OpenMP Cell Distances Benchmarking

Impact of Caches on Performance

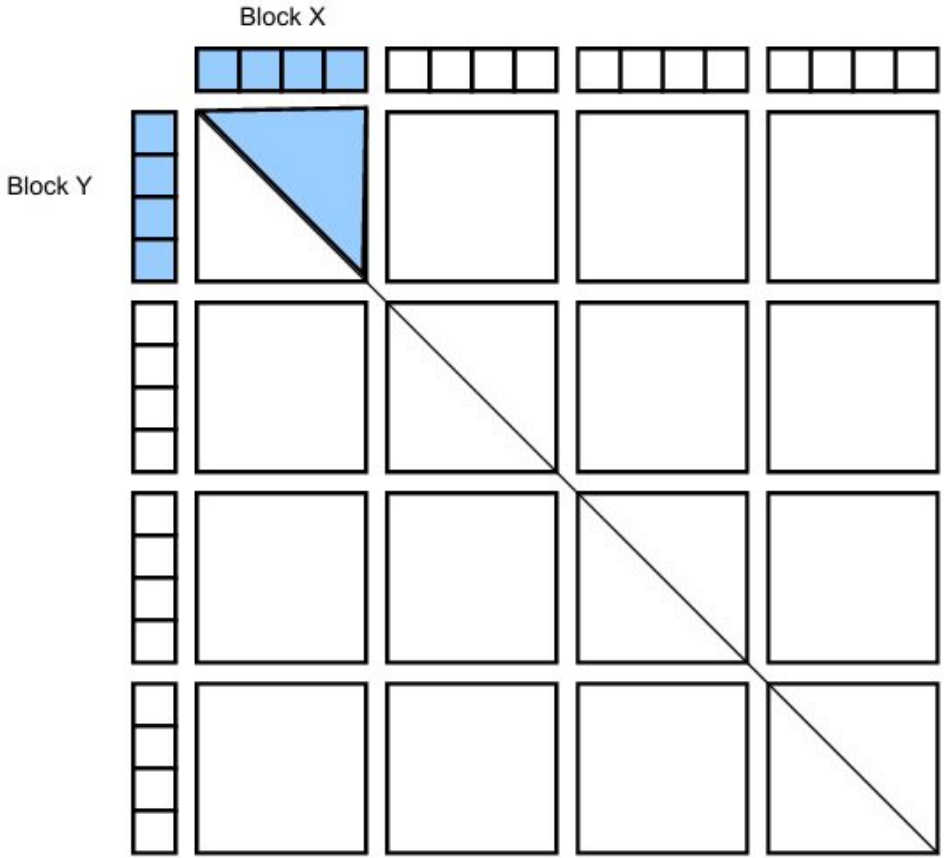
Stefano Ribes

Chalmers University of Technology

Overview

- Algorithm Implementation
- Effect of buffer sizes on performance

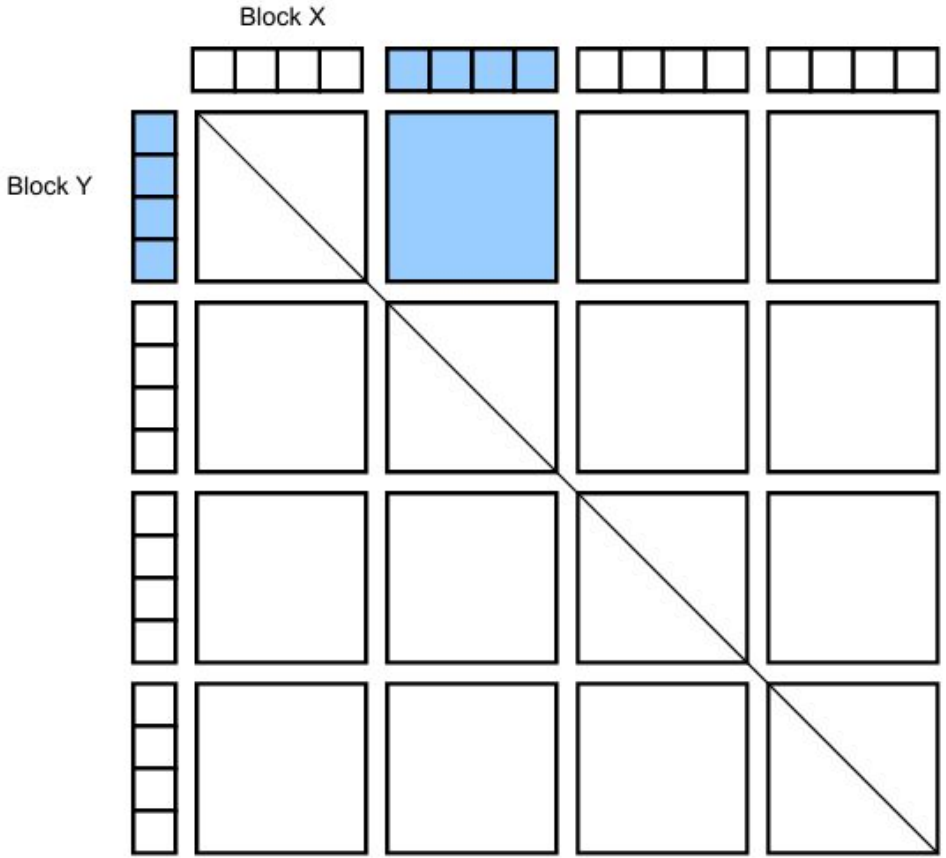
Algorithm Implementation



1. Load next cells into the buffers
2. Compute distances within buffers
3. Determine next block(s) to load
4. If all cells loaded in both dimensions then exit, otherwise repeat

- Both block dimensions can be changed at compile time

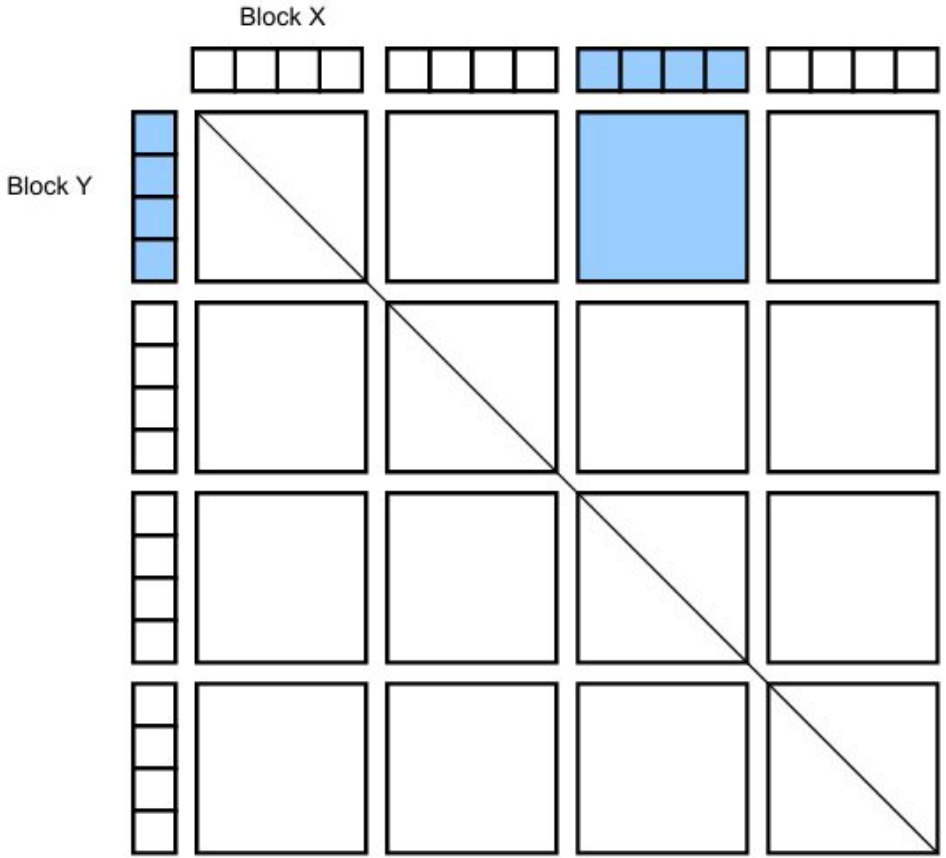
Algorithm Implementation



1. Load next cells into the buffers
2. Compute distances within buffers
3. Determine next block(s) to load
4. If all cells loaded in both dimensions then exit, otherwise repeat

- Both block dimensions can be changed at compile time

Algorithm Implementation



1. Load next cells into the buffers
2. Compute distances within buffers
3. Determine next block(s) to load
4. If all cells loaded in both dimensions then exit, otherwise repeat

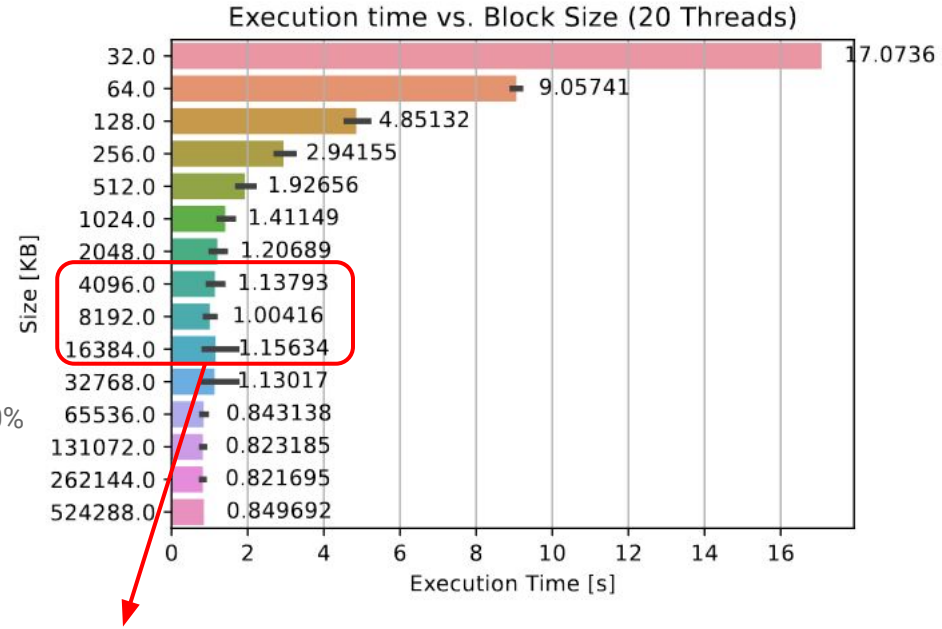
- Both block dimensions can be changed at compile time

Benchmark Setup

- Benchmarking:
  - Number of cells: 10^5
  - Running threads: 20
  - Average runs: 5
- Machine Specifications
  - Cores (w/ hyperthreading = 2): 56
  - Logical Processors: 112
  - Sockets: 2
  - Total L1d: 236 KB (56 instances)
  - Total L2: 730 KB (56 instances)
  - Total L3: 80,740 KB (2 instances)
- Formulas
  - **Cores Utilization** = Hyperthreading / #Threads = 2 / 20
  - **Element Bytes** = 2B
  - **Block Bytes** = Block Size X * Block Size Y * Element Bytes
  - **Bytes per Thread** = Block Bytes * Cores Utilization
  - **L2 Utilization** = Bytes per Thread / L2 Size

Block Byte Size vs. Execution Time

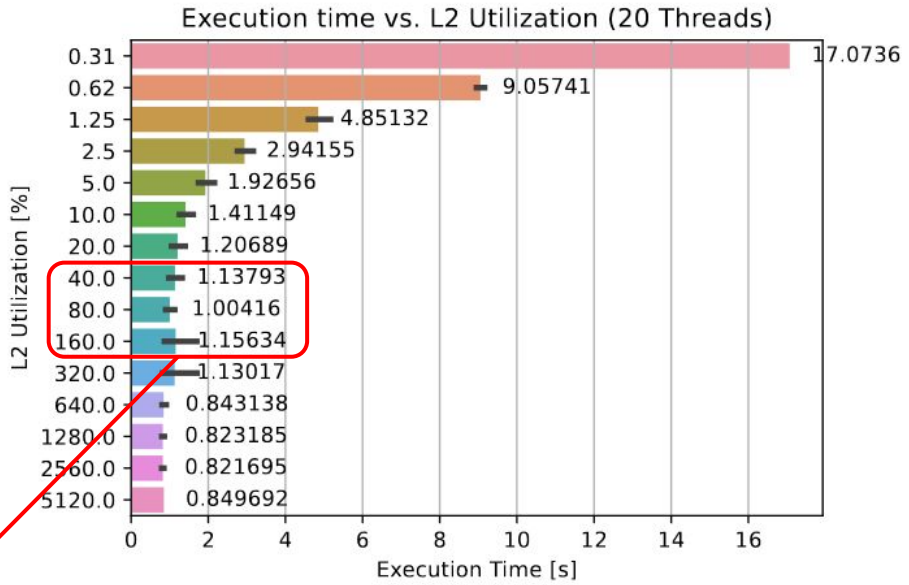
- Benchmarking:
  - Number of cells: 10^5
  - Running threads: 20
  - Average runs: 5
- Machine Specifications
  - Cores (w/ hyperthreading = 2): 56
  - Logical Processors: 112
  - Sockets: 2
  - Total L1d: 236 KB (56 instances)
  - Total L2: 730 KB (56 instances)
  - Total L3: 80,740 KB (2 instances)
- Formulas
  - **Cores Utilization** = Hyperthreading / #Threads = 2 / 20 ~10%
  - **Element Bytes** = 2B
  - **Block Bytes** = Block Size X * Block Size Y * Element Bytes
  - **Bytes per Threads** = Block Bytes * Cores Utilization
  - **L2 Utilization** = Bytes per Threads / L2 Size



8192 * 10% ~ 800KB, close to the total L2 size

L2 Utilization vs. Execution Time

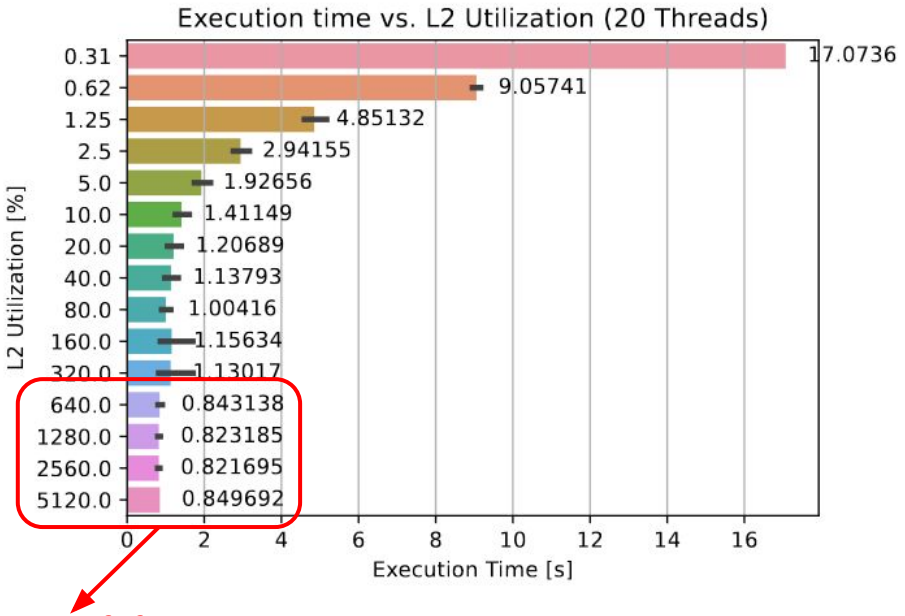
- Benchmarking:
  - Number of cells: 10^5
  - Running threads: 20
  - Average runs: 5
- Machine Specifications
  - Cores (w/ hyperthreading = 2): 56
  - Logical Processors: 112
  - Sockets: 2
  - Total L1d: 236 KB (56 instances)
  - Total L2: 730 KB (56 instances)
  - Total L3: 80,740 KB (2 instances)
- Formulas
  - **Cores Utilization** = Hyperthreading / #Threads = 2 / 20
  - **Element Bytes** = 2B
  - **Block Bytes** = Block Size X * Block Size Y * Element Bytes
  - **Bytes per Threads** = Block Bytes * Cores Utilization
  - **L2 Utilization** = Bytes per Threads / L2 Size



Sweet Spot: we find a local minimum at around 100%, as expected

L2 Utilization vs. Execution Time

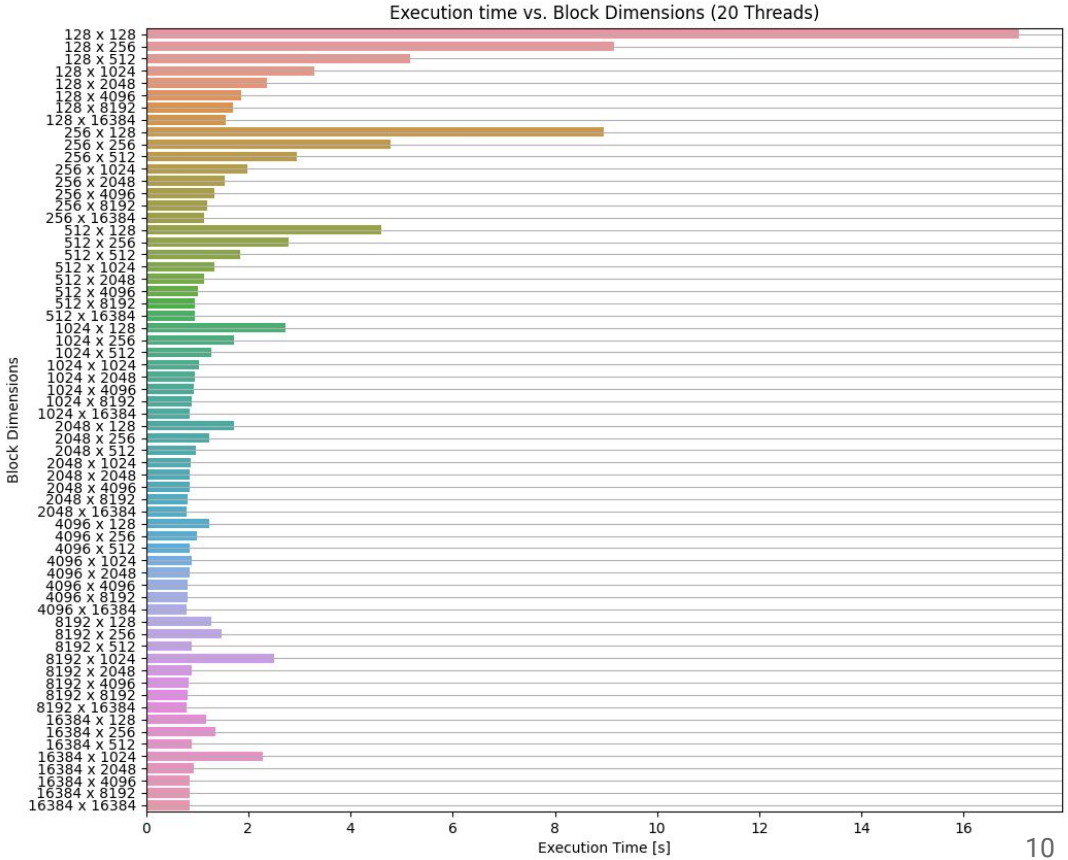
- Benchmarking:
  - Number of cells: 10^5
  - Running threads: 20
  - Average runs: 5
- Machine Specifications
  - Cores (w/ hyperthreading = 2): 56
  - Logical Processors: 112
  - Sockets: 2
  - Total L1d: 236 KB (56 instances)
  - Total L2: 730 KB (56 instances)
  - Total L3: 80,740 KB (2 instances)
- Formulas
  - **Cores Utilization** = Hyperthreading / #Threads = 2 / 20
  - **Element Bytes** = 2B
  - **Block Bytes** = Block Size X * Block Size Y * Element Bytes
  - **Bytes per Threads** = Block Bytes * Cores Utilization
  - **L2 Utilization** = Bytes per Threads / L2 Size



High perf. for larger block sizes might be due to prefetching, since the access pattern is very regular

Block Dimensions vs. Execution Time

- Benchmarking:
  - Number of cells: 10^5
  - Running threads: 20
  - Average runs: 5
- Machine Specifications
  - Cores (w/ hyperthreading = 2): 56
  - Logical Processors: 112
  - Sockets: 2
  - Total L1d: 236 KB (56 instances)
  - Total L2: 730 KB (56 instances)
  - Total L3: 80,740 KB (2 instances)
- Formulas
  - **Cores Utilization** = Hyperthreading / #Threads = 2 / 20
  - **Element Bytes** = 2B
  - **Block Bytes** = Block Size X * Block Size Y * Element Bytes
  - **Bytes per Thread** = Block Bytes * Cores Utilization
  - **L2 Utilization** = Bytes per Thread / L2 Size



Backups

“lscpu” Command

```
Architecture:                    x86_64
        CPU op-mode(s):          32-bit, 64-bit
        Address sizes:           46 bits physical, 48 bits virtual
        Byte Order:              Little Endian
CPU(s):                          112
        On-line CPU(s) list:     0-111
Vendor ID:                       GenuineIntel
        Model name:              Intel(R) Xeon(R) Platinum 8180 CPU @ 2.50GHz
                CPU family:      6
                Model:           85
                Thread(s) per core:  2
                Core(s) per socket:  28
                Socket(s):       2
Caches (sum of all):
        L1d:                     1.8 MiB (56 instances) 1887436.8 B
        L1i:                     1.8 MiB (56 instances) 1887436.8 B
        L2:                      56 MiB (56 instances) 58720256 B
        L3:                      77 MiB (2 instances) 80740352 B
```

“lscpu --caches” Command

NAME	ONE-SIZE	ALL-SIZE	WAYS	TYPE	LEVEL	SETS	PHY-LINE	COHERENCY-SIZE
L1d	32K	1.8M	8	Data	1	64	1	64
L1i	32K	1.8M	8	Instruction	1	64	1	64
L2	1M	56M	16	Unified	2	1024	1	64
L3	38.5M	77M	11	Unified	3	57344	1	64