from *SIAM News*, Volume 33, Number 1

A Review of Regular Domain Partitioning

By Manuel Prieto, Ignacio Llorente, and Francisco Tirado

Ratios of computational power to memory bandwidth have increased to the point that the maximum performance of current microprocessors is limited by the time for memory accesses. Continued worsening of the situation seems likely over the next few years. It has been suggested that this trend could result in a "memory wall," in which application performance would be completely dominated by memory access time [9].

The most common technique for bridging the gap between computational rate and memory performance is a hierarchical memory structure, with large, fast caches close to the processor. Such memory structures have a significant impact on the design and development of code. To keep the processor busy, applications must exploit spatial and temporal locality to make efficient use of the caches. The effectiveness of data locality has been well demonstrated for algorithms of many types, e.g., linear algebra algorithms for dense matrices (the LAPACK project) and iterative methods [8].

APPLICATIONS ON ADVANCED ARCHITECTURE COMPUTERS

Greg Astfalk, Editor

Interconnect networks—the other essential factor in parallel performance—have taken advantage of the increasing integration density offered by VLSI technology. Effective bandwidths of networks are now hundreds of times faster than those of ten years ago [2], in addition to having lower latencies.

We have examined the effects of these trends on the decomposition of regular applications. Traditional three-dimensional decompositions can lead to lower inherent communication-to-computation ratios and thus to more efficient exploitation of interconnect networks. However, lower-dimensional decompositions, in which boundaries with poor spatial locality are not required, have been found to be more efficient on current parallel architectures.

The reason is that the bandwidth reductions resulting from non-unit-stride memory access are more important than those from network contention. Thus, local optimization, as seen with the use of a well-known communication–computation overlapping technique, is increasingly important. The execution time, instead of the reduction we would expect, is increased because of poor cache exploitation.

Overheads in Message-passing

Message-passing between two tasks located on different processors can be divided into three phases: The send and receive overhead phases, when the system interfaces with the interconnect, and a network delay phase, during which the data are physically transmitted between the processors. Specific details of what the system does during these phases vary. Typically, however, during the send phase the message is copied into a system-controlled message-buffering area, and control information is appended to the message. In the same way, in the receive process the message is copied from a system-controlled buffering area into user-controlled memory.

The network delay depends on the first-bit delay, the network bandwidth, and the contention. The first-bit delay is the time required for the first bit of the message to reach the destination processor or memory. This, in turn, depends on the number of hops between adjacent network nodes or the number of switches traversed by the message. The network bandwidth is the rate at which the remainder of the message data arrives at the destination, after the first-bit delay. Contention is the overhead induced by competition for resources with other activities in the system.

As the bandwidths of interconnection networks increase, the send and receive overheads become more dominant. The factors determining these overheads are different in each system, but software overhead, uncached operations, cache misses, and synchronization instructions are usually among those involved.

The degree to which communication contributes to execution time in a real application depends not only on the amount of communication, but also on the way the data are structured in the local memories, the impact of the send and receive overheads, and the way the processor mapping exploits the interconnect network.

Cray T3E Message-passing Performance

The T3E-900 system [1] used in this study has 40 DEC Alpha 21164 (DEC Alpha EV5) processors running at 450 MHz. The EV5 contains no board-level cache, but the Alpha 21164 has two levels of cache on-chip: separate 8-Kbyte first-level instruction and data caches, and a unified, second-level three-way associative, 96-Kbyte write-back cache.

The processors are connected via a 3D-torus network. Its links provide a raw bandwidth of 600 Mbyte/second in each direction with an interprocessor payload bandwidth of 480 Mbyte/second. When MPI is used, the effective one-way bandwidth is smaller because of the overhead associated with buffering and deadlock detection [6, 7].

The maximum bandwidth obtained in this system depends on the mode used for sending and receiving data. Messages can be either buffered by the system or sent directly in a synchronous way by the T3E E-register mechanism. For a ping-pong test, the maximum achievable bandwidth obtained for synchronous communication is approximately 300 Mbyte/second. Because of the extra memory copies required, the best performance attained with buffered communication is only about 160 Mbyte/second.

In a contention-free environment, adding hops does not have a significant effect on the communication bandwidth. As the network load increases, however, the effective bandwidth will depend on the network distance between the two processors involved

[6]. Using synchronous communications, we have found that only zero- and one-hop distances are advisable for neighboring logical processors. With buffered communication, the deterioration in the effective bandwidth is not as important, although the maximum bandwidth attainable in buffered communications is limited to 160 Mbyte/second.

We have measured the impact of exchanging messages that are noncontiguous in memory [6]. In particular, we have considered messages that consist of double-precision (i.e., 64-bit) data elements separated by a constant stride (stride-1 represents contiguous messages). The effective bandwidth also depends on the communication mode. With buffered communications for stride-8, only one word of the eight-word L2 cache line (for the system we used) is useful in the required memory copies. Thus, the effective bandwidth remains almost constant for this stride. The reduction in bandwidth with non-unit stride is important. For a 256-Kbyte message, for example, stride-1 bandwidth is approximately five times larger than stride-32 bandwidth.

Better performance is obtained with synchronous communication because the E-register mechanism allows the second-level cache to be bypassed [2]. As a result of memory bank conflicts, the effective bandwidths for synchronous communication are approximately 122 Mbyte/second for odd-numbered strides, 72 Mbyte/second for strides that are multiples of two, 71 Mbyte/second for multiples of four, and 68 Mbyte/second for multiples of eight.

SGI Origin2000 Message-passing Performance

We repeated the Cray T3E tests on an SGI Origin2000 [4], a cache-coherent distributed–shared-memory system. Each node of the Origin contains two processors connected by a system bus (SysAD bus), a portion of the shared main memory on the machine (512 Mbyte in our system), a directory for cache coherence, the Hub (which is the combined communication/coherence controller and network interface), and an I/O interface called Xbow. In the system used in this study, the MIPS R10000 processor runs at 250 MHz. Each processor has a 32-Kbyte two-way set-associative primary data cache and a 4-Mbyte two-way set-associative secondary data cache.

The peak bandwidths of the bus that connects the two processors and the Hub's connection to memory are 780 Mbyte/second. However, the peak local memory bandwidth is only about 670 Mbyte/second. The Hub's connections to the off-board network router chip and the I/O interface are 1.56 Gbyte/second each. The Origin network is based on a flexible switch, called SPIDER, that supports six pairs of unidirectional links, with each pair providing more than 1.56 Gbyte/second of total bandwidth in the two directions.

At the user level, the actual effective bandwidth between processors—about 120 Mbyte/second for 2-Mbyte messages—is much lower than the peak because of the cache-coherency protocol and other overheads [6]. The measured bandwidth decreases when messages are larger than the L2 cache. The internal message buffers cause this reduction.

Long messages that do not fit in the secondary cache require extra main memory accesses. In this system, as in the T3E, the time required to send a message from one processor to another with an unloaded network varies little with the network distance between the processors. However, the reduction in the actual effective bandwidth caused by extra network loads is larger than on the T3E [6].

The behavior for noncontiguous mes-sages is similar to that for the T3E buffered mode, although experimental results suggest a poor MPI implementation. The second-level cache line size in the Origin is 128 bytes (16 double-precision variables), and the actual bandwidth thus remains constant, beginning with stride-16, at between 9.0 and 10.2 Mbyte/second.

Impact of Spatial Locality on Regular Decompositions

We can now state that in the two systems under study, the bandwidth reduction that results from non-unit-stride memory access is more important than that resulting from contention in the network. How does this conclusion affect real applications? We have considered algorithms for which the main computational portion consists of matrix computations in which groups of neighboring data elements are combined to calculate a new value. Such algorithms are common in image processing, geometric modeling, and the solution of partial differential equations by finite difference or finite volume techniques.

The simplest approach to parallelizing regular applications of this type is to distribute the data among the processors. Values unique to individual subdomains are computed independently on the different processors of the network, after which a communication step between logically neighboring processors updates the boundary values in these subdomains.

As a sample problem, we used a multi-grid-like iterative method [7] in the solution of a three-dimensional finite difference application. The code is written in C, so a three-dimensional domain is stored in a row-ordered (x,y,z) array. For distribution across a one-dimensional mesh of virtual processors, there are three possible partitionings: x-direction, y-direction, and z-direction. The x- and y-direction partitionings were found to be more efficient, because the message data exhibit better spatial locality. As shown in Figure 1, yz and xz boundaries are stride-1 data, except for the gap between different z columns (in our application, four doubles for yz boundaries, and this quantity plus two times the number of elements in a yz-plane for xz boundaries). With z partitioning, a message references data with a stride of twice the number of elements in the z dimension (all the elements are double-precision complex data).

For the 256^3-element problem on the T3E with buffered communications, the performance of x partitioning was approximately twice that of z partitioning. In synchronous mode, the performance difference between contiguous and noncontiguous boundaries was approximately 1.6. Buffered communications are more efficient than synchronous-mode communication, however. As is common in parallel programs, all the processes in this application communicate at the same time, which can overburden the communication network and cause contention.

Obviously, buffering can relieve the problem. In addition, some discrepancies can be attributed to the use of the E-registers on the Cray T3E. The application data have temporal locality; thus, if the data to be transferred are in the data cache, accessing them

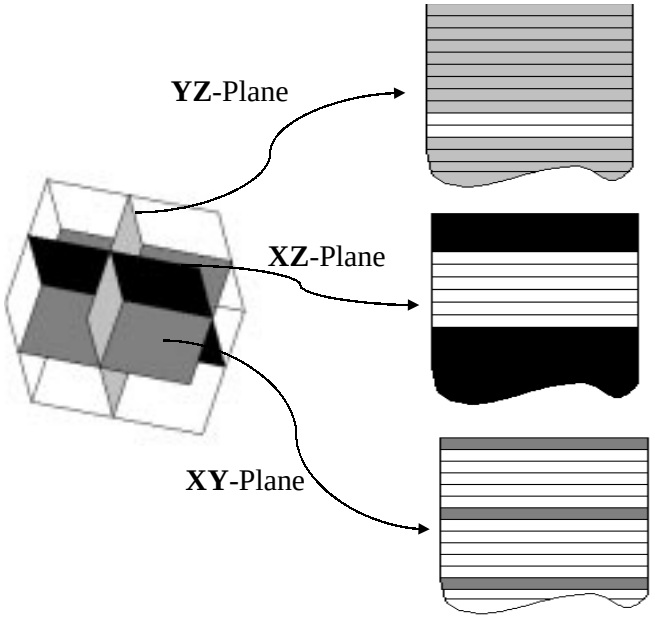

Figure 1. *Data pattern in memory for the three different boundaries in a three-dimensional domain.*

via E-registers invalidates the corresponding cache line, and the temporal locality advantages are lost.

Similar differences, although less important than in the T3E case, are observed for the Origin2000. For the 256^3-element problem on the Origin, *x* partitioning is only 1.3 times better. Because spatial locality is less important, the results are influenced by the large second-level cache of this system, which allows the best exploitation of the temporal locality.

Optimal Partitioning

When the effect of spatial locality is taken into account, the choice of the optimal partitioning becomes a trade-off between reduction of the send and receive overheads—in particular, the spatial locality of the data to be exchanged—and efficient exploitation of the interconnection network. The correct processor mapping reduces contention, and three-dimensional decompositions are thus the best choice for the T3E topology. In addition, for a three-dimensional regular application, the communication requirements for a process increase with the size of the boundaries, while the amount of computation increases with the size of the entire partition. The communication-to-computation ratio is thus a surface area-to-volume ratio. The three-dimensional decomposition leads to a lower inherent communication-to-computation ratio [3, 5]. However, higher-dimensional decompositions require noncontiguous boundaries, and, as discussed earlier, the effective bandwidth is reduced because of non-unit-stride access. Additionally, the messages generated are more numerous, and therefore smaller. Given the fixed component of the send and receive overheads associated with initiating or processing messages, it is better to have fewer, larger messages.

The left-hand plot in Figure 2 compares the different decompositions for our sample application on the Cray T3E. For up to 16 processors, the best performance is obtained with linear, two-dimensional decompositions, where *xy*-plane boundaries are not needed. In the larger problem the best two-dimensional and linear decompositions achieve improvements of 7% over the best three-dimensional decomposition. Obviously, as the number of processors is increased, efficient exploitation of the underlying network becomes more important. In the 32-processor simulation, a two-dimensional decomposition achieves the best trade-off between network exploitation and local memory access. In the larger problem, the best two-dimensional decomposition is 5% and 15% better than the three-dimensional and linear decompositions, respectively.

For the SGI Origin2000 (right-hand plot in Figure 2), we obtained similar results for 32 processors; the two-dimensional decomposition is 22% and 8% better than the one- and three-dimensional decompositions, respectively.

Computation–Communication Overlapping

Users often "hide" communication cost and contention by overlapping the communication with other useful work. Although the results in the previous sections were obtained without overlapping, the algorithms used can be structured so that every process request for remote data is interleaved explicitly with local computation. For this purpose, it is necessary to deal with the boundaries *before* the solution is computed on the interior of the domain. In this way, an immediate send operation can be initiated before it would naturally appear in the program and the message can be received before it is actually needed. Thus, the receive operation might not need to stall while waiting for the message to arrive; it can copy the data directly into the application address space. Therefore, we do not use the following pattern:

1. Exchange artificial boundary:

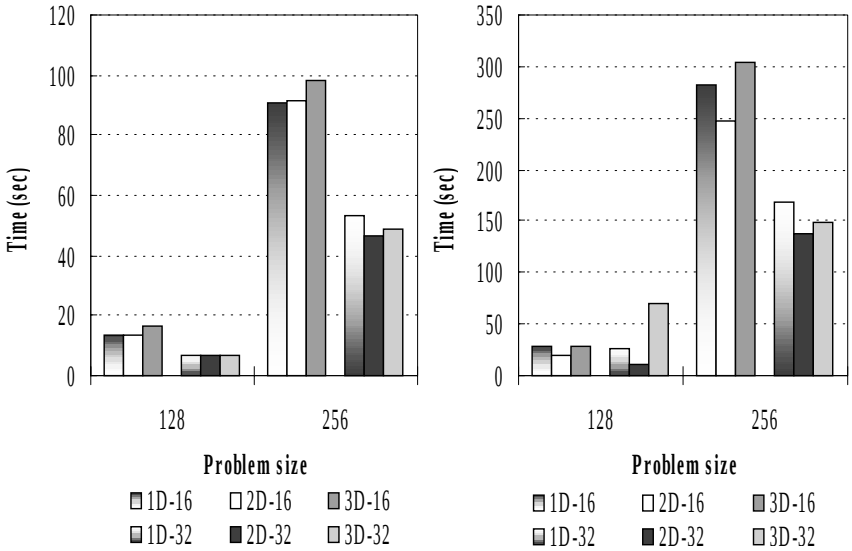

Figure 2. *Different decompositions for the sample application, with 16 and 32 processors on the T3E (left) and the SGI Origin2000 (right). The problem size is the number of cells (two double-precision data items for each) in each dimension.*

(a) Send boundaries to neighbors
(b) Receive artificial boundaries from neighbors
2. Update local domain using artificial boundaries

Instead, we use:

1. Update boundaries
2. Send boundaries to neighbors
3. Update local domain using artificial boundaries
4. Receive artificial boundaries from neighbors

We successfully used communication overlapping on an old IBM SP2 system, where the network delay cost was higher than for either the T3E or the Origin2000. Overlapping was found to be 1.3 times better than non-overlapping for the 128^3-element simulation [7]. The performance difference increases with the problem size (e.g., to 2.5 times for the 256^3-element case). The best decomposition with overlapping was also a two-dimensional decomposition. The differences between the decompositions in this case are a consequence not only of message-passing but also of the different schemes for updating the boundaries. In workstation clusters, because the network is usually a shared resource and has even larger latency, the benefits will be even greater.

On the T3E and the Origin2000, the network delay cost is not as important if a correct processor mapping is chosen. Consequently, the cost due to the different locality properties of the overlapping approach may be greater than the benefits of the communication–computation overlapping. For 32 processors on the CRAY T3E, for the larger problem size with a two-dimensional decomposition, the non-overlapping approach is 7.3% faster than overlapping. For 16 processors it is only 5% faster [7]. Similar differences are seen on the Origin. For 32 processors the difference for the larger problem is 7.5%.

Conclusion

On both systems studied, the Cray T3E and the SGI Origin2000, the bandwidth reductions resulting from non-unit-stride local memory access are more important than those from contention in the network. Data partitioning involves a trade-off between improved locality of the message data and efficient exploitation of the underlying communication system. With up to 32 processors on both systems, an appropriate two-dimensional decomposition, where boundaries with poor spatial locality are not required, is the best choice. With systems like these, for which communication has become a dominant factor in the effective communication bandwidth, no significant effort should be made to overlap communication with computation.

Execution time, the performance metric we considered in this article, is but one of many aspects to be evaluated in a parallel program. If we focused instead on implementation cost, a one-dimensional partitioning might be the best choice, given the ease with which lower-dimensional partitionings can be coded. Such an approach would allow the implementation of fast sequential algorithms in the nonpartitioned directions.

Acknowledgments

This work has been supported by Spanish research grants TIC 96–1071 and TIC IN96–0510 and the Human Mobility Network CHRX–CT94–0459. The authors thank Ciemat and CSC (Centro de Supercom-putacion Complutense) for providing access to the parallel computers used in this research.

References

[1] E. Anderson, J. Brooks, C. Grass, and S. Scott, *Performance of the CRAY T3E multiprocessor*, Proceedings of ACM/IEEE Supercomputing 97, IEEE Computer Society, San Jose, November (1997) (available from http://www.supecomp.org/sc97/proceedings).

[2] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks. An Engineering Approach*, IEEE Computer Society Press, Los Alamitos, California, 1997.

[3] I.T. Foster, *Designing and Building Parallel Programs. Concepts and Tools for Parallel Software Engineering*, Addison-Wesley, New York, 1995.

[4] J. Laudon and D. Lenoski, *The SGI Origin: A ccNUMA highly scalable server*, ACM Comp. Arch. News, 25:2, Special Issue, ISCA'24, Proceedings, May (1997), 241–251.

[5] M. Llorente and F. Tirado, *Relationships between efficiency and execution time of full multigrid methods on parallel computers*, IEEE Trans. Par. and Dist. Sys., 8:6 (1997), 562–573.

[6] M. Prieto, D. Espadas, I.M. Llorente, and F. Tirado, *Message-passing evaluation and analysis on Cray T3E and SGI Origin2000 systems*, Proceedings of Europar'99, Springer-Verlag Lecture Notes in Computer Science, 1685 (1999), 173–182.

[7] M. Prieto, I.M. Llorente, and F. Tirado, *Partitioning of regular domains on modern parallel computers*, Proceedings of VEC-PAR'98, Springer-Verlag Lecture Notes in Computer Science, 1573 (1999), 411–424.

[8] U. Rüde, *Iterative algorithms on high performance architectures*, Proceedings of Europar'97, Springer-Verlag Lecture Notes in Computer Science, 1300 (1997), 57–71.

[9] W.A. Wulf and S.A. McKee, *Hitting the memory wall: Implications of the obvious*, Comput. Arch. News, Association for Computing, March (1995), 20–24.

Manuel Prieto (mpmatias@dacya.ucm.es) is an assistant professor of computer architecture at the Universidad Complutense in Madrid, Spain. Ignacio M. Llorente (llorente@dacya.ucm.es) is an associate professor of computer architecture at the Universidad Complutense in Madrid, Spain. Francisco Tirado (ptirado@dacya.ucm.es) is a professor of computer architecture at the Universidad Complutense in Madrid, Spain.