Tutorial Parallel Programming and High Performance Computing

CUDA Kernels for SpMV

Daniel Butnaru, Christoph Kowitz

January 23, 2012

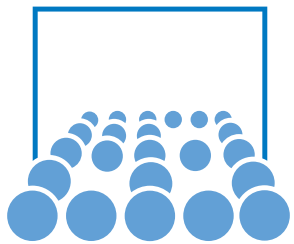

Table of Contents

Compressed Sparse Row (CSR) Kernel 1

First straightforward approach: each thread does a row times vector multiplication

```
1  __global__ void k_csr_mat_vec_mm(ptr, J, Val, x, y)
2  {
3    int row = blockDim.x * blockIdx.x + threadIdx.x ;
4    if (row < num_rows){
5      float dot = 0;
6      int row_start = ptr [ row ];
7      int row_end = ptr [ row+1];
8
9      for (int jj = row_start; jj < row_end; jj++) {
10       dot += Val [ jj ] * x [ J [ jj ]];
11     }
12
13     y[row] += dot ;
14   }
15 }
```

Compressed Sparse Row (CSR) Kernel 1 (cont.)

Observations:

- contiguos storage of column and values vectors, but
- kernel does **non-coalesced** memory access

Access patern:

```
ptr                [0  2  4  7  9]

indices            [0  1  1  2  0  2  3  1  3]
data               [1  7  2  8  5  3  9  6  4]

Iteration 0        [0     1     2        3    ]
Iteration 1        [   0     1     2        3]
Iteration 2        [                  2      ]
```

Compressed Sparse Row (CSR) Kernel 2

tbd. next session

ELLPACK (ELL) Kernel

Straightforward approach: each thread multiplies one row with the vector.

```
1  __global__ void k_ell_mat_vec_mm (N, int num_cols_per_row,
2                                    indices, data, x, y) {
3    int row = blockDim.x * blockIdx.x + threadIdx.x ;
4
5    if ( row < N ){
6      float dot = 0;
7      for ( int n = 0; n < num_cols_per_row ; n ++){
8              int col = indices [ N * n + row ];
9              float val = data [ N * n + row ];
10             if ( val != 0)
11                     dot += val * x [ col ];
12     }
13     y [ row ] += dot ;
14   }
15 }
```

ELLPACK (ELL) Kernel 1 (cont.)

Observations:

- kernel **does** coalesced memory access
- **x** vector might not necessarily be accessed contiguosly

Access patern:

```
data                [1  2  5  6  7  8  3  4  *  *  9  *]
indices             [0  1  0  1  1  2  2  3  *  *  3  *]

Iteration 0         [0  1  2  3                        ]
Iteration 1         [          0  1  2  3              ]
Iteration 2         [                      0  1  2  3]
```

Assignment 1 : Page Rank

- load M (matrix market format) matrix from disk (eg. *flickr.mtx*)
  `http://www.cise.ufl.edu/research/sparse/MM/Gleich/flickr.tar.gz`
- transform it from COO to CSR
- apply pagerank on M
  - transpose M (during COO to CSR transformation)
  - make M stochastic
  - multiply using CSR kernel
  - requlatize
  - stop on error criterium

Tips

- do on paper several COO to CSR transformations to make sure you understand the format
- write your own very small examples to test
- check for CUDA errors after **each** CUDA call

Assignment 2 : Solve 1D heat equation

tbd. next session

Literature

Nathan Bell and Michael Garland
Efficient Sparse Matrix-Vector Multiplication on CUDA.
2008.