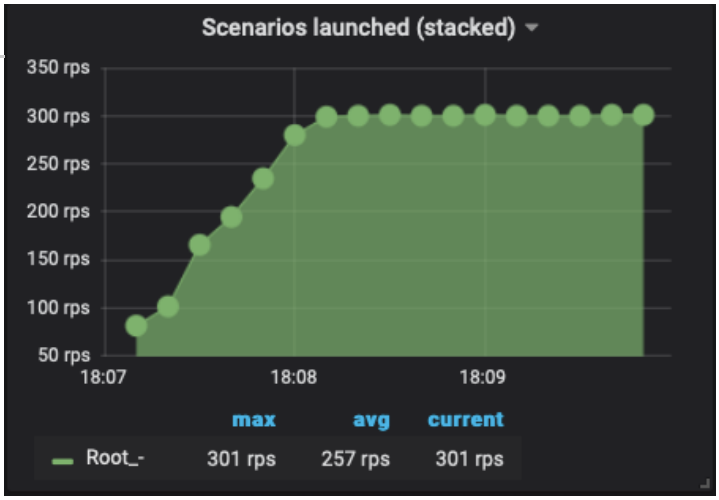
Informe de Perf Testing Arquitectura de Software

Comenzaremos este informe detallando las bases del proyecto llevado a cabo. En este trabajo practico nos planteamos el objetivo de analizar atributos de calidad de un sistema dummy el cual tiene unos pocos endpoints. A travez de herramientas como Artillery, Graphite, Grafana, y CAdvisor generaremos y recopilaremos informacion de forma tal que podamos llevar a cabo un analisis seguido de una reflexion final de las conclusiones obtenidas. Nuestro sistema dummy tendra dos servidores, uno en Node.js y otro en Python (el cual utilizara Flask y Gunicorn para funcionar como servidor).

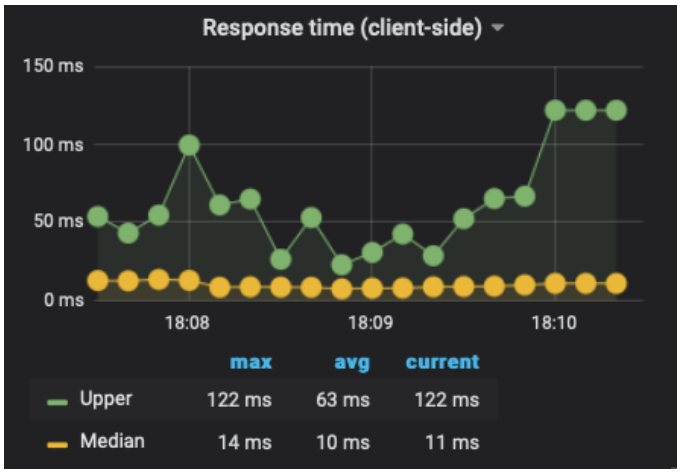
Load test

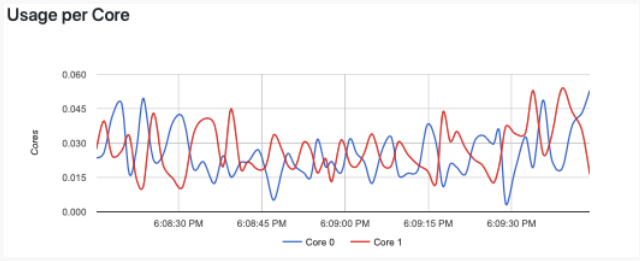
Para el caso del load test primero seguimos un esquema muy simple de "Phases". Primero habria una rampa de requests subiendo de 5 a 30 en 60 segundos para simular un comienzo estable y luevo mantuvimos la cantidad de requests por segundo en 30 los siguientes dos minutos. Este esquema se repitio en todos los otros casos de load test

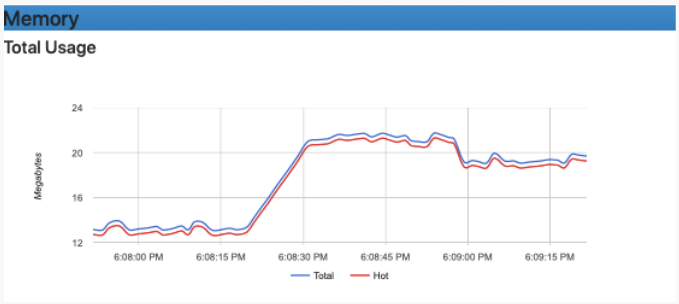


Caso Load Test Ping

En el caso del Ping hay dos particularidades importantes a notar, la cantidad de requests pendientes es bastante estable a travez de los periodos de 10 segundos. El otro hecho a notar muy importante es que el tiempo de respuesta tiene una varianza muy alta con respecto a los otros dos endpoints. En cuanto al uso de la memoria y el procesador no se ve algo muy extraño, el uso de los procesadores se mantiene bastante constante a travez de toda la experiencia y la memoria tambien

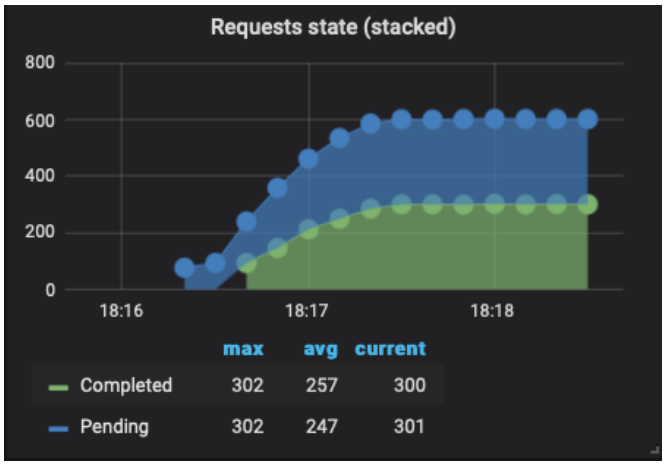


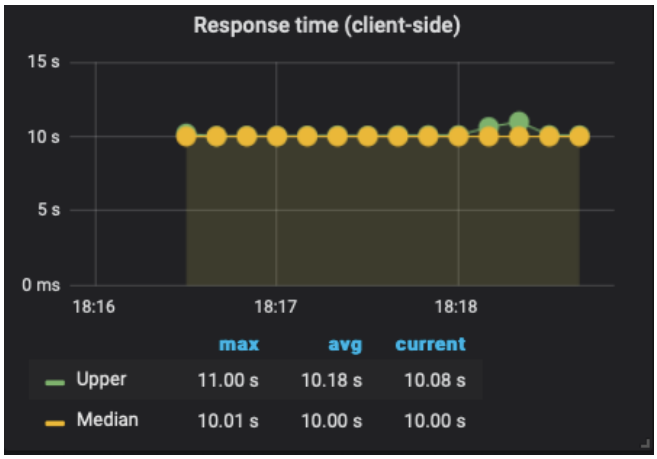


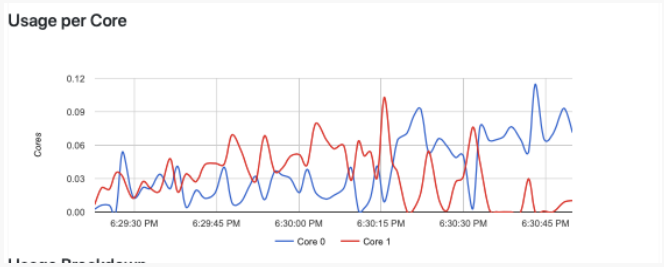


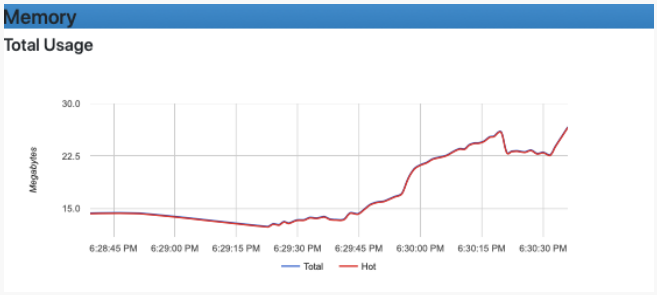
Caso Load Test Node Timeout

En el load test de Node timeout esta la particularidad muy importante a notar, esta es que el response time tiene una varianza muy pequeña, casi no hay diferencia entre los tiempos medios y los mas altos. La cantidad pending requests es constante, siempre se mantiene en 300 en la fase constante. En cuanto al uso de memoria se puede ver tambien como es mas gradual en este caso. Lo mas curioso de este endpoint es como el timeout parece ordenar las ejecuciones del servidor manteniendo un servidor con poca variablidad. Aunque en el caso del ping la variablidad no era extremadamente alta, quizas para un servicio que necesite respuestas en tiempo constante esta informacion es muy util



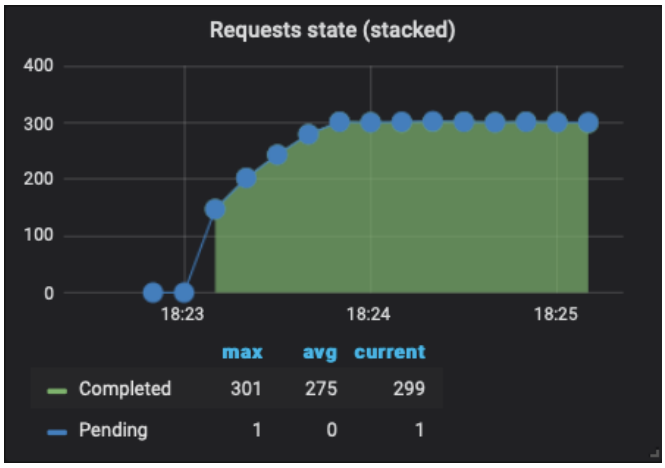


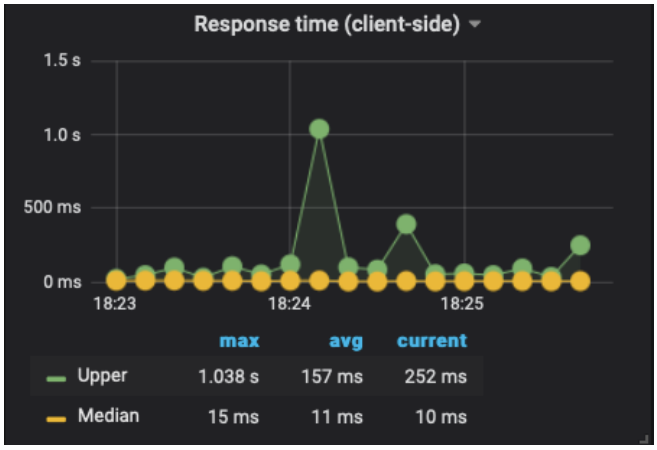


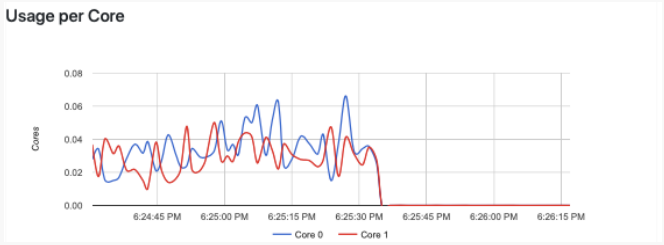


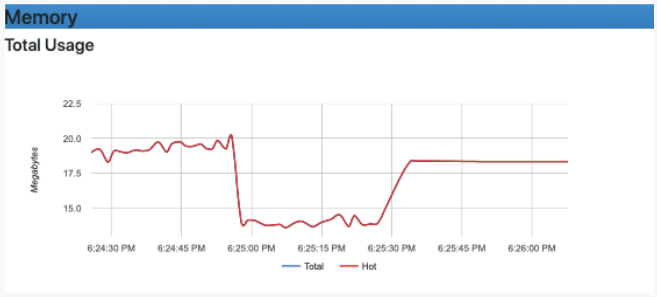
Caso Load Test Node Intensive

En el load test de Node intensive podemos ver que al igual que en el caso del Ping la cantidad de requests que estan pendientes siempre se mantiene bastante bajo, y en cuanto al response time podemos ver tambien una varianza mucho menor, aunque haya ocacionalmente algunos disparos hacia arriba por lo general se mantienen bastante juntas las dos curvas.



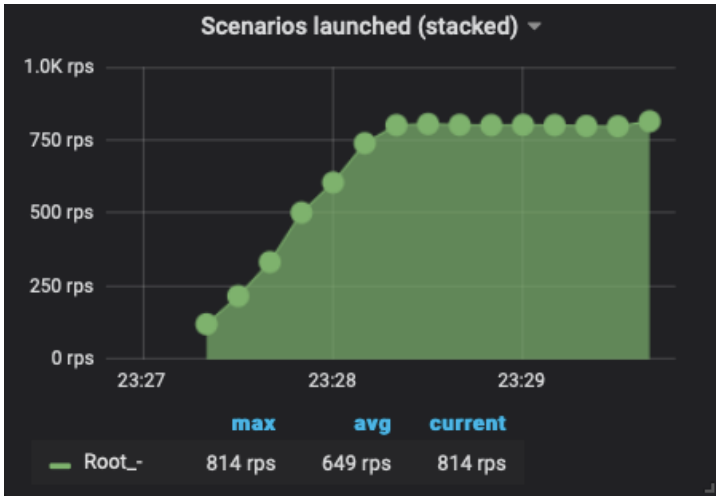






Stress Test

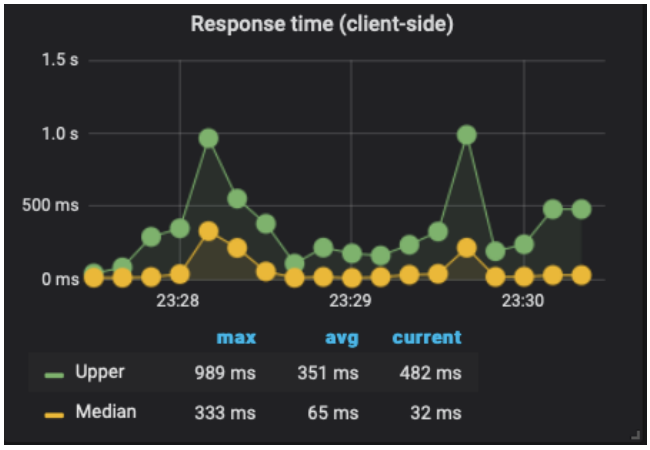
En estas pruebas simulamos un uso mas intensivo de lo comun para el servidor, para esto simulamos 80 usuarios activos por segundo, un total de 4800 requests por minuto. Mantuvimos masomenos las mismas "Phases" que en la prueba anterior, con la diferencia de que esta vez subimos hasta 80. Este esquema se repitio en todos los casos menos en el ultimo, donde probamos un escenario donde quizimos poner al limite al servidor que mas capacidad deberia soportar (Ping) a ver cual era el resultado.

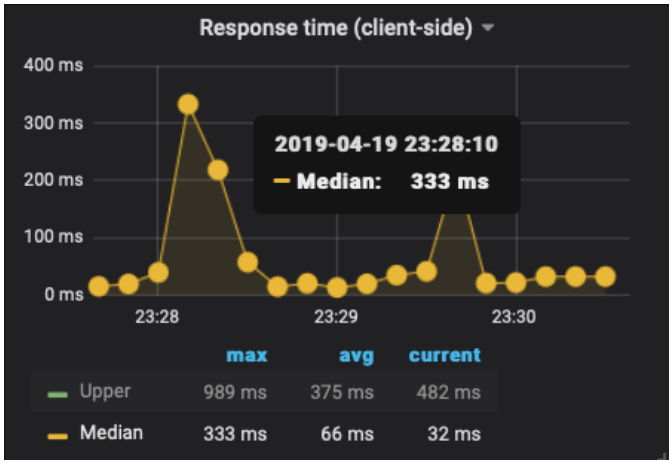


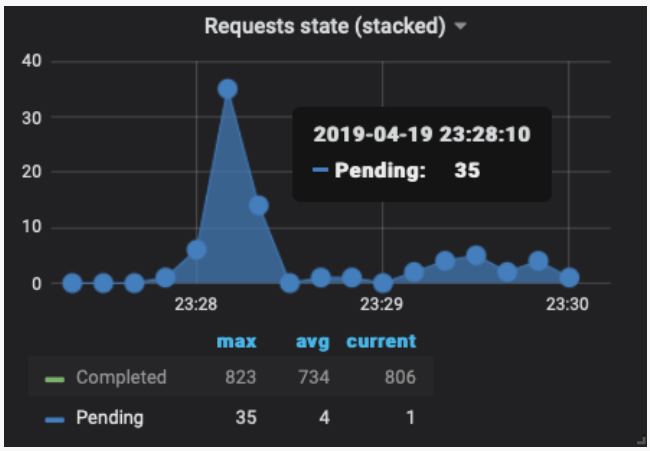
Caso Stress Test Ping

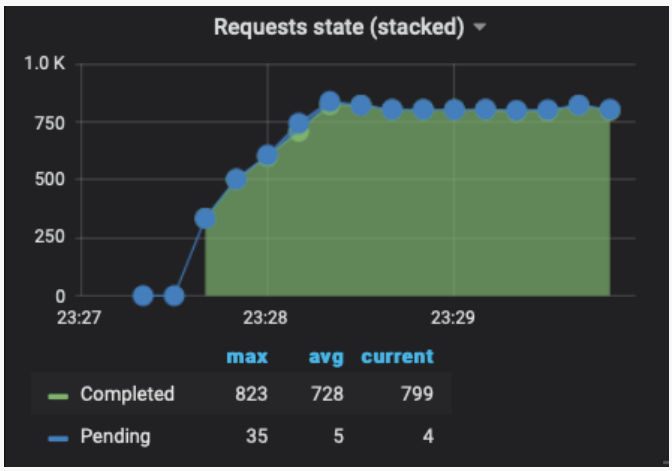
En este endpoint pudimos ver un comporamiento similar al del escenario anterior, aunque los

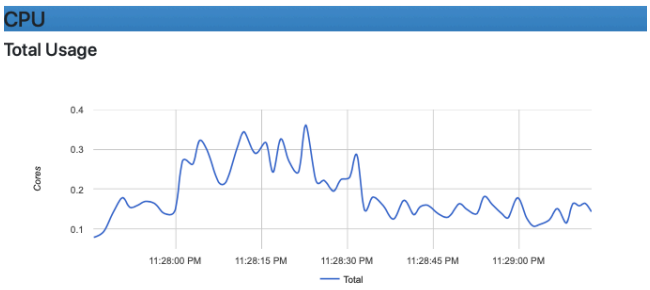
tiempos fueron por lo general mas altos lo cual era de esperarse la curvas tuvieron un comportamiento similar, mientras que el tiemp medio se mantenia dentro de los parametros bajo los tiempos mas altos eran mas erraticos. En cuanto al estado de los requests a medida que pasaba el tiempo tambien podemos ver un comporamiento similar. En resumidas palabras sucedio lo que se esperaba, mismo patron de comportamiento pero con numeros mas altos

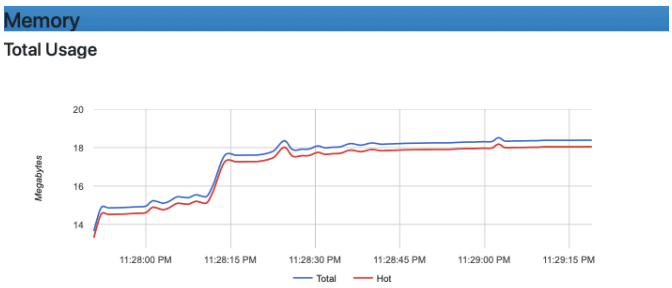








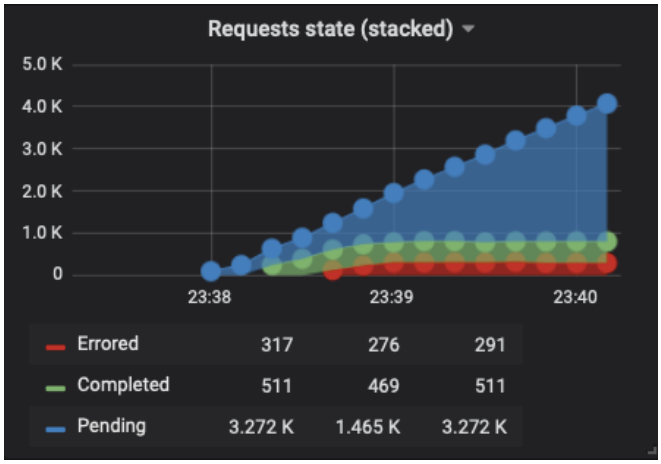


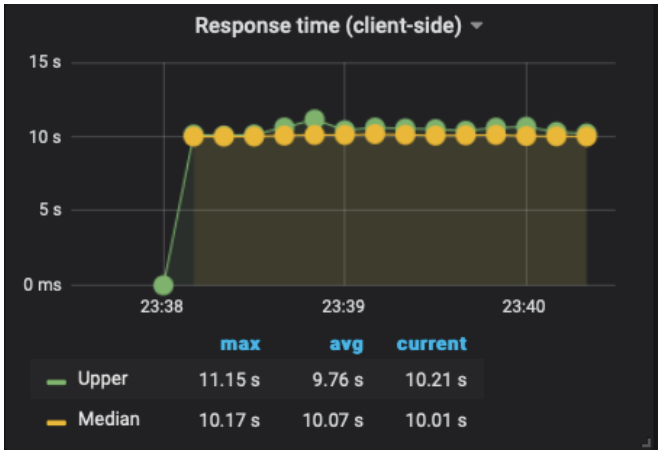


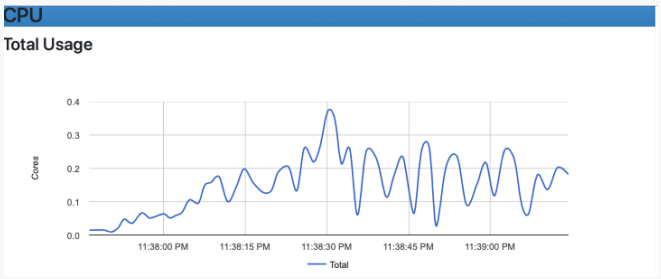
Caso Stress Test Timeout

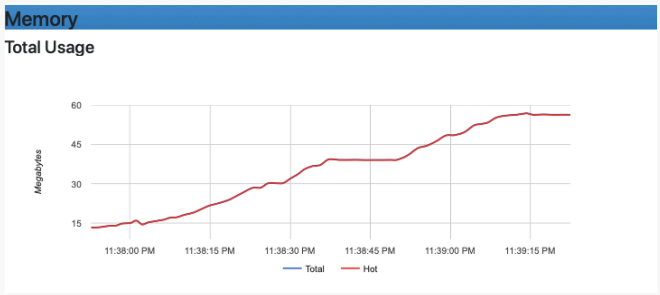
Con el endpoint de timeout podemos ver tambien un comporamiento similar en cuanto a los tiempos de respuesta los cuales se mantuvideron todos bastante parecidos, lo cual fortaleceria

nuestra teoria de que el timeout funciona como un ordenador de los requests y hace que tengan un tiempo de respuesta constante. Por otro lado hubo varios requests que presentaron errores ECONNRESET, que indica que la conexion se cerro abruptamente, esto lo podriamos llegar a adjudicar al overhead del timeout, pero luego de buscar en internet parece ser que la implementacion del setTimeout de Node es bastante eficiente y no deberia haber problema con el timeout.





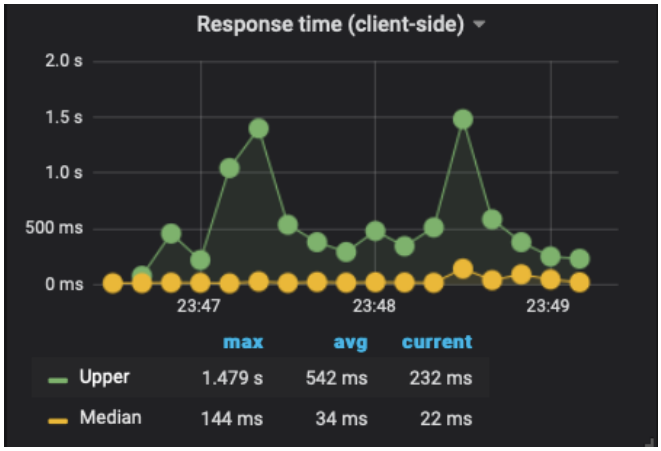


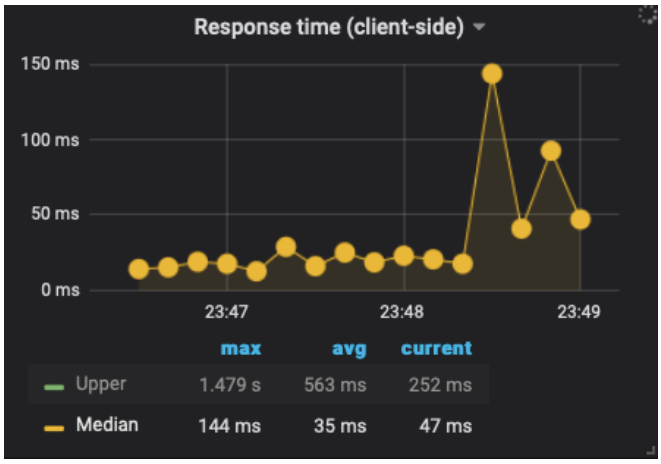


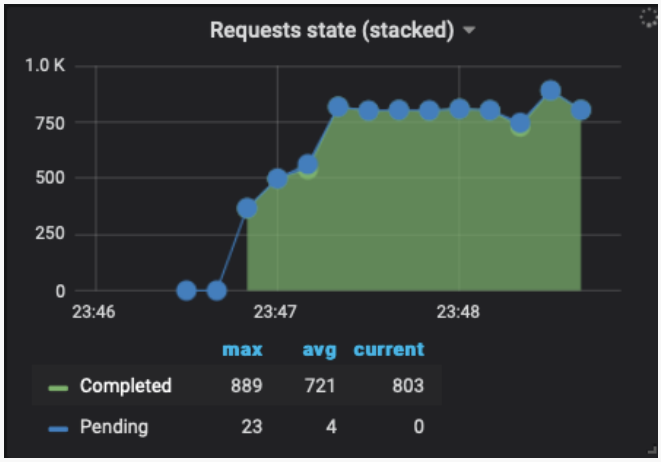

```
All virtual users finished
Summary report @ 23:40:53(-0300) 2019-04-19
  Scenarios launched:  12230
  Scenarios completed: 8288
  Requests completed:  8288
  RPS sent: 64.1
  Request latency:
    min: 9992.6
    max: 11145
    median: 10027.9
    p95: 10303.2
    p99: 10549.5
  Scenario counts:
    Root (/): 12230 (100%)
  Codes:
    200: 8288
  Errors:
    ECONNRESET: 3942
```

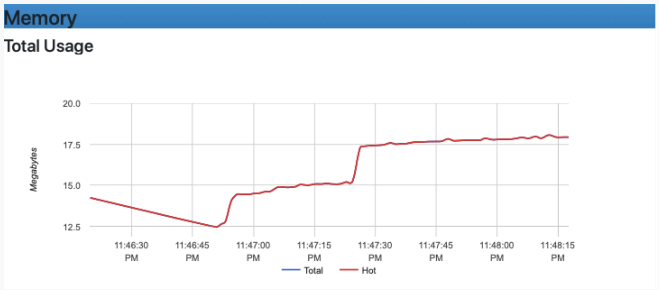
Caso Stress Test Intensive

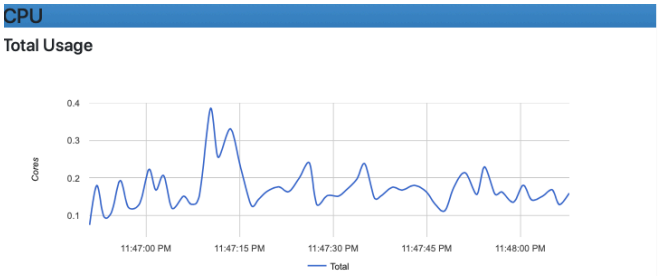
Para este endpoint podemos ver que el patron de comportamiento cambio bastante, no mantuvo una varianza baja, como en el caso del Load Test. Podemos ver igualmente que los valores no crecieron mucho, sino que crecieron como mucho un 50%, en cuanto al resto de las mediciones no hay nada muy importante a notar.







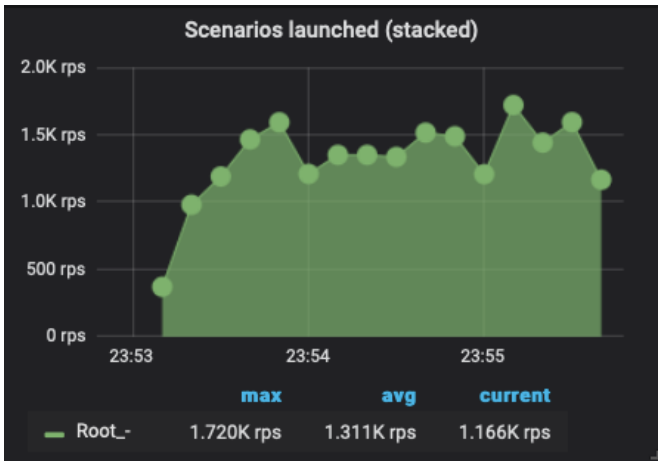


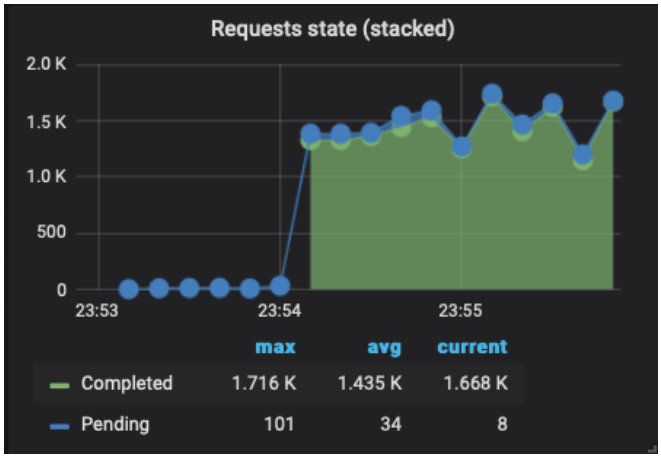


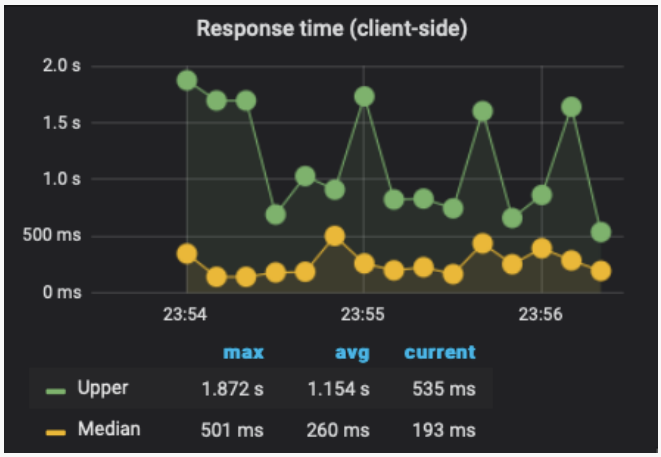
Caso Heavy Stress Test Ping

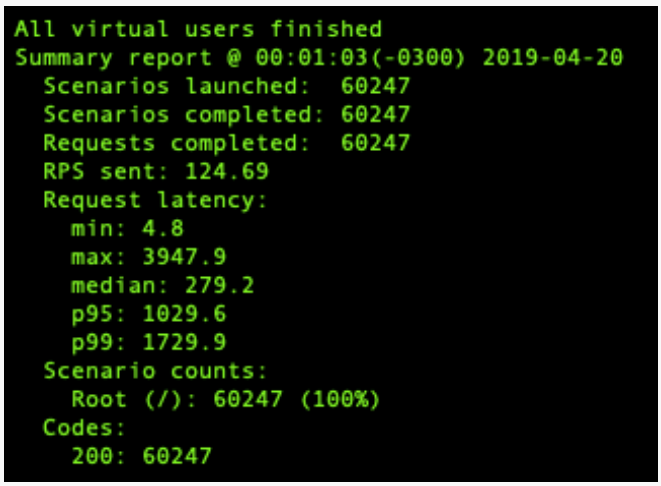
Para este ultimo caso probamos el endpoint Ping con 400 requests por segundo. Lamentablemente artillery no puedo generar tantas como quisimos, y esto lo podemos ver primero por la irregularidad de lo que deberia ser la meseta de request en la imagen de

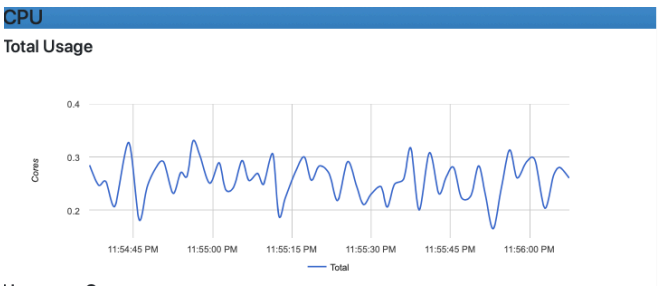
"Scenarios Launched", en su punto mas alto variaba desde 1200 requests cada 10 segundos y 1800. El escenario termino durando 8 minutos (5 mas de lo esperado). En cuanto al tiempo de respuesta podemos ver un comportamiento similar a todos los que probamos el endpoint Ping, hay una varianza muy alta en los tiempos de respuesta.

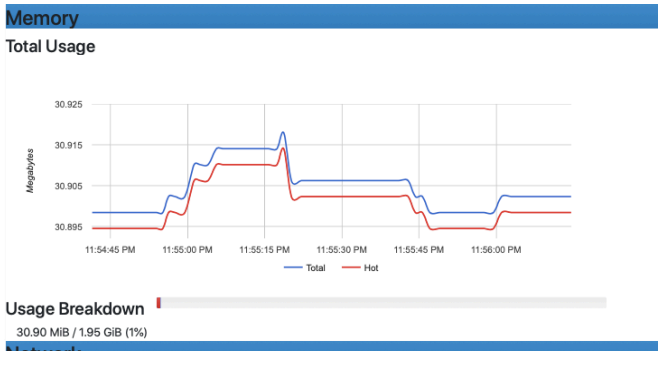








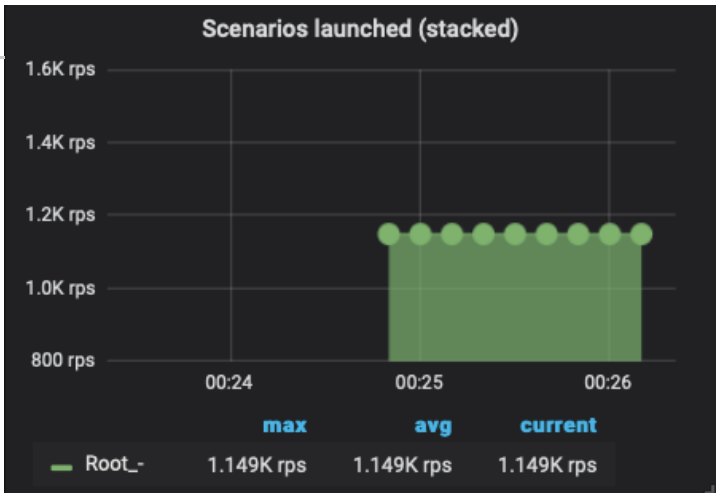




Spike test

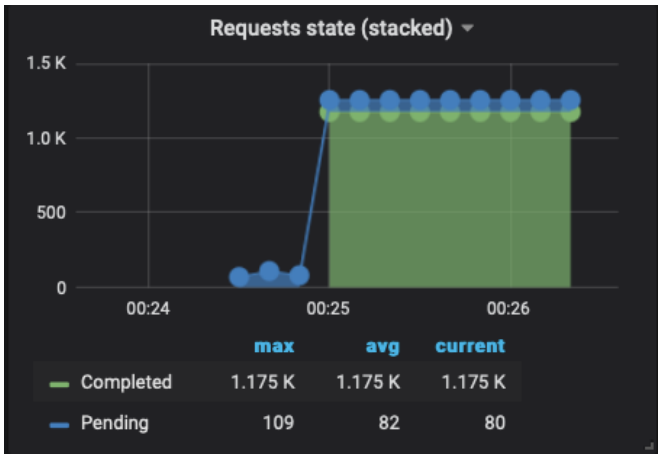
Spike test

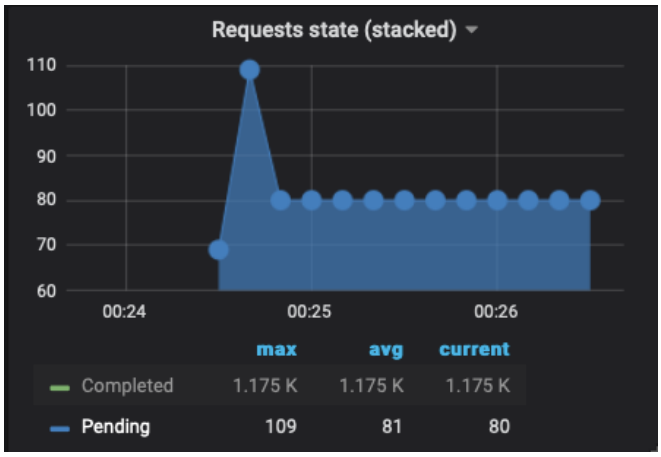
En este ultimo escenario decidimos probar el comportamiento del servidor cuando la cantidad de requests por segundo aumenta una forma no esperada, casi exponencial. Para esto creamos una rampa de 5 requests a 400 requests en 20 segundos, a ver cual era el comportamiento que habria en cada uno de los endpoints del servidor de Node.

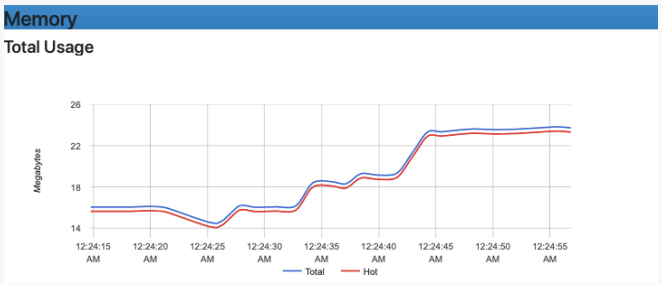


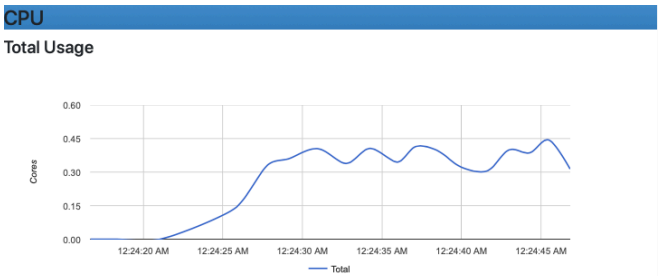
Caso Spike Test Ping

Este fue el escenario mas extraño de todos los que probamos. podemos ver que aunque los tiempos de respuesta tuvieron una gran separacion entre los tiempos medios y los tiempos maximos, la diferencia se mantuvo constante. Puede ser que esto se deba a la duracion del escenario ya que duro solo 20 segundos, pero no tenia sentido subir la cantidad de requests el limite para estos lo encontramos en el escenario pasado y pudimos ver que estaba cerca de 1500 cada 10 segundos.







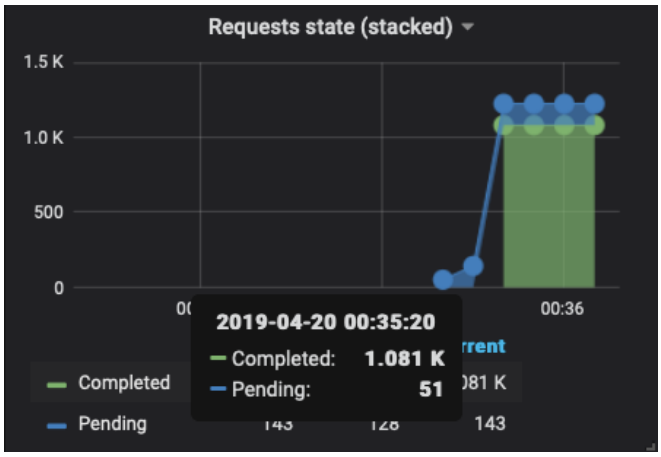


Caso Spike Test Timeout

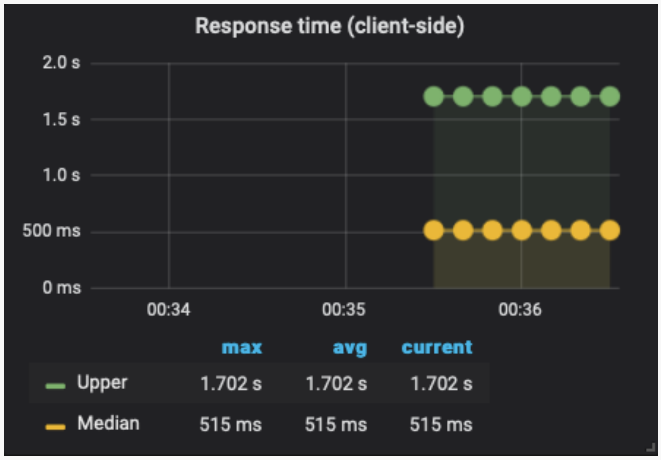
Como era de esperarse al igual que en el caso del Stress Test un determinado numero de requests fallaron en el caso del timeout, pero el patron de comportamiento fue similar

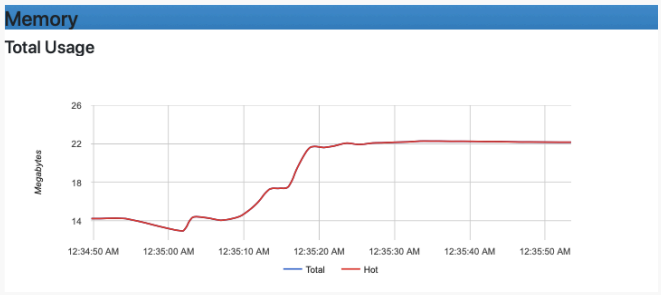
Caso Spike Test Intensive

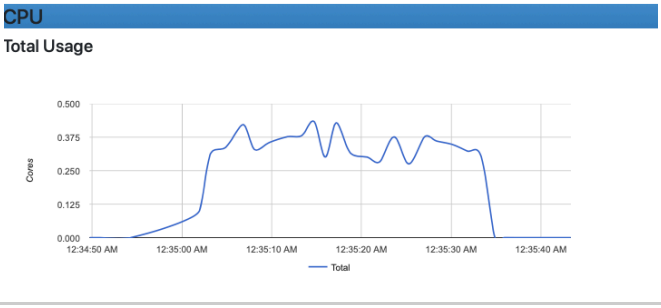
Por ultimos en el caso del endpoint Intensive, tuvimos un comportamiento muy similar al del enpoint Ping el resto de los parametros arrojaron resultados bastante similares.


Requests state (stacked)
1.5 K
1.0 K
500
0
00:36
2019-04-20 00:35:20
Completed: 1.081 K
Pending: 51
Completed
Pending
143


Requests state (stacked)
Completed
100
50
0
00:34
00:35
00:36
max
avg
current
Completed 1.081 K 1.081 K 1.081 K
Pending 143 130 143


Response time (client-side)
2.0 s
1.5 s
1.0 s
500 ms
0 ms
00:34
00:35
00:36
max
avg
current
Upper 1.702 s 1.702 s 1.702 s
Median 515 ms 515 ms 515 ms


Memory
Total Usage
Megabytes
26
22
18
14
12:34:50 AM
12:35:00 AM
12:35:10 AM
12:35:20 AM
12:35:30 AM
12:35:40 AM
12:35:50 AM
Total
Hot


CPU
Total Usage
Cores
0.500
0.375
0.250
0.125
0.000
12:34:50 AM
12:35:00 AM
12:35:10 AM
12:35:20 AM
12:35:30 AM
12:35:40 AM
Total