UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

Laboratorio 1 - MPI

Integrante: Esteban Arenas
Bryan Salgado

Asignatura: Sistemas Distribuidos

28 de abril de 2024

Índice de Contenidos

1. Introducción.....2
2. Diseños de solución.....2
3. Metodología.....3
4. Resultados y discusión.....3
5. Conclusiones.....3

1. Introducción

El presente laboratorio se enfoca en tres objetivos principales: comprender y utilizar herramientas de comunicación para cómputo paralelo, diseñar e implementar soluciones distribuidas para el procesamiento de datos, y comparar múltiples estilos arquitecturales en el contexto de un desafío de cómputo paralelo.

El desafío conocido como "The One Billion Row Challenge " ha sido propuesto a la comunidad de usuarios de Github, inicialmente como una evaluación de la eficacia del código Java moderno. Sin embargo, este desafío ha trascendido las fronteras del lenguaje, generando resultados en diversos entornos como Rust, C++, Go, entre otros. El reto consiste en procesar grandes volúmenes de datos, específicamente registros de mediciones de temperatura provenientes de estaciones meteorológicas, con el objetivo de calcular la temperatura mínima, máxima y promedio para cada estación.

Para nuestro caso de estudio, se propone la implementación y comparación de tres soluciones con distintos enfoques arquitecturales: monolítico (no distribuido), basado en servicios y basado en eventos. Cada solución será evaluada a lo largo de este informe detallando los diseños de solución, la metodología utilizada para obtener los resultados, los resultados obtenidos junto con un análisis preliminar y las conclusiones derivadas de la experiencia.

2. Diseños de solución

El proyecto considera tres enfoques arquitectónicos diferentes: monolítico, basado en servicios y basado en eventos. Cada una de estas arquitecturas tiene sus propias características y ventajas, a continuación se detalla cada solución.

- Monolítico: En esta solución, todo el programa se ejecuta dentro de un solo proceso, aunque podría haber múltiples hilos de ejecución, se restringen a sólo uno, de esta forma todos los componentes interactúan directamente dentro de un mismo proceso.
- Microservicios: Para este enfoque la solución se divide en múltiples servicios independientes unos de otros. Se puede decir que cada servicio vive en su propio proceso y se comunican de manera independiente.
- Eventos: Esta última arquitectura permite que servicios o procesos emitan eventos, los cuales son procesados y enrutados por un middleware. De esta manera los componentes se comunican a través de estos eventos en lugar de llamadas directas.

El desarrollo de este proyecto se realizará utilizando el lenguaje de programación Python. Además se utilizará la librería MPI (Interfaz de Paso de Mensajes), especificación estándar para la comunicación entre procesos en sistemas de memoria distribuida, para el manejo de múltiples procesos.

3. METODOLOGÍA

En esta sección se explorarán las diferentes soluciones diseñadas que emplean un enfoque arquitectónico específico con el fin de abordar la creación de una solución y se analizará cómo la ejecución de los algoritmos se encaminan en la construcción de un resultado correcto.

Cabe mencionar que, debido a la naturaleza del laboratorio, existirán métodos entre las diferentes arquitecturas que tendrán una lógica similar y solamente varían en la adaptación que estas requieren al tener que emplear un enfoque de programación distinto. Estos métodos están relacionados con la lectura de un archivo que contiene los datos necesarios para lograr obtener un resultado, la manipulación de los datos apoyándose en la lógica para poder crear la solución y, por último, hacer entrega de los resultados y que estos sean guardados en un nuevo archivo.

La aplicación monolítica es manejada mediante un bloque principal que contiene todos los llamados a las funciones para la construcción de solución, comenzando por la lectura de un archivo de entrada. Los archivos pueden variar en la dimensión de datos que contenga y son leídos línea a línea guardando la información que está separada por comas, los valores corresponden a una estación y la temperatura de la misma.

El siguiente paso en ejecución es el uso de los datos obtenidos anteriormente para poder encontrar para cada estación su temperatura mínima, temperatura máxima y el promedio de estas. Una vez obtenidos estos resultados se realiza el último llamado la función para guardar todos estos datos dentro de un archivo de salida manteniendo así mantener la persistencia de los datos.

La solución que es dividida en servicios, sigue una línea de ejecución a la descrita recién, la diferencia radica en la implementación de la librería MPI. Los métodos que son los mismos de la arquitectura monolítica, ahora pueden ser llamados servicios y cada uno podrá ser ejecutado en un proceso distinto a los demás servicios.

Esta tecnología es brindada por MPI, permitió que en el bloque principal sea posible diseñar una lógica para manejar que diferentes servicios sean ejecutados en diferentes procesos. De esta manera, el método de lectura se puede encontrar funcionando en el proceso 3, mientras que el cálculo de las temperaturas se da en el proceso 2 y el guardado de los resultados estaría terminando el proceso 1.

Por último, la arquitectura basada en eventos añade dificultad al código, la lectura de archivo anteriormente se daba con la lectura línea por línea y una vez todos los datos fuesen guardados recién se terminaba su ejecución, ahora la lectura de una línea emite un evento.

Cuando es leída una fila del archivo, se creará un evento y su destino será el cálculo de las temperaturas para ese dato envíado, esto se dará repetidamente hasta terminar la lectura completa del archivo. Las temperaturas serán calculadas para todas las estaciones, luego los resultados serán enviados uno por uno creando un nuevo evento en el envío de estos datos.

Esto lleva al último paso que es guardar los datos en un archivo de salida, la cual se dará luego de haber recibido todos los resultados envíados y no antes de ello. Destacar que este enfoque no deja de lado la idea de manejar estos eventos viviendo en procesos diferentes, dado que también utiliza la librería MPI y es con esta misma que se envío de los eventos. Y de esta forma es como se logró implementar todos los enfoques para el mismo problema y obtener el mismo resultado en los tres casos.

Hasta ahora no ha sido mencionado, pero cada solución posee su propia interfaz gráfica las cuales tampoco varían en su diseño. Estas poseen un botón para poder iniciar la ejecución del programa para luego poder visualizar los resultados.

4. Resultados y discusión

Para el desarrollo de las pruebas de rendimiento en relación al tiempo de ejecución de los programas basados en las 3 arquitecturas, se trabajó en torno a la cantidad de filas de cada archivo. Con el fin de encontrar una relación lineal al procesamiento de datos, se realizaron medidas con una cantidad de filas muy reducida, una mínima de 20, hasta una máxima de 10.000.000 filas.

Cabe señalar que la máquina que se utilizó para testear las distintas arquitecturas poseía las siguientes características: Sistema operativo Windows 10, Procesador de i5-10300 H (4 núcleos) y 8 GB de RAM.

Monolítica	
Filas	Tiempo
20	0.0069926
30	0.0040308
50	0.0037777
100	0.0010102
500	0.0019945
1000	0.0039849
5000	0.010027
10000	0.025
50000	0.1160046
100000	0.2060275
500000	1
1000000	2,0480704
10000000	22

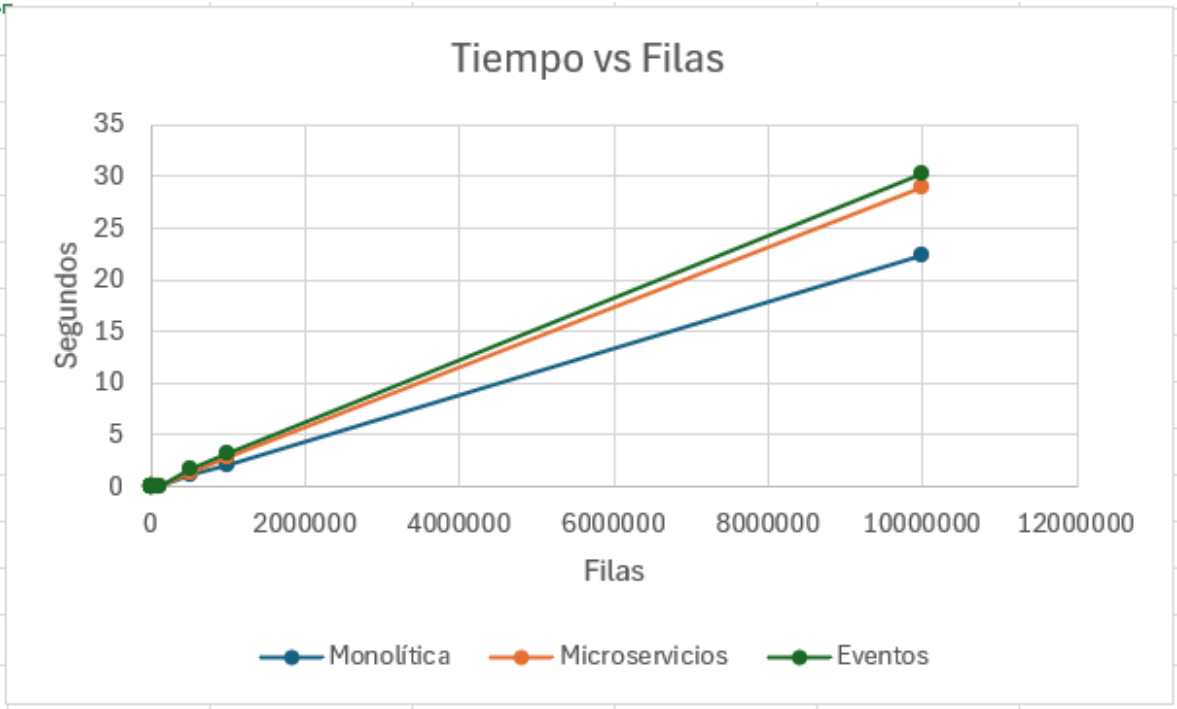
Microservicios	
Filas	Tiempo
20	0.0020051
30	0.0010002
50	0.0010002
100	0.0009987
500	0.0019983
1000	0.003002
5000	0.013
10000	0.0289997
50000	0.1419972
100000	0.2790322
500000	1
1000000	3
10000000	29

Eventos	
Filas	Tiempo
20	0.0009998
30	0.0010004
50	0.0010014
100	0.000997
500	0.0019687
1000	0.0040025
5000	0.013
10000	0.030
50000	0.1899986
100000	0.3380005
500000	2
1000000	3
10000000	30

Las tablas obtenidas resumen el tiempo observado de ejecución. Estas nos permiten notar que los archivos de pocas filas tienen subidas y bajadas que no parecieran reflejar linealidad, pero al ver los resultados en aquellas pruebas con más filas, se hace evidente la relación lineal.

Esta peculiaridad debe darse por la naturaleza de los archivos, ya que un factor importante en el procesamiento de las ciudades y sus temperaturas, es la cantidad de distintas ciudades que puede tener una archivo, puesto que el programa tendrá que gastar más tiempo indexando la ciudad encontrada para comparar sus valores de temperatura obtenidos.

En cambio, aquellos archivos de más de miles de filas, mantuvieron su número de ciudades distintas constantes y sólo aumentaron en observaciones de temperatura.



Al comparar los resultados obtenidos por las 3 distintas arquitecturas, se puede ver que la Monolítica obtuvo el mejor rendimiento en tiempo, mientras que la peor fue la de Eventos.

Aún así, también se puede concluir que se lograron resultados muy parecidos, postura que se puede respaldar en que a pesar de usarse arquitecturas distintas, por la forma de abordar la lectura y análisis de datos, no se pudieron separar tan bien los procesos para aprovechar que los servicios o eventos, en relación a la Monolítica, se daban de forma parcialmente independiente.

5. Conclusiones

A través del diseño, creación y realización de pruebas, se ha observado que la arquitectura monolítica proporciona el mejor rendimiento en términos de tiempo de ejecución, a pesar de ello puede que no sea la más escalable. Las arquitecturas basadas en servicios y eventos, aunque hayan tenido un rendimiento ligeramente inferior, se podría inferir que escalaría mejor con conjuntos de datos más grandes en un ambiente real debido a su capacidad de ejecutar procesos en paralelo.

Por otro lado, se ha logrado dar con los objetivos propuestos: comprender y utilizar herramientas de comunicación para cómputo paralelo, diseñar e implementar soluciones distribuidas para el procesamiento de datos, y comparar múltiples estilos arquitecturales en el contexto del desafío planteado.