Parallel TwigStack using GPGPU

Vincent Jordan (exchange research student)

Kitagawa Data Engineering Laboratory
University of Tsukuba

XML&P2P seminar on July 20, 2010
Outline

1. **TwigStack algorithm**
   - Data pre-process
   - Algorithm overview
   - Stack compact encoding
   - Parallel TwigStack

2. **GPU architecture**
   - Hardware and software architectures mappings
   - Limitations and resulting debugging difficulties

3. **Implementation**
   - TwigStack parallelization
   - GPU limitations overcoming
   - Further improvements
<?xml version="1.0" encoding="UTF-8" ?>
<library>
  <category name="France">
    <book>
      <title language="English">The Little Prince</title>
      <title language="日本語">星の王子さま</title>
      <author>Antoine de Saint-Exupéry</author>
    </book>
  </category>
  <category name="England">
    <book>
      <title language="English">Alice in Wonderland</title>
      <title language="日本語">ふしぎの国のアリス</title>
      <author>Lewis Carroll</author>
    </book>
  </category>
</library>
Example query

Request

What is the English title of French books of the library?

XPath query

```
/library/category[@name=France]/book/title[@language=English]
```

Answer

```
<title language="English">The Little Prince</title>
```
Example query

**Request**

What is the English title of French books of the library?

**XPath query**

/library/category[@name=France]/book/title[@language=English]
Example query

**Request**
What is the English title of French books of the library?

**XPath query**
/library/category[@name=France]/book/title[@language=English]

**Answer**
<title language="English">The Little Prince</title>
XML query process algorithm

Holistic twig joins: optimal XML pattern matching

Nicolas Bruno, Nick Koudas, Divesh Srivastava
International Conference on Management of Data
Proceedings of the 2002 ACM SIGMOD international conference on Management of data
Outline

1. **TwigStack algorithm**
   - Data pre-process
   - Algorithm overview
   - Stack compact encoding
   - Parallel TwigStack

2. **GPU architecture**
   - Hardware and software architectures mappings
   - Limitations and resulting debugging difficulties

3. **Implementation**
   - TwigStack parallelization
   - GPU limitations overcoming
   - Further improvements
XPath query paths

/\texttt{library/category[@name=France]/book/title[@language=English]}
TwigStack algorithm
GPU architecture
Implementation

Data pre-process
Algorithm overview
Stack compact encoding
Parallel TwigStack

XPath query tree

library

category

@name
France

book

title

@language
English

library

category

@name
France

book

title

@language
English

Parallel TwigStack using GPGPU
XML metadata

```xml
<library>
  <category name="France">
    <book>
      <title>language="English">
        The Little Prince
      </title>
    </book>
  </category>
</library>
```

<table>
<thead>
<tr>
<th>value</th>
<th>left</th>
<th>right</th>
<th>depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>library</td>
<td>1</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>category</td>
<td>2</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>book</td>
<td>5</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>title</td>
<td>6</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>name</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>language</td>
<td>7</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>France</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>English</td>
<td>8</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>The</td>
<td>9</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>Little</td>
<td>10</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>Prince</td>
<td>11</td>
<td>11</td>
<td>4</td>
</tr>
</tbody>
</table>

Parallel TwigStack using GPGPU
Outline

1. **TwigStack algorithm**
   - Data pre-process
   - Algorithm overview
   - Stack compact encoding
   - Parallel TwigStack

2. **GPU architecture**
   - Hardware and software architectures mappings
   - Limitations and resulting debugging difficulties

3. **Implementation**
   - TwigStack parallelization
   - GPU limitations overcoming
   - Further improvements
XPath query string
\texttt{elem1/elem2[@arg1=text1][/elem3=text2]}

XML data file

metadata

<table>
<thead>
<tr>
<th>elem1</th>
<th>1</th>
<th>18</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>elem2</td>
<td>2</td>
<td>17</td>
<td>1</td>
</tr>
<tr>
<td>elem3</td>
<td>3</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>text2</td>
<td>4</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>text3</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

query tree

TwigStack algorithm

Tree list = result

Phase 1

Paths list = intermediate result

Phase 2

Parallel TwigStack using GPGPU
Outline

1. **TwigStack algorithm**
   - Data pre-process
   - Algorithm overview
   - Stack compact encoding
   - Parallel TwigStack

2. **GPU architecture**
   - Hardware and software architectures mappings
   - Limitations and resulting debugging difficulties

3. **Implementation**
   - TwigStack parallelization
   - GPU limitations overcoming
   - Further improvements
Stack compact encoding 1

XML data

```xml
<library>
  <category name="France">
    <book>
      <title language="English">The ...</title>
    </book>
  </category>
</library>

<library>
  <category name="England">
    <book>
      <title language="English">Alice ...</title>
    </book>
  </category>
</library>
```

query tree

```
library
  category
    book
      title
        @name = France
        @language = English
```

query node stack

1. library
2. category
3. title
   - @name = France
   - @language = English
4. book

Parallel TwigStack using GPGPU
Stack compact encoding 2

XML data

```
<library>
  <category name="France">
    <book>
      <title language="English">The [...]</title>
    </book>
  </category>
</library>
```

```
<library>
  <category name="England">
    <book>
      <title language="English">Alice [...]</title>
    </book>
  </category>
</library>
```
Stack compact encoding 3

XML data

```xml
<library1>
  <category2 name3="France4">
    <book5>
      <title6 language7="English8">The9 [...]</title12>
    </book13>
  </category14>
  <category15 name16="England17">
    <book18>
      <title19 language20="English21">Alice22 [...]</title25>
    </book26>
  </category27>
</library28>
```

query tree

```
library
  category
    book
      title
        @name France
        @language English
```

query node stack

```
1 library
15 category
18 book
19 title
  @name
  @language
20 France
21 English
```
Stack compact encoding 4

/library//@name=France//@language=English

same XML tag can be nested

<library>
  <category name="by country">
    <category name="Europe">
      <category name="Western Europe">
        ... 17 1 9 20
        same XML tag can be nested
      </category>
      <category name="France">
        <book>
          <title language="English">The Little Prince</title>
        </book>
      </category>
    </category>
  </category>
</library>

Parallel TwigStack using GPGPU
Merge (phase 2)

these nodes can be merged according to the query tree

(1,48,0) (2,26,1) (5,25,2) (6,12,3) (7,7,4) (8,8,5)
(1,48,0) (2,26,1) (3,3,2) (4,4,3)
(1,48,0) (27,48,1) (30,47,2) (31,37,3) (32,32,4) (33,33,5)
Outline

1. **TwigStack algorithm**
   - Data pre-process
   - Algorithm overview
   - Stack compact encoding
   - Parallel TwigStack

2. **GPU architecture**
   - Hardware and software architectures mappings
   - Limitations and resulting debugging difficulties

3. **Implementation**
   - TwigStack parallelization
   - GPU limitations overcoming
   - Further improvements
Parallel TwigStack

A Study on Parallel Holistic Twig Joins for XML Query Processing

Imam Machdi
PhD dissertation at Kitagawa Data Engineering lab., March 2010
Outline

1 TwigStack algorithm
   - Data pre-process
   - Algorithm overview
   - Stack compact encoding
   - Parallel TwigStack

2 GPU architecture
   - Hardware and software architectures mappings
   - Limitations and resulting debugging difficulties

3 Implementation
   - TwigStack parallelization
   - GPU limitations overcoming
   - Further improvements
Hardware architecture

- **Global GPU memory**: 1.5GB
- **Main CPU memory**: 4GB
- **Cache L3**: 12MB
- **Cache L2**: 1MB
- **Cache L1**: 32KB
- **PCIExpress 16x**: 250MB/s × 16 lanes
- **DMA transfers**: 8KB (read-only)
- **Texture memory cache**: 8KB (read-only)
- **Register**: 16K × 32bits
- **Multicore**: out of 24
- **Special function unit**: transcendental & double precision operations
- **NVidia Quadro FX 4800**: Parallel TwigStack using GPGPU
- **Intel Xeon E5630**: Hardware and software architectures mappings

Limitations and resulting debugging difficulties

TwigStack algorithm
GPU architecture
Implementation
Software architecture

The diagram illustrates the hardware and software architectures mappings, including Grid, Block, Warp, and Thread. It also shows the GPU code execution from CPU with the following code snippet:

```c
myCUDAKernel<<gridSize, blockSize>>(int* arg1, ...);
```
Mapping of software and hardware representations

Hardware representation

Software representation

Parallel TwigStack using GPGPU
Massively parallel architecture

Warp, block and grid sizes have hardware limits

1. Warp size: 32
2. Max threads per blocks: 512 (or 16 warps)
3. Max thread simultaneously active per multiprocessor: 1024
4. Max resident warps per multiprocessor: 32
5. Max resident blocks per multiprocessor: 8

Example

32 threads x 4 warps x 8 blocks x 24 multiprocessors = 24,576
Outline

1. **TwigStack algorithm**
   - Data pre-process
   - Algorithm overview
   - Stack compact encoding
   - Parallel TwigStack

2. **GPU architecture**
   - Hardware and software architectures mappings
   - Limitations and resulting debugging difficulties

3. **Implementation**
   - TwigStack parallelization
   - GPU limitations overcoming
   - Further improvements
Limitations of current hardware architecture

**SIMT architecture** "Single Instruction Multiple Threads": threads in the same warp execute the same instruction at the same time.

**No function calls** functions are inlined at compilation time. Recursive functions are not allowed.

**No dynamic memory allocation** everything has to be allocated before execution.

**No synchronization among threads** critical section handling is not recommended.

**No hardware cache** each access to global memory costs several hundreds of GPU cycles.

**Local memory pointers** only depends on the ability of the compiler to match them.
Debugging difficulties

Functions are inlined unable to "jump" a function while executing GPU code step-by-step. No functions backtrace.

No function calls no printf or trace of execution progress.

No dynamic memory allocation big chunks of linear memory are allocated. Wrong memory accesses are often in allocated memory and don’t trigger error.
Outline

1. **TwigStack algorithm**
   - Data pre-process
   - Algorithm overview
   - Stack compact encoding
   - Parallel TwigStack

2. **GPU architecture**
   - Hardware and software architectures mappings
   - Limitations and resulting debugging difficulties

3. **Implementation**
   - TwigStack parallelization
   - GPU limitations overcoming
   - Further improvements
TwigStack algorithm

**GPU architecture**

**Implementation**

TwigStack parallelization

GPU limitations overcoming

Further improvements

XPath query string

```
 elem1/elem2[@arg1=text1][/elem3=text2]
```

XML data file

```
data.xml
```

**TwigStack algorithm**

Paths list = intermediate

Result

```
Tree list = result
```

Parallel TwigStack using GPGPU
Initialization (→ CPU)

/library/category[@name=France]/book/title[@language=English]

<library1>
<category2 name3="France4">
<book5>
<title6 language7="English8">The9 […]</title12>
<title13 language14="日本語15">星の王子さま16</title17>
<author18>Antoine19 de20 Saint-Exupéry21</author22>
</book25>
</category26>
<category27 name28="England29">
<book30>
<title31 language32="English33">Alice34 […]</title37>
<title38 language39="日本語40">ふしぎの国のアリス41</title42>
<author43>Lewis44 Carroll45</author46>
</book47>
</category48>
</library49>
(1,48,0);(27,57,1);(30,56,2) are duplicated since they are required by both partitions.
Parallel execution of phase 1 (→ GPU)

Thread 1:
- Library
- Category
- Book
- Title
- @name: France
- @language: English

Thread 2:
- Library
- Category
- Book
- Title
- @name: France
- @language: English

Parallel TwigStack using GPGPU
Second phase: merging (→ CPU)

these nodes can be merged according to the query tree

(1,48,0) (1,48,0)
(2,26,1) (2,26,1)
(5,25,2) (3,3,2)
(6,12,3) (4,4,3)
(7,7,4)
(8,8,5)

(1,48,0)
(27,48,1)
(30,47,2)
(31,37,3)
(32,32,4)
(33,33,5)
Outline

1 **TwigStack algorithm**
   - Data pre-process
   - Algorithm overview
   - Stack compact encoding
   - Parallel TwigStack

2 **GPU architecture**
   - Hardware and software architectures mappings
   - Limitations and resulting debugging difficulties

3 **Implementation**
   - TwigStack parallelization
   - GPU limitations overcoming
   - Further improvements
Memory pool

GPU global memory

bitfield of allocated memory chunks

Parallel TwigStack using GPGPU
Critical section

/* critical section start */
#ifdef CUDA_DEVICE_MODE
    while(1) {
        if(atomicExch(&data->lock_i, 1) == 0) {
            break;
        }
    }
#else
    pthread_mutex_lock(&data->lock_m);
#endif
v_array data-structure

new row can be prepended or appended with low memory access cost

Upload v_array from CPU to GPU memory

v_array_post_copy(
    q_data->metadata elems,
    &cuda_malloc_func,
    NULL, /* malloc function arg. (e.g memory pool) */
    &cuda_memcpyH2D_func);
Designed for GPU

1. all chunks have the same size (→ memory manager)
2. chunks contain the real data (memcpy), not pointer (→ aligned efficient memory fetch)
3. highly customizable: malloc, memcpy and free functions are given as parameters.
   Same source code can be used for host and device (even for ”host→device” or ”device→host” transfers).
4. if each chunk contains 32 rows, a warp could process it using 1 thread per row.
PRE_FUNC_ATTR v_array_iter_t PREFIX_ARRAY(get_row)(v_array_iter_t i) {
    int cur_size_sum = 0;
    /* find next array element containing the row idx */
    if(i.idx > 0) {
        while((cur_size_sum + i.chunk->num_row) <= i.idx) {
            cur_size_sum += i.chunk->num_row;
            i.chunk = (v_array_t)i.chunk->next;
        }
        /* find prev array element containing the row idx */
    } else if(i.idx < 0) {
        while(cur_size_sum > i.idx) {
            i.chunk = (v_array_t)i.chunk->prev;
            cur_size_sum -= i.chunk->num_row;
        }
    }
    i.idx = i.idx - cur_size_sum; /* idx relative to the current a */
    /* update pointer to the row in the current array element */
    i.result.data = PREFIX_ARRAY(elem_get_row)(i.chunk, i.idx);
    return i;
}
Outline

1. **TwigStack algorithm**
   - Data pre-process
   - Algorithm overview
   - Stack compact encoding
   - Parallel TwigStack

2. **GPU architecture**
   - Hardware and software architectures mappings
   - Limitations and resulting debugging difficulties

3. **Implementation**
   - TwigStack parallelization
   - GPU limitations overcoming
   - Further improvements
Further improvements

**Memory pool** improve bitfield primitive memory management

**XMalloc: A Scalable Lock-free Dynamic Memory Allocator for Many-core Machines**

*Xiaohuang Huang, Christopher I. Rodrigues, Stephen Jones, Ian Buck, and Wen-mei Hwu*

Proceedings of the First International Workshop on Frontier of GPU Computing (FGC), in conjunction with the 10th IEEE International Conference on Computer and Information Technology (CIT 2010), June 2010.

**Shared memory** array chunk could be automatically stored in shared memory while processing rows (avoiding many global memory accesses)

**Phase 2 parallelization** (task parallelism)

**Make use of the 31 other threads in one warp?**
Parallel TwigStack using GPGPU

Global GPU memory

Dynamic memory pool

Allocated during GPU exec.

Allocated before GPU exec.

Copy from global memory to shared memory

Multicore

Shared memory