

MEMORIA EII – PRÁCTICA III

Alonso Rodríguez Iglesias – EII 2020/21 – Práctica 3

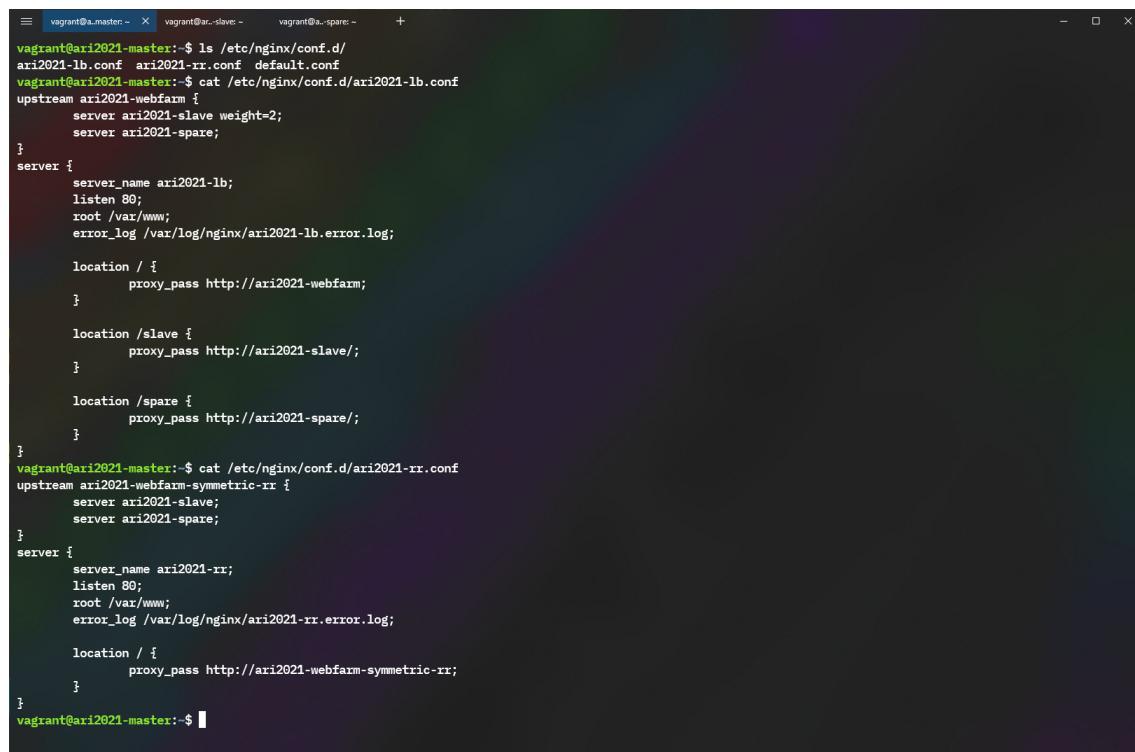
JUSTIFICACIÓN DE LA PRIMERA PARTE

Se han modificado los **index.html** de **slave** y **spare** para que al lado de **It works!** esté destacado entre paréntesis el dispositivo que sirve la página. Así, si se accede a **slave** pondrá **It works! (SLAVE)**, y lo mismo con **(SPARE)** para el **spare**.

Se adjuntan capturas de la ejecución (en orden) de los siguientes comandos:

En este apartado, me sacaba la configuración por defecto también, así que he preferido mostrar las configuraciones por separado, ya que creo que es mucho más legible, y además me cabe todo en una sola captura.

```
ls /etc/nginx/conf.d/  
cat /etc/nginx/conf.d/ari2021-lb.conf  
cat /etc/nginx/conf.d/ari2021-rr.conf
```



A terminal window showing the execution of three nginx configuration files. The first command lists the files in the directory. The second command displays the contents of the 'ari2021-lb.conf' file, which defines an upstream 'ari2021-webfarm' containing two servers: 'ari2021-slave' and 'ari2021-spare'. The third command displays the contents of the 'ari2021-rr.conf' file, which defines an upstream 'ari2021-webfarm-symmetric-rr' containing two servers: 'ari2021-slave' and 'ari2021-spare'. Both files include a server block for port 80, root directory /var/www, and error log /var/log/nginx/ari2021-[lb/rr].error.log.

```
vagrant@ari2021-master:~$ ls /etc/nginx/conf.d/  
ari2021-lb.conf  ari2021-rr.conf  default.conf  
vagrant@ari2021-master:~$ cat /etc/nginx/conf.d/ari2021-lb.conf  
upstream ari2021-webfarm {  
    server ari2021-slave weight=2;  
    server ari2021-spare;  
}  
server {  
    server_name ari2021-lb;  
    listen 80;  
    root /var/www;  
    error_log /var/log/nginx/ari2021-lb.error.log;  
    location / {  
        proxy_pass http://ari2021-webfarm;  
    }  
    location /slave {  
        proxy_pass http://ari2021-slave/;  
    }  
    location /spare {  
        proxy_pass http://ari2021-spare/;  
    }  
}  
vagrant@ari2021-master:~$ cat /etc/nginx/conf.d/ari2021-rr.conf  
upstream ari2021-webfarm-symmetric-rr {  
    server ari2021-slave;  
    server ari2021-spare;  
}  
server {  
    server_name ari2021-rr;  
    listen 80;  
    root /var/www;  
    error_log /var/log/nginx/ari2021-rr.error.log;  
    location / {  
        proxy_pass http://ari2021-webfarm-symmetric-rr;  
    }  
}  
vagrant@ari2021-master:~$
```

```
10x curl http://ari2021-lb 2> /dev/null | grep "It works!"
1x curl http://ari2021-lb/slave 2> /dev/null | grep "It works!"
1x curl http://ari2021-lb/spare 2> /dev/null | grep "It works!"
```

```
vagrant@ari2021-master:~$ curl http://ari2021-lb 2> /dev/null | grep "It works!"
  It works! (SLAVE)
vagrant@ari2021-master:~$ curl http://ari2021-lb 2> /dev/null | grep "It works!"
  It works! (SLAVE)
vagrant@ari2021-master:~$ curl http://ari2021-lb 2> /dev/null | grep "It works!"
  It works! (SPARE)
vagrant@ari2021-master:~$ curl http://ari2021-lb 2> /dev/null | grep "It works!"
  It works! (SLAVE)
vagrant@ari2021-master:~$ curl http://ari2021-lb 2> /dev/null | grep "It works!"
  It works! (SLAVE)
vagrant@ari2021-master:~$ curl http://ari2021-lb 2> /dev/null | grep "It works!"
  It works! (SPARE)
vagrant@ari2021-master:~$ curl http://ari2021-lb 2> /dev/null | grep "It works!"
  It works! (SLAVE)
vagrant@ari2021-master:~$ curl http://ari2021-lb 2> /dev/null | grep "It works!"
  It works! (SLAVE)
vagrant@ari2021-master:~$ curl http://ari2021-lb 2> /dev/null | grep "It works!"
  It works! (SPARE)
vagrant@ari2021-master:~$ curl http://ari2021-lb 2> /dev/null | grep "It works!"
  It works! (SLAVE)
vagrant@ari2021-master:~$ curl http://ari2021-lb/slave 2> /dev/null | grep "It works!"
  It works! (SPARE)
vagrant@ari2021-master:~$ curl http://ari2021-lb/spare 2> /dev/null | grep "It works!"
  It works! (SPARE)
vagrant@ari2021-master:~$
```

(Como podemos ver, alterna entre dos conexiones al **slave** y una al **spare**, y si accedemos a **/slave** o a **/spare** nos redirige al correcto)

```
6x curl http://ari2021-rr 2> /dev/null | grep "It works!"
```

```
vagrant@ari2021-master:~$ curl http://ari2021-rr 2> /dev/null | grep "It works!"
  It works! (SLAVE)
vagrant@ari2021-master:~$ curl http://ari2021-rr 2> /dev/null | grep "It works!"
  It works! (SPARE)
vagrant@ari2021-master:~$ curl http://ari2021-rr 2> /dev/null | grep "It works!"
  It works! (SLAVE)
vagrant@ari2021-master:~$ curl http://ari2021-rr 2> /dev/null | grep "It works!"
  It works! (SPARE)
vagrant@ari2021-master:~$ curl http://ari2021-rr 2> /dev/null | grep "It works!"
  It works! (SLAVE)
vagrant@ari2021-master:~$ curl http://ari2021-rr 2> /dev/null | grep "It works!"
  It works! (SPARE)
vagrant@ari2021-master:~$ curl http://ari2021-rr 2> /dev/null | grep "It works!"
  It works! (SLAVE)
vagrant@ari2021-master:~$ curl http://ari2021-rr 2> /dev/null | grep "It works!"
  It works! (SPARE)
vagrant@ari2021-master:~$
```

(Como vemos aquí, el balanceador por round robin puro, distribuye alternativamente una petición a cada servidor, por lo que se nos intercalan, primero una a **slave** y otra a **spare**, repetidamente)

JUSTIFICACIÓN DE LA SEGUNDA PARTE

Tablas para accesos a **ari2021-lb**:

Parámetros: n=10000 c=1												
Ejecución	Tiempo total	req/s	Velocidad de transferencia	Tiempo de conexión total			Porcentaje de solicitudes					
				Min	Med	Max	50	75	90	95	98	100
1	5,685	1759,05	19230,88	0	1	9	1	1	1	1	1	9
2	5,588	1789,42	19562,95	0	1	5	1	1	1	1	1	5
3	5,708	1751,98	19153,62	0	1	9	1	1	1	1	1	9
4	5,925	1777,63	19434,05	0	1	14	1	1	1	1	1	14
5	5,565	1796,82	19643,85	0	1	5	1	1	1	1	1	5
6	5,827	1716,22	18762,65	0	1	11	1	1	1	1	1	11
7	5,652	1769,14	19341,27	0	1	4	1	1	1	1	1	4
8	5,633	1775,18	19407,24	0	1	11	1	1	1	1	1	11
9	5,633	1775,28	19408,33	0	1	10	1	1	1	1	1	10
10	5,641	1772,85	19381,81	0	1	11	1	1	1	1	1	11
Media	5,69	1768,36	19332,67	0,00	1,00	8,90	1,00	1,00	1,00	1,00	1,00	8,90
Desviación	0,11	22,43	245,23	0,00	0,00	3,25	0,00	0,00	0,00	0,00	0,00	3,25

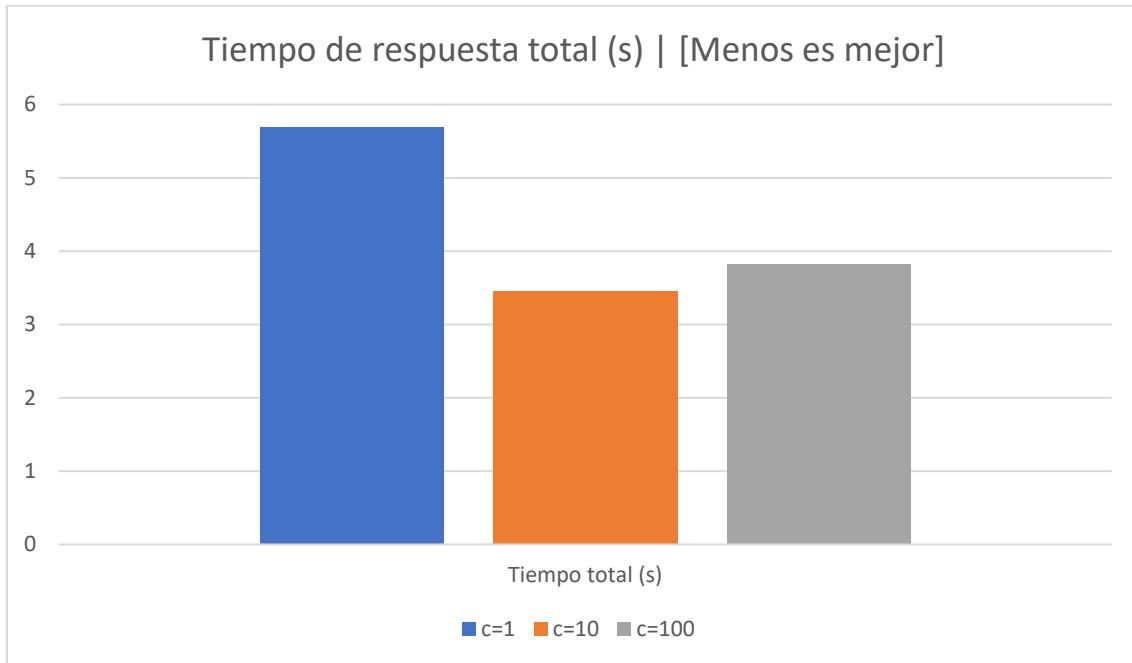
Parámetros: n=10000 c=10												
Ejecución	Tiempo total	req/s	Velocidad de transferencia	Tiempo de conexión total			Porcentaje de solicitudes					
				Min	Med	Max	50	75	90	95	98	100
1	3,447	2901,15	31716,99	1	3	12	3	4	4	4	5	12
2	3,472	2879,89	31484,64	1	3	13	3	4	4	4	5	13
3	3,442	2905,51	31764,64	1	3	10	3	4	4	4	5	10
4	3,465	2886,25	31554,18	1	3	12	3	4	4	4	5	12
5	3,44	2906,68	31777,5	1	3	9	3	4	4	4	5	9
6	3,434	2912,04	31836,16	1	3	11	3	4	4	4	5	11
7	3,411	2932,03	32054,66	1	3	14	3	4	4	4	4	14
8	3,459	2899,97	31605,75	1	3	13	3	4	4	4	5	13
9	3,456	2893,34	31631,59	1	3	15	3	4	4	4	5	15
10	3,439	2907,41	31785,42	1	3	10	3	4	4	4	5	10
Media	3,45	2901,53	31721,15	1,00	3,00	11,90	3,00	4,00	4,00	4,00	4,90	11,90
Desviación	0,02	14,90	162,90	0,00	0,00	1,91	0,00	0,00	0,00	0,00	0,32	1,91

Parámetros: n=10000 c=100												
Ejecución	Tiempo total	req/s	Velocidad de transferencia	Tiempo de conexión total			Porcentaje de solicitudes					
				Min	Med	Max	50	75	90	95	98	100
1	3,84	2604,04	28468,91	11	38	50	38	39	40	41	44	50
2	3,824	2614,78	28586,26	7	38	50	38	39	39	40	44	50
3	3,809	2625,2	28700,18	7	38	55	38	39	40	40	43	55
4	3,803	2629,49	28747,05	7	38	53	38	39	39	40	41	53
5	3,846	2600,02	28424,89	7	38	56	38	39	40	40	45	56
6	3,866	2586,82	28280,58	7	39	60	38	39	40	41	44	60
7	3,801	2630,55	28758,72	15	38	51	38	39	39	40	43	51
8	3,816	2620,29	28646,48	7	38	52	38	39	39	40	42	52
9	3,806	2627,4	28724,28	7	38	52	38	39	39	40	42	52
10	3,828	2612,14	28557,42	10	38	54	38	39	39	40	44	54
Media	3,82	2615,07	28589,48	8,50	38,10	53,30	38,00	39,00	39,40	40,20	43,20	53,30
Desviación	0,02	14,47	158,19	2,72	0,32	3,09	0,00	0,00	0,52	0,42	1,23	3,09

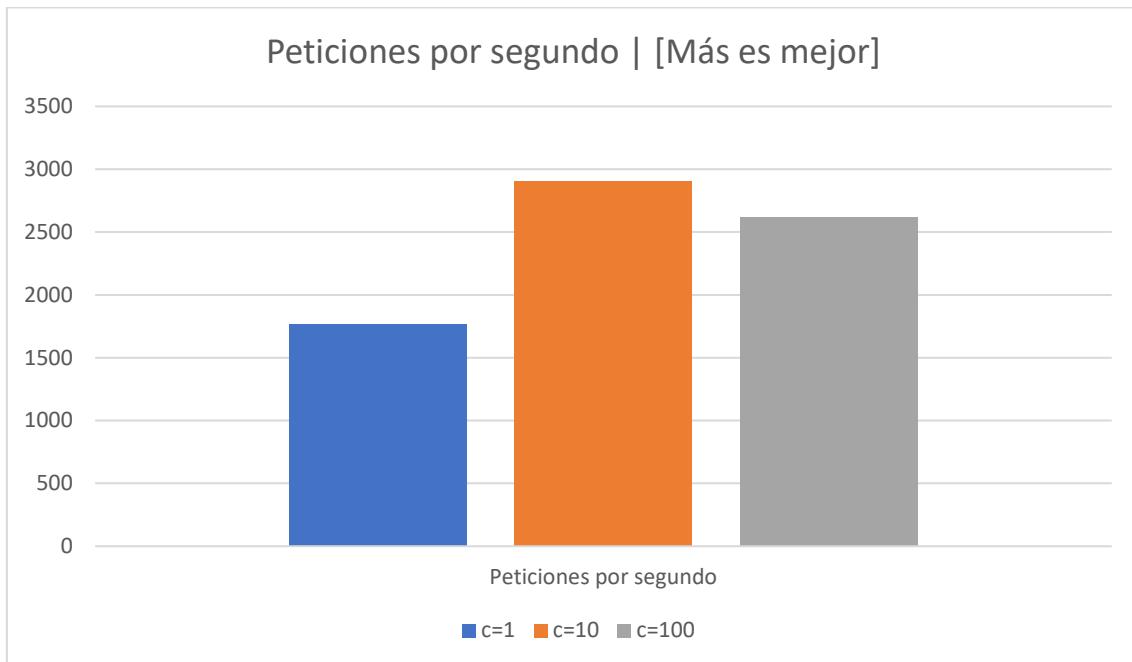
Como podemos apreciar, el mejor caso de estos tres es con **c=10**, ya que es con el que obtenemos un menor tiempo, y por tanto (ya que son inversamente proporcionales), una mayor cantidad de peticiones por segundo, así como una mayor velocidad de transferencia.

Se adjuntan gráficas a continuación:

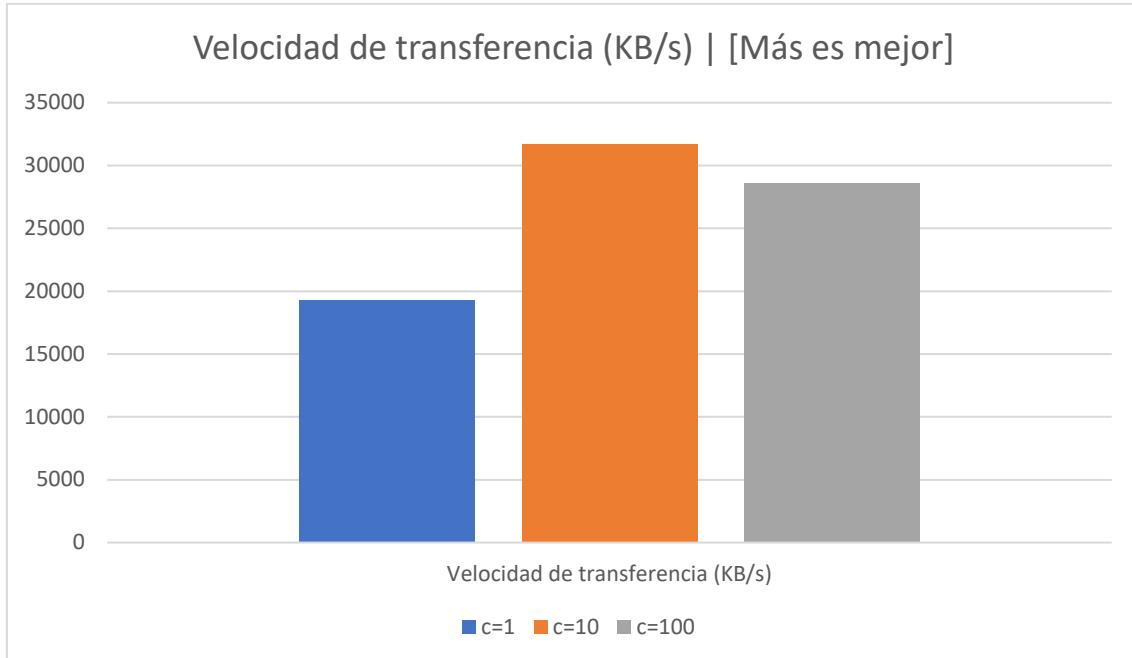
1. Tiempo total



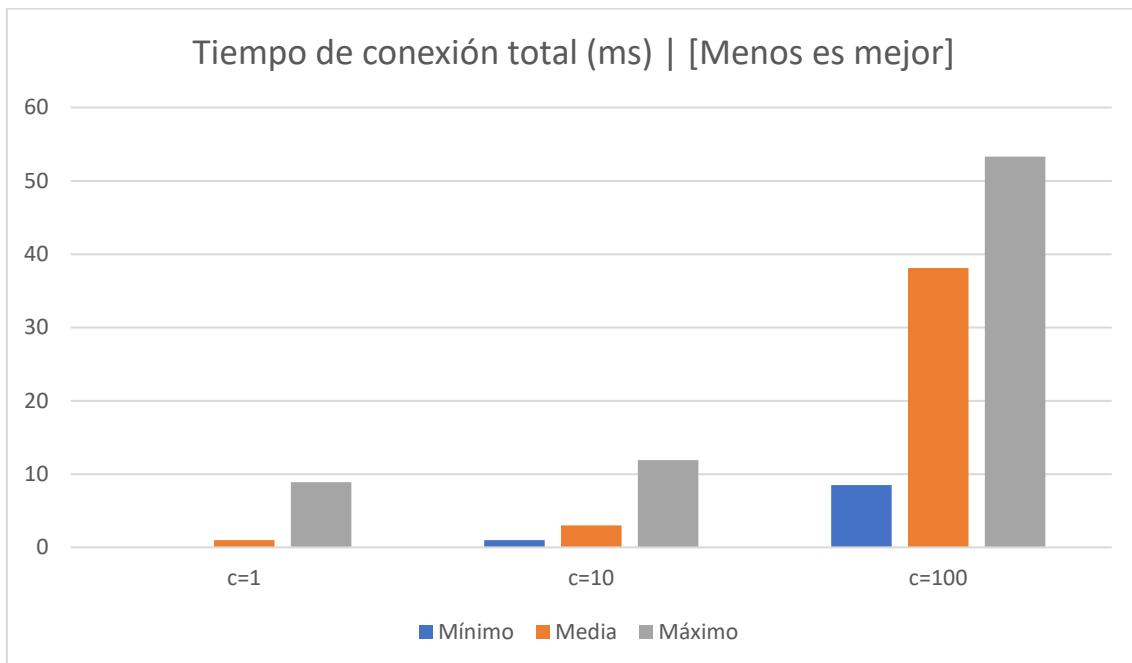
2. Peticiones por segundo



3. Velocidad de transferencia

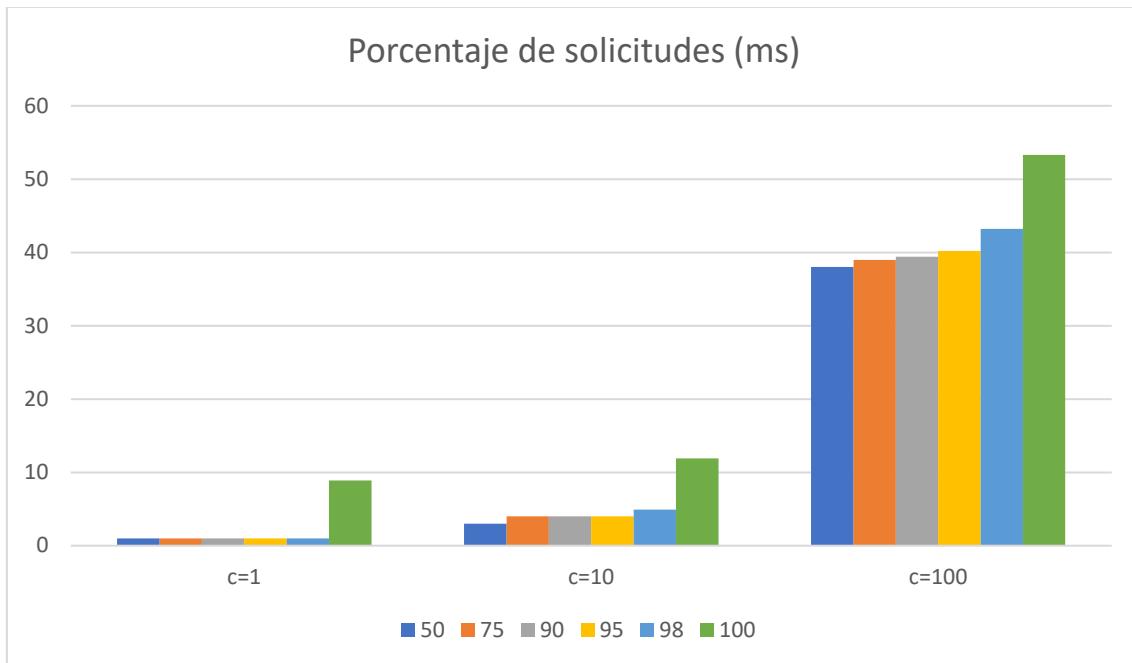


4. Tiempo de conexión total



En esta situación es cierto que el tiempo de respuesta es menor para $c=1$, pero como explico más adelante, sacrificamos ancho de banda total por esta mejora con respecto a $c=10$.

5. Porcentaje de solicitudes



Como podemos ver en todas las gráficas anteriores, y como conclusión, el mejor balance está en las 10 peticiones concurrentes, ya que muy probablemente lo que está pasando es que los recursos necesarios para responder 100 peticiones simultáneas, y el overhead vinculado a ellos está ralentizando el tiempo de respuesta de esas 100 peticiones concurrentes. Esto posiblemente cambiaría si tuviésemos, por ejemplo, 10 núcleos, o 100. En nuestro caso como solo tenemos uno por máquina, el caso de $c=1$ no es el mejor por desaprovechamiento de recursos, pero el de $c=100$ provoca demasiado overhead.

El 100% nos indica el tiempo de respuesta del peor caso. Si nos fijamos, éste coincide con el valor del tiempo de conexión total máximo.

A partir de ahí, podemos ver cómo baja. Esto sería algo como “de cada 100 clientes que se conectan, uno va a tener {100%}ms, dos de cada 100 van a tener {98%}ms, etc...”

Me parece una medida de lo más útil, pero también es cierto que tuve que hacer cierta búsqueda de información, ya que la página web oficial no aclara muy bien lo que es, a pesar de que se intuye que es una tabla de probabilidad acumulada.