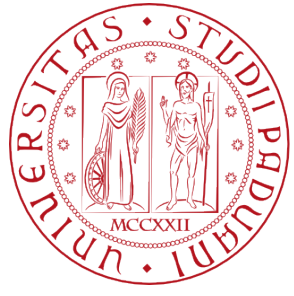
Accelerazione GPU nella simulazione/controllo/identificazione di sistemi fisici

Diego Casella, Fabio Marcuzzi, Marco Virgulin

February 24, 2015



Università degli Studi di Padova

Contenuti

Introduzione

Vogliamo capire come possiamo utilizzare le GPU per accelerare la simulazione/controllo/identificazione di sistemi fisici, e se possibile anche la co-simulazione.
I problemi sono tutti quelli trattabili da CfL[1] (...), piú la co-simulazione di un sistema di controllo.

[1] CfL link

Parallelismo massivo

- Quanti processori (es. nella nostra scheda abbiamo 7 multiprocessors, ognuno dei quali ha 48 cores) e come si tengono impegnati:

  - blocks: sono completamente paralleli
  - threads: possono
    - comunicare (shared memory per i threads di uno stesso blocco; i dati devono transitare per la memoria globale)
    - sincronizzarsi (tutti i threads devono raggiungere la barriera);
    - esempi utili: un thread elabora uno stencil
    - c'é un numero massimo di threads per blocco, da cui il numero di blocchi, detto N dimensione del problema diventa (#define BLOCKS (N + (THREADS 1)) / THREADS)

$\rightarrow$ ogni core pu eseguire in parallelo un *warp* di threads ! (spesso 32 threads)

Parallelismo massivo

- il collo di bottiglia della memoria: tutti questi processori devono essere alimentati di operandi ...

- device (GPU):
  - locale
  - shared
  - globale (la CPU puó spedire dati solo qui)
- host (CPU):
  - cache I e II livello
  - principale (RAM)
  - secondaria (disco)

Esempio 1: PDE

Saltando la generazione di meshes, vediamo la distribuzione della discretizzazione del dominio (mesh) e dei dati del problema:

```
data = zeros((NT,9),dtype = 'float32') # coordinate dei vertici
    degli elementi in formato  [x11,x12,x13,y11,y12,y13,...]
data[:,0:3] = self.mesh.get_nx()[self.mesh.get_triangles()]
data[:,3:6] = self.mesh.get_ny()[self.mesh.get_triangles()]
data = data.flatten()
...
gal_p1_A(cuda.InOut(data),cuda.Out(idxs),cuda.In(ijk),...,block =
    (256, 1, 1), grid=(int(np.ceil(NT/256)), 1))
```

dove

```
code = open('gFEM.cu', 'r').read()
mod = SourceModule(code)
gal_p1_A = mod.get_function("compute_A")
```

e corrisponde alla chiamata (del modulo CUDA su GPU):

```
compute_A<<<grid, block>>>(float* data, int* idxs, int* ijk, ...)
```

Esempio 1: PDE

- la creazione del modello discreto:

  - la memorizzazione della matrice sparsa;

    ```
    scipy.sparse.coo_matrix
    ```

  - la costruzione delle matrici locali;

    ```
    gal_p1_A(cuda.InOut(data),cuda.Out(idxs),cuda.In(ijk),NN,NT,
        sigma,mu,cuda.In(b),tDt,NN,NT,block = (256, 1, 1), grid=(np
        .ceil(NT/256), 1))
    ```

    calcola i contributi dei termini PDE su ogni triangolo della mesh (ad ogni thread viene associato un triangolo).

  - l'assemblaggio;

    `thrust::reduce_by_key` (Thrust library)
    per sommare i contributi sui nodi con lo stesso indice (che é la chiave).

Parallelismo massivo

Esempio 1: PDE

Soluzione del problema discreto con:

`cusp::krylov::gmres` (Cusp library)

Generalized Minimum Residual (GMRES) method.

La libreria viene caricata da `ctypes` (e contiene al suo interno i moduli CUDA):

```
cusp_so = ctypes.CDLL(FOLDER +'pycusp' + '.so')
cusp_gmres = cusp_so.gmres
...
cusp_gmres(c_ptr(rows_d.ptr),c_ptr(cols_d.ptr),c_ptr(vals_d.ptr),
    A_coo.shape[0],A_coo.shape[1],A_coo.nnz,c_ptr(b_d.ptr),c_ptr(x_d
    .ptr),500,ctypes.c_float(1e-6))
```

Esempio 2: Problemi Inversi

```
while (<condizione di terminazione>) {
  ...
  /* Data generation */
  dim3 threadsPerBlockBuild(2, N_delta, N_parameters);
  BuildXtemp<<<1, threadsPerBlockBuild>>>(X_temp, X_temp2, deltatemp
       , deltatemp2, u, param, param_min, param_max, delta_ini,
      delta_max);
  ...
  /* Psi generation */
  dim3 threadsPerBlockPsi(N_delta, N_parameters);
  BuildPsi<<<1, threadsPerBlockPsi>>>(Psi, vdiff, X_temp, X_temp2,
      deltatemp, deltatemp2);
  ...
  /* Gauss-Newton approximation */
  dim3 threadsPerBlockGN(N_updates);
  GN<<<1, threadsPerBlockGN>>>(app_delta, err, u, Psi, vdiff, param,
       param_min, param_max);
  ...
}
```

Esempio 3: Linear Algebra

Parallelismo massivo

Esempio 4: co-simulazioni multiple
- “multiple (CPU) threads can share a device”.

Programmazione

- CUDA: un'estensione del C per il parallelismo e la gestione del device

Programmazione

- PyCUDA

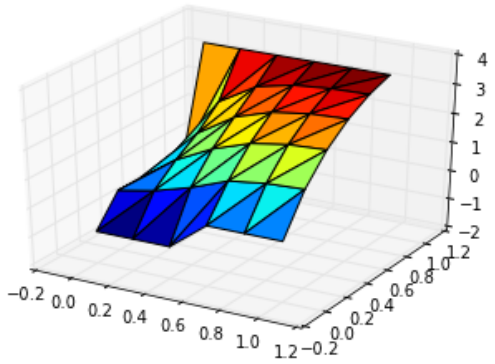
- Thrust

Linguaggi ed ambienti di sviluppo

- nvcc

- Marco, puoi scrivere qualcosa sul profiler ?

Esempi



(a) Soluzione FEM

- esempi PDE tesi Barasti

Esempi

- applicazione alla finanza

Esempi

- tesi Marco su fMRI ?

Esempi

Q&A