



Databend

# Tracing in Databend

Tracing 赋能 Databend 的可观测性

Databend 贡献者养成计划

尚卓燃 (PsiACE)



# 目录

## CONTENT

- ① Databend 与 Tracing
- ② Jaeger 分布式跟踪
- ③ 探迹 tokio-console
- ④ 还能做什么



WWW.DATABEND.COM

# Databend 与 Tracing

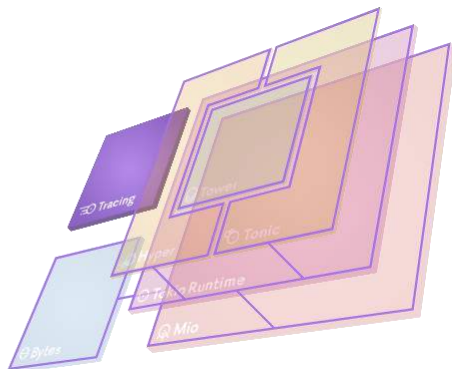
初步了解 Databend 中的跟踪系统

Databend 贡献者养成计划



# 什么是 Tracing

WWW.DATABEND.COM



Tracing 由 Tokio 项目维护，无需 Tokio 运行时  
检测 Rust 程序来收集结构化的、基于事件的诊断信息

<https://github.com/tokio-rs/tracing>

```
use tracing::{info, Level};
use tracing_subscriber;

fn main() {
    let collector = tracing_subscriber::fmt()
        // filter spans/events with level TRACE or higher.
        .with_max_level(Level::TRACE)
        // build but do not install the subscriber.
        .finish();

    tracing::collect::with_default(collector, || {
        info!("This will be logged to stdout");
    });
    info!("This will _not_ be logged to stdout");
}
```



Databend 的 tracing-subscriber 被统一整合在 common/tracing，由 query 和 meta 共同。

```
// Use env RUST_LOG to initialize log if present.
// Otherwise use the specified level.
let directives = env::var(EnvFilter::DEFAULT_ENV).unwrap_or_else(|_x| level.to_string());
let env_filter = EnvFilter::new(directives);
let subscriber = Registry::default()
    .with(env_filter)
    .with(JsonStorageLayer)
    .with(stdout_logging_layer)
    .with(file_logging_layer)
    .with(jaeger_layer);

#[cfg(feature = "console")]
let subscriber = subscriber.with(console_subscriber::spawn());
```

利用 tracing-bunyan-formatter 进行处理

存储在 \_logs 目录下

opentelemetry-jaeger

tokio console 支持



具体到内部的 tracing ，大致有两种

```
// 普通：就像使用 log 一样，利用 info!、debug! 来收集信息

use common_tracing::tracing;

tracing::info!("{:?}", conf);
tracing::info!(
    "DatabendQuery v-{}",
    *databend_query::configs::DATABEND_COMMIT_VERSION,
);
```

```
// Instruments: 在调用函数时创建并进入 tracing span (跨度)
// span 是表示程序在特定上下文中执行的时间段

use common_tracing::tracing::debug_span;

#[tracing::instrument(level = "debug", skip_all)]
pub async fn read(&mut self) -> Result<DataBlock> {
    ...

    debug_span!("block_reader_read_column").or_current()

    ...
}
```

```
{"v":0,"name":"databend-query-test_cluster@0.0.0.0:3307","msg":"db name: db1,
engine: ","level":50,"hostname":"dataslime","pid":127203,"time":"2022-01-
06T04:14:57.4974979Z","target":"databend_query::catalogs::impls::mutable_catalog",
"line":154,"file":"query/src/catalogs/impls/mutable_catalog.rs"}
```



# “亲密接触”的小 Tips

WWW.DATABEND.COM

Databend 与 Tracing “亲密接触”的几种方式：

1. http tracing (不推荐)：访问 `localhost:8080/v1/logs`
2. stdout/filelog：检查终端输出或 `_logs` 目录（根据配置）
3. system.tracing 表：执行 `select * from system.tracing limit 20;`
4. jaeger：运行 jaeger，访问 `http://127.0.0.1:16686/`
5. console：按特定方式构建后，运行 `tokio-console`



WWW.DATABEND.COM

# Jaeger 分布式跟踪

使用 Jaeger 探索 Databend 运行原理

Databend 贡献者养成计划





# Jaeger & OpenTelemetry

WWW.DATABEND.COM

Jaeger 是一个开源的端到端分布式跟踪系统。由 Uber 捐赠给 CNCF。它可以用于监视基于微服务的分布式系统：

- 分布式上下文传播
- 分布式事务监视
- 根本原因分析
- 服务依赖性分析
- 性能/延迟优化



OpenTelemetry 是工具、API 和 SDK 的集合。使用它来检测、生成、收集和导出遥测数据（度量、日志和跟踪），以帮助分析软件的性能和行为。





# Step by Step

WWW.DATABEND.COM

---

## 1. 构建二进制程式

`cargo build` (可以使用 `--bin` 指定)

## 2. 将日志级别设定为 `DEBUG`，运行需要调试的应用程式

例如，`LOG_LEVEL=DEBUG ./databend-query`

## 3. 运行 `jaeger`

`docker run -d -p6831:6831/udp -p6832:6832/udp -p16686:16686  
jaegertracing/all-in-one:latest`

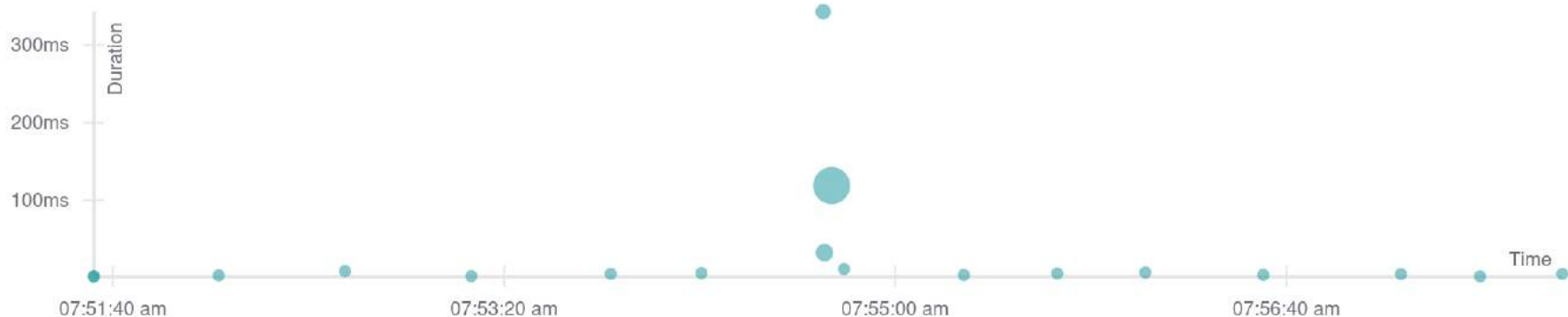
## 3. 打开 `http://127.0.0.1:16686/` 以查看 `jaeger` 收集的信息

执行

```
CREATE TABLE t1(a INT);  
INSERT INTO t1 VALUES(1);  
INSERT INTO t1 SELECT * FROM t1;
```

可以得到下图所示跟踪结果

x轴是执行时刻，y轴是持续的时间，圆点反映 span 的聚集程度





# 结果探析 - Timeline

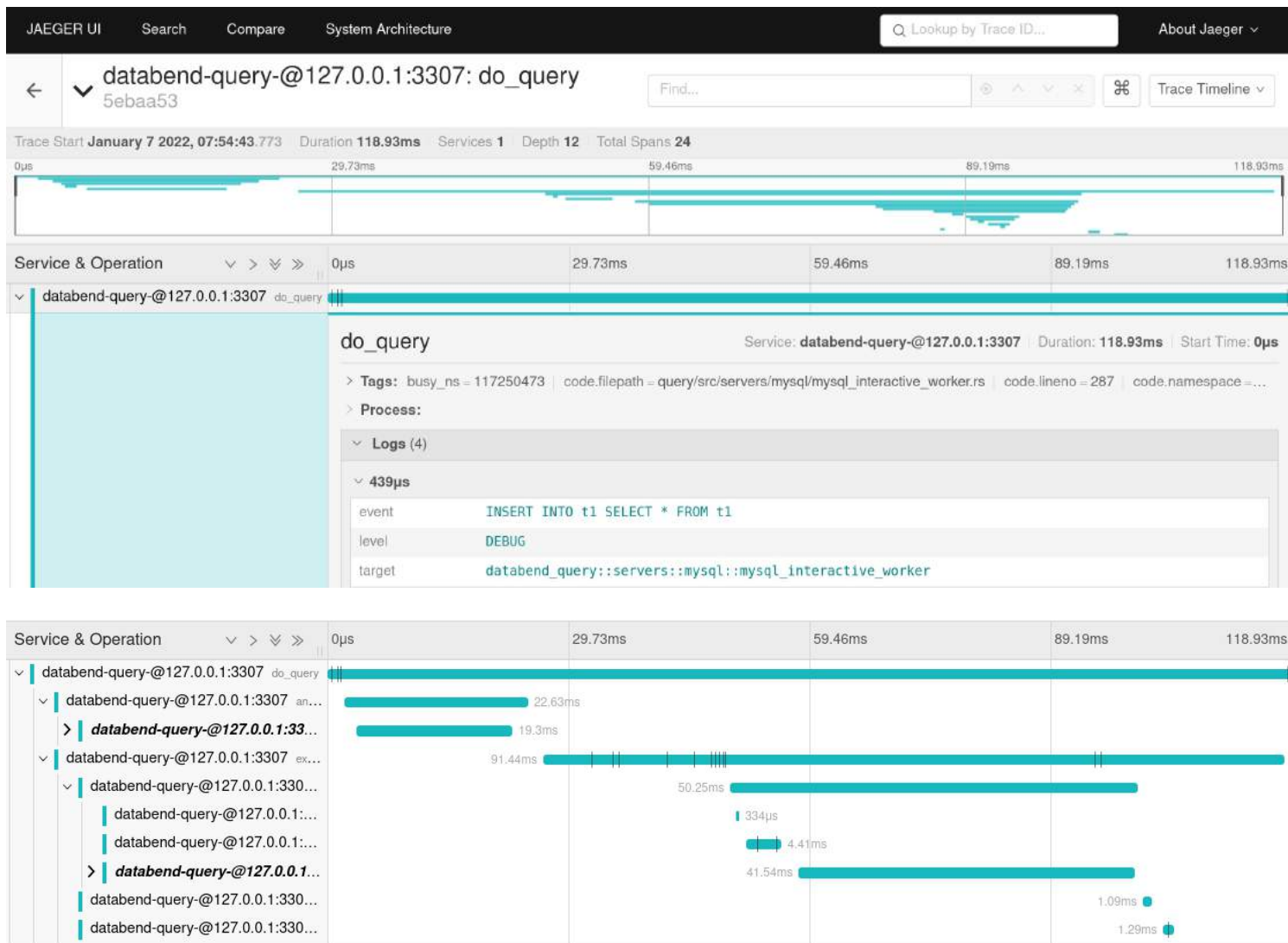
WWW.DATABEND.COM

左图是点击最大的圆点得到的跟踪情况：

使用 timeline 模式来展现 tracing 的各个跨度之间的关系。

以时间为主线进行分析,方便使用者观看在某个时间点观看程序信息

点开第一个跨度，可以看到这是 INSERT INTO t1 SELECT \* FROM t1 查询时的情况





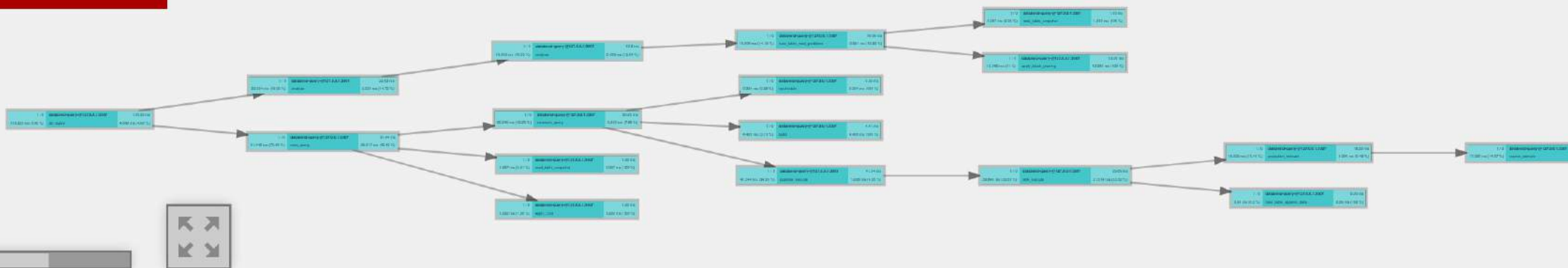
## 结果探析 - Graph

WWW.DATABEND.COM

切换到 graph 模式，可以看到各个 span 之间的调用链，每个 span 具体用时，以及百分比  
通过这个视图使用者很容易知道系统瓶颈，快速定位问题。

1 / 0	databend-query-@127.0.0.1:3307	19.3 ms
19.303 ms (16.23 %)	analyze	2.439 ms (12.64 %)

### Experimental





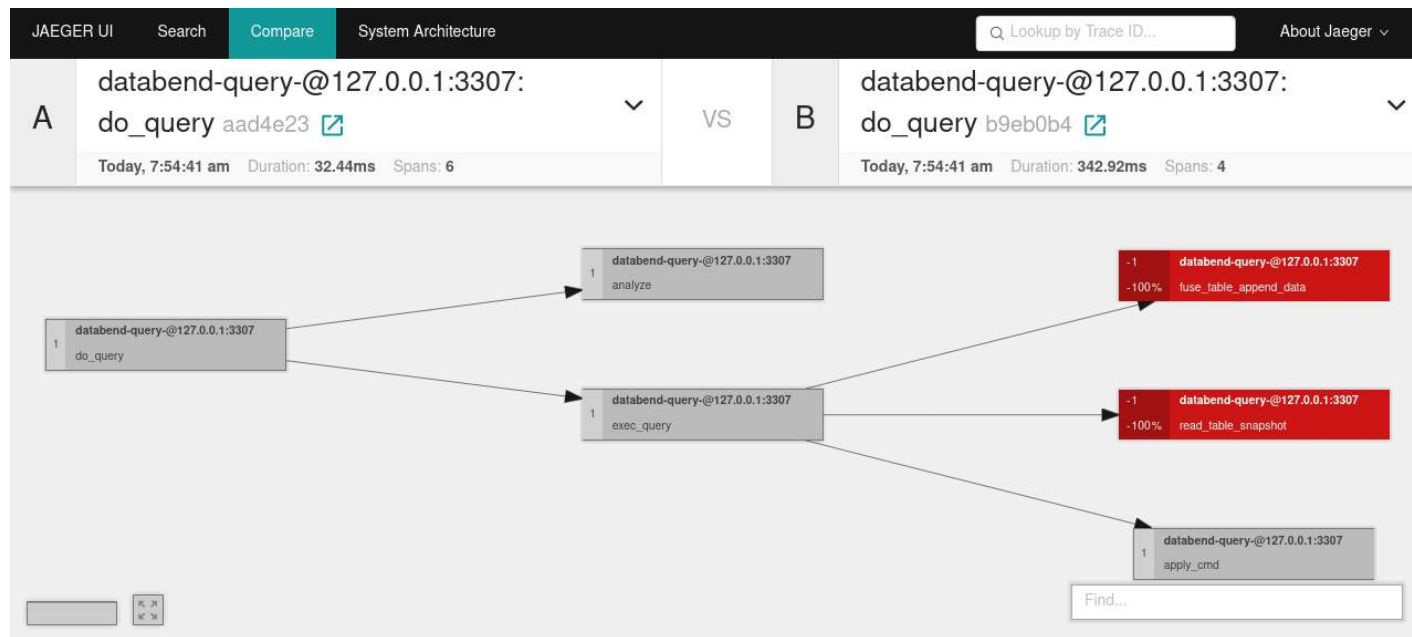
## 结果探析 - Compare

WWW.DATABEND.COM

连起来的各个部分形成整个 trace 的调用链。

因为比较时一般会比较两个相同类型的调用，所以看到的是重合后的视图。

- 深绿色，表示这个 span 只存在于 trace-B 中，A 没有这个 span
- 深红色，表示这个 span 只存在于 trace-A 中，B 没有这个 span
- 浅绿色，表示这个 span 在 trace-B (右边这个) 的数量多
- 浅红色，表示这个 span 在 trace-A (左边这个) 的数量多





WWW.DATABEND.COM

# 探迹 **tokio-console**

**tokio-rs** 团队出品的诊断和调试工具

Databend 贡献者养成计划



tokio-console 是专为异步程序设计的调试与诊断工具，能够列出 tokio 的任务，提供对程序的任务和资源的实时、易于导航的视图，总结了它们的当前状态和历史行为。

- <https://github.com/tokio-rs/console>
- 传输诊断数据的协议
- 用于收集诊断数据的仪器
- 当然，用于展示和探索诊断数据的实用工具

```
tokio-console 0.1.0
```

```
Eliza Weisman <eliza@buoyant.io>, Tokio Contributors <team@tokio.rs>
```

```
The Tokio console: a debugger for async Rust.
```

```
USAGE:
```

```
tokio-console [OPTIONS] [TARGET_ADDR]
```





1. 使用特定的 rustflags 和 features 来构建：  
`RUSTFLAGS="--cfg tokio_unstable" cargo build --features tokio-console`  
也可以只构建单个二进制程式，使用 `--bin` 进行指定
2. 将日志级别设定为 TRACE ，运行需要调试的应用程式  
`LOG_LEVEL=TRACE databend-query`  
或者  
`databend-meta --single --log-level=TRACE`  
目前，因为潜在的端口抢占问题，一次只能对单个程序进行调试，所以只要待调试应用处于 TRACE 日志级别即可（当然，可以使用 `TOKIO_CONSOLE_BIND` 分别指定端口）
3. 运行 `tokio-console` # default connection: `http://127.0.0.1:6669`



## 结果探析 - 任务

WWW.DATABEND.COM

先看什么是 tokio 任务

1. 任务是一个轻量级的、非阻塞的执行单元。类似操作系统的线程，但是是由 tokio 运行时管理，一般叫做“绿色线程”，与 Go 的 goroutine，Kotlin 的 coroutine 类似。

2. 任务是协同调度的。大多数操作系统实现抢占式多任务。操作系统允许每个线程运行一段时间，然后抢占它，暂停该线程并切换到另一个线程。另一方面，任务实现协同多任务。一个任务被允许运行直到它让出执行权，运行时切换到执行下一个任务。

connection: http://127.0.0.1:6669/ (CONNECTED)

views: t = tasks, r = resources

controls:  $\leftrightarrow$  = select column (sort),  $\updownarrow$  = scroll,  $\leftarrow$  = view details, i = invert sort (highest/lowest), q = quit

### Warnings

⚠️ 1 tasks have woken themselves over 50% of the time

⚠️ 3 tasks have lost their waker

Tasks (17) ▶ Running (3) || Idle (14)

Warn	ID	State	Name	Total	Busy	Idle	Polls	Target	Location
⚠️ 1	2			97.0531s	946.5460μs	97.0522s	1	tokio::task	<cargo>/console-subscriber-0.1.
⚠️ 1	3	▶	console::aggregate	97.0524s	24.3728s	72.6796s	166	tokio::task	<cargo>/console-subscriber-0.1.
	4		console::serve	97.0521s	4.0682ms	97.0480s	2	tokio::task	<cargo>/console-subscriber-0.1.
	16	▶		97.0305s	97.0299s	584.5170μs	1	tokio::task::blocking	<unknown location>
	19			97.0259s	11.6837ms	97.0142s	3	tokio::task	query/src/servers/mysql/mysql_h
⚠️ 1	20			97.0246s	1.0523ms	97.0236s	1	tokio::task	common/base/src/runtime.rs:89:4
	23	▶		97.0197s	97.0178s	1.8552ms	1	tokio::task::blocking	<unknown location>
	26			97.0135s	507.4020μs	97.0130s	1	tokio::task	query/src/servers/clickhouse/cl
⚠️ 1	27			97.0128s	409.0280μs	97.0124s	1	tokio::task	common/base/src/runtime.rs:89:4
	30			97.0021s	6.1143ms	96.9960s	1	tokio::task	query/src/common/service/http_s
	33			96.9882s	2.1969ms	96.9860s	1	tokio::task	query/src/common/service/http_s



通过左右切换，可以得到总忙时间或轮询次数等指标对任务进行排序。控制台通过高亮来提示较大差异，比如从毫秒到秒的切换。

控制台还实现了一个“警告”系统。通过监视应用程序中任务的运行时操作，控制台可以检测可能提示 bug 或性能问题的行为模式，并突出显示这些行为模式供用户分析。比如已经运行了很长时间而没有让步的任务，唤醒的次数比其他任务唤醒的次数还要多的任务。

```
connection: http://127.0.0.1:6669/ (CONNECTED)
views: t = tasks, r = resources
controls: ←→= select column (sort), ↑↓= scroll, ⇐= view details, i = invert sort (highest/lowest), q = quit

Warnings
⚠ 3 tasks have lost their waker

Tasks (17) ▶ Running (3) || Idle (14)
>>

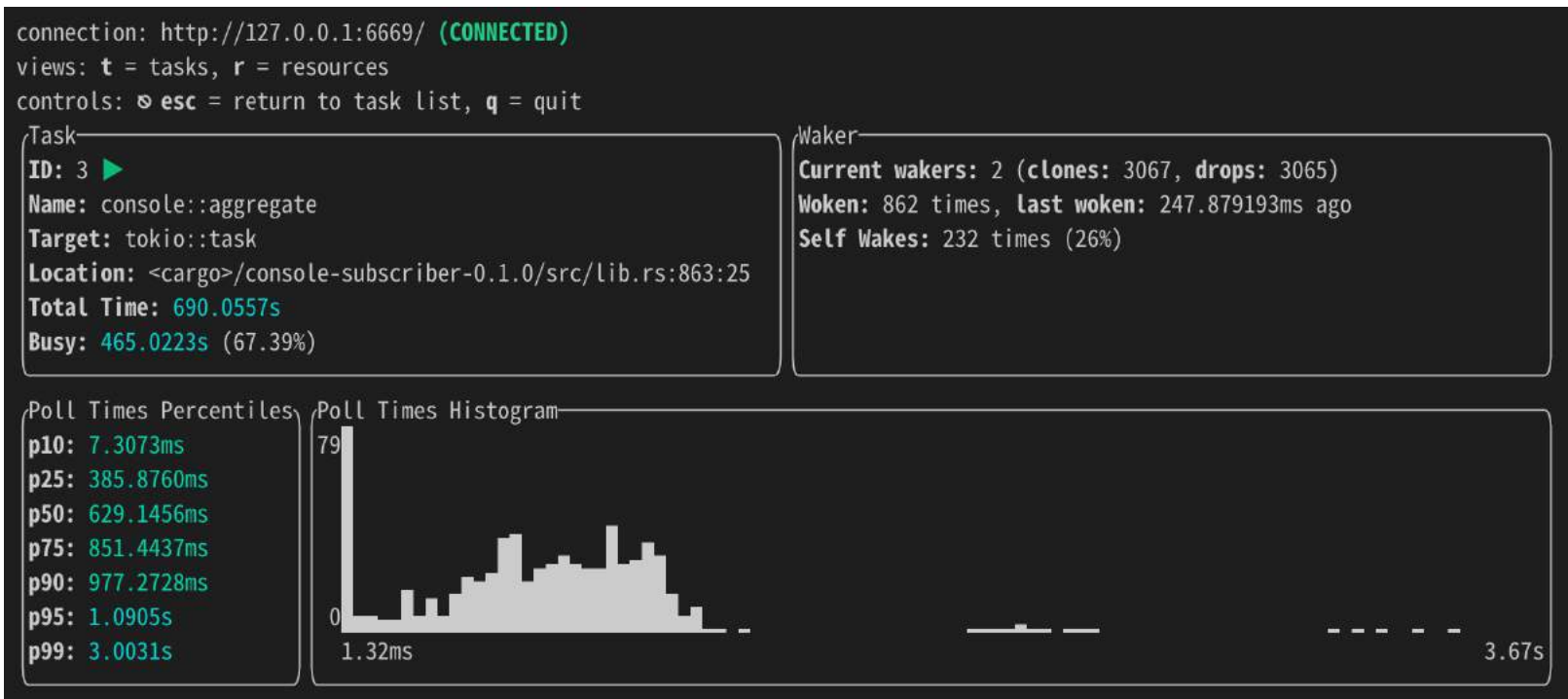

| Warn ID | State | Name               | Total     | Busy       | Idle       | Polls | Target                | Location                     |
|---------|-------|--------------------|-----------|------------|------------|-------|-----------------------|------------------------------|
| 16      | ▶     |                    | 517.0246s | 517.0240s  | 584.5170µs | 1     | tokio::task::blocking | <unknown location>           |
| 23      | ▶     |                    | 517.0138s | 517.0119s  | 1.8552ms   | 1     | tokio::task::blocking | <unknown location>           |
| 3       | ▶     | console::aggregate | 517.0464s | 301.9041s  | 215.1423s  | 586   | tokio::task           | <cargo>/console-subscriber-0 |
| 62      |       |                    | 508.9300s | 12.2044s   | 496.7256s  | 512   | tokio::task           | <cargo>/hyper-0.14.16/src/co |
| 63      |       |                    | 508.8538s | 11.6209s   | 497.2329s  | 1019  | tokio::task           | <cargo>/hyper-0.14.16/src/co |
| 41      |       |                    | 516.9451s | 593.5984ms | 516.3515s  | 18    | tokio::task           | query/src/clusters/cluster.r |
| 19      |       |                    | 517.0200s | 11.6837ms  | 517.0083s  | 3     | tokio::task           | query/src/servers/mysql/mysq |
| 30      |       |                    | 516.9962s | 6.1143ms   | 516.9901s  | 1     | tokio::task           | query/src/common/service/htt |
| 4       |       | console::serve     | 517.0462s | 4.0682ms   | 517.0421s  | 2     | tokio::task           | <cargo>/console-subscriber-0 |
| 33      |       |                    | 516.9823s | 2.1969ms   | 516.9801s  | 1     | tokio::task           | query/src/common/service/htt |
| 36      |       |                    | 516.9676s | 2.0650ms   | 516.9655s  | 1     | tokio::task           | query/src/common/service/htt |
| 37      |       |                    | 516.9664s | 1.2735ms   | 516.9651s  | 1     | tokio::task           | query/src/api/rpc_service.rs |


```



上下切换选中任务，enter 查看关于每个任务的翔实数据，比如轮询持续时间的直方图

不仅列出任务。  
console 还会插入异步  
互斥和信号量等资源。  
Tokio Console 的资源  
详细信息视图显示了哪  
些任务已经进入临界区，  
哪些任务正在等待获得  
访问权限。





WWW.DATABEND.COM

# 还能做什么

与分布式跟踪和日志系统相关的一些其他内容

Databend 贡献者养成计划



目前还有一系列关于可观测性和 Tracing 的 Issue:

- Configure different console ports for query and meta #3758
- Configure on jaeger tracing address similar to metrics api server #3633
- Support dashboard api to integrate with observability plugins such as prometheus and grafana #2346
- Summary of todos about distributed tracing #1227
- Query traces and analysis, based on user behavior. #1177
- Http stack traces #1085
- Shuffle read/write metrics #1004

另外，更进一步的考量是，如何基于可观测性来自动/半自动地发现问题并对系统进行调优



了解更多...

WWW.DATABEND.COM

Tracing:

- <https://github.com/tokio-rs/tracing>
- <https://docs.rs/tracing/latest/tracing/>
- <https://tokio.rs/blog/2019-08-tracing>

Jaeger:

- <https://github.com/open-telemetry/opentelemetry-rust/tree/main/opentelemetry-jaeger>
- <https://21-lessons.com/ship-rust-tracing-spans-to-jaeger/>

tokio-console:

- <https://github.com/tokio-rs/console>
- <https://hackmd.io/@aws-rust-platform/ByChcdB-t>
- <https://tokio.rs/blog/2021-12-announcing-tokio-console>





Databend

# 讨论时间

感谢您的观看，让我们一起聊聊

Databend 贡献者养成计划

尚卓燃 (PsiACE)