TASK QUARTERLY 8 No 3 (2004), 327–332

PARALLEL COMPUTING IN A NETWORK OF WORKSTATIONS

RAFAŁ OGRODOWCZYK[1] AND KRZYSZTOF MURAWSKI[2]

[1] *Institute of Mathematics and Informatics, Higher Vocational State School in Chelm, Pocztowa 54, 22-100 Chelm, Poland*
rogrodow@pwsz.chelm.pl

[2] *Institute of Physics, UMCS, Radziszewskiego 10, 20-031 Lublin, Poland*
kmurawsk@tytan.umcs.lublin.pl

(Received 15 April 2004)

Abstract: In this paper we describe a few architectures and software for parallel-processing computers. We have tested a cluster constructed with the use of MPI. All tests have been performed for one- and two-dimensional magneto-hydrodynamic plasma. We have concluded from the results of these tests that a simple problem should be run in a sequential node, as its execution time does not essentially decrease with the number of processors used. At the same time, the execution time of a complex problem decreases significantly with the number of processors. In the case of two-dimensional plasma the acceleration factor has reached the value of 3.7 with the use of 10 processors.

Keywords: parallel computing, clusters, parallel-processing systems

1. Introduction

We often face challenging scientific problems. Our strategy is then to test the existing methods or develop new theories. As a consequence of the internal complexity of these problems, fast computers are often the only tools leading to solutions of physically realistic models. However, fast computers are usually too expensive for small scientific groups. Therefore, many research centers continue to work in order to link personal computers into clusters and write software to logically connect these nodes [1]. Such clusters are able to execute parallel codes and divide a large task into smaller parts which can be solved simultaneously [1, 2].

The idea of linking computers is not new. It was first developed by the US Army in the 1960s as protection against a potential Soviet nuclear attack. Such supercomputers were designed as complex machine architectures. The cost of their production was very high and the computing problem was difficult to implement. In the early 1990s, with the sudden development of PCs, scientists started to connect computers and wrote appropriate software which made it possible to use joined PCs. In

a computer center we can implement a few types of cluster architectures and parallel-processing systems and every such system has its advantages and disadvantages for advanced computations.

The purpose of this paper is to present a few architectures and software of parallel-processing systems (see Section 2). We show similarities and differences between several different implementations of parallel-processing systems in Section 3. In Section 4 we describe the results of the tests we have performed with our cluster for numerical simulations of MHD waves.

2. Architecture of parallel-processing computers

A parallel-processing computer is a system containing at least two processors able to solve a computational task simultaneously [1]. Parallel computers may be classified by the way in which data is processed and memory is accessed (according to Flynn's taxonomy) [2].

Single Instruction Multiple Data (SIMD) computers can execute the same instructions in many data sets. We can divide this group into vector computers with Global Shared Memory (SM-SIMD) and a class of processors with Local Distributed Memory (DM-SIMD). These systems are very efficient in processing a specific problem by solving a task with a small degree of date exchange between the processors and the memory. Programming this type of parallel computers does not require any specific construction of the algorithm because a compiler parallels the code automatically.

Multiple Instruction Multiple Data (MIMD) systems can simultaneously process many datasets with the use of many processors. Owing to the application potential of these systems, the technology is presently dynamically expanding and we can distinguish systems with common memory, shared by all processors (SM-MIMD), and systems with local memory, typically for individual processors (DM-MIMD).

SM-MIMD has three basic ways of hardware realization. First, a model which guarantees the processor Uniform (equal) Memory Access (UMA). To achieve such access is very expensive and it does not guarantee the fastest computations. Systems with a shared memory that do not realize UMA standards and access times are different. They are called Non-Uniform Memory Access (NUMA) systems and are produced to decrease access time to memories integrated with processors (ccNUMA). Such architecture allows faster access to the most often used data sets. Design and construction of SM-MIMD systems with regard to subtle technical solutions is very expensive and makes it impossible to extend the scale of such supercomputers. The advantage of this architecture is a relatively simple way of setting parallel software while executing the calculations on a computer with a common address memory space.

The most popular parallel-processing architecture is DM-MIMD. This standard characterizes a large number of processors with their own memory suitable to execute autonomic tasks. These processors work on aggregated data in their local memories and synchronize the calculations by exchanging messages. The advantages of these systems are their scalability and low costs. The main disadvantage of these systems is the necessity of installation of specialist software which synchronizes parallel computing and communication time. One of the ways to realize an DM-MIMD architecture is to connect a network of PC's and use the appropriate software.

3. Parallel-processing systems software

The software used in parallel-processing systems should quickly and efficiently distribute processes between accessible processors and exchange data sets and messages which synchronize the systems' work. Realizations of this task differ depending on the type of architecture of parallel-processing computers. We have three main implementations for DM-MIMD systems: Multicomputer Operating Systems for UnIX (MOSIX), Parallel Virtual Machines (PVM), Bulk Synchronous Parallel Computation (BSP) and Message Passing Interface (MPI).

MOSIX is an integral part of an operating system [3]. This patch can automatically distribute processes to all nodes, remove processes from slower to faster nodes, load balancing tasks and migrate computing tasks from computers with full memory in order to prevent swapping. Its main disadvantages are as follows:

- MOSIX is unable to parallel processes automatically,
- processes cannot share memory, and
- we have to use nodes of the same structure to build clusters.

PVM is a software application that allows us to connect PCs through a network in order to use them as a single parallel system [4]. Such a cluster of computers can be built from Unix and Windows computers. They can have different parameters and architectures. PVM is able to pass messages dynamically, it synchronizes and processes the management, but it is not an integral part of the operating system. We can add and separate nodes from virtual machines during while this system is at work. Moreover, PVM can balance load.

BSP is a programming standard which allows us to create a scalable software application [5]. BSPlibs supports Single Program Multiple Date (SPMD) programming model. It can be implemented in C++ and Fortran, deals with data set distribution and introduces a minimization mechanism of communication. The BSP standard is implemented in various machine architectures and operating systems. This model is developed in European research centers and as an alternative to the American MPI project.

MPI is a message-passing standard to communicate parallel-processing systems [6]. This library specification with prototype procedures and functions enables communication and synchronization of calculations between different nodes of a cluster. Messages are sent between the processes of the same programme only. The MPI standard does not support such services as load balancing, dynamic migration of processes and memory management. Its main disadvantage is that a breakdown or disconnection of one node destroys the parallel calculation. MPI has many implementations on different platforms and it is supported by most scientific software packages.

4. Our network of workstations

We have decided to use DM-MIMD architectures in the construction of our parallel-processing system. The main reason for such choice is that we could use a network of laboratory computers.

We connected 10 nodes of Intel Pentium IV 2.6 GHz and 256 MB of RAM with a Fast Ethernet network. We used the MPI standard as the software configuration

for our Network of Workstations (NOW). This software package is implemented in the FLASH code, which is designed to solve magneto-hydrodynamic equations. This code was developed at the ASCI/Alliance Center for Astrophysical Thermonuclear Flashes at the University of Chicago [7]. In the FLASH code, a numerical problem is implemented by selecting appropriate modules.

5. Numerical model

We have performed two numerical tests by simulating wave processes in plasma. The processes can be described by the MHD equations:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{V}) = 0, \tag{1}$$

$$\rho \frac{\partial \mathbf{V}}{\partial t} + \rho(\mathbf{V} \cdot \nabla)\mathbf{V} = -\nabla p + \frac{1}{\mu}(\nabla \times \mathbf{B}) \times \mathbf{B}, \tag{2}$$

$$\frac{\partial p}{\partial t} + \nabla \cdot (p\mathbf{V}) = -(\gamma - 1) p \nabla \cdot \mathbf{V}, \tag{3}$$

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{V} \times \mathbf{B}), \tag{4}$$

$$\nabla \cdot \mathbf{B} = 0, \tag{5}$$

where ρ is the mass density, $\mathbf{V}$ is the plasma velocity vector, p denotes the gas pressure, $\mathbf{B}$ is the magnetic field, and $\gamma = 5/3$ is the adiabatic index [8]. We have considered one- and two-dimensional models of wave propagation and investigated the performance of the NOW cluster.

In the one-dimensional model we have taken the following equilibrium into account:

$$\rho_0(x) = \rho_l(1 + \sin x), \tag{6}$$

$$\mathbf{V}_0 = 0, \quad p_0 = \text{const}, \quad \mathbf{B}_0 = [B_{0x}, 0, 0], \quad B_{0x} = V_A\sqrt{\mu\rho_0}, \quad B_{0x} = \text{const}. \tag{7}$$

Here ρ_l is a constant mass density. B_{0x} is the x component of a magnetic field and $V_A = 10^6\,\mathrm{m/s}$ is the Alfvén speed.

For the two-dimensional model the equilibrium is as follows:

$$\rho_0(x, y) = \rho_l(1 + \sin x + \sin y), \tag{8}$$

$$\mathbf{V}_0 = [0, 0, 0], \quad p_0 = \text{const}, \quad \mathbf{B}_0 = [0, B_{0y}, 0], \quad B_{0y} = V_A\sqrt{\mu\rho_0}, \quad B_{0y} = \text{const}. \tag{9}$$

This equilibrium is perturbed by initially launched pulses in mass density and pressure. For instance, in the one-dimensional case these pulse are:

$$\rho(x, t = 0) = A_\rho e^{-[(x - x_0)/w]^2}, \tag{10}$$

$$p(x, t = 0) = A_p e^{-[(x - x_0)/w]^2}, \tag{11}$$

where A_ρ and A_p denote relative amplitudes of the pulse, $x_0 = 2 \cdot 10^6\,\mathrm{m}$ is its initial position and $w = 1.25 \cdot 10^6\,\mathrm{m}$ is the pulse's width.

6. Numerical tests

We define the acceleration coefficient, $S(n,p)$, which allows us to describe quantitatively properties of the MPI cluster. $S(n,p)$ is defined as the following ratio:

$$S(n,p) = \frac{T(n,1)}{T(n,p)}. \tag{12}$$

Here $T(n,p)$ is the execution time of an n-size task run by p processors.

First, we consider a one-dimensional case for which $n = 1$. We run all numerical jobs up to time $t = 10$s and set an initial numerical grid which consists of 500 blocks over the spatial domain $0 \le x \le 5 \cdot 10^9$ cm. Each block contains 8 interior cells, which are surrounded by 4 ghost cells on each side. Table 1 displays the dependence of execution time on the number of processors and the value of $S(1,p)$. The test results have shown that for the case of $n = 1$ the execution time does not decrease significantly with the number of nodes (Figure 1). It results from the fact that communication and synchronization between parallel computing processes take a relatively long time. As we have $S(1,2) = 0.43$ in the case of two processors, it follows that computation is more than twice slower than in the single processor case and – in consequence – we conclude that a simple problem should be run in a sequential mode instead.

We now consider the two-dimensional problem for which the equilibrium state is described by Equations (8) and (9).This equilibrium is perturbed by the impulses given by Equations (10) and (11). We run this problem up to $t = 1$s with the use of 4 500 blocks over the spatial domain $0 \le x \le 4 \cdot 10^9$ cm, $0 \le y \le 4 \cdot 10^9$ cm.

Table 2 and Figure 2 show the dependence of execution time on the number of processors and the value of $S(2,p)$.

The execution time apparently decreases with the increasing number of processors. For the case of a single processor the execution time is 1066s and it drops to 285s for 10 processors, achieving an acceleration of 3.70 ($S(2,10) = 3.70$). Consequently, we can conclude that a cluster can significantly reduce the execution time of complex tasks.

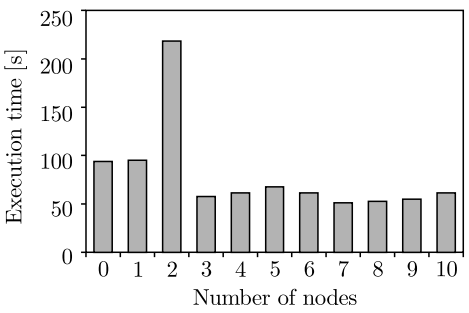

Figure 1. Execution time versus number of processors; zero means running without the use of MPI on one processor

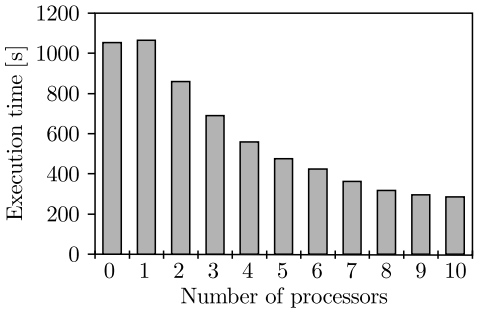

Figure 2. Execution time as a function of the number of processors

Table 1. Execution time and the acceleration coefficient, $S(1,p)$, versus number of processors, p

Numbers of processors, p	Execution time [s]	$S(1,p)$
0	96	1.00
1	97	0.99
2	223	0.43
3	59	1.63
4	63	1.52
5	69	1.39
6	63	1.52
7	52	1.85
8	54	1.78
9	56	1.71
10	63	1.52

Table 2. Execution time and the acceleration coefficient, $S(2,p)$, versus number of processors, p

Numbers of processors, p	Execution time [s]	$S(2,p)$
0	1054	1.00
1	1066	0.99
2	859	1.23
3	689	1.53
4	560	1.88
5	476	2.21
6	425	2.48
7	362	2.91
8	317	3.32
9	295	3.57
10	285	3.70

7. Summary

In this paper we have discussed parallel computing technology. We have performed two tests for one- and two-dimensional plasma described by a set of magneto-hydrodynamic equations.

Our main conclusion is that single problems should be run on a sequential computer, while the use of an MPI cluster significantly reduces the execution time of complex tasks.

The results prove that a delay in communication between different nodes significantly reduces the computing efficiency. While choosing a parallel-computing system we should take into account the network's capacity and protocol, delays of the input/output devices, as well as the system's requirements of the problem, *viz.* the quantity of transferred data and the frequency of its exchange.

We have performed two tests to check the efficiency of systems with load balancing. We are going to implement MOSIX and MPI simultaneously in the NOW cluster. This solution will assure dynamic scalability of the parallel-processing system.

The software used in this work was in part developed by the DOE-supported ASCI/Alliance Center for Astrophysical Thermonuclear Flashes at the University of Chicago.

References

[1] Spector D 2000 *Building Linux Clusters*, O'Relly
[2] Hargrove W W, Hoffman F M and Sterling T 2001 *The Do-IT-Yourself Superkomputer*, Scientific America
[3] http://www.mosix.org/
[4] http://www.csm.ornl.gov/pvm/pvm_home.html
[5] http://www.bsp-worldwide.org/
[6] http://www-unix.anl.gov/mpi
[7] http://flash.uchicago.edu/flashcode
[8] Murawski K 2002 *Analytical and Numerical Methods for Wave Propagation in Fluids*, World Scientific, Singapore