**华南农业大学数学与信息学院**

综合性实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| **实验题目** | 生产者-消费者模型实现 | **起止日期** | **2019学年第一学期** |
| **课程名称** | 分布与并行计算 | | |
| **学生资料** | 学号 201725110105 姓名 邓泽骏 | | |
| **实验内容** | 通过消息队列完成分布式进程相互通信。其中 生产者，消费者 完成任务为“判断素数”。生产者生产一个随机数字，消费者来判断，数据传输通过消息队列来进行。 | | |
| **评**  **语**  **与**  **成**  **绩** | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **项目/分数** | **优** | **良** | **中** | **及格** | **不及格** | | 系统分析设计 | □ | □ | □ | □ | □ | | 程序设计水平 | □ | □ | □ | □ | □ | | 内容完成情况 | □ | □ | □ | □ | □ | | 设计报告撰写质量 | □ | □ | □ | □ | □ | | 课程设计总结情况 | □ | □ | □ | □ | □ |   附加说明：  成绩 (优/良/及格/不及格) 指导教师 林毅申  2019 年 12 月 | | |

# 问题描述

解决问题：“判断素数”

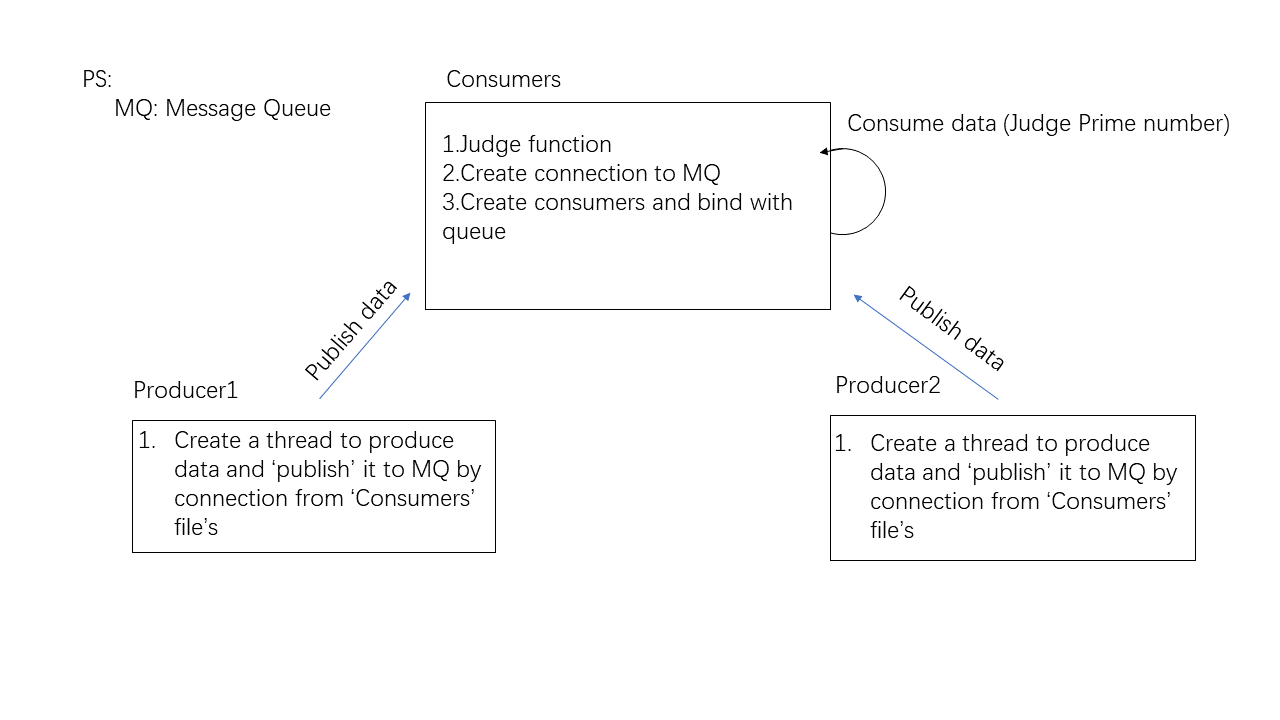
数据规模：每个生产者生产 (10^5) 个数据

单个数据规模：基准 (2\*10^10) 偏差 (10^3) ，数据得出 = 基准 + rand(偏差) 即在偏差的范围内加上基准，为所得单个数据

消费者判断方法：通过循环 i = 2-sqrt(n) 的数据范围，若 n % i = 0 则为合数，全部循环完之后依旧不能整除，则为素数。

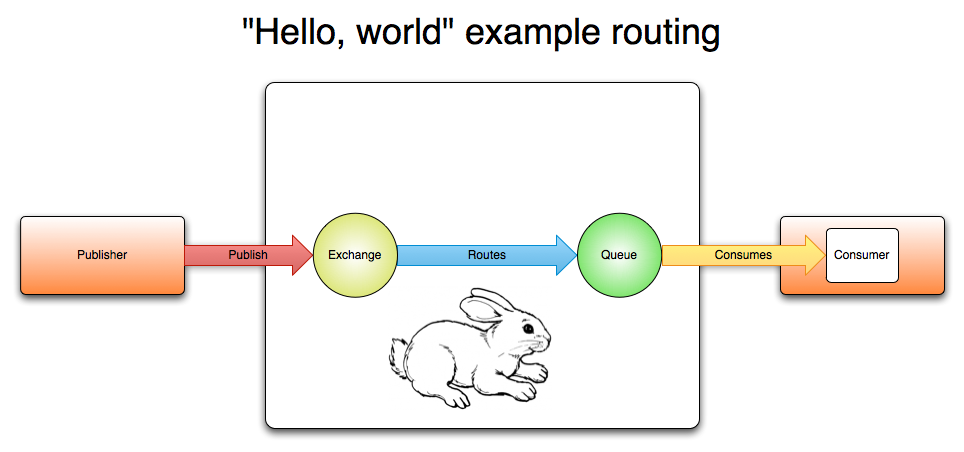
通过程序编码完成此问题求解

# 2、系统架构



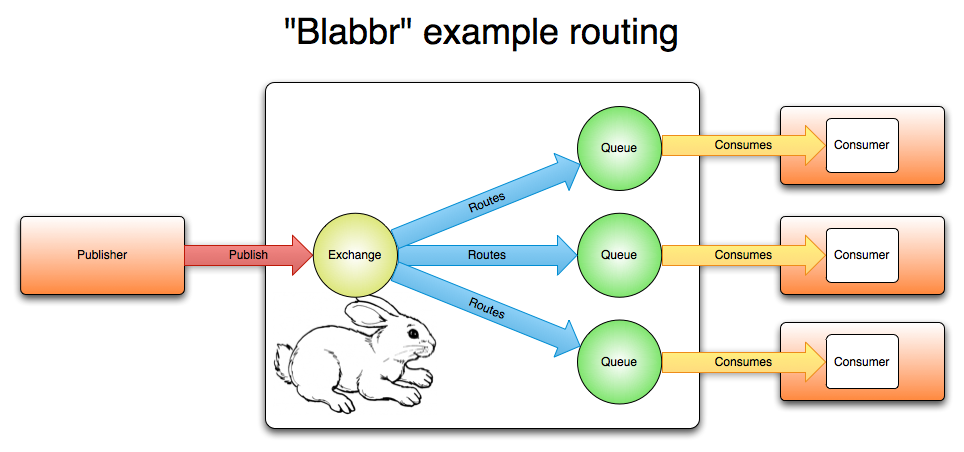
1-1

(From [2] instant photo, ps: ‘bunny’ has the meaning of young rabbit)



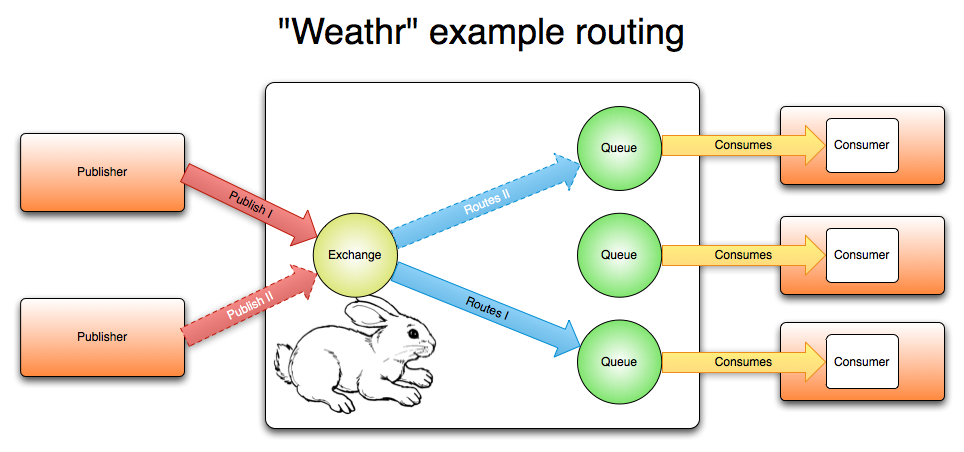
1-M (Multiple)

(From [2] instant photo)



M-M (Multiple)

(From [2] instant photo)



# 3、关键技术实现

开发环境：Windows 10 , (Ubuntu bash with wsl)

开发工具：visual studio code

开发语言：ruby

运行环境：Ubuntu bash on Windows 10 wsl ,

环境要求：支持ruby ，RabbitMQ , bunny

**代码演示均为 2P-4C 实验代码**

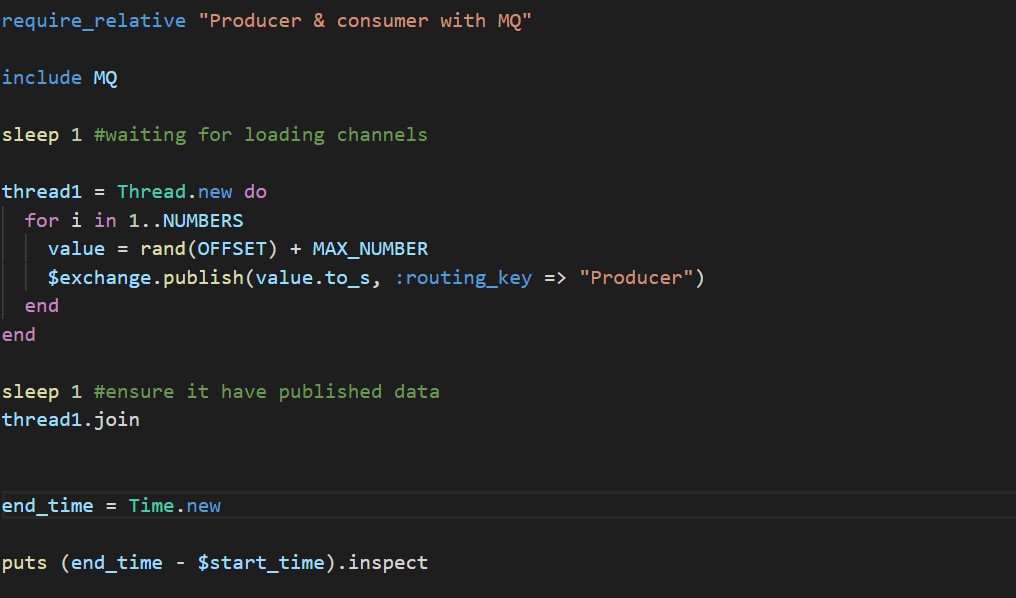
**由于 ruby 动态语言相对于静态语言的功能实现较为简单，某些功能可能C++需要一个函数完成，而ruby只需一条语句，所以实现部分代码展示仅仅展示稍稍有难度的部分。**

生产者：

1.随机数产生

2.发送数据

3.计算时间

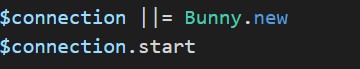


消费者：

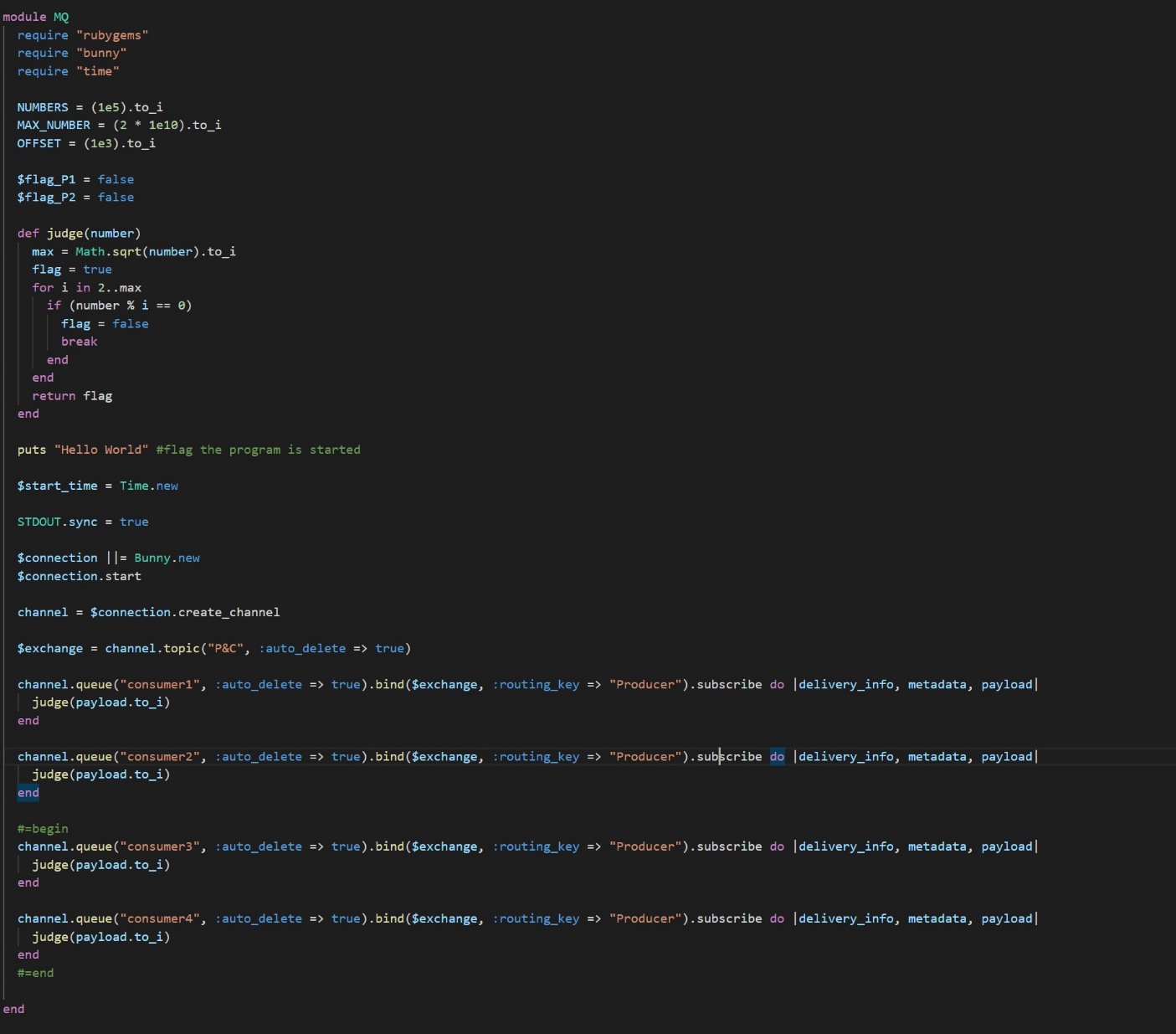
1.判断素数

2.开始计时

3.创建与消息队列服务器的TCP连接



4.创建信道



消息队列：

1.连接消息队列服务器

2.在连接之上创建信道

3.在信道中创建队列作为消费者

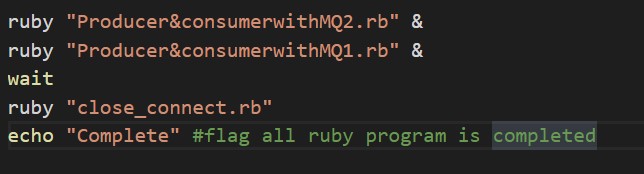
Tips 消息之间的传递只能是string类型，需要 封装和解包

代码见消费者代码

运行：

由于 ruby MRI 中的 GIL ，导致无法通过真正的多线程来完成多个生产者。于是通过同时启动两个ruby文件生成两个进程，对消费者进行数据共享来实现。（最终时间按照最慢的计算）

实现：通过 shell 脚本实现：



# 实验结果及讨论

表格如下

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 类型 | 1-1 | 1-2 | 2-2 | 2-4 |
| 时间消耗(s) | 19.8311305 | 16.6405008 | 22.5640746 | 15.9927747 |

对比 1-1 与 1-2 可以发现 增加消费者数量能够明显地提升整个软件的整体运算速度

对比 1-2 与 2-2 可以发现 增加生产者后，速度明显降低

对比 2-2 与 2-4 可以发现 消费者数目的增多，能够显著的提升整个软件的性能

由上述实验数据可以得出，消费者的增加能够显著地提升整个软件的性能；生产者的增加对整体系统的负担增大，相同数量的消费者将需要更多时间来处理数据

# 本课程收获

本次实验收获良多，第一次实际应用消息队列解决这个问题。对于RabbitMQ ，bunny 的使用有了初步的了解，与此同时，加深了我对于ruby语言的分布式开发应用，在查找资料的同时，了解到 RabbitMQ 在 Rails 中的广泛应用。由于 ruby MRI 中 GBI 的限制，导致一个文件无法直接完成多生产者的任务，而 jruby 所对应的 RabbitMQ 插件 march\_hare 无法安装，而我的本机装不上rbx，所以是没有办法通过多线程生产者模式来在一个文件中完成。于是通过同时执行两个文件来模拟多个生产者的情况。因此，我学习了一点shell脚本语言，来完成本次实验。收货颇丰。

**参考资料(Reference)**

[1] <http://rubybunny.info/articles/exchanges.html> (How to use ‘exchange’ function on bunny)

[2] <http://rubybunny.info/articles/getting_started.html> (Most important example to introduce how to use ‘RabbitMQ’ with bunny in ruby)

[3] <https://www.rabbitmq.com/tutorials/tutorial-one-ruby.html> (A detailed tutorial to explain how the ‘Hello World’ program on RabbitMQ work)

[4] <https://www.rabbitmq.com/#getstarted> (The official website of ‘RabbitMQ’)

[5] <http://rubybunny.info/> (The official website of ‘Bunny’)

[6] <https://www.rabbitmq.com/install-windows.html> (Install ‘RabbitMQ’ on Windows 10 tutorial)