git gc

> 适用于`存在大文件`，且`多次提交都只是轻微改动该大文件`的场景，因为这些提交都会生成大小相近的大文件`blob对象`，非常占用磁盘空间

1. Git最初向磁盘中存储对象使用`松散`的格式，后续会将多个对象打包为一个二进制的`包文件`（`packfile`），以`节省磁盘空间`
2. `.pack`文件存储了`对象的内容`
3. `.idx`文件存储了`包文件`的`偏移信息`，用于`索引具体的对象`
4. 打包对象时，查找命名和大小相近的文件，保留文件`不同版本之间的差异`（`最新一版保存完整内容`，访问频率最高）
5. `verify-pack -v *.idx`：查看压缩包内容

添加随机(压缩率低)字符大文件`bigfile`并提交

```
$ git init
Initialized empty Git repository in /home/zhongmingmao/demo/.git/

$ dd if=/dev/urandom of=./bigfile bs=1k count=10240 # use urandom , hard to be compressed
10240+0 records in
10240+0 records out
10485760 bytes (10 MB, 10 MiB) copied, 0.84948 s, 12.3 MB/s

$ du -sh bigfile
10M bigfile

$ du -sh .git
96K .git

$ du -sh .git/objects
12K .git/objects
```

```

$ git add bigfile

$ du -sh .git
11M .git

$ du -sh .git/objects
11M	.git/objects

$ git commit -m 'add bigfile'
[master (root-commit) 6091c0e] add bigfile
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 bigfile

$ git cat-file -p master^{tree}
100644 blob 41b939bb0968e9a0ff69fcc50007107d94d9d3c8 bigfile


$ git cat-file -s 41b939bb0968e9a0ff69fcc50007107d94d9d3c8
10485760 # 10MBytes

$ du -sh .git/objects/41/b939bb0968e9a0ff69fcc50007107d94d9d3c8
11M .git/objects/41/b939bb0968e9a0ff69fcc50007107d94d9d3c8
```

轻微改动`bigfile`后提交

```
$ echo 'zhongmingmao' >> bigfile

$ git commit -am 'echo zhongmingmao >> bigfile'
[master 5834cad] echo zhongmingmao >> bigfile
 1 file changed, 0 insertions(+), 0 deletions(-)

$ du -sh .git # double size!!
21M .git
```

```

$ du -sh .git/objects
21M .git

$ git cat-file -p master^{tree}
100644 blob ce4134c5eecf2b379d2eac3f812409f1b602cd85 bigfile

$ git cat-file -s ce4134c5eecf2b379d2eac3f812409f1b602cd8
10485773

$ du -sh .git/objects/ce/4134c5eecf2b379d2eac3f812409f1b602cd85
11M .git/objects/ce/4134c5eecf2b379d2eac3f812409f1b602cd85

```

`git gc`压缩，`41b939`参照`ce4134`（旧版本参照新版本，`最新版本保存完整内容`）

```
$ git gc
Counting objects: 6, done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), done.
Total 6 (delta 1), reused 0 (delta 0)

$ find .git/objects -type f
.git/objects/info/packs
.git/objects/pack/pack-73a2d8537fba854db8f9f413799b0d5274a52135.pack
.git/objects/pack/pack-73a2d8537fba854db8f9f413799b0d5274a52135.idx

$ du -sh .git
11M .git

$ du -sh .git/objects/*
8.0K  .git/objects/info
11M   .git/objects/pack
```

```
$ git verify-pack -v .git/objects/pack/pack-73a2d8537fba854db8f9f413799b0d5274a52135.idx
5834cadac71e81cbcba954f49611feeaaa92cd3a commit 249 153 12
6091c0e06a518c510e70d09a6892b122672ab837 commit 184 121 165
ce4134c5eecf2b379d2eac3f812409f1b602cd85 blob   10485773 10488983 286
70f4ea5a7ec96577205e4f8a9e1daba81f4ed6f7 tree   35 46 10489269
59c311f2a94eb4936ccc39fc1dae89fb3660ffe7 tree   35 46 10489315
41b939bb0968e9a0ff69fcc50007107d94d9d3c8 blob   327 249 10489361 1 ce4134c5eecf2b379d2eac3f812409f
non delta: 5 objects
chain length = 1: 1 object
.git/objects/pack/pack-73a2d8537fba854db8f9f413799b0d5274a52135.pack: ok

$ git cat-file -p master^{tree}
100644 blob ce4134c5eecf2b379d2eac3f812409f1b602cd85 bigfile

$ git cat-file -s ce4134c5eecf2b379d2eac3f812409f1b602cd85
10485773

```

git prune

> git clone 会下载 整个 项目提交历史，如果 曾经 添加过 大文件 ，后续 git rm 了，每次 git clone 依旧会下载那个大文件对应的 blob对象 ，将要介绍的方法会 重写提交历史 ，请 谨慎使用

1. git count-objects -v ：快速查看 object databse 的概要情况
2. git rev-list --all --objects ：显示 所有commit及其所关联的所有对象
3. git log --branches -- $filename ：查看哪些对 $filename 做出了修改的 commit
4. git filter-branch --index-filter 'git rm --ignore-unmatch --cached $filename' -- $sha1
    - --index-filter ：不 checkout 到 working directory ，只修改 index 的文件， 速度快很多
    - --ignore-unmatch ：尝试 删除的模式无法匹配 时， 不提示错误

    - `--cached`：从`index`删除
5. `git prune --expire now`：立马删除`object database`中`不可达的对象`

`dd02204`引入了大文件`bigfile`，后续在`f161ef0`删除了`bigfile`，但`object database`大小没有减少，`git gc`也无法减少磁盘占用

```
$ git log --oneline --decorate --graph --all
* cc12440 (HEAD -> master) C4
* db72b36 C3
* 5eebbac C2
* d49df27 C1
* 19460ce C0

$ dd if=/dev/urandom of=./bigfile bs=1k count=10240
10240+0 records in
10240+0 records out
10485760 bytes (10 MB, 10 MiB) copied, 0.876271 s, 12.0 MB/s

$ git add bigfile && git commit -m 'add bigfile'
[master dd02204] add bigfile
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 bigfile

$ du -s .git/objects
10364	.git/objects

$ git rm bigfile

rm 'bigfile'

$ git commit -m 'git rm bigfile'
[master f161ef0] git rm bigfile
 1 file changed, 0 insertions(+), 0 deletions(-)
 delete mode 100644 bigfile
```

```

$ du -s .git/objects
10372    .git/objects

$ git gc
Counting objects: 15, done.
Compressing objects: 100% (13/13), done.
Writing objects: 100% (15/15), done.
Total 15 (delta 5), reused 0 (delta 0)

$ git count-objects -v
count: 0
size: 0
in-pack: 15
packs: 1
size-pack: 10245 # 10M
prune-packable: 0
garbage: 0
size-garbage: 0
```

查找`object database`中的大文件`bigfile`对应的`blob对象`的`ID`
如果非常清除大文件`bigfile`是从哪一个`commit`引入的，可以直接跳到`filter-branch`

```
$ git verify-pack -v .git/objects/pack/pack-695de68a1a8a850a46c18f5619bd51e5efbd8cdc.idx
f161ef0b13c784cefca1316578e671d4bdc375ec commit 235 152 12
dd02204bdd017cee28cf1fa156f161cf844951a3 commit 232 150 164
cc1244046a2534b2097819439ab705c579396937 commit 71 82 314 1 f161ef0b13c784cefca1316578e671d4bdc
db72b3604bad5a3f76083f4140517e588e6032cf commit 223 145 396
5eebbac0f01a8c179f7fd4ca3b800f1cef6d28cd commit 223 145 541
d49df27f3df12877e347d78b6441254bcc718296 commit 223 144 686
19460cee4babe9a801cd96148aac236121a3a2cb commit 175 115 830
e69de29bb2d1d6434b8b29ae775ad8c2e48c5391 blob   0 9 945
814058ae67e19720f67c9a0cc7ee5550ca1fa4a0 tree   185 85 954
```

```
7b3dc4c67a91b8d838b3ca02421de3a50dcb68d2 tree   6 16 1039 1 814058ae67e19720f67c9a0cc7ee5550ca1fa
27323e81a35521d9fd8705885b1593cf7f3a953b blob   10485760 10488970 1055
d8b75f412dd1563d651c6f6e222ddde31b1befda tree   5 18 10490025 1 814058ae67e19720f67c9a0cc7ee5550c
8cc551af62c753b24e3f2710b9316b0c02e321ef tree   4 14 10490043 2 d8b75f412dd1563d651c6f6e222ddde31
fdfe0e3dc93fa9340d58cd0c0a406e07769c42d9 tree   4 14 10490057 2 d8b75f412dd1563d651c6f6e222ddde31
ecac4a6cf836d86ef14a942fe00287a26534261c tree   30 41 10490071
non delta: 10 objects
chain length = 1: 3 objects
chain length = 2: 2 objects
.git/objects/pack/pack-de5573c64f095927d05eaad6c39cd98e53ee4c93.pack: ok

$ git cat-file -s 27323e81a35521d9fd8705885b1593cf7f3a953b
10485760
```

显示`27323e`对应的`文件名`

```
$ git rev-list --all --objects | grep 27323e
27323e81a35521d9fd8705885b1593cf7f3a953b bigfile
```

显示对`bigfile`做出过修改的`commit`

```
$ git log --oneline --branches -- bigfile
f161ef0 git rm bigfile
dd02204 add bigfile
```

重建提交历史

```
$ git filter-branch --index-filter 'git rm --ignore-unmatch --cached bigfile' -- dd02204^..
Rewrite dd02204bdd017cee28cf1fa156f161cf844951a3 (1/2) (0 seconds passed, remaining 0 predicted)    rm 'bi
Rewrite f161ef0b13c784cefca1316578e671d4bdc375ec (2/2) (0 seconds passed, remaining 0 predicted)
Ref 'refs/heads/master' was rewritten

$ git log --oneline --decorate --graph --all
* 7c504c6 (HEAD -> master) git rm bigfile
* 98c862a add bigfile
| * f161ef0 (refs/original/refs/heads/master) git rm bigfile
| * dd02204 add bigfile
|/
* cc12440 C4
* db72b36 C3
* 5eebbac C2
* d49df27 C1
* 19460ce C0

$ git count-objects -v
count: 2
size: 8
in-pack: 15
packs: 1
size-pack: 10245 # 10M
prune-packable: 0
garbage: 0
size-garbage: 0
```

删除`.git/refs/original`和`.git/logs`（使得`bigfile的blob对象`成为`不可达`）然后再清除`不可达对象`

```
$ rm -rf .git/refs/original

$ rm -rf .git/logs

$ git gc
Counting objects: 13, done.
```

```
Compressing objects: 100% (9/9), done.
Writing objects: 100% (13/13), done.
Total 13 (delta 5), reused 8 (delta 2)

$ git count-objects -v
count: 4
size: 10256 # 10M

in-pack: 13
packs: 1
size-pack: 2
prune-packable: 0
garbage: 0
size-garbage: 0

$ git prune --expire now

$ git count-objects -v
count: 0
size: 0
in-pack: 13
packs: 1
size-pack: 2
prune-packable: 0
garbage: 0
size-garbage: 0

$ du -sh .git/objects
24K .git/objects # 10M -> 24K

$ git log --oneline --decorate --graph --all
* 7c504c6 (HEAD -> master) git rm bigfile
* 98c862a add bigfile
* cc12440 C4
* db72b36 C3
* 5eebbac C2
```

```
[illegible]
* d49df27 C1
* 19460ce C0
```