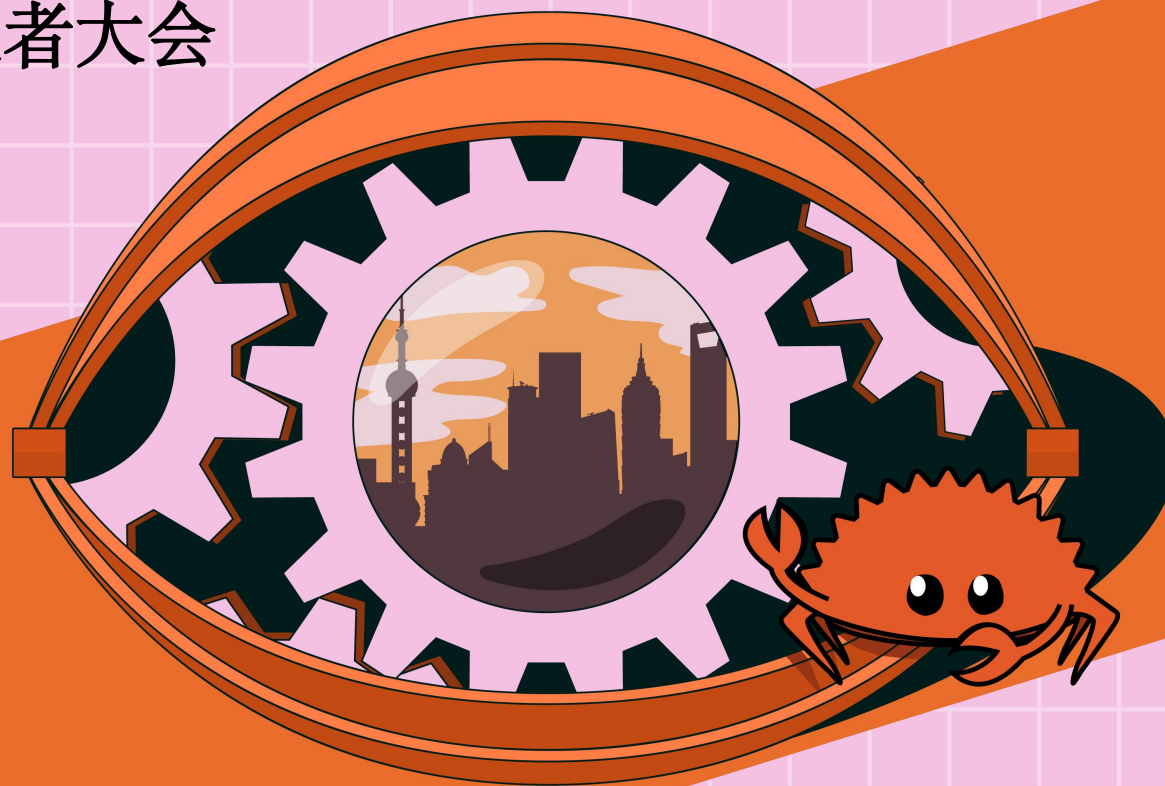


RUST CHINA CONE 2023

第三届中国Rust开发者大会



6.17-6.18 @Shanghai

Rust China Conf 2023

CnosDB时序数据库的Rust实践

Yongtao Liu

CnosDB 研发工程师





CnosDB 是一款基于 Rust 开发的
开源的分布式时序数据库



Community Edition

Free, Open Source, Eco-Friendly

- ✓ A fully open source product based on the Rust language, integrating with the existing time-series ecosystem
- ✓ Use distributed clusters for free with no functional limitations
- ✓ Rapid product iteration prioritizes ultimate product features
- ✓ Supported by a wide range of open source community users and developer communities

[View Source Code](#)

[Download](#)



Enterprise Edition

Private Cloud, Expert Support

- ✓ Cloud-native friendly, supports various server and container environments
- ✓ High-performance, high-availability distributed clusters with customizable management and operation tools
- ✓ Product fault support with up to 7x24 response time
- ✓ Flexible pricing model, low-cost access to Enterprise Edition

[Contact Us](#)



CnosDB Cloud

Serverless, Out-Of-The-Box

- ✓ Cloud-native serverless, fully leveraging the convenience of cloud infrastructure and integrating seamlessly into cloud-native ecosystems
- ✓ Out-of-the-box, elastic scaling support, supporting bidirectional resource expansion of storage and computing
- ✓ Native multi-tenant and pay-as-you-go models for lower costs
- ✓ Liberate operations engineers from heavy workloads and easily manage cloud services
- ✓ Integrated with cloud-native OLAP/CloudAI data ecosystem

[Learn More](#)



CnosDB Embedded

A Collaborative Embedded Time-Series Database For Cloud And Edge

- ✓ Ultra-light kernel for embedded devices
- ✓ Supports deployment on ARM/Raspberry Pi and other edge architectures
- ✓ Truly achieves a cloud-edge integrated data model
- ✓ Collaboration between cloud and edge with multi-level storage for cost control

[Contact Us](#)



1. CnosDB 架构与选型
2. 为何从 Go 切换到 Rust
3. 使用 Rust 经验分享
4. 反哺社区



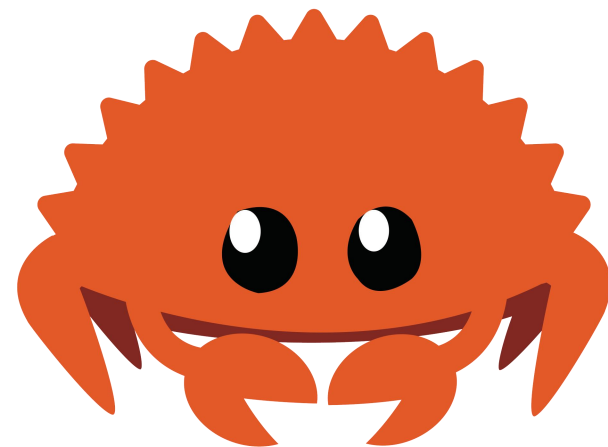
1. CnosDB 架构与选型



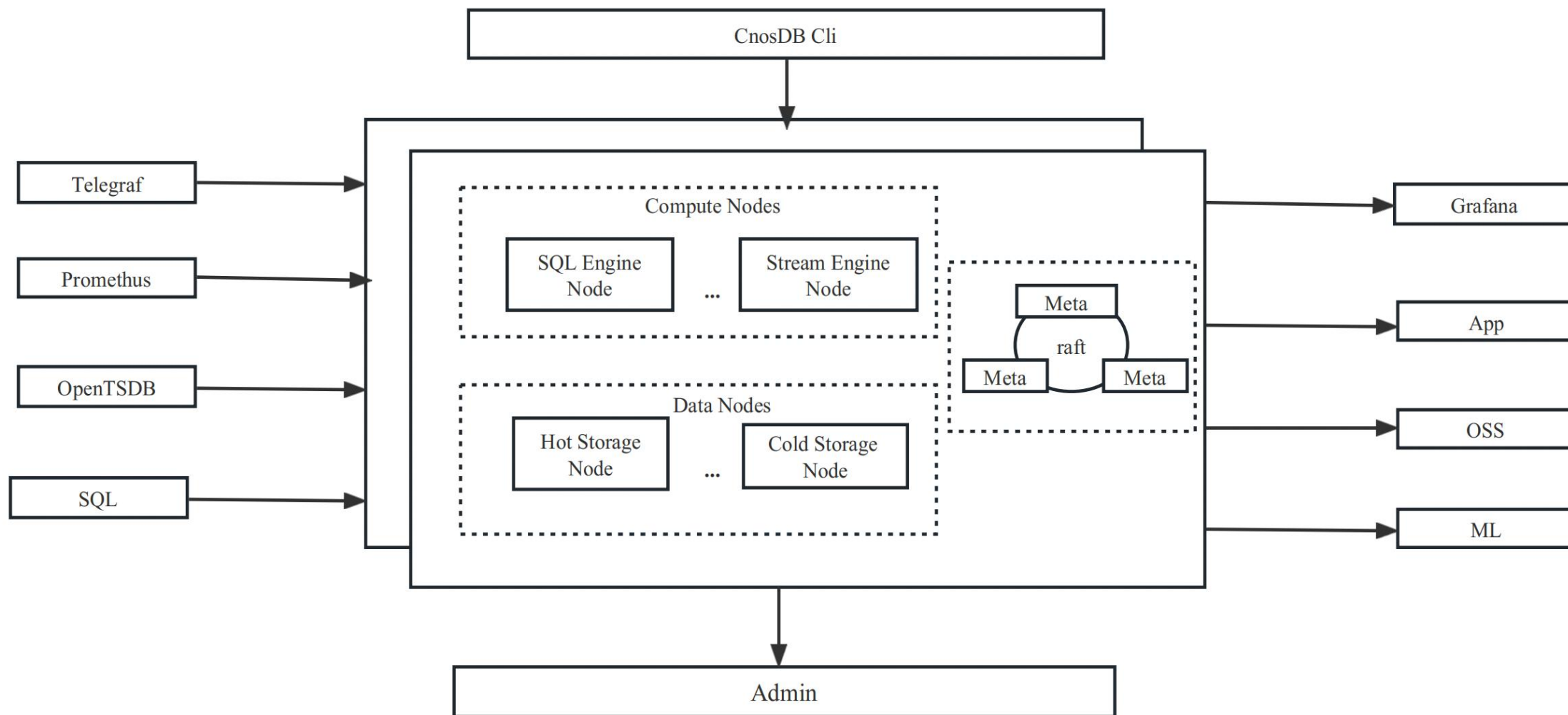
特性

- 横/纵 向扩展
- 计算存储分离
- 平衡存储性能与成本
- 查询引擎支持矢量化查询
- 兼容多种时序协议
- 可观测性

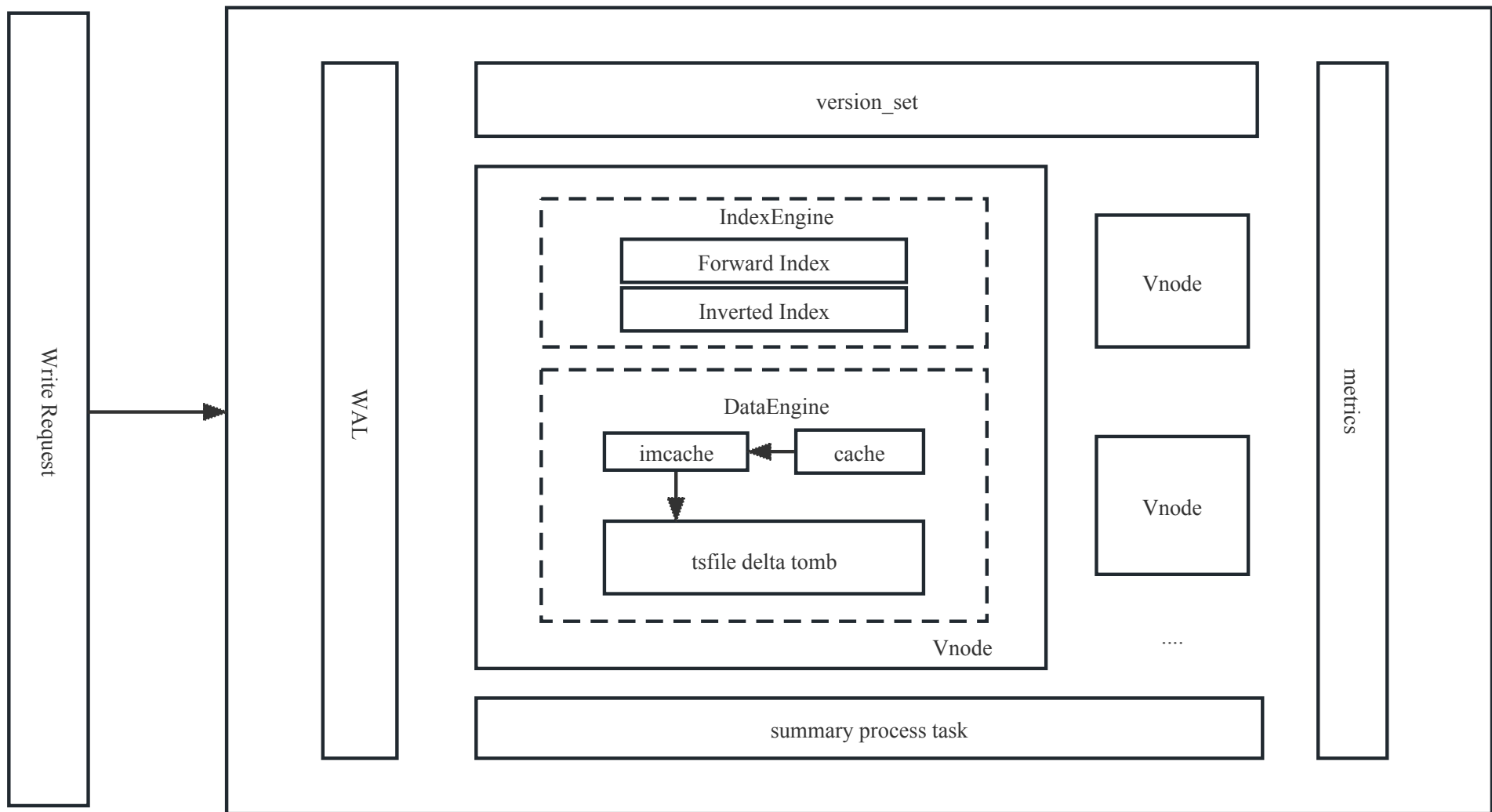
- 支持云原生
- 原生支持多租户
- 租户Quota可动态配置
- 云边端协同
- 云上生态融合



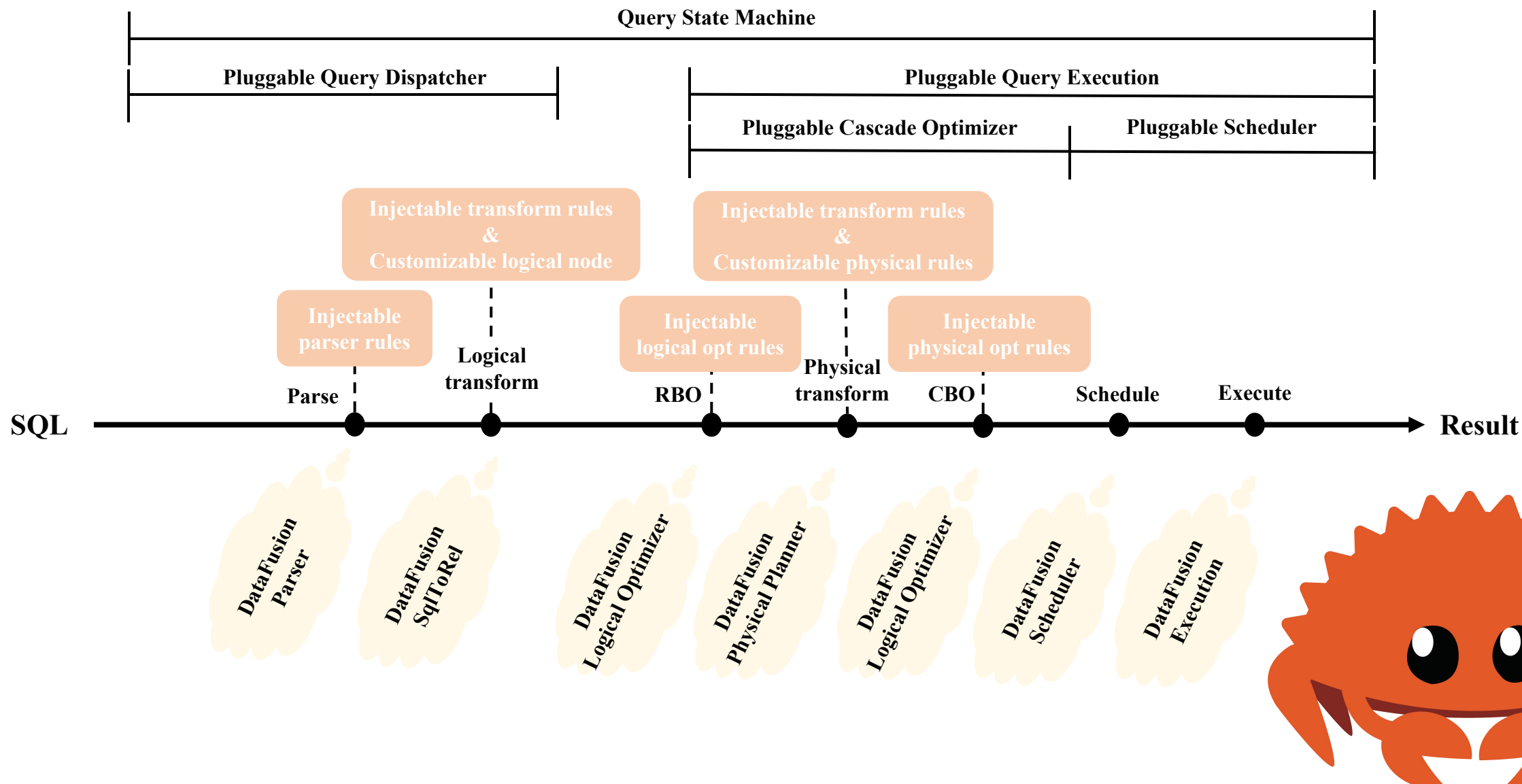
整体架构



1.2 存储引擎



1.3 基于 DataFusion 的高性能查询引擎



1.4基于DataFusion的高性能查询引擎

- 扩展数据源
- 扩展 SQL 语句
- 扩展流处理引擎
- 扩展优化规则
- 扩展时序函数

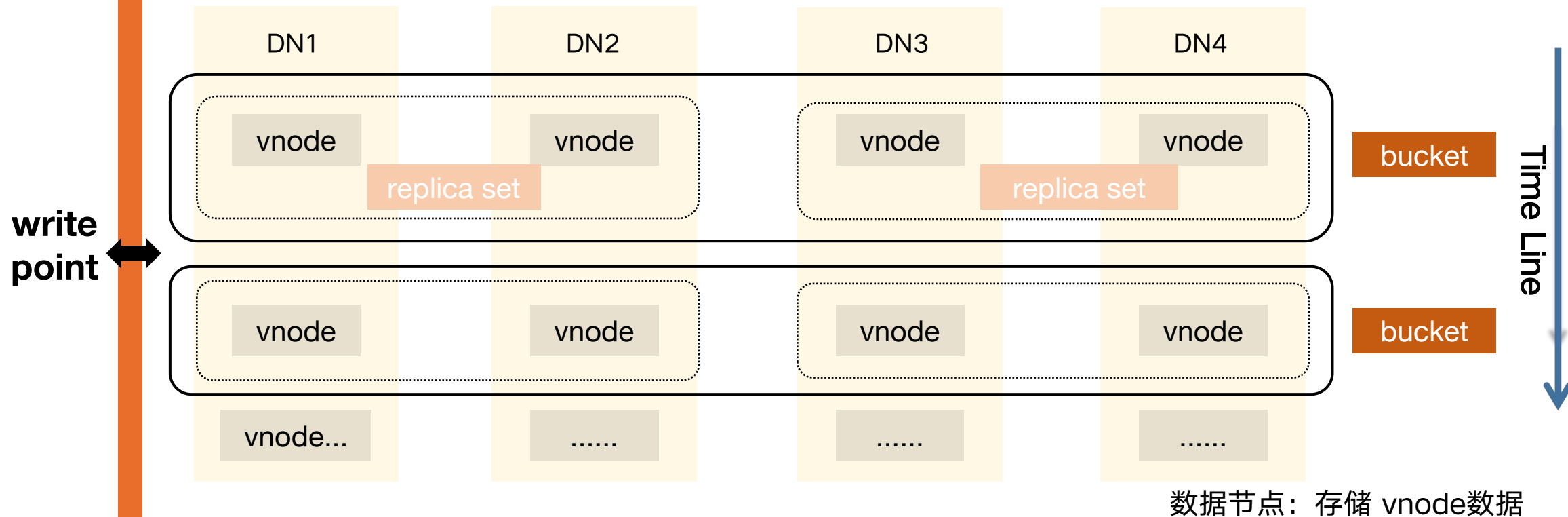
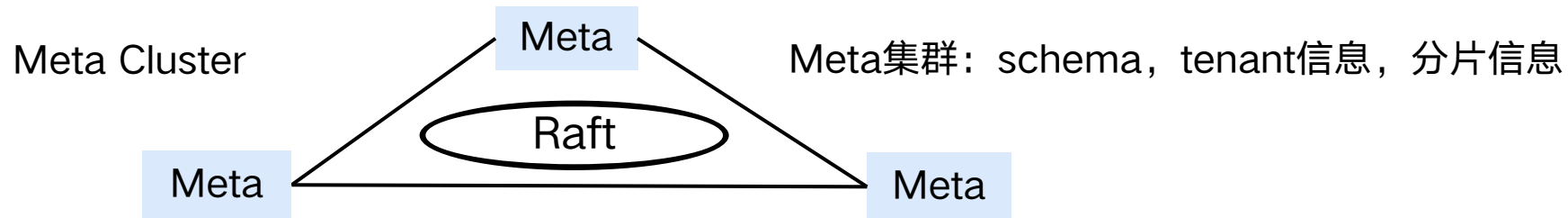


1.5 分布式

1. Shared nothing
2. Leaderless
NRW
Peer to peer model
3. Leader replica group (Coming soon)



1.5 分布式



2. 为何从 Go 切换到 Rust



- 高性能：无 GC 实时控制
- 安全：内存安全 线程安全
- 表达能力强：支持范型，match 表达式



3. Rust 使用经验分享



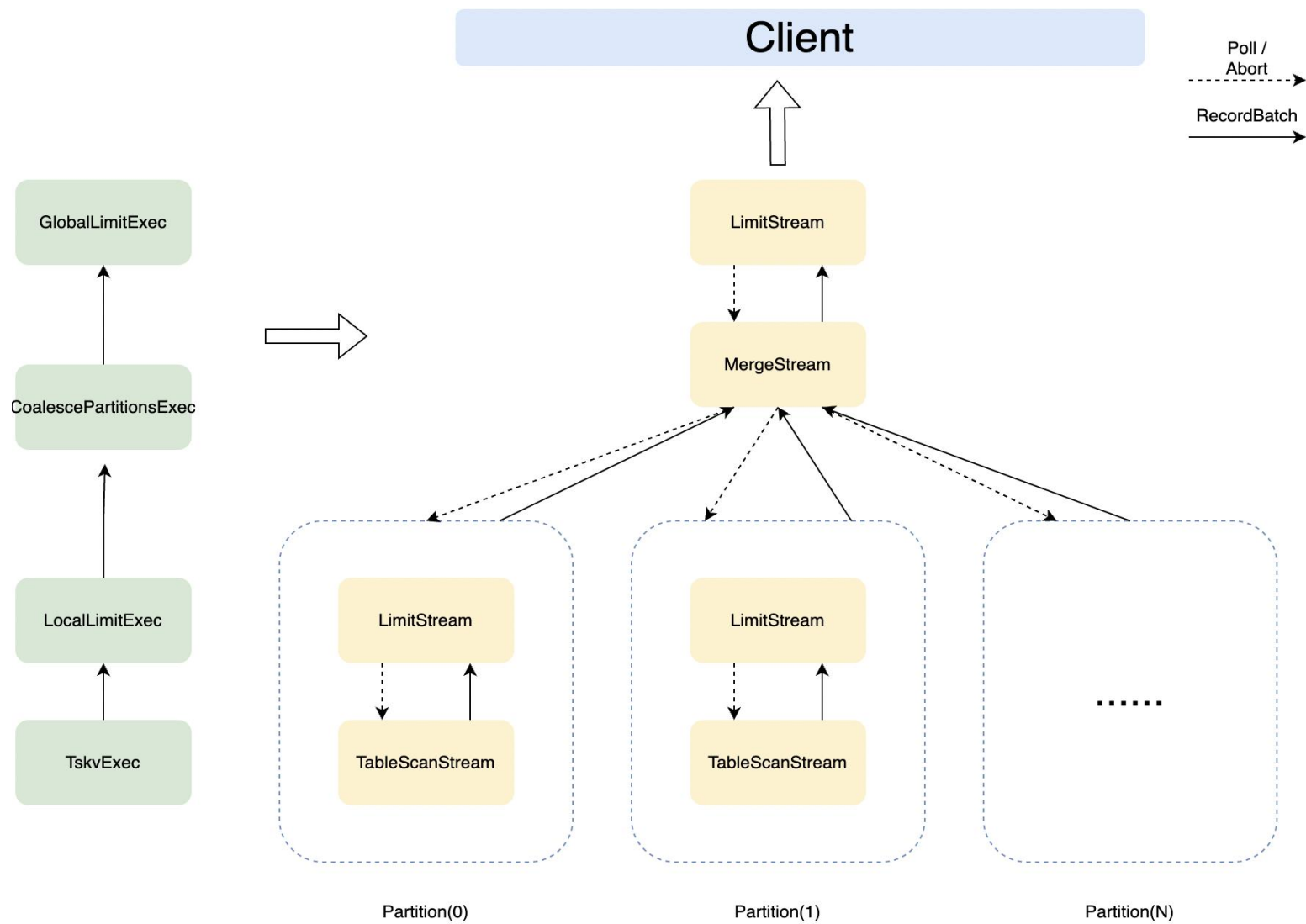
3.1 内存布局

```
pub enum Number {  
    Integer(i64),  
    Float(f64),  
    Complex { real: f64, imaginary: f64 },  
}
```

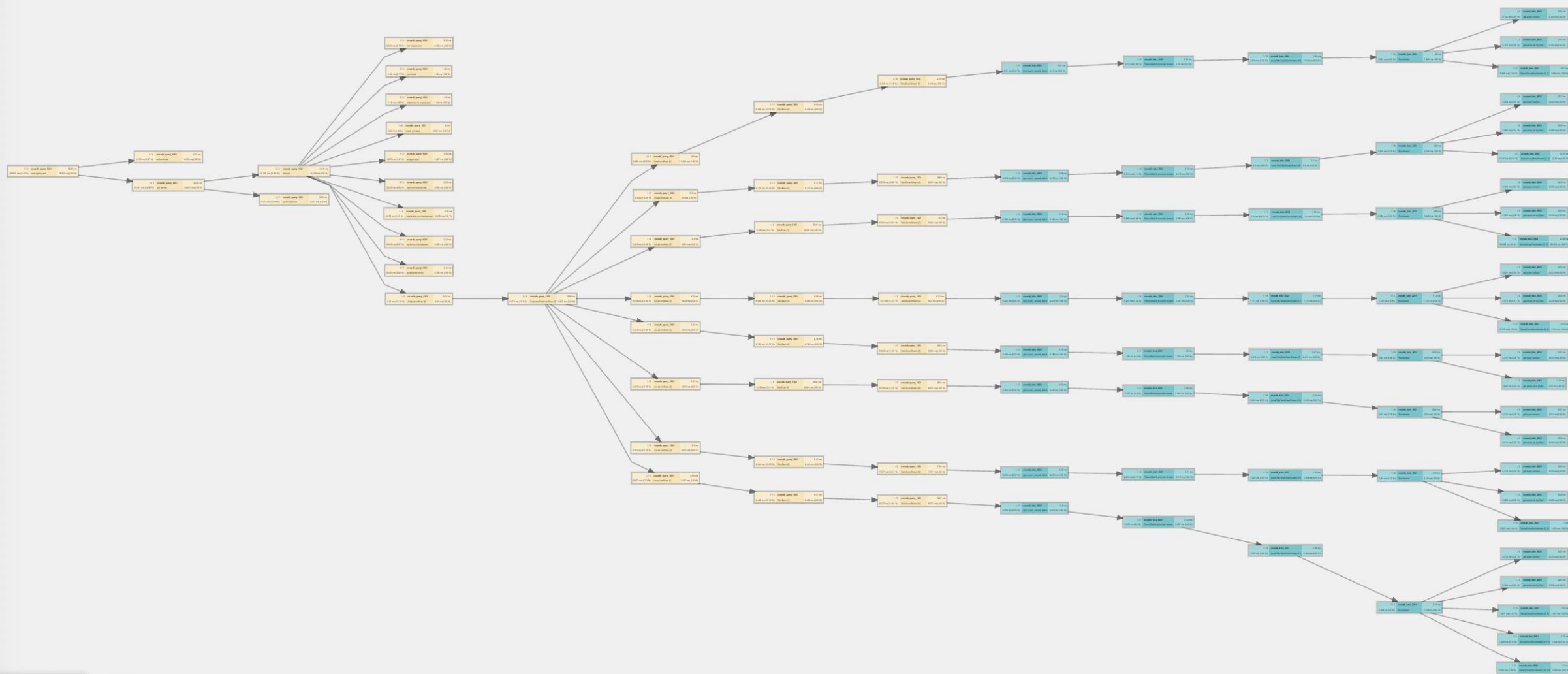
Rust 中的枚举与 C 语言中的联合体类似，如左图所示的枚举代码，实际大小为16字节加8个字节的鉴别器，总共24字节。

| offset | Integer | Float | Complex |
|--------|-------------------|-------------------|-------------------|
| 0 | Discriminator (0) | Discriminator (1) | Discriminator (2) |
| 8 | i64 | f64 | f64 |
| 16 | | | f64 |

3.2 流式处理



3.2 流式处理



3.3 Rust 的锁

- 1. sync rwlock parking_lot
 async rwlock tokio

- 2. 异步死锁检测 <https://github.com/tokio-rs/async-backtrace>

```
taskdump::foo::{{closure}} at backtrace/examples/taskdump.rs:22:1
└─ taskdump::bar::{{closure}} at backtrace/examples/taskdump.rs:27:1
    └─ taskdump::buz::{{closure}} at backtrace/examples/taskdump.rs:37:1
        └─ taskdump::baz::{{closure}} at backtrace/examples/taskdump.rs:42:1
            └─ taskdump::fiz::{{closure}} at backtrace/examples/taskdump.rs:47:1
```



3.4 Rust 交叉编译

1. 找出目标系统的三元组 {arch}-{vendor}-{sys}-{abi}
2. Rust编译工具链 `rustup target add $target, --target=$rustc_target`
3. 配置链接器 `-C linker=$gcc_prefix-gcc` 指示rustc采用的C链接器程序

cross 提供了 “零设置” 的交叉编译 rust crate

它提供了一个环境、交叉工具链和交叉编译库，可以生成最便携的二进制文件



■ 3.5 异步 IO

1. IO 异步化

- 平台兼容性
- 隔离阻塞 IO 异步化

2. io_uring

- 性能有40%提高



4. 反哺社区



4.1 DataFusion

1. <https://github.com/cnosdb/cnosdb/issues/852>
2. <https://github.com/cnosdb/cnosdb/issues/784>
3. <https://github.com/apache/arrow-datafusion/issues/4401>
4. <https://github.com/cnosdb/cnosdb/issues/830>
5. <https://github.com/apache/arrow-datafusion/issues/4843> (该复)
6. <https://github.com/apache/arrow-datafusion/issues/4947>
7. <https://github.com/apache/arrow-datafusion/issues/3778>
8. <https://github.com/apache/arrow-datafusion/issues/4075>
9. <https://github.com/cnosdb/cnosdb/issues/782>
10. <https://github.com/cnosdb/cnosdb/issues/807>
11. <https://github.com/apache/arrow-datafusion/issues/4339>
12. <https://github.com/apache/arrow-datafusion/issues/4297>
13. <https://github.com/apache/arrow-datafusion/issues/4080>
14. <https://github.com/apache/arrow-datafusion/issues/3832>
15. <https://github.com/apache/arrow-datafusion/issues/3830>
16. <https://github.com/apache/arrow-datafusion/issues/4452>

在开发过程中，我们发现一些 DataFusion 的 bug，我们也多次为 DataFusion 提出 issue 和提交 pr。

DataFusion:

<https://github.com/apache/arrow-datafusion>

文章地址:

<https://mp.weixin.qq.com/s/76Y7nnXLlxOkE9Lp9eBkQ>



4.2 Rust 分享

我们在B站有上传 Rust 分享课程，为萌新提供一个学习 Rust 的渠道，同时也分享我们在开发中学习到的内容。

 我们的B站账户名称:  CnosDB

直播间地址: <https://space.bilibili.com/36231559>

♥ 欢迎大家点赞+分享+关注 ~ ~



4.3 CnosDB

官网: <https://www.cnosdb.com>

使用手册: <https://docs.cnosdb.com>

源代码: <https://github.com/cnosdb/cnosdb>



♡♡ 添加社区小助手CC, 欢迎入群技术交流哦~



Thank you!

