

On the **Shoulders** of **Giants**: An Introduction to **Altinity Antalya Builds**

Alexander Zaitsev - Altinity CTO
Robert Hodges - Altinity CEO



Run Open Source ClickHouse® Better

Altinity.Cloud Enterprise Support

Altinity® is a Registered Trademark of Altinity, Inc.
ClickHouse® is a registered trademark of ClickHouse, Inc.;
Altinity is not affiliated with or associated with ClickHouse, Inc.



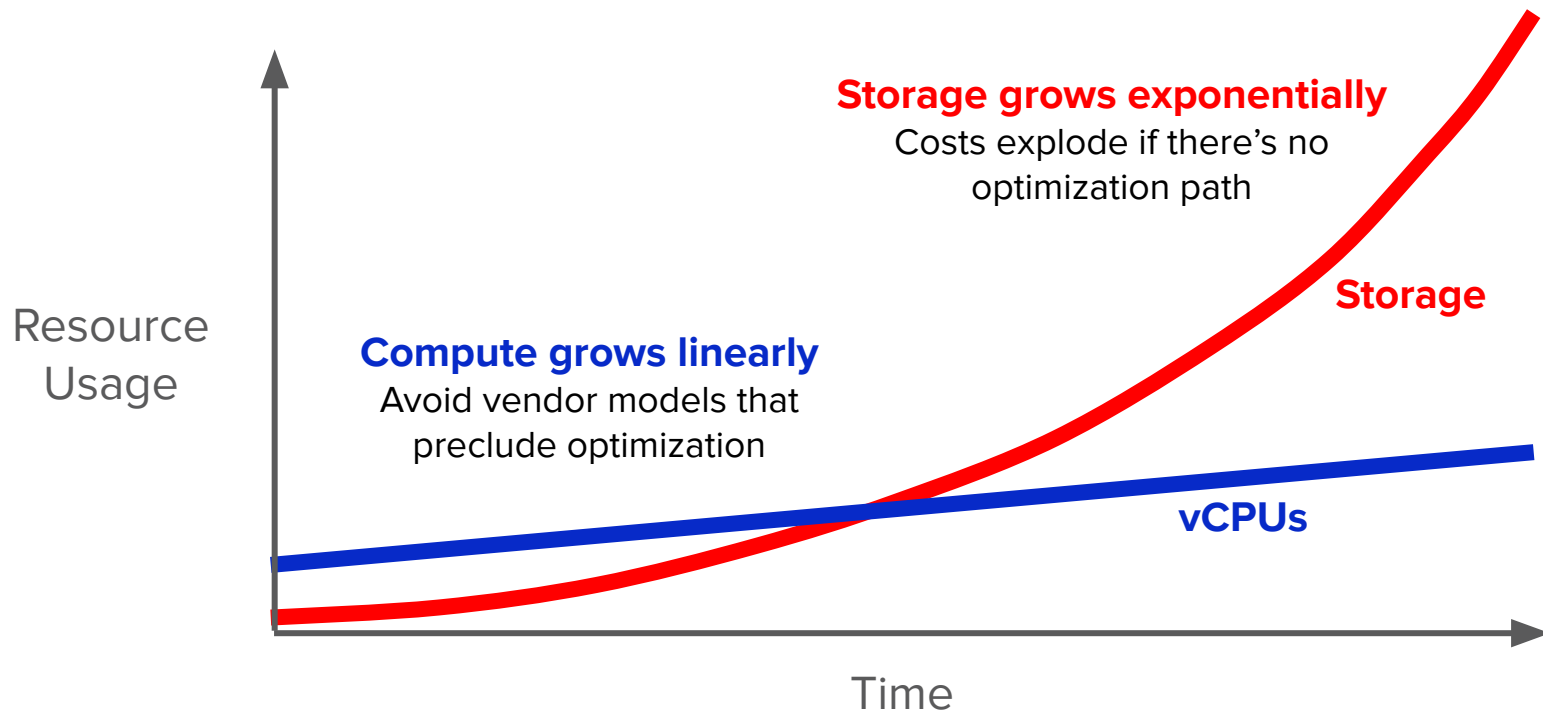
What are Antalya builds and why did we create them?

What is Antalya?

Antalya extends open source ClickHouse software: outfitted with features to lower cost, strengthen security, and increase performance — and no lock-in

- 100% compatible with upstream ClickHouse
- Operates in any environment
- Compute/storage separation
- Optimized for data lakes
- Rapid evolution to meet new use cases in AI and analytics
- 100% open source

How do analytic systems grow?



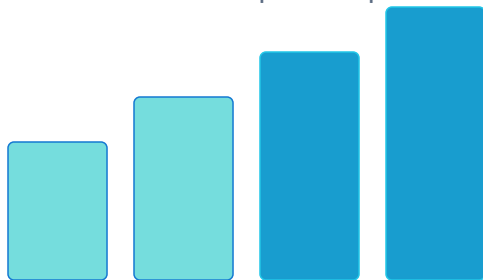
These growth patterns create practical ClickHouse issues

Data volume pushed storage cost up while bursty workloads left compute idle

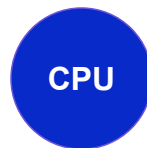


Data grows too fast

Replicated block storage becomes the expensive part of the system.



Offload the long tail to Iceberg



Compute is over-provisioned

Capacity for peaks sits idle during normal operation.



Apply stateless compute only when needed

We held a development offsite to investigate solutions rigorously

Dreams about
Parquet metadata
caching



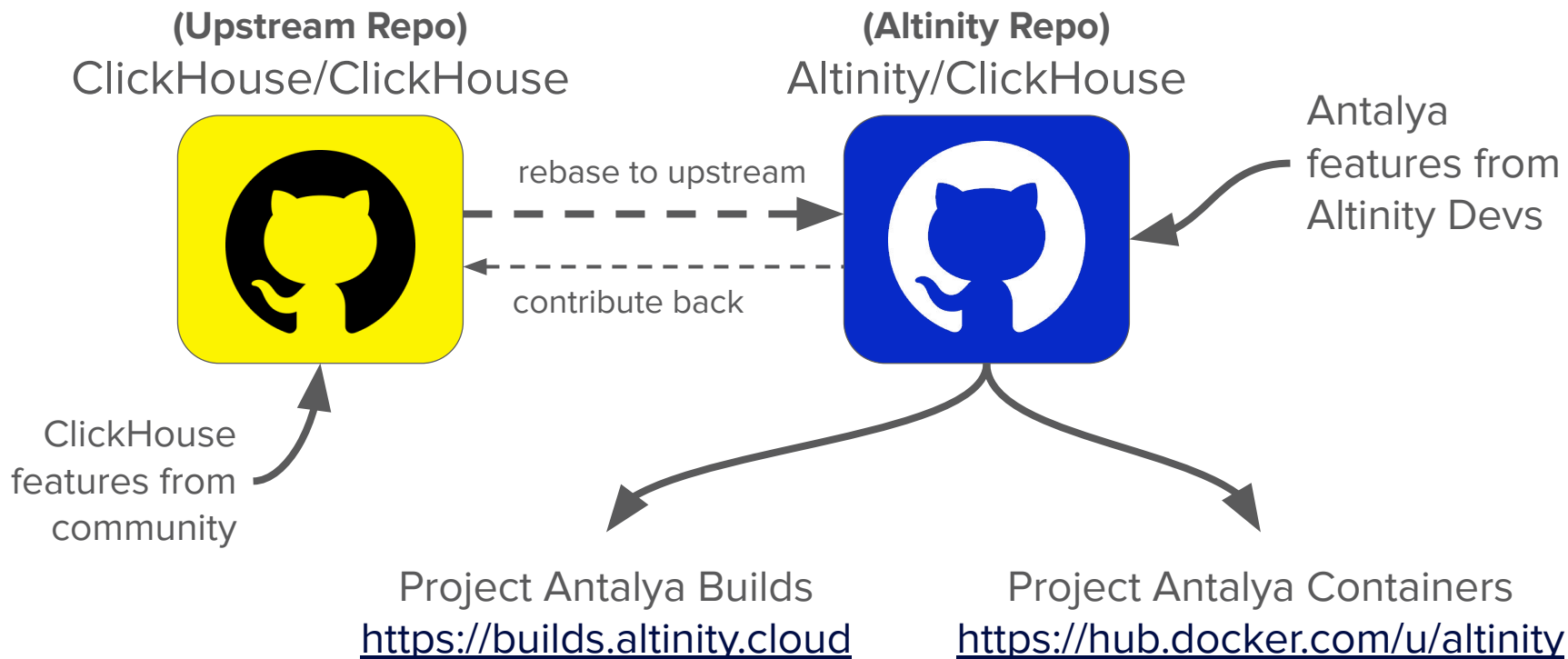
Years for
compute storage
separation

Secretly
designing shared
object storage

Silences
ClickHouse alerts
with a single glance

Arm wrestles
ClickHouse
builds for fun

Project Antalya was the result – a seamless extension of ClickHouse



Project Antalya in Numbers

Merged PRs to Altinity/ClickHouse

- Features 27
- Improvements 89
- Bug fixes 88

Titak: 204

Submitted to ClickHouse/ClickHouse

- Features 4
- Improvements 16
- Bug fixes 16

Total: 36



Introduction to Antalya Features

The shades of Antalya

16

One code line, four ways to create value



DATA LAKE QUERY ENGINE

Put compute near the data



INTEGRATED DATA LAKE

MergeTree now, Iceberg later



CAS SEPARATION

Extend MergeTree onto shared storage

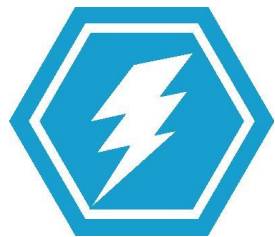


OAuth SECURITY

Identity across users, tools, and data

Compute and storage separate cleanly

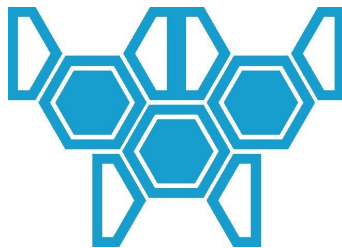
Stateless swarms run close to shared Iceberg data



Query initiator

Plans the query and merges results

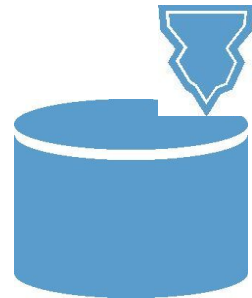
Independent scale



Stateless swarm

Scales out for each workload

Elastic compute

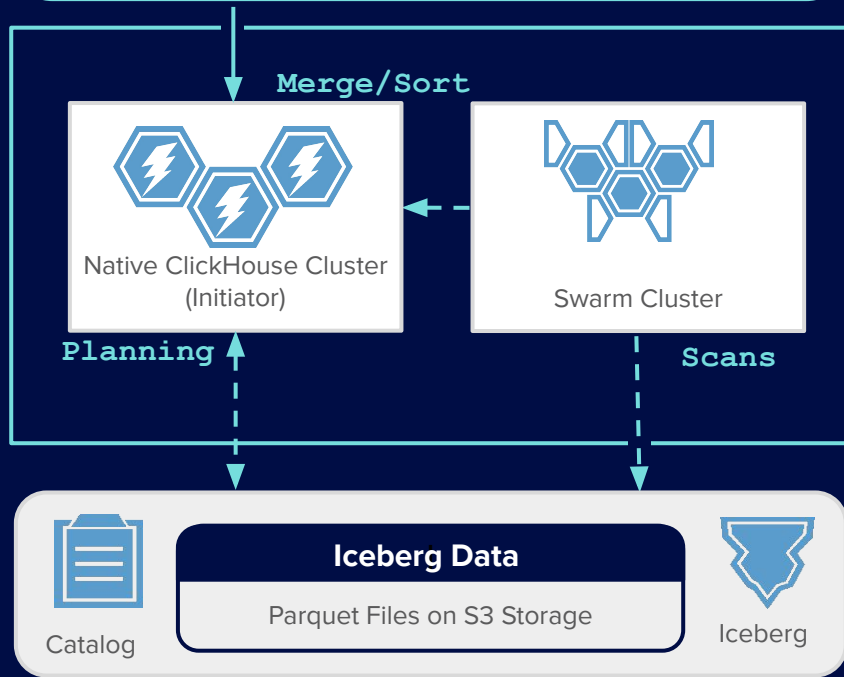


Iceberg storage

Parquet files in object storage

One shared copy

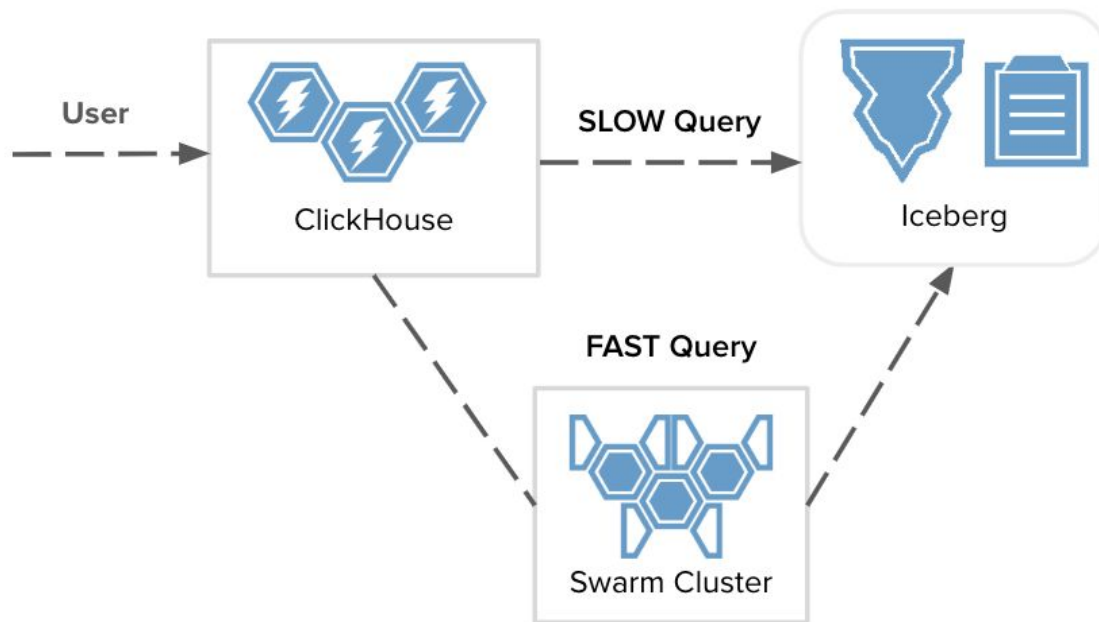
```
SELECT data, sum(output_count)
FROM iceberg.`btc.transactions`
WHERE date >= '2024-01-01'
GROUP BY date ORDER BY date
SETTINGS
object_storage_cluster = 'swarm'
```



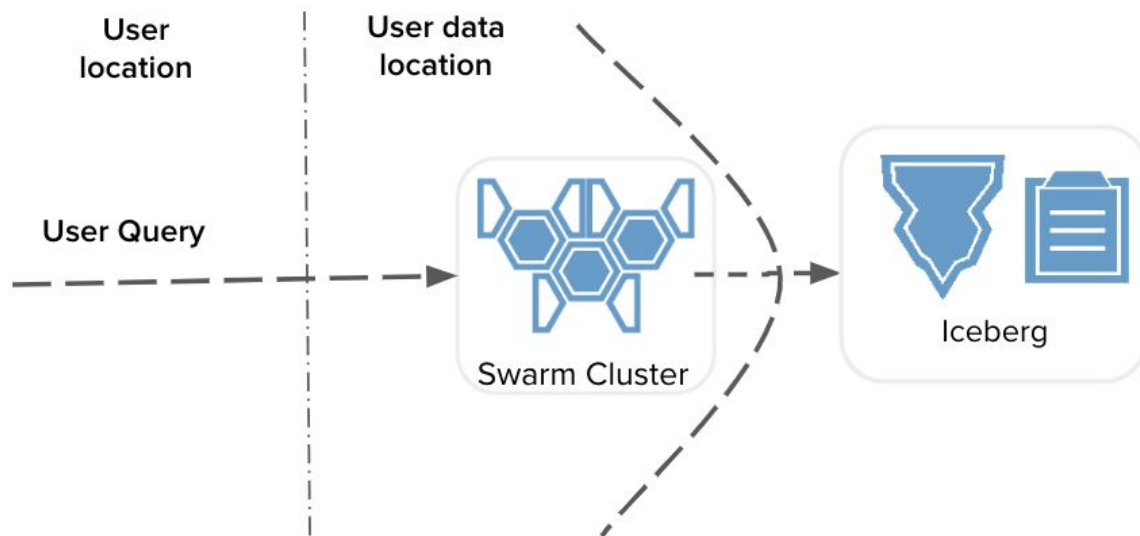
Project Antalya query model

- **Initiator** plans the query
- **Swarm Cluster nodes** scan shared files
- **Initiator** merges and sorts responses

Scaling Data Lake Queries

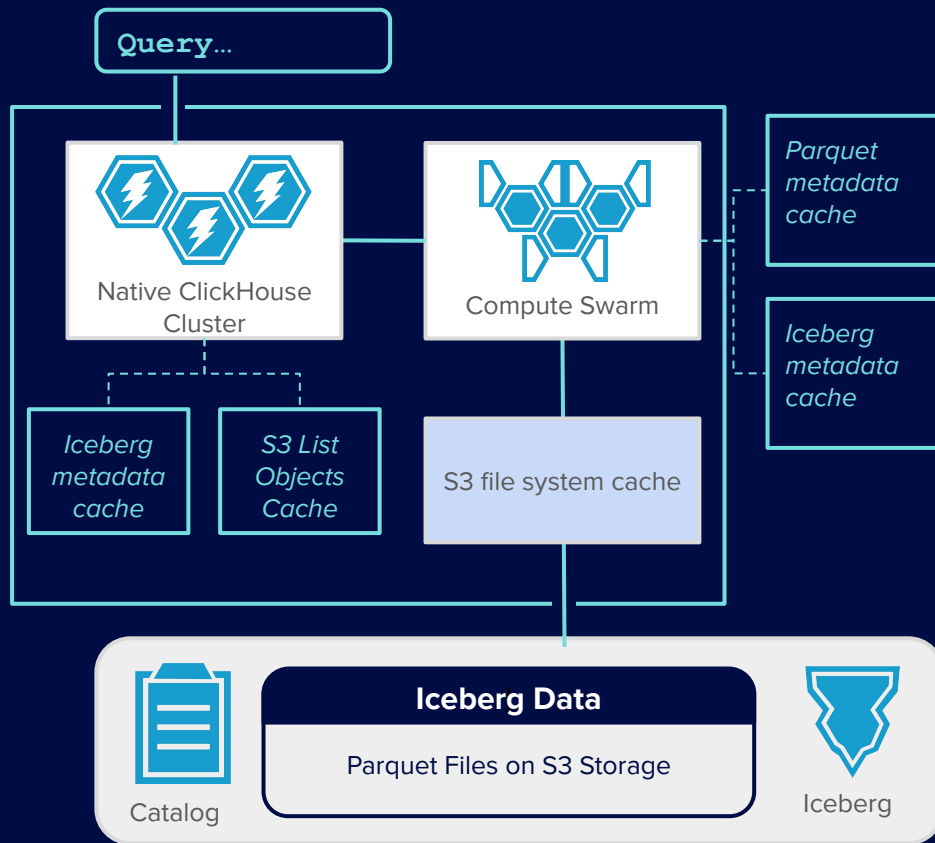


Deploying Compute Close to the Data



Swarm Query Settings

Session Setting Name	Description	Value
<code>enable_filesystem_cache</code>	Use filesystem cache for S3 blocks	0 or 1
<code>use_parquet_metadata_cache</code> *	Cache Parquet file metadata	0 or 1
<code>object_storage_cluster</code>	Swarm cluster name	String
<code>object_storage_max_nodes</code>	Number of swarm nodes to use (defaults to all nodes)	Integer
<code>use_hive_partitioning</code>	Files follow Hive partitioning	0 or 1
<code>use_iceberg_metadata_files_cache</code>	Cache parsed Iceberg metadata files in memory	0 or 1
<code>use_iceberg_partition_pruning</code>	Prune files based on Iceberg data	0 or 1

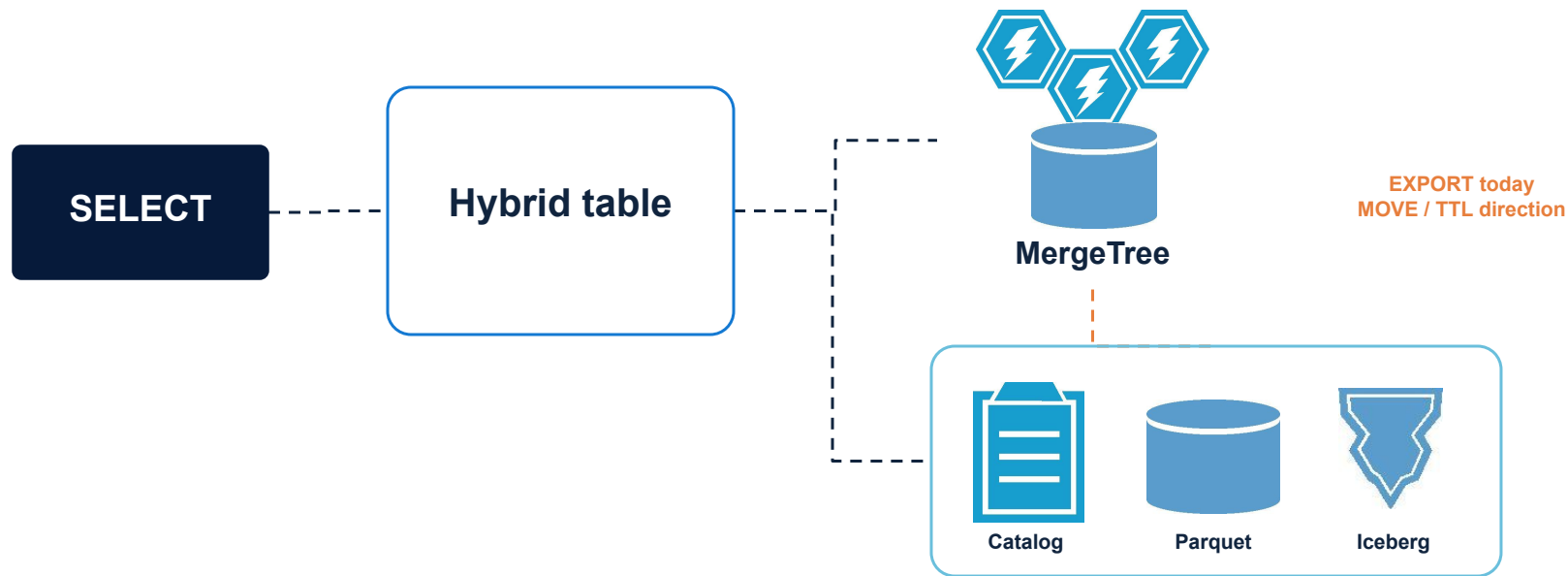


Caching is Important

- Iceberg metadata cache
- Parquet metadata cache
- S3 List Objects Cache
- S3 file system cache

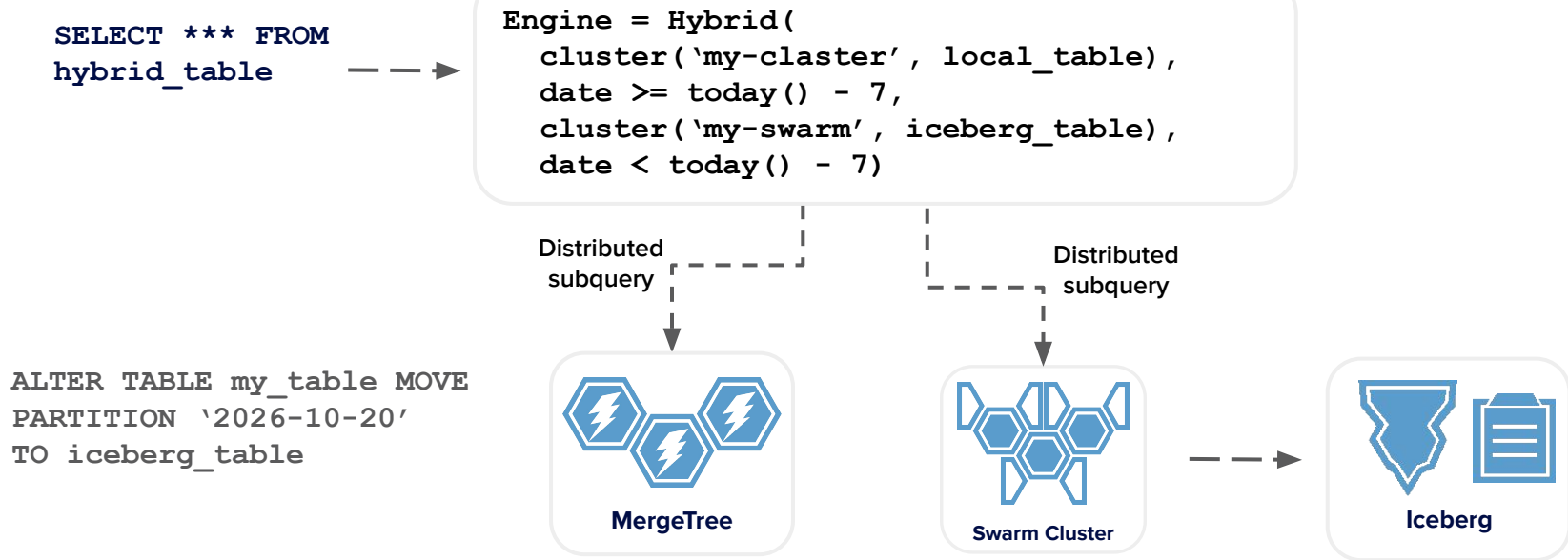
Iceberg becomes a first-class ClickHouse tier

Hot MergeTree data and shared Iceberg data appear behind one query surface



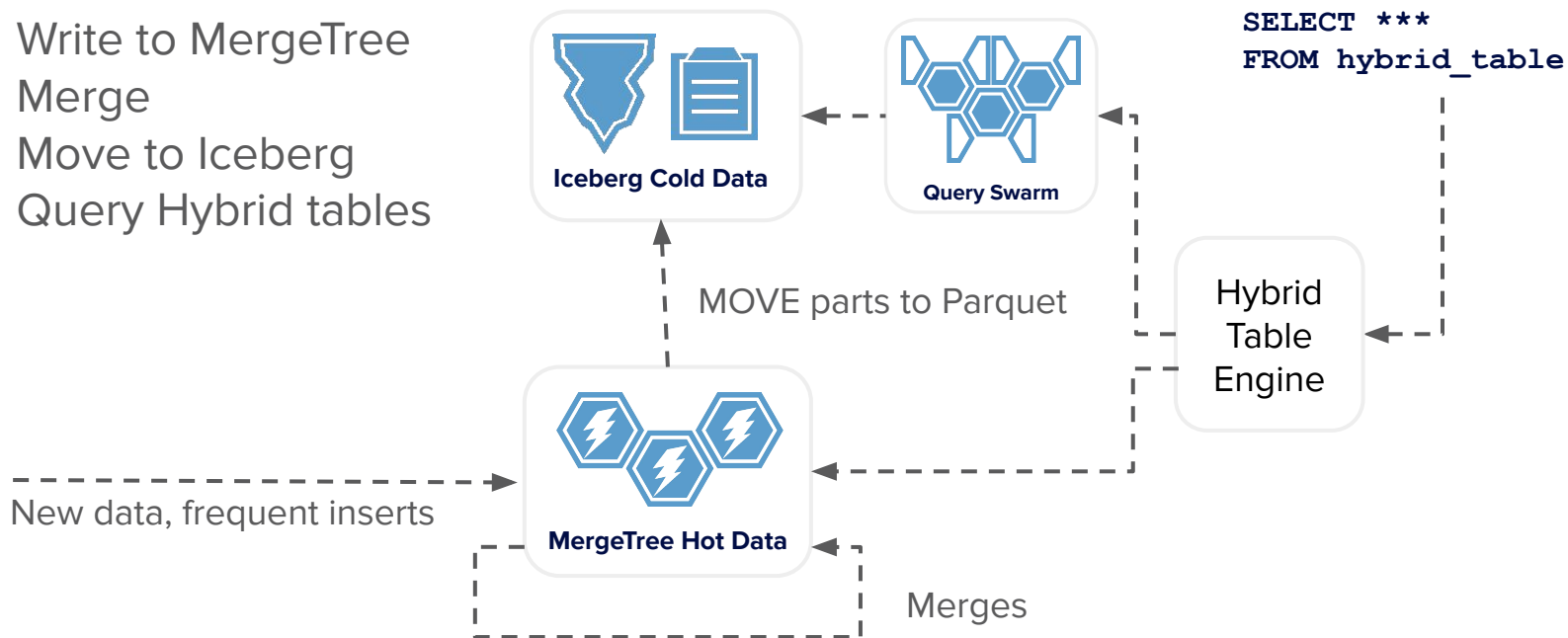
Iceberg opens the data to AI, data science, and batch engines

Hybrid Tables – MergeTree “extension” into Iceberg



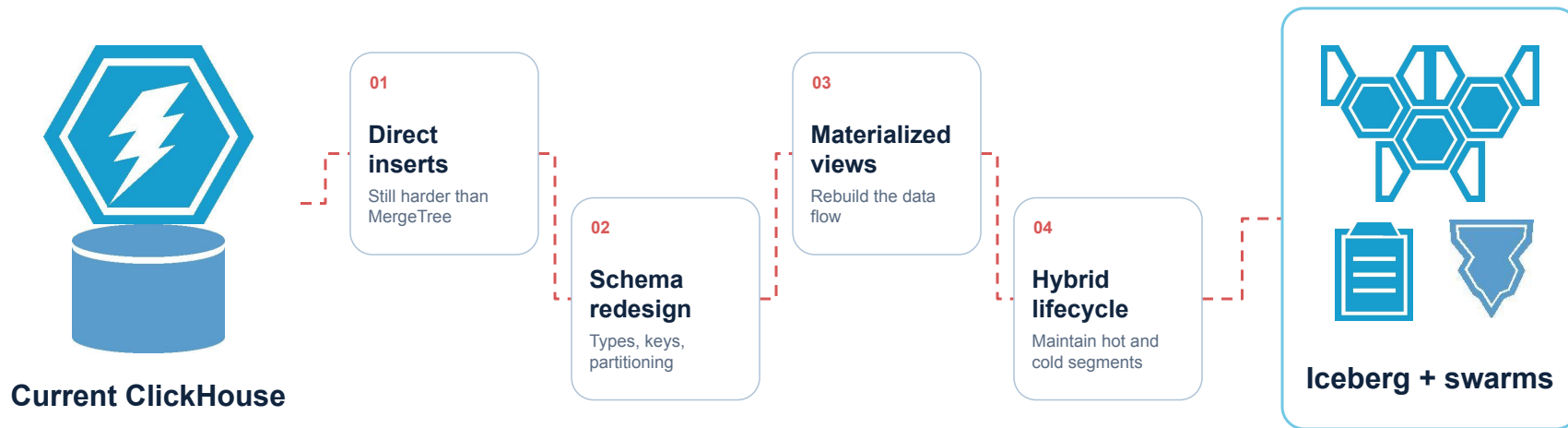
Hybrid Tables for Hybrid Architecture

1. Write to MergeTree
2. Merge
3. Move to Iceberg
4. Query Hybrid tables



The Iceberg path carries a migration tax

The destination is valuable, but the route changes the data model and operating model



Migration remains a project

CAS makes shared storage much easier

Content Addressed Storage preserves MergeTree while moving durable parts to an object-store pool

INSERT

Existing MergeTree write path

NO HYBRID

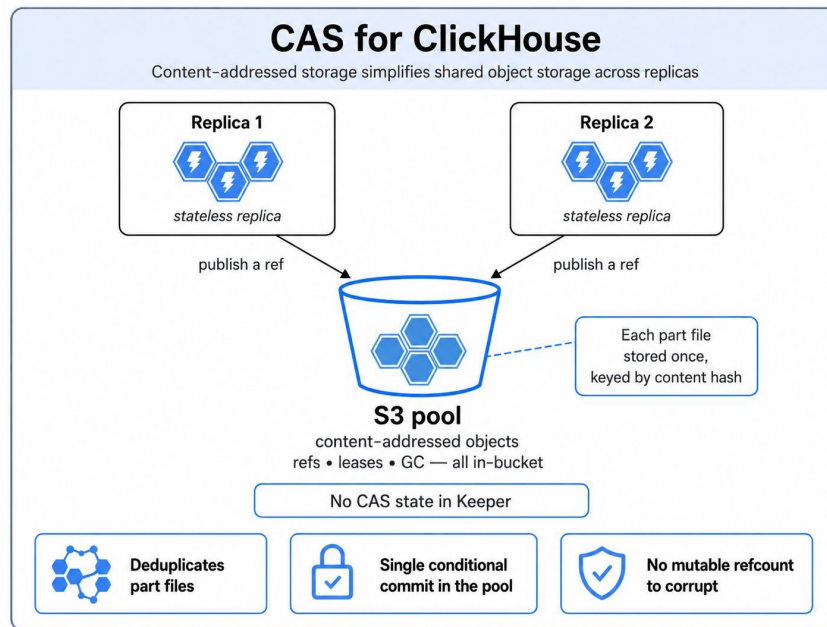
No Hybrid table required

MOVE / TTL

Standard storage policies

ADOPT

Upgrade, configure, move data



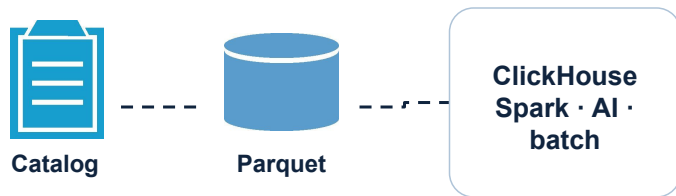
EXPERIMENTAL IN ANTALYA 26.6

Iceberg and CAS answer different questions

Iceberg opens a table to many engines. CAS extends native MergeTree storage.

ICEBERG

Open table format



READERS

Many engines read the same table

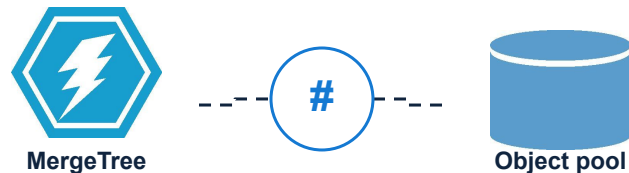
ADOPTION

Export data and redesign schemas, types, and workflows

Interoperability

CAS

MergeTree storage backend



READERS

ClickHouse reads native MergeTree tables

ADOPTION

Add a CAS disk, then use standard MOVE and TTL policies

Seamless ClickHouse tiering

CAS is an Extension of ClickHouse Storage Model

```
<clickhouse>
  <storage_configuration>
    <disks>
      <cas>
        <type>object_storage</type>
        <object_storage_type>s3</object_storage_type>
        <metadata_type>cas</metadata_type>
        <cas_server_root_id>{replica}</cas_server_root_id>
        <endpoint>...</endpoint>
      </cas>
    ...
  <policies>
    <cas>
      <volumes>
        <main>
          <disk>cas_cache</disk>
        </main>
      ...
    ...
  ...
```

1. Define disk
2. Define storage policy
3. Use with regular MergeTree tables!

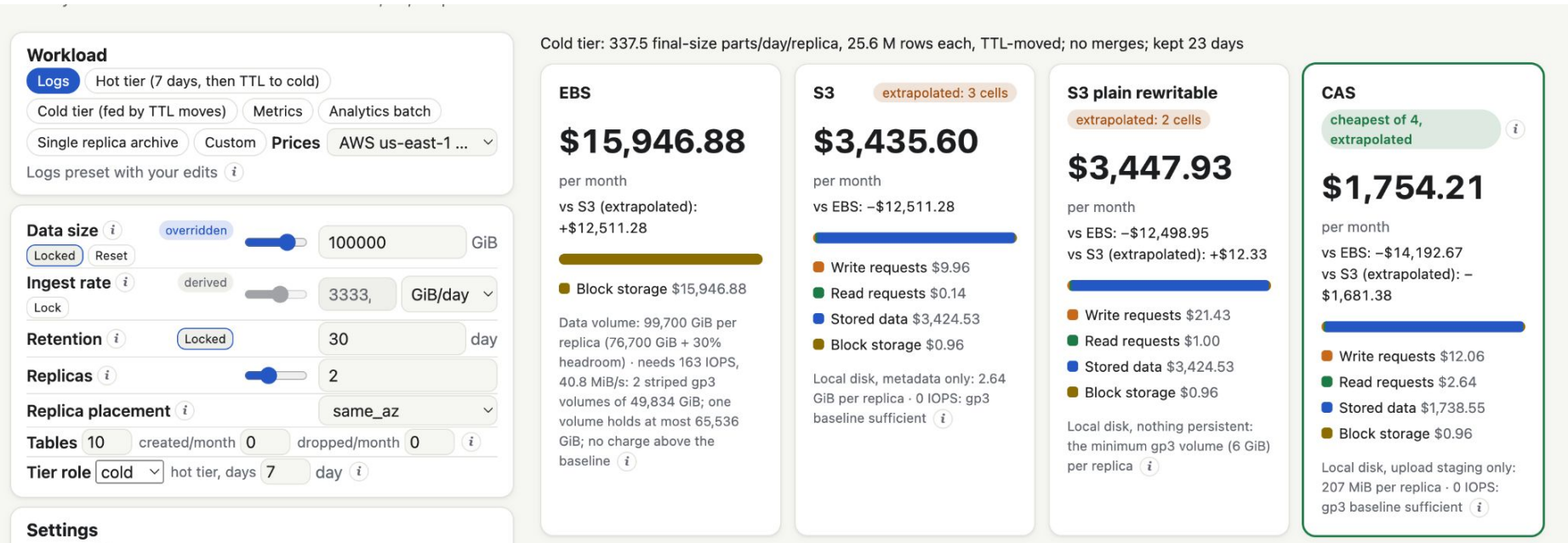
```
CREATE TABLE my_table
...
ENGINE = ReplicatedMergeTree
PARTITION BY toYYYYMM(ts)
ORDER BY (ts, id)
SETTINGS storage_policy = 'cas';
```

CAS is an Extension of ClickHouse Storage Model

S3 disk	CAS disk
<pre><clickhouse> <storage_configuration> <disks> <s3> <type>object_storage</type> <object_storage_type>s3</object_storage_type> </s3> </disks> <endpoint>https://<bucket>/clickhouse/{cluster}/{shard}/{replica}</endpoint> </storage_configuration> </clickhouse></pre>	<pre><clickhouse> <storage_configuration> <disks> <cas> <type>object_storage</type> <object_storage_type>s3</object_storage_type> <metadata_type>cas</metadata_type> <cas_server_root_id>{server_uuid}</cas_server_root_id> <endpoint>https://<bucket>cas/{cluster}</endpoint> </cas> </disks> <cas_cache> <type>cache</type> <disk>cas</disk> <path>cas_cache</path> <max_size>50Gi</max_size> </cas_cache> </storage_configuration> </clickhouse></pre>

Almost identical!

Storage Cost Calculator Demo





CAS

OR



SharedMergeTree

CAS shares parts. SharedMergeTree shares table state

The largest difference is how compute joins and leaves the system



CAS



SharedMergeTree

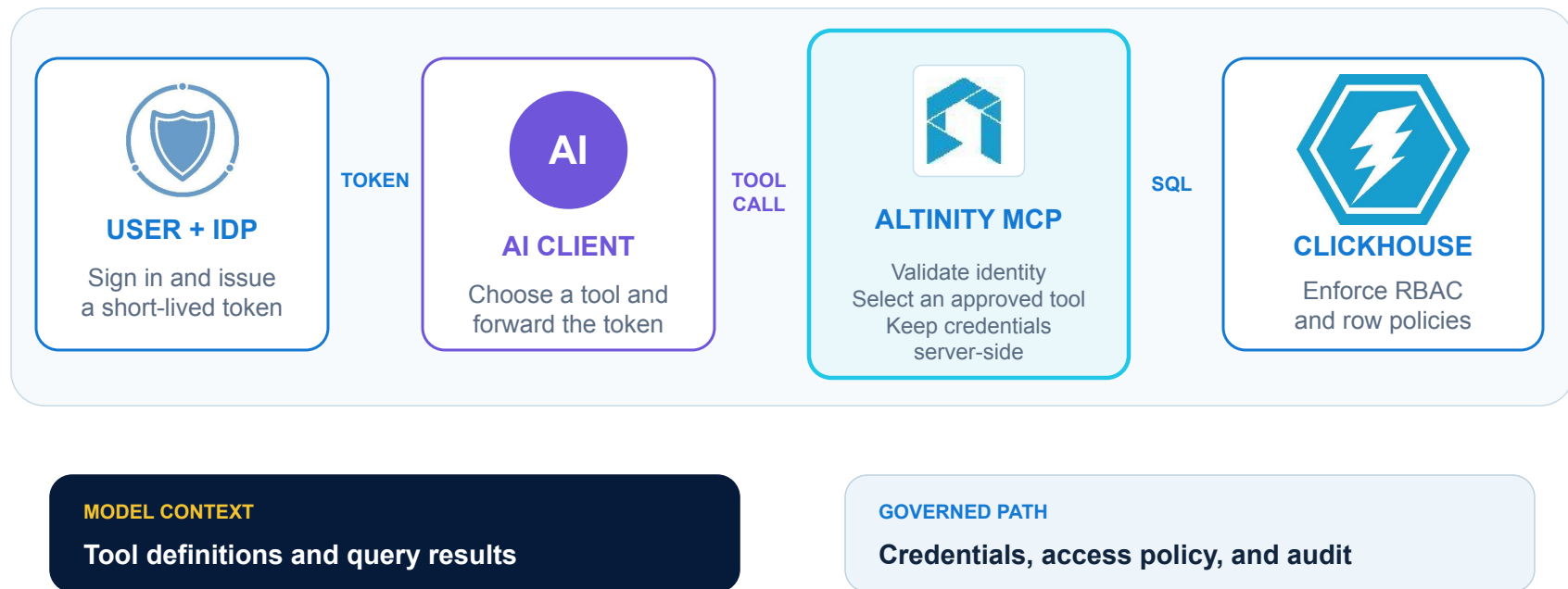
What is shared	Content-addressed MergeTree parts. Each replica keeps its own table state.	Data and table metadata live in shared storage.
Deployment	Anywhere: self-hosted, Kubernetes, or cloud	ClickHouse Cloud
Elastic compute model	Scale conventional replicas. New replicas join the topology and rebuild local state.	Compute nodes attach to shared table state. Add or remove nodes without copying the table.

CAS reduces storage cost. SharedMergeTree changes the compute lifecycle.

Use case	CAS	SharedMergeTree
Single-tier inserts	Slower due to S3 metadata and concurrency locks	Designed for ingestion into shared object storage. Shared metadata supports concurrent writes.
Multi-tier inserts	Hot inserts and early merges stay on local disks. Settled data moves to CAS—the preferred deployment.	N/A
Query performance	Hot data can be read locally; cold reads use caches and object storage.	Reads use caches and shared object storage. Distributed cache reduces cold reads
Cost	Shared storage avoids duplicate copies across replicas. Tiering reduces intermediate object-store writes. Can be run anywhere, including on-prem	Shared storage avoids duplicate copies across replicas. S3 API calls may be expensive on inserts and merges. Available only in ClickHouse Cloud

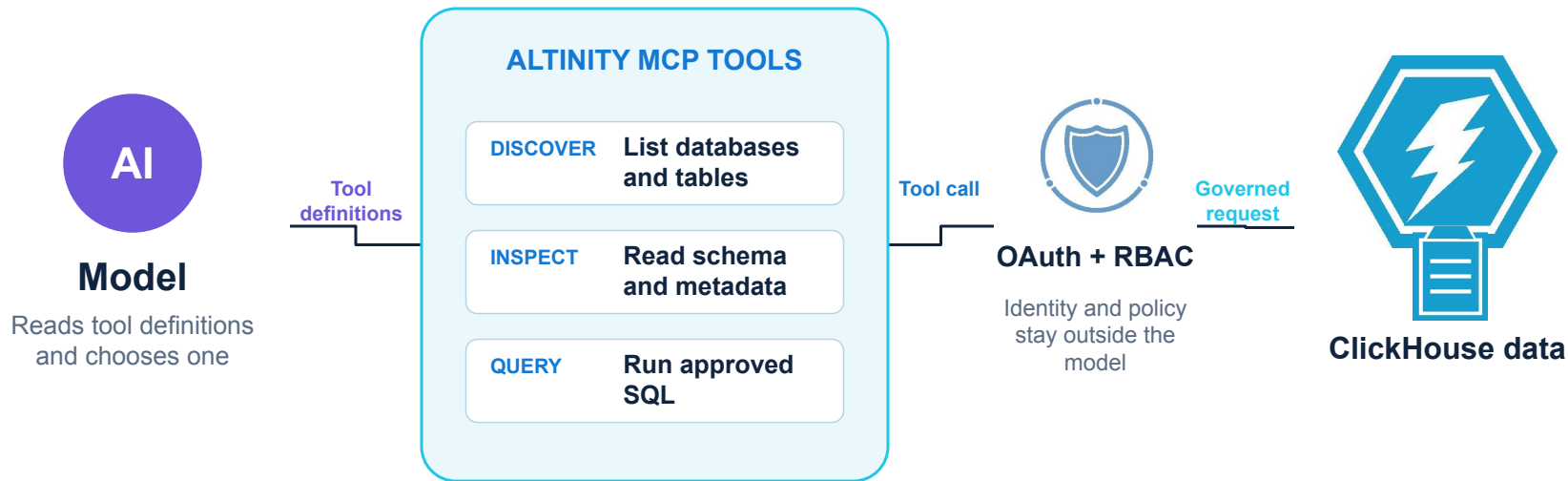
OAuth carries user identity to ClickHouse

The model sees tools and results. Credentials, policy, and audit stay in the governed path.



MCP gives models a controlled way to use databases

The model chooses a tool. Altinity executes it against governed ClickHouse data.



The model receives a structured result, not a database credential

2026: Antalya is moving from incubation to product

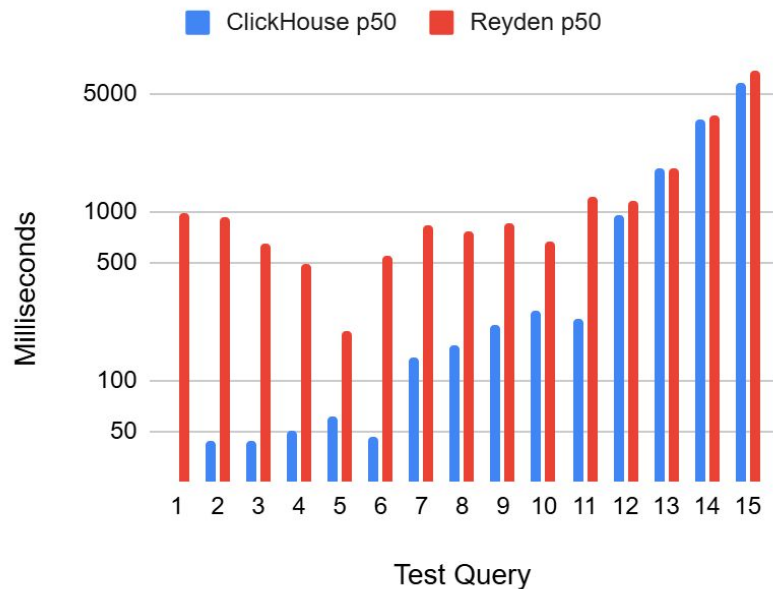
17

What has matured, what is shipping, and what still belongs to the roadmap

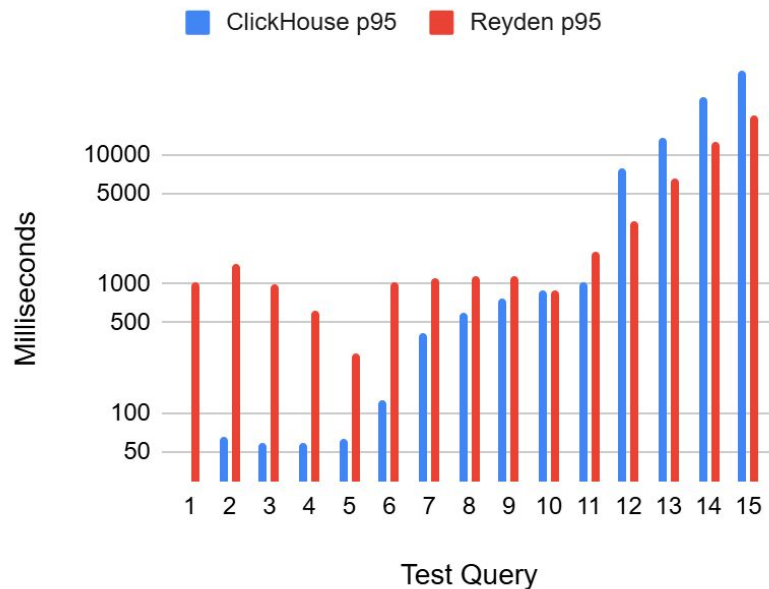
MATURED	NEW IN 26.6	NEXT
1 Query swarms 2 Hybrid tables 3 OAuth / OIDC GA in 26.3 4 Partition export	1 Direct standalone swarms 2 CAS disk backend 3 Iceberg v3 deletion vectors	1 Unity liquid clustering 2 OAuth to catalogs 3 CAS GA target: end of 2026 4 Insert, compaction, serverless swarms

Test Results - Altinity Antalya versus Databricks Reyden

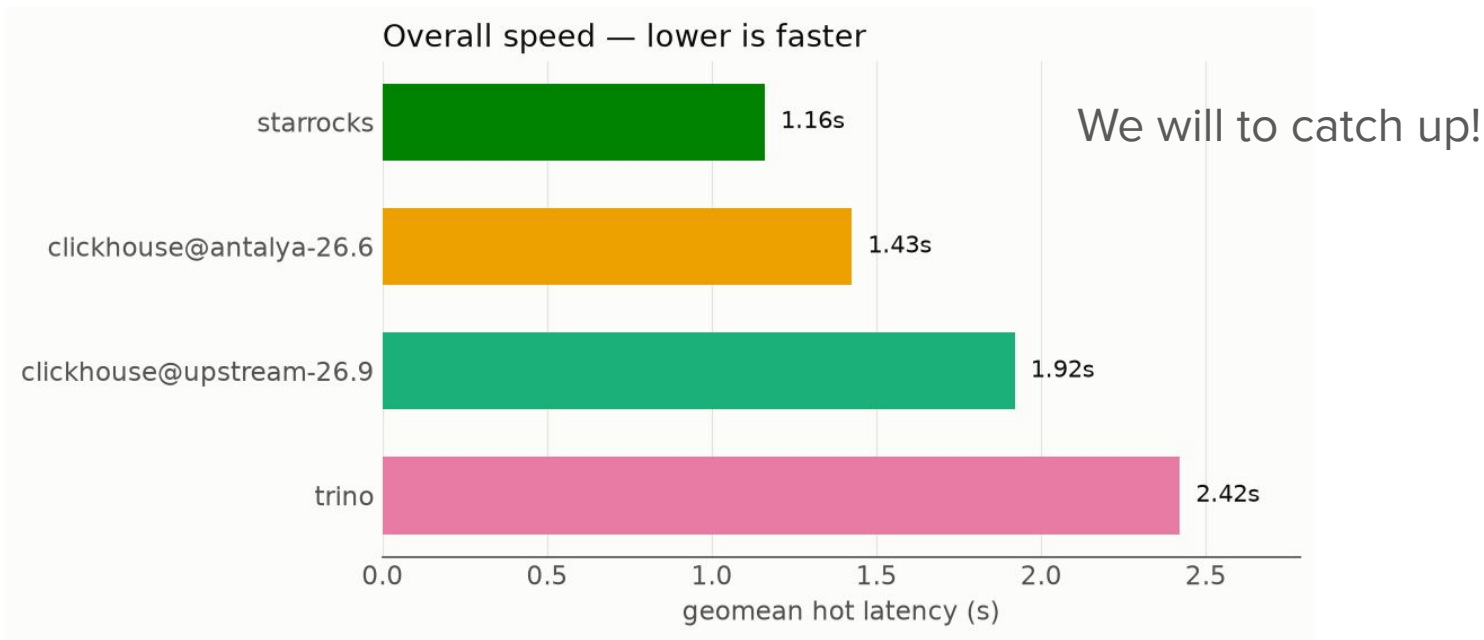
P50 Response (Smaller Is Better)



P95 Response (Smaller Is Better)



IcebergBench



<https://github.com/Altinity/iceberg-bench/blob/main/BENCHMARK.md>



Getting started with Antalya

How Antalya builds work

Quick how-to

How to find more information

All about Antalya Builds

Project Antalya Builds

<https://builds.altinity.cloud>

Project Antalya Containers

<https://hub.docker.com/u/altinity>

- Antalya builds are based on upstream LTS releases (25.8, 26.3, 26.8...)
- We rebase our extensions on each new LTS branch
 - Ensures compatibility with upstream
- Build and test follow the same CI/CD protocol as Altinity Stable Builds
 - Run all ClickHouse tests + Altinity tests
- New Antalya LTS releases after 26.8 will have 3 year maintenance tails

Running and fixing Antalya

How to run Antalya builds:

- In Altinity.Cloud - <https://altinity.com/managed-clickhouse/bring-your-own-cloud/>
- In Kubernetes - Use the [Altinity Kubernetes operator](#)
- On Linux - Binaries/packages from <https://builds.altinity.cloud/#altinityantalya>

How to log issues:

- <https://github.com/Altinity/ClickHouse/issues>

How to submit pull requests:

- git clone <https://github.com/Altinity/ClickHouse>
- Commit to default branch (currently antalya-26.6)
- Submit PR from your branch

Altinity.Cloud operates the full Antalya stack

The management plane deploys, scales, secures, and observes every shade of Antalya



How to find out more about Antalya builds

Altinity Documentation - <https://docs.altinity.com/altinityantalya/>

Altinity Playground - <https://github.com/Altinity/altinity-playground>

Antalya Code - <https://github.com/Altinity/ClickHouse> (Use default branch)

Thank you! Questions?

Antalya Demos



How to contact us:

<https://altinity.com/contact/>

<https://altinity.com/slack>

We love open source
and we are hiring!!!



**Altinity ClickHouse
Repo**