Laboratorio 7: Pruebas de carga
Informe de resultados de los tests

Francisco Ramírez Cañadas // Jorge Repullo Serrano

Resultado de **smoke-test.js:**

```
  THRESHOLDS

    http_req_duration
    ✓ 'avg<100' avg=5.07ms

    http_req_failed
    ✓ 'rate==0' rate=0.00%


  TOTAL RESULTS

    HTTP
    http_req_duration.....................................................: avg=5.07ms min=1.54ms med=4.59ms max=43.33ms p(90)=6.08ms p(95)=7.1ms
      { expected_response:true }..........................................: avg=5.07ms min=1.54ms med=4.59ms max=43.33ms p(90)=6.08ms p(95)=7.1ms
    http_req_failed.......................................................: 0.00% 0 out of 300
    http_reqs.............................................................: 300   4.971239/s

    EXECUTION
    iteration_duration....................................................: avg=1s     min=1s     med=1s     max=1.05s   p(90)=1s     p(95)=1s
    iterations............................................................: 300   4.971239/s
    vus...................................................................: 5     min=5        max=5
    vus_max...............................................................: 5     min=5        max=5

    NETWORK
    data_received.........................................................: 85 kB 1.4 kB/s
    data_sent.............................................................: 23 kB 388 B/s



running (1m00.3s), 0/5 VUs, 300 complete and 0 interrupted iterations
default ✓ [======================================] 5 VUs  1m0s
```

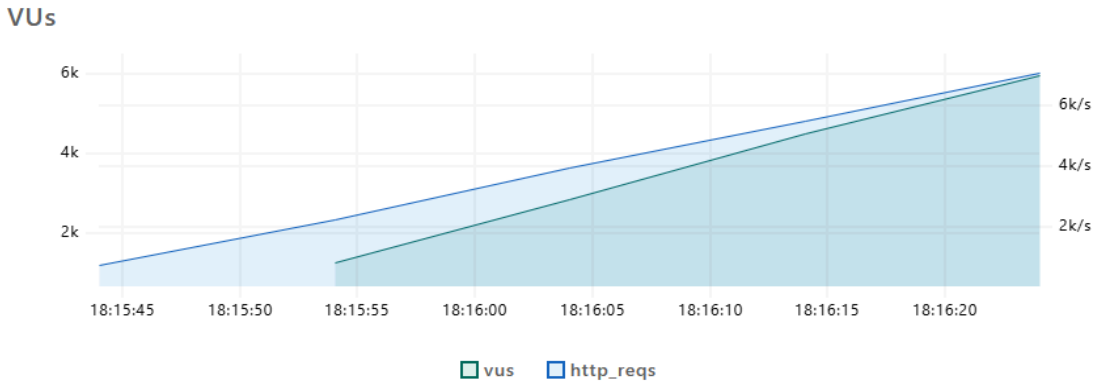
Resultado de **breakpoint_executor-test.js con executor:**

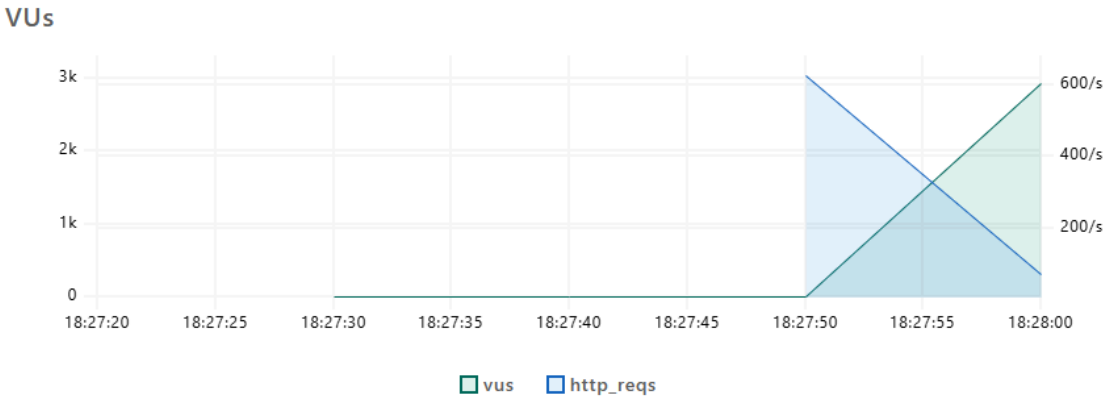
```
  THRESHOLDS

    http_req_failed
    ✗ 'rate<=0.01' rate=1.39%


  TOTAL RESULTS

    checks_total.....................: 276691 4596.140669/s
    checks_succeeded.................: 98.62% 272880 out of 276691
    checks_failed....................: 1.37%  3811 out of 276691

    ✗ status is 200
      ↳  98% — ✓ 272880 / ✗ 3811

    HTTP
    http_req_duration.....................................................: avg=4.12ms min=0s med=0s max=470.14ms p(90)=996.8µs p(95)=2.54ms
      { expected_response:true }..........................................: avg=4.18ms min=0s med=0s max=470.14ms p(90)=996.9µs p(95)=2.64ms
    http_req_failed.......................................................: 1.39%  3855 out of 276735
    http_reqs.............................................................: 276735 4596.871557/s

    EXECUTION
    dropped_iterations....................................................: 22462  373.118431/s
    iteration_duration....................................................: avg=1s     min=1s med=1s max=1.85s    p(90)=1s     p(95)=1.02s
    iterations............................................................: 267344 4440.876758/s
    vus...................................................................: 9627   min=70           max=9627
    vus_max...............................................................: 9628   min=1000         max=9628

    NETWORK
    data_received.........................................................: 78 MB  1.3 MB/s
    data_sent.............................................................: 21 MB  354 kB/s



running (01m00.3s), 00000000/00009635 VUs, 267228 complete and 9634 interrupted iterations
breakpoint ✗ [==>-------------------------------] 00009632/00009634 VUs  01m00.0s/10m0s  009988.71 iters/s
ERRO[0060] thresholds on metrics 'http_req_failed' were crossed; at least one has abortOnFail enabled, stopping test prematurely
```

Resultado de **breakpoint_no_executor-test.js:**

```
  THRESHOLDS

    http_req_failed
    ✗ 'rate<=0.01' rate=4.26%


  TOTAL RESULTS

    checks_total.....................: 6770   311.053272/s
    checks_succeeded.................: 98.46% 6666 out of 6770
    checks_failed....................: 1.53%  104 out of 6770

    ✗ status is 200
      ↳  98% — ✓ 6666 / ✗ 104

    ✗ status is 200
      ↳  98% — ✓ 6666 / ✗ 104

    HTTP
    http_req_duration..............................................: avg=213.11ms min=0s med=133.19ms max=949.95ms p(90)=596.59ms p(95)=715.76ms
      { expected_response:true }...................................: avg=222.61ms min=0s med=146.31ms max=949.95ms p(90)=604.95ms p(95)=715.86ms
    http_req_failed................................................: 4.26%  297 out of 6963
    http_reqs......................................................: 6963   319.920817/s

    EXECUTION
    iteration_duration.............................................: avg=1.37s    min=1s med=1.21s    max=2.53s    p(90)=2.03s    p(95)=2.13s
    iterations.....................................................: 4083   187.596826/s
    vus............................................................: 3076   min=0          max=3076
    vus_max........................................................: 100000 min=11775      max=100000

    NETWORK
    data_received..................................................: 1.9 MB 89 kB/s
    data_sent......................................................: 533 kB 25 kB/s



running (00m21.9s), 000000/100000 VUs, 4081 complete and 3081 interrupted iterations
default ✗ [----------------------------------] 000181/100000 VUs  00m08.5s/10m00.0s
```

Podemos observar cómo con “executor” llegamos a un máximo de 9.5k VUs, mientras que usando stages llegamos a 3k (punto de rotura de referencia). Si comparamos las gráficas, primero la del caso de executors y segundo la de stages:



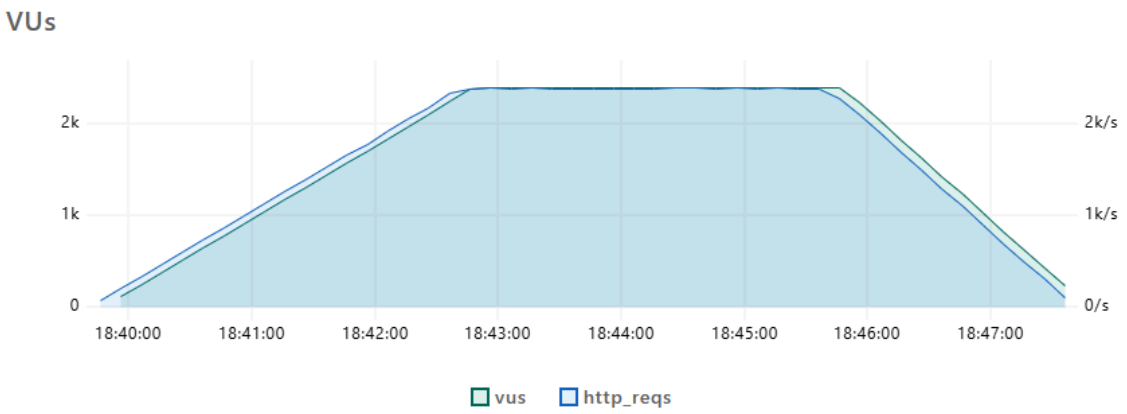


Podemos ver claramente como se aprecia en el primer caso el “ramping-arrival-rate”, mientras que en el segundo es más “de golpe”, por ello rompe antes que con executor.

Resultados de **stress-test.js:**

```
  THRESHOLDS

    http_req_duration
    ✓ 'avg<1000' avg=849.68µs

    http_req_failed
    ✓ 'rate<0.01' rate=0.00%


  TOTAL RESULTS

    HTTP
    http_req_duration.....................................................: avg=849.68µs min=0s med=589.7µs max=90.56ms p(90)=1.64ms p(95)=2.28ms
      { expected_response:true }..........................................: avg=849.68µs min=0s med=589.7µs max=90.56ms p(90)=1.64ms p(95)=2.28ms
    http_req_failed.......................................................: 0.00%  0 out of 791891
    http_reqs.............................................................: 791891 1646.811003/s

    EXECUTION
    iteration_duration....................................................: avg=1s      min=1s med=1s      max=1.14s   p(90)=1s     p(95)=1s
    iterations............................................................: 791891 1646.811003/s
    vus...................................................................: 1      min=1          max=2400
    vus_max...............................................................: 2400   min=2400       max=2400

    NETWORK
    data_received.........................................................: 225 MB 468 kB/s
    data_sent.............................................................: 62 MB  129 kB/s



running (8m00.9s), 0000/2400 VUs, 791891 complete and 0 interrupted iterations
default ✓ [======================================] 0000/2400 VUs  8m0s
```

Lo pasa satisfactoriamente.

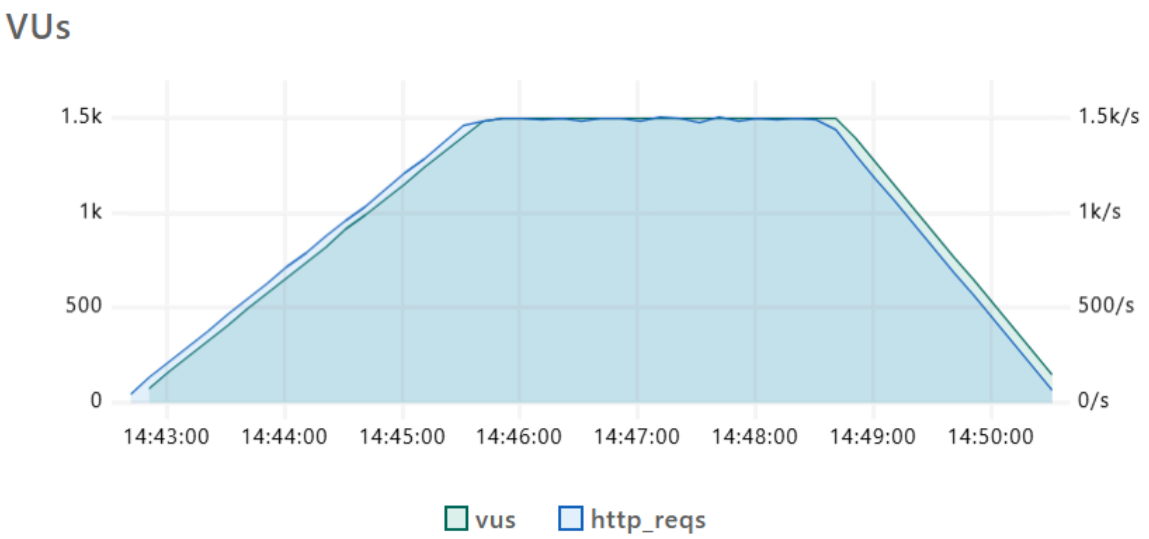


Vemos en su gráfica como sube durante 3m hasta mantenerse constante en los 2.4k (80% de VUs del punto de rotura sin executors), y en los últimos 2m empieza a descender el número de VUs hasta llegar a 0.

Resultados de **average-test.js:**

```
  THRESHOLDS

    http_req_failed
    ✓ 'rate<0.01' rate=0.00%


  TOTAL RESULTS

    HTTP
    http_req_duration.....................................................: avg=601.58µs min=0s med=517µs max=326.54ms p(90)=1.13ms p(95)=1.55ms
      { expected_response:true }..........................................: avg=601.58µs min=0s med=517µs max=326.54ms p(90)=1.13ms p(95)=1.55ms
    http_req_failed.......................................................: 0.00%  0 out of 495150
    http_reqs.............................................................: 495150 1030.620587/s

    EXECUTION
    iteration_duration....................................................: avg=1s      min=1s med=1s    max=1.32s    p(90)=1s     p(95)=1s
    iterations............................................................: 495150 1030.620587/s
    vus...................................................................: 9      min=7          max=1500
    vus_max...............................................................: 1500   min=1500       max=1500

    NETWORK
    data_received.........................................................: 141 MB 293 kB/s
    data_sent.............................................................: 39 MB  80 kB/s



running (8m00.4s), 0000/1500 VUs, 495150 complete and 0 interrupted iterations
default ✓ [======================================] 0000/1500 VUs  8m0s
```

Lo pasa satisfactoriamente.

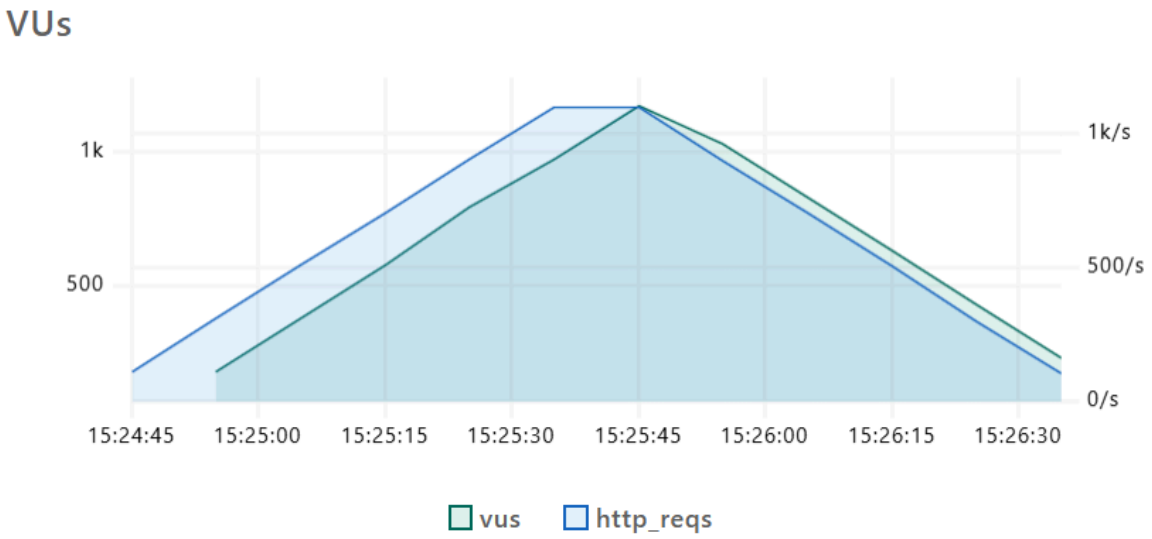


Vemos en su gráfica como el número de usuarios virtuales asciende hasta los 1500 (50% de VUs del punto de rotura sin executors) durante 3 minutos, después se mantiene otros 3, y finalmente baja hasta 0 en los 2 últimos minutos.

Resultado de **spike-test.js:**

```
  THRESHOLDS

    http_req_failed
    ✓ 'rate<=0.005' rate=0.00%


  TOTAL RESULTS

    HTTP
    http_req_duration.....................................................: avg=685.85µs min=0s med=544µs max=55.97ms p(90)=1.11ms p(95)=1.44ms
      { expected_response:true }..........................................: avg=685.85µs min=0s med=544µs max=55.97ms p(90)=1.11ms p(95)=1.44ms
    http_req_failed.......................................................: 0.00%  0 out of 72538
    http_reqs.............................................................: 72538  601.777104/s

    EXECUTION
    iteration_duration....................................................: avg=1s      min=1s med=1s    max=1.05s   p(90)=1s     p(95)=1s
    iterations............................................................: 72538  601.777104/s
    vus...................................................................: 13     min=13         max=1197
    vus_max...............................................................: 1200   min=1200       max=1200

    NETWORK
    data_received.........................................................: 21 MB  171 kB/s
    data_sent.............................................................: 5.7 MB 47 kB/s



running (2m00.5s), 0000/1200 VUs, 72538 complete and 0 interrupted iterations
default ✓ [======================================] 0000/1200 VUs  2m0s
```

Lo pasa satisfactoriamente.
A continuación podemos observar el “pico” generado por el número de usuarios virtuales:



Afirmamos que el sistema es capaz de gestionar el 40% de la carga máxima en tan sólo 2 minutos.