The Hierarchical Factor Algorithm for All-to-All Communication

Peter Sanders[1,⋆] and Jesper Larsson Träff[2]

[1] Max-Planck-Institut für Informatik,
Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany,
sanders@mpi-sb.mpg.de, http://www.mpi-sb.mpg.de/~sanders/
[2] C&C Research Laboratories, NEC Europe Ltd.,
Rathausallee 10, 53757 Sankt Augustin, Germany,
traff@ccrl-nece.de

Abstract. We present an algorithm for *regular, personalized all-to-all communication*, in which every processor has an individual message to deliver to every other processor. Our machine model is a cluster of processing nodes where each node, possibly consisting of several processors, can participate in only one communication operation with another node at a time. The nodes may have different numbers of processors. This general model is important for the implementation of all-to-all communication in libraries such as MPI where collective communication may take place over arbitrary subsets of processors. The algorithm is optimal up to an additive term that is small if the total number of processors is large compared to the maximal number of processors in a node.

1 Introduction

A successful approach to parallel programming is to write a sequential program executing on all processors and delegate interprocessor communication and coordination to a communication library such as MPI [9]. With this approach, many parallel computations can be expressed in terms of a small number of *collective communication operations*, where "collective" means that a subset of processors is cooperating in a nontrivial way. One such frequently used collective communication operation is *regular, personalized all-to-all message exchange*: Each of p processors has to transmit a personalized message to itself and each of $p-1$ other processors, i. e., for every pair of processor indices i and j a message m_{ij} has to be sent from processor i to processor j. In *regular* all-to-all exchange, all messages are assumed to have the same length. Examples of subroutines using all-to-all communication are matrix transposition and FFT.

This paper presents an algorithm for regular all-to-all communication on clusters of processing nodes where each node may consist of several processors. We assume that only a single processor from each node can be involved in internode communication at a time. Prime examples of such hierarchical systems are

⋆ Partially supported by the Future and Emerging Technologies programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

B. Monien and R. Feldmann (Eds.): Euro-Par 2002, LNCS 2400, pp. 799–803.

clusters of SMP nodes, where processor groups of 2–16 processors communicate via a shared memory, and where some medium to large number of nodes are interconnected via a commodity interconnection network. For example, the Earth Simulatorand the NEC SX-6 supercomputerhave up to 8 processors per node; the IBM SP POWER3 allows up to 16 processors per node.The difficult case is when nodes have differing numbers of processors participating in the all-to-all exchange. This situation must be handled efficiently in a high-quality communications library because it arises naturally if a job is assigned only part of the machine, or if the exchange is only among a subset of the processors in a job.

We use a simple machine model that allows an efficient implementation portable over a large spectrum of platforms. The nodes are assumed to be fully connected. Communication is *single ported* in the sense that at most one processor per node can communicate with a processor on another node at a time. The single-ported assumption is valid for current interconnection technologies like Myrinet, Giganet, the Scalable Coherent Interface (SCI), or for the crossbar switch used on the NEC machines and the Earth Simulator.

Our algorithm for all-to-all communication extends a well-known algorithm for non-hierarchical systems based on *factoring* the complete graph into matchings. The new algorithm is optimal with respect to the time a processor spends waiting or transmitting data up to an additive term that is bounded by the time needed for data exchange inside a node. This time is comparatively small if the total number of processors is large compared to the maximum number of processors in a node. Our algorithm runs in phases, in each phase getting rid of nodes with the minimum number of processors among the surviving nodes. The main issue is to balance the communication volume of nodes with many processors over the phases so that the number of communication steps is minimized.

All-to-all communication has been studied intensively, and we mention only a sample of the known results. Most work focuses on non-hierarchical systems with specific interconnection networks [7,2,10]. Trade-offs between communication volume and number of communication start-ups were studied in [1,2], which achieve algorithms that are faster for small messages. Collective communication on hierarchical systems has recently received some attention [8,5,4]. Huse [4] reports experiments with a regular all-to-all algorithm which ensures that only one processor per node is involved in inter-node communication at a time. Algorithmic details and properties are not stated.

2 The Non-hierarchical Factor Algorithm

The basis for our algorithm is a well-known algorithm for the single-ported, non-hierarchical case [2,6,10]. This algorithm exploits the existence of a 1-*factorization* of the complete graph [3]. Our formulation of the all-to-all communication problem requires the inclusion of self-loops in the graph, whereas the usual construction has no self-loops. We give the construction and the proof here. It is perhaps interesting to note that self-loops *simplify* the construction.

Lemma 1. *Let G be the complete graph with p vertices including self-loops. G is* 1-factorizable*, i. e., $G = (V, E)$ can be decomposed into p subgraphs $G^i = (V, E_i), i = 0, \ldots, p-1$ in which each vertex has degree 1 (*1-factors*).*

Proof. Let $V = \{0, \ldots, p-1\}$. The ith factor $G^i = (V, E_i)$, is constructed as follows. For $u \in V$ define $v^i(u) = (i - u) \bmod p$. Define $E_i = \{(u, v^i(u)) | u \in V\}$. Since $v^i(v^i(u)) = (i-((i-u) \bmod p)) \bmod p = u$ all vertices have degree exactly one. Furthermore, any edge $(u, v) \in E$ will find itself in some factor, namely in factor $G^{(u+v) \bmod p}$. In particular, the self-loop (u, u) will find itself in $G^{2u \bmod p}$. ∎

The non-hierarchical *factor algorithm* is the basis for our hierarchical algorithm explained in the next section. It requires p communication rounds for any number of p processors. In the ith round, all processors u and v that are neighbors in G^i are *paired* and exchange their messages m_{uv} and m_{vu}.

3 All-to-All Communication on Hierarchical Systems

We now generalize the factor algorithm to clustered, hierarchical systems. Let N be the number of processor nodes, and let G denote the N-node complete graph with self-loops. Let G_A denote the subgraph of G induced by a subset of nodes A, and G_A^i the ith 1-factor of G_A. We use U and V to denote processor nodes of the system, and u and v for individual processors. By size(U) we denote the number of processors in node U, and by $l(u)$ the *local index* of processor u within its node, $0 \leq l(u) < \text{size}(U)$ for $u \in U$. To specify what messages should be exchanged when two nodes U and V are paired we impose the node ordering

$$U \preceq V \text{ if } \text{size}(U) < \text{size}(V), \text{or } \text{size}(U) = \text{size}(V) \wedge U \leq V$$

where $U \leq V$ relates to an arbitrary total ordering of the nodes. The algorithm is shown in Fig. 1 using this notation.

The outermost loop iterates over a number of phases, each of which considers a 1-factorization of the set of *active nodes* A that have not yet exchanged all their messages. The second loop iterates over the 1-factors G_A^i of G_A. The parallel loop considers all node pairs (U, V) that are neighbors in the given 1-factor G_A^i. The node ordering $U \preceq V$ is used to conveniently describe the message exchange between processors on node U and processors on nodes V necessary for reestablishing the invariant for the outermost loop after 'done' has been increased to 'current'. When $U = V$, the bidirectional exchange is replaced by a unidirectional send because otherwise, intra-node messages would be transmitted twice. Sending m_{uu} from u to u means copying m_{uu} from source buffer to destination buffer of u.

Theorem 1. *Algorithm Hierarchical-AllToAll performs a personalized all-to-all exchange in a number of steps equal to the maximal number of messages that the processors in a node have to send.*

Algorithm Hierarchical-AllToAll:

```
A ← {0, ..., N − 1}                                          // set of active nodes
done ← 0
while A ≠ ∅ do                                               // phase
    loop invariant: ∀(U, V) ∈ G : ∀u ∈ U, v ∈ V :
        (U ⪯ V ∧ 0 ≤ l(u) < done) ⇒ m_uv and m_vu have been delivered
    current ← min{size(U) | U ∈ A}
    for i = 0, ..., |A| − 1 do                               // round
        for all (U, V) ∈ G^i_A where U ⪯ V pardo
            for each u ∈ U, done ≤ l(u) < current do
                for each v ∈ V do                            // step
                    if U = V then send m_uv from u to v
                    else exchange m_uv and m_vu between u and v
    done ← current
    A ← A \ {U | size(U) = done}
```

Fig. 1. The hierarchical factor algorithm.

Proof (Outline). Regarding correctness, let $0 = S_0 < S_1 < \ldots < S_k$ be the sequence of different node sizes. The algorithm performs k phases. In phase i nodes U with $\text{size}(U) \geq S_i$ are active. In particular, the outer loop terminates. Furthermore, at the end of the algorithm $\text{done} = \max_{U \in \{0,\ldots,N-1\}} \text{size}(U)$. The loop invariant implies that all messages have been exchanged.

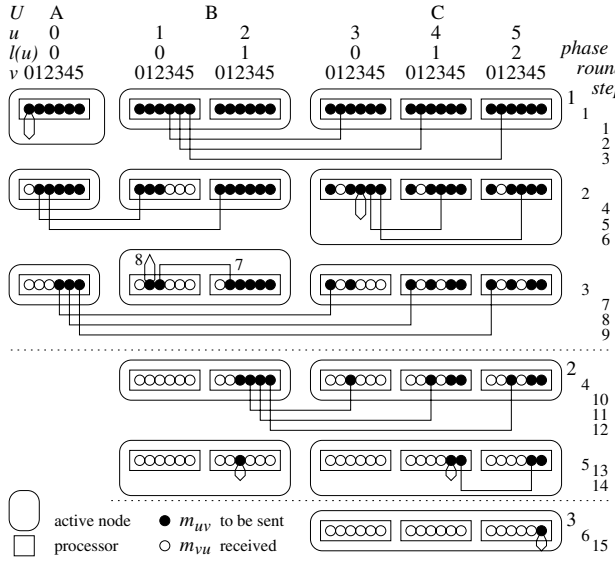

Fig. 2. Example execution of algorithm Hierarchical-AllToAll for three nodes with size 1, 2, and 3 respectively. The algorithms goes through 3 phases, and 3, 2 and 1 rounds respectively are required, for a total of 15 steps. Step 6 of phase 1 in which no inter-node communication takes place, can easily be moved to the end of the computation.

The bound on the number of steps follows since all nodes with the maximum number of processors are participating in a communication in every step and because no message is sent twice. ■

The reason why the algorithm is not optimal in all cases is that a node paired with itself communicates only unidirectionally in each step. If at the same step two other nodes with maximum number of processors are paired, they communicate bidirectionally and hence take longer to complete a round. This is not optimal since at least in some cases there are schedules which avoid such situations. However, there are only few such inefficient steps: Consider a node U with maximal number $n = \text{size}(U)$ of processors. Our algorithm performs pn steps. At most n^2 of these steps — a fraction of n/p — can be inefficient for node U. Hence, the inefficient steps are few compared to the efficient steps for $p \gg n$.

Although the algorithm was formulated for single-ported communication using 1-factorizations, generalizations to multi-ported communications are possible. The same basic scheme applies if the complete graph is decomposed into graphs with degrees at most k or into permutations (directed cycles). Decomposition into permutations is particularly interesting since several all-to-all algorithms for non-fully connected networks are known that are based on this approach [7,2,10].

References

1. J. Bruck, C.-T. Ho, S. Kipnis, E. Upfal, and D. Weathersby. Efficient algorithms for all-to-all communications in multiport message-passing systems. *IEEE Transactions on Parallel and Distributed Systems*, 8(11):1143–1156, 1997.
2. S. E. Hambrusch, F. Hameed, and A. A. Khokar. Communication operations on coarse-grained mesh architectures. *Parallel Computing*, 21:731–751, 1995.
3. F. Harary. *Graph Theory.* Addison-Wesley, 1967.
4. L. P. Huse. MPI optimization for SMP based clusters interconnected with SCI. In *7th European PVM/MPI User's Group Meeting*, volume 1908 of *Lecture Notes in Computer Science*, pages 56–63, 2000.
5. N. T. Karonis, B. R. de Supinski, I. Foster, W. Gropp, E. Lusk, and J. Bresnahan. Exploiting hierarchy in parallel computer networks to optimize collective operation performance. In *Proceedings of International Parallel and Distributed Processing Symposium (IPDPS'2000)*, pages 377–384, 2000.
6. P. Sanders and R. Solis-Oba. How helpers hasten h-relations. *Journal of Algorithms*, 41:86–98, 2001.
7. D. S. Scott. Efficient all-to-all communication patterns in hypercube and mesh topologies. In *Sixth Distributed Memory Computing Conference Proceedings*, pages 398–403, 1991.
8. S. Sistare, R. vandeVaart, and E. Loh. Optimization of MPI collectives on clusters of large-scale SMPs. In *Supercomputing*, 1999. `http://www.supercomp.org/sc99/proceedings/techpap.htm\#mpi`.
9. M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra. *MPI – The Complete Reference*, volume 1, The MPI Core. MIT Press, second edition, 1998.
10. Y. Yang and J. Wang. Optimal all-to-all personalized exchange in self-routable multistage networks. *IEEE Transactions on Parallel and Distributed Systems*, 11(3):261–274, 2000.