2016 IEEE International Parallel and Distributed Processing Symposium Workshops

Energy Claims at Scale: Decreasing the Energy Demand of HPC Workloads at OS Level

Christopher Eibel, Timo Hönig, and Wolfgang Schröder-Preikschat
Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU)
Email: {ceibel,thoenig,wosch}@cs.fau.de

***Abstract*—The ever-increasing performance and power demand of HPC systems requires sophisticated approaches that improve energy-efficient job execution. Targeted goals such as the 20-MW limit for an exascale system, set by the Department of Energy (DoE), are not achievable with advances at hardware level only. Instead, it is inevitable to establish new concepts at system-software level. We propose a system software in which the operating system acquires a key position in the energy-reduction process. Our approach advocates heterogeneous hardware components—especially those that currently emerge from the embedded-system domain—to achieve both high performance and high energy efficiency.**

I. Introduction

The power consumption and peak performance of complex, distributed HPC systems is continuously increasing at an extreme scale. Two generations of the Tianhe supercomputer demonstrate this: Tianhe-1A (2010) consumes 4.04 MW (0.635 gigaflops per watt), just three years later its successor Tianhe-2 (2013) already consumes 17.6 MW (1.935 gigaflops per watt) [12]. The increasing resource demands of such HPC systems ask for novel, yet unexplored ways for effectively allocating system resources which, on the one hand, provide sufficient computational power to the system, and entail electrical power consumption on the other hand. At this, the effectiveness can be determined by how well a system design can handle and serve conflicting, if not mutually exclusive requests during the processing of computational jobs.

Although early measures to decrease the power consumption at hardware level [7] were initially developed for mobile computing systems, the area of power reduction of computing systems is explored in a much wider context in the meantime and also addresses power consumption concerns of HPC systems [17]. Today, the scope of power-saving features exceeds the pure computational part of an overall system and addresses system components of neighboring systems (e.g., maintenance components, cooling systems) and the infrastructure of entire buildings [5]. Even though operating systems are responsible for allocating and freeing resources in order to process pending work most efficiently, up until now their impact on the power-consumption reduction is mainly driven by local decisions, which are often missing a global view of the overall system. We propose a system design that puts the operating system into the center of local *and* global power-saving control-mechanisms, which operate under the consideration of their global impact with regards to diverse system criteria (e.g., response times, result accuracy, and performance). Our proposed system design incorporates several knowledge sources (e.g., programmer, application) to actively support the operating system at allocating resources required to process pending jobs in an energy-efficient manner while retaining predefined quality-of-service attributes (e.g., response times).

The paper is structured as follows. We motivate our work in Section II, discuss background work in Section III, and present our system design in Section IV. Section V discusses the prototypical implementation of our system design as future work, and we conclude our work in Section VI.

II. Motivation

In comparison to desktop computers or HPC systems, saving power and energy has always been one of the key design objectives for embedded systems. For this reason, most power-management technologies first emerged from embedded hardware architectures, but eventually were transferred to larger systems at a later point in time. For larger systems, power capping is a traditional power-management technique that has been broadly explored in the HPC research community [16], [2]. Currently, embedded-system components gain traction within the field of high-performance computing [11] as they offer decent processing power at lower energy costs. For example, Grasso et al. [8] make a first step towards the use of embedded GPUs as a replacement for embedded CPUs that can be used in next-generation HPC systems. Generally, jobs to be executed differentiate from each other in diverse characteristics such as size and varying resource demands. Furthermore, the type of workload can require special-purpose hardware to be executed efficiently. We propose to use a mix of high-performance compute nodes on the one side and highly energy-efficient compute nodes on the other side. A corresponding system architecture provides maximum flexibility for cases with mixed performance requirements but also different requirements to underlying hardware components. Our proposed system 1) analyzes jobs and their sub-jobs for any hardware dependencies and their expected usage of resources (e.g., CPU cores, memory, energy, power) in advance, 2) maps them at runtime to the respective hardware components for maximum (energy-)efficient execution, and 3) continuously keeps track of system processes to dynamically react to any state changes (e.g., new and completed jobs, hardware failures). Ideally, the integration of these three steps has minimal impact; that is, it does not require to create whole new HPC systems and management software on top. Instead,

978-1-5090-3682-0/16 $31.00 © 2016 IEEE
DOI 10.1109/IPDPSW.2016.69

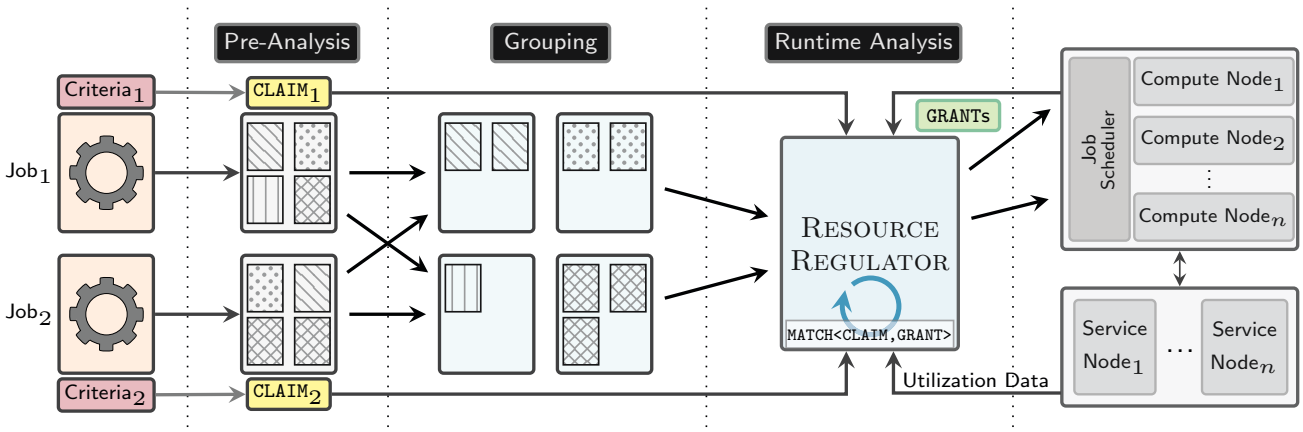

Fig. 1. The proposed system design employs a RESOURCEREGULATOR which implements the runtime–decision-making process. The use of available system resources is optimized based on individual job criteria. This optimization accounts for a reduction of energy consumption at OS level.

our system design is flexible enough to be incorporated into hardware and software components of existing HPC systems. Our approach introduces well-defined interfaces for the system to peek for application characteristics.

With our system design (Section IV), we react on a shift of responsibilities, which is induced by the increasing heterogeneity at hardware level. Power-saving strategies need to weave in local and global aspects in order to pursue the shared objective of optimizing the performance yield while reducing the energy footprint of the overall system. Results from previous research [9] suggest the operating system to be the right level of abstraction [18] for this functionality as it has the necessary information available for provisioning input to existing job schedulers. Such external schedulers take care of global operations whereas the operating system carries out self-governed local optimization actions.

III. BACKGROUND

The reduction of energy consumption without suffering tremendous performance degradation serves as primary goal of our system design. However, there are other dependencies that are in close relation to energy consumption. For example, it can be beneficial not to exceed a power limit because of temperature implications (e.g., modern CPU designs follow an over-provisioning approach by prohibiting the simultaneous clocking at maximum speed of all cores [14]) or limited available power because of power outages or brownouts. Keeping the temperature below a certain value can dramatically decrease cooling costs of the whole data center [13].

Our proposed approach for increasing the energy proportionality of HPC systems is motivated by results of our previous research. First, it is necessary to statically analyze and optimize the program code to be executed. We encourage to use corresponding tooling support [10] which assists developers in the process of analyzing and optimizing the program code. Second, we argue that support at the level of the operating system is inevitable. Here, we build upon previous work using a Linux-based operating-system kernel [9] which reduces the energy demand for executing tasks. This kernel provides energy measurements at runtime which are controlled by system calls and it implements fine-grained energy optimizations by exploiting an extended set of power-saving features (e.g., DVFS, C-states, and sleep states). Conceptually, our presented approach is operating-system agnostic and, hence, can be applied to systems such as IBM's BLRTS [1]. We discuss integration aspects of our approach in Section V. Third, in the context of distributed, replicated systems, we have shown that the energy proportionality of a system can also be improved with dynamic software decisions at runtime [6]. Thus, although the software exploits many hardware mechanisms to achieve that (e.g., using power caps), it is not the hardware manufacturers' responsibility alone to improve energy proportionality with hardware innovations.

IV. SYSTEM DESIGN

In this section, we present the substantials of our system design. Figure 1 gives an overview of all phases (pre-analysis, grouping, runtime analysis) that bundled tasks (=jobs) to be processed run through. In general, there is no limitation of jobs that can be submitted and processed simultaneously; however, jobs are likely to be retained in favor of, for example, processing them as energy efficiently as possible.

At first, jobs are being analyzed to derive *resource claims* (see § IV-A and § IV-B). Subsequently, assuming that most jobs are moldable [4], the pre-analyzed jobs are partitioned and grouped into bundles that can be efficiently executed together. After this static analysis process, a dynamic analysis process follows, which continuously decides in a feedback loop when and where sub-jobs or whole jobs are actually being executed. This decision is based on information and feedback, so-called *resource grants* (see § IV-C), retrieved from the nodes in the cluster. Our system design builds around a main component, the RESOURCEREGULATOR, which is responsible for the runtime–decision-making process and forwarding (sub-)jobs to the job scheduler.

A. Resource Claims

To increase the efficiency of the job processing, it is necessary to map the resources required by each job to the resources available on all potentially highly diverse compute nodes in the cluster. Typical resources are CPU specifics, such as number of cores and available CPU features (e.g., SSE, TSX, VT-x), or overall disk and memory space. Moreover, the specifics of peripherals (e.g., maximum network bandwidth between certain machines, disk speed) need to be incorporated into the pool of available resources. Machines differentiate from each other by the availability of special-purpose hardware components, such as hardware accelerators, making them especially useful or even mandatory for certain kinds of jobs.

Before starting a job, it is necessary to claim the required resources ($\texttt{RES}_1, ..., \texttt{RES}_n$) for its execution by sending a resource-claim message $\texttt{CLAIM<RES}_1, \ldots, \texttt{RES}_n\texttt{>}$ to the regulator. These claims consist of certain criteria that have to be defined individually for each job. The higher the weight of a certain criteria, the higher its importance and the higher its consideration in the regulator's decision process that follows.

Figure 2 shows a possible criteria configuration (green, solid line). Values that are closer to the outside margin imply that the respective criterion is more important. For example, a higher value for the power criterion means that the power value should not exceed a certain value. On the contrary, a higher value for the performance criterion implies that the job has to be processed as fast as possible—potentially within a predefined period of time. Therefore, it is not enough to define the level of importance for each criteria but to describe the criteria with concrete values for each limit (e.g., absolute precision). We further differentiate between *strict* and *loose* criteria, where strict criteria must be fulfilled and loose criteria may be ignored within specific limits in order to fulfill all strict criteria. In Figure 2, strict criteria are marked with a circle. In the shown example, accuracy and costs do not have the highest importance values but they are strict criteria, meaning that the regulator's solution for this optimization problem must fulfill the respective criteria's concrete values. A possible configuration found by the regulator is exemplified by the dotted line: In this case, in order to fulfill the strict criteria for accuracy and costs, it is necessary to degrade all other criteria. The red, dashed line indicates a criteria configuration that is possible when the user is only interested in an energy-optimal solution and has no further requirements to the other criteria. In this instance, the other criteria are lowered in order to achieve the highest amount of energy savings.

B. Application-Induced Claims

To establish an initial set of resource claims, we encourage a pre-analysis that autonomously retrieves requirements from existing information such as user-supplied code annotations or the program source code. The resource claims contain all relevant resources, such as the number of required compute cores, but also the type of hardware that is necessary to efficiently execute program code (e.g., CUDA-enabled GPUs, cryptographic accelerators, special-purpose arithmetic units).

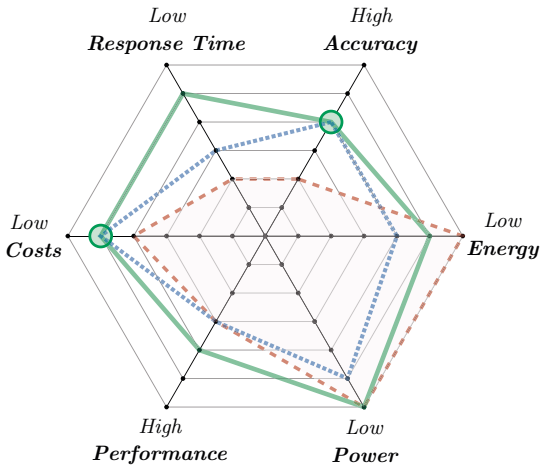

Fig. 2. Spiderweb showing examples for weighted strict and loose criteria.

Resource claims which express energy demands need to be treated differently, as energy requirements can not directly be retrieved from hardware specifications. To determine the energy consumption of individual hardware components, we use platform-dependent energy profiles. That is, energy profiles are the basis for estimating a program's or job's energy consumption. This information is supplemented during the job processing with runtime statistics, for example, performance-counter data. Other useful periodically monitored statistics are CPU utilization and memory usage, network bandwidth, and execution durations. The system runs a continuous feedback loop, taking into account all available data points in each decision to improve the matching of pending resource claims. This monitoring mechanism can be coupled with existing system components, for example, service nodes. To extract runtime information also from compute nodes, according API extensions must be established if they are unavailable yet.

C. Resource Grants

The regulator, depicted in Figure 1, is continuously invoking a `MATCH<CLAIM,GRANT>` function. We speak of (partial) *resource grants* when a job's resource offer is (partially) accepted by the compute nodes' job scheduler. If a resource offer can not be fully granted, resource claims as requested may be violated with regard to one or more criteria. At the same time this may be of benefit for another resource which is part of the resource claim. It is then the regulator's responsibility to realign the resource offers and find an optimal set of matches. When a resource claim is granted, the regulator notifies and informs the compute nodes' job scheduler about available resources via an API call. Although it is the scheduler's responsibility to use these available resources as efficiently as possible and meet the resource grants' parameters, the regulator can additionally propose suggestions on static and dynamic power management. We see a great potential in exploiting power-saving features emerging from the domain of embedded systems, as they gradually become available in the domain of larger systems (i.e., HPC), too.

V. Integration and Implementation

Currently, we evaluate the exact hardware and software configuration to implement our presented system design. For the system hardware we consider a heterogeneous combination of powerful, yet energy-efficient ARM processors in combination with nodes which provide high compute performance. We consider the use of many-core systems (e.g., Intel Xeon Phi) as part of our cluster of compute nodes. We will build a system that comprises a pool of nodes which are grouped in classes (i.e., compute, management, and service nodes). The nodes of the individual classes will operate with a highly customized version of the Linux operating-system kernel. In previous work [9], we successfully deployed such a kernel in order to reduce the energy demand of the system. The low-level kernel code of our operating-system kernels will make aggressive use of both, power-saving and power-capping features which are available on the distinct hardware platforms. In addition to our node-specific operating-system kernels we will evaluate a ResourceRegulator implementation aligned to our system design (Section IV). We will explore how existing HPC-cluster software [3] benefits from supplemental system software as proposed in this paper. For multi-criteria optimization in the course of resource management, we consider to build on recent approaches such as [19].

We will further investigate the benefits of our approach for established supercomputer systems such as IBM Blue Gene [1] or Tianhe-2 [12]. We see a great potential in heterogeneous HPC systems that contain a large number of energy-efficient embedded processors. Rajovic et al. have shown [15] that such systems can already keep up with existing HPC systems. As novel functional features (e.g., improved floating-point units) currently are implemented for embedded processors, too, they will become also available to HPC applications. Thus, we expect that the performance gap between a traditional and a hybrid HPC system will gradually decrease whereas the energy efficiency greatly improves.

VI. Conclusion

To address current changes and upcoming challenges, we proposed a system design for energy-efficient HPC systems. Our approach extends the scope of power-management functionalities at operating-system level and exploits heterogeneity at hardware level. The adaptation of powerful yet energy-efficient embedded-computing components is eased by the proposed system design. Only by exploiting operating-system support it is possible to align the requirements and characteristics of HPC workloads with available hardware resources to efficiently decrease the system's overall energy demand.

Acknowledgments

This work was supported in part by the German Research Foundation (DFG) under grants no. SCHR 603/11-2, SCHR 603/13-1, and the Transregional Collaborative Research Centre "Invasive Computing" (SFB/TR89, Project C1).

References

[1] G. Almási, R. Bellofatto, J. R. Brunheroto, C. Cascaval, J. G. Castaños, L. Ceze, P. Crumley, C. C. Erway, J. Gagliano, D. Lieber, X. Martorell, J. E. Moreira, A. Sanomiya, and K. Strauss. An overview of the Blue Gene/L system software organization. In *Proceedings of the 9th International Conference on Parallel and Distributed Computing (Euro-Par '03)*, pages 543–555, 2003.

[2] P. Bailey, D. Lowenthal, V. Ravi, B. Rountree, M. Schulz, and B. de Supinski. Adaptive configuration selection for power-constrained heterogeneous systems. In *Proceedings of the 2014 International Conference on Parallel Processing (ICPP '14)*, pages 371–380, 2014.

[3] E. Caron and F. Desprez. DIET: A scalable toolbox to build network enabled servers on the grid. *International Journal of High Performance Computing Applications*, 20(3):335–352, 2006.

[4] W. Cirne and F. Berman. A model for moldable supercomputer jobs. In *Proceedings of the 15th IEEE International Parallel & Distributed Processing Symposium (IPDPS '01)*, pages 59–67, 2001.

[5] S. Dawson-Haggerty, A. Krioukov, J. Taneja, S. Karandikar, G. Fierro, N. Kitaev, and D. Culler. BOSS: Building operating system services. In *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI '13)*, pages 443–457, 2013.

[6] C. Eibel and T. Distler. Towards energy-proportional state-machine replication. In *Proceedings of the 14th ACM Workshop on Adaptive and Reflective Middleware (ARM '15)*, pages 19–24, 2015.

[7] S. Gary, P. Ippolito, G. Gerosa, C. Dietz, J. Eno, and H. Sanchez. PowerPC 603, a microprocessor for portable computers. *Design & Test of Computers*, 11(4):14–23, 1994.

[8] I. Grasso, P. Radojkovic, N. Rajovic, I. Gelado, and A. Ramirez. Energy efficient HPC on embedded SoCs: Optimization techniques for Mali GPU. In *Proceedings of the 28th IEEE International Parallel & Distributed Processing Symposium (IPDPS '14)*, pages 123–132, 2014.

[9] T. Hönig, C. Eibel, B. Herzog, H. Janker, P. Wägemann, and W. Schröder-Preikschat. Playing hare and tortoise: The FigarOS kernel for fine-grained system-level energy optimizations. In *Proceedings of the 2015 Brazilian Symposium on Computing Systems Engineering (SBESC '15)*, 2015.

[10] T. Hönig, H. Janker, O. Mihelic, C. Eibel, R. Kapitza, and W. Schröder-Preikschat. Proactive energy-aware programming with PEEK. In *Proceedings of the 2014 USENIX Conference on Timely Results in Operating Systems (TRIOS '14)*, pages 1–14, 2014.

[11] D. Jensen and A. F. Rodrigues. Embedded systems and exascale computing. *IEEE Computing in Science & Engineering*, 12(6):20–29, Nov. 2010.

[12] X. Liao, L. Xiao, C. Yang, and Y. Lu. MilkyWay-2 supercomputer: System and application. *Frontiers of Computer Science*, 8(3):345–356, June 2014.

[13] E. Pakbaznia and M. Pedram. Minimizing data center cooling and server power costs. In *Proceedings of the 2009 ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED '09)*, pages 145–150, 2009.

[14] T. Patki, D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. de Supinski. Exploring hardware overprovisioning in power-constrained, high performance computing. In *Proceedings of the 27th ACM International Conference on Supercomputing (ICS '13)*, pages 173–182, 2013.

[15] N. Rajovic, A. Rico, N. Puzovic, C. Adeniyi-Jones, and A. Ramirez. Tibidabo: Making the case for an ARM-based HPC system. *Future Generation Computer Systems*, 36:322–334, July 2014.

[16] B. Rountree, D. H. Ahn, B. R. de Supinski, D. K. Lowenthal, and M. Schulz. Beyond DVFS: A first look at performance under a hardware-enforced power bound. In *Proceedings of the 26th IEEE International Parallel & Distributed Processing Symposium Workshops (IPDPSW '12)*, pages 947–953, 2012.

[17] B. Rountree, D. K. Lownenthal, B. R. de Supinski, M. Schulz, V. W. Freeh, and T. Bletsch. Adagio: Making DVS practical for complex HPC applications. In *Proceedings of the 23rd ACM International Conference on Supercomputing (ICS '09)*, pages 460–469, 2009.

[18] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277–288, Nov. 1984.

[19] M. Shafique and J. Henkel. Agent-based distributed power management for kilo-core processors. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '13)*, pages 153–160, 2013.