Energy efficiency aspects of high performance computing for PDEs

Dominik Göddeke (and the FEAST group)

Institut für Angewandte Mathematik (LS3) TU Dortmund dominik.goeddeke@math.tu-dortmund.de

High Performance Computing in Science and Engineering Hotel Soláň, Czech Republic, May 29, 2013

fakultät für 📂





#### Parallelism, specialisation and heterogeneity

- Frequency scaling is over, we now scale 'cores' and SIMD width
- Visible on all architectural levels
  - Fine-grained: SSE/AVX, GPU 'threads'
  - Medium-grained: GPUs, MICs, CPUs, NUMA within a node
  - Coarse-grained: MPI between heterogeneous nodes
- Even CPU-only systems are heterogeneous
- These are fun days for programmers

#### This paradigm shift is here to stay

- Power is the root cause of this
- Goal of this talk: try to explain why, and present a fun experiment with novel architectures

#### Hardware background and some politics

- Why is the hardware becoming so exciting / ugly / painful to program / ...?
- The 'money wall problem' in HPC

#### **Two applications**

- Hardware-oriented finite element multilevel solvers
- Seismic wave propagation modeling with spectral elements

#### Scientific computing on low-power architectures

• 'Energy-to-solution' on an ARM ('smartphone') cluster vs. x86

Hardware background (and some politics)

### Energy consumption of big machines

#### Distribution between various components (source: DOE)

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

#### Three principal attack vectors towards better energy efficiency

- More energy-efficient nodes
  - Much wider fine-grained parallelism
  - Low-power architectures not (initially) designed for HPC
  - Implication: reduced cooling cost, e.g. passive instead of active heatsinks
- More efficient cooling: improved airflow, hot-water cooling, ....
- 'Recycling' of dissipated heat

#### **Current technology**

- Top500 #1 (Green500 #3): Xeon + Kepler
  - 18.6 PFLOP @ 8.2 MW: 2.14 GFLOP/W
- Green500 #1: Xeon + Xeon Phi
  - 112 TFLOP @ 44.9 kW: 2.5 GFLOP/W

#### Extrapolation to exascale

- Top500 machine: 54.8.2=443 MW
- Green500 machine: 400 MW

#### Consequences

- Consensus: power envelope of  $\approx$  20 MW 'feasible'
- Need up to 50 times more energy efficient technology
- Where will this factor come from?

#### Short digression into electrical engineering

- Power is proportional to voltage<sup>2</sup>×frequency
- Frequency is proportional to voltage (at same process size)
- Similarly bad: power also proportional to temperature

#### A simple back-of-the-envelope calculation

	cores	V	freq	perf	power	power eff.
				$\sim f$	$\sim V^3$	perf/power
singlecore	1	1	1	1	1	1
faster singlecore	1	1.5	1.5	1.5	3.3	0.45
dualcore	2	0.75	0.75	1.5	0.8	1.88

#### A-ha!

- 50% more performance with 20% less power
- That's why everything has shifted to multi- and manycore

#### A typical yet hypothetical contemporary processor

- 64-bit FPU: 50 pJ for a multiply-add operation, 0.1 mm<sup>2</sup> of die area
- Moving one DP word through a channel of length 1 mm: 25 pJ
- Chip with 4000 FPUs (area 20 mm<sup>2</sup>): 500 pJ to move one DP word across the chip, e.g., from a pin to an FPU
- Moving the same data off-chip: 1nJ

#### Moving data is expensive in terms of energy

- Factor of 20 for this FPU (60 for two inputs and one output)
- My personal explanation why Xeon Phi doesn't scale in hardware

#### Same holds even more for instructions

 Solution: SIMD, amortise instruction cost (fetch, decode, operand decode, ...) over several data items

#### Ignoring SIMD in computations

- Conventional CPUs: 2-8× penalty (128-256 bit SSE/AVX registers, i.e. 2 doubles to 8 floats per instruction)
- GPUs: 16–64× penalty (32 32 bit values in NVIDIA warps, 64 32 bit values in AMD wavefronts)
- Xeon Phi: 8–16× penalty (512 bit registers)

#### Memory accesses are parallel in hardware since i386 (1985!)

- Cache line granularity of 64 byte on current CPUs
  - Access one float (4 byte) in a memory segment of 64 byte
  - 64 byte transferred
  - Never use the other 15 floats: 1/16 of peak memory performance
- NVIDIA GPUs: Cache lines of 32 byte and 128 byte
- Plus: computation is *much* faster than memory access

#### A simple rule of thumb

- 1 MW/a = 1 M EUR/a
  - Depends on your deal with the local energy company

#### Real numbers: our machine in Dortmund

- Deployed in 2009 (#249 in the TOP500) for roughly 1 M EUR
- 440 dual-quadcore nodes, 50 TFLOP/s, designed for capacity
- Roughly 85 % average load per year
- Annual electricity bill 220 K EUR (plus 60 K EUR for A/C)
- Accumulated running cost now exceeds initial acquisition cost

#### Who covers these costs?

Funding agencies surely don't

FEAST and ScaRC FEM-multigrid for large systems

#### General sparse matrices (on unstructured grids)

- CSR (and variants): general data structure for arbitrary grids
- Maximum flexibility, but during SpMV
  - Indirect, irregular memory accesses
  - Index overhead reduces already low arithm. intensity further
- Performance depends on nonzero pattern (DOF numbering)

#### Structured matrices

- Example: structured grids, suitable numbering ⇒ band matrices
- Important: no stencils, fully variable coefficients
- Direct regular memory accesses (fast), mesh-independent performance
- Structure exploitation in the design of MG components

### Approach in FEAST

#### **Combination of respective advantages**

- Global macro-mesh: unstructured, flexible
- Local micro-meshes: structured (logical TP-structure), fast
- Important: structured ≠ cartesian meshes (r-adaptivity)
- Reduce numerical linear algebra to sequences of operations on structured data (maximise locality)
- Developed for larger clusters (later), but generally useful: cache-friendly, locality-maximising



#### Test problem

- $-\Delta u = 1$ , Q1 FEM, cylinderflow grid
- Inhomogeneous Dirichlet boundary conditions
- Multigrid solver

#### Large parameter space

- ELLR-T vs. FEAST matrix format
- ELLR-T with two different sortings (random and lexicographic)
- Jacobi vs. strong smoother (unstructured SPAI or ADI-TRIGS)
- Westmere Corei7-X980 vs. Fermi C2070



### Incremental improvements by hardware-oriented numerics



 Largest to smallest contribution: GPU, structured, smoother, numbering Disclaimer: Not necessarily a good smoother, but a good didactical example.

#### Sequential algorithm

- Forward elimination, sequential dependencies between matrix rows
- Illustrative: coupling to the left and bottom

#### 1st idea: wavefronts / pipelining (exact)



- Pro: always works to resolve *explicit* dependencies
- Con: irregular parallelism and access patterns, implementable?

### Gauß-Seidel smoother

#### Better idea: multicolouring (inexact)

- Always use all already available results
- Jacobi (red) coupling to left and right (green) top and bottom (blue) – all 8 neighbours (yellow)



- Parallel efficiency: 4 sweeps with  $\approx N/4$  parallel work each
- Regular data access, but checkerboard pattern challenging for SIMD/GPUs due to strided access (can be solved)
- Numerical efficiency: no longer naturally numbered

### ScaRC: coarse-grained parallel multigrid

#### ScaRC for scalar systems

- Hybrid multilevel domain decomposition method
- Minimal overlap by extended Dirichlet BCs
- Inspired by parallel MG ('best of both worlds')
  - Multiplicative between levels, global coarse grid problem (MG-like)
  - Additive horizontally: block-Jacobi / Schwarz smoother (DD-like)
- Schwarz smoother encapsulates local irregularities
  - Robust and fast multigrid ('gain a digit'), strong smoothers
  - Maximum exploitation of local structure





Large-scale seismic wave propagation modeling

#### Strong form of the wave equation as IVP

$ ho \partial_t^2 u$	=	$\nabla \cdot \sigma + f$	in $\Omega  imes T$
$\sigma \cdot n$	=	$\overline{t}$	on $\Gamma_N$
u	=	g	on $\Gamma_D$
$u _{t=0}$	=	$u_0$	$\forall x\in \Omega$
$\partial_t u _{t=0}$	=	$u_1$	$\forall x\in\Omega$

- Density  $\rho$ , acceleration  $\partial_t^2 u$ , source terms f
- $\blacksquare$  Constitutive law to relate strains and stresses  $\sigma$
- $\blacksquare$  Body forces (inner forces)  ${\rm div}(\sigma)$  caused by the displacement

#### Can be made arbitrarily complex

- Solids and liquids (outer core, oceans), strongly heterogeneous
- Coriolis force, self-gravitation, PML conditions (chunks of the Earth)

#### Discretisation trade-offs

- (Pseudo-) spectral methods
  - Pro: accuracy and exponential approximation order
  - Con: parallelisation, boundary conditions
- Finite element methods
  - Geometric flexibility (topographic details, faults and fractures, internal discontinuities and material transitions, ...), boundary conditions (parallelisation)
  - Possibly ill-conditioned systems, high-order expensive (wide supports)

#### SEM as a hybrid approach

- Essentially based on high-order interpolation to represent fields per element of a classical unstructured curvilinear FEM mesh
- Good compromise between conflicting goals
- Can be interpreted as a continuous Galerkin method

### Cubed sphere unstructured mesh



#### Discretise the weak form

Same as in finite element methods

#### Base everything on Lagrange polynomials

- Shape functions for each curvilinear element (order 2)
- Interpolation of physical fields inside each element (order 4–10)
- Gauß-Lobatto-Legendre control and cubature points
  - Combination yields strictly diagonal mass matrix
  - Explicit time stepping feasible, trivial 'solving' of linear systems
  - Drawback: timestep depends on mesh spacing

#### Problem to be solved in algebraic notation

- $\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{f}$  for waves in simple solids
- $\blacksquare$  Mass matrix  $\mathbf{M}$ , stiffness matrix  $\mathbf{K}$
- Displacement vector  ${f u}$ , sources  ${f f}$ , velocity  ${f v}={\dot {f u}}$ , acceleration  ${f a}={\ddot {f u}}$

#### Three phases in each second-order Newmark timestep

- Compute new Ku and M (the tricky bit, called 'SEM assembly') to obtain intermediate acceleration

#### Steps 1 and 3 are trivial

- Simple operations on globally numbered data
- Memory-bound on almost all architectures

#### Step 2 is tricky

- Typically, around 90% of the total runtime
- Employ 'assembly by elements' technique on CPUs and GPUs
- Parallelism via colouring or atomics, depends on hardware
- Standard non-blocking MPI
  - On GPUs: some basic tricks to hide additional CPU-GPU transfers
- Well-balanced on GPUs, compute-bound everywhere else

# Energy efficiency Case study on an ARM cluster

#### **Smartphones and tablets**

- Tremendous market volume
- Probably a couple hundred GFLOP/s in this room
- Powerful enough for HD video decoding, mobile gaming, ....

#### Under the hood

- Low-power processor designs stemming from embedded systems now capable enough for HPC: dedicated FPUs, multicores, vector registers, reasonably-sized pipelines ...
- Almost all these chips contain IP by ARM Ltd., e.g. Apple A4/A5, ARM Cortex-A series, Samsung, ...
- Batteries matter, so very energy-efficient

#### Can we use these chips to solve real problems?

#### In fact, a tricky question for application people

- Flops/Watt not meaningful at application level
- Slower units more energy-efficient even if it takes longer?
- Many more units to reach same speed still more energy-efficient?
- Time-to-solution and energy-to-solution!

#### Weak and strong scaling equally critical

- Weak: need many more units because of much lower memory/node
- Strong: need many more units to compensate for slower per-node execution
- Application-specific sweet spots, how much slower can we afford to be if it cuts our electricity bill in half?

#### Machine details

- 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 under 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 strategies counterproductive, bookkeeping and index overhead for deeper nested loops
- Serial codes only reach half of the memory bandwidth per node!



- Weak (left) and strong (right) scaling
- Weak scaling as expected (bump is a granularity effect from 4 to 5 solver iterations)
- Strong scaling quite ok (gigE and a multilevel method)

### Power-performance analysis: FEAST



- Speedup x86 over ARM (left), improvement energy-to-solution ARM over x86 (right)
- Always more beneficial to use the ARM cluster
- As long as  $\geq 2$  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

### Scalability: SPECFEM3D\_GLOBE



Weak (left) and strong (right) scaling

Both perfect (explicit in time, compute-bound, non-blocking MPI)

### Power-performance analysis: SPECFEM3D\_GLOBE



- Speedup x86 over ARM (left), improvement energy-to-solution ARM over x86 (right)
- Not always more beneficial to use the ARM cluster due to many small dense matrix matrix multiplications

## **Summary**

- Power is the root cause of current and mid-term hardware evolution
- Energy efficiency will become the ultimate quality criterion
- Energy efficiency is quite hard to define exactly
- Weak and strong scaling essential even for moderate-size problems to compensate for slower execution
- Low-power architectures not always a win in terms of 'energy-to-solution'

#### Collaborative work with

- M. Geveler, D. Ribbrock (TU Dortmund)
- D. Komatitsch (CNRS Marseille)
- A. Ramirez, N. Rajovic, N. Puzovic (Barcelona Supercomputing Center)

#### Papers, links, tutorials and other stuff...

http://www.mathematik.tu-dortmund.de/~goeddeke

#### Funding

- DFG Priority Program 1648: 'Software for Exascale Computing'
- NVIDIA Teaching Center Program

### More acknowledgments

### Acknowledgment: today's Peta-op/s machines



### 10<sup>12</sup> neurons @ 1 KHz = 1 PetaOp/s \*\*1.4 kilograms, 20 Watts \*\*

[Shamelessly stolen from David Keyes]