Sistemas Distribuidos y Programación Paralela (41409)

Trabajo Práctico Final – Sistema Distribuido de Renderizado de Imágenes

Universidad Nacional de Luján – 2022

Integrantes:

Ledesma Damián (155825)

Patricio Pittavino (121476)

Tabla de Contenidos

1. Objetivo

El presente proyecto es un Trabajo Práctico Final Integrador de Conocimientos para la materia Sistemas Distribuidos y Programación Paralela de la Universidad Nacional de Luján.

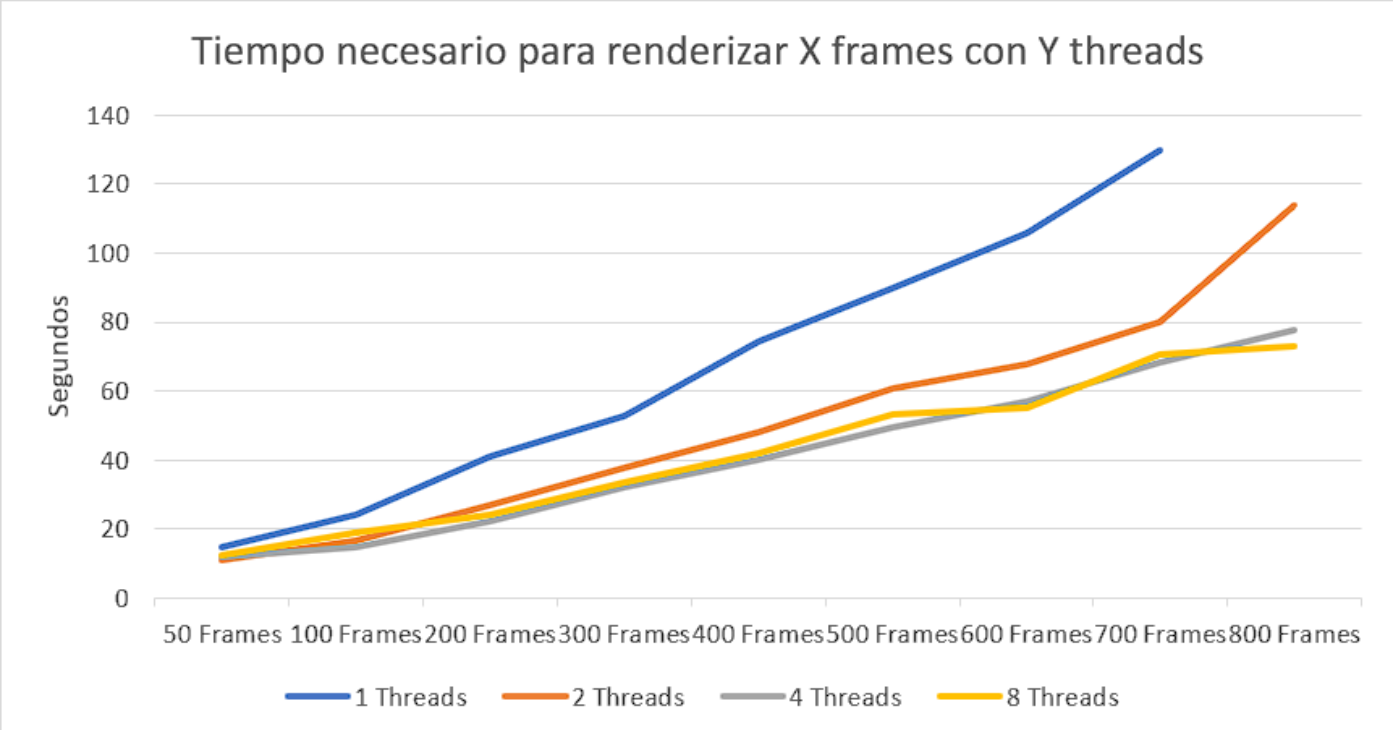
2. Propuesta

La propuesta consiste en crear una red de renderizado de imágenes a partir de un archivo .blend creado en el programa Blender, otorgando así una forma de renderizar modelos y animaciones aún en aquellos computadores sin la potencia requerida para éste trabajo. Ésta red será tolerante a fallos, escalable, multithread y distribuida.

Blender es una suite open source de creación de modelos y animaciones 3D.

Centrándonos en las animaciones y luego de varias pruebas con 1, 2, 4 y 8 threads y 50, 100, 200, 300, 400, 500, 700, y 800 frames, obtenemos el siguiente gráfico, en el cual se aprecia que con un único thread y a mayor cantidad de frames, el tiempo necesario para completar el renderizado es mucho mayor que hacerlo con mayor cantidad de threads.

	50 Frames	100 Frames	200 Frames	300 Frames	400 Frames	500 Frames	600 Frames	700 Frames	800 Frames
1 Threads	14,54	24,17	40,95	52,55	74,4	90	105,65	130,04	
2 Threads	11,04	16,67	26,87	37,57	48,09	60,6	67,97	80,05	113,69
4 Threads	11,75	14,54	22,22	32,05	40,03	49,4	57,04	68,11	77,61
8 Threads	12,2	18,83	24,1	33,48	41,98	53,01	54,96	70,48	72,81



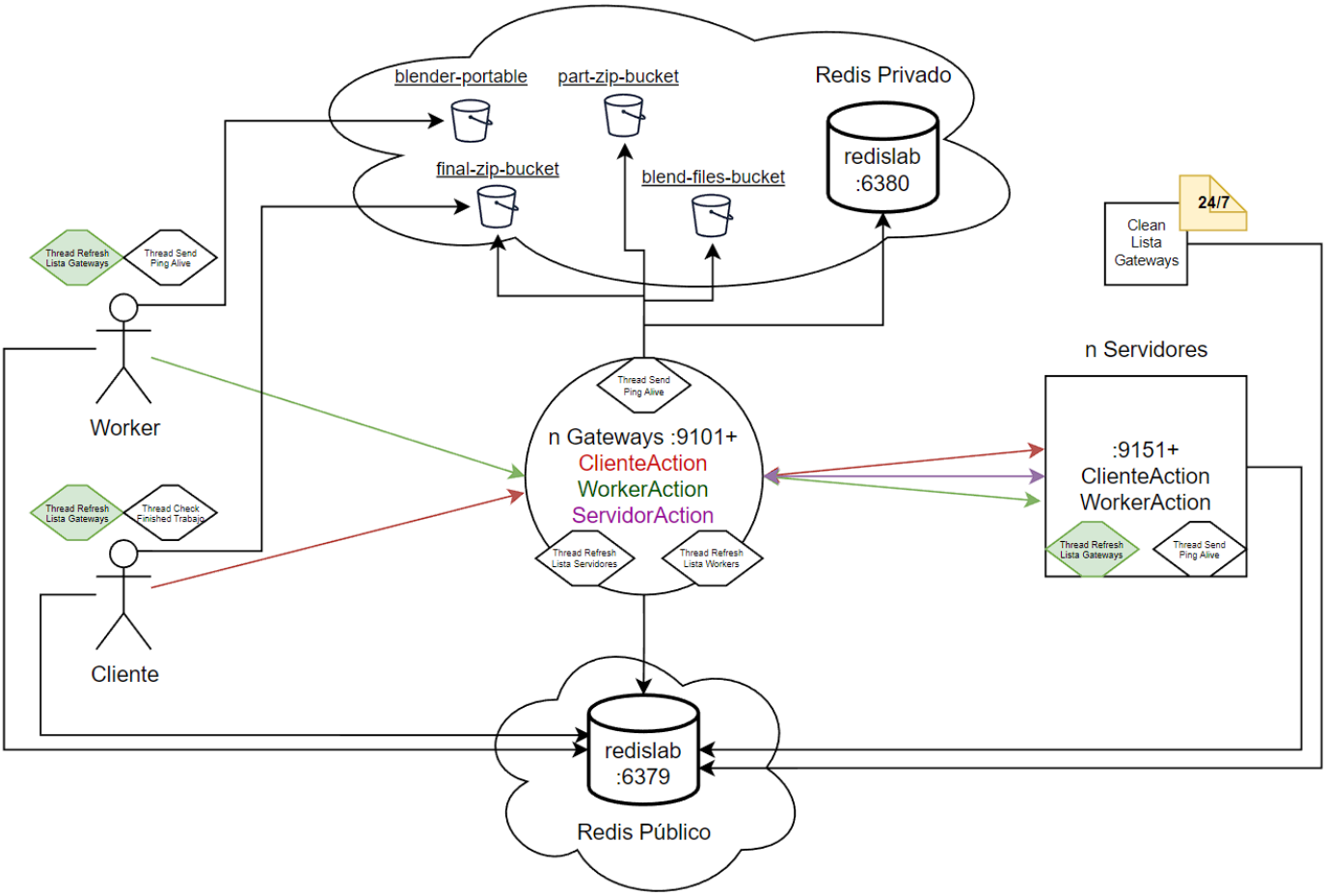
También se observa que el rendimiento con 4 y 8 threads es muy parecido por lo que se decide utilizar como máximo 4 threads.

Basándonos en esto, se propone crear una red distribuida para distribuir la carga de trabajo a lo largo de cada Worker, dividiendo cada trabajo en partes de 300 frames, que a su vez se dividirán en Threads.

De acuerdo a las pruebas, haremos la siguiente división de frames por thread en un Worker:

Frames	Threads
1-49	1
50-99	2
100-300	4

3. Arquitectura



4. Objetivo de cada elemento/nodo

- Redis Público (RPU): base de datos en memoria de acceso público con una única tabla hash llamada ListaGateways(LG) encargada de almacenar en formato JSON los datos de conexión a un Gateway.
- Redis Privado (RPR): base de datos en memoria de acceso restringido, con las tablas hash ListaServidores(LS), ListaWorkers(LW), ListaTrabajos(LT) y ListaPartes(LP) encargadas de almacenar en formato JSON datos referentes a cada elemento.
- Bucket blender-portable: Bucket en GCS de acceso público con dos archivos, .zip y .tar.xz, ambos conteniendo la aplicación Blender, pero el primero sirve para SO Windows y el otro para SO Linux.
- Bucket blend-files-bucket: Bucket en GCS de acceso privado, que contiene los proyectos en formato .blend enviados por los Clientes, y que serán enviados a los Workers cuando lo soliciten. Estos archivos se eliminan una vez que se termina el trabajo de renderizado.
- Bucket part-zip-bucket: Bucket en GCS de acceso privado, que contiene en formato .zip las partes de un trabajo que han sido terminadas. Estos archivos se eliminan una vez que se terminan todas las partes correspondientes a un trabajo.
- Bucket final-zip-bucket: Bucket en GCS de acceso público, que contiene en formato .zip las imágenes renderizadas del trabajo solicitado. Estos archivos son el resultado final de unir todas las partes correspondientes a un trabajo. No se eliminan.
- Clean-Lista-Gateway: Nodo con disponibilidad 100% cuya única función es actualizar la LG de RPU al verificar que el último ping registrado de cada Gateway no exceda los 10s.
- Gateway: Nodo que actúa de interfaz de conexión entre los demás nodos, haciendo transparente la implementación de funciones entre Worker-Servidor, Cliente-Servidor, Servidor-GCS y Servidor-RPR.
    - Es capaz de iniciar y publicar servicios RMI.
    - Es capaz de agregar su información de red en RPU.
    - Es capaz de actualizar su último ping en RPU.
    - Es capaz de recibir peticiones de Workers y Clientes, y redirigirlos aleatoriamente a un Servidor.
    - Es capaz de recibir peticiones de Servidores y conectarse con GCS o RPR.
    - Es capaz de actualizar la LS y la LW de RPU al verificar que el último ping registrado de cada Servidor o Worker no exceda los 10s.
- Servidor: Nodo encargado de procesar las solicitudes de los Workers y Clientes. Su única comunicación es con el Gateway.
    - Es capaz de iniciar y publicar servicios RMI.
    - Es capaz de procesar las solicitudes de un Worker, los cuales pueden ser:
        - Ping.
        - Dar un trabajo.
        - Marcar parte de un trabajo como realizado.
        - Dar un archivo .blend.
    - Es capaz de procesar las solicitudes de un Cliente, los cuales pueden ser:
        - Renderizar .blend.
        - Dar información sobre el trabajo.

    - Es capaz de dividir en partes el trabajo a realizar si el rango de frames es mayor que el límite preestablecido.
    - Es capaz de enviar un Ping para que actualice la LS.
    - Es capaz de enviar una solicitud de CRUD de la LT, LP y en GCS.
    - Es capaz de leer la LG de RPU.
- Worker: Nodo encargado de hacer uso de su GPU para renderizar los trabajos recibidos. Su única comunicación es con el Gateway.
    - Es capaz de enviar un Ping para que actualice la LW.
    - Es capaz de descargar la aplicación Blender en caso de ser necesario.
    - Es capaz de solicitar cada cierto período de tiempo un nuevo trabajo.
    - Es capaz de renderizar un trabajo por línea de comando usando la aplicación Blender, y enviar los resultados.
    - Es capaz de elegir entre 1, 2 y 4 Threads para renderizar el trabajo de acuerdo al rango de frames.
    - Es capaz de leer la LG de RPU.
- Cliente: Nodo cliente quien a través de una GUI envía un proyecto .blend para ser renderizado. Su única comunicación es con el Gateway.
    - Es capaz de seleccionar el archivo .blend deseado, marcando el rango de frames a renderizar.
    - Es capaz de recibir una respuesta con el link de descarga de un archivo .zip con las imágenes renderizadas.

5. Funcionamiento de cada nodo

- Gateway:
    a) Al encender se inician y publican los servicios de RMI: workerAction, clienteAction, servidorAction.
    b) Se conecta a RPU y agrega su ip pública junto con los puertos RMI a la LG.
    c) Se obtiene la LS activos.
    d) Se crea un thread para actualizar cada 5s la LG con el lastPing actualizado. (Thread Send Ping Alive)
    e) Se crea un thread para actualizar cada 10s la LS. (Thread Refresh Lista Servidores)
        i) Se verifica que now.timestamp - servidor.lastPing < 10s.
        ii) Se elimina el Servidor de la LS.
    f) Se crea un thread para actualizar cada 10s la LW. (Thread Refresh Lista Workers)
        i) Se verifica que now.timestamp - servidor.lastPing < 10s.
        ii) Si debe eliminarse el Worker de la LW, primero debe verificarse si tenía un trabajo actualmente para modificar su estado a TODO y liberarlo.
    g) Al recibir un hello/ping de un Servidor, se agrega/actualiza a la LS en RPR con el timestamp actual.
    h) Por cada petición de CRUD a RPR o GCE, se encarga de establecer la conexión con Redis o los Buckets y subir y/o descargar la información necesaria.

i) Por cada petición de un Cliente o Worker, redirige la petición intercalando entre la LS.
    i) Si un Servidor no contesta, continúa con el siguiente.
    ii) Si ninguno contesta, continúa loopeando hasta que reciba confirmación.

- Cliente:
    a) Al encender, se muestra la GUI, el usuario selecciona un archivo .blend, define el rango de frames a renderizar y hace click en 'Renderizar'.
    b) En éste momento se conecta a RPU, obtiene la LG y selecciona al azar uno y le envía el trabajo.
        i) Si un Gateway no contesta, continúa con el siguiente.
        ii) Si ninguno contesta, continúa loopeando hasta que reciba confirmación.
    c) Se crea un thread por cada trabajo para consultar cada 1s si está terminado y recuperar el link de descarga. (Thread Check Finished Trabajo)

- Servidor:
    a) Al encender, se inician y publican los servicios de RMI: workerAction, clienteAction.
    b) Se envía un hello con la ip pública y puertos RMI a un gateway para que éste lo agregue a la LS.
    c) Se conecta a RPU, obtiene la LG y por cada petición que haga, va intercalando conexión entre gateways.
        i) Si un Gateway no contesta, continúa con el siguiente.
        ii) Si ninguno contesta, continúa loopeando hasta que reciba confirmación.
    d) Se crea un thread para actualizar cada 10s la LG. (Thread Refresh Lista Gateways)
    e) Se crea un thread para enviar cada 5s un ping a un gateway. (Thread Send Ping Alive)
    f) Al recibir un hello/ping de un Worker, el Servidor solicita a un Gateway agregar/actualizar la LW en RPR con el timestamp actual.
    g) Al recibir un nuevo trabajo de un Cliente, el Servidor dividir en partes el trabajo a realizar si el rango de frames es mayor que el límite preestablecido, y solicita a un Gateway agregar/actualizar la LT y la LP en RPR.
    h) Al recibir una parte terminada por parte de un Worker, solicita al Gateway actualizar la LW liberando al Worker relacionado, y actualiza la LP, también verifica si el trabajo está terminado o aún hay partes faltantes.
        i) Si todas las partes relacionadas a un trabajo están terminadas, procede a solicitar al Gateway todos los .zip partes, para mergear todas ellas en un solo .zip, y luego subirlo al bucket final-zip-bucket.
        ii) Una vez enviado al bucket, se eliminan las partes del bucket part-zip-bucket.
        iii) Se actualiza la información del trabajo en LT.

- Worker:
    a) Al encender, se conecta a RPU, obtiene la LG y por cada petición que haga, va intercalando conexión entre gateways.
        i) Si un Gateway no contesta, continúa con el siguiente.
        ii) Si ninguno contesta, continúa loopeando hasta que reciba confirmación.
    b) Se verifica que existe la aplicación Blender, y sino se procede a descargarla del bucket público blender-portable.
    c) Se crea un thread para actualizar cada 10s la LG. (Thread Refresh Lista Gateways)
    d) Se crea un thread para enviar cada 5s un ping. (Thread Send Ping Alive)
    e) En el thread principal, cada 1s consulta si hay un nuevo trabajo para hacer.
        i) Al recibir un nuevo trabajo, se procede a descargar el archivo .blend correspondiente.
        ii) Se crean tantos threads como sean necesarios de acuerdo al rango de frames del trabajo. (1, 2 o 4 threads)
        iii) Al finalizar el renderizado, se crea un .zip con las imágenes, y se envían, se eliminan los archivos .blend y .zip y finaliza el trabajo actual.
- Clean-Lista-Gateway:
    a) Actualiza cada 10s la lista de gateways.
        i) Se verifica que now.timestamp - gateway.lastPing < 10s.
        ii) Se elimina el gateway de la lista de gateways.

6. Aspectos Técnicos

6.1. Manejo de nodos

La comunicación entre los nodos es a través de servicios publicados por RMI, el cual por definición tiene soporte para múltiples clientes creando un Thread por cada petición.

Los Servidores, Workers y Clientes solo se comunican con el Gateway utilizando los servicios de RMI publicados, servidorAction, workerAction y clienteAction. Mientras que el Gateway solo se comunica con los Servidores a través del servicio RMI publicado workerAction y clienteAction, los cuales utiliza para redirigir las peticiones que le llegan de los Clientes y Workers.

6.2. Manejo de múltiples clientes en simultáneos

Todos los nodos se comunican entre sí mediante RMI, el cual es inherentemente multithreading. Esto significa que cada solicitud remota de un cliente es un hilo separado y puede ejecutarse simultáneamente con solicitudes de otros clientes.

Los ‘Servidores RMI’ son el Nodo Gateway que publica los servicios de workerAction, clienteAction y servidorAction, y el Nodo Servidor que publica los servicios workerAction y clienteAction. Cabe destacar que el único cliente que se conecta al Nodo Servidor es el Nodo Gateway, mientras que los clientes del Nodo Gateway son el Worker, el Cliente y el Servidor.

6.3. Manejo de Sincronismo

Tanto el Cliente, el Worker como el Servidor, mantienen en memoria durante 10s la LG, el cual mediante la instrucción *'synchronized'* logran bloquear el recurso en cuanto lo necesiten. Al cabo de los 10s un thread actualiza esa lista en memoria de acuerdo a la lista contenida en RPU.

Lo mismo ocurre en el Gateway pero con la LS, mantiene en memoria durante 10s la LS, la cual luego se va a actualizar de acuerdo a la lista contenida en RPR y después de eliminar aquellos servidores por timeout.

En cuanto al acceso a recursos compartidos como Redis, éste es por definición singlethread, por lo que no hay posibilidad que se bloquee ningún recurso.

6.4. Distribución y procesamiento de tareas

En el momento que un Servidor recibe un Trabajo(tarea), éste lo divide en partes de X frames, y se agregan a la LP con el estado TODO. Mientras tanto los Worker están pidiendo constantemente trabajos para hacer, en cuanto reciben una Parte (IN_PROGRESS), la procesan y devuelven el .zip con las imágenes renderizadas y la parte en estado DONE. Cuando el Servidor recibe una una parte DONE, verifica si todas las demas Partes correspondientes a un Trabajo están también en DONE, y de ser así, recupera los .zip de todas esas Partes, las fusiona en un solo .zip y las sube al final-zip-bucket.

6.5. Manejo de transparencia

El cliente cuenta con una interfaz gráfica en la que selecciona su proyecto .blend y define el rango de frames a procesar. No cuenta con ninguna información de cuantos Workers han trabajado en su proyecto ni de cuantos Servidores han manipulado su trabajo o imágenes, así como tampoco de los Buckets ni de Redis, solo conoce a su GUI que le notifica si se terminó de procesar el trabajo correctamente o si falló.

6.6. Contemplar aspectos de eficiencia

De acuerdo al informe inicial donde se observa que cuando la relación threads-frames es mayor, disminuye el tiempo tardado en renderizar, se decidió dividir en partes de 300 frames cada trabajo que recibirá un Worker, para que luego éste divida en Threads la Parte que recibe.

De ésta forma se logran dos objetivos, distribuir un trabajo con un rango de frames muy amplio para que varios Workers participen en el renderizado y obtener el mejor rendimiento colectivo, y por otro lado, una vez que un Worker obtiene una Parte, dividirla otra vez en Threads para obtener el mejor rendimiento individual.

6.7. Contemplar alta disponibilidad

En cuanto a los Buckets en Google Cloud Storage y servicios de Redis en RedisLab, éstos nos proporcionan una 'Alta Disponibilidad' según sus políticas, en GCS se informa que es del 99% mientras que RedisLab no brinda esa información.

Mientras que la disponibilidad de los Nodos Gateway, Servidor y Worker, está dada por la cantidad que haya de cada uno.

6.8. Contemplar escalabilidad del sistema

Al iniciar un nuevo Gateway, éste se agrega a la LG en la RPU y cada 5s actualiza con un ping su último estado. Mientras tanto el proceso CleanListaGateway de disponibilidad 100%, está observando ésta lista para detectar la caída de un Gateway y eliminarlo de la LG.

Al iniciar un nuevo Servidor, éste saluda aleatoriamente a algún Gateway y es agregado a la LS, y cada 5s envía un ping para avisar que está activo. Mientras tanto los demás Gateway actualizan cada 10s su LS para guardarla en memoria, por lo que en el peor de los casos un Gateway tardaría 10s en detectar un nuevo Servidor. Lo mismo ocurre para eliminar un Servidor, tardarían a lo sumo 10s para detectar la caída de un Servidor, eliminarlo y actualizar la LS tanto en memoria como en RPR.

Al iniciar un nuevo Worker, éste envía un ping cada 5s para avisar que está activo, el Servidor lo detecta y solicita al Gateway guardarlo en LW. Para eliminar un Worker caído ocurre igual que con un Servidor, con el agregado que si en ese momento el Worker estaba trabajando en una Parte, se lo recupera de la LP y se actualiza su estado a TODO para liberarlo.

6.9. Registro de actividades del sistema

Se hace uso de la dependencia Logback, el cual registra todos los acontecimientos de tipo Información y Error de cada Nodo del sistema en archivos de log por separado, indicando la Fecha, Hora, Thread, Nivel de Log y Mensaje.

6.10. Manejo de aspectos de seguridad

Para la comunicación con GCS y los buckets se utiliza una Clave de Acceso de formato JSON proporcionado por la Cuenta de Acceso en la configuración de GCS.

En cuanto a la conexión con RedisLab, se crearon roles de acceso público para la LG y acceso privado para la LS, LP, PT, así como también un rol de escritura para la LG. Se asignaron usuarios y contraseñas para dichos roles, y junto con el host y el puerto, se almacenan las credenciales en un archivo .env.

Tanto la clave JSON para GCS como las credenciales en el archivo .env, están ignoradas en git por lo que deben ser solicitadas al equipo de estudiantes o crear unas propias.

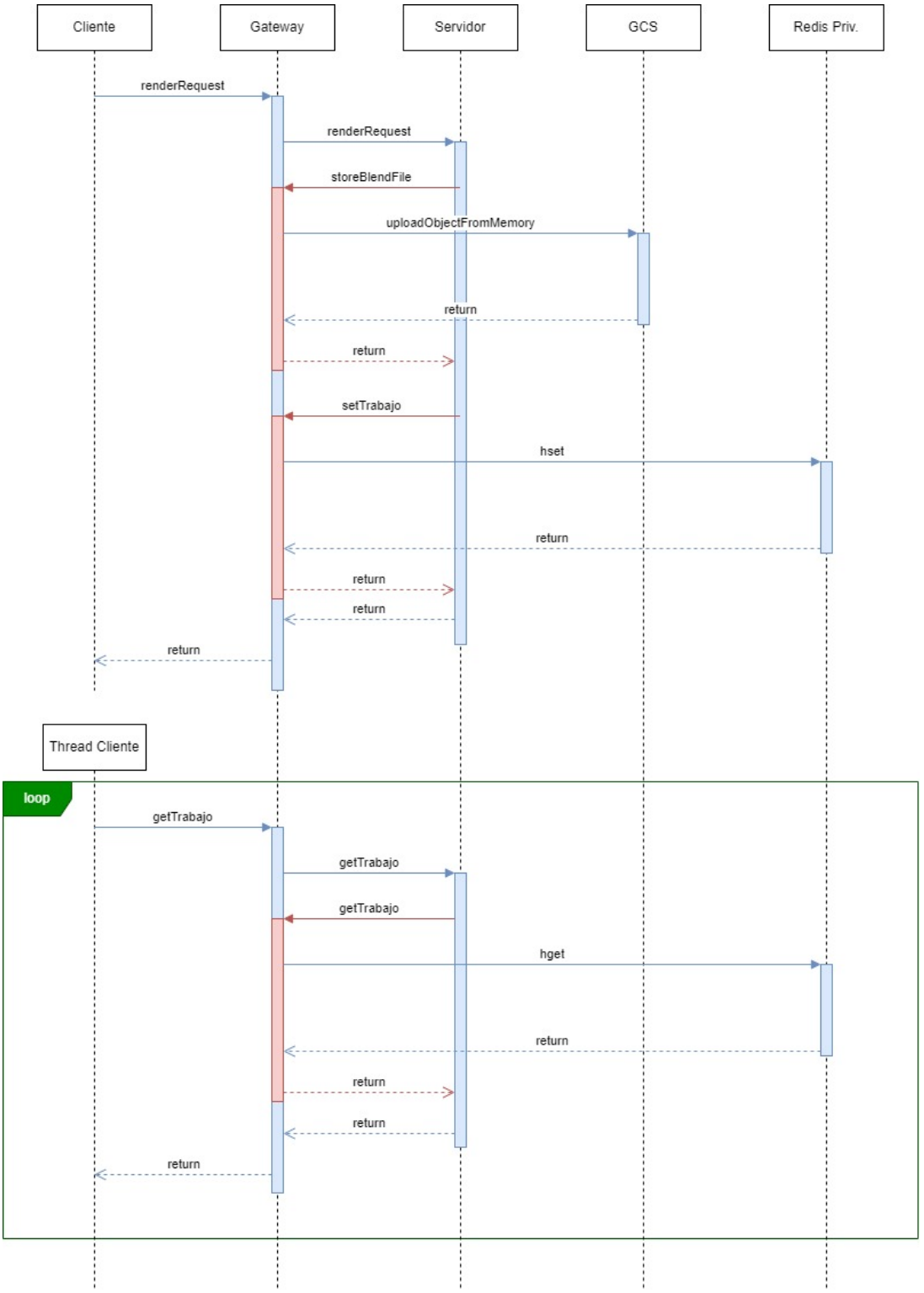
En cuanto a la seguridad en conexiones RMI, se investigó acerca de JAVA RMI con SSL (https://docs.oracle.com/javase/8/docs/technotes/guides/rmi/socketfactory/SSLInfo.html) pero no se implementó.

6.11. Métricas de Rendimiento

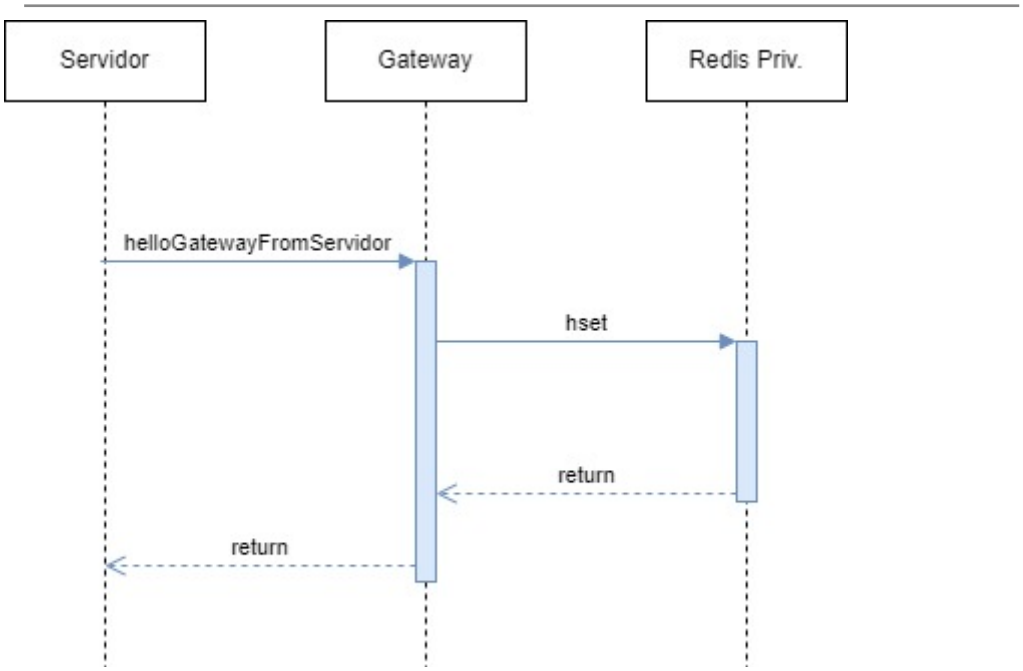
En las siguientes secciones se hará una comparación entre el rendimiento de un renderizado Centralizado, uno Centralizado Multithread y uno Distribuído Multithread.

7. Diagramas de Secuencia

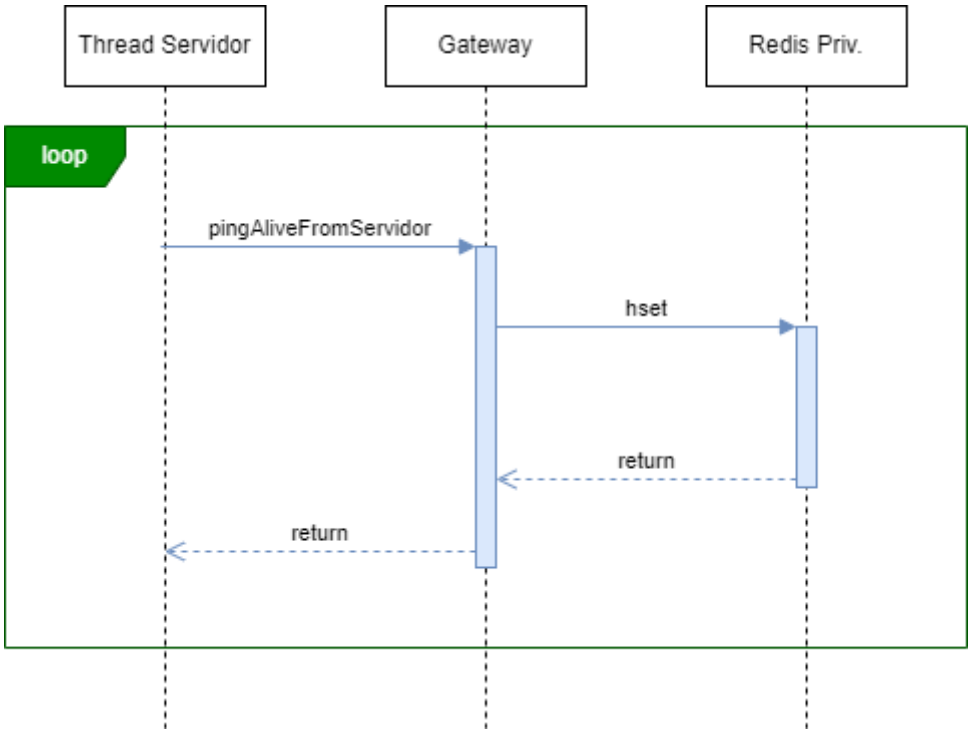
7.1. Cliente: renderRequest



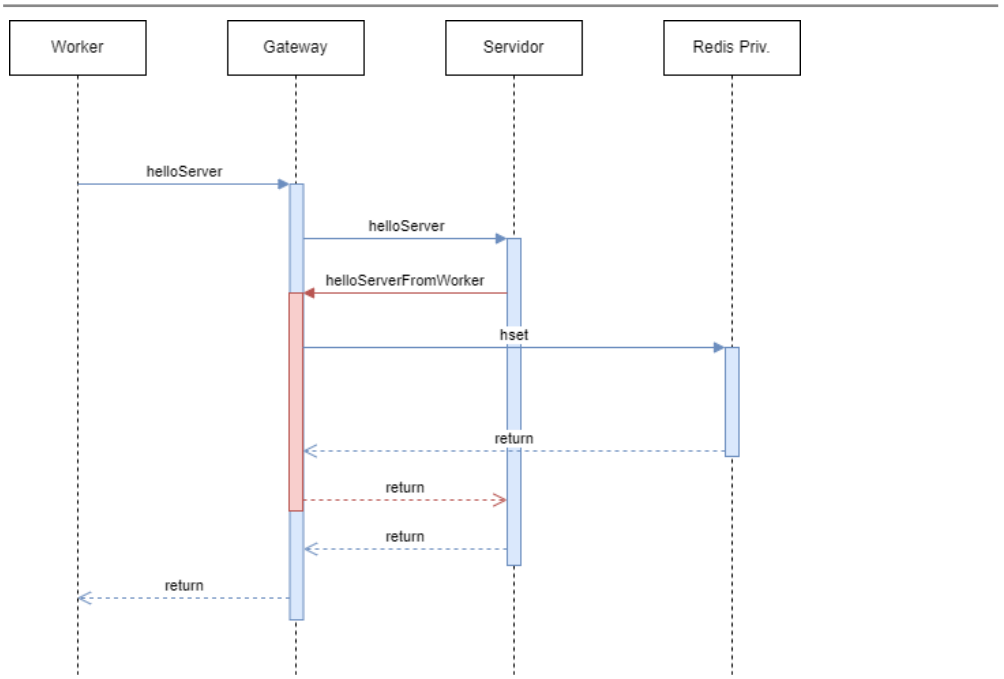
7.2. Servidor: helloGateway



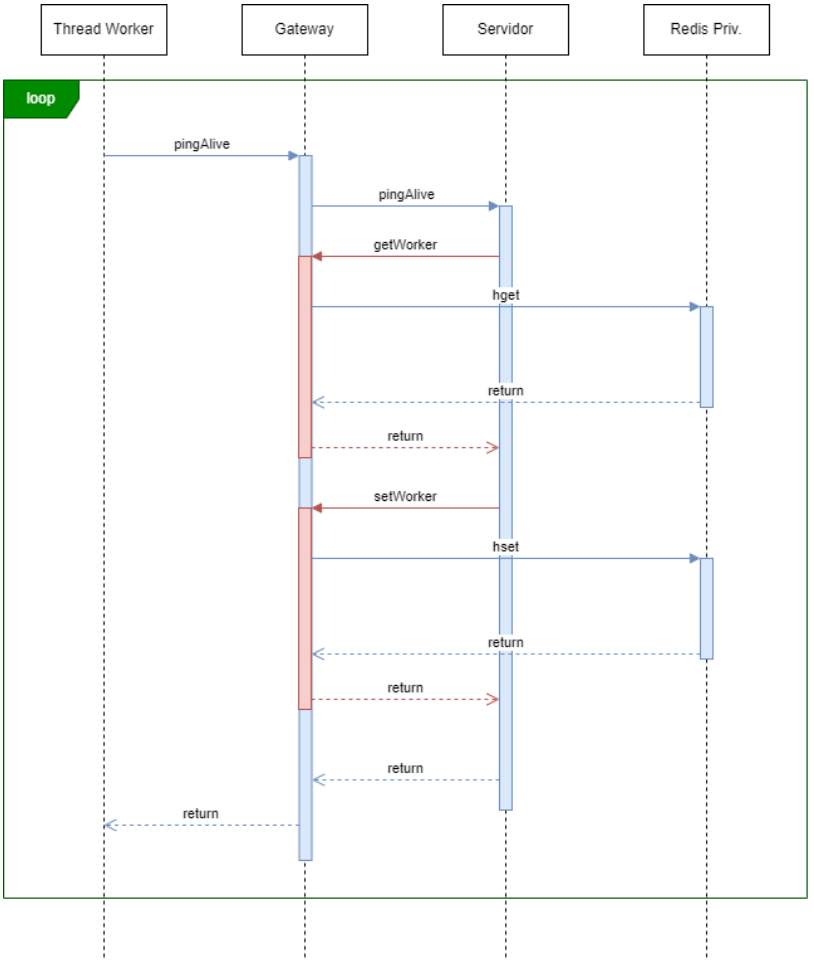
7.3. Servidor: sendPingAlive



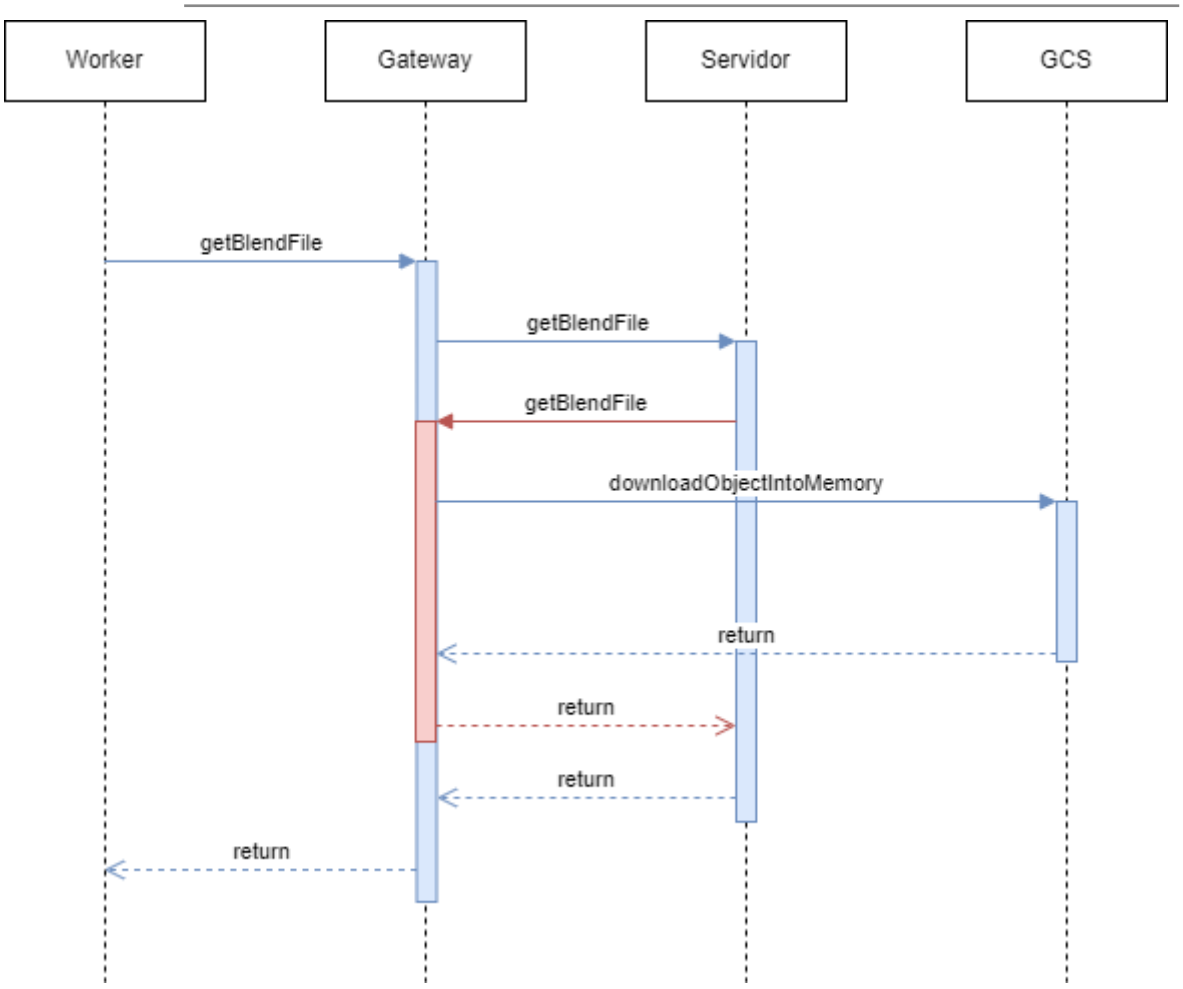
7.4. Worker: helloServer



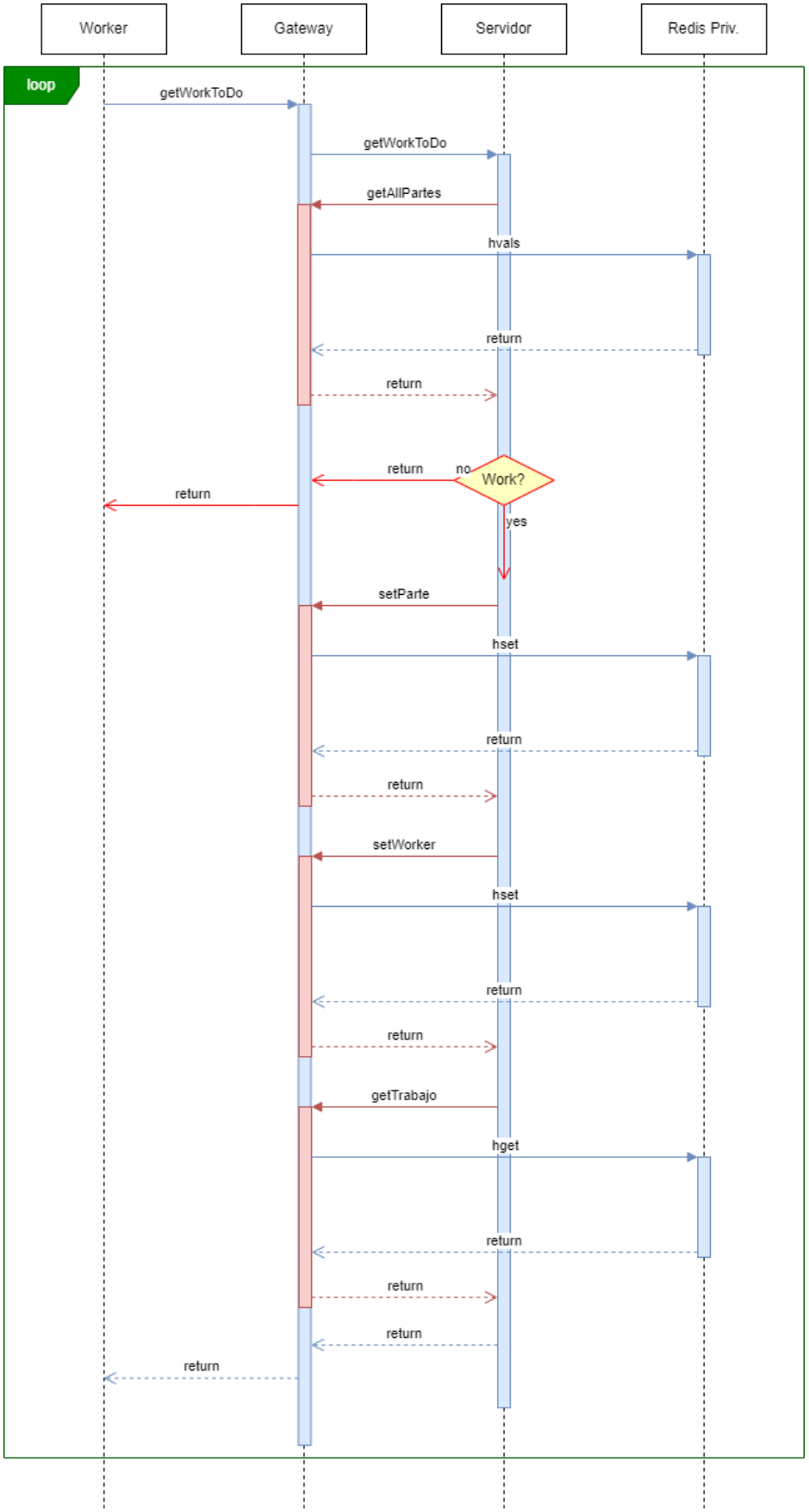
7.5. Worker: sendPingAlive



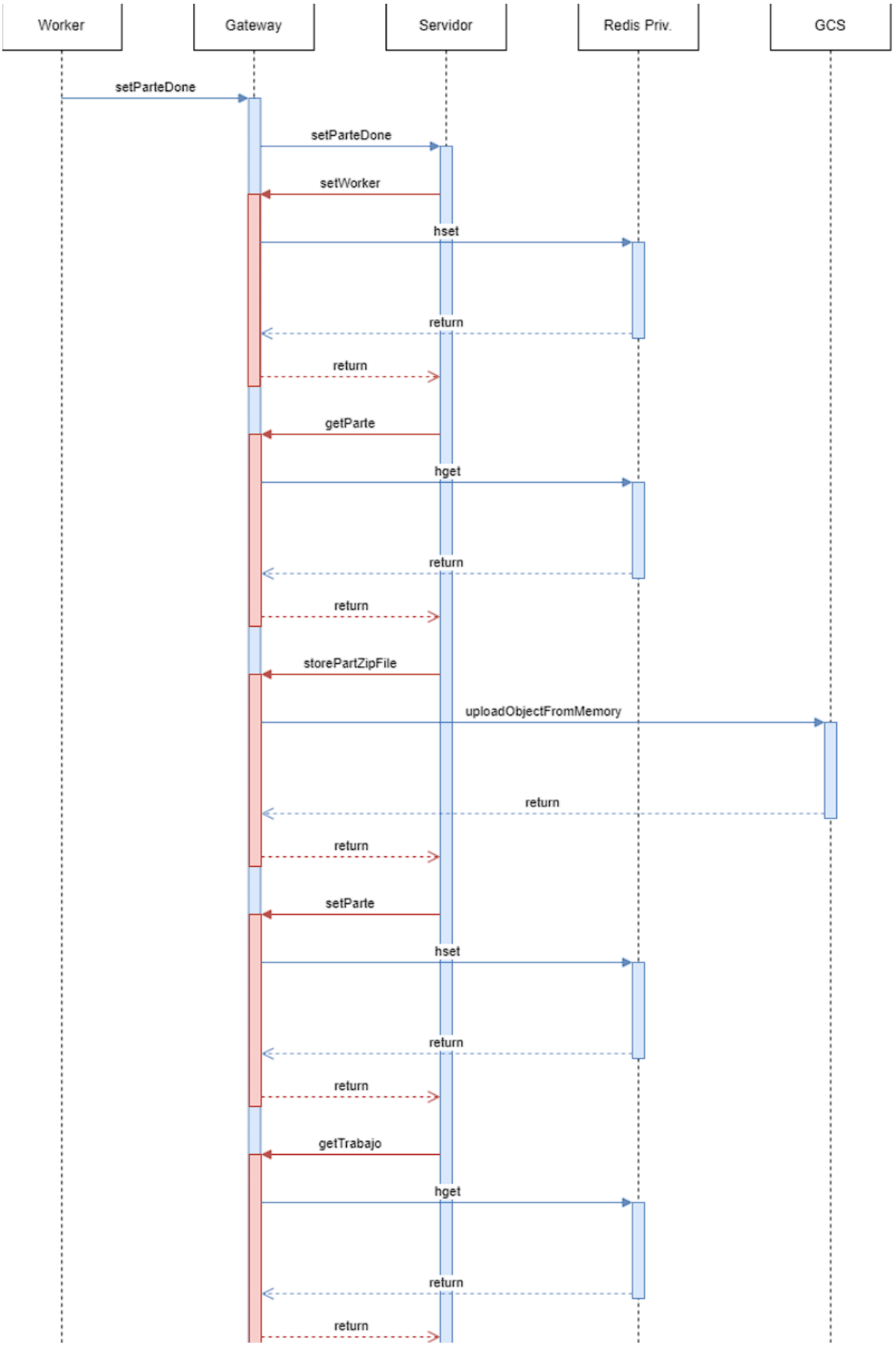
7.6. Worker: sendPingAlive

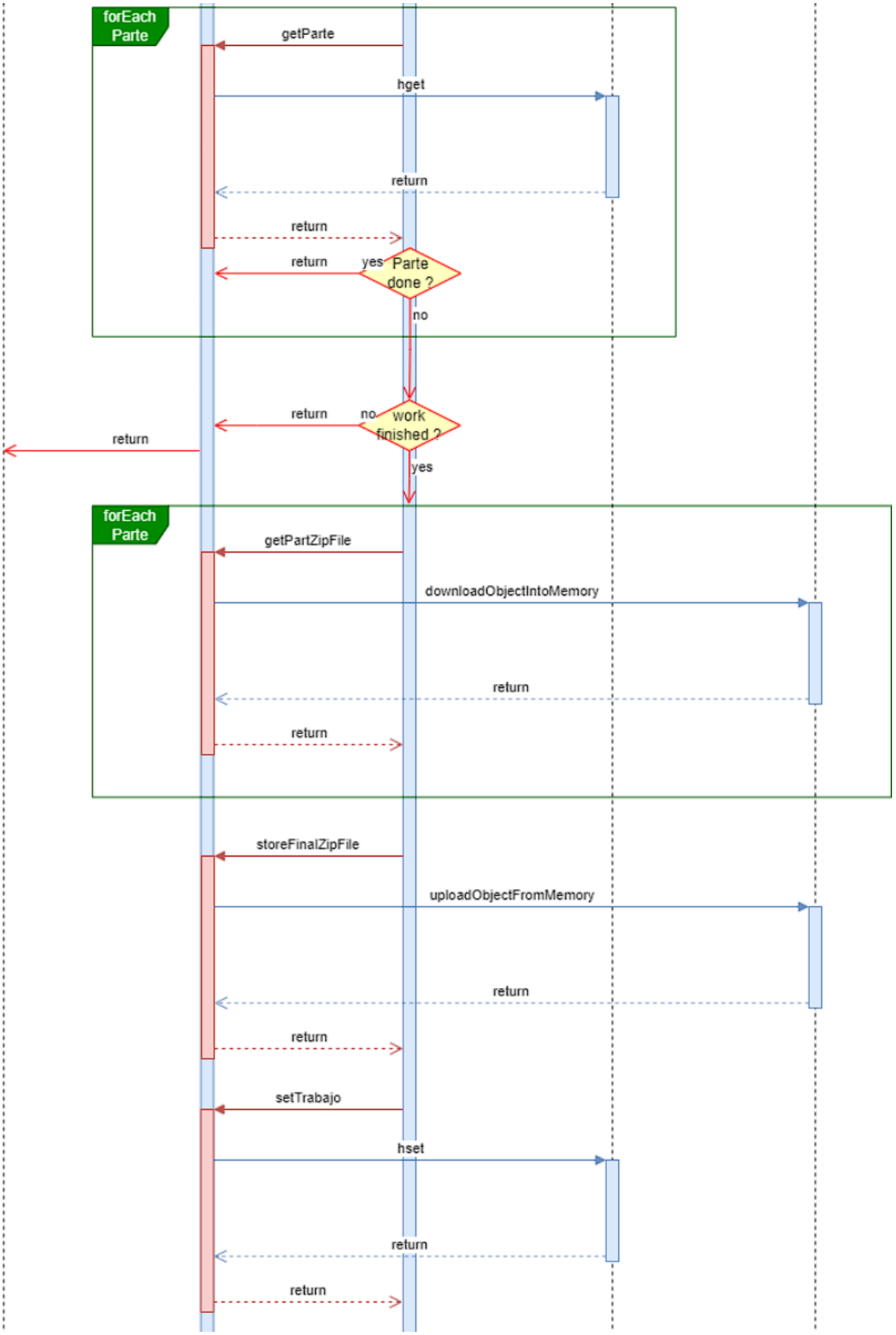


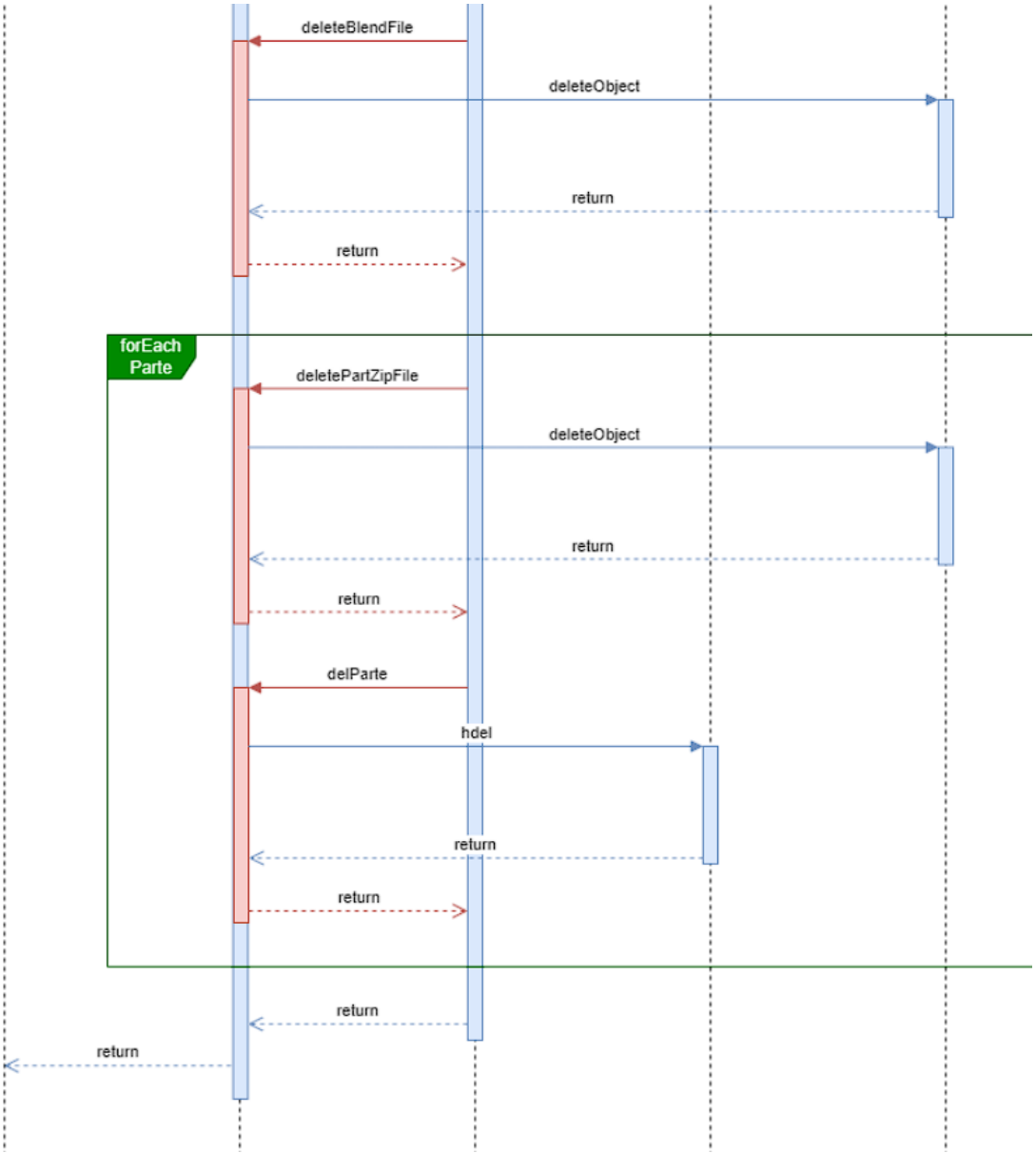
7.7. Worker: getWorkToDo



7.8. Worker: setParteDone




forEach Parte
getParte
hget
return
return
return
yes
Parte done ?
no
return
no
work finished ?
return
yes
forEach Parte
getPartZipFile
downloadObjectIntoMemory
return
return
storeFinalZipFile
uploadObjectFromMemory
return
return
setTrabajo
hset
return
return



8. Preparación de archivos .blend

---

1. Ingresar a https://free3d.com/es/modelos-3d/animated-blender.
2. Buscar un modelo (free).
   Ej.:
   https://free3d.com/es/modelo-3d/black-dragon-rigged-and-game-ready-92023.html
   Recomendado con camara incluida:
   https://storage.googleapis.com/blend-example-bucket/Dragon_2.5_For_Animations.blend
3. Descargar siempre la opción '.blend' y descomprimir en cualquier ubicación.
4. Eliminar cualquier archivo que no sea .blend.

9. Ejecución de la aplicación en ambiente local

1. Run Gateway:
   Dirigirse a la ubicación en la que se encuentra gateway.jar y por consola escribir: ***java -jar gateway.jar***
2. Run Servidor:
   Dirigirse a la ubicación en la que se encuentra servidor.jar y por consola escribir: ***java -jar servidor.jar***
3. Run Worker:
   Dirigirse a la ubicación en la que se encuentra worker.jar y por consola escribir: ***java -jar worker.jar***
4. Run Cliente:
   Dirigirse a la ubicación en la que se encuentra cliente.jar y por consola escribir: ***java -jar cliente.jar***
5. Run Clean-Gateway:
   Dirigirse a la ubicación en la que se encuentra clean-gateway.jar y por consola escribir: ***java -jar clean-gateway.jar***

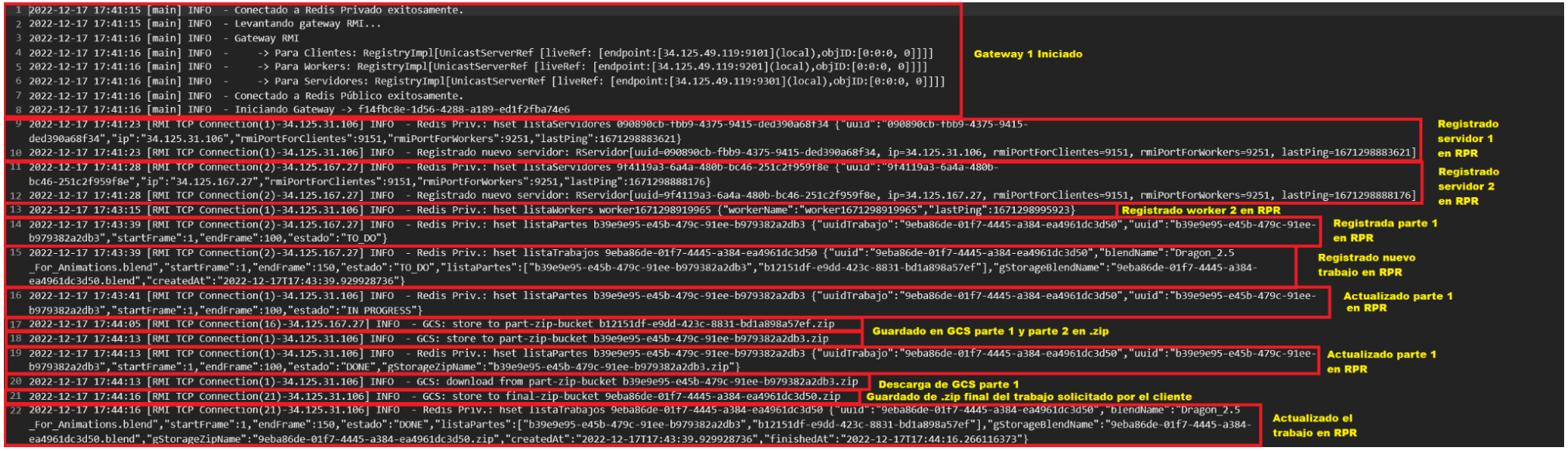
   Nota: Por cada uno se pueden crear múltiples instancias.

10. Pruebas de estabilidad y tolerancia a fallos

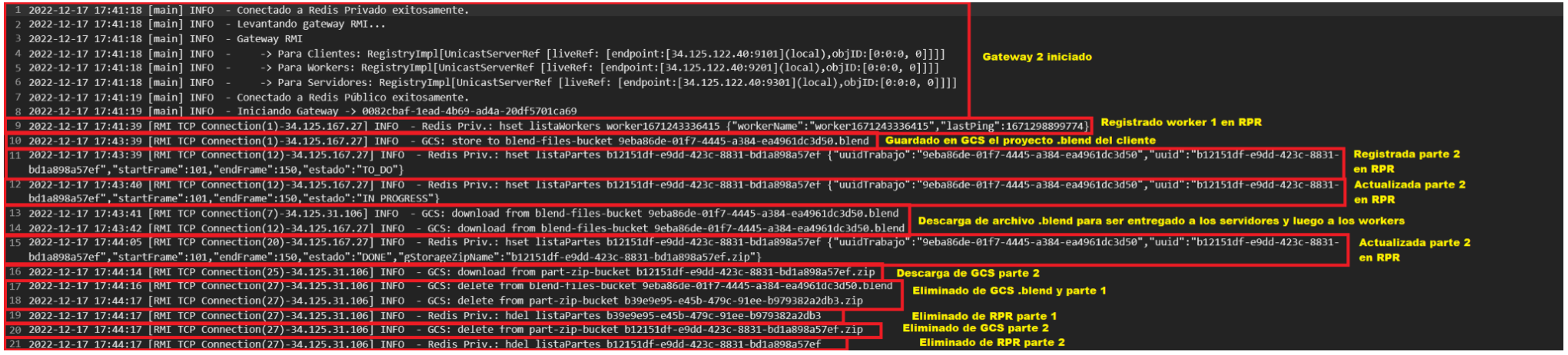
A continuación se muestran y detallan los logs de cada Nodo, luego de ejecutar la siguiente serie de pasos:

- Run Gateway 1
- Run Gateway 2
- Run Servidor 1
- Run Servidor 2
- Run Worker 1
- Run Worker 2
- Run Cliente
- Cliente envia trabajo a realizar

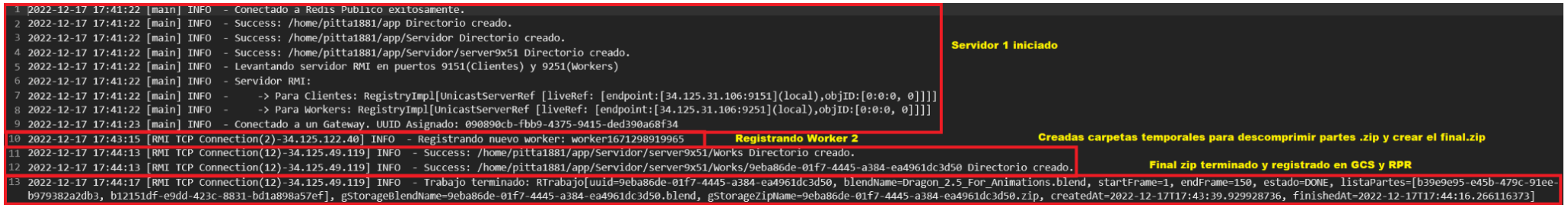
Pantalla Gateway 1



Pantalla Gateway 2

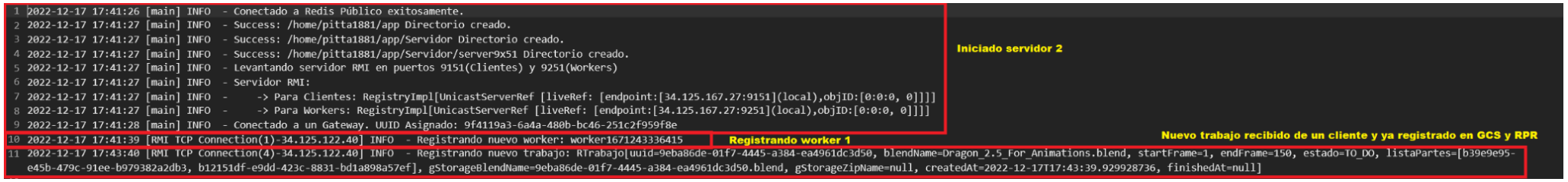


Pantalla Servidor 1



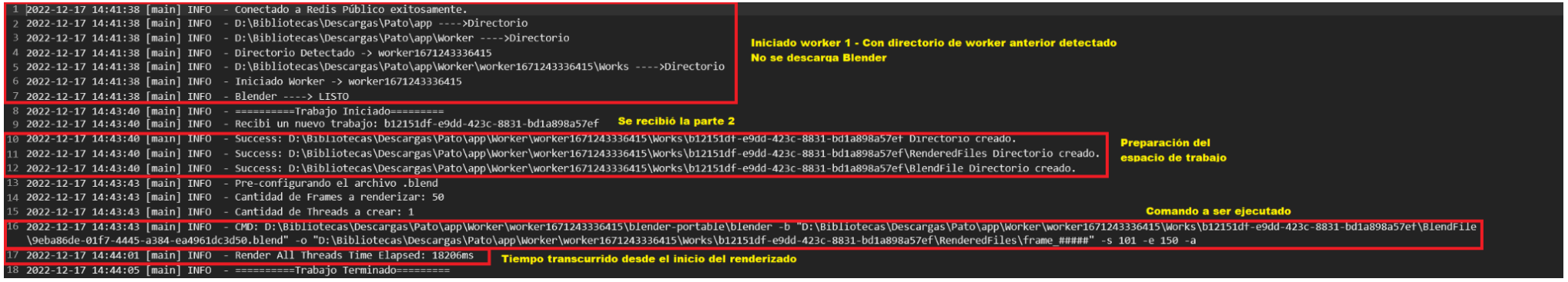

```
2022-12-17 17:41:22 [main] INFO  - Conectado a Redis Publico exitosamente.
2022-12-17 17:41:22 [main] INFO  - Success: /home/pitta1881/app Directorio creado.
2022-12-17 17:41:22 [main] INFO  - Success: /home/pitta1881/app/Servidor Directorio creado.
2022-12-17 17:41:22 [main] INFO  - Success: /home/pitta1881/app/Servidor/server9x51 Directorio creado.
2022-12-17 17:41:22 [main] INFO  - Levantando servidor RMI en puertos 9151(Clientes) y 9251(Workers)
2022-12-17 17:41:22 [main] INFO  - Servidor RMI:
2022-12-17 17:41:22 [main] INFO  -     -> Para Clientes: RegistryImpl[UnicastServerRef [liveRef: [endpoint:[34.125.31.106:9151](local),objID:[0:0:0, 0]]]]
2022-12-17 17:41:22 [main] INFO  -     -> Para Workers: RegistryImpl[UnicastServerRef [liveRef: [endpoint:[34.125.31.106:9251](local),objID:[0:0:0, 0]]]]
2022-12-17 17:41:23 [main] INFO  - Conectado a un Gateway. UUID Asignado: 090890cb-fbb9-4375-9415-ded390a68f34
2022-12-17 17:43:15 [RMI TCP Connection(2)-34.125.122.40] INFO  - Registrando nuevo worker: worker1671298919965
2022-12-17 17:44:13 [RMI TCP Connection(12)-34.125.49.119] INFO  - Success: /home/pitta1881/app/Servidor/server9x51/Works Directorio creado.
2022-12-17 17:44:13 [RMI TCP Connection(12)-34.125.49.119] INFO  - Success: /home/pitta1881/app/Servidor/server9x51/Works/9eba86de-01f7-4445-a384-ea4961dc3d50 Directorio creado.
2022-12-17 17:44:17 [RMI TCP Connection(12)-34.125.49.119] INFO  - Trabajo terminado: RTrabajo[uuid=9eba86de-01f7-4445-a384-ea4961dc3d50, blendName=Dragon_2.5_For_Animations.blend, startFrame=1, endFrame=150, estado=DONE, listaPartes=[b39e9e95-e45b-479c-91ee-b979382a2db3, b12151df-e9dd-423c-8831-bd1a898a57ef], gStorageBlendName=9eba86de-01f7-4445-a384-ea4961dc3d50.blend, gStorageZipName=9eba86de-01f7-4445-a384-ea4961dc3d50.zip, createdAt=2022-12-17T17:43:39.929928736, finishedAt=2022-12-17T17:44:16.266116373]
```

Pantalla Servidor 2



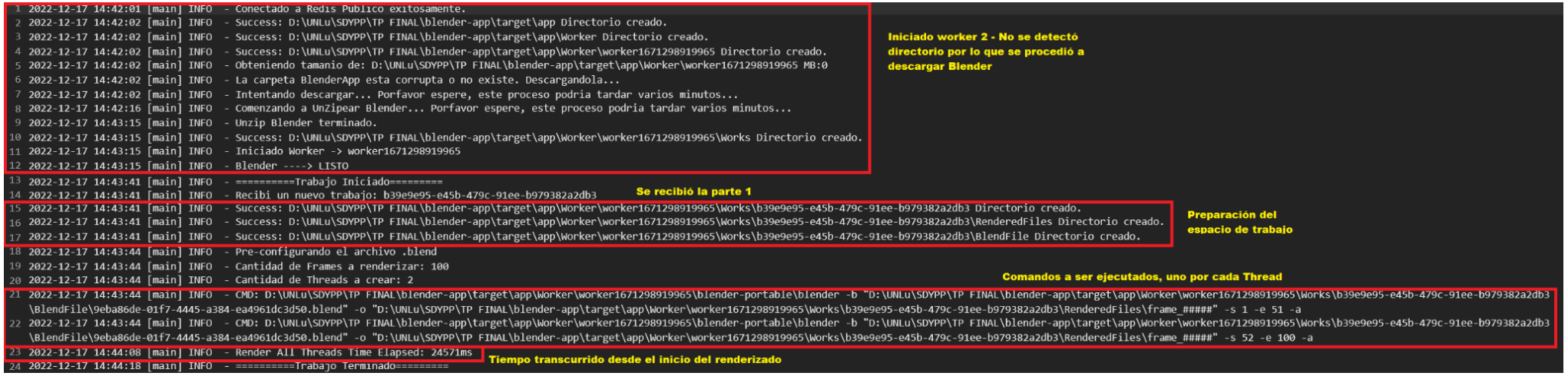

```
2022-12-17 17:41:26 [main] INFO  - Conectado a Redis Público exitosamente.
2022-12-17 17:41:27 [main] INFO  - Success: /home/pitta1881/app Directorio creado.
2022-12-17 17:41:27 [main] INFO  - Success: /home/pitta1881/app/Servidor Directorio creado.
2022-12-17 17:41:27 [main] INFO  - Success: /home/pitta1881/app/Servidor/server9x51 Directorio creado.
2022-12-17 17:41:27 [main] INFO  - Levantando servidor RMI en puertos 9151(Clientes) y 9251(Workers)
2022-12-17 17:41:27 [main] INFO  - Servidor RMI:
2022-12-17 17:41:27 [main] INFO  -     -> Para Clientes: RegistryImpl[UnicastServerRef [liveRef: [endpoint:[34.125.167.27:9151](local),objID:[0:0:0, 0]]]]
2022-12-17 17:41:27 [main] INFO  -     -> Para Workers: RegistryImpl[UnicastServerRef [liveRef: [endpoint:[34.125.167.27:9251](local),objID:[0:0:0, 0]]]]
2022-12-17 17:41:28 [main] INFO  - Conectado a un Gateway. UUID Asignado: 9f4119a3-6a4a-480b-bc46-251c2f959f8e
2022-12-17 17:41:39 [RMI TCP Connection(1)-34.125.122.40] INFO  - Registrando nuevo worker: worker1671243336415
2022-12-17 17:43:40 [RMI TCP Connection(4)-34.125.122.40] INFO  - Registrando nuevo trabajo: RTrabajo[uuid=9eba86de-01f7-4445-a384-ea4961dc3d50, blendName=Dragon_2.5_For_Animations.blend, startFrame=1, endFrame=150, estado=TO_DO, listaPartes=[b39e9e95-e45b-479c-91ee-b979382a2db3, b12151df-e9dd-423c-8831-bd1a898a57ef], gStorageBlendName=9eba86de-01f7-4445-a384-ea4961dc3d50.blend, gStorageZipName=null, createdAt=2022-12-17T17:43:39.929928736, finishedAt=null]
```

Pantalla Worker 1

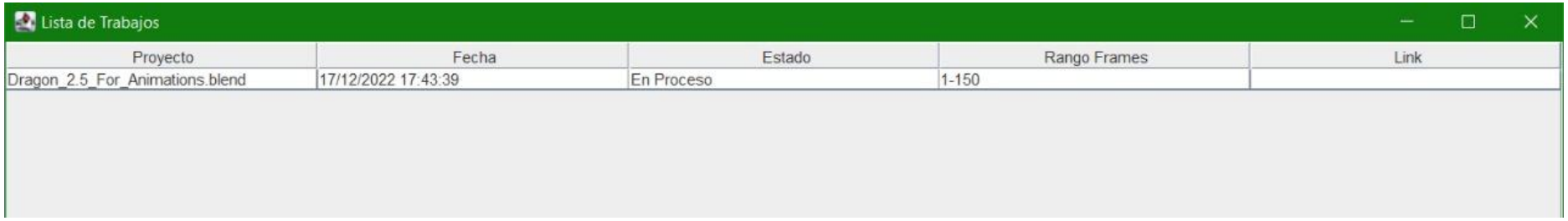


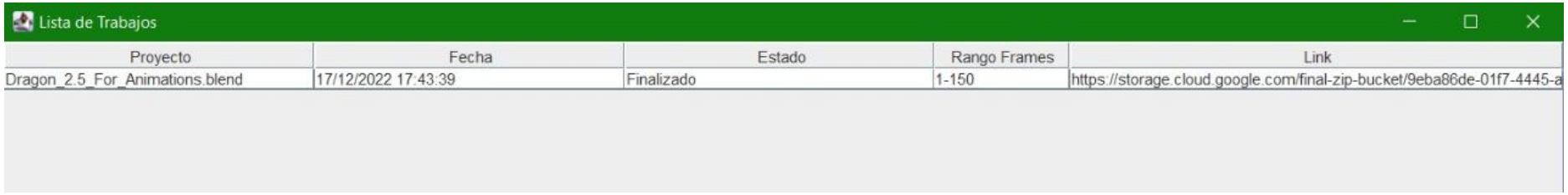

```
2022-12-17 14:41:38 [main] INFO  - Conectado a Redis Público exitosamente.
2022-12-17 14:41:38 [main] INFO  - D:\Bibliotecas\Descargas\Pato\app ---->Directorio
2022-12-17 14:41:38 [main] INFO  - D:\Bibliotecas\Descargas\Pato\app\Worker ---->Directorio
2022-12-17 14:41:38 [main] INFO  - Directorio Detectado -> worker1671243336415
2022-12-17 14:41:38 [main] INFO  - D:\Bibliotecas\Descargas\Pato\app\Worker\worker1671243336415\Works ---->Directorio
2022-12-17 14:41:38 [main] INFO  - Iniciado Worker -> worker1671243336415
2022-12-17 14:41:38 [main] INFO  - Blender ----> LISTO
2022-12-17 14:43:40 [main] INFO  - =========Trabajo Iniciado========
2022-12-17 14:43:40 [main] INFO  - Recibi un nuevo trabajo: b12151df-e9dd-423c-8831-bd1a898a57ef
2022-12-17 14:43:40 [main] INFO  - Success: D:\Bibliotecas\Descargas\Pato\app\Worker\worker1671243336415\Works\b12151df-e9dd-423c-8831-bd1a898a57ef Directorio creado.
2022-12-17 14:43:40 [main] INFO  - Success: D:\Bibliotecas\Descargas\Pato\app\Worker\worker1671243336415\Works\b12151df-e9dd-423c-8831-bd1a898a57ef\RenderedFiles Directorio creado.
2022-12-17 14:43:40 [main] INFO  - Success: D:\Bibliotecas\Descargas\Pato\app\Worker\worker1671243336415\Works\b12151df-e9dd-423c-8831-bd1a898a57ef\BlendFile Directorio creado.
2022-12-17 14:43:43 [main] INFO  - Pre-configurando el archivo .blend
2022-12-17 14:43:43 [main] INFO  - Cantidad de Frames a renderizar: 50
2022-12-17 14:43:43 [main] INFO  - Cantidad de Threads a crear: 1
2022-12-17 14:43:43 [main] INFO  - CMD: D:\Bibliotecas\Descargas\Pato\app\Worker\worker1671243336415\blender-portable\blender -b "D:\Bibliotecas\Descargas\Pato\app\Worker\worker1671243336415\Works\b12151df-e9dd-423c-8831-bd1a898a57ef\BlendFile
\9eba86de-01f7-4445-a384-ea4961dc3d50.blend" -o "D:\Bibliotecas\Descargas\Pato\app\Worker\worker1671243336415\Works\b12151df-e9dd-423c-8831-bd1a898a57ef\RenderedFiles\frame_#####" -s 101 -e 150 -a
2022-12-17 14:44:01 [main] INFO  - Render All Threads Time Elapsed: 18206ms
2022-12-17 14:44:05 [main] INFO  - =========Trabajo Terminado========
```

Pantalla Worker 2



Pantallas Cliente





Continuando con el estado final de los Nodos, a prosigue a ejecutar la siguiente serie de pasos:

- Kill Worker 1
- Kill Servidor 2
- Run Clean-Gateways
- Kill Gateway 2
- Kill Worker 2

Pantalla Gateway 1

```
2022-12-17 17:45:28 [Thread-1] INFO  - Redis Priv.: hdel listaWorkers worker1671243336415
2022-12-17 17:45:28 [Thread-1] INFO  - Worker worker1671243336415 eliminado por timeout.
2022-12-17 17:45:37 [RMI TCP Connection(24)-190.244.135.190] ERROR - Error: Connection refused to host: 34.125.167.27; nested exception is:
      java.net.ConnectException: Connection refused
2022-12-17 17:45:37 [RMI TCP Connection(24)-190.244.135.190] ERROR - Error al conectar con el SERVIDOR 34.125.167.27:9251
2022-12-17 17:45:37 [RMI TCP Connection(24)-190.244.135.190] INFO  - Reintentando con el próximo SERVIDOR ...
2022-12-17 17:45:39 [RMI TCP Connection(24)-190.244.135.190] ERROR - Error: Connection refused to host: 34.125.167.27; nested exception is:
      java.net.ConnectException: Connection refused
2022-12-17 17:45:39 [RMI TCP Connection(24)-190.244.135.190] ERROR - Error al conectar con el SERVIDOR 34.125.167.27:9251
2022-12-17 17:45:39 [RMI TCP Connection(24)-190.244.135.190] INFO  - Reintentando con el próximo SERVIDOR ...
2022-12-17 17:45:41 [RMI TCP Connection(24)-190.244.135.190] ERROR - Error: Connection refused to host: 34.125.167.27; nested exception is:
      java.net.ConnectException: Connection refused
2022-12-17 17:45:41 [RMI TCP Connection(24)-190.244.135.190] ERROR - Error al conectar con el SERVIDOR 34.125.167.27:9251
2022-12-17 17:45:41 [RMI TCP Connection(24)-190.244.135.190] INFO  - Reintentando con el próximo SERVIDOR ...
2022-12-17 17:45:43 [RMI TCP Connection(24)-190.244.135.190] ERROR - Error: Connection refused to host: 34.125.167.27; nested exception is:
      java.net.ConnectException: Connection refused
2022-12-17 17:45:43 [RMI TCP Connection(24)-190.244.135.190] ERROR - Error al conectar con el SERVIDOR 34.125.167.27:9251
2022-12-17 17:45:43 [RMI TCP Connection(24)-190.244.135.190] INFO  - Reintentando con el próximo SERVIDOR ...
2022-12-17 17:45:45 [RMI TCP Connection(24)-190.244.135.190] ERROR - Error: Connection refused to host: 34.125.167.27; nested exception is:
      java.net.ConnectException: Connection refused
2022-12-17 17:45:45 [RMI TCP Connection(24)-190.244.135.190] ERROR - Error al conectar con el SERVIDOR 34.125.167.27:9251
2022-12-17 17:45:45 [RMI TCP Connection(24)-190.244.135.190] INFO  - Reintentando con el próximo SERVIDOR ...
2022-12-17 17:45:46 [RMI TCP Connection(30)-190.244.135.190] ERROR - Error: Connection refused to host: 34.125.167.27; nested exception is:
      java.net.ConnectException: Connection refused
2022-12-17 17:45:46 [RMI TCP Connection(30)-190.244.135.190] ERROR - Error al conectar con el SERVIDOR 34.125.167.27:9251
2022-12-17 17:45:46 [RMI TCP Connection(30)-190.244.135.190] INFO  - Reintentando con el próximo SERVIDOR ...
2022-12-17 17:45:48 [Thread-0] INFO  - Redis Priv.: hdel listaServidores 9f4119a3-6a4a-480b-bc46-251c2f959f8e
2022-12-17 17:45:48 [Thread-0] INFO  - Servidor 9f4119a3-6a4a-480b-bc46-251c2f959f8e eliminado por timeout.
2022-12-17 17:46:48 [Thread-1] INFO  - Redis Priv.: hdel listaWorkers worker1671298919965
2022-12-17 17:46:48 [Thread-1] INFO  - Worker worker1671298919965 eliminado por timeout.
```

Se detecta la caida del worker 1 y se lo elimina de RPR

Entran peticiones de Clientes enviando trabajos o Workers haciendo ping pero no se puede redigirir al servidor 2 porque está caído.

Se detecta la caida del servidor 2 y se lo elimina de RPR

Se detecta la caida del worker 2 y se lo elimina de RPR

Pantalla Gateway 2

```
2022-12-17 17:45:50 [RMI TCP Connection(31)-190.244.135.190] ERROR - Error: Connection refused to host: 34.125.167.27; nested exception is:
      java.net.ConnectException: Connection refused
2022-12-17 17:45:50 [RMI TCP Connection(31)-190.244.135.190] ERROR - Error al conectar con el SERVIDOR 34.125.167.27:9251
2022-12-17 17:45:50 [RMI TCP Connection(31)-190.244.135.190] INFO  - Reintentando con el próximo SERVIDOR ...
```

No se pueden redirigir peticiones al servidor 2, hasta que se actualize la lista de servidores en memoria y lo elimine

Servidor eliminado y peticiones redirigidas solo al servidor 1

Pantalla Servidor 1

```
2022-12-17 17:46:31 [RMI TCP Connection(12)-34.125.49.119] ERROR - Error: Connection refused to host: 34.125.122.40; nested exception is:
     java.net.ConnectException: Connection refused
2022-12-17 17:46:31 [RMI TCP Connection(12)-34.125.49.119] ERROR - Error al conectar con el GATEWAY 34.125.122.40:9301
2022-12-17 17:46:31 [RMI TCP Connection(12)-34.125.49.119] INFO  - Reintentando con el próximo GATEWAY ...
2022-12-17 17:46:39 [RMI TCP Connection(12)-34.125.49.119] ERROR - Error: Connection refused to host: 34.125.122.40; nested exception is:
     java.net.ConnectException: Connection refused
2022-12-17 17:46:39 [RMI TCP Connection(12)-34.125.49.119] ERROR - Error al conectar con el GATEWAY 34.125.122.40:9301
2022-12-17 17:46:39 [RMI TCP Connection(12)-34.125.49.119] INFO  - Reintentando con el próximo GATEWAY ...
2022-12-17 17:46:41 [RMI TCP Connection(12)-34.125.49.119] ERROR - Error: Connection refused to host: 34.125.122.40; nested exception is:
     java.net.ConnectException: Connection refused
2022-12-17 17:46:41 [RMI TCP Connection(12)-34.125.49.119] ERROR - Error al conectar con el GATEWAY 34.125.122.40:9301
2022-12-17 17:46:41 [RMI TCP Connection(12)-34.125.49.119] INFO  - Reintentando con el próximo GATEWAY ...
```

No puede enviar solicitudes ni ping al gateway 2. Va a seguir intentando hasta actualizar la lista de gateways en memoria

Pantalla Clean Gateway

```
2022-12-17 17:46:21 [main] INFO  - Conectado a Redis Público exitosamente.
2022-12-17 17:46:41 [main] INFO  - Redis Pub.: hdel listaGateways 0082cbaf-1ead-4b69-ad4a-20df5701ca69
2022-12-17 17:46:41 [main] INFO  - Gateway 0082cbaf-1ead-4b69-ad4a-20df5701ca69 eliminado por timeout.
```

Inicia el servicio

Detecta la caida del gateway 2 y lo elimina de RPU

Pantalla Worker 2

```
2022-12-17 14:46:36 [Thread-1] ERROR - Error al conectar con el GATEWAY 34.125.122.40:9201
2022-12-17 14:46:36 [Thread-1] INFO  - Reintentando con el próximo GATEWAY ...
2022-12-17 14:46:38 [main] ERROR - Error al conectar con el GATEWAY 34.125.122.40:9201
2022-12-17 14:46:38 [main] INFO  - Reintentando con el próximo GATEWAY ...
2022-12-17 14:46:46 [Thread-1] ERROR - Error al conectar con el GATEWAY 34.125.122.40:9201
2022-12-17 14:46:46 [Thread-1] INFO  - Reintentando con el próximo GATEWAY ...
2022-12-17 14:46:49 [main] ERROR - Error al conectar con el GATEWAY 34.125.122.40:9201
2022-12-17 14:46:49 [main] INFO  - Reintentando con el próximo GATEWAY ...
2022-12-17 14:46:51 [Thread-1] ERROR - Error al conectar con el GATEWAY 34.125.122.40:9201
2022-12-17 14:46:51 [Thread-1] INFO  - Reintentando con el próximo GATEWAY ...
```

No puede enviar solicitudes ni ping al gateway 2. Va a seguir intentando hasta actualizar la lista de gateways en memoria

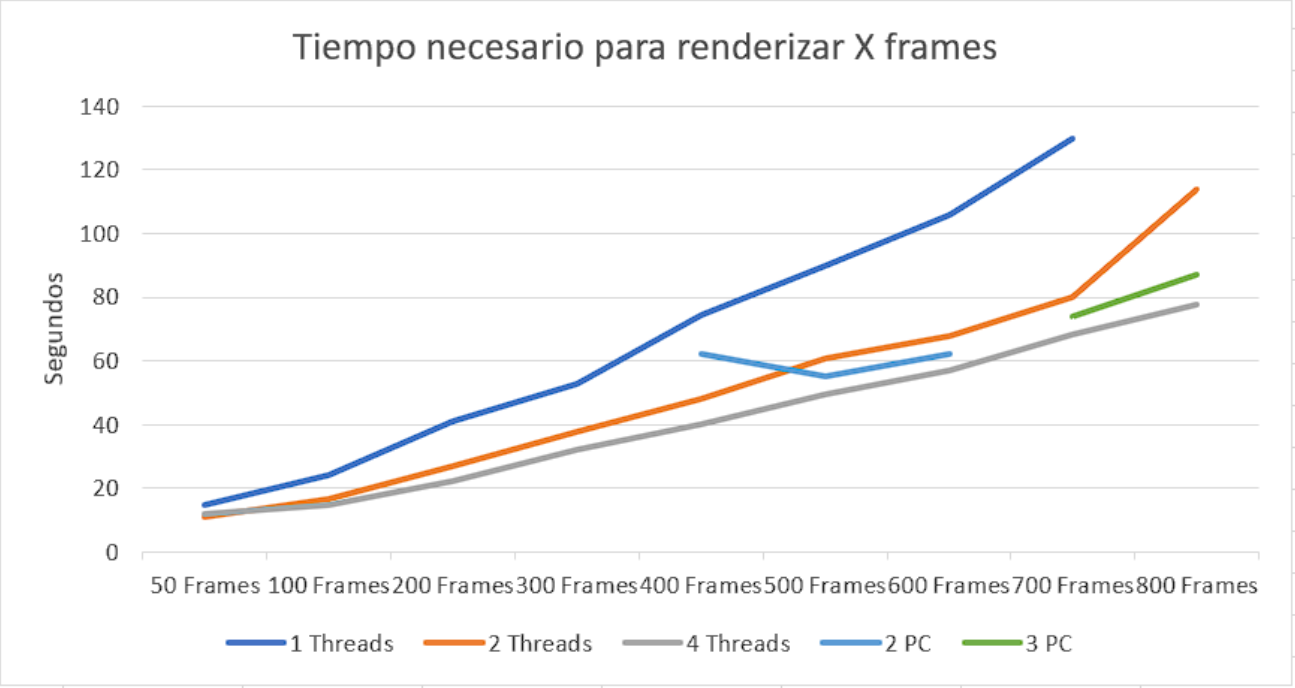
11. Rendimiento Centralizado vs Distribuido

11.1. Hardware utilizado

- PC1:
  - Windows 10
  - CPU: Intel i5 2500k
  - GPU: Radeon RX 580 4gb
  - 16Gb Ram
- PC2 (Notebook Gamer):
  - Windows 10
  - CPU: Intel i5 10300H
  - GPU: GeForce GTX 1650
  - 8Gb Ram
- PC3:
  - Windows 10
  - CPU: AMD Ryzen 3 3200G
  - GPU: Radeon Vega 8
  - 16Gb Ram

11.2. Comparación

	50 Frames	100 Frames	200 Frames	300 Frames	400 Frames	500 Frames	600 Frames	700 Frames	800 Frames
1 Threads	14,54	24,17	40,95	52,55	74,4	90	105,65	130,04	
2 Threads	11,04	16,67	26,87	37,57	48,09	60,6	67,97	80,05	113,69
4 Threads	11,75	14,54	22,22	32,05	40,03	49,4	57,04	68,11	77,61
2 PC					62	55	62		
3 PC								74	87



Observamos que a pesar de usar un renderizado distribuido con 2 y 3 PCs, sigue siendo mejor un renderizado centralizado con 4 threads. Aún así hay que tener en cuenta algunas variables:

- Al ser un renderizado distribuido en varias máquinas con diferente hardware, el tiempo final va a estar atado a aquella con el peor hardware, en nuestro laboratorio era la PC3 que no cuenta con GPU dedicada, por lo que siempre debemos esperarla.
- Al ser un sistema distribuido por internet, debemos tener en cuenta la velocidad de bajada y subida que tengan los Workers en su espacio de trabajo, así como el estado de la red en general.
- Se debe tener en cuenta el proceso de Zipeado y Unzipeado de partes que realiza el Worker y luego el Servidor, el cual es un proceso costoso y depende del poder de la CPU y de la velocidad de lectura/escritura del HDD/SDD.

12. Conclusión

Notamos que debe haber restricciones con especificaciones mínimas de hardware que debe cumplir una máquina para ser un Worker y evitar que el rendimiento general del sistema esté supeditado al Worker con peores especificaciones. Asimismo los Servidores deben tener al menos un SSD y una CPU preferentemente con “aceleración de compresión de hardware” para mejorar la compresión/descompresión de archivos .zip.

Finalmente no debemos olvidar que nuestro sistema brinda un servicio de renderizado distribuido, por lo que si un Cliente no cuenta con GPU dedicada, SIEMPRE le será más conveniente utilizar nuestra aplicación que utilizar su GPU integrada

13. Notas

- Link al repositorio de trabajo: https://github.com/pitta1881/blender-app
- Es posible que el archivo .blend descargado no contenga una 'Camara', por lo que no se podrá renderizar. Para agregar una camara seguir el siguiente tutorial: https://www.youtube.com/watch?t=46&v=aY04h4ujrlY&feature=youtu.be
- Para comprobar cuantos frames tiene una animación, se debe abrir el archivo .blend, ir al tab 'Animation' y en la parte inferior de la pantalla se encuentra dos variables Start y End.
- En el root del repositorio en git, se incluye un archivo .blend con cámara incluida listo para probar (Dragon_2.5_For_Animations.blend). Tambíen puede ser descargado desde https://storage.googleapis.com/blend-example-bucket/Dragon_2.5_For_Animations.blend

14. Glosario

Nodo: Gateway, Servidor, Worker o Cliente.

15. Abreviaturas

SO: Sistema Operativo.
CRUD: Create-Read-Update-Delete.
GCS: Google Cloud Storage.
RPR: Redis Privado.
RPU: Redis Público.
LG: Lista de Gateways.
LS: Lista de Servidores.
LW: Lista de Workers.
LT: Lista de Trabajos.
LP: Lista de Partes.