HERMES

A Prototype Distributed Application Management System

G. A. Hanyzewski[1], J. Spahr[2], C. R. Mechoso[2], R. W. Moore[1]

[1] San Diego Supercomputer Center
San Diego, CA 92186

[2] Department of Atmospheric Sciences
University of California, Los Angeles
Los Angeles, CA 90024

October 22, 1993

Abstract

This paper presents a prototype system for on-line distributed application management. The system allows researchers to determine the exact state of remotely executing applications by allowing interactive acquisition and visualization of output datasets as they are calculated. This paper discusses the general design as well as implementation details, case studies, and future directions for development.

1. Introduction

The computational scientist can count on the increasing availability of high-performance supercomputer-class machines coupled with increasing network connectivity for the foreseeable future. These capabilities are making it possible to execute numerical simulations on ever more complex domains, at higher spatial and temporal resolutions and for longer periods of time than ever before. Simultaneous with these increasing capabilities is a growing interest in being able to assess the progress and relevance of a simulation while it is running. This is currently difficult to accomplish since the compute platforms tend to be remote from the researcher, and the simulations are usually executed in a non-interactive batch environment.

This paper presents a prototype system for addressing the problems of distributed application management. This system, named HERMES, allows for the transmission and inspection of data generated by an executing model.

HERMES utilizes public-domain communications libraries and takes advantage of widely available network connectivity to allow a user to determine the exact state of a remote application. A simple user-callable library is provided that can be added to any existing Fortran or C application code. This library coupled with a user-executable daemon and a data acquisition or visualization client constitutes the tool. A highly usable prototype system results from keeping the user-accessible portions of the tool as simple as possible and by minimizing the potential negative performance impact of the tool on the computational model.

2. Design Issues

The development of the HERMES prototype tool was driven primarily by the needs of the global climate application being distributed across the CASA Gigabit Testbed. This testbed includes vector processors, scalable parallel processors, and high-end workstations at the San Diego Supercomputer Center (SDSC), Los Alamos National Laboratories (LANL), and NASA/Caltech Jet Propulsion Laboratory (JPL) connected by a high-speed (gigabit / second) network. The distributed application consists of an atmospheric model developed at the University of California Los Angeles (UCLA) [**1**] and an oceanographic model developed at the NOAA/Princeton University Geophysical Fluid Dynamics Laboratory (GFDL). [**2**] The oceanographic model was parallelized at SDSC and UCLA. [**3**]

In the CASA Gigabit Testbed Environment, the distributed application is executed on multiple machines that may be at geographically separate locations that are linked by TCP/IP or raw HiPPI connections. [**4**] Initial experiments had the atmospheric portion running on a high-speed sequential vector supercomputer and the oceanographic portion on a scalable parallel supercomputer. Once started, each portion calculates prognostic and diagnostic datasets for its domain as well as the fields involved in coupling the two models. During execution, the models typically display some diagnostic information, but the majority of the calculated data is written to disk file and placed in archival storage for future analysis and visualization. Problems with the model caused by initial condition errors, modeling inaccuracies, or difficulties with the hardware are difficult to detect in this mode of operation and can be missed entirely until output is analyzed and visualized. This is costly in computer resources and researcher analysis time. The HERMES system was designed to allow researchers to view the results being calculated by their models as they are running. One possible application of this ability is the rapid detection of erroneous or ill-behaving runs or parameters that do not produce the desired phenomenon.

A number of design issues were addressed in the development of this system. The main considerations are listed in Table 1 below.

Table 1: Design Specifications

1. **Simple user interface**. The tool must provide a limited number of easy-to-understand library calls that can be used to add functionality to existing programs.
2. Multi-language support. Simulation interface routines must be callable from C or FORTRAN applications. Client interface routines need to be C-callable only.
3. Minimum performance penalty. Use of tool must have a minimal performance impact on the simulation.
4. Support for distributed and heterogeneous environments. The tool must support multiple architectures, word sizes, and byte orderings transparently to the user and allow distribution of components across the Internet.
5. Hidden details. The details of connection creation and data communication must be hidden from the user.

Table 1 shows that the main thrust of the design specifications was to guarantee that HERMES be easy to use. Items 1, 2 and 5 address this directly: The tool must be able to be used by any researcher, regardless of the language in which the application is written or the researchers programming proficiency. Items 3 and 4 also address this issue but with the emphasis on immediate usefulness. The minimum performance penalty allows researchers to gain the advantages of on line data acquisition without adversely affecting the code's performance tuning. The distributed and heterogeneous constraint provides that any researcher, regardless of geographical relationship to the machine running the application or the locally accessible workstation can use and benefit from the tool's functionality. Each of these broad design specifications translates into a concrete lineage of implementation details discussed in the next section.

3. Implementations Details

The main features of the prototype tool were implemented during an extended research sabbatical at SDSC. Additional features and enhancements are an area of ongoing research.

3.1 Communications Mechanism

The design constraints discussed above imply several requirements for the communications mechanism that are critical to the success and usefulness of the system. The communications layer must support rapid communications of large (megabyte and above) datasets with minimal central processor unit (CPU)and memory overhead, and it must be portable across a wide range of architectures. A communication mechanism that provides for the translation of data transparently offers additional advantages. The ability to communicate in blocking or non-blocking modes is also desirable.

Several Inter Processor Communication (IPC) mechanisms were examined including Sun's Remote Procedure Call (RPC) mechanism, Berkeley Internet domain sockets, and NCSA's

Data Transfer Mechanism (DTM). The v2.3R5 release of the latter, developed by Terstriep and Weber [**5**], was selected for the communications layer of the prototype HERMES system. DTM is a simple message-passing facility with strong support for heterogeneous applications, large messages, and run-time creation and modification of connections. DTM connections are based on a reliable communications service such as TCP/IP and have been implemented on UNIX machines using the Berkeley sockets library. DTM messages are self-describing and typed according to predefined message classes. DTM provides a standard DTM message class object from which other application-specific classes are derived.

Applications are built to recognize and process messages based on message class. One "standard" message class is the DTM Scientific Data Set (SDS) class. This class was designed for the communication of multidimensional arrays of data. The HERMES tool messages are derived from and compatible with this class. This allows applications designed to receive SDS class messages to receive messages produced by HERMES.

3.2 Tool Organization

The HERMES prototype tool comprises three logical components:

- The simulation interface
- The client interface
- The communications daemon

The tool is based on a publish and subscribe paradigm. In this paradigm, the simulation makes known at initialization time the names and descriptions of all the datasets it wishes to make available for acquisition. This information is stored in the communications daemon, which also processes all client requests. Clients connect directly with the communications daemon. At client connection, the daemon makes available the list of published datasets. Clients then subscribe to receive any of the published datasets. Once this process is complete, the data are forwarded automatically from the simulation to all subscribing clients whenever it is available. Clients may dynamically change their subscribed datasets and the frequency at which the datasets are received. They may also disconnect from and reconnect to the communications daemon as often as they like. By maintaining an internal table of subscribed datasets and by only sending those datasets that are needed, the overhead incurred by the simulation is kept to a minimum. In addition, when no datasets are subscribed, the overhead for having the tool included in an application becomes negligible.

The three components of the tool may be distributed in any fashion across a TCP/IP based network such as the Internet. More specifically, it must be possible to create Internet domain Berkeley socket connections between all the machines used as part of the tool system. The dependence of the tool on Berkeley sockets is imposed by DTM v2.3R5 and may be eliminated in future releases of DTM.

3.3 Interface Specifics

The HERMES prototype tool supports two-dimensional datasets of any size and of type INTEGER, FLOAT, or DOUBLE. One simulation, for example, may connect to a communications daemon when multiple clients are connected. The number of allowed published datasets and connected clients is determined when the tool is compiled. These values are all set to eight in the prototype tool. The tool is written in ANSI style C and requires an ANSI compatible C compiler for compilation. All communications are done using DTM's non-handshaking "asynchronous" protocol. Although labeled asynchronous by DTM, the communications are done across reliable Berkeley sockets connections.

The simulation interface may be called from either Fortran or C language codes and consists of four routines. These routines register datasets for publication, initialize the connections, send datasets, and close connections. The details of the communication mechanism, the connection creation, and the transmittal of subscribed datasets are completely hidden from the user. Pseudo-code showing the use of the simulation interface is shown in Table 2.

Table 2: Simulation Pseudo-code

```
simulation initialization.
dataset_id[1] = register-dataset("dataset-1", size, data_type, time-stamp, etc...)
dataset_id[2] = register-dataset("dataset-2", size, data_type, time-stamp, etc...)
dataset_id[3] = register-dataset("dataset-3", size, data_type, time-stamp, etc...)
initialize-connections("daemon-host-name", daemon port number)
do while timestep_counter less than max_timestep
    calculate dataset-1
    send-dataset(dataset_id[1], timestep_counter)
    calculate dataset-2
    send-dataset(dataset_id[2], timestep_counter)
    calculate dataset-3
    send-dataset(dataset_id[3], timestep_counter)
enddo
close-connection()
simulation clean-up
exit
```

Table 2 shows the ease with which the tool may be integrated into an existing piece of simulation code. Notice that each dataset is registered once during the initialization phase of the simulation. This registration process generates a table of available datasets and returns a unique identifier to the simulation which can then be used to refer to a dataset unambiguously. The data are published by imbedding the send_dataset calls in the simulation code. The send-dataset routine is called every time a dataset is calculated. The application programmer need never worry about whether a particular dataset is subscribed to by any client. Subscribed datasets will be sent automatically to the appropriate clients by the send_dataset routine. Datasets that are not subscribed are not sent, and the send-dataset routine returns quickly, incurring minimum overhead.

The client interface provides a set of routines allowing researchers to build custom data acquisition clients. These clients can perform interactive visualization, analysis, automated storage, or a number of other functions. Client interface routines are C-callable only and provide the ability to create connections, subscribe to datasets, get dataset descriptions, read datasets, request a change in parameters, send parameters, and close connections to the daemon. If the application uses different array ordering than the client (Fortran vs. C), the researcher must ensure that the data are interpreted in the correct manner, although HERMES does provide information about the array ordering of datasets. Users may adopt one of the existing sample clients included as part of the prototype system, or they may build custom clients using the HERMES client interface library.

The demon is a user-executable process that may be started on any machine. The daemon:

- Handles client connection requests
- Manages multiple clients
- Accepts dataset publishing information from the simulation
- Forwards datasets to subscribing clients

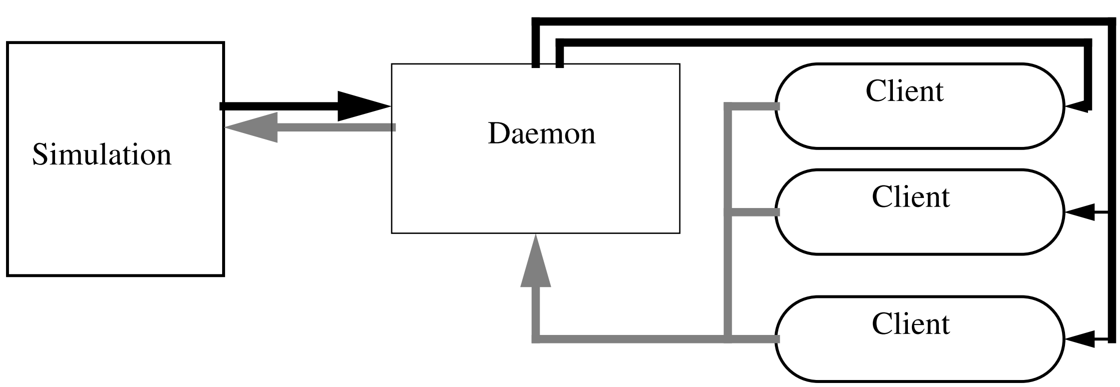
The communications daemon is crucial in providing high interactivity for clients and in minimizing the overhead caused by HERMES on the application. The daemon is designed to be started once and to handle multiple connect and disconnect requests from both clients and simulations correctly. Simulations connect to a single user-specified port while clients connect to a second port. The port numbers used for these two operation are determined at daemon start-up by the user. In the prototype system, data sent by the application is routed through the daemon on its way to subscribing clients. Future versions of the tool will remove this limitation allowing the simulation to send directly to all subscribing clients. The daemon can produce a diagnostic logfile, which may be useful in tracking down system errors.

3.4 System Schematic

A schematic for the prototype tool is shown in Figure 1. Each box represents a logical process that may be executing anywhere on the Internet or other TCP/IP-supporting network. Connections between components are denoted by arrows. The line style denotes the type of message flowing across a connection. Dashed lines indicate control messages, while solid black lines indicate data connections.

All connections are uni-directional. This is a consequence of using DTM for the communications protocol. All clients send control messages to a single input port on the daemon. Data are distributed from the daemon via multiple output ports. The simulation maintains only two connections, one to send data and control messages to the daemon, and one to receive control messages. The daemon is responsible for managing all control messages and for routing subscribed datasets to requesting clients. In this model, clients never send messages directly to the simulation. During initialization, the simulation sends all relevant information to the daemon. From that point on, the daemon is able to interact with clients on the simulation's behalf. The simulation need only periodically check for changes in the set of subscribed datasets, but it does not have to consider the details of client communication. This configuration minimizes the impact of the tool on the simulation while maintaining a high degree of interactivity for the user-operated clients.

Figure 1: System Schematic



4. Case Study

The prototype tool has been tested using the UCLA atmosphere-ocean general circulation model distributed across computers participating in the CASA Gigabit Testbed. In this test case, an idealized version of the atmospheric component of the model executed on SDSC's CRAY Y-MP 8/864 supercomputer while the ocean component ran on an Intel iPSC/860 scalable parallel supercomputer. A special host program was run on a Sun-4 system to provide a communications bridge between the two models. Each of the datasets published by the host program was a 102x102-element two-dimensional array of single precision 32-bit floating point numbers. The daemon executed on the same machine as the host program, while the visualization clients ran on SGI IRIS 4D/440 VGX systems. Network connections between the machines consisted of 10 Mbit / second local Ethernet. Timings were made by bracketing sections of code in the host program with calls to the etime(3F) system call. This routine returns the elapsed execution time broken down into cumulative, system, and user components.

Three timing cases were studied. The first was of the coupled ocean model running with all tool calls commented out. This run was used to establish a baseline for comparing the other timing runs.

The second test was designed to assess the penalty for inclusion of the tool and to test the effectiveness of the implementation in imposing minimum performance penalty on the simulation. In this case, the tool was enabled and all six datasets were published and daemon-simulation connections established, but no client subscribed to any data.

The third test involved a single client connecting to the daemon and subscribing to receive a single dataset. This test measures the efficiency of the data communications. The results for the three tests are presented in Table 3.

Table 3: Result of Three Test Cases

	No Tool	No Subscribed Data	Single Subscribed Dataset
Total execution time	6.12 seconds	6.06 seconds	7.06 seconds
Tool initialization	N/A	.06 seconds	.11 seconds
Avg. elapsed time in send-dataset routine	N/A	0 seconds	0.06 seconds

All times reported in elapsed CPU seconds.

The results shown in Table 3 demonstrate the minimal effects of using the tool on an existing Fortran application. Data were transferred a total of 10 times during each test run. The total time spent sending datasets to subscribing clients in the single subscribed dataset test corresponds to less than 10% of the total execution time. The total overhead from all sources is 15% for the sing subscribed dataset case. The variance in the total execution time for the three cases is caused by the varying load on the heavily used Sun-4 used to run both the model host code and the tool communications daemon. The number of bytes transferred with each call to the send-dataset routine is 41,616 Bytes. This implies an effective data-transfer rate of 5.3 Mbits/ second. Further study with more complex applications and larger data transfers is warranted, but these result are encouraging.

5. Future directions

The HERMES tool discussed here is a working prototype for the application that is being developed. New features in development or planned for future releases are shown in Table 4.

Table 4: Future Features and Enhancements

*	Support for interactive simulation "steering".
*	Support for one and three dimensional datasets.
*	Support for multiple servers communicating with a single daemon
*	Support for receiving datasets directly from distributed / parallel simulations
*	Support for compression of data prior to transfer
*	Support for direct simulation to client routing of data, removing the daemon from the data delivery route
*	Development of additional sample clients for environments such as SGI Explorer, Khoros, and Data Explorer
*	Support for general structured and unstructured data types
*	Elimination of the dependency on a single IPC mechanism

6. Summary

The prototype HERMES tool discussed in this paper addresses the needs of the computational scientist using large-scale complex applications in a network environment. Using this tool, a researcher is able to monitor the current state of a simulation and take corrective action if necessary or simply collect and archive data as it is produced. The driving design constraints of high-interactivity and minimal overhead have been achieved in the prototype system.

Future releases of the tool will eliminate current restrictions in the system, explore new ways of achieving the design goals, and introduce new features. One of the earliest new features to be added is a mechanism to steer a simulation interactively. With this enhancement, a researcher will not only gain significant control over a model, but will also be able to interactively experiment with the impact of changing parameter values on the behavior of the model.

7. Acknowledgments

This work was supported in part by the National Science Foundation and the Advanced Research Projects Agency under Cooperative Agreement NCR-8919038 with the Corporation for National Research Initiatives and by National Science Foundation Cooperative Agreement ASC-8902825.

8. References

[**1**] A. Arakawa and V.R. Lamb, Computational design of the basic dynamical processes of the UCLA general circulation model, *Methods in Computational Physics*, 17, pp 173-265.

[**2**] K. Bryan, A numerical method for the study of the circulation of the world ocean, *J. Comp. Phys*., 4 (1969), pp 347-376.

[**3**] C.D. Scarbnick, C.R. Mechoso, C.C. Ma, J.A. Spahr, J.D. Farrara, R.W. Moore, G.A. Hanyzewski, An update on the CASA gigabit testbed network, Proceedings of the sixth SIAM conference on parallel processing for scientific computing, 2, pp 878-882.

[**4**] C.R. Mechoso, C.C. Ma, J.D. Farrara, J.A. Spahr, and R.W. Moore, Parallelization and distribution of a coupled atmospheric-ocean general circulation model, *Mon. Wea. Rev*., 121, pp 2062-2076.

[**5**] J. Terstriep and D. Weber, NCSA DTM programming manual, NCSA-Blanca Technical Report.