**CSE – 5311: Design and Analysis of Algorithms**

**Project Report**

**Task Scheduler using Red-Black Tree**

**Team Members:**

Ashmi Bhanushali: 1001104720

Meet Brahmbhatt: 1001119131

Prawal Sharma: 1001104720

Yash Shah: 1001111714

**Project Report Content:**

1. **Introduction 3**
2. **System Design 3**
   1. **Data Structures 3**
   2. **Modules Interactions 4**
3. **Analysis of the Algorithm 5**
4. **Experiments** **6**

**Introduction**

The Task Scheduler using Red-Black tree is a system that implements a simpler version of Linux’s Completely Fair Scheduler (CFS). The implementation allows the schedules to take the inputs as a list of task, the time taken for each task and their allocated time, and serves the tasks based on these parameters. The serving is based on the fairness value associated with the task, and hence the values are updated upon each iteration.

This report includes system design, explanation of the data structures used, modules interactions and analysis of the Red-Black tree task scheduler. It is then compared with an implementation of a HEAP version of Task Scheduler. Post which the detailed experiments and reports are being presented.

**System Design**

1. **Data Structures**

The two major data structures used for the project is Red-Black tree and Heap. We have simulated these two tree operations in a methodology to build a scheduler. The Red-Black tree is implemented using classes for a R-B tree. Following are the specifications:

* Task List: Type Array List
* Fairness Value: Type Integer
* Left/Right/Parent Node: Type Node
* Color of the Node: Boolean
* Time (starting time, total time) : Type Integer

The time is represented as an integer and its value is decremented by a factor of 1. For Heaps, the implementation data types are the same except it does not require the details of parents of left/right child (as they are calculated using arrays)

1. **Modules and Interaction**

**Red Black Tree**

There are three main modules in our project, which are TaskSchedulerRBTree, TaskClass and RBTClass. Among these three modules our main module is TaskSchedulerRBTree which controls, runs and calls appropriate functions over the time period. TaskClass is a module which stores the information about the tasks like task id, start time and time required to complete the task. RBTClass is the module which is an implementation Red Black tree data structure.

As TaskSchedulerRBTree is the main class which controls and directs the execution of our project. First, TaskSchedulerRBTree reads the input file and creates objects for respective tasks got from the input file. Then TaskSchedulerRBTree runs the timer till the time allocated to that scheduler. For each time slice, TaskSchedulerRBTree will first find the tasks which should be entered into Red Black tree and will enter those tasks into Red Black tree. After the tasks are being added to the Red Black tree, TaskSchedulerRBTree will call a function which will find the task which has the lowest unfairness value, delete that task from the Red Black tree and serve that task and add that task again into the Red Black tree. When TaskSchedulerRBTree serves the task, the unfairness value will be increased by 1 as it got 1 time slice of CPU. TaskSchedulerRBTree will do same function call till the time reaches to the end which means scheduler got the given time. RBTClass is an implementation of Red Black tree, it has all main methods of the like insert, delete, Red-Red violation check and rotation which is required for the removing red-red violation.

**Heaps**

There are three main modules of the Heap Implementation of the Task Scheduler, which are Scheduler, HeapExecution and TaskList. The main module is the Scheduler module which controls run and calls appropriate functions over the period of time. TaskList is the module which keeps tracks of the fairness value, node details like start time and completed time. The HeapExecution is the implementation of Heaps which has all its operations, including delete, insert, heapify and findMin.

The Scheduler module starts with creating a list of tasks by sorting the tasks according to their start time. It then maintains a dummy clock, and checks for the next tasks to be inserted. Scheduler module then calls the HeapExecution module to insert the tasks at that period of time. HeapExecution then serves the task based on their fairness value (lowest). While serving the task, it removes the node from the Heap with the lowest value of fairness, updates the value and inserts the node at its appropriate position. The value on serve is incremented by 1, demonstrating the time chunks a task is given to the PC. If the task time is equal to the total time required for the task, the task is removed from the list.

1. **Analysis of Algorithm**

**Red-Black Tree Scheduler**

The scheduler which uses Red-Black tree to run itself takes log n time to find the task which has lowest unfairness value because that node will be the left most node among all nodes, it takes constant time 1 to serve that selected node and takes log n time to insert the node into the Red-Black tree and log n time to remove red-red violation from the Red-Black tree in worst case.

**Heap Scheduler**

The Scheduler which uses Heaps to run itself takes constant time 1 to find the task which has lowest unfairness value and for removing that task it tasks it takes log n, then to serve it takes same time as Red Black tree takes which is constant time 1, then to insert that task again into heap it takes log n time.

**Comparison**

If we compare both data structures, total time taken in one time slice by Red-Black tree is 3\*log n while total time taken in one time slice by heaps is 2\*log n. So if we compare both scheduler for more time we can find out the major difference between the schedulers. Red-Black tree will be slow when it runs for longer time than the heaps. You can find the difference in the running time which is given below. We have also displayed the best case, worst case and average cases. Please see the experiments given below.

1. **Experiments**

The experiments are being performed for both the scheduler by varying the input size and the type of input for both the task scheduler. The type of input are best case, average case and worst case inputs. The input size varies from majorly 2000 to 17000 input size. The graph plots and values are given below.

**Best Case**

Red-Back Tree:Time (Minutes) Vs Input Size Heap: Time (Milliseconds) Vs Input Size

**Average Case:**

Red Black Tree: Time (Minutes) vs Input Size Heap: Time (Milliseconds) VS Input Size

**Worst Case:**

Red-Black Tree: Time(Minutes) Vs Input Size Heaps: Time (Milliseconds) Vs Input Size