Makespan computation for GPU threads running on a single streaming multiprocessor

Kostiantyn Berezovskyi
CISTER/INESC-TEC, ISEP
Porto, Portugal

Konstantinos Bletsas
CISTER/INESC-TEC, ISEP
Porto, Portugal

Björn Andersson
SEI CMU
Pittsburgh, USA

Supported by the REGAIN project, ref. FCOMP-01-0124-FEDER-020447, co-funded by National Funds through the FCT-MCTES ERDF through COMPETE under PhD grant SFRH/BD/82069/2011
Stream Processing

Streams
Collection of data.
All data is expressed in streams.

Kernels
Series of operations.
Input: streams.
Output: streams.
Why Streams?

Data parallelism
Stream elements can be processed at once.

Task parallelism
Pipeline.
GPU software application

Large data collections.

Data parallelism.

High arithmetic intensity.

Minimal dependency between data elements.
Application areas
Related work

G. Elliott and J. Anderson. 

**Real-time multiprocessor systems with GPUs.**


**Rgem: A responsive GPGPU execution model for runtime engines.**

R. Mangharam and A. A. Saba. 

**Anytime algorithms for GPU architectures.**
Fermi Architecture

16 streaming multiprocessors
Fermi Streaming Multiprocessor (SM)
Computation entities

thread ID

Warp

a group of threads with consecutive IDs; all should be executed in parallel.
__global__ void kv(int* x, int* y) {
    int i = threadIdx.x + blockDim.x * blockIdx.x;
    int t;
    bool b = f(x[i]);
    if (b)
    {
        // g(x)
        t = g(x[i]);
    }
    else
    {
        // h(x)
        t = h(x[i]);
    }
    y[i] = t;
}
How does it work?

```c
float md = (x - x_1)^2 + (y - y_1)^2;  //minimal distance
int mdp = 1;                           //minimal distance point
for (int i=2; i<=N; i++)
    if ((x - x_i)^2 + (y - y_i)^2 < md) {
        md = (x - x_i)^2 + (y - y_i)^2;
        mdp = i;
    }
```

Parallel Thread Execution (PTX)

CUDA cores

load/store units
\textbf{PTX Assembly Code}

\begin{verbatim}
mov.u32  $r0,  N_addr;
mov.f32  $f1,  x_addr;
mov.f32  $f2,  y_addr;
mov.f32  $f3,  x1_addr
mov.f32  $f4,  y1_addr;
sub.f32  $f5,  $f1,  $f3;
mul.f32  $f6,  $f5,  $f5;
sub.f32  $f7,  $f2,  $f4;
fma.f32  $f8,  $f7,  $f7,  $f6;
mov.f32  $f0,  $f8;
mov.u32  $r1,  1;
mov.u32  $r2,  2;

Loop:
setp.gt.u32  p,  $r2,  $r0;
bra Done,
@p
mov.f32  $f3,  xi_addr;
mov.f32  $f4,  yi_addr;
sub.f32  $f5,  $f1,  $f3;
mul.f32  $f6,  $f5,  $f5;
sub.f32  $f7,  $f2,  $f4;
fma.f32  $f8,  $f7,  $f7,  $f6;
setp.ge.f32  q,  $f8,  $f0;
bra If,
@q
mov.f32  $f0,  $f8;
mov.f32  $f1,  $r2;
If:
add.u32  $r2,  $r2,  1;
bra Loop;
Done:
\end{verbatim}
Kernel instruction string

```c
float md = (x - x_1)^2 + (y - y_1)^2;  // minimal distance
int mdp = 1;                           // minimal distance point
for (int i=2; i<=N; i++)
    if ((x - x_i)^2 + (y - y_i)^2 < md){
        md = (x - x_i)^2 + (y - y_i)^2;  // minimal distance
        mdp = i;
    }
```

CUDA cores  \( \mathcal{C} \)

load/store units \( \mathcal{L} \)
Related work

**An Adaptive Performance Modeling Tool for GPU Architectures**
Sara S. Baghsorkhi, Matthieu Delahaye, Sanjay J. Patel, Willian D. Gropp, Wen-mei W. Hwu

**A Memory-level and Thread-level Parallelism Aware GPU Architecture Performance Analytical Model,**
Sunpyo Hong and Hyesoon Kim

**Optimization Principles and Application Performance Evaluation of a Multithreaded GPU Using CUDA**
Shane Ryoo, Christopher I. Rodrigues, Sara S. Baghsorkhi, Sam S. Stone, David B. Kirk, and Wen-mei W. Hwu
The makespan

### Makespan

<table>
<thead>
<tr>
<th>Clock Cycle</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warp 1</td>
<td>C</td>
<td>L</td>
<td>L</td>
<td>C</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td>L</td>
</tr>
<tr>
<td>Warp 2</td>
<td>C</td>
<td>L</td>
<td>L</td>
<td>C</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td>L</td>
</tr>
</tbody>
</table>

### Worst case

<table>
<thead>
<tr>
<th>Clock Cycle</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warp 1</td>
<td>C</td>
<td>L</td>
<td>L</td>
<td>C</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Warp 2</td>
<td>C</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>C</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Assumptions

Every instruction takes single clock cycle.
Cache

A warp on a single SM

32-bit float instruction

**Cache hit**

1 clock cycle

**Cache miss**

400-600 clock cycles
Assumptions

Every instruction takes single clock cycle.

No cache misses.
### Scheduling Policy

**Non-work-conserving schedule**

<table>
<thead>
<tr>
<th>Clock Cycle</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warp 1</td>
<td>C</td>
<td>L</td>
<td></td>
<td>L</td>
<td>C</td>
<td>L</td>
<td></td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Warp 2</td>
<td></td>
<td>C</td>
<td>L</td>
<td></td>
<td>L</td>
<td>C</td>
<td>L</td>
<td></td>
<td>L</td>
<td></td>
</tr>
</tbody>
</table>

**Work-conserving schedule**

<table>
<thead>
<tr>
<th>Clock Cycle</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warp 1</td>
<td>C</td>
<td>L</td>
<td>L</td>
<td>C</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Warp 2</td>
<td>C</td>
<td>L</td>
<td>L</td>
<td>C</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Assumptions

Every instruction takes single clock cycle.
No cache misses.

Work-conserving scheduling.
The Pessimistic Method

\[ M = M^L + M^C \]

- \( M^C \) a makespan for C-substring
- \( M^L \) a makespan for L-substring
Optimization Problem

Decision variables

Usage of hardware units at particular clock cycle
The pessimistic makespan – an upper bound

Objective function

Maximize the makespan

Constraints

Capacity constraints
Precedence constraints
Work-conserving constraints

Integer Linear Problem (ILP)
## Optimization Problem

Maximize \( \sum_{t=1}^{T} (t \times (LS_{W,I,t} + CC_{W,I,t})) \) subject to

<table>
<thead>
<tr>
<th>iterated variables</th>
<th>expression for constraint</th>
<th>number of constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \forall t )</td>
<td>( \sum_{w=1}^{W} \sum_{i=1}^{I} LS_{w,i,t} \leq \sigma_L )</td>
<td>( T )</td>
</tr>
<tr>
<td>( \forall t )</td>
<td>( \sum_{w=1}^{W} \sum_{i=1}^{I} CC_{w,i,t} \leq \sigma_C )</td>
<td>( T )</td>
</tr>
<tr>
<td>( \forall w = 1..(W - 1) )</td>
<td>( \sum_{t=1}^{T} (t \times (LS_{w,I,t} + CC_{w,I,t})) \leq \sum_{t=1}^{T} (t \times (LS_{W,I,t} + CC_{W,I,t})) )</td>
<td>( W - 1 )</td>
</tr>
<tr>
<td>( \forall w, t )</td>
<td>( \sum_{i=1}^{I} LS_{w,i,t} \leq 1 )</td>
<td>( W \times T )</td>
</tr>
<tr>
<td>( \forall w, t )</td>
<td>( \sum_{i=1}^{I} CC_{w,i,t} \leq 1 )</td>
<td>( W \times T )</td>
</tr>
<tr>
<td>( \forall w, i )</td>
<td>( \sum_{t=1}^{T} LS_{w,i,t} = IL_i )</td>
<td>( W \times I )</td>
</tr>
<tr>
<td>( \forall w, i )</td>
<td>( \sum_{t=1}^{T} CC_{w,i,t} = IC_i )</td>
<td>( W \times I )</td>
</tr>
<tr>
<td>( \forall w, i = 1..(I - 1) )</td>
<td>( \sum_{t=1}^{T} (t \times (LS_{w,i,t} + CC_{w,i,t})) &lt; \sum_{t=1}^{T} (t \times (LS_{w,i+1,t} + CC_{w,i+1,t})) )</td>
<td>( W \times (I - 1) )</td>
</tr>
<tr>
<td>( \forall t )</td>
<td>( E_t \geq 1 - \sigma_L + \sum_{w=1}^{W} \sum_{i=1}^{I} LS_{w',i,t} )</td>
<td>( T )</td>
</tr>
<tr>
<td>( \forall t )</td>
<td>( E_t \times \sigma_L \leq \sum_{w=1}^{W} \sum_{i=1}^{I} LS_{w',i,t} )</td>
<td>( T )</td>
</tr>
<tr>
<td>( \forall w, i, t )</td>
<td>( \frac{1}{2} (TL_{w,i,t} + E_t) \leq TL_{w,i,t} \vee E_t \leq TL_{w,i,t} + E_t )</td>
<td>( W \times I \times T )</td>
</tr>
<tr>
<td>( \forall t )</td>
<td>( G_t \geq 1 - \sigma_C + \sum_{w=1}^{W} \sum_{i=1}^{I} CC_{w',i,t} )</td>
<td>( T )</td>
</tr>
<tr>
<td>( \forall t )</td>
<td>( G_t \times \sigma_C \leq \sum_{w=1}^{W} \sum_{i=1}^{I} CC_{w',i,t} )</td>
<td>( T )</td>
</tr>
<tr>
<td>( \forall w, i, t )</td>
<td>( \frac{1}{2} (TC_{w,i,t} + G_t) \leq TC_{w,i,t} \vee G_t \leq TC_{w,i,t} + G_t )</td>
<td>( W \times I \times T )</td>
</tr>
</tbody>
</table>
The Application

The configuration

The application

The schedule

The ILP-solver

The solution

kernel
c l l
warp_number
8
ls_unit_number
16
cuda_core_number
32
warp_size
32
Resolving the issue of tractability

\[ W = 420 \]
Future Work

Relaxation of the “no cache miss assumption”.

Targeting the “whole” GPU.
Future Work

\[ W = 420, \]

Fermi 48 warps
Questions?
Thank You!