Energy Aware Software

Principal investigator: Dr. Simon Holmbacka
Project starting time: xxx
Duration of the project: 48 months
Site of research:
Åbo Akademi University
Faculty of Science and Engineering
Embedded Systems Laboratory
Vattenborgsvägen 5 20500 Turku, Finland

Sites for research cooperation:

IETR - INSA Rennes
Institut d'Electroniques et de Télécommunications
35708 Rennes Cedex 7, France

TU Wien
Institute for Software Technology and Interactive Systems
1040 Favoritenstrasse 9-11 Vienna, Austria

FernUniversität in Hagen
Fakultät für Mathematik und Informatik
58084, Universitätstrasse 11, Hagen, Germany

Uppsala University
Department of Information Technology
75105, ITC, Lägerhyddsvägen 2, Uppsala, Sweden

1 Rationale

Energy efficiency in computer systems is currently a major concern in field computer engineering for several reasons such as increased frequency of the battery re-charge intervals, monetary cost because of high electrical consumption and thermal effects leading to increased cooling requirements with high noise levels. Furthermore, energy efficiency is also a concern for achieving increased performance. One of the performance limits in current microprocessors because the Dennard scaling [DGR$^+$74] is no longer valid. Dennard scaling forecasts that the power density of transistors stays constant as the manufacturing technology decreases. This means that as transistors become smaller, energy efficiency is improved by transistor technology such that the power remains in proportion to the area. In recent hardware, this scaling no longer applies since the voltage range used in transistors no longer can be lowered. The end of Dennard scaling brings consequences such as an disproportional increase in thermal dissipation leading to dark silicon, which means that all processing power on a chip cannot be used simultaneously because of the limited power envelope.

To tackle these problems, power management systems are implemented to scale the performance of the system according to the current resource demand. The resource demand is measured as the utilization caused by the workload of the CPU. Workload in operating systems is measured as a sliding window average over an active and idle CPU as illustrated in Figure 1. A high workload indicates that the system needs more resources, and the clock frequency is scaled up to decrease the workload on the CPU. Setting the frequency can be based on different policies, and the most common policy is called "ondemand"[Sta06]. This frequency governor scales the clock frequency to the maximum value as soon as a workload threshold is reached. After this it step-wise decreases the clock frequency to the lowest frequency not exceeding the threshold.

The problem with this approach is that the workload does not describe the performance of the applications. High workload does not necessary mean that an application requires more performance, it simply describes how much the application is using the CPU. It is merely an indirect effect of executing an application as illustrated in Figure 1. For example, in case the CPU is set to decode a video, the workload will increase to 100% as soon as the video frames are being decoded. This means that the CPU is clocked to the maximum frequency as long as the frames are being decoded and the decoded frame rate is only limited by the speed of the CPU; even if the required decoding rate is only 25 frames per second, which the human eye is capable of observing. In other

words, the application will be executed unnecessarily fast. As results from [HLL15, HNP$^+$14] show, executing the application unnecessarily fast wastes significantly more energy than executing the application on a moderate, but still at a sufficiently fast, performance level. An attempt to refine the functionality of the frequency governors was presented in [SKK11]. Their frequency governor the "Green Governor" scales the clock frequency not only based on the workload, but also in the memory intensity. The assumption made was that higher memory intensity stalls the CPU enough to not benefit from high clock frequency. The fundamental problem was still unsolved because the governor attempted to model the performance requirement of the application indirectly, and in case the model does not fit the application, either excessive or insufficient resources are allocated to the application.

What we propose is to make the application software itself **energy aware**. This means that the applications are involved in resource allocation directly. An energy aware application can set performance requirement in the own metric, e.g. frames per second. The runtime system monitoring the performance of the application then allocates the hardware resources to meet the demands of the applications. Such a power management system requires an interface between application and runtime system, which is current non-existing. This research is aimed to provide such an interface, and the necessary tools needed to use it. Three main parts are planned for this research:
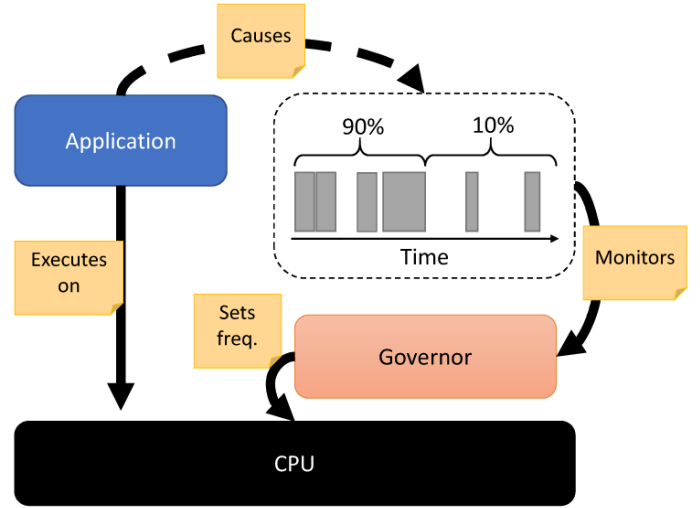

Figure 1: Current approach for power management: The application causes a workload (illustrated by the gray rectangles), and the frequency governor scales the clock frequency according to the load level.

a. A framework for expressing performance requirements in the applications

b. A system monitoring the runtime performance of the applications

c. A controller system which allocates resources based on information from the monitor and the performance requirements

The three parts will be covered in this research project and described as follows:

1.1 Application meta-data

The main challenge in creating energy aware software is to enable the expression of resource requirements in applications. As previously mentioned, the traditional way of solely monitoring the workload is a very inaccurate and often counter-productive both in terms of performance and energy. We tackle this challenge by allowing the applications to express the resource requirements internally. This means that the application should include a small part of code – the meta-data – which expresses the intension of the application. For example, the previously mentioned video decoder can demand a framerate of 25 frames per seconds instead of decoding as *fast as possible*. In this research, we will investigate the types on meta-data needed to describe and application sufficiently in terms of performance requirements. On of the challenges is to obtain a sufficiently descriptive form of meta-data that can be generalized to any new- or legacy application. Once the meta-data descriptions have been established, the injection of the data into both new- and legacy applications must be achieved fairly effortless in order for the programmers to make the additional effort. A trade-off between descriptiveness and programmer effort is to be expected.

1.2 Feedback monitor

Since applications in general are dynamic, the performance of the application depends on a wide set of factors such as memory intensity, use of caches, user interrupts, the data sizes and data types used, interaction with other applications etc. This means that the performance of an application cannot be guaranteed with fixed resources. For example a completely black video frame requires much less resources for decoding than a video frame with many

details and motion vectors. In order to adapt the resources allocated to the application, the performance of the application must be monitored at runtime. Such a monitoring facility can determine the *current* performance of the application to the *required* performance annotated in the meta-data. In case the currently allocated performance is either too high or too low, the monitoring reports this issue to the controller which allocates or deallocates resources to accommodate to the real resource requirement. A main part of this research is therefore to create a proper monitoring framework. Such a framework must be detailed enough and provide sufficient tools for measuring performance, but simple enough to reduce programmer effort for using it. Since the monitoring framework is a runtime system, significantly low overhead must be guaranteed to not interfere with the energy consumption and performance of the host system.

1.3 Resource controller

The third part of the system is the resource controller which allocates and deallocates hardware resources. Such resources are usually in form of clock frequencies and numbers of cores active, but is not necessarily restricted to these resources. Allocating resources efficiently means having a knowledge of how resources of one form affects the performance of the application requesting the resources. This means that a complete system model is required together with rigorous optimization methods implemented with a very low performance overhead on the host system. A prototype controller was developed by the PI in his thesis, and this research project will build upon this idea of multi-criteria optimization. Further efforts will also be put on interdisciplinary platforms with different possible hardware resources, allocation methods and performance requirements.

1.4 Previous work

The embedded systems lab in the Åbo Akademi University has for the last six years focused on low power and energy aware software. During these years a special interest had been put in mobile phone processors and their significance for energy awareness in general. This is because it has been noted that the trend of performance requirements by far exceeds the battery capacity of a mobile device, especially since the mobile multi-core revolution [Epe15, Tri15]. This means that the performance of the hardware and the demand for performance of the user is greater than the energy a battery with limited dimensions can physically store. Because of the slow capacity increase in batteries, the available energy must be used more efficiently in order for such a mobile device to retain its usability. In the thesis of the PI "*Energy Aware Software for Many-Core Systems*" two guidelines were presented for creating energy aware software on modern many-core hardware. Implementing the recommendations in software has proven to reduce the energy consumption up to 50% without degrading the performance [HLL15], especially on mobile multi-core hardware. The recommendations called "Energy aware mapping" and "Energy aware resource allocation" are used to tailor the resource allocation to the software executing, and a prototype runtime system was implemented as a phase of the PhD thesis. Using this software, practical state-of-the-art demonstrators were created, for example a low power video transcoding cloud system[1] demonstrated in the Millennium Pavilion SHOK Summit in Helsinki, and in the DIGILE Workshop "Rolling up the Sleeves". Also, the Android app "Low Energy Player" was implemented as a state-of-the-art demonstrator and is available on Google store[2] This research is intended to go beyond the state-of-the-art by the development of a programming framework for energy aware software and practical demonstrators.

1.5 Related work

"Low energy programming" or "Low power programming" has previously existed in the form focusing on the programming paradigm or on the programming syntax. Guidelines from Intel [BSA11] suggests the use of a certain level of loop unrolling, vectorization, memory intensity cache usage etc. for achieving maximum energy efficiency in combination with Intel compiler tools. Such recommendations are applied only on the algorithms in the program, and do not cover, the intention of the program for functioning efficiently together with the runtime system allocating the resources. The low power programming guidelines from Intel also require the programming to construct the program in a certain way in order to become energy efficient, and the underlying hardware architecture must be known. In the thesis "Developing Energy-Aware Software" by Brinke [tB15] the author describes programming

---

[1]Picture of the ESLab demonstrator available at https://dl.dropboxusercontent.com/u/5260559/ClusterDemo.jpg

[2]https://play.google.com/store/apps/details?id=org.videolan.vlc.LEL.lite.green

languages for modularity and modeling resource consumption for software. The *awareness* of energy, is tightly bound to the application code and the programmer is expected to follow certain programming patterns to make the software energy aware. On the contrary, the planned framework for energy efficient software requires only the insertion of meta-data in the software. This means minimal effort of the programmer and the application algorithms can be implemented without interference from the energy awareness framework. The resource requirements are specified using a simple library, where after the runtime environment allocates the required resources.

The **Carbon Research Group**[3] at MIT has developed a heartbeat framework to evaluate performance as a generic parameter in software construction. The framework is capable of measuring performance of any application as a generic parameter by user inserted API calls to the heartbeat library. Measuring performance is the necessary first step when constructing a feedback-based control system. For example it enables the possibility to measure the framerate in a video decoder, but a controller is then needed to allocate the resources in order to keep the performance on a given setpoint. We consider using the heartbeat framework for measuring generic performance in our energy awareness framework, but we plan to extend the framework considerably in order to add the controller for allocating resources.

The **Hardkernel** project[4] creating the Odroid family boards recently released the Global Task Scheduling (GTS) support for the ARM big.LITTLE devices. "High performance threads" are scheduled to the big high performance cores and "Low performance threads" are scheduled to LITTLE energy efficient cores based on the workload activity of the threads in order to save energy. Even though the activity level of a thread is an early attempt introduce energy awareness in the system, the practical results are poor. In other words, the scheduler most often schedule a thread on *the wrong core*. This results not only in poor energy efficiency, but also in poor performance of the applications and poor user experience. We intend to use this platform as one of the reference models for our energy awareness framework because its SoC is very popular and is being used in millions of Android devices worldwide.

Our research group is currently a partner in the **INTERSYS** project[5] dedicated to standardize and optimize interconnected IoT devices handling streaming data. Since the number of IoT devices are expected to rapidly increase in the near future, the project is extending interoperability notions for handling the massive amount of data streams from small devices to gateways and servers. Still missing in the field of IoT is the notion of energy awareness, which is a crucial point as most devices operate on battery-only power and are expected to operate for long time intervals. We intend to work closely with this project and we plan to introduce the notion of energy awareness in IoT systems.

EMBECOSM[6] focus on providing the GCC compiler with the notion of energy efficiency, in practice this means learning which compiler flags that minimizes the energy consumption for a selected architecture. The outcome of this project is similar to OpenTuner [7] from MIT, which is capable of offline optimization of multi-criteria problems. Both projects provide a metric for offline optimization but runtime support, which we suggest, is not stated in their scope. Runtime optimization is critical in virtually any environment containing multi-node and heterogeneous multi-node platforms. This is because the data used in especially streaming applications like multi-media software is arbitrary or very difficult to predict. Compile time optimizations can for example not predict what kind of video format is being used in a video decoder.

The **StarPU**[8] project at INRIA Bordeaux has presented a runtime system to minimize the performance for heterogeneous architectures. The system builds a performance model of the implemented CUDA or OpenCL kernels based on benchmarking on CPUs and GPUs, after which the system is able to schedule the kernels onto the most performance efficient device. The **PEPPHER**[9] project has used StarPU as a backend and the outcome of the project is a tool capable of generating multi-variant tasks for StarPU (OpenMP, OpenCL etc.). However, StarPU only consider the optimizations in form of performance. When adding more complex criteria with multiple variables such as energy efficiency or monetary cost, StarPU lacks the insight to such resource allocation.

---

[3]http://groups.csail.mit.edu/carbon/

[4]http://www.hardkernel.com/main/main.php

[5]http://iot4health.utu.fi/?p=374

[6]http://www.embecosm.com/

[7]http://opentuner.org/

[8]http://starpu.gforge.inria.fr/

[9]http://www.peppher.eu/

2 Objectives and expected results

2.1 Objectives

Our current design recommendations for energy aware programming described in the thesis of the PI extends the application to signal resource requirements to the runtime system which allocates the hardware resources. **The objectives of this research is to develop a framework for programming energy aware software.** By using the framework, the programmer can specify resource requirement or "Quality-of-Service" requirements directly in the source code. Furthermore, the end-user of the software might- or might not be able to influence these requirements at runtime. For example, a video decoder might occasionally experience buffer underruns in case the decoding framerate is too low. By safe guarding, by using a slightly higher setpoint, the video playback becomes more smooth at the cost of slightly higher energy consumption. Figure 2 illustrates for example the ability to add a Quality-of-Service option to the Netflix app running on Android. The user can use this option to select the trade-off between energy savings and performance in terms of decoding framerate or other parameters. Options can set based on a defined value like "decoding framerate" or as a more abstract option as illustrated in Figure 3. The intension of the energy aware framework is to expose the link between the application and the runtime system, while the design of using its abilities is up to the programmer.
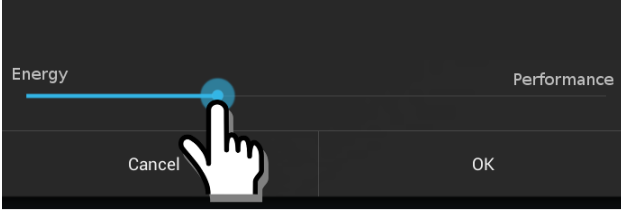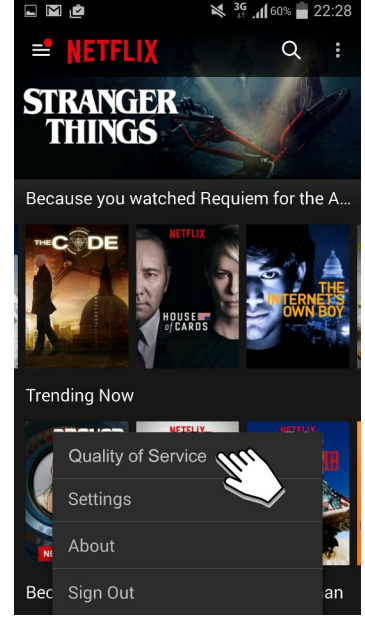

Figure 2: Quality-of-Service as a left click option in the Netflix app on Android

Figure 3: Slider option for QoS in Android app

2.2 Hypothesis

The project states the following Hypotheses:

a. Performance requirements in computer systems will continue to increase significantly faster than battery technology development, software must thus utilize the resources more efficiently.

b. Energy aware software can utilize the hardware resources more efficiently because hardware resources are allocated based on actual performance requirements.

c. Software can be made energy aware with minimal performance overhead and minimal programmer efforts.

d. 50% of energy can be saved using energy awareness in software.

2.2.1 Previous work on energy aware software

In previous work, Eyerman et al. [EEKS09] claim that no single throughput metric is fundamentally generic for multiprogram workloads. Performance should instead be expressed as related to the internal single case-study; a direction adopted in this research. We plan to integrate this direction of thinking into user defined meta-data that expresses resource requirements in software.

In early research, a high-level language CQML [Aag01] was suggested for describing QoS requirements integrated in UML. CQML links a high level QoS description to system performance, and can describe system actions based on

the result. Applications specify a performance setpoint and a lower bound acceptable performance level in context of the application. Applications then monitor own performance and signal this value to the QoS manager periodically. Similar notations as this the language will be considered in this research to describe QoS in applications, but more focus will put on the link between applications and hardware resources in a single computer system. Our methods for energy aware programming will not be strongly tied to a certain programming language and the framework itself will have the flexibility to be integrated from various programming environments such as servers and PC Linux and Android.

On the other hand, runtime systems for minimizing energy consumption in computer systems have been previously proposed. The PowerDial [HSC$^+$11] approach allows graceful degradation in applications based on current application performance measured in heartbeats [HES$^+$10]. The system transforms application parameters (such as peak-signal-to-noise in a video) into dynamic control variables stored in the application itself. A callback function is inserted into the application using which the controller is able to adjust the control variables according to performance and policies. A heartbeat feedback monitors the execution and reports on the updated performance of the application. Also, the work by Segovia [Seg11] suggests graceful degradation of the application QoS by monitoring a happiness value from the application. Based on this value, the runtime system can degrade quality points in the application in order to achieve the requested QoS. Our planned runtime system is inspired by the same approach to treat input signals from applications: the performance is transformed into a generic parameter QoS upon which the controller acts. In contrast, our controller uses no graceful degradation in the applications, but the actual hardware actuators to allocate resources.

In previous research, there have been a strong separation between monitor and control. Several research projects offer the opportunity to monitor an executing application, but supports no control of the hardware. On the other hand, many controller-based research project do not support any proper framework for declaring meta-data requirements and monitoring of the execution. This research project will tie both parts together with the main focus on reducing the energy consumption with minimal programmer effort – an effort not previously done. Our research project will also make the proper balance between academic research and practical usability, which means that there is both a focus on planning the long term usage of the framework in terms of capabilities and scalability, but also practical efforts to enable a programmer to pick up the tools and start developing energy aware software in the industry.

3 Expected scientific and societal impacts and potential breakthrough of the research

Our expected impact is on the research community is to address the energy awareness in software. In other words, the need for communication between the application layer and the runtime environment. We expect to determine the information flow needed to create energy aware software especially for heterogeneous architectures. Our potential breakthrough is to introduce **energy awareness as natural part of programming**.

For decades it has been a natural step to introduce meta-data in the software for creating parallel programs. The programmer has been willing to add #pragmas in OpenMP, Keywords in Cilk or Initializations in OpenCL to create parallel software because of the minimal programming effort and significant performance gain. We intend to extend this notion to energy awareness, and demonstrate the potential reward in terms of energy efficiency. Furthermore, the development of runtime systems becomes more straight forward. Without energy aware programming as the underlying notion, runtime systems in any domain have no common ground on which the decision making is based. Optimizations remain based on ad-hoc ideas and "hacked" hard-code which is usually not portable between either domains or even between different architectures. Even though the implementation of the runtime systems between domains can be different, the core idea of resource allocation decisions based on energy aware programming remains common. With this common denominator, runtime systems engineers between projects and domains can incorporate shared ideas for implementing new, or improving existing runtime systems. For example the GTS scheduler appearing in most high-end Android phones and tablets is currently highly inefficient due to a poor decision making model. Moreover, its implementation model is completely isolated from any other runtime system leaving it highly unportable. By introducing the notion of energy aware programming, the development of runtime systems needed in any modern computer system has the potential to shift from an ad-hoc single-purpose environment to a sharing environment where engineers have a better platform to cooperate on.

3.1 Applicability

- Energy efficient programming: Today most guidelines for energy efficient programming is driven by creating code with efficient algorithms such as using loop unrolling, SIMD vectors, compiler options etc., which generates software specifically for a given platform. By instead relying on application meta-data describing software requirements, similarly as Cilk and OpenMP handle forking and joining task, resource allocation is based on what the software actually needs instead of blindly following the workload of the system.

- Performance portability: one of the major obstacles within the embedded industry is the high portability costs of software which is due to the requirements for high customization of software to particular embedded architectures. With the suggested energy awareness framework, the mapping and scheduling decisions are shifted from the developer to the runtime system. This allows a more architecture generic programming paradigm while still keeping the performance of dedicated code.

- Development of runtime systems: Using the energy awareness framework, the development time of runtime systems is not only decreased but also more standardized. Even potential for automatic runtime system generation emerges as a result of standardizing the foundation.

- Fog computing is currently being proposed as an implementation framework for the IoT. In the fog heterogeneous computing nodes are scattered in the network to enable computations neared to the source to guarantee real-time properties. IoT systems are one the most energy-prone fields of computer engineering because many of such systems are designed to run for long intervals without being connected to the power grid. The framework for energy aware programming is extendable to IoT system as well because the paradigm only requires the ability of inserting resources requirement meta-data into the application software.

3.2 Critical points for success

Our intension is not only to provide theoretical insights into energy aware software, but also the framework needed to enable the programming. Access to hardware platforms is therefore crucial for this project. As the framework for energy aware programming is applicable in an interdisciplinary domain, we intend to apply for accessibility to large scale cloud platforms like Amazon and to Android based platforms. However, our team has previously obtained the Amazon EC2 grant[10] twice enabling access to large scale servers. The PI also has connections to the energy efficient middleware group in ARM Cambridge, the leader of which was invited to one of our project workshops in the PARALLAX project.

Alternative approaches

3.3 Publication plan

We intend to publish at the top journals and conferences based on the JuFo listings, and we will select publication venues that use some form of open access model, most likely green open access. To this end a lump yearly sum is included to cover the publication fees. Further, we plan to also be visible in industrial event and fares to both demonstrate and get feedback on our research work. The publication rate with regard to the project is approximately 3–4 high-quality peer-reviewed publications per year.

4 Research methods and material, support from research environment

We are working extensively within the *streaming applications* paradigm for a number of reasons. Firstly, streaming systems are usually implemented in environments in which energy efficiency is of essence like video playback systems, web servers, digital filtering systems, telecommunications systems and other multi-media or IoT systems. Secondly, since the content of the streaming data is to a large extent arbitrary, no compiler based optimization (or other static solution) can solve the energy problem. The software itself must therefore be energy aware, and backed up by a runtime systems making online decisions. Thirdly, streaming systems are extremely common and is thus providing a large market to work on from tiny IoT systems up to large cloud server systems.

---

[10]https://aws.amazon.com/grants/

The work is highly experimental based, and to validate our approach we need to develop characteristic benchmarks. During the last years our group has gained considerable experience in developing our on benchmarks, as well as our own measurement setups. Results have been summarized in our extensive technical report [HHL+14].

4.1 Management of research material and data

The project will create two kinds of concrete results: 1. Software, and 2. Measurement data. We subscribe to the idea of open reproducible science. The computer architecture area has suffered from problems with reproducibility in that often neither the software nor the full measurement data are available. We intend to be as open as possible about our research. All software will be released under an open-source license. Zenodo[11] has been selected as the platform where we plan to publish measurement results and each submission in Zenodo can get a citable DOI.

4.2 Support form research environment

The research team will be well supported by infrastructure available at through the Centre for Computer Science (TUCS), which is a joint research institute of University of Turku and Åbo Akademi University. TUCS boasts a long history of high-level achievements of its affiliated researchers, in terms of articles in high-level journals and conferences, high number of citations, invitations to speak in the most important conferences in the field, and memberships in editorial boards of many high-level international journals. TUCS has been a Center of Excellence of Research of the Academy of Finland in the very first round of such centers in Finland, 1995-1999. A unit of TUCS, the Centre for Reliable Software Technology (CREST), has also been a Center of Excellence during 2002-2007.

The PI is an active member of the COST action IC1305 Network for Sustainable Ultrascale Computing and the Energy Efficient High Performance Computing Working Group (EE HPC Working Group). Joint articles have been published in the COST action between the PI in Åbo Akademi University and the group in TU Wien led by Ivona Brandić and one between the PI and the group in University of Tirana led by Neki Frasheri, once of which to appear in IEEE Transactions.

The ESLab is leading the energy efficient computing module of the EIT ICT Labs Master School in Embedded Systems and is actively participating in the activities of the thematic action line Smart Energy Systems of EIT ICT Labs. The PI together with EIT Digital recently released an online Coursera course on the subject in "Development of Real-Time Systems"[12] currently followed by over 2500 students world-wide.

Our team has a full time lab technician constructing specialized equipment needed in our experimental work. The team therefore has strong knowledge in manufacturing measurement tools for externally probing and measuring running systems. Crucial to our experiments is our open hardware datalogger with full Linux tool support capable of high-sample power measurement measurements with 0% performance overhead in the host system.

The team is experienced in building tools based on the theoretical models that enable the use of the models in practice. Some of the previous programming tools created by Åbo Akademi University is the Canals data-flow language and the RVC-CAL to OpenCL translator. This competence will be put to good use in creating the energy aware programming framework.

4.3 Utilization of research infrastructure

Points here

5 Ethical issues

This research project has no ethical issues.

---

[11]http://zenodo.org/
[12]https://www.coursera.org/learn/real-time-systems/

6 Implementation: schedule, budget, distribution of work

6.1 Work packages

6.1.1 WP1 – State of the art and exposing the meta-data

The initial part of the project is to determine how to expose the meta-data to the programmer. Therefore the project begins with a state-of-the-art review of currently proposed solutions for defining performance requirements. The WP is intended to firstly define what exactly should be included in the meta-data to get a proper description of performance in applications. Further, what must be determined is guidelines for how, where, and when the meta-data should be injected in software to ensure stability and minimal performance overhead. Included in this WP is tool support for automatic meta-data injection based on the findings earlier in the WP.

- Task 1.1 State-of-the-art review and defining the meta-data
- Task 1.2 Injecting the meta-data in software
- Task 1.3 Tools for introducing meta-data in software

6.1.2 WP2 – Feedback monitor

WP2 covers the monitoring of the application software. This includes how to interconnect the monitor with the application, determine the information flow between the application and the monitor and also how to forward this information to the resource allocation controller. The main problems to tackle here is how the application software can be monitored without affecting the functionality or the performance of the system. State-of-the-art monitoring systems such as the MIT heartbeats [HES$^+$10] will be evaluated as complementary research to the proposed energy awareness monitor. A monitoring interface will be developed and its cross-domain applicability will be evaluated on Android, PC and server systems.

- Task 2.1 Evaluate complementary work and possible integration
- Task 2.2 Construction of the monitoring interface and framework
- Task 2.3 Cross-domain evaluation of monitoring system

6.1.3 WP3 – Energy aware controller

WP3 is about using the meta-data definitions in WP1 and the monitoring data from WP2 and use it for energy optimizations by a runtime system controller. The main challenge in WP3 is to define the resource allocation to achieve maximal energy efficiency. Several well established optimization methods based on system models will be evaluated. This means firstly modeling the underlying system and secondly applying mathematical optimization methods on the model. The work was initiated in the thesis of the PI, and WP3 is intended to refine the optimization methods for a larger selection of platforms and improve on existing optimization methods. WP3 also include the implementation of tools for automatic model generation for increasing the portability of the framework and finally the implementation of the runtime system controller itself.

- Task 3.1: Modeling energy consumption
- Task 3.2: Tools for generating energy models
- Task 3.3: Adopting optimization methods for energy efficiency
- Task 3.4: Implementation of runtime controller

6.1.4 WP4 – A framework for energy aware programming and demonstrator

The final stage of the research is to bind the previously created methods into a package usable for the programmer. Construction of the API framework used for energy aware programming is defined as the first item in WP4. The API is a result from the meta-data declaration in WP1, the monitor interface in WP3 and the runtime system in WP3. Additional efforts in this WP is especially put on supporting interdisciplinary domains of platforms. A software library, pragma or other interface is the final deliverable and is the link between the application software and the runtime system. Additionally, we demonstrate its usability with single small scale use-cases, which also function as evaluation of the API. Lastly a large scale demonstrator is created running a complete version of the energy aware software on an off-the-shelf consumer device such as an Android phone or tablet.

- Task 4.1: Construction of API for energy aware programming
- Task 4.2: Single use-cases
- Task 4.3: Demonstrator

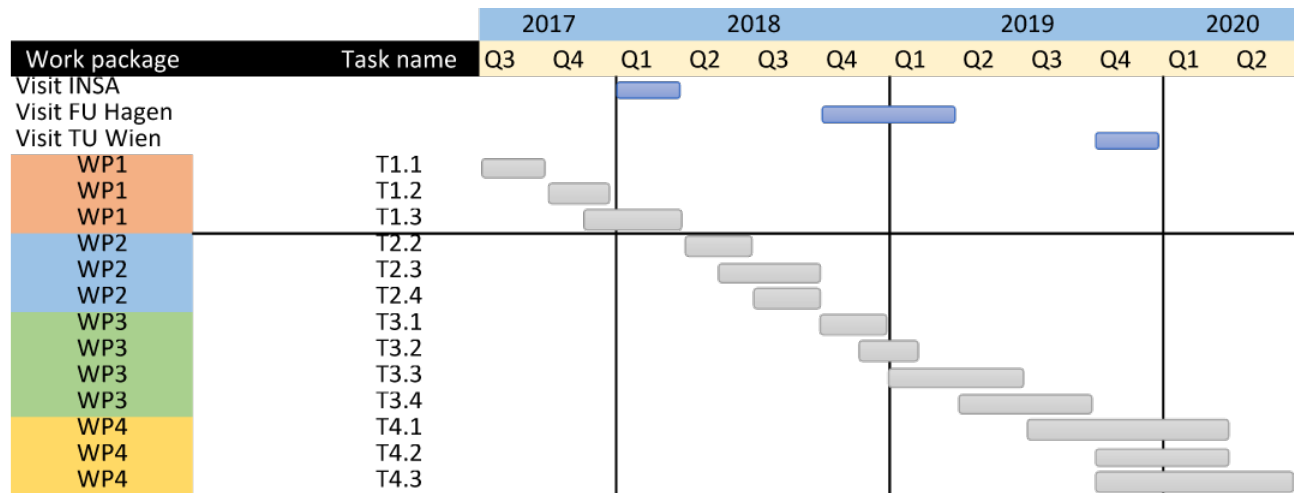The timeline of the research work is summarized in Figure 4.

Figure 4: Time schedule for the project

6.2 Budget

The budget of the project is defined in the following table:

Cost	2017	2018	2019	2020
Salary Post-doc	xxx	xxx	xxx	xxx
Travel	xxx	xxx	xxx	xxx
Open access fees	xxx	xxx	xxx	xxx
Mobility	xxx	xxx	xxx	xxx
Total	xxx	xxx	xxx	xxx

7 Research team and collaboration

The team is internationally well connected and has established cooperation with several teams abroad and nationally. Our lab and the PI has worked within several international and national research projects such as RECOMP, ParallaX and INTERSYS with close ties to Finnish industry like Nokia, abb, Kone, Ericsson etc. The cooperations listed here are the ones relevant to this application and that are documented through the most important joint publications.

7.1 Collaboration

The cooperation between Prof. Jean-Francois Nezan, INSA, Rennes, and the embedded systems lab has concentrated on the run-time management of dataflow networks, and has been executed through exchange of PhD students. INSA de Rennes has a strong background in tool support. The previously developed PREESM tool which was used to determine parallelism in data-flow programs to be used in a power optimizer developed by the PI in Åbo Akademi University. This work resulted in the best paper award at the 2014 DASIP conference and is of great importance

especially for the meta-data injection in WP1 and the demonstrators in WP4. The PI visited INSA de Rennes for 4 months in 2013-2014, which resulted in two journal publications and two proceedings publications.

Previous exchanges between Åbo Akademi and INSA de Rennes:

* Jerome Gorin (INSA ⟶ Åbo) Spring 2010
* Andreas Dahlin (Åbo ⟶ INSA) Spring 2010
* Julien Heulot (INSA ⟶ Åbo) Autumn 2013
* Simon Holmbacka (PI) (Åbo ⟶ INSA) Winter 2013-2014

A cooperation between Holmbacka and Prof. Jörg Keller at the FernUniversität in Hagen started in 2014, since which we have worked on energy efficient scheduling for multi-core systems. The lab group in Hagen has previously worked on performance models for very parallel systems such as the Intel SCC. One scientific journal and one proceedings publication about energy efficient scheduling has been published during this cooperation, and this connection is therefore important mainly for the energy optimization in WP3 and the construction of the controller in WP3. The PI is currently working in Hagen as a Post-doc exchange, and the cooperation with FernUni is currently very active.

The PI has been cooperating with the Electronic Commerce Group in TU Wien led by Prof. Ivona Brandić since spring 2015 inside the COST action IC1305 framework. Work on cost and energy efficient cloud scheduling was done by extending the Philharmonic cloud simulator[13] created at TU Wien with a multi-core model developed by the PI at Åbo Akademi University. A journal has been accepted in the IEEE Transactions journal as a result of this cooperation. Their expertise is complementary for constructing the runtime system in WP3, especially valuable are their expertise on cloud environments.

The most recent of my collaboration network is Dr. Alexandra Jimborean from Uppsala University in Sweden. Their lab group is working on compile-time and runtime code analysis and transformation for performance and energy efficiency using LLVM. Her group has developed a monitoring and feedback system for re-organizing instruction calls based on their memory intensity. With this knowledge, their input is essential in WP2 when constructing the monitor system.

7.2 Relation to strategic centers of research

Since the Strategic Centres for Science, Technology and Innovation are being shut down, there is no concrete cooperation planned. However many of the SHOKs are planning new ways to continue their activities and since Prof. Lilius of the Embedded Systems Lab in Åbo Akademi is a member of the FIMECC SG, he will have the opportunity to connect up the work in this project with the work done within FIMECC.

8 Researcher training and research careers

Researcher training and supervision: The supervision of PhD students are carried out as teamwork within participating senior researchers, but every student has also official supervisors with whom the student makes a study and research plans according to the university regulations. Each laboratory consists of researchers at various levels of their research career. It is paramount for all of them (including professors, researchers and staff) to periodically participate in the researcher training programs and renew their education/training.

Promotion of Research Career: With the current application, we seek funding for a post-doc researcher to work full-time within the project. Combined with the international co-operation, exchange period and close interaction between participating institutes, the project gives to the involved post-doc good basis to proceed in the academic career after the project.

9 Mobility plan

The mobilities are planned as follows and illustrated in Figure 4:

- Simon Holmbacka (PI) will visit Prof. Maxime Pelcat for 3 months in the end of WP1
- Simon Holmbacka (PI) will visit Prof. Jörg Keller for 6 months for constructing the controller in WP3

---

[13]http://philharmonic.github.io/

- Simon Holmbacka (PI) will visit Prof. Ivona Brandić for 3 months in the beginning of WP4 to collaborate on the use-case and demonstrator in a cloud environment

The exact details of the mobilities 2017⟶ will be defined later in the project based on the availability of the collaborators and their time schedule.

Summary of past mobilities by the PI

Project: Post-Doc research
Partners: Jörg Keller, FernUniversität in Hagen, Germany — September 2015 – ongoing
Project: Energy Efficient Cloud Simulation
Partners: Dražen Lučanin, Ilia Pietri, Ivona Brandić, TU Vienna, Vienna, Austria — May 15 – May 30 2015
Project: Power-Aware HEVC Decoding with Tunable Image Quality
Partners: Erwan Nogues, Maxime Pelcat, INSA de Rennes, France — October 2013 – March 2014
Project: Innovation and Creativity for Complex Engineering Systems
Partners: João Fernandes, Universidade do Minho, EFACEC Inc., Portugal — January 31 – 17 February 2012
Project: Safe Motor Controller in Mixed-Critical Environment
Partners: José Luis Gutiérrez, University of Granada, Spain
Miguel Méndez, Seven Solutions Inc., Spain — November 5 – November 17 2012

References

[Aag01] Jan Aagedal. *Quality of Service Support in Development of Distributed Systems*. PhD thesis, University of Oslo, Twente, Netherlands, March 2001.

[BSA11] Teigerwald Bob S and Agrawal Abhisek. Developing green software, June 2011.

[DGR+74] R.H. Dennard, F.H. Gaensslen, V.L. Rideout, E. Bassous, and A.R. LeBlanc. Design of ion-implanted mosfet's with very small physical dimensions. *Solid-State Circuits, IEEE Journal of*, 9(5):256–268, Oct 1974.

[EEKS09] Stijn Eyerman, Lieven Eeckhout, Tejas Karkhanis, and James E. Smith. A mechanistic performance model for superscalar out-of-order processors. *ACM Trans. Comput. Syst.*, 27(2):3:1–3:37, May 2009.

[Epe15] Epec Engineered Technologies. New battery technology developments, May 2015.

[HES+10] Henry Hoffmann, Jonathan Eastep, Marco D. Santambrogio, Jason E. Miller, and Anant Agarwal. Application heartbeats: A generic interface for specifying program performance and goals in autonomous computing environments. In *Proceedings of the 7th International Conference on Autonomic Computing*, ICAC '10, pages 79–88, New York, NY, USA, 2010. ACM.

[HHL+14] Simon Holmbacka, Fredric Hällis, Wictor Lund, Sébastien Lafond, and Johan Lilius. Energy and power management, measurement and analysis for multi-core processors. Technical Report 1117, 2014.

[HLL15] Simon Holmbacka, Sbastien Lafond, and Johan Lilius. Performance monitor based power management for big.little platforms. In Dimitrios Nikolopoulos and Jose-Luis Nunez-Yanez, editors, *Workshop on Energy Efficiency with Heterogeneous Computing*, page 1 6. HiPEAC, 2015.

[HNP+14] Simon Holmbacka, Erwan Nogues, Maxime Pelcat, Sbastien Lafond, and Johan Lilius. Energy efficiency and performance management of parallel dataflow applications. In Ana Pinzari and Adam Morawiec, editors, *The 2014 Conference on Design & Architectures for Signal & Image Processing*, page 1 8. ECDI Electronic Chips & Systems design initiative, 2014.

[HSC+11] Henry Hoffmann, Stelios Sidiroglou, Michael Carbin, Sasa Misailovic, Anant Agarwal, and Martin Rinard. Dynamic knobs for responsive power-aware computing. *SIGPLAN Not.*, 46(3):199–212, March 2011.

[Seg11] Vanessa Segovia. *Adaptive CPU resource management for multicore platforms*. Licentiate thesis, Lund University, Sep. 2011.

[SKK11] V. Spiliopoulos, S. Kaxiras, and G. Keramidas. Green governors: A framework for continuously adaptive dvfs. In *Green Computing Conference and Workshops (IGCC), 2011 International*, pages 1–8, July 2011.

[Sta06] Venkatesh Pallipadi Alexey Starikovskiy. The ondemand governor. In *Proceedings of theLinux Symposium*, 2006.

[tB15] Steven te Brinke. *Developing Energy-Aware Software*. PhD thesis, Universiteit Twente, Oslo, Norway, Juni 2015.

[Tri15] Rob Triggs. How far we've come: a look at smartphone performance over the past 7 years, July 2015.