CheckMed Reporte

Fecha: 30 de Junio de 2021
Autor: Santillán Marcelo
Diseño de Bases de Datos 2020

ÍNDICE

Reporte Comparación de Técnicas

Introducción:

Se realizará un análisis de la aplicación con distintas técnicas, para un sistema que genera tickets (recetas), en los cuales se validan los diagnósticos de medicamentos. Para las pruebas se va generar lotes de tickets e identificar cada lote con un número. para hacerlo equitativo serán lotes de 100 tickets y el mismo número de fibonacci como identificador de lote (de ser un requerimiento real por supuesto que no se optaría por la sucesión de fibonacci, pero si resulta representativo de una actividad con gran consumo de cpu):

```
50th Fibonacci Number: 12586269025
0
```

La idea de la comparación es evaluar cuál técnica es la más apropiada para principalmente dos situaciones, **alta actividad i/o y alta actividad de consumo de cpu**. También voy a utilizar combinaciones de estas técnicas. Para las mediciones voy a usar apache benchmark y medir el uso de cpu y los tiempos.

Voy aplicar:

- Asincronismo
- Modo Cluster
- Worker Threads

Características técnicas de la aplicación:

- Servidor Node js que cada cierto periodo de tiempo procesa información en lotes.(para las pruebas va a ejecutar una vez el script que genera tickets, aunque se puede utilizar cron para reproducir este paso cada cierto tiempo)
- Servicio Api Rest donde se procesan diversos pedidos como la creación de nuevas recetas (tickets), listados, y el crud de comentarios..
- Base de datos MongoDB para los tickets, comentarios y medicamentos.
- BackEnd node js.
- se ejecuta en un equipo con Ubuntu 20.04, 8 gb de ram y con 8 cores.
- Para el reporte voy a utilizar Apache Benchmark.

Comenzaré con tres scripts de node js para evaluar performance:

server.js	solo compartimiento asincrónico
serverClusterMode.js	aplica Cluster Mode
serverWorkerThreads.js	aplica Worker Threads

Apache Benchmark:

```
marcelo880@marcelo880-N56VB:~$ ab -V
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

marcelo880@marcelo880-N56VB:~$
```

https://bobcares.com/blog/apache-benchmark-install-ubuntu/

Asincronismo

En principio se consideran que las técnicas mencionadas dado que van a realizar muchas operaciones de i/o interactuando con la base de datos, creando archivos json, etc van a utilizar asincronismo por medio de callbacks, promises o asyn await (se busco no abusar de esto último ya que no es compatible con todos los navegadores)

```
const http = require('http');
const app = require('./app');
const cron = require('node-cron')
const creaTickets = require('./api/utilities/crea100Tickets')

const port = process.env.PORT || 3000;

const server = http.createServer(app);
```

```
// para simular actividad se generan/validan tickets
// con datos aleatorios cada cierto tiempo
//cron.schedule('00 * * * * *', () => {
   creaTickets.generaTicketsAleatorios();
   console.log('Ejecución de nuevo lote');
//});

server.listen(port);
```

en el código se crea el server y se ejecuta un generador de tickets

Detalle de las pruebas realizadas:

1) concurrencia 8 1000 request
2) Concurrencia 16 request 1000
3) Concurrencia 8 request 2000

1)ab -c 8 -n 1000 http://localhost:3000/

Previamente en otra terminal ejecuto el server.js

```
marcelo880@marcelo880-N56VB: ~/Desktop/checkmed_v1
arcelo880@marcelo880-N56VB:~/Desktop/checkmed_v1$ node server
node:116476) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use
{ useNewUrlParser: true } to MongoClient.connect.
node:116476) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed
he new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
 'aciclovir' ]
bjeto creado: { _id: 60bd24ae6276eec6fd38d131,
  apyn: 'Heidenreich Sophia',
  nroAfiliado: '54428441',
  diagnostico: 'asma',
  idPrestador: '10002',
  prestacion: 'aciclovir',
  fechaCreacion: 2021-06-06T19:40:30.736Z,
  comments: [] }
unning a task every half minute
pta find: { medicinaArray: [ 'asmabron', 'asmavitan' ] }
 'aciclovir' ]

scribiendo ./public/files/creados/10002-Heidenreich Sophia-60bd24ae6276eec6fd38d131.json
 'aciclovir' ]
bjeto creado: { _id: 60bd24ae6276eec6fd38d134,
  apyn: 'Reichel Trevion',
  nroAfiliado: '64084451',
  diagnostico: 'herpes',
  idPrestador: '10002',
  prestacion: 'aciclovir',
  fechaCreacion: 2021-06-06T19:40:30.772Z,
  comments: [] }
scribiendo ./public/files/creados/10002-Reichel Trevion-60bd24ae6276eec6fd38d134.json
 'lopid ud', 'genfibrozil' ]
```

```
marcelo880@marcelo880-N56VB:~/Desktop/checkmed_v1$ ab -c 8 -n 1000
http://localhost:3000/
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:
Server Hostname:        localhost
Server Port:            3000

Document Path:          /
Document Length:        33 bytes

Concurrency Level:      8
Time taken for tests:   0.473 seconds
Complete requests:      1000
Failed requests:        0
Non-2xx responses:      1000
Total transferred:      372000 bytes
HTML transferred:       33000 bytes
Requests per second:    2113.22 [#/sec] (mean)
Time per request:       3.786 [ms] (mean)
Time per request:       0.473 [ms] (mean, across all concurrent requests)
Transfer rate:          767.69 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:     2    4   1.0      3      19
Waiting:        1    2   1.1      2      17
Total:          3    4   1.0      3      19
WARNING: The median and mean for the processing time are not within a normal deviation
        These results are probably not that reliable.
WARNING: The median and mean for the total time are not within a normal deviation
```

```
            These results are probably not that reliable.

Percentage of the requests served within a certain time (ms)
  50%    3
  66%    4
  75%    4
  80%    4
  90%    5
  95%    6
  98%    6
  99%    7
 100%   19 (longest request)
marcelo880@marcelo880-N56VB:~/Desktop/checkmed_v1$
```

2) Concurrencia 16 request 1000

```
marcelo880@marcelo880-N56VB:~/Desktop/checkmed_v1$ ab -c 16 -n 1000
http://localhost:3000/
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:
Server Hostname:       localhost
Server Port:           3000

Document Path:         /
Document Length:       33 bytes

Concurrency Level:     16
```

```
Time taken for tests:   0.406 seconds
Complete requests:      1000
Failed requests:        0
Non-2xx responses:      1000
Total transferred:      372000 bytes
HTML transferred:       33000 bytes
Requests per second:    2460.77 [#/sec] (mean)
Time per request:       6.502 [ms] (mean)
Time per request:       0.406 [ms] (mean, across all concurrent requests)
Transfer rate:          893.95 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       1
Processing:     2    6   1.9      6      17
Waiting:        1    4   1.7      4      16
Total:          3    6   2.0      6      18

Percentage of the requests served within a certain time (ms)
  50%    6
  66%    6
  75%    7
  80%    7
  90%    8
  95%    9
  98%    13
  99%    18
 100%    18 (longest request)
marcelo880@marcelo880-N56VB:~/Desktop/checkmed_v1$
```

3) Concurrencia 8 request 2000

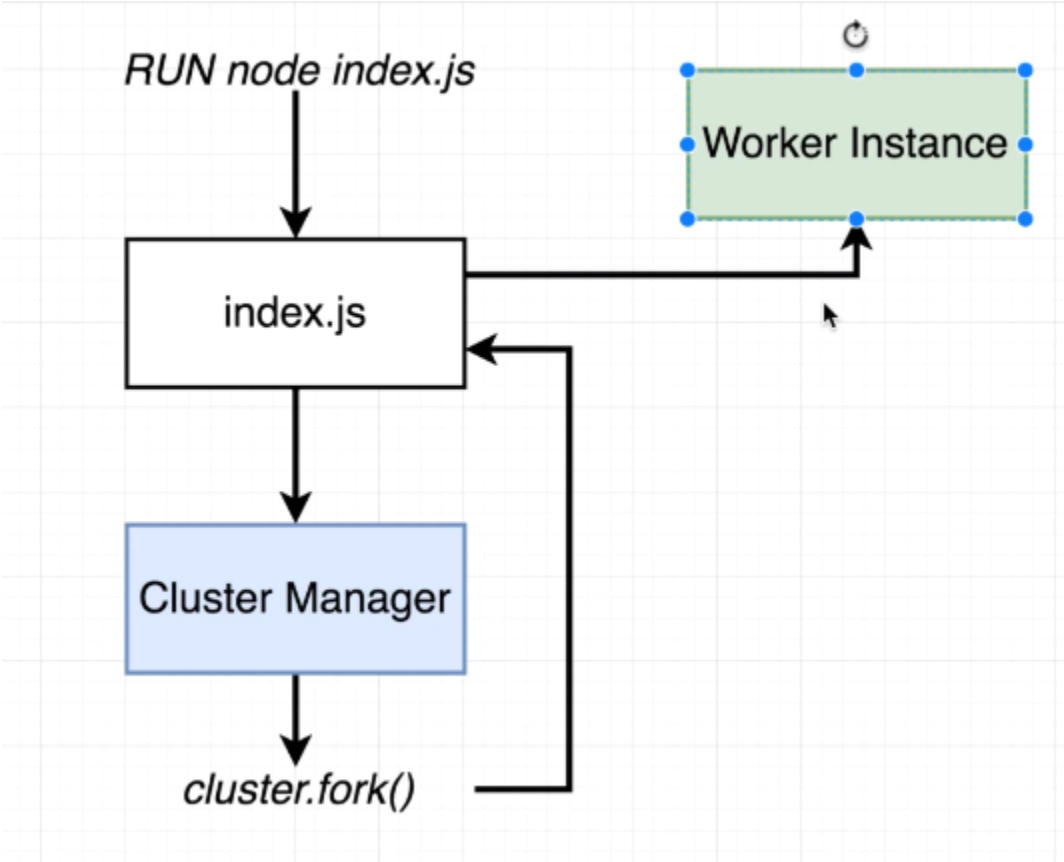
```
marcelo880@marcelo880-N56VB:~/Desktop/checkmed_v1$ ab -c 8 -n 2000
http://localhost:3000/
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 200 requests
```

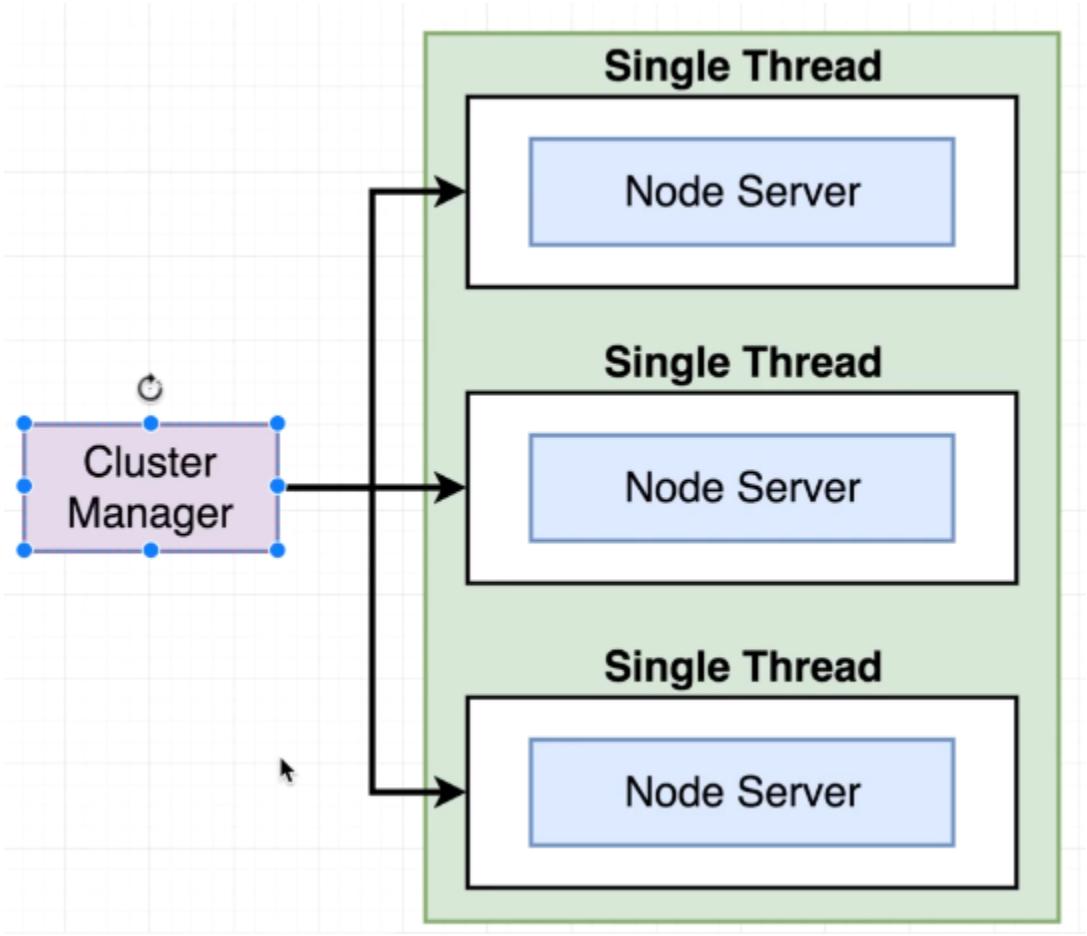
```
Completed 400 requests
Completed 600 requests
Completed 800 requests
Completed 1000 requests
Completed 1200 requests
Completed 1400 requests
Completed 1600 requests
Completed 1800 requests
Completed 2000 requests
Finished 2000 requests


Server Software:
Server Hostname:        localhost
Server Port:            3000

Document Path:          /
Document Length:        33 bytes

Concurrency Level:      8
Time taken for tests:   0.659 seconds
Complete requests:      2000
Failed requests:        0
Non-2xx responses:      2000
Total transferred:      744000 bytes
HTML transferred:       66000 bytes
Requests per second:    3033.19 [#/sec] (mean)
Time per request:       2.637 [ms] (mean)
Time per request:       0.330 [ms] (mean, across all concurrent requests)
Transfer rate:          1101.90 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       1
Processing:     1    2   0.7      2       7
Waiting:        0    1   0.7      1       6
Total:          1    3   0.7      2       7
WARNING: The median and mean for the total time are not within a normal deviation
        These results are probably not that reliable.

Percentage of the requests served within a certain time (ms)
  50%    2
  66%    3
  75%    3
  80%    3
  90%    3
  95%    4
  98%    6
```

```
 99%   6
100%   7 (longest request)
marcelo880@marcelo880-N56VB:~/Desktop/checkmed_v1$
```

Cluster Mode

El cluster manager es responsable de iniciar las instancias (worker instance), mediante la función cluster.fork() node internamente ejecuta el script por segunda vez en las worker instancias, es decir es ejecutado múltiple veces por node js. la primera vez en Cluster manager, y las siguientes veces en worker instances.



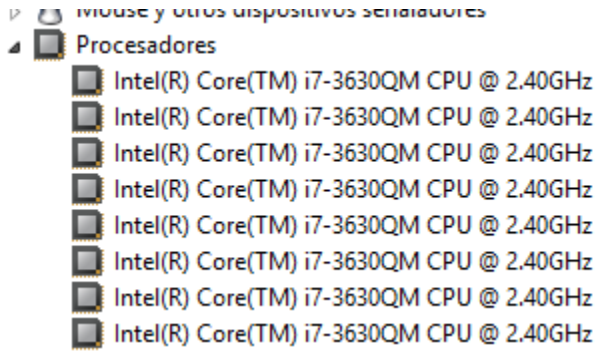


tiene que existir una relación entre cantidad de instancias y cores del equipo. El equipo donde desarrollé la aplicación contiene 8 cores (asus i7), se puede verificar mediante estas líneas de código de node js:

```
JS index1.js > ...
1   //Cuantos cores tiene el equipo
2   const { cpus }    = require('os')
3   const numWorkers = cpus().length
4   console.log("cantidad de cores: " + numWorkers);
5
```

```
C:\Users\marcelo880\Desktop\arqNode\clase1
λ node index1
8
```

o en la carpeta equipo-procesadores:



Código:

serverClusterMode.js

```
//Versión con cluster mode
const http = require('http');
const app = require('./app');
const cron = require('node-cron')
const creaTickets = require('./api/utilities/crea100Tickets')
const port = process.env.PORT || 3000;
const server = http.createServer(app);
//----------------
const { cpus } = require('os')
const numCPUs = cpus().length
console.log("cantidad de cores: " + numCPUs);
const cluster = require('cluster');

if (cluster.isMaster) {
   console.log(`Master ${process.pid} is running`)

   for (let i = 0; i < numCPUs; i++) {
       cluster.fork()
   }
   cluster.on('exit', (worker, code, signal) => {
       console.log(`worker ${worker.process.pid} died`)
   })
```

```
} else {

   // para simular actividad se generan/validan tickets
   // con datos aleatorios cada cierto tiempo
 //  cron.schedule('00 * * * * *', () => {

       creaTickets.generaTicketsAleatorios();
       console.log('Ejecucion de nuevo lote');

  // });

   server.listen(port);
   console.log(`Process ${process.pid} started`)
}
```

Las pruebas se realizan cuando el cron ejecuta el módulo que genera tickets con datos aleatorios, y a la vez se van validando y se agregan a la base de datos (Para el caso comente esa posibildad para que se ejecute una sola vez, pero puede descomentarse para que el cron lo vuelva a correr cierto tiempo)

y también es posible para simular un frontend usar postman donde se ejecutan request

y el script va guardando los json en un carpeta:

Home | Desktop | checkmed_v1 | public | files | creados

Name	Size
10004-Oberbrunner Alverta-60beca8e418790527c23c29b.json	230 bytes
10004-Gulgowski Adaline-60beca52418790527c23c207.json	228 bytes
10001-Windler Broderick-60beca8e418790527c23c25e.json	228 bytes
10001-MacGyver Claudine-60becb7f418790527c23c456.json	228 bytes
10004-Herzog Laurianne-60becb43418790527c23c3d3.json	227 bytes
10003-Reichert Eugenia-60becb42418790527c23c3a2.json	227 bytes
10004-Leffler Christa-60becb7f418790527c23c465.json	226 bytes
10004-Casper Jackson-60becb7e418790527c23c419.json	225 bytes
10003-Grant Vincenzo-60bec872b69cb4fd9e02dfb8.json	225 bytes
10002-Shields Hobart-60bec873b69cb4fd9e02dffd.json	225 bytes
10004-Sanford Zechariah-60beca8e418790527c23c28a.json	224 bytes
10002-Swift Domingo-60becb07418790527c23c364.json	224 bytes
10002-Reilly Angelo-60bec873b69cb4fd9e02e020.json	224 bytes
10001-Roberts Emily-60becb7f418790527c23c432.json	224 bytes

Soporte de pm2: Alternativamente se podría utilizar esta herramienta para el monitoreo, que me pareció interesante pero que no está relacionada con las pruebas evaluadas..

https://www.vultr.com/docs/how-to-setup-pm2-on-ubuntu-16-04

https://pm2.keymetrics.io/docs/usage/quick-start/

Detalles de las pruebas realizadas

Mediciones con Apache Benchmark

En una instancia de terminal de ubuntu ejecuto serverClusterMode.js, donde se puede apreciar que levantó las 8 instancias, una por cada core:

```
Activities   Firefox Web Browser   dom jun 6  19:32   100 %
marcelo880@marcelo880-N56VB: ~/Desktop/checkmed_v1
marcelo880@marcelo880-N56VB:~/Desktop/checkmed_v1$ node serverClusterMode
cantidad de cores: 8
Master 122146 is running
(node:122146) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass optio
n { useNewUrlParser: true } to MongoClient.connect.
(node:122146) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use
 the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
cantidad de cores: 8
Process 122188 started
(node:122188) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass optio
n { useNewUrlParser: true } to MongoClient.connect.
(node:122188) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use
 the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
cantidad de cores: 8
Process 122176 started
(node:122176) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass optio
n { useNewUrlParser: true } to MongoClient.connect.
(node:122176) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use
 the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
cantidad de cores: 8
Process 122182 started
(node:122182) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass optio
n { useNewUrlParser: true } to MongoClient.connect.
(node:122182) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use
 the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
cantidad de cores: 8
Process 122158 started
(node:122158) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass optio
n { useNewUrlParser: true } to MongoClient.connect.
(node:122158) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use
 the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
cantidad de cores: 8
Process 122157 started
(node:122157) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass optio
n { useNewUrlParser: true } to MongoClient.connect.
(node:122157) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use
 the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
```

Y en otra terminal ejecuto el Benchmark:

```
Activities   Terminal   dom jun 6  19:39   100 %
marcelo880@marcelo880-N56VB: ~/Desktop/checkmed_v1
escribiendo ./public/files/creados/10004-Beer Lane-66bd4e67b935ecdd41343345.json
{
  { asnabron }
objeto creado: { _id: 66bd4e67b935ecdd41343348,
  apyn: 'Luellwitz Lucy',
  nroAfiliado: '34395491',
  diagnostico: 'herpes',
  idPrestador: '10002',
  prestacion: 'asnabron',
  fechaCreacion: 2021-06-06T22:38:31.256Z,
  comments: [] }
escribiendo ./public/files/creados/10003-Hand Zion-66bd4e670b8be5dd2f3c6339.json
{ lapid ud , genfibrozil }
objeto creado: { _id: 66bd4e670b8be5dd2f3c633a,
  apyn: 'Feeney Rickey',
  nroAfiliado: '15174373',
  diagnostico: 'herpes',
  idPrestador: '10001',
  prestacion: 'lapid ud,genfibrozil',
  fechaCreacion: 2021-06-06T22:38:31.257Z,
  comments: [] }
gen result: { _id: 66bd4e67b935ecdd41343345,
  apyn: 'Beer Lane',
  nroAfiliado: '65028944',
  diagnostico: 'distipema',
  idPrestador: '10004',
  prestacion: 'herpial',
  fechaCreacion: 2021-06-06T22:38:31.250Z,
  comments:
   [ { _id: 66bd4e67b935ecdd41343346,
       whoComment: 'Checkmed',
       textBody: ' Verifique la medicacion, por favor',
       dateComment: 2021-06-06T22:38:31.253Z } ],
  version: 0 }
rpta find: { medicinaArray: [ 'aciclovir' , 'herpial' ] }
{ asnabron }
gen result: { _id: 66bd4e670b8be5dd2f3c6339,
  apyn: 'Hand Zion',
```

```
marcelo880@marcelo880-N56VB: ~/Desktop/checkmed_v1
marcelo880@marcelo880-N56VB:~/Desktop/checkmed_v1$ ab -c 8 -n 1000 http://localhost:3000/
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:
Server Hostname:        localhost
Server Port:            3000

Document Path:          /
Document Length:        33 bytes

Concurrency Level:      8
Time taken for tests:   0.733 seconds
Complete requests:      1000
Failed requests:        0
Non-2xx responses:      1000
Total transferred:      372000 bytes
HTML transferred:       33000 bytes
Requests per second:    1365.00 [#/sec] (mean)
Time per request:       5.861 [ms] (mean)
Time per request:       0.733 [ms] (mean, across all concurrent requests)
Transfer rate:          495.80 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
```

marcelo880@marcelo880-N56VB:~/Desktop/checkmed_v1$ ab -c 8 -n 1000 http://localhost:3000/

1) concurrencia 8 1000 request
2) Concurrencia 16 request 1000
3) Concurrencia 8 request 2000

1) concurrencia 8 1000 request

```
arcelo880@marcelo880-N56VB:~/Desktop/checkmed_v1$ ab -c 8 -n 1000
http://localhost:3000/
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:
Server Hostname:       localhost
Server Port:           3000

Document Path:         /
Document Length:       33 bytes

Concurrency Level:     8
Time taken for tests:  0.278 seconds
Complete requests:     1000
Failed requests:       0
```

```
Non-2xx responses:   1000
Total transferred:   372000 bytes
HTML transferred:    33000 bytes
Requests per second: 3595.21 [#/sec] (mean)
Time per request:    2.225 [ms] (mean)
Time per request:    0.278 [ms] (mean, across all concurrent requests)
Transfer rate:       1306.07 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       0
Processing:     1    2   1.9      2      29
Waiting:        1    2   1.5      2      22
Total:          1    2   1.9      2      29

Percentage of the requests served within a certain time (ms)
  50%      2
  66%      2
  75%      2
  80%      2
  90%      3
  95%      3
  98%      4
  99%      7
 100%     29 (longest request)
```

2) Concurrencia 16 request 1000

```
marcelo880@marcelo880-N56VB:~/Desktop/checkmed_v1$ ab -c 16 -n 1000
http://localhost:3000/
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
```

```
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:
Server Hostname:        localhost
Server Port:            3000

Document Path:          /
Document Length:        33 bytes

Concurrency Level:      16
Time taken for tests:   0.304 seconds
Complete requests:      1000
Failed requests:        0
Non-2xx responses:      1000
Total transferred:      372000 bytes
HTML transferred:       33000 bytes
Requests per second:    3292.05 [#/sec] (mean)
Time per request:       4.860 [ms] (mean)
Time per request:       0.304 [ms] (mean, across all concurrent requests)
Transfer rate:          1195.94 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       1
Processing:     1    5   3.5      4      28
Waiting:        0    4   2.9      3      28
Total:          1    5   3.5      4      28

Percentage of the requests served within a certain time (ms)
  50%    4
  66%    5
  75%    6
  80%    7
  90%    9
  95%    13
  98%    16
  99%    18
 100%    28 (longest request)
marcelo880@marcelo880-N56VB:~/Desktop/checkmed_v1$
```

3) Concurrencia 8 request 2000

```
marcelo880@marcelo880-N56VB:~/Desktop/checkmed_v1$ ab -c 8 -n 2000
http://localhost:3000/
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 200 requests
Completed 400 requests
Completed 600 requests
Completed 800 requests
Completed 1000 requests
Completed 1200 requests
Completed 1400 requests
Completed 1600 requests
Completed 1800 requests
Completed 2000 requests
Finished 2000 requests


Server Software:
Server Hostname:        localhost
Server Port:            3000

Document Path:          /
Document Length:        33 bytes

Concurrency Level:      8
Time taken for tests:   0.596 seconds
Complete requests:      2000
Failed requests:        0
Non-2xx responses:      2000
Total transferred:      744000 bytes
HTML transferred:       66000 bytes
Requests per second:    3355.39 [#/sec] (mean)
Time per request:       2.384 [ms] (mean)
Time per request:       0.298 [ms] (mean, across all concurrent requests)
Transfer rate:          1218.95 [Kbytes/sec] received
```

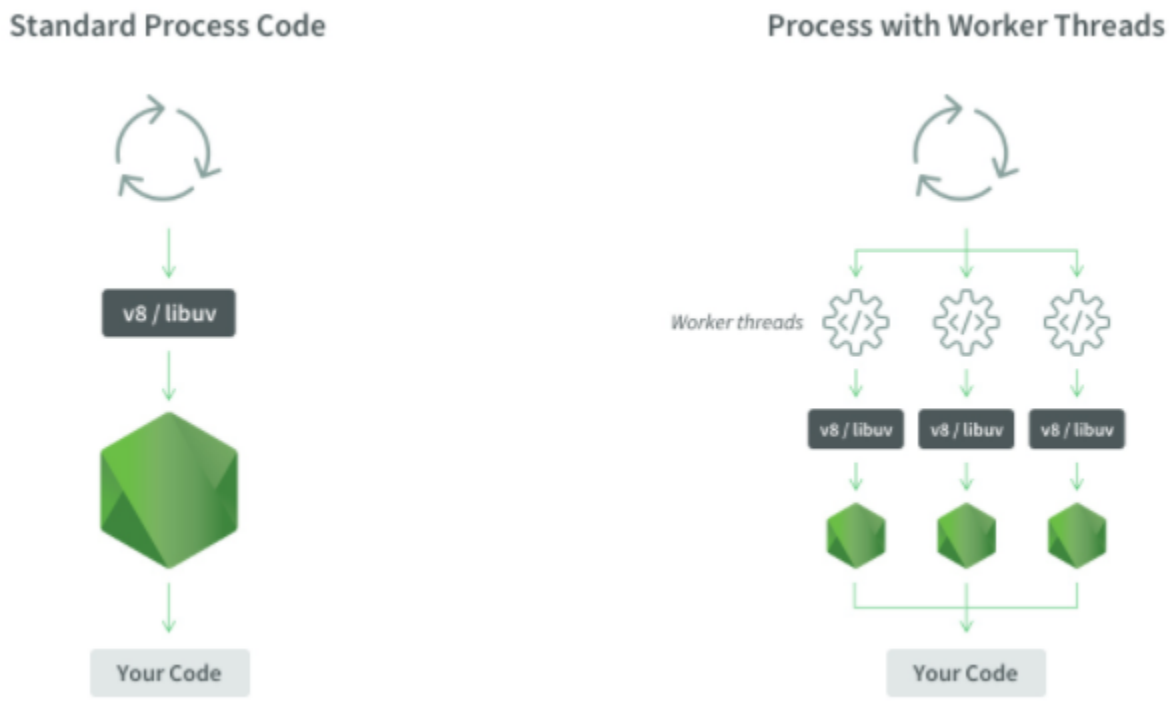
```
Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0        0  0.1  0       0
Processing:     1        2  1.4  2      25
Waiting:        1        2  1.3  2      24
Total:          1        2  1.4  2      25

Percentage of the requests served within a certain time (ms)
  50%    2
  66%    2
  75%    3
  80%    3
  90%    4
  95%    5
  98%    6
  99%    8
 100%   25 (longest request)
```

Es decir de esta primera muestra se observa que en el punto 3, se procesa el doble de request pero se respeta la capacidad de cores del equipo, esto hace más eficiente el procesamiento que en el caso 2, donde se aplicó el doble de concurrencia.

Ventajas y desventajas

Ventajas	Desventajas
No hay bloqueos	Mayor complejidad de debugging
Mejora la velocidad de la aplicación	La cantidad de threads limitada por la cantidad de cores del equipo
cpu óptimamente utilizada	no es recomendable con operaciones de alto consumo de cpu.
es muy bueno cuando tenemos muchas operaciones de I/O	RoundRobin (balanceo de cargas)no se aplica en windows, en otros os como linux está habilitado por default

Worker Threads



https://nodesource.com/blog/worker-threads-nodejs/

Se va utilizar esta técnica adecuadamente cuando se combine en la 2da parte de las pruebas, dado que al ejecutar este código no se realiza la creación de tickets como en los casos anteriores

No obstante al igual que con los casos anteriores voy a realizar pruebas con apache benchmark

```
ab -c 8 -n 1000 http://localhost:3000/
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
```

```
Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:
Server Hostname:        localhost
Server Port:            3000

Document Path:          /
Document Length:        33 bytes

Concurrency Level:      8
Time taken for tests:   0.501 seconds
Complete requests:      1000
Failed requests:        0
Non-2xx responses:      1000
Total transferred:      372000 bytes
HTML transferred:       33000 bytes
Requests per second:    1994.61 [#/sec] (mean)
Time per request:       4.011 [ms] (mean)
Time per request:       0.501 [ms] (mean, across all concurrent requests)
Transfer rate:          724.60 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:     3    4   1.2      3      11
Waiting:        1    2   1.3      2      10
Total:          3    4   1.2      4      11

Percentage of the requests served within a certain time (ms)
  50%    4
  66%    4
  75%    4
  80%    4
  90%    5
  95%    7
  98%    9
```

```
 99%   10
100%   11 (longest request)
```

```
ab -c 16 -n 1000 http://localhost:3000/
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:
Server Hostname:       localhost
Server Port:           3000

Document Path:         /
Document Length:       33 bytes

Concurrency Level:     16
Time taken for tests:  0.477 seconds
Complete requests:     1000
Failed requests:       0
Non-2xx responses:     1000
Total transferred:     372000 bytes
HTML transferred:      33000 bytes
Requests per second:   2096.83 [#/sec] (mean)
Time per request:      7.631 [ms] (mean)
Time per request:      0.477 [ms] (mean, across all concurrent requests)
Transfer rate:         761.74 [Kbytes/sec] received

Connection Times (ms)
```

```
              min  mean[+/-sd] median   max
Connect:        0        0   0.1      0       1
Processing:     2        7   1.3      7      12
Waiting:        1        4   1.7      4      11
Total:          2        8   1.3      7      12

Percentage of the requests served within a certain time (ms)
  50%    7
  66%    8
  75%    8
  80%    8
  90%    9
  95%   10
  98%   11
  99%   12
 100%   12 (longest request)
```

```
ab -c 8 -n 2000 http://localhost:3000/
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 200 requests
Completed 400 requests
Completed 600 requests
Completed 800 requests
Completed 1000 requests
Completed 1200 requests
Completed 1400 requests
Completed 1600 requests
Completed 1800 requests
Completed 2000 requests
Finished 2000 requests


Server Software:
Server Hostname:        localhost
Server Port:            3000

Document Path:          /
Document Length:        33 bytes

Concurrency Level:      8
Time taken for tests:   0.695 seconds
```

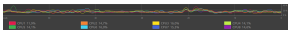
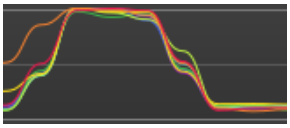
```
Complete requests:      2000
Failed requests:        0
Non-2xx responses:      2000
Total transferred:      744000 bytes
HTML transferred:       66000 bytes
Requests per second:    2878.63 [#/sec] (mean)
Time per request:       2.779 [ms] (mean)
Time per request:       0.347 [ms] (mean, across all concurrent requests)
Transfer rate:          1045.75 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       0
Processing:     0    3   0.8      2       7
Waiting:        0    1   0.8      1       6
Total:          1    3   0.8      3       7
WARNING: The median and mean for the processing time are not within a normal
deviation
        These results are probably not that reliable.

Percentage of the requests served within a certain time (ms)
  50%    3
  66%    3
  75%    3
  80%    3
  90%    4
  95%    4
  98%    5
  99%    6
 100%    7 (longest request)
```

Ventajas y desventajas :

Ventajas	Desventajas
No hay bloqueos	Menor tiempo de respuesta
Los threads esencialmente corren independientes dentro del mismo proceso.	Es experimental no se recomienda en ambientes productivos.
Atomicidad: los procesos concurrentes son más eficientes	No es recomendable con abundantes operaciones I/O

Deadlock detection	Tener presente el alcance de las variables, ya que las funciones en los workers son externas al event loop
Mayor aislamiento, si un proceso es afectado, no afecta a otros.	
recomendable para operaciones cpu intensivas.	

Tabla comparativa resumida:

Cpu previo a las pruebas: 	concurrencia: 8, 1000 request	concurrencia 16 ,request 1000	concurrencia 8 request 2000
Asincronismo	Concurrency Level: 8 Time taken for tests: 0.473 seconds Complete requests: 1000 Failed requests: 0 Non-2xx responses: 1000 Total transferred: 372000 bytes HTML transferred: 33000 bytes Requests per second: 2113.22 [#/sec] (mean) Time per request: 3.786 [ms] (mean) Time per request: 0.473 [ms] (mean, across all concurrent requests) Transfer rate: 767.69 [Kbytes/sec] received	Concurrency Level: 16 Time taken for tests: 0.406 seconds Complete requests: 1000 Failed requests: 0 Non-2xx responses: 1000 Total transferred: 372000 bytes HTML transferred: 33000 bytes Requests per second: 2460.77 [#/sec] (mean) Time per request: 6.502 [ms] (mean) Time per request: 0.406 [ms] (mean, across all concurrent requests) Transfer rate: 893.95 [Kbytes/sec] received	Concurrency Level: 8 Time taken for tests: 0.659 seconds Complete requests: 2000 Failed requests: 0 Non-2xx responses: 2000 Total transferred: 744000 bytes HTML transferred: 66000 bytes Requests per second: 3033.19 [#/sec] (mean) Time per request: 2.637 [ms] (mean) Time per request: 0.330 [ms] (mean, across all concurrent requests) Transfer rate: 1101.90 [Kbytes/sec] received
Cluster Mode 	Concurrency Level: 8 Concurrency Level: 8 Time taken for tests: 0.278 seconds Complete requests: 1000 Failed requests: 0 Non-2xx responses: 1000 Total transferred: 372000 bytes HTML transferred: 33000 bytes Requests per second: 3595.21 [#/sec] (mean) Time per request: 2.225 [ms] (mean) Time per request: 0.278 [ms] (mean, across all concurrent requests) Transfer rate: 1306.07 [Kbytes/sec] received	Concurrency Level: 16 Time taken for tests: 0.304 seconds Complete requests: 1000 Failed requests: 0 Non-2xx responses: 1000 Total transferred: 372000 bytes HTML transferred: 33000 bytes Requests per second: 3292.05 [#/sec] (mean) Time per request: 4.860 [ms] (mean) Time per request: 0.304 [ms] (mean, across all concurrent requests) Transfer rate: 1195.94 [Kbytes/sec] received	Concurrency Level: 8 Time taken for tests: 0.596 seconds Complete requests: 2000 Failed requests: 0 Non-2xx responses: 2000 Total transferred: 744000 bytes HTML transferred: 66000 bytes Requests per second: 3355.39 [#/sec] (mean) Time per request: 2.384 [ms] (mean) Time per request: 0.298 [ms] (mean, across all concurrent requests) Transfer rate: 1218.95 [Kbytes/sec] received
Conclusión: Se observa que con Cluster Mode, la cantidad de request por segundo, en cualquiera de las pruebas es siempre mayor al comparar con el primer script, y el tiempo por request es menor, por lo que se ve más óptima esta segunda técnica sobre la primera.			

Pruebas combinadas

Para las pruebas se buscó combinar las distintas técnicas para evaluar sus resultados, la idea es crear los registros de tickets y para cada nuevo lote generar un número de fibonacci(para tratar siempre con las mismas condiciones se va generar el mismo número fib(50)):

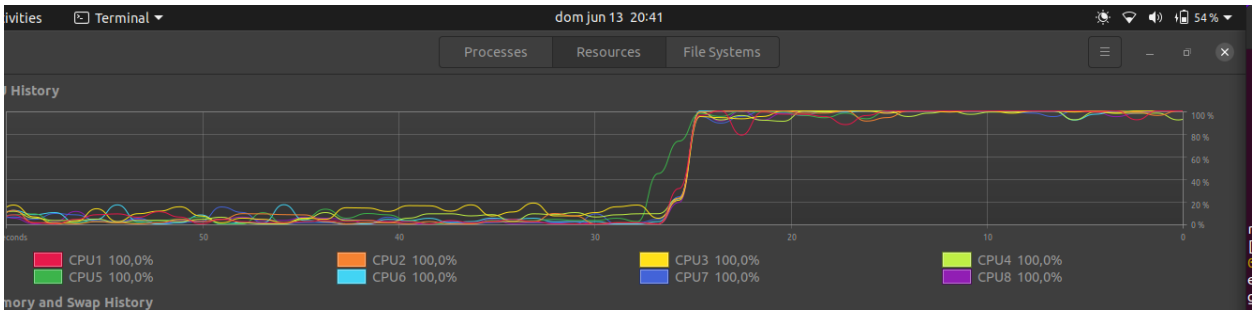
1) Cluster Mode + Worker Threads	serverClusterMode_io_cpu.js	10 minutos con 800 registros sin errores en terminal,usa todas las cpus.	Cluster mode se encarga de i/o, y worker threads operaciones cpu intensivas
2) Cluster Mode sin Worker Threads	serverClusterMode_io_fibonacci.js	12 minutos con 800 registros, pero con errores en terminal y no agrego registros en la colección. usa todas las cpu	Operaciones cpu intensivas sin Worker Threads
3) Worker Threads sin Cluster Mode	serverWorker_io_cpu.js	4 minutos pero solo 100 registros y genera los json y lo agrega a la base. Trabaja de a una cpu a la vez.	Operaciones i/o sin Cluster Mode.
4) Cluster Mode	serverClusterMode.js	En 1 minuto 800 registros, cpus trabajaron todas	Cluster Mode sin calcular fibonacci.

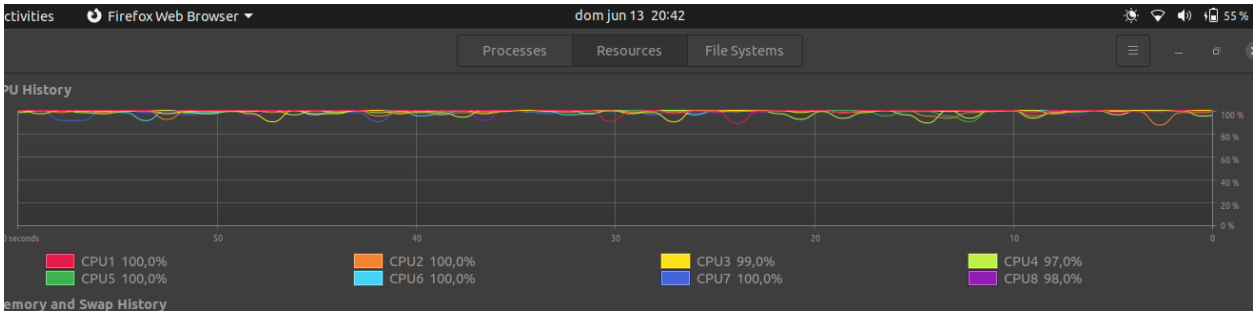
1)Cluster Mode combinado con Worker Threads

(**node --experimental-worker serverClusterMode_io_cpu.js**):

Aquí genera 100 registros por core y también un número de fibonacci, es decir al finalizar tenemos 800 registros y los 8 números que identifican a cada lote. El tiempo total en esta prueba fue de 7 minutos pero en sucesivas pruebas fue más, por lo que puse el promedio de 10 minutos en el cuadro anterior. Durante ese tiempo las cpus estuvieron al 100%:

Inicio



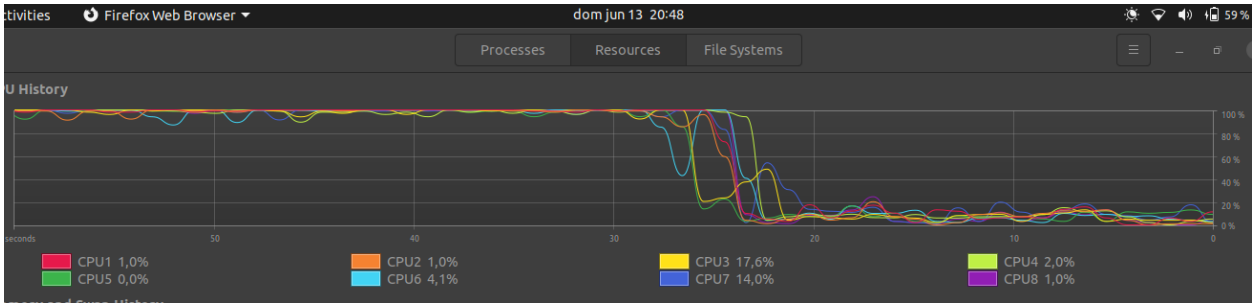


los 8 lotes de 100 registros cada los hizo a una gran velocidad y sin errores

```
 idPrestador: '10003',
 prestacion: 'aciclovir',
 fechaCreacion: 2021-06-13T23:41:23.812Z,
 comments:
  [ { _id: 60c697a36dbfeba03faa4170,
      whoComment: 'Checkmed',
      textBody: ' Verifique la medicacion, por favor',
      dateComment: 2021-06-13T23:41:23.815Z } ],
 version: 0 }
ota find: { medicinaArray: [ 'asmabron', 'asmavitan' ] }
]

scribiendo ./public/files/creados/10001-Grant Lincoln-60c697a36dbfeba03faa4172.json
en result: { _id: 60c697a36dbfeba03faa4172,
 apyn: 'Grant Lincoln',
 nroAfiliado: '75819760',
 diagnostico: 'asma',
 idPrestador: '10001',
 prestacion: 'asmabron',
 fechaCreacion: 2021-06-13T23:41:23.819Z,
 comments: [],
 version: 0 }
```

```
  nroAfiliado: '75819760',
  diagnostico: 'asma',
  idPrestador: '10001',
  prestacion: 'asmabron',
  fechaCreacion: 2021-06-13T23:41:23.819Z,
  comments: [],
  version: 0 }
50th Fibonacci Number: 12586269025
50th Fibonacci Number: 12586269025
50th Fibonacci Number: 12586269025
50th Fibonacci Number: 12586269025
50th Fibonacci Number: 12586269025
50th Fibonacci Number: 12586269025
50th Fibonacci Number: 12586269025
50th Fibonacci Number: 12586269025
```

finalizo: 20:48



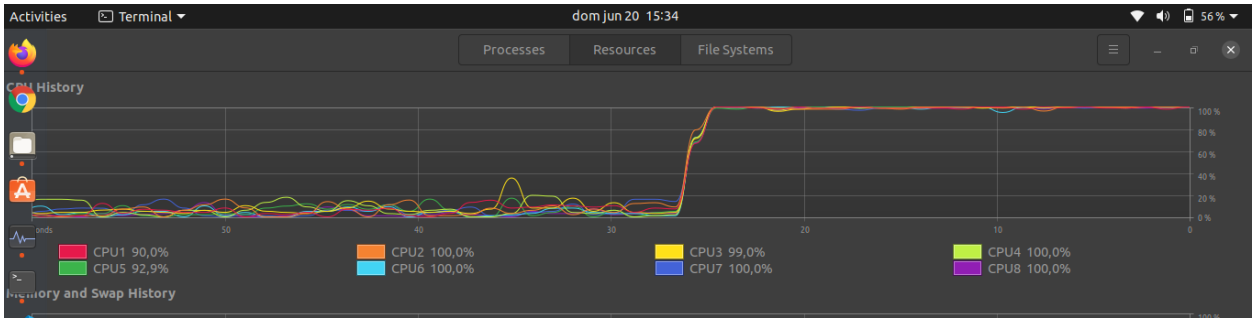
el tiempo total fue: 7 minutos

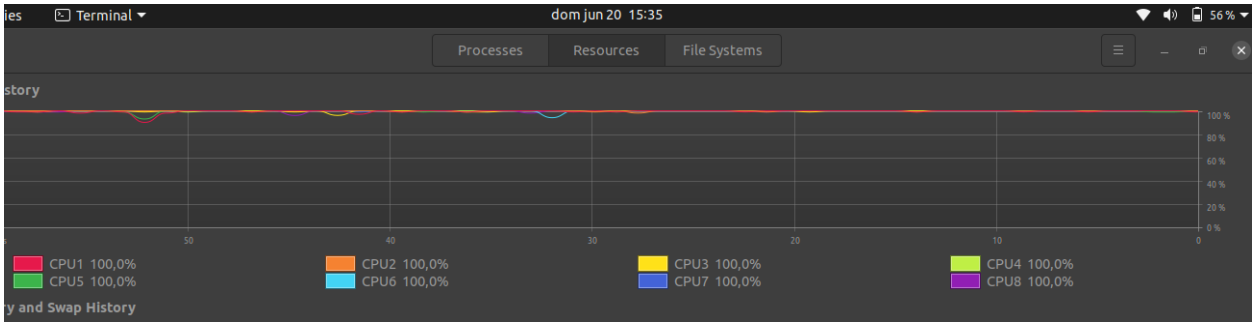
Name	Size	Permissions	Modified — Time	Accessed	Star	Detailed Type
10001-Rodriguez Sylvia-60c6979da0c213a021afdf7c.json	210 bytes	-rw-rw-r--	20:41	20:41	☆	JSON document
10001-O'Conner Colin-60c6979d581a5aa01bd075a1.json	213 bytes	-rw-rw-r--	20:41	20:41	☆	JSON document
10001-Kuvalis Reagan-60c6979da0c213a021afdf8e.json	225 bytes	-rw-rw-r--	20:41	20:41	☆	JSON document
10001-Jerde Hank-60c6979d581a5aa01bd07596.json	221 bytes	-rw-rw-r--	20:41	20:41	☆	JSON document
10001-Grimes Laverne-60c6979d223ce0a0390d3990.json	213 bytes	-rw-rw-r--	20:41	20:41	☆	JSON document
10001-Bechtelar Pattie-60c6979d581a5aa01bd075a0.json	223 bytes	-rw-rw-r--	20:41	20:41	☆	JSON document
10004-Welch Shea-60c694befcf1729f27eb004a.json	221 bytes	-rw-rw-r--	20:29	20:29	☆	JSON document
10004-Corkery Raymond-60c694befcf1729f27eb0047.json	222 bytes	-rw-rw-r--	20:29	20:29	☆	JSON document
10004-Zemlak Brooklyn-60c694bdfcf1729f27eaff72.json	220 bytes	-rw-rw-r--	20:29	20:29	☆	JSON document
10004-Wintheiser Amelie-60c694bdfcf1729f27eafffb.json	211 bytes	-rw-rw-r--	20:29	20:29	☆	JSON document
10004-Wiegand Aniya-60c694bdfcf1729f27eaff63.json	209 bytes	-rw-rw-r--	20:29	20:29	☆	JSON document
10004-Watsica Emmitt-60c694bdfcf1729f27eb001f.json	225 bytes	-rw-rw-r--	20:29	20:29	☆	JSON document
10004-Walsh Filiberto-60c694bdfcf1729f27eaffed.json	213 bytes	-rw-rw-r--	20:29	20:29	☆	JSON document
10004-Waelchi Frieda-60c694bdfcf1729f27eaff7c.json	208 bytes	-rw-rw-r--	20:29	20:29	☆	JSON document
10004-Smitham Lexie-60c694bdfcf1729f27eaffe7.json	205 bytes	-rw-rw-r--	20:29	20:29	☆	JSON document

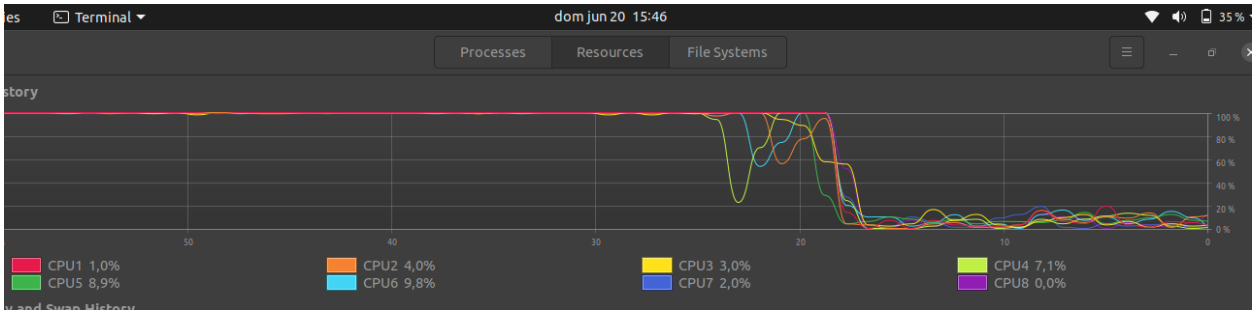
800 items selected (168,8 kB)

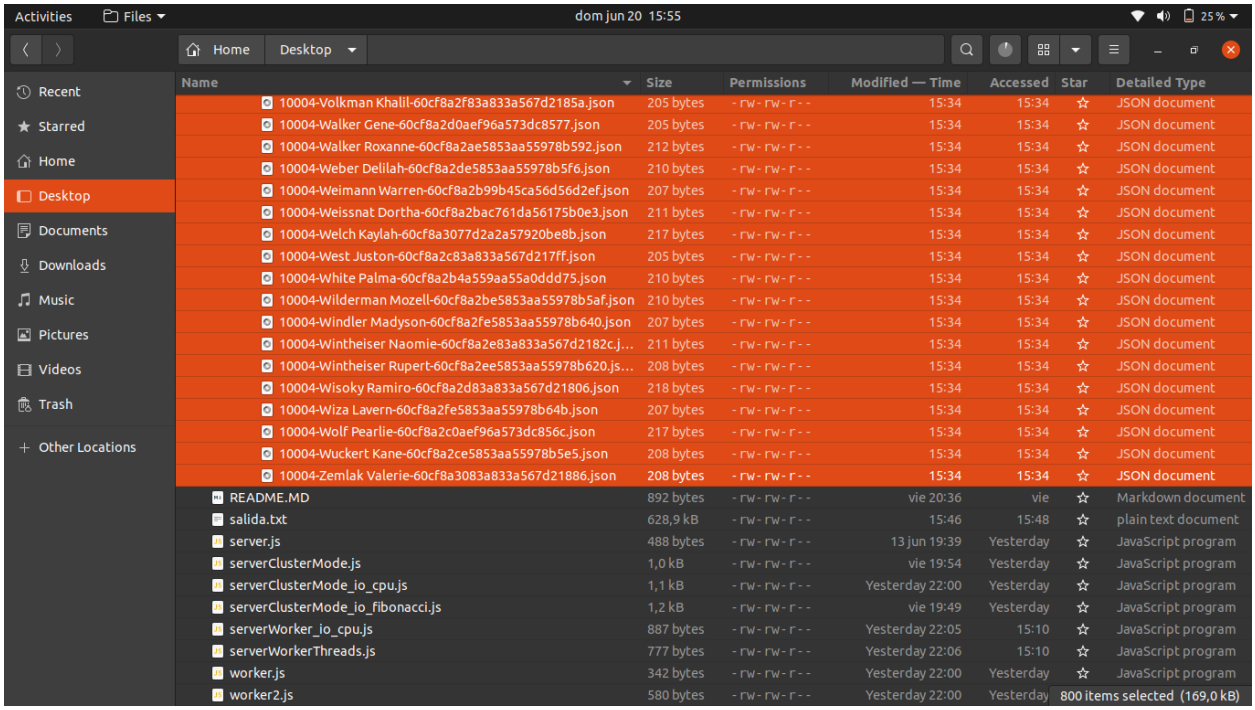
Se observan los 800 registros creados 20:41.

Posteriormente hice otra prueba con las mismas características, el tiempo fue mayor, pero cumplio con los 800 registros y los 8 cálculos de fibonacci:


Activities
Terminal
dom jun 20 15:34
56 %
Processes
Resources
File Systems
CPU History
CPU1 90,0%
CPU2 100,0%
CPU3 99,0%
CPU4 100,0%
CPU5 92,9%
CPU6 100,0%
CPU7 100,0%
CPU8 100,0%
Memory and Swap History


Terminal
dom jun 20 15:35
56 %
Processes
Resources
File Systems
CPU1 100,0%
CPU2 100,0%
CPU3 100,0%
CPU4 100,0%
CPU5 100,0%
CPU6 100,0%
CPU7 100,0%
CPU8 100,0%


Terminal
dom jun 20 15:46
35 %
Processes
Resources
File Systems
CPU1 1,0%
CPU2 4,0%
CPU3 3,0%
CPU4 7,1%
CPU5 8,9%
CPU6 9,8%
CPU7 2,0%
CPU8 0,0%



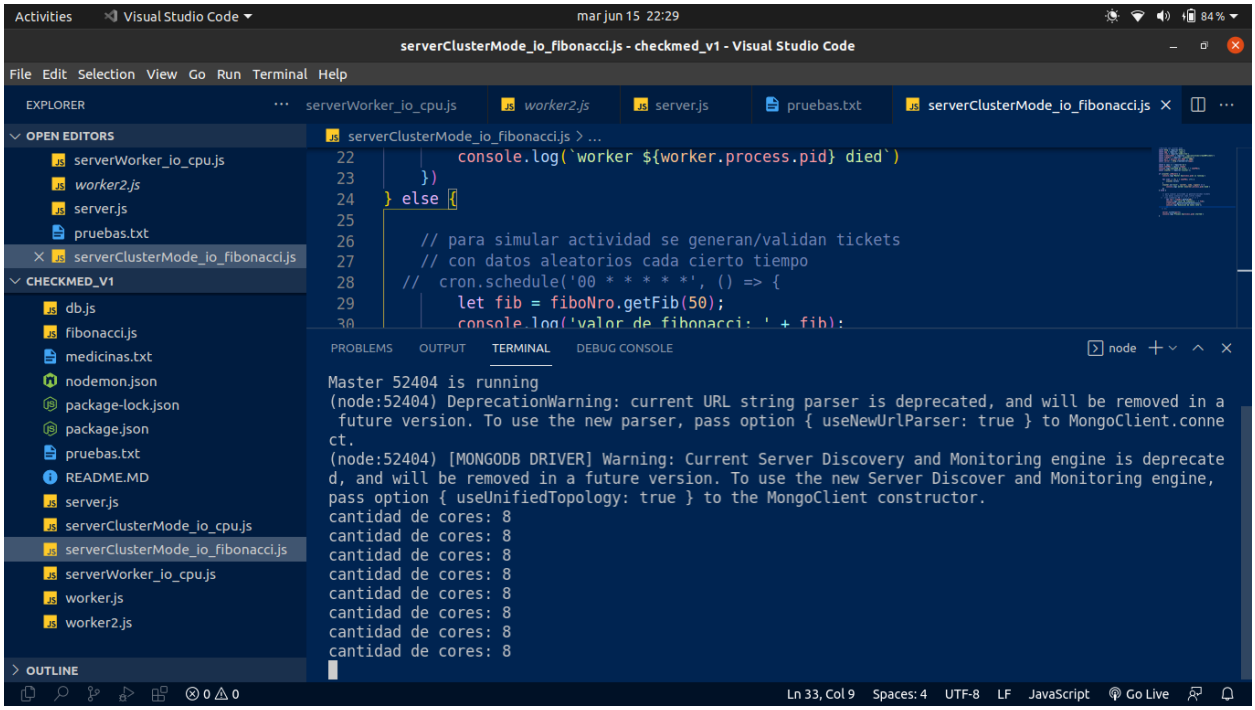
---

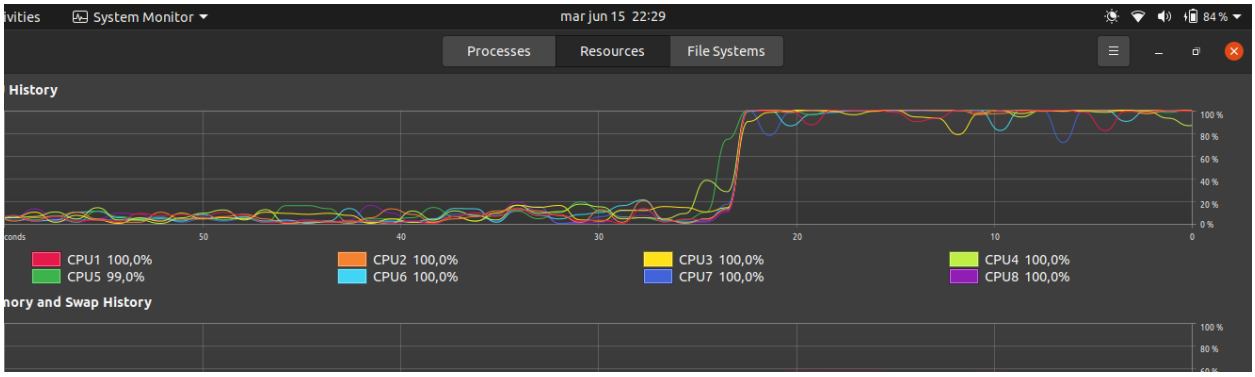
2) Cluster Mode sin Worker Threads

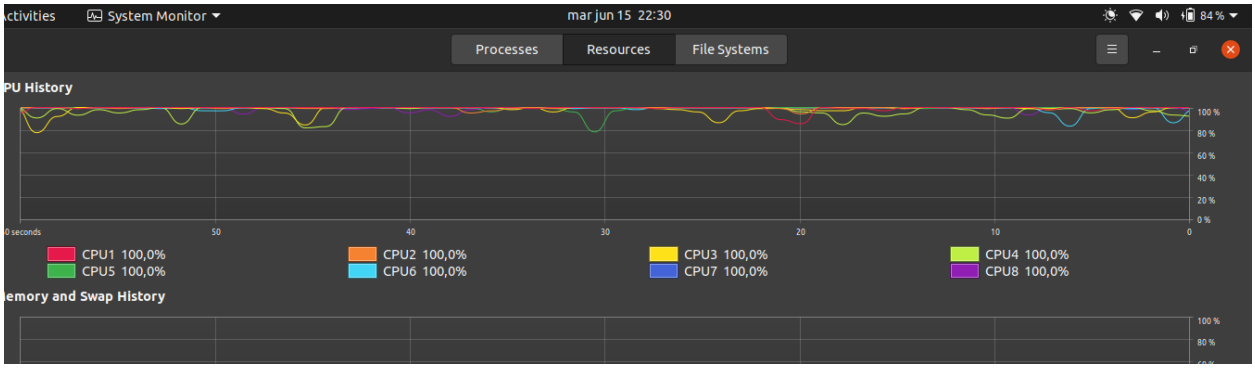
node serverClusterMode_io _fibonacci

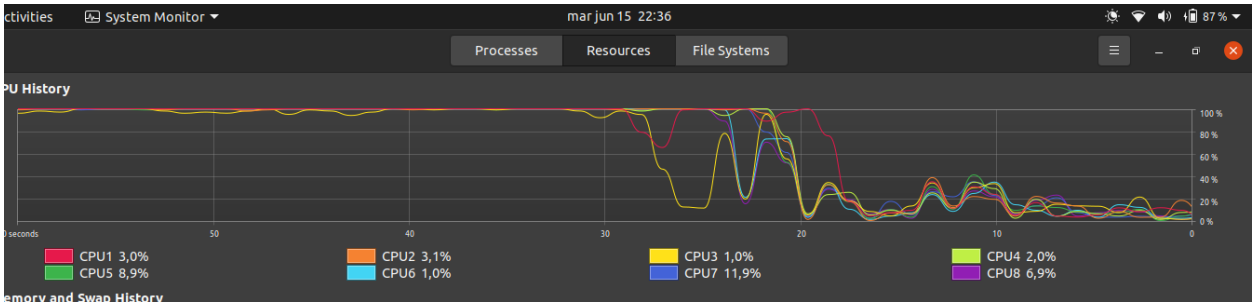
En este caso será ejecutarlo sin worker threads con una utilización intensiva de cpu para calcular el número de fibonacci, a full con los cpu 100% casi todas todo el tiempo

tiempo inicial:22,29


Activities
Visual Studio Code
mar jun 15 22:29
84 %
serverClusterMode_io_fibonacci.js - checkmed_v1 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
serverWorker_io_cpu.js
worker2.js
server.js
pruebas.txt
serverClusterMode_io_fibonacci.js
OPEN EDITORS
serverWorker_io_cpu.js
worker2.js
server.js
pruebas.txt
serverClusterMode_io_fibonacci.js
CHECKMED_V1
db.js
fibonacci.js
medicinas.txt
nodemon.json
package-lock.json
package.json
pruebas.txt
README.MD
server.js
serverClusterMode_io_cpu.js
serverClusterMode_io_fibonacci.js
serverWorker_io_cpu.js
worker.js
worker2.js
OUTLINE
serverClusterMode_io_fibonacci.js > ...
22 console.log(`worker ${worker.process.pid} died`)
23 })
24 } else {
25
26 // para simular actividad se generan/validan tickets
27 // con datos aleatorios cada cierto tiempo
28 // cron.schedule('00 * * * * *', () => {
29 let fib = fiboNro.getFib(50);
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
node
Master 52404 is running
(node:52404) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
(node:52404) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
cantidad de cores: 8
cantidad de cores: 8
cantidad de cores: 8
cantidad de cores: 8
cantidad de cores: 8
cantidad de cores: 8
cantidad de cores: 8
cantidad de cores: 8
Ln 33, Col 9
Spaces: 4
UTF-8
LF
JavaScript
Go Live


System Monitor
mar jun 15 22:29
84 %
Processes
Resources
File Systems
History
100 %
80 %
60 %
40 %
20 %
0 %
50
40
30
20
10
0
CPU1 100,0%
CPU5 99,0%
CPU2 100,0%
CPU6 100,0%
CPU3 100,0%
CPU7 100,0%
CPU4 100,0%
CPU8 100,0%
ory and Swap History

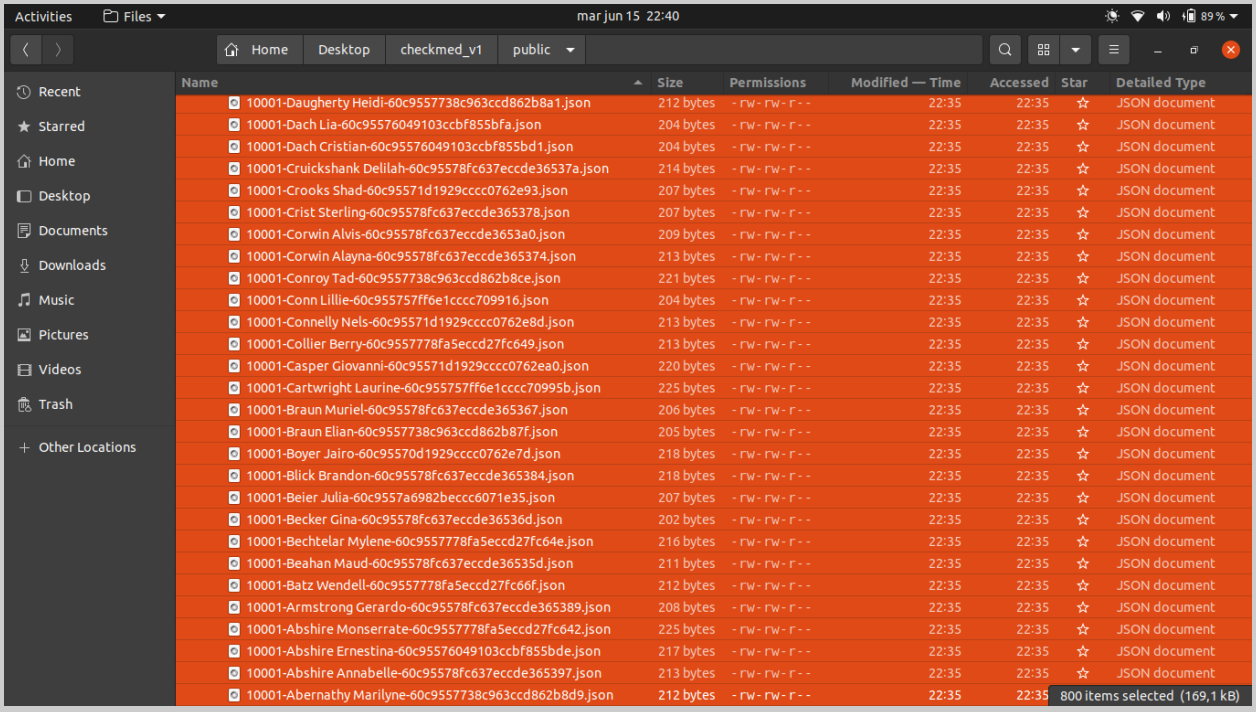




tiempo final: 22,36

y genero 800 registros pero con errores (se esperaba los comentarios mencionando el error en el diagnóstico)

```
(node:52422) UnhandledPromiseRejectionWarning: MongooseError: Operation `medicamentos.find()` buff
ering timed out after 10000ms
    at Timeout.setTimeout (/home/marcelo880/Desktop/checkmed_v1/node_modules/mongoose/lib/drivers/
node-mongodb-native/collection.js:185:20)
    at ontimeout (timers.js:436:11)
    at tryOnTimeout (timers.js:300:5)
    at listOnTimeout (timers.js:263:5)
    at Timer.processTimers (timers.js:223:10)
(node:52422) UnhandledPromiseRejectionWarning: Unhandled promise rejection. This error originated
either by throwing inside of an async function without a catch block, or by rejecting a promise wh
ich was not handled with .catch(). (rejection id: 50)
MongooseError: Operation `medicamentos.find()` buffering timed out after 10000ms
    at Timeout.setTimeout (/home/marcelo880/Desktop/checkmed_v1/node_modules/mongoose/lib/drivers/
node-mongodb-native/collection.js:185:20)
    at ontimeout (timers.js:436:11)
    at tryOnTimeout (timers.js:300:5)
    at listOnTimeout (timers.js:263:5)
    at Timer.processTimers (timers.js:223:10)
(node:52422) UnhandledPromiseRejectionWarning: MongooseError: Operation `medicamentos.find()` buff
ering timed out after 10000ms
    at Timeout.setTimeout (/home/marcelo880/Desktop/checkmed_v1/node_modules/mongoose/lib/drivers/
```



---

3) Worker Threads sin Cluster Mode

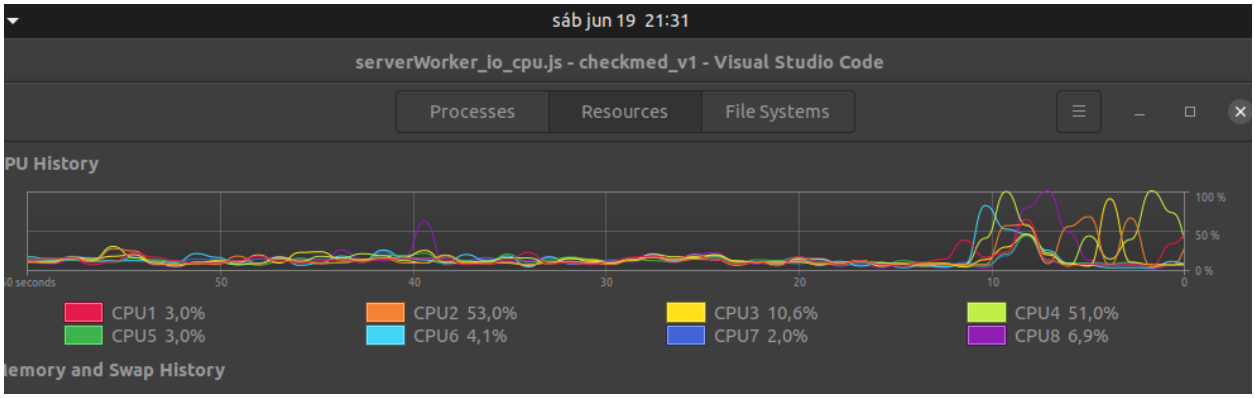
node --experimental-worker serverWorker_io_cpu.js

Se va repetir el cálculo de fibonacci y la generación de archivos considerando la cantidad de cores para equiparar con la prueba (1) pero utilizando worker threads

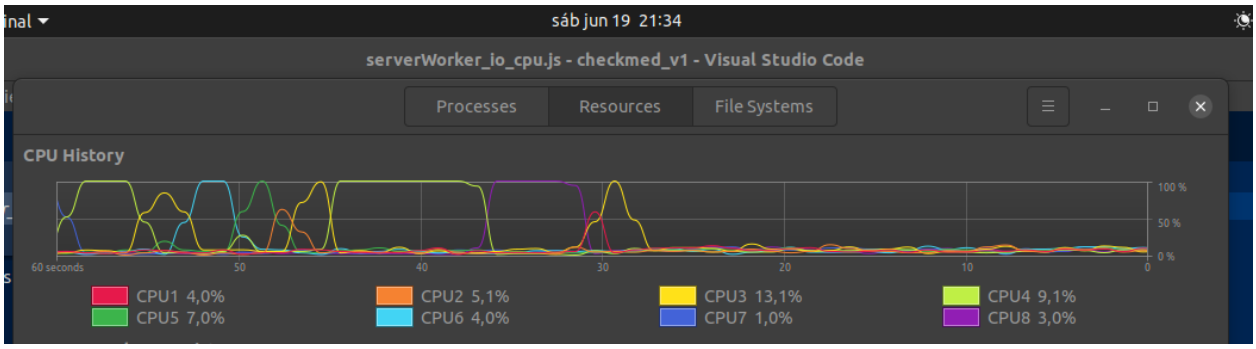
de a un cpu 100%, y al momento generó los registros de tickets pero solamente 100 , no ochocientos como tenía planeado, lo cual es extraño, pero seguramente quedo bloqueado con el uso intensivo de cpu

tiempo inicial,21:31

tiempo final: 21, 34



tiempo final:21:34



```
marcelo880@marcelo880-N56VB: ~/Desktop/checkmed_v1
  fechaCreacion: 2021-06-20T00:31:03.185Z,
  comments:
   [ { _id: 60ce8c471506b018ad884b26,
       whoComment: 'Checkmed',
       textBody: ' Verifique la medicacion, por favor',
       dateComment: 2021-06-20T00:31:03.187Z } ],
  version: 0 }
50th Fibonacci Number: 12586269025
50th Fibonacci Number: 12586269025
50th Fibonacci Number: 12586269025
50th Fibonacci Number: 12586269025
50th Fibonacci Number: 12586269025
50th Fibonacci Number: 12586269025
50th Fibonacci Number: 12586269025
50th Fibonacci Number: 12586269025
0
0
0
0
0
0
0
0
```

Tiempo total: 3 minutos pero incompleto, solo 100 registros.

10002-Ebert Elinore-60ce8c471506b018ad884af8.json	204 bytes	-rw-rw-r--	21:31	21:31	☆	JSON document
10002-Erdman Miguel-60ce8c471506b018ad884ae8.js...	213 bytes	-rw-rw-r--	21:31	21:31	☆	JSON document
10002-Ferry Carolina-60ce8c471506b018ad884b0f.json	219 bytes	-rw-rw-r--	21:31	21:31	☆	JSON document
10003-Auer Anais-60ce8c471506b018ad884b08.json	202 bytes	-rw-rw-r--	21:31	21:31	☆	JSON document
10003-Beatty Kelley-60ce8c471506b018ad884b22.json	206 bytes	-rw-rw-r--	21:31	21:31	☆	JSON document
10003-Bernier Devon-60ce8c471506b018ad884b02.json	224 bytes	-rw-rw-r--	21:31	21:31	☆	JSON document
10003-Cummings Clark-60ce8c471506b018ad884aee.j...	221 bytes	-rw-rw-r--	21:31	21:31	☆	JSON document
10003-Dietrich Madeline-60ce8c471506b018ad884b0...	214 bytes	-rw-rw-r--	21:31	21:31	☆	JSON document
10003-Fritsch Jabari-60ce8c471506b018ad884b06.json	208 bytes	-rw-rw-r--	21:31	21:31	☆	JSON document
10003-Homenick Isaac-60ce8c471506b018ad884b1b.j...	207 bytes	-rw-rw-r--	21:31	21:31	☆	JSON document
10003-Kreiger Hoyt-60ce8c471506b018ad884b07.json	223 bytes	-rw-rw-r--	21:31	21:31	☆	JSON document
10003-Schuppe Dee-60ce8c471506b018ad884ae5.json	209 bytes	-rw-rw-r--	21:31	21:31	☆	JSON document
10003-Sporer Aida-60ce8c471506b018ad884aff.json	211 bytes	-rw-rw-r--	21:31	21:31	☆	JSON document
10003-Wyman Sylvester-60ce8c471506b018ad884af1....	215 bytes	-rw-rw-r--	21:31	21:31	☆	JSON document
10004-Braun Rahul-60ce8c471506b018ad884ae4.json	205 bytes	-rw-rw-r--	21:31	21:31	☆	JSON document
10004-Hintz Kathlyn-60ce8c471506b018ad884b03.json	218 bytes	-rw-rw-r--	21:31	21:31	☆	JSON document
10004-Huels Sherwood-60ce8c471506b018ad884b0c.j...	207 bytes	-rw-rw-r--	21:31	21:31	☆	JSON document
10004-Windler Murl-60ce8c471506b018ad884af7.json	223 bytes	-rw-rw-r--	21:31	21:31	☆	JSON document
10004-Yundt Lesly-60ce8c471506b018ad884b18.json	211 bytes	-rw-rw-r--	21:31	21:31	☆	JSON document
README.MD	892 bytes	-rw-rw-r--	Yesterday 20:36	Yesterday	☆	Markdown document
server.js	488 bytes	-rw-rw-r--	dom 19:39	20:05	☆	JavaScript program
serverClusterMode.js	1,0 kB	-rw-rw-r--	Yesterday 19:54	Yesterday	☆	JavaScript program
serverClusterMode_io_cpu.js	1,1 kB	-rw-rw-r--	Yesterday 20:26	Yesterday	☆	JavaScript program
serverClusterMode_io_fibonacci.js	1,2 kB	-rw-rw-r--	Yesterday 19:49	Yesterday	☆	JavaScript program
serverWorker_io_cpu.js	901 bytes	-rw-rw-r--	Yesterday 20:28	21:13	☆	JavaScript program
serverWorkerThreads.js	791 bytes	-rw-rw-r--	Yesterday 20:28	Yesterday	☆	JavaScript program
Documents	8 items	drwxr-xr-x	jue 00:59	19:09	☆	Folder
Pictures	98 items	drwxr-xr-x	Yesterday 20:35	Yesterday	☆	Folder
Downloads	114 items	drwxr-xr-x	20:58	20:58	100 items selected (21,1 kB)	

4) Cluster Mode : Esta última prueba fue únicamente aplicando Cluster Mode y con operaciones de entrada salida, no se calculo un fibonacci. Todo anduvo bien, creo los 800 registros en 1 minuto, sin errores en terminal.

Conclusión

En las pruebas realizadas se comparó en primer lugar la creación de los registros para la base de datos sin utilizar ninguna de las técnicas vistas, por lo cual trabajó con en un único thread, posteriormente se utilizó cluster mode, donde se vió que adaptándolo a la cantidad de cores del equipo mejoró los tiempos, y no presenta errores al crear, validar y guardar los tickets, en numerosas operaciones de entrada salida.
Posteriormente se quiso utilizar una técnica experimental suponiendo un requerimiento con alto consumo de cpu, pero al combinarla con cluster mode tuvo un comportamiento erróneo al crear los registros.
En resumen me pareció performante y confiable el uso de cluster mode para la generación, validación, y persistencia de registros en una base de datos. Y también se quiso demostrar que tenemos otra alternativa para las actividades cpu intensivas, que muchas veces se tomo como punto débil de node js. Es decir ambas necesidades pueden cubrirse, pero lo ideal sería tratarlas por separado.