FFT Communications Requirement Optimizations on Massively Parallel Architectures with Local and Global Interprocessor Communications Capabilities

Christopher Lee Kuszmaul

July 23, 1990

Abstract

Fast Fourier Transforms [1], Batcher sorting [2], Cyclic Reduction [3], and a host of other recursively defined, divide and conquer style algorithms can be implemented on massively parallel computers which provide for rapid communications between data elements whose indices differ by a power of two. This paper addresses the general issue of how two different communication mechanisms one Global, and one Local, can provide for hybrid performance that substantially exceeds what either could provide separately. In particular, power of two communications schemes are explored for the MP-1 family of massively parallel computers. By using a combination of the eight way nearest neighbor, toroidally wrapped grid and the Global Router on an MP-1 1200 series computer with 16,384 processors (PEs), the communications requirements for a 16,384 point FFT are shown to require less than 2 milliseconds.

1 Introduction

The focus of this paper is on the interplay between different communications mechanisms on the same computer architecture. In particular, it shows how the Global Router, and the X-Net (eight nearest neighbor communications) on the MP-1 can, combined, achieve superior performance in the execution of the communications portion of the FFT. However, the concept is not restricted to the MP-1, or even to parallel computers.

The fundamental metric of interest is the rate at which information can be exchanged between a given pair of memory locations in a computer architecture. For most computer architectures, there is only one way to perform this exchange. However, the fact of a Global Router, as well as an X-Net on the MP-1, and of gather-scatter hardware in addition to Local memory referencing on the Cray, and row-column highways as well as a NEWS grid on the DAP [4] provide examples where there are two distinct methods for moving data on one machine.

In general then we can assume we have two mechanisms for moving data. Let us assume that the performance of each of the two mechanisms depends on the data permutation requested. If both mechanisms behaved identically under all circumstances, there would be no motivation to include both of them in the architecture. In addition, if one mechanism was always superior (faster) than the other, there would be no motivation to build both.

For now, let us postulate two mechanisms: (1) Global and (2) Local. Where the Local communication mechanism is more sensitive to changes in the permutation requested. For now, let us restrict permutations to the kind that occur in an FFT.

This paper will establish a set of functions associated with the cost of communications associated with the FFT. It will show that the functions are not minimized in simply using the Global Router when it is fastest, and the Local Router otherwise. A better way will be shown, which takes advantage of the fact that a Global Router takes the same amount of time to instantiate a given permutation. In particular, it will show that the Four Step FFT [5] can be implemented in parallel, through the use of the Global Router to rearrange the data, so that the least expensive Local communications are used more often. The paper shows that the Four Step FFT, which calls for only one Global data rearrangement is not necessarily optimal. The assumptions made in the paper to this point are sufficiently broad to require specific experiments. The paper then identifies how, and at what performance, power of two communications can be performed on the MP-1. In particular, mechanisms using the Global Router and the X-Net are identified. The performance of the X-Net depends on how the PEs are enumerated. Three enumerations are examined, and their theoretic and actual performances are established. In addition to power of two communications, the rearrangements derived from the Four Step FFT are required. The paper identifies what the rearrangements are for each of the three enumerations. The paper shows that one of the enumerations (raster scan order) is a poor choice in that it requires too many rearrangements. Actual performance numbers are shown for the rearrangements corresponding to the other two enumerations. It turns out that both rearrangements cost the same on the Global Router, and the cost is low enough to justify their use. The paper concludes that what is called cube embedded order is therefore the best choice of how to enumerate the PEs. It also concludes that the strategy of rearranging the data one time in the course of the FFT computation is indeed desirable within the context of cube embedded order.

2 Expected Performance for a Radix 2 FFT

In the case of the radix-2 FFT, the data permutations involve a sequence of butterfly exchanges, where the span of the exchange doubles for each of the stages of the FFT.

The variable here is the width of the butterfly (d). Associated with this variable are three functions: G, C, and L. G is the cost (in normalized time units) to perform a communication over the Global Router. L is the function which gives the cost of performing a given communication using the Local communication mechanism. C is the total communications cost involved in performing an FFT.

By definition, we are assuming G is independent of d:

$$G(d) = 1$$

As such, the theoretical cost of communications for a radix 2 FFT of 2^m points, using a Global communication mechanism is:

$$C(m) = \sum_{s=0}^{m-1} G(2^s) = m$$

Let us assume that the Local communication mechanism is linearly dependent upon the width of the butterfly (d).

$$L(d) = kd$$

This suggests the cost using only a Local communication mechanism is:

$$C(m) = \sum_{s=0}^{m-1} L(2^s) = k(2^m - 1)$$

We can assume k is less than 1, since otherwise the Global mechanism would always out perform the Local mechanism. As such, the naive optimization is to use the Global Router when it is fastest, and use the Local mechanism otherwise.

In this case, the cost would be:

$$C(m) = \sum_{s=0}^{p} L(2^s) + \sum_{s=p+1}^{m-1} G(2^s) = (2 - k) + (m - p)$$

(For simplicity above, we assume $1/k = 2^p$.)

2.1 Rearranging to Optimize Communications

In the case of the FFT, however, there is another option. We know that as the computation proceeds, d = 1, 2, 4, 8 etc.... But we also know we can rearrange the data after some number of stages so that the ensuing stage has d = 1 again, followed again by 2, 4, etc.

This rearrangement method is know as the Four Step FFT [5]. The particular rearrangement necessary has been called the generalized transpose, generalized perfect shuffle, or generalized gather scatter, depending on how you characterize it. If the FFT has m stages (and m is even), and the rearrangement is performed after m/2 stages, then it is best viewed as a transpose. If the rearrangement is performed after only one stage, then it is best viewed as a perfect shuffle. Another way of viewing the rearrangement is as introducing a mixed radix to the FFT.

This introduces a new question: How much does it cost to perform this rearrangement of the data? For the sake of simplicity, we will first assume that the Global mechanism supports a rearrangement in 1 unit of time (that is, $G(rearrange) = 1$).

If we perform, essentially, a Four Step FFT, rearranging after m/2 stages (with the implicit assumption that p is less than m/2), then the optimal algorithm appears to have a cost of:

$$C(m) = 2\sum_{s=0}^{m/2} L(2^s) + G = 2k(2^{m/2+1} - 1) + 1$$

2.2 Multiple Rearrangements

Let's plug in some slightly different, but entirely plausible, numbers for G and L:

Let

$$G(d) = 4$$

and

$$L(d) = d$$

(Assume G is 4 for rearranging as well.)

In addition, let us suppose we wish to perform a 64-point FFT, with one datum on each of 64 PEs. If we proceed with the Local communications only, the cost will be:

$$L(1) + L(2) + L(4) + L(8) + L(16) + L(32) = 63.$$

The cost using a naive hybrid of Local and Global communications would be:

$$L(1) + L(2) + G(4) + G(8) + G(16) + G(32) = 19$$

The cost using one rearrangement (of cost = 4) after the third stage would be:

$$L(1) + L(2) + L(4) + G(rearrange) + L(1) + L(2) + L(4) = 18$$

But if we were to perform TWO rearrangements, one after the second, and one after the fourth stages, we would get cost:

$$L(1) + L(2) + G(rearrange) + L(1) + L(2) + G(rearrange) + L(1) + L(2) = 17$$

Of course, the rearrangements in this last method are different from the one in the method that had only one rearrangement in it. This diminishes all the more the certainty of our expectations about which method is best.

2.3 So, What DO We Expect?

We expect that using a rearrangement to merge Local and Global communications will provide a benefit to the communications cost of radix two FFTs. Critical to the utility of the rearrangement method will be the relative cost of Local and Global communications. The next section of this paper proceeds to identify (as best we can) the fastest ways to perform power of two communications on the MP-1, as well as how best to perform the needed rearrangements that seem to hold so much promise.

Given the results of those efforts, we provide comparisons of the different strategies shown above. We also show how these results prove useful in any algorithm that requires power of two communications.

3 Power of Two Communications on the MP-1

3.1 Introduction

This section will analyze the performance capabilities of the MasPar MP-1 parallel computer, in the realm of power of two communications. Since there are two fundamental mechanisms for performing these communications, we will examine each in turn.

3.2 The MP-1

The MasPar family of massively parallel systems uses a SIMD computational approach (single instruction, multiple data). High performance results from the replication of simple data PEs (from 1,024 to over 16,384 PEs in a scalable array) – each with its own dedicated data memory. The PEs do not fetch instructions from their own memories. Instead, an array control unit decodes instructions from the front end of the system. When a data intensive operation occurs, the array control unit broadcasts the instruction to the whole array of PEs, which execute the single instruction simultaneously, perhaps for multiple thousands of data points.

The MP-1 also provides for two interprocessor communications mechanisms. One is an eight way nearest neighbor toroidally wrapped grid, known as the X-Net. The other is the Global Router, which is implemented in silicon as a sequence of three crossbar connection channels.

3.3 Power of Two Communications Using the MP-1 Global Router

Simulations of the MP-1 Router chip provided us with the expectation that any 'constant offset' communication pattern would have optimal performance. By 'constant offset' communication we mean that for all PEs, each PE p_i could communicate with PE p_{i+k}, where k is independent of i. Power of two communications are a subset of this kind of communication. The simulations indicated that each such communication would require 135 microseconds to complete, for all PEs.

We performed actual timings, using a 4096 PE MP 1200, using calls to MPL, MasPar's C-like parallel language. The communication cost did indeed turn out to be independent of which power of two was being performed. That cost turned out to be 160 microseconds per communication. We expect that overhead costs of using MPL, as opposed to using MPAS (MasPar assembly language) accounted for the difference between expected and actual results.

3.3.1 Better Routing Through Microcode

By addressing some of the microscopic details of the MP-1 architecture, we have discovered how to substantially improve Global Router performance for power of two communications. We discuss these options here briefly, but we did not actually implement any of them, and they are not used in any ensuing analysis.

Each PE chip on the MP-1 has 32 PEs. Two wires leading from each chip pass into the Global Router. This associates 16 PEs with each Global Router wire. The way a message is transmitted using the Global Router is that each group of 16 PEs (each cluster) identifies a PE that will be transmitting or receiving data, and connects, through the Global Router, to the cluster containing the PE that will be transmitted to or received from. The transmission or reception is performed, and the connection is terminated. This process repeats a minimum of 16 times (one for each PE in a cluster). Because of the nature of power of two communications, it turns out that each PE in a given cluster wants to transmit or receive data from a PE in the same cluster. This means that we need only open the connection once, transmit or receive 16 times, and then close the connection.

The cost of opening and closing a connection is comparable to the cost of the actual data transmission. As such, we expect as much as a factor of two improvement in performance.

Another improvement can be garnered by transmitting AND receiving data once a connection is established, thus completing both halves of a butterfly exchange with the cost of half as many connection openings and closings.

3.4 Power of Two Communications Using the X-Net

Each PE is connected to its eight nearest neighbors on a two-dimensional rectilinear grid, with toroidal wrap. By numbering the PEs in 'raster scan order', we can see how power of two communications occur in a fashion more efficient than hypothesized by our original function L.

Consider a hypothetical, non existent machine with 64 PEs numbered as:

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

We can see immediately that for the power of two communication a distance of 8, the cost will be the same as for communicating a distance of 1. Indeed, if we assume the cost will go up linearly with distance on the two dimensional grid, then a 64-point FFT will require communications of costs: 1,2,4,1,2,4.

We could number the grid in 'quad tree' order (interleave the $n/2$ most significant bits with the $n/2$ least significant bits of the binary representation of the numbering for raster scan order, and you get quad tree order):

0	1	4	5	16	17	20	21
2	3	6	7	18	19	22	23
8	9	12	13	24	25	28	29
10	11	14	15	26	27	30	31
32	33	36	37	48	49	52	53
34	35	38	39	50	51	54	55
40	41	44	45	56	57	60	61
42	43	46	47	58	59	62	63

In our example, the communications for the FFT then go as 1,1,2,2,4,4 and take on a flavor very similar to that of our original function L, except that rather than

$$L(d) = kd$$

We have (more or less)

$$L(d) = k\sqrt{d}$$

This is justified theoretically, in that the number of data elements within a given distance, d, on a two dimensional grid is proportional to d^2, while on a one-dimensional grid the number of elements is proportional to d. This is also justified empirically, in that the sequence 1,1,2,2,4,4,... is bounded above by the square root of the sequence 1,2,4,8,16,32....

3.4.1 Using the Diagonals

There is a way to number the PEs so that power of two communications of the style used in FFTs is even more efficient than using either raster scan or quad tree ordering. The numbering (which we call 'cube embedded ordering') takes advantage of the special quality of the MP-1 that allows PEs to communicate with their four nearest diagonal neighbors as well as their four nearest neighbors. Here is this numbering for a 64 PE array:

```
54 1  5  10 14 25 29 50
0  4  3  7  24 28 27 31
45 2  6  17 21 26 30 41
43 47 16 20 19 23 40 44
46 57 61 18 22 33 37 42
56 60 59 63 32 36 35 39
53 58 62 9  13 34 38 49
51 55 8  12 11 15 48 52
```

This numbering is based on kernels of three-dimensional cubes embedded in the two-dimensional mesh. The first such kernel is the PEs numbered 0 through 7:

```
  1 5
0 4 3 7
  2 6
```

View each PE as a node in a three-dimensional cube, with PEs 0,1,2 and 3 on the front face, and 4,5,6,7 on the back face, like so:

```
   1
0   \  3
 \ 2 \  \
  \ \ 5  \
   4 \    7
      6
```

Then just stack the projections of the cubes in a consistent order, such as raster scan or quad tree. We chose quad tree, since it is better for the style of power of two communications that occur in sorting.

The communications cost for a sequence of power of two offsets then is: 1, 1, 1 (along each arc of each embedded cube), then 2, 2, 4, 4 ... .

3.5 Actual Results

The embedded cube method worked best, just as expected. The following table summarizes the relative costs. The first column is the power of two distance communicated, the second column is the measured time to perform the corresponding communication using quad tree order, and the third column corresponds to using the embedded cube ordering.

Distance	Using Quad	Using Cube
16384	180us	178us
8192	180	86
4096	80	84
2048	98	46
1024	58	52
512	48	24

Variances are conjectured to be due to MPL overhead variations.

Cube embedded ordering has also been compared with quad tree order in an implementation of the Batcher sorting algorithm on the MP-1 and found to be more than 12 percent faster [6].

We have accumulated enough data now to determine that both the Global Router and the X-Net are useful in power of two communications, since the Global Router requires 160us to complete any power of two communication and the X-Net may require as much as 180us to perform one, while for the vast majority of power of two communications potentially needed on an MP-1, the X-Net remains superior.

4 Performing the Data Rearrangement on the MP-1

So far, we have assumed that $G(rearrange) = G(d)$ which is a constant. In fact, we know this is unlikely. We also know that the rearrangement needed when using raster scan order, quad tree order, and cube embedded order will be different. In addition, we have not considered what $L(rearrange)$ might be.

4.1 Rearranging Using Local Communications

In some contexts, particularly when each PE holds more than one data item, the use of local communications mechanisms is appropriate to perform the class of data rearrangements we consider here [7]. If one wishes to perform a rearrangement of data, the minimum cost for that rearrangement using the X-Net (or, in general, any Local communication mechanism) is the maximum distance any of the data must travel between PEs.

We do not have the space to consider the rearrangements for each of raster scan, quad tree, and cube embedded order, but let us return to the general case of:

$$L(d) = kd$$

And the specific rearrangement that corresponds to where the data with index number i is exchanged with data in index number j, where i and j are represented by n bits, and the first $n/2$ bits of i are equal to the last $n/2$ bits of j, and vice versa. Thus, the rearrangement occurs after the n/2nd stage of the FFT. In this case, the maximum distance (in one dimension) any piece of data must move to complete the rearrangement corresponds to when the first $n/2$ bits of i are all one, and the rest are zero. So the cost of the rearrangement using the Local communication mechanism is:

$$L(rearrange) = \sum_{s=(n/2)}^{n-1} L(2^s) - \sum_{s=(0}^{(n/2)-1} L(2^s)$$

If we had not done the rearrangement, the cost of all of the remaining power of two communications after the n/2nd stage of the FFT would have been:

$$\sum_{s=(n/2)}^{n-1} L(2^s)$$

Having done the rearrangement, the cost of all of the remaining power of two communications after the n/2nd stage of the FFT would have been identical to that of the first n/2 stages:

$$\sum_{s=0}^{(n/2)-1} L(2^s)$$

Meaning that $L(rearrange)$ costs exactly what it saves! What $L(rearrange)$ is for arbitrary numberings of the PEs, and arbitrary rearrangements, relative to the cost savings, is beyond the scope of this paper. Suffice it to say here that for a substantial class of rearrangements, it is only worthwhile to use the Global Router.

4.2 Rearranging in Raster Scan Order

Let us return to our 8x8 array.

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

In computing the FFT, $L(1) = 1, L(2) = 2$, and $L(4) = 4$, but now we see that $L(8) = 1$ again, $L(16) = 2$, and $L(32) = 4$ (from communicating down the rows of the two-dimensional grid). It may appear that raster scan order has the data rearrangement implicitly included. Actually, raster scan order forces us to perform the rearrangement early, and twice. After the first two stages of the FFT, we can rearrange each row to be:

```
0  4  1  5  2  6  3  7
8  12 9  13 10 14 11 15
16 20 17 21 18 22 19 23
24 28 25 29 26 30 27 31
32 36 33 37 34 38 35 39
40 44 41 45 42 46 43 47
48 52 49 53 50 54 51 55
56 60 57 61 58 62 59 63
```

so that $L(4) = 1$. We can see that $L(8) = 1$ still, as are $L(16) = 2$, and $L(32) = 4$. So we want to also rearrange just before the last stage, so that the overall sequence of communications would be:

$$L(1) + L(2) + G(rearrange) + L(1) + L(8) + L(16) + G(rearrange) + L(8)$$

Since we need two rearrangements, and we are trying to amortize the costs over half the work for each, it seems unlikely that rearranging will be useful based on raster scan order enumeration.

4.3 Rearranging in Quad Tree Order

Quad tree order allows us to combine the two rearrangements needed in raster scan order into one. The rearrangement for quad tree order corresponds to what would have been a transpose if we had kept the data in raster scan order (that is, the data with index number i is exchanged with data in index number j, where i and j are represented by n bits, and the first $n/2$ bits of i are equal to the last $n/2$ bits of j, and vice versa.) This means, in our 8x8 example, the rearrangement of:

```
0  1  4  5  16 17 20 21        0  8  32 40 2  10 34 42
2  3  6  7  18 19 22 23        16 24 48 56 18 26 50 58
8  9  12 13 24 25 28 29        1  9  33 41 3  11 35 43
10 11 14 15 26 27 30 31   is   17 25 49 57 19 27 51 59
32 33 36 37 48 49 52 53        4  12 36 44 5  13 37 45
34 35 38 39 50 51 54 55        20 28 52 60 21 29 53 61
40 41 44 45 56 57 60 61        6  14 38 46 7  15 39 47
42 43 46 47 58 59 62 63        22 30 54 62 23 31 55 63
```

So that the sequence of communications for the corresponding FFT would be:

$$L(1) + L(2) + L(4) + G(rearrange) + L(1) + L(2) + L(4)$$

4.4 Rearranging in Cube Embedded Order

The rearrangement for cube embedded order is the same, in spirit, as for quad tree order. That is, once we have given each PE its new number, i, then the rearrangement corresponds to exchanging data with the PE numbered j, where j and i have the same relationship as above.

Thus, we can generate the permutation — not that we can say much about it intuitively. In our example, the rearrangement of:

```
54 1  5  10 14 25 29 50       54 8  40 17 49 11 43 22
0  4  3  7  24 28 27 31       0  32 24 56 3  35 27 59
45 2  6  17 21 26 30 41       45 16 48 10 42 19 51 13
43 47 16 20 19 23 40 44   is  29 61 2  34 26 58 5  37
46 57 61 18 22 33 37 42       53 15 47 18 50 12 44 21
56 60 59 63 32 36 35 39       7  39 31 63 4  36 28 60
53 58 62 9  13 34 38 49       46 23 55 9  41 20 52 14
51 55 8  12 11 15 48 52       30 62 1  33 25 57 6  38
```

The communications sequence will be identical for cube embedded order as for quad tree order, but $L(2) = 1$ instead of $L(2) = 2$.

4.5 Actual Results

We found that the time for performing both the quad tree and the cube embedded rearrangement were the same: 385us. From this, we can see that introducing the rearrangement provides the following benefits:

It takes 744us to perform each communication of a power of two less than or equal to 8,192 when we use quad tree addressing. It takes 76us to perform each such communication less than or equal to 128. Thus, if the rearrange cost is less than $744 - 2 * 76 = 592$us then the rearrange method provides an advantage. Since the rearrange cost is 385us, the improvement is

$$(744 - (385 + 2 * 76)) = 207us$$

which is a 27 percent improvement.

It takes 570us to perform each communication of a power of two less than or equal to 8,192 when we use cube embedded addressing. It takes 56us to perform each such communication less than or equal to 128. Thus, if the rearrange cost is less than $570 - 2 * 56 = 458$us then the rearrange method provides an advantage. Since the rearrange cost is 385us, the improvement is

$$(570 - (385 + 2 * 56)) = 73us$$

which is a 12.8 percent improvement.

Thus, the optimal time is 497us to complete all power of two communications for 32-bit data. In a complex, single precision FFT of 16,384 data points on 16,384 PEs, 64 bits would need to be transmitted in both directions, requiring 2 milliseconds to complete the communications.

5 Conclusions

Cube embedded enumeration, optimized with a Router based rearrange, is the best mechanism to perform sequences of communications that involve power of two offsets in the index of the data. Compared with a naive Global Router based implementation, the improvement available is a factor of 4.5. We expect that microcode based routing optimizations could improve the rearrange time by as much as a factor of four. This could make interprocessor communication time for FFTs, sorting, and other applications insignificant relative to the computation time.

Most important, we have shown how the combination of a general Global Router and a high speed Local communication mechanism can perform tasks substantially more efficiently than either could alone.

References

[1] L.R. Rabner and B. Gold, *Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

[2] H.S. Stone, 'Parallel processing with the perfect shuffle' *IEEE Transactions on Computing*, C-20, 2 (February, 1971).

[3] B.L. Buzbee, G.H. Golub, and C.W. Nielson, *SIAM Journal of Numerical Analysis*, vol. 7, pp. 627-656, 1970.

[4] R. C. Green, *Digital Signal Processors*, January 9, 1990.

[5] D.H. Bailey, 'FFTs in External or Hierarchical Memory', *Proceedings of Supercomputing*, 1989.

[6] P. A. Lawrence, *Personal Communication.*

[7] C. L. Kuszmaul, 'Rapid Transpose Algorithms on Massively Parallel Computers', *SIAM 1990.*