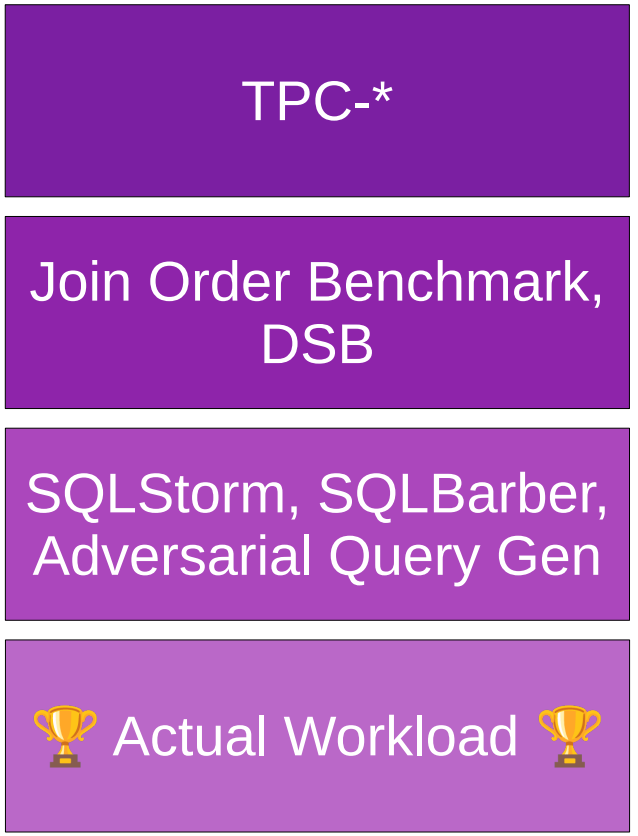
Ryan Marcus, Jeffrey Tao, Peizhi Wu, Zijie Zhao
University of Pennsylvania

https://rm.cab/cidr26

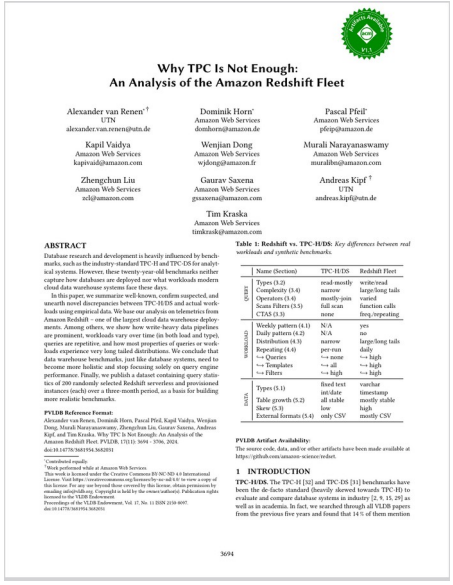
Benchmarks are Key

- How do you evaluate a query optimizer, execution engine, scheduler, cache, etc.?
  - Option 1: Theory
  - Option 2: Benchmarks
- The benchmark is only good when it is *representative*
  - The key assumption: if I improve the benchmark, I'll improve the user's workload!



https://rm.cab/advq

Traces of Actual Workloads



Why TPC Is Not Enough:
An Analysis of the Amazon Redshift Fleet

Alexander van Renen[*†] — UTN — alexander.van.renen@utn.de
Dominik Horn[*] — Amazon Web Services — domhorn@amazon.de
Pascal Pfeil[*] — Amazon Web Services — pfeip@amazon.de
Kapil Vaidya — Amazon Web Services — kapivaid@amazon.com
Wenjian Dong — Amazon Web Services — wjdong@amazon.fr
Murali Narayanaswamy — Amazon Web Services — muralibs@amazon.com
Zhengchun Liu — Amazon Web Services — zcl@amazon.com
Gaurav Saxena — Amazon Web Services — gsxaxena@amazon.com
Andreas Kipf[†] — UTN — andreas.kipf@utn.de
Tim Kraska — Amazon Web Services — timkraska@amazon.com

ABSTRACT

Database research and development is heavily influenced by benchmarks, such as the industry-standard TPC-H and TPC-DS for analytical systems. However, these twenty-year-old benchmarks neither capture how databases are deployed nor what workloads modern cloud data warehouse systems face these days.

In this paper, we summarize well-known, confirm suspected, and unearth novel discrepancies between TPC-H/DS and actual workloads using empirical data. We base our analysis on telemetries from Amazon Redshift – one of the largest cloud data warehouse deployments. Among others, we show how write-heavy data pipelines are prominent, workloads vary over time (in both load and type), queries are repetitive, and have most properties of queries or workloads experience very long tailed distributions. We conclude that data warehouse benchmarks, just like database systems, need to become more holistic and stop focusing solely on query engine performance. Finally, we publish a dataset containing query statistics of 200 randomly selected Redshift serverless and provisioned instances (each) over a three-month period, as a basis for building more realistic benchmarks.

PVLDB Reference Format:
Alexander van Renen, Dominik Horn, Pascal Pfeil, Kapil Vaidya, Wenjian Dong, Murali Narayanaswamy, Zhengchun Liu, Gaurav Saxena, Andreas Kipf, and Tim Kraska. Why TPC Is Not Enough: An Analysis of the Amazon Redshift Fleet. PVLDB, 17(11): 3694 - 3706, 2024.
doi:10.14778/3681954.3682031

[*]Contributed equally
[†]Work performed while at Amazon Web Services

Proceedings of the VLDB Endowment, Vol. 17, No. 11 ISSN 2150-8097.
doi:10.14778/3681954.3682031

Table 1: Redshift vs. TPC-H/DS: *Key differences between real workloads and synthetic benchmarks.*

	Name (Section)	TPC-H/DS	Redshift Fleet
SCHEMA	Types (3.2)	read-mostly	write/read
	Complexity (3.4)	narrow	large/long tails
	Operators (3.4)	mostly-join	varied
	Scans Filters (3.5)	full scan	function calls
	CTAS (3.3)	none	freq./repeating
WORKLOAD	Weekly pattern (4.1)	N/A	yes
	Daily pattern (4.2)	N/A	no
	Distribution (4.3)	narrow	large/long tails
	Repeating (4.4)	per-run	daily
	↪ Queries	↪ none	↪ high
	↪ Templates	↪ all	↪ high
	↪ Filters	↪ high	↪ high
DATA	Types (5.1)	fixed text int/date	varchar timestamp
	Table growth (5.2)	all stable	mostly stable
	Skew (5.3)	low	high
	External formats (5.4)	only CSV	mostly CSV

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/amazon-science/redset.

1 INTRODUCTION

TPC-H/DS. The TPC-H [32] and TPC-DS [31] benchmarks have been the de-facto standard (heavily skewed towards TPC-H) to evaluate and compare database systems in industry [2, 9, 15, 29] as well as in academia. In fact, we searched through all VLDB papers from the previous five years and found that 14 % of them mention

Workload Insights From The Snowflake Data Cloud: What Do Production Analytic Queries Really Look Like?

Jan Vincent Szlang — Snowflake, Inc. — Berlin, Germany
Jonathan Dees — Snowflake, Inc. — Berlin, Germany
Michel Oleynik — Snowflake, Inc. — Berlin, Germany
Sebastian Bress — Snowflake, Inc. — Berlin, Germany
Florian Funke — Snowflake, Inc. — Berlin, Germany
Ismail Oukid — Snowflake, Inc. — Berlin, Germany
Sebastian Cattes — Snowflake, Inc. — Berlin, Germany
Max Heimel — Snowflake, Inc. — Berlin, Germany
Tobias Maltenberger[*] — Snowflake, Inc. — Berlin, Germany
firstname.lastname@snowflake.com

ABSTRACT

Capturing the characteristics of real-world analytical workloads is challenging yet critical for advancing industry practices and academic research. Historically, obtaining accurate query and data characteristics has been difficult, largely because detailed workload information has often been confined to on-premises database systems. With the rise of cloud-native databases like Snowflake, it has become possible to analyze production query workloads at scale and in greater detail. Leveraging this capability, this study presents a comprehensive analysis of analytics workloads across diverse customers and industries. In particular, we investigate the query characteristics of 667 million queries issued by the most popular BI tools against Snowflake over a two-week period. Based on this dataset, this paper makes two primary contributions: first, we conduct a detailed examination of query properties, with particular attention to filters, joins, aggregations, and other previously underexplored aspects. Second, we uncover unique and practically relevant query patterns that are typically absent from standard database benchmarks.

PVLDB Reference Format:
Jan Vincent Szlang, Sebastian Bress, Sebastian Cattes, Jonathan Dees, Florian Funke, Max Heimel, Michel Oleynik, Ismail Oukid, and Tobias Maltenberger. Workload Insights From The Snowflake Data Cloud: What Do Production Analytic Queries Really Look Like?. PVLDB, 18(12): 5126 - 5138, 2025.
doi:10.14778/3750601.3750632

1 INTRODUCTION

Understanding and addressing real-world problems is a cornerstone of database research. Yet, gaining visibility into relevant workloads remains a significant challenge. Most real-world query patterns are locked away in proprietary applications or isolated within private infrastructure, limiting visibility and hindering a deeper understanding of how database systems are actually used. To address this lack of visibility, the database community relies on synthetic benchmarks such as TPC-H [26], TPC-DS [25], or ClickBench [6], which aim to replicate the complexity and characteristics of real-world data processing tasks. While these benchmarks offer a valuable framework for evaluating database systems, they fall short of capturing the full diversity and dynamics of actual workloads, leaving many critical patterns unexplored [11, 26, 29].

The emergence of cloud-based data platforms such as Amazon Redshift [3, 15], Google Big Query [20], and Snowflake [9] is surfacing previously inaccessible insights into real-world workloads. Unlike traditional on-premise systems, these platforms operate as a service, enabling providers to collect detailed telemetry on how customers query and interact with their data. This offers unprecedented visibility into workload patterns, helping providers identify key challenges and prioritize engineering efforts that directly address real-world customer needs. Furthermore, some cloud providers have begun sharing workload traces and other insights with the research community, fostering collaboration and validating emerging ideas [26, 38].

In this paper, we share insights gained from over a decade of experience with analytical workloads at Snowflake. Specifically, we analyze a representative production workload consisting of 667 million analytical queries executed by our customers. Using this dataset, we provide an overview of key workload metrics and conduct an in-depth analysis of SQL query patterns, examining their characteristics and frequency in real-world scenarios. Additionally, we identify query patterns that are significant for industrial applications but are not adequately captured in standard benchmarks, highlighting opportunities to refine evaluation frameworks. These findings offer valuable perspectives for understanding real-world workloads and open up new directions for future research.

Our contributions are two-fold:

(1) We analyze a representative analytical workload, focusing on SQL patterns and other properties that have not been extensively studied in the context of cloud data platforms.
(2) By comparing our findings with the TPC-DS benchmark, we highlight opportunities to broaden its scope, offering recommendations to better capture the diversity and complexity of modern analytical workloads.

The primary focus of our study is to highlight how analytical queries are structured and shaped in practice. This emphasis on detailed query properties complements recent workload analyses – such as the analysis of the Redset dataset published by van Renen et al. [26] – which have primarily investigated data characteristics (e.g., table properties, data skew, and value distributions) and broader workload trends. By explicitly analyzing real-world SQL query texts

[*]Work of Tobias Maltenberger was done when employed at Snowflake Inc., he is now working at Google.

Proceedings of the VLDB Endowment, Vol. 18, No. 12 ISSN 2150-8097.
doi:10.14778/3750601.3750632

Presto: A Decade of SQL Analytics at Meta

Yutian "James" Sun, Tim Meehan, Rebecca Schlussel, Wenlei Xie, Masha Basmanova, Orri Erling, Andrii Rosa, Shixuan Fan, Rongrong Zhong, Arun Thirupathi, Nikhil Collooru, Ke Wang, Sameer Agarwal, Arjun Gupta, Dionysios Logothetis, Kostas Xirogiannopoulos, Bin Fan[‡], Amit Dutta, Varun Gajjala, Rohit Jain, Ajay Palakuzhy, Prithvi Pandian, Sergey Pershin, Abhisek Saikia, Pranjal Shankhdhar, Neerad Somanchi, Swapnil Tailor, Jialiang Tan, Sreeni Viswanadha, Zac Wen, Deepak Majeti[‡], Aditi Pandit[‡], Biswapesh Chattopadhyay
Meta Platform, Inc[†] †Alluxio, Inc ‡Ahana Cloud, Inc

ABSTRACT

Presto is an open-source distributed SQL query engine that supports analytics workloads involving multiple exabyte-scale data sources. Presto is used for low-latency interactive use cases as well as long-running ETL jobs at Meta. It was originally launched at Meta in 2013 and donated to the Linux Foundation in 2019. Over the last ten years, upholding query latency and scalability with the hyper growth of data volume at Meta as well as new SQL analytics requirements have raised impressive challenges for Presto. A top priority has been ensuring query reliability does not regress with the shift towards smaller, more elastic container allocation, which requires queries to run with substantially smaller memory headroom and can be preempted at any time. Additionally, new demands from machine learning, privacy, and graph analytics have driven Presto maintainers to think beyond traditional data analytics. In this paper, we discuss several successful evolutions in recent years that have improved Presto latency as well as scalability by several orders of magnitude in production at Meta. Some of the notable ones are hierarchical caching, native vectorized execution engines, materialized views, and Presto on Spark. With these new capabilities, we have deprecated or are in the process of deprecating various legacy query engines so that Presto becomes the single, unified interactive, ad-hoc, ETL, and graph processing workloads for the entire data warehouse.

CCS CONCEPTS

• **Information systems** → **Database query processing**; **Parallel and distributed DBMSs**; **Online analytical processing engines**.

KEYWORDS

Data Warehouse, Presto, OLAP, SQL, Distributed Database, Data Analytics, ETL

ACM Reference Format:
Yutian "James" Sun, Tim Meehan, Rebecca Schlussel, Wenlei Xie, Masha Basmanova, Orri Erling, Andrii Rosa, Shixuan Fan, Rongrong Zhong, Arun Thirupathi, Nikhil Collooru, Ke Wang, Sameer Agarwal, Arjun Gupta, Dionysios Logothetis, Kostas Xirogiannopoulos, Bin Fan[‡], Amit Dutta, Varun Gajjala, Rohit Jain, Ajay Palakuzhy, Prithvi Pandian, Sergey Pershin, Abhisek Saikia, Pranjal Shankhdhar, Neerad Somanchi, Swapnil Tailor, Jialiang Tan, Sreeni Viswanadha, Zac Wen, Deepak Majeti[‡], Aditi Pandit[‡], Biswapesh Chattopadhyay. 2023. Presto: A Decade of SQL Analytics at Meta. In *Proceedings of To Proceedings of the 2023 International Conference on Management of Data (SIGMOD '23).* ACM, New York, NY, USA, 14 pages. https://doi.org/XXXXXXX.XXXXXXX

SIGMOD '23, June 18–23, 2023, Seattle, WA, USA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

[‡]Authors without superscripts are still or used to contribute to Presto at Meta

1 INTRODUCTION

Presto [44] is an open-source distributed query engine that has supported production analytical workloads at Meta since 2013. It provides a SQL interface to query data stored on different storage systems, such as distributed file systems. Since its donation to Linux Foundation in 2019, Presto has continued growth in utilization and contribution among US tech industry leaders including Uber, Twitter, Intel, Ahana, etc. After the donation, Meta remains active in Presto contributions with 50% commits coming from Meta. The deployment of the Presto fleet at Meta is also on top of the trunk to ensure every release is battle-tested at Meta scale.

Within Meta, Presto is used for interactive, ad-hoc, and extract-transform-load (ETL) workloads at scale. Use cases include dashboarding, A/B testing, ad-hoc analysis, data cleaning, and transformation. With the effort of migrating all SparkSQL [6] workloads to Presto at Meta, Presto will soon provide the only SQL interface to the warehouse in the company.

While Presto was originally designed for exclusively in-memory processing of interactive SQL querying, various trends at Meta challenged its capabilities. Due to its efficiency, employees started to use it for lightweight ETL workloads [46] that run for up to tens of minutes, and, as data grew exponentially, Presto became slower. The transition to a more flexible and elastic resource management model with smaller, ephemeral containers resulted in reduced reliability. Moreover, while there was a growing demand for richer analytics, such as machine learning feature engineering, and graph analytics, they were not well supported. Finally, honoring Meta's data privacy policies required new data abstractions and data storage mechanisms to support privacy enforcement efficiently. The main focus of this paper is to describe how we have improved the architecture of Presto to address these challenges with the following three perspectives.

First, **latency and efficiency**. As data increases, the scan cost of the same query increases leading to longer wait time. As the

AWS Redshift
Analytic workloads are repetitive!

Snowflake
People join on string columns!

Meta
ML data (lists, non-1NF!) is big

Survivorship Bias

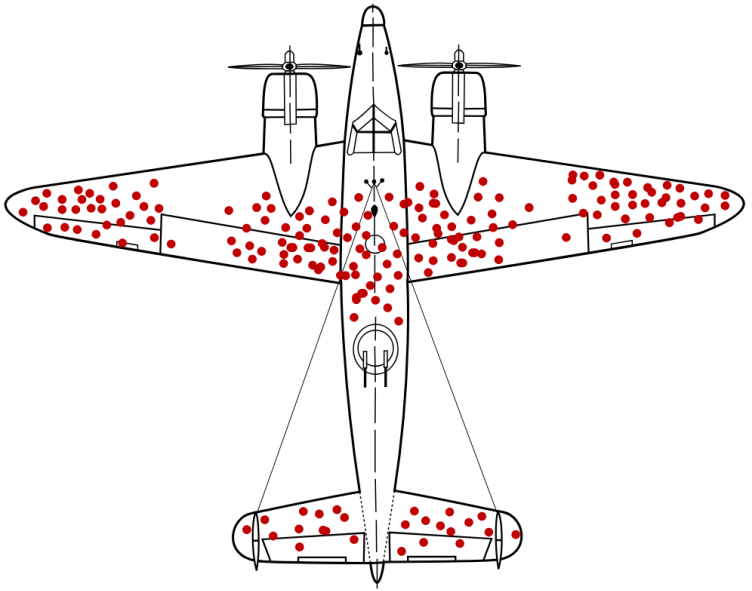
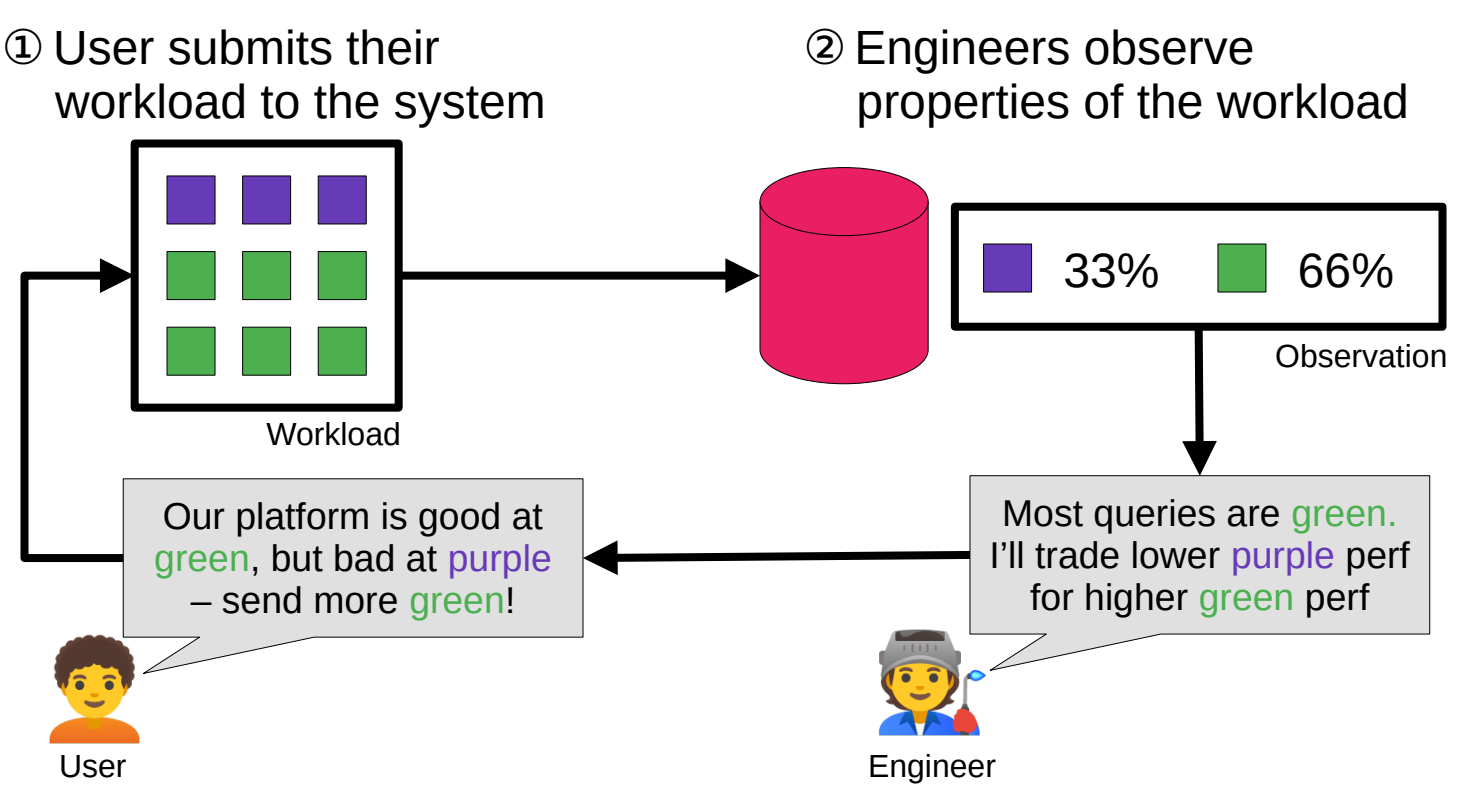


Figure 1: A diagram of bullet holes on returning WWII fighter planes. Since these planes all returned, the sample is biased towards "survivors." One might think that additional armor should be added to the shot areas, but a more effective strategy is to add armor to the areas that appear unscathed.

This Talk

- **A theory of survivorship bias in workloads**
- Some tests of that theory
- Analysis and insights



The queries you run impact what the engineers build.
What the engineers build impacts the queries you run!

This Talk

- A theory of survivorship bias in workloads
- **Some tests of that theory**
- Analysis and insights

Does this loop actually happen?

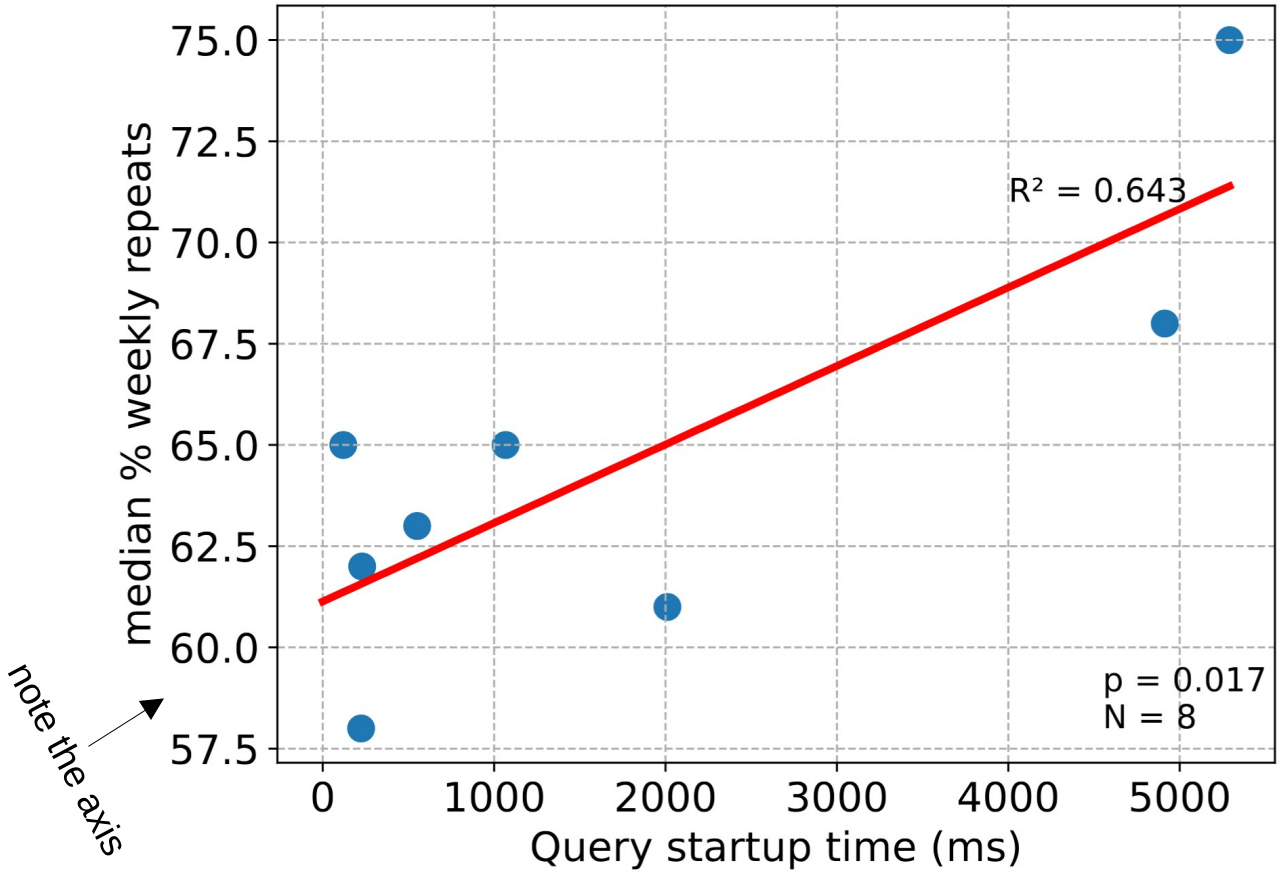
In a perfect world…

- A vendor could artificially slow down a certain type of query to see if their customers stop issuing them.
- A vendor could invest time optimizing something their customers don’t currently do.
- Seeking volunteers.

We’ll settle for…

- Natural experiments!
- Vendors that are better at processing ad-hoc queries should see more of them.
- A vendor that introduces a fast feature should see uptake from users

Exp. 1: Repetitive Queries

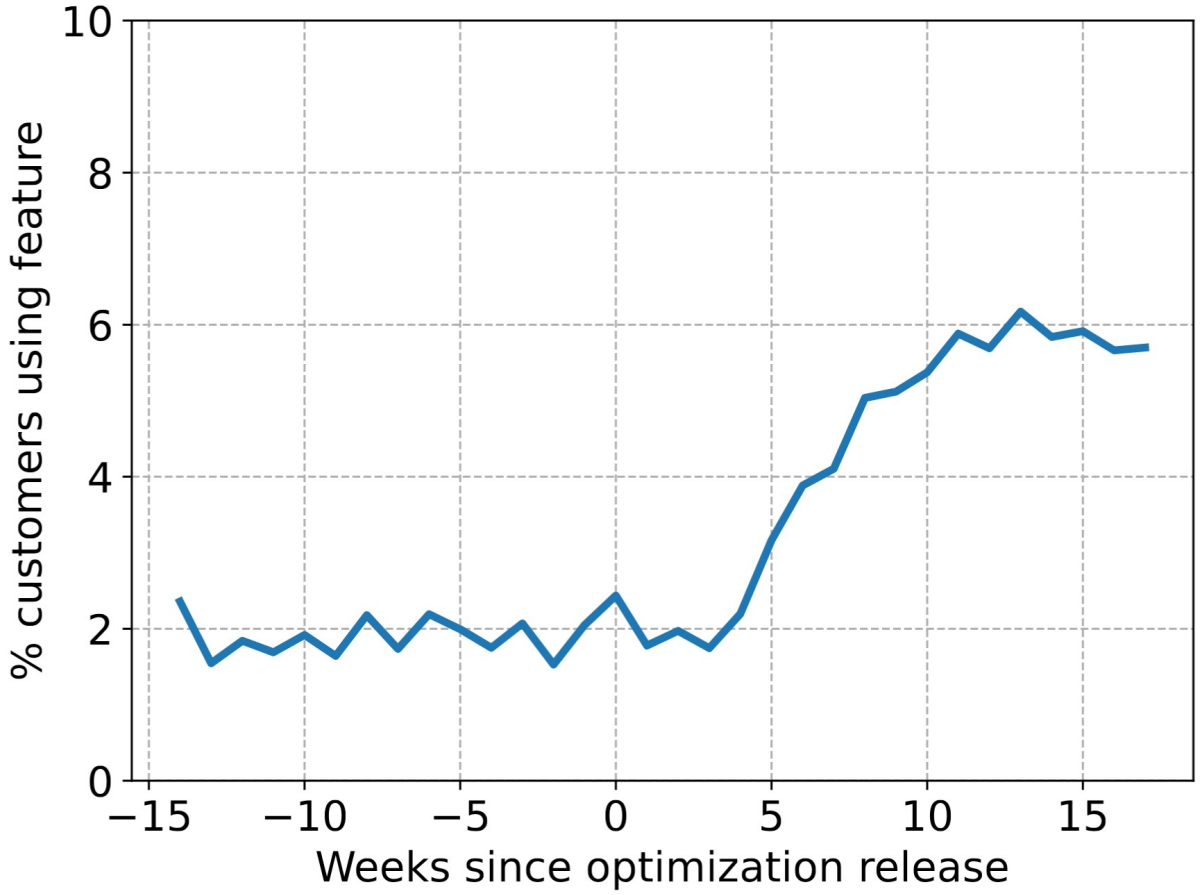


Cold query startup time vs. % repeated queries

Data platforms with longer query startup see more repetitive queries.

Could be platform choice or optimization over time.

Exp. 2: Build it and they will come



% users of a particular feature after optimization

Only 2% usage before optimization, but 6% usage afterwards!

Could be “release notes” teaching about the feature

Survivorship Bias

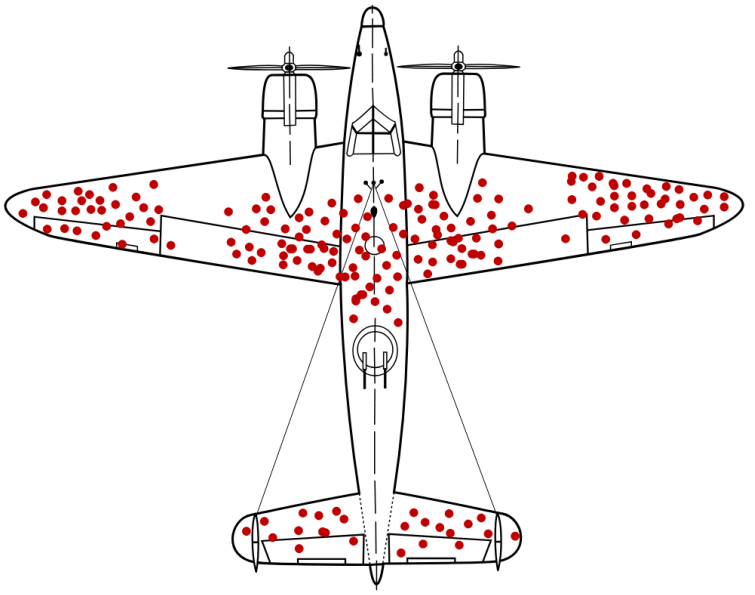
- The DBMS influences what queries the user sends
  - Users use fast features, don’t use slow ones
- The user’s queries influence the DBMS
  - Engineers optimize for what the user sends
- Workload traces *only capture queries that “survive” this negotiation!*

This Talk

- A theory of survivorship bias in workloads
- Some tests of that theory
- **Analysis and insights**

Are we doing it wrong?

- By optimizing existing user workloads, are we putting armor on the wrong part of the plane?
- A customer continuously issuing a query is a certificate of “good enough.”

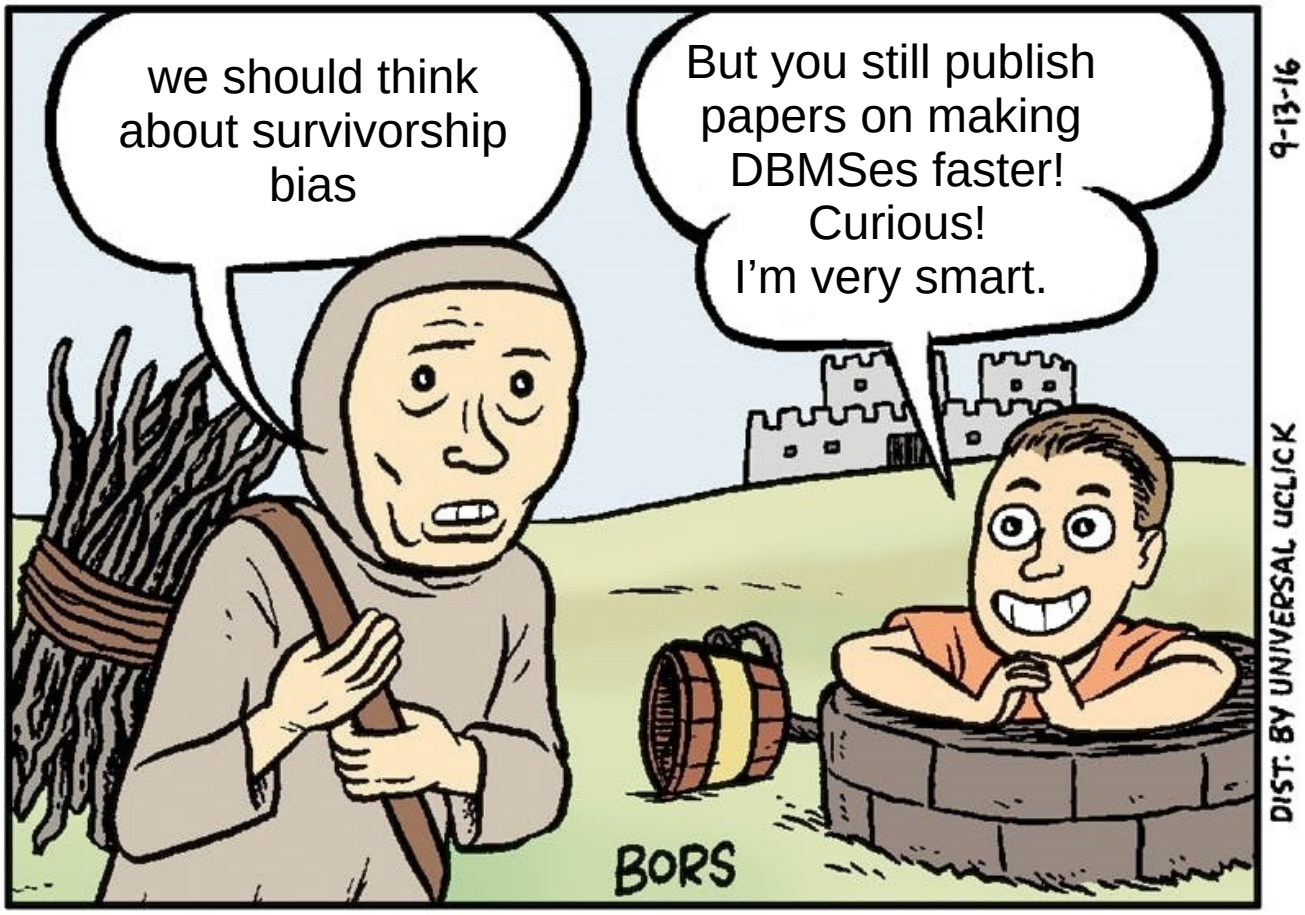


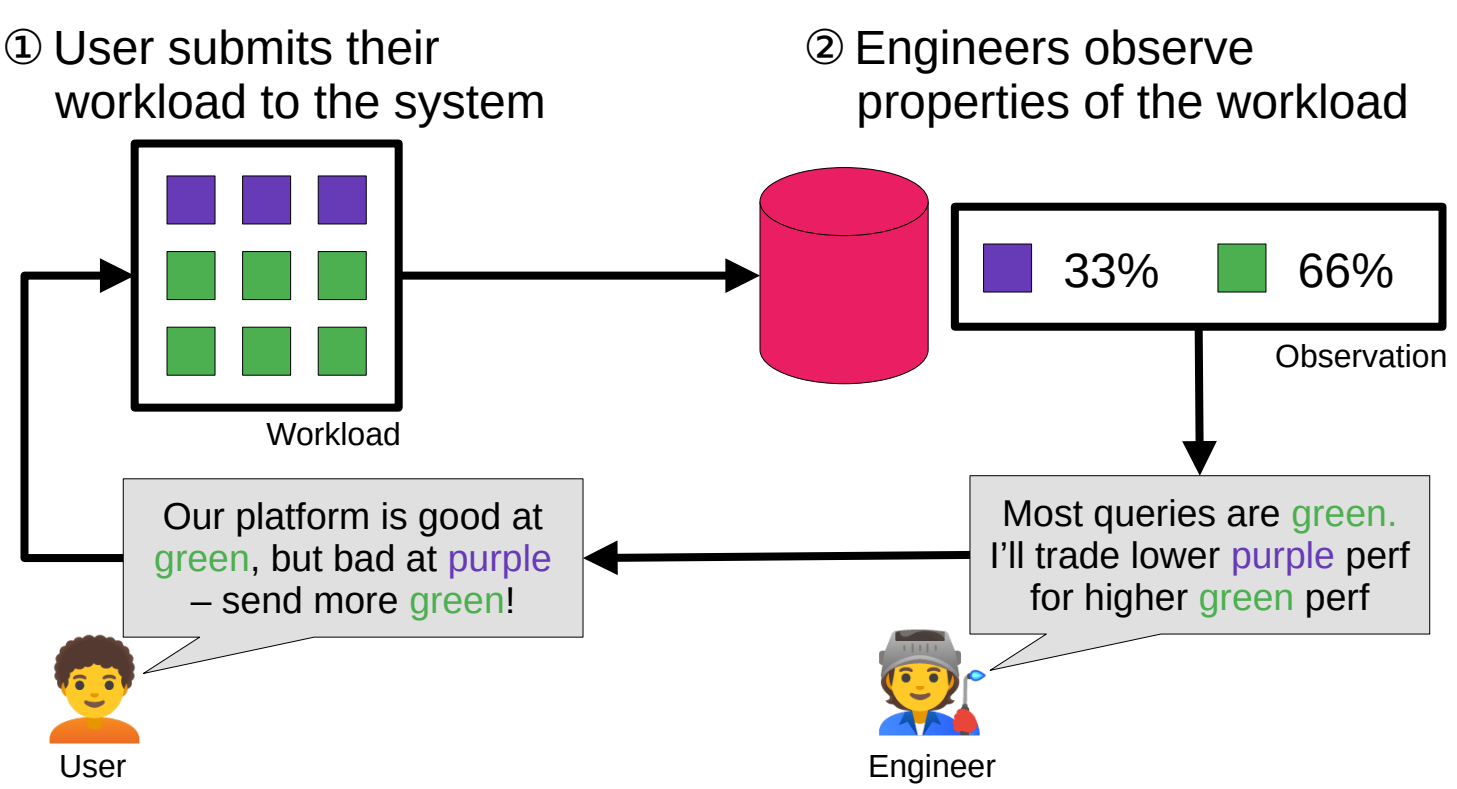
Thought experiment

- Imagine column stores were never invented, and you run a cloud database.
- How many long, OLAP style queries do you think you'd see compared to Redshift?

Faster DBMSes are still good

- Faster DBMSes *are still good*
- … basing that opt work on *real traces* is WAY better than making stuff up





How can we optimize the query the user *wants* to run, but can't?