13 | 云上大数据：云计算遇上大数据，为什么堪称天作之合？

2020-04-01 何恺铎 来自北京

《深入浅出云计算》

你好，我是何恺铎。

今天我们来讨论和学习云计算中的大数据产品与技术，这也是我自己最喜爱的话题之一。

我们都知道，云计算以存储、计算规模和弹性著称，而大数据方面的业务需求，恰恰需要大量的存储，和呼之即来的澎湃算力。所以，云可以说是最适合运行大数据工作负载的平台了。同时，云计算时代数据规模空前扩大，因此大数据也成为了云上最需要解决的重要场景之一。

正因为两者的关系如此紧密，又几乎处于同一个时代，以至于早年有一段时间，很多开发者产生了概念上的混淆，把“云计算”一词当作大数据技术的代称。但事实并非如此，你需要注意甄别。

在当今的技术语言体系中，我们应该这样来理解：**大数据主要是技术手段，是一系列处理海量数据的方法论和技术实现的总称；而云是一种资源和能力的载体，也是一种商业存在，是可以**

运行大数据负载和应用的平台。

举个例子来形容两者的区别：你可以把云比作一艘航空母舰，是一个大型综合作战平台，而大数据呢，就好比战斗机这个武器门类，在航母上就成了舰载机，依托航母可以达到更大的作战纵深和更强的投递能力。

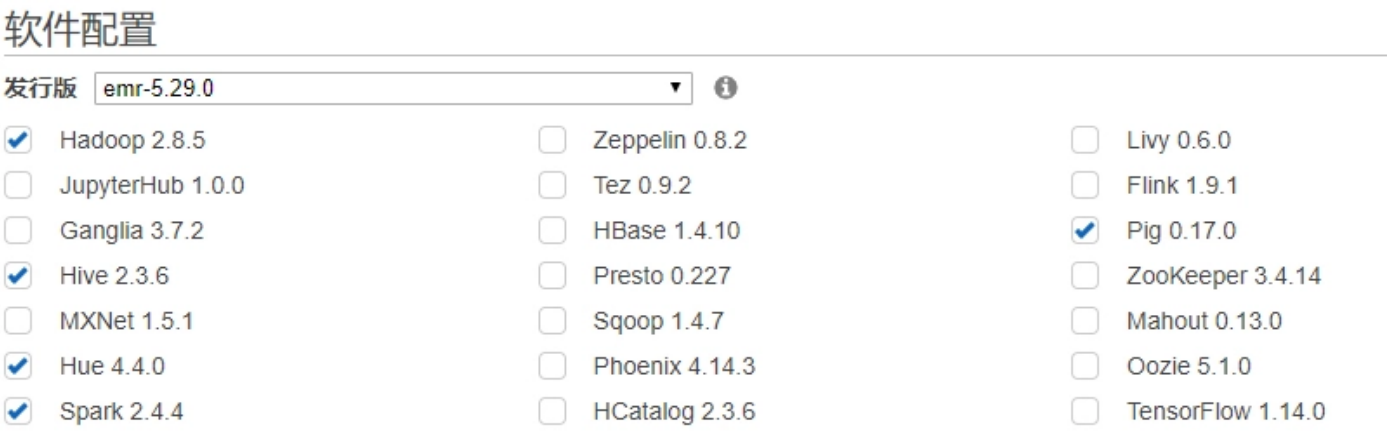
当大数据和云计算交汇，就自然诞生了**云上的大数据 PaaS 产品**，它们是云对大数据技术进行了封装和产品化的成果，也正是我们这一讲的主角。

云上大数据的体验

和其他 PaaS 服务类似，云上大数据的发展，也是从对经典大数据技术的封装开始的。这从一些相关服务的命名中就可以看出来，比如 AWS 早在 2009 年，就发布的大数据服务 **Elastic MapReduce**（简称 EMR）。虽然 EMR 的服务能力早已不限于 MapReduce 这个早期技术，但这个名字一直被保留了下来，这是时间给它打上的烙印。

从技术上讲，云上大数据服务其实一直在迅速跟进着社区的发展，也在不断地延伸自己的服务范围。从早期的 MapReduce，到 Hive 和 HBase，再到后来异军突起的 Spark，都逐渐地纳入到了它的服务体系，现在可以说已经拓展到了整个 Hadoop 和开源大数据的生态。

所以，你在云上能够找到的，不是一个单体的技术组件，而是一系列的大数据服务的组合。下图就展示了 AWS EMR 服务的 5.29.0 版本中，内置支持的丰富组件：



注：从另一个角度来讲，云上大数据服务对开源社区新版本跟进的速度快不快、版本细不细，也是我们评估衡量云服务的一个重要指标。

云上大数据服务最大的特点就是**简便易用，方便管理**。尤其是，我们可以把一个之前可能需要几天时间，来进行安装部署和环境设置的创建集群任务，缩减到短短几分钟，大大降低了我们学习和应用大数据技术的门槛。

说得好不如做得好，接下来我就带你进入**实验环节**，来帮助你对大数据服务形成一个直观的认识。我们要使用国际版 AWS 中的 EMR 服务，运行一段 Spark 程序，来统计小说《双城记》中的单词词频。

首先，我们需要创建一个 EMR 的集群实例。在创建过程中，除了像上面那样指定所需要的组件以外，最重要的一步是**选择集群的配置**：

我选择了 **m5.xlarge** 这个型号的虚拟机，作为底层基础设施的机器。细心的你可能还发现了，**用于执行计算任务的机器组，在这里你可以选择价格相当低廉的竞价实例**。这是一个很有用的节省成本的特性，因为很多时候，大数据服务是用于后台离线计算的，我们可以牺牲一些可用性上的要求。

根据向导，等集群创建完成进入运行状态后，就可以使用了。按照产品提示，我们使用 SSH 就能够成功地连上集群的主节点：

```
client@clientVM:~$ ssh -i ~/awskvpair-california-north.pem hadoop@ec2-52-53-xxx-x
Last login: Sat Feb 15 13:10:50 2020

       __|  __|_  )
```

```
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/

EEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR

[hadoop@ip-172-31-1-100 ~]$
```

小提示：这时如果你去到 EC2 的产品界面查询，你会看到组成这个集群的所有虚拟机都出现在列表中。这体现了 EMR 的一种开放性设计，也方便用户登录到机器上进行测试、排查甚至定制。

你还记得在第 11 讲中，我们《双城记》文本文件在 S3 存储桶中的位置吧？在这里，我们使用 **hadoop fs 命令**来尝试查看一下：

复制代码

```
[hadoop@ip-172-31-1-100 ~]$ hadoop fs -ls s3://geektime-hellocloud/
Found 1 items
-rw-rw-rw-   1 hadoop hadoop     804335 2020-02-07 14:45 s3://geektime-hellocloud
```

你看，通过使用 S3 协议头，EMR 创建的集群是能够自动地和我们的 S3 对象存储“打通”的，相当于把 S3 挂载为了一个文件系统。这为应用程序的访问和存储数据提供了极大的便利。

接下来，进入到集群为我们安装准备好的 **spark-shell**（Spark 的交互式命令行工具），我们可以直接运行一个基于 Spark 的经典 **WordCount 程序**。（为节约篇幅，这里省去了一些准备性语句和一些打印输出）

```
复制代码
1 val book = sc.textFile("s3://geektime-hellocloud/ATaleOfTwoCities.txt")
2 val wordCounts = book.flatMap(l => l.split(" ")).filter(_.length>0).map(w => (w,
3 wordCounts.toDF("word", "count").write.parquet("s3://geektime-hellocloud/ATaleOfT
```

程序顺利执行后，由于我们把计算结果用 **Parquet 列存储格式**，落地存储到了 S3 中，你可以到 S3 的相关界面里确认。在这里，你可以看到目录已经生成了：

点击进入目录，可以看到其中的 Parquet 文件，说明程序已经生成了结果：

为了证明结果的正确性，我们再使用另外一段 Spark 脚本，来读取刚才的结果文件，列出小说中出现频率最高的 5 个单词及出现次数：

```
复制代码
1 scala> spark.read.parquet("s3://geektime-hellocloud/ATaleOfTwoCities-WordCount").
2 [the,7514]
3 [and,4745]
4 [of,4065]
```

```
[to,3458]
[a,2825]
```

EMR 中再次成功地执行了我们的操作，返回了正确的结果，我们的实验圆满完成。

在这个实验中，虽然我们用的数据规模比较小，但流程是比较完整的，这也充分体现了云上大数据服务的**易用性**。

云上大数据的特点

从刚才的例子中可以看出，在使用体验层面，云上大数据看上去和你熟悉的大数据技术栈没什么两样。这也是 PaaS 服务的设计目标之一，也就是尽可能地**保证兼容性**。

不过如果我们深入探究，会发现云上大数据服务并不仅仅是做了开源大数据技术的移植和搬运，还是融入了自己的特色，尤其是进一步地**解耦了大数据架构中的计算和存储**。

对于存储端，在上面的实验中，我们已经看到这种解耦的体现，应用程序直接把对象存储服务，作为大数据架构的数据源和计算结果的存储位置，它可以不需要使用和依赖本地的 HDFS 文件系统。

当然，这并不是 AWS 的专利，事实上几乎所有厂商的云端大数据服务，都与对象存储服务进行了深度的集成和适配，正好匹配对象存储大容量、高吞吐的特点，形成了一对黄金搭档。所以在云上大数据的存储端，**对象存储**一般都是优先选择的存储方式。

提醒：在一些大数据服务中，比如 Azure 的 HDInsight，会将对象存储默认挂载为 Hadoop 的根文件系统’/’，你可以查看 fs.defaultFS 参数来确认。这样使用起来，你就无需存储协议前缀，更加接近传统 HDFS 的体验。

在计算端，你同样可以从解耦中受益。得益于云端的弹性，你可以按需地对大数据集群的规模进行动态的调整，来满足不同计算规模和交付时间的要求。

计算端的另一个常用的最佳实践是，**集群可以动态地创建，做完工作后立刻将集群删除。**

这种**按需启停**的使用方式，让你可以专门为一个目的，甚至一个任务来创建集群，完成后就可以关闭删除。这样不但最大限度地节省了资源，也通过集群专用，解决了不同大数据任务间资源抢占和冲突的问题。要知道，这在云时代之前是不太可能的，因为安装和设置大型集群非常花时间。

一般云上大数据服务还会带有很多**增值服务**，除了常见的性能监控之外，值得一提的是许多云都自带了 **Jupyter Notebook**（一种交互式笔记本），能让你快速方便地和集群交互。另外，有些大厂商还会对大数据框架本身进行改进，以进一步增强自己的竞争力。比如说，AWS 从 EMR 5.28 版本起，就引入了 EMR Runtime for Apache Spark，这个运行时在实现接口和 API 兼容性的前提下，优化了很多场景下的运行性能。

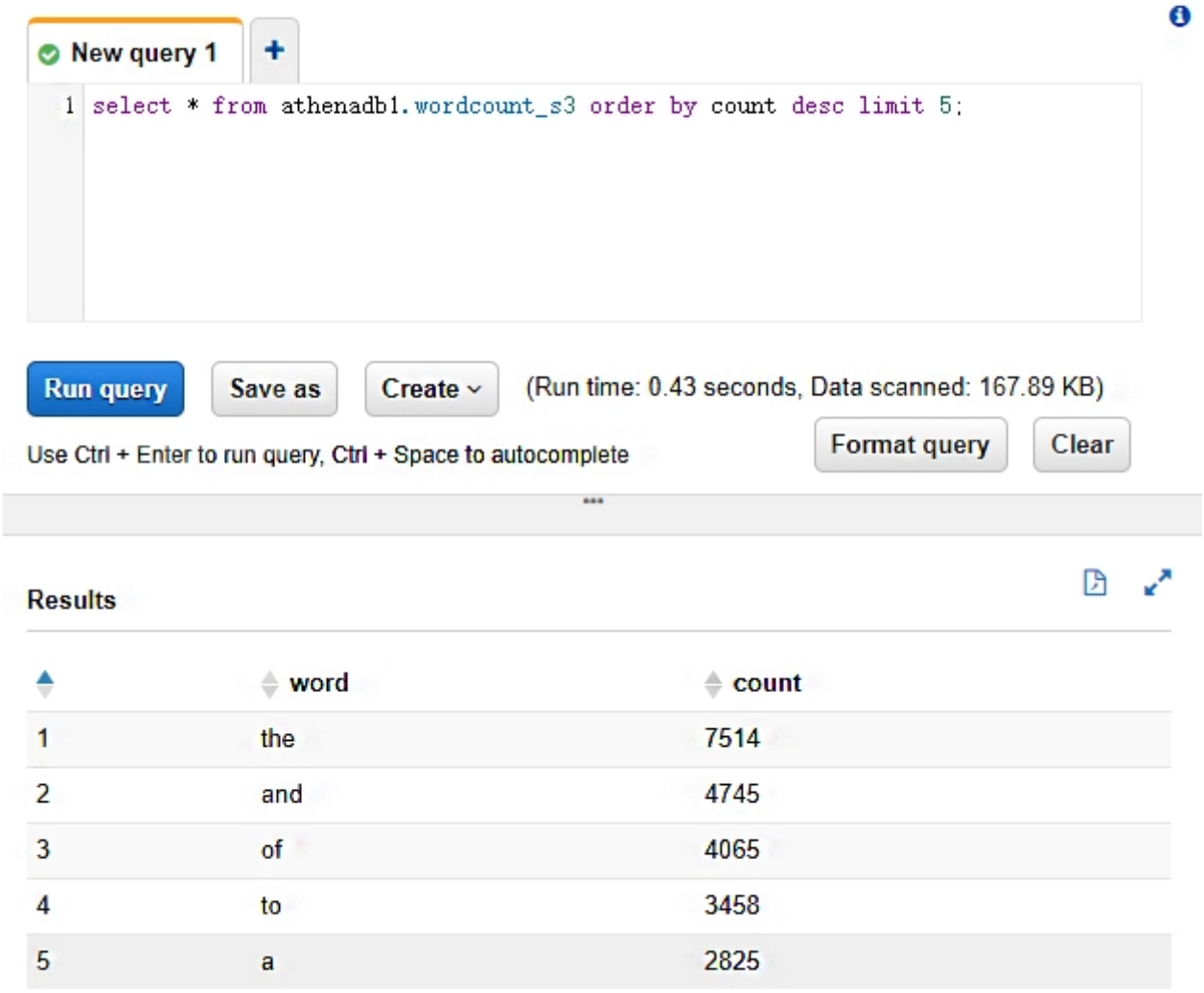
另外，还有一类大数据服务，更是计算存储解耦的终极体现，那就是**无服务器查询服务**，典型的有 AWS 的 Athena，和阿里云的 Data Lake Analytics（DLA，数据湖分析）等。它们甚至不需要你事先申请计算资源，而是可以即查即用，直接查询对象存储中的数据。然后，你也可以“用完即走”，不必操心服务的关闭。这类服务在成本上也很独特，和 EMR 这样的按照集群规模和在线时长计费不同，它只按照读取扫描的字节数收费，非常适合偶发的数据分析需求。

举例来说，前面我们已经有处理好的 Parquet 文件，位于 geektime-hellocloud 存储桶下的 WordCount 目录。现在我们可以在 Athena 中，创建一个**外部表**指向这个存储地点：

复制代码

```
CREATE DATABASE athenadb1;
CREATE EXTERNAL TABLE athenadb1.wordcount_s3 (
  word string,
  count int
) STORED AS PARQUET
LOCATION 's3://geektime-hellocloud/ATaleOfTwoCities-WordCount';
```

然后，我们就可以在 Athena 的交互界面中轻松查询数据了。比如说，我们仍然查询小说中频率最高的 5 个单词，也可以通过 SELECT 语句很快得出结果：



分析型数据库的崛起

除了 MapReduce 和 Spark 这样的计算框架，在大数据技术领域，还有另外一个发展思路和重要分支，那就是 **MPP**（Massively Parallel Processing）数据库，现在也常被称为**分析型数据库**。高压缩比的列式存储和分布式并行查询处理是它的两大法宝。

由于和关系型数据库有着千丝万缕的关系，分析型数据库乍看上去有些“复古”，它使用的是 SQL 这种声明式的表达。但也正是因为这一点，使得它降低了大数据技术的门槛，可以让广大熟悉关系型数据库的用户能够用自己习惯的方式，来查询和处理大数据。常见的分析型数据库有 Greenplum 和 Teradata 等等。

毫无疑问，云计算也积极地进入了这一领域，甚至唱起了主角。AWS 的 Redshift、阿里云的 AnalyticDB 等，都是非常优秀的云上分析型数据库。

云上分析型数据库，不仅保留了 MPP 数据库的特点，而且云端的生态结合得很好。这里我们继续进行上面的实验，使用 AWS 著名的 Redshift 服务，来实际地说明这一点。

小提示：这里我们略过 Redshift 服务本身创建的过程，你可以选择一个最小集群的规模来做实验。

前面我们已经有处理好的 Parquet 文件，位于 S3 中 geektime-hellocloud 存储桶下的目录当中。Redshift 中，同样支持通过外部表的方式来访问 S3 上的文件，这一特性被称为 **Redshift Spectrum**。通过它，我们可以很方便地引用不在 Redshift 自有存储中的数据。

方便起见，这里我们直接创建一个指向前面 Athena 中数据库的外部 Schema，这样前面在 Athena 中定义的外部表 **wordcount_s3**，就可以被 Redshift 识别和利用了：

复制代码

```
create external schema athenadb1
from data catalog database 'athenadb1'
iam_role 'arn:aws:iam::5085xxxx8040:role/mySpectrumRole'
```

注：这里的 mySpectrumRole 赋予了 Redshift 访问 S3 的权限。相关细节在这里不展开了，你可以参阅 AWS 的相关文档。

然后，我们就能在 Redshift 中直接查询 wordcount_s3 表了。但你也可以把这张外部表的数据拉取过来，并落地到 Redshift 自己的存储中。我们使用 **CTAS 语句**来实现这一点：

复制代码

```
create table wordcount distkey(word) sortkey(count) as
select word, count from spectrum.wordCount_s3;
```

小提示：这里我们在创建内部表 wordcount 时，通过 **distkey 和 sortkey** 指定了用于分布和排序的字段，充分体现了 Redshift 的分布式数据库特点。合理地设置这些字段，有利于查询性能的提高。

接着就可以查询这张新表了。一般来讲，查询内部表的响应会更快，也能支持更高的并发。

```
client@clientVM:~$ psql -h my-redshift-cluster-1.cwq9zgi6xxxx.us-west-1.redshift.
 word | count
------+-------
 the  |  7514
 and  |  4745
 of   |  4065
 to   |  3458
 a    |  2825
(5 rows)
```

我们还是查询小说中频率最高的 5 个单词，成功地得出了同样结果。值得注意的是，我使用了**psql 命令行工具**来进行查询，这是一个通用的 PostgreSQL 的客户端。这证明了 Redshift 在接口层面和 PostgreSQL 关系型数据库的兼容性，你可以复用相关的工具链和生态。

好，我们的实验就进行到这里。总得来说呢，云上的分析型数据库，是基于 MPP 架构的云原生数据库，它非常易用而又性能强大，也能够存储和处理很大规模的数据。与云端其他大数据生态的整合，更让它如虎添翼。这也是为什么，云分析型数据库在云上获得了越来越多的关注，也成为了最热门的云上 PaaS 服务之一。

课堂总结与思考

数据是现代应用的核心，也是普遍的需求。云上大数据服务的出现和发展，让我们在云上存储、处理和查询大数据变得简单而高效，它也把云计算的计算存储分离特性，体现得淋漓尽致。所以它们两者呢，真的可以说是天作之合。

云计算落地大数据的形式，既有拿来主义、消化吸收，也有推陈出新、自研改进。这也是我喜欢云的一点，**它没有代替你做技术上的选择，而是同时给你提供了多种各具特色的解决方案**。

在今天的实操环节，我们主要是通过 AWS 的相关服务来进行的。事实上，在大数据这个领域，各个云厂商都是重兵集结、投入很大，现在的产品服务能力也都比较丰富和成熟了。为了便于你了解和比对，我这里也用一个表格，把不同云的相关服务进行了分类说明：

服务类别	AWS	阿里云	Azure
大数据计算	EMR	E-MapReduce, MaxCompute	HDInsight
大数据存储	S3 (EMRFS)	OSS, JindoFS	Blob Storage, Data Lake Storage Gen2
分析型数据库	Redshift	AnalyticDB for MySQL/ PostgreSQL	SQL Data Warehouse (Synapse Analytics)
无服务器查询	Athena	Data Lake Analytics	Azure Data Lake Analytics

通过以上产品和服务的有机组合，你就可以轻松地构建出非常强大的数据仓库和数据湖解决方案了。如果再加上云上的数据管理和数据集成类服务一起配合，比如 AWS Glue、阿里云 DataWorks 等，那几乎就形成一个更广义、更完整的数据平台了。

今天我留给你的思考题是：

> 在 Hadoop 的黄金时代，它最受推崇的一个架构特点就是实现了数据的**就近访问**（Data Locality），这个特性能够将计算移动到数据所在的地方，以获取最高的性能。现在在云上，如果使用远端的对象存储，是否和这个思想背道而驰了呢？背后是什么样的因素在推动这个转变呢？
>
> Hadoop 体系中的 **Hive**，也是一种常用的分布式数据仓库，云上大数据服务中一般也都包含了这个组件。那既然有了 Hive，为什么还要研发 Redshift 这样的分析型数据库呢？它们的应用场景有什么不同？

欢迎你在留言区和我互动。如果觉得有收获，也欢迎你把这篇文章分享给你的朋友。感谢阅读，我们下期再见。

精选留言 (6)

何恺铎 置顶

2020-04-01

[上讲问题参考回答]
1. 防止"删库跑路"的要点，一个在于备份，一个在于权限。很多同学都同时提到了这两点。备份方面要善用云数据库各种备份机制，包括自动的增量/全量备份、事务日志备份、可长期保存的数据库快照等。权限方面主要注意数据库本体和备份的权限分离，不要把所有钥匙交到一个人的手里。如果真的出现意外彻底删除，也要记得迅速联系云厂商后台支持，仍有找回数据的一线生机。
2. 分区当然仍有它的应用价值；对于某些NoSQL类云原生数据库，分区是必选机制，直接影响着存储层的sharding策略；而对于关系型云原生数据库，也大都支持经典的数据库分区操作，和底层分布式存储不但不冲突，还能够有助于减少数据扫描、提高查询性能。

  11

罗辑思维

2020-04-02

问题：现在在云上，如果使用远端的对象存储，是否和这个思想背道而驰了呢？背后是什么样的因素在推动这个转变呢？

个人思考：任何技术手段都是考虑性价平衡，如对读写延迟要求不高，低频，归档的数据可以放在远端。还有所谓远端可能是其他人的近端，可以通过BGP网络或者CDN加速的方式，提供用户就近访问，有效降低云服务器负载。

想到庄振运老师说过通过性能优化替公司省了几千万美刀，才明白技术是生产驱动力，省钱更是硬道理。

作者回复: 思考的角度不错哦，赞。

  5

Christopher

2020-04-01

云计算和大数据就是珠联璧合啊

  4

leslie

2020-04-01

Hadoop没有研究过：不过分析型数据库是初次考虑去涉及。我们不防去提出另一个问题：mongodb为何要对sql做极好的支持？这个是从mongodb初始就有的。
redis和mongodb应当算是同类型的基于内存存储的数据库：为何一个对sql做了极好的支持一个没有？老师分享的图片中有指出阿里云的两种分析型数据库是基于PGSQL和MYSQL，它们背后其实是缺乏OLAP类型DB支持的。
谢谢分享，期待后续课程。

作者回复: 这里要修正一下你的说法，虽然官方的MySQL和PostgreSQL不支持列式存储，但阿里云ADB for MySQL/PostgreSQL都是有定制的列存储或行列混合存储引擎的，所以才称得上是“分析型数据库”。

支不支持SQL，以及优先级如何，更多取决于数据库的应用场景和数据结构特点。很多时候不是能不能的问题，是需不需要的问题。另外MongoDB和Redis的差别还是蛮大的。

2

胖子
2020-04-15

MPP数据库因解决什么问题而发展过来的?

作者回复: 本质上MPP也是为了解决传统关系型数据库的性能和扩展性问题而诞生的，只是和Hadoop生态采用了不同的“大数据”思路和做法。

1

易轻尘
2021-07-28

第一个问题个人感觉是因为目前网络越来越快了，所以query延迟变得能够接受了。另一个原因是基于成本和可用性的考虑，比起企业私有的hadoop集群，云计算服务能够以更便宜的价格提供更专业的服务。