Contents

Foreword 13

Introduction 15

PART ONE. CLUSTER COMPUTING AND I/OS 21

Chapter 1. Motivating I/O Problems and their Solutions 23
Christophe Cérin
Université de Picardie Jules Verne, LaRIA, 5 rue du moulin neuf, 80000 Amiens, France

Hai Jin
Huazhong University of Science and Technology, Wuhan, 430074, China

1.1. Technological facts 24
1.2. Working at the driver level 24
1.3. Parallel programming and I/Os 27
1.3.1. The product of matrices on disk 28
1.3.2. MPI-2 primitives of data distribution management 28
1.3.3. A framework for a possible implementation of matrix product . . 30
1.3.4. Strassen algorithm 31
1.4. Sequential sorting with disks 34
1.4.1. Two-way merge sort 34
1.4.2. Balanced two-way merge sort 35
1.4.3. Balanced k-way merge sort 35
1.4.4. Polyphase merge sort 36
1.5. Tree data structures 38
1.5.1. B-tree and variants 40
1.5.1.1. Some data structures close to the B-tree 40
1.5.2. Buffer-tree and R-tree 42

1.5.3. Structures adapted in search of character strings 43
1.5.3.1. The structure of string B-tree 44
1.5.4. Conclusions about tree structures 44
1.6. I/O (automatic) prefetching 44
1.6.1. Paged virtual memory 45
1.6.2. Explicit I/Os 45
1.6.3. Prefetching in-core programs 46
1.6.4. Prefetching I/O operations 48
1.6.5. Prediction and prefetching 49
1.6.6. Prefetching and file buffer management 51
1.6.7. Prefetching and queued writing 52
1.6.7.1. Complementary definitions 52
1.6.7.2. An algorithm for the output schedule 54
1.6.7.3. Example 54
1.6.7.4. Some remarks and further work 56
1.6.8. Conclusion 58
1.7. Bibliography 58

Chapter 2. Parallel Sorting on Heterogeneous Clusters 63
Christophe Cérin
Université de Picardie Jules Verne, LaRIA, 5 rue du moulin neuf, 80000 Amiens, France

Hai Jin
Huazhong University of Science and Technology, Wuhan, 430074, China

2.1. Introduction 63
2.2. What is an heterogeneous cluster? 67
2.3. Related works about sorting on clusters 68
2.4. Sampling strategies for parallel sorting 69
2.5. A short classification of parallel in-core sorting algorithms 70
2.5.1. The regular sampling technique 71
2.5.2. A general framework for sorting on heterogeneous clusters 71
2.5.3. The spring of partitioning 73
2.5.4. The adaptation of the framework for external sorting 74
2.5.5. The main algorithm for sorting on heterogeneous clusters 75
2.5.6. Experimentation 77
2.5.7. Parallel Sample Sort (PSS) revisited 81
2.5.7.1. Introduction 81
2.5.7.2. Heterogeneous Sample Sort 82
2.5.7.3. Sample Sort with a performance vector configuration as an homogeneous cluster 83
2.5.8. Parallel sorting by overpartitioning revisited 84
2.5.8.1. Introduction 84

2.5.8.2. The heterogeneous case 85
2.5.8.3. Experiments 86
2.6. Conclusion 86
2.7. Bibliography 87

PART TWO. SELECTED READINGS 91

Chapter 3. A Sensitivity Study of Parallel I/O under PVFS 93
Jens Mache, Joshua Bower-Cooley, Robert Broadhurst, Jennifer Cranfill and Clark Kirkman

Lewis & Clark College, Portland, OR 97219, USA

3.1. Introduction 93
3.2. Related work 94
3.3. Background 94
3.3.1. Parallel I/O and parallel file systems 95
3.3.2. PVFS 95
3.4. Performance evaluation 96
3.4.1. Experimental configuration 96
3.4.2. Sensitivity to hardware 97
3.4.3. Sensitivity to configuration 97
3.4.3.1. Number of I/O nodes 100
3.4.3.2. Number of compute nodes 100
3.4.3.3. Number of I/O vs. compute nodes if cluster size is fixed . . 100
3.4.3.4. Overlapped I/O and compute nodes 101
3.4.4. Sensitivity to programming choices 102
3.4.4.1. Logical file view 102
3.4.4.2. Physical stripe size 102
3.4.4.3. Exploiting local data 102
3.5. Conclusions 102
3.6. Bibliography 103

Chapter 4. Effective I/O Bandwidth Benchmark 107
Rolf Rabenseifner*, Alice E. Koniges**, Jean-Pierre Prost*** and Richard Hedges**

**High-Performance Computing Center (HLRS), University of Stuttgart, Allmandring 30, D-70550 Stuttgart, Germany*

***Lawrence Livermore National Laboratory, Livermore, CA 94550, USA*

****IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA*

4.1. Introduction 107
4.2. Benchmark design considerations 109
4.2.1. Multidimensional benchmarking space 109

4.2.2. Criteria 111
4.3. Definition of the effective I/O bandwidth 112
4.4. Comparing systems using b_eff_io 114
4.4.1. Detailed insight 117
4.5. Discussion of b_eff_io 119
4.6. The time-driven approach 121
4.7. The influence of file hints 122
4.7.1. Support for file hints 123
4.7.2. Experimentation with IBM's MPI-IO/GPFS prototype 123
4.7.3. Results with hints enabled 127
4.8. Future work 129
4.9. Conclusion 129
4.10.Bibliography 130

Chapter 5. Parallel Join Algorithms on Clusters 133

Peter Kirkovits and Erich Schikuta

Institut für Informatik und Wirtschaftsinformatik, University of Vienna, Rathausstraße 19/9, A-1010 Vienna, Austria

5.1. Introduction 133
5.2. Parallel framework 135
5.2.1. Cluster system architecture 135
5.2.2. Declustering 135
5.3. Parallel joins 136
5.3.1. General information 136
5.3.2. General cost model 137
5.3.3. Grace hash-join 140
5.3.3.1. Cost model 141
5.3.4. Hybrid hash-join 143
5.3.4.1. Cost model 144
5.3.5. Nested loop join 145
5.3.5.1. Cost model 145
5.3.6. Sort merge join 147
5.3.6.1. Cost model 149
5.4. Performance analysis 150
5.5. Conclusions 153
5.6. Acknowledgements 154
5.7. Bibliography 154

Chapter 6. Server-side Scheduling 157
Robert B. Ross* and Walter B. Ligon III**

**Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, USA*

***Holcombe Department of Electrical and Computer Engineering, Clemson University, Clemson, SC 29634, USA*

6.1. Introduction 157
6.2. Server-side scheduling 158
6.2.1. Data sieving 159
6.2.2. Two-phase I/O 160
6.2.3. Disk-directed I/O 161
6.2.4. Server-directed I/O 161
6.2.5. Stream-based I/O 162
6.3. PVFS design 162
6.3.1. PVFS metadata 163
6.3.2. I/O daemons and data storage 163
6.3.3. PVFS request processing 165
6.3.3.1. Receiving and queuing requests 165
6.3.3.2. Servicing requests 166
6.3.4. Limitations 168
6.4. Scheduling algorithms 168
6.5. Workloads 171
6.6. Experimental results 173
6.6.1. Single block accesses 174
6.6.2. Strided accesses 174
6.6.3. Random block accesses 175
6.6.4. Fairness 175
6.7. Conclusions and future work 176
6.8. Bibliography 177

Chapter 7. A Large Virtual NVRAM Cache for Software RAID 183
Xubin He* and Qing Yang**

* *Department of Electrical and Computer Engineering, Tennessee Technological University, Cookeville, TN 38501, USA*

** *Department of Electrical and Computer Engineering, University of Rhode Island, Kingston, RI 02881, USA*

7.1. Introduction 183
7.2. Architecture and design of the virtual NVRAM cache 185
7.3. Implementation 188

7.3.1. In memory data structure . . . 188
7.3.2. Cache disk organization . . . 190
7.3.3. Basic operations . . . 191
7.3.3.1. Write . . . 191
7.3.3.2. Read . . . 191
7.3.3.3. Destages . . . 191
7.3.4. Reliability analysis . . . 192
7.4. Performance evaluations . . . 193
7.4.1. Experimental setup . . . 193
7.4.2. Benchmarks . . . 194
7.4.2.1. ServerBench 4.1 . . . 194
7.4.2.2. Bonnie . . . 195
7.4.2.3. PostMark . . . 195
7.4.2.4. Untar/Copy/Remove . . . 195
7.4.3. Numerical results and discussions . . . 196
7.4.3.1. Transactions Per Second (TPS) . . . 196
7.4.3.2. Response times . . . 198
7.4.3.3. Bonnie throughput . . . 200
7.4.3.4. PostMark throughput . . . 202
7.5. Related work . . . 203
7.6. Concluding remarks . . . 204
7.7. Bibliography . . . 205

PART THREE. APPENDICES . . . 209

Appendix 1. Matrix Product MPI-2 Codes . . . 211

Christophe Cérin
Université de Picardie Jules Verne, LaRIA, 5 rue du moulin neuf, 80000 Amiens, France

Hai Jin
Huazhong University of Science and Technology, Wuhan, 430074, China

Appendix 2. Selected Web Sites Related to I/O . . . 235
Christophe Cérin
Université de Picardie Jules Verne, LaRIA, 5 rue du moulin neuf, 80000 Amiens, France

Hai Jin
Huazhong University of Science and Technology, Wuhan, 430074, China

Index . . . 239

List of Figures

1.1 Architecture of a PC card 25

1.2 Activity of buses with two concurrent readings 26

1.3 Optimized concurrent readings 26

1.4 Experimentation: direct access versus normal access 27

1.5 Matrix product 29

1.6 The abstract type “file” 30

1.7 B-tree layout 40

1.8 Example of re-balancing operations in a B-tree 41

1.9 Representation of an R-tree 42

1.10 Representation of a Buffer-tree 43

1.11 Access pattern used in the example 49

1.12 Steps of data structure construction for example in Figure 1.11 50

1.13 Trace execution of suggested example 55

1.14 Foata Normal Form: schedule of suggested example 59

2.1 See also [VIT 94a]. Models of disks: (a) $P = 1$, for which D disks are connected to a common processor (b) $P = D$, for which D disks are connected to a different processor. This organization corresponds to the cluster organization 64

2.2 Example of PSRS execution [SHI 92] 72

2.3 Springs of partitioning on heterogeneous clusters 74

3.1 Two compute nodes sending a total of 6 logical blocks of data to two I/O nodes 96

3.2 Read performance (top) and write performance (bottom) for IDE disks and Fast Ethernet 98

3.3 Read performance (top) and write performance (bottom) for SCSI disks and Gigabit Ethernet 99

3.4 Completion time of ray-tracing application depending on number of I/O nodes and number of compute nodes 101

4.1 Access patterns used in b_eff_io. Each diagram shows the data accessed by **one** MPI-I/O write call 112

4.2 Comparison of b_eff_io for different numbers of processes on T3E and SP, measured partially without pattern type 3 115

4.3 Comparison of the results on SP, T3E, SR8000, and SX-5 116

4.4 128 nodes on the Blue Pacific RS 6000/SP with ROMIO 117

4.5 Comparison of b_eff_io for various numbers of processes at HLRS and LLNL, measured partially without pattern type 3. b_eff_io releases 1.x were used, except for the NEC system (rel. 0.6) 119

4.6 GPFS block allocation used by MPI-IO/GPFS in data shipping mode (using the default stripe size) 124

4.7 The effect of hints on the ASCI White RS 6000/SP testbed at LLNL . 126

5.1 Architectural framework of a cluster system 135

5.2 Theoretical Grace hash-join speed-up 142

5.3 Theoretical Grace hash-join scale-up 143

5.4 Theoretical Hybrid hash-join speed-up 145

5.5 Theoretical nested loop join speed-up 146
5.6 Theoretical nested loop join cost per tuple 147
5.7 Suboptimal sort merge join phase 148
5.8 Optimal sort merge join phase 148
5.9 Theoretical sort merge join speed-up 150
5.10 Theoretical sort merge join cost per tuple 151
5.11 Real Grace hash join speed-up 152
5.12 Real Hybrid hash join speed-up 153
5.13 Real sort merge join speed-up 153
5.14 Real nested loop join speed-up 154

6.1 Data sieving example 159
6.2 Example of metadata and file distribution 163
6.3 I/O stream example 164
6.4 File access example 165
6.5 Creating accesses from a request 166
6.6 Jobs in service 167
6.7 Test workloads 172
6.8 Single block read performance 179
6.9 Strided read performance 180
6.10 Random (32 block) read performance 181
6.11 Task service time variance for reads 182

7.1 Possible approaches to VCRAID. (a) one RAM buffer and one cache disk (b) one RAM buffer and several cache disks (c) one RAM buffer and several cache disks, each cache disk is associated with a disk in the array (d) Several RAM buffers and cache disks, each RAM buffer and cache disk are associated with a disk in the array 187

7.2 Procession of write requests 189

7.3 RAM buffer layout. RAM buffer consists of slot entries and slots. The hash table, LRU list and Free list are used to organize the slot entries 190

7.4 VCRAID vs. RAID5 and RAID 0 (Mean request size=1k) 197

7.5 scd vs. mcd (mean request size=0.5k) 198

7.6 Effect of RAM buffer size 199

7.7 Results of untar/copy/remove 201

7.8 Results for Bonnie Benchmark 202

List of Tables

2.1 Initial data layout with $N = 32, D = 4, B = 2$ 65

2.2 Configuration: 4 Alpha 21164 EV 56, 533Mhz - Fast Ethernet 78

2.3 External sequential sorting on our cluster (cf. Table 2.2) 79

2.4 External parallel sorting on cluster (see Table 2.2), message size: 32Kbyte, 15 intermediate files, 30 experiments 80

2.5 Heterogeneous Sample Sort (2MB of data, heterogeneous configuration of performance vector) 82

2.6 Heterogeneous Sample Sort (16MB of data, heterogeneous configuration of performance vector) 83

2.7 Heterogeneous Sample Sort (16MB of data, homogeneous configuration of performance vector) 84

2.8 Heterogeneous Sample Sort (2MB of data, homogeneous configuration of performance vector) 84

2.9 Heterogeneous PSOP (1973785 integers, heterogeneous configuration of performance vector) 86

2.10 Heterogeneous PSOP (16777215 integers, heterogeneous configuration of performance vector) 87

2.11 Summary of main properties of algorithms 88

4.1 Details of access patterns used in b_eff_io 113

4.2 Summary of the effect of hints on the total bandwidth on the ASCI White testbed . . . 127

4.3 b_eff_io without and with IBM_largeblock_io hint on the ASCI White testbed . . . 127

5.1 Basic parameters of the cost model . . . 138

5.2 Derived cost functions of the cost model . . . 139

5.3 Specific values of the basic parameters . . . 151

5.4 Values for derived functions of the cost model . . . 152

6.1 Example of *Opt 1* scheduling . . . 170

6.2 Example of *Opt 2* scheduling, starting with O_{last}=100 . . . 170

6.3 Example of *Opt 3* scheduling with W_{sz}=600, starting with O_{last}=100 170

6.4 Example of *Opt 4* scheduling, starting with O_{last}=100 . . . 171

7.1 Disk parameters . . . 193

7.2 PostMark results in terms of transactions per second . . . 203

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 10μs. 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.

Guest Editors:

Christophe Cérin	Hai Jin
Université de Picardie Jules Verne	Internet and Cluster Computing Center
LaRIA, bat Curi,	Vice-Dean, College of Computer
5, rue du moulin neuf	Huazhong University of Science
80000 AMIENS – France	and Technology
	Wuhan, Hubei, 430074 - China
Email: c.cerin@computer.org	Email: hjin@hust.edu.cn
URL: http://www.u-picardie.fr/˜cerin	http://www.hust.edu.cn/hjin/

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.