Survey on Optimization of Operating System

Aniket B Tate ¹, Laxmi A Bewoor ²
¹ PG Student, Dept. of Computer Engineering, SPPU Pune
² Assistant Professor, Dept. of Computer Engineering, SPPU Pune

Vishwakarma Institute of Information Technology
Pune, India

Abstract—Scheduler is one of the important components of operating system which decides system’s performance. Scheduler’s job is to allocate CPU to each process. Paper focuses on system optimization algorithms. Literature survey shows work done to improve performance of system. The paper proposed an idea to improve system’s performance by optimizing workload on CPU by meta-heuristic algorithm.

Keywords—Operating System, CPU scheduling, Scheduling Algorithms, Workload, Metaheuristics.

I. INTRODUCTION

Operating System consists of several components like scheduler, memory manager, file system manager, and I/O device manager. Several parameters from these components are used to tune the Operating System to handle a particular workload. Workload is the number of processes running on specific core at particular time. To effectively use the system and to improve system’s performance there is need to work on workload distribution of processes. Scheduler is one of the major components which affect system performance largely. Scheduling is a fundamental operating-system function. The aim of the scheduling is to share the resources by number of processes. Almost all computer resources are scheduled before use. Scheduling is central to an Operating-system’s design and constitutes an important topic in the computer science curriculum. There are number of parameters to be considered while scheduling a process on core like average waiting time, turnaround time, time quantum for a process, number of context switches, earliness, tardiness of process etc. Each process use CPU for its allocated burst time only. Scheduling problems are NP-hard problems. Heuristics like Genetic Algorithm (GA), Ant Colony Optimization (ACO) ([2], [4], [5], [9], [10]) shows good results to solve optimization problems.

There are many scheduling algorithms like First Come First Serve (FCFS), Shortest Job First (SJF), Round Robin (RR), Priority scheduling. [3]

1. FCFS - Processes are assigned the CPU in the order they request it.

Algorithm:
Step 1: The first requested process is allocated to the CPU first.
Step 2: The new processes are added at the tail of the ready queue.
Step 3: When the process terminates, dequeue the next process from head of ready queue and run it.

2. SJF - When the CPU is available, it is allocated to the process that has the smallest next CPU burst.

Algorithm:
Step 1: Allocate the CPU to the process that has the shortest burst time.
Step 2: If one or more process has the same burst time
    { Allocate the CPU to the process according to the FCFS scheduling }

3. RR - Each process is given a limited amount of CPU time, called a time slice, to execute. If the required CPU burst of the process is less than or equal to the time slice, it releases the CPU voluntarily. Otherwise, the scheduler will pre-empt the running process after one time slice and put it at the back of the ready queue, then dispatch another process from the ready queue.

Algorithm:
Step 1: Choose the time quantum and assign it for each process.
Step 2: Allocate the CPU to the process according to the FCFS scheduling.
Step 3: If (burst time of the process < time quantum).
    { Allocate the CPU to that process till it terminates. }
Else
    { The process will occupy the CPU till the time quantum and it is added to the tail of the ready queue for the next round of execution. }

4. Priority scheduling - The O/S assigns a fixed priority rank to each process. Lower priority processes get interrupted by incoming higher priority processes.
Algorithm:
Step 1: Assign priority to each of the processes in the ready queue.
Step 2: Allocate the CPU to the process that has the highest priority and so on.
Step 3: If two or more process has equal-priority then
   { Allocate the CPU to the process according to the FCFS.
   }

The rest of the paper is organized as follows. Section II gives overview of literature survey done. Section III gives discussion on idea of improving system’s performance by optimizing workload distribution using meta-heuristic. Section IV gives steps for tracing the Linux system and showed the trace results and workload distribution. Section V presents conclusion of this paper.

II. LITERATURE SURVEY

A. Punhani et al. [1] proposed GA to optimize CPU scheduling based on two objective 1.waiting time 2.execution time with priority. Authors have used two fitness functions to represent low average waiting time and priority of jobs. GA is used to find good solution, delete bad solution and make copy of good solution. Rank based selection is used with two point crossover. GA parameters used in paper are population size: 10, generation count: 10, probability of crossover: 0.8, probability of mutation: 0.2. They used NSGA-II to show results for 10 cases for 5 processes. Paper presents pseudo code for NSGA-II algorithm. Results are compared with SJF and priority scheduling. Author suggested that starvation should be taken into consideration as future scope.

T. Maktum et al. [2] proposed GA for processor scheduling based on average waiting time. The authors have used fitness function to represent average waiting time. Number of waiting processes to the total number of processes. Chromosomes of size 5 are taken which represents set of processes. Crossover of single point and mutation rate is 0.005 is considered. 10 cases are taken and compare with FCFS and SJF. Graphical representation is given for set of processes against average waiting time. Future scope suggests consider turnaround time and context switch.

M. Sindhu et al. [3] explained the concept of scheduler, types of scheduler. With diagram they explained the states of scheduler. They have explained four algorithms named FCFS, SJF, RR, Priority scheduling with their algorithms and characteristics. They have showed giant chart of each scheduling algorithm and compared waiting time, turnaround time in tabular form. They propose an algorithm which calculate time quantum which is sum of all burst time. Then algorithm assigns this quantum time to each process and shuffle the process in queue. Giant chart of proposed algorithm is showed and waiting time, turnaround time of proposed algorithm is compared with FCFS, SJF, RR and Priority scheduling. The proposed algorithm shows minimum waiting and turnaround time.

C. Chiang et al. [4] considered heterogeneous cluster system. The paper presented scheduling as Acyclic Directed Graph (DAG) where node denotes task (programs are decomposed into tasks) and edge denote transition from one state to another state. Paper used ACO for task matching and scheduling problem and they proposed new algorithm called ACO-TMS. This algorithm is proposed based on new state transition rule and local/global updating rules. The proposed algorithm takes less time to find satisfactory solution. Paper presents pseudo code of proposed algorithm. The algorithm has used two state transition rules 1. Processing element rule 2. Task selection rule. The proposed algorithm used three procedures for local optimization implementation. Procedure A: attempt to shorten the critical path to shorten scheduling length. Procedure B: attempt to enhance the efficiency of procedure A. Procedure C: this procedure is used only when large data to be transferred from one state to other state. Paper shows comparison table with cluster size 2, 4, 8 and number of tasks are 60. Results are compared with GA, dynamic priority scheduling (DPS).

A. Kaur and B. Khedra [5] proposed optimization of execution time and penalty cost of processes with common deadline on single processor. Paper calculates earliness and tardiness to calculate penalty. They have used 3 processes with deadline length 12. The paper has presented 6 orders of scheduling an calculated earliness and tardiness of each process. Paper used GA with initial population size 20, roulette wheel selection, two point crossover and 0.05 mutation rate. Results show GA gives less execution time than other heuristics.

Mahesh Kumar M. et al. [6] calculated approximate next CPU burst time with the help of dual simplex method in SJF. They calculate waiting time and turnaround time then convert the burst time, waiting time, and turnaround time into Linear Programming Model (LPP). Predicted and calculated burst time is nearly same. They have taken two test cases to compare original burst time and approximate calculated burst time. They use dual simplex method to calculate CPU burst.

A. Yasin et al. [7] suggested technique to RR so that waiting time, turnaround time, and context switches get minimized. The scheduling algorithm depends on following selection criteria 1.Fairness 2.CPU Utilization 3.Throughput 4.Response Time 5.Waiting Time 6.Turnaround Time 7.Context Switch. The algorithm works with priority. They have presented Flow Chart of Prioritized Fair Round Robin (FPRR) algorithm along with its step by step...
The assumptions are it is single processor, overhead is solved by considering priority and burst time of process. Results are compared with RR, IRRVQ and Priority RR. Conclusion suggests that the proposed algorithm is good for time sharing system.

S. Suranauwarat [8] presented a simulator for various scheduling algorithms (FCFS, RR, SJF, SRTF, MLFP) for single CPU. It shows 1. Realistic process use 2. Actual working of process 3. Manual change in scheduling. Simulator has two modes 1. Simulation mode 2. Practice mode. In simulation mode, the user can start and stop the simulation and watch it step-by-step. In practice mode, the user can predict when and how long each process is in a particular state and why it is in that state and check whether his answer is correct or not with the simulator. The simulator is written in java.

K. Kotecha and A. Shah [9] present combination of two algorithms called EDF and ACO for scheduling. When system is under loaded EDF is use and when system is overloaded ACO is used. Switching occurs at run time. Adaptive algorithm is compared with EDF and ACO by success ratio and effective CPU uses with 10 cases. Load at particular time is calculated by sum of execution time of task to the total period of tasks. Success ratio is total scheduled jobs to the total arrived jobs.

George Anderson et al. [10] show software performance optimization methodology. Scheduling solutions based on data mining. Parameters considered for optimization through the paper are response time, throughput, scalability, waiting time, turnaround time, and resource utilization. They studied system performance by two methods

- Experimentation
- Analytical modelling.

The paper suggested the optimization in various fields:

The work suggests solution for

1. Distributed system
2. Single processor
3. Symmetric multiprocessor
4. Asymmetric multiprocessor system

The work shows resource scheduling with machine learning, schedule java thread to minimize garbage collection and synchronization, trace data used to find patterns and analyse them, optimization of e-commerce websites, finding frequent patterns on trace data to detect bugs, capturing semantics of data, machine learning to recognize workload and to select appropriate disk scheduling policy, GA library developed in kernel to tune various part of kernel, dynamic customization of system, self-tuning of kernel activities, tests to recognize correlation between OS performance and scheduling parameters, decompose workload and again reallocate jobs to machines, manipulating threads priority, core to core data transmission with low cost, scheduling algorithm which calculate number of events and match it with expected performance ratio, queue is used to add jobs from one end and delete on other end; if no job is found in queue that queue extract job from other queue.

S. Kuntumalla et al. [11] and A. Ahluwalia et al. [12] showed how task scheduling is performed in cloud computing and how the load balancing is achieved in cloud computing system for multimedia system.

III. DISCUSSION

Literature survey focuses on the scheduling of processes so that waiting time, turnaround time etc. will get reduced. Each process should get processed fairly to optimize the system’s performance.

One application of system optimization is Linux kernel tracing then analysing the load on system and applying met heuristics to optimize the workload distribution. First system will schedule the processes at its own then we analyse the Linux trace and apply met heuristic.

IV. LINUX TRACE AND WORKLOAD DISTRIBUTION ANALYSIS

LTTng is an open source tracing framework for Linux. This tracing tool provides details about processes running on system.

To use LTTng on Ubuntu following packages needs to be installed on system:

1. LTTng-tools: libraries and command line interface to control tracing sessions.
2. LTTng-modules: Linux kernel modules for tracing the kernel
3. LTTng-UST: user space tracing library

LTTng installation steps:

1. Create a session:
   sudo lttng create
2. Enable some/all events for this session:
   sudo lttng enable-event --kernel sched_switch,sched_process_fork
   OR
   sudo lttng enable-event --kernel --all
3. Start tracing:
   sudo lttng start
4. To stop tracing:
   sudo lttng stop
   sudo lttng destroy

Install babeltrace as:

sudo apt-add-repository ppa:lttng/ppa
sudo apt-get update
sudo apt-get install babeltrace

View trace:

babeltrace ~/lttng-traces/my-session

*my-session contains path at which trace is stored.
V. CONCLUSION

The paper focuses on optimization strategies evolved for Operating System. The basic idea is to trace the Linux system and analysing how the scheduler distributes the processes on different cores of CPU. We can then use Metaheuristics to optimize the workload distribution so that cores will be used more effectively.

REFERENCES


