



Super-performant dashboards with dbt, Firebolt and Looker

Mark Rittman CEO, Rittman Analytics & Son N. Nguyen, Analytics Engineer, Hiflylabs
May 2022, Budapest, Hungary

Introductions



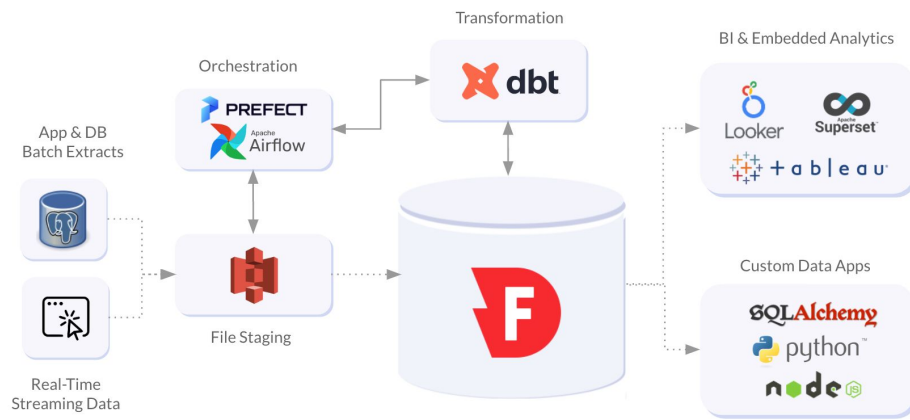
Mark Rittman - Rittman Analytics
mark.rittman@rittmananalytics.com
CEO



Son N. Nguyen - Hiflylabs
son.nguyen.nam@hiflylabs.com
Analytics Engineer

What Is Firebolt? And Why Is it Interesting?

- Firebolt is a new data warehousing platform combining **scale + low-latency queries**
- Startup founded by ex-Sisense execs, team of ex-Google/ex-Looker execs/engineers
- Purpose-built to provide sub-second querying on terabyte-scale data sets
- Integration with Looker, dbt, Prefect, Tableau + Python, Node etc
- Modern scalable cloud architecture
 - Data Warehouse-as-a-Service
 - Separation of Storage and Compute
 - Extreme Performance at Scale
 - Match Engines to Workloads
 - Integration using JDBC and RESTful API

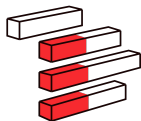


How Does Firebolt Work?



Objective : move & scan much less data

Most queries look for very small and specific parts of it. Moving and scanning full partitions is a huge performance bottleneck.



Accelerator : sparse indexing

Sparse indexes point to small data ranges within files, which can be scanned individually. Aggregating and Join Indexes accelerate expensive queries by pre-calculating results



Enabler : Firebolt F3 Storage Format

To enable sparse indexing, data is stored pre-sorted and compressed.

```
CREATE TABLE fact_round (  
    event_id,  
    date,  
    ...  
) PRIMARY INDEX event_id, date, customer_id
```

Aggregating Indexes and Join Indexes

Aggregating Index

Materialized views stored with sparse indexing

- Great for repeating workloads such as dashboards/reports
- Defined on the table once, and managed automatically. Add as many as needed.

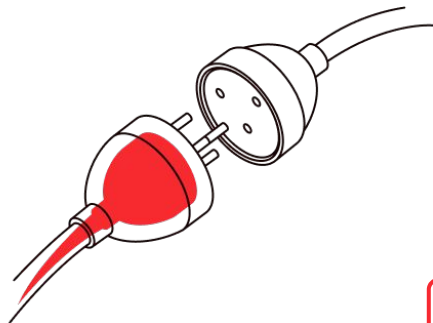
```
CREATE AGGREGATING INDEX idx_agg_rounds ON fact_round
(  
  game_code,  
  player_code,  
  currency_code,  
  count(distinct round_id),  
  sum(credit),  
  avg(credit),  
  sum(debit),  
  sum(total_events)  
)
```

Join Index

Pre-computes expensive fact/dimension table joins

- No need to denormalize schemas to avoid join slowdowns
- Join indexes are RAM stored objects that accelerate joins dramatically
- Can work together with aggregating indexes

```
CREATE JOIN INDEX idx_join_games ON dim_games
(  
  game_code,      -- join column  
  game_studio,   -- dim column  
  game_currency  -- dim column  
)
```



Beta

Firebolt's **dbt adapter**

Accelerated Analytics

Deliver **sub-second analytics** to your users and enjoy faster, more frequent, and more cost efficient model builds.

Infrastructure-as-Code

Create and manage **all Firebolt index types** using dbt's version-controlled model configurations.

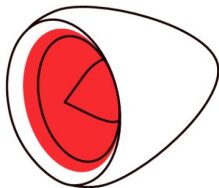
Simplified Data Ingestion

Ingest data from S3 using dbt-external-tables married with Firebolt's external tables.



Firebolt and Looker

- Looker dashboards powered by Firebolt typically run 3x - 10x faster than before
- Potential for sub-second query performance - “analytics at the speed of thought”



33-300x faster

According to customers' own benchmarks, Looker dashboards and queries run 33-300x faster on Firebolt than on Snowflake, Redshift or Athena.



Unlimited scale

With Firebolt, companies have been able to sub-seconds performance for dashboards at gigabyte to petabyte scale, and unlimited room for growth.




5-10x lower cost

Firebolt's 10x greater efficiency, choice of any node type and number, and price transparency has enabled Looker customers to get sub-seconds performance at 5-10x lower cost.

So how does it work?

Hacker News Public Dataset as Example Scenario



Hacker News

Y Combinator

Stories and comments since 2006

[VIEW DATASET ↗](#)

[OVERVIEW](#) [SAMPLES](#)

Overview

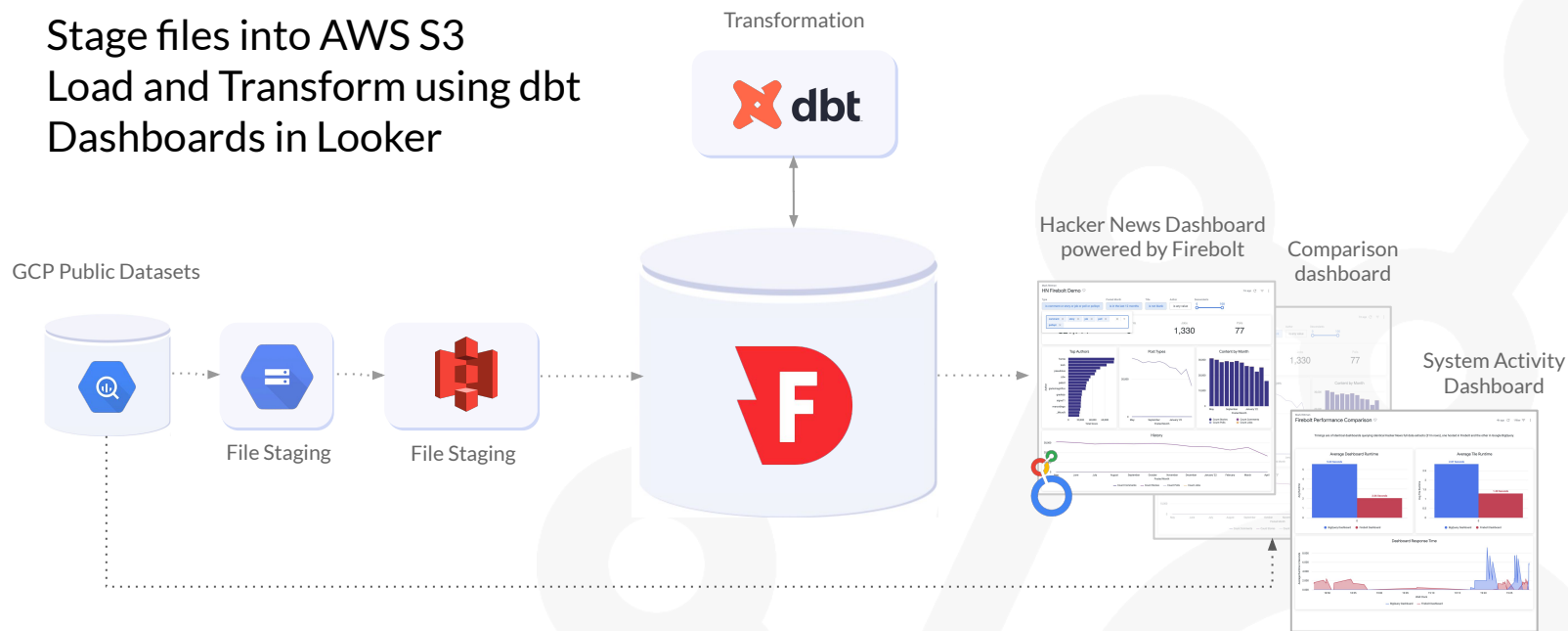
This dataset contains all stories and comments from Hacker News from its launch in 2006 to present. Each story contains a story ID, the author that made the post, when it was written, and the number of points the story received.

[Y](#) **Hacker News** [new](#) | [past](#) | [comments](#) | [ask](#) | [show](#) | [jobs](#) | [submit](#)

- ▲ [Show HN: PostgresML, now with analytics and project management](#) (postgresml.org)
236 points by levkk 4 hours ago | hide | 38 comments
- ▲ [Personal Knowledge Management Is Bullshit](#) (otherlife.co)
70 points by leephillips 2 hours ago | hide | 46 comments
- ▲ [Bored Ape Virtual Land Sale Breaks Ethereum, Wastes \\$180M in Fees](#) (vice.com)
174 points by danso 1 hour ago | hide | 245 comments
- ▲ [An engineered barley plant that 'orders' soil bacteria to manufacture fertiliser](#) (cam.ac.uk)
50 points by montalbano 1 hour ago | hide | 16 comments
- ▲ [The \\$440M software error at Knight Capital \(2019\)](#) (henricodolffing.com)
112 points by bfm 4 hours ago | hide | 62 comments
- ▲ [An illustrated guide to plastic straws \(2021\)](#) (hwfo.substack.com)
197 points by worldvoyageur 4 hours ago | hide | 132 comments
- ▲ [Ask HN: Who is hiring? \(May 2022\)](#)
294 points by whoishiring 7 hours ago | hide | 551 comments
- ▲ [Square-Enix sells all of its Western game studios and their games to Embracer](#) (arstechnica.com)
209 points by zdw 7 hours ago | hide | 186 comments
- ▲ [Queenly \(YC W21\) Is Hiring](#) (ycombinator.com)
1 hour ago | hide
- ▲ [SF Conservancy now accepting copyright assignment for any GPL software](#) (sfconservancy.org)
14 points by logic 49 minutes ago | hide | discuss
- ▲ [Grindr user data has been for sale for years](#) (wsj.com)
172 points by pondsider 8 hours ago | hide | 94 comments
- ▲ [The appeal of using plain HTML pages](#) (utoronto.ca)
179 points by todsacerdoti 7 hours ago | hide | 142 comments
- ▲ [Show HN: I am building a free version of Strava](#) (mtbx.bike)
189 points by rlrhaeck 4 hours ago | hide | 50 comments
- ▲ [Configurator 1.0: Common Lisp based declarative configuration management system](#) (spwhittor)
30 points by pabs3 3 hours ago | hide | discuss

Scenario

- Extract from GCP via GCS
- Stage files into AWS S3
- Load and Transform using dbt
- Dashboards in Looker



Installation and Setup of dbt-Firebolt Adapter

1

```
pip install dbt-firebolt
```

2

```
firebolt_hackernews:  
  target: default  
  outputs:  
    default:  
      type: firebolt  
      user: "{{ env_var('FIREBOLT_USER') }}"  
      password: "{{ env_var('FIREBOLT_PASSWORD') }}"  
      database: dbt_meetup_hackernews  
      schema: wh  
      engine_name: dbt_meetup_hackernews_general_purpose  
      threads: 1  
      account_name: hiflylabs
```

3

Feature	Supported
Table materializations	✓
Ephemeral materializations	✓
View materializations	✓
Incremental materializations - append	✓
Incremental materializations - insert_overwrite	✗
Incremental materializations - merge	✗
Snapshots	✗
Seeds	✓
Tests	✓
Documentation	✓
Custom schemas	✗ (see workaround)
Custom databases	✗
Source freshness	✓
External tables	✓
Primary indexes	✓
Aggregating indexes	✓
Join indexes	✗ (syntax supported, but not effective)

4

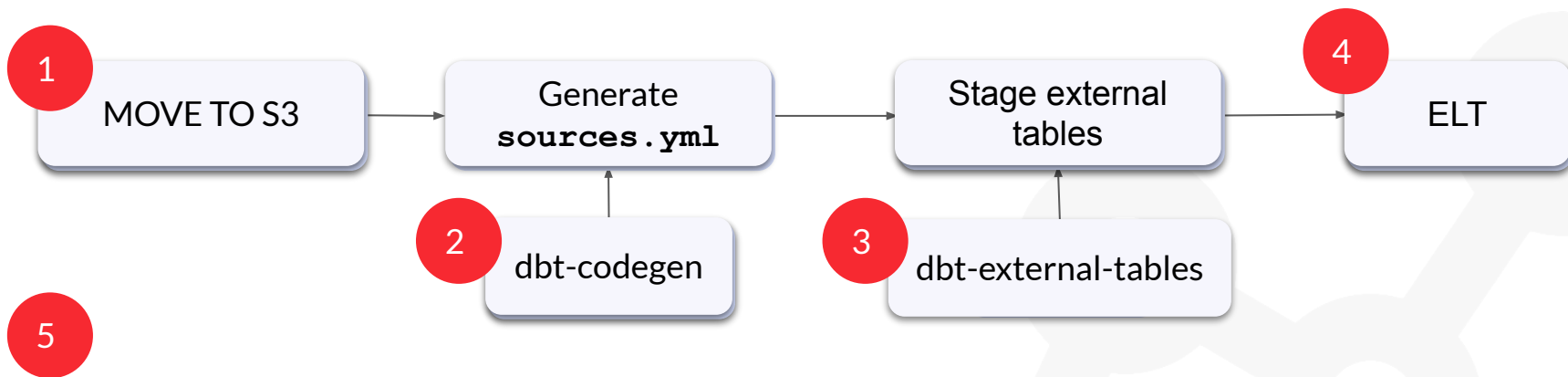
Setup Recommendations

For the best experience we recommend that you make the following changes to your dbt project:

- Set an explicit value for `quote_columns`
- Add the `generate_alias_name` macro to your project



Implementation



```
view: hn_comments_fact {
```

```
  sql_table_name: {% if _user_attributes['connection'] == 'firebolt' %}
```

```
    ANALYTICS.WH_COMMENTS_FACT
```

```
  {% else %}
```

```
    {{ _user_attributes['dbt_database'] }}.{{ _user_attributes['dataset'] }}.COMMENTS_FACT
```

```
  {% endif %}
```



Accessing data from S3 - external tables

```
version: 2
sources:
- name: firebolt_external
  schema: "{{ target.schema }}"
  tables:
  - name: hn_comments_ext
    external:
      url: 's3://ra-hacker-news/comments/'
      object_pattern: '*.parquet'
      type: '(parquet)'
      credentials:
        internal_role_arn: "{{ env_var('INTERNAL_ROLE_ARN')
        }}"
        external_role_id: "{{ env_var('EXTERNAL_ROLE_ID')
        }}"
    columns:
    - name: ID
      data_type: INTEGER
```

```
dbt run-operation stage_external_sources --vars "ext_full_refresh:
true"
21:05:33 Running with dbt=1.1.0
21:06:36 1 of 1 START external source wh.hn_comments_ext
21:06:36 1 of 1 (1) DROP TABLE IF EXISTS hn_comments_ext
21:06:36 1 of 1 (1) OK
21:06:36 1 of 1 (2) -- CREATE EXTERNAL TABLE hn_comments_ext ("ID"
INTEGER, "BY..."
21:06:38 1 of 1 (2) OK
```

Materialization and Indexing

```
{{
config(
  materialized = 'table',
  alias = 'comments_fact',
  table_type = 'fact',
  primary_index = ['"YEAR"', '"MONTH"', '"AUTHOR"', '"STORY_ID"'],
  indexes = [
    {
      'index_type': 'aggregating',
      'key_column': ['"YEAR"', '"MONTH"', '"AUTHOR"', '"STORY_ID"'],
      'aggregation': ['"SUM(RANKING)"', '"AVG(RANKING)"', '"COUNT(*)"']
    }
  ]
)
}}
with source as (
  select *
  from {{ source('firebolt_external', 'hn_comments_ext') }}
)
select * from source
```

Table/View/Ephemeral materializations	✓
Incremental materializations	(✗)
Snapshots	✗
Custom Databases/Schemas	✗
Source freshness	✓
External tables	✓
Primary indexes	✓
Aggregating indexes	✓
Join indexes	✗

Indexes in Action

Without aggregating index (0.67s)

```
[0] [Projection] COUNT(FB_NODE_3.*) @ FB_NODE_1
  \_[1] [Aggregate] GroupBy: [FB NODE 3.MONTH] Aggregates: [COUNT(FB_NODE_3.*)] @ FB_NODE_2
    \_[2] [StoredTable] Name: 'comments_fact', used 1/13 column(s) FACT @ FB_NODE_3
```

With aggregating index (0.02s)

```
[0] [Alias] COUNT(FB_NODE_4.COUNT(*)) AS COUNT(1) @ FB_NODE_1
  \_[1] [Projection] COUNT(FB_NODE_4.COUNT(*)) @ FB_NODE_2
    \_[2] [Aggregate] GroupBy: [FB_NODE_4.MONTH] Aggregates: [COUNT(FB_NODE_4.COUNT(*))] @ FB_NODE_3
      \_[3] [StoredTable] Name: 'comments_fact_agg_index__FB_AGG_IDX_MV__TABLE', used 2/7 column(s)
AGGREGATING_INDEX_TABLE @ FB_NODE_4
```

Demo

- Explore
- Develop
- Admin

- Home
- Recently Viewed
- Favorites
- Boards
 - Operations & Finance
 - Production
 - Witness Analytics XPIs and...
 - File
- Folders
- Blocks
- Applications

Mark Wilson

HN Firebolt Demo

Update

 Type is comment or story is in the last 24 months is not block is not hidden

Connections



Filter

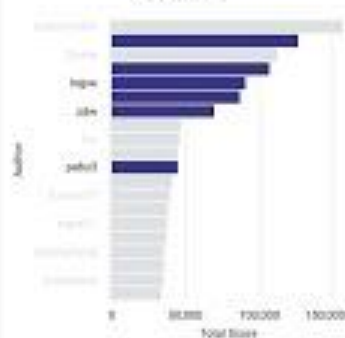
 Stories
23,381

 Comments
0

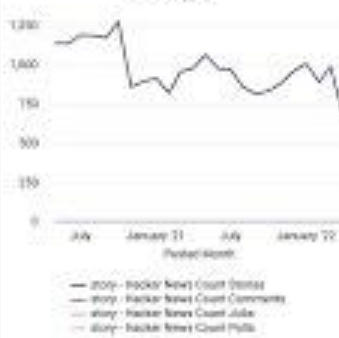
 Jobs
0

 Posts
0

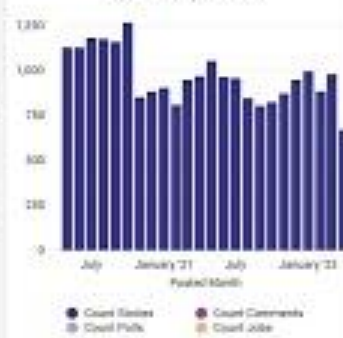
Top Authors



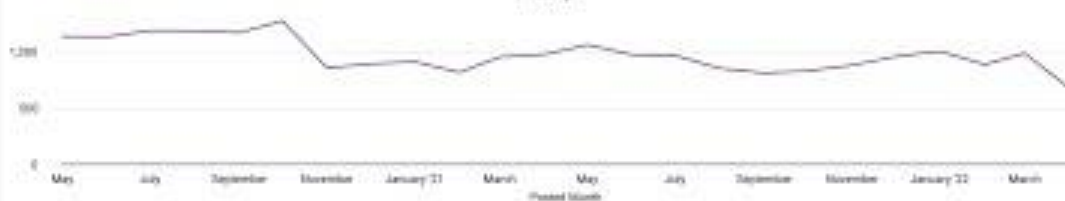
Post Types



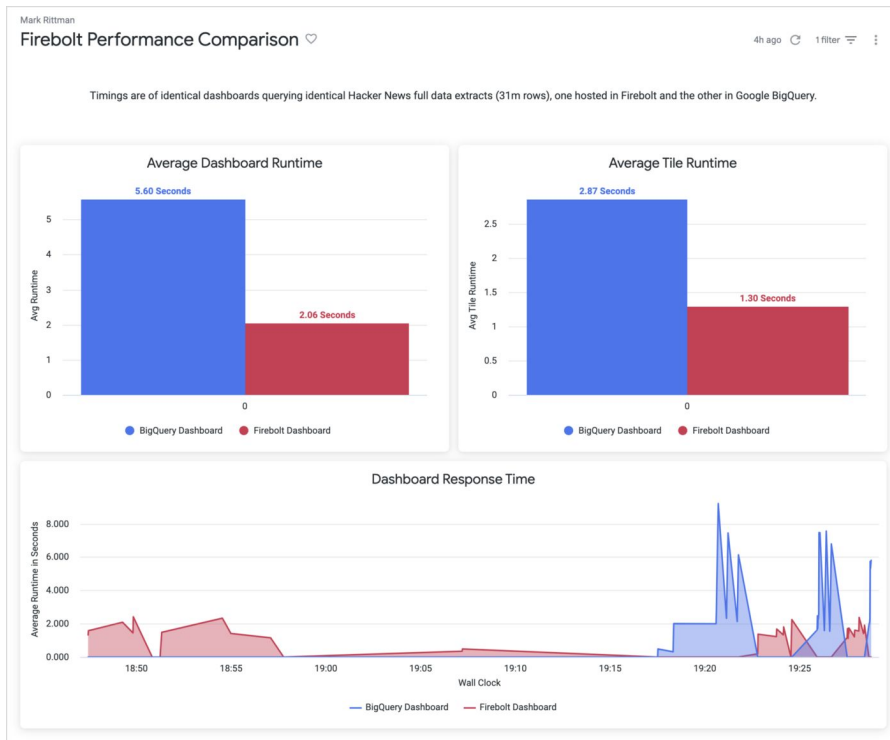
Content by Month



History



Performance Improvement vs. Baseline Dashboard



- 3x improvement in response time
- 2 secs typical refresh time
- 1.3 secs per dashboard tile
- ... and on a relatively small dataset
- 10x+ improvement more typical
- Sweet-spot is datasets 100GB+

Q&A

hello@hiflylabs.com

info@rittmananalytics.com

Interested? Find Out More

Rittman Analytics Home Services Solutions Case Studies Resources **Contact Us**

Analyzing the Hacker News Public Dataset using Firebolt Data Warehouse and Looker

Need Help With Your Project?
Rittman Analytics helps businesses become more data driven using

Something I've been meaning to write about for a while now is Firebolt, a relatively new startup who've been busy raising cheap We've Firebolt partici and qu

<https://rittmananalytics.com/blog/2022/4/25/analyzing-the-hacker-news-public-dataset-using-firebolt-data-warehouse-and-looker>

In full disclosure we're now a Firebolt Partner but we're also Snowflake, Google Cloud Platform and Oracle Partners, as well as regular uses of AWS Redshift, AWS Athena, Clickhouse, Postgres and many other database technologies; Firebolt is an interesting technology that aims to be both faster and cheaper than Snowflake and Google BigQuery, so how does it work and how does it stack-up against its more established rivals?

Firebolt is a new data warehousing technology that promises sub-second response times on very large datasets with highly-concurrent workloads — think of it as "Snowflake with Indexes", or more succinctly — "speed of Clickhouse meets Snowflake architecture".

Firebolt's underlying technology platform has its roots in Clickhouse, an open-source high-performance OLAP database system originally developed by Yandex, now Clickhouse, Inc to generate analytical reports in real-time from non-aggregated data that's constantly added-to in real-time.

Clickhouse's vectorized column store technology could scale-up and run distributed queries on multiple nodes but required the user to setup and manage the infrastructure it all ran on; Firebolt hard-forked the open-source Clickhouse code and re-engineered it to become server-less, decoupling storage and compute and added a metadata layer, query planning and service and orchestration layers as shown in the diagram below from Firebolt's technology whitepaper.

```
graph TD; subgraph Services; direction TB; S1[ ]; S2[ ]; S3[ ]; S4[ ]; end; API[API (REST)]; JDBC[JDBC]; DS[Data Science]; BI[BI Tools]; S1 -- SQL --> API; S2 -- SQL --> JDBC; S3 -- SQL --> DS; S4 -- SQL --> BI;
```

Drill to Detail Ep.89 'Firebolt and the History of Cloud Data Warehousing' with Special Guest Eldad Farkash

Mark Rittman is joined by Eldad Farkash, Co-Founder and CEO at Firebolt to talk about SiSense and Panorama, the history of cloud data warehouses and how Firebolt's technology delivers query results 4-6000x faster than Snowflake, Redshift and AWS Athena.

- [Snowflake's Co-Founder Marcin Żukowski Reflects On His Time At CWI](#)
- [Centrum Wiskunde & Informatica & MonetDB](#)
- [Eldad Farkash, inventor of In-Chip technology, takes top honors with IT Software \(Individual\) award](#)
- [Firebolt Touts Massive Speedup in Cloud Data Warehouse](#)

<https://rittmananalytics.com/drilltodetail/2021/4/27/drill-to-detail-ep89-firebolt-and-the-history-of-cloud-data-warehousing-with-special-guest-eldad-farkash>



Super-performant dashboards with dbt, Firebolt and Looker

Mark Rittman CEO, Rittman Analytics & Son N. Nguyen, Analytics Engineer, Hiflylabs
May 2022, Budapest, Hungary

Spare Slides

Firebolt F3 Storage Format

- Columnar storage format where data is sorted, compressed and sparsely indexed
- More granular storage = more processing in-memory = lower-latency queries



- Columnar micro-partitioned & compressed storage
- “Search optimization service” indexes fields for accelerated point lookup queries (+\$\$\$)
- Data is automatically divided into micro-partitions, with pruning at micro-partition level.



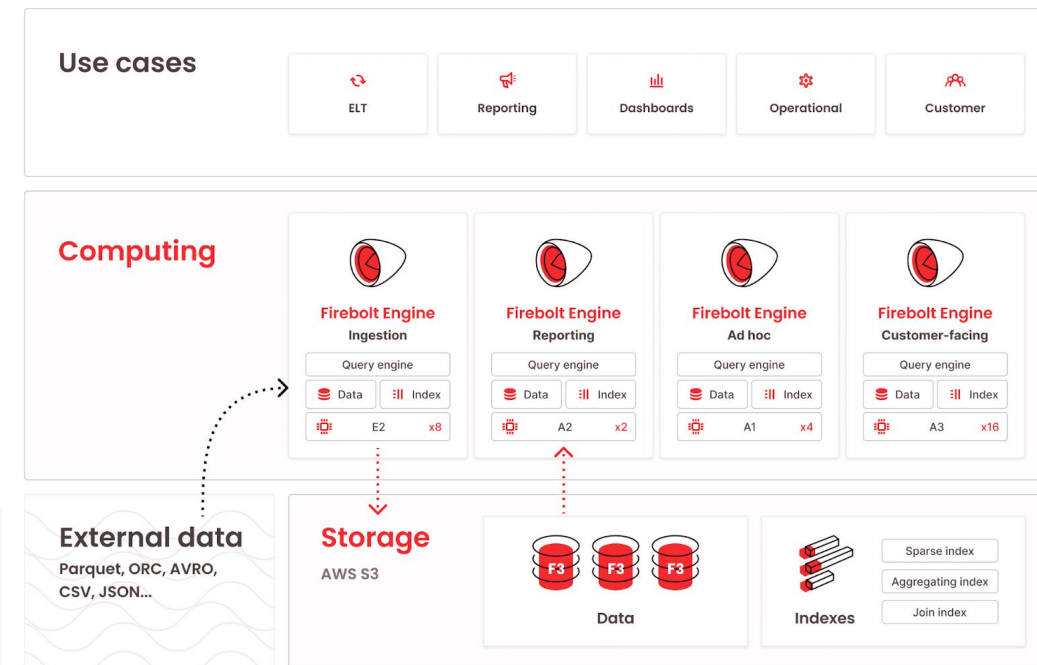
- Columnar & compressed storage (code named “Capacitor”)
- User-defined Table-level partitions. Pruning at partition level.



- Columnar, sorted & compressed & sparsely indexed storage (code named “F3”)
- Data is automatically sorted, compressed and indexed
- Pruning at indexed data-range level, which is dramatically smaller than partitions or micro-partitions.

An Architecture to Support all Analytic Workloads

- Implement one project and engine at a time with decoupled storage and compute
- Pair the best engine with each workload for the best price-performance combination
- Reuse data and engines across teams and projects as needed



Limitations and Considerations

- Generate index on pre-populated table (OOM)
- Auto-wake engines (API)
- How to update index keys?
- Not yet fully integrated into the MDS ecosystem
- Don't go crazy on indexing

```
{% macro agg_comments_fact() %}  
  
CREATE AND GENERATE AGGREGATING INDEX IF NOT  
EXISTS COMMENTS_FACT_AGG_INDEX ON COMMENTS_FACT  
(  
    year,  
    month,  
    author,  
    story_id,  
    sum(ranking) ,  
    avg(ranking) ,  
    count(*)  
);  
  
{% endmacro %}
```