**1 Executive Summary**

Energy consumption is currently a challenge in all domains of computer systems. IoT, cyber physical systems and other mobile systems demand solutions to lower the energy consumption while keeping the same amount of computations, and high-end systems, servers and exascale computing demand more computations for the same energy. These problems were traditionally solved by the hardware evolution, but since we now witness the end of the Dennard scaling [DGR+74], this development is no longer feasible. As the improvement in semi-conductors can no longer keep up with the required pace, evolution in energy efficiency must be shifted over to the software. The vision behind this research is to significantly improve the energy efficiency of computer systems by including the application software in the decision of allocating resources – this branch of self-awareness is called ***“making the software energy aware”***. Energy aware software forms a stronger link between applications and the runtime system by means of an information flow used to steer the allocation of resources, which increases the self-awareness of software. This advancement in software design is required for the research community to take the next step into energy efficient computing.

**2 Rationale**

From tiny embedded systems to large scale server farms and diverse IoT systems, energy efficiency is becoming the main struggle for system usability, expansion, reliability and scalability. The drivers for solving the issues differ depending on the area of focus; the very usability of battery powered devices depends on their ability to provide the user with the satisfactory experience while consuming the minimum amount of energy. Desktop and laptop systems require high levels of energy efficiency in in order to (a) lower the noise levels from active cooling, (b) increase their reliability by reducing the heat and (c) reduce the electric bill. Taking into account the bigger picture, the rapid expansion of large-scale servers is not in line with the ecological development, and now contributes to over 2% of the total energy consumption in the US[[1]](#footnote-1), whilst Facebook alone releases 650k tonnes of carbon annually[[2]](#footnote-2). Lastly, without the increased energy efficiency, high performance computing centres cannot reach exascale performance. The US Department of Energy[[3]](#footnote-3) defines a maximum power envelope of 20 MW for a single data centre, but with the current technology, a modern exascale computing centre requires over 7000 MW.

In order to provide a continuous performance improvement in computing systems, the clock frequency wall was avoided in the beginning of the 2000s by the introduction of multi-core based systems. This led to a new challenge in the mid-2000s: the CPU power wall [DKM+12, WS13]. Indeed, due to the physical limitations of semi-conductor materials, the power dissipation of a fixed chip area is limited. With the evolution in the fabrication of semi-conductor devices from a 10µm manufacturing process in the 1970s to the commercialisation of the 14nm based technology in 2015, more transistors are squeezed into a smaller silicon area, which increases power density [PV07]. This eventually leads to dark silicon, where all of the components of a chip cannot be operated at the same time due to both its power dissipation and the heat generated. Until now, the effects of the power wall have been limited by the design of the more energy efficient transistors with, for example, lower voltage levels. Which has led to an increase in the transistor efficiency in proportion to an increase of the transistor density, - a phenomenon called the *Dennard scaling* [DGR+74]. However, the current increase in the transistor efficiency is no longer proportional to the increase of transistor density [WS13], and the semi-fabrication is currently in a post-Dennard scaling area, where a substantial increase of the performance of computing systems cannot be achieved until both the current power dissipation and conductor device the related energy consumption problems are resolved.

In order to eradicate the above-mentioned constraints in this interdisciplinary domain, specific software must inevitably be introduced to make computer systems energy efficient again! The key challenge is to utilise the available hardware resources as efficiently as possible. Since general purpose applications behave in a very dynamic way, a runtime environment must continuously adjust the usage of the resources based on the execution of the applications. In Linux based systems, attempts have been made to incorporate this behaviour by using clock frequency scaling and deep sleep states to scale the hardware resources according to the demand. The limitations with the existing approaches are that the resources are allocated based on poor metrics. There is no interoperability between applications and the runtime system, and resources are usually allocated based on indirect metrics such as the system *workload*. Such a metric does not describe performance demands in applications, and therefore often causes incorrect resource allocation, which, in its turn, leads to an additional waste of energy[HNP+14, HLL15].

In this research project we will enhance the application software with the inclusion of ***energy awareness***. This new property will enable applications to get directly involved in the allocation of resource. Energy aware software is able to continuously report the performance demand to the runtime system in its own specific metric. With this new vital information, the runtime system would be able to allocate resources (usually with clock frequency scaling and with sleep states) with a much greater accuracy than using indirect metrics, such as the workload. **As a result of the inclusion of energy awareness in software on real-world Linux platforms in our previous research [Hol15], an impressive saving of 50% to 60% in energy was achieved – without compromising on the performance!** We are therefore confident that adding energy awareness in general purpose software can reduce the consumption of energy significantly. Figure 1 illustrates the purpose of the research: Part A) shows the current approach without any interoperability between applications and resources. Using this approach, resources are allocated based on the level of workload or other indirect means of metrics. Part B) in Figure 1 illustrates the inclusion of energy awareness. This is the added interface between the application and the runtime system through which the application communicates. It produces the following benefits: (a) a better accuracy of resource allocation, (b) an increase in self awareness and (c) reduced energy consumption.

The research in energy efficient computing is a broad field spanning several levels of hardware from IoT to exascale. However, the common feature is the ability to propagate information from the highest software layer to the runtime system, and on to the hardware. Throughout all computing domains in the area of energy efficient computing, we have identified three critical challenges to address in our research, which are further described below.

**Challenge 1: Self Awareness**

Applications must be able to properly express their intentions in terms of performance requirements during an execution. Such requirements are added to the application software in form of performance meta-data – a performance goal the application is due to achieve. The challenge is therefore to provide a framework with the capability of handling and expressing performance meta-data in an interdisciplinary domain. As was previously mentioned, the traditional way of solely monitoring the workload is very inaccurate and often counter-productive in terms of both performance and energy. We will 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 intention of the application in its own understanding of performance. For example, the end user of a video decoder is satisfied when the decoder produces a framerate of 25 fps because of the properties of the human eye and the understanding of moving pictures. A higher framerate, whilst not producing any higher satisfaction for the end user, is merely a waste of energy. Therefore, we will investigate the types of meta-data needed to describe the intentions of an application sufficiently accurately. These types of meta-data will have to have a sufficiently descriptive form, which can be generalised 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 should be achieved rather effortlessly in order for the programmers to make an additional effort. A trade-off between descriptiveness and programmer effort is to be expected.

**Challenge 2: Interoperability for Energy Efficiency**

The interoperability challenge in the domain of energy awareness in application software is to create an ecosystem capable of exchanging information about the intentions of an application to the runtime software layer. Secondly, the exchanged information must be interpreted by the runtime layer to make sensible decisions regarding the allocation of resources. Since applications in general are dynamic, the performance of the application depends on a wide range of factors, such as memory intensity, use of caches, user interrupts, the data sizes and data types used, interactions with other applications etc. This means that the performance of an application cannot be guaranteed with fixed resources. In order to adapt the resources allocated to the application, the performance of the application must be monitored at runtime. In case the currently allocated performance is either too high or too low, the monitor reports this issue to the controller, which allocates or deallocates resources to accommodate for the real requirement of resources. Therefore, we recognise the need for a proper monitoring framework to insure the interoperability between applications and the runtime system. 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, a considerably low overhead must be guaranteed to avoid interference with the energy consumption and the performance of the host system.

**Challenge 3: Smart Adaptivity**

Given a proper dataflow from applications to the runtime system, the final challenge is the allocation of resources. Allocating resources efficiently means having a knowledge of how the resources of one form affect the performance of the application requesting the resources. Even though feedback control loops is a mature area of research, the methods must be adapted and applied to the general purpose computing domain. Together with machine learning, large data and pattern recognition, smart adaptivity in the control loop help to bring the framework of energy aware software to a broader domain of computing systems from IoT to exascale systems.

**2.1 Previous Work**

The embedded systems lab at Åbo Akademi University has for the past six years focused on low power and energy aware software. During those years a special interest was placed in mobile phone processors and their significance for energy awareness in general. This is because it was noted that the trend for performance requirements by far exceeded 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 by 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 titled *“Energy Aware Software for Many-Core Systems”* two major guidelines were presented for creating energy aware software on modern many-core hardware. The implementation of these recommendations in software has proven to reduce the energy consumption by up to 50% without compromising the performance [HLL15], especially on mobile multi-core hardware. These same recommendations under the title *“Energy Aware Mapping” and “Energy Aware Resource Allocation”* were applied to tailor the resource allocation to the software execution. In addition, as a sub-project of the PhD thesis, a prototype runtime system was implemented. Based on this software, practical state-of-the-art demonstrators were also created: e.g. a low power video transcoding cloud system[[4]](#footnote-4) demonstrated at the Millennium Pavilion SHOK Summit in Helsinki, and at the DIGILE Workshop “Rolling up the Sleeves”. Also, the Android app “Low Energy Player” was implemented as a state-of-the-art demonstration and is available on Google store[[5]](#footnote-5). This research will go beyond the state-of-the-art through the development of a programming framework for energy aware software and practical demonstrators.

**2.2 Related Work**

The concept of *low energy programming*, or *low power programming*, has previously existed as a focus on the programming paradigm or the programming syntax. Guidelines from Intel [BSA11] suggest the use of a certain level of loop unrolling, vectorisation, memory intensity cache usage etc. for achieving maximum energy efficiency in combination with Intel compiler tools. Such recommendations are only applied onto the algorithms in the programme, and do not cover the intention of the programme to function efficiently alongside the runtime system allocating the resources. The low power programming guidelines from Intel also require the programming to construct the programme in a certain way in order to become energy efficient, and the underlying hardware architecture to be known. In his thesis *“Developing Energy-Aware Software"* by Brinke [tB15] the author describes programming languages for modularity and modelling resource consumption for software. The *awareness of energy* is tightly connected to the application code, and the programmer is expected to follow certain programming patterns to make the software energy aware. In contrast, the planned framework for energy efficient software only requires the insertion of meta-data into the software. This means minimal effort for 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, whereas after the runtime environment allocates the required resources.

Our newly funded Academy of Finland project titled ***“Efficient Stream Computing”*** (ESC) introduces a formal and practical paradigm for measuring the efficiency of software execution through the notion of *Fitness*, which characterises how much of the microarchitecture the task is able to exploit. This will enable the optimisation of Stream Computing Systems for energy efficiency, performance or cost with minimal overhead in the development. The commonality of this project is energy efficient execution, but whilst ESC focuses more on *where to map* a task, the primary purpose of this project is to *optimise the allocation of resources* and demonstrate *how software can influence this decision*.

The ***Carbon Research Group***[[6]](#footnote-6) at Massachusetts Institute of Technology (MIT) has developed a heartbeat framework to evaluate the performance as a generic parameter in software construction. This framework is capable of measuring the 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 measuring the framerate in a video decoder, but a controller is then needed to allocate the resources in order to keep the performance at a given setpoint. We have considered 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[[7]](#footnote-7) creating the Odroid family boards the recently released 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 to introduce energy awareness to the system, the practical results are poor. In other words, the scheduler most often schedules a thread on the *wrong core*. This results not only in energy inefficiency, but also in the poor performance of the applications and an unsatisfactory user experience. We will 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[[8]](#footnote-8) dedicated to standardising and optimising 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 is the notion of energy awareness, which is a crucial point as most devices operate on battery-only power. We will work closely with this project and we will introduce the benefit of energy awareness into the IoT systems.

**EMBECOSM**[[9]](#footnote-9) focuses on providing the GCC compiler with the notion of energy efficiency. Practically, this means learning which compiler flags minimises energy consumption for a selected architecture. The outcome of this project is similar to that in **OpenTuner**[[10]](#footnote-10) by MIT, which is capable of offline optimisation of multi-criteria problems. Both projects provide a metric for offline optimisation. However, runtime support, suggested in our project, is not part of their scope. Runtime optimisation 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 optimisations, for example, cannot not predict what kind of video format is being used in a video decoder.

The **StarPU**[[11]](#footnote-11) project at INRIA Bordeaux and the **PEPPHER**[[12]](#footnote-12) project both present a runtime system for minimising 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. However, they only consider optimisations through performance, not energy efficiency.

Therefore, whilst there are a number of ongoing projects, researching….. none of them so far has endeavoured to solve… (focus on energy efficiency and sustainability!)

**3 Objectives and Expected Results**

**3A Objectives of the Research**

Our objective is to address the energy consumption problem in computer systems. In most recent systems, power managers have been implemented to scale the performance of the system according to the current resource demand. Various implementations exist depending on whether the system is a large cloud farm or a small IoT system, but the main constraint in the current solutions is the inability to accurately express resource demands. In mainstream Linux systems, the resource demand is measured as the utilisation 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 2. A high workload indicates that the system needs more resources, and the clock frequency is scaled up to decrease the workload of the CPU. Frequency setting can be based on different policies implemented in *frequency governors*, and the most common policy is called “ondemand”[Sta06]. This frequency governor scales the clock frequency to the maximum value as soon as the workload threshold is reached. Next, 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 accurately enough. A high workload does not necessarily mean that an application requires more performance; it simply describes how much of the CPU the application is currently using. It is merely an indirect effect of executing an application, as illustrated in Figure 2. An incorrect allocation of resource results in either inefficient performance or energy waste, or both. As shown in the results of our previous work in [HLL15, HNP+14], executing an application unnecessarily fast wastes significantly more energy than executing an application at a moderate, but still sufficiently fast, performance level.

Other work on attempting 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 a higher memory intensity stalls the CPU enough to not benefit from high clock frequency. The fundamental problem, however, still remains unsolved because the governor models the performance requirement of the application indirectly. Therefore, if the model does not fit the application, the latter is allocated either excessive or insufficient resource.

What we propose is to make the application software itself energy aware and the objective of this research is to develop a complete framework for programming energy aware software. Using this software, applications become involved in the resource allocation, which is a necessary step for reaching the next level of energy efficiency in computer systems. Therefore, a power management system compatible with energy aware software requires an interface between the application and the runtime system, which is currently non-existing. This research aims at providing such an interface, as well as the necessary tools needed to use it. Our current design recommendations for energy aware programming extends the application to signal resource requirements to the runtime system, which allocates the hardware resources based directly on the requirements rather than the indirect metrics like the workload. With this framework, resources will be allocated more efficiently leading to a lower energy consumption.

Also, define (or at least try to) clear results/ benefits as a result of this research project – possibly in bullet points!

**Hypotheses**

The project states the following hypotheses:

1. Performance requirements in computer systems will continue to increase significantly faster than transistor technology and battery technology development, software must thus utilise the resources more efficiently.
2. Energy aware software can utilise hardware resources more efficiently because hardware resources are allocated based on actual performance requirements.
3. Software can be made energy aware with minimal performance overhead and minimal programmer efforts.
4. Up to 50% of energy can be saved using energy awareness in software.

**Results**

Define (or at least try to) clear results/ benefits as a result of this research project – possibly in bullet points!

**3.2.1 Previous Work on Energy Aware Software**

In previous work, Eyerman et al. [EEKS09] claim that no single throughput metric is fundamentally generic for multiprogramme 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 their 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 be placed on the link between applications and hardware resources in a single computer system. Our methods for energy aware programming will not be strongly tied onto a certain programming language, and the framework itself will have the flexibility to be integrated from various environments such as servers and PCs and Android.

On the other hand, runtime systems for minimising 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 added 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 monitoring 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 projects do not support any proper framework for declaring meta-data requirements and monitoring 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 previouslyemphasised. Our research project will also ensure a proper balance between academic research and practical usability, which means that there is a focus on both 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.

4 Expected Scientific and Societal Impacts and Potential Breakthrough

of the Research

**3B Effects and Impact Beyond Academia**

Our expected impact on the research community is to introduce energy awareness to software. In other words, the need for communication between the application layer and the runtime environment. Our potential breakthrough is to introduce **energy awareness as a natural part of programming**. For decades it has been a natural step to introduce meta-data in software for creating parallel programmes. The programmer has been willing to add #pragmas in OpenMP, Keywords in Cilk or Initialisations in OpenCL in order 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 straightforward. Without energy aware programming as the underlying notion, runtime systems in any domain have no common ground on which the decision making is based. Optimisations 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 common platform to cooperate on.

4.1 Applicability

* Energy efficient programming: Today most guidelines for energy efficient programming are driven by creating a code with efficient algorithms such as using loop unrolling, SIMD vectors, compiler options etc., which generates software specifically for a given platform. Instead, by relying on application meta-data describing software requirements, similarly as Cilk and OpenMP handle forking and joining task, the allocation of resource is based on what the software actually needs, rather than 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 customisation of software to particular embedded architectures. With the suggested energy awareness framework, the mapping and scheduling decisions are shifted from the developer onto 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 becomes more standardised. Even the potential for automatic runtime system generation emerges as a result of standardising the foundation.
* 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.

**3C Publication Plan**

We intend to publish in 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 yearly lump sum is included to cover the publication fees. Further, we also plan to be visible at industrial events 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 optimisation (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.

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

**Support from Research Environment**

The research team will be well supported by the infrastructure available at through the Centre for Computer Science (TUCS). TUCS boasts a long history of high-level achievements of its affiliated researchers, in terms of articles in high-level journals and conferences and a high number of citations. TUCS has been a Centre of Excellence of Research of the Academy of Finland in the very first round of such centres in Finland, 1995-1999 and in 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 and the group in TU Wien led by Ivona Brandic and one between the PI and the group in University of Tirana led by Neki Frasheri, one of which has been published in IEEE Transactions.

The PI together with EIT Digital recently released an online Coursera course on the subject in “Development of Real-Time Systems”[[13]](#footnote-13) currently followed by over 7000 students world-wide. Our team has a full time lab technician constructing specialised equipment needed in our experimental work. The team therefore has the strong knowledge of 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, with 0% performance overhead in the host system. The team is experienced in building tools based on 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.

**Utilisation of Research Infrastructure**

Since our research is mainly based on theoretical computer engineering and implementation work, no major research infrastructure is expected. We utilise mainly open source tools or other free software from our own research community. For the evaluation and demonstration efforts we require the latest hardware technology especially in the embedded domain and eventually access to larger cloud systems.

**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. To ensure success, the critical points and risk factors are listed as follows:

* *Software design is unfeasible*. With an infeasible design, the intended research results cannot be achieved without a major re-design. With our previous experience in energy aware software, and the results gathered we are confident about the main directions in the research. Although we intend to monitor the feasibility early in the project to ensure an eventual re-design in as early stages as possible.
* *Learning curves lead to delays*. Learning new methods and tools is always a challenge. By exceeding the time budget in methods and tools, the project can be delayed. Our lab environment keep up-to-date with tools and methods by weekly discussions internally in the whole lab group. Practical questions about research can be posted in such sessions, and people share their experience.
* *Relocation of research partners*. This project relies on international cooperation. In the event of relocation of key persons, mobility can be postponed and the research ultimately delayed. By having a large network of contacts, we have a safety net in case of relocation events. Work and timelines can be re-adjusted while keeping the core research intact.
* *Access to hardware platforms*. Without appropriate hardware, the results in evaluation are inadequate and insufficient. For covering this, risk The PI 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. On a larger scale, our team has previously obtained the Amazon EC2 grant[[14]](#footnote-14) twice enabling access to large scale servers.

**Data Management Plan**

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[[15]](#footnote-15) has been selected as the platform where we plan to publish measurement results and each submission in Zenodo can get a citable DOI.

**5 Ethical Issues**

This research project has no ethical issues.

**6 Implementation: Schedule, Budget, Distribution of Work**

**6.1 Work packages**

Without transferring this section into word, the following comments:

1. The work packages and tasks lack in substance. The substance will appear once you are sorted with your **objectives** and **methodology**.
2. The work packages and tasks herein should be coordinated and harmonised with the claimed objectives and methodology. Once methodology is clear, it will be possible to create this section.

**Budget**

**7 Research Team and Collaborative Partners**

The PI was awarded the Researcher of the Year for best PhD thesis during the year 2016; this project is a continuation of this thesis. 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 ESC, RECOMP, ParallaX and INTERSYS with close ties to Finnish industry like Nokia, ABB, Kone, Ericsson etc.

**Collaboration**

The cooperation between Prof. Jean-Francois Nezan, INSA, Rennes, and Åbo Akademi University has concentrated on the run-time management of data-flow networks, and has been executed through an exchange of PhD students. The PI visited INSA de Rennes for 4 months in 2013-2014, which resulted in two journal publications and two proceedings publications. INSA de Rennes has a strong background in tool support. The previously developed PREESM[[16]](#footnote-16) tool which was used to determine parallelism in data-flow programmes to be used in a power optimiser developed by the PI at Åbo Akademi University, and this work resulted in the best paper award at the 2014 DASIP conference. A 3 month research mobility is planned early in the project to bring early tool integration of meta-data injection in WP1, and to determine the meta-data model needed for such tool integration. In the later stages of the project, the utilization of PREESM is expected in form of a demonstration environment for WP4.

A cooperation between Holmbacka and Prof. Jörg Keller at the FernUniversität in Hagen started in 2014, since when 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. The PI is currently working in Hagen as a Post-doc exchange, and the cooperation with FernUni is currently very active. 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 optimisation in WP3 and the construction of the controller in WP3. A 6 month visit is planned in the middle of WP3 during which the most critical part of the smart controller is to be constructed.

The PI has been cooperating with the Electronic Commerce Group in TU Wien led by Prof. Ivona Brandic since the spring of 2015 inside the COST action IC1305 framework. Work on cost and energy efficient cloud scheduling was done by extending the Philharmonic cloud simulator[[17]](#footnote-17) 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. A 3 month research mobility is planned in the early WP2 firstly to evaluate complementary work and its suitability in our project. Secondly, their expertise is of highest value early in WP2 after the meta-data framework has been established in WP1 in order to adapt and evaluate our framework in cloud environments.

The most recent in the 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. Also, our collaboration with overlapping expertise is needed in WP4 constructing the demonstrator. No mobility is planned together with this team, but since Uppsala is located very close to Turku ad-hoc visits can be arranged without extensive pre-planning.

**Relation to Strategic Centres 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 at 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**  **Research Careers, Fulfilment of the Mobility Requirement and Researcher Training**

Researcher training and supervision: The supervision of PhD students is 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.

**10 Mobility plan for the Funding Period**

The mobilities are planned as follows and illustrated in Figure 3, but the exact details will be defined later in the project based on the availability of the collaborators and their time schedule.

* Simon Holmbacka (PI) will visit Prof. Jean-Francois Nezan for 3 months in the beginning of WP1.
* Simon Holmbacka (PI) will visit Prof. Ivona Brandi\_c for 3 months in the beginning of WP2
* Simon Holmbacka (PI) will visit Prof. Jörg Keller for 6 months in the middle of WP3.

1. Statistics according to the U.S. Office of Energy Efficiency & Renewable Energy: <http://energy.gov/eere/femp/resources-> datacenter-energy-efficiency. [↑](#footnote-ref-1)
2. https://sustainability.fb.com/en/our-footprint/. [↑](#footnote-ref-2)
3. http://ascr-discovery.science.doe.gov/2014/11/exascale-road-bumps/. [↑](#footnote-ref-3)
4. Picture of the ESLab demonstrator available at https://dl.dropboxusercontent.com/u/5260559/ClusterDemo.jpg [↑](#footnote-ref-4)
5. https://play.google.com/store/apps/details?id=org.videolan.vlc.LEL.lite.green [↑](#footnote-ref-5)
6. http://groups.csail.mit.edu/carbon/ [↑](#footnote-ref-6)
7. http://www.hardkernel.com/main/main.php [↑](#footnote-ref-7)
8. http://iot4health.utu.fi/?p=374 [↑](#footnote-ref-8)
9. http://www.embecosm.com/ [↑](#footnote-ref-9)
10. http://opentuner.org/ [↑](#footnote-ref-10)
11. http://starpu.gforge.inria.fr/ [↑](#footnote-ref-11)
12. http://www.peppher.eu/ [↑](#footnote-ref-12)
13. https://www.coursera.org/learn/real-time-systems/ [↑](#footnote-ref-13)
14. https://aws.amazon.com/grants/ [↑](#footnote-ref-14)
15. http://zenodo.org/ [↑](#footnote-ref-15)
16. http://preesm.sourceforge.net/website/ [↑](#footnote-ref-16)
17. http://philharmonic.github.io/ [↑](#footnote-ref-17)