Geant4 Geometry on a GPU

- DHRUVA TIRUMALA BUKKAPATNAM

Introduction

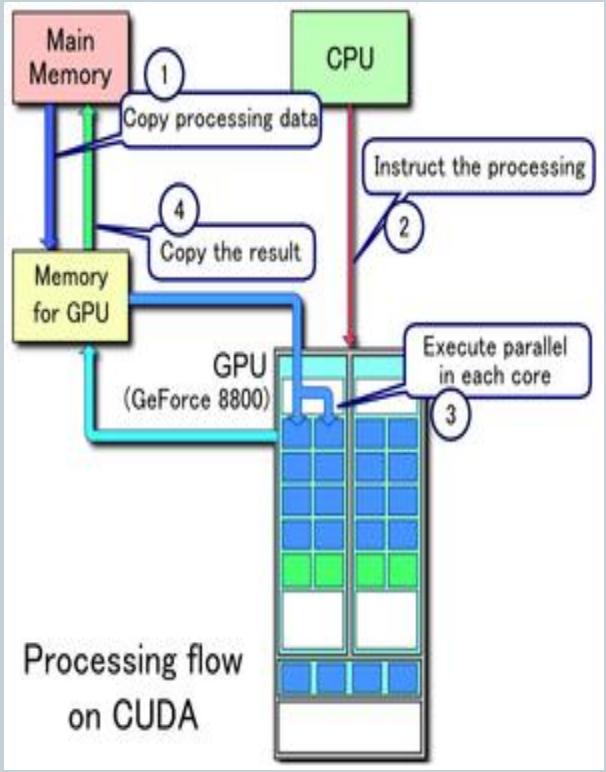
- Outline & goals of project
  - Geant4 for GPU
- Overview of GPUs
  - GPGPU paradigm
  - Stream processing
  - Uses kernels
- Starting point
- Testing
- Progress

GPU limitations

- Avoid Conditionals
  - As far as possible all threads must execute same code
  - Threads dispatched in warps
  - Within a warp, no threads will "get ahead" of any others
- Memory transfer latency
  - Global memory access slow
  - Use shared (cache-like) memory and texture memory.

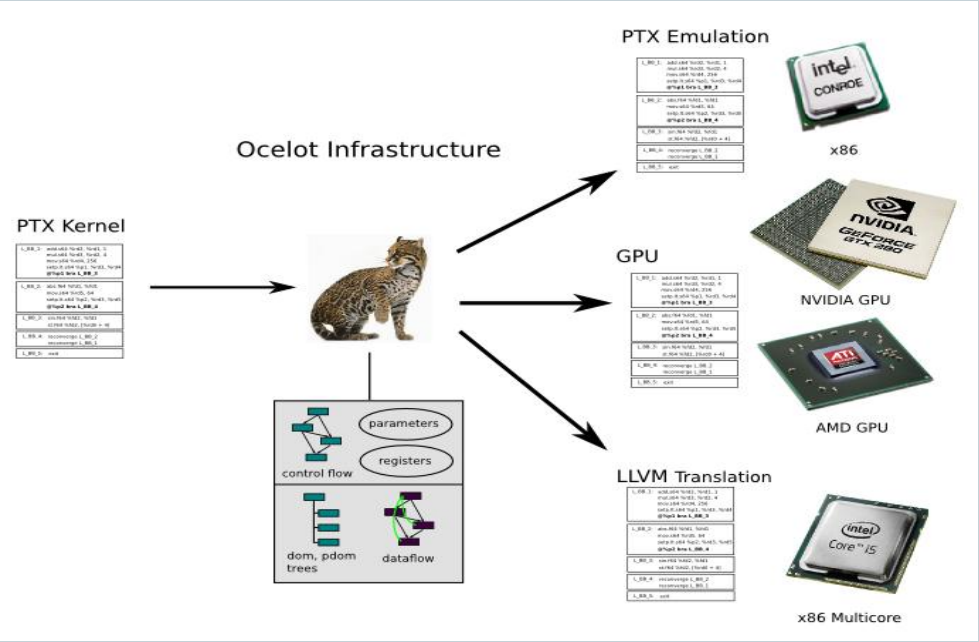
Platform: OpenCL/CUDA

- OpenCL
  - Open source, freely available
  - From the Khronos group
  - Designed to work GPU, CPU and DSP
- CUDA
  - Introduced by nVidia before OpenCL.
  - Easy to understand C like syntax
- Macros
  - Span OpenCL and CUDA – see work of Otto



Platform : Other

- Gpuocelot
  - Emulate CUDA on other platforms
  - Still immature.
  - Potential advantages – No rewrite; Debugging; Speed
- openMP, openACC …



Previous Work: Otto Seiskari (2010)

- Port of core of Navigation exists
  - 5 types of solids: box, orb, tubs, cons, polycone
  - Physics volumes: only placements
  - Has “normal” and “voxel” navigation
  - Defines clones of Geant4 classes through structs.
- Without physics
- Uses Macros to span OpenCL and CUDA

Previous Work: G4VPhysicalVolume

```
typedef struct G4VPhysicalVolume
{
        G4RotationMatrix frot;
        G4ThreeVector ftrans;
        GEOMETRYLOC G4LogicalVolume *flogical;
                                        // The logicalvolume
                                        //   representing the
                                        // physical and tracking attributes of
                                        // the volume
         GEOMETRYLOC G4LogicalVolume   *flmother;
                                        // The current mother logical volume

}
G4VPhysicalVolume;
```

My Work: Goals

- Initially to simulate e- and Gamma particle interactions
  - Two implementations exist
  - We are in touch with the French team and have requested code of gamma & e- physics
- New focus: voxel navigation – critical for HEP
  - to improve the performance of the navigation code.
  - extend the functionality to additional solids

My Work : First steps

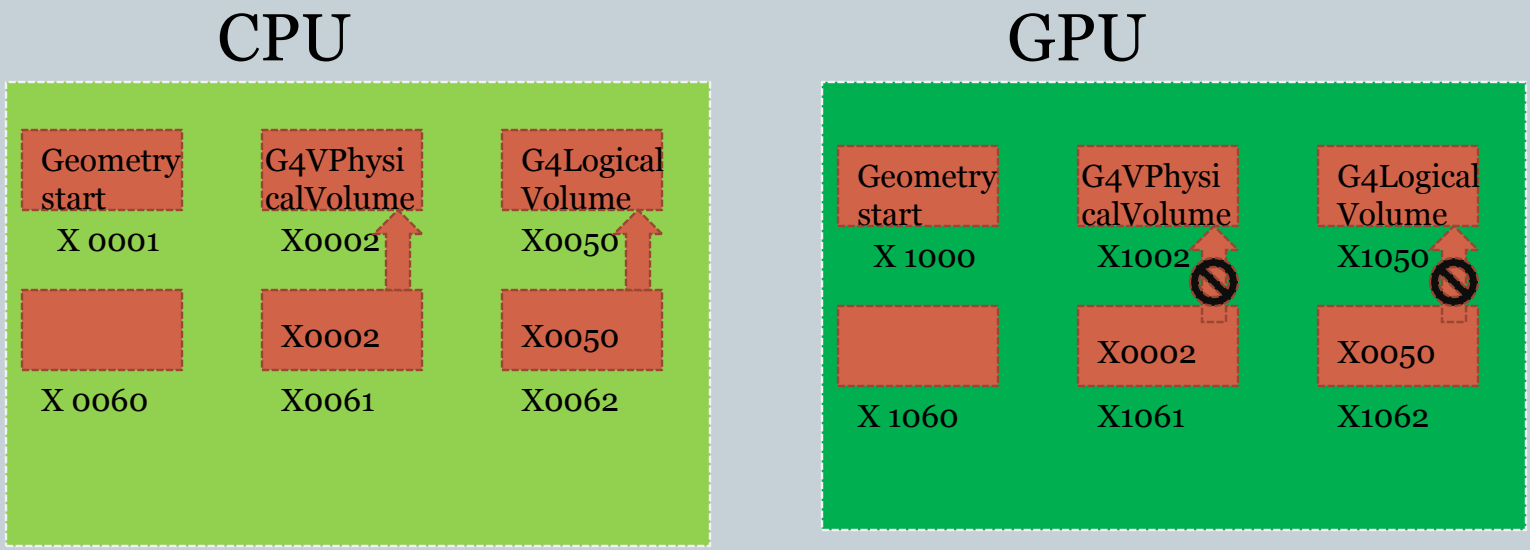
- Get existing code to run
  - Compilation errors
  - System –
    - ATI Mobility Radeon 5700
    - AMD APP SDK 2.7 with OpenCL 1.2 on Ubuntu 12.04
    - Error: kernel arguments can't be declared with types bool/half/size_t/ptrdiff_t/intptr_t/uintptr_t/pointer-to-pointer: __global G4VPhysicalVolume *worldVolumeAndGeomBuffer

My Work : First Steps

- Code ran on another system
  - Difference was in OpenCL version 1.1
- Specifications -:
  - ATI Mobility Radeon 5xxx
  - AMD APP SDK 2.5
  - OpenCL 1.1

My Work : Debugging the code

- Code compiles with OpenCL 1.1
- Problem with pointers on GPU



My Work : Debugging the code

- Solved the problem
  - Relocation on GPU
    - Move pointer offsets
    - Calculate new addresses
  - New way of getting address
    - (int) starting_buffer
    - 64-bit compatibility doubtful
  - Both methods confirmed to work
    - Confirmation by testing

My Work : Creating Tests

- Testing procedure
  - Allocate buffer on GPU
  - Add test integers to struct definition
  - Assign values to these ints on CPU
  - Implement/Modify kernel
  - Move these ints into buffer on GPU
  - Transfer back
  - Compare
- Even with tests, debugging with OpenCL can be hard

My Work : Automating Tests

- Created a set of tests
  - Check for Geometry –
    - Confirm offsets of pointers on GPU
    - Confirm density matches
    - PhysVol->LogicalVol->Solid->Material->Density
  - Check Distance –
    - Basic check.
    - Confirms step == distance moved
- Automated with Macros
- Tests are Solid basis for future improvements

My Work : Optimisation

- Challenge-
  - Avoid overhead of Switch statement for solids (for ‘Virtual’ call)
  - Different threads cannot run different code
    - To get performance all must work on the same type of solid

- **New algorithm-**
  - Threads execute more common code
    - Calculate steps, one solid type at a time.
  - Uses fast local (shared) mem
  - Implemented as a New type of navigation

My Work : Optimisation

- Code tested and run with OpenCL 1.1 with AMD APP SDK.
- Code not yet tested for compatibility with CUDA
- Code still has to be profiled to check for performance gains.

My Work : CUDA v/s OpenCL

- Recently got access to an Nvidia GTX 680 card.
  (courtesy Felice Pantaleo)
- Was able to easily configure CUDA code to run.
- Some discrepancies in results between the two versions.
- Even the legacy versions of the code have this error.
- Normal navigation does not compile for both platforms.

Challenges

- Algorithms may have to be altered
- OpenCL can be challenging.
- CUDA is more C-like
- OpenCL and gpuocelot not easily debugged.

Challenges

- New tools from AMD should help ease the problem
  - Good support for Windows.
  - gDebugger, APP Profiler ..
- Code not tested for
  - Compatibility with newer versions of OpenCL

The way forward

- The next steps for the project-
  - Support more (all?) of Geant4 geometry definition
  - More tests
  - Documentation
  - If the Physics definition from French team can be used, we might be able to run one complete example on the GPU

Thank You

Questions?