Using Integrated Processor Graphics to Accelerate Concurrent Data and Index Structures

Joel Fuentes and Isaac D. Scherson
October 10, 2018

University of California, Irvine. USA.
Universidad del Bío-Bío. Chile.
1. Introduction

2. Integrated GPUs and C for Media

3. Concurrent Data Structures

4. Index Structures

5. Conclusions
Table of Contents

1. Introduction

2. Integrated GPUs and C for Media

3. Concurrent Data Structures

4. Index Structures

5. Conclusions
Introduction

- Multi-core processors computing systems present demanding challenges to programmers:
  - Implementing thread-safe algorithms
  - Achieve scalability when increasing the number of hardware threads
- Integrated GPUs (iGPUs) have been improved achieving GFlops performance
Table of Contents

1. Introduction

2. Integrated GPUs and C for Media

3. Concurrent Data Structures

4. Index Structures

5. Conclusions
• Integration of multiple CPU cores and Intel processor graphics (called GenX) on the same die.
• Bi-directional ring interconnect that has a 32-byte wide bus
• Last Level Cache (LLC) and embedded EDRAM exclusively for iGPU.
Integrated GPUs and C for Media

Shared Virtual Memory between CPU cores and iGPU.
C for Media

- High-level programming environment based on C/C++
- Language designed to efficiently leverage SIMD capability of the iGPU
- Two primary components:
  1. The compiler for the CM language
  2. CM runtime, which provides an API for C/C++ programs

Example:

```cpp
template <typename T> inline vector<T, 16> PrefixSumIn(vector<T, 16> in) {
  vector<T, 32> d = 0;
  d.select<16, 1>(16) = in;

  d.select<16, 1>(16) = d.select<16, 1>(16) + d.select<16, 1>(15);
  d.select<16, 1>(16) = d.select<16, 1>(16) + d.select<16, 1>(14);
  d.select<16, 1>(16) = d.select<16, 1>(16) + d.select<16, 1>(12);
  d.select<16, 1>(16) = d.select<16, 1>(16) + d.select<16, 1>(8);

  return d.select<16, 1>(16);
}
```
Probabilistic data structure that is built upon the general idea of a linked list. It provides good performance for concurrent operations, while still providing $O(\log n)$ complexity for all its operations.
- Chunks of links and keys are used instead of nodes
  - GenX supports SIMD16. Update operations are performed on entire chunks.
- Chunk of links (blue) and keys (yellow):
  - Chunk of links stores offsets of next lists. Chunk of keys stores sorted keys
  - 1 dword is reserved for a link to next chunk within the same list
Example: Inserting 21 with level = 1:

1. Search list and position of 21
2. Create new list. Steal keys > 21 from previous list, including attached chunk
3. Perform **SIMD16 atomic operation** on previous list to remove keys > 21
4. Perform **atomic operations** on upper links to update offset (320) while level ≤ 1
Comparison with state-of-the-art lock-free skiplists for CPU [1, 2, 3, 4, 5, 8]:
Comparison with the M&C's Skiplist [6] for discrete GPU. Experiments carried on a Nvidia GTX 970. Relative efficiency is calculated by
\[
\text{efficiency} = \frac{1}{(\text{time} \times \text{peak\_GFLOPS})}.
\]
Intel iGPU has a peak GFlops of 1,152. Nvidia GTX 970 has a peak GFlops of 3,920.

<table>
<thead>
<tr>
<th>Ops.</th>
<th>CMSL (ms)</th>
<th>M&amp;C (ms)</th>
<th>Relative efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 100, 0]</td>
<td>244</td>
<td>110</td>
<td>1.5</td>
</tr>
<tr>
<td>[0, 50, 50]</td>
<td>212</td>
<td>110</td>
<td>1.6</td>
</tr>
<tr>
<td>[60, 20, 20]</td>
<td>141</td>
<td>69</td>
<td>1.75</td>
</tr>
<tr>
<td>[80, 10, 10]</td>
<td>93</td>
<td>48</td>
<td>1.8</td>
</tr>
<tr>
<td>[90, 5, 5]</td>
<td>72</td>
<td>43</td>
<td>2.0</td>
</tr>
<tr>
<td>[100, 0, 0]</td>
<td>61</td>
<td>44</td>
<td>2.5</td>
</tr>
</tbody>
</table>
Table of Contents

1. Introduction

2. Integrated GPUs and C for Media

3. Concurrent Data Structures

4. Index Structures

5. Conclusions
Index Structures

- Index structures are widely used by search engines, databases, graphics applications, etc.
- Compressed index structures reduce space of regular indices while allowing efficient navigation in compressed form.
- However, their construction is time-consuming.
- Focus on efficient implementation on iGPU: $K^2$-tree, Wavelet tree and Suffix tree
Index Structures: Construction Algorithm

Input
adjacency matrix, text file, etc.

input big enough?

no

yes

Perform work on CPU

Return compact Index structure

Integrated GPU

Compute $m$ levels of index structure

Generate bitstring

Bitstrings: 00101010101011010010101
010101010101001010101010101011
1010110101010101101011010101010
Experimental results: index construction and queries on iGPU compared with SDSL-lite on CPU.

The construction of the index structure $K^2$-tree is **7 times faster** on iGPU. **Speed-up on queries is up to 11x.**
## Table of Contents

1. Introduction

2. Integrated GPUs and C for Media

3. Concurrent Data Structures

4. Index Structures

5. Conclusions
Conclusions

- Integrated GPUs are widely available and powerful computing units for non-graphics workloads.
- We have proposed the first lock-free data structure for iGPU.
  - Concurrent skiplist (CMSL) outperforms state-of-the-art lock-free skiplists for CPU.
- Index structures can benefit from iGPU architecture to accelerate their construction and queries.
- Other index and data structures can be implemented for iGPU using similar design patterns.
Questions?
T. Crain, V. Gramoli, and M. Raynal.  
Brief announcement: a contention-friendly, non-blocking skip list. 

K. Fraser.  
Practical lock-freedom. 

R. Guerraoui and V. Trigonakis.  
Optimistic concurrency with optik. 

M. Herlihy, Y. Lev, V. Luchangco, and N. Shavit.  
A provably correct scalable concurrent skip list. 
M. Herlihy, Y. Lev, V. Luchangco, and N. Shavit.  
**A simple optimistic skiplist algorithm.**  

P. Misra and M. Chaudhuri.  
**Performance evaluation of concurrent lock-free data structures on gpus.**  

W. Pugh.  
**Skip lists: a probabilistic alternative to balanced trees.**  

**Cache-sensitive skip list: Efficient range queries on modern cpus.**  