Scientific Computing Paradigm

John Van Zandt [1]

Report RND-94-002 January, 1994

NAS Systems Development Branch
NAS Systems Division
NASA Ames Research Center
Mail Stop 258-6
Moffett Field, CA 94035-1000

Abstract

The usage model of supercomputers for scientific applications, such as computational fluid dynamics (CFD), has changed over the years. Scientific visualization has moved scientists away from looking at numbers to looking at three-dimensional images, which capture the meaning of the data. This change has impacted the system models for computing. This report details the model which is used by scientists at NASA's research centers.

---

1. This paper was written while the author was an employee of Computer Sciences Corporation, NASA Ames Research Center, Moffett Field, CA 94035-1000.

1.0 Introduction

Back in the dark ages of computing, all of 10 years ago, scientists viewed computers as machines to perform calculations and produce tables of numbers, much the way science had been done for hundreds of years [1]. Scientists would produce reams of computer paper filled with columns of numbers, which somehow would yield insight into nature. The design of a computer system to support this work was well understood: provide fast computation of numbers, printers to produce the results and some disk space for program storage. The era of networks, file servers, disk farms, etc. had yet to come.

Nowadays, scientists are foregoing the tables of numbers in favor of scientific visualization. Their goal is to utilize the pattern-recognition capability of the human visual system to help in the understanding of the mathematics of the many different disciplines. This change is causing the computational systems to change. No longer are printers sufficient for output. Storage requirements for data have increased dramatically. Networking and workstations are the norm. Collaboration by scientists located across the country, using resources in still different places, is desired.

This study focuses on the overall system architecture of a computing model for this new environment. The analysis is designed to direct technology efforts toward providing a complete environment to support scientists using supercomputers. Though the analysis is based on information collected at NASA, the thrust of this study is representative of usage in other organizations. It also is biased toward those scientific approaches which are just now becoming feasible, such as time-accurate visualization of unsteady flows (animated pictures of fluid flow in real-time). It is expected that over the next five years, the dominant form of visualization will move from analyzing a single 3D image to analyzing moving 3D images. This move will require new storage and computational capabilities.

2.0 The Approach

Interviews were conducted with scientists and system developers to understand the current and projected usage models of supercomputers. These were validated by users of NASA's Numerical Aerodynamic Simulation (NAS) systems as well as users at other NASA research facilities. The workloads for the major applications were then analyzed in terms of compute cycles, storage requirements, solution data generated, and visualization techniques.

3.0 The Results

Two forms of potential use were identified: post-processing and computational steering. Each is described below.

3.1 Post-Processing

Post-processing is the form currently in use in the majority of the cases studied. It is characterized by long-running flow-solver applications, such as those for computing the fluid-flow around an aircraft wing, running in batch on a supercomputer. These jobs may run anywhere from hours to months, depending on the patience of the scientist.

While one of these post-processing applications is running in batch, it is computing a solution file (or a set of solution files representing individual time-steps in the simulation). After a solution file is complete, a scientist will sit down at a display and proceed with the analysis. In post-processing, the scientist does not interact with the flow-solver application.

The visualization is either of one time-step (solution file) or is of several time-steps. The user is usually looking at a three-dimensional image, and may perform geometric transformations on the view. The visualizations, whether of one or more time-steps, are often animated to provide further understanding of the flow fields. Scientific visualization applications, such as FAST [2] or PLOT3D [3], allow the scientist to display many aspects of the flow field and the underlying model.

3.2 Computational Steering

Computational steering is a new concept which allows a scientist to interact with the flow-solver application directly, while it is running. An example from CFD would be for a scientist to be watching the solution of turbulence and decide to modify the grid in order to focus in on an unexpected eddy, as the application is running. Or the scientist might detect that the solution is not converging as predicted and go in and modify the equations, again, as the application is running. This *steering* of the application would require computation to occur in minutes, not hours or days. For algorithm studies and debugging, this will undoubtedly be useful. In the opinion of this author, steering will not be a dominant usage model of supercomputers over the next five or ten years. Therefore, the rest of this paper will consider only the post-processing model.

3.3 System Model

Figure 1 shows the functional model which was developed to describe and analyze system usage for post-processing. Work and data progress from left to right.

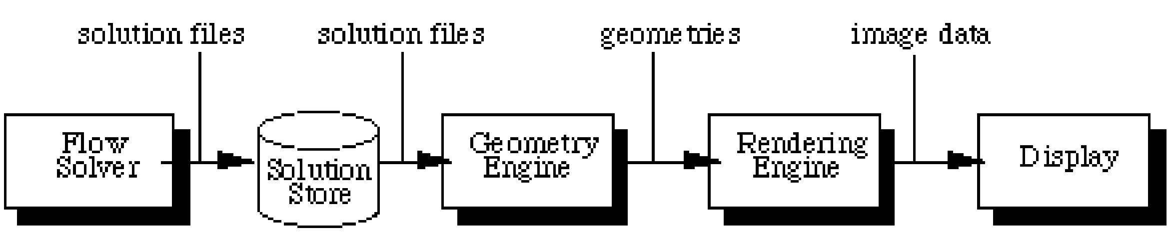


Figure 1. System Model

The boxes represent functions (performed by either software or hardware) which are in the scientists' work-stream. The lines represent information flow between the functions. Several assumptions were made regarding the capabilities of the functions. These will be discussed below. The basic rule was to try and make assumptions based on practical limitations. For example, it is assumed that the *rendering engine* has enough memory to hold at least one entire frame of a moving image, and possibly several frames, but not enough to hold hundreds of frames. Also, though compression can certainly be applied, it will only be by a factor of 2-4, which will not change the basic assumption of data set size made below.

Figures 2 and 3 show different current approaches to decomposing the application domain. FAST is being augmented to assist in the visualization of time-accurate data sets. As shown, it currently runs completely on an SGI workstation. Distributed PLOT 3D, and PLOT 4D both use a Cray or Convex as the backend for the initial step in creating an image. The SGI workstation is then used as the rendering engine in this model.

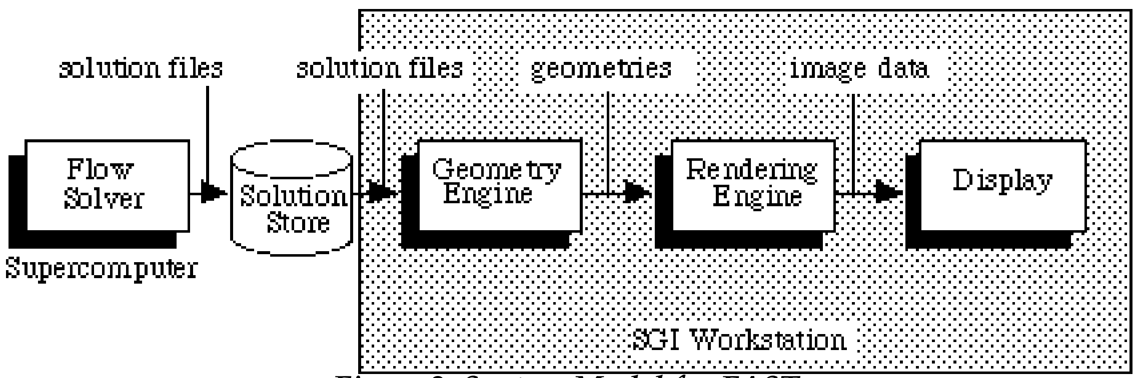


Figure 2. System Model for FAST

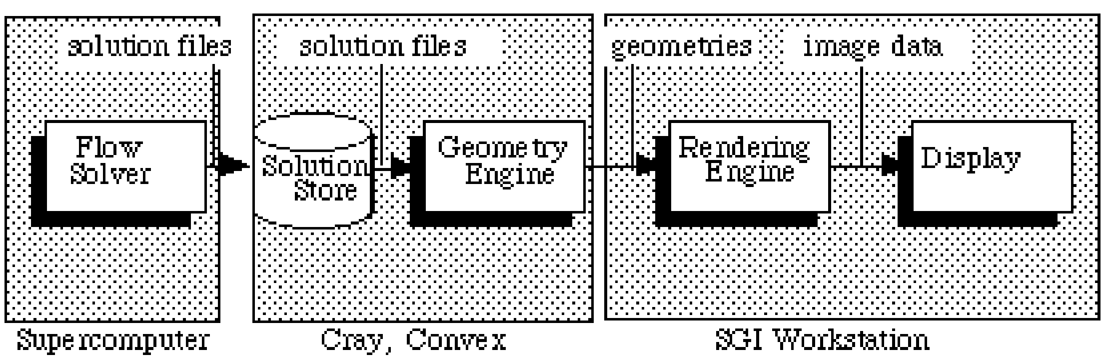


Figure 3. System Model for Distributed PLOT 3D and PLOT 4D

3.3.1 Flow Solver

The *flow solver* is expected to be the large super-computer which actually runs the flow-solver application and produces solution files. For all applications analyzed, the inputs are small relative to the outputs generated. It is assumed that the inputs, intermediate files and solution files place a negligible demand on the model.

3.3.2 Solution Store

The *solution store* holds the results from the run of the flow-solver application. For steady flows, the solution may only be a single time-step representing the steady-state of the flow field. For time-accurate visualization, this file will contain hundreds (or thousands) of time-steps, which will be viewed in animation. The solution store in this model does not hold intermediate results or restart files for the application.

3.3.3 Geometry Engine

The *geometry engine* converts the data from the solution files into the geometric objects that the user requests. Its role is to create a geometrical representation of the data. We assume that the geometry engine can hold at least one, but not all time-steps necessary for an animated sequence. The data for a complete set of animation frames, when represented as geometries, is currently on the order 8-16 gigabytes. In a few years, a complete set of animation frames might be 100 gigabytes. The assumption is that the geometry engine is a shared resource and that there will be multiple concurrent users. This would preclude local storage of a complete animation sequence.

3.3.4 Rendering Engine

The *rendering engine* is used to convert a 3D geometric representation of the graphical objects into the 2D color image appropriate for a display device. The rendering engine also performs all necessary geometric transforms requested by the user (e.g., scaling and rotations). It is assumed that the rendering engine can hold at least one complete time-step for display, but does not have enough storage to hold an entire animated sequence. The reason for this is the practical amount of storage which would be placed close to a scientist. As with the geometry engine, the data could be on the order of 8-100 gigabytes.

3.3.5 Display

The *display* is typically a 1280x1024 color graphics display. In the future, this might be a stereo display with a pair of goggles. The display is considered to have no storage. For effective animation, the image data must be transmitted at a minimum of 15 frames/second, with a total bandwidth of about 57MB/second. Ideal frames/second rate is between 24 and 30.

3.4 Data Requirements

For the vast majority of applications at NAS, problems can be characterized as a simulation of physical properties of real objects. These applications are based on a grid representing the object in question, where the points contain values of the physical characteristics of the object at that point. Calculations are then applied to each point to simulate physical conditions. For example, with a grid representing an airplane, equations would be solved for each grid point around the plane, to simulate the density, momentum and stagnation energy of air flow. In time-dependent data, the calculations are repeated for each time step, as the data is changing. The output from these applications is one or more of these time steps.

An analysis of CFD flow-solver applications' impact on the model was done by determining the average and maximum grid sizes, the number of time-steps saved, and the number of GFLOPS required to compute each time-step. Based on this information, shown in Figure 4, the solution storage required and the bandwidth needed were calculated.

	Avg. Solution Size (MB)	Solution Time (hours)	Avg. Rate (MB/sec)	Max. Solution Size (MB)	Solution Time (hours)	Max Rate (MB/sec)
FY92	9667	17.99	0.15	64000	70.71	0.25
FY93	18980	22.55	0.23	80000	60.76	0.37
FY94	34865	40.84	0.24	120000	91.15	0.37
FY95	61184	21.35	0.80	160000	37.39	1.19
FY96	71539	24.91	0.80	160000	37.39	1.19

Figure 4. Application Data Generation Over Time.

The data traffic between the solution store and the geometry engine was predicated on the need of the geometry engine to sustain a rate of 15 frames per second, which is what the users would like (it should be noted that this is not achievable today with complex visualizations). The assumption, as stated above, is that the geometry engine does not have the storage to hold an entire animation sequence. Figure 5 shows an analysis of data traffic to the geometry engine.

Figure 5. Data Needs of the Geometry Engine

	Avg. Rate (MB/sec)	Max Grid	Max Rate (MB/sec)
FY92	456	2500000	800
FY93	584	2500000	800
FY94	925	3750000	1200
FY95	1280	5000000	1600
FY96	1296	5000000	1600

The output from the geometry engine is a set of polygons which have enough information so that they can be rendered in a colored and shaded 3D scene. The amount of data is dependent on the type of image to be viewed. For NAS applications, three main ways of representing data are *particle traces*, *cutting plane*, and *iso-surfaces* (see the technical report by Gerald-Yamasaki [4]). Other common ways of representing data include: scalar-mapped computational surface, contour lines and vector plots.

Particle traces generate the least amount of data. A particle trace is the simulated stream of particles through the flow-field from a given point. The 3D image of the object to be viewed is transferred once, then a series of lines or vectors are drawn representing the trace. Adding these lines or vectors has no significant impact on the data volume.

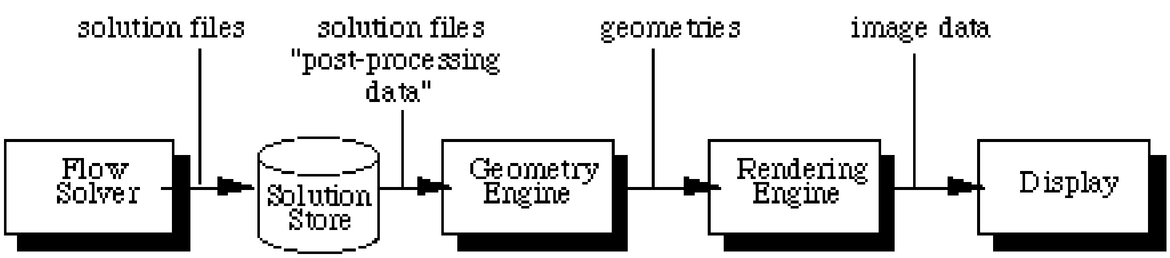
Cutting plane views are generated by taking a 2D slice (a plane) through the 3D solution grid and doing appropriate coloring. Since the amount of data is dependent on the grid size and on the location of the slice, an estimate is used to calculate the storage space required. This estimate assumes that the number of polygons on the cutting plane is two times the two-thirds root of the number of grid points. Thus, for a 100x100x100 grid, the number of polygons would be 2x100x100.

Iso-surfaces are the hardest to estimate. They represent 3D contour surfaces for a fixed number of contour values. These are estimated by taking five times the two-thirds root of the number of grid points for the number of polygons. Thus, for a 100x100x100 grid, the number of polygons would be 5x100x100.

Based on the above equations, Figure 6 shows the data generated by the geometry engine which must be transferred to the rendering engine.

	Avg. Grid Points	Avg. Timesteps	Avg. Size (MB)	Avg. Rate (MB/sec)	Max Grid	Max Timesteps	Max Size (MB)	Max Rate (MB/sec)
FY92	1425000	212	107.38	10.74	2500000	1000	29841	298
FY93	1825000	325	194.14	19.41	2500000	1000	29841	298
FY94	2890000	377	305.96	30.60	3750000	1000	39102	391
FY95	4000000	478	481.79	48.18	5000000	1000	47369	474
FY96	4050000	552	561.01	56.10	5000000	1000	47369	474

Figure 6. Data Generated by the Geometry Engine



	Solution Generation Data Rate (MB/sec)	Post-Processing Data Rate (MB/sec)	Geometry Data Rate (MB/sec)	Image Data Rate (MB/sec)
FY92	0.25	800	298	236
FY93	0.37	800	298	236
FY94	0.37	1200	391	236
FY95	1.19	1600	474	236
FY96	1.19	1600	474	236

Figure 7. Data Rates Mapped onto Model

By looking at the data rates between the different functions, it is clear that the place with the least bandwidth requirements is between the flow solver and the solution store. With a maximum need of 1.19 MB/second, this is well within the technology for wide-area networks. The bandwidths for the image data and the geometry data are also close to what is promised by technologies for local area networks, in the gigabit/second range.

The post-processing data rate (the data rate between the solution store and the geometry engine) is noticeably higher than current—or near-term-foreseeable—technology allows. Current-day technology for I/O subsystems allows only about one gigabyte per second, and that on a Cray C90. But the processing demands for geometry creation do not require supercomputer performance. Further work needs to be done to determine if a less-expensive solution can be found to satisfy this high I/O rate with a smaller processing engine. It is likely that in the future, a parallel processor will be created that is focused on this I/O problem.

4.0 Conclusions

Several things come out of this study. First, in an era of distributing resources, the best place to put a wide-area net is between the flow solver and the solution store. This does not mean the flow solver needs only a small amount of I/O bandwidth. Data shows that a flow solver does a significant amount of I/O due to task swapping/virtual memory, temporary storage management, and general interactive use. The amount of I/O needed for the flow solver (internally) is an area which needs to be studied and will be the topic of a future technical report.

Second, as scientists move to time-accurate visualization, the bandwidths needed to support it are well beyond current technology. A significant area of research and development is needed to determine how to provide the bandwidth between the solution store and the geometry engine. This looks like a fruitful area for research.

Third, the role of data compression should be investigated. If an order of magnitude reduction in the bandwidth requirements can be made, then this could enable production use of time-accurate visualization sooner than technology would otherwise allow.

5.0 References

[1] Bailey, James, “The Ghosts of Computers Past,” Proceedings of the Conference on Scientific Applications of the Connection Machine, ed. Simon, World Scientific, 1992.

[2] Bancroft, G. V., Merritt, F. J., Plessel, T. C., Kelaita, P. G., McCabe, R. K. and Globus, A., “FAST: a multi-processed environment for visualization of computational fluid dynamics,” Proceedings of Visualization ‘90, San Francisco, CA, October 23–26, 1990, pp. 14–23.

[3] V. Watson, P. Walatka, G. Bancroft, T. Plessel and F. Merritt, “Visualization of Fluid Dynamics at NASA Ames”, *Computing Systems in Engineering*, vol. 1, pp. 333–340, 1990.

[4] Gerald-Yamasaki, M. J., “Interactive and Cooperative Visualization of Unsteady Fluid Flow,” NAS Technical Report, March, 1992.