CUDA Programming

Outline

- Querying Device

Querying Device

```
int main(void) {
        int count;
         cudaGetDeviceCount( &count);
        return 0;
}
```

Querying Device

Device Property	Description
`char` name[256]	An ASCII string identifying the device (e.g., "GeForce GTX 280")
`size_t` totalGlobalMem	The amount of global memory on the device in bytes
`size_t` sharedMemPerBlock	The maximum amount of shared memory a single block may use in bytes
`int` regsPerBlock	The number of 32-bit registers available per block
`int` warpSize	The number of threads in a warp
`size_t` totalConstMem	The amount of available constant memory

Querying Device

Device Property	Description
`int` maxThreadsPerBlock	The maximum number of threads that a block may contain
`int` maxThreadsDim[3]	The maximum number of threads allowed along each dimension of a block
`int` maxGridSize[3]	The number of blocks allowed along each dimension of a grid
`int` multiProcessorCount	The number of multiprocessors on the device
`int` concurrentKernels	A boolean value representing whether the device supports executing multiple kernels within the same context simultaneously

Querying Device

```
int main(void) {
        int count;
        cudaDeviceProp prop;

         cudaGetDeviceCount( &count);
        for (int i=0; i< count; i++) {
                cudaGetDeviceProperties( &prop, i );
                //Do something with our device's properties
        }
        return 0;
}
```

Querying Device

```
int main(void) {
        int count;
        cudaDeviceProp prop;

         cudaGetDeviceCount( &count);
        for (int i=0; i< count; i++) {
                cudaGetDeviceProperties( &prop, i );
                //Do something with our device's properties
```

Querying Device

```
        printf( "Multiproc: %d\n", prop.multiProcessorCount );
        printf( "Shared mem: %d\n",prop.sharedMemPerBlock );
        printf( "Registers per mp: %d\n", prop.regsPerBlock );
        printf( "Threads in warp: %d\n", prop.warpSize );
        printf( “Threads: %d\n", prop.maxThreadsPerBlock );
        printf( “Max Thread dimensions: (%d,%d,%d)\n",
                prop.maxThreadsDim[0],
                prop.maxThreadsDim[1],
                prop.maxThreadsDim[2] );
        printf( "Max grid dimensions: (%d, %d, %d)\n",
                prop.maxGridSize[0], prop.maxGridSize[1],
                prop.maxGridSize[2] );
    }
    return 0;
}
```

Querying Device

```
int main(void) {
        cudaDeviceProp prop;
        int dev;

        cudaGetDevice( &dev );
        printf( "ID of current CUDA device: %d\n", dev );
        memset( &prop, 0, sizeof( cudaDeviceProp ) );
        prop.major = 1;
        prop.minor = 3;
        cudaChooseDevice( &dev, &prop ) );
        printf( "ID of device closest to revision 1.3: %d\n", dev );
        cudaSetDevice( dev );
        return 0;
}
```