GPU Computing with CUDA
Lab 6 - Cusp

Christopher Cooper
Boston University

August, 2011
UTFSM, Valparaíso, Chile

Objectives

- Experiment with Cusp
- Outline
  - Implement a Finite Difference solution of Poisson equation
    - Try different
      - Solvers
      - Preconditioners
      - Memory spaces
  - Implement a sparse matrix vector with CSR and ELL

Cusp - Poisson problem

- See end of lecture 7 for details in derivation of the Poisson problem in FD
- Solvers:
  - GMRES, CG, BiCG-stab
- Preconditioners
  - Nothing, diagonal, smoothed aggregation
- Memory spaces
  - Host, device
- Use monitors!

Cusp - Poisson problem

```
// Generate matrix
cusp::coo_matrix<int, float, cusp::host_memory> A;
cusp::gallery::poisson5pt(A, N-2, N-2);

// Generate RHS, solution vector, and analytical solution
cusp::array1d<float, cusp::host_memory> b(A.num_rows, 1.0f);
cusp::array1d<float, cusp::host_memory> u(A.num_rows, 0);
cusp::array1d<float, cusp::host_memory> u_an(A.num_rows, 0);

for (int j=1; j<N-1; j++)
        {       for (int i=1; i<N-1; i++)
                {
                        b[(j-1)*(N-2)+(i-1)] = 8*M_PI*M_PI*sin(2*M_PI*x[j*N
+i])*sin(2*M_PI*y[j*N+i])*h*h;
                        u_an[(j-1)*(N-2)+(i-1)] = sin(2*M_PI*x[j*N+i])*sin(2*M_PI*y[j*N+i])
+ 1.0f;
                        if ((j==1) || (j==N-2))
                        {       b[(j-1)*(N-2)+(i-1)] += 1.0f;
                        }
                        if ((i==1) || (i==N-2))
                        {       b[(j-1)*(N-2)+(i-1)] += 1.0f;
                        }
                }
        }
```

Cusp - Sparse matrix vector

- Do a sparse matrix vector product with CSR and ELL
- CSR

```
1  7  0  0
0  2  8  0
5  0  3  9
0  6  0  4
```

0	2	4	7	9					row offsets
0	1	1	2	0	2	3	1	3	column indices
1	7	2	8	5	3	9	6	4	values

- ELL

```
1  7  0  0
0  2  8  0
5  0  3  9
0  6  0  4
```

values

```
1  7  *
2  8  *
5  3  9
6  4  *
```

rows

entries per row

column indices

```
0  1  *
1  2  *
0  2  3
1  3  *
```

Cusp - Sparse matrix vector

```
float *c_d;
cudaMalloc( (void**) &c_d, N_side*sizeof(float));
thrust::device_ptr<float> c_ptr(c_d);

... //do something on c_d

cusp::array1d<float, cusp::device_memory> c(c_ptr, c_ptr+N_side);
```

```
// array2d from ell comes in column major, need to transpose
// before flatten as raw pointer cast of array2d assumes row major
cusp::array2d <float, cusp::device_memory> At_val;
cusp::transpose(A.values, At_val);

float *val_d = thrust::raw_pointer_cast(&At_val(0,0));
```