git2go vs go-git

Jeremiah Mahler
<jmmahler@gmail.com>

April 27, 2020

Contents

1 Introduction

This paper compares two Golang libraries for interfacing with Git: git2go[1] and go-git.[2] The git2go library provides C bindings to libgit2[3] whereas the go-git library is written entirely in Go. To achieve a real world comparision a single project was first implemented using go-git[4] and then re-implemented using git2go.[5]

[1] *github.com/libgit2/git2go*. [Online; accessed 2-March-2020]. URL: `https://github.com/libgit2/git2go`.
[2] *github.com/src-d/go-git*. [Online; accessed 2-March-2020]. URL: `https://github.com/src-d/go-git`.
[3] *libgit2.org*. [Online; accessed 2-March-2020]. URL: `https://libgit2.org/`.
[4] *github.com/jmahler/mgmirr/tree/go-git*. [Online; accessed 2-March-2020]. URL: `https://github.com/jmahler/mgmirr/tree/go-git`.
[5] *github.com/jmahler/mgmirr/tree/git2go*. [Online; accessed 2-March-2020]. URL: `https://github.com/jmahler/mgmirr/tree/git2go`.

2 Installation

The installation of go-git is quite different from the install of git2go.

go-git is written in pure Go so it is installed using the usual Go workflow: `go get`, `go build`, `go install`, etc (Figure 1).

```
$ mkdir -p $GOPATH/src
$ go get -d github.com/jmahler/mgmirr

$ git -C $GOPATH/src/github.com/jmahler/mgmirr checkout go-git
$ go build github.com/jmahler/mgmirr

$ go test github.com/jmahler/mgmirr
ok      github.com/jmahler/mgmirr       0.260s
$ go test -tags=integration github.com/jmahler/mgmirr
ok      github.com/jmahler/mgmirr       22.455s
```

Figure 1: go-git install steps.

git2go requires the libgit2 library and can be configured to use dynamically loaded libraries (shown here) or statically complied ones (not shown). Source must be downloaded first, then system packages installed, then git2go is built, and finally mgmirr itself can be built (Figure 2).

```
$ mkdir -p $GOPATH/src
$ go get -d github.com/jmahler/mgmirr
$ go get -d github.com/libgit2/git2go

$ # (Ubuntu 18)
$ sudo apt install cmake libgit2-26 libgit2-dev libssh2-1-dev

$ cd $GOPATH/src/github.com/libgit2/git2go
$ git checkout master
$ git submodule update --init
$ make install-dynamic
$ make test-dynamic

$ cd $GOPATH/src/github.com/jmahler/mgmirr
$ go build

$ go test github.com/jmahler/mgmirr
ok      github.com/jmahler/mgmirr       0.156s
$ go test -tags=integration github.com/jmahler/mgmirr
ok      github.com/jmahler/mgmirr       4.076s
```

Figure 2: git2go install steps.

go-git is simpler to install than git2go. It has no system dependencies beyond Go itself. And there are fewer steps to get everything built and installed.

3 Performance

The mgmirr project includes two sets of tests: unit tests which only use local resources, and integration tests which clone from remote sources. The integration tests are slower but a more realistic measure of performance.

Running the unit tests shows that git2go is only slightly faster than go-git (0.05) (Figure 3, 4). However, running the integration tests shows that git2go is over 5 times faster than go-git (4.076 to 22.455).

```
$ go test github.com/jmahler/mgmirr
ok      github.com/jmahler/mgmirr       0.260s
$ go test -tags=integration github.com/jmahler/mgmirr
ok      github.com/jmahler/mgmirr       22.455s
```

Figure 3: go-git test performance.

```
$ go test github.com/jmahler/mgmirr
ok      github.com/jmahler/mgmirr       0.156s
$ go test -tags=integration github.com/jmahler/mgmirr
ok      github.com/jmahler/mgmirr       4.076s
```

Figure 4: git2go test performance.

git2go beats go-git with integration tests which are 5x faster. However, the specific reason for this advantage is unknown. More research into this difference would be interesting.

4 Documentation

On the surface both `go-git` and `git2go` have adequate documentation[6].[7] Looking deeper, `git2go` appears to have less detailed documentation[8] than `git-go`.[9] However, because `git2go` is simply C bindings to the ubiquitous libgit2 library[10] there are actually far more resources available[11].[12] Granted, some translation is necessary to convert these C/Ruby/Python/Rust/etc examples to Go but it's usually straight forward (e.g. camel case).

When the documentation is insufficient both `git2go` and `go-git` have source code available with tests and examples. But again because `git2go` is simply C bindings to the ubiquitous libgit2 library, which has bindings for nearly every programming language,[13] there is a much larger pool of examples available.

The documentation for git2go beats go-git because it is simply a wrapper to the ubiquitous libgit2 library which has a plethora of examples available.

5 Programming Abstractions

5

Implementing the PullAll operation requires walking all the local and remote branches and pulling them from their respective remotes.

Refer to Figure 5 for the following.

[6] *go-git.v4 GoDoc*. [Online; accessed 2-March-2020]. URL: `https://godoc.org/gopkg.in/src-d/go-git.v4`.
[7] *git2go GoDoc*. [Online; accessed 2-March-2020]. URL: `https://godoc.org/github.com/libgit2/git2go`.
[8] *git2go#Clone*. [Online; accessed 2-March-2020]. URL: `https://godoc.org/github.com/libgit2/git2go#Clone`.
[9] *go-git.v4#Clone*. [Online; accessed 2-March-2020]. URL: `https://godoc.org/gopkg.in/src-d/go-git.v4#Clone`.
[10] *libgit2.org*, see n. 3.
[11] *libgit2: Cloning — Ben Straub*. [Online; accessed 2-March-2020]. URL: `https://ben.straub.cc/2013/02/01/stupid-libgit2-tricks-cloning/`.
[12] *libgit2 101 - Clone*. [Online; accessed 2-March-2020]. URL: `https://libgit2.org/docs/guides/101-samples/#repositories_clone_simple`.
[13] *libgit2.org*, see n. 3.

In `go-git` the programmer is given references (`git show-ref`). References included heads, remotes, tags and various other things beyond just branches. Walking the references requires filtering out everything which isn't a branch. The naming conventions used for references must also be accounted for. For example: the reference `refs/heads/fedora/f31` would map to the local `fedora/f31` branch. All of this obscures access to branches and creates confusion.

In `git2go` there are operations for branches. A branch iterator is created which contains both the local and remote branches. And a flag is provided to test whether it is local or remote.

```
mgmirr$ git diff b0d97dbe319..a4ec0f7c03
                                  (git2go)     (go-git)
[...]
+       iter, err := repo.NewBranchIterator(git.BranchRemote)
        if err != nil {
                return nil, err
        }
-       _ = refs.ForEach(func(c *plumbing.Reference) error {
-               ref_branch := c.Strings()[0]
-               if isBranch(ref_branch) {
-                       ref_branches = append(ref_branches, ref_branch)
+       defer iter.Free()
+       for {
+               ref, branch_type, err := iter.Next()
+               if err != nil {
+                       break
                }
-               return nil
-       })
-
-       // refs/heads/fedora/f31 -> refs/remotes/fedora/f31
-       var branches []string
-       for _, ref_branch := range ref_branches {
-               prefix := "refs/remotes/"
-               if strings.HasPrefix(ref_branch, prefix) {
-                       branch := strings.TrimPrefix(ref_branch, prefix)
-                       branches = append(branches, branch)
+               if branch_type != git.BranchRemote {
+                       continue
                }
-               // else ignore local (refs/heads) branches,
-               //      the're accounted for by the remotes.
+               branch, _ := ref.Branch().Name() // fedora/f31
+               branches = append(branches, branch)
        }
[...]
```

Figure 5: Diff between PullAll implementation in git2go and go-git.

6 Conclusion

The benefit of `go-git` is that it is written in pure Go and this makes it easy to install. However, installation is a one time cost. Maintainability and developer productivity is an ongoing cost.

The benefity of `git2go` is that it is simply a wrapper on the widely used and ubiquitous libgit2. libgit2 is fast and stable and widely used across many programming languages. `git2go` is the clear choice over `go-git`.

Appendices

A Unit Tests

There were several side effects of having unit tests. These aren't shortcomings of go-git or git2go but they are still interesting nonetheless.

Minimal changes had to made to get the unit tests from go-git working for git2go (Figure 6). The biggest change was the addition of the `testTrackingBranch` test which wasn't caught during the development of go-git. Other changes were semantic: Clone syntax, library names, URL instead of URLs.

Having unit tests gave confidence that the new git2go implemention was functionally equivalent to the go-git version. Without unit tests the only option would be ad hoc testing which gives very little confidence that it is equivalent.

B Logical Changes

Following guidlines from the development of the Linux Kernel, every patch was seperated in to one logical change[14] as shown in Figure 7. This helped ease the migration from go-git to git2go since each logical change could be migrated and tested incrementally. It took a small amount of effort to migrate one logical change

Because each logical change was the same between go-git and git2go it is easy to see how the implementation differed (Figure 8).

[14] *Linux: Seperate each logical change into a seperate patch.* [Online; accessed 11-March-2020]. URL: `https://www.kernel.org/doc/Documentation/process/submitting-patches.rst`.

```
mgmirr$ git diff go-git:gitutils_test.go git2go:gitutils_test.go
diff --git a/gitutils_test.go b/gitutils_test.go
index ee74ac1..d1b758d 100644
--- a/gitutils_test.go
+++ b/gitutils_test.go
@@ -3,7 +3,7 @@ package mgmirr_test
 import (
        "fmt"
        "github.com/jmahler/mgmirr"
-       "gopkg.in/src-d/go-git.v4"
+       "github.com/libgit2/git2go"
        "io/ioutil"
        "os"
        "os/exec"
@@ -32,20 +32,18 @@ func TestRpmMirror(t *testing.T) {
                t.Fatal(err)
        }

-       repo, err := git.PlainClone(dir, false, &git.CloneOptions{
-               URL: cfg.Origin.URLs[0],
-       })
+       repo, err := git.Clone(cfg.Origin.URL, dir, &git.CloneOptions{Bare: false})
        if err != nil {
[...]

-       // trying to clone a second time should encounter AlreadyExists
-       _, err = git.PlainClone(dir, false, &git.CloneOptions{
-               URL: cfg.Origin.URLs[0],
-       })
-       if err != nil {
-               if err != git.ErrRepositoryAlreadyExists {
-                       t.Fatalf("git (2nd) clone of '%s' to '%s' failed: %v", cfg.Origin.URLs[0], dir
+       // trying to clone a second time should fail because it already exists
+       _, err = git.Clone(cfg.Origin.URL, dir, &git.CloneOptions{Bare: false})
+       if err == nil {
+               t.Fatalf("git (2nd) clone of '%s' to '%s' should've failed", cfg.Origin.URL, dir)
+       } else {
+               if !strings.Contains(err.Error(), "exists and is not an empty directory") {
+                       t.Fatalf("git (2nd) clone of '%s' to '%s' failed: %v", cfg.Origin.URL, dir, er
                }
        }

@@ -70,7 +68,7 @@ func TestRpmMirror(t *testing.T) {
        })

        t.Run("FetchAll", func(t *testing.T) {
-               err = mgmirr.FetchAll(repo, cfg.Remotes)
+               err = mgmirr.FetchAll(repo)
                if err != nil {
                        t.Fatalf("FetchAll failed: %v", err)
                }
@@ -104,6 +102,8 @@ func TestRpmMirror(t *testing.T) {
                        {"other/my/branch/with/lots/of/parts", true},
                }
                testBranches(t, dir, cases)
+
+               testTrackingBranch(t, dir, "fedora/f31", "remotes/fedora/f31")
        })

        t.Run("PullAll", func(t *testing.T) {
@@ -180,6 +180,23 @@ type BranchCase struct {
        Exists bool
 }

+func testTrackingBranch(t *testing.T, dir string, branch string, tracking_branch string) {
[...]
```

Figure 6: Unit test differences between the go-git and git2go branches.

```
mgmirr$ git log
[...]
commit a4ec0f7c030ea671d9cf173873fd2075627757cf
Author: Jeremiah Mahler <jmmahler@gmail.com>
Date:   Sat Dec 21 00:16:55 2019 +0000

    add PullAll

commit 15421f747e4ebf619498f680c3fc1ce4c80d66af
Author: Jeremiah Mahler <jmmahler@gmail.com>
Date:   Wed Dec 11 02:19:31 2019 +0000

    add SetupRpmBranches

commit 0248f5919ed197e75ba178dfcb7f055fbd29f67d
Author: Jeremiah Mahler <jmmahler@gmail.com>
Date:   Wed Dec 11 01:47:22 2019 +0000

    add FetchAll remotes

commit 7426fbfc007c50bf9feddd3a25dab68ebf25c95e
Author: Jeremiah Mahler <jmmahler@gmail.com>
Date:   Tue Dec 10 17:22:25 2019 +0000

    add SetupRpmRemotes

    Add SetupRpmRemotes which takes an existing Git repo
    and sets up the remotes according to the given configs.
[...]
```

Figure 7: Git log showing logical changes made in go-git/git2go branches of mgmirr.

```
mgmirr$ git diff fe865e88fb8b374a4 7426fbfc007c50bf9
diff --git a/gitutils.go b/gitutils.go
index 08c7def..3da347a 100644
--- a/gitutils.go
+++ b/gitutils.go
@@ -2,17 +2,21 @@ package mgmirr

 import (
        "fmt"
-       "gopkg.in/src-d/go-git.v4"
-       "gopkg.in/src-d/go-git.v4/config"
+       "gopkg.in/libgit2/git2go.v27"
        "log"
 )

+type RemoteConfig struct {
+       Name string
+       URL  string
+}
+
 // For an existing Git repo and an RPM (e.g. cowsay) Setup the remotes.
 //
 // This is a best effort procedure.  Not all remotes will be available
 // (fedora might not have package x).  As long as at least one remote
 // works it is a success.
-func SetupRpmRemotes(repo *git.Repository, rcs []config.RemoteConfig) error {
+func SetupRpmRemotes(repo *git.Repository, rcs []RemoteConfig) error {

        var one_worked bool = false

@@ -34,14 +38,10 @@ func SetupRpmRemotes(repo *git.Repository, rcs []config.RemoteConfig) error {
        }
 }

-func setupRpmRemote(repo *git.Repository, cfg *config.RemoteConfig) error {
-       _, err := repo.CreateRemote(cfg)
+func setupRpmRemote(repo *git.Repository, cfg *RemoteConfig) error {
+       _, err := repo.Remotes.Create(cfg.Name, cfg.URL)
        if err != nil {
-               if err == git.ErrRemoteExists {
-                       // OK
-               } else {
-                       return fmt.Errorf("git add remote for '%v' failed: %v", cfg.Name, err)
-               }
+               return fmt.Errorf("git add remote for '%v' failed: %v", cfg.Name, err)
        }
[...]
```

Figure 8: Git diff of the "add SetupRpmRemotes" change in go-git and git2go

References

git2go GoDoc. [Online; accessed 2-March-2020]. URL: https://godoc.org/github.com/libgit2/git2go.

git2go#Clone. [Online; accessed 2-March-2020]. URL: https://godoc.org/github.com/libgit2/git2go#Clone.

github.com/jmahler/mgmirr/tree/git2go. [Online; accessed 2-March-2020]. URL: https://github.com/jmahler/mgmirr/tree/git2go.

github.com/jmahler/mgmirr/tree/go-git. [Online; accessed 2-March-2020]. URL: https://github.com/jmahler/mgmirr/tree/go-git.

github.com/libgit2/git2go. [Online; accessed 2-March-2020]. URL: https://github.com/libgit2/git2go.

github.com/src-d/go-git. [Online; accessed 2-March-2020]. URL: https://github.com/src-d/go-git.

go-git.v4 GoDoc. [Online; accessed 2-March-2020]. URL: https://godoc.org/gopkg.in/src-d/go-git.v4.

go-git.v4#Clone. [Online; accessed 2-March-2020]. URL: https://godoc.org/gopkg.in/src-d/go-git.v4#Clone.

libgit2 101 - Clone. [Online; accessed 2-March-2020]. URL: https://libgit2.org/docs/guides/101-samples/#repositories_clone_simple.

libgit2: Cloning — Ben Straub. [Online; accessed 2-March-2020]. URL: https://ben.straub.cc/2013/02/01/stupid-libgit2-tricks-cloning/.

libgit2.org. [Online; accessed 2-March-2020]. URL: https://libgit2.org/.

Linux: Seperate each logical change into a seperate patch. [Online; accessed 11-March-2020]. URL: https://www.kernel.org/doc/Documentation/process/submitting-patches.rst.