Impact of Interconnection Network Resources on Software Productivity, Reliability, and Maintainability

Wes C. Berseth and John A. Neff, *University of Colorado*
Anis Husain, *Defense Advanced Research Projects Agency*
Bill Wren, *Honeywell Technology Center*

Limitations in processing and memory resources are known to adversely affect software productivity. Our findings indicate that limitations in interconnection network resources affect software as much as or more than processing and memory constraints. This work forms a basis to extend existing software estimation models to describe real-time multiprocessor systems.

Software is of paramount importance to the U.S. military, providing advanced surveillance, intelligence, and weapons capabilities. However, software development projects are often over schedule and over budget, and the resulting software is delivered with an unacceptable number of defects. As a result, considerable concern has been expressed regarding the capability of current software engineering practices to enable the information dominance desired by the U.S. military [1]. With current trends in real-time embedded multiprocessor applications, these problems will become increasingly more difficult to alleviate. The demand to provide greater functionality while reducing cost and cycle time is increasing, and consequently, system design and development are becoming more complex [2, 3]. A greater portion of this functionality is being implemented in software, which is becoming more complex and difficult to develop and which comprises a greater proportion of the cost of an application [3, 4]. For large digital signal processing application software, costs are typically greater than hardware costs, often comprising 70 percent to 80 percent of the total cost, which can be several million dollars.

One strategy to alleviate these growing problems is system modeling in which the impact of the major factors that impact software cost, reliability, and maintenance are explicitly included. With an appropriately modeled system, simulation and trade-off analyses can be performed to optimize the system for cost while minimizing reliability and maintenance costs.

|	Hardware Costs	Software Costs	Total Cost	Development Time
Minimum Hardware Cost	$281,000 (11)	$2,360,000 (89)	$2,640,000	32 Months
Minimum Total Cost	$432,000 (32)	$911,200 (68)	$1,343,200	28 Months

Table 1. *Comparison of costs and development time between minimum hardware cost and minimum total cost scenarios. Percentages are in parentheses. Data taken from [6].*

Here we discuss the impact of hardware on software development and report our findings on the impact of a major multiprocessor hardware component—the interconnection network—on software cost, reliability, and maintenance. We believe this work provides a basis to extend current parametric cost-estimating models to describe real-time multiprocessor systems.

Processing and Memory Resources Impact Software Productivity and Quality

It is well known that limitations in processing and memory resources increase the effort required to develop software. Under these constraints, developers must deal directly with the operating system and hardware; therefore, detailed knowledge of the machine architecture is required. Coding in low-level languages is often required and high-level development tools cannot be used [5]. This effect is expressed in parametric cost-estimating models applied to software development, e.g., Constructive Cost Model (COCOMO), Revised Enhanced Version of Intermediate COCOMO (REVIC), PRICE S, and SEER-Software Estimating Model. For example, with the PRICE S model, when processor utilization—defined as the fraction of available hardware cycle time utilized—is below 0.5, software effort is not affected. As utilization is increased above 0.5, the required software effort increases nonlinearly and is 2.33 and 4.0 times higher relative to nonconstrained utilization when processor utilization is 0.9 and 0.95, respectively (Figure 1). The identical relation-

Figure 1. *Relation between relative software effort (RSE) and processor utilization as determined from the PRICE S software estimation model [5].*
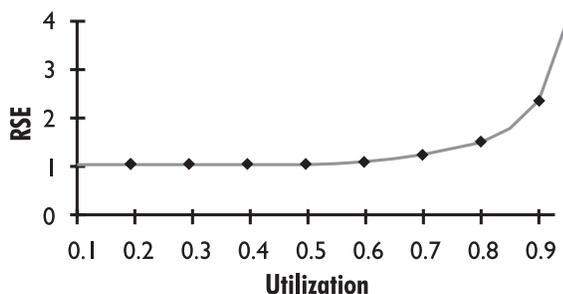
ship is given for memory utilization and software effort [5]. The defect rate, which affects the reliability of the system, increases in a similar fashion, increasing from 1.8 errors per thousand lines of code to 2.9 at processor utilization of 0.5 and 0.9, respectively (Figure 2).

Adding Processing and Memory Resources Can Decrease Software Costs

As part of the RASSP (rapid prototyping of application-specific signal processors) program, Jim Anderson of the Massachusetts Institute of Technology Lincoln Laboratory used parametric cost-estimation techniques to examine the effects of memory and processor constraints on the development costs of a synthetic aperture radar (SAR) processor [6]. For his model system, he chose an unmanned air vehicle (UAV) SAR benchmark developed at Lincoln Laboratory for the RASSP program. According to Anderson, hardware costs can be minimized by supplying enough processing (1 billion floating point operations per second) and memory resources to meet and not exceed application requirements. With commercial hardware, this can be realized with six Mercury MCV6 4 x 4m cards, each with four 40 megahertz Intel I860 processors and 16 megabytes of Dynamic Random Access Memory. In addition, a commercial back-plane-mounted crossbar switch, a Motorola MVME167 system controller card, and a custom radar interface card are required. The resulting processor will have a memory utilization of 86 percent and processor utilization of 88 percent for a total hardware cost of $281,000 (Table 1). According to Anderson, these requirements are not unusual for UAV applications where size, weight, and power must be minimized. However, the software costs and development time corresponding to the above memory and processor constraints are $2,360,000 and 32 months, respectively, as determined by the REVIC software cost estimating model. This cost was determined by estimating the code to be 8,750 uncommitted source lines of code, requiring 155 programmer-months, 152 programmer-hours per programmer-month, and $100 per programmer-hour. With this minimum hardware cost scenario, software development is 89 percent of the total development expense of $2,640,000.

Minimizing the total system cost can be achieved by adding enough hardware resources so that memory and processor

---

utilization does not have an adverse affect on software cost. This occurs when both memory and processor utilization are below 50 percent (Figure 1). This is achieved by increasing the number of Mercury MCV6 cards from six to 11 with no change to the rest of the hardware. The result is an increase in hardware costs by a factor of 1.8, to $432,000, and a decrease in software costs by a factor of 2.59, to $911,200 (Table 1). The development time is also decreased from 32 to 28 months. Anderson's example shows how development cost can be dominated by software and that a greater investment in hardware can substantially reduce overall system development costs. The net result is a superior product at half the cost of the minimum hardware product.

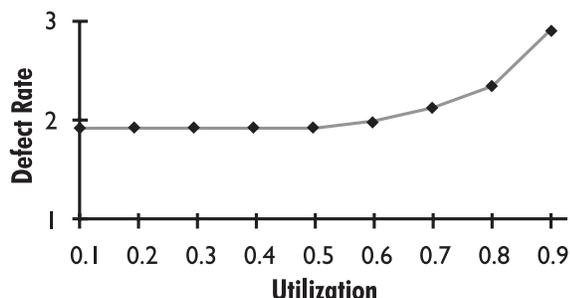## Interconnection Network Resources Impact Software Productivity and Quality

Interconnection bandwidth[1] has been identified as directly impacting the ease of programming large supercomputers for high performance [7, 8]. Consequently, software development costs can be substantial due to the considerable effort required to obtain specific optimizations that are highly tuned to the particular distribution of data and machine [9]. This sentiment was supported by numerous personal communications with experts in the supercomputer and real-time embedded digital signal processing domains. Howard Shrobe has identified limited bandwidth as one of the "seven deadly sins" of software engineering [10]. Although it has been recognized that programming multiprocessors is more difficult when interconnection bandwidth is limited, this relation has not been quantified.

Bisection Bandwidth and Software Productivity

To quantify the relationship between bandwidth constraints and software productivity, estimates were obtained from experts experienced in the development of multiprocessor applications—seven experienced in real-time embedded signal processing and one in high-performance supercomputing. In addition, a questionnaire, developed at the University of Colorado, was used to collect additional information, e.g., impact on software quality and maintenance, and provide a check on model estimates. Eleven multiprocessor experts responded to the questionnaire, including seven of the above eight. We believe this was a suitable approach to determine the general relationship between bandwidth constraints and software productivity, thus reflecting an industry average. As a measure of bandwidth constraint, we use bisection bandwidth utilization (BBU), defined as the average fraction of available bisection bandwidth[2] that is used during data transfers. We believe this to be a useful measure of the difficulty encountered by a software developer, since bisection bandwidth is the critical bottleneck when performing global data transfers such as corner-turn[3] operations inherent in digital signal processing applications. Also, it is consistent with measures of hardware constraints used in existing parametric cost-estimation models, i.e., processor and memory utilization.

As a measure of software productivity, we define relative software effort (RSE) as the ratio of the effort required to de-

velop software relative to the effort required if bandwidth were not constrained. A relative measure was chosen to normalize data and enable pooling from a broad range of application parameters, such as size of application and programming language.

The relationship between BBU and RSE is given in Figure 3. RSE is not affected until BBU reaches 0.3, beyond which RSE increases nonlinearly, increasing to 3.8 at a BBU of 0.9. As BBU approaches 1.0, RSE becomes extremely high and the relationship is undefined, although in practice other factors likely become important. Some of the experts interviewed stated that dramatic increases in software costs initiate other decisions. For example, the program can be temporarily called to a halt while the organization waits until faster hardware becomes available or custom hardware is developed.

These results indicate that bisection bandwidth constraints have a more dramatic impact on software productivity than either memory or processor constraints. The adverse impact of bandwidth constraints sets in at a BBU of 0.3, compared to 0.5 for processor or memory utilization. Also, the RSE is 2.2 for a BBU of 0.8 compared to 1.5 when processor utilization is 0.8 (Figures 1, 3).[4]

Bisection Bandwidth and Software Quality, Complexity, and Reliability

The reliability of a system will depend on the quality of the software, indicated by the defect rate, and it is well known that defect rates are higher when software becomes more complex [11]. As with processing and memory constraints, when bandwidth is limited, the software becomes more complex, which results in higher defect rates.[5] Often, one is forced to decompose the problem by task as opposed to data domain. Task parallelism is more asynchronous than data parallelism, introducing a load balancing problem and making synchronization more difficult. Also, it is often necessary to write low-level communication protocols and reduce communication, both of which are extremely difficult. The code must be made more efficient, which often requires programming in lower-level languages such as Assembly, resulting in complex code that is hard to understand. This can have a significant impact on the test and integration phase of a system as errors are more difficult to detect when code is complex. In fact, many errors go undetected until the operation and maintenance phase.

Figure 3. *Relation between RSE and BBU.*
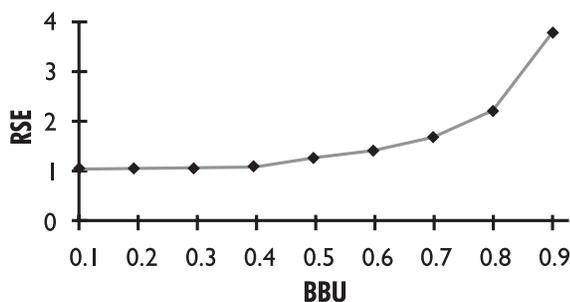
Bisection Bandwidth and Software Maintenance

Software maintenance, which includes fixing defects and upgrading functionality, is generally the most costly phase of the lifecycle [10]. For example, the cost to develop 236,000 lines of code for an F-16 fighter system was $85 million, whereas the maintenance cost was $250 million [3]. It is extremely difficult to fix defects and upgrade functionality on complex systems. When part of the system is changed, the effect on the rest of the system must be determined, which requires considerable testing to ensure that the system is fully operable. For high-reliability systems, 75 percent of the time in an upgrade cycle is spent in testing and analyses [10]. With a bandwidth-constrained system, the maintenance phase requires more effort as the added complexity makes it more difficult to test the system.[6]

Upgrading functionality can put increased demands on communications resources, which results in an increase in the utilization of available bisection bandwidth, thus making software development and testing increasingly difficult[7] (Figure 3). It may also be necessary to reallocate bandwidth on the original application to accommodate the added functionality, which requires additional software design and development. If bandwidth is constrained in the original application and upgrades continually require additional bisection bandwidth, software development will become increasingly more difficult until upgrades are no longer possible.

Advantages of Additional Communication Resources

Given the substantial impact of bandwidth constraints on software development and maintenance costs, it may be strategically advantageous to invest in high-bandwidth interconnection networks or to develop new interconnection technologies. One promising technology is free-space optics in which data is transmitted through free space by unguided optical beams. The Defense Advanced Research Projects Agency (DARPA) recently initiated the Free-Space Optical Interconnect Accelerator Program with the intent to transfer this technology to military systems.

Besides providing direct savings in software costs, additional advantages can be realized by increasing bandwidth. This includes greater capability in Department of Defense (DoD) radar and imaging applications by enabling communication-intensive algorithms and by effectively implementing shared-memory systems. Greater capability may have direct impact on mission effectiveness, i.e., reduced loss of aircraft and life, and, therefore, on national security.

It is widely recognized that shared-memory systems are easier to program than message-passing systems that require extensive "tuning" to achieve optimal performance through locality and are more difficult to modify [8]. A shared-memory machine, with uniform memory access, does not require the programmer to be concerned about data locality to achieve optimal performance, which makes it easier to develop, maintain, and reuse code. However, to effectively implement locality independence and make dynamic load balancing effective, a high bandwidth interconnection network is required [8].

Multiprocessor Model for Software and System Cost Optimization

A number of parametric models have been developed to estimate software development costs for uniprocessors. These models can be used to estimate software effort and related factors such as schedule length once one determines the size of the software and attributes inherent in the project and development process, including

- Product complexity and reliability requirements.
- Memory and processor constraints.
- The level of application and programming experience of the employees.
- The use of modern programming practices and development tools.

To apply these models to multiprocessor systems, the impact of the interconnection network is included implicitly through attributes such as product complexity. However, this does not enable one to exploit the full benefits of modeling, i.e., trade-off analyses, to optimize software and system costs. For meaningful optimization, the relations between individual multiprocessor hardware components (processors, memory, and interconnection network) and software productivity and quality must be explicitly included in the model. Additional factors to consider are the number of processors and the interdependencies between major hardware components, e.g., to accommodate for bandwidth constraints requires increasing processor and memory requirements.

Developers of real-time multiprocessor systems have expressed a desire for such a model. A model of this nature should be extremely useful for the rapid design and prototyping of cost-effective real-time multiprocessing systems. It would enable trade-off analyses to be made in the early stages of the development cycle, e.g., conceptualization, and would support decisions on high-level issues such as technology choices. It would also be desirable to include the maintenance phase in a real-time embedded software estimation model. This would allow one to optimize for upgrades and enable trade analyses based on the entire lifecycle of the system. We believe a modeling tool of this nature would produce substantial savings in costs over the lifecycle of an application.

Summary

Existing models, developed for uniprocessors, consider processing and memory constraints but do not consider parameters unique to multiprocessors such as bisection bandwidth constraints or number of processors. We have found that bisection bandwidth constraints impact software development as much as or more than processing and memory constraints. For this reason, the importance of bisection bandwidth should not be overlooked when estimating software costs for real-time embedded multiprocessors.

This work provides a basis to extend existing parametric software cost-estimating models to describe real-time embedded multiprocessor systems. Such a model would make it possible to perform trade analyses to optimize system cost and performance, which will lead to substantial savings in development and maintenance costs, increased performance, and easier upgrades. ◆

About the Authors

Wes C. Berseth is a professional research associate of the Optoelectronic Computing Systems Center at the University of Colorado. His research includes the application of optoelectronics to imaging and communications systems from an architectural, performance, and cost perspective including the impact of hardware resources on software productivity and quality. He is currently principal investigator of modeling and simulation on a DARPA-sponsored Free-Space Optical Interconnect Accelerator Program. He has a bachelor's degree and a master's degree from York University, Toronto, Canada, and holds a doctorate in physics from York University. He is a member of the American Physical Society, Association for Computing Machinery, and the Institute of Electrical and Electronics Engineers (IEEE).

Optoelectronic Computing Systems Center
Campus Box 525
University of Colorado
Boulder, CO 80309-525
Voice: 303-492-0478
Fax: 303-492-3674
E-mail: wberseth@colorado.edu

John A. Neff is director of the Optoelectronic Computing Systems Center at the University of Colorado. Before joining the University of Colorado in 1991, he was the research manager for optical computing at DuPont. Prior to joining DuPont in 1988, he spent 15 years as a program manager for optical computing with the DoD, first with the Air Force Office of Scientific Research, then with DARPA. He has a bachelor's degree in physics and in mathematics from Ohio Wesleyan University and a master's degree and doctorate in electrical engineering from Ohio State University. He is a fellow and past governor of the Society of Photo-Optical Instrumentation Engineers and is a fellow of the Optical Society of America. He is listed in Strathmore's Who's Who of Business Leaders.

Director
Optoelectronic Computing Systems Center
University of Colorado
Campus Box 525
Boulder, CO 80309-0525
Voice: 303-492-7135
Fax: 303-492-3674
E-mail: jneff@colorado.edu

Anis Husain is assistant director of the Electronics Technology Office at DARPA. From March 1994 until May 1997 he served as program manager of optoelectronics in the DARPA Microelectronics Technology Office (MTO) re-

sponsible for research and development in optical interconnects, free space optical interconnects, visible emitters and detectors, and high-density optical memory. Prior to this, he was affiliated with the Center of High-Performance Computing, Worcester Polytechnic Institute, Boston, Mass. as an on-site consultant to DARPA/MTO. From 1980 to 1995, he worked for Honeywell Technology Center as supervisor of media and network architecture and section manager of photonics. He has a bachelor's degree in electrical engineering from Imperial College, University of London, United Kingdom and holds a doctorate from University College London, United Kingdom. He is a member of the IEEE, IEE, Optical Society of America, Computer Society, Communications Society, and Lasers and Electrooptics Society.

Optical Micro-Machines, Inc.
6160 Lusk Blvd., Suite C205
San Diego, CA 92121
Voice: 703-758-2622
Fax: 619-642-0490
E-mail: ahasain_omm@ibm.net

Bill Wren is a principal research scientist at the Honeywell Technology Center in Minneapolis, Minn. He has broad experience in scalable real-time processor architecture design, system modeling and trades, digital design, signal processing design and programming, software architecture design, and program management. He has a bachelor's degree in electrical engineering from the University of Nebraska. His graduate studies included work at Arizona State University, and he received a master's degree in electrical engineering from the University of St. Thomas, where he studied software systems.

Honeywell Technology Center
18820 DeAnn Circle
Minnetonka, MN 55345
Voice: 612-951-7885
E-mail: wren_bill@htc.honeywell.com

References

1. Donahue, Lt. Gen. William J., "Information Dominance Through Software Technology," *CROSSTALK*, STSC, Hill Air Force Base, Utah, July 1997, pp. 3-4.
2. Stutzke, Richard D., "Software Estimation Technology: A Survey," *CROSSTALK*, STSC, Hill Air Force Base, Utah, May 1996, pp. 17-22.
3. Bunza, Geoff J., "A Journey into Parallel Worlds: Exploring Hardware/Software Systems Integration," *Embedded Systems Conference*, San Jose, Calif., September 1996, Miller Freeman.
4. Bartow, James, "Evolutionary Design of Complex Systems," *Investments in Avionics and Missiles Software Technology Workshop Report, ARPA/SISTO*, Software Productivity Consortium, Inc. Report SPC-95068-CMC, 1995.
5. Minkiewicz, Arlene and Anthony DeMarco, "The PRICE Software Model," PRICE Systems, June 1996.
6. Anderson, James. C., "Projecting RASSP Benefits," *Proceedings of the 2nd Annual RASSP Conference*, ARPA, Arlington, Va., 1995, pp. 65-72.
7. *Accelerated Strategic Computing Initiative (ASCI) PathForward Project Description*, Dec. 27, 1996, http://www.llnl.gov/asci-pathforward.
8. Probst, D.K., "Architectural Visions Versus Business Models: How Soon Will There Be Enabling Technologies for Petaflops Computing?" Report prepared for Gil Weiland at DoE/DP-07 [HQ] December 1995.
9. Blelloch, Guy E., B.M. Maggs, and G.L. Mile, "The Hidden Cost of Low Bandwidth Communication," *Developing a Computer Science Agenda for High-Performance Computing*, ACM Press, 1994, pp. 22-25.
10. Shrobe, Howard, "Evolutionary Design of Complex Software," *Investments in Avionics and Missiles Software Technology Workshop Report, ARPA/SISTO*, Software Productivity Consortium, Inc. Report SPC-95068-CMC, 1995.
11. Jones, Capers, *Applied Software Measurement: Assuring Productivity and Quality*, 2d ed., McGraw-Hill, New York, 1997.

Notes

1. Bandwidth is defined as the rate at which an interconnection link can transfer information. For digital systems it is measured in bits per second.
2. Bisection bandwidth provides a measure of the communication resources of the network. For a symmetric network, bisection bandwidth is determined by dividing the interconnection network into two equal parts, each with half the processing nodes, and summing the bandwidth of all lines crossing the dividing line.
3. The corner-turn, also called a transpose, is an "all-to-all" communication operation in which the processors on the network send data to each other in preparation for the next computation operation. This operation is important in certain signal-processing applications, e.g., two-dimensional fast Fourier transform and SAR, and can severely overload the network and stall computation.
4. This was supported by the questionnaire as seven of 10 claimed that bisection bandwidth constraints can affect software development as much as or more than memory constraints, and eight of 10 claimed that software development is affected as much as or more than when processing is constrained.
5. Ten of 11 experts surveyed claimed that bandwidth constraints increase software complexity and six of six claimed that bandwidth constraints affect defect rate by making code more complex.
6. Eight of 10 experts interviewed stated that bandwidth constraints make the maintenance phase more difficult because the added complexity makes it more difficult to test the system.
7. Seven of 11 experts interviewed stated that adding new functionality to a previously developed application will use additional bandwidth. The remaining four said it can, depending on whether the tasks are scheduled concurrently.