OPSVERSE®

Switching Jaeger Distributed Tracing Storage to ClickHouse for Advanced Performance Monitoring (APM)

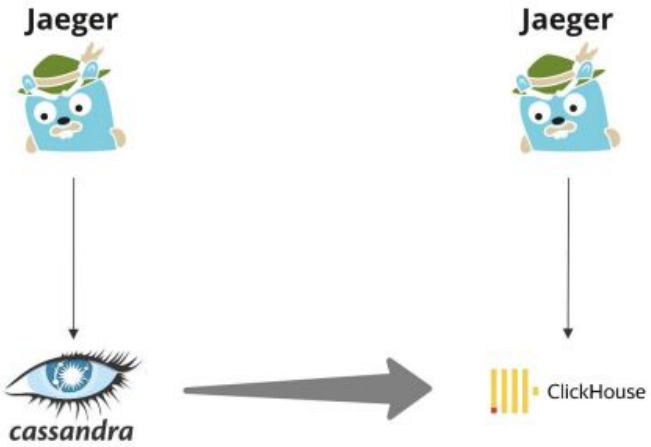
Satbir Chahal, OpsVerse

Open Source Analytics Conference 2022

Purpose

OPSVERSE®

Why and how OpsVerse migrated Jaeger storage to ClickHouse



What is Jaeger?

CLOUD NATIVE COMPUTING FOUNDATION

- A **distributed tracing** system open-sourced by Uber in 2015, now part of CNCF
- Instrumented apps achieve **visibility** of a request's traversal of the system by propagating **trace context** between services
- Each of these operations (e.g., svc-to-svc request) is known as a **span** - a collection of spans compose a trace

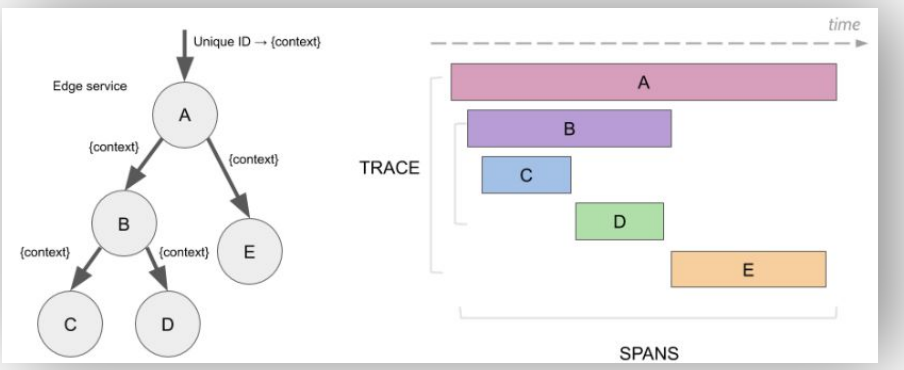
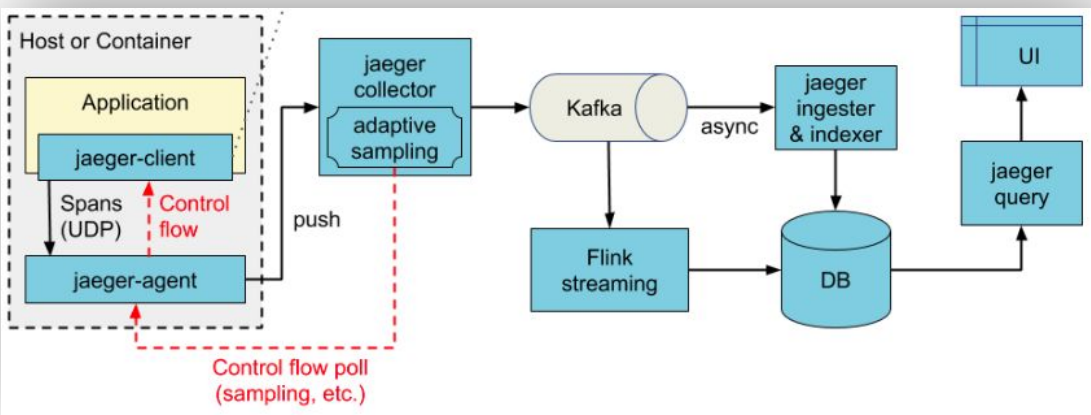


Image courtesy of jaegertracing.io

Jaeger Architecture



** Image courtesy of jaegertracing.io*

*** In our case, jaeger-clients are replaced by OpenTelemetry; an OpenTelemetry Collector is between the Application and the Jaeger Collector.*

- Jaeger has native support for **Elasticsearch** and **Cassandra** as storage
- Later, **gRPC** storage plugin support was added (more on this soon)
- Leaned toward Cassandra due to **operational overhead** of maintaining ES clusters (resources and manual TTLs)
- Cassandra was okay... until we wanted to **do more**

Cassandra

OPSVERSE®

Pros

- Scalable with near-zero maintenance
- Fast writes - Kafka consumer group lags were also near-zero
- General advantages (not pertaining to traces per se):
  - Fault tolerance
  - Replication

Cons

- No JOINs
  - Services index one table
  - Operations index in one table
- Bulk querying load can be expensive
- Minimal aggregate functions

So advanced analytics on the data will require a custom process

Why ClickHouse (Data)

OPSVERSE®

- Great data compression and batch ingestion
- Some existing familiarity:
  - Our team was exposed to ClickHouse when we ran another open source tool using it as a storage backend (PostHog)
  - Near SQL compatibility (for querying)
- Rich set of functions:
  - Aggregate (topk, stdev, etc)
  - JSON, String, Array, Maps, etc
- Its weaknesses are tolerable for distributed tracing:
  - Data consistency isn't guaranteed
  - Delete/updates are slow (batch operations)

Why ClickHouse (Kubernetes)

OPSVERSE®

Some key pre-requisites met:

- Jaeger adds support for gRPC storage plugins
  - Paves way for community to support (virtually) any database
  - Plugin implements the storage gRPC protobuf interfaces
- Community releases https://github.com/jaegertracing/jaeger-clickhouse/ (mid 2021)

So... we need to run a ClickHouse instance (to connect plugin):

- The Kubernetes Operator for ClickHouse (by Altinity) grows in popularity https://github.com/Altinity/clickhouse-operator
- Plenty of community content (blogs, videos, webinars, meetups) allow team to run and PoC quickly
- Becomes the preferred method of running ClickHouse on Kubernetes

The Steps

OPSVERSE®

- Swap out Bitnami Cassandra helm chart for the K8s ClickHouse Operator (CHOP)
- Add a `ClickHouseInstallation` CR template (for CHOP to reconcile) to stack deployment charts
- Make sure the ClickHouse gRPC plugin for Jaeger binary makes it onto the container images that connect to the database:
  - Jaeger Ingester
  - Jaeger Querier
- Update Jaeger helm chart to specify gRPC storage rather than Cassandra

```
extraConfigmapMounts:
  - name: plugin-config
    mountPath: /plugin-config
    configMap: acme-phi-observe-backend-jaeger-clickhouse
```

```
FROM ghcr.io/jaegertracing/jaeger-clickhouse:0.7.0 AS grpc-plugin

FROM registry.devopsnow.io/public/jaegertracing/jaeger-ingester:1.36.0

COPY --from=grpc-plugin /go/bin/jaeger-clickhouse /go/bin/jaeger-clickhouse
```

```
storage:
  type: grpc-plugin
  grpcPlugin:
    extraEnv:
      - name: GRPC_STORAGE_PLUGIN_BINARY
        value: /go/bin/jaeger-clickhouse
      - name: GRPC_STORAGE_PLUGIN_CONFIGURATION_FILE
        value: /plugin-config/config.yaml
```

Some Gotchas (and Best Practices)

OPSVERSE®

- Get familiar with `ClickHouseInstallation` Custom Resource spec:
  - It will create a `LoadBalancer` Service (possibly public) by default (update CR spec `templates.serviceTemplates`)
  - No “readonly” user by default (update CR spec `configuration.users`)
  - Similar for default user (limit network CIDRs), retention periods, disk space and resource request/limits

Some Gotchas (and Best Practices) … cont'd

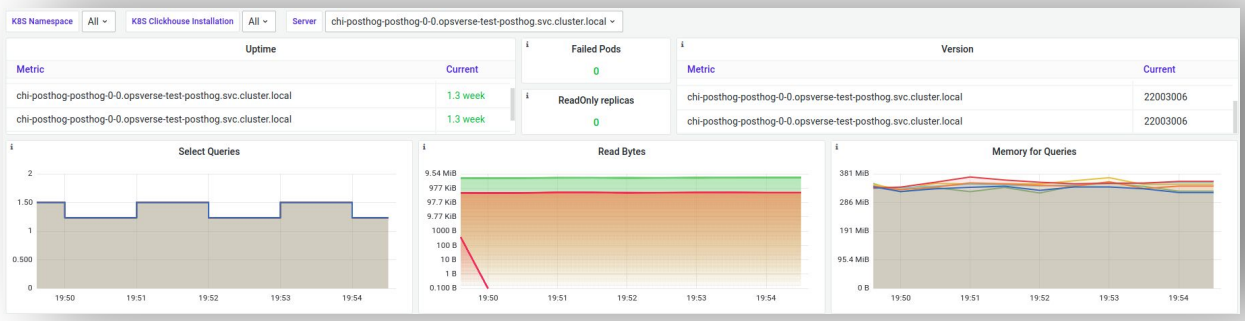
OPSVERSE®

- Resource management conflicts with GitOps tools (e.g., ArgoCD)
  - ArgoCD may think it is managing a CHOP-managed resource (like a PVC)
  - This can lead to unintended `Terminating` state of resource
  - Update ClickHouseInstallation template metadata annotations with:
    - `argocd.argoproj.io/compare-options: IgnoreExtraneous`
    - `argocd.argoproj.io/sync-options: Prune=false`
  - Also, can set CHOP config to exclude the label propagation your specific tool depends on

Sleep Easy at Night

OPSVERSE®

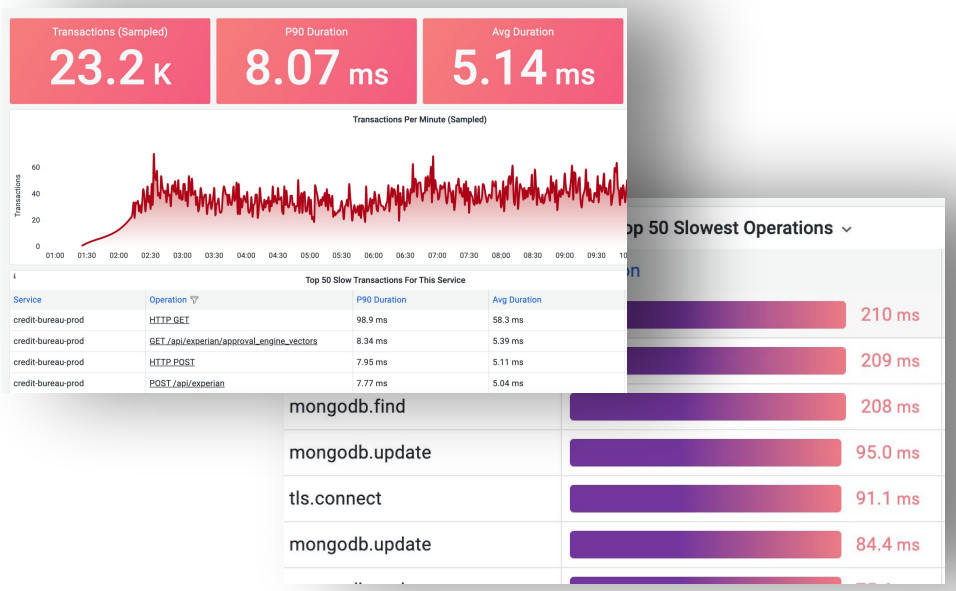
- Enable Backups
  - https://github.com/AlexAkulov/clickhouse-backup as a K8s `CronJob`
  - [In Progress] `ClickHouseBackup` as a CR: https://github.com/Altinity/clickhouse-operator/issues/862
- Enable metrics and alerts:
  - `/metrics` endpoint serves by default (have Prometheus scrape it)



Begin Querying

OPSVERSE®

- Add the ClickHouse instance as a data source to your favorite data visualization tool (Grafana, Apache Superset, etc)
- Rich functions and parsers allow for some query magic
- Use Materialized Columns (or Views) to create additional columns from existing data - for even faster querying




```
SELECT materialized_operation_name AS operation, materialized_duration AS duration, traceID
FROM jaeger_spans_local
WHERE materialized_references != 'null'
  AND materialized_service_name = '$service_name'
ORDER BY toInt32(JSON_VALUE(model, '$.duration')) DESC
```

OpsVerse APM Using Jaeger+ClickHouse

OPSVERSE®

Thank you!

@Satbir Chahal | @OpsVerse
Community Slack tiny.cc/slck
https://opsverse.io