OSSInsight: Scalable GitHub Analysis

Ahmad Ghazal
PingCAP
Sunnyvale, CA, USA
ahmad.ghazal@pingcap.com

Zhiyuan Liang
PingCAP
Sunnyvale, CA, USA
liangzhiyuan@pingcap.com

Sunny Bains
PingCAP
Sunnyvale, CA, USA
sunny.bains@pingcap.com

Hanumath Maduri
Workday
Pleasanton, CA, USA
hanumath.maduri@workday.com

ABSTRACT

GitHub is a platform hosting code, enabling collaboration, and supporting version control for a global community of over 100 million developers. The need for free tools is crucial for researching open-source software. Based on our research, we found out that existing tools lack real-time GitHub data processing or have limited functionalities.

This demonstration presents OSSInsight (https://www.youtube.com/watch?v=82dn0u8Mgc0), an open source tool for researching and analyzing GitHub repositories. We first present the architecture of the tool including its access to nearly 7 billion archived & real time data and how it is powered by TiDB. The demonstration shows how OSSInsight provides analysis of GitHub data along three dimensions: developers, repositories and organizations. All these analysis are based on generated SQL queries submitted to TiDB database. TiDB possesses HTAP capabilities, utilizing its row store for simple SQL queries while relying on its column store for more complex queries. Users can view and edit these SQL queries and also view their execution plan. Finally, OSSInsight provides an innovative tool based on OpenAI, that conducts data analysis using input in English text, yielding visual representations in the form of charts and graphs.

1 INTRODUCTION

GitHub, a central hub for open-source initiatives, supports 99% of new software projects and is used by 70% of companies. In 2022, over 52 million new open-source projects were initiated, underscoring its crucial role in innovation and technological advancements. GitHub's importance highlights the need for analysis tools in open-source software research. However, tools like GitHub, GHArchive, and CHAOSS may be proprietary, lack real-time processing, or have limited visualization and natural language querying capabilities.

In this demonstration we present OSSInsight, an open source tool that provides insights into GitHub. The main audience of the tool are either developers and organizations researching the best open source project that fits their needs or data analysts who wants to get GitHub metrics.

Before we go over the demonstration details, we describe the architecture of OSSInsight. The architecture covers the ETL process leveraging archived and real time time data sources. TiDB is the back-end which provides storage and compute through SQL queries. OSSInsight is used through a user friendly GUI or a data explorer with plain English text queries. The data explorer component calls OpenAI through a prompt based on the GitHub schema and the English based query. The data explorer gets the SQL back from OpenAI, verifies it and send it to TiDB to process the user request.

The demonstration itself is the core part of this paper and is supported by a video that goes over different use cases including analysis with free text search, analysis by repository, developer and organization. The demonstration also shows how users can view and edit the SQL behind these reports and customize them if desired. Overall, this demo has three main contributions to the database community: (1) free text search to GitHub leveraging OpenAI for SQL generation and reinforced learning for SQL syntax and semantics verification, (2) TiDB as back-end is HTAP capable and can support simple and complex queries issued by OSSInsight and (3) ability for users to write their own SQL to query and analyze GitHub data.

The remainder of this paper is structured as follows: Section 2 reviews existing tools and their limitations in comparison to OSSInsight. Section 3 outlines the overall architecture of OSSInsight. Our primary contribution is detailed in Section 4, where we demonstrate the features and user interaction with OSSInsight. Lastly, Section 5 discusses future work.

2 RELATED WORK

As was discussed in section 1, it is crucial to have tools that mine and analyze GitHub data. Examples of such analysis include: time taken for pull requests, software popularity, and geography and growth of new contributors [3, 5]. In this section, we list tools in this area and show how OSSInsight addresses the limitations and gaps in these tools.

Table 1 compares software development tools based on five crucial metrics: Realtime data from GitHub, Visualization, Natural language interface, Custom SQL, and Open Source. GHArchive and GitHub Innovation Graph rely on historical events[1], while CHAOSS and OSSInsight provide real-time analytics by combining historical events and the GitHub stream API. Proprietary tools like HayStack and LinearB are marketed as Engineering productivity tools, pulling data from various sources. The data sources other than GitHub are not considered due to their lack of relevance to the assessment of open source health.

Another vital aspect is visualization support. CHAOSS relies on a project called Grimoirelab [2] which uses Python libraries for visualization, while proprietary tools like HayStack and LinearB

Table 1: OSSInsight vs other tools

Product	GitHub Realtime Data	Visualisation	Natural Lang Interface	Custom SQL	Open Source
OSSInsight	Y	Y	Y	Y	Y
GitHub Innovation Graph	N	N	N	N	N
CHAOSS	Y	Y	N	Y	Y
LinearB	Y	Y	N	N	N
GHArchive	N	Y	N	Y	Y
HayStack	Y	Y	N	N	N

have visualization support limited to predefined metrics. Again OSSInsight stands out by providing a native support to visualize the result data.

Given the success of generative AI tools, supporting a natural language interface for product usage is crucial. OSSInsight is the only tool that supports querying GitHub using English.

All products have a back-end engine, but those with a relational engine offer ad hoc querying support. The three open source tools distinguish themselves with a relational database back-end, enabling ad hoc queries using ubiquitous SQL. Among them, OSSInsight stands out using a HTAP scalable MySQL-compliant database (TiDB) [4], providing an edge in real-time analytics on open source projects.

In summary, all tools, excluding OSSInsight, have limitations in one or more of the following: free text queries, real-time data processing, visualization options, open-source, and custom SQL query editing.

3 ARCHITECTURE

In this section, we describe the architecture of OSSInsight. Figure 1 depicts the different components of OSSInsight including data sources, back-end, user interfaces and OpenAI as a third party tool. Each component is described at a high level below.

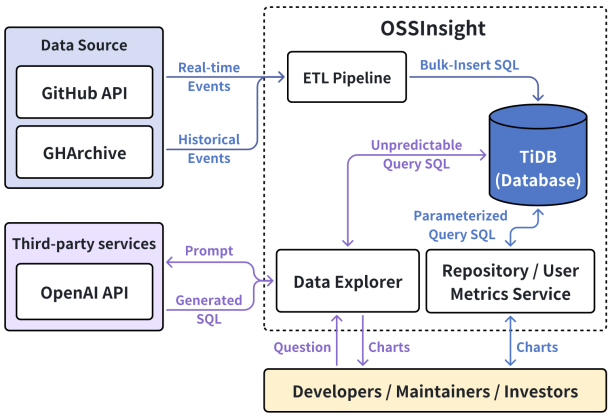

Figure 1: The Architecture of OSSInsight

TiDB

TiDB is a MySQL-compatible database that supports Hybrid Transactional and Analytical Processing (HTAP) capabilities. OSSInsight uses TiDB as a the back-end database to store data and perform data queries. In terms of data analysis queries, TiDB supports accurate secondary indexes for quick lookup like traditional row-based databases, as well as columnar-storage for speeding up large-scale data scans.

Data Source

OSSInsight uses an Extract-Transform-Load (ETL) pipeline to retrieve GitHub event data from two main sources: **GHArchive** and **GitHub API**. GHArchive provides archived historical event data of GitHub from 2011 to today. Initially, historical data were imported into the `github_events` table in TiDB, with hourly updates through a CronJob.

In addition to the historical data, OSSInsight ETL captures real time GitHub events using the GitHub API. The data is also captured in the same TiDB table `github_events`. Periodically, we also use GitHub API to update users and repositories data into the `github_users` and `github_repos` tables respectively.

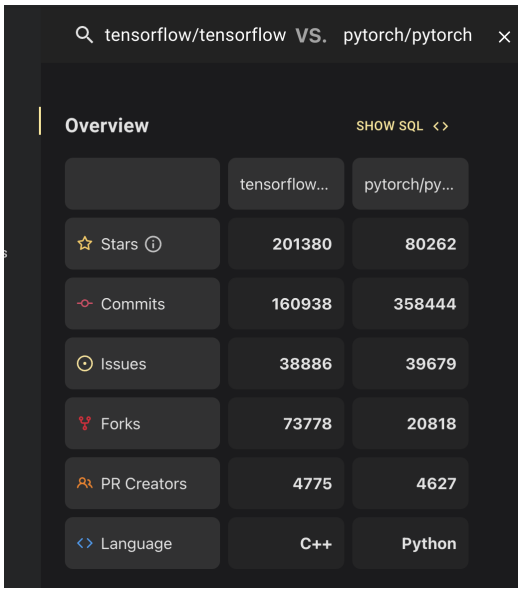

Figure 2: Repository Analysis

Data Explorer

Data Explorer provides users with the feature to perform custom queries in English. When Data Explorer receives the question from the user, it builds a prompt and calls OpenAI. The prompt consists of the relevant table schema and query and the result is a TiDB SQL query. Finally, the data explorer sends the SQL to TiDB which has the intelligence to decide whether to use the row-storage engine, the column-storage engine, or both for query execution. We use multi-agent reinforcement learning to verify and correct the OpenAI results. Currently, we have 80% correctness (correct syntax and semantics) rate for this feature.

Repository / User Metrics Service

The main user interface to OSSInsight is through the repository / user metrics service. This GUI service provides repository-level metrics reporting for open source projects and individual-level dashboards for open source community engagement, with built-in parameterized SQL templates. It populates the SQL templates with the query parameters from user interface to generate metrics SQL and execute them on TiDB, and then visualizes the query results on the front-end to provide users with intuitive charts.

OSSInsight also allows visualization of the SQL and its plan for data explorer and the repository & user metrics service interface. Users can also edit the SQL if necessary. This is more important for the data explorer interface since OpenAI results could be inaccurate and can be corrected directly by the user.

4 DEMO

```sql
SELECT
  `gr`.`repo_name`,
  `gr`.`stars`
FROM
  `github_repos` AS `gr`
  INNER JOIN `github_repo_topics` AS `grt` ON `gr`.`repo_id` = `grt`.`repo_id`
WHERE
  `grt`.`topic` = 'machine-learning'
ORDER BY
  `gr`.`stars` DESC
LIMIT
  3
```

Figure 3: Text Search SQL

The demonstration video can be viewed here (https://www.youtube.com/watch?v=xGlntzMO94w&t=100s) and it starts with an introduction to OSSInsight, its origin, data sources and list of functionalities. The video goes over five different use cases: data explorer, SQL visualization & editing, repository analysis, developer analysis and organization analysis. The remainder of this section describes each use case, illustrating some of user input and output that shows visualisations of metrics and analysis results. For every detailed analysis, the system allows the user to view the SQL (and its plan) that is used for pulling the appropriate data for that specific analysis. Audience of the demonstration can interact with the system similar to what the demonstration video shows through drop down menus and free text queries under "Data Explorer".

Data Explorer

Data explorer allows users to query OSSInsight through text search. The example we used in the video is "Top 3 ML repos by stars"

```
id                          estRows   task      access object
Limit                       3.00      root
└─IndexJoin                 3.00      root
  ├─IndexLookUp(Build)      10611.89  root
  │ ├─IndexFullScan(Build)  10611.89  cop[tikv]  table:gr, index:index_gr_on_stars(stars)
  │ └─TableRowIDScan(Probe) 10611.89  cop[tikv]  table:gr
  └─TableReader(Probe)      3.00      root
    └─Selection             3.00      cop[tikv]
      └─TableRangeScan      1114.16   cop[tikv]  table:grt
```

Figure 4: Text Search Row Store Plan

(see screen shot in Figure 5) and represent a real life example of a user trying to find the best machine learning platform for their project. The user starts with examining top 3 common ML tools and do further analysis of each tool before picking one of them for their solution. The query results in (1) TensorFlow/TensorFlow, (2) HuggingFace/Transformers and (3) Pytorch/Pytorch. As mentioned before, we use OpenAI for SQL generation and we apply validation to the SQL before it is submitted to TiDB. The SQL behind this query are shown in Figure 3.

OSSInsight operates within an HTAP framework, generating simple SQL queries to fetch individual developers or repositories, as well as complex SQL queries to scan entire repositories or organizations. The above query involves a join between `github_repo_topics` and `github_repos` tables on `repo_id`. The TiDB optimizer chose TiKV (row store) to access both tables and used an index join as shown in Figure 4. In the video, we tried a more complex text query "top 50 repos by stars" and the TiDB optimizer decided to use the column store (TiFlash) in TiDB. The plan for the "top 50" is captured in Figure 6 and shows `github_repos` is retrieved by the column store (TiFlash).

Another example that demonstrates varying complexities: comparing the queries "How many pull requests does PingCAP have?" and "Identify the top 20 companies based on the number of pull requests submitted in the last three months." We omit displaying the SQL and the execution plan for these two queries due to space constraints. However, to summarize, the first query is relatively straightforward as it retrieves data for a single organization (PingCAP) and a specific event type ("pull requests"). With appropriate indexing, the row store can effectively handle this query. On the other hand, the second query requires aggregating data from the github_users table, which contains information on all companies. The TiDB optimizer appropriately opted for the column store to process this query.

Repository Analysis

This is available from the search window at the main page of OSSInsight. As a continuation of the use case in data explorer, we compare TensorFlow and PyTorch repositories. The result shows an overview and some details about these two repositories. Figure 2 is a screen shot of the summary results of TensorFlow and PyTorch like stars (how many people bookmarked the project). It also displays other metrics like number of commits, issues, forks and pull requests plus what development language is used.

The detailed results within repository analysis shows metrics and charts. The key detailed metrics are: (1) **People**: stargazers, issue creators and pull request creators, (2) **Commits**: Commits & Pushes History, lines of code changed and commits time distribution, (3) **Pull Requests**: pull request history and pull request time cost, and (4) **Issues**:overview (number of issues, creators, comments and

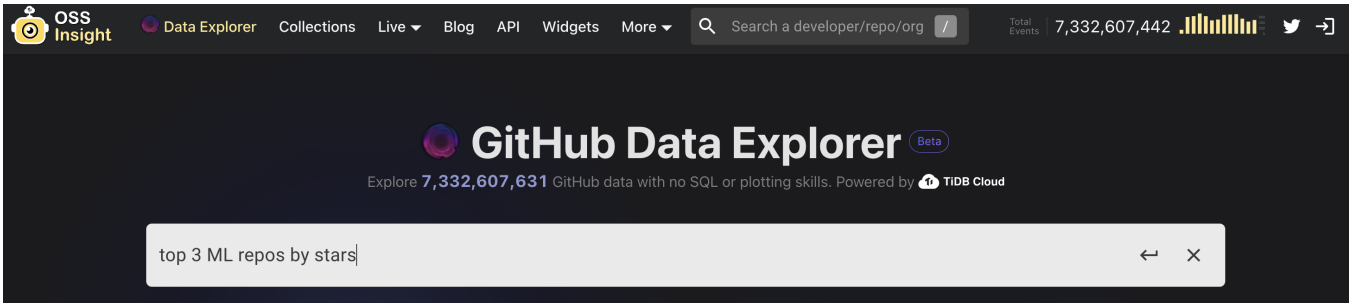

Figure 5: Data Explorer

```
id                          estRows      task          access object
Projection                  50.00        root
└─TopN                      50.00        root
  └─HashAgg                 624835.43    root
    └─IndexHashJoin         11740201.12  root
      ├─TableReader(Build)  663197.05    root
      │ └─ExchangeSender    663197.05    mpp[tiflash]
      │   └─TableFullScan   663197.05    mpp[tiflash]  table:gr
      └─IndexReader(Probe)  11740201.12  root          partition:watch_event
        └─IndexRangeScan    11740201.12  cop[tikv]     table:ge, index:index_ge_on_repo_id
```

Figure 6: Text Search Column Store Plan

comments). The user can make a final choice of which ML tool to use based on the above analysis considering popularity, number of contributors, number of issues, language used, .. etc. One good metric that can be considered here is "language" with TensorFlow using C++ which is complex to use but had good performance. On the other hand, PyTorch uses Python which is simpler than C++ but it is less efficient.

SQL Visualization and Editing

The SQL editing option provided by OSSInsight is currently limited to one table `github_events` with no other limitations. The SQL follows the syntax of TiDB which is MySQL compatible.

In our demo, we tried this option starting from repository analysis for "PingCAP/TiDB". The template is something like "SELECT * FROM github_events WHERE repo_id = 41986369 LIMIT 1" which lists a random GitHub event information for TiDB. We modified it to "SELECT actor_login, count(*) FROM github_events WHERE repo_id = 41986369 group by actor_login order by count(*) desc limit 10" which finds the top 10 contributors to TiDB.

Developer Analysis

OSSInsight can also be used to analyze GitHub data by developer. In the video, we tried the user "feross" as an example for this analysis. High level results for this developer is shown as a table and includes: stars for their repositories & themselves, how many repositories this developer contributed to,how many issues they created, how many pull requests they authored, how many code reviews they conducted and how many lines of code they did. The rest of the analysis drills down into each of the metrics. For example, OSSInsight shows metrics for top 10 repositories "feross" was involved in based on the size of their contribution.

Organization Analysis

Finally, OSSInsight has a feature to analyze organizations in GitHub. In the video we tried "Facebook/Meta" company. Similar to developer analysis, OSSInsight displays summary statistics that include: number of stars earned, number of reviews, number of issues and a list of new participants for repositories related to the organization. Details of these metrics are shown after the summary. For example, popularity of the "Facebook/Meta" is illustrated by showing the star growth of the company over time.

5 FUTURE WORK

In the future, we aim to enhance the user experience of the data explorer by expanding the use of multi-agent reinforcement learning. Our goal is to achieve higher levels of executable queries and correctness. Additionally, we are contemplating the utilization of OSSInsight as a benchmark for Hybrid Transactional/Analytical Processing (HTAP).

6 ACKNOWLEDGEMENTS

The authors express gratitude for the support and guidance on the architecture and functionalities of OSSInsight from Raymond Paik, Lux Li, Wink Yao, and Ed Huang.

7 REFERENCES

REFERENCES

[1] 2024. *Analysing commits on GitHub by @gouv.fr authors – Antoine Augusti.* https://blog.antoine-augusti.fr/2019/04/analysing-commits-on-github-by-gouv-fr-authors/

[2] Santiago Dueñas, Valerio Cosentino, Jesus M. Gonzalez-Barahona, Alvaro del Castillo San Felix, Daniel Izquierdo-Cortazar, Luis Cañas-Díaz, and Alberto Pérez García-Plaza. [n.d.]. GrimoireLab: A toolset for software development analytics. 7, e601 ([n. d.]). https://doi.org/10.7717/peerj-cs.601

[3] Sean Goggins, Kevin Lumbard, and Matt Germonprez. 2021. Open Source Community Health: Analytical Metrics and Their Corresponding Narratives. In *2021 IEEE/ACM 4th International Workshop on Software Health in Projects, Ecosystems and Communities (SoHeal).* 25–33. https://doi.org/10.1109/SoHeal52568.2021.00010

[4] Dongxu Huang, Qi Liu, Qiu Cui, Zhuhe Fang, Xiaoyu Ma, Fei Xu, Li Shen, Liu Tang, Yuxing Zhou, Menglong Huang, Wan Wei, Cong Liu, Jian Zhang, Jianjun Li, Xuelian Wu, Lingyu Song, Ruoxi Sun, Shuaipeng Yu, Lei Zhao, Nicholas Cameron, Liquan Pei, and Xin Tang. 2020. TiDB: a Raft-based HTAP database. *Proc. VLDB Endow.* 13, 12 (aug 2020), 3072–3084. https://doi.org/10.14778/3415478.3415535

[5] Slinger Jansen. 2014. Measuring the Health of Open Source Software Ecosystems: Beyond the Scope of Project Health. *Information and Software Technology* 56 (11 2014). https://doi.org/10.1016/j.infsof.2014.04.006