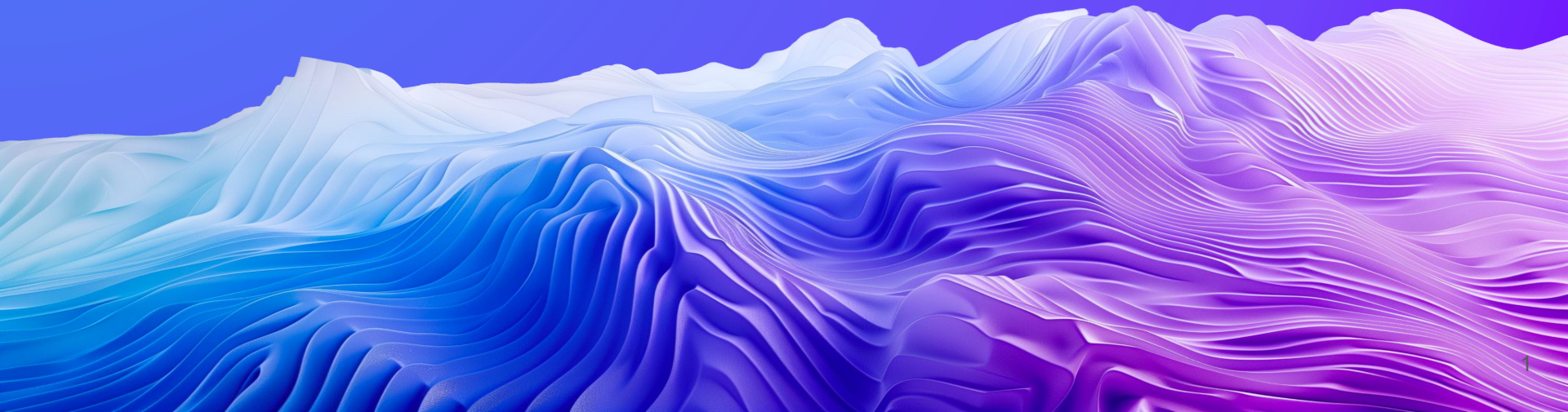
SCARF

Scarf & ClickHouse

December 12, 2024



Background

About me

- Founder & CEO of **Scarf**
- Based in Oakland, CA
- OSS maintainer, entrepreneur
- Board @ Haskell Foundation
- Former engineer @ Pandora

About Scarf

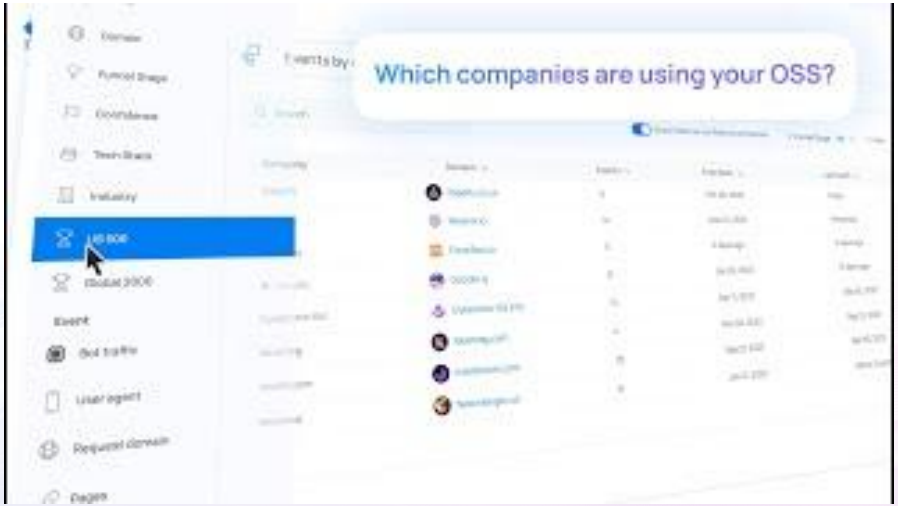
- 5-year-old startup
- 15 people
- Download metrics with artifact registry gateway, package telemetry, and web pixels.
- Works with both commercial and non-commercial open source projects
- Promote OSS sustainability and businesses with responsible but powerful anonymized usage analytics

SCARF © 2024 Scarf Systems, Inc. All rights reserved.

The product

Open source usage analytics for sales & marketing intelligence.

We process over 1B OSS downloads and events each month through our gateway.



SCARF © 2024 Scarf Systems, Inc. All rights reserved.

SCARF © 2024 Scarf Systems, Inc. All rights reserved.

Before Clickhouse

Stage 0: All OSS downloads went to a raw `package_events` table in PostgreSQL.

public.package_events	[BASE TABLE]
id [integer]	
user__id [integer]	
package_release__uuid [text]	
event_type [text]	
created_at [timestamp with time zone]	
raw_platform [text]	
raw_arch [text]	
mac [text]	
ip_address [text]	
dependent_package_name [text]	
dependent_package_version [text]	
dependent_package__uuid [text]	
dependent_package_release__uuid [text]	
package__uuid [text]	
ip_lookups__id [integer]	
package_name [text]	
package_release_version [text]	
scarf_version [text]	
dependent_package_name_hash [text]	
dependent_package_version_hash [text]	
entity__id [integer]	

Stage 1: Known set of aggregations are pre-computed and kept up to date

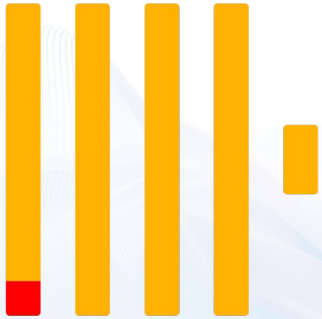
public.aggregates	[BASE TABLE]
id [bigint]	
package__uuid [text]	
aggregation_type [smallint]	
grouping_key_1 [text]	
count [bigint]	
date_ [date]	
unique_count [hll]	
grouping_key_2 [text]	
rollup [character(1)]	
version [smallint]	
artifact__uuid [uuid]	
flags [smallint]	
confidence [double precision]	
company__id [integer]	

SCARF © 2024 Scarf Systems, Inc. All rights reserved.

What problem were we solving for

- Query flexibility
  - Needed a solution that supported real-time ad-hoc queries without re-engineering the database.
  - Aggregations in psql were lossy and limit analytics across various columns
- Scalability challenges
  - The existing PostgreSQL system aggregated data into rollups that was rigid and preventing us from creating new data views quickly.
  - Serving raw events directly out of psql was far too slow for customers with >10M events / month
- Cost
  - Aggregations were difficult to maintain
  - Adding new aggregations meant expensive reprocessing jobs

SCARF © 2024 Scarf Systems, Inc. All rights reserved.



ClickHouse

SCARF © 2024 Scarf Systems, Inc. All rights reserved.

A mega `events` table in ClickHouse

events	[ReplicatedReplacingMergeTree]
endpoint_confidence	[Decimal(9, 3)]
package_version	[String]
entity_id	[UInt64]
entity_name	[String]
package_id	[LowCardinality(UUID)]
package_name	[String]
tracking_pixel_id	[LowCardinality(UUID)]
tracking_pixel_name	[String]
tracking_pixel_referer	[String]
tracking_pixel_referer_domain	[String]
tracking_pixel_referer_query	[Array(Tuple(String, String))]
docker_image	[String]
docker_reference	[String]
docker_backend_registry	[String]
python_version	[String]
python_file_name	[String]
python_redirect_url	[String]
scarfjs_version	[String]
scarfjs_architecture	[String]
scarfjs_platform	[String]
file_variables	[Map(String, String)]
user_imported_version	[String]
dnt	[Bool]
is_bot	[Nullable(Bool)]
platform	[String]
client_name	[String]
client_version	[String]
request_id	[String]
request_method	[LowCardinality(String)]
request_host	[String]
time	[DateTime('UTC')]
request_query	[String]
request_remote_address	[String]
response_status	[UInt16]
origin_id	[UUID]
endpoint_id	[String]
endpoint_provenance	[String]
endpoint_type	[LowCardinality(String)]
endpoint_domain	[String]
endpoint_name	[String]
endpoint_geo_country	[String]
endpoint_geo_state	[String]
endpoint_geo_city	[String]
endpoint_geo_postal_code	[String]
endpoint_geo_latitude	[Float64]
endpoint_geo_longitude	[Float64]
endpoint_company_id	[UInt64]
importance	[LowCardinality(String)]
points	[UInt64]
os	[String]
request_user_agent	[String]
revision	[UInt8]
event_type	[Enum8('unknown' = 1, 'manifest-fetch' = 2, 'pixel-fetch' = 3, 'install' = 4, 'file-download' = 5, 'python-package-download' = 6, 'user-imported' = 7)]
user_imported_event_type	[String]
inserted_at	[DateTime('UTC')]
appraisal_type	[LowCardinality(String)]
appraisal_created_by_user	[Int32]
event_provenance	[LowCardinality(String)]
appraisal_id	[Int64]
appraisal_updated_at	[DateTime('UTC')]
artifact_id	[LowCardinality(UUID)]
artifact_type	[LowCardinality(String)]

SCARF © 2024 Scarf Systems, Inc. All rights reserved.

What we got from ClickHouse

- **Most importantly, we can give our customers impressive real-time filtering capabilities!**
- De-duplication of events for free!
- Query speed
- Real-time and traditional OLAP all in one database
  - Real-time: Queries powering interactive dashboards for customers
  - Traditional: Custom reports and BI
- Large library of useful SQL functions
  - There is a simple function available to answer most analytical questions and powerful variants
  - Existing reporting and analytics tools can connect directly
- System tables are fantastic for debugging and answering questions about our database and queries

SCARF © 2024 Scarf Systems, Inc. All rights reserved.

Our workload

Query volume

- Peak activity
  - ~100 queries per second (QPS)
- Daily Reads
  - Average 1 Read QPS
  - ~100,000 Reads per day

Write volume

- Average
  - 0.7 Write QPS
  - ~60,000 Writes per day
- Event Appends
  - ~50M new events OR ~25 GiB/day
- Batch Inserts
  - Using **async_insert** to batch 500 - 1,500 events at a time
  - Batches are flushed every 2 seconds on average

SCARF © 2024 Scarf Systems, Inc. All rights reserved.

Our cluster

- Nodes: 2 (1 shard, 2 replicas)
- Node Type: m6i.4xlarge
- Node Storage: 10 TiB
- Node Memory: 64 GB
- Node CPU: 16

SCARF © 2024 Scarf Systems, Inc. All rights reserved.

Issues we had to solve for

- There was no mature Haskell client, so we wrote one. Will be open sourced soon.
- Large Aggregations with poor index fit failing due to OOM errors
- Query performance using `OR` → rewrite as `IN()`
- Deduplication in Clickhouse was magical but also confusing: When to use `FINAL` on a query vs when to trust the eventual `OPTIMIZE`?

SCARF © 2024 Scarf Systems, Inc. All rights reserved.

Open problems

SCARF © 2024 Scarf Systems, Inc. All rights reserved.

Operational overhead of index changes

Each table gets a single PRIMARY KEY and ORDER BY key.

In most cases, the only way to change these is a multi-step of creating a new table with desired index, copying over data from previous table, swapping the new table in for the previous table, and backfill any data that got missed in the transition.

SCARF © 2024 Scarf Systems, Inc. All rights reserved.

Pre-Aggregation vs Dynamic Filtering

CH is fast enough to serve interactive dashboards to small - medium customers backed by various aggregations and filterings **on columns outside of the index**.

This doesn't scale well to our large customers. CH has great support for pre-aggregation with Materialized Views, but we'd lose:

1) dynamic filtering (filters need to be applied before aggregation for this to work)
2) upstream deduplicating of ReplacingMergeTree

Refreshable Materialized Views may solve (2)

SCARF © 2024 Scarf Systems, Inc. All rights reserved.

ReplacingMergeTree ties deduplication to disk ordering

- With `ReplacingMergeTree`, the ORDER BY is what's used to deduplicate tables and determine the order data is laid out on disk. **The unique fields that we'd like to deduplicate our tables on are different from the ones we want in our queries.**
  - We settle for including both in our ORDER BY index, with the queryable columns first and the unique key last
- Some work was done on a `UniqueMergeTree` table engine with a `UNIQUE KEY` table property that is separate from the table's `ORDER BY`.

SCARF © 2024 Scarf Systems, Inc. All rights reserved.

SCARF

Thank you!

Avi Press
avi@scarf.sh

Scarf

Avi