NASA Technical Memorandum 105838

# The Engine Design Engine. A Clustered Computer Platform for the Aerodynamic Inverse Design and Analysis of a Full Engine

J. Sanz, K. Pischel, and D. Hubler
*Lewis Research Center*
*Cleveland, Ohio*

Prepared for the
Cray Users Group, "Grand Challenges"
sponsored by the Naval Research Laboratory
Washington, D.C., September 20–22, 1992

NASA

# The Engine Design Engine.

## A Clustered Computer Platform for the Aerodynamic Inverse Design and Analysis of a Full Engine.

**J. Sanz, K. Pischel and D. Hubler.**

**NASA Lewis Research Center.**

### Introduction

The use of parallel processing with advanced supercomputers is permitting the attempt of computational tasks that very few years ago would have seemed unreasonable. Present parallel computing capabilities make perfectly reachable the aero design of a full, multi-staged, turbo engine by direct simulation of the Euler equations rather than depending on the expensive gathering of experimental data for closure of a modeling 'ansatz' of these equations.

Concurrent with the development of supercomputers, powerful workstations have made their way onto researchers' desks. It is not uncommon, though, that these workstations, while having a very heavy interactive use during regular hours, may experiment a much lower use at other times. With workstations of 25 MFLOPs per processor it seems reasonable to develop applications that can harness this tremendous, and frequently hidden, computational power.

This paper describes an experiment in parallel computing performed in collaboration with the Computer Services Division and the Aeropropulsion Analysis Office of the NASA Lewis Research Center. A cluster of seven IBM RS/6000–550 powerstations and two CRAY-YMP main frames has been used for this experiment. The machines have been clustered using the Lewis Ethernet Network, with the TCP/IP protocol. Three hardware configurations have been tested for parallel processing: An RS/6000 cluster, a heterogenous cluster of RS/6000 plus two CRAY-YMP machines, and a stand alone CRAY-YMP using all eight processors.

A version of the code Aida.fe, Aerodynamic Inverse Design and Analysis for a Full Engine, has been parallelized using the Parallel Virtual Ma-

chine (PVM) programing language. The master/slave paradigm suggested by the PVM developers has been followed in the parallelization of the code. In this environment the master runs on the host workstation and the slaves run on each one of the nodes attached to the cluster. The host performs all the I/O operations and file manipulation with little CPU use but with an extensive use of memory. On the other hand, the nodes execute most of the computational work with no I/O needed. The executable is actually the only file, permanent or temporary, used by the node. This is a very desirable feature to have in a multi-user environment, both in a distributed cluster of workstations or when mainframes are attached as nodes. A node retrieves, or sends, through PVM message passing calls, all the necessary information from, or to, the host or the other nodes, while the host does all the pre- and post-processing and storage. In the following sections we describe the three different configurations explored.

## The RS/6000 Cluster

The first test on the clustered system was performed during the month of April 1992. Seven IBM RS/6000–550 powerstations located in different buildings were linked on the Ethernet network. No subnetting was established, so the experiments ran in a shared mode with the regular network traffic at the Lab.

One workstation is used as both host and node, while the remaining six are used as nodes. In the approach taken in the parallelization of the code each node handles the calculation for one blade row or flow passage. It could then be classified as a macro-tasking parallelization. Figure No. 1 represents a schematic of the clustered configuration. The host sends to each node the particular information needed, and broadcasts to all of them the common information they have to share. The nodes return to the host some computed values and, after further manipulation by the host, the nodes are given the order to begin the main execution of the program. In this first test case, the nodes perform the main calculation with no communication between themselves.

Table No. 1 shows the CPU time used by each node on the cluster, compared to the CPU time on one YMP processor of a sequential run for

the same test case. The results show that, with a coarse grid run that achieves a rate of 54 MFLOPs on the YMP, four RS/6000 processors are equivalent, in CPU time, to one YMP processor. Subsequent examples will show the corresponding rates when, with finer grids, the YMP delivers rates of 84 and 112 MFLOPs . The results shown in Table No. 1 were obtained after some vectorization improvements were made. The original run, in April, showed a 3.2 equivalence rate between RS/6000 processors and one YMP processor. Real time comparison was also made by running the cluster on one of the host windows while on another window the same case would run, sequentially, on one YMP processor. Consistently, the cluster would complete execution at the time the YMP would finish the fourth blade row. Because of the short CPU times of the tests, time sharing on the YMP gives high priority to the job, making the real time comparison very close to the CPU comparison shown before.

Since the time at which this test was performed, the cluster has worked consistently well on a sustained mode of operation, including the case in which nodes talk to each other, producing a substantial amount of interprocessor communication. It is obvious, though, that the advent of more users performing parallel work on a distributed environment will require the establishment of separated subnets for those machines forming different clusters. This will restrict the trafic within each subnet.

| node | arch | CPU sec | node | arch | CPU sec |
|---|---|---|---|---|---|
| 1 | RS/6000 | 22.34 | 1 | Cray | 5.68 |
| 2 | RS/6000 | 22.00 | 1 | Cray | 11.04 |
| 3 | RS/6000 | 22.57 | 1 | Cray | 16.52 |
| 4 | RS/6000 | 22.05 | 1 | Cray | 21.88 |
| 5 | RS/6000 | 23.13 | 1 | Cray | 27.30 |
| 6 | RS/6000 | 22.39 | 1 | Cray | 32.65 |
| 7 | RS/6000 | 23.14 | 1 | Cray | 38.15 |
| RISC/YMP processors = 4 | | | YMP MFLOPs = 54. | | |

Table 1. The RS/6000 versus one YMP processor.

## The Heterogeneous Cluster

In the design process established by the code methodology the CPU time needed by different blade passages may vary largely. It seems then reasonable to have the capability of attaching to the cluster, as new nodes, one or more processors of a CRAY-YMP to handle the blade rows that need a more intensive CPU. The host RS/6000 handles perfectly this heterogeneous system in which one or more nodes are processors of a CRAY-YMP. All PVM requires is to recompile the node executable on the new CRAY architecture. For this test, we linked to the cluster the CRAY-YMP at the Lewis Research Center and the NAS CRAY-YMP , located at the Ames Research Lab.

In this new environment, we run, as in case No. 1, the same test from the previous section. Table No. 2 shows the CPU time per node on an RS/6000 node compared to the CPU time on one YMP processor. It shows again that four RS/6000 processors perform at the same rate as one YMP processor. Cases No. 2 and 3, on the same table, show the comparison between the two architectures when the code runs on finer grids with an 85 and 112 MFLOPs count on the YMP. In these cases, six and eight RS/6000 processors, respectively, are needed to produce the same work as one YMP processor.

The host can request a specific architecture, for a given virtual node, at the initialing call. When there are more virtual nodes initialized of a given architecture than there are machines attached to the cluster with the requested architecture, PVM will start as many new processes as required, and will split them within the machines available of the given type. A practical case is the one in which just one of the two YMPs is attached to the cluster. In this case, as many processes are started on this machine as virtual CRAY nodes are required. It is worthwhile noting that when using the heterogenous cluster, and if internode communication is required, the nodes that so require should run on the same architecture in order to have a balanced execution.

Although other architectures could have been tried, the high CPU speed requirements make the combination of RS/6000 and YMPs a very formidable hardware platform for the task at hand.

| case | arch | CPU sec/ node | RISC/ YMP | arch | CPU sec/ node | MFLOPs |
|---|---|---|---|---|---|---|
| 1 | RS/6000 | 23. | 4 | Cray | 5.53 | 54. |
| 2 | RS/6000 | 132.35 | 6 | Cray | 21.91 | 85. |
| 3 | RS/6000 | 994.94 | 8 | Cray | 122.60 | 112. |

Table 2. The heterogenous cluster.

## The Stand alone YMP

The last configuration studied comprises the eight processors CRAY-YMP at the Lewis Research Center acting as a stand alone platform with the host and nodes running on the same machine. The purpose of this test is to have an estimate of the efficiency of PVM, for our particular application, in using the resources available, and to assess the overhead imposed by the interprocessor communication.

Ten test cases were run on this machine on dedicated time. The results are tabulated on Table No. 3. Cases 1, 3, 5, 7 and 9 consist of tests without internode communication, while their counterparts 2, 4, 6, 8 and 10 run the same cases with large internode communication. In cases 1 and 2 only four processors are called, showing in case 1 a 49.2 percent use of the total machine or 98.4 percent of the four processors requested. Case 2 shows that a 20 percent of the machine usage is spent in interprocessor communication.

Cases 3 to 10 required the use of all eight processors. Case 3, with no internode communication uses 98 percent of the full machine. Case 4 adds a 30 percent use for internode communication. Because it is a coarse grid run, the rate of internode communication to CPU usage is large. Cases 5 and 6 are equivalent to cases 3 and 4, but executed on a finer grid. The rate of internode communication to CPU usage is then lower than in case 4, showing a good 4 percent use of the machine for node communication.

Cases 7 and 8 are similar to cases 5 and 6 but executed with a more sustained calculation. Finally, cases 9 and 10 represent a third grid

refinement calculation with a 112 MFLOPs count.

Overall the test seems to indicate that the present application makes an efficient use of the available processors on a super-computer, and this is more evident when finer grids and sustained calculations are performed.

| case | #nodes | arch | CPU sec /node | Wall Clock | %8-CPU /node | %Total CPU | MFLOPs |
|---|---|---|---|---|---|---|---|
| 1 | 4 | Cray | 31.00 | 31.48 | 12.30 | 49.20 | 54. |
| 2 | 4 | Cray | 46.16 | 71.49 | 8.00 | 40.00 | 54. |
| 3 | 8 | Cray | 51.49 | 52.47 | 12.25 | 98.00 | 54. |
| 4 | 8 | Cray | 53.05 | 85.18 | 8.50 | 68.00 | 54. |
| 5 | 8 | Cray | 62.47 | 65.65 | 11.875 | 95.00 | 85. |
| 6 | 8 | Cray | 62.83 | 68.74 | 11.41 | 91.30 | 85. |
| 7 | 8 | Cray | 207.37 | 226.06 | 11.3 | 90.40 | 85. |
| 8 | 8 | Cray | 209.04 | 257.93 | 9.92 | 79.4 | 85. |
| 9 | 8 | Cray | 119.13 | 124.16 | 11.95 | 95.6 | 112. |
| 10 | 8 | Cray | 119.88 | 129.08 | 11.64 | 93.12 | 112. |

Table 3. Eight processors on a YMP.

## Concluding Remarks

An application for parallel computation on a combined cluster of powerful workstations and super-computers has been developed. PVM, Parallel Virtual Machine, is used as message passage language on a macro-tasking parallelization of the Aerodynamic Inverse Design and Analysis for a Full Engine computer Code. The heterogenous nature of the cluster is perfectly handled by the controlling host machine. Communication is established via Ethernet with the TCP/IP protocol over an open network. A reasonable

overhead is imposed for internode communication, rendering an efficient utilization of the engaged processors. Perhaps one of the most interesting features of the system is its versatile nature, that permits the usage of the computational resources available that are experiencing less use at a given point in time.

## Acknowledgment

The authors would like to thank Tony Hackenberg, NASA Lewis Computer Service Division, for his superb assistance on the use of the CRAY-YMP on dedicated time.

## ENGINE DESIGN ENGINE