A24 Balanceo de carga y cluster de procesos

El Round Robin, es un algoritmo de planificación de procesos más simples dentro de un sistema operativo que asigna a cada proceso una porción de tiempo equitativa y ordenada.

Ventajas

- Equitativo.
- Fácil de implementar.

Desventajas

- Normalmente el tiempo de retorno medio es mayor que en SJF (prioridad al más corto), pero el tiempo de respuesta es mejor.

**Ejercicio 1**

```
upstream backend  {
  server 51.254.116.159:3000;
  server 51.254.116.159:3001;

}

server {

        root /var/www/example.com/html;
        index index.html index.htm index.nginx-debian.html;

        server_name dpl.ivandaw.es www.ivandaw.es;

         location / {
 #               try_files $uri $uri/ =404;
                proxy_pass http://backend;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection 'upgrade';
                proxy_set_header Host $host;
                proxy_cache_bypass $http_upgrade;
        }

        location /app2 {
#               try_files $uri $uri/ =404;
                proxy_pass http://51.254.116.159:3001;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection 'upgrade';
                proxy_set_header Host $host;
                proxy_cache_bypass $http_upgrade;
        }

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/dpl.ivandaw.es/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/dpl.ivandaw.es/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
}
server{
    if ($host = dpl.ivandaw.es) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


        listen 80;
        listen [::]:80;

        server_name dpl.ivandaw.es www.ivandaw.es;
    return 404; # managed by Certbot


}
```

**Configuramos nuestro server.js**

```
ivan@vps591614:~/balanceo_carga$ cat server.js
var http = require('http');

const port = process.argv[2];

http.createServer(function (req, res) {
  res.write(`Servidor corriendo en el puerto ${port}`);
  res.end();
}).listen(port);
 console.log(`Corriendo en el ${port}`);

ivan@vps591614:~/balanceo_carga$
```

**Arrancamos nuestro server.js**

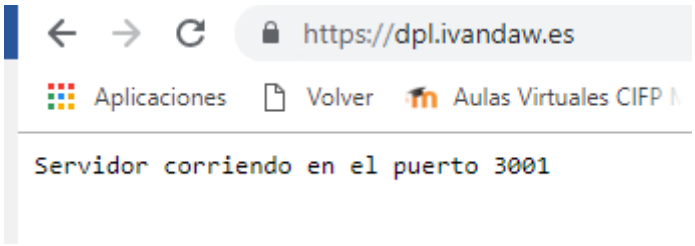
```
ivan@vps591614:~/balanceo_carga$ node server.js 3000
Corriendo en el 3000
^C
ivan@vps591614:~/balanceo_carga$ node server.js 3000 &
[1] 2793
ivan@vps591614:~/balanceo_carga$ Corriendo en el 3000

ivan@vps591614:~/balanceo_carga$ node server.js 3001 &
[2] 2802
```

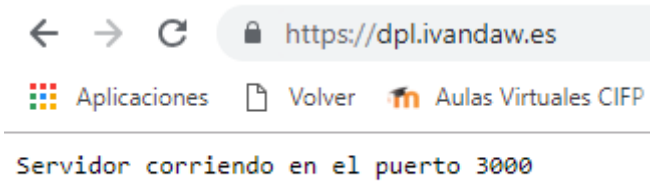
**Observamos como se ejecutan en los puertos**

```
ivan@vps591614:~/balanceo_carga$ netstat -ntl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:27017         0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:443             0.0.0.0:*               LISTEN
tcp        0      0 51.254.116.159:8001     0.0.0.0:*               LISTEN
tcp        0      0 51.254.116.159:8002     0.0.0.0:*               LISTEN
tcp        0      0 51.254.116.159:8003     0.0.0.0:*               LISTEN
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::3000                 :::*                    LISTEN
tcp6       0      0 :::3001                 :::*                    LISTEN
tcp6       0      0 :::443                  :::*                    LISTEN
ivan@vps591614:~/balanceo_carga$
```

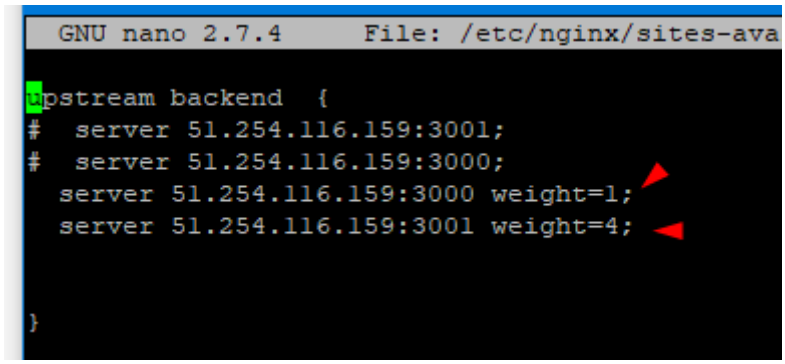
**Probamos en el navegador**



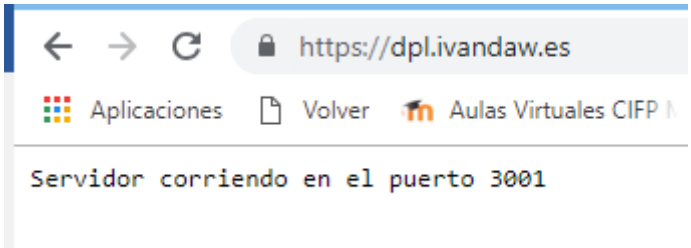

**Después de varias solicitudes, nos ha corrido en el 3000**

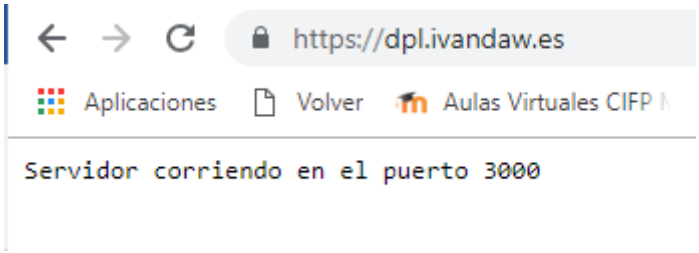



**Probamos la directiva weight**




**Observamos que al hacer una solicitud nos lleva al 3001 y al hacer 4 solicitudes nos lleva al 3000**



**Directiva weight**

**Nos permite dar más peso a ciertas máquinas, de esta manera nginx nos permite asignar un numero, que especifica la proporción de tráfico que debe dirigirse a cada servidor.**

**Directiva ip_hash**

**Esta directiva permite a los servidores responder a los clientes de acuerdo con su dirección IP, enviando a los visitantes al mismo vps cada vez que lo visitan. Si un servidor está inactivo, se marca como inactivo y las direcciones IP que se enrutaban al servidor inactivo se dirigen a otro disponible.**

**Directiva least-connected**

**Está directiva permite que la siguiente solicitud que llegue, se asigne al servidor con el menor número de conexiones activas.**

**Directiva max_fails**

**La directiva max_fails establece el número de intentos fallidos consecutivos de comunicación con el servidor que deberían ocurrir durante fail_timeout.**

**Directiva fail_timeout**

**Define cuanto tiempo se marcará el servidor como fallido.**

**Ejericio 3 Cluster de procesos con PM2**

Node tiene un módulo cluster para gestionar esta situación, sin embargo PM2 aporta mejoras respecto a este módulo. ¿Cuáles son?

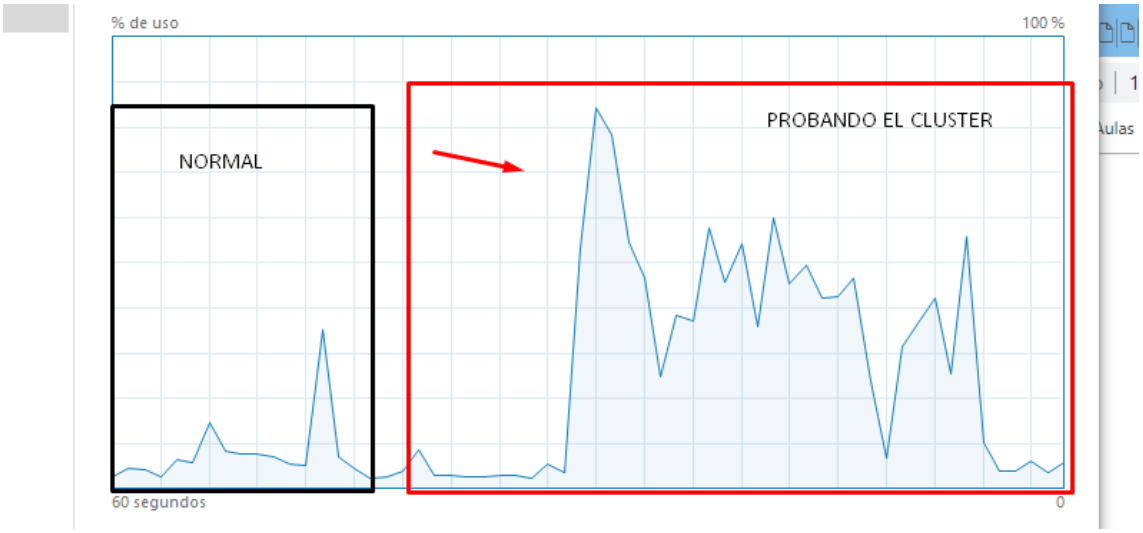
PM2 usará Node Cluster bajo su capa, esto hará las cosas mucho más fáciles, al no tener que estar manejando código.

Ejecutamos pm2 log y vemos como al cargar la página un proceso arranca y otro se desconecta.

```
PM2          | App [cluster:6] starting in -cluster mode-
PM2          | App [cluster:6] online
PM2          | App name:cluster id:7 disconnected
PM2          | App [cluster:7] exited with code [1] via signal [SIGINT]
PM2          | App [cluster:7] starting in -cluster mode-
6|cluster   | Worker 4666 started...
PM2          | App [cluster:7] online
7|cluster   | Worker 4673 started...
PM2          | App name:cluster id:5 disconnected
PM2          | App [cluster:5] exited with code [1] via signal [SIGINT]
PM2          | App [cluster:5] starting in -cluster mode-
PM2          | App [cluster:5] online
5|cluster   | Worker 4688 started...
PM2          | App name:cluster id:6 disconnected
PM2          | App [cluster:6] exited with code [1] via signal [SIGINT]
PM2          | App [cluster:6] starting in -cluster mode-
PM2          | App [cluster:6] online
6|cluster   | Worker 4699 started...
PM2          | App name:cluster id:7 disconnected
PM2          | App [cluster:7] exited with code [1] via signal [SIGINT]
PM2          | App [cluster:7] starting in -cluster mode-
PM2          | App [cluster:7] online
7|cluster   | Worker 4790 started...
```

A continuación, he probado en una máquina virtual el cluster de procesos con pm2.

Aquí tenemos una captura de pantalla del rendimiento de la CPU antes y después de su ejecución. **CPU con 4 Cores y 8 Thread**




En GitHub hay un gif de las pruebas.