Efficient Power Management Schemes for Dual-Processor Fault-Tolerant Systems

Yifeng Guo, Dakai Zhu
The University of Texas at San Antonio

Hakan Aydin
George Mason University
Outline

- Background and Motivation
- System Models
- POED Algorithm
- Application to Energy Efficient Fault-Tolerant Real-Time Systems
- Simulation Results and Discussions
- Conclusions
Energy is Precious: Everywhere!

- **Popularity mobile devices**
  - Smart phones: 492 million in 2011
  - *Battery* operated: limited capacity;
    Smart phones: a few days
    Laptop: 3 – 10 hours

- **Data centers and servers**
  - Excessive heat → cooling
  - Operation cost: 1.5% electricity in US (2007) → billion $
Power Reduction Techniques

- **LCD**
  - Brightness; on/off

- **Memory**
  - Different power states

- **Disks → SSD**
  - Spin down

- **CPU**
  - Voltage/frequency scaling
  - Low power states: 1-5% of the peak power
A Simple System-Level Power Model

Example: A real time task $T$ needs 2 units with processing speed $f$

![Diagram showing execution and power consumption]

$E = P^* t$;
Energy from:
$P_s$: the same;
$P_{ind}$: increases;
$P_d$: decreases

- A simple system power model

$$P(f) = P_s + (P_{ind} + P_d) = P_s + P_{ind} + C_{ef} f^m$$

Minimum energy efficient frequency

$$f_{ee} = \left( \frac{P_{ind}}{C_{ef} \cdot (m - 1)} \right)^{\frac{1}{m}}$$

[HARSH-2013, Shenzhen, China]
Fault-Tolerant System Design

✧ Faults
  - Transient fault
    ✓ Temporary, will disappear
  - Permanent fault
    ✓ System component → replacement/redundancy

✧ Techniques
  - Time redundancy
    ✓ available system slack for recovery execution
    ✓ only tolerate transient fault w/o permanent fault
    ✓ task with utilization > 50% can not be managed
  - Hardware redundancy
    ✓ Both transient & permanent fault (e.g., duplex, TMR system)
    ✓ Tremendous energy consumption

HARSH-2013, Shenzhen, China
Transient Faults/Reliability vs. DVFS

- Transient faults vs. Critical charge $Q_{\text{crit}}$
  - smallest charge needed to flip circuit states

Figure 3. Measured error rates for an 18×18-bit field-programmable gate array multiplier block at 90 MHz and 27°C.
Co-Management of Energy and Reliability

- **Reliability-Aware Power Management** [Zhu ’06, Qi ’11]
  - low supply voltage (DVFS) → more transient fault
  - time redundancy

- **Standby-Sparing** [Ejlali ’09] and [Haque ’11]
  - dual-processor systems, aperiodic/periodic tasks
  - primary processor: primary tasks, DVFS
  - spare processor: backup tasks, DPM, deallocation
  - minimize the overlap between primary and backup
  - tolerate transient fault and one permanent fault

- **Secondary Execution Time Shifting (SETS)** [Unsal ’09]
  - periodic tasks
  - a mixed manner (P/B tasks)
  - static scheme to reduce overlap between primary and backup
Application Model

- $n$ periodic real-time tasks $\Psi = \{T_1, \ldots, T_n\}$;
- $T_i$: $(c_i, p_i)$
  - $c_i$: worst case execution time (WCET) at $f_{max}$ ($f_{max} = 1$);
  - $p_i$: period;
  - WCET = $c_i/f_i$ in a lower frequency;
  - $u_i = c_i/p_i$
  - $U = \sum u_i$, $i = 1, \ldots, n$
- $B_i$: backup for $T_i$
  - same parameters ($c_i$ & $p_i$)
  - no DVFS (transient fault)
  - different CPUs for $T_i$ and $B_i$ (permanent fault)
Problem to Solve

- **Hardware redundancy: Dual-CPU systems**
  - Tolerate a single permanent fault
  - Tolerate transient faults
- **Each task: primary & backup copies**
  - Primary & backup need on different CPUs
- **Standby-Sparing in Dual-CPU** [Haque ’11]
  - Example: $T_1(1, 5)$ and $T_2(2, 10)$

Slack time on secondary CPU is wasted!
Mixed Allocation of P/B Tasks

- Each CPU gets a set of mixed P/B tasks
  - Scale down primary tasks

Problem: backup tasks run concurrently with primary tasks → more energy consumption!
Differentiate Executions of P/B Tasks

- **P/B tasks: different preferences**
  - Primary tasks: as soon as possible (ASAP)
  - Backup tasks: as late as possible (ALAP)

Problem: how to efficiently schedule RT tasks with different preferences on each CPU?
RT Tasks with Preferences and Schedule

- A set of \( n \) periodic tasks: \( \Psi = \{T_1, \ldots, T_n\} \)
  - Each task has a preference: ASAP or ALAP
  - ASAP tasks (\( \Psi_S \)) & ALAP tasks (\( \Psi_L \))

\[
\Psi = \Psi_S \cup \Psi_L
\]  

[Guo TR’12]

A feasible schedule of tasks

- Schedule: \( S: t \rightarrow T_i, 0 \leq t \leq LCM, 1 \leq i \leq n \)
- \( T_i \) is executed in time slot \([t, t+1)\): \( S(t) = T_i \)
- No deadline miss
Accumulated ASAP/ALAP Executions

- **Accumulated ASAP execution before time** $t$
  \[ \Delta(S, t) = \sum_{z=0}^{t} \delta(S, z) \quad 0 \leq t \leq LCM \]
  where \( \delta(S, z) = 1 \) if \( S(z) = T_i \) and \( i \in \Psi_s \)

- **Accumulated ALAP execution after time** $t$
  \[ \Omega(S, t) = \sum_{z=t}^{LCM-1} \omega(S, z) \quad 0 \leq t \leq LCM \]
  where \( \omega(S, z) = 1 \) if \( S(z) = T_i \) and \( i \in \Psi_L \)
Optimal Preference-Oriented Schedules

- An ASAP-optimal schedule: $S_{asap}^{opt}$
  - If $S_{asap}^{opt}$ is a feasible schedule and, for any other feasible schedule $S$, there is:
    \[ \Delta(S_{asap}^{opt}, t) \geq \Delta(S, t) \quad (0 \leq t \leq LCM) \]

- An ALAP-optimal schedule: $S_{alap}^{opt}$
  - If $S_{alap}^{opt}$ is a feasible schedule and, for any other feasible schedule $S$, there is:
    \[ \Omega(S_{alap}^{opt}, t) \geq \Omega(S, t) \quad (0 \leq t \leq LCM) \]

- An PO-optimal schedule: $S^{opt}$
  - If $S^{opt}$ is a feasible schedule and, for any other feasible schedule $S$, there is:
    \[ \Delta(S^{opt}, t) \geq \Delta(S, t) \quad \underline{\text{and}} \quad \Omega(S^{opt}, t) \geq \Omega(S, t) \quad (0 \leq t \leq LCM) \]
Optimal Schedules vs. System Loads

- $U < 1$: discrepant optimal schedules with idle time

  - Example: $T_1 (1, 3)$, $T_2 (1, 4)$ and $T_3 (1, 6)$, $U = 0.75$

  where $\Psi_S = \{T_1\}$, $\Psi_L = \{T_2, T_3\}$

$U = 1$: harmonious optimal schedules
Preference-Oriented Earliest Deadline Heuristic

- **ASAP Scheduling Principle**
  - At any time, if there are ready ASAP tasks, they should be executed first provided that such executions will not lead to deadline miss for ALAP tasks

- **ALAP Scheduling Principle**
  - If there is no ready ASAP tasks, CPU should idle provided that it will not lead to deadline miss for ALAP tasks

- Explicitly manage *idle time with wrapper task* [Zhu ’09]
  - Idle time $\rightarrow$ *wrapper tasks* with deadlines
Preference-Oriented Earliest Deadline Heuristic

❖ POED scheduling algorithm: at time $t$

- If $T_k$ is a ready ASAP task with earliest deadline $d_k$, check look-ahead interval $[t, d_k]$
  - If there is free time, execute $T_k$ (maybe wrapped execution)
  - Otherwise, urgent execute the earliest deadline ALAP task

- If wrapper tasks $T_x$ with deadline $d_x$ (ASAP), check look-ahead interval $[t, d_x]$
  - If there is free time, execute $T_x$ (CPU free)
  - Otherwise, urgent execute the earliest deadline ALAP task

- No ASAP/wrapper tasks: execute ALAP tasks with EDF
Look-Ahead Interval

\[ Q_{la} = \{T_x, T_y\}, \quad T_x, T_y \in \Psi_L \]

\[ Q_{la} = \{T_x, T_y, T_z\}, \quad T_x, T_y \in \Psi_L, \quad T_z \in \Psi_S \]

no free section at the beginning
POED-Based EEFT on Duplex Systems

Steps:
- map primary tasks to two CPUs (e.g., WFD)
- *cross assign* backup tasks to CPUs
- calculate scaled frequency for primary tasks on each CPU
- on each CPU, execute tasks with the POED scheduler
  - Primary tasks have ASAP preference
  - Backup tasks have ALAP preference
  - When a task completes successfully on one CPU, notify other CPU to cancel its backup

Online Extension
- dynamic slack from task cancellation & AET << WCET
- further slow down primary/delay backup
An Example

- \( T_1 (1, 5) \) and \( T_2 (2, 10) \), \( U = 0.4 \)

\[ B_{1,1} \quad B_{1,2} \quad T_{1,1} \quad T_{1,2} \quad B_{2,1} \]

\[ f_1 = \frac{1}{4} \]

\[ f_2 = \frac{1}{4} \]
# Simulation Settings

<table>
<thead>
<tr>
<th>Power</th>
<th>$P_s$</th>
<th>0.01</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$P_{ind}$</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>$C_{ef}$</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$m$</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>frequency levels</td>
<td>0.4, 0.6, 0.8, 1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Application</th>
<th>Num Tasks/Task Set</th>
<th>10, 20, ..., 100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Utilization of Each Task</td>
<td>UUniFast scheme [Bini '04]</td>
</tr>
<tr>
<td></td>
<td>Period of Each Task</td>
<td>$[p_{min}, p_{max}]$ uniform dist.</td>
</tr>
<tr>
<td></td>
<td>$p_{max}$</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>$p_{min}$</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Num Tasks Sets/Data Point</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>$U$ (static load)</td>
<td>0.3, 0.4, ..., 1.0</td>
</tr>
<tr>
<td></td>
<td>$\alpha_i$ (dynamic load)</td>
<td>Uniform dist. w/ average $\alpha$</td>
</tr>
</tbody>
</table>

| Processor | Num of Processors | 2 (dual-processor system) |
Schemes for Comparisons

- **Baseline: Basic-SS**
  - Basic standby-sparing w/o scaled frequency

- **Existing schemes for comparison**
  - **SS-SPM**
    - Standby-sparing w/ offline scaled frequency
  - **SS-DPM (ASSPT [Haque ’11])**
    - Standby-sparing w/ further scaled frequency using online slack

- **Proposed schemes**
  - **POED-SPM**
    - POED w/ offline scaled frequency
  - **POED-DPM**
    - POED w/ further scaled frequency using online slack
Energy Savings: POED vs. Standby-Sparing

Normalized energy consumption vs. static load; 20 tasks per set

HARSH-2013, Shenzhen, China
Energy Savings: POED vs. Standby-Sparing

Normalized energy consumption vs. dynamic load; 20 tasks per set

HARSH-2013, Shenzhen, China
Conclusions & Future Work

- POED-based EEFT for dual-processor systems

Objective
- co-management of energy with reliability

Results
- significant energy savings vs. standby-sparing

Future work
- Effects of additional DVFS transition
- Multiprocessor system with more than two processors
Thanks & Questions

http://www.my.cs.utsa.edu/~yguo