Friedrich-Alexander-Universität
Technische Fakultät

Computer Architecture Practical Exercise

6 Cache Blocking

Kenan Gündogan[1] Philipp Gündisch[1]

[1]Friedrich-Alexander Universität Erlangen-Nürnberg, Chair of Computer Science 3 (Computer Architecture)

December 5, 2024

Caches & 2D Jacobi

Motivation

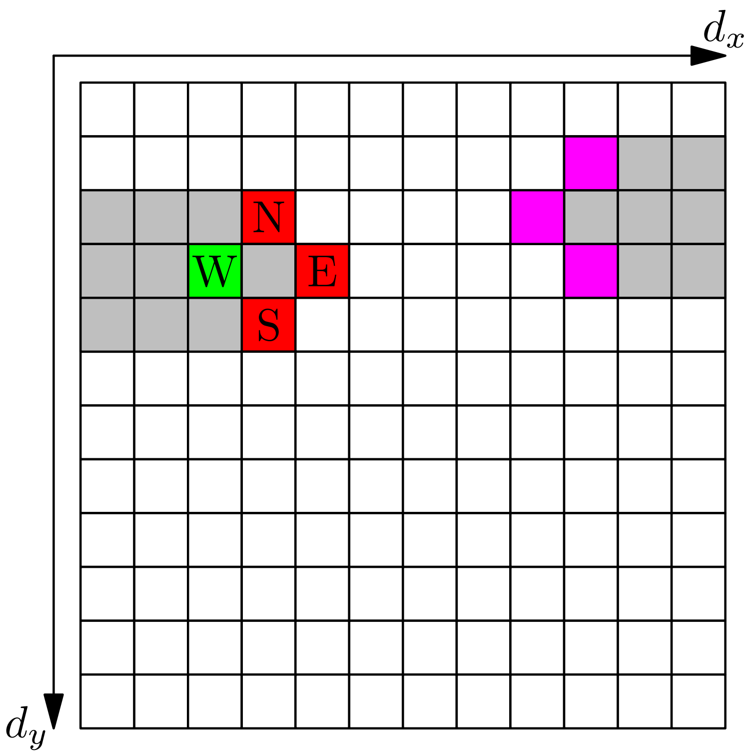
- Physical simulations like Jacobi are time and space discrete
- The higher the grid resolution the more accurate is the simulation
- But: A High resolution leads to an increased memory consumption exceeding the CPU cache capacity significantly
- Solution:
  1. Distributed computing across cluster nodes (not part of this module, see PACL)
  2. Grid size per node can still be in the GBs, vastly exceeding the cache sizes
- Are caches of any help at all in this scenario?

Caches & 2D Jacobi

Are relatively small caches helpful?

Yes! Even if the grid is too large to fit into the L3 cache, caches can still be helpful.

1. The cell to the west (**W**) can be loaded from the cache since it has been used two iterations ago
2. Ideally, also the cells to the north (**N**) and east (**E**) can be loaded from cache since they have been recently used
3. Whether or not the cells are still available in the cache depends on the grid width d_x, the cache size and the cache replacement strategy
4. Only the cell to the south (**S**) needs to be loaded from RAM

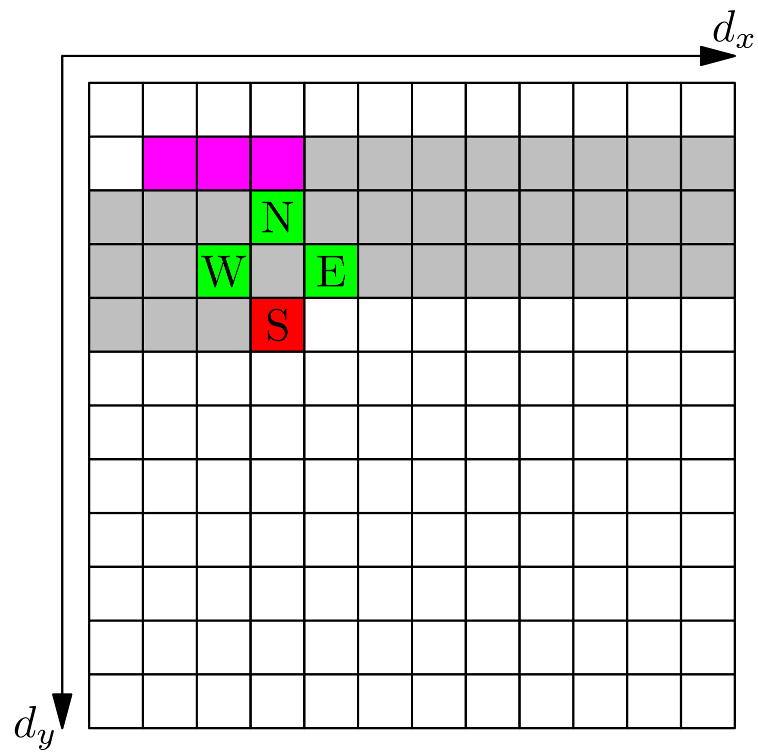


Caches & 2D Jacobi

Layer Condition

To ensure the cells can be read from the cache we formulate a so called **layer condition**.

1. The north (**N**) and east (**E**) neighbor cells can be loaded from cache if the cache is big enough to keep three full grid lines (+1)
2. If c_i is the cache capacity of the L_i-Cache in bytes then the two cells can be read from the cache if $c_i > (3 \cdot d_x + 1) * 8B + Overhead$
3. Note: *Overhead* consists of OS background, cached instructions, ...
4. What about the second grid?
5. Hardware integrated prefetchers also have negative effects on the layer condition (next week)



Example showing a satisfied layer condition.

Task 6.1: Cache Sizes

- Fill out the table below
- What maximum block width d_x (in cells) satisfies the layer condition?
- What minimal jacobi grid size (in bytes) is needed to benefit from the blocking technique?
- Fill the following table

	Cache Size	d_x	Grid Size
L1	48 kiB		
L2	1280 kiB		
L3	54 MiB		

Task 6.2: Cache Effects

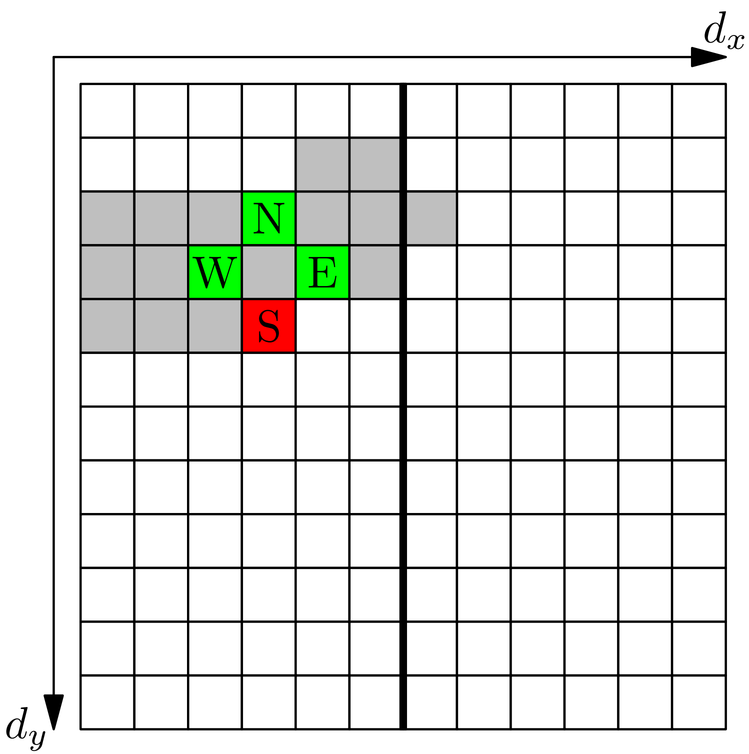
Show that it is very crucial for the performance what data is loaded from which cache.

- Benchmark the AVX 256 bit vectorized jacobi implementation
- This time, benchmark from 1GiB to 128GiB
- Create the performance plots as usual with MUp/s and ArraySize as axis
- Confirm the plots with the theoretical grid sizes of task 6.1
- **IMPORTANT: Do NOT draw the .ppm files for arrays of this size**

Cache Blocking

To improve the AVX 256 bit vectorized jacobi implementation even further we use a cache blocking technique known as **Spatial Blocking**.

1. Loop iterations are artificially limited (blocked) such that cache data can be reused
2. The inner most loop will be split into two loops
3. The outer loop runs over i_b blocks, each of width b_x (except the last block)
4. The inner loop is limited to run from $i_b \cdot b_x$ to $(i_b + 1) \cdot b_x$
5. Hint: That makes three nested loops in total (i_b, y, b_x)
6. Hint: Ensure checking the results for correctness



Task 6.3: Jacobi Spatial Blocking

- Update the AVX 256 bit jacobi implementation with spatial blocking
- Benchmark from 1GiB to 128GiB
- Make the blocking factor b_x a `#define` constant
- Determine a blocking factor b_x which satisfies the L1 layer condition
- Determine a blocking factor b_x which satisfies the L2 layer condition
- Ensure checking the results for correctness
- Extend the plot from task 6.2 with the results of these two versions
- *Hint: Consider only 80% of the cache size to be available for blocking*
- *Hint: 128GiB will take significantly longer than 1s for 1 iteration*
- *Hint: 50K x 50K grid:* $\geq 750 \cdot 10^6$ *Updates/s*
- *Hint: L1 cache blocking might not be as fast as L2 cache blocking*
- **IMPORTANT: Do NOT draw the .ppm files for arrays of this size**

Task Overview

- E 6.1: Cache Blocking
  - Use the layer condition to fill the table
- E 6.2: Cache Levels and Performance `gprof`
  - Benchmark the existing AVX implementation from 1MiB - 64GiB
- E 6.3: Cache Blocking
  - Extend the jacobi AVX implementation with spatial blocking
  - Plot the results to show the difference to the not blocked variant
  - Interprete the results

Appendix: Checklist

Performance Optimization (1/2)

During the timeline of this class new bullet points will be added. Recently added entries are bold.

- Compiling
  - Choice of the compiler (`icx`)
  - Compiler flag to optimize aggressively (e.g. `-O3`)
  - Compiler flag to adapt for specific hardware (e.g. `-xHost`)
- Programming Techniques (if applicable)
  - Use `#define` and `const` instead of variables
  - Data type aware programming
  - Use aligned memory (e.g. with `_mm_malloc()` or `posix_memalign()`)
  - Consecutive address iteration
  - Variable declarations outside of loops
  - Reduce function calls
  - Use intrinsics (to utilize SIMD)
  - **Cache aware programming (Spatial Blocking)**

Appendix: Checklist

Performance Optimization (2/2)

During the timeline of this class new bullet points will be added. Recently added entries are bold.

- Measurement
  - Reasonable benchmark time
  - Reasonable benchmark workload
  - Reduce interference factors to a minimum
- Optimization Process
  - Check assembler code while optimizing
  - Check performance gains while optimizing
  - Use profiling tools
  - Ensure correctness of code
  - Optimize iteratively