Running GADGET2 on GPU's:

Optimizing Tree-Search Algorithms
by
Detailed Profiling of GPU Code

Carsten Eie Frigaard

carsten.frigaard@gmail.com

Jena, 15 July 2014

lab4241

GADGET2: Introduction

- GADGET2 — self-gravitating, collisionless, N-body/SPH solver with periodic/non-periodic boundary conditions
- Volker Springel, Max-Planck-Institut für Astrophysik, 2005.
- Hamiltonian

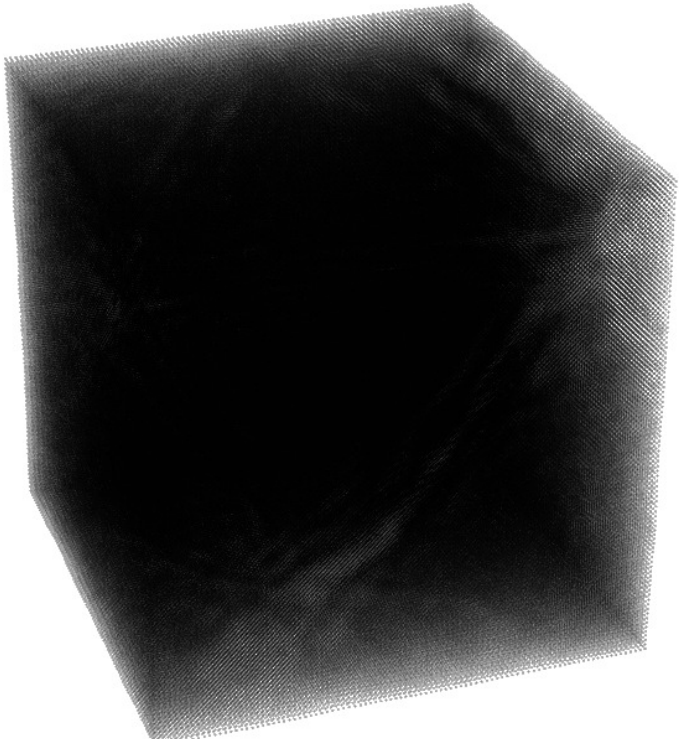
$$H = \sum_i \frac{\mathbf{p}_i^2}{2m_i a(t)^2} + \frac{1}{2}\sum_{ij} \frac{m_i m_j \varphi(\mathbf{x_i} - \mathbf{x_j})}{a(t)}$$

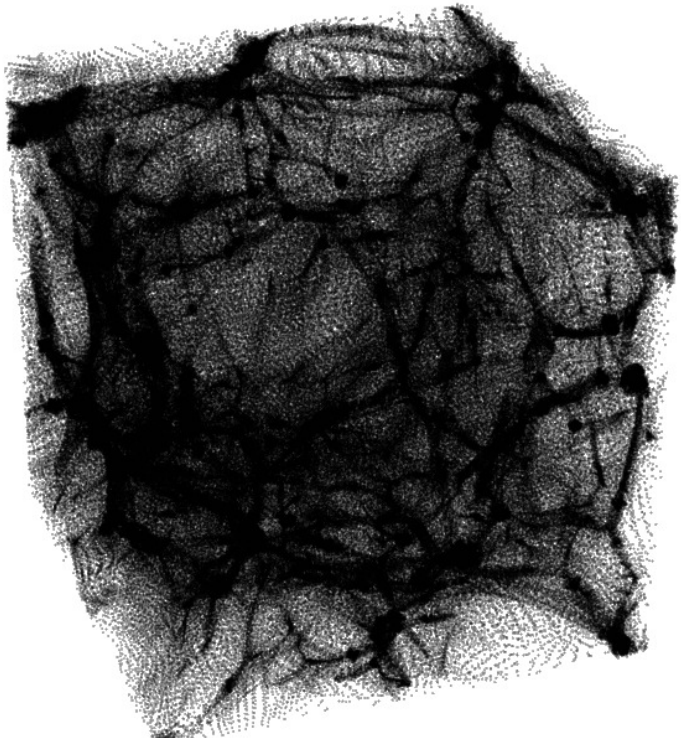
- Interaction potential, φ, solution of

$$\nabla^2 \varphi(\mathbf{x}) = \sum_i 4\pi G(\rho(\mathbf{x}) - \bar{\rho})$$

or in discrete form for a cube with periodic boundary conditions

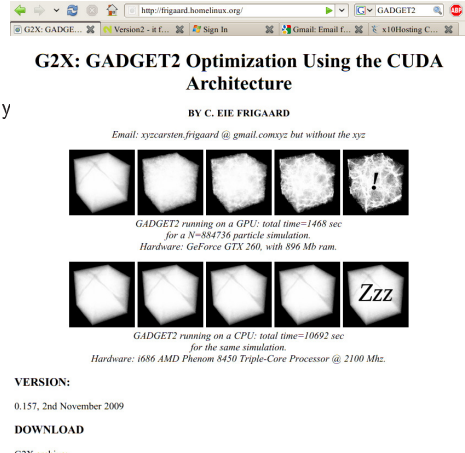
$$\nabla^2 \varphi(\mathbf{x}) = \sum_i 4\pi G \left(-\frac{1}{L^3} \sum_{\mathbf{n}} \tilde{\delta}(\mathbf{x} - \mathbf{n}L) \right)$$



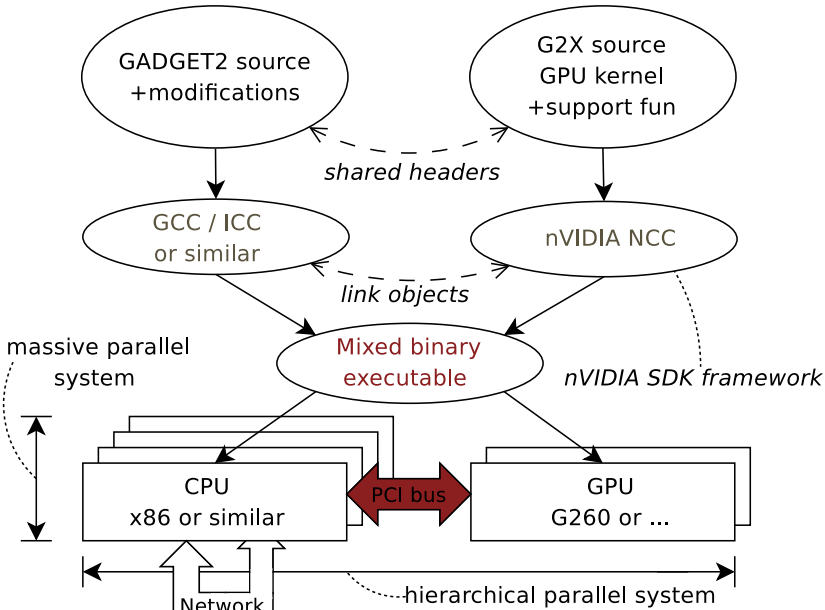


GADGET2/G2X: Introduction

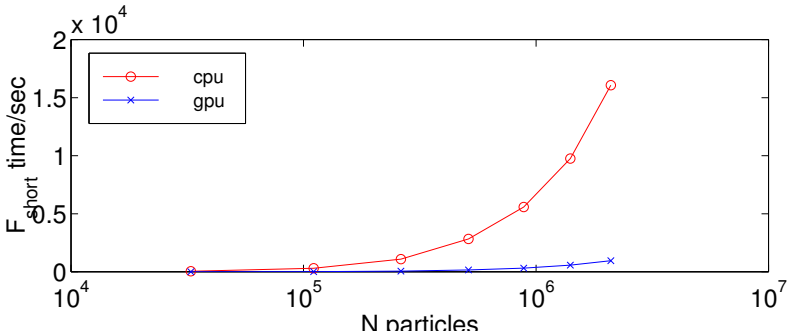
- N-body solver employing hierarchical tree algorithm, $\mathcal{O}(N \log N)$
- Designed as a massive parallel system, runs on many HPC's
- Software stack: MPI, FFTW, GSL, HDF5
- G2X: *GADGET2-eXtreme*, GPU accelerated port
  - Open-source package, GPL3
  - Forcetree/SPH, multi-node/multi-GPU
  - GPU cache (texture), GPU memory hierarchy
  - Warp- and coalescing-'awareness'
  - High register-file pressure
  - Very low GPU memory footprint
- GPUPROF: company lab4241 specializing in GPU optimization

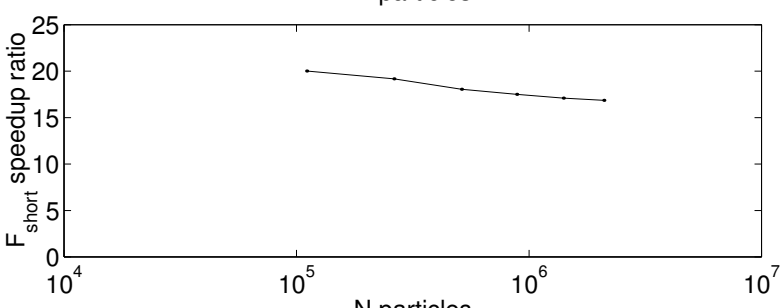


GADGET2/G2X: System Architecture

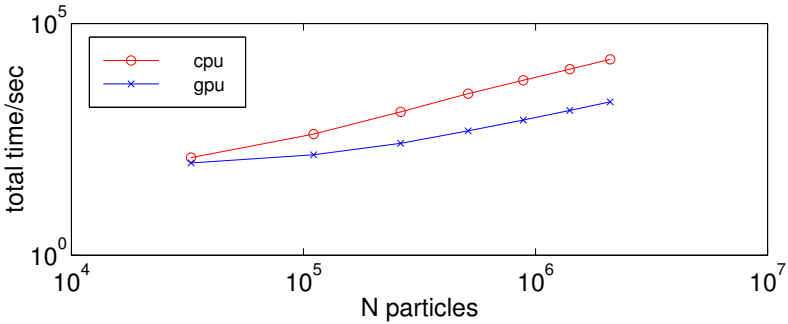


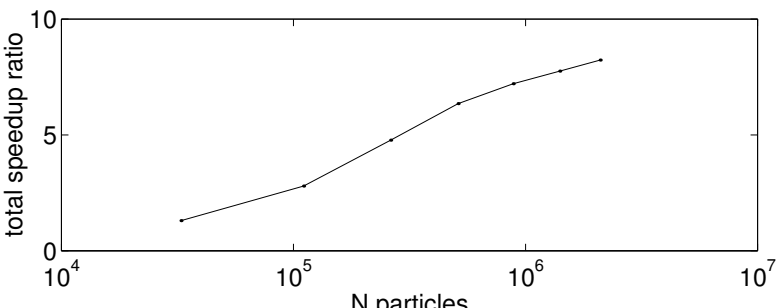
Performance: Local Speedup



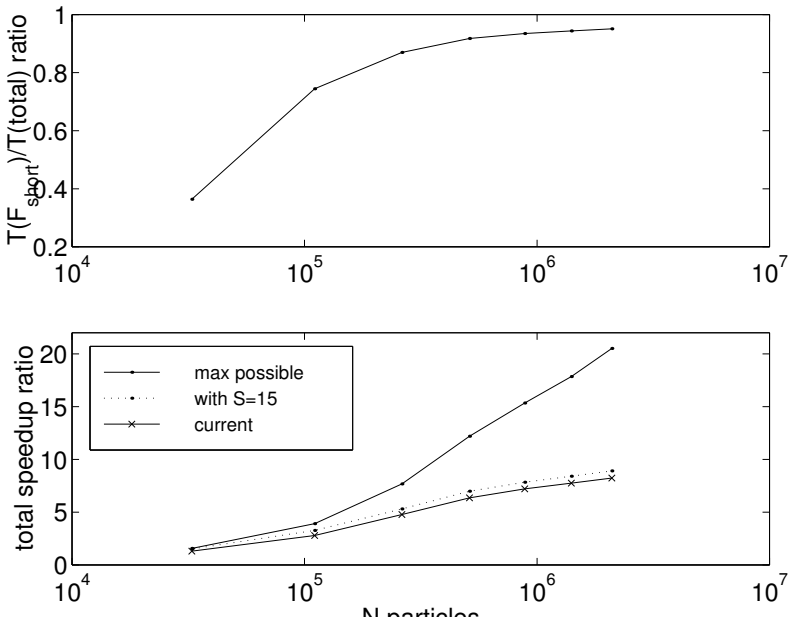


Performance: Global Speedup

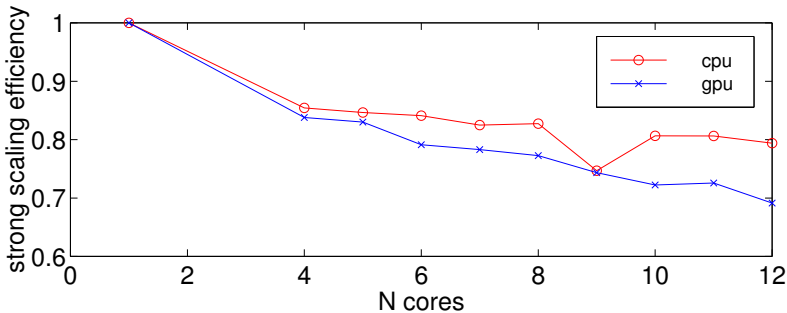


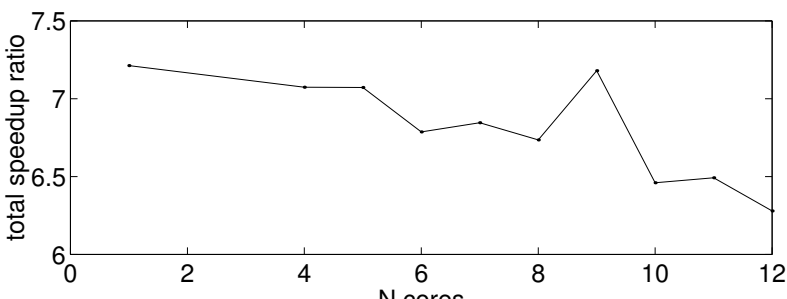


Performance: Speedup and Amdahl's law



Performance: Scaling

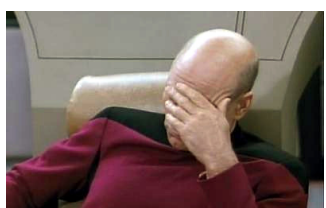




GADGET2 Analysis

Localize bottlenecks / *Analyze* / Re-implement.

- Reverse-engineering of tree algorithm
  - traverses tree for each particle, calculates $\mathbf{F}_i = \mathbf{F}_{\text{short}} + \mathbf{F}_{\text{far}}$
  - `force_treeevaluate_shortrange()`, 346 lines.
  - Matrix-matrix multiplication, $\sim$ 5 lines.
  - highly optimized code, heavy use of preprocessor, "un-rolled" functions.
- *Highly irregular algorithm, complicated memory-access pattern, data-dependent execution path!*



Tree-traversal in G2X: From Micro-benchmarking to an Guided Optimization Process

- Trial-and-error process of optimizing
- Large set of hardware constraints to optimize for
- Mutually opposing optimization constraints
- Tedious, time consuming manual process
- Diminishing marginal return of development time
- Need for a GPU-profiler tool
- $\rightarrow$ Automated, non-intrusive performance profiling of GPU-code

DEMO...

- GPUPROF profiling of:
  - Matrix-matrix Multiplication
  - Gadget2/G2X

DEMO

FILE

SOURCE | PTX | SASS | MEMORY | TEMPORAL | CUCALLS

Kernel	Source	Address	Executed	Exec.scalar	Integer	FP32	Flow	Mov
_Z24MatMulKernel_naive_...								
▼ *info*			*31680*	*1002496*	*28*	*1*	*5*	*5*
name	_Z24MatMulKernel_naive_multi6MatrixS…							
size	19							
maps	6							
time	0.000174046							
estimated_mips	5.8835							
estimated_mflops	132.379							
▼ matdemo.cu@338-342	__global__ void MatMulKernel_naive_m…	0-8;16-…	6656	203776	3		2	
	{							
	volatile const int N=A.width;							
	volatile const int row=threadIdx…							
▼ matdemo.cu@342-345	volatile const int col=blockIdx.…	8-16	2048	65536	1			
	if (row>=N || col>=N) return;							
matdemo.cu@345-346	volatile float t=0;	48-56	32	1024	1			
▼ matdemo.cu@346-349	for(volatile int k=0;k<N;++k) {	56-80;8…	22624	721920	16	1	2	4
	t+=A(row,k)*B(k,col);							
	}							
matdemo.cu@349-350	C.set(row,col,t);	232-304	288	9216	7			1
▼ matdemo.cu@350	}	304-313	32	1024			1	
	void Error(const char*const msg=0)							

DEMO

FILE

SOURCE | PTX | SASS | MEMORY | TEMPORAL | CUCALLS

Kernel	Source	Address	Executed	Exec.scalar	Integer	FP32	Flow	Mov	Memory	S

```
.param .s32 __cudaparm__Z19MatMulKe…
{
        .reg .u32 %r<34>;
        .reg .u64 %rd<29>;
        .reg .f32 %f<52>;
        .reg .pred %p<4>;
        .shared .align 4 .b8 __c…
        .shared .align 4 .b8 __c…
        .loc       3…
        $LDWbegin__Z19MatMulKern…
        .loc       3…
        cvt.s32.u16 %r1, %ctaid.…
        mul24.lo.s32 %r2, %r1, 1…
        cvt.s32.u16 %r3, %ctaid.…
        ld.param.s32 %r4, [__cud…
        mul.lo.s32 %r5, %r3, %r4;
        mul.lo.s32 %r6, %r5, 16;
        add.s32 %r7, %r6, %r4;
        sub.s32 %r8, %r7, 1;
        cvt.s32.u16 %r9, %tid.x;
        cvt.s32.u16 %r10, %tid.y;
```

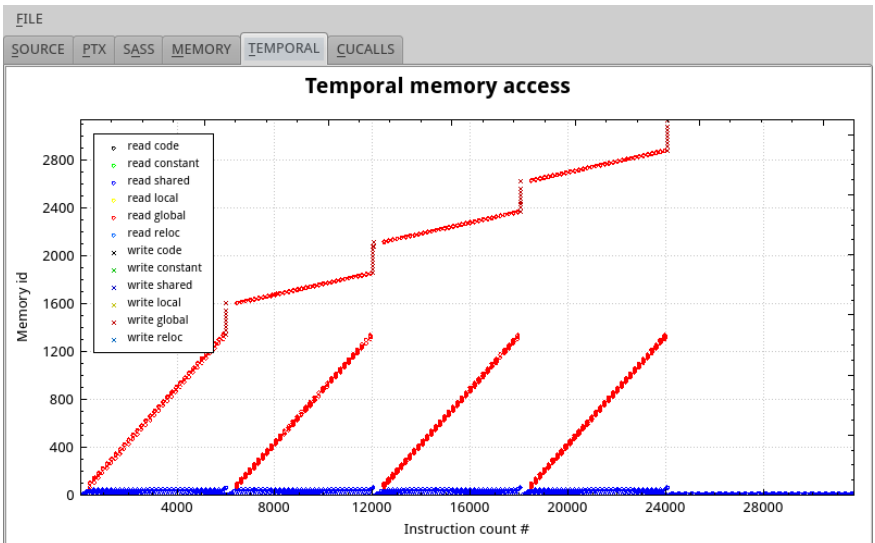
DEMO

FILE

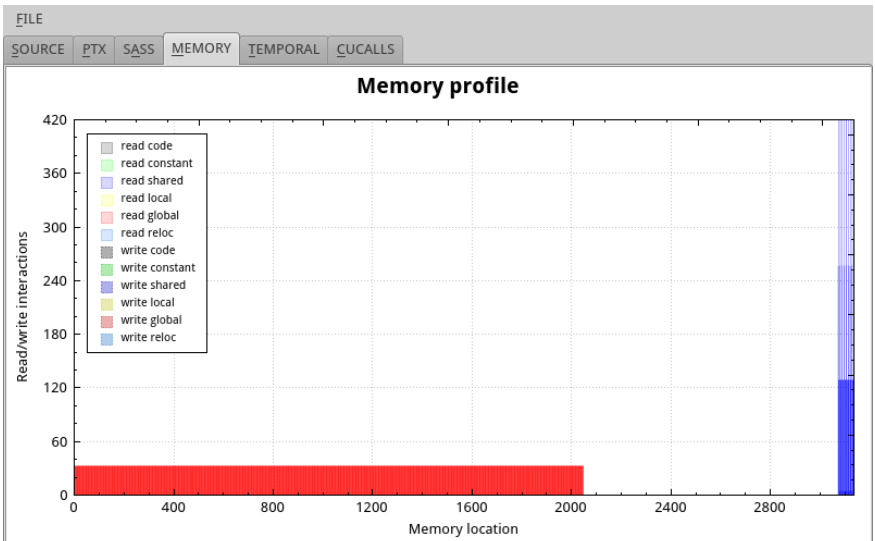
SOURCE | PTX | SASS | MEMORY | TEMPORAL | CUCALLS

Kernel	Source	Address	Executed	Exec.scalar	Integer	FP32	Flow	Mov	Memory	Shared	Cons
./matdemo.multi.e...											
	I2I.U32.U16 R2; R0L		2048	65536	1						
	ISET.S32.C0 o [0x7f]; g […	8	2048	65536	1					1	
	I2I.U32.U16 R4; g [0x6].U…	16	2048	65536	1					1	
	RET C0.NE	24	2048	57344			1				
	ISET.S32.C0 o [0x7f]; g […	32	256	8192	1					1	
	RET C0.EQU	40	256	7168			1				
	ISET.S32.C0 o [0x7f]; g […	48	32	1024	1					1	
	MOV R5; R124	56	32	1024				1			
	MOV R3; R124	64	32	1024				1			
	BRA C0.NE; 0xe8	72	32				1				
	MOV32 R1; g [0x6]	80	1024	32768				1		1	
	MOV32 R0; g [0xe]	84	1024	32768				1		1	
	IMUL32.U16.U16 R7; R2L; R…	88	1024	32768	1						
	IMUL32.U16.U16 R6; R3L; R…	92	1024	32768	1						
	IMAD.U16 R7; R2H; R1L; R7	96	1024	32768	1						
	IMAD.U16 R6; R3H; R0L; R6	104	1024	32768	1						
	SHL R7; R7; 0x10	112	1024	32768	1						
	SHL R6; R6; 0x10	120	1024	32768	1						
	IMAD.U16 R1; R2L; R1L; R7	128	1024	32768	1						
	IMAD.U16 R0; R3L; R0L; R6	136	1024	32768	1						
	IADD32 R1; R1; R3	144	1024	32768	1						
	IADD32 R0; R0; R4	148	1024	32768	1						
	SHL R1; R1; 0x2	152	1024	32768	1						

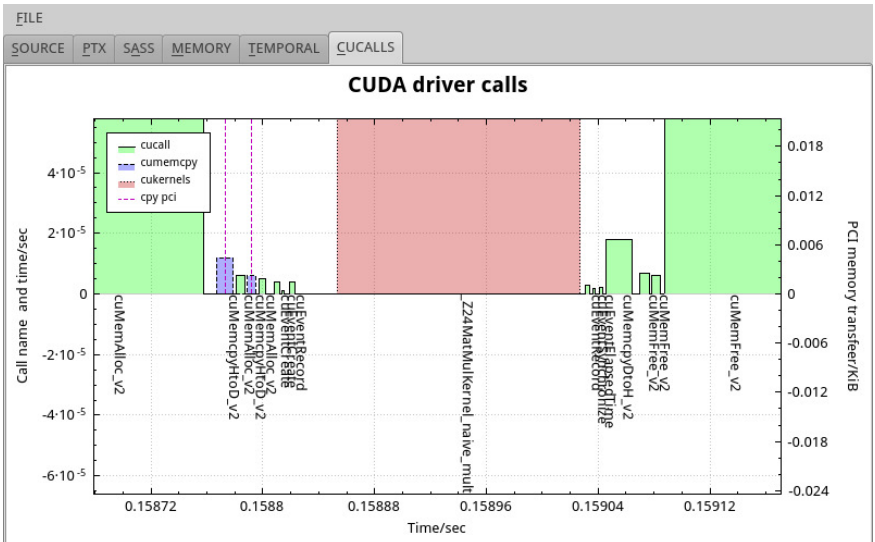
DEMO


FILE
SOURCE
PTX
SASS
MEMORY
TEMPORAL
CUCALLS
Temporal memory access
read code
read constant
read shared
read local
read global
read reloc
write code
write constant
write shared
write local
write global
write reloc
Memory id
0
400
800
1200
1600
2000
2400
2800
4000
8000
12000
16000
20000
24000
28000
Instruction count #

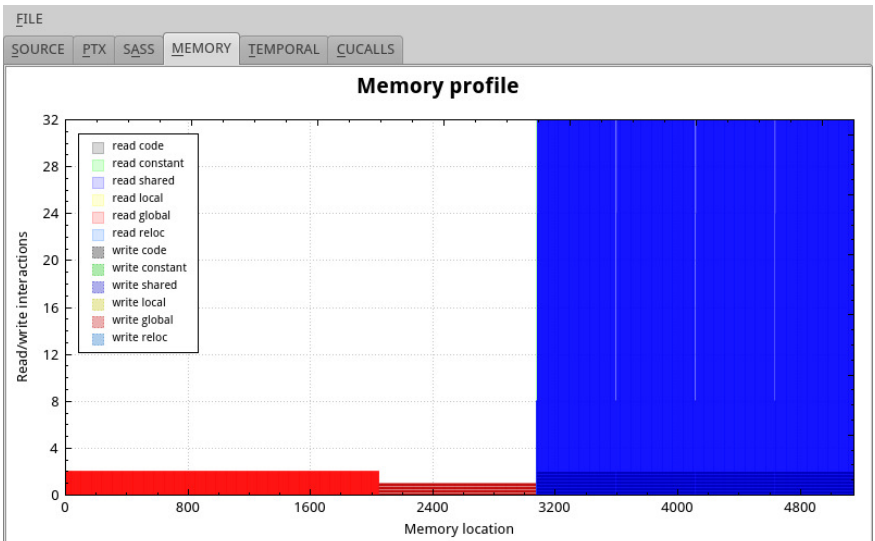
DEMO


FILE
SOURCE
PTX
SASS
MEMORY
TEMPORAL
CUCALLS
Memory profile
read code
read constant
read shared
read local
read global
read reloc
write code
write constant
write shared
write local
write global
write reloc
Read/write interactions
0
60
120
180
240
300
360
420
0
400
800
1200
1600
2000
2400
2800
Memory location

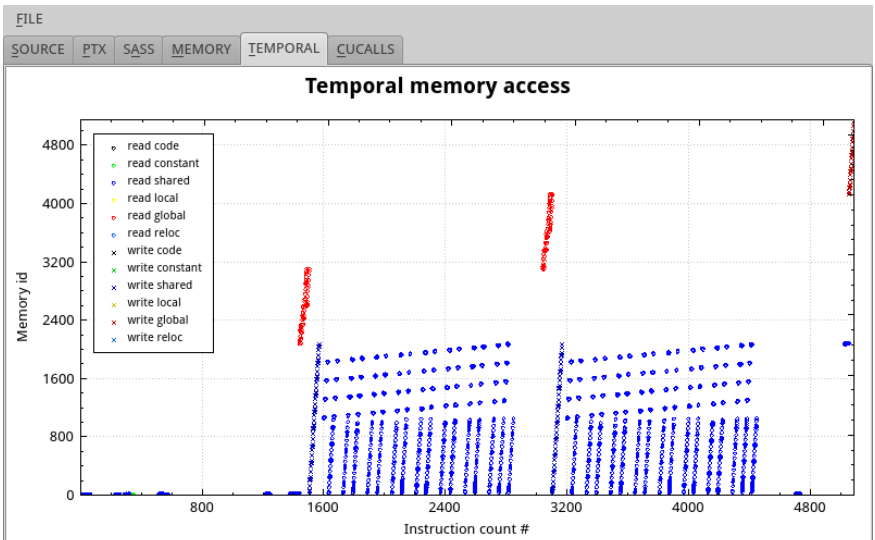
DEMO


FILE
SOURCE
PTX
SASS
MEMORY
TEMPORAL
CUCALLS
CUDA driver calls
cucall
cumemcpy
cukernels
cpy pci
Call name and time/sec
PCI memory transfeer/KiB
Time/sec
cuMemAlloc_v2
cuMemcpyHtoD_v2
cuMemAlloc_v2
cuEventRecord
cuEventRecord
cuEventElapsedTime
Z24MatMulKernel_naive_multi
cuEventElapsedTime
cuEventRecord
cuMemcpyDtoH_v2
cuMemFree_v2
cuMemFree_v2
cuMemFree_v2
4·10-5
2·10-5
0
-2·10-5
-4·10-5
-6·10-5
0.018
0.012
0.006
0
-0.006
-0.012
-0.018
-0.024
0.15872
0.1588
0.15888
0.15896
0.15904
0.15912

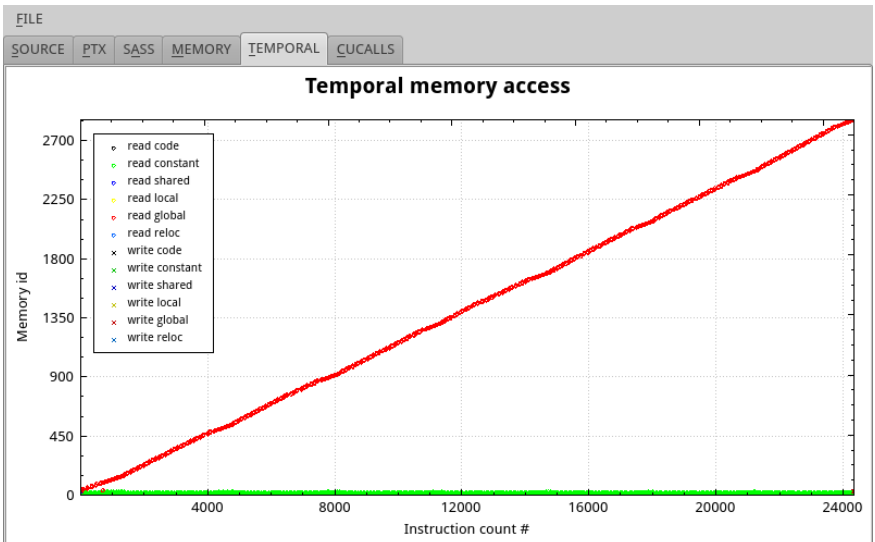
DEMO


FILE
SOURCE
PTX
SASS
MEMORY
TEMPORAL
CUCALLS
Memory profile
read code
read constant
read shared
read local
read global
read reloc
write code
write constant
write shared
write local
write global
write reloc
Read/write interactions
0
4
8
12
16
20
24
28
32
0
800
1600
2400
3200
4000
4800
Memory location

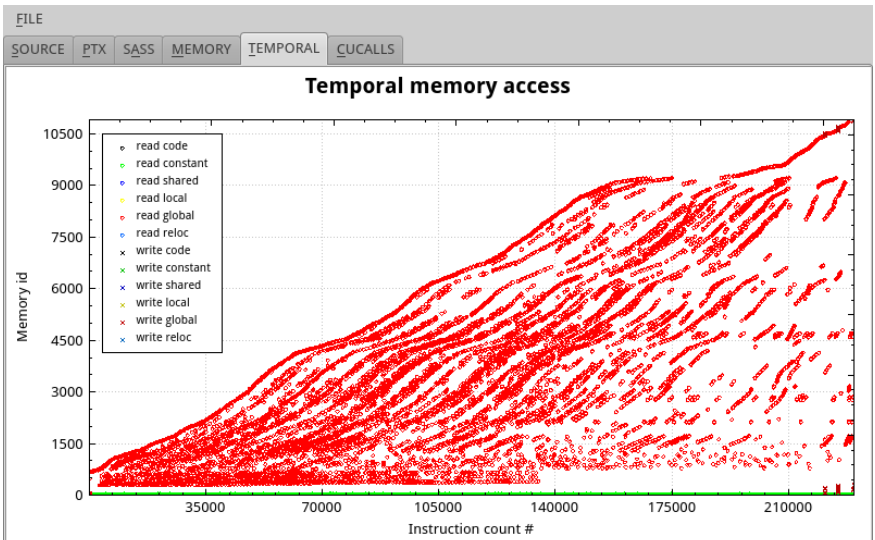
DEMO


FILE
SOURCE
PTX
SASS
MEMORY
TEMPORAL
CUCALLS
Temporal memory access
read code
read constant
read shared
read local
read global
read reloc
write code
write constant
write shared
write local
write global
write reloc
Memory id
0
800
1600
2400
3200
4000
4800
Instruction count #

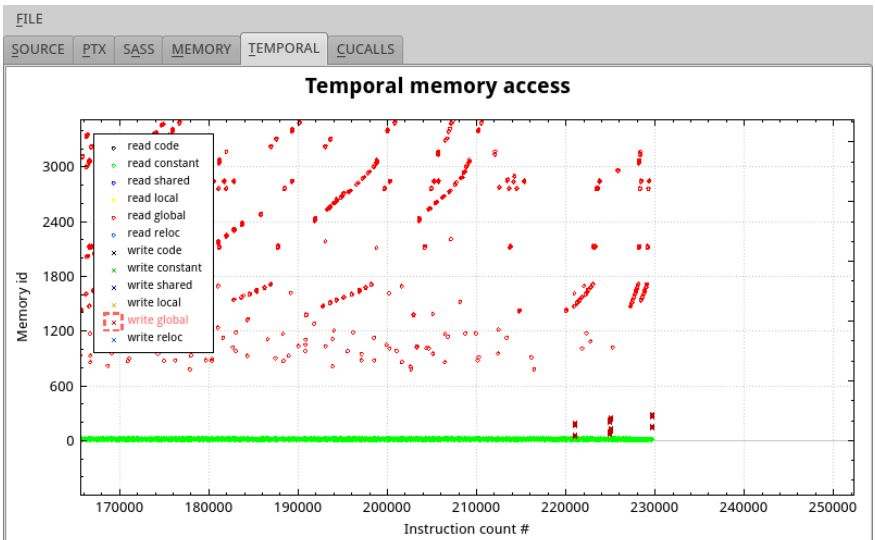
DEMO


FILE
SOURCE
PTX
SASS
MEMORY
TEMPORAL
CUCALLS
Temporal memory access
read code
read constant
read shared
read local
read global
read reloc
write code
write constant
write shared
write local
write global
write reloc
Memory id
2700
2250
1800
1350
900
450
0
4000
8000
12000
16000
20000
24000
Instruction count #

DEMO


FILE
SOURCE
PTX
SASS
MEMORY
TEMPORAL
CUCALLS
Temporal memory access
read code
read constant
read shared
read local
read global
read reloc
write code
write constant
write shared
write local
write global
write reloc
Memory id
10500
9000
7500
6000
4500
3000
1500
0
35000
70000
105000
140000
175000
210000
Instruction count #

DEMO


FILE
SOURCE
PTX
SASS
MEMORY
TEMPORAL
CUCALLS
Temporal memory access
read code
read constant
read shared
read local
read global
read reloc
write code
write constant
write shared
write local
write global
write reloc
Memory id
3000
2400
1800
1200
600
0
170000
180000
190000
200000
210000
220000
230000
240000
250000
Instruction count #

Low-level Optimizations: Divergent Branching in Tree

- Periodic box distance calc optimization:
  - `OLD: x>boxhalf ? x-boxsize : (x<-boxhalf ? x+boxsize : x);`
  - `NEW: return x - boxsize * (int(x>boxhalf) - int(x<-boxhalf));`
- Distance comparison optimization:
  - `OLD: if (dist < -eff_dist || dist > eff_dist)`
  - `NEW: if (fabs(dist)>eff_dist)`
- If and or'ing , what is best:

```
__global__ void test_or()
{
        const int i=threadIdx.x;
        volatile int v=1;
        if (i==42 || i==1)    // 1: logical-or
        //if (i==42 | i==1)   // 2: bitwise-or
        //if (i==42 || i==v)  // 3: using non-const sub-expr.
        // if (int(i==42)+int(i==1)) // 4: arithmetic eval
        {
                g_dbg=3;
        }
}
```

Low-level Optimizations: If and Or'ing

- 1: logical-or

FILE

SOURCE | PTX | SASS | MEMORY | TEMPORAL | CUCALLS

Kernel	Source	Address	Executed	Exec.scalar	Branch tal	Branch not t	Branch mix	In
./or.exe.gpuprof._...								
	I2I.U32.U16 R0; R0L	0x0	2	64				1
	ISET.S32.C0 o [0x7f]; R0; c [0x1] [0x0]; …	0x8	2	64				1
	ISET.S32.C0 o [0x7f] (C0.EQ); R0; c [0x1]…	0x10	2	63				1
	RET C0.EQ	0x18	2	62			2	
	MVC R1; c [0xe] [0x1]	0x20	2	2				
	MVI R0; 0x3	0x28	2	2				
	GST.U32 global14 [R1]; R0	0x30	2	2				
	RET	0x38	2	2	2			
	NOP	0x40						

- 4: arithmetic eval

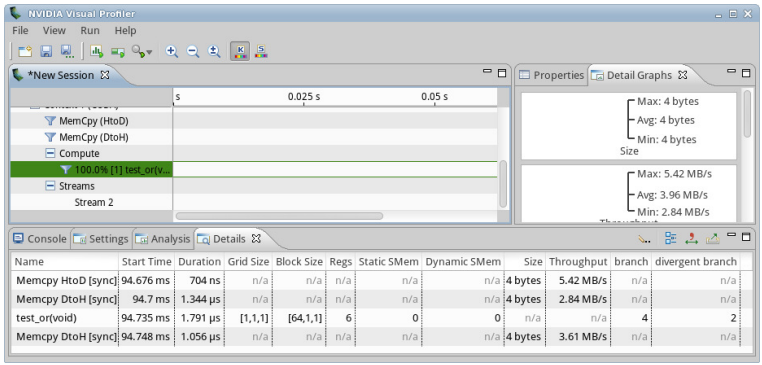
FILE

SOURCE | PTX | SASS | MEMORY | TEMPORAL | CUCALLS

Kernel	Source	Address	Executed	Exec.scalar	Branch tal	Branch not t	Branch mix
./or.exe.gpuprof._...							
	I2I.U32.U16 R0; R0L	0x0	2	64			
	ISET.S32.C0 o [0x7f]; R0; c [0x1] [0x0]; …	0x8	2	64			
	MVC R0 (C0.EQU); c [0x1] [0x0]	0x10	2	1			
	ISET.S32 R0 (C0.NE); R0; c [0x1] [0x1]; EQ	0x18	2	63			
	I2I.S32.S32 R0 (C0.NE); -R0	0x20	2	63			
	ISET.S32.C0 o [0x7f]; R0; R124; EQ	0x28	2	64			
	RET C0.NE	0x30	2	62			2
	MVC R1; c [0xe] [0x1]	0x38	2	2			
	MVI R0; 0x3	0x40	2	2			
	GST.U32 global14 [R1]; R0	0x48	2	2			
	RET	0x50	2	2	2		

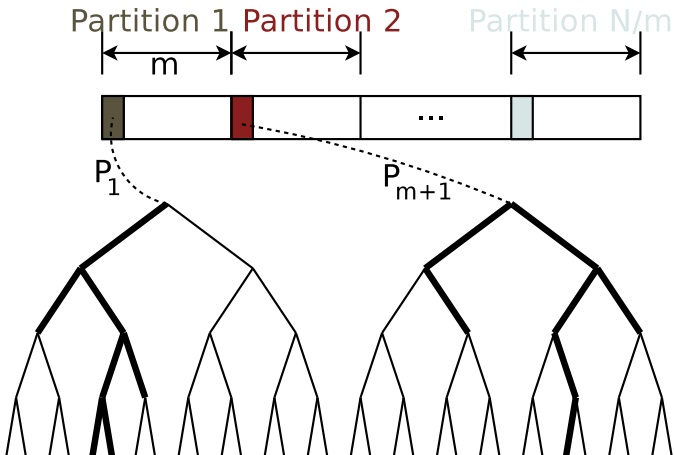
Low-level Optimizations: If and Or'ing

- 4: arithmetic eval in NVVP

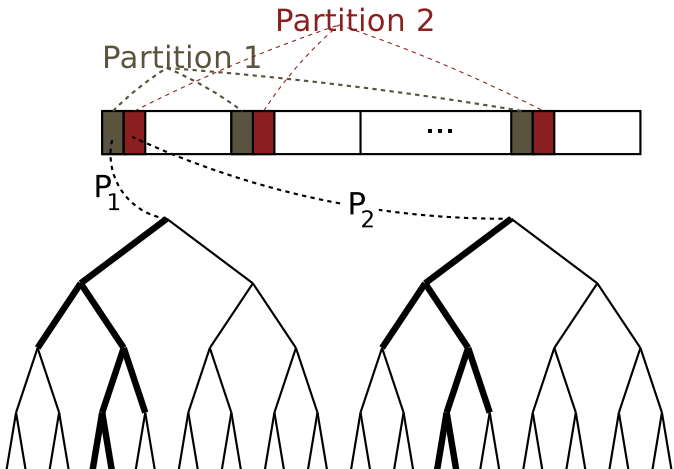


Name	Start Time	Duration	Grid Size	Block Size	Regs	Static SMem	Dynamic SMem	Size	Throughput	branch	divergent branch
Memcpy HtoD [sync]	94.676 ms	704 ns	n/a	n/a	n/a	n/a	n/a	4 bytes	5.42 MB/s	n/a	n/a
Memcpy DtoH [sync]	94.7 ms	1.344 µs	n/a	n/a	n/a	n/a	n/a	4 bytes	2.84 MB/s	n/a	n/a
test_or(void)	94.735 ms	1.791 µs	[1,1,1]	[64,1,1]	6	0	0	n/a	n/a	4	2
Memcpy DtoH [sync]	94.748 ms	1.056 µs	n/a	n/a	n/a	n/a	n/a	4 bytes	3.61 MB/s	n/a	n/a

Medium-level Optimizations: Tree-traversal and Partitioning of Particles

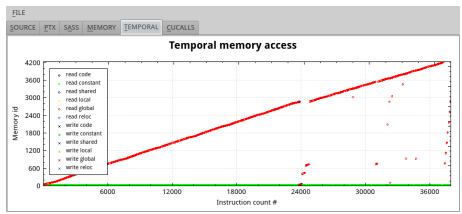


Medium-level Optimizations: Tree-traversal and Partitioning of Particles

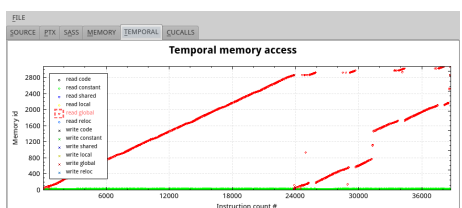


Medium-level Optimizations: Tree-traversal and Partitioning of Particles

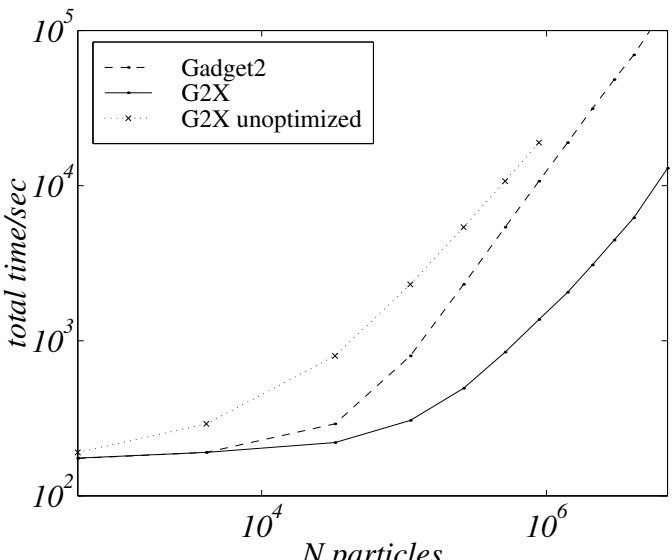
- Force propagation of two particles, p_1, p_2 using one thread:
  - Bad partition, large distance between p_1 and p_2



  - Good partition, small distance between p_1 and p_2



Medium-level Optimizations: Tree-traversal and Partitioning of Particles



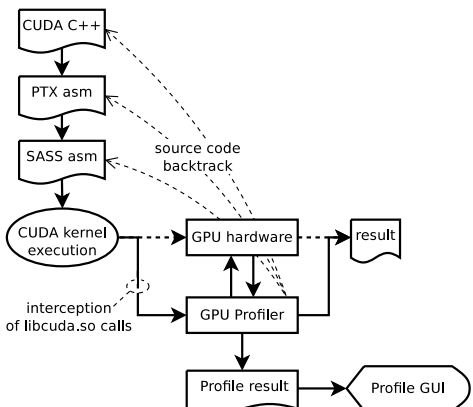
High-level Optimizations: Tree-traversal in General

- Irregular algorithms do generally not fit GPU's
- Development cost per source-line very high
- Use GPUPROF to guide optimizations
- Two general techniques for mapping tree-traversals to GPU's[1]
  - Autoropes: eliminates recursion overhead
  - Lockstepping: promotes memory coalescing
- TODO: use these new methods in GADGET2/G2X

1) General Transformations for GPU Execution of Tree Traversals, Michael Goldfarb, Youngjoon Jo, Milind Kulkarni, ECE Technical Reports, 2011.

GPUPROF: Performance Profiling

- Need for tool to drive down optimization cost
- Help during search of non-continuous optimization-parameter space
- Instrumentation, simulation or hardware profiling registers
- GPUPROF, based on simulation

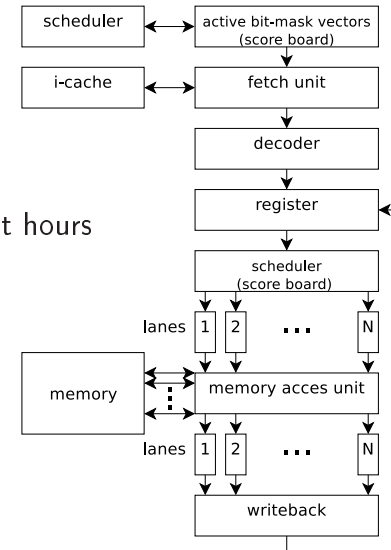


GPUPROF: Performance statistics

- 'Unlimited' performance statistics collection
- Memory-profiling and tracing (debugging)
- Hard-to-get/automate performance data:
  - MIPS, FLOPS
  - GPU memory speed/usage
  - FLOPS to memory intensity
  - coalescing rules
  - branch divergence
  - estimate boundness (ALU, SFU, global memory, PCI bus)
- NOTE: kernel white-box, per-C-source-line profiling!
- Possibility: auto-tuning of kernel parameters

GPUPROF: Software Model

- BARRA: functional simulation of a GPU
- UNISIM: transaction level simulation
- Full GPU SM pipeline model
- Software model expensive in development hours
- Still in α-mode
- Only support limited SM's (1.x) and CUDA versions (4.2)
- Seeking cooperation on GPUPROF
- Free download at `www.lab4241.com`



Resources

- GPUPROF site:
  `http://www.lab4241.com`

- G2X site:
  `http://www.lab4241.com/cef/g2x`

- GADGET2 site:
  `http://www.mpa-garching.mpg.de/gadget`