Capitolo 1

Conclusioni

In conclusione, il nostro progetto ha conseguito pienamente l'obiettivo principale che ci eravamo prefissati. Abbiamo sviluppato un interprete prototipale attraverso il quale è possibile eseguire programmi scritti in HLCostLan, un linguaggio di programmazione appositamente progettato per l'analisi e l'ottimizzazione dei costi computazionali. Questo linguaggio offre una serie di costrutti e funzionalità che consentono agli sviluppatori di esprimere in modo chiaro e conciso il carico computazionale associato ai propri programmi, fornendo al contempo un'astrazione di alto livello che facilita la progettazione e l'implementazione del software.

Questo interprete non solo è in grado di analizzare e interpretare i programmi scritti in HLCostLan, ma offre anche la possibilità di derivare l'equazione di costo associata a tali programmi, che viene poi calcolata attraverso il tool PUBS, per descrivere in modo formale il carico computazionale generato dal programma. Questo passaggio è di fondamentale importanza poiché fornisce agli sviluppatori una chiara comprensione del carico computazionale generato dal proprio codice, consentendo loro di valutare e ottimizzare le prestazioni in modo più accurato, ed eventualmente di prendere decisioni più accurate in merito all'allocazione delle risorse.

Inoltre, il nostro interprete non si limita solo ad analizzare il linguaggio HLCostLan e a calcolare le equazioni di costo dei programmi, ma offre anche la capacità di generare automaticamente il corrispondente codice WebAssembly. Questa caratteristica rappresenta un passo significativo nella traduzione efficiente e affidabile dei programmi scritti in HLCostLan in un formato eseguibile ampiamente supportato. Tale capacità assume un'importanza critica in un'epoca in cui la computazione distribuita e la scalabilità sono sempre più cruciali. Il codice WebAssembly risultante può essere facilmente incorporato in una vasta gamma di ambienti di esecuzione, consentendo una maggiore adozione e interoperabilità del linguaggio HLCostLan.

L'approccio modulare e flessibile con cui è stato sviluppato il nostro interprete è stato pensato per agevolare ulteriori estensioni e per semplificare le operazioni di manutenzione. Questa progettazione consente agli sviluppatori di ampliare le funzionalità dell'interprete in base alle esigenze specifiche dei loro progetti, promuovendo un'evoluzione organica e sostenibile del software nel tempo. Inoltre, abbiamo reso il codice sorgente del nostro interprete liberamente accessibile su GitHub. Questo rende possibile non solo l'esplorazione e la comprensione approfondita del funzionamento interno dell'interprete, ma anche la collaborazione e la partecipazione della comunità nell'ulteriore sviluppo e miglioramento del progetto.

In definitiva, crediamo che il nostro interprete per il linguaggio HLCostLan rappresenti non solo un'importante realizzazione tecnica, ma anche una risorsa preziosa per la comunità degli sviluppatori e dei ricercatori interessati alla programmazione e all'ottimizzazione dei costi computazionali. Siamo fiduciosi che il nostro lavoro continuerà a servire da base solida per ulteriori progressi nel campo e che contribuirà a promuovere una maggiore efficienza e qualità nell'ambito dello sviluppo del software.

1.1 Sviluppi futuri

Il nostro prototipo rappresenta un passo cruciale nello sviluppo dell'architettura del sistema descritto attraverso il linguaggio dichiarativo cApp (definito in **??**). L'obiettivo principale di questo sistema è costruire un'infrastruttura che consenta di ottimizzare le risorse dei nodi worker in un sistema di calcolo distribuito. Attualmente, il nostro interprete HLCostLan costituisce un elemento fondamentale per il calcolo dei costi computazionali dei programmi, ma vi sono numerose possibilità di sviluppo future che potrebbero estendere notevolmente le funzionalità del sistema.

In particolare, il nostro interprete potrebbe essere integrato in un sistema di orchestrazione di container, come Kubernetes, per distribuire e gestire i programmi scritti in HLCostLan su un cluster di nodi worker. L'obiettivo sarebbe quello di sfruttare al meglio le risorse disponibili, garantendo un'allocazione efficiente e dinamica delle risorse in base al carico computazionale effettivo. cApp, essendo un linguaggio dichiarativo, può essere utilizzato per definire delle funzioni di selezione dei worker più adatti, tenendo conto di fattori come latenza, complessità computazionale e altri parametri rilevanti. Questo consentirebbe di massimizzare l'utilizzo delle risorse, migliorare le prestazioni complessive del sistema e ottimizzare i costi di esecuzione.

Alcuni possibili sviluppi futuri che potrebbero arricchire ulteriormente il nostro lavoro includono:

- Integrazione con Kubernetes: Sviluppare un'interfaccia per integrare il nostro interprete con Kubernetes, consentendo di distribuire e gestire i programmi scritti in HLCostLan su un cluster di nodi worker in modo efficiente e dinamico.

- Estendere il linguaggio HLCostLan: Aggiungere nuovi costrutti e funzionalità al linguaggio HLCostLan per consentire una rappresentazione più dettagliata e precisa del carico computazionale, consentendo agli sviluppatori di esprimere in modo più accurato le esigenze dei loro programmi.

- Implementazione di strategie di allocazione avanzate: Sviluppare algoritmi e strategie di allocazione delle risorse più sofisticati, che tengano conto di una gamma più ampia di fattori e metriche per garantire un utilizzo ottimale delle risorse del cluster.

- Supporto per l'ottimizzazione dinamica: Integrare il sistema con meccanismi di ottimizzazione dinamica, consentendo di adattare le risorse allocate in tempo reale in base alle condizioni del sistema e al carico di lavoro.

- Integrazione con sistemi di monitoraggio e gestione delle prestazioni: Collegare il sistema all'infrastruttura di monitoraggio delle prestazioni, consentendo di rilevare e rispondere dinamicamente ai cambiamenti nelle condizioni del sistema e nell'afflusso di lavoro.

- Valutazione dell'efficacia pratica: Condurre studi sperimentali e valutazioni empiriche per testare l'efficacia del sistema in scenari realistici di utilizzo, confrontandolo con altre soluzioni esistenti e analizzando le prestazioni e l'efficienza complessive.

Siamo convinti che esplorando queste direzioni di sviluppo futuro, potremo contribuire in modo significativo al progresso nel campo della gestione delle risorse in sistemi di calcolo distribuito, fornendo agli utenti uno strumento potente ed efficiente per ottimizzare le prestazioni dei loro sistemi e ridurre i costi operativi.