On Resilient Task Allocation and Scheduling with Uncertain Quality Checkers

Qian Zhang, Ting Wang and Qiang Xu

Department of Computer Science & Engineering
The Chinese University of Hong Kong
Outline

- Background
- Preliminaries
- The proposed methodology
  - Probability of quality satisfaction
  - Online scheduler
- Experiments
Emerging Error-Resilient Applications

- Noisy input
- Stochastic Processing
- “Acceptable” instead of precise output
Emerging Error-Resilient Applications

Applications have a mix of resilient and sensitive computations

83% of runtime spent in computations can be approximated

What is Approximate Computing?

- Approximate computing
  - A technique to tradeoff computation quality and computational effort (e.g., energy)
Approximate Computing

Key idea: Trade off computation quality and energy consumption (Unreliable hardware units may produce incorrect results with much lower power.)

- Voltage Over-Scaling
  - Circuits work below the nominal voltage for energy reduction
    - Error vs. Energy

- Resilience-Aware Scheduling
  - Not well explored
  - ApproxMap
Outline

- Background
- Preliminaries
  - *ApproxMap*: Resilience-aware scheduling
- The proposed methodology
  - Probability of quality satisfaction
  - Online scheduler
- Experiments
**ApproxMap: Resilience-Aware Scheduling on Multicore Platforms**

Resilient applications

Mapping and scheduling

How to treat error-resilient tasks and error-sensitive tasks differently for energy gains

How to ensure the target quality requirement, and to meet the application performance requirement

Note: Here we assume the processor cores are architecturally identical and the only source of heterogeneity is their operating voltage levels.
Architecture & Application Model

Data flow graph $G=\langle V, E, R \rangle$

Operating voltage $V = \{V_1, V_2, \ldots, V_K\}$, where $V_1 < V_2 < \cdots < V_K$.

$V_K$ is the nominal voltage, while the other voltage level could potentially impact the correctness of the computation.
ApproxMap

Design time

Resilience Identification → Error probability characterization

Offline scheduler

Initial schedule guaranteeing time constraint in the worst case scenario (ILP model)

Adjust the initial schedule to facilitate online energy saving (Simulated annealing-based algorithm)

Run-Time

Online scheduler

- Slack reclamation
- Voltage scaling set updating

1

2

3
ApproxMap: Offline Schedule

Slack window: resilient task complete before the worst case execution time
ApproxMap: Online Adjustment

\[ S_{ILP}^{1}: \{V_1, V_2, V_4\}; \]

Update its voltage set and execute immediately.
ApproxMap: Unsolved Issue I

- Runtime Quality Satisfaction Issue

Quality checker is unreliable!
(it is usually trained by a learning model and predict quality violation with $p\%$ accuracy.)
Quality Satisfaction

- Runtime Quality Satisfaction Issue
  - Quality satisfaction
    \[ E < TH \]
  
  - Probability of quality satisfaction
    \[ \max \text{ Prob.} (E < TH) \]
ApproxMap: Unsolved Issue II

- Detailed task-core adjustment & voltage set adjustment
  - single-core adjustment may cause the online time slack unusable.

$t_7$ cannot utilize any slack at runtime, because it has to wait for $t_6$, which cannot finish earlier on M2.

However, if assign task $t_6$ on M1 instead of M2, this problem can be solved.
Outline

- Background
- Preliminaries
- The proposed methodology
  - Probability of quality satisfaction
  - Online adjustment
- Experiments
The Proposed Methodology

- Selectively trust each intermediate checking result based on a probability procedure and the runtime situation to maximize the probability of quality satisfaction;
- Characterize voltage tuning table for each resilient task under different voltage levels by jointly considering computation quality and energy consumption;
- Enable multi-core resilient task adjustment.
**Probability of Quality Satisfaction**

- **Assume**
  - The probability of a quality checker can give a correct evaluation is $p$
  - The probability of us believing such a evaluation is $q$

- **Problem**
  - Find such a value/distribution of $q$ that satisfies
    $\max \ Prob. (E < TH)$
Given the initial schedule in design time, we say the system is in different states if it is running different tasks.
State Transfer

- Probability of state transfer
  - i.e., the probability of finishing current resilient task $t$

\[
P(Orange) = P(e \leq th)P(predict_{correct})P(adopt) + P (e \leq th)P(predict_{wrong})P(discard) + P (e > th)P(predict_{correct})P(adopt) + P (e > th)P(predict_{wrong})P(discard) + \]

- Problem

\[
\text{max } P(E < TH) \text{ w. r. t. } p, q, th
\]

- Then we can guarantee the computation quality with maximum probability by selectively believing the checking results based on $q$ and runtime situation.
Online Execution

Execution finished with voltage $V_1$

M_1

$\text{t}_1 \quad \text{t}_2 \quad \text{t}_4$

Resilient task

Sensitive task

Update voltage set $S_2$ and begin to execute $t_2$

Quality Check

State Transfer

State Untransfer

$S_1: \{V_1, V_2, V_3\}$
$S_2: \{V_2, V_3\}$
$S_4: \{V_3\}$

Re-run $t_1$ with $V_2$

Time Slack
Online scheduler

- Update voltage scaling set for task $t_i$ in PEST (potential energy saving tasks):

$$ slack = start_i - time_{current} $$

$$ total\_time = |S_i| + slack $$

$$ S_i = updateS(t_i, total\_time) $$
Online scheduler

- Update voltage scaling set for task $t_i$ in PEST (potential energy saving tasks):

$$slack = start_i - time_{current}$$

$$total\_time = |S_i| + slack$$

$$S_i = updateS(t_i, total\_time)$$

- $updateS(task, available\_time)$
  - Heuristic
    - Sorting the voltage levels by $\frac{potential\ energy\ efficiency}{quality\ degradation}$
    - Update $S_i$ by selecting voltages according to total $available\_time$
Outline

● Background
● Preliminaries and problem definition
● The proposed methodology
  ● Probability of quality satisfaction
  ● Online scheduler
● Experiments
Experimental Setup

- Initial schedule from ApproxMap
  - Gurobi 5.60 with CVX 2.1 in Matlab

- Representative task graphs
  - TGFF 3.5

- Voltage scalable system with 4 processors, and each processor has four operation voltages (1.69 V, 1.46 V, 1.38 V, 1.32 V)

- Variation of datasets
  - Take the mean value over 1000 runs for the same task graph
For a given quality requirement, increasing the portion of resilient tasks can bring benefits on energy savings.

For each case, lowering quality requirement benefits to energy efficiency.
Comparison with ApproxMap

- Probability of Quality Satisfaction (10% quality threshold)
  - Collect the “pass/fail” data over 1000 runs for each application with different resilient portions.

<table>
<thead>
<tr>
<th>Application</th>
<th>Fraction</th>
<th>ApproxMap</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>passed</td>
<td>prob.</td>
</tr>
<tr>
<td>creds1_0</td>
<td>30%</td>
<td>934</td>
<td>93.40%</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>930</td>
<td>93.00%</td>
</tr>
<tr>
<td></td>
<td>70%</td>
<td>901</td>
<td>90.10%</td>
</tr>
<tr>
<td>kbasic_task</td>
<td>30%</td>
<td>899</td>
<td>89.90%</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>810</td>
<td>81.00%</td>
</tr>
<tr>
<td></td>
<td>70%</td>
<td>649</td>
<td>64.90%</td>
</tr>
<tr>
<td>kbasic_parallel_xover</td>
<td>30%</td>
<td>719</td>
<td>71.90%</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>583</td>
<td>58.30%</td>
</tr>
<tr>
<td></td>
<td>70%</td>
<td>798</td>
<td>79.30%</td>
</tr>
<tr>
<td>kbasic_parallel</td>
<td>30%</td>
<td>962</td>
<td>96.20%</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>970</td>
<td>97.10%</td>
</tr>
<tr>
<td></td>
<td>70%</td>
<td>969</td>
<td>96.90%</td>
</tr>
</tbody>
</table>
Comparison with ApproxMap

- **Efficacy of Online Adjustment**
  - As we use the same offline scheduler of ApproxMap, the evaluation of our online procedure is presented by comparing energy consumptions with *ApproxMap*.
  - In terms of normalized energy consumption, wherein we set *ApproxMap* as 1 and error threshold as 10%.
Thank You for Your Attention!