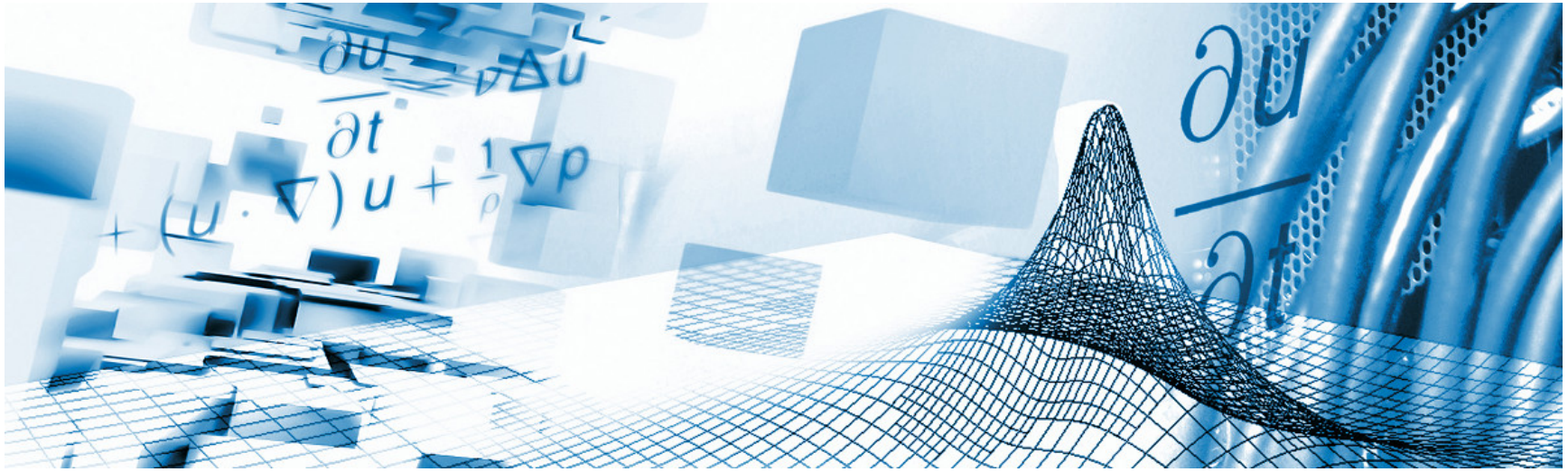
2015-09-14

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

Algorithms, System and Data Centre Optimisation for Energy Efficient HPC

Vincent Heuveline
URZ Computing Centre of Heidelberg University
EMCL Engineering Mathematics and Computing Lab



Energy Efficient High Performance Computing

1. Data centre point of view – HPC installation at URZ

   Computing Centre of Heidelberg University

2. Numerics point of view – current activities at EMCL

   Engineering Mathematics and Computing Lab at
   Interdisciplinary Centre for Scientific Computing (IWR)

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Energy Efficient HPC

1. Data centre point of view – HPC installation at URZ

Energy consumption in data centres

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

– Energy consumption for IT and cooling
  - Servers and network components
  - Chillers, computer room air conditioners – cold air, cold water
  - Power supplies including uninterruptible power supplies (UPS)
  - Fans

– Other energy consumption
  - Heating
  - Air conditioning for offices
  - Warm water
  - Other building equipment, e.g. light

Emergency power generator at URZ

Energy consumption in data centres

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

– Measure for efficiency of energy usage

Power Usage Effectiveness – PUE (The Green Grid, 2007)

$$\mathrm{PUE} = \frac{\text{Total energy data centre (kWh)}}{\text{IT energy (kWh)}}$$

Ideal: PUE = 1.0
Usual: PUE ~ 1.5 (good) ... 2.5 (bad)

SuperMUC @ Leibniz Computing Centre, Munich: PUE < 1.1

Special situation in HPC centres

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

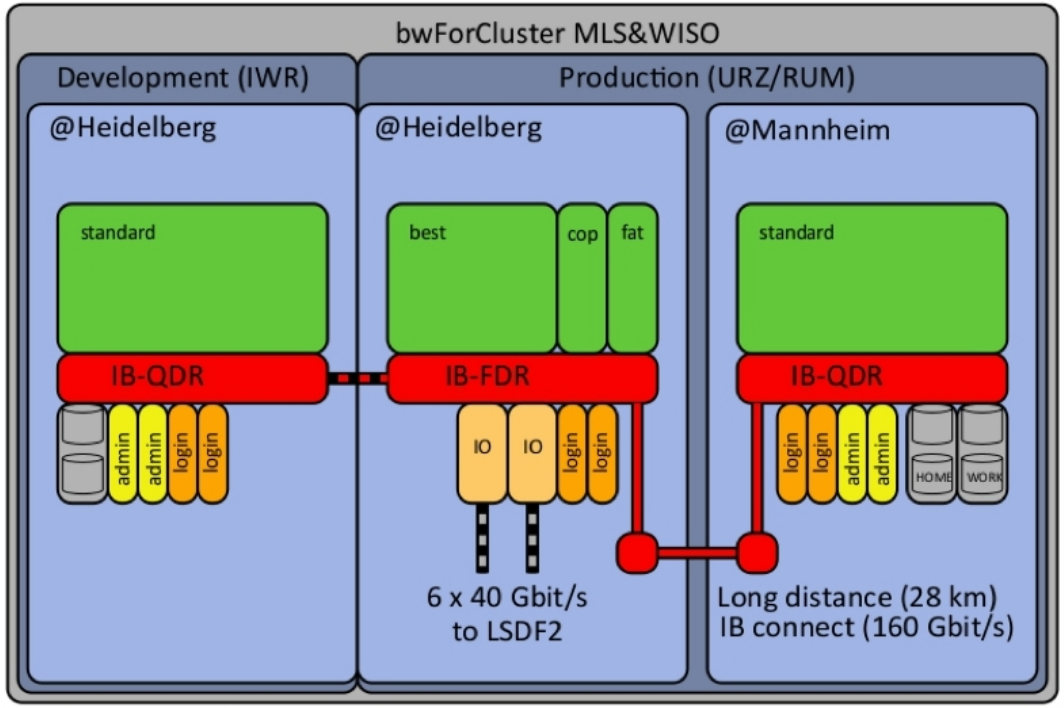
- High packing and energy density
  - 30 kW per rack already operational
  - Envisage 100 kW and more per rack
- High load on servers
  - Want to operate at full capacity
- Often homogeneous infrastructure
  - Dedicated cooling circuit for one machine reasonable
- Reuse of waste heat possible, but not constantly available
  - Unplanned (crash) or planned (maintenance) shutdowns reduce availability

HPC installation at URZ

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

– HPC cluster for research in Baden-Württemberg
  Joint project of Heidelberg and Mannheim Universities

– bwForCluster MLS & WISO
  - Molecular Life Science
  - Economic and Social Science
  - Scientific Computing



HPC installation at URZ

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

- total of ~500 compute nodes based on Intel Haswell
- total of ~9,000 CPU cores
- includes co-processor nodes based on Intel Xeon Phi
- IT power ~200 kW

Server racks in computer room at URZ

HPC installation at URZ

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

– example for 200 kW use

  200 kW * 24 h/day * 365 day/year * 20 Ct/kWh ≈ 350,000 EUR/year

– traditional cold air cooling: PUE ≈ 2

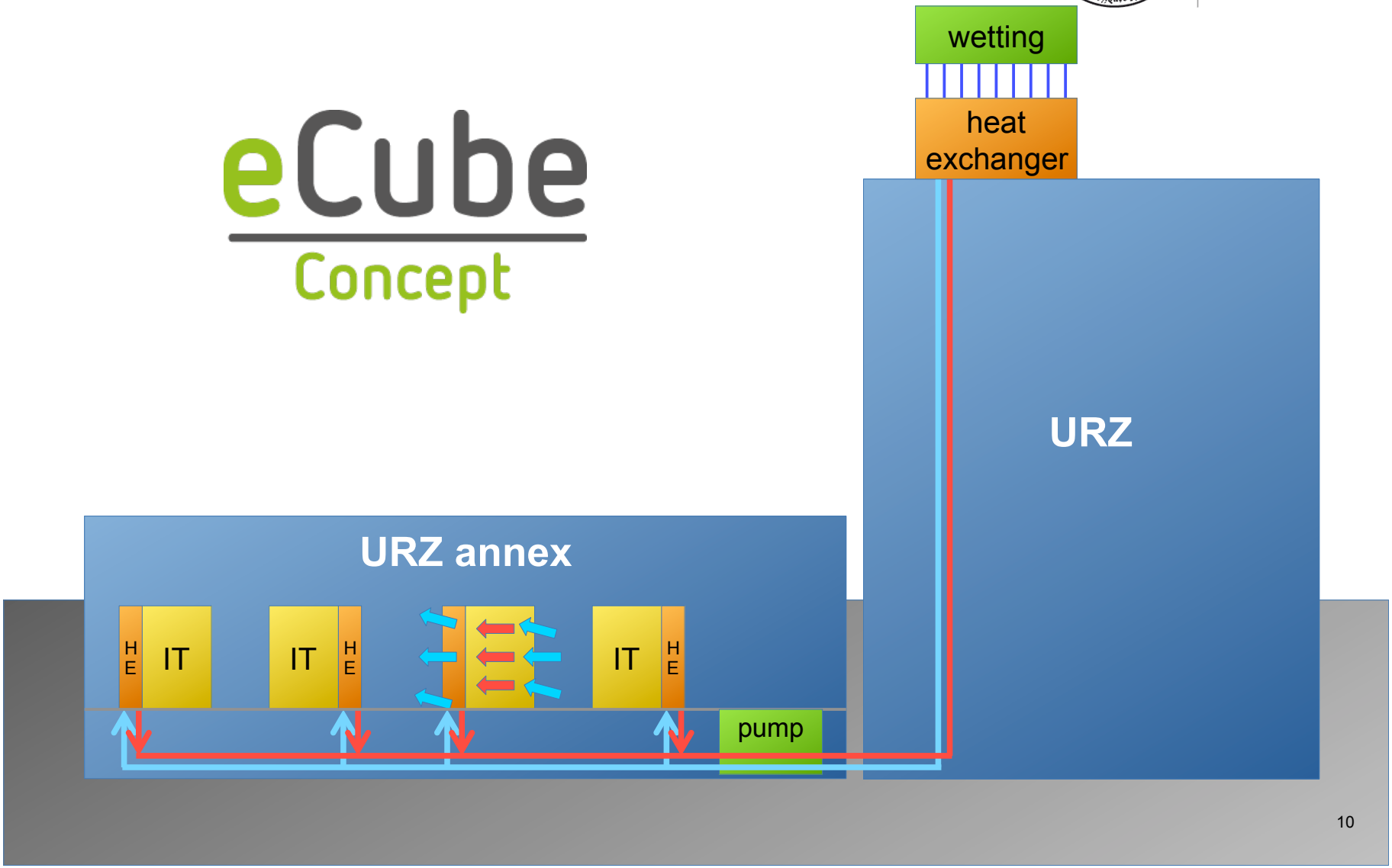
  ► additional 0.35 Mio. EUR/year only for cooling!

– URZ uses passive cooling technology
  - outdoor environmental temperature used to cool IT
  - additional cooling by evaporation in hot summer days
  - goal: achieve PUE ~ 1.1

HPC installation at URZ

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

eCube Concept



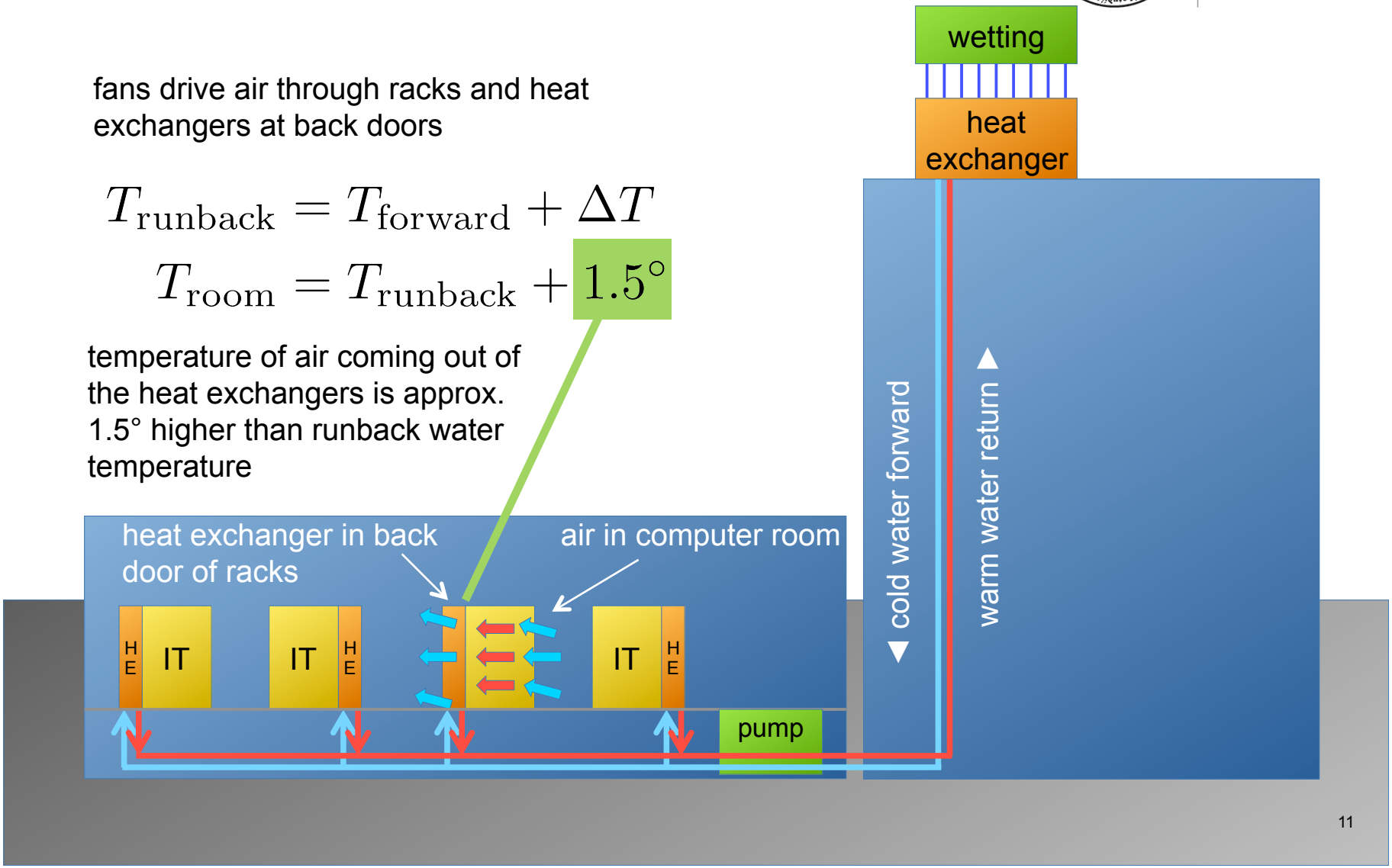
HPC installation at URZ

fans drive air through racks and heat exchangers at back doors

$$T_{\text{runback}} = T_{\text{forward}} + \Delta T$$

$$T_{\text{room}} = T_{\text{runback}} + 1.5°$$

temperature of air coming out of the heat exchangers is approx. 1.5° higher than runback water temperature



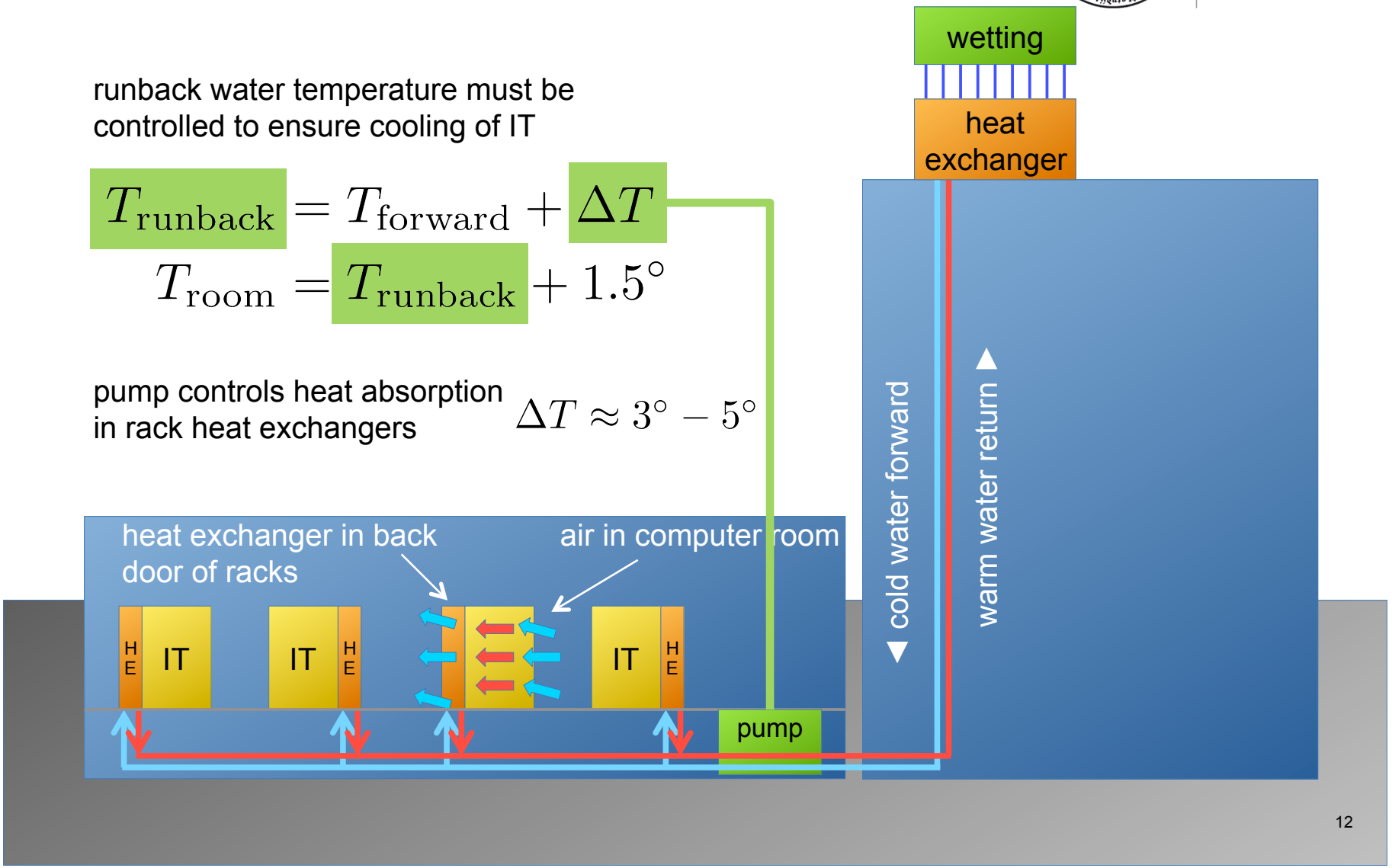
HPC installation at URZ

runback water temperature must be controlled to ensure cooling of IT

$$T_{\text{runback}} = T_{\text{forward}} + \Delta T$$

$$T_{\text{room}} = T_{\text{runback}} + 1.5°$$

pump controls heat absorption in rack heat exchangers $\Delta T \approx 3° - 5°$



HPC installation at URZ

Heat exchanger and wetting on roof of URZ

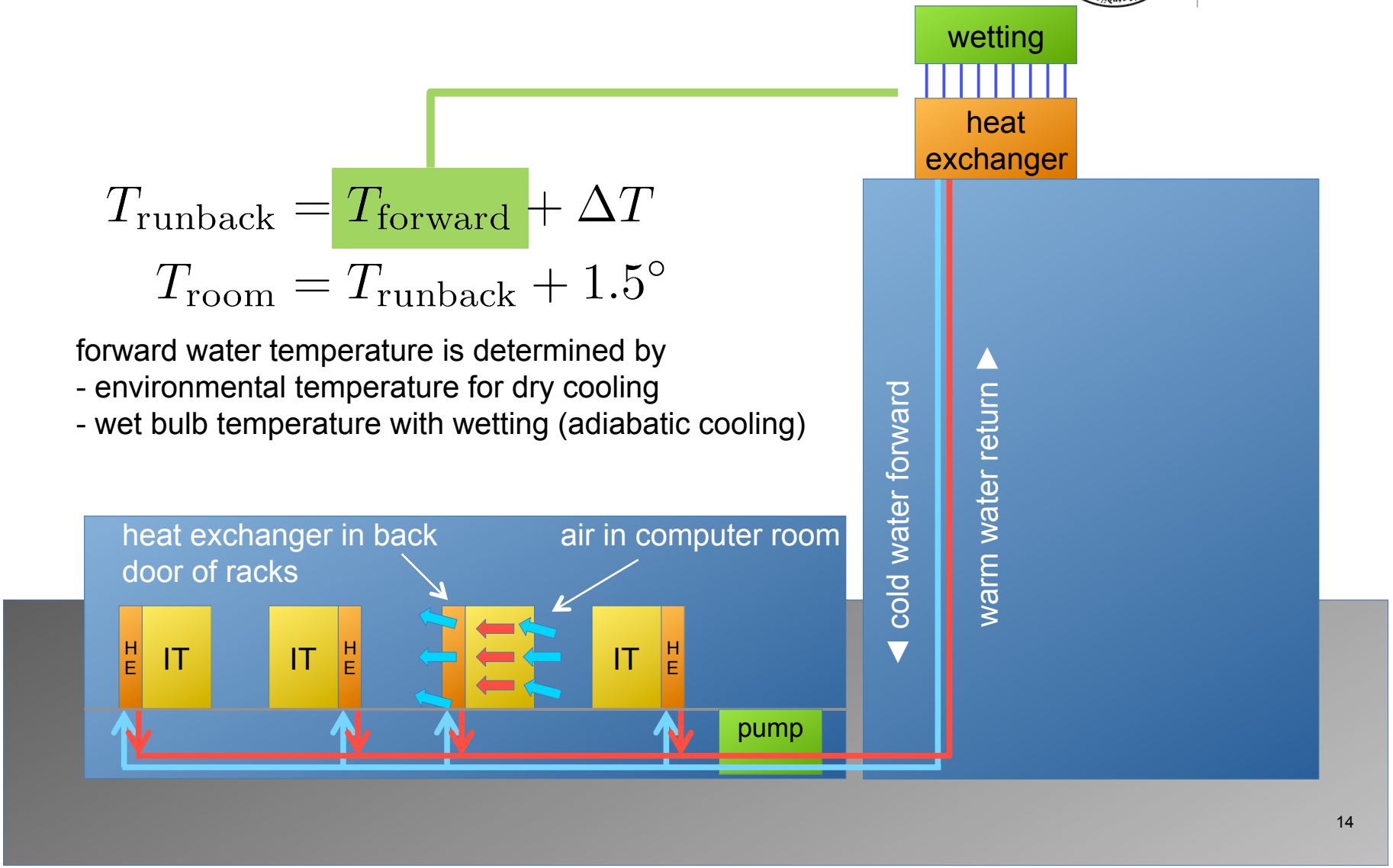
HPC installation at URZ

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

$$T_{\text{runback}} = T_{\text{forward}} + \Delta T$$

$$T_{\text{room}} = T_{\text{runback}} + 1.5°$$

forward water temperature is determined by
- environmental temperature for dry cooling
- wet bulb temperature with wetting (adiabatic cooling)

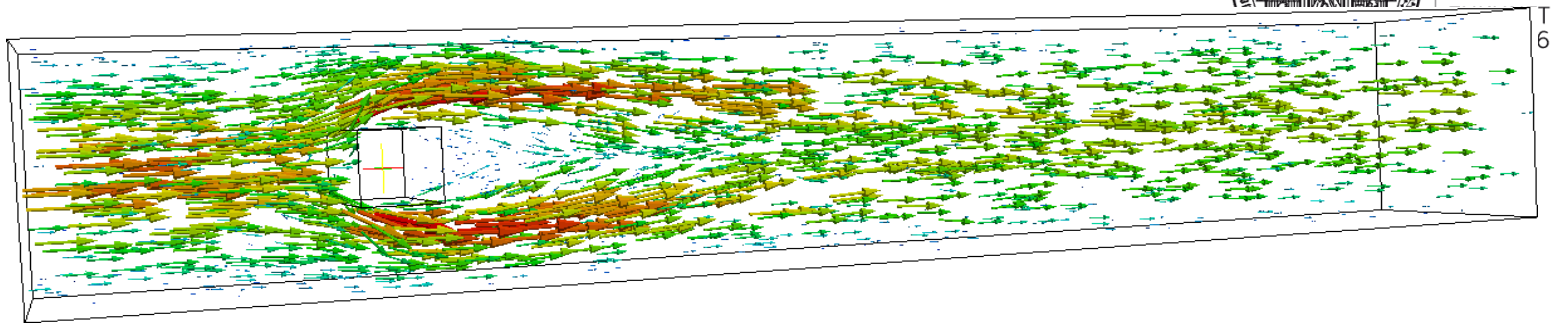


UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Energy Efficient HPC

2. Numerics point of view – current activities at EMCL

Numerical Simulation



Model
- Partial differential equations

$$\partial_t u + (u \cdot \nabla)u - \nu \Delta u + \frac{1}{\rho} \nabla p = 0$$

Discretisation

$$u = \sum_{i=1}^{n} x_i \phi_i$$

Algebraic system
- Linear
- Nonlinear

Iterative methods

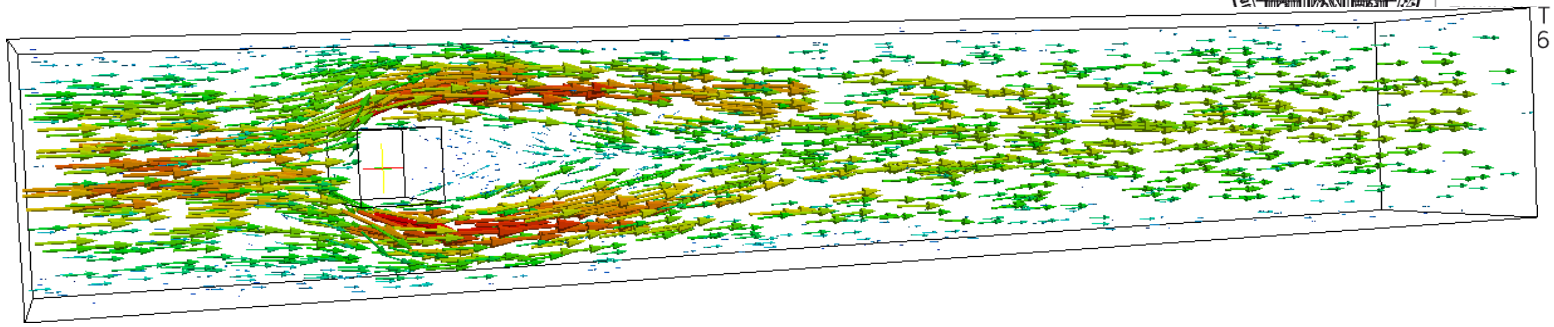
$$x^{(0)} \rightarrow x^{(1)} \rightarrow x^{(2)} \rightarrow \ldots \rightarrow x$$

Krylov subspace
- Conjugate Gradient (CG)
- General Minimal Residual (GMRES)

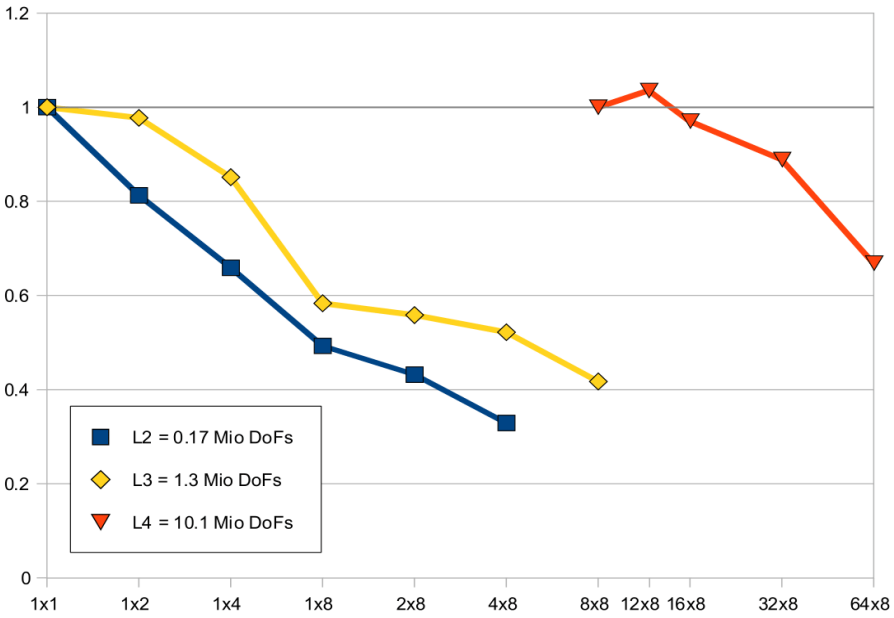
Multilevel
- Algebraic multigrid
- Geometric multigrid

Fluid Flow Benchmark – Scalability Study

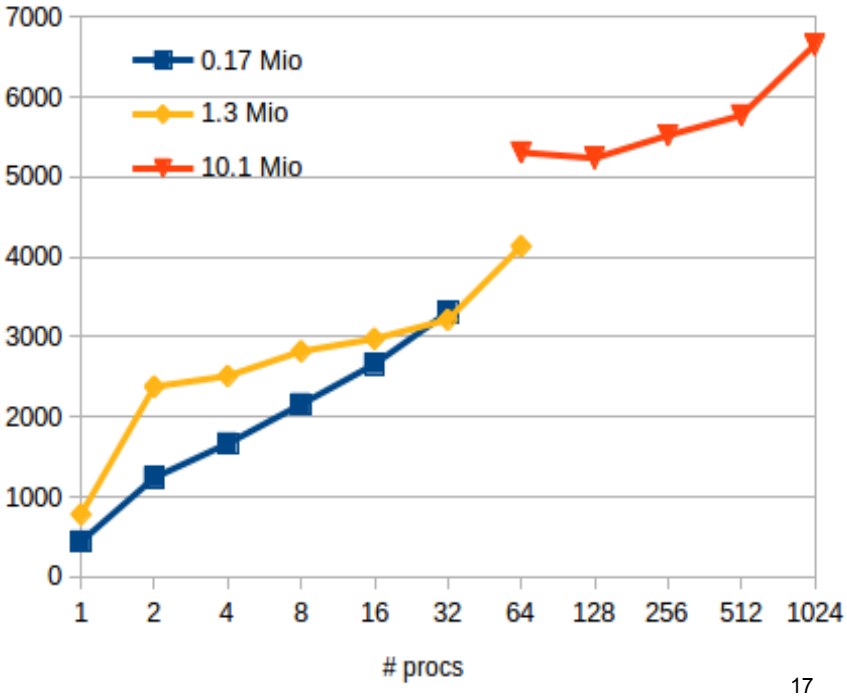
UNIVERSITÄT HEIDELBERG



parallel efficiency



iter preconditioned GMRES

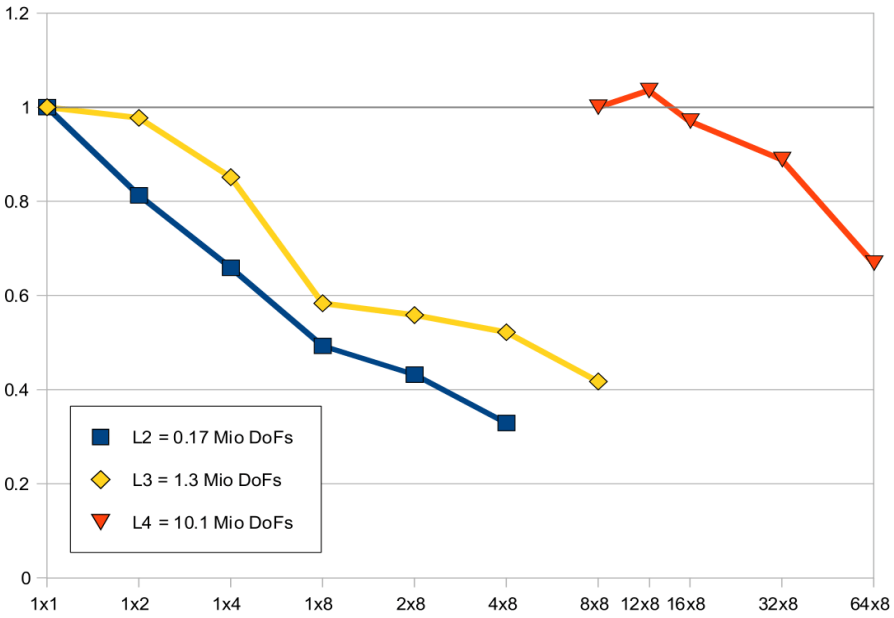


Fluid Flow Benchmark – Scalability Study

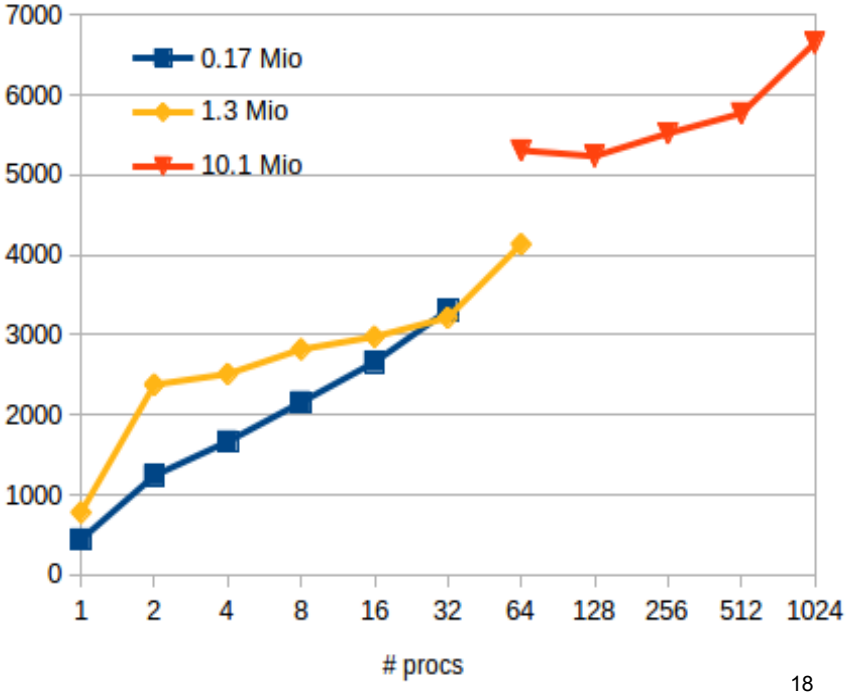
UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

- poor parallel efficiency due to higher number of iterations
- preconditioner is weaker for higher number of MPI processes

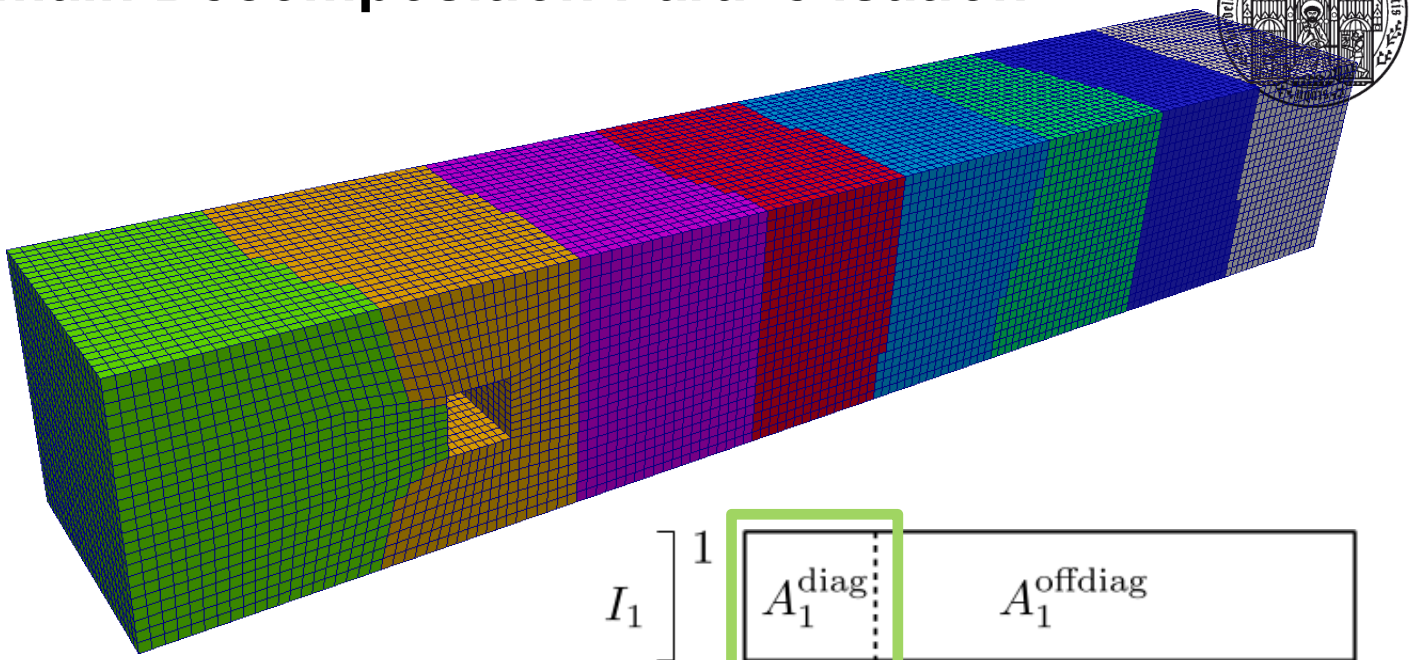
parallel efficiency



iter preconditioned GMRES



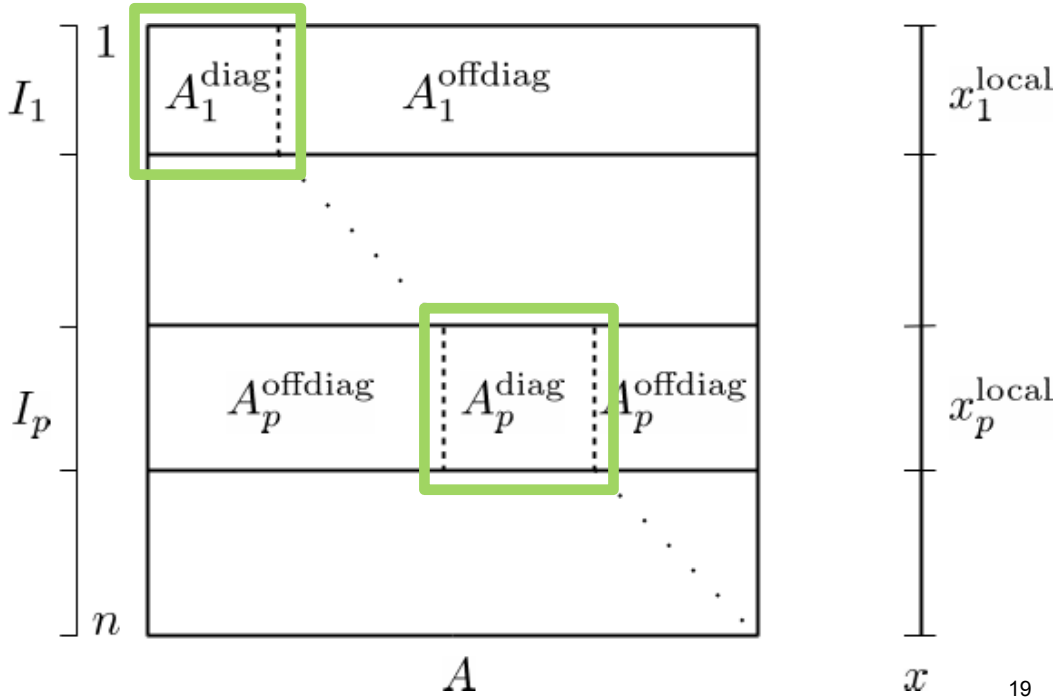
Domain Decomposition Parallelisation



UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

-preconditioner acts only on diagonal blocks

- more processes imply smaller blocks



Bottleneck in Numerical Algorithms

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Theoretical system performance hard to achieve

Challenge to transfer computing power into simulation performance

Parallelization can affect mathematical properties of numerical algorithms

Scalability may suffer from synchronizations

No usage of energy-saving techniques

Iterative Refinement and Multilevel Methods

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

Iterative refinement

Repeat:

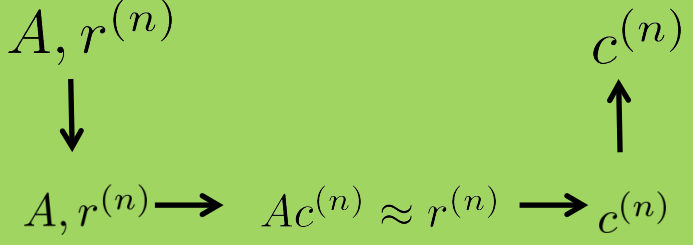
1. Compute residual
   $r^{(n)} = b - Ax^{(n)}$

2. Error correction equation
   $Ac^{(n)} \approx r^{(n)}$

3. Update
   $x^{(n+1)} = x^{(n)} + c^{(n)}$

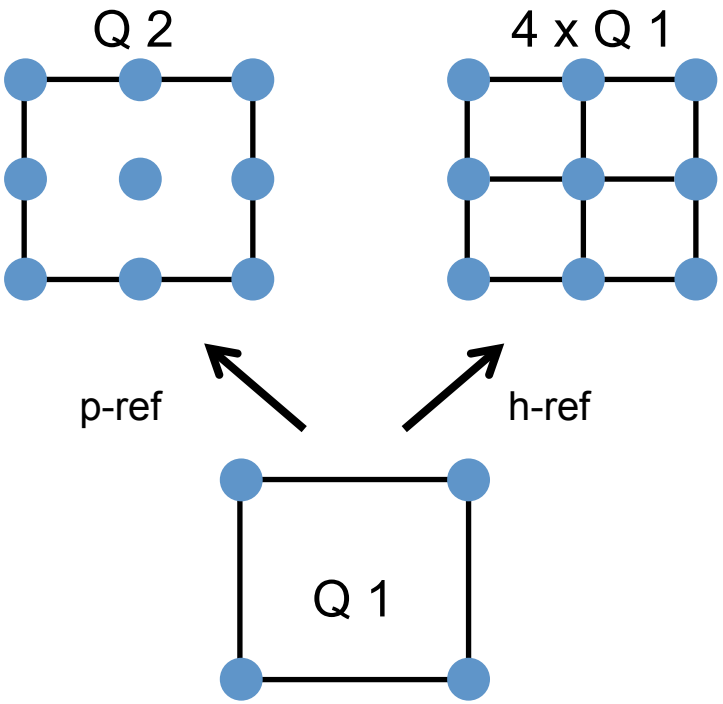
Multilevel methods

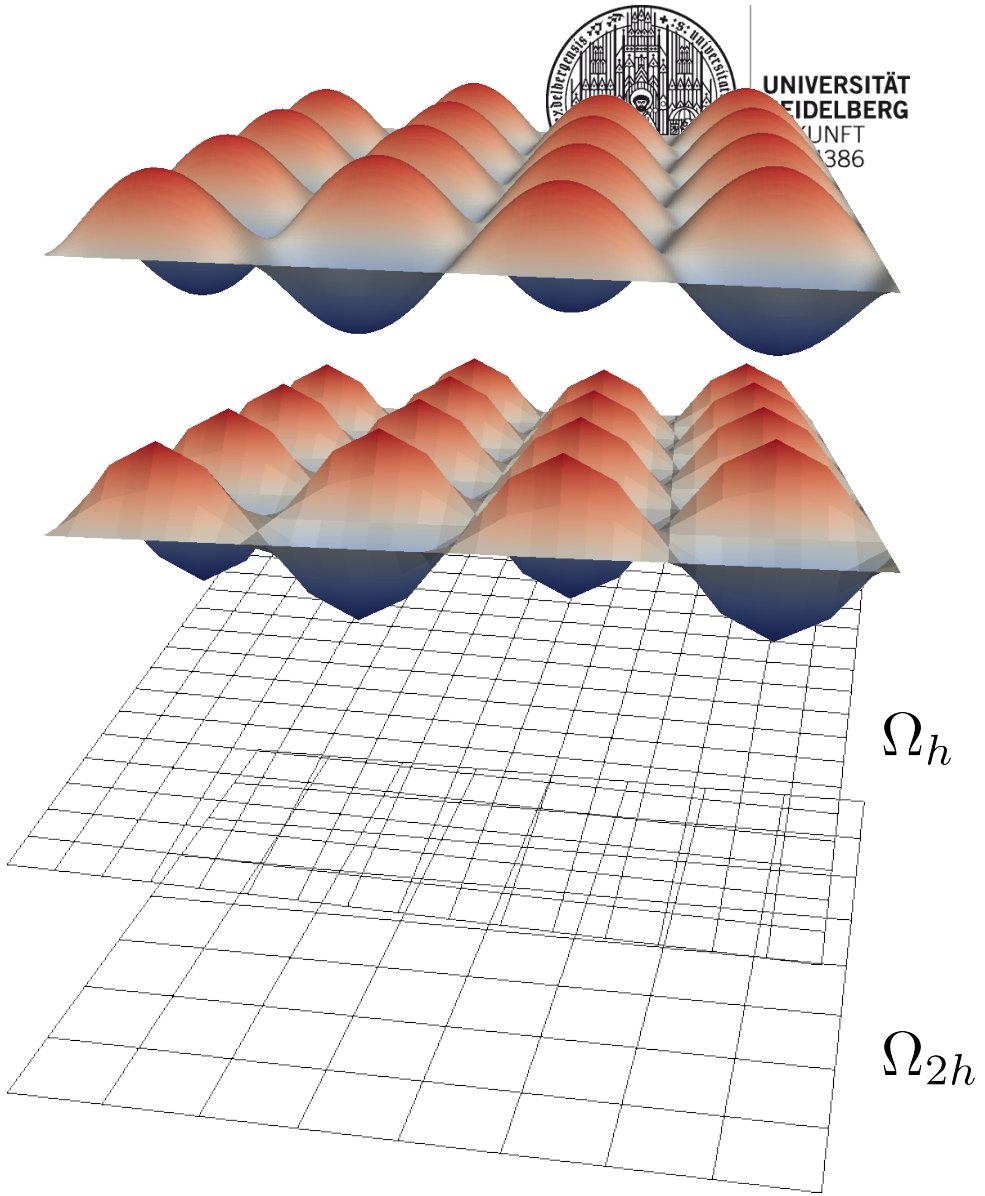
Solve error correction equation only in a subspace



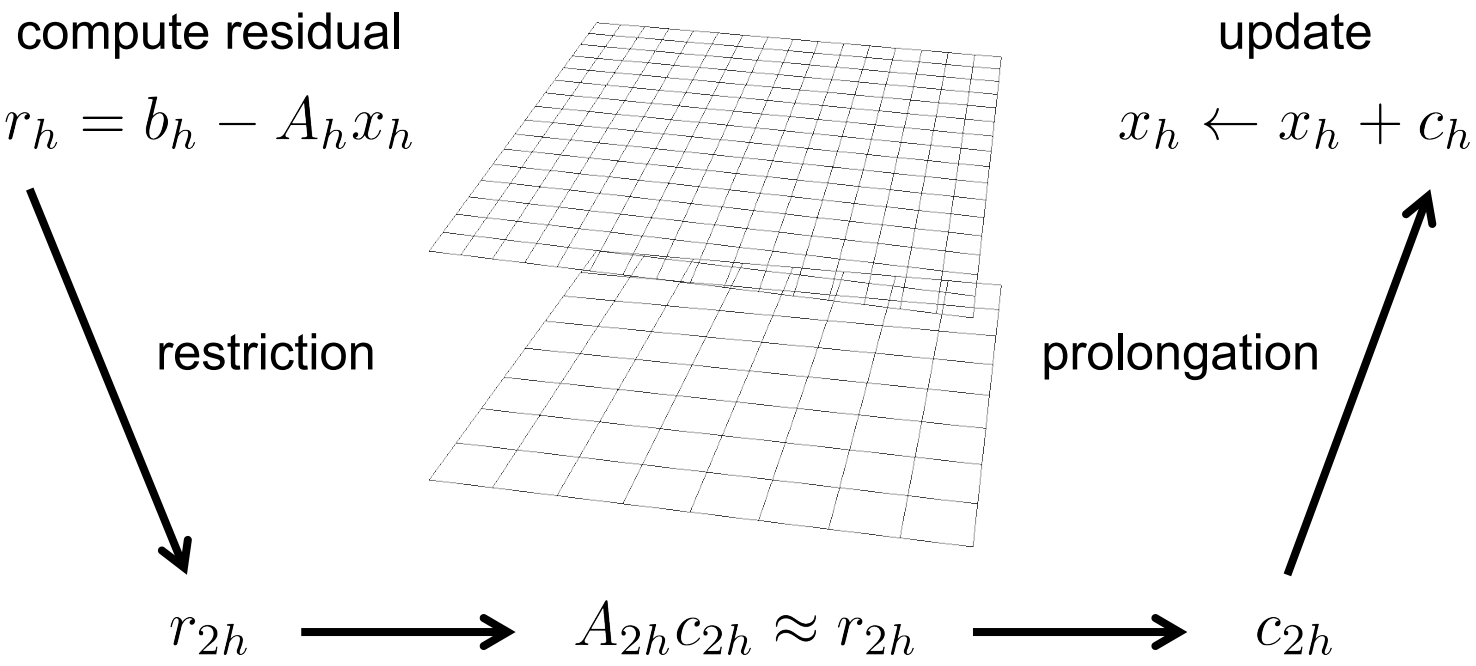
Geometric Multigrid

Finite element
discretisation levels





Geometric Multigrid – Grid Transfer



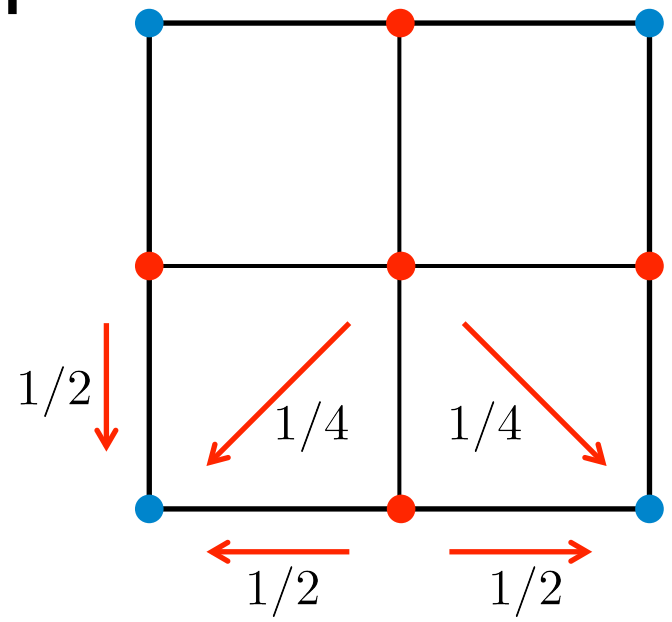
Geometric Multigrid – Restriction

compute residual

$$r_h = b_h - A_h x_h$$

restriction

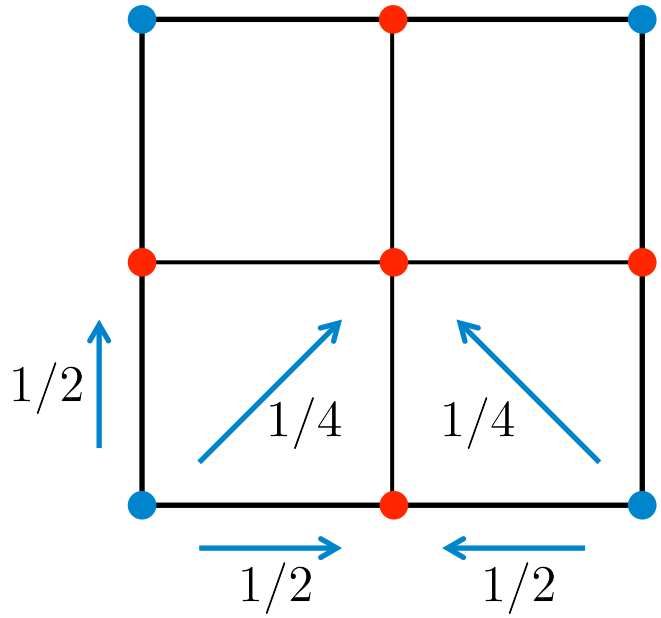
$$r_{2h} = R_{2h}^h r_h$$



$$R_{2h}^h = \begin{bmatrix} 1 & 1/2 & 0 & 1/2 & 1/4 & 0 & \dots \\ 0 & 1/2 & 1 & 0 & 1/4 & 1/2 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

restriction matrix

Geometric Multigrid – Prolongation



prolongation matrix
linear interpolation

$$P_{2h}^{h} = \left[R_{2h}^{h} \right]^{\top}$$

update

$$x_h \leftarrow x_h + P_{2h}^{h} c_{2h}$$

prolongation

c_{2h}

High Oscillations

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

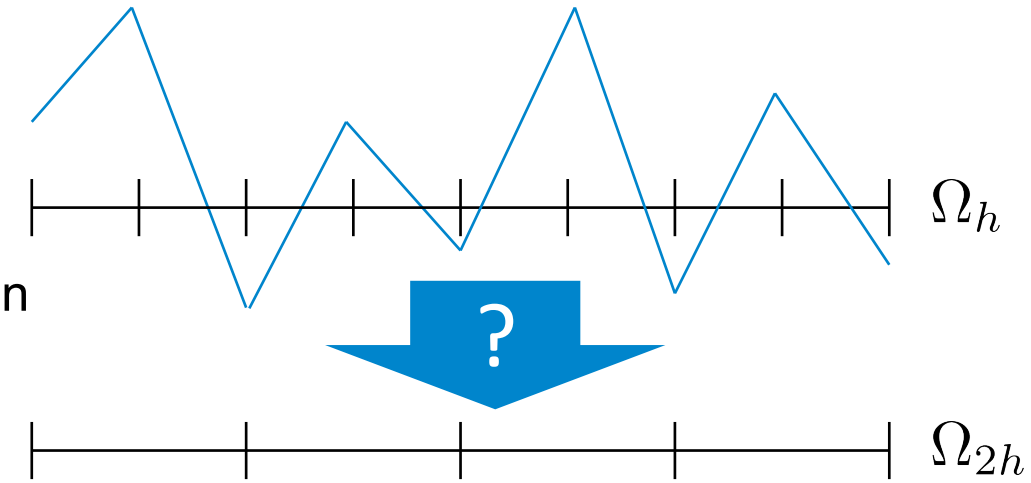
$$A_h x_h = b_h \qquad \text{linear system}$$

$$r_h^{(n)} := b_h - A_h x_h^{(n)}$$

$$e_h^{(n)} := x_h - x_h^{(n)}$$

$$A_h e_h^{(n)} = r_h^{(n)} \qquad \text{error equation}$$

$$A_{2h} c_{2h}^{(n)} \approx r_{2h}^{(n)} \qquad \text{correction}$$



linear iterative schemes
- Jacobi
- Gauss-Seidel

application of the scheme to the current approximation has a smoothing action on the unknown error

$$\tilde{x}^{(n)} = Mx^{(n)} + d$$

$$= Mx - Me^{(n)} + d$$

$$= x - Me^{(n)}$$

$$\Longrightarrow \tilde{e}^{(n)} = Me^{(n)}$$

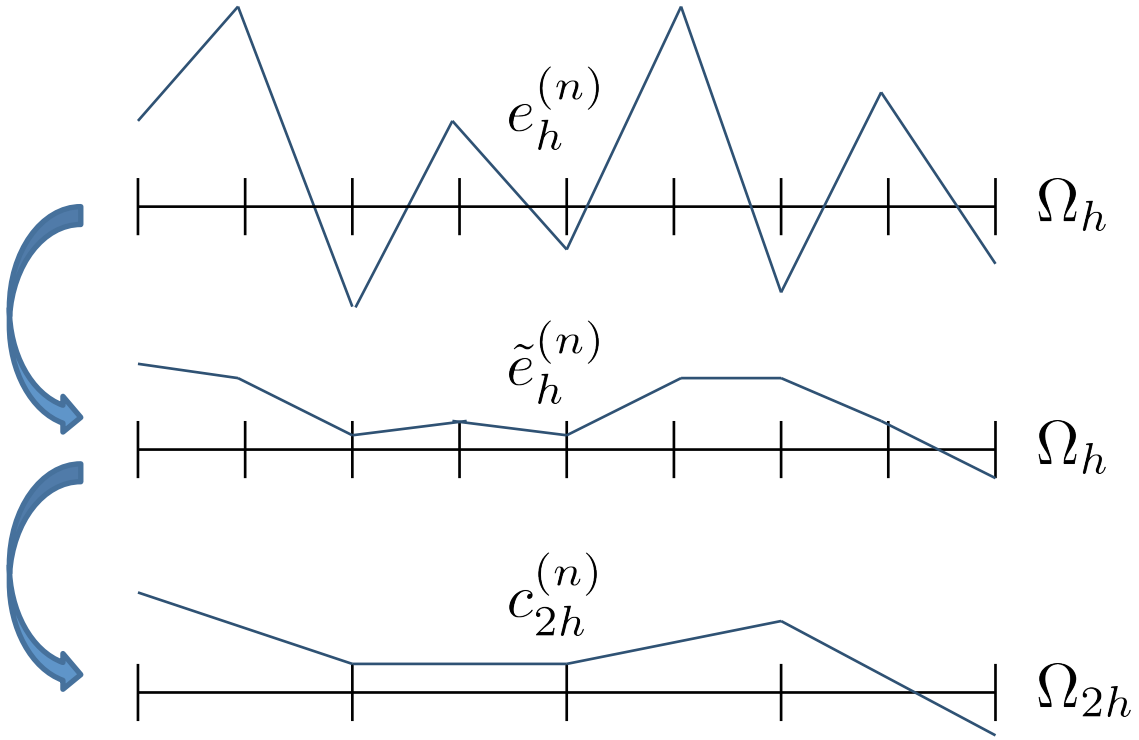
Smoothing

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

$$S_h(x_h^{(n)}) \to \tilde{x}_h^{(n)} \qquad \Longrightarrow \qquad e_h^{(n)} \to \tilde{e}_h^{(n)}$$

smoother damps high error oscillations

approximate on coarser grid

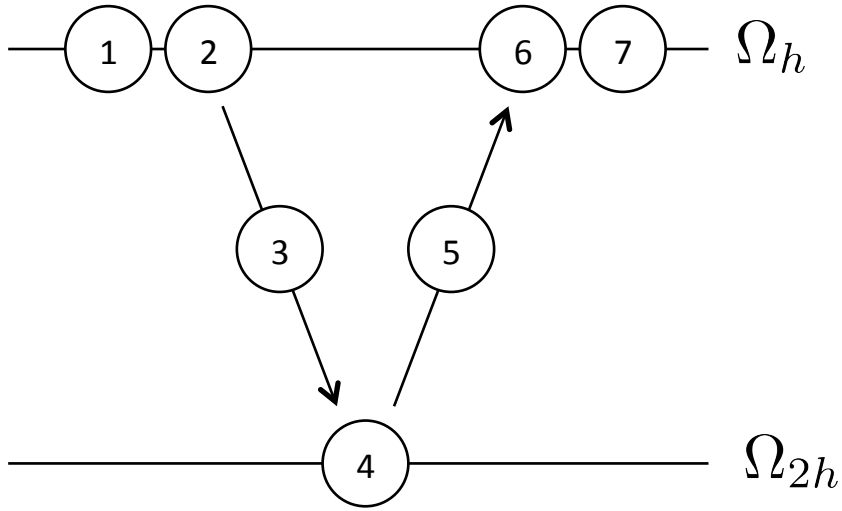


Coarse Grid Correction

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

$$A_h x_h = b_h$$

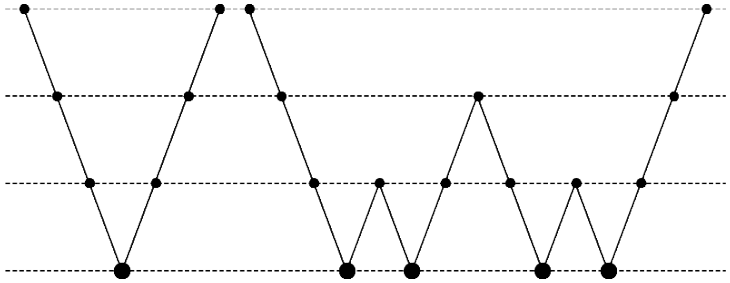
1. $x_h \longleftarrow S_h(x_h)$ pre-smoothing
2. $r_h = b_h - A_h x_h$
3. $b_{2h} = R_{2h}^h r_h$
4. $A_{2h} x_{2h} \approx b_{2h}$
5. $c_h = P_{2h}^h x_{2h}$
6. $x_h \longleftarrow x_h + c_h$
7. $x_h \longleftarrow S_h(x_h)$ post-smoothing



Multigrid Cycle

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

1: **if** $h = H$ **then**
2: $\quad x_h \longleftarrow A_h^{-1} b_h$
3: **else**
4: $\quad x_h \longleftarrow S_h(x_h)$ (pre-smoothing)
5: $\quad r_h \longleftarrow b_h - A_h x_h$
6: $\quad b_{2h} \longleftarrow R_{2h}^h r_h$
7: $\quad x_{2h} \longleftarrow 0$
8: $\quad$ **for** $k = 1, 2, ..., \gamma$ **do**
9: $\quad\quad$ Cycle$(A_{2h}, x_{2h}, b_{2h}, \gamma)$
10: $\quad$ **end for**
11: $\quad c_h \longleftarrow P_{2h}^h x_{2h}$
12: $\quad x_h \longleftarrow x_h + c_h$
13: $\quad x_h \longleftarrow S_h(x_h)$ (post-smoothing)
14: **end if**



UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Energy Efficient HPC

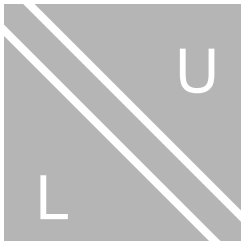
2. Numerics point of view – current activities at EMCL

► 2.1 Modifying the smoother to avoid synchronization for exploiting massively parallel hardware like GPUs

Smoother – Jacobi and Asynchronous Schemes

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

Matrix splitting $A = L + D + U =$



Jacobi iteration $x^{k+1} = D^{-1}\left[b - (L+U)x^k\right]$

Component wise $x_i^{k+1} = \frac{1}{a_{ii}}\left[b_i - \sum_{j \neq i} a_{ij} x_j^k\right]$ strict synchronisation

Basic asynchronous scheme

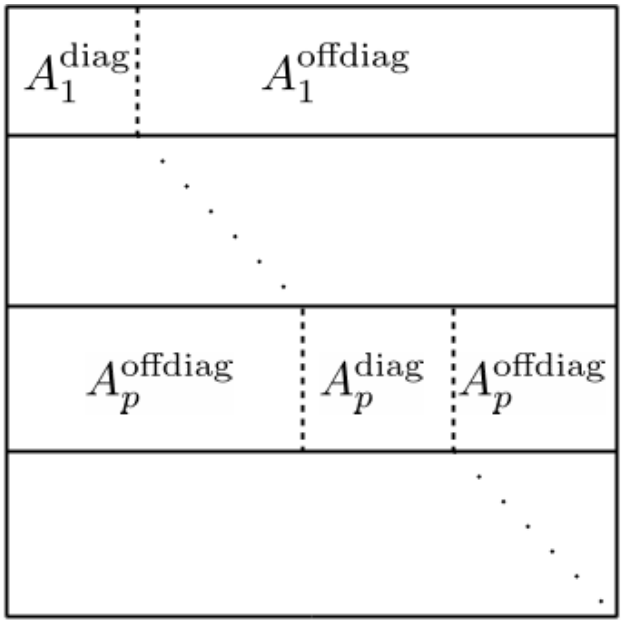
allow older values through shift function
allow arbitrary order of component updates

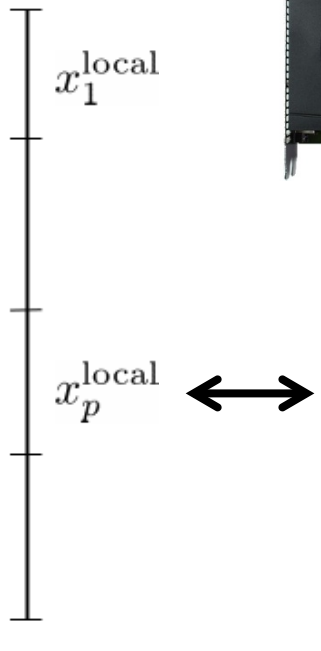
$$x_i^{k+1} = \begin{cases} \frac{1}{a_{ii}}\left[b_i - \sum_{j \neq i} a_{ij} x_j^{k-s(j)}\right] & \text{if } i = u(k) \\ x_i^k & \text{if } i \neq u(k) \end{cases}$$

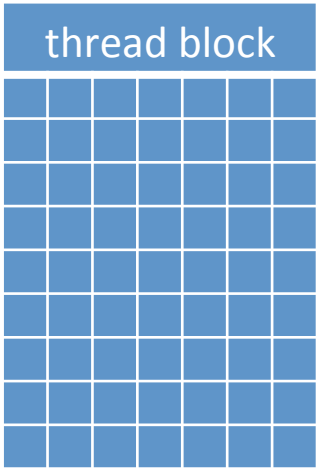
Block-Asynchronous Iteration on GPU

UNIVERSITÄT HEIDELBERG

	SMX = streaming multiprocessor	cores per SMX	max. thread blocks per SMX	max. threads per thread block
Tesla K40	15	192	16	1024



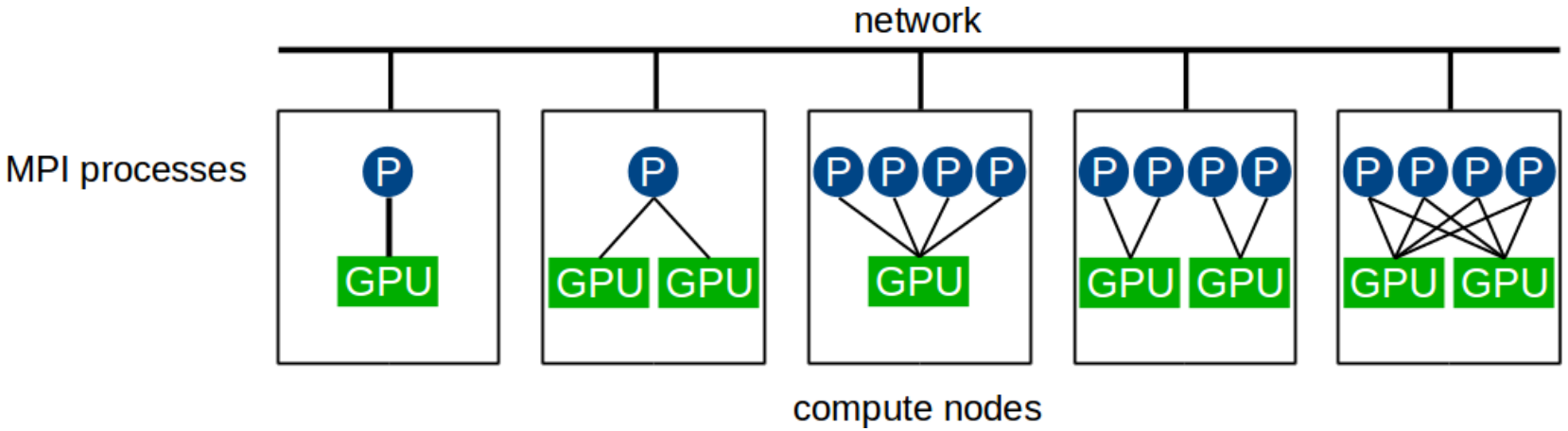




block update $$x_p^{\text{local}} \leftarrow D_p^{-1}\left[b_p^{\text{local}} - A_p^{\text{diag}} x_p^{\text{local}} - A_p^{\text{offdiag}} x_p^{\text{non-local}}\right]$$

Hybrid Parallel Implementation

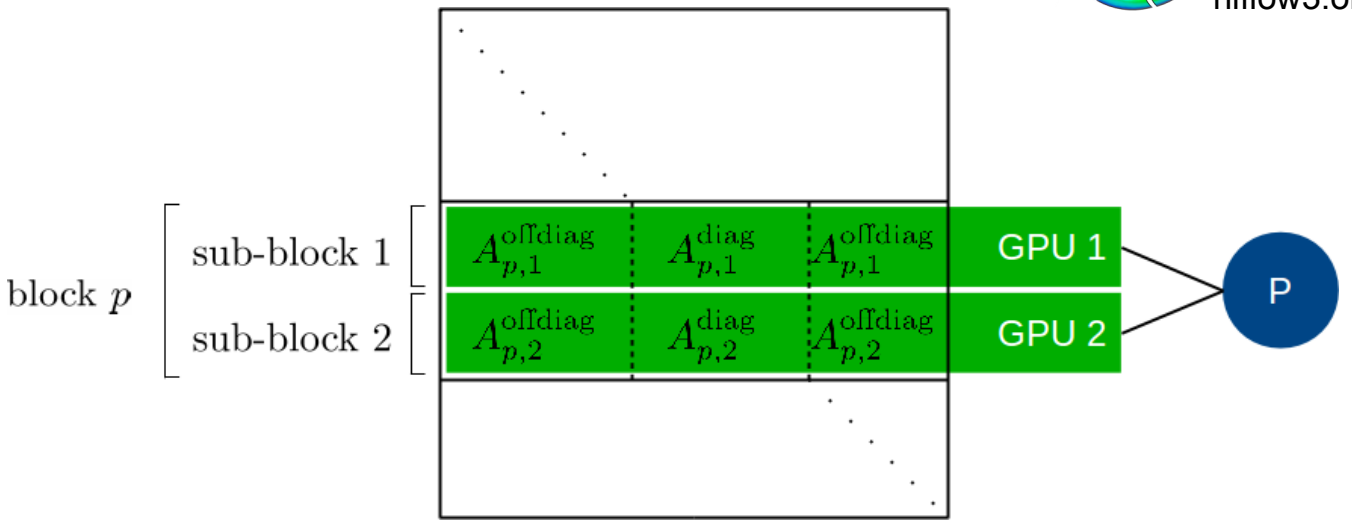
- MPI, OpenMP & CUDA



- from Kepler on: HyperQ (32 concurrent streams)
- must use Multi-Process Service to share one GPU

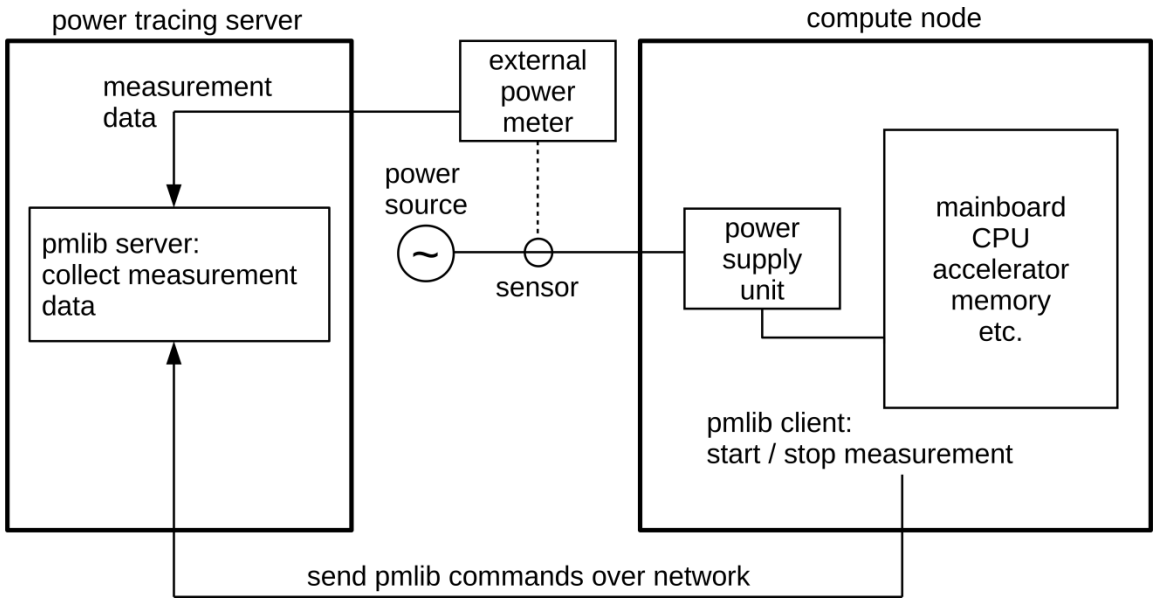
Hybrid Parallel Implementation

- multi-GPU



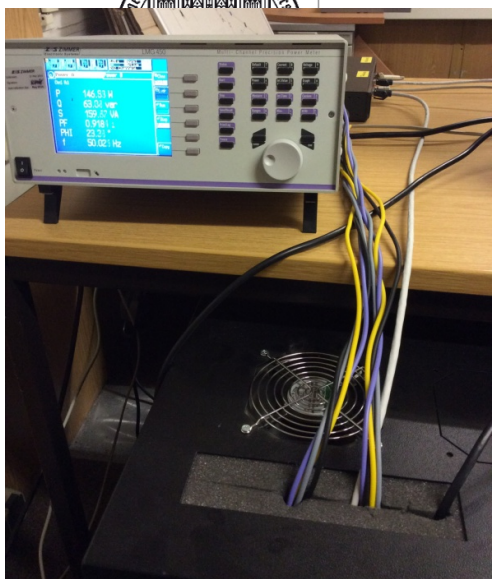
- cannot use Multi-Process Service

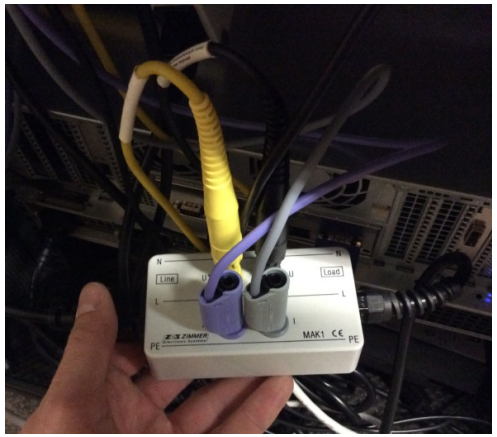
Energy Measurement



pmlib

- source code instrumented with pmlib API calls
- application becomes pmlib client, starts and stops measurements
- extra tracing server controls power meter and collects measurement data
- no perturbation of system under investigation





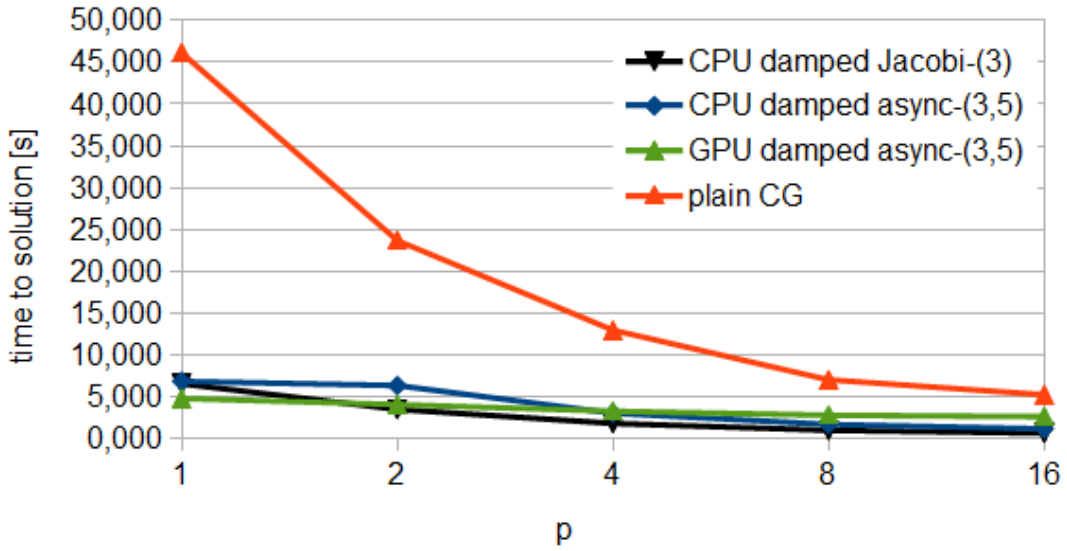
ZES LMG450
external power meter

Geometric Multigrid vs. Conjugate Gradient

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

Showcase to demonstrate the power of multigrid:

- tests up to 16 CPUs and 1 GPU

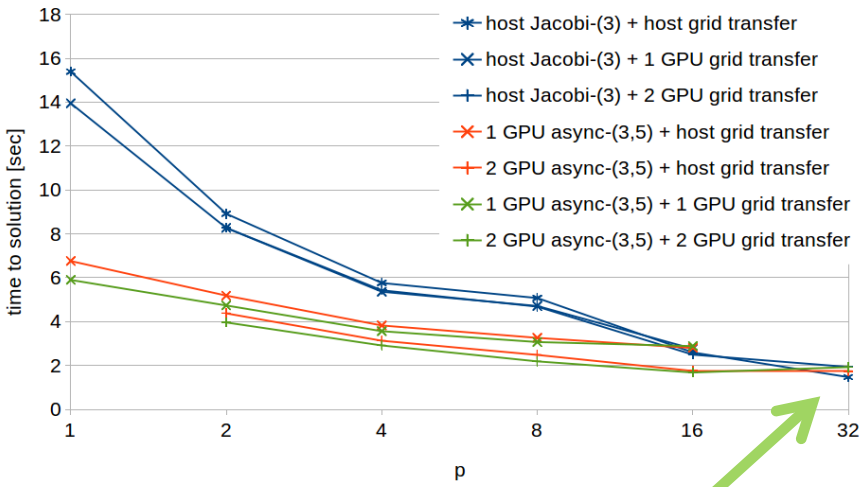


different variants of multigrid outperform standard CG solver

Results
1 to 32 MPI processes, up to 2 GPUs

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

time to solution [sec]

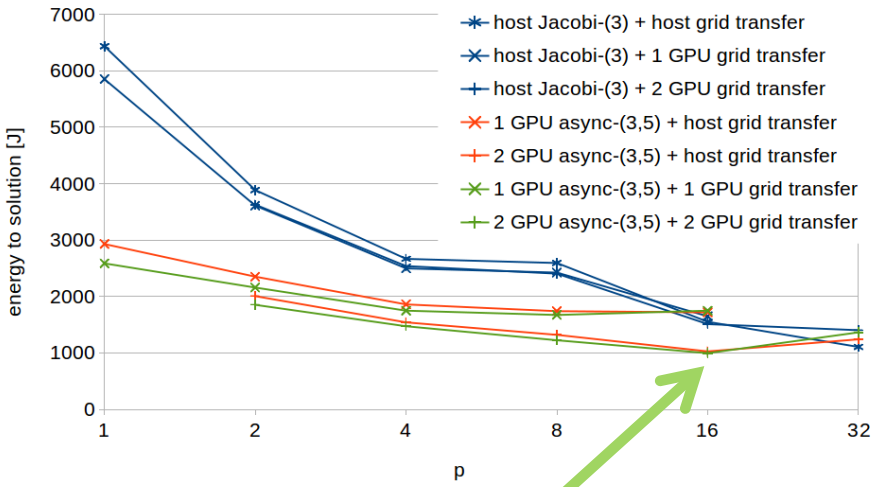


optimal time to solution with
p = 32 MPI processes, no GPU

► t = 1.469 sec, E = 1,106.315 J

saves 14% time over
best energy to solution

energy to solution [J]



optimal energy to solution with
p = 16 MPI processes, 2 GPUs

► t = 1.680 sec, E = 995.810 J

saves 10% energy over
best time to solution

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Energy Efficient HPC

2. Numerics point of view – current activities at EMCL

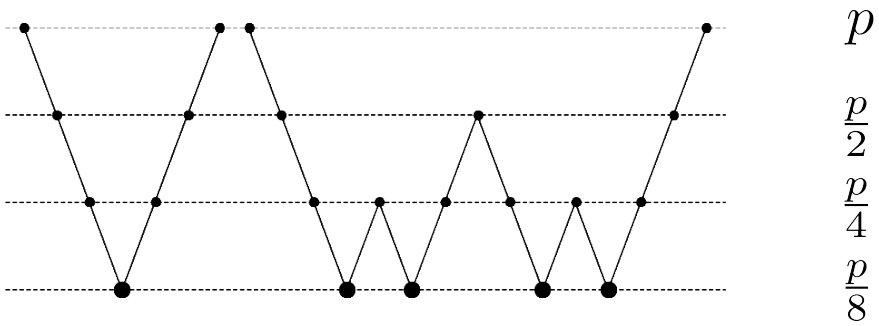
► 2.2 Adjusting hardware activity according to different problem sizes in the grid levels

Individual Decomposition on Grid Levels

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

Smaller problem size on coarser levels
- factor between levels ~ 2^d in d dimensions

Observation: scaling limited by smallest problem size
Idea: use less processes on coarser levels



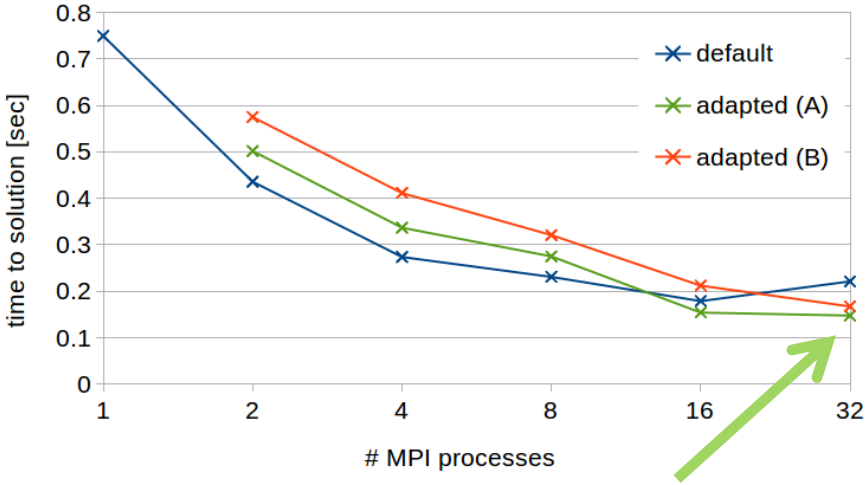
Hope:
- overall multigrid performance not impaired for large number of MPI processes
- energy savings possible due to partly deactivation of hardware

Results: Time and Energy to Solution

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

level	1	2	3	4	5
default	p	p	p	p	p
adapted (A)	p	p	$p/2$	$p/4$	$p/8$
adapted (B)	p	$p/2$	$p/4$	$p/8$	$p/16$

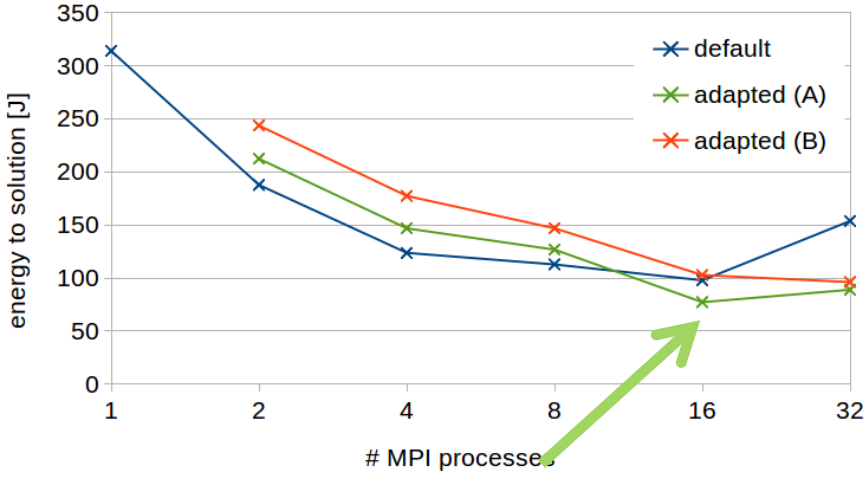
MPI processes per grid level



optimal time to solution
adapted (A) with p = 32

► t = 0.148 sec, E = 88.900 J

saves 13% time over best default



optimal energy to solution
adapted (A) with p = 16

► t = 0.154 sec, E = 77.194 J

saves 14% energy over best default

C-states and Intel RAPL

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

C-states

- CPU energy / activity states
- C0 = active, C1 = halt, etc.
- several further states, depending on architecture
- works by cutting clock signals, and reducing CPU voltage for deeper states

Intel Running Average Power Limit (RAPL)

- development of RAPL driven by thermal design power (TDP): guarantees proper operation of CPU if chassis and cooling system obey the TDP
- provides energy consumption information based on performance counters and power models
- can be used as power indicator

Results: C-states and Intel RAPL