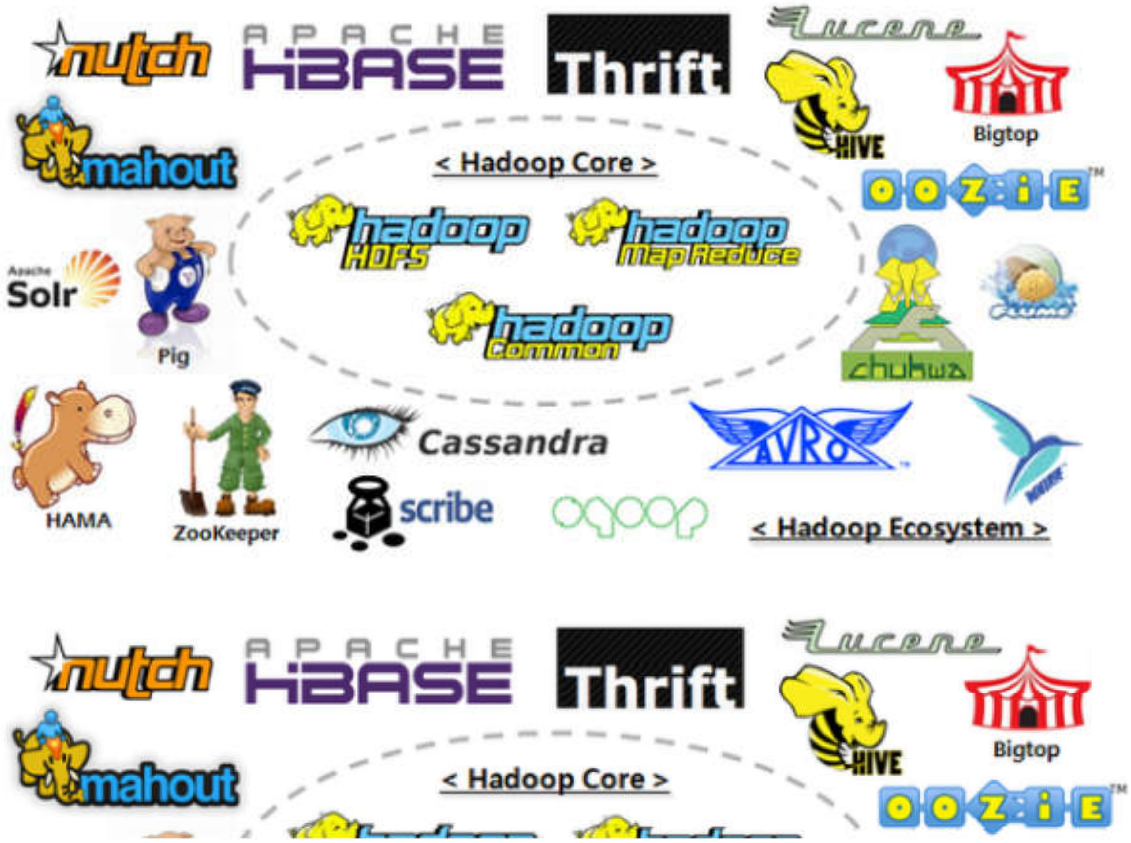
dicksonjin

浅谈开源大数据平台的演变

3年前 408

浅谈开源大数据平台的演变

2015-04-16 腾讯大数据



一说到开源大数据处理平台，就不得不说此领域的开山鼻祖**Hadoop**，它是**GFS**和**MapReduce**的开源实现。虽然在此之前有很多类似的分布式存储和计算平台，但真正能实现工业级应用、降低使用门槛、带动业界大规模部署的就是**Hadoop**。得益于**MapReduce**框架的易用性和容错性，以及

同时包含存储系统和计算系统，使得**Hadoop**成为大数据处理平台的基石之一。**Hadoop**能够满足大部分的离线存储和离线计算需求，且性能表现不俗；小部分离线存储和计算需求，在对性能要求不高的情况下，也可以使用**Hadoop**实现。因此，在搭建大数据处理平台的初期，**Hadoop**能满足**90%**以上的离线存储和离线计算需求，成为了各大公司初期平台的首选。

随着Hadoop集群越来越大，单点的namenode渐渐成为了问题：第一个问题是

单机内存有限，承载不了越来越多的文件数目；第二个问题是单点故障，严重影响集群的高可用性。因此业界出现了几种分布式namenode的方案，用以解决单点问题。此外，为了实现多种计算框架可以运行在同一个集群中，充分复用机器资源，Hadoop引进了YARN。YARN是一个通用资源管理器，负责资源调度和资源隔离。它试图成为各个计算框架的统一资源管理中心，使得同一个集群可以同时跑MapReduce、storm、Tez等实例。

Hadoop解决了大数据平台的有无问题，随着业务和需求的精细化发展，在一些细分领域人们对大数据平台提出了更高的期望和要求，因此诞生了一批在不同领域下的更高效更有针对性的平台。首先基于对Hadoop框架自身的改良，出现了haloop和dryad等变种平台，不过这些平台后来基本上都没有被大规

模部署，其原因要么是改良效果不明显，要么是被跳出Hadoop框架重新设计的新平台所取代了。

为了解决在hadoop平台上更好地进行海量网页分析，进而实现通用的分布式NoSQL数据库的问题，HBase诞生了。Hadoop参照了Google的GFS和MapReduce的设计。而Google的BigTable在Hadoop的生态圈里对应的则是HBase。HBase丰富了Hadoop的存储方式，在hdfs的文件式存储的基础上，提供了表格式存储，使得可以将网页的众多属性提取出来按字段存放，提升网页查询分析的效率。同时，**HBase也广泛被用作通用的NoSQL存储系统**，它是基于列存储的非关系型数据库，弥补了hdfs在随机读写方面的不足，提供低延时的数据访问能力。但HBase本身没有提供脚本语言式（如SQL）的数据访问方式，为了克服数据访问的不便捷问

题，最开始用于Hadoop的PIG语言开始支持HBase。PIG是一种操作Hadoop和Hbase的轻量级脚本语言，不想编写MapReduce作业的人员可以用PIG方便地访问数据。

跟HBase类似的另一个较为有名的系统是**C++编写的Hypertable**，也是BigTable的开源实现，不过由于后来维护的人员越来越少，以及Hadoop生态系统越来越活跃，渐渐地Hypertable被人们遗忘了。还有一个不得不提的系统是Cassandra，它最初由Facebook开发，也是一个分布式的NoSQL数据库。但与HBase和Hypertable是Bigtable的复制者不同，Cassandra结合了Amazon的Dynamo的存储模型和Bigtable的数据模型。它的一大特点是使用Gossip协议实现了去中心化的P2P存储方式，所有服务器都是等价的，不存在任何一个单点问题。Cassandra与HBase的区别在于：**Cassandra配置简**

单，平台组件少，集群化部署和运维较容易，CAP定理侧重于Availability和Partition tolerance，不提供行锁，不适合存储超大文件；**HBase配置相对复杂，平台组件多**，集群化部署和运维稍微麻烦，CAP定理侧重于Consistency和Availability，提供行锁，可处理超大文件。

虽然Hadoop的MapReduce框架足够易用，但是对于传统使用SQL操作的数据仓库类需求时，直接调用Map和Reduce接口来达到类似效果，还是相对繁琐，而且对不熟悉MapReduce框架的使用者来说是一个门槛，因此hive就是为了解决此问题而诞生。它在Hadoop上建立了一个数据仓库框架，可以将结构化的数据文件映射成一张数据库表，并提供类似SQL的查询接口，弥补了Hadoop和数据仓库操作的鸿沟，大大提高了数据查询和展示类业务的生产效率。一方面，熟悉

SQL的使用者只需要很小的成本就可以迁移至hive平台，另一方面，由于量级大而在传统数据仓库架构下已无法存放的数据，也可以较为容易地迁移到hive平台。因此hive平台已经成为了很多公司的大数据仓库的核心方案。

Hive跟hbase在功能上也有小部分重叠的地方，它们的主要区别是：Hbase本质是一个数据库，提供在存储层的低延时数据读写能力，可用在实时场景，但没有提供类SQL语言的查询方式，所以数据查询和计算不太方便（PIG学习成本较高）；hive本质是将SQL语句映射成MapReduce作业，延时较高但使用方便，适合离线场景，自身不做存储。此外，hive可以搭建在Hbase之上，访问Hbase的数据。

Hive的出现桥接了Hadoop与数据仓库领

域，但随着hive的逐步应用，人们发现hive的效率并不是太高，原因是hive的查询是使用MapReduce作业的方式实现的，是在计算层而不是存储层，因此受到了MapReduce框架单一的数据传输和交互方式的局限、以及作业调度开销的影响。为了让基于Hadoop的数据仓库操作效率更高，在hive之后出现了另一个不同的实现方案——impala，它的基于Hadoop的数据查询操作，并不是使用MapReduce作业的方式实现，而是跳过了Hadoop的计算层，直接读写hadoop的存储层——hdfs来实现。由于省去了计算层，因此也就省去了计算层所有的开销，避免了计算层的单一数据交互方式的问题，以及多轮计算之间的磁盘IO问题。直接读写hdfs，可以实现更加灵活的数据交互方式，提高读写效率。它实现了嵌套型数据的列存储，同时采用了多层查询树，使得它可以在数千节点中快速地并行执行查询与结果聚合。据一些公开的资料显示，impala在各个场景下的效

率可以比hive提升3~68倍，尤其在某些特殊场景下的效率提升甚至可达90倍。

Hadoop极大降低了海量数据计算能力的门槛，使得各个业务都可以快速使用Hadoop进行大数据分析，随着分析计算的不断深入，差异化的需求慢慢浮现了。人们开始发现，某些计算，如果时效性更快，收益会变得更大，能提供给用户更好的体验。一开始，在Hadoop平台上为了提高时效性，往往会将一整批计算的海量数据，切割成小时级数据，甚至亚小时级数据，从而变成相对轻量的计算任务，使得在Hadoop上可以较快地计算出当前片段的结果，再把当前片段结果跟之前的累积结果进行合并，就可以较快地得出当前所需的整体结果，实现较高的时效性。但随着互联网行业竞争越来越激烈，对时效性越来越看重，尤其是实时分析统计的需求大量涌现，分钟级甚至秒级输出结果，是大家

所期望的。hadoop计算的时效性所能达到的极限一般为10分钟左右，受限于集群负载和调度策略，要想持续稳定地低于10分钟是非常困难的，除非是专用集群。因此，为了实现更高的时效性，在分钟级、秒级、甚至毫秒级内计算出结果，Storm应运而生，它完全摆脱了MapReduce架构，重新设计了一个适用于流式计算的架构，以数据流为驱动，触发计算，因此每来一条数据，就可以产生一次计算结果，时效性非常高，一般可以达到秒级。而且它的有向无环图计算拓扑的设计，提供了非常灵活丰富的计算方式，覆盖了常见的实时计算需求，因此在业界得到了大量的部署应用。

Storm的核心框架保证数据流可靠性方式是：每条数据会被至少发送一次，即正常情况会发送一次，异常情况会重发。这样会导致中间处理逻辑有可能会收到两条重复的数据。

大多数业务中这样不会带来额外的问题，或者是能够容忍这样的误差，但对于有严格事务性要求的业务，则会出现问题，例如扣钱重复扣了两次这是用户不可接受的。为了解决此问题，Storm引入了事务拓扑，实现了精确处理一次的语义，后来被新的Trident机制所取代。Trident同时还提供了实时数据的join、groupby、filter等聚合查询操作。

跟storm类似的系统还有yahoo的S4，不过storm的用户远远多于S4，因此storm的发展比较迅速，功能也更加完善。

随着大数据平台的逐步普及，人们不再满足于如数据统计、数据关联等简单的挖掘，渐渐开始尝试将机器学习/模式识别的算法用于海量数据的深度挖掘中。因为机器学习/模式识别的算法往往比较复杂，属于计算密集型的算法，且是单机算法，所以在没有Hadoop

之前，将这些算法用于海量数据上几乎是不可行，至少是工业应用上不可行：一是单机计算不了如此大量的数据；二是就算单机能够支撑，但计算时间太长，通常一次计算耗时从几个星期到几个月不等，这对于工业界来说资源和时间的消耗不可接受；三是没有一个很易用的并行计算平台，可以将单机算法快速改成并行算法，导致算法的并行化成本很高。而有了Hadoop之后，这些问题迎刃而解，一大批机器学习/模式识别的算法得以快速用MapReduce框架并行化，被广泛用在搜索、广告、自然语言处理、个性化推荐、安全等业务中。

那么问题来了，上述的机器学习/模式识别算法往往都是迭代型的计算，一般会迭代几十至几百轮，那么在Hadoop上就是连续的几十至几百个串行的任务，前后两个任务之间都要经过大量的IO来传递数据，据不完全统

计，多数的迭代型算法在Hadoop上的耗时，IO占了80%左右，如果可以省掉这些IO开销，那么对计算速度的提升将是巨大的，因此业界兴起了一股基于内存计算的潮流，而Spark则是这方面的佼佼者。它提出了RDD的概念，通过对RDD的使用将每轮的计算结果分布式地放在内存中，下一轮直接从内存中读取上一轮的数据，节省了大量的IO开销。同时它提供了比Hadoop的MapReduce方式更加丰富的数据操作方式，有些需要分解成几轮的Hadoop操作，可在Spark里一轮实现。因此对于机器学习/模式识别等迭代型计算，比**起Hadoop平台，在Spark上的计算速度往往会有几倍到几十倍的提升**。另一方面，Spark的设计初衷就是想兼顾MapReduce模式和迭代型计算，因此老的MapReduce计算也可以迁移至spark平台。由于Spark对Hadoop计算的兼容，以及对迭代型计算的优异表现，成熟之后的Spark平台得到迅速的普及。

人们逐渐发现，Spark所具有的优点，可以扩展到更多的领域，现在Spark已经向通用多功能大数据平台的方向迈进。为了让Spark可以用在数据仓库领域，开发者们推出了Shark，它在Spark的框架上提供了类SQL查询接口，与Hive QL完全兼容，但最近被用户体验更好的Spark SQL所取代。**Spark SQL涵盖了Shark的所有特性，并能够加速现有Hive数据的查询分析，以及支持直接对原生RDD对象进行关系查询，显著降低了使用门槛。**在实时计算领域，Spark streaming项目构建了Spark上的实时计算框架，它将数据流切分成小的时间片段（例如几秒），批量执行。得益于Spark的内存计算模式和低延时执行引擎，在Hadoop上做不到的实时计算，在Spark上变得可行。虽然时效性比专门的实时处理系统有一点差距，但也可用于不少实时/准实时场景。另外Spark上还有图模型领域的Bagel，其实就是Google的Pregel在Spark上

的实现。它提供基于图的计算模式，后来被新的Spark图模型API——GraphX所替代。

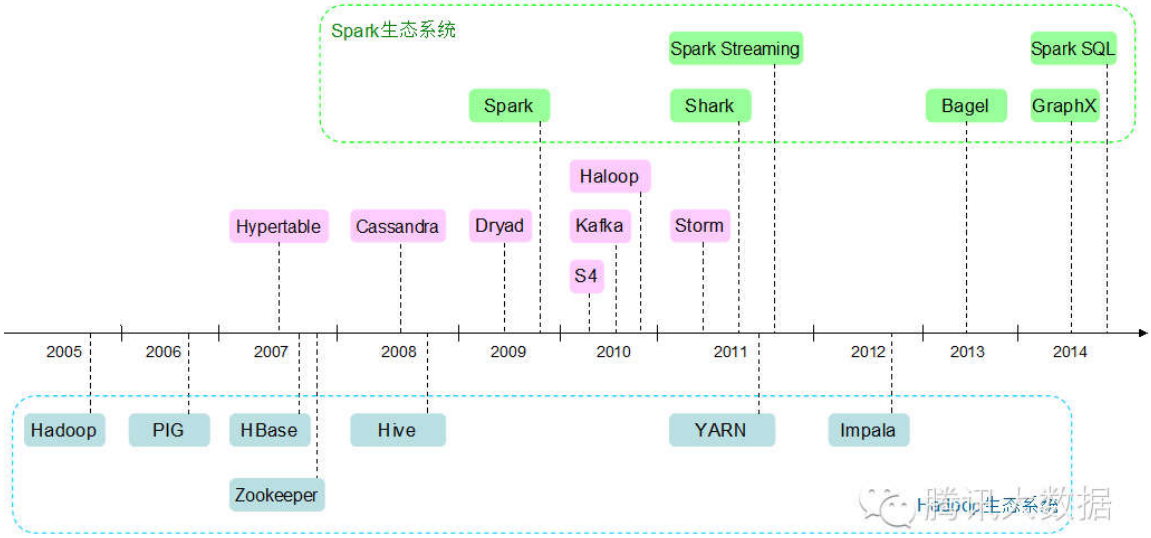
当大数据集群越来越大，出现局部故障的概率也越来越高，集群核心数据的分布式一致性变得越来越难保证。Zookeeper的出现很好地解决了这个棘手的问题，它实现了著名的Fast Paxos算法，提供了一个集群化的分布式一致性服务，使得其他平台和应用可以通过简单地调用它的服务来实现数据的分布式一致性，不需要自己关心具体的实现细节，使大数据平台开发人员可以将精力更加集中在平台自身特性上。例如**Storm平台就是使用Zookeeper来存储集群元信息**（如节点信息、状态信息、任务信息等），从而可以简单高效地实现容错机制。即使某个组件出现故障，新的替代者可以快速地在Zookeeper上注册以及获取所需的元信息，从而恢复失败的任务。除了分布式一致性以

外，Zookeeper还可以用作leader选取、热备切换、资源定位、分布式锁、配置管理等场景。

数据在其生命周期是流动的，往往会有产生、收集、存储、计算、销毁等不同状态，数据可以在不同状态之间流动，也可以从同一个状态的内部进行流动（例如计算状态），流动时上下游的载体有很多种，例如终端、线上日志服务器、存储集群、计算集群等。在后台，多数情况下都是在大数据平台之间流动，因此各个大数据平台并不是孤立的，处理数据时，它们往往成为上下游的关系，而数据从上游流往下游，就需要一个数据管道，来正确连接每份数据正确的上下游，这个场景的需求可以使用Kafka集群来很好地解决。Kafka最初是由LinkedIn开发的，是一个分布式的消息发布与订阅系统。Kafka集群可以充当一个大数据管道的角色，负责

正确连接每种数据的上下游。各个上游产生的数据都发往Kafka集群，而下游则通过向Kafka集群订阅的方式，灵活选择自己所需的上游数据。Kafka支持多个下游订阅同一个上游数据。当上游产生了数据，Kafka会把数据进行一定时间窗口内的持久化，等待下游来读取数据。Kafka的数据持久化方式及内部容错机制，提供了较好的数据可靠性，它同时适合于离线和在线消息消费。

以上平台的诞生时间点如下图所示：



大数据平台极大地提高了业界的生产力，使得海量数据的存储、计算变得更加容易和高

效。通过这些平台的使用，可以快速搭建出一个承载海量用户的应用，移动互联网正是在它们的催化下不断高速发展，改变我们的生活。

分布式 大数据 算法 hbase hadoop 数据仓库 集群 spark 开源大数据 Hive Mapreduce 存储 海量数据

作者

dicksonjin

TA的文章

上海往事之与Allen Zhao小聚

上海往事之与Andy Tang小聚

P3 项目轶事之面试

相关文章

浅谈开源大数据平台的演变

浅谈web网站架构演变过程

浅谈web网站架构演变过程（转）

架构方面的资料集锦

浅谈光伏与农业、工业、商业的结合