Load Balancing for Parallel Computing on Distributed Computers

Y.P. Chien, A. Ecer, J.D. Chen, and H.U. Akay
Purdue School of Engineering and Technology
Indiana University-Purdue University at Indianapolis
723 W. Michigan St. Indianapolis, Indiana 46202
U.S.A.
(317)-274-2760

Abstract: Distributed processing can be used for solving large computation intensive problems. A distributed system may include parallel supercomputers, networked workstations and PCs. This paper discusses load balancing of a parallel job in a distributed computation environment. The information necessary for load balancing is studied. The software tools that automatically collect the information and perform load balancing is described. Parallel computational fluid dynamics examples are used to demonstrate the effectiveness of the load balancing method.

Key-Words: distributed computing, dynamic load balancing.

1. Introduction

Distributed computing allows the utilization of multiple computers for a single parallel job. One common approach in solving large parallel computational fluid dynamics (CFD) problems is domain decomposition. In domain decomposition approach, the original domain is divided into a set of sub-domains (also called blocks). The blocks are distributed among a set of networked computers. When each computer process is solving one block of data, the computer processes have to communicate with their neighbors periodically. The parallel program can be divided in terms of a series of block and interface solvers [1]. 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 complete computation earlier than other processes and wait periodically for information from other processes. Such waiting significantly increases the elapsed program execution time and decreases the efficiency of the system.

Dynamic load balancing (DLB) is to properly distribute the computation processes among computers to ensure the communication time and the waiting time are minimized [2,3]. 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.

In this study, the basic assumptions of DLB are as follows: (1) there are large numbers of computers available in different locations. These computers are grouped in many clusters and connected to each other with different networks. (2) The multi-user computers are operating under Unix and/or Windows NT. (3) The parallel application software is running MPI or PVM [4]. (4) The parallel job may take several hours or days to execute. The time saving by load balancing is justified. (5) A load balancing cycle is defined, e.g., half hour or unbalance is detected.

In this paper, we will describe the information necessary for load balancing, the tools needed for obtaining such information, computer and network models for evaluating the quality of load distribution and the optimization methods used for load balancing. Parallel CFD examples are described to demonstrate the effectiveness of the proposed load balancing method.

2. Information Needed for DLB

Many factors affect the load balance a parallel job. Some factors are related to the parallel applications, such as the sizes of CFD blocks, sizes of interfaces, the computation complexity of block and interface solvers, the algorithm used in the CFD program (such as the method for determining the size of time-steps). Some factors are related to the computation environment, such as the computer speeds, the number of non-parallel processes (we call them extraneous load) executed concurrently with parallel processes on each processor, and the communication speed between each pair of computers on the network during the execution of parallel CFD. Knowing all the factors enable us to derive a model to estimate the quality of any given load distribution. By describing the quality of the load distribution as a cost function, load-balancing problem becomes a standard optimization problem. However, the factors related to computational environment are usually not available to the user of the parallel applications. The factors related to the parallel applications are usually not considered by computer system tools.

3 DLB Tools

We have developed a software tool set DLB to support dynamic load balancing. Several tools are provided by DLB.

GPAR: The GPAR is a parallel CFD data management tool. It provides all load-balancing factors that are related to the parallel application.

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. They gather the timing information related to both the CFD program and the computer network. The development of the stamp library is based on the assumption that the parallel CFD program is written in a format that consists of two components, the block solver and the interface solver. The block solver contains all computations for solving differential equations. The interface solver is responsible for inter-processes communication only. When non-consecutive communications are required within each time-step, multiple block solvers and multiple interface solvers are allowed.

Ptrack: Presently DLB can be used on UNIX and Windows NT based systems. Since both UNIX and Windows NT are multi-user and multi-tasking OS, CPU time is shared by all concurrently running processes, Ptrack (process tracker program) finds the average number of processes on each computer. Since the number of parallel CFD processes running on each computer are known, the load that belongs to other users (extraneous load) can be calculated.

Ctrack: In order to estimate the elapsed execution time for the CFD blocks, information about the communication speed between all computers is needed. Ctrack (communication tracker program) supports communication speed measurement by systematically sending round trip short messages between different pairs of computers. The messages are sent in such a low rate so that it does not affect the network load.

Balance: This tool predicts the computation and communication costs of any given load distribution based on the information of the parallel application and the measured information of the computers. A cost function is defined as the elapsed time to complete the slowest process. This cost functions is minimized at the end of each load balancing cycle by redistributing the blocks among computers. The optimized load distribution is used in the next execution cycle. The tool allows of abandoning busy processors and utilizing additional processors when they are available. Different optimization algorithms (e.g., greedy algorithm, genetic algorithms and mixed genetic-greedy algorithms) can be utilized to minimize the cost function.

DLB Monitor: This tool launches the computation and communication measurement programs, Ptrack and Ctrack, and the application program simultaneously. It gathers the results of time stamp measurements and Ptrack/Ctrack from all parallel computers. It also runs the Balance tool and restart the parallel application with balanced distribution.

RCopy & Rspawn: Rcopy is a tool to copy files from/to remote NT and Unix based computers using message-passing libraries. This tool is necessary for gathering data for load balancing. Rspawn is a tool to execute system commands (or applications) on remote NT and Unix based computers.

Others: There are other miscellaneous useful tools for DLB, such as, ps, killit and cat were developed for Windows NT.

4 Tools Organization

Fig. 1 depicts the relationship between DLB tools and the parallel application. Fig. 2 shows the flowchart of the load balancing cycles. In each load balancing cycle, Ctrack, Ptrack, and CFD applications are started together. After executing the application code for a fixed amount of time steps, the application code, Ptrack, and Ctrack are stopped. The DLB monitor uses RCopy tool to gather the measured information of the parallel application and computational environment and runs the balancing tool to suggest a new load distribution. Then Ctrack, Ptrack, and CFD applications are started together again. This cycle repeats until the execution of the application code completes.

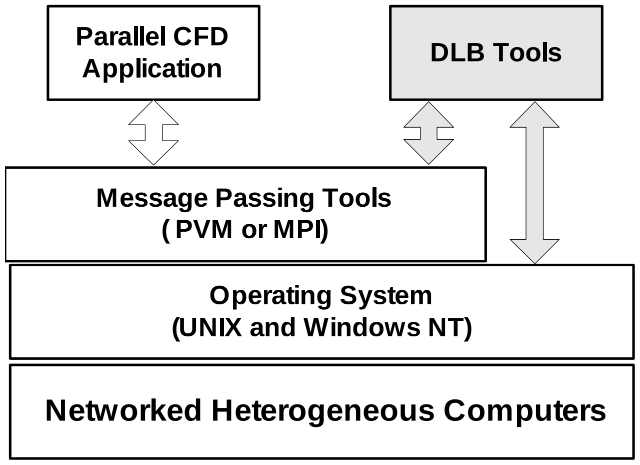
Parallel CFD Application
DLB Tools
Message Passing Tools (PVM or MPI)
Operating System (UNIX and Windows NT)
Networked Heterogeneous Computers

Fig. 1. Distributed computation

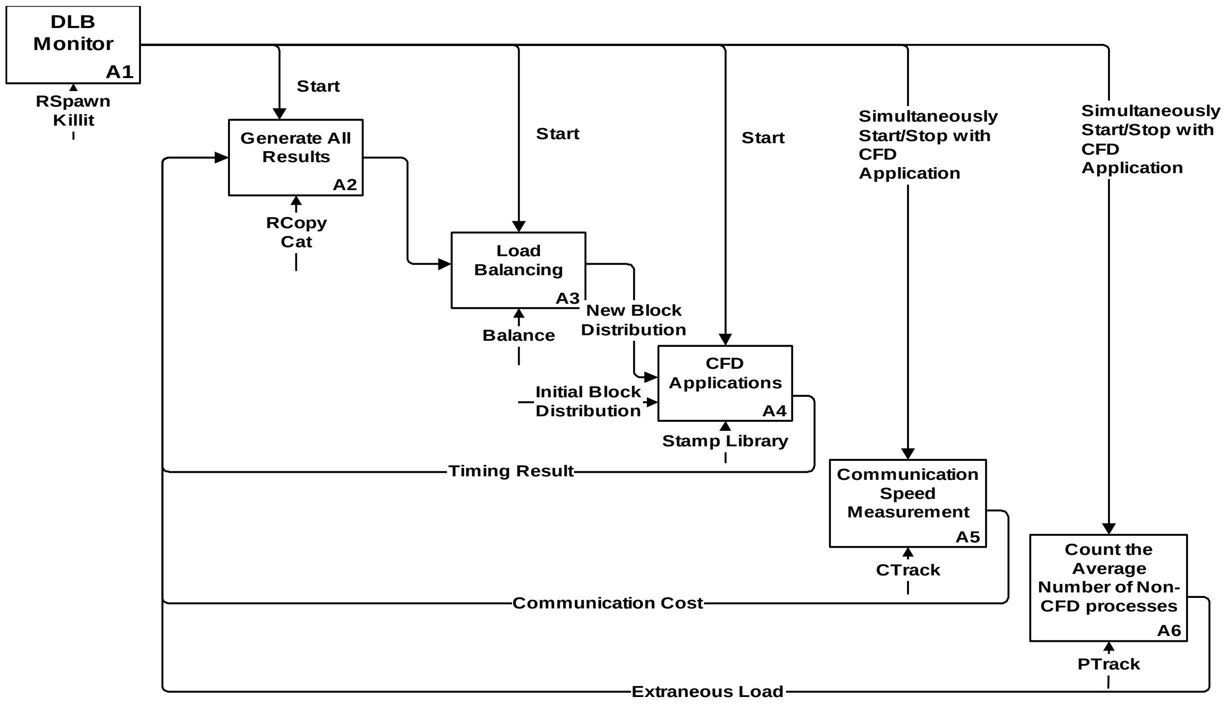

Fig 2. Load balancing cycles.

5 DLB Examples

Two test cases are presented to demonstrate DLB. Both cases include 64 blocks of data. All blocks have similar number of nodes.

In the first case, 24 processors were available: a cluster of 6 IBM RS/6000 workstations in CFD Lab at IUPUI, Indianapolis, Indiana (iw1-iw6), a luster of 10 Windows NT PCs in CFD Lab at IUPUI, Indianapolis, Indiana (ip1-ip7, ip9-ip13), and 8 processors (Thin node 2SC) of the IBM SP in Indiana University, Bloomington, Indiana (b1-b8). The network model is constructed based on the information of computer and network measurement and parallel application. The model shows that each of iw1-iw6 in the CFD lab is approximately 10% faster than the nodes b1 to b5 in Bloomington, 30% faster than nodes b6 to b8 in Bloomington, and nine times faster than one of the PC's. Initial load distribution assigned 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 Fig. 3a. The gray blocks in the figure are the processes of the parallel job. The black block is the extraneous load with respect to parallel job. The balanced distribution is shown in Fig.3b. The white blocks are the blocks being moved buy DLB. Elapsed program execution time was reduced from 13.79 second per time step to 3.71 second per time step due to load balancing.

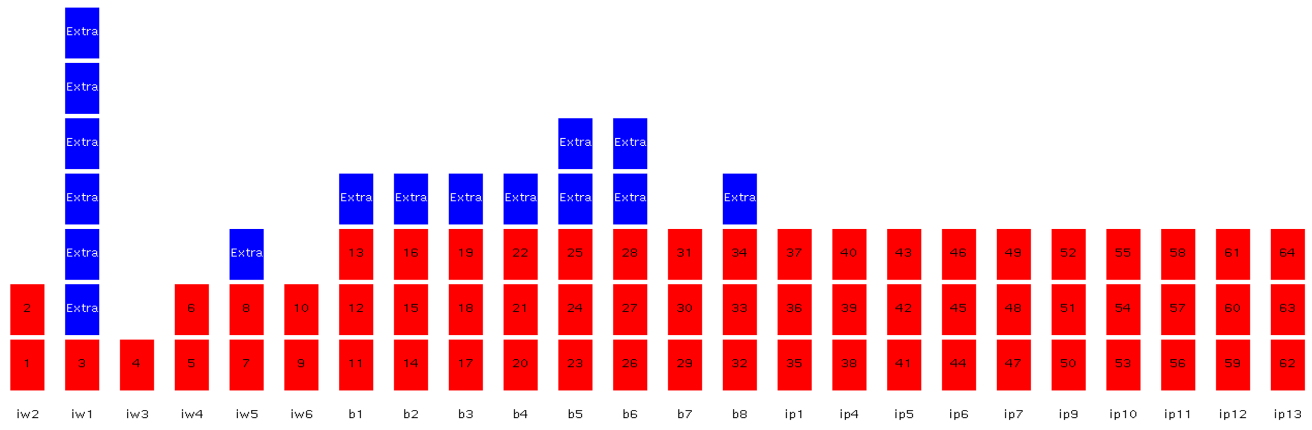

Fig. 3a. DLB test case 1: original distribution

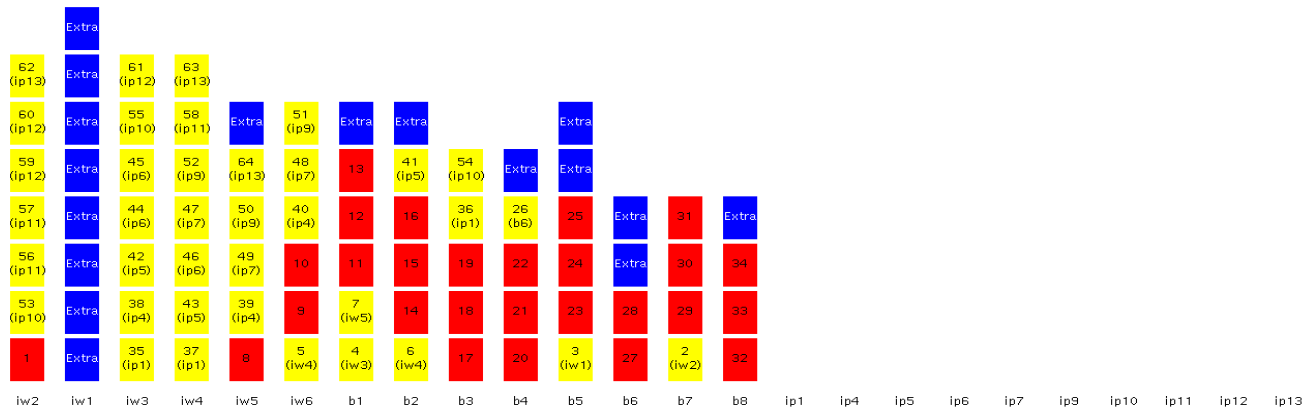

Fig. 3b. DLB test case 1: load balanced distribution.

This example shows the advantage of the online measurements during the application execution. We initially considered all the nodes of the SP in Bloomington to be equal since they use the same CPU and have the same RAM size. But the load balancer puts fewer loads on three of the eight nodes. The output files from the load-balancer indicated that these processors were slower than the others. After discussing with the system administrator, we found out that the five faster nodes have 4 memory boards with 64 MB, the other three nodes have 2 boards with 128 MB each. The five nodes were faster due to the higher memory interleave. Without such measurements, one would try to put equal load on each of these nodes.

The second case involved the following 32 processors: a cluster of 6 IBM RS/6000 in CFD Lab at IUPUI, Indianapolis, Indiana, (iw1-iw6), 8 processors (Thin node 2SC) of the IBM SP at Indiana University, Bloomington, Indiana (b1-b8), and 8 processors (Thin node 2) of the IBM SP, in RUS at Stuttgart, Germany (s1 - s8), 10 processors of the Windows NT PC-Cluster in CFD Lab at IUPUI, Indianapolis (ip1, ip4-ip7, ip9-ip13). Each of iw1-iw6 at CFD Lab at IUPUI is nearly 4 times faster than the SP nodes in Stuttgart. Initial load distribution is shown in Fig. 4a. The load distribution after the DLB is shown in Fig. 4b. Load balance reduced elapsed execution time from 12.71 seconds per time step to 3.45 seconds per time step.

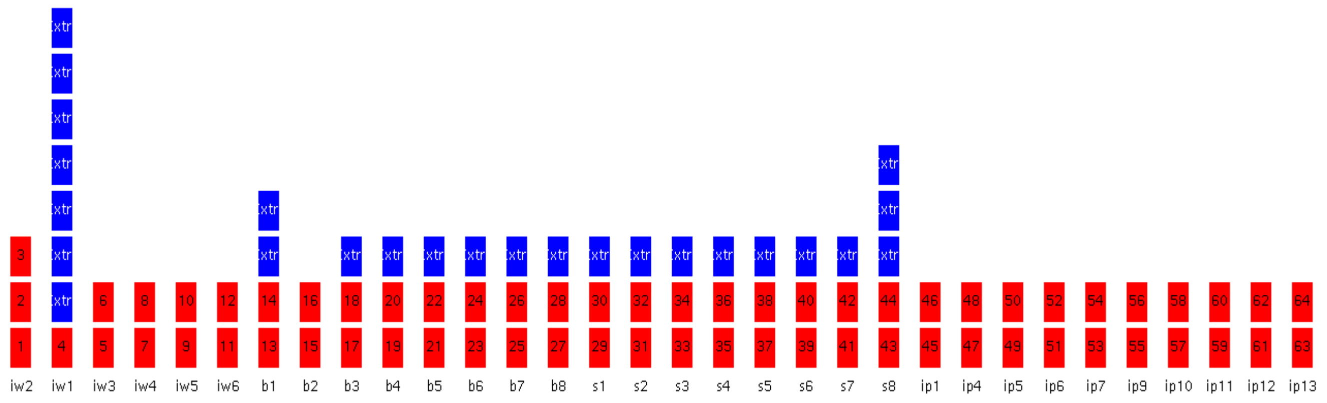

Fig. 4a. DLB test case 2: original distribution

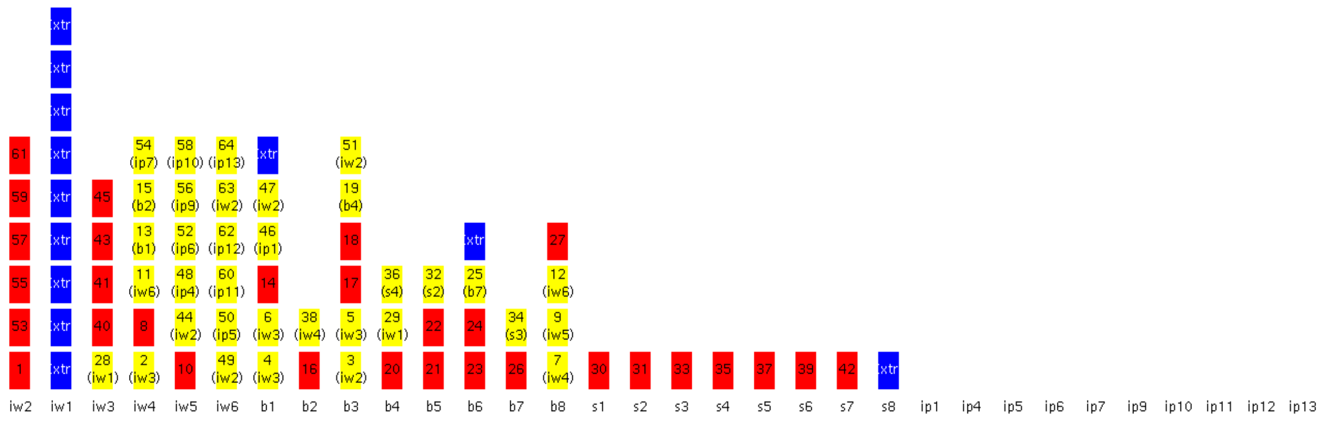

Fig. 4b. DLB test case 2: load balanced distribution

6 Conclusions

The method and tools for the load balancing of parallel processes on distributed computers are studied. The cost functions for load balancing are derived based on the information of both the parallel application and the measured computer and network speed. The different optimization algorithms can be used. The test cases demonstrated the usefulness of DLB.

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, 1992, pp. 101-107.

[2] Chien, Y.P., Ecer, A. Akay, H.U., and Secer, S., "Communication Cost Function for Parallel CFD in a Heterogeneous Environment Using Ethernet," (invited paper) *Proceedings of Parallel CFD '96 Conference* (edited by P. Schiano, et al., Elsevier Science), May 1996, Capri, Italy, pp. 1-10.

[3] Chien*, Y.P., Ecer, A., Akay, H.U., Secer, S., "Cost Estimation for Parallel CFD Using Variable Time-Stepping Algorithms," to appear on *International Journal of Computational Fluid Dynamics.*

[4] Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., and Sunderam, V. (1993), 'PVM 3.0 User's Guide and Reference Manual,' *Oak Ridge National Laboratory Technical Report*, ORNL/TM-12187.