Speeding up Samba by backing up

Experiences in implementing and optimizing Active Directory features in Samba

catalyst

open source technologists

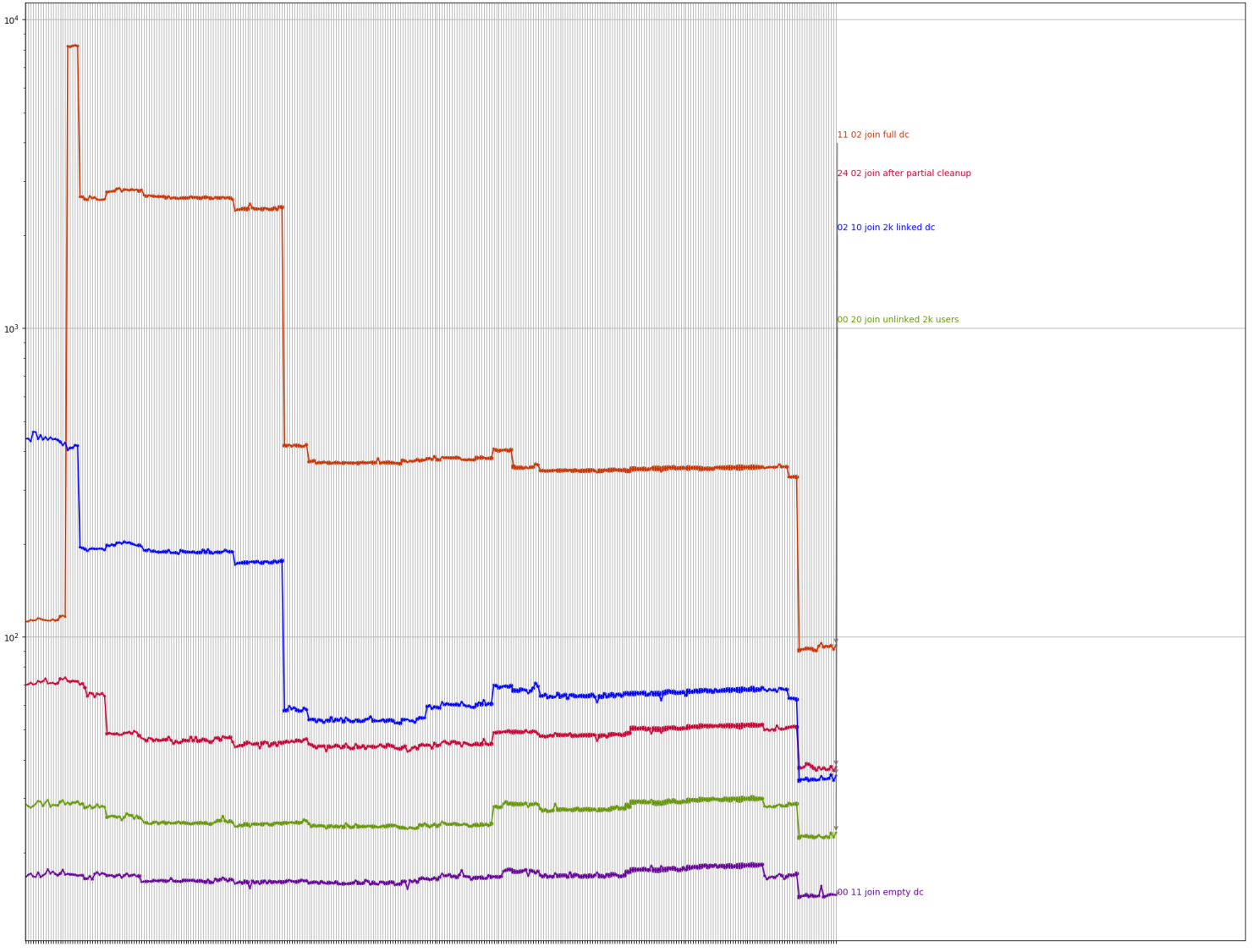
What has been done in the last year?

Samba 4.9

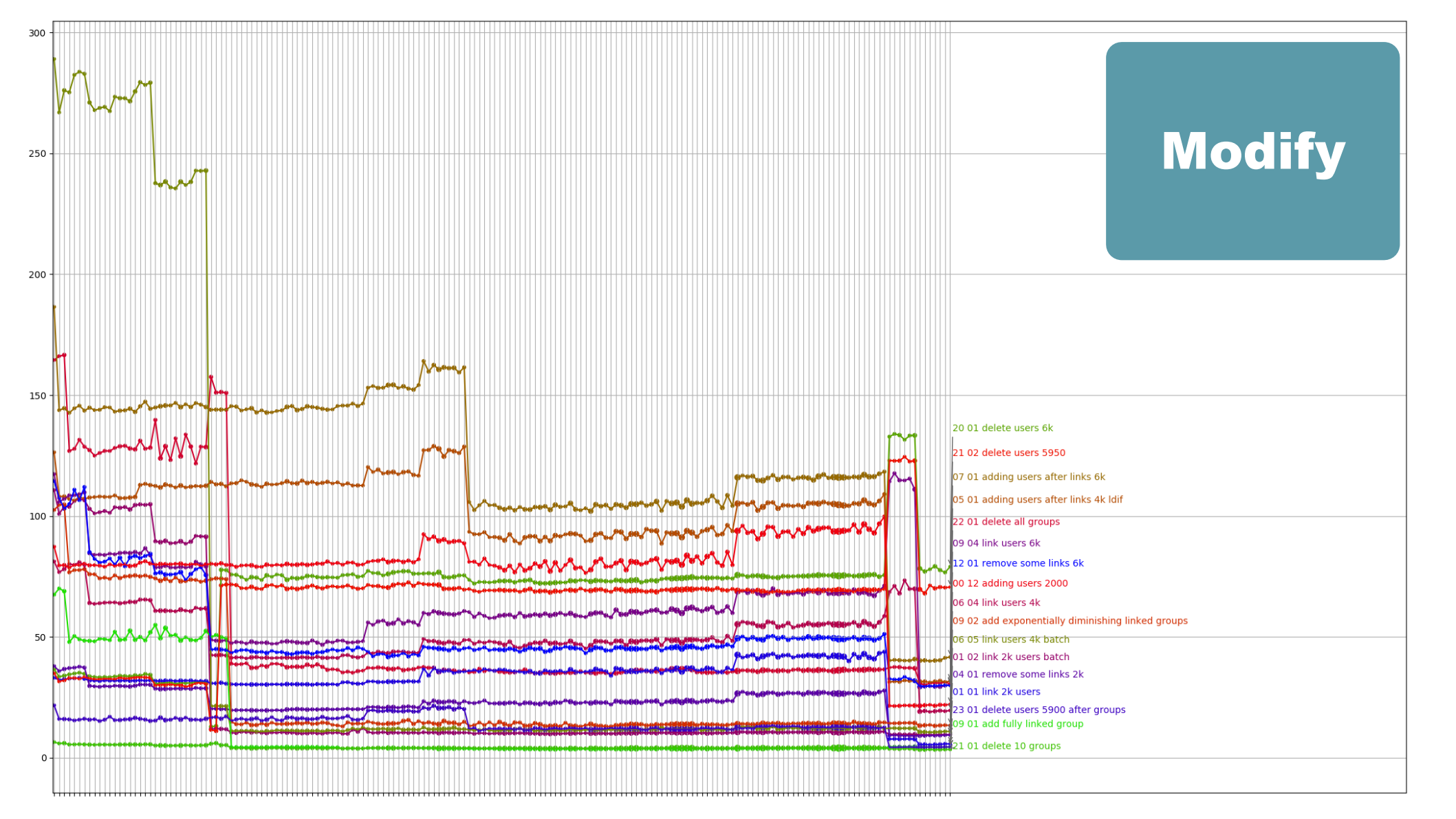
- Password and membership change auditing
- LMDB back-end (semi-experimental)
- Fine grained password policies
- Domain backup, restore and rename tools
- Better DRS partner visualization
- Automatic DNS site coverage
- DNS scavenging support
- Improved trust support and more...

Samba 4.10

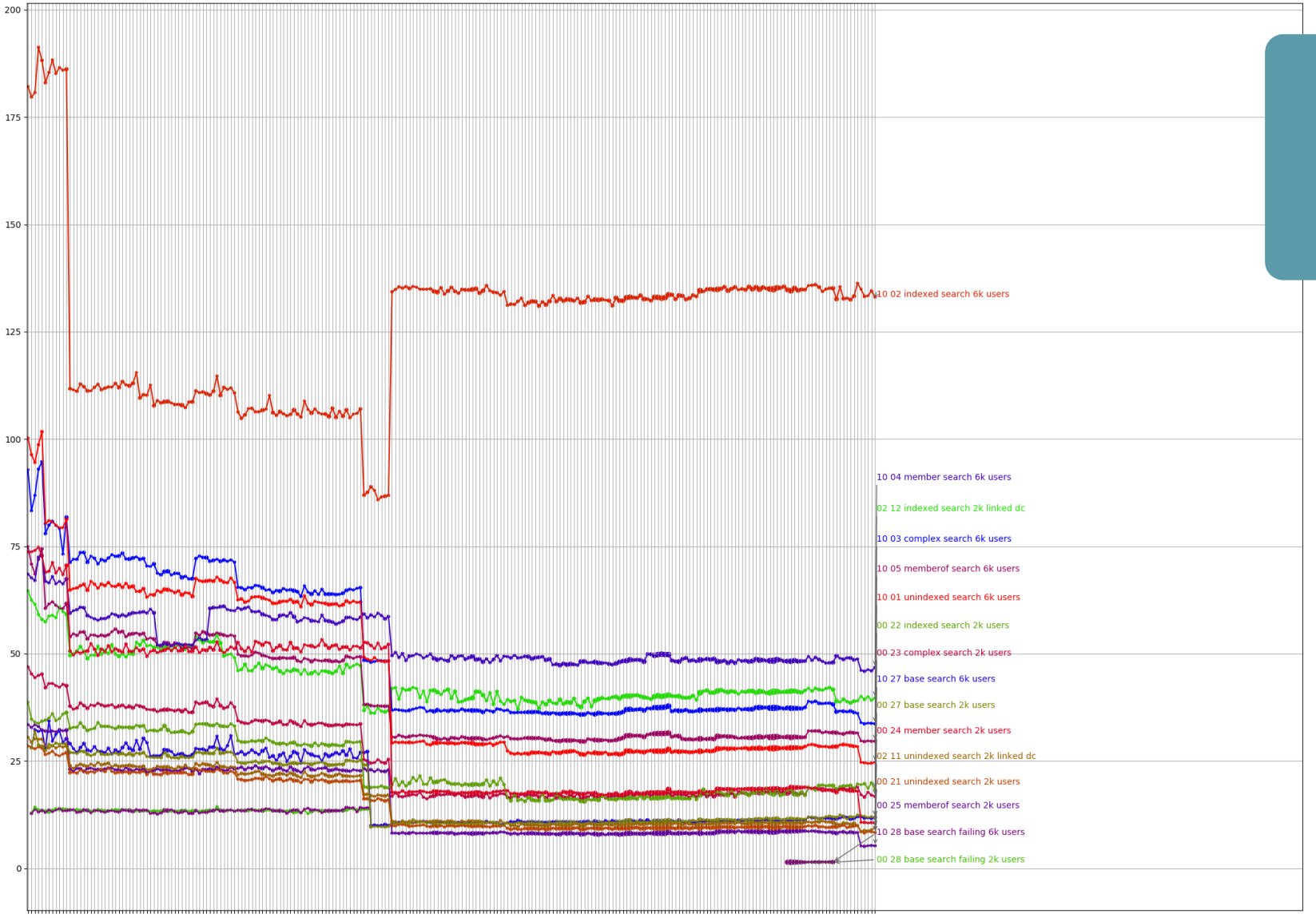
- GPO import and export
- KDC and NETLOGON prefork (default in 4.11)
- (Prefork) improvements for restarting services automatically
- Changes to LDAP paged results to save memory
- Offline domain backup
- Python 3 support
- Audit logging with MS event IDs and more...


10^4
10^3
10^2
11 02 join full dc
24 02 join after partial cleanup
02 10 join 2k linked dc
00 20 join unlinked 2k users
00 11 join empty dc

Join


Modify
300
250
200
150
100
50
0
20 01 delete users 6k
21 02 delete users 5950
07 01 adding users after links 6k
05 01 adding users after links 4k ldif
22 01 delete all groups
09 04 link users 6k
12 01 remove some links 6k
00 12 adding users 2000
06 04 link users 4k
09 02 add exponentially diminishing linked groups
06 05 link users 4k batch
01 02 link 2k users batch
04 01 remove some links 2k
01 01 link 2k users
23 01 delete users 5900 after groups
09 01 add fully linked group
21 01 delete 10 groups

Search


200
175
150
125
100
75
50
25
0
10 02 indexed search 6k users
10 04 member search 6k users
02 12 indexed search 2k linked dc
10 03 complex search 6k users
10 05 memberof search 6k users
10 01 unindexed search 6k users
00 22 indexed search 2k users
00 23 complex search 2k users
10 27 base search 6k users
00 27 base search 2k users
00 24 member search 2k users
02 11 unindexed search 2k linked dc
00 21 unindexed search 2k users
00 25 memberof search 2k users
10 28 base search failing 6k users
00 28 base search failing 2k users

Performance, performance, performance

Replication improvements, linked attribute performance, rename performance, large scale improvements, ... as well as other things like schema updates

Traffic replay runner

Basic steps for replaying traffic

Network trace

Run wireshark and get a pcap output

Traffic summary

Anonymize the traffic and pick out important details to replay

Traffic model (optional)

Create a statistical model for generating proportionally similar traffic

Basic steps for replaying traffic

Play traffic

Run either the summary or the model file

Analyze the results

Successes or failures, median, mean, max, 95th

Basic steps for replaying traffic

Play traffic

Run either the summary or the model file

Analyze the results

Successes or failures, median, mean, max, 95th

That's it!

We're fast, 100,000 users, no problems!

Naive traffic runner results (2 vCPU, 8GB RAM)

v4.6 – 113 operations / second

v4.7 – 94 operations / second (changes to LDAP multi-process)

v4.8 – 154 operations / second (only in new prefork process mode)

v4.9 – 157 operations / second (only in prefork mode)

v4.10 – Same as 4.8 and 4.9

Git master (prefork is default) – possibly 160?

Traffic sample is largely DNS, name resolution, LDAP bind, NETLOGON

So... backing up?

Domain backup

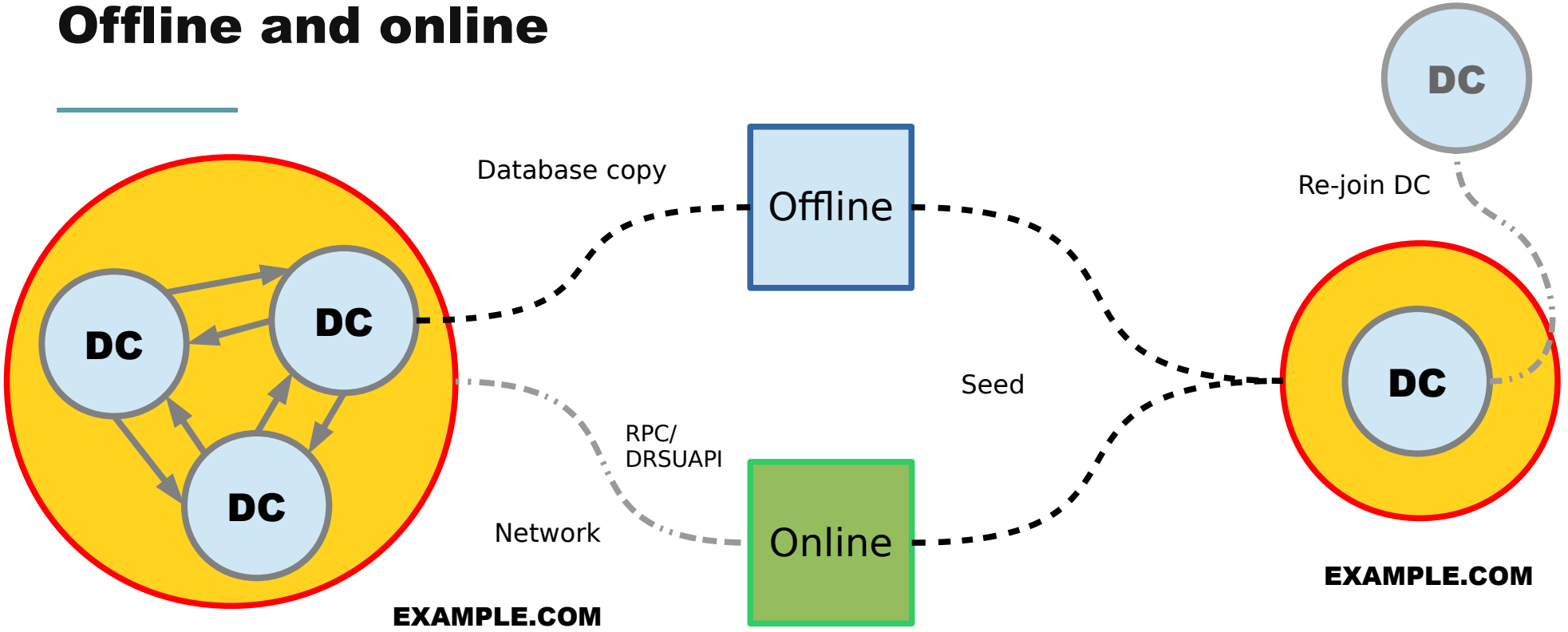
A new method of backing up an AD Domain in Samba 4.9 + 4.10

Why?

- Existing samba_backup script had a number of problems
  - With a running DC it wasn't certain to produce a valid copy
  - It was safer than a standard copy, but didn't respect lock ordering
  - Might have caused deadlocks, corrupt or inconsistent (secrets) data

- Single source of truth of the domain data (multi-master replication)
  - Forcing a pristine backup to override corrupt data elsewhere is non-trivial
  - Restoring into competing data, might look replicated due to old versioning
  - Avoid some database inconsistencies by creating a replication (online) backup

Offline and online



`samba-tool domain backup [online|offline]` → Tar file → `samba-tool domain backup restore`

https://wiki.samba.org/index.php/Back_up_and_Restoring_a_Samba_AD_DC

Issues to resolve?

- The tool doesn't exactly replace samba_backup (despite being removed)
  - samba-tool domain backup can't restore to the same DC name
  - samba-tool domain backup can't restore to the same install location
  - Copying of sysvol still seems buggy from the mailing list

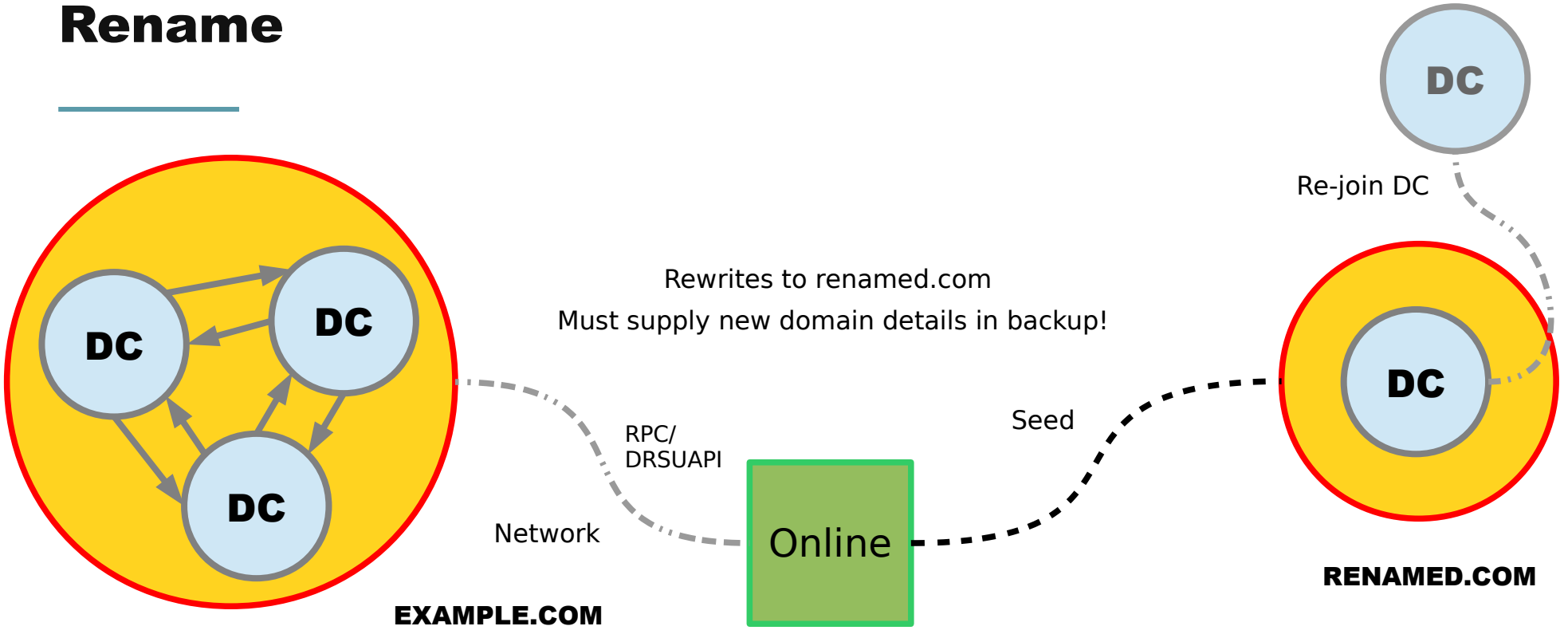
- For those who re-deploy in a certain way, it's the (almost) ideal tool
- For those who know to re-join or re-sync (often not perfectly but perhaps in cases where it isn't that critical) it's a new hassle

- Backup of a domain, or backup of a domain controller?

Domain rename

Create testing environments and lab domains (without passwords and secrets)

Rename



`samba-tool domain backup rename` → Tar file → `samba-tool domain backup restore`

https://wiki.samba.org/index.php/Create_a_samba_lab-domain

Benefits and Caveats

- Much less worries about production and pre-production interacting
- Firewalling should be more straightforward
- Experimenting with load and load testing different hardware

- No explicit secrets (or close to it) isn't anonymized or secret-free
- The data in the domain means it can still serve the old DNS records
- Rebuilding the sites and subnets is still a job on its own (automation?)

- Use in production is debateable...

Benefits and Caveats (custom DC testenv)

```
BACKUP_FILE=backup-offline.tar.bz2 SELFTEST_TESTENV=customdc make testenv
```

- Reproducible testing is easier, upgrade testing is easier
- Testing under different conditions is much easier
- Having a clean DC before every test is possible

Linux Namespaces

Running under socket_wrapper (default test-bed for samba testing), we find a 10-20% performance hit when using LMDB.

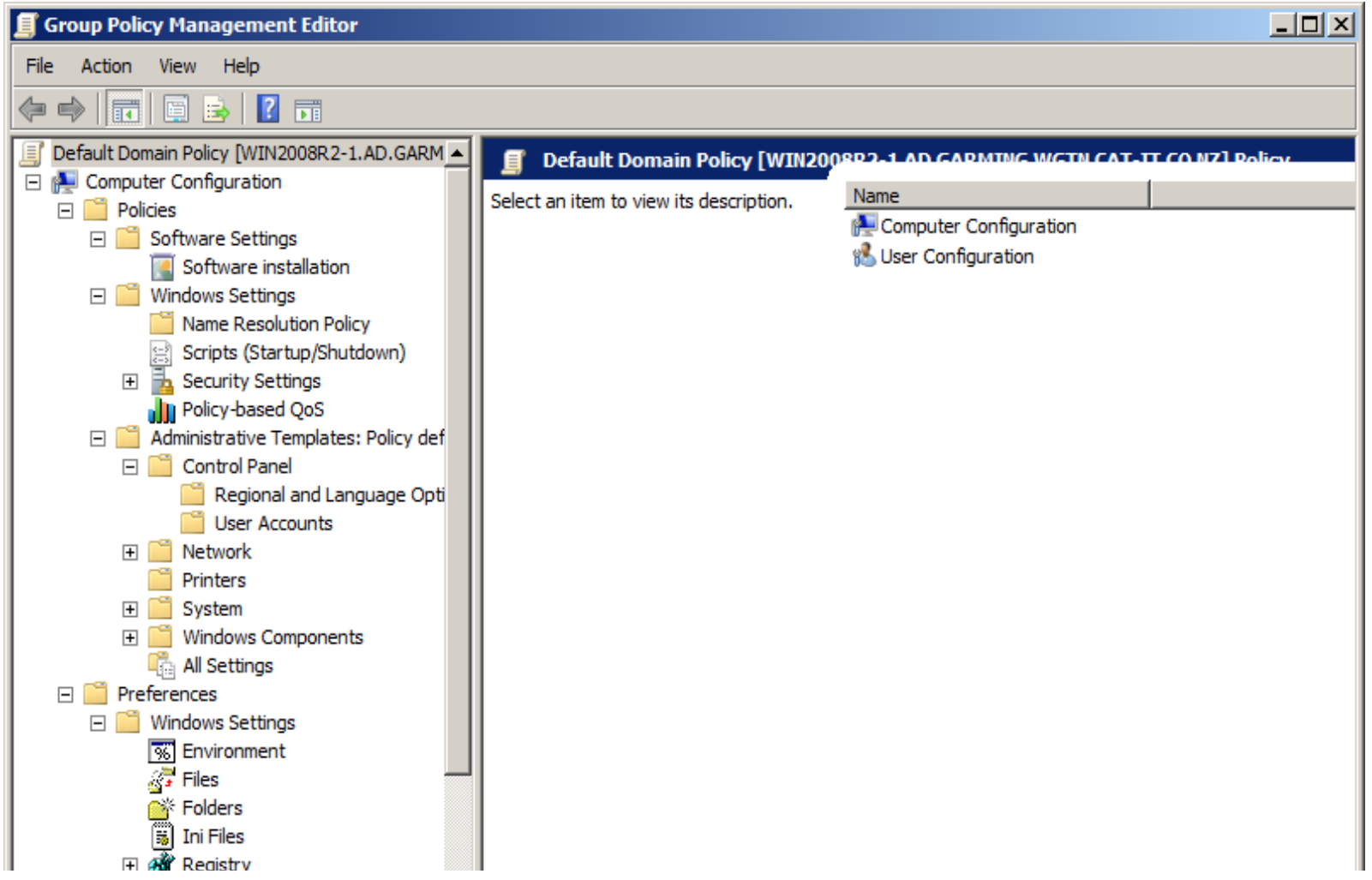
- Why not leave the network faking to the kernel?
- Why not fake our hostnames and override DNS resolution using the kernel?

Completely isolated test-bed using ‘real’ network interfaces that can still be made to interact with the real system and virtual machines. Unfortunately still problems with UID fakery (apparently Docker is hard), but it works.

GPO import/export

A new way of copying over a SYSVOL that functions (ish) across domains

Exports to XML with XML entities
Ideal with domain rename (pre-prod)


Group Policy Management Editor
File
Action
View
Help
Default Domain Policy [WIN2008R2-1.AD.GARM
Computer Configuration
Policies
Software Settings
Software installation
Windows Settings
Name Resolution Policy
Scripts (Startup/Shutdown)
Security Settings
Policy-based QoS
Administrative Templates: Policy def
Control Panel
Regional and Language Opti
User Accounts
Network
Printers
System
Windows Components
All Settings
Preferences
Windows Settings
Environment
Files
Folders
Ini Files
Registry
Default Domain Policy [WIN2008R2-1.AD.GARMING-WGTN.CAT-IT.CO.NZ] Policy
Select an item to view its description.
Name
Computer Configuration
User Configuration

MS-GPOL

MS-GPOD

fdeploy1.ini

MS-GPOL

MS-GPOD

audit.csv

GptTmpl.inf

fdeploy1.ini

MS-GPOL

.xml

registry.pol

MS-GPOD

.aas

audit.csv

GptTmpl.inf

fdeploy1.ini

MS-GPOL

User/Documents & Settings

.xml

registry.pol

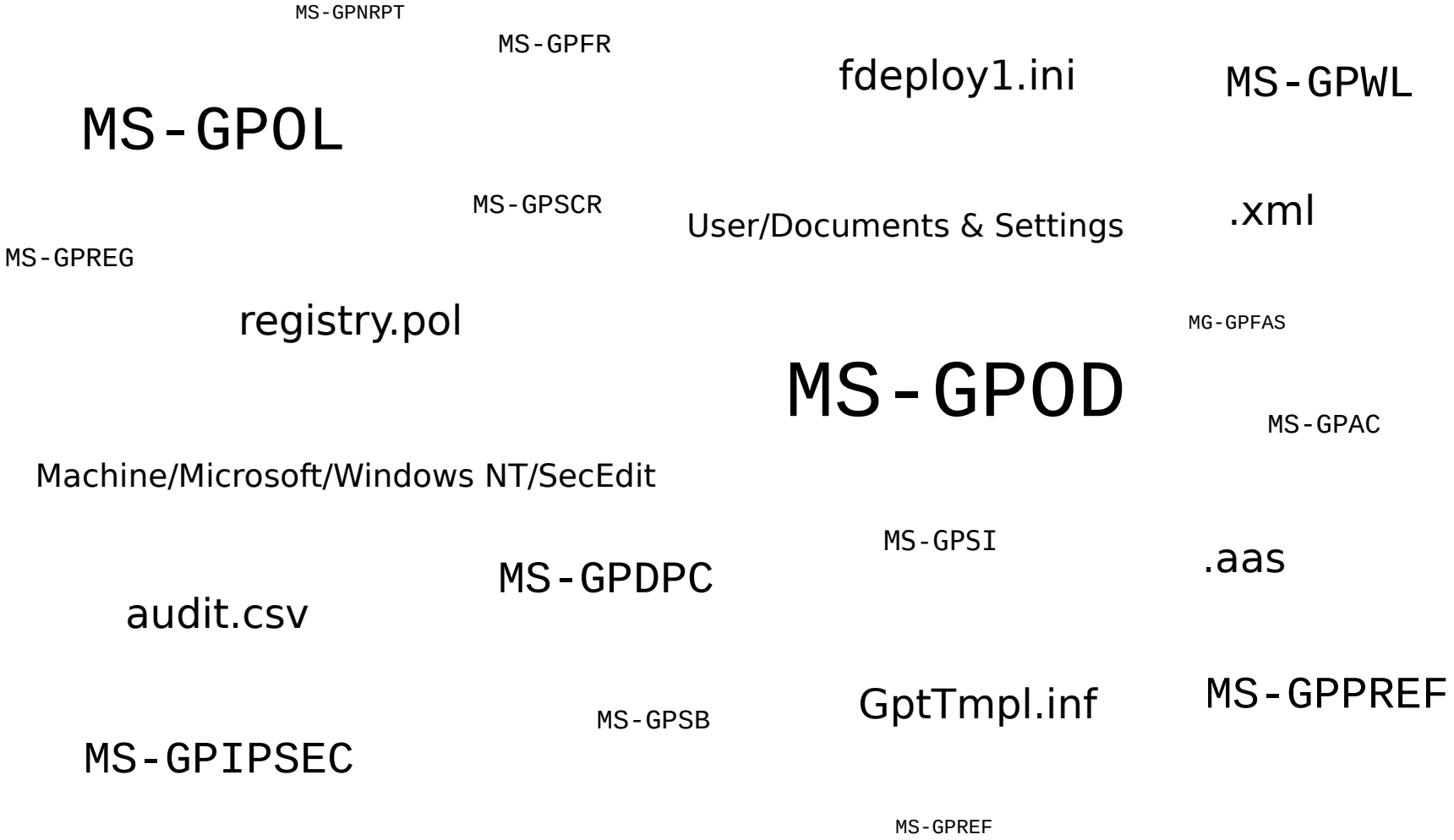
MS-GPOD

Machine/Microsoft/Windows NT/SecEdit

.aas

audit.csv

GptTmpl.inf


MS-GPNRPT
MS-GPFR
fdeploy1.ini
MS-GPWL
MS-GPOL
MS-GPSCR
User/Documents & Settings
.xml
MS-GPREG
registry.pol
MG-GPFAS
MS-GPOD
MS-GPAC
Machine/Microsoft/Windows NT/SecEdit
MS-GPSI
.aas
MS-GPDPC
audit.csv
GptTmpl.inf
MS-GPPREF
MS-GPSB
MS-GPIPSEC
MS-GPREF

Using GPO Import/Export

```
samba-tool gpo backup
samba-tool gpo restore

samba-tool gpo backup --generalize --entities=$OUT_PATH
samba-tool gpo restore --entities=$IN_PATH
```

```
<!ENTITY SAMBA____USER_ID_____7b7bc2512ee1fedcd76bdc68926d4f7b__ "Guest">
```

https://wiki.samba.org/index.php/GPO_Backup_and_Restore

Automation

Actually running the traffic runner for real (making it reproducible and periodic)

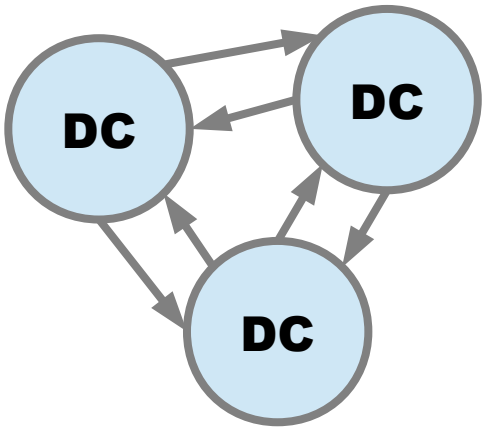
Automation

- Virtual machines → cloud (sometimes too slow)
- Openstack HEAT templates, Bash scripts
- Ansible playbooks

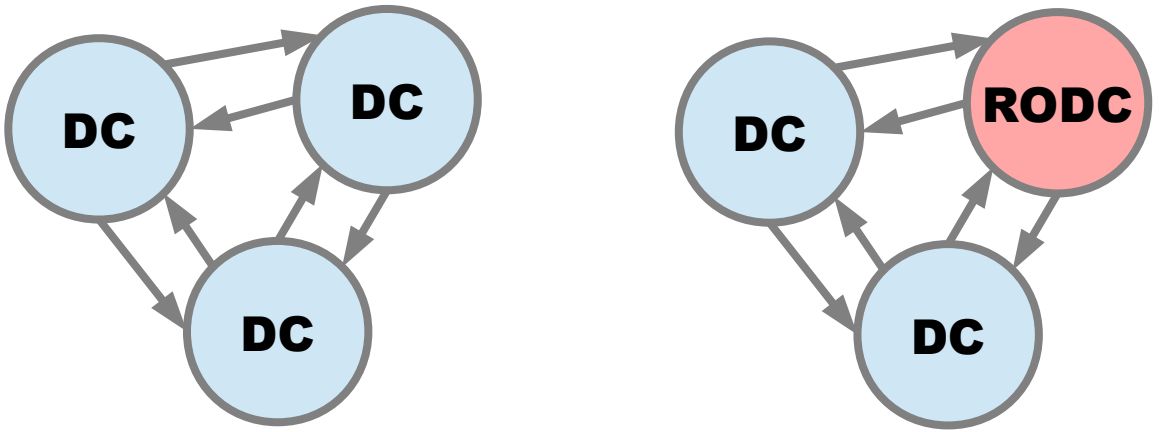
Still has its problems, but we now have a mostly re-usable and composable set of playbooks (modules) for different AD environments using YAML files.

This work has led to upstream automation work, bootstrap code to simplify package installations across different platforms (more natural fit in the source tree).

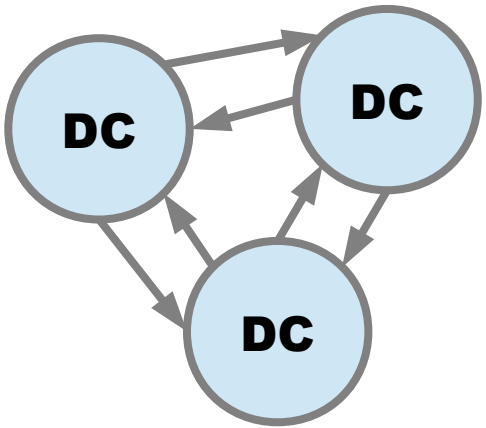
Automation

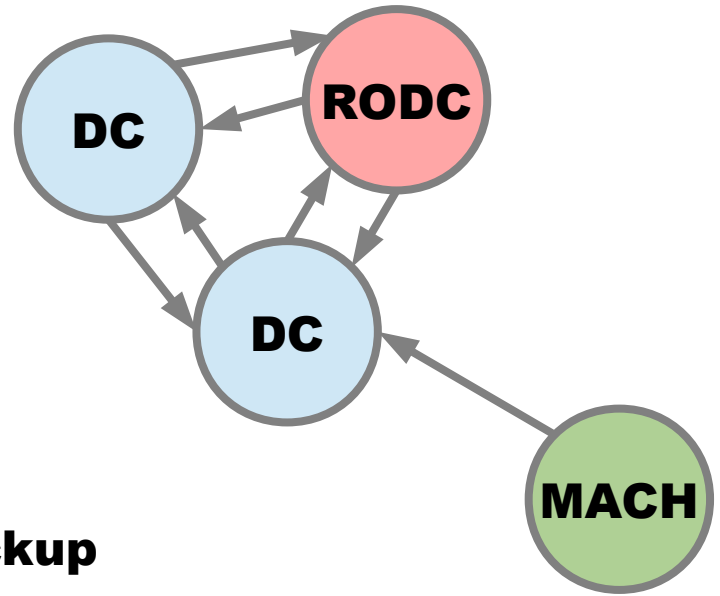


Automation



Automation





Seed AD domain from a backup

Automation

- GUI → YAML
- Backed by Docker or Vagrant instead of Openstack
- How do we integrate the self-test system?
- Can we use this infrastructure to run against Windows regularly?

Useful for development, probably overkill (or not a great fit) for production:
https://gitlab.com/catalyst-samba

ansible-role-samba-dc

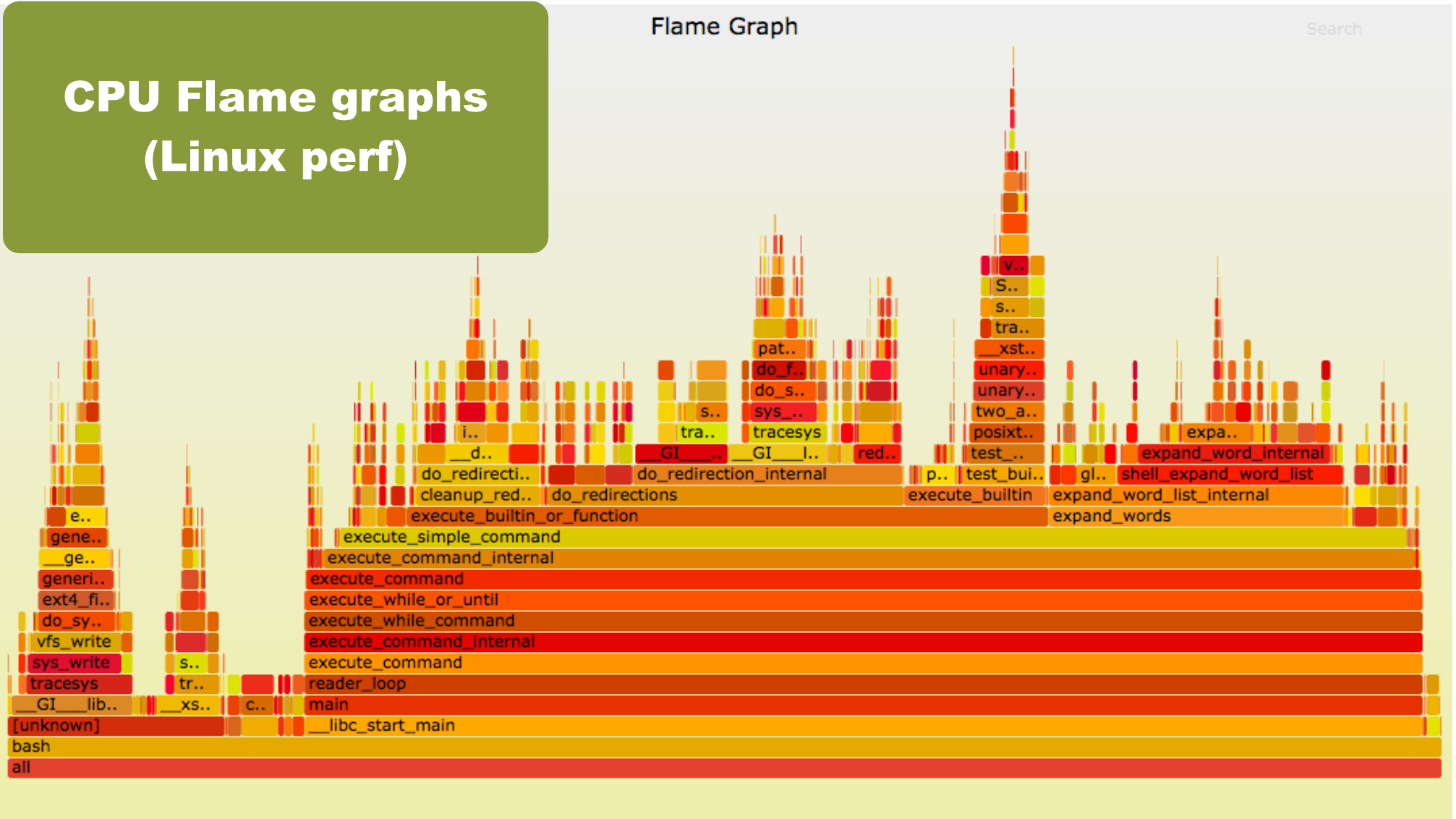
ansible-role-samba-common

Replicating… forever

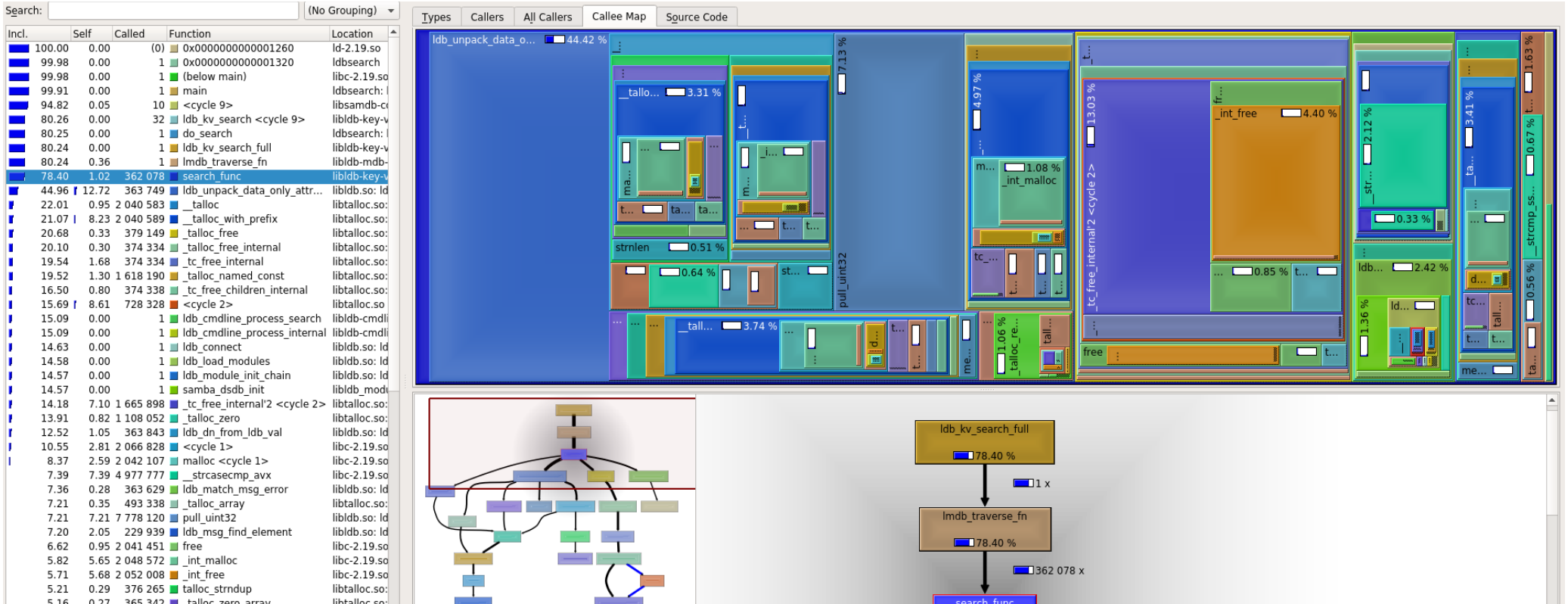
After joining a new domain controller to a restored domain, ongoing replication would never end.

Why doesn’t it only take as long as the join (30 minutes)?

CPU Flame graphs (Linux perf)



Callgrind


Search:
(No Grouping)
Incl. Self Called Function Location
100.00 0.00 (0) 0x0000000000001260 ld-2.19.so
99.98 0.00 1 0x0000000000001320 ldbsearch
99.98 0.00 1 (below main) libc-2.19.so
99.91 0.00 1 main ldbsearch:
94.82 0.05 10 <cycle 9> libsamdb-c
80.26 0.00 32 ldb_kv_search <cycle 9> libldb-key-v
80.25 0.00 1 do_search ldbsearch:
80.24 0.00 1 ldb_kv_search_full libldb-key-v
80.24 0.36 1 lmdb_traverse_fn libldb-mdb-
78.40 1.02 362 078 search_func libldb-key-v
44.96 12.72 363 749 ldb_unpack_data_only_attr... libldb.so: ld
22.01 0.95 2 040 583 __talloc libtalloc.so:
21.07 8.23 2 040 589 __talloc_with_prefix libtalloc.so:
20.68 0.33 379 149 _talloc_free libtalloc.so:
20.10 0.30 374 334 _talloc_free_internal libtalloc.so:
19.54 1.68 374 334 _tc_free_internal libtalloc.so:
19.52 1.30 1 618 190 _talloc_named_const libtalloc.so:
16.50 0.80 374 338 _tc_free_children_internal libtalloc.so:
15.69 8.61 728 328 <cycle 2> libtalloc.so
15.09 0.00 1 ldb_cmdline_process_search libldb-cmdli
15.09 0.00 1 ldb_cmdline_process_internal libldb-cmdli
14.63 0.00 1 ldb_connect libldb.so: ld
14.58 0.00 1 ldb_load_modules libldb.so: ld
14.57 0.00 1 ldb_module_init_chain libldb.so: ld
14.57 0.00 1 samba_dsdb_init libldb_mod
14.18 7.10 1 665 898 _tc_free_internal'2 <cycle 2> libtalloc.so:
13.91 0.82 1 108 052 _talloc_zero libtalloc.so:
12.52 1.05 363 843 ldb_dn_from_ldb_val libldb.so: ld
10.55 2.81 2 066 828 <cycle 1> libc-2.19.so
8.37 2.59 2 042 107 malloc <cycle 1> libc-2.19.so
7.39 7.39 4 977 777 __strcasecmp_avx libc-2.19.so
7.36 0.28 363 629 ldb_match_msg_error libldb.so: ld
7.21 0.35 493 338 _talloc_array libtalloc.so:
7.21 7.21 7 778 120 pull_uint32 libldb.so: ld
7.20 2.05 229 939 ldb_msg_find_element libldb.so: ld
6.62 0.95 2 041 451 free libc-2.19.so
5.82 5.65 2 048 572 _int_malloc libc-2.19.so
5.71 5.68 2 052 008 _int_free libc-2.19.so
5.21 0.29 376 265 talloc_strndup libtalloc.so:
Types
Callers
All Callers
Callee Map
Source Code
ldb_unpack_data_o... 44.42 %
_tallo... 3.31 %
strnlen 0.51 %
0.64 %
pull_uint32
7.13 %
4.97 %
m... 1.08 %
_int_malloc
tc_...
13.03 %
_tc_free_internal'2 <cycle 2>
fr...
_int_free 4.40 %
0.85 %
free
2.12 %
_str... 0.33 %
ldb... 2.42 %
1.36 %
ld...
3.41 %
_ta...
_strcmp_ss... 0.67 %
0.56 %
1.63 %
_tall... 3.74 %
1.06 %
_talloc_re...
ldb_kv_search_full
78.40 %
1 x
lmdb_traverse_fn
78.40 %
362 078 x

Print debugging

top (htop/iotop)

trial and error

basic arithmetic

gdb (attach to pid)

perf top

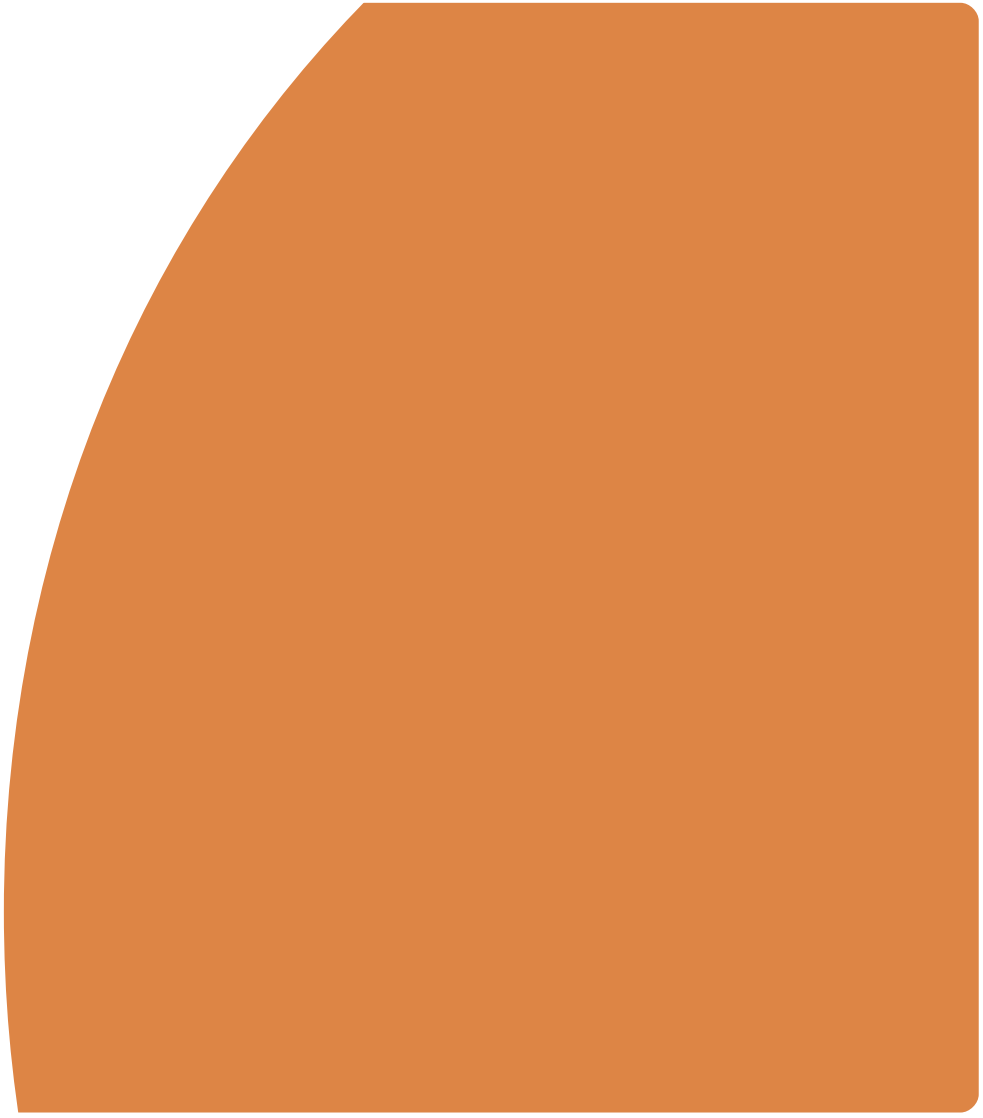
luck

Lessons

- It turns out there was a bug in the backup code, but it found real performance issues that we then fixed
- Replication seems to retrigger despite having just joined (still)
- Accidentally doing the wrong thing means running out of memory quickly with a large database.
    - Piecemeal growth ≠ dealing with everything at once
- LMDB behaves completely differently (copy-on-write)

Re-indexing

Example of an operation where our tooling failed and **SIZE MATTERS**



Re-indexing timings (mm:ss.ss)

```
100,000 users approx 230,000 records.
  Hash size         re-index time
      1,000              14:42.06
     10,000               1:59.56
    100,000                 39.92
    200,000                 37.48
    300,000                 43.16

50,000 users approx 110,000 records.
  Hash size         re-index time
      1,000               3:46:93
     10,000                 37:29
    100,000                 18.95
```

20x improvement

Basically a one line change

Traffic runner on a 50k user DC (with many links)

v4.9 – Targeting 80 operations / second (actual 32 success ops / second)

Protocol	Op Code	Description	Count	Failed	Mean	Median	95%	Range	Max
ldap	0	bindRequest	863	23	4.528840	0.563014	15.961734	203.778658	203.910120

Master – Targeting 80 operations / second (no failures + 2x throughput)

ldap	0	bindRequest	3450	0	0.505355	0.143523	2.496425	9.502704	9.546165

Traffic runner on a 50k user DC (with many links)

v4.9 – Targeting 80 operations / second (actual 32 success ops / second)

Protocol	Op Code	Description	Count	Failed	Mean	Median	95%	Range	Max
rpc_netlogon	39	SamLogonEx	1212	7	1.083997	0.458507	1.335286	60.024080	60.062607

Master – Targeting 80 operations / second (no failures + 2x throughput)

rpc_netlogon	39	SamLogonEx	1568	0	0.082939	0.017412	0.091722	13.487989	13.493821

Some operations we emulate are silly in the large database case (or latency requirements).

Should try to improve 95% numbers, but this is a fairly worst case scenario with large groups.

Working with a (more realistic) 100k user DC

1) Doesn’t page the database into memory correctly, LDAP allocates 3x the database in memory (SSD recommended)

2) Loading into caches from memory can be extremely costly (influencing the database binary storage format for 4.11)

3) LDAP bind doesn’t work pre–4.11 with users in a group of 100,000 users

4) Behaviour of sequential operations is not the same as in parallel

5) DNS???

Final takeaways

1) Real machines matter, fakery doesn’t measure performance (namespaces, docker, VM, bare-metal, modern hardware)

2) Measuring sequential operation also not helping (new tools?)

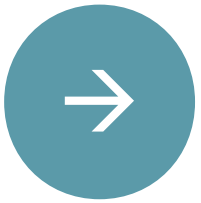
3) Repeating traffic runner runs (sys-admins should try it in a lab)

4) Reducing allocations helps in multi-process more than expected (as well as other memory manipulations)

Thanks

...

garming@catalyst.net.nz



garming@samba.org

linkedin.com/in/garming-sam

catalyst