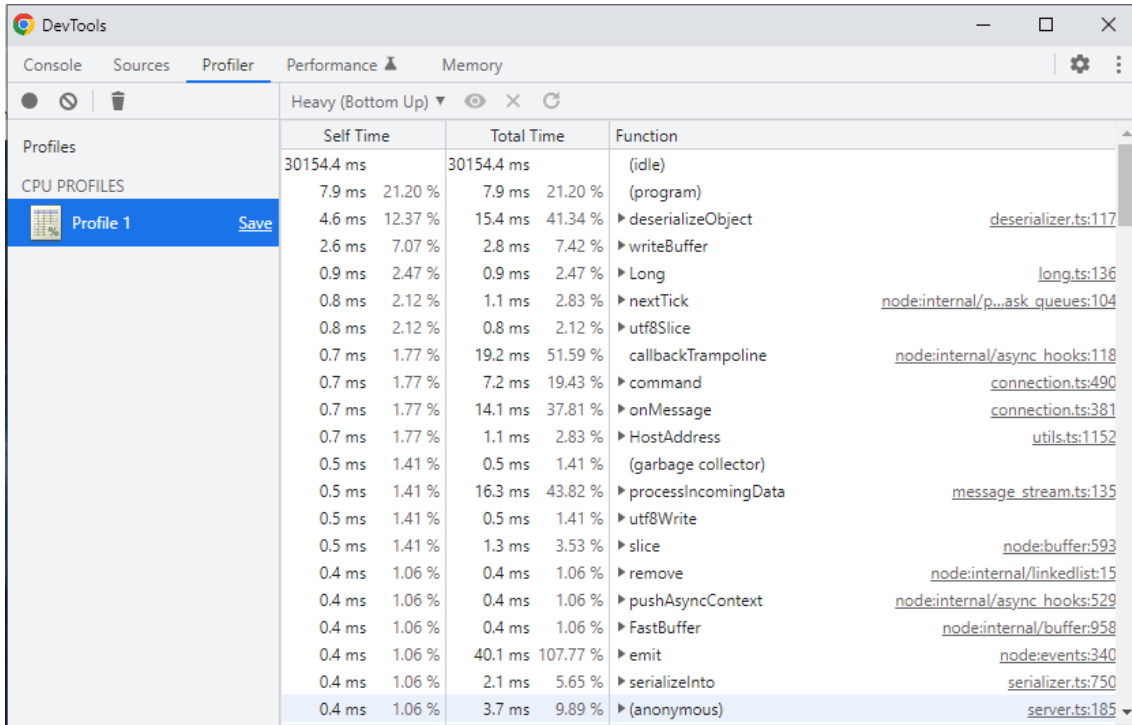


### Tercera Entrega Final - Prueba Performance - Luigi Marquez

- Prueba realizada con Cluster en la ruta <http://localhost:8081/products>



The screenshot shows the Chrome DevTools Performance tab with a CPU profile. The profile is titled 'Heavy (Bottom Up)'. The table below represents the data shown in the 'Function' column of the profile.

Function	Self Time	Total Time
(idle)	30154.4 ms	30154.4 ms
(program)	7.9 ms	21.20 %
deserializeObject	4.6 ms	12.37 %
writeBuffer	2.6 ms	7.07 %
Long	0.9 ms	2.47 %
nextTick	0.8 ms	2.12 %
utf8Slice	0.8 ms	2.12 %
callbackTrampoline	0.7 ms	1.77 %
command	0.7 ms	1.77 %
onMessage	0.7 ms	1.77 %
HostAddress	0.7 ms	1.77 %
(garbage collector)	0.5 ms	1.41 %
processIncomingData	0.5 ms	1.41 %
utf8Write	0.5 ms	1.41 %
slice	0.5 ms	1.41 %
remove	0.4 ms	1.06 %
pushAsyncContext	0.4 ms	1.06 %
FastBuffer	0.4 ms	1.06 %
emit	0.4 ms	1.06 %
serializeInto	0.4 ms	1.06 %
(anonymous)	0.4 ms	1.06 %

```
Phase started: unnamed (index: 0, duration: 1s) 00:47:48(-0300)
Phase completed: unnamed (index: 0, duration: 1s) 00:47:51(-0300)

-----
Metrics for period to: 00:47:50(-0300) (width: 1.414s)
-----

http.codes.200: ..... 295
http.request_rate: ..... 237/sec
http.requests: ..... 319
http.response_time:
  min: ..... 2
  max: ..... 144
  median: ..... 58.6
  p95: ..... 100.5
  p99: ..... 108.9
http.responses: ..... 295
vusers.completed: ..... 3
vusers.created: ..... 28
vusers.created_by_name.0: ..... 28
vusers.failed: ..... 0
vusers.session_length:
  min: ..... 1125.9
  max: ..... 1305.9
  median: ..... 1274.3
  p95: ..... 1274.3
  p99: ..... 1274.3
```

-----  
Metrics for period to: 00:48:00(-0300) (width: 2.393s)  
-----

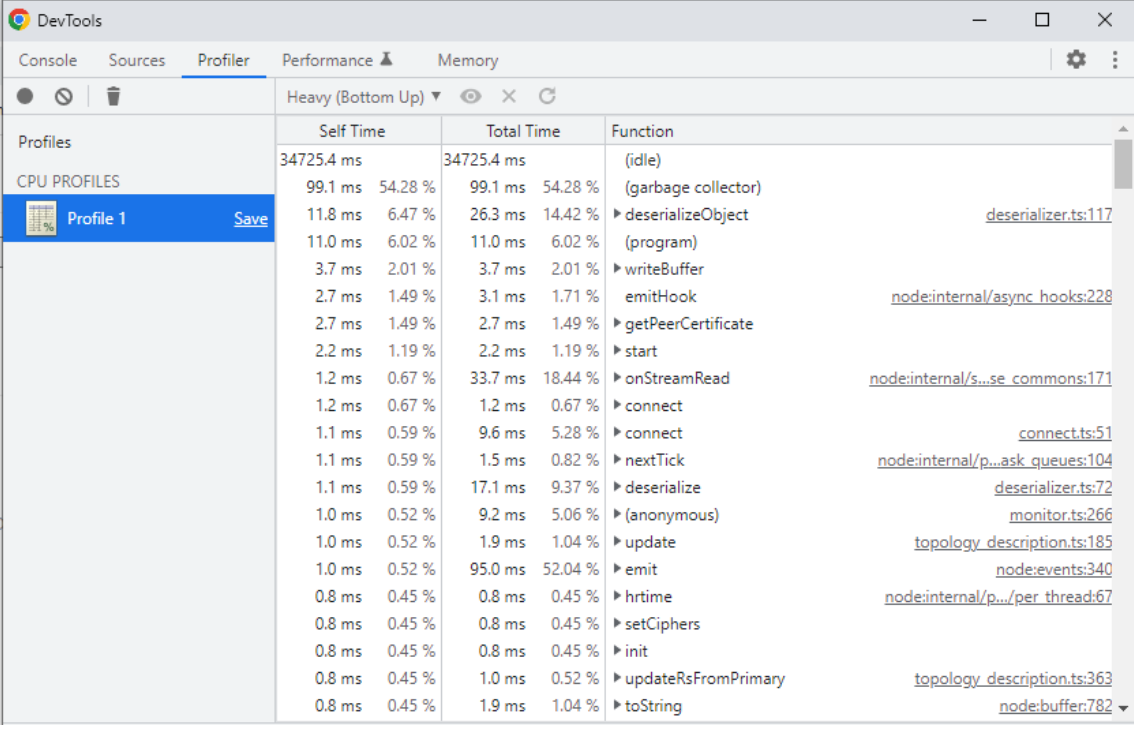
http.codes.200:	705
http.request_rate:	285/sec
http.requests:	681
http.response_time:	
min:	1
max:	186
median:	66
p95:	149.9
p99:	172.5
http.responses:	705
vusers.completed:	47
vusers.created:	22
vusers.created_by_name.0:	22
vusers.failed:	0
vusers.session_length:	
min:	913.8
max:	1965.2
median:	1652.8
p95:	1901.1
p99:	1939.5

All VUs finished. Total time: 4 seconds

-----  
Summary report @ 00:47:53(-0300)  
-----

http.codes.200:	1000
http.request_rate:	261/sec
http.requests:	1000
http.response_time:	
min:	1
max:	186
p99:	172.5
http.responses:	1000
vusers.completed:	50
vusers.created:	50
vusers.created_by_name.0:	50
vusers.failed:	0
vusers.session_length:	
min:	913.8
max:	1965.2
median:	1556.5
p95:	1901.1
p99:	1939.5

- Prueba realizada sin Cluster en la ruta <http://localhost:8081/products>



Self Time		Total Time		Function
34725.4 ms		34725.4 ms		(idle)
99.1 ms	54.28 %	99.1 ms	54.28 %	(garbage collector)
11.8 ms	6.47 %	26.3 ms	14.42 %	▶ deserializeObject <a href="#">deserializer.ts:117</a>
11.0 ms	6.02 %	11.0 ms	6.02 %	(program)
3.7 ms	2.01 %	3.7 ms	2.01 %	▶ writeBuffer
2.7 ms	1.49 %	3.1 ms	1.71 %	emitHook <a href="#">node:internal/async_hooks:228</a>
2.7 ms	1.49 %	2.7 ms	1.49 %	▶ getPeerCertificate
2.2 ms	1.19 %	2.2 ms	1.19 %	▶ start
1.2 ms	0.67 %	33.7 ms	18.44 %	▶ onStreamRead <a href="#">node:internal/s...se commons:171</a>
1.2 ms	0.67 %	1.2 ms	0.67 %	▶ connect
1.1 ms	0.59 %	9.6 ms	5.28 %	▶ connect <a href="#">connect.ts:51</a>
1.1 ms	0.59 %	1.5 ms	0.82 %	▶ nextTick <a href="#">node:internal/p...ask queues:104</a>
1.1 ms	0.59 %	17.1 ms	9.37 %	▶ deserialize <a href="#">deserializer.ts:72</a>
1.0 ms	0.52 %	9.2 ms	5.06 %	▶ (anonymous) <a href="#">monitor.ts:266</a>
1.0 ms	0.52 %	1.9 ms	1.04 %	▶ update <a href="#">topology_description.ts:185</a>
1.0 ms	0.52 %	95.0 ms	52.04 %	▶ emit <a href="#">node:events:340</a>
0.8 ms	0.45 %	0.8 ms	0.45 %	▶ hrtime <a href="#">node:internal/p.../per_thread:67</a>
0.8 ms	0.45 %	0.8 ms	0.45 %	▶ setCiphers
0.8 ms	0.45 %	0.8 ms	0.45 %	▶ init
0.8 ms	0.45 %	1.0 ms	0.52 %	▶ updateRsFromPrimary <a href="#">topology_description.ts:363</a>
0.8 ms	0.45 %	1.9 ms	1.04 %	▶ toString <a href="#">node:buffer:782</a>

```
Phase started: unnamed (index: 0, duration: 1s) 00:50:51(-0300)
Phase completed: unnamed (index: 0, duration: 1s) 00:50:52(-0300)

-----
Metrics for period to: 00:51:00(-0300) (width: 6.03s)
-----

http.codes.200: ..... 1000
http.request_rate: ..... 170/sec
http.requests: ..... 1000
http.response_time:
  min: ..... 4
  max: ..... 582
  median: ..... 198.4
  p95: ..... 459.5
  p99: ..... 539.2
http.responses: ..... 1000
vusers.completed: ..... 50
vusers.created: ..... 50
vusers.created_by_name.0: ..... 50
vusers.failed: ..... 0
vusers.session_length:
  min: ..... 2659.5
  max: ..... 4764.4
  median: ..... 4492.8
  p95: ..... 4676.2
  p99: ..... 4676.2
```

```
All VUs finished. Total time: 7 seconds
```

```
-----  
Summary report @ 00:50:58(-0300)  
-----
```

```
http.codes.200: ..... 1000  
http.request_rate: ..... 170/sec  
http.requests: ..... 1000  
http.response_time:  
  min: ..... 4  
  max: ..... 582  
  median: ..... 198.4  
  p95: ..... 459.5  
  p99: ..... 539.2  
http.responses: ..... 1000  
vusers.completed: ..... 50  
vusers.created: ..... 50  
vusers.created_by_name.0: ..... 50  
vusers.failed: ..... 0  
vusers.session_length:  
  min: ..... 2659.5  
  max: ..... 4764.4  
  median: ..... 4492.8  
  p95: ..... 4676.2  
  p99: ..... 4676.2
```

Para la prueba del profiling, se utilizó artillery con:

```
artillery quick --count 50 --num 20 'http://localhost:8081/products'
```

Se ejecutaron 50 solicitudes con 20 peticiones GET; para el profiling se usó la herramienta de Chrome *DevTools for Node* en **chrome://inspect/** pestaña profiler, corriendo el server **como node --inspect server.js**. De estos resultados, podemos ver que con el Cluster activado (para el caso de esta prueba el procesador tiene 4 hilos), se ejecuta 4600 ms segundos más rápido: 30154ms contra 34725ms. Usando los Cluster distribuimos los recursos de nuestro hardware y se usan con mas eficiencia: al dividir el trabajo se realiza en menor tiempo, disminuyendo el tiempo que se gasta ejecutando nuestra app.