Exascale techniques for Numerics for PDEs
Part II: some ideas for important challenges

Dominik Göddeke and Stefan Turek

Institut für Angewandte Mathematik (LS3)
Fakultät für Mathematik
TU Dortmund

dominik.goeddeke|stefan.turek@math.tu-dortmund.de

Dagstuhl Seminar 13381:
Algorithms and Scheduling Techniques for Exascale Systems
Schloß Dagstuhl, Germany
September 15–20, 2013

technische universität dortmund
fakultät für mathematik

Exascale challenges defined by DFG SPP 1648

Generated using `http://www.wordle.net`

Topics in this talk

Generated using `http://www.wordle.net`

Exascale challenge #1

Energy efficiency

Energy budget of (not only) big machines

Distribution of energy consumption (according to DOE)

- 40–60 %: processors and memories
- 10 %: interconnect and storage
- Remainder (up to 50 %!): infrastructure, i.e. lighting, cooling, ...

Current technology (June 2013)

- Top500 #1: SNB + Xeon Phi
  - 33.9 PFLOP @ 17.8 MW $\Rightarrow$ 525 MW @exascale
- Green500 #1: SNB + K20
  - 98.5 TFLOP @ 30.7 kW $\Rightarrow$ 312 MW @exascale

Road to exascale

- Power envelope of $\leq$ 20 MW 'feasible'
- At least 20–25 times more energy efficient technology than today
- **The money wall problem: 1 MW = 1 M $ per year**

A promising 'new' HPC architecture?

Smartphones and tablets

- Probably a couple hundred GFLOP/s in this room
- Under the hood: SoCs deriving principles from embedded systems
- Very low-power and low-energy, battery runtime and heat dissipation
- Now 'incl. HPC': multicore, FPUs, SIMD, caches, pipelines, ...

HPC at only 0.5–2 W per chip – time and energy to solution

- Are applications really more energy-efficient despite ...
  - ... more nodes because of lower memory/node (weak scaling)?
  - ... longer execution time due to much slower speed?
  - ... many more units to reach same execution time (strong scaling)?
  - ... and of course slack turning smartphones into cluster nodes?
- 0.25 EFLOP/s at 25% the electricity bill is exascale computing!

Let's try this in the field :)

Prototype cluster

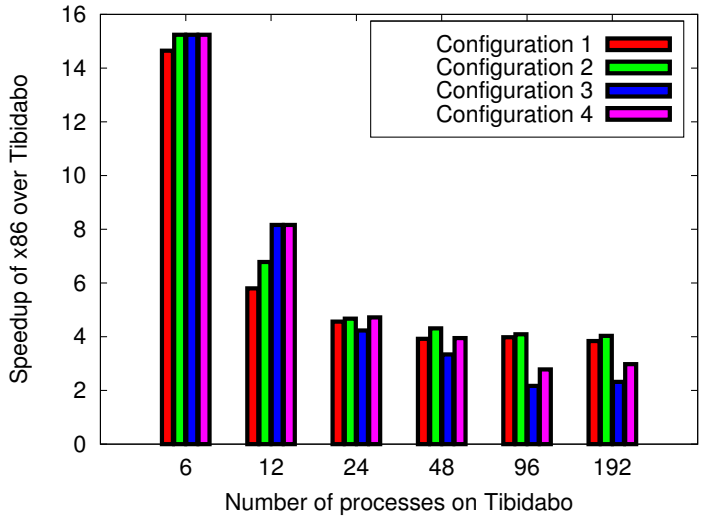
tibidabo @ BSC: EU project 'Mont Blanc'

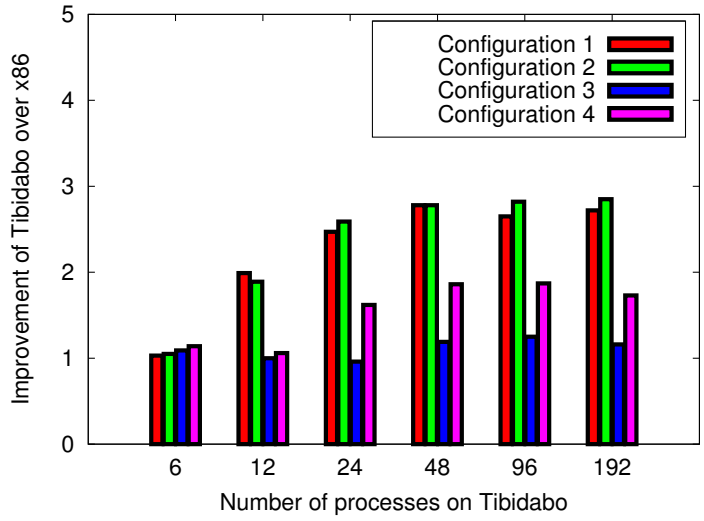
- 96 dual-core NVIDIA Tegra-2 SoCs based on ARM Cortex-A9
- Hosted in SECO Q7 carrier boards (nodes, essentially developer kits)
- 1 GB memory per dualcore, diskless (network FS), 1 GbE MPI
- Measured 5.7–7.5 W per node depending on load
- 2 W for the ARM alone (0.5 W for the dualcore CPU), plus other board components, USB, GbE, blinking LEDs (0.5 W!!!)

Porting effort

- Standard Linux OS with GNU compiler and debugger toolchain
- Main limitation: instruction issue rate
- Sophisticated cache-blocking (tiling) counterproductive, too much bookkeeping and index overhead
- Serial codes only reach half of the memory bandwidth per node

Power-performance analysis: FEAST

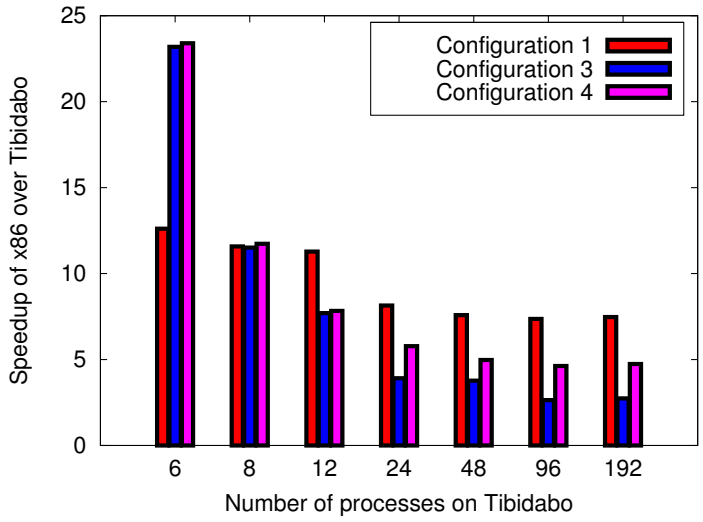


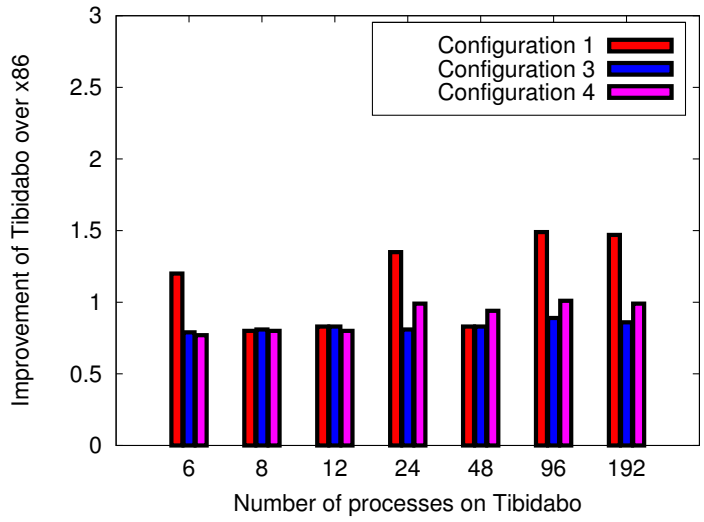


- Finite-element multigrid solver on block-structured grid
- Speedup x86 over ARM (left), improvement energy-to-solution ARM over x86 (right)
- Always more beneficial to use the ARM cluster
- As long as ≥ 2 x86 nodes are necessary, slowdown only 2–4

reference system: 32 2-way 4-core Nehalem system, 24 GB per node
(1) same load per core and partitioning (6 x86 cores/node), (2) re-partition for 8 cores/node, (3) as few x86 nodes as possible, (4) twice of that

Power-performance analysis: SPECFEM3D_GLOBE





- 3D seismic wave propagation modeling, high-order, explicit in time, spectral elements (unstructred hexas) in space
- Speedup x86 over ARM (left), improvement energy-to-solution ARM over x86 (right)
- Not always more beneficial to use the ARM cluster, compute-bound initially

Exascale challenge #2

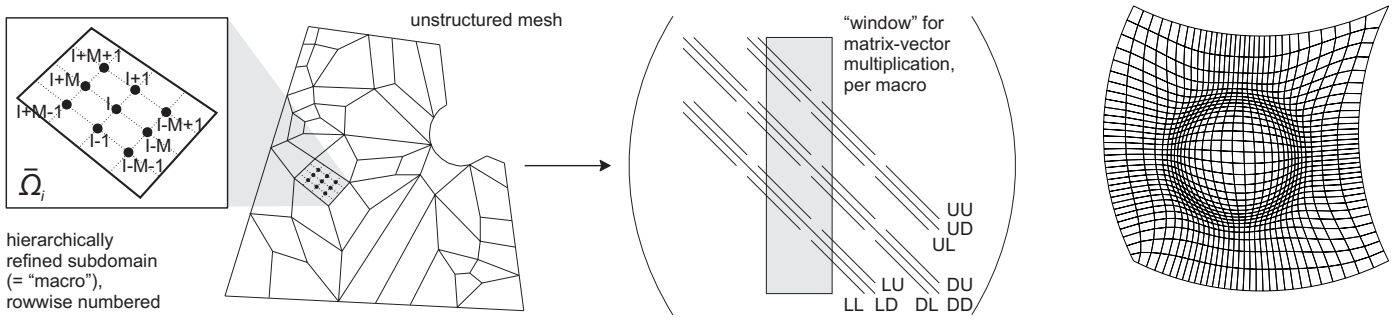
Exposing parallelism

for inherently sequential operations
and in FEAST: scalable single-node-performance

Recall: FEAST principles

Globally unstructured, locally structured

- Global macro-mesh: unstructured, flexible
- Local micro-meshes: logical TP-structure, banded matrices
- Structured $\neq$ cartesian meshes (r-adaptivity)

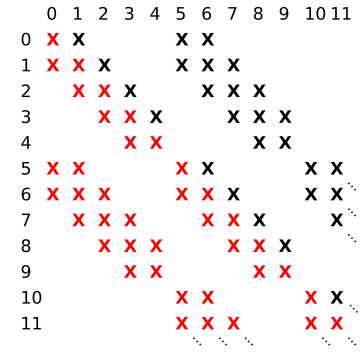


Today: MG components via structure exploitation

- Multigrid performance and numerical tradeoffs
- Strong smoothers: essential for numerical robustness and scalability

Gauß-Seidel relaxation

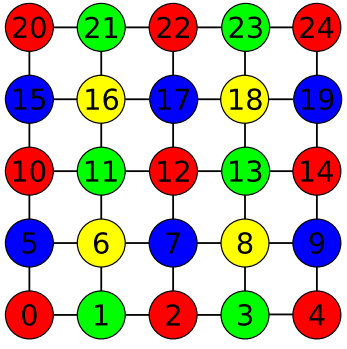
Disclaimer: not numerically competitive, but nice didactical example

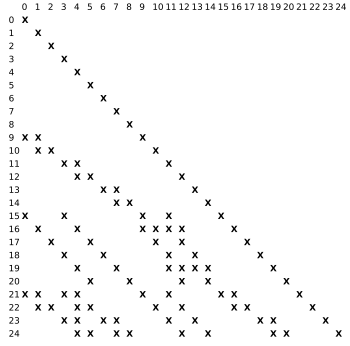


Sequential algorithm

- Forward substitution for $y = M^{-1}x$
- Intuition: 'left-bottom' coupling
- Inherently sequential

Parallel algorithm: decoupling through colouring



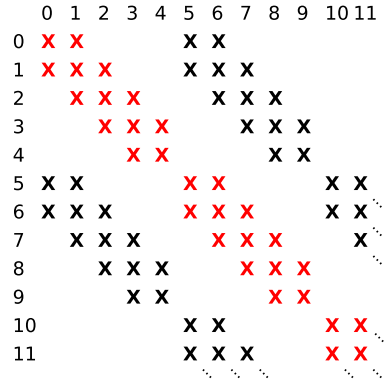


- Parallel efficiency: 4 sweeps with $\approx N/4$ independent parallelism
- Regular access pattern, checkerboard doable by manual caching

Line smoothers

Sequential / trivially parallelisable algorithm

- One tridiagonal system per mesh
- Mesh rows naturally decoupled
- Thomas: forward elimination of diagonal, then backward substitution
- OpenMP-parallelisation: trivial



Numerical advantages over Gauß-Seidel

- Optimal for 'aligned' anisotropies (choose when?)
- Extension: alternating direction implicit, swap row/colwise numbering

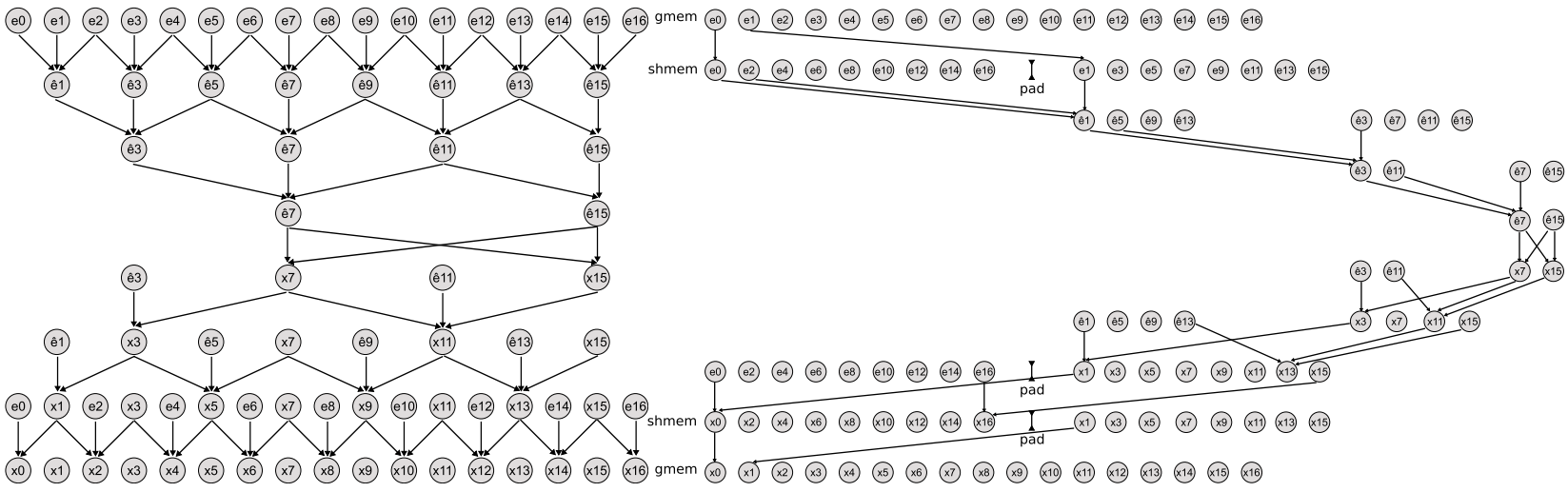
Problem

- Fine-grained SIMD parallelism?

Line smoothers

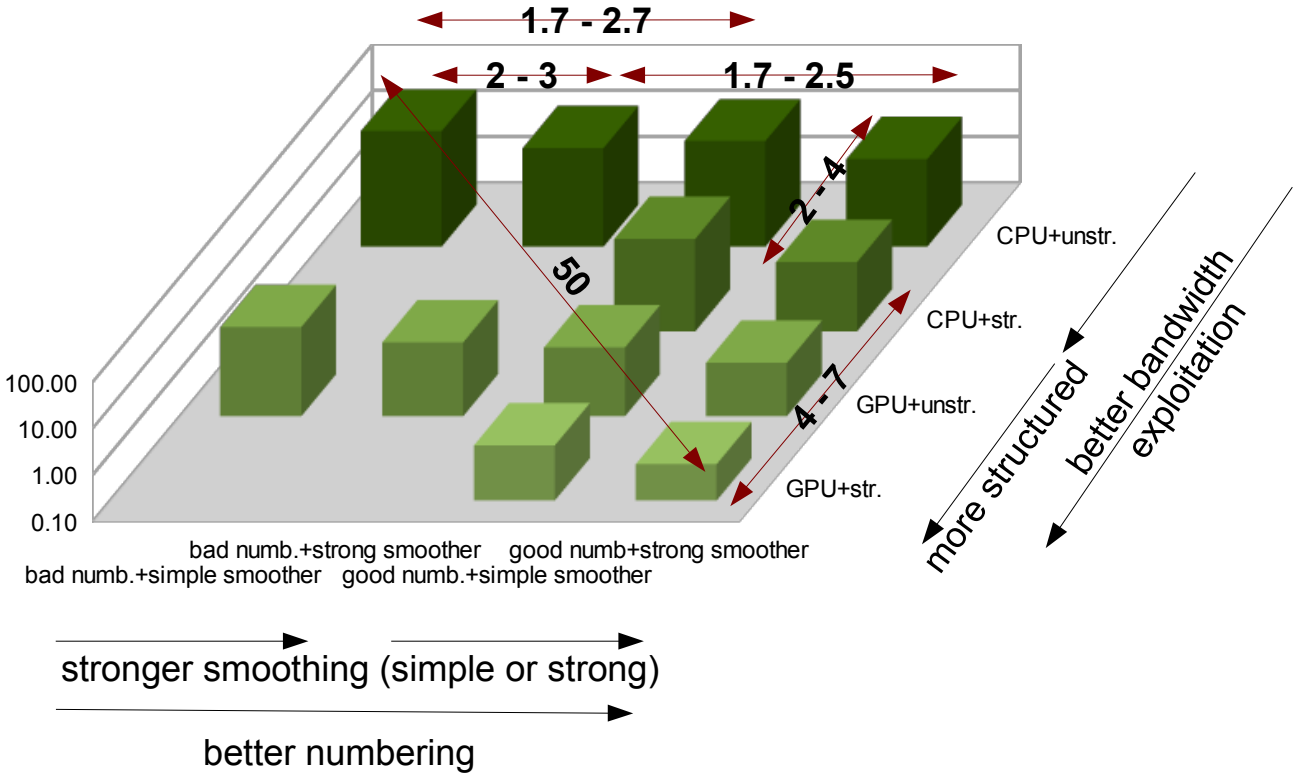
Cyclic reduction (ca. 1965-ish)

- Exact, stable (w/out pivoting), cost-efficient
- Recursive elimination of equations, then backward substitution



- Left (old): parallel computation, no parallel memory access
- Right (new): fully parallel, much faster

Benchmark example



- Q_1, Poisson on 'flow around a cylinder' unstructured grid
- GPU > mesh structure > smoother robustness > numbering

Exascale challenge #3

Algorithmic resilience

Fault tolerance

Challenge at scale

- Increasing probability of hardware failures, small MTBF
- From bitflips (in some exponent?) to loss of full nodes
- Software- instead of hardware-based resilience

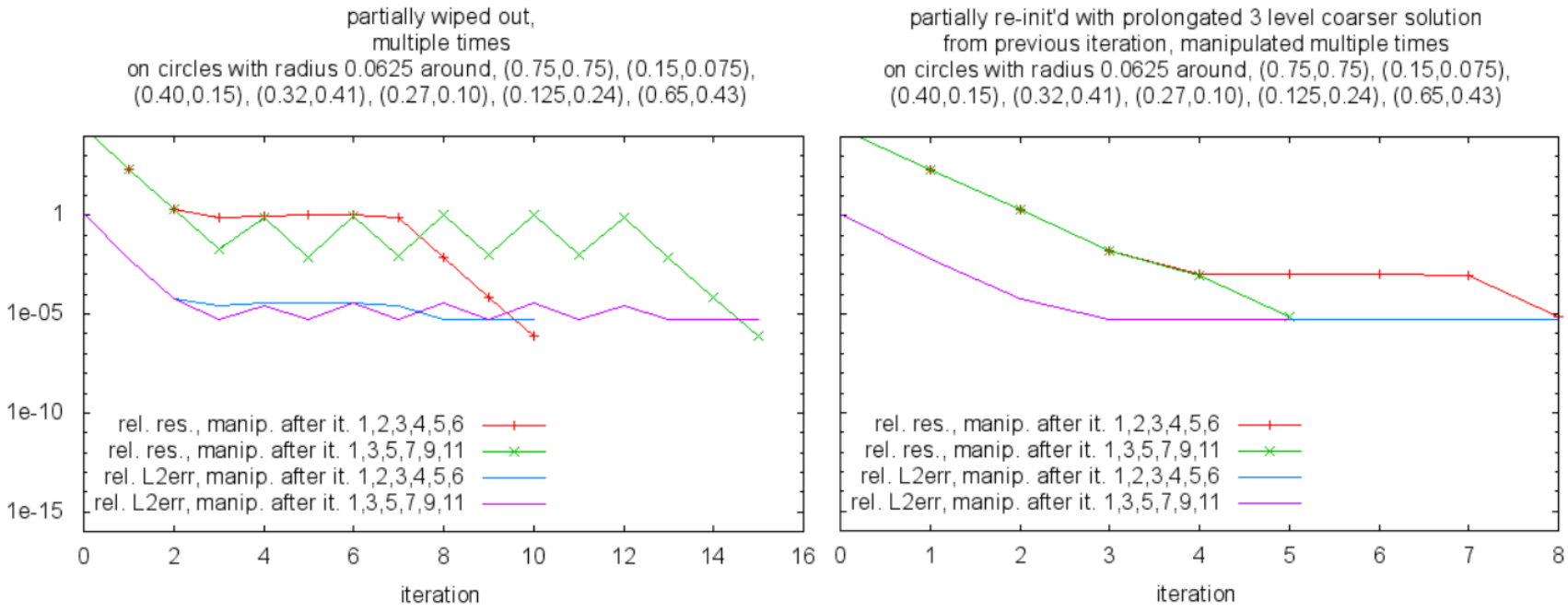
Classical approaches

- Checkpoint-restart of fine-mesh data to disk: too slow, tremendous data volume
- Redundant computations: too energy-consuming

Our idea: exploit multigrid idea

- Multigrid has 'built-in' data compression and is robust
- Checkpoint *exponentially smaller* coarser-level 'solutions'
- Restore with high-order prolongations
- Open numerical question: degree of compression

Numerical example



- Residuals (red, green) and L_2 errors (blue, magenta)
- Errors in iterations 1–6 (red,blue) and 1,3,5,7,9,11 (green, magenta)
- Left: no correction $\Rightarrow$ stagnation and oscillation
- Right: restauration from 64-fold smaller representation, same convergence as without errors as long as error frequency is small

Summary

Summary

Energy efficiency is a non-trivial question

- Speedup vs. greenup: time-to-solution vs. energy-to-solution?
- Weak and strong scaling equally important

Exposing parallelism

- Inherently sequential operations

Algorithmic resilience

- Hierarchy exploitation
- Compression rates

These are fun days for programmers and algorithm designers!

Dagstuhl: open inter-disciplinary challenges

Expertise in this room

Asynchronicity

- Varying compute and communication resources in one simulation
- Similar asynchronicity concepts from exascale to the cloud

Dynamic scheduling and load balancing

- Stefan's talk: 'general scheduling' challenges implied by hardware-oriented numerics
- Together: 'schedulability' of our numerical ideas?
- Possible coffee break topics
  - Intra- and inter-node
  - Numerical expert system, guesstimates of heterogeneous runtime
  - Dynamic load balancing and re-scheduling
  - Flexible mappings based on online data