Progetto di Sistemi Distribuiti e Pervasivi

Laboratorio di Sistemi Distribuiti e Pervasivi

Maggio 2018

1 Descrizione del progetto

Lo scopo del progetto è quello di realizzare un sistema per la raccolta e l'analisi di dati di sensori installati in una *smart city*. I dati dei sensori (che sono distribuiti in vari punti della città) necessitano di essere collezionati e aggregati per poter effettuare delle analisi utili a migliorare la città stessa. Il sistema di raccolta e analisi dei dati è basato su un'architettura di *edge computing*[1]. Per la città vengono distribuiti dei *nodi edge*, che possono essere visti come dei gateway che raccolgono e analizzano i dati di sensori di una particolare zona geografica della città. I *nodi edge* necessitano di coordinarsi periodicamente per comunicare ad un *server cloud* alcune statistiche sullo stato della città. Il sistema deve permettere a degli *analisti* di interrogare il *server cloud* per ottenere informazioni sullo stato della città. Infine, ogni *nodo edge* è dotato di un pannello che periodicamente mostra statistiche della zona monitorata e dello stato globale della città.

1.1 Architettura del sistema

L'architettura del sistema da sviluppare è mostrata in Figura 1. Le misurazioni dei sensori vengono prodotte da un particolare processo chiamato *Simulatore di sensori*, che simula un arbitrario numero di sensori. Ogni sensore comunica con il *nodo edge* più vicino geograficamente. Ogni *nodo edge* è un processo che riceve stream di misurazioni da diversi sensori a lui vicini geograficamente. Queste misurazioni vengono analizzate in tempo reale per generare statistiche locali della zona coperta dal *nodo edge*. I *nodi edge* devono coordinarsi tra di loro per eleggere un coordinatore. Il *nodo edge* coordinatore ha quindi come ulteriore compito quello di ricevere le statistiche locali calcolate dai vari *nodi edge*. Periodicamente, il coordinatore effettua un'ulteriore aggregazione di queste statistiche locali producendo statistiche globali della città, che vengono trasmesse al *server cloud*. Il *server cloud* si occupa di ricevere i dati aggregati dal coordinatore, di salvarli internamente, e di rendere disponibile agli *analisti* dei servizi per ottenere statistiche

[1] `https://en.wikipedia.org/wiki/Edge_computing`

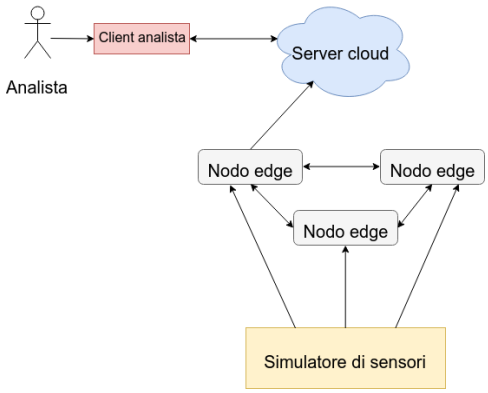

Figura 1: Architettura

sui dati raccolti. Gli *analisti* possono accedere a queste informazioni usando un'apposita applicazione chiamata *client analista*. Nelle seguenti sezioni verranno descritte nel dettaglio tutte le componenti dell'architettura appena presentata.

2 Server cloud

Il *server cloud* si occupa di mantenere una rappresentazione interna della città e di ricevere dal coordinatore statistiche che verranno poi interrogate dagli analisti. Deve quindi offrire due diverse interfacce REST per: a) gestire la rete dei nodi *edge* e ricevere le statistiche, b) permettere agli analisti di effettuare interrogazioni. In aggiunta, è necessaria un'interfaccia REST per permettere ad ogni sensore di ottenere l'indirizzo del *nodo edge* più vicino geograficamente (se esiste).

2.1 Rappresentazione interna della città

La città viene rappresentata come una griglia 100x100. I sensori possono quindi essere posizionati arbitrariamente all'interno di ogni cella. Una singola cella può contenere più sensori. È possibile invece installare un *nodo edge* in una cella solo se non esiste un altro *nodo edge* nella griglia tale per cui la distanza sia minore di 20. La distanza tra due punti (x_1, y_1) e (x_2, y_2) della griglia si misura con la seguente formula:

$$d(x_1, y_1, x_2, y_2) = |x_1 - x_2| + |y_1 - y_2|$$

Il *server cloud* mantiene internamente solo le posizioni dei *nodi edge* e non quelle dei sensori.

2.2 Interfaccia per i nodi edge

2.2.1 Inserimento

Quando vuole inserirsi nel sistema, un *nodo edge* deve comunicare al *server cloud*:

- Identificatore
- Indirizzo IP
- Numero di porta sul quale è disponibile per ricevere misurazioni di sensore
- Numero di porta sul quale è disponibile per comunicare con gli altri *nodi edge*
- Una posizione sulla griglia

È possibile inserire un *nodo edge* nella griglia solo se non esiste un altro *nodo edge* con lo stesso identificatore e se la posizione fornita è valida. Se l'inserimento va a buon fine, il *server cloud* restituisce al *nodo edge* la lista di *nodi edge* presenti nella griglia, specificando per ognuno indirizzo IP e numero di porta per la comunicazione.

2.2.2 Rimozione

Un *nodo edge* può richiedere esplicitamente di rimuoversi dal sistema. In questo caso, il *server cloud* deve semplicemente rimuoverlo dalla griglia.

2.2.3 Statistiche

Il *server cloud* deve predisporre un'interfaccia per ricevere dal coordinatore le statistiche riguardanti la città. Il coordinatore trasmette statistiche sia globali (relative a tutta alla città) che locali (di ogni zona coperta da un *nodo edge*). È sufficiente memorizzare queste statistiche in una struttura dati che permetta successivamente l'analisi.

2.3 Interfaccia per gli analisti

Il *server cloud* deve fornire dei metodi per ottenere le seguenti informazioni:

- Lo stato attuale della città (posizione dei vari *nodi edge* nella griglia)
- Ultime n statistiche (con timestamp) prodotte da uno specifico *nodo edge*
- Ultime n statistiche (con timestamp) globali e locali della città

- Deviazione standard e media delle ultime n statistiche prodotte da uno specifico *nodo edge*
- Deviazione standard e media delle ultime n statistiche globali della città

2.4 Interfaccia per i sensori

Il *server cloud* deve predisporre un'interfaccia per fornire indirizzo IP e numero di porta del *nodo edge* più vicino al sensore che fa richiesta. Il sensore deve quindi comunicare le proprie coordinate per ricevere in output il *nodo edge* più vicino al quale comunicare.

3 Nodo edge

Ogni *nodo edge* è un **processo** che si occupa di ricevere le misurazioni dei sensori e di coordinarsi con gli altri nodi per inviare statistiche al *server cloud*.

3.1 Inizializzazione

Un *nodo edge* deve essere inizializzato specificando il suo identificatore, il numero di porta di ascolto per la comunicazione tra *nodi edge*, il numero di porta di ascolto per la ricezione di misurazioni dai sensori e l'indirizzo del *server cloud*. Una volta avviato, il *nodo edge* deve generare casualmente una posizione e tentare di inserirsi nella griglia comunicando con il *server cloud*. Se la registrazione non va a buon fine per colpa di una posizione non ammissibile, il *nodo edge* deve provare a generarne altre casuali finchè non viene accettata dal sistema. Dopo 10 tentativi il *nodo edge* si arrende terminando. Se l'inserimento va a buon fine, il *nodo edge* analizza la lista di *nodi edge* che riceve in risposta dal *server cloud*. Se non ci sono altri *nodi edge* nella griglia, il *nodo edge* si auto-proclama coordinatore. Altrimenti, deve presentarsi agl altri *nodi edge* della rete e capire quale nodo sia il coordinatore.

3.2 Chiusura esplicita

Si assume che ogni *nodo edge* possa terminare solamente in maniera controllata. Quando viene richiesta la terminazione di un *nodo edge*, questo deve comunicare solamente al *server cloud* la sua intenzione di uscire dalla griglia, chiudere tutte le comunicazioni con sensori e *nodi edge* ed infine terminare.

3.3 Gestione dei dati dei sensori

Ogni *nodo edge* riceve stream di dati da diversi sensori. È necessario aggregare questi stream sfruttando la tecnica della *sliding window* presentata durante le lezioni di teoria, considerando un overlap del 50%. I dati dei diversi stream devono quindi essere inseriti in un buffer comune, ed ogni 40 misurazioni deve essere effettuata la media dei dati nel buffer. Ogni volta che viene calcolata una media, il *nodo edge* la trasmette al coordinatore assieme all'istante di tempo in cui è calcolata (timestamp). Di risposta, il coordinatore trasmette la statistica globale più recente della città. Il coordinatore calcola le statistiche globali ogni 5 secondi considerando la media delle medie ricevute. Ogni volta che viene calcolata una statistica globale, il coordinatore la invia al *server cloud* insieme alle statistiche locali di ogni *nodo edge* e ai timestamp in cui le statistiche sono state calcolate.

3.4 Elezione del coordinatore

Quando un coordinatore termina la sua esecuzione, gli altri *nodi edge* devono rilevare automaticamente la sua scomparsa per eleggere un nuovo coordinatore. È quindi necessario implementare un meccanismo di elezione del coordinatore, scegliendo tra gli algoritmi visti a lezione di teoria: o l'algoritmo di Bully o l'algoritmo ad anello.

3.5 Pannello

Ogni *nodo edge* ha un pannello che va simulato con stampe su standard output. Il coordinatore deve visualizzare le statistiche globali e le statistiche locali di ogni *nodo edge* più recenti (se disponibili). Ogni altro *nodo edge* deve visualizzare le proprie statistiche locali e le statistiche globali ricevute dal coordinatore.

4 Simulatore di sensori

Il *simulatore di sensori* è un processo che si occupa di simulare un certo numero di sensori. Per semplicità consideriamo solo una tipologia di sensore, ovvero i **sensori di inquinamento**. Questi sensori periodicamente misurano il livello di polveri sottili nell'aria (PM10) in $\mu g/m^3$.

4.1 Inizializzazione

L'applicazione *Simulatore di sensori* ha bisogno di due parametri per essere avviata: il numero di sensori da installare e l'indirizzo del *server cloud*. Una volta avviata, l'applicazione deve occuparsi semplicemente di lanciare il numero di simulatori specificato da parametro. Ogni sensore in fase di inizializzazione deve generare una posizione casuale sulla griglia della città.

Dopodichè, comunica al *server cloud* la sua posizione per ottenere il *nodo edge* più vicino (se presente).

4.2 Misurazioni di sensori e comunicazione

Ogni sensore produce periodicamente delle misurazioni. Ogni singola misurazione di sensore è caratterizzata da:

- Valore letto
- Timestamp in millisecondi

La generazione di queste misurazioni viene svolta da opportuni simulatori, il cui codice viene fornito per semplificare lo sviluppo del progetto. Ogni simulatore assegna come timestamp alle misurazioni il numero di secondi passati dalla mezzanotte.
Al link *http://ewserver.di.unimi.it/sdp/simulation_src_2018.zip* è quindi possibile trovare il codice necessario. Questo codice andrà aggiunto come package al progetto e NON deve essere modificato. Ogni *simulatore di sensori* dovrà quindi occuparsi di far partire i simulatori necessari per generare misurazioni di sensori. Ogni simulatore è un thread che consiste in un loop infinito che simula periodicamente (con frequenza predefinita) le misurazioni, comunicandole ad un particolare *nodo edge*. Viene fornita solo l'interfaccia di comunicazione (SensorStream), la quale espone un metodo:

$$void\ sendMeasurement(Measurement\ m)$$

È dunque necessario creare una classe che implementi l'interfaccia. Ogni sensore avrà un suo stream.
Il thread di simulazione usa il metodo *sendMeasurement* per inviare una singola misurazione allo stream.

La rete dei *nodi edge* è dinamica, e il *nodo edge* più vicino ad un sensore può cambiare con il tempo. Ogni sensore deve quindi interrogare ogni 10 secondi il *server cloud* per capire quale sia il *nodo edge* più vicino in modo da inviare a lui le proprie misurazioni. Se un sensore si accorge che non riesce più a comunicare con un *nodo edge*, può anticipare la richiesta al *server cloud*. Se non sono presenti *nodi edge* nella griglia, le misurazioni del sensore vengono comunque prodotte ma vengono perse.

5 Applicazione analista

L'*Applicazione analista* è un'interfaccia a linea di comando che si occupa di interagire con l'interfaccia REST precedentemente presentata in Sezione 2.3. L'applicazione deve quindi semplicemente mostrare all'analista un menu per

scegliere uno dei servizi di analisi offerto dal server, con la possibilità di inserire eventuali parametri che sono richiesti.

6 Semplificazioni e limitazioni

Si ricorda che lo scopo del progetto è dimostrare la capacità di progettare e realizzare un'applicazione distribuita e pervasiva. Pertanto gli aspetti non riguardanti il protocollo di comunicazione, la concorrenza e la gestione dei dati di sensori sono considerati secondari. Inoltre è possibile assumere che :

- nessun nodo si comporti in maniera maliziosa,
- nessun processo termini in maniera incontrollata

Si gestiscano invece i possibili errori di inserimento dati da parte dell'utente. Inoltre, il codice deve essere robusto: tutte le possibili eccezioni devono essere gestite correttamente.

Sebbene le librerie di Java forniscano molteplici classi per la gestione di situazioni di concorrenza, per fini didattici gli studenti sono invitati a fare **esclusivamente uso di metodi e di classi spiegati durante il corso di laboratorio**. Pertanto, eventuali strutture dati di sincronizzazione necessarie (come ad esempio lock, semafori o buffer condivisi) dovranno essere implementate da zero e saranno discusse durante la presentazione del progetto.

Nonostante alcune problematiche di sincronizzazione possano essere risolte tramite l'implementazione di server iterativi, per fini didattici si richiede di utilizzare server multithread. Inoltre, per le comunicazioni via socket, è richiesto di usare formati standard (ad esempio JSON, XML o Protocol Buffer) per lo scambio di dati. Qualora fossero previste comunicazioni in broadcast, queste devono essere effettuate in parallelo e non sequenzialmente. Alternativamente alle socket, è permesso utilizzare il framework *grpc* per la comunicazione tra processi. Lo studente può scegliere liberamente quale delle due tecnologie utilizzare (socket, *grpc* o entrambe). In fase di presentazione del progetto, lo studente è comunque tenuto a conoscere almeno a livello teorico entrambe le tecnologie.

7 Presentazione del progetto

Il progetto è da svolgere individualmente. Durante la valutazione del progetto verrà richiesto di mostrare alcune parti del programma, verrà verificata la padronanza del codice presentato, verrà verificato il corretto funzionamento del programma e verranno inoltre poste alcune domande di carattere teorico

inerenti gli argomenti trattati nel corso (parte di laboratorio). E' necessario presentare il progetto sul proprio computer.

Il codice sorgente dovrà essere consegnato al docente prima della discussione del progetto. Per la consegna, è sufficiente archiviare il codice in un file zip, rinominato con il proprio numero di matricola (es. 760936.zip) ed effettuare l'upload dello stesso tramite il sito http://upload.di.unimi.it . Sarà possibile effettuare la consegna a partire da una settimana prima della data di ogni appello. La consegna deve essere tassativamente effettuata entro le 23:59 del secondo giorno precedente quello della discussione (es. esame il 13 mattina, consegna entro le 23.59 dell'11).

Si invitano inoltre gli studenti ad utilizzare, durante la presentazione, le istruzioni `Thread.sleep()` al fine di mostrare la correttezza della sincronizzazione del proprio programma. Si consiglia vivamente di analizzare con attenzione tutte le problematiche di sincronizzazione e di rappresentare lo schema di comunicazione fra le componenti. Questo schema deve rappresentare il formato dei messaggi e la sequenza delle comunicazioni che avvengono tra le componenti in occasione delle varie operazioni che possono essere svolte. Tale schema sarà di grande aiuto, in fase di presentazione del progetto, per verificare la correttezza della parte di sincronizzazione distribuita.

Nel caso in cui il docente si accorga che una parte significativa del progetto consegnato non è stata sviluppata dallo studente titolare dello stesso, lo studente in causa: a) dovrà superare nuovamente tutte le prove che compongono l'esame del corso. In altre parole, se l'esame è composto di più parti (teoria e laboratorio), lo studente dovrà sostenere nuovamente tutte le prove in forma di esame orale (se la prova di teoria fosse già stata superata, questa verrà annullata e lo studente in merito dovrà risostenerla). b) non potrà sostenere nessuna delle prove per i 2 appelli successivi. Esempio: supponiamo che gli appelli siano a Febbraio, Giugno, Luglio e Settembre, e che lo studente venga riconosciuto a copiare all'appello di Febbraio. Lo studente non potrà presentarsi agli appelli di Giugno e Luglio, ma potrà sostenere nuovamente le prove dall'appello di Settembre in poi. Il docente si riserva la possibilità di assegnare allo studente in causa lo svolgimento di una parte integrativa o di un nuovo progetto.

8 Parti facoltative

8.1 Prima parte

Con l'aumentare del numero di *nodi edge*, il coordinatore diventa un collo di bottiglia. Si richiede quindi di implementare un'architettura di raccolta dati alternativa, dove i *nodi edge* sono organizzati come un albero. La radice dell'albero è il nodo edge che si occupa di comunicare le misurazioni aggregate al *server cloud*. Le foglie dell'albero sono i *nodi edge* che ricevono direttamente i dati dai sensori. I nodi intermedi dell'albero sono *nodi ed-*

ge che effettuano aggregazioni intermedie da comunicare al proprio padre. Sviluppare lato *server cloud* un algoritmo per costruire l'albero dinamicamente. Quando un *nodo edge* chiede al *server cloud* di entrare nella griglia, gli viene comunicato il *nodo edge* padre. In caso di uscita di un *nodo edge* dalla rete, ci sono tre casi. Se il *nodo edge* che è uscito è una foglia, non ci sono problemi. Se il *nodo edge* che è uscito è un nodo intermedio, i suoi figli devono chiedere al *server cloud* chi è il nuovo padre. Se il *nodo edge* che è uscito è la radice, i *nodi edge* devono eleggere una nuova radice e comunicare il risultato dell'elezione al *server cloud*, che dovrà quindi ri-costruire l'albero. Alla fine del processo di elezione ogni *nodo edge* deve chiedere al *server cloud* chi è il suo nuovo padre.

8.2 Seconda parte

Si estenda la prima parte facoltativa in modo tale che la costruzione dinamica dell'albero sia effettuata autonomamente dai *nodi edge* e che non sia guidata in alcun modo dal *server cloud*.

9 Aggiornamenti

Qualora fosse necessario, il testo dell'attuale progetto verrà aggiornato al fine di renderne più semplice l'interpretazione. Le nuove versioni del progetto verranno pubblicate sul sito del corso. Si consiglia agli studenti di controllare regolarmente il sito.

Al fine di incentivare la presentazione del progetto nei primi appelli disponibili, lo svolgimento della parte facoltativa 1 diventa obbligatoria a partire dall'appello di Settembre (incluso), mentre entrambe le parti facoltative (1 e 2) saranno obbligatorie negli appelli successivi.