Tsr Exámenes

Exámenes

Tema 6. Escalabilidad. Parte 2. Test de autoevaluación

Volver a la Lista de Exámenes

Parte 1 de 2 - 4.0 Puntos

Preguntas 1 de 5

2.0 Puntos. Puntos descontados por fallo: 0.67

El módulo 'cluster' de Node.js:

- ○ **Facilita soporte para gestionar servicios distribuidos que deban ejecutarse en clusters de ordenadores.**
- ○ **Soporta implícitamente servicios distribuidos con replicación pasiva.**
- ✔ ○ **Equilibra la carga recibida entre todos los procesos trabajadores creados con su soporte.**
- ○ **Usa, por omisión, un patrón de comunicación ROUTER-DEALER.**

Respuesta correcta: C

Preguntas 2 de 5

2.0 Puntos. Puntos descontados por fallo: 0.67

El módulo "cluster" de Node.js se utiliza para...

- ○ **...gestionar múltiples hilos en un proceso Node.js.**
- ○ **...implantar fácilmente múltiples modelos de consistencia de memoria.**
- ✖ ○ **...desplegar un conjunto de programas Node.js en un "cluster" de ordenadores.**
- ○ **...gestionar un conjunto de procesos Node.js para que puedan compartir algunos recursos; p.ej., un puerto en el que escuchar y un mismo programa a ejecutar.**

Respuesta correcta: D

Parte 2 de 2 - 6.0 Puntos

Preguntas 3 de 5

2.0 Puntos. Puntos descontados por fallo: 0.67

Estamos escribiendo un programa Node.js con el módulo 'cluster'. Utiliza el doble de trabajadores que procesadores haya en el ordenador.

```
var cluster = require('cluster');
var http = require('http');
if (cluster.isMaster) {
  var numReqs = 0;
  var numWorkers = 0;
  /* A */
  function messageHandler(msg) {
    numReqs++;
  }
  var numCPUs = require('os').cpus().length;
  /* B */
  var numWorkers = numCPUs*2;
  for (var i in cluster.workers)
    cluster.workers[i].on('message', messageHandler);
} else {
  http.Server(function(req, res) {
    res.writeHead(200); res.end('hello world\n');
    process.send({ cmd: 'notify' });
  }).listen(8000);
}
```

Supongamos que esos trabajadores iniciales han sido creados en la parte B del código (no mostrada) y que la variable "numWorkers" mantiene el número actual de trabajadores. Necesitamos escribir la siguiente ampliación (en la parte A del programa): Si la cantidad de peticiones servidas cada 10 ms supera el valor actual de "numWorkers", un nuevo trabajador será iniciado. ¿Qué código implanta esa ampliación?

- ◯

```
setInterval( function() { if (numReqs > numWorkers)
  cluster.fork();}, 10000 );
```

- ◯

```
setInterval( function() { if (numReqs > numWorkers) {
  numWorkers++; cluster.fork()}; numReqs=0 }, 10 );
```

- ◯

```
setInterval( function() { if (numReqs > numWorkers) {
  numWorkers++; process.fork()}; numReqs=0 }, 10 );
```

- ✖◯

```
setInterval( function() { if (numReqs > numWorkers)
  process.fork();}, 10 );
```

Respuesta correcta: B

Preguntas 4 de 5

2.0 Puntos. Puntos descontados por fallo: 0.67

Si consideramos el programa siguiente:

```
const cluster = require('cluster');
const http = require('http');
let numCPUs = require('os').cpus().length;
if (cluster.isMaster) {
  for (let i=0; i < numCPUs; i++) cluster.fork();
  cluster.on('exit', function(who, code, signal) {
    console.log('Process '+ who.process.pid +' died');
  });
} else {
  http.createServer(function(req, res) {
    res.writeHead(200);
    res.end('hello world\n');
  }).listen(8000);
}
```

Se puede afirmar que:

- ◯ **El proceso *máster* crea tanto procesos trabajadores como procesadores (o núcleos) haya en el ordenador local.**
- ✖ ◯ **El módulo "cluster" permite desplegar cada trabajador generado en un ordenador distinto.**
- ◯ **El programa falla cuando se crea el segundo trabajador pues todos los trabajadores tratan se usar el mismo puerto (8000) y éste ya está en uso.**
- ◯ **El programa muestra un mensaje cuando el proceso *máster* finaliza.**

Respuesta correcta: A

Preguntas 5 de 5

2.0 Puntos. Puntos descontados por fallo: 0.67

Estamos escribiendo un programa Node.js con el módulo **cluster**. Utiliza tantos trabajadores como procesadores haya. El proceso maestro informará al usuario cada segundo sobre cuántas peticiones han sido atendidas hasta el momento por los trabajadores.

```
var cluster = require('cluster');
var http = require('http');
if (cluster.isMaster) {
  var numReqs = 0;
  setInterval(function(){ console.log("numReqs =", numReqs);}, 1000);
  function messageHandler(msg) {
    numReqs++;
  }
  var numCPUs = require('os').cpus().length;
  for (var i=0; i < numCPUs; i++) cluster.fork();
  /* (1) Worker message management instructions should be here. */
  /* (2) Worker regeneration code should be here, if any.       */
} else {
  http.Server(function(req, res) {
    res.writeHead(200); res.end('hello world\n');
    process.send({ cmd: 'notify' });
  }).listen(8000);
}
```

Elige qué instrucciones necesita el proceso maestro para gestionar (1) los mensajes de los trabajadores:

- ○
  ```
  cluster.on('message', messageHandler);
  ```
- ○
  ```
  cluster.on('notify', messageHandler);
  ```
- ○
  ```
  for (var i in workers) workers[i].on('notify', messageHandler);
  ```
- ✔ ○
  ```
  for (var i in cluster.workers) cluster.workers[i].on('message', messageHandler);
  ```

Respuesta correcta: D

- PoliformaT
- UPV

- Powered by Sakai
-