NOSQL paper整理 18301004 韩璟瑄

bigtable:

概览：

Bigtable是一个用于管理结构化数据的分布式存储系统，其设计目的是扩展到非常大的规模：数千台商品服务器上的PB数据。Google的许多项目将数据存储在Bigtable中，包括web索引、googleearth和googlefinance。Bigtable成功地为所有这些谷歌产品提供了灵活、高性能的解决方案。本文描述了bigtable提供的简单的数据模型，bigtable给了客户端动态控制超目录和格式，并描述了bigtable的设计和实现。Bigtable实现了几个目标：广泛的适用性、可伸缩性、高性能和高可用性。Bigtable被60多个Google产品和项目使用，包括Google Analytics、Google Finance、Orkut、Personalized Search、Writely和Google Earth。

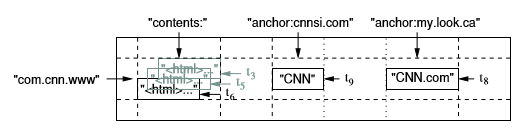
Bigtables类似于数据库：它与数据库共享许多实现策略。并行数据库和主内存数据库实现了可扩展性和高性能，但Bigtable提供了与这些系统不同的接口。

第2节详细描述了DataModel，第3节概述了客户机API。第4节描述了Bigtable依赖的google基础设施。第5节介绍了Bigtable实现的基本原理，第6节描述了我们为提高Bigtable性能所做的一些改进。第7节提供了Bigtable性能的度量。我们在第8节中描述了Google如何使用Bigtable的几个例子，并在第9节讨论了我们在设计和支持Bigtable时学到的一些经验教训。最后，第10节介绍了相关工作，第11节给出了结论。

第2节：data model

1).

数据模型举例：（webpage存储）



2).row:

表中的行键是任意的字符串（目前最大为64KB，但对于大多数用户来说，10-100字节是一个典型的大小）。单个行键下的数据的每次读写都是原子的。表的行范围是动态分区的。每个行范围称为一个板块(tablet)，它是分布和负载平衡的单元。因此，短行范围的读取效率很高，通常只需要与少数机器通信。客户机可以通过选择行键来利用此属性，以便为数据访问获得良好的位置。

3).column family:

列键被分组到称为列族的集合中，列族是访问控制的基本单元。列族必须在将数据存储在该族中的任何列键下之前创建；在创建了族之后，可以使用该族中的任何列键。

列键使用以下语法命名：family: qualifier 列名必须是可打印的，但限定符可以是任意字符串。访问控制以及磁盘和内存记帐都是在列族级别执行的。

1. .timestamps:

Bigtable中的每个单元格可以包含同一数据的多个版本；这些版本是按时间戳索引的。bigtable时间戳是64位比特器。它们可以由Bigtable指定，在这种情况下，它们以微秒表示“实时”，或者由client applications指定。需要避免冲突的应用程序必须自己生成唯一的时间戳。单元格的不同版本按时间戳的降序存储，因此最新版本可以首先读取。

第3节：API

Bigtable API提供用于创建和删除表和列族的函数。它还提供更改集群、表和列族元数据的功能，例如访问控制权限。Bigtable支持允许用户以更复杂的方式操作数据的其他一些功能。1.Bigtable支持单行事务，BigTable目前不支持跨行键的常规事务，尽管它为客户端的批处理写入跨行键提供了接口。 2.Bigtable允许单元格用作整数计数器。 3.Bigtable支持在服务器的地址空间中执行客户端提供的脚本。Bigtable可以与MapReduce一起使用，MapReduce是在Google开发的一个运行大规模并行计算的框架。

第4节：building blocks

Bigtable是基于Google的其他几个基础设施构建的。Bigtable使用分布式Google文件系统（GFS）来存储日志和数据文件。Google SSTable 文件格式用于内部存储Bigtable数据。SSTable提供了一个从键到值的持久有序的不可变映射，其中键和值都是由测试环任意设置的。

Bigtable依赖一个高可用性和持久性的分布式锁服务Chubby。

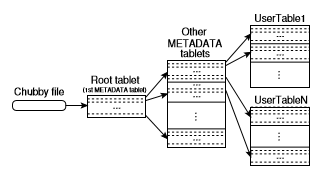
Bigtable使用chubby执行各种任务：确保任何时候最多有一个活动主机；存储Bigtable数据的引导位置（参见第5.1节）；发现平板(tablet?)服务器并确定平板服务器销毁（参见第5.2节）；存储Bigtable 模式信息（每个表的列族信息）和访问控制器。主机负责将Tablets分配到tablet服务器，检测平板服务器的添加和过期情况，平衡平板服务器负载，并在GFS中对文件进行垃圾收集。

第5节：implementation

Bigtable实现有三个主要组件：链接到每一个客户端的库、一个master服务器和许多平板服务器。平板服务器从集群中动态添加（或删除）以适应工作负载中的变化。此外，它还处理模式更改例如表和列族的创建。

5.1Tablet Location

使用类似于B+树的三级层次结构来存储tablet location.



第一级是存储在Chubby中的文件，其中包含根板的位置。根数字化仪包含一个特殊元数据表中所有平板电脑的位置。每个元数据板都包含一组用户板的位置。根tablet只是元数据表中的第一个tablet，但经过特殊处理，它从不拆分，以确保tablet位置层次结构不超过三个级别。

客户端库缓存平板位置。如果客户端不知道平板的位置，或者发现缓存的位置信息不正确，则它会递归地向上移动到平板位置层次结构。如果客户机的缓存是空的，那么定位算法需要三次网络往返，包括一次从Chubby读取。如果客户机的缓存已过时，则定位算法可能需要六次往返，因为过时的CacheEntries只能发现重复项（假设元数据板不会频繁移动）。虽然tablet位置存储在内存中，因此不需要GFS访问，但我们通过让客户端库预取tablet位置来进一步降低此成本：它读取多个表的元数据，而每次读取元数据表。我们还将辅助信息存储在元数据表中，包括与每个平板相关的所有事件的日志（例如服务器开始为其提供服务时）。

5.2Tablet Assignment

每个平板一次分配给一个平板服务器。主机会跟踪实时平板服务器的设置，以及当前平板分配给平板服务器的情况，包括哪些平板未分配。如果未分配平板，并且有足够空间放置平板的平板服务器可用，则主机通过向tabletserver发送TabletLoadRequest来分配平板。

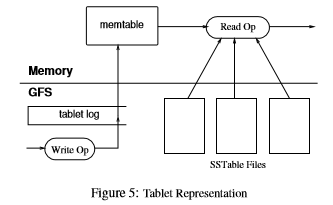
Bigtable使用Chubby跟踪平板服务器。当平板电脑服务器启动时，它会在一个特定的Chubby目录中创建一个唯一命名的文件，并获得独占锁。主服务器监视此目录（ServersDirectory）以发现可覆盖的服务器。如果失去独占锁，Atablet服务器将停止为平板电脑提供服务.

主机负责检测平板服务器何时不再为其平板电脑提供服务，并负责尽快重新分配这些平板电脑。为了检测平板电脑服务器何时不再为其平板电脑提供服务，主机会定期向每个平板电脑服务器询问其锁的状态。主机在启动时执行以下步骤。（1） master在Chubby中获取一个惟一的主锁，这可以防止并发的主实例。（2） 主服务器扫描Chubby中的servers目录以查找活动服务器。（3） 主服务器与每个实时平板服务器通信，以发现已分配给每个服务器的平板电脑。（4） 主服务器扫描元数据表以获取这些时间片。当此扫描遇到尚未分配的平板电脑时，主机会将该平板电脑添加到未分配的平板电脑集合中。

在分配元数据表之前，无法扫描元数据表。因此，在开始此扫描（步骤4）之前，如果在步骤3中未发现根平板的分配，则主机会将根表添加到未分配的平板中。此添加将确保分配根平板。

5.3 Tablet Serving

平板电脑的持久状态存储在GFS中，如图5所示。更新被提交到存储重做记录的提交日志中。在这些更新中，最近提交的更新存储在内存中一个称为memtable的排序缓冲区中；较旧的更新存储在sstable序列中。从平板电脑服务器读取其元数据。



当写入操作到达平板服务器时，服务器检查其格式是否正确，以及发送方是否有权执行变异。在拆分和合并Tablets时，传入的读写操作可以继续。

5.4 Compactions

当执行写操作时，memtable的大小会增加。当内存表大小达到阈值时，将冻结memtable，创建新的memtable，并将冻结的memtable转换为sstable并写入GFS。每一次小压缩都会创建一个新的sstable.将所有sstable重写为一个sstable的合并压缩称为major compression。

1. Reﬁnements

上一节所述的实现需要许多要素来实现用户所需的高性能、可用性和可靠性。本节将更详细地描述实现的部分，以突出这些缺点。

6.1Localitygroups

客户机扫描组将多个列系列集合到一个位置组中。为每个绘图板中的每个位置组生成一个单独的SSTable。将非典型访问的列族分离到单独的区域组中，可以提高读取效率。

6.2Compression

客户机可以控制是否压缩区域组的sstables，如果压缩，使用哪种压缩格式。用户指定的压缩格式应用于每个sstable block（通过区域组特定的调整参数可以控制谁的大小）。虽然我们通过单独压缩每个块而损失了一些空间，但我们的好处在于，sstable的一小部分可以在不解压缩整个文件的情况下读取。许多客户机使用两遍自定义压缩方案。

6.3Cachingforreadperformance

缓存读取性能为了提高读取性能，平板服务器使用两个级别的缓存。扫描缓存是一个高级缓存，用于缓存SSTable接口返回给tablet服务器代码的键值.对块缓存是一个较低级别的缓存，用于缓存从GFS阅读。

6.4Bloomﬁlters

如5.3节所述，读取操作必须从构成平板电脑状态的所有sstable中读取。如果这些sstable不在内存中，我们可能会进行多次磁盘访问。通过允许客户机指定应该为特定位置组中的SSTables创建Bloom过滤器，从而减少访问的数量。

6.5Commit-logimplementation

如果我们将每个平板电脑的提交日志保存在一个单独的日志文件中，大量文件将同时写入GFS中。根据底层文件系统实现的每个文件系统服务器，这些写入操作可能会导致大量磁盘请求写入不同的物理日志文件。此外，每个平板有单独的日志文件也会降低组提交优化的有效性，因为组往往会更小。为了解决这些问题，我们将突变附加到每个平板服务器的单个提交日志中，将不同平板电脑的突变混合在同一物理日志文件中。

在正常操作期间，使用一个日志可以显著提高性能，但这会使恢复复杂化。当一台平板服务器死机时，它所服务的平板电脑将被转移到大量其他平板服务器上：每台服务器通常会装载少量原始服务器的平板电脑。要恢复平板电脑的状态，新的平板电脑服务器需要从原始平板电脑服务器写入的提交日志中重新应用该平板的变化。

6.6Speeding up tablet recovery

如果主机将平板从一个平板服务器移动到另一个平板服务器，则源平板服务器首先会对该平板电脑进行少量压缩。这种压缩减少了tablet服务器提交日志中未压缩状态的数量，从而缩短了恢复时间。

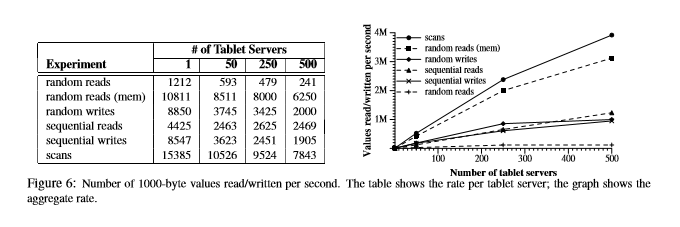
6.7Exploiting immutability

除了SSTable缓存之外，Bigtable系统的其他各个部分都被简化了，因为我们生成的所有SSTable都是不可变的。

SSTables的不变性使我们能够快速分割平板。我们不必为每个子平板生成一组新的sstable，而是让子平板共享parenttable的稳定性。

7 Performance Evaluation

我们建立了一个包含N个平板服务器的Bigtable集群，以测量N个变化时Bigtable的性能和可伸缩性。平板服务器被配置为使用1GB内存，并写入由1786台机器组成的GFS单元，每台机器有两个400GB IDE硬盘。N个客户机生成用于这些测试的Bigtable负载。

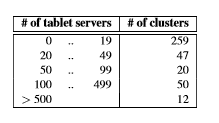


扫描基准与顺序读取基准类似，但使用Bigtable API提供的对行范围内所有值的扫描支持。使用扫描可以减少基准测试执行的RPC数量，因为单个RPC从Tablet服务器获取大量值。

7.1Single tablet-server performance

首先考虑一台平板服务器的性能。随机读卡器可将分析仪操作降低一个数量级或更多。每次随机读取都需要通过网络将一个64kb的SSTable块从GFS传输到平板服务器，其中只有一个1000字节的值被使用。平板服务器每秒大约执行1200次读取，这相当于从中读取大约75MB/s的数据GFS。顺序读取比随机读取执行得更好，因为从GFS获取的每个64kb SSTable块都存储在我们的块缓存中，在那里它用于服务下一个读取请求。扫描速度更快，因为TabletServer可以返回大量值以响应单个客户端RPC，因此RPC开销分摊到较大的值上。

7.2Scaling



当我们将系统中的平板服务器数量从1台增加到500台时，总吞吐量急剧增加，增幅超过100倍。然而，性能并不是线性增长。这种不稳定是由多个服务器配置中的负载不平衡引起的，通常是由于其他进程在争夺CPU和网络。

8 RealApplications

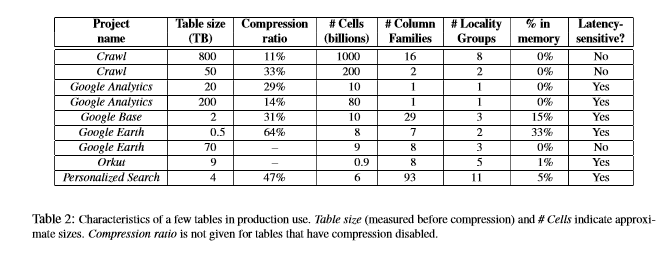
截至2006年8月，共有388个非testbigtablecluster运行在各种Google机器集群中，总共有大约24500台平板服务器。表1显示了每个集群的平板服务器的大致分布情况。其中许多集群用于开发目的，因此在重要时期闲置。一些表存储为用户服务的数据，而另一些表存储用于批处理的数据；这些表的总大小、平均单元大小、以及内存中数据服务的百分比，以及TableSchema的复杂性。在本节的后面，我们简要描述了三个产品团队使用bigtable的情况。

8.1 Google Analytics

谷歌分析(analytics.google.com)是一项帮助网站管理员分析其网站流量模式的服务。它提供了聚合统计信息，如每天的独立访问者数量和每个URL每天的页面浏览量，以及网站跟踪报告，例如在用户之前查看特定页面的情况下进行购买的用户百分比。为了启用这个服务，网站管理员在他们的网页中嵌入了一个小的JavaScript程序。当访问evera页面时调用此程序。它在googleanalytics中记录有关请求的各种信息，例如用户标识符和正在获取的页面的信息。谷歌分析总结了这些数据，并使网站管理员可以使用。我们简要介绍了谷歌分析公司使用的两个表格。

8.2 GoogleEarth

谷歌运营着一系列服务，用户可以通过基于网络的谷歌地图界面访问世界表面的高分辨率卫星图像(maps.google.com)通过谷歌地球(earth.google.com)自定义客户端软件。这些产品允许用户在世界表面导航：他们可以平移、查看和注释不同分辨率的卫星图像。这个系统使用一个表来预处理数据，而另一组表用于存储客户机数据。处理管道可用于存储图像。



图像表中的每一行对应于一个地理段。对行进行命名是为了确保相邻的地理区域段可以相互还原。该表包含一个列族，用于跟踪每个段的数据源。这个列族有大量的列：基本上每个原始数据图像对应一个列。

8.3 Personalized Search

个性化搜索(www.google.com/psearch)是一种选择性加入服务，它记录用户在各种Google属性（如web搜索、图像和新闻）中的查询和单击。他们可以根据自己的历史记录和历史记录来浏览用户的历史记录。个性化搜索将每个用户的数据存储在Bigtable中。

个性化搜索数据在几个大表集群中进行复制，以提高可用性并减少与客户机之间的距离。为了帮助支持共享，我们在Bigtable中添加了一个简单的配额机制，以限制共享表中任何特定客户机的存储消耗；该机制在使用该系统的各个产品组之间为每个用户的信息存储提供一些隔离。

9 Lessons

在设计、实现、维护和支持Bigtable的过程中，我们获得了有益的经验和教训。

学到的一个教训是，大型分布式系统易受多种故障的影响，而不仅仅是标准网络分区和许多分布式协议中假定的故障停止故障。当人们请求分布式事务时，最重要的用途是维护二级索引，我们计划添加一个专门的机制来满足这种需求。与分布式事务相比，新机制不那么通用，但效率更高（尤其是对于跨越数百行或更多行的更新），并且还将更好地与我们的跨数据中心复制方案交互。

另一个教训是，延迟添加新功能非常重要，直到清楚地知道如何使用新功能。

从支持Bigtable中学到的一个实际经验是正确的系统级监视的重要性（即监视Bigtable本身以及使用Bigtable的客户机进程）。

另一个有用监视的例子是每个Bigtable集群都注册在Chubby中。

我们学到的最重要的一课是简单设计的价值。考虑到我们系统的大小（大约100000行非测试代码），以及代码随着时间的推移以意想不到的方式演变，我们发现代码和设计的清晰性对代码维护和调试有着巨大的帮助。

10 Related Work

Boxwood项目[24]的组件在某些方面与Chubby、GFS和Bigtable重叠，因为它提供了分布式协议、锁定、分布式块存储和分布式B树存储。在每一个有重叠的情况下，Boxwood的组件目标都比相应的Google服务低一些。Boxwood项目的目标是为构建高级服务（如文件系统或数据库）提供基础设施，而Bigtable的目标是直接支持希望存储数据的客户端应用程序。许多最近的项目都解决了在广域网（通常是“互联网规模”）上提供分布式存储或更高级别服务的问题。这包括从CAN[29]、Chord[32]、Tapestry[37]和pastry[30]等项目开始的分布式哈希表的工作。这些系统解决了对大表进行公证的问题，如大的可变带宽、不可信的参与者或频繁的重新配置；分散控制和拜占庭式容错不是大目标。

键值对是一个有用的构建块，但它们不应该是提供给开发人员的唯一构建块。我们选择的模型比简单的键值对更丰富，并且支持稀疏的半结构化数据。尽管如此，它仍然足够简单，它可以提供一个非常有效的反映，而且它是透明的（通过本地组）允许用户对系统的重要行为进行调整。一些数据库供应商已经开发出可以存储大量数据的并行数据库。

考虑到Bigtable不寻常的接口，一个有趣的问题是，我们的用户很难适应使用它。新用户有时不确定如何使用大表接口，尤其是他们使用支持通用事务的关系数据库。最后，我们发现在谷歌建立我们自己的存储解决方案有很大的优势。我们从自己设计的表格中得到了大量的数据。此外，我们对Bigtable实现的控制，以及Bigtable依赖的其他google基础设施，意味着我们可以消除瓶颈和效率。

1. Conclusions

Bigtable，一个在Google存储结构化数据的分布式系统。Bigtable集群自2005年4月开始投入生产使用，在此之前我们花了大约7个人年的时间进行设计和实现。截至2006年8月，超过60个项目正在使用Bigtable。我们的用户喜欢bigtableimplementation提供的性能和迁移能力，他们可以通过在资源需求随时间变化时向系统中添加更多计算机来扩展集群的容量。

考虑到Bigtable不寻常的接口，一个有趣的问题是，我们的用户很难适应使用它。新用户有时不确定如何最好地使用Bigtable接口，特别是如果他们习惯于使用支持通用事务的关系数据库。尽管如此，许多Google产品成功使用Bigtable的事实表明，我们的设计在实践中运行良好。我们正在实现几个附加的Bigtable特性，例如支持二级索引和构建具有多个主副本的跨数据中心复制的Bigtable的基础设施。作为一个庞大的团队，我们也不需要为自己的团队部署一个庞大的服务团队。随着服务集群的扩展，我们需要在Bigtable内部处理更多的资源共享问题[3,5]。最后，我们发现在谷歌建立我们自己的存储解决方案有很大的优势。我们从自己设计的表格中得到了大量的数据。此外，我们对Bigtable实现的控制，以及Bigtable依赖的其他Google基础设施，意味着我们可以在出现瓶颈时有效地消除它们。