Research of techniques to improve the performance of explicit numerical methods on the CPU

Authors: A. Ivanov, V. Furgailo, N. Khohlov

Speaker: Vladislav Furgailo

Velikiy Novgorod, 2019

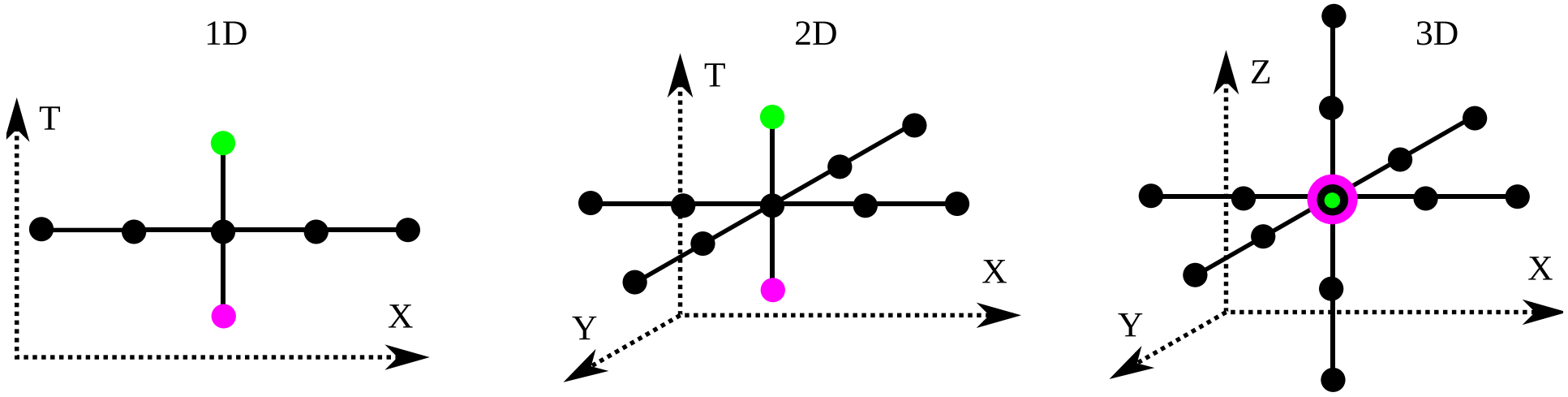
MIPT

Introducing

- Explicit numerical methods are used for a wide range of scientific problems
- Need to speed up stencils calculation
- SIMD-computing
- Tiling (recursive and non-recursive)

Introducing

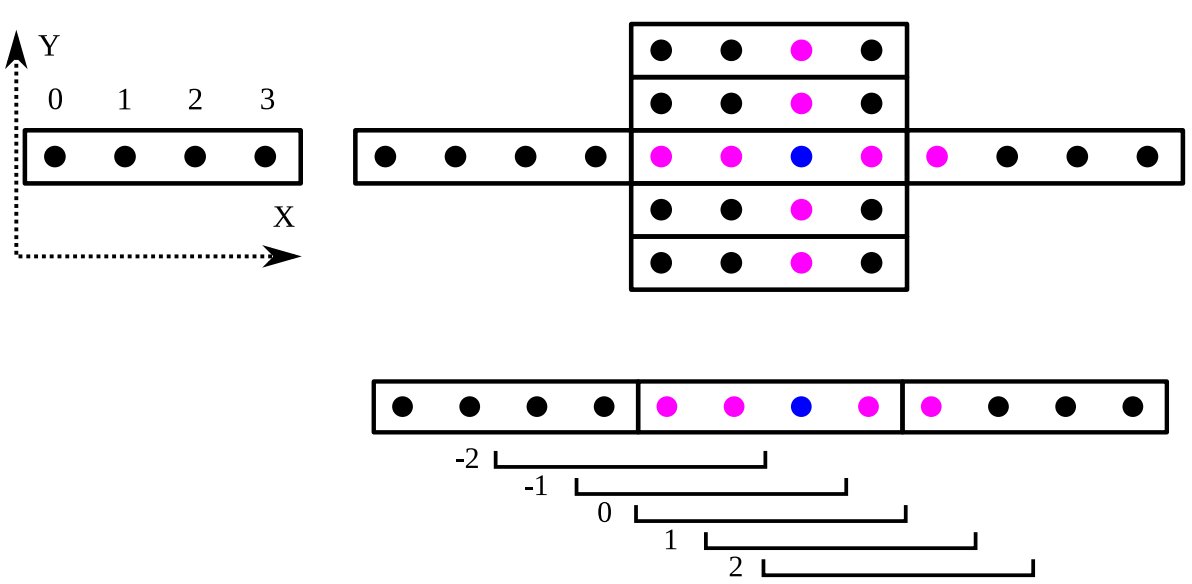
- Explicit numerical method -the solution of the acoustic equation in three dimensional space by the FDTD
- Six-core CPU Intel (R) Xeon (R) E5-2620 v2 (2.1 GHz) 32 KB (L1), 256 KB (L2), 15 MB (L3)
- GCC 8.1.0 with OpenMP
- $N = \frac{N_x * N_y * N_z * N_t * Op}{T_{calc} * Freq_{cpu} * k * M_{kernel}}$ – the percentage of peak performance



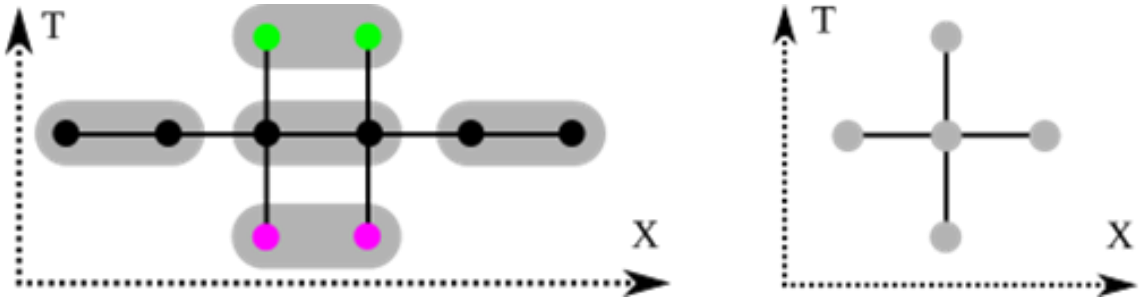
Stencil representation

Vectorization

- Perform the same operation (addition or multiplication) with several data simultaneously
- AVX-instructions (_m256d register) with 4 double values
- Used vectorization for external spatial cycle



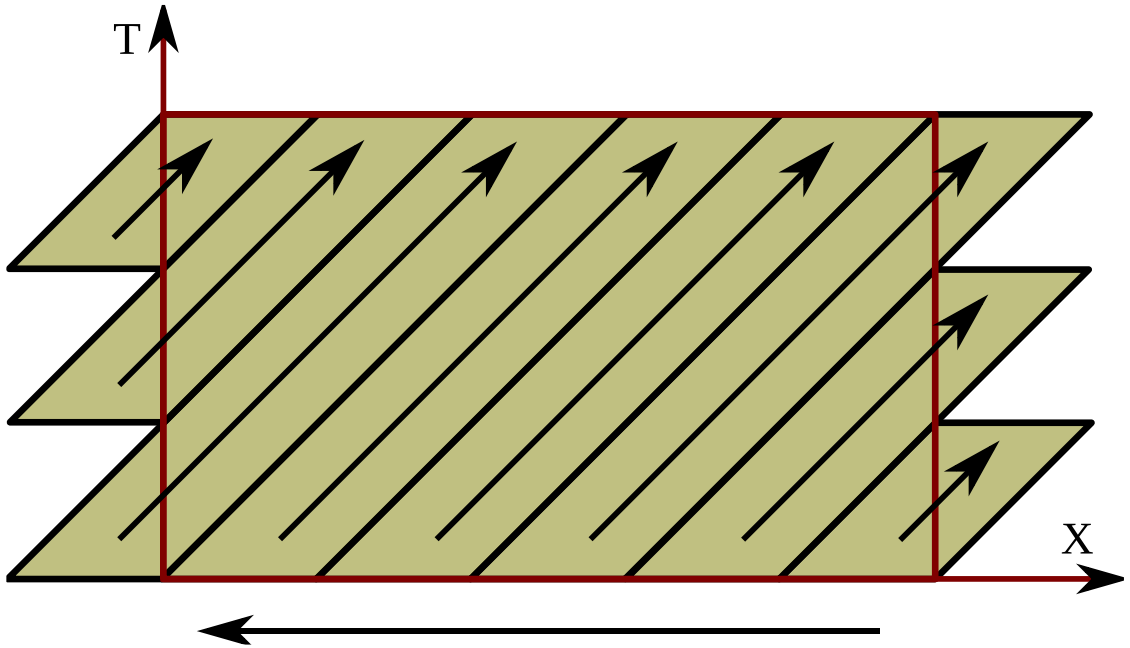
Stencil representation in vectors with shifts



. 3-points stencil

Cube tiling (non-recursive)

- Need to traverse $N_x * N_y * N_z * N_t$
- Tile size fixed

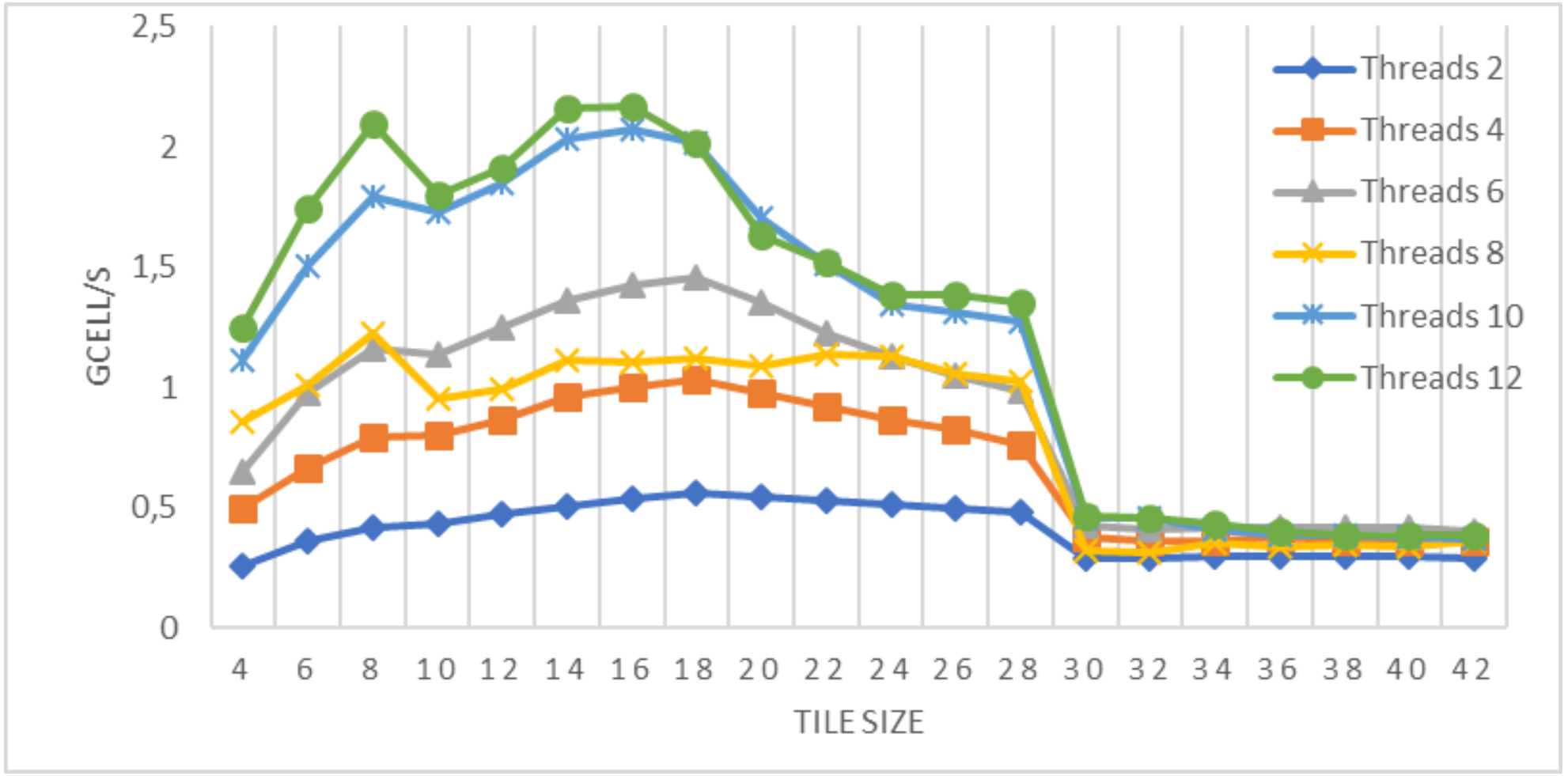


Tile representation in XT

Parallel tile calculation in XY

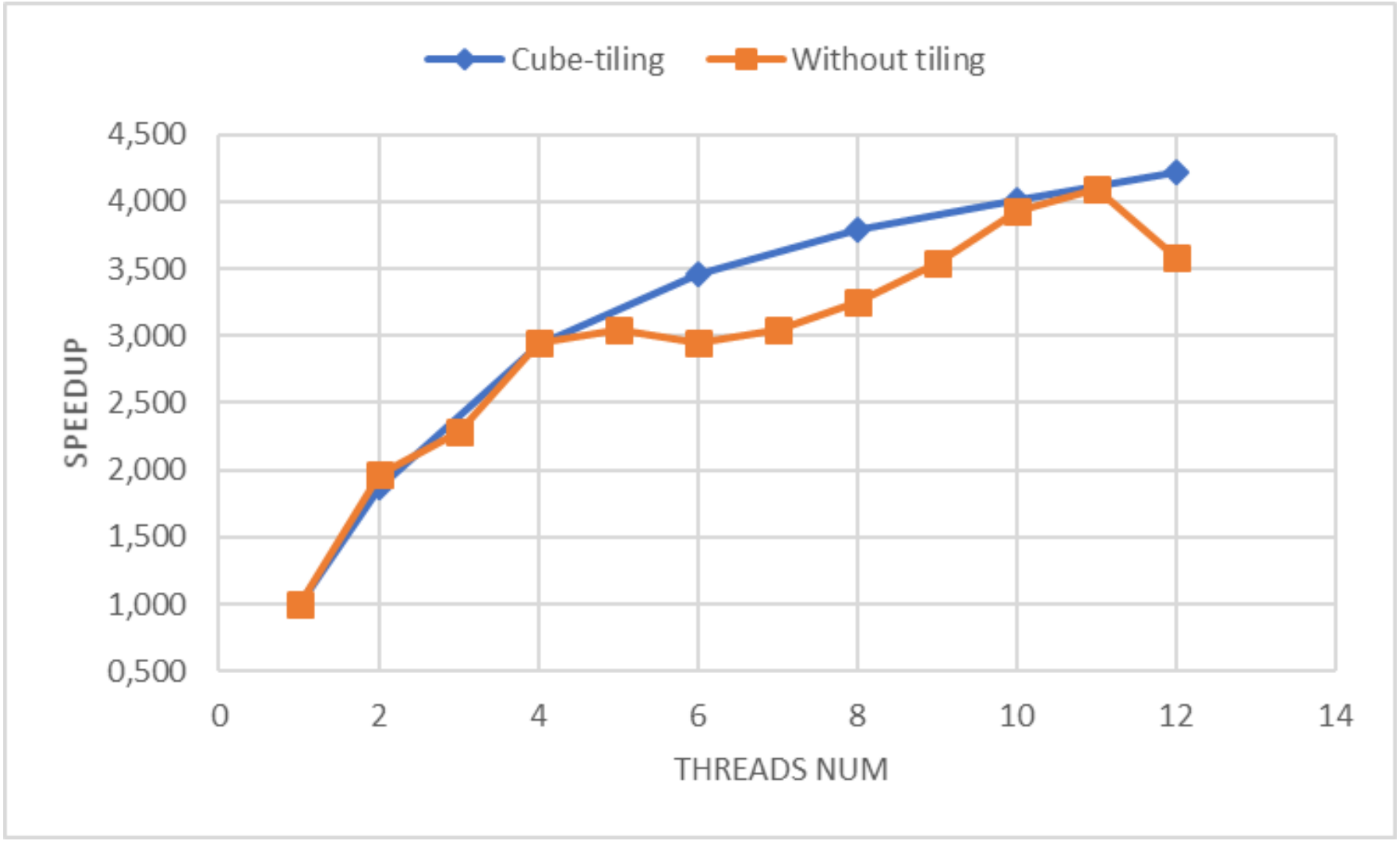
Cube tiling (non-recursive)

- Gcells/s - number of nodes calculated per second



Graph of the number of points calculated per second from a fixed tile size on various threads

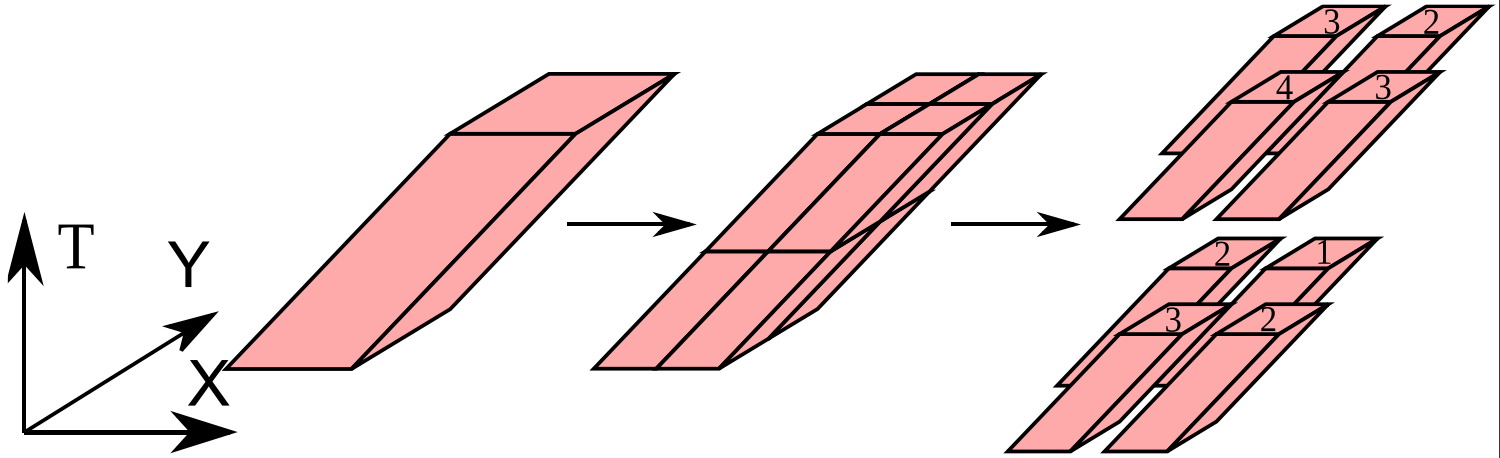
Cube tiling (non-recursive)



Graph of the number of threads on the speed up of calculations for cube-tiling

Cube tiling (recursive)

- $h = 2^{k-1}$ the distance between tiles at the current recursion level



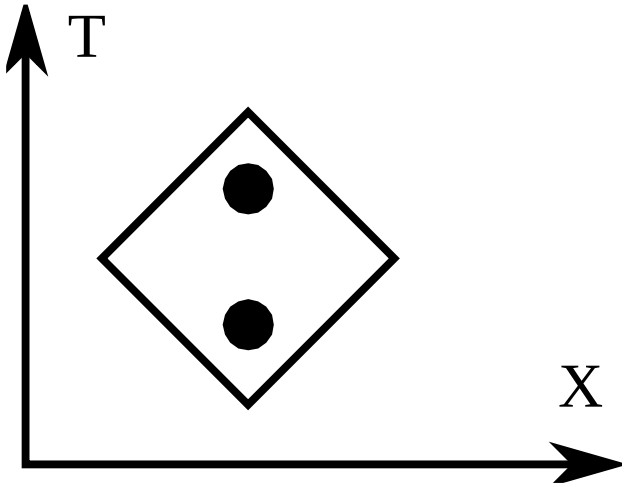
Recursive tile splitting in XYT

Grid Size	GCell/s	Tile size
128^4	**0.284**	64
256^4	0.268	64
1024^4	0.189	64
128^4	0.235	128
256^4	0.263	128
1024^4	0.164	128
256^4	0.263	256
1024^4	0.155	256

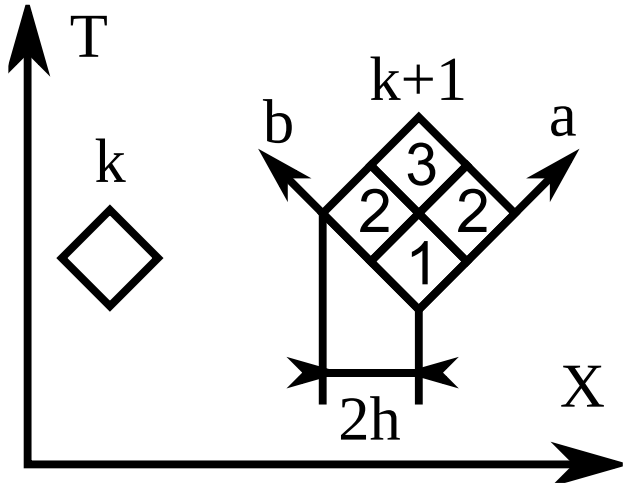
- Best Gcell/s for non-recursive method is 2.07 GCell/s

Diamond-shape tiling (non-recursive)

- Turning cube tile for 45 degree



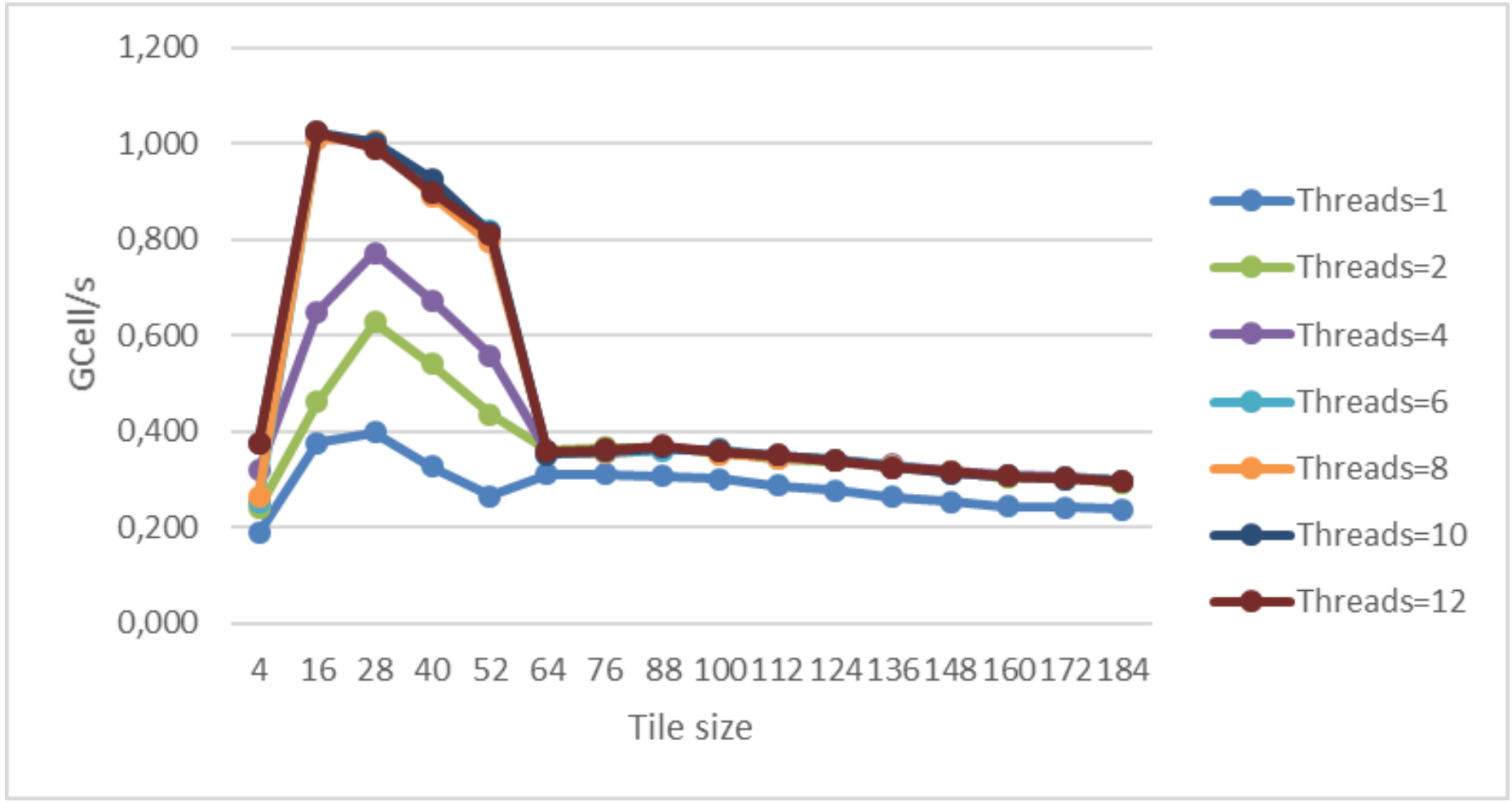
Diamond-shape tile



Tile parallel calculation

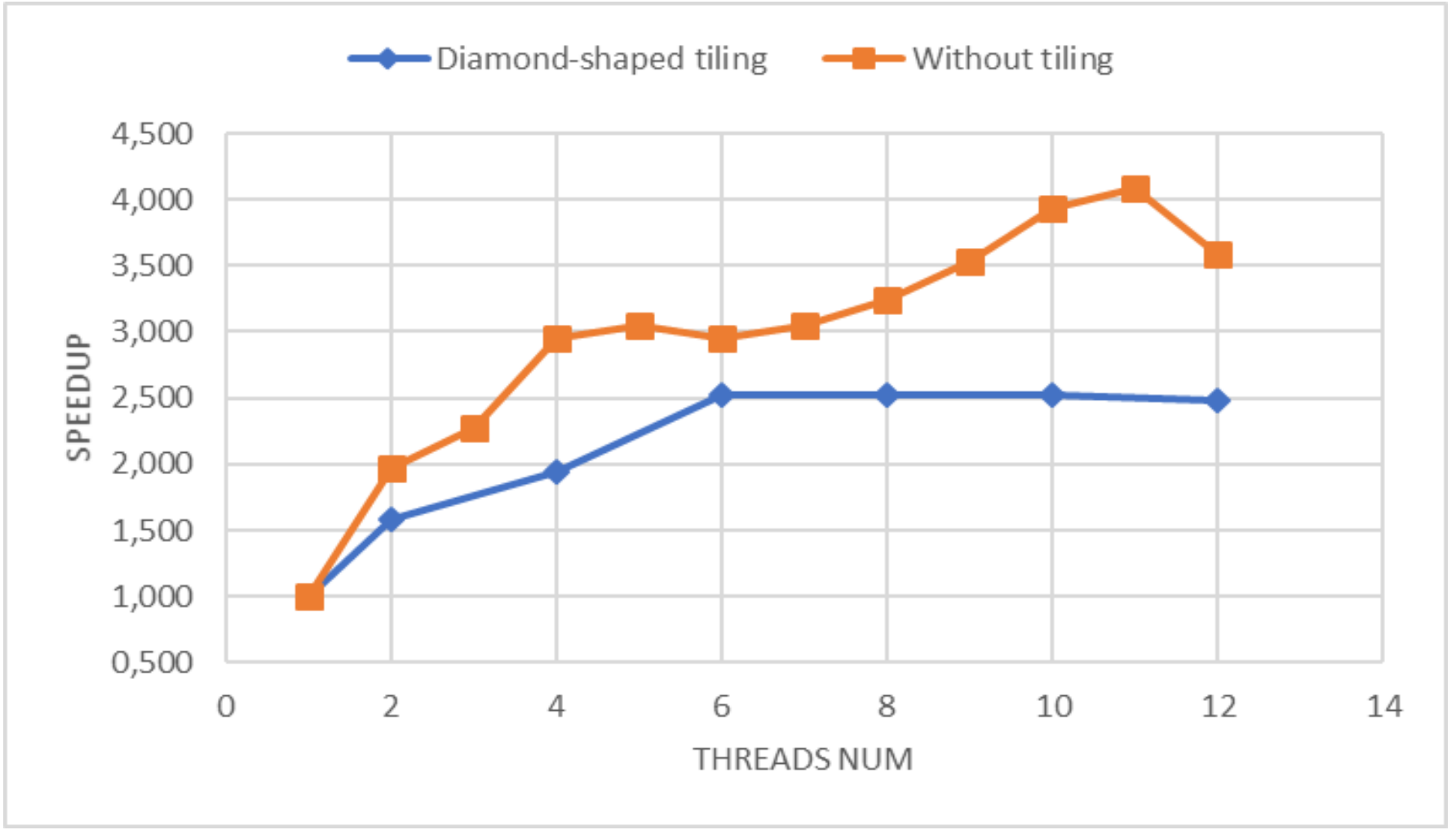
Diamond-shape tiling (non-recursive)

- Cache misses



Graph of the number of points calculated per second from a fixed tile size on various threads

Diamond-shape tiling (non-recursive)



The graph of the number of threads on the speedup of calculation for diamond-shape tiling

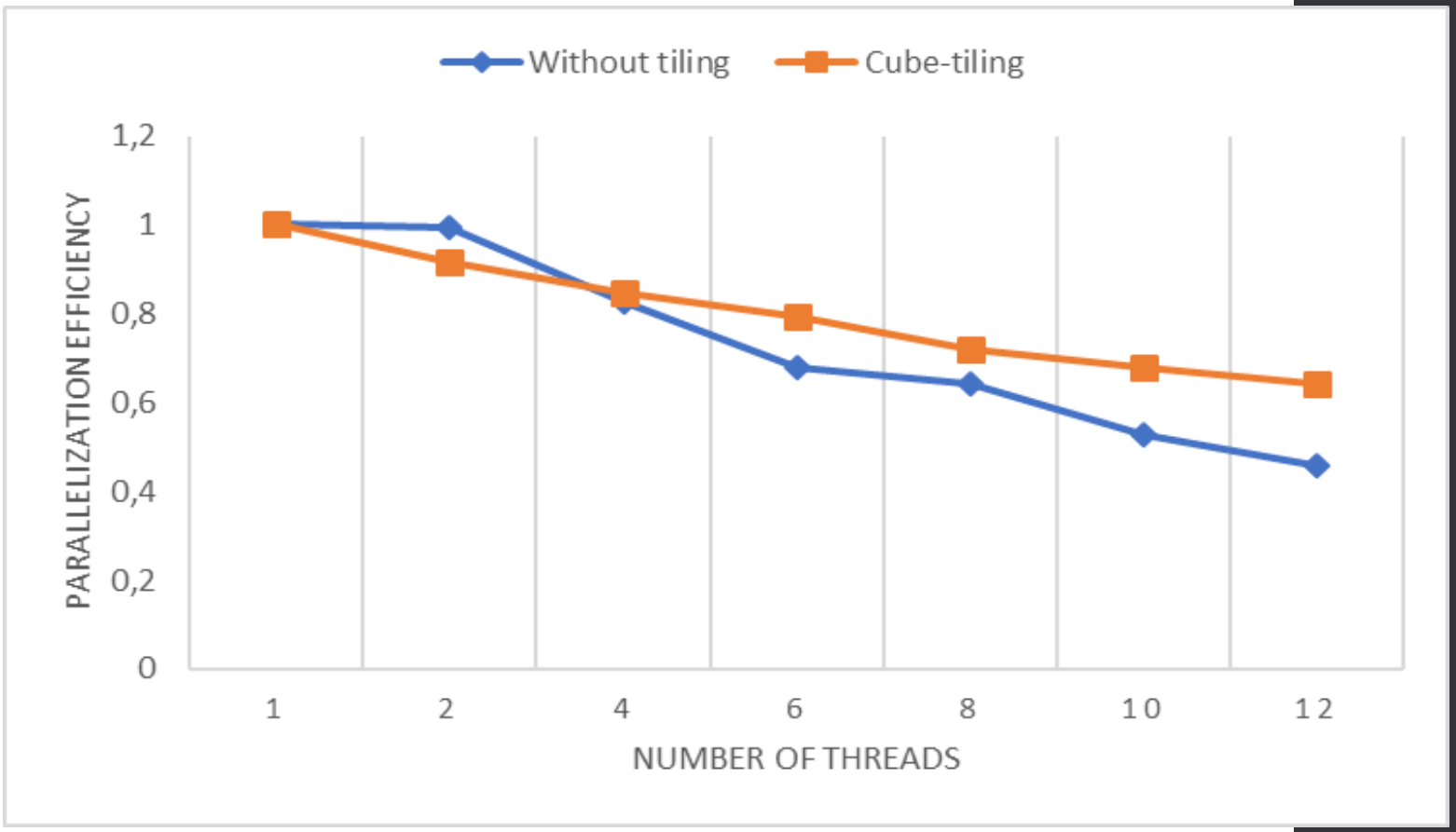
Diamond-shape tiling (recursive)

- Variation of recursion level

Graph of dependence of the number of points calculated per second from the recursion level of the iteration space in X recursive tile, size 128 on various threads

Conclusion

- Non-local vectorization with cube tiling get best performance stencil calculation
- Recursive tiling doesn't improve performance



A plot of the number of threads on the efficiency of paralleling

Version of program	Percentage of peak computing performance
Origin	2.4 %
AutoVec	5.4 %
AlignedVec	5.9 %
CubeTiling	**36.6 %**
DiamTiling	28.7 %

Thanks you for your attention !