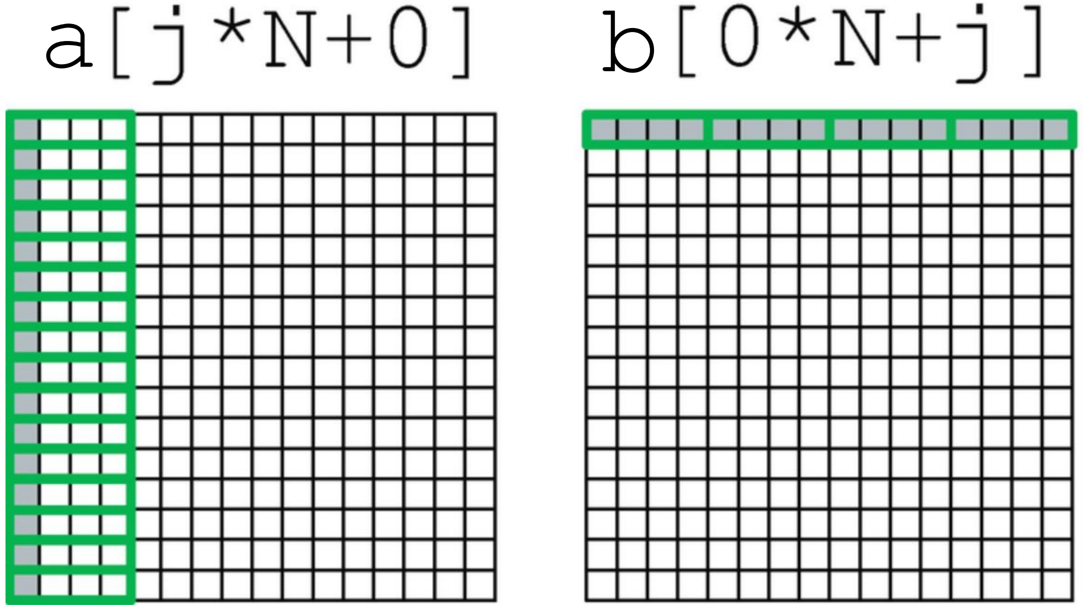
Prof. Dr. Robert Strzodka

Matrix Transpose 2D

Parallel Algorithm Design

Outline

Transpose Algorithm



- switch rows and columns

  → memory bound

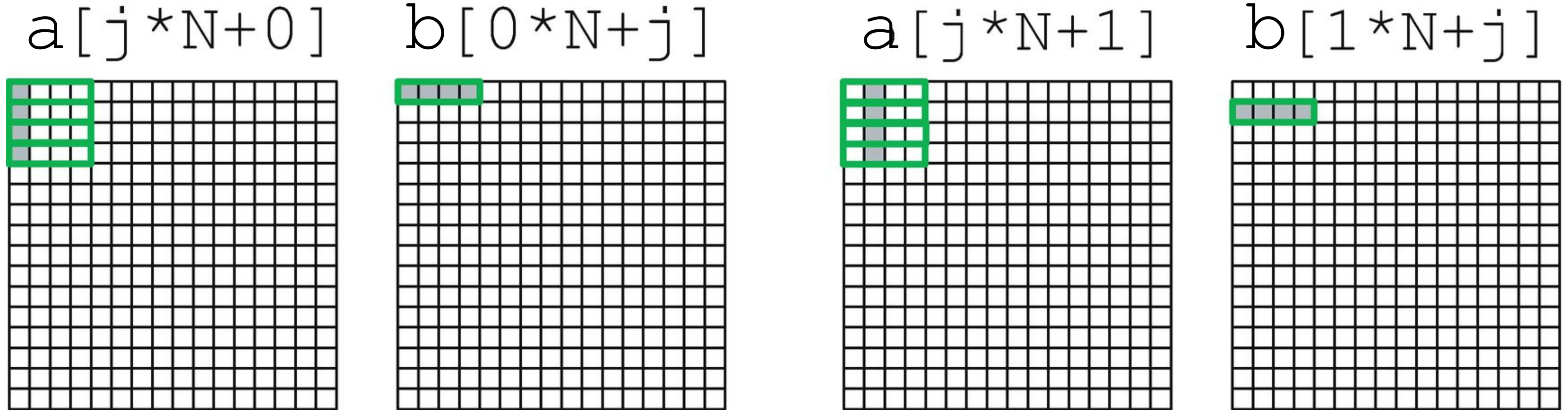
  → stride access

- no use of locality in naive version

  → solution: tiling

Tiling

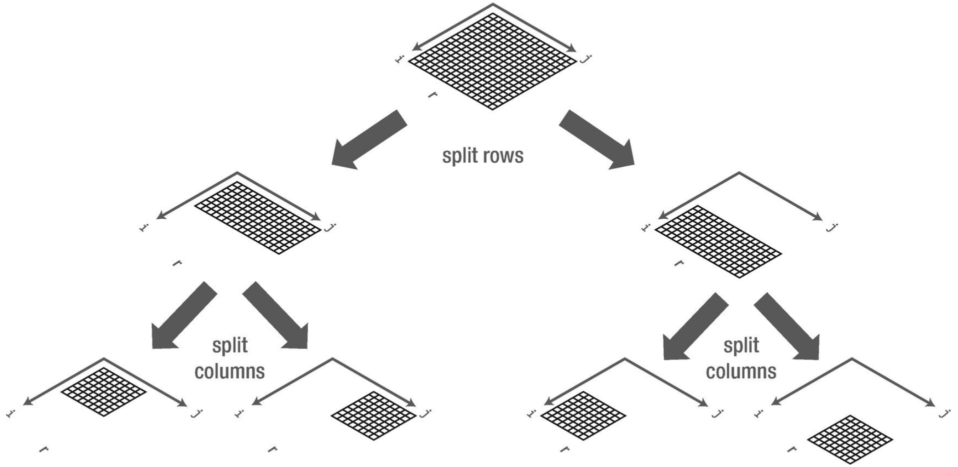
- use of spatial locality
- tiles must fit in cache
- additional control flow instructions

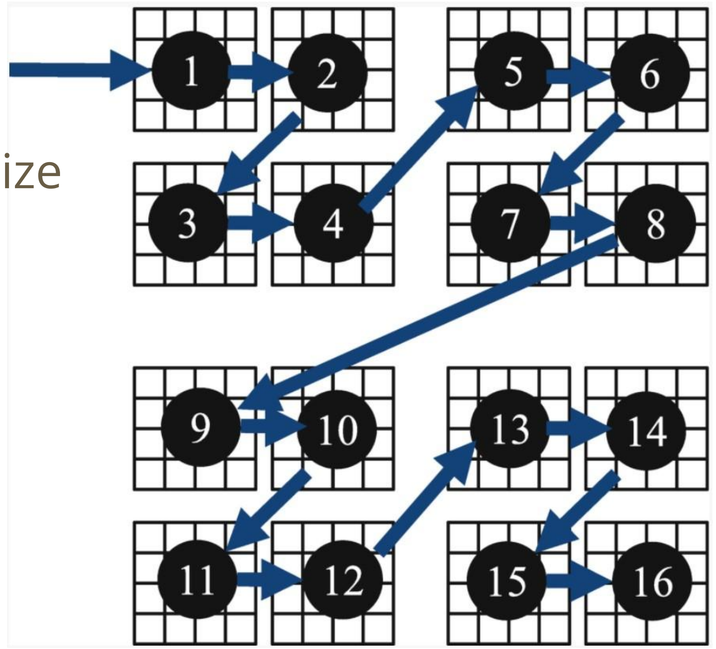


Cache-Oblivious Transpose

TBB: blocked_range2d + simple_partitioner

- termination condition defined by grain size





→ traversal pattern

Used Hardware:

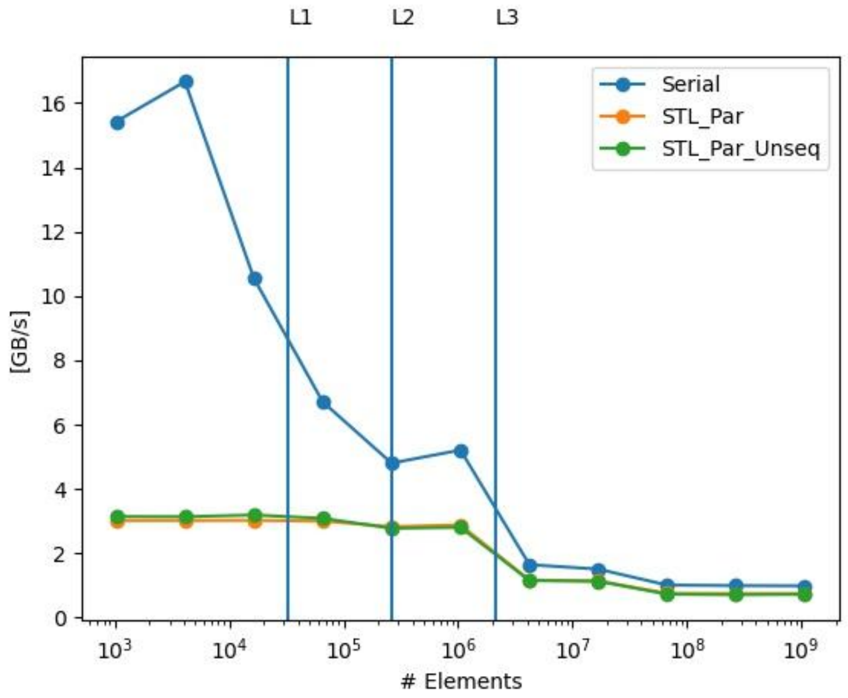
	mp-media (Intel Xeon E3-1585 v5)	ziti-rome (AMD Epyc 7662)
base frequency	3900 MHz	2154.3 MHz
NUMA nodes	1	4
cores	4 physical 8 logical	64 physical 128 logical
cache sizes	4x 32KB L1 4x 256KB L2 1x 8MB L3	64x 32KB L1 64x 512KB L2 16x 16MB L3
memory bandwith (read & write concurrently)	18.2 GB/s	98.9 GB/s

Memory Benchmark STL

- Transposition with std::for_each() and parallel execution policy

- leads to very bad results

- std::for_each() hard to use for matrix transposition

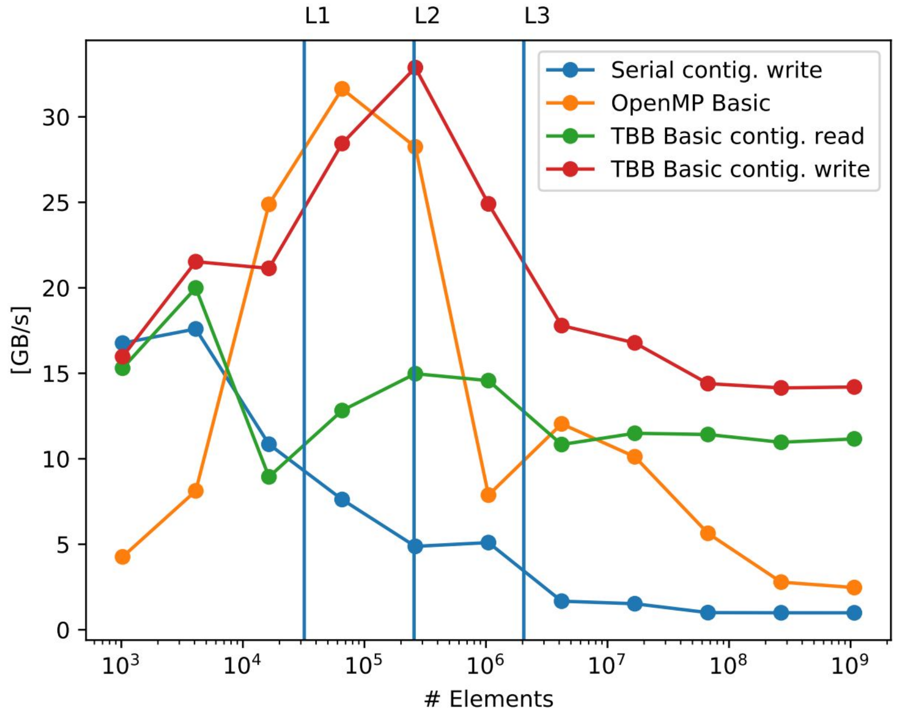


Memory Benchmark Media

- TBB:
  - simple partitioner
  - grain size: 64
- OMP:
  - dynamic scheduling

- basic TBB version performs better

  → use of locality

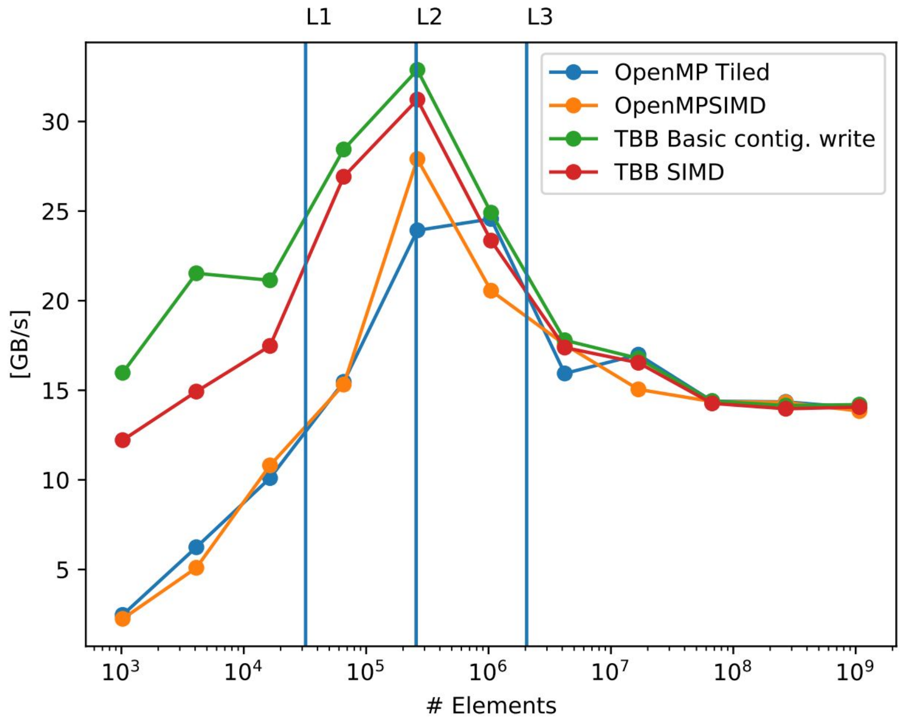


Memory Benchmark Media Tiling

- TBB:
  - simple partitioner
  - grain size: 64
- OMP:
  - dynamic scheduling
  - block size: 64
  - speedup of 5.7 to OMP basic

- no effect by using OMP SIMD

75% peak mem. performance

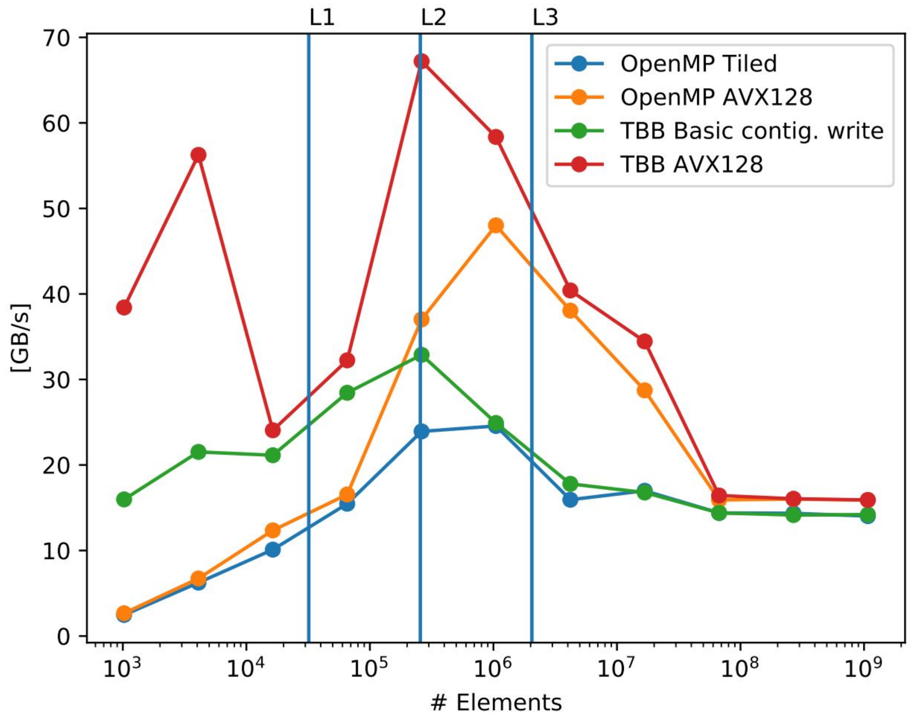


Memory Benchmark Media AVX Intrinsics

- TBB:
  - simple partitioner
  - avx with grain size: 128
- OMP:
  - dynamic scheduling
  - avx with block size: 128

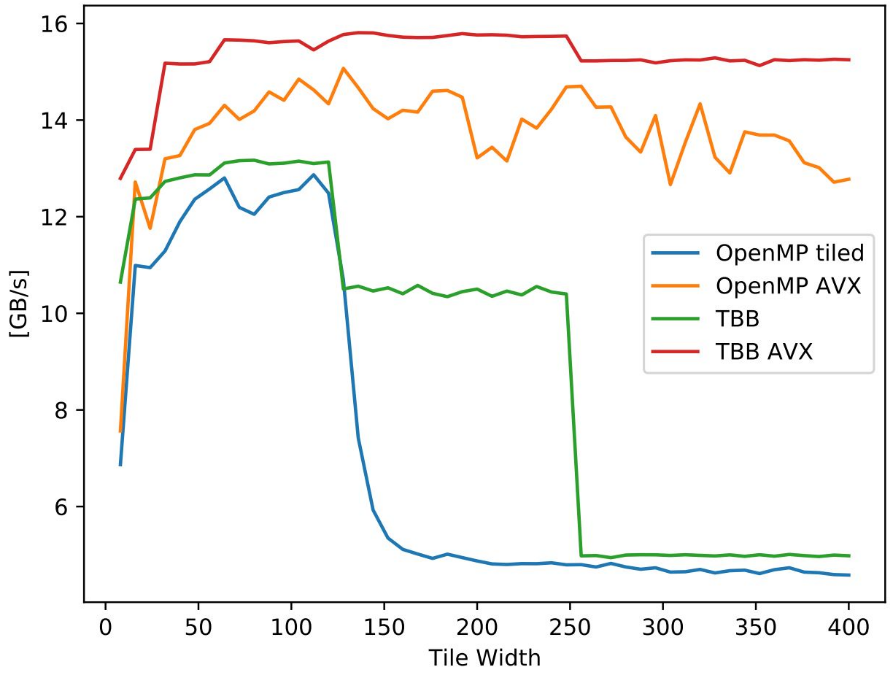
87% peak mem. performance

(non avx versions tile size 64)



Memory Benchmark Media Optimal Tile Size

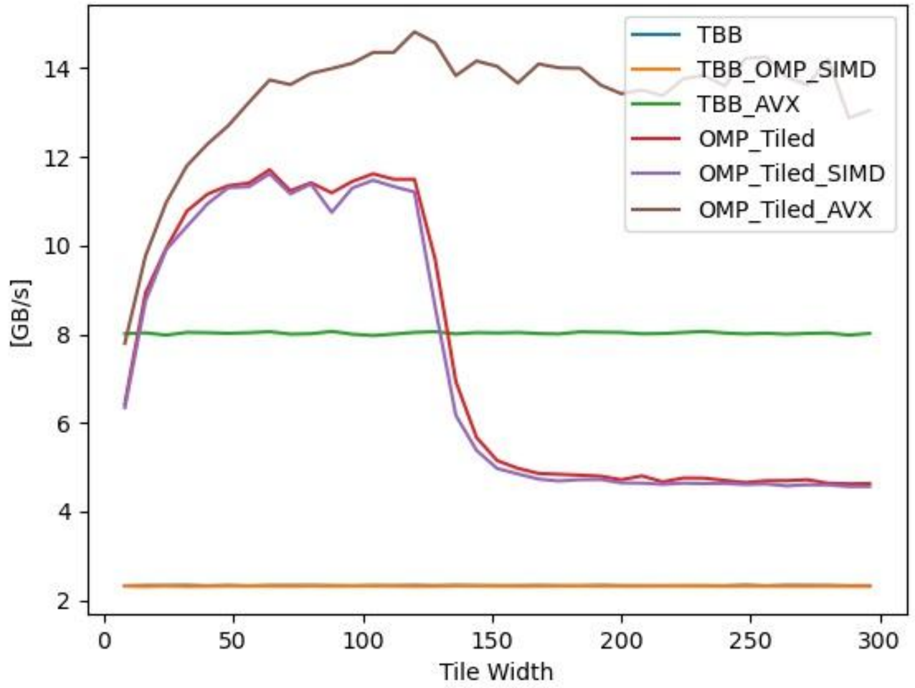
- OMP tiled version extremely bad with wrong tile size
- TBB performs worse when limiting traversal pattern to big tiles
- AVX versions quite stable
    - because of second layer of tiling
    - 8x8 blocks with 256bit registers



Memory Benchmark Media Static Scheduling

- static partitioner with TBB breaks tiling
  → no control over tile size

- static scheduling for OMP slightly worse performance



Memory Benchmark Rome

- with best media config

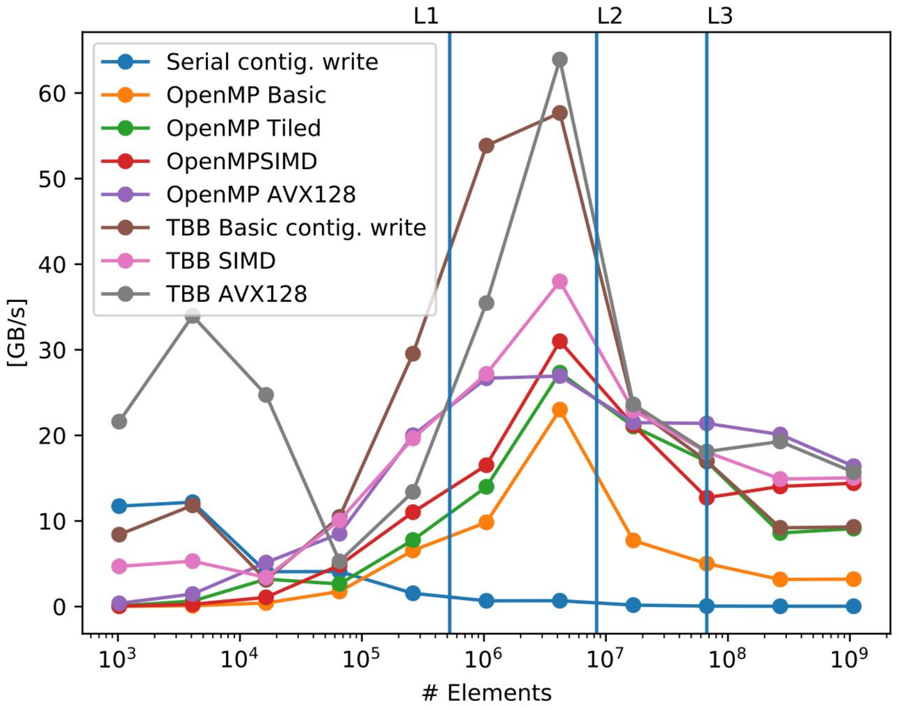
  TBB:

  - simple partitioner
  - avx with grain size: 128

  OMP:

  - dynamic scheduling
  - avx with block size: 128

- bad performance
  → NUMA effects

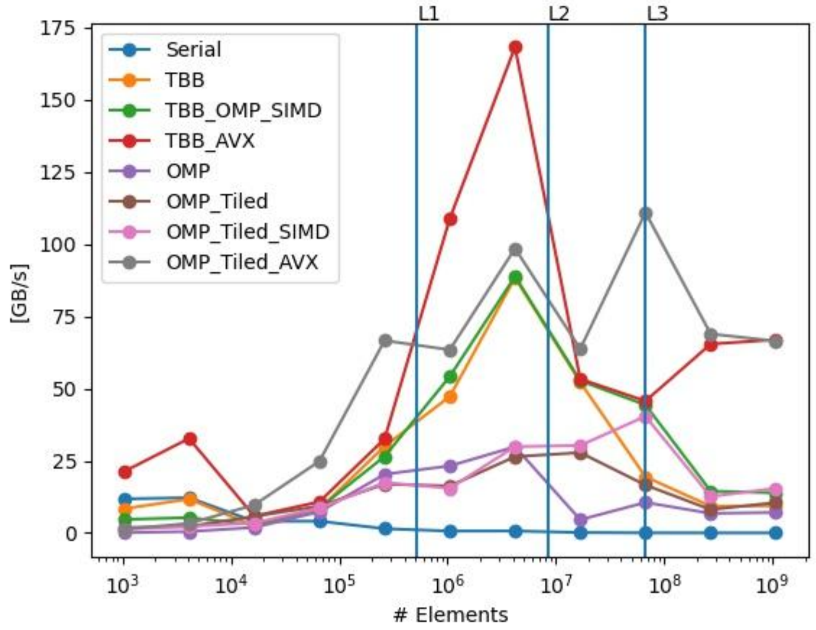


Memory Benchmark Rome

- NUMA aware data placement (static scheduling for OMP)

- TBB still simple partitioner

- non intrinsic versions still bad



Memory Benchmark Rome

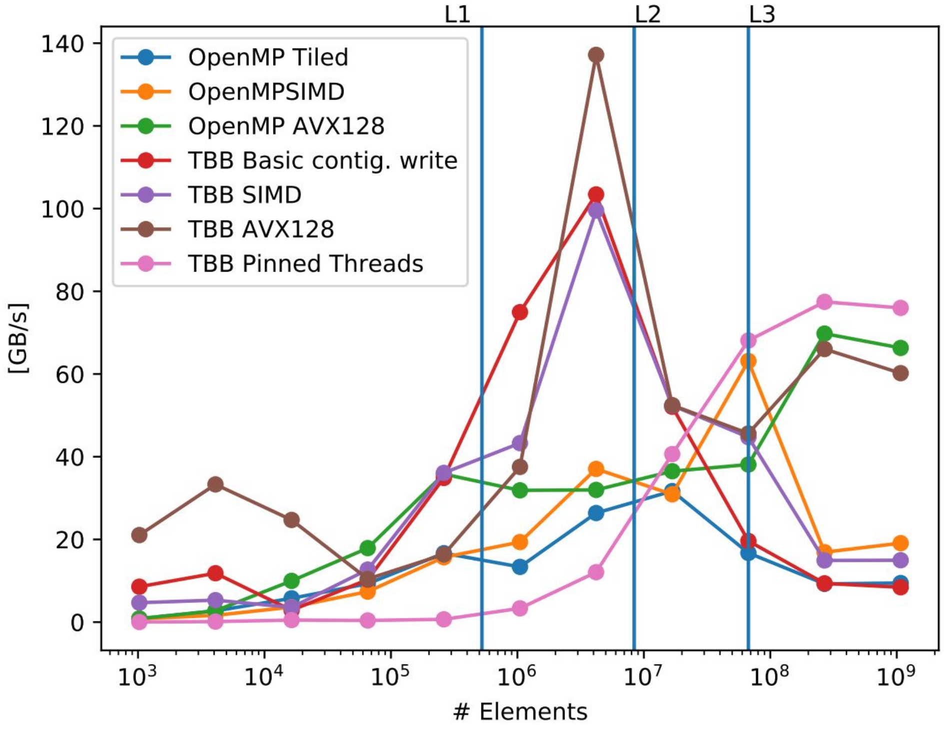
- additionally use of OMP Places
  → threads locked to cores
  → slight improvement

- for TBB version with hwloc

  → matrix divided in 4 parts

  → computed separately
  in NUMA regions

- wrong tile size on new machine?

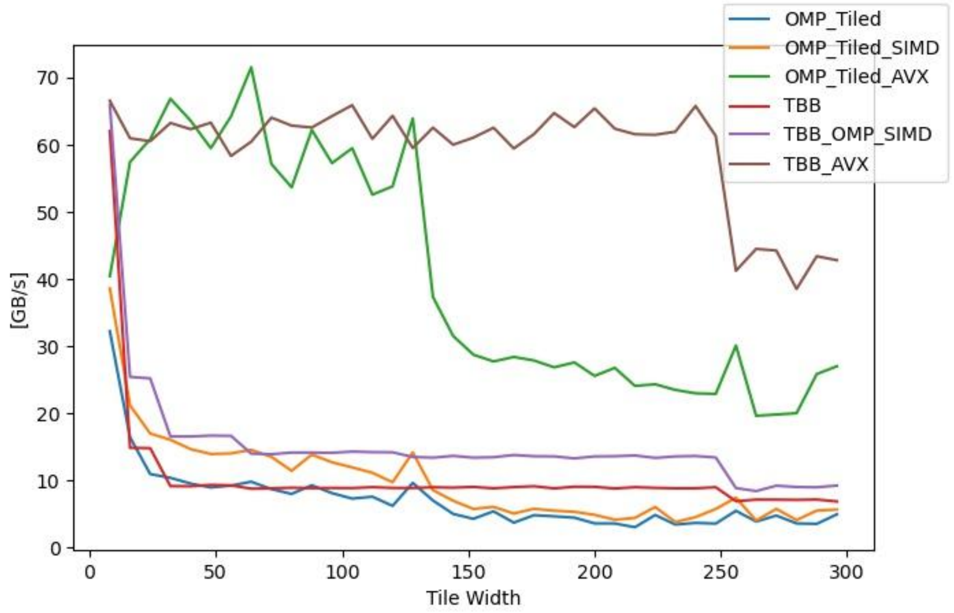


Memory Benchmark Rome Optimal Tile Size

- matrix size fixed to 2^{15} x 2^{15}

- small tiles perform very good

  → 8 x 8 looks promising

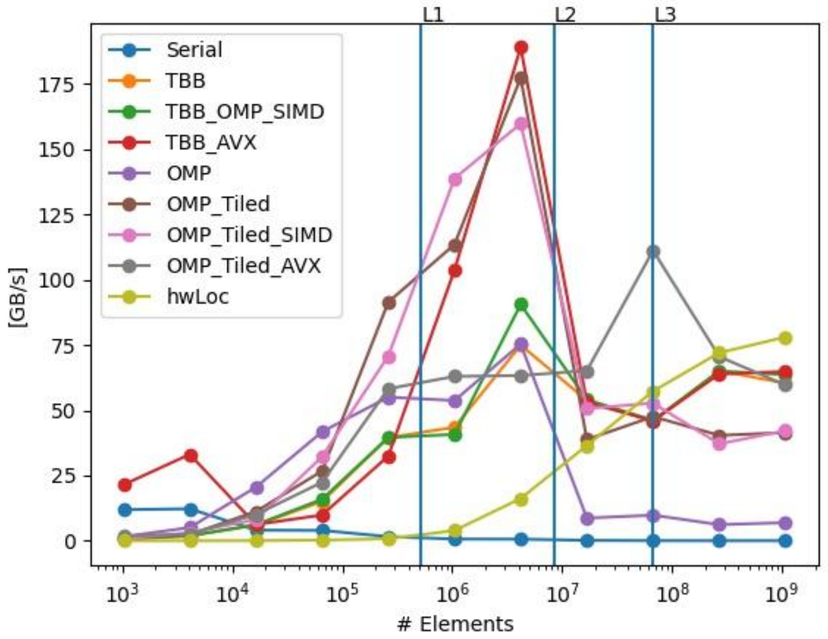


Memory Benchmark Rome Optimal Tile Size

- significant performance increase with 8x8 tiles

- AVX versions still use 64x64 tiles
  <u>but:</u> internal 8x8 tiling

- 80% peak performance with hwloc (third layer of tiling)

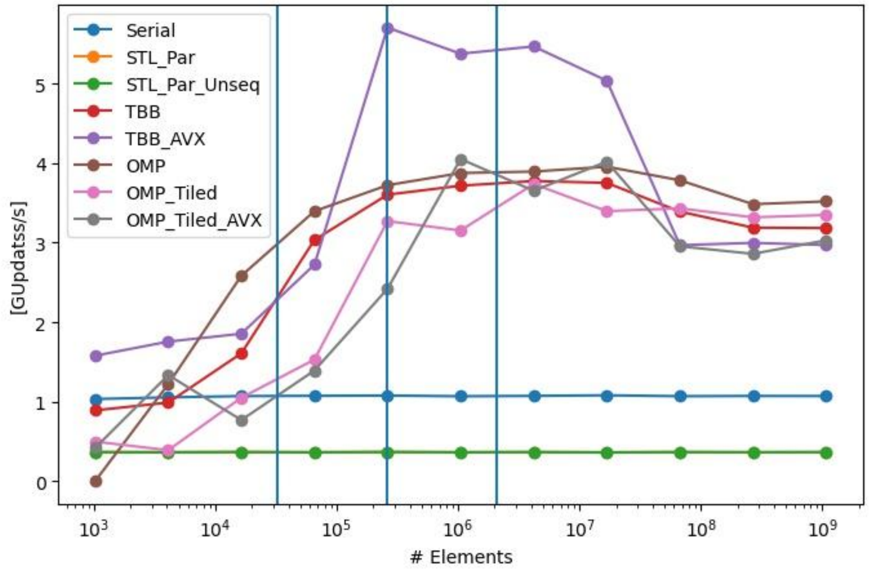


Computational Media

- same parameters from memory benchmark

- basic OMP version best
  - no stride in input matrix
  - contiguous write in output matrix

- overhead in AVX version
  → fill phase of local arrays

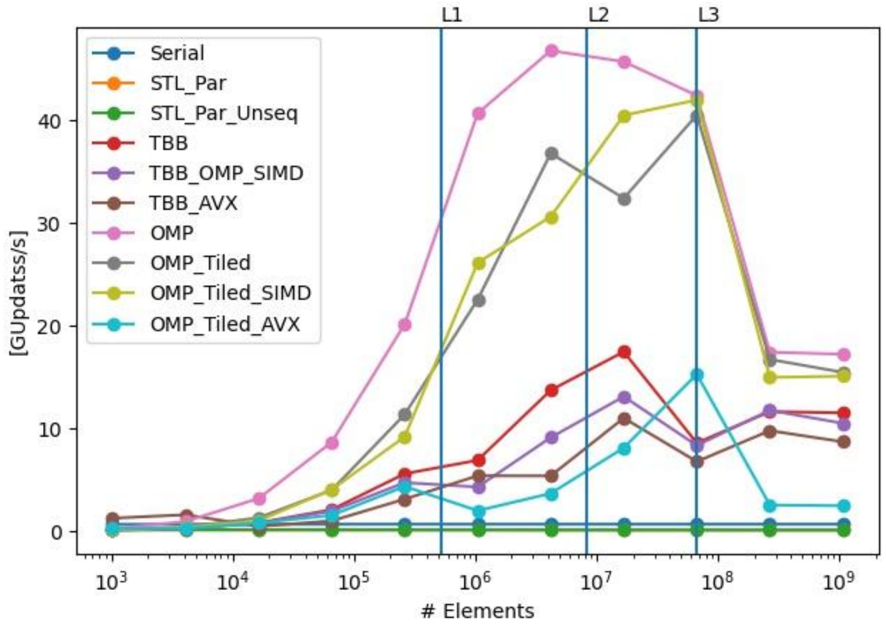


Computational Rome

- same parameters from memory benchmarks (8x8 blocks)

- OMP Basic still best performance

  →maximum write performance



Thanks for your attention