SOME KNOWN BOOK CITATIONS

Barry Wilkinson

October 19, 2008

Citations in "Introduction to Computational Science Modeling and Simulation for the Sciences" by Angela B. Shiflet and George W. Shiflet, Princeton University Press, 2006, Pages 446 and 467.

Page 446

3. The best sequential sorting algorithm that compares elements requires $n \log n$ computational steps. Suppose a sorting algorithm on a parallel system requires $4n$ computational steps. Determine the speedup factor (Wilkinson and Allen 1999).

Page 447

Wilkinson, Barry and Michael Allen. 1999. *Parallel Programming*. Upper Saddle River, N.J.: Prentice-Hall: 431.

Page 467

12. Consider the following generating function for a pseudorandom number generator:

$$r_n = (a\, r_{n-1} + c) \bmod m$$

For k processors, the parallel version is as follows:

$$r_n = (A\, r_{n-1} + C) \bmod m$$

where $A = a^k \bmod m$ and $C = c(1 + a + a^2 + \cdots + a^{k-1}) \bmod m$. Notice that the value of the coefficient is the same as with the version in the text, where $c = 0$. (Wilkinson and Allen 1999)

References

Page 470

Aik, Selim G. 1989. *The Design and Analysis of Parallel Algorithms*. Upper Saddle River, NJ: Prentice-Hall, Inc.: 401.

Andrews, Gregory R. 2000. *Foundations of Multithreaded, Parallel, and Distributed Programming*. Reading, MA: Addison-Wesley-Longman, Inc.: 664.

Miller, Russ, and Laurence Boxer. 2000. *Algorithms Sequential & Parallel, A Unified Approach*. Upper Saddle River, N.J.: Prentice-Hall: 330.

Wilkinson, Barry, and Michael Allen. 1999. *Parallel Programming*. Upper Saddle River, NJ: Prentice-Hall, Inc.: 431.

Technical Activities Forum

Cluster Computing: A High-Performance Contender

Mark Baker, University of Portsmouth
Rajkumar Buyya, Monash University
Dan Hyde, Bucknell University

When you first heard people speak of Piles of PCs, the first thing that came to mind may have been a cluttered computer room with processors, monitors, and snarls of cables all around. Collections of computers have undoubtedly become more sophisticated than in the early days of shared drives and modem connections. No matter what you call them—Clusters of Workstations (COW), Networks of Workstations (NOW), Workstation Clusters (WCs), Clusters of PCs (CoPs)—clusters of computers are now filling the processing niche once occupied by more powerful stand-alone machines.

In its simplest form, the computers in your office that are connected to your local area network constitute a workstation cluster. In addition to the hardware, a workstation cluster also includes the middleware that allows the computers to act as a distributed or parallel system and the applications designed to run on it.

While a system based on low-end workstations and network technologies may not at first seem particularly useful, such systems have been the testbeds for a new computing paradigm: high-performance and high-availability cluster computing. This class of system is becoming increasingly commonplace; in fact, most academic institutions and industries that use high-performance computing either already use or are thinking of using workstation clusters to run their most demanding applications. Even companies that can afford traditional supercomputers are becoming interested in commodity clusters.

Why the switch? For some, cluster-based systems provide a way to stretch their computing dollars, allowing the reuse of seemingly obsolete office or classroom systems. Others have found that a cluster of high-performance workstations can easily compete with the best supercomputers IBM or SGI have to offer. A company can download a few tools from a public Web site and order a collection of machines and network equipment to put together an 8-Gflops system for around $50,000. Assembling a powerful supercomputer would cost around $200,000.

TECHNOLOGIES, COMPONENTS, AND APPLICATIONS

A cluster consists of all the components found on any LAN with PCs or workstations: individual computers with their processors, memory, and disks; network

Cluster Computing Educational Resources

Although many educational institutions teach undergraduate and graduate students about the hardware and software components that make up a cluster, few courses or programs concentrate on the wealth of technologies that constitute the complete cluster environment, from hardware to application development tools. In order to introduce cluster computing into the curricula of more college programs, the Task Force on Cluster Computing has set up a Web site, http://www.coe.uncc.edu/~abw/parallel/links.html. This informative resource provides links to related journals, books, freely available software, projects from both academia and industry, white papers, and descriptions of hardware components. In addition, with our educational donation program, we actively support academic faculty members around the world who are interested in introducing new cluster-based courses by providing sample curricular materials.

With the generous cooperation of leading publishers worldwide, we have arranged for the donation of some current books on cluster computing. While the books will be available for faculty members who request them, the TFCC has reserved 50 percent for donation to academic programs in developing countries. The titles available include

- *High Performance Cluster Computing: Architectures and Systems*, R. Buyya (ed.), Prentice Hall, 1999
- *High Performance Cluster Computing: Programming and Applications*, R. Buyya (ed.), Prentice Hall, 1999
- *In Search of Clusters*, 2nd ed., G.F. Pfister, Prentice Hall, 1998
- *Metacomputing: Future Generation Computing Systems*, W. Gentzsch (ed.), Elsevier, 1999
- *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers*, B. Wilkinson and C.M. Allen, Prentice Hall, 1998

Technical Activities Forum coordinator: Deborah Scherrer, Stanford University, HEPL-4085, Stanford, CA 94305-4085; fax (650) 725-2333; deborah@quake.stanford.edu

Citations in "Structured Computer Organization, 4th edition" by Andrew S. Tanenbaum, Prentice Hall, pages 524 and 545.

p 524

deal of work has been done in the area of parallel architecture. In this chapter, we can barely scratch the surface. Additional information can be found in (Loshin, 1994; Pfister, 1998; Sima et al., 1997; and Wilkinson, 1994).

p. 545

software into account when designing the hardware. For a discussion about software for parallel computers, see (Wilkinson and Allen, 1999).

WILKINSON, B.: *Computer Architecture: Design and Performance, 2nd ed.*, Englewood Cliffs, NJ: Prentice Hall, 1994.

WILKINSON, B. and ALLEN, M.: *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers*, Upper Saddle River, NJ: Prentice Hall, 1999.

Citation in *Grid Computing: A Practical Guide to Technology and Applications*, Charles River Media, Massachusetts, 2004, page 123.

predisposition motivates four types of applications (Figure 7.2) revealed by exploring process granularity. *Granularity* refers to the size of a computation that can be performed between communication or synchronization points.[7] Thus, any point on the vertical axis of Figure 7.2 identifies a specific ratio of computation (increasing from bottom to top) to communication (increasing from top to bottom).[8] Task parallelism, increasing from left-to-right on the horizontal axis, refers to the degree of parallelism present in the application. "Fine" through "Coarse" are used as qualitative metrics, as shown in the figure.

Most scientific problems are implemented initially as *serial* applications (Figure 7.2, Quadrant II[9]). These problems require that each step of the scientific calculation be performed in sequence. Serial applications can be executed on compute architectures ranging from isolated desktops, servers, or supercomputers to compute farms. Compute farms are loosely coupled compute architectures in which system software is used to virtualize compute servers[10] into a single system environment (SSE).

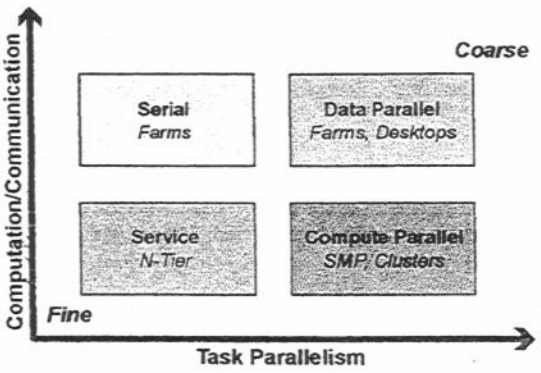

FIGURE 7.2 Applications and architectures.

[7] B. Wilkinson & M. Allen. *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers*, Prentice Hall, Upper Saddle River, NJ, 1999.

[8] If the interest is computing, then communication is viewed as the "overhead" required to achieve the computation. Similarly, computation might be regarded as the overhead required to facilitate certain communications.

[9] The mathematical convention of numbering quadrants counter-clockwise from the upper-right-hand corner is used here.

[10] Although high-density, rack-mounted single/dual processor servers have been used in compute farms, there is an intensifying trend toward the use of higher-density blade servers in these configurations.

Citations in "Introduction to Parallel Computing 2nd edition" by A. Grama, A. Gupta, G. Karypis, and V. Kumar, Addison-Wesley, 2003, pages 9 and 142.

p 9

A number of texts discuss paradigms and languages for programming parallel computers [LB98, Pac98, GLS99, GSNL98, CDK+00, WA98, And91, BA82, Bab88, Ble90, Con89, CT92, Les93, Per87, Wal91]. Akl [Akl97], Cole [Col89], Gibbons and Rytter [GR90], Foster [Fos95], Leighton [Lei92], Miller and Stout [MS96], and Quinn [Qui94] discuss various aspects of parallel algorithm design and analysis.

p 142

Various texts, such as those by Wilson [Wil95], Akl [Akl97], Hwang and Xu [HX98], Wilkinson and Allen [WA99], and Culler and Singh [CSG98], among others, present similar or slightly varying models for parallel programs and steps in developing parallel algorithms.

[WA98] B. Wilkinson and C. M. Allen. *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers*. Prentice Hall, 1998.

[WA99] B. Wilkinson and M. Allen. *Parallel Programming*. Prentice-Hall, NJ, 1999.

Citations in "Foundations of Multithreaded, Parallel, and Distributed Programming" by Gregory R. Andrews, Addison Wesley, 2000, pages 33, 472, and 583.

p. 33

A recent book by Wilkinson and Allen [1999] describes dozens of applications and shows how to write parallel programs to solve them; the book emphasizes the use of message passing, but includes some coverage of shared-variable computing.

p. 472

lems. Examples can be found in most books on parallel computing. A few good sources are Brinch Hansen [1995], Fox et al. [1988], Quinn [1994], and Wilkinson and Allen [1999]. Hardware pipelines are covered in detail in Hwang [1993]. The behavior of both software and hardware pipelines is similar, so their performance can be analyzed in similar ways.

p. 583

Wilkinson and Allen [1999] describe how Jacobi iteration can be represented as a system of linear equations, and they include chapters on image processing and searching and optimization.

Wilkinson, B., and M. Allen. 1999. *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers*. Upper Saddle River, NJ: Prentice-Hall.

Editorial, *The Computer Journal*, Vol. 35, No. 1, 1992, page 2.

mesh or a hypercube. More complex designs of multistage interconnection networks[18,25] can sometimes be valuable.

25. B. Wilkinson, Comparative performance of overlapping connectivity multiprocessor interconnection networks. *The Computer Journal* 34 (3), 207–214 (1991).

"Computer Architecture" by R. J. Baron and L. Higbie, Addison-Wesley, 1992, page 458.

Wilkinson, B. (1987) *Digital System Design*. Englewood Cliffs, N.J.: Prentice-Hall International.

Editorial, *Informatica*, vol 23, 1999, page 2.

[6] B. Wilkinson and M. Allen. *Parallel Programming Techniques and Applications Using Networked Workstations and Parallel Computers*. Prentice Hall, NJ, 1999.

"Theory and Design of Digital Computer Systems 2nd edition" by D. Lewin and D. Noaks, Chapman and Hall, 1995, page 30.

2. Wilkinson, B.R. (1987) *Digital System Design*. Prentice-Hall International (UK).

In "Digital Logic Design" by G. Langholz, A. Kandel, and J. L. Mott (Florida State University), Wm. C. Brown Publishers, 1988, Bibliography, page 463.

Wilkinson, B. *Digital System Design*. London: Prentice-Hall International, 1987.

In "VLSI-Systeme" by A. Möschwitzer and F. Rößler, VEB Verlag Technik, Berlin, 1988, Literaturverzeichnis, page 191. Referenced on pages 29 and 44.

Literaturverzeichnis

[3.2] *Wilkinson, B.*: Digital system design. London: Prentice Hall, 1986.

p.29 Als Maß für die Einschaltverzögerung wollen wir die Zeit t_{d1} definieren, die benötigt wird, um die Kapazität auf die Hälfte ($U_{DD}/2$) der Betriebsspannung zu entladen. Gemäß den Grundlagen der Elektrotechnik [3.2] gilt dafür

$$t_{d1} = C_L R_D \ln 2. \qquad (3.5)$$

p.44 inversion, punch through) sehr zu und kann zu völlig anderen Ergebnissen führen. Es sind dann statt (3.19) bis (3.21) wesentlich kompliziertere Modelle (10- bis 15-Parameter-Modelle) zu verwenden [3.2]. Der Entwerfer hat in jedem Fall vor der Simulation zu entscheiden, welches Transistormodell er aus einer zur Verfügung stehenden Modellbibliothek verwenden muß, um ausreichende Genauigkeit seiner Simulation zu erreichen.

"The Principles of Computer Hardware" by A. Clements, Oxford University Press, 1985, page 436.

Wilkinson, B., and Horrocks, D. (1982). *Computer peripherals*. Hodder and Stoughton. Devoted entirely to input and output devices, backing stores, and (in less detail) computer communications. All these topics are treated fully at an introductory level. This book is strongly recommended to those who wish to learn more about the structure, operation, and characteristics of peripherals.

"Computer Design and Architecture" by L. H. Pollard, Prentice Hall, 1990, pages 68, 127, 338, and 401.

[Wilk87] Wilkinson, B., *Digital System Design*. Englewood Cliffs, NJ: Prentice Hall International, 1987.

Citations using equations derived in journal papers:

Citation in "Cost-Performance Analysis of Cascaded Crossbar Interconnection Multiprocessor" by C Evéquoz, *IEE Proc. Comput. Dist. Tech.*, Vol. 142, No. 2, 1995, page 117. 369

(A paper taking my interconnection network and developing quite complex mathematics proving simulation results.)

1 Introduction

The overall performance of multiprocessor systems is directly related to the efficiency of interconnection networks (INs) that allow processors to access shared resources such as memory banks and input/output devices. Many different INs have been proposed, such as crossbar [1, 2], multistage INs, single and multiple bus [3–6], and others [7].

A major disadvantage of these networks is that they do not take advantage of the locality of communication present in most applications. Tasks exchanging large volumes of communication should be grouped together into cluster of processors with low communication costs. Such a feature proves to be attractive in various application areas such as image processing, neurocomputing, and parallel simulation. Recently, Wilkinson [8] proposed an architecture based on cascaded crossbar INs to exploit this locality. In this architecture, P processors and M memory modules are divided into G stages with each stage connected to two neighbouring stages via crossbar switches. The connection topology may then be a linear array, or unidirectional or bidirectional ring with cost $O(PM/G)$, which is comparable to multistaged INs [8]. Tasks may be assigned to processors in various stages at compile time in a way that preserves the locality of communication and memory reference, i.e., the interaction between the stages decreases as the distance between the stage increases.

We present a queueing model for analysing cascaded crossbar architectures organised in bidirectional rings and operating under synchronous packet switching. Simulation results are employed to estimate the errors resulting from the various assumptions and approximations introduced in the analytical solution. The impact of the locality of reference and comparisons of various cascaded networks configurations are also investigated. It is shown that networks with a higher number of stages have better cost-performance ratios than those with fewer stages. This comparison is done with a fixed number of processors and memory modules.

8 WILKINSON, B.: 'Cascaded rhombic crossbar interconnection networks', *J. Parallel Distrib. Comp.*, 1990, **10**, (1), pp. 96–101

Citations in "Computer Architecture; Pipelined and Parallel Processor Design" by Michael J. Flynn (Stanford University), Jones and Bartlett Publishers, 1995, pages 413 and 552.

p. 413

(2) Bus model with request resubmission.

This is a more complex analysis, and requires an iterative solution. There are several solutions [139. 204. 205, 307, 316], each providing (roughly) the same result.

p. 552

bus bandwidth over that expected by the above analysis. As developed in chapter 6, we estimate the actual bus requests (a) per source [307] as:

$$a = \frac{\rho}{\rho + (\rho_a/\rho)(1-\rho)},$$

and then

$$B(n) = n\rho_a = 1 - (1-a)^n.$$

[307] B. Wilkinson. Comments on "Design and analysis of arbitration protocols". *IEEE Transactions on Computers*, 41(3):348-351, March 1992.