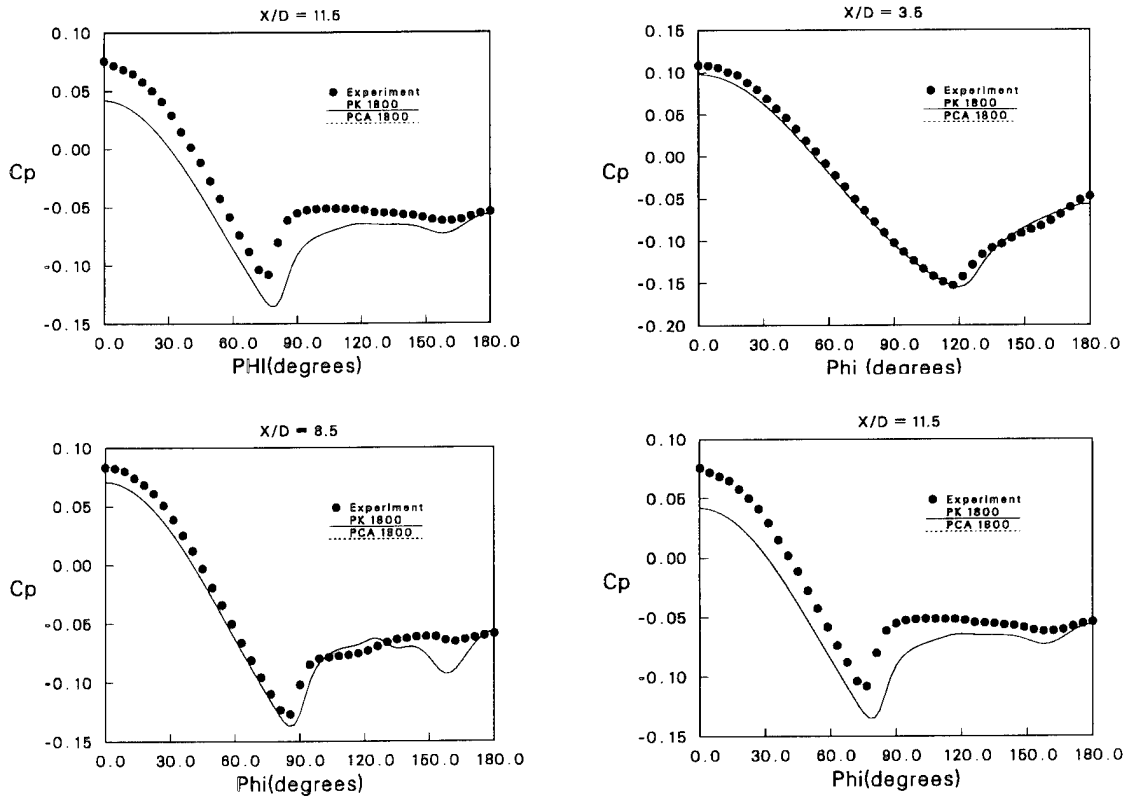


Figure 5. Circumferential Pressure Distribution for BM32—Comparison to C90 Results and Experimental Data.

Parallel- and RISC-optimized performance results for the F3D code running on the SGI PCA and Convex Exemplar for BM32 and BM32A are given in Figures 6 and 7. For the benchmark data set BM32, the 18-processor Power Challenge (75 MHz R8000) runs are roughly 3.0 times faster than the in core version of F3D using one head of a C90 and 3.4 times faster than the out-of-core version (using the Solid State Disk [SSD]). These comparisons have had the initialization and termination costs removed, making this an asymptotic limit for a large number of time steps. The method for removing those costs was to run two cases: one with a large number of time steps and one with a small number of time steps. The run times were then subtracted and the net number of time steps divided into the time to get the time per time step. It was this number that was compared. The comparisons also compared central processing unit (CPU) time on the C90 to elapsed (frequently called wall clock time) on the Power Challenge.

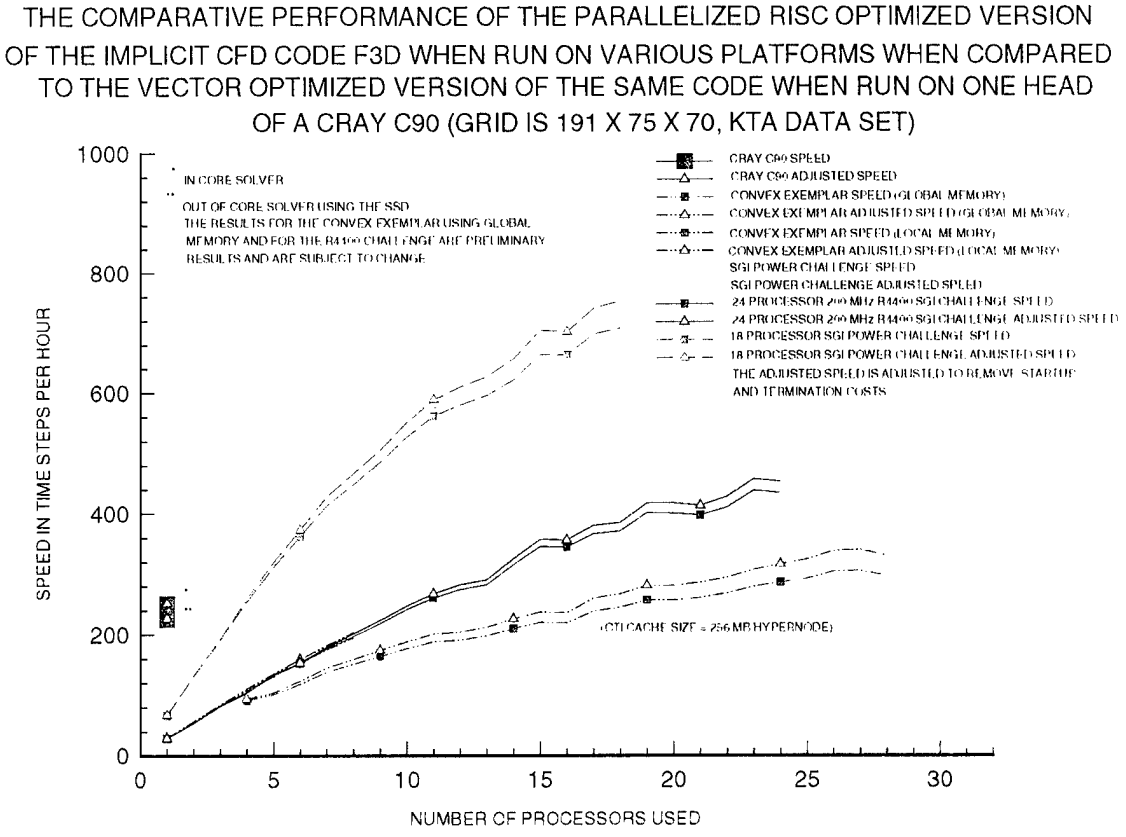


Figure 6. Performance results on BM32: Shared Memory Implementation of Implicit ZNS Solver.

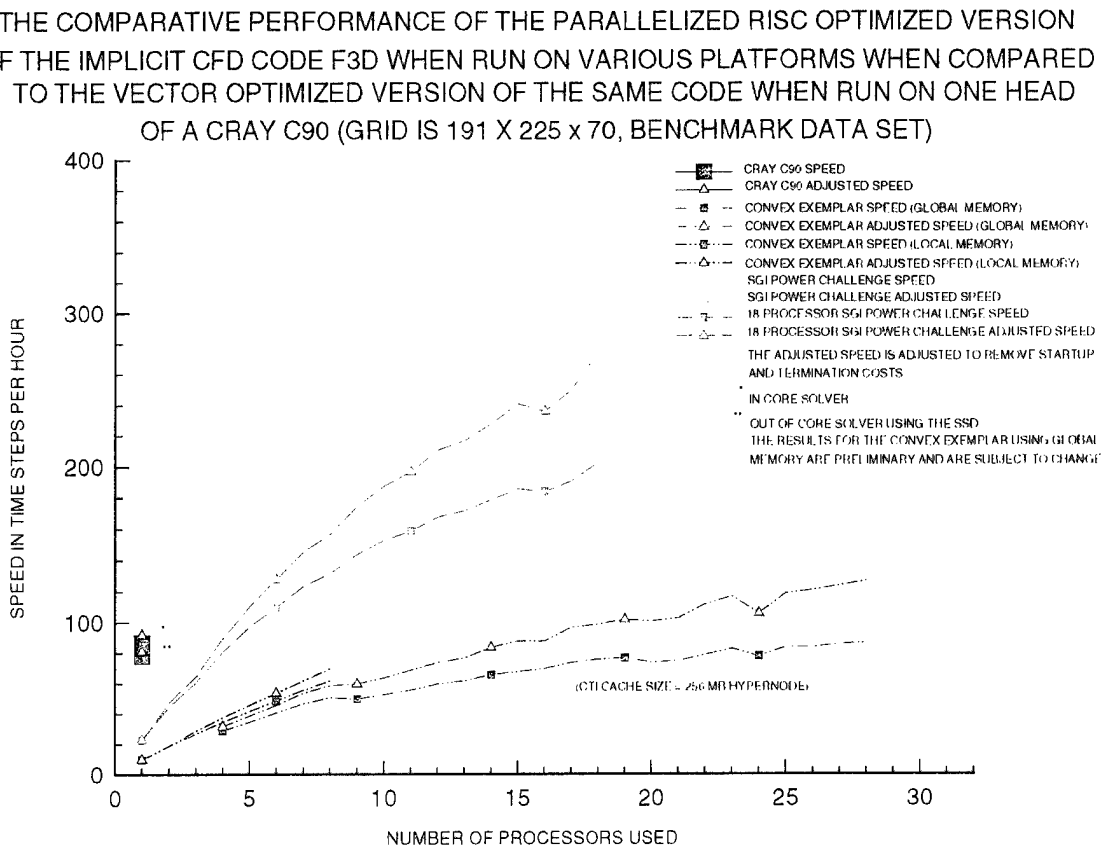


Figure 7. Performance Results on BM32A: Shared Memory Implementation of Implicit ZNS Solver.

It is interesting to note that the relative theoretical peak speed for the hardware was 1.0 GF for the C90 and 5.4 GF for the Power Challenge. Part of the difference for not achieving a comparable speedup is the effect of waiting on cache and translation lookaside buffer (TLB) misses. Another major difference is that it is theoretically impossible to get linear speedup for this code; therefore, the best ratio that could have been achieved is closer to 5 to 1, and there are probably enough other effects that even that number is unlikely to ever be achieved.

The data for the Global memory version of the Convex runs point out that this is a work in progress, but that it is already clear from this and prior work that distributed shared memory architectures are not simple extensions to the shared memory symetric multiprocessor (SMP) programming paradigm. At least that is the case if you are interested in performance. It should also

be noted that other jobs might be better suited for these extensions to the shared memory SMP programming paradigm.

Preliminary experience with the new R10000-based Power Challenge and the 90-Mhz R8000 Power Challenge indicate that these machines are capable of running this code even faster.

2.1.3 Performance and Validation Results on Distributed Memory Architectures. The implicit ZNS code has been rewritten for distributed memory architectures using parallel virtual machine (PVM). We are currently developing this version on the Cray T3D. On the T3D, the PVM calls that reshape the data prior to solving the block tridiagonal system have been replaced with Cray shared memory (SHMEM) calls. This has resulted in a significant performance increase. The latest performance results (given in Figure 8) reflect this change.

Figure 8 shows the timings for some of the most expensive subroutines. These results were obtained using the new PVM version and the Cray C90 version of the implicit ZNS solver. On the T3D, timings were obtained using 128 and 256 processors. At present time, we can achieve a speedup 3.9 times that of a single processor Cray C90 using 128 processors and a speedup of 5.2 with 256 processors. For comparison purposes, the theoretical peak speed for 128 and 256 processors on the T3D are 19.2 and 38.4 times the speed of one head of the C90, respectively. There appears to be two main reasons why the code is not achieving this level of speed-up. The first is the apparent design limitations of the T3D. Secondly, for the particular problem size associated with the benchmark cases, there is some loss of scalability, which is due mainly to the block tridiagonal solver; most of the other major subroutines exhibit scalability (see Figure 8).

Our current approach to validating the T3D results is to compare the residual norm history with the same from a Cray C90 calculation. Figure 9 shows such a comparison. Except for the case where the residual is essentially zero, the norms are identical to six significant digits.

We have experimented with the bulk synchronous parallel (BSP) methodology for parallelizing this code. This work is not yet complete. Preliminary results indicate that the BSP performance is

F3D Performance

BM32 52 Time Steps

- C90/F77 timings

Miscellaneous	=	178.64
Implicit step	=	45.57
Boundary conditions	=	28.89
Tridiagonal solver up	=	219.13
Tridiagonal solver down	=	289.63
Fill tridiagonal z	=	215.54
Fill tridiagonal y	=	266.46
Right hand side	=	244.49
Flux split jacobian z	=	177.51
Flux split jacobian y	=	198.78
Viscous jacobian	=	146.08
Turbulence model	=	0.00
Viscous right hand side	=	58.63
Smoothing	=	167.92
Scale	=	8.05
Boundary data in	=	29.43
Boundary data out	=	32.91
Total	=	2307.64

- T3D/MP timings on 128 PEs

Miscellaneous	=	20.60
Implicit step	=	15.83
Boundary conditions	=	12.09
Tridiagonal solver up	=	103.09
Tridiagonal solver down	=	126.87
Fill tridiagonal z	=	49.77
Fill tridiagonal y	=	53.63
Right hand side	=	36.20
Flux split jacobian z	=	31.94
Flux split jacobian y	=	31.12
Viscous jacobian	=	41.65
Turbulence model	=	0.00
Viscous right hand side	=	15.46
Smoothing	=	35.32
Scale	=	3.13
Boundary data in	=	8.10
Boundary data out	=	8.81
Total	=	593.61

- T3D/MP timings on 256 PEs

Miscellaneous	=	23.56
Implicit step	=	20.64
Boundary conditions	=	13.12
Tridiagonal solver up	=	62.93
Tridiagonal solver down	=	115.36
Fill tridiagonal z	=	33.82
Fill tridiagonal y	=	37.26
Right hand side	=	20.87
Flux split jacobian z	=	18.60
Flux split jacobian y	=	21.47
Viscous jacobian	=	27.15
Turbulence model	=	0.00
Viscous right hand side	=	9.55
Smoothing	=	22.50
Scale	=	1.77
Boundary data in	=	6.06
Boundary data out	=	5.70
Total	=	440.37

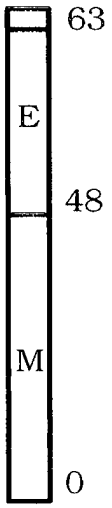
Figure 8. Performance Results on BM32: Distributed Memory Implementation of Implicit ZNS Solver.

F3D Verification

Numerical Result Comparison

- C90/F77 BM32 residual norm history

```
10,   1.539514718111274E-16
10,   2.976273150238765E-7
10,   1.519006922192274E-6
20,   1.529994065244958E-16
20,   2.063416180116936E-6
20,   9.822818993217579E-6
30,   1.637157658973204E-13
30,   6.33558802443001E-6
30,   2.751908970212121E-5
40,   5.440819341052896E-12
40,   1.380431768828975E-5
40,   5.177911091345101E-5
50,   7.216221770382318E-12
50,   2.181499353368315E-5
50,   7.465101253422952E-5
```



- T3D/MP BM32 residual norm history

```
norm: step =   10 l2norm = 0.3243575373081 27E-17
norm: step =   10 l2norm = 0.297627315040179E-06
norm: step =   10 l2norm = 0.151900692229205E-05
norm: step =   20 l2norm = 0.314541820947395E-17
norm: step =   20 l2norm = 0.206341618020403E-05
norm: step =   20 l2norm = 0.982281899367476E-05
norm: step =   30 l2norm = 0.163715694762361E-12
norm: step =   30 l2norm = 0.633558802460311E-05
norm: step =   30 l2norm = 0.275190897030783E-04
norm: step =   40 l2norm = 0.544081933934850E-11
norm: step =   40 l2norm = 0.138043176886223E-04
norm: step =   40 l2norm = 0.517791109144100E-04
norm: step =   50 l2norm = 0.721622177133689E-11
norm: step =   50 l2norm = 0.218149935341319E-04
norm: step =   50 l2norm = 0.746510125353012E-04
```

Figure 9. Validation Results on BM32: Distributed Memory Implementation of Implicit ZNS Solver.

similar to the PVM/SHMEM version of the F3D code; however, when optimized, BSP libraries are available for the Cray T3D, we expect to get better performance with BSP. In addition, BSP should assure the portability of the F3D code to distributed memory architectures other than the T3D.

2.2 Explicit ZNS Code. The DZonal code, a distributed memory version of the Zonal code, is the starting point for the CFD-6 CHSSI project's explicit ZNS code. The DZonal code has been under development at ARL for a few years and has evolved out of the Zonal code, a Cray shared memory code. (See Clark and Edge [13] for more details.) The DZonal code relies on domain decomposition and message passing. In the current version, message passing is implemented using NDGM. The appendix describes NDGM in detail.

At this point, no significant coding changes have been made to this code under the CHSSI project. The work has been focused mainly on characterizing the performance of this code on a benchmark case. The work is described in the following sections.

2.2.1 Description of the Benchmark Case. We are currently using a wrap-around finned missile as a validation and benchmark case for this code. The computational domain consists of a 5-zone grid of approximately 1.4 million grid points with the following dimensions: $(24 \times 96 \times 80)$, $(64 \times 64 \times 32)$, $(96 \times 96 \times 80)$, $(64 \times 64 \times 32)$ and $(64 \times 34 \times 80)$.

This configuration was run on a Cray C-90 with the Zonal code, a Thinking Machines Corp. CM-200 and CM-5 using a beta version of the Zonal code written in CM Fortran, and the SGI PCA. The single CPU SGI PCA calculations were run with the C90 version of the Zonal code. The multiprocessor calculations were obtained using the DZonal code.

The architectural differences of the machines where this application was run provide for an interesting comparison. The Cray C-90 is a large shared memory machine at the Army Corps of Engineers Waterways Experiment Station. The CM-200 is a single-instruction multiple-data set (SIMD) parallel computer located at the Army High-Performance Computing Research Center (AHPCRC). The Zonal code utilized 512 of the available 1,024 nodes, which are arranged in a

hypercube topology. The CM-5, also located at the AHPCRC, is a multiple-instruction multiple-data (MIMD) parallel computer with 32 MB of local memory per cell. The Zonal code used 256 of the available 864 processors. The SGI PCA is a cluster of 8 separate nodes. Each has 12 CPUs, 2-GB main memory, ethernet, ATM, and HIPPI network connections.

2.2.2 Performance Results Using NDGM. The timings for the various calculations are given in Table 1 in seconds per iteration, where an iteration is one time step for all zones. The Zonal code on the C-90 is highly vectorized and its performance on a shared memory machine has been documented previously. The CM-200 and CM-5 versions performed well on interior points of the grid, but spent more than 50% of their time on boundary conditions. This demonstrates a high communication penalty when the problem is decomposed, even on a high-speed communication system.

Table 1. DZonal Performance Results for the Wrap-Around Finned Missile

Computer	CPUs	Seconds/ Iteration	Code
C-90	1	9.52	Vectorized Zonal
CM-5	256	9.52	CM Fortran Zonal
CM-200	512	30.0	CM Fortran Zonal
SGI 1 Node	1	89.5	C90
SGI 1 Node	10	7.69	DZonal
SGI 2 Nodes	20	6.52	DZonal
SGI 3 Nodes	20	4.59	DZonal
SGI 4 Nodes	20	4.94	DZonal

The performance on 1 CPU of the SGI, obtained with the vectorized version of the Zonal code, shows the necessity for serial optimizations on RISC-based machines. Significant speedup would be possible with code reorganization and loop restructuring.

Although the scalar version of the Zonal code does not take advantage of the architecture, it does give a good basis for comparison to the parallel version. When running on multiple nodes of the SGI PCA, the DZonal code used 10 of the 75-MHz R8000 processors (each node contains 12 CPUs; 10 were used for computation: one for the NDGM server and one for operating system [OS] processes). To communicate between SGI nodes, NDGM uses the available high-performance parallel interface (HIPPI). NDGM transfers across the HIPPI of the SGI PCA tend to be in the range of 40–50 MB/s.

While the times for DZonal code running on 10 CPUs is better than 10 times faster than the scalar version of the Zonal code, it is important to remember that these codes are not exactly identical. Better use of the available memory banks and cache may explain this speedup. In any event, all NDGM transfers in this situation translate into system shared memory access that is fast.

As the code is spread to 20 CPUs across two nodes, we begin to suffer from communications across the network. The speed of the HIPPI channel, however, helps to minimize these effects. With an application this small, computation and communication are almost equal. Since only overlapping planes will be written across the network, a larger problem should experience greater speedup.

Additional network communication is necessary as the code is spread across three nodes. This additional communication, however, can be accomplished concurrently. Thus the additional communication is small compared to the reduction in memory bank usage. Each node of the SGI PCA has eight-way interleaved memory. By using three nodes, there are no more than seven DZonal processes on any one node.

Once the saturation of the memory system has been relieved, however, addition of nodes only results in additional communications. This is evident as the code is run on four nodes and 20 CPUs. Additional CPUs would have little benefit since the communication and synchronization, in this situation, is roughly three times the cost of computation.

Clearly, there are many variables contributing to the performance of computationally intensive codes on distributed systems. Therefore, it is not advisable to use these results to compare performance of various computer architectures.

It is important to note that communication between DZonal processes was not "lagged," inner and cross block overlaps were communicated every time step. It is common in parallel distributed flow applications to only communicate every few time steps in order to reduce the impact of communication. This practice was not employed here in order to provide a more direct comparison.

3. Demonstration Calculations

3.1 General. The work done, to date, on the demonstration calculations has been mainly on problem definition. We have two primary test cases to demonstrate the software: steady-state compressible Navier-Stokes solutions of a long-range artillery projectile and a spinning artillery projectile at angle of attack. These represent current Army problems. Because of complex geometry, the long range artillery projectile requires a large number of grid points to accurately resolve the flow field. Accurate predictions of the Magnus force generated on a spinning projectile at angle of attack are critical for trajectory calculations. The boundary layer, complicated by the projectile spin, must be highly resolved to accurately predict the aerodynamic forces. Both of these test cases are challenging applications for scalable computing.

A series of missile and projectile flow field simulations will be performed to demonstrate the scalable ZNS solvers. Results will be compared to experimental data and to computed results using other architectures, such as the C-90 and CM-5.

3.2 Spinning Projectile at Angle of Attack. The implicit F3d Navier-Stokes code will be used to compute three-dimensional flow over a spinning projectile at 4° angle of attack with a spin rate of 4,900 rpm. The model used will consist of a 3.0-caliber secant-ogive nose, a 2.0-caliber cylindrical section, and a 0.5-caliber, 7-degree boattail. The sting-mount at the aft end of the

projectile will also be included in the simulation. Details of the flowfield such as Mach number contours and surface pressure distributions will be compared with experimental data. Aerodynamic force and moment coefficients will be calculated and compared with the data as well. A similar calculation was performed on a Cray-2 computer.

4. Establishing a DOD User Base: Users Group Meeting

On August 14, 1996, the first ARL ZNS users group meeting was held at ARL Aberdeen Proving Ground (APG), MD. The purpose of the meeting was to describe, in detail, the project to the Navy and Air Force participants and to solicit input and suggestions. The meeting was attended by Clint Housh from China Lake, Steve Scherr from Wright Patterson, Drew Wardlaw from Naval Surface Warefare Center (NSWC), Marek Behr from AHPCRC, Steve Davis from the Army Research Office, Bob Post from High-Performance Technologies, Inc. (HPTi); and Charlie Nietubicz, Jubaraj Sahu, Paul Weinacht, Karen Heavey, Harris Edge, Dan Pressel, and Pat Collins from ARL.

Clint Housh, from China Lake, expressed interest in benchmarking a scalable ZNS solver on a lifting-body airframe configuration. This configuration has been analyzed in the past with other codes, and experimental data exists. This is an excellent validation test case for our code. Clint's group does not have the resources to perform code development; so, they are very interested in leveraging the scalable software ARL develops as part of this project.

Stephen Scherr from Wright Laboratory (WL)/FIMC is interested in using a scalable ZNS solver to analyze the Advanced Compact Inlet System. His group is actively researching a portable scalable CFD capability for complex geometries using explicit flow solvers on unstructured grids. The WL/FIMC researchers have experience with porting codes to the SP2, the Paragon, and work station clusters and have agreed to share their expertise with us. The exchange of performance results between ARL and Wright Laboratory (WL) should prove beneficial.

Drew Wardlaw from NSWC Dahlgren Division is interested in steady and unsteady aerodynamics of missiles. His group has developed a number of CFD codes. Several of these codes (ZEUS and SWINT) are widely used in the international missile community. Nearly all of his group's current work is performed on nonscalable DOD assets. As collaborators, the NSWC Dahlgren group will use the scalable ZNS solvers to compute missile flow fields for the Standard Missile.

The first ZNS-users group meeting was a success. We received good suggestions and positive feedback from our non-Army partners. We intend to follow up on some of their suggestions. Our code development efforts will be directed not just towards solving only Army problems, but towards solving Navy and Air Force problems as well.

5. Summary of Progress and Milestones

The first 6 mo of the project have been focused mainly on developing scalable implementations of the implicit ZNS code. This is nearly complete for both the shared memory and distributed memory versions. Once this is finished, there is additional functionality that needs to be added—this will be detailed in the software development plan. Work has been started to add more general boundary conditions to this code, and a technical manual is being written.

The explicit ZNS code's performance has been characterized on the SGI PCA using NDGM. We are now close to the point where we can begin to merge the explicit code with the implicit code in the DIFFS environment.

INTENTIONALLY LEFT BLANK.

6. References

1. Sahu, J. and J. Steger. "Numerical Simulations of Transonic Flows." *Intl. J. for Numerical Methods in Fluids*, vol. 10, no. 8, pp. 855–873, 1990.

2. Sahu, J. "Numerical Computations of Transonic Critical Aerodynamic Behavior." *AIAA J.*, vol. 28, no. 5, pp. 807–816, May 1990.

3. Sahu, J. "Transonic Navier-Stokes Computations for a Spinning Body of Revolution." *Proceedings of the ASME Symposium on Advances and Applications in Computational Fluid Dynamics*, Dallas Texas, November 1990.

4. Nietubicz C. J., and H. Gibeling. "Navier-Stokes Computations for a Reacting, M864 Base Bleed Projectile." AIAA-93-0504, *AIAA J.*, January 1993.

5. Sahu, J. "Numerical Computations of Supersonic Base Flow with Special Emphasis on Turbulence Modelin." *AIAA J.*, vol. 32, no.7, 1993.

6. Nietubicz, C. J. and J. Sahu. "Navier-Stokes Computations of Base Bleed Projectiles." *Paper No. II-2, First International Symposium on Special Topics in Chemical Propulsion: Base Bleed,* Athens, Greece, November 1988.

7. Edge, H. L. "Computation of the Roll Moment Coefficient for a Projectile With Wrap-Around Fins." ARL-TR-23, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, December 1992.

8. Patel, N. R., H. Edge, and J. Clarke. "Three-Dimensional Large Fluid Flow Computations for U.S. Army Applications on KSR-1, CM-200, CM-5, and Cray C-90." ARL-TR-712, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, February 1995.

9. Patel, N. R. "A Study of CFD Algorithms Applied to Complete Aircraft Configurations." AIAA Aerospace Conference, Nevada, January 1993.

10. Patel, N. R. "Implementation and Performance Experience with a 3-D Multi-Block Flow Solver on KSR-1." U.S. Army Research Laboratory Symposium on Massively Parallel Processing, MD, September 1993.

11. Patel, N. R., and H. L. Edge. "Implementation Experience with a Large Fluid Flow Application on KSR-1, CM-5, and Cray C-90." Army Science Conference, Orlando, FL, 20–23 June 1994.

12. Patel, N. R., W. B. Sturek, and G. A. Smith. "Parallel Computation of Supersonic Flows Using a Three-Dimensional, Zonal, Navier-Stokes Code." BRL-TR-3049, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD, November 1989.

13. Clarke, J. and H. L. Edge. "Implementation of a Parallel Fluid Flow Computation in a Distributed Environment by Simulating a Shared Memory System." Preprint 96-022, University of Minnesota, U.S. Army High-Performance Computating Research Center, Minneapolis, MN, April 1996.

Appendix:

Network Distributed Global Memory (NDGM)

INTENTIONALLY LEFT BLANK.

With the growing availability of high-speed network connections, the communications between processors becomes less of a bottleneck. For example, ATM to the local user's desk can sustain multi-megabytes per second and HIPPI connections between a Silicon Graphics (SGI) Power Challenge Array (PCA) realizes over 50 MB/s. The use of existing resources on a local area network could provide the necessary aggregate memory size and computational power to perform large-scale simulations of flow problems.

Programming a distributed memory system using explicit message passing, however, is more difficult than programming a single, large shared memory. This is due to the significant amount of bookkeeping necessary to coordinate the message passing activity between nodes. By simulating a shared memory environment in a distributed memory system, one can take advantage of existing resources to solve flow problems usually reserved for supercomputing facilities. Simulating a shared memory environment also facilitates the porting of existing, shared memory flow codes to distributed environments and allows other applications, such as scientific visualization programs, to easily share data.

Network Distributed Global Memory (NDGM) presents one or more applications with a virtual, unstructured buffer that is accessible to all processes concurrently. Client applications on workstations or multiprocessors can read or write data to a virtual buffer by specifying a destination NDGM address and a pointer in its local address space. To assist in the coordination of parallel tasks, NDGM includes synchronization facilities such as barriers and semaphores.

Applications can connect to the virtual buffer at any time after creation. This allows various applications to nondestructively monitor or actively participate in the overall computational system. Intense computation, visualization, and process monitoring can execute on distributed facilities while accessing the same data. NDGM relieves the application programmer from developing a message passing protocol for sharing information.

The NDGM system consists of an Application Programmers Interface (API), an NDGM Server program, and other NDGM utility applications. Calls to the API result in lower level messages being

sent to the appropriate NDGM Server. Each server keeps track of its piece of the total virtual buffer. The API translates the global memory address into a local address for the server. Requests to get and put to the global buffer result in local memory transfers on the server.

Client programs use the API to access the virtual NDGM buffer as contiguous bytes. In contrast to other systems, no structure is placed upon the NDGM buffer. The application can impose any structure on this buffer that is convenient. In addition, NDGM is designed to implement a system of applications in contrast to a single monolithic parallel application. The API includes facilities to get and put contiguous memory areas, get and put vectors of data, acquire and release semaphores, and to initialize and check into multiple barriers.

A client makes a call to ndgm_init() to connect to the system. If an NDGM server is not running on the local machine, the user must specify the host name location of any NDGM server. Once connected, the client automatically obtains a virtual address to hostname mapping. This allows the API to map requests for data transfer to the appropriate NDGM server. Once initialized, clients can make calls to ndgm_get() or ndgm_put(), to transfer blocks of data and ndgm_vector_get() or ndgm_vector_put() in order to transfer noncontiguous data. For synchronization purposes, the API provides calls to ndgm_barrier_init(), ndgm_barrier_wait(), ndgm_semaphore_get(), and ndgm_semaphore_release(). Client programs can use these calls to coordinate their activity. Checking into a barrier will result in the process blocking until the barrier value reaches zero. Barriers automatically reset to the value set by ndgm_barrier_init() once they reach zero and all waiting clients have been notified. Calls to ndgm_semaphore_get() will block until the client owning the semaphore calls ndgm_semaphore_release().

An additional function, ndgm_dump(), results in requests for servers to write all or part of their data to a disk file. This serves as a parallel I/O facility.

All requests for data transfer and synchronization are handled by the NDGM server process. This is a stand-alone program, started and killed via user commands, that waits for new connections from clients and services their requests.

Each server maintains a memory buffer that maps into the virtual buffer address space. This buffer can be in one of three locations: local address space (obtained via malloc), system shared memory, or a local file. If system shared memory is used, a client executing on the same physical machine as the server accesses the shared memory instead of making requests to a server. This access is transparent to the NDGM client application and results in faster data transfers. Using a file as the server's local storage allows NDGM servers to start with their local memory already initialized and provides a memory image on the local file system.

Since the NDGM server maintains a copy of its portion of the virtual shared memory buffer and a client application is likely to maintain a working copy, the potential for duplication of data is significant. When physical memory size is a constraint, applications must take care to maintain minimal copies of data and scratch arrays.

NDGM servers in a distributed network are started by a Tcl/Tk utility. The utility provides a graphical user interface that allows servers to be started via "rsh" commands. In addition to starting the NDGM servers and initializing the global to local address mapping, the Tcl/Tk utility can monitor NDGM access in a running application. Color bars show NDGM gets, puts, vector gets, and vector puts while client applications are running. This provides a simple, visual, first debugging step for client applications.

Node numbers (and TCP/IP port numbers) in the NDGM system are based on the users's UID. This number can be overridden by use of an environment variable. By assigning node numbers at run time, different users can run NDGM on the same machine and not collide.

The design goal of NDGM was to efficiently implement "put" and "get" functionality over multiple message passing facilities. Therefore, while the API is insulated from the low-level message passing, care has been taken to minimize the overhead of this level of abstraction. This layered architecture allows additional message passing facilities to be supported in a heterogeneous environment while maintaining a consistent API.

The API is implemented on top of the message passing interface. Similar in concept to well-known message passing interfaces like PVM or MPI, this layer provides a level of abstraction freeing the upper layers from the details of reading and writing data. The NDGM message-passing layer has fewer facilities than either PVM or MPI, but is designed to pass NDGM data efficiently with minimal copying. This layer provides calls to establish connections, send messages, probe for incoming messages, read messages, and close connections.

Message headers are converted into external data representation (XDR) format by the API before they are sent. This insures that machines with different internal binary formats can communicate. The data within the message, however, are left unchanged. This allows for maximum speed and reduces the number of times the data is copied or processed on networks with identical binary formats. It is left to the application to do any necessary conversion of the data. To aid in this task, a library of routines is included that utilizes the XDR library found on most UNIX systems.

The actual interprocess data transfer is accomplished by the "drivers." Current drivers include: TCP/IP sockets, PVM, and Fifos. An Intel NX driver is currently available for use on the Intel Paragon. Each driver has functions to open as a client, open as a server, read, write, and probe for incoming messages. When possible, each driver also implements a "select" function in order to monitor several open connections. A single NDGM system can mix nodes utilizing different drivers.

NDGM is written in standard C and is very portability across Unix machines. It currently runs on SGI PCAs, Cray C90 & J90 machines, and Sun work stations. Future plans are to implement NDGM on Cray T3E & IBM SP machines and the ASCI machine. One potential disadvantage of NDGM is its memory requirements. Since the NDGM server maintains a copy of its portion of the virtual shared memory buffer and a client application is likely to maintain a working copy, the potential for duplication of data is significant. When physical memory size is a constraint, applications must take care to maintain minimal copies of data and scratch arrays.

NO. OF COPIES ORGANIZATION

2 DEFENSE TECHNICAL INFORMATION CENTER
DTIC DDA
8725 JOHN J KINGMAN RD
STE 0944
FT BELVOIR VA 22060-6218

1 HQDA
DAMO FDQ
DENNIS SCHMIDT
400 ARMY PENTAGON
WASHINGTON DC 20310-0460

1 CECOM
SP & TRRSTRL COMMCTN DIV
AMSEL RD ST MC M
H SOICHER
FT MONMOUTH NJ 07703-5203

1 PRIN DPTY FOR TCHNLGY HQ
US ARMY MATCOM
AMCDCG T
M FISETTE
5001 EISENHOWER AVE
ALEXANDRIA VA 22333-0001

1 PRIN DPTY FOR ACQUSTN HQS
US ARMY MATCOM
AMCDCG A
D ADAMS
5001 EISENHOWER AVE
ALEXANDRIA VA 22333-0001

1 DPTY CG FOR RDE HQS
US ARMY MATCOM
AMCRD
BG BEAUCHAMP
5001 EISENHOWER AVE
ALEXANDRIA VA 22333-0001

1 DPTY ASSIST SCY FOR R&T
SARD TT T KILLION
THE PENTAGON
WASHINGTON DC 20310-0103

1 OSD
OUSD(A&T)/ODDDR&E(R)
J LUPO
THE PENTAGON
WASHINGTON DC 20301-7100

NO. OF COPIES ORGANIZATION

1 INST FOR ADVNCD TCHNLGY
THE UNIV OF TEXAS AT AUSTIN
PO BOX 202797
AUSTIN TX 78720-2797

1 DUSD SPACE
1E765 J G MCNEFF
3900 DEFENSE PENTAGON
WASHINGTON DC 20301-3900

1 USAASA
MOAS AI W PARRON
9325 GUNSTON RD STE N319
FT BELVOIR VA 22060-5582

1 CECOM
PM GPS COL S YOUNG
FT MONMOUTH NJ 07703

1 GPS JOINT PROG OFC DIR
COL J CLAY
2435 VELA WAY STE 1613
LOS ANGELES AFB CA 90245-5500

1 ELECTRONIC SYS DIV DIR
CECOM RDEC
J NIEMELA
FT MONMOUTH NJ 07703

3 DARPA
L STOTTS
J PENNELLA
B KASPAR
3701 N FAIRFAX DR
ARLINGTON VA 22203-1714

1 SPCL ASST TO WING CMNDR
50SW/CCX
CAPT P H BERNSTEIN
300 O'MALLEY AVE STE 20
FALCON AFB CO 80912-3020

1 USAF SMC/CED
DMA/JPO
M ISON
2435 VELA WAY STE 1613
LOS ANGELES AFB CA 90245-5500

NO. OF
COPIES ORGANIZATION

1 US MILITARY ACADEMY
MATH SCI CTR OF EXCELLENCE
DEPT OF MATHEMATICAL SCI
MDN A MAJ DON ENGEN
THAYER HALL
WEST POINT NY 10996-1786

1 DIRECTOR
US ARMY RESEARCH LAB
AMSRL CS AL TP
2800 POWDER MILL RD
ADELPHI MD 20783-1145

1 DIRECTOR
US ARMY RESEARCH LAB
AMSRL CS AL TA
2800 POWDER MILL RD
ADELPHI MD 20783-1145

3 DIRECTOR
US ARMY RESEARCH LAB
AMSRL CI LL
2800 POWDER MILL RD
ADELPHI MD 20783-1145

ABERDEEN PROVING GROUND

2 DIR USARL
AMSRL CI LP (305)

NO. OF COPIES	ORGANIZATION
1	ARMY HIGH PERFORMANCE COMPUTING RSRCH CNTR M BEHR SUITE 101 1100 WASHINGTON AVE SOUTH MINNEAPOLIS MN 55415
1	ARMY AEROFLIGHT DYNAMICS DIRECTORATE R. MEAKIN MS 258-1 MOFFETT FIELD CA 94035-1000
1	ARL AMSRL PS E B PERLMAN FORT MONMOUTH NJ 07703
3	DIRECTOR AHPCRC T TEZDUYAR B BRYAN G CANDLER 1200 WASHINGTON AVE SOUTH MINNEAPOLIS MN 55415
1	NAVAL SURFACE WARFARE CENTER CODE B44 DR A B WARDLAW SILVER SPRING MD 20903-5640
1	NAVAL RESEARCH LAB J BORIS CODE 6400 4555 OVERLOOK AVE SW WASHINGTON DC 20375-5344
1	NAVAL RESEARCH LAB R RAMAMURTI CODE 6410 WASHINGTON DC 20375-5344

NO. OF COPIES	ORGANIZATION
1	NAVAL RESEARCH LAB OCEAN DYNAM & PRED BR DR J MCCAFFREY JR CODE 7320 STENNIS SPACE CENTER MS 39524
1	US AIR FORCE WRIGHT LAB WL/FIM DR J SHANG 2645 FIFTH ST STE 6 WRIGHT-PATTERSON AFB OH 45433-7912
1	WL/FIMC S SCHERR 2645 FIFTH ST STE 7 WRIGHT-PATTERSON AFB OH 45433-7913
1	WL/FIMC BLDG 450 B STRANG 2645 FIFTH ST STE 7 WRIGHT-PATTERSON AFB OH 45433-7913
1	US AIR FORCE PHILIPS LAB OLAC PL/RKFE CPT S G WIERSCHKE 10 EAST SATURN BLVD EDWARDS AFB CA 93524-7680
1	US AIR FORCE ROME LAB RL/OCTS R W LINDERMAN GRIFFISS AFB NY 13441-5700
1	COMMANDER CODE C2892 C HOUSH 1 ADMINISTRATION CIRCLE CHINA LAKE CA 93555

NO. OF COPIES	ORGANIZATION
1	USAE WATERWAYS EXP STATION CEWES HV C DR J P HOLLAND 3909 VICKSBURG MS 39180-6199
1	NCCOSC RDTE DIV CODE 404 R A WASILAUSKY 53570 SILVERGATE AVE SAN DIEGO CA 92152-5180
1	NCCOSC RDTE DIV CODE 7601T DR K BROMLEY 5180 SILVERGATE AVE SAN DIEGO CA 92152-5180
1	DPT OF ASTRONOMY PROF P WOODWARD 356 PHYSICS BLDG 116 CHURCH ST SE MINNEAPOLIS MN 55455
1	CHSSI DR R FOSTER 1110 N GLEBE RD STE 650 ARLINGTON VA 22201

NO. OF COPIES	ORGANIZATION
	ABERDEEN PROVING GROUND
16	DIR USARL, AMSRL-CI-HA, C NIETUBICZ W STUREK AMSRL-CI-HC, D PRESSEL J COLLINS D HISLEY C ZOLTANI J GROSH A PRESSLEY T KENDALL P DYKSTRA AMSRL-WM-PB, H EDGE J SAHU K HEAVEY P WEINACHT AMSRL-WM-TC, K KIMSEY AMSRL-SC-S, A MARK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1997	Progress, 1 Apr-30 Sep 96

4. TITLE AND SUBTITLE
ARL Zonal Navier-Stokes Solvers CHSSI CFD-6 Project Annual Report, 1 April - 30 September 1996

5. FUNDING NUMBERS
6U19C0

6. AUTHOR(S)
James Collins, Daniel Pressel, Charles Nietubicz, Jubaraj Sahu, Karen Heavey, Paul Weinacht, Harris Edge, Marek Behr,* and Jerry Clarke**

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: AMSRL-CI-H
Aberdeen Proving Ground, MD 21005-5067

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-MR-364

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)
HPC Modernization Office, ATTN: Dr. Roger Foster
110 N. Glebe Rd. , Suite 650
Arlington, VA 22201

10.SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
*Army High Performance Computing Research Center
**Raytheon E-Systems

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The goal of the Common High-Performance Computing (HPC) Software Support Initiative (CHSSI) Computational Fluid Dynamics (CFD)-6 is to develop scalable versions of two Army Navier-Stokes solvers that can efficiently utilize the Department of Defense's (DOD) HPC resources. At the completion of this project, these codes will be made available to the DOD Research, Development, Test, and Engineering (RDT&E) community for analysis and design of weapons systems. This report documents the first 6 mo of this effort.

14. SUBJECT TERMS
Navier-Stokes, scalable algorithms, CHSSI, CFD-6, CFD, NDGM

15. NUMBER OF PAGES
35

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500
Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

INTENTIONALLY LEFT BLANK.

USER EVALUATION SHEET/CHANGE OF ADDRESS

This Laboratory undertakes a continuing effort to improve the quality of the reports it publishes. Your comments/answers to the items/questions below will aid us in our efforts.

1. ARL Report Number/Author ARL-MR-364 (Nietubicz) Date of Report August 1997

2. Date Report Received _____

3. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which the report will be used.) _____

4. Specifically, how is the report being used? (Information source, design data, procedure, source of ideas, etc.) _____

5. Has the information in this report led to any quantitative savings as far as man-hours or dollars saved, operating costs avoided, or efficiencies achieved, etc? If so, please elaborate. _____

6. General Comments. What do you think should be changed to improve future reports? (Indicate changes to organization, technical content, format, etc.) _____

CURRENT ADDRESS

_____
Organization

_____
Name E-mail Name

_____
Street or P.O. Box No.

_____
City, State, Zip Code

7. If indicating a Change of Address or Address Correction, please provide the Current or Correct address above and the Old or Incorrect address below.

OLD ADDRESS

_____
Organization

_____
Name

_____
Street or P.O. Box No.

_____
City, State, Zip Code

(Remove this sheet, fold as indicated, tape closed, and mail.)

(DO NOT STAPLE)

DEPARTMENT OF THE ARMY

OFFICIAL BUSINESS

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO 0001,APG,MD

POSTAGE WILL BE PAID BY ADDRESSEE

DIRECTOR
US ARMY RESEARCH LABORATORY
ATTN AMSRL CI H
ABERDEEN PROVING GROUND MD 21005-5067