IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 17, Issue 1, Ver. I (Jan – Feb. 2015), PP 35-37
www.iosrjournals.org

An Autonomous Self-Assessment Application to Track the Efficiency of a System in Runtime Environment

[1]Arif Aziz, [2]Inderjit Lal, [3]Mayukh Chowdhury, [4]Dr. P S Jagadeesh Kumar
ESL Lab, Saltlake City, Kolkata – 700064

Abstract: *In this paper we are proposing a system which will intelligently determine the running time of a process according to the current processor state and the priority of the process. Moreover, given a pool of processes, the software will determine the order in which it should be run to get the optimal result without wasting the computer resource. It can be used as a tool to compute the efficiency of a system as well as to calculate the overhead time of a system.*

I. Introduction

We are using several scheduling algorithms to optimize the result. The algorithms we are using are enlisted below.

Shortest Job First (SJF) scheduling:

In SJF scheduling the approach is to minimize the waiting time. This is one of the best scheduling algorithm. But, this method is impossible to implement. The primary problem with this method is that the processor should know in advance how much time a process will take.

First Come First Serve (FCFS) scheduling:

In this scheduling algorithm the jobs are executed on first come, first serve basis.This method is easy to understand and implement.The major issue is, the performance gets poor by time because the process with high priority has to wait for a long time.

Priority Scheduling

In Priority scheduling each process is assigned a priority. Process with highest priority is to be executed first and so on.Processes with same priority is executed on first come first serve basis. During runtime, the priority can be decided based on memory requirements, time requirements or any other resource requirement.

Round Robin scheduling:

According to this algorithm, each process is provided a fixed time to execute called quantum.Once a process is executed for given time period. Process is preempted and other process executes for given time period.Context switching is used to save states of the preempted processes.

Multilevel Queue scheduling:

Multilevel Queue Scheduling is one of the most efficient scheduling algorithm. In this method, multiple queues are maintained for processes.Each queue can have its own scheduling algorithms and priorities are assigned to each queue.

II. Motivation

In computing scheduling is the method by which threads, processes o data flows are given access to the system resources like processor time, communication bandwidth. This is usually done to load balance and share system resources effectively or achieve a target quality of service. The need for a scheduling algorithm arises from the requirement for most modernsystems to perform multitasking and multiplexing. We thought of designing a system which will help us to determine the expected running time of a given process or a series of processes. Moreover, we thought of a system which will help us to understand how the scheduling of those processes will take place. It will not only help to understand the small systems but also high end system like servers.

III. Methodology

a. Predicting the approximate running time of a process:

For achieving the proposed goal of the system we have divided our work in several parts and developed them independently. We have used different algorithms for different parts of the project. Our system has two

parts. Initially, we have given each process a particular weight according to the number of instruction present in that process. After getting the weight of the process, we are checking the running time of the process. After getting that, we are storing the data along with the time. In a long run, it will generate a huge amount of data. In the next step of our project, we are using those data to approximately determine how much amount of time will a certain process take to run in that system. Fig. 1 is depicting the top view of our system.

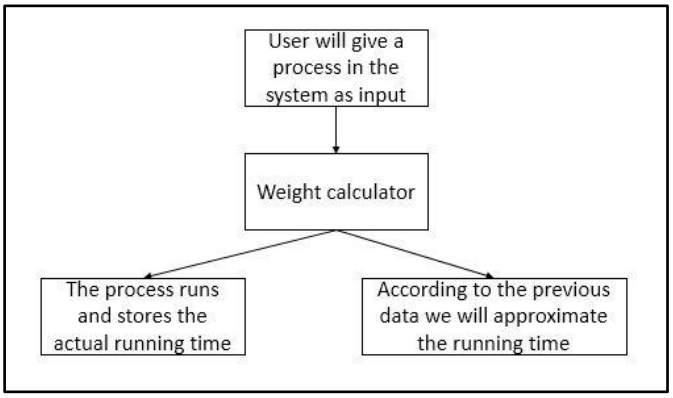

Fig. 1 – Top view of our system

To calculate the weight of a particular process, we have designed a parser program which will calculate the number of instructions present in that process. Moreover, it will enhance the weight of the program in some particular cases, i.e. if there are nested loops or file operations or nested data structures. Now, according to the previous data from the database, we are predicting the running time of the system. After that, the process will run and the actual running time will be stored in the database for further uses.

a. Scheduling a series of process:

Given a set of process, we are creating a queue for running the processes to get optimal result. The algorithm for scheduling the processes is described below.

Algorithm for scheduling the process:

1. Initially, we are calculating the weight of each of the program using the parser program we have designed earlier.
2. According to the weights, we are arranging the programs using the algorithm, priority based scheduling.
3. Now, we have set our threshold at 2 seconds.
4. If a process having smaller weight and waited more than the threshold value will automatically increase the priority.
5. That smaller process will be executed after the currently running process.

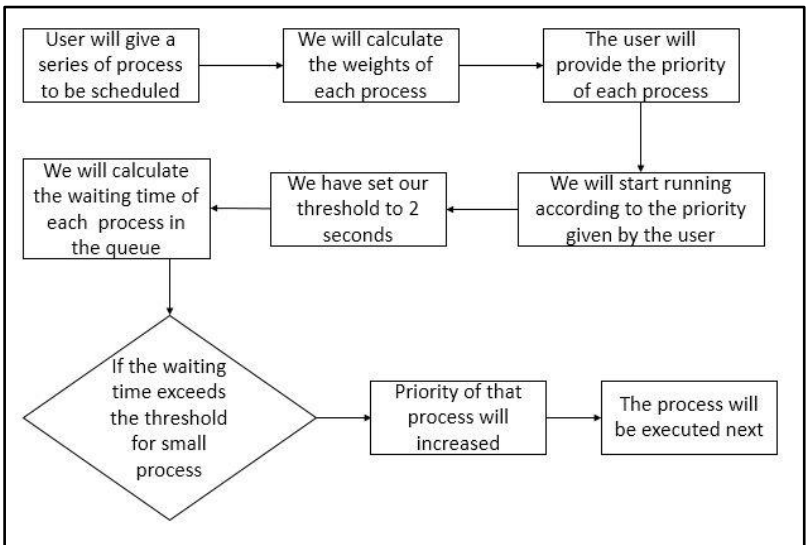

Fig. 2 – The scheduling algorithm we are using to optimize the system

Data Structure:

We have used data structures stored in file to reduce the space and the complexity. Fig. 3 depicts one of the data structures we have used to store the information of the weight of the process and the running time of the process.

```
[{"weight":42,"time":0.0000134},{"weight":40,"time":0.0000132},...]
```

Fig. 3 – Data structure we are using to store the information

IV. Results

The following screenshot depicts the expected running time of the processes given by the user. The parser program is calculating the weights of each process. After that, using the previous data saved in the data structures the algorithm is determining the expected time for each process.

```
IDLE 2.6.6      ==== No Subprocess ====
>>>
Enter the file name: p1.py
Enter the file name: p2.py
Enter the file name: p3.py
Enter the file name: p4.py
Enter the file name: done
List of processes...
--------------------
['p1.py', 'p2.py', 'p3.py', 'p4.py']

Weight of each process...
-------------------------
{'p1.py': 5, 'p3.py': 12, 'p2.py': 42, 'p4.py': 100}
expected running time...
------------------------
p1.py  :  0.0499999523163
p3.py  :  0.119999885559
p2.py  :  0.419999599457
p4.py  :  0.999999046326
>>>
```

Fig. 4 – Figure showing the result for expected running time of each process

V. Conclusion

Our system is working as per our expectation. The algorithms we had previously developed on papers, this time we have implemented them in simulator. The results are satisfactory but there still a chance to make it better. We are still working on it to make the system better. In future we will try to implement our algorithms in a real device so that we can actually measure the performance of the algorithm we have developed.