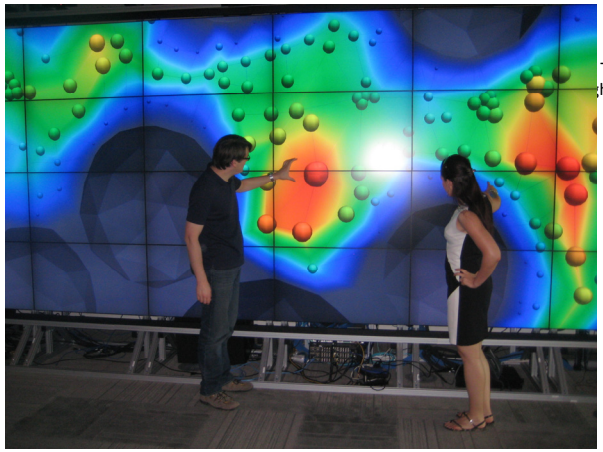
HPC for Environmental Simulations

PD Dr. rer. nat. habil. Ralf-Peter Mundani
Computation in Engineering / BGU
Scientific Computing in Computer Science / INF

18th International Symposium on Symbolic and Numerical Algorithms for Scientific Computing
September 24–27, 2016
Timisoara, Romania

Motivation

"High-performance computing must now assume a broader meaning, encompassing not only flops, but also the ability, for example, to efficiently manipulate vast and rapidly increasing quantities of both numerical and non-numerical data."



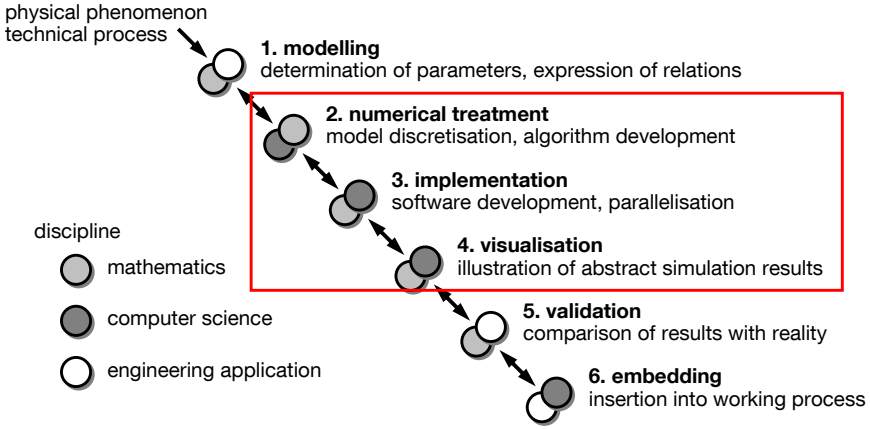
The White House.
h-performance-computing

KAUST

AESOP: 40× 46" NEC panels with total res. of 13,600 × 3,072 pixels (~42 MPixel)

Motivation

- simulation – from phenomena to prediction



Motivation

- why parallel programming and HPC?
  - complex problems (especially the so called *`grand challenges´*) demand for more computing power
    - climate or geophysics simulation (tsunami, e.g.)
    - structure or flow simulation (crash test, e.g.)
    - development systems (CAD, e.g.)
    - large data analysis (Large Hadron Collider at CERN, e.g.)
    - military applications (crypto analysis, e.g.)
    - …
  - performance increase due to
    - faster hardware, more memory *(`work harder´)*
    - more efficient algorithms, optimisation *(`work smarter´)*
    - parallel computing *(`get some help´)*

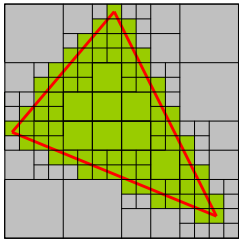
Motivation

- objectives (in case all resources would be available *N*-times)
  - *throughput*: compute *N* problems simultaneously
    - running *N* instances of a sequential program with different data sets (`*embarrassing parallelism*´); SETI@home, e.g.
  - *response time*: compute one problem at a fraction (1/*N*) of time
    - running one instance (i.e. *N* processes) of a parallel program for jointly solving a problem; finding prime numbers, e.g.
  - *problem size*: compute one problem with *N*-times larger data
    - running one instance (i.e. *N* processes) of a parallel program, using the sum of all local memories for computing larger problem sizes; iterative solution of SLE, e.g.

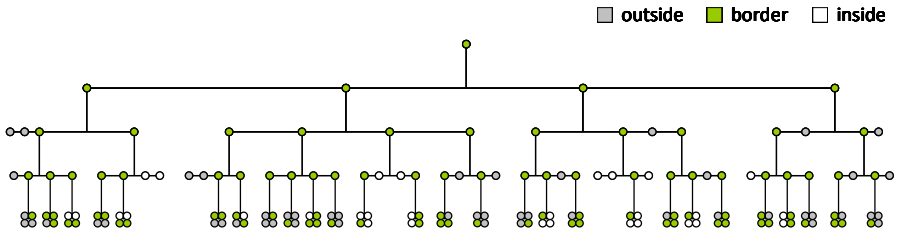
- **overview**
  - geometric and physical modelling
  - foundations / parallel architectures
  - multigrid methods
  - towards massive parallel HPC…
  - interactive visual data exploration

Geometric and Physical Modelling

- spacetrees
  - hierarchical data structure (cf. quadtrees in 2D and octrees in 3D)
  - built via recursive bi-section in every dimension
    ➔ 2^D children / node
  - reduced complexity, i.e. amount of voxels compared to equidistant discretisation $\mathrm{O}(N^3)$
    ➔ $\mathrm{O}(N)$ in 2D and $\mathrm{O}(N^2)$ in 3D on average

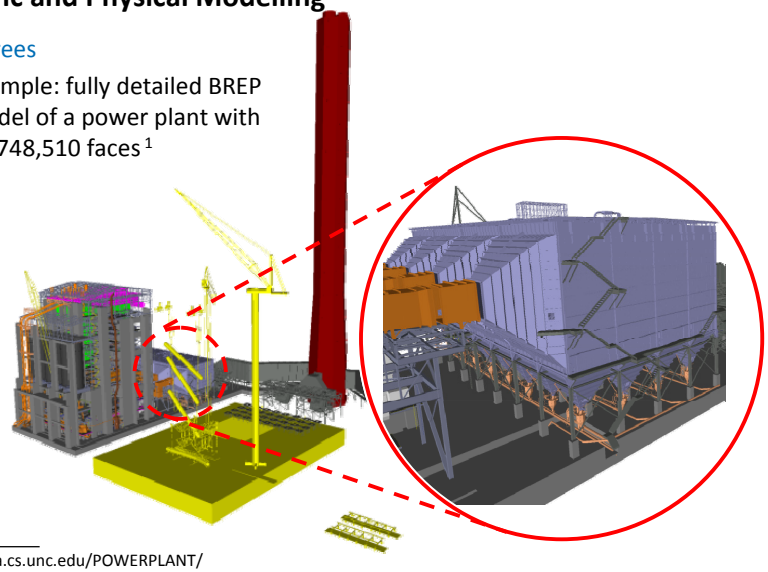


quadtree



Geometric and Physical Modelling

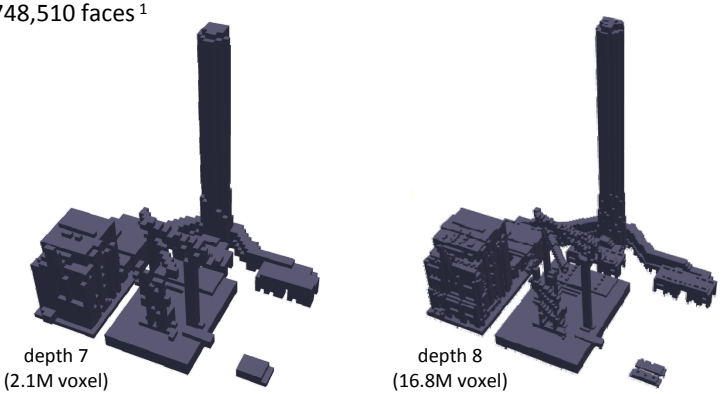
- spacetrees
  - example: fully detailed BREP model of a power plant with 12,748,510 faces [1]



[1] http://gamma.cs.unc.edu/POWERPLANT/

Geometric and Physical Modelling

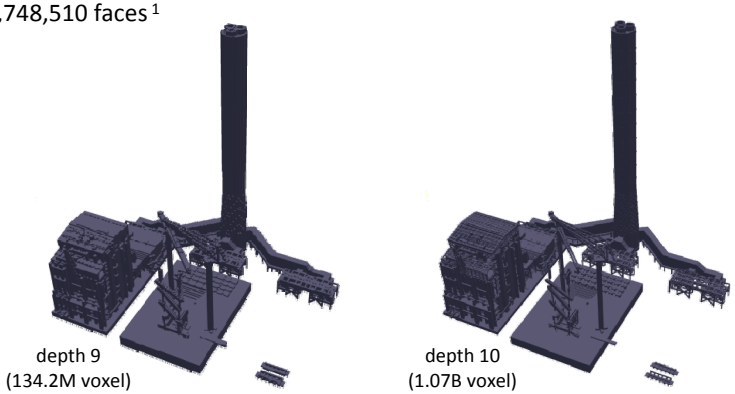
- spacetrees
  - example: fully detailed BREP model of a power plant with 12,748,510 faces [1]



[1] http://gamma.cs.unc.edu/POWERPLANT/

Geometric and Physical Modelling

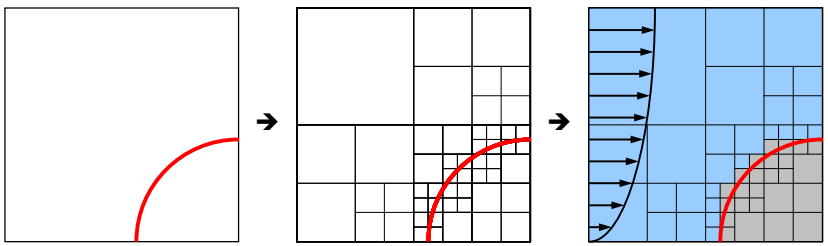
- spacetrees
  - example: fully detailed BREP model of a power plant with 12,748,510 faces [1]



[1] http://gamma.cs.unc.edu/POWERPLANT/

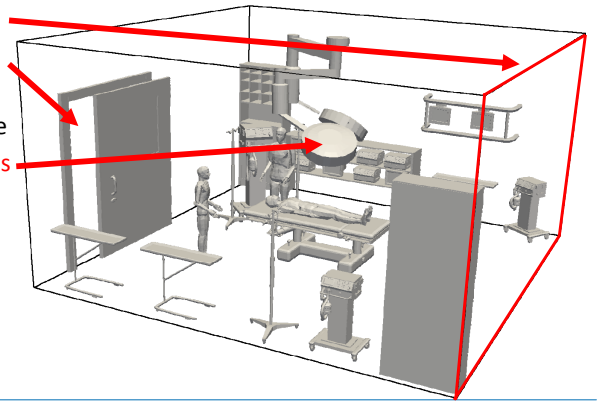
Geometric and Physical Modelling

- generation of computational domain
  1. discretisation of computational domain
  2. tree balancing (1:2) to avoid numerical instabilities
  3. setting cell attributes (fluid / obstacle)
  4. setting boundary conditions (inflow /outflow / wall / …)



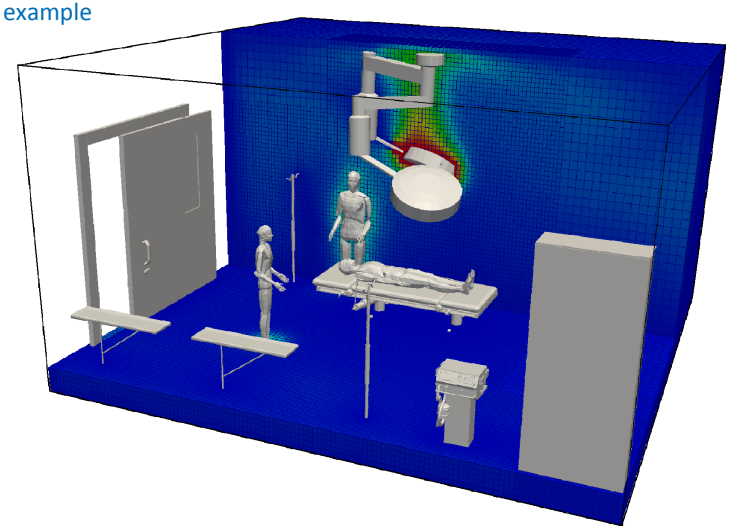
Geometric and Physical Modelling

- complex example
  - operating theatre at `Klinikum rechts der Isar´ (MRI)
  - dimensions: 6.30×6.25×3.50 m
  - ventilation:
    - inflow: right wall
    - outflow: door slit
  - idea: keep air above patient pollutant-free
  - but hot surgical lamps influence fluid flow
  - thermal simulation using adaptive grids of different depths



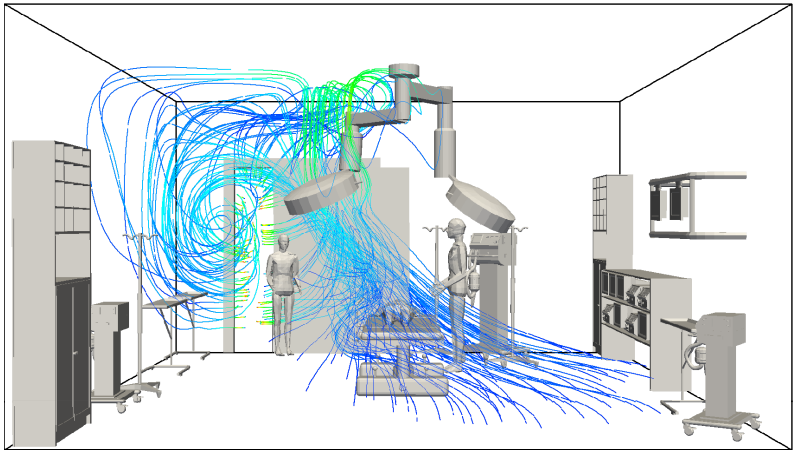
Geometric and Physical Modelling

- complex example



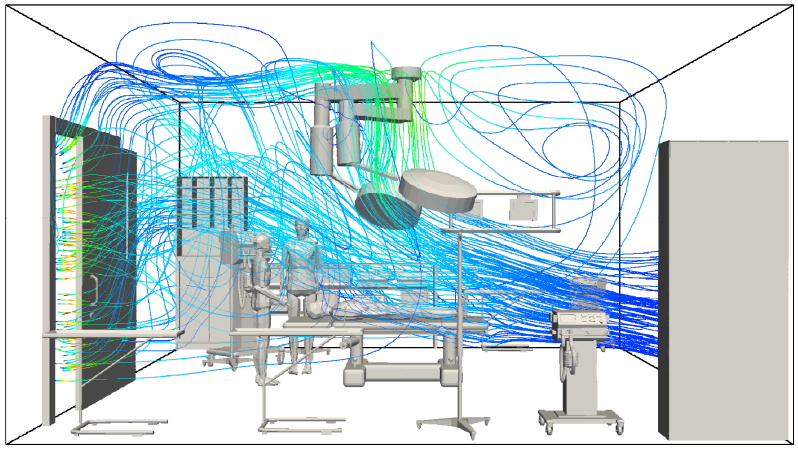
Geometric and Physical Modelling

- complex example



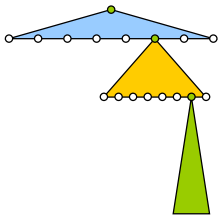
Geometric and Physical Modelling

- complex example

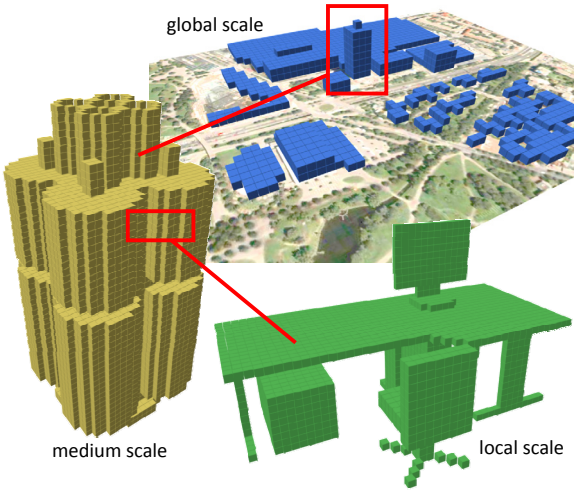


Geometric and Physical Modelling

- level of detail concepts

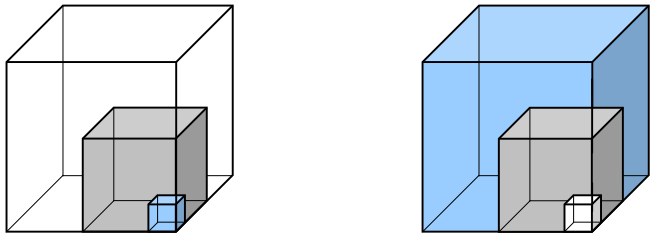


BMW „Vierzylinder“, Munich



Geometric and Physical Modelling

- level of detail concepts
  - towards multiscale simulations



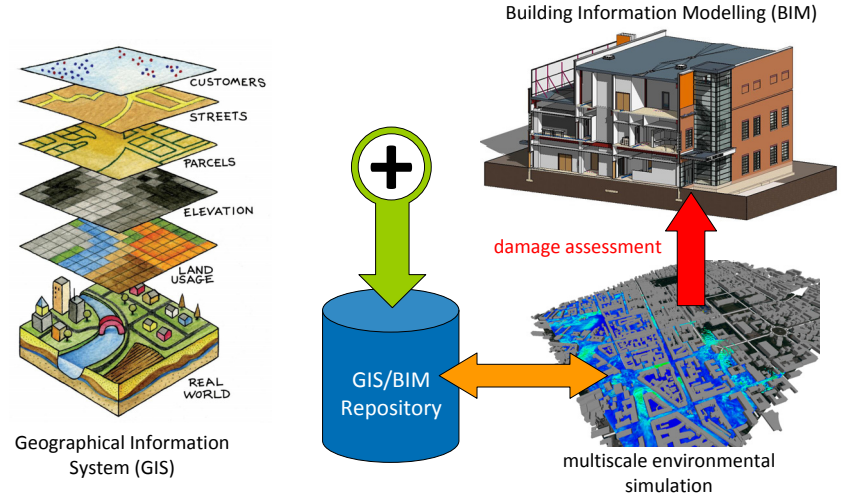
deductive approach vs. inductive approach

e.g. flood scenarios / natural disasters → local damage assessment

e.g. viral outbreak / artificial disasters → global damage assessment

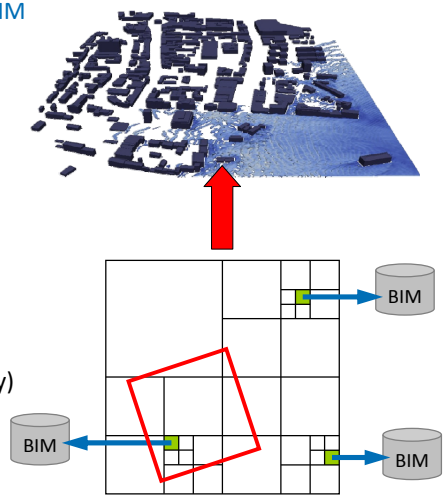
Geometric and Physical Modelling

- bridging worlds and scales: GIS and BIM



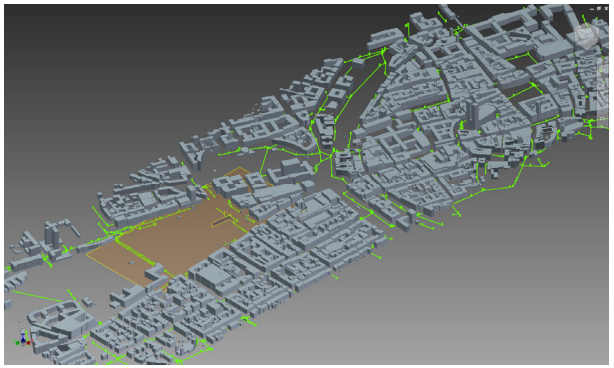
Geometric and Physical Modelling

- bridging worlds and scales: GIS and BIM
  - global scale (GIS)
    - height fields
    - low-fidelity geometries
    - sewerage system
  - local scale (BIM)
    - high-fidelity product models
    - context information
  - GIS/BIM repository (spacetrees)
    - location awareness (proximity)
    - LoD decisions / abstractions
    - selecting region of interest



Geometric and Physical Modelling

- bridging worlds and scales: GIS and BIM
  - coupling with city's sewerage system
    - 3D fluid flows ←→ 1D fluid flow
    - water head from 3D simulation as BC for 1D simulation

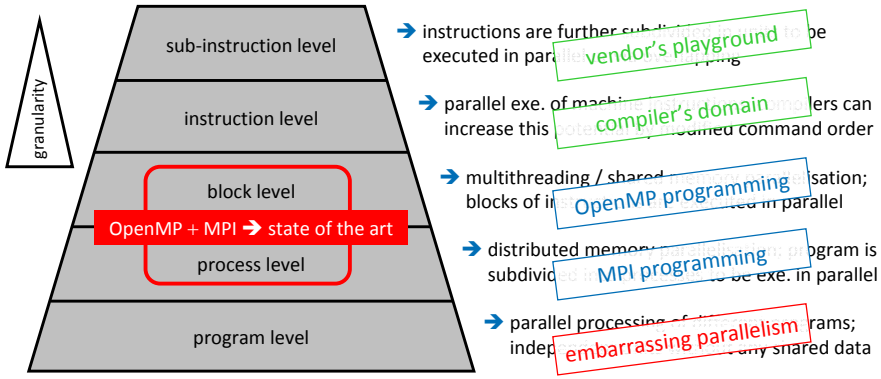


Munich city centre plus sewerage

- **overview**
  - geometric and physical modelling
  - foundations / parallel architectures
  - multigrid methods
  - towards massive parallel HPC…
  - interactive visual data exploration

Foundations / Parallel Architectures

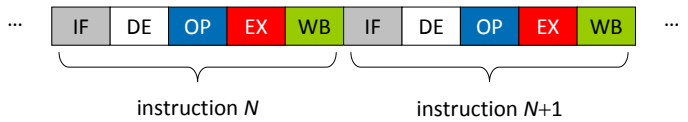
- levels of parallelism



Foundations / Parallel Architectures

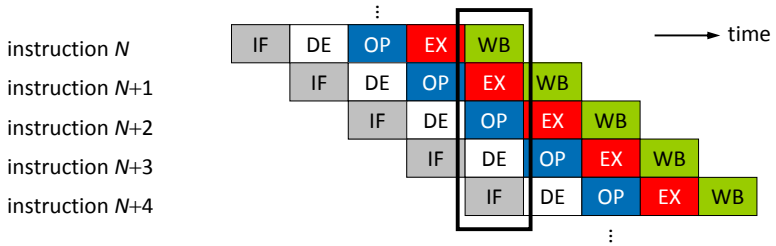
- a brief history of time: instruction pipelining
  - instruction execution involves several operations
    1. instruction fetch (IF)
    2. decode (DE)
    3. fetch operands (OP)
    4. execute (EX)
    5. write back (WB)

    which are *executed successively*
  - hence, only one part of CPU works at a given moment



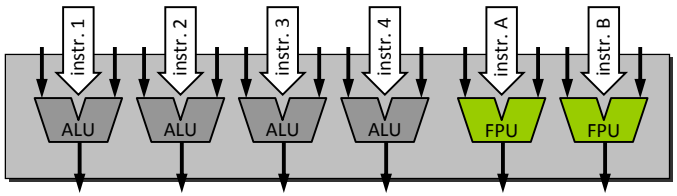
Foundations / Parallel Architectures

- a brief history of time: instruction pipelining
  - observation: while processing particular stage of instruction, other stages are idle
  - hence, multiple instructions to be overlapped in execution ➔ instruction pipelining (similar to assembly lines)
  - advantage: no additional hardware necessary



Foundations / Parallel Architectures

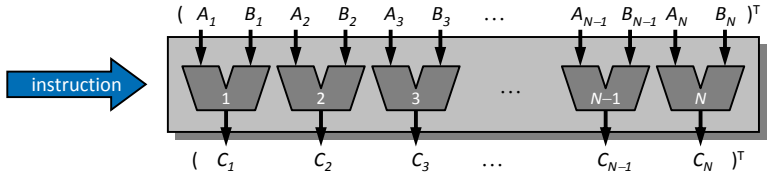
- a brief history of time: superscalar
  - faster CPU throughput due to simultaneously execution of instructions within one clock cycle via redundant functional units (ALU, multiplier, …)
  - dispatcher decides (during runtime) which instructions read from memory can be executed in parallel and dispatches them to different functional units
  - for instance, PowerPC 970 (4 × ALU, 2 × FPU)



  - but, performance improvement is limited (intrinsic parallelism)

Foundations / Parallel Architectures

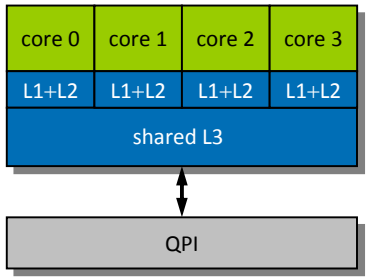
- a brief history of time: vector units
  - simultaneously execution of *one instruction* on a *one-dimensional array of data* (≡ vector)
  - VU first appeared in 1970s and were the basis of most supercomputers in the 1980s and 1990s

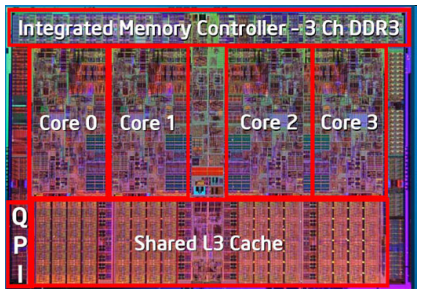


  - specialised hardware ➔ very expensive
  - limited application areas (mostly Computational Fluid Dynamics, Computational Structures Dynamics, …)

Foundations / Parallel Architectures

- INTEL Nehalem Core i7



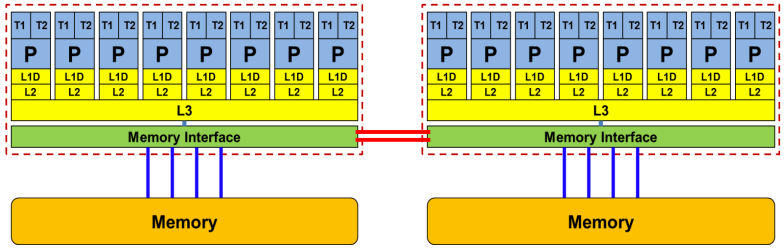


source: www.samrathacks.com

QPI: QuickPath Interconnect replaces FSB (QPI is a point-to-point interconnection – with a memory controller now on-die – in order to allow both reduced latency and higher bandwidth ➔ up to (theoretically) 25.6 GByte/s data transfer, i.e. 2× FSB)

Foundations / Parallel Architectures

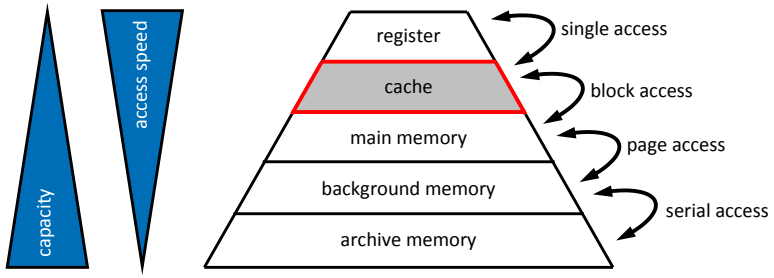
- Intel E5-2600 Sandy-Bridge Series
  - 2 CPUs connected by 2 QPIs (Intel Quick Path Interconnect)
  - Quick Path Interconnect (1 sending and 1 receiving port)
    - 8 GT/s · 16 Bit/T payload · 2 directions / 8 Bit/Byte = 32 GB/s max bandwidth per QPI
    - 2 QPI links ➔ 2 · 32 GB/s = 64 GB/s max bandwidth



source: G. Wellein, RRZE

Foundations / Parallel Architectures

- reminder: memory hierarchy
  - memory hierarchy
    - exploitation of program characteristics such as locality
    - compromise between costs and performance
    - components with different speeds and capacities



Foundations / Parallel Architectures

- reminder: memory hierarchy
  - example: SCHOENAUER vector triad benchmark
    - main kernel

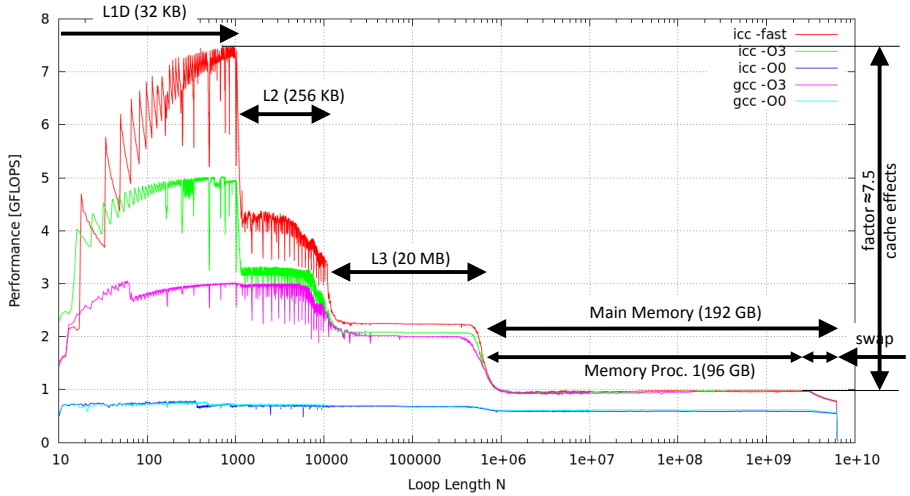
```
double *A, *B, *C, *D

for i ← 0 to N–1 do
  A[i] ← B[i] + C[i] * D[i]
od
```

    - report performance for different *N*
    - kernel is limited by data transfer performance for all memory levels
    - using different compilers on Sandy-Bridge architecture
      - Intel Compiler 13.0.0 (icc)
      - GNU Compiler 4.6.3 (gcc)

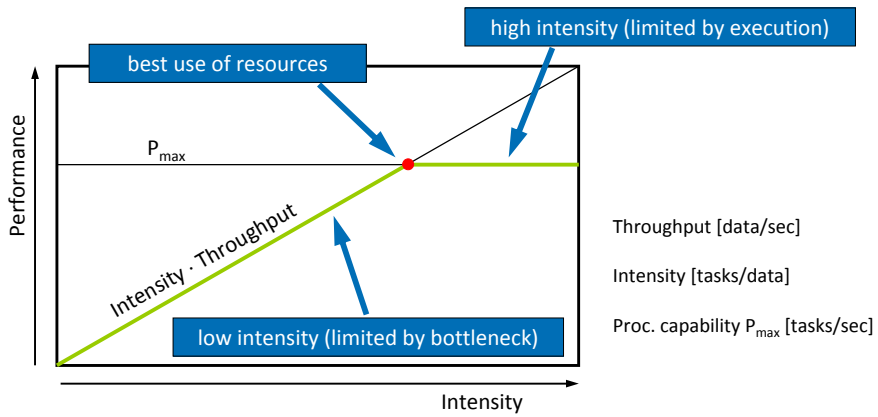
Foundations / Parallel Architectures

- reminder: memory hierarchy



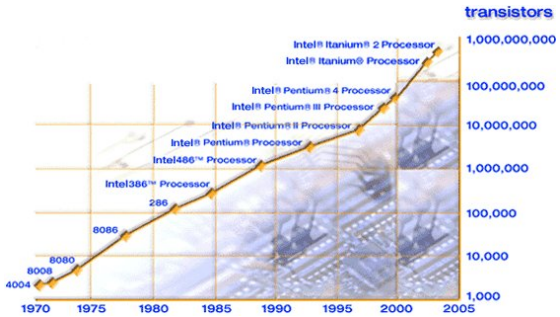
Foundations / Parallel Architectures

- roofline model
  - an optimistic performance model (for node level optimisation)



Foundations / Parallel Architectures

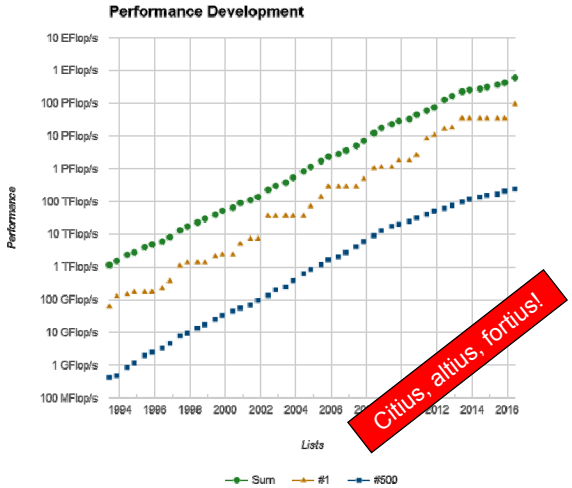
- MOORE's law
  - observation of Intel co-founder Gordon E. MOORE, describes important trend in history of computer hardware (1965)



"number of transistors that can be placed on an integrated circuit is increasing exponentially, doubling approximately every two years"

Foundations / Parallel Architectures

- some numbers: Top500 (as of June 2016)



Foundations / Parallel Architectures

- some numbers: Top500 (as of June 2016)

Foundations / Parallel Architectures

- the 10 fastest supercomputers in the world (as of June 2016)

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6	125,435.9	15,371
2	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
3	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
4	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
5	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660
6	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,586.6	10,066.3	3,945
7	DOE/NNSA/LANL/SNL United States	**Trinity** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect Cray Inc.	301,056	8,100.9	11,078.9	
8	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Cray Inc.	115,984	6,271.0	7,788.9	2,325
9	HLRS - Höchstleistungsrechenzentrum Stuttgart Germany	**Hazel Hen** - Cray XC40, Xeon E5-2680v3 12C 2.50GHz, Aries interconnect Cray Inc.	185,088	5,640.2	7,403.5	
10	King Abdullah University of Science and Technology Saudi Arabia	**Shaheen II** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect Cray Inc.	196,608	5,537.0	7,235.2	2,834

Rpeak ≡ theoretical peak performance

Rmax ≡ sustained peak performance

- **overview**
  - geometric and physical modelling
  - foundations / parallel architectures
  - multigrid methods
  - towards massive parallel HPC…
  - interactive visual data exploration

Multigrid Methods

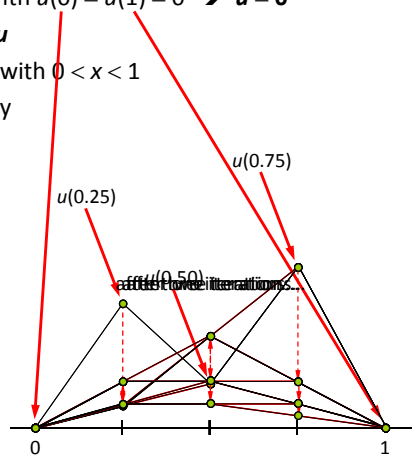
- solvers for linear systems
  - many PDEs result in a system of linear equations $\boldsymbol{A}\cdot\boldsymbol{u} = \boldsymbol{f}$
  - solution of such linear systems via
    - direct solvers
    - iterative solvers
  - typical iterative solvers
    - RICHARDSON method
    - JACOBI method
    - GAUSS-SEIDEL method
    - relaxation methods

    simple / moderate parallelisation effort
    - CG and derivatives
    - multigrid methods

    most effective and considered to be S.O.T.A. ☺

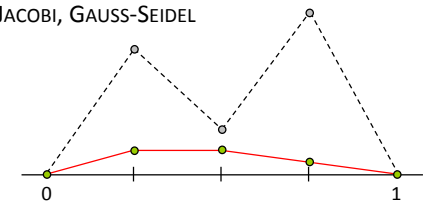
Multigrid Methods

- something about smoother
  - model BV problem: $-u'' = 0$ with $u(0) = u(1) = 0$ ➔ $\boldsymbol{u} = \boldsymbol{0}$
  - from the above follows $\boldsymbol{e} = -\boldsymbol{u}$
  - arbitrary start values for $u(x)$ with $0 < x < 1$
  - initial error $\boldsymbol{e}$ highly oscillatory
  - now applying a smoother…



Multigrid Methods

- something about smoother
  - model BV problem: $-u'' = 0$ with $u(0) = u(1) = 0$ ➔ $\boldsymbol{u} = \boldsymbol{0}$
  - from the above follows $\boldsymbol{e} = -\boldsymbol{u}$
  - arbitrary start values for $u(x)$ with $0 < x < 1$
  - initial error $\boldsymbol{e}$ highly oscillatory
  - now applying a smoother…
- some observations
  - high frequency parts of error are *smoothed* out by standard solvers such as JACOBI, GAUSS-SEIDEL
  - on smooth functions above solvers become ineffective



Multigrid Methods

- a more analytical approach
  - one smoothing step to be represented as

$$\boldsymbol{u}_1 = \boldsymbol{R}\cdot\boldsymbol{u}_0 + \boldsymbol{g}$$

with $\boldsymbol{R}$ denoting the iteration matrix of the smoother; furthermore, the exact solution $\hat{\boldsymbol{u}}$ is a fixed-pointed of the iteration, that means

$$\hat{\boldsymbol{u}} = \boldsymbol{R}\cdot\hat{\boldsymbol{u}} + \boldsymbol{g}$$

  - with $\boldsymbol{e} = \hat{\boldsymbol{u}} - \boldsymbol{u}$ subtracting the last two expressions yields

$$\boldsymbol{e}_1 = \boldsymbol{R}\cdot\boldsymbol{e}_0$$

  - repeating this, after m smoothing steps the error is given by

$$\boldsymbol{e}_m = \boldsymbol{R}^m\cdot\boldsymbol{e}_0$$

  - with $\rho(\boldsymbol{R}) < 1$, the error is forced to zero as the iteration proceeds

Multigrid Methods

- a more analytical approach
  - let $\boldsymbol{w}_k$ denoted the k-th eigenvector of $\boldsymbol{R}$, then it is possible to expand $\boldsymbol{e}_0$ as

$$\boldsymbol{e}_0 = \sum_{k=1}^{n-1} c_k\cdot\boldsymbol{w}_k$$

with coefficients $c_k \in \mathbb{R}$ denoting weighting factors for each $\boldsymbol{w}_k$ in the error

  - using

$$\boldsymbol{e}_m = \boldsymbol{R}^m\cdot\boldsymbol{e}_0$$

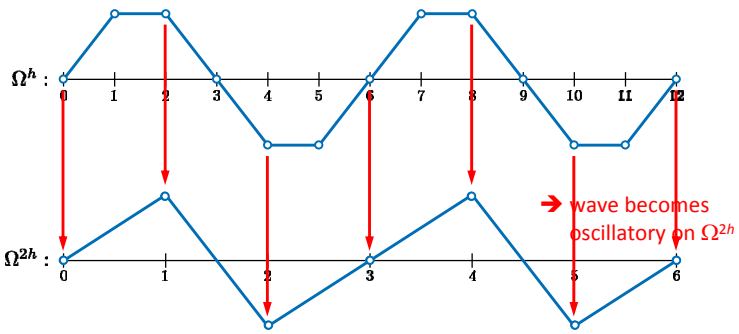
and the eigenvector expansion for $\boldsymbol{e}_0$, we get

$$\boldsymbol{e}_m = \boldsymbol{R}^m\cdot\boldsymbol{e}_0 = \sum_{k=1}^{n-1} c_k\cdot\boldsymbol{R}^m\cdot\boldsymbol{w}_k \overset{\boldsymbol{R}\cdot\boldsymbol{w}_k = \lambda_k(\boldsymbol{R})\cdot\boldsymbol{w}_k}{=} \boxed{\sum_{k=1}^{n-1} c_k\cdot\lambda_k(\boldsymbol{R})^m\cdot\boldsymbol{w}_k}$$

  - from above expansion we see that small eigenvalues ($\cong 0$) corresponding to high frequency parts of the error diminish faster than large eigenvalues ($\cong 1$) corresponding to low frequency parts of the error

Multigrid Methods

- towards multigrid
  - how do smooth components look like on coarser grids?
  - consider some fine (Ω^h) and coarse (Ω^{2h}) grid with double grid spacing
  - given some smooth wave on Ω^h with $n = 13$ points
  - Ω^{2h} representation with $n = 7$ points via direct projection



Multigrid Methods

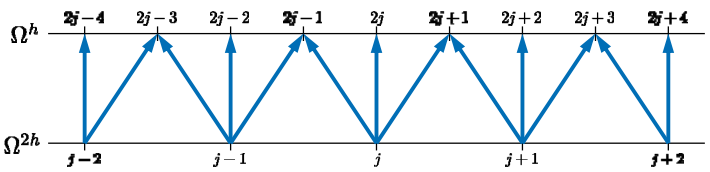
- towards multigrid
  - idea: when relaxation begins to stall, signalling the predominance of smooth error modes, move to a coarser grid as smooth error modes appear oscillatory there
  - basic two-grid correction scheme

relax on $\boldsymbol{A}\cdot\boldsymbol{u} = \boldsymbol{f}$ on Ω^h to obtain an approximation $\boldsymbol{v}^h$
compute residual $\boldsymbol{r} = \boldsymbol{f} - \boldsymbol{A}\cdot\boldsymbol{v}^h$
relax on $\boldsymbol{A}\cdot\boldsymbol{e} = \boldsymbol{r}$ on Ω^{2h} to obtain an approximation to the error $\boldsymbol{e}^{2h}$
correct $\boldsymbol{v}^h \leftarrow \boldsymbol{v}^h + \boldsymbol{e}^{2h}$ on Ω^h with error estimate $\boldsymbol{e}^{2h}$ obtained on Ω^{2h}

  - question: how to transfer residual $\boldsymbol{r}^h$ from Ω^h to Ω^{2h} (called restriction) and how to transfer the error estimate $\boldsymbol{e}^{2h}$ back from Ω^{2h} to Ω^h (called interpolation or prolongation)?

Multigrid Methods

- towards multigrid
  - prolongation operator $\mathbf{I}_{2h}^{h}$
    ➔ produces fine-grid vectors from coarse ones according to $\mathbf{I}_{2h}^{h}\boldsymbol{v}^{2h} = \boldsymbol{v}^{h}$
  - simplest approach: linear prolongation

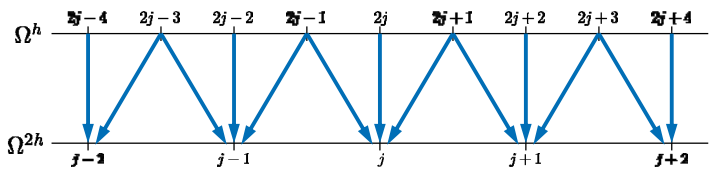


with

$$v_{2j}^{h} = v_{j}^{2h}$$
$$v_{2j+1}^{h} = \frac{1}{2}\left(v_{j}^{2h} + v_{j+1}^{2h}\right) \quad , 0 \le j \le \frac{n}{2} - 1$$

Multigrid Methods

- towards multigrid
  - restriction operator $\mathbf{I}_{h}^{2h}$
    ➔ produces coarse-grid vectors from fine ones according to $\mathbf{I}_{h}^{2h}\boldsymbol{v}^{h} = \boldsymbol{v}^{2h}$
  - typical approach: full weighting



with

$$v_{j}^{2h} = \frac{1}{4}\left(v_{2j-1}^{h} + 2v_{2j}^{h} + v_{2j+1}^{h}\right) \quad , 0 \le j \le \frac{n}{2} - 1$$

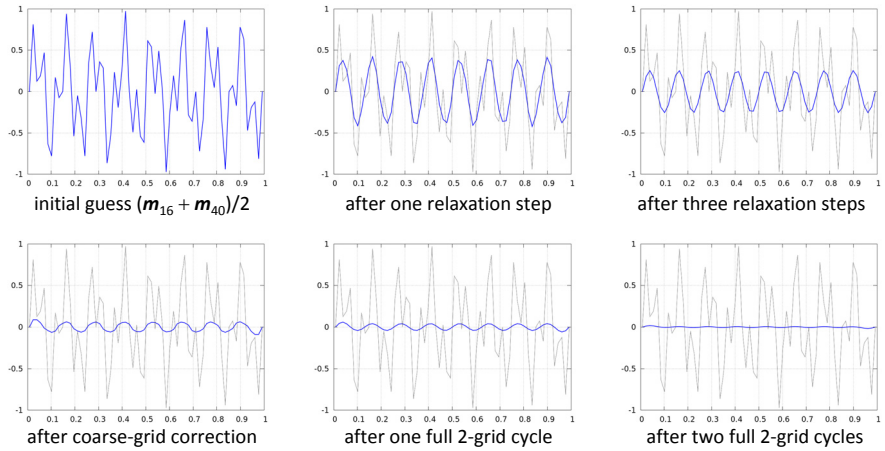
Multigrid Methods

- two-grid correction scheme
  - now using well-defined ways to transfer vectors between grids
  - parameters ν_1, ν_2 control number of relaxation steps and are in practice often 1, 2, or 3

relax ν_1 times on $\boldsymbol{A}^h \cdot \boldsymbol{v}^h = \boldsymbol{f}^h$ on Ω^h with initial guess $\boldsymbol{v}^h$
compute residual $\boldsymbol{r}^h = \boldsymbol{f}^h - \boldsymbol{A}^h \cdot \boldsymbol{v}^h$
restrict residual $\boldsymbol{r}^h$ to coarse grid by $\boldsymbol{r}^{2h} = \mathbf{I}_{h}^{2h}\boldsymbol{r}^h$
solve $\boldsymbol{A}^{2h} \cdot \boldsymbol{e}^{2h} = \boldsymbol{r}^{2h}$ on Ω^{2h}
prolongate coarse-grid error $\boldsymbol{e}^{2h}$ to fine grid by $\boldsymbol{e}^h = \mathbf{I}_{2h}^{h}\boldsymbol{e}^{2h}$
correct fine-grid approximation $\boldsymbol{v}^h \leftarrow \boldsymbol{v}^h + \boldsymbol{e}^h$
relax ν_2 times on $\boldsymbol{A}^h \cdot \boldsymbol{v}^h = \boldsymbol{f}^h$ on Ω^h with corrected approximation $\boldsymbol{v}^h$

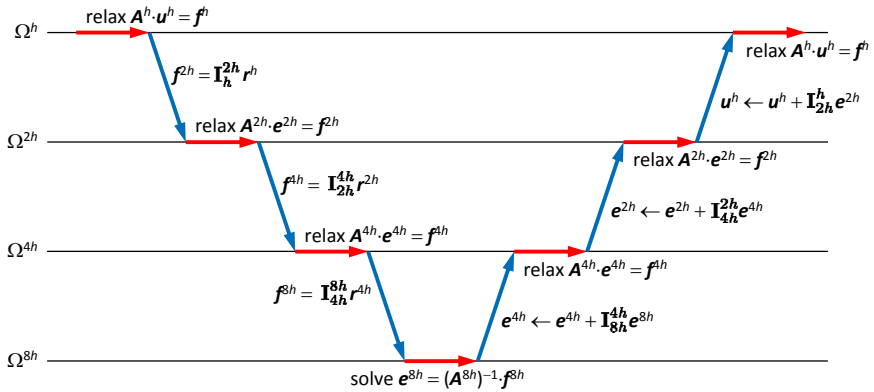
Multigrid Methods

- two-grid correction scheme
  - example ($\boldsymbol{u} = \mathbf{0}$) with overlay of FOURIER modes $\boldsymbol{m}_{16}$ and $\boldsymbol{m}_{40}$ as initial guess



Multigrid Methods

- V-cycle scheme
  - why restricting approach to two grids only?
  - idea: recursive algorithm



Multigrid Methods

- V-cycle scheme

$v^k \leftarrow \mathrm{MG_V}(v^k, f^k)$

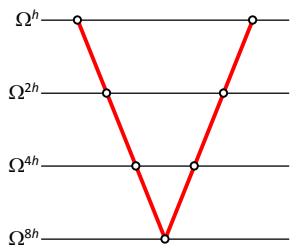
1. relax ν_1 times on $A^k \cdot v^k = f^k$ with initial guess v^k
2. if Ω^k = coarsest grid, then go to step 4

   else

   $f^{2k} \leftarrow I_k^{2k}(f^k - A^k \cdot v^k)$

   $v^{2k} \leftarrow \mathbf{0}$

   $v^{2k} \leftarrow \mathrm{MG_V}(v^{2k}, f^{2k})$
3. correct $v^k \leftarrow v^k + I_{2k}^{k} v^{2k}$
4. relax ν_2 times on $A^k \cdot v^k = f^k$



Multigrid Methods

- full multigrid V-cycle

$v^k \leftarrow \mathrm{FMG}(f^k)$

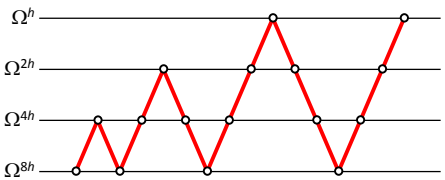
1. if Ω^k = coarsest grid, set $v^k \leftarrow \mathbf{0}$ and go to step 3

   else

   $f^{2k} \leftarrow I_k^{2k}(f^k)$

   $v^{2k} \leftarrow \mathrm{FMG}(f^{2k})$
2. correct $v^k \leftarrow I_{2k}^{k} v^{2k}$
3. $v^k \leftarrow \mathrm{MG_V}(v^k, f^k)$ ν_0 times

Here, the idea is to use coarse grids in order to obtain better initial guesses, a strategy called nested iteration.

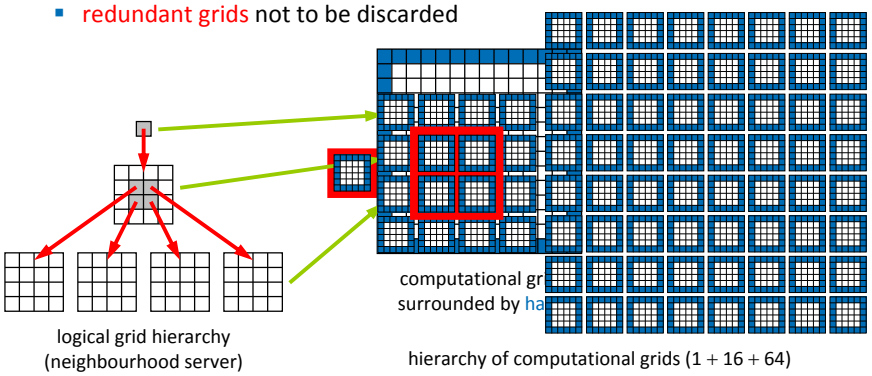


full multigrid with $\nu_0 = 1$

- **overview**
  - geometric and physical modelling
  - foundations / parallel architectures
  - multigrid methods
  - towards massive parallel HPC…
  - interactive visual data exploration

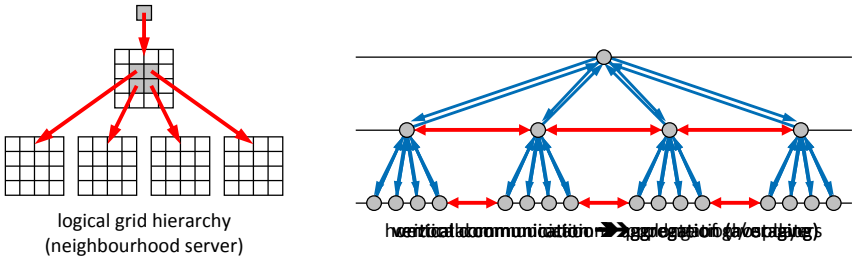
Towards Massive Parallel HPC…

- data structure / grid layout
  - nested non-overlapping block-structured orthogonal grids
  - management (i.e. neighbourhood server) hidden from application
  - each logical cell links to a computational grid surrounded by halo
  - redundant grids not to be discarded



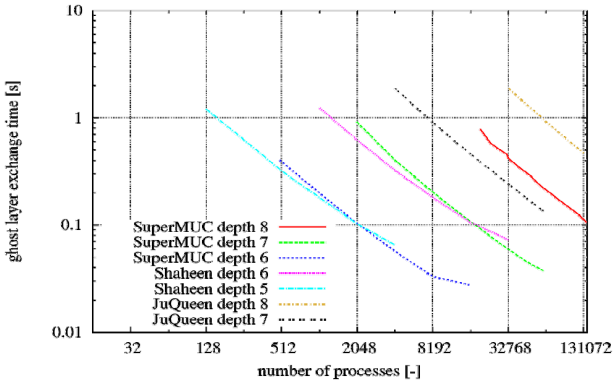
Towards Massive Parallel HPC…

- data structure / grid layout
  - nested non-overlapping block-structured orthogonal grids
  - management (i.e. neighbourhood server) hidden from application
  - each logical cell links to a computational grid surrounded by halo
- data flow
  - vertical communication (aggregation / prolongation of values)
  - horizontal communication (update of ghost layers)



Towards Massive Parallel HPC…

- data flow between grids
  - time for one full processing, i.e. bottom-up + horizontal + top-down communication between all grids (no computation done)



depth 6: layout with 2×2×2 refinement and 16×16×16 blocks up to 16'384 procs.

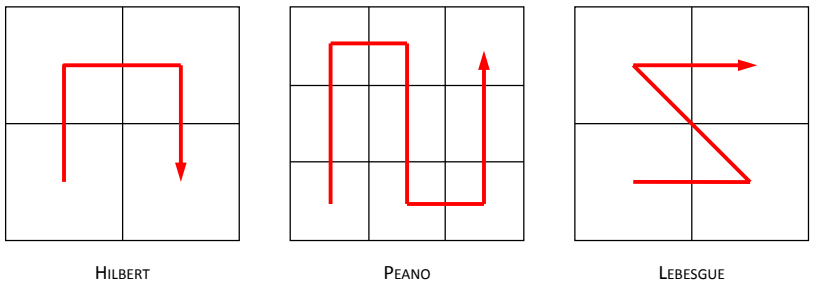
depth 7: layout with 2×2×2 refinement and 16×16×16 blocks up to 65'536 procs.

depth 8: layout with 2×2×2 refinement and 16×16×16 blocks up to 147'456 procs.

depth 8: 4096×4096×4096 (total of 80B computing cells; 707B transferred variables)

Towards Massive Parallel HPC…

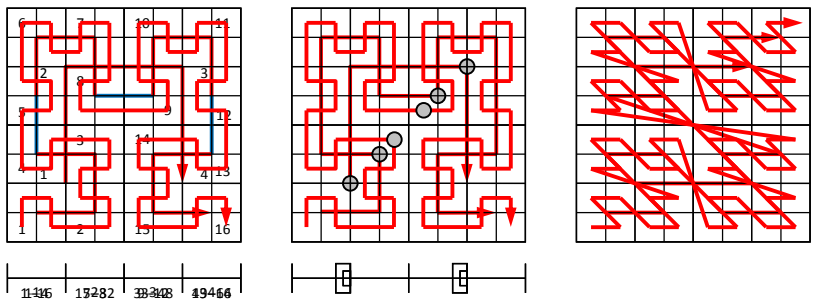
- space-filling curves (SFC)
  - continuous, surjective mapping $f : [0, 1] \rightarrow [0, 1]^D$
  - advantage: preserving neighbourhood relations
  - typical representatives (generator or 'Leitmotiv')



- SFC due to recursive approach starting with one 'Leitmotiv' above

Towards Massive Parallel HPC…

- space-filling curves (SFC)
  - continuous, surjective mapping $f : [0, 1] \rightarrow [0, 1]^D$
  - advantage: preserving neighbourhood relations
  - typical representatives (generator or 'Leitmotiv')



➔ SFC due to recursive approach starting with one 'Leitmotiv' above

Towards Massive Parallel HPC…

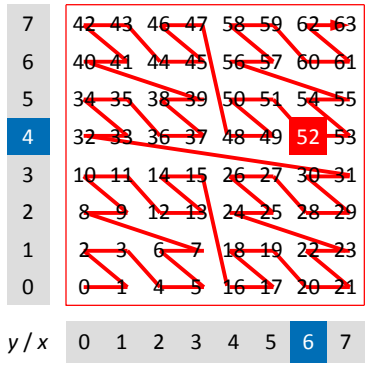
- space-filling curves (SFC)
  - for load distribution inverse function $f^{-1} : [0, 1]^D \rightarrow [0, 1]$ necessary
  - simple conversion of Z-index in case of LEBESGUE's SFC possible
  - idea: bitwise interleaving of coordinate values

$x = 6 \rightarrow$ **110**

$y = 4 \rightarrow$ **100**

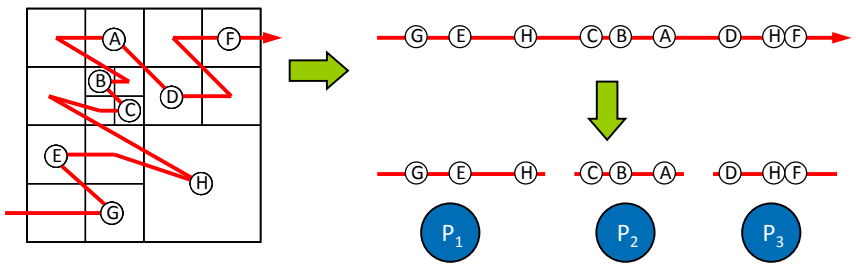
110100 $\rightarrow 52 = Z$

➔ simple conversion $(6, 4) \leftrightarrow 52_Z$



Towards Massive Parallel HPC…

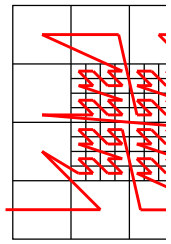
- space-filling curves (SFC)
  - load distribution / balancing
    - assign some iteration of SFC to points in 2D-space
    - linearise data according to SFC
    - simple partition of data (preserving locality) to processors possible



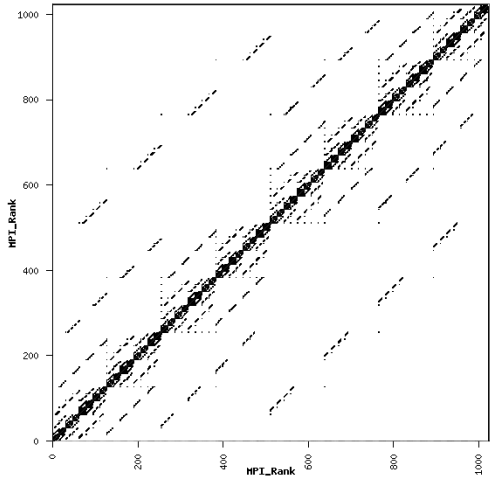
Towards Massive Parallel HPC…

- grid distribu
  - space-fi
  - neighbo
  - simple g

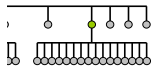
processes



SFC grid distrib



communication pattern (obtained with IPM[1])



n) answers queries
ls ➔ grids don't need
ores / processes

[1] Integrated Performance Monitoring, http://ipm-hpc.sourceforge.net/

Towards Massive Parallel HPC…

- grid distribution / load balancing
  - example: temperature distribution – grid migration

Towards Massive Parallel HPC…

- computational kernel
  - NS equations, FV for spatial, Adams-Bashforth (2nd order FD) for temporal discretisation
  - fractional step (Chorin's projection) for solving time-dependent incompressible flow equations, i.e. iterative procedure between velocity and pressure during one time step
  - thermal coupling realised by Boussinesq approximation (modified body term in NSE momentum equation)

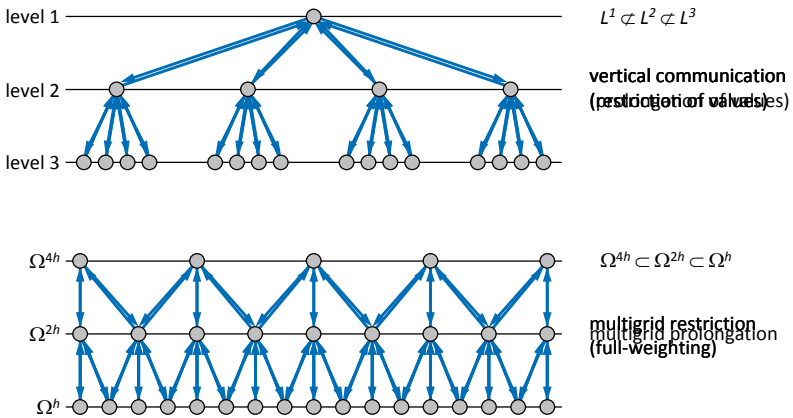
$$\nabla \cdot \vec{u} = 0$$

$$\frac{\partial \rho_\infty u_i}{\partial t} + \nabla \cdot (\rho_\infty u_i \vec{u}) = \nabla \cdot (\mu \nabla u_i) - \nabla \cdot (p \vec{e}_i) - \rho_\infty \cdot \beta \cdot (T - T_\infty) g_i \quad , \text{ with } i \in \{x, y, z\}$$

$$\frac{\partial T}{\partial t} + \nabla \cdot (T \vec{u}) - \nabla \cdot (\alpha \nabla T) - \frac{q_{int}}{\rho_\infty \cdot c_p} = 0$$

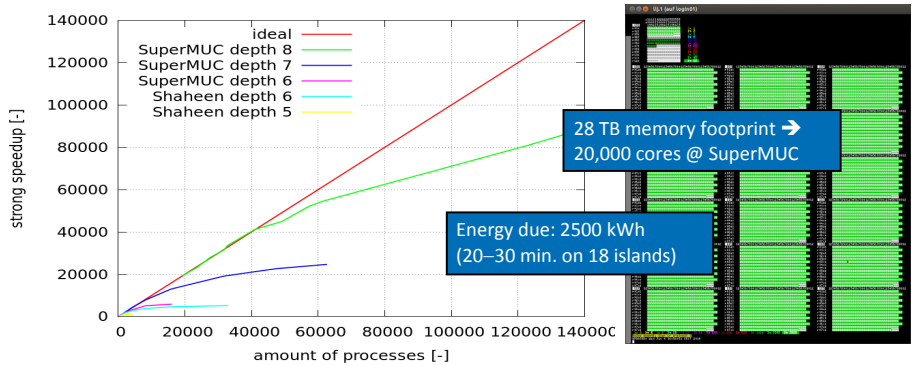
Towards Massive Parallel HPC…

- parallel multigrid(-like) solver
  - comparison: vertical communication vs. multigrid transfer functions



Towards Massive Parallel HPC…

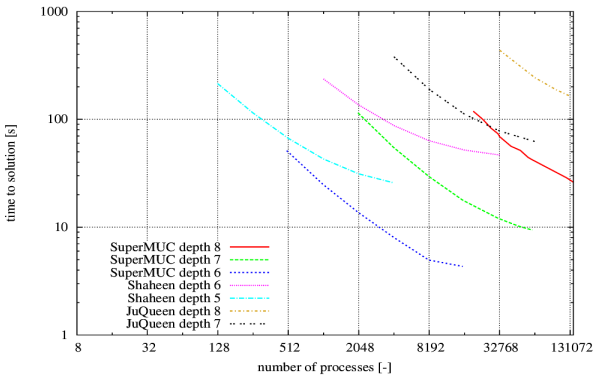
- parallel multigrid(-like) solver



solving $\Delta u = 0$ for 3D domain with 19'173'961 grids and resolution 4096×4096×4096 (i.e. approx. 707B DOFs); times obtained on SuperMUC and Shaheen (IBM Blue Gene/P)

Towards Massive Parallel HPC…

- parallel multigrid(-like) solver
  - time to solution for one time step (repeated V-cyles with adaptive relaxation steps (and secret scaling factor ☺) until convergence)



depth 6: layout with 2×2×2 refinement and 16×16×16 blocks up to 16’384 procs.

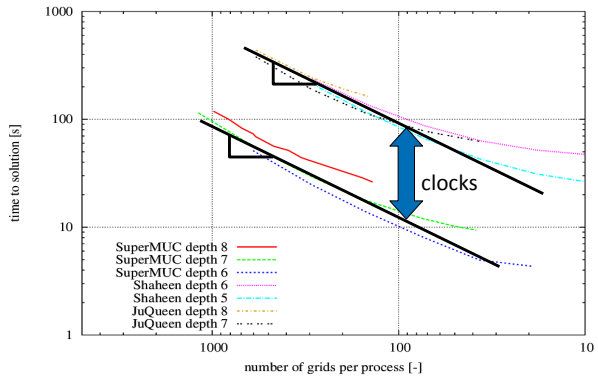
depth 7: layout with 2×2×2 refinement and 16×16×16 blocks up to 65’536 procs.

depth 8: layout with 2×2×2 refinement and 16×16×16 blocks up to 147’456 procs.

depth 8: 4096×4096×4096 (total of 80B computing cells; 707B degrees of freedom)

Towards Massive Parallel HPC…

- parallel multigrid(-like) solver
  - time to solution for one time step (repeated V-cyles with adaptive relaxation steps (and secret scaling factor ☺) until convergence)



depth 6: layout with 2×2×2 refinement and 16×16×16 blocks up to 16’384 procs.

depth 7: layout with 2×2×2 refinement and 16×16×16 blocks up to 65’536 procs.

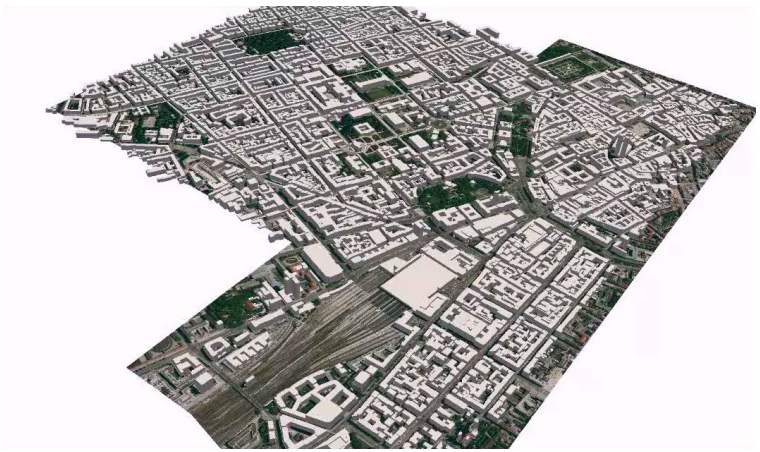
depth 8: layout with 2×2×2 refinement and 16×16×16 blocks up to 147’456 procs.

depth 8: 4096×4096×4096 (total of 80B computing cells; 707B degrees of freedom)

Towards Massive Parallel HPC…

Towards Massive Parallel HPC…

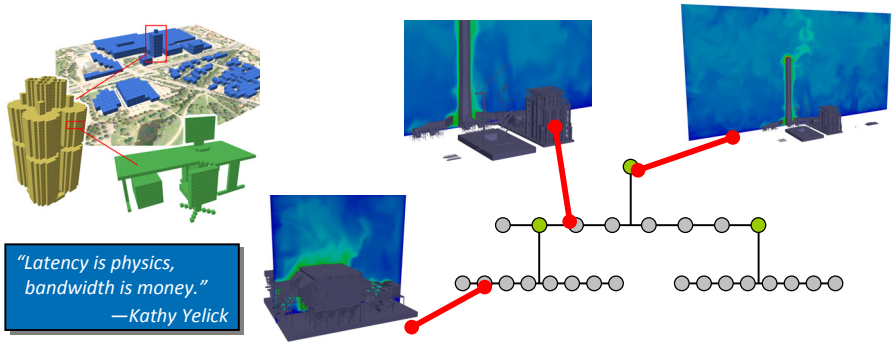
- multiscale flood simulation



- **overview**
  - geometric and physical modelling
  - foundations / parallel architectures
  - multigrid methods
  - towards massive parallel HPC…
  - interactive visual data exploration

Interactive Visual Data Exploration

- sliding window
  - idea: online navigation through details

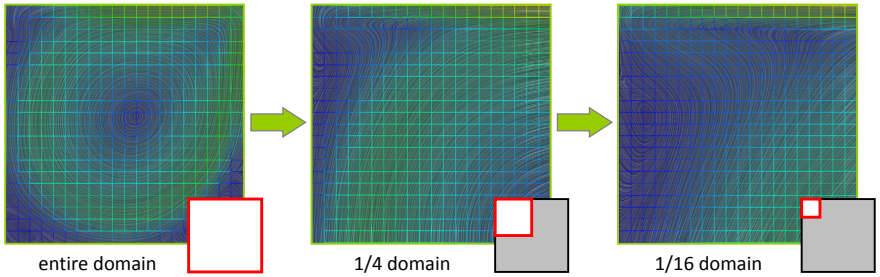


"Latency is physics, bandwidth is money."
—Kathy Yelick

"High-performance computing must now assume a broader meaning, encompassing not only flops, but also the ability, for example, to efficiently manipulate vast and rapidly increasing quantities of both numerical and non-numerical data."

Interactive Visual Data Exploration

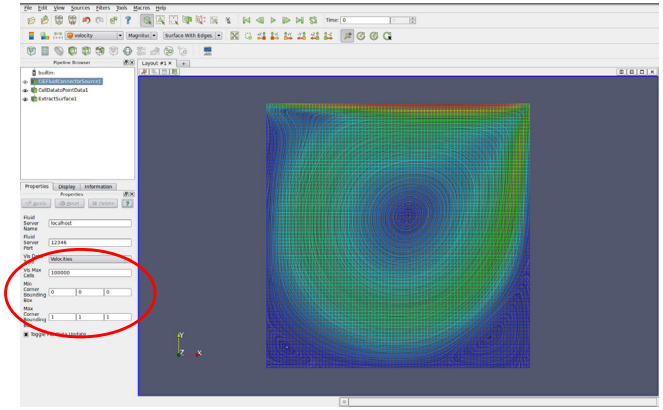
- sliding window concept
  - problem: high resolutions hinder interactive exploration
  - solution: user moves / sizes 'window' through domain for data exploration ➔ amount of details increases seamlessly
  - constant bandwidth of data transmission ➔ simple postprocessing



entire domain (every 16th point) | 1/4 domain (every 4th point) | 1/16 domain (all points)

Interactive Visual Data Exploration

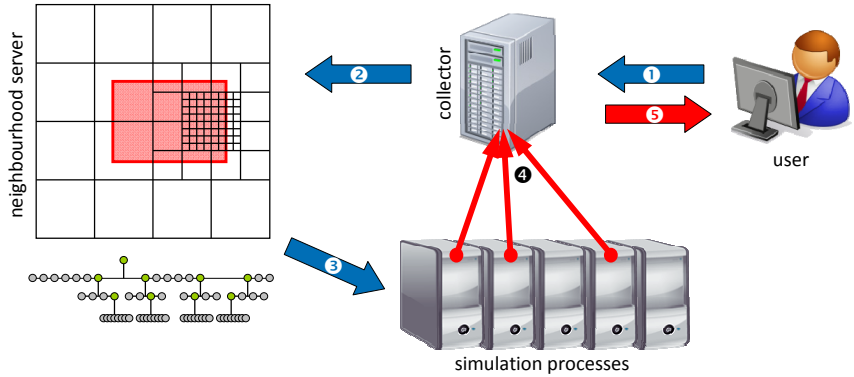
- sliding window concept
  - simple part: what happens on the front-end…



ParaView plug-in for setting window's size and location

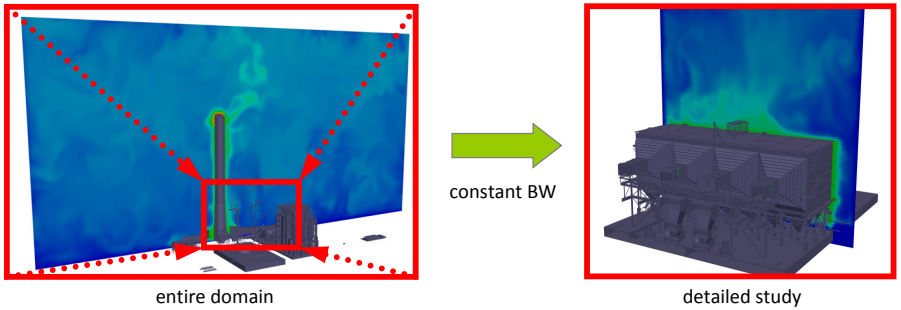
Interactive Visual Data Exploration

- sliding window concept
  - complex part: what happens on the back-end…
  - collector node handles queries and ‘fills’ data stream top-down



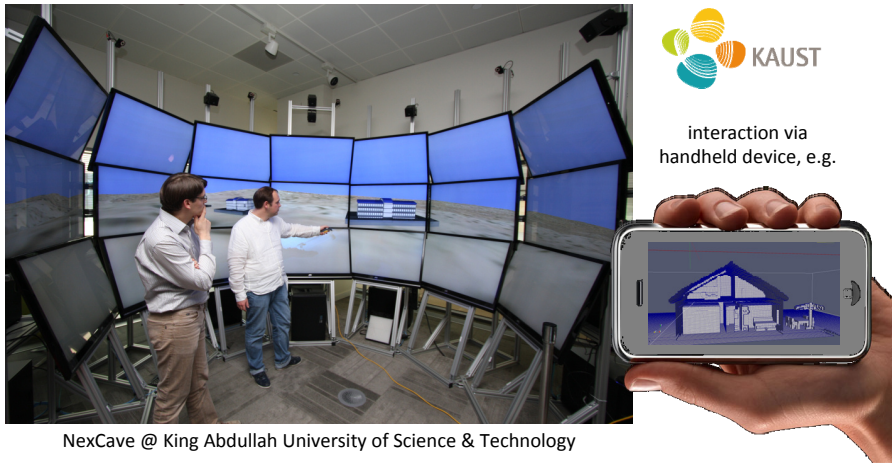
Interactive Visual Data Exploration

- sliding window concept
  - geometric model: power plant (BREP with 12,748,510 faces)
  - user selects window for details interactively during runtime



Interactive Visual Data Exploration

- interactive 3D data exploration: size does matter!



NexCave @ King Abdullah University of Science & Technology

contact: mundani@tum.de

- acknowledgements

KAUST lrz JÜLICH FORSCHUNGSZENTRUM