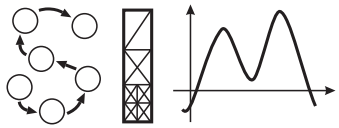
System Simulation Group
Friedrich–Alexander–University
Erlangen–Nuremberg



GAMM 2001 Annual Meeting, ETH Zürich

Cache–friendly multigrid codes

Markus Kowarschik, Ulrich Rüde

Universität Erlangen–Nürnberg

Christian Weiß, Wolfgang Karl

Technische Universität München

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

Outline

1. Introduction
2. Data locality optimizations for stencil–based operations
   (a) Data layout optimizations + performance results
   (b) Data access optimizations + performance results
3. The *DiMEPACK* library
4. Conclusions and future work

Introduction

Performance results for a DEC PWS 500au (Alpha 21164 CPU), theoretically available peak performance: 1000 MFLOPS

We consider the method of Gauss–Seidel:

- Structured grid, 5–point stencil, $\approx$ 64.000 unknowns, $\approx$ 3.5 MB ($<$ size of L3 $=$ 4 MB): 45 MFLOPS
- Unstructured grid, compressed row storage scheme, $\approx$ 64.000 unknowns: 16 MFLOPS

Profiling analysis of the structured grid code *(DCPI)*:
$>$ 50% of the execution time are spent for moving data!

$\Longrightarrow$ Need for cache–aware codes
$\Longrightarrow$ Our project: *DiME* — **D**ata–local **i**terative **ME**thods

Profiling multigrid codes

5–point stencil, MG V–cycles, CGC, finest grid: 1025×1025 nodes, full–weighting, linear interpolation, red–black GS smoother:

% of CPU time for V(4,4) cycles	% of CPU time for V(2,2) cycles	function name
82.63	70.12	smooth
7.83	13.36	restriction
7.32	12.73	prolong
2.23	3.80	init_grid

$\Longrightarrow$ Optimize the smoother first!

Data locality optimizations:
Data layout optimizations + Data access optimizations

Data layout optimizations: array padding

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

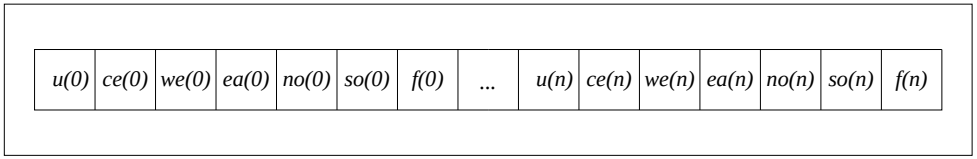
Array padding:

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

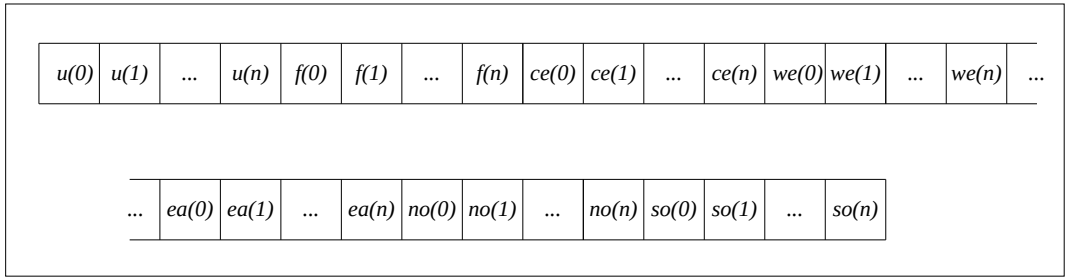
Cache–aware data storage schemes

Many possibilities, example: scalar equation, 5–point stencil:

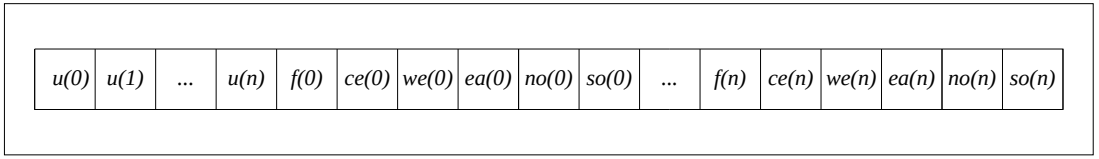
Equation–oriented storage scheme:



Band–wise storage scheme:



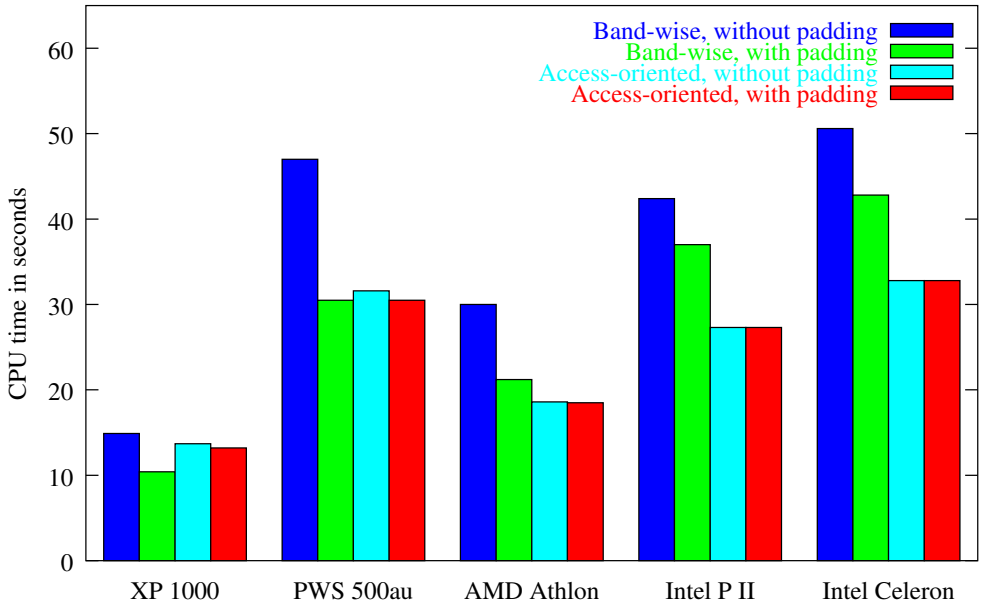
Access–oriented storage scheme:



Execution times for multigrid codes

- Equation–oriented data storage scheme yields poor performance
- CPU times for the multigrid codes based on the remaining two data layouts with and without suitable array paddings:

ca. 25 MFLOPS →

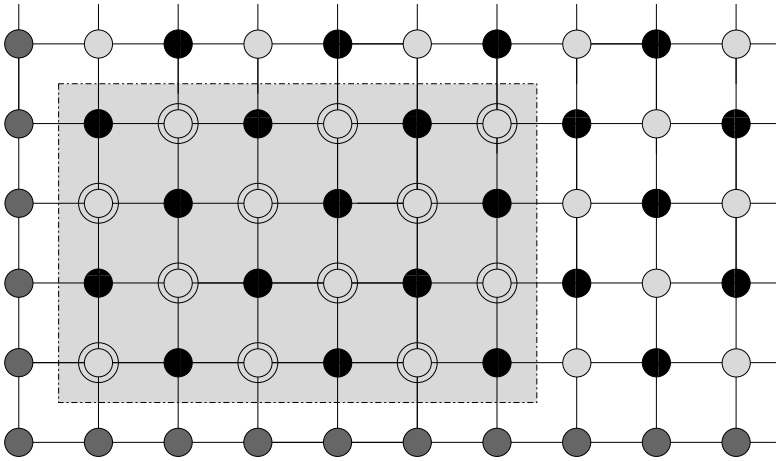


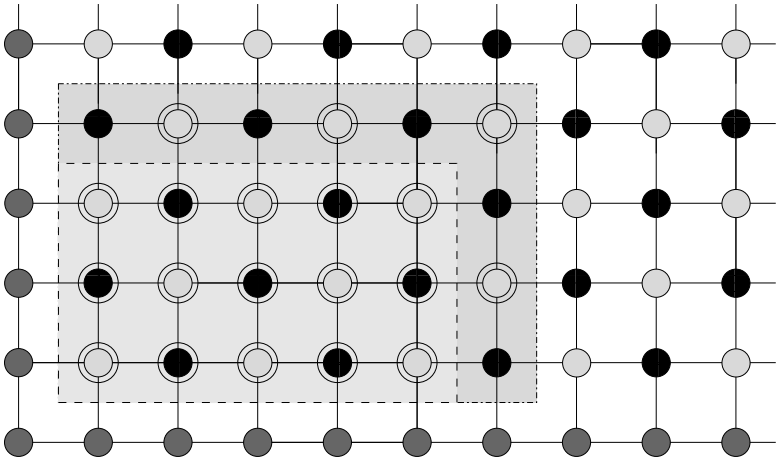
⇒ The band–wise storage scheme is very sensitive, the access–oriented storage scheme is rather insensitive to array padding

Survey: data access optimizations

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

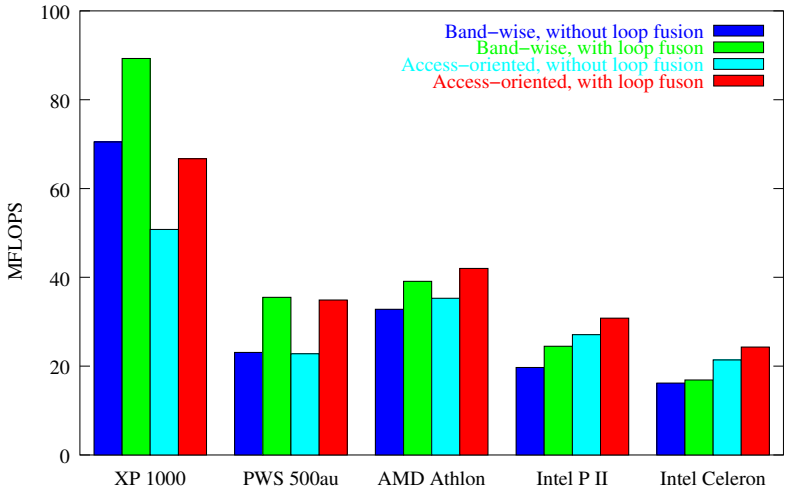
- *Loop fusion:* one sweep per smoothing iteration
- *1D loop blocking (tiling):* several iterations per sweep
- *2D loop blocking*, e.g. *square blocking*:

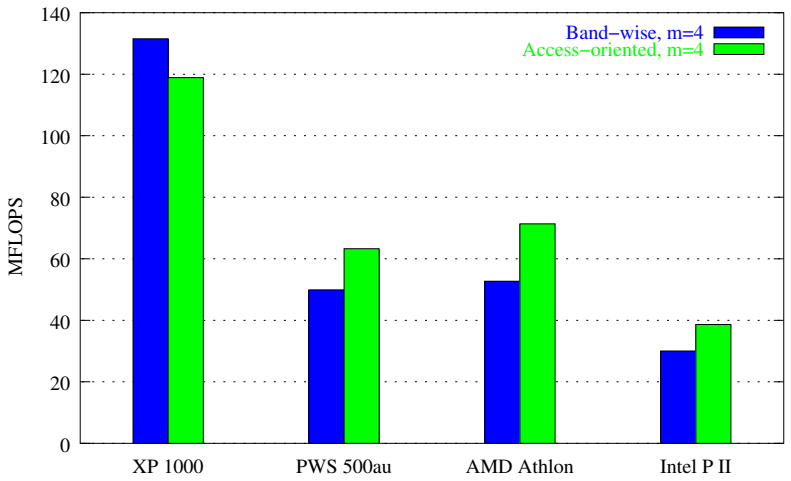




Performance results (2)

- MFLOPS for r/b GS (suitable paddings are used):





- Summary of cache misses and CPU times, 40 GS iterations, 1025×1025 grid nodes ($\approx$ 56 MB), A21164 DEC machine:

	Standard	Loop fusion	1D blocking	2D blocking
L1 misses	$3.7 \cdot 10^8$	76%	73%	78%
L2 misses	$1.5 \cdot 10^8$	87%	93%	41%
L3 misses	$7.5 \cdot 10^7$	52%	19%	16%
CPU time	20.0s	13.3s	10.0s	8.0s

The DiMEPACK library

- Multigrid correction scheme: V–cycles, full multigrid
- Rectangular domains, $h_x \neq h_y$ is permitted, standard coarsening
- So far: 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
- Cache–optimized intergrid transfer operations
- Direct solution of the problems on the coarsest grid (LAPACK)
- Single precision/double precision floating–point arithmetic
- etc.

DiMEPACK is available on demand, contact *cache@in.tum.de*

Conclusions and future work

- Speedup factors of 2–3 are possible for stencil–based computations on structured grids
- Even higher speedups for constant coefficient problems
- Extension of the multigrid library *DiMEPACK*
- Inherently data–local algorithms and data structures
  e.g.:
  - Patch–based grid structures, patch–adaptivity
  - Adaptive smoothers
  - Fully adaptive multigrid methods (*FAMe*, U. Rüde)
- Practical application of our codes and techniques
- Unstructured grids:
  collaboration with C.C. Douglas, University of Kentucky
- Optimizing preprocessors *(ROSE)*:
  collaboration with D. Quinlan, Lawrence Livermore National Lab