

ClickHouse



meetup

Google Campus Seoul
@ B2 Autoway Tower
Seoul, South Korea

November 18 at 7:00 PM KST

ClickHouse Slack

커뮤니티에
참여하세요



사진 및 영상 동의 Photo & Video Consent

사진과 영상을 촬영하여 Dev Korea와 ClickHouse
간에 공유할 예정입니다.
촬영에 포함되기를 원하지 않으신다면, 저희에게
알려주세요.

We'll be taking photos and videos and sharing
them between Dev Korea and ClickHouse. If you
prefer not to appear in them, please let us know.

Cuckoo Real-Time AI Translator



Meet the ClickHouse crew



Sunny Yoon / 윤선정



Account Executive, Korea
영업 담당자



Ken Lee / 이기훈



Solution Architect, Korea
이사



Rana Banerji



Sales Leader, ASEAN



Derek Chia



Principal TAM Architect



Cheryl Tuquib



Head of Field Marketing, APJ



테크 토크 Tech Talks



ClickHouse 소개, 그리고 내가 가장 좋아하는 기능들 **An Introduction to ClickHouse and My Favourite Features**

Ken Lee / 이기훈, Solution Architect / 이사
및 Derek Chia, Principal TAM Architect @ ClickHouse



ClickHouse를 활용해 ClickHouse를 최적화한 여정 **A Journey of Using Clickhouse to Optimize ClickHouse**

Antoine Grondin, Founder @ humanlog.io

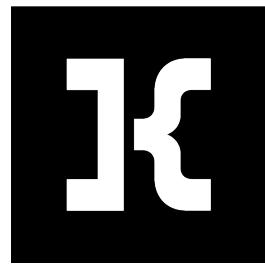


호랑이 리더보드와 한국어 LLM 평가에 대한 성찰 **Reflections on the Horangi leaderboard and Korean LLM Evaluation**

Hyunwoo Oh / 오현우, AI Solution Engineer /
AI 솔루션 엔지니어 @ Weights & Biases



Thank you to our partners!



**dev
korea**

Google
for
Startups





ClickHouse 소개

An Introduction to ClickHouse

Ken Lee / 이기훈 (ken.lee@clickhouse.com)

Solution Architect / 이사

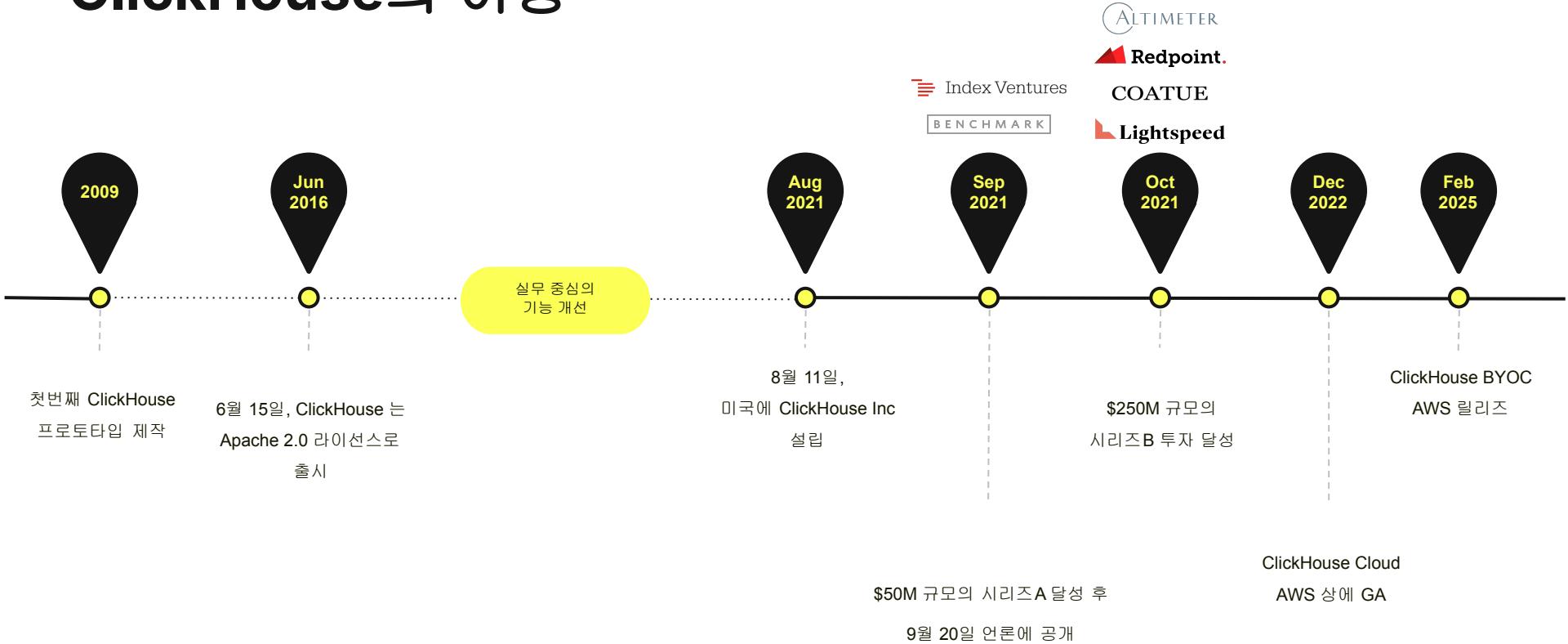
ClickHouse Inc.

2025년 11월



**ClickHouse에 대해 얼마나
알고 계신가요 ?**
**How familiar are you with
ClickHouse?**

ClickHouse의 여정



창업멤버



Aaron Katz

CEO



Alexey Milovidov

CTO

ClickHouse



Yury Izrailevsky

Co-founder, President of
Product & Engineering



18개의 국가에서 지속적으로 확장 중 (300+ 직원)



40%

북미

45%

유럽

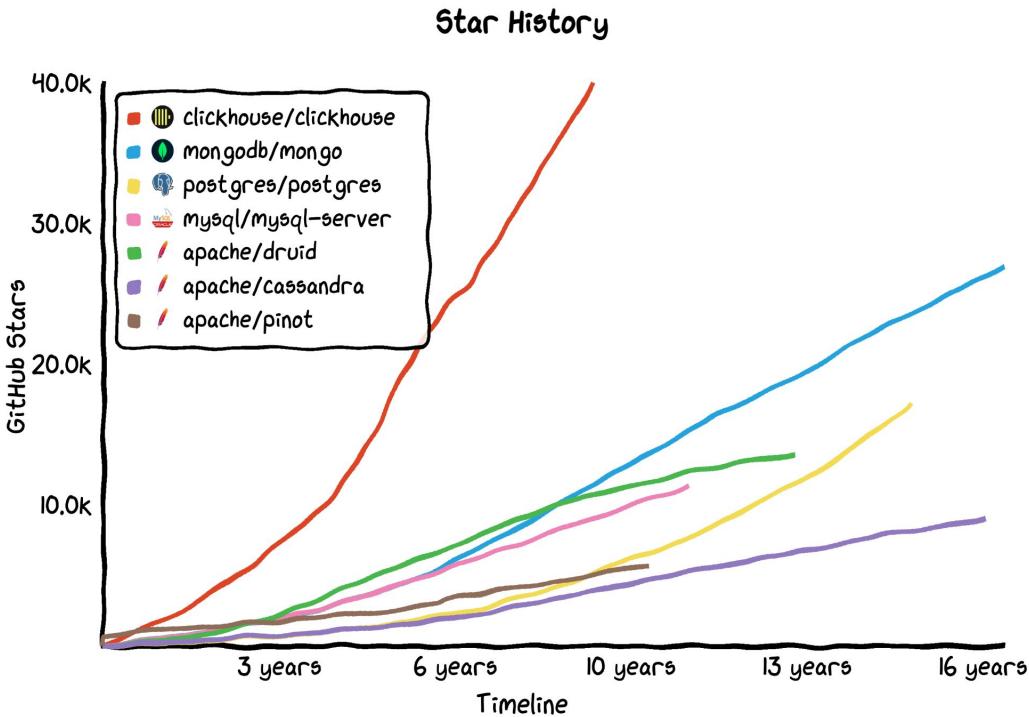
15%

아시아태평양

ClickHouse 란?



ClickHouse 의 성장



ClickHouse Open Source

- 40k stars
- 7k forks
- 1.6k contributors
- 490 active contributors
- 10k Slack members

ClickHouse Cloud

- 2022년 출시 후 수천 개의 트라이얼과 상용 고객
- 매일 55억 건 이상의 쿼리가 수행되며, 상위 100PB의 데이터에서 3,500조 건의 데이터가 탐색됨
- 2024년 12월, AWS와 5년간의 전략적 파트너십 사인 (SCA: strategic collaboration agreement)

유즈케이스

REAL-TIME ANALYTICS DATASTORE

ClickHouse는 수조 건의 인터넷 요청에 대한 로그를 효율적이고 안정적으로 분석하여 악성 트래픽을 식별하고 고객에게 풍부한 분석 정보를 제공하는 데 도움을 줍니다.



BACKEND FOR OBSERVABILITY PLATFORMS

수천만 건의 비디오 스트리밍 이벤트를 ClickHouse Cloud로 수집하고 쿼리를 실행하여 복잡한 대시보드를 생성함으로써 고객의 QOS 및 QOE를 실시간으로 모니터링, 알림 및 문제 해결을 수행합니다.



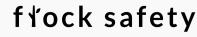
ALTERNATIVE TO TRADITIONAL WAREHOUSES

이에 비해 BigQuery는 성능이 떨어지고 스캔한 바이트 기준으로 과금하는 가격 모델로 인해 비용이 2배 더 많이 들었습니다. 이는 20B 행과 20TB 데이터로 구성된 단일 테이블을 사용하여 22 qps로 테스트한 결과입니다.



MODEL TRAINING AND INFERENCE ENGINE

ClickHouse는 ML 모델 학습을 위한 오프라인 피처 스토어입니다. 매우 빠르고 비용 효율적이지만, 가장 중요한 것은 모든 것과 연결된다는 점입니다. 빠른 반복 시간은 데이터 사이언스 팀 워크플로우의 핵심입니다.



ClickHouse의 기술적 차별점

1

매우 빠른 데이터 쿼리

대규모 데이터셋에 대한 분석 쿼리, 집계 및 계산이
매우 빠릅니다

2

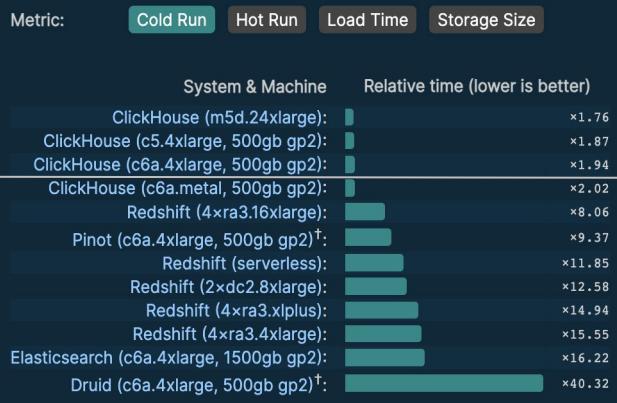
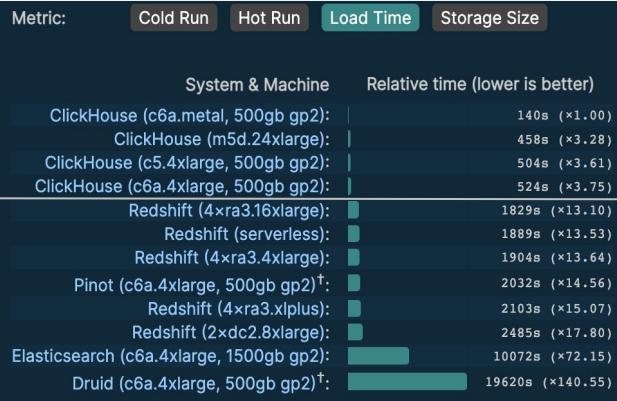
높은 리소스 효율성

업계 최고 수준의 데이터 압축 - 경쟁 제품
대비 10-100배의 스토리지 효율성을
제공합니다

3

쉬운 사용

다양한 소스(예: S3, Delta Lake, Iceberg, Hudi)에서
셀프 서비스 데이터 온보딩, 분석가 친화적인 표준
SQL 구문 활용이 가능합니다.



benchmark.clickhouse.com



ClickHouse의 차별성

데이터 레이크



분산 데이터 처리
프레임워크



데이터 웨어하우스



다른 데이터베이스 및
데이터소스



실시간 분석
데이터베이스



Slower

Ad-hoc 및
주기적 워크플로우
기본 데이터 탐색, 보고서 생성

빠른 응답을 요구하는
높은 동시성의 워크플로우
분석 워크플로우, 모니터링, 애플리케이션 인텔리전스

Faster

ClickHouse Cloud

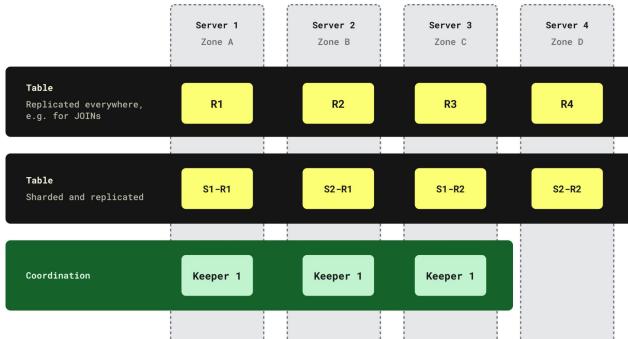


ClickHouse

Self-managed

- ✓ 인프라 환경 제약 없는 설치
- ✓ 높은 데이터 압축률
- ✓ 다양한 기능 포함
- ✓ 빠르고, 확장성 높은, 안정적 플랫폼
- ✓ 활발한 오픈소스 커뮤니티

Sample self-managed architecture



ClickHouse

단순화 된 운영, 빠른 Time-to-Value, 비용효율성

- ✓ 오토스케일링
- ✓ 자동화 된 백업
- ✓ 자동화 된 업그레이드 및 패치
- ✓ 컴퓨터 스토리지 분리
- ✓ SQL 콘솔 UI
- ✓ ClickPipes를 활용한 다양한 입수 파이프라인 자동화
- ✓ 오브젝트 스토리지 활용을 통한 멀티 레벨 캐싱
- ✓ 동적 스케일링 지원
- ✓ 데이터 암호화 지원
- ✓ 전담 기술 지원 서포트

The ClickHouse Cloud interface displays the following components:

- Services:** Shows Compute-Compute Separation Cluster, Network Event Logging, Production - Log Analytics, nitro, test_for_fun, Compute-Compute Ingest, and Compute-Compute Query.
- Monitoring:** Dashboards for Network Event Logging, Production - Log Analytics, and Compute-Compute Ingest.
- Data Sources:** Integration with various cloud services:
 - Cloud object stores: Amazon S3, Google Cloud Storage
 - Databases, data lakehouses: Postgres CDC, Amazon MSK, Amazon Kinesis, Confluent Cloud, Azure Event Hubs, WarpStream
 - Event streams: Apache Kafka, Redpanda
- Setup:** Step-by-step guide for "Select the data source" and "Setup your ClickPipe Connection".

ClickHouse Cloud

단순화 된 운영, 빠른 Time-to-Value, 비용효율성

고유의 아키텍처

- 컴퓨트-스토리지 분리 (Compute-storage separation)
- 컴퓨트-컴퓨트 분리 (Compute-compute separation)
- 자동 확장 및 수평 확장 지원 (Auto-scaling)
- 가용성과 복제 기능 (High availability and replication)
- 오브젝트 스토리지를 기반으로 한 계층형 스토리지(Tiered storage on top of object store, multi-level caching)

간소화된 관리

- 자동 업그레이드 및 패치
- 자동 백업 (Automated backups)

데이터 적재 및 분석 도구

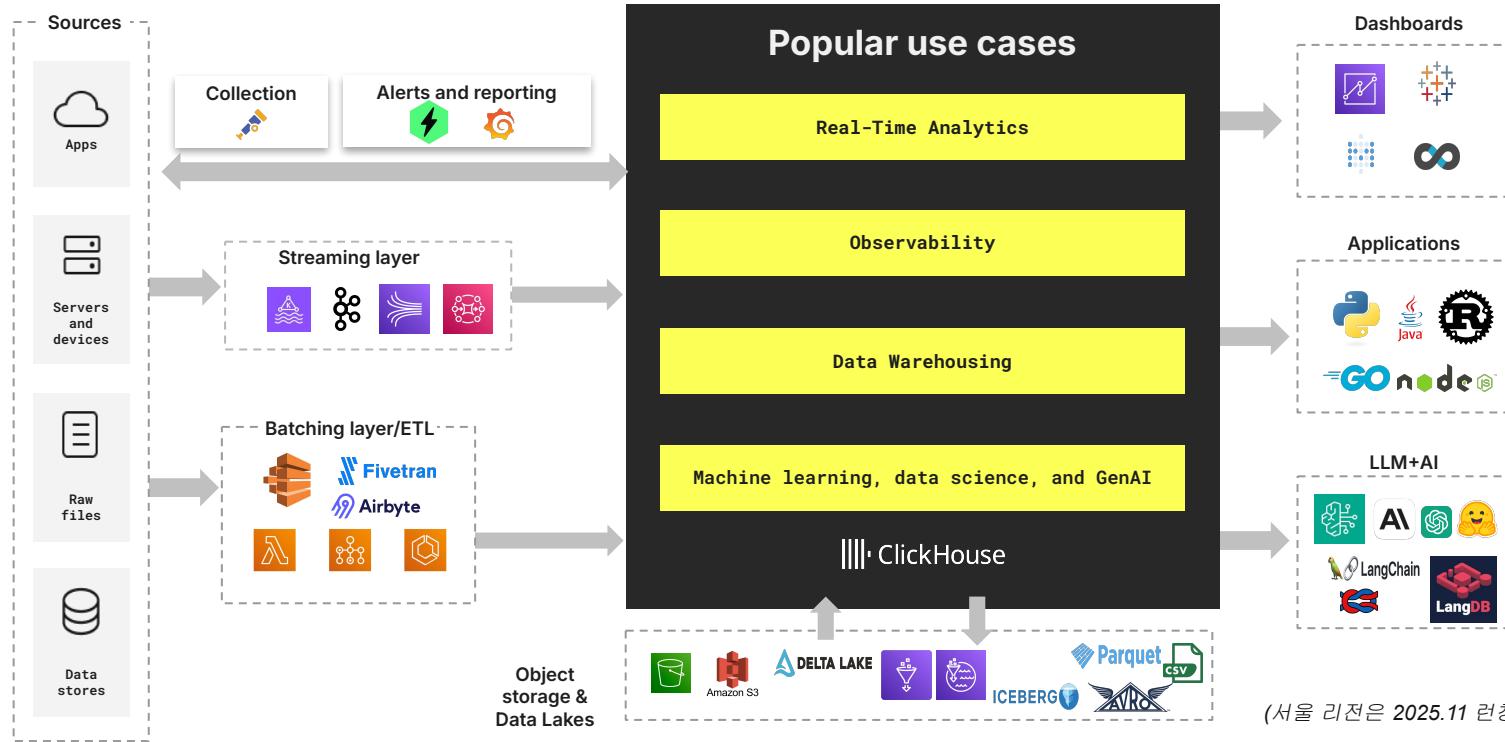
- 내장형 SQL 콘솔 UI 및 대시보드 도구
- ClickPipes를 이용한 강력한 데이터 수집 커넥터

보안 및 컴플라이언스

- 고객이 관리하는 데이터 암호화 키
- SOC2, HIPAA, PCI compliance
- BYOC

The screenshot shows the ClickHouse Cloud Services interface. At the top, there's a search bar and a 'Select a service' dropdown. Below that, a sidebar lists services: Network Event Logging, Production - Log Analytics, nitro, test_for_fun, Compute-Compute Separation Cluster, Compute-Compute Ingest, and Compute-Compute Query. The main area displays a 'Compute-Compute Separation Cluster' with two entries: 'Compute-Compute Ingest' (AWS Oregon us-west-2) and 'Production - Log Analytics' (GCP S. Carolina us-east1). Both entries show '8 GiB per replica / 2 replicas' and a 'Read-write' button. The 'Compute-Compute Separation Cluster' entry also has a 'Read-write' button. There are also sections for 'Network Event Logging' and 'Production - Log Analytics' with their respective details.

ClickHouse Cloud



(서울 리전은 2025.11 런칭 계획)

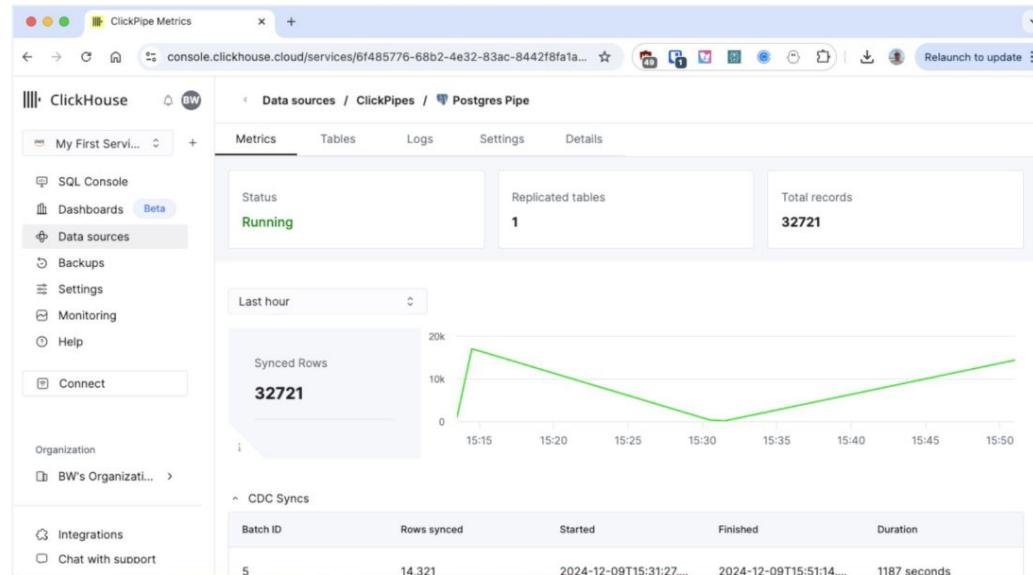
솔루션 최신 업데이트

ClickPipes (Postgres CDC)

AWS re:Invent 2024 공개

PeerDB 인수를 통한 통합

- ClickPipes에 통합됨
- Postgres 데이터를 ClickHouse로 이동하는 가장 빠르고 간단한 방법
- ETL 도구나 추가 인프라가 필요 없음
- MySQL, MongoDB CDC 추가 예정

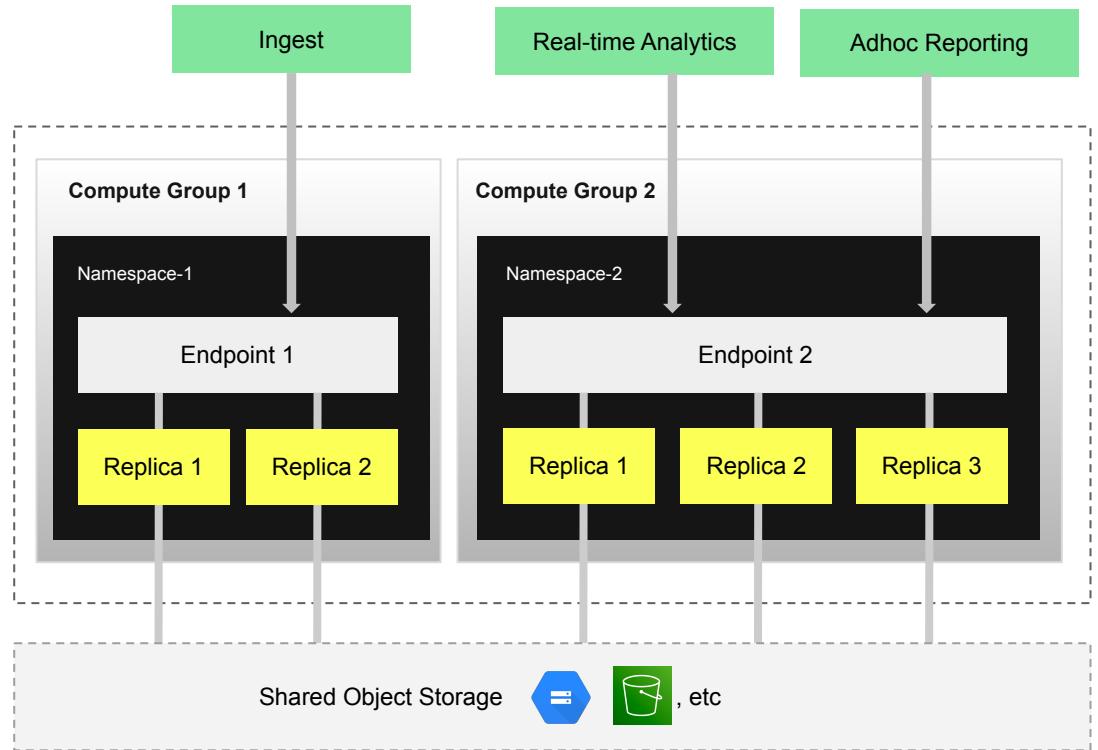


Compute-Compute 분리

2025년 1월 출시

서비스는 특정 작업(예: 스트리밍 수집
(Streaming Ingest) vs 임시 보고(Adhoc Reporting))에 전용 컴퓨팅 리소스를 할당할 수 있습니다. 이때 동일한 스토리지를 공유하면서도 각각의 연산 환경을 독립적으로 운영할 수 있습니다.

- 컴퓨트 유닛(Compute units)은 독립적으로 확장 가능
- 리소스 경쟁으로 인한 병목 현상 제거



관측(Observability) 활용 사례의 증가

JSON 데이터 탑재 지원 및 ClickHouse 네이티브 인터페이스 강화

JSONBench — a Benchmark For Data Analytics On JSON 🌙☀️

Methodology | Reproduce and Validate the Results | Add a System | Report Mistake | See also: ClickBench

System: All ClickHouse (lz4) ClickHouse (zstd) DuckDB Elasticsearch (no source, best compression)
Elasticsearch (no source, default) Elasticsearch (best compression) Elasticsearch (default) Elasticsearch
MongoDB (snappy, covered index) MongoDB (zstd, covered index)
PostgreSQL (lz4) PostgreSQL (pglz)

Scale: 1 million 10 million 100 million 1000 million

Metric: Cold Run Hot Run Storage Size Data Quality

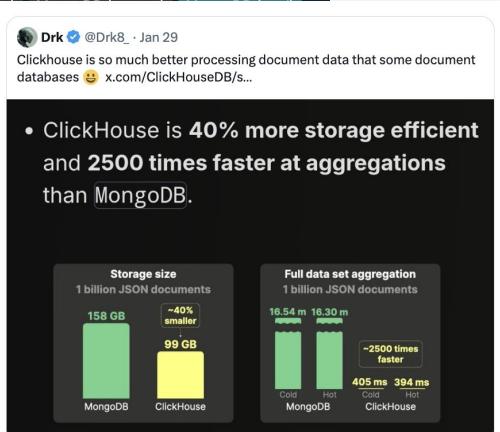
System Relative time (low) ClickHouse (lz4): | Elasticsearch (default): | DuckDB: | MongoDB (zstd): | PostgreSQL (lz4): |

Detailed Comparison

	ClickHouse (lz4)	Elasticsearch (default)	DuckDB
Data size:	148.86 GiB (x1.03)	454.54 GiB (x3.14)	440.14 GiB (x3.04)
Data quality:	999999258	999999153	9744000000
Q1.	0.389s (x1.00)	3.026s (x7.61)	3731.713s (x9352.69)
Q2.	5.542s (x1.00)	27.380s (x4.93)	3737.248s (x673.14)
Q3.	2.334s (x1.00)	24.787s (x10.58)	3729.542s (x1591.11)
Q4.	0.556s (x1.00)	8.731s (x15.44)	3735.854s (x6600.47)
Q5.	0.573s (x1.00)	9.733s (x16.71)	3736.934s (x6409.85)

3:29 PM · Jan 31, 2025 · 10.5K Views

1698.160s (x2912.81) 4873.680s (x8359.67)



ClickHouse speed + HyperDX UX = the observability platform you've been waiting for 🚀



ClickHouse + HyperDX

ClickHouse acquires HyperDX: The future of open-source observability

From clickhouse.com

8:01 AM · Mar 13, 2025 · 14.2K Views

1 28 89 21

Post your reply

Mr. Code @perpetualtalk · Mar 13 That's exciting news.

128



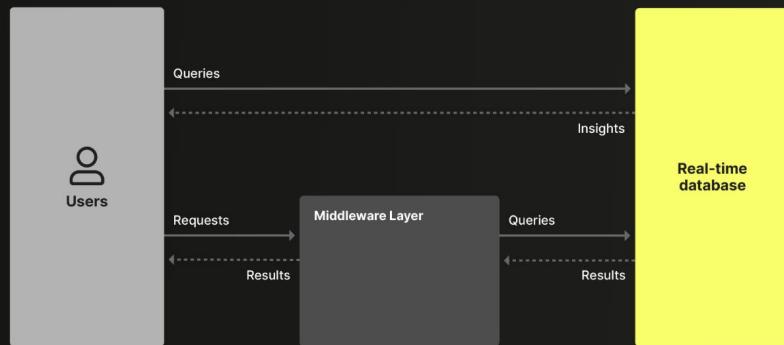
Agent (MCP) 분석 쿼리 대응

AI 에이전트는 실시간 데이터베이스의 새로운 사용자로 등장

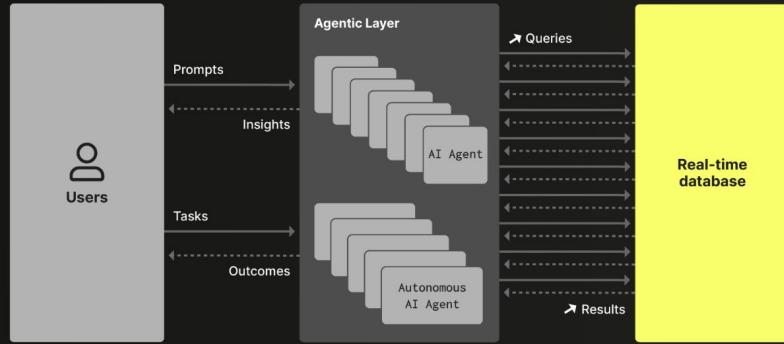
- AI 에이전트 기반 도구는 점점 성숙해지고 접근성이 높아짐 (MCP 표준의 도입이 이러한 트렌드를 가속화)
- 이러한 에이전트들은 사용자를 대신해 데이터베이스와 직접 상호작용하며, 사용자당 훨씬 더 많은 쿼리를 생성
- 이러한 워크로드는 대규모 데이터셋에서의 빠른 쿼리 속도를 요구하며, 실시간 상호작용형 경험(*interactive experience*)을 위해 필수적임
- 실시간 분석 데이터베이스(Real-time analytical DB)의 새로운 트렌드의 가장 큰 수혜자가 될 것으로 전망

<https://clickhouse.com/blog/agent-facing-analytics>

Classic real-time analytics



Agentic real-time analytics





그리고 내가 가장 좋아하는 기능들

My Favourite Features

Derek Chia (derek.chia@clickhouse.com)

Principal TAM Architect
ClickHouse Inc.

2025년 11월

A view into the list of features in 2025

ClickHouse

Release 25.1

- MinMax index at table level
- Faster parallel hash join
- Better Merge tables
- Auto increment
- ...and many more

Engineering

ClickHouse Release 25.1

The ClickHouse Team · Feb 7, 2025

ClickHouse

Release 25.2

- Streaming events over HTTP
- Minimizing join contention
- Bloom filters for Parquet
- Backup database engine
- ...and many more

Engineering

ClickHouse Release 25.2

The ClickHouse Team · Mar 7, 2025

ClickHouse

Release 25.3

- AWS Glue and Unity catalogs
- Query condition cache
- New array functions
- CREATE DATABASE demo_catalog ENGINE = DataLakeCatalog SETTINGS catalog_type = 'glue', region = 'us-west-2', aws_access_key_id = 'AKIA...', aws_secret_access_key = '....'
- ...and many more

Engineering

ClickHouse Release 25.3

The ClickHouse Team · Mar 27, 2025

ClickHouse

Release 25.4

- Correlated subqueries for EXISTS
- Apache Iceberg time travel
- Lazy materialization
- ...and many more

Engineering

ClickHouse Release 25.4

The ClickHouse Team · May 9, 2025

ClickHouse

Release 25.5

- Implicit table in clickhouse-local
- Vector search is beta
- Geo types in Parquet
- ...and many more

Engineering

ClickHouse Release 25.5

The ClickHouse Team · Jun 2, 2025

ClickHouse

Release 25.6

- Filtering by multiple projections
- Bloom filter index optimization
- Single snapshot for SELECT
- ...and many more

Engineering

ClickHouse Release 25.6

ClickHouse Team · Jul 2, 2025

ClickHouse

Release 25.7

- Inserts Into Apache Iceberg
- AI-powered SQL generation
- Lightweight updates
- ...and many more

Engineering

ClickHouse Release 25.7

ClickHouse Team · Aug 8, 2025

ClickHouse

Release 25.8

- Writes with Hive-style partitioning
- A new, faster Parquet reader
- Data Lake improvements
- ...and many more

Engineering

ClickHouse Release 25.8

ClickHouse Team · Sep 5, 2025

ClickHouse

Release 25.9

- Secondary indices streaming
- Global join reordering
- A new text index
- ...and many more

Engineering

ClickHouse Release 25.9

The ClickHouse Team · Oct 2, 2025

ClickHouse

Release 25.10

- Lazy column replication in JOINs
- Bloom filters in JOINs
- QBit data type
- ...and many more

Engineering

ClickHouse Release 25.10

The ClickHouse Team · Nov 7, 2025

<https://clickhouse.com/blog>

Ver.	Features, Optimizations, Fixes	Highlights
25.1	15 new features 🐔 36 performance optimizations 🛠️ 77 bug fixes 🌋	In this release, we've accelerated the parallel hash join algorithm using a two-level hash map, introduced MinMax indices at the table level , improved Merge tables, added auto-increment functionality, and more!
25.2	12 new features 🐔 15 performance optimizations 🍔 72 bug fixes 🌸	This release brings improved parallel hash join performance, Parquet bloom filter writing , transitive condition inference for queries, a backup database engine , delta rust kernel integration , and more!
25.3	18 new features 🌱 13 performance optimizations 🐔 48 bug fixes 🌸	This release brings query support for the AWS Glue and Unity catalogs , the new query condition cache , automatic parallelization when querying S3 , and new array functions! JSON data type is production-ready .
25.4	25 new features 🌸 23 performance optimizations 🦋 58 bug fixes 🐝	This release brings lazy materialization , Apache Iceberg time travel , correlated subqueries for the EXISTS clause , and more!
25.5	15 new features 🌸 23 performance optimizations 🦋 64 bug fixes 🐝	This release sees the vector similarity index move into beta, support for the Hive metastore catalog for Iceberg tables , the ability to track when new functions were added to ClickHouse, implicit table in clickhouse-local , and more.
25.6	25 new features 🌸 32 performance optimizations 🦋 105 bug fixes 🐝	This release brings a single consistent snapshot across all subqueries and table references within a query , filtering by multiple projections , JSON in Parquet , and more!
25.7	27 new features 🌸 26 performance optimizations 🍦 98 bug fixes 🐞	This release brings a targeted optimization for count() aggregations that reduces memory and CPU usage, making these queries even faster.
25.8	45 new features 🌻 47 performance optimizations 🐞 119 bug fixes 🐝	This release brings a new, faster Parquet reader , Data Lake improvements , writes with Hive-style partitioning , initial PromQL support, and more!
25.9	25 new features 🍎 22 performance optimizations 🍁 83 bug fixes 🌱	This release brings automatic global join reordering , streaming for secondary indices , a new text index , and more!
25.10	20 new features 🐚 30 performance optimizations 🌎 103 bug fixes 🎃	This release introduces a collection of join improvements , a new data type for vector search , late materialization of secondary indices , and more!



Focus Area for 2025

JOINS	Distributed Query	Data Lakes	Mutable Data	Inverted Index
<p>Expand supports for JOINs.</p> <ul style="list-style-type: none">• Focus on Performance. Improvement on TPC-H benchmark• Better default experience• Better resource usage	<p>Distribution of queries designed for cloud workload</p> <ul style="list-style-type: none">• Leverage replicas• Expanded support for parallel replica• Better default experience	<p>Focus on Iceberg and Delta Lake support</p> <ul style="list-style-type: none">• Catalog integrations• Best-in class support for Iceberg and Delta• Expand Data Lake operational workload	<p>Improved experience with mutable data</p> <ul style="list-style-type: none">• Lightweight operations• Improved mutations internals• RMT Improvements	<p>Better experience for observability workload</p> <ul style="list-style-type: none">• Additional index type• Index aimed at improving search experience for logging workload



Recent acquisitions in 2025 - My favourite features



LibreChat



**ClickHouse acquires LibreChat:
Introducing the open-source
Agentic Data Stack**

||||· ClickHouse

Product

**ClickHouse welcomes LibreChat:
Introducing the open-source
Agentic Data Stack**

Ryadh Dahimene and Danny Avila · Nov 4, 2025

<https://clickhouse.com/blog/clickhouse-acquires-hyperdx-the-future-of-open-source-observability>



HyperDX

||||· ClickHouse + ⚡ HyperDX

Product

**ClickHouse acquires HyperDX: The
future of open-source observability**

The ClickHouse Team · Mar 13, 2025

<https://clickhouse.com/blog/clickhouse-acquires-hyperdx-the-future-of-open-source-observability>

|||· ClickHouse

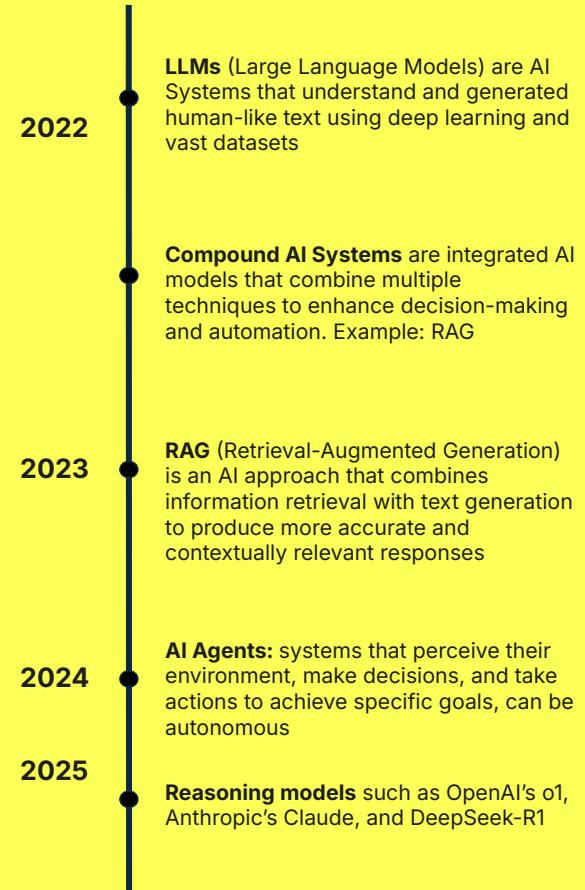
LibreChat

github.com/danny-avila/LibreChat

The Emergence of Agents and Reasoning Models

An agentic revolution

- Powered by LLMs, agents can understand, decide, and act
- Tools supercharge agents with real-world abilities
- Reasoning models introduce deliberate thinking before action
- AI agents are a new kind of “user” - tireless, autonomous, and prolific



Real-time analytics needs...

Always fresh data

with

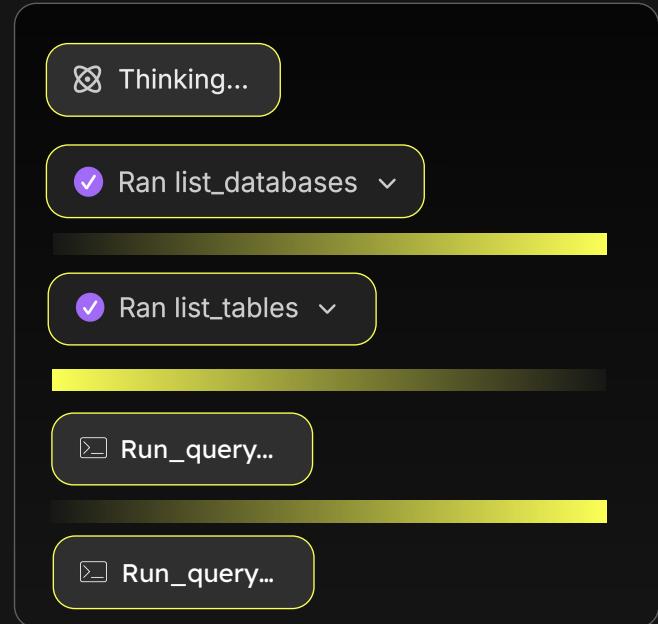
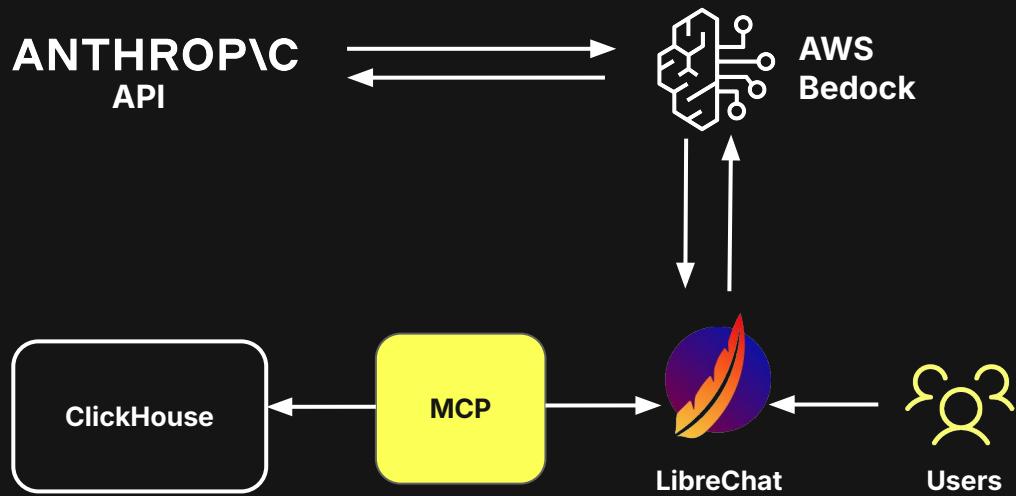
blazing fast queries

and

scalability to ~~thousands of users~~

millions of agents

Architecture





Model Context Protocol (MCP)

LLMs mostly rely on training data

MCP links AI apps with external tools and private data

EXAMPLES

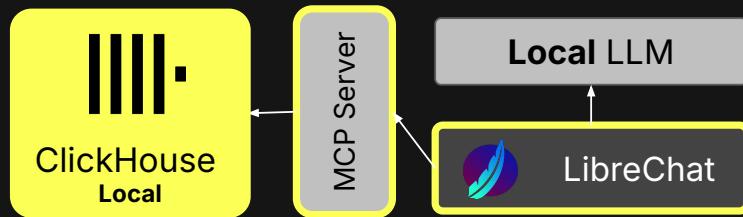
- Company DB queries
- Local file access
- Smart-home control
- Live service data

Fully managed remote MCP server

- Turn-key experience, no infra to setup or manage
- Built into ClickHouse Cloud
- Leverage your data with external agents and MCP-compatible clients
- Secured with OAuth for authentication
- Bring your own MCP-compatible client
 - Claude, Cursor, Windsurf, 25+ others

ClickHouse + mcp-clickhouse + LibreChat

1. Everything Local

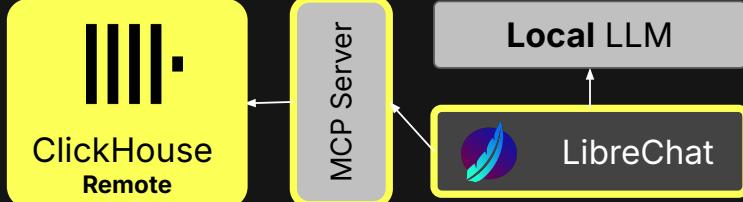


Secure, air-gapped access to ClickHouse, LibreChat and local LLM through ClickHouse MCP server

Deployment Architecture

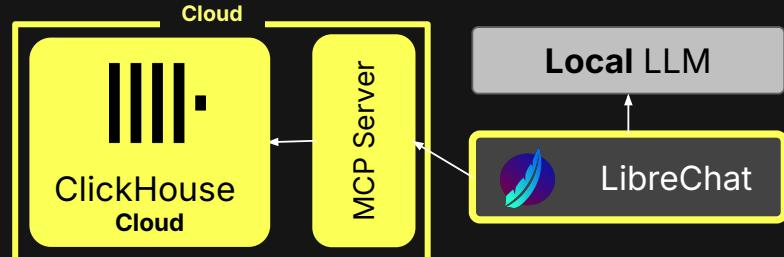
1. **ClickHouse**: Local, Remote or Cloud
2. **ClickHouse MCP Server**: Offers tools such as `list_databases`, `list_tables`, `run_select_query`
3. **LibreChat**: ChatGPT clone with chat interface for Agents & Tools connectivity over MCP

2. Remote ClickHouse



Connection to any remote ClickHouse

3. ClickHouse Cloud



Connection to ClickHouse Cloud with integrated MCP server
Secured with OAuth for authentication

Demo



ClickStack

clickhouse.com/o11y

github.com/hyperdxio/hyperdx



ClickStack

The ClickHouse Observability Stack

- Open source across the whole stack
- Deploy and run in minutes anywhere with any schema
- Built for observability: frontend to infra
- First class OpenTelemetry and JSON support

SQL

Powered

High
Cardinality
Event
Deltas

Natural Language Search

10x to 100x

Compression Ratios

Open
Source

Logs
Traces
Metrics
Exceptions

Session

Real-Time Fast

< 500ms on 50PB

ClickStack

ClickHouse · HyperDX · OpenTelemetry

Compute
Storage
Separation

Schema Agnostic

Any Schema, Not
Just Ours

↗ OpenTelemetry
Native



Alerts



4x More Telemetry Stored



Session Replay

Open Formats

Parquet Iceberg Delta Avro

ClickStack Public Demo

HyperDB

Search

Client Sessions

Service Map **BETA**

Dashboards

+ Create Dashboard

PRESETS

ClickHouse Services

Kubernetes

Ready to deploy on ClickHouse Cloud?

Get Started for Free

Demo Logs **Schema**

SELECT Timestamp, ServiceName, SeverityText, Body

ORDER BY (Time)

460,006 Results

Search your events w/ Lucene ex. column:foo

SQL | Lucene Live Tail

Results Table

Event Patterns

Filters

Denoise Results

ServiceName

- accounting
- ad
- artillery-loadg-
- cart
- checkout
- currency
- email
- fraud-detection
- frontend
- frontend-proxy
- Show more
- SeverityText
- ResourceSche
- ScopeVersion

More filters ▾

Timestamp (Local)	ServiceName	SeverityText	Body
Nov 17 10:14:16.978 AM	frontend-proxy	info	[2025-05-25T19:13:22.998Z] *GET /api/recommendations?prox...
Nov 17 10:14:16.967 AM	frontend-proxy	info	[2025-05-25T19:13:22.978Z] *GET /api/products/9ECAV9K1...
Nov 17 10:14:16.955 AM	cart	info	GetCartAsync called with userId=53288ec-399c-11f0-8cd...
Nov 17 10:14:16.954 AM	ad	trace	info: cart.cartstore.ValleyCartStore[0]
Nov 17 10:14:16.954 AM	cart	Information	GetCartAsync called with userId=[userId]
Nov 17 10:14:16.954 AM	ad	INFO	Targeted ad request received for [assembly]
Nov 17 10:14:16.948 AM	frontend-proxy	info	[2025-05-25T19:13:22.959Z] *GET /api/data/?contextKey=t...
Nov 17 10:14:16.947 AM	ad	trace	2025-05-25 19:13:22 - oteldemo.AdService - Targeted ad :
Nov 17 10:14:16.946 AM	ad	INFO	Targeted ad request received for [assembly]
Nov 17 10:14:16.946 AM	ad	trace	2025-05-25 19:13:22 - oteldemo.AdService - Non-targeted
Nov 17 10:14:16.946 AM	ad	trace	2025-05-25 19:13:22 - oteldemo.AdService - Targeted ad :
Nov 17 10:14:16.946 AM	ad	INFO	Non-targeted ad request received, preparing random response.
Nov 17 10:14:16.946 AM	ad	trace	2025-05-25 19:13:22 - oteldemo.AdService - Targeted ad :
Nov 17 10:14:16.946 AM	ad	INFO	Targeted ad request received for [assembly]
Nov 17 10:14:16.946 AM	ad	INFO	[2025-05-25T19:13:22.957Z] *GET /api/products/2ZYFJ3GM2N HTTP/1.1" 200 -
Nov 17 10:14:16.946 AM	ad	trace	2025-05-25 19:13:22 - oteldemo.AdService - Targeted ad request received f...
Nov 17 10:14:16.946 AM	ad	INFO	Targeted ad request received for [accessories]
Nov 17 10:14:16.946 AM	ad	trace	2025-05-25 19:13:22 - oteldemo.AdService - Non-targeted ad request receiv...
Nov 17 10:14:16.946 AM	ad	INFO	Targeted ad request received for [binoculars]
Nov 17 10:14:16.946 AM	ad	INFO	Targeted ad request received for [accessories]
Nov 17 10:14:16.945 AM	ad	INFO	Non-targeted ad request received, preparing random response.
Nov 17 10:14:16.941 AM	frontend-proxy	info	[2025-05-25T19:13:22.951Z] *GET /api/recommendations?productId=9ECAV9K1...
Nov 17 10:14:16.939 AM	frontend-proxy	info	[2025-05-25T19:13:22.959Z] *GET /api/products/64vCH5HNUP HTTP/1.1" 200 -
Nov 17 10:14:16.937 AM	frontend-proxy	info	[2025-05-25T19:13:22.948Z] *GET /api/products/2ZYFJ3GM2N HTTP/1.1" 200 -
Nov 17 10:14:16.910 AM	frontend-proxy	info	[2025-05-25T19:13:22.921Z] *GET /api/data/?contextKey=telescopes HTTP/1.1" 200 -
Nov 17 10:14:16.905 AM	frontend-proxy	info	[2025-05-25T19:13:22.916Z] *GET /api/recommendations?productId=64vCH5HNU...
Nov 17 10:14:16.894 AM	frontend-proxy	info	[2025-05-25T19:13:22.905Z] *GET /api/data/?contextKey=telescopes HTTP/1.1" 200 -
Nov 17 10:14:16.890 AM	frontend-proxy	info	[2025-05-25T19:13:22.901Z] *GET /api/products/0PUK46EV0 HTTP/1.1" 200 -
Nov 17 10:14:16.884 AM	frontend-proxy	info	[2025-05-25T19:13:22.899Z] *GET /api/cart HTTP/1.1" 200 - via upstream -
Nov 17 10:14:16.881 AM	frontend-proxy	info	[2025-05-25T19:13:22.897Z] *GET /api/products/4F027WVYCF HTTP/1.1" 200 -

HyperDB

Kubernetes Dashboard **Demo Logs** **Schema** **Demo Metrics** **Schema**

Nov 15 10:16:49 - Nov 17 10:16:49

Pod Node Namespace Cluster

Search query

Chart Explorer Client Sessions Service Map Dashboards

PRESETS ClickHouse Services Kubernetes

CPU Usage Memory Usage

Ready to deploy on ClickHouse Cloud? Get Started for Free

Presets ClickHouse Services Kubernetes

Running Succeeded Pending Failed All

Name	Namespace	Node	Status	CPU/Limit	Mem/Limit	Age	Restarts
flag-b77cd4089-hnms	otel-demo	gke-pew-k8s-standard-pool-e23e	HyperDB	jitt			
collector-b21wd	gip-system	gip-pew-k8s-standard-pool-768d	Client Sessions	ktx			
gke-metrics-agent-ftvg	kube-system	gke-pew-k8s-standard-pool-e23e	Metrics	dashboards			

HTTP Service Database Errors

Request Error Rate Overall By Endpoint

Latest Kubernetes Warning Events

Timestamp (Local) Severity Kind Node

Timestamp (Local)	Severity	Kind	Node
Nov 17 10:14:16.940 AM	Warning	Node	node
Nov 17 10:14:16.940 AM	Warning	Node	node

Ready to deploy on ClickHouse Cloud? Get Started for Free

Presets ClickHouse Services Kubernetes

Top 20 Most Time Consuming Endpoints

Endpoint	Req/Min	P95 (ms)	Median (ms)	Total (ms)	Errors/Min
GET /	0	11245.54	5706.23	1284983.53	0
GET /avatax/croute	1.7	359.38	63.64	2289978.83	0
GET /	0	0.348	0.298	0.298	0
GET /packages/route	0	0.298	0.298	0.298	0
HEAD /dashboard/{package_name}	0	0.008	0.008	0.008	0

Request Latency 95th Percentile Median Avg

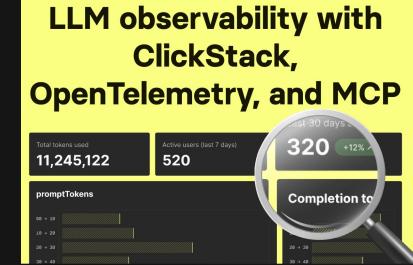
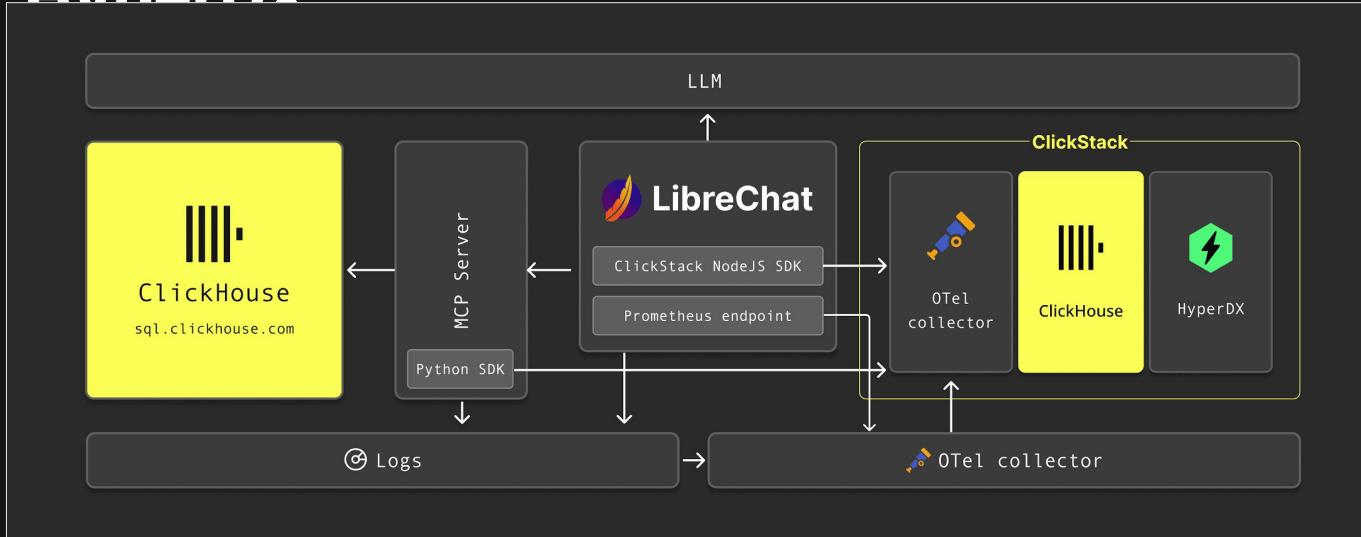
Top 20 Most Time Consuming

Endpoint	Req/Min	P95 (ms)	Median (ms)	Total (ms)	Errors/Min
GET /	0	11245.54	5706.23	1284983.53	0
GET /avatax/croute	1.7	359.38	63.64	2289978.83	0
GET /dashboard/{package_name}	7.4	8323.33	3648.22	35416538.18	0

Sort by Time Sort by Errors

<https://play-clickstack.clickhouse.com>

Putting this together: ClickHouse + mcp-clickhouse + LibreChat + HyperDX



- Instrumenting LibreChat with MCP server using OpenTelemetry (OTel)
- Capture meaningful traces and metrics from LibreChat with minimal code changes
- Define thresholds and automatic alerting on high usage or unexpected spikes

<https://clickhouse.com/blog/llm-observability-clickstack-mcp>

Putting this together: ClickHouse + mcp-clickhouse + LibreChat + HyperDX

Logs

Cloud

Thoughts

I'll help you explore the relationship between USD and GBP. Let me first check what databases and tables are available that might contain relevant financial or currency data.

Great! I can see there are several relevant databases including "torus" and "stock" that might contain USD/GBP data. Let me explore these databases to see what tables are available.

Run all_tables

Perfect! I can see there are several forex tables with bid/ask data for different currency pairs. Let me explore what USD/GBP data is available and analyze their relationship.

Run mcp_select_query

Great! I can see we have GBP/USD data (British Pound to US Dollar). Let me analyze the relationship between these currencies by looking at recent exchange rate data, trends, and statistics.

Run mcp_select_query

Let me check if the query with the duplicate alias:

Run mcp_select_query

Let me check what date range is available in the data:

Run mcp_select_query

Perfect! Now we have GBP/USD data from 2000 to 2022. Let me analyze the relationship between USD and GBP by looking at recent trends and key statistics:

Run mcp_select_query

Now let me also look at longer-term historical trends and key statistics to better understand the USD/GBP relationship:

Run mcp_select_query

Let me also check for any correlations and volatility patterns:

Run mcp_select_query

Now let me check some recent patterns and correlations:

Run mcp_select_query

Based on my analysis of the forex data, here's how USD and GBP are related:

USD/GBP Relationship Analysis

Current Exchange Rate Pattern

- GBP/USD is the currency pair traded (British Pound priced in US Dollars)
- Recent 2022 data shows GBP/USD trading around 1.17-1.37 range, meaning 1 British Pound = ~1.17-1.37 US Dollars

Historical Trends & Key Relationships

1. Long-term Decline from Peak (2007-2022)

- 2007 Peak: GBP/USD averaged ~2.00 (1 GBP = 2 USD)
- 2022 Recent: GBP/USD averaged ~1.25 (1 GBP = 1.25 USD)
- This represents a ~37% decline in GBP strength relative to USD over 15 years

2. Major Crisis Impact

- 2008 Financial Crisis: Extreme volatility (5% swing from 1.35 to 2.04)
- Brexit Referendum (2016): 25% volatility, GBP weakened significantly
- COVID-19 (2020): 20% volatility, temporary weakening

3. Recent USD Strength

- 2022 shows particularly strong USD performance
- GBP has multi-decade lows against USD since 1971

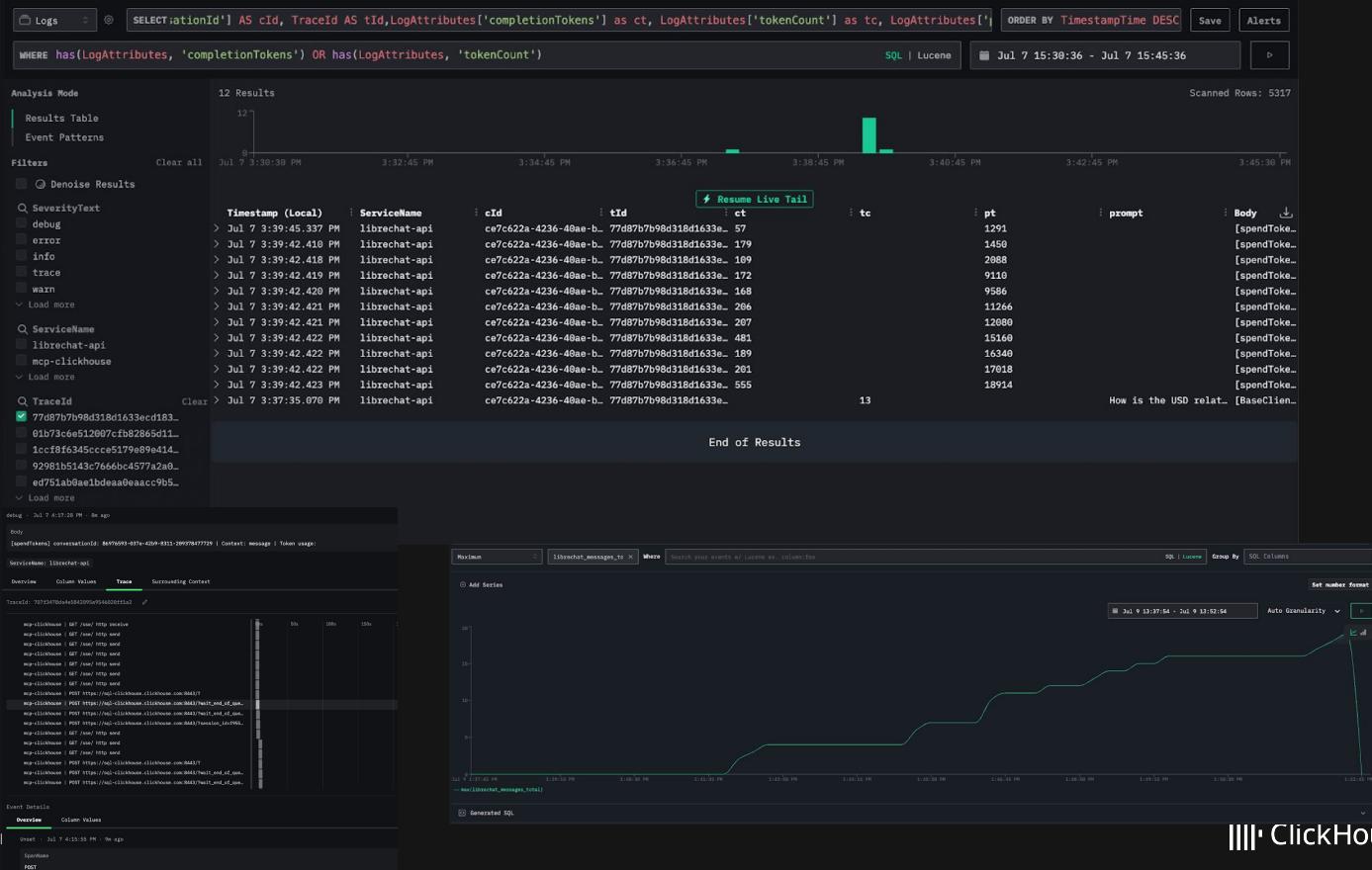
Economic Relationship Factors

1. Inverse Relationship

- When USD strengthens → GBP/USD rate falls
- When GBP strengthens → GBP/USD rate rises

2. Key Drivers

- Interest Rate Differentials: Fed vs Bank of England policies
- Economic Performance: GDP growth, inflation, employment
- Political Stability: Brexit impact, government policies



ClickHouse

**now supports
faster JOINS.**

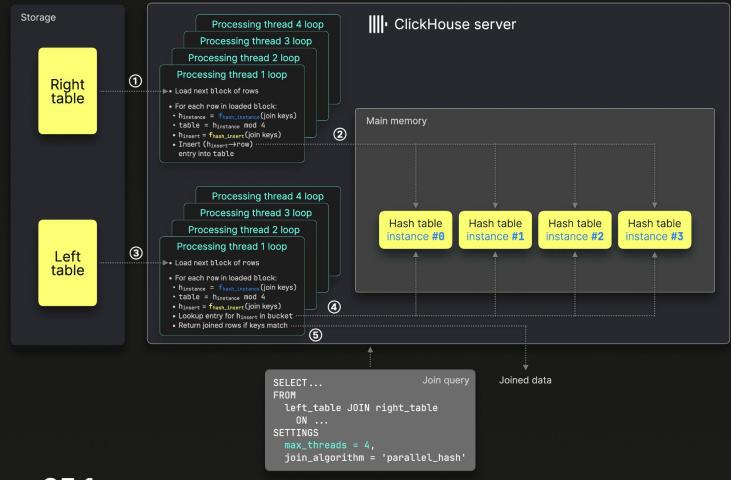
JOINS in ClickHouse

Great support for:

- Inner, left/right/full outer, ASOF JOINs
- ALL, ANY, SEMI, ANTI JOIN strictnesses
- Equi and non-equi JOIN predicates
- Hash, sort-merge, direct JOIN algorithms
- In-memory and disk-spilling JOINs

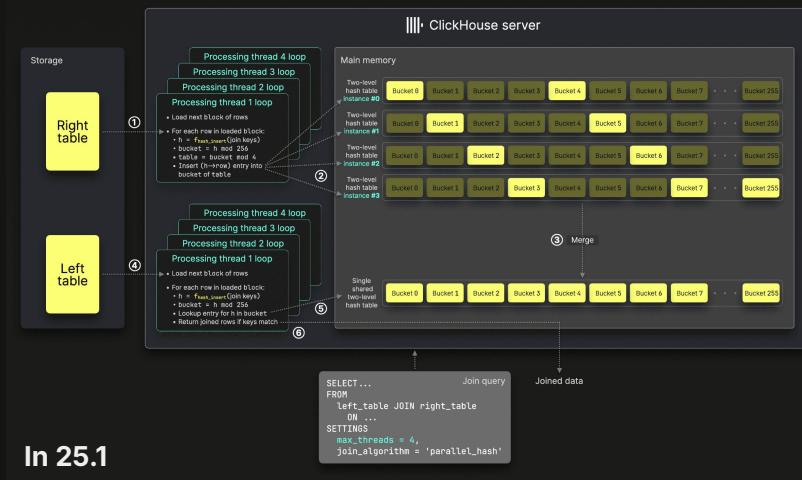
The most common paths are now much faster!

Improvement #1 - Parallel Hash JOIN (25.1, 25.2, ...)



Before 25.1

- Build phase:** in a loop, data from the **right table** is split and each row is hashed into **multiple hash table instance**.
- Probe phase:** similar to build phase, stream data from the **left table** and perform lookup against hash table. If lookup is success, returned joined rows
- Parallel hash join's concurrent **build phase** introduces overhead in the **probe phase**, as input blocks from the **left table** must first be split and routed to the appropriate hash table instances.



In 25.1

- Build phase** is still executed concurrently. However, when `max_threads` is set to N, instead of filling N separate hash table instances in parallel, the process now uses N **two-level hash table instances**.
- Probe phase** now uses a **single shared hash table** that all processing threads can access concurrently, just like in the non-parallel hash join. This eliminates the need for input block splitting, reduces overhead, and improves efficiency.

Improvement #1 - Parallel Hash JOIN (25.1, 25.2, ...)

```
SELECT count(c), version() FROM numbers_mt(200_000_000) AS a
INNER JOIN
(
    SELECT number, toString(number) AS c FROM numbers(2_000_000)
) AS b ON (a.number % 10_000_000) = b.number
SETTINGS join_algorithm = 'parallel_hash'
```

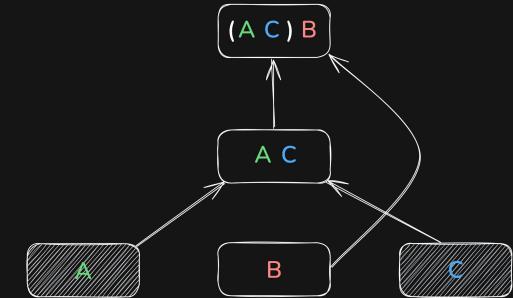
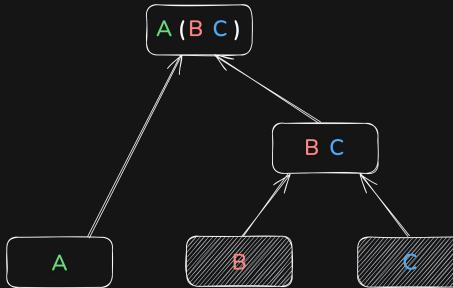
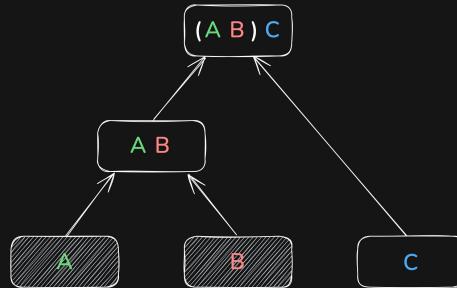
24.12	docker run -d --name clickhouse-server-24-12 -p 2412:9000 -e CLICKHOUSE_SKIP_USER_SETUP=1 clickhouse@clickhouse-server:24.12	1 row in set. Elapsed: <u>1.903 sec.</u> . Processed 202.00 million rows, 1.62 GB (106.18 million rows/s., 849.40 MB/s.) Peak memory usage: 214.69 MiB.
25.1	docker run -d --name clickhouse-server-25-01 -p 2501:9000 -e CLICKHOUSE_SKIP_USER_SETUP=1 clickhouse@clickhouse-server:25.1	1 row in set. Elapsed: <u>1.045 sec.</u> . Processed 202.00 million rows, 1.62 GB (193.37 million rows/s., 1.55 GB/s.) Peak memory usage: 210.41 MiB.
25.10	docker run -d --name clickhouse-server-25-10 -p 2510:9000 -e CLICKHOUSE_SKIP_USER_SETUP=1 clickhouse@clickhouse-server:25.10	1 row in set. Elapsed: <u>0.769 sec.</u> . Processed 202.00 million rows, 1.62 GB (262.59 million rows/s., 2.10 GB/s.) Peak memory usage: 244.91 MiB.



Improvement #2 - Automatic JOIN reordering (24.12)

JOINS are associative

```
SELECT ...  
FROM A, B, C  
WHERE ...
```



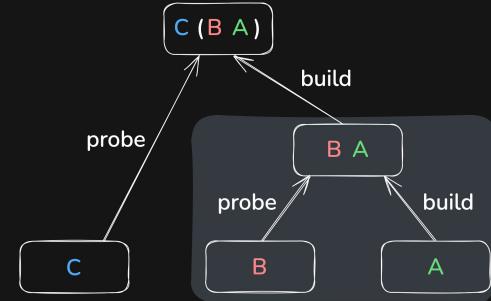
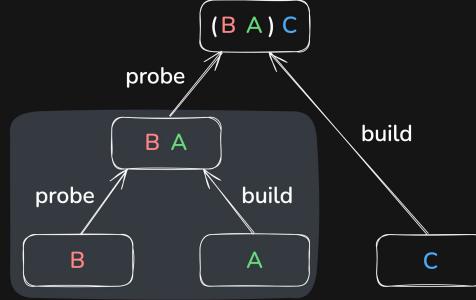
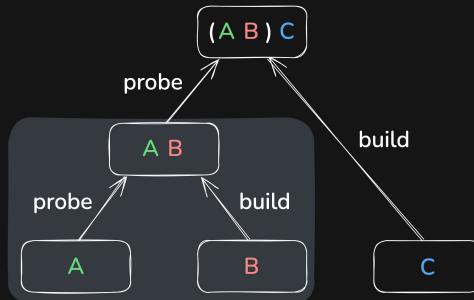
Goal: minimize hash table size for maximum Lx cache efficiency

Since results for all 3 scenarios are the same, in 24.12, ClickHouse now automatically moves the smaller of both tables to the build side. This optimization moves the small of both tables to the right (build) side and therefore reduces the effort needed to build the hash table.

Improvement #2 - Better Automatic JOIN reordering (25.9)

Global JOIN reordering

```
SELECT ...  
FROM A, B, C  
WHERE ...
```



Global JOIN reordering is much harder! As the number of possible join orders grows exponentially with the number of joined tables, ClickHouse cannot explore the search space exhaustively.

In 25.9, ClickHouse introduced global join reordering to determine the optimal join order (build vs probe) of more than two tables during query optimization. This is done through column statistics.

Improvement #2 - Better Automatic JOIN reordering (25.9)

```
create or replace table t_big (id UInt32, a UInt32, b UInt32) order by id;
create or replace table t_medium (id UInt32, a UInt32, b UInt32) order by id;
create or replace table t_small (id UInt32, a UInt32, b UInt32) order by id;

insert into t_big select number, rand(), number % 1_000_000 from numbers100_000_000;
insert into t_medium select number, rand(), number % 100_000 from numbers1_000_000;
insert into t_small select number, rand(), number % 10_000 from numbers100_000;
```

```
select * from t_small join t_medium using b join t_big using b
```

24.11	-	100000000 rows in set. Elapsed: 6.553 sec . Processed 101.10 million rows, 1.21 GB (15.43 million rows/s., 185.15 MB/s.) Peak memory usage: 2.80 GiB.
25.8	settings query_plan_join_swap_table = 'auto';	100000000 rows in set. Elapsed: 4.682 sec . Processed 101.10 million rows, 1.21 GB (21.59 million rows/s., 259.12 MB/s.) Peak memory usage: 3.68 GiB.
25.10	settings query_plan_join_swap_table = 'auto', query_plan_optimize_join_order_limit = 3, allow_statistics_optimize = 1	100000000 rows in set. Elapsed: 2.407 sec . Processed 101.10 million rows, 1.21 GB (42.00 million rows/s., 504.05 MB/s.) Peak memory usage: 132.99 MiB.



ClickHouse now supports faster JOINS

Faster parallel hash JOIN, automatic JOIN reordering

JOIN ME IF YOU CAN
ClickHouse vs Databricks & Snowflake



Engineering

Join me if you can: ClickHouse vs. Databricks vs. Snowflake - Part 1

AI Brown and Tom Schreiber · Jun 24, 2025

JOIN ME IF YOU CAN
ClickHouse vs Databricks & Snowflake

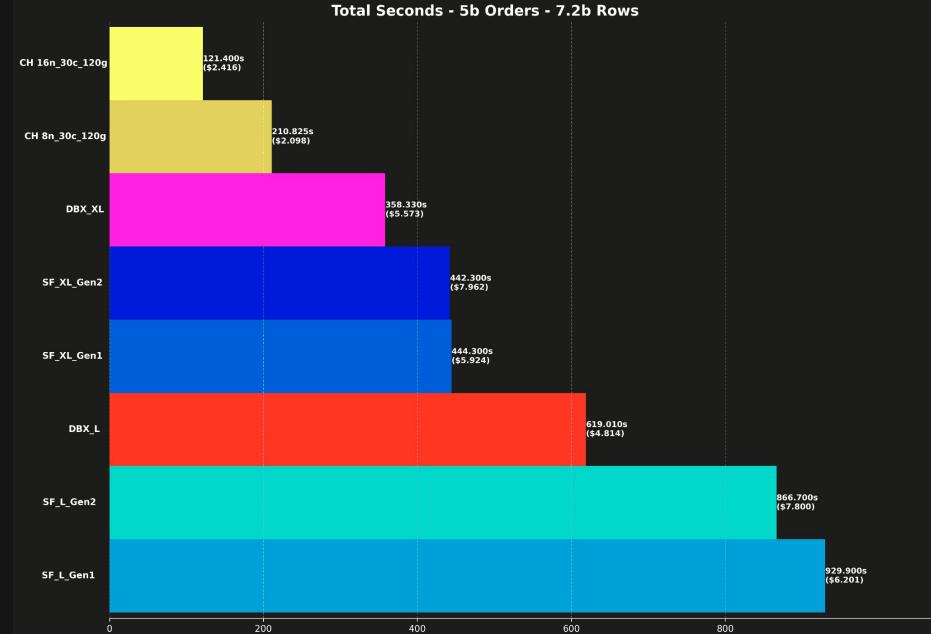


Engineering

Join me if you can: ClickHouse vs. Databricks vs. Snowflake - Part 2

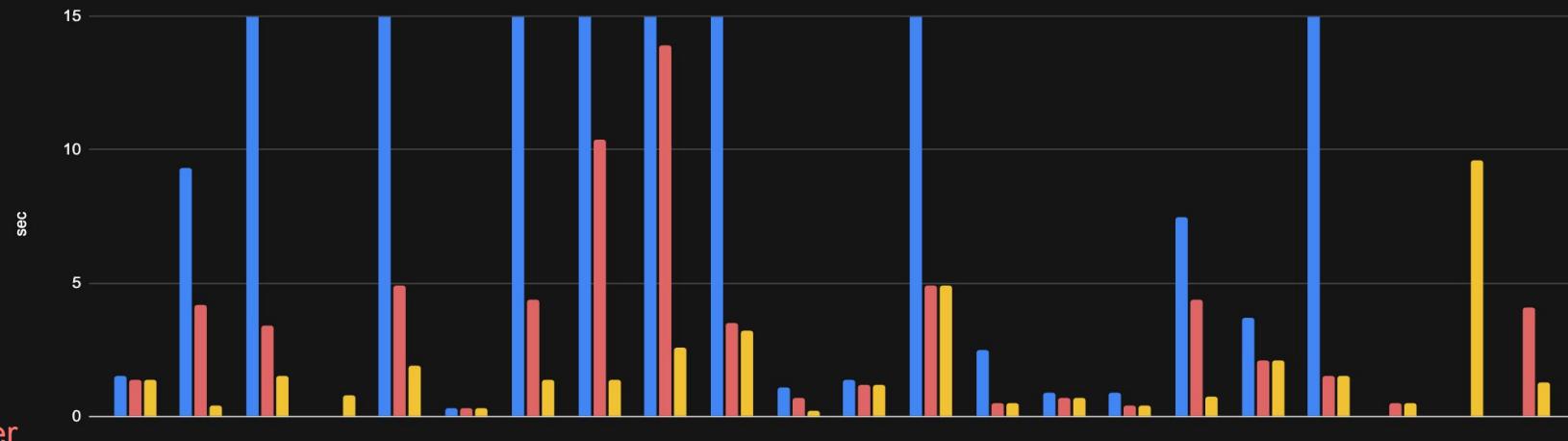
AI Brown and Tom Schreiber · Jun 25, 2025

<https://clickhouse.com/blog/join-me-if-you-can-clickhouse-vs-databricks-snowflake-join-performance>
<https://clickhouse.com/blog/join-me-if-you-can-clickhouse-vs-databricks-snowflake-part-2>



Benchmarks: TPC-H

- scale factor = 100
- 32 vCPUs, 64 GB DRAM



20.7x faster



	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22
v24.10	1.5	9.3	52.5	oom	72.7	0.3	78.8	>300	>300	23.5	1.1	1.4	45.3	2.5	0.9	0.9	7.5	3.7	>300	csq	csq	csq
v25.05	1.4	4.2	3.4	oom	4.9	0.3	4.4	10.4	13.9	3.5	0.7	1.2	4.9	0.5	0.7	0.4	4.4	2.1	1.5	0.5	oom	4.1

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22
v25.05 global reordering	1.4	0.4	1.5	0.8	1.9	0.3	1.4	1.4	2.6	3.2	0.2	1.2	4.9	0.5	0.7	0.4	0.76	2.1	1.5	0.5	9.6	1.3

45.3x faster

Faster JOINs in ClickHouse (Open House) - https://www.youtube.com/watch?v=gd3OyQzB_Fc



ClickHouse와 연결하기

Connect with ClickHouse



slack
from Salesforce



GitHub



clickhou.se/workshop-level1-25nov25

가상 실시간 분석 실습 워크숍 -
레벨 1

Virtual Real-Time Analytics
hands-on workshop

2025년 11월 25일 오후 12시 30분
(KST)

November 25, 12:30 PM KST

Try ClickHouse for your use case

- ClickHouse Cloud
- Download open source

Learn

- Academy / certifications
- Blogs / YouTube

Engage with our community

- Community Slack
- Monthly Community calls
- Meetups / events

We are Hiring. Come Work with Us!



감사합니다.
Thank you.