The Wizard Of OS

AMAN BAWANE[1,†], SHOURYA GOEL[2,†], ANANT JAIN[3,†], ATUL KUMAR[4,*], PARVATHI NAIR[5,*], KSHITIJ SHAH[6,†], ARYAN DHAS[7,*], AND VAIBHAVI MAKWANA[8,*]

[1] *Indian Institute of Technology Roorkee, 22114020, 2nd Year, bawane_am@cs.iitr.ac.in, 9325921931*
[2] *Indian Institute of Technology Roorkee, 22114090, 2nd Year, shourya_g@cs.iitr.ac.in, 9971749585*
[3] *Indian Institute of Technology Roorkee, 22114005, 2nd Year, anant_j@cs.iitr.ac.in, 8266892764*
[4] *Indian Institute of Technology Roorkee, 22114015, 2nd Year, atul_k@cs.iitr.ac.in, 6204109034*
[5] *Indian Institute of Technology Roorkee, 22115115, 2nd Year, parvathi_n@cs.iitr.ac.in, 9446300211*
[6] *Indian Institute of Technology Roorkee, 22114049 , 2nd Year, kshitij_s@cs.iitr.ac.in, 9409830487*
[7] *Indian Institute of Technology Roorkee, 22117046, 2nd Year, dhas_as@cs.iitr.ac.in, 9373595528*
[8] *Indian Institute of Technology Roorkee, 22114051, 2nd Year, makwana_vl@cs.iitr.ac.in, 6352776838*
[†] *ML Team*
[*] *CFS Scheduler Team*

Compiled April 18, 2024

Efficient resource allocation and task scheduling are fundamental to optimizing performance in the Linux operating system. This paper proposes a novel approach that leverages machine learning (ML) techniques to enhance the capabilities of the Linux scheduler. The proposed ML engine takes in historical data on process behavior, system load, and resource utilization. This data is then employed to train a ml model, enabling the engine to capture intricate patterns and relationships within the data. By leveraging these learned insights, the ML engine makes intelligent decisions regarding task scheduling and resource allocation. This research presents a pioneering approach to integrate ML into the Linux scheduler, paving the way for advancements in efficiency and responsiveness for Linux-based systems across diverse domains, encompassing cloud computing and embedded devices.

1. INTRODUCTION

A. Background

A.1. CPU Scheduling

CPU scheduling is the procedure of assigning processors to different processes. It aims to maximize CPU usage and ensure a fast response time for interactive users. Various CPU scheduling algorithms exist, each with distinct benefits and drawbacks. Common algorithms include Shortest Job First (SJF), Shortest Time-to-Completion First (STCF), and Round Robin.

A.2. Completely Fair Scheduler (CFS)

The Completely Fair Scheduler (CFS) is Linux's default scheduler, designed to ensure fair allocation of CPU. It employs a system of virtual runtime (vruntime) to give priority to processes that haven't run lately. Using Red-Black trees, it guarantees efficient process selection. While CFS aids in preventing process starvation, it may disadvantage processes that frequently undergo short sleep periods.

A.3. Machine Learning Techniques

Machine learning (ML) is a subfield of artificial intelligence (AI) that focuses on developing algorithms and models that enable computers to learn from data and make predictions or decisions without being explicitly programmed. ML techniques such as linear regression, decision trees, random forests, and support vector regression (SVR) enable computers to analyze and extract patterns from data, leading to insights and predictions in various domains. These models leverage mathematical and statistical principles to learn from labeled or unlabeled data, adapt to changing environments, and improve their performance over time. ML plays a crucial role in tasks such as predictive modeling, pattern recognition, anomaly detection, and optimization, revolutionizing industries and driving innovation in fields such as healthcare, finance, marketing, and beyond.

Linear Regression is a fundamental statistical technique used for modeling the relationship between a dependent variable and one or more independent variables. The model assumes a linear relationship between the variables, meaning that changes in the independent variables are associated with a linear change in the dependent variable.

Decision tree is a non-linear predictive model that uses a tree-like graph of decisions to make predictions based on input features. Each node in the tree represents a decision based on the value of a specific feature, and each branch represents the possible outcomes of that decision. The leaves of the tree contain the predicted outcome for a given set of input features.

Random forest is an ensemble learning method that combines multiple decision trees to improve predictive accuracy and reduce overfitting. It builds multiple decision trees during training and outputs the average prediction of the individual trees for regression tasks or the mode of the predictions for classification tasks.

Support Vector Regression (SVR) is a type of support vector machine (SVM) that is used for regression tasks. It extends the principles of SVM from classification to regression by fitting a hyperplane in a high-dimensional space that best captures the relationship between the input features and the continuous target variable.

B. Motivation

Our motivation for this project stems from the desire to enhance the efficiency and responsiveness of Linux-based systems. We focused on the situations in which a process which is about to finish is preempted from the CPU because it had exhausted it's time slice, but this will increase the overall run time because of the two extra context switches that need to occur for the earlier process to finish. If we had a method to inform the scheduler about the approximate end times of the processes to the scheduler we can stop the scheduler from switching out processes that are about to finish and reduce the total CPU runtime.

C. Application

Operating Systems Optimization: Implementing your scheduler in an operating system can improve resource utilization and system responsiveness. By dynamically adjusting time slices based on predicted burst times, the operating system can allocate CPU resources more efficiently, leading to better overall system performance.

Cloud Computing: In cloud computing environments, where multiple virtual machines or containers share physical resources, your scheduler can help optimize resource allocation. By accurately predicting burst times and adjusting time slices accordingly, cloud providers can improve the performance of their services and better meet the needs of their customers.

Real-Time Systems: In real-time systems where meeting deadlines is critical, your scheduler can help prioritize tasks based on their predicted burst times. Tasks with shorter burst times can be given higher priority to ensure timely execution, improving the overall responsiveness and reliability of the system.

2. PROPOSED SYSTEM

A. Details

Our system can be thought of as having two main components – a ML model and a scheduler:

A.1. ML Model

We have used an ML model to predict the expected CPU burst time of a process based on process characteristics such as RSS memory, VMS memory, shared memory, text segment, data segment etc.

We have created our own model which is trained on our own created dataset. The dataset was created using a python library and contains process characteristic and known CPU burst times of some standard processes.

A.2. Scheduler

We have worked on the Completely Fair Scheduler (CFS) and made changes to it so that it can schedule a process based on the expected burst time of the process which it gets from the ML model.

Figure 1 explains this with a class diagram.

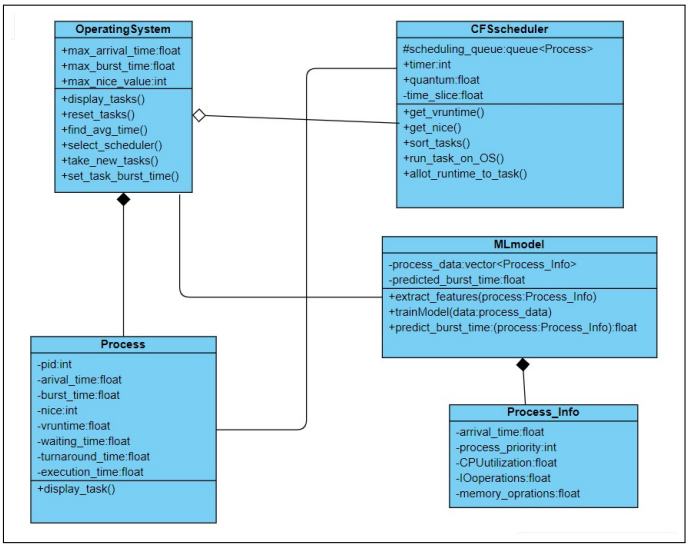

Fig. 1. Class Diagram

Hence the basic working of our system can be summarized as: A process enters the system for execution -> It's characteristics are fed to the ML model -> It outputs the expected CPU burst time -> The scheduler is given the expected burst time -> The scheduler schedules the process accordingly with a priori knowledge.

Figure 2 shows a pictorial representation of this.

Figure 3 shows the sequence diagram.

B. Algorithm

The way that our scheduler is superior in scheduling processes is by having a priori knowledge of burst times. In traditional scheduling whenever a process has exhausted it's time slice it is preempted and another process having minimum virtual runtime is scheduled. However, it is possible that a process is "almost" complete, i.e., the remaining CPU burst time of the process is so small that the overhead of context switching comes into picture and reduces the performance of the system.

Hence, if we can predict the burst time and we realise that allowing the process to run for some more time will actually improve the performance of the system, then we let the process run with an improved time slice.

Algorithm 1. CFS Algorithm

1: **procedure** CFS($process, model$) ▷ New process
2: $process\_data \leftarrow collect\_process\_data(new\_process)$
3: $features \leftarrow extract\_features(process\_data)$
4: $predicted\_burst\_time \leftarrow model.predict(features)$
5: $new\_time\_slice \leftarrow time\_slice(preditced\_burst\_time)$
6: $schedule\_process(process, new\_time\_slice)$

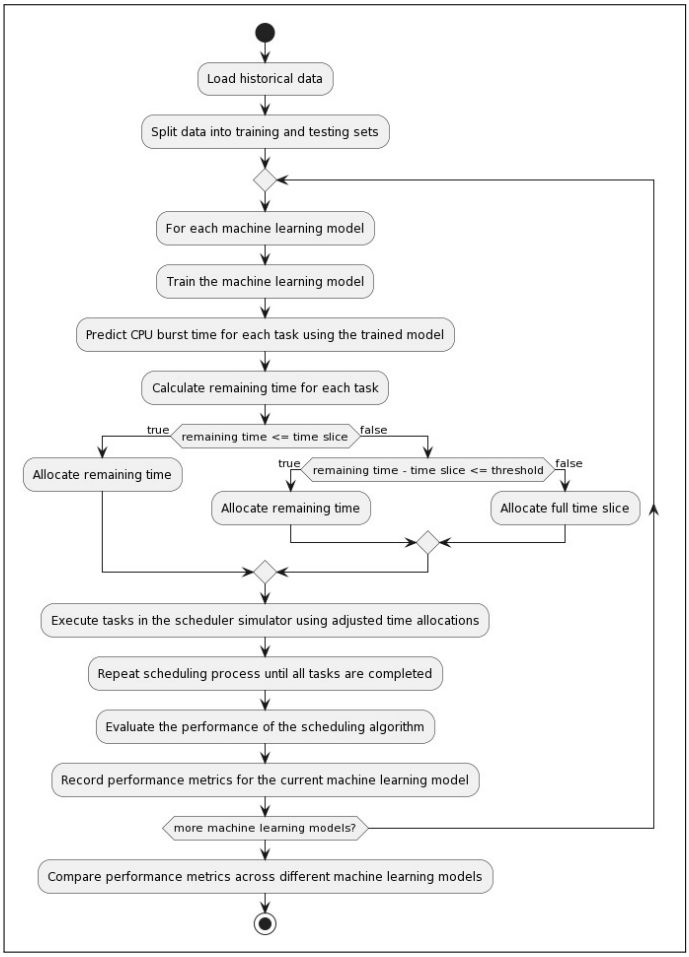

Fig. 2. Flowchart

C. Novelty

Our project introduces novelty in CPU scheduling by leveraging machine learning to predict CPU burst times and dynamically adjust the time slice for processes accordingly. Traditional CPU schedulers typically use fixed time slices or dynamically adjust them based on runtime metrics like virtual runtime or priority levels. However, our approach goes a step further by integrating machine learning models to predict future burst times based on historical data or other relevant features.

Here's how our project stands out:

Prediction of burst times: Traditional schedulers often rely on heuristics or runtime metrics to make scheduling decisions. By utilizing machine learning, our scheduler can make more accurate predictions of future burst times for processes. This predictive capability allows your scheduler to anticipate when a process is close to completion and avoid unnecessary context switches, thus improving system performance.

Dynamic time slice adjustment: Instead of strictly adhering to fixed time slices or predefined scheduling policies, our scheduler dynamically adjusts the time slice for processes based on the predicted burst times. This adaptive approach ensures that processes are given sufficient CPU time to complete their tasks efficiently, without being prematurely preempted due to fixed time slice constraints.

Integration of ML Integrating machine learning into CPU scheduling introduces a novel paradigm where scheduling decisions are informed by predictive models trained on historical data. This not only enhances the accuracy of burst time predictions but also opens avenues for further optimizations and enhancements based on continuous learning and adaptation.

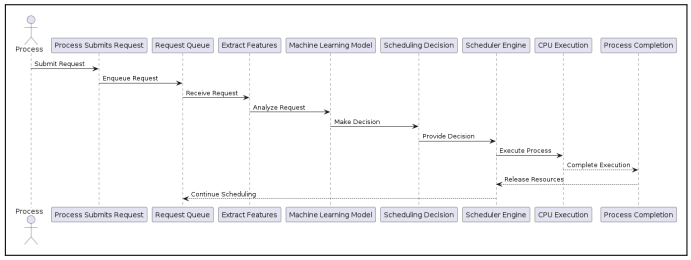

Fig. 3. Sequence Diagram

3. IMPLEMENTATION

Data collection:

1.Using psutil library psutil extract process's attributes from the process structure

Table 1. Attributes on which data is collected

Parameters	Description
RSS memory	Memory occupied in the RAM
VMS memory	Virtual memory of the process
Shared memory	Passing data between programs
Text	Size of the text section
Data	Size of the data section
Lib	Size of shared libraries

2.Polling the process for each millisecond to check if the process is terminated to calculate the final runtime.

3.Params were exported to a csv file

4.Benchmarks used to collect runtime data.

matmul(variable size matrix) ,ml(variable size and dimensions, input linear reg.),sort(variable size array to sort),fileio(variable length strings stream to the other file) .

Data pre-processing:

Standard scaling and droping process in 0 runtime (Using sckitlearn).

Model training :

Random forest regressor,Decision tree regressor, linear regression ,SVR with RBF kerenel,(Using Standard APIS),Also tried to use xg boost.

Model Testing:

Accuracy measured using mean square error from the skit lib.

Hyperparameters tuning:

Grid Search Cross validation

4. REFERENCES

1. Jain, N Nikhil, Srinivas Kumar R, and Syed Akram. "Improvising Process Scheduling Using Machine Learning." In 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT), 1379–82. Bangalore, India: IEEE, 2018. https://doi.org/10.1109/RTEICT42901.2018.9012330.

2. Negi, Atul, and P. Kumar. "Applying Machine Learning Techniques to Improve Linux Process Scheduling." In TENCON 2005 - 2005 IEEE Region 10 Conference, 1–6. Melbourne, Australia: IEEE, 2005. https://doi.org/10.1109/TENCON.2005.300837.

3. Nemirovsky, Daniel, Tugberk Arkose, Nikola Markovic, Mario Nemirovsky, Osman Unsal, and Adrian Cristal. "A Machine Learning Approach for Performance Prediction and Scheduling on Heterogeneous CPUs." In 2017 29th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), 121–28. Campinas: IEEE, 2017. https://doi.org/10.1109/SBAC-PAD.2017.23.

4. Tehsin, Sara, Yame Asfia, Naeem Akbar, Farhan Riaz, Saad Rehman, and Rupert Young. "Selection of CPU Scheduling Dynamically through Machine Learning." In Pattern Recognition and Tracking XXXI, 11400:67–72. SPIE, 2020. https://doi.org/10.1117/12.2559540.