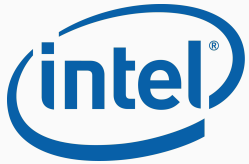


Using Integrated Processor Graphics to Accelerate Concurrent Data and Index Structures

Joel Fuentes and Isaac D. Scherson

October 10, 2018

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

Outline

1. Introduction
2. Integrated GPUs and C for Media
3. Concurrent Data Structures
4. Index Structures
5. Conclusions

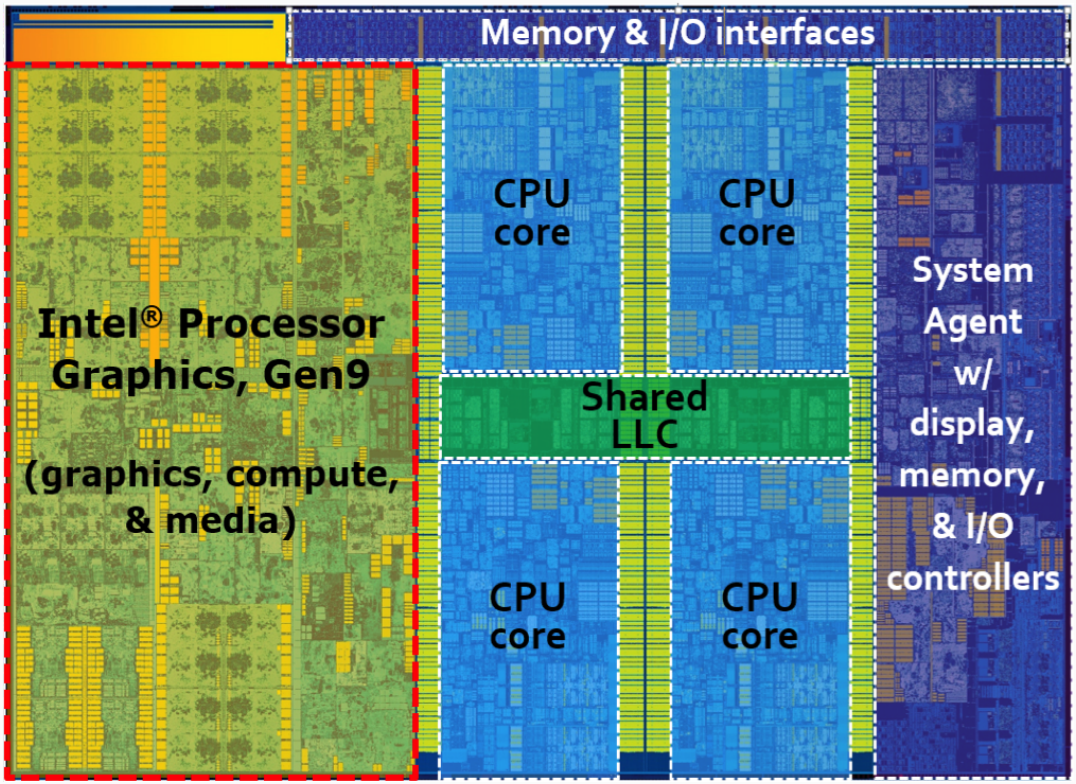
Table of Contents

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

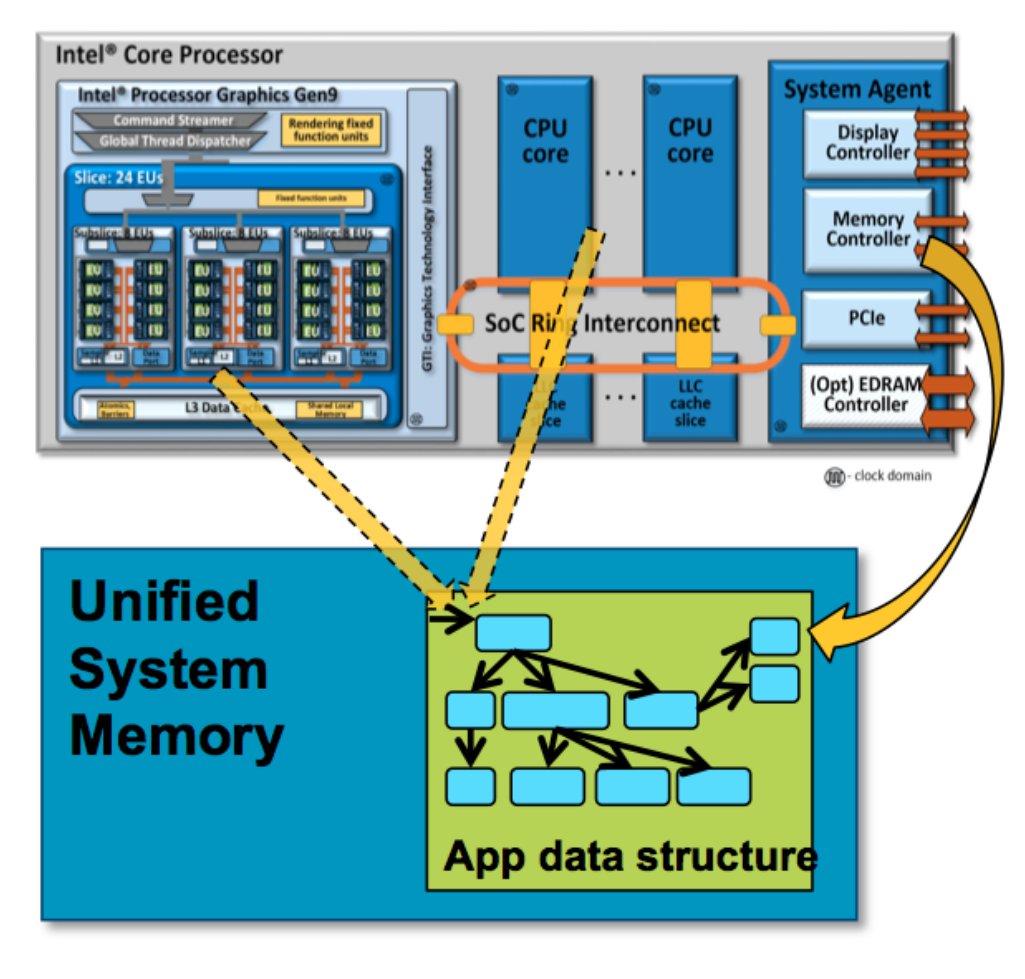
Table of Contents

Integrated GPUs and C for Media



- 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

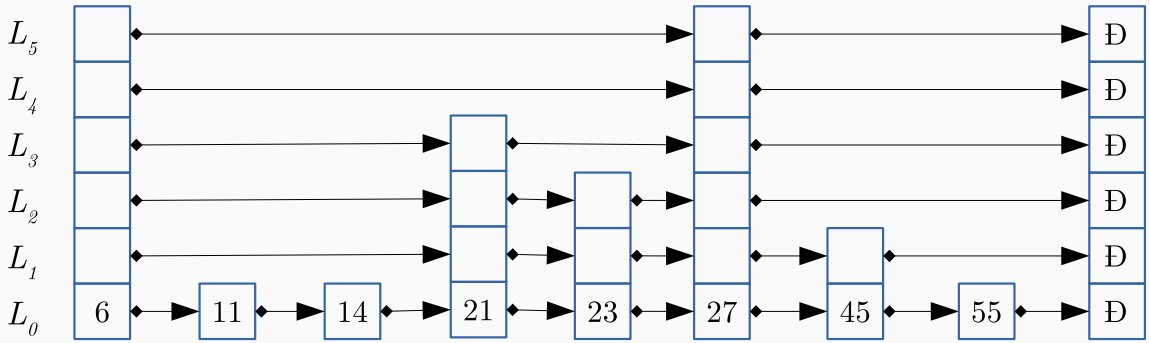
- High-level programming environment based on C/C++.
- Language designed to efficiently leverage SIMD capability of the iGPU
- Two primary components:
  1. The compiler for the CM language
  2. CM runtime, which provides an API for C/C++ programs

Example:

```
1
2
3  template <typename T> inline vector<T, 16> PrefixSumIn(vector<T, 16> in) {
4    vector<T, 32> d = 0;
5    d.select<16, 1>(16) = in;
6
7    d.select<16, 1>(16) = d.select<16, 1>(16) + d.select<16, 1>(15);
8    d.select<16, 1>(16) = d.select<16, 1>(16) + d.select<16, 1>(14);
9    d.select<16, 1>(16) = d.select<16, 1>(16) + d.select<16, 1>(12);
10   d.select<16, 1>(16) = d.select<16, 1>(16) + d.select<16, 1>(8);
11
12   return d.select<16, 1>(16);
13 }
14
```

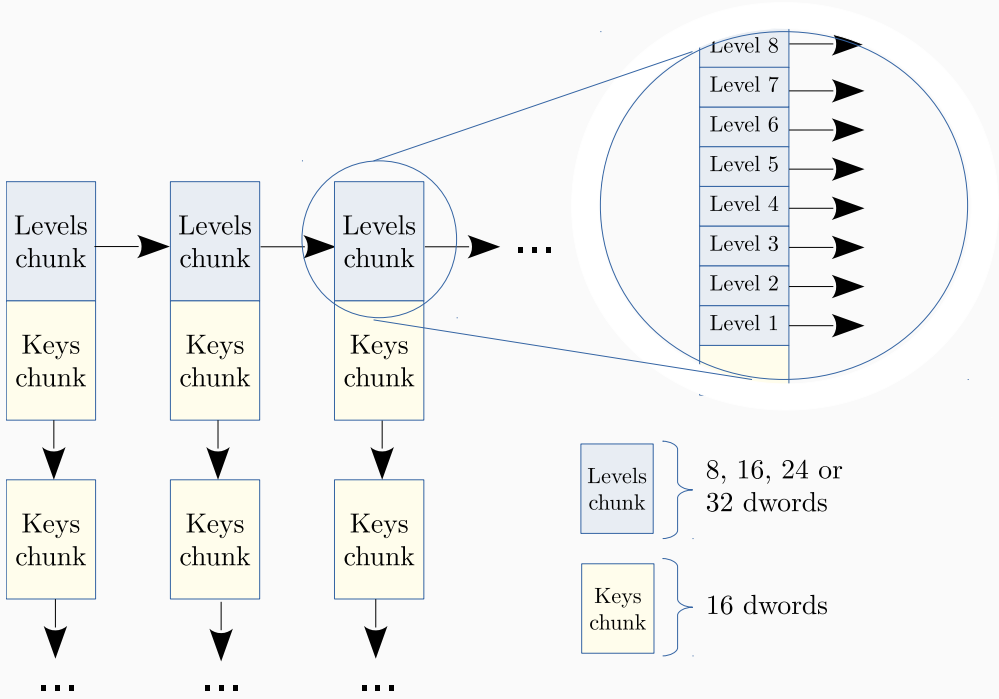
Table of Contents

Lock-free Skiplist



Probabilistic data structure that is built upon the general idea of a linked list It provides well performance for concurrent operations, while still providing $O(logn)$ complexity for all its operations.

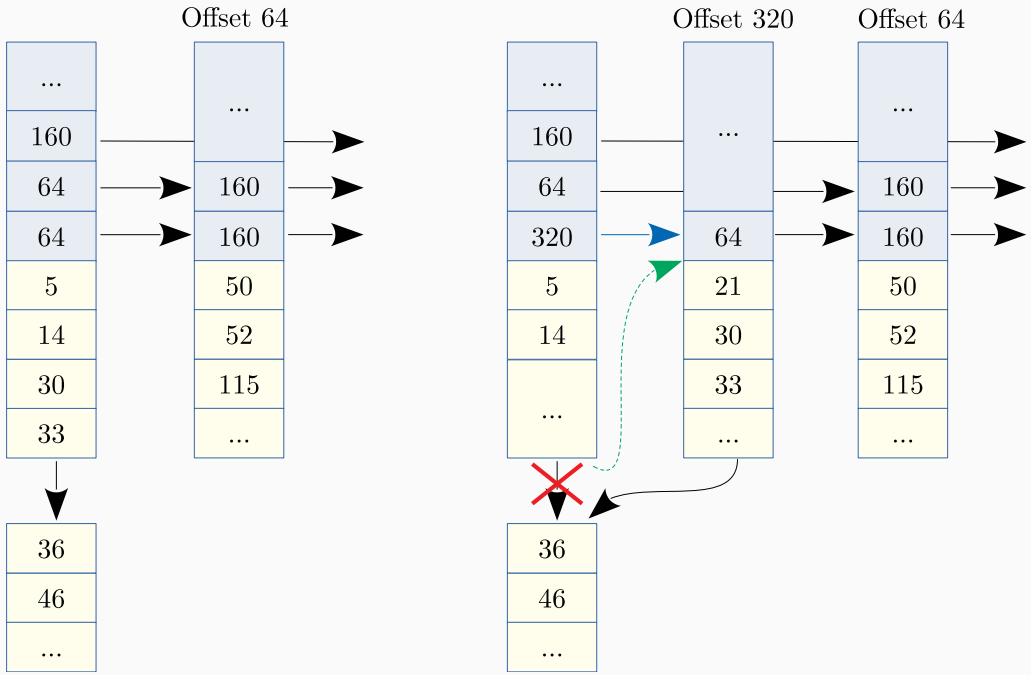
Lock-free Skiplist: Our proposal



- 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

Lock-free Skiplist: Our proposal

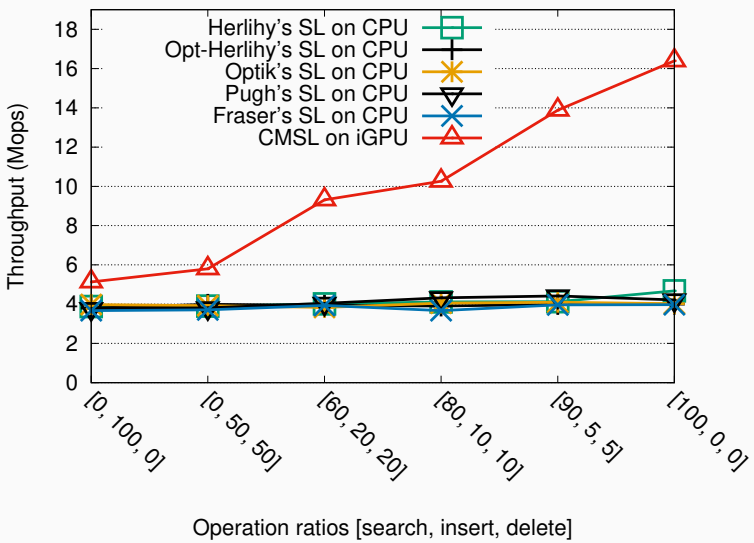
Example: Inserting 21 with *level* = 1:

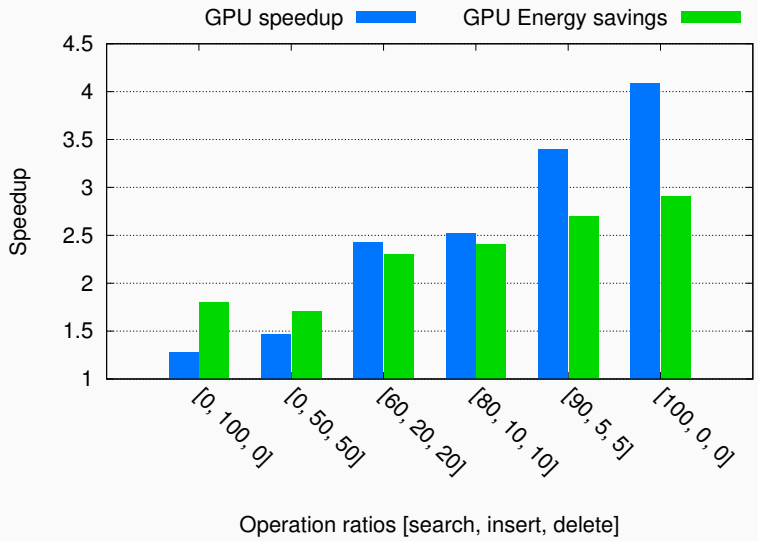


1. Search list and position of 21
2. Create new list. Steal *keys* > 21 from previous list, including attached chunk
3. Perform SIMD16 atomic operation on previous list to remove *keys* > 21
4. Perform atomic operations on upper links to update offset (320) while *level* ≤ 1

Lock-free Skiplist: Experimental results

Comparison with state-of-the-art lock-free skiplists for CPU [1, 2, 3, 4, 5, 8]:





Lock-free Skiplist: Experimental results

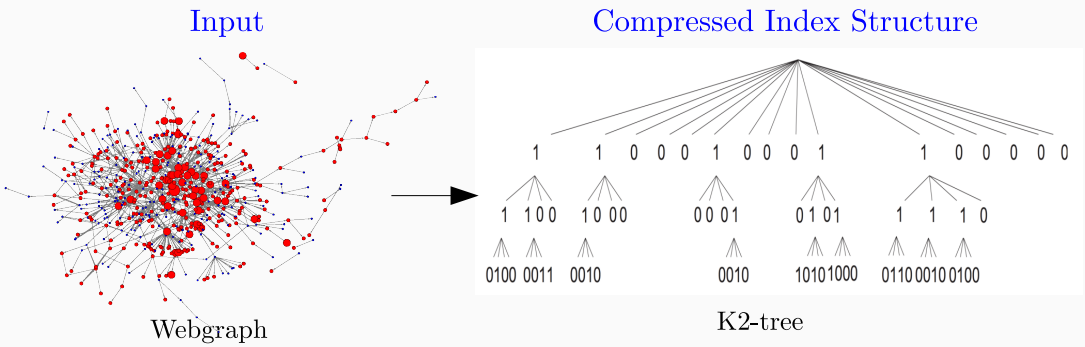
Comparison with the M&C's Skiplist [6] for discrete GPU. Experiments carried on a Nvidia GTX 970. Relative efficiency is calculated by $efficiency = 1/(time * peak\_GFLOPS)$. Intel iGPU has a peak GFlops of 1,152. Nvidia GTX 970 has a peak GFlops of 3,920.

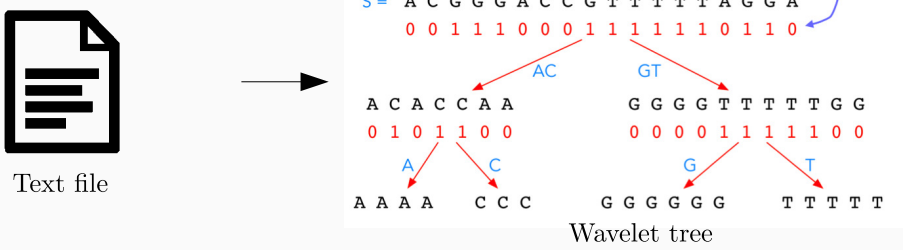
Ops. [search, insert, delete]	CMSL (ms)	M&C (ms)	Relative efficiency
[0, 100, 0]	244	110	1.5
[0, 50, 50]	212	110	1.6
[60, 20, 20]	141	69	1.75
[80, 10, 10]	93	48	1.8
[90, 5, 5]	72	43	2.0
[100, 0, 0]	61	44	2.5

Table of Contents

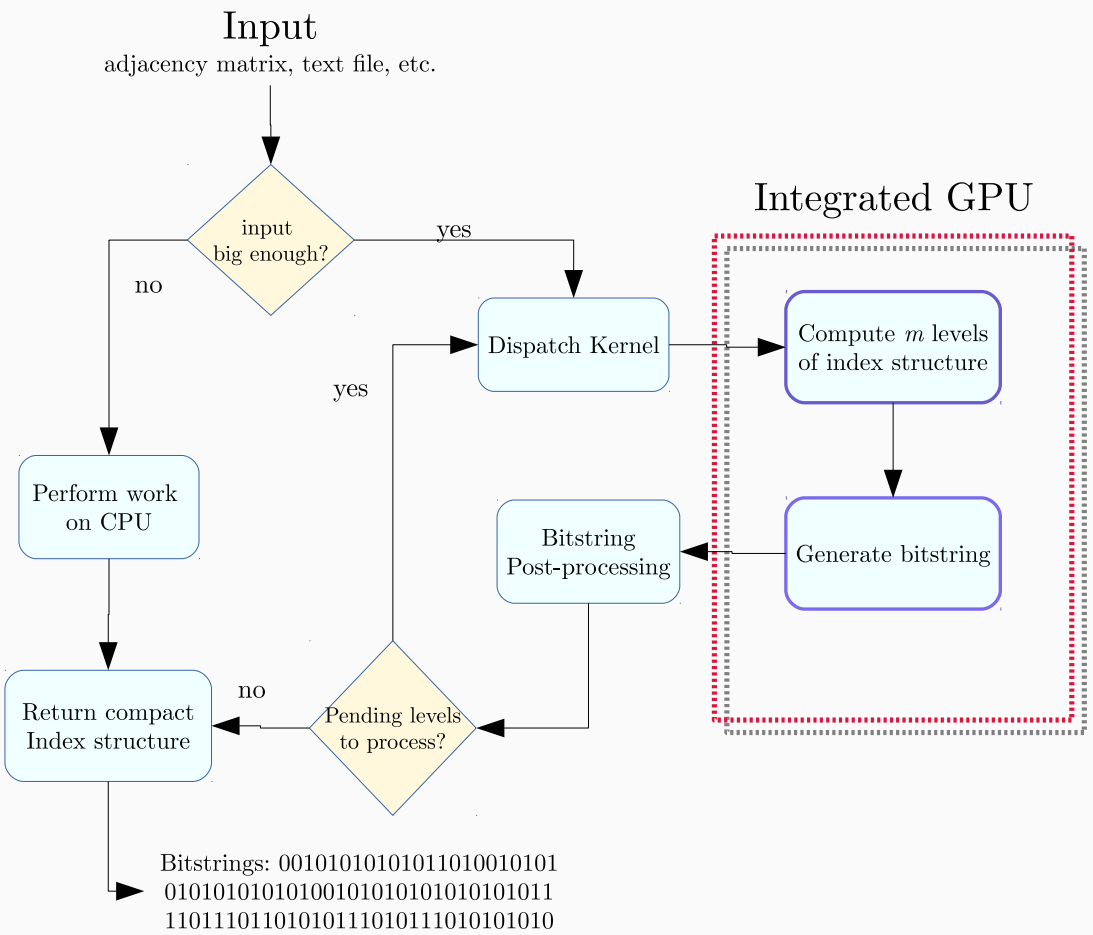
Index Structures

- 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



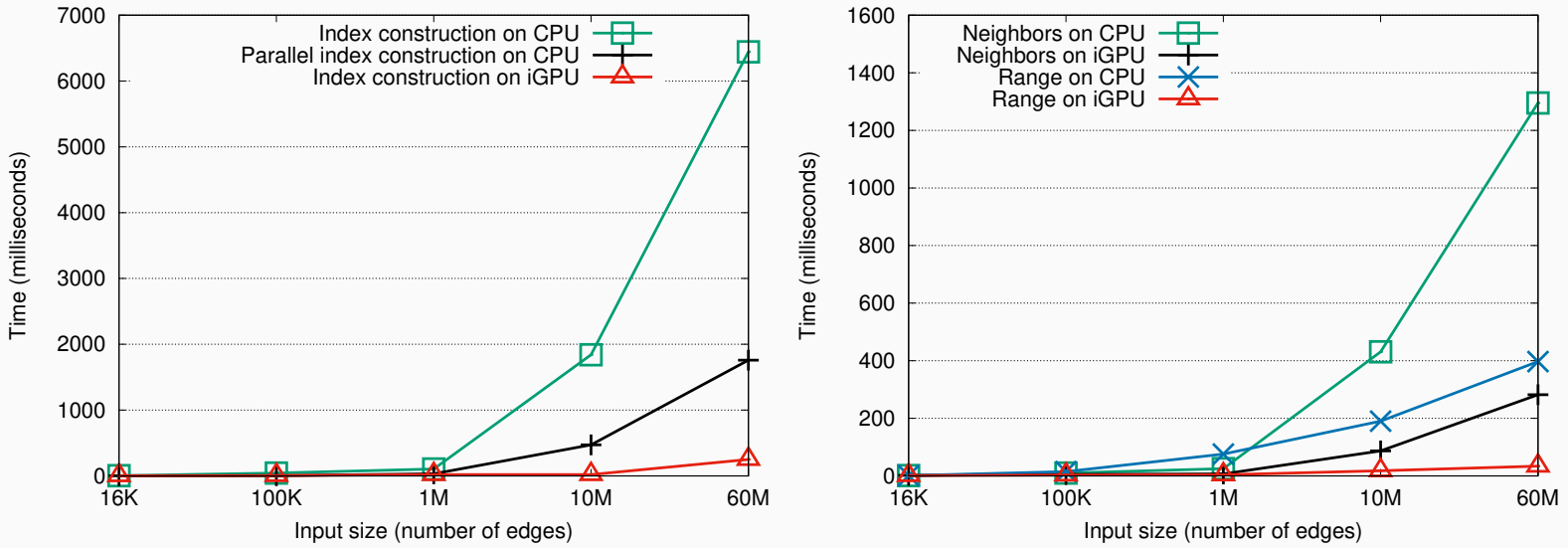


Index Structures: Construction Algorithm



Index Structures: Experimental results

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

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?

References I


T. Crain, V. Gramoli, and M. Raynal.
Brief announcement: a contention-friendly, non-blocking skip list.
In *International Symposium on Distributed Computing*, pages 423–424. Springer, 2012.


K. Fraser.
Practical lock-freedom.
Technical report, University of Cambridge, Computer Laboratory, 2004.


R. Guerraoui and V. Trigonakis.
Optimistic concurrency with optik.
In *ACM SIGPLAN Notices*, volume 51, page 18. ACM, 2016.


M. Herlihy, Y. Lev, V. Luchangco, and N. Shavit.
A provably correct scalable concurrent skip list.
In *Conference On Principles of Distributed Systems (OPODIS)*. Citeseer, 2006.

References II


M. Herlihy, Y. Lev, V. Luchangco, and N. Shavit.
A simple optimistic skiplist algorithm.
In *International Colloquium on Structural Information and Communication Complexity*, pages 124–138. Springer, 2007.


P. Misra and M. Chaudhuri.
Performance evaluation of concurrent lock-free data structures on gpus.
In *Parallel and Distributed Systems (ICPADS), 2012 IEEE 18th International Conference on*, pages 53–60. IEEE, 2012.


W. Pugh.
Skip lists: a probabilistic alternative to balanced trees.
Communications of the ACM, 33(6):668–676, 1990.


S. Sprenger, S. Zeuch, and U. Leser.
Cache-sensitive skip list: Efficient range queries on modern cpus.
In *Data Management on New Hardware*, pages 1–17. Springer, 2016.