海量数据存储技术结课论文

移动应用程序数据同步
的可靠性，一致性和高效性

院（系）名 称：计算机学院

专 业 名 称 ：计算机技术

学 生 姓 名 ：何健

学 生 学 号 ：2017282110267

指 导 教 师 ：何水兵 教授

二〇一七年十二月

摘　要

论文来自 Younghwan Go 的《Reliable, Consistent, and Efficient Data Sync for Mobile Apps》in FAST15,February 16－19, 2015。作者构建了一个 Simba 数据同步服务，解决移动应用程序跨设备的管理数据，为用户提供各种功能，如无缝访问，协作和离线编辑。同时具有预测和处理大量的本地和网络故障，并且保持数据的一致性的优点，为移动应用程序开发人员提供了高级本地编程虚拟环境。对于移动环境要做到这一点，一个应用程序必须在移动设备的带宽和设备电池的节约使用方面要有所体现。上述要求因此给应用程序开发人员带来了巨大的负担，统一表格和对象数据（这是移动应用程序的通用需求），并以可靠性，一致性和透明性的方式，透明高效地处理数据存储和同步。在本文中，作者详细介绍 Simba 的客户端软件，它作为数据同步基础设施的网关。作者的评估显示，Simba 在快速开发功能强大的移动应用程序方面的效果，在所有故障情况下保持一致，与使用 Dropbox 开发的应用程序对比有所不同，更加高效。Simba 应用程序对于移动应用资源也有高效利用回收。

目 录

目 录.....3
1 引言.....4
2 移动应用可靠性研究.....6
3 Simba 的设计.....8
3.1 Simba 服务器（sCloud）.....8
3.2 Simba 客户端（sClient）.....8
3.2.1 Simba 客户端数据存储.....10
3.2.2 同步处理.....11
4 透明故障处理.....13
5 透明的网络效率.....14
6 结论.....15
7 参考文献.....15

1 引言

随着科技的快速发展，个人智能设备无处不在，用户现在可以随时享受各种各样的应用程序。许多这样的应用程序是有数据中心的，经常依靠基于云的资源来存储，分享和分析数据。除了用户界面和各种功能，这样的应用程序的开发人员需要构建底层的数据管理基础架构。例如，为了提供高质量的印象笔记 Evernote 等应用程序，开发人员必须构建一个支持丰富多媒体的数据管理平台笔记，关于数据和元数据的查询，协作和离线操作，同时确保可靠性和一致性面对的一切故障情况。而且，一个移动应用开发者同时还需要满足上述要求，有效利用移动设备上的有限资源如移动带宽和电池电量。如果开发人员处理解决了上述问题，那么该应用将吸引和留住用户的可能性就越大。

第一，它必须透明地处理可靠性，一致性，和高效率，很少涉及应用程序开发者，这是具有挑战性的。作为 Dropbox 的制造商也注意到，在外面提供简单的用户可能需要巨大的复杂性和努力。其次，数据同步服务必须提供一个对大多数人有利的数据模型应用;虽然文件同步是很普遍的，但实际上很多应用程序在相互依存的结构化和非结构化的基础上运行数据。一个包含表的高级数据模型而且文件对于应用程序开发人员和透明度来说是非常有用的必须适用于这个数据模型。数据同步服务必须保存，代表应用程序，结构化和非结构化之间的一致性数据在存在故障时被存储和共享。考虑一下照片共享应用程序的例子 Picasa 和 Instagram;通常这样的应用程序将存储表中的专辑信息和实际的图像文

件系统或对象存储。在这种情况下，同步服务需要确保永远不会有悬摆指针从专辑到图像。由于移动应用可能会崩溃或由于各种原因频繁停顿应用程序是在数据操作的中间（本地写或同步）发生故障时，同步服务需要重新启动，检测并恢复到一致的状态。

因此，我们建立了 Simba 来管理移动应用程序的数据，它提供了统一文件和高级抽象表。这些表可能包含两个基本类型的列（字符串，整数等）和任意大小的对象，全部通过类似于 CRUD 的界面访问。为了方便采用，接口保持类似的那些已经熟悉 iOS 和 Android 开发人员。应用可以构建跨越表和对象的数据模型 Simba 确保所有数据的可靠性，一致性，并且服务器和其他移动设备非常同步。Simba 由用于开发移动应用程序的 SDK 组成，用于移动设备的 Simba 客户端应用程序（sClient），以及 Simba 云服务器（sCloud）。所有与 Simba SDK，Simba 应用程序只与本地通信作为所有交互代理的 sClient 实例与 sCloud。

在本文中，我们关注透明度它影响 Simba 应用程序的高级抽象。因此主要讨论 sClient,整个 Simba 服务在其他地方有更详细的介绍。通过案例研究，我们展示了 Simba 如何启用，我们很快就开发了几款移动应用程序增加开发的便利性和功能。Simba 应用程序客户端的失败透明度大大受益;使用 Dropbox 编写的应用程序无法保持原子性导致破坏更新的整个数据对象在失败的情况下同步不一致的数据。受益 Simba 能够以编程方式整合延迟容忍数据传输，Simba 应用程序也展出减少网络足迹和给设备增加的机会关闭移动无线电。

2 移动应用可靠性研究

我们研究了一些流行的移动应用的可靠性和同步服务（在 Android 上），系统地介绍网络中断故障、本地应用程序崩溃和设备功率损失，并观察恢复结果。我们研究中的应用程序使用表和文件等对象，并依靠各种现有的服务，即 Dropbox，Parse 和 Kinvey 进行数据同步。

我们设置了两个 Android 具有相同应用和初始状态的设备。模拟一个网络中断，我们启动了飞行模式：（1）手动杀死应用程序，（2）拉电池出;两次碰撞测试的结果没有差异我们列出一次。对于网络中断测试，一些应用程序（例如 Hiyu，Tumblr）导致数据丢失，如果同步失败没有重新连接后立即处理。如果应用程序（或该通知）已经关闭，没有发生恢复重新开始。有些应用程序（UPM，TomDroid，Keepass2）没有通知用户同步失败。大多数应用程序都需要用户手动重新同步失败后，这个疏忽导致数据永久挂起同步，一些应用程序表现出其他形式的不一致。对于 TomDroid，如果第二个设备联系了它的服务器甚至在同步中没有更改，删除操作阻止。对于 Evernote，创建中断后手动重新同步一遍又一遍的同一个音符的多个副本。

对于崩溃测试，只有桌面的应用程序才能正确恢复因为他们完全依赖于 SQLite 的崩溃一致性。但是，带有对象的应用程序显示出问题包括腐败和不一致的行为。对于 YouTube，即使对象（视频）已成功上传，该应用程序本身丢失了帖子。 Instagram 和 Keepass2 都创建了一个局部的部分对象; Keepass2 另外未能恢复导致 a 的表数据悬挂指向对象的指

针。 Dropbox 创建了一个冲突文件与部分对象（局部损坏）和传播像 Evernote 一样损坏到第二个设备。

我们的研究表明，移动应用程序仍然失败或损坏数据尽管有大量的先前的研究，分析工具和数据同步服务。首先，大多数应用程序都有处理特别对象的问题。我们的研究中没有应用程序能够在对象更新期间正确地从崩溃中恢复。其次，不是确保正确的恢复，而是一些应用程序完全禁用对象更新。第三，在几种情况下，应用程序无法通知用户导致进一步的错误。进一步的研究激励我们透明地采取全面的方法处理数据同步服务中的故障，并提供一个有用的高级抽象到应用程序。

3 Simba 的设计

3.1 Simba 服务器（sCloud）

该服务器是一个管理数据的可扩展云存储跨多个应用程序，表和客户端。它提供了一个用于数据同步的网络协议，基于其中的模型，sClient 有责任从中提取更新服务器并推送任何本地修改代表所有设备本地 Simba 应用程序。sClient 可以注册服务器将被通知更改订阅表。

同步协议：讨论 sClient 的设计，我们需要参考服务器提供的语义网络协议。服务器有望提供耐用性，行更新的原子性和多版本并发性控制。因此，sClient 暴露于版本，其中伴随着与交换的消息中的任何数据服务器。 Simba 实现了一个版本向量的变体提供了因果一致性的并发控制语义。由于所有的客户端同步到一个中央的 sCloud，我们简化了版本方案以拥有一个版本每行数量而不是矢量。每个 sRow 都有如果从主键生成唯一标识符 IDrow 一个是存在的，或者是随机的，一个版本是 Vrow，行版本在每个服务器上递增行的更新;表中最大的行版本保持为表格版本 Vtable，允许我们快速确定哪些行需要同步。在协议中使用类似的方案。由于 Simba 支持可变大小，可能很大，对象和协议消息明确地标识对象的'部分改变的集合需要自动应用。

3.2 Simba 客户端（sClient）

sClient 允许联网的 Simba 应用程序继续有一个本地 I / O 模型显示更

容易程序;sClient 隔离来自服务器的应用程序和网络中断，并允许更好的整体用户体验。图显示了简化的客户端架构设计，它被设计为在设备范围内运行。

（1）为所有 Simba 应用程序提供访问权限到他们的表和对象数据。

（2）管理一个 devicelocal 副本以启用断开操作。

（3）确保容错性，数据一致性和行级原子性。

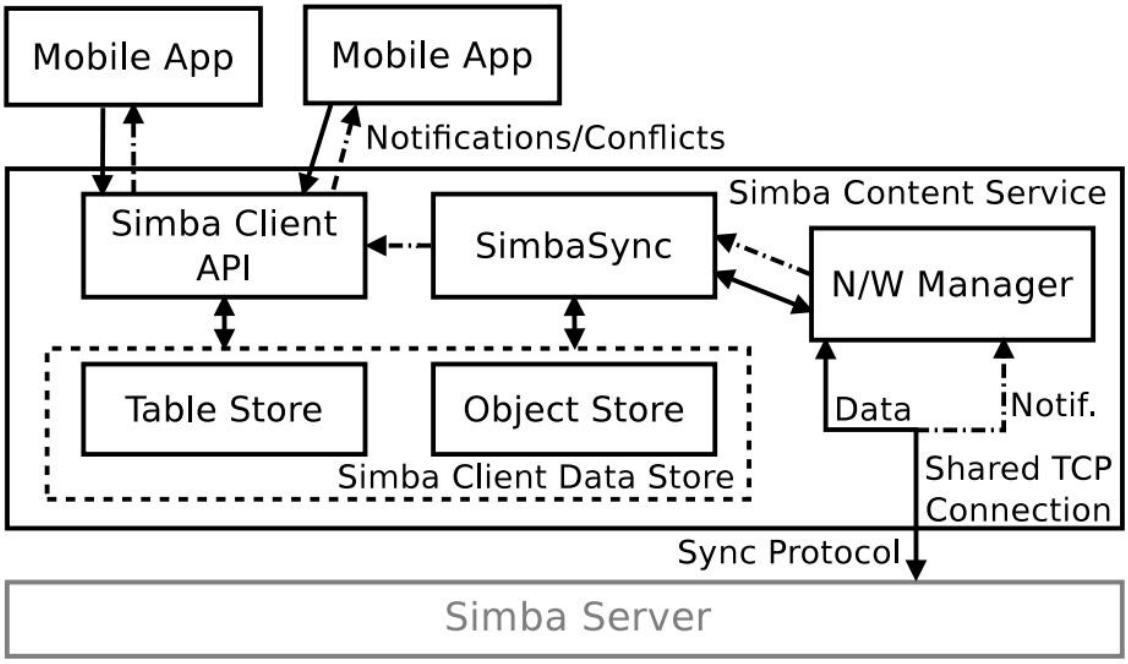
（4）通过网络执行所有同步相关的操作。

Simba 应用通过轻量级与 sClient 链接提供 Simba 客户端接口的库（sClientLib）并将客户端操作转发给 sClient;应用程序通过上传提醒事件（例如新的数据冲突）发生在后台。最后，sClient 监视应用程序的活跃度，以便存储资源可以在应用程序崩溃的情况下被释放。sClient 数据存储提供统一的抽象一个表存储和一个对象存储。 SimbaSync 使用 sCloud 执行同步处理。同步协议和本地数据共同为所有人提供透明的故障处理 Simba 应用程序。网络管理器处理所有网络连接和服务器通知 sClient;它提供了设备蜂窝的有效利用无线电通过凝聚和延迟容忍。

执行：sClient 目前正在执行然而，Android 的设计原则可以应用到其他移动平台，如 iOS。sClient 被执行作为一个叫做 Simba 内容服务的守护进程（SCS），这是由移动应用程序通过本地 RPC 访问。在 Android 上，我们使用 AIDL 接口进行通信应用程序和服务之间。另一种方法直接与应用程序链接之后是 Dropbox 和解析，但我们的方法允许 sClient 为同一台设备上的所有 Simba 应用程序调整网络流量从而受益于几个跨应用程序的优化。而使用持久连接的好处早已为人所知，个别应用程序使用 TCP 连接以频繁连接的次优方式建立和拆解。

sClient 的设计允许它使用一个持久的 TCP 连接到 sCloud 代表多个

应用程序。同样的连接也被重用由服务器发送通知，提供额外的节省，类似于 Thialfi 。行为不当的应用程序可能会产生不利影响其他 Simba 应用程序。在实践中，我们相信开发者已经有了写好行为的应用程序的动机保持用户的满意。在未来，细粒度的会计的数据，类似于 Android 的会计，可以建立进入 Simba 进一步阻止这种行为。



Simba Client Architecture

3.2.1 Simba 客户端数据存储

sClient 数据存储（SDS）负责存储应用程序数据在移动设备的永久存储，通常是内部闪存或外部 SD 卡。对于 Simba 应用程序，这意味着有能力以逻辑上统一的方式存储表格数据和对象。SDS 的主要设计目标是，并有效地支持 CRR 操作。这要求商店支持原子更新本地数据。此外，由于对象是变量而且可能很大，商店也需要支持这些对象的原子同步。频繁查询和本地修改，它必须有效地支持上游的变化检测同步，SDS

应该能够快速确定子对象的变化。对象被细分为固定大小的块存储在支持范围的键值存储（KVS）中查询。 KVS 的选择受到影响对于追加和覆盖都需要良好的吞吐量因为针对随机写入的优化对于移动来说是重要应用程序。

本地状态：sClient 保持额外的本地状态，持久性和易失性，用于同步和故障处理。两个当前的每行标志 FlagTD 和 FlagOD，用于识别本地修改的数据，上游同步所需的。为了防止部分对象同步，我们维护每行 CountOO 的数字的对象打开更新。写入交易所有打开的对象都被视为关闭。每一行都有两个持久标志 FlagSP（同步未决）和 FlagCF（冲突），跟踪其当前同步状态。最后是一个内存脏块表（DCT）跟踪已被本地修改的块，但是尚未同步。这消除了查询商店的需要在正常操作期间进行这些更改。

实现：我们利用 SQLite 来实现带有附加数据类型表格的表格存储一个对象标识符（object id）。对象存储被执行使用 LevelDB。这是一个基于 KVS 的日志结构合并（LSM）树。LevelDB 满足吞吐量标准为本地附加和更新。性 LevelDB 也有我们利用的快照功能原子同步。Android 的 LevelDB 没有本地端口所以我们使用了原来的 C ++ LevelDB 代码和 Android 的本地开发套件（NDK）。我们用一个 LevelDB 的实例保留所有表的对象以确保顺序写入更好的本地表现。由于本地状态存储在 sRow 的表格部分，SQLite 确保其一致的更新。

3.2.2 同步处理

一个 sClient 独立执行上游和下游同步。上行同步是基于指定个别表

的周期，并使用本地状态保持在（FlagTD，FlagOD）中以确定是否访问数据;这些标志在数据收集时被重置。对于脏的对象的行，块逐个阅读和直接打包成网络消息。

由于收集脏数据并将其同步到服务器可能需要很长时间，我们使用以下技术以允许前台应用程序的并发操作。首先，sClient 从 LevelDB 收集对象修改当前版本的快照。由于 sClient 同步修改对象只有在关闭和本地状态之后更新（将 CountOO 递减 1），sClient 总是确保对快照的 sRows 的一致看法。第二，我们允许 sClient 继续进行修改以前的同步操作正在进行中;这是特别的如果客户端断开连接并且正在等待同步，则有益持续时间更长。这些更改设置了 sRow 的本地旗帜，FlagTD 或 FlagOD，在随后的过程中收集同步。为此，sClient 保持一个同步待定标志为脏行设置的 FlagSP 一旦被更改被收集，并且一旦服务器指示成功就重置。如果另一个同步操作在前一个之前开始完成后，已经设置了 FlagSP 的行将被忽略。

下游同步也由 sClient 作为响应而启动向服务器发送对表的更改通知。该客户端拉动所有版本大于本地的行 Vtable，将下游数据分级到所有的块一行接收，然后逐行应用到 sClient 数据存储以增加的 Vrow 顺序。

上行同步冲突是通过确定在服务器上的 Vrow 不匹配，而对于下游的检查收到的行的本地脏标志。启用应用程序自动解析[31]或呈现给它用户，服务器返回的冲突数据在本地进行由 sClient 和相关的 Simba 应用程序通知，sClient 旨在优雅地处理冲突的更新。我们相信这大大提高了用户体验，因为应用程序不必突然中断冲突发生时的操作。

4 透明故障处理

移动应用程序在拥塞的移动网络下运行，网络中断、频繁的服务和应用程序崩溃和电池损失，移动操作系统内存管理也可以积极杀死应用程序。

故障透明度是设计目标的一个关键，它通过三个相互关联的方面来实现。首先，机制是全面的：系统检测每种可能的故障类型和恢复使系统处于定义明确的状态。其次，恢复不仅仅是一个已知的系统状态，而是遵循高水平的一致性与统一的数据模型。第三，sClient 是明智的，在交换可用性和恢复成本（其中本身在移动环境中可能是禁止的）。除非一些优化，一个 sClient 维护足够的本地元数据以避免分布状态与服务器为恢复的目的。 sClients 有状态的原因：它允许同步服务，有许多移动客户端，可能会频繁遭受故障和中央服务器，以解除其故障恢复从而提高可用性。

sClient 旨在使用状态机全面处理故障。每个成功的操作从一个 welldefined 转换 sClient 状态到另一个。不同种类的故障导致到不同的错误状态，每个状态都有明确的恢复。我们首先讨论只影响网络故障同步操作。如前所述，服务器响应上游同步可以指示成功或冲突和下游同步，可以指示任何成功或不完整。网络断开后的每个唯一状态，代表上游或下游同步无论是无故障还是故障情况。对于后者，复苏政策和行动是指定的 sClient。

5 透明的网络效率

Simba 同步是为了明智地使用移动带宽和设备电池，通过定制网络协议有两个优化：延迟容忍和合并。

延迟容忍和合并：通常，许多应用程序运行在后台作为服务，例如发送/接收电子邮件、更新天气、同步 RSS 源和新闻，并更新社交网络。sClient 被设计作为一种设备范围的服务，使多个独立的同步数据应用程序可以一起管理和转移通过一个共享的持久的 TCP 连接。进一步，Simba 支持可以延迟容忍的数据调度在每个表格的基础上进行控制。延迟容忍和合并有两个好处。 1）改进的网络足迹：允许数据传输被聚集，减少网络活动并提高设备关闭的几率收音机。来自服务器的控制消息是主题采取相同的措施。 2）改进数据压缩的范围：多个应用程序的传出数据被合并到改善压缩。

细粒度变化检测：需要整个对象如果只有一部分发生变化，则不会同步。即使数据是每行版本，sClient 都保持内部软状态（DCT）在可配置的块级检测对象变化;Simba 服务器对下游同步也是一样的。

实施：即使 sRows 是合乎逻辑的同步单元，sClient 的网络管理器打包网络包含多行数据的消息表和应用程序，以减少网络足迹。Simba 的网络协议是使用 Protobufs 来实现的有效地对结构化数据进行编码，以及 TLS 进行安全网络通信;目前的原型使用双向带有客户端和服务器证书的 SSL 认证。

6 结论

建立高质量的以数据为中心的移动应用程序，要求开发人员建立一个可靠和高效的数据管理基础设施。移动应用程序开发人员不需要担心网络和数据管理的复杂性而是能够专注于他们最擅长的事情，实现用户界面和功能，并交付很棒的应用程序给用户，作者建立了 Simba 来帮助开发者快速开发和部署强大和高效手机应用，通过它的移动客户端守护应用程序，sClient 通过灵活的策略提供后台数据同步，适合大量的移动应用程序，同时透明处理故障并有效利用移动资源。

7 参考文献

Younghwan Go, Nitin Agrawal, Akshat Aranya, and Cristian Ungureanu. Reliable, Consistent, and Efficient Data Sync for Mobile Apps. In FAST'15. February 16–19, 2015.