厦门大学计算机科学系研究生课程

《大数据技术基础》

第12章 Google Spanner
（2013年新版）

林子雨

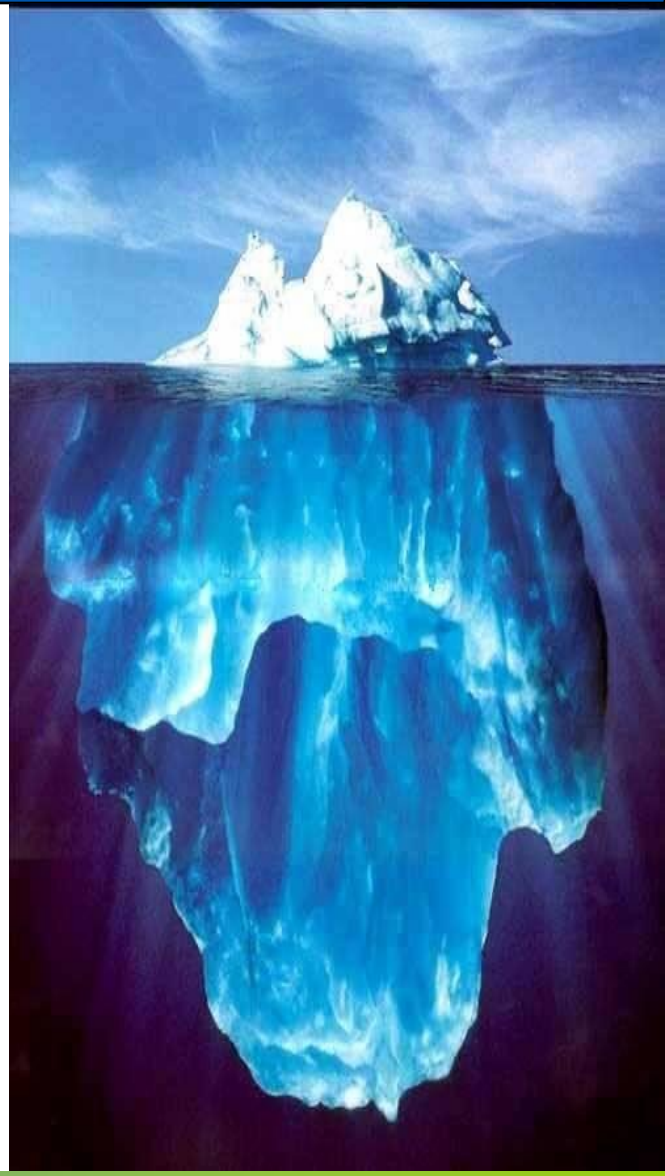
厦门大学计算机科学系

E-mail: ziyulin@xmu.edu.cn

主页：http://www.cs.xmu.edu.cn/linziyu

提纲

- Spanner背景
- 与BigTable、Megastore对比
- Spanner的功能
- 体系结构
- Spanserver
- Directory
- 数据模型
- TrueTime
- Spanner并发控制
- 参考文献



本讲义PPT存在配套教材，由林子雨通过大量阅读、收集、整理各种资料后编写而成
下载配套教材请访问《大数据技术基础》2013班级网站：http://dblab.xmu.edu.cn/node/423

Spanner

- Spanner是个可扩展，多版本，全球分布式还支持同步复制的数据库。
- 他是Google的第一个可以全球扩展并且支持外部一致的事务。
- Spanner能做到这些，离不开一个用GPS和原子钟实现的时间API。这个API能将数据中心之间的时间同步精确到10ms以内。
- 主要功能：无锁读事务，原子模式修改，读历史数据无阻塞。

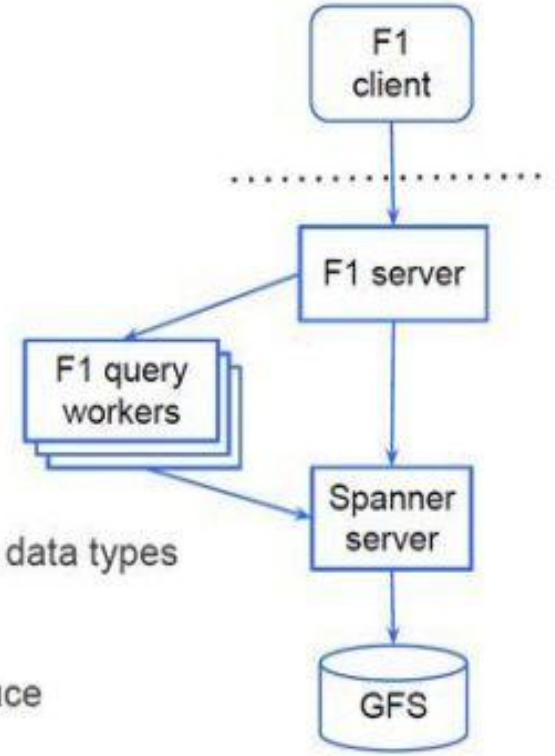
Spanner背景

要搞清楚Spanner原理，先得了解Spanner在Google的定位。
Spanner位于F1和GFS之间，承上启下。

F1 Google

Architecture

- Sharded Spanner servers
  - data on GFS and in memory
- Stateless F1 server
- Pool of workers for query execution



Features

- Relational schema
  - Extensions for hierarchy and rich data types
  - Non-blocking schema changes
- Consistent indexes
- Parallel reads with SQL or Map-Reduce

F1

和众多互联网公司一样，在早期Google大量使用了Mysql。Mysql是单机的，可以用Master-Slave来容错，分区来扩展。但是需要大量的手工运维工作，有很多的限制。因此Google开发了一个可容错可扩展的RDBMS——F1。

F1有如下特点：

- 7×24高可用。哪怕某一个数据中心停止运转，仍然可用。
- 可以同时提供强一致性和弱一致。
- 可扩展
- 支持SQL
- 事务提交延迟50-100ms，读延迟5-10ms，高吞吐

Colossus（GFS II）

Colossus是第二代GFS。Colossus是Google重要的基础设施，因为他可以满足主流应用对FS的要求。Colossus的重要改进有：

- 优雅Master容错处理 (不再有2s的停止服务时间)
- Chunk大小只有1MB (对小文件很友好)
- Master可以存储更多的Metadata(当Chunk从64MB变为1MB后，Metadata会扩大64倍，但是Google也解决了)

Colossus可以自动分区Metadata。使用Reed-Solomon算法来复制，可以将原先的3份减小到1.5份，提高写的性能，降低延迟。客户端来复制数据。

与BigTable、Megastore对比

•Spanner主要致力于跨数据中心的数据复制上，同时也能提供数据库功能。

•BigTable在Google得到了广泛的使用，但是他不能提供较为复杂的Schema，还有在跨数据中心环境下的强一致性。

•Megastore 有类RDBMS的数据模型，同时也支持同步复制，但是他的吞吐量太差，不能适应应用要求。

•Spanner不再是类似BigTable的版本化 key-value存储，而是一个“临时多版本”的数据库。

•Google官方认为 Spanner是下一代BigTable，也是Megastore的继任者。

Google Spanner设计

- 功能
- 体系结构
- Spanserver
- 目录与放置
- 数据模型
- TrueTime
- Google Spanner并发控制

功能

- 从高层看Spanner是通过Paxos状态机将分区好的数据分布在全球的。数据复制全球化的，用户可以指定数据复制的份数和存储的地点。 Spanner可以在集群或者数据发生变化的时候将数据迁移到合适的地点，做负载均衡。用户可以指定将数据分布在多个数据中心，不过更多的数据中心将造成 更多的延迟。用户需要在可靠性和延迟之间做权衡，一般来说复制1，2个数据中心足以保证可靠性。

功能

Spanner提供一些有趣的特性：

- 应用可以细粒度的指定数据分布的位置。
- Spanner还有两个一般分布式数据库不具备的特性：读写的外部一致性，基于时间戳的全局的读一致。

体系结构

Spanner由于是全球化的，所以有两个其他分布式数据库没有的概念。

· Universe。一个Spanner部署实例称之为一个Universe。目前全世界有3个。一个开发，一个测试，一个线上。因为一个Universe就能覆盖全球，不需要多个。

· Zones. 每个Zone相当于一个数据中心，一个Zone内部物理上必须在一起。而一个数据中心可能有多个Zone。可以在运行时添加移除Zone。一个Zone可以理解为一个BigTable部署实例。

体系结构

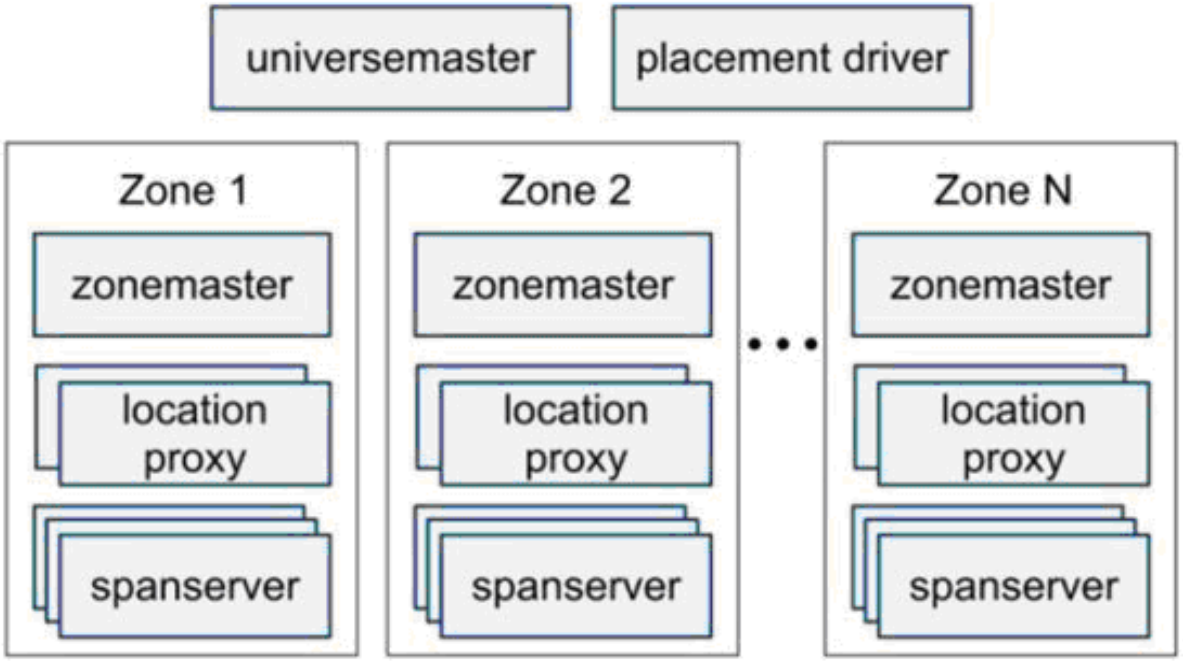


Figure 1: Spanner server organization.

- Universemaster: 监控这个universe里zone级别的状态信息
- Placement driver：提供跨区数据迁移时管理功能
- Zonemaster：相当于BigTable的Master。管理Spanserver上的数据。
- Location proxy：存储数据的Location信息。客户端要先访问他才知道数据在那个Spanserver上。
- Spanserver：相当于BigTable的ThunkServer。用于存储数据。

体系结构

Spanner由于是全球化的，所以有两个其他分布式数据库没有的概念。

•Universe。一个Spanner部署实例称之为一个Universe。目前全世界有3个。一个开发，一个测试，一个线上。因为一个Universe就能覆盖全球，不需要多个。

•Zones. 每个Zone相当于一个数据中心，一个Zone内部物理上必须在一起。而一个数据中心可能有多个Zone。可以在运行时添加移除Zone。一个Zone可以理解为一个BigTable部署实例。

Spanserver

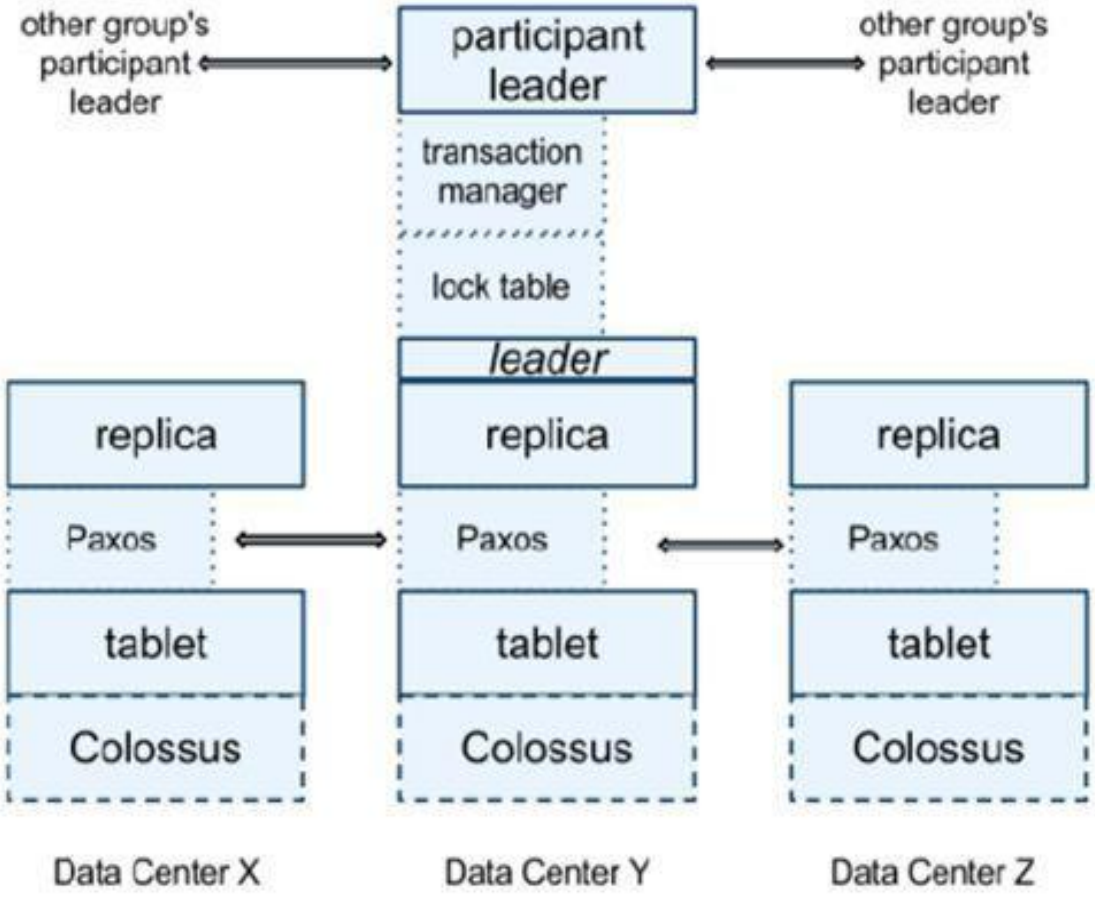


Figure 2: Spanserver software stack.

Spanserver

- 从下往上看。每个数据中心会运行一套Colossus (GFS II)
- 和BigTable不同的是BigTable里面的 tablet存储的是Key-Value都是string，Spanner存储的Key多了一个时间戳：

  (Key: string， timestamp: int64) ->string。
- 每个Tablet上会有一个Paxos状态机。Paxos是一个分布式一致性协议。Table的元数据和log都存储在上面。
- 每个leader replica的spanserver上会实现一个lock table还管理并发。
- 每个leader replica的spanserver上还有一个transaction manager。

Directory

- 之所以Spanner比BigTable有更强的扩展性，在于Spanner还有一层抽象的概念directory， directory是一些key-value的集合，一个directory里面的key有一样的前缀。更妥当的叫法是bucketing。Directory是应用控制数据位置的最小单元，可以通过谨慎的选择Key的前缀来控制，以在paxos group里面移来移去。
- Directory可以在不影响client的前提下，在后台移动。
- Directory是一个抽象的概念，管理数据的单元；而tablet是物理的东 西，数据文件。
- 在paxos group之间移动directory是后台任务。这个操作还被用来移动replicas。
- Directory还是记录地理位置的最小单元。

数据模型

- Spanner的数据模型来自于Google内部的实践。在设计之初，Spanner就决心有以下的特性：
  - 支持类似关系数据库的schema
  - Query语句
  - 支持广义上的事务
- 数据模型是建立在directory和key-value模型的抽象之上的。
- Spanner的数据模型也不是纯正的关系模型，每一行都必须有一列或多列组件。

数据模型

```
CREATE TABLE Users {
  uid INT64 NOT NULL, email STRING
} PRIMARY KEY (uid), DIRECTORY;

CREATE TABLE Albums {
  uid INT64 NOT NULL, aid INT64 NOT NULL,
  name STRING
} PRIMARY KEY (uid, aid),
  INTERLEAVE IN PARENT Users ON DELETE CASCADE;
```

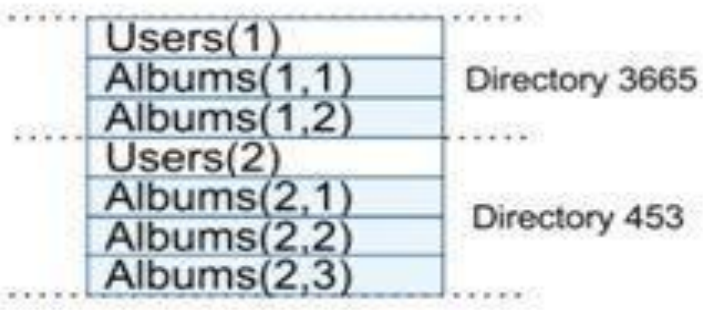


Figure 4: Example Spanner schema for photo metadata, and the interleaving implied by INTERLEAVE IN.

上图是一个例子。对于一个典型的相册应用，需要存储其用户和相册。

TrueTime

Method	Returns
TT.now()	*TTinterval*: [*earliest*, *latest*]
TT.after(t)	true if t has definitely passed
TT.before(t)	true if t has definitely not arrived

Table 1: TrueTime API. The argument t is of type *TTstamp*.

- TrueTime API 是一个非常有创意的东西，可以同步全球的时间。上表就是TrueTime API。TT.now()可以获得一个绝对时间TTinterval，这个值和UnixTime是相同的，同时还能够得到一个误差e。 TT.after(t)和TT.before(t)是基于TT.now()实现的。

TrueTime

- TrueTime API实现靠的是GFS和原子钟。
- 实际部署的时候，每个数据中心需要部署一些Master机器，其他机器上需要有一个slave进程来从Master同步。
- 每个Slave后台进程会每个30秒从若干个Master更新自己的时钟。

Google Spanner并发控制

Spanner使用TrueTime来控制并发，实现外部一致性。支持以下几种事务。

- 读写事务
- 只读事务
- 快照读，客户端提供时间戳
- 快照读，客户端提供时间范围

Operation	Concurrency Control	Replica Required
Read-Write Transaction	pessimistic	leader
Read-Only Transaction	lock-free	leader for timestamp; any for read
Snapshot Read, client-provided timestamp	lock-free	any
Snapshot Read, client-provided bound	lock-free	any

Google Spanner并发控制

- 单独的写操作都被实现为读写事务； 单独的非快照被实现为只读事务。
- 时间戳的设计大大提高了只读事务的性能。
- 对于快照读操作，可以读取以前的数据，需要客户端指定一个时间戳或者一个时间范围。Spanner会找到一个已经充分更新好的replica上读取。
- 还有一个有趣的特性的是，对于只读事务，如果执行到一半，该replica出现了错误。客户端没有必要在本地缓存刚刚读过的时间，因为是根据时间戳读取的。只要再用刚刚的时间戳读取，就可以获得一样的结果。

Google Spanner并发控制

读写事务

- 正如BigTable一样，Spanner的事务是会将所有的写操作先缓存起来，在Commit的时候一次提交。这样的话，就读不出在同一个事务中写的数据了。不过这没有关系，因为Spanner的数据都是有版本的。
- 在读写事务中使用wound-wait算法来避免死锁。
- leader首先会上一个写锁，他要找一个比现有事务晚的时间戳。通过Paxos记录。每一个相关的都要给coordinator发送他自己准备的那个时间戳。
- Coordinatorleader一开始也会上个写锁，当大家发送时间戳给他之后，他就选择一个提交时间戳。这个Coordinator将这个信息记录到Paxos。
- 在让replica写入数据生效之前，coordinator还有再等一会。需然后coordinator将提交时间戳发送给客户端还有其他的 replica。他们记录日志，写入生效，释放锁。

Google Spanner并发控制

只读事务

- 对于只读事务，Spanner首先要指定一个读事务时间戳。还需要了解在这个读操作中，需要访问的所有的读的Key。Spanner可以自动确定Key的范围。
- 如果Key的范围在一个Paxos group内。客户端可以发起一个只读请求给group leader。leader选一个时间戳，这个时间戳要比上一个事务的结束时间要大。然后读取相应的数据。这个事务可以满足外部一致性，读出的结果是最后 一次写的结果，并且不会有不一致的数据。
- 如果Key的范围在多个Paxos group内，就相对复杂一些。其中一个比较复杂的例子是，可以遍历所有的group leaders，寻找最近的事务发生的时间，并读取。客户端只要时间戳在TT.now().latest之后就可以满足要求了。

参考文献

- [1] James C. Corbett, Jeffrey Dean, Michael Epstein, etc. Spanner: Google’s Globally-Distributed Database.OSDI'2012.

主讲教师和助教

主讲教师：林子雨

单位：厦门大学计算机科学系
E-mail: ziyulin@xmu.edu.cn
个人网页：http://www.cs.xmu.edu.cn/linziyu
数据库实验室网站：http://dblab.xmu.edu.cn

助教：赖明星

单位：厦门大学计算机科学系数据库实验室2011级硕士研究生（导师：林子雨）
E-mail: mingxinglai@gmail.com
个人主页：http://mingxinglai.com

欢迎访问《大数据技术基础》2013班级网站：http://dblab.xmu.edu.cn/node/423
本讲义PPT存在配套教材《大数据技术基础》，请到上面网站下载。

Department of Computer Science, Xiamen University, Sep, 2013