

Outline Intro and motivation Streaming model **NVIDIA GPUs** Optimization CUDA

GPU/CUDA programming

Dimitar Lukarski

Division of Scientific Computing
Department of Information Technology
Uppsala Programming for Multicore Architectures Research Center
(UPMARC)
Uppsala University

GPUs and UU

Examples Libraries

Programming of Parallel Computers, March, 2014

500

jliji

D. Lukarski, March, 2014, Uppsala



What You Should Know...

Intro and motivation Streaming model

Outline

CUDA NVIDIA GPUs

Optimization Examples

GPUs and UU Libraries

► Programming in C/C++

- Basics in hardware CPU/Accelerators
- Concepts of parallel programming
- To have practice OpenMP, pthreads,...
- Know some metric Amdahl's law, Gustafson's law,...
- Parallel programming is FUN!

D. Lukarski, March, 2014, Uppsala

 $\| f \|_1$



Outline

Introduction and motivation

Streaming programming model

CUDA

Outline Intro and motivation Streaming model NVIDIA GPUs - hardware overview

NVIDIA GPUs

CUDA

Optimization Examples Libraries

Optimization

GPUs and UU

CUDA examples

GPU Libraries

GPUs and Uppsala University

500

jilji

D. Lukarski, March, 2014, Uppsala

UPPSALA UNIVERSITET

Why GP-GPU?

Features

► High performance computing

► High bandwidth

► High parallelism

Intro and motivation Streaming model CUDA NVIDIA GPUs

Architecture

► The largest part of the chip is dedicated for computing

► Smaller section is for local memories

GPUs and UU

Libraries

Optimization Examples ► None (almost) for control logic (ILP, coherence protocols, branching, ...)

Price!

► Low - after all we need to buy this device

► The main market is the game/graphic industry

D. Lukarski, March, 2014, Uppsala



GP-GPU

 $\mathsf{GP-GPU} = \mathsf{General} \; \mathsf{Purpose} \; \text{-} \; \mathsf{Graphical} \; \mathsf{Processing} \; \mathsf{Unit}$

- ▶ It becomes a factor in the HPC world
- ► Main target (numbers) is the Personal Super Computer

Outline Intro and motivation Streaming model

NVIDIA GPUs

CUDA

Optimization Examples Libraries

Comparison to UPPMAX

- ▶ Tintin: 160 × dual Opteron 6220
- ► Each Opteron 6220 = 192 GFlop/s
- ightharpoonup In total (theoretical) = 61.4 TFlop/s

GPUs and UU

Equivalent GPU-based configuration

- \blacktriangleright K20 device = 1.17 TFlop/s
- 53 cards = 62 TFlop/s
- ▶ 14 nodes with 4 GPUs (56 cards)

500

D. Lukarski, March, 2014, Uppsala



UPPSALA UNIVERSITET

Intro and motivation Streaming model

CUDA
NVIDIA GPUS

Optimization Examples

Libraries GPUs and UU

GP-GPU as

- main compute power
- additional compute power
- depends on the algorithm

Main feature of the GPU - Scalable Architecture!

- ► Designed to be scalable
- ► Can perform many many many parallel threads

D. Lukarski, March, 2014, Uppsala



GPU History

UPPSALA UNIVERSITET Outline Intro and motivation Streaming model NVIDIA GPUs GPUs and UU Optimization Examples Libraries CUDA

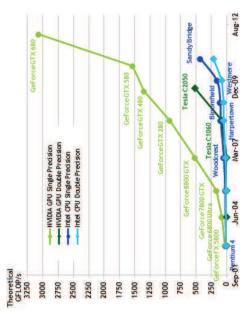


Figure 1 Floating-Point Operations per Second for the CPU and GPU

D. Lukarski, March, 2014, Uppsala

500 php



GPU History

UPPSALA UNIVERSITET Intro and motivation Streaming model CUDA NVIDIA GPUs GPUs and UU Optimization Examples Libraries

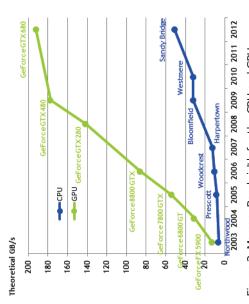


Figure 2 Memory Bandwidth for the CPU and GPU

D. Lukarski, March, 2014, Uppsala



Expectable Speed-ups

Outline Intro and motivation Streaming model

NVIDIA GPUs CUDA

Optimization

GPUs and UU Examples Libraries

It depends on the

- ► Complexity
- ► Data dependency
- Branching
- Locality
- Your programming skills

Pure numbers

- ► Compute performance 4x-10x faster than CPU
- ► Bandwidth 4x-12x faster than CPU

D. Lukarski, March, 2014, Uppsala

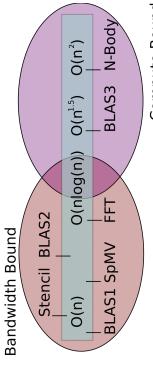
500

UPPSALA UNIVERSITET

Expectable Speed-ups

Intro and motivation Streaming model CUDA
NVIDIA GPUs
Optimization
Examples
Libraries

GPUs and UU



Compute Bound



Expectable Speed-ups

Typicall numbers - 0.1x - 100x

- ► Monte Carlo Methods Outline Intro and motivation Streaming model
 - Molecular Dynamics
- Statistics
- ► Image analysis

NVIDIA GPUs

CUDA

Optimization Examples Libraries

- Dense Linear Algebra
- Sparse Linear Algebra

GPUs and UU

Note that too high/low numbers are typically attributed to bad CPU/GPU implementation!

D. Lukarski, March, 2014, Uppsala

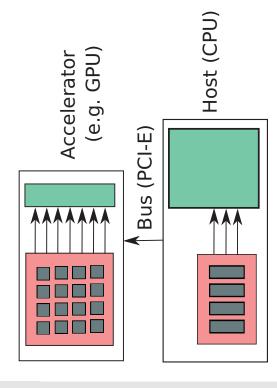
500

jliji



Intro and motivation Streaming model CUDA NVIDIA GPUs

GPU in a Computer



GPUs and UU Optimization Examples Libraries

200

php



CPU - GPU

UPPSALA UNIVERSITET

CPU:

Outline Intro and motivation Streaming model

NVIDIA GPUs CUDA

Optimization

GPUs and UU Examples Libraries

► A few (big/fat) cores ► Big caches (MB) ► General purpose (HDD, Network, Sound ,...)

▶ Parallelism = none (seq) to 2x # cores

GPU:

► Many small cores

► Small caches (KB)

► Optimized for streaming data

► Parallelism = > hundred(s)

D. Lukarski, March, 2014, Uppsala

500 php



Streaming programming model

Intro and motivation Streaming model CUDA NVIDIA GPUs

Outline

GPUs and UU Optimization Examples Libraries

D. Lukarski, March, 2014, Uppsala



Outline Intro and motivation Streaming model CUDA

NVIDIA GPUs

Optimization

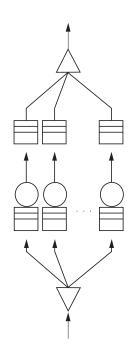
Examples Libraries

GPUs and UU

Overview - fork-join model

You should know

- ► OpenMP/pthreads/MPI
 - ► Fork-join model



D. Lukarski, March, 2014, Uppsala

500 php



Outline

Intro and motivation Streaming model

CUDA NVIDIA GPUs

Optimization Examples Libraries

GPUs and UU

Overview - fork-join model

This allows you to have

- ▶ long and complex threads
- ► locking mechanism / global barriers
 - for all threads
- ► global message passing
 - for all threads

Well, the GPU cannot do that!

D. Lukarski, March, 2014, Uppsala



Outline Intro and motivation Streaming model NVIDIA GPUs GPUs and UU Optimization Examples Libraries CUDA

Example

$$A+B=C,\,A,B,C\in\mathbb{R}^{N,N}$$

► No dependency between the elements

```
for (int i=0; i<N; ++i)
for (int j=0; j<N; ++j)
c[i][j] = a[i][j] + b[i][j];</pre>
#pragma omp parallel for
```

- ► Fork-join model
- ► Similar with pthreads/MPI

D. Lukarski, March, 2014, Uppsala

500 jliji



Streaming Model

Array B Array A Intro and motivation Streaming model CUDA NVIDIA GPUs Optimization Examples Libraries

Outline

► Kernel +

Array C

GPUs and UU

► Implicit indexing

D. Lukarski, March, 2014, Uppsala

 $\| f \|_1$



Outline Intro and motivation Streaming model

NVIDIA GPUs CUDA

Optimization

Examples Libraries

GPUs and UU

Pure Streaming Model

- No data dependency! lack
- No global barriersNo global communication
- ► Very high level of parallelism
- In our A + B = C example For N = 1000, parallelism = 1 M
- ► The origins of this model come from the graphics
 - Every pixel is independent of the other
 Rendering / Ray-tracing algorithms

500

php

D. Lukarski, March, 2014, Uppsala



CUDA

Intro and motivation Streaming model CUDA NVIDIA GPUs

Outline

Optimization Examples

GPUs and UU

Libraries

D. Lukarski, March, 2014, Uppsala



Outline Intro and motivation Streaming model CUDA

NVIDIA GPUs Optimization

GPUs and UU Examples Libraries

General Overview

CUDA is

- ► Language extension (C,C++,Fortran)
- ► Based on stream/data programming model

You need to

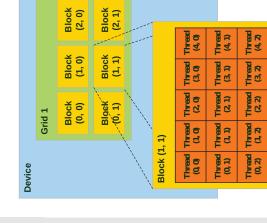
- ► Write a **kernel**
- ► Perform operations wrt the kernel IDs

D. Lukarski, March, 2014, Uppsala

500 jilji



Grid/Thread Blocks/Threads



Intro and motivation Streaming model

Outline

CUDA NVIDIA GPUs

GPUs and UU Optimization Examples Libraries

D. Lukarski, March, 2014, Uppsala

 $\| f \|_1$



Outline Intro and motivation Streaming model

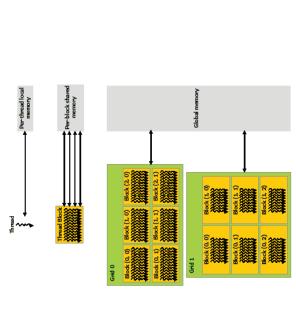
CUDA

Optimization

NVIDIA GPUs

Examples Libraries GPUs and UU

Grid/Thread Blocks/Threads



D. Lukarski, March, 2014, Uppsala

000



UPPSALA UNIVERSITET

Intro and motivation Streaming model

Outline

CUDA

NVIDIA GPUs

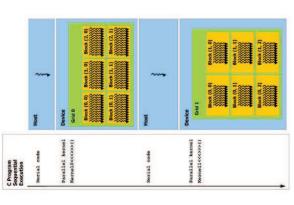
Optimization

Examples

Libraries

GPUs and UU

Grid/Thread Blocks/Threads

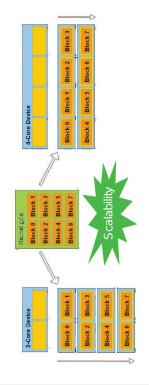




CUDA Model is Scalable!



Outline Intro and motivation Streaming model NVIDIA GPUs GPUs and UU Optimization Examples Libraries CUDA



D. Lukarski, March, 2014, Uppsala

500 jilji



Example: Vector Update

```
x=x+\alpha y, where x,y\in\mathbb{R}^N and \alpha\in\mathbb{R}
```

```
x[i] = x[i] + alpha*y[i];
                                for (int i=0; i< N; ++i)
CPU code:
```

GPU code:

Intro and motivation Streaming model

Outline

CUDA NVIDIA GPUs

Optimization Examples

GPUs and UU

Libraries

```
alpha, const double *x)
                                                                                  int i = blockIdx.x*blockDim.x + threadIdx.x;
                                                                                                                                                                                                                                               vupdate<<<nbr/>blocks, 256>>>(n, y, alpha, x);<br/>D.Lukarski March, 2014. Uposala
                              vupdate(const int n, const double *y,
                                                                                                                                     x[i] = x[i] + alpha*y[i];
                                                                                                                                                                                                                        int nblocks = n/256 + 1;
                                                          const double
--global_- void
                                                                                                               < n)
                                                                                                               if (i
```



Kernel

CUDA Kernel:

UPPSALA UNIVERSITET

Outline Intro and motivation Streaming model NVIDIA GPUs GPUs and UU Optimization Examples Libraries CUDA

► The different path is based on the kernel ID ► All threads run the **same** kernel

Kernel IDs are

► thredIDx - Thread ID ▶ blockIDx - Block ID

▶ **blockDim** - Block dimension

► gridDim - Grid dimension

Function declaration

__global__ - GPU code / called from host

__device__ - GPU code / called from GPU

__host__ - host code / called from host

D. Lukarski, March, 2014, Uppsala

500



Kernel: Example 1/4

UPPSALA UNIVERSITET

```
Intro and motivation
Streaming model
                                  CUDA
NVIDIA GPUs
Outline
```

__global__ void kernel(int *a) {

GPUs and UU Optimization Examples Libraries

D. Lukarski, March, 2014, Uppsala

 $\mu \mu$



Kernel: Example 2/4

UPPSALA UNIVERSITET

```
Output:
Outline
Intro and motivation
Streaming model
                                            NVIDIA GPUs
                                                                                      GPUs and UU
                                                        Optimization
                                                                  Examples
Libraries
                                 CUDA
```

```
int idx = blockIdx.x*blockDim.x + threadIdx.x;
                                                                                                          a[idx] = blockIdx.x*blockDim.x + threadIdx.x;
__global__ void kernel( int *a )
{
                                                                                                                                                                                                                       1 2 3 4 5 6 7 8 9 10 11 12 13 14
```

500

php

D. Lukarski, March, 2014, Uppsala



Kernel: Example 3/4



```
Intro and motivation
Streaming model
                                 CUDA
NVIDIA GPUs
                                                       Optimization
Examples
Libraries
                                                                                           GPUs and UU
Outline
```

```
int idx = blockIdx.x*blockDim.x + threadIdx.x;
--global__ void kernel( int *a )
{
                                                                                                                                            0000111122223333
                                                                       a[idx] = blockIdx.x;
                                                                                                                                                               4 threads per block
                                                                                                                           Output:
```

 $\| f \|_1$



Kernel: Example 4/4

Outline Intro and motivation Streaming model NVIDIA GPUs GPUs and UU Optimization Examples Libraries CUDA

```
500
                                                  int idx = blockIdx.x*blockDim.x + threadIdx.x;
__global__ void kernel( int *a )
{
                                                                                                                                         0123012301230123
                                                                     a[idx] = threadIdx.x;
                                                                                                                                                           4 threads per block
                                                                                                                       Output:
```

D. Lukarski, March, 2014, Uppsala



UPPSALA UNIVERSITET

Outline

Intro and motivation Streaming model

CUDA NVIDIA GPUs

Optimization Examples Libraries

GPUs and UU

Memory Model

Thread:

Registers

► Local memory (not really local)

Thread block:

- ► Shared memory
- ► Low latency a few cycles
- ► High bandwidth

Grid (all threads):

- ► Device memory
- ► Latency 400-600 cycles
- Bandwidth 200 GB/s (K20)

D. Lukarski, March, 2014, Uppsala

 $\| f \|_1$



Memory Model

Registers

Outline Intro and motivation Streaming model CUDA

NVIDIA GPUs

Optimization

GPUs and UU Examples Libraries

► Where – on chip

Scope – thread ► Access – R/W

Lifetime – thread

Local Memory

► Where – off chip

► Access – R/W

Scope – thread

► Lifetime – thread

D. Lukarski, March, 2014, Uppsala

500



Memory Model

UPPSALA UNIVERSITET

Outline

Intro and motivation Streaming model

CUDA
NVIDIA GPUs
Optimization
Examples
Libraries

GPUs and UU

► Where – on chip Shared Memory

- ► Access R/W
 - ► Scope block
- ► Lifetime block

Global Memory

- ► Where off chip
- ► Access R/W
- ► Scope All (dev+host)
- Lifetime application



Memory Model

► Where – off chip Contant Memory UPPSALA UNIVERSITET

Outline Intro and motivation Streaming model

NVIDIA GPUs CUDA

Optimization

Examples Libraries

GPUs and UU

 $\mathsf{Scope}-\mathsf{All}(\mathsf{dev}\!+\!\mathsf{host})$

► Access – R

Lifetime – application

Cached!

Texture Memory

► Where – off chip

Scope - All(dev+host) ► Access – R

► Lifetime – application

Cached!

D. Lukarski, March, 2014, Uppsala

500



Memory Model

Constant Texture Local Global Device To Host Intro and motivation Streaming model CUDA NVIDIA GPUs GPUs and UU Optimization Examples Libraries

Outline

Multiprocessor Multiprocessor

Multiprocessor Registers Shared Memory

Constant and Texture

Caches

D. Lukarski, March, 2014, Uppsala



Memory Allocation UPPSALA UNIVERSITET

Outline Intro and motivation Streaming model NVIDIA GPUs GPUs and UU Optimization Examples Libraries CUDA

```
cudaMalloc((void**) &d_a, n);
int n = 1024*sizeof(int);
                                                                                                                                                              cudaMemset(d_a, 0, n);
                      int* d_a = NULL;
                                                                                                                                                                                                           cudaFree(d_a);
```

500

jilji

D. Lukarski, March, 2014, Uppsala



Memory Moving

```
int n = 256, num_bytes = n*sizeof(int);
                                                                                                                                                               cudaMalloc( (void**) &d_a, num_bytes);
                                                                                                                                                                                                                                                                         cudaMemcpyDeviceToHost);
                                                                                                                                                                                                                                                cudaMemcpy(h_a, d_a, num_bytes,
                                                                                                          h_a = (int*) malloc(num_bytes);
                                                                                                                                                                                            cudaMemset(d_a, 0, num_bytes);
                                                       int *d_a = NULL, *h_a = NULL;
void main() {
                                                                                                                                                                                                                                                                                                                                                        cudaFree(d_a)
                                                                                                                                                                                                                                                                                                                                 free(h_a);
                                                                                       Intro and motivation
Streaming model
```

CUDA NVIDIA GPUs

Outline

GPUs and UU Optimization Examples Libraries

 $\| f \|_1$



Memory Moving

Outline Intro and motivation Streaming model CUDA

NVIDIA GPUs

Optimization

Examples Libraries

GPUs and UU

cudaMemcpyAsync()

cudaMemcpy(h_a,d_a,num_bytes, how); how:

- cudaMemcpyHostToHost
- cuda Memcpy Host To Device
- cuda Memcpy Device To Host

cudaMemcpyHostToDevice

- Asynchronous coping
- Return on the host immediately



500 jilji



Launch a Kernel

- ► Define as __**global**_
- Syntax <<<>>>>
- ► Specify number of threads/number of blocks

Kernel

Intro and motivation Streaming model

Outline

NVIDIA GPUs

CUDA

Optimization Examples

```
= blockIdx.x*blockDim.x + threadIdx.x;
                                              const double *x) {
                       vupdate(const int n, const double *y,
                                                                                                             x[i] = x[i] + alpha*y[i];
                                            const double alpha,
--global -- void
                                                                                         if (i < n)
                                                                                                                                                                        Calling
                                                                     int i
```

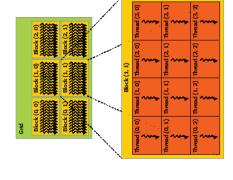
GPUs and UU

Libraries

```
\label{eq:condition} $$\operatorname{vupdate}(n, y, alpha, x);$$ \operatorname{condition}(n, y, alpha, x);$$ \operatorname{condition}(n, y, alpha, x);$$ \operatorname{condition}(n, y, alpha, x);$$
int nblocks = n/256 + 1;
                                                                                                                                                    D. Lukarski, March, 2014, Uppsala
```



Grid/Thread Blocks/Threads



Outline Intro and motivation Streaming model

NVIDIA GPUs

CUDA

Optimization Examples Libraries

GPUs and UU

500

php

D. Lukarski, March, 2014, Uppsala

UPPSALA UNIVERSITET

Outline

Intro and motivation Streaming model

CUDA
NVIDIA GPUs
Optimization
Examples
Libraries

GPUs and UU

Kernel

Kernels:

- ► The control is immediately return to the host
- ► No variable number of arguments
- ► No dynamic declarations (only at compiler time)
- ► Recursion is in CUDA 5

Blocks:

- ► The blocks must be independent
- ► The blocks are executed arbitrary
- No synchronization within the blocks
- ► Independence = Scalability

D. Lukarski, March, 2014, Uppsala



Compiler

UPPSALA UNIVERSITET

Outline Intro and motivation Streaming model

NVIDIA GPUs CUDA

Optimization

GPUs and UU Examples Libraries

Compilation:

- ightharpoonup Add path to the bin/library (32/64)
- ► nvcc program.cu>

Options

- ► Architecture (e.g. 2.0) -arch sm_20
- ► Linking -lcudart, -lcublas, -lcusparse

D. Lukarski, March, 2014, Uppsala

500 php



Intro and motivation Streaming model CUDA NVIDIA GPUs

Outline

NVIDIA GPUs - hardware overview

Optimization Examples Libraries

GPUs and UU

D. Lukarski, March, 2014, Uppsala



Outline Intro and motivation Streaming model CUDA

NVIDIA GPUs

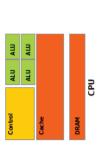
Optimization

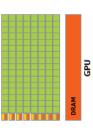
Examples Libraries GPUs and UU

General Overview

GPUs have

- ► More core
- ► Less control units
- ► Less cache/local memories





D. Lukarski, March, 2014, Uppsala

200 php



Scalar Processor



Outline Intro and motivation Streaming model CUDA NVIDIA GPUs

Optimization Examples Libraries

GPUs and UU

SP

D. Lukarski, March, 2014, Uppsala





8x SP

UPPSALA UNIVERSITET

Outline Intro and motivation Streaming model CUDA

NVIDIA GPUs

Optimization Examples Libraries GPUs and UU



D. Lukarski, March, 2014, Uppsala

000



Stream Multiprocessor



Outline
Intro and motivation
Streaming model
CUDA
NVIDIA GPUs
Optimization
Examples
Libraries

GPUs and UU

SFU SFU MT Issue
C-Cache
SP SP
SP SP SP SP SM

D. Lukarski, March, 2014, Uppsala



32x SP

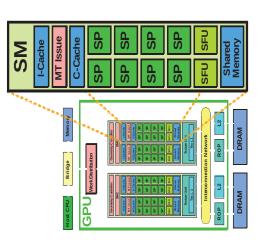
UPPSALA UNIVERSITET

Outline Intro and motivation Streaming model CUDA

NVIDIA GPUs

Optimization

Examples Libraries GPUs and UU



500 jilji

D. Lukarski, March, 2014, Uppsala



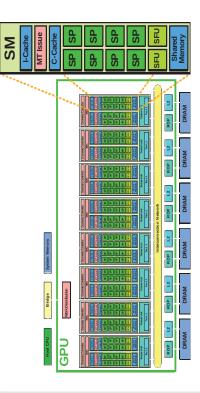
128x SP

Outline

Intro and motivation Streaming model CUDA NVIDIA GPUs

Optimization Examples Libraries

GPUs and UU

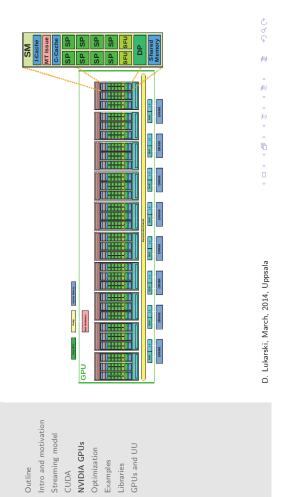


D. Lukarski, March, 2014, Uppsala

ph



240x SP





UPPSALA UNIVERSITET

Kepler GPU has

Outline
Intro and motivation
Streaming model
CUDA
NVIDIA GPUs
Optimization
Examples
Libraries

GPUs and UU

2496 cuda cores!





Outline Intro and motivation Streaming model CUDA

NVIDIA GPUs

Optimization

Examples Libraries

GPUs and UU

Threads are executed on SP

Thread Blocks are executed on SM (physically)

32 Threads = 1 Wrap

16 Threads = 1/2 Wrap execute a single memory

Blocks cannot migrate

Several threads block can be active

One kernel consists of a grid of blocks

One/Multiple kernel can be performed at the same

D. Lukarski, March, 2014, Uppsala

500 php



Intro and motivation Streaming model

Outline

Optimization

CUDA
NVIDIA GPUs
Optimization
Examples
Libraries

GPUs and UU



CUDA Memory

UPPSALA UNIVERSITET

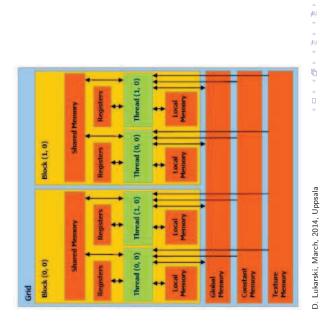
Outline Intro and motivation Streaming model

CUDA

Optimization

NVIDIA GPUs

GPUs and UU Examples Libraries



500

jilji

Shared Memory

UPPSALA UNIVERSITET

Outline

Intro and motivation Streaming model

CUDA NVIDIA GPUs Optimization

Examples Libraries

GPUs and UU

- ► Very low latency
- ► Very high bandwidth
- ▶ 16-48KB per multi-processor
- Shared between 1-8 thread blocks
- Programmer is responsible for data movements (dataracing)
- Possible performance decrease due to bank-conflicts
- The size of the buffer should be determine at compiler time
- int sm_buffer[SIZE]; __shared__



Outline Intro and motivation Streaming model NVIDIA GPUs GPUs and UU Optimization Examples Libraries CUDA

Local Barrier

- ► Threads in a block can synchronize
- ► Thread Blocks cannot synchronize
- Typical usage to avoid dataracing when using shared memory

```
__shared__ int sm_buf[SIZE];
                                                                                                                                             // Ensure all data is copied
                                                                                                                                                                                                                       // Compute data from sm_buff
                                                                       // Load data into sm_buff
// Declare shared memory
                                                                                               Load(buff, sm_buff);
                                                                                                                                                                                                                                                Compute(sm_buff);
                                                                                                                                                                      --syncthreads();
```

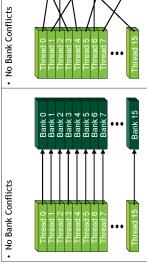
2000

jilji

D. Lukarski, March, 2014, Uppsala



Bank Conflicts



Intro and motivation Streaming model CUDA NVIDIA GPUs

Outline

Bank 0 Bank 1 Bank 2 Bank 3 Bank 4 Bank 6 Bank 6

GPUs and UU

Optimization

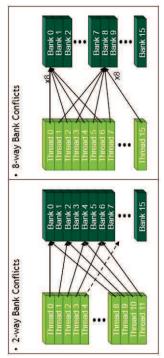
Examples Libraries

••• Bank 15



Bank Conflicts





This example is for devices with lower compute capability

D. Lukarski, March, 2014, Uppsala

500

plipl



Data Transfers

UPPSALA UNIVERSITET

Intro and motivation Streaming model Outline

CUDA NVIDIA GPUs

Optimization Examples Libraries

GPUs and UU

Host-GPU-Host

- Use pinned (non-pageable) memory
- ► Faster (2x) copy over PCI-E
- ► Do not use malloc/new and free/delete
- ► Use cudaHostAlloc/cudaFreeHost

GPU-GPU

- Fermi and Kepler
- Direct transfer with host interaction
- ► Requires new/special bus controller



Memory Access/Alignment

Outline Intro and motivation Streaming model NVIDIA GPUs GPUs and UU Optimization Examples Libraries CUDA

Global memory access

GPU (as CPU) reads memory in segments

32B, 64B, 128B segments

First address = multiple of the segment size

This is called coalescing memory access

Memory access (kernel)

► Memory access is performed in half-wrap

Goal: to have smallest number of transactions

D. Lukarski, March, 2014, Uppsala

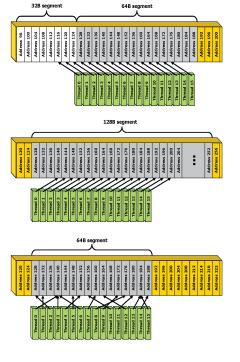
200 jilji

UPPSALA UNIVERSITET

Intro and motivation Streaming model CUDA NVIDIA GPUs

Outline

Memory Access



GPUs and UU

Optimization

Examples Libraries



Branching

▶ i.e. 32 threads work at the same time ► But threads can take different path

► One warp is physically executed

Outline Intro and motivation Streaming model CUDA

NVIDIA GPUs

▶ if-else

Optimization

Examples Libraries GPUs and UU

► This leads to unbalance in the SM

► Avoid if (threadIdx.x % 2) {...} else {...}

► To instead if (threadIdx.x / WARP_SIZE % 2) $\{...\}$ else $\{...\}$

D. Lukarski, March, 2014, Uppsala

500 php



Warp Execution

Marp Scheduler Instruction Dispatch Warp 8 Instruction 43 Warp 8 Instruction 12 Warp 4 Instruction 9 Warp 14 Instruction 9	Warp Schedule	Unit Instruction Dispate	Warp 3 instruction (Warp 15 instruction	 Warp 9 instruction	6 Warp 3 instruction	Warn 15 instruction
	Warp Scheduler	Instruction Dispatch Unit	Warp 2 instruction 42	Warp 14 instruction 95	 Warp 8 instruction 12	Warp 14 instruction 96	Warp 2 instruction 43

Intro and motivation Streaming model

Outline

CUDA
NVIDIA GPUs
Optimization
Examples
Libraries

GPUs and UU

D. Lukarski, March, 2014, Uppsala

1000 H



Hiding Latency

Outline Intro and motivation Streaming model CUDA

NVIDIA GPUs

Optimization

Examples Libraries

GPUs and UU

Switching different warps

► The number of warps depends of the kernel

 $lack High\ register\ usage\ in\ the\ kernel=low\ number\ of$ active warps

Tips

 \blacktriangleright Smaller kernels -> more active warps -> more switching between warps— > better performance

You can check/calculate this occupancy

D. Lukarski, March, 2014, Uppsala

500 jliji



Atomics

Intro and motivation Streaming model

Outline

CUDA
NVIDIA GPUs
Optimization
Examples
Libraries

GPUs and UU

► There is no global synchronization

But there are atomic operations

Add, Sub, Exch, Min, Max, Inc, Dec, CAS

And, Or, Xor

Example

Histograms





Tools

UPPSALA UNIVERSITET

CUDA-GDB

Outline Intro and motivation Streaming model

Debugger

CUDA

cuda-gdb coda-gdb ► Compile with -G -g

► Like GNU debugger

NVIDIA GPUs Optimization

GPUs and UU Examples Libraries

MEM-CHECK

Memory checker

► cuda-memcheck program>

D. Lukarski, March, 2014, Uppsala

500 php



Intro and motivation Streaming model

Outline

CUDA
NVIDIA GPUs
Optimization
Examples

CUDA examples

GPUs and UU

Libraries

D. Lukarski, March, 2014, Uppsala



1D Stencil UPPSALA UNIVERSITET

Outline Intro and motivation Streaming model

NVIDIA GPUs CUDA

Optimization

Examples

GPUs and UU Libraries

- ► Array A, N elements
- Sum the neighboring elements with radius 3
- N inputs, N-3 outputs = 2N



Naive implementation

- ► For all i (1...N), load i-3,i-2,i+1,i,i+1,i+2,i+3
- ightharpoonup 7 Loads + 1 Store = 8N

D. Lukarski, March, 2014, Uppsala

500 jliji

UPPSALA UNIVERSITET

1D Stencil

```
void stencil(const int* input,
                  int* output) {
--global--
```

```
int ix = blockIdx.x*blockDim.x + threadIdx.x;
```

```
int value = 0;
```

Intro and motivation Streaming model CUDA NVIDIA GPUs

Outline

```
for (int i=0; i<=RADIUS, i++)
                                                                            for (int i=1; i<=RADIUS, i++)
                           value += input[ix+i];
                                                                                                     value += input[ix-i];
```

GPUs and UU

Libraries

Optimization Examples

```
= value;
output[ix]
```

D. Lukarski, March, 2014, Uppsala

 $\mu \mu$



1D Stencil

Outline Intro and motivation Streaming model CUDA

NVIDIA GPUs

Optimization

Examples

GPUs and UU Libraries

Use shared memory

- ► Each thread block **blockDim.x** elements
- Each block loads **blockDim.x** + 2×3
- We call this -3 and +3 elements ghost cells
- Synchronize all threads in the block
- Compute the output results
- ► Write the results to the memory



200 jilji



UPPSALA UNIVERSITET

1D Stencil

```
--global_- void stencil(const int *input,
                             int *output){
```

```
__shared__ int s_a[SIZE+2*RADIUS];
```

Intro and motivation Streaming model

Outline

NVIDIA GPUs

CUDA

Optimization Examples

```
= blockIdx.x*blockDim.x
                                       = threadIdx.x + RADIUS;
                    + threadIdx.x;
int global_ix
                                         int local_ix
```

```
= input[global_ix];
s_a[local_ix]
```

GPUs and UU

Libraries

```
\| f \|_1
                                                input[global_ix - RADIUS];
                                                                                                                                        input[global_ix + SIZE + RADIUS];
                                                                         s_a[local_ix + SIZE + RADIUS]
if ( threadIdx.x < RADIUS ){</pre>
                       s_a[local_ix - RADIUS]
                                                                                                                           D. Lukarski, March, 2014, Uppsala
```



1D Stencil

UPPSALA UNIVERSITET

```
Outline
Intro and motivation
Streaming model
CUDA
NVIDIA GPUs
Optimization
Examples
Libraries
GPUs and UU
```

```
--syncthreads();
int value = 0;
for( offset = -RADIUS; offset<=RADIUS; offset++)
value += s_a[ local_ix + offset ];
output[global_ix] = value;
}</pre>
```

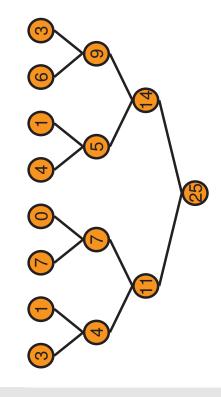
500

php

D. Lukarski, March, 2014, Uppsala



Parallel Reduction



Intro and motivation Streaming model CUDA NVIDIA GPUs Optimization Examples

Outline

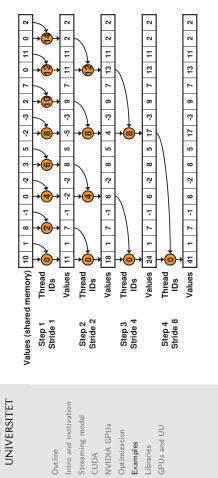
GPUs and UU

Libraries

D. Lukarski, March, 2014, Uppsala



Parallel Reduction



Libraries GPUs and UU

NVIDIA GPUs

CUDA

Optimization Examples

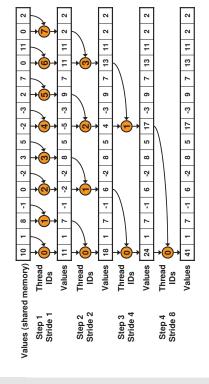
Warps diverge

D. Lukarski, March, 2014, Uppsala

500 jilji



Parallel Reduction



Intro and motivation Streaming model

Outline

CUDA NVIDIA GPUs

Optimization Examples

GPUs and UU

Libraries

Bank conflicts

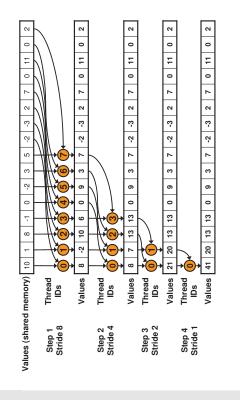
D. Lukarski, March, 2014, Uppsala

pliji



Outline Intro and motivation Streaming model

Parallel Reduction



Examples Libraries GPUs and UU

NVIDIA GPUs

CUDA

Optimization

500

jliji

D. Lukarski, March, 2014, Uppsala



Parallel Reduction

	Time (2^{22} ints)	Bandwidth	Step Speedup	Cumulative Speedup
Kernel 1: interleaved addressing with divergent branching	8.054 ms	2.083 GB/s		
Kernel 2: interleaved addressing with bank conflicts	3.456 ms	4.854 GB/s	2.33x	2.33x
Kernel 3: sequential addressing	1.722 ms	9.741 GB/s	2.01x	4.68x
Kernel 4: first add during global load	0.965 ms	17.377 GB/s	1.78x	8.34x
Kernel 5: unroll last warp	0.536 ms	31.289 GB/s	1.8x	15.01x
Kernel 6: completely unrolled	0.381 ms	43.996 GB/s	1.41x	21.16x
Kernel 7: multiple elements per thread	0.268 ms	62.671 GB/s	1.42x	30.04x

Intro and motivation
Streaming model
CUDA
NVIDIA GPUs
Optimization
Examples
Libraries

Outline

GPUs and UU

Note: very old hardware



ntroduction and motivation

Streaming programming model

Outline Intro and motivation Streaming model

NVIDIA GPIIs - hardware over

Ontimization

NVIDIA GPUs

CUDA

Optimization Examples Libraries

.

GPUs and UU

GPU Libraries

GPUs and Uppsala University

D. Lukarski, March, 2014, Uppsala

000

php



Libraries

CUBLAS

► Dense operations

CUSPARSE

Intro and motivation Streaming model

Outline

- Sparse operations
- Various formats

CUDA
NVIDIA GPUs
Optimization
Examples
Libraries

Thrust

GPUs and UU

- Parallel algorithms
- ► Similar to the C++ Standard Template Library (STL)

D. Lukarski, March, 2014, Uppsala ← □ ▶ ← 🗇 ▶ ← 🖹 ▶



Outline Intro and motivation Streaming model

NVIDIA GPUs

CUDA

Optimization Examples Libraries

GPUs and UU

GPUs and Uppsala University

D. Lukarski, March, 2014, Uppsala

500

UPPSALA UNIVERSITET

Outline

Intro and motivation Streaming model CUDA NVIDIA GPUs

GPUs and UU Optimization Examples Libraries

GPUs and UU

We do GPU computing and we interested in

- New algorithms
- Adapting old algorithms
- ▶ Developing software

Uppsala University is also





References (from this talk)

UPPSALA UNIVERSITET

Outline
Intro and motivation
Streaming model
CUDA
NVIDIA GPUs
Optimization
Examples
Libraries
GPUs and UU

CUDA 5.0 documentationNVIDIA/CUDA slides (google)

D. Lukarski, March, 2014, Uppsala

000