Documento de pruebas del proyecto

Programación de Sistemas Concurrentes y Distribuidos

2º curso, Grado de Ingeniería en Informática, 2021-2022

Número de equipo: 11

Integrantes:

Héctor Arcega Vela, 818058

Cecilia Monteagudo Redrado, 820417

Berta Olano Guillén, 815935

Carlos Palomera Oliva, 820949

Alonso Lucas Juberías, 815767

Simón Alonso Gutiérrez, 821038

Índice de contenidos

1. Introducción

El objetivo de este documento es dejar constancia de las pruebas realizadas por el equipo para que finalmente todo funcione correctamente. La metodología llevada a cabo es, para las conexiones simplemente ejecutar los programas con los canales en los puertos requeridos, y para la función de cada programa por separado, crear pequeños programas aparte que ya cuenten con que el resto del trabajo funciona de forma adecuada y vayan comprobando que cada función de los procesos realiza su trabajo.

2. Funcionalidad clave del sistema y alcance de los correspondientes casos de prueba

En primer lugar probamos que las conexiones mediante los canales se realicen correctamente sin que el programa haga realmente nada. Segundo, que cada proceso por separado realice la tarea que le corresponde asumiendo que los demás cumplen lo establecido en el enunciado. Luego comprobar que en localhost los programas ya desarrollados se conectan de manera adecuada. Observar también si en localhost en la totalidad del trabajo se obtiene el resultado esperado y por último comprobar que todo va bien en varias máquinas.

3. Definición de los casos de prueba

En esta sección se van a enumerar y describir en detalle cada uno de los casos de prueba que se han realizado para verificar la funcionalidad y el correcto comportamiento del sistema.

3.1 Pruebas gestor de colas

Comunicación del gestor de colas con el exterior	Responsable	Hector y Simón
	Fecha	3/01/2022

Descripción:

Para este caso vamos a comprobar si el gestor de colas es capaz de conectarse con el master, los workers y los analizadores.

Prerrequisitos (si fueran necesarios)

Las conexiones con el gestor de colas deben estar implementadas en el resto de programas. Las conexiones deben hacerse es este orden:

master > workers > analizadores

Pasos (metodología):

Primero hemos comprobado que el gestor de colas se pueda conectar con el master, una vez hecho eso y asegurada la conexión del máster con streaming, comprobamos que la información del máster la reciba correctamente el gestor de colas.

Segundo, comprobamos que los workers puedan pedir la petición de enviar información para que el gestor de colas les envíe la información de la cola correspondiente. En el caso de que los workers pidan la petición de recibir, eso significa que los workers van a enviar información para que el gestor de colas la reciba. Almacenando la información en la cola correspondiente ya sean tags o qos.

Tercero, comprobamos que los analizadores puedan pedir la petición de recibir información para que el gestor de colas les envíe la información correspondiente a cada uno de los analizadores. Y comprobamos que cada uno va por el canal correspondiente y que no se entrelazan.

Resultado esperado:

El resultado esperado es que el master consiga enviarle la información al gestor de colas, que el gestor de colas reciba las peticiones de los workers para saber si estos están enviando o recibiendo información. Y por último, que el gestor de colas reciba las peticiones para enviar la información a los dos analizadores(tags y qos).

Resultado obtenido:

En primer lugar conseguimos conectar con master pero la conexión de master y streaming tenía algún pequeño problema, por lo que después de excluir en el proceso master la conexión con streaming pudimos conectarnos sin problema. Cuando este error se solucionó, master recibía los tweets correctamente y los enviaba al gestor sin fallos.

Las conexiones con los workers funcionaron correctamente cuando decidimos que enviasen un mensaje al gestor indicando si querían envíar o recibir datos ya que la conexión con estos es bidireccional.

Con los analizadores no hubo problema ya que nuestra función era solo enviar los tags y qos de sus respectivas colas a los analizadores.

Exclusión mutua y correcto funcionamiento del gestor de colas	Responsable	Hector y Simón
	Fecha	22/12/2021

Descripción:

Para este caso vamos a comprobar si las colas no se interrumpen unas con otras, es decir, que si una está bloqueada por que ha alcanzado el tamaño máximo eso no impida que las demás puedan continuar su funcionamiento.

Prerrequisitos (si fueran necesarios)

Conexión del gestor con el monitor

Pasos (metodología):

Primero, comprobamos que cada monitor tenga una cola, con dos operaciones, una de escritura y otra de lectura. Y que están en exclusión mutua.

Segundo, comprobamos que las operaciones funcionan de modo adecuado con las BoundeQueue(colas con tamaño máximo), es decir, que si la cola está llena que no pueda añadir más elementos, y si está vacía que no pueda retirar elementos.

Resultado esperado:

Acceso a las diferentes colas del gestor y que a pesar de que alguna de ellas esté bloqueada, se pueda acceder a las demás sin esperas innecesarias.

Resultado obtenido:

En un principio pensamos en usar un solo monitor para las 3 colas pero al ver que daba problemas de bloqueo decidimos usar un modelo de monitor común (ya que las tres colas usan las mismas operaciones) pero inicializando uno para cada una de las colas para evitar estos problemas.

3.2 Pruebas master

Función separarcadena	Responsable	Cecilia, Berta
	Fecha	28/12/2021

Descripción:

Comprobamos que la función separarcadena es capaz de separar un buffer tantas veces como delimitadores hay y almacena las partes en un vector

Prerrequisitos (si fueran necesarios)

Los tweets están separados mediante /n entres sí, el vector tiene tantas componentes como tweets queremos extraer del bloque (5).

Pasos (metodología):

Primero, creamos un proceso que solo tiene como función separar un bloque en tweets, para asegurarnos que los problemas no están en ninguna otra parte del máster.

Segundo, buscamos diferentes opciones en internet que ofrecen una función similar a la que buscamos y elegimos la más cómoda y eficiente.

Después realizamos los cambios necesarios hasta comprobar que la función es capaz de realizar la tarea deseada.

Resultado esperado:

El bloque queda separado en tweets y cada uno se guarda en la componente de un vector, al final de cada tweet hay un espacio y un /n ya que ese es el formato necesario para poder extraer los tags más adelante.

Resultado obtenido:

Al principio nos daba errores en la declaración de un vector<string> que necesitábamos para utilizar las funciones push_back y erase. Tras varias pruebas conseguimos que compilara correctamente e implementamos la función en el proceso máster. Además esta función la reutilizamos en otras partes del programa, por ejemplo, para separar un tweet por palabras.

3.3 Pruebas workers

Función extraertags	Responsable	Cecilia, Berta
	Fecha	29/12/2021

Descripción:

Comprobamos que la función de extraer tags obtiene como resultado la fecha del tweet, los hashtags, las menciones y los enlaces separados por ‘;’. Guardamos los tags de cada tweet en la componente de un vector y lo enviamos en un solo string separando los tags de cada tweet por /n.

Prerrequisitos (si fueran necesarios)

Los tags están separados entre ellos por ‘;’ y los de cada tweet por ‘/n’

Pasos (metodología):

Primero necesitamos separar los tweets por palabras mediante la función separarcadena para identificar qué palabras son las que consideramos como tags.

Para extraer menciones y hashtags utilizamos la función palabras.front para ver si la primera letra es ‘@’ o ‘#’ y si lo es lo añadimos al vector. Para buscar los enlaces, tuvimos que probar diferentes cosas porque nos daban problemas. Al final encontramos la función .find(“https”) que permite comprobar si es un enlace.

Resultado esperado:

En el vector tags[] quedan guardados la fecha, enlaces, menciones y hashtags separados por ‘;’

Resultado obtenido:

Tras varias pruebas comprobamos que necesitábamos añadir un espacio al final de cada tweet para que no diese problemas y detectase bien los tags de la última palabra.

QoS unidad de tiempo	Responsable	Cecilia, Berta
	Fecha	7/12/2021

Descripción:

Uno de los qos que elegimos fue el tiempo que se tarda en procesar un tweet. Inicialmente lo habíamos puesto en milisegundos, lo que daba resultados muy similares o incluso erróneos, 0 por ejemplo.

Prerrequisitos (si fueran necesarios)

Se extrae correctamente el tiempo dentro del string qos.

Pasos (metodología):

Cambiar la unidad de tiempo de milisegundos a microsegundos en la función chrono y comprobar que el resultado extraído es más preciso.

Resultado esperado:

Obtener un tiempo diferente y preciso para cada tweet.

Resultado obtenido:

Inicialmente el resultado era 0 ya que habíamos puesto el tiempo como entero en vez de como real, pero una vez modificado esto el tiempo en microsegundos se obtenía de manera precisa y diferente para cada tweet.

3.4 Pruebas analizadores

Fiabilidad del Programa Analizador tras un largo periodo de tiempo en funcionamiento	Responsable	Carlos, Alonso
	Fecha	12/01/2022

Descripción:

Vamos a dejar en ejecución el sistema entero durante un periodo superior a los 10 minutos a la espera de que la ejecución no se interrumpa y los ficheros se sigan actualizando pasado este tiempo

Prerrequisitos (si fueran necesarios)

Todos los programas con los canales correctos que compilen adecuadamente.

Pasos (metodología):

Ejecución en orden de todos los programas a posterior una espera de 10 minutos o superior y observación durante un número de ciclos de actualización de los ficheros de texto generados por los Analizadores.

Resultado esperado:

Que tras el tiempo esperado los ficheros de texto se sigan modificando y que la ejecución no haya sido finalizada en ningún momento.

Resultado obtenido:

El fichero de texto seguía siendo actualizado periódicamente y la ejecución de los programas no había sido finalizada.

Manejo de los Tags/QOS en los Analizadores	Responsable	Carlos, Alonso
	Fecha	06/01/2022

Descripción:

Ejecución con pausas forzadas mediante el uso de std::cin para ver paso a paso cómo se guardan los tags en su vector dinámico y como se crean a partir del formato de llegada.

Prerrequisitos (si fueran necesarios)

Conexión con un programa que nos mande tags con el mismo formato que haría el gestor de colas y la correcta compilación de ambos programas (Analizadores y programa que envía tags/qos).

Pasos (metodología):

Ejecución del programa de envío de mensajes y ejecución de los analizadores, tras la creación de un tag o un qos visualización por pantalla de este (con todos sus datos como num_Veces o su posición en el vector) gracias a la pausa por cin, luego continuar la ejecución.

Resultado esperado:

Creación del Tag a partir del formato que es enviado rellenando todos sus campos de manera correcta y su posterior asignación a una coordenada del vector o actualización de su campo num_Veces y la creación de un qos a partir del formato que es enviado y la correcta actualización de las variables tiempoTotal y cuentaTotal.

Resultado obtenido:

Incorrecta manipulación del string del formato de qos con las funciones dadas por la librería string e incorrecto uso del vector dinámico de la librería vector, tras la corrección el resultado fue igual que el esperado.

3.5 Pruebas Streaming

Funcionalidad del filtro	Responsable	Carlos, Alonso
	Fecha	23/12/2021

Descripción:

Al crear el filtro que comprueba si el tweet posee los campos necesarios y elimina los caracteres inválidos, comprobamos su fiabilidad escribiendo los tweets ya filtrados en un documento de texto y vamos cotejando su fiabilidad comparándolo con los tweets “en bruto”.

Prerrequisitos (si fueran necesarios)

Tweets con todos sus caracteres, tal cual el usuario los escribe en la aplicación.

Pasos (metodología):

Abre el fichero "tweets-sinProcesar.csv", lo leerá carácter por carácter y extraerá de él los diferentes apartados de los tweets. Al leer el texto del tweet se asegurará de que sea un carácter permitido. Una vez leído lo escribirá por pantalla.

Resultado esperado:

Los tweets escritos por pantalla cumplen el formato indicado.

Resultado obtenido:

Se comprueba que el formato de los tweets es el adecuado.

El streaming crea los string con los tweets correctamente	Responsable	Carlos, Alonso
	Fecha	13/01/2022

Descripción:

Antes de enviar los tweets al proceso master nos tenemos que asegurar de que construya el string de forma correcta.

Prerrequisitos (si fueran necesarios)

El filtro del proceso ya funciona adecuadamente.

Pasos (metodología):

Los tweets, para ser filtrados, se separan en distintos apartados que luego hay que juntar de nuevo para enviarlos al master. Al juntarlos tenemos que asegurarnos de que cumplan con el formato establecido, si no lo hacen son desechados.

Resultado esperado:

Los tweets finales que luego serán compartidos con el master cumplen con el formato adecuado.

Resultado obtenido:

Comprobamos que el string con el tweet es creado de forma correcta y cumple con el formato requerido.

Manejo del filtro, su fiabilidad a largo plazo con el fichero y su conexión con el programa master.	Responsable	Carlos, Alonso
	Fecha	24/12/2021

Descripción:

Dejaremos durante el tiempo necesario (las fechas se reinicien en los tweets que mandamos) el programa streaming funcionando conectado con el programa master ambos aislados.

Prerrequisitos (si fueran necesarios)

Los programas Streaming y Master conectados correctamente y con sus conexiones correctas.

Pasos (metodología):

Ejecutaremos el master y el Streaming en orden en localhost y esperaremos un tiempo indeterminado mientras escribimos por pantalla los mensajes que se mandan.

Resultado esperado:

Una ejecución que no finalice y pueda reabrir el fichero una vez terminado.

Resultado obtenido:

El servicio de Streaming entra en un bucle infinito tras acabar el fichero, tras solucionar el problema la ejecución no termina nunca y los mensajes son correctos.

3.6 Pruebas generales

Mensajes muy largos para diferentes máquinas	Responsable	Todo el equipo
	Fecha	11/01/2022

Descripción:

Los mensajes enviados en strings por los diferentes canales eran demasiado extensos y aparecían errores a la hora de enviarlos y almacenarlos.

Prerrequisitos (si fueran necesarios)

La conexión que probemos debe estar bien implementada en localhost. Todos los programas deben enviar y almacenar correctamente los strings. Por ejemplo el master con el gestor de colas tiene que poder enviar los mensajes y este recibirlos correctamente.

Pasos (metodología):

En un principio el servicio streaming enviaba todos los tweets en un mismo string al master. El master los separaba en bloques de 5 tweets por string y se almacenaban directamente en la cola, pero como esto daba problemas, decidimos envíar los tweets uno por uno, dentro del gestor estos se volvían a unir cuando llegaban 5 tweets en sus respectivos bloques para almacenarse en la cola(cada bloque son 5 tweets) y se separaban de nuevo a la hora de enviarlo a los workers. El streaming también termina enviando tweets uno por uno y en general todos los programas.

Resultado esperado:

Que los mensajes se envíen y reciban correctamente.

Resultado obtenido:

Al principio como enviábamos mensajes muy largos(un bloque de 5 tweets o los 25 tweets en un mismo string), estos no llegaban correctamente y daba error, por lo que decidimos enviar los mensajes uno a uno, es decir, un tweet por cada send. De este modo los mensajes no exceden cierto tamaño y se envían y reciben correctamente.

Conexión Streaming y Gestor de colas	Responsable	Todo el equipo
	Fecha	3/01/2022

Descripción:

Al ejecutar el master y los workers con el resto de programas dan el error Segmentation Fault.

Prerrequisitos (si fueran necesarios)

Todos los programas con los canales correctos que compilen adecuadamente.

Pasos (metodología):

Primero lanzamos todos los procesos y localizamos el error en master y workers.

Cambiamos la forma de medir la longitud de un string de sizeof a una variable con un valor concreto.

Resultado esperado:

Que los programas se conecten correctamente.

Resultado obtenido:

Las conexiones entre Streaming-Master-Gestor de colas-Workers se realizan correctamente.