Cache-Blocking Tiling of Large Stencil Codes at Runtime

István Z Reguly, Gihan R. Mudalige, Mike Giles
[1](reguly.istvan@itk.ppke.hu) PPCU ITK Hungary
[2](g.mudalige@warwick.ac.uk) University of Warwick, UK
[3](mike.giles@maths.ox.ac.uk) University of Oxford, UK

Introduction

Lots of research on compiler approaches:
- polyhedral compilers, such as Pluto [1], Polymage, Halide
- stencil compilers, such as Pochoir [2]
- they work well, when a few loops repeat in a time iteration

What they do not do:
- multiple compilation units
- different code paths

CloverLeaf 3D hydrodynamics mini-app:
- 6000 lines of code, part of Mantevo suite [3]
- 141 different loopnests, 600 long chain per time iteration
- 30 values per gridpoint, 46 stencils
- spread across 15 source files
- versus tests with 1-5 loops in 1 compilation unit

+ CloverLeaf 2D, TeaLeaf 2D, OpenSBLI
- similar specs, same challenges

OPS Library

Domain Specific Language for structured meshes [4]:
- define blocks & datasets on them, handing all data to the library: memory movement & locality managed
- computations expressed as parallel sweeps over a block, accessing data with pre-defined stencils

```
//user kernels
void copy(double *d2, const double *d1) {
  d2[OPS_ACC0(0)] = d1[OPS_ACC1(0)];
}
void calc(double *d1, const double *d2) {
  d1[OPS_ACC0(0)] = d2[OPS_ACC1(0)] +
                    d2[OPS_ACC1(1)] +
                    d2[OPS_ACC1(-1)];
}
```

"User kernel": computaiton to be applied to each gridpoint

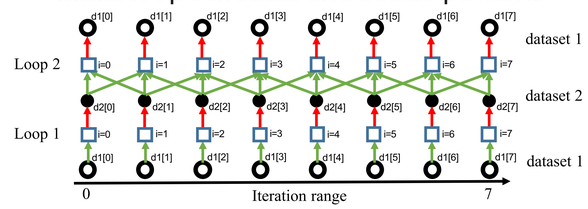
Loop description: kernel, range, dataset, stencil, R/W

```
int range[4] = {0,8};
ops_par_loop(copy, block, 1, range,
      ops_arg_dat(d2,S2D_0,"double",OPS_WRITE),
      ops_arg_dat(d1,S2D_0,"double",OPS_READ));
ops_par_loop(calc, block, 1, range,
      ops_arg_dat(d1,S2D_0,"double",OPS_WRITE),
      ops_arg_dat(d2,S2D_1,"double",OPS_READ));
```

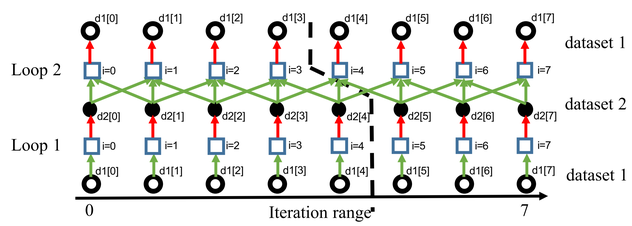
Analysis and execution

Given the ops_par_loop construct, we don't need to execute immediately; queue up computational loops until something has to be returned to the user:
- sequence of loops determined at runtime, oblivious to compilation units and control structures
- we have all information required for dependency analysis and for transforming loop schedules
- simple skewed tiling, intra-tile parallelism with OpenMP

Initial loop schedule and access patterns



Split Loop 2 into tiles so each fits in cache, then move to loop 1 and set loop bounds so data dependencies are satisfied. Repeat. Execute tile 1 of all loops, then tile 2, etc.



Single socket performance

Comparison to Pluto & Pochoir on 2D Heat equation:
v1: Pluto 4.6x, OPS 3.1x
v2: Pochoir 3.2x, OPS 2.3x

Speedup on other codes:
CloverLeaf 3D: 1.96x
TeaLeaf: 3.56x
OpenSBLI: 1.71x

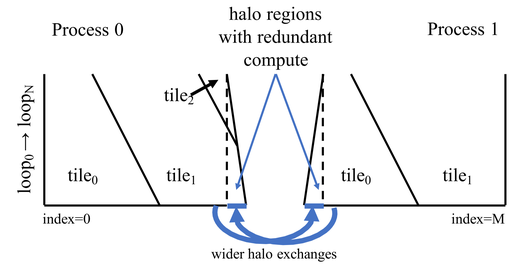
CloverLeaf 2D: Machine: E5-2650 v3

Phase	Timestep	Ideal Gas	Viscosity	PdV	Revert	Acceleration	Fluxes
Baseline Time	0.71	0.89	0.89	2.36	0.37	0.92	0.62
Tiled Time	0.69	0.65	0.36	1.54	0.08	0.4	0.34
Tiled GB/s	40.9	41.27	38.85	47.36	143.86	77.1	65.37
Tiled Speedup	1.03x	1.37x	2.45x	1.53x	4.7x	2.28x	1.8x

Phase	Cell Advection	Momentum Advection	Reset	Update Halo	Field Summary	The Rest	Total
Baseline Time	4.01	6.68	0.74	0.07	0.11	0.31	18.68
Tiled Time	2.23	1.95	0.25	0.26	0.05	0.52	8.73
Tiled GB/s	54.34	112.66	91.51	0.73	96.42	3.8	66.12
Tiled Speedup	1.8x	3.43x	3x	0.28x	2.02x	0.61x	2.14x

Extension to MPI

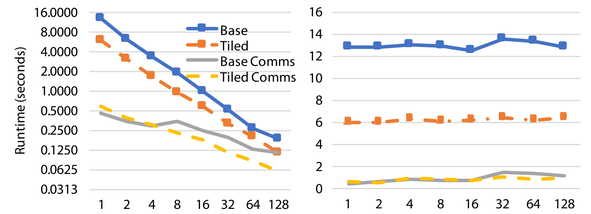
OPS manages partitioning and all halo exchanges, based on the loop description.
- extend tiles to work on halos: overlapped tiling scheme
- one large halo exchange in the beginning, none after



Different skew along boundary: disappearing tiles

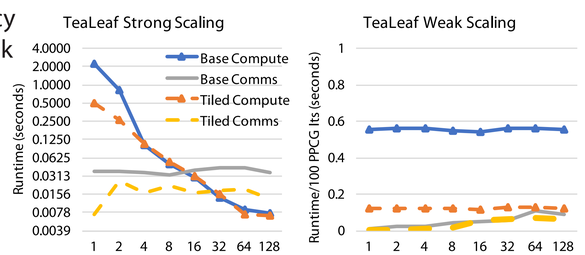
Scaling performance

Singificantly improved MPI communication times: fewer, larger messages. Scaling to 128 nodes on Cineca's Marconi:



2x E5-2697 v4 per node
Omni-Path
100 Gbps

Scaling Efficiency Strong/Weak
CloverLeaf 2D 95%/99%
CloverLeaf 3D 88%/99%
OpenSBLI 92%/87%



Acknowledgements & References

We acknowledge PRACE for awarding us access to resource Marconi based in Italy at Cineca. This work was supported by the János Bólyai Research Scholarship of the Hungarian Academy of Sciences, as well as UK EPSRC projects EP/K038494/1, EP/K038486/1, EP/K038451/1 and EP/K038567/1 on Future-proof massively-parallel execution of multi-block applications" and EP/J010553/1 Software for Emerging Architectures" (ASEArch) project.

[1] U. Bondhugula, M. Baskaran, S. Krishnamoorthy, J. Ramanujam, A. Rountev, and P. Sadayappan, "Automatic transformations for communication-minimized parallelization and locality optimization in the polyhedral model," in International Conference on Compiler Construction (ETAPS CC), 2008.
[2] Y. Tang, R. A. Chowdhury, B. C. Kuszmaul, C.-K. Luk, and C. E. Leiserson, "The pochoir stencil compiler," in Proceedings of the Twenty-third Annual ACM Symposium on Parallelism in Algorithms and Architectures, 2011
[3] Heroux, Michael A., et al. "Improving performance via mini-applications." Sandia National Laboratories, Tech. Rep. SAND2009-5574 3 (2009).
[4] I. Z. Reguly, G. R. Mudalige, M. B. Giles, D. Curran, and S. McIntosh-Smith, "The OPS Domain Specific Abstraction for Multi-block Structured Grid Computations," in Proceedings of the Fourth International Workshop on Domain-Specific Languages and High-Level Frameworks for High Performance Computing 2014, pp. 58–67.