

CSCI596 Assignment 2—Message Passing Interface

Due: September 16 (Wed), 2020, 11:59 pm

Goal: Implement Your Own Global Summation with Message Passing Interface

In this assignment, you will write your own global summation program (equivalent to `MPI_Allreduce`) using `MPI_Send` and `MPI_Recv`. Your program should run with $P = 2^l$ processes (or MPI ranks), where $l = 0, 1, \dots$. Each process contributes a partial value, and at the end, all the processes will have the globally-summed value of these partial contributions.

Your program will use a communication structure called butterfly, which is structured as a series of pairwise exchanges (see the figure below where messages are denoted by arrows). This structure allows a global reduction among P processes to be performed in $\log_2 P$ steps.

$$\begin{aligned} & a_{000} + a_{001} + a_{010} + a_{011} + a_{100} + a_{101} + a_{110} + a_{111} \\ = & ((a_{000} + a_{001}) + (a_{010} + a_{011})) + ((a_{100} + a_{101}) + (a_{110} + a_{111})) \end{aligned}$$

At each level l , a process exchanges messages with a partner whose rank differs only at the l -th bit position in the binary representation (Fig. 1).

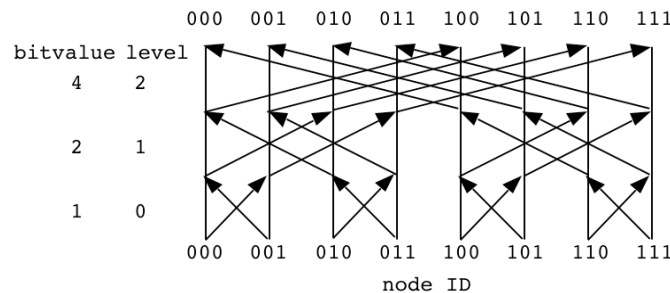


Fig. 1: Butterfly network used in hypercube algorithms.

HYPERCUBE TEMPLATE

We can use the following template to perform a global reduction using any associative operator OP (such as multiplication or maximum), $(a \ OP \ b) \ OP \ c = a \ OP \ (b \ OP \ c)$.^{1,2,3}

```
procedure hypercube(myid, input, logP, output)
begin
  mydone := input;
  for l := 0 to logP-1 do
    begin
      partner := myid XOR 2l;
      send mydone to partner;
      receive hisdone from partner;
      mydone = mydone OP hisdone
    end
  output := mydone
end
```

USE OF BITWISE LOGICAL XOR

Note that

$$\begin{aligned} 0 \text{ XOR } 0 &= 0 \text{ XOR } 1 = 1; \\ 0 \text{ XOR } 1 &= 1 \text{ XOR } 0 = 1. \end{aligned}$$

so that $a \text{ XOR } 1$ flips the bit a , i.e.,

```
a XOR 1 =  $\bar{a}$ 
a XOR 0 = a
```

where \bar{a} is the complement of a ($\bar{a} = 1 - a$ for $a = 0$ or 1). In particular, $\text{myid} \text{ XOR } 2^l$ reverses the l -th bit of the rank of this process, myid :

$abcdefg \text{ XOR } 0000100 = abcd \bar{e}fg$

Note that the XOR operator is \wedge (caret symbol) in the C programming language.

ASSIGNMENT

Complete the following program by implementing the function, `global_sum`, using `MPI_Send` and `MPI_Recv` functions and the hypercube template shown above.

Submit the source code as well as the printout from a test run on 4 processors and that on 8 processors.

```
#include "mpi.h"
#include <stdio.h>

int nprocs; /* Number of processes */
int myid;   /* My rank */

double global_sum(double partial) {
    /* Implement your own global summation here */
}

int main(int argc, char *argv[]) {
    double partial, sum, avg;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
    MPI_Comm_size(MPI_COMM_WORLD, &nprocs);

    partial = (double) myid;
    printf("Node %d has %le\n", myid, partial);

    sum = global_sum(partial);

    if (myid == 0) {
        avg = sum/nprocs;
        printf("Global average = %le\n", avg);
    }

    MPI_Finalize();
    return 0;
}
```

References

1. Slides 20-24 in <http://cacs.usc.edu/education/cs596/MPI-VG.pdf>.
2. [https://en.wikipedia.org/wiki/Hypercube_\(communication_pattern\)](https://en.wikipedia.org/wiki/Hypercube_(communication_pattern)).
3. I. Foster, *Designing and Building Parallel Programs* (Addison-Wesley, 1995) Chap. 11—Hypercube algorithms <https://www.mcs.anl.gov/~itf/dbpp/text/node123.html>.