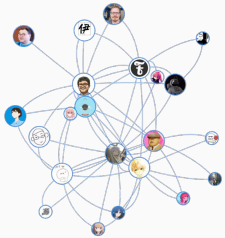
Optimizing Breadth-First Search on Modern Multicore CPUs

Student: Salvatore D. Andaloro
Supervisors: Flavio Vella, Thomas Pasquali

Department of Information Engineering and Computer Science, University of Trento

Breadth-First Search

- Breadth-First Search is a fundamental algorithm in graph analysis
- Used in many algorithms: Dijkstra, Maximum Flow, MSP...

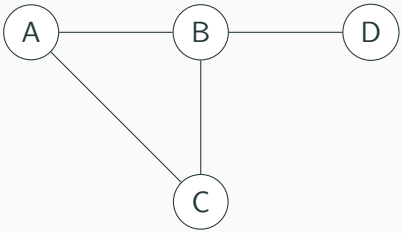


Social network

Road network

Breadth-First Search Example

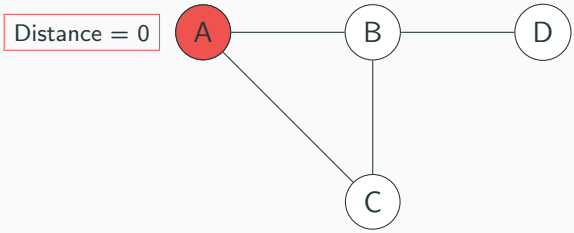
Vertices are labeled based on the distance from a given *source* vertex



Source vertex: A

Breadth-First Search Example

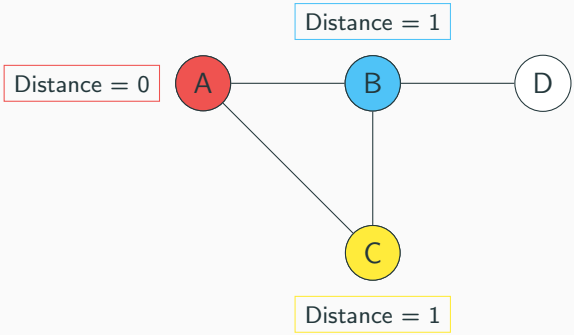
Vertices are labeled based on the distance from a given *source* vertex



Frontier: A

Breadth-First Search Example

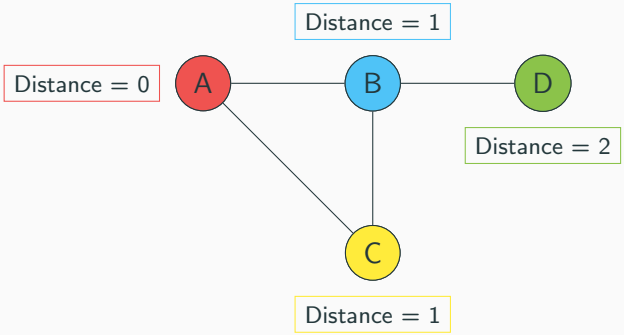
Vertices are labeled based on the distance from a given *source* vertex



Frontier: B, C

Breadth-First Search Example

Vertices are labeled based on the distance from a given *source* vertex



Frontier: D

Modern Computer Architectures

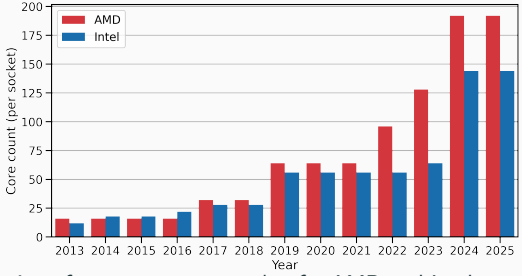
- BFS has $\mathcal{O}(V + E)$ time and space complexity (under RAM model)

Modern Computer Architectures

- BFS has $\mathcal{O}(V + E)$ time and space complexity (under RAM model)
- In practice, it is a **memory-bound algorithm**
  - Graph algorithms exhibits poor cache locality

Modern Computer Architectures

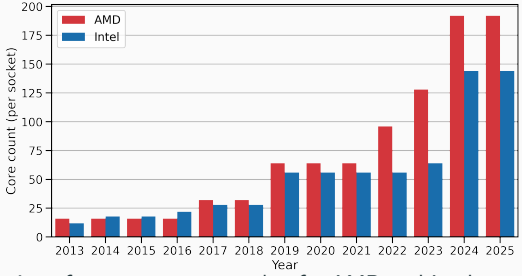
- BFS has $\mathcal{O}(V + E)$ time and space complexity (under RAM model)
- In practice, it is a **memory-bound algorithm**
  - Graph algorithms exhibits poor cache locality
- CPUs exhibit growing amount of **parallelism**...



Evolution of core counts per socket for AMD and Intel processors

Modern Computer Architectures

- BFS has $\mathcal{O}(V + E)$ time and space complexity (under RAM model)
- In practice, it is a **memory-bound algorithm**
  - Graph algorithms exhibits poor cache locality
- CPUs exhibit growing amount of **parallelism**...
- ...and new architectures are coming to the market (ARM, RISC-V)



Evolution of core counts per socket for AMD and Intel processors

Contents

- Propose an improved version of the *MergedCSR* data structure[1]

[1] Ryan Torok. ***Improving Graph Workload Performance by Rearranging the CSR Memory Layout.*** 2020.

Contents

- Propose an improved version of the *MergedCSR* data structure[1]
- Designed two optimized parallel implementations (OpenMP and pthreads)

HICREST

[1] Ryan Torok. ***Improving Graph Workload Performance by Rearranging the CSR Memory Layout.*** 2020.

Contents

- Propose an improved version of the *MergedCSR* data structure[1]
- Designed two optimized parallel implementations (OpenMP and pthreads)
- Evaluated against GAP Benchmark suite
- Speedups compared on three different architectures (AMD x86, RISC-V, ARM)

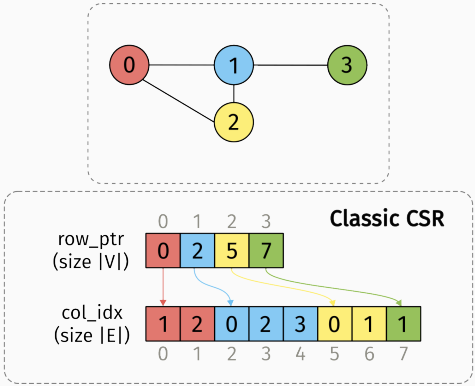
GAP suite logo

Compared architectures

[1] Ryan Torok. ***Improving Graph Workload Performance by Rearranging the CSR Memory Layout.*** 2020.

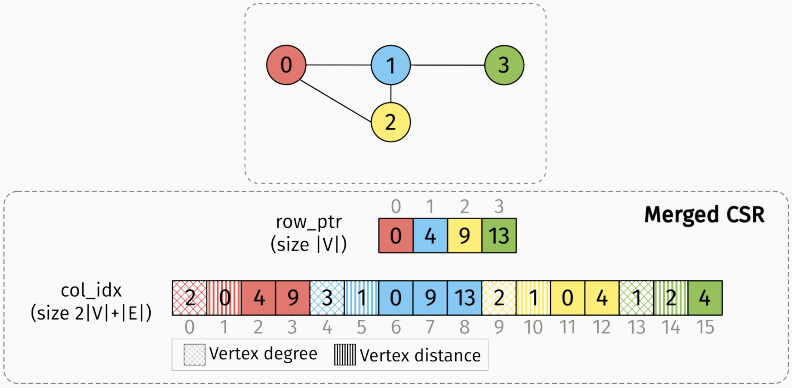
From CSR to MergedCSR

- Graphs are usually stored in the Compressed Sparse Row format (CSR)



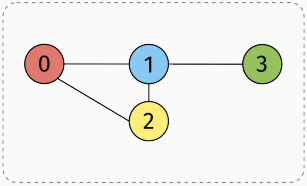
From CSR to MergedCSR

- Graphs are usually stored in the Compressed Sparse Row format (CSR)
- MergedCSR core idea: access only `col_idx` array during BFS traversal
  - `col_idx` array contains also algorithm-specific metadata (ex. distance)



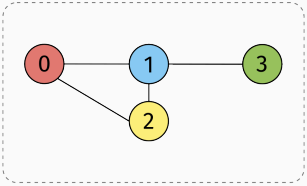
From CSR to MergedCSR

- Graphs are usually stored in the Compressed Sparse Row format (CSR)
- MergedCSR core idea: access only `col_idx` array during BFS traversal
  - `col_idx` array contains also algorithm-specific metadata (ex. distance)



From CSR to MergedCSR

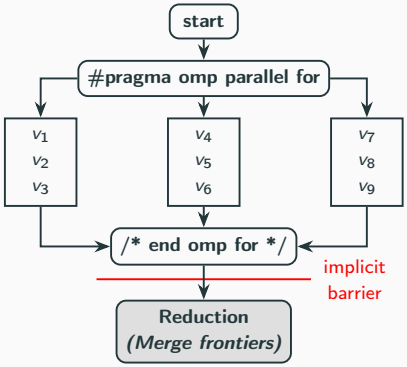
- Graphs are usually stored in the Compressed Sparse Row format (CSR)
- MergedCSR core idea: access only `col_idx` array during BFS traversal
  - `col_idx` array contains also algorithm-specific metadata (ex. distance)



OpenMP implementation

- **OpenMP** is a widely used framework for **parallel programming** in C and C++
- Uses simple compiler directives called pragmas

```
#pragma omp parallel for
↪  reduction(vec_add : next_frontier)
for (const auto &v : this_frontier) {
  // Process vertex v
}
```



Inefficiencies of the OpenMP implementation

- Merging step is not parallel

Inefficiencies of the OpenMP implementation

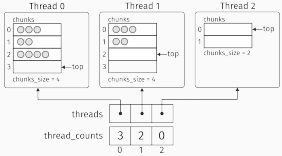
- Merging step is not parallel
- In graphs with large diameter, OpenMP enters the `parallel for` region more than 10k times for each BFS run on certain graphs

Pthreads implementation summary

- Pthreads: low-level threading library to create and manage threads in C

Pthreads implementation summary

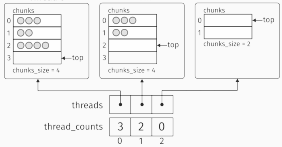
- Pthreads: low-level threading library to create and manage threads in C



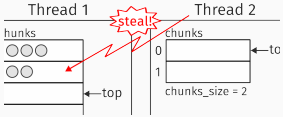
Custom frontier data structure

Pthreads implementation summary

- Pthreads: low-level threading library to create and manage threads in C



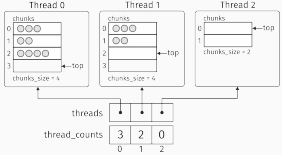
Custom frontier data structure



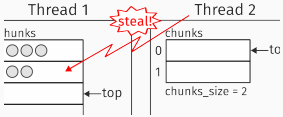
Work-stealing mechanism

Pthreads implementation summary

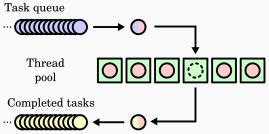
- Pthreads: low-level threading library to create and manage threads in C



Custom frontier data structure



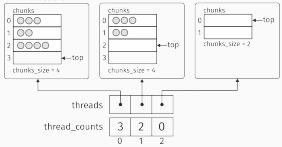
Work-stealing mechanism



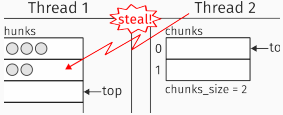
Thread pool

Pthreads implementation summary

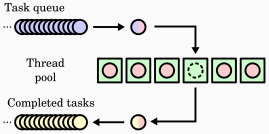
- Pthreads: low-level threading library to create and manage threads in C



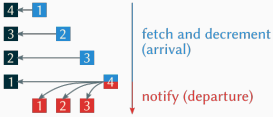
Custom frontier data structure



Work-stealing mechanism

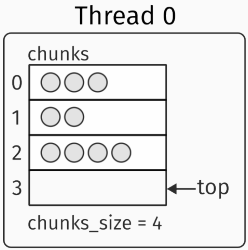


Thread pool

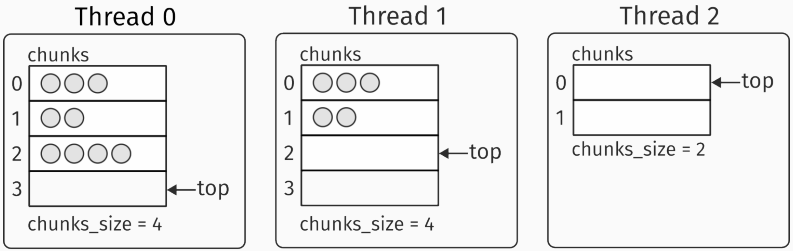


Custom barrier

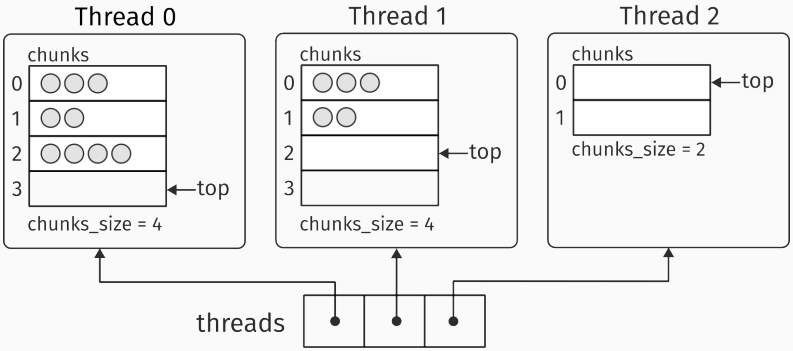
Frontier implementation



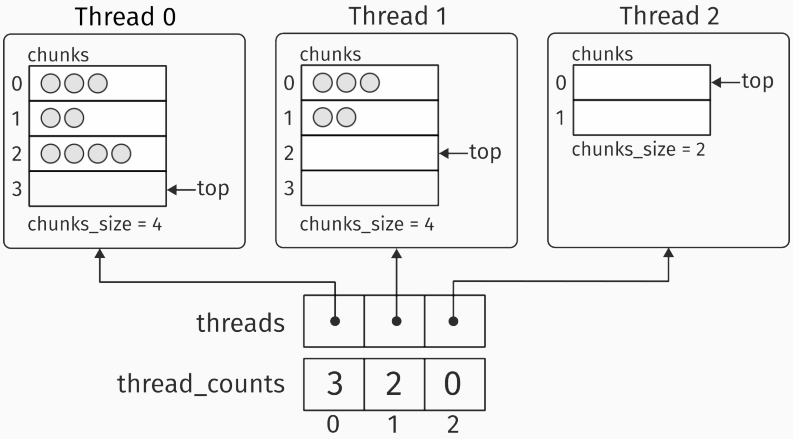
Frontier implementation



Frontier implementation

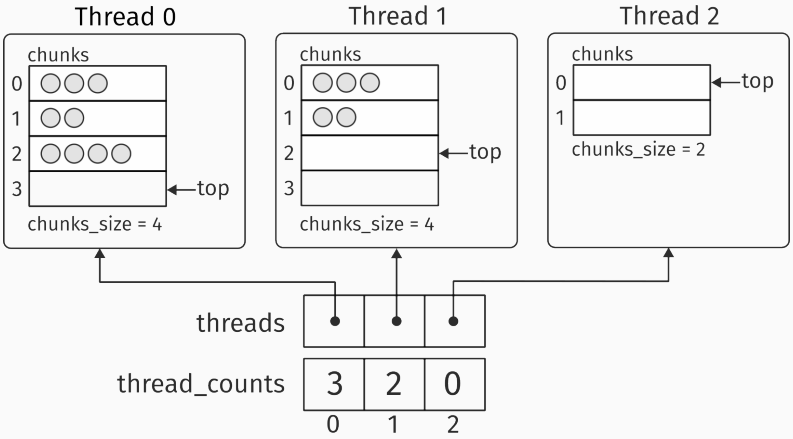


Frontier implementation



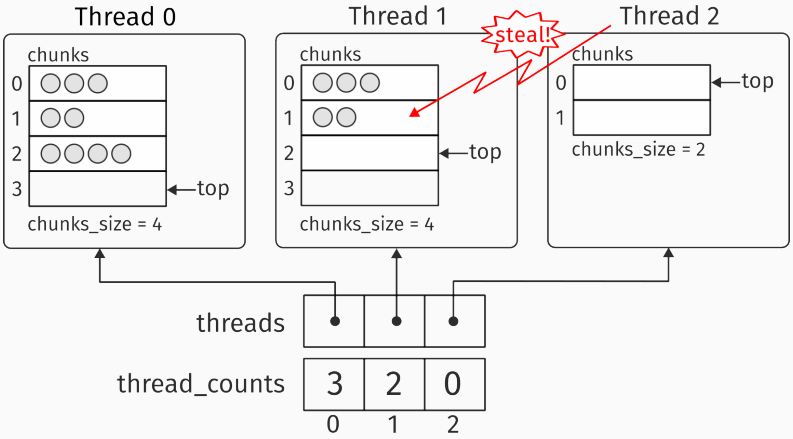
Work-stealing mechanism

Thread 2 is out of work! It will attempt a steal soon...



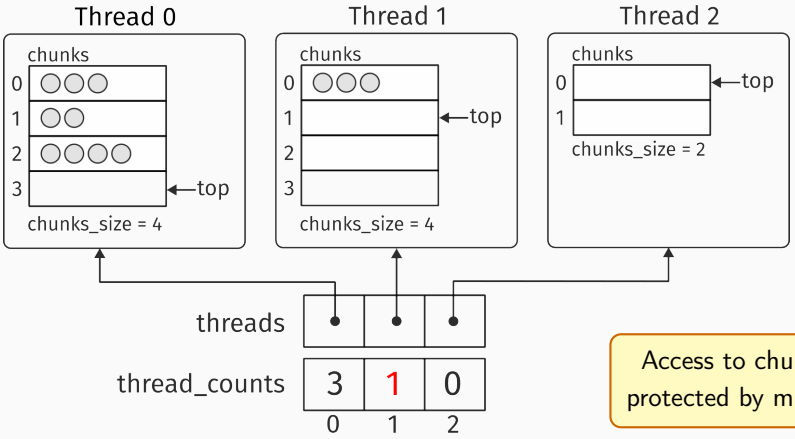
Work-stealing mechanism

Thread 2 steals a chunk of work from Thread 1...



Work-stealing mechanism

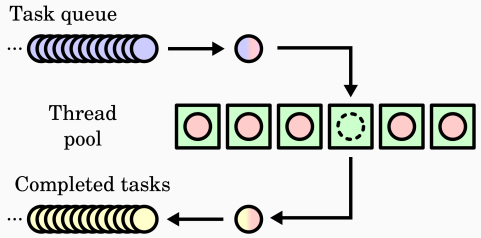
Thread 2 processes the stolen vertices and updates the global count.



Access to chunks is protected by mutexes.

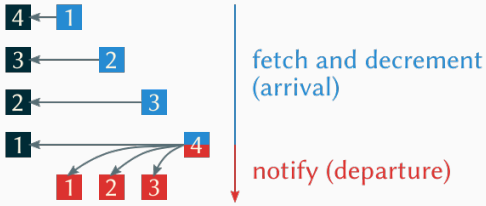
Thread pool

- Optimization to avoid repeated thread creation
- When the program is run, a group of threads is spawned
- At the beginning of each BFS run, the threads are awaken
  1. Process own chunks
  2. Steal work from other threads



Sense-Reversal Centralized Barrier

- Barrier: point that threads must reach before any can proceed
- Procedure:
  1. Threads decrease global counter and wait
  2. Last thread resets counter + increments distance
  3. All threads are released together



Experimental setup

- Experiments run on 3 platforms:
  - **x86**: AMD EPYC 7543 CPU @ 2.8 GHz (32 cores)
  - **RISC-V**: Sophon SG2042 CPU @ 2.0 GHz (64 cores)
  - **ARM**: NVIDIA Grace CPU Superchip @ up to 3.0 GHz (144 cores)

[2]Scott Beamer, Krste Asanović, and David Patterson. ***The GAP benchmark suite.*** 2015.

Experimental setup

- Experiments run on 3 platforms:
  - **x86**: AMD EPYC 7543 CPU @ 2.8 GHz (32 cores)
  - **RISC-V**: Sophon SG2042 CPU @ 2.0 GHz (64 cores)
  - **ARM**: NVIDIA Grace CPU Superchip @ up to 3.0 GHz (144 cores)
- Datasets: 3 road networks (USA, Europe, Asia), 3 FEM meshes (Earth's crust, steel hook, porous material), 1 random geometric graph (RGG)

[2]Scott Beamer, Krste Asanović, and David Patterson. ***The GAP benchmark suite.*** 2015.

Experimental setup

- Experiments run on 3 platforms:
  - **x86**: AMD EPYC 7543 CPU @ 2.8 GHz (32 cores)
  - **RISC-V**: Sophon SG2042 CPU @ 2.0 GHz (64 cores)
  - **ARM**: NVIDIA Grace CPU Superchip @ up to 3.0 GHz (144 cores)
- Datasets: 3 road networks (USA, Europe, Asia), 3 FEM meshes (Earth's crust, steel hook, porous material), 1 random geometric graph (RGG)
- Tools: GCC compiler, SBatchMan

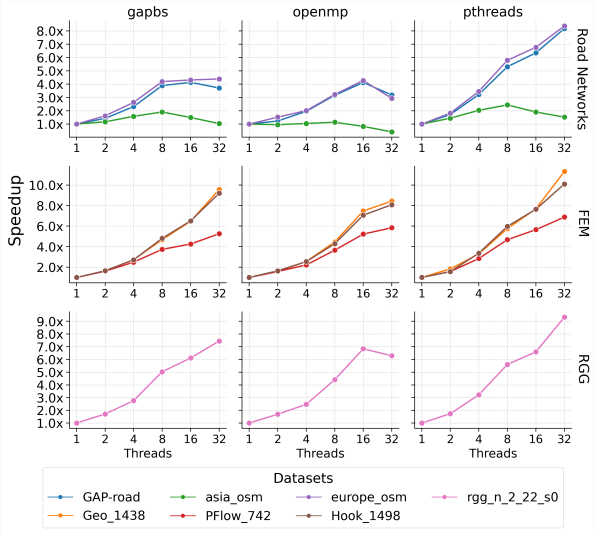
[2]Scott Beamer, Krste Asanović, and David Patterson. ***The GAP benchmark suite.*** 2015.

Experimental setup

- Experiments run on 3 platforms:
  - **x86**: AMD EPYC 7543 CPU @ 2.8 GHz (32 cores)
  - **RISC-V**: Sophon SG2042 CPU @ 2.0 GHz (64 cores)
  - **ARM**: NVIDIA Grace CPU Superchip @ up to 3.0 GHz (144 cores)
- Datasets: 3 road networks (USA, Europe, Asia), 3 FEM meshes (Earth's crust, steel hook, porous material), 1 random geometric graph (RGG)
- Tools: GCC compiler, SBatchMan
- Compared against the GAP benchmark suite[2]

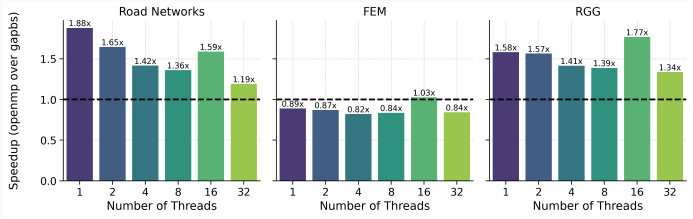
[2]Scott Beamer, Krste Asanović, and David Patterson. ***The GAP benchmark suite.*** 2015.

Scalability

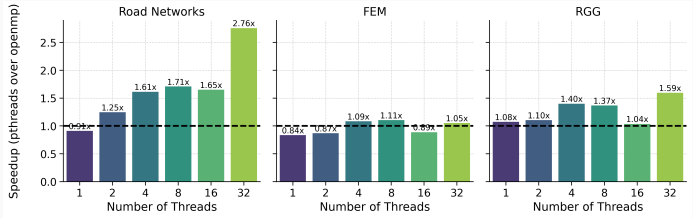


HICREST

Speedup

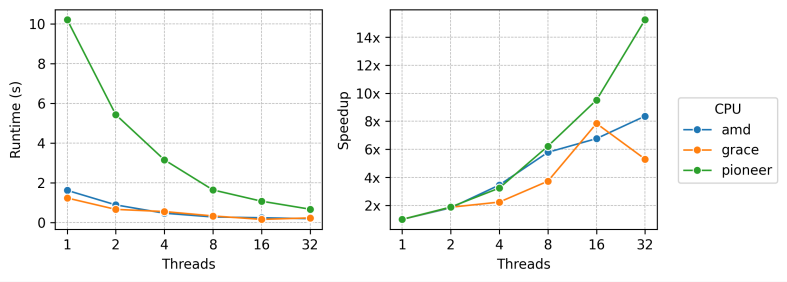


Speedup of the OpenMP implementation compared to the GAPBS implementation



Speedup of the pthreads implementation compared to the OpenMP implementation

Comparison on different architectures



Execution time and speedup on different architectures for the Europe road network dataset

Conclusions

- Designed two **multithreaded implementations** using an optimized CSR format and custom components tuned for today's multicore processors
- Compared them on different **architectures** (x86, RISC-V, ARM) and different **datasets**
- Achieved $\approx 1.5x$ geomean speedup for OpenMP and $\approx 2x$ speedup for Pthreads compared to the GAP benchmark suite
- Future work: explore other graph algorithms, optimize for more graph types
- Some of the results presented were developed for the Fastcode Programming Challenge 2025 @ PPoPP'25 ACM conference[3]

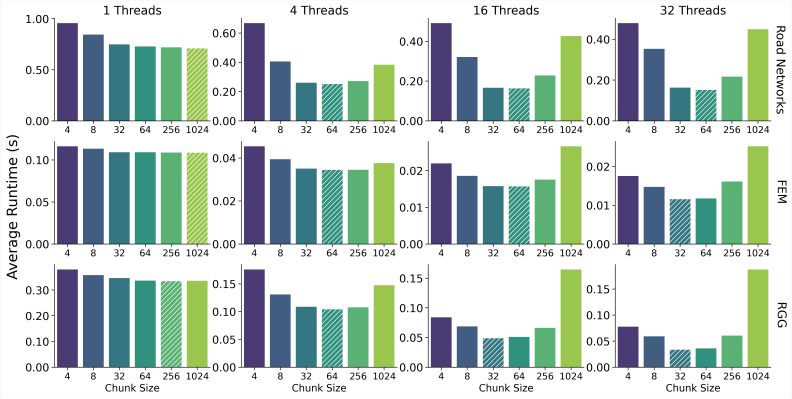
[3] Salvatore Andaloro, Thomas Pasquali, and Flavio Vella. **"Cache-optimized BFS on multi-core CPUs".** In: *Proceedings of the 1st FastCode Programming Challenge*. Association for Computing Machinery, 2025.

Thank You!

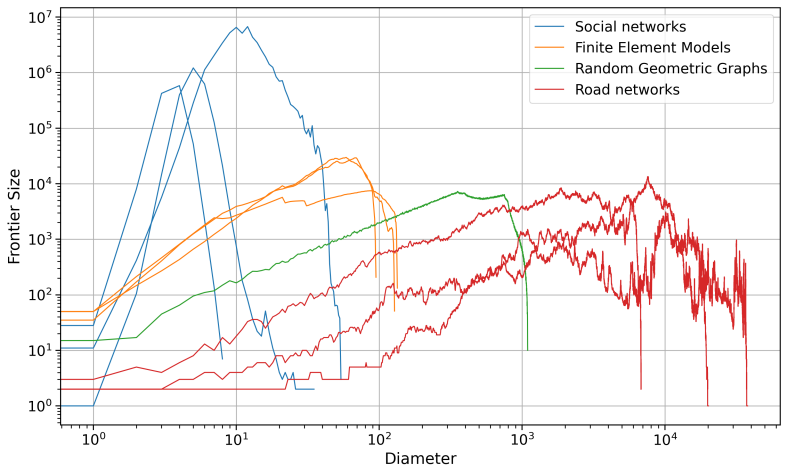
Questions?

Chunk size impact on performance

- Chunk size determines the number of vertices in a chunk
- Chunk sizes of 32 and 64 are optimal for most datasets in multithreaded environments



Frontier sizes for different datasets



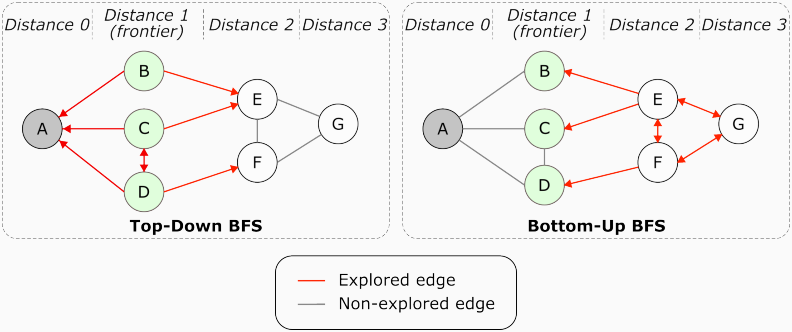
The evolution of BFS frontier size for different graphs, colored by their graph class.

Datasets used

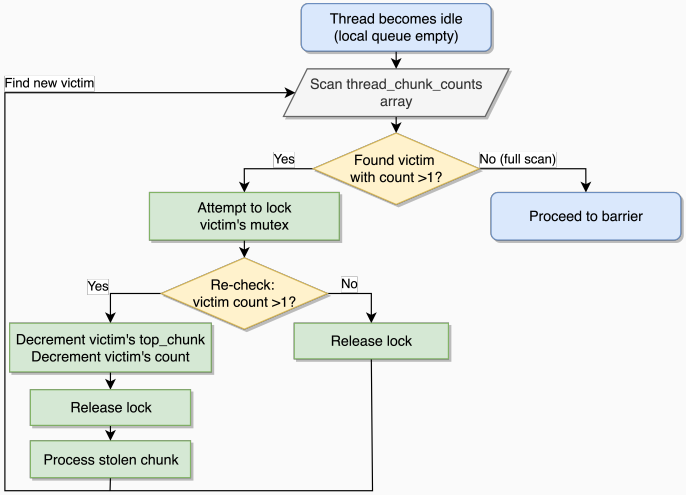
Name	Graph class	Vertices	Edges	Notes
GAP-Road	Road Network	23.9M	57.7M	Road network of the USA.
europe_osm	Road Network	50.9M	108M	Road network of Europe.
asia_osm	Road Network	11.9M	25.4M	Road network of Asia.
Geo_1438	FEM	1.4M	60.2M	Geomechanical model of Earth's crust.
Hook_1498	FEM	1.4M	59.4M	3D model of a steel hook.
PFlow_742	FEM	0.7M	37.1M	3D pressure-temp in porous media.
rgg_n_2_22_s0	RGG	4.2M	60.7M	Random Geometric Graph.

Datasets used in the evaluation.

Top-Down and Bottom-Up approaches



Work-Stealing protocol



Thread pool implementation