FABRIC ENGINE®

Porting Fabric Engine to NVIDIA Unified Memory: A Case Study

Peter Zion
Chief Architect
Fabric Engine Inc.

FABRIC ENGINE®

What is Fabric Engine?

- A high-performance platform for building 3D content creation applications, effects and tools.
  - Optimized native code
  - Parallelism
  - High-end 3D content for media and entertainment
- Applications can be standalone and/or embedded in DCCs (Maya, Soft, 3ds Max, …)

FABRIC ENGINE®

What is Fabric Engine?

- (teaser video: http://vimeo.com/groups/fabric/videos/70421665)

FABRIC ENGINE®

What is Fabric Engine?

- Applications are a combination of Python (or a DCC) and KL
  - Python/DCC: UI, construction of 3D scenes
  - KL: rendering, simulation, effects and data import/export
  - Python/DCC drives execution of KL code

FABRIC ENGINE®

What is Fabric Engine?

- Python applications construct a dynamic 3D scene graph
  - All code is editable
  - KL code is editable at runtime
- 3D scene graph maintains and executes a core dependency graph
  - Data containers and KL operators

FABRIC ENGINE®

The KL Language

- Procedural / object-oriented
- JavaScript-like syntax
- Rich type system
  - Ints, Booleans, Floats, Strings
  - Arrays and dictionaries
  - Structures, Objects and Interfaces
- Pointer-free

FABRIC ENGINE®

The KL Language

- Bindings to third-party libs
  - OpenGL
  - Alembic, Bullet, …
- Rich extension mechanism

FABRIC ENGINE®

The KL Language

- A simple language
  - High-level
  - JITted
  - Accessible to “technical artists”
- A powerful language
  - Fabric Polymesh, RTR code are written in KL

The KL Language

- KL is built on LLVM
  - Targets many platforms
  - Rich optimizations
  - Amazing API
- KL was originally designed with only CPUs in mind
  - Can it target the GPU?

FABRIC ENGINE®

Supporting CUDA GPUs

- Goals
  - Allow most KL code to run without modification on CUDA GPUs
  - Allow KL code on CPU to perform a parallel evaluation of other KL code on GPU
  - Make memory management as easy as possible

Supporting CUDA GPUs

- Challenges
  - KL runtime library in C++
  - Multiple address spaces on GPUs
  - KL is high-level
    - Dynamic memory management
    - Exceptions
    - “Virtual functions”

First Attempt

- Pre-CUDA 6 (Jan-Feb 2013) first attempt
  - Try to manage transfer of data in LLVM IR output from KL compiler
  - Extremely complex, lots of cases not handled well
    - Read-only vs. read-write data
    - Memory with partial writes
    - Need OS/driver support to do it well
  - Lots of progress but had to wait for NVIDIA!

Second Attempt

- Most problems from first attempt are addressed by CUDA 6 unified memory
  - `cuMemAllocManaged` replaces all “manual” work
  - Need to ensure that all data used by both CPU and GPU are allocated through this call
    - Dynamically allocated memory regions (easy)
    - Stack data (slightly less easy)

FABRICENGINE®

PEX Operation

```
operator add<<<index>>>(Vec3 a[], Vec3 b[], io Vec3 c[]) {
  c[index] = a[index] + b[index];
}

operator entry(Vec3 a[], Vec3 b[], io Vec3 c[]) {
  c.resize(a.size);
  add<<<a.size>>>(a, b, c);
}
```

PEX Operation: GPU

```
operator add<<<index>>>(Vec3 a[], Vec3 b[], io Vec3 c[]) {
  c[index] = a[index] + b[index];
}

operator entry(Vec3 a[], Vec3 b[], io Vec3 c[]) {
  c.resize(a.size);
  add<<<a.size@true>>>(a, b, c);
}
```

FABRIC ENGINE®

PEX Operation: Runtime Decision

```
operator add<<<index>>>(Vec3 a[], Vec3 b[], io Vec3 c[]) {
  c[index] = a[index] + b[index];
}

operator entry(Vec3 a[], Vec3 b[], io Vec3 c[]) {
  c.resize(a.size);
  add<<<a.size@(a.size > 1024)>>>(a, b, c);
}
```

Parallel Execute (PEX) Operation

- KL parallel PEX primitive adapted for GPU execution
  - Compiles KL code to GPU kernel (if not cached)
  - Creates “trampoline” from CPU to GPU in CPU code
    - Passes arguments to kernel
      - Shallow argument copy before and after call

FABRICENGINE®

KL Runtime Library

- Originally, KL runtime library was written in C++
  - Not GPU-compatible
- LLVM is very good at inlining
- Entire runtime library was converted into code that builds LLVM IR (compare: libdevice)
  - Effectively, runtime library is now dynamically compiled
  - Very low level, eg. conversion of float to string

Multiple Address Spaces

- GPU differentiates between pointers to local, shared and global memory
- Rewrote KL code generators to account for address spaces
  - If same function is used with two different combinations of pointer type, function is generated twice
- Need to revisit for virtual functions

FABRICENGINE®

Dynamic Memory Allocation

- KL supports dynamic allocation
  - Internal to certain types
    - Variable-length arrays, strings, dictionaries
  - `cuMemAllocManaged` on CPU
  - Well-known GPU allocation algorithms
    - eg. ScatterAlloc
  - What about mixed allocation?

Dynamic Memory Allocation

```
operator cpuKernel() {
  UInt32 a[][];
  a.resize(4096); // alloc CPU mem
  for (Index i=0; i<4096; ++i) a.resize(i%32); // alloc CPU mem
  gpuKernel<<<4096@true>>>(a);
  a.clear(); // free GPU mem and CPU mem
}

operator gpuKernel<<<index>>>(UInt32 a[][]) {
  a[index].resize(index%64); // free CPU mem, alloc GPU mem
}
```

Dynamic Memory Allocation

- How to manage mixed allocation?
  - Defer incompatible frees
  - GPU kernels atomically append GPU pointers to be freed to a list
  - CPU frees pointers when kernel finishes
  - CPU can free GPU pointers
    - Using either system atomics or a simple mutex

FABRIC ENGINE®

LLVM vs. NVVM

- Originally used LLVM back-end for PTX
  - Stable but slow
- Converted to NVVM
  - A few hours of work
    - Mostly, converting IR to older syntax
  - Result: up to 7x performance improvement for executed kernels

Results

- (show Mandelbrot video)

Results

- “Deep” Mandelbrot set:
  - 23fps GPU vs. 2.1fps CPU
- Deformation in Maya:
  - 24fps vs. 5.1fps
- (K5000 GPU, 4x3.6GHz CPU)

FABRIC ENGINE®

Results

- Paradigm shift for programmatic effects
  - TDs can make run-time changes to GPU code and see the results in real-time

FABRIC ENGINE®

Ongoing Work

- OpenGL interop
  - Tag KL arrays as bound to VBOs
- GPU-to-GPU PEX
- Virtual functions on GPU
- Heuristics for where to run
- Debugger for GPU

Roadmap

- Release with initial support targeted for end of May 2014
- Initial limitations:
  - No support for objects and interfaces
    - However, can still work with their data!
  - No support for GPU-to-GPU PEX

http://FabricEngine.com/