Software for Parallel Processing

Alecu Felician
Academy of Economic Studies, Bucharest, Romania
E-mail: alecu.felician@ie.ase.ro

Abstract

Few years ago, parallel computers could be found only in research laboratories and they were used mainly for computation intensive applications such as numerical simulations of complex systems.

Today, parallel computers are used to execute both data intensive applications in commerce and computation intensive applications in science and engineering.

Parallel programming languages are used to develop parallel commercial or scientific applications. There are a lot of parallel programming languages available on the market. Some of them are based on adding parallel structures to classical sequential programming languages like Pascal, C and Fortran. The parallel structures help the programmer to define processes and threads and also to design the communication and synchronization mechanisms used by parallel program.

Concurrency and scalability becomes a fundamental requirement for algorithms and programs. A program has to be able to use a variable number of processors and also has to be able to run on multiple architectures.

The base entity in computers programming is the process or task. A process is a dynamic entity and represents an execution instance of a code segment. The parallelism can be achieved by executing multiple processes on different processors. Threads are lighter processes used to reduce the process switch overhead.

A distributed operatring system is a special kind of software used on a distributed system. It manages the system shared resources used by multiple processes, the process scheduling activity (how processes are allocated on available processors), the communication and synchronization between running processes and so on.

Multiprocessors are known as tightly coupled systems and multicomputers as loosely coupled systems. The software for parallel computers could be also tightly coupled or loosely coupled.

Combining loosely and tightly coupled hardware and software we can identify four distributed operating systems categories. The loosely coupled software and tightly coupled hardware case is not met in practice and therefore will not be covered in this paper.

Introduction

A parallel computer is a set of processors that are able to work cooperatively to solve a computational problem. Based on this definition, a parallel computer could be a supercomputer with hundreds or thousands of processors or could be a network of workstations.

According to Tanenbaum, a distributed system is a set of independent computers that appear to the user as a single one. So, the computers have to be independent and the software has to hide individual computers to the users. MIMD computers and workstations connected through LAN and are examples of distributed systems.

Content

1. Trends in Applications

Nowadays applications have growing computational needs. As computers become ever faster, we can expect that one-day computers will become fast enough. However, new applications will arise and these applications will demand faster computers.

In our days, commercial applications are the most used on parallel computers. These applications include graphics, virtual reality, decision support, parallel databases, medicine diagnosis and so on. A computer that runs such an application has to be able to process large amount of data in sophisticated ways. We can say with no doubt that commercial applications will define future parallel computers architecture, but scientific applications will still remain important users of parallel computing technology.

Trends in commercial and scientific applications are merging as commercial applications perform more sophisticated computations and scientific applications become more data intensive.

Today, more parallel programming languages and compilers, based on dependencies detected in source code, are able to automatically split the program into multiple processes and/or threads to be executed concurrently on a parallel system. Processes and threads can communicate each other using shared memory or message passing communication mechanisms. It is still difficult to detect, analyze and manage complex programs dependencies.

2. Distributed Operating Systems

A distributed system is a set of computers that can communicate and collaborate each other using software and hardware interconnecting components. Multiprocessors (MIMD computers using shared memory architecture), multicomputers connected through static or dynamic interconnection networks

(MIMD computers using message passing architecture) and workstations connected through local area network are examples of such distributed systems.

An operating system used on distributed system is named distributed operating system and it is the extension for multiprocessor architectures of multitasking and multiprogramming operating systems.

Multitasking operating systems can execute concurrently multiple processes on single processor computers using resources sharing.

Multiprogramming operating systems are multitasking systems that have the ability to protect the memory and to control concurrent processes access to the shared resources.

2.1. Processes and Threads

When running an executable file, a process is created by the operating system. Processes can be found in all operating systems (multiprogramming, multitasking, parallel and distributed).

A process will receive a process ID – a unique number that will identify the process in the system.

Every process has its own virtual addresses space representing the range of addresses that can be accessed. A process cannot access other process addresses space.

A lot of operating systems are implementing a virtual memory mechanism for loading just a part of a process addresses space into the physical memory. The virtual memory mechanism can use paging, swapping or a combination of these methods.

Every process has its own control block where the operating system stores information about the process. When a new process is created, the operating system creates a new control block for that process. When the process ends its execution, the control block is removed from the memory and destroyed. A process control block consists of process ID (PID), process priority, process state (ready to run, suspended, execution), hardware status (processor registers and flags), scheduling information and resources used by the process (files, input, output).

The operating system can switch between two processes. The operation is named process switch and is requested by the operating system only. Process switch is a complex operation with a significant overhead that can affect system performance.

A thread is a software method used to reduce the process switch overhead that affects the system performance. A thread is a lighter process with a simplified status. In multithreading systems, a thread is the base scheduling entity.

A thread has its own stack and hardware status. The thread switch implies the saving and retrieving of the hardware status and of the stack. The thread switch between threads of the same process is fast and efficient because the parent process manages all other resources, except processor. Unfortunately, the thread switch between threads of different processes involves process switch overhead.

A process can be created dynamically by another running process using the *fork* operating system function. A new child process is created when invoking fork function.

The fork operation is used to divide a program in two sequences that can be run in parallel. The child process will execute a program segment and its parent will execute the other segment. Child and parent processes are executed from the same text segment. The child process, at creation time, receives a copy of parent data segment.

Using the *exec* operating function, it is possible to change the text segment of the child process. The parent and child processes will have different text and data segments.

The parent process can be forced to wait for child process execution end using the *join* function.

2.2. Types Of Distributed Operating Systems

Multiprocessors are known as tightly coupled systems and multicomputers as loosely coupled systems.

The software for parallel computers could be also tightly coupled or loosely coupled.

The loosely coupled software allows computers and users of a distributed system to be independent each other but having a limited possibility to cooperate. An example of such a system is a group of computers connected through a local network and using some shared resources such files and printers. If the interconnection network broke down, individual computers could be used but without some features like accessing files over the network.

At the opposite side is tightly coupled software. If we use a multiprocessor in order to solve an intensive computational problem, every processor will operate with a data set and the final result will be obtained combining partial results. In such a case, an interconnection network malfunction may result in the incapacity of the computer to solve the problem.

Combining loosely and tightly coupled hardware and software we can identify four distributed operating systems categories. The loosely coupled software and tightly coupled hardware case is not met in practice and therefore will not be covered below.

Network operating systems represent the loosely coupled hardware and loosely coupled software case. A typical example of such a system is a set of workstations connected together through a local area network (LAN).

There are only a few processing requirements at system level. Processes executed over the network do not need to be synchronized.

The network operating system has to manage the communication between individual workstations. This is why the workstations do not need to use the same operating system.

A *real distributed operating system* is the case of tightly coupled software used on a loosely coupled hardware.

Because the interconnection network is transparent for the users, the set of computers appears like a single multitasking system and not like a set of independent computers.

The computers appear to the users like a uniprocessor system (virtual uniprocessor), which makes the users not to be concerned by the number of computers. All computers from the network use the same operating system that will manage local resources like virtual memory and process scheduling.

A process has to be able to communicate with any other process, no matter if the second process is running on the same computer or on different one.

Multiprocessing operating systems represent the tightly coupled software and tightly coupled hardware case.

A multiprocessing operating system is a multitasking operating system running on a multiple processor system. There is a single list of ready to run processes. When a new process is ready to run it is added to the list located in the shared memory area. The list can be accessed by any process. When a processor is free, it extracts a process from the list and executes it. The operating system has to implement mutual exclusion mechanisms (semaphores, monitors, locks or events using various protocols) in order to protect the concurrent accesses to the ready to run processes list. Using such mechanisms, a process has exclusive access to the processes list and it can extract the first list entry. Then, the list is released and the process is executed.

The system also appears as a virtual uniprocessor for users and the same operating system is executed in every processing unit.

References

1. A. Agarwal, *Performance Tradeoffs in Multithread Processors*, IEEE, 1992
2. G. Dodescu, *Operating Systems*, ASE, 1997.
3. B. P. Lester, *The Art of Parallel Programming*, Prentice Hall, 1993.
4. M. Milencovic, *Operating Systems*, McGraw-Hill Book Company, 1992.
5. R. H. Perrot, *Parellel Programming*, Addison-Wesley, Reading, MA, 1997.
6. M. Quinn, *Parallel Computing*, McGrow-Hill, 1994.
7. A. S. Tanenbaum, *Operating Systems: Design and Implementation*, P. Hall, 1987.
8. A. S. Tanenbaum, *Distributed Operating Systems*, Prentice Hall, 1995.