Load Testing RCDN

David Collier-Brown

The Old Way

- Invent a synthetic test
- Buy H-P LoadRunner licenses
- Get misleading answers
  - The H-P product is evil
  - And *inventing* the truth isn't a good plan
- Switch to Jmeter
- Which is free and good
  - But keep trying to invent a valid test load ...

The Samba team did it better

- They carefully created a load script from a debug=10 log
- Laboriously!
  - And just once…
- It's easier now
  - Everything uses REST
    - And Apache, and Nginx, and so on
  - They log one line per request-response pair
  - So we capture those and replay at 1x, 2x, 10x

Nginx example

- Logs are the same format as Apache (standardized)

```
10.76.2.1 - - [28/Nov/2017:06:55:04 -0500] "GET /81eae740-a93a-467c-
90d5-c555db9dc8a7 HTTP/1.1" 200 3994 "-" "Dalvik/1.6.0 (Linux; U;
Android 4.4.4; Nexus 7 Build/KTU84P)"
```

- Reformat to a particular format:

```
#date       time latency xferTime think bytes url rc op

2017-Nov-28 06:54:52 0 0 0 12578 /81eae740-a93a-467c   200 GET
```

- That's identical to the output format
  - An output can be used as an input for reruns
  - Or compared for improvement/degradation

First, run a smoke test

- loadGenerator -v --rest --tps 1 --for 1 log.csv
- This is super verbose
- If the return is not a 2XX, it prints the body (eg, an nginx “no response”)

#yyy-mm-dd hh:mm:ss latency xfertime thinktime bytes url rc op

2017-12-04 19:45:54.987 3.050636 0.000000 0 0 /download/images/00003003-64a4-4e25-ac2e-6c412bbf494d 444 GET expected=200

2017/12/04 19:45:58 restOps.go:182: error getting http response, Get http://calvin//download/images/00003003-64a4-4e25-ac2e-6c412bbf494d: dial tcp 192.168.0.3:80: getsockopt: no route to host

A success

```
#yyy-mm-dd hh:mm:ss latency xfertime thinktime bytes url rc
2017/11/11 21:11:20 runLoadTest.go:194: starting, at 1 requests/second
2017/11/11 21:11:20 runLoadTest.go:137: Loaded 1 records, closing input
2017/11/11 21:11:22 restOps.go:189:
Request:
GET /zaphod-beebelbrox.jpg HTTP/1.1
Host: calvin
User-Agent: Go-http-client/1.1
Cache-Control: no-cache
Accept-Encoding: gzip

Response headers:
    Length: 122944
    Status code: 200  OK
Response contents:
HTTP/1.1 200 OK
Content-Length: 122944
Accept-Ranges: bytes
Connection: keep-alive
Content-Type: image/jpeg
Date: Sun, 12 Nov 2017 02:11:47 GMT
Etag: "598db85d-30f2"
Last-Modified: Fri, 11 Aug 2017 13:59:57 GMT
Server: nginx/1.10.3 (Ubuntu)

Body:
 ���'���OJ�����cDe��*�7;
```

followed by many lines of gibberish from viewing a gif as text.

Now run from end to end

- Instead of `-for 1`, run through the whole file at some convenient speed
- If the system is expected to handle 100 request/second (TPS), try running at --tps 100 --crash, and see if you can get a clean run from beginning to end.
- Any error will put the verbose switch on, and --crash will stop on the first error

They you can try a load test

- Once you have a test that will run from end to end at a moderate load,try a test with a load varying from small to perhaps ten time the maximum
- 10x so you find the point at which the response time curve turns upwards in the classic hockey-stick, "_/".

```
ulimit -n 100000

runLoadTest --rest --tps 100 --progress 10 \
    --duration 10  --strip "images/" \
    ./resize_test_oldfiles.csv \

    http://10.92.10.202:80/images/v1/images.s3.kobo.com \

    >raw.csv
```

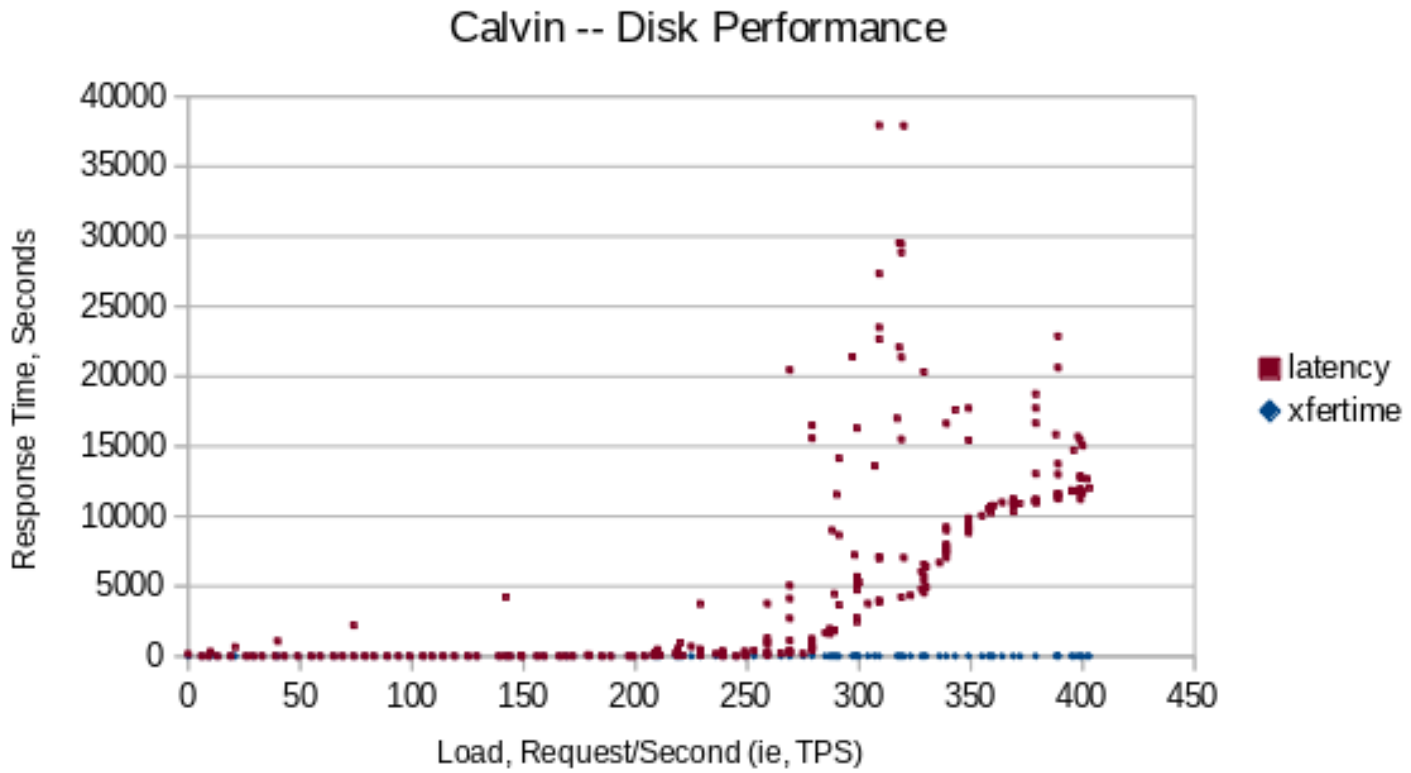
Convert to 1-second samples

- `$ perf2seconds raw.csv`

```
#date       time     latency  xfertime think bytes transactions
2017-09-04  15:44:15 0.004726 0.018895 0  1176955      6
2017-09-04  15:44:16 0.007143 0.01705  0  1976766      9
2017-09-04  15:44:17 0.005686 0.004471 0  471158       9
2017-09-04  15:44:18 0.00705  0.006686 0  672129       9
2017-09-04  15:44:19 0.009134 0.012883 0  1099113      9
...
2017-09-04  15:44:25 0.010516 0.005597 0  783936       13
2017-09-04  15:44:26 0.009564 0.012994 0  1398995      19
```

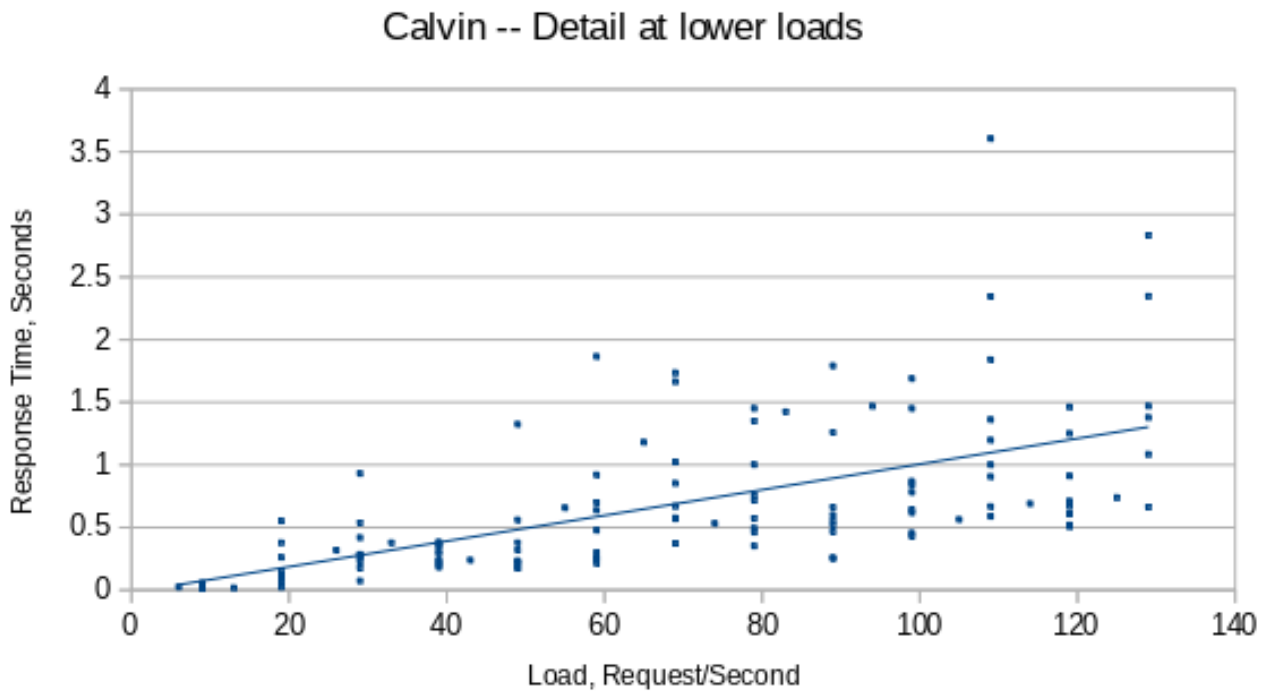
And plot as a scattergram

- It should look like "_/"



Detail

- At low load, the slope should be gentle



Draw Conclusions

- Down in the low range we expect, it's linear and pretty quick
  - At "normal overload" it's ordinarily slow
  - At > 250 TPS, it finally hits the wall
- If I want to build a ceph array of those disks, that's the information I need.
  - Ditto if it were any other server.
- Latency gently increases, from a quite pleasant 0.4s at 40 requests/second to an ugly 1.25 seconds at 120.

Links

Github repo

- https://github.com/davecb/Play-it-Again-Sam/

Tutorial

- https://github.com/davecb/Play-it-Again-Sam/blob/master/Running_Record-Reply_Tests.md

Man Page

- https://github.com/davecb/Play-it-Again-Sam/blob/master/cmd/runLoadTest/runLoadTest.md

Appendix: how to set up

- Start on a suitable machine
  - I used one of the haproxy LXC virtual machines as it had “headroom” for running other stuff
- Pick a target
  - That was an haproxy instance on another machine in the same cluster
  - I could easily use grafana to watch out for too much CPU usage on the two
- Install the load generator

Start on a suitable machine

- I used one of the haproxy LXC virtual machines as it had “headroom” for running other stuff
- You need to have an account there, mind you

```
$ ssh -2 -A -Y dcollierbrown@10.92.10.202
The authenticity of host '10.92.10.202 (10.92.10.202)' can't be established.
ECDSA key fingerprint is SHA256:GkBIOWi/+v3Ym8WEdBvGGuQ2UAtm8sqFL6AqpQqWioY.
Are you sure you want to continue connecting (yes/no)? Yes
dcollierbrown@vt-haproxy-002:~$
```

- Or take a copy of /home/dcollierbrown on 10.92.10.201 (vt-haproxy-001)

Install the load generator

- Create working directories under your ${HOME}
  - `mkdir -p go/src/github.com/davecb/Play-it-Again-Sam`
  - `cd go/src/github.com/davecb`
- Git clone either the saved Rakuten or the newest github copy of the generator
  - `git clone git@github.rakops.com:david-collierbrown/Play-it-Again-Sam.git`
  - Or new, `git clone git@github.com:davecb/Play-it-Again-Sam.git`
- Install the generator and go
  - `cd Play-it-Again-Sam/cmd/runLoadTest/localhost`
  - `make setup`
- Create a working directory by renaming ./localhost to, for example, vt-haproxy-002