

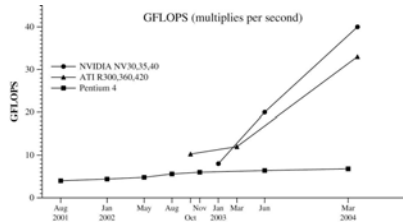


Brook for GPUs



Ian Buck, Tim Foley, Daniel Horn, Jeremy Sugerman, Kayvon Fatahalian, Mike Houston, and Pat Hanrahan

Motivation



Development

- Parallel effort to RStream compiler
- Developed at Stanford
- Brook for GPUs: <http://brook.sourceforge.net>
- Friendly programming environment
 - Programmer need not know GL
- Versions
 - New ATI (420) and NVIDIA (NV40) hardware
 - Linux and Windows
 - DX and OpenGL

In the News...

- open source
 - <http://brook.sourceforge.net>
 - <http://sourceforge.net/projects/brook>
- over 5,600 downloads in 5 months
- 163k page hits
- GPGPU SIGGRAPH Course
- in the news
 - IEEE Computer <http://slashdot.org>
 - Linux and Windows <http://opengl.org>
 - <http://ggpu.org>

Brook GPU Language Additions

```
kernel void foo (float a<>,
float b<>, float t,
float array[], iter float n<>,
out float result<>) {
    result = array[a] + t*b + n;
}
```

Gather Streams

- Indirect addressing of stream data allowed inside kernels.
- Argument passed with array syntax
 - float array[]
- Maps to depended texture lookup

```
float a<100>;
float b<100>;
float c<100>;
float array<25>
iter float n<100> = iter(0,
10);
```

Iterator Streams

- Special stream type which are preinitialized with sequential values (1,2,3,4,...).
- iter float s<100> = iter(0.0f, 100.0f);
 - s initialized with 0.0, 1.0, 2.0, ..., 99.0

```
foo (a, b, 3.2f, array, n, c);
```

Stream Operators

- can be implemented with iterator and gather streams
- streamRepeat & streamStride gpu-optimized

```
float a<20>;
float b<5>;
float c<10>;
foo (a, b, c);
```

Kernel Code

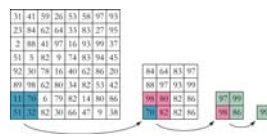
```
kernel void updatepos (float4 pos<>,
float4 vel[100][100],
float timestep,
out float4 response<>) {
    response = pos + vel[pos]*timestep;
}
```

Compiled GPU Code

```
float4 main (uniform float4 updatepos : register (c0),
uniform sampler tex_pos : register (s0),
uniform float4 vel : register (s1),
uniform float4 vel_normalbias : register (s2),
uniform float4 timestep : register (c2);
float4 pos;
float4 response;
pos = tex2D(tex_pos, tex_pos).xyz;
response = pos + tex2D(vel, pos).xyz * timestep;
return response;
}
```

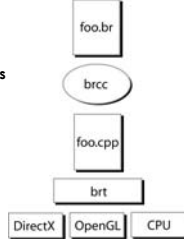
Brook GPU Runtime

- kernel execution
 - Issue screen size quad
 - texture coords provide stream positions
- reductions
 - multi-pass method



BRCC Compiler

- based on ctool
 - <http://ctool.sourceforge.net>
- leverage vendor shader compilers
 - Microsoft: fxc
 - NVIDIA: cgc
- converting kernels into shaders
 - stream fetch and store
 - gather operations
 - register mapping
 - stub function

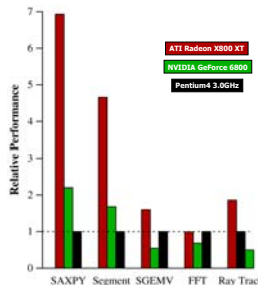


GPU Results

Applications

- ray-tracer
- fft
- segmentation
- linear algebra:
 - BLAS: SAXPY & SGEMV

- compared against:
- Intel Math Library
 - Atlas Math Library
 - CACHED blocked segmentation
 - FFTW
 - SSE-opt Ray Triangle code



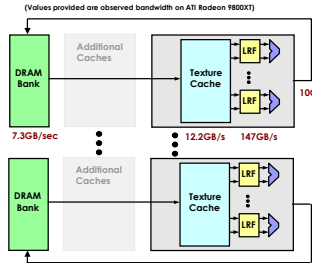
Future Work

GPU Clusters

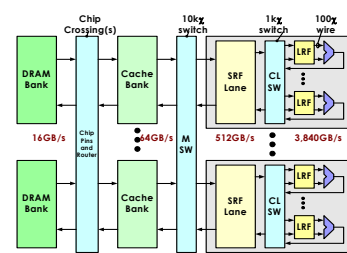
- Compute
 - 32 2.4GHz P4 Xeons
 - 16GB DDR
 - 1.2TB disk
 - Intel E7505 chipset
- Network
 - Infiniband 4X
 - GigE
- Graphics
 - ATI Radeon 9800 Pro 256MB



Modern GPU Bandwidth Hierarchy



Future Merrimac Bandwidth Hierarchy



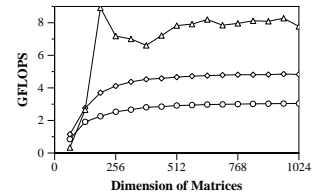
- Cannot exploit producer-consumer locality across kernels (kernel result must go to memory)
- High bandwidth from DRAM compared to modern CPUs
- Low bandwidth from very small local caches

- Exploits locality
 - producer-consumer within kernel in the LRF
 - producer-consumer across kernels in the SRF
- Reduces the distance data travels
- Supports large number of ALUs

Dense Matrix-Matrix Multiplication on the GPU and Pentium 4

Multiplication of 1024x1024 Matrices

	GFLOPS	% Peak	Bandwidth
NV GeForceFX 5900 Ultra	3.04	8%	9.07
ATI Radeon 9800 XT	4.83	18%	12.06
Pentium 4 3Ghz	7.78	65%	27.68



Maximum Observable FLOPS and Bandwidth

	GFLOPS	Cache BW	Seq Read BW
NV GeForceFX 5900 Ultra	39.99	11.08	4.40
ATI Radeon 9800XT	26.14	12.20	7.33

- Near-optimal algorithms cannot approach peak performance on GPUs
 - NVIDIA - 12 math operations required per float fetched from cache in order to obtain peak perf
 - ATI - 8 to 1 math to fetch ratio
- Cache blocking allows CPU to outperform GPUs despite lower DRAM bandwidth and peak FLOPS
- Next generation GPUs will outperform CPU due to greater peak FLOPS and cache bandwidth, but still exhibit low efficiency
- Richer memory hierarchy (faster/larger on-chip caches, SRF) required to keep GPU arithmetic units busy.

Analyzing Matrix-Matrix Multiplication on Merrimac*

- Matrix-Matrix multiplication on Merrimac Simulator achieves 80% of 128 GFLOP peak
 - Fast cross-kernel communication
 - Blocking computation in SRF, Large block sizes
- Hand-tuned Merrimac implementation expected to achieve nearly 95% of peak.

* Merrimac Performance study conducted by Tim Knight.