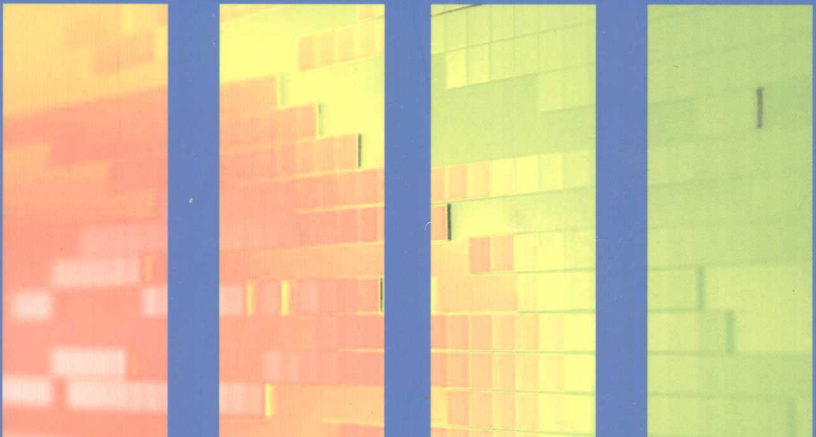
Proceedings of the Inaugural Symposium on Parallel Algorithms, Architectures and Programming

Edited by Hong Shen, Guangzhong Sun and Min Lv



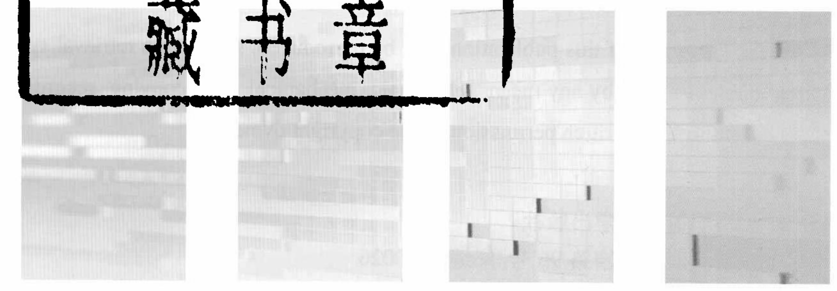
University of Science and Technology of China Press

Proceedings of the Inaugural Symposium on Parallel Algorithms, Architectures and Programming

Edited by Hong Shen, Guangzhong Sun and Min Lv

Sponsored by National Natural Science Foundation of China
University of Science and Technology of China

Organized by University of Science and Technology of China



Sept. 16−Sept. 18, 2008, Hefei, China

University of Science and Technology of China Press

图书在版编目（CIP）数据

并行算法、结构和编程国际研讨会论文集 =Proceedings of Inaugural Symposium on Parallel Algorithms, Architrctures and Programmings: 英文/沈鸿，孙广中，吕敏主编. 合肥：中国科学技术大学出版社, 2008.9

ISBN 978-7-312-02252-4

I. 并… II.① 沈…② 孙…③ 吕… III. 并行算法 — 国际学术会议 — 文集 — 英文 IV. TP301.6-53

中国版本图书馆 CIP 数据核字(2008) 第 140626 号

Proceedings of Inaugural Symposium on Parallel Algorithms, Architectures and Programmings

edited by Hong Shen, Guangzhong Sun and Min Lv

出版发行	中国科学技术大学出版社 安徽省合肥市金寨路 96 号, 邮编：230026 网址 http://press.ustc.edu.cn
印　刷	安徽辉隆农资集团瑞隆印务有限公司
经　销	全国新华书店
开　本	710 mm × 1000 mm　1/16
印　张	12.25　插页2
字　数	180千
版　次	2008年9月第1版
印　次	2008年9月第1次印刷
定　价	60.00元

This book is dedicated to

Academician of the Chinese Academy of Sciences,

Professor Guoliang Chen's 70th birthday

陈国良 院士

Conference Organization
General Chair

Guoliang Chen, University of Science and Technology of China
Benjamin W. Wah, University of Illinois, Urbana-Champaign

Program Chair

Hong Shen, University of Science and Technology of China

Program Committee

Hamid Arabnia, University of Georgia
Francis Chin, University of Hong Kong
Qingshi Gao, University of Science and Technology Beijing
Guangcan Guo, University of Science and Technology of China
Yijie Han, University of Missouri at Kansas City
Yanxiang He, Wuhan University
Tao Jiang, University of California - Riverside
Ming Li, University of Waterloo, Canada
Minglu Li, Shanghai Jiao Tong University
Xiaoming Li, Peking University
Xuandong Li, Nanjing University
Huimin Lin, Institute of Software, Chinese Academy of Sciences
Zhiyong Liu, Institute of Computing Technology, Chinese Academy of Sciences
Koji Nakano, Hiroshima University
Yi Pan, Georgia State University
Yong Qi, Xi'an Jiaotong University
Depei Qian, Beihang University
Hong Shen, University of Science and Technology of China
Xubang Shen, China Aerospace Science and Technology Corporation
XianHe Sun, Illinois Institute of Technology
Zhongxiu Sun, Nanjing University
Baowen Xu, Southeast University
XinYao, University of Birmingham
XiaoDong Zhang, Ohio State University
Weimin Zheng, Tsinghua University

Local Arrangement Chair

Guangzhong Sun, University of Science and Technology of China

Preface

Welcome to the Inaugural Symposium on Parallel Algorithms, Architectures and Programming (PAAP'08). Organized for celebration in honour of the 50th anniversary of University of Science and Technology of China and 70th birthday of Academician of Chinese Academy of Sciences Prof. Guoliang Chen. PAAP'08 is an international forum for scientists, engineers, and practitioners to present the latest research ideas, developments and applications in all areas of parallel and distributed computing with the focuses on parallel algorithms, architectures and programming techniques.

The University of Science and Technology of China was established by the Chinese Academy of Sciences (CAS) in1958 in Beijing as a new type of university in the country, and then moved to Hefei in Anhui Province in 1970. Since its establishment, USTC has made distinguished achievements in talent fostering, scientific research and technology transfer. It has become an important base of top-quality talent training and high-level scientific research for the nation. The University is regarded by the Ministry of Science and Technology as one of the best 4 universities in scientific research performance in China. The University ranks consistently among the best in the review of Chinese top universities by the US journal "Science" and the French journal "Research". It is our honour that PAAP'08 is presented for USTC's 50th anniversary celebration.

Prof. Guoliang Chen, Academician of CAS, is one of the pioneers in the field of parallel computing in China. Over the past 20 years, he has established a complete subject system of "Theory - Design - Implementation - Applications" for parallel algorithms and developed a set of integrated methodologies of "architectures - algorithms - programming" for parallel computing. Prof. Chen has supervised many outstanding researchers in the field. Several papers in this proceedings are authored by his former students. It is our honour to produce this proceedings also specially for Prof. Chen's 70th birthday celebration.

We are very pleased to have three distinguished researchers to deliver keynote addresses at the conference: "Quality Assessment of VoIP Conversations over the Internet" by Benjamin W. Wah from University of Illinois at Urbana-Champaign , "Hyper Parallel Processing (HPP) Architecture: Integration of Scientific and Utility Computing " by Prof. Guojie Li, Academician of Chinese Academy of Engineering, from Institute of Computing Technology of Chinese Academy of Sciences, "Multi-core Computing: Any New Challenges?" by Prof. Lionel M. NI from Hong Kong University of Science and Technology. We are also pleased to have eleven invited speakers from major research institutions to present their state-of-art work in the research field.

This proceeding contains qualified papers selected from submissions for presentation at PAAP'08. The three-day technical program consists of two keynote sessions, sessions for invited talks and technical sessions. We would like to express our thanks to all authors who submitted their papers for this symposium and to all Program Committee members and external referees for their help for the paper reviewing process. We would like to extend our special thanks to the members of symposium organization committee, particularly to Dr. Guang-Zhong Sun and Dr. Min Lv for their great effort and excellent work in organizing the symposium.

Hong Shen
Program Chair
September 2008

contents

Scalable Computing in the Multicore Era

Xian-He Sun, Yong Chen and Surendra Byna

Illinois Institute of Technology, Chicago IL 60616, USA

Abstract. Multicore architecture has become the trend of high performance processors. While it is generally accepted that we have entered the multicore era, concerns exist on scaling multicore processors. Technology is available, but major vendors are hesitant in entering the multicore market with processors that have large number of cores, citing Amdahl's law. This is a very interesting phenomenon, where history seems to repeat itself on the scalability debate of parallel processing occurred 20 years ago. Following the scalable computing concept, especially the fixed-time and memory-bounded speedup metrics, in this study, we argue that the scalability of multicores is not limited by Amdahl's law. We study two speedup models of multicore architecture from the scalable computing point of view. These two models show that multicores have a good scalability and add a new dimension of scalable computing.

1 Introduction

High performance computing has received long missed intensive attention from industry and academia recently. This renewed interest is due to two new developments in computing: cloud computing and multicore processors. Cloud computing employs a cloud of computers, usually a cluster of supercomputers, and Grid technology to provide virtual computers on demand. Multicore processors integrate many cores into one chip to overcome the physical constraints of uni-processor architecture and deliver high computing power with single chip.

Cloud computing and traditional high-end computing applications demand high performance power; multicore architecture has emerged as an able technology to meet that demand. By scaling up the number of cores, multicore processors provide a new dimension to scale up performance.

However, recently we have noticed a very interesting phenomenon. While some small start-up companies are making large-scale multicores [16][10], established companies are reluctant to enter the multicore market with processors that have large number of cores. IBM's Cell processor [6], for instance, has only 8 cores (plus a master core). AMD's mainstream processors, Phenom and Opteron families, have only four cores. While Intel has a road map for multicores, building an 80-core processor in 2011 [15], it is too conservative and slow moving. This slow movement has its theoretical foundations. Based on Amdahl's law, many researchers believe multicore systems are not scalable [1][9]. Amdahl's law states that if a portion of a computer can be improved and another portion of the architecture cannot be improved, then the portion that cannot be improved will quickly dominate the performance, and further improvement of the improvable portion will have little impact. With the known memory-wall problem [14], many believe that multicore processor with a small number of cores, such as 8, is a good design choice. History seems to repeat itself and reminds the days before Gustafson introduced the scalable computing concept for parallel processing in 1988 [7]. IBM and Cray were making parallel machines with 2 to 8 processors, such as IBM 7030 Stretch Data Processing System and Cray Y-MP before scalable computing was introduced. The introduction of the scalable computing concept changed the view of parallel processing and made major vendors entered the massively parallel processing arena. Today, IBM's Petaflop machine Roadrunner at Los Alamos National Laboratory has 25,200 processors. The Ranger supercomputer at Texas Advanced Computing Center has 15,744 processors. Unfortunately, the scalable computing concept [7][13][5][12] has not been well introduced into the multicore processors design at this time. Studying the scalability of multicore processors is a timely research effort. Recently Hill and Marty have studied the Amdahl's law applicability for multicore design and call for models of multicore performance [8]. In response to their call, we study the scalability of multicore processors and analyze two speedup models following the results in scalable parallel processing in this research. We first revisit the three speedup models of parallel processing, fixed-size (Amdahl's law), fixed-time and memory-bounded speedup; we then extend them to multicore scalability analysis. Detailed case studies are presented. We conclude that multicore architecture is scalable and has the potential to achieve linear speedup in scalable computing, where problem size is increased with the number of cores.

2 Speedup Models of Parallel Processing

In this section, we review the three classic speedup models, Amdahl's law [1], Gustafson's law [7], and Sun and Ni's law [13], of parallel processing in brief.

2.1 Amdahl's Law

The original idea presented by Amdahl [1] is a general observation about the performance improvement limit of any enhancement, and was later summarized as the well-known Amdahl's law. When we apply Amdahl's law to parallel processing, we have the speedup metric as:

$$S_{Amdahl} = \frac{Performance_{new}}{Performance_{old}} = \frac{SequentialExecutionTime}{ParallelExecutionTime} = \frac{T_s}{T_p} \quad (1)$$

Suppose α is the fraction of the code that is sequential, which cannot be parallelized, and p is the number of processors. Assuming that all overheads are ignored, we have: $T_p = \alpha T_s + (1-\alpha)T_s/p$. Therefore,

$$S_{Amdahl} = \frac{T_s}{T_p} = \frac{T_s}{\alpha T_s + (1-\alpha)T_s/p} = \frac{1}{\alpha + (1-\alpha)/p} \quad (2)$$

This formula is called Amdahl's law for parallel processing. When p, the number of processors, increases to infinity, the speedup becomes $\lim_{p\to\infty} S_{Amdahl} = \lim_{p\to\infty} \frac{1}{\alpha+(1-\alpha)/p} = 1/\alpha$. This equation shows that the speedup is limited by the sequential fraction, a nature of the problem under study, even when the number of processors is scaled to infinity. Amdahl's law advocates that a large-scale parallel processing is less interesting because the speedup has an upper bound of $1/\alpha$.

2.2 Gustafson's Law

A tacit assumption in Amdahl's law is that the problem size, or the workload, is fixed. The speedup emphasizes time reduction of a given problem. Amdahl's law is thus also called *fixed-size speedup* model. In 1988, Gustafson introduced *fixed-time speedup* model [7] to motivate large-scale parallel processing. The fixed-time speedup model suggests powerful machines can be designed for solving large problems and the problem size should be scaled to match the increased computing capability in a parallel processing system. Thus, the fixed-time speedup is defined as:

$$S_{Gustafson} = \frac{SequentialTimeofSolvingScaledWorkload}{ParallelTimeofSolvingScaledWorkload} \tag{3}$$

Suppose the original workload and the scaled workload finished in the same amount of time are W and W', respectively. We have $W' = \alpha W + (1-\alpha)pW$. Therefore,

$$S_{Gustafson} = \frac{SequentialTimeofSolvingW'}{SequentialTimeofSolvingW} = \frac{W'}{W} = \alpha + (1-\alpha)p \tag{4}$$

This equation is known as Gustafson's law. It states that the fixed-time speedup is a linear function of p if the workload is scaled up to maintain a fixed execution time. Gustafson's law suggests that it is beneficial to build a large-scale parallel system as the speedup can grow linearly with the system size.

2.3 Sun and Ni's Law

Many parallel applications cannot scale up to meet the time bound constraint due to some physical constraint. In practice, the physical constraint is often the memory limitation. In distributed-memory machines, the number of processors and memory are increased in pair. Out-of-core computing will reduce the performance significantly and is largely prohibited. With this in mind, Sun and Ni proposed *memory-bounded speedup* model [13]. Let W^* be the scaled workload under memory space constraint. The memory-bounded speedup is defined as:

$$S_{SunNi} = \frac{SequentialTimeofSolvingScaledWorkload, W^*}{ParallelTimeofSolvingScaledWorkload, W^*} \tag{5}$$

Assume that the parallel portion of the workload can be scaled up $G(p)$ times. That is, the scaled workload is $W^* = \alpha W + (1-\alpha)G(p)W$. The factor $G(p)$ reflects the increase in the workload as the memory capacity increases p times. Therefore,

$$S_{SunNi} = \frac{\alpha W + (1-\alpha)G(p)W}{\alpha W + (1-\alpha)G(p)W/p} = \frac{\alpha + (1-\alpha)G(p)}{\alpha + (1-\alpha)G(p)/p} \tag{6}$$

Sun and Ni's law is a generalization of Amdahl's law and Gustafson's law, where Amdahl's law is a special case with $G(p) = 1$, and Gustafson's law is a special case with $G(p) = p$. In general, the computational workload increases faster than the memory requirement, thus $G(p) > p$ and the memory-bounded speedup model gives a higher speedup than the fixed-size and fixed-time speedup.

3 Multicore Architecture Assumptions and Definitions

We follow the models of parallel processing to study the scalability of multicore architectures. We take data access as the bottleneck that we cannot improve, and study the scalability of multicores in terms of cores in a processor.

3.1 Multicore Architecture

To simplify the discussion, this study assumes symmetric multicore processor architectures. We assume that the multicore processor under study has n cores, and each core has a dedicated primary cache, L1 cache, and all cores share remaining levels of the memory hierarchy. This assumption matches with most of existing multicore/manycore processors that are either commercially available or in production. For simplicity, we also assume that there is no context switch while running a parallel application on a multicore processor.

3.2 Definitions

We introduce the following definitions in order to describe the scalability analysis model of a multicore architecture.

Definition 1. *The work (or workload, or problem size) is defined as the number of instructions that are to be executed.*

Let I denote the number of instructions, or the work. The work is composed of computation instructions and data access instructions. Let I_p denote the number of computation instructions, and I_c denote the number of data communication instructions. Therefore, $I = I_p + I_c$.

Definition 2. *The execution time is defined as the number of CPU cycles spent for executing the instructions, either for computation or for data access.*

Let T denote the execution time. The execution time is composed of the computation time spent on processing units and the communication to wait the data to be ready. Let T_p denote the computation time, and T_c denote the data access time. Therefore,

$$T = T_p + T_c \tag{7}$$

Note that, in the context of parallel processing, this formula is under two assumptions: the load is balanced and every processing unit performs computation and communication at the same time. Following Amdahl's law of parallel processing, we assume that computing is perfectly parallelized and the load is balanced. Since we have assumed that the studied multicore processor is a symmetric architecture, the computation instructions are issued with a same speed on each core. Equation 7 stands under our study. If we assume the execution speed of each core is τ instructions per second, we have $T_p = I_p/\tau$. We can further analyze the data access time T_c as well in a typical multicore architecture. A major fraction of data access time is spent on the data transfer between the lowest-level cache memory, L2 cache in this study, and the main memory. The data access time can thus be roughly modeled as:

$$T_c = (I_c \times (1 - H_{L1}) \times (1 - H_{L2}) \times [F \times L_{word} + (1 - F) \times L_{cache}])/B \quad (8)$$

where H_{L1} is the L1 cache hit ratio, H_{L2} is the L2 cache hit ratio, L_{word} and L_{cache} are word size and L2 cache line size respectively, F is a locality factor to represent the spatial locality characteristic of the work, ranging from 0 in the case of totally random accesses to 1 in the case of totally sequential accesses, and B is the memory bandwidth. When the cache hit ratio, locality factor, word size and cache line size are fixed for a study, we rewrite T_c as: (c_1 is a constant)

$$T_c = c_1 I_c \quad (9)$$

In computer architecture, speedup is a measure of improvement. The following definition explains the speedup concept we employs in this study.

Definition 3. *The speedup, in the context of computer architecture, is defined as the ratio of the execution time in the original architecture and the execution time in the enhanced architecture.*

4 Scalability of Multicore Architecture

Now, we are ready to extend the three speedup models of parallel processing into multicore architectures and present theoretical analysis.

4.1 Amdahl's Law on Multicore Architecture

Amdahl's law [1] is a basic law of architecture design. It shows that the inherited limitation of any architecture improvement is determined by some performance factor that cannot be improved with architecture improvement. For parallel computing, this factor is the portion of the application which must be solved sequentially. In the context of multicore architecture, the data access delay is such a limiting factor that cannot be enhanced no matter how many cores are utilized in computation. With a similar assumption used in the Amdahl's law in parallel processing, we assume that computing can be perfectly parallelized and data access is independent of problem size and the number of cores. Then with an n-core processor architecture, the new execution time is: $T' = T'_p + T'_c = T_p/n + T_c$. Thus, the Amdahl's speedup is:

$$S_{FS} = \frac{T}{T'} = \frac{T_p + T_c}{T_p/n + T_c} \tag{10}$$

Amdahl's law states that the performance gain of a multicore architecture is quickly limited by the data access delay, with an upper bound of $(T_p + T_c)/T_c$. The gap between data access and computing speed has been growing larger and larger during the last three decades and is known as the *memory-wall problem* [14]. By Amdahl's law, the multicore architecture is not scalable. It is a pessimistic view, and has accepted by many in both academia and industry [8][9].

As in parallel processing, Amdahl's law of multicore architecture is under a tacit assumption: the application workload is fixed. Multicore is also a way of parallel processing. The scalable computing concept of parallel processing should be extended to multicore design. Considering data access as the non-improvable performance factor, we study two scaled speedup models in the following section.

4.2 Multicore Architecture for Scalable Computing

Allowing problem size increases with computing power, we now study the scalability of multicore architectures from scalable computing point of view. We first present a *fixed-time* scaled speedup model.

Definition 4. *The fixed-time speedup of a multicore architecture machine is defined as the ratio of execution time of solving the scaled workload on a single core to execution time of solving the scaled workload on multiple cores, where the scaled workload is the amount of work that is finished in the enhanced mode within the same amount of time as in the original mode.*

Following a similar assumption of Gustafson's law of parallel processing, we assume that the communication work/cost is fixed. The assumption is valid since the goal of this study is to show the potential scalability of multicore architectures. According to the definition and the assumption, the scaled workload is $nT_p + T_c$. Therefore, the fixed-time speedup of the multicore architecture from the scaled computing viewpoint is:

$$\begin{aligned} S_{FT} &= \frac{Time of Solving Scaled Workload in Original Mode}{Time of Solving Scaled Workload in Enhanced Mode} \\ &= \frac{Time of Solving Scaled Workload in Original Mode}{Time of Solving Original Workload in Original Mode} = \frac{nT_p+T_c}{T_p+T_c} \end{aligned} \tag{11}$$

This result reveals that, from the scalable computing viewpoint, the multicore architecture is linearly scalable and suitable for large-scale manufacturing as long as the data communication time is fixed. When the number of cores, n, goes to infinity, the speedup can grow linearly with n. This finding confirms that multicore architecture with large number of cores is meaningful and has real scalability potential. We do not need to reduce the data access delay, but the assumption here is the data access delay is fixed and does not increase with the number of cores and the problem size. While formula (11) shows the potential of large-scale multicores, data access remains as a technical hurdle that needs to be overcome.