Science & Technology Facilities Council

Introduction of load balancers at the RAL Tier-1

Andrew Lahiff, Alex Dibbo, Ian Collier

Rutherford Appleton Laboratory

HEPiX Fall 2016 Workshop, LBNL

Overview

- Motivation
- Load balancers
- Examples
  - FTS3
  - OpenStack
- Monitoring & alerting
- Experience
- Summary & future plans

Motivation

- What we used to do & still do for almost all services:
  - services directly exposed to the internet using via DNS (multiple A records)

- What's wrong with this?

Motivation

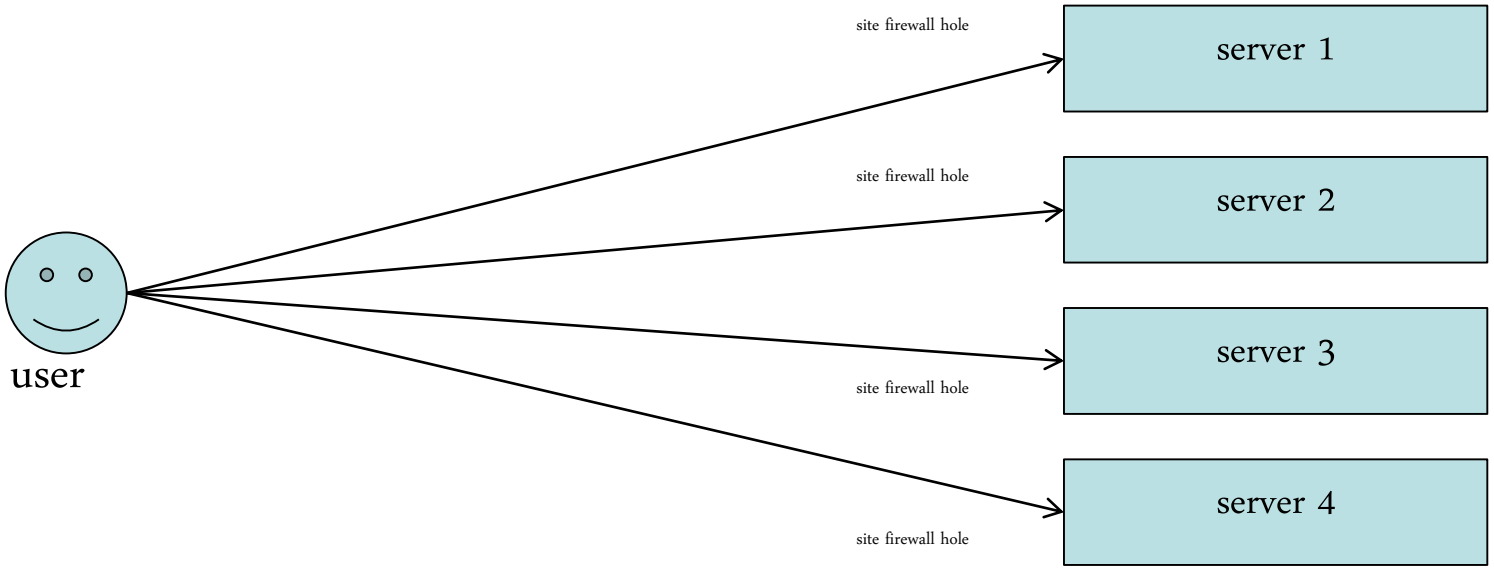
- What if one ATLAS SRM dies overnight?
  - the dead machine is still visible to users
    - Some fraction of requests from users will fail (~25%)
  - to resolve this, need to either
    1. we fix the machine, or
    2. contact RAL networking to change the alias
- What about upgrades & reboots?
  - since machines in intervention are still in DNS, maintainence is visible to users
    - Some fraction of requests will fail (~25% in this case)
- Ideally both machines failing & upgrades/reboots should be invisible to users
  - without requiring human intervention

Motivation

- We don't control our DNS: need to contact RAL networking team
  - But what if we could control our own DNS?
  - Or what if we had a more dynamic DNS?
- This is perhaps better, but still has problems
  - It's not unheard-of for applications to
    - not respect DNS TTLs
    - cache the results of name lookups
  - Issues with IPv6 DNS round-robin
    - round-robin doesn't always work – many clients will always pick the "first" host
    - if the "first" host is down, you have a problem

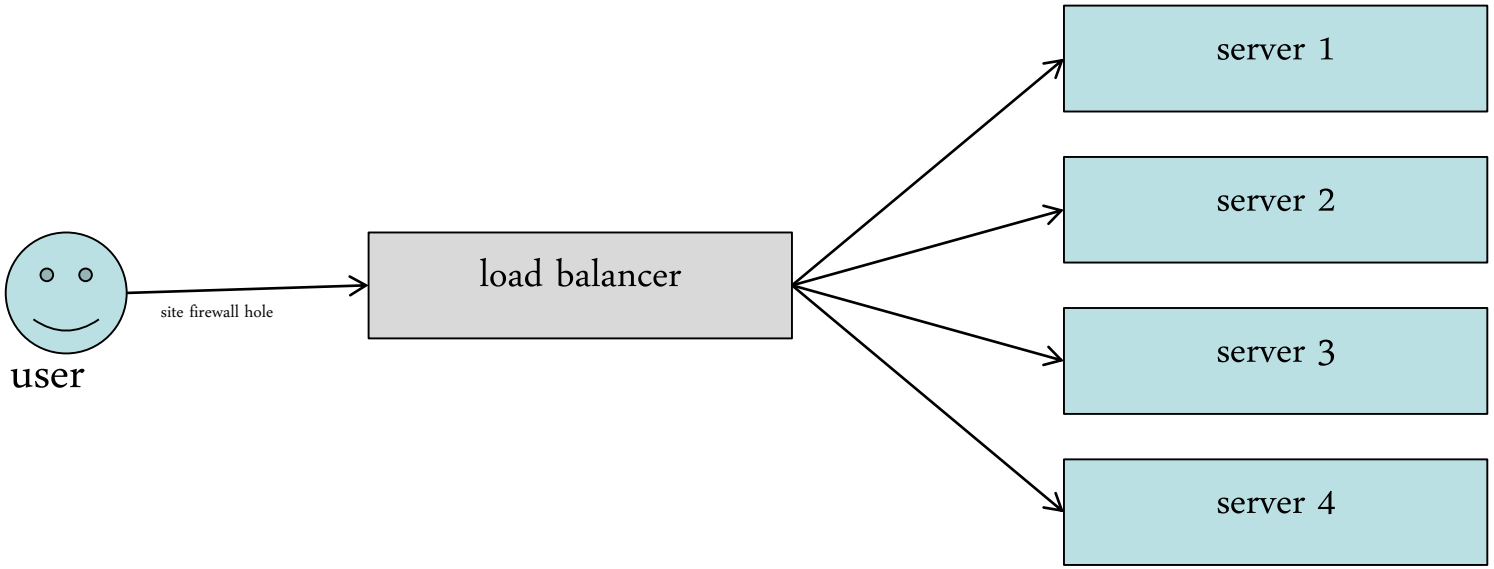
The alternative

- Instead of users connecting directly to servers...



The alternative

- Put a load balancer in between users and the servers



Building blocks

- HAProxy
  - Open source load balancer for TCP & HTTP
- Keepalived
  - Linux Virtual Server (LVS) router
  - Can provide floating IP addresses using Virtual Router Redundancy Protocol (VRRP)

Example: FTS3

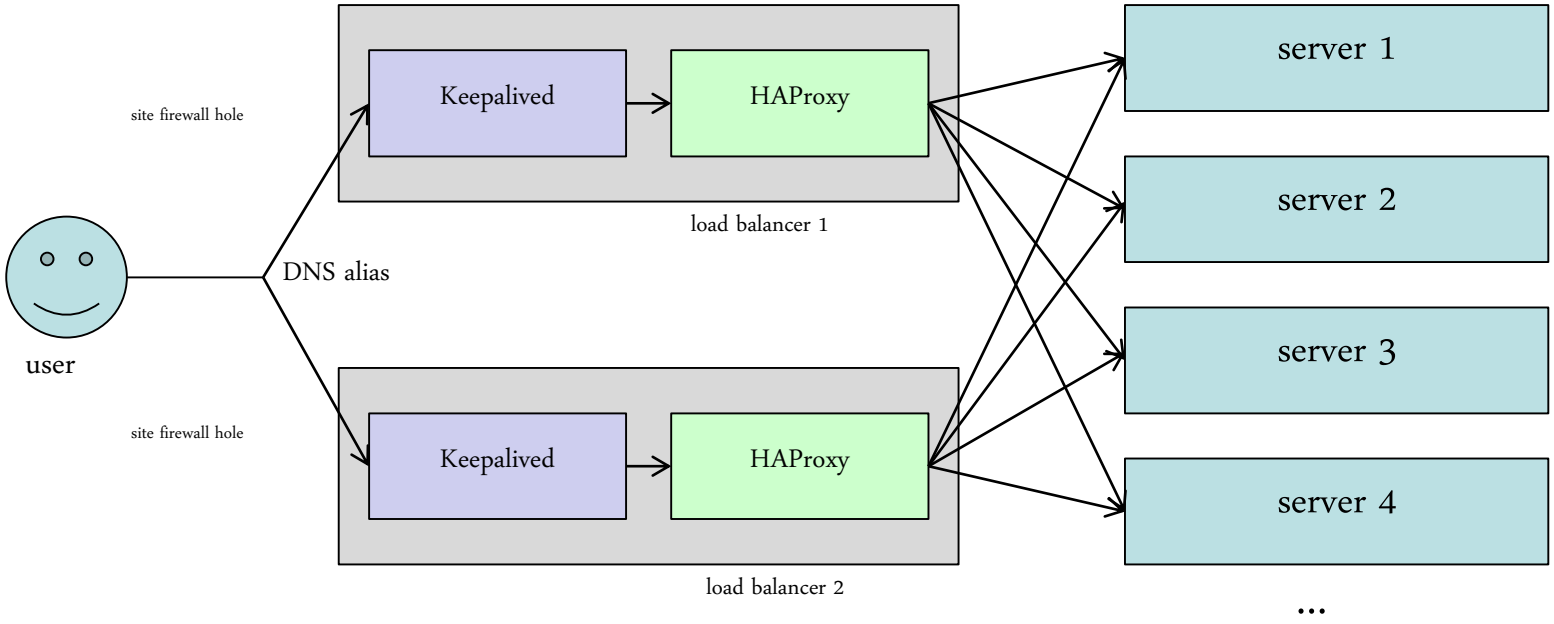
- First production service at RAL using load balancers
- 3 proxies configured in HAProxy
  - SOAP API (port 8443)
  - RESTful API (port 8446)
  - monitoring app (port 8449)
- HAProxy load balancing
  - round-robin
    - each backend server used in turn
    - used for SOAP, RESTful APIs
  - source
    - each client IP goes to the same backend server
    - used for the monitoring app

FTS3

- How to check if the backend servers are healthy?
  - HAProxy has configurable built-in checks
  - Currently using
    - tcp (SOAP API)
    - SSLv3 (RESTful API, monitoring app)
  - In future plan to move to more complete checks
    - e.g. HAProxy will stop sending requests to a server if host certificate expired, or CPU load very high, ...

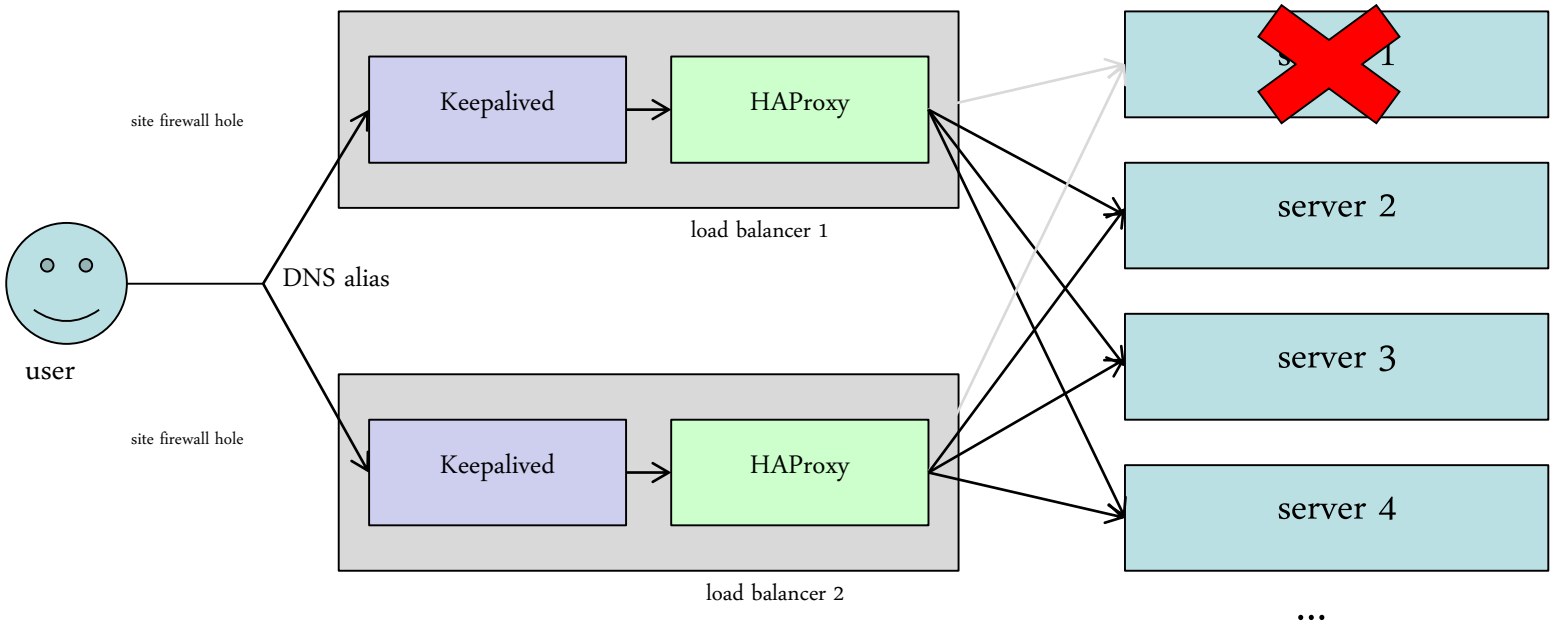
Architecture at RAL

- 2 floating IP addresses associated with the DNS entry for FTS3
  - traffic normally flows through both HAProxy instances



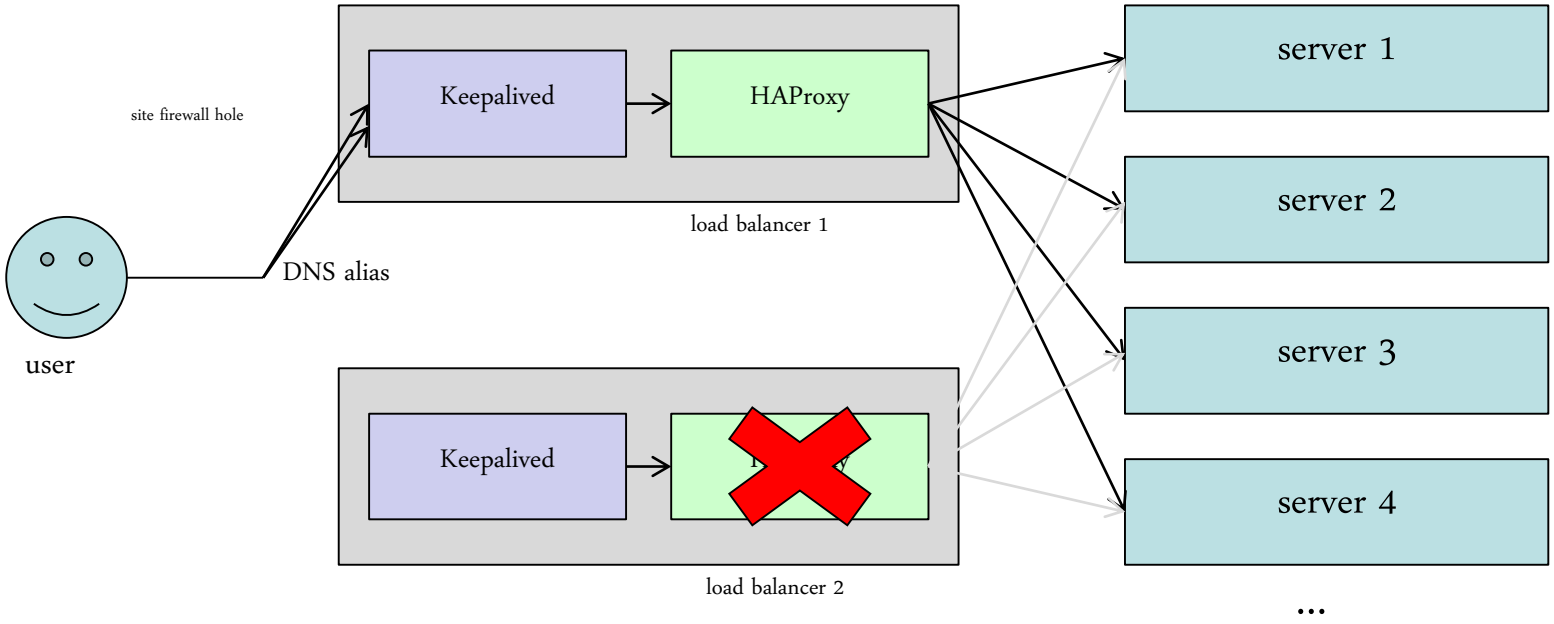
If a backend server dies...

- HAProxy stops sending requests to the broken server



If a HAProxy instance dies

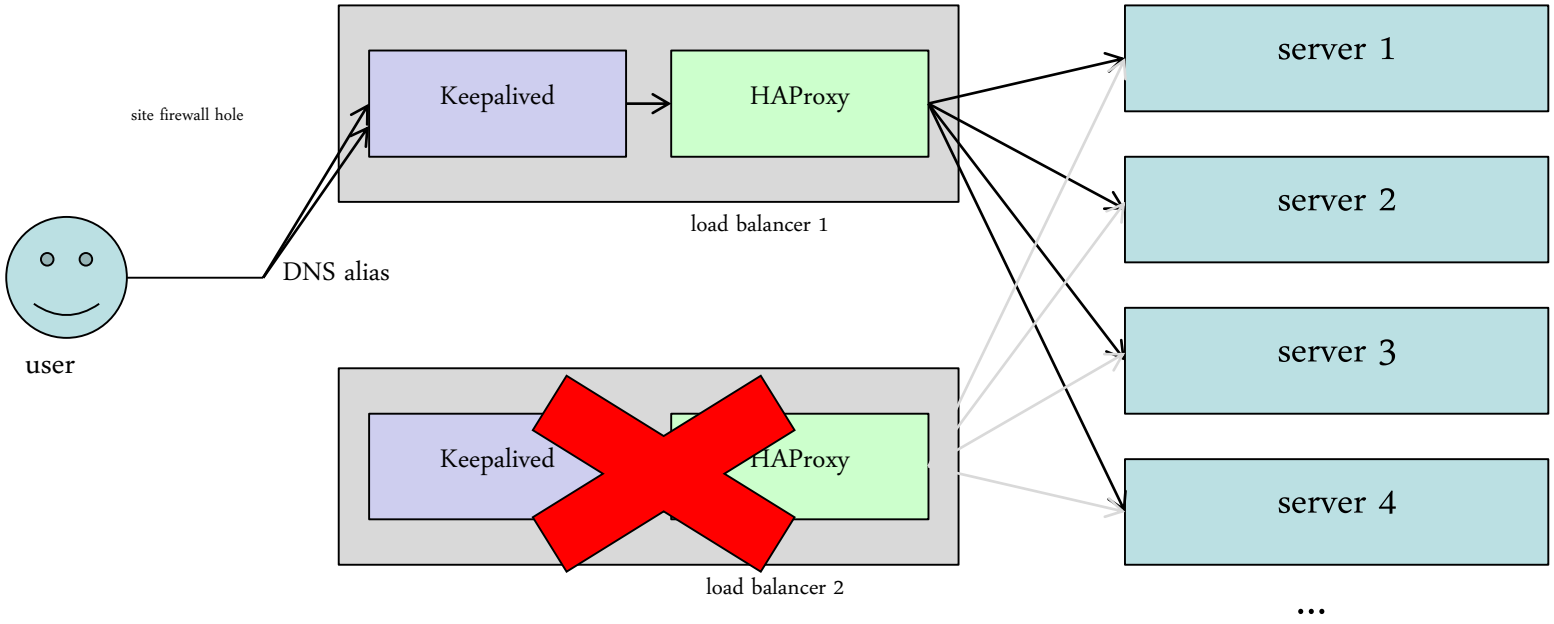
- The floating IP moves to the other load balancer



Keepalived checks if HAProxy is running

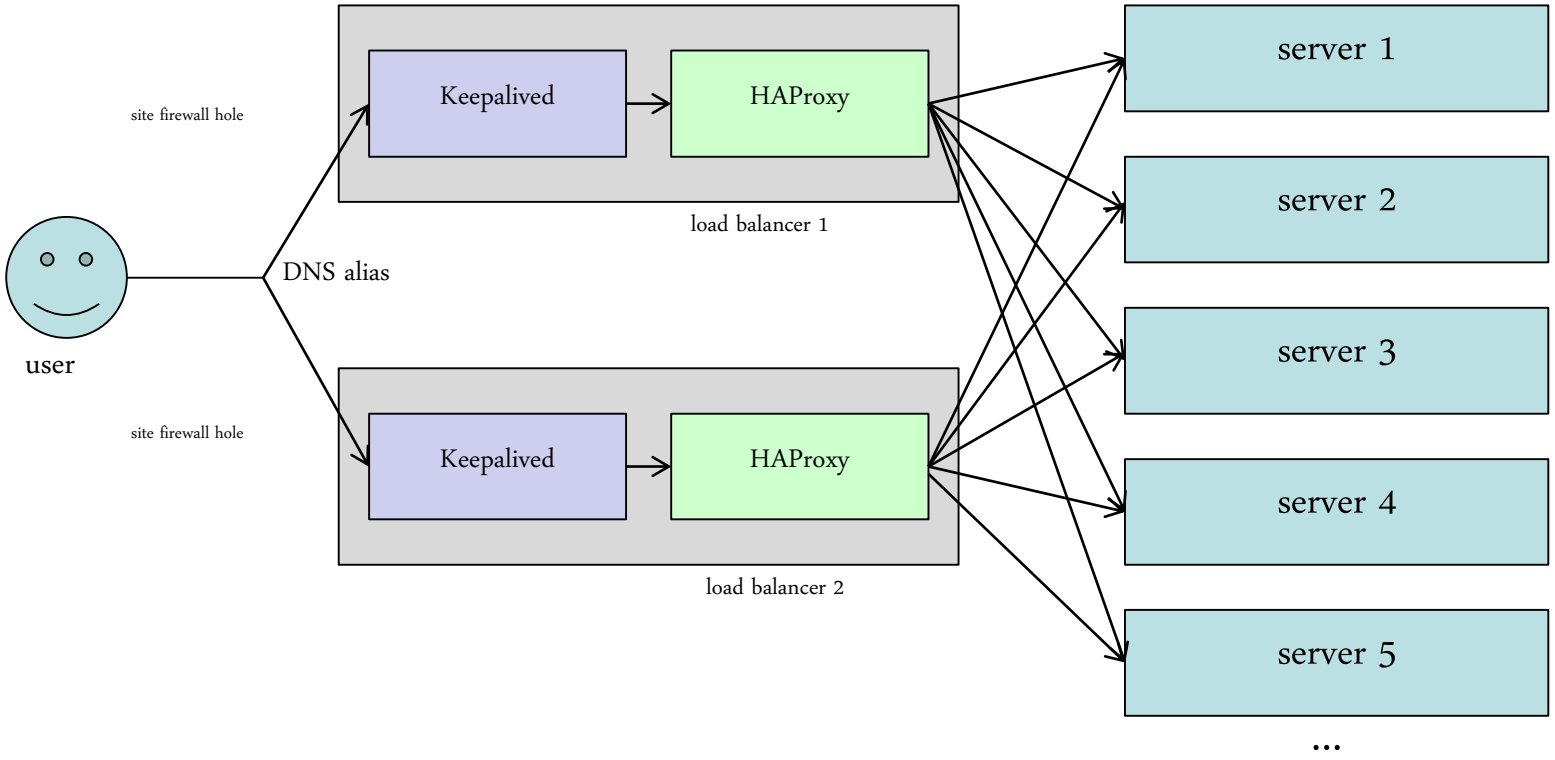
If a load balancer host dies

- The floating IP moves to the other load balancer



Add a new backend server

- Only need to update the HAProxy configuration
  - no need to ask RAL networking to update DNS

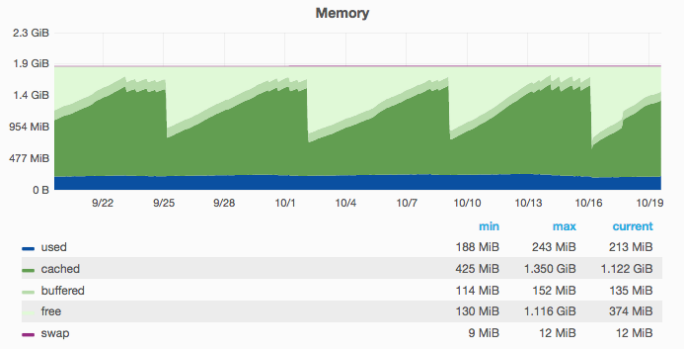


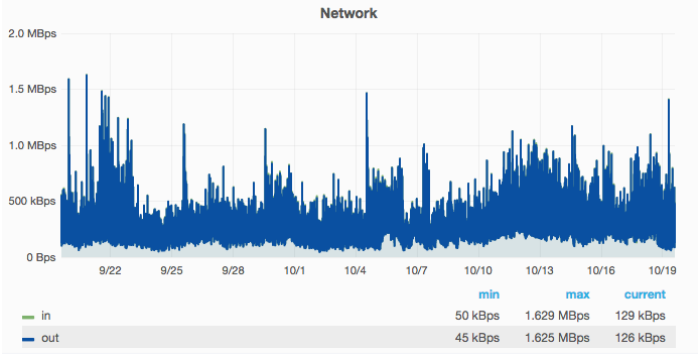
Maintenance

- Load balancers
  - each host can be upgraded/rebooted transparently (one at a time!)
- FTS3 servers
  - for planned interventions we put the appropriate backend server(s) in HAProxy into the “drain” state
    - stops any new connections going to the server
    - existing connections allowed to continue
  - after the intervention we put the server(s) back into the “ready” state

Infrastructure at RAL

- 2 VMs being used in production
  - each with 2 GB memory, 2 cores, 1 Gb networking
- For FTS3 only, resource usage very low
  - e.g. past 30 days:





Monitoring

- HAProxy stats page has lots of detailed information

fts3-prod-soap

	Queue			Session rate			Sessions						Bytes		Denied		Errors		Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				6	134	-	1	41	4 000	23 159 348			224 908 065 549	187 251 100 969	0	0	0					OPEN								
lcgfts01.gridpp.rl.ac.uk	0	0	128	1	17		0	6	-	2 894 992	2 894 992	1s	28 125 059 702	23 451 948 567		0		0	5	0	1	26d6h UP	L4OK in 0ms	100	Y	-	2	1	1m14s	-
lcgfts02.gridpp.rl.ac.uk	0	0	128	1	17		0	7	-	2 894 858	2 894 858	1s	28 101 667 849	23 359 961 517		0		0	5	0	3	26d5h UP	L4OK in 0ms	100	Y	-	2	1	3m19s	-
lcgfts05.gridpp.rl.ac.uk	0	0	128	1	16		1	8	-	2 894 958	2 894 958	0s	28 128 608 186	23 438 183 725		0		0	1	0	4	26d5h UP	L4OK in 0ms	100	Y	-	2	1	1m24s	-
lcgfts06.gridpp.rl.ac.uk	0	0	128	1	17		0	7	-	2 894 812	2 894 812	0s	28 112 284 819	23 360 494 483		0		0	19	0	2	26d5h UP	L4OK in 0ms	100	Y	-	2	1	3m19s	-
lcgfts07.gridpp.rl.ac.uk	0	0	128	1	17		0	10	-	2 894 938	2 894 938	1s	28 116 693 414	23 424 376 011		0		0	5	0	1	26d5h UP	L4OK in 0ms	100	Y	-	3	1	2m4s	-
lcgfts08.gridpp.rl.ac.uk	0	0	128	1	17		0	6	-	2 894 976	2 894 976	1s	28 114 668 807	23 394 727 503		0		0	3	0	2	26d5h UP	L4OK in 0ms	100	Y	-	3	1	1m14s	-
lcgfts09.gridpp.rl.ac.uk	0	0	128	0	17		0	10	-	2 894 934	2 894 934	2s	28 098 833 615	23 387 847 445		0		0	4	0	0	26d5h UP	L4OK in 0ms	100	Y	-	3	1	2m19s	-
lcgfts10.gridpp.rl.ac.uk	0	0	128	1	17		0	10	-	2 894 894	2 894 894	0s	28 110 249 157	23 433 561 718		0		0	5	0	2	26d5h UP	L4OK in 0ms	100	Y	-	4	1	2m44s	-
Backend	0	0		6	134		1	41	400	23 159 348	23 159 362	0s	224 908 065 549	187 251 100 969	0	0		0	47	0	15	39d23h UP		800	8	0		0	0s	

fts3-prod-rest

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	39	-	0	52	4 000	1 706 112			12 529 044 655	28 136 194 852	0	0	0					OPEN								
lcgfts01.gridpp.rl.ac.uk	0	0	128	0	5		0	8	-	212 923	212 923	11s	1 565 852 353	3 510 196 523		0		0	0	0	1	39d23h UP	L6OK in 1ms	100	Y	-	7	0	0s	-
lcgfts02.gridpp.rl.ac.uk	0	0	128	0	4		0	7	-	212 923	212 923	7s	1 565 708 698	3 503 881 171		0		0	1	0	0	39d23h UP	L6OK in 0ms	100	Y	-	6	0	0s	-
lcgfts05.gridpp.rl.ac.uk	0	0	128	0	5		0	10	-	212 923	212 923	3s	1 567 147 725	3 534 549 467		0		0	1	0	0	39d23h UP	L6OK in 1ms	100	Y	-	6	0	0s	-
lcgfts06.gridpp.rl.ac.uk	0	0	128	0	5		0	8	-	212 922	212 922	28s	1 565 431 424	3 512 621 484		0		0	0	0	0	39d23h UP	L6OK in 1ms	100	Y	-	3	0	0s	-
lcgfts07.gridpp.rl.ac.uk	0	0	128	0	5		0	8	-	212 922	212 922	25s	1 565 777 590	3 542 839 763		0		0	0	0	0	39d23h UP	L6OK in 1ms	100	Y	-	6	0	0s	-
lcgfts08.gridpp.rl.ac.uk	0	0	128	0	5		0	7	-	212 922	212 922	21s	1 566 976 558	3 506 603 581		0		0	0	0	0	39d23h UP	L6OK in 1ms	100	Y	-	5	0	0s	-
lcgfts09.gridpp.rl.ac.uk	0	0	128	0	5		0	8	-	212 922	212 922	18s	1 566 211 920	3 505 725 270		0		0	0	0	1	39d23h UP	L6OK in 6ms	100	Y	-	3	0	0s	-
lcgfts10.gridpp.rl.ac.uk	0	0	128	0	5		0	8	-	212 922	212 922	14s	1 565 938 387	3 519 777 593		0		0	1	0	2	39d23h UP	L6OK in 1ms	100	Y	-	5	0	0s	-
Backend	0	0		0	39		0	52	400	1 706 112	1 703 379	3s	12 529 044 655	28 136 194 852	0	0		0	3	0	4	39d23h UP		800	8	0		0	0s	

fts3-prod-mon

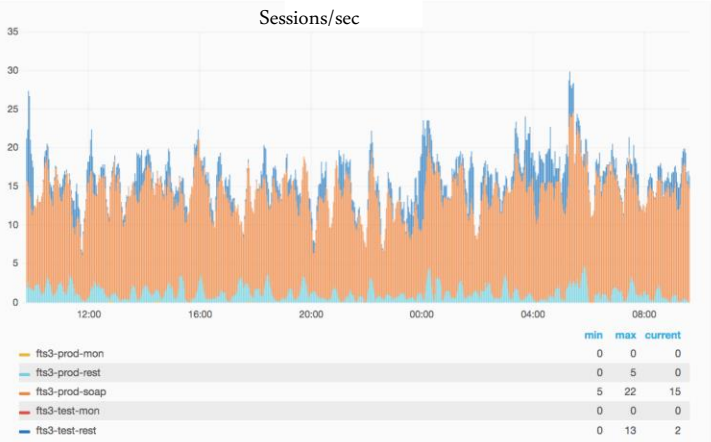
	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	128	-	0	12	4 000	32 672			55 316 478	307 618 913	0	0	0					OPEN								
lcgfts01.gridpp.rl.ac.uk	0	0	128	0	25		0	7	-	19 210	19 210	25m15s	20 067 869	141 025 775		0		0	0	0	0	39d23h UP	L6OK in 1ms	100	Y	-	7	0	0s	-
lcgfts02.gridpp.rl.ac.uk	0	0	128	0	23		0	6	-	179	179	47m18s	141 749	1 010 389		0		0	0	0	0	39d23h UP	L6OK in 0ms	100	Y	-	6	0	0s	-
lcgfts05.gridpp.rl.ac.uk	0	0	128	0	16		0	6	-	719	719	6h27m	596 661	4 829 779		0		0	0	0	0	39d23h UP	L6OK in 1ms	100	Y	-	5	0	0s	-
lcgfts06.gridpp.rl.ac.uk	0	0	128	0	7		0	6	-	2 893	2 893	4d1h	4 800 444	61 924 174		0		0	0	0	0	39d23h UP	L6OK in 1ms	100	Y	-	1	0	0s	-
lcgfts07.gridpp.rl.ac.uk	0	0	128	0	16		0	6	-	300	300	5h41m	327 943	5 153 239		0		0	0	0	0	39d23h UP	L6OK in 1ms	100	Y	-	5	0	0s	-
lcgfts08.gridpp.rl.ac.uk	0	0	128	0	14		0	12	-	147	147	6h29m	206 666	1 146 400		0		0	0	0	0	39d23h UP	L6OK in 1ms	100	Y	-	5	0	0s	-
lcgfts09.gridpp.rl.ac.uk	0	0	128	0	43		0	6	-	665	665	4d15h	853 509	3 284 841		0		0	0	0	0	39d23h UP	L6OK in 6ms	100	Y	-	2	0	0s	-
lcgfts10.gridpp.rl.ac.uk	0	0	128	0	128		0	6	-	8 551	8 551	56s	28 321 637	89 244 316		0		0	0	0	0	39d23h UP	L6OK in 1ms	100	Y	-	3	0	0s	-
Backend	0	0		0	128		0	12	400	32 672	32 664	56s	55 316 478	307 618 913	0	0		0	0	0	0	39d23h UP		800	8	0		0	0s	

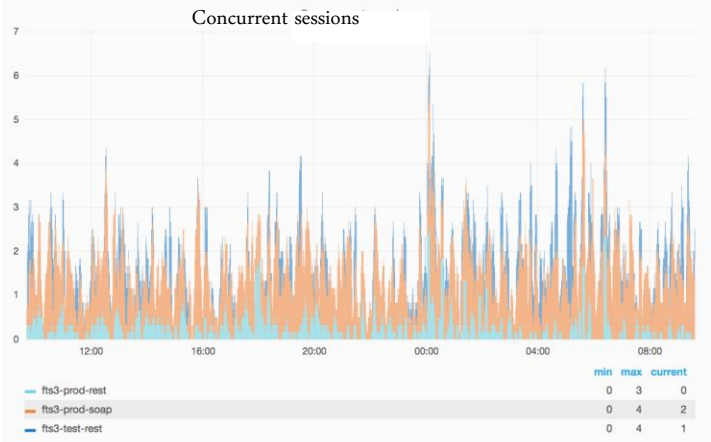
fts3-test-soap

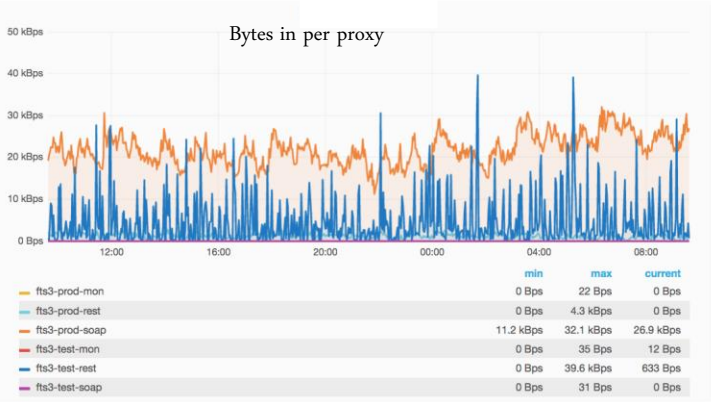
	Queue			Session rate			Sessions						Bytes		Denied		Errors		Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	28	-	0	4	4 000	247			149 120	691 256	0	0	0					OPEN								

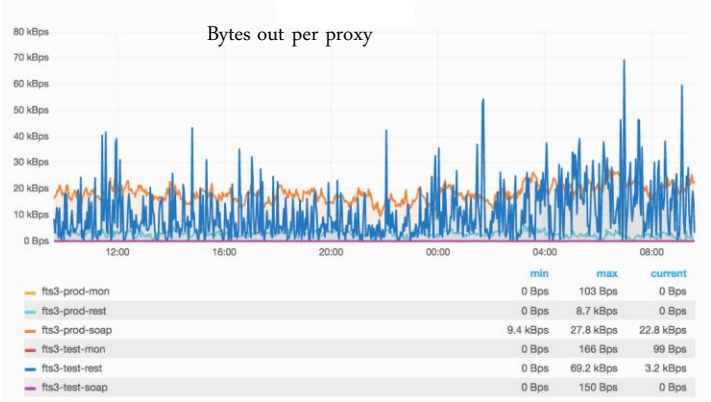
Monitoring

- Using Telegraf to send HAProxy metrics to InfluxDB
  - Telegraf has an input plugin for HAProxy









FTS alerting

- Nagios tests per load balancer
  - check that the number of healthy backend servers* for each service is about a minimum threshold

Check proc HAProxy	PASV	OK	13:15:13	92d 2h 11m 16s	1/3	PROCS OK: 1 process with command name haproxy
Check proc monit	PASV	OK	13:15:00	92d 2h 9m 53s	1/3	PROCS OK: 1 process with command name monit
Check procs keepalived	PASV	OK	13:15:13	92d 2h 8m 36s	1/3	PROCS OK: 3 processes with command name keepalived
Check proxy fts3-prod-mon		OK	13:02:17	4d 19h 38m 32s	1/3	Check haproxy OK - checked proxies: fts3-prod-mon
Check proxy fts3-prod-rest		OK	13:02:17	4d 19h 38m 32s	1/3	Check haproxy OK - checked proxies: fts3-prod-rest
Check proxy fts3-prod-soap		OK	13:02:17	4d 19h 38m 32s	1/3	Check haproxy OK - checked proxies: fts3-prod-soap
Check proxy fts3-test-mon		OK	13:02:17	5d 1h 42m 34s	1/3	Check haproxy OK - checked proxies: fts3-test-mon
Check proxy fts3-test-rest		OK	13:02:17	5d 1h 42m 34s	1/3	Check haproxy OK - checked proxies: fts3-test-rest
Check proxy fts3-test-soap		OK	13:02:17	5d 1h 42m 34s	1/3	Check haproxy OK - checked proxies: fts3-test-soap

- Tests for floating IPs
  - basic TCP checks from the Nagios server

Service	Status	Last Check	Duration	Attempt	Status Information
Check floating IP fts3-prod-ip1	OK	13:16:16	31d 22h 54m 33s	1/5	TCP OK - 0.000 second response time on port 8446
Check floating IP fts3-prod-ip2	OK	13:16:16	18d 23h 3m 18s	1/5	TCP OK - 0.000 second response time on port 8446
Check floating IP fts3-test-ip1	OK	13:16:16	88d 20h 33m 30s	1/5	TCP OK - 0.002 second response time on port 8446
Check floating IP fts3-test-ip2	OK	13:16:16	98d 23h 40m 20s	1/5	TCP OK - 0.000 second response time on port 8446

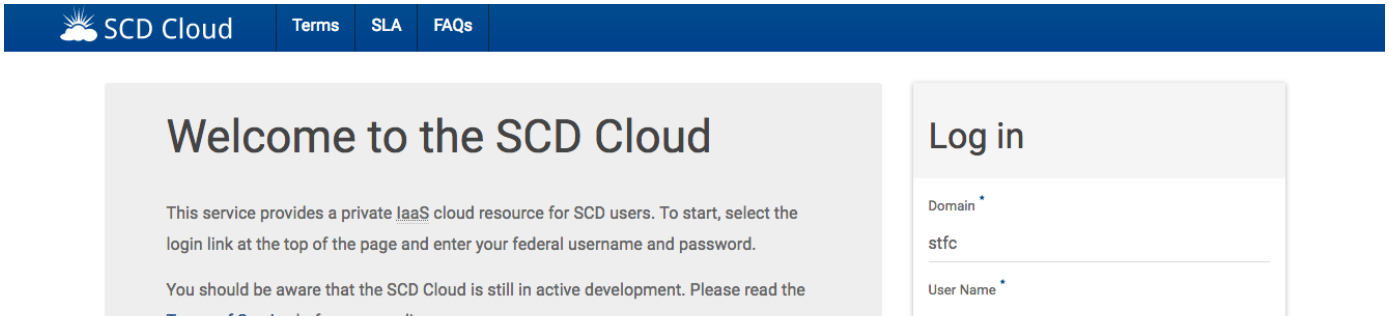
according to health checks run by HAProxy

FTS pager alarms

- What we used to do with FTS2
  - pager triggered even if a single FTS server had issues
- FTS3 (before use of load balancers)
  - pager triggered only if more than 2 FTS servers had issues
  - service would continue to function but be degraded
    - some fraction of attempts to access the service would fail
- FTS3 (with load balancers)
  - pager triggered only if more than 2 FTS servers have issues
  - service continues to function
  - problem is invisible to users

Another example: OpenStack

- Our OpenStack deployment designed to be highly available at every possible level
- Multiple nodes for each OpenStack service
  - Galera cluster for HA MariaDB
  - MongoDB with replication for Ceilometer
  - RabbitMQ with HA queues
  - Neutron with Distributed Virtual Routers using OpenVSwitch
  - HAProxy & Keepalived for load balancing & SSL



OpenStack

- All OpenStack communication goes through HAProxy
  - single set of 3 hosts running HAProxy
- Almost all OpenStack APIs have SSL termination by HAProxy
  - Keystone has Apache terminating SSL for use with VOMS for EGI FedCloud
  - However Keystone traffic still goes through HAProxy
- Keepalived is used to provide a floating IP between HAProxy nodes

Experience so far

- Keepalived
  - working reliably
  - some confusion initially caused by ntpd
    - it sometimes deletes virtual IP addresses!
    - adjusted ntpd configuration so it only listens on the host's actual IP address
  - a few incidences of unexplained packet loss on the RAL network caused the floating IP addresses to "flap" occasionally
    - Keepalived on each LB couldn't see the VRRP ads from the other host
    - not noticed by users
- HAProxy
  - working reliably

Current status

- Production services using the load balancers
  - FTS3 “test” instance (ATLAS) since 26th April
  - FTS3 “prod” instance (CMS + other VOs) since 31st May
- Services in development using load balancers
  - OpenStack
  - Dynafed
- New services likely to use load balancers in production
  - GridFTP gateway to Ceph
  - S3/Swift APIs for Ceph
- Existing services where load balancers could be beneficial
  - CASTOR SRMs
  - BDIIs, LFCs (still used by some non-LHC VOs)
  - MyProxy
  - NGI Argus

Summary & future plans

- Successfully using a highly available load balancer in front of a standard grid service – FTS3
  - enables us to more transparently carry out interventions
  - problems no longer visible to users
  - can add/remove hosts transparently
- It's an important step towards being able to have a more dynamic environment
  - internal infrastructure is hidden from users
  - essential for a dynamic container-based infrastracture
- Future plans
  - More thorough HAProxy health checks
  - Migrate more services