

Joshua Nasman

May 28 Update

	Actual	Original Simulation	Newer simulation
Allreduce	0.028838	0.048222	0.039102
Reduce	0.000124	0.000015	0.000012
Barrier	0.003801	0.000182	N/A
AllGather	0.000011	0	0
AllGatherv	0.000051	0	0
Isend	0.001504	0.002723	0.000432
Ireceive	0.000239	0	0
Waitall	0.000815	0	0
Finalize	0.000001	0	0
Init	0.000007	0	0

These results were obtained using a modified version of the second program. The modified version used the wrapper on the simulated machine and immediately multiplied the time of any call times the timing for that call. Although not ultimately useful since simulating a machine on itself would be completely useless, it should collect the same information as the other. The guess on where the mistake lies that produces the discrepancy in these results is that the lookup for a call in the adjustedmypi.c file is not correctly finding the correct time. This could explain how the allreduce call is actually taking longer in the simulated version if it is not getting the right size for the operation. It is also noticed that in the intermediate file the largest allreduce is recorded last. This is the one the program would default to if it could not find the right number.

Currently the allreduce calls and reduce calls were made using the (probably mistaken) assumption that doing an allreduce over the same amount of memory (8 doubles being the same as a  $8 * \text{sizeof}(\text{double})$  operation). This does not explain the peculiarities of Isend.

Plan for collective calls which are not long enough:

Problem: Collective calls are forced to wait until all processes arrive at the same point. The emulator currently does not account for this. The time waited here is not directly dependent upon the network but instead dependent upon where things are waiting.

Solution: In order to account for this time in the adjustedmypi.c the time waited (multiplied by the appropriate factor for computation time) must be added to the looked up value and then the time it takes to do the Barrier on the system being used must be subtracted. That is:

$$\textit{Time waited in barrier} * \textit{CPU factor} + \textit{Looked up time} - \textit{time to call barrier}$$

Problem: waitall not implemented yet.

Solution: Still thinking

Things to do:

1. In adjustedmypi.c: throw an error if the appropriate size is not found in the file
2. Adjust reduce and all reduce calls for sizes better if necessary.
3. Discuss lsend issues
4. implement idea for collective calls in last paragraph

Questions:

1. Are there reasons lsend would take longer? It is non-blocking so there wouldn't seem to be.
2. How are allReduce times affected by the MPI data type. Is using ints, doubles and multiple doubles sufficient?
  - a. If they are affected how do we generate same data type?
  - b. Meet with Ken to discuss where data types are generated