Relazione ISS seconda parte

Repository github con i progetti della prima parte: https://github.com/Lorenzo-Zoccadelli/iss_lab_2025.git

Obiettivi generali

Il principale obiettivo della seconda parte del corso è stato quello di approfondire come si possa analizzare e progettare un sistema distribuito basato su attori e microservizi. Come caso di studio principale si è scelto di modificare l'applicazione "Conway life" realizzata nella prima parte del corso in modo che ogni cella potesse essere posta su un nodo fisico distinto e potesse interagire con le altre. In questo modo ogni cella è un'entità indipendente su cui gira un microservizio che cooperando realizza l'applicazione in modo distribuito.

L'obiettivo è quello di far girare ogni cella sul una raspberry pi, in modo da poter approfondire anche gli aspetti dei sistemi IoT e di come le entità possano interagire anche con il modo esterno.

Problemi dei sistemi distribuiti

I sistemi distribuiti offrono molti vantaggi come la scalabilità, la replicabilità dei nodi (che permette di realizzare in qualche modo la tolleranza ai guasti) e una maggiore dinamicità, ma richiedono spesso un'attenta progettazione dal punto di vista delle interazioni tra gli attori che li compongono e del coordinamento tra essi. Inoltre, nel caso di conway è stato necessario anche gestire la memorizzazione dello stato in modo distribuito.

In generale si voleva che ogni cella fosse indipendente e per calcolare il proprio stato futuro comunicasse con i propri vicini. Ma ciò fa sorgere immediatamente alcuni problemi:

- Come ogni cella può sapere qual è la sua posizione sulla griglia nel momento in cui entra a far parte del sistema
- Come ogni cella può sapere chi sono i suoi vicini
- Come ogni cella può sapere quando effettuare il calcolo del proprio stato futuro e quando poterlo effettivamente attuare

A questi problemi è possibile far fronte attraverso una soluzione coreografata, in cui ogni nodo sa quello che deve fare individualmente e ciò gli basta per potersi coordinare in modo corretto con gli altri (solitamente più costosa e complessa, ma più dinamica e flessibile), o orchestrata, in cui ogni entità è indipendente ma esiste comunque un'entità centrale che funge da "orchestratore" e fornisce servizi a ogni nodo semplificando la coordinazione complessiva del sistema.

In questo caso è stata scelta la versione orchestrata, in cui l'entità centrale permetteva a ogni nuova cella di ottenere una posizione corretta sulla griglia e dava il tempo per l'aggiornamento dello stato di ogni nodo. Ogni nodo, anche grazie all'utilizzo di Mqtt poteva comunicare direttamente con i propri vicini (calcolati a partire dalla propria posizione) senza necessariamente conoscere la loro posizione fisica.

Analisi e modellazione di un sistema distribuito

Caratteristica fondamentale di un sistema distribuito è la costante e intensa comunicazione tra i vari attori che lo compongono. Per questo motivo l'analisi di un problema di questo tipo non può prescindere dall'analisi delle interazioni che avvengono tra i vari attori e dei tipi di messaggi che si devono scambiare.

Ogni comunicazione deve avere una propria semantica che permette di definire in modo chiaro e non ambiguo come gli attori interagiranno in caso si scambino quei determinati messaggi. Per fare ciò è di estrema utilità affiancare l'analisi a parole con un "modello eseguibile" che permette in tempi brevi di mappare i requisiti in un semplice prototipo. Il linguaggio qak utilizzato in questa parte del corso serviva proprio a questo. Esso introduce vari concetti di alto livello (come quello di contesto, attore, messaggi di vario tipo) che permette di realizzare velocemente un prototipo di sistema funzionante concentrandosi sulla semantica e sul modello che si vuole realizzare, tralasciando i dettagli tecnici e implementativi. Inoltre, la modellazione degli attori come automi a stati finiti permette di renderne chiaro e semplice il comportamento e di modellare le entità del sistema esprimendo in modo lineare il loro comportamento

DSL e differenze tra linguaggio e libreria

Il linguaggio qak utilizzato è un DSL (domain specific language), ovvero un linguaggio realizzato per lo scopo specifico di progettare sistemi distribuiti basati su attori. Ciò è molto potente perché permette di mettere in risalto attraverso keyword di linguaggio concetti più astratti e di fornire l'abstraction gap necessario per modellare in maniera più chiara, comoda e veloce un sistema specifico.

La vera differenza tra linguaggio e libreria è che una libreria può arricchire il linguaggio per cui è stata implementata solamente di semantica, fornendo la possibilità di compiere azioni di alto livello attraverso funzioni, oggetti e metodi, mentre un linguaggio (DSL in questo caso) può associare la nuova semantica e le nuove funzionalità a una nuova sintassi. Ciò permette di vedere le cose a livello più alto e fornisce un livello di astrazione ed espressività maggiore.

Raspberry pi e IoT

Il concetto di sistema distribuito e della sua modellazione è di fondamentale importanza per un sistema basato su dispositivi IoT. Per realizzarne uno è stato scelto di utilizzare delle raspberry pi. Per quanto riguarda il progetto di conway distribuito l'idea era quella che la raspberry potesse comandare un semplice led che mostrasse lo stato della cella associata.

Si è inoltre affrontato il problema di un robot che possa muoversi e avere la percezione dell'ambiente intorno a esso attraverso un sonar. In questo caso si dota il dispositivo di un modo per ottenere un input dall'ambiente esterno.

Queste due caratteristiche si possono gestire attraverso sensori e attuatori che possono essere comandati attraverso codice (è stato utilizzato python) e il flusso dei dati può essere modellato in maniera semplice ed efficace attraverso gli attori e il linguaggio qak