Programming task:
Reduce and AllReduce for non-powers-of-two

Nejmeddine Douma

May 31, 2017

1 Goal of the task

Implement AllReduce() and Reduce() in Thrill's network layer for p workers with $p \neq 2^k$ and add enough test cases to check the code's correctness.

Thrill requires AllReduce and Reduce collective operations. For powers of two the hypercube algorithm is used. However, for non-powers-of-two, only a trivial $2 \log_2 p$ reduction algorithm was used. For performance, using a $\log_2 p + O(1)$ algorithm is highly wanted.

2 Algorithm

2.1 The 3-2 Elimination Step

Given a group of three processes p_0, p_1 and p_2, a 3-2-elimination step is used to share the p_2's vector with p_0 and p_1 so we can eliminate p_2 from the next rounds of the computation and keep only p_0 and p_1. The vector value in p_2 is updated at the end of the computations.

2.2 The protocol

R. Rabenseifner and J.L. Träff [1] presents an algorithm that requires $\lceil \log_2 p \rceil + 1$ rounds for small vectors and twice the number of rounds for large vectors. The algorithm main idea is to use 3-2-elimination steps to reduce the number of the involved hosts to its next smaller power-of-two and then apply the reduction algorithm for powers-of-two.

3 Implementation

A recursive implementation of the algorithm was used. At each round i, hosts are separated into groups of size 2^i with i starting from 0. If the group count is even, butterfly style exchanges are executed between the group pairs. If the group count is odd, a 3-2-elimination is applied on the last 3 groups and butterfly style exchanges are executed between the remaining group pairs. The recursion ends when all the non-eliminated hosts are part of a single group. At this point the computation results are shared with all the previously eliminated hosts, this last operation is done in only one extra round.

Algorithm 1: AllReduce for Non-Powers-Of-Two

Result: All hosts store the same reduced value

`reduceRound(`$host, groupSize, remainingHostsCount$`)`

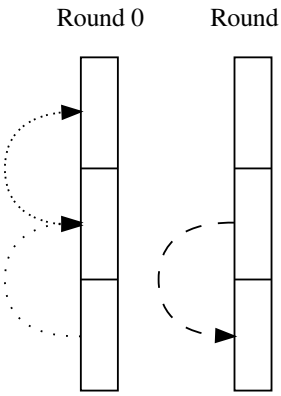
```
1  groupsCount = remainingHostsCount/groupSize;
2  if groupsCount is even then
3  |  // exchange and reduce values with neighbor at distance groupsSize;
4  else
5  |  if host is part of the last 3 groups then
6  |  |  // host participates in a 3-2 elimination
7  |  else
8  |  |  // no elimination, exchange and reduce values with neighbor at
   |  |    distance groupsSize
9  |  endif
10 |  remainingHostsCount- = groupsSize;
11 endif
12 groupsSize <<= 1;
13 // Recursion
14 if groupsSize < remainingHostsCount then
15 |  reduceRound(host, groupSize, remainingHostsCount)
16 else if  host has an eliminated neighbor then
17 |  //send value to eliminated neighbor
18 endif
```

4 Results

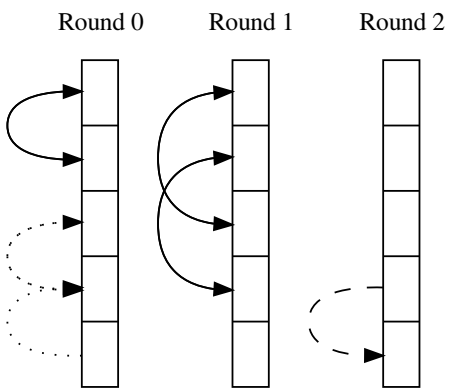
4.1 Communication graphs

Graphs illustrating the communications between various numbers of hosts are available in the appendix 1

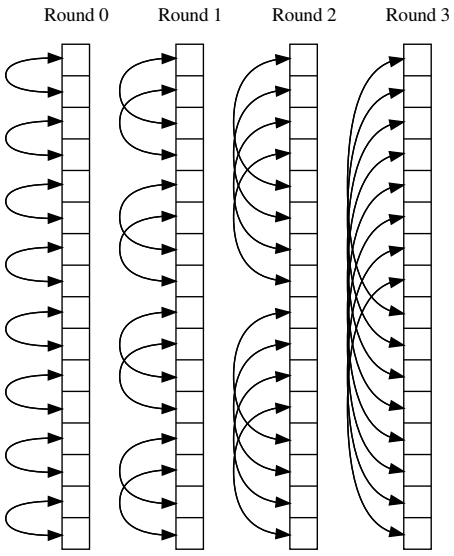
Appendix



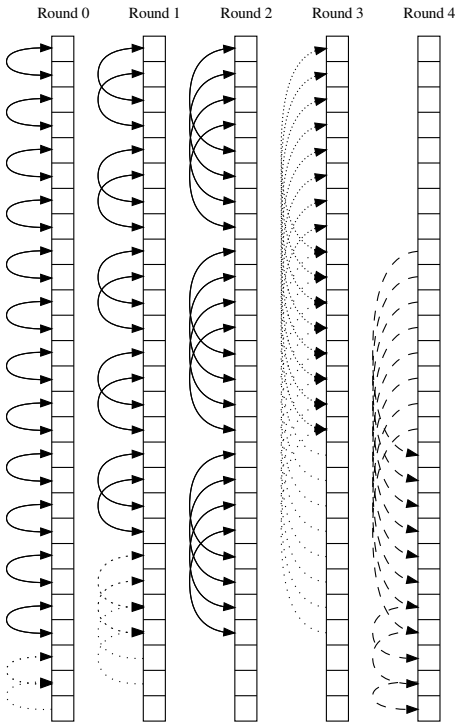
(a) 3 hosts



(b) 5 hosts



(c) 16 hosts



(d) 27 hosts

Figure 1: Communications between various numbers of hosts
The exchanges that are part of a 3-2-elimination are represented with dotted edges and those used for gathering the results are represented with dashed edges.

References

[1] Rolf Rabenseifner and Jesper Larsson Träff. More efficient reduction algorithms for non-power-of-two number of processors in message-passing parallel systems. In *European Parallel Virtual Machine/Message Passing Interface Users Group Meeting*, pages 36–46. Springer, 2004.