UNIVERSITÀ DEGLI STUDI DI
CASSINO E DEL LAZIO MERIDIONALE

DIPARTIMENTO DI INGEGNERIA ELETTRICA
E DELL'INFORMAZIONE "MAURIZIO SCARANO"

CORSO DI LAUREA IN INGEGNERIA INFORMATICA
E DELLE TELECOMUNICAZIONI

Assessment delle performance di Elixir nell'ambito IOT.

Relatore:
Prof. Ciro D'Elia

Candidato:
Nico Fiorini

ANNO ACCADEMICO 2022/2023

Questa è una dedica

La perfezione non è il nostro obbiettivo ma la nostra tendenza
Omar Palermo

Abstract

L' industria del software si trova a fronteggiare la necessità di sviluppare software sempre più scalabili e performanti per fronteggiare l'aumento degli utenti e di servizi che ne fanno utilizzo. In questo contesto, Elixir, un linguaggio di programmazione funzionale e concorrente basato su Erlang, emerge come una scelta promettente per la costruzione di sistemi altamente affidabili e reattivi, semplificando di molto lo sviluppo di software concorrenziale.

Questo studio si propone di analizzare le caratteristiche di Elixir e le sue performance attraverso una serie di esperimenti empirici esplorando diversi aspetti delle performance mettendo in rilievo vantaggi e svantaggi nell'adottarlo.

I risultati di questa ricerca forniranno una comprensione approfondità delle capacità di Elixir in termini di prestazioni e affidabilità consentendo agli sviluppatori di fare una scelta pensata alle esigenze dei loro progetti.

Indice

1 Introduzione **1**

2 Caratteristiche di Elixir **3**
2.1 Introduzione 3

Bibliografia **4**

Capitolo 1

Introduzione

Elixir è un linguaggio di programmazione dinamico e funzionale sviluppato nel 2012 da José Valim, con l'obbiettivo di favorire una maggiore scalabilità e produttività nella macchinia virtuale di Erlang, mantenendo al contempo la compatibilità con l'ecosistema di Erlang[1]. Elixir si è affermato come una promettente scelta nell'industria del software,specialmente in contesti dove è richiesta scalabilità, tolleranza agli errori e reattività grazie al suo approccio concorrenziale.

In particolare, Elixir può risultare vantaggioso nel campo dell'IoT per diversi motivi:

1. **Concorrenza**: Nell'ambito dell'IoT, la gestione simultanea di dispositivi è essenziale. Elixir, grazie alla sua capacità di gestire facilmente la concorrenza, consente il monitoraggio e il controllo efficiente di numerosi dispositivi contemporaneamente.

2. **Fault Tolerance**: Data la natura degli ambienti IoT, dove i dispositivi possono guastarsi improvvisamente, Elixir offre strumenti per la supervisione e la gestione degli errori, garantendo la continuità delle operazioni anche in caso di fallimenti.

3. **Sviluppo Rapido e Manutenzione**: Elixir è un linguaggio moderno che offre una sintassi efficiente e snella, oltre a strumenti di sviluppo come Mix per la gestione delle dipendenze e l'ambiente interattivo iex. La presenza di un package manager (Hex)[2] e la possibilità di generare automaticamente la documentazione facilitano il processo di sviluppo e manutenzione del codice.

Il trattato esplora Elixir concentrandosi su due aspetti principali: la semplicità e le performance. Si analizzano i punti di forza di un linguaggio funzionale

e come questi sono sfruttati in Elixir, con un focus sulla concorrenza. Nella scelta di un linguaggio, la semplicità è fondamentale e deve essere accessibile a tutti i programmatori. Tuttavia, l'efficienza è altrettanto importante, quindi vengono condotti test empirici per valutare le performance di Elixir.

In particolare il lavoro effettuato è così ripartito:

- Nel capitolo 2 si discute del linguaggio funzionale, esaminando le astrazioni offerte da Elixir per lo svilluppo di codice affidabile, si tratta la concorrenza e come la Erlang VM si occupa della gestione dei processi.
- Nel capitolo 3 si spiega il lavoro sperimentale svolto e i risultati ottenuti (continuare)

Capitolo 2

Caratteristiche di Elixir

2.1 Introduzione

In questo capitolo, esamineremo le caratteristiche distintive di Elixir, un linguaggio di programmazione funzionale e concorrente che sfrutta appieno la potenza della piattaforma OTP (Open Telecom Platform).

Elixir, scritto in Erlang e eseguito sulla macchina virtuale Erlang (BEAM), eredita gli obiettivi di Erlang, ma apporta miglioramenti significativi per rendere il linguaggio più appetibile e moderno.

Erlang, nato nel 1986, è stato progettato per semplificare lo sviluppo di software concorrente e robusto. Elixir si basa su queste fondamenta solide, offrendo un'API più pulita e astrazioni avanzate che consentono ai programmatori di ragionare a un livello più elevato, facilitando la scrittura di codice concorrente in modo intuitivo.

Una delle massime principali di Erlang e, di conseguenza, di Elixir, è "Let it crash" (Lascia che si schianti), che riflette l'approccio alla gestione degli errori nei sistemi concorrenti, incoraggiando la gestione degli errori tramite il rilancio e la supervisione anziché il blocco del processo.

Per capire come lavorare con questo linguaggio, bisogna affrontare un po' di questioni e farsi un po' di domande. Bisogma capire come la macchina virtuale Beam affronta la concorrenza, Elixir in particolare è un linguaggio orientato alla concorrenza nel senso che le astrazioni che fornisce sono proprio per far sì che si programmi in modo concorrenziale portando ad avere un codice responsivo, e gestendo bene i processi anche robusto. Un altro punto da affrontare è l'immutabilità dei dati, è un concetto chiave in Elixir ed Erlang, è proprio questa caratteristica che ci semplifica la programmazione concorrenziale. Infine .....

Bibliografia

[1] *Elixir (programming language) - Wikipedia*. https://en.wikipedia.org/wiki/Elixir_(programming_language).

[2] *Hex*. https://hex.pm/.