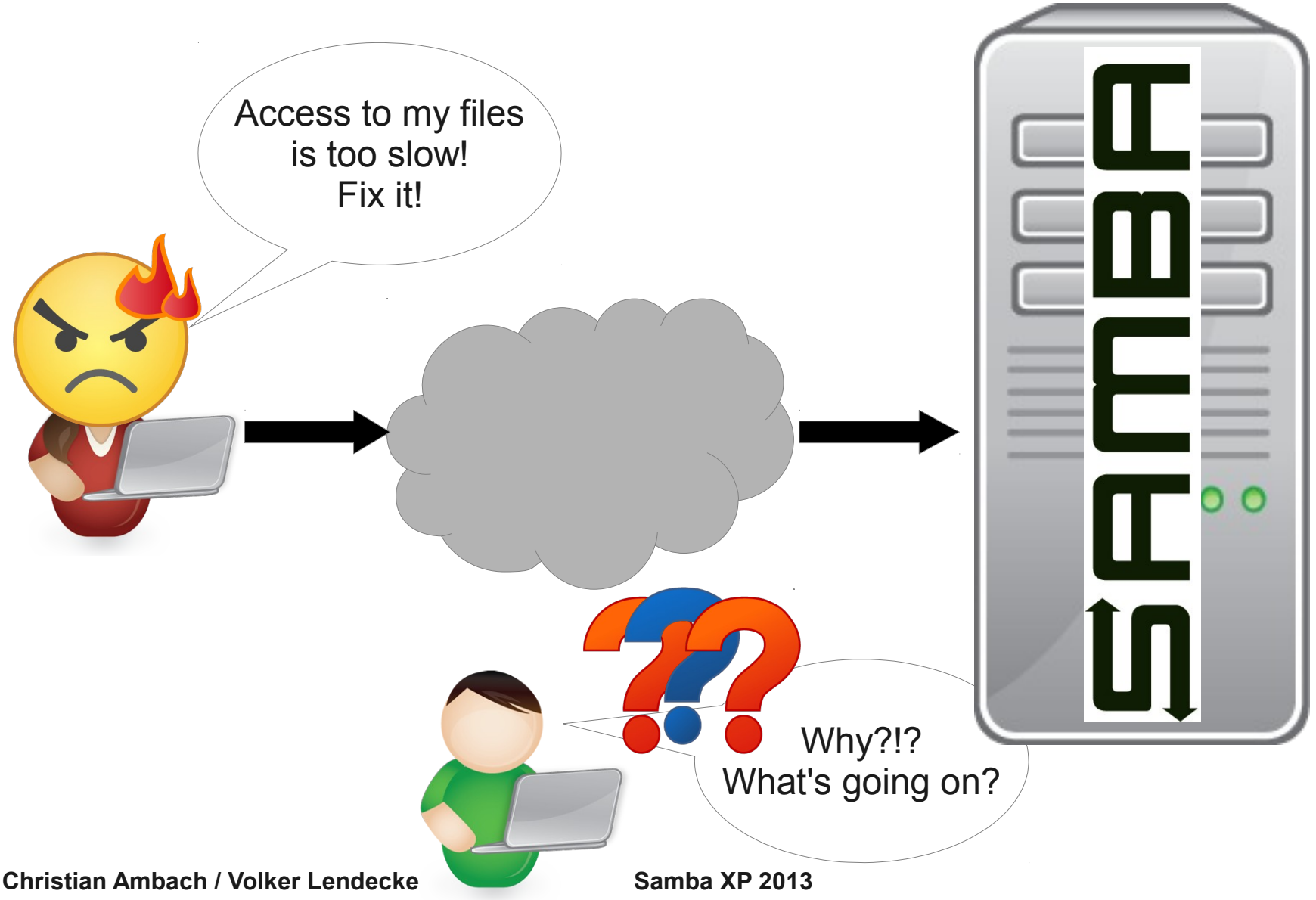
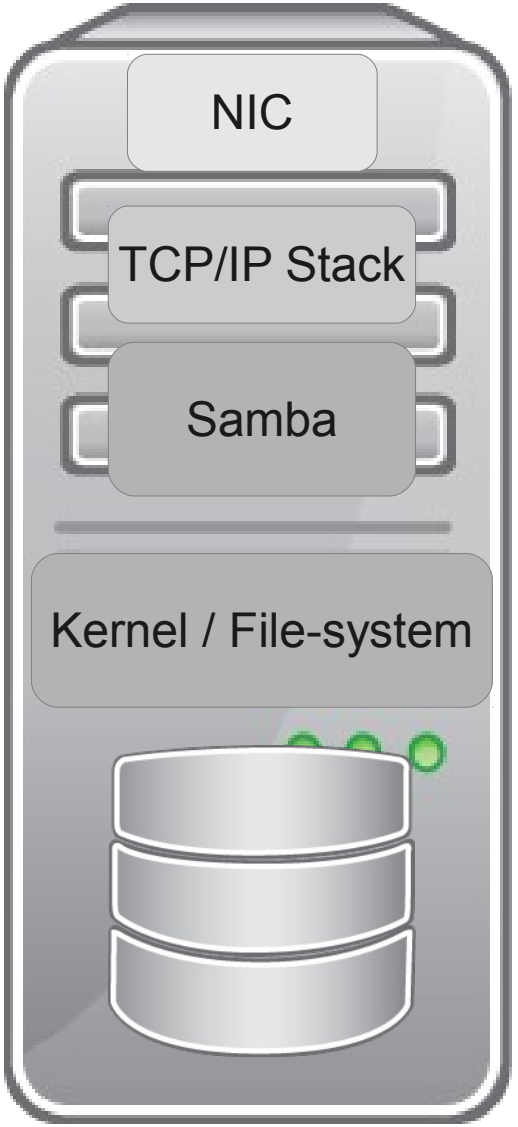
Samba Performance Tuning

Christian Ambach, IBM / Samba Team
Volker Lendecke, SerNet / Samba Team

The administrator's dilemma



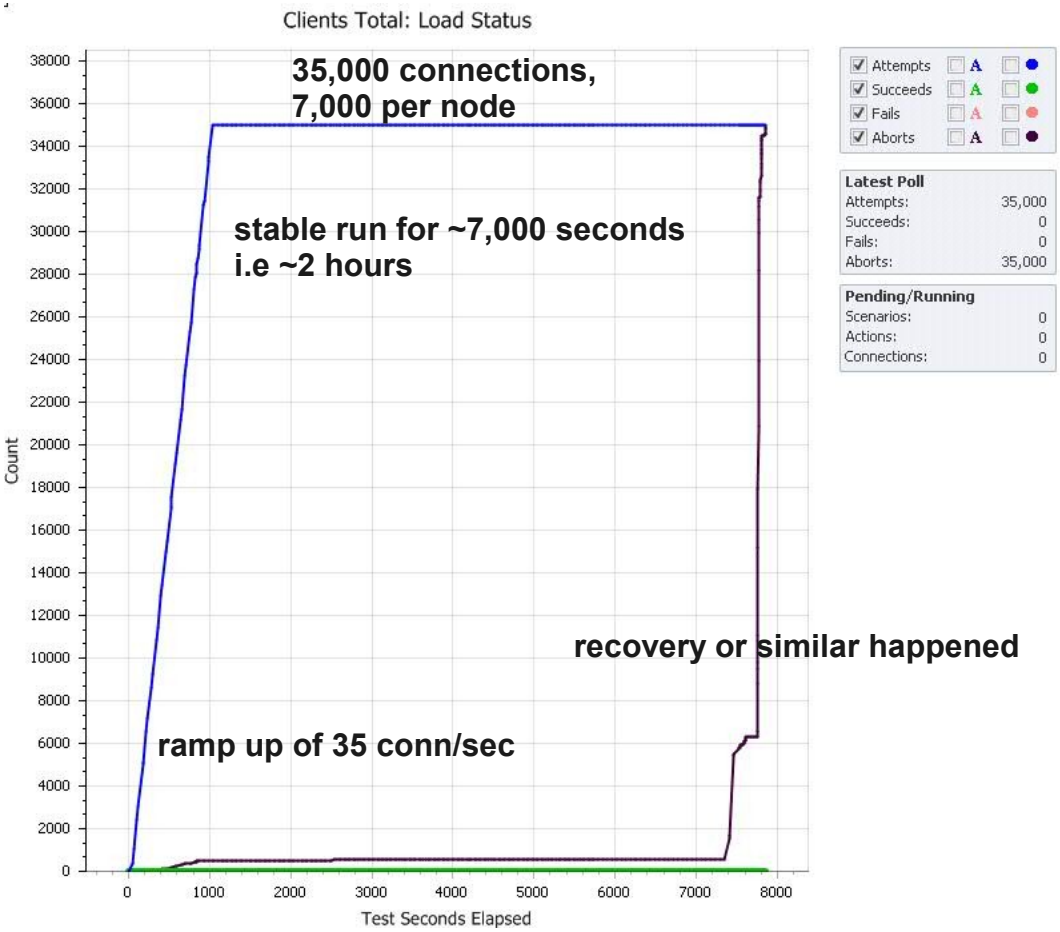
Key components of a file-server



- Bottlenecks can be located in any of the components
- Samba cannot monitor the layers above of it
- Samba can and should monitor all parts below of it and more important: itself

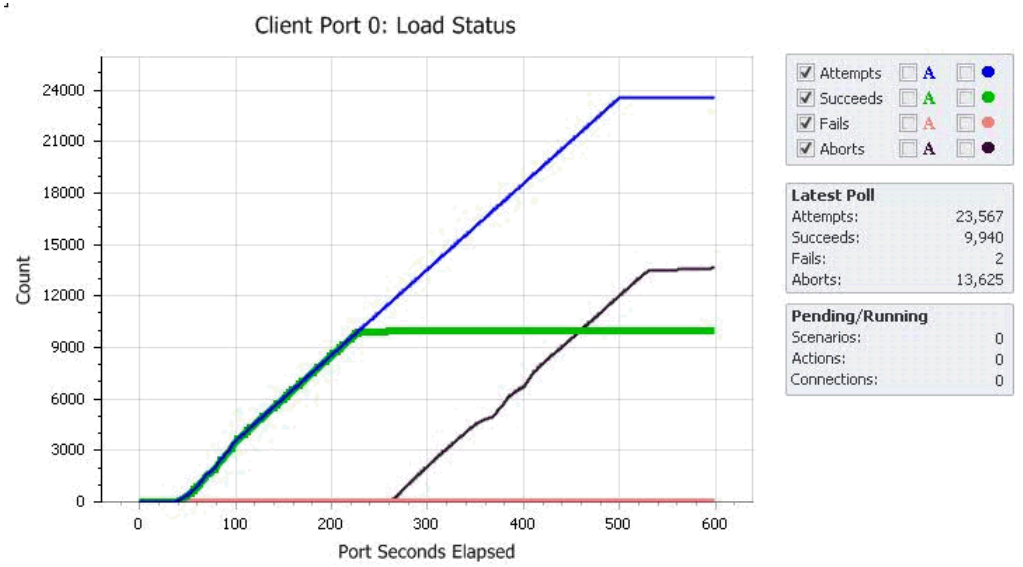
Large Scale Testing

- simulation of a scale-out home directory workload
  - 35K concurrent users with light IO workload each
  - ramp up of 35 users per second
  - expected stable run after complete ramp up of several hours
  - including administrative workloads (backup, snapshots, etc.)
- Improvements made
  - speed up of winbindd to support ramp-up
  - ctdb low level optimizations to lower ctdb load (networking, events)
  - exchange fcntl locks by robust mutexes to resolve kernel bottleneck

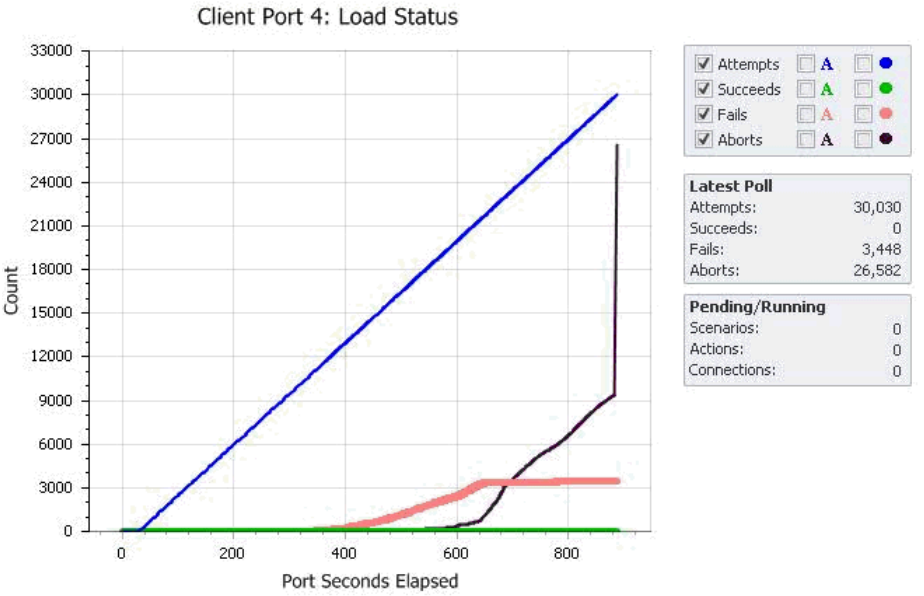


Where we started

- initial testing
  - ramp up only up to ~10K users
  - no stable phase at all



- some weeks later
  - issues start during ramp up at ~12K users
  - breakdown at ~30K users



Performance inhibitors

- SMB/CIFS: strict request/response pattern
  - Each delay in serving a request immediately affects performance
- Delays can be located in Samba itself
- Delays can also be located outside Samba
  - Saturated network
  - Saturated disks
  - Saturated system resources
    - RAM, CPU
  - Missing scalability of the kernel in IPC methods
    - e.g. fcntl
  - Choice of file-system is also important
    - Advanced file-system functions like snapshots might cause additional IO on the disks (and increased CPU usage of the filesystem itself)

Potential causes of delay inside Samba stack

- Serving request takes too long
- Wait for IPC resource
  - TDB lock
  - CTDB
- Currently looking at messages like this

```
smbd[3494971]:   db_ctdb_fetch_locked for /var/ctdb/brlock.tdb.3 key
FF5BD7CB3EE3822E024B1725000000000000000000000000 needed 1 attempts,
16558 milliseconds
```

Authentication scalability

- Each NTLM authentication needs a round-trip to domain controller
- A Samba server can only reach the authentication rate that the DC allows
- Establish multiple connections to allow more parallelism

Identifying bottlenecks

- Sometimes hard to tell why smbd took so long to send response
- Tools like perf only show functions that consume lots of CPU, but not wall time
- But wall time is what matters for throughput

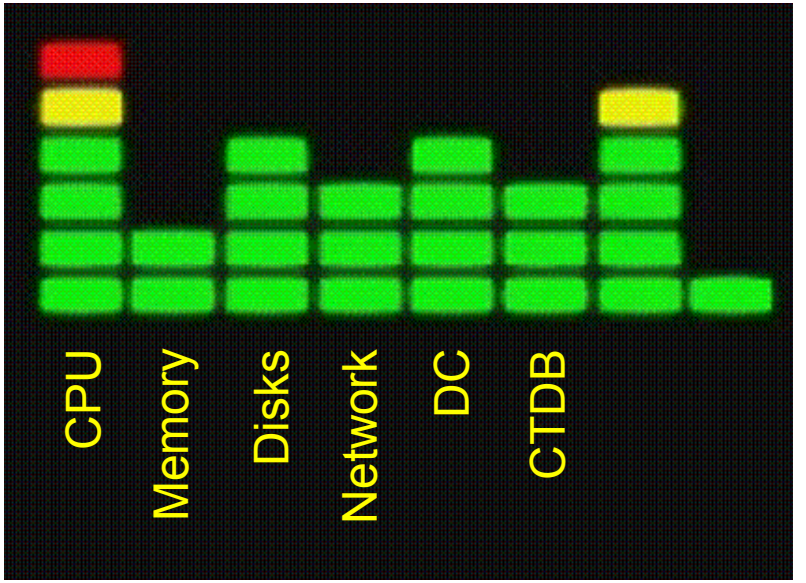
Requirements

- Capture data that allows to identify why performance is bad
- Use a single tool instead of a whole collection
- Allow collection of overall statistics and also statistics per client
- Counters should allow to infer characteristics of workload
- Numbers should not only assist Samba developers but also normal admins

Vision

- What we have
  - dstat, vmstat, iostat
  - atop
  - perf, strace
  - tcpdump, Wireshark
  - ...
- The 1mio € question:
  - Where exactly do we spend our time, per request please

- What we need



sizes:	<=512	<=1KB	<=2KB	<=4KB	<=8KB	<=16KB
read:	6084	1992	5382	76934	4563	2225
write:	7908	117	1638	9011	1175	1873

strace

- strace -ttT shows time spent in syscalls
- Large overhead, slows things down
- Best tool to point at kernel and file systems
- Sometimes it shows that Samba is doing silly things...
  - Case insensitive file name lookup
  - gpfs_getrealfilename

perf

- Great tool to identify hot code paths in both user- and kernel space
- Not suitable to detect delays where code waits on external resource and wall time keeps ticking

From samba mailinglist:

```
53.07%  [kernel][k] hypercall_page
36.33%  smbd    [.] SHA256_Update
```

=> High CPU load caused by SMB2 signing

Ltt-ng / IBM lite timers

- Have not looked at it yet in detail, but looking at its description it seems promising
- Needs instrumentation of the code
  - Can be done using the existing profiling macros
- IBM proprietary library that measures wall time spent in functions

IBM lite timers

- Examples taken while running SpecSFS 2008:

```
reply_ntcreate_and_X() 8228 165.776s 20.148ms 20.51%
├── vfswrap_stat() 8238 65.926s 8.003ms 8.16%
├── vfswrap_fstat() 8228 20.392ms 2.478us 0.00%
├── vfswrap_get_alloc_size() 65824 11.484ms 0.174us 0
├── vfswrap_kernel_flock() 8228 2.887ms 0.351us 0.00%
├── vfswrap_realpath() 8233 134.643ms 16.354us 0.02%
├── vfswrap_get_nt_acl() 8228 982.244ms 119.378us 0.1
│   └── vfswrap_stat() 8228 92.021ms 11.184us 0.01%
├── fswrap_fchmod_acl() 5 1.236us 0.247us 0.00%
├── get_file_infos() 8228 508.518ms 61.803us 0.06%
│   └── fetch_share_mode_unlocked() 8228 475.948ms 57
└── vfswrap_open() 8228 207.666ms 25.239us 0.03%
reply_nttrans() 1177 245.302ms 208.413us 0.03%
└── call_nt_transact_query_security_desc() 1177 204.4
    └── vfswrap_fget_nt_acl() 1177 89.402ms 75.957us
        └── vfswrap_fstat() 1177 20.534ms 17.446us 0.
reply_read_and_X() 16651 201.693s 12.113ms 24.95%
├── vfswrap_getlock() 16651 58.361ms 3.505us 0.01%
└── vfswrap_pread() 16651 201.096s 12.077ms 24.88%
```

```
call_trans2qfilepathinfo() 96905 821111030954 8473360 41.1%
├── vfswrap_stat() 174902 19552197989 111789 0.979%
├── vfswrap_fstat() 9807 110098356 11226 0.00551%
├── vfswrap_get_alloc_size() 96552 151432246 1568 0.00758%
├── vfswrap_realpath() 87451 3138479322 35888 0.157%
└── get_file_infos() 96552 771922758835 7994891 38.6%
    └── fetch_share_mode_unlocked() 96552 771566642958 7991203 38.6%
        ├── dbwrap_fetch_lock_db() 96552 414905563248 4297223 20.8%
        └── parse_share_modes() 96552 355461564600 3681555 17.8%
            ├── talloc() 96552 177664913 1840 0.00889%
            └── ndr_pull_struct_blob() 96552 312215451530 3233650 15.6%
```

vfs_time_audit

- monitors all calls in the Samba VFS
- If call takes longer than defined threshold it will output a warning message, including operation, file and time spent
- Good to detect sporadic hangs or overload situations
- As it intercepts all VFS calls, interpretation is sometimes difficult (e.g. create_file vs open)

vfs_iohist

- New module in Christian's performance wip branch
- Mostly finished, will present on samba-technical soon
- Only intercepts VFS calls that are “near” to the filesystem
  - open/close
  - stat
  - read/write/pread/pwrite
  - readdir
  - unlink
- Records information about
  - number of operations and time they took in buckets
  - Read/write sizes

vfs_iohist sample results

RAM disk

op	total	<=0.001ms	<=0.010ms	<=0.100ms	<=1.000ms	<=10.000ms	<=100.000ms	<=1000.000ms	<=10000.000ms	>10000.000ms
open	35351	0	23381	11962	7	1	0	0	0	0
close	35312	0	32820	2487	5	0	0	0	0	0
read	75229	29	54622	16245	4321	12	0	0	0	0
write	24114	0	8314	10208	5589	3	0	0	0	0
stat	644635	104483	505630	34467	53	2	0	0	0	0
readdir	692679	604345	78642	9679	13	0	0	0	0	0
unlink	9868	0	4619	5128	121	0	0	0	0	0
sizes:	<=512	<=1KB	<=2KB	<=4KB	<=8KB	<=16KB	<=32KB	<=64KB	<=128KB	<=256KB	<=512KB	<=1MB
read:	4056	1328	3588	51428	3042	1483	234	10070	0	0	0	0
write:	5309	78	1092	6040	785	1258	943	8609	0	0	0	0

USB3 stick with vfat

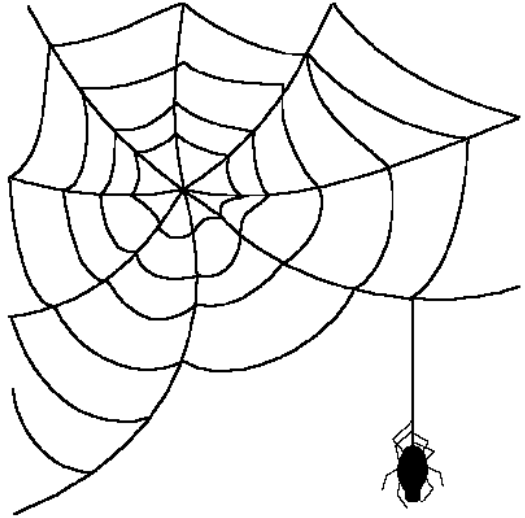
op	total	<=0.001ms	<=0.010ms	<=0.100ms	<=1.000ms	<=10.000ms	<=100.000ms	<=1000.000ms	<=10000.000ms	>10000.000ms
open	52779	0	23801	28943	35	0	0	0	0	0
close	52778	0	47493	5281	4	0	0	0	0	0
read	112624	996	85489	19786	6349	4	0	0	0	0
write	35852	0	7001	18478	10350	12	8	3	0	0
stat	961798	92892	702455	166368	82	1	0	0	0	0
readdir	1036021	891289	102175	42523	32	2	0	0	0	0
unlink	14480	0	3463	9832	1177	2	5	1	0	0
sizes:	<=512	<=1KB	<=2KB	<=4KB	<=8KB	<=16KB	<=32KB	<=64KB	<=128KB	<=256KB	<=512KB	<=1MB
read:	6084	1992	5382	76934	4563	2225	351	15093	0	0	0	0
write:	7908	117	1638	9011	1175	1873	1407	12723	0	0	0	0

Ten year old USB2 stick

op	total	<=0.001ms	<=0.010ms	<=0.100ms	<=1.000ms	<=10.000ms	<=100.000ms	<=1000.000ms	<=10000.000ms	>10000.000ms
open	24158	0	10498	12761	899	0	0	0	0	0
close	24136	0	21418	2716	2	0	0	0	0	0
read	51311	362	38871	8774	3303	1	0	0	0	0
write	16583	0	3007	8339	5208	14	7	6	2	0
stat	439310	38086	315553	85550	118	3	0	0	0	0
readdir	473256	409202	28712	35226	116	0	0	0	0	0
unlink	6596	0	1707	4271	590	2	12	14	0	0
sizes:	<=512	<=1KB	<=2KB	<=4KB	<=8KB	<=16KB	<=32KB	<=64KB	<=128KB	<=256KB	<=512KB	<=1MB
read:	2757	906	2448	35081	2067	1012	160	6880	0	0	0	0
write:	3649	54	752	4133	534	858	644	5959	0	0	0	0

perfcount modules

- Bitrot?
- Initially from Isilon
- Not SMB2 aware
- Only work on packets, not requests



smbd profiling

- Not suitable for modern architectures like NUMA
  - single shared memory segment that all processes work on
  - leads to cacheline thrashing and high interconnect usage
- No per-client statistics
- Pro: counters already spread all over the code at the relevant spots
  - Not necessarily in newer code
- Let's revive it!

```
START_PROFILE(SMBtcon);

if (req->buflen < 4) {
        reply_nterror(req, NT_STATUS_INVALID
        END_PROFILE(SMBtcon);
        return;
}
```

Proposed smbd profiling redesign

- Add size and time buckets
  - Similar to vfs_iohist
- Use TDB as shared memory, each process has its own record and uses mmap to directly write to it
  - The TDB can be put into /dev/shm
- Pros of this approach
  - allows to collect counters from single connection
  - By traversing database, summary can be created as well
- Review / fix existing counters
- Add new ones, e.g. for CTDB interaction
- TDB mutexes are prerequisite for this (they imply mmap)

Grandpa talks about the war...

- vfs_preopen
  - Media playout: 1 file per HD frame, 60 frames/sec
  - open(2) can take 20 milliseconds or more
  - vfs_preopen: fork helpers to open and read the next 10 files -> everything cached
- fcntl:
  - strace shows fcntl(F_UNLK) can take seconds – WTH is going on??
  - Thundering herd in the kernel on a single spinlock
  - Robust mutexes to the rescue

Questions?

Thank you!