Arvados - Feature #9066

[Keep] keepstore -max-requests limits concurrent requests

04/26/2016 08:20 PM - Tom Clegg

Status:	Resolved	**Start date:**	04/26/2016
Priority:	Normal	**Due date:**	
Assigned To:	Tom Clegg	**% Done:**	100%
Category:	Keep	**Estimated time:**	0.00 hour
Target version:	2016-05-11 sprint		

Description

Background

Currently, keepstore will accept an unbounded number of client requests, each one with a goroutine blocking until a memory buffer is available. If too many clients connect, keepstore will run out of file descriptors. This means it's unable to accept new connections, and even worse, it sabotages the requests at the front of the queue: keepstore can't open files on disk, or make connections to remote storage services.

When there are lots of queued requests, the ones at the back of the queue will eventually time out anyway; it would be better if those clients could find out right away that the server is overloaded.

Proposed fix

Add a -max-requests argument. When this many requests are already being handled/queued, respond immediately to new requests with "503 Server Busy".

If the -max-requests argument is not given, or is set to 0, use 2x max-buffers.

Note this applies to all requests, even ones that don't wait for buffers (e.g., index and status).

Implementation

Add a ConnectionLimiterFunc function to source:sdk/go/httpserver

```
struct limiterHandler {
         clients chan struct{}
         handler http.Handler
}

func NewConnectionLimiter(maxClients int, handler http.Handler) {
        return &limiterHandler{
                clients: make(chan struct{}, maxClients),
                handler: handler,
        }
}

func (h *limiterHandler) ServeHTTP(resp http.ResponseWriter, req *http.Request) {
        select {
        case h.clients <- struct{}{}:
        default:
               // reached max clients
               resp.WriteHeader(http.StatusServiceUnavailable)
               return
        }
        h.handler(resp, req)
        <- clients
}
```

In source:services/keepstore/logging_router.go, wrap MakeRESTRouter() with this handler.

Additional features, time permitting:

- Add a Len() method ("return len(h.clients)") to limiterHandler, and report that in keepstore's status.json

Subtasks:

Task # 9096: Review 9066-max-requests	**Resolved**

Related issues:

Related to Arvados - Bug #8825: [Keepstore] many connections in CLOSE_WAIT	**Resolved**	**03/07/2017**

Associated revisions

Revision 1535c8d1 - 04/29/2016 05:00 PM - Tom Clegg

Merge branch '9066-max-requests'

refs #9066

History

#1 - 04/27/2016 07:08 PM - Tom Clegg

- Target version changed from Arvados Future Sprints to 2016-05-11 sprint

#2 - 04/27/2016 07:08 PM - Tom Clegg

- Assigned To set to Tom Clegg

#3 - 04/28/2016 01:19 PM - Tom Clegg

- Description updated

#4 - 04/28/2016 01:20 PM - Tom Clegg

- Subject changed from [Keep] keepstore -max-clients limits concurrent connections to [Keep] keepstore -max-requests limits concurrent requests

#5 - 04/29/2016 01:54 PM - Radhika Chippada

- Is the following a reasonable test scenario?
  - Create a request limiter of size 1
  - Send first request
  - While first request is in progress, send a second one (which gets 503)
  - Complete request 1
  - Send third request, which should get 200

#6 - 04/29/2016 02:20 PM - Tom Clegg

Yes, good point. Added to TestRequestLimiter1 in source:sdk/go/httpserver/request_limiter_test.go:

```
        if n200 != 1 || n503 != 9 {
                t.Fatalf("Got %d 200 responses, %d 503 responses (expected 1, 9)", n200, n503)
        }
+       // Now that all 10 are finished, an 11th request should
+       // succeed.
+       go func() {
+               <-h.inHandler
+               h.okToProceed <- struct{}{}
+       }()
+       resp := httptest.NewRecorder()
+       l.ServeHTTP(resp, &http.Request{})
+       if resp.Code != 200 {
+               t.Errorf("Got status %d on 11th request, want 200", resp.Code)
+       }
 }
```

→commit:574a11b

#7 - 04/29/2016 07:33 PM - Tom Clegg

- Status changed from New to In Progress

#8 - 05/11/2016 05:11 PM - Tom Clegg

- Status changed from In Progress to Resolved