Visualization Co-processing of a CFD Simulation

Arsi Vaziri
Numerical Aerospace Systems, NASA Ames Research Center, Moffett Field, CA 94035
(vaziri@nas.nasa.gov)

Abstract

OVERFLOW, a widely used CFD simulation code, is combined with a visualization system, pV3, to experiment with an environment for simulation/visualization co-processing on a SGI Origin 2000 computer(O2K) system. The shared memory version of the solver is used with the O2K 'pfa' preprocessor invoked to automatically discover parallelism in the source code. No other explicit parallelism is enabled.

In order to study the scaling and performance of the visualization co-processing system, sample runs are made with different processor groups in the range of 1 to 254 processors. The data exchange between the visualization system and the simulation system is rapid enough for user interactivity when the problem size is small.This shared memory version of OVERFLOW, with minimal parallelization, does not scale well to an increasing number of available processors. The visualization task takes about 18 to 30% of the total processing time and does not appear to be a major contributor to the poor scaling. Improper load balancing and inter-processor communication overhead are contributors to this poor performance. Work is in progress which is aimed at obtaining improved parallel performance of the solver and removing the limitations of serial data transfer to pV3 by examining various parallelization/communication strategies, including the use of the explicit message passing.

1. Introduction

Computational Fluid Dynamics (CFD) simulations that are used in the investigation of flows encountered in aeronautics design studies are increasingly being run on massively parallel processors (MPPs). One such computer system is the 256 processor SGI Origin2000 at NASA Ames Research Center. The O2K system is a distributed shared memory (DSM) multiprocessor with a single image, globally addressable memory.

During each iteration of a time-dependent CFD simulation, several hundred megabytes of data can be generated as part of the solution. The total solution volume can approach several gigabytes in size.

Immediately after each time iteration, the solution data reside on the local storage of each compute node. The visualization task requires access to this distributed data. Several approaches are commonly used:

1. Post processing: data at each node is collected, combined, and then transferred to a file system for post-simulation processing, usually on a large graphics workstation.
2. Serial visualization: data are combined and/or copied to one node of the MPP where serial visualization software is used for interactive or batch processing.
3. Parallel visualization: a parallel visualization algorithm is used to access the distributed data *in situ* on each processor. Visualization may use the existing solver data distribution or a different distribution. Since most visualization algorithms are "embarrassingly parallel", solver data distribution choices are usually adequate.

There are advantages and disadvantages with each method. Method 1 avoids complications with interactive data transfer on a network and timing conflicts between the visualization and simulation tasks, but it misses the opportunity for interaction with the simulation and the possibility of a simulation steering. Large, powerful SGI Onyx-class graphics workstations may be required for efficient processing. Method 2 is only adequate for smaller size problems due to the limitations in data throughput; but it is easier to implement. Method 3 can pose serious limitations on the real-time interactivity due to network congestion. This model facilitates interactivity with the solver while the solution is being generated. It can thus utilize the MPP for the visualization task. The latter may be important due to the emerging popularity of the Windows/Windows NT workstation as a scientific visualization platform: these workstations may need to have an external visualization compute engine.

Previous work on co-visualization includes a diverse array of software and hardware systems. Woodward [3] utilized an array of workstation and an MPP for CFD steering and design studies. PORTAL [2] is a data communication library developed at Argonne National Laboratory for parallel visualization work. CM/AVS [4][5], and pV3[3][9] have also been used for parallel visualization on an MPP. SCIRun [6], a data flow system, has been used for computational steering. In this study, we use pV3 mainly due to its ability to interact with a variety of MPP architectures. pV3 is also non-intrusive to the solver code and the communication between the main body of the solver and pV3 can be accommodated through two subroutine calls. However, with system-level automatic parallelization, the data exchange between the two systems takes place serially from one processor, the master thread. The O2K appears as a single node to pV3. This is not a limitation of pV3, but only imposed by the current choice in parallelization of the solver.

2. The Simulation/Visualization Software

We used OVERFLOW [7], a large Computational Fluid Dynamics (CFD) research code developed at NASA Ames Research Center. OVERFLOW was chosen for this study since it is a "real" simulation code for aeronautics design studies with wide user base. It is a CFD scheme for solving the Navier-Stokes equations on a structured overlapping grids. Complex geometries encountered in aeronautical bodies are decomposed into a union of overlapping, body-fitting grid zones. Data for holes and overlapping parts of the zones are interpolated from their neighbors. The solution proceeds by updating the boundary conditions after each time step iteration. Various parallel versions of OVERFLOW are available [8]. To establish a benchmark, and test the functionality of the integrated co-visualization system, we used minimal parallelization achievable by running the 'pfa' preprocessor to automatically discover parallelism in the source code. This also enables the multiprocessing directives, if present. OVERFLOW presents an opportunity to provide a co-visualization tool for the CFD investigators.

The visualization software used is pV3, [9], which is suitable for interaction with a parallel simulation running on a distributed memory MPP. A desirable characteristic of a visualization system for co-processing is that it provides loose coupling with the simulation system and that modifications to the simulation code are minimal. Loose coupling will allow plugging in and out of the simulation as desired. This will minimize the visualization processing overhead and will alleviate the need to attend the simulation session all the time. The advantage of keeping changes to the simulation code at a minimum are obvious. pV3 satisfies the two characteristics, above, nicely. Two subroutine calls are added to the simulation code, one to initialize pV3 and another to update when the solution space changes. Some interface routines need to be written to cast the coordinate and solution data into the pV3 format.

While the CFD computations are being performed on the MPP, one or more of the compute nodes are exposed to the network depending on the programming model employed. A user may start the visualization session from a workstation and interacts with the simulation which is already running. Communication between the MPP and the workstation is done under PVM while the visualization session is in progress.

On the workstation side, the user selects the visualizations to be displayed locally. pV3 communicates these choices to the extractor routines on the solver side. During the solver iteration, a call to the pV3 update routine is made and the new solution data is made available to pV3. If there are outstanding visualization requests, they are fulfilled.A minimal set of the extracted data is calculated for transmission back to the workstation. One of the common problems in MPP visualization is to decide how much of the data needs to be sent and where it is needed for display. pV3 deals with data transfers efficiently by calculating and sending "extracts", a minimal description of the visualization which is just sufficient for rendering on the graphics workstation. This data is transferred to the workstation in a lock-step manner to provide time-accurate synchronization. The visualization system can be started and stopped on the graphics workstation while the simulation continues to progress.

3. The Results

The behavior of parallel visualization on DSM multiprocessors is not well understood. However, there has been some research and development on parallelization of CFD solvers for this architecture.Various approaches to visualization and data communication in this non-uniform memory access architecture

need to be examined. Therefore, our initial results for this study comprise of demonstrating the basic system functionality and obtaining parallel performance and scaling data for a minimal automatic parallelization of the solver. These results will be useful for quantifying future improvements in performance through various parallelization and communication strategies. The test problem is compressible air flow past a cylinder with a hemispherical end. This 'ogive cylinder' has a single-zone of 50x61x69 grid points, characterizing a small problem compared to large CFD simulations of practical interest.Test runs were made on various number of processors in the range of one to 254. Each test was run for a 100 iterations with a standard set of visualization computations. Since the visualization decisions are made interactively, the selected set is repeated for each run. For comparison, the amount of visualization calculations for each run are qualitatively the same. The visualization set for each run included calculation and display of two density isosurfaces, 10 instantaneous streamlines, 20 particles paths seeded from circular and straight line patterns, and intermittent bubbles from the current cursor position. Since the system works in a lock-step mode all data had to be calculated and displayed on the workstation before the simulation would proceed to the next step.

Table 1 shows the execution times, in seconds, for 100 iterations of each test run comprising of the solver calculations, visualization computations, and communications. Various components of the timings are also included.

Table 1: Execution Times for 100 Iterations (seconds)

No. of Procs.	Total Exec. Time (s)	Total Vis Time (s)	Max vis time/ step (s)	Vis as% Total Exec Time
1	589.1	108.6	2.2	18.4
2	534.0	107.5	2.2	20.1
4	409.6	113.3	2.2	27.6
8	408.1	121.0	2.3	29.6
16	394.4	111.8	2.2	28.5
32	420.1	108.7	2.2	25.9
64	471.2	116.4	2.2	24.7
128	472.3	111.9	2.2	23.7
160	504.2	121.2	2.2	24.0
254*	647.2	115.1	2.2	17.8

*Only 254 of the 256 processors are available to a user.

As evident from Table 1, the total execution time of the system decreases between 1 to 16 processors, but gradually gets worse as we increase the number of processors in the system. Total visualization time only varies narrowly indicating the serial nature of the visualization communication. The poor scaling with the higher number of processors is due to the fact that the processors are spending more time shuffling the data around than actually computing. Had the problem had multiple grid zones, the performance could have actually been even worse. This communication timing can be improved upon by employing an explicit parallel communication strategy resulting in better inter-processor communication. Explicit mes-

sage passing will also generate multiple, 'heavy threads' which would be visible to pV3 on the network. This will invoke parallel data transfer to pV3 and enables parallel computation/parallel visualization, the larger scope goal of this study. Column 4 is the maximum time for visualization during one iteration in the test. This time is indicative of the interactivity of the system. It shows that with the type of visualizations selected and the size of the problem at hand, visualizations resulting from each time step were updated at a rate of no longer than little over two seconds. The last column is a measure of the percentage of overall execution taken up by visualization. The variations are in a narrow range from 18 to 30 percent. This percentage needs to be kept small so that the simulation generates data fast enough for visualization calculations.

Figure 1 shows a snapshot of a frame of the flow field with geometry, density isosurface, streamlines, particle traces and a cutting plane.

The data obtained from this case study indicate that combining visualization and computations in an MPP with the O2K type architecture is not trivial and the performance of the system is affected by many factors. One is that automatic parallelization utilizing 'pfa preprocessor' does not provide proper scaling and performance for the system.Performance deteriorates beyond16 processors and is actually slower than the single processor serial version of the code if we go past 160 processors. It is to be noted that this convoluted performance behavior is due to the choice of parallelization made. It can be improved upon significantly by going to a more explicitly parallelized programming mode.

In this study in effect we have performed a parallel computation, serial visualization process, model 2 of parallel visualization elucidated in the introduction. As mentioned above, in order to utilize the parallel capability of pV3, "heavy threads" need to be utilized, where more processors are accessible by pV3 for parallel data transfer. One method to accomplish this is to utilize an explicit message passing strategy; few of these are available at NASA Ames and are the subject of the future work of this study.

Acknowledgment

I thank Dennis Jesperson for providing the OVERFLOW code and various clarifications and advise during the course of this case study. Robert Haimes provided the pV3 visualization software and some of the interface routines.

References

[1] Woodward, P. R. 1993. "Interactive Scientific Visualization of Fluid Flow."*Computer*, Val. 26, No. 10.

[2] Rowlan, J. S. and B. T. Wightman. 1994."PORTAL: A Communication library for run-time visualization of distributed, asynchronous data." *Proceedings of the 1994 Scalable High-Performance Computing Conference*, Knoxville. May 23-25.

[3] Haimes,R. and T. Barth. "Application of the pV3 Co-Processing Visualization Environment to 3-D Unstructured Mesh Calculations on the IBP SP2 Parallel Computer." CAS '95 Workshop, NASA Ames Research Center, March 1995.

[4] Vaziri, A. and M. Kremenetsky, Visualization and Tracking of Parallel CFD Simulations, Proceedings, High Performance Computing (HPC'95), Society of Computer Simulation, Phoenix, AZ, April 9-13, 1995

[5] Kremenetsky, M., A. Vaziri, and R. Haimes: "Real-Time Visualization of an HPF-based CFD simulation, Parallel CFD 96, Capri, Italy May 96

[6] Parker, S. and C. Johnson: "SCIRun: A Scientific Programming Environment for Computational Steering. SC95, 1995

[7] Buning, P. et al.:" Overflow User's Manual", NASA Ames Research Center, 1998.

[8] Jespersen, D.: "Parallelizing OVERFLOW", NASA HPCCP/CAS Workshop, August 1998.

[9] Haimes, R., 1994: "pV3: A Distributed System for Large-Scale Unsteady CFD Visualization." *AIAA Paper 94-0321*. 32nd AIAA Aerospace Sciences Meeting. Reno, Nevada.

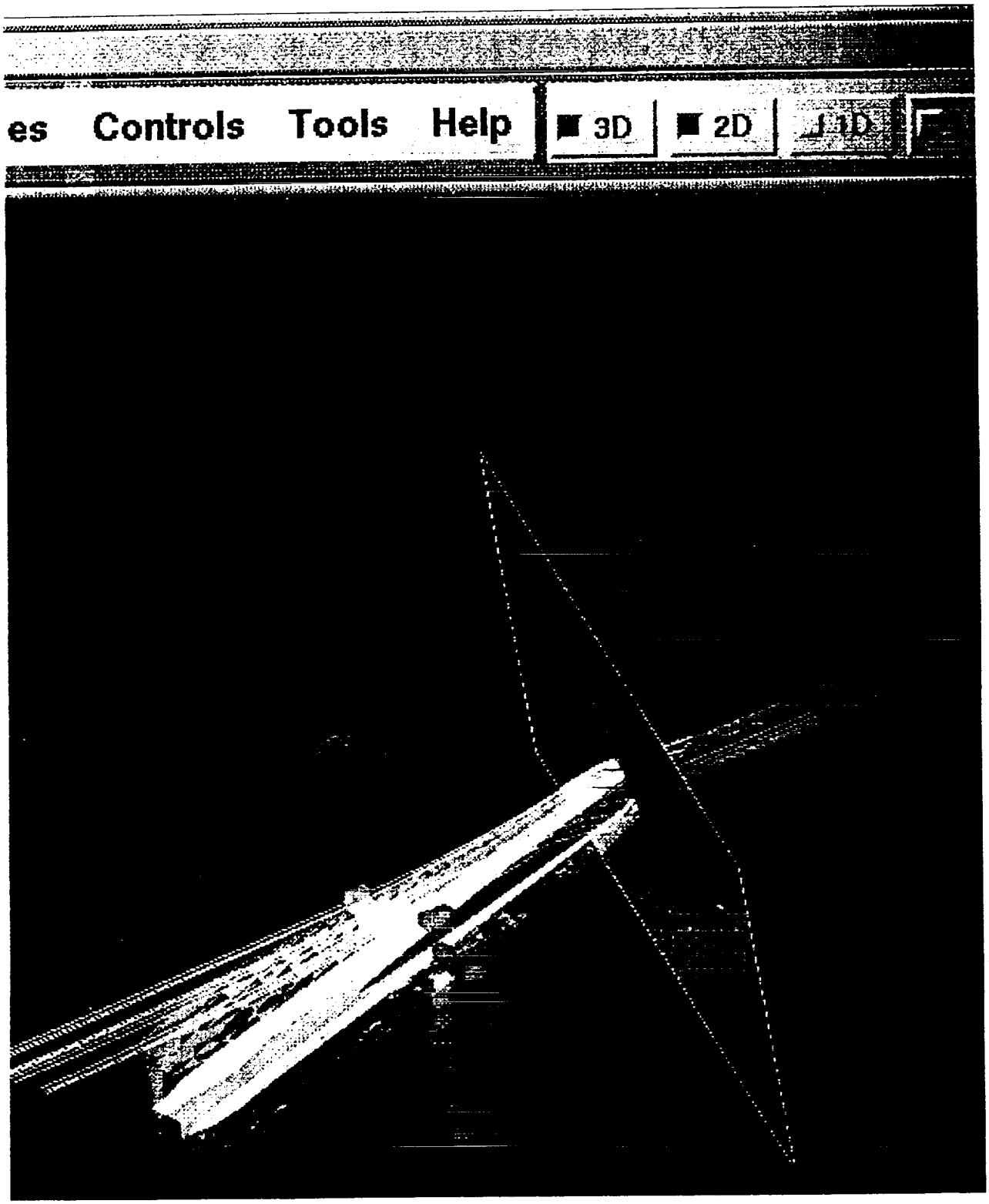

Figure 1. Flow past an ogive cylinder. The standard set of visualizations used for each test run. Two isosurfaces of density, 10 instantaneous streamlines, 20 particle traces and an intermittent bubble emitter.