# IoTDB 云边数据同步方案调研

王建波

2024 年 9 月 24 日

# 目录


1. IoTDB 云边同步整体结构
2. IoTDB 存储和写流程详解
3. 数据同步
4. IoTDB 云边同步分析与总结

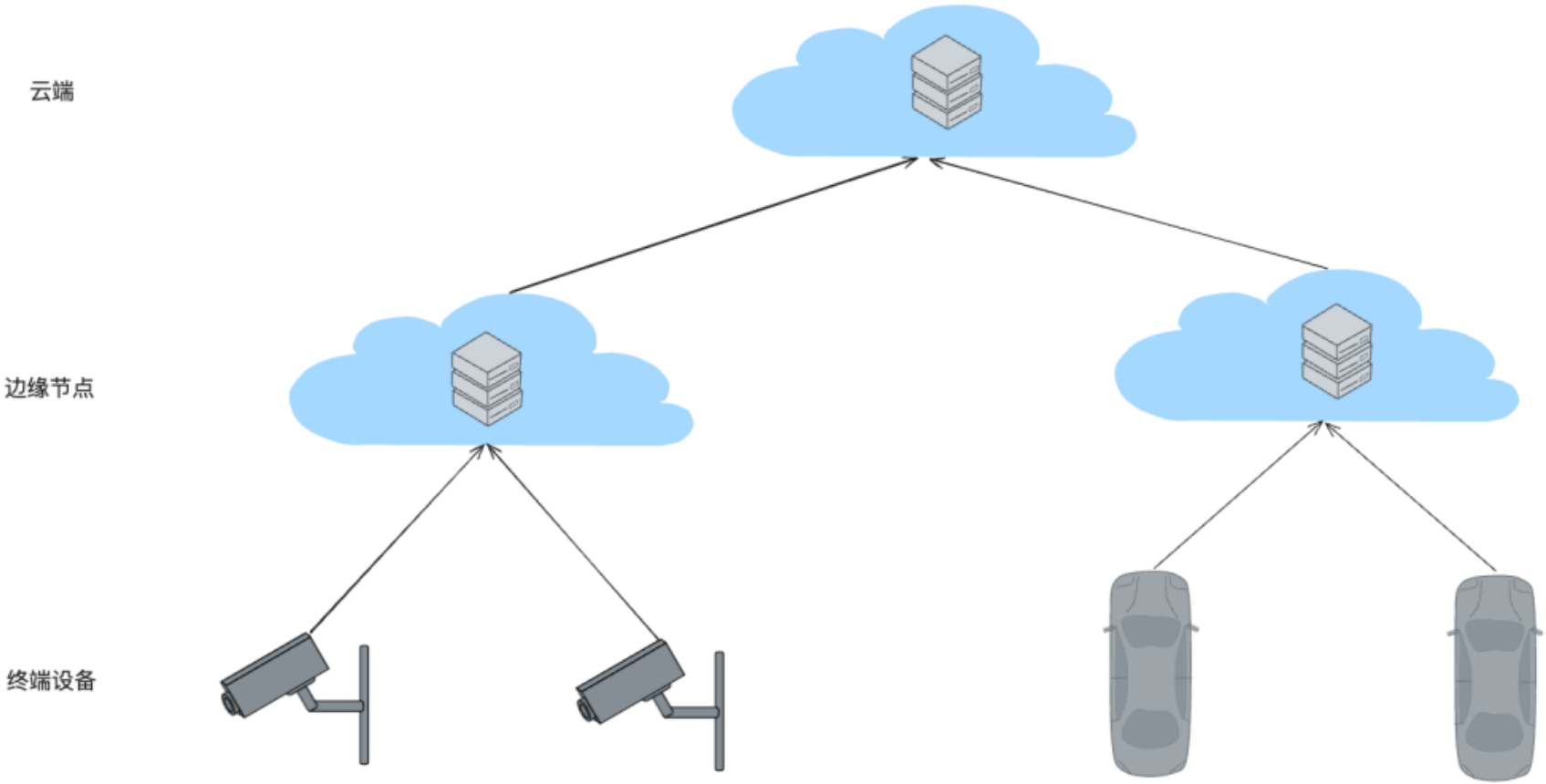
# 一、IoTDB 云边同步整体架构概述

# IoTDB 云边同步整体架构概述

IoTDB 的云边同步旨在实现数据在云端和边缘端之间的高效传输与同步。整体结构主要包含以下几个关键部分：

- 边缘数据源：边缘设备产生的数据源头，如传感器、智能设备等。
- 边缘数据库：在边缘端存储和管理数据的地方，通常具有一定的数据处理和缓存能力。
- 数据同步管道：负责将边缘数据传输到云端或在不同边缘节点之间进行数据同步的通道。
- 云端数据库：接收来自边缘端的数据，并进行集中存储和分析。

# 整体结构示意图

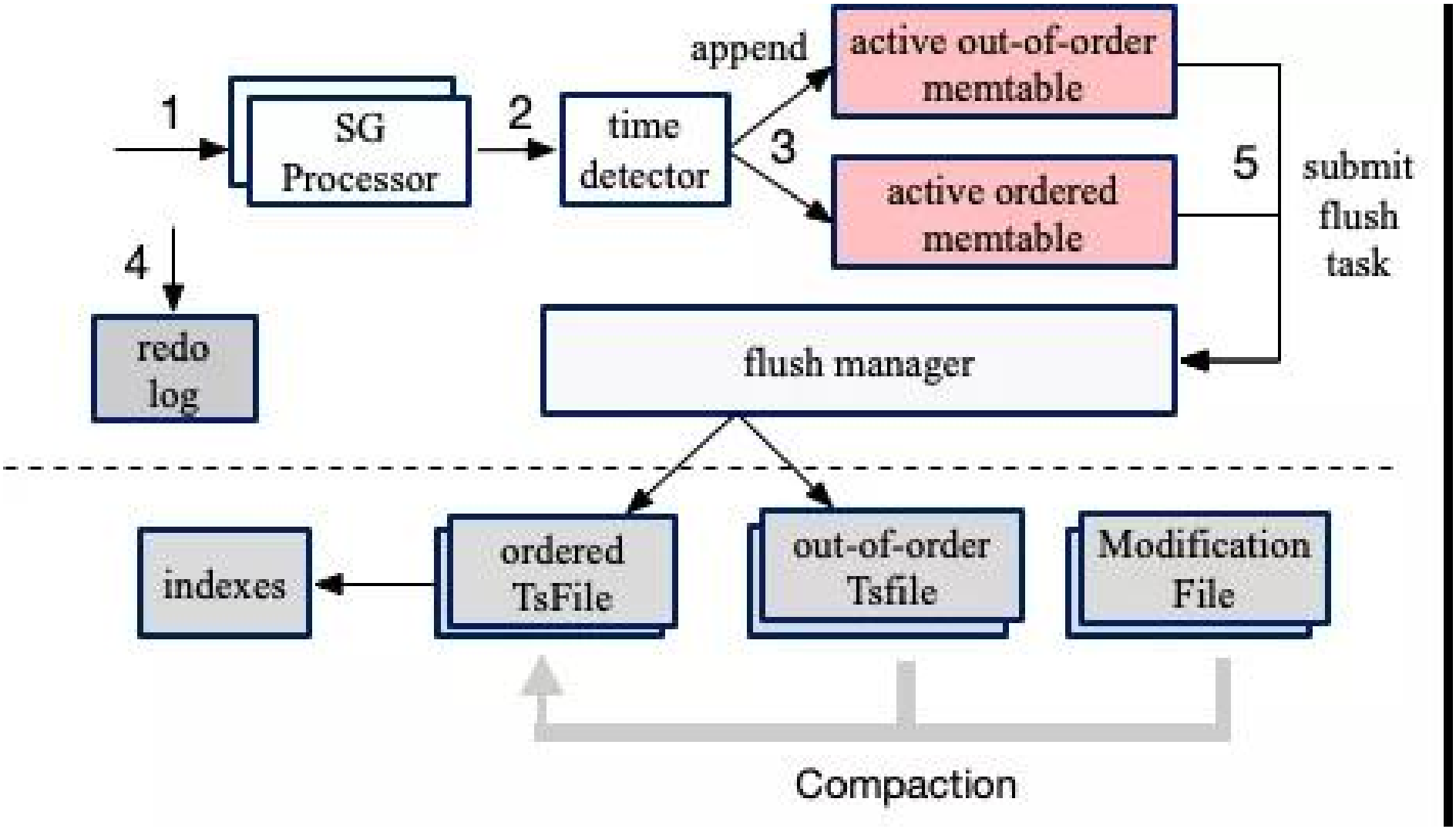


# 二、IoTDB 存储和写流程详解

# IoTDB 存储结构

IoTDB 是一体化收集、存储、管理与分析物联网时序数据的软件系统。

# 存储结构示例图


append
active out-of-order memtable
1
SG Processor
2
time detector
3
active ordered memtable
5
submit flush task
4
redo log
flush manager
indexes
ordered TsFile
out-of-order Tsfile
Modification File
Compaction

# 存储架构

基于 LSM Tree 结构设计，数据先写 WAL，再到内存 memtable，后台刷到磁盘 TsFile，TsFile 进行 Compaction 保证查询效率。

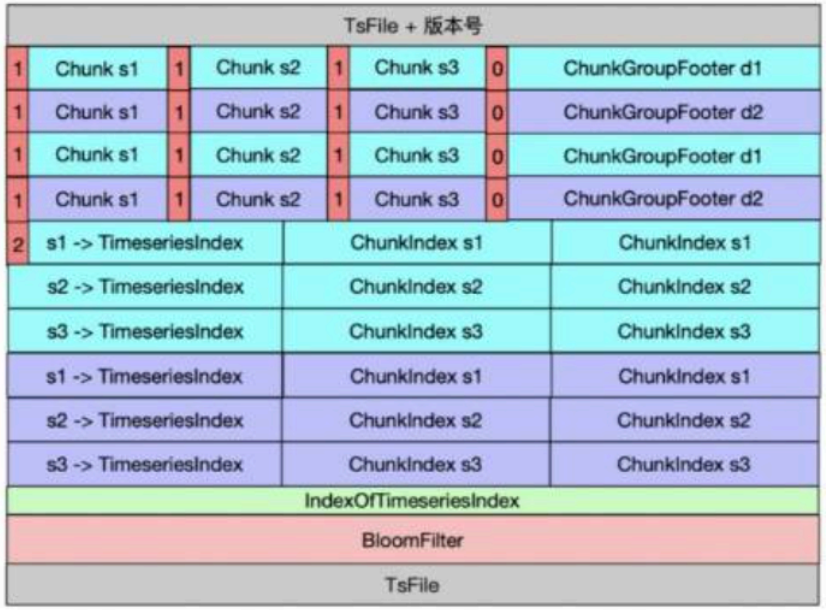
# LSM Tree

- MemTable：内存中保存最近更新数据，按 Key 有序组织，通过 WAL 保证数据可靠性。
- SSTable：磁盘上有序键值对集合，可建索引和布隆过滤器加快查找。

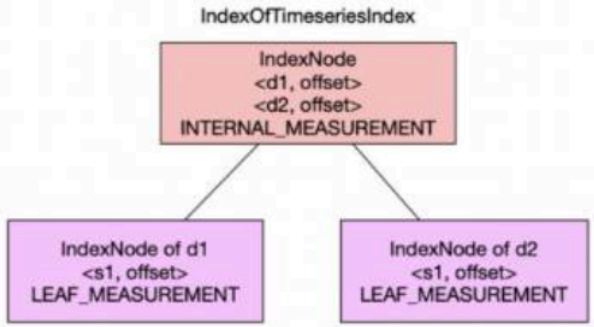
# IoTDB 写入流程

- StorageEngine 负责实例写入和访问，管理 StorageGroupProcessor。
- StorageGroupProcessor 负责存储组一个时间分区内数据写入和访问，管理 TsFileProcessor。
- TsFileProcessor 负责 TsFile 文件数据写入和访问。

# ts-file 结构


TsFile + 版本号
1 Chunk s1 1 Chunk s2 1 Chunk s3 0 ChunkGroupFooter d1
1 Chunk s1 1 Chunk s2 1 Chunk s3 0 ChunkGroupFooter d2
1 Chunk s1 1 Chunk s2 1 Chunk s3 0 ChunkGroupFooter d1
1 Chunk s1 1 Chunk s2 1 Chunk s3 0 ChunkGroupFooter d2
2 s1 -> TimeseriesIndex ChunkIndex s1 ChunkIndex s1
s2 -> TimeseriesIndex ChunkIndex s2 ChunkIndex s2
s3 -> TimeseriesIndex ChunkIndex s3 ChunkIndex s3
s1 -> TimeseriesIndex ChunkIndex s1 ChunkIndex s1
s2 -> TimeseriesIndex ChunkIndex s2 ChunkIndex s2
s3 -> TimeseriesIndex ChunkIndex s3 ChunkIndex s3
IndexOfTimeseriesIndex
BloomFilter
TsFile


Chunk
Chunk Header
Page Header 1
Page data 1
Times
Values
.....
Page Header N
Page data N


IndexOfTimeseriesIndex
IndexNode
<d1, offset>
<d2, offset>
INTERNAL_MEASUREMENT
IndexNode of d1
<s1, offset>
LEAF_MEASUREMENT
IndexNode of d2
<s1, offset>
LEAF_MEASUREMENT

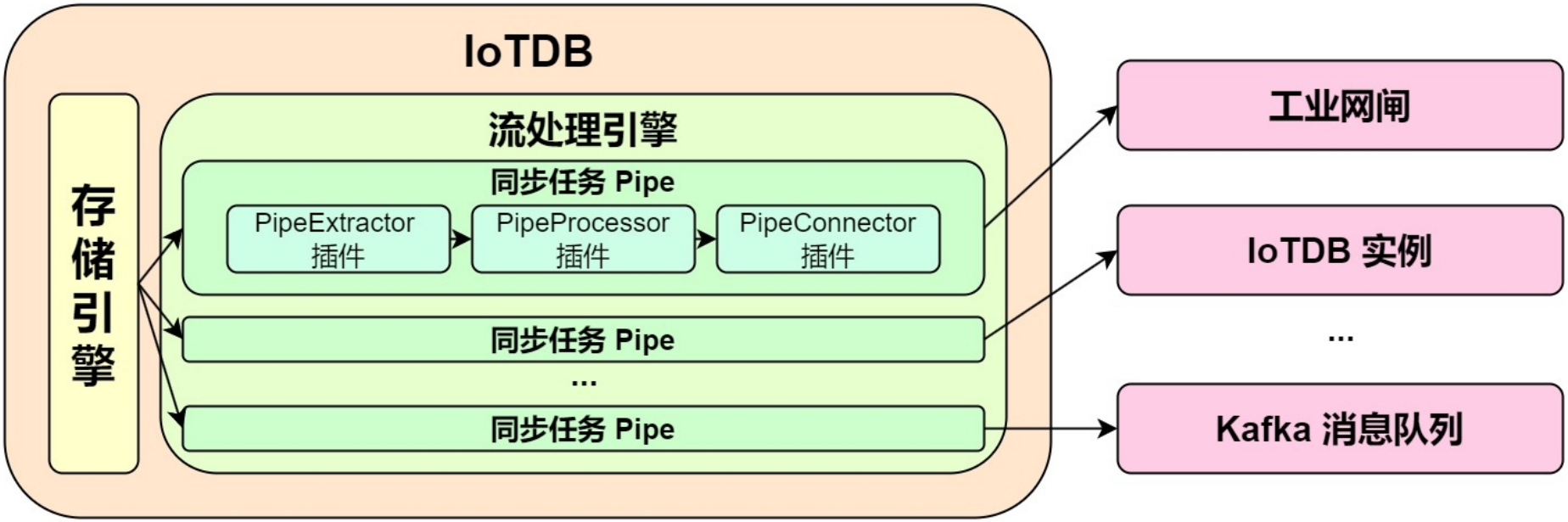
# 三、数据同步方案

# IoTDB 数据同步概述

IoTDB 主要通过创建管道（pipe）实现数据同步。

- 数据同步是通过 Pipe Extractor、Pipe Processor 和 Pipe Connector 三个插件实现。
  (1)Pipe Extractor 用于抽取数据
  (2)Pipe Processor 用于处理数据
  (3)Pipe Connector 用于发送数据，最终数据将被发至外部系统

# IoTDB 数据同步图解



# IoTDB 数据同步语法

```
CREATE PIPE <PipeId> -- PipeId 是能够唯一标定同步任务任务的名字
WITH EXTRACTOR (
  -- 默认的 IoTDB 数据抽取插件
  'extractor'                    = 'iotdb-extractor',
  -- 路径前缀，只有能够匹配该路径前缀的数据才会被抽取，用作后续的处理和发送
  'extractor.pattern'            = 'root.timecho',
  -- 是否抽取历史数据
  'extractor.history.enable'     = 'true',
  -- 描述被抽取的历史数据的时间范围，表示最早时间
  'extractor.history.start-time' = '2011.12.03T10:15:30+01:00',
  -- 描述被抽取的历史数据的时间范围，表示最晚时间
  'extractor.history.end-time'   = '2022.12.03T10:15:30+01:00',
  -- 是否抽取实时数据
  'extractor.realtime.enable'    = 'true',
)
WITH PROCESSOR (
  -- 默认的数据处理插件，即不做任何处理
  'processor'                    = 'do-nothing-processor',
)
WITH CONNECTOR (
  -- IoTDB 数据发送插件，目标端为 IoTDB
  'connector'                    = 'iotdb-thrift-connector',
  -- 目标端 IoTDB 其中一个 DataNode 节点的数据服务 ip
  'connector.ip'                 = '127.0.0.1',
  -- 目标端 IoTDB 其中一个 DataNode 节点的数据服务 port
  'connector.port'               = '6667',
)
```

# IoTDB 数据同步方案

IoTDB 主要通过指定创建管道（pipe）参数实现不同类型的数据同步。

- 全量数据同步：用于将一个实例所有数据迁移到另一个实例。
- 范围数据同步：同步特定时间范围和测点的数据。
- 数据降采样同步：转换高频率数据为低频率。
- 级联数据同步：多个场站联动汇聚数据，依次创建管道形成链式同步。
- 双向数据同步：实现两个实例数据实时镜像，如在 A、B 实例分别执行特定 SQL 创建双向同步管道。

# IoTDB 底层数据同步插件

| 插件名称 | 类型 | 介绍 | 适用版本 |
| --- | --- | --- | --- |
| iotdb-source | source 插件 | 默认的 source 插件，用于抽取 IoTDB 历史或实时数据 | 1.2.x |
| iotdb-thrift-sink | sink 插件 | 用于 IoTDB（v1.2.0及以上）与 IoTDB（v1.2.0及以上）之间的数据传输。使用 Thrift RPC 框架传输数据，多线程 async non-blocking IO 模型，传输性能高，尤其适用于目标端为分布式时的场景 | 1.2.x |
| iotdb-air-gap-sink | sink 插件 | 用于 IoTDB（v1.2.2+）向 IoTDB（v1.2.2+）跨单向数据网闸的数据同步。支持的网闸型号包括南瑞 Syskeeper 2000 等 | 1.2.2 及以上 |
| iotdb-thrift-ssl-sink | sink plugin | 用于 IoTDB（v1.3.1及以上）与 IoTDB（v1.2.0及以上）之间的数据传输。使用 Thrift RPC 框架传输数据，单线程 sync blocking IO 模型，适用于安全需求较高的场景 | 1.3.1 及以上 |

# 四、IoTDB 云边同步分析与总结

# IoTDB 云边同步的不足之处

- 网络依赖：同步过程高度依赖网络连接的稳定性和带宽。如果网络出现问题，可能导致同步中断或延迟。
- 数据一致性挑战：在复杂的同步场景下，尤其是双向同步和级联同步中，确保数据的一致性可能较为困难。
- 资源消耗：数据同步可能消耗大量的计算资源和存储资源，尤其是在大规模数据同步或高频率同步的情况下。
- 配置复杂性：对于不同的同步方案，需要进行较为复杂的配置和管理，可能对用户的技术水平要求较高。

# 总结

IoTDB 端边云同步操作简单，仅需执行一句 SQL 即可完成。同时支持加密传输、压缩传输，用户可探索高阶使用方法。然而，也需要注意其在网络依赖、数据一致性、资源消耗和配置复杂性等方面的不足。