
# Managing ClickHouse Dictionaries at Cloudflare

ClickHouse London Users Meetup
September 2023

CLOUDFLARE

# Intro

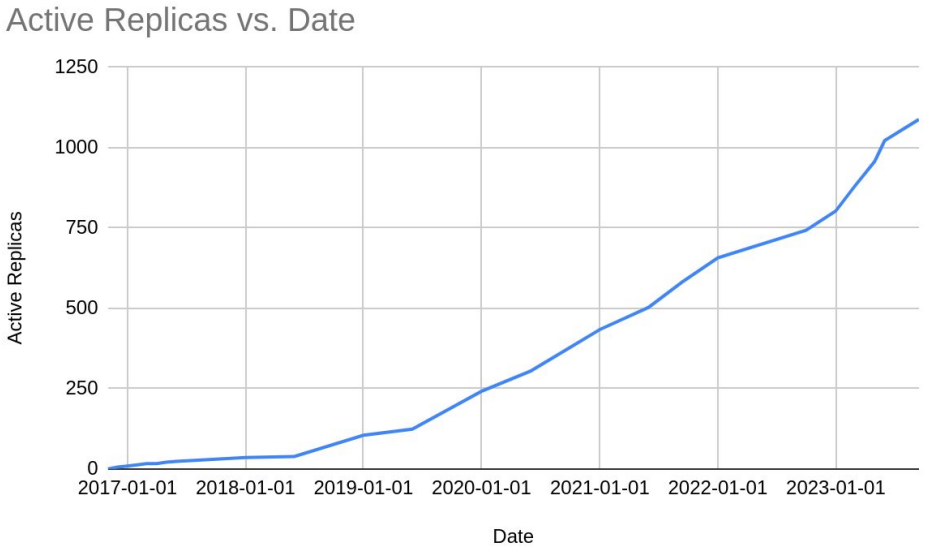
1. History & context
2. Problems we've faced
3. Future improvements


- James Morrison
- At Cloudflare for two years
- Part of our internal ClickHouse team
- One tiny upstream contribution (so far)
- jmorrison@cloudflare.com
- github.com/jawm

# Some history, some context

# ClickHouse at Cloudflare

- Yandex open sources June 2016
- We deploy to production before EoY
- Hundreds of millions of rows / sec
- Sampling → multiple events per row
- HTTP Requests, DNS Lookups, etc.
- Try yourself: Workers Analytics Engine



CLOUDFLARE

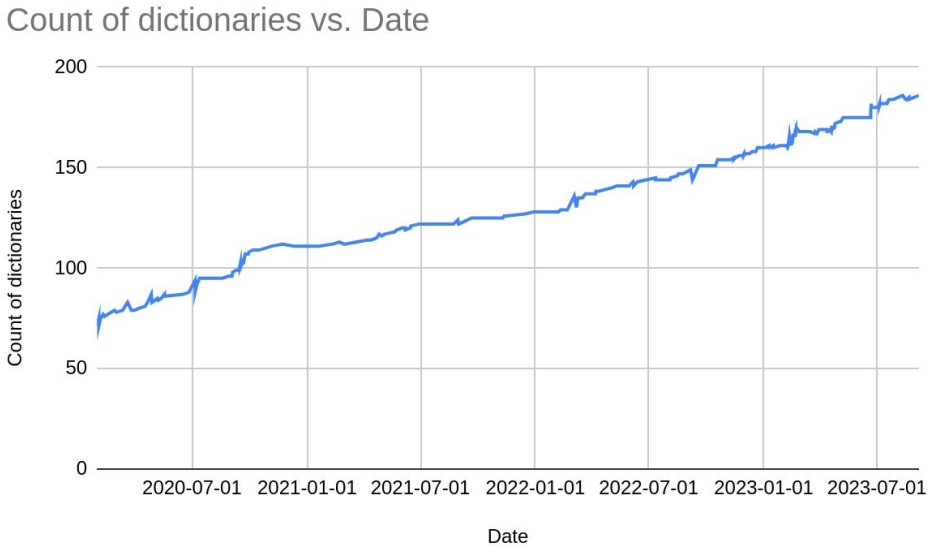
# What are dictionaries

- Allow key-value lookup in queries
- Data (typically) in memory
- Sourced from external systems
- Allows efficient JOIN-like behaviour
- For example, our “node” dictionary maps machine ID to hostname, location etc.
- *Many* configuration options

CLOUDFLARE

# We have a lot of dictionaries

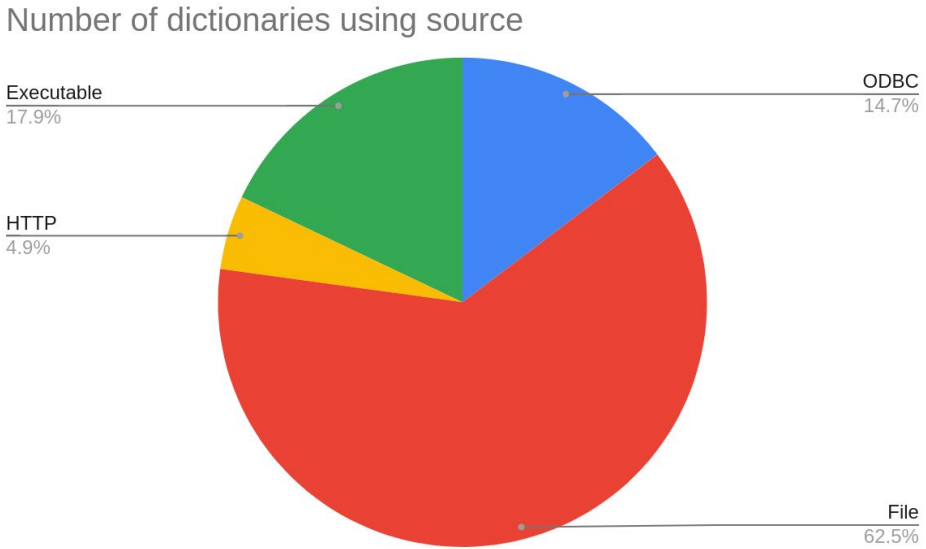
- 184 at the time of writing
- Growth seems to be linear
- Dictionaries contributed from people all across Cloudflare



CLOUDFLARE

# Dictionaries pull from many sources

- Breaks down into “static” and “dynamic”
- Static: File source – data in Git
- They’re small, typically <1k entries
- Dynamic: Various external sources, larger datasets
- They can be much larger, with millions of entries
- External sources are owned by other teams, or even other companies



CLOUDFLARE

# Many use cases

Roughly in order of importance:

- Products
- GraphQL
- INSERT path
- Grafana dashboards
- Internal analysts
- Some dictionaries are unused :(

# Some problems, some solutions

CLOUDFLARE

# Problem: Memory consumption

- Growth of many dictionaries is directly correlated with company growth. Exponential we hope!
- Our largest dictionary exceeded twenty gigs of memory!
- Less space for marks, page cache etc.
- Our largest dictionaries used the Hashed layout
- This uses one hashmap per attribute – lots of overhead

CLOUDFLARE

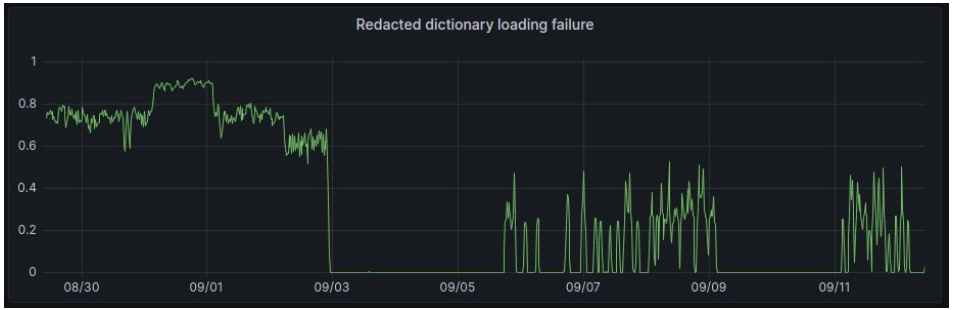
# Solution: Memory consumption

- Use HashedArrays layout instead of Hashed.
- We saw about 4x memory reduction on our largest dictionaries
- It has an array per attribute
- A single hashmap gives the index associated with each key in the arrays
- Memory savings correlated with number of attributes
- Extra pointer dereference, we don’t observe impact
- So, I recommend everyone to try switching

CLOUDFLARE

# Problem: Upstream could fail

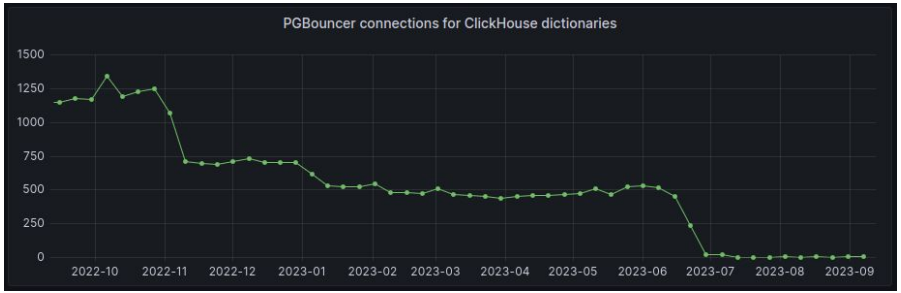
- We don’t own the external data sources
- They can fail for many reasons
- At best the ClickHouse replica can keep stale data in memory
- But if ClickHouse restarts, the dictionary will fail to load

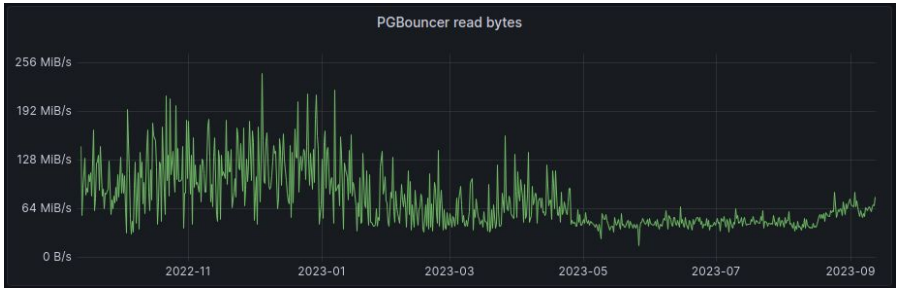


CLOUDFLARE

# Problem: More replicas → more load on upstream

- Applies to any remote source, such as ODBC or HTTP
- We had many issues where large dictionaries are being scraped by hundreds of ClickHouse replicas via ODBC
- It seems a bit silly that all the replicas hit the upstream to get the exact same data
- Caching results can help, but makes things more complicated, and doesn't always work



CLOUDFLARE

# Problem: We look after things we don’t understand

- With Executable dictionary, we have arbitrary code running on our servers.
- We didn’t write these scripts, we don’t understand them
- They are easy to break unless you know precisely what they depend on
- An embarrassing example; Python 2 goes away in Debian Bullseye
- Turns out some old dictionary required it

CLOUDFLARE

# Problem: ClickHouse sometimes has bugs

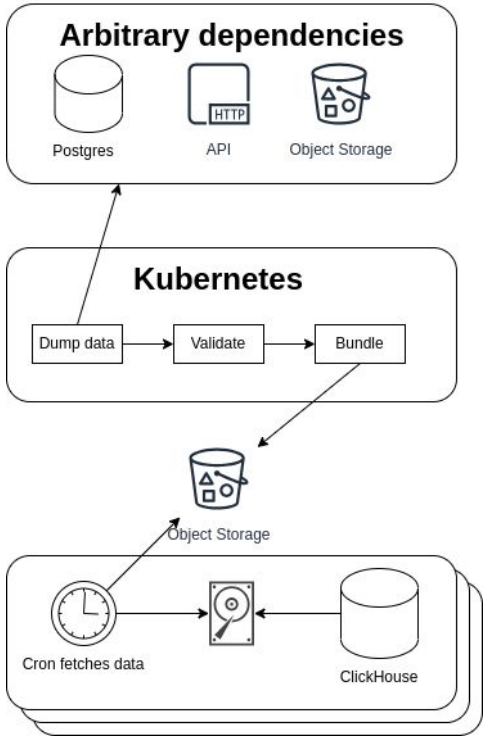
- We've occasionally observed incorrect behaviour
- For example, ClickHouse reporting dictionaries as loaded, when in fact they are not
- We try to submit patches for these issues when we can, but using the full available feature set means more surface area for bugs
- Overall though, the bugs are rare :)

| Pull request | Description |
|---|---|
| 48775 | Executable dictionaries showed the wrong loading status |
| 39385 | Direct layout dictionary had an incorrect optimization of dictHas function |

CLOUDFLARE

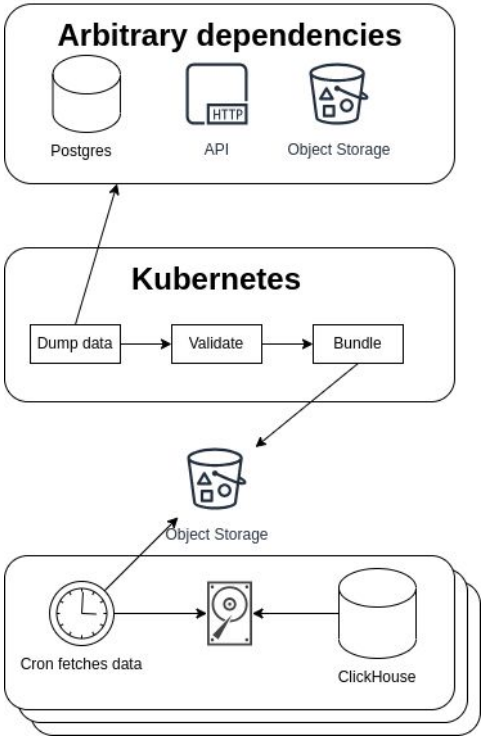
# Solution: ClickHouse Dictionary Dumper

- At a high level, we separate the production of data from the consumption.
- Data is dumped by a CronJob in Kubernetes
- If it is dumped and validated successfully, then we push it to Object Storage
- Asynchronously, timer on each replica pulls the bundle from Object Storage, extracts it onto disk
- ClickHouse replica always reads dictionary from files on disk, no external traffic



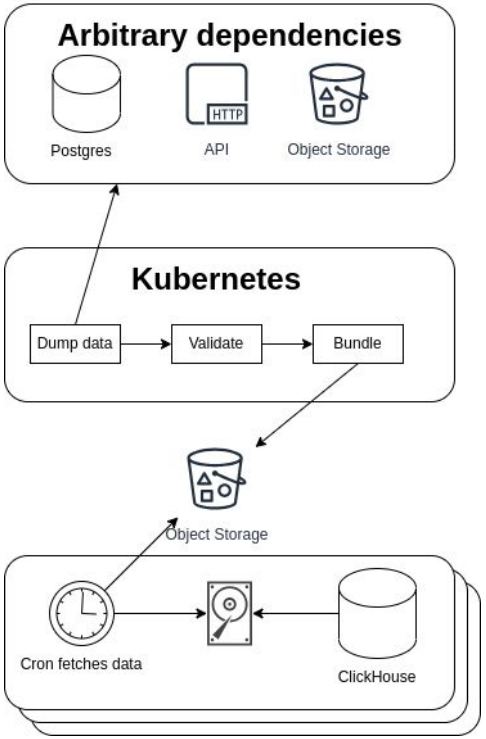
# Solution: Upstream failure

- If the upstream fails, the dumper will fail
- No new bundles are pushed to object storage until upstream becomes healthy
- Replicas will have stale data
- Replica restart is safe, data is saved on disk



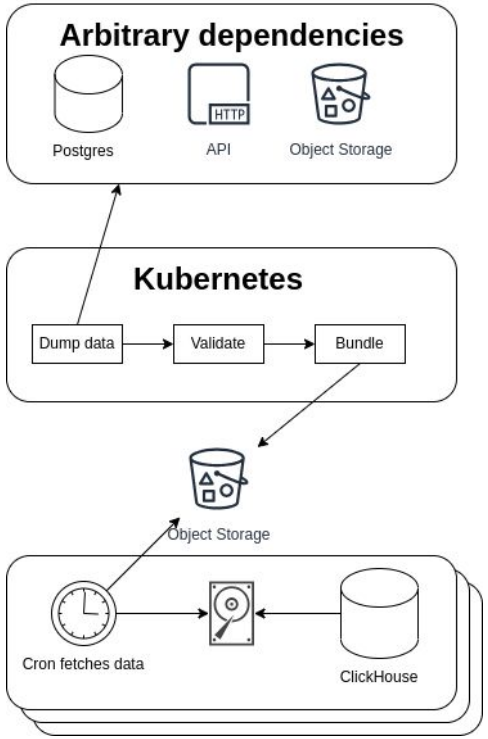
# Solution: Upstream overload

- We no longer have N ClickHouse replicas slamming external data source
- Only single pod in Kubernetes
- We do have N machines pulling from Object Storage
- This is ok, it's designed to run at massive scale
- If it ever *did* have an issue, each replica has stale copy on disk



# Solution: Looking after things we don't understand

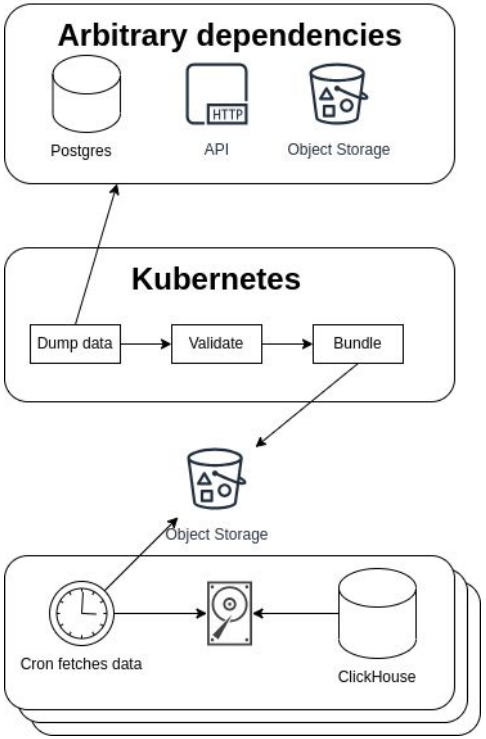
- We no longer look after other teams scripts
- They provide the image to run in Kubernetes
- They manage what their dependencies are, and when they choose to update them
- Clear division of ownership – we own bundling of data in Kubernetes, and everything on the replicas



CLOUDFLARE

# Solution: ClickHouse bugs

- Again, bugs are rare
- But, with every dictionary using the File source, less code paths exercised
- So we should reduce number of ClickHouse bugs encountered
- Maybe moves some bugs to the dumper step, but this is less critical



# Ideas & tasks yet not done

CLOUDFLARE

# Idea: More memory reductions

- We still store unbounded datasets in memory
- Compress data in-memory
- Extend CODEC support to dictionaries
- Store data on disk? Probably not
- Serve data over network? Probably not
- Spiky workloads with millions of lookups/sec

CLOUDFLARE

# Idea: Don’t use dictionaries?

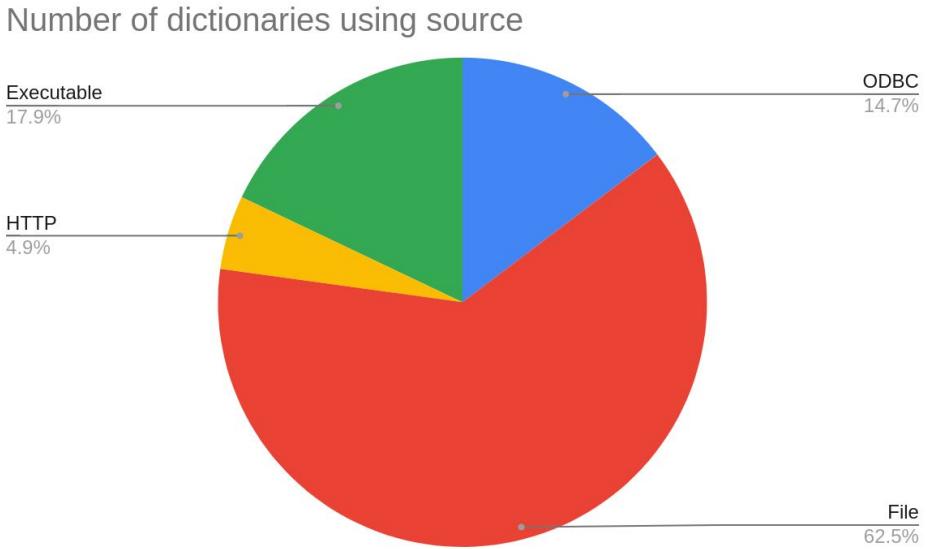
- Some dictionaries could be omitted
- LowCardinality(String) can (*sometimes*) be great for static data
- Enum column also an option
- Drop some rarely-used attributes
- Need more metrics!

```
0    UNKNOWN
1    in
2    out
```

CLOUDFLARE

# Idea: Static file dictionaries could sometimes be dynamic

- ⅔ of dictionaries are static
- Some change semi-regularly, requiring engineer to update
- They *could* be dynamic, but not always easy
- With clickhouse-dictionary-dumper, it becomes easier



# Questions?