TEMPUS: Activity 2
S-JEP-8333-94

Parallel Algorithms

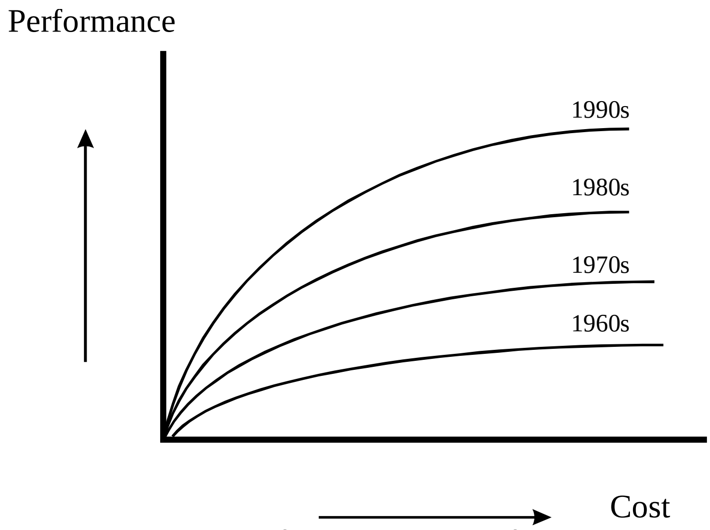
Edited by István Erényi, KFKI-MSZKI

Lecture notes

Introduction

Ever since the appearance of John von Neumann type stored program controlled computers more than half a century ago, their control in order to execute complex tasks has been in the focus of consideration of computer researchers and scientists. The algorithm on the base of which it has been designed and developed defines the behavior of the program. A lot of very effective algorithms have been developed for a tremendous number of applications. Both the theory of design and development and also metrics and evaluation of sequential algorithms have been extensively studied. Well-defined methodologies exist for the assessment of sequential algorithms without considering the type of computer and software they are implemented in. Also an accepted model for both the architecture and the control mechanism of a single processor computer exists: consisting of the processor and random-access memory. The first one is responsible for program execution (both data processing functions and program control actions) while the latter stores the data to be processed and the control information, which realizes the algorithm. The control model is also rather simple. Elementary algorithmic steps (at the architecture and program level called instructions) are one by one read from the memory, interpreted and executed in the order as they are stored in the memory unless a special program control action (branch) exists (program control actions usually are not considered during algorithm design). In the case of program control action, the control instruction itself changes the instruction execution order.

Today computers are used in all fields. These days more than ever, a high demand exists for very powerful, very high throughput computers, high bandwidth of communication lines among them, and extensive computer networks, all of them with tremendous storage capacity besides the computational power. Though sequential computers have gone through substantial evolution – their speed has steadily increased in order to attempt to match the needs of newer applications – they do not provide the necessary computing and storage capacity for challenging applications, neither the structure of widespread networks for a new type of distributed work and business organizations. The performance of sequential computers is approaching saturation. It may be much more expensive to achieve *n*-fold throughput increase by improving a serial computer than buying, installing and operating *n* computers (processor, memory, and etc. collections). The cost-performance curves in Figure 1. show that at the lower end performance increase with cost is close to the linear, while beyond some cost figure it saturates and very small gains in performance may cost tremendous expenses. The cost-performance curves have become sharper today than before. Today a lot of very inexpensive VLSI microprocessors are available. Their speed is within an order of magnitude of the fastest serial computers, but they cost many orders of magnitude less than the processor of these computers. Why not then connect several microprocessors together to achieve the necessary computing power?



Cost-performance curve of serial computers
and its change over the decades

Figure 1.

Using several processors their joined computation capability to solve tasks leads us to the problems of parallel computers, parallel programming. The need for a faster solution of larger tasks requires for the application of such systems. However, efficient programming of parallel computers is a task incomparably more complex than developing serial programs. So is the question of studying the basis of parallel computing: the development and application of parallel algorithms, as well as efficient parallelization of algorithms.

The execution of parallel algorithms is in much-much closer dependence with the architecture of the computer it is implemented in and the number of processor it runs on. Thus the design and evaluation of parallel algorithms has to pay real attention to the **mutual**

interdependence of algorithms, architectures, and number of processors. So, unlikely to serial algorithms, *when discussing problems of parallel algorithms and their development we can not apart from the architectures of parallel computers.* Thus throughout the chapters of this material we have to refer to the architecture of the system, its architectural parameters, etc.

1.1 Parallel computing

Parallel computers consist of an ensemble of a certain number of processors, memories and other storage units, input-output devices, all connected together through a powerful high throughput communication network. The architecture of the computer is defined by all these components: number of processors, memory units, communication channels among them, and naturally the communication network arrangement that defines the **topology of the system**. The efficiency of the task solution heavily depends on the way and speed as processors may exchange data among each other. That is why the topology of a parallel system is also a basic parameter when developing the algorithm for parallel systems.

Parallel algorithms are implemented in parallel software. Parallel software languages require for special language constructs allowing the definition of parallel operations, communication among the parallel program parts (processes), organization of a problem-free operation of these parallel processes, the resource usage among them, and in some cases the asynchronous control of the processors allowing synchronization actions between them when that becomes necessary. Though in the material we do not discuss such language constructs, more detailed development and implementation of the parallel algorithms need to consider these questions, as well. As it has already been stated above, effective execution requires optimal matching of architectures and algorithms, so the software system has to be tuned to the architecture parameters. The best parallel algorithm and a very powerful parallel computer may perform very poorly and ineffectively if they are not optimally mapped to each other.

1.2 Application of parallel computing

A lot of computer application fields go through sufficient changes when parallel processing is put to use. A lot of applications have been beyond the capacity of sequential computers because of one-two reasons: either the high price of powerful computers, or their limited computing power and storage capacity.

Among those applications one can mention fields where huge amount of data has to be gathered and processed, i.e.: weather forecasting, pollution monitoring, atmosphere, ocean, and biosphere modeling, thermodynamics, image data processing, remote sensing. In these problems the field to be computed is modeled by a three-dimensional grid, where the basic parameters are measured or processed at grid points. The granularity of the grid closely affects the feasibility and precision of the model, as well as the amount of data treated. Other challenging applications are computationally intensive fields like planning, Computer Aided Design systems (e.g. for VLSI circuits), state-space evaluation. The solution of real problems in artificial intelligence that are widely studied as to their application in bio-medicine, bioengineering, natural language understanding and translation, etc. requires also tremendous computational power and cooperative activity of a lot of distributed processing units. Many of these belong to the category of so called **grand challenge problems,** the successful solution of which would have great economical, social, health-care benefits.

Summing up, the main purpose of the usage of parallel computers is to provide faster problem solution, perform fast computations in large and very large size problems. To achieve these goals, a high number of processors has to be collected, interconnected together in order to provide the necessary data exchange means among them and naturally their activity has to be properly coordinated and harmonized to each other for efficient problem solution.

A great variety of parallel computer systems exists. Since the effectiveness of their operation is defined not only by the parallel algorithm design, but also by the successful matching of the algorithm with the architecture, we have to provide some orientation among parallel computer categories.

1.3 Classification of parallel computers

A serial computer in the end user's view is a rather complex system of sophisticated hardware modules and operational software tools. The complexity of parallel computers is even more noticeable. Various parallel solutions at different levels of the processors, interconnection network, as well as in the software system make the number of parallel computer categories incomprehensible. There are neither need, nor ways to consider all the implementation details in parallel computers when developing parallel algorithms for them. In the next chapter, we try to give a brief summary of the basic taxonomy issues. The guidelines for parallel computer classification may be the variety of *architectural features* and *modes of operation*. The aim of this classification outline is to discuss parallel algorithms that are suitable for effective implementation on the architectural categories.

1.4 General metrics for evaluation of the performance of the parallel algorithms

The main goal of the material is to provide fundamentals for development of algorithms that are capable to use efficiently the architectural resources. Since the problem is much more complicated than in the case of sequential algorithm development, we need metrics that may evaluate the performance of the parallel-algorithm–parallel-computer ensemble. Later, in Chapter 1, Section 4 on *Scalability* issues we will give more details, here we only provide the definitions for the basic metrics.

A parallel computer with a parallel algorithm is expected to execute the task much faster than its sequential counterpart. The very natural parameter then for evaluation of the behavior of the parallel system is the **Speedup** factor. We may define it as follows. For a given computational task P with an input size of n items the sequential algorithm has a complexity (expressed in relation with the execution time for the task) $T^*(n)$. The denotation with * means that it is the best know sequential algorithm. If we have a parallel algorithm solving the task in time $T_p(n)$ on a parallel system with p processors, then we may define the speedup measure $S_p(n)$ for the parallel algorithm as:

$$S_p(n) = T^*(n) / T_p(n)$$

In ideal case we would expect to achieve $S_p(n) = p$. Naturally, in most cases the system suffers from some inefficiency, reasons of it are discussed in Chapter 1, Section 4. Thus we need another basic performance metrics parameter, that expresses **efficiency** $E_p(n)$ of the parallel system. It expresses the effectiveness of the usage of the p processors.

$$E_p(n) = T_1(n) / p\ T_p(n)$$

Here $T_1(n)$ refers to the execution time of the given algorithm (not necessarily the best known version) on a single processor system.

If the efficiency parameter value is equal or close to 1 for a given p; i.e. the algorithm runs p times faster on p processors, this means that each processor is executing useful work during the total time of the task solution.