|  |
| --- |
| DIFFERENTIAL TESTING OF REAL TIME SCHEDULERS |

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author(s) | Summary of Changes |
| 1.0 | 25/10/2018 | Anurag M | First Release |
|  |  |  |  |

**Q1**. What is your project trying to accomplish?

A real time scheduler needs to not only schedule all tasks in an efficient way, but it also must do so in a timely manner. Each task must also meet its deadline. Every real time application has a unique set of requirements which it needs to cater and for each such application, a suitable scheduler algorithm needs to be selected. In addition to this, there is a one to many mappings between the scheduling algorithms and the schedulers available in the market. This makes it even more difficult in choosing the right scheduler for the application. A Real Time scheduler ‘A’ is said to be as good to be real time scheduler ‘B’ if, both ‘A’ and ‘B’ can schedule a set of tasks without missing a deadline.

Differential testing is a technique in which a set of inputs is given to two applications/algorithms or different implementations of the same algorithm and observe the anomalies in the output. If the output differs, algorithm/program under test is said to be not optimal. This concept of differential testing can be used to randomly generate a set of tasks and feed them in two different implementations of the schedulers and effectively test them.

The end goal of the project would be to evaluate two real time schedulers based on their schedulability of tasks assuming the tasks are schedulable i.e the tasks satisfy the sufficient and necessary conditions for the scheduler under test.

**Q2**. What have you done so far in the project?

**Part 1** – **Identifying Inputs**

To start off, we first zoned in on the class of schedulers which we will be testing. Given the timeline for the project, we decided to do the tests for only a particular class of schedulers instead of directly jumping in to a generic scheduler which would be able to test any class of schedulers. For the project, we are restricting ourselves to test and compare Rate Monotonic schedulers as this category of schedulers is widely used in many commercials and open source popular real time operating systems. Another reason for fixating on RM algorithm is that it is a static scheduling algorithm, all scheduling decisions are made a priori unlike dynamic schedulers. Moreover, we would be considering periodic tasks for the analysis and the deadline of the task would be equal to the period of the task and the tasks would be independent of each other.

We are currently trying to figure out how to randomly generate a task set using Nezha tool to generate a set of tasks which are feasible for scheduling and each task has its computation time and period associated with it.

**Part 2 – Designing the System**

Design STC

STCs

STCs tasks

Prepare STCs for tasks

Task sequence output generated.

Run Algorithm under test

Compare result with the STCs

The scheduler test cases (STC) will be used to evaluate the tasks sequence output generated by the code under test and the desired output of the sequence under ideal conditions. The STCs and the desired result will be generated using random set of tasks. The desired output of the task sequence for the random set of tasks will be computed using the standard implementation of the RM algorithm. (Currently not sure if this would be the correct approach but we are uncertain as to how to verify the correctness of the output as both the algorithms which are being tested using differential testing might give incorrect output and there would be no way to test the correctness of the system if we do not have a standard reference).

**Part 3 – Getting Hands on Nezha**

As we have decided to use Nezha as the base tool for undertaking the project, we are currently studying how the tool was designed and built to get a deeper understanding of the tool and its internals. We are reviewing the Nezha Paper “NEZHA: Efficient Domain-Independent Differential Testing”. The algorithms mentioned in the paper helped us in coming up with the basic algorithm for the project which has been depicted in the block diagram above. At the same time we also tried to set up the tool on our machines and tinker with the existing examples so understand the key concepts which we feel would be very helpful for designing the project. In the coming weeks, we plan to explore more on the Nezha aspect and get more knowledge.

**Q3**. What challenges have you encountered? Describe at least one technical challenge that you have encountered in your project. This description should be at least half a page long. Feel free to list more than one technical challenge. Provide as much details as possible about each challenge.

**Logistics Issue** - One of the major technical challenge we have faced is we could not install Nezha on our machines. We kept getting an error during installation. We are still figuring out how to install the tool. During the installation, it fails to install libFuzzer library. We tried this on eniac machines too but couldn’t get it to work there as it asks for root password and we used ‘vm’ for the password. It then tried to install Nezha but ended up showing same error. We tried to install libFuzzer independently and link it to nezha but couldn’t. We are planning to work on this issue during the weekend.

**Technical Issues –**

1) The issue which came to surface during our initial study was that Nezha is built using cpp and many of the schedulers which we considered for testing are written in c. We are still unsure if Nezha can be used to test programs written in c. As an alternative in worst case scenario, we are also coming up with a list of popular real time operating systems which use RM Scheduling and are programmed in cpp.

2) We are trying to figure out how to generate a set of tasks randomly. The generated tasks should be feasible and schedulable (Satisfying few conditions). In addition to the computation and period to be random, the total number of tasks in the task set should also be random. Assuming we have a set of randomly generated tasks which satisfy the pre condition of feasibility, we are not sure how we can invoke the program under test using this generated task set. Each program under test might have a different syntax for invoking the tasks and we are trying to come up with a mapper which would take in the set of tasks and generate a procedure for invoking the tasks in program under test.

**Q4**. What do you plan to do by the next milestone? Describe what you plan to accomplish in your project by the next milestone (Nov. 15). This description should be at least half a page long. List each task separately and provide as much details as possible. Ensure that the challenges described above are addressed by these tasks.

<put few milestones here>

**Q5**. Do you have any additional comments? This question is optional. Please describe anything you wish to communicate to the course staff, e.g. any logistical challenges you have encountered.

< I do not know what to put here>