并行与分布式计算

姓名：阚双祥
苏州大学计算机科学与技术学院
Email: 20185227018@stu.suda.edu.cn

摘要：本报告主要叙述了利用python语言来模拟并行和分布式计算。将一个大的计算任务划分成几个小任务，然后通过一个管理节点向其他计算节点发送这几个不同的小的计算任务，由各个计算节点得到计算结果，将各个计算结果发送给其中的某个计算节点，由该计算节点汇总，最后将结果发送回控制节点。

关键词：并行算法，分布式，python，多线程，socket

1 实验内容

计算内容：给定一个自然数的范围$2 \sim n$，计算该范围内共有多少个素数和整个计算过程所消耗的时间

输入：需要计算素数个数的范围最大值n

输出：该范围的素数总个数s和总的消耗时间t

2 重点难点分析

- 管理节点将计算代码发送给各个计算节点之后，计算代码对于每一个计算节点来说都是一样的，那么每个计算节点如何知道自己的计算任务
- 每个计算节点完成计算任务后，如何将计算的局部结果发给其中的指定的计算节点
- 指定的计算节点如何收集其他计算节点发来的结果
- 指定的计算节点如何将最后的汇总结果发送给管理节点

3 计算的技术路线

该实验主要利用python语言实现，利用python语言中的socket模块以及多线程和消息队列来模拟管理节点和计算节点。其中管理节点作为client，计算节点作为server。该实验中一共有4 个文件，分别是

- 作为管理节点的20185227018control.py文件
- 作为计算节点的20185227018node.py文件

- 需要执行的代码文件20185227018.py，其中有两个参数，分别是计算节点编号以及计算节点总数
- 存有所有计算节点IP地址的hosts.txt文件

管理节点将根据hosts.txt的IP地址向各个计算节点发出连接请求，连接成功后，将需要执行的代码文件20185227018.py、计算节点编号、计算节点总数以及收集其他计算节点局部结果的特定节点的IP 地址（这里假设该节点的编号为**1**）发送给对应IP 地址的计算节点。由于代码文件对于每一个计算节点来说都是一样的，所以每个计算节点的计算任务将根据自身的编号和计算节点总数来确定自身的工作量。对于本实验计算素数个数来说，采用均匀的分片方式，即根据需要计算的自然数的总个数除以总的计算节点数，将其按顺序均匀分成相应的范围，然后由各个计算节点分别计算自己的相应范围内的数（这样做的一个问题是会导致负载不均衡，即越到后面的数字越大，其相应的计算时间也就越长）。

管理节点在运行过程中采用多线程的方式，即根据hosts.txt中的IP地址个数，利用循环分别开启相应的线程，利用相应线程去传递相关数据。在对所有计算节点传递完数据后，将会开启一个接收最终计算结果的线程，等待计算节点编号为**1**的节点（该计算节点收集其他计算节点的局部结果并汇总）将最终结果传递给管理节点，由管理节点输出最后的计算结果，即总的素数的个数，并输出总的运行时间。

由于每个计算节点的代码都是相同的，而节点编号为**1**的计算节点的任务和节点编号不为**1**的计算节点的任务是不同的，因此需要根据不同的节点编号来确定不同的任务。计算节点在运行时，首先开启一个线程来接收控制节点的发来的数据，并将自己的编号和计算节点总数作为参数传递给代码文件20185227018.py 执行，得到计算结果。然后根据自己的编号为决定自己接下来的任务。

- 节点编号为**1**，除了开启一个线程来接收控制节点发来的数据外，同时不停的监听其他计算节点是否将局部计算结果发过来，如果监听到有传过来的其他的局部计算结果，就开启一个线程去接收该局部计算结果，并将其放到消息队列中。同时在自己的计算任务完成后，也将计算结果放到消息队列中。最后将消息队列中的消息的总个数与所有计算节点总数相比较，如果相等。则将消息队列中的结果全部取出相加，得到总的素数个数。并将其发送给控制节点。然后结束相应的线程，等待下一次的计算任务。
- 节点编号不为**1**，完成相应的计算后，根据控制节点传递过来的节点编号为**1**的IP地址，开启一个client socket，去连接节点编号为**1**的计算节点，将计算结果传递过去。然后线程结束，等待下一次的计算任务。

控制节点接收到编号为**1**的计算节点传送过来的最终计算结果后，输出计算结果。接收线程结束。同时显示总的计算时间，控制节点程序结束。

假设有两个计算节点，节点编号分别为**1**和**2**。其中计算节点**1**收集**2**的结果，并将结果发送给控制节点。根据以上的计算流程，可以得到如下的信息交互表Table 3-1以及状态转换图：

4 实验结果及分析

本实验中，我们将自然数的范围设置在2 $\sim$ 1000000, 总的计算节点个数为3 个，以节点编号

步骤	管理节点client	计算节点**1**server	计算节点**2**server
1	控制节点连接计算节点		
2	发送：”control”		
3		发送：”receive ready”	发送：”receive ready”
4	发送节点编号、节点总数、节点1的IP		
5		发送消息：”dict success”	发送：”dict success”
6	逐行发送代码文件中的代码，以”$”结束		
7	以”###”表示代码发送结束		
8		计算结果	计算结果
9			发送”node”给节点**1**
10		发送”receive ready”给节点**2**	
11			发送计算结果
12		发送”ok”给节点**2**	
13		将结果放到消息队列中	
14		汇总结果	
15		发送最终结果给管理节点	
16	收到结果，发送”ok”给节点**1**		

Table 3-1 实验结果

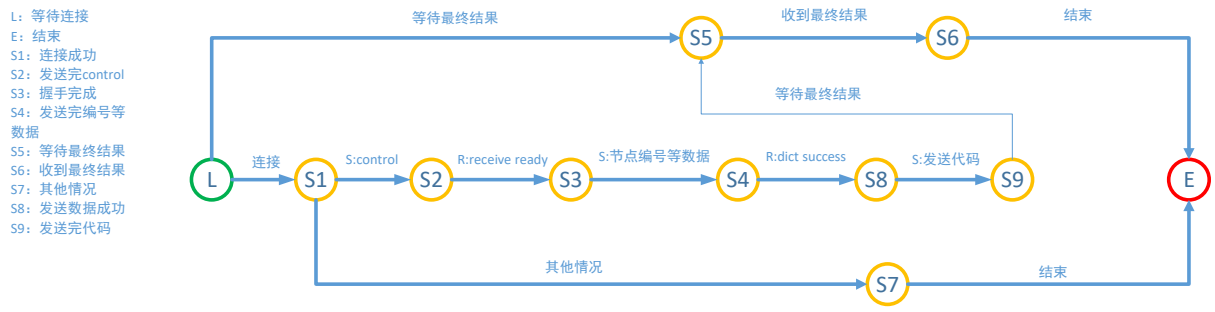

Figure 3-1 控制节点

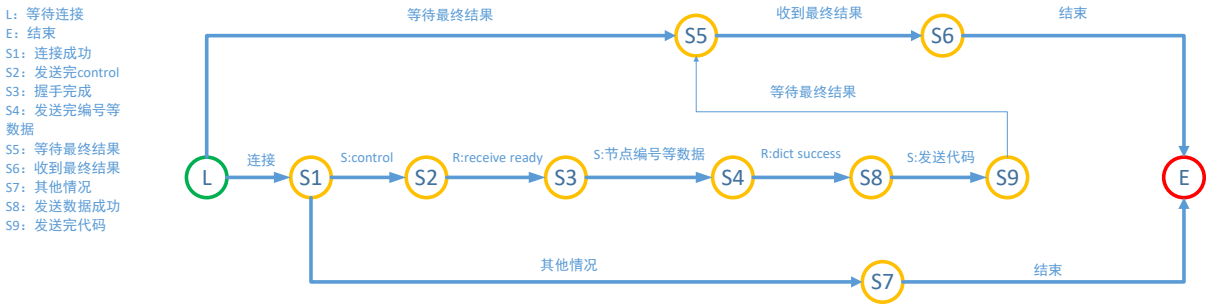

Figure 3-2 控制节点

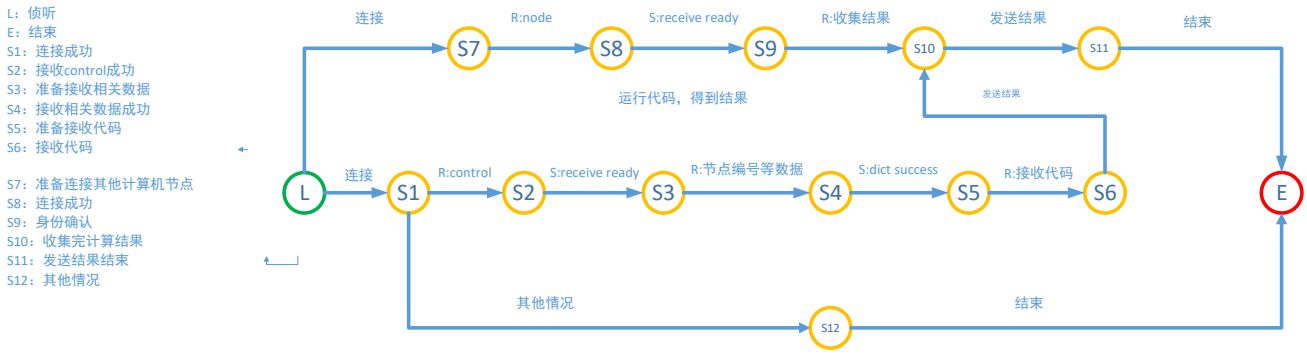

Figure 3-3 计算节点**1**

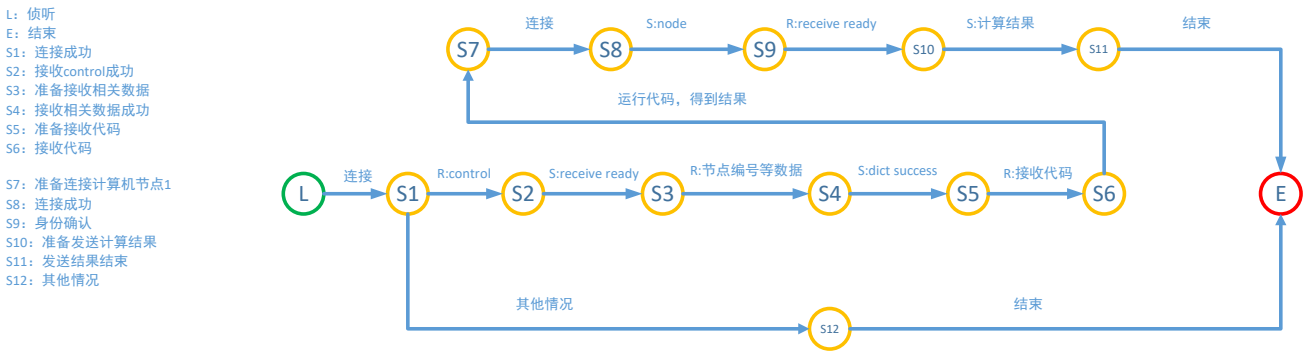

Figure 3-4 计算节点**2**

自然数范围	计算节点个数	结果	运行时间（保留5位小数）
$2 \sim 1000000$	1	78498	5.93713
$2 \sim 1000000$	2	78498	4.02452
$2 \sim 1000000$	3	78498	2.94231

Table 4-2 实验结果

为**1**的计算节点为收集节点。并分别进行三次实验，这三次实验分别以1个计算节点，2个计算节点，3个计算节点分别计算$2 \sim 1000000$ 的素数个数，比较这三次的结果并进行分析。

从Table 4-2中可以看出，随着计算节点个数的增加，完成整个计算任务所需要的时间也就越少，但是并没有按照3个计算节点、2个计算节点和1个计算节点相应的节点个数减少。原因就在于每个计算节点的计算量并不相同，节点编号越大的计算节点，虽然需要计算的整数的个数与前面的节点一样，但是其中的每个整数都大于编号小的节点中的整数，因此每个整数所需要的计算时间都要更长。所以节点编号最大的那个计算节点完成计算的时间就成为多个计算节点同时计算的所需总时间的关键，这也是需要优化的地方。