به نام ایزد بخشاینده مهربان

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

گزارش پروژه پایانی

درس مبانی رایانش ابری

استاد: دکتر سید احمد جوادی

اعضای گروه:

علی شفیعی - ۹۵۳۱۸۰۱

سید سجاد پیشوائیان – ۹۵۳۱۰۱۵

بهمن ۱۳۹۹

لازم به ذکر است که تمامی بخش‌ها با همکاری هر دو عضو با استفاده از skpye زده شده است و پیاده‌سازی کاملا در هر بخش دو نفره و اشتراکی بوده است.

بخش اول:

توازن بار به فرآیند توزیع مجموعه‌ای از وظایف بر روی مجموعه‌ای از منابع (واحدهای محاسباتی)، با هدف کارآمدتر کردن پردازش کلی آنها اشاره دارد. تکنیک‌های توازن بار می توانند زمان پاسخگویی را برای هر کار بهینه کنند و از پربار شدن ناهموار گره‌های محاسباتی در حالی که سایر گره‌های محاسباتی بیکار هستند جلوگیری می کند. هدف از توازن بار، بهینه‌سازی استفاده از منابع، بیشینه ساختن توان عملیاتی، به حداقل رساندن زمان پاسخ و جلوگیری از هرگونه بار اضافی بر روی یک منبع می‌باشد. استفاده از اجزاء متعدد (چندگانه) به همراه توازن بار، به جای یک جزء، موجب افزایش قابلیت اطمینان و دردسترس بودن از طریق افزونگی (منابع) می‌شود .دو رویکرد اصلی وجود دارد: الگوریتم‌های ساکن، که وضعیت ماشین‌های مختلف را در نظر نمی‌گیرند و الگوریتم‌های پویا ، که معمولاً کلی تر و کارآمدتر هستند ، اما به تبادل اطلاعات بین واحدهای محاسباتی مختلف نیاز دارند که می‌تواند باعث دست دادن کارایی کلی آن شود.

HAProxy ، مخفف High Availability Proxy ، یک نرم افزار محبوب منبع باز TCP / HTTP برای توازن بار است که می تواند در Linux، Solaris و FreeBSD اجرا شود. متداولترین کاربرد آن بهبود عملکرد و قابلیت اطمینان محیط سرور از طریق توزیع بار کاری در چندین سرور (به عنوان مثال وب ، برنامه ، پایگاه داده) است. این در بسیاری از محیط های با مشخصات بالا از جمله: GitHub، Imgur، Instagram و Twitter مورد استفاده قرار می گیرد.

الگوریتم توازن بار تعیین می کند که کدام یک از سرورها در هنگام توازن بار انتخاب می شود. HAProxy چندین گزینه برای الگوریتم ها ارائه می دهد. علاوه بر الگوریتم توازن بار ، می توان به سرورها یک پارامتر وزن اختصاص داد تا میزان انتخاب سرور را در مقایسه با سایر سرورها دستکاری کند.

برخی از الگوریتم هایی که در HAProxy برای توازن بار استفاده می‌شوند به شرح زیر هستند:

۱- Round Robin: این الگوریتم با استفاده از هر سرور پشت تراز کننده بار، به ترتیب با توجه به وزن آنها کار می کند. همچنین احتمالاً منصفانه‌ترین الگوریتم است زیرا زمان پردازش سرورها به‌طور مساوی توزیع می شود. به عنوان یک الگوریتم پویا ، Round Robin اجازه می‌دهد تا وزن سرور در حین اجرا تنظیم شود.

۲- Static Round Robin: مشابه Round Robin ، هر سرور به ترتیب در هر وزن خود استفاده می شود. برخلاف Round Robin ، تغییر وزن سرور در حین اجرا وجود ندارد اما از نظر تعداد سرورها محدودیتی در طراحی وجود ندارد.

۳- Least Connections: در این الگوریتم ، سرور با کمترین تعداد اتصال، اتصال را دریافت می کند. این نوع متعادل سازی بار هنگامی توصیه می‌شود که جلسات بسیار طولانی مانند LDAP، SQL، TSE و غیره انتظار می‌رود. با این وجود برای پروتکل هایی که از جلسات کوتاه مانند HTTP استفاده می‌کنند مناسب نیست. این الگوریتم نیز مانند Round Robin پویا است.

۴- Source: این الگوریتم IP منبع را هش می‌کند و آن را بر وزن کل سرورهای در حال اجرا تقسیم می‌کند. همان IP کلاینت همیشه به سرور یکسانی می رسد به شرطی که سرور روشن یا خاموش نشود. اگر نتیجه هش به دلیل تغییر تعداد سرورهای در حال اجرا تغییر کند ، کلاینت‌ها به سرور دیگری هدایت می شوند. این الگوریتم به طور کلی در حالت TCP استفاده می شود که در آن کوکی ها نمی توانند وارد شوند. به طور پیش فرض نیز ساکن است.

۵- URI: این الگوریتم یا قسمت چپ URI یا کل URI را هش می کند و مقدار هش را به وزن کل سرورهای در حال اجرا تقسیم می‌کند. تا زمانی که هیچ سروری روشن یا خاموش نشود ، همان URI همیشه به همان سرور هدایت می شود. این نیز یک الگوریتم ثابت است و به همان روش الگوریتم Source کار می‌کند.

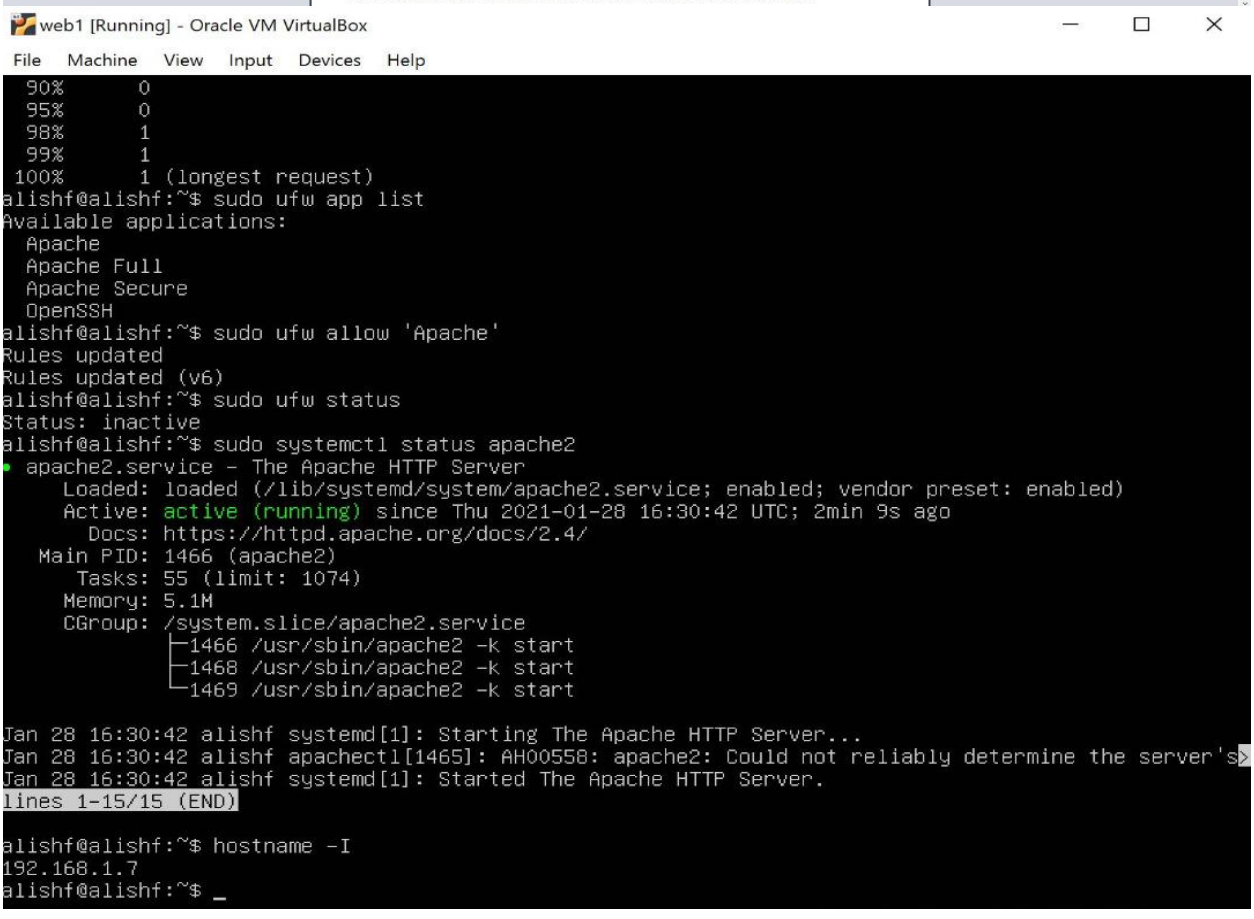
۶- URL Parameter: این الگوریتم ایستا فقط می‌تواند در پس‌زمینه HTTP استفاده شود. پارامتر URL مشخص شده در رشته جستجوی هر درخواست HTTP GET جستجو می‌شود. اگر پارامتر پیدا شده با علامت و مقدار مساوی دنبال شود ، مقدار هش شده و بر وزن کل سرورهای در حال اجرا تقسیم می شود.

بخش دوم:

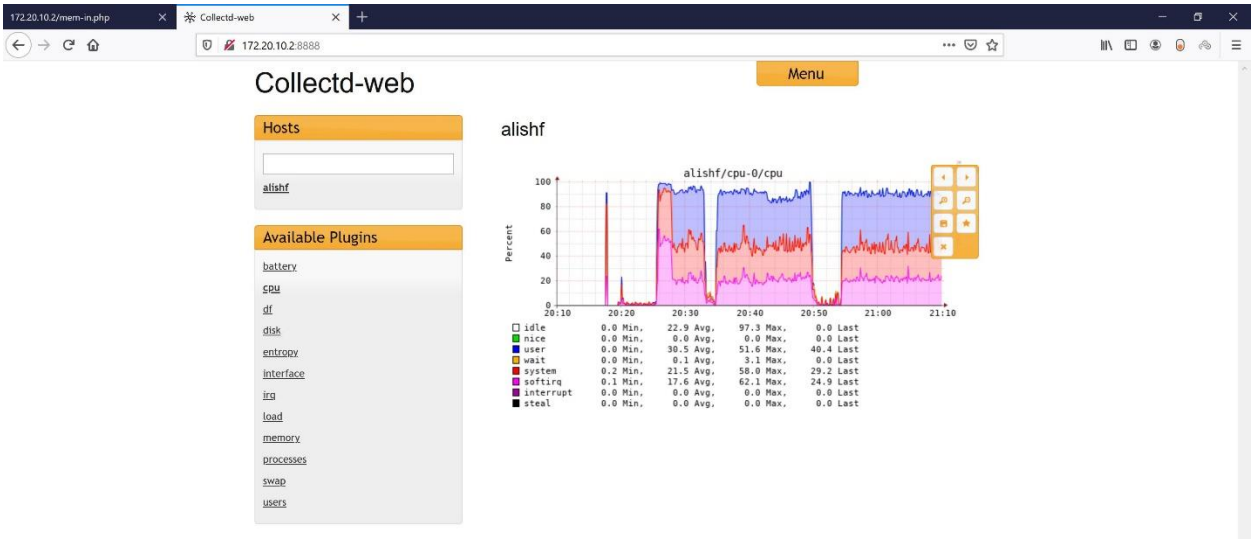
آدرس IP ماشین‌های مجازی:

Webserver1: 172.20.10.2
Webserver2: 172.20.10.5
Webserver3: 172.20.10.7
Workload generator: 172.20.10.6
Load balancer: 172.20.10.3

۱-۲: ایجاد webserver1 و نصب صحیح apache webserver:

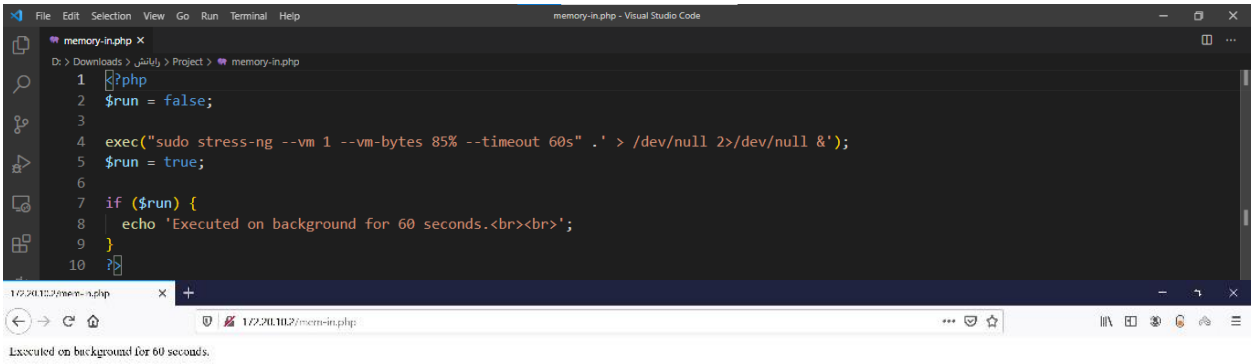


نصب collectd و نمایش میزان مصرف cpu و memory روی webserver1:

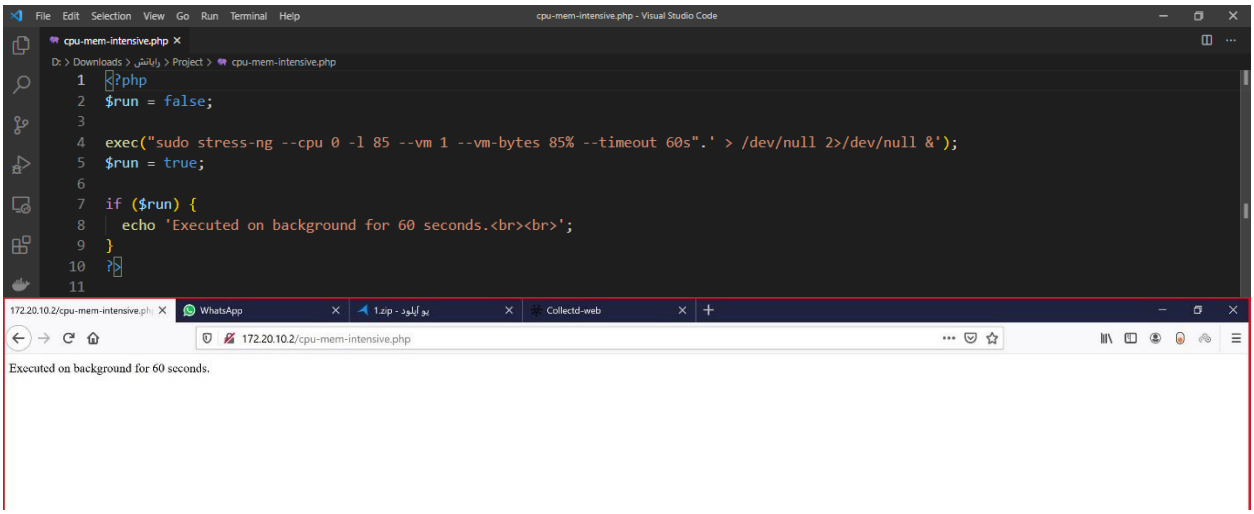


۲-۲: فایل cpu-intensive:

فایل memory-intensive:



فایل cpu-memory-intensive:



۳-۲:

ارسال درخواست از workload generator به webserver1 با hey:

بدین منظور دستور زیر را در workload generator وارد کردیم:

Hey -n 100000 -c 100 -q 30 {webserver1_url}

که n تعداد کل درخواست فرستاده شده به webserver1، c تعداد worker هایی که همزمان کار می‌کنند و q محدودیت ارسال درخواست هر worker است(در ثانیه)

برای فایل cpu-intensive:

```
  GNU nano 4.8                                  fwebcpu30.txt

Summary:
  Total:        1107.2074 secs
  Slowest:      393.5061 secs
  Fastest:      0.0013 secs
  Average:      1.0665 secs
  Requests/sec: 90.3173

  Total data:   4600000 bytes
  Size/request: 46 bytes

Response time histogram:
  0.001 [1]     |
  39.352 [99978]        |■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
  78.702 [15]   |
  118.053 [1]   |
  157.403 [1]   |
  196.754 [1]   |
  236.104 [0]   |
  275.455 [0]   |
  314.805 [0]   |
  354.156 [0]   |
  393.506 [3]   |


Latency distribution:
  10% in 0.1678 secs
  25% in 0.5946 secs
  50% in 0.8830 secs
  75% in 1.3144 secs
  90% in 1.9066 secs
  95% in 2.3962 secs
  99% in 3.6101 secs


alishf@alishf:~$ hey -n 100000 -c 100 -q 30 -t 0 http://172.20.10.2/cpu-in.php > fwebcpu30.txt
```

برای فایل memory-intensive:

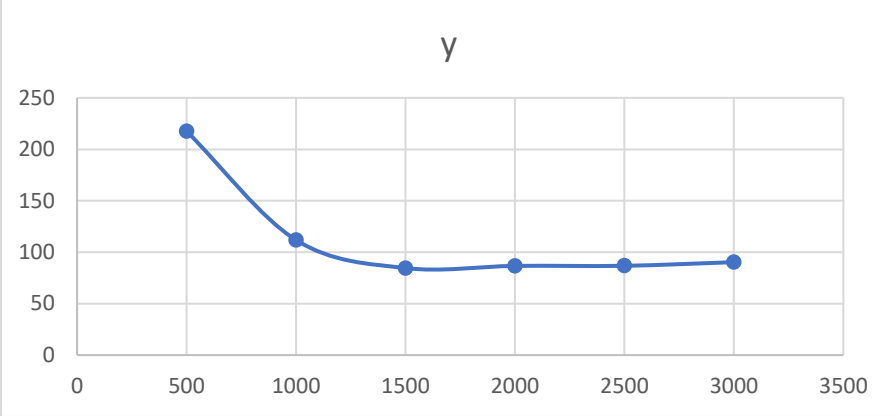
```
  GNU nano 4.8                               fwebmem30.txt

Summary:
  Total:        1392.1307 secs
  Slowest:      33.1967 secs
  Fastest:      0.0014 secs
  Average:      1.3479 secs
  Requests/sec: 71.8323

  Total data:   4600000 bytes
  Size/request: 46 bytes

Response time histogram:
  0.001 [1]      |
  3.321 [95598]  |■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
  6.640 [4227]   |■■
  9.960 [129]    |
  13.280 [18]    |
  16.599 [0]     |
  19.919 [0]     |
  23.238 [0]     |
  26.558 [6]     |
  29.877 [10]    |
  33.197 [11]    |


Latency distribution:
  10% in 0.3569 secs
  25% in 0.7063 secs
  50% in 1.1241 secs
  75% in 1.7670 secs
  90% in 2.5771 secs
  95% in 3.2086 secs
  99% in 4.6759 secs
                                         [ Read 46 lines ]
^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo
```

برای فایل cpu-memory-intensive:

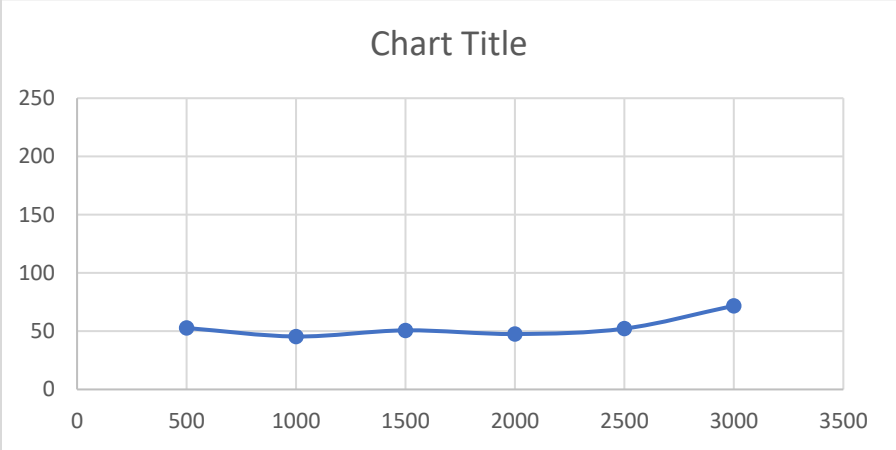
```
Summary:
  Total:        623.3659 secs
  Slowest:      14.9283 secs
  Fastest:      0.0010 secs
  Average:      0.6024 secs
  Requests/sec: 160.4194

  Total data:   4800000 bytes
  Size/request: 48 bytes

Response time histogram:
  0.001 [1]      |
  1.494 [95317]  |■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
  2.986 [4585]   |■■
  4.479 [22]     |
  5.972 [16]     |
  7.465 [17]     |
  8.957 [14]     |
  10.450 [0]     |
  11.943 [11]    |
  13.436 [12]    |
  14.928 [5]     |


Latency distribution:
  10% in 0.0061 secs
  25% in 0.2816 secs
  50% in 0.5627 secs
  75% in 0.7992 secs
  90% in 1.1478 secs
  95% in 1.4629 secs
  99% in 2.0693 secs


alishf@alishf:~$ hey -n 100000 -c 100 -q 30 -t 0 http://172.20.10.2/cpu-mem-intensive.php > 100cpume
m30.txt_
```

۴-۲: در این قسمت محورx مقدار request/second ایده آل را که توسط hey داده‌شده است(حاصل q*c) و محور y مقدار request/second واقعی را که در عکس‌های بالا نیز قابل مشاهده است نشان می‌دهد.

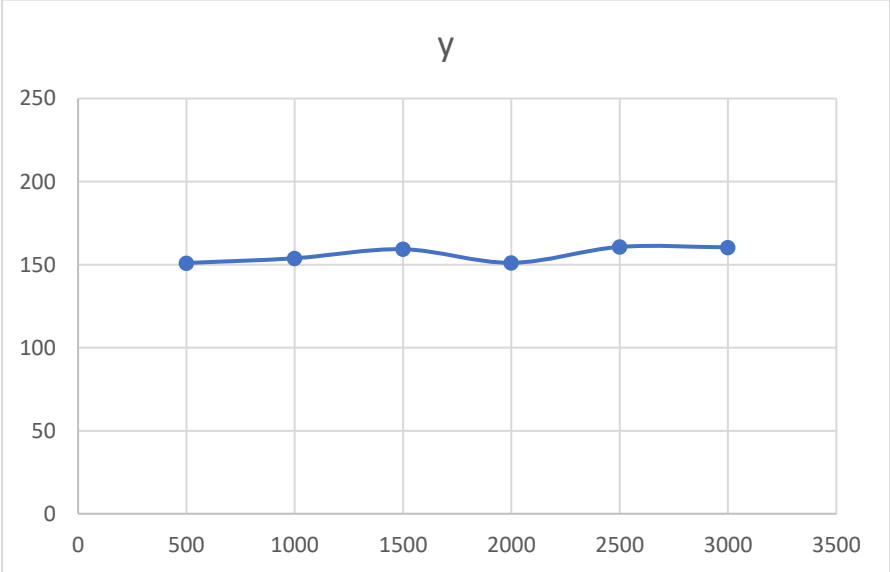
نمودار فایل cpu-intensive:



نمودار فایل memory-intensive:



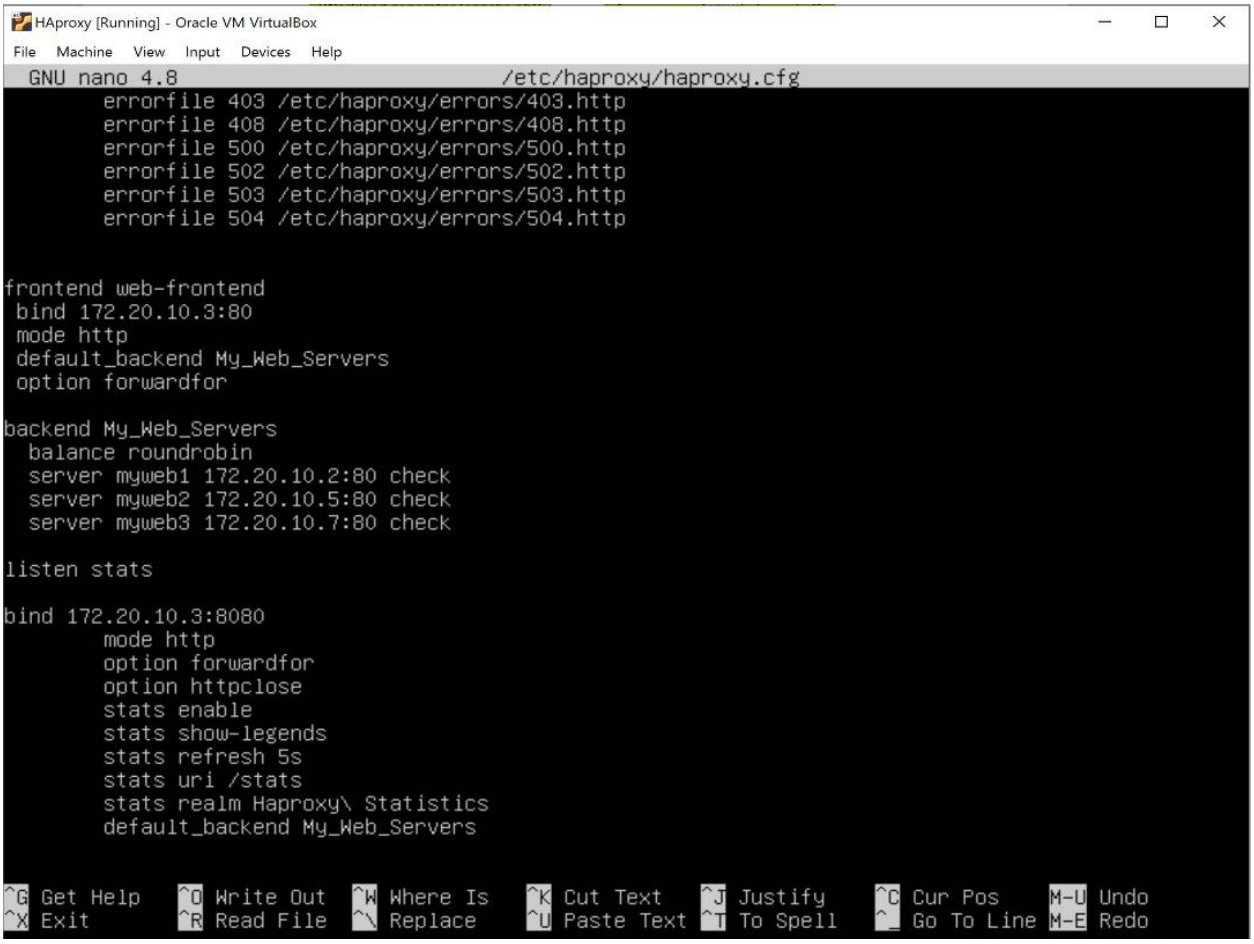
نمودار فایل cpu-memory-intensive:

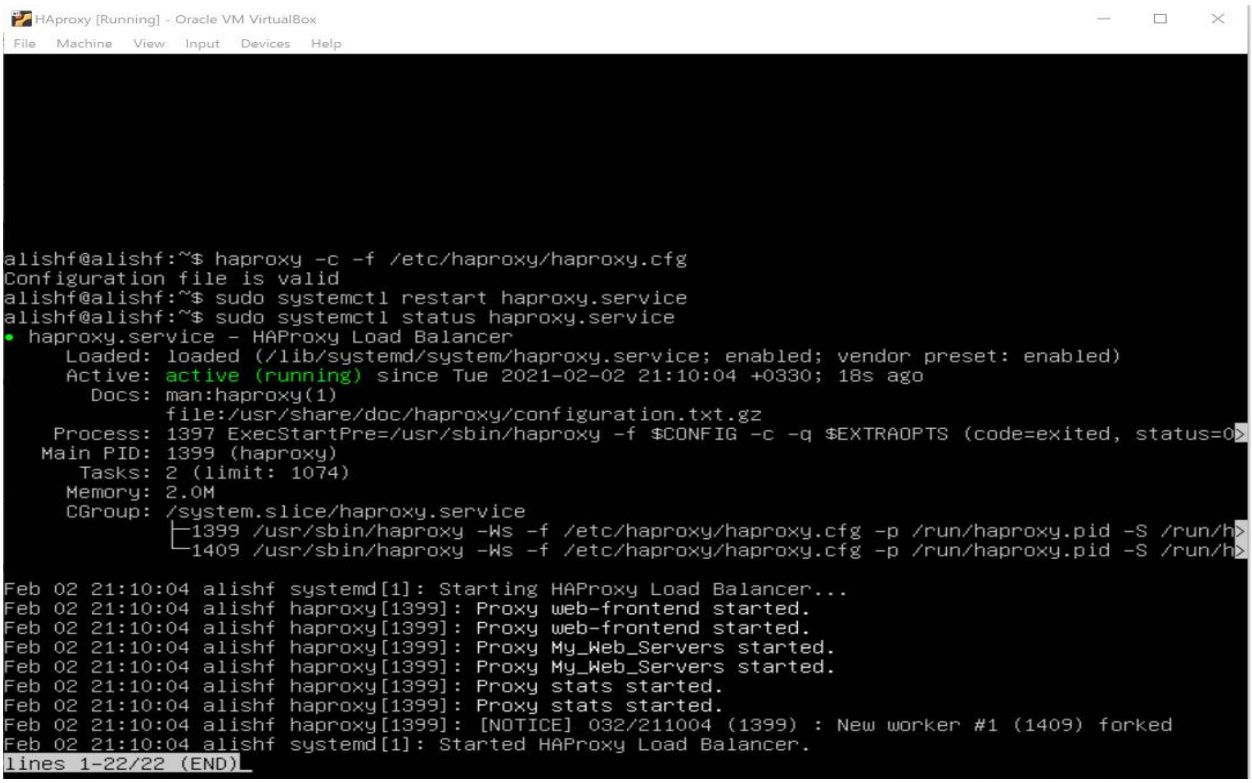


با توجه به نمودارهای این قسمت، تاثیر افزایش q روی فایل cpu-intensive به صورت نزولی و برای فایل های memory-intensive و cpu-memory-intensive تقریبا بی‌تاثیر است بدین معنی که بالا رفتن q هیچ تاثیری در دریافت بار روی memory ندارد ولی هرچه q بیشتر شود پردازش درخواست از طرف cpu کمتر می‌شود.

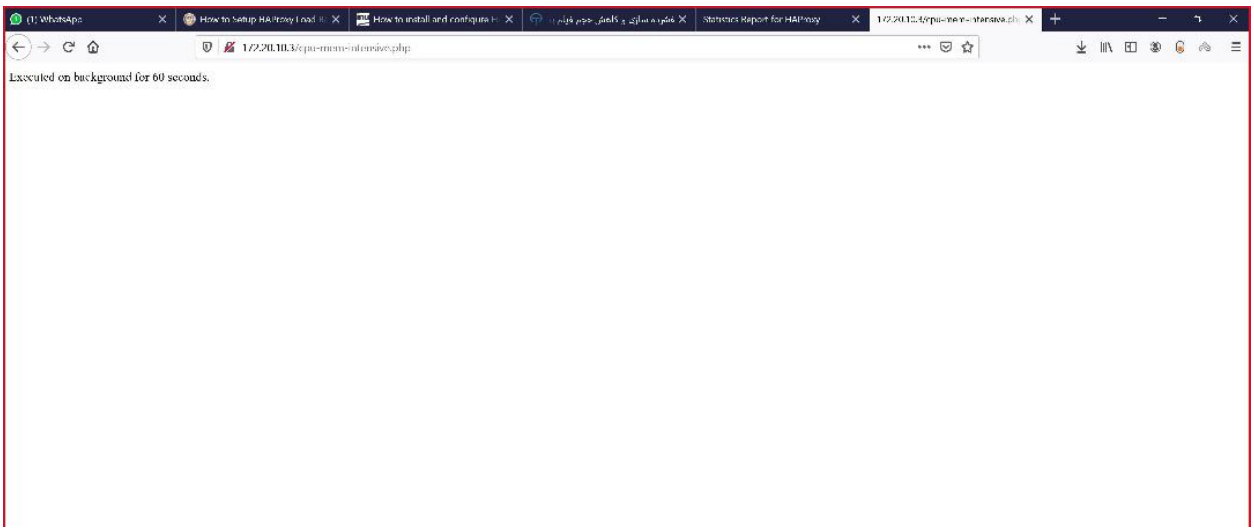
۵-۲:

پیکربندی HAproxy:





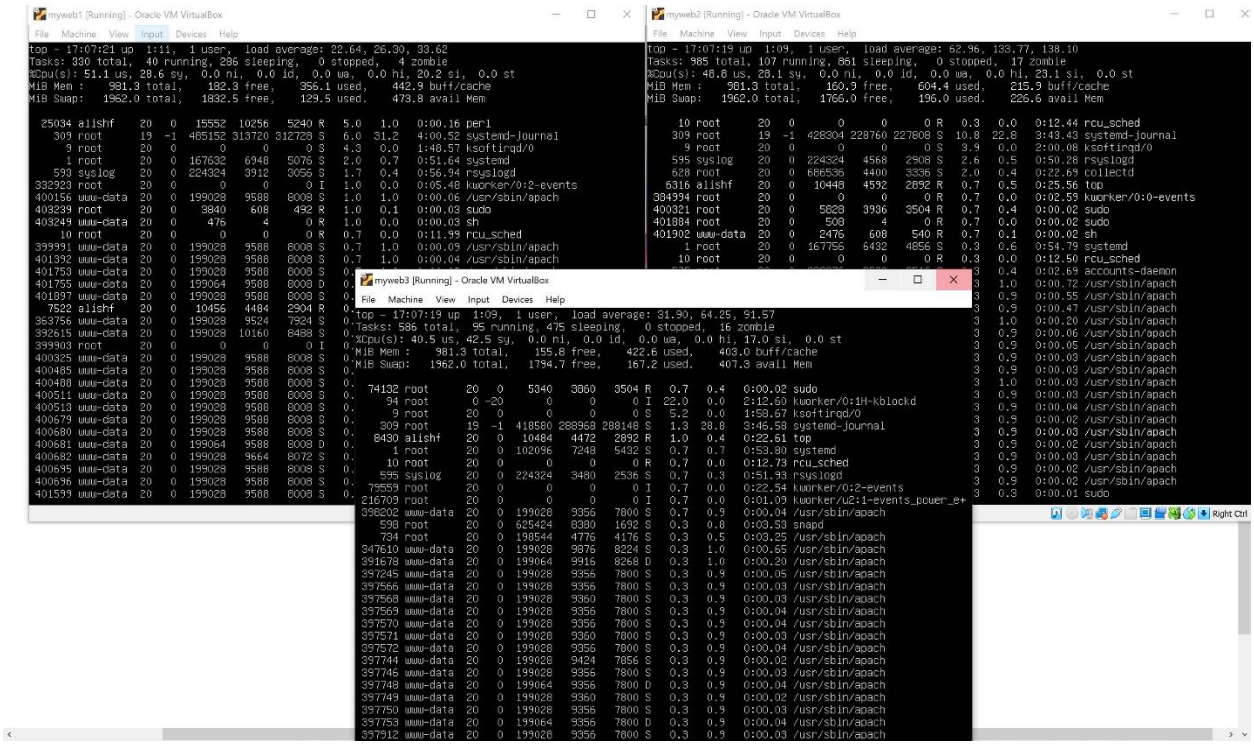
دسترسی به فایل‌ها از طریق آدرس load balancer در مرورگر:



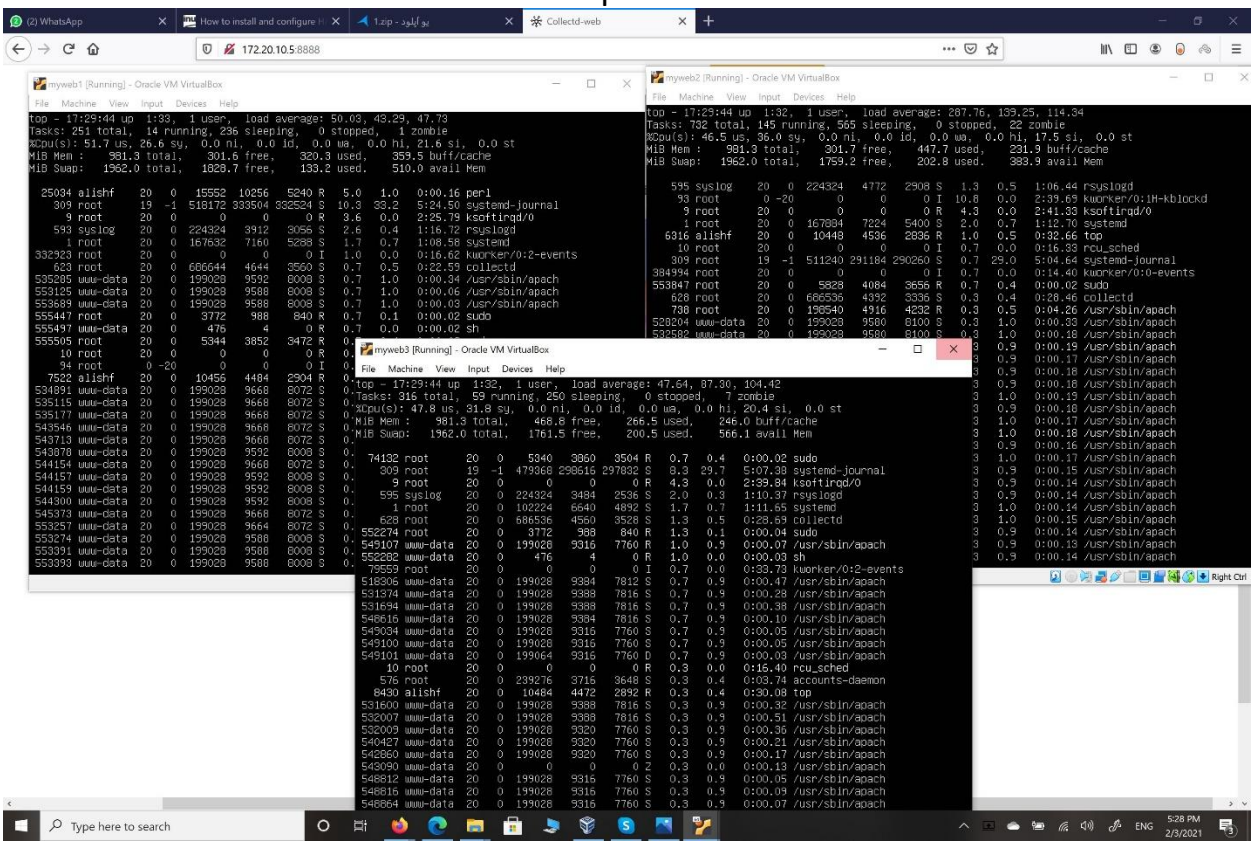
ارسال درخواست برای فایل‌ها و مشاهده log:

cpu-intensive:

q=5

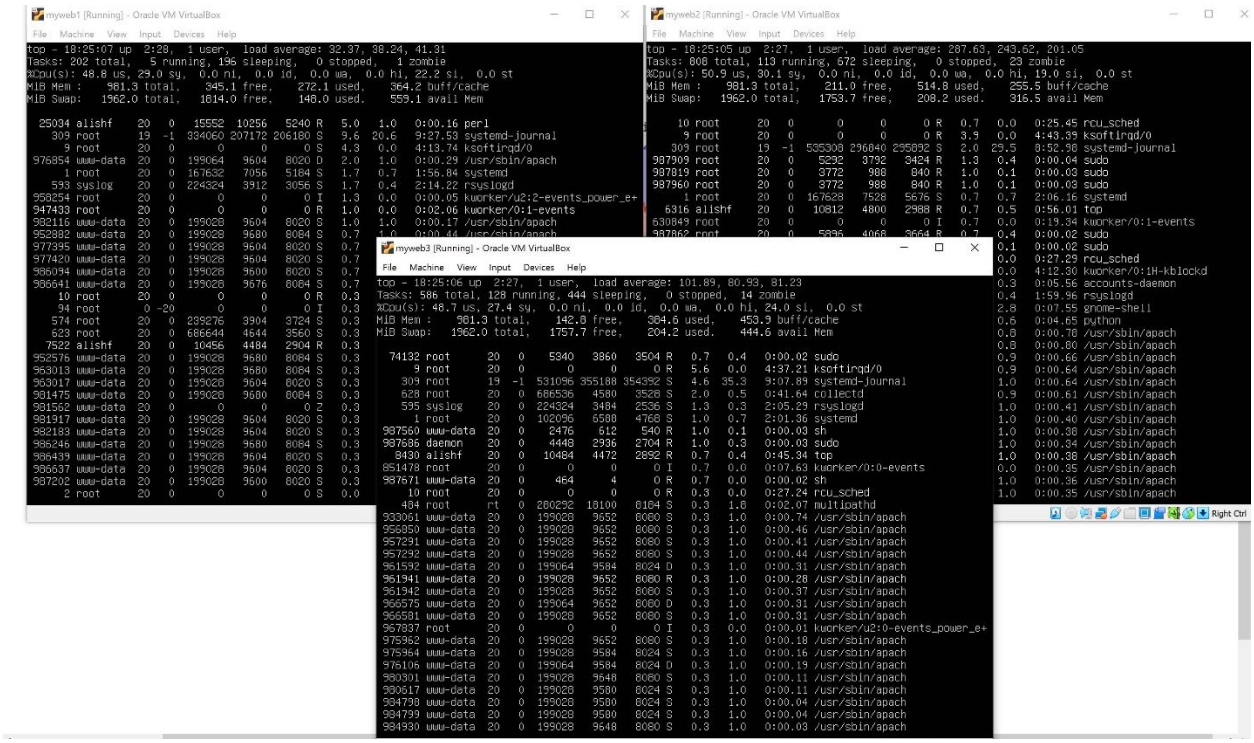


q=20

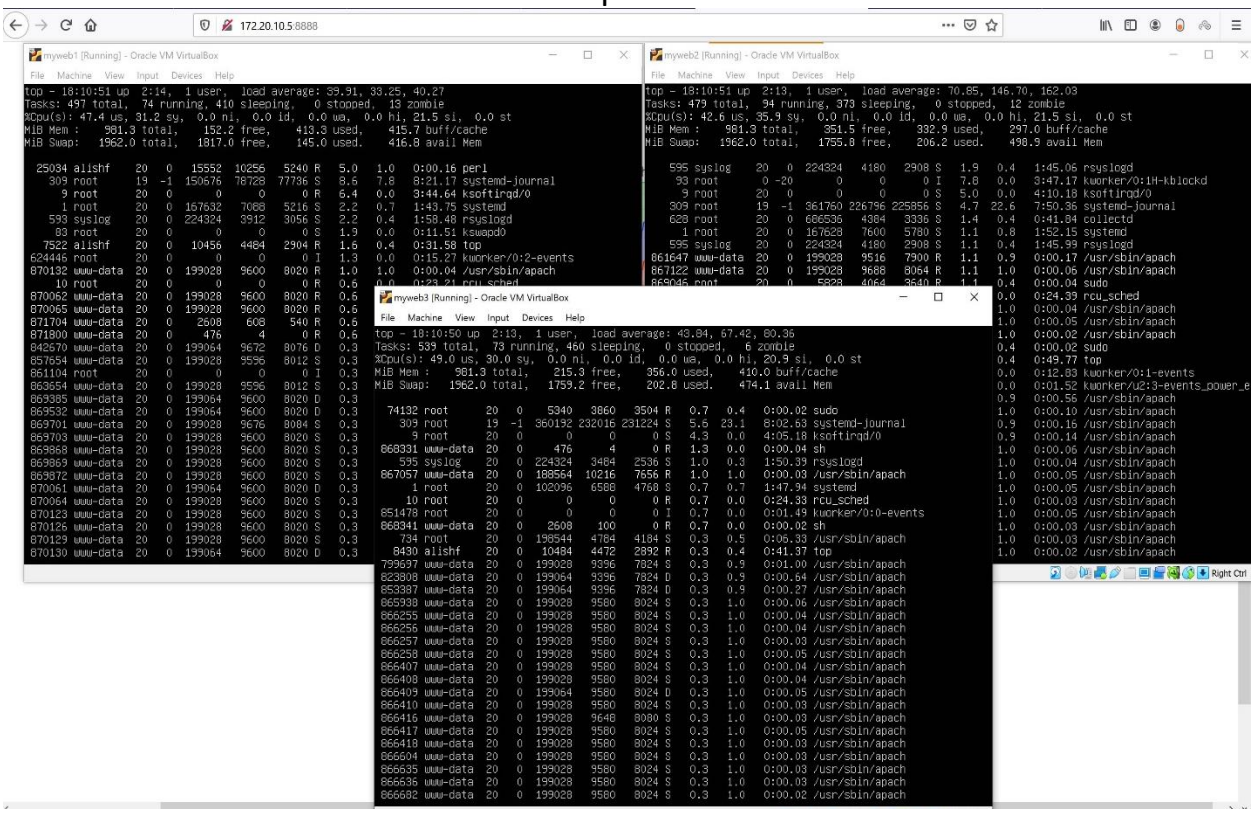


:memory-intensive

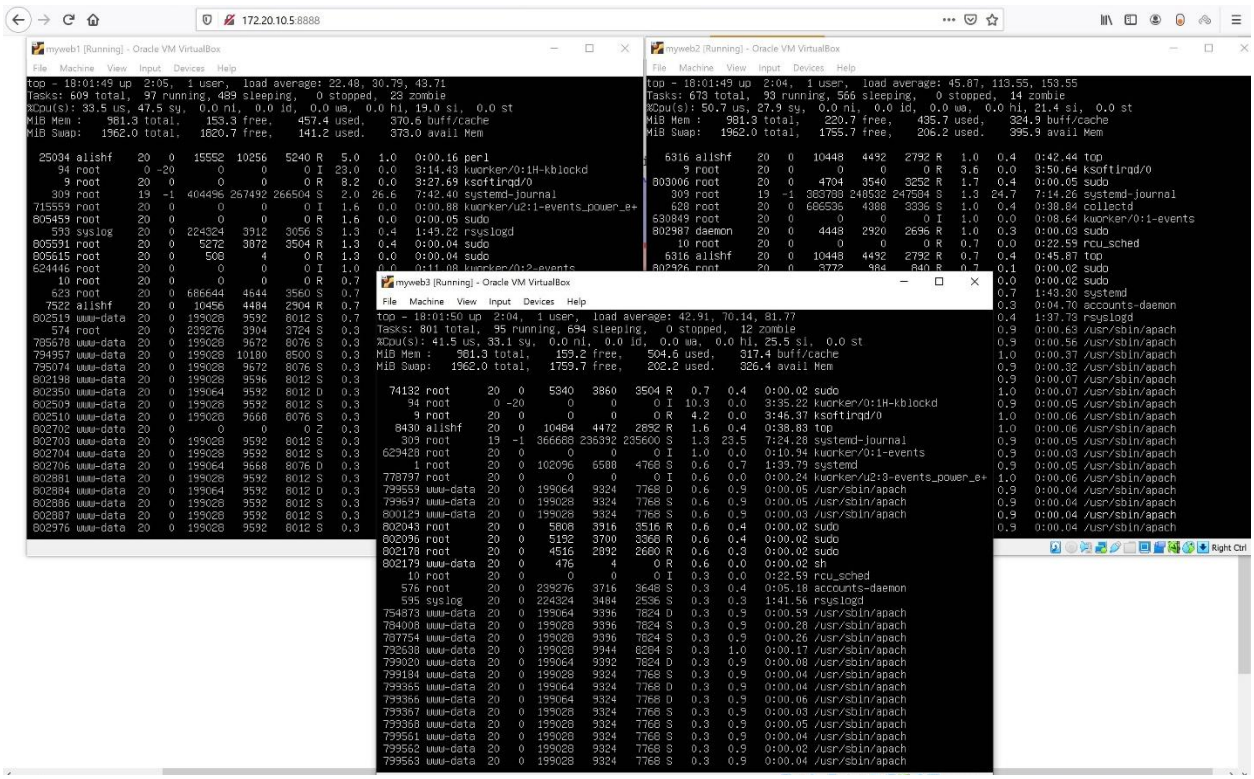
q=20


myweb1 [Running] - Oracle VM VirtualBox
myweb2 [Running] - Oracle VM VirtualBox
myweb3 [Running] - Oracle VM VirtualBox

q=25

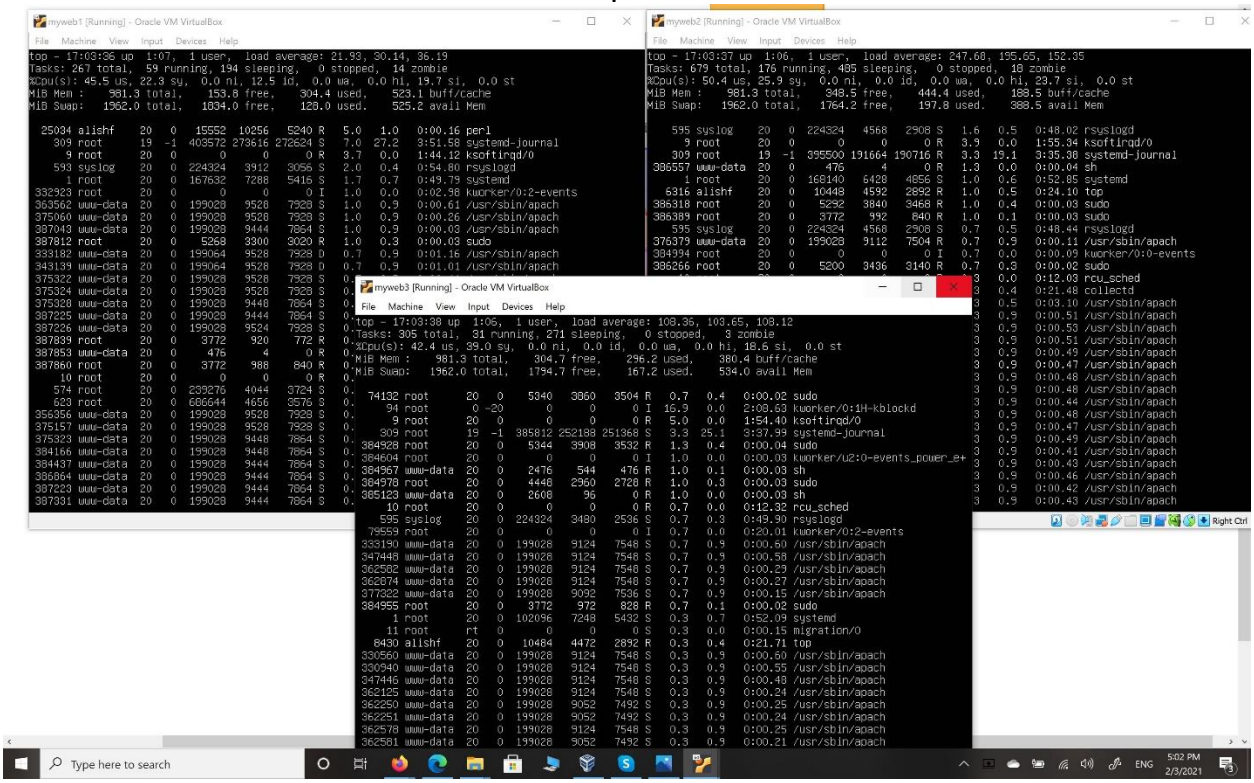

172.20.10.5:8888
myweb1 [Running] - Oracle VM VirtualBox
myweb2 [Running] - Oracle VM VirtualBox
myweb3 [Running] - Oracle VM VirtualBox

q=30

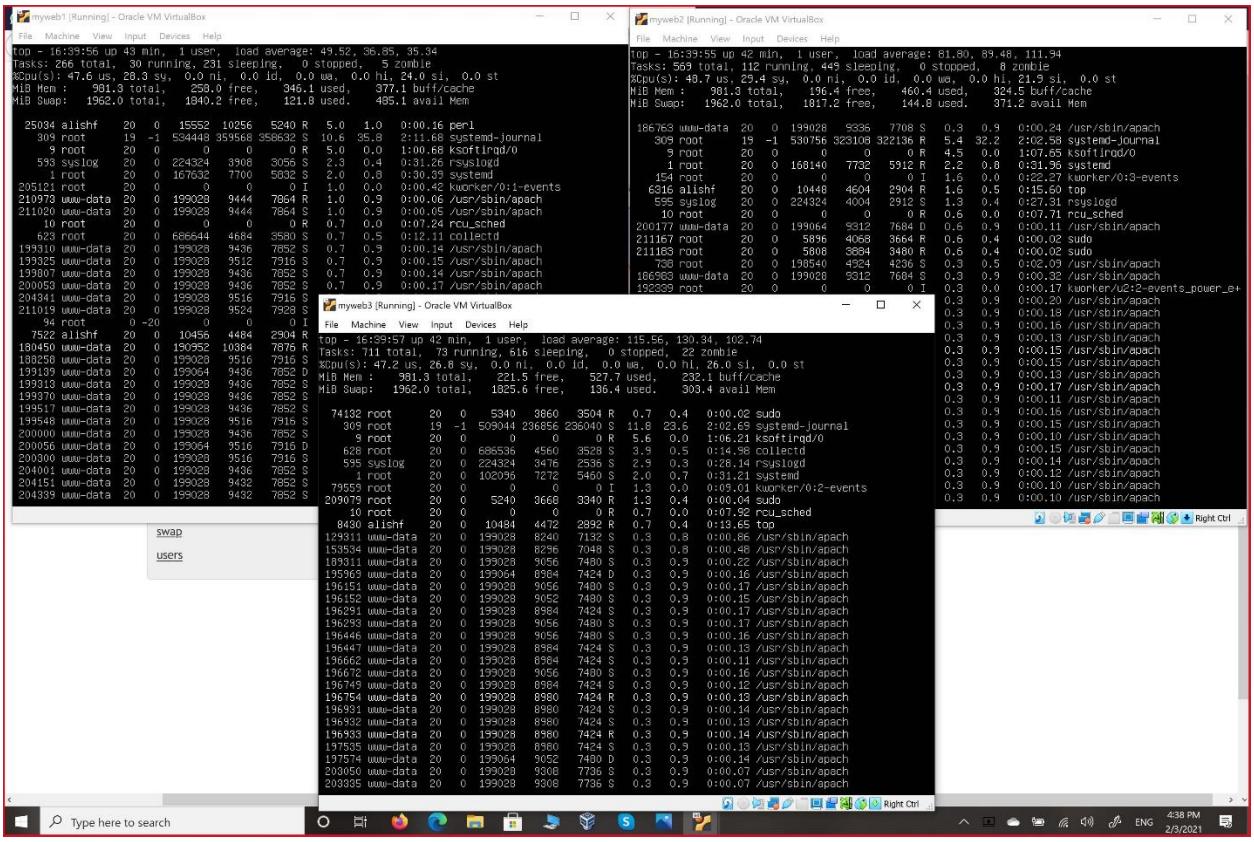


:cpu-memory-intensive

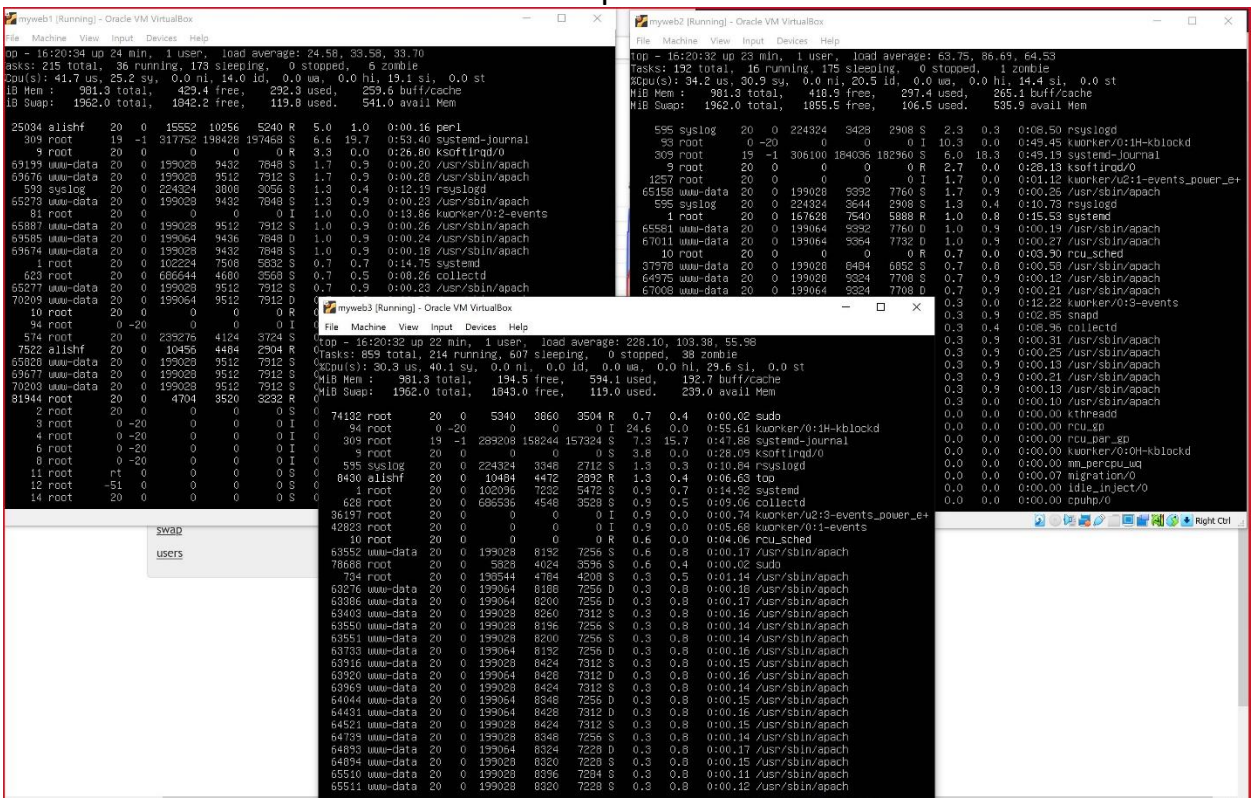
q=5



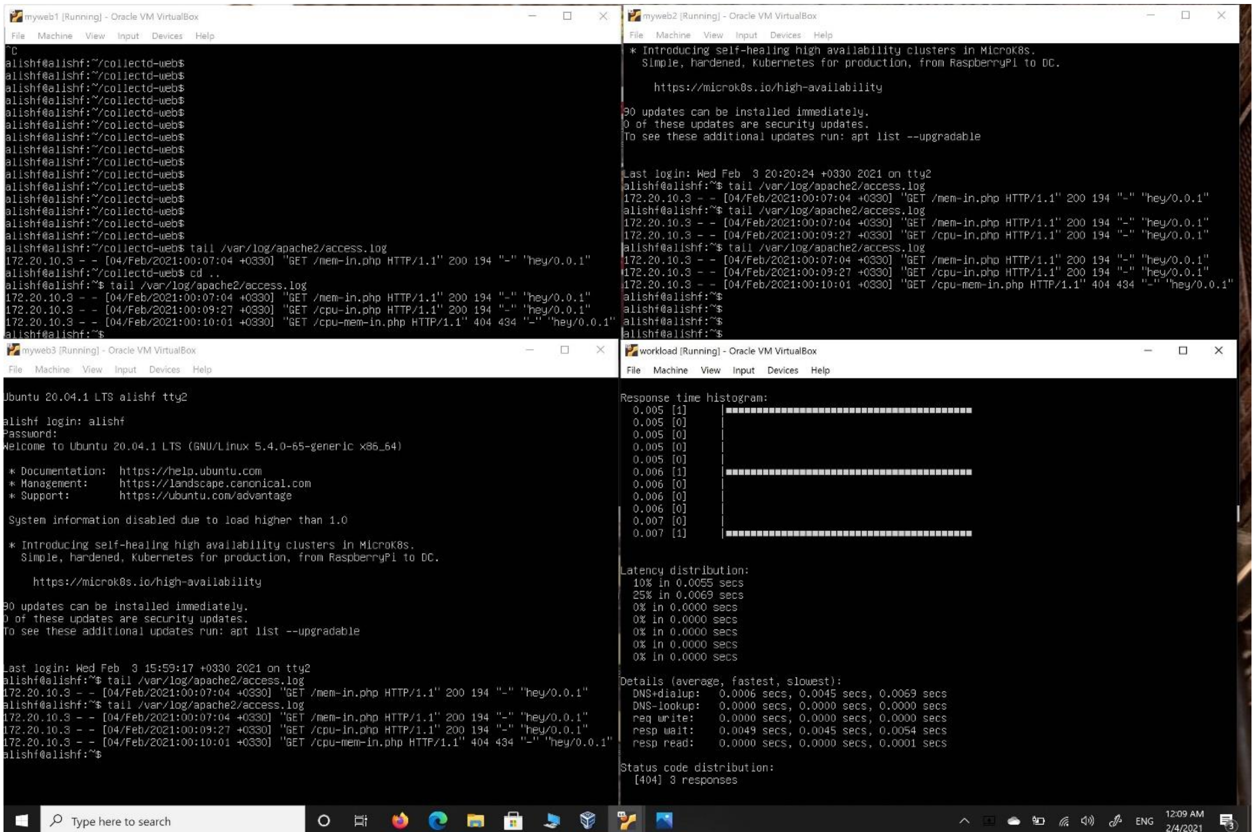
q=15

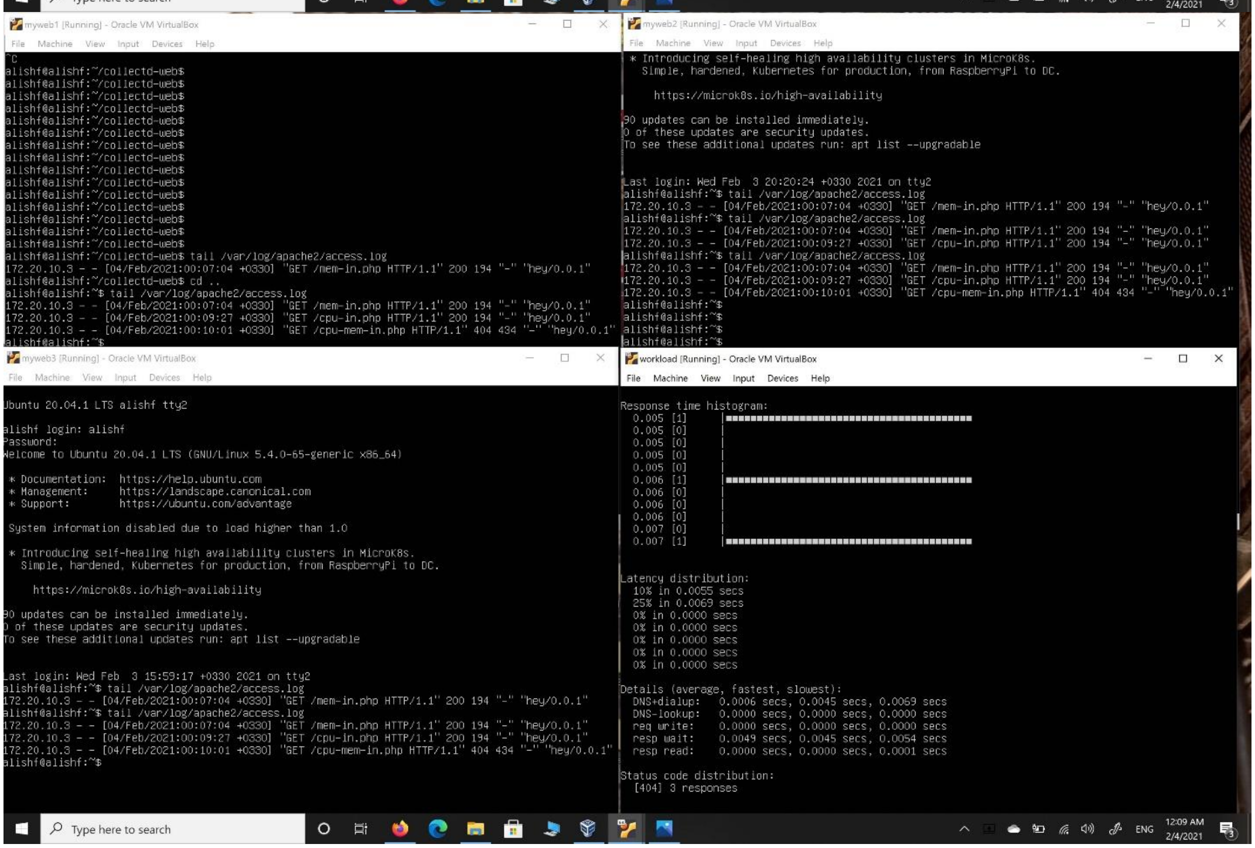


q=25

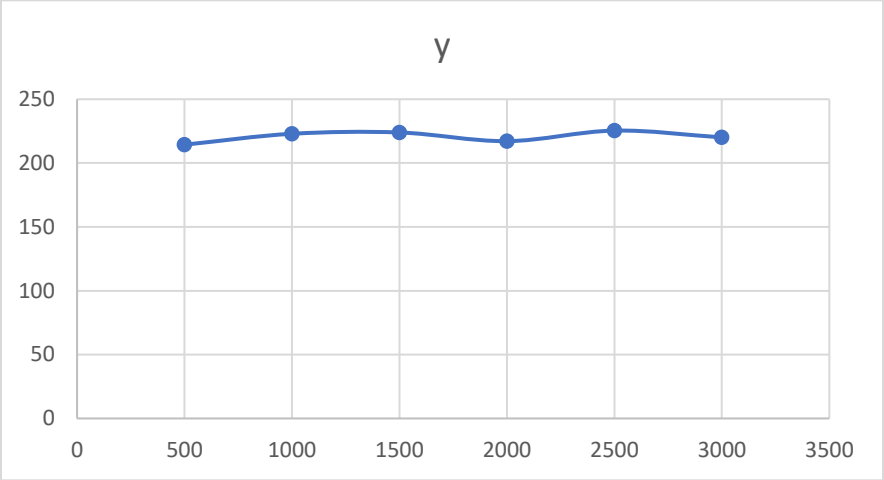


فایل log:

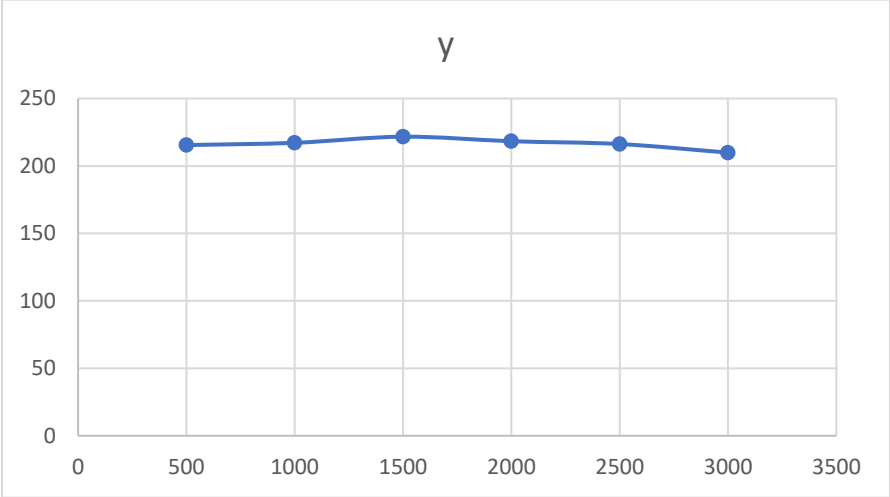

myweb1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
^C
alishf@alishf:~/collectd-web$
alishf@alishf:~/collectd-web$ tail /var/log/apache2/access.log
172.20.10.3 - - [04/Feb/2021:00:07:04 +0330] "GET /mem-in.php HTTP/1.1" 200 194 "-" "hey/0.0.1"
alishf@alishf:~/collectd-web$ cd ..
alishf@alishf:~$ tail /var/log/apache2/access.log
172.20.10.3 - - [04/Feb/2021:00:07:04 +0330] "GET /mem-in.php HTTP/1.1" 200 194 "-" "hey/0.0.1"
172.20.10.3 - - [04/Feb/2021:00:09:27 +0330] "GET /cpu-in.php HTTP/1.1" 200 194 "-" "hey/0.0.1"
172.20.10.3 - - [04/Feb/2021:00:10:01 +0330] "GET /cpu-mem-in.php HTTP/1.1" 404 434 "-" "hey/0.0.1"
myweb2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
* Introducing self-healing high availability clusters in MicroK8s.
Simple, hardened, Kubernetes for production, from RaspberryPi to DC.
https://microk8s.io/high-availability
90 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable
Last login: Wed Feb 3 20:20:24 +0330 2021 on tty2
alishf@alishf:~$ tail /var/log/apache2/access.log
172.20.10.3 - - [04/Feb/2021:00:07:04 +0330] "GET /mem-in.php HTTP/1.1" 200 194 "-" "hey/0.0.1"
alishf@alishf:~$ tail /var/log/apache2/access.log
172.20.10.3 - - [04/Feb/2021:00:07:04 +0330] "GET /mem-in.php HTTP/1.1" 200 194 "-" "hey/0.0.1"
172.20.10.3 - - [04/Feb/2021:00:09:27 +0330] "GET /cpu-in.php HTTP/1.1" 200 194 "-" "hey/0.0.1"
alishf@alishf:~$ tail /var/log/apache2/access.log
172.20.10.3 - - [04/Feb/2021:00:07:04 +0330] "GET /mem-in.php HTTP/1.1" 200 194 "-" "hey/0.0.1"
172.20.10.3 - - [04/Feb/2021:00:09:27 +0330] "GET /cpu-in.php HTTP/1.1" 200 194 "-" "hey/0.0.1"
172.20.10.3 - - [04/Feb/2021:00:10:01 +0330] "GET /cpu-mem-in.php HTTP/1.1" 404 434 "-" "hey/0.0.1"
myweb3 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Ubuntu 20.04.1 LTS alishf tty2
alishf login: alishf
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-65-generic x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
System information disabled due to load higher than 1.0
* Introducing self-healing high availability clusters in MicroK8s.
Simple, hardened, Kubernetes for production, from RaspberryPi to DC.
https://microk8s.io/high-availability
90 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable
Last login: Wed Feb 3 15:59:17 +0330 2021 on tty2
alishf@alishf:~$ tail /var/log/apache2/access.log
172.20.10.3 - - [04/Feb/2021:00:07:04 +0330] "GET /mem-in.php HTTP/1.1" 200 194 "-" "hey/0.0.1"
alishf@alishf:~$ tail /var/log/apache2/access.log
172.20.10.3 - - [04/Feb/2021:00:07:04 +0330] "GET /mem-in.php HTTP/1.1" 200 194 "-" "hey/0.0.1"
172.20.10.3 - - [04/Feb/2021:00:09:27 +0330] "GET /cpu-in.php HTTP/1.1" 200 194 "-" "hey/0.0.1"
172.20.10.3 - - [04/Feb/2021:00:10:01 +0330] "GET /cpu-mem-in.php HTTP/1.1" 404 434 "-" "hey/0.0.1"
alishf@alishf:~$
workload [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Response time histogram:
0.005 [1]
0.005 [0]
0.005 [0]
0.005 [0]
0.005 [0]
0.006 [1]
0.006 [0]
0.006 [0]
0.006 [0]
0.007 [0]
0.007 [1]
Latency distribution:
10% in 0.0055 secs
25% in 0.0069 secs
0% in 0.0000 secs
0% in 0.0000 secs
0% in 0.0000 secs
0% in 0.0000 secs
0% in 0.0000 secs
Details (average, fastest, slowest):
DNS+dialup: 0.0006 secs, 0.0045 secs, 0.0069 secs
DNS-lookup: 0.0000 secs, 0.0000 secs, 0.0000 secs
req write: 0.0000 secs, 0.0000 secs, 0.0000 secs
resp wait: 0.0049 secs, 0.0045 secs, 0.0054 secs
resp read: 0.0000 secs, 0.0000 secs, 0.0001 secs
Status code distribution:
[404] 3 responses
Type here to search
ENG
12:09 AM
2/4/2021


myweb1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
^C
alishf@alishf:~/collectd-web$
alishf@alishf:~/collectd-web$ tail /var/log/apache2/access.log
172.20.10.3 - - [04/Feb/2021:00:07:04 +0330] "GET /mem-in.php HTTP/1.1" 200 194 "-" "hey/0.0.1"
alishf@alishf:~/collectd-web$ cd ..
alishf@alishf:~$ tail /var/log/apache2/access.log
172.20.10.3 - - [04/Feb/2021:00:07:04 +0330] "GET /mem-in.php HTTP/1.1" 200 194 "-" "hey/0.0.1"
172.20.10.3 - - [04/Feb/2021:00:09:27 +0330] "GET /cpu-in.php HTTP/1.1" 200 194 "-" "hey/0.0.1"
172.20.10.3 - - [04/Feb/2021:00:10:01 +0330] "GET /cpu-mem-in.php HTTP/1.1" 404 434 "-" "hey/0.0.1"
myweb2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
* Introducing self-healing high availability clusters in MicroK8s.
Simple, hardened, Kubernetes for production, from RaspberryPi to DC.
https://microk8s.io/high-availability
90 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable
Last login: Wed Feb 3 20:20:24 +0330 2021 on tty2
alishf@alishf:~$ tail /var/log/apache2/access.log
172.20.10.3 - - [04/Feb/2021:00:07:04 +0330] "GET /mem-in.php HTTP/1.1" 200 194 "-" "hey/0.0.1"
alishf@alishf:~$ tail /var/log/apache2/access.log
172.20.10.3 - - [04/Feb/2021:00:07:04 +0330] "GET /mem-in.php HTTP/1.1" 200 194 "-" "hey/0.0.1"
172.20.10.3 - - [04/Feb/2021:00:09:27 +0330] "GET /cpu-in.php HTTP/1.1" 200 194 "-" "hey/0.0.1"
alishf@alishf:~$ tail /var/log/apache2/access.log
172.20.10.3 - - [04/Feb/2021:00:07:04 +0330] "GET /mem-in.php HTTP/1.1" 200 194 "-" "hey/0.0.1"
172.20.10.3 - - [04/Feb/2021:00:09:27 +0330] "GET /cpu-in.php HTTP/1.1" 200 194 "-" "hey/0.0.1"
172.20.10.3 - - [04/Feb/2021:00:10:01 +0330] "GET /cpu-mem-in.php HTTP/1.1" 404 434 "-" "hey/0.0.1"
myweb3 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Ubuntu 20.04.1 LTS alishf tty2
alishf login: alishf
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-65-generic x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
System information disabled due to load higher than 1.0
* Introducing self-healing high availability clusters in MicroK8s.
Simple, hardened, Kubernetes for production, from RaspberryPi to DC.
https://microk8s.io/high-availability
90 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable
Last login: Wed Feb 3 15:59:17 +0330 2021 on tty2
alishf@alishf:~$ tail /var/log/apache2/access.log
172.20.10.3 - - [04/Feb/2021:00:07:04 +0330] "GET /mem-in.php HTTP/1.1" 200 194 "-" "hey/0.0.1"
alishf@alishf:~$ tail /var/log/apache2/access.log
172.20.10.3 - - [04/Feb/2021:00:07:04 +0330] "GET /mem-in.php HTTP/1.1" 200 194 "-" "hey/0.0.1"
172.20.10.3 - - [04/Feb/2021:00:09:27 +0330] "GET /cpu-in.php HTTP/1.1" 200 194 "-" "hey/0.0.1"
172.20.10.3 - - [04/Feb/2021:00:10:01 +0330] "GET /cpu-mem-in.php HTTP/1.1" 404 434 "-" "hey/0.0.1"
alishf@alishf:~$
workload [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Response time histogram:
0.005 [1]
0.005 [0]
0.005 [0]
0.005 [0]
0.005 [0]
0.006 [1]
0.006 [0]
0.006 [0]
0.006 [0]
0.007 [0]
0.007 [1]
Latency distribution:
10% in 0.0055 secs
25% in 0.0069 secs
0% in 0.0000 secs
0% in 0.0000 secs
0% in 0.0000 secs
0% in 0.0000 secs
0% in 0.0000 secs
Details (average, fastest, slowest):
DNS+dialup: 0.0006 secs, 0.0045 secs, 0.0069 secs
DNS-lookup: 0.0000 secs, 0.0000 secs, 0.0000 secs
req write: 0.0000 secs, 0.0000 secs, 0.0000 secs
resp wait: 0.0049 secs, 0.0045 secs, 0.0054 secs
resp read: 0.0000 secs, 0.0000 secs, 0.0001 secs
Status code distribution:
[404] 3 responses
Type here to search
ENG
12:09 AM
2/4/2021

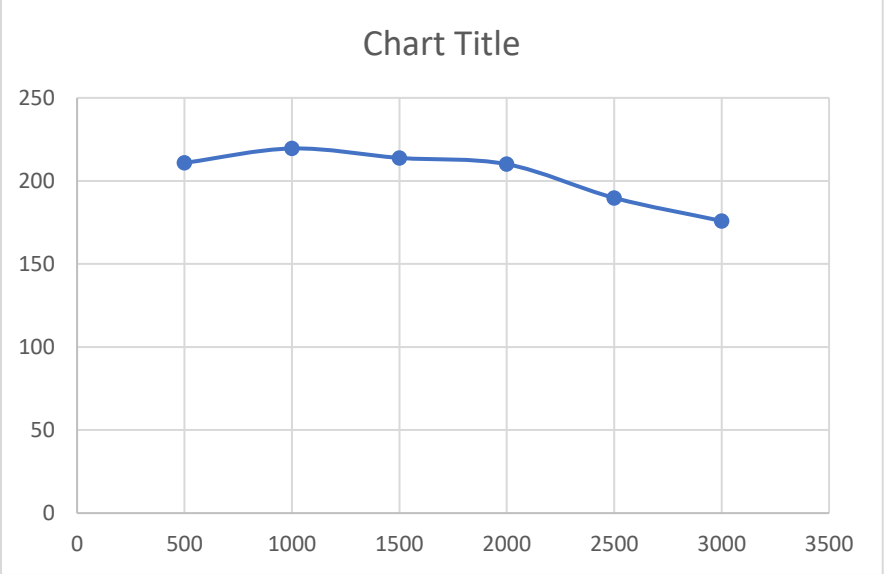
۶-۲: نمودار فایل cpu-intensive:



نمودار فایل memory-intensive:



نمودار فایل cpu-memory-intensive:



در مقایسه با قسمت ۴-۲ متوجه می‌شویم مقدار request/second به طور محسوسی افزایش یافته است زیرا load balancer درخواست‌ها را به طور متعادل بین webserver ها پخش می‌کند و بار روی webserver کاهش می‌یابد در نتیجه هر webserver تعداد درخواست بیشتری را می‌تواند پردازش کند.

۷-۲: در این بخش دو کد balancer.py و controller.py استفاده کردیم که اولی روی webserver ها اجرا می‌شود و اطلاعات cpu و memory در حال استفاده را ذخیره می‌کند و دومی روی سیستم‌عامل اجرا شده و با توجه به اطلاعات دریافتی از balancer شروط مشخص را اجرا می‌کند.

منابع:

[1] https://www.digitalocean.com/community/tutorials/an-introduction-to-haproxy-and-load-balancing-concepts

[2] https://en.wikipedia.org/wiki/Load_balancing_(computing)

[3] https://blog.stackpath.com/load-balancing-haproxy/#:~:text=HAProxy%20Algorithms&text=It%20works%20by%20using%20each,be%20adjusted%20on%20the%20go.

[4] https://linuxways.net/ubuntu/how-to-install-and-configure-haproxy-load-balancer-in-linux/

[5] https://github.com/rakyll/hey