A project paper on

RMS Scheduling using Inchron Tool Suite

Submitted for the fulfilment of requirements of course

Advanced Real Time Systems

High Integrity Systems

Winter semester 2016-2017

Under the guidance of-

Prof. Dr. Jens Liebehenschel
Computer Science and Engineering
Frankfurt University of Applied Sciences

Submitted By-

Charu Sharma
Enrolment no. 1186114
Group 1

RMS Scheduling using Inchron Tool Suite

Charu Sharma
High Integrity Systems
Frankfurt University of Applied Sciences

Abstract
With dedicated machines taking over majority of tasks in production and other aspects, the demand of real time systems have escalated more than ever. With the degree of functionality being implemented, the complexity of such systems varies in proportion. Since for such systems hardware and software components are integrated very closely, it becomes difficult to articulate the functional points and erroneous regions posing risks with increase in complexity.
Inchron Tool Suite is a software that provides simulation of such real time systems with respect to processes and cores implemented, thus, providing an effective means to study the control architecture of a real time system before its deployment. The various features of the tool suite not only makes it feasible to frame complex real time hardware but also provides a glimpse of various computational details that are very essential in observing the functionality of the system under consideration.
With this paper, an in depth study of this tool is done and is used to observe a real time rate monotonous scheduling operation with interrupts, priority and core management.

***Index terms*- Rate monotonous scheduling, Inchron tool suite, priority, RMS.**

1. INTRODUCTION

A real time system is any data preparing framework which needs to react to remotely created input cues inside a limited and indicated time period. In case of real time systems the accuracy depends on the legitimate outcome as well as the time it was conveyed. The inability to react is just as harmful as the wrong reaction. Outlining real time frameworks is a challenging assignment. The majority of the challenge originates from the way that real time frameworks need to communicate with real world elements. These communications can get genuinely complicated.

What makes real time systems so demanding are the expectations which are put for them to be fulfilled in real time scenarios with a cost objective kept in mind. Not only are these systems expected to be able to have fast context switches but also must be extremely small so as to be in compatibility of the machinery with which it is deployed. Since these systems are deployed to carry out real world applications a quick response to external triggers is a must for a real time system which can be ensured by providing a low level interface with the ability of high processor utilization. For this objective, mostly, multiple processors are used and context switching between them is applied. (Petters)

These assignments are additionally provided control by an ongoing working framework, however this can likewise complete applications which require high-precision timing and a high level of unwavering quality. This is especially critical in estimation and automation frameworks, as failure cycles and time consumed are expensive, a deferred program execution can represent a danger. Real time frameworks that can ensure a most extreme measure of time to play out these operations are regularly termed as "hard" real time frameworks, while those that must be utilized as a part of a significant section of the cases are termed as "soft" real time frameworks. Practically speaking, these classes have just a restricted use, as every arrangement with a constant working framework has unique execution qualities that deliberately depends upon the system requirements (Real-Time scheduling: from hard to soft real-time systems, 2015).

Continuous working frameworks give software engineers a high level of control over how processes are organized, and can mostly give affirmation that critical work processes are carried out without failure.

2. CHALLENGES

In order to understand the objectives of the course and the need of Inchron tool suite, it is important to observe the challenges faced while developing real time systems. Real time systems require intelligent control mechanisms in order to meet the expectation of quick responses. Real time systems need to meet their targeted objectives with very limited amount of hardware resources and so there are a lot of challenges that come into observation while developing such systems. Inchron tool suit may be used for developing soft real time systems but its true functionality is tested when it comes to analysing and implementing hard real time systems where physical constraints are very influencing factors that help to choose appropriate actions for avoiding catastrophic failures. The aim of real time systems is to develop such methods that guarantee the rate of reaction of the control system under observation is in compatibility with rate of change in environment (Michael W. Masters, undated).

Some of the most common issues faced during the design and deployment of real time systems are-

1. Maintaining response threshold:
   Successful culmination of an operation relies on the right and well-timed operation of the framework. During the architecture design stage, the equipment and programming engineers cooperate to choose the correct framework design that will meet the necessities. A more straightforward design has a superior possibility of meeting the real time prerequisites. For the most part, stacking a connection more than 40% is not considered a good practice (Finding Response Time in a Real Time System, 1985). A CPU with truly high usage will prompt to unlikely real time behaviour. Additionally, it is conceivable that the high priority processes in the framework will starve the low priority ones of any CPU time. Similarly as with connection, it is advised to keep the peak of CPU use under 50 %.
2. Recovery: Contrary to desktop applications, real time applications don't have the advantage of popping an error box and crashing on detection of a failure event. Real time systems need to be designed to defend against failure conditions. This turns out to be considerably more imperative in a real time framework since grouping of events or triggers can bring about a substantial number of critical situations. It may not be conceivable to test every one of the cases in the lab environment. Hence additional protective checks need to be applied to recoup from failures. (DeVale, 1999).
3. Distributed Architecture: Most real time frameworks include processing on a few distinct nodes. The framework itself disperses the processing load among a few processors. Keeping up information structure consistency is a test when different processors are included in targeted execution. Distributed frameworks might keep running on different processors, yet they need to distribute assets from a mutual pool. Shared resource assignment is commonly overseen by a solitary processor apportioning assets from the mutual pool. On the off chance that the framework is not planned intricately, the common resource allocator can turn into a bottleneck in accomplishing full framework limit (Reveliotis, undated).
4. Other challenges:
   Apart from these major challenges, asynchronous communication also forms a major challenge. Remote procedure calls are utilized to run procedures on a remote machine similar to local procedures but due to the fact that real time communication is mostly asynchronous and dependent on events, using RPC's to determine the next state of machine becomes problematic (Khalili, n.d.). Race conditions and timing issues are also prime challenges while developing real time systems. Inchron tool suite provides a variety of options to deal with these challenges and allows its users to develop real time systems efficiently.

3. RATE MONOTONIC SCHEDULING

RMA is a very crucial algorithm based on the occurrence of events. It is very commonly used in practical applications and utilizes priorities of each task for scheduling. Priorities are assigned to tasks based on how frequently they occur in the system. The task which has the highest rate of occurrence is given the highest priority and vice versa. In case of static priority real time scheduling, RMA is considered as the most optimal option.

Priority, $p = k/p_i$ where k is a constant and p_i is the period of task T. The priority of a task increases with the rate of arrival of the task. It

also varies inversely proportional of the duration of the task.

A test needs to be performed to check whether a set of tasks can be correctly scheduled using RMA by determining an outcome based on worst case execution time (WCET) and the duration of it. The WCET is usually obtained using simulation of the processes. For a set of tasks to be able get scheduled using RMA, this condition must be met with:

$$\sum_{i=1}^{n} \frac{e_i}{p_i} = \sum_{i=1}^{n} u_i \leq 1$$

where e_i is the WCET, p_i is the duration of the process and u_i is the utilization of CPU because of task T. The sufficient condition for the test of RMA scheduling is the Liu and Layland's condition described as

$\sum_{i=1}^{n} u_i \leq n(2^{\frac{1}{n}} - 1)$ where $u_i$is utilization of CPU because of task T (Real Time Systems: Theory and Practice, 2009).

In case of RMA, we assume that the tasks are released at the start of the period and are considered to be running independent of each other. If the total utilization calculates to greater than 100%, the system will have scheduling problems. However, if the total utilization is between the utilization bound and 100%, the test is considered to be inconclusive and a more precise test must be used. The response time (RT) test allows analysis of schedulability based upon the following theorem:

For a set of tasks that operated independently and are periodic, if each task is able to meet its deadline with worst case task phase in execution, the deadline will always be met. The RT test requires computation of the response time of each task being executed in the system. If each response time is less than its corresponding period, the system is schedulable. The following is the calculation for a_n or the response time of task i:

$a_{n+1} == C_i + \sum_{j=1}^{i-1} \frac{a_n}{T_j} * C_j$ where $a_0 = \sum_{j=1}^{i} C_j$

The test terminates when $a_{n+1} = a_n$. The system is schedulable if each response time finishes before its deadline (CSE40463/60463).

4. INCHRON TOOL SUITE

This tool provides users a very efficient way to test out their system architecture by providing a simulation environment which allows easy manipulation of architecture elements. So instead of first preparing the entire system and then testing it, the testing can be performed right from the beginning of development cycle and changes can be made depending on the output thus obtained. This suite provides different tools to analyse and test various aspects of real time system development (Overview, n.d.).

In this project, we have used Inchron tool suit to model a system having various tasks and learned how it schedules the tasks based on assigned priorities using rate monotonous scheduling algorithm. With the graphs thus obtained, we made observations as to how a system with one or multiple processor cores juggles and schedules algorithms with different priorities and different process execution times. We also learned how interrupts work in real time scenarios and active jitters in process handling.

5. EXERCISE

In order to increase feasibility and reduce redundancy of the work, we as a team have decided to provide the answers to the exercise questions we have done individually.

Question 2) Consider a pre-emptive fixed-priority schedule which consists of periodic tasks A, B, and C with periods of $(A) = 1\ ms$, $(B) = 2\ ms$, and $T(C) = 5\ ms$.

a) What is implied for the task priorities P(x) by following a rate-monotonic scheduling strategy?

In this case, the task having assigned priority 1, has the lowest priority and increase with the value assigned henceforth. Therefore,

T(A) has priority 3 (highest priority).

T(B) has priority 2.

T(C) has priority 1 (lowest priority).

According to the principle of rate monotonic scheduling, the task having shortest execution duration has the highest priority.

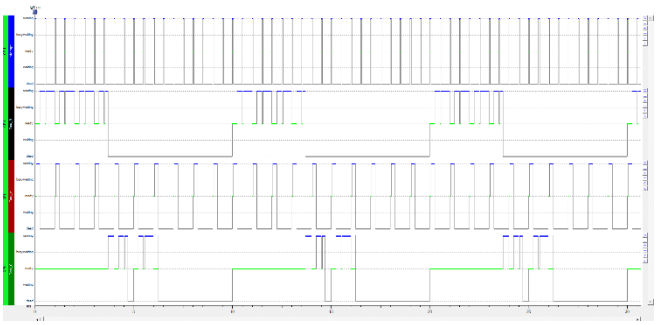

Figure 5.2.1: task execution diagram

From the execution diagram it is observed that the CPU executes task A as soon as it appears, followed by task B which has second highest priority and lastly task C which has lowest priority.

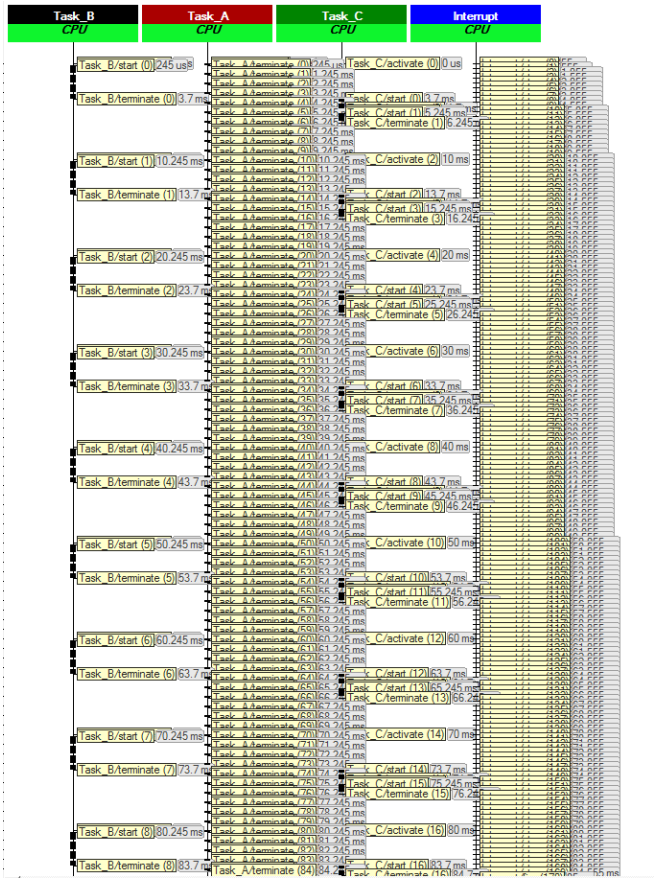

Figure 5.2.2: Sequence diagram

Similar observation is derived from studying the sequence of execution of task which follows the pattern: task A-> task B-> task C.

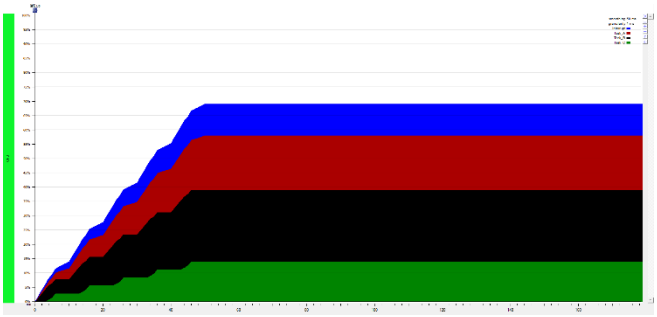

Figure 5.2.3: load diagram

b) From an application point of view, what do you think is the main advantage of RMS?

From an application point of view, it can be stated that RMS proves itself to be optimal in case of static priority scheduling. RMS, when applied coupled with protocols such as priority inversion and priority ceiling protocols provide an optimal way of solving hard real time problems. Moreover, while considering hard real time computations, static priority scheduling proves out to be more compatible with tasks that need to be processed and the order of the execution.

Rate monotonic analysis can be applied to systems where tasks share resources and the issue of unbounded priority inversion pops up. Unbounded priority inversion occurs when a high priority task fails to meet a hard deadline due to being blocked by a low priority task that has acquired a shared resource.

One other major advantage is provided by the schedulability/feasibility tests like the Liu and Layland's condition and Response Time Analysis that allows the developers to check whether a collection of processes will be schedulable using RMS or not. In practice =, it is much more efficient to initially find out that the algorithm being applied will pan out or not instead of building a system only to find out that the tasks are not meeting their deadlines (Teaching).

Question 5) Re-run the simulation with rate-monotonic priority assignment and worst-case execution time distribution. You should note, that task Task_50 suffers activation violations due to its low priority. Now open a Load Diagram and select Granularity & Smoothing from the Diagram Settings menu. Apply the following parameter values.

a) What is the approximate time distance between two load peaks? How do you explain those peaks?

Before applying the granularity interval, the following load diagram is obtained having approximate time distance between two load peaks equal to 11 ms.

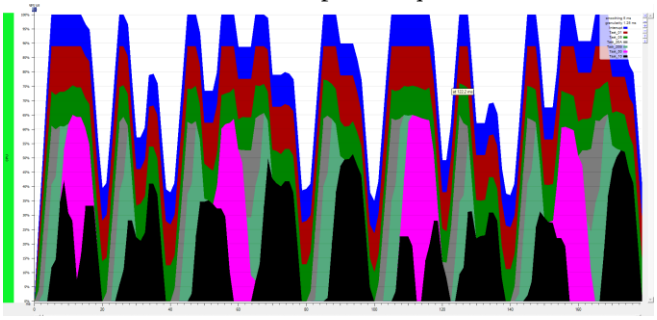

Figure 5.5.1 Load peaks before granularity adjustment

This diagram displays the computational utilization of processing resources by the tasks in the system with each band signifying the CPU utilization done by respective task.

After applying the granularity interval, the following load diagram is obtained with time distance between two load peaks averaging out to 13.5 ms.

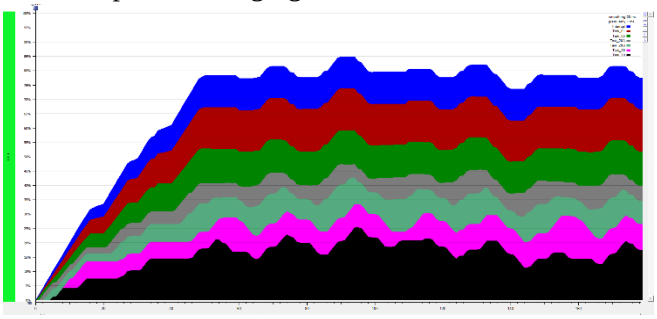

Figure 5.5.2 Load peaks after granularity adjustment.

It is observed that the load peaks flatten out after granularity adjustment. Granularity is the time distance existing between two load calculation points. This parameter sets up a compromise between load data precision points and drawing speed. It is considered to

be an optimal practice to keep the value of granularity parameter below 1% of the total time range of the diagram. A smoothening interval is also applied so that each data point displays the arithmetic mean of the CPU utilization of the time interval of 1 ms before and up to that point. Here smoothening interval= n*granularity (in this case n=50).

b) How could you change the scheduling configuration in order to reduce those peaks?
In order to reduce the load peaks, a single offset value can be changed to obtain a smoother CPU execution.

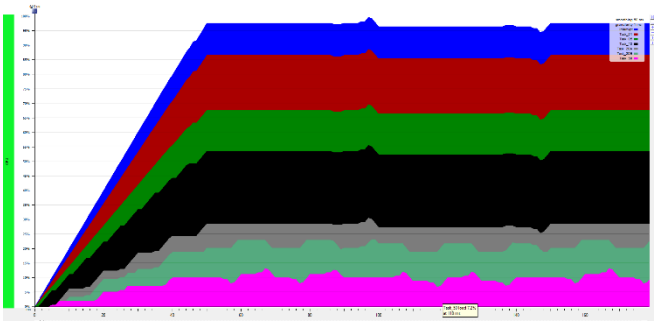

Figure 5.5.3 Load peaks after changing offset value.

This load diagram is obtained after changing the start time of Task 20A and Task 20B. As can be observed from the diagram it provides a smoother CPU execution by distributing the load on the CPU for task execution.

6. CONCLUSION

This paper covers the basics of real time systems with the challenges faced during the development and how a simulation tool like Inchron Tool Suite can help in understanding the system being developed and the processes that will be executed in real time. A simple exercise was conducted to implement rate monotonic scheduling to understand the concept of static priority scheduling and observing the task violations which occur along with methods to minimize them. The tool suite also provides a method to observe the CPU utilization and how it can be smoothened to manage the load carried out during internal computations in real time environment.

REFERENCES

CSE40463/60463. (n.d.). Retrieved from University of NotreDame: https://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwjVtdyuh_XRAhXFiSwKHXopDpAQFggcMAA&url=http%3A%2F%2Fwww3.nd.edu%2F~cpoellab%2Fteaching%2Fcse40463%2Fslides4.pdf&usg=AFQjCNF3G9uQKJLqUKyvfiLcgfxYBhfhEA&sig2=iD55AqiqZE1

DeVale, J. (1999, Spring). *Checkpoint-Recovery* . Retrieved from Carnegie Mellon University : https://users.ece.cmu.edu/~koopman/des_s99/checkpoint/

Finding Response Time in a Real Time System. (1985). *The Computer Journal, Oxford Journals*, 1,2,5.

Khalili, B. (n.d.). *Issues In Using RPC's For Distributed Communication*. Retrieved from http://ranger.uta.edu/~khalili/Distributed%20Systems%20-%20RPC.htm

Michael W. Masters, L. R. (undated). Challenges for Building Complex Real-time Computing Systems. *Scalable Computing: Practice and Experience, Scientific International Journal for Parallel and Distributed Computing*.

Overview. (n.d.). Retrieved from Incron. Shaping Real-Time Systems Engineering of Tomorrow: https://www.inchron.com/tool-suite/tool-suite.html

Petters, S. M. (n.d.). Retrieved from UNSW Sydney: https://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&ved=0ahUKEwjg08u1rNPRAhVZF8AKHT7CB4wQFggkMAI&url=http%3A%2F%2Fwww.cse.unsw.edu.au%2F~cs9242%2F08%2Flectures%2F09-realtimex2.pdf&usg=AFQjCNGEjRBuC1DFzTsi61uTkLQ8kdRDcw&sig2=8m-hZDdpdQPfFIfpIUhR

Real Time Systems: Theory and Practice. (2009). In R. Mall. Pearson Education India.

Real-Time scheduling: from hard to soft real-time systems. (2015, Dec 7). Retrieved from Cornell University Library: https://arxiv.org/pdf/1512.01978v1.pdf

Reveliotis, S. (undated). Real-Time Management of Complex Resource Allocation Systems: Necessity, Achievements and Further Challenges. *Georgia Institute of Technology Publications*, 1-2.

Teaching. (n.d.). Retrieved from Prof. Brian L. Evans: https://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=0ahUKEwjE59zV6PTRAhUCtxQKHfnaBskQFggoMAE&url=http%3A%2F%2Fusers.ece.utexas.edu%2F~bevans%2Fcourses%2Fee382c%2Fprojects%2Ffall99%2Fforman%2Flitsurvey.pdf&usg=AFQjCNE5C_fot1d1