Università di Catania

Elaborato di Fine corso

Corso di Distributed Systems & Big Data

Svolto da

Leone Damiano – 1000029422

Spadaro Sapari Lorenzo – 1000038030

Sommario

Abstract

Il progetto è composto da una pipeline che acquisisce metriche da Node Exporter tramite Prometheus, esegue test e calcoli statistici per ricavare informazioni circa stagionalità, trend, ciclicità delle metriche oltre a prevedere anche Max, Min e valore medio per i prossimi 10 min.

Eseguendo questi test statistici e calcoli sulle metriche in tempo reale, la pipeline consente un'analisi più dettagliata e accurata delle metriche, che può essere utilizzata per il monitoraggio, la risoluzione dei problemi e la previsione delle prestazioni. In seguito, i dati elaborati vengono inviati ad un altro servizio tramite un broker Kafka.

In questo nuovo servizio i dati ricevuti sono salvati in un database MySQL, in modo da archiviare le metriche in modo persistente, rendendo possibile l'accesso ai dati storici per ulteriori analisi.

Sono stati realizzati altri 2 microservizi: uno di data retrieval, che tramite interfaccia gRPC-based, in modo asincrono, permette di estrarre le informazioni presenti nel database; ed un altro, basato su REST API, che permette di definire un SLA Set di 5 metriche - definito come coppia SLI, SLO – che tramite l'utilizzo di un apposito client, rende possibile effettuare chiamate HTTP tramite metodo POST e restituire le eventuali violazioni nelle ultime 1,3,12 ore e le possibili violazioni per i prossimi 10 minuti.

Questo progetto è stato realizzato con lo scopo di soddisfare i requisiti di un sistema di monitoraggio ed alerting con funzionalità di analisi e archiviazione dei dati in tempo reale, oltre a fornire un modo semplice per accedere ai dati passati delle metriche e impostare le politiche SLA.

Ciascuno dei 4 microservizi descritti e il database MySQL sono eseguiti ciascuno nel proprio container Docker per poter eventualmente essere in grado di scalare ciascun componente in modo indipendente.

Introduzione

Questo progetto è realizzato tramite una *data pipeline* che acquisisce le metriche dall'exporter *Node Exporter* di Prometheus, un toolkit di monitoraggio, esegue test statistici e calcoli su tali metriche; quindi, invia i dati elaborati a un broker Kafka.

Apache Kafka è una piattaforma di streaming di messaggi distribuita, fault-tolerant e high-throughput. Viene utilizzata per creare pipeline di dati in tempo reale. Consente l'archiviazione e l'elaborazione di grandi quantità di dati, in modo *fault-tolerant* e scalabile. Si basa su un modello broker-based, publish-subscriber, in cui i *producers* scrivono su determinati *topics* e i consumatori leggono da tali topic sottoscrivendosi, senza avere un legame diretto tra publisher e subscriber.

I dati vengono quindi letti dal broker da un nuovo microservizio e salvati in un database MySQL.

Inoltre, nel progetto sono inclusi altri due microservizi, uno che consente di recuperare i dati prodotti dalla pipeline, e presenti nel database, tramite un'interfaccia gRPC e l'altro che consente la creazione di un set di SLA di cinque metriche e monitora le metriche specifiche, informando su possibili violazioni nelle ore passate ed in futuro, facendo uso di REST API. Ogni microservizio viene eseguito nel proprio *container* grazie all'utilizzo di *Docker*, fornendo una soluzione indipendente dalla piattaforma e facilmente distribuibile.

Docker è una piattaforma per lo sviluppo, lo *shipping* e l'esecuzione di applicazioni. Essa consente di separare le applicazioni in running dall'infrastruttura sottostante in modo da poter distribuire il software rapidamente. Tramite Docker si ha la possibilità di impacchettare ed eseguire ciascun microservizio in un ambiente isolato chiamato *container*.

L'isolamento e la sicurezza consentono di eseguire più container contemporaneamente su un determinato host. Infatti, essi sono leggeri e contengono tutto il necessario per eseguire l'applicazione.

Lo scopo di questo progetto è realizzare un sistema di monitoraggio in grado di fornire analisi e archiviazione in tempo reale delle metriche ricavate da Prometheus, consentendo un monitoraggio e una risoluzione dei problemi più dettagliati e accurati del sistema. Eseguendo test statistici e calcoli sulle varie metriche prese in considerazione, l'applicazione può fornire preziose informazioni sulle prestazioni del sistema, come l'identificazione dei trend e la

previsione delle prestazioni future. Inoltre, viene fornito un modo per archiviare le metriche in modo persistente, rendendo possibile l'accesso ai dati storici per ulteriori analisi.

Gli altri due microservizi che sono stati implementati nel progetto sono progettati per fornire un facile accesso ai dati cronologici e per impostare criteri di accordo sul livello di servizio (SLA). Il primo microservizio consente di recuperare i dati dal database tramite un'interfaccia gRPC, facilitando l'accesso ai dati passati per ulteriori analisi. Il secondo microservizio consente la creazione di un SLA set di 5 metriche e informa su possibili violazioni nelle ultime 1,3,12 ore e nei 10 minuti successivi all’osservazione, utilizzando le REST API, consentendo il monitoraggio dinamico e l'avviso di possibili violazioni dell’SLA.

In sintesi, il progetto mira a fornire l'analisi e l'archiviazione dei dati in tempo reale delle metriche prese da Prometheus, nonché un facile accesso ai dati storici e l'impostazione delle policies del SLA attraverso le interfacce gRPC. Ogni microservizio viene eseguito nel proprio contenitore utilizzando Docker, fornendo una soluzione indipendente dalla piattaforma e facilmente distribuibile, nonché scalabile.

Scelte architetturali

Descrizione dei microservizi

Il progetto è composto da diversi microservizi, ognuno dei quali è responsabile di un'attività specifica nella pipeline. Di seguito è riportata una panoramica di alto livello dei microservizi e delle relative interazioni.

Data pipeline: questo microservizio legge le metriche da Prometheus attraverso l'uso di Node Exporter, esegue test statistici e calcoli sulle metriche, come il test di Hodrick-Prescott, il test di Dickey-Fuller, il calcolo dell'autocorrelazione, e di vari valori significativi delle metriche nelle ultime 1,3,12 ore, in più fornisce la previsione delle prestazioni future in termini di valori max, min, e medi. Quindi invia questi metadati e le previsioni ad un broker Kafka sul topic "prometheusdata".

Data storage: questo microservizio ha il compito di leggere i dati dal broker Kafka e per ogni messaggio letto, salvarne i valori ottenuti un database MySQL.

Data Retrieval: questo microservizio consente di recuperare i dati dal database tramite un'interfaccia gRPC, permettendo l'accesso ai dati cronologici per ulteriori analisi. Per ogni metrica permette di estrarre i metadati relativi alla metrica stessa, i valori max, min, media e deviazione standard per le ultime 1,3,12 ore e le previsioni su tali valori.

SLA Manager: questo microservizio consente la creazione di un SLA set composto da 5 metriche e informa su possibili violazioni nelle ultime 1,3,12 ore e possibili violazioni nei 10 minuti successivi, interfacciandosi tramite REST API, consentendo quindi il monitoraggio e l'avviso di possibili violazioni dell'SLA.

Ciascuno di questi microservizi viene eseguito nel proprio container, che consente una facile scalabilità e distribuzione. I contenitori comunicano tra loro tramite l'uso di un broker di messaggi (Kafka) e interfacce gRPC.

L'uso di un broker di messaggi consente il disaccoppiamento dei microservizi, consentendo loro di scalare in modo indipendente. L'uso delle interfacce gRPC consente una comunicazione efficiente tra i microservizi, poiché si tratta di un framework open source ad alte prestazioni per la creazione di API RPC (Remote Procedure Call), che mira a rendere indipendenti client e server dal linguaggio di programmazione usato, grazie all'impiego di un protofile, che permette la serializzazione e deserializzazione dei messaggi scambiati.

Schema di Comunicazione dei Microservizi

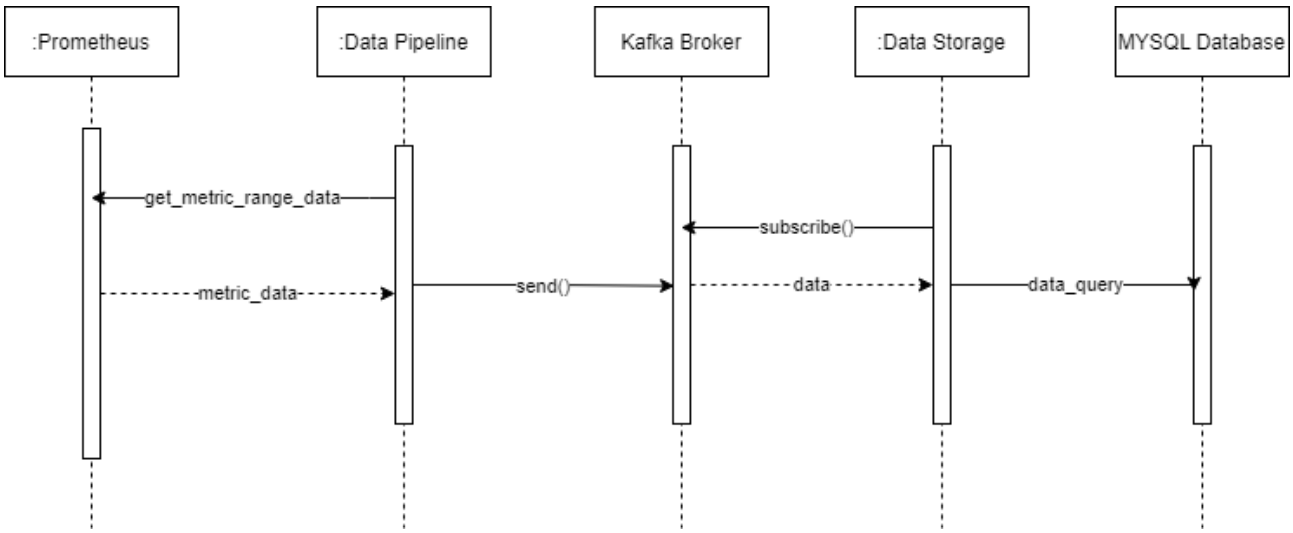

Grafico 1

Il grafico precedente illustra le primitive tramite cui i microservizi Data Pipeline e Data Storage comunicano con Prometheus, con il broker Kafka e con il Database MySQL in cui sono salvati i metadati delle metriche.

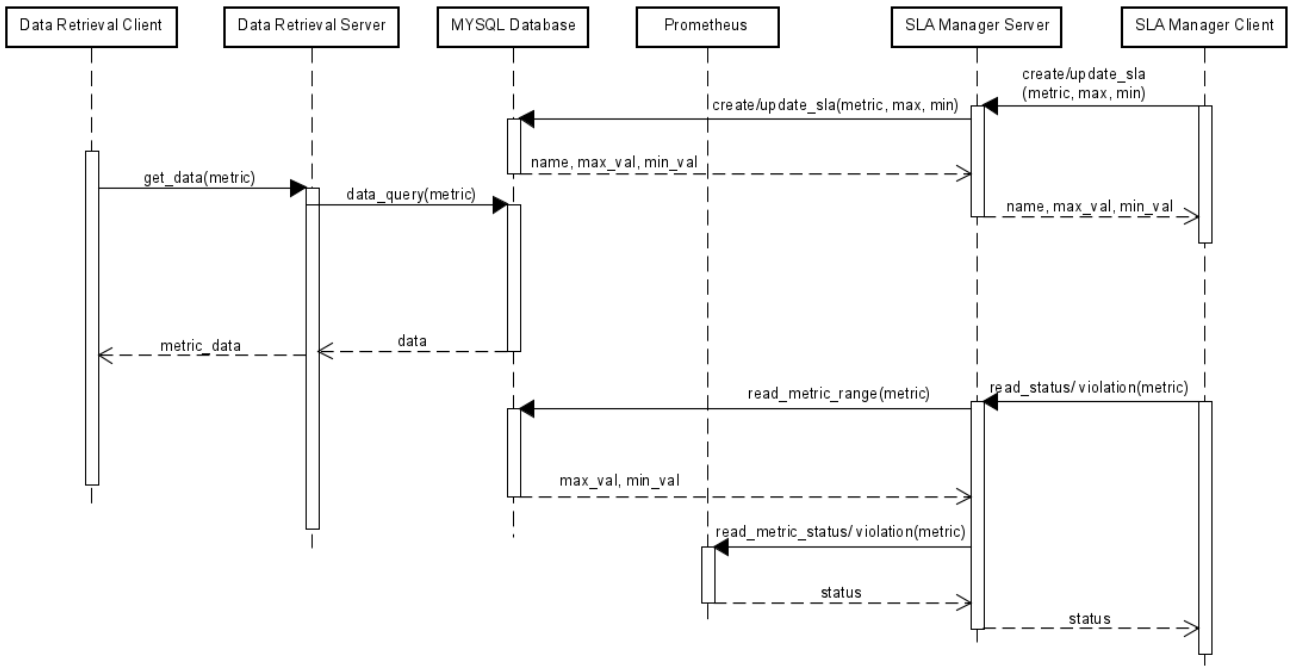

Grafico 2

Il grafico precedente illustra le primitive tramite cui i microservizi Data Retrieval e SLA Manager comunicano con Prometheus, e con il Database MySQL.

Api Implementate

Di seguito verranno elencate le API implementate nel progetto, con una breve descrizione delle loro funzionalità.

gRPC (generalized Remote Procedure Calls):

L'interfaccia **gRPC** (**g**eneralized **R**emote **P**rocedure **C**alls), rappresenta un framework open source ad alte prestazioni per la creazione di API RPC. Consente la creazione di microservizi altamente performanti ed efficienti che possono comunicare tra loro utilizzando un protocollo binario anziché HTTP/JSON.

Le gRPC rappresentano un'estensione del concetto di chiamata a procedura locale modellandolo nei sistemi distribuiti, ciò può avvenire in quanto il client possiede l'interfaccia dei metodi, mentre il server fornisce l'implementazione di essi.

Esse mirano a rendere indipendente client e server dal linguaggio di programmazione usato, grazie all'impiego di un protofile, che permette la serializzazione e deserializzazione dei messaggi scambiati.

Le API che implementano questa interfaccia sono utilizzate principalmente nel microservizio di Data Retrieval e permettono l'interazione con il database MySQL in cui sono contenute le informazioni sulle metriche.

REST API:

REST (o, nella sua forma completa, **Re**presentational **S**tate **T**ransfer) è un tipo di architettura che costituisce un vero e proprio standard per la creazione di Web API.

Il termine REST è largamente usato per descrivere ogni tipo di interfaccia capace di trasmettere dati per mezzo del protocollo HTTP, senza l'ausilio di tecnologie ausiliari come cookie o protocolli vari per lo scambio di messaggi.

Ciò ha reso possibile la creazione di comunicazioni client-server completamente *stateless* (dove quindi ogni richiesta è completamente svincolata dalle altre).

Una comunicazione di questo tipo è utilizzata nel microservizio SLA Manager che permette di creare e controllare lo stato di un SLA.

Informazioni per Build & Deploy

- Aprire un terminale e posizionarsi nella cartella contenente il file **docker-compose.yml.**

- Eseguire il comando **docker compose up -d** per avviare la costruzione e l'avvio dei vari container in cui sono eseguiti i microservizi.

- Il database verrà configurato automaticamente all'avvio, ma avendo bisogno di un certo periodo di tempo per la configurazione, non sarà possibile per i container **data storage** e **server data retrieval** connettersi e quindi saranno stoppati automaticamente. Basterà semplicemente riavviarli.

- Per far sì che si avvii il client del servizio di data retrieval, è necessario che il file **asynch_client.py** venga eseguito tramite CLI, trattandosi di un programma che richiede input da tastiera. È necessario quindi eseguire il comando **docker exec -it <id_container> python3 app/asynch_client.py**.

- Aspettare affinché il database si riempia.

- Adesso è possibile effettuare le richieste tramite il container del **client data retrieval,** che tramite gRPC chiederà al server i dati, che a sua volta contatterà il database MySQL e restituirà la risposta al **client data retrieval**.

- Collegarsi al server apache relativo al container **SLA manager** sull'indirizzo localhost alla porta 80, ed iniziare ad effettuare chiamate POST, tramite un client HTTP (e.g POSTMAN), iniziare con **localhost/create.php** in modo da definire l'SLA Set relativo alle 5 metriche da analizzare, e riempiere la tabella sla_manager del database. Specificare come body della richiesta il nome della metrica con il relativo range di max value e min value (È presente qualche esempio in formato json nella directory: sla_manager/SLI.json).