Analyzing and Optimizing Energy Efficiency of Algorithms on DVS Systems:
--A First Step towards Algorithmic Energy Minimization--

TETSUO YOKOYAMA

Nagoya University

Joint work with Gang Zeng, Hiroyuki Tomiyama and Hiroaki Takada
Optimizing Energy Consumption at Algorithmic Level

Optimizing energy consumption:
• Many HW mechanisms available (DVFS, DPM, ...)
• Important in system level design
• Used in earlier stages of SW development phase

However, fundamental concepts have not yet been completed.
• Differences from performance optimization?
• Metrics?
• Programming logics and structure?
• Dataflow?

More precisely, we cannot answer:
Which, either Quicksort or Heapsort, is more energy optimal?
Target and Objective

• Target
  – DVS systems
  – Deadline constraints
  – Algorithmic level

• Objective
  – Clarify the difference between energy optimization and performance optimization
  – Propose a measure for energy consumption
  – Study a case of algorithmic energy optimization
  – Answer “Quicksort vs Heapsort. Which is more energy optimal?”
### IntraDVS: Basic Concepts

**Related work: Control flow graph**

- The selected branch decides the remaining cycles
  - Ex. Either block B or block C is executed

**Our approach: Data flow graph**

- Voltage/frequency can be reduced

- The sizes of divided subproblems decide the remaining cycles

Ex.1

- d
- \(d_1\)
- \(d_2\)
- 1Mcycle
- 9Mcycle
- 10Mcycle

Ex.2

- d
- \(d'_{1}\)
- \(d'_{2}\)
- 4Mcycle
- 4Mcycle
- 8Mcycle

Voltage/frequency can be reduced
Review of Quicksort

- Workload variation $\rightarrow$ slack time
  - WCEC: $\delta_w(n) \propto n^2$
  - ACEC: $\delta_a(n) \propto n \log n$
- Problem: WCEC is too big!
  $\rightarrow$ Heapsort does not have much workload variance.

The smaller problems are firstly processed.
Remaining Predicted Execution Cycles

- **WCEC**

\[ \delta_w(d) = c(d) + \max_{d_1,d_2 \in \text{div}(d)} (\delta_w(d_1) + \delta_w(d_2)) \]

\[ \delta_w(d): \text{WCEC of processing } d \]

\[ c(d) + \max(\delta_w(d_1) + \delta_w(d_2)) \]

Dividing time + Worst of sum of WCECs
Comparison of Remaining WCET of qsort

- Remaining WCET w/o concatenation:
  \[ \sum_{i=1}^{m} \delta_{wc}(\text{length}(xS_i)) \] (Approximation)

- In ordinary libraries, the smaller problem is firstly processed
  - **PRO**: Memory consumption is bound by \(O(\log n)\)
  - **CON**: Remaining WCET does not rapidly decrease (early division technique)

<table>
<thead>
<tr>
<th></th>
<th>Decrease of RET</th>
<th>Memory Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smallest</td>
<td>Slow</td>
<td>Low</td>
</tr>
<tr>
<td>Largest</td>
<td>Moderate</td>
<td>Middle</td>
</tr>
<tr>
<td>Larger</td>
<td>Rapid</td>
<td>High</td>
</tr>
</tbody>
</table>

Our approach: Processing larger problems
Energy-efficient Hybrid Sorting Algorithm

- Energy Efficiency = Original Algo. + Optimization

<table>
<thead>
<tr>
<th></th>
<th>Original Algo.</th>
<th>Energy Optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quicksort</td>
<td>$O(n^2)$</td>
<td>Very good</td>
</tr>
<tr>
<td>Heapsort</td>
<td>$\Theta(n \log n)$</td>
<td>Bad</td>
</tr>
<tr>
<td>Hybrid</td>
<td>$\Theta(n \log n)$</td>
<td>Very good</td>
</tr>
</tbody>
</table>

- Our solution: Hybrid sort = Quicksort + Heapsort
  - First, Quicksort: fast on average
    - Early division technique is used.
    - Performance is very high for almost all of the input.
  - At worst case, changed into Heapsort; WCEC is bounded.
  $\Rightarrow$ The energy efficiency of the sorting algorithm can be optimized on DVS systems.
The data was obtained on a MIPS R5000 processor with 512 KB of secondary cache and 64 MB of main memory, using version 7.2.1 of the Silicon Graphics MIPSpro C++ compiler.
Normalized Energy of Sorting Programs Using DVS

62.6%
Effects of the Stack Size on Energy Consumption Using SVS
Effects of the stack size on energy consumption

• A tradeoff between energy and memory
• The greater stack size it has, the more energy is saved.
• Energy savings saturated to about 25%
• This saturation occurs at line $y = 3000x^{0.85}$
Measure for Evaluating Energy Consumption of Algorithm

- SVS: voltage scheduling before execution
- The optimal voltage of each task
  = The minimum voltage to finish the task execution exactly at the deadline

**Lemma** (Optimal SVS) For SVS without scaling bound, an algorithm of a given task is optimal on average iff
\[ \delta_w^2 \delta_a \]
is minimum.

**Implications:**
- Deadline does not affect the comparison of two algorithms
- Firstly, WCET \( \delta_w \) should be reduced
- Secondly, ACET \( \delta_a \)
Proof

1. Let D be deadline.
2. The execution of frequency $\delta_w/D$ finishes exactly on deadline.
3. The corresponding power consumption is $P(\delta_w/D)$, which is the smallest in the case that $f = \delta_w/D$ because of the monotonicity of $P$.
4. The execution time is the number of cycles $X$ divided by frequency, i.e., $X/(\delta_w/D)$.
5. Thus, energy consumption becomes $P(\delta_w/D) \cdot DX/\delta_w$.
6. Now, we are comparing the different algorithms under the same deadline, and therefore $D$ is constant.
7. The average energy consumption is proportional to the average of $(\delta_w)^2X$.

CAVEAT: If frequencies range over $[f_{\text{min}}, f_{\text{max}}]$, the objective function becomes $(\delta_w)^2 \cdot \max(Df_{\text{min}}, \min(Df_{\text{max}}, \delta_a))$.
Normalized Energy of Sorting Programs Using SVS
Comparison of Energy Consumption of Sorting: A Case Study

• Question:
  – What is an energy efficient sorting on SVS systems?

• Our Answer†:
  – When the input size is small,
    Energy optimization ≡ Performance optimization
  – When the input size is large,

<table>
<thead>
<tr>
<th>Average Energy</th>
<th>Hybrid &lt;&lt; Heap &lt;&lt; Quick</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Exec. Time</td>
<td>Quick = Hybrid &lt;&lt; Heap</td>
</tr>
<tr>
<td>Worst Energy</td>
<td>Heap &lt; Hybrid &lt;&lt; Quick</td>
</tr>
<tr>
<td>Worst Exec. Time</td>
<td>Heap &lt; Hybrid &lt;&lt;&lt; Quick</td>
</tr>
</tbody>
</table>

• Implications:
  – The proposed metrics enables this comparison
  – Energy optimization ≠ Performance optimization
  – Fastest on average ≠ Energy optimal on average

†Only consider Quicksort, Heapsort and Hybrid sort
Concluding Remarks

• Algorithmic energy efficiency is meaningful.

• We propose
  – Energy efficient sorting algorithm
  – A measure for evaluating the optimal energy of algorithms
  – IntraDVS strategies using data flow information

⇒ We can discuss which, either Quicksort or Heapsort, is more energy efficient.

• Future work
  – How to write energy efficient programs (partially published)
  – Compare energy efficiency of other algorithms (ongoing)
  – Expose a tradeoff with energy (ongoing)