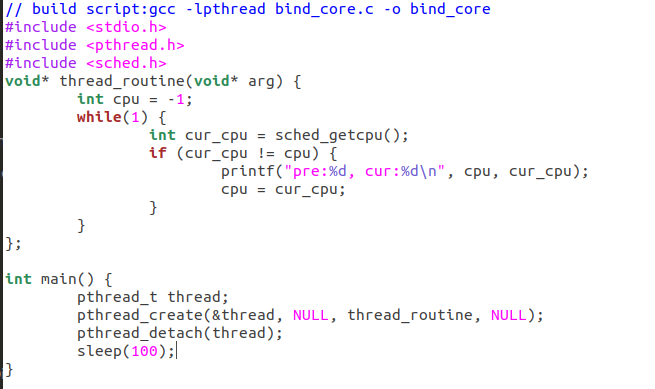
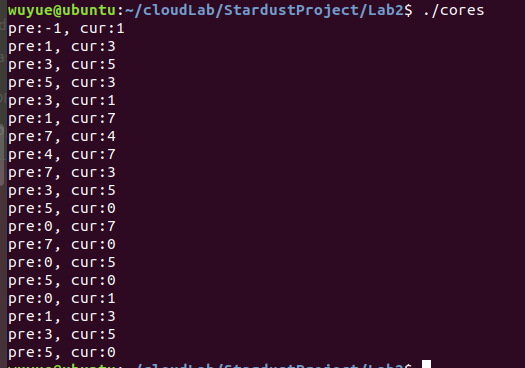
**1.在各种服务器计算机环境上运行时，测试您的服务器每秒可以处理多少个HTTP请求。例如，更改服务器CPU内核数，启用/禁用超线程等。**

Cores.c这段代码中启动了一个线程，通过sched\_getcpu函数不停检测当前占用的逻辑核心编号。



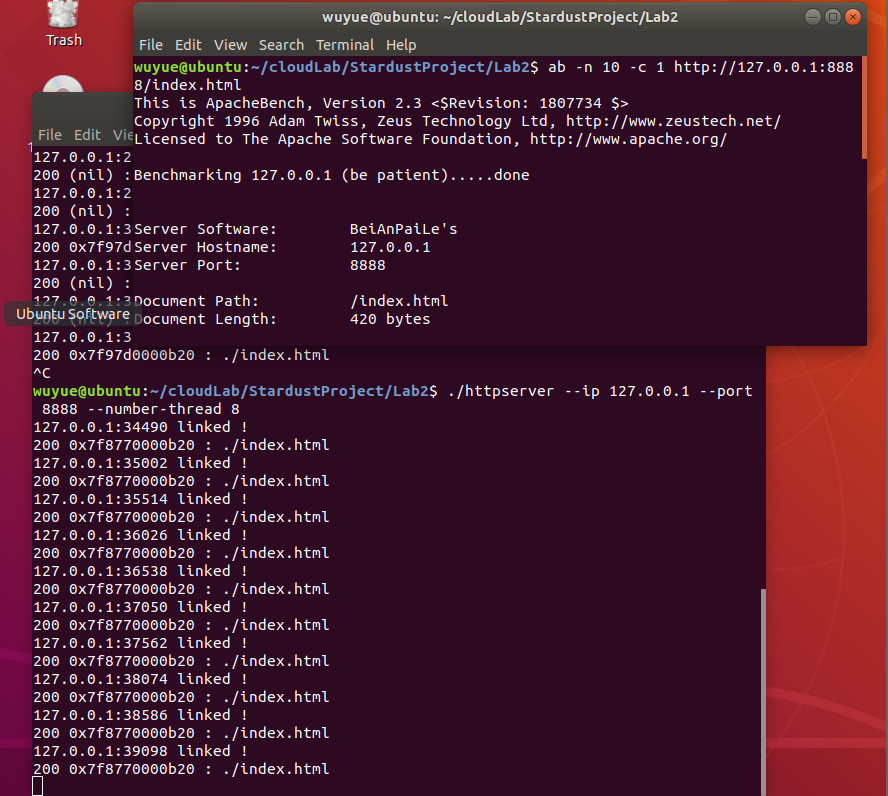
输出如下：



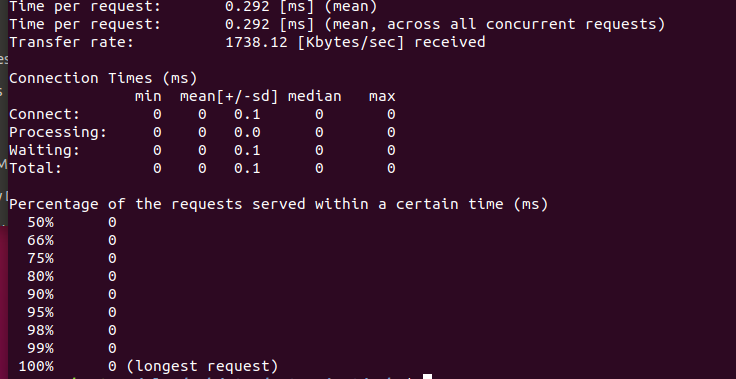
上图可以看出，程序分别在：0,1,2,3,4,5,6,7号逻辑核上运行过。为了让CPU在固定的核心上执行，我们可以使用taskset指令，让程序绑定逻辑核心。

-------------------------------------------------------------

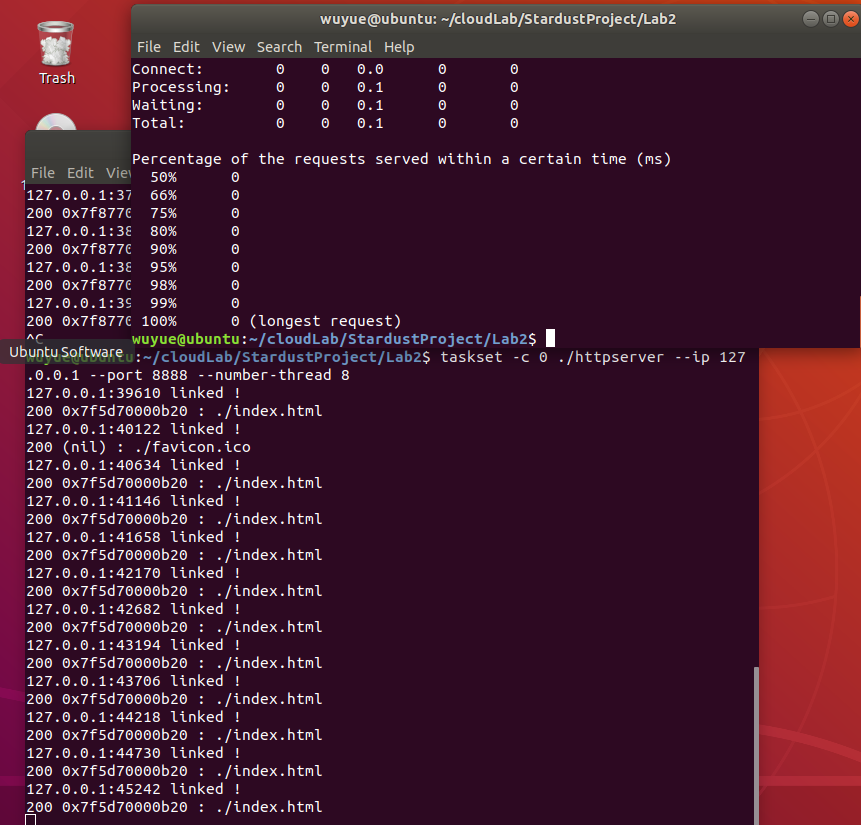
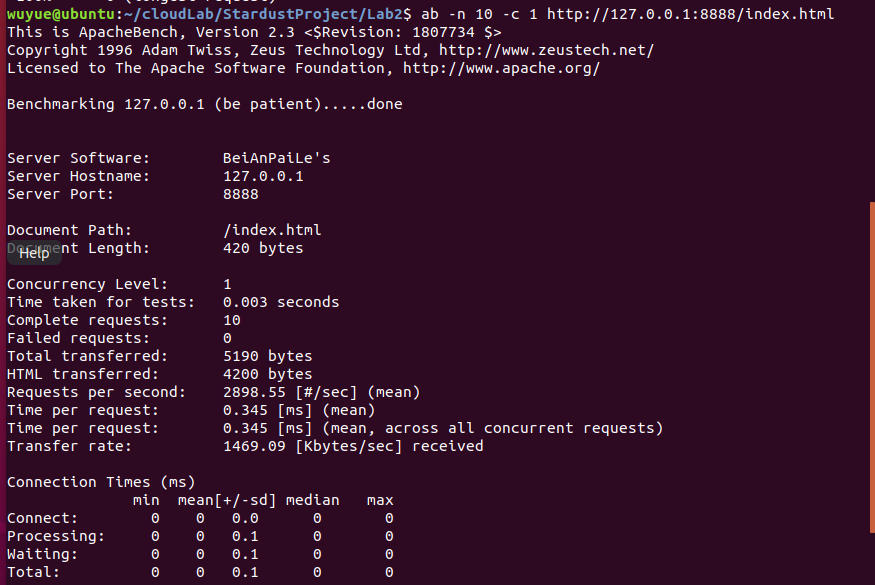
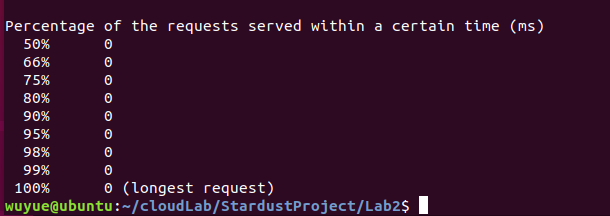
原始状态：



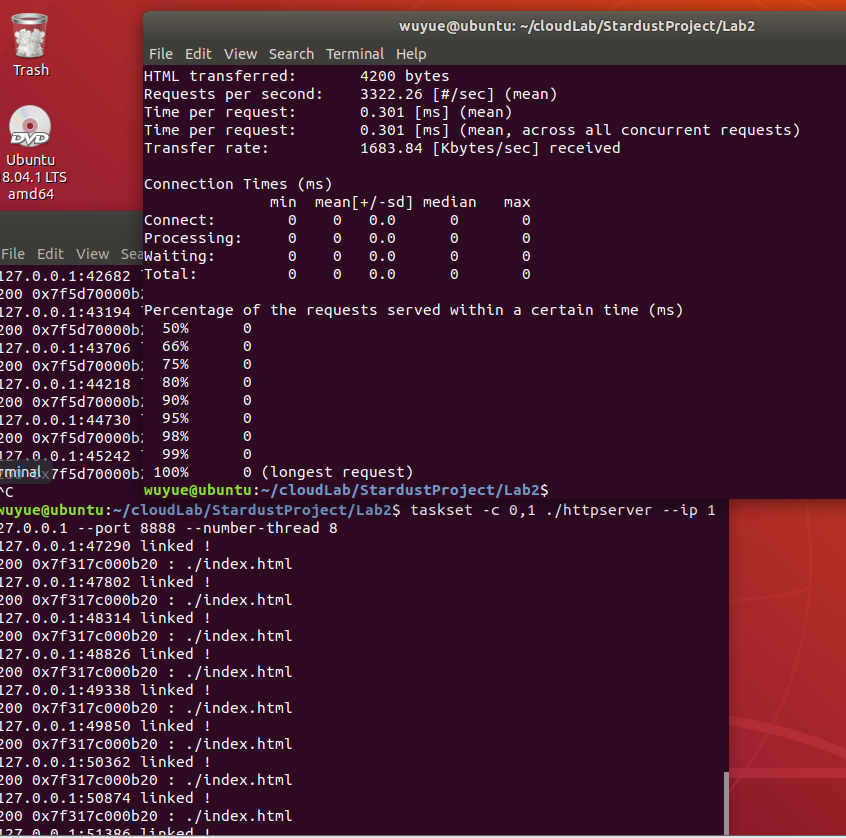
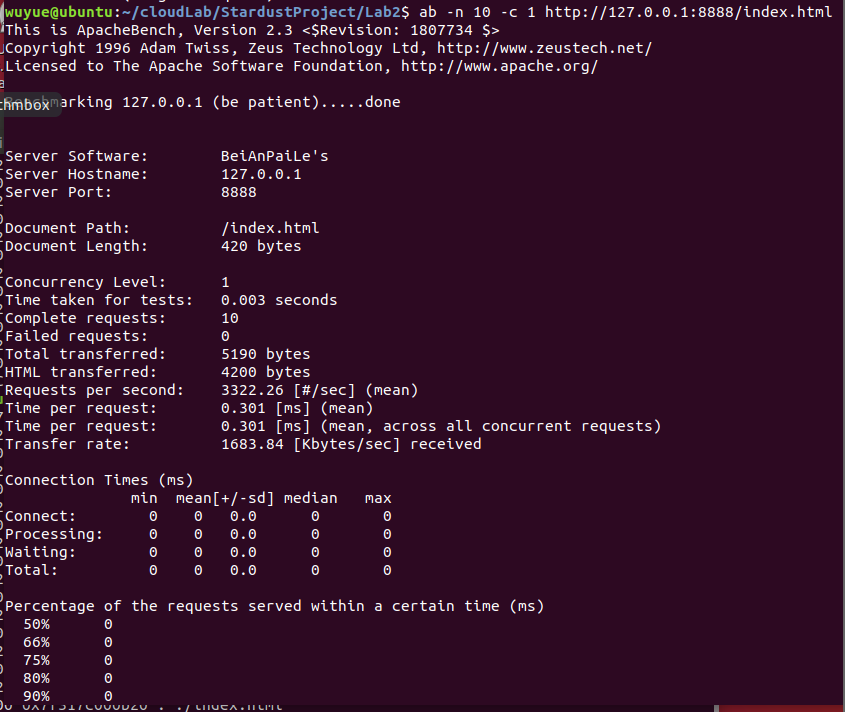
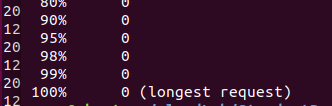




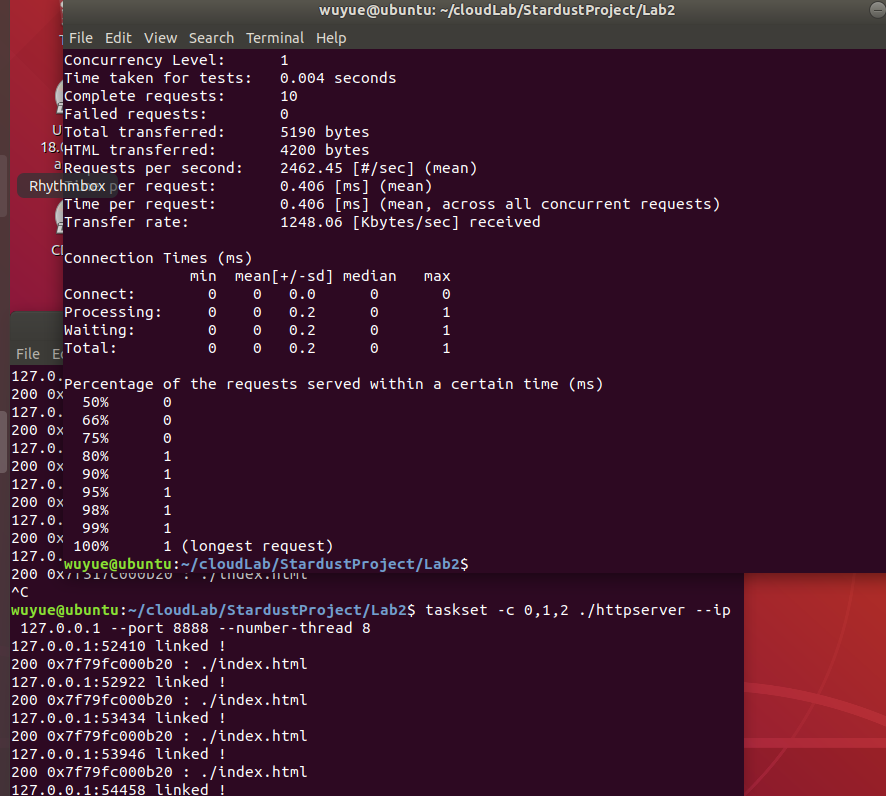
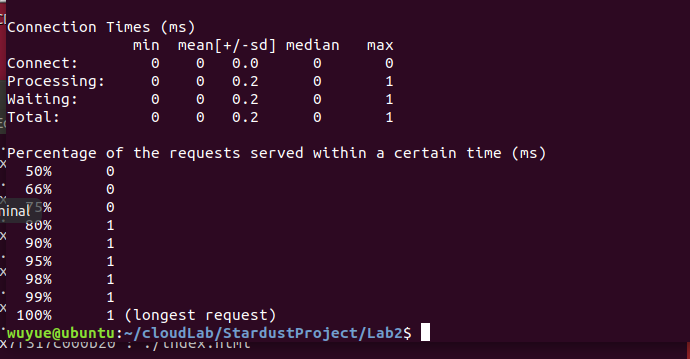
绑定在一个核上：

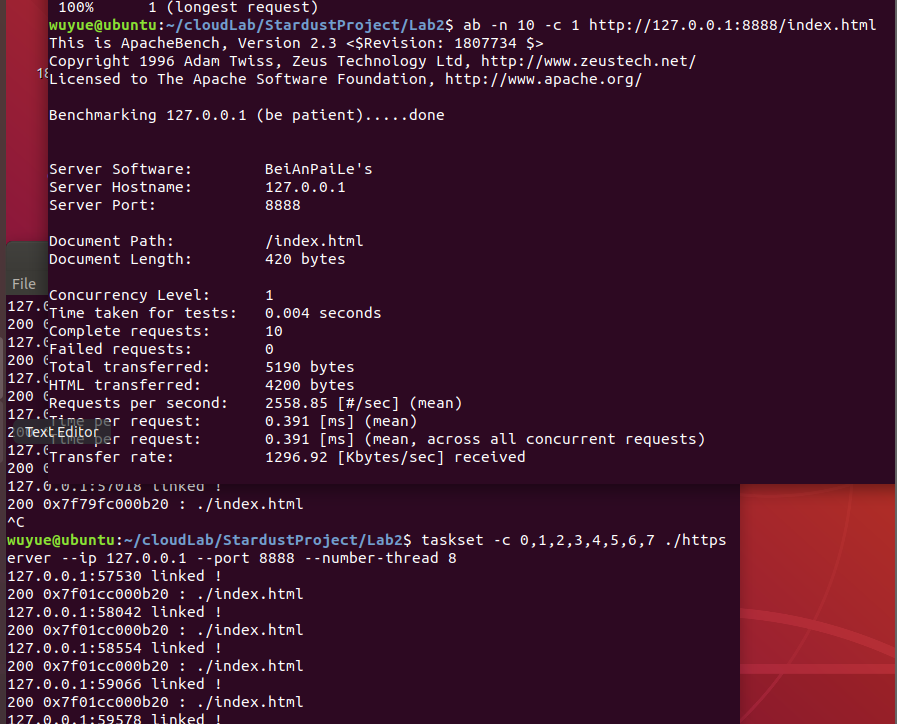
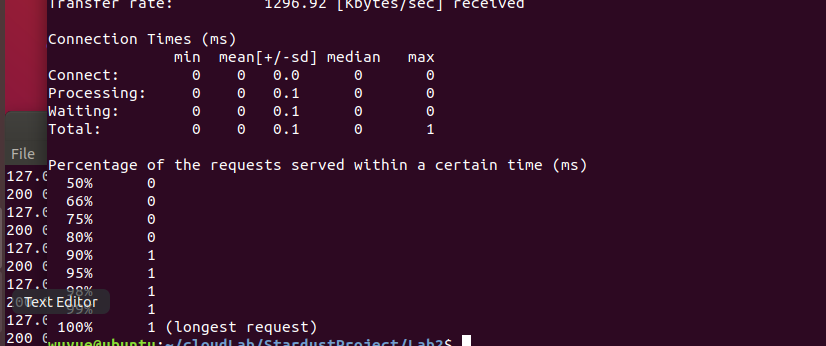
绑定在2个核上：

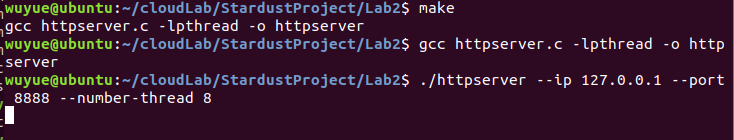
绑定在3个核上:

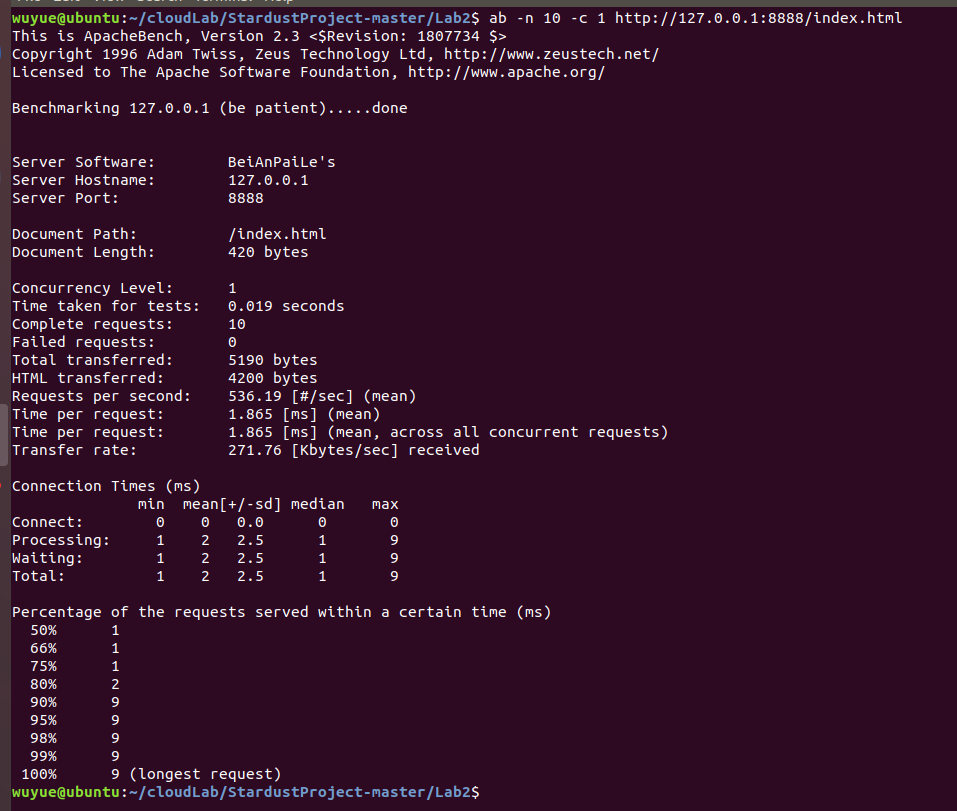
绑定在8个核上：

**2.通过更改同时向服务器发送请求的并发客户端数，测试您的服务器每秒可以处理多少个HTTP请求。一定要改变客户的工作量。例如，测试客户端何时将新的TCP连接用于新请求，或何时客户端将旧的TCP连接重新用于新请求。此外，如果实施高级版本，请尝试更改同一客户端的TCP连接上的出站请求数。您可以编写一个自己发送HTTP Get的简单客户端（可以在同一台计算机上运行多个客户端程序来模拟多个客户端），也可以使用一些现有的HTTP客户端测试工具，例如ab-Apache HTTP服务器基准测试工具。**



1. 请求数：10 并发数：1



**服务器信息：**



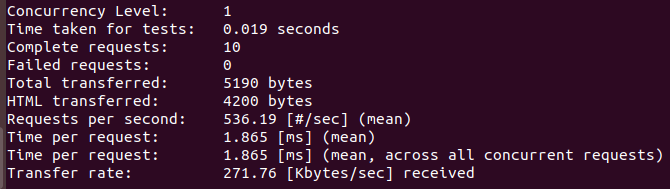
这段展示的是web服务器的信息，可以看到服务器采用的是BeiAnPaiLe’s，域名是127.0.0.1，端口是8888

**文档信息：**



这段是关于请求的文档的相关信息，所在位置“/index.html”，文档的大小为420bytes（此为http响应的正文长度）

**压力测试的几个重要指标：**



Concurrency Level: 1

//并发请求数

Time taken for tests: 0.019 seconds

//整个测试持续的时间

Complete requests: 10

//完成的请求数

Failed requests: 0

//失败的请求数

Total transferred: 5190 bytes

//整个场景中的网络传输量

HTML transferred: 4200 bytes

//整个场景中的HTML内容传输量

Requests per second: 536.19 [#/sec] (mean)

//吞吐率，相当于 LR 中的每秒事务数，后面括号中的 mean 表示这是一个平均值

Time per request: 1.865 [ms] (mean)

//用户平均请求等待时间，相当于 LR 中的平均事务响应时间，后面括号中的 mean 表示这是一个平均值

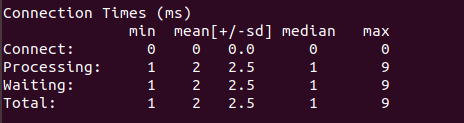
Time per request: 1.865 [ms] (mean, across all concurrent requests)

//服务器平均请求处理时间

Transfer rate: 271.76 [Kbytes/sec] received

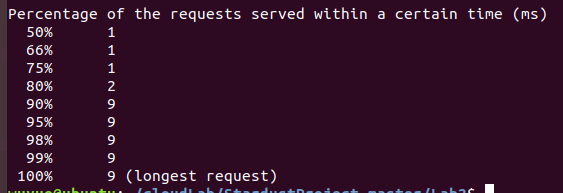
//平均每秒网络上的流量，可以帮助排除是否存在网络流量过大导致响应时间延长的问题

**网络消耗时间：**

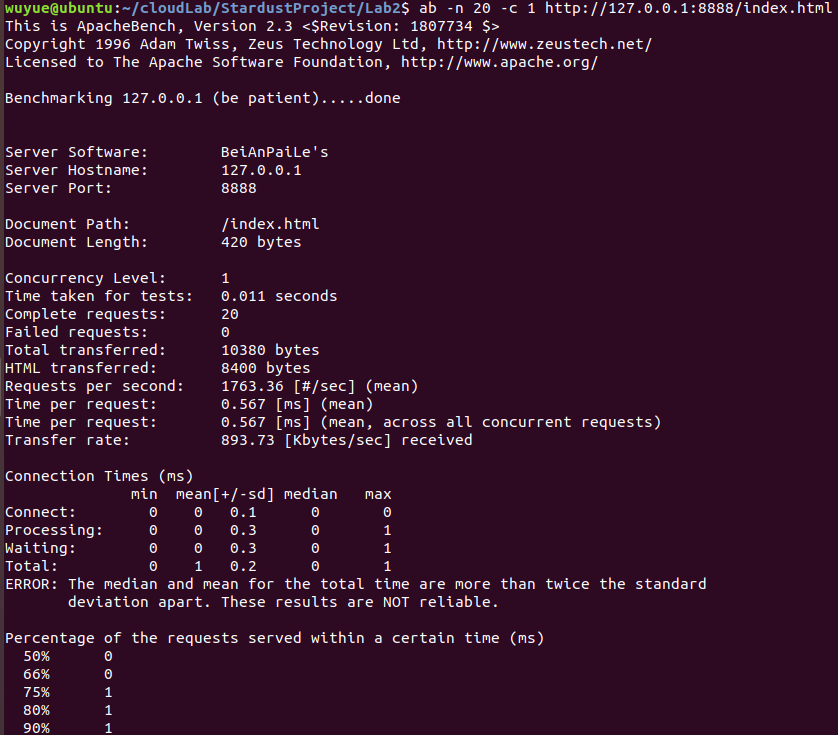
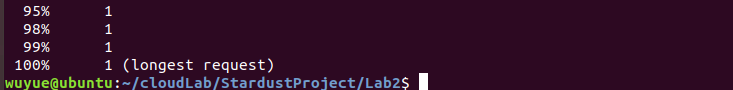


**响应情况：**

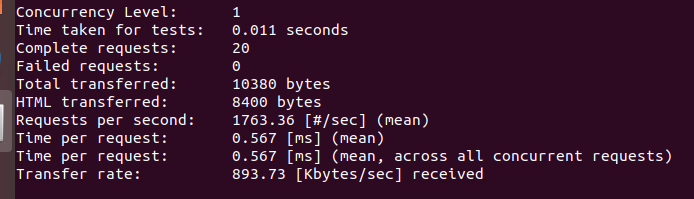
这段是每个请求处理时间的分布情况，50%的处理时间在1ms内，66%的处理时间在1ms内...，重要的是看90%的处理时间。



1. 请求数：20 并发数：1

压力测试的几个重要指标：



Concurrency Level: 1

//并发请求数

Time taken for tests: 0.011 seconds

//整个测试持续的时间

Complete requests: 20

//完成的请求数

Failed requests: 0

//失败的请求数

Total transferred: 10380 bytes

//整个场景中的网络传输量

HTML transferred: 8400 bytes

//整个场景中的HTML内容传输量

Requests per second: 1763.36 [#/sec] (mean)

//吞吐率，相当于 LR 中的每秒事务数，后面括号中的 mean 表示这是一个平均值

Time per request: 0.567 [ms] (mean)

//用户平均请求等待时间，相当于 LR 中的平均事务响应时间，后面括号中的 mean 表示这是一个平均值

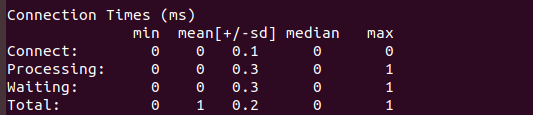
Time per request: 0.567 [ms] (mean, across all concurrent requests)

//服务器平均请求处理时间

Transfer rate: 893.73 [Kbytes/sec] received

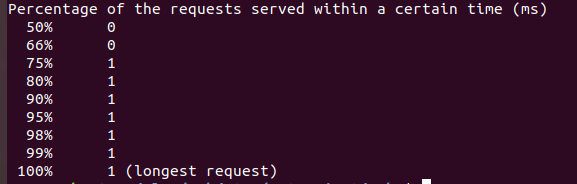
//平均每秒网络上的流量，可以帮助排除是否存在网络流量过大导致响应时间延长的问题

网络消耗时间：



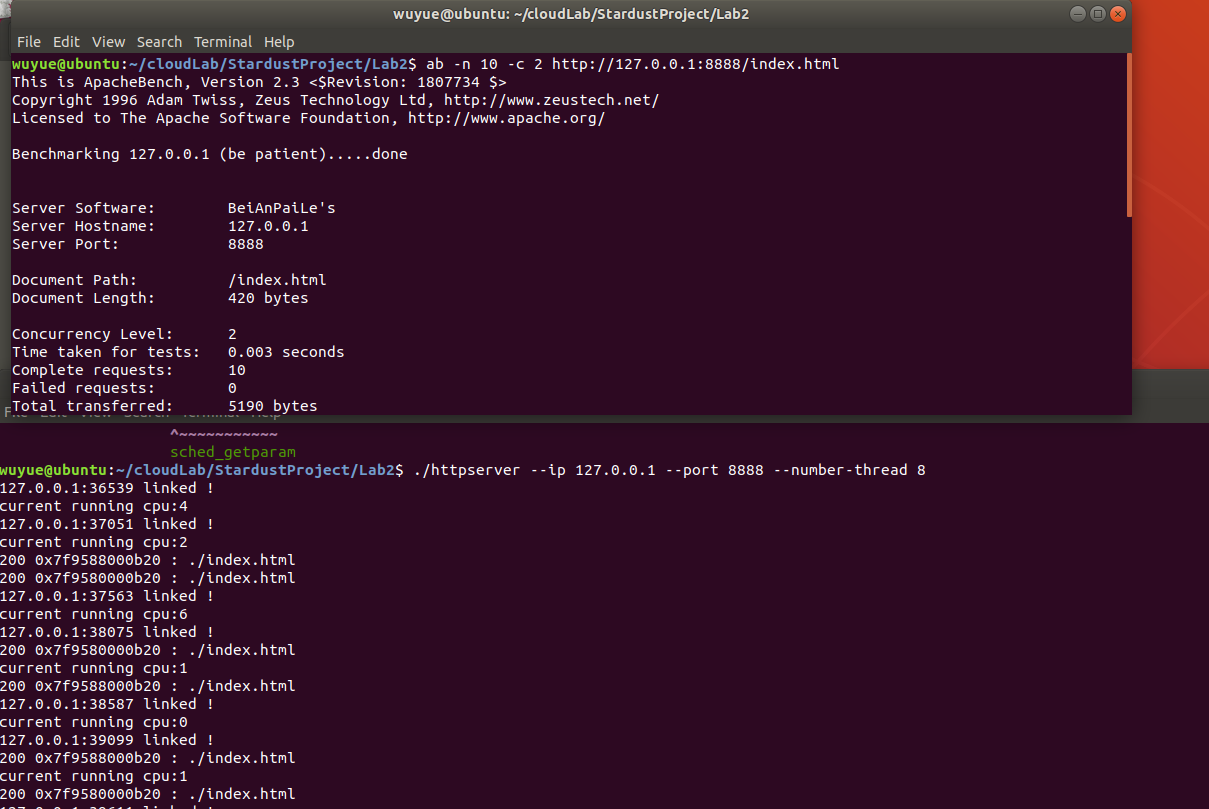
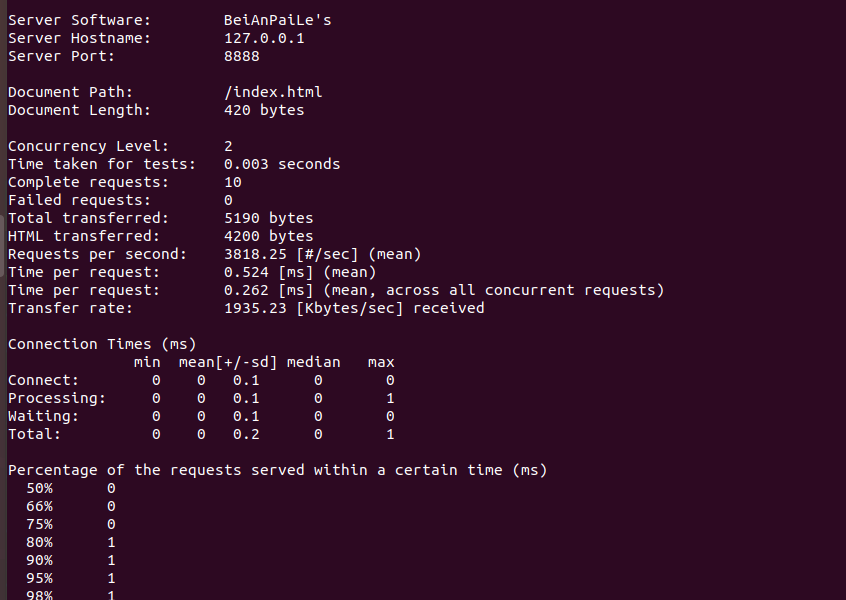
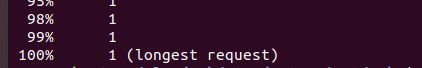


响应情况：

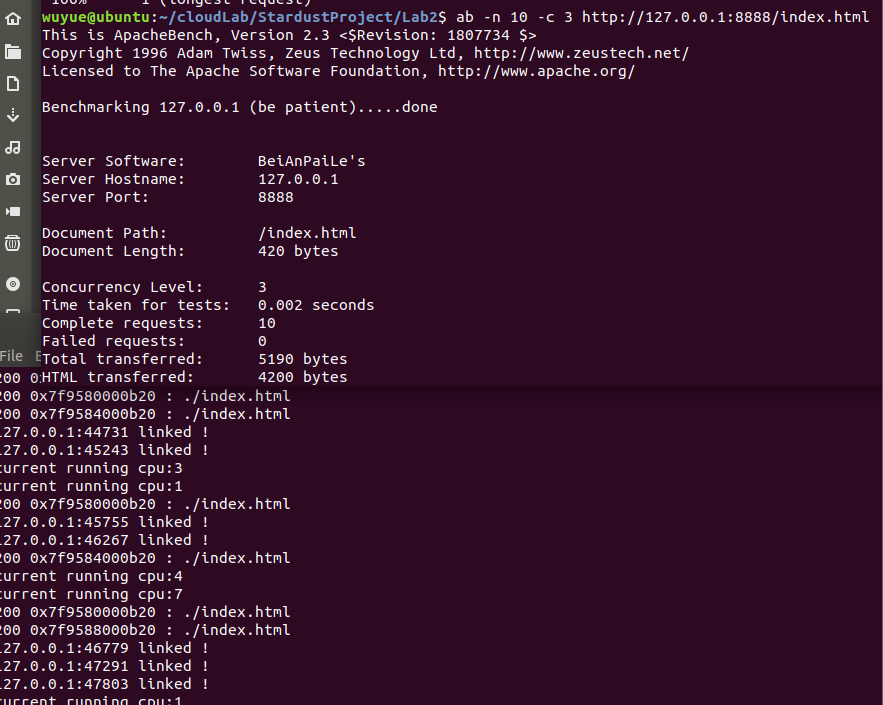
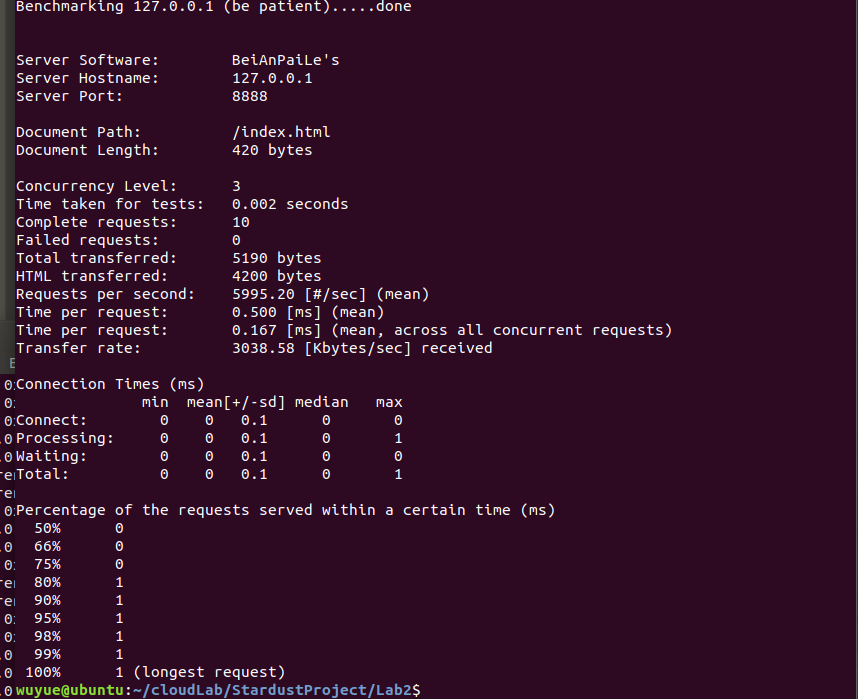


这段是每个请求处理时间的分布情况，75%的处理时间在1ms内，80%的处理时间在1ms内……

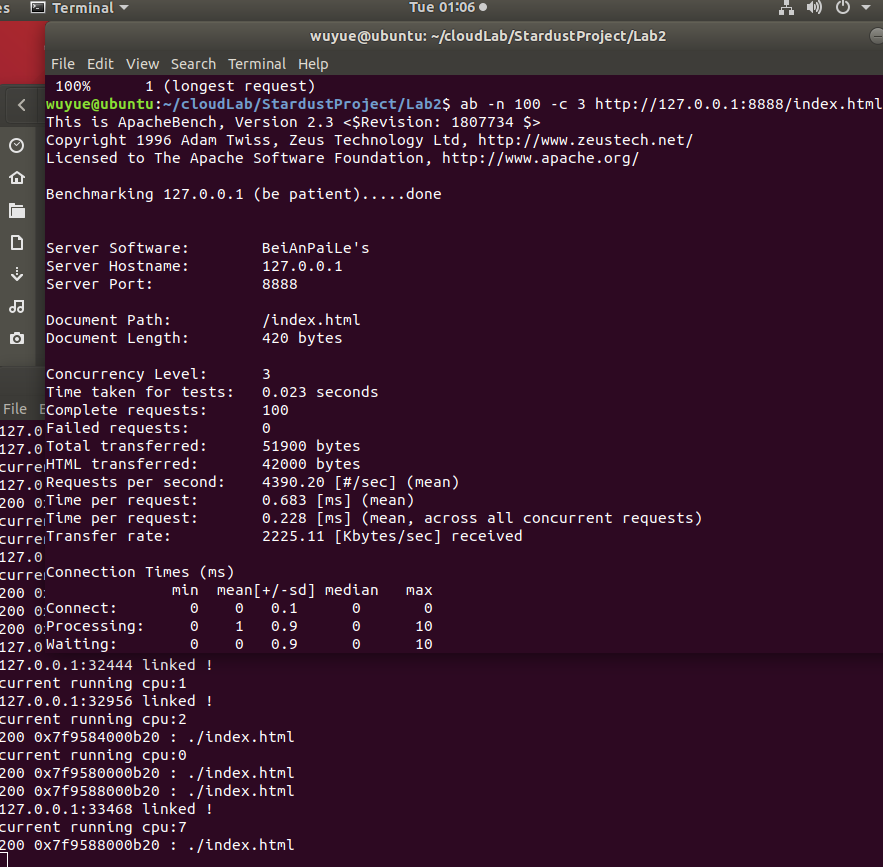
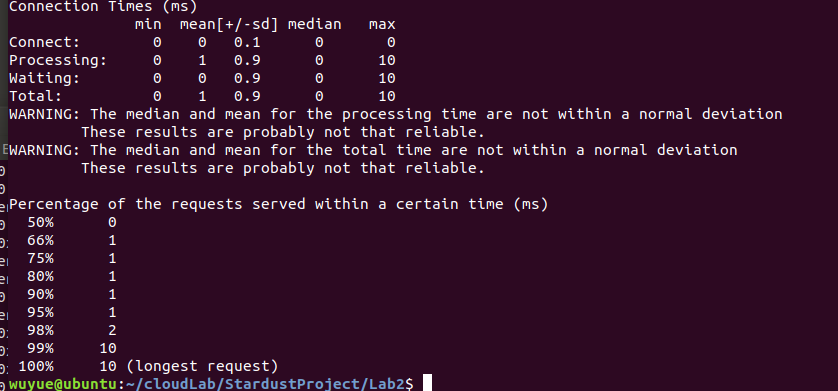
1. 请求数：10 并发数：2

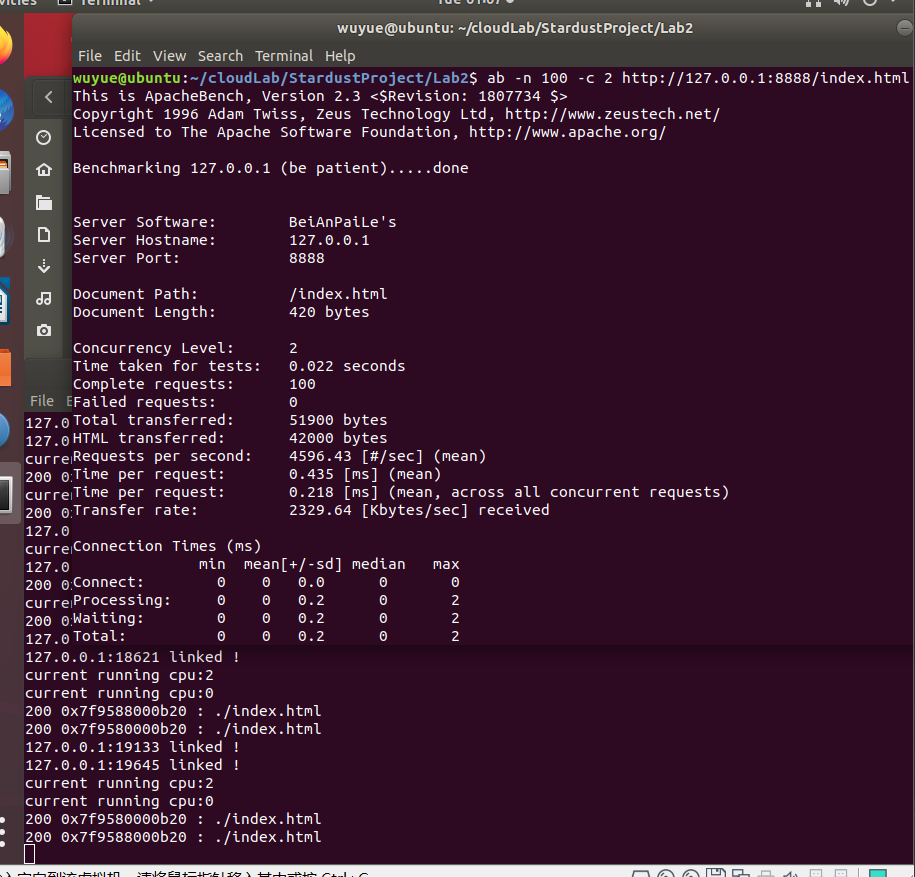
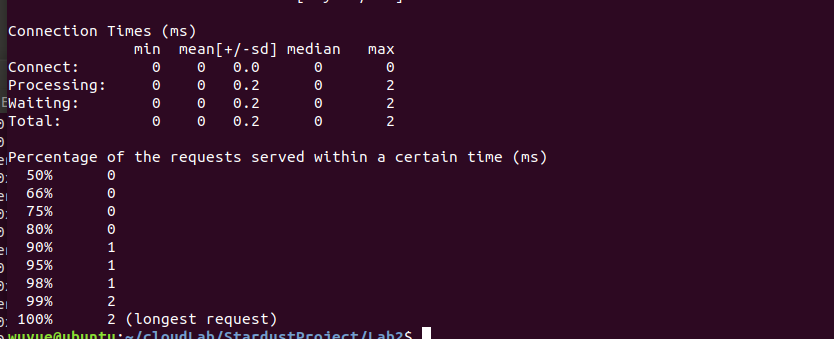
1. 请求数：10 并发数：3

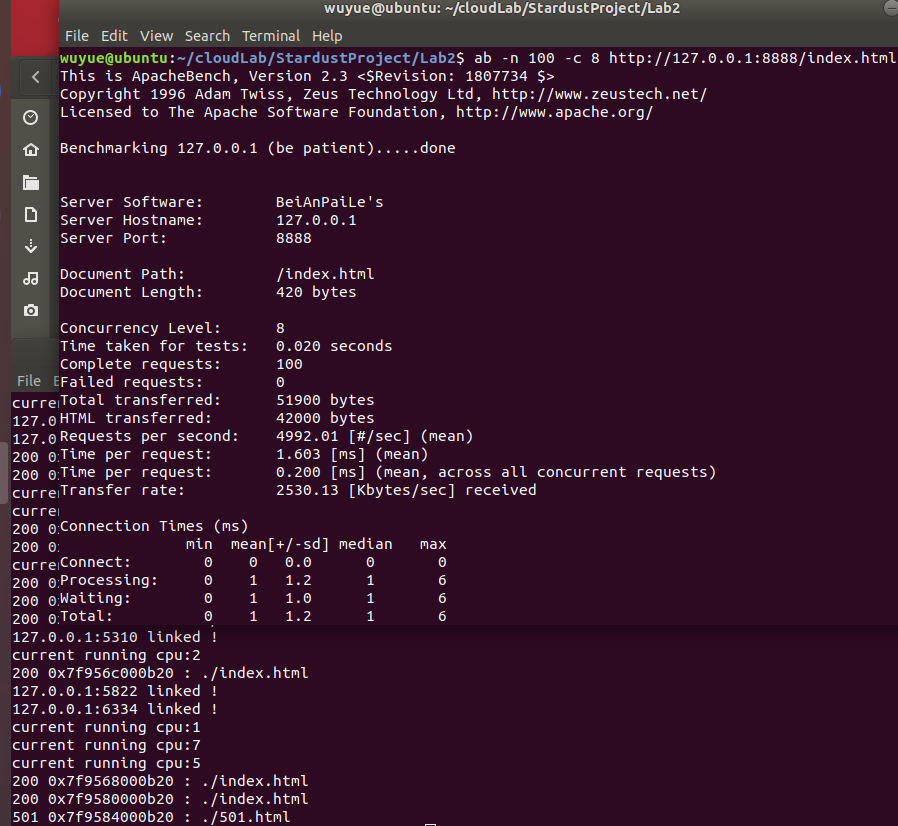
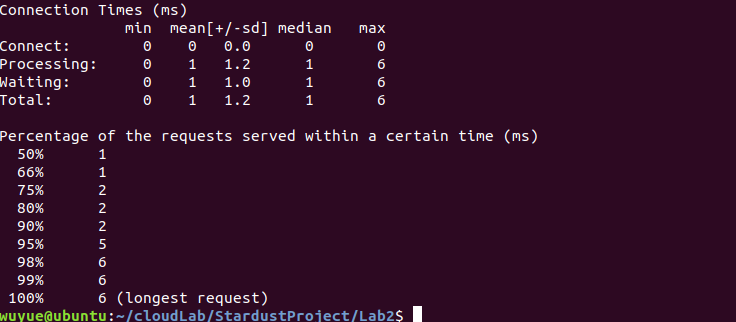
请求数100 并发数3

请求数100 并发数2

请求数100 并发数8

请求数10 并发数8

