

UNIVERSIDAD NACIONAL DE LUJÁN



DOCENTES

David Petrocelli
Juan Martín Rodriguez

INTEGRANTE

Juan Ignacio Sacco

Trabajo Práctico Final

Renderización distribuida e Image Stacking

Propuesta

Se propone realizar un sistema distribuido con el objetivo de obtener un resultado de calidad competitiva y realista en la renderización de objetos 3D en la herramienta Blender 2.80.

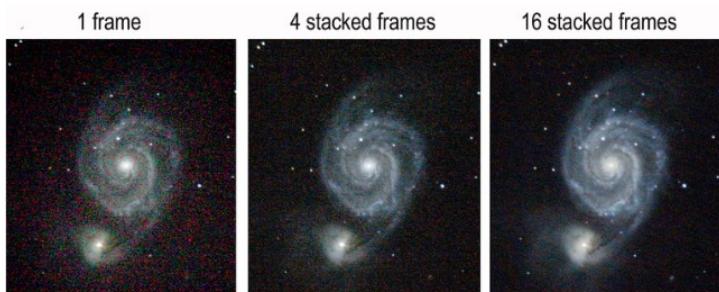
En general, un renderizado de buena calidad en la herramienta Blender (dependiendo del hardware) podría llegar a tardar más de una hora, dependiendo de la cantidad de objetos de una escena y calidad que se busque.

Lo que se propone en este trabajo es distribuir el proceso de renderización en diferentes nodos a través de la red, y que estos nodos realicen pequeñas renderizaciones de poco tiempo (lo que se traduce en imágenes mala calidad con mucho ruido). Una vez terminado el trabajo del nodo, cada uno publicará su trabajo realizado.

Una vez que se hayan publicado todos los trabajos realizados, el nodo maestro comenzará un proceso de “Image stacking”, proceso que consiste en tomar un pixel de una imagen y compararlo con el pixel de otra imagen que esté en la misma posición, se genera una nueva imagen, y se coloca en la posición de los pixeles comparados el valor de la media de los dos pixeles anteriores. Este proceso se puede dar gracias a que el ruido en las imágenes es aleatorio. Si apilamos las imágenes, podemos promediar pixel por pixel para obtener una imagen más suave y limpia.

Teóricamente en el proceso de Image stacking, la relación entre la señal y el ruido se reduce a la ley del cuadrado inverso.

| # de imágenes apiladas | Cantidad de ruido comparado con la imagen original |
|------------------------|--|
| 4 | 1/2 |
| 9 | 1/3 |
| 16 | 1/4 |
| 25 | 1/5 |
| 100 | 1/10 |



38 frames stacked and fully processed



Definición teórica del sistema

Metodologías de trabajo:

- Por tiempo: En este caso el Servidor dará un tiempo máximo para entregar los trabajos, cuando el tiempo se termine, obtendrá todos los elementos de la cola “trabajo realizado” y procederá a aplicar el algoritmo de Image Stacking. Esta es la forma mas apropiada de realizar de utilizar la aplicación, ya que, podremos darle al usuario casi el tiempo exacto de lo que va a tardar el proceso y ademas podemos aprovechar al máximo el hardware, esto es debido a que si tenemos 2 nodos con diferente hardware, uno renderizará mas cantidad de samples que otro.
- Por cantidad de samples: En este caso el Servidor publicará el archivo a renderizar con un objetivo de X cantidad de samples a realizar. Este proceso no es optimo, ya que como en el caso anterior, imaginemos que tenemos un equipo que renderiza 10 imágenes en 5 minutos, y otro que renderiza 10 imágenes en 20 minutos, el primer equipo estará ocioso 15 minutos.

Funciones del sistema por cada rol:

Cliente:

- Debe ser capaz de enviarle el archivo a renderizar, la metodología(tiempo o sample) y el numero de frame a renderizar al servidor.
- Debe poder recibir la respuesta del servidor (una imagen) y persistir el resultado.

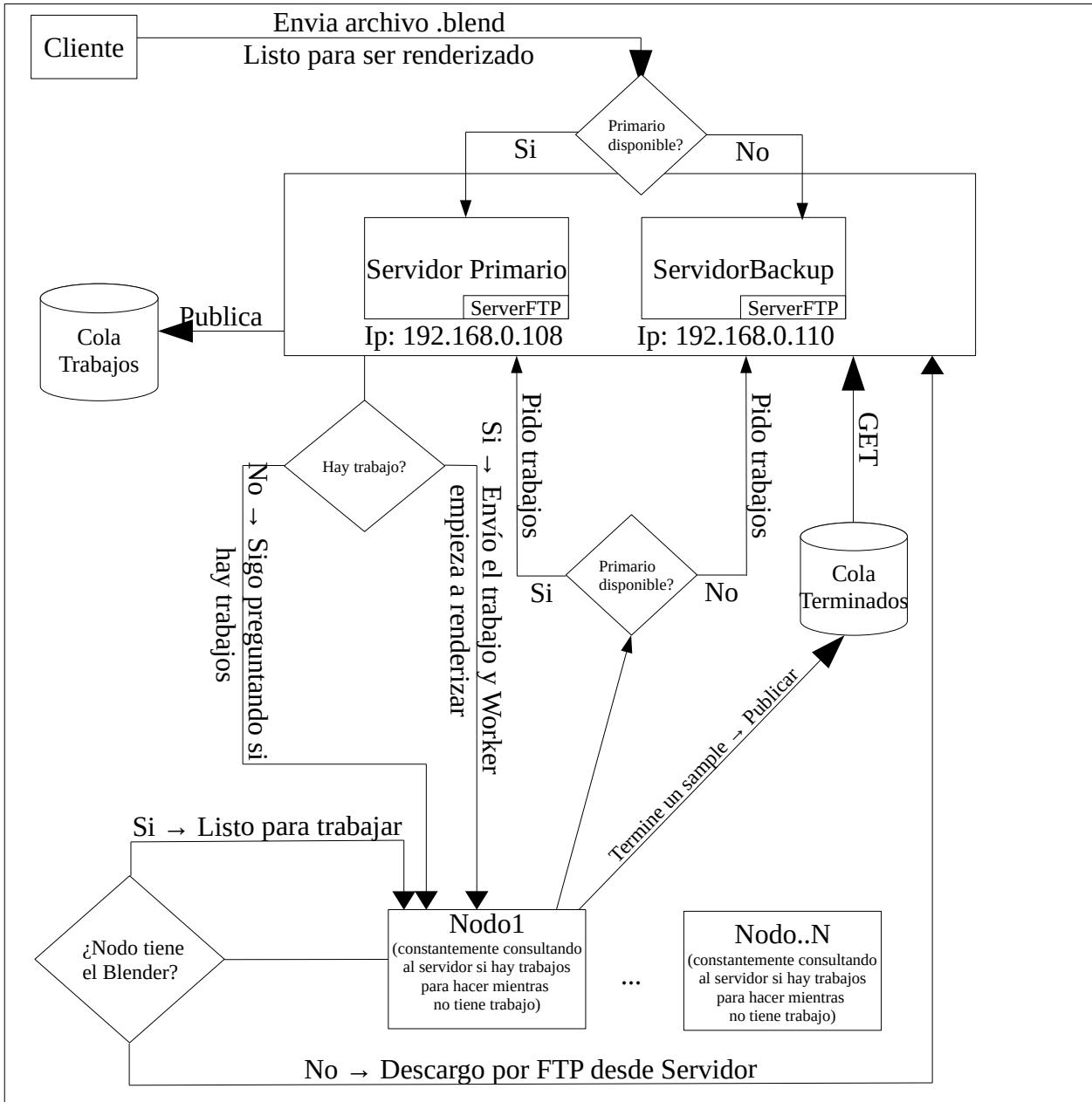
Servidor:

- Debe ser capaz de levantar un servidor RMI y publicar sus servicios.
- Debe ser capaz de levantar un servidor FTP (en el momento que este sea requerido) y suspenderlo cuando ya no se utilice.
- Debe ser capaz de recibir peticiones de un cliente y resolverlas en hilos separados.
- Debe ser capaz de crear las colas necesarias: “queueTrabajos” y “queueTerminados”.
- Debe ser capaz de poder diferenciar los diferentes trabajos terminados y resolverlos de tal forma que no se mezclen renderizados de diferentes trabajos.
- Debe ser capaz de eliminar los trabajos ya realizados de las colas para optimizar el espacio disponible.
- Debe ser capaz de aplicar el algoritmo de Image Stacking y enviarle el resultado al cliente.

Worker:

- Debe ser capaz de conectarse al servicio de colas.
- Debe ser capaz de calcular el tamaño de la carpeta Blender-app para decidir si necesita descargar el software Blender del servidor vía FTP.
- Debe poder descargar vía FTP los archivos necesarios para funcionar.
- Debe poder consultar la cola “queueTrabajos” constantemente mientras no tenga trabajos.
- Debe poder distinguir si los trabajos publicados en la cola “queueTrabajos” ya los realizo, y en caso de ser así, debe buscar otro trabajo ya publicado o esperar que publiquen uno nuevo.
- Debe poder preparar el proyecto “.blend” aplicándole los scripts python necesarios para generar ruido aleatorio y configurar el renderizado antes de comenzar a renderizar.
- Debe poder renderizar el proyecto “.blend”
- Debe poder subir a la cola de trabajos terminados todos los samples que haya hecho.

Arquitectura básica



Estado del arte

La técnica de distribuir el trabajo de renderizado en varios nodos, es una técnica con varios años de investigación y desarrollo. Este sistema de distribución se lo suele llamar comúnmente “granjas” de render y existe una considerable variedad software para cada aplicación de modelado en 3D.

Aplicaciones de modelado 3d

Estas aplicaciones son utilizados en sectores como la impresión 3D, los juegos, la animación, la arquitectura y el diseño industrial. Dentro de las aplicaciones mas destacadas se encuentran:

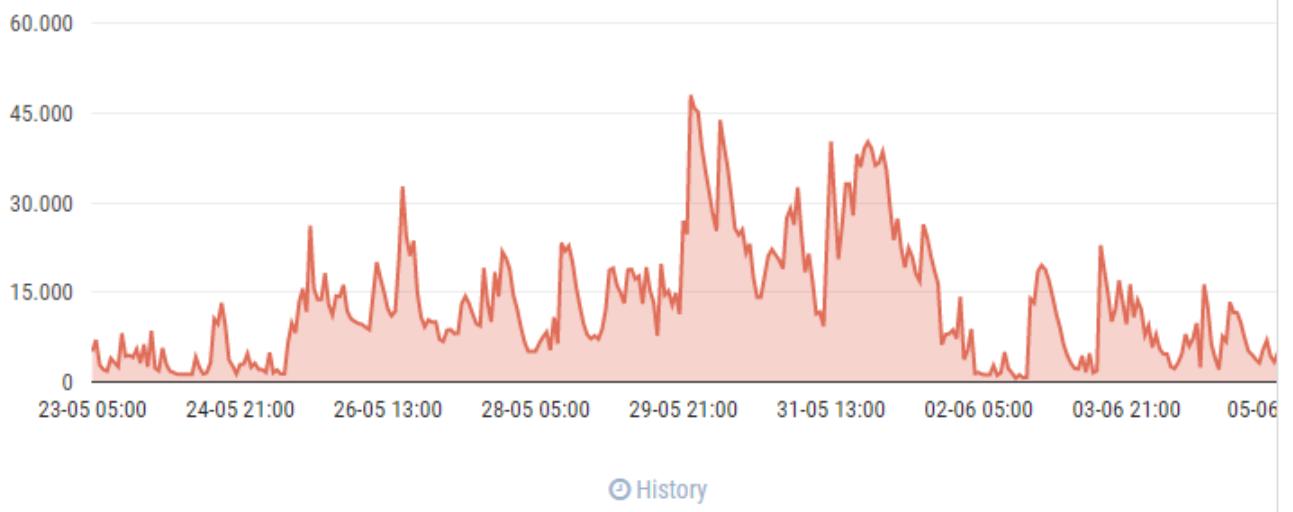
| Nombre | Nivel | Plataforma | Coste |
|------------|-------------|----------------------|--|
| SketchUp | Intermedio | Windows y Mac | Versión gratis. Versión Pro: 270 € |
| 3ds Max | Profesional | Windows | 1984,40 €/año |
| AutoCAD | Profesional | Windows y Mac | 2075,15 €/año |
| Blender | Profesional | Windows, Mac y Linux | Versión Gratis |
| ZBrush | Profesional | Windows y Mac | Versión Gratis Licencia educativa 400€ Usuario individual educativo 720€ |
| Solidworks | Industrial | Windows | 9950€ Licencias educativas disponibles |

El desarrollo e investigación será específicamente sobre la aplicación Blender, que como se puede apreciar: es de uso profesional; la que mas plataformas soporta y es completamente gratis. Ademas, tiene una gran comunidad de desarrollo detrás de ella.

Sistemas de renderizado distribuidos existentes:

- **BitWrk:** Se trata de una red p2p de renderizado alimentada por bitcoin, se puede ganar bitcoins por ofrecer tu equipo para renderizar, o se puede pagar en bitcoin para tener renderizados. Tiene soporte para Windows, Mac y Linux. Si bien se trata de una idea muy buena, el sistema no posee Peers/Actividad desde hace mucho tiempo y por tanto no sirve ya que el sistema necesita si o si de peers para funcionar.
- **RebusFarm:** Se trata de un sistema de renderizado en la nube, en el que ellos ponen los servidores y un software llamado RebusFarm2.0, dando soporte a todos las aplicaciones de modelado descriptas arriba y mas también. El precio es por Ghz, costando 1,4 centavos de dólar cada GHz.
- **BlenderGrid:** Otra página web de renderizado, en este caso, se envia el proyecto blend en un comprimido .zip, se eligen los parámetros de renderizado y ellos envían por mail una cotización del renderizado. El precio aproximado es de 1 dólar por frame en una resolución de 2048x858 y 600 Cycles Samples

- **SheepIt:** Es una red colaborativa que actualmente poseen aproximadamente 550 maquinas, es gratis pero al registrarse y bajarse el software ya se esta poniendo nuestro hardware a disposición de la red, y por tanto, sera utilizada para renderizar también.



[History](#)

Historial de cantidad de frames renderizados.

- **Blender Network Render:** Se trata de un software desarrollado por Paul Lutus en lenguaje Python bajo la licencia GPL, es un script que se puede instalar en las computadoras que se van a utilizar para renderizar y permite la creación de distintos perfiles adaptándose a la velocidad de renderizado de la computadora, esto se utiliza para partir la imagen a renderizar de manera equilibrada en relación a potencia/cantidad de trabajo.

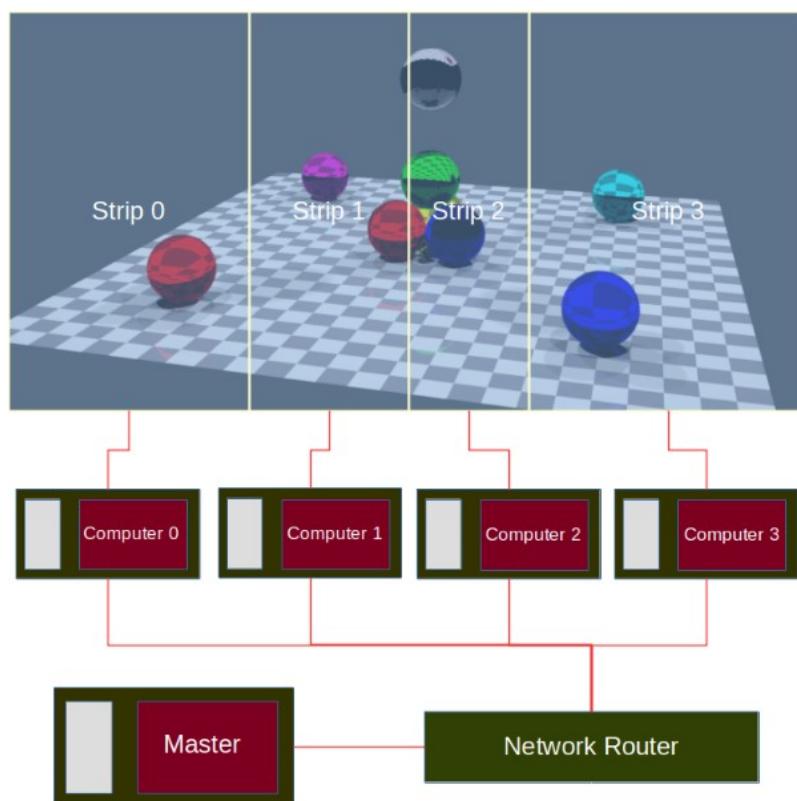
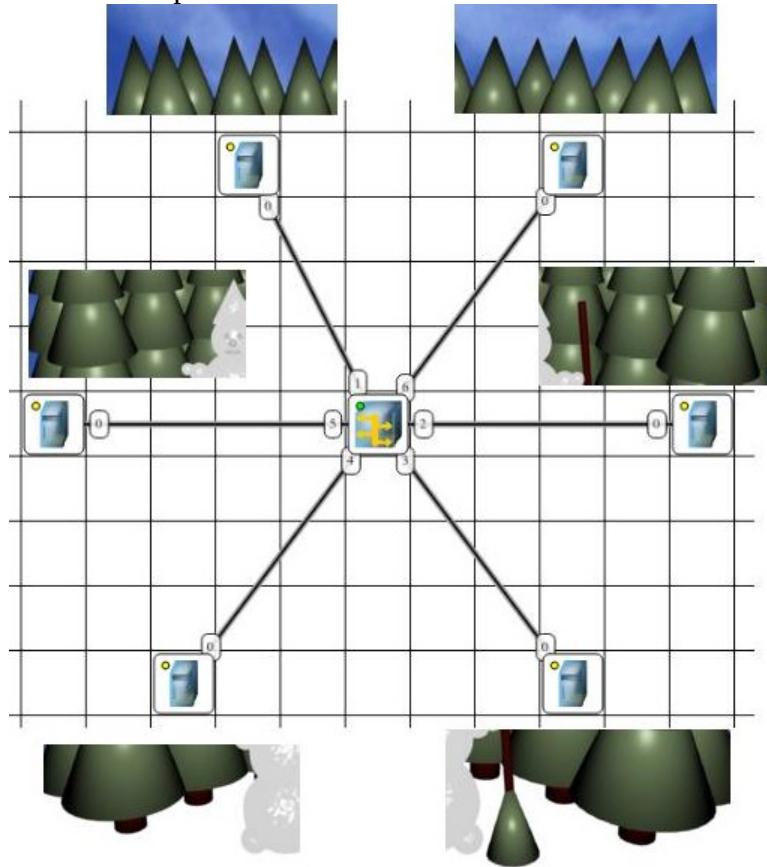


Figure 4: Network render configuration after speed test (adjusted image strip widths)

- **Gabbler:** Desarrollado por Khait Denis, al ser una persona de origen ruso la documentación esta toda en dicho idioma, divide el trabajo por partes y utiliza a los usuarios registrados para trabajar cada render. Soporta la versión 2.78a de Blender.



Hasta aquí todos los proyectos dividen el trabajo a realizar, y luego, una vez realizado juntan las partes para mostrar el resultado final. Como se ha visto anteriormente, algunos utilizan el poder de procesamiento de los usuarios participes de la red; otros dan el software para poder realizar la distribución de trabajo en forma privada dentro de un dominio propio de computadoras; y por ultimo, están los que montaron un negocio con esto y directamente cobran por el tiempo de renderizado ofreciendo privacidad y seguridad (no es un tema menor compartir estos archivos, ya que si caen en manos malintencionadas, podrían copiar parte de nuestro proyecto).

Lo que se propone es un sistema distribuido con algunas similitudes a los anteriores, desarrollado totalmente en Java, con la característica principal y foco en renderizar muchas imágenes de relativamente baja calidad(cantidad baja de Cycles samples) forzando ruido en diferentes lugares del frame para aplicar el algoritmo de Image Stacking, el cual recorrerá todas las imágenes renderizadas realizando un proceso que consiste en tomar pixel a pixel de una imagen y compararlo con los de otra imagen en la misma posición, se generará una nueva imagen, y en ella se colocará el valor de la media de los píxeles comparados. Al tener poco tiempo de procesamiento se podrá utilizar en equipos no recomendados para hacer render (antiguados), se buscará una baja calidad pero suficiente para obtener un resultado limpio al juntar las imágenes renderizadas.

Aspectos Técnicos

- Manejo de nodos: La comunicación entre los nodos y el servidor central se realizará por RMI. cuando un nodo se conecta al sistema emite un mensaje de saludo al servidor y este lo registra en una lista de Workers conectados. Una vez que el worker se conecto y le envió el mensaje al servidor, invoca un método por RMI al servidor solicitando una tarea para realizar y queda a la espera hasta que haya una tarea (el retorno de la función que envió el worker puede ser: un mensaje diciendo que no hay tareas o puede ser la tarea a realizar). Una vez obtenida la tarea el nodo se pondrá a trabajar en ella. Cuando termine el nodo guardará en una variable las tareas que realizó y volverá a consultar al servidor por mas tareas. Si no hay tareas el nodo borrará sus tareas realizadas (esto es así debido a que si un nodo termina con una tarea y quiere comenzar otra que este en espera el servidor debe saber cual asignarle).

- Manejo de múltiples clientes en simultáneos: Los clientes se comunican al servidor mediante RMI. El cliente invoca un método que se encargará de instanciar una clase (que implementa Runnable) pasando todos los parámetros necesarios para funcionar (como por ejemplo una conexión con el manejador de colas y la lista de los workers activos), luego se lanzará en un thread esta instancia y de esta manera el sistema invocará un thread nuevo por cada solicitud de cliente. Una vez que se resolvió la tarea a realizar, el thread morirá y se devolverá el resultado al cliente (terminando así con la conexión rmi).

- Manejo de Sincronismo: Cuando un trabajo llega al servidor, este lo subirá a la cola queueTrabajos, y lo enviará a los workers que estén pidiendo trabajos para que ellos se lo descarguen en su almacenamiento local dentro de una carpeta temporal(cuando termine de trabajar borrará todos los archivos) y puedan trabajar individualmente por su cuenta.

- Distribución y procesamiento de tareas: Se puede procesar las tareas en dos modalidades distintas: por tiempo y por cantidad de samples.

En la primer modalidad el proceso es mas sencillo, ya que, una vez que el cliente envía el trabajo y el servidor lo recibe comienza una cuenta regresiva definida por la cantidad de segundos que haya ingresado el cliente para dicho trabajo. Los workers reciben el trabajo y con el reciben el tiempo en el que comenzó la cuenta regresiva y la cantidad de segundos. Calculan el tiempo promedio que tardan en realizar cada Sample y con ese tiempo promedio pueden decidir si seguir trabajando o no (si les alcanza el tiempo). Una vez que se termina el tiempo, el servidor toma todos los trabajos almacenados en la cola de trabajos terminados (no sabe que cantidad de samples hay, al ser por tiempo este es un dato que se puede intuir(basado en experiencias anteriores sobre el mismo hardware) pero no saber a ciencia cierta) y los procesa aplicándole el filtro, una vez terminado la aplicación del filtro le devuelve el resultado al cliente. Aquí un dato a tener en cuenta es que si hay dos clientes que envían trabajos, los workers se pondrán a trabajar en uno hasta llegar al tiempo dado y luego se trabajará en el otro trabajo del segundo cliente. Por lo tanto, se “atenderá” de a un cliente a la vez (aunque el sistema permita recibir muchos trabajos juntos).

La segunda modalidad (por cantidad de samples), el servidor recibe el trabajo, lo carga en la cola de trabajos y envía a los workers el nuevo trabajo, los workers trabajarán en el hasta que hayan cumplido la cantidad de samples que se le solicitaba. Si hay otro trabajo que realizar se pondrán a realizar los trabajos que estén pendientes. El servidor irá anotando la cantidad de trabajos que le entregó cada worker, cuando la cantidad de trabajos sea igual a la cantidad que se solicitaba lo dará

al worker por finalizado. Una vez que todos los workers hayan finalizado le aplicará el filtro a las imágenes y le enviará el resultado final al cliente. En caso de que un worker se haya caído el servidor borrará al worker de la lista, lo que ocasionará que también se actualice la lista de nodos que faltan terminar el trabajo, y si no está en la lista workers se eliminará también de la lista de nodos que aun no habían terminado.

- **Manejo de transparencia:** El cliente cuenta con una interfaz grafica básica en la que puede subir un proyecto, definir la modalidad de renderizado (tiempo o por samples), definirá una cantidad de tiempo en segundos o cantidad samples y quedará a la espera de un resultado. Lo referido a servidor y workers le será totalmente transparente ya que no contará con ninguna información de ello, solo su interfaz y una pantalla que le notifique si se terminó de procesar correctamente o si falló.

- **Contemplar aspectos de eficiencia:** Para tener una mejor eficiencia se implementó el sistema con dos modalidades de trabajo (por tiempo y por cantidad de samples), cada modalidad tiene aspectos a favor y en contra. Si se toma la modalidad de tiempo se podrá exprimir al máximo el hardware de cada equipo durante el tiempo que se haya indicado. La contra de esta modalidad es si hay múltiples clientes enviando archivos, ya que su déficit se encuentra en que se procesará de a un archivo para poder usar el trabajo de todos los nodos. Si se toma la modalidad de trabajo por samples, se deberá esperar hasta que el equipo mas lento entregue el ultimo fragmento de trabajo para empezar a aplicar el filtro. El aspecto a favor de esta modalidad, es que si hay múltiples clientes, los workers con hardware mas rápidos, irán despachando las tareas a medida que las vayan terminando.

- **Contemplar alta disponibilidad:** Este sistema es muy dependiente del middleware de colas, y para tener una alta disponibilidad se utilizará un cluster de colas replicado en dos equipos diferentes, dando la posibilidad a recuperarse en el caso que se caiga el equipo que tenía el servidor de RabbitMQ. En uno de los dos equipos se correrá el Servidor primario y en el otro equipo correrá un servidor de respaldo. El servidor de respaldo estará checkeando cada 10 segundos si el servidor primario está disponible, en caso de no estarlo, tomará el rol de servidor primario y registrará a los workers, atenderá clientes hasta que el servidor primario este de nuevo disponible, en el caso de que justo este realizando trabajos, esperará que la cola de trabajos este vacía para volver a ser el servidor de respaldo.

- **Contemplar escalabilidad del sistema:** Cada worker envía un mensaje al server cada 1 minuto con el fin de comunicarle que está vivo. El servidor cuando necesita tareas controla cada 1 segundo el estado de los nodos, si la ultima vez que envió un mensaje el worker es hace mas de 100 segundos, el servidor elimina al worker de su lista de workers.

Luego, por parte de los clientes, la cantidad máxima de trabajos que se podrá delegar al servidor sera proporcional a la cantidad de espacio disponible que tenga el disco rígido del servidor de colas. Si hay muchos clientes solicitando trabajo, se los atenderá de a uno por vez* y en forma de cola.
**Si un worker ya terminó con el trabajo y detecta que hay más trabajos pendientes se pondrá a trabajar en el próximo trabajo.*

- **Registro de actividades del sistema:** Se utiliza un framework de log llamado LOGBack, el cual registra todos los acontecimientos(Información, errores, controles de debug) de cada entidad del sistema en archivos de log por separado. Indicando el tiempo y fecha de cada mensaje registrado.

- Manejo de aspectos de seguridad: El sistema se comunica entre Worker – Servidor – Cliente por RMI, en caso de necesitar implementar seguridad se podrá instalar en el servidor el “RMISecurityManager” y podremos definir los permisos a gusto mediante el fichero “java.policy”

- Métricas de Rendimiento: Finalmente, con el objetivo de evaluar las prestaciones y confiabilidad del sistema distribuido, deberán tomarse métricas como: Tiempo de respuesta, latencia, overhead del protocolo, Throughput (Número de trabajos por hora), utilización del sistema, fallas y errores, otros.

Hardware del laboratorio:

PC1: (Servidor y Worker)

Microporcesador: AMD FX8350 4 núcleos, 8 hilos @ 4GHz

Memoria: 16Gb DDR3 1333Mhz

Disco: SSD 240Gb Kingstone A400

VGA: XFX RX560 4Gb

PC2: (Worker)

Microporcesador: I3 3240 2 nucleos, 4 hilos @ 3,40GHz

Memoria: 6Gb DDR3 1600Mhz

Disco: SSD 120Gb Sandisk

VGA: Intel HD Graphics 2500. Integrada.

PC3: (Worker)

Microporcesador: I5 3320M @ 2,60GHz

Memoria: 8Gb DDR3 1600Mhz

Disco: SSD 180Gb Intel

VGA: Intel HD Graphics 4000. Integrada.

PC4 (Worker)

Microporcesador: AMD FX6200 6 núcleos, 6 hilos @ 3,80GHz

Memoria: 8Gb DDR3 1600Mhz

Disco: HDD 500gb Hitachi

VGA: AMD R7 240 2Gb

Todas los equipos eventualmente pueden hacer de cliente, ya que los recursos que requiere el cliente son mínimos y no implican bajas en el rendimiento.

Archivos necesarios para cada entidad:

Worker: Necesita la carpeta “WorkerData” en la que se encuentra:

- Blender-app: Aplicacion del blender, preferentemente la version 2.8
- BlendFiles: Carpeta temporal en la que se almacenarán los .blend
- RenderedImages: Carpeta temporal en la que se almacenarán las imágenes procesadas
- defaultRenderOptions.py: Script que se utiliza para configurar el renderizado
- randomSeed.py: Script que se aplica a cada .blend para producir ruido aleatorio
- scriptTemplate.txt: Template que se utiliza para crear el defaultRenderOptions.py con datos del usuario

- workerConfig.json: Archivo de configuracion con las siguientes configuraciones:
 - Server:
 - IP: Direccion ip del servidor primario
 - IPBak: Direccion ip del servidor de respaldo
 - Port: Puerto RMI
 - FTP: Puerto FTP
 - Queue:
 - IP: Direccion ip del servidor de colas
 - IPBak: Direccion ip del servidor de colas
 - Port: Puerto del servidor de colas
 - User: Usuario usado para publicar en el servidor de colas
 - Pass: Contraseña usada para publicar en el servidor de colas
 - Paths:
 - "myBlendDir": Directorio de los archivos .blend
 - "myBlenderApp": Path a la aplicación de blender.exe
 - "myFinishedWorks": Directorio donde se almacenaran temporalmente las imágenes procesadas

Servidor: Necesita la carpeta ServerData que contiene:

- Directorio FTP: En el se encuentran los archivos que pedirá el worker en caso de no tener la aplicación de blender.
- White.list: Control de usuarios del servidor ftp.
- ServerConfig.Json:
 - Server:
 - "ip": Direccion ip del servidor principal
 - "ipBak": Direccion ip del servidor de respaldo
 - Queue:
 - IP: Direccion ip del servidor de colas
 - IPBak: Direccion ip del servidor de colas
 - Port: Puerto del servidor de colas
 - User: Usuario usado para publicar en el servidor de colas
 - Pass: Contraseña usada para publicar en el servidor de colas
 - RMI:
 - portCli: Puerto para atender a clientes
 - portSv: Puerto para atender a workers y servidor de respaldo
 - FTP:
 - Directory: Directorio raiz que se compartirá por ftp
 - Port: Puerto del servicio de ftp.

Cliente: Necesita el archivo clienteConfig.json en el que se encuentra:

- Server:
 - "ip": Direccion ip del servidor principal
 - "ipBak": Direccion ip del servidor de respaldo
 - "port": Puerto rmi para conectarse con el servidor.

Pruebas de estabilidad y fallas

Se correrá el servidor principal, se lanzará un worker y se detendrá la ejecución del servidor principal, dando lugar a que lo releve el servidor de backup, y por ultimo se volverá a levantar el principal y se demostrará que esta totalmente operativo.

Pantalla del servidor primario:

```
20:34:22 [main] INFO - RabbitMQ inicio correctamente.
20:34:22 [main] DEBUG - File configured, will try loading
20:34:22 [main] DEBUG - File found on file system
1
Thread #1
{miaFplet=jusacco.TPFinal.Servidor.Tools.ServerFtp$2@382db087}
20:34:22 [main] INFO - FTP Configurado correctamente. Listo para usar en puerto:2121. Compartiendo carpeta: C:\Users\sacco\eclipse-workspace\TPFinal\ServerData\ftp\
20:34:22 [main] INFO - Levantando servidor RMI...
20:34:22 [main] INFO - Servidor RMI{
20:34:22 [main] INFO -     Client:RegistryImpl[UnicastServerRef [liveRef: [endpoint:[192.168.0.108:9000](local),objID:[0:0:0, 0]]]]
20:34:22 [main] INFO -     Server:RegistryImpl[UnicastServerRef [liveRef: [endpoint:[192.168.0.108:9001](local),objID:[0:0:0, 0]]]]
}
20:34:29 [RMI TCP Connection(4)-192.168.0.108] DEBUG - Registrando nuevo worker: 192.168.0.108
```

Servidor corriendo normalmente, el worker esperando trabajos.

Pantalla del servidor de respaldo:

```
20:34:23.208 [main] INFO jusacco.TPFinal.Servidor.Servidor - El servidor principal esta ON. Esperando estar listo para migrar al principal
20:34:23.224 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Obteniendo servicios RMI.
20:34:23.224 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Obteniendo stub...
20:34:23.224 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Attemps 0
20:34:23.239 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:34:23.252 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:34:43.259 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:34:53.262 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:35:03.270 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:35:13.276 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:35:23.281 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:35:33.283 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:35:43.297 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:35:53.297 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
```

*Servidor principal: OK corresponde a que el servidor le contesto positivamente sobre su peticion de disponibilidad.

ACCIÓN: Se cierra el servidor primario.

Pantalla del servidor de respaldo relevando al servidor primario:

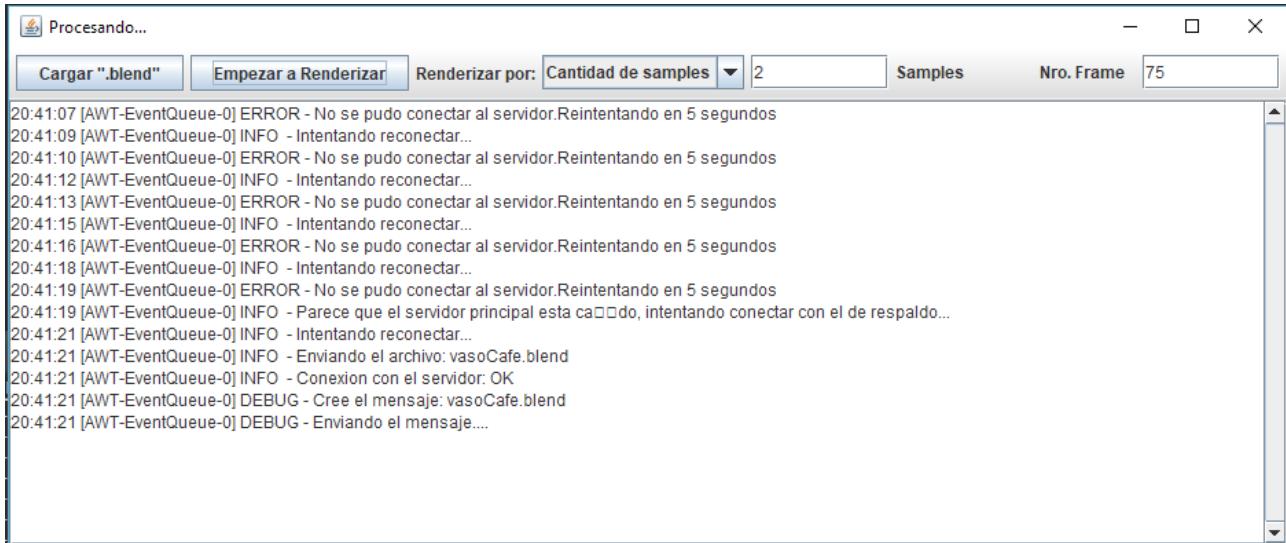
```
20:36:35.353 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: Caido. Creando nuevo principal
20:36:35.353 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Levantando servidor en: 192.168.0.110
20:36:35.384 [main] INFO jusacco.TPFinal.Servidor.Servidor - RabbitMQ inicio correctamente.
20:36:41.685 [main] DEBUG org.apache.ftpserver.usermanager.impl.PropertiesUserManager - File configured, will try loading
20:36:41.685 [main] DEBUG org.apache.ftpserver.usermanager.impl.PropertiesUserManager - File found on file system
1
Thread #1
{miaFplet=jusacco.TPFinal.Servidor.Tools.ServerFtp$2@cd705f}
20:36:41.685 [main] INFO jusacco.TPFinal.Servidor.Servidor - FTP Configurado correctamente. Listo para usar en puerto:2121. Compartiendo carpeta: C:\Users\intel\Desktop\TPFinal\packages\Server_0.8\ServerData\ftp\
20:36:41.685 [main] INFO jusacco.TPFinal.Servidor.Servidor - Levantando servidor RMI...
20:36:41.685 [main] INFO jusacco.TPFinal.Servidor.Servidor - Servidor RMI{
20:36:41.685 [main] INFO jusacco.TPFinal.Servidor.Servidor -     Client:RegistryImpl[UnicastServerRef [liveRef: [endpoint:[192.168.0.110:9000](local),objID:[0:0:0, 0]]]]
20:36:41.685 [main] INFO jusacco.TPFinal.Servidor.Servidor -     Server:RegistryImpl[UnicastServerRef [liveRef: [endpoint:[192.168.0.110:9001](local),objID:[0:0:0, 0]]]]
}
20:36:51.335 [RMI TCP Connection(31)-192.168.0.108] DEBUG jusacco.TPFinal.Servidor.WorkerAction - Registrando nuevo worker: 192.168.0.108
```

Pantalla del worker:

```
20:36:38 [main] INFO - Conexion perdida con el servidor principal, conectando con el servidor backup
20:36:38 [main] INFO - Borrando los archivos ya existentes en las carpetas temporales.
20:36:38 [main] INFO - Obteniendo servicios RMI.
20:36:38 [main] INFO - Obteniendo stub...
20:36:39 [main] ERROR - RMI Error: Connection refused to host: 192.168.0.110; nested exception is:
    java.net.ConnectException: Connection refused: connect
20:36:39 [main] INFO - Re-intentando conectar al servidor principal: 192.168.0.110:9001
20:36:40 [main] INFO - Re-intentando en...
20:36:41 [main] INFO - Re-intentando en...
20:36:42 [main] INFO - Re-intentando en...
20:36:43 [main] INFO - Re-intentando en...
20:36:44 [main] INFO - Re-intentando en...
20:36:44 [main] INFO - Obteniendo servicios RMI.
20:36:44 [main] INFO - Obteniendo stub...
20:36:45 [main] ERROR - RMI Error: Connection refused to host: 192.168.0.108; nested exception is:
    java.net.ConnectException: Connection refused: connect
20:36:45 [main] INFO - Re-intentando conectar al servidor backup: 192.168.0.108:9001
20:36:46 [main] INFO - Re-intentando en...
20:36:47 [main] INFO - Re-intentando en...
20:36:48 [main] INFO - Re-intentando en...
20:36:49 [main] INFO - Re-intentando en...
20:36:50 [main] INFO - Re-intentando en...
20:36:50 [main] INFO - Obteniendo servicios RMI.
20:36:50 [main] INFO - Obteniendo stub...
```

ACCIÓN: Ingresa un cliente y pide un trabajo (mientras esta el servidor de respaldo)

Pantalla cliente ya con el trabajo terminado y recibido:



*Los mensajes de error corresponden a que primero se intentó conectar al servidor primario y luego se conectó con el servidor de respaldo.

**La GUI que se muestra corresponde a una interfaz de usuario para desarrolladores, la interfaz para todos los usuarios será aun mas basica y sin mostrar toda la información interna del sistema.

Pantalla servidor de respaldo procesando un pedido:

```
20:41:21.957 [RMI TCP Connection(34)-192.168.0.108] INFO jusacco.TPFinal.Servidor.Servidor - Se conecto el cliente192.168.0.108
20:41:22.957 [RMI TCP Connection(34)-192.168.0.108] DEBUG jusacco.TPFinal.Servidor.Servidor - -----
20:41:22.957 [RMI TCP Connection(34)-192.168.0.108] DEBUG jusacco.TPFinal.Servidor.Servidor - ##### Thread[RMI TCP Connection(34)-192.168.0.108,5,RMI Runtime] #####
20:41:22.957 [RMI TCP Connection(34)-192.168.0.108] DEBUG jusacco.TPFinal.Servidor.Servidor - | Pedido de trabajo: vasoCafe.blend
20:41:22.957 [RMI TCP Connection(34)-192.168.0.108] DEBUG jusacco.TPFinal.Servidor.Servidor - -----
20:41:22.957 [Thread-3] INFO jusacco.TPFinal.Servidor.ThreadServer - Trabajando en: vasoCafe.blend Modalidad: Por samples. Cantidad: 2 Samples
20:41:22.957 [Thread-3] INFO jusacco.TPFinal.Servidor.ThreadServer - Tiempo inicio: 20:41:22.057
20:41:23.098 [RMI TCP Connection(32)-192.168.0.108] DEBUG jusacco.TPFinal.Servidor.WorkerAction - 192.168.0.108: realizedWorks[]| listaTrabajos[vasoCafe.blend:192.168.0.108]
20:41:23.098 [RMI TCP Connection(32)-192.168.0.108] DEBUG jusacco.TPFinal.Servidor.WorkerAction - 192.168.0.108: Delegando tarea[vasoCafe.blend:192.168.0.108]
20:41:23.098 [RMI TCP Connection(32)-192.168.0.108] DEBUG jusacco.TPFinal.Servidor.WorkerAction - Result lenght: 28
20:41:32.665 [Thread-3] INFO jusacco.TPFinal.Servidor.ThreadServer - -----
20:41:32.665 [Thread-3] INFO jusacco.TPFinal.Servidor.ThreadServer - Worker: 192.168.0.108 Render count:1 @ vasoCafe.blend:192.168.0.108
20:41:32.665 [Thread-3] INFO jusacco.TPFinal.Servidor.ThreadServer - -----
20:41:32.665 [Thread-3] DEBUG jusacco.TPFinal.Servidor.ThreadServer - Cant. Worker que terminaron: 0 Cant. Worker total: 1
20:41:32.665 [Thread-3] DEBUG jusacco.TPFinal.Servidor.ThreadServer - Cant. Img: 1 Cantidad esperada: 2
20:41:32.665 [Thread-3] DEBUG jusacco.TPFinal.Servidor.ThreadServer - Nodos sin terminar: [192.168.0.108]
20:41:33.665 [Thread-3] DEBUG jusacco.TPFinal.Servidor.ThreadServer - Nodos sin terminar: [192.168.0.108]
20:41:34.665 [Thread-3] DEBUG jusacco.TPFinal.Servidor.ThreadServer - Nodos sin terminar: [192.168.0.108]
20:41:35.676 [Thread-3] DEBUG jusacco.TPFinal.Servidor.ThreadServer - Nodos sin terminar: [192.168.0.108]
20:41:36.583 [Thread-3] DEBUG jusacco.TPFinal.Servidor.ThreadServer - Nodos sin terminar: [192.168.0.108]
20:41:37.001 [Thread-3] INFO jusacco.TPFinal.Servidor.ThreadServer - -----
20:41:37.001 [Thread-3] INFO jusacco.TPFinal.Servidor.ThreadServer - Worker: 192.168.0.108 Render count:2 @ vasoCafe.blend:192.168.0.108
20:41:37.001 [Thread-3] INFO jusacco.TPFinal.Servidor.ThreadServer - -----
20:41:37.001 [Thread-3] DEBUG jusacco.TPFinal.Servidor.ThreadServer - Cant. Worker que terminaron: 1 Cant. Worker total: 1
20:41:37.001 [Thread-3] DEBUG jusacco.TPFinal.Servidor.ThreadServer - Cant. Img: 2 Cantidad esperada: 2
20:41:37.001 [Thread-3] DEBUG jusacco.TPFinal.Servidor.ThreadServer - Termine: vasoCafe.blend
20:41:37.079 [Thread-3] DEBUG jusacco.TPFinal.Servidor.ThreadServer - Tengo que borrar vasoCafe.blend
20:41:37.079 [Thread-3] INFO jusacco.TPFinal.Servidor.ThreadServer - Cantidad de imagenes: 2
20:41:37.983 [Thread-3] INFO jusacco.TPFinal.Servidor.ThreadServer - endTime aplicarFiltro: 20:41:37.983
20:41:37.983 [Thread-3] INFO jusacco.TPFinal.Servidor.ThreadServer - Delta time aplicarFiltro: PT15.926S
```

Continua pantalla...

```
20:41:37.001 [Thread-3] INFO jusacco.TPFinal.Servidor.ThreadServer - -----
20:41:37.001 [Thread-3] INFO jusacco.TPFinal.Servidor.ThreadServer - Worker: 192.168.0.108 Render count:2 @ vasoCafe.blend:192.168.0.108
20:41:37.001 [Thread-3] INFO jusacco.TPFinal.Servidor.ThreadServer - -----
20:41:37.001 [Thread-3] DEBUG jusacco.TPFinal.Servidor.ThreadServer - Cant. Worker que terminaron: 1 Cant. Worker total: 1
20:41:37.001 [Thread-3] DEBUG jusacco.TPFinal.Servidor.ThreadServer - Cant. Img: 2 Cantidad esperada: 2
20:41:37.001 [Thread-3] DEBUG jusacco.TPFinal.Servidor.ThreadServer - Termine: vasoCafe.blend
20:41:37.079 [Thread-3] DEBUG jusacco.TPFinal.Servidor.ThreadServer - Tengo que borrar vasoCafe.blend
20:41:37.079 [Thread-3] INFO jusacco.TPFinal.Servidor.ThreadServer - Cantidad de imagenes: 2
20:41:37.983 [Thread-3] INFO jusacco.TPFinal.Servidor.ThreadServer - endTime aplicarFiltro: 20:41:37.983
20:41:37.983 [Thread-3] INFO jusacco.TPFinal.Servidor.ThreadServer - Delta time aplicarFiltro: PT15.926S
20:41:38.332 [RMI TCP Connection(34)-192.168.0.108] DEBUG jusacco.TPFinal.Servidor.Servidor - -----
20:41:38.332 [RMI TCP Connection(34)-192.168.0.108] DEBUG jusacco.TPFinal.Servidor.Servidor - ##### Thread[RMI TCP Connection(34)-192.168.0.108,5,RMI Runtime] #####
20:41:38.332 [RMI TCP Connection(34)-192.168.0.108] DEBUG jusacco.TPFinal.Servidor.Servidor - | Lista trabajo: [vasoCafe.blend:192.168.0.108]
20:41:38.332 [RMI TCP Connection(34)-192.168.0.108] DEBUG jusacco.TPFinal.Servidor.Servidor - | Acaba de terminar: vasoCafe.blend
20:41:38.332 [RMI TCP Connection(34)-192.168.0.108] DEBUG jusacco.TPFinal.Servidor.Servidor - | Eliminando... vasoCafe.blend
20:41:38.332 [RMI TCP Connection(34)-192.168.0.108] DEBUG jusacco.TPFinal.Servidor.Servidor - -----
```

Continua pantalla. Worker ya realizando el trabajo:

```
20:36:50 [main] INFO - Obteniendo servicios RMI.
20:36:50 [main] INFO - Obteniendo stub...
20:41:23 [main] DEBUG - Nuevo trabajo: vasoCafe.blend:192.168.0.108
20:41:23 [main] INFO - La cola tiene: 1 mensajes
20:41:23 [main] INFO - Mensaje: vasoCafe.blend:192.168.0.108 | trabajo: vasoCafe.blend:192.168.0.108
20:41:23 [main] INFO - Encontre el trabajo. Dando NACK. La cola tiene ahora: 1 mensajes
20:41:26 [main] INFO - Pre-configurando el archivo .blend
CMD: C:\Users\sacco\eclipse-workspace\TPFinal\WorkerData\Blender-app\blender.exe -b "C:\Users\sacco\eclip
Line: Read prefs: C:\Users\sacco\AppData\Roaming\Blender Foundation\Blender\2.80\config\userpref.blend
Line: found bundled python: C:\Users\sacco\eclipse-workspace\TPFinal\WorkerData\Blender-app\blender-windows\2.80\python
Line: Read blend: C:\Users\sacco\eclipse-workspace\TPFinal\WorkerData\BlendFiles\vasoCafe.blend
Line:
Line: Blender quit
20:41:27 [main] INFO - Calculando Seed Random
Line: Read prefs: C:\Users\sacco\AppData\Roaming\Blender Foundation\Blender\2.80\config\userpref.blend
Line: found bundled python: C:\Users\sacco\eclipse-workspace\TPFinal\WorkerData\Blender-app\blender-windows\2.80\python
Line: Read blend: C:\Users\sacco\eclipse-workspace\TPFinal\WorkerData\BlendFiles\vasoCafe.blend
Line:
```

Pantalla worker trabajo realizado:

```
Line: Fra:75 Mem:99.31M (0.00M, Peak 99.66M) | Time:00:00.69 | Rendering 62 / 64 samples
Line: Fra:75 Mem:99.31M (0.00M, Peak 99.66M) | Time:00:00.69 | Updating Probes
Line: Fra:75 Mem:99.31M (0.00M, Peak 99.66M) | Time:00:00.69 | Rendering 63 / 64 samples
Line: Fra:75 Mem:99.31M (0.00M, Peak 99.66M) | Time:00:00.70 | Updating Probes
Line: Fra:75 Mem:99.31M (0.00M, Peak 99.66M) | Time:00:00.70 | Rendering 64 / 64 samples
Line: Fra:75 Mem:58.76M (0.00M, Peak 99.66M) | Time:00:00.78 | Sce: Scene Ve:0 Fa:0 La:0
Line: Saved: 'C:\Users\sacco\.eclipse-workspace\TPFinal\WorkerData\RenderedImages\2 from frame 0075.png'
Line: Time: 00:01.64 (Saving: 00:00.85)
Line:
Line: Error: Not freed memory blocks: 229, total unfreed memory 0.082458 MB
Line:
Line: Blender quit
20:41:35 [main] INFO - =====Termino=====
C:\Users\sacco\.eclipse-workspace\TPFinal\WorkerData//BlendFiles\vasoCafe.blend -> Eliminado.
C:\Users\sacco\.eclipse-workspace\TPFinal\WorkerData//BlendFiles\vasoCafe.blend1 -> Eliminado.
C:\Users\sacco\.eclipse-workspace\TPFinal\WorkerData//RenderedImages\1 from frame 0075.png -> Eliminado.
C:\Users\sacco\.eclipse-workspace\TPFinal\WorkerData//RenderedImages\2 from frame 0075.png -> Eliminado.
```

ACCIÓN: Vuelve a conectarse el servidor principal

Pantalla servidor principal:

```
20:53:26 [main] INFO - RabbitMQ inicio correctamente.
20:53:26 [main] DEBUG - File configured, will try loading
20:53:26 [main] DEBUG - File found on file system
1
Thread #1
{miaFplet=jusacco.TPFinal.Servidor.Tools.ServerFtp$2@382db087}
20:53:26 [main] INFO - FTP Configurado correctamente. Listo para usar en puerto:2121. Compartiendo carpeta: C:\Users\sacco\eclipse-workspace\miaFplet
20:53:26 [main] INFO - Levantando servidor RMI...
20:53:26 [main] INFO - Servidor RMI{
20:53:26 [main] INFO -     Client:RegistryImpl[UnicastServerRef [liveRef: [endpoint:[192.168.0.108:9000](local),objID:[0:0:0, 0]]]]
20:53:26 [main] INFO -     Server:RegistryImpl[UnicastServerRef [liveRef: [endpoint:[192.168.0.108:9001](local),objID:[0:0:0, 0]]]]
20:53:26 [main] }
20:53:28 [RMI TCP Connection(4)-192.168.0.108] DEBUG - Registrando nuevo worker: 192.168.0.108
```

Pantalla worker:

```
20:53:28 [main] INFO - Perdi la conexion con el Backup server, conectando a servidor primario.
20:53:28 [main] INFO - <-- [STEP 1] - LEYENDO ARCHIVO DE CONFIGURACION -->
20:53:28 [main] INFO - <-- [STEP 2] - PREPARANDO EL BANCO DE TRABAJO -->
20:53:28 [main] INFO - Borrando los archivos ya existentes en las carpetas temporales.
20:53:28 [main] INFO - <-- [STEP 3] - REALIZANDO CONEXION RMI -->
20:53:28 [main] INFO - Obteniendo servicios RMI.
20:53:28 [main] INFO - Obteniendo stub...
20:53:28 [main] INFO - <-- [STEP 4] - LANZANDO THREAD ALIVE -->
20:53:28 [main] INFO - <-- [STEP 5] - REALIZANDO CONEXION CON RABBITMQ -->
20:53:28 [main] INFO - RabbitMQ: Conexion creada con exito.
20:53:28 [main] INFO - <-- [STEP 6] - REVISANDO ARCHIVOS NECESARIOS -->
20:53:28 [main] INFO - C:\Users\sacco\eclipse-workspace\TPFinal\WorkerData ---->Directorio
20:53:28 [main] INFO - C:\Users\sacco\eclipse-workspace\TPFinal\WorkerData\Blender-app ---->Directorio
20:53:29 [main] INFO - Obteniendo tamano de: C:\Users\sacco\eclipse-workspace\TPFinal\WorkerData\Blender-app MB:453062
20:53:29 [main] INFO - Blender ----> LISTO
20:53:29 [main] INFO - <-- [STEP 6] - ESPERANDO TRABAJOS -->
```

Pantalla servidor de respaldo:

```
20:53:27.434 [main] INFO jusacco.TPFinal.Servidor - El servidor principal esta ON. Esperando estar listo para migrar al principal
20:53:27.465 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Obteniendo servicios RMI.
20:53:27.465 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Obteniendo stub...
20:53:27.465 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Attemp 0
20:53:27.465 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:53:27.699 [RMI TCP Connection(32)-192.168.0.108] DEBUG jusacco.TPFinal.Servidor.WorkerAction - 192.168.0.108: realizedWorks[vasoCafe.blend:192.ajos[1234567890exit]
20:53:27.699 [RMI TCP Connection(32)-192.168.0.108] DEBUG jusacco.TPFinal.Servidor.WorkerAction - 192.168.0.108: Delegando tarea[1234567890exit]
20:53:27.699 [RMI TCP Connection(32)-192.168.0.108] DEBUG jusacco.TPFinal.Servidor.WorkerAction - Result lenght: 14
20:53:37.468 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:53:47.470 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:53:57.484 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:54:07.495 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:54:17.495 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:54:27.498 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:54:37.500 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:54:47.500 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:54:57.513 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
```

*Delegando tarea[1234567890exit] se refiere a un punto de corte para avisarle al worker que volvió el servidor principal y el servidor de respaldo pasará a ser servidor de respaldo otra vez. Esto es debido a una limitación técnica en RMI, ya que, RMI no cierra las conexiones desde el servidor hacia el cliente dando como resultado que el worker se quede conectado a el servidor de respaldo.

ACCION: El recién vuelto servidor principal tomará una tarea y hará trabajar al worker

Pantalla servidor primario:

```
20:53:28 [RMI TCP Connection(4)-192.168.0.108] DEBUG - Registrando nuevo worker: 192.168.0.108
20:58:58 [RMI TCP Connection(7)-192.168.0.108] INFO - Se conecto el cliente192.168.0.108
20:58:59 [RMI TCP Connection(7)-192.168.0.108] DEBUG - -----
20:58:59 [RMI TCP Connection(7)-192.168.0.108] DEBUG - |## Thread[RMI TCP Connection(7)-192.168.0.108,5,RMI Runtime] ##
20:58:59 [RMI TCP Connection(7)-192.168.0.108] DEBUG - |Pedido de trabajo: Tienda.blend
20:58:59 [RMI TCP Connection(7)-192.168.0.108] DEBUG - -----
20:58:59 [Thread-1] INFO - Trabajando en: Tienda.blend Modalidad: Por samples. Cantidad: 2 Samples
20:58:59 [Thread-1] INFO - Tiempo inicio: 20:58:59.854
20:59:02 [RMI TCP Connection(4)-192.168.0.108] DEBUG - 192.168.0.108: realizedWorks[vasoCafe.blend:192.168.0.108]| listaTrabajos[Tienda.blend:192.168.0.108]
20:59:02 [RMI TCP Connection(4)-192.168.0.108] DEBUG - 192.168.0.108: Delegando tarea[Tienda.blend:192.168.0.108]
20:59:02 [RMI TCP Connection(4)-192.168.0.108] DEBUG - Result lenght: 26
20:59:31 [Thread-1] INFO - ---
20:59:31 [Thread-1] INFO - Worker: 192.168.0.108 Render count:1 @ Tienda.blend:192.168.0.108
20:59:31 [Thread-1] INFO - ---
20:59:31 [Thread-1] DEBUG - Cant. Worker que terminaron: 0 Cant. Worker total: 1
20:59:31 [Thread-1] DEBUG - Cant. Img: 1 Cantidad esperada: 2
20:59:31 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108]
20:59:32 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108]
```

Continua pantalla...

```
20:59:45 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108]
20:59:46 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108]
20:59:47 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108]
20:59:48 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108]
20:59:49 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108]
20:59:50 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108]
20:59:51 [Thread-1] INFO - ---
20:59:51 [Thread-1] INFO - Worker: 192.168.0.108 Render count:2 @ Tienda.blend:192.168.0.108
20:59:51 [Thread-1] INFO - ---
20:59:51 [Thread-1] DEBUG - Cant. Worker que terminaron: 1 Cant. Worker total: 1
20:59:51 [Thread-1] DEBUG - Cant. Img: 2 Cantidad esperada: 2
20:59:51 [Thread-1] INFO - Termine: Tienda.blend
20:59:52 [Thread-1] DEBUG - Tengo que borrar Tienda.blend
20:59:52 [Thread-1] INFO - Cantidad de imagenes: 2
20:59:53 [Thread-1] INFO - endTime aplicarFiltro: 20:59:53.864
20:59:53 [Thread-1] INFO - Delta time aplicarFiltro: PT54.01S
20:59:54 [RMI TCP Connection(7)-192.168.0.108] DEBUG - -----
20:59:54 [RMI TCP Connection(7)-192.168.0.108] DEBUG - |## Thread[RMI TCP Connection(7)-192.168.0.108,5,RMI Runtime] ##
20:59:54 [RMI TCP Connection(7)-192.168.0.108] DEBUG - |Lista trabajo: [Tienda.blend:192.168.0.108]
20:59:54 [RMI TCP Connection(7)-192.168.0.108] DEBUG - |Acaba de terminar: Tienda.blend
20:59:54 [RMI TCP Connection(7)-192.168.0.108] DEBUG - |Eliminando... Tienda.blend
20:59:54 [RMI TCP Connection(7)-192.168.0.108] DEBUG - -----
```

Pantalla worker:

```
Line: Fra:75 Mem:763.03M (0.00M, Peak 1791.06M) | Time:00:06.46 | Mem:521.90M, Peak:533.89M | Scene, RenderLayer |
Line: Fra:75 Mem:764.30M (0.00M, Peak 1791.06M) | Time:00:06.46 | Mem:523.17M, Peak:533.89M | Scene, RenderLayer |
Line: Fra:75 Mem:764.31M (0.00M, Peak 1791.06M) | Time:00:06.46 | Mem:522.92M, Peak:533.89M | Scene, RenderLayer |
Line: Fra:75 Mem:764.31M (0.00M, Peak 1791.06M) | Time:00:06.46 | Mem:523.17M, Peak:533.89M | Scene, RenderLayer |
Line: Fra:75 Mem:764.31M (0.00M, Peak 1791.06M) | Time:00:06.46 | Mem:523.17M, Peak:533.89M | Scene, RenderLayer |
Line: Fra:75 Mem:764.31M (0.00M, Peak 1791.06M) | Time:00:15.29 | Mem:523.17M, Peak:533.89M | Scene, RenderLayer |
Line: Fra:75 Mem:239.86M (0.00M, Peak 1791.06M) | Time:00:15.37 | Sce: Scene Ve:0 Fa:0 La:0
Line: Saved: 'C:\Users\sacco\eclipse-workspace\TPFinal\WorkerData\RenderedImages\2 from frame 0075.png'
Line: Time: 00:15.75 (Saving: 00:00.37)
Line:
Line:
Line: Blender quit
20:59:50 [main] INFO - =====Termino=====
C:\Users\sacco\eclipse-workspace\TPFinal\WorkerData//BlendFiles/Tienda.blend -> Eliminado.
C:\Users\sacco\eclipse-workspace\TPFinal\WorkerData//BlendFiles/Tienda.blend1 -> Eliminado.
C:\Users\sacco\eclipse-workspace\TPFinal\WorkerData//RenderedImages/1 from frame 0075.png -> Eliminado.
C:\Users\sacco\eclipse-workspace\TPFinal\WorkerData//RenderedImages/2 from frame 0075.png -> Eliminado.
```

Pantalla servidor de respaldo:

```
20:53:37.468 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:53:47.470 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:53:57.484 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:54:07.495 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:54:17.495 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:54:27.498 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:54:37.500 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:54:47.500 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:54:57.513 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
20:55:07.518 [main] INFO jusacco.TPFinal.Servidor.BackupServer - Servidor principal: OK.
```

ACCIÓN: Se pedirá un trabajo, dos worker lo comenzarán a hacer y uno intencionalmente sera detenido.

Pantalla servidor primario:

```
21:05:42 [main] INFO - RabbitMQ inicio correctamente.
21:05:42 [main] DEBUG - File configured, will try loading
21:05:42 [main] DEBUG - File found on file system
1
Thread #1
{miaFtpplet=jusacco.TPFinal.Servidor.Tools.ServerFtp$2@382db087}
21:05:42 [main] INFO - FTP Configurado correctamente. Listo para usar en puerto:2121. Compartiendo carpeta: C:\Users\sacco\eclipse-workspace\TPFinal\ServerData\\ftp\
21:05:42 [main] INFO - Levantando servidor RMI...
21:05:42 [main] INFO - Servidor RMI{
21:05:42 [main] INFO -     Client:RegistryImpl[UnicastServerRef [liveRef: [endpoint:[192.168.0.108:9000](local),objID:[0:0:0, 0]]]]
21:05:42 [main] INFO -     Server:RegistryImpl[UnicastServerRef [liveRef: [endpoint:[192.168.0.108:9001](local),objID:[0:0:0, 0]]]]
}
21:05:48 [RMI TCP Connection(4)-192.168.0.108] DEBUG - Registrando nuevo worker: 192.168.0.108
21:07:40 [RMI TCP Connection(6)-192.168.0.110] DEBUG - Registrando nuevo worker: 192.168.0.110
```

Pantalla servidor primario tomando tarea y delegándola:

```
21:08:45 [RMI TCP Connection(8)-192.168.0.108] INFO - Se conecto el cliente192.168.0.108
21:08:45 [RMI TCP Connection(8)-192.168.0.108] DEBUG - 
21:08:45 [RMI TCP Connection(8)-192.168.0.108] DEBUG - |### Thread[RMI TCP Connection(8)-192.168.0.108,5,RMI Runtime] ###
21:08:45 [RMI TCP Connection(8)-192.168.0.108] DEBUG - | Pedido de trabajo: alfombralumieb.blend
21:08:45 [RMI TCP Connection(8)-192.168.0.108] DEBUG - 
21:08:45 [Thread-1] INFO - Trabajando en: alfombralumieb.blend Modalidad: Por samples. Cantidad: 10 Samples
21:08:45 [Thread-1] INFO - Tiempo inicio: 21:08:45.090
21:08:45 [RMI TCP Connection(4)-192.168.0.108] DEBUG - 192.168.0.108: realizedWorks[] listaTrabajos[alfombralumieb.blend:192.168.0.108]
21:08:45 [RMI TCP Connection(4)-192.168.0.108] DEBUG - 192.168.0.108: Delegando tarea[alfombralumieb.blend:192.168.0.108]
21:08:45 [RMI TCP Connection(4)-192.168.0.108] DEBUG - Result lenght: 34
21:08:46 [RMI TCP Connection(6)-192.168.0.110] DEBUG - 192.168.0.110: realizedWorks[] listaTrabajos[alfombralumieb.blend:192.168.0.108]
21:08:46 [RMI TCP Connection(6)-192.168.0.110] DEBUG - 192.168.0.110: Delegando tarea[alfombralumieb.blend:192.168.0.108]
21:08:46 [RMI TCP Connection(6)-192.168.0.110] DEBUG - Result lenght: 34
```

Pantalla worker1 tomando trabajo:

```
21:05:49 [main] INFO - <- [STEP 6] - ESPERANDO TRABAJOS          -->
21:08:46 [main] DEBUG - Nuevo trabajo: alfombralumieb.blend:192.168.0.108
21:08:46 [main] INFO - La cola tiene: 1 mensajes
21:08:46 [main] INFO - Mensaje: alfombralumieb.blend:192.168.0.108 | trabajo: alfombralumieb.blend:192.168.0.108
21:08:46 [main] INFO - Encontre el trabajo. Dando NACK. La cola tiene ahora: 1 mensajes
21:08:49 [main] INFO - Pre-configurando el archivo .blend
CMD: C:\Users\sacco\eclipse-workspace\TPFinal\WorkerData\Blender-app\blender-windows\blender.exe -b "C:\Users\sacco\eclipse-workspace\TPFinal\WorkerData\Blender\2.80\config\userpref.blend"
Line: Read prefs: C:\Users\sacco\AppData\Roaming\Blender Foundation\Blender\2.80\config\userpref.blend
```

ACCIÓN: Se detiene un worker, queda uno solo trabajando.

Pantalla servidor principal con entregas parciales:

```
21:09:54 [Thread-1] INFO - ---
21:09:54 [Thread-1] INFO - Worker: 192.168.0.110 Render count:2 @ alfombralumieb.blend:192.168.0.108
21:09:54 [Thread-1] INFO - ---
21:09:54 [Thread-1] DEBUG - Cant. Worker que terminaron: 0      Cant. Worker total: 2
21:09:54 [Thread-1] DEBUG - Cant. Img: 2      Cantidad esperada: 20
21:09:54 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108, 192.168.0.110]
21:09:55 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108, 192.168.0.110]
21:09:56 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108, 192.168.0.110]
21:09:57 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108, 192.168.0.110]
21:09:58 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108, 192.168.0.110]
21:09:59 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108, 192.168.0.110]
21:10:00 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108, 192.168.0.110]
21:10:01 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108, 192.168.0.110]
21:10:02 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108, 192.168.0.110]
21:10:03 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108, 192.168.0.110]
21:10:03 [Thread-1] INFO - ---
21:10:03 [Thread-1] INFO - Worker: 192.168.0.108 Render count:1 @ alfombralumieb.blend:192.168.0.108
21:10:03 [Thread-1] INFO - Worker: 192.168.0.110 Render count:2 @ alfombralumieb.blend:192.168.0.108
21:10:03 [Thread-1] INFO - ---
21:10:03 [Thread-1] DEBUG - Cant. Worker que terminaron: 0      Cant. Worker total: 2
21:10:03 [Thread-1] DEBUG - Cant. Img: 3      Cantidad esperada: 20
21:10:03 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108, 192.168.0.110]
21:10:04 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108, 192.168.0.110]
```

Pantalla servidor principal elimina al worker inactivo por time out.

```
21:10:48 [Thread-1] INFO - ---
21:10:48 [Thread-1] INFO - Worker: 192.168.0.108 Render count:4 @ alfombralumieb.blend:192.168.0.108
21:10:48 [Thread-1] INFO - Worker: 192.168.0.110 Render count:2 @ alfombralumieb.blend:192.168.0.108
21:10:48 [Thread-1] INFO - ---
21:10:48 [Thread-1] DEBUG - Cant. Worker que terminaron: 0      Cant. Worker total: 2
21:10:48 [Thread-1] DEBUG - Cant. Img: 6      Cantidad esperada: 20
21:10:48 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108, 192.168.0.110]
21:10:49 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108, 192.168.0.110]
21:10:50 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108, 192.168.0.110]
21:10:51 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108, 192.168.0.110]
21:10:52 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108, 192.168.0.110]
21:10:52 [main] ERROR - Eliminando al nodo 192.168.0.110. Motivo time-out de 72 segundos.
```

```
21:10:53 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108]
21:10:54 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108]
21:10:55 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108]
21:10:56 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108]
21:10:57 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108]
21:10:58 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108]
21:10:59 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108]
21:11:00 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108]
21:11:01 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108]
21:11:02 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108]
21:11:03 [Thread-1] DEBUG - Nodos sin terminar: [192.168.0.108]
21:11:04 [Thread-1] INFO - ---
21:11:04 [Thread-1] INFO - Worker: 192.168.0.108 Render count:5 @ alfombralumieb.blend:192.168.0.108
21:11:04 [Thread-1] INFO - Worker: 192.168.0.110 Render count:2 @ alfombralumieb.blend:192.168.0.108
21:11:04 [Thread-1] INFO - ---
21:11:04 [Thread-1] DEBUG - Cant. Worker que terminaron: 0      Cant. Worker total: 1
21:11:04 [Thread-1] DEBUG - Cant. Img: 7      Cantidad esperada: 10
```

Pantalla servidor principal con el trabajo terminado:

```
21:11:50 [Thread-1] INFO - Worker: 192.168.0.108 Render count:8 @ alfombralumieb.blend:192.168.0.108
21:11:50 [Thread-1] INFO - Worker: 192.168.0.110 Render count:2 @ alfombralumieb.blend:192.168.0.108
21:11:50 [Thread-1] INFO - ---
21:11:50 [Thread-1] DEBUG - Cant. Worker que terminaron: 0 Cantidad. Worker total: 1
21:11:50 [Thread-1] DEBUG - CANT. Img: 10 Cantidad esperada: 10
21:11:50 [Thread-1] INFO - Termine: alfombralumieb.blend
21:11:50 [Thread-1] DEBUG - Tengo que borrar alfombralumieb.blend
21:11:50 [Thread-1] INFO - Cantidad de imagenes: 10
21:11:56 [Thread-1] INFO - endTime aplicarFilto: 21:11:56.658
21:11:56 [Thread-1] INFO - Delta time aplicarFilto: PT3M11.568S
21:11:56 [RMI TCP Connection(8)-192.168.0.108] DEBUG -----
21:11:56 [RMI TCP Connection(8)-192.168.0.108] DEBUG - |### Thread[RMI TCP Connection(8)-192.168.0.108,5,RMI Runtime] ###
21:11:56 [RMI TCP Connection(8)-192.168.0.108] DEBUG - |Lista trabajo: [alfombralumieb.blend:192.168.0.108]
21:11:56 [RMI TCP Connection(8)-192.168.0.108] DEBUG - |Acaba de terminar: alfombralumieb.blend
21:11:56 [RMI TCP Connection(8)-192.168.0.108] DEBUG - |Eliminando... alfombralumieb.blend
21:11:56 [RMI TCP Connection(8)-192.168.0.108] DEBUG -----
```

ACCIÓN: Se conectan dos clientes y dan un trabajo para hacer. Habrá dos worker, uno mas rápido que otro, se espera que cuando uno termine con un trabajo, comience a trabajar en el otro.

Pantalla servidor principal:

```
03:44:58 [RMI TCP Connection(2)-192.168.0.108] DEBUG - Registrando nuevo worker: 192.168.0.108
03:45:13 [RMI TCP Connection(4)-192.168.0.105] DEBUG - Registrando nuevo worker: 192.168.45.1
03:45:29 [RMI TCP Connection(6)-192.168.0.108] INFO - Se conecto el cliente192.168.0.108
03:45:30 [RMI TCP Connection(6)-192.168.0.108] DEBUG -----
03:45:30 [RMI TCP Connection(6)-192.168.0.108] DEBUG - |### Thread[RMI TCP Connection(6)-192.168.0.108,5,RMI Runtime] ###
03:45:30 [RMI TCP Connection(6)-192.168.0.108] DEBUG - |Pedido de trabajo: Tienda.blend
03:45:30 [RMI TCP Connection(6)-192.168.0.108] DEBUG -----
03:45:30 [Thread-1] INFO - Trabajando en: Tienda.blend Modalidad: Por samples. Cantidad: 2 Samples
03:45:30 [Thread-1] INFO - Tiempo inicio: 03:45:30.697
03:45:31 [RMI TCP Connection(8)-192.168.0.108] INFO - Se conecto el cliente192.168.0.108
03:45:31 [RMI TCP Connection(8)-192.168.0.108] DEBUG -----
03:45:31 [RMI TCP Connection(8)-192.168.0.108] DEBUG - |### Thread[RMI TCP Connection(8)-192.168.0.108,5,RMI Runtime] ###
03:45:31 [RMI TCP Connection(8)-192.168.0.108] DEBUG - |Pedido de trabajo: vasoCafe.blend
03:45:31 [RMI TCP Connection(8)-192.168.0.108] DEBUG -----
03:45:31 [Thread-2] INFO - Trabajando en: vasoCafe.blend Modalidad: Por samples. Cantidad: 2 Samples
03:45:31 [Thread-2] INFO - Tiempo inicio: 03:45:31.643
03:45:31 [RMI TCP Connection(4)-192.168.0.105] INFO - 192.168.45.1: Trabajos realizados: [] | Trabajos para realizar:[vasoCafe.blend:192.168.0.108]
03:45:31 [RMI TCP Connection(4)-192.168.0.105] INFO - 192.168.45.1: Delegando tarea [vasoCafe.blend:192.168.0.108]
03:45:31 [RMI TCP Connection(2)-192.168.0.108] INFO - 192.168.0.108: Trabajos realizados: [] | Trabajos para realizar:[vasoCafe.blend:192.168.0.108]
03:45:31 [RMI TCP Connection(2)-192.168.0.108] INFO - 192.168.0.108: Delegando tarea [vasoCafe.blend:192.168.0.108]
```

Pantalla servidor principal: Worker comienza a hacer la tarea nueva mientras que el otro aun no termino.

```
03:45:42 [Thread-2] DEBUG - Cantidad workers que finalizaron: 0 Cantidad de workers total: 2
03:45:42 [Thread-2] DEBUG - Cantidad de imagenes: 1 Cantidad esperada: 4
03:45:43 [Thread-2] DEBUG - Nodos sin terminar: [192.168.0.108, 192.168.45.1]
03:45:49 [Thread-2] INFO - Mensaje leido: vasoCafe.blend:192.168.0.108 | Buscando trabajo: vasoCafe.blend:192.168.0.108 | La cola tiene: 0 mensajes
03:45:49 [Thread-2] INFO - Encontre el mensaje:
    From:192.168.0.108
    Name:vasoCafe.blend:192.168.0.108
    Nro.Render:3
    La cola tiene ahora: 2 mensajes
03:45:49 [Thread-2] INFO - ---
03:45:49 [Thread-2] INFO - Worker: 192.168.0.108 Render count:2 @ vasoCafe.blend:192.168.0.108
03:45:49 [Thread-2] INFO - ---
03:45:49 [Thread-2] DEBUG - Cantidad workers que finalizaron: 1 Cantidad de workers total: 2
03:45:49 [Thread-2] DEBUG - Cantidad de imagenes: 2 Cantidad esperada: 4
03:45:49 [Thread-2] DEBUG - Nodos sin terminar: [192.168.45.1]
03:45:50 [RMI TCP Connection(2)-192.168.0.108] INFO - 192.168.0.108: Trabajos realizados: [vasoCafe.blend:192.168.0.108] | Trabajos para realizar:[vasoCafe.blend:192.168.0.108, Tienda.blend:192.168.0.108]
03:45:52 [RMI TCP Connection(2)-192.168.0.108] INFO - 192.168.0.108: Delegando tarea [Tienda.blend:192.168.0.108]
```

*El worker que termino primero le hace saber a el servidor que ya realizo un trabajo.

Pantalla servidor principal: Termina el worker mas lento el primer trabajo

```
03:46:05 [Thread-2] INFO - Encontre el mensaje:
    From:192.168.45.1
    Name:vasoCafe.blend:192.168.0.108
    Nro.Render:3
    La cola tiene ahora: 2 mensajes
03:46:05 [Thread-2] INFO - ---
03:46:05 [Thread-2] INFO - Worker: 192.168.45.1 Render count:2 @ vasoCafe.blend:192.168.0.108
03:46:05 [Thread-2] INFO - ---
03:46:05 [Thread-2] DEBUG - Cantidad workers que finalizaron: 2 Cantidad de workers total: 2
03:46:05 [Thread-2] DEBUG - Cantidad de imagenes: 4 Cantidad esperada: 4
03:46:05 [Thread-2] DEBUG - Nodos sin terminar: []
03:46:06 [Thread-2] INFO - Trabajando en: vasoCafe.blend##
03:46:06 [Thread-2] INFO - Cantidad de imagenes procesadas: 4
03:46:07 [RMI TCP Connection(2)-192.168.0.105] INFO - 192.168.45.1: Trabajos realizados: [vasoCafe.blend:192.168.0.108] | Trabajos para realizar:[vasoCafe.blend:192.168.0.108, Tienda.blend:192.168.0.108]
03:46:07 [RMI TCP Connection(2)-192.168.0.105] INFO - 192.168.45.1: Delegando tarea [Tienda.blend:192.168.0.108]
03:46:08 [RMI TCP Connection(8)-192.168.0.108] DEBUG -----
03:46:08 [RMI TCP Connection(8)-192.168.0.108] DEBUG - |### Thread[RMI TCP Connection(8)-192.168.0.108,5,RMI Runtime] ###
03:46:08 [RMI TCP Connection(8)-192.168.0.108] DEBUG - |Lista trabajo: [vasoCafe.blend:192.168.0.108, Tienda.blend:192.168.0.108]
03:46:08 [RMI TCP Connection(8)-192.168.0.108] DEBUG - |Acaba de terminar: vasoCafe.blend
03:46:08 [RMI TCP Connection(8)-192.168.0.108] DEBUG - |Eliminando... vasoCafe.blend
03:46:08 [RMI TCP Connection(8)-192.168.0.108] DEBUG -----
```

Pantalla servidor principal: Terminan de realizar los dos trabajos y quedan ociosos hasta que haya un nuevo trabajo.

```
03:47:20 [Thread-1] DEBUG - Nodos sin terminar: [192.168.45.1]
03:47:53 [Thread-1] INFO - Mensaje leido: Tienda.blend:192.168.0.108 | Buscando trabajo: Tienda.blend:192.168.0.108 | La cola tiene: 0 mensajes
03:47:53 [Thread-1] INFO - Encontre el mensaje:
    From:192.168.45.1
    Name:Tienda.blend:192.168.0.108
    Nro.Render:3
    La cola tiene ahora: 2 mensajes
03:47:53 [Thread-1] INFO - ---
03:47:53 [Thread-1] INFO - Worker: 192.168.45.1 Render count:2 @ Tienda.blend:192.168.0.108
03:47:53 [Thread-1] INFO - ---
03:47:53 [Thread-1] DEBUG - Cantidad workers que finalizaron: 2      Cantidad de workers total: 2
03:47:53 [Thread-1] DEBUG - Cantidad de imagenes: 4      Cantidad esperada: 4
03:47:54 [Thread-1] DEBUG - Nodos sin terminar: []
03:47:54 [Thread-1] INFO - ###Termine: Tienda.blend###
03:47:54 [Thread-1] INFO - Cantidad de imagenes procesadas:      4
03:47:56 [Thread-1] INFO - Tiempo tardado:      2 minutos.
03:47:56 [RMI TCP Connection(6)-192.168.0.108] DEBUG - -----
03:47:56 [RMI TCP Connection(6)-192.168.0.108] DEBUG - |### Thread[RMI TCP Connection(6)-192.168.0.108,5,RMI Runtime] ###
03:47:56 [RMI TCP Connection(6)-192.168.0.108] DEBUG - |Lista trabajo: [Tienda.blend:192.168.0.108]
03:47:56 [RMI TCP Connection(6)-192.168.0.108] DEBUG - |Acaba de terminar: Tienda.blend
03:47:56 [RMI TCP Connection(6)-192.168.0.108] DEBUG - |Eliminando... Tienda.blend
03:47:56 [RMI TCP Connection(6)-192.168.0.108] DEBUG - -----
```

Ejecución

Cliente: Desde una linea de comandos dirigirse a la carpeta donde se encuentra “cliente.jar” y clienteConfig.json. Luego escribir: java -jar cliente.jar

Worker: Desde una linea de comandos dirigirse a la carpeta donde se encuentra “worker.jar” y la carpeta WorkerData. Luego escribir: java -jar worker.jar

Servidor: Desde una linea de comandos dirigirse a la carpeta donde se encuentra “servidor.jar” y la carpeta ServerData. Luego escribir: java -jar servidor.jar

Servidor Backup: Desde una linea de comandos dirigirse a la carpeta donde se encuentra “servidorbackup.jar” y la carpeta ServerData. Luego escribir: java -jar servidorbackup.jar

Algunas consideraciones

En algunos casos de equipos que tengan poca memoria y/o estén utilizando mucha memoria en otros procesos, se le dará poca memoria a la maquina virtual de Java, ocasionando un error de Java OutOfBoundMemory, lo que significa que el proceso necesitaba mas memoria de la que Java podía darle. Esto sucede, por lo general, cuando se requiere levantar archivos .blend ya sea desde el File System para el caso del cliente, o desde la cola de RabbitMQ para el worker. Una posible solución que funcionará en algunos casos es ejecutar el JAR de la siguiente manera:

java -Xmx {cantidad_maxima_que_se_pueda_asignar}*m -jar {worker.jar|cliente.jar|server.jar}

*Se recomienda 4096m en adelante, dependiendo del tamaño de archivos a manipular.

Pruebas de rendimiento

Prueba #1

Workers: PC1, PC2, PC3, PC4

Tiempo total: 17 Minutos

Cantidad de samples: 47

Tiempo en aplicar Image Stacking: 37 Segundos

Tiempo medio en procesar PC1/PC2/PC3/PC4: 34s/57s/71s/51s

Modalidad: Por tiempo (1000 Segundos)

Resultado:



Renderizado de 10 minutos en PC2



Prueba #2

Workers: PC1, PC2, PC3, PC4

Tiempo total: 7 Minutos

Cantidad de samples: 153

Tiempo en aplicar Image Stacking: 46 Segundos

Tiempo medio en procesar PC1/PC2/PC3/PC4: 4s/7s/10s/5s

Modalidad: Por tiempo (360 Segundos)

Resultado:



Renderizado de 1 minuto en PC1:



Prueba #3

Workers: PC1, PC4

Tiempo: 13 Minutos

Cantidad de samples: 30

Tiempo en aplicar Image Stacking: 29 Segundos

Modalidad: Por Samples (15 Samples)

Resultado:



Renderizado de 6 minutos en PC4:



Prueba #4

Workers: PC1, PC2, PC3, PC4

Tiempo total: 8 Minutos

Cantidad de samples: 53

Tiempo en aplicar Image Stacking: 25 Segundos

Tiempo medio en procesar PC1/PC2/PC3/PC4: 21s/52s/41s/48s

Modalidad: Por tiempo (500 Segundos)

Resultado:



Renderizado de 8 minutos en PC4:



Prueba #5

Workers: PC1, PC2, PC3, PC4

Tiempo total: 9 Minutos

Cantidad de samples: 38

Tiempo en aplicar Image Stacking: 13 Segundos

Tiempo medio en procesar PC1/PC2/PC3/PC4: 18s/46s/39s/45s

Modalidad: Por tiempo (500 Segundos)

Resultado:



Renderizado de 13 minutos en PC4:



Prueba #6

Workers: PC1, PC2, PC3, PC4

Tiempo total: 10 Minutos

Cantidad de samples: 5

Tiempo en aplicar Image Stacking: 4 Segundos

Tiempo medio en procesar PC1/PC2/PC3/PC4: 149s/341s/397s/334s

Modalidad: Por tiempo (600 Segundos)

Resultado:



Renderizado de 7 minutos en PC2:



Prueba #7

Workers: PC1, PC2, PC3, PC4

Tiempo total: 10 Minutos

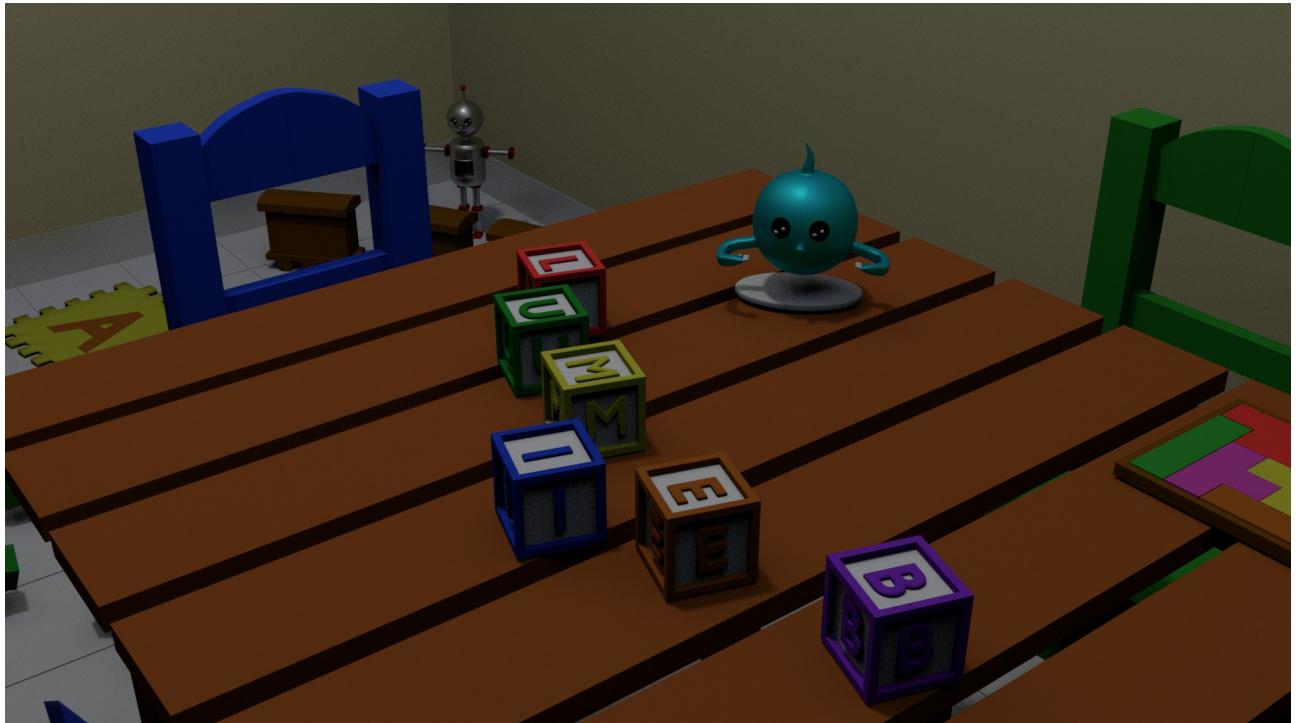
Cantidad de samples: 44

Tiempo en aplicar Image Stacking:

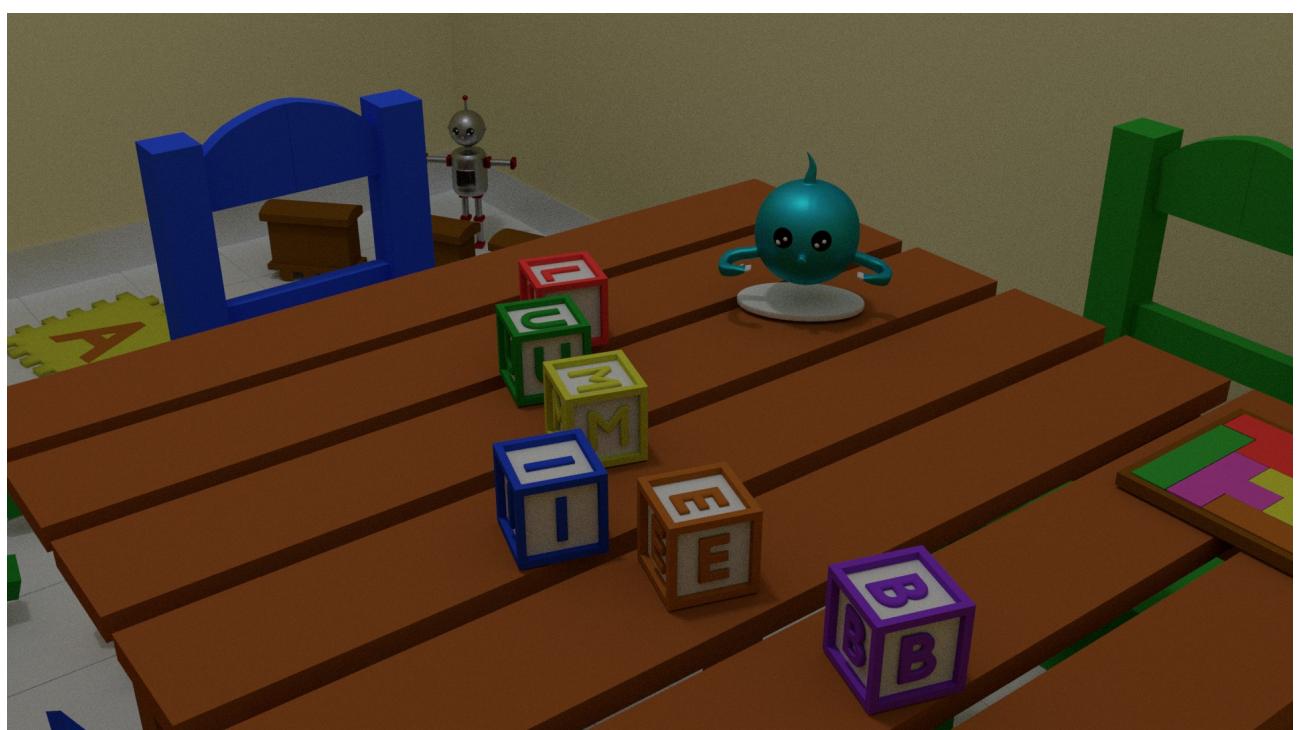
Tiempo medio en procesar PC1/PC2/PC3/PC4: 27s/55s/65s/58s

Modalidad: Por tiempo (600 Segundos)

Resultado:



Renderizado de 9 minutos en PC4:



Prueba #8

Workers: PC1, PC2, PC4

Tiempo total: 10 Minutos

Cantidad de samples: 8

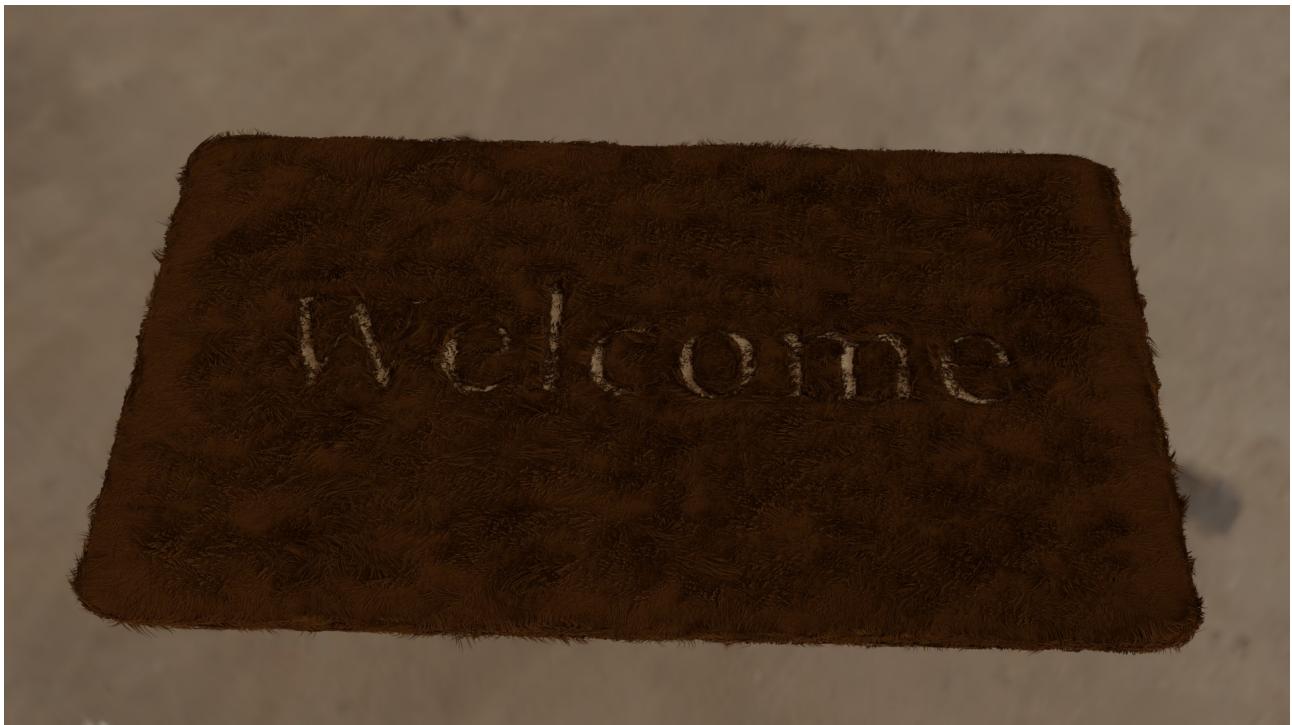
Tiempo en aplicar Image Stacking: 3 Segundos

Tiempo medio en procesar PC1/PC2/PC3/PC4: 86s/342s/-/315s

Modalidad: Por tiempo (600 Segundos)

Opcion: HighRender (En la configuración de render se establecieron parámetros de mas calidad)

Resultado:



Renderizado de 11 minutos en PC2:



Resultado de pruebas

La columna resultado corresponde a la observación sobre la calidad de imagen final obtenida en relación al tiempo que se tardo entre la imagen lograda con una arquitectura distribuida y la imagen lograda con una arquitectura centralizada, tomando como punto de partida el resultado de la arquitectura distribuida, es decir, si el resultado es “malo” es porque la imagen de la arquitectura centralizada fue mejor.

| #Prueba | Workers involucrados | Tiempo total (Arq. Distr.) | Cant. samples | Modalidad | Tiempo total (Arq. Central) | Resultado |
|---------|-----------------------------|----------------------------|---------------|-----------------------------|-----------------------------|--|
| 1 | PC1, PC2, PC3, PC4 | 17 Minutos | 47 | Por tiempo 1000 segundos | 10 Minutos | Bueno. El nivel de ruido es mínimo |
| 2 | PC1, PC2, PC3, PC4 | 7 Minutos | 153 | Por tiempo 360 segundos | 1 Minuto | Regular. Al ser una escena simple no se genera tanto ruido, de todas formas la imagen obtenida en la arquitectura distribuida es de mejor calidad. |
| 3 | PC1,PC4 | 13 Minutos | 30 | Por samples 15 samples | 6 Minutos | Malo. En esta modalidad no se aprovecha al 100% el uso de los nodos, sumado a que solo trabajaron 2 nodos. La arq. Centralizada logro un resultado mínimamente inferior en menos tiempo. Esta escena se volvió a realizar en la prueba #5 con los 4 nodos. |
| 4 | PC1, PC2, PC3, PC4 | 8 Minutos | 53 | Por tiempo 500 segundos | 8 Minutos | Bueno. Si bien no hubo niveles altos de ruido en ninguno de las dos arquitecturas, la calidad de imagen de la arq. distribuida es superior a la central. |
| 5 | PC1, PC2, PC3, PC4 | 9 Minutos | 38 | Por tiempo 500 segundos | 13 Minutos | Bueno. En este caso tenemos dos calidades casi en el mismo nivel, pero la diferencia está en el tiempo tardado. La arquitectura distribuida tardo menos que la central y obtuvo un resultado similar o un poco mejor. |
| 6 | PC1, PC2, PC3, PC4 | 10 Minutos | 5 | Por tiempo 500 segundos | 7 Minutos | Muy bueno. El nivel de ruido de la arquitectura centralizada es altísimo en comparación de la arq. distribuida. |
| 7 | PC1, PC2, PC3, PC4 | 10 Minutos | 44 | Por tiempo 600 segundos | 9 Minutos | Bueno. Si bien ninguno de los dos resultados se ven excelentes, el renderizado distribuido parece tener mas fidelidad que el centralizado. |
| 8 | PC1, PC2, PC4 | 10 Minutos | 8 | Por tiempo 600 segundos | 11 Minutos | Bueno desde el punto de vista del rendimiento, ambos resultados no poseen ruido y se ven muy bien. Esta prueba fue la primera con la opción de alta calidad, mostrando resultados muy óptimos. |

Adicionalmente, en todas las imágenes obtenidas por la arquitectura distribuida se nota una mejora en el contraste y el color, esto es debido a el proceso de Image Stacking

Conclusión

En algunos aspectos el uso de esta herramienta puede significar una ventaja grande en comparación del tiempo y la calidad. Ademas, elimina completamente el ruido agudo* generado cuando se realizan tomas individuales. En algunos casos, se puede ver que la mejora no es significativa desde ningún punto de vista. El aprovechamiento de esta herramienta se verá completamente ligado a el tipo de proyecto que se le pida renderizar.

Si bien la herramienta desarrollada esta algo lejos en performance de competir con herramientas actualmente en el mercado de la misma índole (granjas de renderizado), se pudo aprender mucho en su fase desarrollo e implementación, desde la ejecución de sentencias por linea de comando en Java, hasta la implementación por RMI del sistema distribuido, aprendiendo sus virtudes o defectos, y entendiendo cuando es optimo implementar este tipo de comunicación. Secundariamente también se aprendió a manejar scripts en Python (necesarios para el Blender) y configuraciones propias del programa que no hacen a la programación.

La principal característica que se buscó en el desarrollo es la de un sistema distribuido con recuperación frente a fallas en sus nodos y recuperación frente a fallas en el servidor principal, lo cual fue concretado con éxito, resultado de horas de investigación y ayudas del equipo docente. Dejando en segundo plano (por cuestiones de tiempo) la calidad del producto final e interfaces de usuario.

Por otra parte, si nos abstraemos un poco dentro del código, este sistema se podría (con algunas modificaciones) transformar en una herramienta para manejar trabajos distribuidos en cualquier tipo de aplicación por linea de comando.

Para finalizar, las conclusiones son muy positivas, y si bien, lo que se buscaba era una herramienta competitiva a nivel calidad/performance, se logró una herramienta muy personalizable en cuestiones de configuración del renderizado, con base en aplicar un filtro que disminuye el nivel de ruido a nivel del cuadrado inverso y de trabajo distribuido.

*Ruido agudo:

