Efficiency of Parallel Processing in Multi-Core Processors

Seyed Javad AZIMFAR

Department of Information Technology, Information Technology Group, Faculty of Engineering and Technical, University of Tarbiat Modares, Tehran, Iran

Received: 22.03.2015; Accepted: 29.05.2015

Abstract. This paper evaluates efficiency in the parallelization of multi-core processors. Early computer systems would work in series. With the aim of enhancing efficiency, processors were directed towards parallelization. This study evaluates parallelization efficiency at the level of multi-core processors, compares the efficiency and productivity of Super Scalar with Pipe line, Super Pipe line techniques, discusses parallelization techniques and examines challenges that may affect these techniques.

Keywords: parallel processing, multi-core processor, parallelization techniques, efficiency evaluation, SMT technique

1. Introduction

Nowadays, computers are one of the essential components of human life and their use is very broad from the most complex scientific and statistical computing to watching a movie and listening to music. But what provides the implementation of the wide range of tasks is the mastermind behind the computer processor. Processors that are known today as CPUs (Central Processing Units) are, in fact, integrated circuit transistors that led to the birth of the third generation of computers. The type of microprocessor used in any PC, has an undeniable impact on all aspects of its life and in a large extent determines, the operational capacity and user satisfaction from the implementation of computer applications. With the increasing use of computers for research works in universities, research and businesses centers, the need for faster processing has increased and turned into a basic requirement. Today, parallel processing plays a major role in meeting this need. The speed of current personal computers has steeply increased than their ancestors but even this astronomical speed is too slow for the implementation of some advanced applications. Among the areas which need high speed processing computers, one can point to simulation application in nuclear research, nanotechnology, weather forecasting applications, computerized filmmaking applications, professional animation-making applications and many different fields that require high speed processing to become effective in a proper time. One way to achieve a very high processing speed is to use parallel processing [1, 11].

This article is organized in five sections. Section 2 explained basic premises, the concept of multi-core processor, parallelization and evaluation. Section 3 examines the efficiency of parallelization in processors with and without SMT technique. Section 4 discusses the efficiency evaluation of parallelization techniques, introduces challenges that may threaten these techniques and provides solutions to meet these challenges. Finally, the last section is summary and conclusions.

2. Basic concepts

2.1. Multi-core processor

Processors were initially developed only by one core. A core is a part of a processor that actually performs the task of reading and implementing instructions. Single-core processors can execute a single instruction, at any given moment. Of course, the internal pipe lines of these processors allow them to process multiple instructions simultaneously, but these instructions enter the pipe line one by one [11-
21]. No doubt that single-core processor will be forgotten in a short time. Today, even processors that are used in simpler devices like smart phones and netbooks tend to multi-core architecture. Therefore, we must accept that tomorrow is the era of multi-core processors and parallel processing. The techniques of multi-core processor design have been highly regarded. A multi-core microprocessor is usually a combination of two or more independent cores located in a package [2, 8, 9 and 13]. In multi-core processors, cores may share an integrated cache or may each have a separate and independent cache [6, 13 and 21]. Each core independently applies parallelization approaches such as optimization operations. The efficiency of multi-core processors, compared with single-core ones, depends on the type of problems they solve and algorithms used and how to implement the relevant software. In addition, the more the power of benefiting from parallelization of core performance, the result will be better. A multi-core processor system has multiple advantages over a system with multiple processors. Due to multiple cores in a package, the cache coherence management system works at a higher clock rate compared to when the signals must go out of the CPU. In multi-core processors, signals will have less distance to travel to switch between CPUs which leads to higher data process capacity per unit of time and lower errors in the signal transduction [2, 9, 13, 21 and 22].

2.2. Parallel processing

Parallel computing refers to the implementation of different parts of an application in multiple processors in order to achieve the fastest results. Normally, applications implement instructions sequentially, and the application goes through a specific path and only one instruction can run at a time. In sequential processing, instructions run sequentially by the processor, and the implementation speed is proportional to the processor speed (Figure 1).

![Figure 1](image1.png)

**Figure 1.** Series calculations.

In its simplest definition, parallel processing refers to the simultaneous use of computing resources. In this way, the problem is divided into multiple smaller problems. Each problem is finally processed simultaneously using computing resources and produces the desired result. In fact, in parallel processing, instructions run in multiple processors but the running speed is not necessarily equal to the number of processors multiplied by the speed of a processor (Figure 2).

![Figure 2](image2.png)

**Figure 2.** Parallel computing and its application.
A computational problem can be solved by parallel processing method that has certain characteristics. For example, it can be divided into multiple parts that can run simultaneously. On the other hand, at any time, multiple instructions can run together (without interference) and finally, calculations can be done in less time using multiple CPUs \[19-22 \text{, 10-13}\].

Among primary reasons for using parallel processing, one can point to saving time, the ability to solve larger problems in a fixed time interval and performing multiple tasks simultaneously. Another advantage of parallel processing is to use non-local sources, for example, the use of computers in a network or Internet when enough local resources are not available. In addition, the use of cheaper and available computing resources in parallel can save the purchase cost of mainframes. When a single computer is used, there is a limitation on its maximum amount of memory. To overcome this limitation for solving large-scale problems, one can use the memory of multiple computers.

Parallel processing was developed over the past 20 years as a necessary tool to solve problems in many scientific areas. Methods to achieve the necessary efficiency in parallel processors are wider and more than single processors. Since processors at first were single-core and worked in series, they were directed towards parallelization with the aim of enhancing the efficiency and speed in computer systems. With the advent of pipe line techniques, the series operations were divided into multiple small parallel operations, each of which run in a part of the pipe line along with other small operations, running at other parts. Once the SMT technique was introduced, parallelization emerged in single-core processors, and the process improved with the advent of multi-cores and multi-CPU's \[3-5, 11, 23-27\]. The important thing is that synchronization in processors is the same as parallelization in processors which brings speed of information processing. Synchronization means that the processor can perform multiple tasks simultaneously \[6, 13, 21 and 25\].

To better understand the concept of series and parallel, consider this example: the information of a word, enter a shift register either in series (one bit by one bit), or in parallel at once. At higher levels, parallel processing is to have multiple operational units that do a single task or multiple different operations simultaneously in parallel. For example, arithmetic, logical and shift operations can be done in three distinct cycles, and operands are given to them under the supervision of a control unit. Parallel processing systems are classified in different ways. For example, these systems can be classified in terms of internal components of processors, connection of processors and finally the way information flows. The instruction set read from the memory is called instruction string. Parallel processing may be applied on instruction strings, or data or both. Based on Flynn's classification, computers can be divided into four groups: single instruction and single data (SISD), single instruction and multiple data (SIMD), multiple instruction and single data (MISD), multiple instruction and multiple data (MIMD). In fact, the SISD system is a typical computer with one calculation unit, memory and control in which instructions are executed in series and the system may have the internal features of parallel processing. In this case, parallel processing may be done by pipe line or multiple operating units. The SIMD systems include multiple operating units, working under the supervision of a control unit. All these units receive a similar instruction from the control unit and execute it on different data. The MIMD method is applied in computers that can run multiple applications simultaneously. Most multi-processor or multi-computer systems are part of this category. The MISD systems are important only in theory without any practical example \[5, 11, 16, 19, 22\text{and }28\].

2.3. Evaluation

Evaluation means that how the system does its functions. Evaluation of a system \[1, 13, 23, 26\], has different aspects including functionality, efficiency, reliability, and scalability. Efficiency depends on both resource allocation and response time. As these two factors increase, efficiency decreases \[3, 4\text{ and }13\].

The SMT (Simultaneous Multi-Threading) technique allows multiple independent threads to be carried out simultaneously in a core. For example, if a thread is waiting for completion of decimal operations, another process can use correct units, but it does not allow two concurrent operations of the same type to be carried out in a core. Although SMT is not a true parallel processor, it can be said that SMT brings about parallelization in a single processor.
One of the evaluation criteria for a parallel system is speed factor \( S(p) \) which is the ratio of the time required to solve a problem by a \( T_{\text{sequential}} \) processor to \( T_{\text{parallel}} \) (time required to solve the same problem by a parallel system consists of \( P \) processors).

\[
S(p) = \frac{T_{\text{sequential}}}{T_{\text{parallel}}} \quad (1)
\]

If \( T_{\text{parallel}} = O\left(\frac{T_{\text{sequential}}}{p}\right) \), the speed factor is equal to \( p \) and the parallel system is optimized. In practice, it is very difficult to achieve a linear increase in speed (speed increase proportional to the number of processors). This problem stems from the nature of algorithms; i.e., some parts of an algorithm can be parallelized, other parts not. Another criterion used in the evaluation of parallel systems is efficiency \( E(p) \), which is the ratio of the cost of implementing an algorithm in a sequential system to the cost of implementing the same algorithm in a parallel system consists of \( p \) processors. Efficiency is calculated as follows:

\[
E(p) = \frac{1 \times T_{\text{sequential}}}{p \times T_{\text{parallel}}} = \frac{S(p)}{p} \quad (2)
\]

To evaluate the practical efficiency of algorithms, the MergeHull algorithm with a memory space of \( \frac{n}{p} \geq p^2 \) and 64 processors is used. Each processor sends its maximum \( p \) data as the separator set to processors with higher numbers. The next required elements are calculated and returned. Each processor sends the data between the two selected separators to processors with higher numbers, and each processor sends its next element to the processor with higher number. Then each processor identifies results according to the received information. The \( p \)-th processor is the processor receiving the largest amount of data. This processor receives \( p \) \((p-1)\) data in stage 1, \((p-1)\frac{n}{p^2}\) data in stage 2 and \(p-1\) data in stage 3. When returning data, this amount of data is sent and the \( i \)-th processor acts as the \( p \)-th processor. The total communication time is calculated with the following formula:

\[
T_{\text{comm}} = O\left(p^2 + \frac{n}{p}\right)
\]

Calculations are done locally in each processor on \( n/p \) local data as well as data from other processors. The local computation time can be calculated as follows:

\[
T_{\text{loc}} = O\left(\frac{n}{p} + p^2 \log \frac{n}{p} + \frac{n}{p} \log \frac{n}{p}\right) \quad (4)
\]

Given the above relationships, the total time of algorithm implementation is \( T_{\text{tot}} = T_{\text{loc}} + T_{\text{comm}} \). Next, test results are presented. In charts, the vertical axis represents runtime in microseconds. In practice, runtime for a series of random points is considered equal to the average runtime of the algorithm on 10 sets of random points. Figure (3) shows test results in the case that the number of processors is increasing and the number of points is constant. In random mode, when the number of processors is low, a regular declining behavior is observed. But gradually by increasing the number of processors, runtime increases because less data must be transferred between many processors which will increase the cost of communication. However, the local computing cost is low. Thus, the decreasing trend of time is maintained until communication time can be ignored compared to the local calculation time.
3. Assessment of the efficiency of parallelization in processors with SMT technique

Increasing efficiency was a goal that directed processors towards multi-core. These processors include two or more cores that can perform two or more operations, even of one type, in parallel and simultaneously. In the multi-core CPU, the application is divided into multiple sub-applications or strings, and strings are implemented separately, at the same time on different cores. This means that the total application is implemented earlier compared to a single-processor computer. Therefore, the multi-core processor efficiency is much more than a single-processor core. Although in multi-core processor, at any time, multiple instructions or sub-applications can operate simultaneously on different cores, but in a core, only one instruction can be executed at any time [2, 9 and 13]. With the advent of SMT technique in multi-core processors, the CPU efficiency greatly increased, because, not only different cores could carry out independent and non-independent operations at any time, but with the SMT technique, at any time, multiple independent operations could be run on one core (for example, a floating point operation and an integer operation can operate on one core at the same time). This state has higher efficiency than all previous states because processing time, response time and waiting time are much lower [2-4, 9and13].

4. Discussion and evaluation of the efficiency of parallelization techniques

4.1. Pipe line technique

Pipe line is a technique by which series operations can be divided into multiple parallel sub-operations, so that each of these sub-operations can be run in a part of the pipe line, along with other sub-operations in other parts of the pipe line. Basically the pipe line can be thought as a part of a series of operations of different parts, so that binary data flows in it and any part performs a specific arithmetic operation on the binary data. Thus, the final result is achieved when the related binary data pass from all parts of the pipe line. Therefore, the pipe line structure can be explained as follows: at the beginning, the pipe line is empty, and then instructions gradually enter until it is full and instructions are completed and go out of the pipe line. Finally, the pipe line is emptied. The best pipe line state occurs when it is full, because it has maximum efficiency in the system, but this is not practical because of hazards. The pipe line technique allows the parallelization of instructions at runtime without adding hardware. Figure 4 shows a view of a pipe line which is fully loaded in cycle 6, which is the best state of pipe line, because the pipe line works with maximum power. From clock 6 onwards, an instruction is completed with each clock and leaves the pipe line . [13, 10]
Figure 4. Execution stages of 6 instructions with the pipe line technique (Pipe Line).

The instruction cycle is as follows:
FI: Fetch Instruction
DI: Decode Instruction
CO: Calculate Operand
FO: Fetch Operand
EI: Execute Instruction
WO: Write back Operand

Although the Pipe line technique brings parallelization, it is not quite optimal, because it cannot perform two similar operations at the same time. For example, in the pipe line technique, the Fetch of an instruction can be implemented along with the Decode of another instruction at the same time, while the Fetch of two instructions cannot be implemented at the same time.

4.2. Super Pipe line technique

This technique is the improved version of Pipe line technique, where resources are doubled, i.e., there is a need for additional hardware. The Super Pipe line needs more hardware resources and thus it is more expensive than Pipe line. In Super Pipe line, each stage is divided into two sub-stages. For example, the Fetch of the second instruction can be executed along with the second part of the Fetch of the first instruction. To better understand this, see Figure 5 [10, 13]. Super Pipe line has higher efficiency than Pipe line, because two similar operations can be performed simultaneously which will lead to the earlier implementation of instructions. Among the other benefits of Super Pipe line, one can point to increased efficiency, and reduced waiting time and response time. Despite these advantages, there are disadvantages and limitations in Super Pipe line. Super pipe line needs additional hardware which will increase costs. On the other hand, adding resources is possible to some extent, because adding excessive resources does not increase the system efficiency, but leads to dependencies and higher hazard and overhead in the system [10, 13].

Figure 5. Implementation stages of 6 instructions with Super Pipe line. Here each stage is divided into two sub-stages.

4.3. Super Scalar technique

Super Scalar is the improved version of Pipe line. In the Super Scalar architecture, multiple instructions can start at the same time and run independently. Since, resources are doubled in this architecture, we need to add hardware. A Super Scalar is hardware if multiple Pipe lines work in parallel, i.e., multiple
Efficiency of Parallel Processing in Multi-Core Processors

Instructions are issued and completed within a clock pulse. This feature is not possible in Pipeline. Figure 6 shows the implementation stages of 6 instructions in Super Scalar [7, 13, 14, 17 and 29].

Super Scalar let us publish and complete multiple instructions, based on the number and type of parallel units in one cycle. It consists of multiple pipe lines, each of which runs in parallel multiple instructions. Therefore, it can be concluded that the Super Scalar efficiency is greater than Super pipe line and Pipeline. In Super Scalar, multiple instructions are issued together and completed together, so waiting time and response time are very lower than previous techniques. Moreover, the efficiency (useful work per unit time) will also increase. It can be concluded that, although Super Scalar has higher efficiency than previous techniques, doubling of resources will almost double costs but speed will not double because of hazards [7, 13, 14, 17 and 30]. Obstacles leading to lack of efficiency in Super Scalars are very similar to barriers in other pipeline processes, but the problem of these obstacles in Super Scalars is more than 7 times higher than the other pipe lines, because potential for parallelization is higher in Super scalars, thus more parallelization opportunities are lost. Among the challenges threatening the parallel processing area, one can point to struggle for resources, control dependencies, struggle for data, jumps and hazards [13, 17, 18 and 30]. Delay Branching is a hazard reduction strategy. This solution refers to the movement of independent instructions and placing them immediately after conditional instruction(s). The reason for this is that an independent instruction can be executed at the time when a conditional instruction is run, in order to reduce instruction delays. Another approach to reduce hazard is Branch Prediction which can be studied in future research [13].

Summary and Conclusions

Hardware and software developments always provide access to more powerful and lower-cost systems. Parallel processing is used as an essential tool to solve problems in many scientific fields. In parallel processing, multiple tasks can be processed concurrently, as the result, calculation speed increases, but it requires hardware and software redundancies which will lead to increased costs. Parallelization techniques include Pipeline, Super Pipeline, and Super Scalar. Although the main goal is to increase efficiency in processors, it is difficult to achieve in practice due to hazards. However, techniques such as Jump Prediction can minimize hazards. It was also revealed that the efficiency and productivity of Super Scalar is more than Pipeline, Super Pipeline. In Super Scalar the speed of hazards is not doubled but resources and cost are doubled. Finally, although the system efficiency and its evaluation method are important, but the system design is far from reality, only on the basis of efficiency without taking into account other parameters such as cost, function and other requirements. Jump prediction as a solution for reducing hazards could be studied in future research.

References


Efficiency of Parallel Processing in Multi-Core Processors

