CSR:Small:Exploring Timing-Energy Tradeoffs on Heterogeneous Computing Platforms

Project Summary

Summary.

Intellectual merit.

Broader impacts.

Keywords:

1 Introduction

In recent years, the microprocessor industry has turned to heterogeneous multi-core processor designs for the next generation of embedded systems. By integrating relatively slower, lower-power processor cores with relatively more powerful and power-hungry ones, it is possible to dramatically improve energy efficiency while achieving high performance. Such heterogeneous computing platforms are seeing widespread adoption in many cyber-physical systems (CPS), which are more sophisticated and intelligent computing systems that closely interact with the physical world, including advanced automotive systems, medical CPS, and smart robotics. For example, the ARM big.LITTLE, which is a heterogeneous computing architecture integrating relatively slow and fast cores, has been widely adopted in the automotive systems [**?**]. This heterogeneous computing technology has been used to serve multiple automotive needs such as infotainment as well as advanced driver assistance systems, and is shown to be able to "deliver exceptional power and performance that aligns to the vision of scalable solutions for automotive customers [**?**].

Two Fundamentally Conflicting Goals of Heterogeneous Computing in CPS: Timing Correctness and Energy Guarantees. A major challenge of reliably adopting heterogeneous multicore processors in CPS such as automotive systems is the need to ensure predictable real-time correctness (i.e., enabling timing constraints to be analytically validated at design time). The functional correctness of a CPS hinges crucially upon the temporal correctness, as the control operations depend on the processing of certain environmental sensing and computation tasks. Establishing real-time correctness requires real-time resource allocation methods, whose focus is on judiciously allocating resources to various tasks so that all the required timing constraints can be satisfied. Although traditional real-time resource allocation methods can ensure timing correctness, they often fail to make or mostly ignore energy guarantees, which is another critical goal of equal importance for many CPS due to the stringent size, weight, and power (SWaP) constraints imposed by such systems (even with sub-components powered by batteries). Energy guarantees would be critical for safety reasons of many systems and in general beneficial for system designers who have a fixed energy budget and want to support the maximum amount of workloads that require real-time correctness within that budget.

It is quite challenging to simultaneously consider the two goals of achieving real-time correctness and energy guarantees, because they are fundamentally in conflict. On one hand, real-time resource allocation methods often need to maximize the resource utilization and make greedy scheduling choices to guarantee timing correctness for the worst case. On the other hand, however, methods that yield energy guarantees typically require scheduling decisions to be energy-efficient for the current case, implying that in some cases it is wise to scale down the voltage and frequency of certain cores or even force them to idle.(?Cong: Best to incorporate Hank's data support and citations herein.?) A good real-time resource allocation algorithm may yield excessive amount of energy or thermal hotspots on the many-core chip; while energy-aware methods often fail to make real-time guarantees. Besides this challenge, the heterogeneity among cores can greatly complicate resource allocation because "choices" must be made when selecting the core upon which a task will execute. When jointly considering timing correctness and energy efficiency in the presence of such heterogeneity and dynamic computing needs, resource allocation becomes even more interesting but complicated due to the huge design space of timing and energy tradeoffs. Resolving these issues will be the focus of this proposal.

Proposed approach. Most existing works on real-time resource allocation in heterogeneous computing systems focus on guaranteeing timing correctness (e.g., see [**?**, **?**, **?**] for an overview). A few recent works [**?**] focusing on exploring the timing/energy tradeoff space in a heterogeneous computing system made the following critical assumption: all processor cores can operate at max speed all the time while sustaining safely. This assumption unfortunately invalid due to the critical "dark silicon" problem, i.e., essentially all processors are over-provisioned and have much larger max compute capacity than they can safely sustain (this has never been true for embedded systems before but it is now [**?**]). The thrust of this research is to

simultaneously achieve the goals of timing correctness and energy guarantees on heterogeneous computing platforms containing processor cores with varying speeds by answering the following research questions: **(*i*) how to guarantee timing while running processor cores as slow as possible to respect given energy budgets? (*ii*) how to quickly detect and avoid timing errors by utilizing the over-provisioned processing capacity in short bursts?** Our proposed research is fundamentally motivated by the observations that (*i*) it is significantly more energy efficient when slowing down processor cores (compared to the race-to-idle strategy), and (*ii*) processor cores cannot operate at max speed all the time and are (sometimes significantly) over-provisioned for what they can sustain safely. (Sec. 2 will provide detailed data support for these observations.)

?Cong: the above paragraph needs more work, as I feel I have not concisely and clearly shown the unique angle that was considered in existing works. That invalid assumption made in existing works needs to be re-worked.?

We intend to show that the fundamental timing/energy tradeoffs can be explored by leveraging recent research by the PI's group that has led to a new set of *optimal* resource allocation algorithms for uniform heterogeneous multicore-based systems containing processor cores with varying speeds [**?**, **?**, **?**, **?**, **?**]. These algorithms can analytically guarantee fast and analytically bounded response times for complex workloads implemented using rather general graph-based formalism and executed in a heterogeneous computing system. We will consider such algorithms as the basis for determining the best configurations of resource allocation strategies and processor cores' dynamic voltage and frequency scaling (DVFS) settings, which are most energy efficient for a given set of workloads. Significant further development is needed that considers the dynamic and heterogeneous performance and energy characteristics exhibited on the processor cores. Since there are numerous choices that can be made regarding resource allocation algorithm, the task-to-core mapping strategy, and DVFS settings, the potential solution space for this problem is vast. Tasks can be allocated for CPU resources globally (i.e., a task can be mapped onto any resource) or via partitioning (a task can only be mapped onto a designated resource). Also, task priorities may be either static or dynamic. The efficiency of DVFS-incurred energy saving can also be different for cores with different speeds. In this project, we propose to carefully evaluate the various alternatives in the space of potential solutions on top of real hardware and determine which configurations are preferable are given workloads. An real implementation-based empirical evaluation is thus a focus of this project.

Research objectives. We will pursue the following four-step plan to accomplish our research goal.

- **Step1: Identify the most energy-efficient configurations of resource allocation and DVFS for a given workload that guarantee timing:** We will first design workload mapping and resource scheduling algorithms for processing workloads on a heterogeneous multicore processor. The goal is to guarantee timing while minimizing energy consumption by running processor cores as slow as possible. For each devised algorithm, formal timing validation tests will be developed.
- **Step 2: Detect and avoid timing errors.** Although slowing down cores may be most energy-efficient, it may cause timing errors more easily (e.g., due to sudden workload bursts). Thus, we will further develop robust mechanisms for quickly detecting and avoiding timing errors. Our main idea is to achieve this goal through exploiting the over-provisioned processing capacity in short bursts.
- **Step 3: Carry out overhead-aware schedulability studies.** The research in the first two steps will provide solutions for simultaneously achieving timing correctness and energy guarantees on heterogeneous computing platforms. To evaluate their effectiveness in practice, we will incorporate them in real hardware (using the ARM big.LITTLE architecture) and conduct large-scale overhead-aware schedulability experiments with both synthetic workloads with widely varied parameters and benchmarks. We plan to apply an extensive methodology that is designed for comparing real-time resource allocation strategies in an overhead-cognizant way [**?**, **?**, **?**], which is proven to be effective for many real-time application systems.

- **Step 4: Conduct case-studies.** To determine if our proposed methods can be applied in practice, we intend to conduct case-study evaluations using several specific real-time workloads seen in automotive systems. For example, one such application we will consider is real-time object recognition, which is heavily performed in automotive systems for implementing tools such as collision avoidance and traffic sign recognition. We will evaluate the performance in terms of several specific metrics for using one or more preferable configurations identified in Step 3 to support such workloads.

Qualification. The PIs are well-qualified to conduct this research. The proposing team has worked on various aspects of real-time systems, heterogeneous multicore computing, and energy optimization in the past years, ranging from theoretical real-time analysis [**?**], real-time operating systems [**?**], heterogeneous multicore architecture [**?**], and energy efficient computing under various computer architectures [**?**]. These efforts have resulted in a number of publications accepted by several systems- and architecture-related premium conference and journal venues such as SOSP, RTSS, ISCA, ASPLOS, IPDPS, Micro, PACT, and IEEE Transactions on Parallel and Distributed Systems. PI Liu has been working on developing device driver and OS support for heterogeneous real-time systems accelerated by co-processors such as GPUs [**?**, **?**,**?**,**?**,**?**,**?**,**?**,**?**], resource allocation on heterogeneous platforms [**?**], and real-time schedulability analysis [**?**, **?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**]. PI Hoffman at the University of Chicago has rich experience in ?INPUT FROM HANK?

2 Background and Related Work

¡¡¡¡¡¡¡ HEAD In this section, we first describe the heterogeneous computing architecture and the associated energy and performance characteristics of such platforms. ARM's big.LITTLE will be used as an example heterogeneous computing platform we choose to focus in this document. To ensure timing correctness, real-time resource allocation techniques must be used. We will thus discuss traditional real-time scheduling-related terminology and workload models. Finally, we describe the experimental platform to be used in our research. Due to space constraints, our overview of prior work focuses on research of direct relevance to our agenda. ======= This section motivates the need for the proposed research. We first illustrate how the emergence of heterogeneous single-ISA architectures and dark silicon change timing and energy tradeoffs making previously useful resource allocation strategies extremely inefficient. We then discuss background in real-time scheduling and schedulability. Finally, we discuss related work scheduling for heterogenous systems. *Critically, we find that prior work has not...* ¿¿¿¿¿¿¿ 13d73883f1b0cb55eabc89896f2e4ce55936aaf0

2.1 Hardware Platform

2.1.1 Architecture Trends

Modern computing systems are constrained by *dark silicon*; *i.e.*, not all transistors can be powered at all times [**?**,**?**]. The emergence of dark silicon has led hardware architects to rethink processor design. Recent designs incorporate differing cores with a common instruction set and programming model [**?**, **?**]. Some cores are complex with deep pipelines and large issue width, while others are much simpler [**?**]. These *heterogeneous single-ISA* designs improve energy efficiency by executing computation on the most appropriate type of core.

While these processors present a wide range of performance and energy tradeoffs, they create a difficult scheduling problem. Not only, does a software scheduler have to map tasks to core types, it also most ensure that energy is minimized (to increase battery life) and that critical power threshholds are not violated. For example, the Exynos 5 processor (based on an ARM big.LITTLE architecture [**?**]) has a 5.5W peak power that is nearly $2\times$ the maximum sustainable heat dissipation, limiting peak speed to less than 1 second [**?**].

In addition, our prior work has shown that this processor achieves greater energy efficiency as it slows down [**?**].

Such a processor desing creates both a challenge and an opportunity for real-time embedded processing. The challenge is how to ensure that timing constraints are met while operating in the most energy-efficient configuration. The opportunity comes from the fact that the processor is overprovisioned, it has a much higher compute capacity than it can safely sustain. This extra compute capacity could be used in short bursts to prevent timing errors. The next section illustrates these concepts with empirical results.

2.1.2 Preliminary Results

Table 1: Two embedded platforms with different configurable components.

Type	Processor	Cores	Core Types	Speeds (GHz)	TurboBoost	HyperThreads	Idle Power (W)	Configurations
Homogeneous	Intel Haswell Mobile	2	1	.6–1.5	yes	yes	2.5	45
Heterogeneous	Samsung Exynos5 Octa	8	2 (A15 & A7)	.8–1.6 (A15) .5-1.2 (A7)	no	no	0.12	69

We use an example application to compare the different tradeoffs on a heterogeneous single-ISA processor and a homogeneous multicore. As a target application, we use a video encoder, composed of jobs, where each job encodes a frame. We instrument the encoder to report job latency and measure each processor's energy consumption. The two processors have different configurable resources, shown in Table 1.

The different shapes of these tradeoff spaces lead to different optimal resource allocation strategies. Empirical studies show that the *race to idle* heuristic, which makes all resources available and then idles after completing a job, is near optimal on systems like the homogeneous processor [**?**, **?**, **?**, **?**, **?**]. On systems like the heterogeneous processor, recent approaches save energy by keeping the system constantly busy and *never idle* [**?**, **?**, **?**, **?**, **?**].

To demonstrate how different scheduling decisions affect the different platforms, we compare the energy consumption of both race-to-idle and never-idle. We set a latency target equal to the worst observed latency for this application on each processor and measure the energy consumption of encoding 500 video frames using each heuristic. Figure **??** shows the results, normalized to the optimal energy found by measuring every possible resource configuration for each frame. Both heuristics meet the latency target, but their energy consumptions vary tremendously. On the homogeneous processor, *race to idle* is near optimal, but *never idle* consumes 13% more energy. Conversely, *never idle* is near optimal for the heterogeneous processor, but *race to idle* consumes 2$\times$ more energy.

These results demonstrate two key points:

- The conventional approach of allocating all resources and transitioning to a low power idle state is extremely inefficient on the heterogeneous processor.
- The heterogeneous processor can reach speeds of up to XX faster than it can safely sustain.

2.2 Real-Time Scheduling Algorithms and Schedulability Tests

The goal of real-time scheduling is to guarantee timing predictability; in other words, every task is schedulable if it meets the predefined timing constraints at design time. Being "schedulable" depends on whether task deadlines are hard or soft. For a hard real-time (HRT) task, its deadline must be met; while for a soft real-time (SRT) task, missing some deadlines can be tolerated. For example, in an automotive system, autonomous control and emergent collision avoidance are HRT tasks; while automatic lane following and traffic sign recognition can be viewed as SRT tasks. SRT constraints can be defined in several ways. We assume a well-studied SRT notion [**?**, **?**, **?**, **?**, **?**, **?**] that a SRT task is schedulable if its response time can be provably bounded. (Such bounds would be expected to be reasonably small.) In this research, we consider both HRT and SRT possibilities.

To ensure timing predictability, we must develop two kinds of algorithms: *real-time scheduling algorithms* and *schedulability tests*. A scheduling algorithm (i.e., scheduler) determines when to execute which task and on which resource. Scheduling algorithms can usually be divided into two major categories: (*i*) partitioning, and (*ii*) global scheduling. Under partitioning, tasks are statically partitioned among processors [**?**,**?**,**?**,**?**,**?**,**?**]. Different processors can apply different scheduling algorithms. A partitioning algorithm example is partitioned earliest-deadline-first (EDF), which uses the EDF algorithm as the per-processor scheduler. Under EDF, jobs with earlier deadlines have higher priority. In contrast to partitioning, under global scheduling, a single global ready queue is used for storing ready jobs [**?**, **?**, **?**, **?**]. Jobs are allowed to migrate from one processor to another at any time. A global scheduling algorithm example is global EDF (GEDF), under which jobs are EDF-scheduled using a single ready queue. Clustered scheduling is hybrid of partitioned and global scheduling in which tasks are statically assigned to a cluster of processors, among which the task can freely migrate. Note that the ARM's big.LITTLE platform we choose to focus in this research has already implemented the global and clustered scheduling policies. Moreover, under another categorization method of scheduling algorithms, we have two kinds of schedulers, fixed-priority and dynamic-priority schedulers, where a fixed-priority scheduler assigns a fixed priority to each task and all the jobs released by this task inherit the same task priority (e.g., rate-monotonic which assign higher priorities to tasks with shorter periods) and a dynamic-priority scheduler assign every job a different priority (e.g., EDF).

Depending on the scheduling algorithm being used, at design time a corresponding schedulability test verifies whether a task system will be schedulable, provided that the system behaves according to its parameterized specification. Finding an optimal scheduling algorithm that experiences no capacity loss (i.e., 100% resource utilization) is not always possible. If a non-optimal scheduling algorithm is used, then a schedulable task system may be deemed unschedulable by the corresponding schedulability test due to algorithmic capacity loss. Our goal of designing energy-efficient scheduling algorithms and schedulability tests is to minimize the overall capacity loss.

Workload Models. Real-time embedded systems commonly perform recurring operations. For example, the object recognition application commonly involved in implementing many autonomous-driving assisted functionality is naturally recurrent. Specifically, the camera sensor captures an image at a certain frequency; each image is compressed and then analyzed using certain object recognition algorithms, and is expected to be completed by the invocation of the next frame; e.g., each invoked job ideally should complete the corresponding workloads within 33*ms* for a 30 frame-per-second camera, as illustrated in Figure 2.

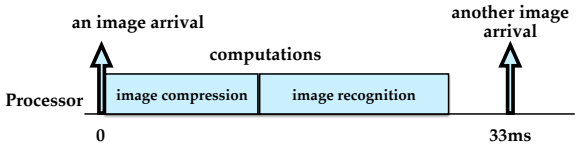

Figure 1: The sporadic job releasing pattern of object recognition applications.

Generally speaking, a specified real-time system consists of a collection of recurring real-time processes called "tasks". A widely studied general model of recurrent workloads is the *sporadic task model* [**?**]. In this model, the specified system is composed of a collection of recurring[1] sequential tasks, $\tau = \{\tau_1, ..., \tau_n\}$. Each such task τ_i releases a succession of *jobs* $J_{i,1}, J_{i,2}, ...$ and is defined by specifying a *period* T_i, a *relative deadline* D_i, and a *worst-case execution time (WCET)* C_i. Successive jobs of τ_i are released at least T_i time units apart, and one that is released at time t has a deadline at time $t + D_i$. Also, each such job executes for at most C_i time units. Each job released by any task has a *response time* that defines the duration from its release to its finish time. A task τ_i's response time equals the maximum job response time among all of its released jobs.

Automotive workloads. Heterogeneous computing platforms may be applied in a number of application domains where a wide range of timing and energy constraints are required. A particularly focused

[1] Although our focus in this document is on the notion of recurrence found in the sporadic model, we intend to consider other such notions in our work.

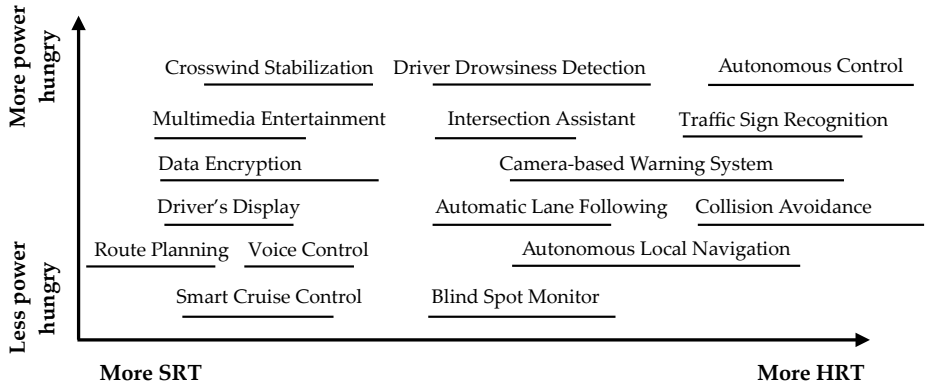

Figure 2: Spectrum of possible timing and energy requirements for a number of automotive workloads.

application domain we will investigate in this research is advanced driver-assisted automotive systems, where workloads with various timing and energy requirements are seen [**?**], as illustrated in Fig. 3. The automotive system represents an application domain where using heterogeneous computing platforms to achieve real-time performance and energy efficiency is particularly compelling. In such systems, cores with different performance and energy characteristics can be allocated to workloads with different timing requirements. For example, more powerful cores can be used to process HRT workloads such as traffic sign recognition and collision avoidance; while less powerful but more energy efficient cores can be used to process SRT workloads such as voice control and route planning. Dosing so may yield significantly less energy consumption compared to homogeneous platforms because less power-hungry workloads do not need to be processed by powerful cores, which results in necessarily high performance at the cost of energy loss. Due to these reasons, heterogeneous computing platforms such as our targeted ARM's big.LITTLE have been seeing widespread adoption in automotive systems [**?**].

Real-time OS. The OS platform in this project will be an open-source real-time Linux extension called LITMUS^{RT} [**?**]. The PI was involved in developing this testbed [**?**, **?**]. The main purpose of LITMUS^{RT} is to provide a useful experimental platform for applied real-time systems research. Currently LITMUS^{RT} extends Linux (version 3.10.5) by implementing a set of common scheduling algorithms as plug-in components. Numerous publications [**?**, **?**, **?**, **?**, **?**, **?**, **?**] have shown LITMUS^{RT} to be a very valuable testbed for experimentally evaluating real-time scheduling algorithms by considering real system overheads. Since big.LITTLE supports Linux (after setting a scheduler patch to add the three native big.LITTLE schedulers as discussed earlier) and LITMUS^{RT} is developed as a patch to the Linux kernel, we can run LITMUS^{RT} on big.LITTLE for our evaluation purposes.

2.3 Related Work

Real-time scheduling in heterogeneous systems. The real-time scheduling of sporadic task systems on a homogeneous multiprocessors has received an enormous amount of attention [**?**, **?**]. On a heterogeneous multiprocessor that contains processor cores with varying speeds, several works [**?**] have been conducted, which showed that partitioned scheduling-based approaches are able to yield HRT and SRT schedulability tests with various degrees of capacity loss. In a recent work [**?**], PI Liu's research group developed the first optimality result for this problem, which presents a global scheduling algorithm and its associated schedulability test that guarantees SRT correctness with no capacity loss. Another recent work [**?**] developed a new global scheduling algorithm and the associated schedulability test that guarantees HRT correctness with no capacity loss.

Exploring timing-energy tradeoffs in heterogeneous systems. For systems without timing requirements, a number of approaches have been proposed to explore the performance-energy tradeoffs (e.g., throughput versus energy) [**?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**]. For the specific big.LITTLE platform, [**?**] proposed a new fair scheduler which is aware of the asymmetric platform and implemented in Linux on top of the big.LITTLE scheduler. In [**?**], a price theory-based dynamic power management framework is designed to save energy on ARM's big.LITTLE. [**?**] proposed utilization-aware load balancing mechanism target for energy efficiency of big.LITTLE processor. [**?**] developed an performance-aware Governor for mobile games to reduce energy consumption by leveraging power management tools such as DVFS.

For systems with timing requirements, a rather few works have been conducted exploring the timing-energy tradeoffs in a heterogeneous computing system. [**?**, **?**] represent the only works targeting at this topic. Besides yielding rather pessimistic schedulability tests (possibly due to the mere consideration of partitioned scheduling), these works apply a similar design philosophy as those designed for homogeneous platforms, where only timing-related parameters are considered when making scheduling decisions. Energy saving techniques such as DVFS are applied afterwards to the maximum degree to save energy while not causing timing violations. As will be discussed next in Section 3.1.1, we believe that fundamental energy efficiency due to using a heterogeneous platform can only be achieved if a scheduler considers the *heterogeneous energy characteristics of tasks exhibited on different processors* when making scheduling decisions.

DVFS approach overview. DVFS has been proven to be an effective technique that reduces energy consumption through dynamically adjusting the supply voltage and operating frequency of processors. DVFS has also been widely studied in real-time and embedded systems (see [**?**] for a detailed survey on this topic). DVFS-based real-time scheduling algorithms can be generally divided into four categories: (*i*) *static* DVFS, where DVFS are statically applied offline after all scheduling decisions (based on tasks' worst-case execution times) are made such that the frequency on every processor is minimized while still being able to guarantee timing correctness. (*ii*) *dynamic intra-task* DVFS, where variations (often called slack) in jobs' actual execution time and their pre-defined worst-case execution times are exploited. Slacks are dynamically reallocated inside the same task to further lower the frequency at runtime. (*iii*) *dynamic inter-task* DVFS, which is similar to the dynamic intra-task DVFS approach. The major difference is that this approach redistribute dynamic slack either among all ready tasks or to immediate priority single ready task only at task boundaries. As will be seen in Section 3.1, we will explore all these three DVFS methods in our design space.

3 Detailed Research Plan

3.1 Step 1: Identifying Energy-Efficient Configurations that Guarantee Timing

We plan to develop real-time scheduling algorithms that are fundamentally energy efficient and the associated schedulability tests that can be applied on heterogeneous platforms. Before describing our proposed approach, we first argue that the existing schedulers that are designed to guarantee timing correctness while minimizing energy consumption on a heterogeneous platform may not be energy-efficient.

3.1.1 Motivation: Why Existing Energy-Efficient Real-Time Schedulers may be Energy-Inefficient?

Existing energy-efficient real-time schedulers on heterogeneous platforms (e.g., [**?**,**?**]) apply a similar design philosophy as those designed for homogeneous platforms. That is, tasks are first prioritized and scheduled to different processors according to their timing parameters in order to ensure timing correctness, then energy saving techniques such as DVFS or dynamic power management are applied to the maximum degree

to save energy while not causing timing violations. While it is generally more energy-efficient to use heterogeneous multiple processors instead of identical ones, we argue that fundamental energy efficiency due to using a heterogeneous platform can only be achieved if a scheduler considers the *heterogeneous energy characteristics of tasks exhibited on different processors* when performing task prioritization and processor selection, which are the two core operations of scheduling.

As discussed earlier in Section 2.1, a key observation seen in Table **??** is that for certain workloads, the energy consumed on "big" processors with the lowest frequency is actually much larger than the energy consumed on "LITTLE" processors with the highest frequency. This indicates that if a task is first scheduled on a "big" processor to achieve the best timing performance and then executed with a lowed frequency to save energy (i.e., the current practice), the resulting energy consumption may ironically be much larger than scheduling the task on a "LITTLE" processor according to its heterogeneous energy characteristics. However, this observation does not simply suggest that all workloads shall be first scheduled to "little" processors to be energy efficient because (*i*) doing so may cause timing violations due to the overloaded "LITTLE" processors, and (*ii*) there also exist tasks whose heterogeneity in terms of the energy characteristics on different processors is rather small. Thus, we argue that the computing capacity provided by different processors to execute a particular task is required to be properly selected to save energy as the heterogeneity of processors arises, which can be further motivated by the following example.

Example. An example job set instance is comprised of the following three jobs:

- J_1 = (0, 10, 100, 120); J_2 = (0, 5, 15, 115); J_3 = (0, 5, 15, 35),

Job	Execution time (*ms*)		Energy Consumption (*mJ*)		Frequency (*GHz*)		Deadline
	big	LITTLE	big	LITTLE	big	LITTLE	
Job 3	5	15	100	7.5	3	1.5	35
Job 2	5	15	100	7.5	3	1.5	115
Job 1	10	100	200	50	3	1.5	120

Figure 3: An example job set run on big.LITTLE.

where $J = (r, c^{big}, c^{LITTLE}, d)$ denotes a real-time job arriving at time r and needing to execute for c^{big} time units on "big" processors or c^{LITTLE} time units on "LITTLE" processors by a deadline at time d. The platform associated parameters are shown in Figure 4. Suppose that we wish to schedule this instance on a big.LITTLE platform in which processor speeds will remain static during run-time (note that we can apply DVFS on different processors later to further save energy). We consider three different schedulers. The first scheduler is a classical real-time scheduler namely global-early-deadline-first (GEDF), with the resulting schedule shown in Figure 5a. Under GEDF, the job with the earliest deadline will be scheduled on an earliest available processor. The GEDF schedule completes all three jobs by their deadlines, but yielding a total energy consumption of 307.5 mJ. Figure 5b shows a modified work-conserving schedule where job prioritization and core selection are made by considering both timing- and energy-related parameters. Since J_1 exhibit the highest degree of heterogeneity in terms of the difference between the energy consumed on two types of processors, we assign the highest priority to J_1, the second highest priority to J_3, and the lowest priority to J_2 (which has the longest deadline). Doing so allows J_1 to be allocated to the "LITTLE" processor and the resulting energy consumption can be reduced to 250 mJ. Notice that J_2 is assigned to the "big" processor since this processor idles first at time 5. Under the third scheduler, as shown in Fig. 5c, the energy consumption can be further reduced to 157.5 mJ (which is optimal for this example). This is by employing a non-work-conserving scheduler, which enforces J_2 to wait to be scheduled on its favorite processor in terms of energy consumption, even if the unfavorable "big" processor becomes idle at time 5.

The above example highlights the motivation behind our design of energy-efficient real-time schedulers on heterogeneous platforms, that is, fundamental energy efficiency on heterogeneous platforms can be achieved when a scheduler determines both job prioritization and processor selection by considering tasks' heterogeneous energy characteristics on different processors as "first-class citizen" parameters.

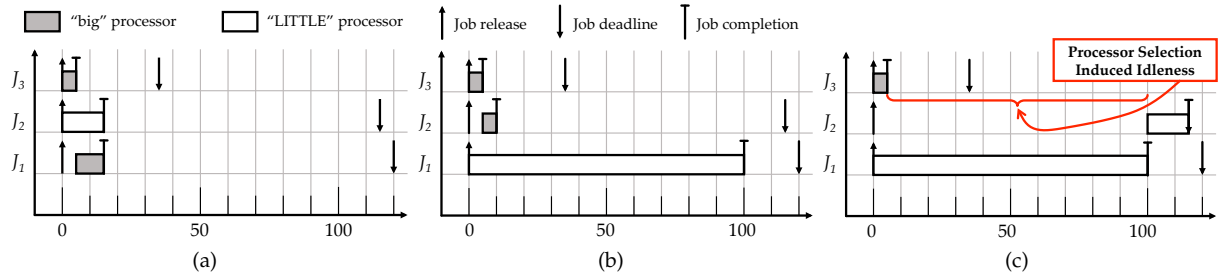

Figure 4: (a) G-EDF yields 307.5 mJ; (b) A work-conserving energy-aware scheme yields 250 mJ; (c) A non-work-conserving energy-aware scheme yields 157.5 mJ.

Energy-oriented task prioritization and processor selection schemes. We propose to investigate a number of potential task prioritization and processor selection schemes that make decisions by considering tasks' heterogeneous energy characteristics. Note that any of these schemes will be viewed as a core component of each different scheduling methodology (e.g., partitioned, clustered, or global scheduling, as will be discussed later in Section 3.1.2). We will investigate the following two categories of such schemes:

1. **Energy-efficient schemes (EES):** We first apply classical real-time task prioritization schemes (e.g., EDF), and then schedule each task to the processor that yields the minimum energy consumption while guaranteeing timing correctness. The tradeoff herein is that applying classical timing-oriented task prioritization may yield tighter schedulability analysis; while applying energy-oriented processor selection may yield better energy efficiency.

2. **Energy-aggressive schemes (EAS):** For more aggressive energy savings, we will apply a set of energy-oriented task prioritization rules, and then schedule each task to the processor that yields the minimum energy consumption while guaranteeing timing correctness. We propose to consider the following energy-oriented task prioritization rules: (*i*) largest-energy-heterogeneity-ratio-first: tasks with the higher energy heterogeneity ratio will be assigned higher priorities, where a task's energy heterogeneity ratio is defined to be the energy consumed for running it on a "big" processor divided by the energy consumed for running it on a "LITTLE" processor. The intuition is that more energy would be wasted if a task with a higher energy heterogeneity ratio is scheduled to an unfavorite type of processor in terms of energy. (*ii*) largest-energy-difference-first: tasks with larger energy difference values will be assigned higher priorities, where a task's energy difference is defined to be the difference between the energy consumed for running it on a "big" processor and a "LITTLE" processor. (*iii*) largest-energy-timing-ratio-first: tasks with higher energy-timing ratio will be assigned higher priorities, where a task's energy-timing ratio is defined to be the minimum value between the energy consumed for running it on either a "big" or a "LITTLE" processor divided by the task's relative deadline.

We will discuss next how to incorporate the above energy-oriented task prioritization and processor selection schemes into the large solution space.

3.1.2 Identifying Energy-Efficient Configurations that Guarantee Timing

We plan to devise energy-oriented real-time resource allocation methods and the corresponding HRT and SRT schedulability tests that can be applied on heterogeneous computing platforms. These tests will account for runtime system overhead incurred under different methods.

DVFS-based approaches. The basic resource allocation methods that we believe will be the most fruitful is to apply DVFS combined with energy-oriented scheduling to reduce energy consumption. This is motivated by our experimental data showing that for many applications, scaling down the frequency yields considerably more energy saving than race-to-idle (as discussed in Section ?where?). The solution space for defining such resource allocation methods is vast. As noted earlier, processor capacity can be allocated on a partitioned, clustered, or global basis. For each allocation methodology, the three DVFS methods can be applied (as discussed in Section 2.3).

Fig. 4 depicts the various configurations that result under dynamic-priority or fixed-priority EES and EAS scheduling algorithms (as defined in Section 3.1.1) and indicates DVFS methods that can be used to support each option. For example, when processors are globally managed, capacity can be allocated using dynamic-priority EES with processing frequency controlled by the static DVFS method. Each configuration may have its benefits. For example, as noted earlier, partitioned approaches generally perform well in HRT domains, while global approaches typically perform better in SRT domains. Also note that we do not intend to investigate static-priority approaches under global scheduling, as previous works [**?**] have shown that such approaches suffer from significant pessimism in terms of capacity loss on multiprocessor platforms.

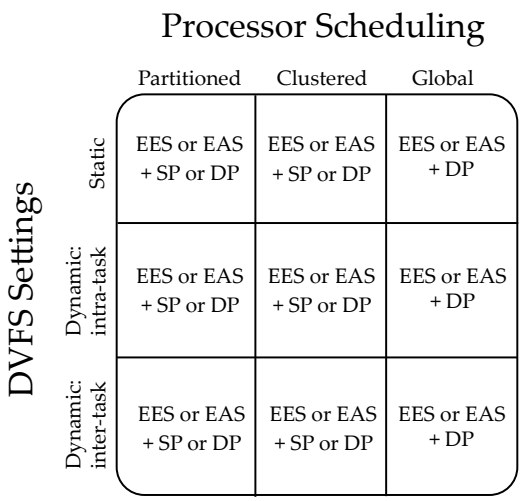

Figure 5: Processor scheduling and DVFS settings under either fixed-priority (FP) or dynamic-priority (DP) EES and EAS scheduling schemes.

While Figure 6 implies that there are only nine different scheduler configurations, each configuration can be further specialized. For instance, a single "big" processor does not necessarily need to be bound to a single "LITTLE" processor under clustered scheduling. Instead, a single "big" processor could be clustered along with several "LITTLE" processors. However, this minor change yields new challenges when we assign tasks to processors, since the symmetric processor structure (i.e., one "big" and one "LITTLE" processor) now changes to an asymmetric one (i.e., one "big" and multiple "LITTLE" processors). This change forces the consideration of other alternatives. We have identified over twenty possible big/LITTLE scheduling/organizational combinations for dynamic-priority systems alone. The situation is similar for static-priority systems. Matters are further complicated by other factors. For example, HRT and SRT systems are subject to different schedulability analysis. Finally, to be practicably applicable, schedulability tests must be augmented to account for system overheads such as OS activities and cache-related preemption and migration costs. The proper accounting of such overheads is extraordinarily tedious as we experienced in our previous works [**?**]. The major goal in Part 1 of this project is to sift through all of these analysis alternatives, devise the tightest schedulability analysis that is practicably possible for each alternative. The most energy-efficient ones among these alternatives will also be identified through implementation and extensive evaluation (as detailed in Sections **??** and **??**).

Multi-frequency scheduling heuristics.

Non-DVFS-based energy saving approaches.

3.2 Step 2: ...

In step 2, we will develop new algorithms and techniques that address energy consumption for mixed critical real-time systems such as automotive computing. In both cases, we plan to leverage two key insights about heterogeneous architectures like those presented in Figure **??**:

- Their energy efficiency increases as they slow down.
- They have tremendous additional compute capacity that can only be used for short amounts of time.

Specifically, we will develop two techniques, each based off one of these insights, and each providing a different capability:

- **Pace-to-sprint**: The goal of this technique is to reduce energy consumption for hard real-time workloads. Where traditional techniques allocate for worst case and then idle if work is completed early (i.e., race-to-idle), pace-to-sprint will develop scheduling techniques to spend as much time as possible in slower, more energy-efficient states.
- **Energy Guarantees**: The goal of this technique will be to guarantee energy consumption for mixed critical workloads; i.e., workloads that contain jobs with both hard and soft real-time requirements. In this work, we will use the additional, unsustainable compute capacity to process hard real-time tasks and ensure that their deadlines are respected (building off our work in step 1). In addition, we will develop algorithms and interfaces for admitting additional soft real-time jobs that will only be scheduled if they do not violate energy constraints. Thus, this technique will allow us to bound both timeliness and energy consumption, possibly at the cost of not executing some non-critical tasks.

3.2.1 Pace-to-Sprint

We will develop pace-to-sprint techniques to minimize energy consumption for workloads with hard real-time requirements. The key insight here is that allocating for worst case execution time wastes energy, yet many application's have average-case behavior significantly less demanding than their worst case behavior. On our target systems, allocating for worst case is not practical, as such an allocation would violate thermal constraints (see Figure **??**). Where traditional approaches would start with worst case allocation and transition to an idle state when they finish, in this approach we will begin execution in an energy-efficient state. If the current job is not a worst-case job, then it will finish in this energy-efficient mode. As it is executing, if its time in the energy efficient state exceeds some threshold, then we will transition to a thermally unsustainable, yet fast resource allocation to ensure timing deadlines are met.

More formally, assume a set of jobs starts at time 0 and must complete W units of work by time t. The system supports C configurations $c \in \{0, \ldots, C-1\}$, each of which has a computation rate s_c and a power consumption of p_c. These configurations represent settings for all configurable components in the system (e.g., DRAM speed, processor speed, and number of active cores). Here, we assume these values are known based on our work in Step 1. The system has a unique *idle* configuration $c = 0$ with $s_0 = 0$ and $p_0 = p_{idle}$, where p_{idle} is a system dependent (and workload independent) value representing the system's power consumption when not executing a computation. We assume configuration $C-1$ represents making all resources available to a workload. The goal is to assign a time $0 \leq t_c \leq t$ to each machine configuration. Note that configuration c will contribute $t_c \cdot s_c$ work at a cost of $p_c \cdot t_c$ energy. We have used similar formulations in our prior work [**?**,**?**,**?**]. Thus, the problem of minimizing energy while completing the work

can be expressed as:

$$\text{minimize} \sum_{c} t_c \cdot p_c \tag{1}$$

subject to

$$\sum_{c} t_c \cdot s_c = W \tag{2}$$

$$\sum_{c} t_c = t \tag{3}$$

$$0 \leq t_c \leq t, \text{ for } c = 0, \ldots, C-1 \tag{4}$$

Equations 1–4 assign values for all t_c to minimize total energy consumption (Equation 1) subject to the constraints that the total work accomplished is equal to the required work W (Equation 2) and the deadline is met (Equation 3). The final constraint ensures that the time spent in any configuration is non-negative.

The traditional, race-to-idle approach represents a heuristic solution to this optimization which uses only two configurations: $C-1$, where all resources are available and 0 which idles the machine. Our proposed approach is to develop new heuristics that beat race-to-idle. In the past we have developed heuristics that are appropriate to soft real-time schedules [**?**, **?**]. In the proposed work, we will extend these heuristics and develop new algorithms to ensure hard real-time deadlines.

Specifically, we will look for solutions that use the configuration e as much as possible, where e represents the most energy efficient configuration; i.e., $\forall c \ r_e/p_e \geq r_c/p_c$. We will determine how long the system can stay in that configuration before it risks violating timing and we will then transition to a faster (and less energy efficient configuration) to ensure that timing constraints are not violated. We will examine algorithms for determining when to switch both statically and dynamically. Two insights drive this work: 1) when jobs finish faster than worst case, we will not have wasted energy, because we will have been operating in the most energy efficient configuration and 2) when jobs are in danger of violating timing we can switch to a computationally intensive configuration that will ensure safety.

3.2.2 Energy Guarantees for Mixed Critical Workloads

We will develop techniques to provide energy consumption guarantees for systems with both hard and soft real-time requirements. The basic intuition in this step will be to divide the tasks into two groups. For hard real-time tasks, we will guarantee deadlines are met. For soft real-time tasks we will apply a best-effort approach.

We will realize this approach by dividing scheduling periods into two fixed-sized quanta. In the first, we will schedule hard real-time tasks which may require high power consumption. In the second quanta, we will determine the remaining average power consumption that will meet the energy goal given the remaining time. During this second time quantum, we will set the system to a resource configuration that is guaranteed not to exceed the required power and schedule as many soft real-time tasks as possible.

More formally, we will meet an energy budget E for a scheduling period by dividing this period into two quanta: t_h for scheduling hard real-time tasks and t_s for scheduling soft real-time tasks. Using results from step 1, we will configure the system to use resources that are guaranteed to schedule the hard real-time tasks and measure the energy consumption E_h. We then know the remaining energy available for the soft real-time tasks is $E_s = E - E_h$. Since the time allocated for soft real-time tasks is fixed, we simply determine the power consumption that respects the energy goal $P_s = E_s/t_s$. We will then configure the system to a resource usage that is guaranteed not to exceed this power consumption.

We believe this ability to guarantee energy consumption for an embedded real-time system is a new capability not previously available.

3.2.3 Evaluation

We will evaluate our techniques developed in step 2 using two metrics:

- **Energy Consumption:** We will instrument boards with commercially available power and energy sensors. We expect that pace-to-sprint should exhibit lower energy consumption than existing techniques. While specific savings will vary with workload, our preliminary results (Figure **??**) show that energy reductions of up to $2\times$ are possible compared to the commonly used race-to-idle heuristic. When evaluating our ability to provide energy guarantees, we will measure the number of violations that occur for a workload, where a violation means that a scheduling period exceeded its budgeted energy.
- **Schedulability:** We will evaluate pace-to-sprint by ensuring that all hard deadlines are indeed met. In addition, we will evaluate our ability to provide energy guarantees by determining how many of the soft real-time tasks we can successfully execute. Clearly, we could execute none and simply idle the processor, but our goal is to execute as many as possible without violating the energy constraints. We will compare our approach to a static scheme that does not dynamically monitor energy and adjust resource usage. We hope to show that our approach greatly increases schedulability, allowing more work to be done on the same platform.

3.3 Step 3: Implementation and Overhead-Conscious Schedulability Studies

After conducting the research summarized in Secs. **??** and **??**, we will have a set of energy-efficient real-time resource allocation methods on heterogeneous computing platforms. This section discusses our implementation plan and evaluation work needed to assess which methods are preferable from the perspectives of overhead-conscious schedulability, runtime response times, and energy consumption.

Implementation plans. We will implement our proposed resource allocation methods in real systems to evaluate their performance in practice. The evaluation platform will be LITMUS^{RT} (i.e., the OS platform) [**?**]. As discussed earlier, our goal of using LITMUS^{RT} is to have a useful experimental platform for evaluating overhead-conscious schedulability of our proposed resource allocation methods. As discussed earlier, ARM's big.LITTLE supports Linux. The two open-source scheduling modules, i.e., kernel switcher and global task scheduling, can be patched to the Linux kernel. Thus, we will patch LITMUS^{RT} to the Linux kernel running on big.LITTLE. Each of our proposed methods will be implemented in LITMUS^{RT} as a plugin component. (This implementation method allows users to switch between different scheduling policies at runtime.) An interesting set of experiments would be to compare the timing and energy performance between our proposed methods and the native kernel switcher and global task scheduling methods on big.LITTLE. To measure energy consumption, we will use the two currently available tools. The first tool is the ARM DS-5 Streamline tool set [**?**], which is a performance analyzer that brings together system performance metrics, software tracing, statistical profiling and power measurement through user interface. The second tool is to the INA231 Current/Power monitor chip produced by TI [**?**, **?**, **?**]. The power data is obtained through measuring both the shunt voltage drops and bus supply voltage. The ODROID-XU3 board [**?**] powered by ARM big.LITTLE technology already integrates four INA231 monitors that can measure the energy data of the "big" cluster, "LITTLE" cluster, GPU, and memory [**?**]. We thus plan to buy ODROID-XU3 boards for implementation and evaluation purposes.

We also intend to consider heterogeneous multicore platforms from other manufactures as well as those adopting similar heterogeneous system architectures. For instance, we plan to consider another two heterogeneous multicore architectures that are similar to big.LITTLE, i.e., NVIDIA's Tegra 3 SoC [**?**] and Freescale's Vybrid SoC [**?**]. Tegra 3 adopts a variable symmetric multiprocessing technology, which aims to minimize the active standby state power consumption but still deliver required performance. Tegra 3 includes an

ARM's quard-core Cortex A9 processor for high performance and one companion CPU which is energy-efficient but delivers relatively low performance. The platform can save energy by scheduling less resource-demanding task to the companion CPU. Compared to big.LITTLE, Tegra 3 may yield a milder solution for energy saving and a smaller range for energy-performance tradeoff. Freescale's Vybrid SoC also applied asymmetric architecture, but for the purpose of separating user-interface workloads and real-time workloads. The platform consists of one ARM Cortex-A5 processor running Linux and one ARM Cortex-M4 core running real-time operating systems like freescale's MQX. A multicore communication protocol is designed to coordinate the interactions between the two processors. Compared to big.LITTLE architecture, the Vybrid architecture mainly deals with the conflicts between real-time and non-real-time workloads other than the tradeoff between energy and performance.

Overhead-conscious schedulability and energy studies. A research focus herein is to sift through all resource allocation options identified in Sections 3.1 and 3.2 through evaluating the performance with respect to schedulability and energy efficiency when practical scheduler-incurred overheads are considered. The overhead-conscious evaluation is important as a theoretically-sound scheduler may not perform well in practice. For example, although global scheduling approaches generally yield better theoretical schedulability tests with less capacity loss than partitioned approaches, they may perform worse than partitioned approaches due to the more frequent task preemption and migrations.

To evaluate the performance with respect to overhead-conscious schedulability and energy efficiency, we will follow a mature experimental process applied in many previous work [**?**, **?**, **?**, **?**, **?**]. Actual system overheads will be gathered and considered in this process. Scheduling actions may incur various types of overheads, including costs due to scheduling decisions, context switch operations, task queue management, interrupt servicing, and preemption and migration. By implementing our proposed energy efficient heterogeneity-aware scheduling algorithms in LITMUS^{RT}, we will be able to gather these overheads under different approaches using millions of randomly generated task systems and benchmarks. After collecting overheads, we will factor them into task parameters and check the schedulability of all generated task systems. The outcome of this experimental process is a collection of schedulability graphs showing how different scheduling approaches perform and compare in light of real system overheads. Besides schedulability, energy consumption, response time bounds (analytically derived under our derived schedulability tests) and run-time response times are also important. We will also evaluate the proposed methods in terms of energy consumption, response time bounds, and maximum and average run-time response times.

3.4 Step 4: Conduct Case Studies

In order to further access whether our proposed methods can be applied in practical systems, we intend to conduct case-study evaluations using actual workloads from an advanced automotive system. Due to space constraints, we describe one proposed case study in (some) detail, and briefly note others.

Automotive workloads. As mentioned earlier, a strong motivation behind the proposed project is to leverage heterogeneous computing platforms to support a set of advanced driving-assisted tools in an power-constrained automotive system. In prior work on such systems, timing predictability and resource provisioning issues have been the major focus while the critical energy efficiency issues are ignored. Due to lacking real-time energy-heterogeneity-aware resource allocation schemes, significant energy lost may occur in order to guarantee timing correctness. Even worse, without analytically verifiable energy consumption and timing correctness guarantees, such driving-assisted features could not easily be legally certifiable, and safety thus becomes a major issue.

We plan to evaluate a set of common driving-assisted workloads seen in automotive systems using one or more preferable scheduling plugins identified in Sec. **??**. Real-time object recognition represents one of such workloads, which serves as a major task in implementing the automatic collision avoidance and the traffic sign recognition functionality. Fig. **??** shows the workload breakdown and parameters of

a typical object recognition task using a popular recognition algorithm [**?**]. Each camera sensor installed on a vehicle continuously captures images at a certain frequency. Then each image will be compressed to reduce irrelevance and redundancy of the image data and processed or stored in an efficient manner. Each compressed image will then be processed, where computation-intensive object recognition and analysis algorithms will be executed to recognize the desired objects in each image. Performing such tasks in an automotive system is challenging because different stages often require different amount of computational capacity and energy. For example, ?image decoding v.s. image recognition?

We plan to use our proposed energy efficient real-time scheduling methods to support such tasks and evaluate the performance w.r.t. four important metrics: *(i)* timing performance, i.e., schedulability and max/average response times, *(ii)* energy performance, i.e., max/average power and energy consumption, *(iii*) overall system throughput, i.e., number of object recognition tasks that can be supported in parallel, and *(iii)* recognition accuracy. We plan to adopt a number of existing image compression and object recognition algorithms with different runtime complexity to investigate the trade-offs among energy consumption, response time performance, and recognition accuracy. Moreover, given specific timing requirements (enforced by safety requirements) such as response time bounds, our measured system performance may be useful in determining minimum system provisioning requirements. In addition to the real-time object recognition task, there exist many other computer vision-based such as autonomous navigation, automatic lane following, intelligent cruise control. We will study a mixed set of such workloads in the case study. Among these workloads seen in an advanced automotive system, video and image processing-based tasks are the most common ones, as briefly noted below.

Real-time video decoding. Emerging demands for real-time video decoding brings the current hardware resources to its limits. We would like to investigate whether our proposed scheduling methods can efficiently utilize heterogeneous core capacity and thus support the concurrent processing of more videos with guaranteed timing correctness while minimizing energy consumption. As discussed in Section [**?**], for many video decoding workloads, executing them on big cores with the lowest DVFS setting may still yields considerably higher energy consumption than on LITTLE cores with the highest DVFS setting. We will also investigate the tradeoffs among video quality, responsiveness, and energy consumption. An experiment we intend to conduct is to compare our methods against the native schedulers on big.LITTLE, which could answer an interesting question: whether analysis-based energy-efficient scheduling techniques are superior to widely used general purpose OS schedulers in terms of energy and timing performance?

Real-time image analysis. Image processing is also a candidate that may benefit greatly from using heterogeneous computing platforms. One specific application we plan to evaluate is the high dynamic range imaging (HDRI), which is used to capture a greater dynamic range between the lightest and darkest areas of an image. HDRI can be naturally modeled as a set of recurrent real-time tasks that exhibit various levels of computational intensiveness and energy characteristics (refer to [**?**] for a detailed description of HDRI). Our work may include the development of systems that can process a large-scale set of images in a timing-correct fashion while dramatically minimizing energy consumption by consider energy heterogeneity inherent to both workloads and processors.

3.5 Timeline

Note that throughout the project, we intend to work on algorithmic and schedulability-analysis issues, implementation, and evaluations in parallel.

- **Year 1:** During Year 1, we will focus on identifying energy-efficient configurations that yields the tightest schedulability tests, as summarized in Sec. **??**. In the second half of this year, we will begin developing more aggressive energy-saving techniques, as summarized in Sec. **??**
- **Year 2:** During Year 2, we expect to complete all research problems related to the design and analysis

of resource allocation methods. We will then begin working on conducting overhead-aware energy and schedulability studies, as described in Sec. 3.3.

- **Year 3:** During Year 3, we intend to focus our attention on experimental and case-study evaluations (and may improve implementation details based on evaluation), as described in Sec. 3.4.

We expect that our research will result in approximately five to eight conference and journal papers per year (based upon our past publication rates).

4 Broader Impact of the Proposed Work

Scientific and societal impact of research: The results from this project will be of interest to industry as well as US government agencies. Developed techniques will synergistically improve safety critical systems by 1) allowing them to incorporate more functionality in the same platform (reducing costs and adding new capabilities) or 2) greatly reducing the energy consumption of current systems (allowing extended time between charging and allowing harvested energy systems to perform more when energy resources are scarce. Increasing our ability to build safey-critical, energy-efficient systems will benefit medical, defense, transportation, and automotive applications.

Dissemination of of research and collaboration: We will present tutorials and workshops at conferences, and seek to publish our outcomes in premier embedded, real-time, and operating system journals and conferences, e.g., SOSP, ASPLOS, RTSS. We will make all the real-time benchmarks, simulation framework/models, software, and tools publicly available, so that others can repeat experiments, and adopt and build on the ideas. PI Liu and his collaborators have already ¡CONG: list any publicly available software or industry impacts here¿. PI Hoffmann and his collaborators have released a number of software packages as open source. This includes the Application Heartbeats API for monitoring software performance and health [**?**, **?**], the POET toolkit for portable energy efficiency [**?**, **?**], and LEO for learning computer system performance and energy tradeoffs [**?**, **?**].

The PIs also have a strong track record of transitioning research and technology developed in academia into industry. PI Liu ¡CONG: fill in any industry collaborations¿ PI Hoffmann was part of the team that transferred MIT's Raw Architecture to industry through the founding of the Tilera Corporation [**?**, **?**]. At MIT Lincoln Laboratory, PI Hoffmann developed the foundations for the VSIPL++ parallel image processing standard which is now used worldwide [**?**]. In collaboration with Mozilla Reseach, PI Hoffmann's students have incorporated their dynamic instrumentation and energy monitoring framework into the open source release of Servo, Mozilla's open-source mobile web browser [].

Outreach and Diversity: We will seek to actively recruit under-represented minority and women graduate and undergraduate students into our research groups. PI Hoffmann is actively involved in encouraging under-represented minorities to participate in research through the Leadership Alliance at the University of Chicago. Currently 33% of his PhD students are from traditionally under-represented backgrounds. The Leadership Alliance is a national academic consortium of 33 colleges and universities focused on training and mentoring a diverse group of undergraduate students into competitive graduate programs and professional research careers. The University of Chicago is among the 22 Alliance institutions that offer the Summer Research Early Identification Program (SR-EIP). In summer 2015, PI Hoffmann mentored two students through the Leadership Alliance program [**?**].

Curricular Development Activities: PI Liu teaches... ¡CONG: fill in¿

PI Hoffmann teaches a graduate class on power and energy-aware computing. This course covers power and energy issues at the operating system, architecture, and microarchitectural levels. The proposed research will both help him incorporate new lessons on energy management into the class and it will provide research problems for students' course projects. This class is currently being taught for the second time.

The first iteration of the class produced 2 projects that are in submission for publication and 1 project that became the student's Master's thesis.

5 Results From Previous and Current NSF Support

¡CONG: Your stuff goes here.¿
XPS: FULL: CCA: Collaborative Research: CASH: Cost-aware Adaptation of Software and Hardware (CCF 1439156, PI: Hoffmann, 09/2014-08/2018, $300,000)
EAGER: HAWKEYE: A Cross-Layer Resilient Architecture to Tradeoff Program Accuracy and Resilience Overheads ¡HANK: Fill in details¿
CNS: SMALL: BreezeFS
II-New: RIVER: A Research Infrastructure to Explore Volatility, Energy-Efficiency, and Resilience (CNS 1405959, PI: Chien, Co-PIs: Foster, Gunawi, Hoffmann, Scott, 07/2014-07/2017, $997,432)

All projects advance the study of adaptive computing systems. CASH studies the modeling and allocation of fine-grain hardware structures to produce cost savings in infrastructure as a service (IaaS) clouds. HAWKEYE studies the ability to dynamically trade program accuracy for error detection overhead. BreezeFS studies adaptive management of multi-store systems for clouds. Among other things, RIVER's infrastructure supports study of large-scale adaptation to meet power and performance goals in distributed computing systems. Both projects began less than 6 months ago.
Intellectual Merit: In the CASH project we have developed novel hardware structures for performance monitoring as well as novel combinations of machine learning and control theory for allocating fine-grain hardware resources and minimizing costs. Both HAWKEYE and BreezeFS are in their early stages, but both will study application of adaptive tradeoff management to new problems (resilience and storage management, respectivel).
Broader Impact: The CASH project has already resulted in four pulbished papers from the University of Chicago [**?**, **?**, **?**, **?**] and one submitted publication whose outcome is still pending as of this writing.

Cong Liu

Department of Computer Science, University of Texas at Dallas, Richardson, TX 75081
http://www.utdallas.edu/~cong/

Professional Preparation

Wuhan Univ. of Technology	Computer Science	B.S. with honor, 2005
Auburn Univ.	Computer Science	M.S., 2008
Univ. of North Carolina at Chapel Hill	Computer Science	Ph.D., 2013

Appointments

Assistant Professor, Department of Computer Science, University of Texas at Dallas, 9/13-present.

Products

(i) Most Relevant Publications

- Husheng Zhou and Cong Liu, "*Task Mapping in Heterogeneous Embedded Systems for Fast Completion Time*", Proceedings of the 14th ACM International Conference on Embedded Software (EMSOFT), 2014.
- Guangmo Tong and Cong Liu, "*Supporting Soft Real-Time Sporadic Task Systems on Heterogeneous Multiprocessors with No Uilitzation Loss*", IEEE Transactions on Parallel and Distributed Systems (TPDS), 2015.
- Husheng Zhou, Guangmo Tong, and Cong Liu, "*GPES: A Preemptive Execution System for GPGPU Computing*", Proceedings of the 21st IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2015.
- Cong Liu, Jian Li, Wei Huang, Juan Rubio, and Evan Speight, "*Power-Efficient Time-Sensitive Mapping in CPU/GPU Heterogeneous Systems*," Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques (PACT), pp. 23-32, 2012.
- Guangmo Tong and Cong Liu, "*Supporting Read/Write Applications in Embedded Real-time Systems via Suspension-aware Analysis*", Proceedings of the 14th ACM International Conference on Embedded Software (EMSOFT), 2014.

(ii) Additional Products

- Jianjia Chen, Wenhung Huang, and Cong Liu, "*K2U: A General Framework from k-Point Effective Schedulabiliy Analysis to Utilization-based Tests*", Proceedings of the 36th IEEE Real-Time Systems Symposium (RTSS), 2015
- Jianjia Chen and Cong Liu, "*Fixed-Relative-Deadline Scheduling of Hard Real-Time Tasks with Self-Suspensions*", Proceedings of the 35th IEEE Real-Time Systems Symposium (RTSS), 2014.
- Cong Liu and Jianjia Chen, "*Bursty-Interference Analysis Techniques for Analyzing Complex Real-Time Task Models*", Proceedings of the 35th IEEE Real-Time Systems Symposium (RTSS), 2014.
- Cong Liu and James Anderson, "An O(m) Analysis Technique for Supporting Real-Time Self-Suspending Task Systems," Proceedings of the 33th IEEE Real-Time Systems Symposium (RTSS), pp. 373-382, 2012.
- Cong Liu and James Anderson, "Task Scheduling with Self-Suspensions in Soft Real-Time Multiprocessor Systems," Proceedings of the 30th IEEE Real-Time Systems Symposium (RTSS), pp.425-436, 2009, **Best Student Paper Award.**

Synergistic Activities

Service to the Scientific and Engineering Community

- TPC, IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2015, 2016.
- TPC, IEEE Real-Time Systems Symposium (RTSS), 2014, 2015.
- TPC, Euromicro Conference on Real-Time Systems (ECRTS), 2016.

Collaborators and Other Affiliations

(i) Collaborators

Dr. Evan Speight, IBM Research Austin Lab;
Dr. Jian Li IBM Research Austin Lab;
Dr. Tian He, University of Minnesota, Twin City;
Dr. Jianjia Chen, TU Dortmund (German);
Dr. Yu Gu, Singapore University of Technology and Design (Singapore);
Dr. Wei Gao, University of Tennessee, Knoxville;
Dr. Shinpei Kato, Nagoya University (Japan).

(ii) Graduate and Post-doctoral Advisors

Dr. James Anderson, Professor, Department of Computer Science, University of North Carolina at Chapel Hill.

(iii) Thesis Advisor and Post-graduate Scholar Sponsor

Mr. Guangmo Tong, PhD Student, Computer Science, UTD (since 2014)

Mr. Husheng Zhou, PhD Student, Computer Science, UTD (since 2014)

Ms. Xia Zhang, PhD Student, Computer Science, UTD (since 2014)

Mr. Yuchuan Liu, PhD Student, Computer Science, UTD (since 2015)

Mr. Gbadebo Ayoade, Masters Student, Computer Science, UTD (since 2013)

Facilities, Equipment, and Other Resources

General computing environment. The Department of Computer Science hosts more than 40 research labs with access 24 hours a day, 7 days a week; 2 open labs for graduate students; and a 128-seat open access lab for undergraduate students. These labs consist of over 500 state-of-the-art, high-performance workstations and high-end PCs, all connected via gigabit Ethernet switches with a redundant fiber up-link to provide fast access to the campus network and the Internet. Nine classrooms and one large lecture hall with the latest computer and audio-visual equipment are available. Academic coursework, project, and computing systems are comprised of Linux x86_64, Windows Server, and Solaris 10 executing on a collection of physical servers and virtual server private clouds. The servers are connected via fiber channel and iSCSI to a thin-provisioned 14TB 3PAR mesh-active storage array.

Specific computing facilities. Computer Science Open Access Lab: This lab is accessible to all students currently enrolled in any course in the department of computer science (128 seats). The lab consists of 128 Dell OptiPlex 980 machines with i7-870 Processor (8M Cache, 2.93 GHz), 4GB RAM, 22" (Dell P2210) flat-panel display, Windows 7, 64-bit; Two HP Laserjet 9040 printers are also located in the lab. The lab is open from 8am to 11pm, M-F, and 11pm-7pm Sat/Sun. The Lab is maintained by the CS Tech Staff. A monitor (typically a graduate student hired half-time) is on premises all the time.

Graduate Windows Lab: This 15 seat lab is accessible to all graduate students. Equipment consists of Dell Precision T1650 computers with Intel XEON CPU E3 1225 v2, @3.20GHz, 8GB RAM, 24" flat-panel monitors, Windows 7 Enterprise and an HP Laserjet P4015n printer. The lab is accessible 24/7 with smart card access.

Project Design Lab: This 20 seat lab is accessible to all CS undergraduate/graduate students who take senior design project class. It consists of Dell Optiplex 760 machines with Intel Core2 Duo, E8400@3.00GHz, 2GB RAM, 17"flat-panel monitors, Windows 7 Enterprise. The lab is accessible 24/7 with smart card access.

General Access Systems: The Department also maintains several systems in its server room to which remote access is provided. These include the Network Programming System that are ccessible to all CS students enrolled in network programming and parallel processing courses (45 systems); Students access these systems via SSH; All these 45 systems run Linux CentOS 6.2 x86_64. The Department also maintains two large Linux compute servers (Linux CentOS 6.2 x86_64, Dell PowerEdge R710 w/2 six-core 2.53GHz Intel Xeon, 32GB RAM) and two large Solaris computer servers (Sun Fire V440 w/4x1GHz UltraSPARC-IIIi, 8GB RAM).

Server clouds: The Department also maintains a general purpose server cloud, a VMware vSphere 4 operating on 5 ESX hosts with dual 6-core processor 128GB RAM servers, used for teaching the cloud computing related courses as well as a network security courses server cloud, a VMware vSphere 4 executing on 4 processors of 6-core Opterons, 128GB RAM, and four 4Gb redundant fiber channel SAN connections.

In addition to the above labs and facilities, CS students also have access the UTDesign Studio that is equipped with modern computing facilities and equipment with more than 30,000 square feet of dedicated space, where students and corporate partners can create, innovate, design, build and learn. Further information can be found at http://www.utdallas.edu/utdesign/corporate/studio/. The UTDesign studio is primarily used by undergraduate seniors taking the capstone project course.

A 30′ x 20′ lab is available for Dr. Liu's students. The lab can comfortably accommodate 8 students and the computing equipment. Eight workstations equipped with state-of-the-art NVIDIA GPUs currently in the PI's lab will be available to the project.

Data Management Plan

This data management plan covers all of the data products to be produced in this project. It includes all source code that implements the ecosystem of GPU resource management, any source code added to LITMUS^{RT}, the suite of case-study programs, and scripts used to test and evaluate the implementation of the proposed components, along with the data collected during evaluation efforts.

Source code. All source code will be freely available for direct download from public web servers maintained by the UTD Computer Science Department under an open source GPL license. Additional case-study test programs and scripts will be freely available under the open source BSD license and directly downloadable from public web servers maintained by the UTD Computer Science Department. Our intend in making these data products freely available is that other researchers can verify evaluation results, and relevant research conducted by other interested groups can be facilitated.

Evaluation data. Evaluation data (including raw data and resulting graphs) can be several terabytes in size. It is thus currently not feasible to make this data available to the public through direct download. We will store and back up such data on secured servers maintained by the UT-Dallas Computer Science Department. Interested parties may request copies of this data through making arrangement for private direct download or using the postal system.

Privacy. No personal data will be acquired in this project. There are no privacy concerns for the storage of data.

Long-term storage. All data products produced in this project will also be archived in the UTD computer science computing repository for permanent storage.