Memoria de P4

XDR: filtros de conversión (Parte II)

Shunya Zhan

28 de abril del 2025

ÍNDICE

Introducción

En la actualidad, la gestión eficiente de datos meteorológicos es crucial para diversas aplicaciones, desde la predicción del clima hasta la investigación científica y la planificación agrícola. Con el avance de la tecnología, se ha vuelto posible implementar sistemas distribuidos que permiten la recopilación y análisis de datos meteorológicos de manera más efectiva y precisa. Este proyecto se centra en la implementación de un sistema meteorológico distribuido utilizando RPC (Remote Procedure Call), una técnica que facilita la comunicación entre diferentes componentes de un sistema distribuido.

El sistema desarrollado en este proyecto permite registrar estaciones meteorológicas, consultar datos de sensores (temperatura, humedad y presión), calcular medias globales y eliminar estaciones registradas. La arquitectura cliente-servidor utilizada en este proyecto asegura que las operaciones se realicen de manera eficiente y que los datos se gestionan de forma centralizada.

Desarrollo

Arquitectura del Sistema

El sistema meteorológico distribuido está compuesto por tres componentes principales: el servidor, el cliente y la interfaz RPC. Cada uno de estos componentes desempeña un papel crucial en la gestión y consulta de datos meteorológicos.

1. **Servidor (m_server.c):**
    - **Gestión de Estaciones Meteorológicas**: El servidor es responsable de gestionar las estaciones meteorológicas y sus sensores. Almacena las estaciones en una lista fija con un número máximo de estaciones configurado.
    - **Implementación de Funciones RPC:** El servidor implementa las funciones RPC necesarias para registrar estaciones, consultar datos de sensores y calcular estadísticas. Estas funciones permiten la interacción remota entre el cliente y el servidor.
    - **Almacenamiento de Datos:** Los datos de las estaciones meteorológicas y sus sensores se almacenan en estructuras de datos adecuadas, garantizando la integridad y accesibilidad de la información.
2. **Cliente (m_client.c):**
    - **Interacción con el Usuario:** El cliente permite al usuario interactuar con el servidor mediante comandos específicos. Estos comandos incluyen registrar estaciones, consultar datos de sensores, calcular medias globales y eliminar estaciones.
    - **Implementación de Funciones de Cliente:** El cliente implementa funciones que envían solicitudes al servidor para realizar las operaciones mencionadas. Estas funciones aseguran que las solicitudes se realicen de manera correcta y eficiente.
    - **Validación de Entradas:** El cliente incluye validaciones robustas para asegurar que las entradas del usuario sean correctas y estén dentro de los rangos permitidos.
3. **Interfaz RPC (m.x):**
    - **Definición de Funciones RPC**: La interfaz RPC define las funciones RPC y las estructuras de datos utilizadas por el cliente y el servidor. Esto incluye la especificación de los tipos de datos y los parámetros necesarios para cada función.

- **Generación Automática de Archivos:** La interfaz RPC genera automáticamente los archivos necesarios para la comunicación RPC, facilitando la implementación y el mantenimiento del sistema.

Funciones Principales

El sistema incluye varias funciones principales que permiten la gestión y consulta de datos meteorológicos:

Servidor:

- setestacion_1_svc: Registra una nueva estación meteorológica.
- getmediasensor_estacion_1_svc: Calcula la media de un sensor específico de una estación.
- getsensor_estacion_1_svc: Devuelve todas las muestras de un sensor específico de una estación.
- getmediasallestaciones_1_svc: Calcula las medias globales de todos los sensores de todas las estaciones.
- darbaja_1_svc: Elimina todas las estaciones registradas.

Cliente:

- registrar_estacion: Solicita al servidor registrar una nueva estación.
- obtener_media_sensor: Solicita al servidor la media de un sensor específico.
- obtener_sensor: Solicita al servidor todas las muestras de un sensor específico.
- obtener_medias_globales: Solicita al servidor las medias globales de todos los sensores.
- dar_baja: Solicita al servidor eliminar todas las estaciones.

Memoria

La gestión de memoria es un aspecto crucial en el desarrollo de sistemas distribuidos, especialmente cuando se manejan datos dinámicos como los de las estaciones meteorológicas y sus sensores. A continuación, se describen los principales problemas de memoria encontrados durante el desarrollo del proyecto y las soluciones implementadas para resolverlos.

Fallos/Errores/Problemas Encontrados y su Solución

1. **Error: "Segmentation fault" al registrar una estación**
    - Causa: Este error se produjo debido al uso incorrecto de punteros y la falta de inicialización de estructuras. Cuando se intentaba registrar una nueva estación, los punteros no inicializados causaban accesos inválidos a la memoria.
    - Solución: Para resolver este problema, se inicializan correctamente las estructuras antes de utilizarlas y se validaron los punteros para asegurarse de que apuntarán a ubicaciones válidas en la memoria. Esto incluyó la asignación de memoria para las estructuras y la verificación de que los punteros no fueran nulos antes de acceder a ellos.
2. **Error: Tipo de sensor inválido**
    - Causa: El cliente permitía introducir valores fuera del rango permitido para el tipo de sensor, lo que podía causar accesos inválidos a la memoria o resultados incorrectos.
    - Solución: Se agregó una validación en el cliente para asegurarse de que el tipo de sensor esté dentro del rango permitido (0 para Temperatura, 1 para Humedad y 2 para Presión). Esta validación previene la introducción de valores incorrectos y asegura que los datos sean consistentes.
3. **Error: Comando mediasglobales o baja muestra ayuda en lugar de ejecutarse**
    - Causa: La validación incorrecta de argumentos en el cliente causaba que estos comandos no se ejecutarán correctamente y mostraran la ayuda en lugar de realizar la operación solicitada.
    - Solución: Se corrigió la validación de argumentos en el cliente para que estos comandos no requieren argumentos adicionales. Esto asegura que las funciones se ejecuten correctamente y que los datos se gestionan adecuadamente.
4. **Error: Datos incorrectos al consultar sensores**
    - Causa: La falta de inicialización de las estructuras de consulta causaba que los datos retornados fueran incorrectos o inconsistentes.

- Solución: Se inicializan correctamente las estructuras de consulta antes de enviarlas al servidor. Esto asegura que los datos sean consistentes y que las consultas se realicen correctamente.

Compilación y ejecución

Compilación: Abre una terminal y levantar el docker, e introducir en terminal siguiente comando para compilar Makefile (generado automaticamente al ejecutar **rpcgen -a fichero.x**):

- make -f Makefile.m

```
rpcuserclient@937a38ebb92c:~/p4_new$ make -f Makefile.m
cc -g -I/usr/include/tirpc   -c -o m_clnt.o m_clnt.c
cc -g -I/usr/include/tirpc   -c -o m_client.o m_client.c
cc -g -I/usr/include/tirpc   -c -o m_xdr.o m_xdr.c
cc -g -I/usr/include/tirpc    -o m_client  m_clnt.o m_client.o m_xdr.o -lnsl -ltirpc
cc -g -I/usr/include/tirpc    -o m_server  m_svc.o m_server.o m_xdr.o -lnsl -ltirpc
```

Nota Importante: antes de ejecutar el Makefile debe asegurar que el fichero Makefile.m haya estas dos líneas:

CFLAGS += -g -I/usr/include/tirpc

LDLIBS += -lnsl -ltirpc

Y los códigos principal de servidor y cliente está proporcionado junto con el fichero zip.

Ejecución: En la terminal, ejecuta el servidor con el siguiente comando:

- ./servidor
- ./cliente <servidor> <comando> [argumentos...]

Prueba y resultado

Registrar

```
rpcuserclient@937a38ebb92c:~/p4_new$ ./m_client localhost registrar SUR
Conexión al servidor localhost establecida correctamente.
Registrando estación: SUR
(DENTRO) Registrando estación: SUR
Nombre de estación: SUR
Estación 'SUR' registrada correctamente.
Estación registrada correctamente.
```

```
rpcuserclient@937a38ebb92c:~/p4_new$ ./m_server
LLAMANDO A REGISTRO
[OK-1] REGISTRANDO: SUR
[OK-1] Estación registrada correctamente
```

Media

```
rpcuserclient@937a38ebb92c:~/p4_new$ ./m_client localhost media SUR 1
Conexión al servidor localhost establecida correctamente.
Obteniendo media del sensor 1 en la estación SUR
Media del sensor 1 de la estación 'SUR': 2.50
Media del sensor obtenida correctamente.
```

```
[OK-2] Accediendo al sensor 1 de la estación SUR.
[OK-2] Resultado obtenido: 2.50
```

Sensor

```
rpcuserclient@937a38ebb92c:~/p4_new$ ./m_client localhost sensor SUR 0
Conexión al servidor localhost establecida correctamente.
Obteniendo valores del sensor 0 en la estación SUR
Valores del sensor 0 de la estación 'SUR':
  Muestra 1: 10.00
  Muestra 2: 6.00
Valores del sensor obtenidos correctamente.
rpcuserclient@937a38ebb92c:~/p4_new$ ./m_client localhost sensor SUR 1
Conexión al servidor localhost establecida correctamente.
Obteniendo valores del sensor 1 en la estación SUR
Valores del sensor 1 de la estación 'SUR':
  Muestra 1: 1.00
  Muestra 2: 4.00
Valores del sensor obtenidos correctamente.
rpcuserclient@937a38ebb92c:~/p4_new$ ./m_client localhost sensor SUR 2
Conexión al servidor localhost establecida correctamente.
Obteniendo valores del sensor 2 en la estación SUR
Valores del sensor 2 de la estación 'SUR':
  Muestra 1: 6.00
  Muestra 2: 3.00
Valores del sensor obtenidos correctamente.
```

```
[OK-3] Accediendo al sensor 0 de la estación SUR.
[OK-3] Copiados 2 elementos del sensor 0 en el resultado.
[OK-3] Accediendo al sensor 1 de la estación SUR.
[OK-3] Copiados 2 elementos del sensor 1 en el resultado.
[OK-3] Accediendo al sensor 2 de la estación SUR.
[OK-3] Copiados 2 elementos del sensor 2 en el resultado.
```

Medias Globales

```
rpcuserclient@937a38ebb92c:~/p4_new$ ./m_client localhost mediasglobales
Conexión al servidor localhost establecida correctamente.
Obteniendo medias globales de todas las estaciones.
Medias globales de todas las estaciones:
  Temperatura: 8.00
  Humedad: 2.50
  Presión: 4.50
Medias globales obtenidas correctamente.
```

```
[OK-4] Medias calculadas: Temperatura=8.00, Humedad=2.50, Presión=4.50
```

Baja

```
rpcuserclient@937a38ebb92c:~/p4_new$ ./m_client localhost baja
Conexión al servidor localhost establecida correctamente.
Dando de baja todas las estaciones.
Todas las estaciones han sido eliminadas.
Todas las estaciones han sido eliminadas correctamente.
```

```
[OK-5] Dando de baja el servicio
[OK-5] Servicio dado de baja correctamente
```

```
rpcuserclient@937a38ebb92c:~/p4_new$ ./m_client localhost sensor SUR 1
Conexión al servidor localhost establecida correctamente.
Obteniendo valores del sensor 1 en la estación SUR
Valores del sensor 1 de la estación 'SUR':
No se encontraron muestras.
```

```
[ERROR-3] No se ha encontrado la estación deseada
```

Conclusión

El sistema meteorológico distribuido basado en RPC se implementó con éxito, cumpliendo con los requisitos funcionales y proporcionando una solución robusta para la gestión de datos meteorológicos. A lo largo del desarrollo del proyecto, se abordaron y resolvieron varios desafíos técnicos, incluyendo la validación de entradas, el manejo de memoria y la comunicación cliente-servidor.

1. **Implementación Eficiente:** Se logró implementar un sistema distribuido que permite registrar estaciones meteorológicas, consultar datos de sensores, calcular medias globales y eliminar estaciones registradas. La arquitectura cliente-servidor utilizada asegura que las operaciones se realicen de manera eficiente y que los datos se gestionan de forma centralizada.
2. **Validaciones Robustas:** Se implementaron validaciones robustas para asegurar la integridad de los datos y prevenir errores comunes. Esto incluye la validación de entradas del usuario y la verificación de punteros antes de su uso.
3. **Manejo de Memoria:** Se abordaron problemas de manejo de memoria, como la inicialización de estructuras y la liberación de memoria asignada dinámicamente. Estas soluciones previenen fugas de memoria y aseguran que los recursos se gestionen de manera eficiente.
4. **Pruebas Exhaustivas:** Se realizaron pruebas exhaustivas para verificar la funcionalidad del sistema en diferentes escenarios. Estas pruebas confirmaron que el sistema funciona correctamente y que los datos se gestionan de manera adecuada.

El sistema desarrollado proporciona una base sólida para futuros desarrollos en el ámbito de la meteorología y la gestión de datos distribuidos. Su diseño modular y extensible permite agregar nuevas funcionalidades y mejorar las existentes. Además, la implementación de RPC demuestra su eficacia para la comunicación entre componentes de un sistema distribuido, lo que puede ser aplicado en otros proyectos y áreas de investigación.

Anexo

- Código fuente
- Campus Virtual
- Copilot