Development of A GUI Analysis Tool to Visualize Real-time Task Scheduling Simulation

Wakako Suzuki(1310706)

School of Information Science,
Japan Advanced Institute of Science and Technology

02/04/2017

Keywords: task, scheduling, real-time, simulation, deadline, Java.

1 Introduction

Nowadays real time tasks are executed on a daily basis in various embedded systems. Demand for real-timeness is increasing more and more, and a deadline miss can be fatal in some hard real-time systems. So, ensuring the schedulability of a task set in real-time system is becoming more and more important. To guarantee real-timeness, it is important to develop a real-time task scheduling algorithm which prevents deadline misses and minimizes response time. But the results of a task scheduling simulation are usually generated in the form of a text file with which we find it difficult to visually recognize the frequency of deadline misses and response times. This study aims at developing a GUI tool to visualize the result of a real-time task scheduling simulation to support development of real-time systems. This tool visualizes task execution, response time and deadline misses to make an analysis of the results of a simulation easier. Visualizing a result of a simulation not only helps us analyze schedulability or performance of a task set but also makes it easier to find problems or bugs of a scheduling algorhism.

2 Overview of The GUI Tool

This tool is developed with JavaFX 8 and supports multiple platforms such as Windows, Linux and Mac OS. The input of the GUI tool developed in this study is CSV file. Input data is delimited by commas and contains the time when each event occured, its task number, job number and the type of the event (request to release a task, start of an execution, end of an execution, deadline, deadline miss). As long as a task simulator outputs a result of its simulation in this format, the result can be visualized by this tool. This GUI tool visualizes the result as a chart in which vertical axis represents different tasks and horizontal axis represents time. Events are shown in the following methods:

- **request to release a task : up-arrow**
- **deadline of a task : down-arrow**
- **execution of a task : rectangle**
- **deadline miss : cross mark**

3 Development of The GUI Tool

The GUI tool is developed with JavaFX 8 which is a rich client platform to develop graphical user interfaces. JavaFX 8 is bundled with Java Runtime Environment(JRE) 8 or Java Development Kit(JDK) 8, so it is easy to prepare an environment to execute this tool. This chapter describes the functions of the tool and how they are implemented. This tool provides the following functions:

- **Choosing an input file from a file selection dialog**
- **Visualizing a simulation result as a chart**
- **Displaying partial chart in case the simulation contains data covering a long period of time**
- **Scrolling the chart**
- **Zooming in/out the chart**
- **Exporting the chart to an image file of a png format**
- **Searching/jumping to display a deadline miss if there is any**

This tool first shows a file selector to choose an input file and then creates lists of the events and any deadline misses. For this tool to display a chart, I customized GanttChart class provided in stackoverflow[4]. Scrolling, zooming in/out and exporting the chart to an image file are implemented using JavaFX fuctionalities. This chapter also describes some problems encountered during development and how they are solved. When I displayed a data covering a long period of time and zoomed in many times so that we can recognize all the events, two problems happened. One was a chart rendering issue in a virtualized environment and the other was unhandled exception thrown when exporting to an image file. These issues are caused by JavaFX attempt to use hardware graphics accerelation rendering by default. The solution was to force the use of pure-software rendering by adding a command line flag when starting the tool. Finally, the list of the files that construct the GUI tool and their sizes is shown.

4 Examples of Analyzing Actual Task Scheduling Simulation

I analyzed the results of an actual task simulation using the GUI tool developed in this study. Those task sets contain both periodical tasks and non-periodical tasks and the number of tasks is between 4 and 7, CPU usage is between 60% and 95% at an interval of 5%. There are both cases with deadline misses and without deadline misses. In this analysis, both periodical tasks and non-periodical tasks are visualized correctly even when a preemption occured because another task with higher priority needs to be executed. Zooming in and out and exporting the result to a png file worked fine. Searching deadline misses and jumping to display a deadline miss also worked fine. The GUI tool developed in this study was useful in analyzing the result of the task scheduling simulation by visualizing the task execution and ensuring the schedulability of task sets easily throughout the analysis.

5 Conclusion

In this study, I developed a GUI tool to visualize the result of real-time task scheduling simulators. The tool enables us to visualize the result of a task scheduling simulation as a chart, scrolling it, zooming in/out and exporting the chart to an image file to use in papers and articles. It makes it much easier to analyze the response time or the frequency of deadline misses and to ensure the schedulability of the task sets. The tool was useful in analyzing the result of actual task scheduling simulation. In the future, I improve this tool by fixing the chart's vertical axis so that task numbers are always displayed when scrolling horizontally.

References

[1] "Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications", 3rd Edition, Springer, 2011

[2] "Oracle JDK 8 and JRE 8 Certified System Configurations Contents", http://www.oracle.com/technetwork/java/javase/certconfig-2095354.html, November 9, 2016

[3] "Overview (JavaFX 8)", https://docs.oracle.com/javase/8/javafx/api/toc.htm, October 21, 2016

[4] "Gantt chart from scratch", http://stackoverflow.com/questions/27975898/gantt-chart-from-scratch, October 21, 2016

[5] "Zooming inside a Scrollpane", https://pixelduke.wordpress.com/2012/09/16/zooming-inside-a-scrollpane/, October 24, 2016

[6] "Node (JavaFX 8)", https://docs.oracle.com/javase/8/javafx/api/javafx/scene/Node.html#snapshot-javafx.util.Callback-javafx.scene.SnapshotParameters-javafx.scene.image.WritableImage-, October 24, 2016

[7] "Using the Image Ops API", http://docs.oracle.com/javafx/2/image_ops/jfxpub-image_ops.htm, October 24, 2016

[8] "How to generate chart image using JavaFX chart API for export without displying first", http://stackoverflow.com/questions/29721289/how-to-generate-chart-image-using-javafx-chart-api-for-export-without-displying, October 24, 2016

[9] "JavaFX: Getting Started with JavaFX", https://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-architecture.htm#A1106308, November 18, 2016