Clustered Samba performance analysis and improvements

Christian Ambach / Mathias Dietz - IBM Research & Development

Introduction

Christian Ambach and Mathias Dietz

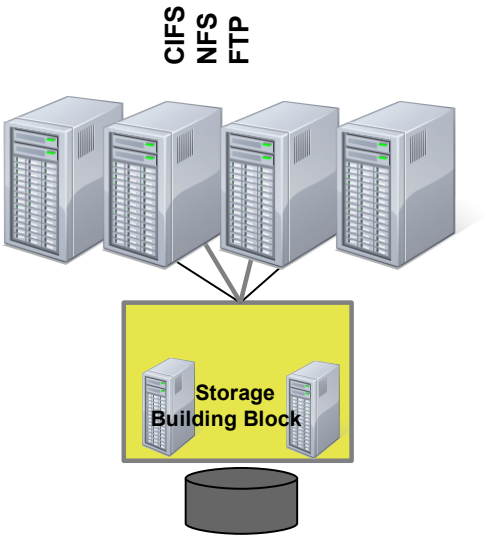
- Working for *IBM Research and Development* in Mainz, Germany on the IBM SONAS product since 2008.
- Have experiences with Samba as part of the IBM SoFS offering since 2006 and the IBM OESV offering since 2003.

IBM SONAS - Scale Out Network Attached Storage

Modular high performance storage with massive scalability and high availability.

Supports multiple petabytes of storage for organizations that need billions of files in a single file system.

Based on Open Source technologies (Samba, CTDB, Linux, ...)



Link: http://www.ibm.com/systems/storage/network/sonas/

Agenda

Presenting various customer cases that we worked on in 2009 and 2010.

Customers are mostly running digital-media applications and were facing issues with the performance of their CTDB cluster.

We will present

- the identified bottlenecks
- derived actions
- status of the fixes and improvements

Customer I

Visual effects company located in AP

Using Pixar rendering farm

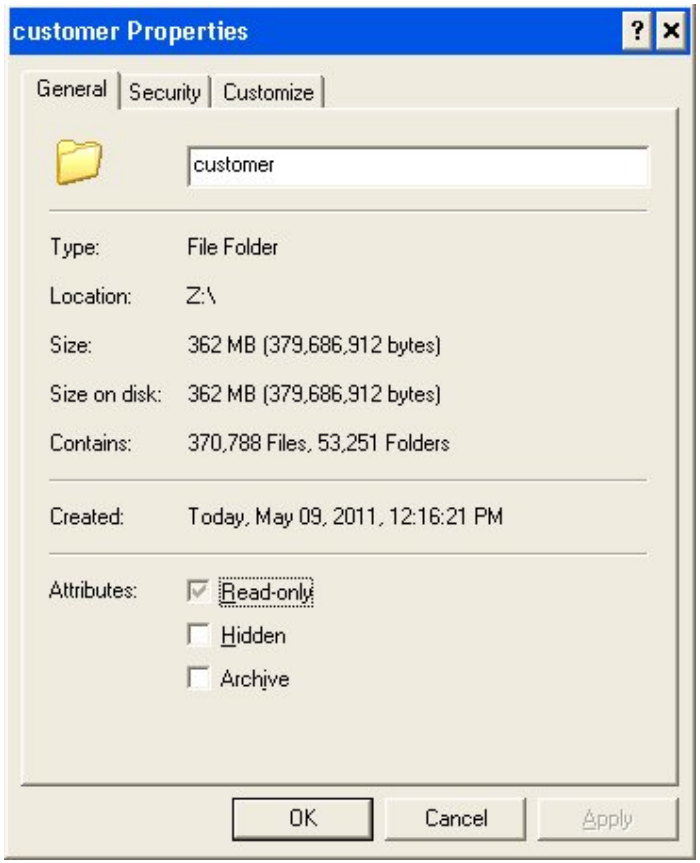
Workload: Application scans of large directory structure to see if a rendering job can be started

Problem:

- Customer used an AIX machine with CIFS server on it
- Directory scan finished in about 2 minutes
- New SONAS box took 5 minutes (two nodes, 120 harddisks)

Emulation in the development lab:

- Right-click folder in Windows Explorer and select “Properties”
- Old machine took 6 min
- SONAS took 15 min



What is happening on the wire?

```
Trans2 Request, FIND_FIRST2, Pattern: \customer
Trans2 Response, FIND_FIRST2, Files: customer
Trans2 Request, FIND_FIRST2, Pattern: \customer\*
Trans2 Response, FIND_FIRST2, Files: . .. G[illegible]
Trans2 Request, QUERY_PATH_INFO, Query File Basic Info, Path: \customer\me[illegible]
Trans2 Response, QUERY_PATH_INFO
Trans2 Request, FIND_FIRST2, Pattern: \customer\me[illegible]\*
Trans2 Response, FIND_FIRST2, Files: . .. e[illegible]
Trans2 Request, QUERY_PATH_INFO, Query File Basic Info, Path: \customer\me[illegible]\jane
Trans2 Response, QUERY_PATH_INFO
```

Windows recursively walks the directory structure

It uses FIND_FIRST2 to determine all children of the current directory

Before descending into a subdirectory, it issues a QUERY_PATH_INFO request for the directory

For each subdirectory of the current directory, it calls FIND_FIRST2 and then descends into each subdirectory found

Issue 1: lstat() processing of wildcards

When processing the FIND_FIRST2 requests (e.g. \customer\dir*), Samba calls lstat() for the path including the *.

The file system sometimes takes an unusual long time to answer these requests

- seems to be related to a cache miss when directory is entered first time
- lstat() calls for existing files are answered much quicker

```
stat("customer/<some more dirs>/*", 0x7fffa00527a0) = -1 ENOENT (No such file or directory) <0.171325>
stat("customer/<some more dirs>/*", 0x7fffa00527a0) = -1 ENOENT (No such file or directory) <0.388874>
stat("customer/<some more dirs>/f1.rib", {st_mode= ...}) = 0 <0.000053>
stat("customer/<some more dirs>/inst02.0027.rib", {st_mode= ...}) = 0 <0.000053>
```

Actions taken:

- **Workaround:** As * is an invalid file name in the CIFS protocol, we hacked a filter into Samba that will identify lstat() calls ending with “/*” and returns ENOENT instead of asking the file system
- Opened bug against file system to fix the long wrong response times for non existent files

locking.tdb is very busy

When processing the FIND_FIRST2 and QUERY_PATH_INFO requests, Samba needs to check if an object has the DELETE_ON_CLOSE flag set.

Entries with the flag set will be omitted from the FIND_FIRST2 results and QUERY_PATH_INFO will be replied with DELETE_PENDING

So for each file, Samba does one look-up in locking.tdb

For each directory, two look-ups are done due to the two requests

Performance is slow due to two issues:

- Samba needs to fetch records twice from CTDB
- Records are not well distributed in the local TDB and across the cluster

For testing, we patched Samba to not make look-ups into locking.tdb for FIND_FIRST2 and QUERY_PATH_INFO. This gave a dramatic speed-up, but actually violates the CIFS protocol as files/directories pending deletion will not be reported as such any more.

Issue 2: DMASTER role for empty records

In case the locking record Samba needs to look at is in the local TDB and the node is the DMASTER for the record, Samba does not have to ask CTDB to fetch the record from the cluster.

If the record cannot be found locally or the local node is not the DMASTER for the record, Samba does an unlocked fetch call to CTDB to retrieve the record.

If no record in locking.tdb exists for an object (true for 99% of the cases in this workload), CTDB will implicitly create an empty record on the requesting node and on the regular LMASTER for the record and will assign the LMASTER to be the DMASTER for this empty record.

When Samba needs to access the record a second time (for all QUERY_PATH_INFO calls), most of the empty records are not in the local TDB and so Samba has to fetch them a second time across the network. Only records for which the local node would be the LMASTER can be found in the local TDB.

Workaround: make node creating new record the DMASTER

Solution: read-only locks in CTDB

CTDB read-only locks

Currently under design

Will allow multiple nodes to have a read-only copy of a record in their local TDB.

In principle this is a caching protocol.

Nodes can request to get a read-only copy of a record from the DMASTER

DMASTER has a list of all nodes possessing read-only copies

Only the DMASTER can perform updates on the record

In case record gets updated or migrated, other nodes will be notified to invalidate their local copy

During a recovery, local copies will be thrown away

Issue 3: Record distribution in (C)TDB

key for locking records in a cluster is based on inode numbers

inode numbers are very similar

tdb_hash() does a bad job distributing the entries among the hash buckets

Half of the hash buckets were unused in the local TDB

LMASTER assignment was also very uneven, only half of the nodes was used for being LMASTER

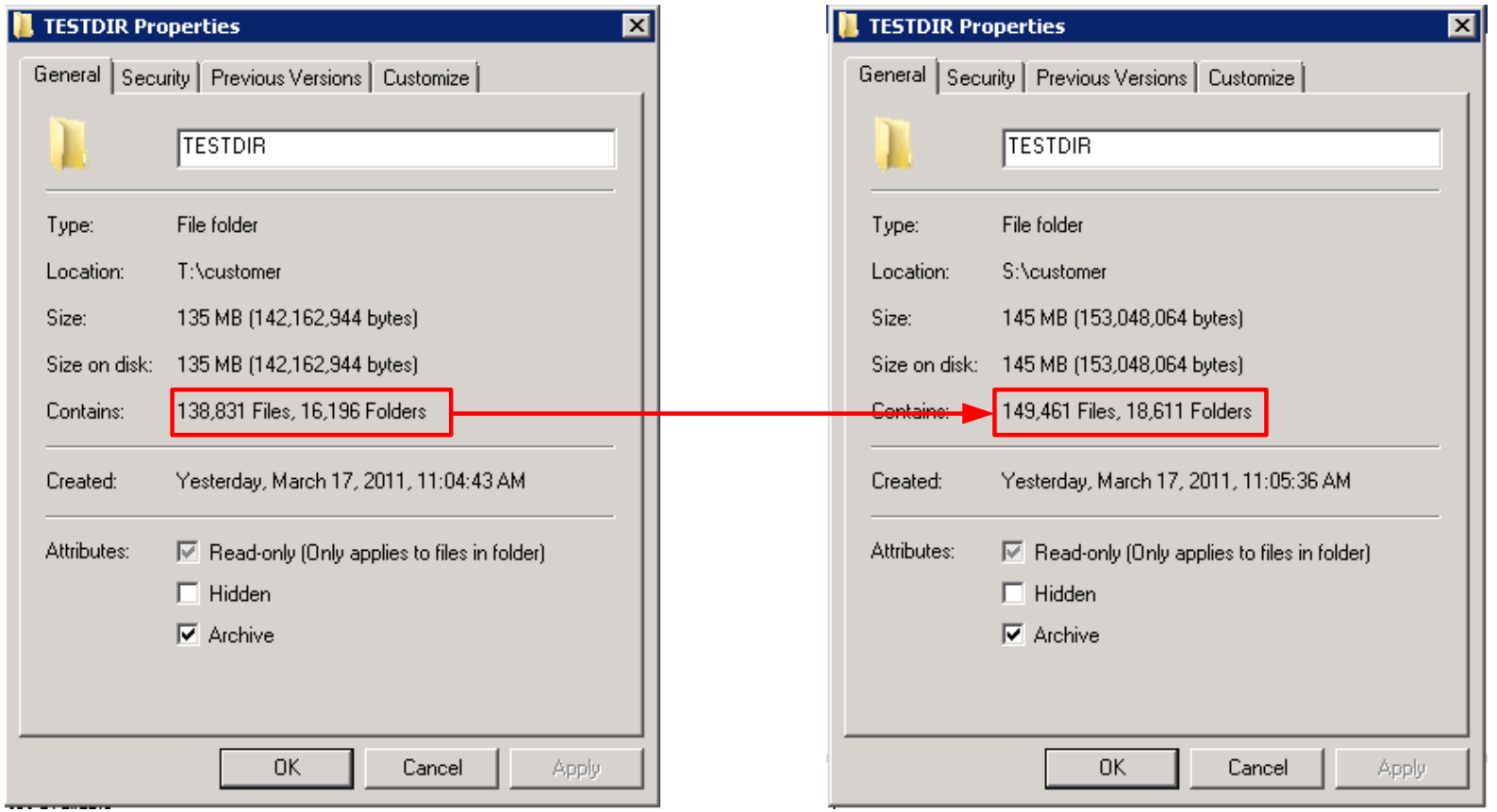
Solution:

replace tdb_hash with jenkins hash

- balanced distribution of records among hash buckets in local TDB
- balanced LMASTER assignments for records in CTDB

CIFS versus SMB2

progress after 1 minute of directory scanning



Customer II

Movie rendering company located in India

Using Maya rendering software

Workload:

- At the same time, multiple users try to open the Maya project file in their 3D modeling application for reading
- Customer uses a central Maya project file which contains links to the textures and RIB files
- The Maya project file is also loaded at the beginning of the rendering process by their rendering farm

Problem:

- Slow start-up of rendering application
- Some clients saw time-outs when accessing the project file

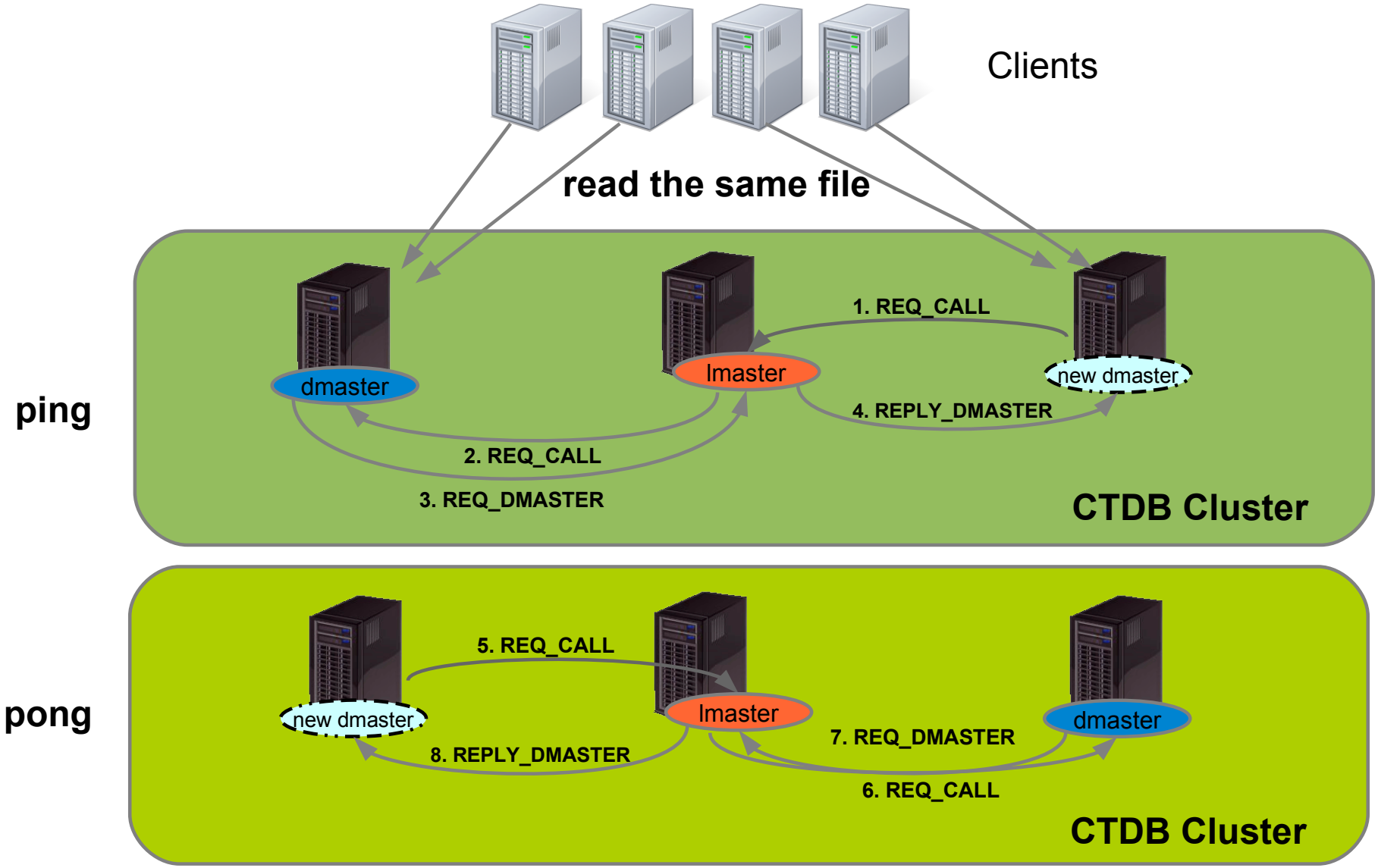
Issue 4: DMASTER Ping-Pong

When the same file is accessed through multiple cluster nodes, Samba will need to check the record for the file in brlock.tdb on each read request.

This leads to many DMASTER migrations that put load on CTDB and increase the latency.

```
172.31.136.2 172.31.132.1 CTDB    REQ_CALL 1->0
172.31.132.1 172.31.136.2 CTDB    REQ_DMASTER 0->3
172.31.132.1 172.31.136.2 CTDB    REQ_CALL 0->3
172.31.136.2 172.31.132.1 CTDB    REPLY_DMASTER 3->0
172.31.136.2 172.31.132.1 CTDB    REQ_CALL 1->0
172.31.132.1 172.31.136.2 CTDB    REQ_DMASTER 0->3
172.31.132.1 172.31.136.2 CTDB    REQ_CALL 0->3
172.31.136.2 172.31.132.1 CTDB    REPLY_DMASTER 3->0
172.31.136.2 172.31.132.1 CTDB    REQ_CALL 1->0
172.31.132.1 172.31.136.2 CTDB    REQ_DMASTER 0->3
172.31.132.1 172.31.136.2 CTDB    REQ_CALL 0->3
```

Issue 4: DMASTER Ping-Pong



Issue 4: DMASTER Ping-Pong

Code changes made for the DMASTER for empty records might have made things worse as DMASTER migration now happens on every fetch and not only after the fifth consecutive fetch from the same node.

Workaround:

- Make sure all clients access the file through the same cluster node

Final solution:

- Implement CTDB read-only locks

Customer III

Customer from the industrial sector located in Germany

Workload:

- Research departments around the world storing data
- large Active Directory with many users which are member of many groups (>300)

Problem:

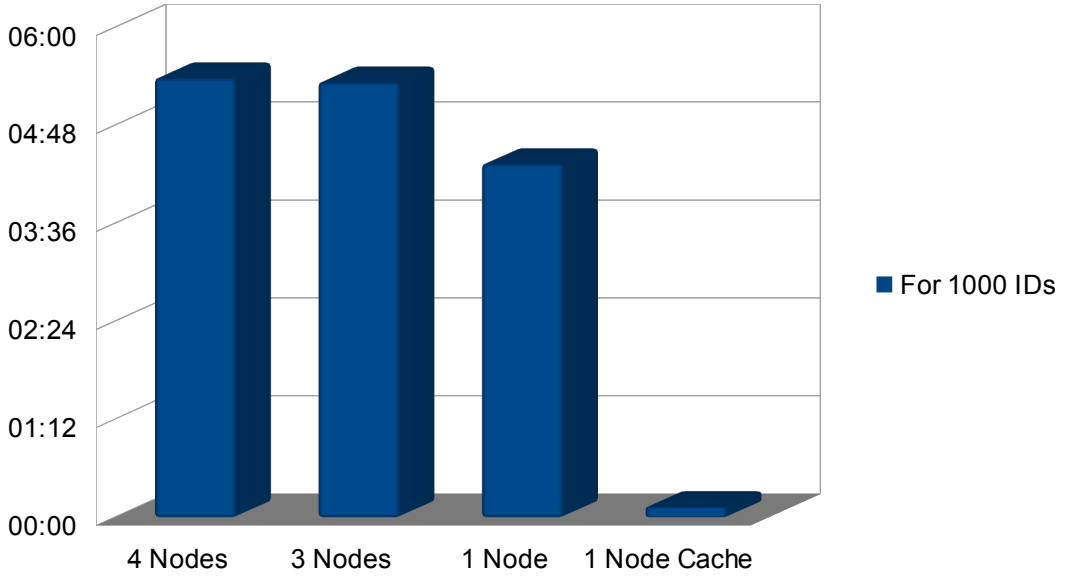
- Customer ran into time-outs and connection losses during session setup
- The timeouts only happen during the very first login of a user to the SONAS cluster

Initial Analysis:

- Creation of ID mappings for the user and groups took longer than the CIFS client connect timeout
- If many new users access SONAS at the same time, some of them failed (time-out)
- The same happens if a user is in many groups - connection failed (time-out)

ID Mapping Performance in a cluster (idmap_tdb2)

Performance test of the idmap_tdb2 backend with multiple cluster nodes:



Conclusion:

ID mapping performance was slow (~200 IDs per minute – cluster-wide!)

Creating ID mappings for 1000 IDs took >5 minutes.

Distributing the load across multiple cluster nodes is counter-productive.

Lookup for existing ID mappings is fast (8 seconds for 1000 IDs)

*4-Nodes=5:23 3-Nodes=5:20 1-Node=4:20 1-Node Cache=8s

Issue 5: ID Mapping Performance in a cluster (idmap_tdb2)

Samba stores the ID mappings in a persistent TDB (idmap_tdb2) that is clustered by CTDB and uses transactions to ensure the integrity.

Allocating an ID from the high water mark and actually writing the mapping was not a single but two transactions.

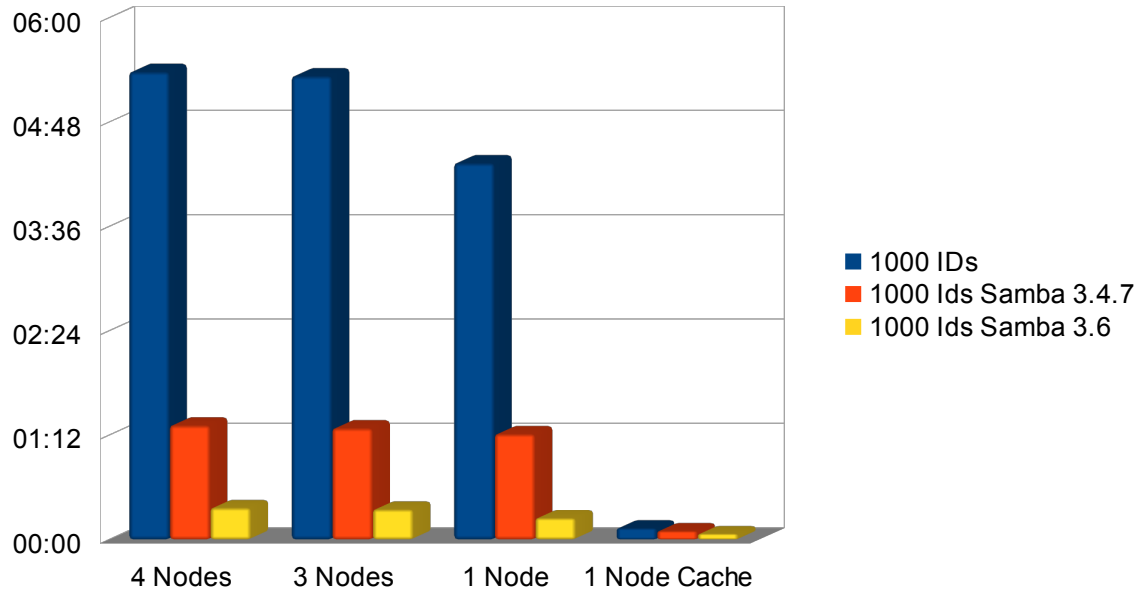
A single transaction is started for each group where the user is member of.

Solutions:

- Provide a tool to pre-fill the ID map database
- Wrap "high water mark and actually writing the mapping" into a single transaction (Samba 3.4.7)
- For user in many groups problem - map many SIDs to gids at once in one single transaction (Samba 3.6)
- Implement new ID mapping module based on idmap_rid but with less configuration overhead (idmap_autorid)

ID Mapping Performance in a cluster (idmap_tdb2)

After Samba improvements (3.2.1 vs 3.4.7 vs 3.6)



Other improvements:
Create ID mapping for all member groups in a single transaction (Samba 3.6)

idmap_autorid

combines the approaches of idmap_tdb and idmap_rid

idmap_tdb is the favorite backend but has some drawbacks

- it is non-deterministic: Ids are given on first-come, first-serve
- it uses a database to store the mappings and you should never loose this database

idmap_autorid applies the rid algorithm to all domains, but automatically defines ranges for each domain in the forest instead of requiring manual configuration for each domain

- easy to configure: just one optional parameter
- no need to up Samba configuration when trusted domain is added
- deterministic mapping behavior
- fast operation
- mappings are easy to replicate to second cluster

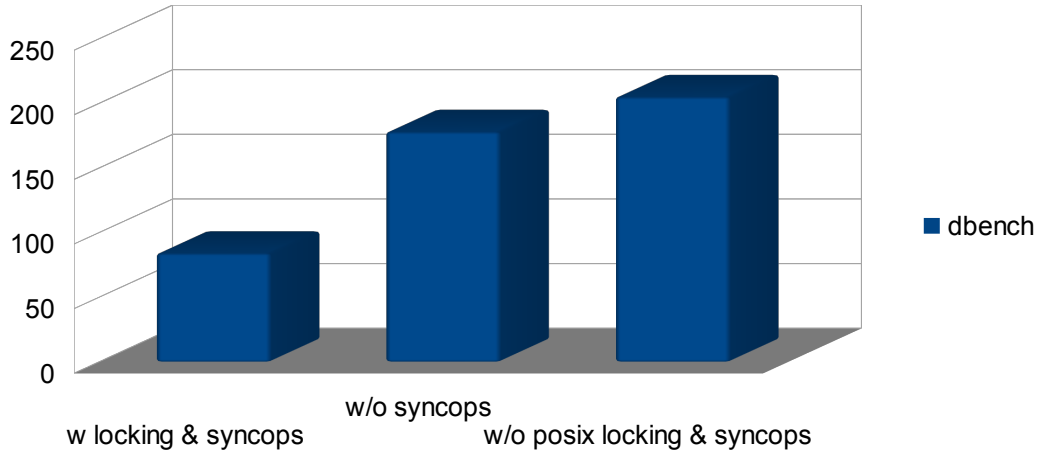
Give it a try!

Metadata synchronization performance (vfs_syncops module)

Metadata operations (like mkdir,unlink,..) must be sync()'ed to disk immediately to avoid potential data inconsistency problems on cluster node failover.

We use the Samba vfs_syncops module (which does a fsync() on the parent directory)

Tridge and Ronnie did some benchmarking with dbench



Results:

- strace showed that most of cost is fsync()
- The various additions for clustered Samba are costing up to 2/3 of the performance for this workload (which is heavy on meta-data ops).

Metadata synchronization performance (vfs_syncops module)

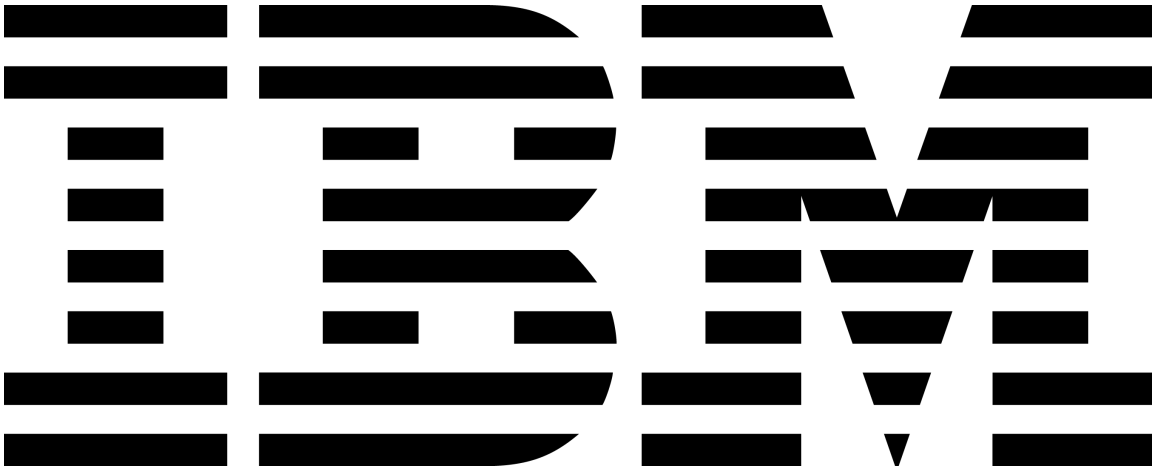
Solution:

- Introduce new (internal) GPFS parameter
- GPFS will automatically commit all the metadata operations (mkdir, unlink, etc.) if the new option is enabled.
- Metadata ops usually involve multiple objects, and often Samba won't have explicitly opened these (e.g., two directories involved in a rename). Hence, to commit a single metadata operation Samba would need to do several opens, fsyncs and closes.
- vfs_syncops module used for data sync only (sync on close)

Performance improvements to be quantified ....

Fix Status

File system fix for wildcard stats		
Filter out * stat calls in Samba • Consider filtering this out cleanly in Samba code, Jeremy seems to have already worked on that		Samba 3.6?
Make (C)TDB record distribution more balanced		CTDB 1.2.8
Fix DMASTER for empty records		CTDB 1.2.14
Implement CTDB read-only locks		CTDB 1.3?
Increase ID Map Performance		Samba 3.6
Speed up metadata synchronisation		Samba 3.6 GPFS 3.3


IBM