Experiences in Clustering CIFS for IBM Scale Out Network Attached Storage (SONAS)

Dr. Jens-Peter Akelbein

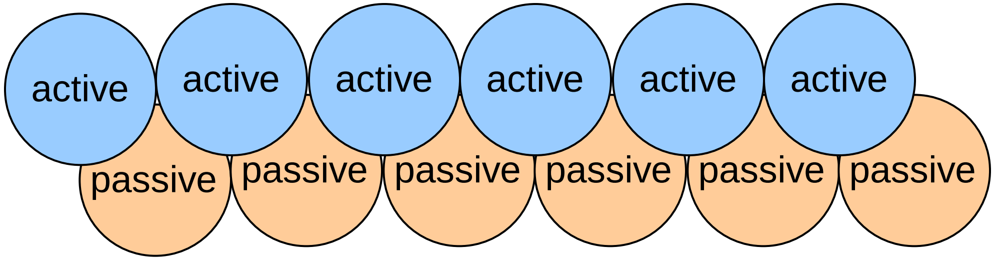
Mathias Dietz, Christian Ambach

IBM Germany R&D

IBM SONAS Overview

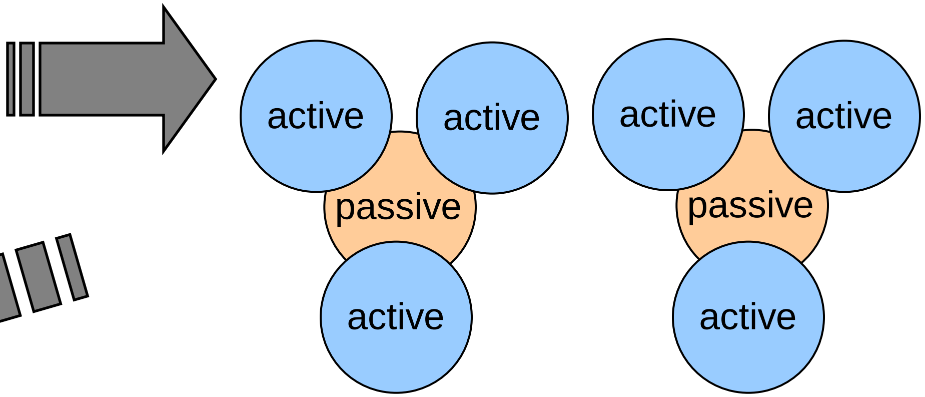
- Enterprise class network attached storage
- Scale out design to offer linear scaling of performance and capacity growth up to 14,4 petabytes
- Provides single global repository to manage multiple petabytes of storage
- Up to a billion files in a single file system
- Policy-driven file lifecycle management (disk and tape)
- Supports CIFS, NFS, HTTP, FTP
- Provides centralized management by CLI or GUI
- Enable desaster recovery and business continuity with asynchronous replication
- NDMP designed to provide full and incremental backup of files
- Integrated Tivoli Storage Manager for backup and HSM
- Securing data with Antivirus integration from most commonly deployed ISV applications

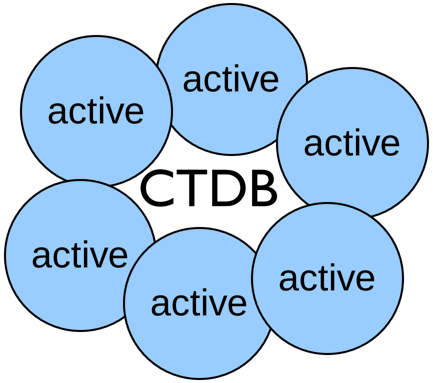
Active/Active nodes



- **Reduce redundancy**
  - 6 nodes, 8 servers
  - After the first node failure you run with a chance of 3 out of 7 the next failure will cause an outage
  - You still have 33% more power consumption

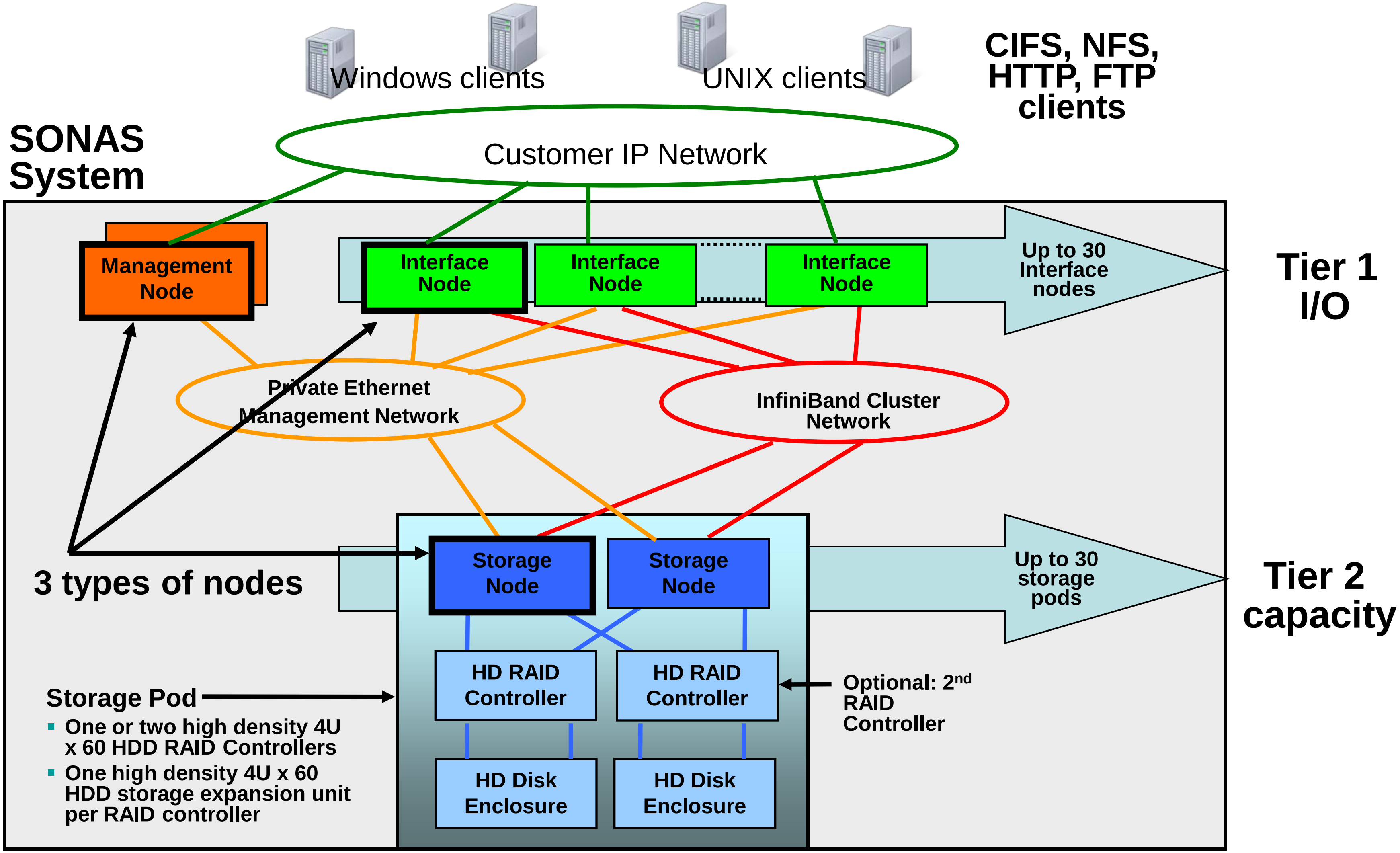
- **100% redundancy**
  - 6 nodes, 12 servers
  - After the first node failure you run with a chance of 1 out of 11 the next failure will cause an outage
  - You have twice as many power consumption





- **0% redundant servers**
  - 6 nodes consolidated as 1 server
  - First node failure just means there are 5 nodes left for NAS protocol I/O (16,6% bandwidth reduction)
  - Second node failure never cause an outage (33,3% bandwidth reduction)
  - Third node failure, again, no outage
  - 6 All-Active nodes provides vastly higher redundancy even than the most expensive 12 node Active/Passive configuration

SONAS hardware structure

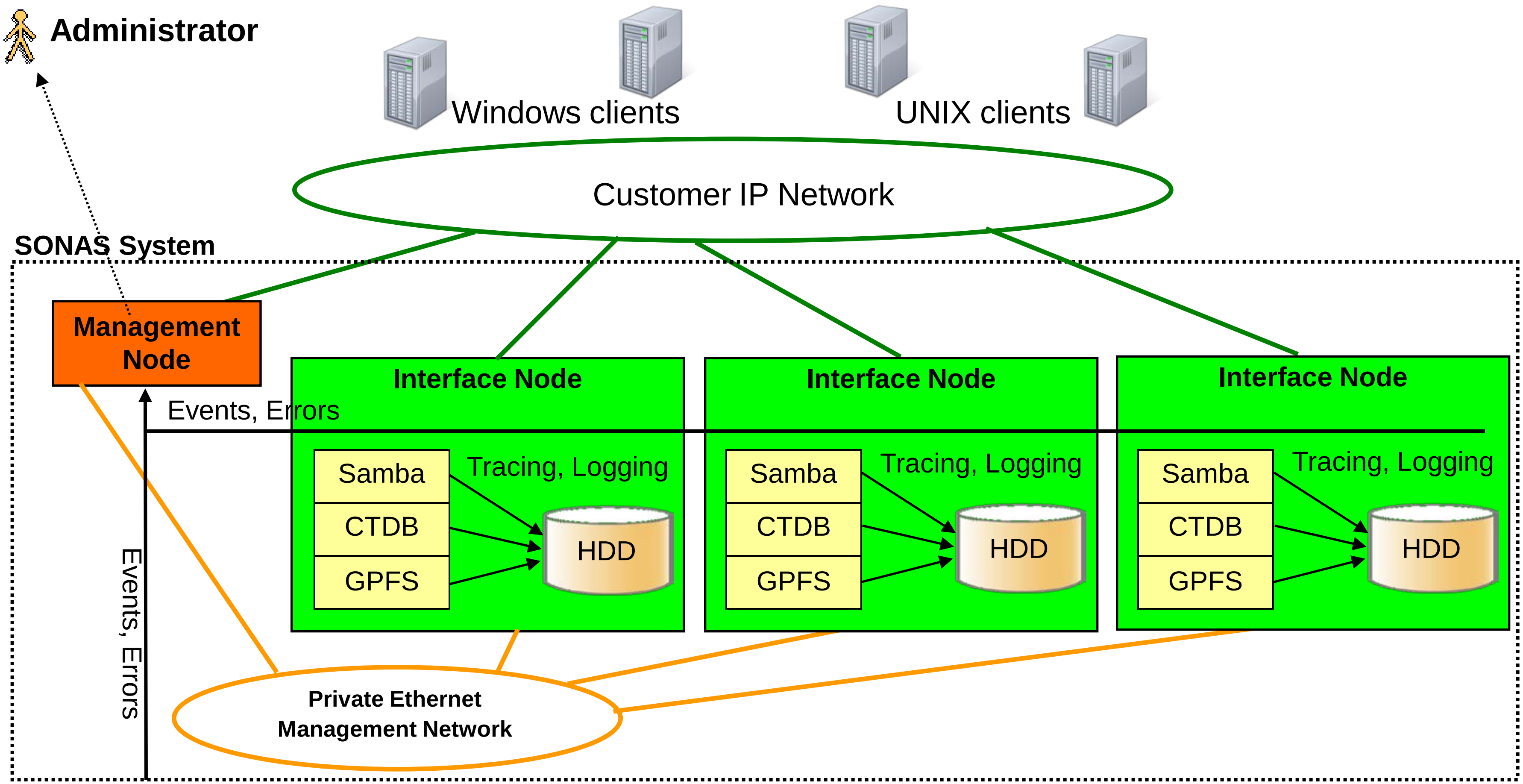


- One or two high density 4U x 60 HDD RAID Controllers
- One high density 4U x 60 HDD storage expansion unit per RAID controller

Some challenges in clustering CIFS

- Monitoring of several nodes (SONAS has max. 92) within a cluster instead of only one or two nodes
- Keep the configuration synchronous across all nodes in a cluster
- Concurrent code upgrade of the software stack
- Performance depends on the right implementation of in-band functions requiring interaction between different nodes in the cluster
- Workload specific scenarios

Monitoring clustered CIFS



Consistent CIFS configuration

- Larger clusters may contain nodes in various states
  - nodes may be taken offline, nodes can be added or removed

- Samba stores its configuration in smb.conf by default
  - distributing smb.conf across the cluster can result in various error situations impacting consistency

- CTDB already distributes persistent TDBs (locking.tdb, secrets.tdb etc.) for Samba across the cluster

=> Registry was developed for Samba 3.2 being stored as a clustered registry in CTDB

- CTDB pushes configuration consistently across the cluster

How about other protocols?

- Other protocols (NFS, HTTP, FTP) use configuration files, too
  - Why not using the same mechanism sharing them via CTDB?

- Store config files in custom registry keys in the CTDB
  - CTDB keeps registry keys synchronized across all nodes
  - Push configuration from the registry key into the local configuration file on all nodes
  - CTDB provides the rc.local hook calling a script each time when an event script is being called; rc.local script pushes configuration locally
  - Problems starts with timing as the runtime of rc.local is hardly predictable
  - Other problems lead to deadlocks, hard to debug, unclear separation of duty between clustering protocols and service configuration management

=> Configuration Management Daemon was introduced running independently of CTDB

Concurrent Code Upgrades

- Requirements: ideally zero downtime of a NAS services provided by the cluster
- Challenges for CIFS using Samba:
  - keep changing data structures consistent across different versions
  - Samba / CTDB do not have versioned data structures for communication so far; no guarantee on having different versions being compatible

- Solution for code upgrades (2 node example)
  - □ disconnect node 1
  - □ upgrade node 1
  - □ disallow any Samba/ctdb configuration changes by disabling services on node2
  - □ dump TDBs on node2 and send backups to node 1
  - □ shutdown ctdb on node2
  - □ restart ctdb on node 1 as disabled
  - □ restore all relevant persistent TDBs on node 1
  - □ allow configuration changes by fully restarting ctdb
  - □ upgrade node2
  - □ node 2 rejoins the cluster
- Extension for n nodes: update each half of the cluster in turn

Node2 provides service

Downtime

Node1 provides service

Central project file access patterns

Movie rendering companies using Maya rendering software

Workload:

Customer uses a central Maya project file which contains links to the textures and RIB files

At the same time, multiple users try to open the Maya project file in their 3D modeling application for reading

The Maya project file is also loaded at the beginning of the rendering process by their rendering farm

Problem:

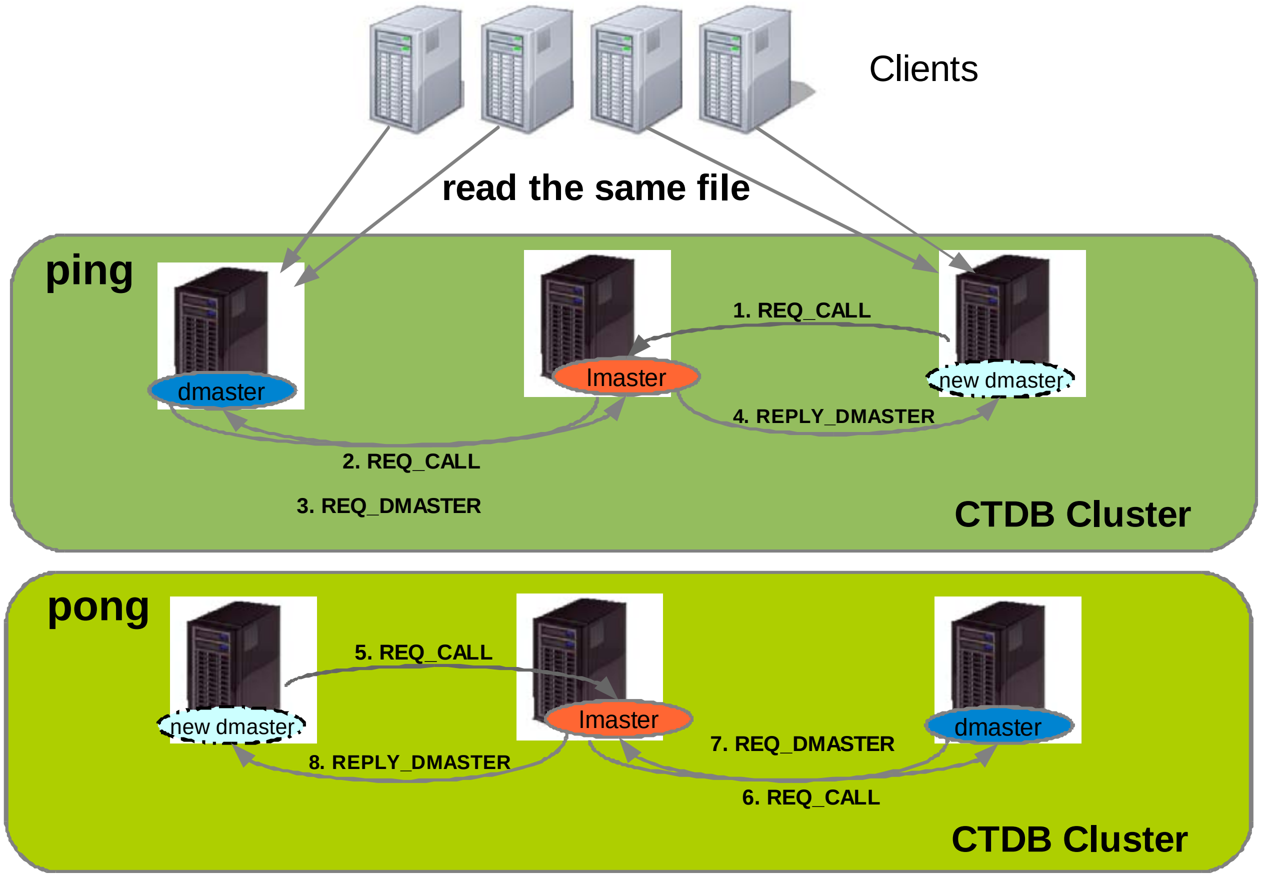
Slow start-up of rendering application

Some clients saw time-outs when accessing the project file

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 requests/responses that put load on CTDB and incre the latency.



Workaround: Make sure all clients access the file through the same cluster

CTDB read-only locks

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

In principle this is a caching protocol.

Only the DMASTER can perform updates on the record

In case record gets updated, other nodes need to invalidate their local copy and fetch a fresh copy

During a recovery, local copies will be thrown away

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

ID mapping performance (1)

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 (2)

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.

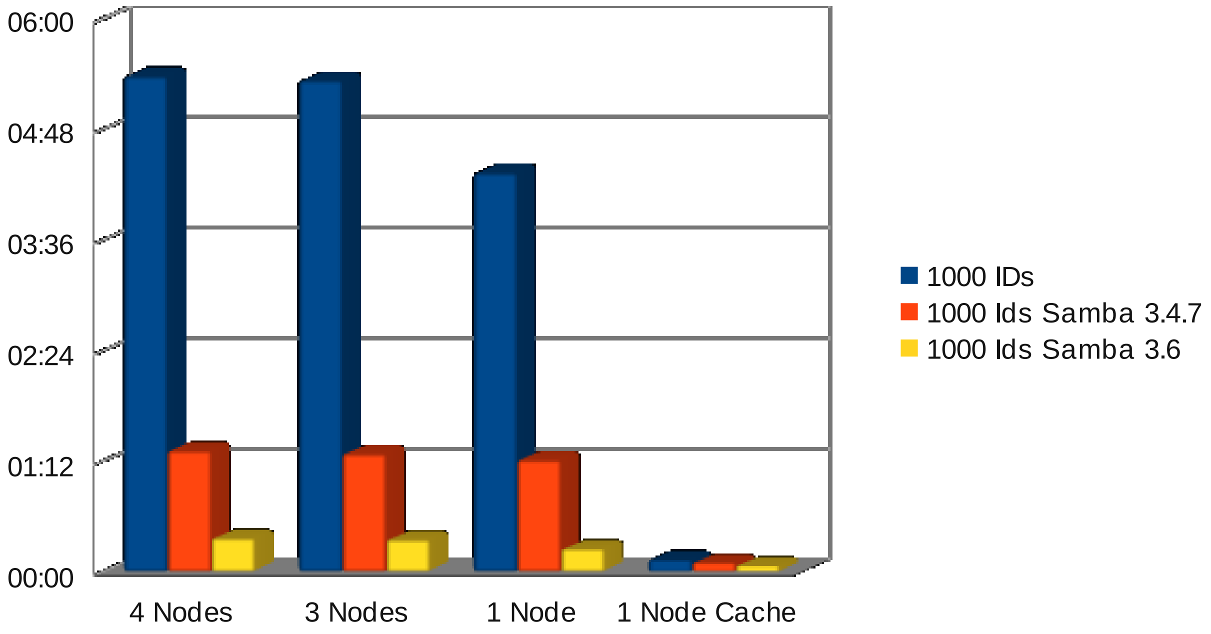
A single transaction is started for each group where the user is member of.

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 (3)

After Samba improvements (3.2.1 vs 3.4.7 vs 3.6)



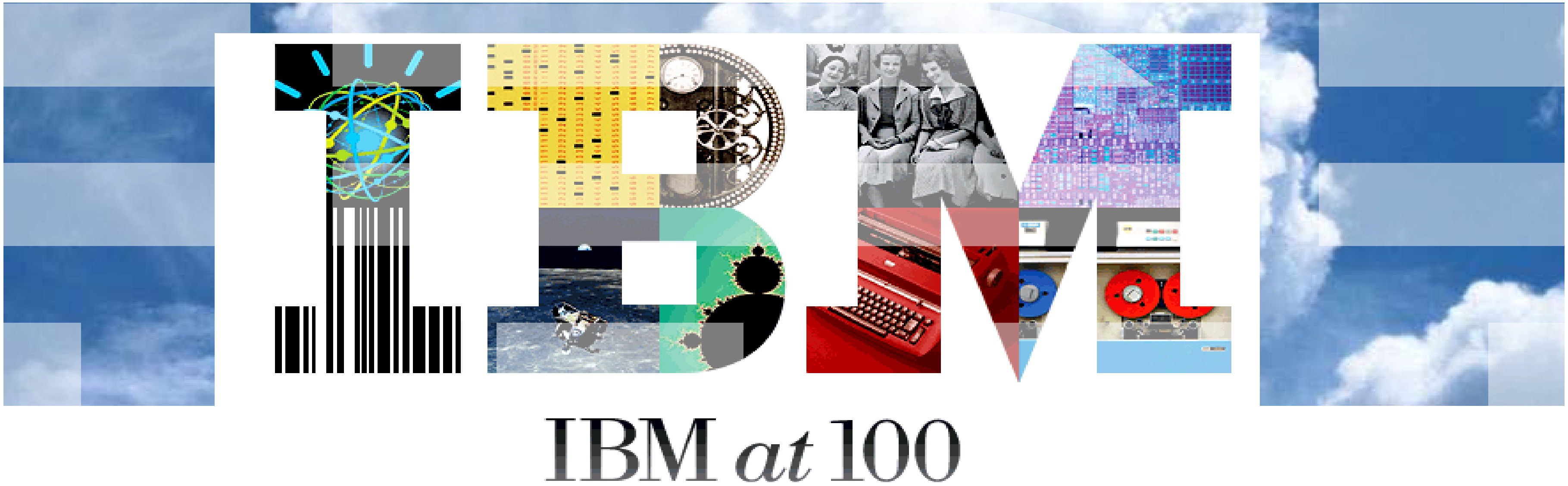
Other improvements:

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

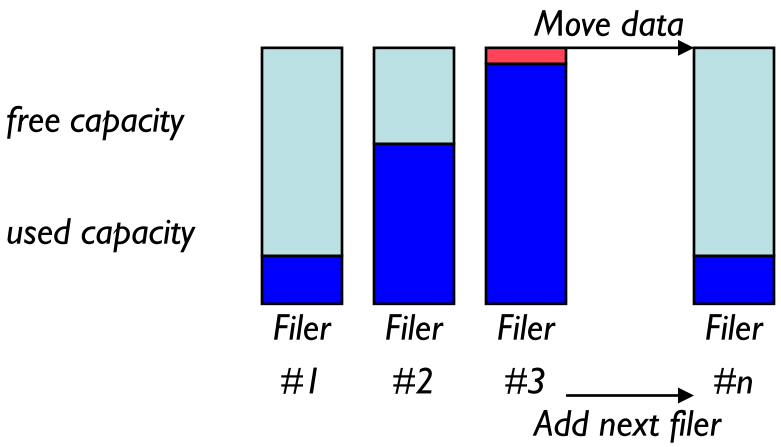
Summary

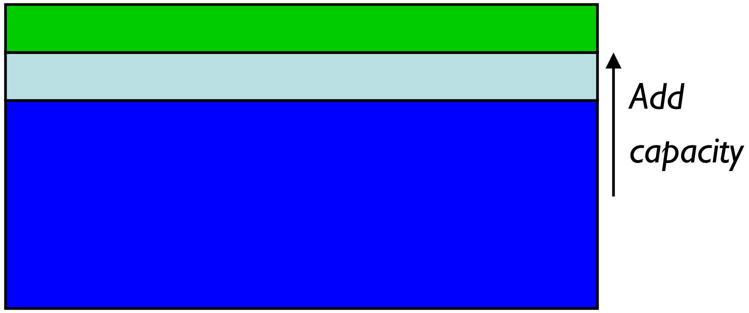
- Clustering CIFS involves more aspects to consider compared to other protocols
- Several improvements in Samba and CTDB over the past years done in regards of Reliability, Availability, Servicability, and Performance
- Upcoming enhancements in open source addresses further improvements for special workloads
- All improvements done in the SONAS software stack without dependency to special hardware

Questions?



True single namespace





Scale Out NAS with a single namespace

How does your NAS infrastructure look like with traditional NAS filers?

- Data growth leads to adding filers again and again
- If you have a few Petabyte you end up with many many small islands
- Someone here having >100 NAS filers?
- If you have petabytes you end up with many filers
- How much of your capacity is really used?
- How much maintenance needs to be spend for large amount of filers?

True single namespace

- Everything is in one large file system with one namespace, no data moves required
- Capacity of one large file system grows by adding storage
- All free capacity is shared; no data aggregation and no data replication needed
- **Active/Active**, all nodes have access to all data