Cost-Effective Parallel Computing*†

David A. Wood and Mark D. Hill

Computer Sciences Department
University of Wisconsin–Madison
1210 West Dayton Street
Madison, WI 53706 USA
{david,markhill}@cs.wisc.edu

Abstract

Many academic papers imply that parallel computing is only worthwhile when applications achieve nearly linear speedup (i.e., execute nearly p times faster on p processors). This note shows that parallel computing is cost-effective whenever speedup exceeds *costup*—the parallel system cost divided by uniprocessor cost. Furthermore, when applications have large memory requirements (e.g., 512 megabytes), the costup—and hence speedup necessary to be cost-effective—can be much less than linear.

Categories and Subject Descriptors: C.1.2 [**Processor Architectures**]: Multiple Data Stream Architectures (multiprocessors); C.4 [**Computer Systems Organization**]: Performance of Systems; K.6 [**Management of Computing and Information Systems**]: Installation Management—*pricing and resource allocation.*

Introduction

Suppose that you need to run many simulations that require large amounts of memory. You may run the simulations on a uniprocessor or a p-processor parallel system. You know that your simulations cannot be parallelized perfectly, so speedups will be less than linear. Parallel simulation will reduce response time, but your task is to maximize job throughput per unit cost, or equivalently, to minimize cost-performance (cost divided by performance).

Which system do you select? Conventional wisdom says use the uniprocessor, since speedups are less than linear[1]. We show, however, that the parallel system provides better (i.e., lower) cost-performance whenever speedup exceeds *costup*—the parallel system cost divided by uniprocessor cost. Our result is not tied to simulation, but holds for all applications.

Furthermore, we find that when applications have large memory requirements (e.g., 512 megabytes), the costup—and hence speedup necessary to be cost-effective—can be much less than linear. This is because the parallel system does not need p times the memory of the uniprocessor, since parallelizing a job rarely multiplies its memory requirements by p.

Three decades ago Amdahl argued that each million-instructions-per-second (MIPS) of processing power should be accompanied by 1 megabyte of memory [4] (p. 17). Our results can be interpreted as the converse of Amdahl's dictum: Each 1 megabyte of memory should be accompanied by 1 MIPS of processing power. If one processor does not provide enough power, multiple processors should be used to make effective use of the memory's capacity and bandwidth.

*__To the referees:__ We intend this to be a short, pointed note in the tradition of Gustafson's "Reevaluating Amdahl's Law" (CACM May '88, pp. 532-533) or J. Smith's "Characterizing Computer Performance with a Single Number" (CACM Oct '88, pp. 1202-1206).

†This paper generalizes the cost model introduced by Falsafi and Wood (Cost/Performance of a Parallel Computer Simulator, in Proceedings of PADS '94, July 1994). This work is supported in part by Wright Laboratory Avionics Directorate, Air Force Material Command, USAF, under grant #F33615-94-1-1525 and ARPA order no. B550, NSF PYI Awards MIP-8957278 and CCR-9157366, NSF Grant MIP-9225097, and donations from A.T.&T. Bell Laboratories, Digital Equipment Corporation, Sun Microsystems, Thinking Machines Corporation, and Xerox Corporation. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Wright Laboratory Avionics Directorate or the U.S. Government.

[1]Alternatively, you could use p uniprocessors to increase throughput and yet retain the same cost-performance as one uniprocessor—p times the cost divided by p times the performance.

Speedup & Costup

To formalize our results, let the time to execute a job with p processors be $time(p)$. Parallel system performance is often characterized using *speedup*:

$$speedup(p) = \frac{1/time(p)}{1/time(1)} = \frac{time(1)}{time(p)}.$$

Let the cost with p processors be $cost(p)$. The cost could be just the hardware cost (for processors, memory, I/O devices, backplanes, power supplies, etc.) or include software costs (the costs of building the parallel application and system software, amortized over their expected lifetime). Analogous to speedup, we introduce *costup* to characterize parallel system cost:

$$costup(p) = \frac{cost(p)}{cost(1)}.$$

To determine the cost-effectiveness of a system, performance and cost are often combined to get *cost-performance*:

$$cost\text{-}performance(p) = \frac{cost(p)}{1/time(p)}.$$

Parallel computing is more cost-effective whenever its cost-performance is better (smaller) than a uniprocessor's:

$$cost\text{-}performance(p) < cost\text{-}performance(1).$$

Substitution reveals our principal result that parallel computing is more cost-effective whenever:

$$speedup(p) > costup(p).$$

Our result is true, in general, and does not depend on the assumptions we make below to calculate specific values. What constitutes cost depends upon one's point of view. A computer vendor may see costs as the sum of research and development, components, manufacturing, and advertising, while a customer may view cost as purchase price.

While this theoretical result is interesting, it has practical impact when costups are less than linear. We show below that this happens when memory dominates system cost.

Remember Memory

Memory is an important component in the hardware costs of today's machines. Assume that our job requires m megabytes on a uniprocessor and m' megabytes with p processors. (If virtual memory is used, m and m' are determined by the job's working set requirements rather than by the maximum memory referenced.) Usually m' is larger than m to permit the replication of some application or operating system code or data structures, or because parallel working sets are larger. When m is small or p is large, m' may be much larger than m, because m/p puts too few memory chips with each processor to adequately satisfy the processor's bandwidth requirements. Consider a processor that needs an 8-byte datapath to each of two interleaved banks. If the memory is implemented with 4 megaword-by-4 bit dynamic RAMs, the minimum memory size per processor is 64 megabytes (4 megaword $\cdot$ 8 bytes per bank $\cdot$ 2 banks).

Usually, however, significant memory cost will tend to make costups less than linear. This is because a parallel system does not need p times the memory of the uniprocessor, since parallelizing a job rarely multiplies its memory requirements by p (i.e., $m' \ll p \cdot m$). We can emphasize this in new speedup and costup equations:

$$speedup(p, m, m') = \frac{time(1, m)}{time(p, m')},$$

$$costup(p, m, m') = \frac{cost(p, m')}{cost(1, m)}.$$

Parallel computing is more cost-effective when:

$$speedup(p, m, m') > costup(p, m, m').$$

But how does memory affect real costups?

A Multi Example

As a concrete example, we use current Silicon Graphics (SGI) prices to show that actual costups can be much less than linear for systems with hundreds of megabytes of main memory. We consider hardware costs, but not software ones since we do not know how to non-controversially measure the latter. All prices are list prices in U.S. dollars as of July 15, 1994 [5]. We ignore the volume discounts that may favor uniprocessors. Since we take the ratio of two list prices, our quantitative results also hold exactly when a vendor gives a customer the same discount on both systems.

Silicon Graphics products range from low-cost desktop workstations to million-dollar shared-memory multiprocessors. We focus on their server products, so our comparison will not be biased by expensive graphics engines and monitors. At the low-end, the Silicon Graphics CHALLENGE S is a highly-competitively-priced monitor-less uniprocessor workstation, with a list price of \$16,600. However, because it is packaged as a small desktop unit, the CHALLENGE S has a maximum memory size of 256 megabytes. While 256 megabytes

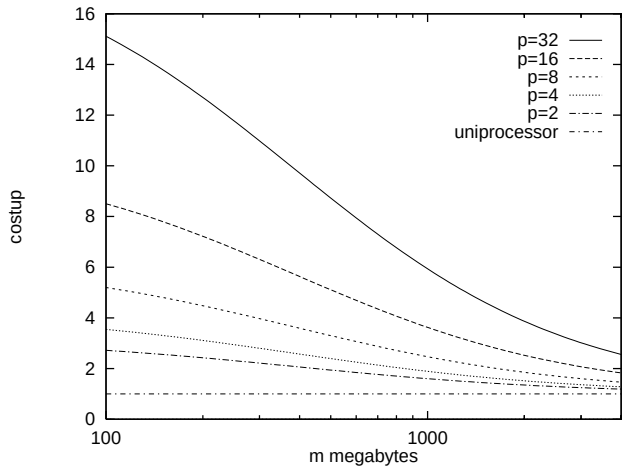

Figure 1: SGI costups with no memory overhead

Parallel computing is more cost-effective when speedups exceed the $costup(p, m, m')$ for p processors (different lines) and memory size m megabytes (x-axis). This graph assumes no memory overhead ($m' = m$). The "uniprocessor" line represents a uniprocessor with degenerate costup of 1.

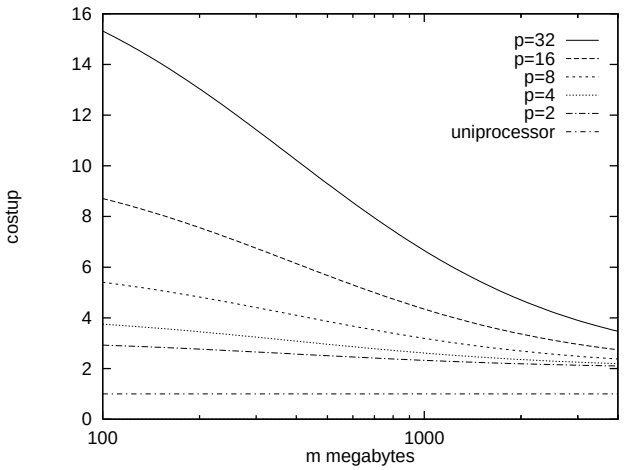

Figure 2: SGI costups with 100% memory overhead

Parallel computing—even if it uses 100% more memory ($m' = 2m$)—is more cost-effective when speedups exceed the $costup(p, m, m')$.

is sufficient for many computations, it is far too small for many of the large and long running applications we might want to parallelize.

To achieve larger memory capacity requires purchasing a deskside configuration, such as the CHALLENGE DM. These deskside units can support upto 6 gigabytes of physical memory, but at a significant premium: a uniprocessor CHALLENGE DM lists for about $38,400 plus about $100 per megabyte. This results in a uniprocessor cost of:

$$cost(1, m) = \$38400 + \$100 \cdot m.$$

For comparison, we use the Silicon Graphics CHALLENGE XL as the parallel system[2]. The CHALLENGE XL is a rack-mounted bus-based multiprocessor that supports 2 to 40 processors with a cost that closely follows:

$$cost(p, m') = \$81600 + \$20000 \cdot p + \$100 \cdot m'.$$

Substitution reveals:

$$costup(p, m, m') = \frac{2.125 + 0.521 \cdot p + 0.0026 \cdot m'}{1 + 0.0026 \cdot m}.$$

Figure 1 illustrates costups with SGI prices and the optimistic assumption that parallel computing requires no additional memory ($m' = m$). Different lines represent the number of processors p, while the x-axis gives the memory size m in megabytes. The data supports our principal result:

> With real price data, parallel computing can be more cost-effective at speedups much less than p for large but practical memory sizes.

For systems requiring 512 megabytes, for example, 8-, 16-, and 32-processor systems are more cost-effective than a uniprocessor when speedups exceed 3.3, 5.0, and 8.6, respectively. These speedups correspond to *efficiencies*—$speedup(p, m)/p$—of only 0.41, 0.32, and 0.27. While 512 megabytes may sound like a lot of memory for a uniprocessor, it is only 64, 32, and 16 megabytes per processor for 8-, 16-, and 32-processor systems.

But what happens when parallel computing requires more memory than using a uniprocessor? Figure 2 illustrates costups with 100% memory overhead; that is, $m' = 2 \cdot m$. Our principal result is qualitatively unchanged: parallel computing can still be cost-effective at speedups much less than linear. When memory is small, doubling parallel memory cost has little effect. When it is large, costups approach 2.0 instead of 1.0, but are still much less than linear.

[2] This comparison is somewhat biased towards the uniprocessor, since the CHALLENGE DM uses a 100MHz R4400 processor rather than the 150MHz R4400 processor of the CHALLENGE XL. Silicon Graphics does not currently sell a uniprocessor deskside unit with the faster processor.

More Generally

While SGI costups are interesting, we can generalize the result using a simple hardware cost model:

$$cost(1, m) = f(1) + g(m),$$
$$cost(p, m') = f(p) + g(m'),$$

where g is memory cost and f is the cost of everything else (e.g., processor(s), disks, backplane, power supply), normalized so that $f(1) = 1$. This model assumes that memory costs the same in a uniprocessor or a parallel system of any size. While this assumption seems reasonable given current technologies, marketing considerations can make parallel system memory more expensive [3]. Using this model, costup is:

$$costup(p, m, m') = \frac{f(p) + g(m')}{1 + g(m)}.$$

If memory costs are negligible, then costup is $f(p)$. The value of $f(p)$ can be less than p if there is a significant fixed cost for both a uniprocessor and a parallel system. On the other hand, $f(p)$ can be more than p if the fixed cost for a uniprocessor is much less than the fixed cost for a parallel system or the interconnection network constitutes a large part of the parallel system cost. When $f(p) > p$, a parallel system cannot be cost-effective (even with linear speedups) until memory costs become significant.

Our principal result, however, is manifest when the memory costs are significant. When memory cost dominates, the costup approaches $g(m')/g(m)$. If $g(m)$ is proportional to m, then $g(m')/g(m) = m'/m$ and is likely to be much less than p. More importantly, costups can be small when even memory cost are significant but not dominant (e.g., if memory is half the uniprocessor's cost, $g(m) = 1.0$).

This result may come as a surprise to those who define parallel system efficiency with $speedup(p)/p$. With this definition, "efficiency" is maximized at 1.0 when $p = 1$. Why then do we find parallel systems—with even modest speedups—to be more "efficient"? The explanation is that $speedup(p)/p$ is processor-centric: it measures the utilization of processors but ignores memory. Our results show that when memory is sufficiently large (and expensive), more than one processor should be used to make effective use of the memory capacity and bandwidth. This result may also call into question the wisdom of time-sharing large-memory jobs without considering memory-processor interaction metrics like the space-time product [1].

Related Work

Few papers address the cost-effectiveness of parallel computing. Fuller [3] compared the multiprocessor CMU C.mmp (based on the DEC PDP 11/20 and 11/40 processors) with the uniprocessor DEC PDP-10. He found C.mmp to be three to four times more cost-effective; however, his results were dependent upon the specific processor and (differing) memory costs of these systems.

Falsafi and Wood [2] investigated the cost-effectiveness of the Wisconsin Wind Tunnel (WWT) parallel simulator. WWT runs on a Thinking Machines CM-5 (the host), but models the processors and memories of alternative cache-coherent shared-memory machines (the targets) with enough detail to run target executables. Falsafi and Wood found that WWT is more cost-effective than uniprocessor simulations for studying large target systems (e.g., 32 or more nodes), because those runs demand vast host memory. Our work generalizes their result.

Conclusions

This paper compared the cost-performance of a uniprocessor and a parallel system for maximizing throughput. We found that parallel computing is cost-effective whenever speedup exceeds *costup*—the parallel system cost divided by uniprocessor cost. Furthermore, when applications have large memory requirements (e.g., 512 megabytes), the costup—and hence speedup necessary to be cost-effective—can be much less than linear. Intuitively, when memory is sufficiently large (and expensive), more than one processor may be needed to efficiently utilize the memory.

Amdahl argued that each MIPS of processing power should be accompanied by 1 megabyte of memory. We find the converse: Each 1 megabyte of memory should be accompanied by 1 MIPS of processing power. If one processor does not provide enough power, multiple processors should be used to balance the memory's capacity and bandwidth.

Acknowledgements

This work was performed as part of the Wisconsin Wind Tunnel project, which is co-led by Profs. Mark Hill, James Larus, and David Wood (`http://www.cs.wisc.edu/p/wwt/Mosaic/wwt.html` or `ftp ftp.cs.wisc.edu; cd wwt`). Babak Falsafi was a co-author of the paper that inspired this work [2]. Viranjit Madan provided SGI price data. Doug Burger, Babak Falsafi, Jim Goodman, Shubu Mukherjee, David Nicol,

Anne Rogers, Guri Sohi, and Jim Smith gave valuable feedback.

References

[1] Peter J. Denning. Virtual Memory. *ACM Computing Surveys*, 2(3):153–189, September 1970.

[2] Babak Falsafi and David A. Wood. Cost/Performance of a Parallel Computer Simulator. In *Proceedings of PADS '94*, July 1994.

[3] Samuel H. Fuller. Price/Performance Comparison of C.mmp and the PDP-10. In *Proc. Third International Symposium on Computer Architecture*, pages 195–202, January 1976.

[4] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach.* Morgan Kaufmann, 1990.

[5] Ed Reidenbach. *CHALLENGE Server Perdiodic Table.* Silicon Graphics Computer Systems, October 1993.