Energy efficiency of the simulation of three-dimensional coastal ocean circulation on modern commodity and mobile processors

A case study based on the Haswell and Cortex-A15 microarchitectures

Dominik Göddeke

University of Stuttgart
Germany

SimTech
Cluster of Excellence

<u>Markus Geveler,</u>
Stefan Turek

technische universität
dortmund

fakultät für
mathematik

NVIDIA
GPU
EDUCATION
CENTER

I.C.A.R.U.S.

Balthasar Reuter,
Vadym Aizinger

FAU
FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
NATURWISSENSCHAFTLICHE
FAKULTÄT

Department
MATHEMATIK

EnA-HPC, ISC, Frankfurt, 2016 / 6 / 23

markus.geveler@math.tu-dortmund.de

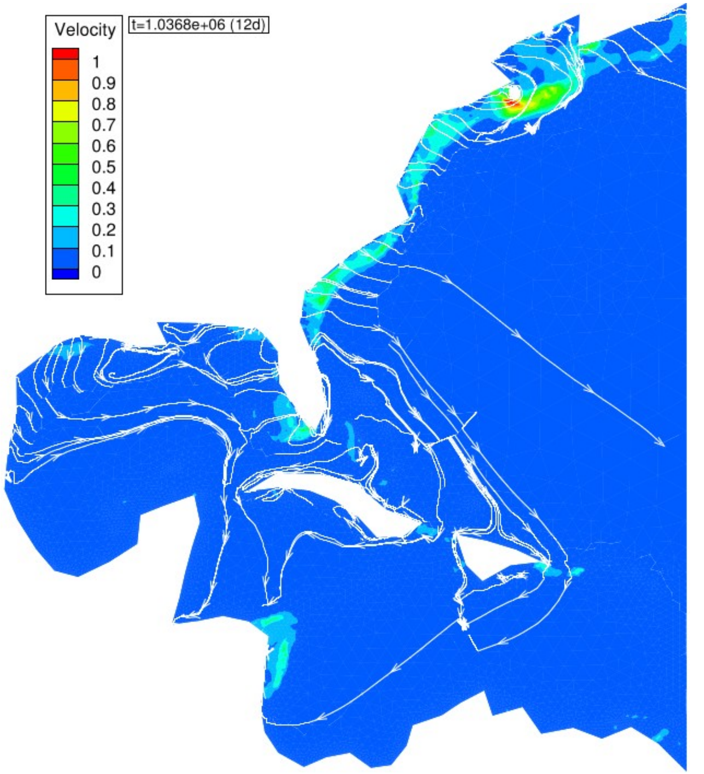
Outline

How can we take control of *energy to solution* in simulation software?

→ expectations and misconceptions
→ modeling of energy
→ control
→ **unconventional hardware for more energy-efficiency**

Concrete example: Coastal ocean circulation simulation

→ high-end 3D geophysical flow dynamics
→ **on Intel Haswell and ARM Cortex-A15 processors**



HPC Hardware

Today's HPC facilities (?)

www.green500.org Green500 list Nov 2015

Green 500 rank	Top 500 rank	Total power [kW]	MFlops per watt	Year	Hardware architecture
1	133	50	7031	2015	ExaScaler-1.4 80Brick, Xeon E5-2618Lv3 8C 2.3GHz, Infiniband FDR, **PEZY-SC**
2	392	51	5331	2013	LX 1U-4GPU/104Re-1G Cluster, Intel Xeon E5-2620v2 6C 2.1GHz, Infiniband FDR, **NVIDIA Tesla K80**
3	314	57	5271	2014	ASUS ESC4000 FDR/G2S, Intel Xeon E5-2690v2 10C 3GHz, Infiniband FDR, **AMD FirePro S9150**
4	318	65	4778	2015	Sugon Cluster W780I, Xeon E5-2640v3 8C 2.6GHz, Infiniband QDR, **NVIDIA Tesla K80**
5	102	190	4112	2015	Cray CS-Storm, Intel Xeon E5-2680v2 10C 2.8GHz, Infiniband FDR, **Nvidia K80**
6	457	58	3857	2015	Inspur TS10000 HPC Server, Xeon E5-2620v3 6C 2.4GHz, 10G Ethernet, **NVIDIA Tesla K40**
7	225	110	3775	2015	Inspur TS10000 HPC Server, Intel Xeon E5-2620v2 6C 2.1GHz, 10G Ethernet, **NVIDIA Tesla K40**

No Top500 top-scorer

Top500 #1: 17,000 kW, 1,900 MFlop/s/W

unconventional hardware

HPC Hardware

Today's HPC facilities

→ comprise heterogeneous compute nodes
→ multicore CPU(s) + some accelerator very common (GPU, XEON Phi)
→ heterogeneity on-a-chip (SoCs, APUs)
→ **cost efficiency dominated by energy-efficiency**

Today's large-scale HPC codes

→ have to adapt to target hardware
→ heterogeneity and frameworking
→ parallelisation of applications (DD mostly)
→ parallelisation of core components (e.g. 'linear solver on GPU')
→ optimisation with respect to many details (data flow and SIMD mostly)
→ **can we have the same results with less energy-consumption?**

Hardware evolution is (usually) out of our control – hardware-choice and software-design are not

Total efficiency of simulation software

Aspects

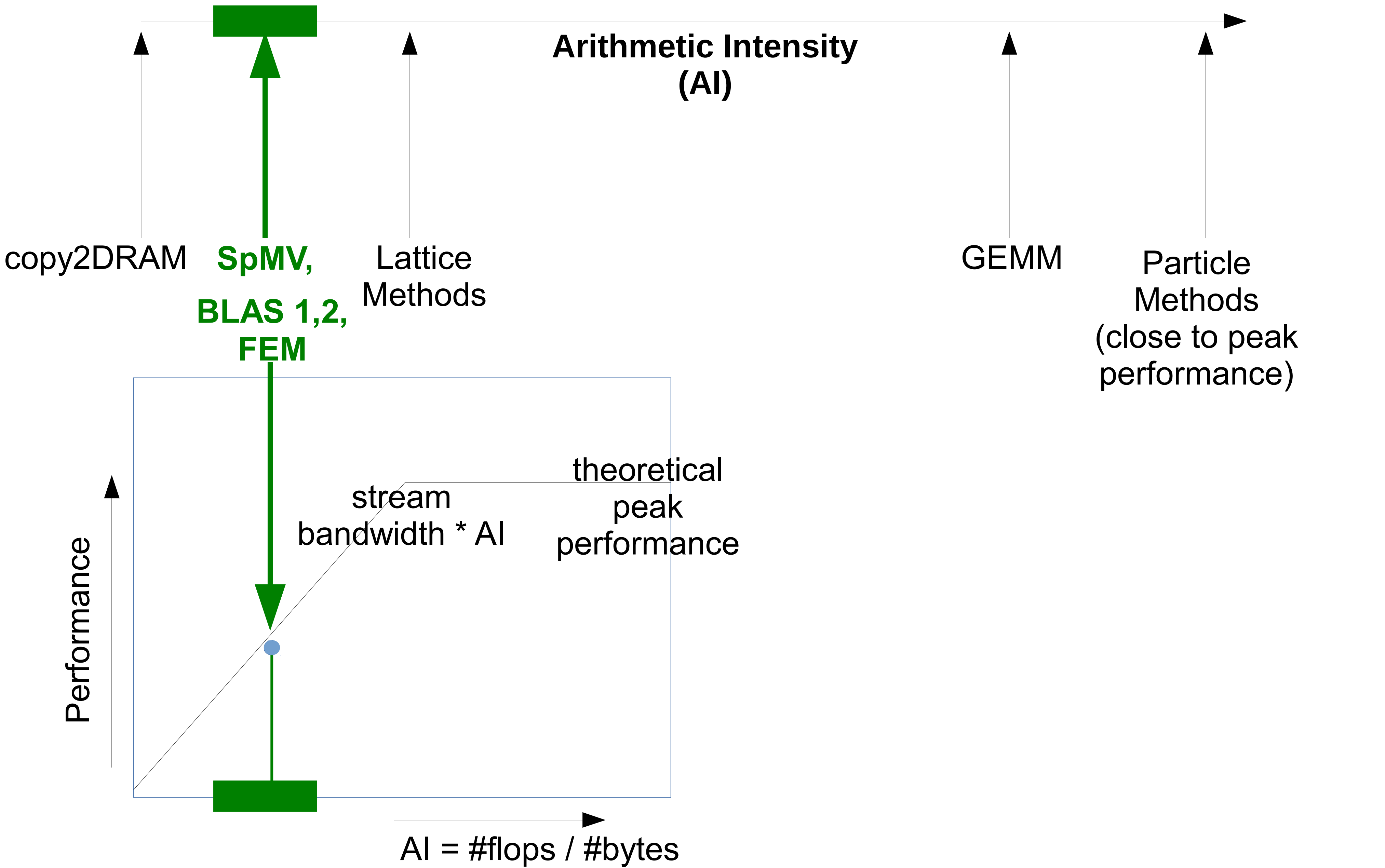
→ **Numerical efficiency** dominates asymptotic behaviour and wall clock time

→ **Hardware-efficiency**
→ exploit all levels of parallelism provided by hardware (SIMD, multi-threading on a chip/device/socket, multi-processing in a cluster, hybrids)
→ then try to reach good scalability (communication optimisations, block comm/comp)

→ **Energy-efficiency**
→ by hardware:
→ what is the most energy-efficient computer hardware? What is the best core frequency? What is the optimal number of cores used?
→ by software as a direct result of performance
→ but: its not all about performance

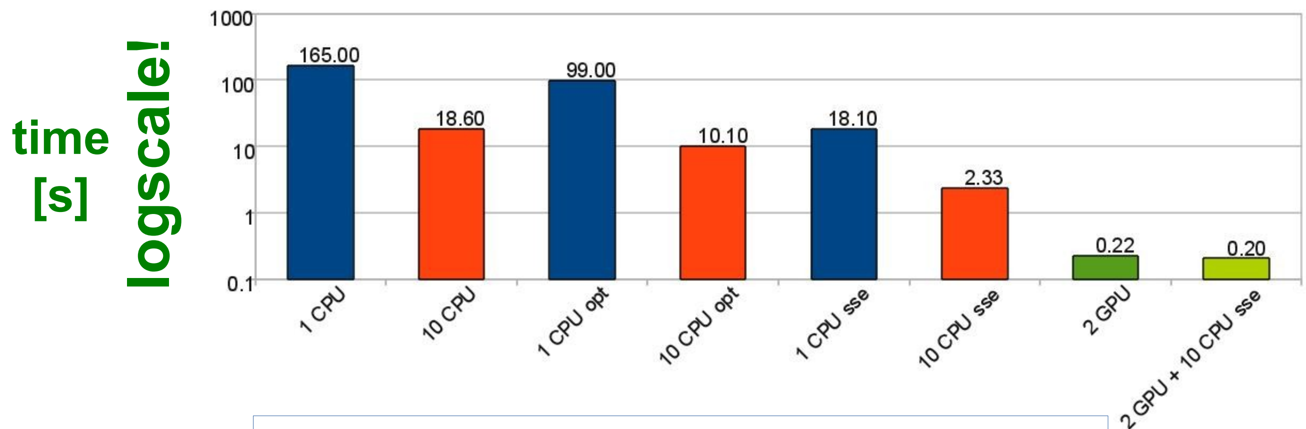
Hardware-oriented Numerics: Enhance hardware- and numerical efficiency simultaneously, use (most) energy-efficient Hardware(-settings) where available! Attention: codependencies!

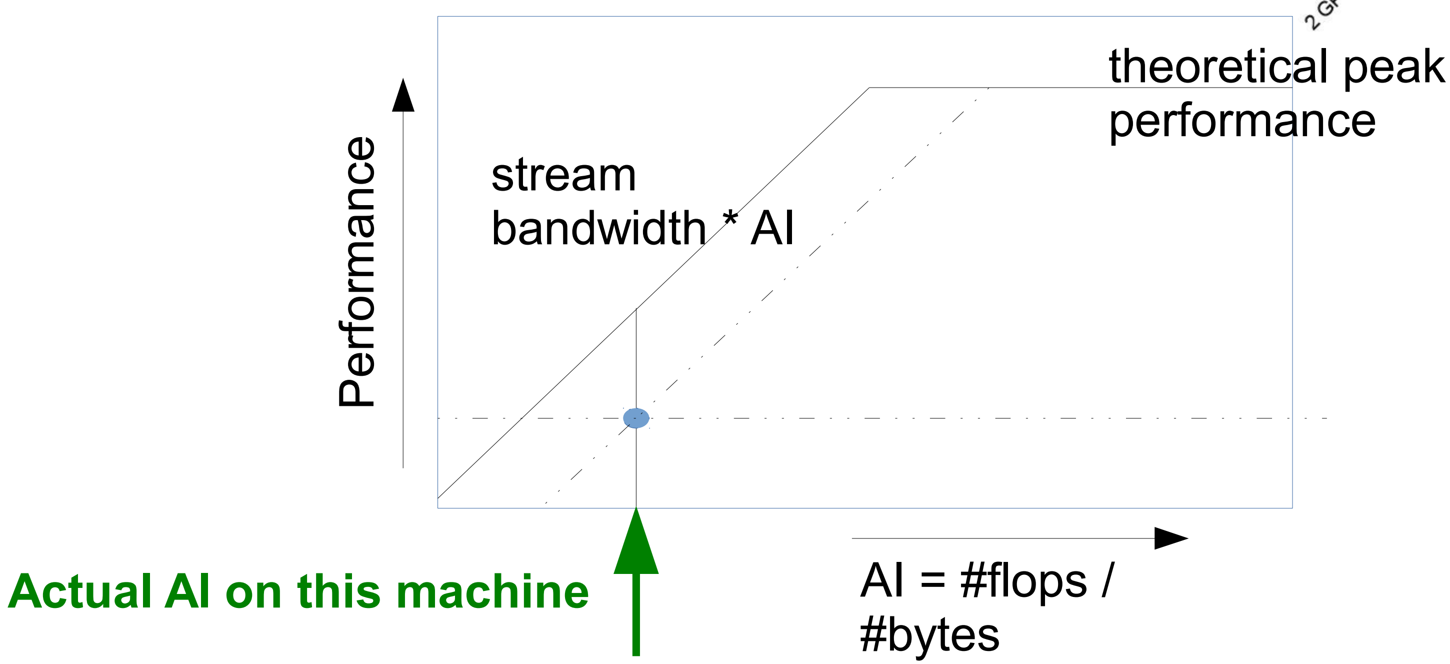
What we can expect from hardware



Hardware-oriented Numerics

Hardware Efficiency: apply 'classical' roofline models until optimal

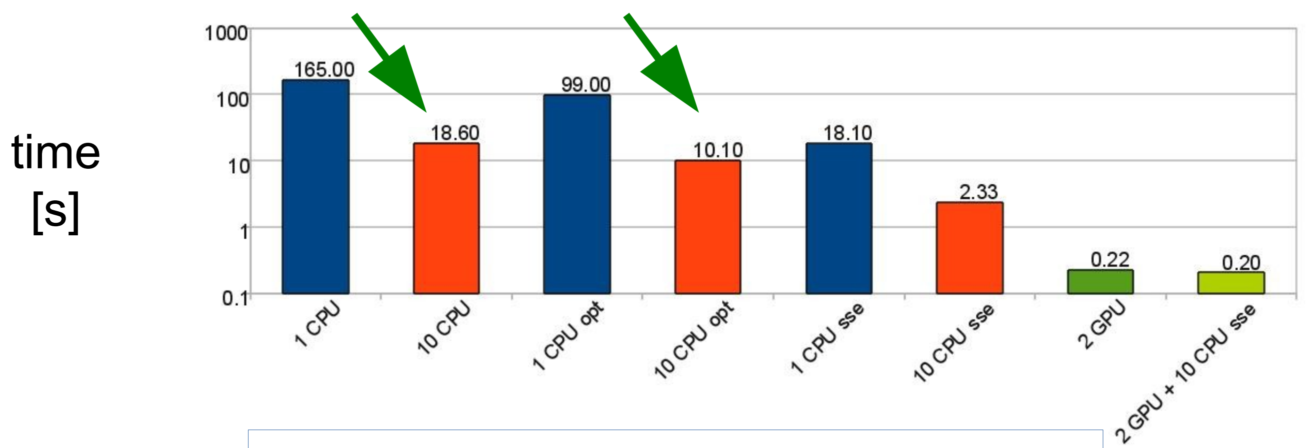


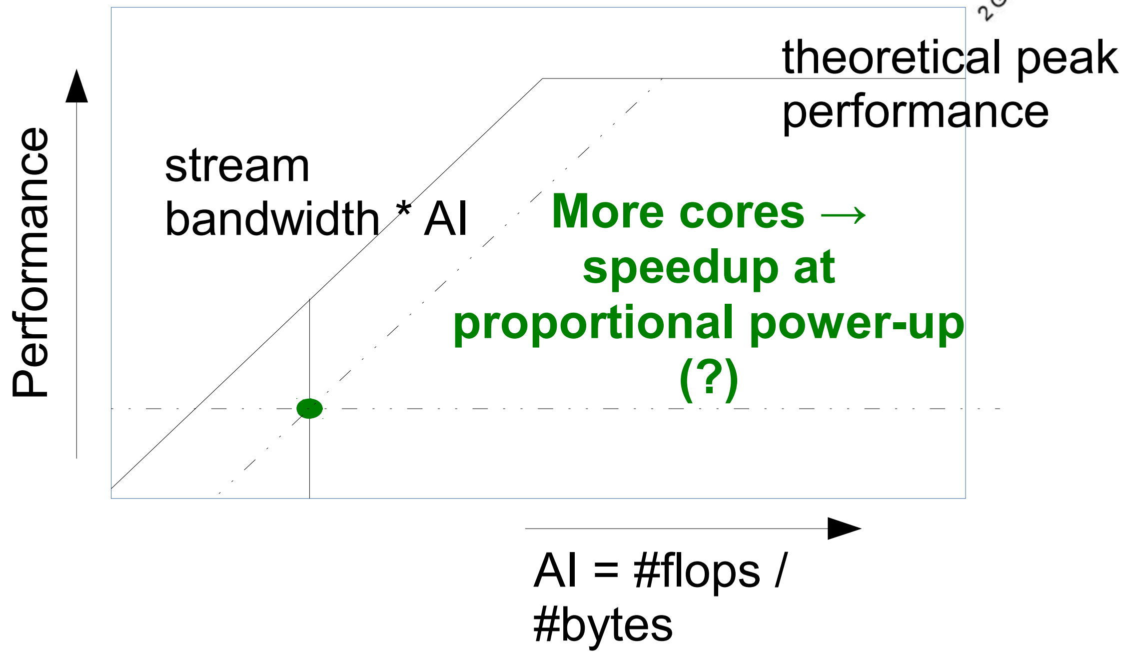


Hardware-oriented Numerics

Hardware Efficiency: apply 'classical' roofline models until optimal

the 'good scaling trap'

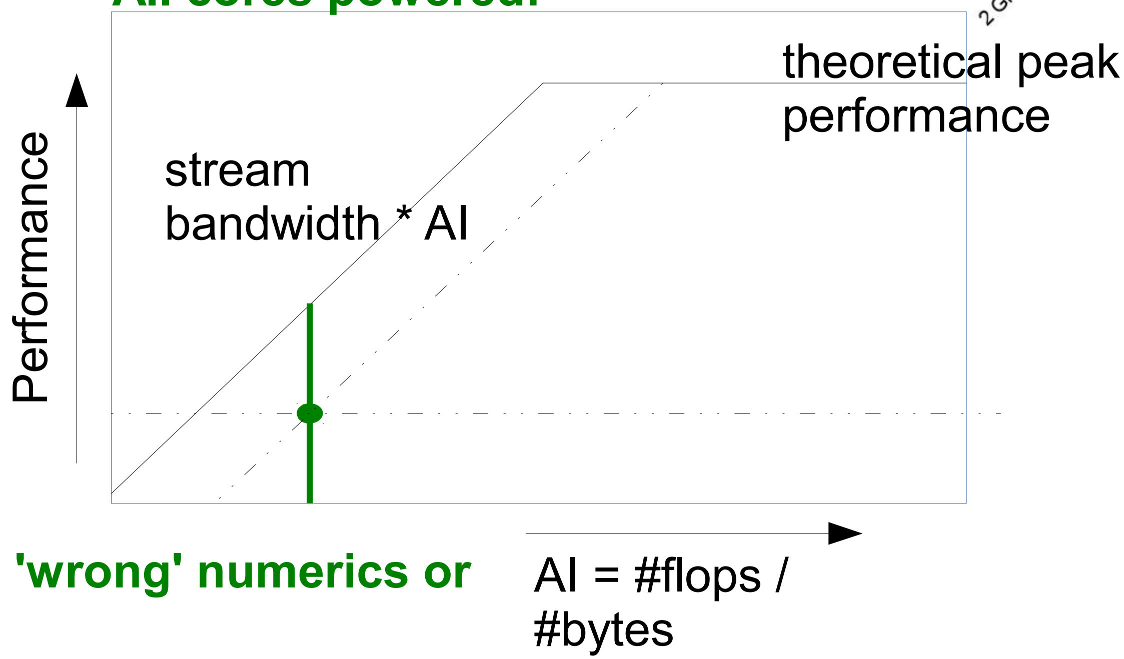




Performance-penalties

Hardware Efficiency: apply 'classical' roofline models until optimal

All cores powered!

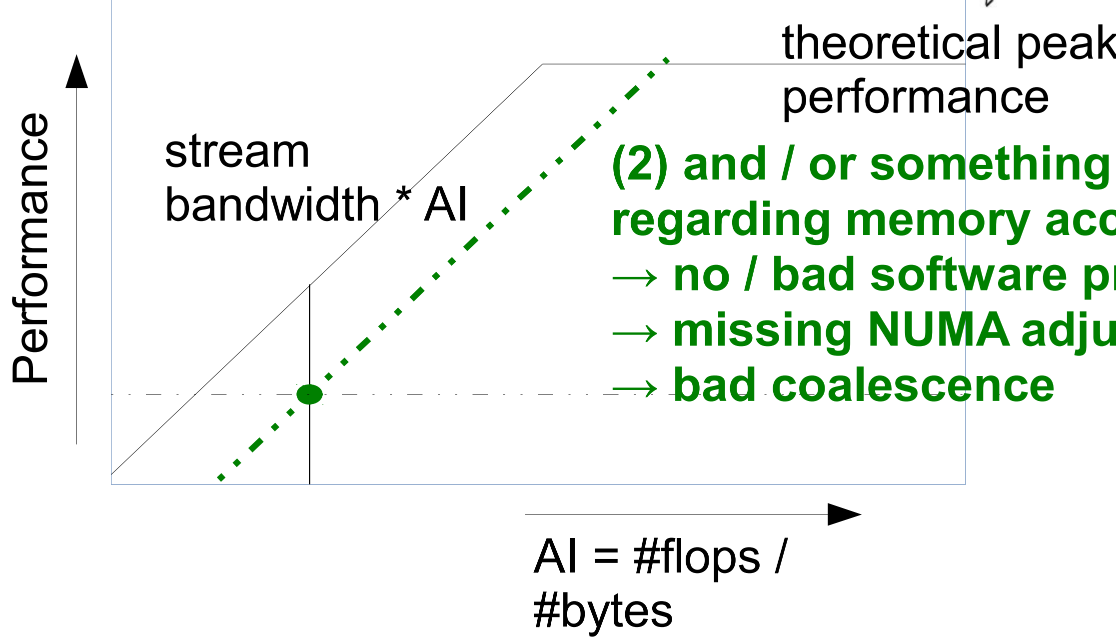


(1) bad AI due to 'wrong' numerics or implementation

Performance-penalties

Hardware Efficiency: apply 'classical' roofline models until optimal

All cores powered!

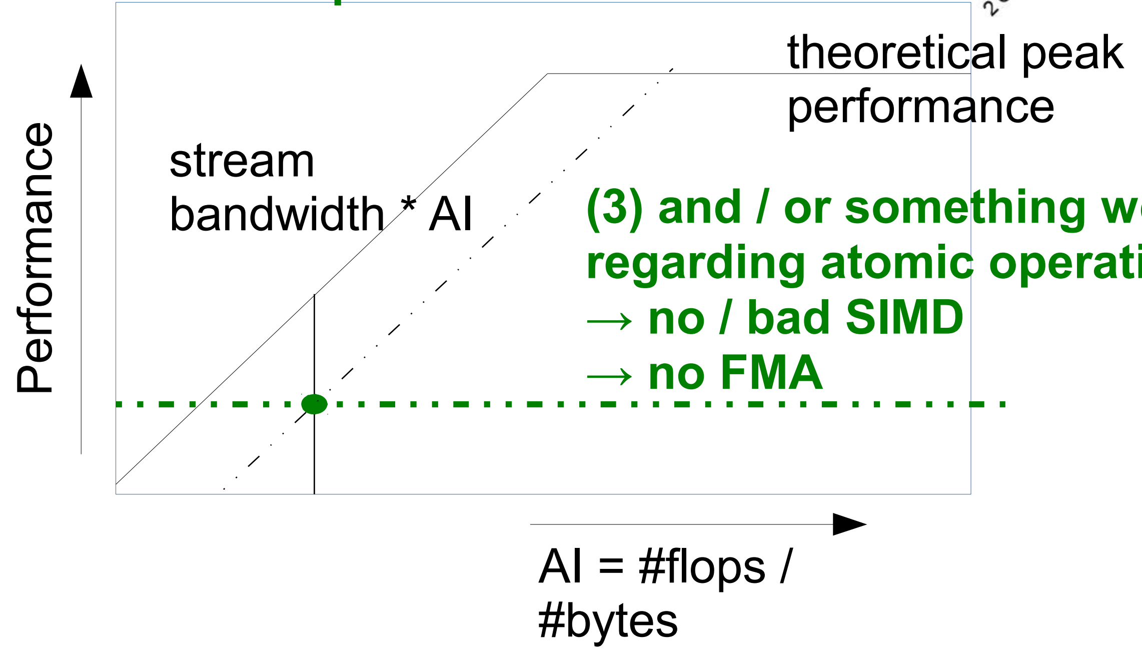


(2) and / or something went wrong regarding memory access patterns:
→ no / bad software prefetching
→ missing NUMA adjustments
→ bad coalescence

Performance-penalties

Hardware Efficiency: apply 'classical' roofline models until optimal

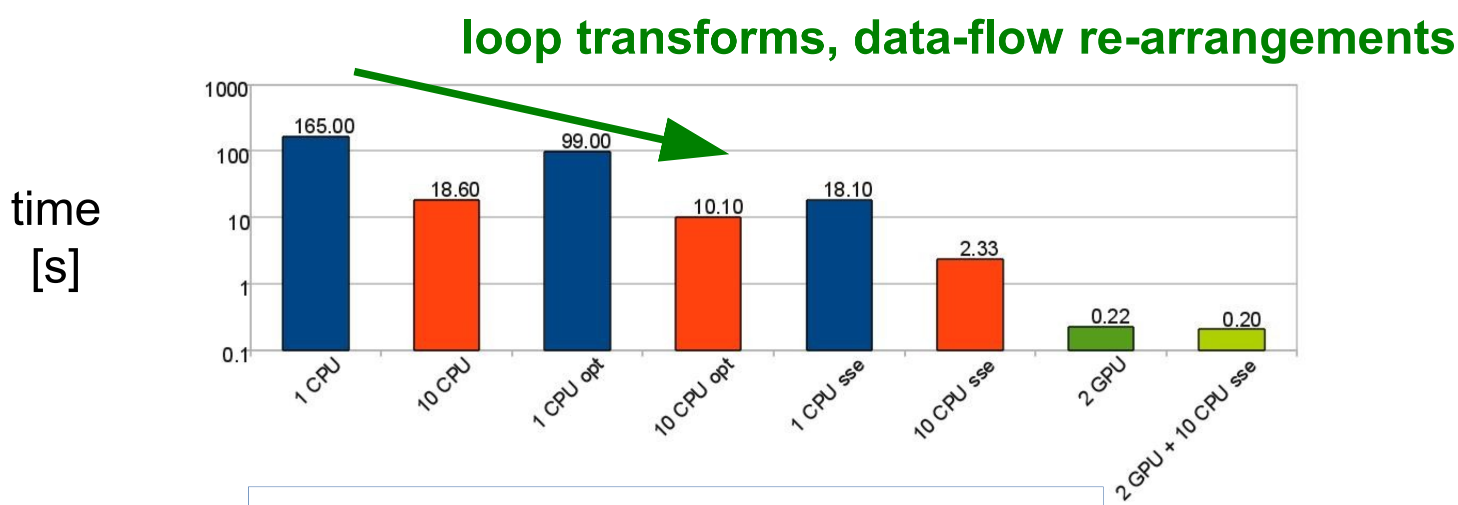
All cores powered!

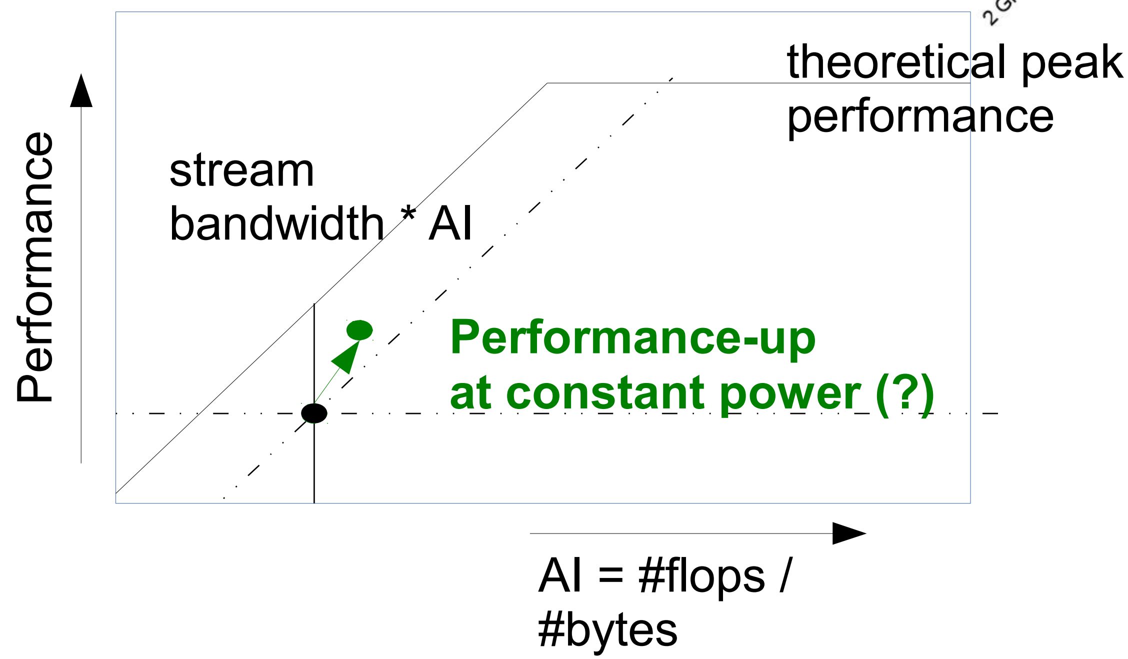


(3) and / or something went wrong regarding atomic operations, most likely:
→ no / bad SIMD
→ no FMA

Optimization

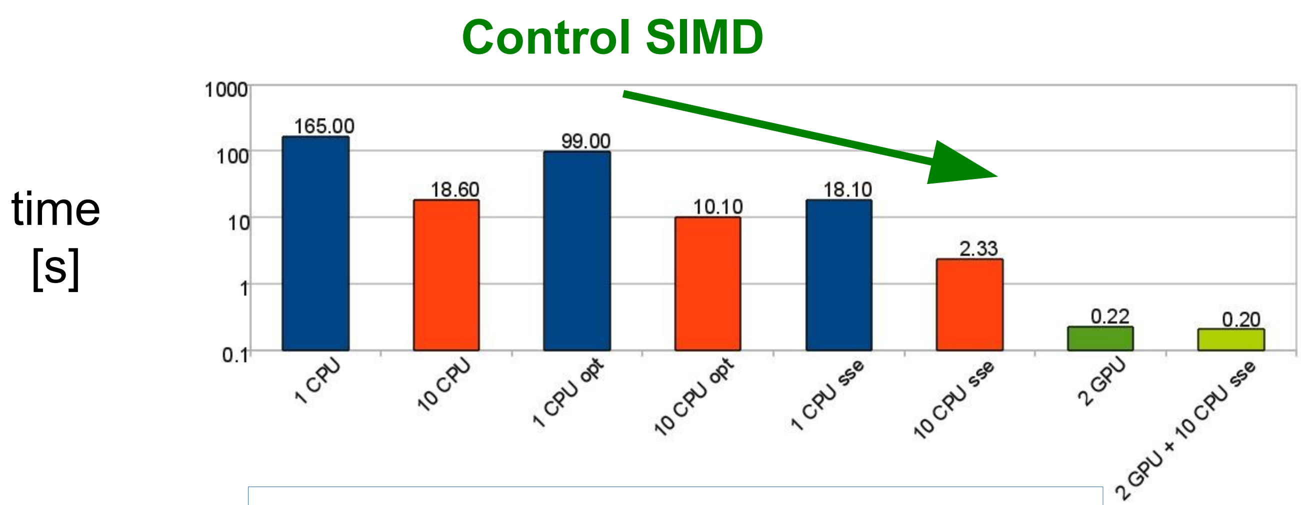
Hardware Efficiency: apply 'classical' roofline models until optimal

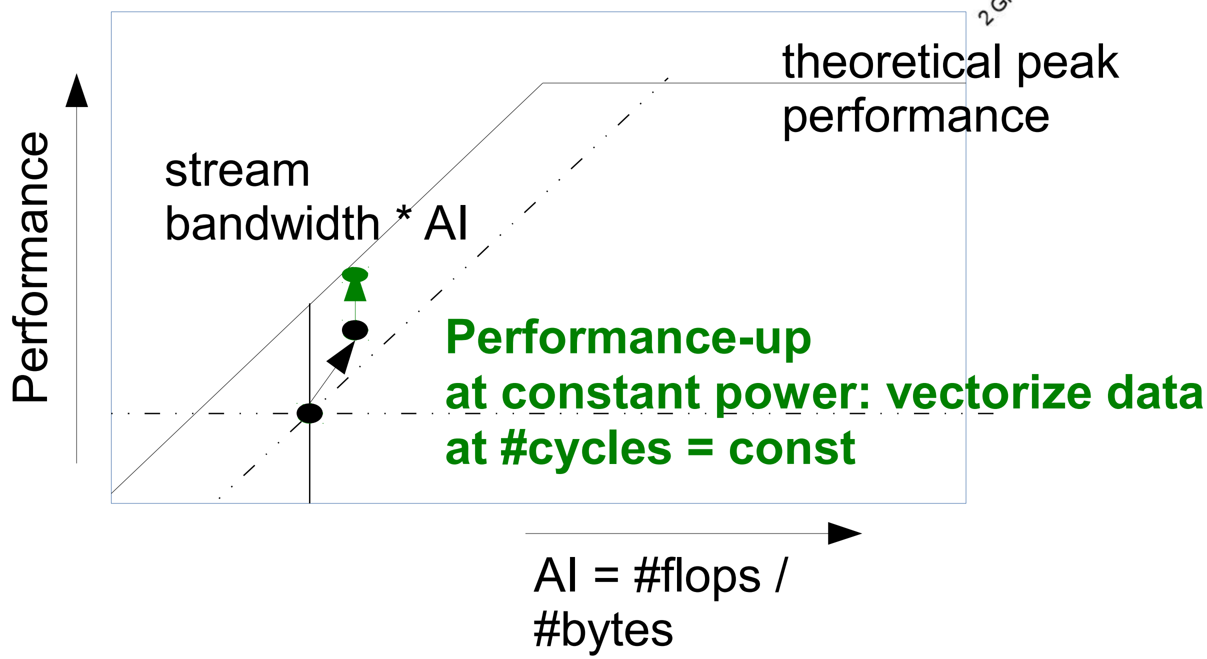




Optimization

Hardware Efficiency: apply 'classical' roofline models until optimal

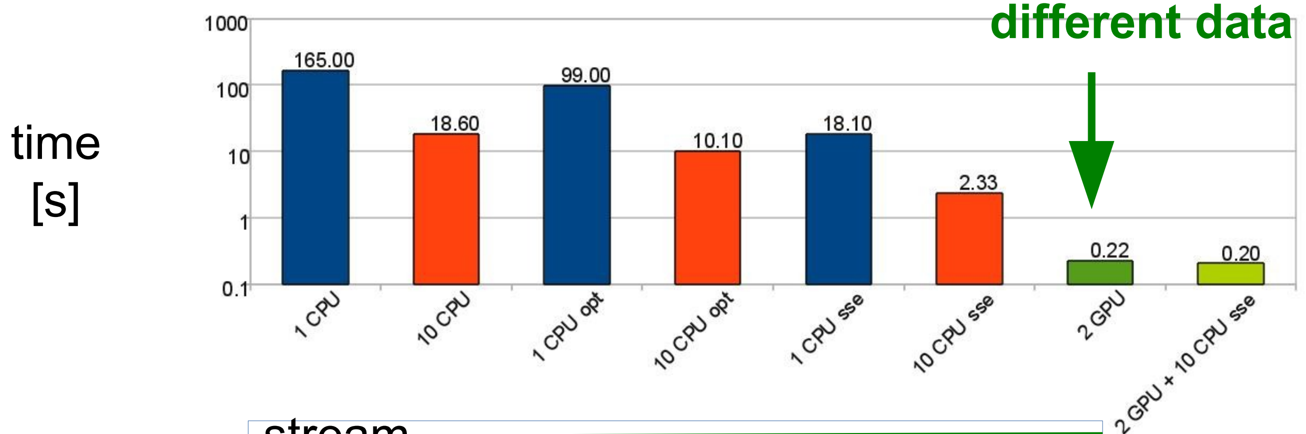


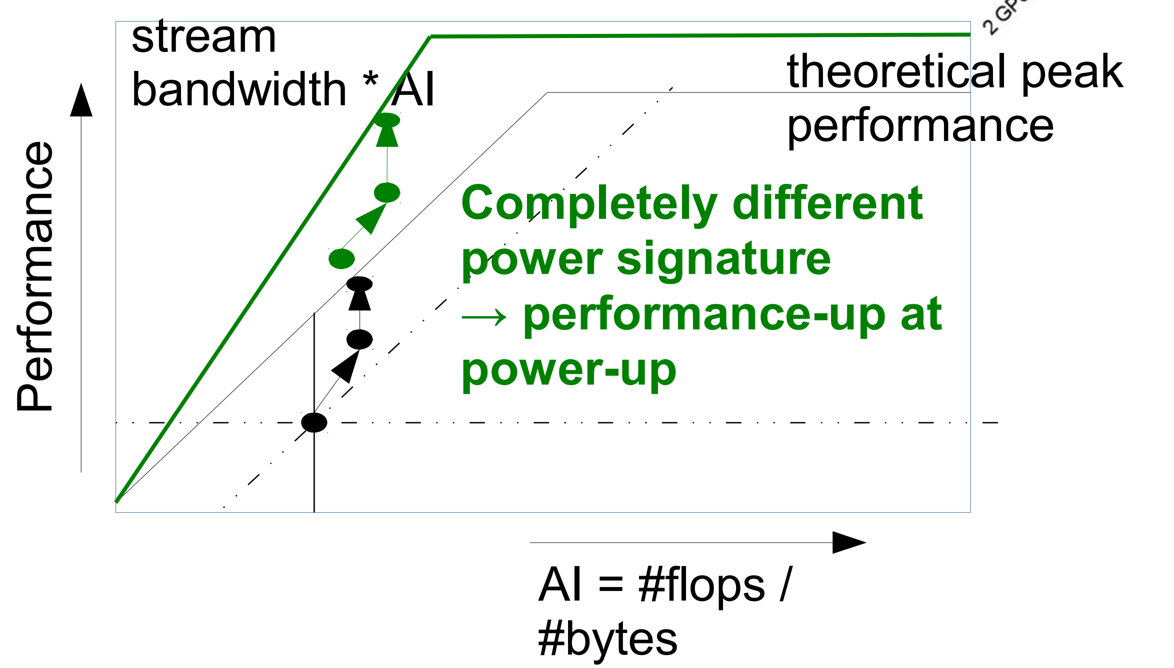


Optimization

Hardware Efficiency: apply 'classical' roofline models until optimal

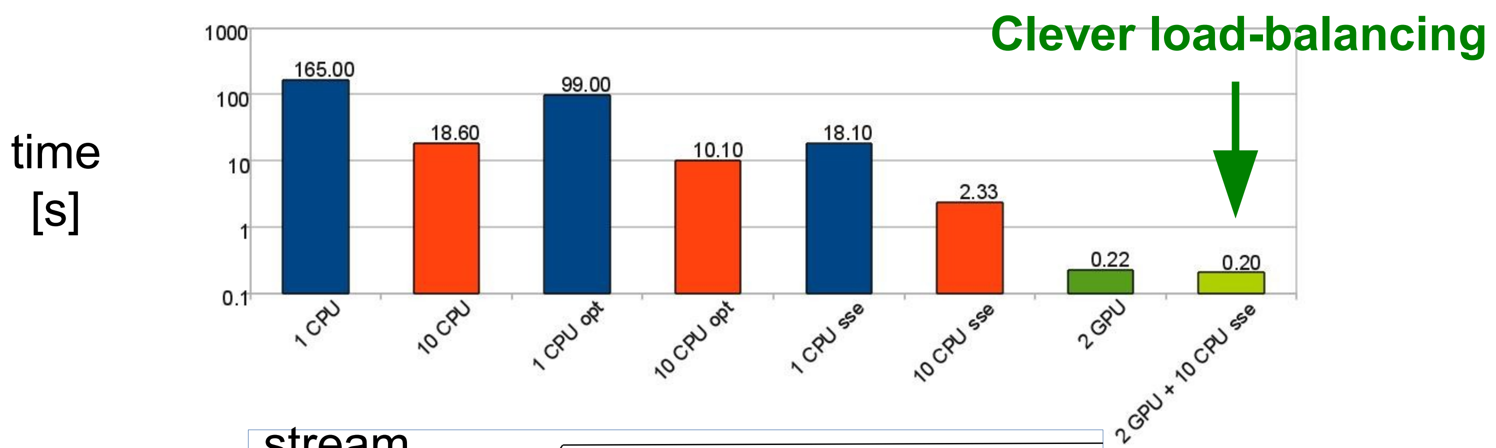
Use accelerator:
new perfmodel,
different data storage

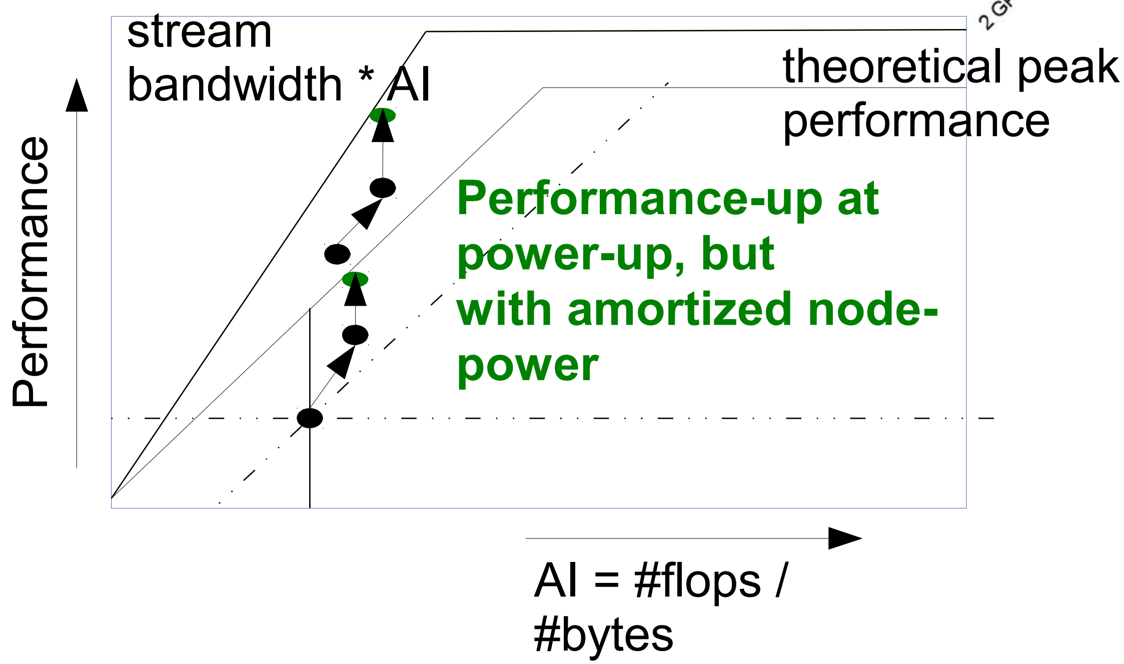




Optimization

Hardware Efficiency: apply 'classical' roofline models until optimal



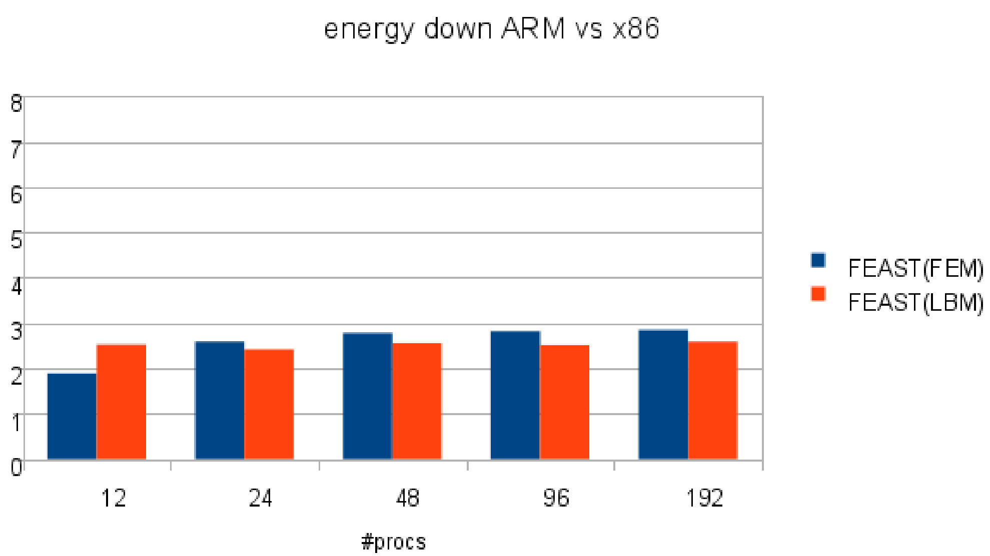


Hardware-oriented Numerics

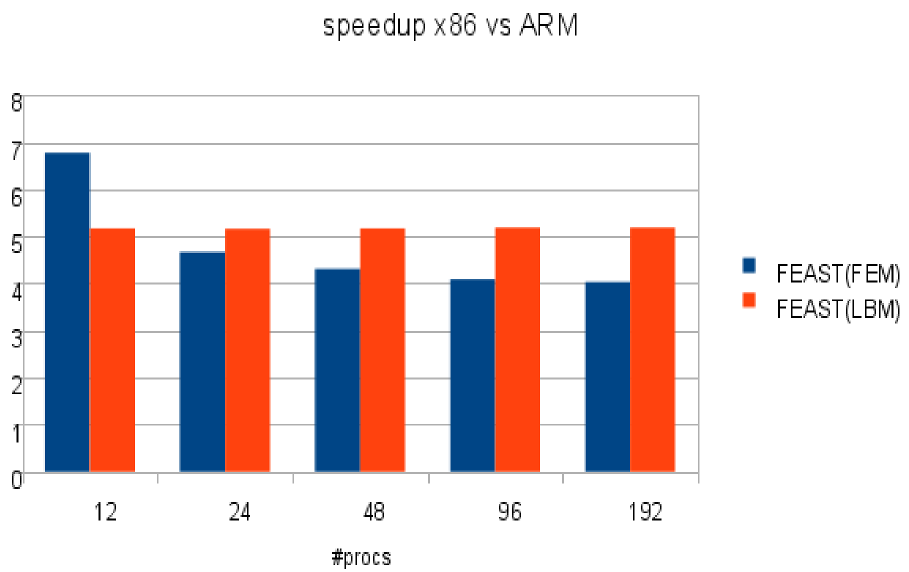
Energy Efficiency

- energy consumption/efficiency is one of the major challenges for future supercomputers → 'exascale'-challenge

- in 2012 we proved: we can solve PDEs for less energy 'than normal'
- simply by switching computational hardware from commodity to embedded
- Tegra 2 (2x ARM Cortex A9) in the Tibidabo system of the MontBlanc project
- tradeoff between energy and wall clock time



~3x less energy



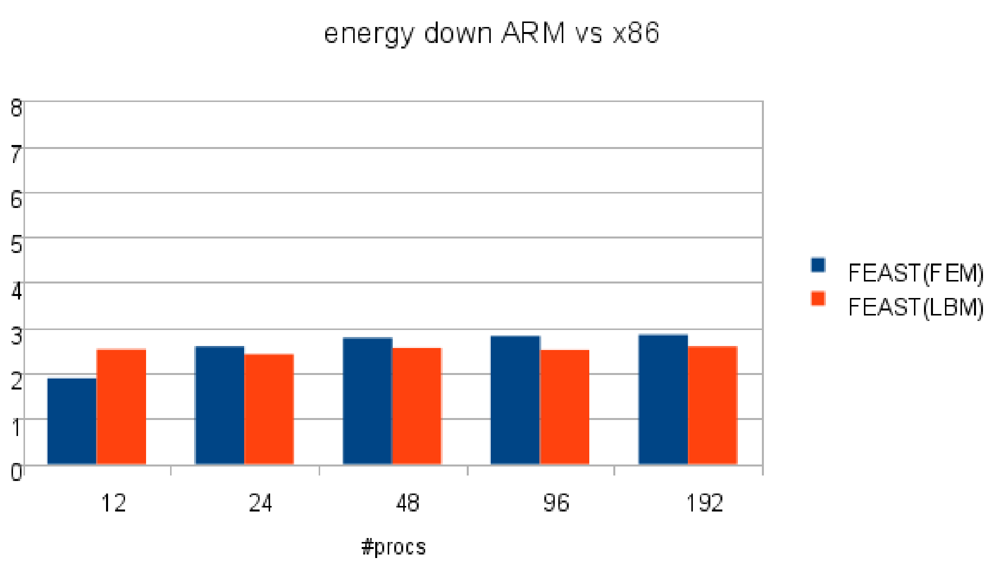
~5x more time!

Hardware-oriented Numerics

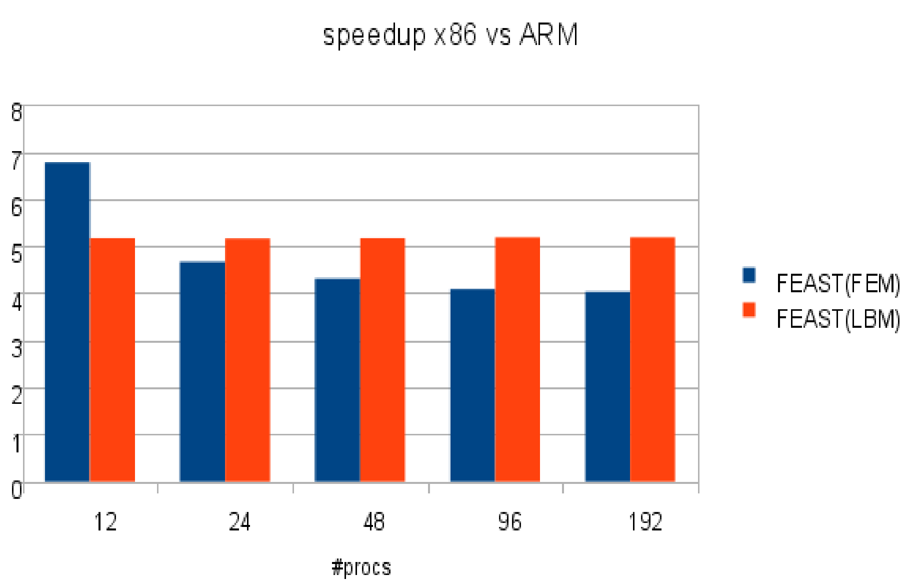
Energy Efficiency

To be more energy efficient with different computational hardware, this hardware would have to dissipate *less power* at the *same performance* as the other!

→ More performance per Watt! → power-down > speed-down



~3x less energy



~5x more time!

Hardware-oriented Numerics

Energy Efficiency: technology of ARM-based SoCs since 2012

Something has been happening in the mobile-computing hardware evolution:

→ Tegra 3 (late 2012) was also based on A9 but had 4 cores
→ Tegra 4 (2013) is build upon the A15 core (higher frequency) and had more RAM and LPDDR3 instead of LPDDR2
→ Tegra K1 (32 Bit, late 2014) CPU pretty much like Tegra 4 but higher freq., more memory

→ TK1 went GPGPU and comprises a programmable Kepler GPU on the same SoC!
→ the promise: 350+ Gflop/s for less than 11W
→ for comparison: Tesla K40 + x86 CPU: 4200 Gflop/s for 385W

→ 2.5x higher EE promised

→ interesting for Scientific Computing! Higher EE than commodity accelerator!

3D baroclinic Shallow Water Equations

Elevation

$$\partial_t h + \nabla \cdot \int_{z_b}^{\xi} \mathbf{u}_{xy} \mathrm{d}z \; = \; 0$$

Continuity

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

Momentum

$$\partial_t \mathbf{u}_{xy} + \nabla \cdot (\mathbf{u}_{xy} \otimes \mathbf{u}) - \partial_z (\nu_t \partial_z \mathbf{u}_{xy}) + \nabla_{xy}(gh + p) - f_c \mathbf{k} \times \mathbf{u}_{xy} = \mathbf{F}_u - \nabla_{xy} z_b$$

Temperature-/ salinity-transport

$$\partial_t r + \nabla \cdot (\mathbf{u} r) \; - \; \partial_z \left(\nu_r \partial_z r\right) \; = \; F_r$$

Turbulent quantities transport

$$\partial_t m - \partial_z \left(\nu_m \partial_z m\right) \; = \; F_m$$

Density forcing

$$p(x, y, z) = \frac{g}{\rho_0} \int_z^{\xi} \left(\rho(\theta(x, y, \tilde{z}), s(x, y, \tilde{z})) - \rho_0\right) \tilde{z}$$

UTBEST3D

Simulation pipeline

→ in general: piecewise constant, linear and quadratic DG

→ 2nd order Runge Kutta

Semi-implicit with fixed #iterations in implicit part:

No convergence-theory needed in PM

Input global parameters
Input 2D mesh
Create 3D mesh
$t \leftarrow t_0$
while $t < t_{end}$ **do**
  Update free surface elevation (every 10th time step)
  $i \leftarrow 0$
  while $i < n_{\text{stages}}$ **do**
    Perform slope limiting on θ and s
    Compute body forcings for right hand sides: F_i-terms
    Project density field and compute density forcing in (1b)
    Integrate terms in (1a), (1b), (1d) over lateral faces
    Integrate bottom boundary condition in (1a), (1b), (1d)
    Solve (1c) to obtain the vertical velocity w
    Integrate terms in (1a), (1b), (1d) over interior and surface horiz. faces
    Integrate terms in (1a), (1b), (1d) over prisms
    Compute next stage of Runge-Kutta method for h, u, v, θ, s
    $i \leftarrow i + 1$
  end while
  Perform slope limiting on turbulence variables
  Assemble diffusion matrices and right hand sides:
    Integrate terms in (1b), (1d), (1e) over all horizontal faces
    Integrate terms in (1b), (1d), (1e) over prisms
  Solve for diffusion contributions
  $t \leftarrow t + dt$
end while

explicit

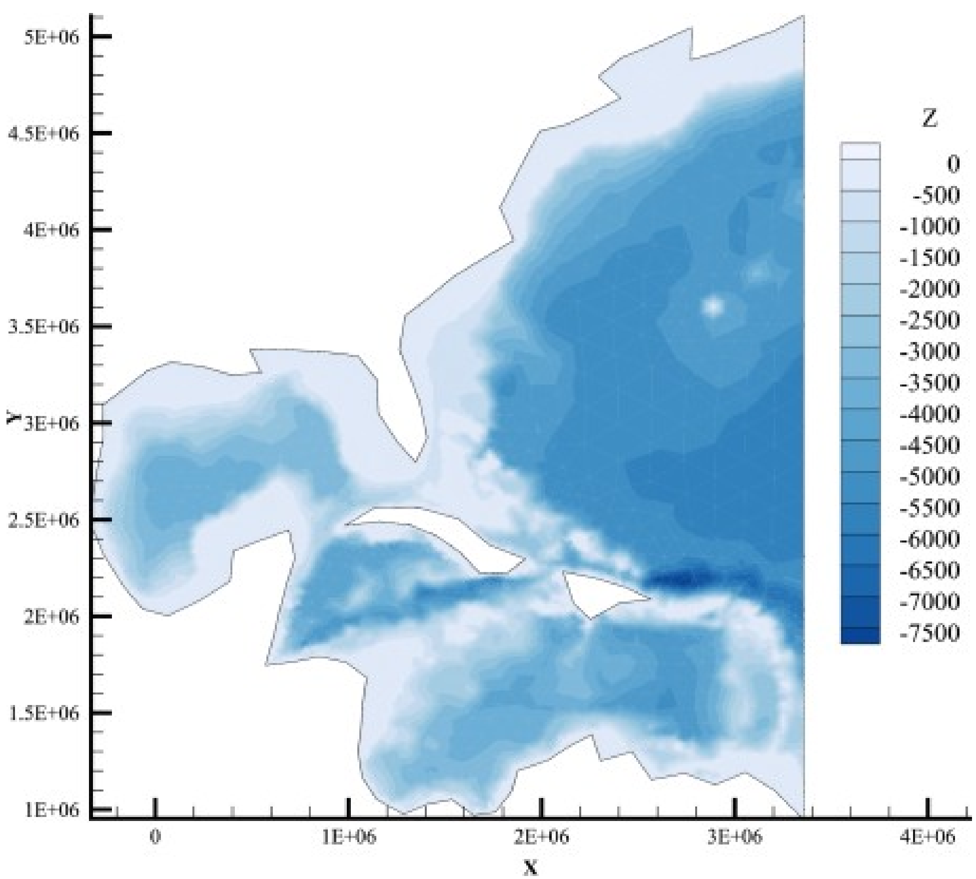
implicit

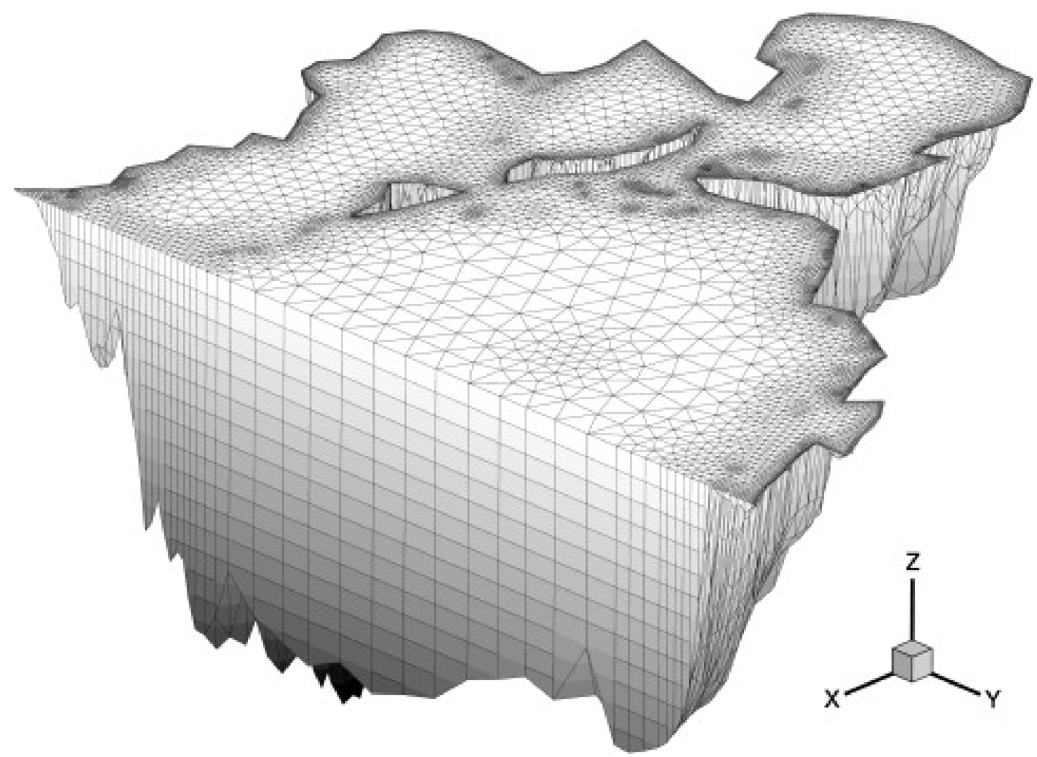
Test configuration

Tidal flow

→ part of the Atlantic Ocean adjoining the eastern seaboard of North America, Gulf of Mexico, and the Caribbean

→ a priori adapted horizontal FE mesh:
~19k triangles: ~1km (coastal) up to 120km (open waters)





Testhardware

Complete 'box'

→ measure power at
AC-converter (inlet)
→ all power needed for the node

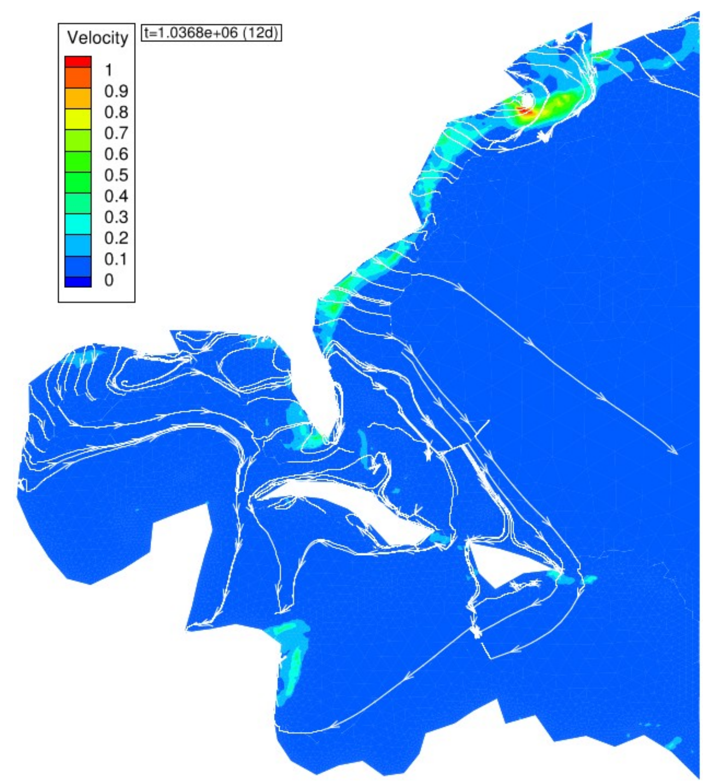
	i5-4690K	Jetson TK1
micro-architecture	Haswell	Cortex-A15 (Tegra K1)
N_{cores}	4	4
clock speed	3.50 GHz (turbo 3.9 GHz)	2.3 GHz
L1-cache	4x 32 KB + 4x 32 KB	32 KB + 32 KB
L2- / L3-cache	4x 256 KB / 6 MB	2 MB / –
memory type	DDR3	LPDDR3
peak memory bandwidth	25.6 GByte/s	14.9 GByte/s
P_{base}	41 W (Intel chipset)	3.9 W (Jetson TK1)
	(2015)	(2014)

Sample results

Measurements

→ preset core-frequency
→ fixed problem size
→ increase #cores

	i5-4690K	Jetson TK1
micro-architecture	Haswell	Cortex-A15 (Tegra K1)
N_{cores}	4	4
clock speed	3.50 GHz (turbo 3.9 GHz)	2.3 GHz
L1-cache	4x 32 KB + 4x 32 KB	32 KB + 32 KB
L2- / L3-cache	4x 256 KB / 6 MB	2 MB / –
memory type	DDR3	LPDDR3
peak memory bandwidth	25.6 GByte/s	14.9 GByte/s
P_{base}	41 W (Intel chipset)	3.9 W (Jetson TK1)



Sample results

Measurements

→ preset core-frequency
→ fixed problem size
→ increase #cores

	i5-4690K	Jetson TK1
micro-architecture	Haswell	Cortex-A15 (Tegra K1)
N_{cores}	4	4
clock speed	3.50 GHz (turbo 3.9 GHz)	2.3 GHz
L1-cache	4x 32 KB + 4x 32 KB	32 KB + 32 KB
L2- / L3-cache	4x 256 KB / 6 MB	2 MB / –
memory type	DDR3	LPDDR3
peak memory bandwidth	25.6 GByte/s	14.9 GByte/s
P_{base}	41 W (Intel chipset)	3.9 W (Jetson TK1)

k	T [s]	P [W]	E [J]
			Haswell
1	0.0768	65.6	5.038
2	0.0434	80.7	3.502
3	0.0352	91.4	3.217
4	0.0316	100.3	3.169
			Cortex-A15
1	0.477	7.5	3.577
2	0.249	10	2.49
3	0.173	12	2.076
4	0.170	13.9	2.363

Sample results

Measurements

→ **performance-up and energy increase at the same time?**

	i5-4690K	Jetson TK1
micro-architecture	Haswell	Cortex-A15 (Tegra K1)
N_{cores}	4	4
clock speed	3.50 GHz (turbo 3.9 GHz)	2.3 GHz
L1-cache	4x 32 KB + 4x 32 KB	32 KB + 32 KB
L2- / L3-cache	4x 256 KB / 6 MB	2 MB / –
memory type	DDR3	LPDDR3
peak memory bandwidth	25.6 GByte/s	14.9 GByte/s
P_{base}	41 W (Intel chipset)	3.9 W (Jetson TK1)

k	T [s]	P [W]	E [J]
			Haswell
1	0.0768	65.6	5.038
2	0.0434	80.7	3.502
3	0.0352	91.4	3.217
4	0.0316	100.3	3.169
			Cortex-A15
1	0.477	7.5	3.577
2	0.249	10	2.49
3	0.173	12	2.076
4	0.170	13.9	2.363

this is why we need to model energy

Sample results

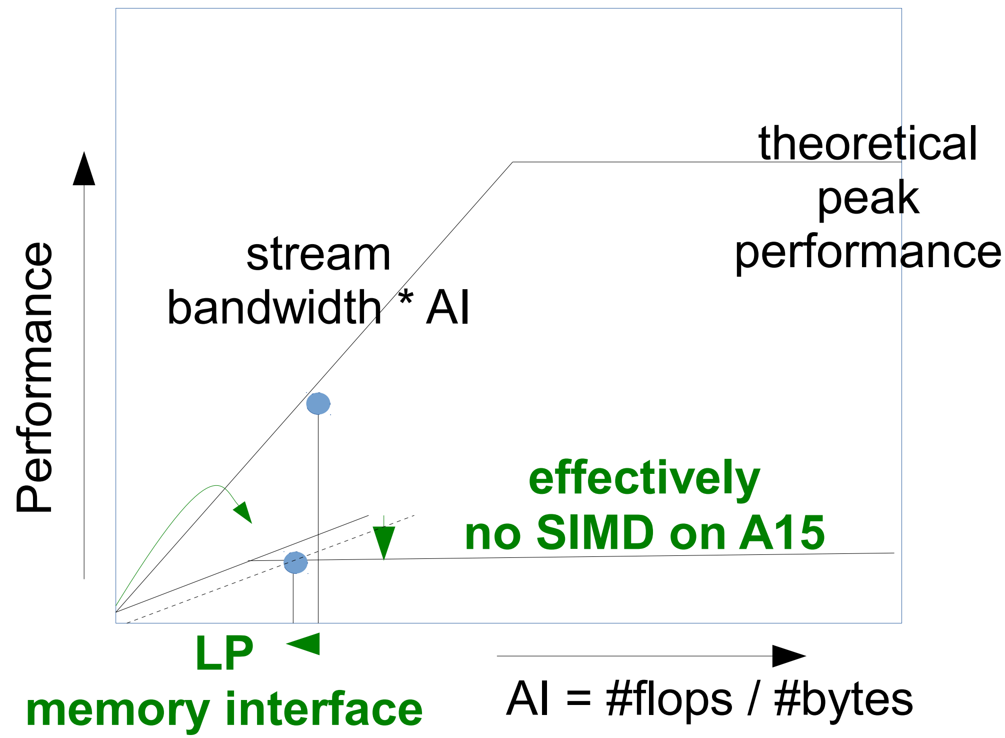
Performance

	i5-4690K	Jetson TK1
micro-architecture	Haswell	Cortex-A15 (Tegra K1)
N_{cores}	4	4
clock speed	3.50 GHz (turbo 3.9 GHz)	2.3 GHz
L1-cache	4x 32 KB + 4x 32 KB	32 KB + 32 KB
L2- / L3-cache	4x 256 KB / 6 MB	2 MB / –
memory type	DDR3	LPDDR3
peak memory bandwidth	25.6 GByte/s	14.9 GByte/s
P_{base}	41 W (Intel chipset)	3.9 W (Jetson TK1)

memory-bandwidth -bound,

speed-down x5

k	T [s]	P [W]	E [J]
			Haswell
1	0.0768	65.6	5.038
2	0.0434	80.7	3.502
3	0.0352	91.4	3.217
4	0.0316	100.3	3.169
			Cortex-A15
1	0.477	7.5	3.577
2	0.249	10	2.49
3	0.173	12	2.076
4	0.170	13.9	2.363



Sample results

Power

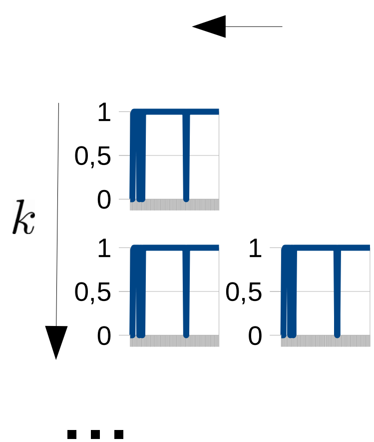
	i5-4690K	Jetson TK1
micro-architecture	Haswell	Cortex-A15 (Tegra K1)
N_{cores}	4	4
clock speed	3.50 GHz (turbo 3.9 GHz)	2.3 GHz
L1-cache	4x 32 KB + 4x 32 KB	32 KB + 32 KB
L2- / L3-cache	4x 256 KB / 6 MB	2 MB / –
memory type	DDR3	LPDDR3
peak memory bandwidth	25.6 GByte/s	14.9 GByte/s
P_{base}	41 W (Intel chipset)	3.9 W (Jetson TK1)

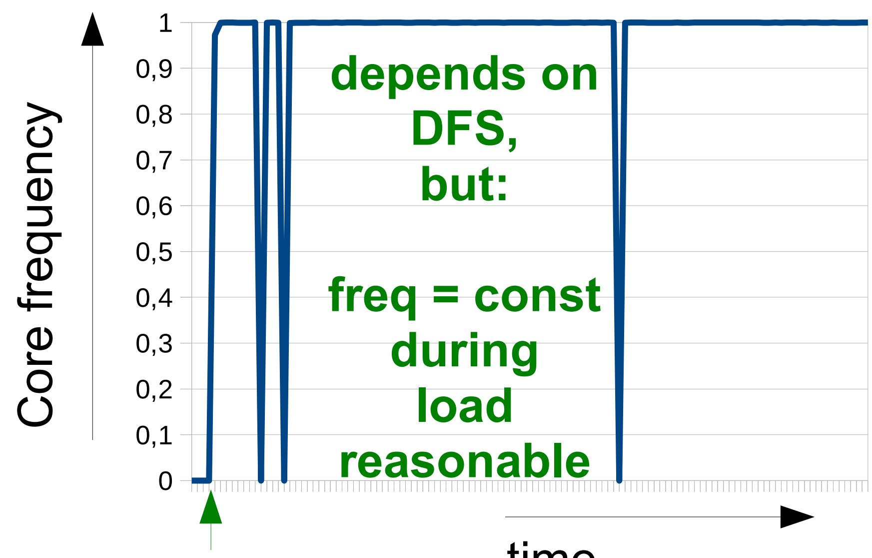
→ race to idle:
core is either 'on' or 'off'

→ power-down x8

→ energy-down x1.7

k	T [s]	P [W]	E [J]
		Haswell	
1	0.0768	65.6	5.038
2	0.0434	80.7	3.502
3	0.0352	91.4	3.217
4	0.0316	100.3	3.169
		Cortex-A15	
1	0.477	7.5	3.577
2	0.249	10	2.49
3	0.173	12	2.076
4	0.170	13.9	2.363





incoming load

A simple model

Power dissipation as a function of used cores

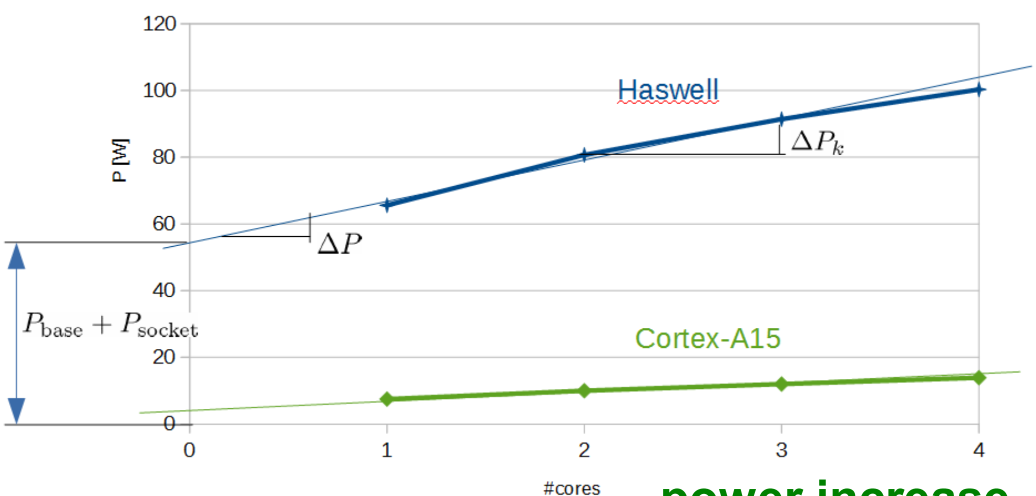
$$\Delta T = \Delta T(k) = T_{k-1} - T_k$$

$$\Delta P = \Delta P(k) = P_k - P_{k-1}$$

k	T [s]	P [W]	E [J]	ΔT [s]	ΔP [W]
		Haswell			
1	0.0768	65.6	5.038	–	–
2	0.0434	80.7	3.502	0.0334	15.1000
3	0.0352	91.4	3.217	0.0082	10.7000
4	0.0316	100.3	3.169	0.0036	8.9000
		Cortex-A15			
1	0.477	7.5	3.577	–	–
2	0.249	10	2.49	0.228	2.5000
3	0.173	12	2.076	0.076	2.0000
4	0.170	13.9	2.363	0.003	1.9000

→ predict nodal *E as*

$$\mathrm{E(k)} = (P_{\text{base}} + P_{\text{socket}} + k\Delta P)\, T(k)\,, \quad k \geq 1\,.$$

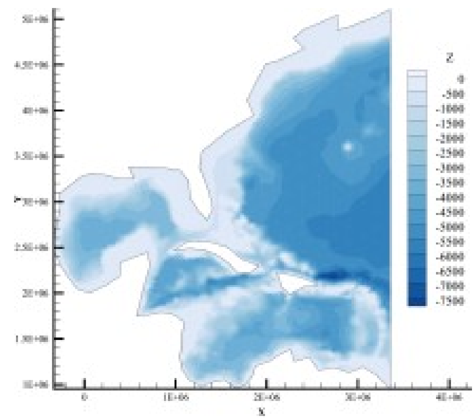


power increase per core is smaller on ARM

Model validation

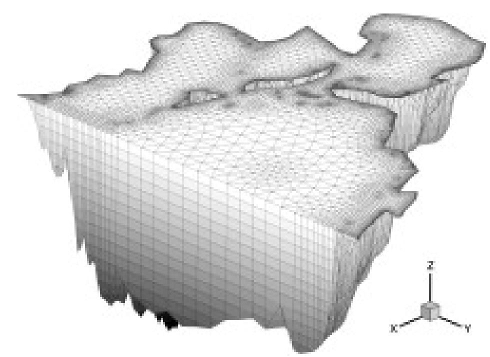
Test case 1

→ no vertical diffusion, single layer of prisms,
→ no free surface updates
→ explicit time-stepping



Test case 2: test case 1 plus

→ add algebraic eddy viscosity model
→ add free surface updates



Test case 3: test case 2 plus

→ add temperature and salinity transport (barotropic → baroclinic)

Test case 4: test case 3 but

→ replace algebraic eddy viscosity with the k-ϵ closure

Test cases

Elevation

$$\partial_t h + \nabla \cdot \int_{z_b}^{\xi} \mathbf{u}_{xy} \mathrm{d}z = 0$$

Continuity

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

Momentum

test cases 2 - 4

test cases 3 - 4

$$\partial_t \mathbf{u}_{xy} + \nabla \cdot (\mathbf{u}_{xy} \otimes \mathbf{u}) - \partial_z (\nu_t \partial_z \mathbf{u}_{xy}) + \nabla_{xy}(gh + p) - f_c \mathbf{k} \times \mathbf{u}_{xy} = \mathbf{F}_u - \nabla_{xy} z_b$$

Temperature-/ salinity-transport

$$\partial_t r + \nabla \cdot (\mathbf{u} r) - \partial_z (\nu_r \partial_z r) = F_r$$

test cases 3 - 4

Turbulent quantities transport

$$\partial_t m - \partial_z (\nu_m \partial_z m) = F_m$$

test case 4

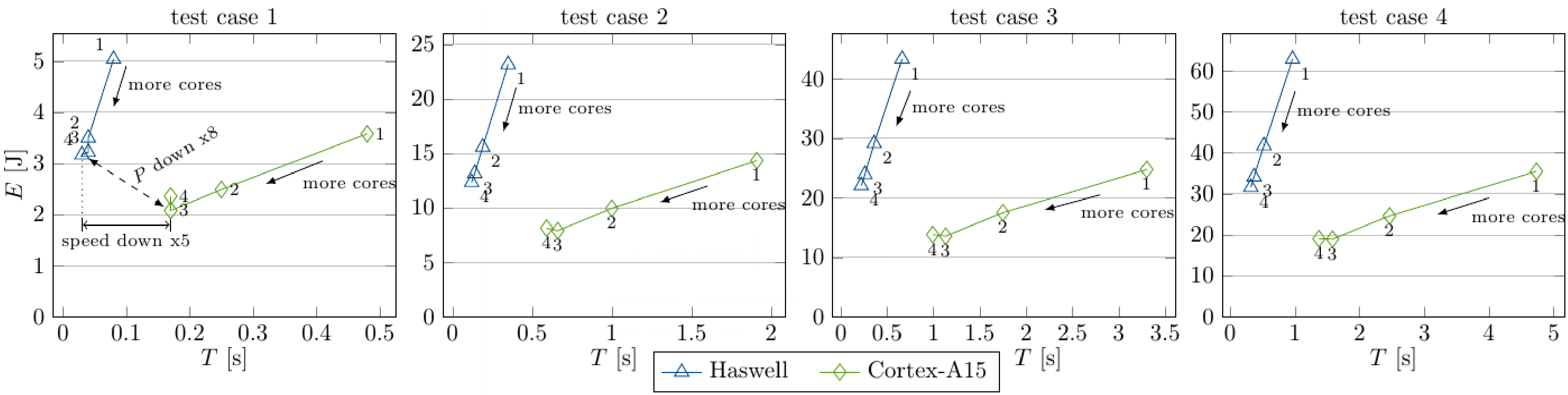
Density forcing

test cases 3 - 4

$$p(x,y,z) = \frac{g}{\rho_0} \int_z^{\xi} (\rho(\theta(x,y,\tilde{z}), s(x,y,\tilde{z})) - \rho_0)\, \tilde{z}$$

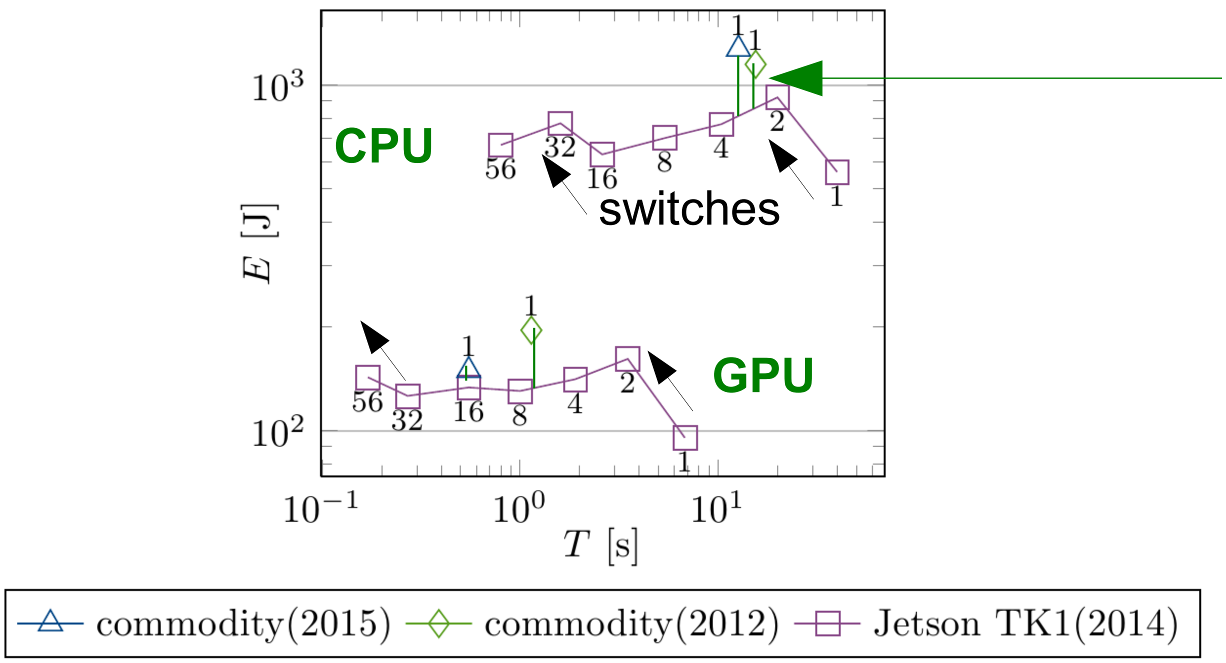
Results

There is an optimal *k*, independent of the simulation



Results (outlook)

We can build better computers



Haswell,
GTX 980 Ti

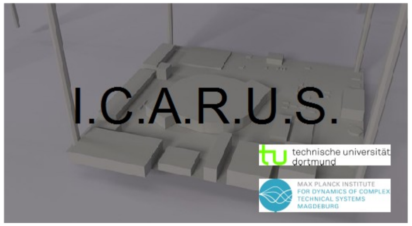
Ivybridge,
GTX 660

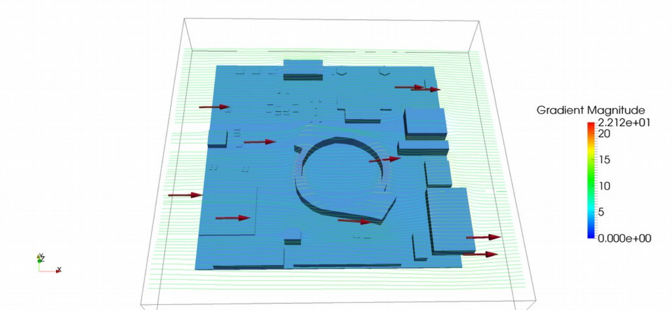
Cortex-A15,
GK20 Kepler

This is what we are shooting at:

Less energy at Perf = const

An off-grid compute center of the future



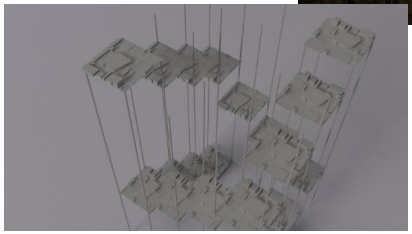


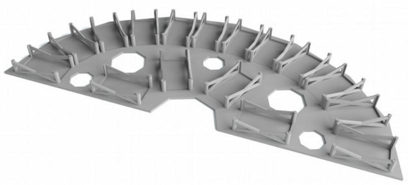
Vision

- **I**nsular
- **C**ompute-center for
- **A**pplied Mathematics with
- **R**enewables-provided power supply based on
- **U**nconventional compute hardware empaired with
- **S**imulation Software for Technical Processes

Motivation

- **system integration** for Scientific HPC
  → high-end unconventional compute hardware
  → high-end renewable power source (photo-voltaic)
  → specially tailored numerics, simulation software
- **no future spendings due to energy consumtion**
- **SME-class resource: <80K€**
- **Scalability, modular design**
- (simplicity)
- (maintainability)
- (safety)
- ...





Cluster

Whitesheet

→ **nodes:** 60 x NVIDIA Jetson TK 1
→ #cores (ARM Cortex-A15): 240
→ #GPUs (Kepler, 192 cores): 60
→ RAM/core: 2GB LPDDR3
→ switches (GiBit Ethernet): 3xL1, 1xL2
→ **cluster theoretical peak perf: ~20TFlop/s SP**
→ **cluster peak power: < 1kW, provided by PV**

→ **PV capacity: 8kWp**
→ **battery: 8kWh**

→ **Software: FEAT (optimised for Tegra K1): www.featflow.de**

Conclusion and outlook

Power and energy modelling

→ smaller power dissipation alone is not the deal
→ performance modelling/-engineering of software for EE is needed
→ during simulation, complex power signatures
  start to happen → kernel-based PMs insufficient

Shallow Water Simulations on ARM

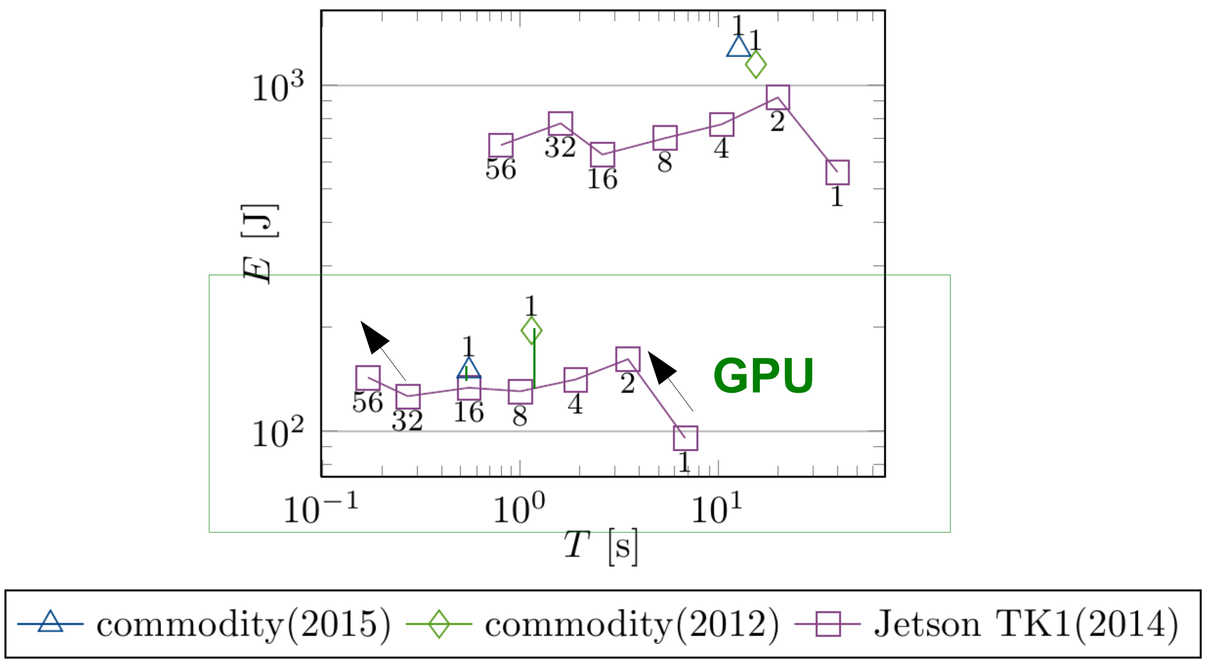
→ there is a number of TK1 boards we need to beat commodity hardware
→ for UTBEST: 6 TK1 boards deliver the CPU performance of a workstation
→ energy-down x1.7

Hardware-/Software Co-Design

→ Embedded tech has a different history than commodity hardware
→ Energy Efficiency is just starting to arrive in HPC
→ System Integration with state-of-the-art PV tech (or other renewables) is promising

Conclusion and outlook

We can build better computers



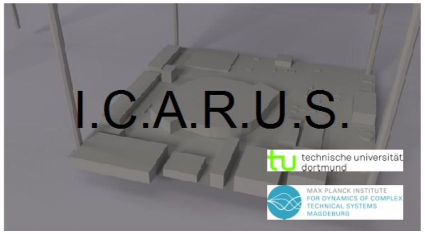
Haswell,
GTX 980 Ti

Ivybridge,
GTX 660

Cortex-A15,
GK20 Kepler

Will the trend continue?
→ TX1(8x Cortex-A57 + Maxwell GPU) vs 1080, P100, ...

Thank you



www.icarus-green-hpc.org

Meet us @ UcHPC'16, Euro-Par'16, Grenoble, France, August 22nd-23rd 2016
(ICARUS white-paper presentation)

This work has been supported in part by the German Research Foundation (DFG)
through the Priority Program 1648
‘Software for Exascale Computing’
(grants TU 102/48, GO 1758/2), and through the individual grant AI 117/1.

ICARUS hardware is financed by MIWF NRW under the lead of MERCUR.
.