Energy-Aware Programming Utilizing the SEEP Framework and Symbolic Execution

Timo Höning, Christopher Eibel, Rüdiger Kapitza, and Wolfgang Schröder-Preikschat

Introduction and Motivation

Resource-constrained devices, such as wireless sensor nodes, smart phones, laptops, and tablet computers most notably suffer from the limited amount of energy they have available.

<table>
<thead>
<tr>
<th>Year</th>
<th>RAM</th>
<th>Storage</th>
<th>Network</th>
<th>Battery Life</th>
</tr>
</thead>
<tbody>
<tr>
<td>1981</td>
<td>4 Kbyte</td>
<td>360 Kbyte</td>
<td>300 bps</td>
<td>1 h</td>
</tr>
<tr>
<td>1991</td>
<td>2 Mbyte</td>
<td>80 Mbyte</td>
<td>56 Kbps</td>
<td>2 h</td>
</tr>
<tr>
<td>2001</td>
<td>256 Mbyte</td>
<td>20 Gbyte</td>
<td>100 Mbps</td>
<td>4 h</td>
</tr>
<tr>
<td>2011</td>
<td>4 Gbyte</td>
<td>500 Gbyte</td>
<td>1 Gbps</td>
<td>10 h</td>
</tr>
</tbody>
</table>

Emerging battery technologies addressing this issue were unable to improve the situation significantly. While storage and transfer rates have seen growth rates up to factor 10^2, battery life could merely be improved by a factor of 10^3.

Research efforts led to energy-aware system design, however, writing energy-efficient programs in the first place has only received limited attention.

The SEEP Framework

SEEP is a three-tier framework exploiting symbolic execution and platform-specific energy profiles. It is composed of three main components which are consecutively executed.

Architectures: Three-Tier Framework

- Source Code
- Symbolic Execution
- Predefined Code
- Code Paths
- Constraints
- Basic Blocks
- Trace Counters
- Cross Compile
- Native Trace
- Path Explorer
- Complexity Explorer
- Profile Merger

SEEP executes the code under test symbolically to extract all code paths and their path constraints. On basis of these results, SEEP crafts program code with predefined input data, so-called path entities.

For each path entity, SEEP generates a runtime execution trace by executing them on a test system and increments a block counter for every basic block (branchless sequence of code) each time it is executed.

Energy Demand of a Basic Block

<table>
<thead>
<tr>
<th>Evaluation System</th>
<th>Basic Block (ARM)</th>
<th>Target System</th>
<th>Energy Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>movl 12(%ebp), %eax</td>
<td>13.316 mJ</td>
<td>13.316 mJ</td>
<td></td>
</tr>
<tr>
<td>movw 12(%ebp), %eax</td>
<td>13.483 mJ</td>
<td>13.483 mJ</td>
<td></td>
</tr>
<tr>
<td>testb %al, %al</td>
<td>1.572 mJ</td>
<td>1.572 mJ</td>
<td></td>
</tr>
<tr>
<td>addl $1, %eax</td>
<td>1.633 mJ</td>
<td>1.633 mJ</td>
<td></td>
</tr>
<tr>
<td>subw %eax, %eax</td>
<td>1.665 mJ</td>
<td>1.665 mJ</td>
<td></td>
</tr>
<tr>
<td>testb %eax, %eax</td>
<td>11.895 mJ</td>
<td>11.895 mJ</td>
<td></td>
</tr>
<tr>
<td>jae .L55</td>
<td>17.938 mJ</td>
<td>17.938 mJ</td>
<td></td>
</tr>
</tbody>
</table>

Each block’s energy demand is determined by accumulating the instructions’ energy costs and adding inter-instruction costs when necessary. The basic block counters are used to transfer the gathered knowledge about the runtime behaviour of the code from the test platform to the actual target platforms. Adding up these interim results yields the energy demand for each path entity.

The SEEP Approach

Today, developing energy-efficient applications is a backward-looking process. Disclosing non-functional software defects which affect battery life is a time-consuming task.

Status Quo: Backward-Looking Process

Defects are revealed by manual testing or after deployment. Challenges to fix these defects include runtime analysis of the program code, time-consuming series of energy measurements and today’s variety of target platforms.

Energy-Aware Programming using SEEP

With SEEP we turn the task of energy-aware programming into a forward-looking process. This allows design decisions based on the software’s energy demand at the earliest possible time during development.

Evaluation and Demo

We have designed a custom circuit board based on the concept of a current mirror. Two capacitors under the control of a flip-flop are being charged and discharged on an alternating basis and the flip-flop’s switching events are counted.

Measurement Device: Schematics and Circuit Board

To verify SEEP’s energy estimates we have evaluated several path entities of an application with three different code paths. The energy estimates varied by 0.089 mJ at the maximum with an average deviation of 0.017 mJ.

Status and Future Work

SEEP provides accurate estimations for the base energy demand of program code, even for unavailable platforms. This helps developers to identify and address energy hot spots early during development.

Energy-Aware Programming with SEEP: A Forward-Looking Process

Currently, we are extending the framework to support local I/O and energy costs for network links. Also, we are exploring ways to model non-deterministic factors (e.g., memory access) in order to support complex scenarios.