Secured Visualization: Authorization, Graphs, and CSV

Aniket Tiwari, UNC Charlotte
Dr Dong Dai, College of Computing and Informatics

UNIVERSITY OF NORTH CAROLINA CHARLOTTE

Introduction

Importance

Meeting vital needs in data-driven applications, this research integrates security, dynamic visualization, and efficient data management for a cohesive user experience in modern applications

Where it is Used

Relevant to contemporary data visualization, it emphasizes enhanced security and dynamic visualization, crucial for scenarios protecting user credentials, empowering users with interactive graphs, and efficiently managing personalized CSV data

Objectives

- **Secure File Access: Design a system where users can only access files they have uploaded, enhancing data security and user-specific file management.**
- **Column Verification and Data Validation: Implement a column verification mechanism for uploaded CSV files, ensuring data integrity and adherence to specific criteria, enhancing the quality of visualizations.**
- **Unified Data Management and Personalized User Experience: Leverage SQLite to seamlessly upload, store, and manage CSV data, optimizing user interaction and delivering personalized and efficient data storage solutions.**

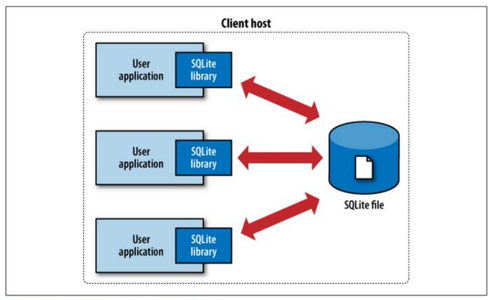
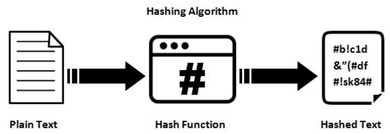


Figure 1-2. The SQLite server-less architecture.

Method

Security Integration

- Implemented a robust security login function utilizing SQLite for user authentication, ensuring the confidentiality of user credentials.
- Enhanced security measures include SHA256 authorization to safeguard against unauthorized access.



Graph Plotting Dynamics

- Utilizing the Streamlit framework, the application features an interactive interface for users.
- Dynamic graph plotting capabilities were implemented, allowing users to seamlessly read, analyze, and visualize complex graph data in real-time.

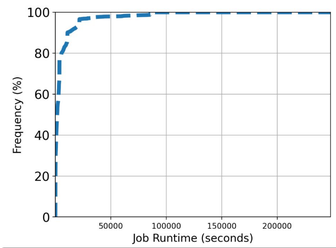


Figure 1.1 depicts the cumulative distribution function (CDF) of theta data from a GPU CSV file uploaded by the user, with the corresponding plotted graph.

CSV Data Management

- A streamlined process for users to upload and save CSV data was facilitated, enhancing overall user data interaction.
- Utilizing SQLite for data storage ensures efficiency in managing and retrieving user-specific data, contributing to a seamless and organized user experience.

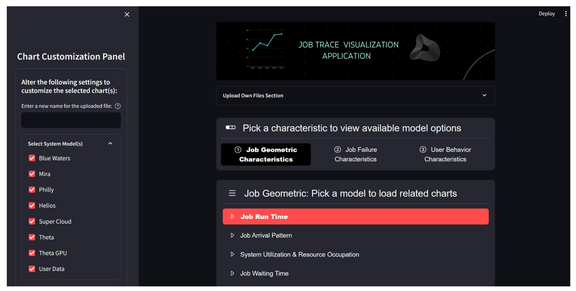
```python
import sqlite3
import hashlib
import io
import concurrent.futures
import chardet

st.set_page_config(page_title="Job Trace Visualization Application", page_icon="📊")
curr_dir = os.path.dirname(__file__)
conn = sqlite3.connect("auth.db")
cursor = conn.cursor()

cursor.execute('''
    CREATE TABLE IF NOT EXISTS users (
        id INTEGER PRIMARY KEY,
        username TEXT UNIQUE NOT NULL,
        password TEXT NOT NULL
    )
''')

cursor.execute('''
    CREATE TABLE IF NOT EXISTS files (
        id INTEGER PRIMARY KEY,
        user_id INTEGER,
        file_name TEXT,
        file_content TEXT,  -- Change BLOB to TEXT to store CSV data as text
        upload_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        FOREIGN KEY (user_id) REFERENCES users (id)
    )
''')

conn.commit()
```

Figure 1.3 :- showcases a snippet of the code, illustrating the integration of SQLite3, a Python SQL library. This snippet focuses on the creation of a database file responsible for storing user credentials and previously entered files. Notably, the security aspect is emphasized, as the user data is hashed using the SHA-256 algorithm for enhanced protection.

Results

Streamlit Proficiency

Acquired expertise in Streamlit, a Python library for rapid development of interactive web applications. Streamlit facilitated rapid development with minimal code, accelerating the creation of interactive features.



CSV Upload and Display

Introduced an efficient process enabling users to seamlessly upload and dynamically visualize CSV files, as depicted in Figure 2.1. Additionally, a functionality has been implemented to generate dynamic graphs based on user-uploaded GPU data. Users now have the capability to visualize customized Cumulative Distribution Function (CDF) graphs tailored to their specific datasets, reminiscent of the illustration in Figure 1.1 presented earlier.

	job	user	project	state	gpu_num	cpu_num	node_num	submit_time	wait_time	run_time	wall_time	node_hour	new_status
0	639,487	0	0	0	0	512	8	1,672,529,301	71	3,632.92	3,600	8.0732	Killed
1	639,448	0	0	0	0	16,384	256	1,672,507,846	7,772	21,654.66	21,600	1,539.8869	Pass
2	639,135	0	0	0	0	262,144	4,096	1,672,174,846	340,972	21,655.97	21,600	24,639.6814	Killed
3	638,720	6,325,3	,325,381	0	0	262,144	4,096	1,671,905,072	632,522	3,648.7	3,600	4,151.4098	Killed
4	639,463	0	0	0	0	8,192	128	1,672,533,950	23,424	5,995.18	10,800	213.162	Pass
5	638,722	6,325,3	,325,381	0	0	262,144	4,096	1,671,905,120	636,224	3,634.13	3,600	4,134.8324	Killed
6	638,723	6,325,3	,325,381	0	0	262,144	4,096	1,671,905,152	639,947	3,669.52	3,600	4,175.0983	Killed
7	639,488	0	0	0	0	8,192	128	1,672,543,325	45	5,880.94	10,800	209.1001	Pass
8	639,489	0	0	0	0	8,192	128	1,672,549,223	66	5,929.52	10,800	207.2718	Pass
9	639,490	0	0	0	0	8,192	128	1,672,555,053	46	5,235.68	10,800	186.1575	Pass
10	639,492	0	0	0	0	8,192	128	1,672,556,039	4,395	57.07	10,800	2.0292	Pass
11	639,497	0	0	0	0	8,192	128	1,672,560,303	490	5,247.99	10,800	186.5952	Pass
12	639,491	0	0	0	0	12,288	192	1,672,555,205	33	10,862.96	10,800	579.3579	Killed
13	639,498	0	0	0	0	8,192	128	1,672,566,007	175	5,288.25	10,800	188.0267	Pass
14	639,499	0	0	0	0	8,192	128	1,672,571,423	47	5,236.58	10,800	186.1895	Pass
15	639,500	0	0	0	0	8,192	128	1,672,576,677	70	5,264.65	10,800	187.1876	Pass
16	639,501	0	0	0	0	8,192	128	1,672,581,984	69	5,203.48	10,800	185.0126	Pass
17	637,860	0	0	0	0	163,840	2,560	1,671,479,504	1,069,365	42,388.64	46,800	30,143.0329	Pass
18	638,453	0	0	0	0	40,960	640	1,671,747,802	801,087	43,245.01	43,200	7,688.0018	Killed
19	639,502	0	0	0	0	8,192	128	1,672,587,200	37	5,313.58	10,800	188.9273	Pass
20	638,659	0	0	0	0	51,328	802	1,671,868,236	680,633	43,721.95	86,400	9,740.2789	Pass
21	639,505	0	0	0	0	8,192	128	1,672,592,513	37	5,293.23	10,800	188.2037	Pass
22	639,508	0	0	0	0	8,192	128	1,672,600,471	39	38.53	10,800	1.37	Pass

In Figure 2.1, the presentation of uploaded CSV data is showcased, revealing the structured view with distinct columns. Notably, columns such as cpu_num, node_hour, state, submit_time, etc., deemed crucial for cumulative distribution function (CDF) analysis, are expected. In the event of their absence, a user-friendly error message guides the user to adhere to the correct CSV data format.

Conclusions

Conclusions

- Explored features such as file upload, data visualization, and dynamic content rendering in Streamlit, gaining practical insights into SQLite3 for user authentication and data storage.
- Implemented robust access controls, ensuring users can exclusively manage and access the files they have personally uploaded.
- Future plans involve integrating a Cumulative Distribution Function (CDF) graph that combines user-generated data with pre-existing data from supercomputers for comprehensive analysis. This will entail displaying the user-provided CSV file alongside the pre-existing supercomputer data. Refer to Fig 1.1, which will be showcased alongside Fig 2.3.

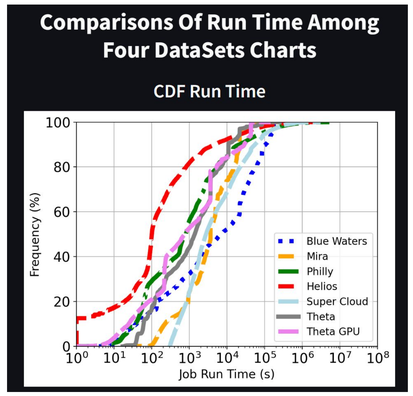


Figure 2.3 highlights the Cumulative Distribution Function (CDF) runtime across seven distinct system models: Blue Waters, Mira, Philly, Helios, Super Cloud, Theta, and Theta GPU.

References

Global Information Assurance Certification Paper - GIAC, www.giac.org/paper/gslc/7633/security-data-visualization/122613. Accessed 29 Nov. 2023.