Confronto tra versione sequenziale e versioni parallelizzate con CUDA e OpenMP di convoluzioni

Silvia Dani
Indirizzo e-mail
silvia.dani@stud.unifi.it

Niccolò Niccoli
Indirizzo e-mail
niccolo.niccoli@stud.unifi.it

Abstract

Questo esperimento vuole andare confrontare i tempi di esecuzione di una sezione di codice che applica un filtro su di un'immagine utilizzando le convoluzioni (normali e separabili) che viene eseguita in modalità sequenziale, parallelizzata con OpenMP e parallelizzata con CUDA. Successivamente va a misurare lo speed up che si ottiene mediante la parallelizzazione.

1. Introduzione

Questo elaborato verte sul confronto del codice necessario ad eseguire delle convoluzioni (sia separabili che non) su di un'immagine. Il codice viene eseguito in modo sequenziale, parallelo con OpenMP e parallelo con CUDA. Le convoluzioni sulle immagini vengono eseguite in modo tale che venga applicato un filtro su ogni pixel dell'immagine di riferimento.
L'immagine viene suddivisa in tiles e su ognuna di esse viene applicata la maschera. Nelle convoluzioni separabili è necessario essere sicuri che la convoluzione fatta in una direzione abbia terminato prima di eseguirla nell'altro verso.

2. Progettazione

Il metodo per applicare il filtro esamina in sequenza tutti i pixel dell'immagine. Per ciascuno di essi si moltiplica il valore di quest'ultimo e i valori dei pixel confinanti per i valori corrispondenti nel kernel. I risultati vengono poi sommati e il pixel iniziale viene impostato a questo risultato finale. Il filtro che viene applicato nel corso di questi esperimenti è un box blur. I pixel di bordo sono stati gestiti considerando quelli fuori dall'immagine come pixel neri.

3. Parallelizzazione

3.1. Con OpenMP

Nella versione parallelizzata con OpenMP per effettuare la parallelizzazione si è deciso di utilizzare

```
#pragma omp parallel
#pragma omp for
```

per applicare il filtro su ogni chunk sia per la versione separabile che non. La scomposizione in chunk può essere apprezzata nel seguente codice:

```
    #pragma omp parallel default
        (none) shared (src, dst,
        kernel, tileHeight,
        numProcs, kernelSize,
        offset)
#pragma omp for
    for (int threadIdx = 0;
        threadIdx < numProcs;
        threadIdx++) {
          filter(src, dst, kernel,
              kernelSize, offset,
            tileHeight * threadIdx
            , tileHeight * (
            threadIdx + 1));
    }
```

3.2. Con CUDA

Nella versione parallelizzata con CUDA utilizziamo un approccio *tile based* dove tutti i thread di un blocco caricano in memoria condivisa ma solo alcuni eseguono la convoluzione e salvano il risultato nell'immagine di output. Questa strategia viene seguita sia quando viene eseguita la convoluzione con un filtro separabile che non.

Nel caso della convoluzione separabile vengono eseguiti due kernel, uno che riceve come parametro un vettore colonna e uno che riceve un vettore riga.

Le dimensioni del blocco e della griglia dipendono sia dall'immagine su cui deve essere applicato il filtro, sia dal tipo di filtro che viene applicato:

- La dimensione del blocco è definita come

```
#define BLOCK_WIDTH (
    TILE_WIDTH + MASK_WIDTH -
    1)
dim3 dimBlock(BLOCK_WIDTH,
    BLOCK_WIDTH);
```

- La dimensione della griglia è invece definita come

```
dim3 dimGrid((src.cols -
    1) / TILE_WIDTH + 1, (
    src.rows - 1) /
    TILE_WIDTH + 1, src.
    channels());
```

Al fine di migliorare i tempi di accesso alla memoria[1] dove sono contenuti i dati che vengono utilizzati dai kernel è stato scelto di utilizzare le keyword *const* e *__restrict__* così da dire al compilatore che la locazione di memoria a cui punta un certo puntatore non viene mai scritta ma solo letta. Un esempio di questo accorgimento si può vedere tra i parametri della funzione che serve per eseguire le convoluzioni "normali":

```
__global__ void convolution(
    const uchar* __restrict__
    src, int srcWidth, int
    srcHeight, int srcChannels
    ,const float* __restrict__
     convKernel, int
    kernelWidth, int
    kernelHeight, uchar* dst)
```

4. Esperimenti

Gli esperimenti che sono stati svolti su entrambe le versioni parallelizzate e sulla versione sequenziale consistono nel misurare il tempo di esecuzione al variare delle dimensioni del filtro che viene applicato (3x3, 5x5, 9x9, 15x15, 31x31) e delle dimensioni dell'immagine su cui viene applicato il suddetto filtro (512x512, 1024x1024, 2048x2048, 4096x4096, 8192x8192).

Nella versione parallelizzata con CUDA la variazione della dimensione del filtro comporta anche una variazione della dimensione del tile perché il numero massimo di thread per blocco è 1024 [2] e quindi siccome la dimensione del blocco è definita in base alla dimensione del tile e del filtro, all'aumentare della grandezza del filtro può corrispondere una diminuzione della grandezza del tile. In particolare la larghezza "base" dei tile che è stata scelta è 16 pixel e quindi quando è stato utilizzato il filtro 31x31, è stato necessario ridurre la larghezza del tile è a 2 pixel.

Con OpenMP è stato effettuato anche un esperimento che vede fissate le dimensioni dell'immagine e del filtro e analizza i tempi di esecuzione al variare del numero di thread che vengono utilizzati per parallelizzare il calcolo. I tempi sono stati misurati relativamente al codice parallelizzato utilizzando 2, 4, 8, 16 e 32 thread.

L'hardware su cui sono stati eseguiti questi esperimenti è composto da un processore con 8 core logici e una scheda video Nvidia GeForce GTX 1650 (CC 7.5). Ciascuna versione parallelizzata viene confrontata con la versione sequenziale

[1] https://developer.nvidia.com/blog/cuda-pro-tip-optimize-pointer-aliasing/

[2] https://forums.developer.nvidia.com/t/maximum-number-of-threads-on-thread-block/46392

Figure 1. Immagine originale

Figure 2. Convoluzione sequenziale

Figure 3. Convoluzione sequenziale separabile

Figure 4. Convoluzione con CUDA

Figure 5. Convoluzione con CUDA separabile

Figure 6. Convoluzione con OpenMP

Figure 7. Convoluzione con OpenMP separabile

Figure 8. Esempio di output delle diverse implementazioni.
Come si può vedere gli output sono identici e quindi le implementazioni sono equivalenti.

per mettere in risalto lo speed up che si ottiene. Il codice che confronta la versione CUDA con la versione sequenziale è stato compilato utilizzando nvcc e msvc. Il codice che invece confronta la versione OpenMP con quella sequenziale è stato compilato con gcc. Questo fatto comporta l'ottenimento di tempi di esecuzione differenti anche per quanto riguarda la versione sequenziale, quindi al fine di confrontare le due strategie di parallelizzazione viene preso in considerazione lo speed up rispetto alla versione sequenziale.

5. Risultati

5.1. OpenMP

Dalle tabelle 2, 4, 6, 8 e 10 (in appendice) si nota che lo speed up massimo relativo alle convoluzioni non separabili che è stato possibile ottenere con OpenMP è 5.979 e che in generale non ci sono grosse differenze dovute all'aumento di dimensioni dell'immagine. Si vede inoltre che all'aumentare del numero dei thread lo speed up aumenta fino ad arrivare a quando il numero dei thread utilizzati è uguale al numero dei thread logici della CPU, a questo punto il tempo di esecuzione smette di decrescere e quindi anche lo speed up resta stabile.

Sempre dalle stesse tabelle ma guardando i dati relativi alle convoluzioni separabili si può vedere che il ragionamento fatto per il caso precedente vale ancora; tuttavia lo speed up massimo che è stato registrato è 2.115.

5.2. CUDA

Dalle tabelle 1, 3, 5, 7 e 9 si può invece vedere che lo speed up ottenuto con CUDA è nettamente più alto di quello ottenuto utilizzando OpenMP. Parallelizzando utilizzando la scheda video è possibile arrivare a impiegare fino a 162.74 volte meno rispetto a eseguire le stesse operazioni sulla CPU in modo sequenziale (tabella 7).

Ponendo attenzione sulla tabella 9 si può notare che in questo caso lo speed up è molto minore rispetto a tutti gli altri casi. Questo è dovuto alla riduzione della grandezza del tile (in tutti gli altri esperimenti la larghezza del tile era 16, in questo

è stata impostata a 2).

6. Conclusioni

Dai risultati degli esperimenti è possibile concludere che la parallelizzazione dell'operazione di applicazione di un filtro mediante delle convoluzioni comporta dei vantaggi sensibili dal punto di vista del tempo di esecuzione. Questi vantaggi sono maggiori se l'algoritmo viene parallelizzato utilizzando CUDA.

Nonostante lo speed up che si ottiene con le convoluzioni separabili sia minore di quello che si ottiene nell'altro caso, siccome l'algoritmo è in generale meno dispendioso conviene utilizzare filtri separabili.

7. Appendice

Dimensioni immagine (in pixel)	Sequenziale	Parallelo	Speed up	Sequenziale separabile	Parallelo separabile	Speed up (separabile)
512x512	0.025167	0.000443	56.77193774	0.012808	0.00066	19.4091529
1024x1024	0.096339	0.001486	64.83135935	0.054289	0.002197	24.71483201
2048x2028	0.42395	0.00559	75.8462144	0.225709	0.007158	31.53064931
4096x4096	1.75098	0.022824	76.71560574	0.842219	0.02763	30.48215882
8192x8192	6.95061	0.130942	53.08159338	3.8581	0.108906	35.42596368

Table 1. Tempi di esecuzione al variare della dimensione dell'immagine. Dimensioni del filtro applicato: 3x3. Parallelizzato utilizzando CUDA.

Dimensioni immagine (in pixel)	Sequenziale	Parallelo (con 2 core)	Parallelo (con 4 core)	Parallelo (con 8 core)	Parallelo (con 16 core)	Parallelo (con 32 core)	Speed up
512x512	0.115631	0.06383	0.031729	0.023122	0.044956	0.024347	5.000887
1024x1024	0.455554	0.230751	0.16388	0.128871	0.116055	0.118892	3.925329
2048x2028	1.70542	0.921283	0.545917	0.346267	0.333915	0.349029	5.107348
4096x4096	7.51593	3.78123	2.20127	1.26843	1.2923	1.27522	5.92538
8192x8192	29.6518	15.1769	8.71391	4.95909	5.06372	5.02872	5.979282
	Sequenziale separabile	Parallelo separabile (con 2 core)	Parallelo separabile (con 4 core)	Parallelo separabile (con 8 core)	Parallelo separabile (con 16 core)	Parallelo separabile (con 32 core)	Speed up (separabile)
512x512	0.068624	0.057459	0.043367	0.043981	0.044229	0.04641	1.58242
1024x1024	0.270193	0.198722	0.169576	0.159173	0.158443	0.153971	1.75483
2048x2028	1.25745	0.84075	0.713053	0.629894	0.612942	0.621292	2.051499
4096x4096	4.48741	3.32345	2.73495	2.6376	2.65347	2.5754	1.742413
8192x8192	18.8068	13.9642	11.8933	10.246	11.1065	11.0182	1.835526

Table 2. Tempi di esecuzione al variare della dimensione dell'immagine. Dimensioni del filtro applicato: 3x3. Lo speed up è calcolato come tempo sequenziale corrispondente ad una certa immagine / minimo dei tempi ottenuti mediante parallelizzazione, quindi il numero presente in tabella rappresenta l'accelerazione massima che è stata ottenuta. Parallelizzato utilizzando OpenMP.

Dimensioni immagine (in pixel)	Sequenziale	Parallelo	Speed up	Sequenziale separabile	Parallelo separabile	Speed up (separabile)
512x512	0.052131	0.000604	86.29565	0.021837	0.000763	28.60506
1024x1024	0.208695	0.001898	109.9494	0.082454	0.002138	38.56211
2048x2028	0.825305	0.007358	112.1719	0.342456	0.007503	45.64254
4096x4096	3.8867	0.028793	134.99	1.22595	0.028869	42.46538
8192x8192	15.0385	0.198526	75.75078	5.31706	0.116713	45.55671

Table 3. Tempi di esecuzione al variare della dimensione dell'immagine. Dimensioni del filtro applicato: 5x5. Parallelizzato utilizzando CUDA.

Dimensioni immagine (in pixel)	Sequenziale	Parallelo (con 2 core)	Parallelo (con 4 core)	Parallelo (con 8 core)	Parallelo (con 16 core)	Parallelo (con 32 core)	Speed up
512x512	0.250932	0.166562	0.090484	0.052825	0.080976	0.052518	4.778037
1024x1024	1.0595	0.582708	0.342345	0.229535	0.212922	0.241021	4.976001
2048x2028	4.49249	2.4226	1.40465	0.937522	0.811	0.822516	5.539445
4096x4096	17.6396	9.42933	5.32626	3.11804	3.24441	3.32639	5.657272
8192x8192	71.625	37.0639	20.5266	12.5618	12.4892	12.3613	5.794293
	Sequenziale separabile	Parallelo separabile (con 2 core)	Parallelo separabile (con 4 core)	Parallelo separabile (con 8 core)	Parallelo separabile (con 16 core)	Parallelo separabile (con 32 core)	Speed up (separabile)
512x512	0.106104	0.075026	0.067557	0.05907	0.057405	0.062648	1.848341
1024x1024	0.455962	0.296701	0.242869	0.229488	0.238076	0.274681	1.986866
2048x2028	1.89029	1.16617	0.966116	0.944536	0.893693	0.897217	2.115145
4096x4096	7.34521	4.94548	4.10266	3.70754	3.55758	3.51825	2.087745
8192x8192	30.6917	20.3374	15.7459	15.2128	15.1844	15.0332	2.041595

Table 4. Tempi di esecuzione al variare della dimensione dell'immagine. Dimensioni del filtro applicato: 5x5. Lo speed up è calcolato come tempo sequenziale corrispondente ad una certa immagine / minimo dei tempi ottenuti mediante parallelizzazione, quindi il numero presente in tabella rappresenta l'accelerazione massima che è stata ottenuta. Parallelizzato utilizzando OpenMP.

Dimensioni immagine (in pixel)	Sequenziale	Parallelo	Speed up	Sequenziale separabile	Parallelo separabile	Speed up (separabile)
512x512	0.138503	0.001211	114.3802	0.025986	0.000924	28.12588
1024x1024	0.553139	0.004477	123.5402	0.102715	0.003139	32.71804
2048x2028	2.24156	0.017806	125.8879	0.412621	0.01168	35.32593
4096x4096	9.84972	0.069944	140.8227	2.32176	0.046119	50.34314
8192x8192	38.3746	0.266941	143.7569	7.53318	0.161333	46.69336

Table 5. Tempi di esecuzione al variare della dimensione dell'immagine. Dimensioni del filtro applicato: 9x9. Parallelizzato utilizzando CUDA.

Dimensioni immagine (in pixel)	Sequenziale	Parallelo (con 2 core)	Parallelo (con 4 core)	Parallelo (con 8 core)	Parallelo (con 16 core)	Parallelo (con 32 core)	Speed up
512x512	0.835089	0.521776	0.263256	0.190928	0.203877	0.185587	4.499717
1024x1024	3.53149	1.87496	1.04074	0.738005	0.704193	0.664733	5.312644
2048x2028	14.5263	7.38185	4.19136	2.62607	2.48568	2.60837	5.843994
4096x4096	57.9624	29.6818	16.5983	10.0316	9.94278	10.114	5.829597
8192x8192	226.334	117.78	65.4336	39.7032	39.7869	39.752	5.700649
	Sequenziale separabile	Parallelo separabile (con 2 core)	Parallelo separabile (con 4 core)	Parallelo separabile (con 8 core)	Parallelo separabile (con 16 core)	Parallelo separabile (con 32 core)	Speed up (separabile)
512x512	0.179755	0.133012	0.116957	0.100853	0.104406	0.099552	1.805641
1024x1024	0.82335	0.533279	0.437647	0.479628	0.399795	0.411965	2.05943
2048x2028	3.05478	2.17786	1.7929	1.68635	1.62273	1.69583	1.882494
4096x4096	12.7151	9.15445	7.56755	7.0546	7.03753	6.88634	1.846423
8192x8192	48.3518	35.8448	29.9048	27.0298	27.0637	27.4809	1.788833

Table 6. Tempi di esecuzione al variare della dimensione dell'immagine. Dimensioni del filtro applicato: 9x9. Lo speed up è calcolato come tempo sequenziale corrispondente ad una certa immagine / minimo dei tempi ottenuti mediante parallelizzazione, quindi il numero presente in tabella rappresenta l'accelerazione massima che è stata ottenuta. Parallelizzato utilizzando OpenMP.

Dimensioni immagine (in pixel)	Sequenziale	Parallelo	Speed up	Sequenziale separabile	Parallelo separabile	Speed up (separabile)
512x512	0.339669	0.002712	125.2559	0.041986	0.001093	38.39985
1024x1024	1.38293	0.01047	132.085	0.171274	0.003948	43.37918
2048x2028	5.44511	0.041135	132.3717	0.688943	0.014724	46.7908
4096x4096	23.0781	0.329958	69.94254	2.90478	0.047319	61.3877
8192x8192	92.7028	0.56964	162.7393	13.1883	0.188492	69.96743

Table 7. Tempi di esecuzione al variare della dimensione dell'immagine. Dimensioni del filtro applicato: 15x15. Parallelizzato utilizzando CUDA.

Dimensioni immagine (in pixel)	Sequenziale	Parallelo (con 2 core)	Parallelo (con 4 core)	Parallelo (con 8 core)	Parallelo (con 16 core)	Parallelo (con 32 core)	Speed up
512x512	2.29458	1.26523	0.704288	0.408954	0.442293	0.444197	5.610851
1024x1024	9.45043	4.96044	2.77246	1.70731	1.65969	1.68792	5.694093
2048x2028	37.2596	19.673	11.1669	6.65982	6.73033	6.7282	5.594686
4096x4096	150.963	78.1658	44.4156	26.7598	26.7126	26.9166	5.651378
8192x8192	593.859	320.729	178.507	107.768	107.43	107.727	5.527869
	Sequenziale separabile	Parallelo separabile (con 2 core)	Parallelo separabile (con 4 core)	Parallelo separabile (con 8 core)	Parallelo separabile (con 16 core)	Parallelo separabile (con 32 core)	Speed up (separabile)
512x512	0.302936	0.21865	0.166837	0.157668	0.172167	0.165403	1.921354
1024x1024	1.22954	0.823556	0.729218	0.632062	0.618914	0.613304	2.004781
2048x2028	4.7157	3.35847	2.73474	2.46721	2.67188	2.49863	1.911349
4096x4096	18.5384	13.7261	11.4416	10.4014	10.3249	10.9171	1.795504
8192x8192	73.6205	54.7865	46.7803	41.2145	40.8399	42.0113	1.802661

Table 8. Tempi di esecuzione al variare della dimensione dell'immagine. Dimensioni del filtro applicato: 15x15. Lo speed up è calcolato come tempo sequenziale corrispondente ad una certa immagine / minimo dei tempi ottenuti mediante parallelizzazione, quindi il numero presente in tabella rappresenta l'accelerazione massima che è stata ottenuta. Parallelizzato utilizzando OpenMP.

Dimensioni immagine (in pixel)	Sequenziale	Parallelo	Speed up	Sequenziale separabile	Parallelo separabile	Speed up (separabile)
512x512	1.29859	0.203361	6.385639	0.068066	0.02711	2.510736
1024x1024	5.20358	0.681825	7.631841	0.294446	0.103219	2.852634
2048x2028	21.0456	2.98823	7.042831	1.16747	0.41218	2.832428
4096x4096	88.2998	11.189	7.891661	5.11679	1.64945	3.102119
8192x8192	377.616	43.4623	8.688357	22.6498	6.60364	3.429896

Table 9. Tempi di esecuzione al variare della dimensione dell'immagine. Dimensioni del filtro applicato: 31x31. Parallelizzato utilizzando CUDA.

Dimensioni immagine (in pixel)	Sequenziale	Parallelo (con 2 core)	Parallelo (con 4 core)	Parallelo (con 8 core)	Parallelo (con 16 core)	Parallelo (con 32 core)	Speed up
512x512	9.72364	5.03389	2.85831	1.80663	1.76988	1.71396	5.673201
1024x1024	36.2949	20.0278	11.6975	6.85185	6.98169	7.01361	5.297095
2048x2028	157.498	79.7285	46.3621	27.8501	28.0162	28.2433	5.655204
4096x4096	615.233	326.167	183.875	112.21	112.361	113.672	5.482871
8192x8192	2262.36	1306.15	725.673	454.073	455.736	456.605	4.982371
	Sequenziale separabile	Parallelo separabile (con 2 core)	Parallelo separabile (con 4 core)	Parallelo separabile (con 8 core)	Parallelo separabile (con 16 core)	Parallelo separabile (con 32 core)	Speed up (separabile)
512x512	0.542885	0.368192	0.330302	0.285492	0.345428	0.298921	1.901577
1024x1024	2.09803	1.57377	1.30119	1.22649	1.23138	1.2422	1.710597
2048x2028	9.00222	6.1985	5.39465	4.68247	5.03463	4.69261	1.922537
4096x4096	35.6195	25.9324	20.9516	19.3709	19.8486	19.8083	1.838815
8192x8192	136.169	99.7976	86.5429	80.245	79.5241	80.4697	1.712299

Table 10. Tempi di esecuzione al variare della dimensione dell'immagine. Dimensioni del filtro applicato: 31x31. Lo speed up è calcolato come tempo sequenziale corrispondente ad una certa immagine / minimo dei tempi ottenuti mediante parallelizzazione, quindi il numero presente in tabella rappresenta l'accelerazione massima che è stata ottenuta. Parallelizzato utilizzando OpenMP.