SerNet

# Ctdb performance

1. März 2013

Volker Lendecke

SerNet GmbH, Göttingen - Berlin

# SerNet GmbH

SerNet

- gegründet 1997
- Büros in Göttingen und Berlin
- Themen: Informationssicherheit und Datenschutz
- spezialisiert auf Open Source Software
- verinice.: Open Source ISMS Tool
- SAMBA: Windows/Linux-Interoperabilität, Clustering und Private Clouds
- Zertifizierungen und Audits, IT-Grundschutz und ISO 27001
- Firewalls und VPN-Lösungen für mittlere und große Einrichtungen
- Old Economy: kein Risiko-Kapital, keine Bank-Kredite
- über 1500 Bestandskunden in DE, EU, US

# SMB: Server Message Block

SerNet

- historisch gewachsenes Fileserverprotokoll
- Ursprünge basierend auf MS-DOS Systemaufrufen
- Semantik lässt sich auf Single-Tasking DOS zurückführen
- "Laufwerk D: übers Netz

# SMB 1, 2, 3

SerNet

- SMB1: bis Windows Vista einziges Protokoll
- Evolutionäre Weiterentwicklung von DOS bis Windows 2003:
  - Anpassung an NTFS-Semantik
  - Unicode-Dateinamen
- Hunderte von Server und Client Implementationen
- SMB2 mit Vista: Neue Implementationen
- SMB3 (Windows 8 / 2012): Wesentliche neue Features
  - Skalierbarkeit, Hochverfügbarkeit

# Dateien Öffnen in Posix

SerNet

- Prüfung auf dem Dateipfad:
  - Existenz der Verzeichnisse
  - x-Bit auf allen Verzeichnissen vorhanden
- Datei vorhanden?
  - Berechtigungen ausreichend?
  - -> Datei geöffnet
- Für existierende Dateien reine Disk-Lese-Operationen notwendig
- Jeder Prozess oder Clusterknoten kann das lokal tun
- Kein anderer Prozess muss informiert werden

# Dateien Öffnen in SMB

SerNet

- Ähnliche Operationen wie bei Posix
- Pfadprüfungen, Berechtigungen
- Aber: Share Modes
- Siehe Windows CreateFile API, Parameter dwShareModes
  - If this parameter is zero and CreateFile succeeds, the file or device cannot be shared and cannot be opened again until the handle to the file or device is closed.

# Samba Architektur

SerNet

- Single Threaded, multi-Prozess File Server "smbd"
- Threads für async pread/pwrite
- Pro Client eine TCP-Verbindung
- Pro TCP-Verbindung ein Prozess
- Viele Benutzer und Freigaben über eine TCP-Verbindung

# Share Modes

SerNet

- Jeder Samba-Prozess weiss von allen offenen Dateien
- Shared Memory
- Trivial Database tdb
- Multi-Writer Key/Value
- Share Modes indiziert durch Device/Inode

# Tdb

SerNet

- "Trivial Database": Erste Implementation unter 1000 Zeilen
- Shared Memory durch mmap(2), Fallback mit pread/pwrite
- Hashtabelle
  - Lineare Listen (Hashketten) zur Kollisionsauflösung
- Speicherverwaltung durch einfach verkettete Freelist
- Listen sind durch fcntl-Locks geschützt
- Transaktionen: Änderungen durch msync/fsync geschützt

# Offene Dateihandles

SerNet

- Share Modes in locking.tdb
- Jede Datei ein Datensatz
- Jedes offene Handle ein Eintrag im Datensatz
- CreateFile prüft Konflikte anhand von dwShareModes
  - ggf NT_STATUS_SHARING_VIOLATION

# smbd Nachrichten

SerNet

- Oplocks: Clients können gegenseitig Sperren brechen
  - „Ab jetzt bitte nicht mehr cachen“
- SMB kennt blockierende Locks
  - Request kann auf ein Lock warten
  - Unlock muss Wartende informieren
- smbd's tauschen Nachrichten aus
- Nachrichten über messages.tdb
  - Ein Datensatz pro smbd
  - Signal USR1 für asynchrone Information

# Clustered Samba

SerNet

- Aufbau auf Posix Cluster-Dateisystem
  - GFS, OCFS, GPFS, Gluster, etc
  - SMB Semantik in Samba implementiert
- Multi-Prozess Modell hilft
  - Prozessgrenzen klar definiert
  - tdb und Messaging
- smbd-Datenmodell unverändert
- Multi-Threaded Server sind hier im Nachteil

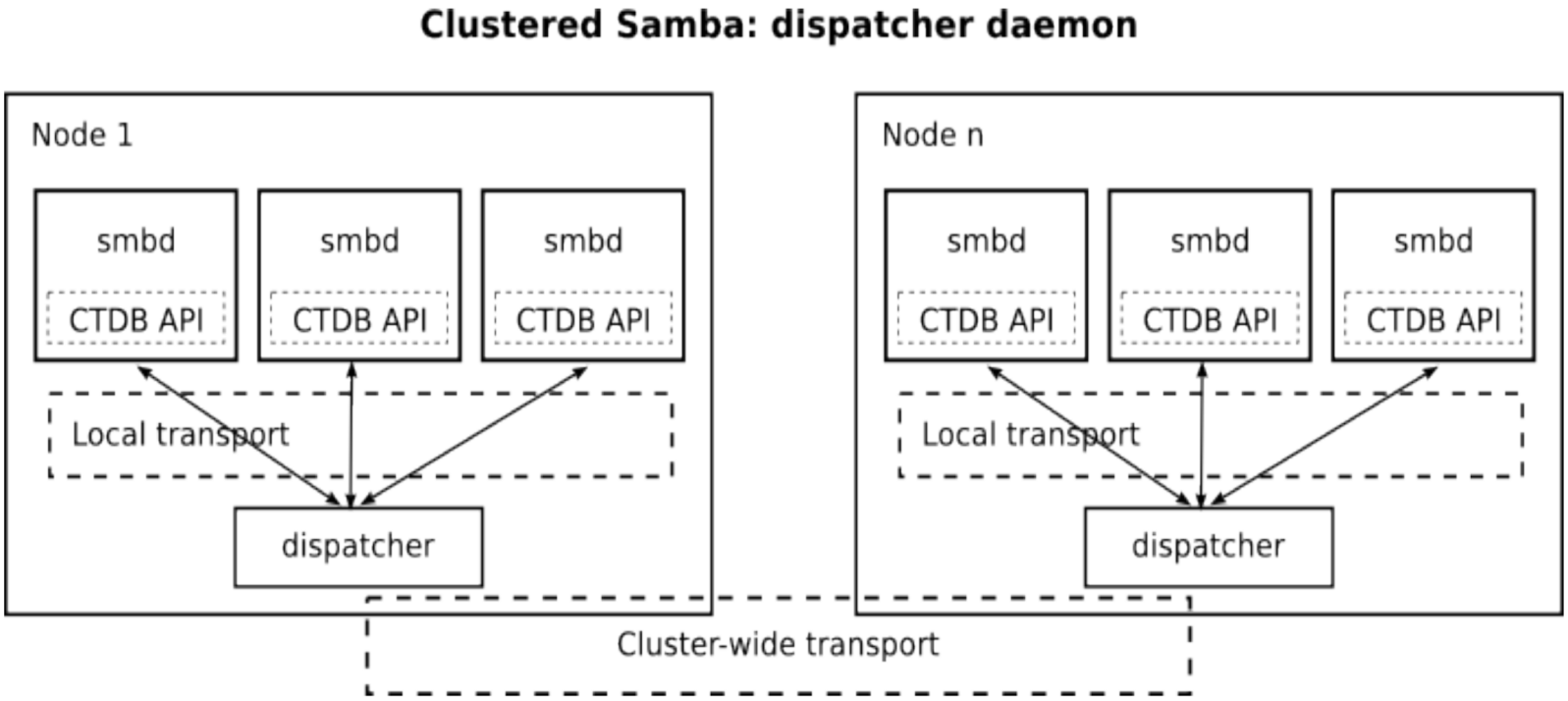
# ctdb (Clustered Tdb)

SerNet

- Pro Clusterknoten ein ctdbd
- ctdbd sorgt für "lokale" tdb-Semantik
- Datensätze werden zwischen Knoten ausgetauscht
  - L(ocation)MASTER wird per hash berechnet, kennt den
  - D(ata)MASTER
  - DMASTER hält den eigentlichen Datensatz
  - Jede Migration erhöht eine Sequenznummer
- smbd schaut lokal in locking.tdb
  - Wenn nicht DMASTER, wird ctdbd gefragt

# ctdb Architektur

SerNet



# ctdb Recovery

SerNet

- Knoten fällt aus
- Alle Datenbanken werden aufgeräumt
- Recovery-Dämon lässt alle Datenbanken einfrieren
  - Allrecord Lock überall
- Alle Datensätze werden eingesammelt
- Last Writer Wins (höchste Record Sequence Number)
- Alle Datensätze werden an alle verteilt
- Allrecord Lock wird aufgegeben

# fcntl Byte Range Locks

SerNet

- ```
  short l_type  Type of lock; F_RDLCK, F_WRLCK, F_UNLCK.
  off_t l_start Relative offset in bytes.
  off_t l_len   Size; if 0 then until EOF.
  ```
- Beliebige Dateibereiche können reserviert werden
- Unabhängig von Dateigröße
- Nur Hinweischarakter (advisory)
  - read/write weiter möglich, reiner IPC-Mechanismus
- Automatisches Aufräumen bei Prozess-Exit
- (fast) Ideal für tdb
  ```

# fcntl Skalierbarkeit

SerNet

- Recovery auf einem belasteten Cluster
- ctdbd nimmt sich ein allrecord lock
  - fcntl lock über die gesamte Datenbank
- Während der Recovery warten tausende smbds
- Allrecord Unlock: tausende Prozesse werden aufgeweckt
- Bei 100.000 Hash Chains eigentlich kein Problem, oder?

# /usr/src/linux/fs/locks.c

SerNet

```
static DEFINE_SPINLOCK(file_lock_lock);
void lock_flocks(void) {
     spin_lock(&file_lock_lock);
}
```

- Alle fcntl-Operationen müssen durch ein einziges Spinlock
- Überbleibsel des Big Kernel Lock
- Deadlock Detection:
  - fcntl(F_SETLKW) kann EDEADLK
- Prüfung über Dateien hinweg
- Tausende Prozesse, die auf 32 Kernen auf ein Spinlock warten
  - Thundering Herd

# pthread_mutex_t

SerNet

- Basis-Lock für Threads
- glibc unter Linux: Atomare CPU-Operationen
  - Nur im Konfliktfall (Mutex blockiert) ein Systemaufruf
- Koordination innerhalb eines Prozesses
- pthread_mutexattr_setpshared:
  - Mutexe über Prozessgrenzen hinweg

# Robuste Mutexes

SerNet

- Mutexes müssen normalerweise nicht "aufräumen"
  - Wenn ein Thread chrasht, ist der Prozess weg
- pthread_mutexattr_setrobust:
  - Automatisches Aufräumen (EOWNERDEAD)
  - "Modernes" Linux und OpenSolaris
- Linux nicht 100% Robust, aber vermutlich "gut genug"
  - Prozesse pflegen „robust list“ selbst
- Kein globales spinlock

# tdb mit Mutexes

SerNet

- fcntl locks können Dateibereiche Sperren
- Mutexes repräsentieren nur 1 Bit (lock/unlock)
  - Perfekt für Hash Chain locks
- Allrecord Lock nicht so einfach
  - Im wesentlichen ein pthread_rwlock_t
- Leider kein pthread_rwlockattr_setrobust

# ctdb Architektur

SerNet

- Ein Single Threaded Prozess pro Knoten
- 4-8 Kerne mit smbds können einen ctdbd gut auslasten
- Clusterknoten heute mit 32 oder mehr Kernen
- Records sind im Normalbetrieb komplett unabhängig
  - Theoretisch pro record ein LMASTER/DMASTER Prozess

# Lokales Messaging

SerNet

- Intra-Knoten Messaging im Moment über eine tdb
  - fcntl :-(
- Messaging zu ctdbd per unix domain stream socket
- Messaging zwischen Knoten: TCP, Infiniband möglich
- Umstellung auf Unix Domain Datagram Sockets
  - Verlässlich
  - skalierbar (/dev/log ?)
  - größenbeschränkt

# Messaging zwischen Knoten

SerNet

- Nicht alles über einen "broker"
- Viele lokale Absender (tausende smbds)
  - Pro Absender eine TCP-Verbindung ist zu viel
  - Proxy-Prozesse für Inter-Knoten Messaging
- LMASTER/DMASTER
  - LMASTER/DMASTER an Knoten gebunden
    - LMASTER als "Server" im messaging-System
    - Registrierung als spezielles messaging-Target
  - beliebig viele LMASTER pro Knoten

## glibc Speicherfragmentierung:

SerNet

- kurzfristig viel Speicher, viele kleine Objekte
- glibc malloc gibt die nicht mehr zurück
  - malloc kann keine Objekte verschieben
  - Allokationen über 128KB direkt als mmap
- interne tdb
  - >128KB am Stück
- tdb_repack möglich, tdb gibt keine Pointer heraus

# Copy on Write

SerNet

- Elternprozess mit 1GB
- fork für kleine Aufgaben
  - fcntl lock ist blockierend, nicht asynchron
  - Während Recovery davon hunderte
- CoW macht das zunächst billig
  - Elternprozess schreibt. Hunderte von Kopien
- vfork/exec wirft pages "billig" weg

# Lineare Listen

SerNet

- Angemessen, wenn wenige Elemente
- Können sehr teuer werden
- Messen, Messen, Messen
- Interne tdb für populäre Datenstrukturen
  - Hashtabelle
  - Red-Black Tree

# Kontakt

SerNet


## Volker Lendecke, VL@sernet.de

**SerNet GmbH**

**Bahnhofsallee 1b**
**37081 Göttingen**

**Schützenstr. 18**
**10117 Berlin**

**tel +49 551 370000-0**
**fax +49 551 370000-9**

**+49 30 5 779 779 0**
**+49 30 5 779 779 9**

**http://www.sernet.de**