No. 554 January 2017

# How applied sciences can accelerate the energy revolution-A pleading for energy awareness in scientific computing

M. Geveler, S. Turek

ISSN: 2190-1767

# How applied sciences can accelerate the energy revolution - A pleading for energy awareness in scientific computing

Markus Geveler*, Stefan Turek†

## Abstract

In this paper we propose a course of action towards a better understanding of energy consumption-related aspects in the development of scientific software as well as in the development and usage of 'unconventional'compute hardware in applied sciences. We demonstrate how the applied sciences community can make a significant contribution in reducing the energy footprint of their computations.

**Keywords:** energy efficient computing, performance engineering, unconventional computer hardware, embedded hardware, renewable energy

## 1 Introduction

### 1.1 Scientific computing cannot continue to be done the way it has been

There is no denial that the transition from nuclear- and fossil-driven energy supplies to more sustainable options is one of the most challenging tasks of our time. While this transitioning is usually referred to as 'the energy revolution' its basic pillars are not limited to alternative energy production but also are agreed to include better energy grids as well as more energy-efficient *consumers*. In this sense computing in general and particularly scientific computing as the basic tool for applied sciences contain a great deal of energy consumption since the computers (devices), compute clusters and data-/compute centers do.

Recently the Semiconductor Industry Association (SIA) released a report where a prediction on the world's total energy production was made alongside an analogous prediction for the energy consumption of the world's computers [SIA 2015]. In order to demonstrate the urgency of what shall later be proposed here let us first summarize these findings. For this purpose consider Figure 1. Here the world's total energy production is extrapolated with a comparatively slow increase leveling out at approximately one Zettajoule in 2015. Speaking on these scales this value is not expected to increase much in this century. On the demand side energy consumption due to usage of computers is expected to increase much faster: Based on current technology (with today's (digital) computer technology and the way single devices are built and clustered to larger units) the total energy consumed (only by computing) will exceed the world's

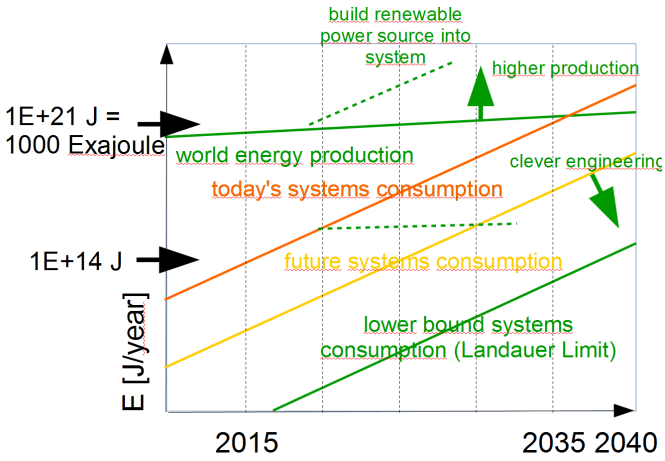


**Figure 1:** *Prediction of world's total energy production and overall energy consumption by computing*

energy production and supply by around 2040. Even based on extrapolating current technology to a hypothetical future technology level (mainly based on improved manufacturing processes leading to smaller transistors) this would lead to only an insignificant delay of this point in time. Consider this as the catastrophe that it is: No additional electrical device would be able to be plugged into the power grid any more. Applications on the other side are continuing to increase their hunger for ever more computational resources. Consider the lowest plot in Figure 1. It represents a theoretical lower bound for the energy consumption per year based on a hypothetical device that needs the minimum amount of energy to flip a bit of information. This limit is called the Landauer Limit. Due to an increasing demand for computational capacities this limit will also be increasing over time since more and more computers will be built. It is more or less increasing with the same rate as today's or close future's computer systems' consumption and such optimal computers would therefore also only postpone the inevitable.

### 1.2 Hardware and software in applied sciences are blind on the 'energy eye'

Today's compute and data centers mostly rely on massively parallel distributed memory clusters. The compute nodes are also multi-level parallel and heterogeneous. They usually comprise one or more high-end server CPUs based on the x86, Power, or SPARC architectures optionally accelerated by GPUs or other (accelerator) hardware. Large HPC sites of this type have substantial energy requirements so that the associated expenses over the lifetime of the system may exceed the initial acquisition costs. In addition, the energy supply for supercomputers is not always an integral part of its overall design - consumers (such as the compute cluster, cooling, networking, management hardware) are often developed independently from the key technologies of the energy revolution, e.g. renewable energy sources, battery and power grid techniques. Taking a look at the largest supercomputers today it can be observed that as a consequence of decades of performance-centric hardware development there is a huge gap between pure performance and en-

*TU Dortmund, email:markus.geveler@math.tu-dortmund.de
†TU Dortmund, ECCOMAS co-chairman for Scientific Computing, email:ture@featflow.de

ergy efficiency in these designs: The Top500 list's best performing HPC system (dissipating power in the 20 Megawatts range making a power supply by local solar farming for instance an impossible-to-achieve aim) is only ranked 84th on the corresponding Green500 list whereas the most energy-efficient system in place only performs 160th in the metric of raw floating point performance [Meuer et al. 2015; Feng et al. 2015]. It is well known [Schäppi et al. 2009] [Lawrence Berkeley National Laboratory 2006] that 40–60% of the energy consumption of an HPC site can be attributed to the compute nodes (processors and memories).

Using and developing scientific *software* on the other hand requires knowledge of the specific target hardware architecture which implies adjustments of numerical methods and their implementation. Otherwise efficiency losses are axiomatic and always imply too much energy spent. Therefore scientific software should not follow a hardware-oblivious design paradigm. In the resulting performance engineering studies for decades energy efficiency has been eclipsed by computational performance and only recently power and energy metrics started being included into performance models for numerical software [Hager et al. 2014; Benner et al. 2013; Anzt and Quintana-Ortí 2014; Malas et al. 2014]. This in-depth understanding of energy and power requirements of different classes of scientific applications is essential on two levels: First application developers and users must control the overall (energy) costs; second HPC site operators (such as data centers / universities) have to identify the most economical way to operate their computational facilities. Both levels are related since the application user is very much interested in efficient utilization of available computational resources whereas system operators are just as motivated that platform-optimized application codes are being used.

### 1.3 Consequences and paper contribution

From the previous section we can find two major facts: (1) With current knowledge and (digital) computer technology there is no instant solution available to a possible 'black out'situation in scientific computing: Providing better devices only leads to a global energy consumption converging to a theoretical limit that also ultimately leads to an energy shortage in the mid 21st century. Remember that these numbers are only for computer devices and do not even cover all other energy consumers. (2) For too many years performance engineering and hardware engineering has been eclipsed by the misdirected longing for ever more performance where *faster* seemed to be the only paradigm. A movement towards incorporation of power and energy into performance models gains momentum but resulting efforts are often limited to simple, basic kernels and not very visible in applied sciences.

In the following sections we describe how the applied sciences community can contribute to tackle the fundamental problem of limited future energy supplies for (scientific) computing. This is achieved by presenting a simple course of action in scientific hardware usage and software development. This requires thinking more 'out-of-the-box'in two aspects: Application software usage and development as well as hardware usage and development.

## 2 Ways to tackle a future energy crisis in scientific computing

### 2.1 Hardware-oriented Numerics revisited

The major aspect in performance engineering for energy efficiency is the numerical methods used in application and their implementation: (1) **'Classical'performance engineering can be applied to enhance the efficiency of the current method on the target hardware** and/or in many cases **numerical alternatives can be found that might better fit to the hardware in use** and/or (2) **other numerical methods can be found to improve the numerical efficiency**. Both are heavily interdependent: Overall tuning in (1) might have negative effects on the numerical scaling whereas improving in (2) often results in numerically stronger but slower/less hardware-efficient methods. Tuning both simultaneously is what we used to call *hardware-oriented numerics* [Turek et al. 2006; Turek et al. 2010; Geveler et al. 2013]. Now we plead for adding a new dimension to this former dualism of hardware and numerical efficiency: (3) energy-efficiency. Although tuning (1) and (2) normally leads to improvements in (3) this is not always the case as we exemplify in the following section. In this example everything essentially breaks down to powering the memory interface. This is a very representative case for many simulation codes.

It also shows that for all improvements of application codes proper *performance modeling* is key. A performance model is intended to predict performance in some metric, i.e. number of floating point operations per time unit or its inverse: time to solution. The prediction can then be related to obtained measurements (for instance execution wall clock times) and the information one gains is how good a given code performs on the hardware compared to sustainable performance on that hardware. We demonstrate how such a model can be derived empirically that is, by taking a few time and power measurements.

### 2.2 Taking control of energy consumption on the application level

The importance of modeling energy to solution in scientific codes can be illustrated by a simple example: Many performance-critical kernels of numerical partial differential equation (PDE) solvers involve stencil discretizations or large linear equation systems. Such applications tend to be memory bandwidth-bound that is the overall computational intensity is comparatively small. As a consequence any increase in the number of parallel cores reduces the core saturation resulting in poor parallel scaling i. e., there exists a certain number of cores $N_{\text{max}}$ for which adding another core produces marginal performance gains. From the user's point of view (and even from the classical performance-engineers' point of view) running the application with $N_{\text{max}}$ or $N_{\text{max}} + 1$ shouldn't make a difference because there is no loss in performance.

Additional cores however usually dissipate the same power (for a short period of time) as fully saturated ones making the performance gains infinitely disproportional to energy consumption rises. In other words: With regard to the energy-efficiency there is an optimal number of cores $N_{\text{opt}}$ that is usually smaller than the maximum number of available cores $N_{\text{cores}}$ (because of the saturated memory interface). Since the majority of codes are still lacking reliable energy models running an application on $N_{\text{cores}}$ is still a common choice. Although power signatures of sophisticated simulation codes as well as the hardware behavior on the specific workloads are very complex the measurement, modeling, tuning and prediction of the energy to solution can often be achieved quite easily. In recent work we have shown that this goal can be achieved by only a few power and time measurements on the system level resulting in a robust model that predicts energy consumption of a whole application class as well as prevents the user from choosing the wrong run time parameters such as – like in the scenario rendered above – launching a suboptimal amount of threads on a compute node [Geveler et al. 2016a].

In the following we show how to deduce a model that predicts the energy to solution $E$ for a given CFD code. This model hence is simply an equation, that determines $E$ when running the applica-

tion with certain run time parameters. This model is given in Equation 1. Its construction requires empirical data – in this case, time and energy measurements.

In order to deduce a model that incorporates the energy to solution as a metric for energy efficiency we consider the averaged time to solution for a time step to complete ($T$), the averaged power dissipation during that time ($P$), and the resulting energy to solution ($E$). For this type of application once a problem is big enough to saturate the memory interface its wall clock time can be used to make predictions for other problem sizes because the time to solution is expected to behave as $T(N) = (N/N_{\text{measured}})T_{\text{measured}}$. All three $T$, $P$ and thus $E$ are then functions of the number of used cores, $k$. Additionally, we introduce variables $\Delta T$ for the total decrease in wall clock time and $\Delta P$ for the total increase in power dissipation

$$\Delta T = \Delta T(k) = T_{k-1} - T_k, \qquad \Delta P = \Delta P(k) = P_k - P_{k-1}\,.$$

We summarize some of the performance measurements that lead to the performance model described in the subsequent Equation (1) for a 3D PDE solver for the simulation of global ocean circulation on two different hardware architectures: the Intel Haswell [Intel Corp 2015a; Intel Corp 2015b] and the Cortex-A15, the CPU in an NVIDIA Tegra K1 [NVIDIA Corp 2014] SoC, in Table 1. With these results substantiating our hypothesis that we have in

| $k$ | $T$ [s] | $P$ [W] | $E$ [J] | $\Delta T$ [s] | $\Delta P$ [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 |

**Table 1:** *Performance and power measurements and values of basic model variables*

fact a memory bandwidth-bound situation here the expected solution time for this code (given the problem size and application parameters but independent from the hardware architecture) can be modeled as a super-linear function in the number of used threads with quickly decreasing slopes, that is, $\Delta T(k) = 0$ for moderate values of $k$. Power on the other hand behaves more linearly: First one can notice that when leaving the idle state a comparatively large power jump occurs which can be explained with the CPU being reconnected to the system clock or in other words, with all four cores being provided with a baseline power independent of the core workload denoted by $P_{\text{base}}$. Once at least one core is tasked with a job, i. e., $k \geq 1$, the chipset baseline power is increased by an additional constant power dissipation called $P_{\text{socket}}$. Second it can be seen that $\Delta P$ is roughly 10 W for Haswell and ca. 2 W for Tegra K1. Based on these findings power (and therefore energy) for this application can be modeled as a simple linear function in the number of used cores:

$$E(k) = P(k)T(k) = \left(P_{\text{base}} + P_{\text{socket}} + k\Delta P\right)\, T(k)\,, k \geq 1 \tag{1}$$

The practical effects of this model can be understood better when the above values are displayed differently in an energy / time chart like in Figure 2. Because of an *over-proportional power increase compared to performance increase* there is an optimal number of threads as described above: Using more than three threads is obviously not beneficial and – for one of the two hardware architectures

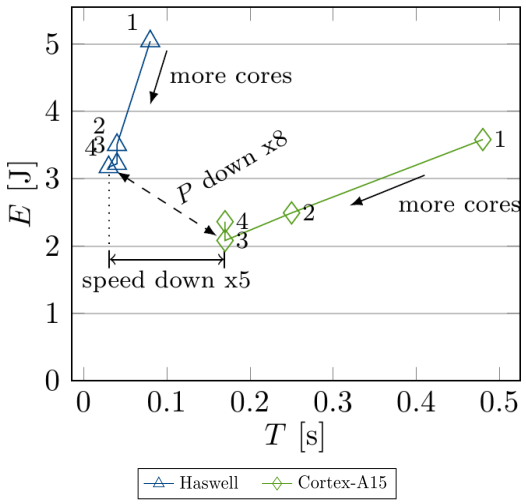


**Figure 2:** *Energy and time to solution of a memory bandwidth-bound application on two different hardware architectures*

– sometimes even comprises an energy efficiency penalty since energy to solution increases with no benefits for the execution time. For many codes this qualitative finding by simple means holds true for many different runs on many nodes of a local cluster or on a workstation offering a great source of energy savings.

Hence we propose a **rigorous performance measurement, modeling and engineering policy in the development of applied sciences codes**.

### 2.3 Taking control of the hardware to target

Let us reconsider Figure 2. Another aspect that is depicted here is that the hardware architectures used for scientific computing are fundamentally different: The two processors used there can be both considered multi-purpose processors with one (the Cortex-A15) not originating from the field of computing. Recently a game-changing impulse regarding energy-efficient compute hardware comes from mobile/embedded computing with devices featuring a long history of being developed under one major aspect: they have had to be operated with a (limited) battery power supply. Hence as opposed to x86 and other commodity designs (with a focus on chipset compatibility and performance) the resulting energy efficiency advantage can be made accessible to the scientific community.

In our example the CFD code shows higher energy efficiency on the mobile CPU than on the commodity CPU. This is possible because comparing the respective run on the two micro architectures the **power down is larger than the speed down**. Obviously this comes with a price: the execution time is larger when using the embedded processor. This exemplifies a general problem: Energy to solution is not only barely visible to the applied sciences community but also minimizing it is not an inherent goal of what is usually done and even worse: Using the most energy-efficient hardware penalizes the user with a slowdown. We shall demonstrate how to overcome this issue by scaling such unconventional hardware resources and numerically scale the code, see below.

However in situations when it is possible in future times **energy efficiency should be favored over performance** and in this case a good knowledge of the energy-related hardware specifics is needed. Luckily the performance modeling described in the above section offers a good way to explore these. Devices of the family used in

the above benchmarks are on the march and are more and more built into servers and data centers. Normally in a local data center in future times or even now there might be a variety of types of compute nodes available to the user or a department comprises several clusters and workstations of different kind. As a basic pattern to reduce the energy footprint of applied sciences work **the performance modeling allows for choosing the most energy-efficient hardware for a workload or application**.

To fortify our point here we demonstrate the performance and energy efficiency on the device level for very different hardware architectures and very different types of codes in Figures 3 and 4.

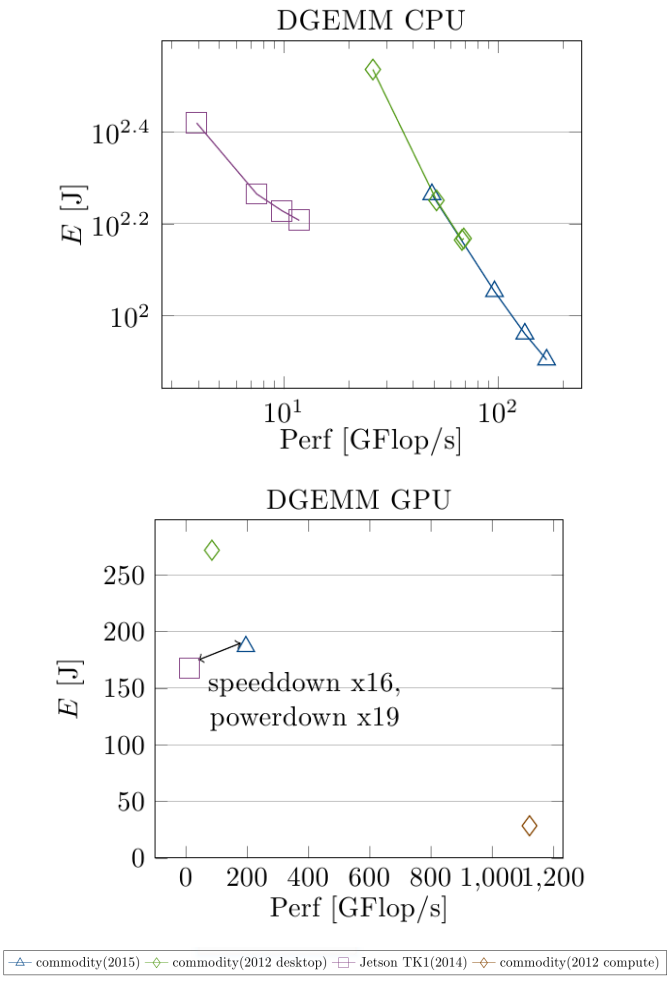


**Figure 3:** *Energy and time to solution of compute-bound basic kernels on different hardware architectures*

The results depict two very well known but often not addressed phenomena: (1) Performance and energy efficiency are functions of device hardware and (the operational intensity of) code; and (2) the hardware ecosystem is evolving very fast over time.

For (1) consider for example the top diagram in Figure 3. The purple plot marks performance for the same ARM Cortex-A15 CPU as in Figure 2. Note that this time for a compute-bound kernel (dense matrix matrix multiply, GEMM) performance and energy efficiency are considerably higher when using commodity processors based on the x86 architecture. Hence over-generalized assumptions like 'ARM-based devices/computers are always more energy-

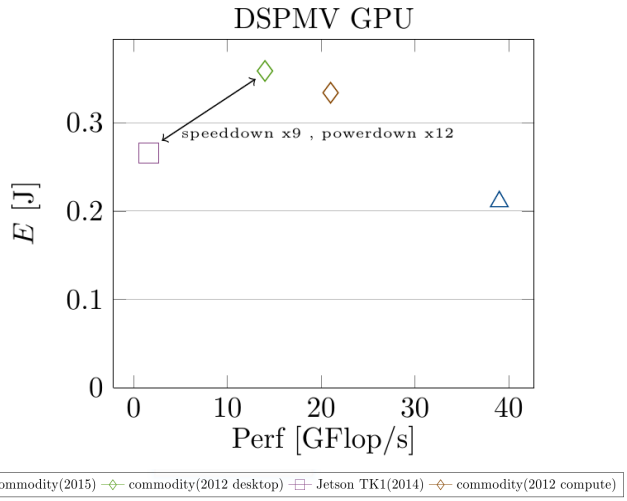


**Figure 4:** *Energy and time to solution memory bandwidth-bound basic kernels on different hardware architectures*

efficient than x86 ones' are as wrong as they are futile. For (2) consider the bottom plot in Figure 3 where this time commodity and embedded GPUs of different hardware generations are compared. Here we examined the low power GPUs of the NVIDIA Tegra-K1 SoC [Geveler et al. 2016b]. Note how the low power GPU is able to beat desktop CPUs of their time as well as later generation desktop GPUs in terms of energy efficiency whereas same-generation compute GPUs (Tesla line) are the most energy efficient floating point accelerators of that time. For a different type of kernel the picture is changing: In the memory bandwidth-bound case in Figure 4 (as with the sparse matrix vector multiply) powering memory interfaces is much more important and thus hardware generation determines the energy-efficiency due to the enhanced memory interface of the 2015 GPU. Also the embedded GPUs are evolving and the next generations (Tegra X1 and X2) may turn this picture upside down again. Hence **continuous work regarding measurement and modeling energy for application codes and available hardware is crucial and nothing should be taken as carved in stone**.

Another point here is that computing is not all about CPUs and memory interfaces. A compute node in a cluster may be a complex architecture which is then aggregated into an even more complex system with communication switches storage and control hardware. This cluster is integrated into a housing with cooling systems. All these systems drain energy for the sake of computations. Let us now take a look at the cluster level where everything is scaled up. In Figure 5 we demonstrate results from a cluster of compute nodes consisting of single Tegra K1 SoCs which combine 4 Cortex-A15 CPU cores and a low power Kepler GPU. These results show that together with energy-efficient switches we can use a number of nodes comprising the 'unconventional' computer hardware to be both more energy-efficient as well as faster. Hence **we can scale the energy-efficiency bonuses by embedded hardware up to a point where the resulting cluster performs faster and uses less energy at the same time as compared to single commodity devices**. It is very important to understand that this is only possible by **developing numerics that fit to the underlying hardware and are able to be scaled numerically and hardware efficiency-wise to reach that point**.

Another aspect concerning hardware is that the energy revolution is not all about energy efficiency. Up to now we basically proposed to

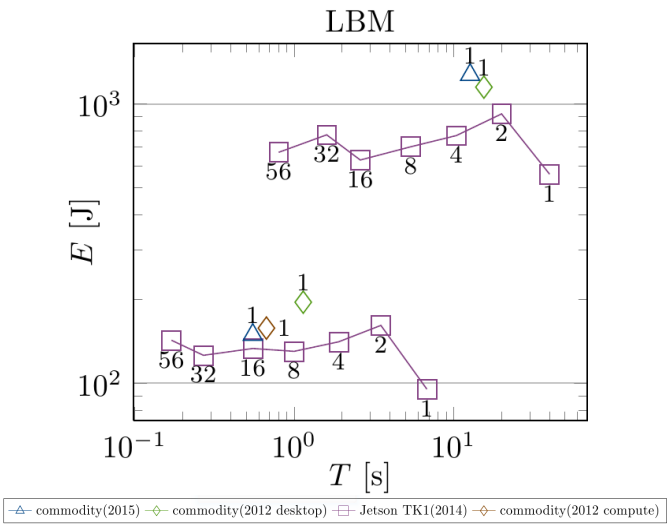


**Figure 5:** *Energy and time to solution of a CFD application on a cluster of Tegra K1 SoCs and several workstations comprising commodity hardware*

**Figure 6:** *Snapshots of the ICARUS cluster at TU Dortmund*

reduce energy by enhancing hardware or software energy efficiency on the scale of our own codes and hardware devices. This does not ease the problem that even with optimal devices the problem of too few energy supplies for future computing cannot be resolved – at least not with 'standard' computers. This kind of thinking led us to a system integration project where we **built a compute-cluster alongside with its power supply by renewable energies**, see Figure 6 for snapshots of the ICARUS project site [Geveler and Turek 2016]. This prototypical data and compute center comprises a great deal of theoretical peak performance (around 20 TFlop/s) provided by 60 Tegra K1 processors that all have a low power GPU on the SoC. By applying a 45 $m^2$ solar farm offering a 8 kWp power source alongside with a Lithium-Ion battery with a capacity of 8 kWh we can maintain operation of the 1 kW peak power dissipation computer even during nighttime for several hours. Even with comparatively cloudy weather and considerably large nightly workloads we can recharge the battery at day times while computing under full load. This computer has all its energy needs fed by renewables even the cooling system and it is not connected to the power grid at all.

The idea of the project is: If we cannot tackle the overall problem of energy demand increasing much faster than its supply why not build any new needed supply directly with the system? In recent work we proved that this is possible [Geveler et al. 2016b]. With this we come back to Figure 1. Note that finally building new renewable energy sources alongside with its demand in the model used by the SIA the supply curve would be parallel to the consumption. Higher energy efficiency is still needed because renewable energy sources such as photovoltaic units impose new constraints like area or initial cost that should be minimised. Building compute centers this way we could reduce the follow-up energy cost of a computational resource to zero.

## 3 Conclusion

Keep in mind that the system integration described in the previous section is only possible due to simple yet very effective performance modeling which allows for choosing hardware and numerics as well as tuning them properly. Therefore **hardware-oriented numerics is the central aspect here**: The approach is successful for a specific type of numerics that can be scaled effectively i.e. numerically and in terms of hardware- and energy-efficiency. Hardware-oriented numerics therefore means:

(1) The extension of the original paradigm by the aspect of **energy-efficiency**. New methods in performance modeling have to be developed and applied and energy to solution has to be internalized into tuning efforts of software on the application level.

(2) The **selection of compute hardware should be based on these models**. In any decision and where necessary **energy efficiency should be favored over raw performance** although we have shown how this can be bypassed by clustering unconventional computer hardware and enhancing scalability properties of a given code.

(3) Furthermore **knowledge concerning energy efficient computing has to be spread in the applied science community**. A knowledge base should be installed that disseminates methods in power and energy measurement as well as profiling and benchmarking techniques in order to bring up sophisticated performance models for the application level, performance engineering for energy-efficiency, energy consumption of compute devices in local compute resources and energy consumption of compute clusters and whole data centers.

(4) After hardware selection many **hardware parameters have to be tuned during operation**. We demonstrated how to determine an optimal number of threads for a maximum of energy efficiency. Another good example here is finding an optimal preset core frequency. Although with Dynamic Frequency Scaling modern processors show very complex behavior for different workloads and especially during runs of complex applications, in many cases one can find an optimal frequency for a certain type of applications with similar sets of measurements like in our example.

(5) Finally, the development of the ICARUS system has been accompanied by a two semesters student project where the participants actively contributed to the system design. Here they learned how to co-develop hard- and software/numerics from scratch for computations being powered by renewables and batteries starting with performance modeling and hardware details up to integrating

everything into a future-proof resource. It is **bringing sensitivity for energy efficiency and consumption into the peoples' minds which starts with being integrated into teaching** which is in the end maybe the most important thing one can do.

#### Acknowledgments

ICARUS hardware is financed by MIWF NRW under the lead of MERCUR. This work has been supported in part by the German Research Foundation (DFG) through the Priority Program 1648 'Software for Exascale Computing' (grant TU 102/48). We thank the participants of student project Modeling and Simulation 2015/16 at TU Dortmund for initial support.

## References

ANZT, H., AND QUINTANA-ORTÍ, E. S. 2014. Improving the energy efficiency of sparse linear system solvers on multicore and manycore systems. *Phil. Trans. R. Soc. A 372*, 2018. doi: 10.1098/rsta.2013.0279.

BENNER, P., EZZATTI, P., QUINTANA-ORT, E., AND REMN, A. 2013. On the impact of optimization on the time-power-energy balance of dense linear algebra factorizations. In *Algorithms and Architectures for Parallel Processing*, R. Aversa, J. Koodziej, J. Zhang, F. Amato, and G. Fortino, Eds., vol. 8286 of *Lecture Notes in Computer Science*. Springer International Publishing, 3–10. doi: 10.1007/978-3-319-03889-6_1.

FENG, W., CAMERON, K., SCOGLAND, T., AND SUBRAUMANIAM, B., 2015. Green500 list, jul. http://www.green500.org/lists/green201506.

GEVELER, M., AND TUREK, S., 2016. ICARUS project homepage. http://www.icarus-green-hpc.org.

GEVELER, M., RIBBROCK, D., GÖDDEKE, D., ZAJAC, P., AND TUREK, S. 2013. Towards a complete FEM–based simulation toolkit on GPUs: Unstructured grid finite element geometric multigrid solvers with strong smoothers based on sparse approximate inverses. *Computers and Fluids 80* (July), 327–332. doi: 10.1016/j.compfluid.2012.01.025.

GEVELER, M., REUTER, B., AIZINGER, V., GÖDDEKE, D., AND TUREK, S. 2016. 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. *Computer Science - Research and Development*, 1-10 (June). doi: 10.1007/s00450-016-0324-5.

GEVELER, M., RIBBROCK, D., DONNER, D., RUELMANN, H., HÖPPKE, C., SCHNEIDER, D., TOMASCHEWSKI, D., AND TUREK, S. 2016. The icarus white paper: A scalable, energy-efficient, solar-powered hpc center based on low power gpus. In *Workshop on Unconventional HPC*, Springer, LNCS, Euro-Par '16. accepted.

HAGER, G., TREIBIG, J., HABICH, J., AND WELLEIN, G. 2014. Exploring performance and power properties of modern multi-core chips via simple machine models. *Concurrency and Computation: Practice and Experience*. doi: 10.1002/cpe.3180.

INTEL CORP, 2015. Desktop 4th Generation Intel Core Processor Family, Desktop Intel Pentium Processor Family, and Desktop Intel Celeron® Processor Family Datasheet Volume 1 of 2. http://www.intel.com/content/www/us/en/processors/core/4th-gen-core-family-desktop-vol-1-datasheet.html.

INTEL CORP, 2015. Desktop 4th Generation Intel Core Processor Family, Desktop Intel Pentium Processor Family, and Desktop Intel Celeron® Processor Family Datasheet Volume 2 of 2. http://www.intel.com/content/www/us/en/processors/core/4th-gen-core-family-desktop-vol-2-datasheet.html.

LAWRENCE BERKELEY NATIONAL LABORATORY, 2006. High-performance buildings for high-tech industries: Data centers. http://hightech.lbl.gov/datacenters.htm.

MALAS, T. M., HAGER, G., LTAIEF, H., AND KEYES, D. E. 2014. Towards energy efficiency and maximum computational intensity for stencil algorithms using wavefront diamond temporal blocking. *CoRR abs/1410.5561*. http://arxiv.org/abs/1410.5561.

MEUER, H., STROHMEIER, E., DONGARRA, J., SIMON, H., AND MEUER, M., 2015. Top500 List, jul. http://top500.org/lists/2015/06/.

NVIDIA CORP, 2014. NVIDIA Jetson TK1 Development Kit - Bringing GPU-accelerated computing to Embedded Systems. http://developer.download.nvidia.com/embedded/jetson/TK1/docs/Jetson_platform_brief_May2014.pdf.

SCHÄPPI, B., PRZYWARA, B., BELLOSA, F., BOGNER, T., WEEREN, S., HARRISON, R., AND ANGLADE, A. 2009. Energy efficient servers in Europe – energy consumption, saving potentials and measures to support market development for energy efficient solutions. Tech. rep., Intelligent Energy Europe Project, June.

SIA, 2015. Rebooting the it revolution: A call to action, sept. http://www.semiconductors.org/clientuploads/Resources/RITR%20WEB%20version%20FINAL.pdf.

TUREK, S., BECKER, C., AND KILIAN, S. 2006. Hardware–oriented numerics and concepts for PDE software. *Future Generation Computer Systems 22*, 1–2, 217–238. doi:10.1016/j.future.2003.09.007.

TUREK, S., GÖDDEKE, D., BECKER, C., BUIJSSEN, S., AND WOBKER, H. 2010. FEAST —- realisation of hardware–oriented numerics for HPC simulations with finite elements. *Concurrency and Computation: Practice and Experience 6* (May), 2247–2265. Special Issue Proceedings of ISC 2008. doi:10.1002/cpe.1584.