Probabilistic Real-Time Systems

Liliana Cucu-Grosjean, INRIA Nancy-Grand Est
liliana.cucu@inria.fr

The results presented are joint work with PROARTIS partners
Outline

• **Probabilistic Real-Time Systems**
• **Probabilistic Worst-Case Execution Time**
  - PROARTIS: measurement based approach
• **Static probabilistic timing analysis**
• **Measurement based timing analysis**
• **Conclusion**
Probalistic Real-Time Systems

A periodic task $\tau_i = (O_i, C_i, T_i, D_i)$ is characterized by:
- release $O_i$
- (worst-case) execution time $C_i$
- (minimal) inter-arrival time $T_i$
- deadline $D_i$

$\tau_1 = (1, 2, 5, 4)$
One processor, preemptive fixed-priority

\[ \tau_i = (O_i, C_i, T_i, D_i) \]

\[ \tau_1 = (0,1,3,3) \]
\[ \tau_2 = (3,2,6,6) \]
\[ \tau_3 = (2,1,6,6) \]
One processor, preemptive fixed-priority

\[ \tau_i = (O_i, C_i, T_i, D_i) \]

\[ \tau_1 = (0, 1, 3, 3) \quad \tau_2 = (3, 2, 6, 6) \quad \tau_3 = (2, 1, 6, 6) \]
One processor, preemptive fixed-priority scheduling

\[ \tau_i = (O_i, C_i, T_i, D_i) \]

\[ \tau_1 = (0, 1, 3, 3) \quad \tau_2 = (3, 2, 6, 6) \quad \tau_3 = (2, 1, 6, 6) \]
One processor, preemptive fixed-priority

\[ \tau_i = (O_i, C_i, T_i, D_i) \]

\[ \tau_1 = (0, 1, 3, 3) \quad \tau_2 = (3, 2, 6, 6) \quad \tau_3 = (2, 1, 6, 6) \]
One processor, preemptive fixed-priority scheduling

\[ \tau_i = (O_i, C_i, T_i, D_i) \]

\[ \tau_1 = (0, 1, 3, 3) \quad \tau_2 = (3, 2, 6, 6) \quad \tau_3 = (2, 1, 6, 6) \]
Probabilistic real-time systems

\[ \tau_i = (O_i, C_i, T_i, D_i) \]

with the notation for the discrete case

\[ X = \left( P(X = x_k) \right) \]

- Uncertainties from the system and the environment
- Probabilistic guarantees \( P(\mathbb{R}_i > D_i) < \varepsilon \)
What a schedule looks like?

\[ \tau_1 = (0, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, 3, 3) \]
\[ \tau_2 = (3, \begin{pmatrix} 2 & 3 & 4 \\ 0.50 & 10.4 \end{pmatrix}, 6, 6) \]
\[ \tau_3 = (2, \begin{pmatrix} 1 & 2 \\ 0.50 & 5 \end{pmatrix}, 6, 6) \]
What a schedule looks like?

$\tau_1 = (0, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, 3, 3)$

$\tau_2 = (3, \begin{pmatrix} 2 & 3 & 4 \\ 0.50 & 10.4 \end{pmatrix}, 6, 6)$

$\tau_3 = (2, \begin{pmatrix} 1 \\ 2 \\ 0.50 & 0.5 \end{pmatrix}, 6, 6)$

$(O_i, C_i, T_i, D_i)$
What a schedule looks like?

\(\tau_1 = (0, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, 3, 3)\)

\(\tau_2 = (3, \begin{pmatrix} 2 & 3 & 4 \\ 0.50 & 10.4 \end{pmatrix}, 6, 6)\)

\(\tau_3 = (2, \begin{pmatrix} 1 & 2 \\ 0.50 & 5 \end{pmatrix}, 6, 6)\)

\((O_i, C_i, T_i, D_i)\)
What a schedule looks like?

\( \tau_1 = (0, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, 3, 3) \)  

\( \tau_2 = (3, \begin{pmatrix} 2 & 3 & 4 \\ 0.50 & 10.4 \end{pmatrix}, 6, 6) \)  

\( \tau_3 = (2, \begin{pmatrix} 1 & 2 \\ 0.50 & 0.5 \end{pmatrix}, 6, 6) \)
What a schedule looks like?

\[ \tau_1 = (0, \left( \begin{array}{c} 1 \\ 1 \end{array} \right), 3, 3) \quad \tau_2 = (3, \left( \begin{array}{c} 2 & 3 & 4 \\ 0.50 & 10.4 \end{array} \right), 6, 6) \quad \tau_3 = (2, \left( \begin{array}{c} 1 & 2 \\ 0.50 & 5 \end{array} \right), 6, 6) \]
What a schedule looks like?

\( \tau_1 = (0, \left( \begin{array}{c} 1 \\ 1 \end{array} \right), 3, 3) \) \quad \tau_2 = (3, \left( \begin{array}{c} 2 \\ 0.50.10.4 \end{array} \right), 6, 6) \quad \tau_3 = (2, \left( \begin{array}{c} 1 \\ 0.50.5 \end{array} \right), 6, 6) \)
Outline

- Probabilistic Real-Time Systems
- Probabilistic Worst-Case Execution Time
  - PROARTIS: measurement based approach
- Static probabilistic timing analysis
- Measurement based timing analysis
- Conclusion
PROARTIS: Probabilistically Analysable Real-Time Systems

http://proartis-project.eu/

The research leading to these results has received funding from the European Community’s Seventh Framework Programme [FP7/2007-2013] under grant agreement n° 249100.
What to Analyse

Platform Theme
WP1: Probabilistic Platforms
WP2: Hardware-Aware Software Design

Analysis Theme
WP3: Probabilistic Analysis and Tools

Validation Theme
WP4: Case Studies

Which Requirements

How to Analyse
Outline

- Probabilistic Real-Time Systems
- Probabilistic Worst-Case Execution Time
  - PROARTIS: measurement based approach
- Static probabilistic timing analysis
- Measurement based timing analysis
- Conclusion
SPTA*

- **Static Probabilistic Timing Analysis**
- The \( pWCET \) estimate is obtained by convolving the ETPs of each instruction

\[
\{(2, 3, 8, 11), (0.2, 0.7, 0.05, 0.05)\}
\]

\[
\begin{array}{cccc}
2 & 3 & 8 & 11 \\
0.2 & 0.7 & 0.05 & 0.05 \\
\end{array}
\]

*ACM TECS 2012 – « Probabilistically analysable real-time systems»
SPTA/2: convolution and requirements

- **Static Probabilistic Timing Analysis**

\[
\begin{pmatrix}
1 & 2 \\
0.7 & 0.3
\end{pmatrix} \times \begin{pmatrix}
7 \\
1
\end{pmatrix} = \begin{pmatrix}
1+7 & 2+7 \\
1\cdot0.7 & 1\cdot0.3
\end{pmatrix} = \begin{pmatrix}
8 & 9 \\
0.7 & 0.3
\end{pmatrix}
\]

- **Requirements of SPTA**
  - The convolution requires independent random variables
  - The ETPs must be defined by a single distribution

*Independent and identically distributed random variables*
I.i.d. random variables

- **Independent random variables**
  - Random variables that describe events which are not related
  - “Event A: my laptop died” and “Event B: the beamer does not like my laptop”

- **Identically distributed random variables**
  - Random variables that have the same distribution function
  - “Event A: arrival of a client in a bank” and “Event B: arrival of a car at a gas station”

- **Both properties checked through statistical tests**
Required properties: Independence

- **Data must be independent**
  - Hypothesis testing with *Wald-Wolfowitz* test
    - A parameter $P$ is obtained to quantify how often consecutive execution times are higher/lower than the median
    - If the data are random, $P$ should fall in the central part of a Gaussian distribution
      - Threshold (0.05) is chosen based on experience/common practice. $P$ should fall in the central 95% part of the distribution

![P should be in this range](image)
Required properties: Identically distributed

- **Data must be identically distributed**
  - Hypothesis testing with *Kolmogorov-Smirnov* test
    - Two subsets of data randomly obtained from the trace
    - Distributions compared. A parameter $P$ obtained based on the maximum distance among both data sets
    - If $P$ is above a given threshold, data are identically distributed
      - Threshold (0.05) is chosen based on experience/common practice
SPTA: i.i.d. random variables

- **Static Probabilistic Timing Analysis**
- **Ensured at the level of processor instructions**
- **Example: randomisation at cache level, evict on access**
  - Pathological eviction patterns are bounded pessimistically (N is the number of cache entries, K is the number of memory accesses between two consecutive accesses to the same cache entry)

\[
P(\text{hit}) = \begin{cases} \left(\frac{N-(K-1)}{N-(K-1)}\right)^K & \text{if } K < N \\ 0 & \text{if } K \geq N \end{cases}
\]

- **Some numbers**
  - i for instruction cache, d for data cache

0: \{(2, i_0 \times d_0), (i_0 \times d_0 + i_0 \times d_0), (i_0 \times d_0)\}
1: \{(2, i_1), (i_0)\}
2: \{(2, i_2 \times d_2), (i_2 \times d_2 + i_2 \times d_2), (i_2 \times d_2)\}
Outline

• Probabilistic Real-Time Systems
• Probabilistic Worst-Case Execution Time
  ➢ PROARTIS: measurement based approach
• Static probabilistic timing analysis
• Measurement based timing analysis
• Conclusion
Measurement Based Probabilistic Timing Analysis

The pWCET estimate is obtained by applying Extreme Value Theory

Let \( \{X_1, X_2, \ldots, X_n\} \) be a sequence of i.i.d. random variables and let \( M_n = \max\{X_1, X_2, \ldots, X_n\} \). If \( F \) is a non degenerate distribution function and there exists a sequence of pairs of real numbers \((a_n, b_n)\) such that \( a_n \geq 0 \) and \( \lim_{n \to \infty} P\left( \frac{M_n - b_n}{a_n} \leq x \right) = F(x) \), then \( F \) belongs to either the Gumbel, the Frechet or the Weibull family.

*ECRTS 2012 – « Measurement based probabilistic timing analysis for multi-path programs»
Measurement Based Probabilistic Timing Analysis

First sound utilisation of EVT to pWCET estimating

- The hypothesis of independence and identical distribution is checked before any EVT utilisation
  - I.i.d. hypothesis is ensured at the level of processor instructions
- Block maxima applied with a proper definition of minimum number of runs

- Utilisation of a (proven) correct statistical test to check that data belong to the Gumbel domain
Required properties: Gumbel

- **Data must fit a Gumbel distribution**
  - Hypothesis testing with *exponential tail* test
    - A parameter $P$ is obtained for the actual data
    - A confidence interval $CI$ is obtained for the actual data assuming it fits a Gumbel distribution
    - If $P$ belongs to $CI$ then the data fit a Gumbel distribution

Execution times:

```
6372
3728
6321
8328
8231
9827
...
```

Parameter $P$ + Gumbel

Confidence interval $CI$
Steps of applying EVT (single-path programs)
- Observations
- Grouping
- Fitting
- Comparison
- Tail extension

Convergence
- Continuous rank probability score

\[ CRPS = \sum_{i=0}^{+\infty} [f_X(i) - f_Y(i)]^2 \]
Measurement Based Probabilistic Timing Analysis

Multi-path programs

- Independent and identical distributions
  - Identical distributions ensured by the convergence process

- Minimum number of observations
  - Each path must be observed a minimum number of times and it is ensured by the convergence process

- Path coverage
  - pWCET estimate is obtained for the observed paths
Conclusions and future work

• **SPTA offers a tight pWCET estimate, but impractical for complex programs**

• **MBPTA is a sound utilisation of EVT, but limited by the path coverage**
  ➢ Proof of the limited pessimism of EVT application

• *Proposition of hybrid approaches*

• **Extending the probabilistic techniques proposed for the single-core case to the multi-core case**
  ➢ Identify the existing probabilistic techniques for multi-core
  ➢ Ensure the requirements of the PROARTIS single-core
  ➢ Modify accordingly the probabilistic techniques for the multi-core