Linee Guida per l'Esame - Modulo 2

Corso di Sistemi di Elaborazione Accelerata M

Anno Accademico 2025/2026

Queste linee guida sono state elaborate per supportare la realizzazione del progetto d'esame del corso di Sistemi di Elaborazione Accelerata M e la preparazione della prova orale del Modulo 2[1]. Sebbene il progetto possa riguardare sia temi del Modulo 1 (SIMD) che del Modulo 2 (GPU Programming, CUDA), questo documento si focalizza perlopiù sullo sviluppo di progetti basati su GPU e sulla preparazione della prova orale relativa a tali argomenti.

Si ricorda che l'esame di Sistemi di Elaborazione Accelerata M consiste di due parti principali:

- Una discussione del progetto, che può essere sviluppato individualmente o in gruppi di massimo due persone, previa approvazione da parte di entrambi i docenti.
- Due prove orali sui contenuti dei moduli, svolte in maniera indipendente con i rispettivi docenti.

Il voto finale dell'esame è determinato dalla somma di tre componenti:

- Valutazione del progetto (fino a 18 punti).
- Prova orale Modulo 1 (fino a 6 punti).
- Prova orale Modulo 2 (fino a 6 punti).

Il seguente documento è strutturato in due parti:

- Linee guida per il progetto: indicazioni metodologiche per affrontare le varie fasi del progetto, dalla sua concezione alla presentazione finale.
- Linee guida per la prova orale: indicazioni sulla preparazione e svolgimento della prova orale relativa al Modulo 2.

[1] Il presente documento può subire modifiche e aggiornamenti

Linee Guida per il Progetto

Sebbene il progetto possa riguardare sia temi del Modulo 1 (SIMD) che del Modulo 2 (GPU Programming, CUDA), questo documento si focalizza prevalentemente sullo sviluppo di progetti basati su CUDA, fornendo indicazioni specifiche per coloro che scelgono di approfondire questi aspetti.

Linee Guida Generali

1. Studio del Problema

Iniziare con un'analisi dettagliata del problema da risolvere. Questa fase deve includere lo studio della natura dei dati in input e output, la comprensione delle operazioni computazionali richieste e l'identificazione delle dipendenze tra i dati. Questa fase preliminare è cruciale per comprendere se e come il problema possa beneficiare della parallelizzazione su GPU, identificare eventuali limitazioni o criticità intrinseche e definire una strategia iniziale di decomposizione del problema. Le considerazioni fatte in questa fase guideranno le scelte implementative successive.

2. Implementazione Sequenziale Naïve

Realizzare una versione sequenziale standard del programma che servirà come riferimento per valutare i miglioramenti delle successive implementazioni parallele. Il codice dovrebbe essere scritto preferibilmente in C, come fatto durante il corso, ma è possibile utilizzare altri linguaggi come C++ o Python se più adatti al problema specifico.

L'implementazione sequenziale deve:

- Focalizzarsi sulla correttezza funzionale.
- Servire come baseline per il calcolo degli speedup.

Opzionalmente, per mostrare i concetti appresi nel Modulo 1, è possibile sviluppare anche versioni ottimizzate utilizzando istruzioni SIMD. Questo permetterebbe di evidenziare i benefici delle diverse tecniche di parallelizzazione (SIMD su CPU vs CUDA su GPU) e fornire un confronto più completo delle prestazioni. Tale approfondimento, sebbene non obbligatorio, può arricchire l'analisi e la comprensione delle diverse strategie di ottimizzazione disponibili.

3. Decomposizione e Struttura Parallela

Definire la strategia di decomposizione del problema identificando le componenti parallelizzabili e progettando una struttura parallela di alto livello. Questa fase richiede particolare attenzione all'organizzazione gerarchica di griglie, blocchi e thread, alla definizione del mapping tra dati e thread, e all'identificazione dei pattern di accesso ai dati.

4. Prima Implementazione Parallela Naïve

Sviluppare una prima versione del kernel CUDA concentrandosi sulla correttezza funzionale, senza particolare attenzione alle ottimizzazioni. Questa implementazione servirà come punto di partenza per identificare le aree di miglioramento.

Utilizzare Nsight Compute per un'analisi approfondita delle prestazioni, concentrandosi sui pattern di accesso alla memoria, resource partitioning e occupancy degli Streaming Multiprocessor (SM), presenza e impatto della divergenza dei warp, overhead dovuto a sincronizzazioni e stalli, e altri aspetti rilevanti emersi durante il corso.

L'analisi di questa versione naïve è fondamentale per stabilire una baseline di riferimento, identificare i principali colli di bottiglia e guidare le scelte implementative delle versioni successive. È importante in questa fase non limitarsi a riportare i valori delle metriche, ma interpretarli nel contesto specifico dell'applicazione per comprendere dove concentrare gli sforzi di ottimizzazione.

5. Ottimizzazioni Incrementali

Procedere con implementazioni successive migliorando gradualmente la performance. Prima di ogni ottimizzazione, verificare sempre la correttezza funzionale della soluzione rispetto alla versione sequenziale di riferimento. Per una corretta analisi delle prestazioni, utilizzare le metriche di durata fornite da Nsight Compute per i kernel CUDA, che garantisce misurazioni precise a livello di kernel, mentre per il codice CPU utilizzare timer appropriati per misurare il tempo di esecuzione della parte computazionale.

Ogni versione ottimizzata dovrebbe:

- Mostrare lo speedup relativo rispetto alla baseline sequenziale.
- Evidenziare criticità specifiche e le soluzioni adottate.
- Riportare metriche chiave come tempo di esecuzione, FLOPS, throughput di memoria, occupancy, dettagli sullo scheduler dei warp, ecc.

Nel processo di ottimizzazione, considerare le diverse tecniche viste a lezione quali l'uso di shared memory, loop unrolling, configurazioni di cache L1, riduzione di istruzioni condizionali, scelta ottimale del tipo di dato, ottimizzazione dell'occupancy (anche attraverso l'utilizzo dell'Occupancy Calculator), minimizzazione della divergenza dei warp, e altre strategie specifiche per il problema in esame. La scelta delle tecniche da applicare dovrebbe essere guidata dalle criticità emerse dall'analisi delle prestazioni e dalle caratteristiche specifiche dell'hardware utilizzato. È importante considerare la compute capability del device target, in quanto determina quali funzionalità e ottimizzazioni sono effettivamente disponibili e utilizzabili.

6. Analisi con il Roofline Model

Dove possibile, effettuare un'analisi tramite il Roofline Model. È importante notare che la possibilità di generare questa analisi attraverso Nsight Compute dipende dalla natura specifica del problema e dal tipo di operazioni coinvolte. L'analisi è disponibile per operazioni in virgola mobile (FP16, FP32, FP64), mentre presenta limitazioni per operazioni intere non tensoriali e kernel che utilizzano prevalentemente operazioni logiche, shift, confronti o calcoli di indirizzi.

Quando disponibile, il Roofline Model è fondamentale per comprendere se il kernel è limitato dalle capacità di calcolo (*compute bound*) o dalla bandwidth di memoria (*memory bound*), identificare il potenziale margine di miglioramento e guidare la scelta delle successive strategie di ottimizzazione. In caso contrario, è necessario basare l'analisi delle performance sulle altre metriche fornite da Nsight Compute, motivando adeguatamente le scelte di ottimizzazione effettuate.

7. Studio delle Prestazioni su Diversi Dataset

È fortemente consigliato valutare il comportamento dell'implementazione al variare delle dimensioni dei dati in input. Testare l'algoritmo con dataset di dimensioni significativamente diverse (es. da 500K a 64M elementi per array o matrici) è fondamentale per caratterizzare completamente il suo comportamento e validare l'efficacia delle ottimizzazioni implementate.

L'analisi deve identificare la soglia minima di dati necessaria per sfruttare efficacemente il parallelismo GPU, i punti di forte scaling dove l'aumento della dimensione porta a miglioramenti significativi dello speedup, e i punti di saturazione. Per ogni dimensione analizzata, riportare i tempi di esecuzione sequenziali e paralleli, lo speedup ottenuto e le metriche di Nsight Compute che variano con la dimensione dei dati.

Oltre alla dimensione del dataset, è consigliabile analizzare anche l'impatto dei parametri specifici dell'algoritmo implementato (ad esempio, dimensione delle finestre per filtri convoluzionali, raggio per algoritmi di neighborhood, ecc.) sulle prestazioni complessive del sistema.

Presentazione del Progetto

Contenuti

La presentazione deve fornire una visione chiara e precisa del lavoro svolto, illustrando:

- Il problema affrontato: motivazione, fondamenti teorici e applicazioni pratiche.
- Le fasi di sviluppo e ottimizzazione.
- Discussione dei colli di bottiglia e delle strategie adottate per superarli.
- I risultati ottenuti con analisi completa delle metriche.
- Eventuali altre strategie o metriche rilevanti per il progetto.

Analisi dei Risultati

L'utilizzo di Nsight Compute deve essere accompagnato da un'analisi approfondita e mirata. Non è sufficiente mostrare dei semplici screenshot di dati di profiling lasciando al docente l'interpretazione: è responsabilità dello studente evidenziare e discutere le metriche più significative, spiegando perché sono rilevanti per il caso specifico e come hanno guidato le scelte di ottimizzazione. Per ogni visualizzazione dei dati riportati da Nsight Compute, è necessario:

- Identificare chiaramente le metriche chiave per il contesto analizzato.
- Spiegare il significato e l'importanza dei valori osservati.
- Collegare le metriche alle decisioni di ottimizzazione successive.

La presentazione deve dimostrare il ragionamento alla base delle scelte implementative e delle strategie di ottimizzazione adottate. Si consiglia di supportare l'analisi con grafici e tabelle comparative che evidenzino chiaramente i miglioramenti ottenuti nelle varie fasi.

Formato e Consegna

La presentazione può essere preparata in formato PDF o PowerPoint, possibilmente non superiore a 15 pagine. È consigliato dedicare le prime pagine ad una chiara descrizione del dominio applicativo e del problema che si intende risolvere, fornendo il contesto necessario per comprendere le scelte progettuali successive. Per consentire una revisione adeguata, il materiale (Codice + Presentazione) deve essere inviato ai docenti almeno 4-5 giorni prima dell'esame. Durante la prova d'esame, la presentazione sarà discussa in maniera esaustiva, permettendo di esaminare le scelte progettuali e i risultati ottenuti.

Uso di GPU su Google Colab

Per chi non dispone di una GPU NVIDIA personale, si consiglia l'uso delle GPU disponibili su Google Colab. Queste consentono di:

- Utilizzare Nsight Compute per generare report puntuali.
- Generare file report con il comando:

  ```
  ncu --set full -o test_report ./application_name
  ```

- **Scaricare i file generati** (con estensione .ncu-rep) e **visualizzarli** tramite il software Nsight Compute installabile su PC.

È importante notare che è possibile installare il CUDA Toolkit e Nsight Compute localmente anche in assenza di una GPU NVIDIA. Questo permette di:

- Bypassare l'installazione dei driver GPU.
- Importare e analizzare i report generati su Colab.
- Utilizzare l'interfaccia grafica completa di Nsight Compute per un'analisi completa.

Si sconsiglia l'utilizzo di GPU personali che non supportano Nsight Compute, in quanto l'analisi esaustiva delle prestazioni attraverso questo strumento è un requisito fondamentale del progetto. La GPU Tesla T4 disponibile su Google Colab rappresenta quindi la soluzione consigliata, garantendo sia le funzionalità di profiling necessarie che prestazioni adeguate per lo sviluppo del progetto. Eventuali eccezioni all'utilizzo di Colab devono essere preventivamente discusse e approvate dal docente, ad esempio nel caso di necessità di

utilizzare librerie esterne non supportate dall'ambiente Colab. In questi casi, sarà necessario concordare soluzioni alternative per l'analisi delle prestazioni.

Nel caso in cui sia necessaria l'analisi delle interazioni host-device, Nsight System non è disponibile su Colab, ma è possibile utilizzare il profiler integrato di CUDA (`nvprof`) che fornisce informazioni basilari sui tempi di esecuzione dei kernel e sui trasferimenti di memoria tra host e device.

Linee Guida per la Prova Orale

L'orale rappresenta una verifica della comprensione dei concetti fondamentali del corso e non va sottovalutato. È importante sottolineare che si tratta di una componente distinta dalla presentazione del progetto: il docente porrà domande dirette sulla teoria e sugli argomenti trattati durante il corso per verificare l'effettiva comprensione dei concetti chiave.

Focus dell'Esame

Saranno oggetto di verifica tutti gli argomenti principali affrontati: dal modello di programmazione CUDA e la gerarchia dei thread, passando per il modello di esecuzione con particolare attenzione alla struttura e al funzionamento degli SM, alla gestione della divergenza dei warp, ai meccanismi di latency hiding e resource partitioning, fino all'occupancy e al CUDA Dynamic Parallelism. Inoltre, sarà rilevante la conoscenza del modello di memoria, includendo tutti gli aspetti della gerarchia delle memorie, i meccanismi di gestione come Unified Memory, zero-copy e pinned memory, l'Unified Virtual Addressing, i pattern di accesso alla memoria globale, l'utilizzo e i conflitti di banco della shared memory.

Cosa Non È Richiesto

Non è necessario memorizzare:

- Nomi specifici di funzioni CUDA e relativi parametri (es. `cudaFuncSetAttribute`, `cudaHostGetDevicePointer`, `cudaMemPrefetchAsync`).
- Parametri di compilazione (es. `-Xptxas -dlcm=cg`).
- Dettagli tecnici di ogni generazione/famiglia di architettura NVIDIA.
- Novità specifiche di ogni generazione di GPU.

- Aspetti storici o di mercato di NVIDIA.
- Dettagli di grafica 3D presentati come contesto introduttivo.
- Argomenti esplicitamente indicati come facoltativi durante il corso (es. librerie CUDA e loro applicazioni).

Struttura dell'Esame

Durante l'orale:

- Non verrà richiesto di scrivere codice.
- Verranno poste domande sugli argomenti fondamentali trattati nel corso. In particolare:
  - Modello di Programmazione CUDA
  - Modello di Esecuzione CUDA
  - Modello di Memoria CUDA
- Potrebbero essere proposti esercizi di ragionamento che lo studente dovrà svolgere e commentare passo dopo passo durante l'orale. Ad esempio:
  - Calcolo dei warp necessari per un certo throughput usando la legge di Little.
  - Analisi del grado di serializzazione dovuto ai bank conflict nella shared memory.
  - Stima dell'occupancy teorica dato un certo utilizzo di risorse.
  - Stima del numero di transazioni di memoria necessarie per un certo pattern di accesso in memoria globale.

Consigli Finali

La memorizzazione di caratteristiche specifiche non è richiesta, mentre è essenziale la comprensione approfondita dei principi di funzionamento. La prova si svolge come un dialogo costruttivo finalizzato alla valutazione della preparazione complessiva sulla materia.

Guida alle Domande d'Esame

Come supporto per comprendere meglio la tipologia di domande che potrebbero essere poste durante l'orale, si riportano **alcuni esempi** puramente indicativi.

Esempi di Domande Possibili

Di seguito alcuni esempi di possibili domande sui temi del corso:

- Come funziona il meccanismo di latency hiding e quale ruolo gioca la legge di Little?
- Quali sono le differenze chiave nel trattamento della divergenza dei warp prima e dopo l'introduzione dell'architettura Volta?
- Come influisce l'utilizzo della shared memory sull'occupancy di un SM?
- Come viene organizzata una matrice in memoria lineare e come si accede ad un elemento specifico date le coordinate della matrice?
- Quali sono le differenze tra pinned memory e pageable memory?

Esempi di Domande che Non Verranno Poste

Di seguito sono riportati alcuni esempi di domande che **NON** saranno sicuramente oggetto dell'esame orale, in quanto richiedono dettagli tecnici specifici non rilevanti per la comprensione dei concetti essenziali:

- Qual è la sintassi esatta della funzione `cudaFuncSetAttribute`?
- Elenca tutte le novità introdotte con l'architettura Ampere.
- Scrivi il codice per implementare la riduzione parallela.
- In che anno è stata introdotta l'Unified Memory?
- Qual è la dimensione esatta di una transazione di memoria sulle GPU con compute capability 3.x?
- Quanti registri sono disponibili per ogni thread su un SM dell'architettura Fermi?
- Qual è la configurazione predefinita della cache L1 su GPU con compute capability 6.0?

- Elenca tutte le caratteristiche della compute capability 7.0.
- Qual è la latenza media di un'istruzione di load/store sulla memoria globale, misurata su una Tesla K80?
- Quanti clock cycle sono necessari per completare una divisione in floating-point su una GPU Pascal?
- Qual è la larghezza di banda per direzione PCIe 4.0?
- La GPU NVIDIA H100 utilizza memoria GDDR o HBM?
- Quanti CUDA core hanno le GPU della famiglia Maxwell?
- Qual è il peak throughput FP64 dell'architettura A100?
- Quali caratteristiche sono introdotte dalla terza generazione di Tensor Core?

Nota Importante

Questi esempi sono forniti a titolo puramente indicativo e non costituiscono un elenco esaustivo.