Progetto di Ingegneria Informatica 2023/24

FLOATING POINT TO FIXED POINT CONVERSION

Autore: Emanuele Caruso

Matricola: 981776

Tutor: prof. G. Agosta

#### 1.1 Obiettivo:

Lo scopo del progetto è quello di valutare le performance di un sistema che converte l'aritmetica a virgola mobile in aritmetica a virgola fissa attraverso un processo semi-automatico (TAFFO), mediante la conversione di una piccola applicazione di benchmark della suite Mantevo , nello specificola la miniAMR (versione originale consultabile su github). L'obiettivo è confrontare le prestazioni della versione a virgola fissa in confronto all’originale. La conversione da aritmetica a virgola mobile a aritmetica a virgola fissa è un aspetto cruciale nell'ottimizzazione delle prestazioni dei sistemi informatici, soprattutto in ambiti come l'elaborazione scientifica e l'high-performance computing.

#### 1.2 Specifica

Per il progetto è necessario installare <https://github.com/TAFFO-org/TAFFO>. Questo è un software preposto alla conversione dei valori in floating point to fixed point, citando gli autori, “TAFFO is an autotuning framework which tries to replace floating-point operations with fixed-point operations as much as possible.”

Mi è stato richiesto di partire dall’applicazione originale per riscriverla e riadattarla in modo tale da poter utilizzare il paradigma di TAFFO.

Tra le varie mini-app ho scelto <https://github.com/Mantevo/miniAMR> "Mini-applicazione per il Raffinamento Adattivo della Maglia" essendo affascinato dalla modellazione matematica dei sistemi fisici.

#### 1.3 Analisi Della Specifica

La trasformazione della mini-app per renderla adatta a TAFFO è molto lunga ed ha richiesto molto lavoro sul codice originale, nelle slide del PPT mostro un esempio di come sia necessario trasformare le strutture dati e il codice in generale per permettere a Taffo di poter lavorare correttamente. Tra le modifiche più sostanziali ritengo sia imporante citare: la modifica della struttura dati di partenza, il riadattamento degli algoritmi in base alla nuova struttura dati, le annotazioni di Taffo sulle variabili principali, il calcolo dei range ottimali e l’adattamento del codice per migliorare i test. Tra i file consegnati è presente lo zip con tutto il codice della miniAMR\_TAFFO nel caso si desiderino maggiori informazioni e approfondimenti sul codice vero e proprio.

Come è scritto sulla repository di git-hub,TAFFO si compone di cinque plugin LLVM, ognuno dei quali contiene un passaggio di ottimizzazione o di analisi:

1. **TaffoInitializer:** Questo passaggio si occupa dell'inizializzazione del framework.
2. **TaffoVRA (Value Range Analysis):** Questo passaggio esegue un'analisi del range dei valori per determinare le possibili conversioni da floating point a fixed point.
3. **TaffoDTA (Data Type Allocation):** Questo passaggio si occupa dell'allocazione dei tipi di dati e delle conversioni di tipo necessarie.
4. **LLVMFloatToFixed:** Questo passaggio effettua la conversione effettiva da floating point a fixed point.
5. **LLVMErrorPropagator:** Questo passaggio si occupa della propagazione degli errori e dell'aggiornamento del Feedback Estimator.

#### 2.1 Sviluppo del progetto

In questa sezione, vorrei riassumere le modifiche più sostanziali apportate alla versione originale della mini-app per adattarla a Taffo il che ha richiesto alcune modifiche significative all’architettura del programma.

L’obiettivo ultimo di Taffo richiede che L'utente debba annotare il codice sorgente e fornire informazioni sull'intervallo dinamico dei valori delle variabili di input.

Le mini-applicazioni della suite Mantevo includono librerie di codice utilizzate per eseguire determinate funzioni o operazioni cardine delle applicazioni complete per facilitarne la verifica della correttezza di esse o gli output previsti da certi input.

Come riportato dal paper della suite Mantevo <https://mantevo.github.io/pdfs/MantevoOverview.pdf>, le mni-app portano vari benifici nelle seguenti situazioni:

 Nuovi linguaggi e modelli di programmazione: Le mini applicazioni possono essere rifattorizzate o completamente riscritte in nuovi linguaggi e modelli di programmazione.

 Ottimizzazione del compilatore: Le mini applicazioni forniscono un ambiente focalizzato per i sviluppatori di compilatori per migliorare il codice compilato.

Questi ultimi due punti sono i concetti chiave di questo progetto, che sfrutta TAFFO per riaddatare l’applicazione secondo il principio dell’architettura FIXED POINT.

La mini-AMR si concentra su un dominio computazionale a forma di cubo unitario, suddiviso in blocchi. I file.c costituiscono le diverse parti del programma e contengono le varie funzioni necessarie per eseguire le operazioni di comunicazione, raffinamento, calcolo degli stencil, bilanciamento del carico e altro ancora. Ogni blocco ha lo stesso numero di celle in ogni direzione e comunica valori "fantasma" con i blocchi vicini. Con la raffinazione adattativa della griglia, i blocchi possono rappresentare diversi livelli di dettaglio nella griglia più ampia. . I calcoli sulle variabili in ogni cella consistono in una media dei valori nello stencil scelto (si veda.stencil.c). Il processo di raffinamento e di raffinamento dei blocchi è guidato dagli oggetti che vengono spinti attraverso la griglia. Se un blocco interseca la superficie o il volume di un oggetto, quel blocco può essere raffinato.

Inizialmente la struttura dati che rappresenta il blocco, è stata definita come una struct. Tuttavia, per come è strutturato internamente Taffo, la gestione delle struct risulta problematica perciò la prima operazione è quella di riadattare la struttura dati estraendo l’array (di double) che memorizza i valori dei blocchi, allocando, successivamente, la struttura direttamente nel main. Nello specifico riporto la nuova struttura dati:

**Immagine che contiene testo, schermata, software

Descrizione generata automaticamente**

Questa soluzione mi ha permesso di annotare la variabile principale secondo le linee guida del Software.

Naturalmente, sono moltissime altre le variabili che sono state annotate all’interno del codice per permettere a TAFFO di lavorare nel modo più ottimizzato possibile.

Chiaramente la soluzione ha comportato un necessario cambiamento degli algoritmi core dell’applicazione che hanno il compito di performare la raffinazione della maglia adattiva. Per chiarezza, di seguito allego un esempio della nuova versione:

Immagine che contiene testo, schermata, Carattere, software

Descrizione generata automaticamente

Un’altra questione cruciale per il funzionamento di Taffo è quella di scegliere il valore ottimale del limite inferiore e superiore di ogni variabile annotata.

Prendendo sempre in considerazione l’esempio del “block”, i valori sono stati settati tra 0.0 e 1.0, seguendo le specifiche della precedente versione dell’applicazione, generati in maniera casuale per permettermi di avere dei test più generalizzati e controllati.

Immagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

In conclusione, vorrei mostrare il vantaggio dell’utilizzo di Taffo. Cruciale nel programma sono le funzioni che eseguono lo stencil. Lo stencil è una tecnica utilizzata nell'ambito della computazione numerica per calcolare valori approssimati in punti di una griglia tramite la combinazione pesata dei valori dei punti vicini.

Di fondamentale importanza è quindi avere dei valori più precisi possibili, e per questo attraverso Taffo, la versione della miniAMR “nuova” presenta più accuratezza.

VERSIONE FIXED POINT:

Immagine che contiene testo, schermata

Descrizione generata automaticamente

#### 3.1 Test eseguiti

I test eseguiti hanno due obiettivi principali.

Il primo è quello per misurare i tempi di esecuzione dell’applicazione per metterla a confronto con la versione originale.

Nonostante le FPU (floating-point-units) siano molto efficienti e vengano sollecitate molto all’interno della mini-app , il tempo di esecuzione totale su 100 esecuzioni delle due versioni sono praticamente uguali!

Taffo version:

Immagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamente

Original Version:

Immagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamente

Il secondo obiettivo è quello di misurare l’errore assoluto: EA=∣ValoreTaffo−Valorefloat∣

Per fare ciò è necessario uno script Python che prenda come argomenti un possibile esempio di valori richiesti dall’applicazione(se si desiderano altri valori di esempio, si possono cambiare facilmente dagli args del file main.py, si raccomanda di seguire le linee guida del READ.me dell’applicazione originale)di seguito viene riportato il grafico che mostra i valori dell’errore ed il suo andamento.

Immagine che contiene testo, linea, Diagramma, Carattere

Descrizione generata automaticamente

#### 4.1 Strumenti utilizzati

#### Durante lo sviluppo del progetto, sono stati utilizzati diversi strumenti e software, tra cui il framework TAFFO, linguaggi di programmazione come C (per la scrittura della miniapp) e Python(debugging e considerazioni finali di testing su output), e strumenti per la visualizzazione e l'analisi dei dati e misurazione delle tempistiche.

#### 5.1 Conclusioni

Partecipare a questo progetto è stato estremamente gratificante, soprattutto perché mi ha offerto l'opportunità di collaborare direttamente con il team di ricercatori e sviluppatori di Taffo. Senza il loro sostegno, sarebbe stato estremamente difficile, considerando che era la mia prima esperienza con questo software. Inoltre, mi ha permesso di essere coinvolto nell'individuazione dei bug e nello sviluppo, aspetti cruciali del processo. Inoltre, l'argomento del progetto si integra perfettamente con i miei interessi accademici, in particolare con il mio percorso futuro di laurea magistrale in High Performance Computing.