Hybrid Approach to System-Level Performance Analysis

Simon Künzli
Bologna, May 22, 2006

contains joint work with Francesco Poletti,
Luca Benini, and Lothar Thiele

Design Space Exploration

• need for fast performance evaluation methods

• interest in non-functional properties, e.g. timing behavior, memory requirement
Formal Methods
• possibilities to answer questions limited by method
• good coverage (worst case)
• fast
• coarse

Simulation
• can answer virtually any questions about
  performance
• can model arbitrary complex systems
• average case (single instance)
• time-consuming
• detailed

Design Evaluation

Combination
• Use hybrid approach for analysis

⇒ Compose existing performance analysis models
Combination II

- Interface definition between analysis domains
- Applicability shown using case study for multi-processor system

Goals

- Generated trace should be:
  - consistent with specification curves
  - representative for short term characteristics (bursts)
  - representative for long term characteristics (average case)

- These properties should be observed anywhere in the generated trace
Problems for generation

Proposed trace generation algorithm handles these problems and generates valid traces.

/* generate event at time t */
generateEvent(t);
while (!stopGeneration) {
    while ( t < swt ) {
        if (state == 0) {
            if ( canIGenerateNow(t) )
                generate = true;
        } else{
            if ( !canIStillWait(t) )
                generate = true;
        }
        if (generate) {
            generateEvent(t);
            updateHistoryWithEvent(t);
        }
        t = t + timeStep;
        generate = false;
    }
    swt = getNextSwitchingTime(t);
    state = (state + 1) mod 2;
}

/* initialize variables */
t = 0;
generate = false;
state = 0;
swt = getNextSwitchingTime(t);
Goals (revised)

• Generated trace should be:
  – consistent with specification curves
  – representative for short term characteristics (bursts)
  – representative for long term characteristics (average case)

• These properties should be observed anywhere in the generated trace

New quality indicator to measure these properties

Quality indicator (I)

1. Select all trace snippets $T_i$ of length $\tau$ in trace $T$.

2. Compute the upper and lower curve $[\alpha^l_c, \alpha^u_c]$ from each trace snippet $T_i$. 
Quality Indicator (II)

3. Set $Z(T_i) = 1$, if $\alpha_c^u(\Delta) = \alpha_c^u(\Delta)$ and
$\alpha_c^l(\Delta) = \alpha_l^l(\Delta)$, for all $0 \leq \Delta \leq \frac{\tau}{2}$
and $Z(T_i) = 0$, otherwise.

\[ \text{specification} \quad = \quad \text{derived from trace} \]

Quality Indicator (III)

4. Compute $P_T = \frac{1}{N} \sum_{T_i \in T} Z(T_i)$
where $N$ denotes the number of considered trace snippets $T_i \in T$.

\[ T_1 T_2 T_3 \]
Quality Indicator (IV)

5. Set \( I = \min_{\forall \tau \leq L} P_\tau \)

\[ \tau \tau \tau \]

→ Measure for self-similarity of trace
→ The larger \( I \), the “better” the trace \( T \) represents the specification curves

How to determine Switching time?

• Deterministic algorithm leads to optimal indicator value (optimal under certain conditions)
  – Problem: randomized traces preferable for analysis

• Randomized version of deterministic algorithm
  → uniform distribution of switching times

• Weibull distribution [Anastasi’98],[Barford’98] used as control runs
Examples for Generated Traces

Experiments
<table>
<thead>
<tr>
<th>Nr.</th>
<th>Switching Time Determination</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Deterministic algorithm as presented in Sect. 3 up to window size $L$</td>
</tr>
<tr>
<td>2</td>
<td>Weibull-distributed with expectation $\frac{L}{2}$ and $\alpha = 0.5$</td>
</tr>
<tr>
<td>3</td>
<td>Weibull-distributed with expectation $\hat{L}$ and $\alpha = 0.5$</td>
</tr>
<tr>
<td>4</td>
<td>Weibull-distributed with expectation $2L$ and $\alpha = 0.5$</td>
</tr>
<tr>
<td>5</td>
<td>Weibull-distributed with expectation $3L$ and $\alpha = 0.5$</td>
</tr>
<tr>
<td>6</td>
<td>Weibull-distributed with expectation $\frac{L}{2}$ and $\alpha = 0.3$</td>
</tr>
<tr>
<td>7</td>
<td>Weibull-distributed with expectation $\hat{L}$ and $\alpha = 0.3$</td>
</tr>
<tr>
<td>8</td>
<td>Weibull-distributed with expectation $2L$ and $\alpha = 0.3$</td>
</tr>
<tr>
<td>9</td>
<td>Weibull-distributed with expectation $3L$ and $\alpha = 0.3$</td>
</tr>
<tr>
<td>10</td>
<td>Uniformly distributed with expectation $\frac{L}{2}$</td>
</tr>
<tr>
<td>11</td>
<td>Uniformly distributed with expectation $\hat{L}$</td>
</tr>
<tr>
<td>12</td>
<td>Uniformly distributed with expectation $2L$</td>
</tr>
<tr>
<td>13</td>
<td>Uniformly distributed with expectation $3L$</td>
</tr>
</tbody>
</table>

**Comparison**

![Comparison chart](chart.png)
Case Study

save evaluation time
  • less simulation runs needed for good coverage
  • single simulation run is faster

shorter development times for evaluation models
  • use available models

→ suitable for design space exploration

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Evaluation Method</th>
<th>Evaluation Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Simulation</td>
<td>508</td>
</tr>
</tbody>
</table>

Conclusion

• Definition of interfaces needed for hybrid performance models

• Applicability shown using example

• Automated tool chain at hand for hybrid approach