Using PVM and MPI for Co-Processed, Distributed & Parallel Scientific Visualization

Robert Haimes
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology
haimes@orville.mit.edu

and

Kirk E. Jordan
IBM Corporation
404 Wyman Street
Waltham, MA 02254
kjordan@us.ibm.com

January 30, 1998

Abstract

This paper discusses the combined use of MPI and PVM in the parallel visualization tool kit, pV3. The implementation provides for efficient co-processed, distributed parallel visualization of large-scale 3D time dependent simulations. The primary goals of pV3 include the ability; to handle large-scale transient 3D simulations, to take full advantage of available hardware encompassing both parallel compute engines and graphics workstations, to visualize the data as the computation progresses, to interact and interrogate the data as its computed, and to steer the simulation by adjusting parameters obtaining immediate feedback from the changes. Based on a client/server model, the original implementation of pV3 was founded on PVM. The client portion executes coupled closely with the application code and 'extracts' from the distributed data volume, in place, lower dimensional data. The distilled data is transferred to the interactive server portion of pV3 (executing on a graphics workstation) where the rendering and display are performed. This unique architecture frees up the compute engine(s) from dealing with the graphics, and reduces the large data-set to a size that can be rapidly transported to the server using standard network connections.

There are some drawbacks in using PVM on parallel machines which stem from TCP/IP communications. These include, in some instances, not taking full advantage of the hardware interconnect on the machine as communication must occur with the graphics workstation. This can result in smaller messages and increased latency. In addition, pV3's dynamic attachment required that the

graphics workstation be accessible from each node on the machine with PVM's 'group' functionality intact, which poses a problem on some parallel systems. The pV3 'concentrator' module was designed to overcome such problems on distributed memory platforms. All pV3 messages on the parallel machine use the high-speed interconnect via the MPI protocol. Communications off the parallel machine to the graphics workstation is handled via PVM. Using the communicator facility of MPI, the concentrator is able to keep the solver messages separated from the pV3 client messages. An added advantage of the concentrator module for a large number of processors is that it simplifies the pV3 start up procedure.

1 Design Goals

During the early development of pV3 [1, 2] the following considerations influenced the design and development of this software system:

- Large-scale Transient 3D Simulations
  Provide a platform for the visualization of unsteady Computational Fluid Dynamics (CFD) calculations on unstructured as well as structured meshes. The supporting grids may be on the order of millions of nodes. The volume of interest may have been partitioned to take advantage of cluster computing or distributed memory hardware.
- High Performance
  Take advantage of the available hardware to get the best performance out of the entire compute arena. To do this, portions of the visualization system must run on the equipment where the data is being generated.
- Co-processing
  An important concept is the ability to visualize the data as the solver or model progresses in time. The co-processing scheme must be able to fit in, or adapt, to the solver. Usually a great deal of effort goes towards making the solver run efficiently. The visualization of the results must be subservient to the solution scheme and easily coupled. An interesting example of an attempt to do this is the code CUMULVS [3] which uses PVM and AVS as the visualization system. Unfortunately for large scale problems the entire field must be transmitted to the viewer and this can overwhelm any network. The visualization tool-kit should provide a complete CFD-style viewer.
  Dynamic attachment is important. If the solution procedure takes hours to days, the visualization must be able to 'plug-into' the calculation, allow viewing of the data as it changes, then can 'unplug'. The viewing must be available away from the hardware performing the calculation. The ability to go to some other office and initiate the viewer to interact with a college is a significant feature.
- Interactive
  This goal is of utmost importance. Interactivity is a requirement that facilitates interrogation of the data. If batch processing of a 'movie' is all that is done, then all that can be viewed and understood is what has been 'storyboarded' a priori.

The goal of any scientific visualization package should be to allow the assimilation of the vast amounts of data produced by the models and solvers in order to better understand the underlying physics. Movies are important for presentations and communication to others on the design team, therefore they should be made after the interrogation. Movies should never be the sole visualization method.

- Steering
  Solution steering is, of course, possible if one can watch the results of an unsteady simulation as they progress. Though it has always been a vision of many CFD researchers, today most 3D transient simulations do not allow solution steering. This is because most systems do not provide co-processing for the CFD application.

2 PVM Implementation

pV3 has been successfully used in cluster environments using PVM for data movement and process management. For the first pV3 releases the visualization software consisted of two parts, see Figure 1. The client portion executes on the compute engine(s) closely coupled with the solver. This allows the reduction of the distributed data-set, in-place, to manageable lower-dimensional 'extracts'. The distilled data is transferred to the interactive server portion of pV3 (executing on a graphics workstation) where the rendering and display are performed. This unique architecture frees up the compute engine(s) from dealing with the graphics, and reduces the large data-set to a size that can be rapidly transported to the server using standard Ethernet or FDDI connections.

Certain rules must be followed in using pV3 with multiple clients where the simulation is also using PVM for message passing. This is because PVM does not currently provide a mechanism to segregate messages (i.e., solver messages and visualization messages). Therefore, these rules insure that messages for the visualization and those for the compute task(s) do not interfere with each other.

- Open Send Buffer
  It is assumed by pV3 that the default send buffer is free and available for use. This should not be a restriction because any of pV3 client-side calls should be executed at times when inter-client solver communication is at a completed stage.
- Broadcasts
  Broadcasts need to be avoided. The task will end up sending messages to the pV3 server(s). The server will report then ignore messages without the proper pV3 signature. In general, any client need not send the pV3 server any messages (all the data communication necessary is handled internally by pV3). If a broadcast facility is required, the multiple-cast send can be used, and messages sent only to known tasks.
- Receives
  Wild-cards for the task id in the receive calls to PVM must be avoided, otherwise pV3 requests will be trapped.

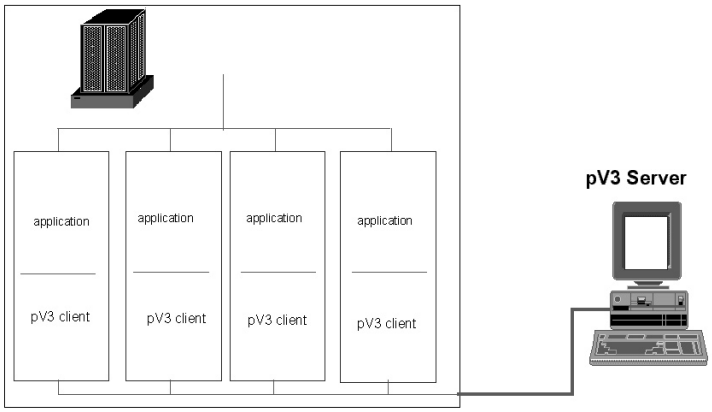

Fig. 1: pV3 - computational-visualization co-processing environment. Each pV3-Client communicate directly with the pV3-Server (the graphics workstation).

The message collection performed by the pV3 client-side only takes messages from known pV3 server(s), leaving other message traffic alone. The server tasks are identified by using the PVM group routines. The group concept is also used to allow for the dynamic attachment of the server(s) to the executing solver. Every time the solution is updated and the pV3 client-side call is made, the number of members in the group pV3Server is checked. If no servers are found, this routine returns (essentially providing a NO-OP). When a server starts it enrolls in a specified group. The next time into the client-side, an initialization message is processed to the new server and that visualization session begins. Each subsequent time pV3 is called, visualization state messages and 'extract' requests are gathered, the appropriate data calculated, collected and sent to the server.

When the user is finished with the visualization, the server sends a termination message and exits. The clients receive the message and clean up any memory allocations used for that visualization session. When all servers have terminated, the pV3 client-side code reverts to looking for server initialization via enrollment in the PVM group.

3 Communication patterns

Robustness is an important characteristic of any attached system. pV3's design keeps all visualization state information within the server. Requests for data are

sent out (based on the state, what data is current, and the new data required server-side). Each client acts on only the request and sends back the resultant data. The servers are designed so that if any of this data is not received, the process can continue. These requests are generally small and are concatenated together for each client so that one message can be sent to request most of the information required. The resultant data from a single request can be voluminous. This is usually only the case for tools where the result is a surface (e.g., an iso-surface). Where possible, the data sent back to the server is also concatenated (but pV3 does maintain a internal limit on the size of any single message).

Most scientific visualization tools are embarrassing parallel. This is due to the fact that the algorithms can be cast to work on a cell at a time, within the volume or partition. Therefore, no information is required outside the domain so the communication is only server-client and client-server. This means that the extraction portion of the visualization is performed in parallel with efficiencies approximating balance (if the data has been partitioned based on cells). But, there is a suite of tools that are serial by nature. These are instantanous streamlines and unsteady particle traces. These tools integrate positions through the volume and may require passing from one partition to another. Therefore, these operations can potentially stall the visualization (and the solver) waiting for the information to be passed thoughout the system.

The problem is relieved by prioritizing the messages and dealing with two classes of requests and resultant data. Low priority requests (those from the parallel tools) are only dealt with after all high priority requests (from the serial tools) have been exhausted. All resultant data back to the server is low priority, but requests to continue the serial operation in another partition are high priority. This high priority data is handled at the server before any normal data and bounced to the appropriate client(s) as a new high priority request (usually not concatenated with any other). Direct client to client pV3 communication is avoided for simplicity and robustness (also the server can track the process of these serial operations). See Sujudi and Haimes [4] for a more complete description of this procedure.

Dealing with two classes of messages within PVM is done via the message ID. Some messages IDs have high priority and others are low priority. This is simplified in the coding of pV3 by changing the receive matching function (this also assists in the avoidance of non-pV3 messages at the client-side).

4 MPI & Concentrator

pV3's dynamic attachment requires that the graphics workstation be accessible off the compute hardware with PVM's 'group' functionality intact (this is a problem for both the CRAY T3D and the Intel Paragon). Also, for most installations of IBM SPs the Ethernet interface may be the only avenue to the graphics workstation. This may require that the simulation run on the slower Ethernet network between the nodes instead of the high-speed interconnect (particularly if the solver also uses PVM for message-passing). Not running over the switch

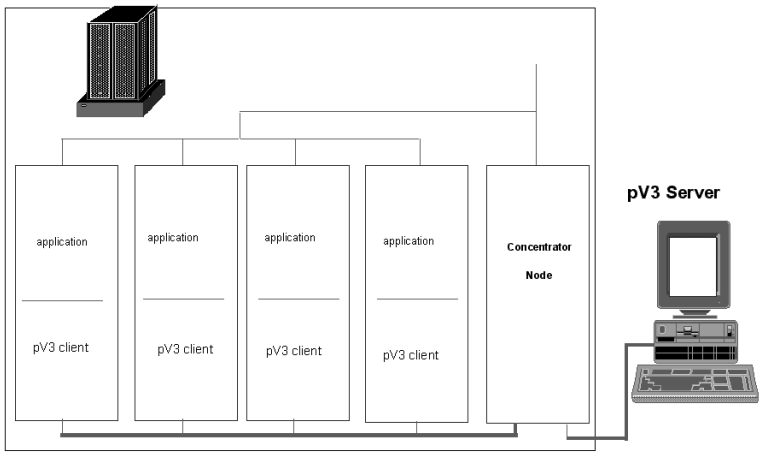

Fig. 2: pV3 - computational-visualization co-processing environment. Each pV3-Client communicates with the "concentrator node" which communicates with the pV3-Server (the graphics workstation).

poses significant performance penalties in comparison to on-switch bandwidth and user-mode latencies.

Ethernet, specifically, and TCP/IP communication in general, is the major contributor of performance problems that effect the visualization and therefore the solver when coupled together. Each Ethernet packet may encounter as much as 1 millisecond of latency. Early versions of pV3 required at least two packets per 'extract' per client; a request for the data from the server and then the resultant data back from the client. If the job uses 16 clients and the investigator is viewing only 10 'extracts' then 0.32 seconds are wasted in latency alone. The other problem with Ethernet is the lack of bandwidth that is improved with other "wires" used in TCP/IP supported networks (such as FDDI or HIPPI).

The pV3 "concentrator" module is used in order to solve the latency, bandwidth and PVM 'group' problems for most distributed memory platforms. This concentrator executes on the front-end of the machine (or a node that has an appropriate off-machine high-speed network interface), see Figure 2. All pV3 messages on the machine use the high-speed interconnect via the MPI protocol. Communication off the computer is performed by the concentrator using PVM to the graphics workstation in a point to point manner. Off-MPP communication is further improved by the concatenation of smaller messages at the concentrator to avoid the accumulated latency encountered by many packets. Some additional details on applying this approach to an application can be found in [5].

A minor change to the pV3 client-side API was needed to support the addition of this software component. This change was required because the MPI specification [6] does not define how to start a mix of tasks (the solver components and the concentrator). Some MPI executives (or startup scripts) assume that all the tasks to be started are the same making the mix impossible.

To enable the concentrator to start, a call was added to the pV3 client side API that allows one task to be specified as the concentrator (and does not return for that task). Also a new MPI communicator is returned from this call for all other tasks. This communicator does not have the concentrator included and must be used for the solver's communication (instead of MPI_COMM_WORLD). A different network module is required at solver link in order to use MPI (instead of PVM). When the server starts at the graphics workstation, the concentrator acts like a normal pV3 client, but passes through the requests to the actual clients and transmits back the requested data. When pV3 reports any client data (such as from the point-probe) the actual client-ID is displayed (not the ID from the concentrator) for proper feedback to the investigator.

The concentrator uses two MPI communicators internally. The first is used for low priority messages (the normal requests and data back from the clients). The high priority communicator is used for all pV3 serial operations and communication on change of status for the servers active within the system.

Another advantage to this network mode is the ease of startup. PVM clients need PVM functioning at initialization. If the PVM daemon is not running, the startup and enrollment fails. This routine returns an error indication. The application programmer can test for this and try to initialize at some later time. The concentrator simplifies the startup situation. It does not require PVM functioning at initialization. Under these circumstances, the concentrator waits for PVM to start then starts checking for the existence of any pV3 server. The MPI clients never return with the PVM error condition.

5 Future

While care has been taken to minimize network traffic on parallel computers and between the parallel machine and the graphics workstation, the effort here has been mostly qualitative rather then quantitative. An important aspect of any future work is to put in the necessary hooks to pV3 to quantify both the computational and the network loads. Through the development of appropriate metering and analyzing the performance the identification of computational and network bottle-necks can be accomplished. The performance metrics will also assist with implementing alternative algorithms that will better scale with large scale applications running on hundreds of nodes. This will also help in understanding how to best use new architectures such as SMP (symmetric multi processor) nodes and non-unified memory systems like the SGI Origin 2000.

Specifically, this is necessary in order to quantitatively measure the complex network traffic generated by the distributed visualization and its impact on the entire compute arena. We need to have these numbers so that we can; a) see

improvements in algorithms on the parallel machines, b) understand scaling based on the number of processors, c) measure the impact on network traffic, d) understand the impact of hardware improvements both with respect to processor speed and number of SMP processors, e) network hardware and protocols, and f) network software such as MPI2. Specifically, network issues such as bandwidth and latency must be addressed so that decisions can be made on how to balance the overall application with the hardware resources available. When automation or concealing these findings is not possible "rules of thumb" will result to assist individuals dealing with these complex applications.

6 Conclusion

In attempting to meet pV3's design goals a fairly complex software suite was generated. Depending on the hardware (and network) architecture PVM and possibly MPI are used for message passing. Due to the limitations of MPI there is no pure pV3 MPI port. This is due to the fact that MPI does not support either dynamic attachment or task initialization at some later time (other than start-up). MPI2 may relieve these restrictions but it is not clear whether it will deal well with heterogeneous clusters (needed when the graphics workstation is not the same type of machine as where the solver runs).

Acknowledgements

We wish to thank David A. Yuen from the University of Minnesota and Mary F. Wheeler from the University of Texas at Ausitn both of whom have worked closely with us and have provided interesting and diverse applications that have been integrated with pV3. We are grateful to IBM for support of this work through IBM's Shared University Research Program and through IBM's University Partnership Program.

References

1. R. Haimes, pV3: A Distributed System for Large-Scale Unsteady Visualization. AIAA Paper 94-0321, 1994.
2. R. Haimes, Unsteady Visualization of Grand Challenge Size CFD Problems: Traditional Post-Processing vs. Co-Processing. Proceedings of ICASE/LeRC Symposium on Visualizing Time-Varying Data, 1995.
3. J. Kohl and P. Papadopoulos, The Design of CUMULVS: Philosophy and Implementation PVM User's Group Meeting, 1996.
4. D. Sujudi and R. Haimes, Integration of Particle Paths and Streamlines in a Spatially-Decomposed Computation, Proceeding of Parallel CFD '95, June 1996.
5. K.E. Jordan, D.A. Yuen, D.M. Reuteler, S. Zhang and R. Haimes, Parallel Interactive Visualization of 3D Mantle Covection, IEEE Computaional Science and Engineering, Vol. 3, No. 4, Winter 1996.
6. Message Passing Interface Forum, MPI: A Message-Passing Interface Standard.May 5, 1994.