

Programming with CUDA

Mihai Niculescu, Ciprian Mihai Mitu
Institute for Space Sciences

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
 - CUDA version for **Basic Linear Algebra Subprograms**
- CUFFT
 - CUDA implementation Fast Fourier Transform

Example1: Matrix Addition

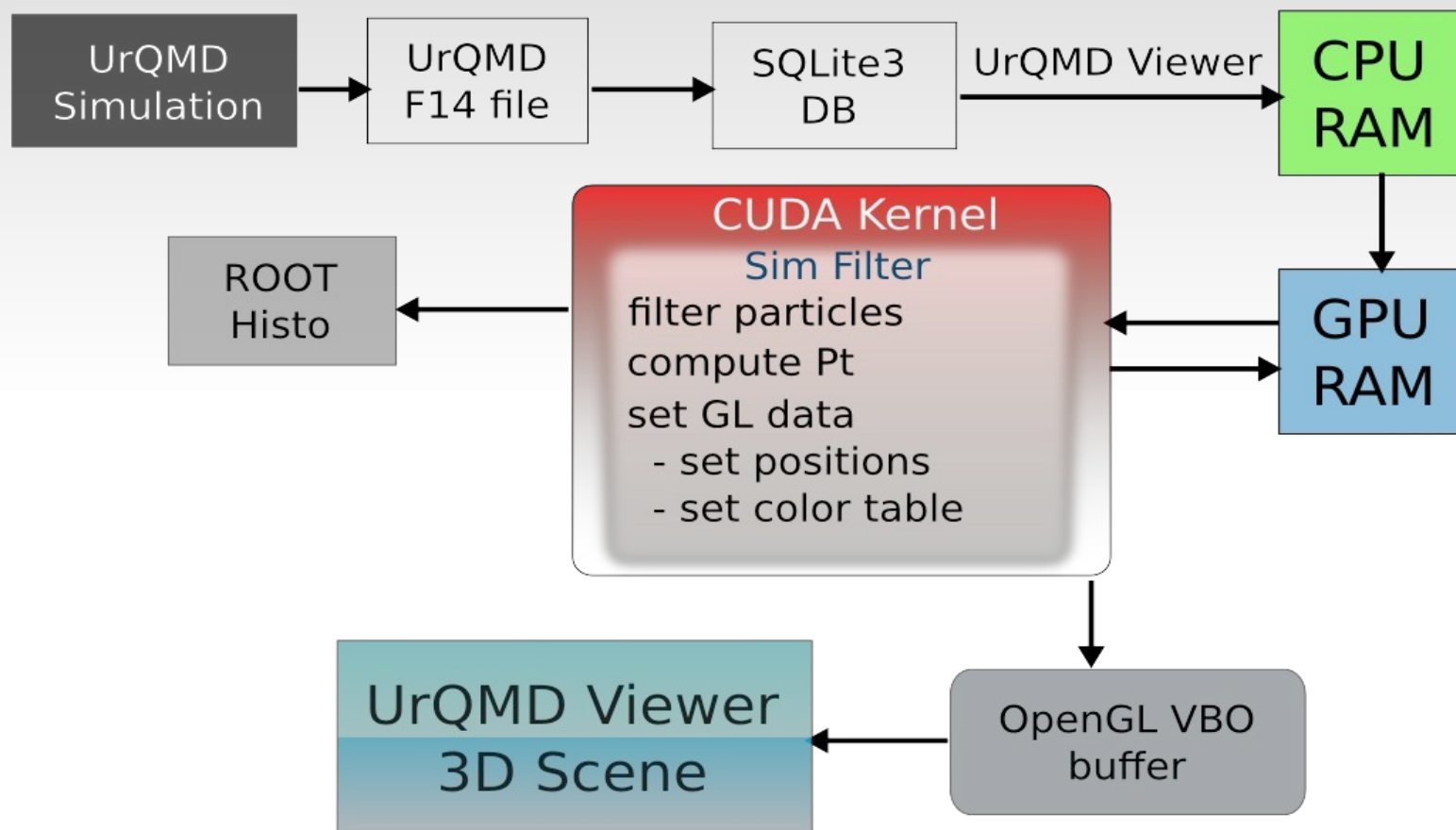
```
1 // CPU C program
2
3 ...
4
5 void add_matrix(float *a, float *b, float *c, int N)
6 {
7     int i, j, index;
8
9     for(i=0; i<N; i++){
10         for(j=0; j<N; j++){
11             index = i+j*N;
12             c[index] = a[index]+b[index];
13         }
14     }
15 }
16
17 void main()
18 {
19     ...
20
21     add_matrix(a,b,c,N);
22
23     ...
24 }
```

```
1 // CUDA C program
2
3 ...
4
5 __global__ void add_matrix(float *a, float *b, float *c, int N)
6 {
7     int i = blockIdx.x*blockDim.x+threadIdx.x;
8     int j = blockIdx.y*blockDim.y+threadIdx.y;
9
10    int index = i+j*N;
11
12    if(i < N && j<N) c[index] = a[index]+b[index];
13 }
14
15
16 void main()
17 {
18     ...
19
20    dim3 dimBlock(blockSize, blockSize);
21    dim3 dimGrid(N/dimBlock.x, N/dimBlock.y);
22
23    add_matrix<<dimGrid, dimBlock>>(a,b,c,N);
24
25     ...
26 }
```

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

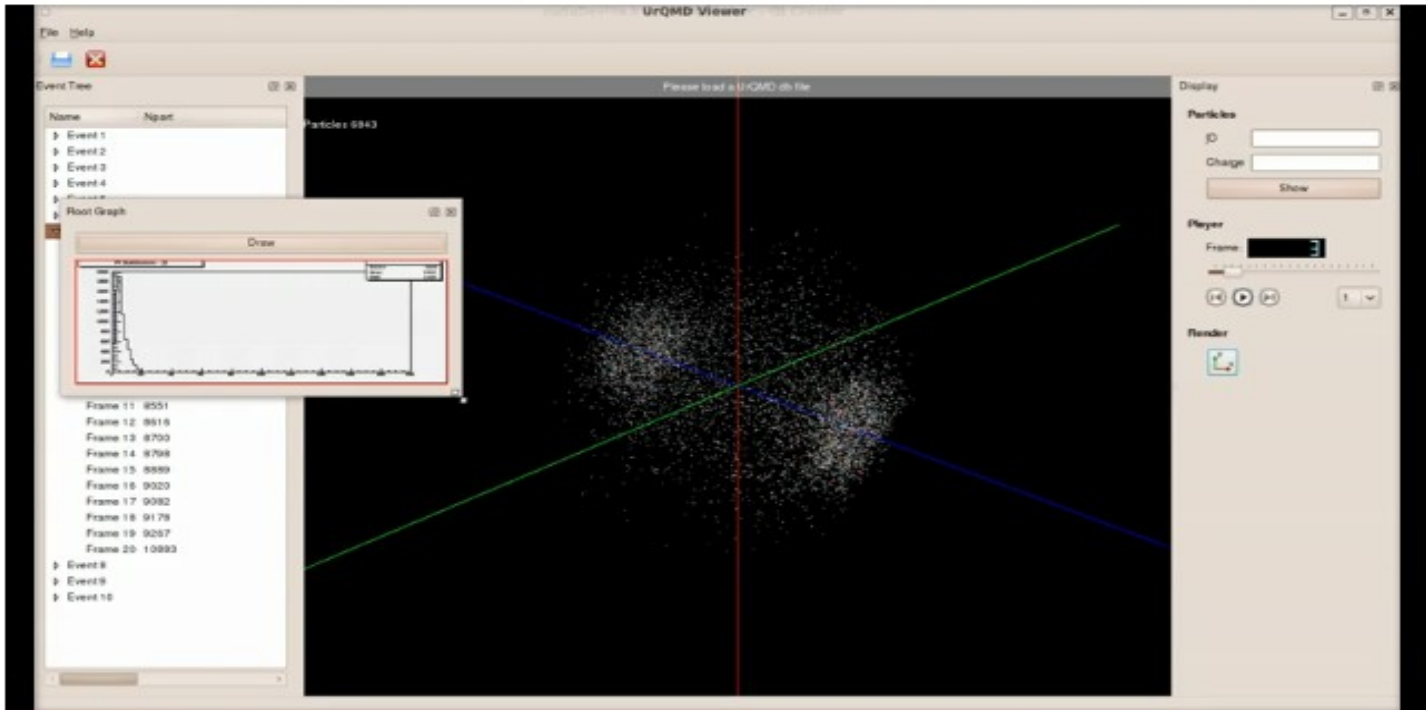


search for UrQMD Viewer

You Tube

UrQMD Viewer - Showing Pt Distribution

Qutork 2 videos



The screenshot shows the UrQMD Viewer interface. The main window displays a 3D visualization of particles (Pt distribution) with a histogram overlay. The histogram shows a distribution of particles across different frames. The interface includes a menu bar (File, Help), a toolbar with buttons for File, Print, and Help, and a status bar at the bottom showing the current frame (1:02 / 1:25) and resolution (360p). The histogram data is as follows:

Frame	Count
Frame 11	8551
Frame 12	8616
Frame 13	8700
Frame 14	8768
Frame 15	8889
Frame 16	9020
Frame 17	9082
Frame 18	9178
Frame 19	9257
Frame 20	10893

Suggestions

- [UrQMD Viewer](#) by Qutork 39 views 2:02
- [fire simulation using version 2](#) by chrismauzey 200 views 0:30
- [CUDA Agent Based - Zombie attack i](#) by FebretPository 156 views 1:06
- [CUDA Agent Based - 10000 agents](#) by FebretPository 72 views 1:19
- [Real-Time Rendering](#) by GouletDesigns 19,369 views 1:40
- [CUDA Agent Based - Zombie attack i](#) by FebretPository

Like **93 views**

Qutork | May 20, 2010 | 1 likes, 0 dislikes
An app that uses CUDA, Qt and OpenGL to render frames in real time from a UrQMD

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