Contents

List of Figures

List of Tables

Foreword

While applications for mass markets (Internet) and also niche markets (scientific computing) are almost entirely data driven by nature, the ultimate performance of the machine will depend heavily on the quality of input/output (I/O) operations. Yet, all too often researchers in the parallel computing field have relegated the issue of I/O to the realm of assumption ("let us assume that the data has been pre-loaded in the various processors," or "suppose that data can be sent to the processors in linear time," or a myriad of other false premises). Instead, they have concentrated on the easier (which is not to say easy) problem of executing parallel complex applications and have allowed themselves to be driven by the simple law of computing which predicts the doubling of transistors on a chip every 18 months or so (Moore's law). However, at the same time, real disk storage densities have progressed at a rate of 60 to 80% per year but disk access time improvement has been less than 10% per year.

This has proven to be a pitfall. For one thing, Amdahl's law has warned us for a long time that it is the sequential portion of an application which dominates its performance and that further attempts at speeding up the parallel portion would be futile. Better to spend more design time reducing the amount of sequentiality than increasing parallelism or else risk investing too much effort in a cost ineffective approach! As it turns out, poor planning may easily turn I/O operations into a perfectly sequential stream of instructions which will bog down the most massively powerful parallel machines. To deny the existence of the I/O problem has all too often been a fundamental error.

To be sure, parallel computing had to undergo this first stage: it is hardly imaginable to build parallel machines without first making sure that they could be programmed efficiently. The fallacy was for us to imagine implementing easily shared disk storage to duplicate data on sites close to data requests instead of increasing local disk storage. Then, we would need also to increase the speed of the CPU to balance the increase in the amount of available data. Instead, parallel I/O corresponds to this

idea of sharing affordable (and small) disks and to keep them as busy as possible to get the best throughput of data.

Consequently, it is quite heartening to find this book which reflects the new challenges and opportunities of parallel I/Os. The authors have contributed in an original manner to our recognition of the I/O issue as a whole and have endeavored to take you, the reader, on a journey of understanding through this difficult and sometimes ignored topic.

This book is organized into two parts: an introduction to selected topics and a selection of effective solutions in different subtopics of the field.

Part one introduces some motivating I/O problems both for the sequential case and the parallel one. Sequential I/O means in general that I/O requests reach the processor, driver, interface one after the other, while parallel I/O means that there is potentially more than one concurrent I/O operation at any given time. The chapters which make up part one include a study of optimizing the implementation of distant reading/writing (from disk to disk). Then the product of matrices is introduced to illustrate how MPI-2 captures different I/O abstractions. Sequential sorting on disks is presented to emphasize the differences between in-core algorithms and out-of-core algorithms while special optimization techniques (polyphase mergesort) are described. Further, the authors have not ignored the importance of data structures and special attention has been given to this problem (B-trees, special purpose trees, etc.). The authors have recognized the importance of heterogeneity for the future of computing and devote most of Chapter 2 to parallel sorting on heterogeneous clusters.

The second part of the book introduces five research studies: a study of parallel I/O under a parallel file system (Chapter 3), a presentation of benchmarking I/O (Chapter 4), parallel join algorithms on clusters (Chapter 5), a study about scheduling I/Os (Chapter 6) and the design and implementation of a large cache for software RAID (Chapter 7).

It has been an honor to be asked to write this foreword. Indeed, I have enjoyed reading this manuscript. The wide spectrum of topics covered is a testimony to the vitality of the field and I believe that the contributions of this book will be invaluable to students and practitioners in the field.

Jean-Luc Gaudiot
Professor
University of Irvine, California, USA

Introduction

Data sets in large applications such as databases and data-mining applications are often too large to fit entirely inside the computer's internal memories. The resulting input/output communications (or I/O) between internal memory and slower external medium (such as disks) can be a major performance bottleneck: the current ratio between memory access time and disk access time is still about 10. In this context, exploiting locality is one of the very important challenges in the field.

Among the computer architectural solutions proposed to deal with the challenges, we have today the very effective "Cluster Computing Platforms" [BUY 99a, BUY 99b] which are collections of non dedicated systems (for instance PCs) based on low cost hardwares. Thus clusters present themselves as affordable and suitable I/O systems.

Cluster architecture

At the end of the 90s, *networks of workstations* or *clusters* appeared as a variety of distributed architecture sharing "more than nothing" as one says frequently. The "sharing" can consist of sharing the configuration of machines or in a more technical way, the space of addressing files or memory, to globalize the physical resources of the various machines.

This architecture is a reasonable alternative [BUY 99a, BUY 99b] to dedicated parallel machines. It is now noticed that the main factors that led to this acceptance are connected to economic factors: the ratio between cost of the machine and the performance are very advantageous when one can count on mass product component.

A network of workstations consists of:

– standard components (microprocessors, DRAM, disks...) such as one buys in big distributors;

– general and affordable motherboards (one finds also bi-processor cards for as little as 150 euros);

– networks of interconnection based on a tried standard: Fast-Ethernet (100Mbits/s) and Gigabit-Ethernet (1Gbits/s) or dedicated networks such as Myrinet who offer communications of more of 1Gbits/s measured with a latency of the order of $10\mu s$. These performances are of the same order as those of the networks of parallel computers as the IBM SP... while the cost of a Myrinet card is about 1000$USD (only);

– parallel programming libraries, generally by sending messages with MPI or BspLib [MIL 94] or BUP7 [BON 99] which implement the BSP model [VAL 90].

Besides the advantageous economic aspect, the networks of workstations allow *scaling*: the extension of a system is very easily realized and very often, at a very reasonable cost. Indeed, in architecture of client/server type, when the number of clients increases, the server is the bottleneck because it is by definition the only one to return a service. In shared memory architecture, the traffic on the bus becomes a bottleneck, generally from a dozen nodes. In architecture of cluster type, the data are distributed on all the machines and potentially reduce problems of bottleneck.

Managing Inputs and Outputs (I/O): a key challenge

Storage systems represent a vital and growing market and deal with cached, historical, multimedia and scientific data. As the ubiquity of clusters has grown in the past decade by the use of appropriate System Area Networks (SAN), we still have to prove that the "cluster paradigm" is a serious candidate to handle large data sets. One way to achieve good performance, is to allow I/O operations to perform more and more in parallel since a typical disk drive is 10^5 times slower in performing a random access than is achieved in the main memory of a computer. The factor can be 10^6 and even more if we consider the access time of an internal CPU register of an 1Gz PC.

This book is devoted to the recent evolution of Parallel I/O [JIN 01, KOR 97, MAY 00] in the context of cluster computing field. It presents "state of the art" contributions on the subject of I/O management. Thus, the aim of the book is also in the presentation of recent, practical and theoretical advances that could help potential new users in the field. The book attempts to cover the main topics:

– File Systems and Parallel I/O for Clusters;

– Data Distribution and Load Balancing in the Presence of I/O Operations;

– Novel Hardware and Software I/O Architectures;

– Parallel Disk Models and Algorithms;

– Parallel I/O Support for Databases;

– I/O Performance Analysis: Resources and Tools for Benchmarking;

– Drivers and Application Programming Interfaces;

– Advances in Storage Technology;

– Tools for Operating and Managing I/O Operations;

– Compilers Techniques for High Performance I/O Operations;

– Language and Runtime Libraries;

– Network Attached Storage and Storage Area Network;

– Standards and Industrial Experiences with Massive Data Sets.

The book covers in depth the first eight topics. It is divided into two parts. In the first part we introduce all the necessary vocabulary and we exemplify some particular problems. Our goal is both to motivate the reader and to survey some widely used and general problem/solutions. The part could be used to introduce the field to students: it is educationally oriented. It covers in the first chapter some solutions to accelerate disk transfers by working at the driver level; then we focus on how MPI-IO allows one to specify I/O operations and we introduce algorithmic issues for sequential sorting. We also focus on tree data structures and prefetching techniques. Chapter 1 ends by considering disk scheduling algorithms. In chapter 2, we develop recent advances in parallel out-of-core (external) sorting in the case of an heterogeneous cluster.

In the second part of the book, we present recent approaches and solutions that researchers in the field have contributed in the last two years.

The book is not really a textbook in the sense that it is not organized along problem statements, questions, pauses, exercices, problems and research. We want this book to help the reader to understand better the fundamental questions and effective answers underlying the I/O problems. We introduce in the first part some fundamental notions motivated by examples, and then in the second part, we present recent research advances that use some prerequisites presented in the first part. We hope that this book will help readers to produce efficient I/O programs in the future and also will help educators to find sources of exercises.

We guess that the reader is familar with the PC hardware as it is taught at undergraduate course level [CLE 00, DAV 98, HEN 02] and with operating systems [TAN 01, STA 01, NUT 01] and has some background about the following notions in order to facilitate the reading of the first part:

– Parallel programming with MPI;

– Parallel algorithms [AKL 97, JÁJ 92] (complexity notion of (parallel) algorithms);

– Unix file system;

– Benchmarking;

– Sort algorithms and join operation in databases [ULL 02];

– Cache and RAID systems.

The work would not have accomplished without the key efforts of the editorial board as listed below. We would also address a special thank to Rajkumar Buyya who has been one of the main supporter of the book idea. His strong involvement in the IEEE Task Force on Cluster Computing has been a model for our motivation.

General interest bibliography

[AKL 97] AKL S., *Parallel Computation, Models and Methods*, Prentice Hall, 1997.

[BON 99] BONORDEN O., JUURLINK B., VON OTTE I., RIEPING I., "The Paderborn University BSP (PUB) Library - Design, Implementation and Performance", *13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing, 12 - 16 April, 1999, San Juan, Puerto Rico, available electronically through IEEE Computer Society*, 1999.

[BUY 99a] BUYYA R., *High Performance Cluster Computing, Volume 1: Architectures and Systems*, P T R Prentice-Hall, Englewood Cliffs, NJ 07632, USA, 1999.

[BUY 99b] BUYYA R., *High Performance Cluster Computing, Volume 2: Programming and Applications*, P T R Prentice-Hall, Englewood Cliffs, NJ 07632, USA, 1999.

[CLE 00] CLEMENTS A., *The Principles of Computer Hardware (third Edition)*, Oxford University Press, 2000.

[DAV 98] DAVID CULLER J. S., GUPTA A., *Parallel Computer Architecture: A Hardware/Software Approach*, Morgan Kaufmann Publishers, 1998.

[HEN 02] HENNESSY J., PATTERSON D., *Computer Architecture, A Quantitative Approach (third edition)*, Morgan Kauffmann, 2002.

[JÁJ 92] JÁJÁ J., *Introduction to Parallel Algorithms*, Addison Wesley, 1992.

[JIN 01] JIN H., CORTES T., BUYYA R., Eds., *High Performance Mass Storage and Parallel I/O: Technologies and Applications*, IEEE Computer Society Press and Wiley, New York, NY, 2001.

[KOR 97] KORFHAGE R. R., *Information Storage and Retrieval*, John Wiley & Sons, 1997.

[MAY 00] MAY J. M., *Parallel I/O for High Performance Computing*, Morgan Kaufmann Publishers, Los Altos, CA 94022, USA, 2000.

[MIL 94] MILLER R., REED J., The Oxford BSP Library : User's Guide., Report , Oxford University Computing Laboratory, 1994.

[NUT 01] NUTT G., *Operating Systems: A Modern Perspective, Lab Update*, Addison Wesley, 2001.

[STA 01] STALLINGS W., *Operating Systems: Internals and Design Principles, 4/e*, Prentice Hall, 2001.

[TAN 01] TANENBAUM A. S., *Modern Operating Systems, 2/e*, Prentice Hall, 2001.

[ULL 02] ULLMAN J. D., WIDOM J. D., *First Course in Database Systems, A, 2/e*, Prentice Hall, 2002.

[VAL 90] VALIANT L., "A Bridging Model for Parallel Computation", *Communications of the ACM*, , num. 33, p. 103-111, August 1990.