Harvesting-Aware Energy Management for Multicore Platforms with Hybrid Energy Storage

Yi Xiang and Sudeep Pasricha
Department of Electrical and Computer Engineering,
Colorado State University, Fort Collins, CO, USA
E-mail: {yix, sudeep}@colostate.edu

ABSTRACT
In this paper, we propose a novel framework for energy and workload management in multi-core embedded systems with solar energy harvesting and a periodic hard real-time task set as the workload. Compared to prior work, our energy management framework possesses several advantages, including (i) a battery-supercapacitor hybrid energy storage module for more efficient system energy management, (ii) a semi-dynamic scheduling heuristic that continuously adapts to run-time harvested power variations without losing the consistency of the periodic task set, and (iii) a coarse-grained core shutdown heuristic for additional energy savings. Experimental studies show that our framework results in a reduction in task miss rate by up to 61% and task miss penalty by up to 65% compared to the best known prior work.

Categories and Subject Descriptors
C.3 [**Computer Systems Organization**]: Special-Purpose and Application-Based Systems -- *Real-time and embedded systems*. B.8.2 [**Performance and Reliability**]: Performance Analysis and Design Aids

General Terms
Algorithms, Design, Performance

Keywords
Energy Harvesting, Supercapacitor, Dynamic Power Management, Dynamic Voltage and Frequency Scaling, Task Scheduling

1. INTRODUCTION
Power and energy constraints have led to significant changes in the design of contemporary computing systems. In the last decade, the concept of thread-level parallelism (TLP) to improve performance within a power budget has seen widespread adoption across various computing platforms, ranging from high-end servers to desktops, as well as embedded devices. Recent years have also witnessed a significant increase in the use of multi-core processors in low-power embedded devices [1]. With advances in parallel programming and power management techniques, embedded devices with multi-core processors and TLP support are outperforming single-core platforms in terms of both performance and energy efficiency [2]. But as core counts rise to cope with increasing application complexity, techniques for workload distribution and energy management are the key to achieving energy savings in emerging multi-core embedded systems.

GLSVLSI'13, May 2–3, 2013, Paris, France.

For some applications, we need energy autonomous devices that utilize ambient energy to perform computations without relying entirely on an external power supply or frequent battery charges. As the most widely available energy source, solar energy harvesting has attracted a lot of attention and is rapidly gaining momentum. [3][4] To fully exploit the capability of energy harvesting systems, a considerable amount of work has explored task scheduling, primarily for embedded systems with real-time task sets. An early work [8] proposed the lazy scheduling algorithm (LSA) that executed tasks as late as possible, reducing deadline miss rates when compared to the EDF algorithm. However, the approach in [8] does not consider DVFS and always executes tasks at full speed. Because a processor's dynamic power is generally a convex function of frequency, operating the processor at a lower frequency often results in higher energy efficiency. Liu et al. [9] proposed the EA-DVFS technique that takes processor DVFS into consideration. EA-DVFS utilized task slack to slow down execution speed, thus achieving more energy savings than LSA especially when total task utilization is low. Later the same authors proposed a more intelligent technique called HA-DVFS [10], which improved energy efficiency mainly by distributing multiple arriving tasks as evenly as possible over time and executing them with more uniform frequency. However, these works focus on uni-processor systems and have not considered execution on multi-core platforms. Recently, a utilization-based technique (UTB) was proposed in [11] to better address periodic task scheduling in energy-harvesting system. UTB takes advantage of predictability provided by periodic task information for better task distribution. Moreover, UTB proposed a simple extension to support multi-core platforms by allocating a subset of tasks to each core and executing the single-core UTB algorithm separately on each core.

All of these prior efforts on harvesting aware task scheduling assume an ideal battery as the energy storage medium limited merely by it capacity, ignoring other factors such as rate capacity, recovery effect, and lifetime in terms of recharge cycles [6]. When applied to real-world platforms, overlooking these factors can result in suboptimal or even false scheduling that diminishes system efficiency, stability and lifespan. For example, rate capacity effect leads to decreasing battery capacity when discharging current increases [12]. Supercapacitors present an interesting alternative to batteries for energy storage. A substantial amount of research on supercapacitors has demonstrated their benefits over electrochemical batteries, including orders of magnitude higher recharge cycles, much less charge overhead, and significantly higher efficiency with high current discharge. However, high capacity supercapacitors are not practical for small-package low-power embedded systems due to their significantly lower energy density and higher leakage overhead than an electrochemical battery, even with the state-of-art supercapacitor technology [14]. Recent work by Ongaro et al. [5] and Mirhoseini et al. [12] has shown that a battery-supercapacitor hybrid system can overcome the limitations of both types of energy storage mediums.

In this paper, *we propose a novel framework based on a semi-dynamic algorithm (HY-SDA) for energy and workload management in multi-core embedded systems with solar energy harvesting and a hybrid battery-supercapacitor energy storage system*. HY-SDA aims to minimize deadline miss rate or penalty of periodic tasks in the presence of variant and insufficient energy harvesting. Compared to prior work, HY-SDA reacts to run-time energy shortages and fluctuations proactively to find significantly greater scope for energy savings, especially in multi-core platforms. At the system level, HY-SDA is triggered at specified time epochs to adjust inter-core task allocation and set a per-core execution strategy based on the energy budget provided by the hybrid energy storage system. Our experimental studies show that HY-SDA outperforms the best known prior work (UTB [11]), achieving superior task drop rate reduction and energy efficiency.

2. PROBLEM FORMULATION

2.1 System Model

Our focus is on the problem of effective workload and energy management of real-time multi-core embedded systems with periodic tasks, powered by solar energy, as shown in Figure 1.

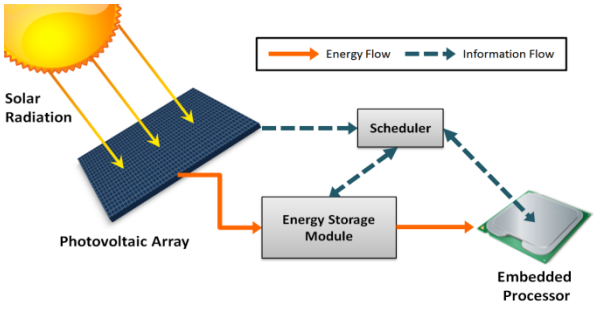

Figure 1. Real-time embedded processing with energy harvesting

2.1.1 Energy Harvesting and Energy Storage Module

A photovoltaic (PV) array is used as a power source for our embedded system, converting ambient solar energy into electric power. Naturally, the amount of harvested power varies over time due to changing environmental conditions, like angle of sunlight incidence, cloud density, temperature, humidity, etc. To cope with the unstable nature of the solar energy source, rechargeable batteries and/or supercapacitors can be used to buffer solar energy collected by photovoltaic cells. In our study, the converted solar power at time t is denoted as $P_H(t)$. The energy E_H charged into energy storage system between time instances t_1 and t_2 is given by

$$E_H(t_1 \sim t_2) = \eta_{harv} \int_{t1}^{t2} P_H(t)\, dt \qquad (1)$$

where η_{harv} is a coefficient between 0 and 1 to represent charging efficiency of the energy storage system. The capacity of the energy storage device is limited. Clearly harvested energy will be wasted if the energy storage device is already fully charged. We assume that task execution must be halted when the remaining energy in the system is less than 10 percent, thus reserving enough storage to maintain system status and ensure graceful shutdown. In this paper, we focus on the scenario where nominal power of the PV array exceeds peak power consumption of the embedded processors. Thus it is necessary to have a storage system that can preserve a considerable amount of energy for later system execution when solar energy is insufficient. A hybrid battery-supercapacitor storage system meets these goals, and is described in Section 3.2.1.

2.1.2 Periodic Real-Time Task Model

We assume a task set of N independent periodic real-time tasks ψ: $\{\tau_1, \ldots, \tau_N\}$, in which each periodic task τ_i has a characteristic triplet (C_i, D_i, T_i), $i \in \{1, \ldots, N\}$. C_i is the maximum number of CPU clock cycles needed to finish a job instance of task τ_i, referred to as worst-case execution cycles (WCEC). The relative deadline of the task, D_i, is the time interval between a job's arrival time and its deadline. A job instance is missed if it is not finished before its deadline. T_i is the period of the task. At the beginning of each period, a new job instance of that task will be dispatched to the system. Like most recent works on periodic task scheduling (e.g., [11]) we assume that D_i equals T_i, with all jobs finishing before the arrival of the next job instance of the same task. In addition, we define an attribute X_i, which is the miss penalty associated with each task. Each time that a task's job misses the deadline, the job will be aborted and the penalty applied to the system. Thus, we can refine the triplet for task τ_i as (C_i, T_i, X_i). The relative importance of a task can be characterized by a *penalty density*, defined as the ratio of the task miss penalty and WCEC (X_i/C_i) [15].

2.1.3 DVFS-Enabled Processor Model

We consider an embedded system with homogeneous multi-core processors. The processors possess DVFS capability at the core level and have support for task preemption. Each core has M discrete voltage and frequency levels: φ: $\{L_0, \ldots, L_M\}$. Each level is characterized by L_j: (v_j, p_j, f_j), $j \in \{1, \ldots, M\}$, which represents voltage, average power, and frequency respectively. We consider power-frequency levels of the Xscale processor as shown in Table 1. Here level 0 represents the idle power of the processor when no task is executed while the system stays in active state. Typically, the dynamic power-frequency function is convex. Thus, a processor running at lower frequency can execute the same number of cycles with lower energy consumption. However, this is not always the case when static power is considered. To find an energy optimal frequency, we represent energy efficiency of a v-f level L_i by δ_i = cycles executed/energy consumed = f_i/p_i. From Table 1 we can conclude that level 2 is the most energy efficient because executing at this level consumes the least energy for a given number of cycles. We call the most energy efficient level as the *critical level* and thus $f_{crt} = f_2$. Although it is desirable to execute tasks at this critical frequency level, due to unique task timing constraints executing tasks at the critical level may end up being insufficient to finish all task instances by their deadlines.

We also define utilization of a periodic task (U) with respect to the full speed provided by the processor; i.e., a task's utilization is its execution time under highest frequency divided by its period,

$$U_i = \frac{C_i/f_{max}}{T_i} \qquad (2)$$

The utilization for an entire task set is simply the accumulation of all tasks' utilization. In preemptive real-time system, a task set is schedulable by the EDF algorithm under a frequency level j if it meets the following condition:

$$U_{total} \leq \frac{f_j}{f_{max}} \qquad (3)$$

When total utilization is known, the most energy efficient frequency can be deduced from the equation, assuming $f_j \geq f_{crt}$. In this paper, we focus on a fully subscribed system with U_{total} = 100% to maximally utilize the available system resources. Our proposed framework aims to finish as many tasks as possible in such a system under any given energy harvesting profile.

Table 1. XScale Processor [13] Power and Frequency Levels

Level	0	1	2	3	4	5
Voltage(V)	-	0.75	1.0	1.3	1.6	1.8
Power(mW)	40	80	170	400	900	1600
Frequency(MHz)	idle	150	400	600	800	1000
Energy Efficiency	0	1.875	2.353	1.5	0.889	0.625

2.1.4 Energy Manager Module

A scheduler module is an important component of the system for information gathering and execution control. The scheduler gathers information by monitoring the energy storage medium and multi-core processor state (Figure 1). The gathered data, together with profiled periodic task set information, informs a management algorithm in our scheduler that coordinates operation of the multi-core platform. Each core is eventually assigned a strategy by the scheduler to guide intra-core task execution.

2.2 Scheduling Objective

Our primary objective is to *reduce total task miss rate or penalty* under variant harvested energy conditions at run-time. Our technique should react to changing harvested energy dynamics to complete as much (critical) work as possible, thus maximizing overall system utility and cost effectiveness.

3. PROPOSED FRAMEWORK

3.1 Motivation

Most prior work deals with dynamic solar energy variation by halting, dropping, or speeding up execution of a current task, changing instantly from an initial schedule deduced offline. For energy harvesting aware periodic task set scheduling, the most recent work, UTB [11], also follows this route. Although UTB deduces an optimal initial schedule offline assuming sufficient energy, it does not cope well with run-time energy variations, and there is scope for significant improvement as discussed below:

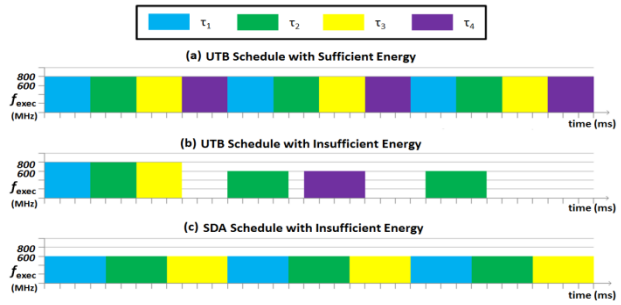

Figure 2. Motivation for proposed semi-dynamic approach

1) The task drop mechanism in UTB reacts to run-time energy shortages *passively*, only when the current task lacks sufficient energy to finish in time. In the motivational example shown in Figure 2, we assume a task set with four periodic tasks ($\tau_1 \sim \tau_4$), where each task has WCEC of 2.4 million CPU cycles and task period of 12ms. According to Table 1, Eq. (2) and Ineq. (3), UTB initially sets execution frequency to 800MHz so that all tasks can finish with the best efficiency if energy is sufficient, as shown in Figure 2(a). However, the real issue lies in the run-time energy management with an insufficient energy budget. Let us assume remaining energy in the energy storage is 7200µJ and harvested power in the next 36ms (3 periods) is 200mW, i.e., 200µJ of incoming energy per microsecond. After finishing three jobs, the energy storage is depleted, and UTB has to drop jobs due to insufficient energy, as shown in Figure 2(b). Only 6 out of 12 job instances are finished with UTB, resulting in a 50% miss rate. With the same energy budget, our proposed semi-dynamic algorithm (SDA) copes with energy shortage by *proactively* dropping tasks. It drops one task, τ_4, based on the energy budget which helps to execute the remaining tasks steadily at a lower frequency of 600 MHz. According to Table 1, executing at 600MHz has power consumption of 400mW, which is dramatically lower than 900mW at 800MHz due to the nonlinear relation between frequency and power consumption. As can be seen in Figure 2(c), all accepted job instances for $\tau_1 \sim \tau_3$ are finished and the overall miss rate is 25%, which is significantly lower than 50% achieved by UTB.

2) UTB encourages dropping tasks with longer execution time, because finishing them requires more energy than others. This biased dropping may be undesirable for real-world applications, as tasks with longer execution time may represent complex applications of high importance. Furthermore, the fundamental problem is the absence of flexibility in UTB to schedule tasks with controlled priority. In spite of the fact that SDA drops all job instances of τ_4 in Figure 2(c), its semi-dynamic framework provide flexibility to deal with this issue; e.g., by dynamically increasing miss penalty of τ_4 later to give it more priority in the next time epoch when rescheduling becomes possible again.

3) On multi-core platforms, UTB partitions tasks into separate sets and then executes each set on a core using a single-core scheduling algorithm. However, as all cores are dependent on the same energy source, such isolated run-time adjustment is not amenable to learning upcoming energy requirements of other cores, leading to suboptimal or even faulty schedules. In addition, static task partitioning in UTB wastes the flexibility provided by a multi-core platform. In contrast, SDA triggers task rescheduling to exploit multi-core flexibility.

In summary, there are many limitations with the best known prior work on harvesting-aware task scheduling. Our HY-SDA approach (described in the next section) addresses these limitations.

3.2 Semi-Dynamic Algorithm

In this section, we describe our novel energy and workload management framework (HY-SDA) based on a semi-dynamic algorithm, for systems with hybrid battery-supercapacitor energy storage, solar energy harvesting, and periodic task sets.

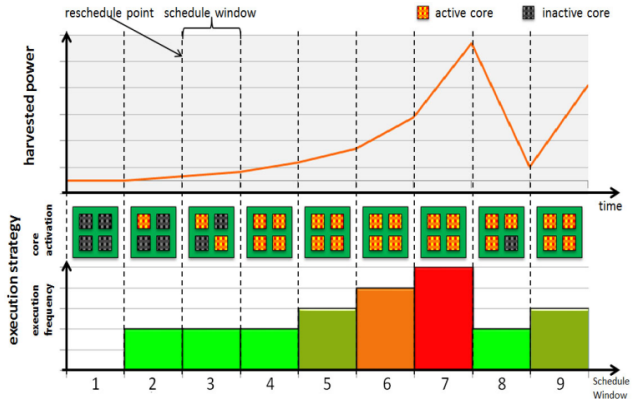

Figure 3. Illustration of semi-dynamic framework

One of the underlying ideas behind HY-SDA is to exploit time-segmentation during energy management, as illustrated in Figure 3. At each specified time interval (epoch), there is a reschedule point, where the execution strategy can be adjusted based on the energy budget available in the hybrid energy storage system. A time frame between two reschedule points is called a *schedule window*, within which a strategy specified at the beginning is in effect until the next reschedule point. Thus reschedule points provide dynamic adaptivity needed by the energy harvesting aware system to adjust the task execution strategy, while the schedule window enables stable execution so that periodic task information can be utilized for better energy savings, as in Figure 2(c). From schedule window 1 to 4 in Figure 3, it can be seen that under low energy conditions, HY-SDA maintains execution at optimal low (critical) frequency and increases the number of active cores to finish more tasks as

harvested energy increases. Cores only execute at higher frequency when the energy harvested is abundant. Thus HY-SDA has better energy efficiency as execution frequency within a schedule window remains stable.

At each reschedule point, our technique is composed of three stages: *(i) energy budgeting,* to take advantage of our proposed hybrid storage system, *(ii) active core count selection*, which selects the number of processing cores to activate, and *(iii) penalty-aware task rejection,* to filter out subset of tasks that are less important and cannot be supported by the energy budget. These three stages are organized in an order that successor stages make use of efforts made by previous stages, rather than diminishing them, and are described in the following sections (3.2.1-3.2.3).

3.2.1 Hybrid Storage System and Energy Budgeting

In this section, we introduce our hybrid energy storage system and an energy budgeting heuristic to make use of its properties.

Battery-Supercapacitor Hybrid Energy Storage: Inspired by Ongaro et al.'s work [5], we propose a hybrid energy storage system with one Li-Ion battery and two separate supercapacitors connected by a dc bus, as shown in Figure 4. During each schedule window, one capacitor is used to collect energy extracted from the PV array, while the other one is used as a power source for system operation or battery charging. At each reschedule point, the two supercapacitors switch their roles. Supercapacitors charge the battery only when their saved energy exceeds peak requirements of processors running at full speed. The PV array, battery and supercapacitors are coupled with bidirectional dc-dc converters to serve the purpose of voltage conversions between components with maximum power point tracking (MPPT) [7] and voltage level compatibility. This hybrid battery and dual-supercapacitor design has many advantages:

- It requires small capacity for the two supercapacitors, as each of them is only used to keep energy harvested during one single schedule window and is discharged in the next schedule window;
- The supercapacitor with energy buffered during the last schedule window acts as a known stable energy source for the system, which filters out short term solar energy variation. Thus, no energy prediction scheme is needed in our design, avoiding complexity and inaccuracy introduced by non-ideal prediction mechanisms. Moreover, the stable energy source makes it possible to charge the battery with a steady constant current for more effective charging [6];
- The supercapacitors can support embedded processors directly, taking advantage of a much lower charging/discharging overhead compared to an electrochemical battery;
- The battery offers high capacity to preserve energy especially in scenarios with excessive harvested energy.

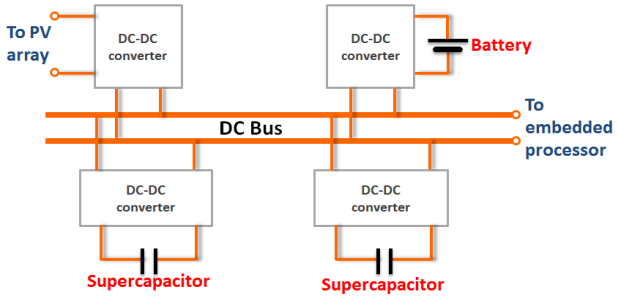

Figure 4. Proposed hybrid energy storage module

Energy Budgeting: Our energy budgeting heuristic selects among energy sources (supercapacitors and battery), sets the amount of energy to charge the battery for (E_{chrg}), and, assigns energy budget for system execution in the upcoming schedule window (E_{window}), as shown in Algorithm 1. The heuristic is based on storage levels of the battery (LV_B) and supercapacitor (LV_C). We assume that LV_B is provided directly by the energy storage, while the storage level of the supercapacitor can be classified into three levels of LV_C, (lines 1-3). As we want to avoid battery charging/discharging overhead, there are only 2 scenarios where the battery is selected as a power source: one is when energy harvested in the supercapacitor is below a critical level ($LV_C = 1$); the other is when battery storage level is high ($LV_B = 3$) such that battery overflow becomes a possibility (line 4). On the other hand, the battery is charged only when energy in the supercapacitor exceeds peak requirements of the processors (lines 12-14).

Algorithm 1 Energy Budgeting Heuristic

Input: (i) harvested energy in charged capacitor, E_C; (ii) battery energy storage level, LV_B; (iii) energy budget to execute one core at critical frequency, E_{crt}; (iv) energy budget to execute one core at maximum frequency, E_{max}; (v) number of cores in embedded processor, NUM_CORE
Output: (i) assigned energy budget for next schedule window, E_{window}; (ii) decision to activate battery as power source, B_{on}; (iii) energy to be charged into battery during next window with constant current, E_{chrg}

1. **if** $E_C < E_{crt}$ **then** $LV_C = 1$
2. **else if** $E_C > E_{max} \times$ NUM_CORE **then** $LV_C = 3$
3. **else** $LV_C = 2$
4. **if** $LV_B > LV_C$ **then**
5. B_{on} = true
6. **if** $LV_B = 2$ **then** $E_{window} = E_{crt} \times$ NUM_CORE
7. **if** $LV_B = 3$ **then** $E_{window} = E_{max} \times$ NUM_CORE
8. **else**
9. B_{on} = false
10. **if** $LV_C = 1$ **then** $E_{window} = 0$
11. **if** $LV_C = 2$ **then** $E_{window} = E_C$
12. **if** $LV_C = 3$ **then**
13. $E_{window} = E_{max} \times$ NUM_CORE
14. $E_{chrg} = E_C - E_{window}$

3.2.2 Active Core Count Selection

The main reason for having an active core count selection heuristic is that running a processor below its critical frequency actually decreases energy efficiency, as can be seen in Table 1. This situation can occur when the energy budget is so low that only a small subset of tasks can be accepted, i.e., after evenly distributing these tasks to all cores, utilization on each core is smaller than maximum utilization supported by the critical frequency. With our active core count selection heuristic, we can shut down some cores at each reschedule point based on the estimated energy budget. The power dissipated by inactive cores is negligible and the remaining cores can then receive enough workload to run at critical frequency. Also the associated power state switching overhead is minimal as we only trigger core shutdown at reschedule points. Also our heuristic should compare resulting efficiencies before making shutdown decisions.

The pseudo code of the active core count selection heuristic is given in Algorithm 2. Initially, the scheduler gets the energy budget for the upcoming schedule window from the hybrid energy storage system. Then, the core shutdown procedure is triggered when the energy budget is unable to support all active cores to execute at the critical frequency (line 2). Subsequently (lines 3-10) if one less active core results in a better efficiency, then the scheduler shuts down one core. If the energy budget for the current schedule window is extremely low, eventually all cores in the system will be shut down to save harvested energy for future execution. Recursively, these steps set the number of cores to keep

active. Finally, the objective task-set utilization for penalty-aware task rejection is obtained by summing up supported utilization of each core (line 11). As a result of this selection, the number of cores activated is tightly related to the energy budget available.

Algorithm 2 Active Core Count Selection Heuristic

Input: (i) energy budget for coming schedule window, E_{window}; (ii) energy budget to execute one core at critical frequency, E_{crt}; (iii) dual-speed method energy efficiency profile for task utilizations from 0 to 1, δ(U); (iv) number of cores in embedded processor, NUM_CORE
Output: (i) number of cores to active in next schedule window, num_active_core; (ii) objective utilization for next window, U_{obj}

1. num_active_core ← NUM_CORE
2. **while** $E_{per\_core} < E_{crt}$ **and** num_active_core > 0 **do**
3. $E_{num\_core} \leftarrow E_{window}$ / num_active_core
4. $E_{num\_core-1} \leftarrow E_{window}$ / (num_active_core-1)
5. calculate f_{num_core-1} and f_{num_core}, maximum frequencies supported by E_{num_core-1} and E_{num_core}
6. based on Inequality (3), calculate U_{num_core-1} and U_{num_core}, maximum utilization supported by f_{num_core-1} and f_{num_core}
7. look up profile for δ (U_{num_core}) and δ (U_{num_core-1})
8. **if** δ(U_{num_core}) < δ(U_{num_core-1}) **then**
9. num_active_core ← num_active_core – 1
10. update E_{per_core} , U_{per_core}
11. $U_{obj} \leftarrow U_{per\_core} \times$ num_active_core

3.2.3 Penalty-Aware Task Rejection and Assignment

To add task priority control in HY-SDA, we distinguish a task's importance by assigning a miss penalty to each task [15]. In this step, our scheme rejects tasks with lower penalty density (see Section 2.1.2) first, rather than simply drop tasks with longer execution time to allocate the limited energy budget to more important tasks for miss penalty reduction. In particular, for the case when all tasks are assigned an identical miss penalty, this scheme reduces miss penalty equivalent to miss rate. We describe our task rejection heuristic below in Algorithm 3.

Algorithm 3 Penalty Aware Task Rejection and Assignment Heuristic

Input: objective utilization from algorithm 2, U_{obj}
Output: optimal execution frequency for each core, f_{opt}(core_id)

1. sort task set T in non-decreasing order of tasks' penalty densities
2. $T_{accepted} \leftarrow T$
3. **for** n = 1:N **do**
4. **if** $U_{accepted} > U_{obj}$ **then**
5. reject n^{th} task
6. **else**
7. done with task rejection, **break**
8. sort accepted task set $T_{accepted}$ in non-increasing order of task utilization
9. **for** n = 1:$N_{accepted}$ **do**
10. assign n^{th} task to active core with the lowest utilization
11. get assigned task utilization for each active core, $U_{(core\_id)}$
12. based on Inequality (3), calculate f_{opt} (core_id)
13. execute assigned tasks on each core with dual-speed heuristic

In lines 1-7, we sort all tasks in non-decreasing order of tasks' penalty densities so that we can then reject tasks one by one until the remaining tasks' total utilization is lower than objective utilization given by Algorithm 2. The remaining tasks form the accepted task set and are assigned to all active cores using a simple but effective heuristic in lines 8-10. This heuristic not only enables priority control among tasks, but also evenly distributes workload to each core for execution under a stable frequency for better efficiency. After all accepted tasks are assigned, we get actual utilization and optimal frequency for each core. A dual-speed heuristic is implemented to approximate each core's designated optimal frequency by switching between two discrete frequency levels available in our XScale model [17].

4. EXPERIMENTAL STUDIES

4.1 Experiment Setup

We developed a simulator in C++ to evaluate the effectiveness of our proposed semi-dynamic energy and workload management algorithm with hybrid energy storage (HY-SDA). We use the approach given in Mirhoseini et al.'s work to model rate capacity effect of batteries [12]. We compared our scheme to the state of the art Utilization-Based Algorithm (UTB) [11], which we modeled in our environment. In addition, we implemented three variants of HY-SDA, namely (i) BA-SDA: SDA for battery-only system with doubled battery capacity; (ii) CA-SDA: SDA for supercapacitor-only system with doubled supercapacitor capacity, and (iii) MISS-SDA: a modified version of HY-SDA to focus on miss rate reduction. UTB, BA-SDA and CA-SDA rely on *moving average* algorithm for energy harvesting prediction [10] as they do not have a supercapacitor to buffer harvested energy in each schedule window. The processor power model is shown in Table 1. The energy harvesting profile is obtained from historical weather data from Golden, Colorado, USA, provided by the Measurement and Instrumentation Data Center (MIDC) of National Renewable Energy Laboratory (NREL) [16]. Our system only executes during daytime over a span of 750 minutes, from 6:00 AM to 6:30 PM, when solar energy is available. We randomly generate 50 task sets with full utilization. In each task set, tasks are assigned with miss penalty ranging from 1 to 100 with uniform distribution. Finally, we set schedule window size (for all SDA based algorithms) to be five minutes compared to the average task execution time of a few hundred milliseconds – not too short to cause frequent changes in the execution policy and not too long to lead to high capacity requirements from the supercapacitors in the system.

4.2 Experiment Results

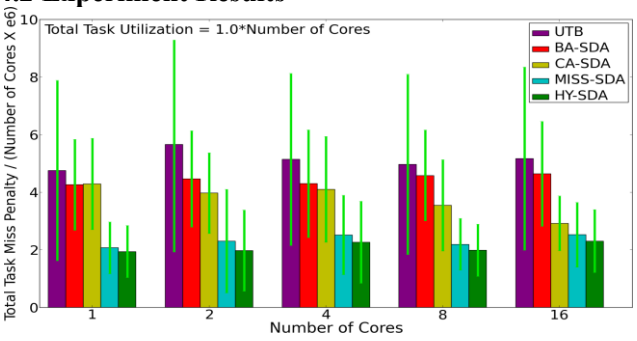

Figure 5. Overall miss penalty comparison

We first compared average overall miss penalty for the various techniques, with increasing multi-core platform complexity (1 to 16 cores). Capacities of batteries and supercapacitors, and nominal harvested energy scale linearly with number of cores in the processors. The results for this experiment are shown in Figure 5. UTB, BA-SDA and CA-SDA can be seen to have a much higher miss penalty. Respectively, the performances of these techniques mainly suffer from unstable execution frequencies, lower charging/discharging efficiency of the battery, and limited capacity of the supercapacitor. Moreover, UTB has much higher variation in results, due to the limitation that its run-time task dropping scheme does not take task miss penalty into consideration. Also note that CA-SDA has an advantage over BA-SDA with increasing number of cores in system. This is due to the rate capacity effect of the battery and the fact that more cores means higher current demand. On the other hand, our HY-SDA scheme leads to the lowest task miss penalty, while MISS-SDA results in slightly higher miss penalty than HY-SDA because it instead focuses on miss rate

reduction. Figure 6 shows a comparison of the miss rate for the various techniques. The miss rate of MISS-SDA is lower and has less variation compared to that of HY-SDA, which is to be expected as MISS-SDA aims to reduce miss rate. Note that although HY-SDA is aimed at miss penalty reduction, it still achieves much lower miss rate than UTB, BA-SDA and CA-SDA.

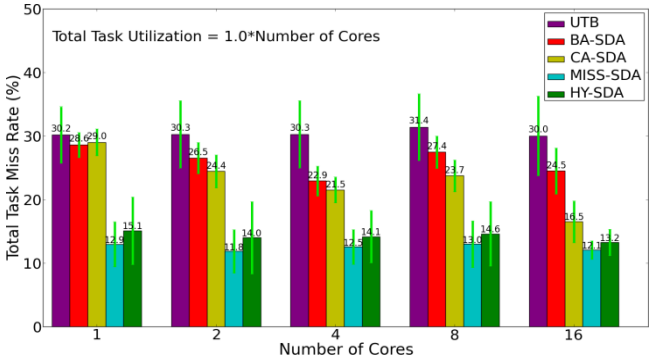

Figure 6. Overall miss rate comparison

We also compared HY-SDA with UTB for a scenario where harvested power fluctuates over time. The result of this experiment on a 16 core system is shown in Figure 7. First of all, we can see that the HY-SDA actually results in a higher miss rate than UTB at the beginning, because it waits until the supercapacitor is charged to the critical level. Subsequently, higher miss rate reduction for HY-SDA is achieved when harvesting power is low or changes dramatically, reflecting the advantage that HY-SDA has over UTB to cope with stringent energy budgets and filter out solar harvesting variations. Moreover, HY-SDA results in a more significant miss rate reduction after 12 PM. The reason for this is that HY-SDA's high energy efficiency leads to more energy savings in the battery, which enables more tasks to be executed and meet their deadlines.

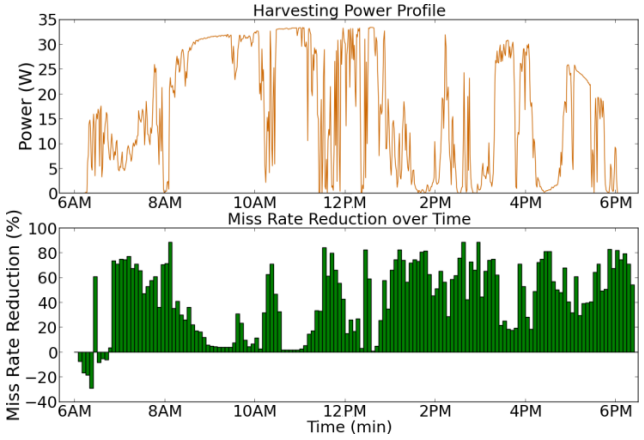

Figure 7. Miss rate reduction of HY-SDA compared to UTB

To highlight the advantage of the dual-supercapacitor design in our proposed hybrid energy storage system, we define a new metric, *budget violation rate*, which is the percentage of unfinished jobs for the accepted tasks. Results for a 16-core configuration are shown in Figure 8. HY-SDA and MISS-SDA, which make use of the hybrid energy storage system, have much lower budget violation rates, because for most of the time their energy budgeting is based on the known amount of energy buffered in supercapacitor. In contrast, other schemes' energy budgeting can be misled by inaccuracy in their solar energy harvesting prediction, which explains their higher energy budget violation rates.

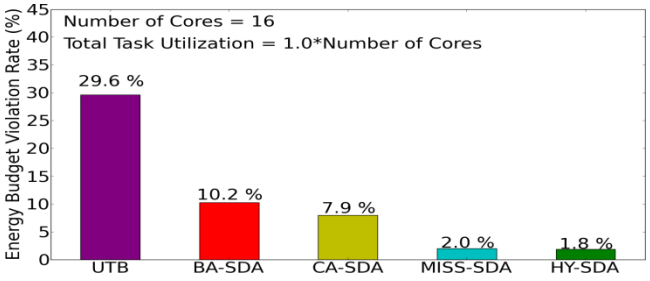

Figure 8. Budget violation rate comparison

5. CONCLUSION

In this paper, we proposed a new framework for energy and workload management (HY-SDA) based on a semi-dynamic algorithm, for real-time multiprocessor embedded systems with solar energy harvesting and a hybrid battery-supercapacitor energy storage system. Compared to the best known previous work, our approach is very promising, reducing miss rate by up to 61 % and miss penalty by up to 65% for high intensity workloads.

ACKNOWLEDGEMENTS

This research is sponsored in part by grants from NSF (CCF-1252500) and SRC.

REFERENCES

[1] Nvdia Tegra 3 processor, http://www.nvidia.com/object/tegra-3-processor.html.

[2] The benefits of multiple CPU cores in mobile devices, http://www.nvidia.com/content/PDF/tegra_white_papers/Benefits-of-Multi-core-CPUs-in-Mobile-Devices_Ver1.2.pdf.

[3] V. Raghunathan et al., "Design considerations for solar energy harvesting wireless embedded systems," in IPSN, 2005, pp. 457-462.

[4] X. Jiang, J. Polastre, and D. Culler, "Perpetual environmentally powered sensor networks," IPSN, 2005, pp. 463-468.

[5] F. Ongaro, S. Saggini, and P. Mattavelli, "Li-Ion battery-supercapacitor hybrid storage system for a Long Lifetime, Photovoltaic-Based Wireless Sensor Network," IEEE Trans. Power Electron., vol. 27, issue 9, pp. 3944-3952, Sept. 2012.

[6] B. Carter, J. Matsumoto, A. Prater, and D. Smith, "Lithium ion battery performance and charge control," in IECEC, 1996, vol. 1, pp. 363-368.

[7] N. Femia et al., "Distributed maximum power point tracking of photovoltaic arrays: novel approach and system analysis," IEEE Trans. Indust. Electron., vol. 55, no. 7, pp. 2610-2621, Jul. 2008.

[8] C. Moser, D. Brunelli, L. Thiele, and L. Benini, "Lazy scheduling for energy-harvesting sensor nodes," in DIPES, 2006, pp. 125-134.

[9] S. Liu, Q. Qiu, and Q. Wu, "Energy aware dynamic voltage and frequency selection for real-time systems with energy harvesting," in DATE, 2008, pp. 236-241.

[10] S. Liu, J. Lu, Q. Wu, and Q. Qiu, "Harvesting-aware power management for real-time systems with renewable energy," IEEE Trans. VLSI Syst., vol. 20, no. 8, pp. 1473-1486, Aug. 2012.

[11] J. Lu, Q. Qiu, "Scheduling and mapping of periodic tasks on multi-core embedded systems with energy harvesting," in IGCC, 2011.

[12] A. Mirhoseini, F. Koushanfar, "HypoEnergy: hybrid supercapacitor-battery power-supply optimization for energy efficiency," DATE '11.

[13] Intel XScale, http://download.intel.com/design/intelxscale/27347302.pdf

[14] Z. Xu et al., "Electrochemical supercapacitor electrodes from Sponge-like graphene nanoarchitectures with ultrahigh power density", J. Phys. Chem. Lett., pp. 2928-2933, Oct, 2012 (3).

[15] J. Chen, T. Kuo, C. Yang, and K. King, "Energy-efficient real-time task scheduling with task rejection", in DATE, 2007, pp. 1-6.

[16] NREL Measurement and Instrumentation Data Center (MIDC), http://www.nrel.gov/midc/.

[17] D. Rajan, R. Zuck, and, C. Poellabauer, "Workload-aware dual-speed dynamic voltage scaling", in RTSCA, 2006, pp. 251-256