Relazione finale progetto Struct - Omar Moh'd (n° matricola: 510448)

URL del repository github: https://github.com/omarmohd/struct.jl

Introduzione

Struct è una sezione della libreria Julia "LAR.jl". Quest'ultima viene utilizzata per eseguire calcoli geometrici su complessi cellulari espressi attraverso la Rappresentazione Algebrica Lineare(LAR).

L'obiettivo di Struct è quello di rappresentare oggetti complessi e descriverli nel loro rispettivo sistema di coordinate, il che risulta particolarmente comodo per specificarne i vertici.

Gli oggetti complessi sono rappresentati mediante modelli chiamati proprio "Struct", trattati e intesi, anche nel codice, come grafi orientati aciclici. Queste sono strutture gerarchiche formate da vari componenti che vivono in diversi sistemi di riferimento.

Obiettivi

Gli obiettivi perseguiti nella realizzazione di questo progetto sono vari.
Innanzitutto è stata focalizzata l'attenzione ad una quanto più possibile comprensione del package Struct e degli argomenti che più ne sono affini, nonostante questi siano temi che non sono mai stati trattati in altri corsi accademici. Comprensione che poteva incrementare anche grazie allo studio degli esempi contenuti nel package LinearAlgebraicRepresentation. Dopo una prima comprensione del funzionamento complessivo di Struct, l'obiettivo diventava quello di andare ad investigare il funzionamento di ciascuna singola funzione che lo componeva, al fine di individuare quelle parti che potevano essere in qualche modo modificate. Per quanto riguarda proprio l'ottimizzazione del codice, a partire dallo studio del libro *"Julia High Performance - Second Edition - Optimizations, distributed computing, multithreading, and GPU programming with Julia 1.0 and beyond by Avik Sengupta"* applicare i principi imparati durante le lezioni e dal trattato era il metodo migliore di agire.

Risultati

I risultati ottenuti riguardano la maggior parte degli obiettivi raggiunti. Nello specifico questi sono stati ottenuti mediante una sequenza ben definita di processi. Inizialmente lo studio del package è stato il focus principale, ovviamente per mettere mano al codice bisogna prima capire il funzionamento di esso. Dopo aver installato tutti gli ambienti di lavoro necessari all'esecuzione del codice ed dopo aver quindi eseguito i test relativi agli esempi per verificarne il comportamento, è stato realizzato il grafo delle dipendenze delle funzioni con i parametri di ingresso e di uscita, che è stato utile per capire il comportamento delle chiamate delle funzioni. Successivamente si è passato al vero e proprio studio delle singole funzioni pre-esistenti: è stato descritto il compito o i compiti di ogni funzione, interpretandone il significato, e ogni funzione che eseguiva più compiti elementari, è stata suddivisa in funzioni sequenziali mono-task più piccole seguendo ciò che è descritto nel capitolo 3 "Kernel methods and function barriers" del libro *Julia High Performance*.
Successivamente, utilizzando i notebook IJulia, sono state utilizzate le macro @code_*(@code_typed, @code_llvm, @code_warntype), @benchmark e @btime per misurare i comportamenti delle vecchie e nuove funzioni, riguardo i tempi di esecuzione(minimi, medi e massimi) e la memoria allocata. Per migliorare l'efficienza del codice sono stati utilizzati diversi principi studiati nel trattato precedentemente citato. Alcuni esempi riguardano lo studio delle dichiarazioni dei tipi, la rimozione dei keyword arguments, la verifica di come gli array erano stati implementati, l'utilizzo di array statici dove opportuno o possibile, oppure ancora l'utlizzo di macro quali @inbounds(per la rimozione dei bound checking), @fastmath(per velocizzare le operazioni matematiche), @views o @simd. Nonostante non siano state effettuate articolate operazioni di parallelizzazione del codice(date le ampie difficoltà riscontrate nel corso del progetto, dovute a mancate nozioni relative alla modellazione in 3 dimensioni con le strutture LAR) le nuove misurazioni effettuate a valle delle modifiche e riportate all'interno dei rispettivi notebook, consentono di osservare i miglioramenti in termini di tempo e allocazione di memoria.
Infine la creazione dei test elementari per le funzioni appena create, è stata utile anche per verificare le corrette implementazioni delle modifche. Ulteriori test sono stati eseguiti creando istanze apposite e verificandone il funzionamento con il package ViewerGL.
Come detto, anche se le ottimizzazioni effettuate riguardano una moderata parte di tutte quelle applicabili, gli obiettivi principali sono stati raggiunti, ovvero quelli della comprensione del funzionamento del package Struct e di come modellare complessi cellulari espressi attraverso la Rappresentazione Algebrica Lineare(LAR) per l'esecuzione di calcoli geometrici.