
Práctica 3.3

Índice

Pruebas sin clúster.....3

Pruebas con clúster.....4

Pruebas con loadtest.....5

Pruebas con PM2.....6

Tarea.....7

Cuestiones.....8

Pruebas sin clúster

Vamos a hacer dos comprobaciones para ver la ejecución y lo rápido que es, ahora vamos a poner un valor pequeño

¿Quieres restaurar las páginas?
Chrome no se ha cerrado correctamente.
[Restaurar](#)

Nombre	Estado	Tipo	Iniciador	Tamaño	Hora	Cascada
50	200	document	Otros	178 B	4 ms	
page-script.js	200	script	content-script			

Puesto en cola: 0
Hora de inicio: 1.40 ms

Evento	DURACIÓN
Programación de recursos	1.40 ms
En cola	
Inicio de la conexión	0.68 ms
Detenida	0.25 ms
Negociación de proxy	
Solicitud/Respuesta	47 µs
Solicitud enviada	2.47 ms
Esperando la respuesta del servidor	0.73 ms
Descarga de contenido	5.32 ms

Solicitudes: 2 | Se ha transferido 13.3 kB | Recursos: 13.2 kB | Fil...

Consola: Bloqueo de solicitudes de red

Tiempos del servidor
Durante el desarrollo, puedes usar la API de tiempos del servidor para añadir estadísticas a los tiempos de esta solicitud en el servidor.

Ahora vamos a comprobar la velocidad con un valor más grande, como podemos comprobar tarda bastante más.

¿Quieres restaurar las páginas?
Chrome no se ha cerrado correctamente.
[Restaurar](#)

```
GNU nano 7.2 practica_sin_cluster.js
const express = require("express");
const app = express();
const port = 3000;

app.get("/", (req, res) => {
  res.send("Hello World!");
});

app.get("/api/:n", function (req, res) {
  let n = parseInt(req.params.n);
  let count = 0;

  if (n > 5000000000) n = 5000000000;

  for (let i = 0; i <= n; i++) {
    count += i;
  }

  res.send(`Final count is ${count}`);
});

app.listen(port, () => {
  console.log(`App listening on port ${port}`);
});
```

Nombre	Estado	Tipo	Iniciador	Tamaño	Hora	Cascada
5000000000	200	document	Otros	263 B	5.19 s	
page-script.js	200	script	content-script			

Puesto en cola: 0
Hora de inicio: 1.33 ms

Evento	DURACIÓN
Programación de recursos	1.33 ms
En cola	
Inicio de la conexión	0.38 ms
Detenida	0.22 ms
Negociación de proxy	
Solicitud/Respuesta	88 µs
Solicitud enviada	5.19 s
Esperando la respuesta del servidor	1.62 ms
Descarga de contenido	5.19 s

Solicitudes: 2 | Se ha transferido 13.4 kB | Recursos: 13.2 kB | Fil...

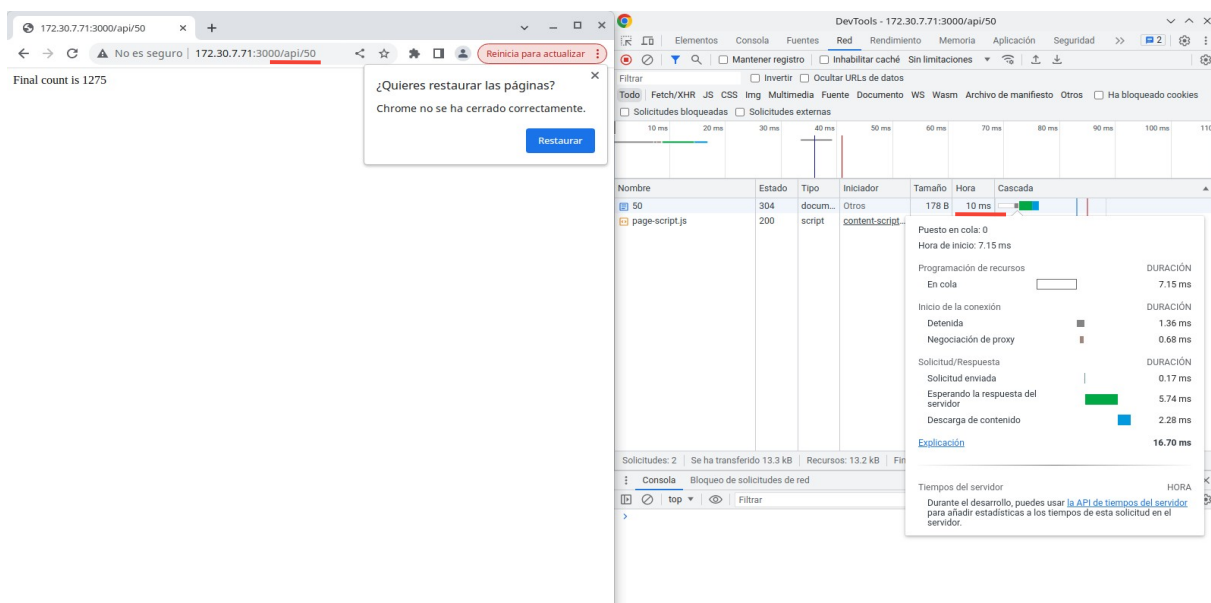
Consola: Bloqueo de solicitudes de red

Tiempos del servidor
Durante el desarrollo, puedes usar la API de tiempos del servidor para añadir estadísticas a los tiempos de esta solicitud en el servidor.

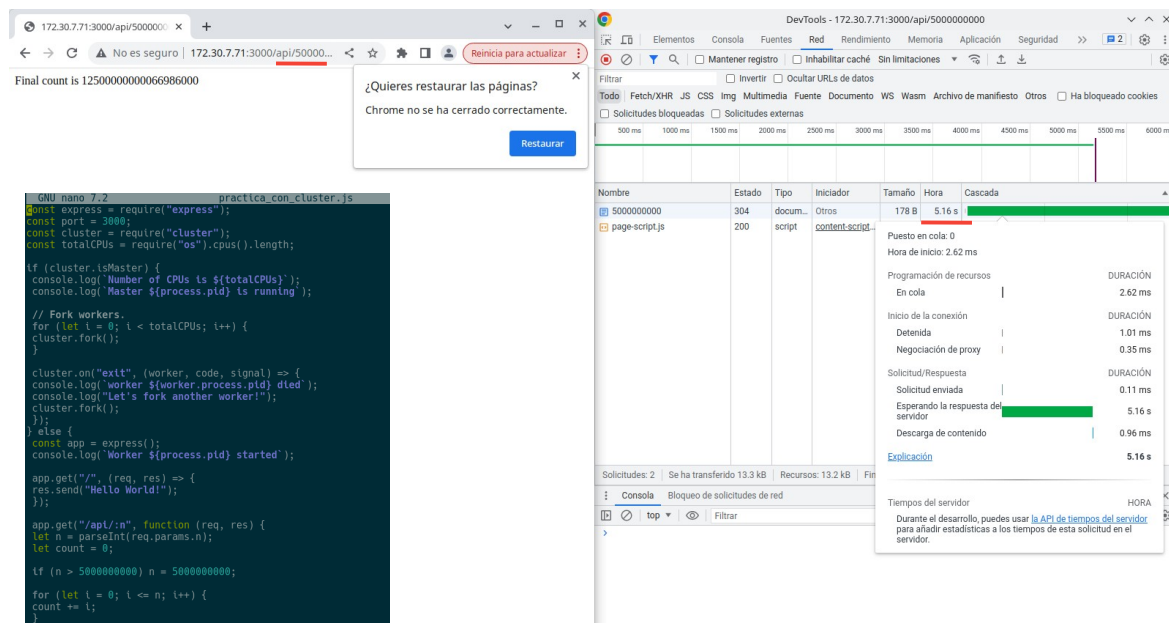
16:51
16/11/23

Pruebas con clúster

Ahora vamos a comprobar la practica anterior pero usaremos el módulo clúster, con un valor pequeño nos tarda poco.



A continuación vamos a probar con un valor más grande, tendría que reducirse el tiempo porque cuando usamos el modulo clúster el proceso lo divide en los módulos de la CPU peor por algún motivo de la maquina virtual solo esta utilizando un núcleo y por eso nos esta tardando la solicitud.



Pruebas con loadtest

Para la siguiente practica nos tendremos que instalar loadtest y ejecutar la aplicación sin el clúster y con el clúster, por algún motivo de la maquina virtual nos salen resultados parecidos.

Sin clúster

```
mpozo@mpozo:~$ loadtest http://localhost:3000/api/500000 -n 1000 -c 100

Target URL:      http://localhost:3000/api/500000
Max requests:    1000
Concurrent clients: 200
Running on cores: 2
Agent:           none

Completed requests: 1000
Total errors:       0
Total time:        1.142 s
Mean latency:      203.7 ms
Effective rps:     876

Percentage of requests served within a certain time
 50%    217 ms
 90%    229 ms
 95%    231 ms
 99%    239 ms
100%    244 ms (longest request)
mpozo@mpozo:~$
```

Con clúster

```
mpozo@mpozo:~$ loadtest http://localhost:3000/api/500000 -n 1000 -c 100

Target URL:      http://localhost:3000/api/500000
Max requests:    1000
Concurrent clients: 200
Running on cores: 2
Agent:           none

Completed requests: 1000
Total errors:       0
Total time:        0.965 s
Mean latency:      166.3 ms
Effective rps:     1036

Percentage of requests served within a certain time
 50%    163 ms
 90%    209 ms
 95%    247 ms
 99%    263 ms
100%    266 ms (longest request)
mpozo@mpozo:~$
```

Pruebas con PM2

Para la última practica vamos a instalar PM2, a continuación ejecutaremos el js sin clusterizar.

```
mpozo@mpozo:~/Documentos/sinCluster$ pm2 start Node.js -i 0
[PM2] Applying action restartProcessId on app [Node](ids: [ 0, 1 ])
[PM2] [Node](0) ✓
[PM2] [Node](1) ✓
[PM2] Process successfully started
```

id	name	mode	u	status	cpu	memory
0	Node	fork	1	online	0%	39.0mb
1	Node	fork	30	online	0%	13.0mb

```
mpozo@mpozo:~/Documentos/sinCluster$
```

Podemos detener la aplicación con stop.

```
mpozo@mpozo:~/Documentos/sinCluster$ pm2 stop Node.js
[PM2] Applying action stopProcessId on app [Node.js](ids: [ 0, 1 ])
[PM2] [Node](0) ✓
[PM2] [Node.js](1) ✓
```

id	name	mode	u	status	cpu	memory
0	Node	fork	1	stopped	0%	0b
1	Node	fork	45	stopped	0%	0b

```
mpozo@mpozo:~/Documentos/sinCluster$
```

Ahora vamos a crear un archivo de configuración llamado Ecosystem para establecer configuraciones específicas y que nos clusterizar solo nuestros archivos.

```
GNU nano 7.2 ecosystem.config.js
module.exports = {
  apps: [
    {
      name: "practica3_3",
      script: "Node.js",
      instances: 0,
      exec_mode: "cluster",
    },
  ],
};
```

Y podemos iniciar directamente ecosystem.

```
mpozo@mpozo:~/Documentos/sinCluster$ pm2 ls
```

id	name	mode	u	status	cpu	memory
0	Node	fork	1	stopped	0%	0b
1	Node	fork	45	stopped	0%	0b
2	practica3_3	cluster	0	online	0%	60.6mb
3	practica3_3	cluster	0	online	0%	60.9mb

Tarea

pm2 ls: Nos muestra una lista de todos los procesos de PM2 con detalles avanzados.

```
mpozo@mpozo:~/Documentos/sinCluster$ pm2 ls
```

id	name	mode	u	status	cpu	memory
0	Node	fork	1	stopped	0%	0b
1	Node	fork	45	stopped	0%	0b
2	practica3_3	cluster	0	online	0%	60.6mb
3	practica3_3	cluster	0	online	0%	60.9mb

pm2 logs: Muestra los registros en tiempo real.

```
mpozo@mpozo:~/Documentos/sinCluster$ pm2 logs
```

```
PM2      | 2023-11-16T16:05:11: PM2 log: App [Node:1] online
PM2      | 2023-11-16T16:05:11: PM2 log: App [Node:1] exited with code [1] via sign
al [SIGINT]
PM2      | 2023-11-16T16:05:11: PM2 log: App [Node:1] starting in -fork mode-
PM2      | 2023-11-16T16:05:11: PM2 log: App [Node:1] online
PM2      | 2023-11-16T16:05:12: PM2 log: App [Node:1] exited with code [1] via sign
al [SIGINT]
PM2      | 2023-11-16T16:05:12: PM2 log: Script /home/mpozo/Documentos/sinCluster/N
ode.js had too many unstable restarts (16). Stopped. "errored"
PM2      | 2023-11-16T16:06:06: PM2 log: Stopping app:Node id:0
PM2      | 2023-11-16T16:06:06: PM2 log: App [Node:0] exited with code [0] via sign
al [SIGINT]
PM2      | 2023-11-16T16:06:06: PM2 log: pid=1950 msg=process killed
PM2      | 2023-11-16T16:06:06: PM2 log: Stopping app:Node id:1
PM2      | 2023-11-16T16:06:06: PM2 error: app=Node id=1 does not have a pid
PM2      | 2023-11-16T16:09:03: PM2 log: App [practica3_3:2] starting in -cluster m
ode-
```

pm2 monit: Inicia un panel de control en tiempo real con información de cada aplicación.

```
Process List
```

id	name	Mem
[0]	Node	Mem:
[1]	Node	Mem:
[2]	practica3_3	
[3]	practica3_3	

```
practica3_3 Logs
```

```
Custom Metrics
```

Metric	Value
Used Heap Size	
Heap Usage	
Heap Size	10.29
Event Loop Latency	
Active handles	1
Active requests	

```
Metadata
```

Key	Value
App Name	practica3_3
Namespace	default
Version	1.0.0
Restarts	0
Uptime	2m
Script path	/home/mpozo/Documentos/sinCluste
r/Node.js	
Script args	N/A

left/right: switch boards | up/down/mouse: scroll | Ctrl-C: exit To go further check out <https://pm2.io/>

Cuestiones

¿Sabrías decir por qué en algunos casos concretos, como este, la aplicación sin clusterizar tiene mejores resultados?

Estará utilizando un sistema mononúcleo donde sea ineficiente clusterizar ya que Node.js se ejecuta en un solo hilo.