Smaller, faster, scalier

Performance improvements for Samba 4 AD DC

SambaXP 2013 May 16th

Matthieu Patou
Samba Team

mat@samba.org

Agenda

» Disclaimer

» How did it started

» Speeding up schema modification (1/2)

» A word on callgrind

» Speeding up schema modification (2/2)

» Making faster and scalier: things you thought for granted

» Making faster and scalier: some details on how our current database

» Making it faster and scalier: Things in progress

» Making it faster and scalier: Current results

» What about the smaller, from the title ?

- What about OpenLDAP ?
- My queries are slow, what can I do ?

Disclaimer

The views and opinions in this presentation are my own
and do not necessarily reflect the views and opinions of my present,
past and future employers.

How did it started ?

- » Last year around last Samba XP, We were trying to confirm that Samba handling of the schema was quite ok
- » So we said, let's check that you can install Microsoft Exchange server on top of Samba
- » Because Exchange modifies a lot the schema of AD
- » So if it manages to install the schema without trashing the provision we could declare that our handling of schema is "quite ok"*
- » After some tweaking we somehow managed to pass the schema change in ~2 hours
- » It takes ~ 5 to 10 minutes with a Windows DC, in a VirtualBox VM ...

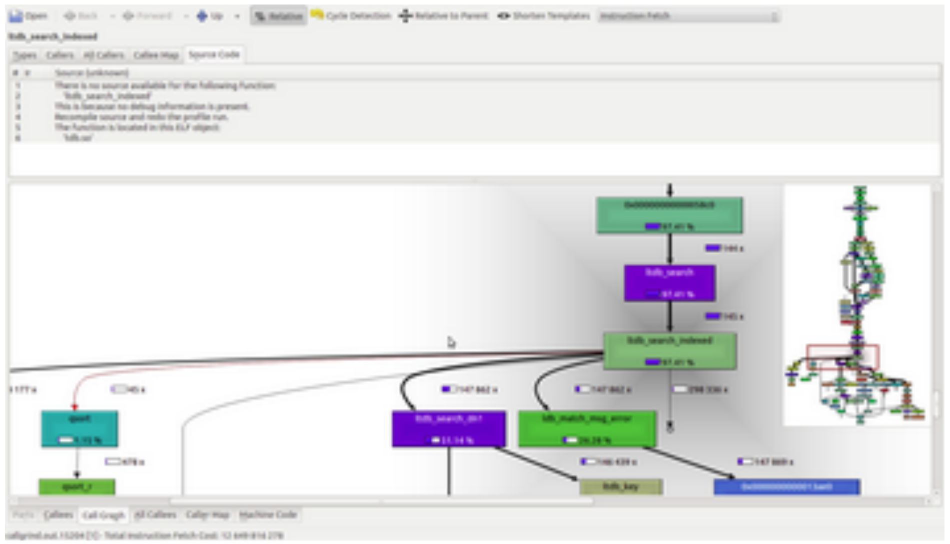
- » There was room for improvement

* Ask Metze for more details on what we don't check that could break your schema and that's why schema modification are not enabled by default

Speeding up schema modification (1/2)

- Need to trace where the time is spent
  - *gprof was not giving any insight*
  - *callgrind was (too much?) great*
- Debug where the time was spent with schema modifications in exchange was not super simple because modifications were numerous and tracing would take ages
- Hopefully adding 5 classes and 10 attributes was also much slower than it should be
- Tracing showed that we were spending most of the time reloading the current schema from the database

A word on callgrind



- » Part of the valgrind familly
- » Has a pretty neat frontend: kcachegrind
- » Has the *--instr-atstart=* option, that can be used to trace only when you need and skip the initialisation for instance (and speed-up the tracing by side effect
- » Has to be used with *callgrind_control* to control when you want to start and stop tracing (sleep can help to leave enough time to issue the commands)
- » A brain is still needed because sometime presentation of the number can be missleading

- It slows down quite a lot applications so be prepared

Speeding up schema modification (2/2)

Why it was slow:

- » In order to comply with all Microsoft implementation, schema is queried several times per LDAP or RPC request, so in order to speedup this Samba 4 (as Windows I suspect) is maintaining some structure so that every query didn't end up reading the database.
- » Creation of those structures is not cheap, and has to be redone every time the schema is changed (via RPC) or if we are asked to reload it (schemaUpdateNow via LDAP) but in real life you are not modifing the schema every day (do you ?)
- » In order to make our developer life easier, Samba 4 used to reload the schema every time it was changed when Windows

would do it only after 120 seconds or if the schemaUpdateNow "command" was issued

What has been done:

- » Brilliant idea #1: let's not reload the schema every time it changed but only after xx seconds or after the schemaUpdateNow "command"
- » Brilliant idea #2: not everybody is modifying the schema at the same time so let's cache some information (ie. next msDS-IntId) and check the USN to see if we can rely on our cache or if we have to do a real search

=> we are able to do the full install of exchange in < 30 minutes, a bit faster than Windows on my VMs

Making faster and scalier: things you thought for granted

Context:

- 70K objects (users, groups, DNS entries, contacts, what not, ...)
- Search on baseDN "*DC=test.corp,CN=MicrosoftDNS,CN=System,DC=test,DC=corp*" with criteria *(dc=server1)*
  - *Will do a full scan on your root partition will load every single record in this partition, check if it match the criteria and then filter on the dn to check it match the DN of the search*
  - *not good !*
- Search on baseDN "*OU=SmallOU,OU=MyCompany,DC=test,DC=corp*" with

criteria *(objectclass=user)*

- » *Will do an indexed search (great) and load all objects that match the criteria (a lot) and then will filter those that match the DN of the search*
- » *not good !*

Making faster and scalier: some details on how our current database

- Samba 4 use ldb for storing the data, it's a tdb database behind the scene
- Key is the casefolded version of the DN
- Value is complete LDAP record, you have to unpack the whole record to easily access a field
- Index are also key/value, where the value is actually the DN of the record that match the entry

So:

- For every indexed search, we search the index(es), get the DN, fetch the objects that match the index, finish the filtering for non indexed attributes

- Each DN of the index has to be casefolded in order to be able to search for the associated record
- For each record we will casefold the DN in order to see if match the DN and the scope of the search
- Unpacking the object means a lot of talloc calls (between 50 and 100 per object)

Making it faster and scalier: Things in progress

Code is not yet in master but at my git tree in the branch ldb_perfs

- Casefolding can represent 50% of the time spent in the search
    - *Let's store the DN in the index already case folded, better pay the price for case folding only once (when stored) than million time (when read)*
    - *Let's use the DN of the index for cheching if the record match the DN and the scope of the search, so that we don't recasefold*
- Freeing talloced object can at the end be expensive, allocating and freeing object that you don't need is a important waste of resource and can account for 20/30% of the time of the search
    - *Check if the object match the DN and the scope of the search before loading the object, by using either DN from the index*

    - *entry or the key (which is a DN) passed the search_func. So that we don't load objects that we don't need*
    - *Provide a list of attribute we need to the unpack function so that we skip the attributes we don't need*
- Remove double loops in the operational module, so that we speedup the processing of each object returned by the search

Making it faster and scalier: current results

More detail in the thread ldb + samdb perfs and ideas for perf improvement

- » Search on baseDN "*OU=SmallOU,OU=MyCompany,DC=test,DC=corp*" with criteria *(objectclass=user)*, 600/700% improvement
- » Search on baseDN "*DC=test,DC=corp*" with criteria *(objectclass=user)* and just the samaccountname as requested attribute, 500%/700% improvement
- » Search on baseDN "*DC=test.corp,CN=MicrosoftDNS,CN=System,DC=test,DC=corp*" with criteria *(dc=server1)* and attribute dnsZone, 100/130% improvement

- » 20% speed improvments in make test for AD DC related tests

What about the smaller, from the title ?

Future development

- Range indexing, ie being able to index things like (uSNChanged<=3456)
- Storing GUID instead of DN as index entries
  - *Will reduce the size of the database as a GUID is 16 bytes and most DN are much more larger*
- Storing NT security descriptor in a separate database, and just store the reference to the SD in the object record
  - *SD are very large and object of the same class have often the same SD, chances are that with 40K users you have 40K identical SD, storing one copy instead + 40K references will be much smaller than storing 40K SD*

- *Should speed up the verification of NT ACLs, as once the verification has been done for a given SD then we can keep the result of identical SD*

What about OpenLDAP ?

Disclaimer: I have a bias

- OpenLDAP has a good reputation of being able to deliver high performance LDAP
- Speed gains from switching tdb to openldap as the backend need still to be proved

My queries are slow, what can I do ?

- Use

```
LDB_WARN_UNINDEXED=y ldbsearch -H PATH_TO_LDB criteria
```

- If suspicious request come through LDAP and you are not able to identify them, restart samba with *LDB_WARN_UNINDEXED* exported
- Use callgrind (perf ?) when you have a precise scenario, be ready to share the trace with the team (but I don't know if some data are leaked in traces)
- Help us to get script to get automatically a realisitic database representing your usecase so that we can reproduce it

Questions

Questions ?

Thank you

Thanks for listenning

A copy of the slides will be at www.matws.net/mat/pres/xp2013