Repofactoring

Getting performance back for your repositories

Charles Bailey, Bloomberg LP

@hashpling

Git doesn’t “handle” binary objects

Binary delta compression

```
$ git init -q
$ cp /usr/lib/x86_64-linux-gnu/libc.a .
$ git add libc.a && git commit -q -m "Add libc.a"

$ git cat-file --batch-all-objects\
>  --batch-check='%(objectname) %(objecttype) %(objectsize:disk)'
0a77b5fd295ec58c3b17055336862178cddb1008 tree 51
24452d704533c07939713414fb92cbcd0b3de5f9 commit 132
52dfffd8c1bb7d1a2d9017bfdb0773fe2fff19f3 blob 1332815
```

```
$ git gc --quiet
$ git cat-file --batch-all-objects\
>  --batch-check='%(objectname) %(objecttype) %(objectsize:disk)'
0a77b5fd295ec58c3b17055336862178cddb1008 tree 45
24452d704533c07939713414fb92cbcd0b3de5f9 commit 128
52dfffd8c1bb7d1a2d9017bfdb0773fe2fff19f3 blob 1104406
```

```
$ ar d libc.a dl-load.o
$ git add libc.a
$ git cat-file --batch-all-objects\
>  --batch-check='%(objectname) %(objecttype) %(objectsize:disk)'
0a77b5fd295ec58c3b1705536862178cddb1008 tree 45
24452d704533c07939713414fb92cbcd0b3de5f9 commit 128
52dfffd8c1bb7d1a2d9017bfdb0773fe2fff19f3 blob 1104406
a26dbdf72f9bac1fa4a6aca733fdd24225d61d36 blob 1318422
```

```
$ git gc --quiet
$ git cat-file --batch-all-objects\
>  --batch-check='%(objectname) %(objecttype) %(objectsize:disk)'
0a77b5fd295ec58c3b1705536862178cddb1008 tree 45
24452d704533c07939713414fb92cbcd0b3de5f9 commit 128
52dfffd8c1bb7d1a2d9017bfdb0773fe2fff19f3 blob 1104406
a26dbdf72f9bac1fa4a6aca733fdd24225d61d36 blob 5207
```

Filtering out a “base” object

```
$ git gc --quiet
$ git cat-file --batch-all-objects\
>  --batch-check='%(objectname) %(objecttype) %(objectsize:disk)' |
> grep blob
52dfffd8c1bb7d1a2d9017bfdb0773fe2fff19f3 blob 1104406
a0f087027f1ca796f5722be17fcee8e18ba24298 blob 5219
a26dbdf72f9bac1fa4a6aca733fdd24225d61d36 blob 5207
b3ca316fef362478af802df6dc8e988cab414f2f blob 5243
```

```
$ git update-ref -d refs/original/refs/heads/master
$ git reflog expire --expire=now --all
$ git gc --prune=now --quiet
```

```
$ git gc --quiet
$ git cat-file --batch-all-objects\
>  --batch-check='%(objectname) %(objecttype) %(objectsize:disk)' |
> grep blob
a0f087027f1ca796f5722be17fcee8e18ba24298 blob 5202
a26dbdf72f9bac1fa4a6aca733fdd24225d61d36 blob 1091029
b3ca316fef362478af802df6dc8e988cab414f2f blob 5212
```

Analyzing usage

Find the “on disk” size of all objects

```
$ git cat-file --batch-all-objects\
>  --batch-check='%(objectname) %(objecttype) %(objectsize:disk)'\
>  --buffer
```

Select blobs

awk

```
if ($2 == "blob") { print }
```

Follow the delta chains

Identify delta chains by the root object in each chain

Assign the *mean* size to each member of the same delta chain.

(Perl one-liner too big for this slide.)

Choose the “big” blobs

awk (again)

```
if ($3 >= 100000) { print }
```

Determine what the file is

```
while read id
do
        git cat-file "$id" | file -
done
```

Determine where the file is

```
git log -c --raw --no-abbrev --pretty=commit\ %h
```

```
 commit af40944

:000000 100644 0000000... fc6fe17... A  Documentation/RelNotes/2.6.3
:100644 100644 4585103... c2e2a94... M  Documentation/git.txt
:100755 100755 7876709... abfdd9c... M  GIT-VERSION-GEN
:120000 120000 0223580... 21b4dd6... M  RelNotes
```

Rewriting history

`git filter-branch`

- Comes with Git
- Save the commit map for later
- Perform other filtering at the same time
- Can eliminate empty commits

`bfg` repo cleaner

- Separate tool
- Requires Java runtime environment
- Fast!
- Preserve the exact shape of the commit graph

Recording the commit map

Using a commit filter

```
tmp=$(git_commit_non_empty_tree) &&
{ [[ $tmp = "$GIT_COMMIT" ]] ||
        echo >&7 "$GIT_COMMIT $tmp"; } &&
echo $tmp
```

Incremental filter-branch

Needs a parent filter to ensure continuity

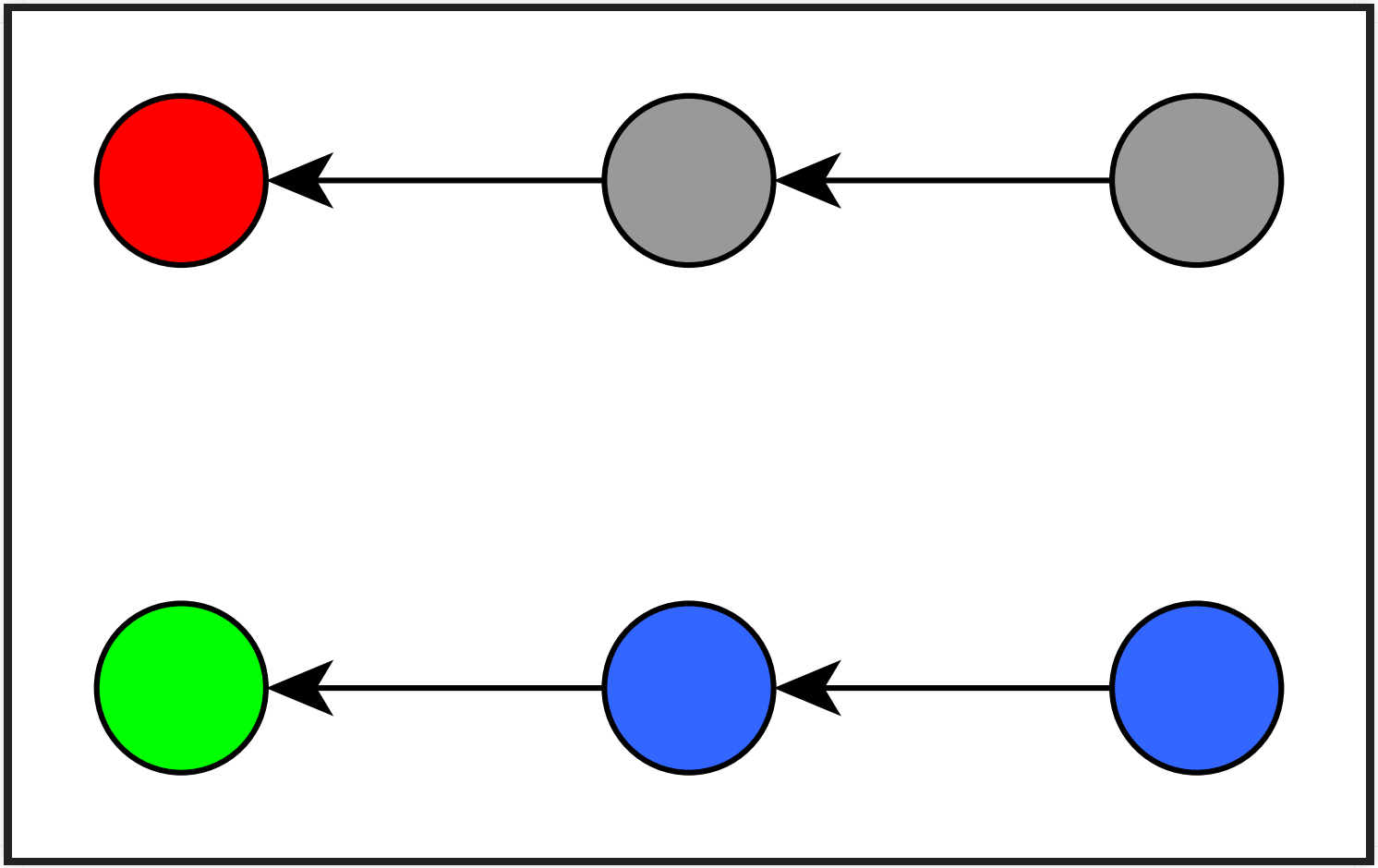
Bringing the team along with you

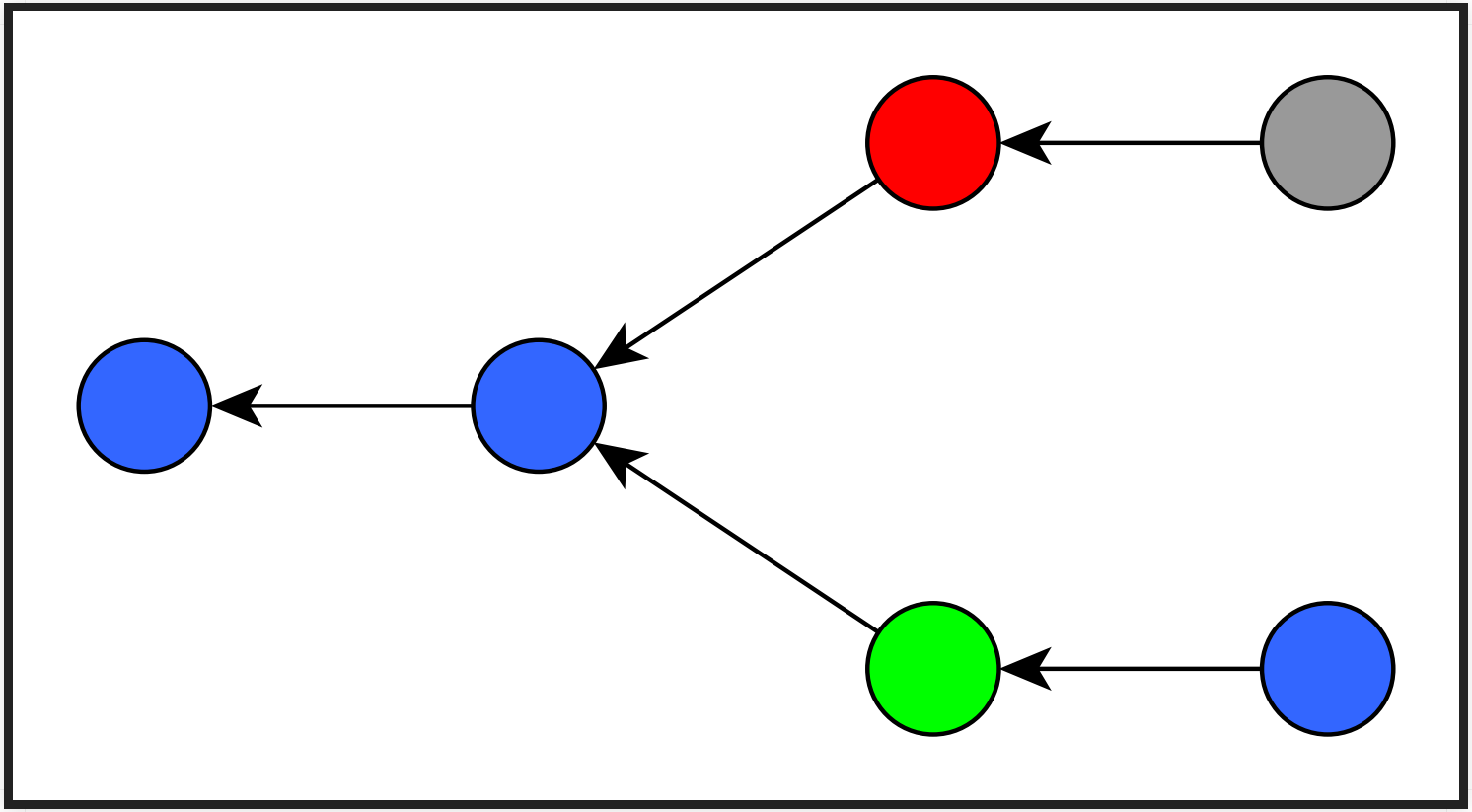
Steps

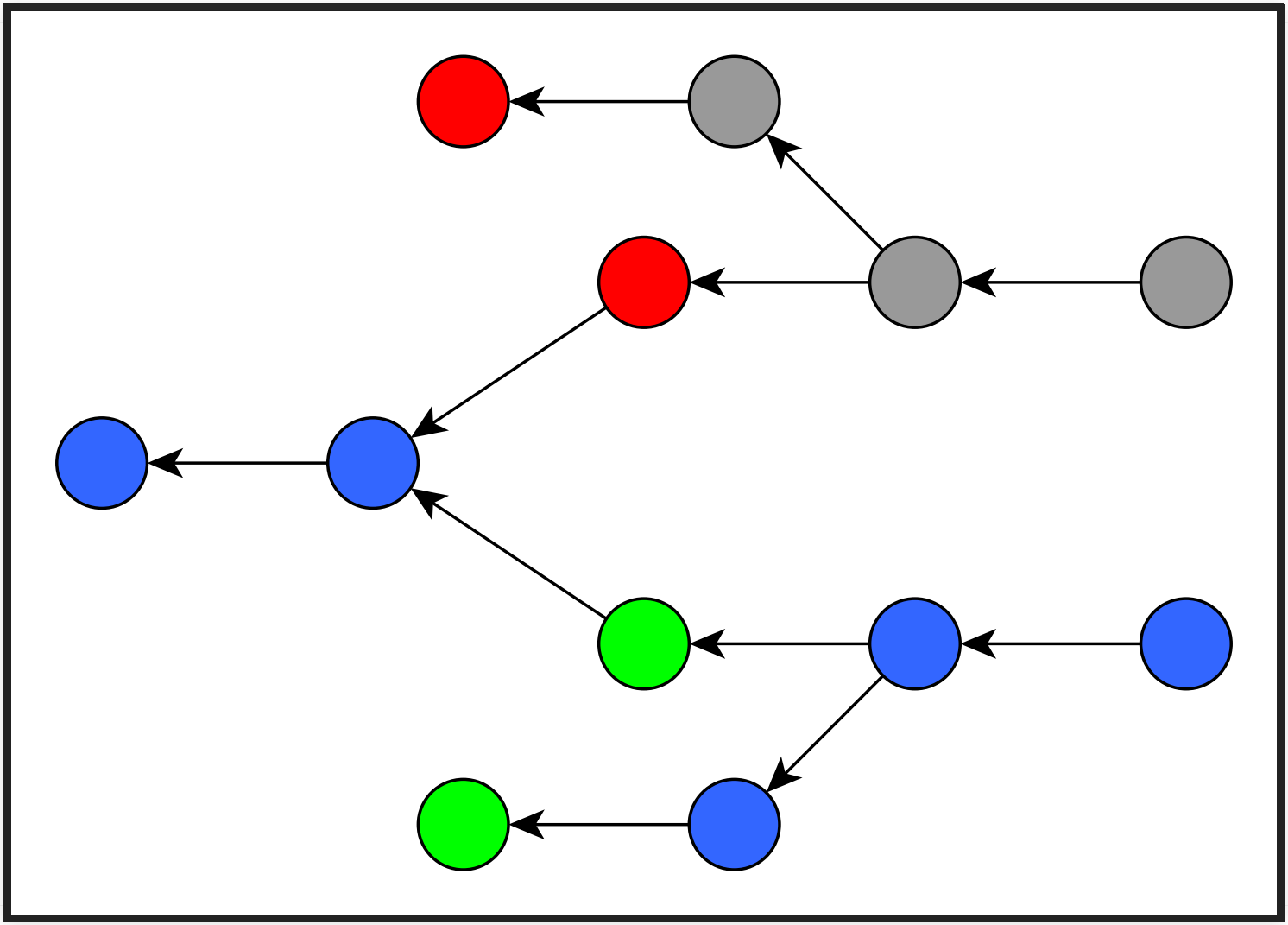
- Determine if branch needs migrating
- Find an old history base commit...
  ...and the corresponding new history commit
- Perform the migration

Does this branch need migrating?

- Use “virtual” roots







Enumerating virtual roots

- List only the “bad” part of history

```
git rev-list --reverse --parents
  "${old_history_tips[@]}"
  --not
  "${new_history_tips[@]}"
```

- Discard a commit if we already have any parent

`git merge-base --is-ancestor`

Finding a suitable base

- **`git merge-base`** with tips of the old history (bfg)
- Walk history until we find an old history commit (f-b)

Mapping to the new history

- **`git describe --contains`**
  `prefilter-master~10^2~39^2`
  maps to
  `postfilter-master~10^2~39^2`
- Look up found base in the commit map

Migrating to the new history

- Guided `rebase`
- Scripted `filter-branch`

- Find large objects in the repository

- Find large objects in the repository
- Rewrite history to optimize size

- Find large objects in the repository
- Rewrite history to optimize size
- Migrate to the new history