Programming with CUDA

Mihai Niculescu, Ciprian Mihai Mitu
Institute for Space Sciences

Nov 30, 2010 HP-SEE Training, Sofia, Bulgaria

CUDA Review

- Device = GPU
- Kernel = GPU program
- Grid = array of thread blocks
- Thread Block = group of SIMD thread

CUDA API

- Extensions to C
  - To target execution on the device
- A runtime library
  - Built-in vector types
  - Host component - control and access one or more device from the host
  - Device component – device specific functions

Language Extensions Functions Types

- Function type qualifiers
  - __device__ float deviceFunction()
    - No recursion
    - No static variable declarations
    - No variable number of arguments
    - No function pointers
  - __global__ void kernelFunction()
  - __host__ float hostFunction()

Language Extensions Variable Types

- Variable type qualifiers
  - __device__ __shared__ int sharedVar;
  - __device__ int globalVar;
  - __device__ __constant__ int constVar;
  - Pointer can only point to memory in global memory
    - Allocated in the host and passed to the kernel
    - Obtained as the address of a global variable

Language Extensions Execution

- A kernel must be called with an execution configuration:

```
__global__ void KernelFunc(...);
dim3    DimGrid(100, 50);     // 5000 thread blocks
dim3    DimBlock(4, 8, 8);    // 256 threads per block
size_t SharedMemBytes = 64; // 64 bytes of shared memory
KernelFunc<<< DimGrid, DimBlock, SharedMemBytes >>>(...);
```

- Call to kernel function is asynchronous
  - Control returns to CPU immediately

Runtime Library
Built-in Vector Types

- [u]char [1..4], [u]short[1..4],[u]int[1..4], [u]long[1..4],float[1..4]
  - Structures with fields: x,y,z,w
- dim3 ~uint3
  - Used to specify dimension (as for kernel)

Mathematical Functions

- sinf, cosf, tanf, sinhf, coshf, tanhf
- asinf, acosf, atanf, atan2f
- ceil, floor, round, trunc
- powf,sqrtf,expf,logf
- Etc.

Host Runtime Component

- Functions to deal with:
  - Device management
  - Memory management
  - Texture management
  - Interoperability with OpenGL and Direct3D9
  - Error handling
- A host thread can execute device code on only one device

Device Runtime Component

- Mathematical functions (__sin(x), __pow(x,a), etc)
- Atomic integer operations
  - Add, sub,min,max,...
  - And,or,xor,..
  - Increment,decrement
- Texture functions
  - Texture references bound to device memory
- Sync function __syncthreads()

CUDA Libraries

- CUBLAS
  - **CU**DA version for **B**asic **L**inear **A**lgebra **S**ubprograms
- CUFFT
  - CUDA implementation Fast Fourier Transform

Example1: Matrix Addition

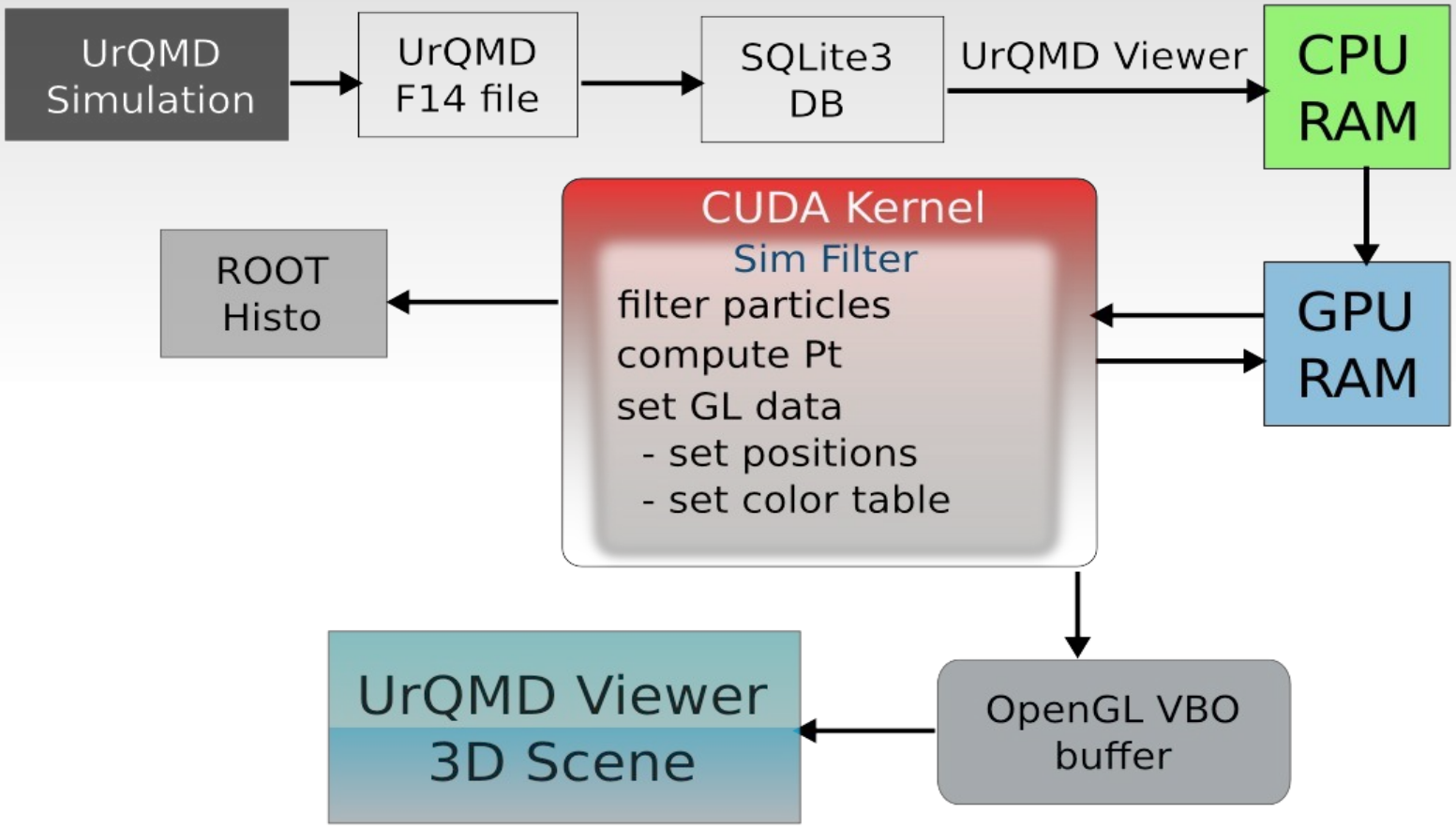
```cpu
// CPU C program

...

void add_matrix(float *a, float *b, float *c, int N)
{
    int i, j, index;

    for(i=0; i<N; i++){
        for(j=0; j<N; j++){
            index = i+j*N;
            c[index] = a[index]+b[index];
        }
    }
}

void main()
{
    ...

    add_matrix(a,b,c,N);

    ...
}
```

```cuda
// CUDA C program

...

__global__ void add_matrix(float *a, float *b, float *c, int N)
{
    int i = blockIdx.x*blockDim.x+threadIdx.x;
    int j = blockIdx.y*blockDim.y+threadIdx.y;

    int index = i+j*N;

    if(i < N && j<N) c[index] = a[index]+b[index];

}

void main()
{
    ...

    dim3 dimBlock(blockSize, blockSize);
    dim3 dimGrid(N/dimBlock.x, N/dimBlock.y);

    add_matrix<<dimGrid, dimBlock>>(a,b,c,N);

    ...
}
```

Example2: HEP Interaction Viewer

- developed at ISS
- online analysis and visualization for the interactions of relativistic nuclear collisions.
- version Alpha
  - Import data – SQLite3 database
  - GUI – Qt4
  - Processing – CUDA
  - Data visualization – ROOT
  - 3D Visualization - OpenGL

HEP Interaction Viewer Diagram



search for UrQMD Viewer