Refinement-based Exact Response-Time Analysis

Martin Stigge

Uppsala University, Sweden

Joint work with Nan Guan and Wang Yi
Response-Time Analysis

- Useful for
  - Schedulability analysis
  - Jitters in larger systems
  - ...

- Standard RTA for static priorities + periodic/sporadic tasks

\[ R_j = C_j + \sum_{i \in hp(j)} \left\lceil \frac{R_j}{T_i} \right\rceil C_i \]
Not everything is periodic!
The Digraph Real-Time (DRT) Task Model
(S. et al., RTAS 2011)

- Generalizes periodic, sporadic, GMF, RRT, …
- **Directed graph** for each task
  - Vertices $v$: jobs to be released (with WCET and deadline)
  - Edges $(u, v)$: minimum inter-release delays $p(u, v)$
DRT: Semantics

Path $\pi = (v_4)$

Path $\pi = (v_4, v_2)$

Path $\pi = (v_4, v_2, v_3)$
DRT: Semantics

Path $\pi = (v_4)$

$\langle 2, 5 \rangle$ $\langle 1, 8 \rangle$ $\langle 1, 5 \rangle$

$\langle 3, 8 \rangle$ $\langle 5, 10 \rangle$

Martin Stigge

Refinement-based Response-Time Analysis
DRT: Semantics

Path $\pi = (v_4, v_2)$
Path $\pi = (v_4, v_2, v_3)$
Response-Time Analysis for DRT
Problem: Path Combinations

Combinatorial Explosion!

Response time

Response time
Problem: Path Combinations

Combinatorial Explosion!
Fahrplan
Fahrplan
Step 1: From Paths to Functions
Step 1: From Paths to Functions

\[ v_1 \langle 2, 5 \rangle \rightarrow v_2 \langle 1, 8 \rangle \rightarrow v_3 \langle 3, 8 \rangle \]
\[ v_1 \langle 2, 5 \rangle \rightarrow v_2 \langle 1, 8 \rangle \rightarrow v_4 \langle 5, 10 \rangle \]
\[ v_1 \langle 2, 5 \rangle \rightarrow v_5 \langle 1, 5 \rangle \]

\[ rf(t) \]

\[ rf(v_4, v_2, v_3) \]
Request Functions

Useful for deriving response time:

\[ R_{SP}(v, \bar{rf}) = \min \left\{ t \geq 0 \mid e(v) + \sum_{T' > T} rf(T')(t) \leq t \right\} \]

\[ R_{SP}(v) = \max_{rf \in RF(\tau)} R_{SP}(v, \bar{rf}) \]
Request Functions

Useful for deriving response time:

\[ R_{SP}(v, \bar{rf}) = \min \left\{ t \geq 0 \mid e(v) + \sum_{T' > T} rf(T')(t) \leq t \right\} \]

\[ R_{SP}(v) = \max_{\bar{rf} \in RF(\tau)} R_{SP}(v, \bar{rf}) \]

Combinatorial Explosion?!
Step 2: Abstraction Trees
Abstract Request Functions

\[ rf(t) \]

\[ rf(v_4, v_2, v_3) \]

Refinement-based Response-Time Analysis
Abstract Request Functions

\[ rf(t) \]

\[ rf(v_4, v_2, v_3) \]

\[ rf(v_5, v_4, v_2) \]
Abstract Request Functions

\[ rf(t) \]

Martin Stigge
Refinement-based Response-Time Analysis
Abstraction Tree

Define an *abstraction tree* per task:

- Leaves are concrete $rf$
- Each node: maximum function of child nodes
- Root is maximum of *all* $rf$
Define an *abstraction tree* per task:

- Leaves are concrete *rf*
- Each node: maximum function of child nodes
- Root is maximum of all *rf*
Define an *abstraction tree* per task:

- Leaves are concrete *rf*
- Each node: maximum function of child nodes
- Root is maximum of *all* *rf*
Define an *abstraction tree* per task:

- Leaves are concrete $rf$
- Each node: maximum function of child nodes
- Root is maximum of all $rf$
Define an *abstraction tree* per task:

- Leaves are concrete \( rf \)
- Each node: maximum function of child nodes
- Root is maximum of *all* \( rf \)
Define an *abstraction tree* per task:

- Leaves are concrete $rf$
- Each node: maximum function of child nodes
- Root is maximum of *all* $rf$

Allows stepwise refinement!
Step 3: Refinement Algorithm
Step 3: Refinement Algorithm

Tuple: \( \overline{rf} = (rf(T_1), rf(T_2), rf(T_3)) \)
Step 3: Refinement Algorithm

Tuple: \[
\bar{r}_f = (rf(T_1), rf(T_2), rf(T_3))
\]

Response time: \[
R_{SP}(v, \bar{r}_f) = 23
\]

Using: \[
R_{SP}(v, \bar{r}_f) = \min \left\{ t \geq 0 \mid e(v) + \sum_{T' > T} rf(T')(t) \leq t \right\}
\]
Step 3: Refinement Algorithm

Store

(23, \bar{f}_1)
Step 3: Refinement Algorithm

Step:

\[ \bar{rf}_1 = (rf(T_1), rf(T_2), rf(T_3)) \]

\[ \downarrow \]

\[ \bar{rf}_2 = (rf'(T_1), rf(T_2), rf(T_3)) \]

\[ \bar{rf}_3 = (rf''(T_1), rf(T_2), rf(T_3)) \]

In \( T_1 \):

\[ rf' \]

\[ rf'' \]

Store

(23, \( \bar{rf}_1 \))
Step 3: Refinement Algorithm

Step:

\[
\bar{rf}_1 = (rf(T_1), rf(T_2), rf(T_3))
\]

\[
\downarrow
\]

\[
\bar{rf}_2 = (rf'(T_1), rf(T_2), rf(T_3)) \rightarrow 18
\]

\[
\bar{rf}_3 = (rf''(T_1), rf(T_2), rf(T_3)) \rightarrow 21
\]

In \( T_1 \):

[Diagram of a tree showing \( rf' \) and \( rf'' \) as children of \( rf \)]

Store

(23, \( \bar{rf}_1 \))
Step 3: Refinement Algorithm

Step:

\[ \bar{rf}_1 = (rf(T_1), rf(T_2), rf(T_3)) \]

\[ \downarrow \]

\[ \bar{rf}_2 = (rf'(T_1), rf(T_2), rf(T_3)) \rightarrow 18 \]

\[ \bar{rf}_3 = (rf''(T_1), rf(T_2), rf(T_3)) \rightarrow 21 \]

In \( T_1 \):

Store

- (23, \( \bar{rf}_1 \))
- (21, \( \bar{rf}_2 \))
- (18, \( \bar{rf}_3 \))

Using:

\[ R_{SP}(v, \bar{rf}) = \min \{ t \geq 0 | e(v) + \sum_{T'} T' > T_{rf}(T') \leq t \} \]
Step 3: Refinement Algorithm

Store

(21, $\bar{rf}_2$)

(18, $\bar{rf}_3$)
Step 3: Refinement Algorithm

Step:

\[ \bar{rf}_2 = (rf(T_1), rf(T_2), rf(T_3)) \]

Store

(21, \bar{rf}_2)

(18, \bar{rf}_3)
Step 3: Refinement Algorithm

Step:

\[
\bar{r}_f_2 = (r_f(T_1), r_f(T_2), r_f(T_3))
\]

\[
\downarrow
\]

\[
\bar{r}_f_4 = (r_f(T_1), r'_f(T_2), r_f(T_3))
\]

\[
\bar{r}_f_5 = (r_f(T_1), r''_f(T_2), r_f(T_3))
\]

In \( T_2 \):

Store

(21, \bar{r}_f_2)

(18, \bar{r}_f_3)
Step 3: Refinement Algorithm

In $T_2$:

1. $\bar{rf}_2 = (rf(T_1), rf(T_2), rf(T_3))$
2. $\bar{rf}_4 = (rf(T_1), rf'(T_2), rf(T_3)) \rightarrow 20$
3. $\bar{rf}_5 = (rf(T_1), rf''(T_2), rf(T_3)) \rightarrow 17$

Store:

- $(21, \bar{rf}_2)$
- $(18, \bar{rf}_3)$
Step 3: Refinement Algorithm

Step:

\[ \tilde{r}_f_2 = (rf(T_1), rf(T_2), rf(T_3)) \]

\[ \downarrow \]

\[ \tilde{r}_f_4 = (rf(T_1), rf'(T_2), rf(T_3)) \rightarrow 20 \]

\[ \tilde{r}_f_5 = (rf(T_1), rf''(T_2), rf(T_3)) \rightarrow 17 \]

In \( T_2 \):

Store 
- \( (21, \tilde{r}_f_2) \)
- \( (20, \tilde{r}_f_4) \)
- \( (18, \tilde{r}_f_3) \)
- \( (17, \tilde{r}_f_5) \)
Step 3: Refinement Algorithm

Store

(20, \(\bar{r}f_4\))

(18, \(\bar{r}f_3\))

(17, \(\bar{r}f_5\))

\ldots

Using:

\[ R_{SP}(v, \bar{r}f) = \min \{ t \geq 0 | e(v) + \sum_{t' < T_{rf}(T')} t \leq t \} \]

...
Step 3: Refinement Algorithm

Initialization:
• Most abstract functions

Each iteration:
• Replace functions along *abstraction trees*

Termination:
• All functions are *concrete*
Step 3: Refinement Algorithm

Initialization:
  • Most abstract functions

Each iteration:
  • Replace functions along *abstraction trees*

Termination:
  • All functions are *concrete*

Store

(20, $\bar{r}f_4$)
(18, $\bar{r}f_3$)
(17, $\bar{r}f_5$)

Pluggable Path Abstractions!
Step 3: Refinement Algorithm

Initialization:
- Most abstract functions

Each iteration:
- Replace functions along *abstraction trees*

Termination:
- All functions are *concrete*

Pluggable Path Abstractions!
Path Abstractions: SP + EDF
Path Abstractions: Static Priorities

\[ rf_{\pi}(t) := \max \{ e(\pi') | \pi' \text{ is prefix of } \pi \text{ and } p(\pi') < t \} \]
Path Abstractions: EDF

\[ p(\pi) < t \text{ and } d(\pi) \leq t' \]
Path Abstractions: EDF

\[ wf_{\pi}(t, t') := \max\{e(\pi') \mid \pi' \text{ is prefix of } \pi, \]
\[ p(\pi') < t \text{ and } d(\pi') \leq t' \}. \]
Evaluation
Evaluation: Run-time Scaling

10-20 tasks with 5-10 vertices each, branching degree 1-3

(Busy window extension for EDF.)
Evaluation: Precision Improvement

Type A: lower parameter variance
Type B: higher parameter variance
Summary

- Exact solution for NP-hard problem
  - *Efficient* method
  - Iterative refinement
- Pluggable path abstractions
  - Static Priorities
  - EDF
  - *Flexible*

- Ongoing work:
  - Apply to other problems
Q & A

Thanks!