Optics and Lasers in Engineering 50 (2012) 608–617

Contents lists available at ScienceDirect

Optics and Lasers in Engineering

journal homepage: www.elsevier.com/locate/optlaseng

Parallel computing in experimental mechanics and optical measurement: A review

Wenjing Gao, Qian Kemao*

School of Computer Engineering, Nanyang Technological University, Singapore 639798, Singapore

ARTICLE INFO

Available online 20 July 2011

Keywords:
Experimental mechanics
Optical measurement
Parallel computing
GPU
FPGA

ABSTRACT

Because of the merits of non-destruction, high speed, and high sensitivity, optical techniques have been developed for experimental mechanics and optical measurement. In commercial optical systems, the speed performance becomes more important and real-time systems are pursued. Among many acceleration methods, using parallel computing hardware is proven effective. In this paper, the main principles of parallel computing at an application level are introduced; the hardware platforms that support parallel computing are compared; the applications of parallel computing in experimental mechanics and optical measurement are reviewed. Parallel hardware platforms are seen to be useful for the acceleration of various problems. When the computation is time-consuming or real-time performance is required, hardware acceleration is a possible approach for consideration.

1. Introduction

In the past few decades, optical techniques have been developed for experimental mechanics and optical measurement. Because of the merits of non-destruction, high speed and high sensitivity, they are applied to many industrial applications [1,2]. Although the accuracy is usually most concerned in these techniques, high speed is always desired. In commercial optical systems, the speed performance becomes more important and real-time systems are pursued. Many efforts have been put into the speedup of the computations in experimental mechanics and optical measurement including developing faster algorithms and using parallel hardware [3]. Compared with faster algorithms, parallel hardware is a more feasible approach to boost the speed performance while maintaining the required accuracy [3].

Nowadays the concept of parallel computing has permeated into personal computers (PCs) by using general-purpose multicore processors [4]. Although researchers have been working on automatically adapting sequential programs for parallel PCs, it is unfortunately very difficult, and will take at least several decades according to [5]. On the contrary, parallel hardware is more mature and ready for use. This paper reviews the works using parallel hardware in experimental mechanics and optical measurement.

Main principles of parallel computing at an application level and hardware platforms that support parallel computing are introduced in Section 2. In Sections 3–5, the works using parallel hardware in experimental mechanics and optical measurement are reviewed. The discussions and conclusions are given in Sections 6 and 7, respectively.

2. Parallel computing

2.1. Parallel methods

Parallel computing aims to solve a problem faster by dividing the problem into a set of sub-problems which are simultaneously executed by multiple independent computing units (nodes) [6]. The commonly used parallelism methods consist of task parallelism, pipeline parallelism, and data parallelism [7,8]. Most problems are combinations of the above three parallelism methods.

- Task parallelism distributes tasks across different parallel computing nodes. As shown in Fig. 1(a), tasks are implemented as threads, and each thread is executed in a different node to process the same or different data simultaneously.
- Pipeline parallelism is a particular case of task parallelism where multiple tasks depend on each other and process different data simultaneously. Illustrated in Fig. 1(b), three tasks are executed by three different nodes. The execution time for different tasks are denoted as t_1, t_2, and t_3, respectively; and n is the number of datasets. The total computation time is reduced from $t_s=n*(t_1+t_2+t_3)$ to $t_p=(n-2)*\max(t_1,t_2,t_3)+\max(t_1,t_2)+\max(t_2,t_3)+t_1+t_3$. For example, if $t_1=t_2=t_3=t$, and n is large, the computation time is reduced from $t_s=3nt$ to $t_p=(n+2)t\approx nt$, and a speedup of 3 is achieved.
- Data parallelism distributes data across multiple parallel nodes as shown in Fig. 1(c). The same task is executed in each

* Corresponding author.
E-mail address: mkmqian@ntu.edu.sg (Q. Kemao).

0143-8166/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.optlaseng.2011.06.020

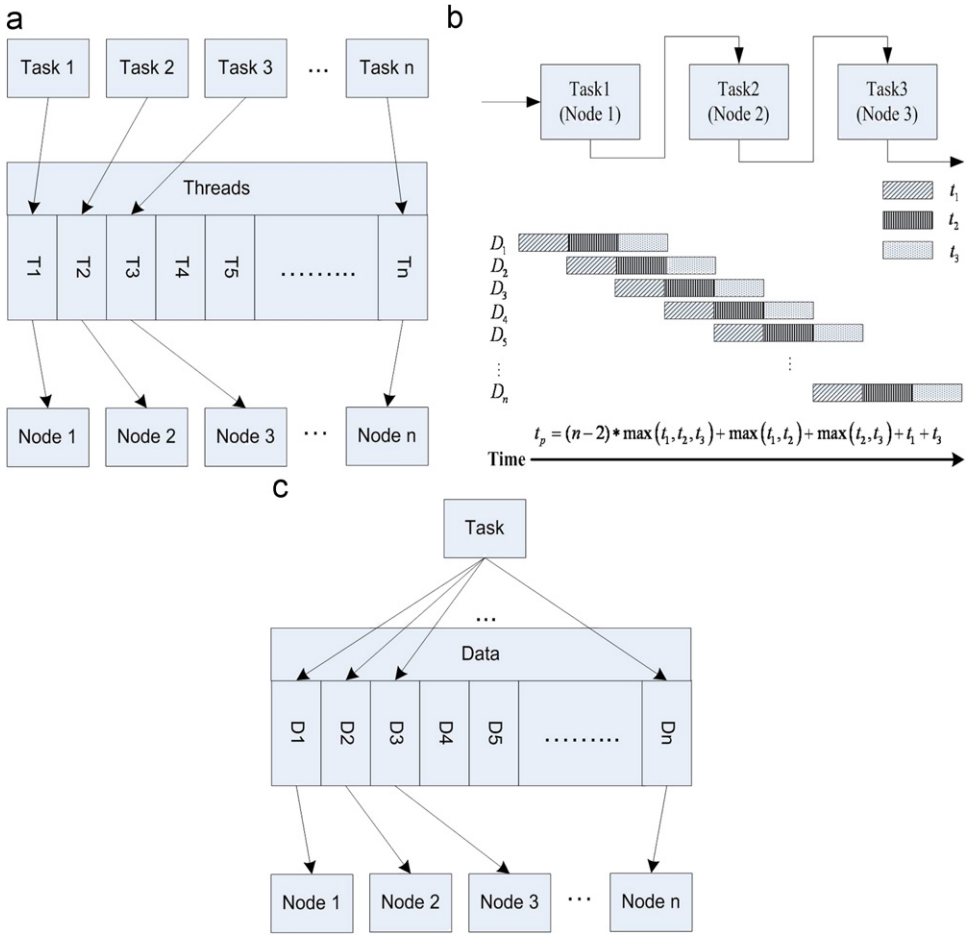

Fig. 1. The mechanism of parallelism methods. (a) Task parallelism; (b) pipeline parallelism and (c) data parallelism.

of the parallel nodes. Data parallelism is especially useful for pixel- and block-wise operations that can be very efficiently parallelized when the number of parallel nodes is large.

2.2. Amdahl's Law [9]

Since usually not all the parts of a given problem can be parallelized, the efficiency of improvement using parallel computing can be roughly estimated by Amdahl's Law [9]. Suppose in a problem, the execution time of the computations that can and cannot be parallelized be t_p and t_u, respectively; the percentage of the parallelizable parts be $p=t_p/(t_p+t_u)$; and the amount of improvement by parallel computing be n. The total execution time is reduced from t_p+t_u to $t_p/n+t_u$, and the overall speedup is,

$$s=\frac{1}{(1-p)+p/n}. \tag{1}$$

Two extreme cases are considered. When p is very large and approaching 1, $s\approx n$ is obtained, which indicates that speedup increases with the number of the nodes. This is denoted as an embarrassingly parallel problem [5] that can be most efficiently parallelized. When p is very small and approaching 0, $s\approx 1$, which shows that the problem is sequential and cannot be accelerated by parallel computing. It should be noted that Amdahl's Law only addresses a theoretically predicted speedup. In practical cases, the extra time introduced by the parallel computing should also be taken into account.

2.3. Hardware platforms

During the past few decades, with the development of semiconductor techniques, the parallel platforms have also been developed. Based on Flynn's taxonomy [10,11], hardware systems can be classified into four architectures: single-instruction-single-data (SISD), single-instruction-multiple-data (SIMD), multiple-instruction-single-data (MISD), and multiple-instruction-multiple-data (MIMD). SISD is equivalent to the sequential hardware architecture; SIMD is for data parallelism; MISD is rarely used; MIMD is for task parallelism and is the most common type of parallel platform. It should be noted that although MIMD targets at task-parallel problems, data-parallel problems can be accelerated by an MIMD hardware with same copy of program executed in each unit. However, this method is not efficient when compared with using SIMD hardware platforms. Many hardware platforms are hybrids of both SIMD and MIMD architectures.

2.3.1. MIMD platforms

The common parallel hardware based on MIMD consists of computer clusters, shared memory multiprocessors, general-purpose multicore processors, and pipeline processors. The grid computing [12] and cloud computing [13] are also common parallel hardware platforms but not given below because they are for grand challenge problems [14] such as protein folding financial modeling and climate modeling.