

CUDA: Hardware e Prestazioni

Argomenti

1 Caratteristiche hardware e prestazioni

Banda di trasferimento dati

Un kernel cuda opera su dati residenti sulla memoria globale: può svolgere direttamente i conti su essi, o trasferire i dati sulla memoria condivisa. La banda di trasferimento dati dalla memoria globale ai registri dei thread è un parametro facilmente calcolabile a partire dalle caratteristiche fisiche dell'hardware.

Hardware		Memoria			
famiglia	Dispositivo	interfaccia [bit]	tipo	clock [MHz]	banda [GB/s]
G80	GeForce 8800 GTX	384	DDR2	900	86.4
GT200	GTX280	512	GDDR3	1107	141.7
fermi	C2070	384	GDDR5	1500	144.0

Esempio: GTX280

interfaccia [bit]

bit per clock

$$(512 * 2 * 1107) / 8 = 141.7 \text{ GB/s}$$

frequenza di clock [MHz]

Capacità computazionale

La capacità computazionale di una scheda GPU rappresenta il numero massimo di elaborazioni floating-point sostenibile dall'architettura.

Hardware		cores			
famiglia	Dispositivo	sp	clock [Mhz]	op. per clock	capacità comp. [GFlop/s]
G80	GeForce 8800 GTX	128	1350	3	518.4
GT200	GTX280	240	1296	3	933.1
fermi	C2070	448	1150	3	1545.6

Esempio: GTX280

cores (sp)

operazioni per clock

$$240 * 3 * 1296 = 933.12 \text{ GFlop/s}$$

frequenza di clock [MHz]

Operazioni per clock: 1 MAD (2 operazioni) + 1 valutazione di funzione speciale.

Banda di trasferimento e capacità computazionale sono i limiti fisici imposti dall'architettura scelta. Per un dato kernel, possiamo calcolare il rapporto tra le operazioni fp e gli accessi alla memoria globale:

$$\text{CGMA} = \frac{\# \text{ fp-op}}{\# \text{ access}}$$

CGMA: Compute to Global Memory Access (i.e. Operational Intensity)
Grazie a questo rapporto, possiamo individuare i fattori che limitano le prestazioni del nostro kernel.

Esempio (I)

Prima implementazione del kernel per il calcolo matrice-matrice

```
__global__ void my_sgemv_1kernel (int M, int K, float *A,
                                int lda, float *B, int ldb, float *C, int ldc)
{
    int idx_x = blockIdx.x * blockDim.x + threadIdx.x;
    int idx_y = blockIdx.y * blockDim.y + threadIdx.y;
    int kk; float sum = 0;

    if (idx_x < M && idx_y < M){
        for(kk = 0; kk < K; kk++){
            sum += A[idx_y*lda+kk] * B[kk*ldb + idx_x];
        }
        C[idx_y*ldc + idx_x] += alpha*sum;
    }
}
```

$$\text{CGMA} = 1.0$$

Per le variabili `float` (4 bytes), ci aspettiamo che la capacità computazionale del nostro kernel non superi $86.4/4 = 21.6$ GFlop/s per l'architettura GeForce 8800 GTX

Esempio (II)

Seconda implementazione del kernel per il calcolo matrice-matrice

```

__global__ void my_sgemv_2kernel (int M, int K, float *A,
                                int lda, float *B, int ldb, float *C, int ldc)
{
    __shared__ float As[TILEWIDTH][TILEWIDTH];
    __shared__ float Bs[TILEWIDTH][TILEWIDTH];

    int idx_x = blockIdx.x * TILEWIDTH + threadIdx.x;
    int idx_y = blockIdx.y * TILEWIDTH + threadIdx.y;
    int jj, kk; float sum = 0;

    for(kk = 0; kk < K/TILEWIDTH; kk++){
        As[threadIdx.y][threadIdx.x] =
            A[idx_y*lda + (kk*TILEWIDTH + threadIdx.x)];
        Bs[threadIdx.y][threadIdx.x] =
            B[(kk*TILEWIDTH + threadIdx.y) * ldb + idx_x];

        __syncthreads();

        for(jj = 0; jj < TILEWIDTH; jj++)
            sum += As[threadIdx.y][jj] * Bs[jj][threadIdx.x];

        __syncthreads();
    }
    C[idx_y*ldc + idx_x] += alpha*sum;
}

```

Esempio (III)

In questo caso, CGMA è aumentato di un fattore pari a TILEWIDTH, in particolare:

$$\text{CGMA} = \text{TILEWIDTH}$$

Per le variabili `float` (4 bytes), ci aspettiamo che la capacità computazionale del nostro kernel non superi:

$$86.4/4 * 16 = 345.6 \text{ GFlop/s}$$

per l'architettura GeForce 8800 GTX e TILEWIDTH 16.