PDPTA'01, Las Vegas, June 25–28 2001

DiMEPACK — A Cache–Optimized Multigrid Library

Markus Kowarschik

Universität Erlangen–Nürnberg, Germany

presently: Lawrence Livermore National Laboratory

Christian Weiss

Technische Universität München, Germany

Our project is presently funded by the *Deutsche Forschungsgemeinschaft* (DFG)

Outline

1. Motivation
2. Cache optimization techniques for structured grid computations
   (a) Data layout optimizations
   (b) Data access optimizations
3. Features of the *DiMEPACK* library
4. Performance results
5. Conclusions and future work

Motivation

Our work is motivated by two independent observations:

1. Computer architecture:
   - Growing gap between CPU execution speed and memory performance
   - Introduction of cache memories to mitigate the *memory wall* effect
2. Numerical analysis:
   Multigrid methods are among the fastest algorithms for the solution of large systems of equations

Idea: Develop *locality–optimized* multigrid codes: *DiMEPACK*
Data–local iterative methods for the efficient solution of PDEs

Data layout optimizations

Architecture of cache memories:

- Cache memories are split into *cache lines*
- Cache types:
  - n–way set–associative
  - $n = 1$: direct mapped
  - $n = \max$: fully associative

Cache thrashing:

- Necessary data mutually throw each other out of the cache
- Cache thrashing is very likely for stencil–based computations

Array padding:

Allocate arrays larger than necessary — i.e., change the relative distances in memory — in order to avoid/mitigate cache thrashing

Data layout optimizations (2)

Cache–aware data structures for computations on structured grids:

Idea: Merge data which are needed together to increase *spatial locality:* cache lines contain several data items

Example: Gauss–Seidel, 2D, 5–point stencils:

$$u_i^{(k+1)} = a_{i,i}^{-1} \left(f_i - \sum_{j<i} a_{i,j} u_j^{(k+1)} - \sum_{j>i} a_{i,j} u_j^{(k)} \right)$$

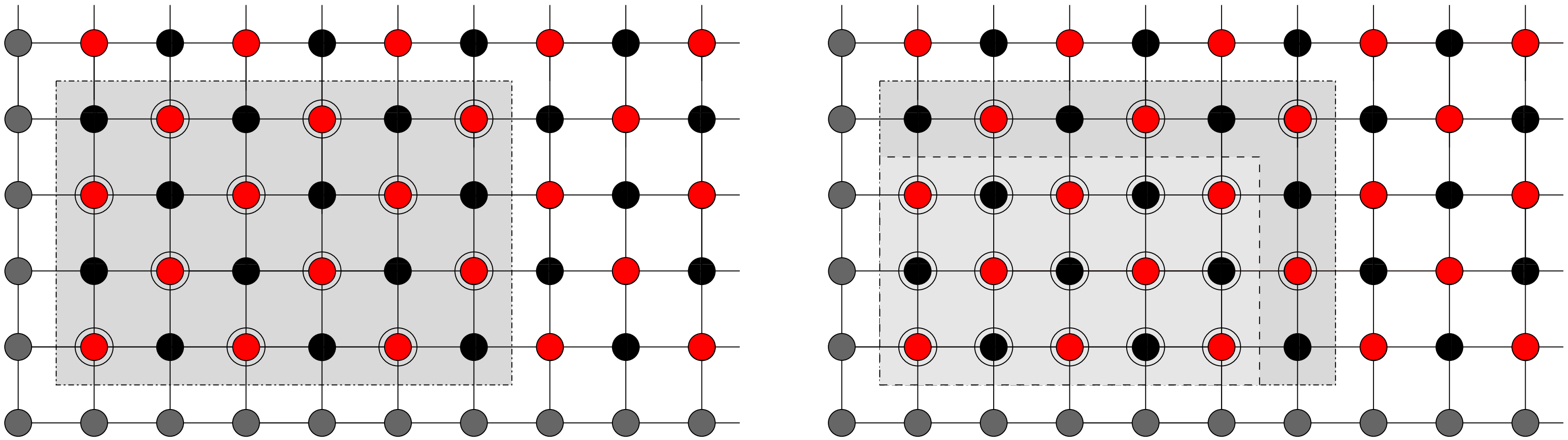
```
typedef struct {
  double f;
  double cNorth, cEast, cSouth, cWest;
} eqnData;

double  u[N][N];             // Solution vector
eqnData rhsAndCoeff[N][N]; // Right-hand side and coefficients
```

Survey: data access optimizations

Our access optimization techniques have been presented previously, e.g. *ACM/IEEE Supercomputing Conference 1999*:

- *Loop fusion:* one sweep per smoothing iteration
- *1D loop blocking (tiling):* several iterations per sweep
- *2D loop blocking*, e.g. *square blocking*:



The DiMEPACK library

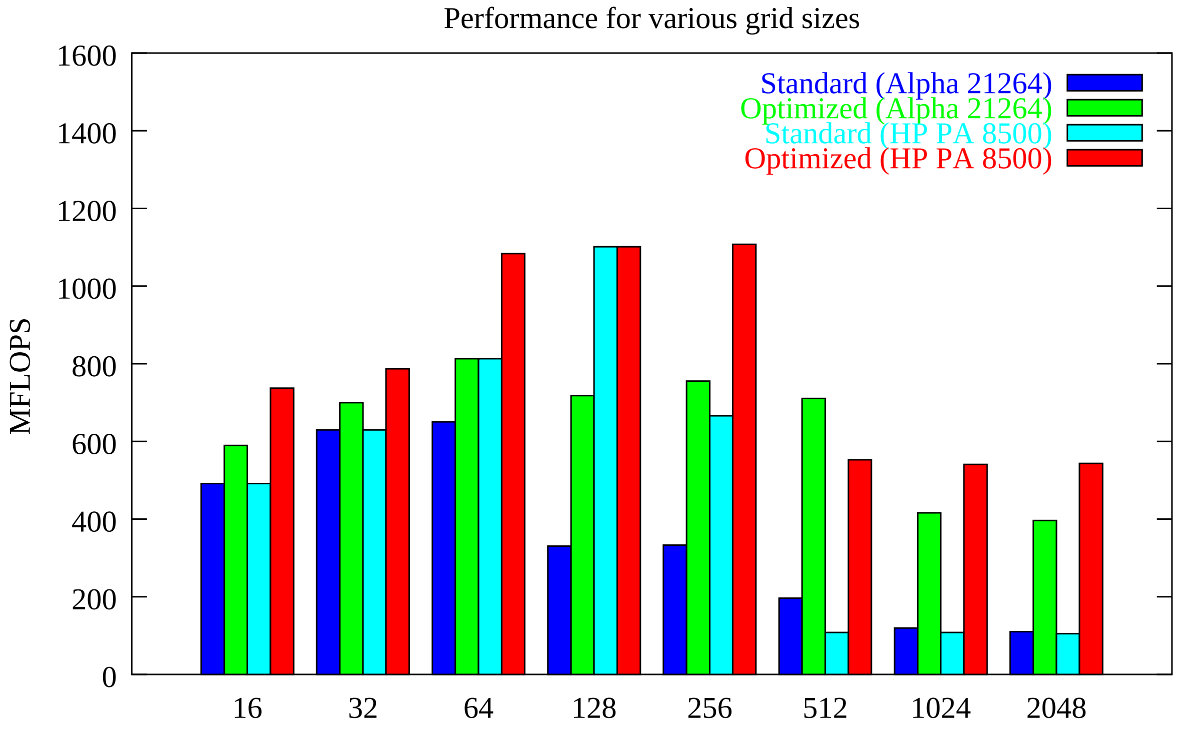
- Multigrid correction scheme: V–cycles, full multigrid
- Constant 5–/9–point stencils
- Boundary conditions of Dirichlet/Neumann type
- Restriction operators: half weighting, full weighting
- Prolongation operator: linear interpolation
- C++ interface, fast Fortran 77 subroutines
- Optimized problem–specific r/b Gauss–Seidel/SOR smoothers (generally, the most time–consuming MG component)
- Cache–optimized intergrid transfer operations (save passes)
- Direct solution of the problems on the coarsest grid (LAPACK)
- Single precision/double precision floating–point arithmetic
- etc.

See our paper and the *DiMEPACK* technical report for details

Performance results for GS smoother

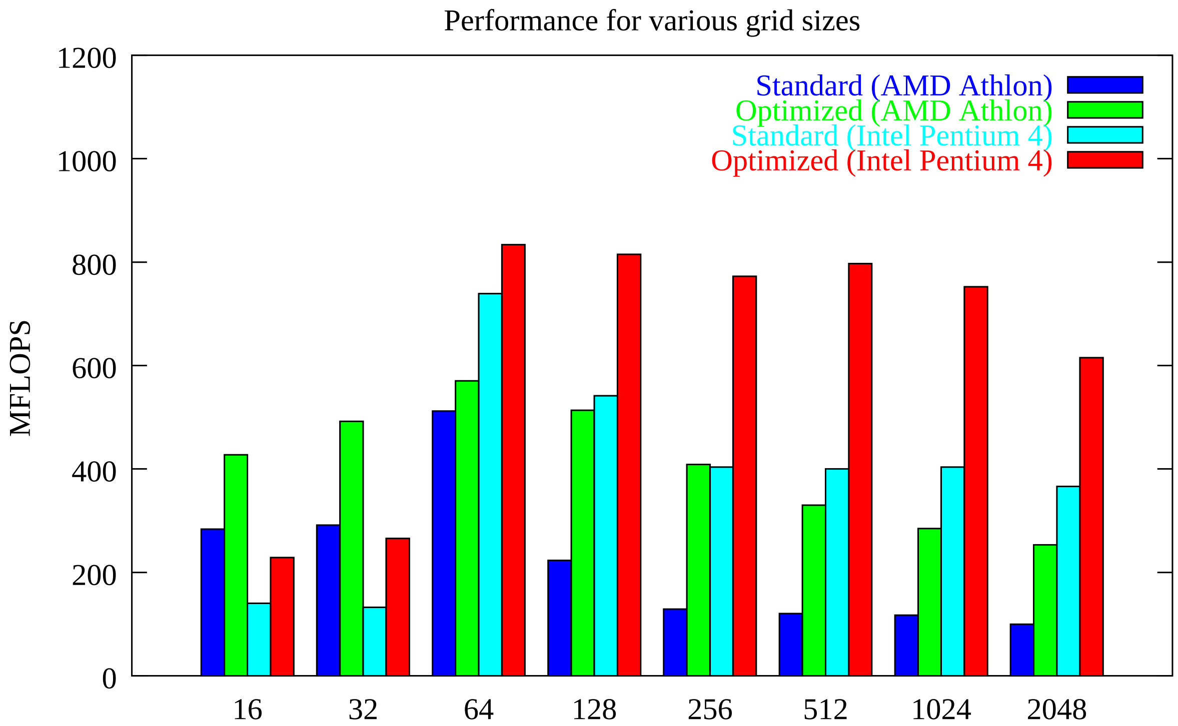
DiMEPACK smoothers; i.e., constant coefficients

Significant improvements, esp. for out–of–cache problem sizes



Performance results for GS smoother (2)

Different machines:



Observation: Lower MFLOPS rates for variable coefficient codes, speedup factors of 2–3 are still possible

Conclusions and future work

- Cache optimization techniques can yield significant speedups for iterative numerical methods
- We have implemented the cache–based MG library *DiMEPACK*
- Practical application of *DiMEPACK* : chip layout optimization, LEA, TUM
- Inherently data–local algorithms and data structures

  This includes:
  - Patch–based grid structures, patch–adaptivity
  - (Residual–driven) adaptive smoothing algorithms
  - Fully adaptive multigrid methods (*FAMe*, U. Rüde)
- Collaboration with D. Quinlan et al., LLNL: *ROSE*, integration of cache optimization techniques into optimizing preprocessors
- Contact `cache@in.tum.de`