Cost-Effective Parallel Computing

David A. Wood and Mark D. Hill
University of Wisconsin, Madison

Many academic papers imply that parallel computing is only worthwhile when applications achieve nearly linear speedup (that is, execute p times faster on p processors). We show, to the contrary, that parallel systems—multiprocessor workstations, networks of workstations, and even massively parallel processors—can be cost-effective at modest speedups when memory cost is a significant fraction of system cost.

Consider the following example. Suppose that you need to run many simulations that each require large amounts of memory. You may run the simulations on a uniprocessor or a p-processor parallel system. You know that your simulation cannot be parallelized perfectly, so speedups will not be linear. Parallel simulation will reduce response time, but your task is to select the system that maximizes 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 multiprocessor speedups are less than linear. 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. We show, however, that the parallel system provides better (that is, 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, when applications have large memory requirements (for example, 512 Mbytes), the costup—and hence the speedup necessary to be cost-effective—can be far less than linear. This is because the parallel system does not need p times the uniprocessor memory, since parallelizing a job rarely multiplies its memory requirements by p.

Three decades ago, Gene Amdahl argued that each million instructions per second (MIPS) of processing power should be accompanied by 1 Mbyte of memory.[1] Our results can be interpreted as the converse of Amdahl's maxim: Each 1 Mbyte 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.

Large memories can make parallel computing cost-effective even with modest speedups.

SPEEDUP AND 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:

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

Speedups are called linear when speedup(p) $\approx p$.

0018-9162/95/$4.00 © 1995 IEEE

Let the cost with p processors be cost(p). The cost can be only the hardware cost (for processors, memory, I/O devices, backplanes, power supplies, and so forth) or can include software costs (the costs of building the parallel application and system software, amortized over their expected lifetime). Software life-cycle costs are very important if each new system requires significant software updates, as are common in today's parallel systems. Analogous to speedup, we introduce *costup* to characterize parallel system cost: costup(p) = cost(p)/cost(1).

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

$$\text{costperf}(p) = \frac{\text{cost}(p)}{1/\text{time}(p)}$$

Parallel computing is more cost-effective whenever its cost/performance is better (smaller) than a uniprocessor's: *costperf*(p) < *costperf*(1). Substitution reveals our principal result:

> Parallel computing is more cost-effective whenever speedup(p) > costup(p).

Our result does not depend on the assumptions we make below to calculate specific values. What constitutes cost depends on 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.

This theoretical result has practical impact when costups are less than linear. We show below that this happens when memory is a significant fraction of system cost.

REMEMBER MEMORY

Memory is an important component in the hardware costs of today's machines. Assume that our job requires m Mbytes on a uniprocessor and m' Mbytes 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 partial replication of the application's code, an operating system's code, and their 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 data path 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 Mbytes (4 megaword × 8 bytes per bank × 2 banks).

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

$$\text{speedup}(p, m, m') = \text{time}(1, m)/\text{time}(p, m'),$$
$$\text{costup}(p, m, m') = \text{cost}(p, m')/\text{cost}(1, m)$$

Parallel computing is more cost-effective when:

$$\text{speedup}(p, m, m') > \text{costup}(p, m, m')$$

But how does memory affect real costups?

A MULTIPROCESSOR EXAMPLE

As a concrete example, we use 1994 Silicon Graphics (SGI) prices to show that actual costups can be far less than linear for systems with hundreds of Mbytes of main memory. We consider hardware costs but not software costs, since we do not know how to noncontroversially measure the latter. All prices are list prices in US dollars (as of July 15, 1994).[2] 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 that our comparison will not be biased by expensive graphics engines and monitors. At the low-end, the Silicon Graphics Challenge S is a very competitively priced monitorless 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 Mbytes. While 256 Mbytes 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 up to 6 gigabytes of physical memory, but at a significant premium: a uniprocessor Challenge DM lists for about $38,400 plus about $100 per Mbyte. This results in a uniprocessor cost of:

$$\text{cost}(1, m) = \$38{,}400 + \$100 \times m$$

For comparison, we use the Silicon Graphics Challenge XL as the parallel system. (This comparison is somewhat biased in favor of the uniprocessor, since the Challenge DM uses a 100-MHz R4400 processor rather than the 150-MHz R4400 processor of the Challenge XL. Silicon Graphics does not currently sell a uniprocessor deskside unit with the faster processor.) The Challenge XL is a rack-mounted, bus-based multiprocessor that supports two to forty processors with a cost that closely follows

$$\text{cost}(p, m') = \$81{,}600 + \$20{,}000 \times p + \$100 \times m'$$

Substitution reveals:

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

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 log-scale x-axis gives

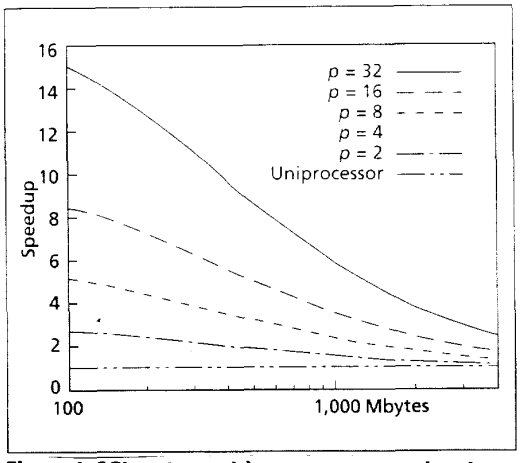

Figure 1. SGI costups with no memory overhead ($m' = m$).

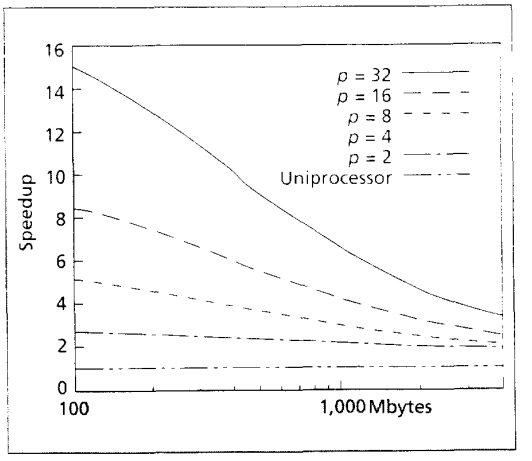

Figure 2. SGI costups with 100-percent memory overhead ($m' = 2 \times m$).

the memory size m in Mbytes. The uniprocessor line represents a uniprocessor with degenerate costup of one. 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 example, in systems requiring 512 Mbytes, 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. Although 512 Mbytes may sound like a lot of memory for a uniprocessor, it is only 64, 32, and 16 Mbytes per processor for 8-, 16-, and 32-processor systems, respectively.

But what happens when parallel computing requires more memory than a uniprocessor does? Figure 2 illustrates costups with 100-percent memory overhead; that is, $m' = 2 \times m$. Our principal result is qualitatively unchanged: Parallel computing can still be cost-effective at speedups far from 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.

MORE GENERALLY

We can generalize the result using a simple hardware cost model:

$$\text{cost}(1, m) = f(1) + g(m),$$
$$\text{cost}(p, m') = f(p) + g(m')$$

where g is memory cost and f is the cost of everything else (for example, 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,

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

If memory costs are negligible, 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 if 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 even when memory costs are significant but not dominant. When memory in each system is half the uniprocessor's cost ($g(m) = g(m') = 1.0$, for example), costups are $f(p)/2 + 1/2$, or approximately half of $f(p)$ for large systems.

This result may surprise those who define parallel-system efficiency in the traditional way: 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.[4]

OUR RESULTS SHOW THAT PARALLEL COMPUTING can cost-effectively maximize throughput whenever speedup exceeds costup (the parallel system cost divided by

uniprocessor cost). Furthermore, when applications have large memory requirements (for example, 512 megabytes), the costup—and hence speedup necessary to be cost-effective—can be far less than linear.

Intuitively, when memory is sufficiently large (and expensive), more than one processor may be needed to efficiently utilize the memory. This result can be thought of as the converse of Amdahl's maxim: Rather than accompanying each 1 MIPS of processing power with 1 Mbyte of memory, we find that each 1 Mbyte 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.

This work indicates that parallel computing can be cost-effective. However, a broader view of cost should also include the software development costs needed to parallelize applications. Getting these costs under control is necessary for parallel computing to flourish. I

Acknowledgments

This work was performed as part of the Wisconsin Wind Tunnel Project, which is led by Mark Hill, James Larus, and David Wood (http://www.cs.wisc.edu/~wwt or ftp.cs.wisc.edu/wwt). Babak Falsafi was a coauthor of the paper that inspired this work. 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.

This work is supported in part by Wright Laboratory Avionics Directorate, Air Force Material Command, US Air Force, 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 AT&T Bell Laboratories, Digital Equipment Corporation, Sun Microsystems, Thinking Machines Corporation, and Xerox Corporation.

References

1. J.L. Hennessy and D.A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, Palo Alto, Calif., 1990, p. 17.
2. E. Reidenbach, *Challenge Server Periodic Table*, Silicon Graphics Computer Systems, Mountain View, Calif., 1994.
3. S.H. Fuller, "Price/Performance Comparison of C.mmp and the PDP-10," *Proc. Third Int'l Symp. Computer Architecture*, IEEE CS Press, Los Alamitos, Calif., Order No. 099 (microfiche only), 1976, pp. 195-202.
4. P.J. Denning, "Virtual Memory," *ACM Computing Surveys*, Vol. 2, No. 3, Sept. 1970, pp. 153-189.

RELATED WORK

Few papers address the cost-effectiveness of parallel computing. Fuller[1] compared the CMU C.mmp multiprocessor (based on the DEC PDP 11/20 and 11/40 processors) with the DEC PDP-10 uniprocessor. He found C.mmp to be three to four times more cost-effective; however, his results depended on 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. The 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 the WWT is more cost-effective than uniprocessor simulations for studying large target systems (32 or more nodes), because those runs demand vast host memory. Our work generalizes their result.

References

1. S.H. Fuller, "Price/Performance Comparison of C.mmp and the PDP-10," *Proc. Third Int'l Symp. Computer Architecture*, IEEE CS Press, Los Alamitos, Calif., Order No. 099 (microfiche only), 1976, pp. 195-202.
2. B. Falsafi and D.A. Wood, "Cost/Performance of a Parallel Computer Simulator," *Proc. Eighth Workshop Parallel and Distributed Simulation (PADS 94)*, IEEE CS Press, Los Alamitos, Calif., Order No. 6495-02, 1994, pp. 173-182.

David A. Wood *is an assistant professor in both the Computer Sciences and Electrical and Computer Engineering Departments at the University of Wisconsin, Madison. His interests include the design and evaluation of computer architectures, with an emphasis on memory systems for shared-memory multiprocessors. He codirects the Wisconsin Wind Tunnel Project.*

Wood received his BS degree in electrical engineering and computer science and his PhD in computer science from the University of California, Berkeley, in 1981 and 1990, respectively. He is a 1991 recipient of the National Science Foundation's Presidential Young Investigator award and a member of ACM, IEEE, and the IEEE Computer Society.

Mark D. Hill *is an associate professor in both the Computer Sciences Department and the Electrical and Computer Engineering Department at the University of Wisconsin, Madison. His work targets the memory systems of shared-memory multiprocessors and high-performance uniprocessors. He codirects the Wisconsin Wind Tunnel Project.*

Hill earned a BSE degree in computer engineering from the University of Michigan, Ann Arbor, in 1981, and MS and PhD degrees in computer science from the University of California, Berkeley, in 1983 and 1987, respectively. He is a 1989 recipient of the National Science Foundation's Presidential Young Investigator award, a director of ACM SIGArch, and a member of IEEE, IEEE Computer Society, and ACM.

Readers can contact the authors at the Computer Sciences Department, University of Wisconsin, 1210 W. Dayton Street, Madison, WI 53705-1685; e-mail {david, markhill}@cs.wisc.edu. The Web page for the Wisconsin Wind Tunnel Project is http:/www.cs.wisc.edu/~wwt.