Notions of Efficiency – The Role of Algorithm & Performance Engineering

Hans-Joachim Bungartz

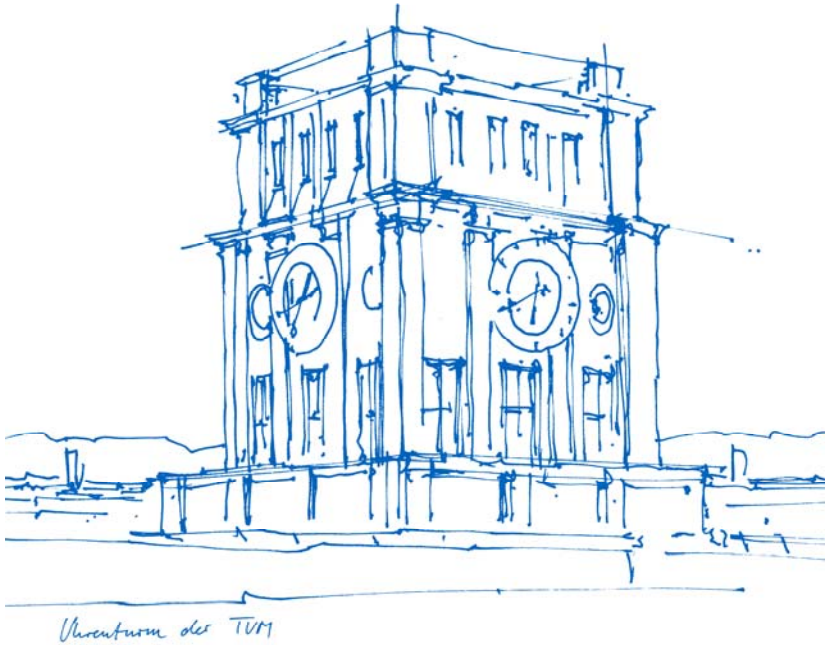
Technical University of Munich

Department of Informatics

Chair of Scientific Computing

Workshop “Big Data Meets Computation”

IPAM, Los Angeles, January 30, 2017



Contents

Preface

Algorithms & Performance in CSE & HPC

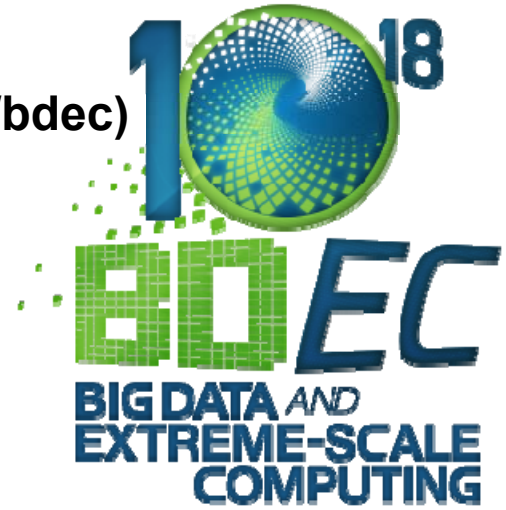
Some examples – or best practices, hopefully

- Eigensolvers / Material Science / ELPA @ FHI-aims
- Block-adaptivity / Plasma Physics / GENE
- Data structures & more / Chemical Engineering / ls1-mardyn
- Complete re-engineering / Geophysics / SeisSol

Concluding remarks

Preface: BD & HPC – not this Workshop's Invention …

… there are initiatives, such as BDEC (www.exascale.org/bdec)



- Wider activities started 2013
- World-wide participants (universities, research institutions, funding agencies, industry, …)
- Events (BOFs …) @ SC, ISC, …
- "Pathways to convergence" report

… there are national and international research agendas

… and there are professional societies

… but there is still a long way to go (this talk being more a proof for that than a solution ☺ …)

A Recent SIAM Activity

Workshop “Future Directions in CSE Education & Research”

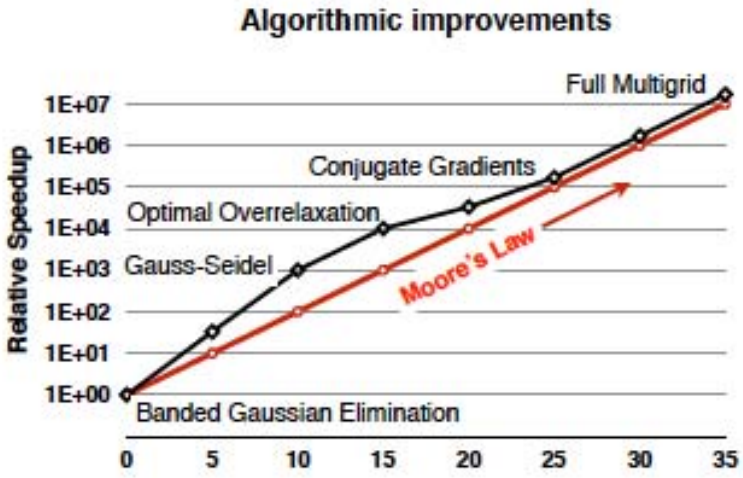
- August 2014, Breckenridge, CO
- Some 30+ participants from all over the world
- Co-organized by SIAM (its CSE branch) and EESI (European Exascale SW Initiative)
- Goals:
  - In general: positioning – re-shaping – forward-looking – branding – marketing
  - More concrete: update of the report “Graduate Education in CSE” (2001)
- Understanding of “Computational”, esp. w.r.t. HPC & Data Science

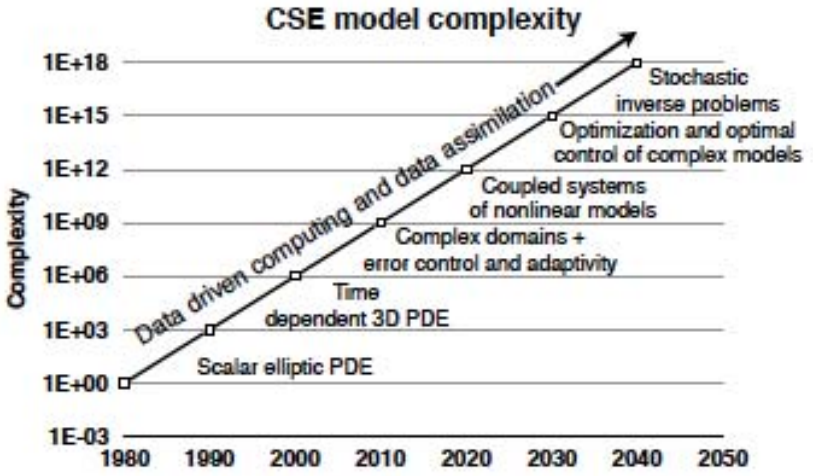
Final version currently under review

(authors: Rüde, Willcox, Curfman McInnes, De Sterck, et al.)

- Cover page
- Table of contents
- Central findings

Just two Figures from that Upcoming Report





Shows the interplay HPC-HW – HPC-Algo – HPC-Apps/Data

Expected to appear in SIAM REV, at or shortly after SIAM CS&E

Data & Computation – Incomplete pictures

The new, fourth paradigm / pillar:

1. Experiment
2. Theory
3. Simulation (with *Computational Science & Engineering* as a discipline emerged)
4. Data (*with Data Science & Engineering* as an emerging discipline)

➔ Neglects the fact that the core of both CSE and DSE is computing

The extended third paradigm / pillar:

1. Experiment
2. Theory
3. “Computational”: simulation, then optimization, now data

➔ Neglects the fact that there is “data” outside “computational”

CSE & HPC: The classical SIM&OPT cycle

Mathematical model

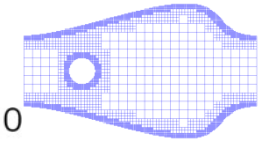
$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\,\mathbf{u} + \frac{1}{\rho}\nabla p - \nu\,\Delta\mathbf{u} = \mathbf{0}$$

$$\nabla\cdot\mathbf{u} = 0$$

Discretization & solver

$$M\mathbf{u}_h = 0$$

$$A\dot{\mathbf{u}}_h + D\mathbf{u}_h + C(\mathbf{u}_h)\mathbf{u}_h - M^T p_h/\rho = \mathbf{0}$$



Impact of each step on all other steps!

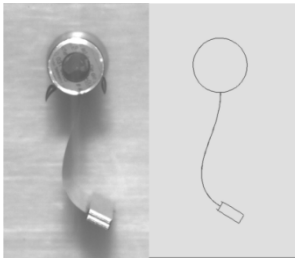
Today #1: no pipeline any more, no cycle, but a complete graph

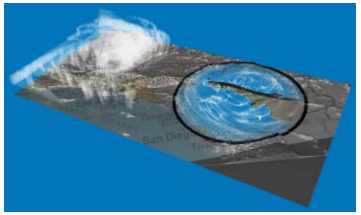
Toady #2: less space for single-field experts

Today #3: data to get integrated closer than via pre-/post-processing

Parallel implementation, HPC

Validation





Insight, Design

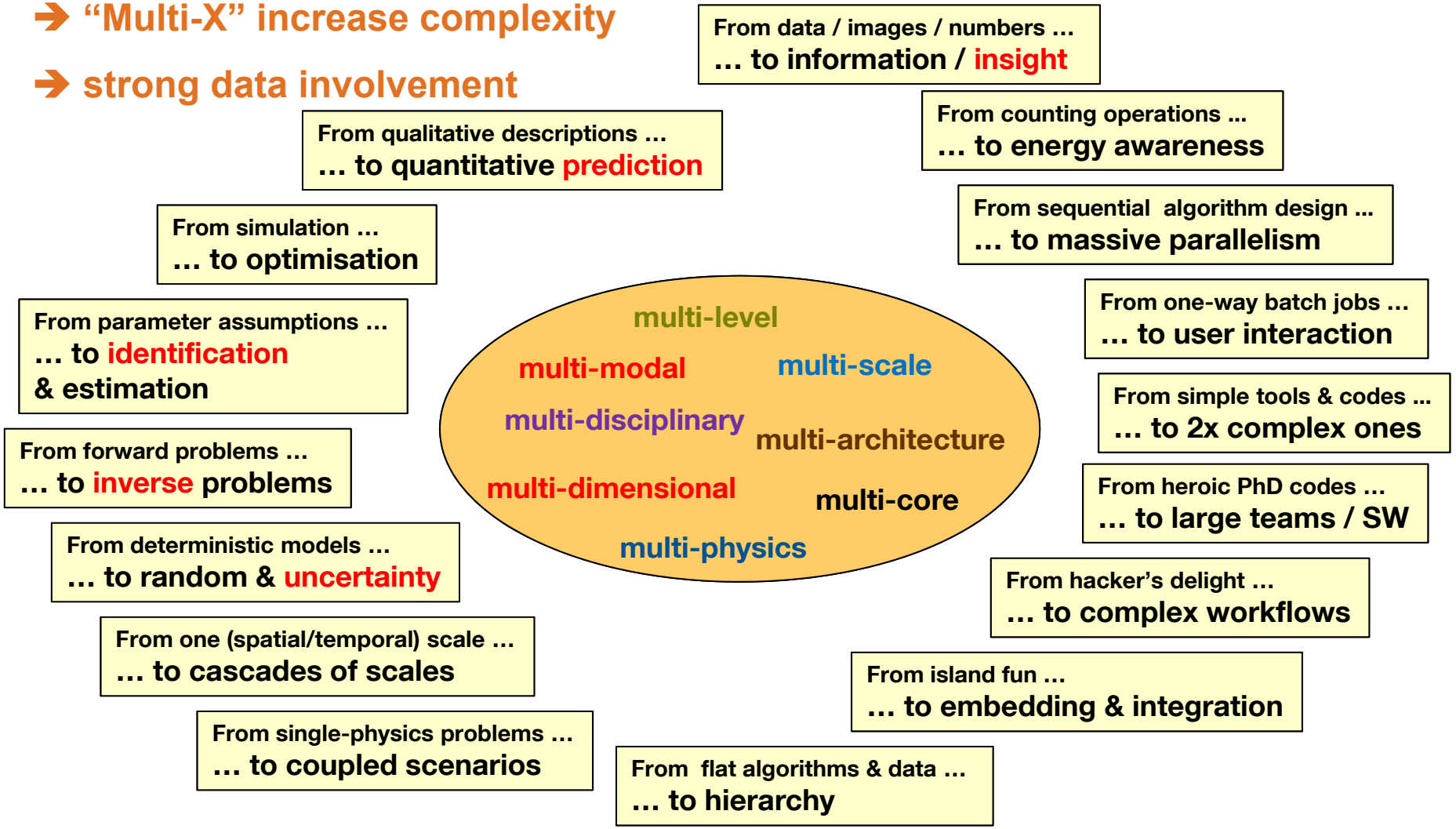
Exploration

Software

CSE&HPC Challenges

➔ “Multi-X” increase complexity

➔ strong data involvement



What is Changing / Will Change / is Necessary to Change

Closer interaction of the involved disciplines:

- Numerical Mathematics & Scientific Computing
- Modeling
- Statistics
- Data Analytics & Machine Learning
- Scientific Domains

Closer interaction of data-driven simulations, simulation-driven HPC, analytics-driven HPC, HPC-enabled simulation, HPC-enabled analytics, statistics&stochastics&numerics, and …

Finding the right interplay: simulations did not "kill" experiments, data analysis won't "kill" simulations ➔ new possibilities of linking data

Efficiency concern spreading out from HPC

To put it simple: more HPC in Big Data, more Big Data in HPC

Observation: Portion of data analytics or data-driven simulations is increasing in almost all HPC centers ➔ algorithm engineering & performance engineering must adapt their focus

Contents

Preface

Algorithms & Performance in CSE & HPC

Some examples – or best practices, hopefully

- Eigensolvers / Material Science / ELPA @ FHI-aims
- Block-adaptivity / Plasma Physics / GENE
- Data structures & more / Chemical Engineering / mardyn
- Complete re-engineering / Geophysics / SeisSol

Concluding remarks

Algorithms are the essence

Functionality

- Discretization schemes, numerical schemes
- Parallelization, communication, load balancing
- Compression, dimension reduction
- Statistic, stochastic
- Analytical (ML, …)

Algorithmics: design – analysis – implementation – tuning

➔ **Algorithm Engineering: design – analysis** – implementation – tuning

➔ **Performance Engineering:** design – analysis – **implementation – tuning**

Efficiency – absolute and relative notions:

- **#ops – #flops – #%peak – #bytes – #Watt – time-to-solution, …**
- **digits per flop, flops per Watt, flops per byte, science (pubs) per flop, science (insight) per flop, …**

Cost and Benefit of Numerical Algorithms

- **General Principle:**
  - For a continuous problem **P** with solution u, provide discrete approximations $\mathbf{P}_n$ and u_n such that some error ε gets small.
- **Cost-benefit consideration (cf. notion of ε-complexity):**
  - **benefit**: obtained error/accuracy ε
  - **cost (I)**: the classical ones – $N(\varepsilon)$ degrees of freedom to obtain ε, $g(N)$ ops. to solve $\mathbf{P}_n$
  - **cost (II):** (parallel) runtime $T(g(N))$, energy consumption $E(g(N))$ for building & solving $\mathbf{P}_n$
  - **cost (III):** more metrics – memory efficiency, cache efficiency, communication avoiding, …
- **Classical approach (*d* dimensions, regularity *p*):**
  - exponential growth of N: $N = O(\varepsilon^{-d/p})$ (curse of dimension)
  - polynomial growth of g in N: $g = O(N^k)$
  - T and E just observed, tunings for T, more experience than theory … *[but be aware! ☺]*
- **Example:**
  - $\varepsilon = 10^{-4}$ (4 digits), $p = 1$ (first-order regularity), $d = 4$ (think of CM – space plus time)
  - $k = 2$ (a non-optimal non-multilevel solver, as it is still widespread)
  - result: $N = O(10^{16})$, $g = O(10^{32})$, T & E already relevant at all?

Cost and Benefit of Numerical Algorithms

- **General Principle:**
  - For a continuous problem **P** with solution u, provide discrete approximations $\mathbf{P}_n$ and u_n such that some error ε gets small.
- **Cost-benefit consideration (cf. notion of ε-complexity):**
  - **benefit**: obtained error/accuracy ε
  - **cost (I)**: the classical ones – $N(\varepsilon)$ degrees of freedom to obtain ε, $g(N)$ ops. to solve $\mathbf{P}_n$
  - **cost (II):** (parallel) runtime $T(g(N))$, energy consumption $E(g(N))$ for building & solving $\mathbf{P}_n$
  - **cost (III):** more metrics – memory efficiency, cache efficiency, communication avoiding, …
- **Classical approach (d dimensions, regularity p):**
  - exponential growth of N: $N = O(\varepsilon^{-d/p})$ (curse of dimension)
  - polynomial growth of g in N: $g = O(N^k)$
  - T and E just observed, tunings for T, more experience than theory … *[but be aware! ☺]*
- **Starting points for improvements:**
  - fast solvers for $\mathbf{P}_n$ (multilevel): g **optimal** (linear in N, for example)
  - efficient discretizations: **improve** N (small, no or only weak d-dependence)
    - approaches of higher order, adaptive mesh refinement (problem-dependent)
    - sparse grids **(a priori grid structure**, not depending on given problem)

Frequently discussed in HPC: Co-Design

Not the original from the embedded world: HW – SW

But the frequent copy in the HPC world: systems – applications
(i.e. for a computing center: center – users)

The incarnation – and a huge issue: tuning of sim codes for current systems … and preparing next-gen HW for well-established applications??

There is a lot between Applications & Systems

(My) experience: application codes often …

- … reside, have been developed, and are maintained in groups representing the respective application domain
- … are inspired and driven by researchers standing for concrete simulation scenarios and the underlying (phys./math.) models
- … are not putting too much emphasis on algorithmic aspects (“we scanned the literature, found that method, and implemented it”)
- … prefer minimal-invasive code surgeries (“my poor baby …”)
- … look for painless performance gains:
  - “buy a bigger machine” preferred to “change the code”
  - “change the code” preferred to “change the underlyings”

Not blaming anyone, but often:

➔ code not really ready for tuning, need for algorithmic re-engineering, first

➔ need for collaboration

The Applications' Perspective: Help Needed!

PA: Where are the bottlenecks (sequ./par.) with that code?

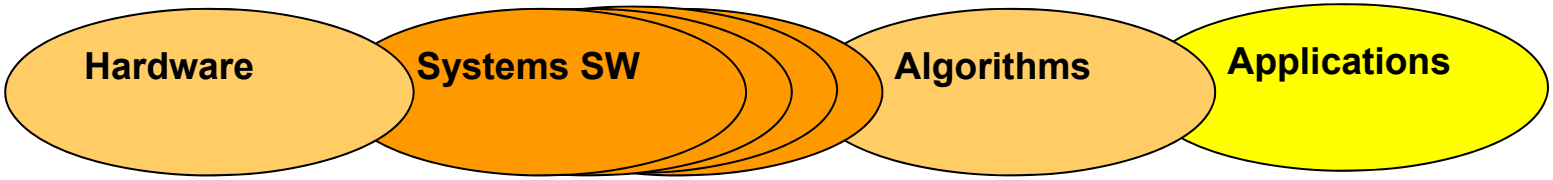
- Take model and algorithms as given (if apparently fine)
- Use existing PA tools
- Principally satisfied with functionality – but hard to understand/exploit as a part-time performance engineer (not only a sub-optimal visualization issue!)

PO (1): What can be done internally?

- i.e. without changing model & algorithm essentials ➔ advice should come from PA

PO (2): What can be done externally?

- i.e. leaving the minimal-invasive terrain ➔ hints typically more hidden in/by PA data



Hypothesis

Co-design of applications and *algorithms*

PLUS

Co-design of *algorithms* and SW-stack / systems

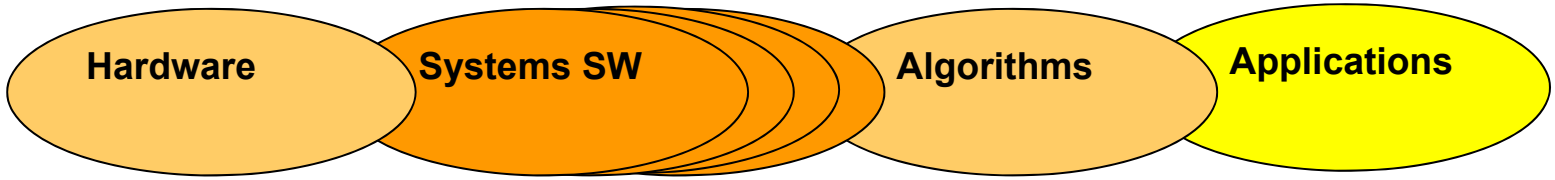
… in a tightly interwoven way (… and maybe even more sub-steps)

A shortcut *"applications & systems"* is often not equivalent!

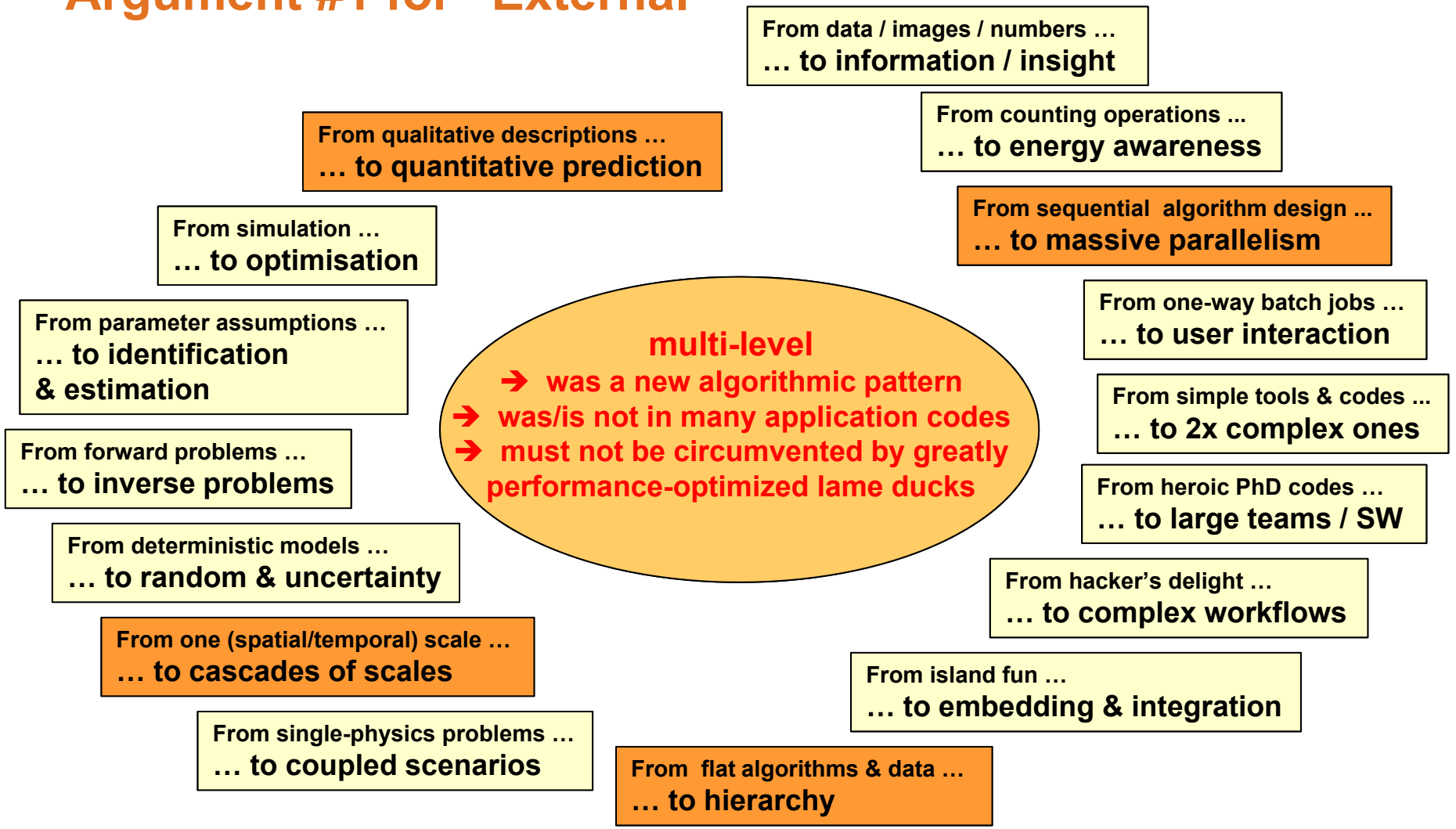
➔ *what about new/future algorithms?*

➔ *"100 hours of 80% peak vs. 10 hours of 8% peak" (heretic, OK …)*

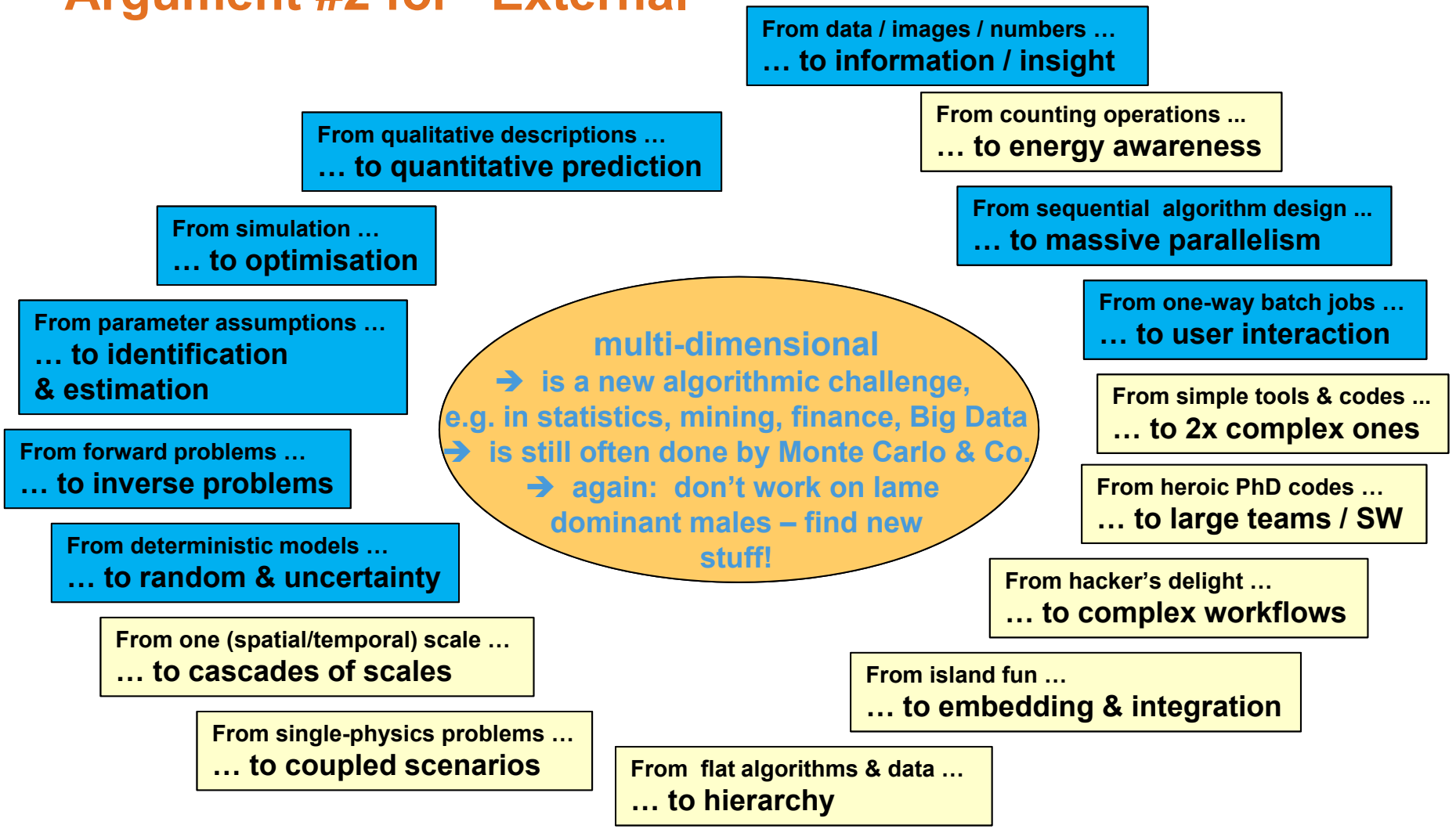
➔ Next question: *"10 hours if it's not the proper thing?"*



Argument #1 for “External”



Argument #2 for “External”



Contents

Preface

Algorithms & Performance in CSE & HPC

Some examples – or best practices, hopefully

- Eigensolvers / Material Science / ELPA @ FHI-aims
- Block-adaptivity / Plasma Physics / GENE
- Data structures & more / Chemical Engineering / mardyn
- Complete re-engineering / Geophysics / SeisSol

Concluding remarks

Example #1 – ELPA

T. Huckle, T. Auckenthaler, M. Rippl, …

Code: FHI-aims (Max-Planck-/FHI-hosted electron structure code / DFT)

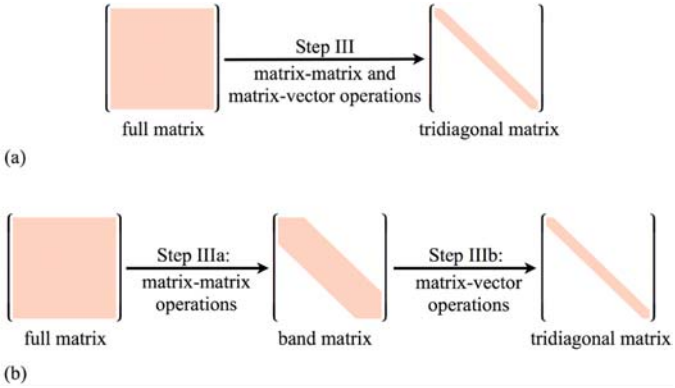
- **Bottleneck**: symmetric eigenvalue solvers (scalability)
- **Path**: BMBF-funded projects ELPA (‘09-’12) & ELPA-AEO (‘16-’19)
- **Team**: mathematics (BGU Wuppertal, informatics (TUM), chemistry (FHI, TUM), computing center (MPCDF)
- **Algorithm**: 2-step tridiagonalization, block reduction; full & sparse
- **Parallel**: starting with BG/P, Power6, hybrid MPI-OpenMP, …
- **Strategy**: most of the development/analysis/tuning outside the application code, but frequent transfer/integration
- **Competitors:** MAGMA (Dongarra), EIGENEXA (Riken, Imamura)
- **Performance**: competitive, often leading (flagship paper ‘14)
- **Results**: one improved code plus public library routines, widely used

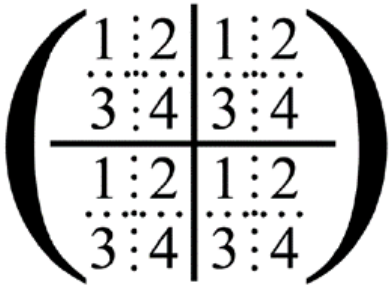
➔ gain by algorithmic improvement / change

➔ driven by one code/application, but general algorithmic result

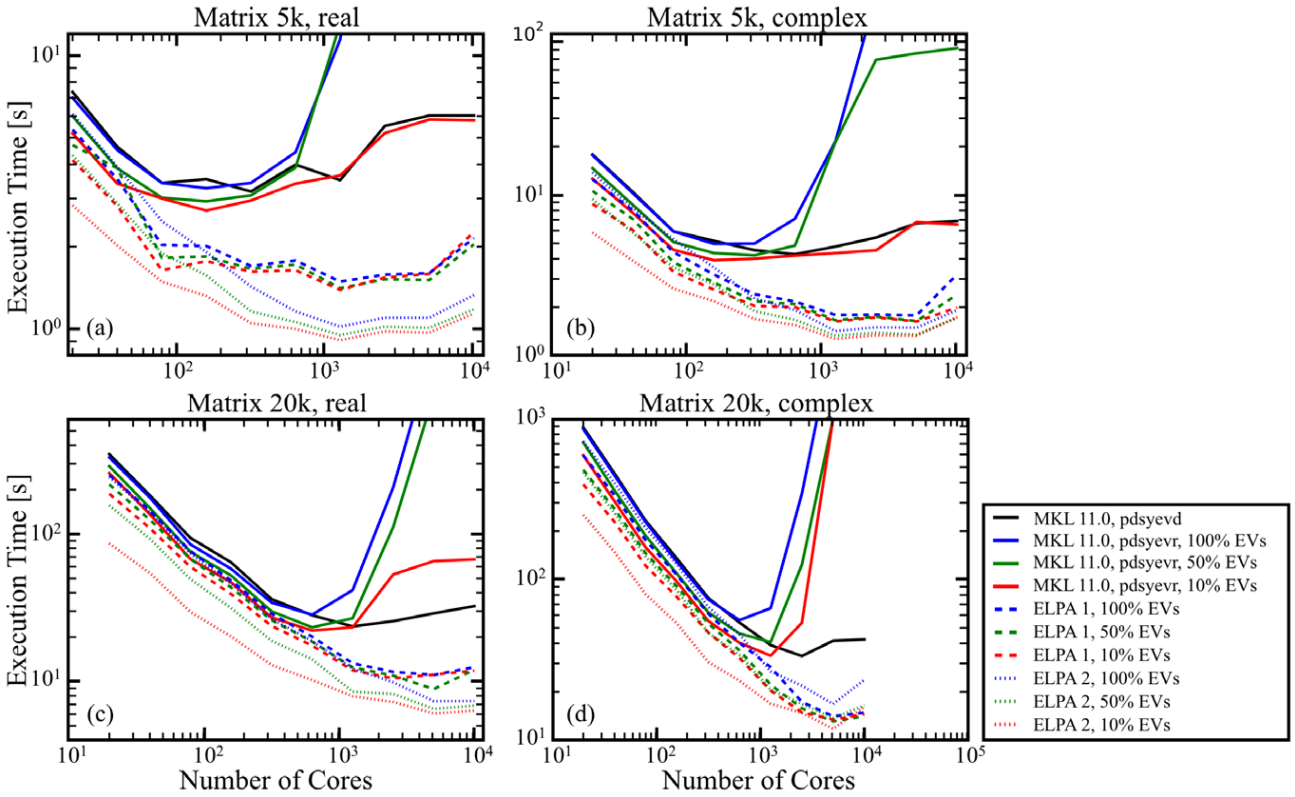
ELPA – Algorithmics

- **Starting point:** dense eigensolvers strongly sequential, only BLAS1/2
- **Main idea:** 2-step transformation ➔ banded ➔ tridiagonal
- **Advantage:** #1 blockwise BLAS3, #2 cheap, #3 (solve) via D&C
- **Further goals:**
  - Efficient data distribution (block-cyclic)
  - Efficient QR decomposition (block-Householder, Tall&skinny QR, …)
  - General eigenvalue problems
  - Include further solvers (FEAST, MRRR)



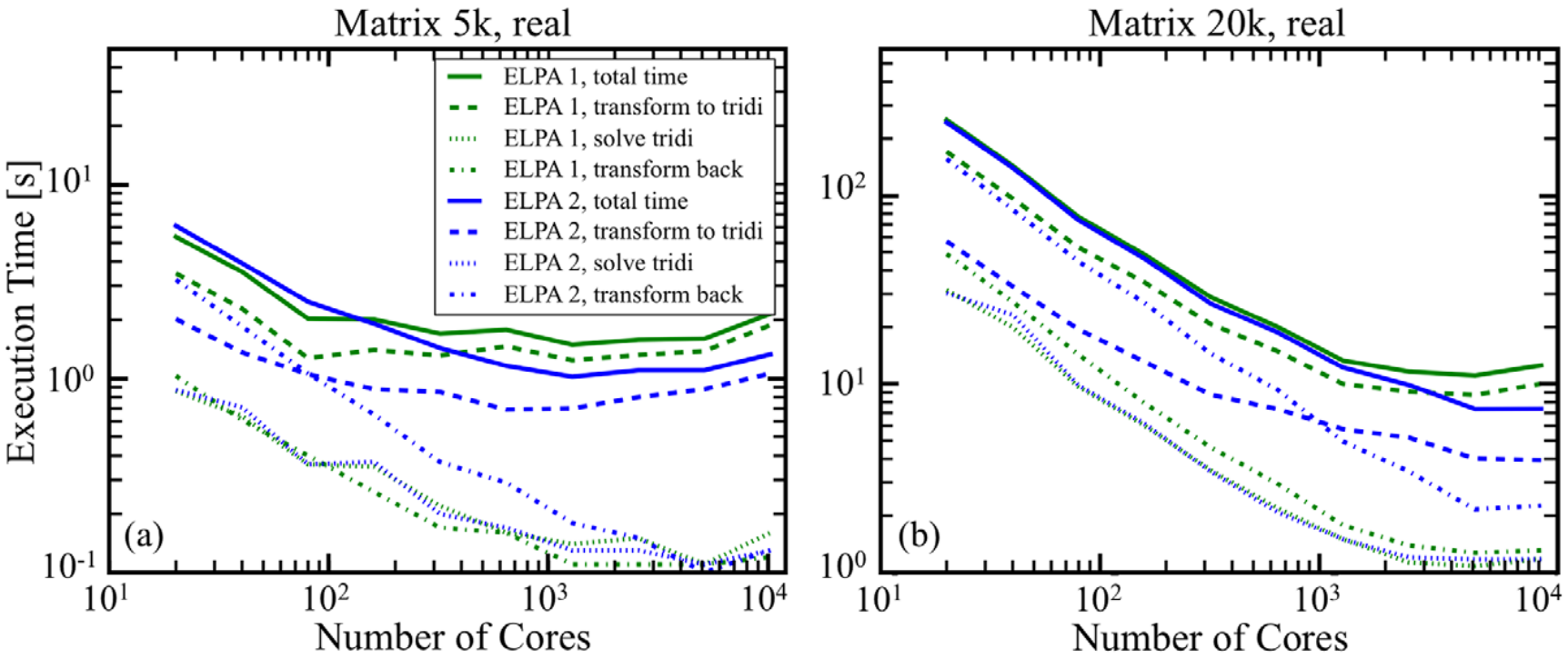


ELPA – Results



ELPA and MKL 11.0 on Intel SandyBridge.
20 cores (1 node) up to 10 240 cores (512 nodes). 4 different matrices (real/complex, N=5,000/20,000).
For each matrix, fractions of 100%, 50%, and 10% of the eigenvector spectrum were calculated.

ELPA – Results



ELPA 1 and ELPA 2 for real valued matrices.
Size *N* = 5000 (*a*) *N* = 20 000 (*b*), 100% of the eigenvalue spectrum is computed. Shown are the total time to solution (solid) and the timings of the three major steps transformation to tridiagonal form (dashed), solution in tridiagonal form (dotted), and back transformation (dashed–dotted). For large processor counts, the transformation to tridiagonal form is the most expensive and also determines the scaling behavior.

Example #2 – GENE

C. Kowitz, T. Neckel, D. Jarema, A. Parra, F. Jenko, …

GENE: Gyrokinetic Electromagnetic Numerical Experiment

highly parallel and heavily used Eulerian gyrokinetic code for plasma turbulence simulations

solves 5D integro-differential system of equations (Vlasov's & Maxwell's)

http://genecode.org

Example #2 – GENE

C. Kowitz, T. Neckel, D. Jarema, A. Parra, F. Jenko, …

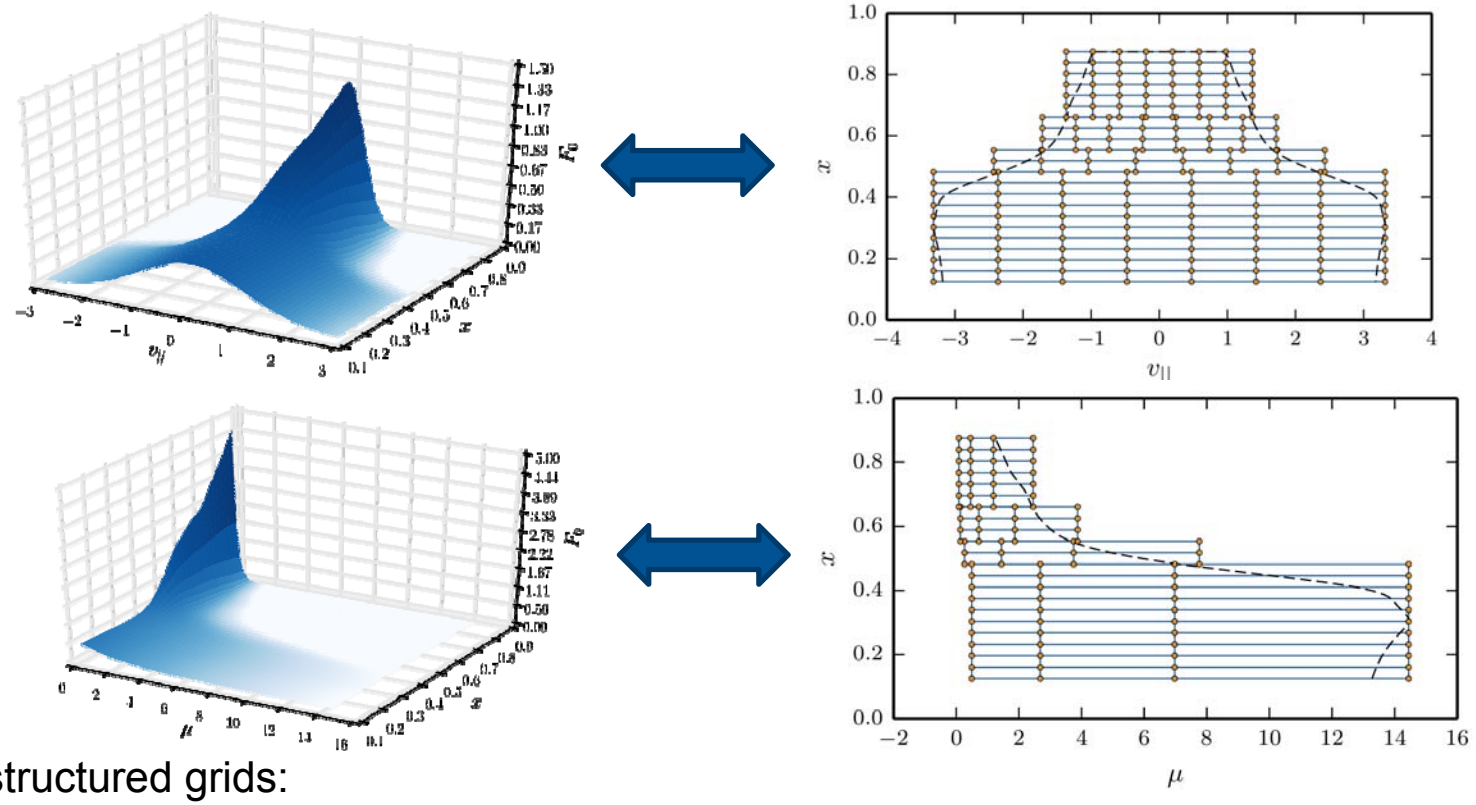
Code GENE (Max-Planck-/IPP-hosted plasma physics code, cf. Frank & Dirk)

- **Pros:** wide HPC experience of the team, large-scale scalability
- **Bottleneck**: restrictions in problem size (“Numerical ITER” far away)
- **Algorithmic diagnosis**: simple solvers, no adaptivity, dimensionality (5/6)
- **Paths**:
  - Solvers: sophisticated preconditioners implemented
  - Adaptivity: implement first block adaptivity, then maybe local adaptivity
  - Dimensionality: use sparse grids to overcome the curse of dimensionality
- **Team**: maths (ANU, U BN), CS (TUM, U S), HPC (LRZ, RZG), physics (IPP)
- **Strategy**: strict algorithm-application co-design; development versions
- **Results**: recent results on solvers and dimensionalities very promising

➔ gain by algorithmic improvement / change; PA/tuning interwoven

Algorithmic improvements: Block-structured adaptivity

Steep temperature radial profiles lead to varying velocity space structures ➔ adaptivity!

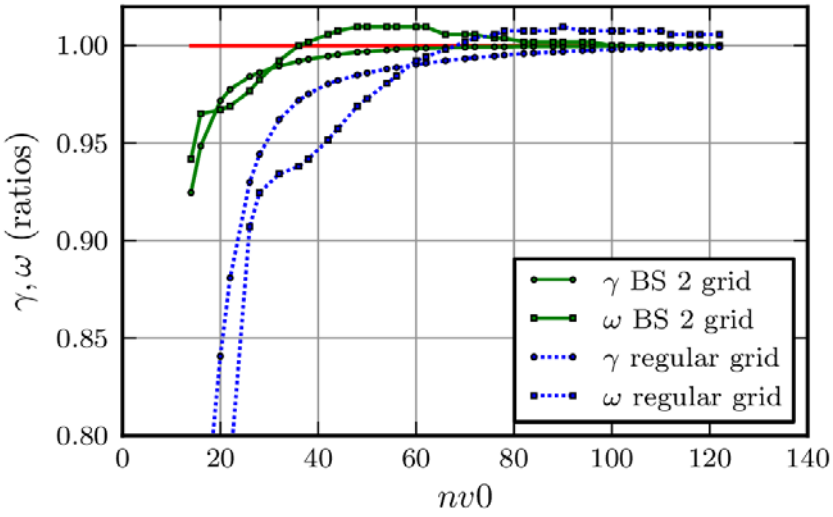


Block-structured grids:

- adjust velocity space ranges and resolutions correspondingly
- preserve physical velocity coordinates (to avoid nontrivial changes in equations)

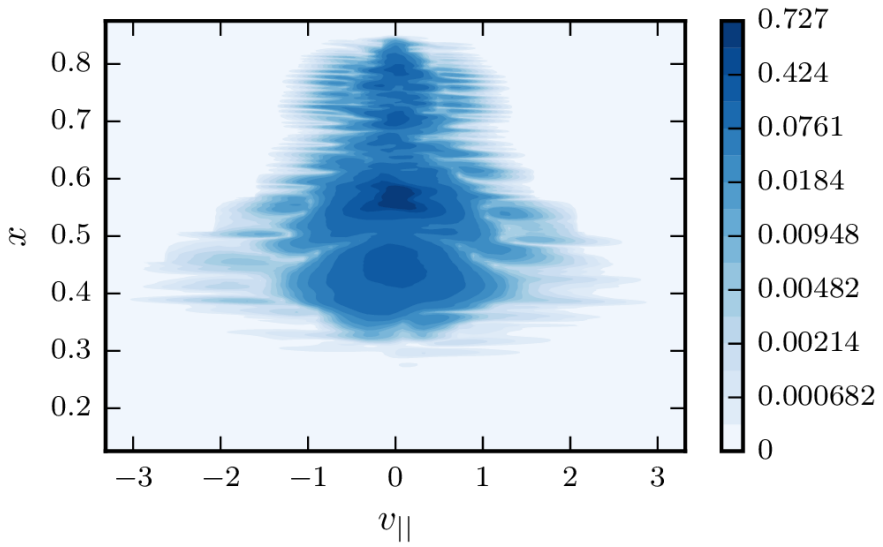
Block-Structured Grids in GENE: Selected Results

Linear simulations



- Much faster convergence of growth rate γ and frequency ω compared to original regular grids
- Speedup depends on temperature profiles, in the tested scenarios ~ 2

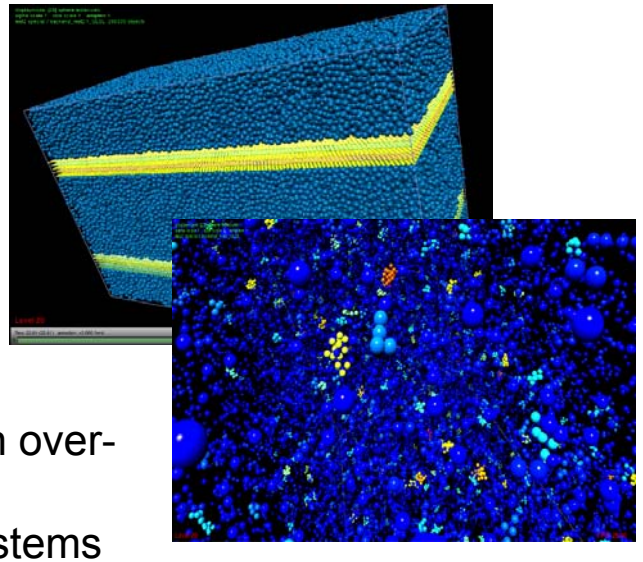
Non-linear simulations



- Block-structured grids resolve the perturbed part of the distribution function with much less discretization nodes than original regular grids
- Observed speedup ~ 10

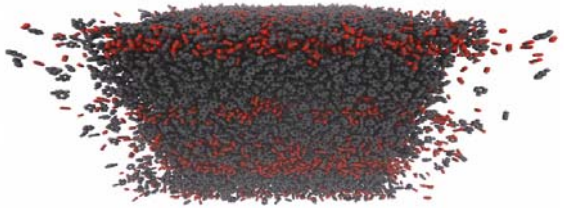
Example #3 – ls1-mardyn

M. Buchholz, W. Eckhardt, P. Neumann, N. Tchipev, S. Seckler, …



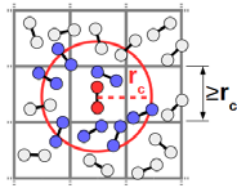
Code: ls1/mardyn

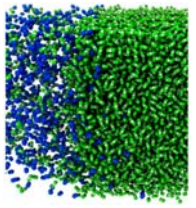
- MD code, simulation of arbitrary mixtures of rigid, multi-centered molecules (ext. versions exist)
- Chemical reactors, better understanding of nucleation in over-saturated / heterogen. settings
- **Engineering** & **industrial** relevance: targeting large systems & larger time scales
- MPI-parallel, written in C++
- Joint venture: TU KL, U PB, HLRS, LRZ, TUM

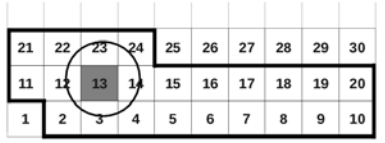


Simulation & PO context:

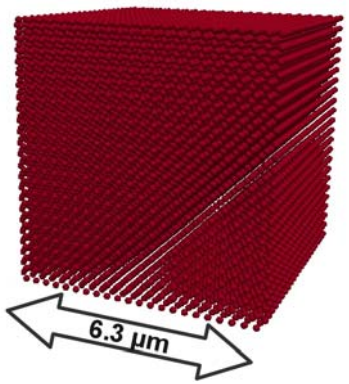
- PO anticipating large-scale experiments to study the size dependence of interfacial quantities such as surface tension
- Hand-coded compute kernels for AVX, tailored to Intel SB
- Light-weight shared-memory parallelization with OpenMP to exploit hyper-threading (12% performance gain)
- Memory optimization: only 32 Bytes per molecule required

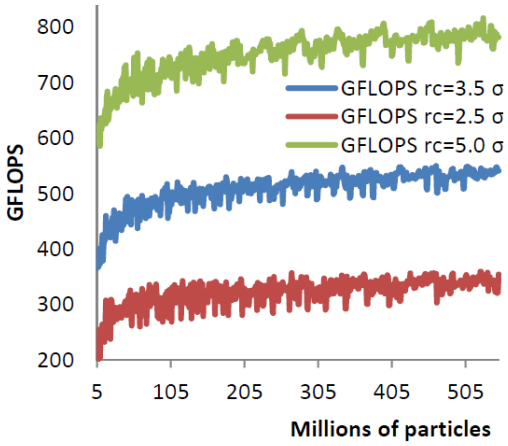


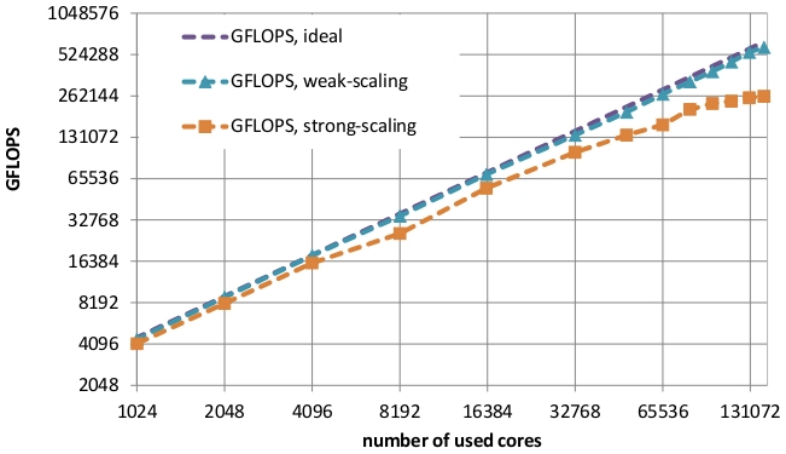




Memory Efficiency: a look at the 2013 World Record for largest Molecular Dynamics Simulation







Simulation Details:

- 4.125 * 10^{12} particles
- Single center Lennard Jones
- Number density $\rho\sigma^3 = 0.78$
- **Cutoff radius $r_c=3.5\sigma$**
- Krypton: cube side-length 6.3 µm
- 146016 MPI x 2 OpenMP

PRACE Award 2013

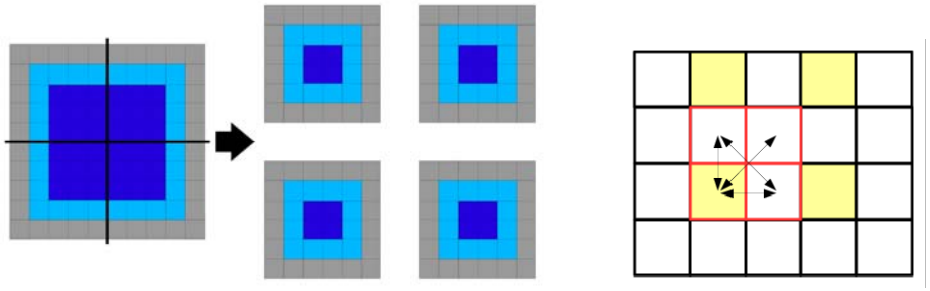
Weak scaling:

- 4.52*10^8 particles / node
- On 146,016 cores (292,032 Threads):
  - 591 TFLOPS
  - 9.4% of peak performance
  - Runtime 40 s / iteration
  - Maximum number of 4.25*10^{12} particles (world record)

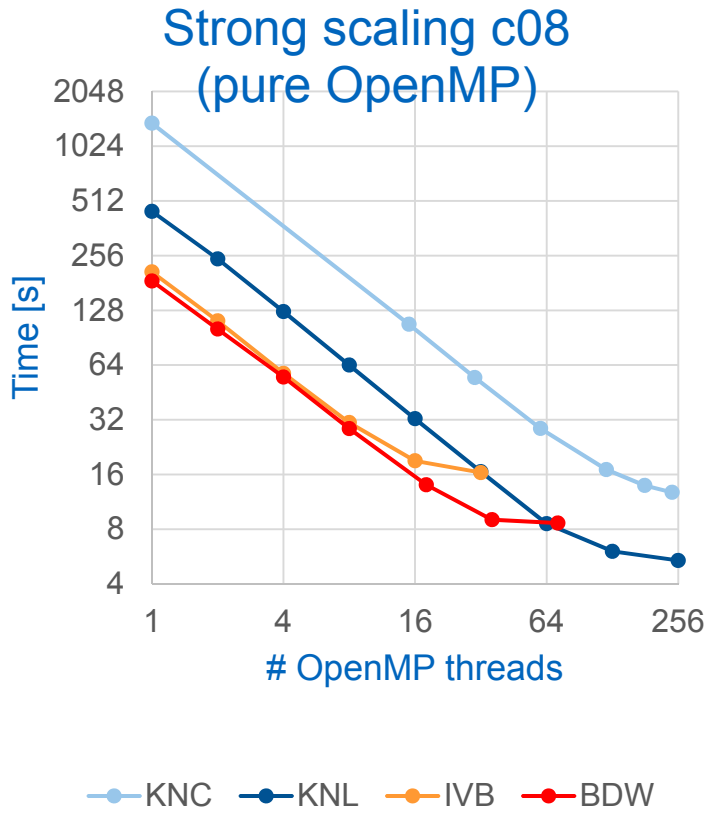
Memory Efficiency: What Xeon Phi taught us

native/symmetric mode on Knights Corner – problems arrived quickly:

- need up to 240 threads or processes to utilize full KNC potential
- can‘t fit so many MPI ranks in just 8GB RAM!
- Need for a scalable, memory-efficient OpenMP!

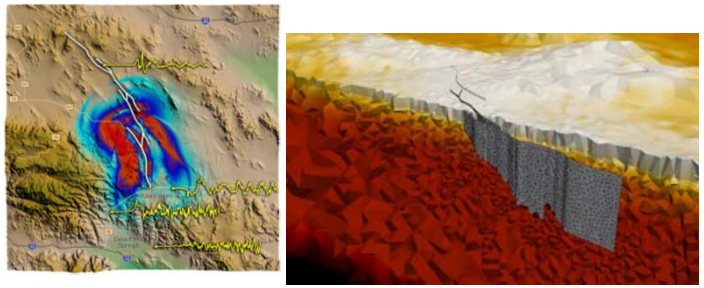


Solution found: c08 coloring scheme!



Example #4 – SeisSol

M. Bader, A. Breuer, …



Code: SeisSol – www.seissol.org

- Multiphysics simulation of dynamic rupture and resulting ground motion
- Unstructured adaptive tetrahedral meshes to resolve complex geometries (branched fault systems, geological material layers)
- ADER-DG discretisation: high-order in space and time
- **Bottleneck**: critical intra-element performance (MatMatProd)
- **Algorithmic diagnosis**: need for tailored fast matrix kernels, among others
- **Path**: create an enriched offline phase that generates optimized and vectorized kernels for each operation (sparsity patterns are known a-priori)
- **Team**: CS (TUM), HPC (LRZ), geophysics (LMU), in parts Intel & IBM
  [Heinecke/Breuer/Rettenberger/Bader (TUM/LRZ), Pelties/Gabriel (LMU), Smelyanskiy (Intel)]
- **Strategy**: rigorous re-design in those algorithmic components
- **Results**: significant increase in speed about 1.4 PF on SuperMUC, + MIC
  ➔ gain by algorithmic & code improvement

SeisSol+SuperMUC/MIC – the Perfect Match?

Standard computing center approach (done by LRZ user support):

- Compile with architecture flags, e.g. -mavx, -mmic
- Identify hot loops and regions by performance analysis tools
- Add pragmas (vectorization or parallelization) to hot loops or code regions
- Do not change / critically reflect the code (solver, communication scheme, …)

Results:

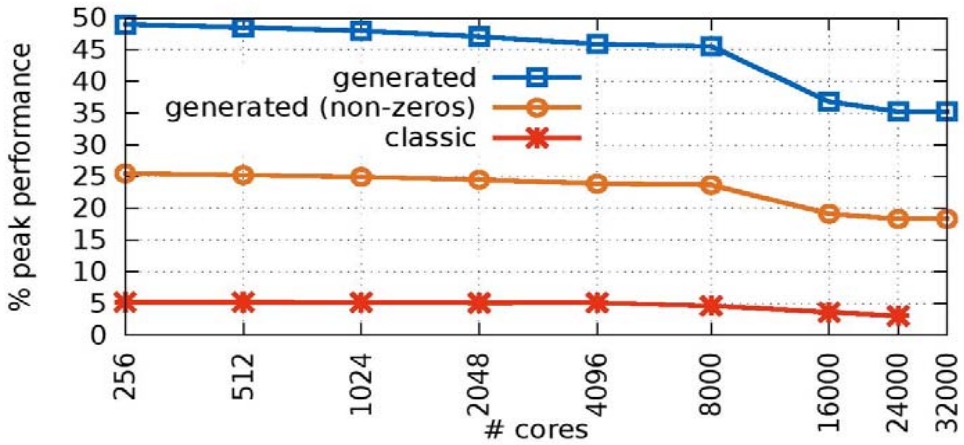
- Scaling on MIC (single card) identical to the scaling on SuperMUC
- SeisSol cannot be scaled across multiple coprocessor cards
- Real-world scenarios cannot be simulated due to limited memory size on the card
- Single MIC card is 25X slower than dual-socket SuperMUC node

Conclusion: (OK, but shows the difference between Perf. & Alg. Eng)

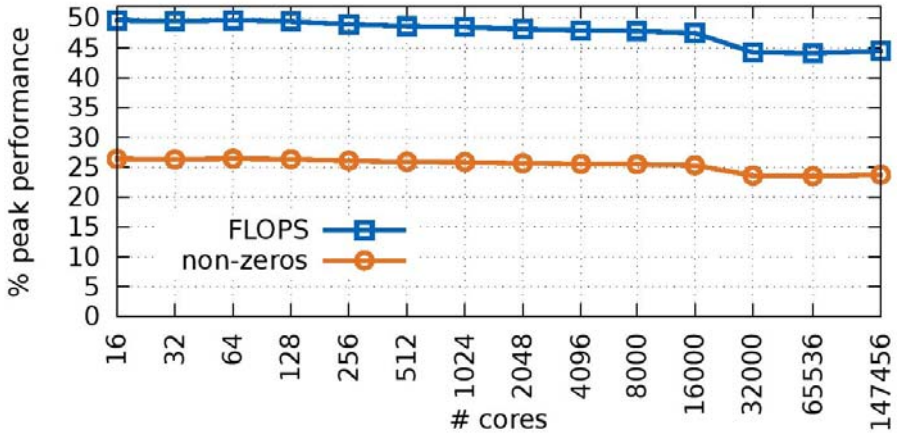
"Regarding the maturity of MIC-clusters, we would consider Intel Xeon Phi based clusters still as prototype systems and not really suited for production runs of real-world MPI based applications yet."

➔ This should not be the end of the story, if algorithm design based on abstract hardware concepts is applied

Then a 2nd Try … 1.4 PF @ SuperMUC (sustained, weak)



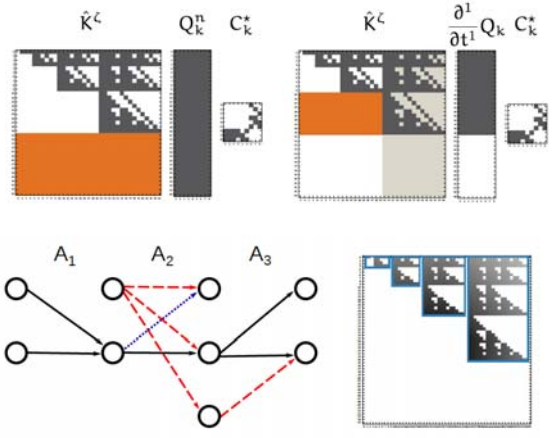
- 7,252,482 cells, 6th order (3.6 B unknowns)
- 2.71 TF on 256 cores (49% peak), 246 TF on 32K cores (35.5% peak)
- 5x speedup in time to solution due to kernel generation and shared memory
- 8x speedup as we can strong-scale to more cores
- Efficiency decrease caused by SuperMUC‘s island concept



- 60,000 cells per core, 21GB out of 32 GB node memory to show the limits of the code
- 6th order (4.5 trillion unknowns) using 30% of SuperMUC's memory
- 1.4 PF, more than 30% of peak performance

PRACE Award 2014, Gordon Bell finalist 2014

Latest Developments

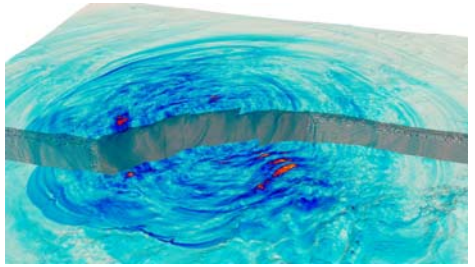


Optimization of matrix kernels (C. Uphoff)

- Implementation based on small (sparse) matrix multiplications
- Derive effective sparsity patterns using graph representation
- Generate block-sparse patterns
- Use efficient library for small matrix multiplications: libxsmm

Cluster-based local time stepping

- ADER allows local time stepping: each element may advance at an individual time step – however: not efficient/scalable
- Define „cluster“ as all elements with a time step in an interval – leads to **non-contiguous clusters;** perform cluster-based local time stepping
- SuperMUC phase 2 (3072 Haswell nodes): Wave propagation in volcano Mount Merapi – 100 Mio elements ➔ less performance (1.4 PF compared to 2.0 for GTS), but factor 4-5 reduction in time-to-solution
- Hazel Hen (2000 nodes): dynamic rupture, Landers earthquake, 191 M elements, 100 B unknowns ➔ 0.75 PF or 38% peak



Contents

Preface

Algorithms & Performance in CSE & HPC

Some examples – or best practices, hopefully

- Eigensolvers / Material Science / ELPA @ FHI-aims
- Block-adaptivity / Plasma Physics / GENE
- Data structures & more / Chemical Engineering / mardyn
- Complete re-engineering / Geophysics / SeisSol

Concluding remarks

Concluding Remarks

Some remarks on “Where does the HP in HPC come from?”

Yep, there was a strong bias towards the computation side (thus, underlining the need for convergence – what is the gain of 100% efficiency if you work in the wrong direction …?)

Nevertheless: algorithms are a common feature, algorithm & performance engineering can be links

Lessons learned here are valuable there

Thanks for your attention!

… looking forward to a great workshop and to discussions on and ideas for our upcoming Long Program

“Science @ Extreme Scales – Where Big Data Meets Large-Scale Computing” …