two-level HPAM with an $r1 \times c1$ processor mesh in the first level and an $r2 \times c2$ processor mesh in the second level. With or without hardware support, the above collective communication operations can be performed by partitioning the second level into sub-meshes. Without hardware support, the optimal sub-mesh size in the second level for the broadcast operation was found to be $\frac{r_2}{2} \times c_2$ regardless of the message size and the ratio of the communication cost between the first and the second levels. Similarly, the optimal sub-mesh size for the broadcast with hardware support was found to be $\frac{r_2}{2} \times 1$. A reduction of nearly 50% in execution time can be achieved with the inter-level hardware support. Furthermore, this gain is sustained at nearly a constant rate as the size of the message increases or as the ratio of communication cost between the level increases. A slight increase in gain was observed as the size of the machine increased.

Without hardware support the optimal sub-mesh size $(p_2 \times q_2)$ for the scatter and gather operations satisfies $q_2 \leq \frac{c_2}{c_1}$. This optimal sub-mesh size varies with the size of the message and the size of the processor meshes in the two levels of the machine.

For the scatter operation and with hardware support for inter-level multicast, the size of the optimal sub-mesh size also varies. Furthermore, the gain achieved by using the inter-level hardware support rapidly decreases as the message size increases and as the number of processors in the machine increases.

Future work includes the implementation of the algorithms described in this study as part of the message passing library of a simulator (HPAM_Sim [9]) of HPAM-like machines.

References

[1] A. Agrawal. Limits on Interconnection Network Performance. *IEEE trans. Par. and Dist. Systems*, 2(4):398–412, October 1991.

[2] Almasi, G.S. and Gottlieb, A. *Highly Parallel Computing.* The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1994.

[3] Barnett, M., Payne D., and others. *Optimal Broadcasting in Mesh-Connected Architecture.* Tech. Rep. TR-91-38, CS Dept., The University of Texas at Austin, 1991.

[4] Barnett, M., Payne D., and others. Broadcasting on Meshes with Wormhole Routing. *Par. and Dist. Computing*, 35(2):111–122, 1996.

[5] Ben-Miled, Z. *A Hierarchical Heterogeneous Solution to High Performance Cost-Efficient Computing. Ph.D. Thesis,* School of ECE, Purdue University, June 1997.

[6] Ben-Miled, Z., Eigenmann, R., Fortes, J.A.B., and Taylor, V. Hierarchical processors-and-memory architecture for high performance computing. *Frontiers of Massively Parallel Computation Symp.*, pages 138–145, October 1996.

[7] Ben-Miled, Z. and Fortes, J.A.B. A heterogeneous hierarchical solution to cost-efficient high performance computing. *Par. and Dist. Processing Symp.*, pages 138–145, October 1996.

[8] Ben-Miled, Z., Fortes, J.A.B., Eigenmann, R., and Taylor, V. A simulation-based cost-efficiency study of hierarchical heterogeneous machines for compiler and hand parallelized applications. *to appear in 9th Int. Conf. on Par. and Dist. Computing and Systems*, October 1997.

[9] Ben-Miled, Z., Fortes, J.A.B., Eigenmann, R., and Taylor, V. Towards the design of a heterogeneous hierarchical machine: A simulation approach. *30th Simulation Symp.*, pages 126–136, April 1997.

[10] Intel Supercomputer Systems Division. *Paragon System Overview.* Intel Corp., Santa Clara, CA, 1994.

[11] McKinley, P.K., Tsai, Y.J., and others. *A Survey of Collective Communication in Wormhole-Routed Massively Parallel Computers.* Tech. Rep. MSU-CPS-94-35, Michigan State Univ., 1994.

[12] McKinley, P.K., Tsai, Y.J., and Robinson, D. Collective Communication in Wormhole-routed Massively Parallel Computers. *IEEE Computer*, pages 39–50, December 1995.

[13] McKinley, P.K., Xu, H., and others. Unicast-Based Multicast Communication in Wormhole-Routed Networks. *IEEE Trans. on Par. and Dist. Systems*, 5(12):1254–1265, December 1994.

[14] Mitra, P., Payne, D., and others. Fast Collective Communication Libraries, Please. *Proceedings of the Intel Supercomputing Users' Group Meeting*, 1995.

[15] Moore, S.Q. and Ni, L.M. The effects of Network Contention on Processor Allocation Strategies. *Int. Par. Processing Symp.*, pages 268–273, April 1996.

[16] Tsai, Y.J. and McKinley, P.K. A dominating set model for broadcast in all-port wormhole-routed 2d mesh networks. *Eighth ACM Int. Conf. on Supercomputing*, pages 126–135, July 1993.

[17] Watts, J. and van de Geijn, R. A pipelined broadcast for multidimensional meshes. *Parallel Processing Letters*, 5(2):281–292, 1995.

first scenario, the same final message size is assumed for the processors in the first as well as the second level. In the second scenario, the final message size of the processors in the first level is $Q \cdot S$ where S is the size of the final message of the processors in the second level.

When the processors in the first and second level receive a message of the same size, the original message in the source processor of the scatter operation is $\overline{S} = (r_1 \cdot c_1 + r_2 \cdot c_2) \cdot S$. The scatter operation can be performed by distributing the message across the processors in the row of the source processor in the first level. At the end of this step each processor in this row holds a message of size $r_1 \cdot S + r_2 \cdot f_2 \cdot S$. In this expression $r_1 \cdot S$ corresponds to the size of the message to be scattered across the processors in the columns of the first level. The second term corresponds to the size of the message destined to the processors in the second level. In order to optimize the execution time of the overall scatter operation, the scatter algorithm has to perform the inter-level transfer as early as possible. From the analysis of previous cases (Equation 14 and 15), it was established that the optimal sub-mesh size in the second level satisfies $q_2 \leq f_2$. Thus, all the processors in the row of the source processor in the first level participate in the initial step of the scatter operation. This step is then followed by the inter-level transfer. Finally the intra-level scatter in the first and second level proceed concurrently.

All the tests conducted on different HPAM configurations and using a varying values of Q indicate the same behavior seen in previous cases of the scatter operation. Namely, large sub-mesh sizes should be used with small messages and small sub-mesh sizes should be used with large messages.

In the presence of a hardware support for inter-level multicast the scatter operation follows similar steps to the hybrid-based scatter operation. However, the multicast is used for the inter-level transfer.

Figure 10 compares the hybrid and the multicast-based scatter approaches when $Q = 8$ for a selected set of test cases. The behavior depicted in this figure is consistent with previous observations. The scatter operation benefits most from hardware support of inter-level multicast when the size of the message is small. Furthermore, these figures show that as the size of the processor mesh in the second level increases the percent gain in execution time due to the use of the inter-level multicast hardware support decreases.

When the processors of the first level and second level receive a message of size $Q \cdot S$ and S, respectively, the size of the optimal sub-mesh is larger than

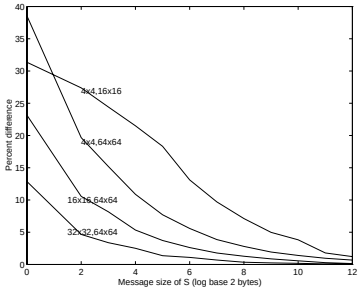

Figure 10: Percent difference between the optimal hybrid and the optimal multicast scatter when $Q = 8$.

that of the corresponding same size message case, for the same range of message sizes. Thus, the scatter algorithm tries to compensate for different message sizes in the two levels by increasing the size of the sub-mesh in the second level. This trend is applicable to all the cases tested and to the hybrid as well as the multicast based scatters.

5 Gather

A gather operation is an all-to-one communication where one processor receives a message from all the other processors [11]. This operation is the exact inverse of the scatter operation. If the scatter operation is started in reverse order, that is, each send is replaced by a receive and each receive is replaced by a send, then the operation becomes a gather. Furthermore, , $_{12}$ = , $_{21}$ and $\Omega_{12} = \Omega_{21}$. Thus all the equations related to the hybrid scatter (Equations 10, 12 and 13) and their associated figures and results also apply to the gather operation.

The use of a multicast hardware support for gather requires that the inter-level routers be capable of combining messages from several incoming channels into one message and sending it on an outgoing channel. Adding such features to the inter-level routers requires the addition of hardware logic and buffer space to the router. If such a router is available then all the equations and results related to the multicast-based scatter (Equations 11 and 14) also apply to the multicast-based gather.

6 Conclusions and Future Work

In this study the X-Y dimension ordered routing was extended to a multilevel heterogeneous machine (HPAM). This extended routing inherits the features of the traditional X-Y dimension ordered routing in homogeneous 2D meshes.

Using this routing, several implementations of broadcast, scatter and gather operation were analyzed. Additionally, each of the above collective communication operations was studied with and without hardware support for inter-level multicast within a

a is the row of the source processor and $b = \lfloor \frac{j \cdot q_2}{f_2} \rfloor$ ($f_2 = \frac{c_2}{c_1}$ as defined in Equation 1). The first step is called partial scatter because some of the processors in the row of the source processor may not participate in the first step. In order to determine the number of processors involved in the partial scatter, two cases are considered:

case a: $q_2 \leq f_2$

In this case all the processors in the row of the source processor are involved in the partial scatter. Furthermore, at the end of this partial scatter each processor holds a message of size $r_2 \cdot \frac{c_2}{c_1} \cdot S$. The execution time of the scatter operation in this case is given by:

$$t_h^a = \underbrace{B_{11}^S(lc_1, r_2 \cdot \frac{c_2}{c_1}(c_1 - 1)) + \frac{\Lambda}{2} lc_1^+}_{\text{intra-level 1}} + \underbrace{\frac{c_2}{c_1 \cdot q_2} \cdot \frac{r_2}{p_2} \left[B_{12}^S(1, p_2 \cdot q_2) + \Lambda(r_1 + \frac{r_2 - p_2}{2 \cdot p_2}) \right]}_{\text{inter-level}} + \underbrace{B_{22}^S(lp_2 + lq_2, p_2 \cdot q_2 - 1) + \frac{\Lambda}{2} \left[lp_2^+ + lq_2^+ \right]}_{\text{intra-level 2}} \quad (12)$$

case b: $q_2 > f_2$

In this case the number of processors in the row of the source processor in the first level involved in the partial scatter is $\frac{c_2}{q_2}$. Furthermore, at the end of this partial scatter each processor holds a message of size $r_2 \cdot q_2 \cdot S$ and the execution time of the scatter operation is given by:

$$t_h^b = \underbrace{B_{11}^S(lc_2 - lq_2, r_2(c_2 - q_2)) + \Lambda(lc_2 - lq_2)(lc_1 + \frac{lc_2 - lq_2 - 1}{2})}_{\text{intra-level 1}} + \underbrace{\frac{r_2}{p_2} \left[B_{12}^S(1, p_2 \cdot q_2) + \Lambda(r_1 + \frac{r_2 - p_2}{2 \cdot p_2}) \right]}_{\text{inter-level}} + \underbrace{B_{22}^S(lp_2 + lq_2, p_2 \cdot q_2 - 1) + \frac{\Lambda}{2} \left[lp_2^+ + lq_2^+ \right]}_{\text{intra-level 2}} \quad (13)$$

Several HPAM configurations were tested when $Q = 2$, 8 and 32 and the results obtained indicate that the sub-mesh size of choice satisfies $q_2 \leq f_2$. Thus, all the rows of the source processor in the first level are involved in the partial scatter. Additionally, the height of the optimal sub-mesh depends on the message size. Small messages have a corresponding optimal sub-mesh with large height. Large messages have a corresponding optimal sub-mesh with small height.

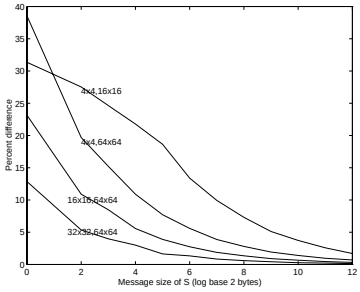

Figure 9: Percent difference between the optimal hybrid and the optimal multicast scatter when $Q = 8$.

With hardware support for inter-level multicast and given that the destination of the scatter is the processors in the second level only, the execution time of the scatter operation is given by:

$$t_m = \underbrace{B_{11}^S(lc_1 + \frac{c_2}{c_1} \cdot r_2(c_1 - 1)) + \frac{\Lambda}{2} \cdot lc_1^+}_{\text{intra-level 1}} + \underbrace{\frac{r_2}{p_2} \left[B_{12}^S(1, p_2 \cdot \frac{c_2}{c_1}) + \Lambda(r_1 + \frac{r_2 - p_2}{2 \cdot p_2}) \right]}_{\text{inter-level}} + \underbrace{B_{22}^S(lp_2, p_2 - 1) + \frac{\Lambda}{2} lp_2^+}_{\text{intra-level 2}} \quad (14)$$

All the test conducted in this case of the scatter operation indicate that large sub-meshes should be used with small messages and small sub-meshes should be used with large messages.

Figure 9 shows the difference in execution time between the hybrid-based scatter and the multicast-based scatter with the optimal sub-mesh size when $Q = 8$ for a selected set of test cases. In this figure each curve is identified by the size of the processor mesh in the first level followed by the size of the processor mesh in the second level. This figure indicates that as the message size increases, the gain due to the use of hardware support for inter-level multicast decreases.

For all the scatter operation cases studied above, the size of the message at the final processor destination is always the same for all the processors. A different situation arises when the destination of the scatter operation is the processors in both the first and second levels. Since an HPAM machine is heterogeneous, there is a different type of processor in each level. Particularly, the processors in the first level are faster than the processors in the second level. Thus, for load balancing purposes the size of the final message for the processors in the first level may be larger than the final message for the processors in the second level. Two possible scenarios are considered. In the

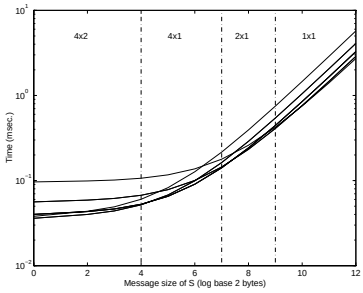

Figure 6: Hybrid scatter on a 4×4 mesh when $Q = 2$.

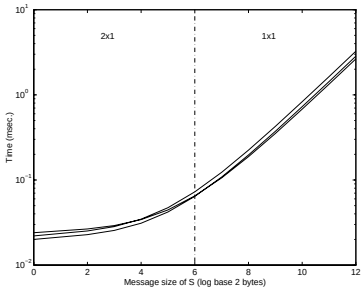

Figure 7: Multicast scatter on a 4×4 mesh when $Q = 2$.

Conceivably, a router switch can be built so that a message is distributed across its outgoing channels for the multicast-based scatter. However, this requires additional logic hardware and possibly buffer space in the inter-level router. In this study, the inter-level router for the multicast-based scatter is assumed to be capable of only replicating entire messages across outgoing channels. Given this multicast hardware, each sub-mesh in the second level is one processor wide ($q_2 = 1$) and the height (p_2) can vary. The execution time for multicast-based scatter operation is given by:

$$t_m = \underbrace{\frac{r_2}{p_2}\left[B^S_{12}(1, p_2 \cdot c_2) + \Lambda\right]}_{\text{inter-level}} + \underbrace{B^S_{22}(lp_2, p_2 - 1) + \frac{\Lambda}{2} lp_2^+}_{\text{intra-level}} \quad (11)$$

Figure 7 shows the execution time of the multicast-based scatter when the size of the processor mesh in the second level is 4×4 for $Q = 2$. This figure indicates that the optimal sub-mesh sizes are $p_2 \times q_2 = 2 \times 1$ and $p_2 \times q_2 = 1 \times 1$ for small and large message sizes, respectively. Additional tests conducted on large mesh sizes and varying values of Q, indicate that a small sub-mesh size is optimal for large messages. Furthermore, the size of the optimal sub-mesh increases as the message size decreases. Additionally, the difference in execution time of the multicast-based scatter due to different size sub-meshes in the second level decreases with increasing message sizes for large mesh sizes in the second level.

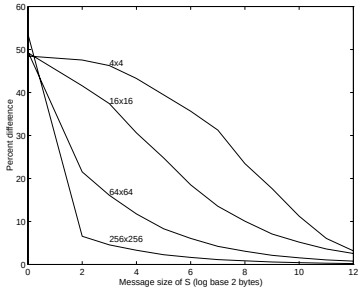

Figure 8: Percent difference between the optimal hybrid and the optimal multicast scatter when $Q = 8$.

Unlike the broadcast operation, the gain in execution time due to the use of the hardware support for inter-level multicast decreases as the message size increases. Except when $S = 0$, the percent difference between the optimal hybrid and the optimal multicast-based scatter operations decreases as the size of the mesh increases in the second level. This is illustrated in Figure 8 for $Q = 8$ and when the mesh size in the second level is 4×4, 16×16, 64×64 and 256×256. Furthermore, for a given size mesh, the percent difference slightly increases as the value of Q increases. However, the percent difference is more sensitive to changes in the size of the processor mesh in the second level than to changes in the value of Q. Additionally, as the size of the message increases, the percent gain decreases. Small to medium meshes (4×4 and 16×16) experience a gradual decrease in percent difference, whereas large processor meshes (64×64, 256×256) experience a sharp decrease in percent difference as the message size increases (Figure 8). Thus, providing hardware support for inter-level multicast for the scatter operation is justified for meshes of sizes up to 64×64. The percent difference is less than 8% for all values of Q when $S > 0$ and the mesh size is 256×256 in the second level.

4.2 Scatter in two-level HPAM machines with multiple processors in the first level

Without hardware support for inter-level multicast and when the destination of the scatter is the processors of the second level only, the scatter operation can be performed by first executing a partial scatter in the row of the source processor. This step is followed by an inter-level transfer and a scatter operation in the second level. As previously mentioned in Section 3, since the size of each sub-mesh in the second level is $p_2 \times q_2$, the root of all these sub-meshes is defined by the set $\cup_{i=0}^{\frac{r_2}{p_2}-1} \cup_{j=0}^{\frac{c_2}{q_2}-1} (i \cdot p_2, j \cdot q_2)_2$. Each subset $\cup_{i=0}^{\frac{r_2}{p_2}-1} (i \cdot p_2, j \cdot q_2)_2$ is serviced by the same processor in the first level, namely, processor $(a, b)_1$, where

for the worst case scenario. Minor modifications are required in the inter-level transfer time to account for other cases. For example in the above equation, if the source processor is in row x of the first level, then the term $\Lambda \cdot r_1$ should be replaced by $\Lambda \cdot (r_1 - x)$.

The above equation applies to the general case when the destination of the broadcast consists of the processors in the first and second level. When the destination of the broadcast consists of the processors in the second level only, the first term of the maximum operation in Equation 7 reduces to zero and the cost of the operation is dictated by the intra-level broadcast in the second level rather than the intra-level broadcast in the first level. In most instances (if the mesh size in the first level is smaller than the mesh size in the second level), even when the destination of the broadcast consists of the processors in the first as well as the second level, this is going to be the case.

In the presence of the hardware support for replication discussed above, the steps involved in the broadcast are (1) Perform a partial intra-level broadcast in the first level, (2) Perform an inter-level multicast and (3) Perform the remaining part of the binary-tree based broadcast in the first level concurrently with the binary-tree broadcast in the second level.

The size of the initial sub-mesh in the second level was varied for the three values of Q (i.e., 2, 8 and 32). As was found in section 3.1, the optimal size sub-mesh is $\frac{r_2}{2} \times 1$. The execution time of this broadcast using the optimal size sub-mesh is given by:

$$t_m = \underbrace{lc_1\left(A^S_{11} + \frac{\Lambda}{2}(lc_1+1)\right)}_{\text{intra-level (level1 part1)}} + \underbrace{2\left[A^S_{12} + \Lambda(r_1 + \frac{r_2}{4})\right]}_{\text{inter-level}} + \max\Big(\underbrace{lr_1 \cdot A^S_{11} + \frac{\Lambda}{2} lr_1^+}_{\text{intra-level (level 2 part2)}}, \underbrace{(lr_2 - 1)A^S_{22} + \frac{\Lambda}{2} lr_2^-}_{\text{intra-level (level 2)}}\Big) \quad (8)$$

For an HPAM machine with $r_1 \times c_1 = 4 \times 4$ and $r_2 \times c_2 = 16 \times 16$, the multicast-based broadcast has an execution time less than that of the hybrid-based approach by more than 31%, 41% and 45% when $Q =$ 2, 8, and 32, respectively. This percent difference increases as the size of the mesh in the second level and the value of Q increases. For example, percent differences of 38% to 46% were obtained when the size of the mesh in the second level was 64×64.

4 Scatter

Unlike broadcast, scatter entails sending a different message from a single node to every node [11]. A scatter can be efficiently implemented in 2D homogeneous meshes using a binary tree algorithm similar to the one used for the broadcasting. The difference between the two algorithms is that the message forwarded to the root of the sub-mesh at every step consists of the collection of all the messages destined to all the processors in the sub-mesh.

Let $\overline{S}$ represent the size of the original message (i.e., the collection of all the messages to be sent by the source node) and S represent the size of the message at each final destination of the scatter. Thus, in the homogeneous 2D mesh, $\overline{S} = r_l c_l S$. For notational simplicity, let $B^S_{lk}(a,b)$ denote the expression $a \cdot {}_{,lk} + b \cdot S \cdot \Omega_{ll}$. Using this notation, the execution time of the scatter in a 2D homogeneous mesh is given by:

$$t_b = B^{\overline{S}}_{ll}(lr_l + lc_l, \tfrac{r_l \cdot c_l - 1}{r_l \cdot c_l}) + \tfrac{\Lambda}{2}\left[lr_l^+ + lc_l^+\right] \quad (9)$$

If the last term of Equation 11 ($\frac{\Lambda}{2}\left[lr_l^+ + lc_l^+\right]$) is ignored, the execution time of the scatter operation is the same as the one reported in [14].

An analysis approach similar to the one used for the broadcast is adopted for the scatter in a two level HPAM machine. The case where there is only one processor in the first level (i.e., the source processor) will be discussed first in the following subsection. The case when the first level has more than one processor is discussed in Subsection 4.2.

4.1 Scatter in two-level HPAM machines with one processor in the first level

Let the processor mesh in the second level be partitioned into $p_2 \times q_2$ sub-meshes. The execution time of the scatter operation is given by:

$$t_h = \underbrace{\frac{r_2}{p_2} \cdot \frac{c_2}{q_2}\left[B^S_{12}(1, p_2 \cdot q_2) + \Lambda(1 + \frac{r_2 - p_2}{2})\right]}_{\text{inter-level}} + \underbrace{B^S_{22}(lp_2 + lq_2, p_2 \cdot q_2 - 1) + \frac{\Lambda}{2}\left[lp_2^+ + lq_2^+\right]}_{\text{intra-level}} \quad (10)$$

Figure 6 shows the execution time of the scatter operation as given by the above equation when the mesh size in the second level is 4×4 for $Q = 2$. This figure is partitioned into different regions and for each region the optimal sub-mesh size is indicated. Unlike the broadcast, the optimal sub-mesh size for the scatter varies with the size of the message and the value of Q. Additional tests were conducted on mesh sizes 4×4, 8×8, 16×16, 64×64, and 256×256 with varying values of Q (i.e. $Q =$ 2,8, and 32). The trend indicated by these tests is that large sub-meshes should be used with small messages and small sub-meshes should be used with large messages. Furthermore, partitioning across the columns first is better than partitioning across the rows. As in the broadcast case, this is due to the fact that the inter-level interconnect favors narrow width sub-meshes.

the size of the mesh, the size of the message or the value of Q. This result is counter-intuitive. Since the communication parameters in the first level are smaller than those of the second level by a factor of Q, it was expected that an increasing number of sub-meshes should be used as the value of Q increases. However, (1) as the value of Q increases the inter-level communication becomes dominated by the receiving side (i.e., the second level); (2) the binary tree broadcast in the second level cuts the communication time in half at each step; (3) the execution time of an inter-level communication step is greater than half that of an intra-level communication step in the second level. These observations justify the results obtained. With a binary tree approach the execution time of the broadcast is reduced by half at every step. However, with the use of inter-level communication as a substitute for some of the steps in the intra-level broadcast, the best execution time that can be achieved is higher than half the execution time of any given step in the intra-level broadcast.

It is possible to reduce the execution time of the inter-level broadcast with additional hardware support for multicast. There are several proposals of hardware support for multicast in homogeneous machines. These hardware support approaches include intermediate reception and replication [11]. Intermediate reception allows a message to be delivered to the local processor while being forwarded to the next processor in the path. Replication is a hardware capability that allows an incoming message to be replicated across several outgoing channels.

In order to keep the hardware cost and complexity within a practical range, a simple replication-based multicast hardware support is proposed. This hardware would allow a message to be broadcasted along all the inter-level outgoing channels *simultaneously*. With such a hardware support, the execution time of the broadcast is given by:

$$t_m = \underbrace{\frac{r_2}{p_2}\left[A_{12}^S + \Lambda(1 + \frac{r_2 - p_2}{2})\right]}_{\text{inter-level}} + \underbrace{lp_2 \cdot A_{22}^S + \frac{\Lambda}{2} lp_2^+}_{\text{intra-level}} \quad (6)$$

The notation "multicast" and "hybrid" will be used in the remainder of this paper to distinguish between an implementation that requires inter-level hardware support for multicast and an implementation that does not require this hardware support, respectively. In order to fully utilize the hardware support for inter-level multicast, the processor mesh in the second level is partitioned into $p_2 \times 1$ sub-meshes (i.e., $q_2 = 1$). The execution time of the multicast-based broadcast as defined by Equation 6, was analyzed for several test cases (i.e., 4×4, 8×8, 16×16, 64×64 and 256×256; Q = 2,8,32). This analysis revealed that the optimal sub-mesh size is $\frac{r_2}{2} \times 1$.

The added hardware reduces the execution time of the broadcast by more than 43% (i.e., 100· [optimal hybrid - optimal multicast] / optimal hybrid]) for a 4×4 mesh in the second level. A similar comparison for larger size meshes revealed that the advantage of the multicast-based broadcast slightly increases as the size of the mesh in the second level and the value of Q increases. The percent difference between the two approaches is more than 46% for 16×16, 64×64 and 256×256 mesh sizes in the second level. Thus, the inter-level multicast hardware support is a desirable feature in multilevel machines such as HPAM if it simple and easy to implement.

3.2 Broadcast in two-level HPAM machines with multiple processors in the first level

The most efficient algorithm for a broadcast in an HPAM machine with more than one processor in the first level is one that overlaps most of the broadcast in the first level with the broadcast in the second level.

In the absence of inter-level multicast support, and given that the optimal number of sub-meshes is two, the different steps in the algorithms are: 1- Perform the first part of broadcast in the first level. This step creates two sub-meshes in the first level. 2- Perform the inter-level broadcast. This step creates two sub-meshes in the second level. 3- Perform the remaining part of the binary-tree-based broadcast in the first level concurrently with the binary tree broadcast in the second level. In the worst case (i.e., when the source processor is in the first row of the first level), the execution time of this algorithm is given by :

$$t_h = \underbrace{A_{11}^S + \Lambda\frac{c_1}{2}}_{\text{intra-level (level1 part1)}} + \underbrace{A_{12}^S + \Lambda \cdot r_1}_{\text{inter-level}} + \max(\underbrace{(lr_1 + lc_1 - 1)A_{11}^S + \frac{\Lambda}{2}\left[lr_1^+ + lc_1^-\right]}_{\text{intra-level (level1 part2)}}, \underbrace{(lr_2 + lc_2 - 1)A_{22}^S + \frac{\Lambda}{2}\left[lr_2^+ + lc_2^-\right]}_{\text{intra-level (level2)}}) \quad (7)$$

For the remainder of this study when there is more than one processor in the first level, the worst case for the inter-level transfer is always assumed. This worst case occurs when the source processor is in the first row of the first level. When applicable, all the equations in the remainder of this study are derived

total time for this broadcast operation is:

$$t_b = \quad (lr_l + lc_l)A_{ll}^S + \frac{\Lambda}{2}\left[lr_l^+ + lc_l^+\right] \tag{4}$$

The implementation of a broadcast operation in a two-level HPAM machine is discussed in the following two subsections. The first subsection considers the case when there is only one processor in the first level. The second subsection considers the case when there is more than one processor in the first level.

3.1 Broadcast in two-level HPAM machines with one processor in the first level

Figure 4 shows possible broadcasting algorithms from the first level to the second level of a two-level HPAM machine with one processor in the first level. Unlike the homogeneous case, in the heterogeneous case, the algorithm starts by partitioning the physical mesh in the second level into *initial sub-meshes* of equal size. Within each initial sub-mesh a binary-tree-based broadcast is used. Furthermore, the processor in the first level broadcasts the message to the root of each initial sub-mesh using a "brute-force" method (i.e., the processor in the first level sequentially sends a message to each root). Figures 4a, 4b and 4c depict this algorithm when the second level is a 4×4 mesh and the initial sub-mesh sizes are 1×1, 4×2 and 4×1, respectively. This approach will be denoted "hybrid" in the remainder of this paper because it consists of a combination of the "brute-force" and the binary tree approaches. When there is only one sub-mesh, the algorithm consists of sending the broadcasted message to a processor in the second level (e.g. $(0,0)_2$), then using the binary tree broadcasting algorithm to broadcast the message to the remaining nodes in the second level. When the sub-mesh size is 1×1, then the hybrid approach is the same as the "brute-force" method.

Let the mesh in the second level be partitioned into $p_2 \times q_2$ sub-meshes. Thus, the total number of sub-meshes in the second level is $\frac{c_2}{p_2} \cdot \frac{r_2}{q_2}$. Furthermore, the set of roots of these sub-meshes consists of $\cup_{i=0}^{\frac{r_2}{p_2}-1} \cup_{j=0}^{\frac{c_2}{q_2}-1} (i \cdot p_2, j \cdot q_2)$ of processors. The number of hops (η) from the processor in the first level to a given root $(i \cdot p_2, j \cdot q_2)$ of a sub-mesh is $i \cdot p_2 + 1$. The number of hops does not depend on the width of the sub-mesh because of the topology of the interconnect used between the two levels. This property makes partitioning the processor mesh of the second level along columns result in smaller inter-level transfer latencies than if the mesh is partitioned along rows. The total broadcast operation using the hybrid approach consists of two major steps. In the first step the message is broadcasted to the root of each sub-mesh (inter-level transfer). In the second step, a binary-tree-based broadcast is performed locally within each sub-mesh (intra-level broadcast). The execution time of these two components is given by:

$$t_h = \underbrace{\frac{r_2}{p_2} \cdot \frac{c_2}{q_2}\left[A_{12}^S + \Lambda(1 + \frac{r_2 - p_2}{2})\right]}_{\text{inter-level}} + \underbrace{(lp_2 + lq_2)A_{22}^S}_{\text{intra-level}} + \underbrace{\frac{\Lambda}{2}\left[lp_2^+ + lq_2^+\right]}_{\text{intra-level}} \tag{5}$$

Parameter	value
$C_{send}^1 = C_{recv}^1(\mu s)$	2
$C_{send}^2 = C_{recv}^2(\mu s)$	$2 \cdot Q$
$\frac{\lambda_1'}{W_1'}(\mu s/byte)$	0.01
$\Lambda = \frac{\lambda}{W}(\mu s/byte)$	0.01
$\frac{\lambda_2'}{W_2'}(\mu s/byte)$	$0.01 \cdot Q$

Table 1: Values of the communication parameters.

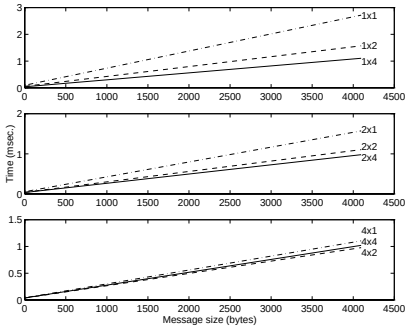

Figure 5: Hybrid broadcast on a 4×4 mesh when $Q = 2$.

Table 1 shows the values of the communication parameters used in this study. In this table Q represents the ratio of the communication parameters in the second level to those of the first level. Figures 5 shows the execution time of an inter-level broadcast operation when $r_2 \times c_2 = 4 \times 4$ and $Q = 2$. In this figure the initial sub-mesh size corresponding to each curve is indicated by $p_2 \times q_2$. For all values of Q tested the best sub-mesh size in the second level of the example two-level HPAM machine is 4×2. This figure also shows that as expected partitioning the mesh in the second level along the columns yields smaller execution time than if the mesh is partitioned along the rows.

Additional tests were conducted using several values of Q (i.e., 2, 8, and 32) and $r_2 \times c_2$ (i.e., 8×8, 16×16, 64×64 and 256×256). In all the tests conducted, the "brute force" approach results in worst case performance. Additionally, all of the above tests indicate that the "optimal" partitioning consists of two sub-meshes constructed along the columns of the mesh (i.e., $r_2 \times \frac{c_2}{2}$) in the second level regardless of

that spans two or more levels. Two examples of this extended routing are shown in Figures 3 (b) and 3 (c). The dashed boxes in these examples represent the fact that the first level can be virtually extended to appear as an extension to the second level. The dashed routing paths depict the virtual routing paths, whereas the solid paths represent the physical routing paths.

In general, a message from processor $(i,j)_l$ in level l to processor $(m,n)_{l+1}$ in level $l+1$ is routed along the path: $(i,j)_l \rightsquigarrow (i,\lfloor\frac{n}{f_l}\rfloor)_l \rightsquigarrow (r_l-1,\lfloor\frac{n}{f_l}\rfloor)_l \rightsquigarrow (0,n)_{l+1} \rightsquigarrow (m,n)_{l+1}$. The notation $a \rightsquigarrow b$ denotes a path from a to b along the same direction. Similarly, a message from processor $(m,n)_{l+1}$ to processor $(i,j)_l$ is routed along the path : $(m,n)_{l+1} \rightsquigarrow (m,\chi)_{l+1} \rightsquigarrow (0,\chi)_{l+1} \rightsquigarrow (r_{l-1},j) \rightsquigarrow (i,j)_l$, where

$$\chi = \begin{cases} j \cdot f_l & \text{if } n < j \cdot f_l \\ (j+1)f_l - 1 & \text{if } n > (j+1)f_l - 1 \\ n & \text{otherwise} \end{cases} \quad (2)$$

In all cases the path between source and destination processors is deterministic and unique. In particular, this routing defines not only the path that the message will be traversing within a given level but also the link that the message will take while crossing from one level to the next.

As previously mentioned, X-Y dimensioned ordered routing is deadlock-free in 2D meshes. It can be easily shown that the proposed extended X-Y dimension ordered routing inherits this feature of the traditional X-Y dimension ordered routing [5].

2.4 Latency

For a given machine, the communication latency (t_{comm}) is the sum of four major components [15]: the source processor software startup latency (t_{send}), the hardware transfer latency through the network (t_{net}), the time a message is blocked due to contention (t_{cont}), and the destination processor software startup latency (t_{recv}). For a message of size S bytes and a given level l, $t^l_{send} = C^l_{send} + \frac{\lambda'_l}{W'_l} \cdot S$ and $t^l_{recv} = C^l_{recv} + \frac{\lambda'_l}{W'_l} \cdot S$. In the expression of t^l_{send} and t^l_{recv}, C^l_{send} and C^l_{recv} are constant. In these same expressions, the term that varies with the size of the message (i.e., $\frac{\lambda'_l}{W'_l} \cdot S$) is mainly due to copy operations required during a send or a receive. The evaluation of the network transfer latency (t_{net}) depends on the network topology and, according to the model in [1], it can be estimated as: $t_{net} = \frac{\lambda}{W} \cdot (S+\eta)$, where W is the point to point width between a processor and its nearest neighbor. The parameter λ represents the time it takes to route W bytes between two nearest neighbors. The parameter η is the number of hops between the source and destination.

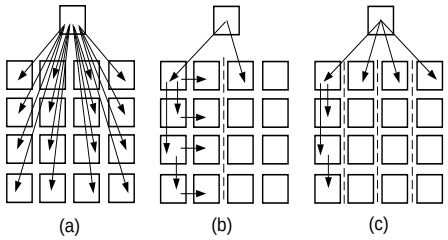

Figure 4: Possible broadcast algorithms for a two-level HPAM machine with initial sub-mesh size (a) 1×1, (b) 4×2 and (c) 4×1.

The term t_{cont} is zero if there is no contention in the network during the transmission of a given message. When there is contention in the network, t_{cont} accounts for the time the message is blocked until the contention is resolved.

By denoting the source and destination levels l and k, respectively, and reordering the parameters, the transfer latency between two processors (where l can be equal to k) is expressed by:

$$t^{lk}_{comm} = , _{lk} + \Omega_{lk} \cdot S + \Lambda \cdot \eta + t_{cont} \quad (3)$$

where $, _{lk} = C^l_{send} + C^k_{recv}$, $\Omega_{lk} = \frac{\lambda'_l}{W'_l} + \frac{\lambda}{W} + \frac{\lambda'_k}{W'_k}$ and $\Lambda = \frac{\lambda}{W}$. In the above equation we make the assumption that λ has the same value within a level as well as across the levels of a given HPAM machine. Similarly, W has the same value within a level and across the levels of a given HPAM machine. Thus, the interconnect between two processors in any level is assumed to be equally fast and to have the same bandwidth.

The communication model defined by Equation 5 will be used throughout the remainder of this study. When $l = k$, Equation 5 corresponds to a homogeneous communication model widely used in previous studies such as [4].

3 Broadcast

An efficient tree-based broadcast algorithm for one-level homogeneous machines was proposed in [3]. This algorithm was used for intra-level broadcast in [8]. For an $r_l \times c_l$ mesh, the algorithm completes in $\log_2(r_l) + \log_2(c_l)$ steps [11]. The first step divides the original mesh into two sub-meshes. Each additional step divides the meshes obtained from the previous step into two sub-meshes. At each step there are no messages communicated across meshes created by the previous step. There are two desirable goals in collective communication operations. The first is to minimize the number of steps; the second is to avoid contention. The above algorithm achieves both goals.

For notational simplicity, given an integer α, let $l\alpha$ denote $\log_2(\alpha)$. Also let $l\alpha^+$ and $l\alpha^-$ denote $\log_2(\alpha)(\log_2(\alpha)+1)$ and $\log_2(\alpha)(\log_2(\alpha)-1)$, respectively. Furthermore, let A^S_{lk} denote $, _{lk} + \Omega_{lk} \cdot S$. The

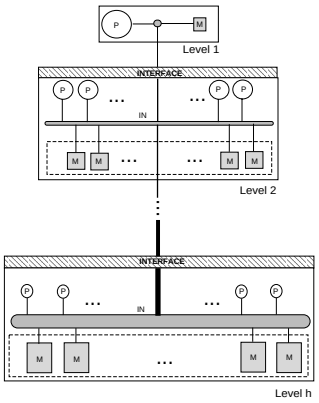

Figure 1: HPAM organization.

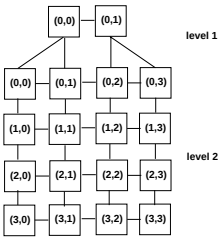

Figure 2: HPAM interconnection example.

The communication model used in this study is described in Sections 2. Sections 3, 4 and 5 are dedicated to the implementation of broadcast, scatter and gather, respectively, in a two-level HPAM machine. Section 6 includes conclusions and future work.

2 Communication Model

This section outlines the different components of the communication model used in this study. These components include network topology, routing scheme and communication cost (or transfer latency).

2.1 Port Model

Routers in multiprocessor systems can be either uni-ported or multi-ported depending on the number of channels directly connected to the local processor [11]. In this study, the uni-port router model is adopted. In the context of this port model, there are two uni-directional channels connected to the local processor. Thus, a processor can either send or receive one message at a time. Additionally, there are also two uni-directional remote channels between a router and another router directly connected to it. These remote channels can carry data in opposite directions at the same time.

2.2 Topology

The interconnection network of any HPAM machine consists of an inter-level and an intra-level processor interconnect. In general, an HPAM organization is not restricted to a given inter-level or intra-level topology. However, in this study each level in HPAM uses a mesh for the intra-level interconnect. Let level l consist of an $r_l \times c_l$ processor mesh. Also, let processors be represented by $(row, column)_{level}$ tuples such that the topmost row and the leftmost column of a given level are labeled zero. Furthermore, let levels be numbered in increasing order starting from the top level. A given processor $(0, a)_l$ in level l is connected to processor $(r_{l-1} - 1, b)_{l-1}$ in level $l - 1$. The relationship between a and b is given by:

$$b = \lfloor \frac{a}{f_l} \rfloor \quad \text{where} \quad f_l = \frac{c_l}{c_{l-1}} \quad \text{and} \quad 0 \leq a < c_l \tag{1}$$

In the above equation $c_{l-1} \leq c_l$. Furthermore, both parameters are powers of two. Figure 2 shows the network topology for an example two-level HPAM machine. The above inter-level network was selected because it allows upper levels to behave as extensions of lower levels while using existing connectivity efficiently and without the need for additional connectivity. Thus, this inter-level interconnect allows a given physical HPAM machine to be configured into different virtual HPAM machines. The importance of such reconfigurability was established in [8].

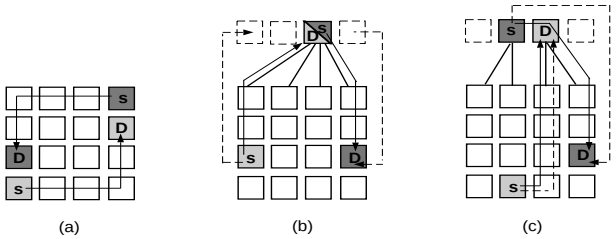

Figure 3: Intra-level and inter-level routing: (a) X-Y dimension ordered routing, (b) inter-level routing when there is only one processor in the first level of the two-level HPAM machine (c) inter-level routing when there are two processors in the first level of the two-level HPAM machine.

2.3 Routing

In each level of the HPAM machine, *X-Y dimension ordered* routing [2] is adopted. An example of this routing for a 4×4 mesh is shown in Figure 3(a). There are several attractive features to X-Y dimension ordered routing which make it widely used in commercial multiprocessors, such as the Intel Paragon [10]. In X-Y dimension ordered routing the path between a source and a destination is fixed and can be statically determined. Furthermore, deadlocks can be avoided by enforcing a strict monotonic order on the dimension in the routing [11]. These features make the routing easy to implement and manage in multiprocessor systems. Furthermore, the path defined by an X-Y dimension ordered routing is the shortest path between the source and the destination.

The routing adopted for the multilevel HPAM machine is an extended X-Y dimension ordered routing

On the Implementation of Broadcast, Scatter and Gather in a Heterogeneous Architecture*

Zina Ben Miled
Department of EE
Purdue University
Indianapolis, IN46202

José A. B. Fortes
School of ECE
Purdue University
West Lafayette, IN 47907-1285

Rudolf Eigenmann
School of ECE
Purdue University
West Lafayette, IN 47907-1285

Valerie Taylor
Department of ECE
Northwestern University
Evanston, IL 60208-3118

Abstract

This paper considers the implementation and evaluation of broadcast, scatter and gather communication operations in heterogeneous machines organized as a Hierarchy of Processor-And-Memory (HPAM). The top levels of the hierarchy consist of a small number of fast processors, whereas the bottom levels consist of a large number of slow processors. Each HPAM level consists of a homogeneous processor mesh. Routing within each level and across levels uses an extended form of X-Y dimension ordered routing. The execution times of three collective communication operations are analytically evaluated in the context of two-level HPAM machines. For each operation, two different alternatives are considered depending on the absence or presence of hardware support. For each alternative, an efficient implementation is characterized. Furthermore, the two alternatives are compared and the gain achieved by using hardware support for these operations is assessed.

1 Introduction

Broadcast, gather and scatter operations are particular cases of one-to-all/all-to-one collective communication operations. The importance of efficient software implementation of collective communication in homogeneous multiprocessors has been established in several previous studies ([12], [14] and references therein). An efficient implementation of a collective communication operation is usually specific to the underlying architecture [12]. Although previous studies provide efficient implementation of several collective communication operations for homogeneous multiprocessors ([13], [16], [17] and references therein), they do not address the issues involved in collective communication for heterogeneous machines.

This study evaluates different broadcast, scatter and gather algorithms for a multilevel hierarchical heterogeneous machine (HPAM: Hierarchical Processor-and-Memory) previously introduced in [7] and [6]. The top levels of HPAM (Figure 1) have a small number of fast processors. These levels can efficiently execute serial and low parallelism sections of an application. The low levels have a large number of slow processors and can efficiently execute portions of code with high degree of parallelism.

A simulation-based study of various HPAM and homogeneous multiprocessors organizations [8] revealed that HPAM can deliver higher performance than homogeneous multiprocessor systems under fixed budget constraints. This study also revealed (due to their frequent use) the importance of efficient implementation of broadcast, gather and scatter across the levels of an HPAM machine. However, while this study used optimized collective communication operations for the homogeneous machines, collective communication in HPAM was not carefully assessed. Actually, in most cases a "brute force" method was used for inter-level collective communication operations. In this paper, "brute force" denotes an approach where the source processor sequentially sends a message to each destination processor.

This study shows that broadcast, scatter and gather across the levels of an HPAM machine can be efficiently implemented. Furthermore, an efficient implementation of these operations in HPAM machines is different from an efficient implementation of the same operations in homogeneous machines. This difference is due to the heterogeneous aspect of the processors and network interface in HPAM machines. This study also evaluates the execution time of the above collective communication operations in the presence of a simple inter-level communication hardware support. The reduction in execution times achieved by the addition of this hardware support in performing a broadcast, a scatter or a gather is evaluated.

*This research was supported in part by NSF grants ASC-9612133, ASC-9612023 and CDA-9617372.