MPI 快速排序

教师：何克晶
助教：游灵聪

快速排序简单原理

获取断开的位置的函数

```
int partition (int a[],int l,int r);
```

快速排序主函数

```
Void quicksort(int a[],int l,int r)
{
        int p = partition(a,l,r);
        quicksort(a,l,p-1);
        quicksort(a,p+1,r);
}
```

使用MPI的基本原理

- 通过使用多个进程，每个进程处理一小段需要排序的数据，最终将每个进程的数据组合在一起，所有主要有三个步骤：

a) 划分：划分数据到不同的进程

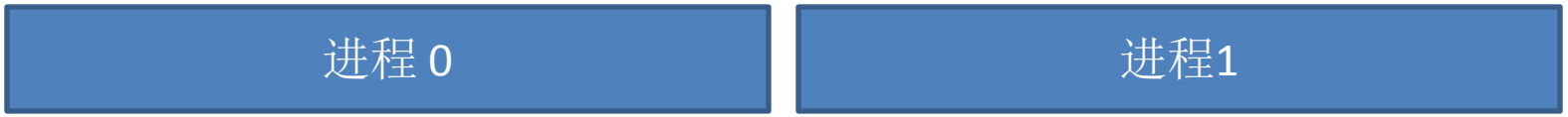
b) 处理：每个进程处理自身的数据（主要是递归调用MPI的快速排序函数）

c) 合并：将处理好的数据传回来

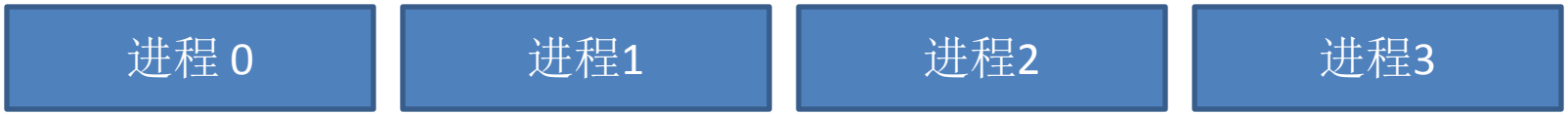
如何划分数据

- 原始数据：

- 2个进程进行划分后：



- 4个进程进行划分后：



- 2^m个进程进行划分后：

处理器的个数

- mpirun 指令中能通过-np参数对处理器的个数进行设置
- 但是在使用MPI写快速排序算法的时候并不是设置多少个处理器，那么多处理器就会被用到，因为按照上述办法，2^m个进程就足够了，所以就算设置了np的值为15，实际上的进程数还是只有8个，其它的7个是不会被用到的

MPI编写程序

- 主要通过多个进程实现MPI的并行策略
- 进行恰当的划分任务是实现MPI程序的主要内容
- 根据不同的进程ID，处理不同的内容，因为不是每个进程都被划分的同样的任务
- 在进程间传递数据是MPI的主要步骤

使用MPI编写快速排序

- 划分任务

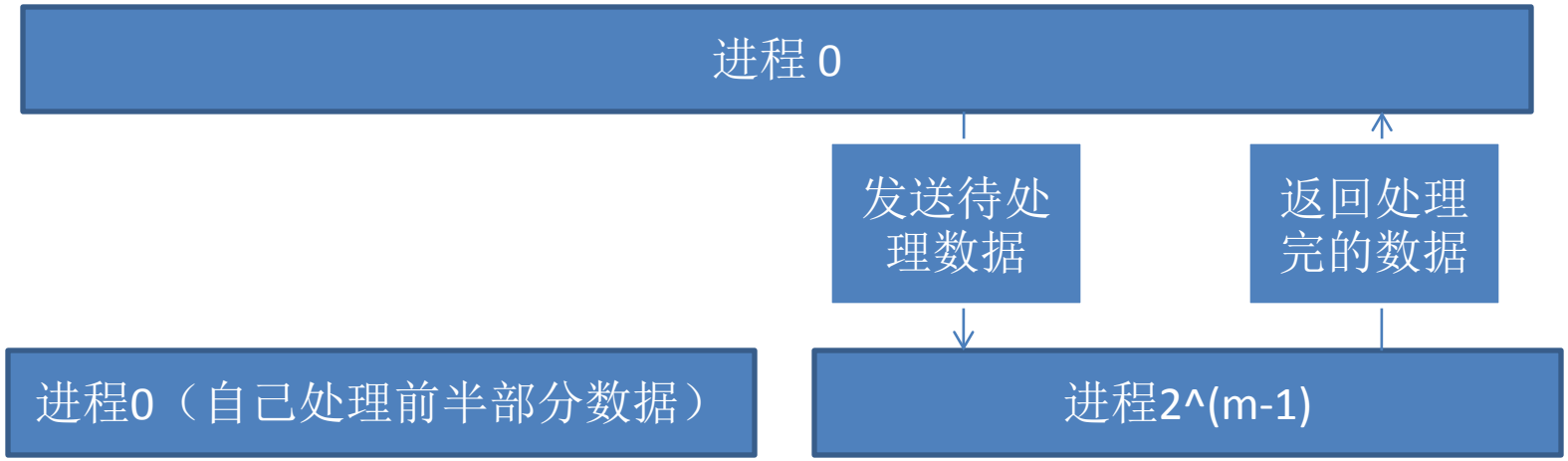
  分成两种进程：

  一种为划分进程，处理划分任务和处理划分后数据的前一部分，最后接受处理完的后一部分数据；

  另一种为接受进程，接收并处理数据划分后的后一部分，最终返回处理完的数据

使用MPI编写快速排序

- 划分任务演示：



进程0：划分和处理划分的前一部分，发送后一部分数据

进程2^(m-1)：接收并处理划分的后一部分数据，最终返回处理完的数据

MPI编写快速排序

- 按照划分的任务使用使用两种不同的进程，在不同的进程中按照划分的任务做相应的处理，就可以完成快速排序了。

实验报告

- 包括实验内容及简要分析、代码、运行结果截图、简要总结。。
- 考试前发到我的邮箱：ylcaa@foxmail.com
- 最终评分按照平时签到和实验报告给分。