
AMALTHEA-based Task-Chain Analysis using Worst- and Best-Case Response Times

Release 1.0

Mar 15, 2020

Contents

1	under Partitioned Fixed-Priority Preemptive Scheduling	1
1.1	Chapters	1
1.1.1	Introduction	1
1.1.2	Metrics for Timing Analysis	2
1.1.3	Implementation	2
1.1.4	Measurement and Discussion	2
1.1.5	Conclusion and Future Work	2
1.1.6	User Interface (APP4RTA)	2
1.1.7	Bibliography	8
1.1.8	Repositories	9
1.1.9	Contact	9

under Partitioned Fixed-Priority Preemptive Scheduling

Modern automotive software runs on a centralized, high-performance system with multiple heterogeneous CPUs, accelerators, and GPUs. Since it is important to achieve systems' real-time determinism for modern HPC applications, an application that can help developers analyze the response time with various constraint concepts in a heterogeneous platform is essential.

The purpose of this research project is to come up with an application that analyzes the worst- and best-case response time for each software component and E2E latency for understanding task chain behaviors under several constraint configurations. The application is developed on *Eclipse APP4MC 0.9.7* API platform which provides the whole design flow in the context of *AMALTHEA* toolchain.

The used approaches are introduced in *Metrics for Timing Analysis Chapter* and demonstrated in *Implementation Chapter*. *Measurement and Discussion Chapter* shows the measurement results in regard of response time and task chain latency under a certain set of constraint configuration with discussion. Moreover, the final conclusion and the issues which expand the project further are addressed in *Conclusion and Future Work Chapter*.

1.1 Chapters

1.1.1 Introduction

Motivation

With the transition from homogeneous to heterogeneous multi-core platforms, there is an increasing demand for software mapping that minimize the latency of a task chain to facilitate high computing performance.

While achieving the minimum task response time is a key to solving the challenge, satisfying the optimized response time for every task is hard to realize since the limited number of processing units and task deadlines are concerned.

Therefore, it is reasonable to analyze mapping models based on the sum of all tasks' response time and extract the model with the minimum sum value. Finding the optimized solution is taken care of by a meta-heuristic algorithm such as GA or SA. This leads to the need for establishing an application that has a built-in response-time calculation algorithm and works with the meta-heuristic which would make the entire developing circle quicker.

Furthermore, the response time results obtained through the meta-heuristic are used to measure E2E task chain latency to help determine how long it would take to execute a certain set of software instructions. E2E latency measurement depends on the measurement target, the communication model and the simulation scenario. The ultimate goal of this project is to devise E2E latency measurement methodologies in an analytical way, and analyze the results of the methodologies to determine how a specific set of constraints configurations affects E2E latency.

AMALTHEA

AMALTHEA is a project as well as its results with a purpose to provide AUTOSAR compatible environment for efficient multi-core system development which is publicly available open-source. The project results cover a comprehensive system model, trace model, and framework which provides support tools when needed. The [\[11\] WATERS Industrial Challenge](#) (-, 2019) model used in this project is also created with the AMALTHEA framework and can be utilized and manipulated on [\[3\] APP4MC](#) (-, 2020) version 0.9.7 platform.

1.1.2 Metrics for Timing Analysis

1.1.3 Implementation

1.1.4 Measurement and Discussion

1.1.5 Conclusion and Future Work

1.1.6 User Interface (APP4RTA)

Set Up

For analyzing response time & end-to-end event-chain latency

APP4RTA

AMALTHEA MODEL ChallengeModel_TCs.amxmi Search Amalthea

Task Name	PU Num		0: Denver	Response Time	1: Denver	Response Time	2: A57	Response Time	3: A57	Response Time
OS_Overhead	4	Default IA	Planner	13358534500	DASM	1302430000 p	Localization	392590097500	CANbus_poll	
Lidar_Grabber	1	Enter IA	PRE_Detectio	73565439500	Lidar_Grabber	18265272000			PRE_SFM_g	
DASM	1								PRE_Lane_d	
CANbus_polling	3	<input checked="" type="radio"/> Synchronous							PRE_Localiz	
EKF	4	<input type="radio"/> Asynchronous								
Planner	0	<input checked="" type="radio"/> Worst-Case								
PRE_SFM_gpu...	3	<input type="radio"/> Average-Case								
PRE_Localizati...	3	<input type="radio"/> Best-Case								
PRE_Lane_det...	3	Calculate								
PRE_Detection...	0	Reset								
SFM	6									
Localization	2	Schedulability								
Lane_detection	5	Scheduleable! :)								
Detection	6	Cumulated Memory-Access Cost								
		5361668000 ps								
		Cumulated Contention								
		24795710000 ps								
		Computation								
		635075050500 ps								
		Response Time Sum								
		665232428500 ps								

EVENT CHAIN MODEL CA-EK-P-DA Calculate Reset ☐ Direct ☒ Implicit

Direct & Implicit Communication Paradigm

WC Reaction	81302942000 ps
BC Reaction	16086298500 ps

Task Chain Age (Direct & Implicit)

WC Age	5250000000 ps
BC Age	4750000000 ps

Task Chain >

1: Core3 (A57)	CANbus_polling
2: Core4 (A57)	EKF
3: Core0 (Denver)	Planner
4: Core1 (Denver)	DASM

LET Communication Paradigm

WC Reaction	80000000000 ps
BC Reaction	45000000000 ps

Early Reaction (Direct & Implicit)

WC E-Rct	64791310000 ps
BC E-Rct	64011310000 ps

Data Age

Contained Labels	Worst-case Age	Best-case Age
Occupancy_grid_host	17900000000 ps	12100000000 ps
Vehicle_status_host	10200000000 ps	9800000000 ps
x_car_host	15780000000 ps	12100000000 ps
y_car_host	15780000000 ps	12100000000 ps
yaw_car_host	15780000000 ps	12100000000 ps
vel_car	15780000000 ps	12100000000 ps
yaw_rate	15780000000 ps	12100000000 ps
steer_objective	5250000000 ps	4750000000 ps
...	5250000000 ps	4750000000 ps
	17900000000 ps	12100000000 ps

Before executing the code, please install the Java GUI softwares.

- To install Java GUI softwares:

- Eclipse > Help
- Install New Software > Work with: Eclipse Repository (<http://download.eclipse.org/releases/oxygen>)
- General Purpose Tools > all click from Swing Designer to WindowBuilder XML Core (requires Eclipse WTP/WST)
- Next > Next > accept > Finish

Search Amalthea

APP4RTA

AMALTHEA MODEL

Task Name PU Num

1. Select an Amalthea Model

Response Time & Mapping Analysis

EVENT CHAIN MODEL

☐ Direct ☐ Implicit

Direct & Implicit Communication Paradigm Task Chain Age (Direct & Implicit) Task Chain >

WC Reaction WC Age

BC Reaction BC Age

LET Communication Paradigm Early Reaction (Direct & Implicit)

WC Reaction WC E-Rct

BC Reaction BC E-Rct

Data Age

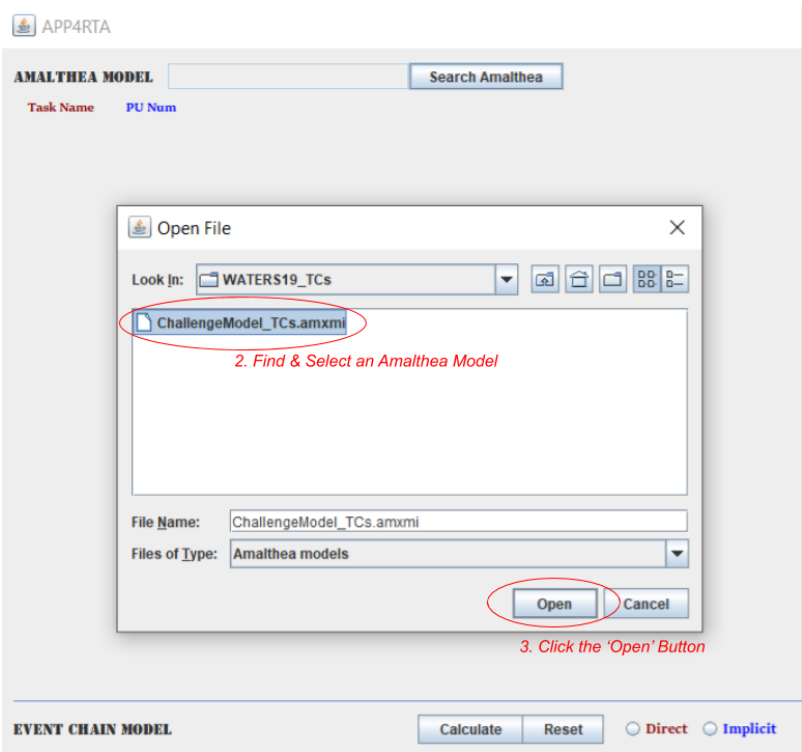
Contained Labels Worst-case Age Best-case Age

End-to-End Event-Chain Latency Part

Run APP4RTA.java in org.eclipse.app4mc.gsoc_rta.ui package, then this window will show up. Based on the horizontal line on the middle, the upper part is for response time & mapping analysis, and the lower part is for end-to-end event-chain latency analysis. The first thing to do is deciding a target Amalthea model.

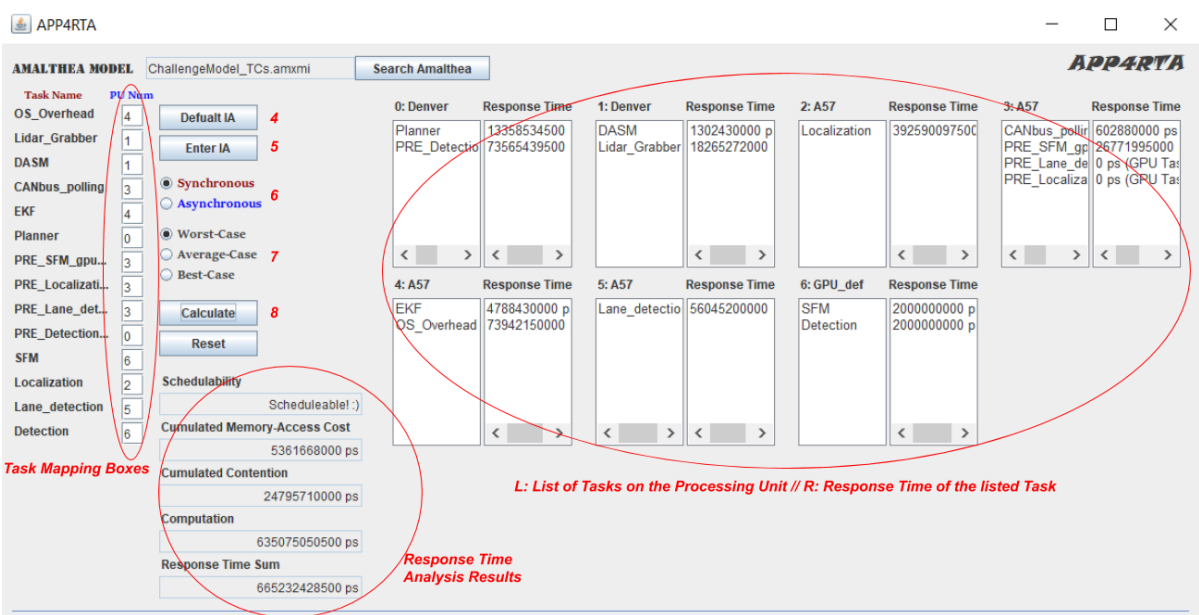
1. The window browser for searching Amalthea models shows up when the Search Amalthea button clicked.

Direct & Select Amalthea



2. When the search browser shows up, direct to the path where the target Amalthea model file is located and select the model file.
3. Click the Open button.

UI Features (RTA)



Then the empty space will be filled with the tasks and processing units of the selected model. On the left-hand side, tasks' names with empty boxes can be found. On the right-hand side, seven pairs of lists are seen (It means the selected model has seven processing units). The list on the left side of each pair is for listing names of the tasks which are mapped to the corresponding processing unit while one on the right side is for listing response times of the corresponding tasks. Basically, we can map the tasks with these boxes by entering the number of each processing unit which is stated on the top of the lists on the left-side.

4. The user can either manually type numbers for every box or simply click the `Default IA` button which would automatically fill up every box with the pre-defined integer array values.
5. Once every `PU Num` box is filled, click `Enter IA` button to assign tasks to processing units according to each integer value. Once this is done, the mapped tasks would appear on the left-side lists.
6. Choose the offloading mode between `Synchronous` case and `Asynchronous` case.
7. Choose the execution case between `Worst` case and `Average` case and `Best` case.
8. By clicking the `Calculate` button, all calculation results will be printed out on the text-fields (`Schedulability`, `Cumulated Memory-Access Cost`, `Cumulated Contention`, `Computation`).

For the implementation details, see CPURta-reference.

Select an Event-Chain

APP4RTA

AMALTHEA MODEL ChallengeModel_TCs.amxmi

Task Name	PU Num	Default IA	Enter IA	0: Denver	Response Time	1: Denver	Response Time	2: A57	Response Time	3: A57	Response Time
OS_Overhead	4			Planner	13358534500	DASM	1302430000 p	Localization	392590097500	CANbus_polling	
Lidar_Grabber	1			PRE_Detectio	73565439500	Lidar_Grabber	18265272000			PRE_SF	
DASM	1									PRE_Lane_d	
CANbus_polling	3	<input checked="" type="radio"/> Synchronous								PRE_Localiz	
EKF	4	<input type="radio"/> Asynchronous									
Planner	0	<input checked="" type="radio"/> Worst-Case									
PRE_SF	3	<input type="radio"/> Average-Case									
PRE_Localizati...	3	<input type="radio"/> Best-Case									
PRE_Lane_det...	3	<input type="button" value="Calculate"/>									
PRE_Detection...	0	<input type="button" value="Reset"/>									
SFM	6										
Localization	2	Schedulability									
Lane_detection	5	<input type="text" value="Scheduleable :)"/>									
Detection	6	Cumulated Memory-Access Cost									
		<input type="text" value="5361668000 ps"/>									
		Cumulated Contention									
		<input type="text" value="24795710000 ps"/>									
		Computation									
		<input type="text" value="635075050500 ps"/>									
		Response Time Sum									
		<input type="text" value="665232428500 ps"/>									

9. Choose an Event-Chain

EVENT CHAIN MODEL ☐ Direct ☐ Implicit

Direct & Implicit Com

WC Reaction

BC Reaction

LET Communication P

WC Reaction

BC Reaction

Task Chain Age (Direct & Implicit)

WC Age

BC Age

Task Chain >

Early Reaction (Direct & Implicit)

CE-Rct

BC E-Rct

Data Age

Contained Labels

Worst-case Age

Best-case Age

The event-chain combo-box becomes visible once the user clicks Enter IA to assign tasks to processing units according to each integer value in the boxes.

9. To analyze end-to-end event-chain latency, an event-chain in the combo-box should be selected first.

UI Features (E2ELatency)

EVENT CHAIN MODEL CA-EK-P-DA Calculate Reset ☐ Direct ☒ Implicit **9. Select an Offloading Mode**

10. Click the 'Calculate' Button.

Direct & Implicit Communication Paradigm

WC Reaction Task Chain Age (Direct & Implicit) WC Age

BC Reaction BC Age

LET Communication Paradigm

WC Reaction Early Reaction (Direct & Implicit) WC E-Rct

BC Reaction BC E-Rct

Data Age

Contained Labels	Worst-case Age	Best-case Age
Occupancy_grid_host	17900000000 ps	12100000000 ps
Vehicle_status_host	10200000000 ps	9800000000 ps
x_car_host	15780000000 ps	12100000000 ps
y_car_host	15780000000 ps	12100000000 ps
yaw_car_host	15780000000 ps	12100000000 ps
vel_car	15780000000 ps	12100000000 ps
yaw_rate	15780000000 ps	12100000000 ps
steer_objective	5250000000 ps	4750000000 ps
...	5250000000 ps	4750000000 ps
< >	17900000000 ps	12100000000 ps

Task Chain >

- 1: Core3 (A57)
- CANbus_polling
- 2: Core4 (A57)
- EKF
- 3: Core0 (Denver)
- Planner
- 4: Core1 (Denver)
- DASM

10. Select the communication paradigm between direct Communication and implicit communication.

11. Finally, click the Calculate button.

Then all calculation results regarding reaction, age of data, task-chain in the worst and best cases will be printed out to the corresponding text fields or lists.

For the implementation details, see E2ELatency-reference.

Download PDF file to see offline.

1.1.7 Bibliography

[1] (GSoC 2019) CPU-GPU Response Time and Mapping Analysis (Junhyung Ki, 2019)

[2] AMALTHEA (-, 2014)

[3] APP4MC (-, 2020)

[4] AUTOSAR Communication Paradigms (-, 2018)

[5] Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment (C. L. Liu, James W. Layland, 1973)

[6] Earliest Deadline First scheduling (-, -)

[7] Model-based Task Chain Latency and Blocking Analysis for Automotive Software (Robert Hötter, The Bao Bui, Junhyung Ki, Burkhard Igel and Olaf Spinczyk, Not yet published)

[8] CPU-GPU Response Time and Mapping Analysis for High-Performance Automotive Systems (Höttger, Robert and Ki, Junhyung and Bui, Thebao and Igel, Burkhard and Spinczyk, Olaf, 2019)

[9] Finding Response Times in a Real-Time System (M.Joseph, P.Pandya, 1986)

[10] Exact Best-Case Response Time Analysis of Fixed Priority Scheduled Tasks (Redell, Ola and Sanfridson, Martin, 2002)

[11] WATERS Industrial Challenge (-, 2019)

1.1.8 Repositories

1.1.9 Contact

Name: Junhyung Ki

Personal Email: kijoonh91@gmail.com

Student Email: junhyung.ki001@stud.fh-dortmund.de

[LinkedIn](#)