Systems & Technology Group

Experiences with NAS protocols in clustered environments

Christian Ambach
Mathias Dietz

Agenda

- What is clustered NAS?
- Scale-out File Services
- Cross-Protocol interoperability
- Managing expectations
- Typical caveats
- Examples
- Future work

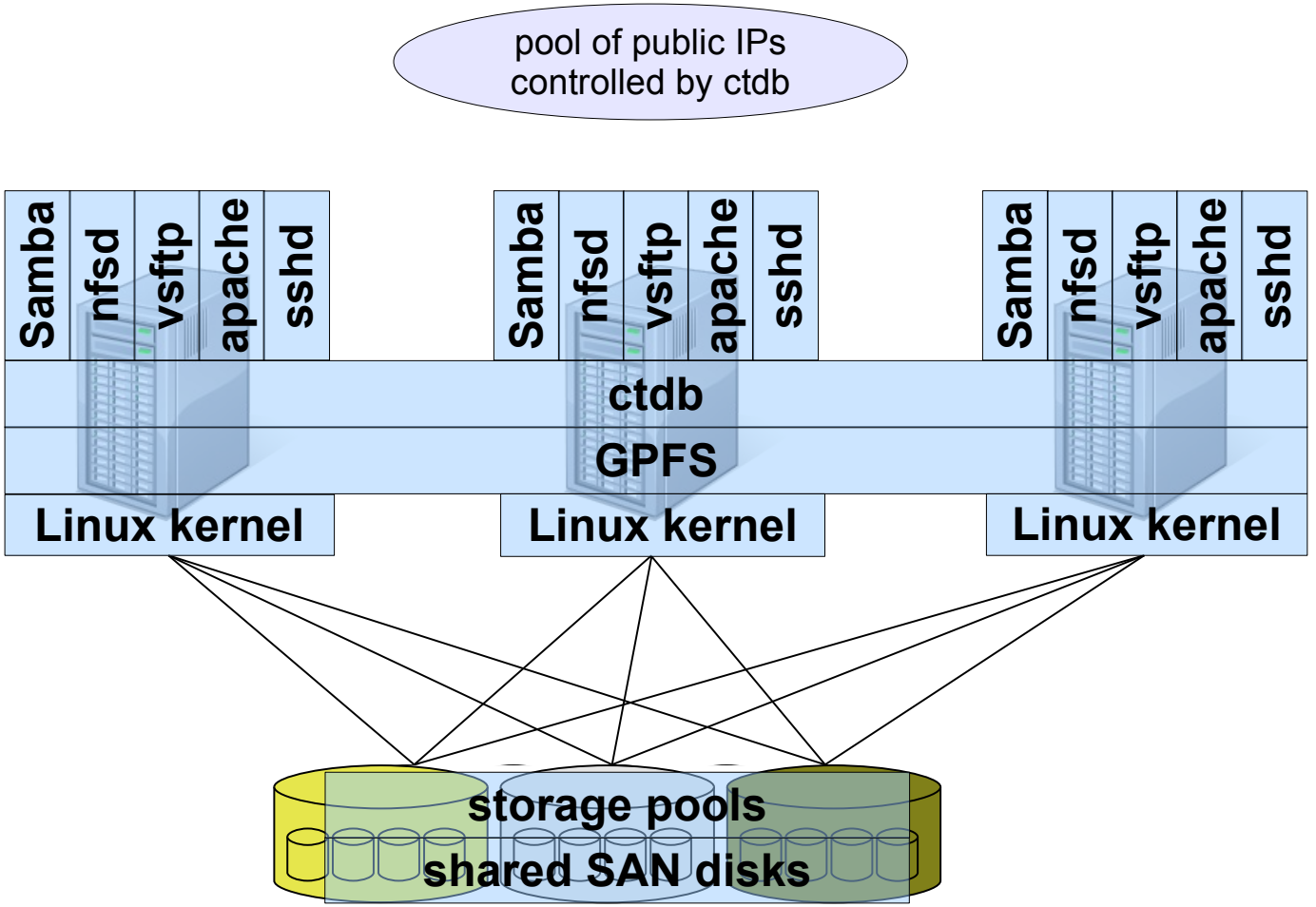
What is clustered NAS?

- **Serving the same data from multiple nodes**
  - Backed by clustered file system
  - Active-active configuration
- **Scalability: Add a node to**
  - handle more clients
  - get more throughput
- **Reliability**
  - When a node in the cluster fails, another takes over

Scale-out File Services (SoFS)

- **IBM's service offering for clustered NAS**
- **built on RHEL 5.3**
- **uses CTDB/Samba**
- **Supports a variety of NAS protocols:**
  - CIFS
  - NFS
  - FTP
  - HTTP
  - SCP

SoFS components



How it all began

- **2006**
  - Tridge and Volker sat together in an IBM lab in Mainz thinking about approaches how to make Samba perform well on top of GPFS
- **2007**
  - Development of CTDB started
  - First GPFS related Samba patches submitted
  - First release of SoFS
- **Since then**
  - Making it all work together even better :)

Cross-Protocol interoperability in SoFS

- **Most important: avoid data corruption**
  - Consolidate file locking across the protocols
  - In the POSIX world locks are advisory while they are mandatory on Windows
  - All locks will be set in GPFS as the central instance
  - Enabling posix locking in Samba will let Samba set locks with each operation in GPFS and not only in brlocks.tdb
  - NFS clients can use byte-range locks
  - FTP only supports BR locks while writing
  - HTTP/SCP clients cannot

Cross-Protocol interoperability in SoFS

- **Sharemodes**
  - Concept unknown to the POSIX world
  - Extended GPFS and Samba so sharemodes can be set in GPFS and are enforced
- **Oplocks/leases**
  - Added clustered lease support in GPFS
  - leases can now be broken anywhere in the cluster and Samba process holding oplock gets signal

Cross-Protocol interoperability in SoFS

- **ACLs**
  - GPFS supports NFSv4 ACLs
  - NFSv4 export of GPFS not yet supported on Linux
  - NFS clients will only see the mode bits but ACL is still enforced on the server
  - The same applies to FTP/HTTP/SCP clients

Wrong assumptions about CTDB clusters

- **A cluster gives more performance to my single client**
  - This is not true for single-stream operations!
- **Node fail-over is transparent to the client**
  - CTDB actively terminates TCP connections
  - sends Tickle-ACKs to speed up client reconnect
  - Clients need to reestablish TCP connection to a cluster node
  - Depending on the protocol it might be transparent (NFS can do it, CIFS cannot)

Prices to pay with clustered filesystems

- **Additional latencies**
  - Synchronization of metadata updates
  - Lock coherency (for cross-protocol interoperability)
- **Work best if data to metadata ratio is high**
  - Small file I/O involves too much cluster internal communication to make it really performing
  - Best throughput can be achieved transferring large files sequentially

GPFS specifics

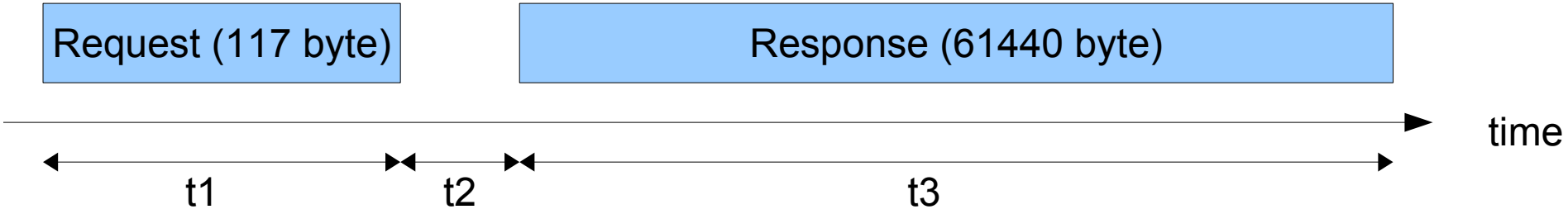
- **ACLs**
  - Stored in a single meta-data file
  - Multiple inodes can refer to the same ACL entry
  - Only one single writer to this file allowed
- **Extended attributes**
  - Single sparse file for all inodes
  - updates slow

Some comments on performance

- **Clusters usually cannot outperform a stand-alone fileserver for a single client because of the additional latencies introduced by internal communication**
- **But clusters will outperform stand-alone servers in aggregate throughput for many clients**
- **Scalability factor is good:**

1 node 109 Mbytes/sec
2 nodes 210 Mbytes/sec
3 nodes 278 Mbytes/sec
4 nodes 308 Mbytes/sec
(courtesy of Tridge)

Impact of additional latencies on throughput



- **Example with Read_AndX**
- **Ideal world: t2 = 0 ms**
- **Estimated throughputs:**

Network speed	latency (ms)	t1 (ms)	t2 (ms)	t3 (ms)	Σt (ms)	responses/s	throughput(MB/s)
1 Gb	0.001	0.0009	0.0000	0.4578	0.4606	2170.91	127.2
1 Gb	0.010	0.0009	0.0000	0.4578	0.4786	2089.27	122.42
1 Gb	0.100	0.0009	0.0000	0.4578	0.6586	1518.29	88.96
10 Gb	0.001	0.0001	0.0000	0.0458	0.0479	20892.73	1224.18
10 Gb	0.010	0.0001	0.0000	0.0458	0.0659	15182.91	889.62
10 Gb	0.100	0.0001	0.0000	0.0458	0.2459	4067.3	238.32

Impact of additional latencies on throughput (II)

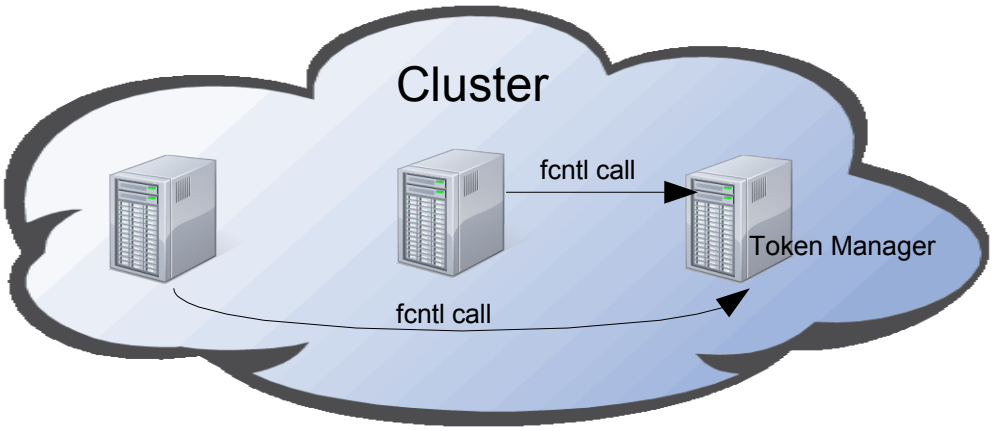
Now adding some clustering related latencies (t2)

Network speed	latency (ms)	t1 (ms)	t2 (ms)	t3 (ms)	Σt (ms)	responses/s	throughput(MB/s)
1 Gb	0.001	0.0009	1.0000	0.4578	1.4606	684.63	40.12
1 Gb	0.001	0.0009	2.0000	0.4578	2.4606	406.4	23.81
1 Gb	0.001	0.0009	3.0000	0.4578	3.4606	288.96	16.93
1 Gb	0.010	0.0009	1.0000	0.4578	1.4786	676.3	39.63
1 Gb	0.010	0.0009	2.0000	0.4578	2.4786	403.45	23.64
1 Gb	0.010	0.0009	3.0000	0.4578	3.4786	287.47	16.84
1 Gb	0.100	0.0009	1.0000	0.4578	1.6586	602.91	35.33
1 Gb	0.100	0.0009	2.0000	0.4578	2.6586	376.13	22.04
1 Gb	0.100	0.0009	3.0000	0.4578	3.6586	273.33	16.02
10 Gb	0.001	0.0001	1.0000	0.0458	1.0479	954.32	55.92
10 Gb	0.001	0.0001	2.0000	0.0458	2.0479	488.31	28.61
10 Gb	0.001	0.0001	3.0000	0.0458	3.0479	328.1	19.22
10 Gb	0.010	0.0001	1.0000	0.0458	1.0659	938.21	54.97
10 Gb	0.010	0.0001	2.0000	0.0458	2.0659	484.06	28.36
10 Gb	0.010	0.0001	3.0000	0.0458	3.0659	326.17	19.11
10 Gb	0.100	0.0001	1.0000	0.0458	1.2459	802.66	47.03
10 Gb	0.100	0.0001	2.0000	0.0458	2.2459	445.26	26.09
10 Gb	0.100	0.0001	3.0000	0.0458	3.2459	308.08	18.05

Latency example

Limited read performance for a single Windows client mainly caused by:

- **Samba does additional fcntl() calls for “Posix Locking” in front of each read or write**
- **In a clustered environment the lock manager (token manager) for a file might be another node.**
- **Due to the additional network latency, the single stream read performance was limited to 100MB/s on a 10 GbE network**
- **Read is more affected than write, because Windows will not send out subsequent read requests in parallel. Write requests are send out overlapped.**



Example in Wireshark

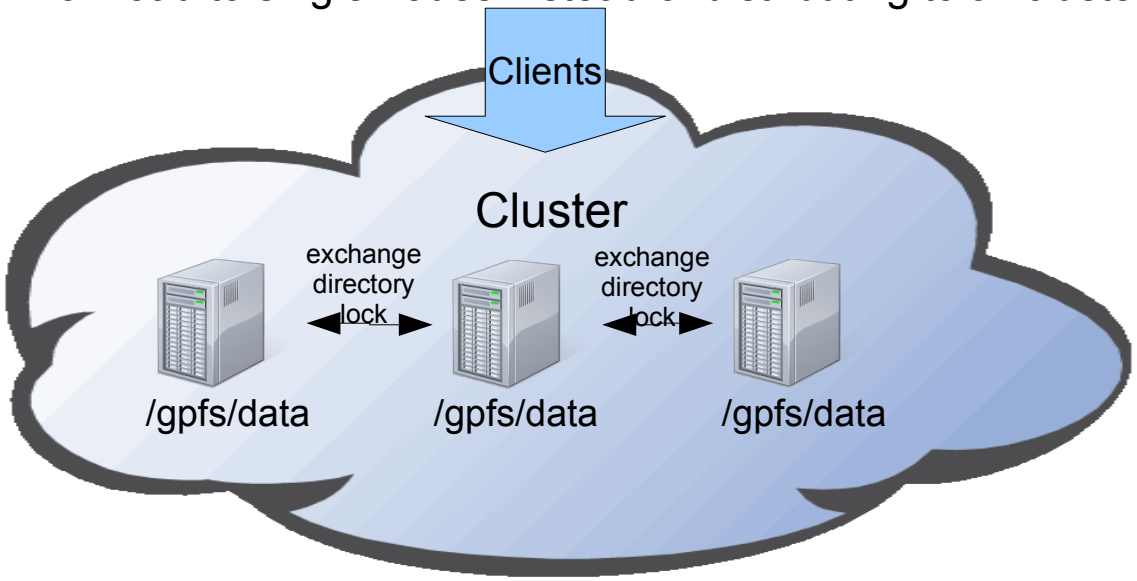
- **Two clients reading the same file over different nodes**

54311	3.766566	61.103	61.84	SMB	Read AndX Request, FID: 0x361c, 61440 bytes at offset 155258880
54312	3.766913	61.84	61.85	CTDB	REQ_CALL 0->1
54313	3.767003	61.85	61.84	TCP	ctdb > 51942 [ACK] Seq=1 Ack=35033 Win=4625 Len=0 TSV=1256701667 TSER=16755539
54314	3.767033	61.85	61.84	CTDB	REPLY_CALL 1->0
54315	3.767044	61.84	61.85	TCP	ctdb > 58628 [ACK] Seq=1 Ack=35713 Win=3688 Len=0 TSV=16755539 TSER=1256701667
54316	3.767229	61.84	61.85	TCP	gpfs > 59166 [PSH, ACK] Seq=109959 Ack=34909893 Win=5454 [TCP CHECKSUM INCORRECT] Len=152 TSV
54317	3.767319	61.85	61.84	TCP	59166 > gpfs [ACK] Seq=34909893 Ack=110111 Win=1946 Len=0 TSV=1256701667 TSER=16755537
54318	3.767327	61.85	61.84	TCP	59166 > gpfs [PSH, ACK] Seq=34909893 Ack=110111 Win=1946 Len=44 TSV=1256701667 TSER=16755537
54319	3.767468	61.84	61.103	SMB	Read AndX Response, 61440 bytes

0.761ms spent on cluster internal traffic

Difficult workloads

- **Two clients accessing the same files/directory over different cluster nodes**
- **For operations like create, unlink, readdir within the same directory the cluster's file system has to ensure the consistency with a “directory lock”**
- **A workload which mostly works with many files in the same directory will probably not scale with the cluster.**
- **GPFS 3.2 introduced “fine grained directory locks” (FGDL) to reduce the impact.**
- **Recommendations**
  - If possible change the application workload to work with many directories
  - Stick workload to single nodes instead of distributing to all cluster nodes.



CTDB related issues I

- **Especially on large clusters we experienced a high CTDB load due to locking requests.**
- **When the same files are accessed from many cluster nodes, ctdb has to ask the dmaster for given record.**
- **In our case, customer wrote into the same file from multiple clients.**
- **Sometimes it forced a dmaster transfer**
- **Many locking requests on the same file across the cluster lead to many dmaster requests/responses and put load on ctdb.**
- **Recommendations**
  - Keep enough CPU spare capacity
  - Use a low latency network (e.g. InfiniBand) for the internal cluster communication.

CTDB related issues II

Performance impacts due to ctdb vacuum process.

- **CTDB by default executes a ctdb vacuum every 5 minutes.**
- **CTDB vacuum will lock each record for a short period of time**
- **This can delay a client operation and cause performance drops.**
- **Recommendation**
  - Change schedule for ctdb vaccum to execute when the system is under light load (e.g. in the night)

Other limitations

- **File notify implementation of Samba does not scale across a cluster.**
- **Ask Volker and Ronnie for details :-)**
- **Recommendation**
  - Turn it off if possible

Level 2 Oplocks

- **We recently discovered an inconsistency in the Level 2 Oplocks implementation of Linux across multiple nodes**
- **Ask Volker for details**
- **Recommendation:**
  - Turn off Kernel Oplocks (breaking protocol interoperability)

Balancing features and performance

- **You do not need cross-protocol interoperability?**
  - Turn it off to get much more performance
- **Do you really need very complex ACLs?**
  - Keep it simple to get more speed
- **Adjust your workload to utilize the cluster optimally**

GPFS and Samba interoperability future

- **GPFS recently released a Windows client**
- **GPFS team had to add some Windows specific functionality we can now make use of in Samba**
- **Store Windowsattributes in inode**
  - Avoids mapping to POSIX bits
  - No need to use slow xattrs
- **Create timestamp**
  - GPFS stores it so Samba can make use of it
- **More to come :)**

Questions?