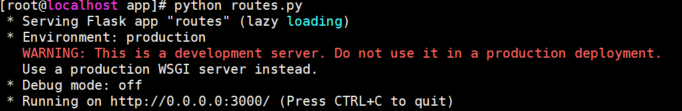
How to start the program:

In the 10.239.85.85 machine(IP may be changed ask Li Xiaoran):

/home/xiaoran/ui\_pmem/pmem\_demo/app

Things In this path is the right version, not be misled by new\_ui or the pmem\_demo in the /root.

Type in command: python routes.py



Access in your Chrome browser with 10.239.85.85:3000

If not responsive, type in command: iptables -F to close firewalls and access again

To debug this program, you had better use hard refresh to clean the cache.

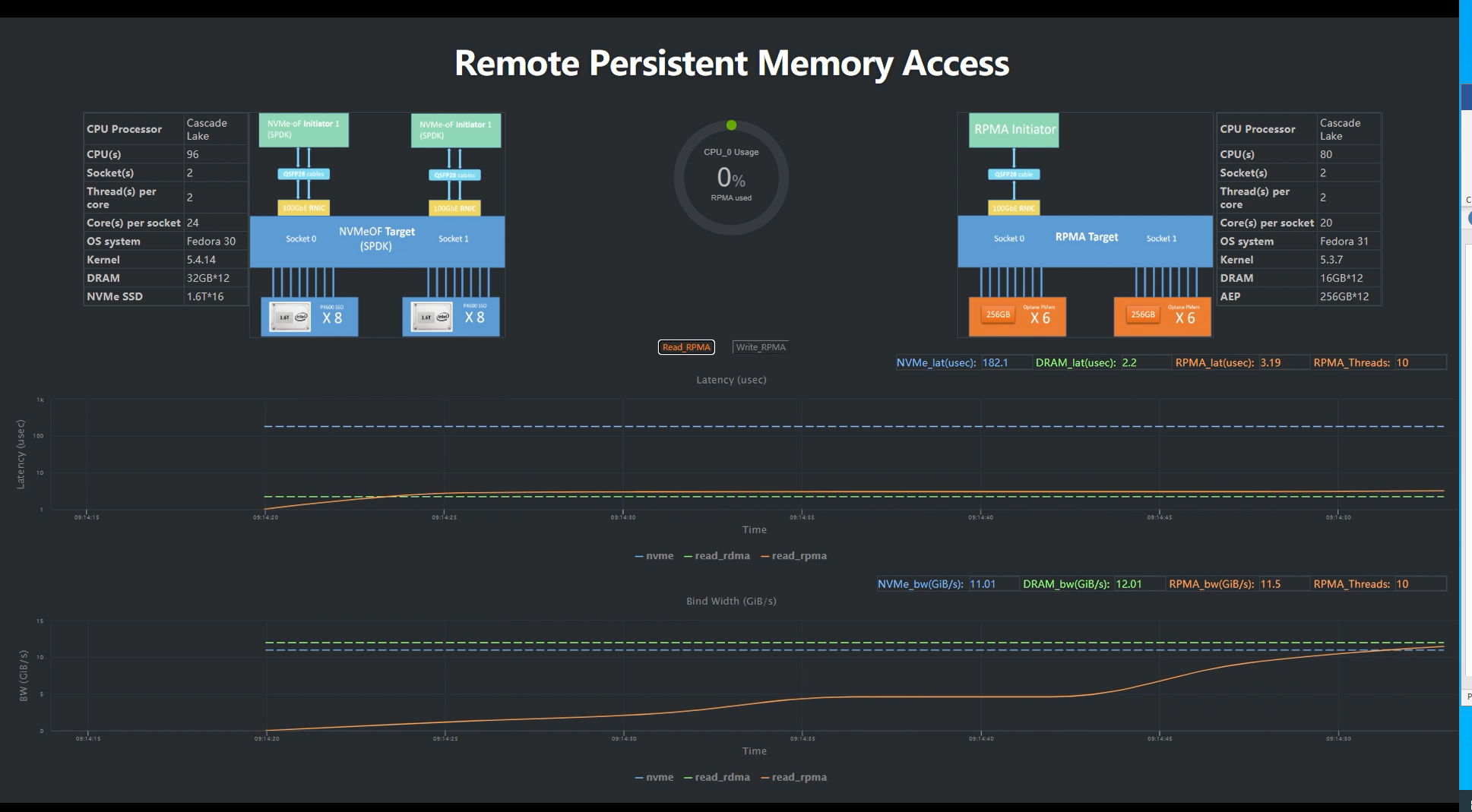
If the server’s files are lost, access my personal github: <https://github.com/hfyxqxxing/pmem_UI>

(There are also versions before this one with some other functions. )

\*现在这个程序不能正常运行，因为fio脚本环境有变，测试相关条件另行了解。可以运行一下看个大概，数据有错，终端需要及时中断。

以下是使用手册

Intel AEP observing system



Introduction

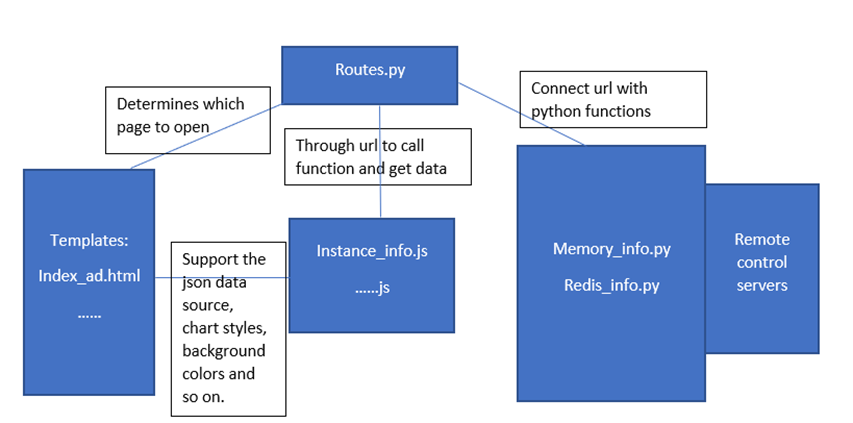
This web application is generally aimed to monitor the fio test results in real time. In this version, it compares the bind width and latency of Remote Persistent Memory Access(RPMA) and of Remote Direct Memory Access(RDMA) and of NVMe tests to demonstrate the development of the pmem technology and its advantages. It was used in the 2020 tencent techo park meeting for pmem group worker to introduce pmem to more people in this field. So this version is specially designed for that presentation. If you want to reuse this program, you should understand fully on how this program works.

这是一个展示rpma测试和rdma测试的结果对比的web程序。

Basic Logic

Front-end

Back-end



As you can see here, it is a flask web application. Front-end is responsible for the page design, and information demonstration. Back-end controls the state of process, data generation, supports the application to keep running on the server. There are also connections between front-end and back-end. Here we use jquery to support javasrcipts to make web page better and ajax to communicate between front-end and back-end. Here are the files used in front-end and back-end:

这是一个flask程序，分前后端，分别是html, js文件的前端和几个python文件的后端

Front-end:

Index\_demo.html

Instance\_info\_bw\_demo.js

*(Environments)*

*Export-data.js*

*Exporting.js*

*GaugeMeter.js*

*Highcharts.js*

*Jquery-2.1.1.min.js*

*And files in /css, /fonts, /img*

Back-end:

\_\_init\_\_.py

Routes.py(The foundation file)

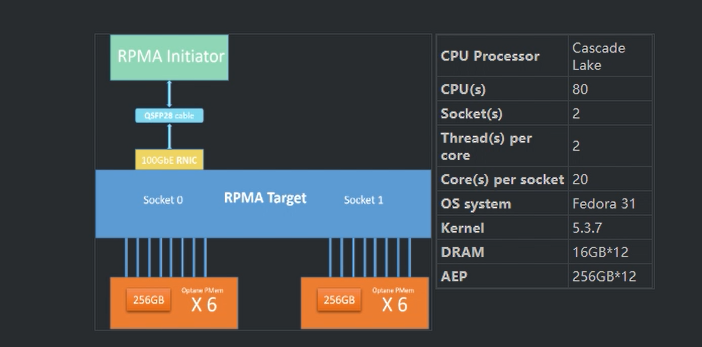
Memory\_info.py

Redis\_info.py

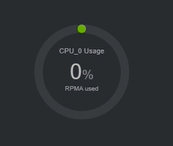
较大字体的文件是主要修改和影响程序的地方

This web application can be simply seen as two parts. The above part is the information of conditions. Whether the server’s condition is good or bad also determines the performance of hardware tests to a great extent. So you should use a dictionary variable to return the real-time values of the machine with specified commands. In the back-end: create and send the content variable by render\_template (xxx.html , content) method. In the front-end: directly use content.cpus to show the conditions of the server.

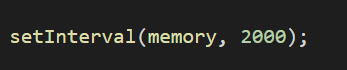
网页页面分上下两部分，上面一部分是机器测试的环境展示。可以通过render\_template 直接调用



There is also another CPU usage monitor to display the usage of CPU in real time to prove the performance of tests.



As it has to be a real-time monitor, the data should be refreshed in a fixed time. Like every two seconds change one time. So we need a timer to execute the related function. In javasript, we have a simple function called setInterval(functionX, 5000) to run the functionX every 5 seconds. Memory() function in instance\_info\_bw.js catch the data by ajax and show it in webpage by element ID.



这是一个监控cpu利用率的部件，通过js文件里的定时器定期获取机器现在状态并展示到页面上

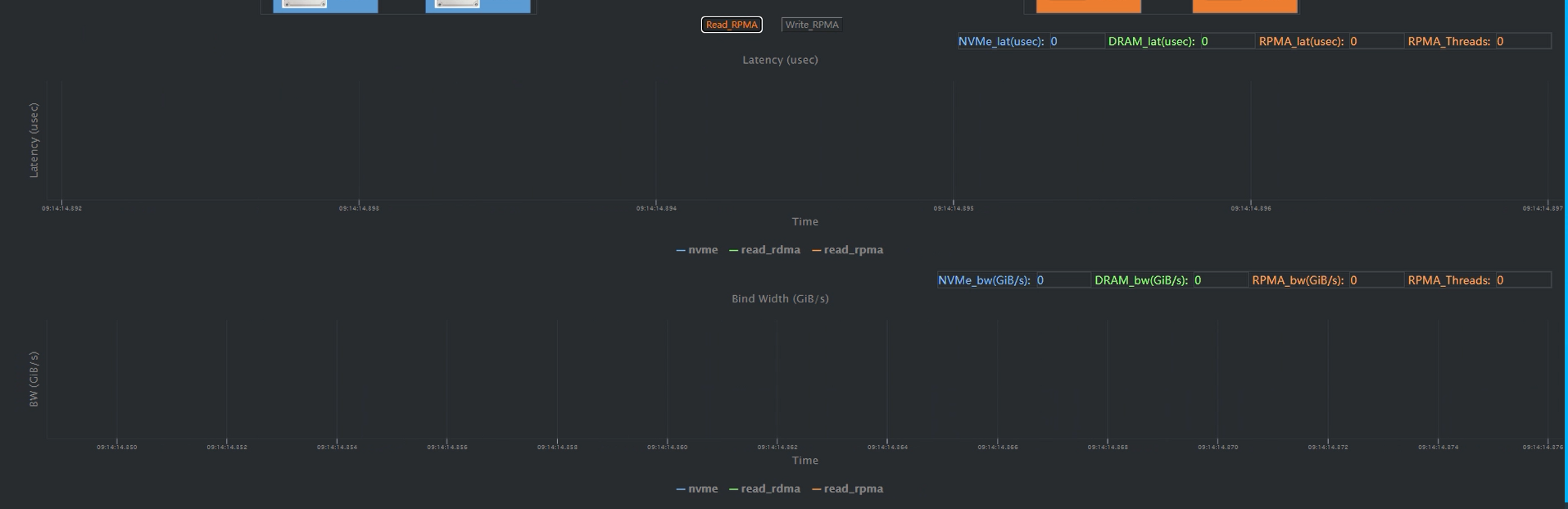
The second part is constructed by two highcharts to display the trend of test results.

The basic chart styles and more detailed instructions can be found in highcharts official website

<https://www.highcharts.com.cn/> <https://www.highcharts.com/>

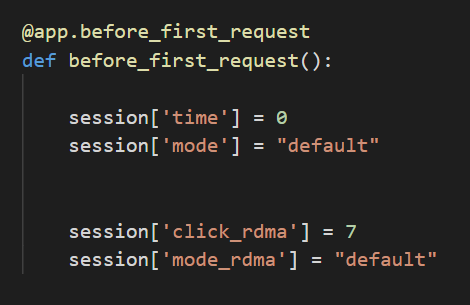
It is a useful tool to help you make charts and deal with data.

页面下半部分是两个highcharts图表，具体细节可以上官网看。



The main idea of this chart in our program is that we control the state of data, or to say which line to show in the back-end: routes.py file by the “session”.(There are many ways to realize this)

图表有个关键要素是决定展示的是哪条线和对应数据。这个变量放在routes.py 进行控制。



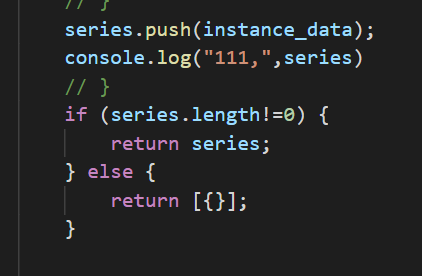
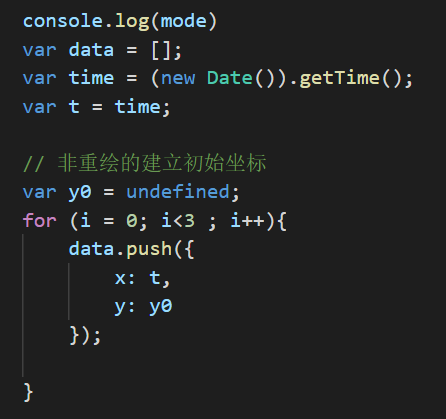
It is because as this real-time page needs to refresh and access again and again, only in routes.py we can record a global variable that will not be cleared by refreshing data.

作为一个定时刷新的flask程序，只有routes.py里的session变量不会被重复访问刷新消失导致无法控制。

Then this variable will be used in both front-end and back-end:

In the front-end, the highcharts requires a preset data series to become a chart, especially in our real-time chart. Then we use the “mode” variable to give the series(one line in the chart) a specific name to generate an initial empty chart.

这个变量在前端用来创建highcharts的空数据组。



Then go to the back-end: we use this variable to execute the corresponding test shell script, access them and catch data that we need.



后端用来访问对应路径的脚本执行测试

Two points need to be mentioned here is:

Python的命令行执行有两个坑：

First, the os.system command is also a line in python, so typically only when it is all executed then the next command will start. It reflects in real case means that the web page will be stuck and not accessible by mouse or keyboard as the program has stopped to do the test script. So we need a multi-thread method to keep both the test script and web running.

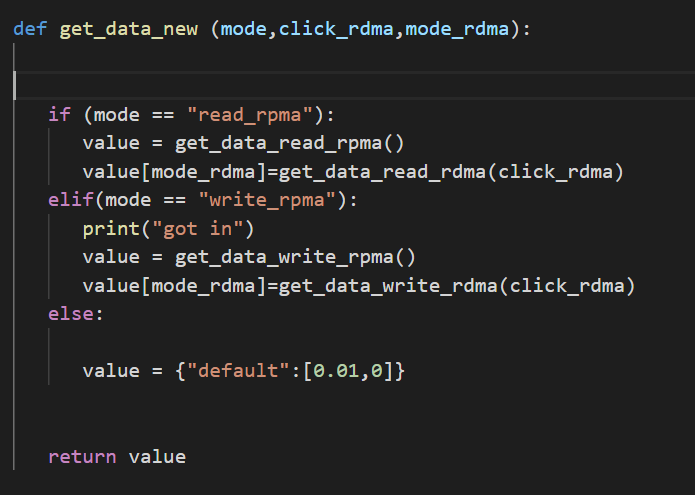
多线程执行脚本才不会影响flask运行。

Second, all the os.system is executed in a new generated python shell program. It means that it is naturally started in the path of your routes.py. And there is no use to cd in this python file. The python shell will be shut down once the os.system command is finished. So we should only directly execute the commands with absolute path or commands without path. Detailed commands should all be included in the shell script you want to execute, of course they should also be written with absolute path.

Python里执行脚本需要绝对路径来正确执行终端命令，最好通过脚本一次性执行。

Access the data by the mode

也通过这个变量获取数据



(value here is a dict type variable)

Inside the get\_data\_read\_rpma, we go to the right path and read file. Use regex to catch and deal with the data we need. (like bind width and latency)

So the returned value is a dict variable like {“read\_rpma” : [x0,y0] , “read\_rdma” : [x1,y1] }. The instance\_info\_bw\_demo.js accepts such data and add this point to the corresponding line.

访问对应路径的文件返回字典给前端



With another setInterval function to keep addPoints().By the co-work of front-end and back-end, the web page can show the test results in real time.

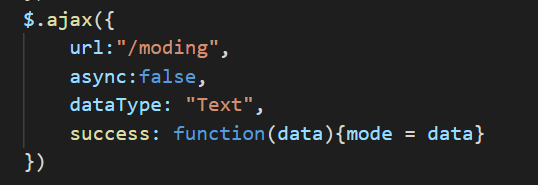
用另一个定时器固定更新图表的时间

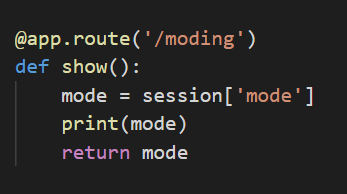
Ajax

Front-end and Back-end should work together and pass data to each other to run this program. They need a method to pass and connect. In this program, we use jquery and its ajax method to connect both ends and the key is the “url”

这个程序用AJAX来连接前后端传输数据

First kind of ajax:

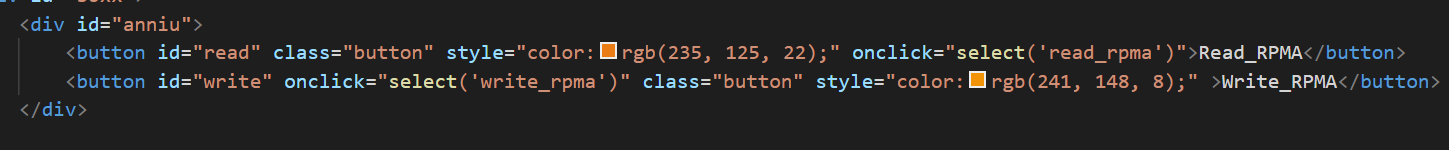




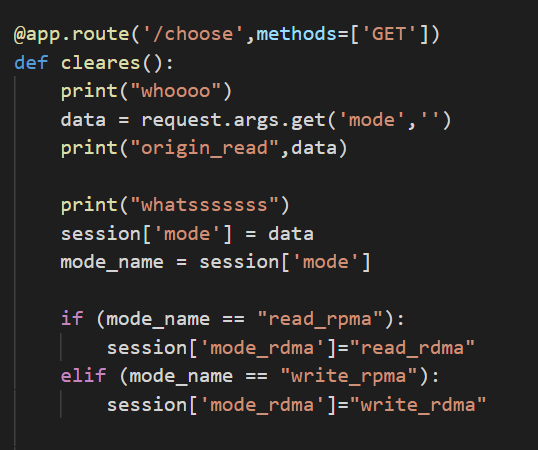
A simple way to get the value from the back-end. Access the moding url and run the show() function and return this value as “data”, assign this “data” to the variable in js file.

这个ajax单纯用来获取后端变量数值并赋值给js里的变量

Second kind of ajax:



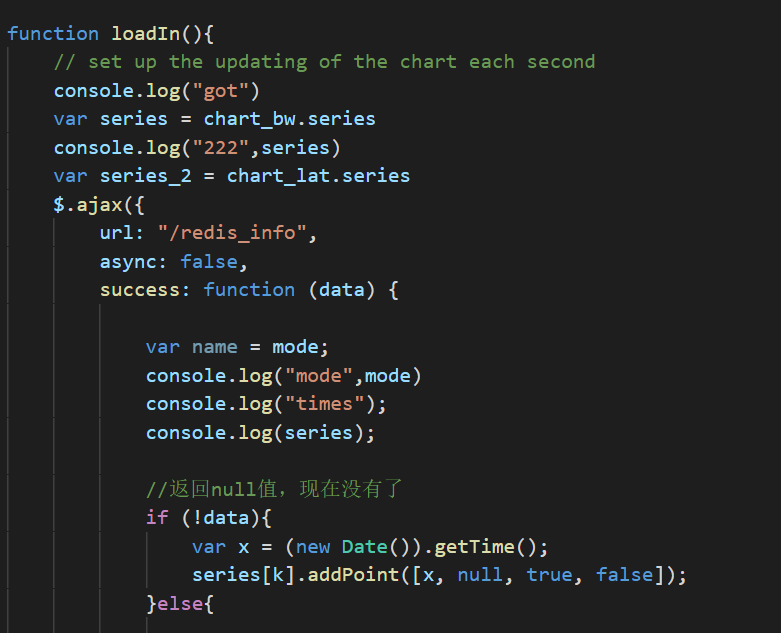


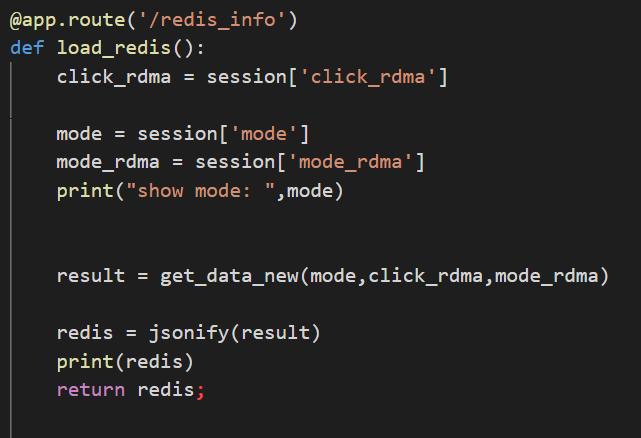


This one requires the interaction through the web page button to js file and then the back-end. What makes difference is that it passes a value from the web page(triggered by user click) in this process. In the program we use this button to change lines to show. So we should pass the new state of “mode” to the routes.py, and it is realized by the above methods.

这个ajax涉及从html到js到后端，用户点击一个按钮，传值给js然后再传值给后端并执行对应函数。

The third kind of ajax:





This kind focus on triggering the function in the back-end to run and catches its data and then run the commands after the data is acquired. In this program, we use this LoadIn to add points to the chart.

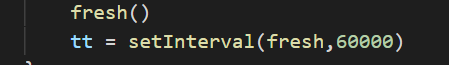
这个主要是从后端获取数据然后等到数据获取成功后执行其他命令比如增加新的数据点。

By more methods of using ajax to connect javascript and python functions, you can realize more requirements by your thoughts.

掌握ajax的互动方式，就能实现更多功能。

Except the main logic of access and return data above, there are also another relationship between the front-end and back-end to realize the function of refreshing the whole test process. As time moves, the test data will be too much for computer to deal and visitors to understand. So every 6 minutes the chart will be cleared and start a new test automatically again. The logic is simple, clear the origin data and add new data lines and restart the shell. To get more details, you can access the program and read it by yourself.

这个程序还有一个刷新页面的逻辑，定时刷新页面提供给流动的观看者从头开始的体验。可以自行查看对应代码并理解这个功能的实现方式。



\*\* 还有一份更啰嗦基础繁琐的全英文菜鸟教程: 