technische universität dortmund

fakultät für mathematik

Realization of a low energy HPC platform powered by renewables - A case study: Technical, numerical and implementation aspects

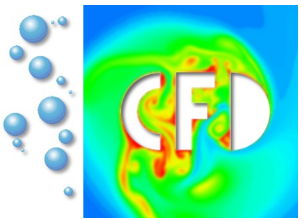
Markus Geveler, Stefan Turek, Dirk Ribbrock

PACO Magdeburg
2015 / 7 / 7

markus.geveler@math.tu-dortmund.de

Institute f. Applied Mathematics

14. April 2015



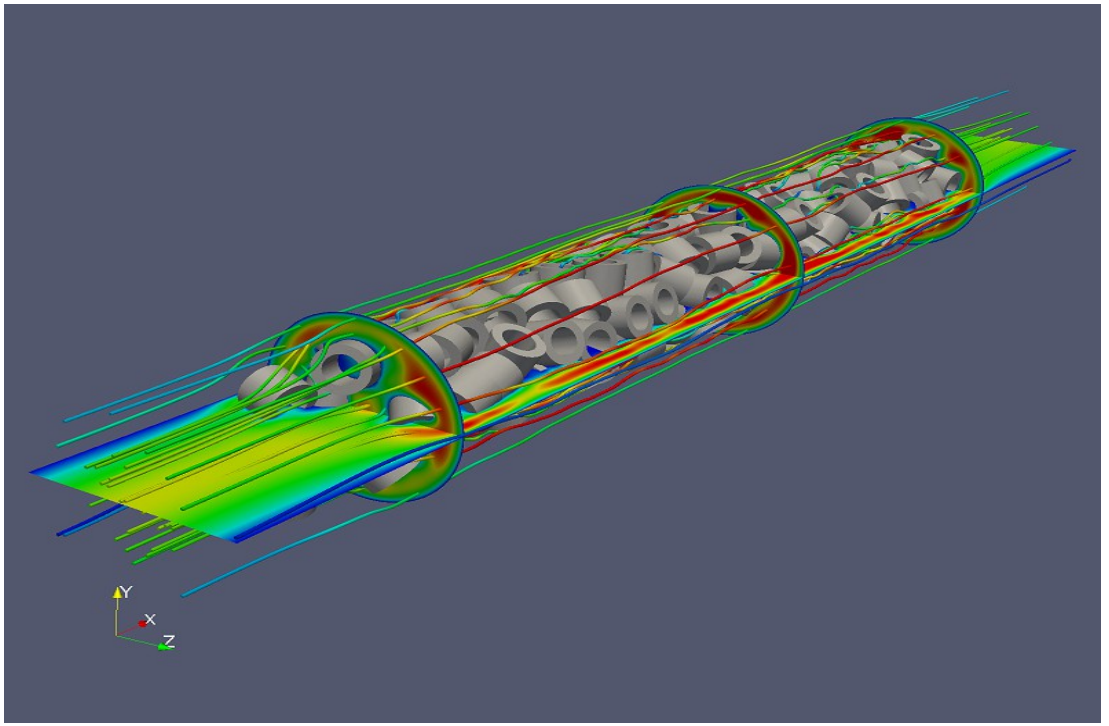
outline

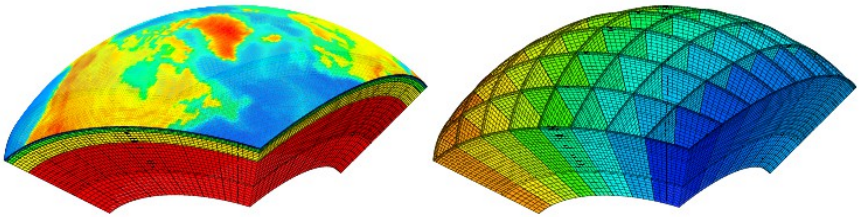
three parts

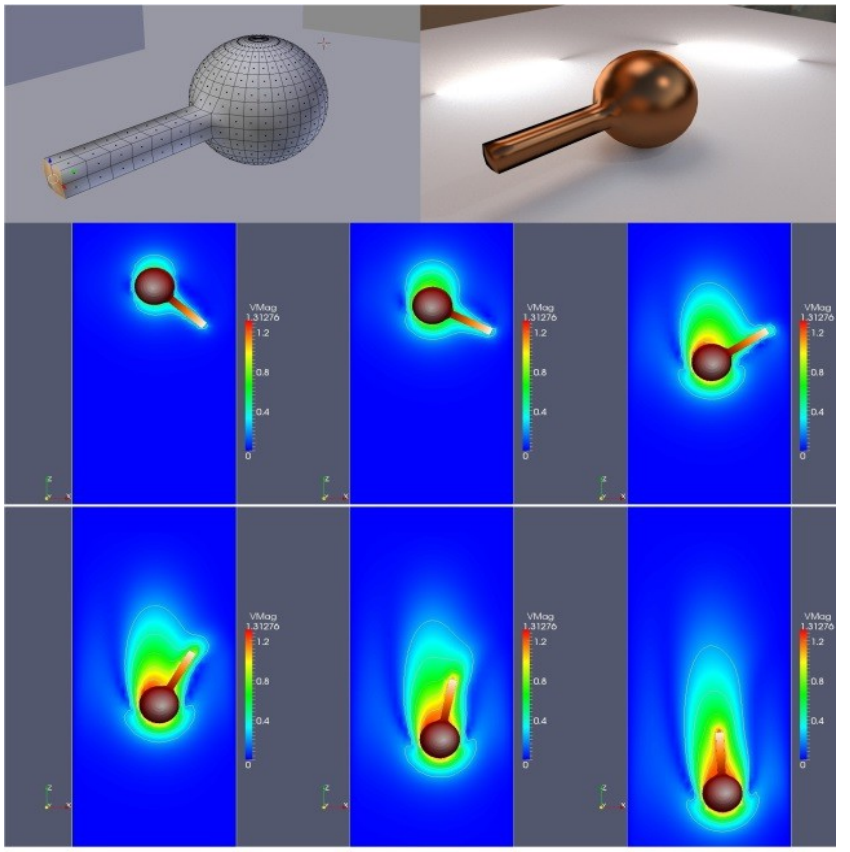
→ motivation and preliminary work
→ the I.C.A.R.U.S system
→ main part:
performance engineering for the Tegra K1 with focus on energy efficiency

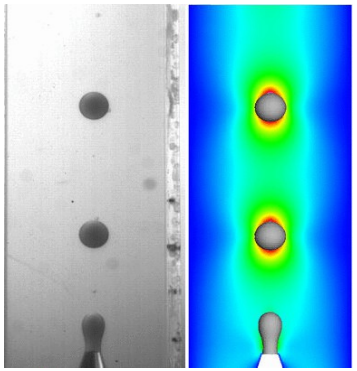
where it all leads...

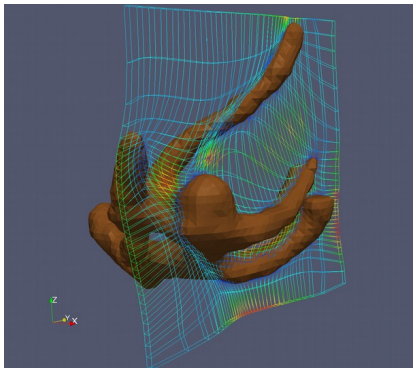
simulation of technical flows

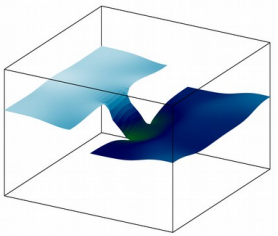






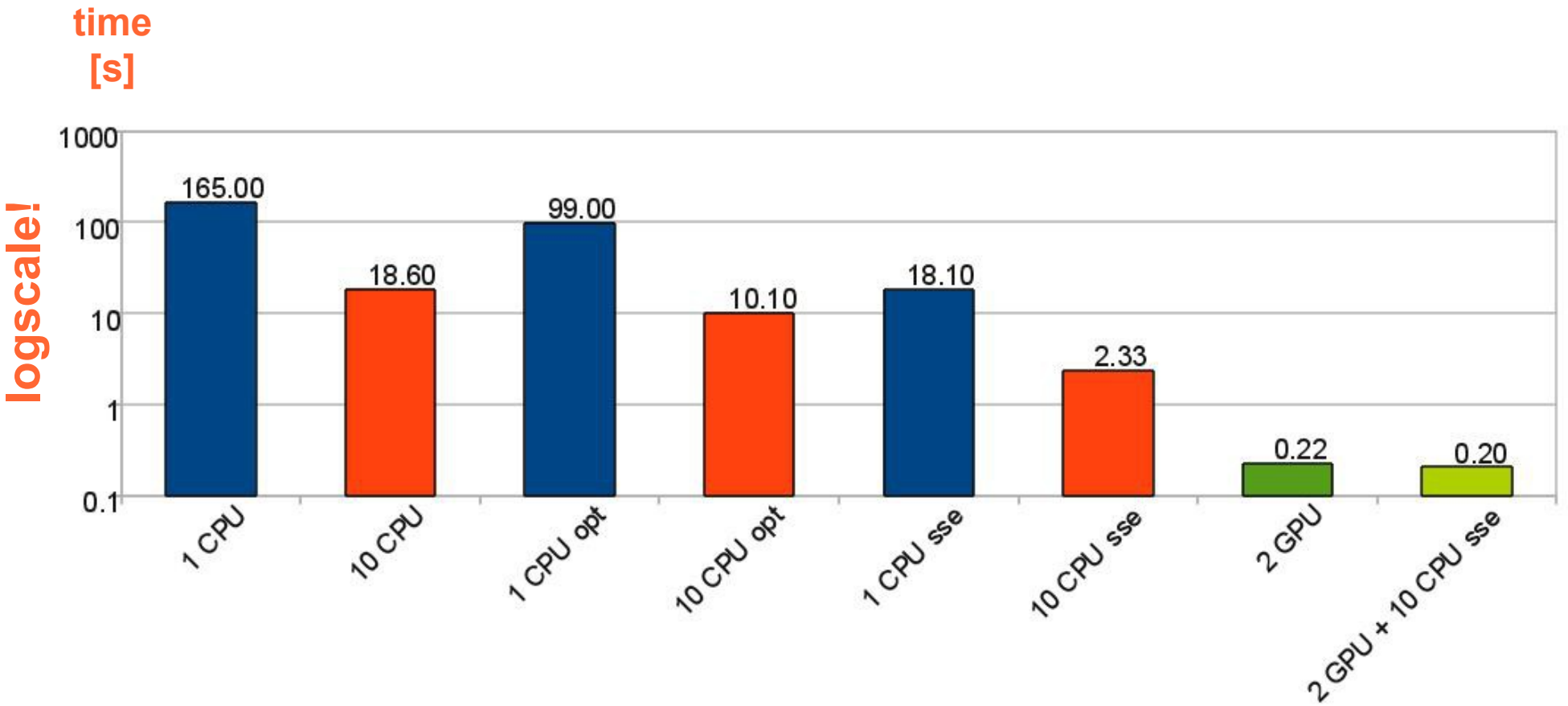






Hardware-oriented Numerics

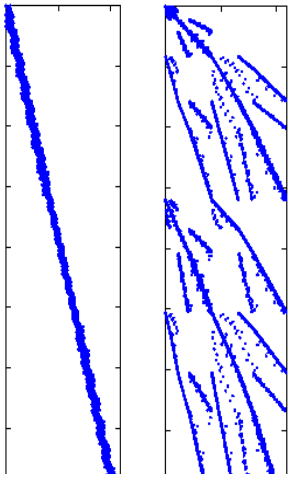
(I) : hardware efficiency



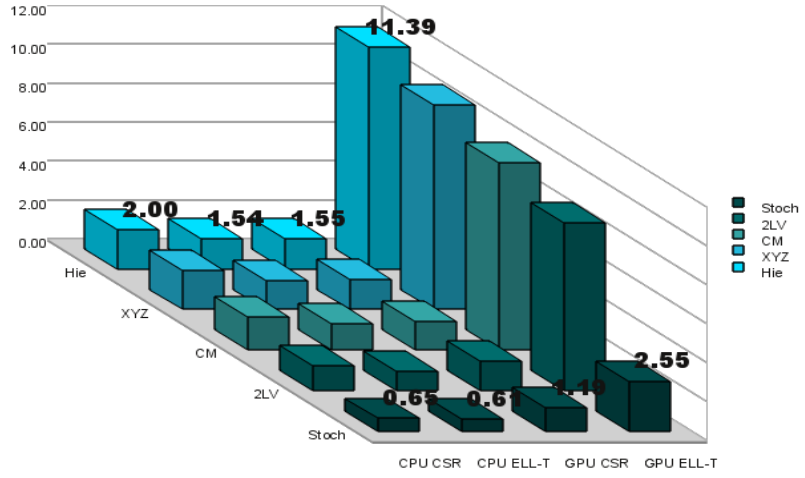
Hardware-oriented Numerics

(I) hardware efficiency: kernel-based optimisation: SpMV

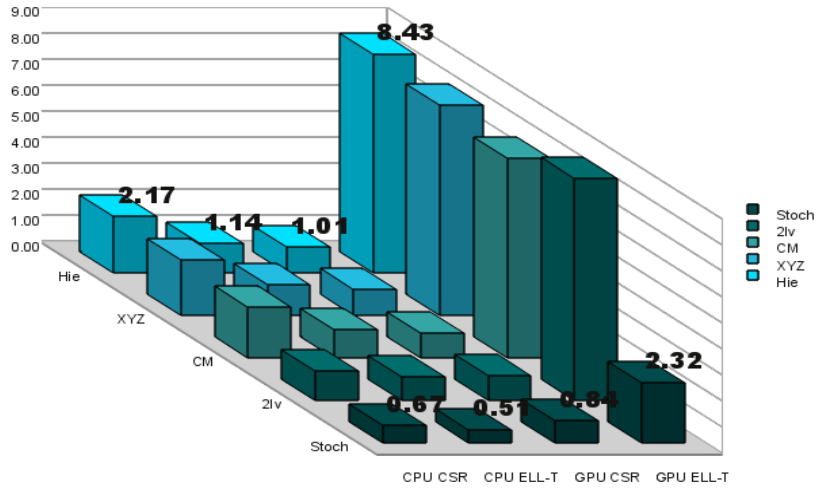
- one of the most prominent kernels in solving PDEs with high-end FEM
- memory access matters a lot
- hardware efficiency considerations start early: DOF numbering
- hardware-efficiency requires different matrix storage
- FE space matters



Performance [Gflop/s]



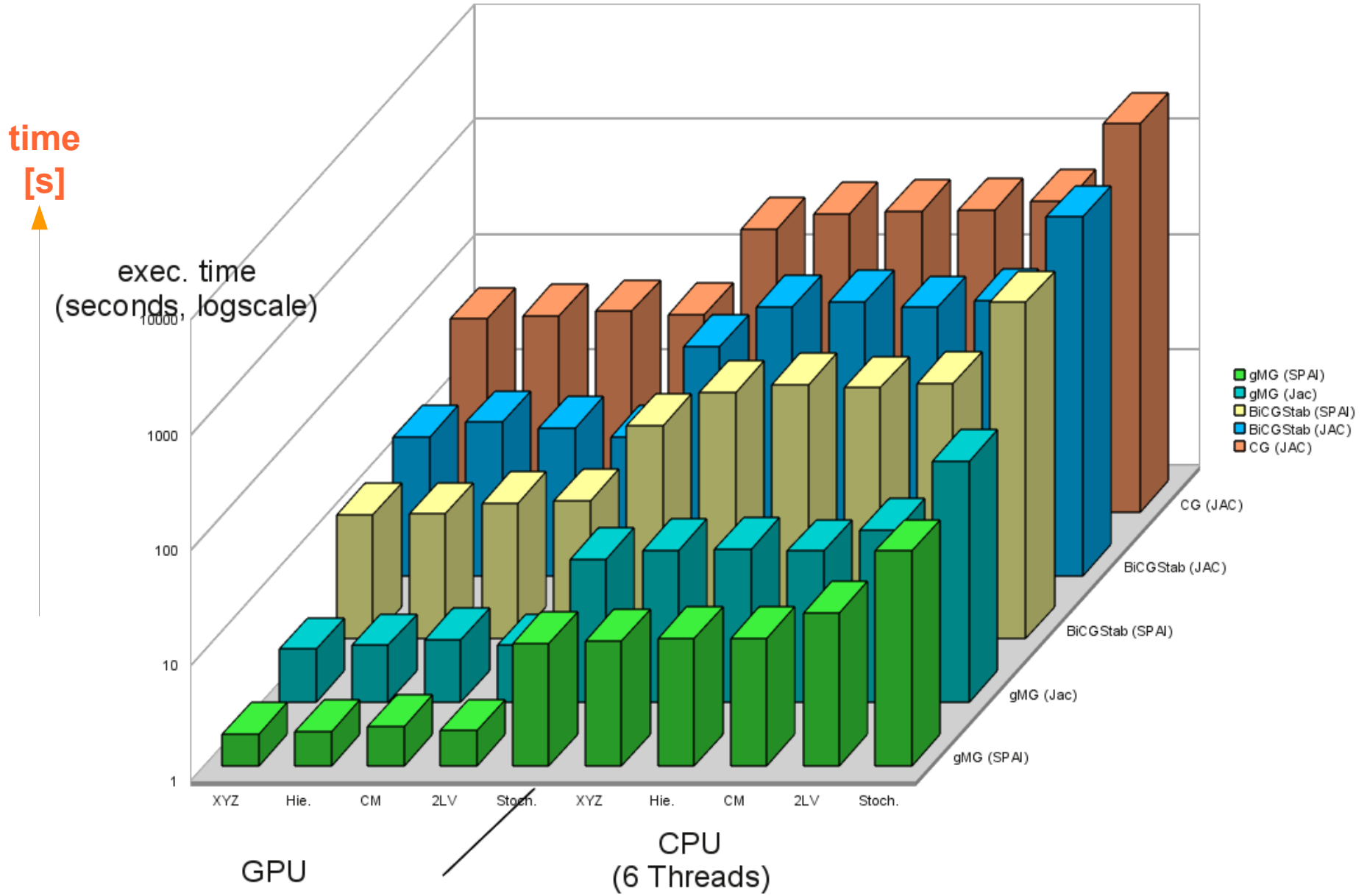
Q1



Q2

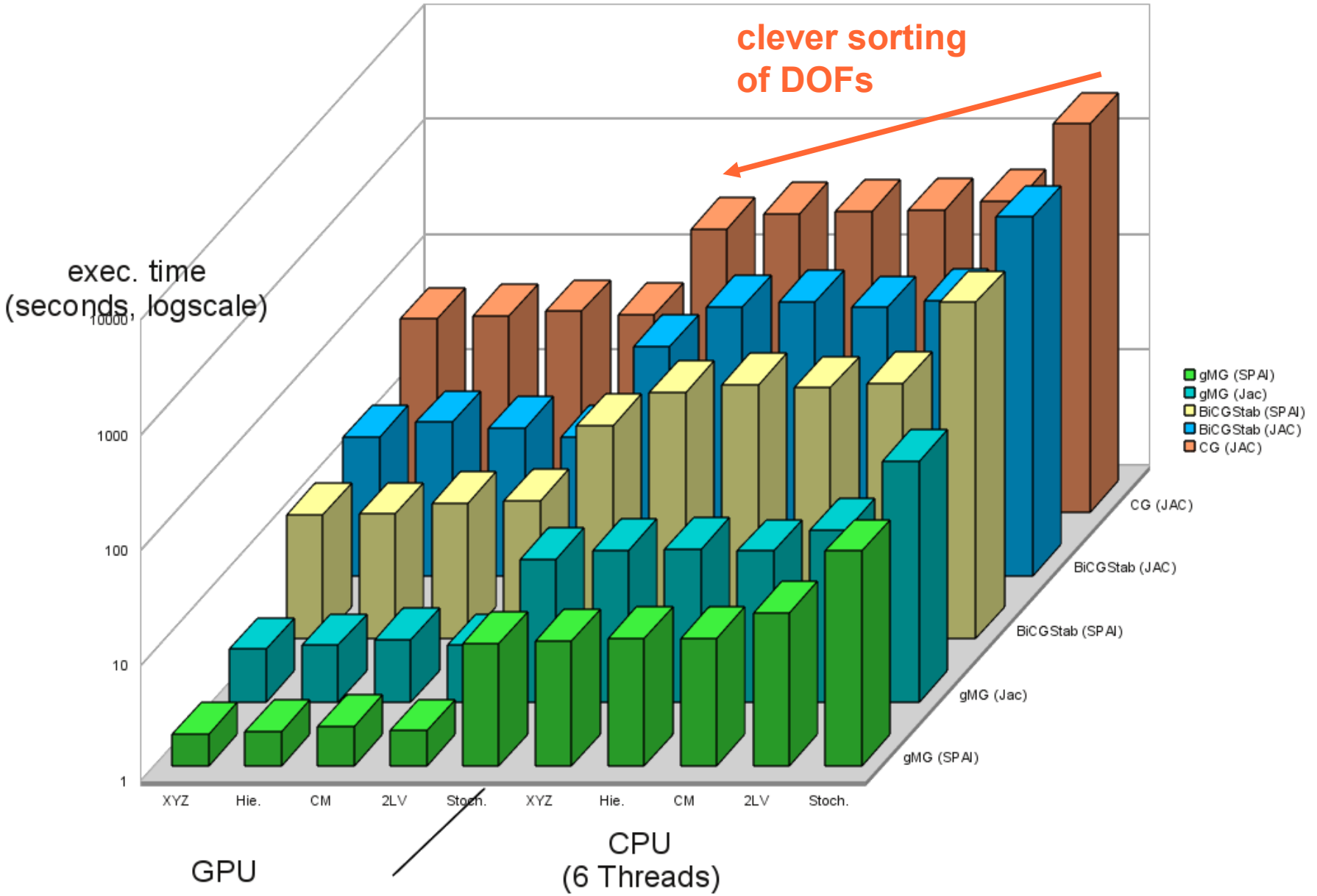
Hardware-oriented Numerics

(II) numerical efficiency



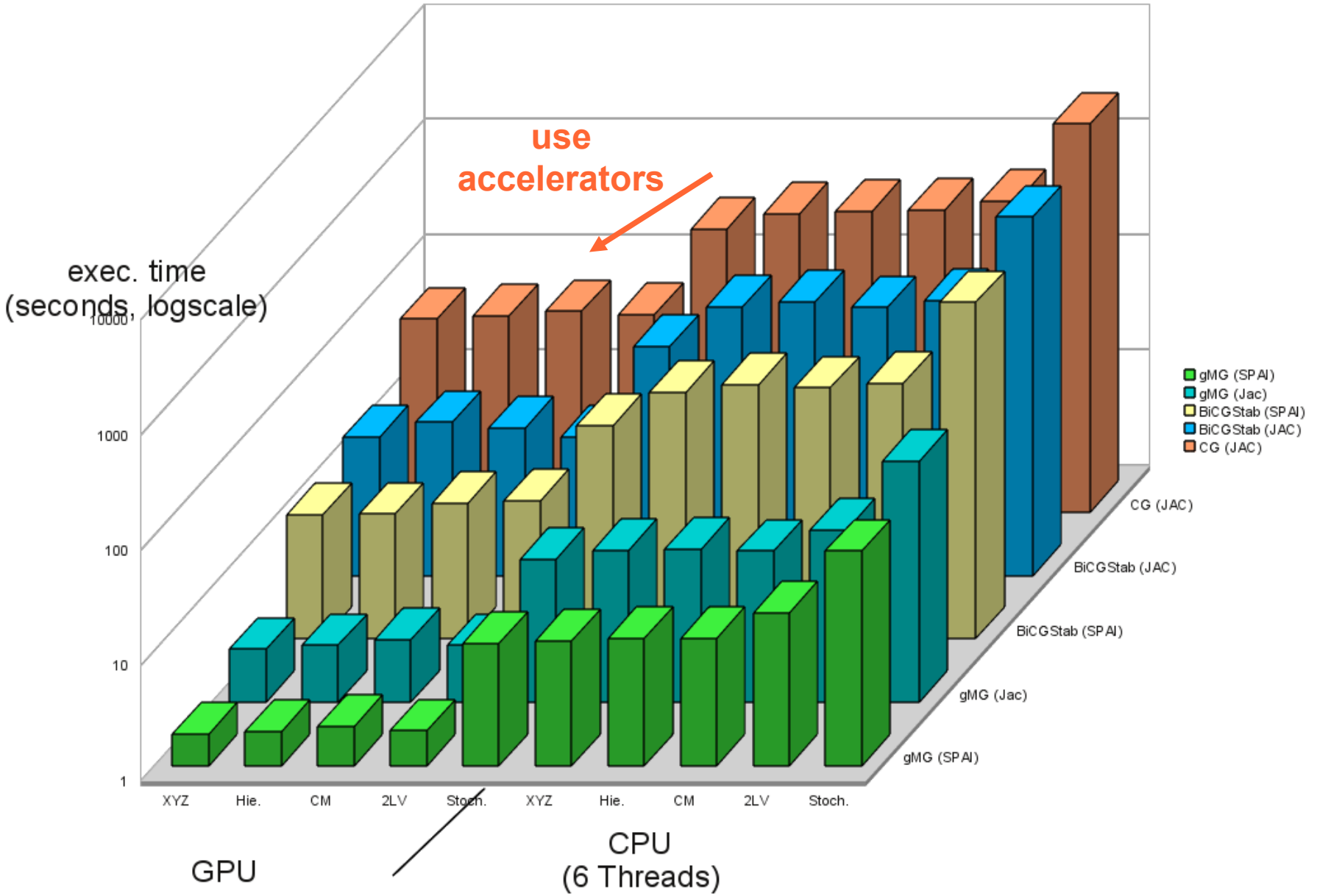
Hardware-oriented Numerics

(II) numerical efficiency



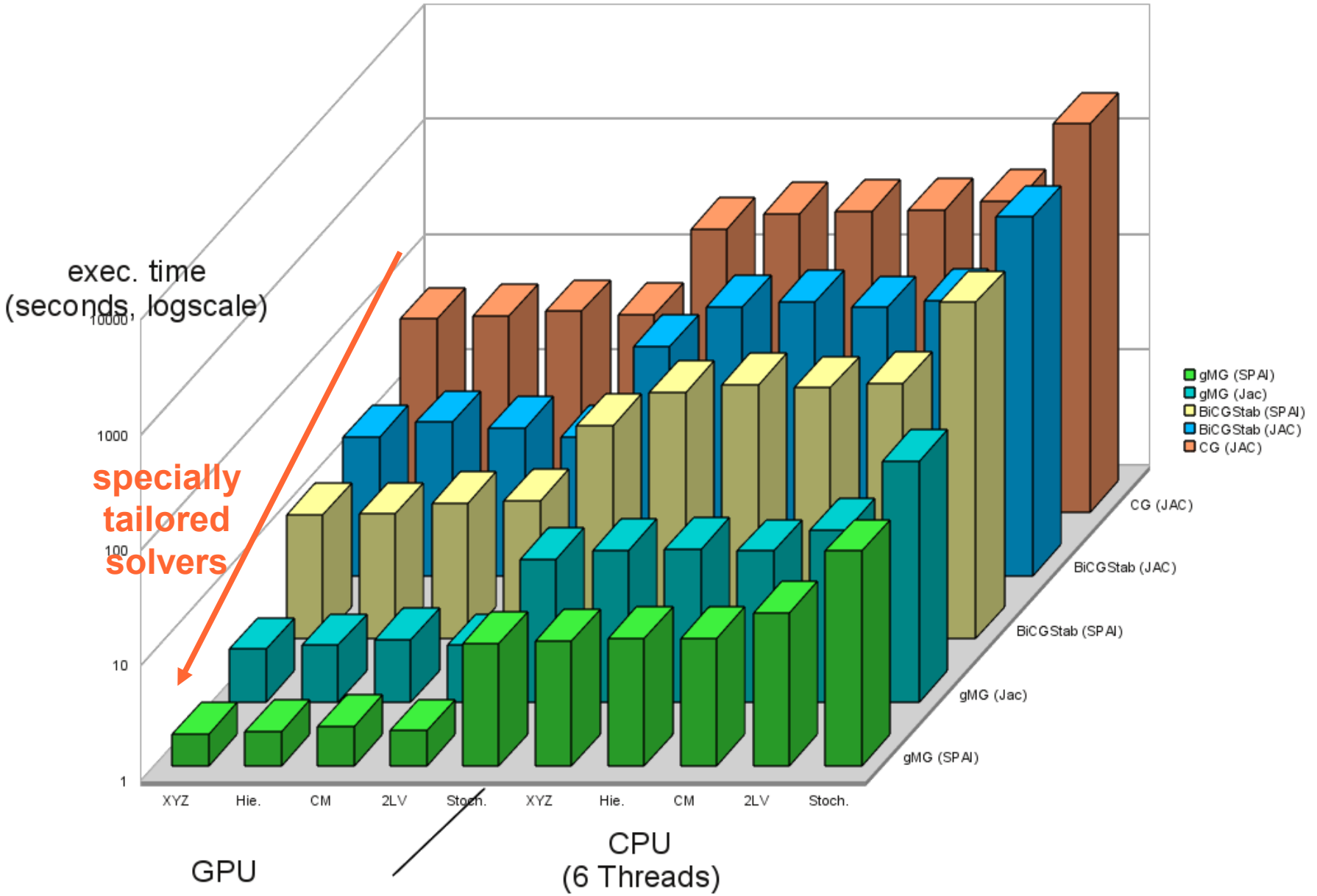
Hardware-oriented Numerics

(II) numerical efficiency



Hardware-oriented Numerics

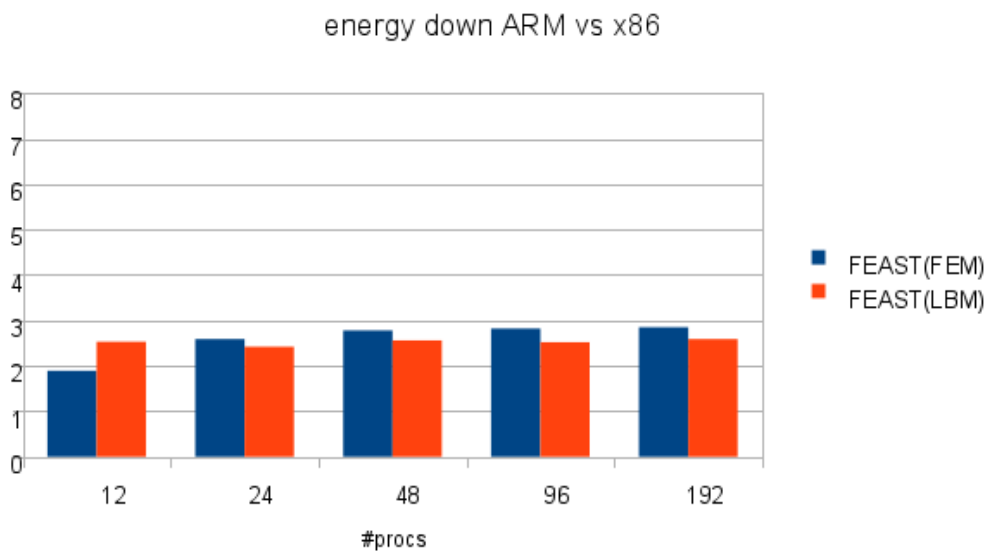
(II) numerical efficiency



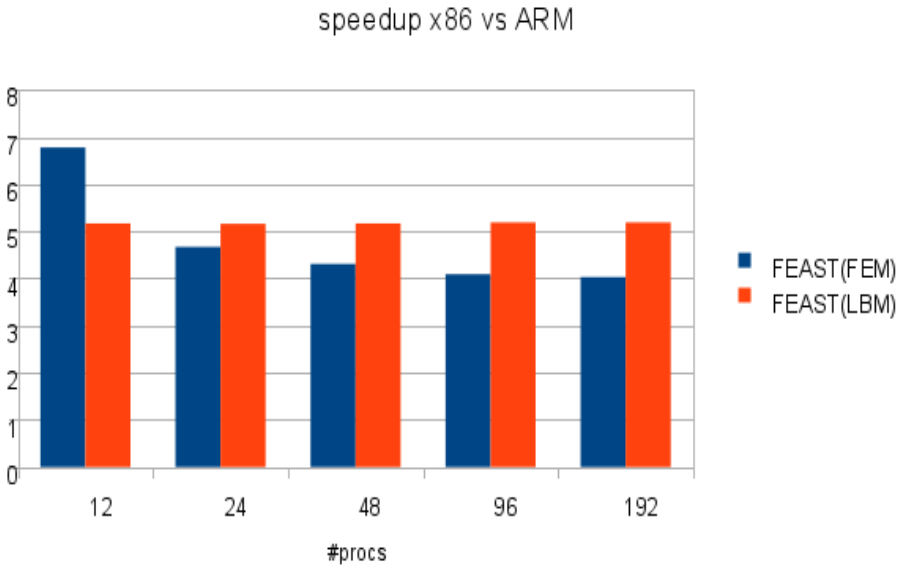
Hardware-oriented Numerics

(III) *energy* efficiency (?)

- energy consumption/efficiency is one of the major challenges for future supercomputers
- we can not afford to go all 'macho-flops' any more
- 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 (like powering down your x86)



~3x less energy



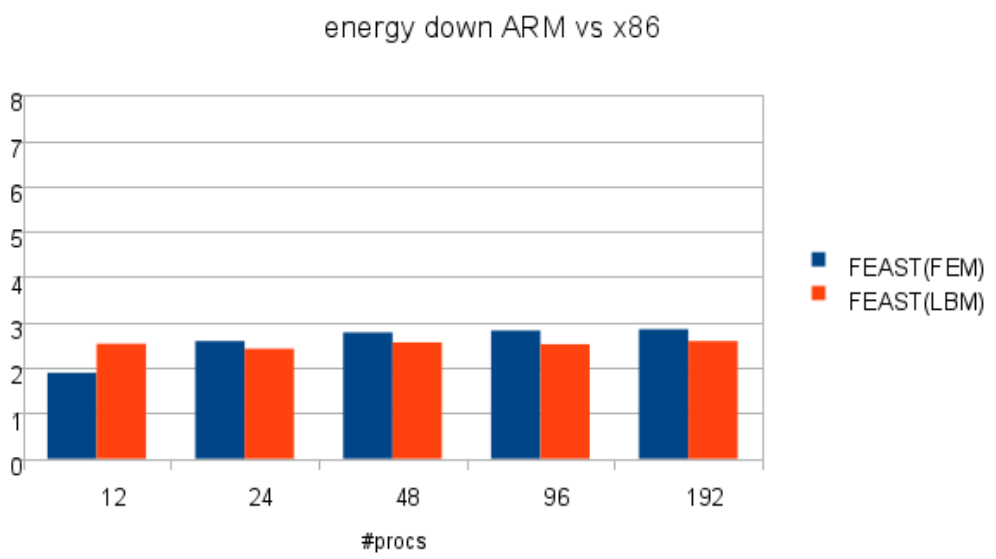
but: also ~5x more time!

Hardware-oriented Numerics

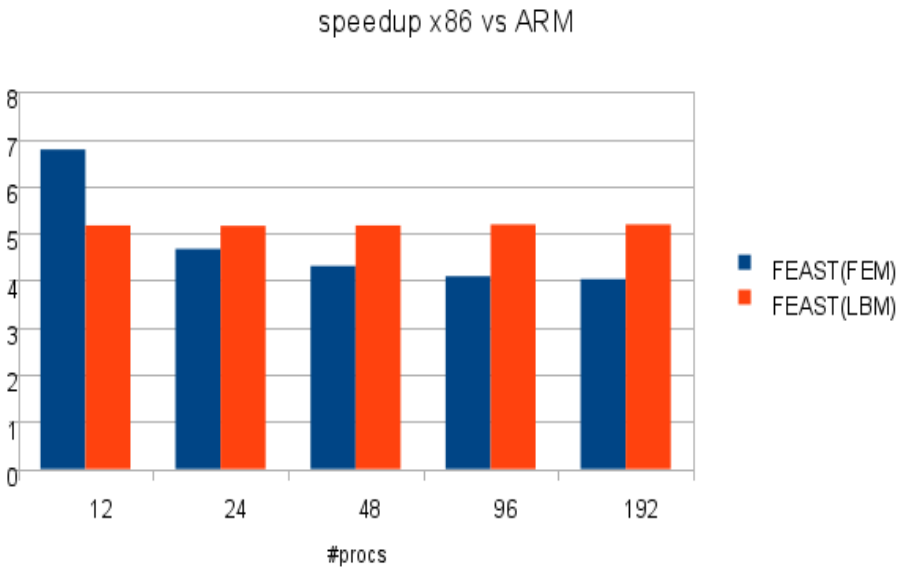
(III) *energy* efficiency (?)

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

→ more performance per Watt



~3x less energy



but: also ~5x more time!

Hardware-oriented Numerics

(III) *energy* efficiency: technology of ARM-based SoCs since 2012

- one word in advance: there are many more SoC designs (like from TI, Qualcomm, …)

→ Tegra 3 (late 2012) was also based on A9 but had 4 cores
→ Tegra 4 (2013) are 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

but more importantly: 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

→ 2.5 higher EE promised

→ interesting for Scientific Computing! Higher EE than commodity!

other aspects in modern HPC

so far

→ Hardware-oriented Numerics is three-fold now
→ there are well-understood methods for PDE simulations w.r.t. hardware- and numerical efficiency
→ we hope to tackle the energy challenge by using unconventional hardware from the mobile computing realm

in addition

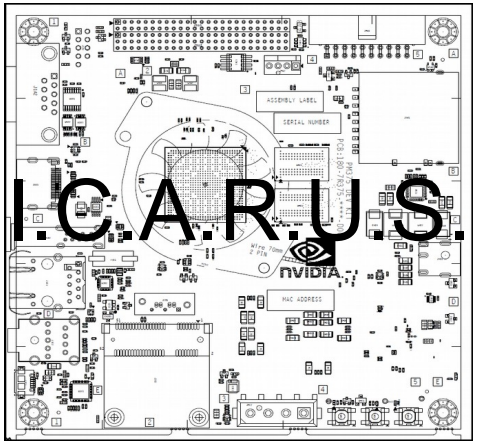
- German 'Energiewende' will not work without consumers to adjust (alongside producers and nets). The scientific community and simulation-related SMLEs are considered 'users' of our idea

  idea: build a small prototype of a low-energy cluster with Tegra K1

a 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

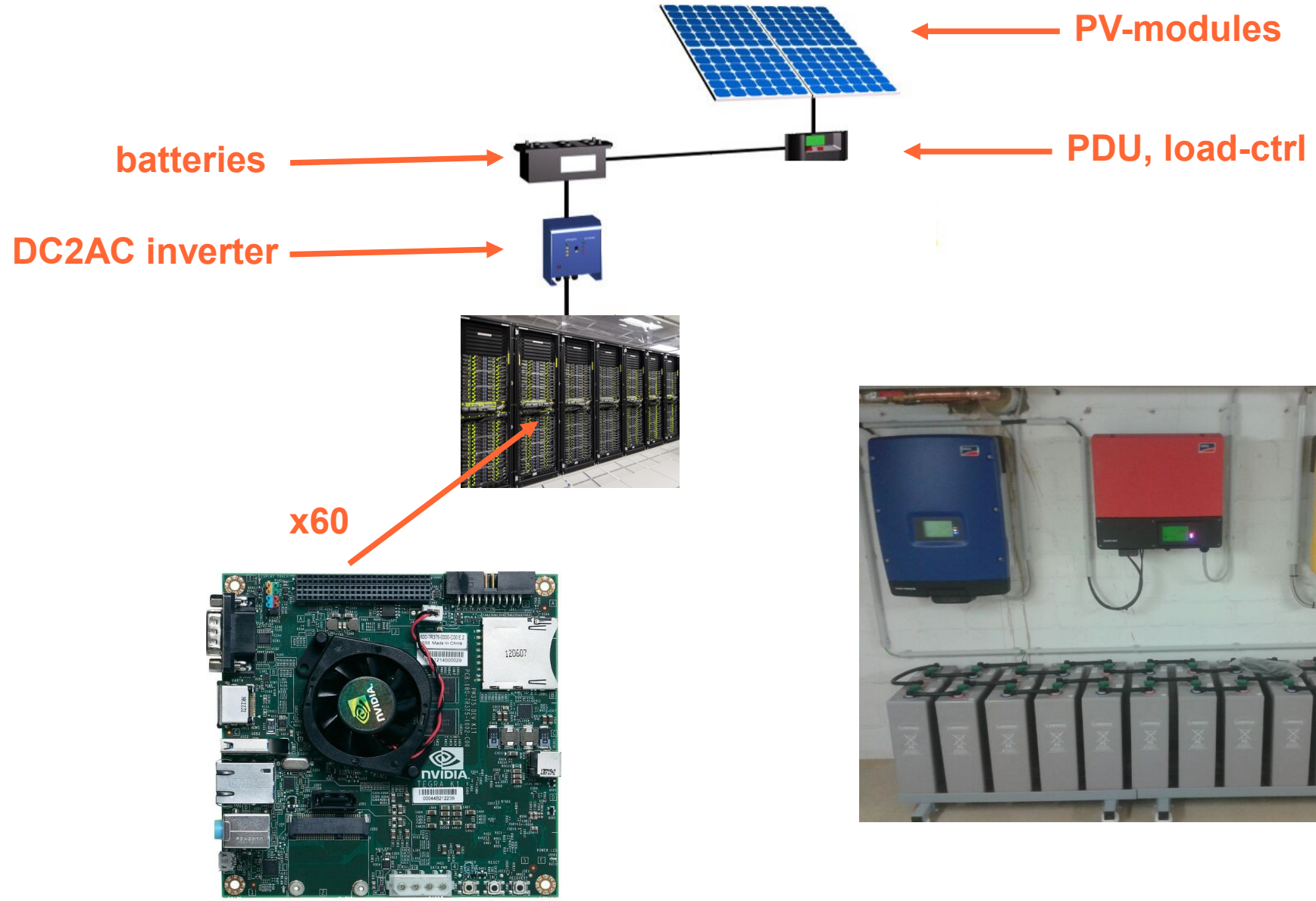


motivation

- **system integration** for Scientific HPC
  - → high-end unconventional compute hardware
  - → high-end renewable power source (photo-voltaic)
  - → specially tailored numerics and simulation software
- **no future spendings due to energy consumption**
- **SME-class resource: <100K€**
- **Scalability, modular design**
- (simplicity)
- (maintainability)
- (safety)
- ...

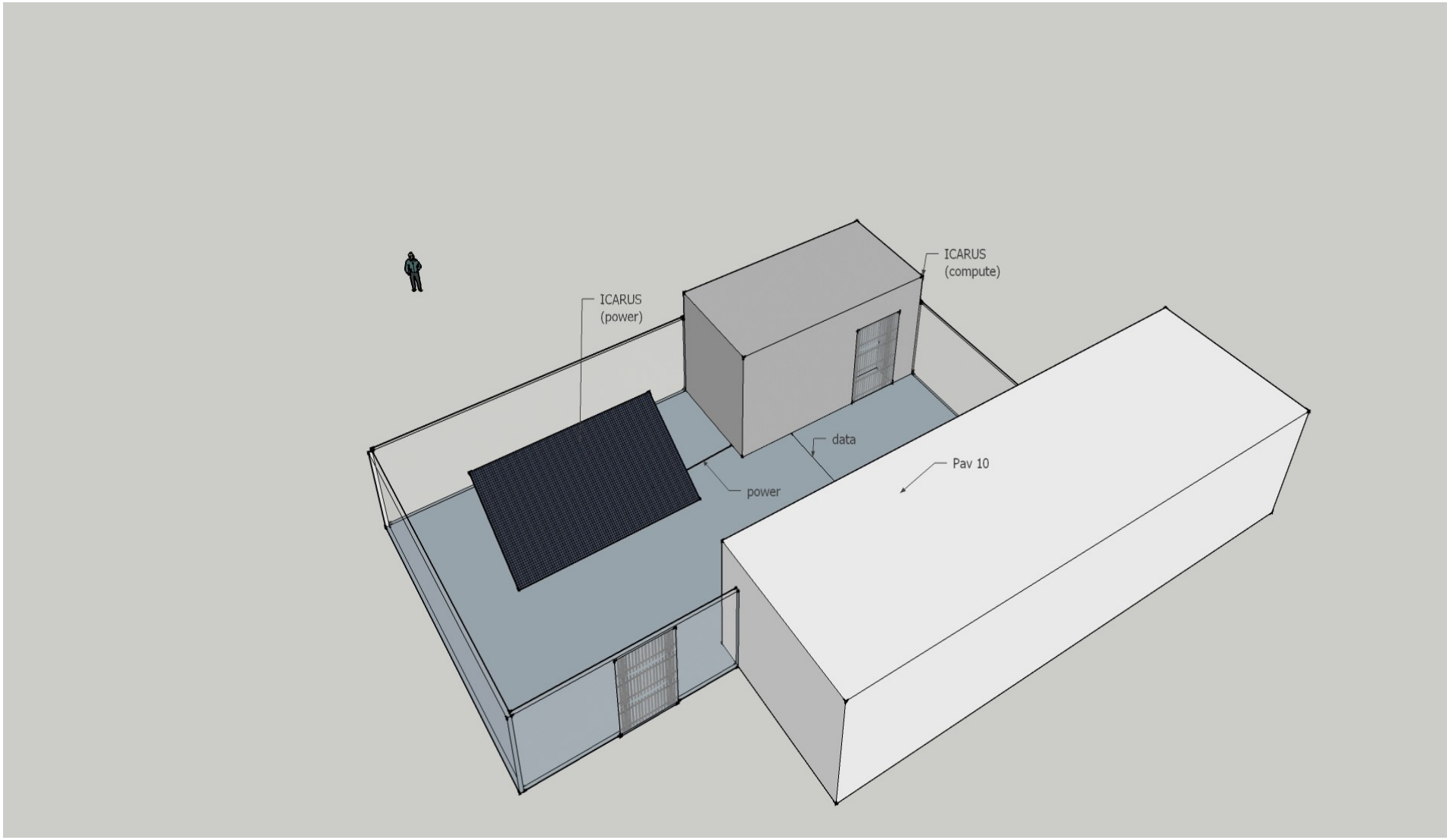
powering the system

Photovoltaic

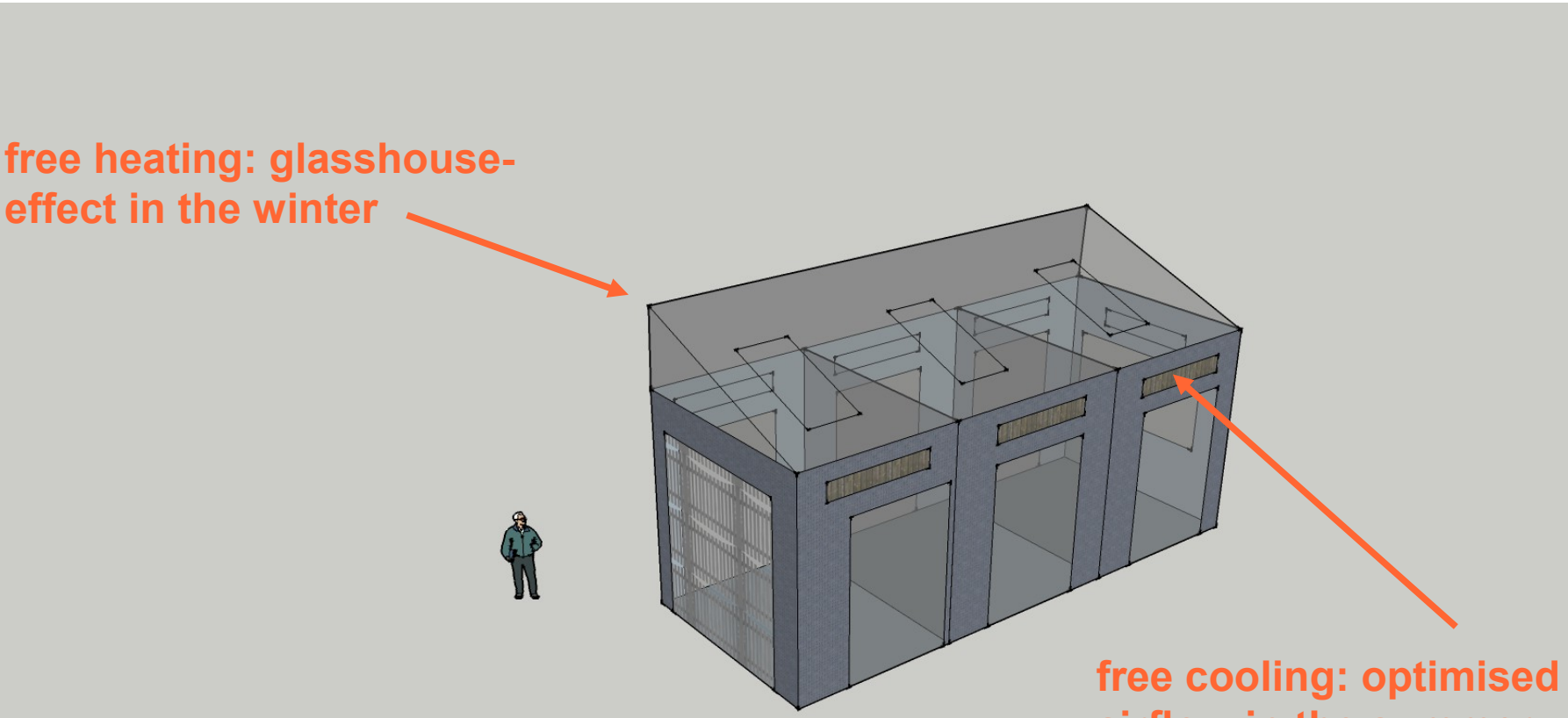


I.C.A.R.U.S construction site

solar modules delivering 6kWp cluster built into container

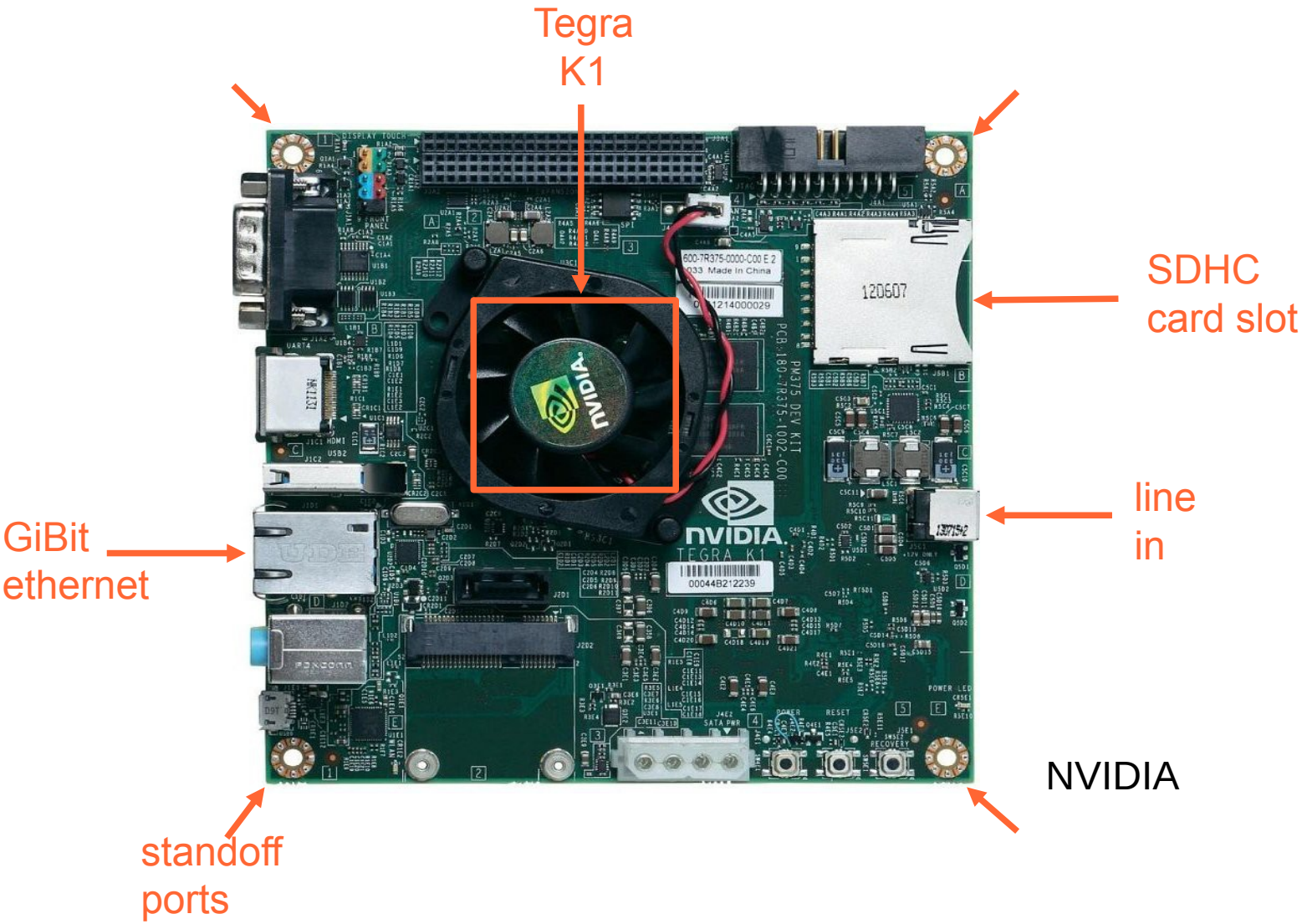


I.C.A.R.U.S climate control



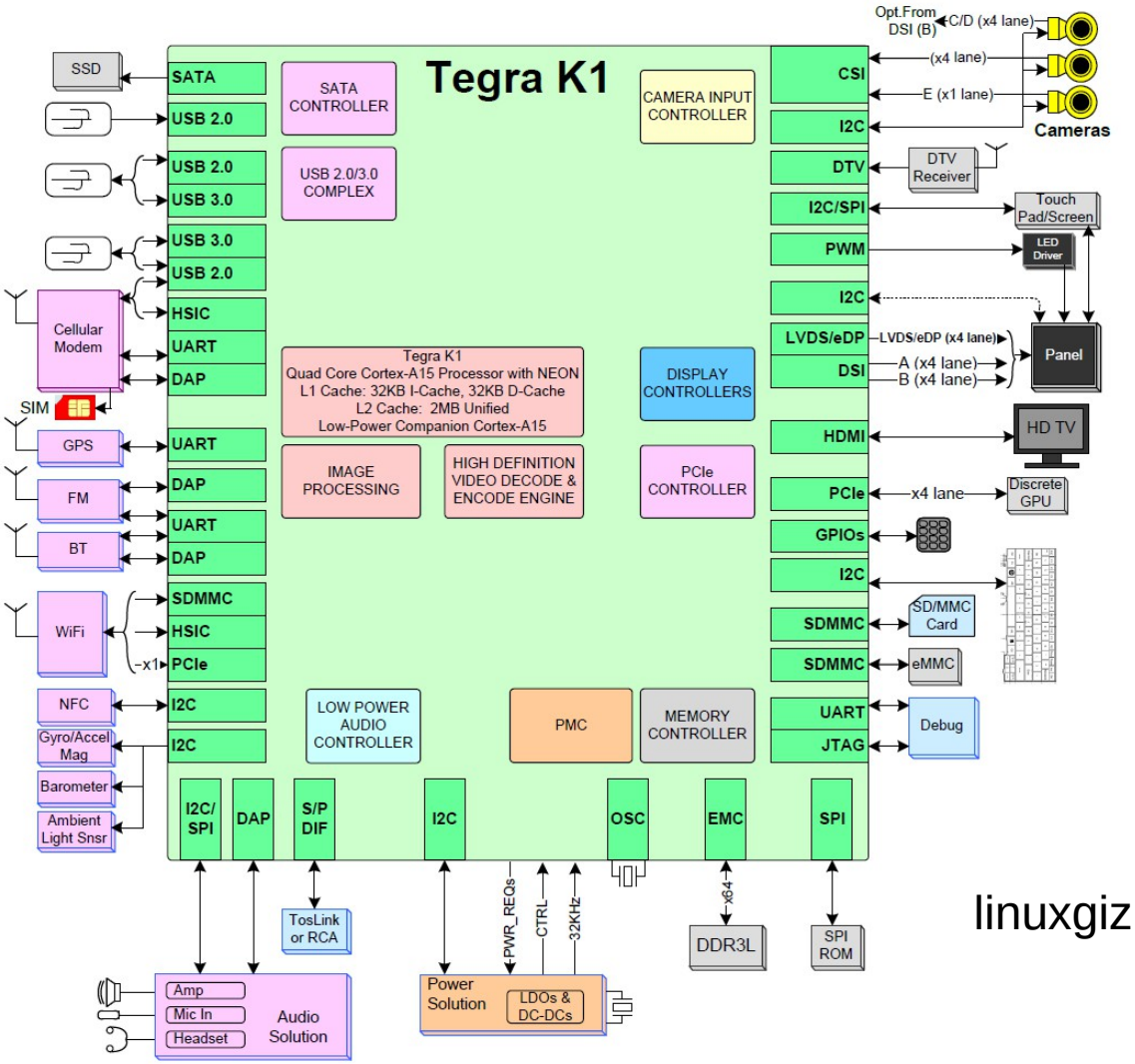
one node of I.C.A.R.U.S.

Jetson TK1



one node of I.C.A.R.U.S.

Tegra K1



linuxgizmos.com

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

whitesheet

→ **nodes:** 60 x NVIDIA Jetson TK 1
→ #cores (ARM Cortex-A15): 240
→ #GPUs (Kepler, 192 cores): 60
→ RAM/core: 2GB LPDDR3
→ Storage: 60x128 GB UHSDHC
→ switches (GiBit Ethernet): 3xL1, 1xL2

→ **cluster theoretical Peak Perf: ~20TFlop/s SP**
→ **cluster Peak Power (including cooling/heating): < 3kW**
→ **cluster powered by renewables**

programming Tegra K1

Jetson carrier board comes with full Linux, CUDA 6.5 currently supported

```
void AXPY<GenericOpenMP>::exec(SimpleVector<float>& r, const SimpleVector<float>& a, const SimpleVector<float>& x, const SimpleVector<float>& y)
{
      const unsigned long i_end(r.size);
      float* __restrict__ __attribute__ ((aligned (16))) rd(r.data);
      float* __restrict__ __attribute__ ((aligned (16))) ad(a.data);
      float* __restrict__ __attribute__ ((aligned (16))) xd(x.data);
      float* __restrict__ __attribute__ ((aligned (16))) yd(y.data);
      unsigned long i;
#pragma omp parallel for schedule(static) shared(rd,ad,xd,yd)
#pragma ivdep
#pragma vector always
      for( i = 0 ; i < i_end ; ++i)
      {
        rd[i] = ad[i] * xd[i] + yd[i];
      }
}
```

```
void AXPY<NEON>::exec(SimpleVector<float>& r, const SimpleVector<float>& a, const SimpleVector<float>& x, const SimpleVector<float>& y)
{
      const int i_end(r.size);
      float* __restrict__ __attribute__ ((aligned (16))) rd(r.data);
      float* __restrict__ __attribute__ ((aligned (16))) ad(a.data);
      float* __restrict__ __attribute__ ((aligned (16))) xd(x.data);
      float* __restrict__ __attribute__ ((aligned (16))) yd(y.data);
      for(int i(0) ; i < i_end ; i += 4)
      {
        float32x4_t rdv, adv, xdv, ydv;

        adv = vld1q_f32(&ad[i]);
        xdv = vld1q_f32(&xd[i]);
        ydv = vld1q_f32(&yd[i]);

        rdv = vmlaq_f32(ydv, xdv, adv);

        vst1q_f32(&rd[i], rdv);
      }
}
```

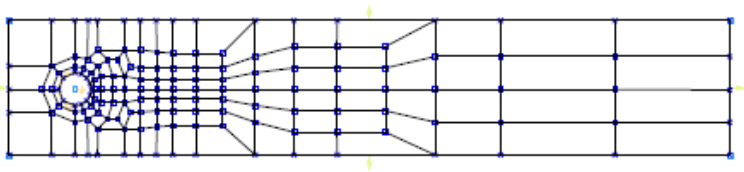
programming Tegra K1

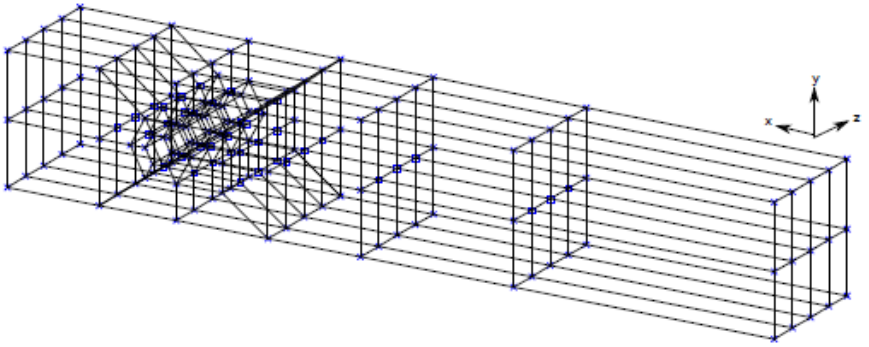
Jetson carrier board comes with full Linux, CUDA 6.5 currently supported

```
template <typename DT_, typename IT_>
__global__ void cuda_product_matvec_ell(DT_ * r, DT_* x, DT_* Ax, const IT_ * Aj, const IT_ * Arl, IT_ stride, IT_ rows)
{
  const IT_ idx = threadIdx.x + blockDim.x * blockIdx.x;
  if (idx >= rows)
    return;

  const IT_ * tAj(Aj);
  const DT_ * tAx(Ax);
  DT_ sum(0);
  tAj += idx;
  tAx += idx;

  const IT_ max(Arl[idx]);
  for(IT_ n(0); n < max ; n++)
  {
    const DT_ A_ij = *tAx;

    const IT_ col = *tAj;
    sum += A_ij * x[col];

    tAj += stride;
    tAx += stride;
  }
  r[idx] = sum;
}
```

→ or cuBLAS, cuSPARSE when applicable

benchmark setup





$$\begin{cases} -\Delta u = 1, & \mathbf{x} \in \Omega \\ u = 0, & \mathbf{x} \in \Gamma_1 \\ u = 1, & \mathbf{x} \in \Gamma_2 \end{cases}$$

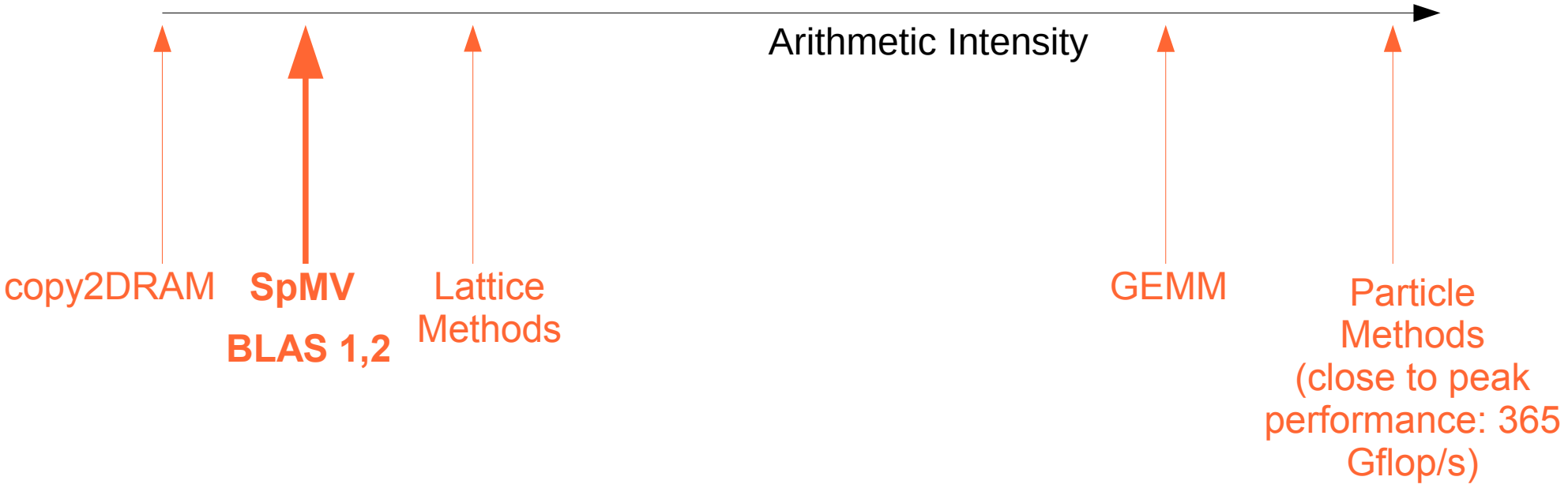
Intel(R) Core(TM) i5-3470 Ivybridge CPU @ 3.20GHz + GeForce GTX 660

V

Tegra K1 on Jetson: ARMv7 Cortex-A15, Kepler ULP GPU

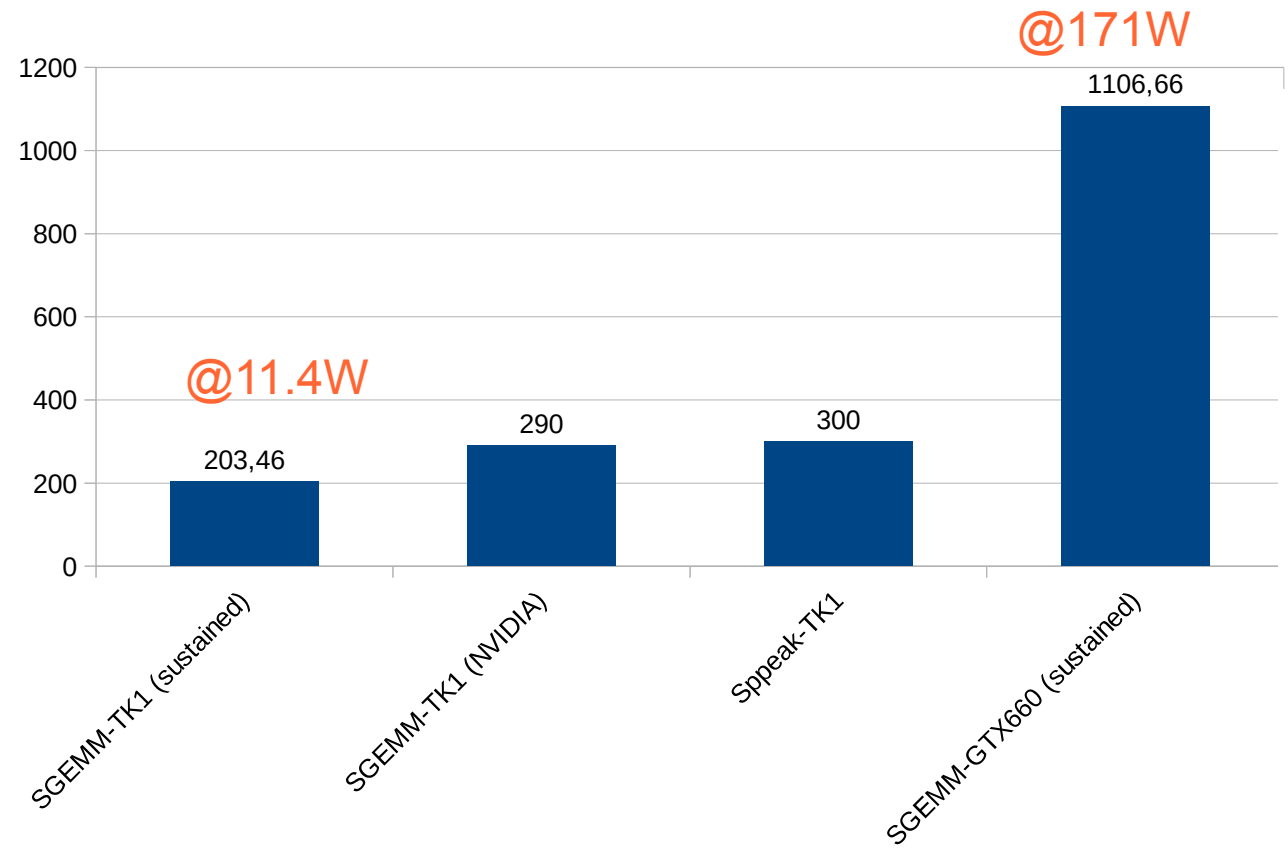
→ energy measurements: 'boxed external' with high resolution (0.3W)
→ baselines: 3.7W (Jetson TK1), 59W (CPU+GPU box)

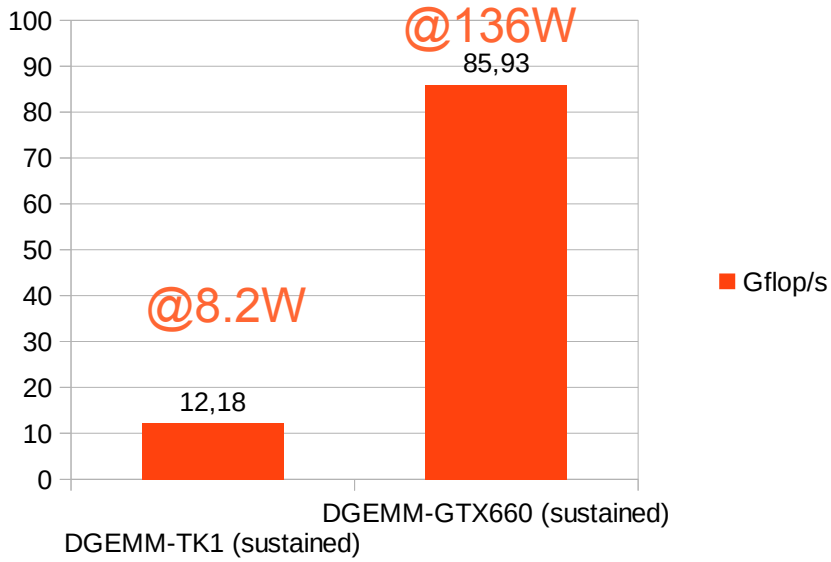
what we can expect...



power consumption and performance of basic kernels

S/DGEMM

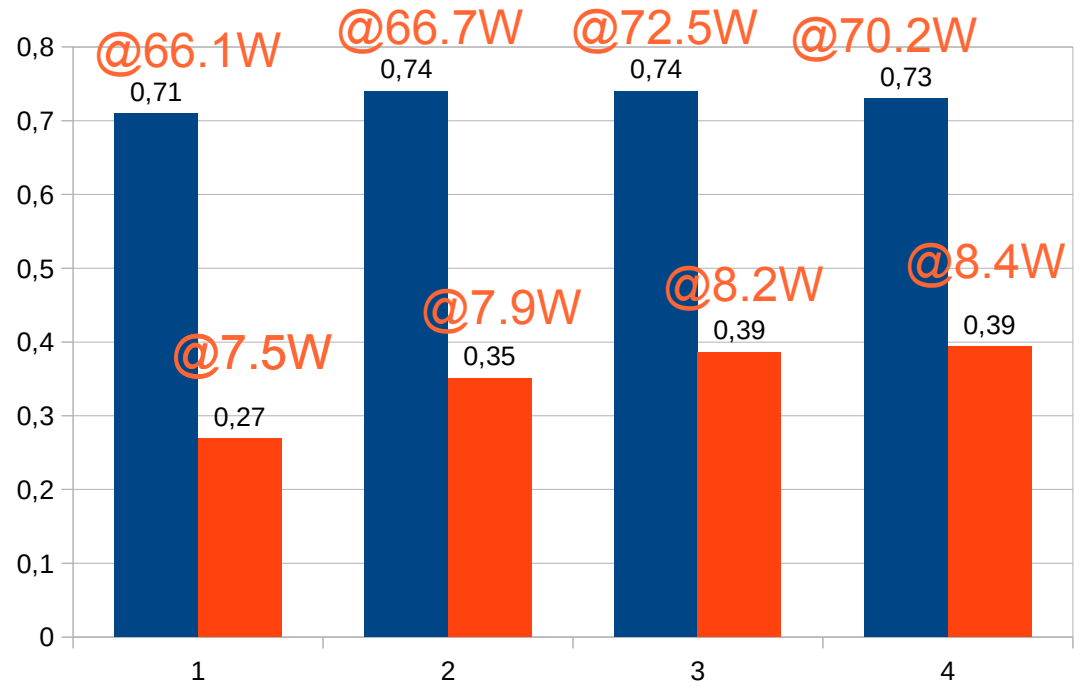


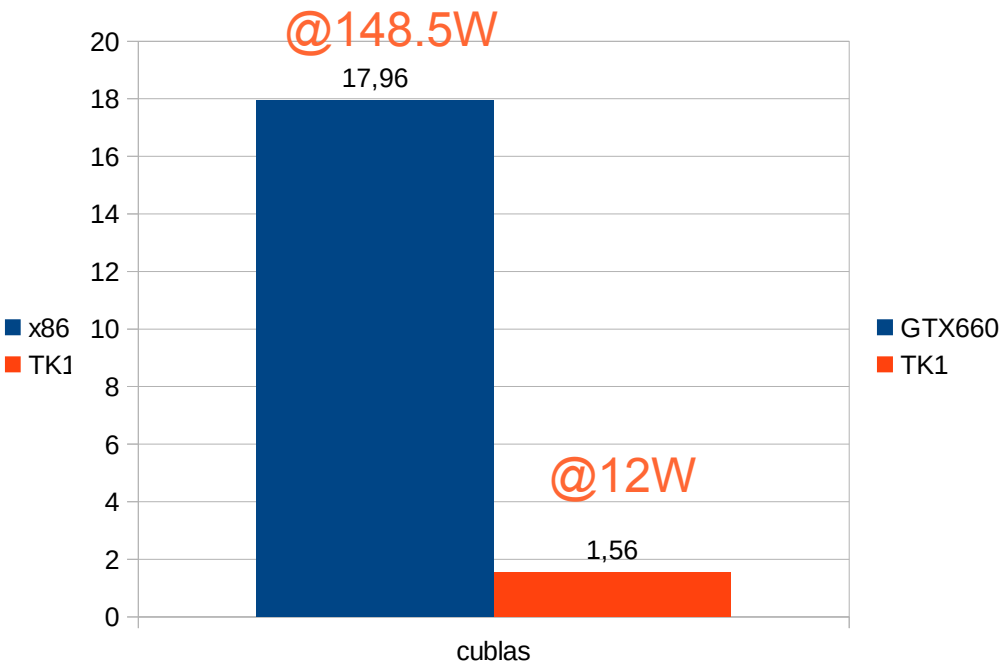


→ TK1: **17.85 Gflop/s/W** SP, **1.49 Gflop/s/W** DP
→ GTX660: **6.47 Gflop/s/W** SP, **0.631 Gflop/s/W** DP
→ why SP matters: we can use mixed precision methods on a node
→ (Jetson) TK1 is 2-3 times better in this metric

power consumption and performance of basic kernels

SAXPY (float from now: mixed precision)

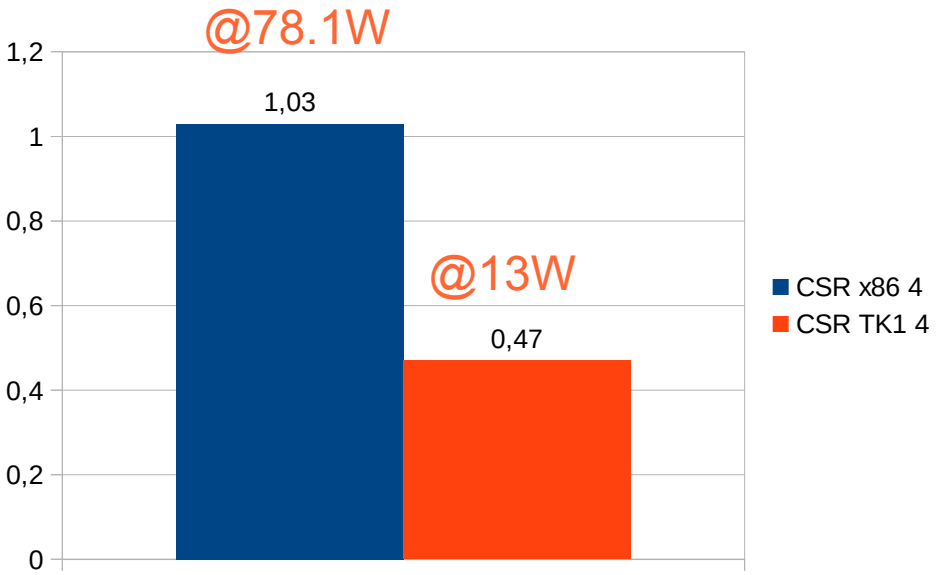


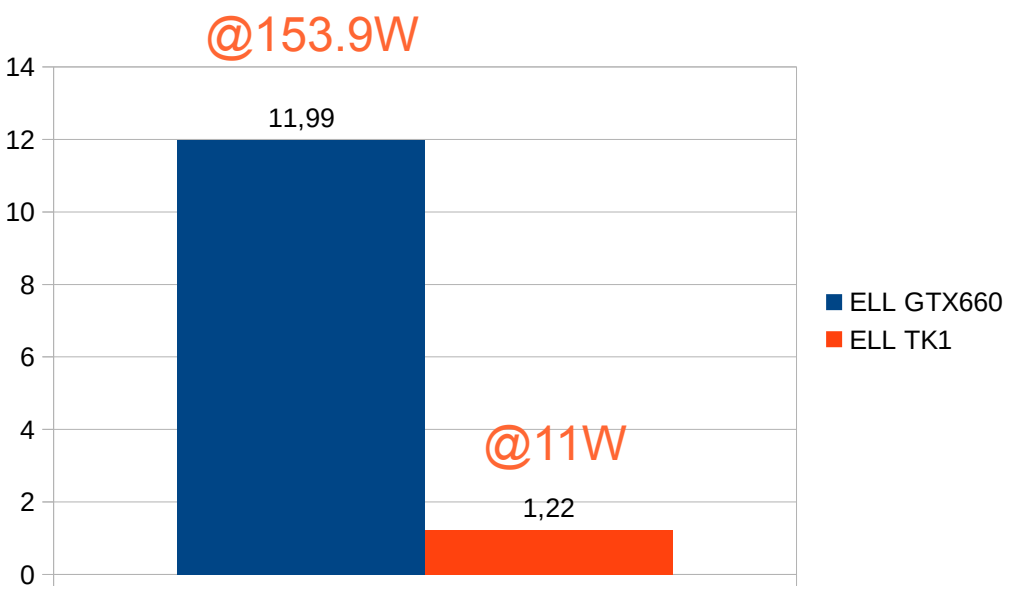


→ core occupancy can be seen in power consumption
→ Cortex-A15: **0.05 Gflop/s/W**
→ core i5: **0.01 GFLop/s/W**
→ TK1-Kepler: **0.13 Gflop/s/W**
→ GTX660: **0.12 Gflop/s/W**

power consumption and performance of basic kernels

SpMV SP

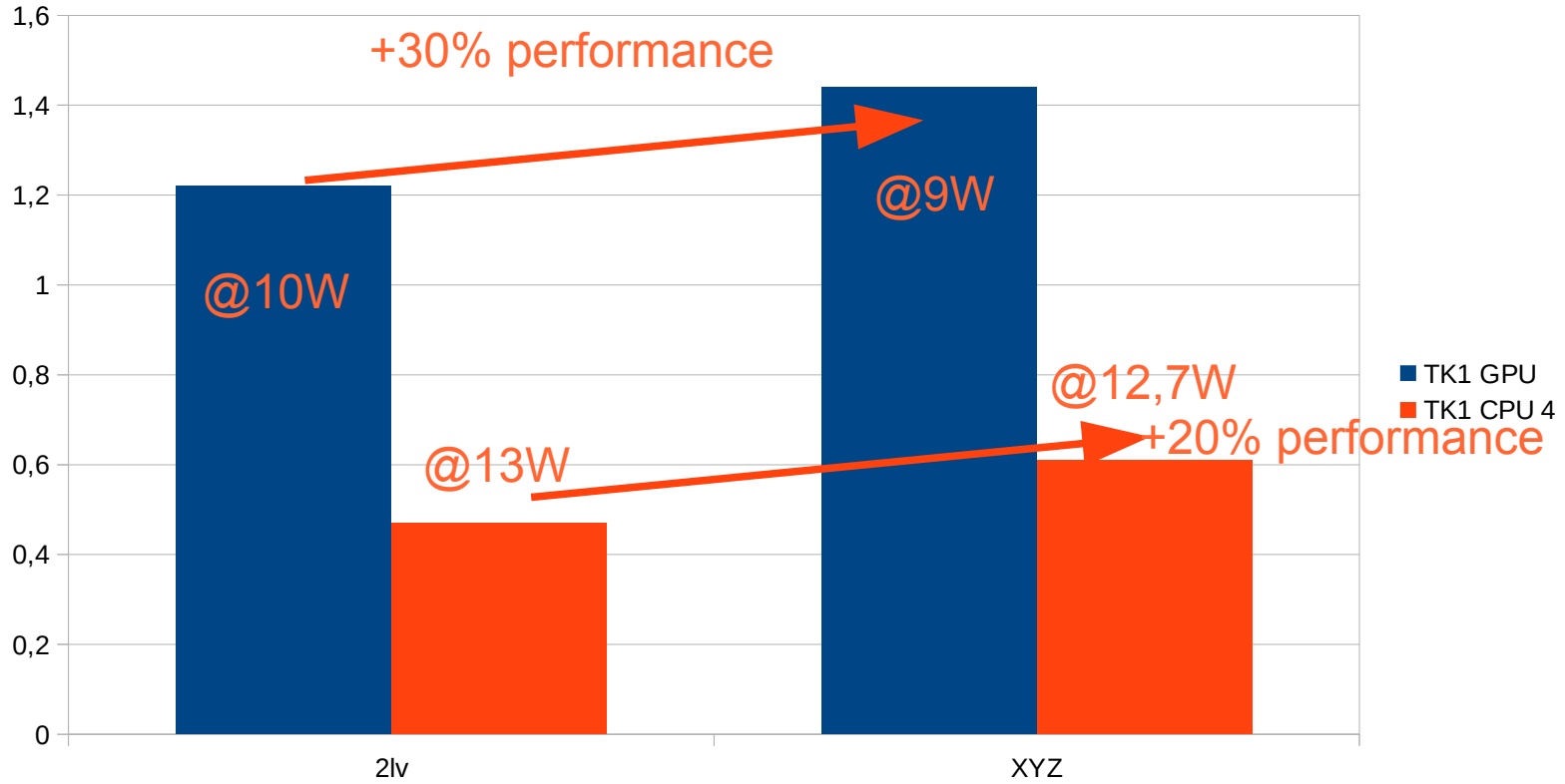




→ Cortex-A15: **0.036 Gflop/s/W**
→ core i5: **0.013 GFLop/s/W**
→ TK1-Kepler: **0.11 Gflop/s/W**
→ GTX660: **0.077 Gflop/s/W**

power consumption and performance of basic kernels

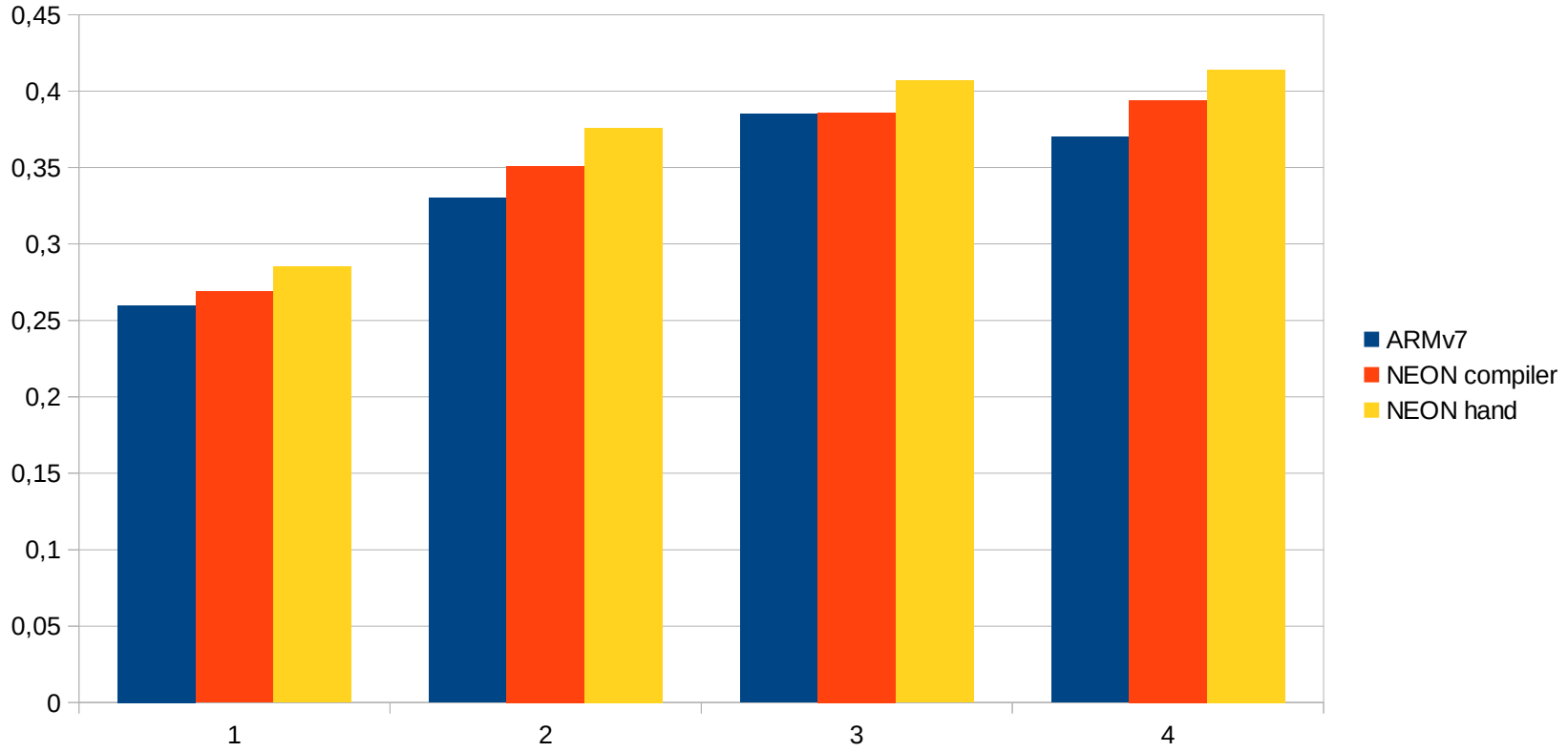
SpMV: remember HWON: DOF sorting



→ performance engineering concepts remain valid
→ power consumption seems to reduce slightly due to matrix bandwidth optimisations
→ on chip / chip2DRAM memory traffic expensive in terms of energy as well
→ same on x86 / GTX but with lower rates

power consumption and performance of basic kernels

SIMD on Tegra K1 (AXPY)



→ NEON: 128 Bit non-IEEE compliant SIMD unit of the Cortex-A15, SP only
→ ARMv7 FPU can do it with same amount of cycles
→ only slight IL improvements → disappointing

energy cost

SAXPY E[Ws]

x3

#DOFs	534144			
	TK1 CPU 4	TK1 GPU	x86	GTX660
WCT	0,0103	0,0027	0,0057	0,00059
P	9,1	11	70,2	148,5
E	0,09373	0,0297	0,40014	0,087615
E / DOF	1,75E-007	5,56E-008	7,49E-007	1,64E-007

x3 - x4

energy cost

SpMV

X1.5 GPU v GPU

#DOFs	534144	2LV		
	TK1 CPU 4 CSR	TK1 GPU ELL	x86 CSR	GTX660 ELL
WCT	0,103	0,0078	0,0092	0,00079
P	13	10	78,1	153,9
E	0,2639	0,078	0,71852	0,121581
E / DOF	4,94E-007	1,46E-007	1,342E-006	2,28E-007

#DOFs	534144	XYZ		
	TK1 CPU 4 CSR	TK1 GPU ELL	x86 CSR	GTX6600 ELL
WCT	0,0155	0,0066	0,0084	0,00077
P	12,7	10,4	79,1	154,4
E	0,19685	0,06864	0,66444	0,118888
E / DOF	3,68E-007	1,28E-007	1,24E-006	2,23E-007

energy cost

SpMV: regular refinement: cost factor between levels in gMG

#DOFs	534144	XYZ		
	TK1 CPU 4 CSR	TK1 GPU ELL	x86 CSR	GTX660 ELL
WCT	0,0155	0,0066	0,0084	0,00077
P	12,7	10,4	79,1	154,4
E	0,19685	0,06864	0,66444	0,118888
E / DOF	3,68E-007	1,28E-007	1,24E-006	2,23E-007

speedup = energy-down = 4

#DOFs	135952	XYZ		
	TK1 CPU 4 CSR	TK1 GPU ELL	x86 CSR	GTX6600 ELL
WCT	0,0038	0,0021	0,0021	0,00018
P	12,7	11	78,4	155
E	0,04826	0,0231	0,16464	0,0279
E / DOF	3,55E-007	1,70E-007	1,21E-006	2,05E-007

energy cost

SpMV: higher order FEM

#DOFs	133952	XYZ	Q1	
	TK1 CPU 4 CSR	TK1 GPU ELL	x86 CSR	GTX660 ELL
WCT	0,0038	0,0021	0,0021	0,00018
P	12,7	11	78,4	155
E	0,04826	0,0231	0,16464	0,0279
E / DOF	3,55E-007	1,70E-007	1,21E-006	2,05E-007

speeddown = energy-up = 2 - 3

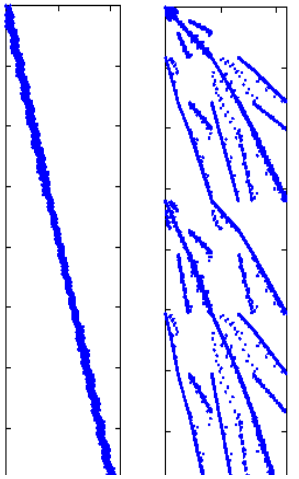
#DOFs	534144	XYZ	Q2	
	TK1 CPU 4 CSR	TK1 GPU ELL	x86 CSR	GTX6600 ELL
WCT	0,0076	0,00522	0,00431	0,00058
P	11	11	78	155
E	0,0836	0,05742	0,33618	0,0899
E / DOF	1,57E-007	1,07E-007	6,29E-007	1,68E-007

energy cost

SpMV: grid transfer in Geometric Multigrid

$$(P_{2h}^h)_{ij} = \varphi_{2h}^{(j)}(\xi_h^{(i)})$$

$$R_h^{2h} = (P_{2h}^h)^T$$



#DOFs	534144	XYZ		
	TK1 CPU 4 CSR	TK1 GPU ELL	x86 CSR	GTX660 ELL
WCT	0,0051	0,0026	0,003	0,00029
P	11	11	78	155
E	0,0561	0,0286	0,234	0,04495
E / DOF	1,05E-007	5,35E-008	4,38E-007	8,42E-008
speedup A	3,03	2,53	2,8	2,65

performance model for E to solution

Richardson (as smoother in gMG) with simple and SPAI preconditioner

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k + \omega M(\mathbf{b} - A\mathbf{x}^k)$$

$$\| I - MA \|_F^2 = \sum_{k=1}^{n} \| e_k^{\mathrm{T}} - m_k^{\mathrm{T}} A \|_2^2 = \sum_{k=1}^{n} \| A^{\mathrm{T}} m_k - e_k \|_2^2$$

$$min_{m_k} \| A^{\mathrm{T}} m_k - e_k \|_2, \quad k = 1, \ldots n.$$

→ one Richardson iteration at the cost of two general SpMV (SPAI-1) or 1 SpMV + 1 AXPY (SPAI-0, Jacobi)

performance model for E to solution

GMG V-cycle

→ s x Richardson iteration, followed by restriction per level → l = L

→ l = L -1, E(L-1) = E(L)/4

→ coarse solver (usually Krylov, here: PCG) …

→ 2 x SpMV + #iters x → l = 1, E(1)

→ 2 x SpMV + 4 x Dot, + 3 x AXPY …

→ l = L -1, E(L-1) = E(L)/4

→ prolongation followed by s x Richardson iterations per level → l = L

performance model for E to solution

GMG

JAC	#iters	P peak	E cycle	E
TK1 CPU	17	11	6,35E+000	1,08E+002
TK1 GPU	17	11	2,18E+000	3,70E+001
x86	17	78	2,33E+001	3,97E+002
GTX660	17	155	2,28E+001	3,88E+002

performance model for E to solution

GMG

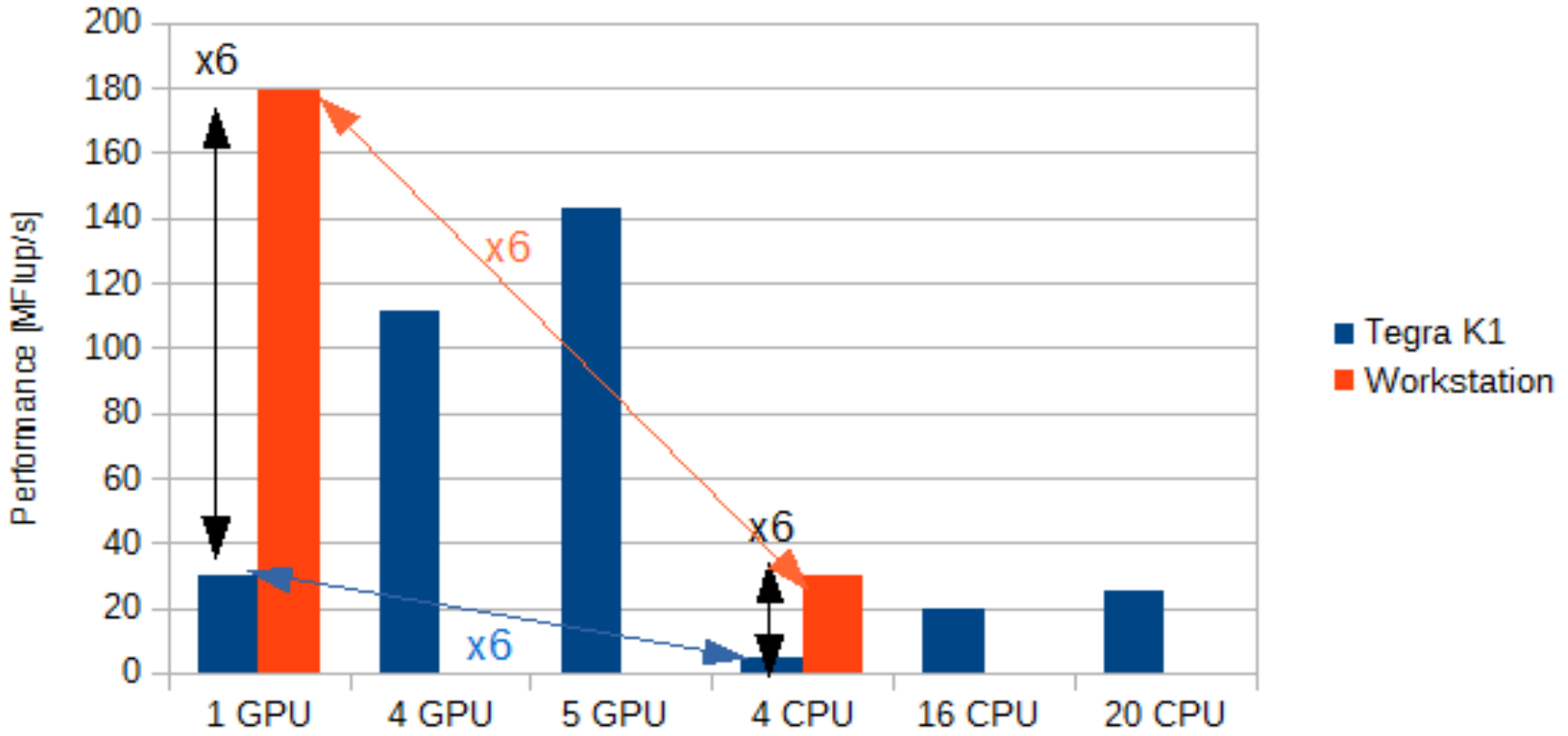
SPAI	#iters	P peak	E cycle	E
TK1 CPU	7	11	8,55E+000	5,99E+001
TK1 GPU	7	11	3,01E+000	2,11E+001
x86	7	78	2,90E+001	2,03E+002
GTX660	7	155	2,85E+001	1,99E+002

→ 10 x less energy to solution than commodity hardware
→ 2 x less energy than simple smoother case

I.C.A.R.U.S. 'eval stage'

going multi-node

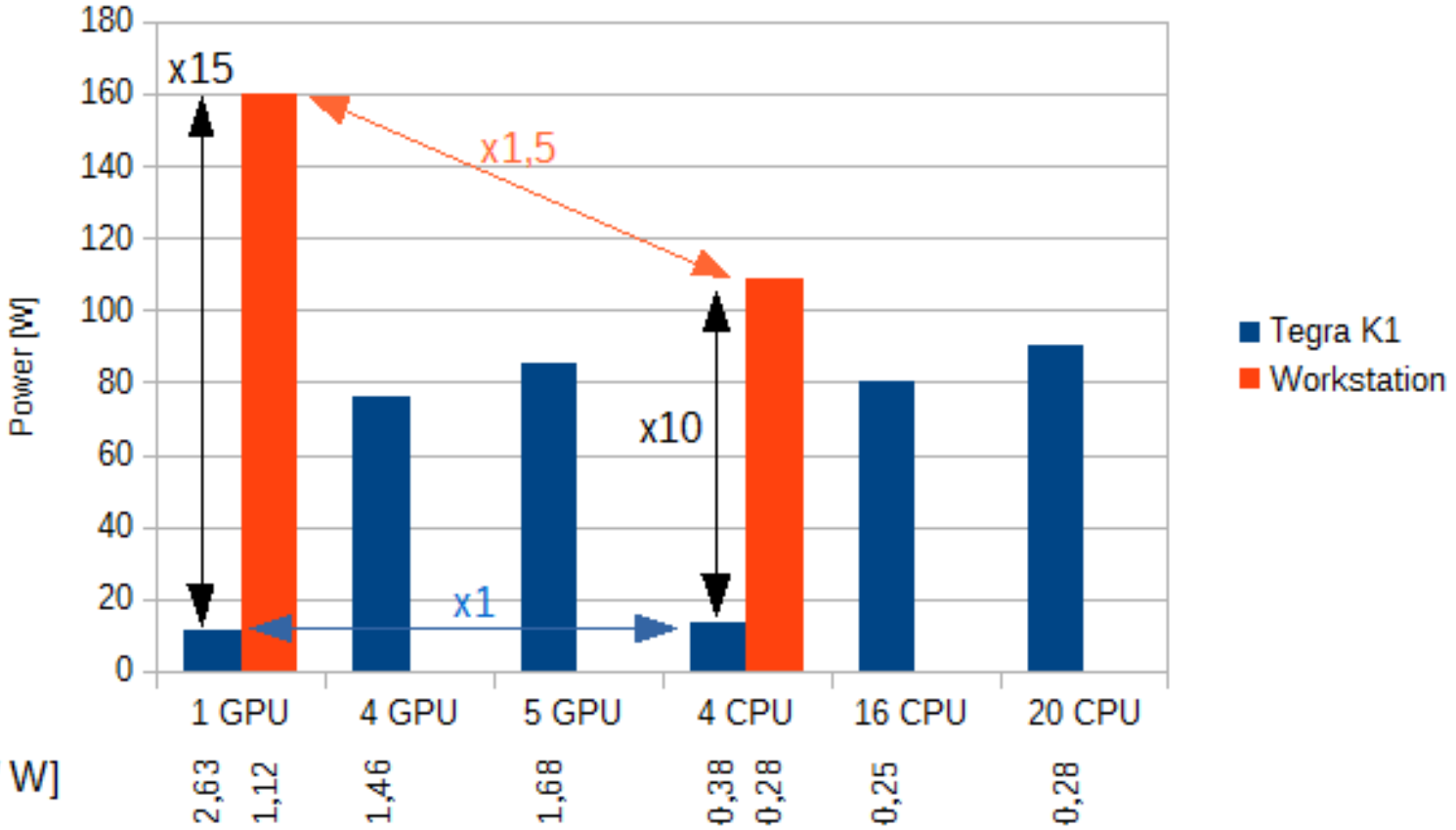
flow solver on I.C.A.R.U.S. (Tier-0), FEAT software family



→ prediction: with 7 Jetson boards, we can beat this GPU

going multi-node

flow solver on I.C.A.R.U.S. (Tier-0), FEAT software family



→ prediction: with 8 Jetson boards + switch, we can beat this GTX GPU
→ this would take 123 W (switches increase baseline)
→ energy efficiency can be transported to the cluster level

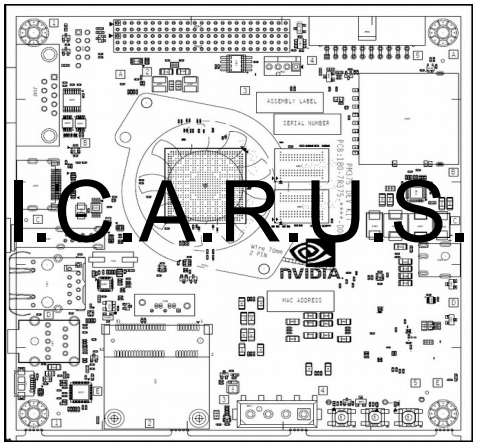
conclusions

→ energy efficiency can be improved by using TK1 SoCs
  → closely coupled to performance: HWON leads to benefits
  → EE scales on the cluster

  → simple performance modelling for energy to solution can be used to predict energy to solution of complex kernels
    → based on critical kernels

thank you

- Stefan Turek (scientific supervision)
- Markus Geveler (system design)
- Dirk Ribbrock (system administration)



This work was also supported (in part) by the German Research Foundation (DFG) through the Priority Programme
1648 `Software for Exascale Computing' as well as grant TU 102/50-1.

I.C.A.R.U.S. hardware is financed by MIWF NRW under the lead of MERCUR.