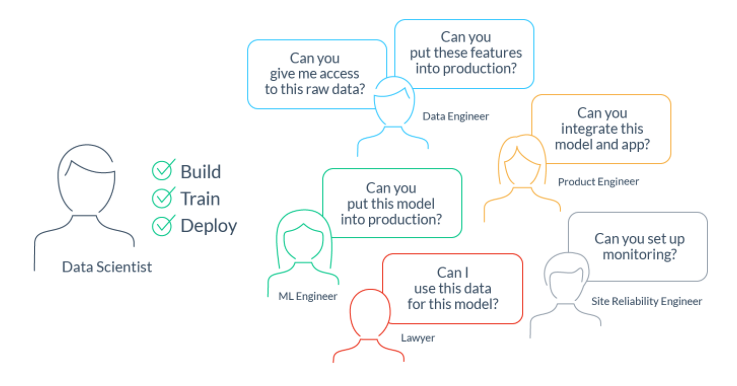
OpenMLDB 调研报告

# 背景

Mechine Learning 应用投入生产十分困难，训练开发一个模型到投入生产，在这个漫长的过程中可能需求就已经过时了，导致许多机器学习应用没有等到投产的那一步。因为对于 Ai 工程来说，95% 的时间和精力都消耗在了跟数据相关的工作负载上。

Feature Store 是什么，引用我们对特征平台的描述：特征平台是为模型服务（包括模型训练和模型预测）更快、更好、更健康地提供特征数据的工程系统，托管了特征生产、注册、存储、服务和管控全流程，使得各业务团队从工程泥潭解放出来，专注于特征效果的探索。同时，通过对特征链路和数据提供统一的抽象与标准，降低特征数据生产、复用门槛，经验共享，提升模型整体效果。



# OpenMLDB 是什么？

OpenMLDB是一个开源机器学习数据库，它提供了一个功能平台，能够实现一致的培训和推理功能。

实时特征计算对许多机器学习应用至关重要，例如实时个性化推荐和风险分析。然而，数据科学家开发的特征工程脚本（通常为Python代码）往往无法直接部署到生产环境进行在线推理，因为它通常无法满足**低延迟、高吞吐量**和高可用性等工程要求。因此，需要工程团队介入，使用数据库或C++等高性能技术对源代码进行重构和优化，以确保其效率和鲁棒性。

由于开发和部署生命周期中涉及两个团队和两套技术栈，保持线上线下特征一致性验证变得尤为关键，这一过程通常会耗费大量时间和人力成本。

# OpenMLDB 的特性？

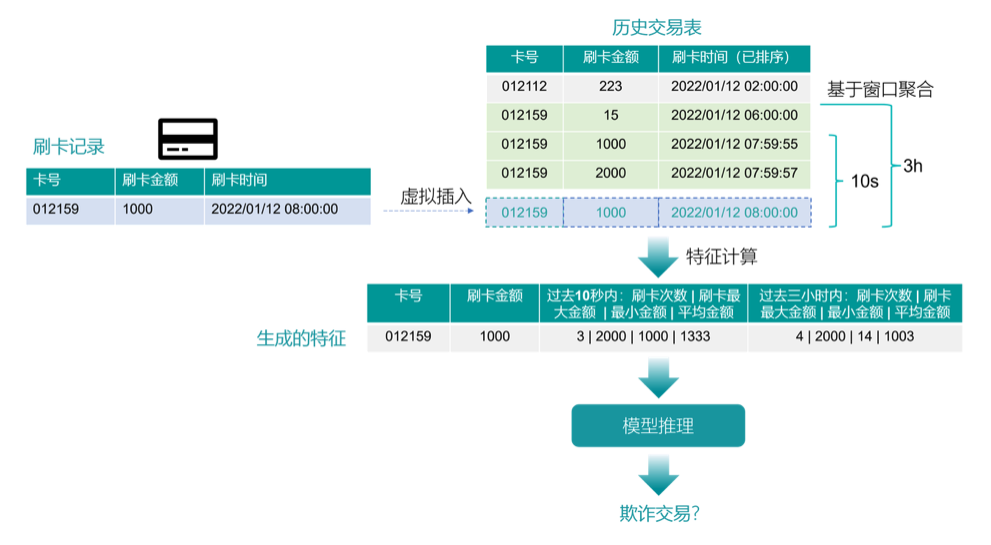
OpenMLDB 给自己说明是：https://openmldb.ai低门槛（SQL 开发）、线上线下一致、实时性高性能、满足生产级需求（HA、可扩展等）



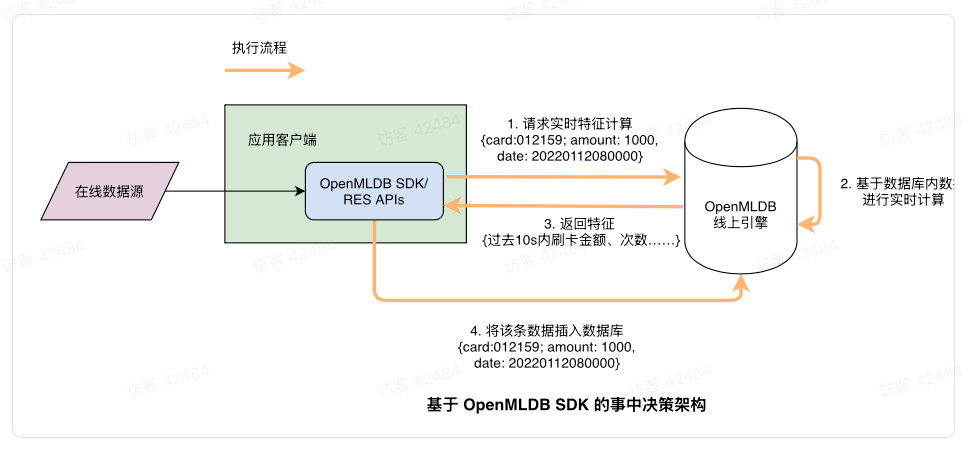
先解释一下实时计算（面向特征计算的定制化优化），OpenMLDB 大部分的文档和网站都是介绍自己专注于时间序列数据的实时特征计算，可以达到毫秒级的延迟计算，**这一点是区别与我们的主要场景，除了 On-demand transform 场景需要实时特征计算，其他的可以通过预先特征抽取+存储来“提速”。**

场景特点：需要实时计算的数据，比如实时欺诈检测、个性化推荐等领域，需要计算的**特征和即时数据**有很大关系，如果频繁的更新存储中的特征会带来较大的压力，所以采用实时计算的方式。

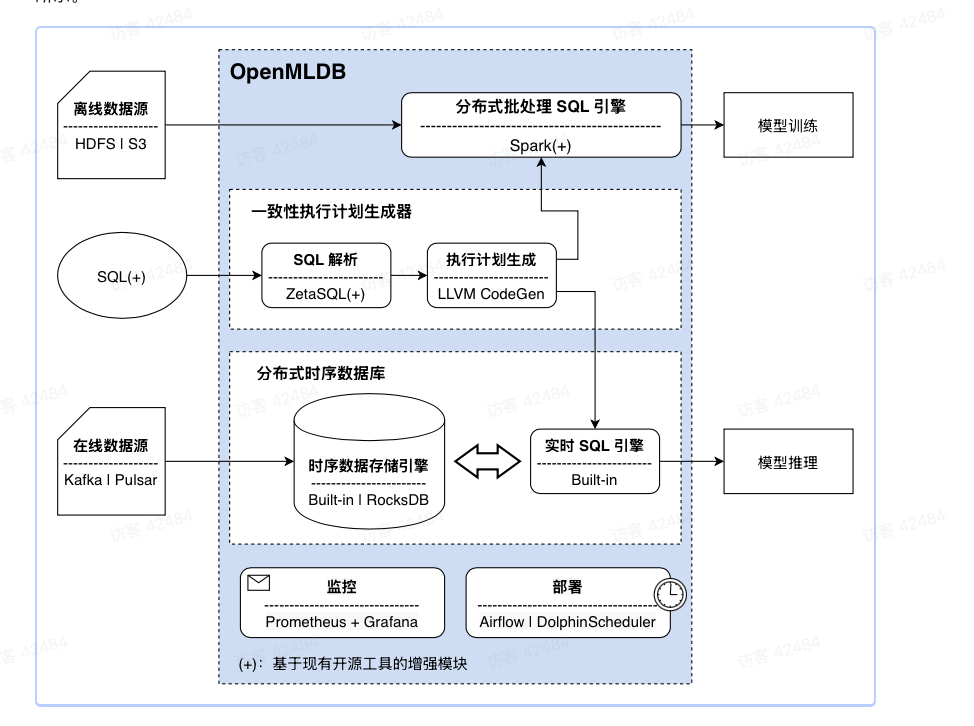
**Open MLDB默认计算模式为支持事中决策的应用，**主要特点是***当前事件产生的行为数据也会被纳入决策考量中。*** [如下](https://openmldb.feishu.cn/wiki/wikcndfgIqvhHTKmwR3PW3dW9Sb#doxcnagKQM0E4u6uesoFutqsx7P)：



OpenMLDB 扮演的角色是一个特征提取引擎：



OpenMLDB 的整体架构如下：



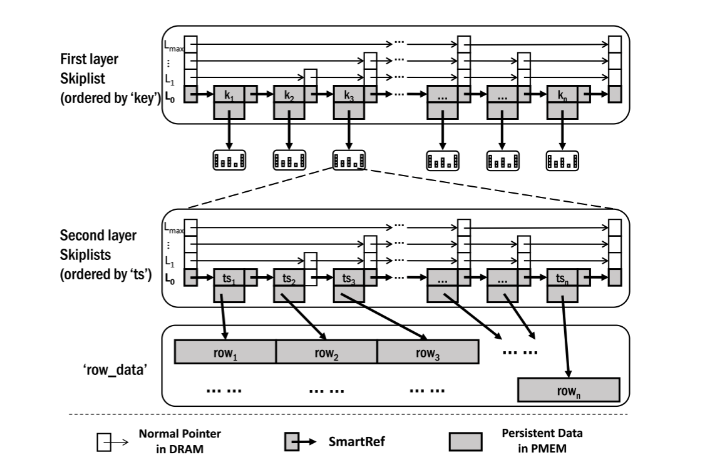
## 基于 SQL 语言的特征提取开发

OpenMLDB 对外暴露 SQL 作为统一的使用接口，自研了 [FEQL](https://doc.weixin.qq.com/doc/w3_ATUA1AZfANsyvIiVlg8Tv0tnI5uiu?scode=AJEAIQdfAAoyB9aUZXATUA1AZfANs)开发语言，并且在标准 SQL 的基础上做了功能扩展。

## **如何实现实时计算，达到毫秒级别的在线特征提取时延呢？**

OpenMLDB 对内存数据库做了针对时序数据流的优化[OpenMLDB的实时特征数据库介绍](https://doc.weixin.qq.com/doc/w3_ATUA1AZfANsyvIiVlg8Tv0tnI5uiu?scode=AJEAIQdfAAo1haYvdXATUA1AZfANs)，（1）设计了专为时序特征计算的**分布式内存数据库 FEDB ；**（2）同时也为大窗口计算设计了预聚合技术；（[参考文章](https://openmldb.ai/openmldb-进阶使用攻略和高级特性介绍/)）；（3）针对性优化，增强了 OpenMLDB 的流式特征的计算能力；

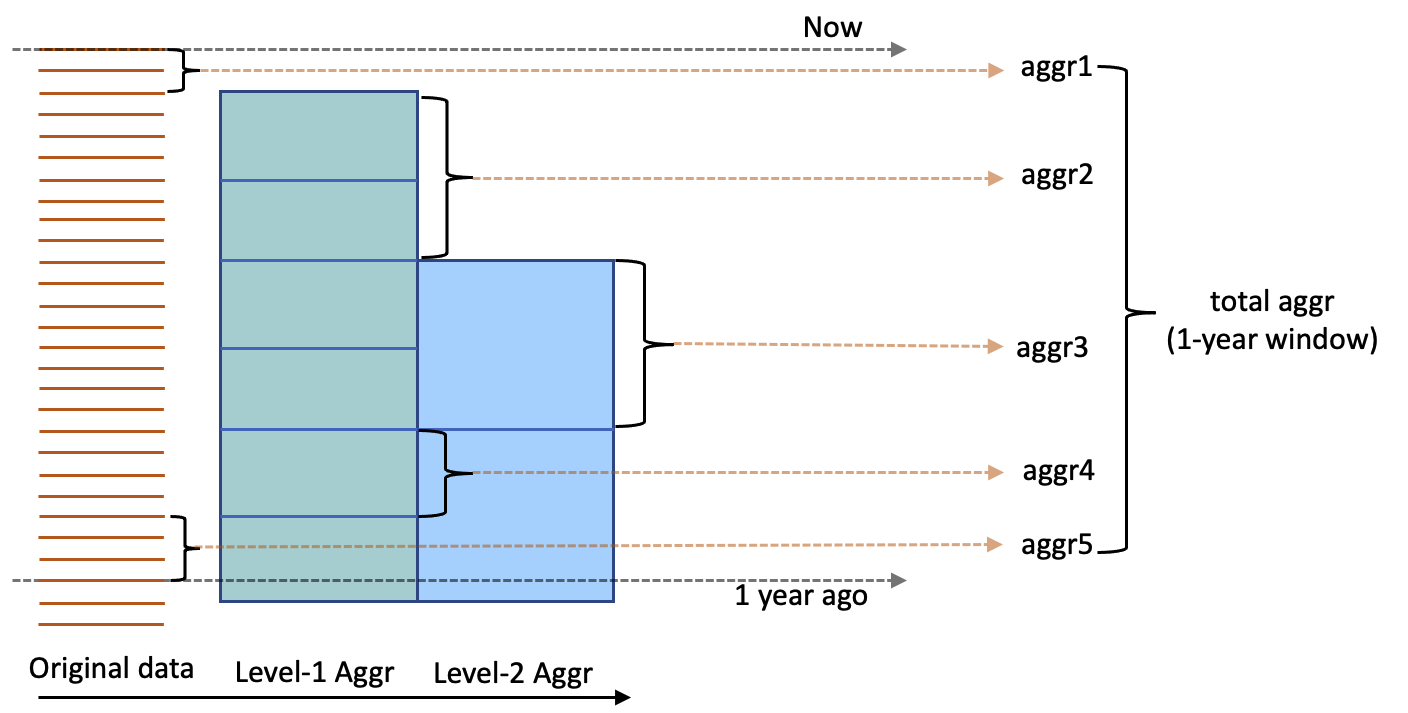
**（1）FEDB 的主要数据结构是一个双层跳表：**

****

第一层跳表中的 key 对应于索引列的具体值，value 是一个指针指向二级跳表下已经被聚合在一起的所有对应于 key 值的行的集合。该索引需要优化的是类似数据库中的分组操作（即 group by）。对应于特征工程，即为快速找到数据表中某一特定键值（比如某个用户）的所有相关记录。

第二层跳表的 key 一般是时间戳，value 是对应的这一行数据。数据行按照时间戳从大到小的时间降序排序。这一级的跳表优化的即是基于特定时间窗口内的聚合计算，需要高效的找到窗口内的所有数据。

**（2） 预聚合操作**



在一些典型场景中，时序特征的窗口内数据量可能很大（比如窗口的时间跨度横跨三年），我们把这种时序特征称之为“长窗口”特征。对于长窗口的特征计算，遍历计算会随着数据量而增加；另外，相邻的特征计算大概率包含重复计算。

预聚合技术可以利用预先聚合结果来减少计算。比如计算图中 aggr = agg（agg1，agg2，agg3，agg4，agg5），2～4 都是预先计算好的结果。

*但是使用限制是：仅支持一个物理表，不支持 Join 和 union 操作，聚合运算也只支持：sum、avg、count、min、max、 以及\*\_where，\*就是前面五个聚合，限制比较多。*

**（3）优化流式计算的关键操作 Interval Join**

OpenMLDB 的流式特征计算能力，对其中的关键操作 Interval Join 进行了深度优化，取得了比目前工业界普遍使用的算法高达数量级的吞吐和延迟优化。基于该优化算法，克服了流式计算在实时机器学习领域所遇到的性能瓶颈，可以支持在金融、风控、推荐等领域的毫秒级实时流式特征计算的需求。

OpenMLDB 发布了一篇论文来解释了针对 Online Interval Join 操作的优化：[OpenMLDB 的可扩展在线区间连接](https://doc.weixin.qq.com/doc/w3_ATUA1AZfANsbGeS1bSjRPi2sCzSTE?scode=AJEAIQdfAAoLnv7LtzATUA1AZfANs)

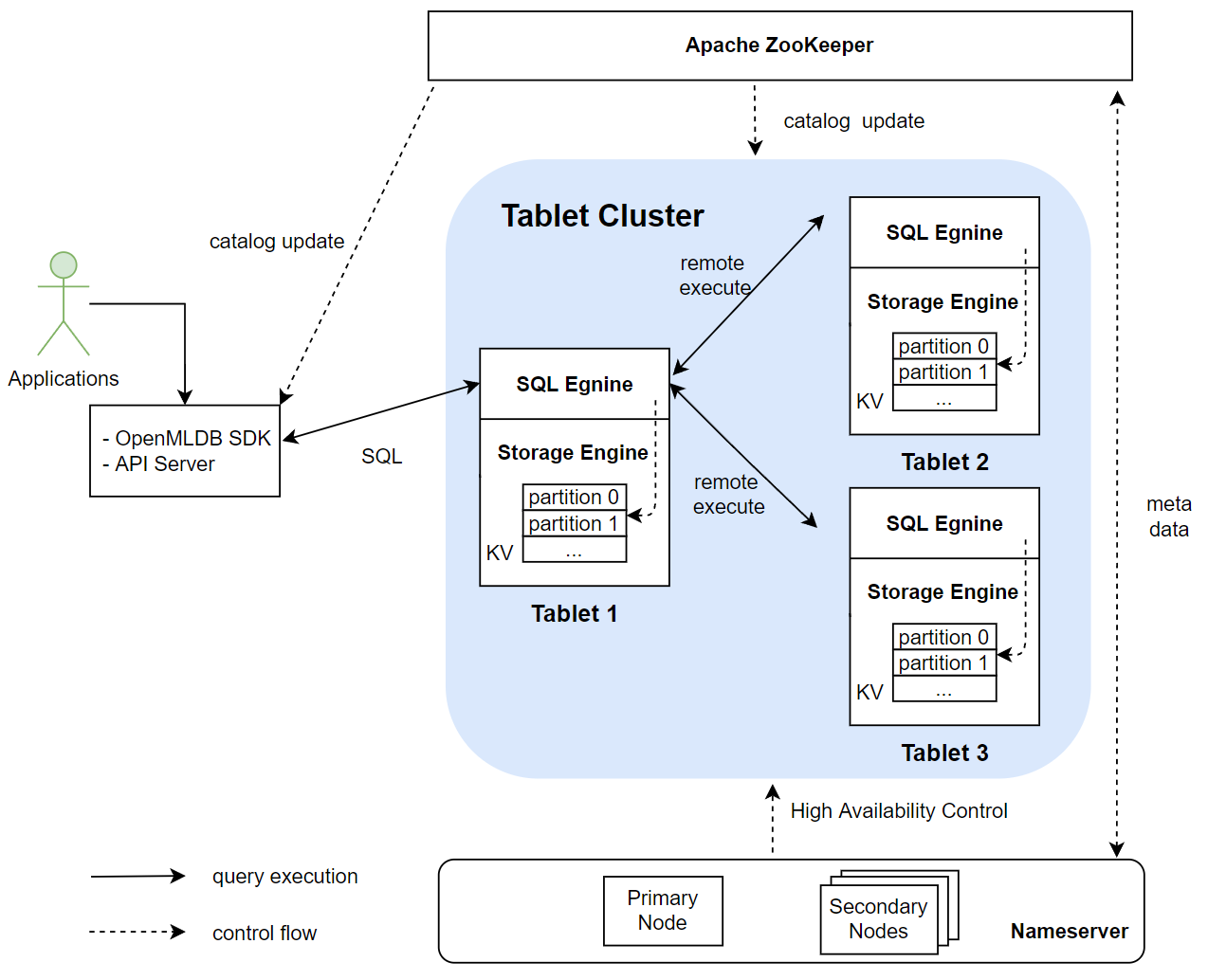
## 如何实现线上线下一致性？

这个任何 Feature Store 好像都可以实现，比如我们的平台好像可以使用 python DSL 来屏蔽掉底层实现，借此来达到一致的目的？

OpenMLDB 是通过使用一致性执行计划生成器来实现的，实际上， 能获得的线上线下一致性信息，更偏向于使用同样的 SQL 脚本部署在线上线下来实现逻辑一致性，通过线上到线下数据自动同步工具实现数据一致性。关于如何使用一致性计划生成器实现线上线下分别优化来达到线上线下一致性计算结果倒是没有怎么提到。

## 如何满足生产级别要求？

OpenMLDB的在线架构主要模块包括Apache ZooKeeper、nameserver集群（nameserver cluster）和存储计算节点集群（tablet cluster）。每个存储计算节点（tablet）由SQL引擎和存储引擎组成。ZooKeeper和nameserver主要用于辅助功能，例如元数据管理和高可用保障。



OpenMLDB 依赖 Zookeeper 实现服务发现和元数据的存储和管理，ZooKeeper 会与 OpenMLDB 的 SDK/API Server、Tablet（存储计算节点） 和 Nameserver 进行交互，负责元数据的分发与更新。

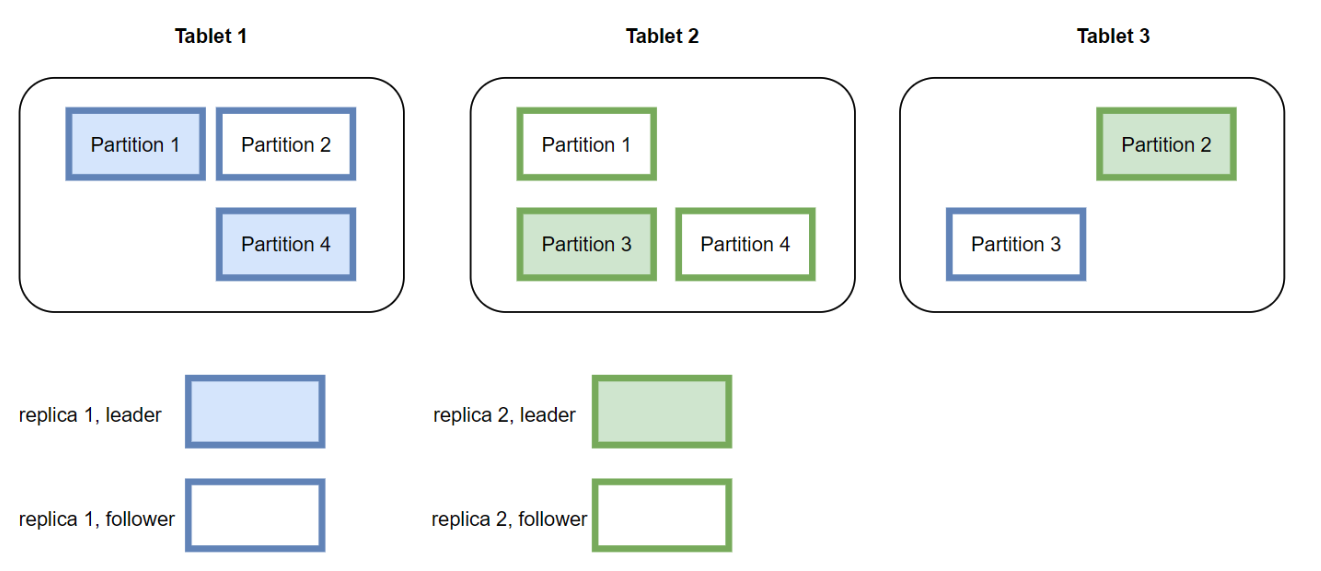
nameserver主要用于存储计算节点（Tablet）的管理与故障转移。当某个存储计算节点发生故障时，命名服务器会被触发执行一系列任务以完成故障切换（fail over） ；待节点恢复后，命名服务器会重新将数据加载至该节点。

为确保命名服务器自身的HA，系统会采用主从（primary/secondary）部署模式同时部署多个命名服务器实例，但同一时间仅允许一个主节点（primary node）存在。主节点通过ZooKeeper的抢占机制从多个命名服务器实例中选举产生。若当前主节点意外下线，其余从节点（secondary nodes）将借助ZooKeeper重新选举出新的主节点。

另外，OpenMLDB 集群版是一个分布式数据库，采用**数据分片（partitions）**和**多副本（replications）**来实现高可用性与提升分布式查询效率。

* 为了确保高可用性并提高分布式查询效率，数据表会以多个副本的形式存储，这些副本被称为replicas。
* 当表（或具体某个副本）存储时，会被进一步划分为多个分区以实现分布式计算。partition 特性如下：分区数量在建表时候指定，不可动态修改；是主从同步、扩缩容的最小单元；可在不同存储计算节点（tablets）间灵活迁移；同一个表的不同分区可以并行计算，提升分布式计算性能；

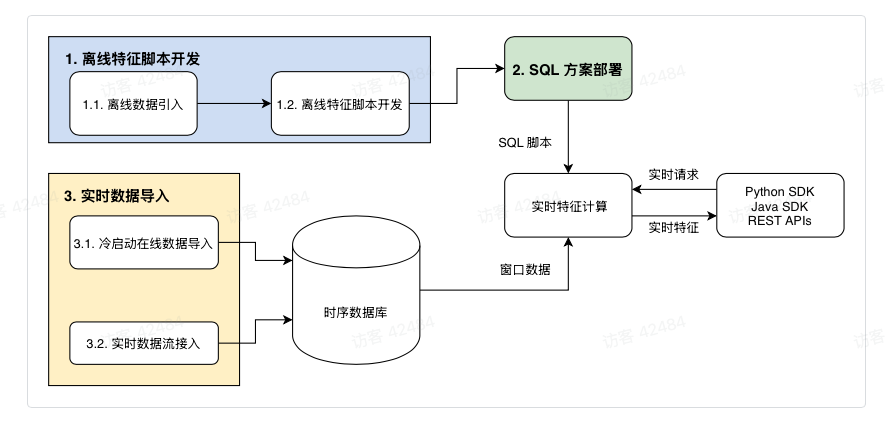
partition 采用主从架构，其中**主分区（leader）**接收并执行计算请求， **从分区（follower）**用于保障高可用性，多个分区分布在不同的 tablet 上。



**问题： Partition 是按照什么进行切片划分的？**

# OpenMLDB 的特征提取流程

使用 OpenMLDB 进行特征开发的具体流程大致如图：

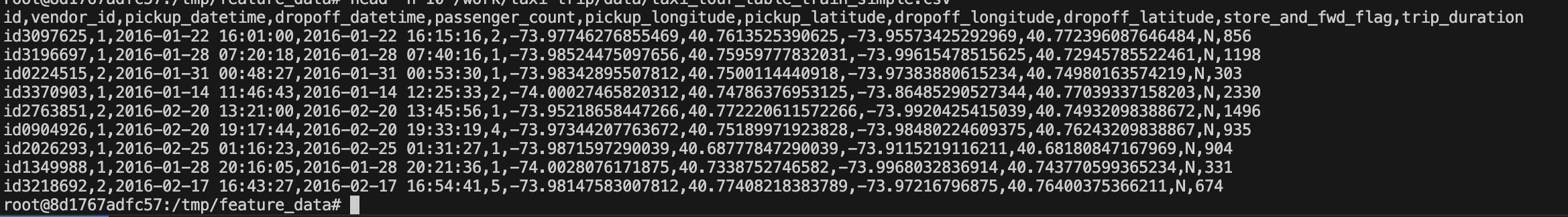


OpenMLDB 给了一个根据上车地点、车载人数和车辆品牌来进行预测形成时间的特征提取例子：

1. **创建一个表结构**

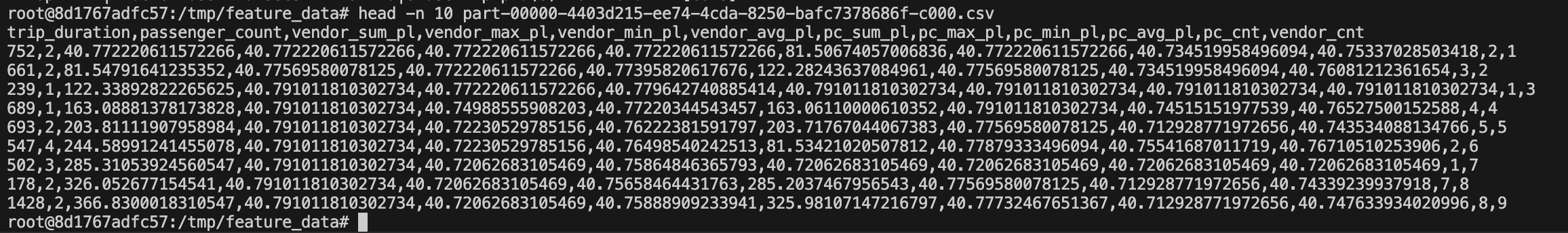
|  |  |  |
| --- | --- | --- |
| **列名** | **数据类型** | **描述** |
| id | STRING | 行程ID |
| vendor\_id | INT | 供应商ID |
| pickup\_datetime | TIMESTAMP | 上车时间 |
| dropoff\_datetime | TIMESTAMP | 下车时间 |
| passenger\_count | INT | 乘客数量 |
| pickup\_longitude | DOUBLE | 上车经度 |
| pickup\_latitude | DOUBLE | 上车纬度 |
| dropoff\_longitude | DOUBLE | 下车经度 |
| dropoff\_latitude | DOUBLE | 下车纬度 |
| store\_and\_fwd\_flag | STRING | 存储转发标志 |
| trip\_duration | INT | 行程持续时间(秒) |

1. **离线导入数据：**



1. **特征设计（对应图中的特征开发）**

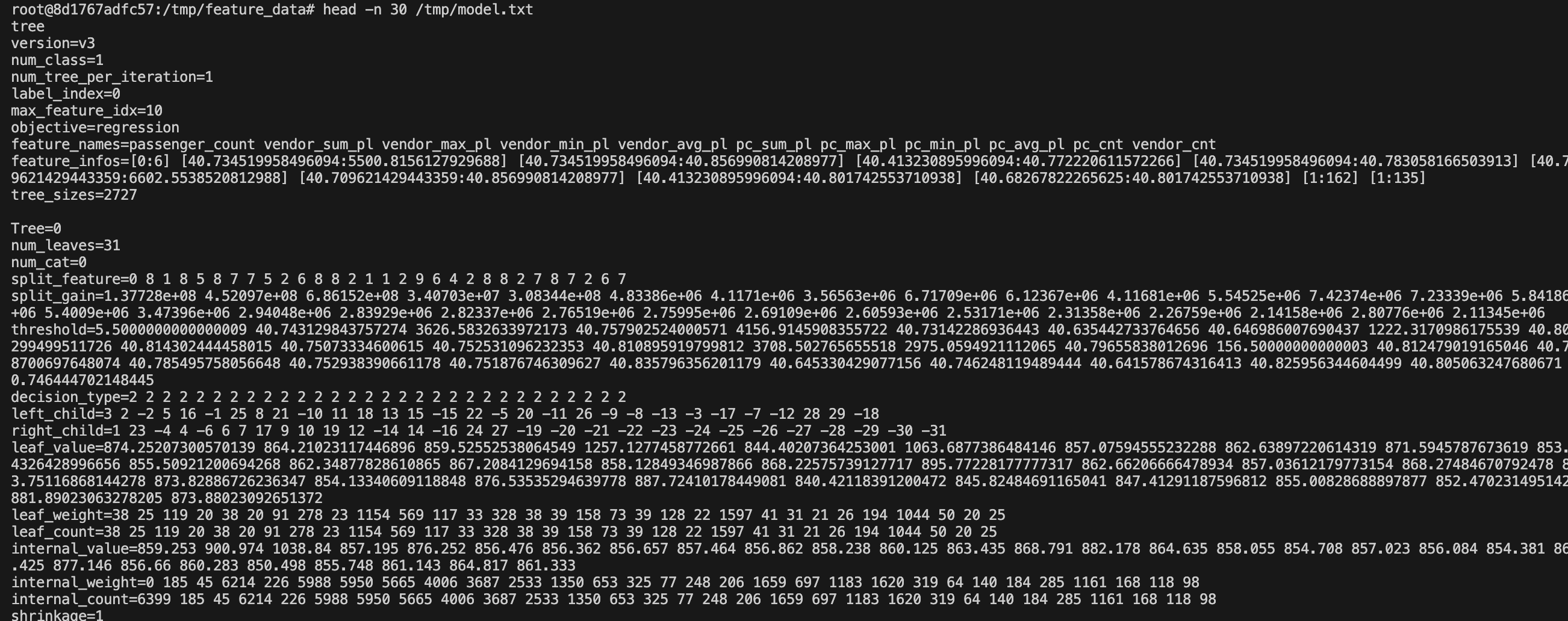
**执行后的结果如下，将特征提取出来（）：**

****

其中，原始数据为 10000 条，计算得到的特征数据也是10000 条。

1. **模型训练**

进行训练，得到模型文件。



1. **部署特征提取脚本**

将特征提取脚本部署作为线上特征提取使用。

1. **在线数据导入**
2. **开启模型服务**
3. **进行预测**

预测请求如下：

预测结果如下：

总结： OpenMLDB 是一个专为时间序列数据设计的开源的分布式内存数据库系统，专注于高性能、可靠性和扩展性，适合处理海量时间序列数据和在线功能的实时计算。

在线数据源： OpenMLDB 可以接受来自于 kafka、Pulsar、RocketMQ 的高速流数据。

离线数据源： OpenMLDB 提供了与 Hive、Amazon S3、Iceberg、TiDB

参考：

[OpenMLDB的实时特征数据库介绍](https://doc.weixin.qq.com/doc/w3_ATUA1AZfANsyvIiVlg8Tv0tnI5uiu?scode=AJEAIQdfAAoinpyuPaATUA1AZfANs)

[OpenMLDB 的可扩展在线区间连接](https://doc.weixin.qq.com/doc/w3_ATUA1AZfANsbGeS1bSjRPi2sCzSTE?scode=AJEAIQdfAAoLnv7LtzATUA1AZfANs)

<https://km.woa.com/articles/show/604009>