NASA/CR—2000-209939

Dynamic Load-Balancing for Distributed Heterogeneous Computing of Parallel CFD Problems

A. Ecer, Y.P. Chien, J.D. Chen, T. Boenisch, and H.U. Akay
Purdue School of Engineering and Technology, Indianapolis, Indiana

March 2000

The NASA STI Program Office . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the Lead Center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA's counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized data bases, organizing and publishing research results . . . even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA Access Help Desk at (301) 621-0134

- Telephone the NASA Access Help Desk at (301) 621-0390

- Write to:
  NASA Access Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076

NASA/CR—2000-209939

Dynamic Load-Balancing for Distributed Heterogeneous Computing of Parallel CFD Problems

A. Ecer, Y.P. Chien, J.D. Chen, T. Boenisch, and H.U. Akay
Purdue School of Engineering and Technology, Indianapolis, Indiana

Prepared for the
Computational Aerosciences Workshop
sponsored by the High Performance Computing and Communications Program
Moffett Field, California, February 15–17, 2000

Prepared under Contract NAS3–2260

National Aeronautics and
Space Administration

Glenn Research Center

March 2000

Acknowledgments

The authors would like to express their appreciation to management of the High Performance Computing and Communications Program and to the NASA R&T Base Program for supporting NPSS.

This report contains preliminary findings, subject to revision as analysis proceeds.

Available from

NASA Center for Aerospace Information
7121 Standard Drive
Hanover, MD 21076
Price Code: A03

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22100
Price Code: A03

DYNAMIC LOAD-BALANCING FOR DISTRIBUTED HETEROGENEOUS COMPUTING OF PARALLEL CFD PROBLEMS

A. Ecer, Y. P. Chien, J. D. Chen, T. Boenisch and H.U. Akay
Purdue School of Engineering and Technology
723 W. Michigan St.
Indianapolis, Indiana 46202
(317)-274-9712
ecer@engr.iupui.edu

1. INTRODUCTION

Parallel processing is widely used for solving computation intensive problems. Parallel algorithms have been implemented on parallel supercomputers, networked workstations, and combinations of workstations and supercomputers. One common approach in solving large CFD problems is to utilize block-structured solution schemes. The original domain is divided into a series of blocks. The parallel code then distributes the blocks among a set of networked computers. While each computer processor is responsible for solving one block of data it has to communicate with others during the execution of the parallel code. One can define the parallel code in terms of a series of block and interface solvers (Akay, 1993). While the block solver is for computing the solution for a block, the interface solver is for exchanging information between block boundaries. The execution time of each process is affected by several time-varying factors, e.g., the load of computers, the load of the network, the solution scheme used for solving each block, and sizes of blocks. Therefore, some processes may finish much earlier than other processes and wait for information from other processes. Such waiting significantly increases the program execution time and decreases the efficiency of the system. Dynamic load balancing (DLB) is a tool to properly distribute the processes among computers to ensure the communication time and the waiting time are minimized (Chien, 1994). To increase the efficiency and speed of parallel computation, the computation load should be distributed to computers in such a way that the elapsed execution time of the slowest computer is minimized. DLB is essential for efficient use of complex and dynamic parallel and distributed computing resources.

Parallel algorithms allow the utilization of multiple computers for a single job. The speed of a parallel algorithm is measured in terms of number of processors which can be accessed without loosing parallel efficiency. The main objective of the present effort is to run parallel codes on all available computer resources at any given time. This includes clusters of parallel supercomputers and networked workstations communicating through complex networks. The specific problem is to distribute the parallel job to available processors in a most efficient way. It is assumed that available computers are heterogeneous: operating under UNIX and NT operating systems. There are several clusters or computers connected to each other with different

networks and scheduled by different system managers: LSF, PBS, Load Leveler. Multi-user environment is considered: there are several parallel jobs running on the system. When the computers by different owners need to be accessed and many parallel jobs need to be executed, the efficient utilization of resources becomes a difficult task. The DLB scheme is developed for improving the efficiency of parallel computing in such an environment. More important, the load-balancing task is defined as a responsibility of the owner of the application program rather than the system manager. As mentioned above, block-structured solution schemes are supported. It is assumed that the problem is divided into a number of blocks which is greater than the number of available processors. The division of the grid is performed only once. Either greedy or genetic algorithms are utilized for calculating the distribution of the number of blocks among available processors.

Previously reported load balancing techniques commonly assumes that there is only one parallel job running on a given set of computers. This assumption is valid when the same owner owns all the computers for parallel processing and the owner can dedicate a set of computers for a particular parallel process. When many owners own the computers and many parallel jobs need to be executed on these computers, the reservation of dedicated time for a parallel process becomes difficult. In a multi-user environment, multiple parallel jobs may be executed concurrently on the same set of computers. One unsolved problem in parallel processing is how to distribute multiple mutually dependent parallel jobs among computing resources to achieve optimal computation speed for each parallel job. When the mutually dependent load is not balanced, computers that finish the computation early need to wait the other computers periodically in order to collect essential information to proceed further, thus loosing their share of computing resources and not achieving the best computation speed. When load balancers for single parallel jobs are used together, conflicts of interests makes them to interfere with each other. In this paper, we introduce a new load balancing method which addresses multiple parallel jobs. The objective is to ensure that every parallel job, as well as the entire system is load balanced.

2. BASIC DLB PROCEDURE

The basic assumptions of DLB are as follows:

- There are large numbers of computers available in different locations, which are managed by different owners.
- At the initiation of the run a set of available computers is defined by the user. The user can access all or any subset of these computers.
- Each of the multi-user computers is operating under Unix or Windows NT.
- The parallel application software is running MPI or PVM as shown in figure 1.
- It is assumed that the job will take several hours, possibly all evening.
- A load balancing cycle is defined, e.g., 20-40 minutes, which is larger than the time required to move the jobs around.

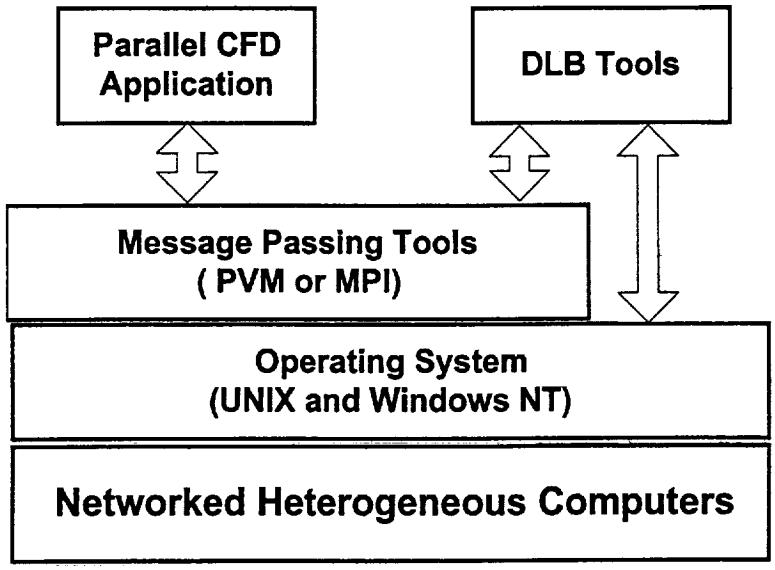

Figure 1. DLB Environment

The dynamic load balancing capability includes the following:

- Software assigned to each computer to measure computer and network speed.
- A load-balancing tool assigned to each parallel job to optimize the load distribution.

A master coordinator is not needed while each cluster or computer may be running under a different job scheduler. The basic procedure, which is shown in figure 2, can be summarized as follows:

Measure and Estimate Computation Cost Parameters During a DLB Cycle:

1) The speed of the computer,
2) The computation cost of each process on a given computer; and,
3) The total number of active processes on that computer.

Measure and Estimate Communication Cost Parameters During a DLB Cycle:

1) The speed of the network; and,
2) The communication cost of sending each message on a given network.

The unit of cost is defined as elapsed time. A cost function is defined as the elapsed time to complete the slowest process. This cost functions is then minimized at the end of each cycle to calculate the optimal load distribution for a given job. The jobs are re-distributed and the process is repeated. The procedure allows for abandoning busy processors and utilizing additional processors when they are available. Different optimization algorithms can be utilized to minimize the cost function: Greedy, Genetic and mixed Genetic-Greedy algorithms have been implemented.

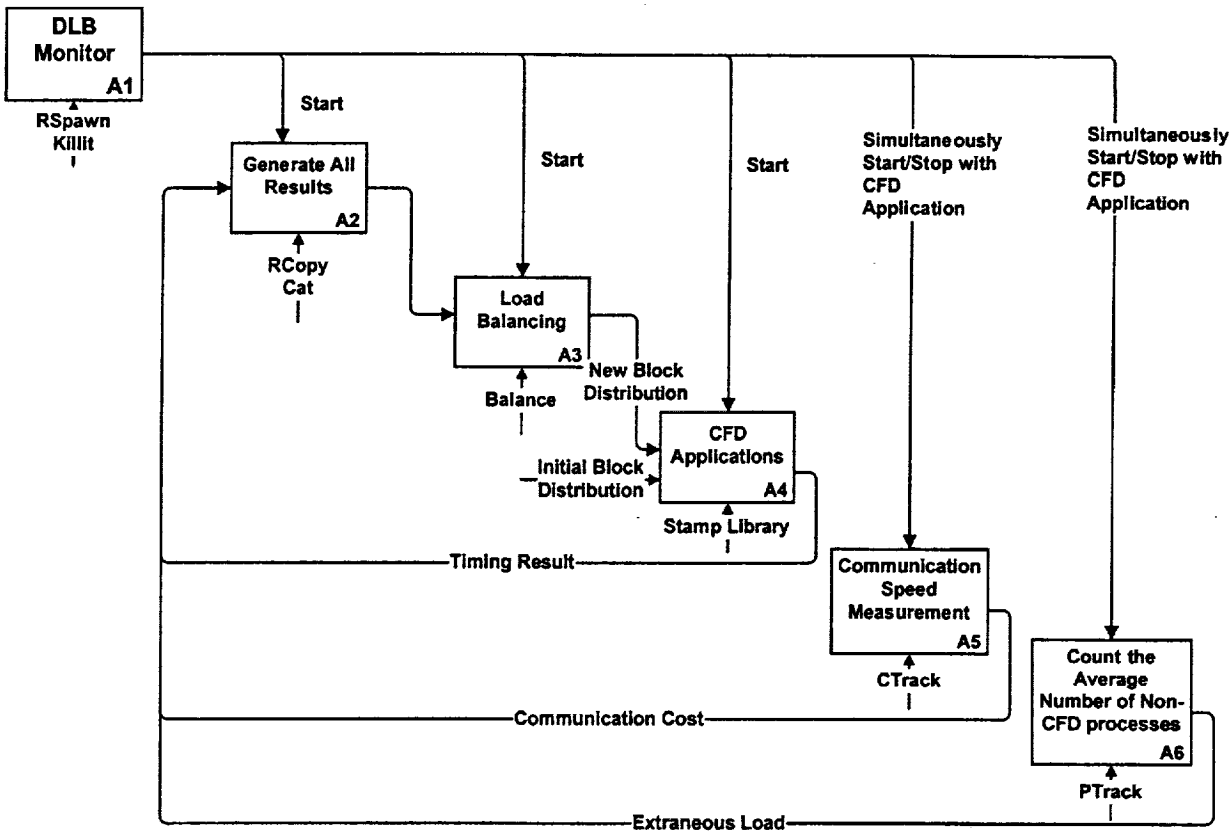

Figure 2. DLB Procedure

3. DLB TOOLS

The DLB package provides several tools.

Stamp Library: The stamp library is a collection of functions which can be called by C or FORTRAN programs. They can be embedded into CFD programs, and they gather the timing information related to both the CFD program and the computer network.

Ctrack: In order to estimate the elapsed execution time for the CFD blocks, information about the communication speed between all computers is needed. The CTrack-Communication Tracker program supports communication speed measurement.

Ptrack: Both UNIX and NT are multi-user and multi-tasking Operating Systems (OS). CPU time is shared by all concurrently running processes. The number of parallel CFD processes running on each machine is defined and Ptrack (process tracker program) finds the average number of processes belonging to other users, (extraneous load).

Balance: This is designed for load balancing. Balance predicts the computation and communication costs of any given load distribution. It finds a time optimal load distribution for the next execution cycle.

DLB Monitor: This tool initializes the computation and communication cost model while running the application program simultaneously with Ptrack and Ctrack. It gathers the time stamp and Ptrack/Ctrack results from all parallel computers for load balancing.

RCopy & Rspawn: Rcopy copies files from/to remote NT and Unix based computers using message-passing libraries which are necessary for gathering data for load balancing. Rspawn executes system commands (or applications) on remote NT and Unix based computers.

Others: Several UNIX tools useful for DLB, for example, ps, killit and cat were developed for Windows NT.

4. DLB EXAMPLES

Two test cases are presented to illustrate the basic concept of DLB. Both cases include 64 blocks.

In the first case the following processors were available:
6 Processors of the IBM RS/6000 Cluster, CFD Lab, IUPUI, Indianapolis (iw1-iw6);
10 Processors of the Windows NT PC-Cluster, CFD Lab, IUPUI, Indianapolis (ip1-ip7, ip9-ip13); and
8 Processors (Thin node 2SC) of the IBM SP, IU, Bloomington, Indiana (b1 - b8).
One processor of the IBM RS/6000 Cluster computing the application is approximately 10% faster than the nodes b1 to b5 in Bloomington, 30 - 40% faster than nodes b6 to b8 in Bloomington and nine times faster than one of the PC's since the application was compiled without optimization. Initial distribution involved 3 blocks on each processor at the PC-Cluster and at the SP in Bloomington, 1 or 2 blocks at the RS/6000s of the CFD Lab as shown in figure 3. The balanced distribution is also shown. Elapsed time for the run was reduced 1379 seconds to 371 seconds.

The second case involved the following processors:
6 Processors of the IBM RS/6000 Cluster, CFD Lab, IUPUI, Indianapolis, Indiana, (iw1-iw6);
8 Processors (Thin node 2SC) of the IBM SP, IU, Bloomington, Indiana (b1 - b8);
8 Processors (Thin node 2) of the IBM SP, RUS, Stuttgart, Germany (s1 - s8); and
10 Processors of the Windows NT PC-Cluster, CFD Lab, IUPUI, Indianapolis, (ip1, ip4-ip7, ip9-ip13).

One processor of the IBM RS/6000 Cluster IUPUI, computing the application is nearly 4 times faster than the SP nodes in Stuttgart. Initial distribution and the distribution after the DLB is shown in figure 4. The elapsed time for this case is reduced from 1271seconds to 345 seconds.

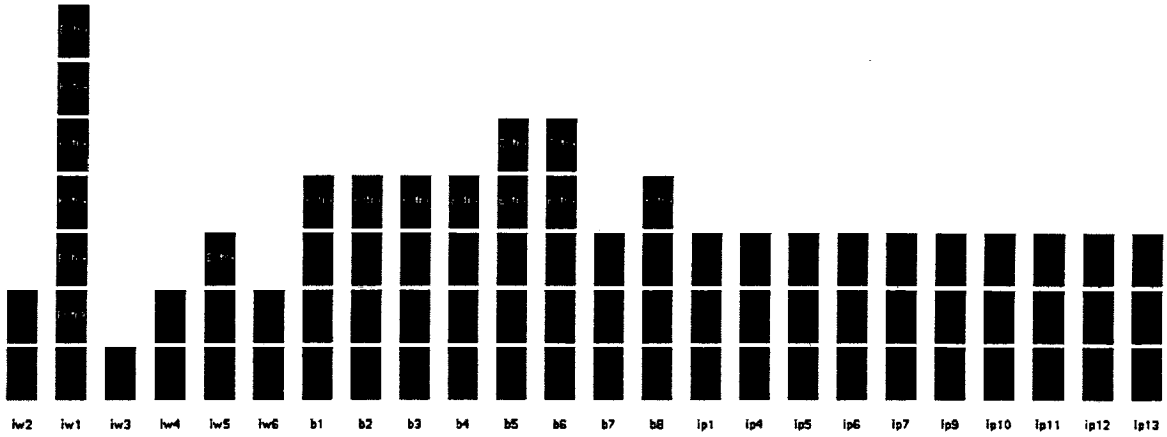

Figure 3a. DLB Test Case 1: Original Distribution

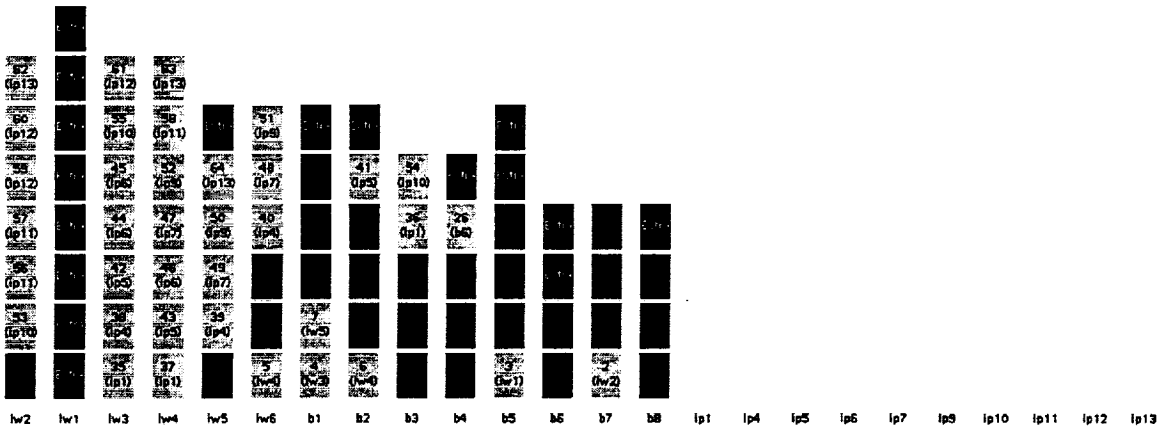

Figure 3b. DLB Test Case 1: Load Balanced Distribution

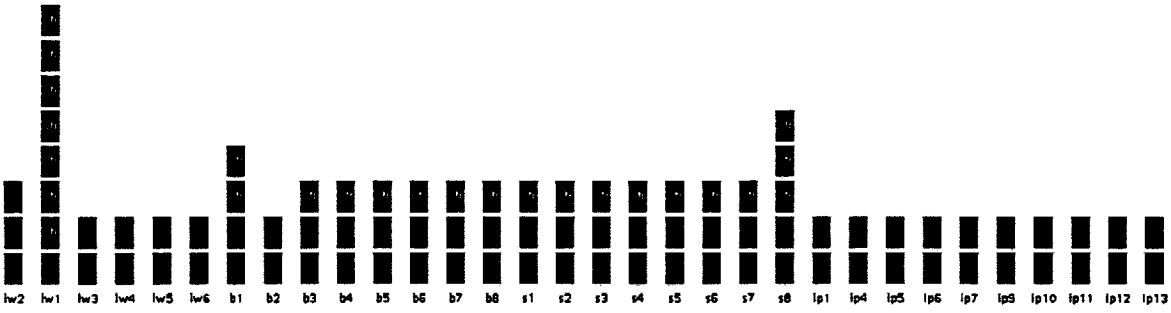

Figure 4. DLB Test Case 2: Original Distribution

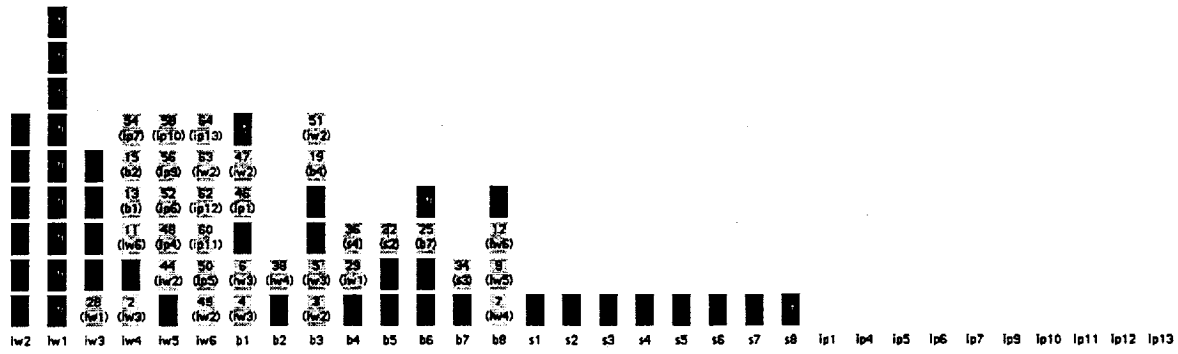

Figure 4b. DLB Test Case 2: Load Balanced Distribution

5. MULTI_USER DLB

5.1. Problem description

The following assumptions are used for the development of load balancing tools for multiple parallel jobs:

1) Every user can use all available hosts (computers).
2) A DLB Monitor is associated with each parallel job. The DLB monitor is responsible for the load balancing of that particular parallel job.
3) A DLB Monitor can request information about computation load and communication speed of any computer.
4) A DLB Monitor only has the detailed knowledge of its own application. It only knows the number of processes generated by the other users on each host (extraneous loads). It does not have the details of the parallelism for these jobs.
5) The DLB Monitor may request information from any number of hosts for the parallel jobs. However, the DLB monitor determines the optimal number of hosts to be used for the parallel job.
6) Each DLB Monitor suggests a new load distribution to the computers for the parallel job.

The task of the load balancer for multiple parallel jobs is to ensure that the DLB monitors do not interfere with each other. In other words, load distribution suggested by a DLB monitor should not affect the balance of other currently executing parallel jobs.

5.2. Dynamic load balancing for multiple parallel jobs

In a multi-user environment, multiple parallel jobs may be executed concurrently on the same set of computers. Therefore, different DLB monitors may have conflicting interests. In order to resolve the conflict, a higher level manager may be needed. In the proposed approach, distributed DLB monitors for each application resolve conflicts without the aid of a higher level manager.

5.2.1 The organization of the software tools

The organization of the software tool, shown in figure 5, is as follows:

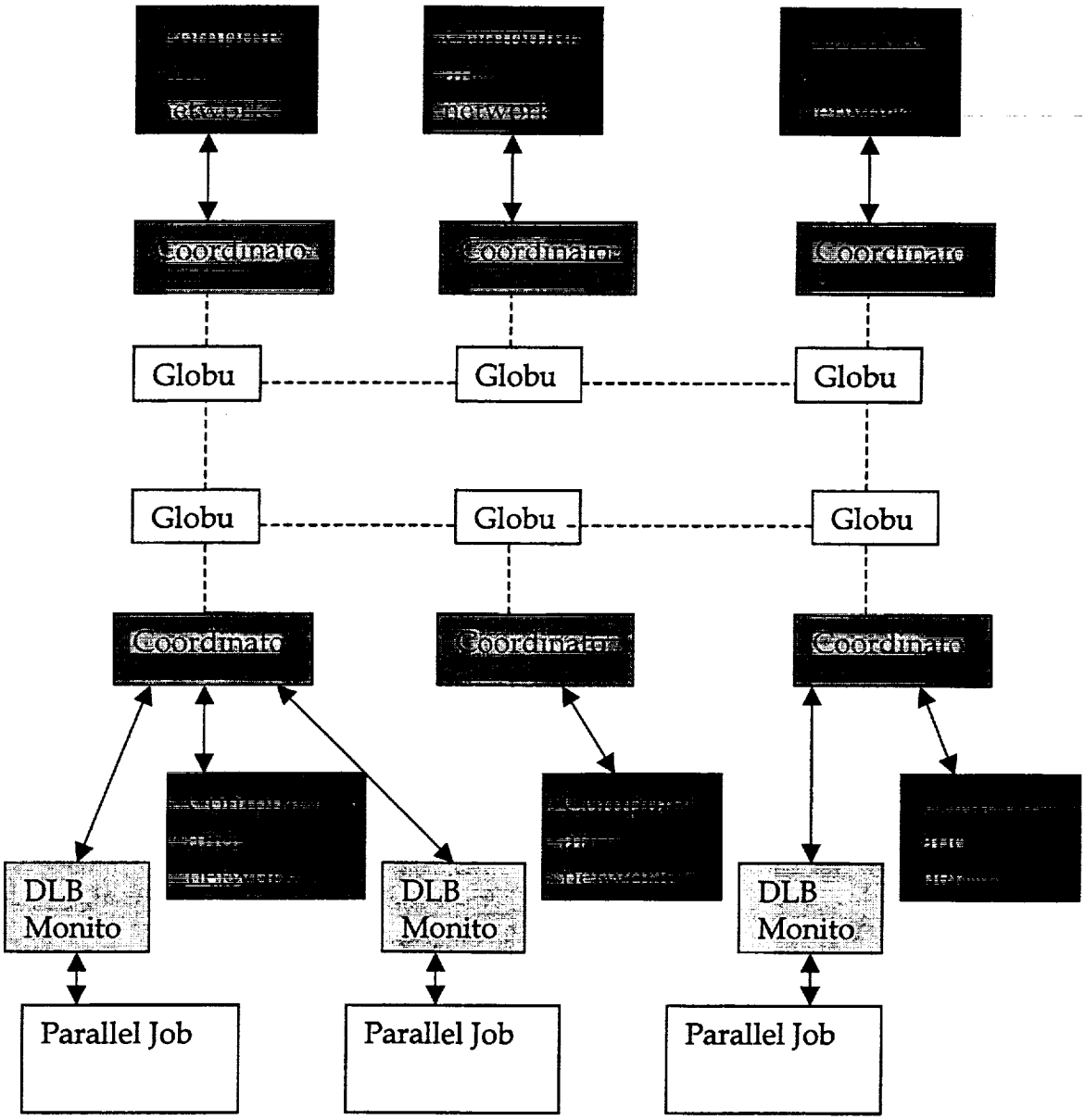

Figure 5. Organization of Software Tools for Load Balancing of Multiple Parallel Jobs

1) A DLB monitor is associated with each parallel job.
2) One coordinator demon is running on every host. Each coordinator can manage multiple DLB monitors. The duty of the coordinator is to support the DLB monitors that run on the same host. The coordinator helps the DLB monitors on the host to send messages to and receive messages from the coordinators on other hosts.
3) One system monitor is running on each computer to gather local computer and network speed information.

4) The coordinator communicates with the system monitor to obtain the local system information.
5) Each DLB monitor requests information of computation load and communication speed of any available hosts through coordinators.
6) The tools are able to support different optimization methods for load balancing.
7) Load balancing is performed in a distributed manner. There is no need for a higher level manager. This requirement is essential since hundreds or more hosts from different organizations may be used for parallel computing by a variety of applications. The breakdown of any host should not affect the load balancing ability of other hosts. As indicated in the figure, access to different hosts can be achieved through Globus.

5.2.2 The load-balancing algorithm

The organization of the software tools supports the efficient exchange of information between DLB monitors. There are several possible approaches to use the shared information for load balancing. In this paper, we discuss the round robin load-balancing algorithm. In round robin load balancing, each parallel job can stop its execution and request load balancing at any time. However, the system only allows one parallel job to perform a load balance at any particular time. Load balancing for multiple parallel jobs is then performed in sequence.

The basic idea for this approach is as follows: It is assumed that the computers are multiple-user and multiple-process computers, for example operating under Windows NT and/or Unix. It is also assumed that the application processes on each computer have the same execution priority. Therefore, the CPU power of a computer is equally divided among all application processes executing on that computer. In a multiple-processing environment, if one process does not fully use the given time slice, then the CPU power will be shared by the other processes on that computer. The responsibility of the load balancing coordinator is to monitor the computer load, the network load, and communicate with all parallel processes executing on the computer. To prevent any conflict of interest, only one DLB monitor is allowed to load balance at each time instance. For this purpose, all the computer hosts participating in the parallel computing form a token ring. A token is passed among the hosts. Only the DLB monitor on the host that has the token is allowed to perform load balancing. If there are more than one DLB monitors on a host, the coordinator allows the DLB monitors to perform load balancing one after another.

6. CONCLUSIONS

This paper studies an optimization algorithm for the load balancing of multiple parallel processes belonging to different users, competing for the same resources. It is assumed that (1) a parallel process does not have detailed knowledge of other parallel processes, (2) each parallel process has its own load-balancer, and (3) each parallel process can share its load distribution on any computer with other parallel processes

that use the same computer. There is a load balancing coordinator on each computer for sharing information between the users. It monitors the computer load, the network load, and communicates with all parallel processes executing on that computer. Under such conditions, a DLB monitor can load balance a specific parallel application without the aid of a higher level manager.

7. ACKNOWLEDGEMENTS

The authors would like to express their appreciation to management of the High Performance Computing and Communications Program and to the NASA R&T Base Program for supporting NPSS.

8. REFERENCES

1.Akay, H.U., Blech, R., Ecer, A., Ercoskun, D., Kemle, B, Quealy, A. and Williams, A., A Database Management System for Parallel Processing of CFD Algorithms, Parallel Computational Fluid Dynamics '92, Ed. By R.B. Pelz, et. al., Elsevier Science Publishers.

2. Chien, Y.P., Ecer, A., Akay, H.U., Carpenter, F. and Blech, R.A., Dynamic Load Balancing on a Network of Workstations for Solving Computational Fluid Dynamics Problems, Computer Methods in Applied Mechanics and Engineering, 119 (1994) 17-33.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 2000	Final Contractor Report

4. TITLE AND SUBTITLE

Dynamic Load-Balancing for Distributed Heterogeneous Computing of Parallel CFD Problems

5. FUNDING NUMBERS

WU–509–10–24–00
NAS3–2260

6. AUTHOR(S)

A. Ecer, Y.P. Chien, J.D. Chen, T. Boenisch, and H.U. Akay

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Purdue School of Engineering and Technology
723 W. Michigan Street
Indianapolis, Indiana 46202

8. PERFORMING ORGANIZATION REPORT NUMBER

E–12187

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
John H. Glenn Research Center at Lewis Field
Cleveland, Ohio 44135–3191

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA CR—2000-209939

11. SUPPLEMENTARY NOTES

Prepared for the Computational Aerosciences Workshop sponsored by the High Performance Computing and Communications Program, Moffett Field, California, February 15–17, 2000. Project Manager, Isaac Lopez, Aeronautics Directorate, organization code 2900, (216) 433–5893.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Categories: 01 and 61 Distribution: Nonstandard

This publication is available from the NASA Center for AeroSpace Information, (301) 621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

The developed methodology is aimed at improving the efficiency of executing block-structured algorithms on parallel, distributed, heterogeneous computers. The basic approach of these algorithms is to divide the flow domain into many subdomains called *blocks*, and solve the governing equations over these blocks. Dynamic load balancing problem is defined as the efficient distribution of the blocks among the available processors over a period of several hours of computations. In environments with computers of different architecture, operating systems, CPU speed, memory size, load, and network speed, balancing the loads and managing the communication between processors becomes crucial. Load balancing software tools for mutually dependent parallel processes have been created to efficiently utilize an advanced computation environment and algorithms. These tools are dynamic in nature because of the changes in the computer environment during execution time. More recently, these tools were extended to a second operating system: NT. In this paper, the problems associated with this application will be discussed. Also, the developed algorithms were combined with the load sharing capability of LSF to efficiently utilize workstation clusters for parallel computing. Finally, results will be presented on running a NASA based code ADPAC to demonstrate the developed tools for dynamic load balancing.

14. SUBJECT TERMS

Parallel CFD; Load balancing; Distributed computing

15. NUMBER OF PAGES

16

16. PRICE CODE

A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	