Weekly Report

CS3500: Operating Systems

Visualisation Tool for Process Scheduling

Computer Science and Engineering

Indian Institute of Technology Madras

Jul - Nov 2024

Under the supervision

of

Prof. Janakiram D

submitted by

Team 8

- Anjali Samudrala (CS22B046)
- Chaitanya Sai Teja G (CS22B036)
- Jwala Likitha Reddy M (CS22B078)
- Karthikeya P (CS22B026)
- Naveen Koushik Reddy E (CS22B006)
- Navya Sree B (CS22B045)
- Rushi Babu G (CS22B040)
- Sravya Rangu (CS22B044)
- Yashwanth Sai P (CS22B002)
- Yaswanth Sai V (CS22B043)

1 Frontend Team

1.1 Tasks

- The tasks for this week is to identify the components that are required for the frontend. This includes the following.
  - Coming up with a frontend design and layout.
  - Ideation on what all different plots can be displayed.
  - Informing the backend team about the requirements of various informations.

1.2 Progress

1.2.1 Frontend Design and Basic implementation

-

1.2.2 Ideas for Plots and information from backend

- **Processes Running on the System:**
  -
- **Gantt Chart of various Processes**
  -
- **Status wise distribution of processes**
  -
- **CPU Utilization**
  -

2 Backend Team

2.1 Linking Backend and Frontend

The objective is to create a backend server that fetches real-time system process statistics using the `pidstat` command and streams the data to a frontend via WebSockets. The application uses Flask as the web framework and Flask-SocketIO to establish real-time communication between the backend and frontend.

2.1.1 Tools and Libraries Used

- **Flask**: A lightweight web framework for Python that simplifies the development of web applications.
- **Flask-SocketIO**: An extension for Flask that enables real-time communication between the server and the client using WebSockets.
- **subprocess**: A Python module used to run external commands. In this case, it is used to execute the `pidstat` command, which collects CPU statistics for processes running on the system.

- **re (Regular Expressions)**: A Python module for matching patterns in strings. It is used to parse the output of the `pidstat` command.
- **threading**: A Python module used to create background threads. In this project, it allows the data-fetching process to run concurrently with the main server.

2.1.2 Architecture

- **Backend (Flask Server with SocketIO) :** The backend is responsible for fetching real-time process statistics using the `pidstat` command, parsing the output, and sending the data to the frontend using WebSockets.
- **Real-Time Data Fetching :**
  - The `pidstat` command is used to gather CPU statistics for processes every second. The output contains several fields, including process ID (PID), user and system CPU usage, and the process command.
  - A background thread is created to run the `pidstat` command continuously, fetching data at specified intervals (e.g., every 2 seconds).
  - The data is parsed using a regular expression, and relevant statistics are extracted and formatted into a dictionary.
- **Real-Time Communication :**
  - `Flask-SocketIO` is used to emit the parsed data to the frontend in real time. This allows the frontend to display the latest statistics as they are gathered by the backend.
  - The data is emitted as a WebSocket event (`'pidstat_data'`), making it available for frontend visualization.

2.2 Process Migration

2.2.1 Enabling tracking to log the process migration

- Navigate to the tracing directory.

  ```
  cd /sys/kernel/debug/tracing
  ```

- Enable "sched_migrate_task" that allows to track and log the migration of tasks.

  ```
  echo 1 | sudo tee events/sched/sched_migrate_task/enable
  ```

- Start tracing the events.

  ```
  echo 1 | sudo tee tracing_on
  ```

- Wait for some time and let tracer log some migrations.
- Stop the tracing.

  ```
  echo 0 | sudo tee tracing_on
  ```

- Check the trace file.

  ```
  cat trace
  ```

2.2.2 Logged Data

- An example of the logged data is shown below.

```
Chrome_ChildIOT-33594   [003] d..2. 27300.882725: sched_migrate_task: comm=
```

- In the above log entry, Chrome with PID 33594 initiated the migration of the Compositor process with PID 32759 from CPU 0 to CPU 3.
- $prio = 120$ is the priority of the process. Lower prio means higher priority.
- 27300.882725 is the timestamp of the event. (migration).

2.3 Studying the proc folder

2.3.1 What is the proc folder?

2.3.2 Add the folder name you studied!!

- try to put important points in bullet points
- At the end add some points on what you feel can be done with that information.