SerNet

Samba 3.2 Infrastructure

SambaXP, Göttingen

April 18th, 2008

Volker Lendecke

SerNet

Samba Team

Network Service in a Service Network

SerNet

Volker Lendecke

- Co-founder SerNet - Service Network GmbH
  - Free Software as a successful business model
  - Network Security for the industry and the public sector
  - Samba-Support/Development in Germany
- For almost 20 years concerned with Free Software
- First patches to Samba in 1994
- Consultant for industry in IT questions
- Co-founder emlix GmbH (Embedded Systems)

Network Service in a Service Network

SerNet

Samba 3.0 -> 3.2

- git diff –stat origin/v3-0-test.. shows 429313 insertions and 195937 deletions
- ***Something*** has happened
  - Inbuf/Outbuf is gone
  - Infrastructure to migrate rpc server to PIDL
  - Cluster patches (messaging, dbwrap API)
  - In-memory cache
  - talloc_tos(), Talloc pools

Network Service in a Service Network

Inbuf / Outbuf

- Basic SMB request API in 3.0:
  - **`reply_close(conn, inbuf, outbuf, size, bufsize)`**
  - Inbuf, outbuf assumed to be 128k + safety margin
- **reply_close is expected to return the reply packet size, -1 for deferred requests**
- Inbuf and outbuf are allocated once and for all, for async requests they are copied somewhere.
- Inbuf/outbuf statically allocated is good for cache locality, but wastes >256k for each smbd

SMB request API in 3.2

- `void reply_close(struct smb_request *req)`
- ```
  struct smb_request {
            const uint8 *inbuf;
            uint8 *outbuf;
            connection_struct *conn;
            ... header fields ... };
  ```
- Inbuf allocated by process.c, the individual request must allocate outbuf itself.
- inbuf/outbuf carry their size themselves
- smb_request is a potential talloc parent for per-request data
  ```

Migrate RPC Server to PIDL

- At the SNIA conf in 2006 some RPC servers were converted to PIDL
- For smaller pipes, this is ok.
  - For the „big three" LSA, SAMR and NETLOGON this won't work
- In October 2006 lsa_Close() was converted individually (and wrongly, but this bug was not detected until 2008)
- Günther went in and did the step by step conversion

Clustered Samba

- Samba4 started out because Samba3 was not seen to ever be clusterable.
- Very first step: Rewrite oplocks to get rid of receive_next_smb()... (who remembers that now? :-)
- Smbd was structured quite well to become clustered
  - Separate processes
  - Communication only via messaging.c and tdb.

Clustered Samba

- Messaging was rewritten to allow multiple backends
- `struct db_context *db_open(filename, ...);`

```
struct db_context {
        struct db_record (*fetch_locked)(.. key ..);
        int (*traverse)(.. callback ..); };

struct db_record {
        TDB_DATA key, value;
        NTSTATUS (*store)(rec, data, flag);
        NTSTATUS (*delete)(rec); };
```

- Pretty much all tdbs rewritten to this fake C++
- db_open_ctdb calls out to a cluster db

Internal caches

- Tons of silly little ad-hoc caches
  - statcache.c (neither silly nor little, but a cache)
  - Sid2uid & friends
  - Getcwd, getpwnam
- Most of these little caches are done via static variables holding exactly one entry
- No cache expiry
- Gencache is there, but that's a global on-disk tdb

Memcache.h

- 3.2 has a caching data structure
- ```
  enum memcache_type { STAT_CACHE, UID_SID_CACHE,
                       GID_SID_CACHE, ...};
  struct memcache *memcache_init(mem_ctx, max_size);
  void memcache_add(cache, cache_number, key, val);
  bool memcache_lookup(cache, cache_number, key, *val);
  ```
- Memcache does not grow beyond max_size
- LRU cache cleaning
- Internally linux-rbtree based
- No home-grown in-memory tdb „malloc“
  ```

talloc_stackframe()

- sid_string_static() needs to go
  - Static variables are bad, sid_string_static() can't display two SIDs in a single debug statement
  - But - sid_string_static() is just too handy
- Alternative: sid_string_talloc()
  - This means we need to pass a talloc context around
  - Temporary talloc contexts need to be freed

talloc_stackframe()

- 3.2 has talloc_stackframe() and talloc_tos()
- talloc_stackframe() allocates a temporary talloc context on a global stack
  - The stackframe is a normal talloc context that must be freed with talloc_free()
  - Nested talloc_stackframe() calls are robust: If you forget to talloc_free the inner one, the outer talloc_free() of a frame takes care of it
- talloc_tos() refers to the current topmost frame
- Whenever a temporary talloc context is needed, use talloc_stackframe()

Talloc pools

- Speed tests with nbench showed that 3.2 is considerably slower than 3.0
- Topmost CPU user: malloc
  - Having no static buffers are nice, but you pay the price for malloc
  - OpenLDAP learnt this lesson for the multi-threaded case, we thought malloc would be cheap without threads
- Talloc provides a memory hierarchy

Talloc pools

- Normal smbd use pattern for talloc: Per smb request we do stuff, tallocing memory to just throw it away immediately
- Very little long-lived memory
- talloc_pool() creates a memory chunk
  - Talloc children from this talloc pool just increment a pointer, talloc_free() of these children does not really free memory
- At the beginning of every SMB request we do a talloc_pool(8192), include that as the lowest stackframe, and no later talloc calls from talloc_tos() does a system malloc

External named pipe handlers

- RPC server infrastructure is completely built into smbd
- Samba 4 has the „dcerpc remote“ backend: Proxy specific RPC interfaces to external servers, mainly used for debugging
- Samba 3 should also provide this functionality to become a DCE/RPC infrastructure component
- Two steps:
  - LANMAN pipes will be able to be handled by unix domain sockets
  - Similarly for the RPC interface level

SerNet

Questions/comments?

Volker Lendecke, VL@SerNet.DE

SerNet – Service Network GmbH
Bahnhofsallee 1b
37081 Göttingen

Tel: +49 551 370000 0
Fax: +49 551 370000 9
http://www.SerNet.DE

http://Samba.SerNet.DE

Network Service in a Service Network