Arvados - Story #9005

[SDKs] Go SDK's arvadosclient and keepclient should share http.Client objects by default

04/19/2016 01:22 PM - Tom Clegg

<table>
<thead>
<tr>
<th>Status:</th>
<th>Resolved</th>
<th>Start date:</th>
<th>05/29/2017</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
<td>Due date:</td>
<td></td>
</tr>
<tr>
<td>Assigned To:</td>
<td>Tom Clegg</td>
<td>% Done:</td>
<td>100%</td>
</tr>
<tr>
<td>Category:</td>
<td>SDKs</td>
<td>Estimated time:</td>
<td>0.00 hour</td>
</tr>
<tr>
<td>Target version:</td>
<td>2017-06-07 sprint</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Description

Background

As an example, keep-web currently

- gets an arvadosclient.ArvadosClient from an arvadosclient.ClientPool
- configures it with the current token
- passes it to keepclient.MakeKeepClient() in order to get a keepclient.Client

(The public APIs for creating a keepclient.Client all involve passing in an arvadosclient.ArvadosClient, so we can't just make a pool of keepclient.Clients and assume they won't leak credentials across requests.)

At this point, keepclient.MakeKeepClient() creates a new http.Client object. This means

- Consecutive keep-web requests cannot reuse http connections to keep services
- (Currently) the HTTP connections to keep services are left open anyway, because we don't tell the http.Client that we won't be reusing it

In general, sharing an http.Client across requests is beneficial because it reduces TCP and TLS handshaking.

Proposed improvements

keepclient should use kc.Arvados.Client instead of its own Client field unless the caller specifies otherwise.

- keepclient.New() should leave the Client field set to nil by default.
- keepclient methods that use an http client should get it from a private httpClient() method that works along the lines of the one in source/sdk/go/arvados/client.go: use the KeepClient field if non-nil, otherwise the Arvados one.

(This change alone should reduce the proliferation of http.Client objects significantly.)

Also, arvadosclient.MakeArvadosClient should reuse the same two http.Client objects for all new ArvadosClient objects -- e.g.,

arvadosclient.DefaultHTTPClient when ApiInsecure is false, arvadosclient.DefaultInsecureHTTPClient when true. The caller can override it later if desired.

arvadosclient.DefaultHTTPClient should default to http.DefaultClient. (keepclient modifies client timeouts and keepalive settings according to proxy/non-proxy config, and we shouldn't muck with the http package globals this way.)

(This change should reduce the proliferation of http.Client objects further, especially in programs that don't use an arvados client pool.)

Subtasks:

- Task # 11773: use RefreshKeepServers in keep-web
- Task # 11761: Review 9005-keep-http-client
- Task # 11772: Share http.Client and http.Transport by default
- Task # 11795: Review 9005-share-discovery

Associated revisions

Revision b51781b9 - 04/19/2016 03:17 PM - Tom Clegg

Merge branch '9004-close-keep-connections'

refs #9004
Peter Amstutz wrote:

API client and keepstore have different socket settings, which probably why they were not combined in the first place.

The pointer to the Arvados client object a public field of the KeepClient struct. If it's okay to pool ArvadosClient objects and update the ApiToken field, then it should be okay to pool the KeepClient objects and update the Arvados field.

I assume you're referring to the code that sets timeout and keepalive settings in `source/sdk/go/keepclient/support.go` after seeing the list of keep services.

It would seem rude to alter http.DefaultClient this way, but I think it would be reasonable if the default behavior used just one http.Client object per ArvadosClient -- i.e., keepclient should default to using arvadosclient's http.Client instead of creating its own, and it should not do anything that sabotages arvadosclient's own use of that http.Client.

The pointer to the Arvados client object a public field of the KeepClient struct. If it's okay to pool ArvadosClient objects and update the ApiToken field, then it should be okay to pool the KeepClient objects and update the Arvados field.
True (at least with the current code). I think it would be even better if keepclients were efficient enough to make pools irrelevant. Perhaps if the list of keep services were cached in arvadosclient instead of keepclient (and we make the above changes to http.Client usage) then there would be no remaining motivation for pooling?

(I suppose ideally clients like keep-web shouldn't even need to know about a "keepclient" thing -- they should just say stuff like arv.GetCollection(uuid).Open(filename))

#7 - 05/19/2017 02:15 PM - Tom Clegg
- Category set to Keep
- Status changed from New to In Progress
- Assigned To set to Tom Clegg

#8 - 05/19/2017 04:01 PM - Peter Amstutz
- Status changed from In Progress to New
- Assigned To deleted (Tom Clegg)
- Category deleted (Keep)

Tom Clegg wrote:

Peter Amstutz wrote:

API client and keepstore have different socket settings, which probably why they were not combined in the first place.

I assume you're referring to the code that sets timeout and keepalive settings in source/sdk/go/keepclient/support.go after seeing the list of keep services.

It would seem rude to alter http.DefaultClient this way, but I think it would be reasonable if the default behavior used just one http.Client object per arvadosclient -- i.e., keepclient should default to using arvadosclient's http.Client instead of creating its own, and it should not do anything that sabotages arvadosclient's own use of that http.Client.

I don't see how we can share the same HTTP client for both API requests and keep blocks when we have settings like this:

```go
client.Timeout = 20 * time.Second
```

We allow API calls to take a lot longer than that to return results.

The pointer to the Arvados client object a public field of the KeepClient struct. If it's okay to pool ArvadosClient objects and update the ApiToken field, then it should be okay to pool the KeepClient objects and update the Arvados field.

True (at least with the current code). I think it would be even better if keepclients were efficient enough to make pools irrelevant. Perhaps if the list of keep services were cached in arvadosclient instead of keepclient (and we make the above changes to http.Client usage) then there would be no remaining motivation for pooling?

Keeping the list of keep services on ArvadosClient would be an improvement, but it still needs to be updatable (I'm thinking the keep service roots should get their own struct with a lock that ArvadosClient can point to). Although we can do that and have the keep services list stay on the keep client object, and then the keep client object is safely copyable again.

#9 - 05/19/2017 06:10 PM - Tom Clegg

Peter Amstutz wrote:

I don't see how we can share the same HTTP client for both API requests and keep blocks when we have settings like this:

```go
client.Timeout = 20 * time.Second
```

We allow API calls to take a lot longer than that to return results.

The pointer to the Arvados client object a public field of the KeepClient struct. If it's okay to pool ArvadosClient objects and update the ApiToken field, then it should be okay to pool the KeepClient objects and update the Arvados field.

True (at least with the current code). I think it would be even better if keepclients were efficient enough to make pools irrelevant. Perhaps if the list of keep services were cached in arvadosclient instead of keepclient (and we make the above changes to http.Client usage) then there would be no remaining motivation for pooling?

Keeping the list of keep services on ArvadosClient would be an improvement, but it still needs to be updatable (I'm thinking the keep service roots should get their own struct with a lock that ArvadosClient can point to). Although we can do that and have the keep services list stay on the keep client object, and then the keep client object is safely copyable again.

Good point. ArvadosClient would need two http.Clients, one for Keep and one for API.

Keeping the list of keep services on ArvadosClient would be an improvement, but it still needs to be updatable (I'm thinking the keep service roots should get their own struct with a lock that ArvadosClient can point to). Although we can do that and have the keep services list stay on the keep client object, and then the keep client object is safely copyable again.

In order for this to be a reasonable experience for callers I think we just have to use the appropriate number of http.Client objects, discovery doc requests, and keep_services/accessible requests, no matter how many times the caller does this:

```go
arv := arvados.NewClientFromEnv()
arv.Token = token
```
It should be trivial for callers to get efficient resource usage without juggling pools of ArvadosClient and KeepClient objects.

#10 - 05/19/2017 07:56 PM - Tom Clegg
Some related fixes:
9005-conn-leak @ cad24bb2240b47f59bc5719a035e85ff5e60ef

#11 - 05/23/2017 07:09 PM - Tom Morris
- Target version changed from Arvados Future Sprints to 2017-06-07 sprint

#12 - 05/24/2017 06:50 PM - Tom Morris
- Assigned To set to Tom Clegg

#13 - 05/29/2017 01:35 PM - Tom Clegg
- Category set to SDKs
- Status changed from New to In Progress

#14 - 05/30/2017 01:23 PM - Tom Clegg
9005-keep-http-client @ 6fe6390690471cee8ba23984e3560fc4ced8b180
I'd also like to make keep-web remember the list of keep services from one request to the next. In the meantime the above branch should bring back HTTP keepalive without leaking unlimited FDs.

#15 - 05/31/2017 06:05 PM - Peter Amstutz
Seems like this should be in sync with either the "disk" or "non-disk" keep client settings. How about declaring some constants in keepclient.go for all the various timeouts just so its a bit more visible, then use those constants here?

```go
h := &proxyHandler{
    Handler: rest,
    KeepClient: kc,
    timeout: timeout,
    transport: &http.Transport{
        Dial: (&net.Dialer{
            Timeout: 20 * time.Second,
            KeepAlive: 10 * time.Second,
        }).Dial,
        TLSClientConfig: arvadosclient.MakeTLSConfig(kc.Arvados.ApiInsecure),
        TLSHandshakeTimeout: 10 * time.Second,
    },
    ApiTokenCache: &ApiTokenCache{
        tokens: make(map[string]int64),
        expireTime: 300,
    },
}.
```

#16 - 05/31/2017 06:09 PM - Peter Amstutz
Also, instead of spinning up a goroutine to periodically call CloseIdleConnections() there are settings on http.Transport we could use instead:

https://golang.org/pkg/net/http/

For control over proxies, TLS configuration, keep-alives, compression, and other settings, create a Transport:

```go
tr := &http.Transport{
    MaxIdleConns: 10,
    IdleConnTimeout: 30 * time.Second,
    DisableCompression: true,
}
client := &http.Client{Transport: tr}
resp, err := client.Get("https://example.com")
```

#17 - 05/31/2017 07:07 PM - Tom Clegg
9005-keep-http-client @ 75c5b123e0b4cd8bc60de9b15364a97c220994740
SDK & keepproxy now copy http.DefaultTransport and modify the timeouts instead of building from scratch, so we can benefit from the default Go
settings (non-zero MaxIdleConns, etc).

#18 - 06/01/2017 01:53 AM - Tom Clegg
9005-share-discovery @ 916cf89b440ld13a9b9c055d817b34d339442ea3

TBD

- keepclient.ClearCache() - better name? maybe RefreshServiceDiscovery()?  

#19 - 06/01/2017 06:42 PM - Peter Amstutz

So the strategy is to keep a global map of API server address to keep service list, spin up a goroutine which keeps that service list up to date, and request the latest service list on a channel whenever we need it.

- It looks like it uses the first API token passed to it as the API token for subsequent keep services requests? That should probably be reflected in the comment to MakeKeepClient().
- Agree that "ClearCache" is not a good name. RefreshServiceDiscovery() is fine, or ResetServiceCache()
- Using RW Mutex on KeepClient instead of a regular mutex is a bit less efficient than a regular Mutex when there isn't much contention and/or the critical section is really small: https://groups.google.com/forum/#!topic/prometheus-developers/qi-A4IV3Z-Y
- Nitpick, inconsistent implementation of parsing booleans:

NewClientFromEnv:

```go
strings.ToLower(os.Getenv("ARVADOS_API_HOST_INSECURE")); s == "1" || s == "yes" || s == "true"
```

MakeArvadosClient:

```go
var matchTrue = regexp.MustCompile("^\(?i:1|yes|true\)$")
```

Workbench CollectionUploadTest is failing.

#20 - 06/01/2017 09:44 PM - Tom Clegg
9005-share-discovery @ 05295ae31362eefc182f3a2329b3903d6f82a3b

- Made a "StringBool" function and used that everywhere I found a copy&pasted matchTrue regexp.
- Renamed ClearCache to RefreshServiceDiscovery
- Punted RW Mutex change

passed https://ci.curoverse.com/job/developer-run-tests/313/

Seems like it'll be a problem to remember the first token we see for a given API host and reuse it forever.

Is there any reason we shouldn't make the /arvados/v1/keep_services/accessible API exempt from the token-checking middleware, like we do for the discovery doc?

#21 - 06/02/2017 05:44 PM - Peter Amstutz

Tom Clegg wrote:

> Seems like it'll be a problem to remember the first token we see for a given API host and reuse it forever.

Is there any reason we shouldn't make the /arvados/v1/keep_services/accessible API exempt from the token-checking middleware, like we do for the discovery doc?

The SDK will need to be backwards compatible with current API servers that require a token. Otherwise I don't have a problem with making the "accessible" route exempt from token checking.

#22 - 06/02/2017 05:46 PM - Peter Amstutz

Another option is to use the anonymous user token, which could be published in the discovery doc if it isn't already?

#23 - 06/02/2017 07:50 PM - Tom Clegg
9005-share-discovery @ 5d03f6499055ef109ca2c8d5c59941b25de1fa47

- ignores auth status for keep_services/accessible, so it doesn't matter which token the client uses.
- ideally we'd clear the api token in keepclient/discover.go and omit the Auth header if token is empty... but for now it seems more useful to at least
have a chance of working with older API servers.

Probably the sanest way of dealing with the upgrade is to add a note to Upgrade to master, "if you upgrade keep-web, make sure to upgrade API server too" ...?

#24 - 06/05/2017 02:04 PM - Peter Amstutz
LGTM @ 5d03f649055ef109ca2ce8d5d59941b25de1fa47

#25 - 06/05/2017 02:15 PM - Tom Clegg
- Status changed from In Progress to Resolved
- % Done changed from 75 to 100

Applied in changeset arvadosjcommit:cb230b07e0125d819991bc74a1f528740068157d.