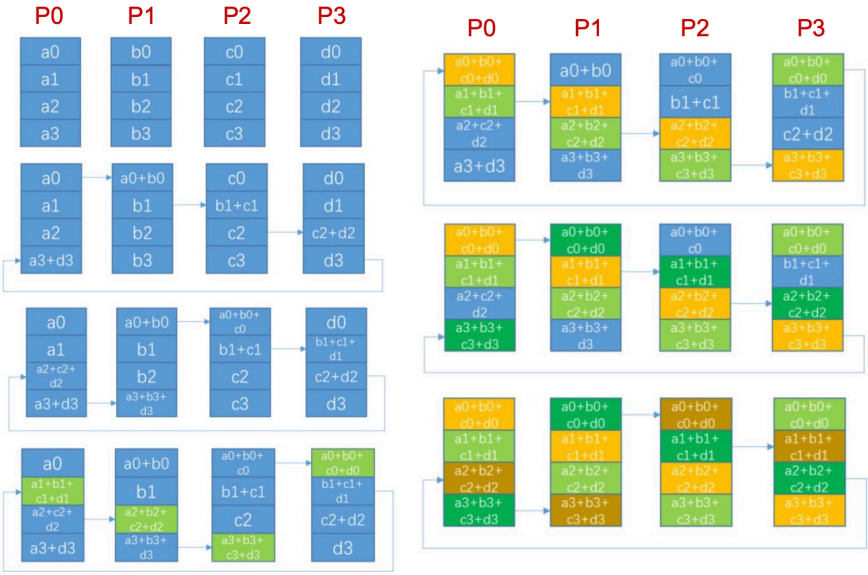
MPI Allreduce 实验报告

容逸朗　2020010869

Ring Allreduce 实现

- 首先，我们要处理进程数不整除数据量的情况：
  - 一种简单的方法是，除最后一个数据块以外，其余的长度统一，均为 $len = \lfloor \frac{n}{comm\_sz} \rfloor$；
  - 此时最后一个数据块的长度是 $n - (comm\_sz - 1) \times \frac{n}{comm\_sz}$ ；
  - 这样做的好处在于，我们可以通过块位置乘以 len 的方式直接得到数据块的偏移量；
- 接下来实现 Ring Allreduce 算法：
  - 算法分为两部分，第一部分是 Reduce-scatter，主要是通过分块方法计算结果；
  - 第二部分是 Allgather，即把每一进程的结果复制到其他进程的对应位置上；



  - 左图为第一阶段的操作过程，右图则为第二阶段；
- 对于两部分而言，操作的过程是十分相似的：
  - 每次传输时，首先根据当前轮数、进程号找到发送和接收数据的正确位置；
  - 然后利用 MPI_Irecv 和 MPI_Isend 收发数据；

    注意，第一阶段应当发出 sendbuf 的数据，而第二阶段则是 recvbuf 的数据。
  - 再使用 MPI_Waitall 等待数据收发完成；
  - 若处于第一阶段，则数据就位后，还需要把接收到的数据 (recvbuf) 与自身的数据 (sendbuf) 相加，然后把结果分别复制到 recvbuf 和 sendbuf 之中；
- 上面的情况并没有考虑到进程数为 1 的情况，故当进程数为 1 时，应该把 sendbuf 的数据直接复制到 recvbuf 中；
- 经过上面的流程后，结果就在 recvbuf 的正确位置上了！

通信时间测试

节点数	进程数	通信量	MPI 用时/ms	Naive 用时/ms	Ring 用时/ms
		10^4	0.032437	0.036498	0.028017
1	1	10^6	3.61272	3.5662	3.49388
		10^8	928.606	930.581	463.589
		10^4	0.957494	1.88975	0.614788
1	7	10^6	41.2156	58.4814	32.4495
		10^8	4080.69	6571.67	3365.52
		10^4	0.817381	8.50503	0.677773
1	14	10^6	52.2498	68.4995	42.3224
		10^8	5775.77	7193.14	5346.5
		10^4	1.17186	3.84474	1.29839
1	28	10^6	148.692	132.214	101.063
		10^8	11723.7	12463.1	12020.4
		10^4	0.317486	0.446723	0.214491
2	2	10^6	13.6354	20.6378	9.35509
		10^8	1830.09	2633.28	1047.68
		10^4	1.15006	13.807	0.916153
2	14	10^6	49.8921	69.7773	41.1224
		10^8	4776.41	7054.59	4591.8
		10^4	1.93288	2.41345	1.65981
2	28	10^6	65.624	68.7127	53.5499
		10^8	5900.24	8259.14	5770.66
		10^4	1.80787	4.21147	2.10809
2	56	10^6	145.027	155.703	106.355
		10^8	11868.5	12803.3	12187.2