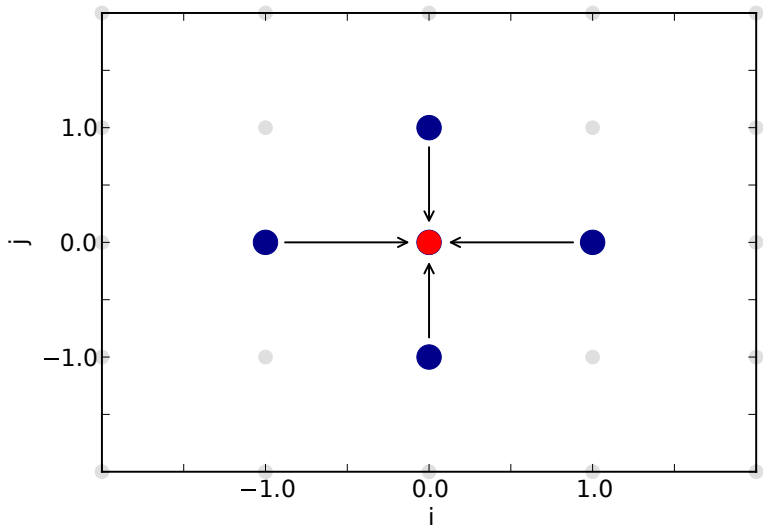
Motivation: Tiling stencil computations for GPUs

Tobias Grosser

Jan 20, 2015, PolyCOMP Tutorial at HiPEAC'15

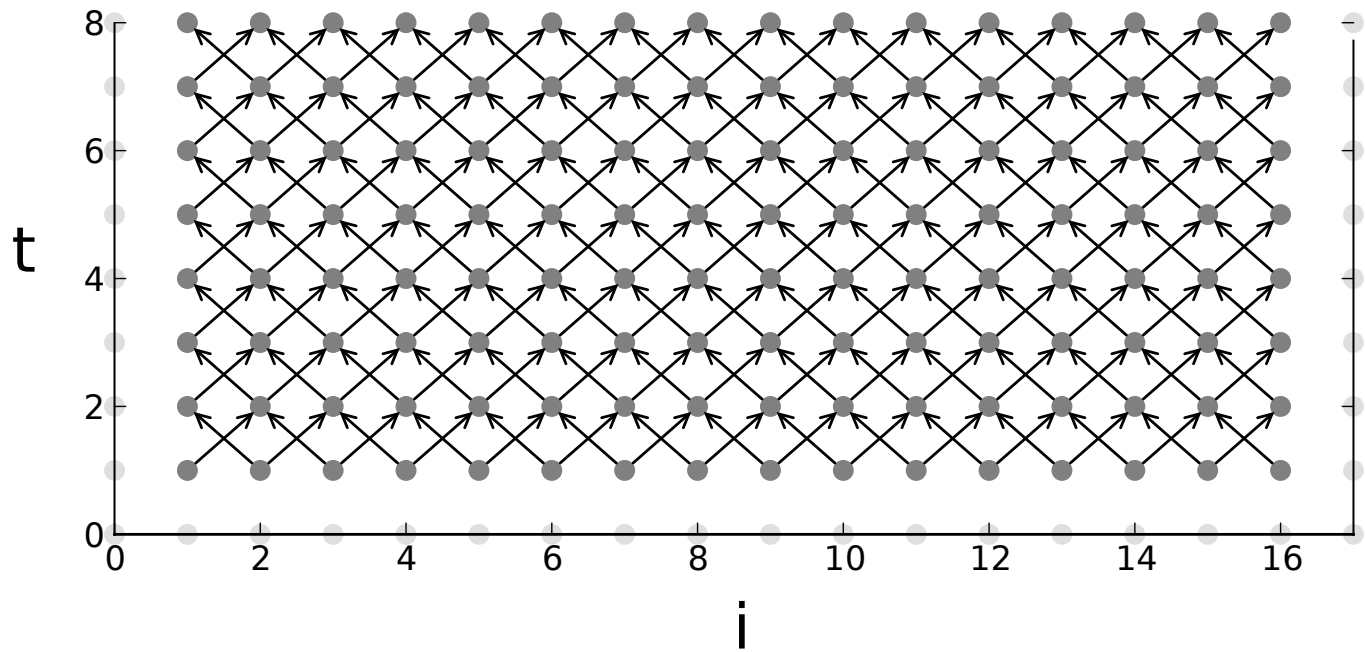
Iterative Stencil Computations



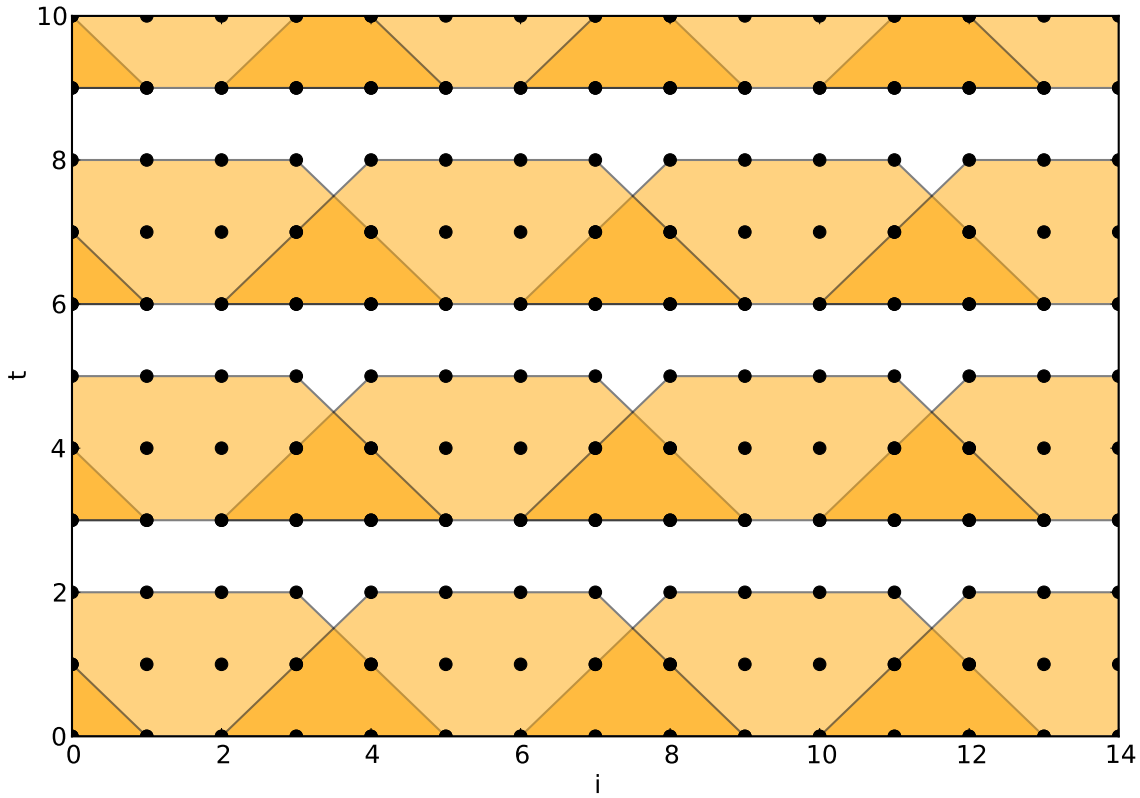

```
for (t=0; t < T; t++)
  for (i=1; i < N - 1; i++) // parallel
    for (j=1; j < N - 1; j++) // parallel
      A[(t+1)%2][i][j] = 0.2 * (A[t%2][i][j]
                        + A[t%2][i][j-1] + A[t%2][i][j+1]
                        + A[t%2][i-1][j] + A[t%2][i+1][j]);
```

Iteration Space - 1D Time / 1D Space

```
for (t=0; t < T; t++)
  for (i=1; i < N - 1; i++) // parallel
      A[(t+1)%2][i] = A[t%2][i-1] + A[t%2][i+1];
```

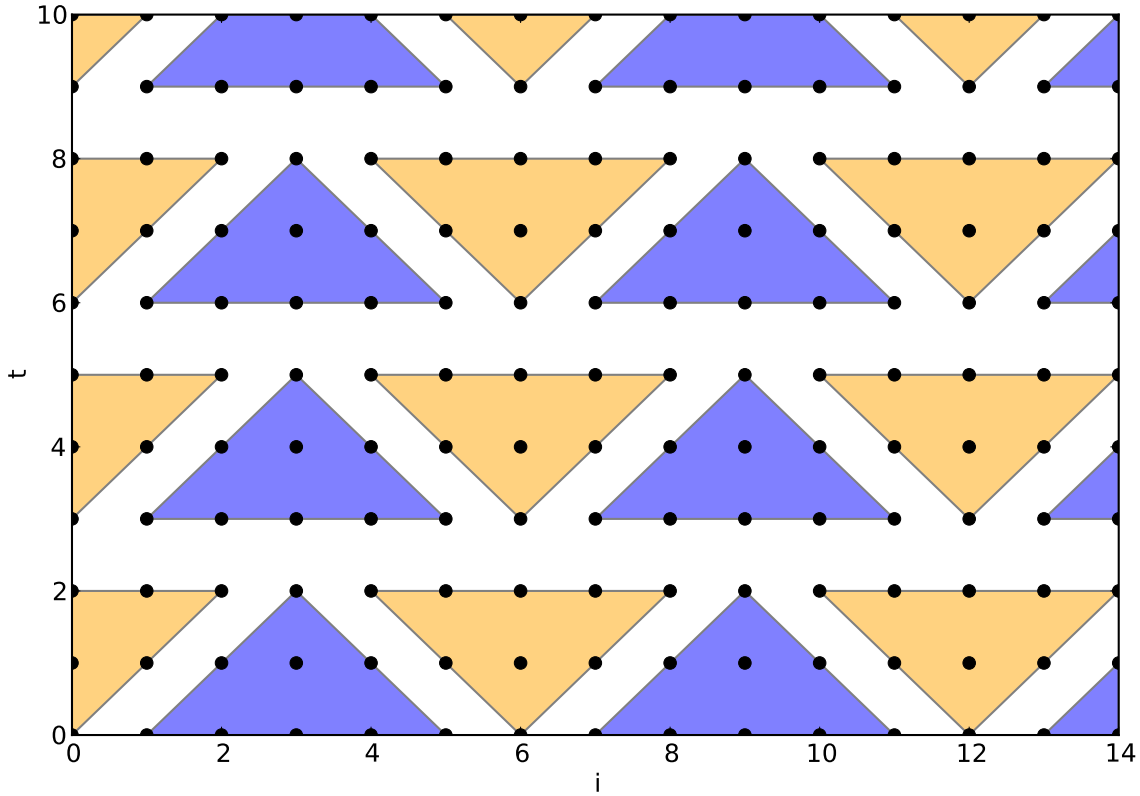


Overlapped Tiling



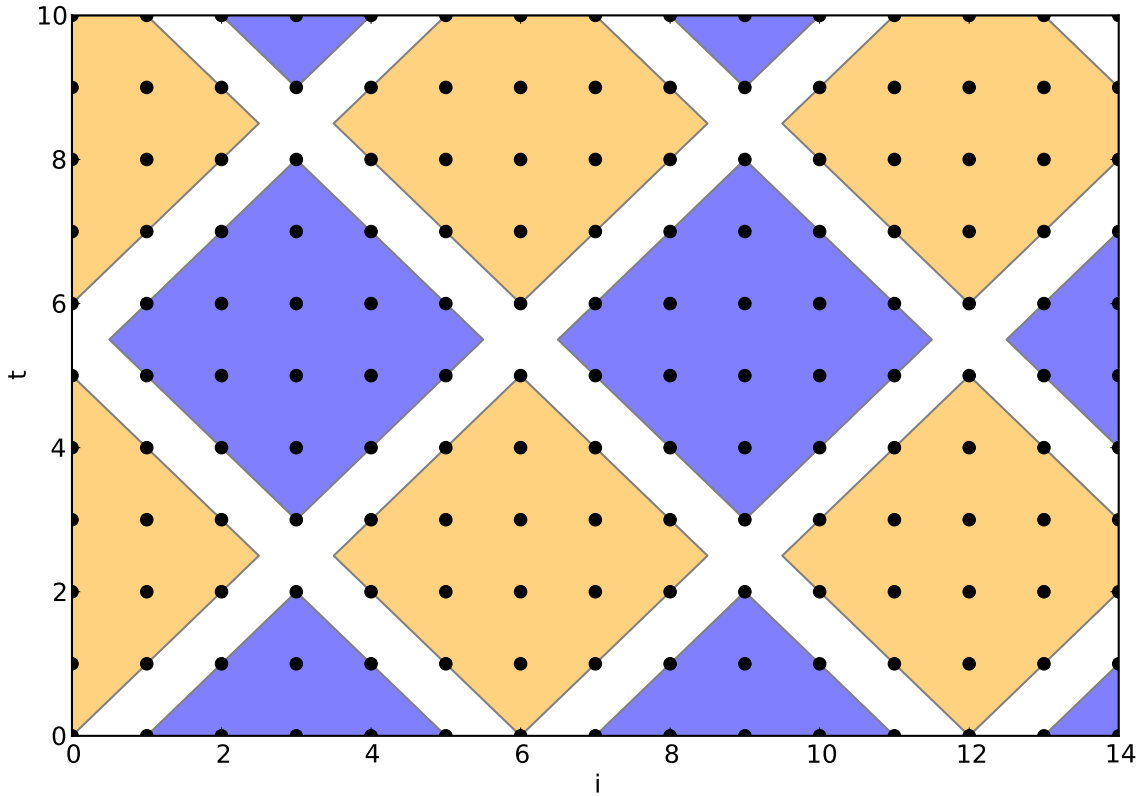
Justin Holewinski, Louis-Noël Pouchet, and P Sadayappan. "High-performance code generation for stencil computations on GPU architectures". In: *International Conference on Supercomputing (ICS)*. 2012

Split tiling



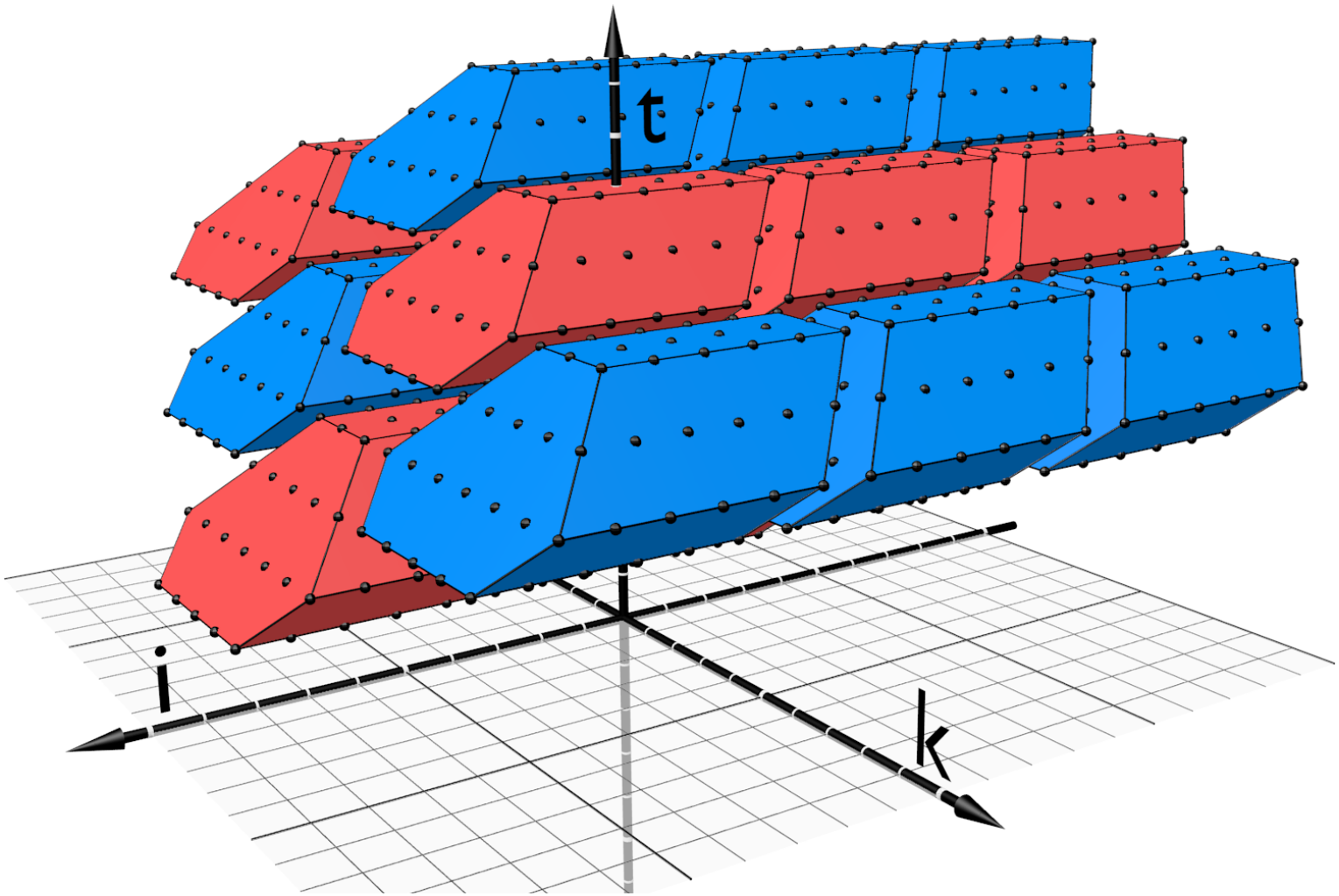
Tobias Grosser et al. "Split tiling for GPUs: automatic parallelization using trapezoidal tiles". In: *Proceedings of the 6th Workshop on General Purpose Processor Using Graphics Processing Units*. ACM. 2013

Diamond tiling



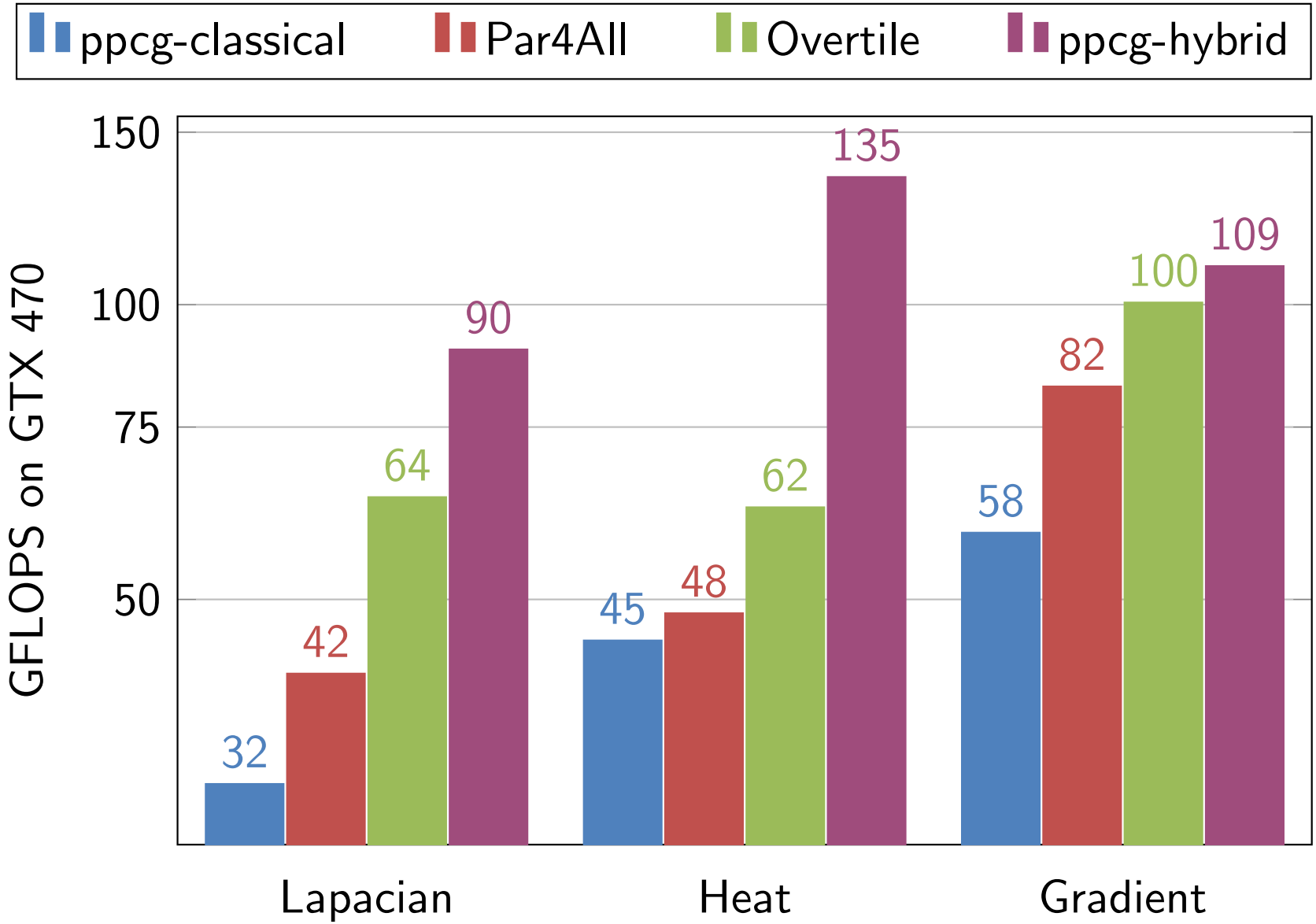
Vinayaka Bandishti, Irshad Pananilath, and Uday Bondhugula. "Tiling stencil computations to maximize parallelism". In: *Supercomputing*. IEEE Computer Society Press. 2012

Hybrid Hexagonal/Parallelogram Tiling (2D Space)

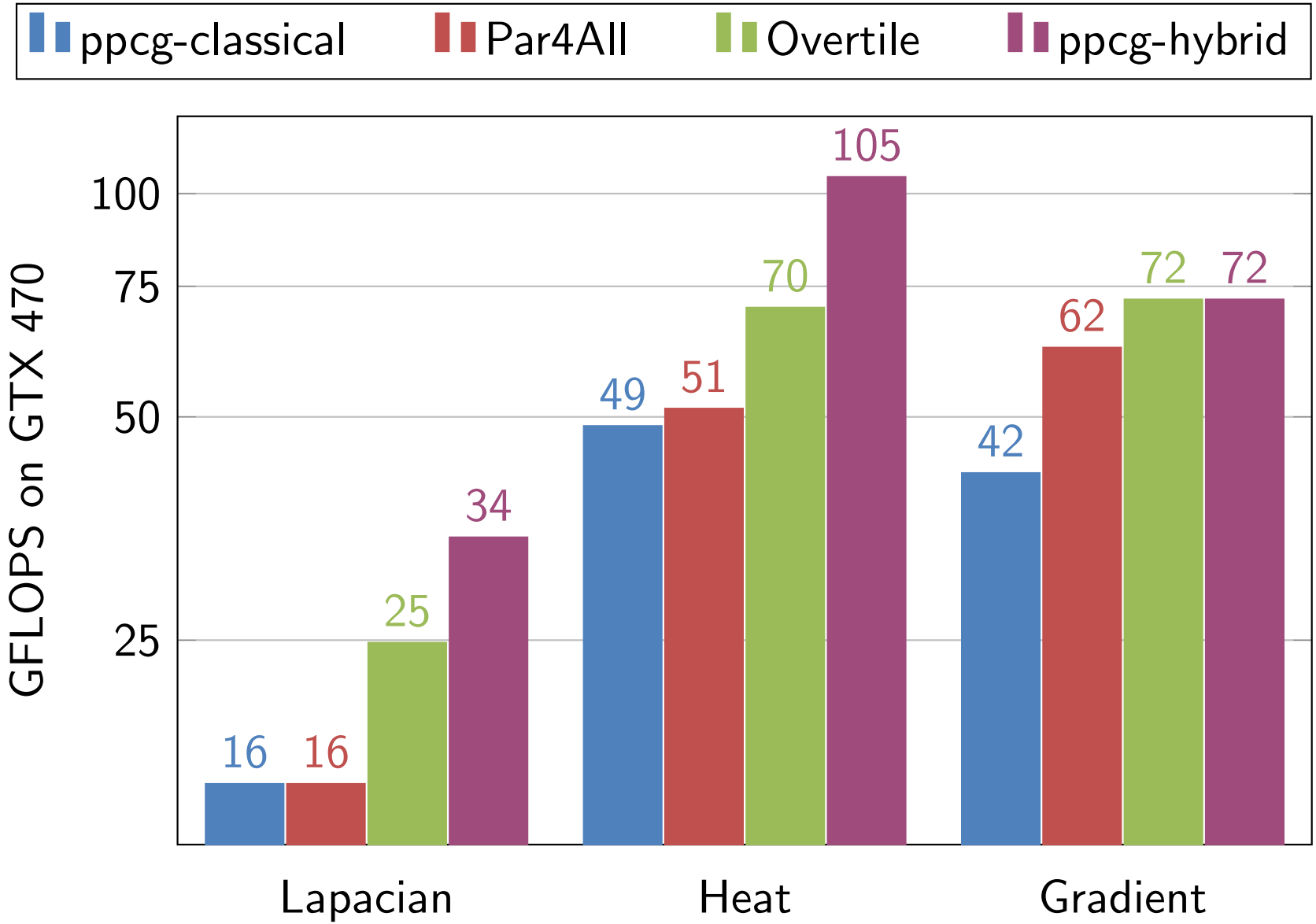


Tobias Grosser et al. "Hybrid Hexagonal/Classical Tiling for GPUs". In: *International Symposium on Code Generation and Optimization (CGO)*. Orlando, FL, United States, 2014

Experimental Results 2D - 3072^2



Experimental Results 3D - 384^3



Overall

- Many different tiling schemes
- Finding a good scheme is a research question
- GPU optimization involves:
  - Dependence analysis
  - Data-usage and memory footprint analysis
  - Generation of copy-in and copy-out code
  - Generation of compute code
- Reason about shapes and points, not source code