Github 的清点对象算法

作者： 阮一峰

日期： 2015年9月30日

使用 Github 的时候，你有没有见过下面的提示？

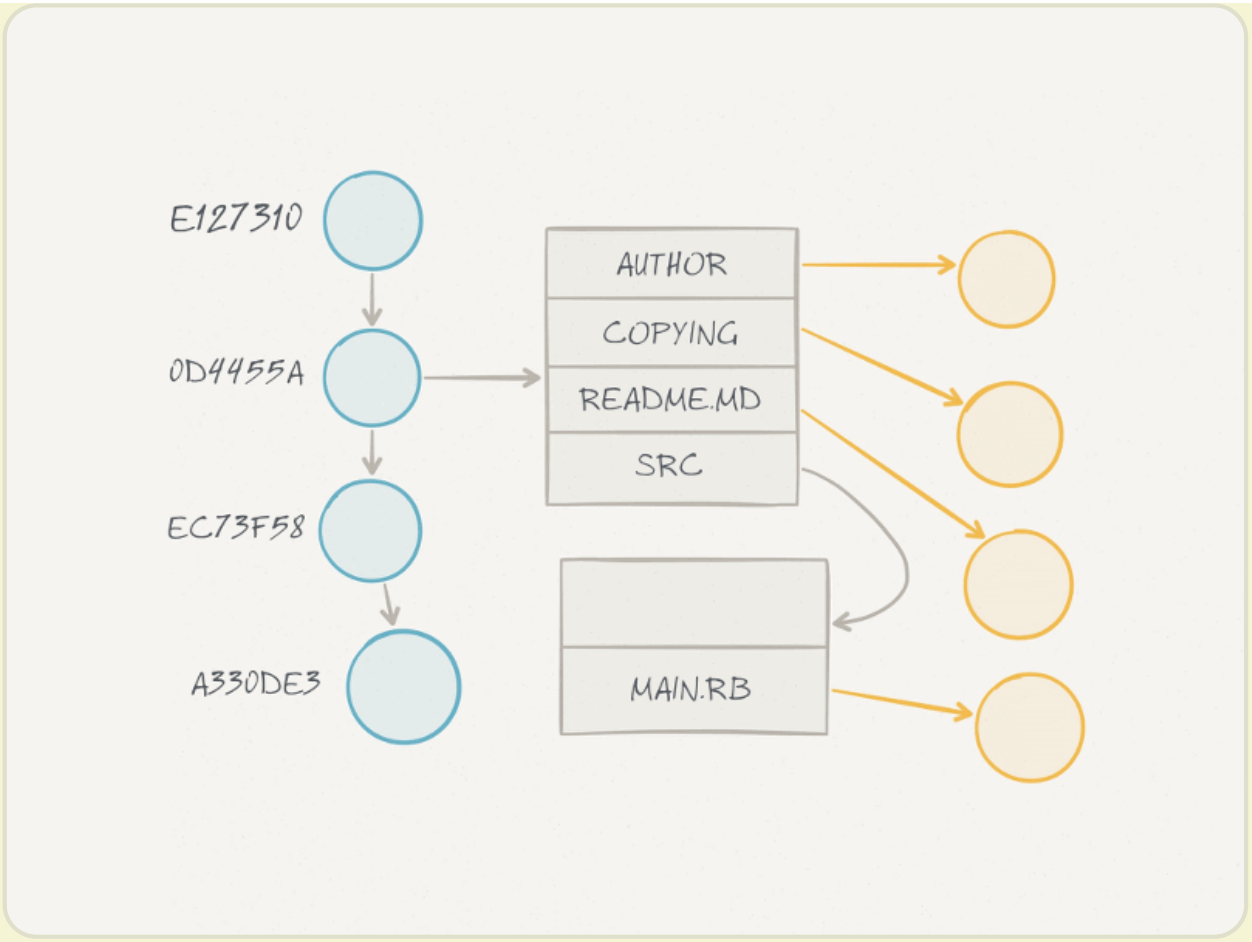
```
$ git clone https://github.com/torvalds/linux
Cloning into 'linux'...
remote: Counting objects: 4350078, done.
remote: Compressing objects: 100% (4677/4677), done.
Receiving objects:   4% (191786/4350078), 78.19 MiB | 8.70 MiB/s
```

这段提示说，远程代码库一共有4350078个对象需要克隆。

这就叫"清点对象"（**counting objects**），**Github**需要实时计算出来，需要克隆的对象总数。

这个过程非常慢，根据Github的披露，像Linux kernel这样巨大的库，清点一次需要8分钟！也就是说，发出`git clone`命令后，会干等八分钟，然后才会开始真正的数据传输。这当然是无法忍受的。Github团队一直想解决这个问题。

后来，他们终于发现了一种新的算法，现在清点一次只要3毫秒！

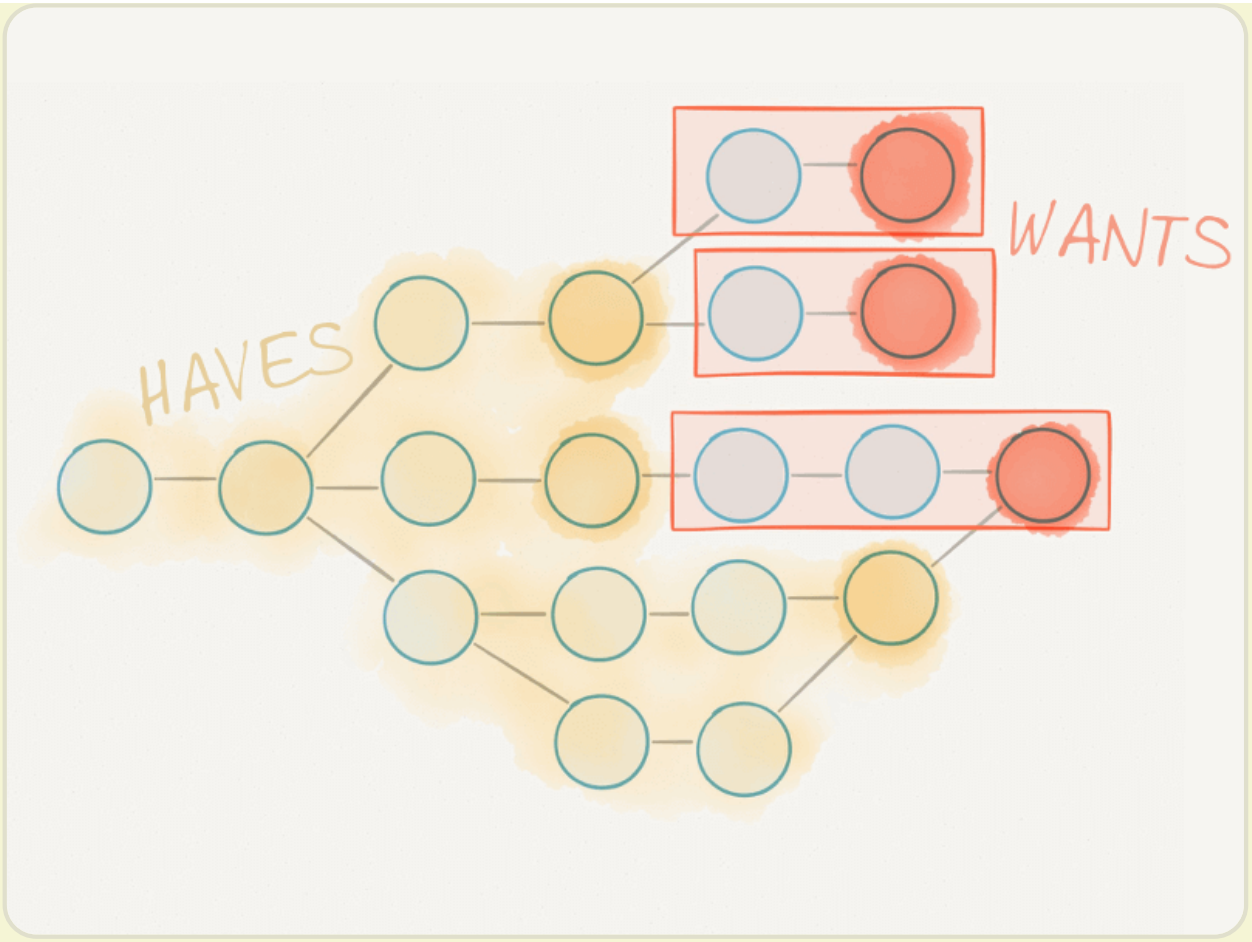


为了理解这个算法，你必须先知道，什么是Git的对象。简单说，对象就是文件，最重要的对象有三种。

- 快照对象（Commit）
- 目录对象（Directory）
- 文件对象（File）

每次提交代码的时候，会生成一个commit对象，里面有对应的当前"目录对象"的名字。"目录对象"保存了代码根目录所含有的子目录和文件信息。每一个子目录就是另一个"目录对象"，每一个文件则是"文件对象"，里面是具体的文件内容。

所以，"清点对象"就是清点各种**commit**、目录、文件等。git clone和git fetch操作都需要清点对象，因为需要知道，到底下载哪些对象文件。



清点对象的原始算法如下。

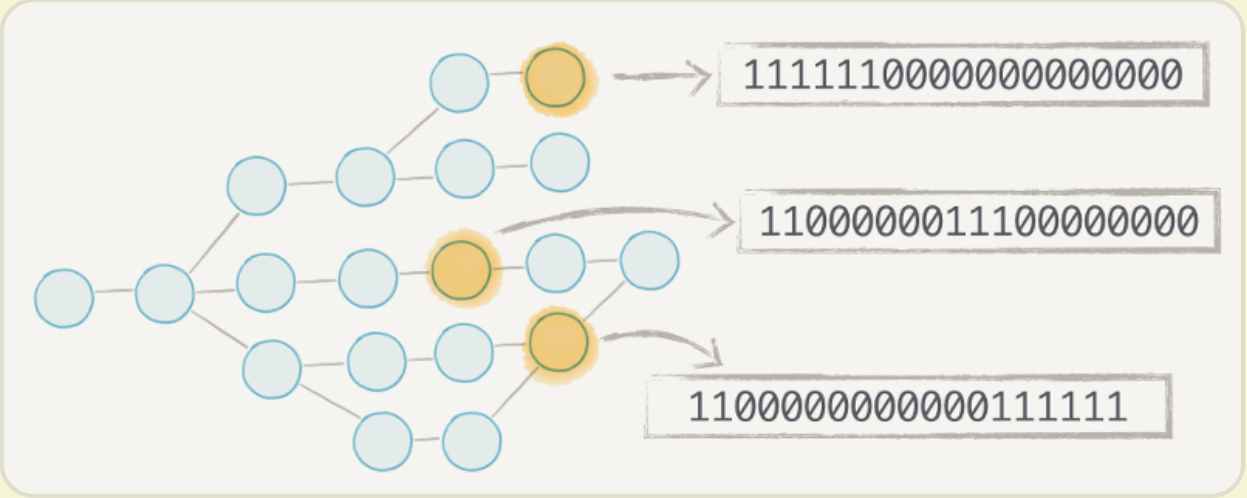
1. 列出本地所有分支最新的一个commit
2. 列出远程所有分支最新的一个commit
3. 两者进行比较，只要有不同，就意味着分支发生变动
4. 每一个发生变动的commit，都清点其中具体变动的子目录和文件
5. 追溯到当前commit的父节点，重复第四步，直至本地与远程的历史一致为止
6. 加总所有需要变动的对象

上面的过程说明，"清点对象"是一个文件遍历算法，变动的对象会被一一清点到，这就意味着大量的文件读操作。对于大型代码库来说，这个过程非常慢。

Github团队想到的新算法，是建立一个**Bitmap**索引，即为每一个**commit**生成一个二进制值。

打开本地Github仓库的.git/objects/pack/目录，你会看到一个索引文件和一个数据文件，它们就是Bitmap。简单说，这两个文件索引了当前代码库的所有对象，然后使用一个二

进制值代表这些对象。有多少个对象，这个二进制值就有多少位。它的第n位，就代表数据文件里面的第n个对象。



每个commit都会有一个对应的二进制值，表示当前快照包含的所有对象。这些对象对应的二进制位都为1，其他二进制位都为0。

这样做的好处是，不用读取commit对象，只要读取这个二进制值，就会知道当前commit包含了哪些节点。更妙的是，两个二进制值只要做一次XOR运算，就会知道哪些位（即哪些对象）发生了变动。而且，因为新的对象总是添加到现有二进制位的后面，所以只要读取多出来的那些位，就知道当前commit比上一次commit多出了哪些对象。

这样一来，"清点对象"就变成了二进制值的比较运算，因此速度极快。进一步的介绍，请参看官方文档《Bitmap的解释》，《Bitmap的格式》。

目前，Github的生产环境已经部署了这套算法，用户再也不用为了清点对象，而苦苦等待了。而且，Github团队还把它合并进了Git，这意味着，从此所有Git实现都可以使用Bitmap功能了，因此将来肯定还会有更多好玩的用法出现。

（完）