Git 分布式版本控制实现机制探讨

◆ 庞双玉

摘要：论文探讨分布式版本控制工具Git 实现版本控制的机制，通过比较集中式版本控制工 和分布式版本控制各自的特点，阐明分布式版本控制系统的优越性。

关键词：Git；版本控制；分布式；集中式

一、Git

版本控制工具是任何一个协作开发项目所需要使用的一个很重要的基础工具，尤其是软件开发项目，不管是个人开发或是团队协作开发，都可以通过版本控制系统获得巨大的好处。没有版本控制系统的话，代码可能被别人或自己不小心覆盖或遗失、也不知道是谁因为什么原因改了这段代码、也没办法可以复原回前几天的修改。有了版本控制系统，开发人员只要将每次代码的变更都记录（Commit）起来，并且通过版本控制系统进行更新[1]。

Git 是一个分布式的版本控制系统，可以有效、高速的处理从很小到非常大的项目版本管理。所谓分布式，就是一个多任务系统，将一个业务拆分，部署在不同机器上。Git 实现了版本库的本地存储，每台机器都是等价的，都有工作区和版本库，开发人员无需联网就可直接在本地进行工作，工作成果可直接保存到本地版本库，任何两台机器都可通过相互交换各自的版本库来更新信息。

二、Git 实现机制

Git 在实现机制上，有三个关键点，分别表述如下：

● 版本库的本地化

Git 将绝大多数的资源保存在本地，Git 在本地保存了绝大部分的代码仓库，用户无论是复制，更新，还是查询，只需要跟本地文件打交道，无需连接网络到远程服务器端。Git 在本地磁盘上就保存着所有当前项目的历史更新。项目更新的提交可以在网络便利的时候进行，实现了版本库的离线提交。

版本库本地化，支持离线提交，相对独立不影响协同开发。每个开发者都拥有自己的版本控制库，在自己的版本库上可以任意的执行提交代码、创建分支等行为。例如，开发者认为自己提交的代码有问题的时候因为版本库是自己的，回滚历史、反复提交、归并分支并不会影响到其他开发者。

本地化保证了更少的版本库互相影响。Git 对于每个工程只会产生一个 .git 目录，这个工程所有的版本控制信息都在这个目录中，不会像 SVN 那样在每个目录下都产生 .svn 目录。

把内容按元数据方式存储，完整克隆版本库。所有版本信息位于.git 目录中，它是处于你的机器上的一个克隆版的版本库，它拥有中心版本库上所有的东西，例如标签、分支、版本记录等。

本地化实现了支持快速切换分支方便合并，合并性能好。在同一目录下即可切换不同的分支，方便合并，且合并文件速度比 SVN 快。分布式版本库，无单点故障，内容完整性好。内容存储使用的是 SHA-1 哈希算法。这能确保代码内容的完整性，确保在遇到磁盘故障和网络问题时降低对版本库的破坏。

● 文件的快照保存

Git 并不保存用户提交内容的前后差异，而是存储在某个时间的快照。快照的英文是 snapshot，snapshot 的含义是关于指定数据集合的一个完全可用拷贝，该拷贝包括相应数据在某个时间点（拷贝开始的时间点）的映像。快照可以是其所表示的数据的一个副本，也可以是数据的一个复制品[2]。

用户在提交文件的时候，如果文件内容没有发生变化，Git 只是保存一个指向这个文件上次保存的索引，如果发生变化，则保存文件在这个时间点的快照。Git 的工作方式如下图所示：

图 1 Git 的备份机制

Git 会把出现变更的文件直接拷贝，形成新的 blob（二进制大文件对象），而非与上一个版本的 diff（区别）。所以一旦需要查看某版本直接 load 即可，而其他差异版本控制需要做 merge，所以快。并非每个当前版本都需要做备份，如果没有改变，那么快照其实是链接上一个版本。Git 会在隐藏目录 .git 里存在 object 里，定期会优化，保证快照空间，和读取时间的平衡。

● 基于内容的寻址空间

Git 是一套内容寻址（content-addressable）文件系统，Git 采用 HashTable 的方式进行查找，通过简单的存储键值对（key-value pair）的方式来实现内容寻址的，而 key 就是文件（头 + 内容）的哈希值（采用 sha-1 的方式，40 位），value 就是经过压缩后的文件内容。如下公式所示：

Key = sha1(file_header + file_content)

Value = zlib(file_content)

Git 对象的类型包括：BLOB、tree 对象、commit 对象。BLOB 对象可以存储几乎所有的文件类型，全称为 binary large object，顾名思义，就是大的二进制表示的对象，这种对象类型和数据库中的 BLOB 类型（经常用来在数据库中存储图片、视

频等）是一样的，当作一种数据类型即可；tree 对象是用来组织 BLOB 对象的一种数据类型，你完全可以把它想象成二叉树中的树节点，只不过 Git 中的树不是二叉树，而是“多叉树”；commit 对象表示每一次的提交操作，由 tree 对象衍生，每一个 commit 对象表示一次提交，在创建的过程中可以指定该 commit 对象的父节点，这样所有的 commit 操作便可以连接在一起，而这些 commit 对象便组成了提交树，branch 是这个树中的某一个子树。Commit 对象中包含了历次提交信息。Git 所涉及到的对象数据结构信息和关系如下图所示：

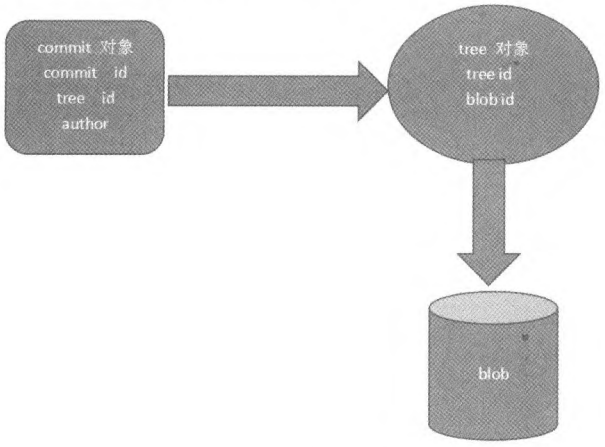

图 2 Git 对象关系图

三、集中式版本控制系统

集中式版本控制系统是出现较早的版本控制系统，典型代表是 SVN，集中式版本控制系统的最大特点是所有的开发资料都集中存放。所有用户提交的资料全部放在服务器中，用户想访问某个时期的资料，必须通过网络连接到服务器。

集中式版本控制系统一般具有如下的特点：高度依赖于服务器；高度依赖网络，工作前必须跟服务器发生通信；按照原始文件存储，提交较大；用户创建的分支是共享的；提交的文件会直接记录到版本库；提交的文件是基于差异记录的。

集中式版本控制系统与 Git 相比，集中式版本控制系统更适合于项目管理，集中式版本控制系统按目录存放文件，项目管理各个环节的文件，按目录存放在服务器中，便于项目管理者随时查看。

四、Git 的优越性

首先 Git 是一个分布式的系统，分布式意味着，大部分的负载和工作由多个机器共同完成，这使得 Git 具有集中式版本控制系统所没有的优势，具体如下：

（1）用户无需联网，即可获得历史版本和代码库。

（2）用户创建的分支为单独享有，避免了 SVN 中共同享有分支所造成的复杂度。

（3）Git 采用快照模式来记录文件，而不是差异模式，并且采用校验和压缩，因此恢复文件的速度非常快。

（4）Git 采用的代码库是分布在各个用户机上的，跟集中式版本控制系统相比，更加安全。

五、结语

在信息技术世界，正在从互联网走向分布式和云计算的时代，Git 的出现正是顺应了这一趋势，因此，Git 具有分布式系统所具有的强大优越性，尤其是对于大规模的软件开发，Git 对于代码的管理是非常高效的，其运行在 Linux 系统上的特性，也使其更利于软件开发和部署，相信 Git 会越来越优化。

参考文献

[1] 蒋鑫著 .Git 权威指南 [M]. 机械工业出版社 ,2011,06.

[2]（美）罗力格（美）麦卡洛著，王迪，等 .Git 版本控制管理 [M]. 人民邮电出版社 ,2015,03.

（作者单位：深圳技师学院电子信息技术系）