COL862: Low Power Computing

MEANTIME: Achieving Both Minimal Energy and Timeliness with Approximate Computing

Authors: Anne Farrell and Henry Hoffmann,
University of Chicago.
2016 USENIX Annual Technical Conference.

Presented By: Rajesh Kedia, Radhika D.

Date: 05-09-2016.

Summary

- Conflicting requirements:
  - Hard timing constraints require pessimistic resource allocation.
  - Energy efficiency requires just-right resource allocation.
- Solution:
  - Compromising on accuracy (approximate computing) can reduce energy consumption.
  - Exploit approximate computing to meet timeliness and energy efficiency.
- Contributions:
  - A run-time framework for approximate computing to meet minimal energy as well as hard timing constraints.
  - Results from 6 different applications on Linux/ARM demonstrating the effectiveness of the technique.

Problem Statement

- Runtime varies for a task depending on input conditions (see Table below).
- Meeting hard deadlines require allocating resources for the worst case.
- Scheduling based on worst case execution time (WCET) can lead to waste of resources/energy.
- Energy aware scheduling can lead to missing deadlines at times.
- No solution has been proposed to meet lower energy as well as hard deadlines.

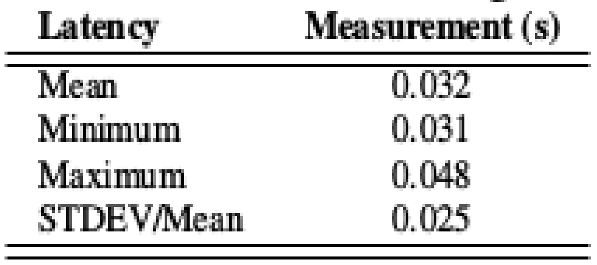
Table 2: Radar timing.

Latency	Measurement (s)
Mean	0.032
Minimum	0.031
Maximum	0.048
STDEV/Mean	0.025

Approximate Computing

- What is Approximate Computing?
- Examples?
- Tradeoff application characteristics to achieve energy/cost reduction.
- Well suited for signal, media, image processing applications where application accuracy is quantifiable and tradeoff is acceptable (e.g. reduced energy but lower SNR).
- Other metrics for tradeoff: Precision, flickers, user experience, etc.
- Brings in a 3rd dimension to energy/ execution time tradeoff.

Motivational Example

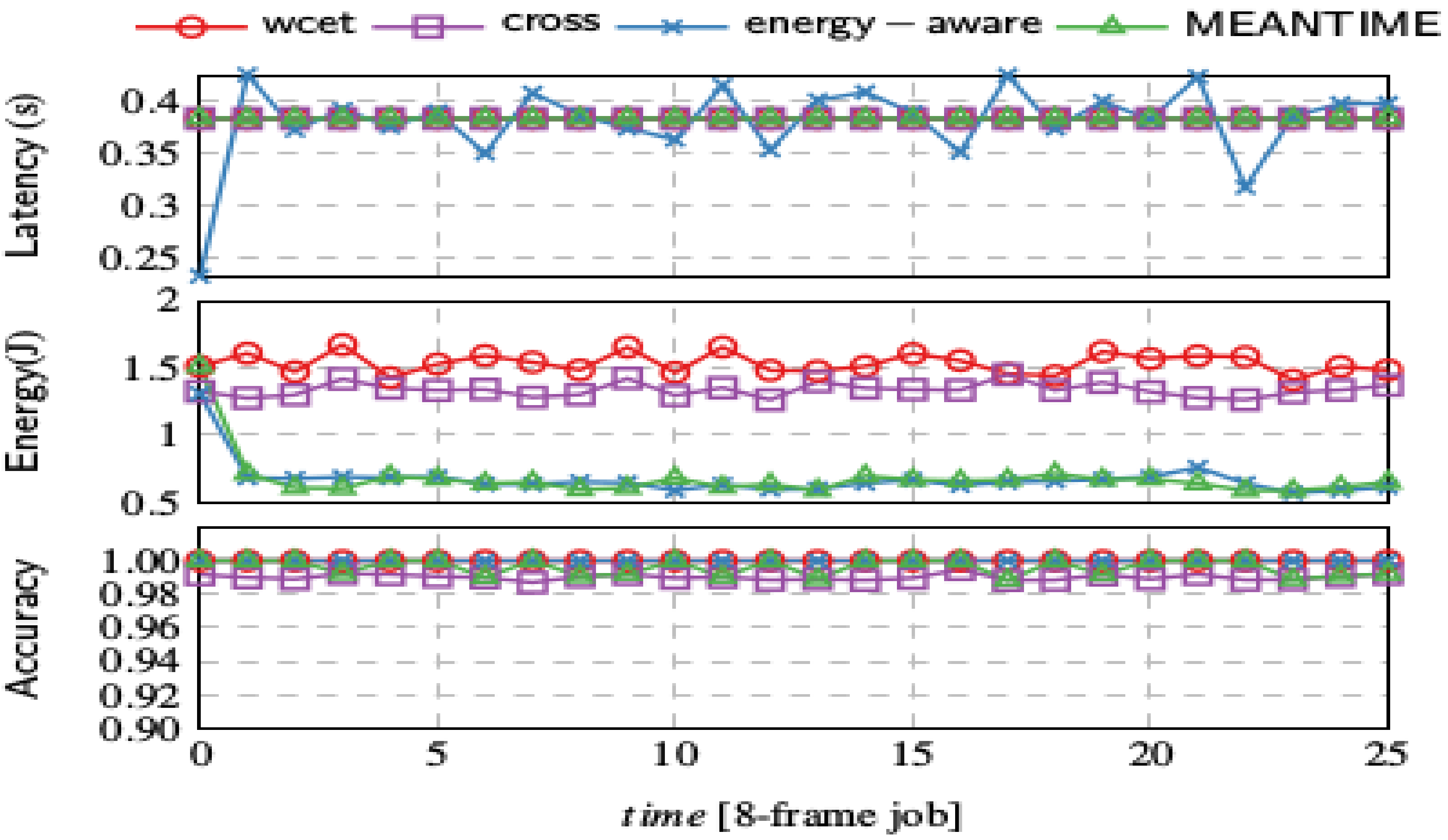

Figure 2: Comparison of techniques for the radar.

Prior Work (1)

- Manipulate slack to meet deadlines and save energy. e.g. use DVFS, sleep modes, etc.
  - Soft timing contrained systems have more flexibility since occasional timing violations are acceptable.
- Dynamic tailoring of behavior
  - Maintain accuracy while minimizing energy.
  - Maximize accuracy within given energy.
  - Meet performance goal while maximizing accuracy.
  - None of them guarantee hard real time behavior.

Prior Work (2)

- Cross layer optimization: Combine application approximation and resource allocation.
  - Coordinating OS and application to meet desired energy goals.
  - Hierarchical: Making system level adaptation and then application tuning.
  - Pick up any 2 out of 3 parameters (performance, power and accuracy). Soft guarantees in chosen 2 while optimizing the third.
  - Again, none of these approaches provide hard real time guarantee.

Prior Work (3)

- System level approaches
  - Coordinating CPU power states, memory and disk to meet performance goals while minimizing power.
  - Clock speed, cache and memory bandwidth coordinated.
  - Provide:
    - Hard real time guarantee without any consideration of energy.
    - Or, Energy optimization but only soft real time behavior.

MEANTIME (Minimal Energy ANd TIMEliness)

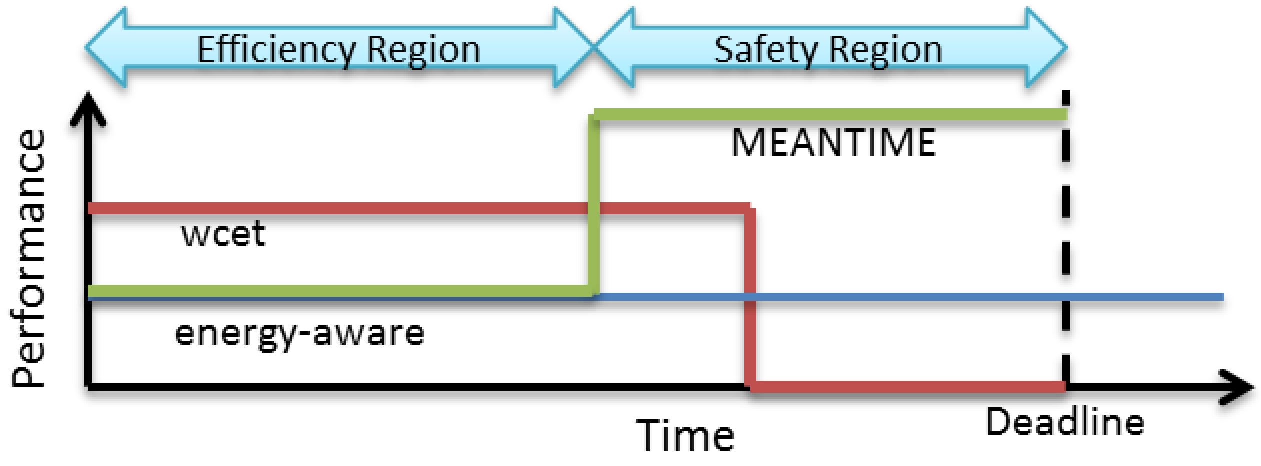

Figure 1: Conceptual model of MEANTIME compared to worst-case and energy-aware resource allocation.

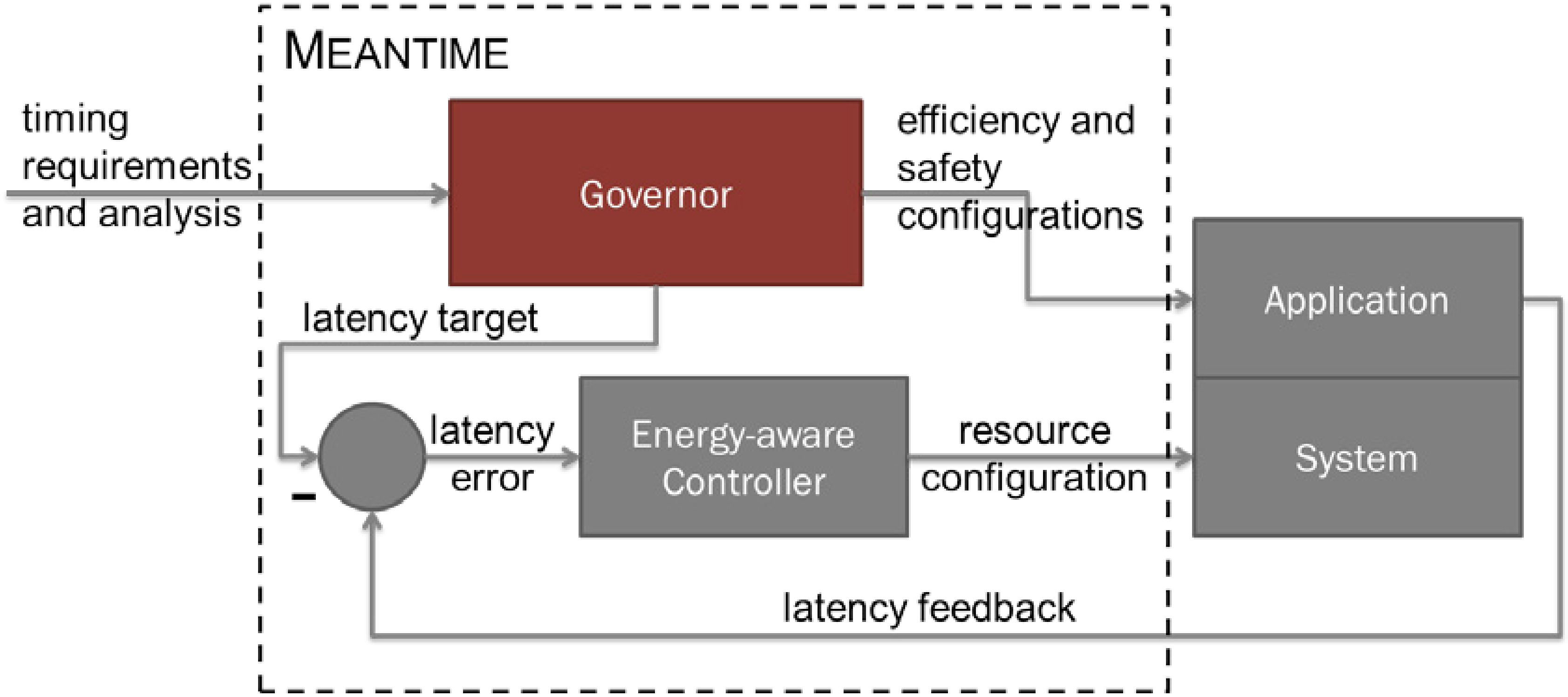

Figure 3: Overview of the MEANTIME approach.

MEANTIME Components

- Approximate Computing:
  - MEANTIME relies on applications supporting approximate computing.
  - Accuracy need to be a total order, exact values needn't be known, but just a relative ranking.
- Control for Energy Efficiency
  - An open source energy efficient control unit is used named POET.
  - The controller allocates resources which are best from an energy perspective, and reacts to any timing violation.
  - Controller can provide only soft timing guarantees since it adapts to a violation to fix the timing error.

Governor: Notation and Goal

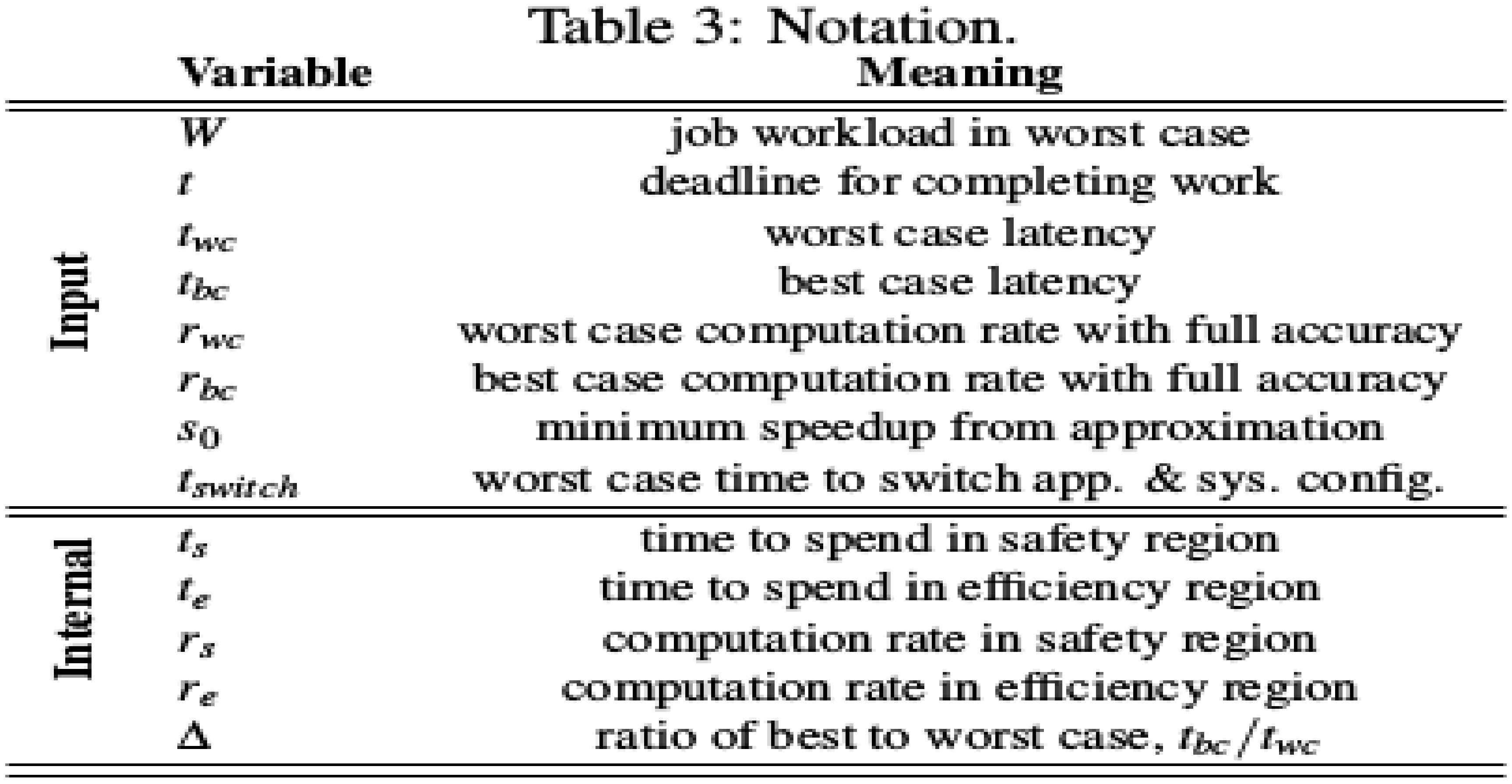

Table 3: Notation.

	Variable	Meaning
Input	W	job workload in worst case
	t	deadline for completing work
	t_{wc}	worst case latency
	t_{bc}	best case latency
	r_{wc}	worst case computation rate with full accuracy
	r_{bc}	best case computation rate with full accuracy
	s_0	minimum speedup from approximation
	t_{switch}	worst case time to switch app. & sys. config.
Internal	t_s	time to spend in safety region
	t_e	time to spend in efficiency region
	r_s	computation rate in safety region
	r_e	computation rate in efficiency region
	Δ	ratio of best to worst case, t_{bc}/t_{wc}

$$\begin{aligned} t_s \cdot r_s + t_e \cdot r_e &\geq W \\ t_s + t_e &\leq t \\ t_s,\ t_e &\geq 0 \end{aligned}$$

$$t_e = \frac{t_{wc} - s_0 \cdot (t - t_{switch})}{1 - s_0}$$

Governor: Minimizing accuracy loss

- Various approximation methods can provide different accuracy versus speed tradeoff.
- Need to use best accuracy within given hard timing constraint.
- Can be solved once only during initialization.

$$maximize \sum_i \frac{t_s^i}{t} \cdot a^i$$

$$s.t.$$

$$\begin{aligned}
\sum_i t_s^i \cdot r_{wc} \cdot s^i + t_e \cdot \Delta r_{wc} &\geq W \\
t_{switch} + \sum_i t_s^i + t_e &\leq t \\
t_s^i,\ t_e &\geq 0
\end{aligned}$$

Governor: Maintain Responsiveness

- Problem:
  - When an application shifts from a computationally intensive phase to an easy phase, the controller reacts to this change by reducing resource usage.
  - Now when an application requires higher computation, the deadline get
  - The control system reacts by detecting an error between the desired behavior and observed behavior.
  - But in MEANTIME, the control system will not detect the phase shift since every deadline is respected.
- Solution:
  - MEANTIME estimates what the latency would have been if the application had not switched to a less accurate configuration and passes this latency to the controller.
  - $$\hat{t} = t_e + t_s \cdot s_0 + t_{switch}$$

MEANTIME Usefulness

- Can MEANTIME be used for all applications?
- MEANTIME is not designed for all embedded applications, but for those that
  - Have viable performance/accuracy trade-offs,
  - Must satisfy hard real-time constraints and minimize energy consumption despite large fluctuations in application workload, and
  - Have progress indicators and models of completion.

BACKUP

Prior Work

Green: A Framework for Supporting Energy-Conscious Programming using Controlled Approximation.

W. Baek and T. Chilimbi.
PLDI, June 2010.

Summary

- Energy efficient computing by compromising slightly on QoS.
- The framework takes complete implementation of functions and their approximate versions as an input. It also takes the QoS measurement criteria as an input.
- Similarly, loops are approximated by running them with a fewer number of iterations.
- The framework has a calibration phase in which the program is run with multiple calibration inputs to understand QoS vs. runtime improvements.
- Then use this model to decide the operating points under a given QoS constraint.
- Also, online monitoring is done to check if the expected QoS and actual QoS are within limits else an error correction is done.

Benchmarks used

- Bing Search:
  - The base version of Bing Search processes all the matching candidate documents. Instead, we can limit the maximum number of documents (M) that each query must process to improve performance and reduce energy consumption while still attempting to provide a high QoS.
- Graphics: 252.eon:
  - The main loop in 252.eon iterates N^2 iterations and sends a ray at each iteration to refine the rasterization. As the loop iteration count goes higher, QoS improvement per iteration can become more marginal. In this case, the main loop can be early terminated while still attempting to meet QoS requirements.
- Machine Learning: Cluster GA:
  - By terminating the main loop earlier, we can achieve significant improvement in performance and reduction in energy consumption with little QoS degradation.

Benchmarks used

- Signal Processing: Discrete Fourier Transform:
  - In the core of DFT, sin and cos functions are heavily used. Since the precise version implemented in standard libraries can be expensive especially when the underlying architecture does not support complex FP operations, the approximated version of sin and cos functions can be effectively used if it provides sufficient QoS. We implement several approximated versions of sin and cos functions and apply them to our DFT application.
- Finance: blackscholes:
  - The core computation makes heavy use of the exponentiation exp and logarithm log functions. We provided a series of approximate versions of these functions that use the corresponding Taylor series expansions with varying number of polynomial terms.

Prior Work

Eon: A Language and Runtime System for Perpetual Systems.

J. Sorber, A. Kostadinov, M. Garber,
M. Brennan, M. D. Corner, and E. D. Berger.
SenSys'07

Summary

- For energy harvesting systems, depending on the available energy, the systems can adjust the service level.
- However, predicting the energy consumption is difficult and hence the authors propose Eon as an energy aware programming language.
- Eon’s automatic energy management then dynamically adapts these states to current and predicted energy levels.
- Eon’s adaptation algorithms require hardware support. We have built a new charging and energy management board.
- Applications explored: Turtle Tracking, Automobile Tracking, Remote Camera.

Prior Work

Managing Battery Lifetime with Energy-Aware Adaptation

J. Flinn and M. Satyanarayanan.
ACM Trans. Comp. Syst. 22.2 (May 2004).

Summary

- Implements powerscope, to measure the current consumption and profiling of individual applications.
- Fidelity/accuracy of applications can be traded off for energy savings.
- Implements OS calls where-in user specifies his goal of battery lifetime and the OS adjusts the fidelity accordingly. provides system calls like fidelity_register() to register various fidelity levels, begin_fidelity_op(), end_fidelity_op(). cooperative model: app can specify whatever they support, OS uses only that information to optimize.
- Maintains priority of applications and when energy demand increases, it lowers the fidelity of the lowest priority application and continues until the demand is in budget.

Prior Work

Energy-Efficient Soft Real-Time CPU Scheduling for Mobile Multimedia Systems

W. Yuan and K. Nahrstedt.
ACM SIGOPS Operating Systems Review 37.5 (2003), pp. 149–163.

Summary

- For a soft real time system, performs CPU scheduling to meet the QoS and keep it energy efficient.
- Extracts stochastic parameters (e.g. execution cycle count, etc.) to perform scheduling under given QoS requirements.
- Uses online estimation for cycle counts and uses them during the optimization. Computes CDF (histogram) for cycle count for task instances.
- Gradually increase the CPU freq. within a task so that if task finishes earlier, lesser energy is consumed. Uses "starting a job at a lower speed and then accelerating as it progresses" technique.
- Changes done to linux scheduler
- syscalls added: start_srt(), finish_job() and exit_srt(). //srt=soft realtime
- set_budget and set_dvspnt functions added to tell the scheduler about task's requirements.
- Also modified the PCB (Process Control Block) to add soft realtime related info about process.