首本Druid技术书作者介绍最火实时大数据OLAP技术原理和实践

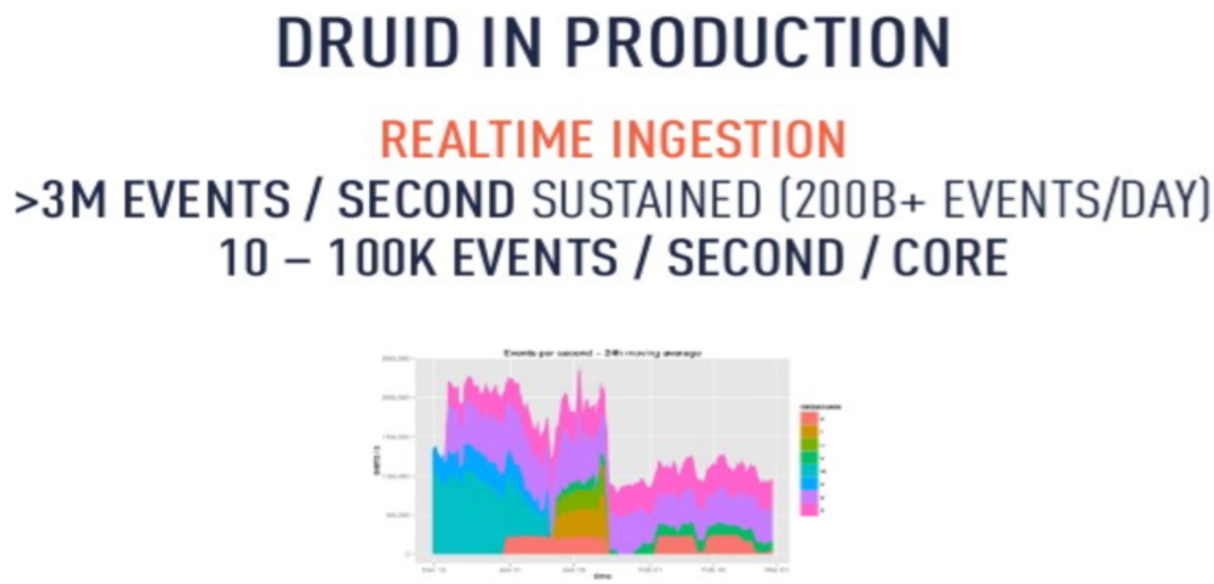
Druid是什么

Druid在大数据领域已经不是新人了，因此可能很多读者都已经听说过Druid，甚至用过Druid，但是未必每个人都真正清晰地了解Druid到底是什么，以及在什么情况下可以用Druid。同时，为了避免大家听了半天，却一直陷在各种细节中但仍然不知道到底在听什么东西，我们还是有必要在开始的时候先总体谈一谈Druid到底是什么。

简单来说，Druid 是一个分布式的、支持实时多维OLAP分析的数据处理系统。它既支持高速的数据实时摄入处理，也支持实时且灵活的多维数据分析查询。因此Druid最常用的场景就是大数据背景下、灵活快速的多维OLAP分析。 另外，Druid还有一个关键的特点：它支持根据时间戳对数据进行预聚合摄入和聚合分析，因此也有用户经常在有时序数据处理分析的场景中用到它。

Druid用户群能够迅速发展的一个原因是它在大数据背景下集群依然具备优秀的性能和可扩展性。Imply.io公司是Druid创始人创办的公司，我们可以先通过它公布的一些集群性能介绍图片来概览一下Druid集群的一些特点。

数据实时消费能力：



集群的数据规模：

DRUID IN PRODUCTION

CLUSTER SIZE

>500TB OF SEGMENTS (>30 TRILLION RAW EVENTS)

>5000 CORES (>350 NODES, >100TB RAM)

IT'S CHEAP

MOST COST EFFECTIVE AT THIS SCALE

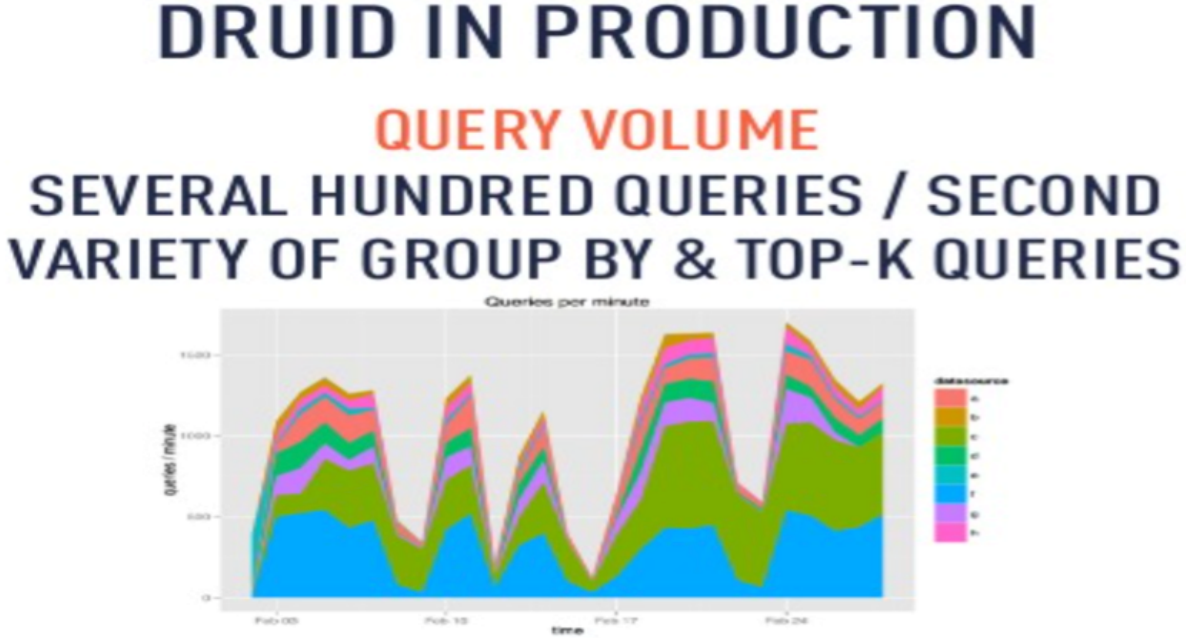
查询速度：

DRUID IN PRODUCTION

QUERY LATENCY (500MS AVERAGE)

90% < 1S 95% < 2S 99% < 10S

查询的并发量：



不难看出，Druid集群在大数据场景下表现优异。

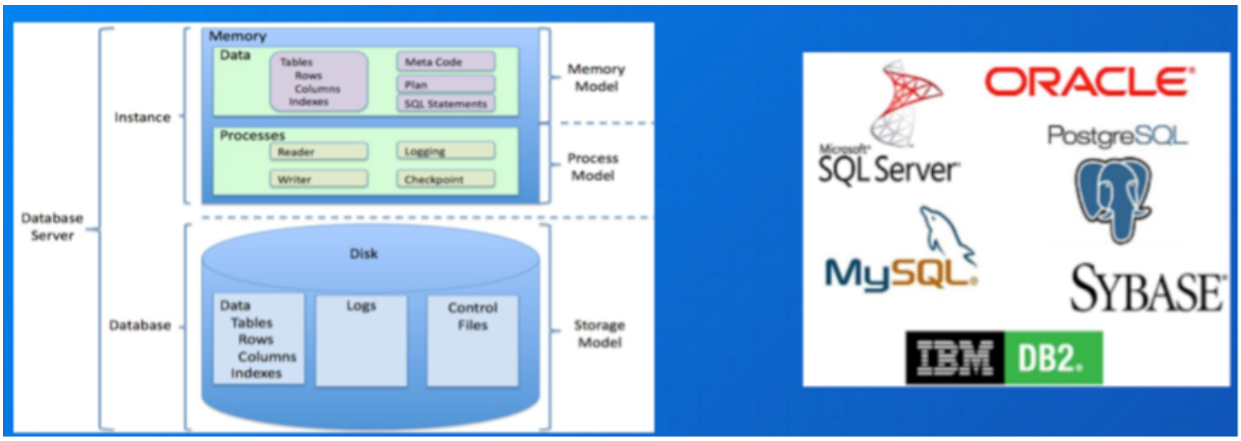
接下来，我们也顺便聊一聊Druid的历史。Druid 单词来源于西方古罗马的神话人物, 中文常常翻译成德鲁伊。传说 Druid 教士精通占卜, 对祭祀礼仪一丝不苟,也擅长于天文历法、医药、天文和文学等。同时, Druid 也 是执法者、吟游诗人和探险家的代名词。美国广告技术公司 MetaMarkets 于 2011 年创建了 Druid 项目, 并且于 2012 年开源了 Druid 项目。Druid 设计之初的想法就是为分析而生,它在处理数据的规模、数据处理的实时性方面, 比传统的 OLAP 系统有了显著的性能改进, 而且拥抱主流的开源生态, 包括 Hadoop 等。多年以来, Druid 一直是非常活跃的开源项目。另外, 阿里巴巴也曾创建过一个开源项目叫作 Druid(简称阿里 Druid), 它是一个数据库连接池的项目。阿里 Druid 和本书讨论的 Druid 没有任何关系,它们解决完全不同的问题。

Druid 的官方网站是 http://druid.io。

技术选型的思考

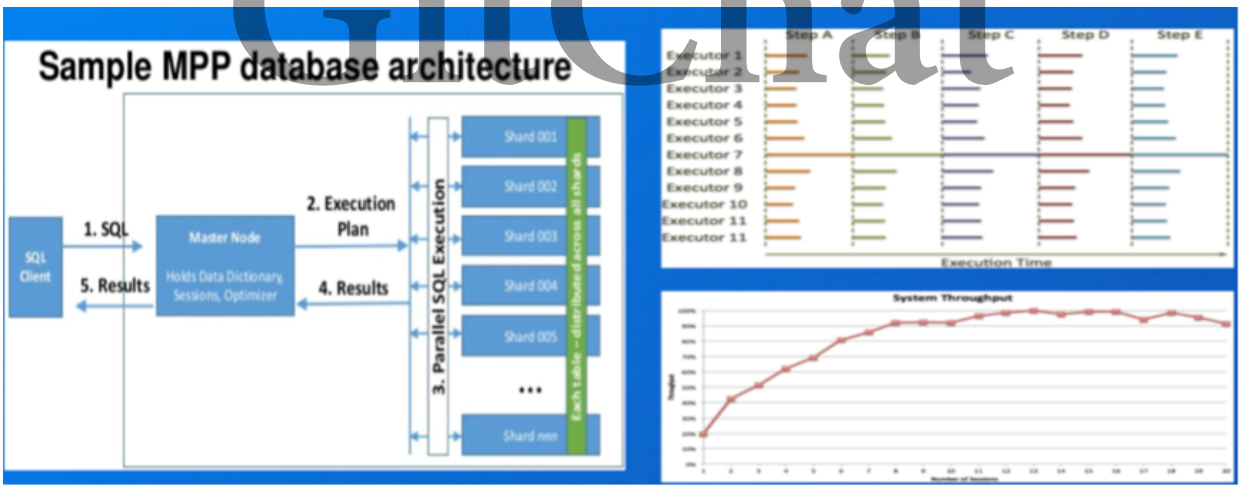
各个公司在技术选型的时候都会货比三家，因此在考察Druid的时候也一定会问为什么需要选择Druid。在本段落中，我们将会对用户经常会拿来和Druid做比较的组件中的其中几个来做一些简单的说明。但是在比较之前，我需要阐明一个基本的观点：这些不同组件或技术有各自的特点和适用范围，并没有一个技术就能彻底替代其它技术这种说法，而本文就仅拿“大数据量下的实时多维OLAP分析”这一场景来做比较。

传统关系型数据库:



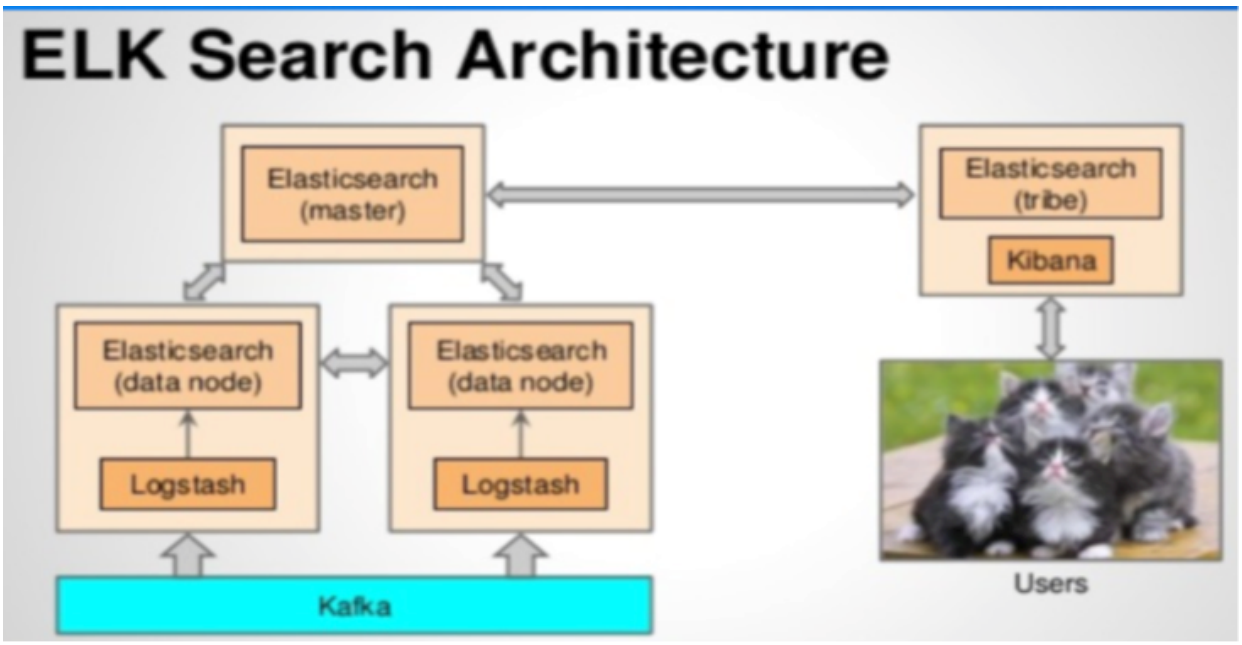
传统关系型数据库（RDBMS）历史久远、技术成熟，应用者广泛，因此采用的学习成本和技术风险也相对较低。典型的RDBMS有Oracle、DB2和MySQL等。从理论上讲，因为RDBMS能保证数据的立即一致性，因此特别适合OLTP的场景。然而由于自身的特点所限，RDBMS在可扩展性方面却表现欠佳，很难轻易和低成本低进行线性扩展以处理更大的数据量。即便现在大多数据库可以通过分库分表等方法来使自己可以容纳更多的数据，但是这种方式也存在着管理复杂和成本高昂等明显短板，因此实际应用中也存在着数据量的天花板。因此，考虑既定的场景“大数据量下的实时多维OLAP分析”，RDBMS首先就很难跨过“大数据量”这一关。

Massively Parallel Processing 数据库:



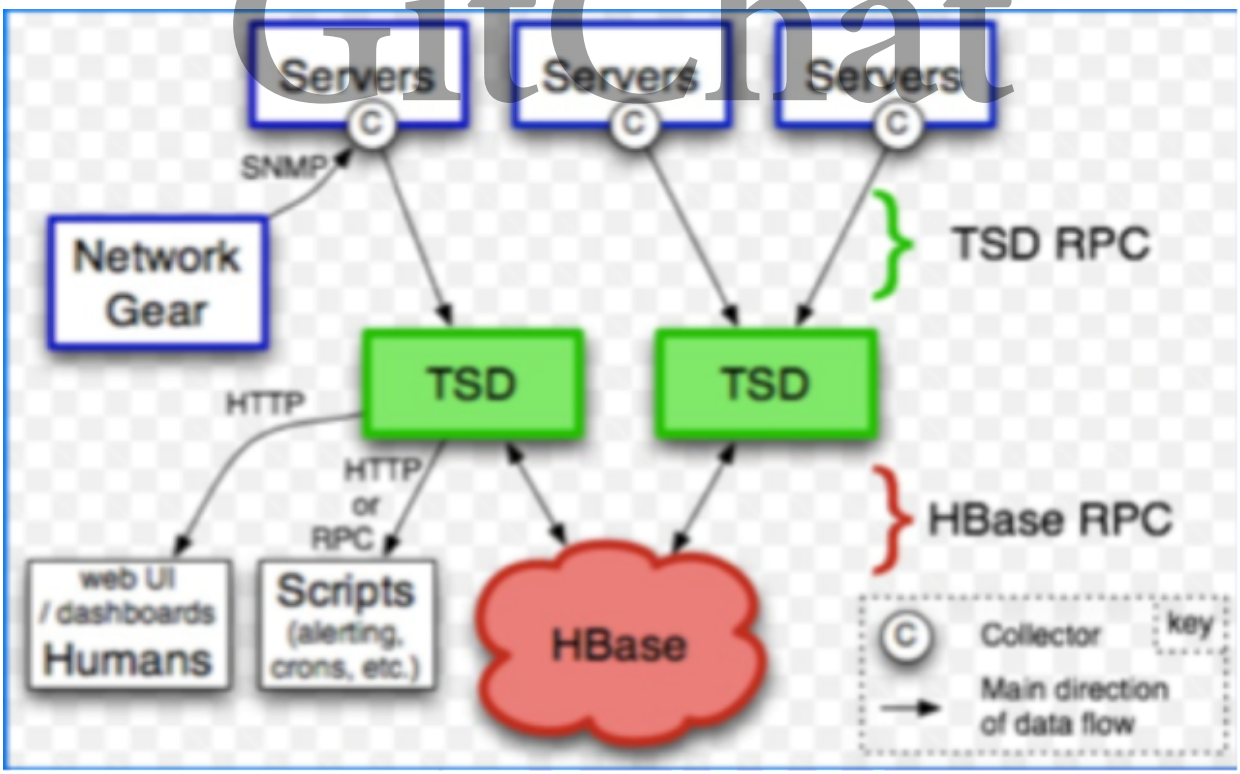
大规模并行处理（MPP，massively parallel processing）数据库的代表是Teradata、Greenplum、Vertica和Impala等。它们的优点是适合将RDBMS的应用扩展到集群范围，以处理更大的数据量，同时继承了RDBMS的很多优点。然而，它依然没法从根本上满足处理海量数据的需求，因为它的系统性能很难随集群的扩展一直线性增长，所以其集群扩展性有限；而且它的集群容错性也有限，比如当其Raid磁盘出错后可能会导致其节点响应速度变慢。

ElasticSearch:



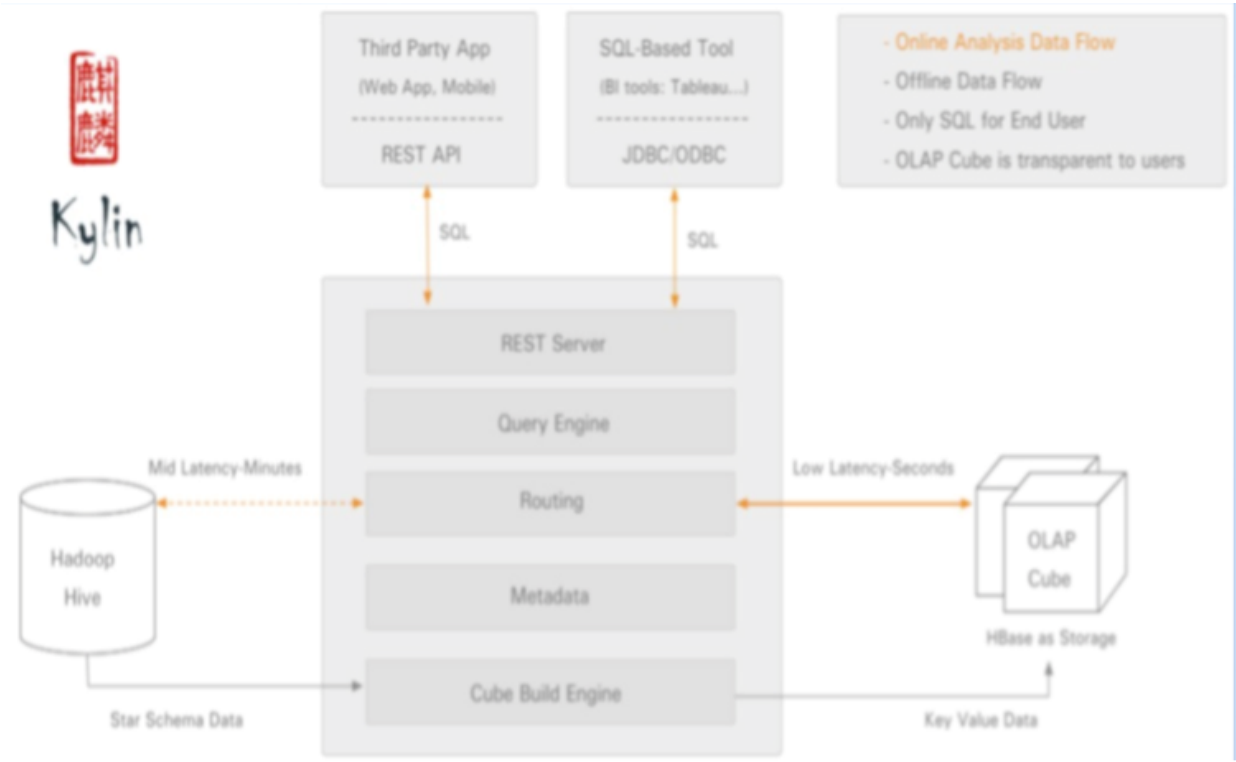
ElasticSearch几乎已经是当下实时上的搜索服务器标准，而它与Logstash和Kibana的搭配（简称ELK）也是应用很广的套件组合。ElasticSearch有很强的文档索引和全文检索的能力，也支持丰富的查询功能（包括聚合查询），并且扩展性好，因此很多用户也常常在拿Elasticsearch和Druid做比较。然而，在我们的既定场景下，Elasticsearch有着其明显的短板：不支持预聚合；组合查询性能欠佳；等等。

OpenTSDB:



OpenTSDB是一款优秀的时序数据库，且基于用户广泛的HBase数据库，因此有着较多的客户群。它的优势在于查询速度快、扩展性好，且schemaless。然而，它也有一些缺点：查询的维度组合数量需要提前确定好，即通过存储中的tag组合来确定，因此缺乏了灵活性；数据冗余度大；基于HBase，对于运维人员能力要求较高；等等。

Kylin:



Kylin出自eBay，并且是Apache的新顶级项目，是一款非常不错的分布式OLAP系统。它的优势在于项目来自于丰富的实践、发展前景光明、直接利用了成熟的Hadoop体系（HBase和Hive等）、cube模型支持较好、查询速度快速等。但它也在一些方面有着自己的局限，比如：查询的维度组合数量需要提前确定好，不适合即席查询分析；预计算量大，资源消耗多；集群依赖较多，如HBase和Hive等，属于重量级方案，因此运维成本也较高；等等。

Druid基础知识

数据存储格式

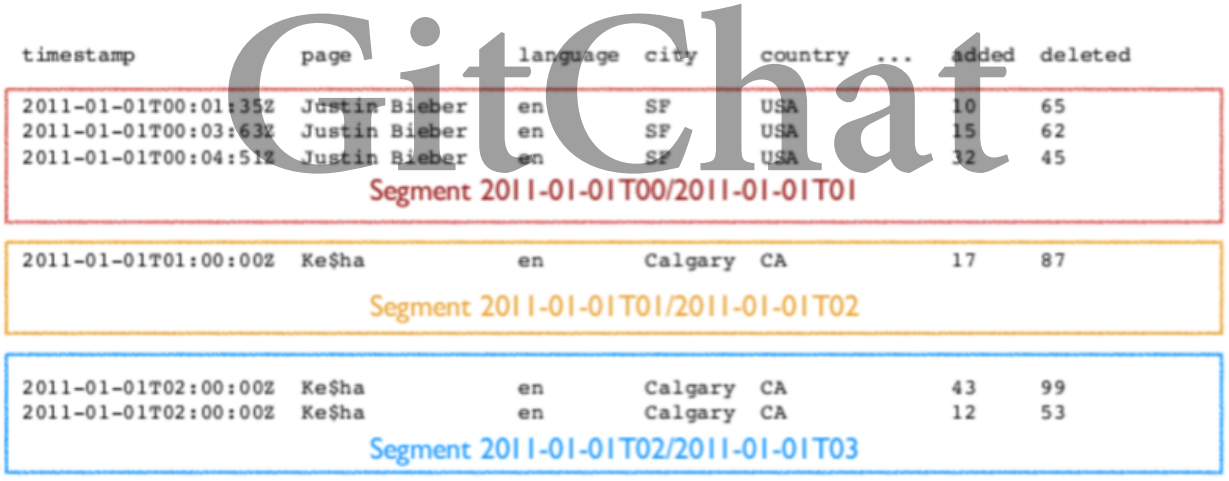
Druid自己有着数据存储的逻辑和格式（主要是DataSource和Segment），而且其架构和实现也是围绕着它们而进行的，因此我们有必要对其数据存储格式做一些介绍。

首先，若与传统的关系型数据库管理系统(RDBMS)做比较,Druid 的 DataSource 可以理解为 RDBMS 中的表(Table)。正如前面章节中所介绍的,DataSource 的结构包含以下几个方面。

- 时间列(TimeStamp): 表明每行数据的时间值，默认使用 UTC 时间格式且精确到毫秒级别。这个列是数据聚合与范围查询的重要维度。
- 维度列(Dimension): 维度来自于OLAP的概念，用来标识数据行的各个类别信息。
- 指标列(Metric): 指标对应于OLAP概念中的Fact，是用于聚合和计算的列这些指标列通常是一些数字，计算操作通常包括 Count、 Sum 和 Mean 等。

无论是实时数据消费还是批量数据处理，Druid 在基于 DataSource 结构存储数据时即可选择对任意的指标列进行聚合(Roll Up)操作。相对于其他时序数据库，Druid 在数据存储时便可对数据进行聚合操作是其一大特点, 并且该特点使得 Druid 不仅能够节省存储空间，而且能够提高聚合查询的效率。

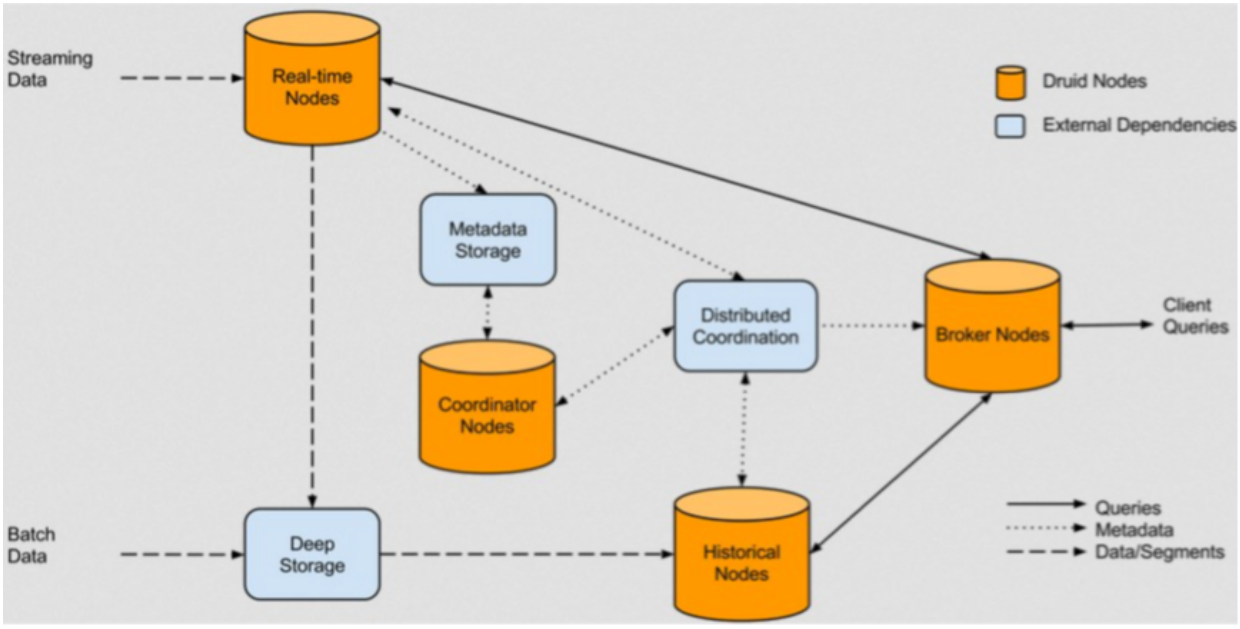
DataSource 是一个逻辑概念，Segment 却是数据的实际物理存储格式，Druid 正是通过 Segment 实现了对数据的横纵向切割(Slice and Dice)操作。从数据按时间分布的角度来看, 通过参数 segmentGranularity 的设置，Druid 将不同时间范围内的数据存储在不同的 Segment 数据块中，这便是所谓的数据横向切割。这种设计为 Druid 带来一个显而易见的优点：按时间范围查询数据时，仅需要访问对应时间段内的这些 Segment 数据块,而不需要进行全表数据范围查询,这使效率得到了极大的提高。



同时,在 Segment 中也面向列进行数据压缩存储，这便是所谓的数据纵向切割。而且对 Segment 中的维度列使用了 Bitmap 技术对其数据的访问进行了优化。其中，Druid会为每一维度列存储所有列值、创建字典（用来存储所有列值对应的ID）以及为每一个列值创建其bitmap索引以帮助快速定位哪些行拥有该列值。

基于实时节点的架构

Druid最早的架构是基于实时节点（Realtime Node）的。它的特点是有专门的实时节点来负责消费数据。



Druid自身包含以下 4 类节点。

- 实时节点(RealtimeNode): 即时摄入实时数据，以及生成Segment数据文件。
- 历史节点(HistoricalNode): 加载已生成好的数据文件，以供数据查询。
- 查询节点(Broker Node): 对外提供数据查询服务，并同时从实时节点与历史节点查询数据,合并后返回给调用方。
- 协调节点(CoordinatorNode): 负责历史节点的数据负载均衡，以及通过规则(Rule) 管理数据的生命周期。

同时，集群还包含以下三类外部依赖。

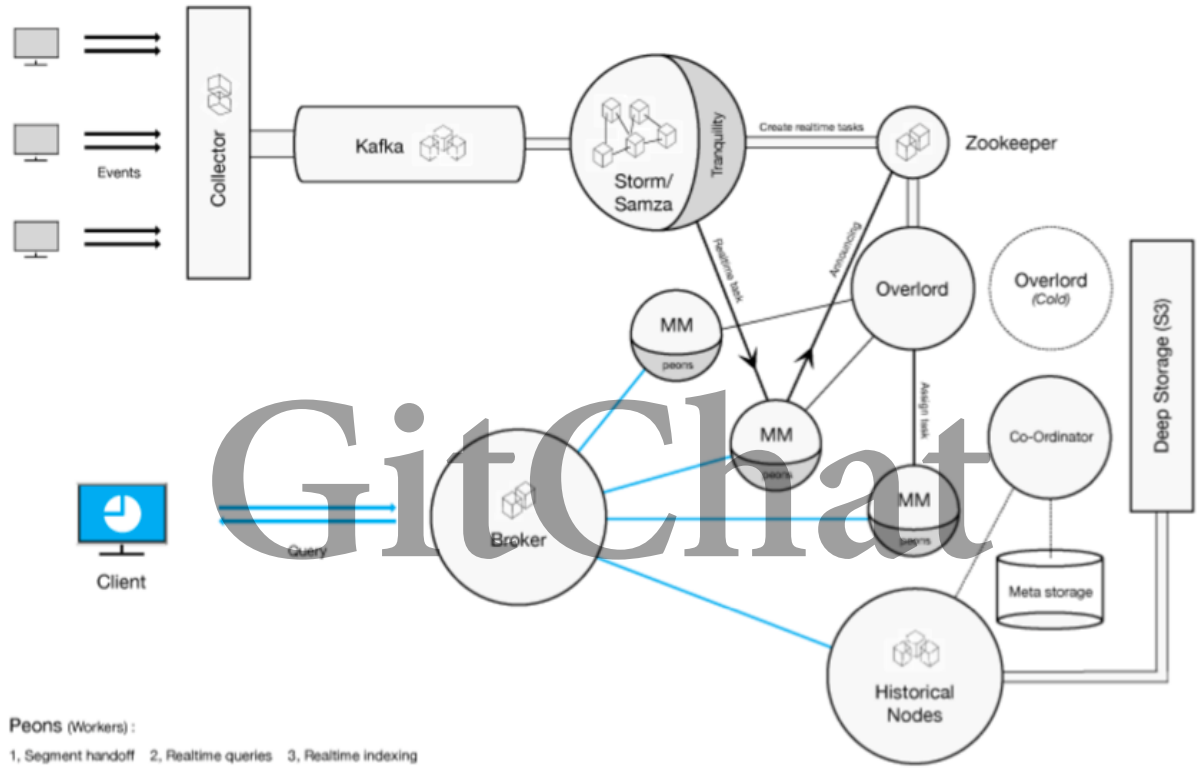
- 元数据库(Metastore): 存储Druid集群的原数据信息，比如Segment的相关信息，一般用 MySQL 或 PostgreSQL。
- 分布式协调服务(Coordination): 为Druid集群提供一致性协调服务的组件，通常为 Zookeeper。
- 数据文件存储库(DeepStorage): 存放生成的Segment数据文件，并供历史节点下载 对于单节点集群可以是本地磁盘，而对于分布式集群一般是 HDFS 或 NFS。

从数据流转的角度来看，数据从架构图的左侧进入系统,分为实时流数据与批量数据。实时流数据会被实时节点消费,然后实时节点将生成的 Segment 数据文件上传到数据文件存储库；而批量数据经过 Druid 集群消费后会被直接上传到数据文件存储库。同时，查询节点会响应外部的查询请求，并将分别从实时节点与历史节点查询到的结果合并后返回。

基于Indexing Service的架构

在基于Indexing Service的架构中，实时节点被废弃掉了，转而代之的Overlord节点和MiddleManager节点。其中，Overlord节点作为索引服务的主节点,对外负责接收任务请求,对内负责将任务分解并下发到从节点即MiddleManager节点上。而MiddleManager节点就是索引服务的工作节点,负责接收统治节点分配的任务,然后启动相关苦工（Peon）即独立的 JVM 来完成具体的任务。这样的架构实际与 Hadoop Yarn 很像。

较之基于实时节点的架构，基于Indexing Service的架构有不少优点：除了对数据能够用 pull 的方式外,还支持 push 的方式；不同于手工编写数据消费配置文件的方式，可以通过 API 的编程方式来灵活定义任务配置；可以更灵活地管理与使用系统资源；可以完成 Segment 副本数量的控制；能够灵活完成跟 Segment 数据文件相关的所有操作，如合并、删除 Segment 数据文件等。



查询

Druid的默认提供了HTTP REST 风格的查询接口。一个典型的 curl 命令如下:
curl -X POST ':/druid/v2/?pretty' -H 'Content-Type:application /json' -d @ 。

其中，queryable_host:port 为查询节点的 IP 地址和端口；query_json_ le为POST到查询节点的查询请求。

为了便于用户进行灵活的OLAP分析，Druid提供了丰富的查询接口，比如过滤器、聚合器和后置聚合器等。由于这部分内容比较好理解，而且Druid官网上有详细的介绍，因此本文就不再过多说明，这里仅聚一个实际的查询例子如下：

```
{
    "queryType": "timeseries",
    "dataSource": "visitor_statistics",
```

```
    "granularity": "all",
    "filter": {
        "type": "and",
        "fields": [
            {
                "type": "selector",
                "dimension": "host",
                "value": "www.mejia.wang"
            },
            {

            }
        ]
},
"aggregations": [
    {
        "type": "longSum",
        "name": "pv",
        "fieldName": "counbt"
    },
    {

    },
    {
        "type": "hyperUnique",
        "name": "new_visitor_count",
        "fieldName": "new_visit_count"
    },
    {

    }
],
"postAggregations": [
    {
        "type": "arithmetic",
        "name": "new_visitor_rate",
        "fn": "/",
        "fields": [
            {
                "type": "hyperUniqueCardinality",
                "fieldName": "new_visitor_count"
            },
            {
                "fieldName": "visitor_count"
            }
        ]
    },
    {
        "type": "arithmetic",
        "name": "click_rate",
        "fn": "/",
        "fields": [
```

```
            {
                "type": "hyperUniqueCardinality",
                "fieldName": "click_visitor_count"
            },
            {
                "type": "hyperUnique",
                "name": "click_visitor_count",
                "fieldName": "click_visit_count""type":
"hyperUniqueCardinality",
                "type": "hyperUniqueCardinality",
                "fieldName": "visitor_count"
            }
        ]
    }
],
"intervals": [
    "2016-08-07T00:00:00+08:00/2016-09-05T23:59:59+08:00"
]
}
```

返回结果可能如下：

```
[
    {
        "timestamp": "2016-08-27T16:00:00.000Z",
        "result": {
            "pv": 30.000000061295435,
            "visit_count": 5.006113467958146,
            "new_visitor_count": 0,
            "click_visitor_count": 5.006113467958146,
            "new_visitor_rate": 0,
            "click_rate": 1,

    }
}
]
```


丰富的辅助功能

Druid除了基本的核心数据消费和查询功能外，还提供了丰富的辅助功能，以帮助用户更好地基于Druid完成数据处理工作。本文简单列举几个：

DataSketches aggregator：近似计算COUNT DISTINCT等问题，如PV、留存等；精度可调节。

Multi-value dimensions：对于同一维度列，允许不同行拥有不同数量的数据值：这使得Druid也能够有类似schemaless的功能。

{“timestamp”: “2011-01-12T00:00:00.000Z”, “tags”: [“t1”,”t2”,”t3”]} #row1
{“timestamp”: “2011-01-13T00:00:00.000Z”, “tags”: [“t3”,”t4”,”t5”]} #row2

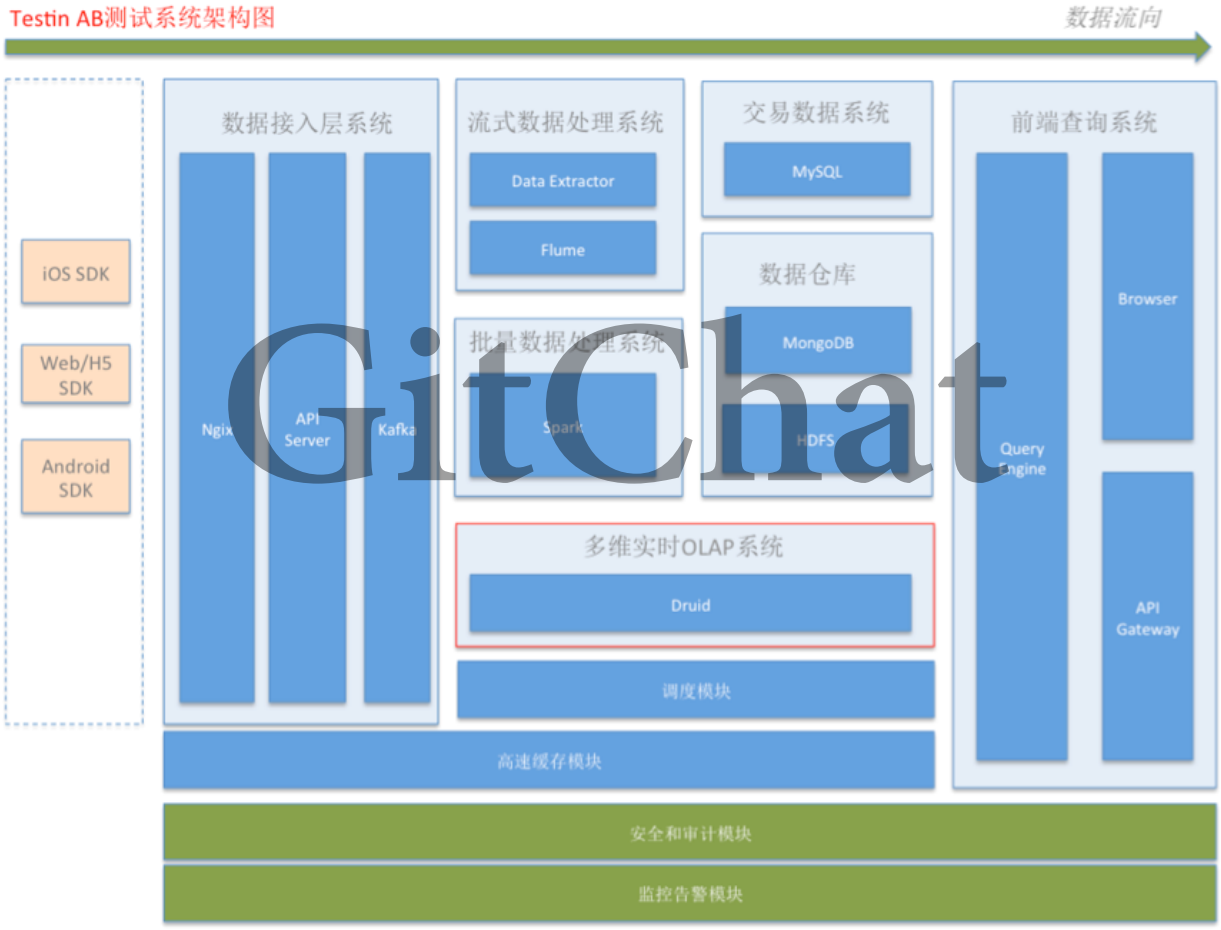
Router Node：协助Broker Node实现负载均衡。

Query Caching：帮助提升查询性能，可以用在Historical Node或Broker Node上。

实践

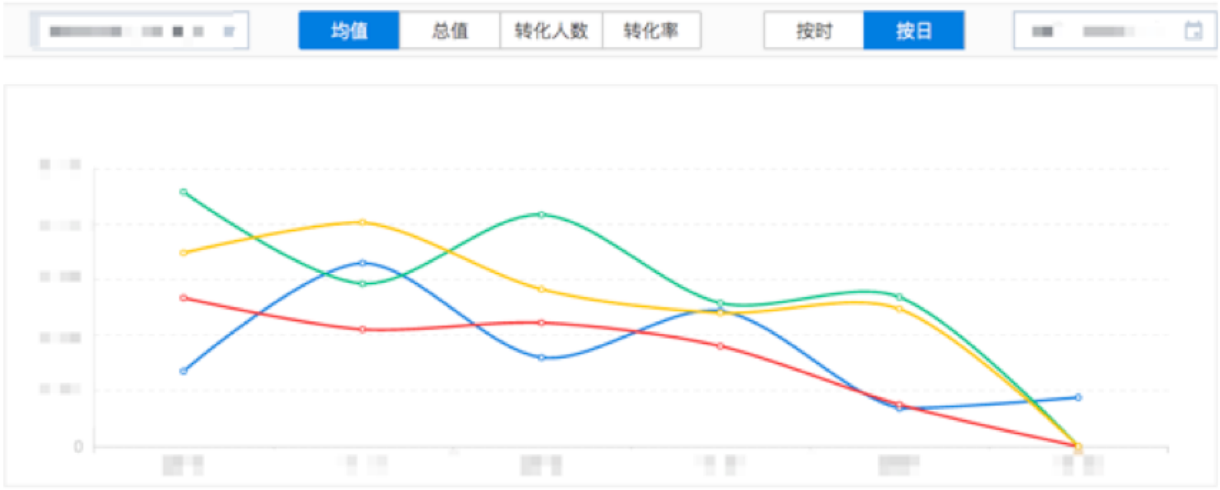
最后，介绍一下笔者目前在北京云测信息技术有限公司（即Testin公司）的AB测试产品中的Druid实践经验。

下图是我们AB测试产品的架构图，可以看到Druid在其中主要负责多维实时OLAP分析。



我们使用的是Druid Indexing Service的架构，而且使用了Kafka Indexing Service的功能，这使得我们可以很方便地保证了Druid不丢弃超过时间窗口的数据。

从使用效果来看，Druid赋予了我们产品实时的数据消费和查询，使得用户可以在秒级便看到最新的数据分析结果，如下图：



同时，Druid使得我们可以进行灵活的多维实时分析：

引用

本文的一些素材和语言段落摘录自笔者所著的《Druid实时大数据分析原理与实践》一书。


Broadview
精通实时大数据分析
Druid
实时大数据分析
原理与实践
欧阳辰 刘麒赟 张海雷 高振源 等著
腾讯、小米、优酷、云测等互联网公司的一线实践经验
为你解读海量实时OLAP平台
电子工业出版社
中国工信出版集团