Scalable filesystems
boosting Linux storage solutions

Daniel Kobras
science + computing ag
IT-Dienstleistungen und Software für anspruchsvolle Rechnernetze
Tübingen | München | Berlin | Düsseldorf
Motivation

▪ **User's view:**
  ▪ Storage is a scarce resource that is always
    ▪ too small
    ▪ too slow
  ▪ If it isn't today, it will be in shorter time than expected

▪ **Admin's view:**
  ▪ Storage is a precious resource that is always
    ▪ too unreliable
    ▪ too inflexible

▪ **Boss's view:**
  ▪ Storage is a necessary evil that is always
    ▪ too expensive
Motivation

- Ideal storage solution
  - can grow in capacity and bandwidth
  - allows to transparently move around data
  - is cheap
  - is easy to use
  - doesn't fail
- Scalable filesystems can't do miracles, but they get you closer
Agenda

- Storage solutions that scale (and those that don't)
  - Terminology
  - Theory of operation
  - Implementations
- Case study
- Present and future developments
„I need this future-proof storage infrastructure – what are my options today?“
Grandma's networked storage (NAS)
Traditional NAS: Properties

- **Re-exports** of local filesystems
- „Proxy filesystem“
  - **Network protocol** only, no dedicated on-disk data layout
- Concurrency of local and remote access
- Implementations: NFS, SMB/CIFS
Traditional NAS: Pros and Cons

- **Mature** concepts and implementations
- **Ubiqitous**, usually part of standard installation
- **Convenient** integration
- Single fileservers **cannot scale** with rising demands on bandwidth and capacity
- Multiple fileservers **partition namespace** at physical boundaries
- Client-side virtualisation techniques (autofs, amd, DFS) mitigate problem, but still are subject to partition boundaries
- Virtualisation of backend storage (SAN) solves capacity constraints and increases reliability, but still requires access through conventional fileservers
Clustered storage in the SAN age

- Client 1
- Client 2
- SAN/iSCSI
- Block Device

Locking
SAN-based filesystems: Properties

- Access same filesystem on **shared block device** from multiple hosts
  - Filesystem manages concurrent access through **locking service**
  - **Dumb „server“** (block device), complexity handled on client side
- Terminology: SAN filesystem, cluster filesystem
- Free implementations: OCFS2, GFS
- Proprietary implementations: mostly from major storage vendors (cXFS, MPFS, PolyServe, TotalStorage SFS, Veritas CFS, ...)

© 2008 science + computing ag
SAN-based filesystems: Pros and Cons

- Requires SAN (or SAN-like) infrastructure
  - additional fabric (FibreChannel)
  - suboptimal fabric (iSCSI over Ethernet, etc.)
- Virtualized backend storage allows
  - Replication
  - Relocation
  - Resizing
- Typical problems:
  - Quorum of clients needed for filesystem operation
  - Limited scalability, dependent on implementation characteristics of locking service and supported access patterns
- Usually limited to servers, uncommon on end-user machines
- NFS/CIFS re-export necessary
SAN-based clustered storage

Client 1

SAN/iSCSI

Block Device

Client 2

NFS/CIFS

NFS/CIFS

SAN-based clustered storage

Client 1

SAN/iSCSI

Block Device

Client 2

NFS/CIFS

NFS/CIFS

SAN-based clustered storage

Client 1

SAN/iSCSI

Block Device

Client 2

NFS/CIFS

NFS/CIFS
Serving files from a distributed system

Client

LAN/WAN

Data Server 1

Data Server 2

Block Device

Block Device

Meta-data Server

fs1
Distributed Filesystems: Properties

- Data distributed to **local storage** on multiple servers
- **Metadata service** ties distributed data into single filesystem
  - decouples namespace from physical layout
  - metadata either on single server, or distributed across several nodes
- Implementations:
  - Special purpose: Hadoop, GoogleFS, ...
  - Open Source: AFS, Lustre/HP SFS, Ceph, PVFS2
  - Proprietary: GPFS, PanFS, FhGFS
Distributed Filesystems: Pros and Cons

- Complex system on client and server side
- High **scalability**: additional servers increase bandwidth and capacity
- High **flexibility** due to decoupling of namespace and physical storage
- Data servers: „Block devices with intelligence attached“
  - Some locking complexity can be offloaded on single server instance, allowing to serve high numbers of clients
  - Ubiquitous deployment to all servers and end-user systems possible
„I fancy my data - does it work for real?“
Case study: Scalable filesystem setup

- Automotive engineering, computational fluid dynamics
- Pre-existing storage:
  - Stand-alone Linux servers
  - local storage
  - NFS/CIFS export
  - partitioned namespace
- New storage solution:
  - Single Lustre filesystem
  - Linux cluster nodes, servers, workstations as Lustre clients
  - CIFS export to Windows clients
  - 2 redundant metadata servers, external SCSI storage
  - 2 data servers (Sun „Thumper“)
Case study: Benefits

- Scalable storage link for compute clusters
  - extensible capacity
  - extensible bandwidth
- Single, unified filesystem for clusters and workstations
- Avoids temporary local storage on cluster nodes to prevent data loss on node failure
- Simplified workflow due to central storage
- Increased job turnaround times as copy processes become superfluous
Case study: Network layout
Case study: Lustre bulk I/O

Lustre - 8 Stripes, 2 OSS, IB

bandwidth (kB/s)

record size (kB)
Case study: Lustre random I/O

Lustre - 8 Stripes, 2 OSS, IB

<table>
<thead>
<tr>
<th>record size (kB)</th>
<th>bandwidth (kB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>25000</td>
</tr>
<tr>
<td>16</td>
<td>50000</td>
</tr>
<tr>
<td>32</td>
<td>75000</td>
</tr>
<tr>
<td>64</td>
<td>100000</td>
</tr>
<tr>
<td>128</td>
<td>125000</td>
</tr>
<tr>
<td>256</td>
<td>150000</td>
</tr>
<tr>
<td>512</td>
<td>175000</td>
</tr>
<tr>
<td>1024</td>
<td>200000</td>
</tr>
<tr>
<td>2048</td>
<td>225000</td>
</tr>
<tr>
<td>4096</td>
<td>250000</td>
</tr>
<tr>
<td>8192</td>
<td>275000</td>
</tr>
<tr>
<td>16384</td>
<td>300000</td>
</tr>
<tr>
<td>32768</td>
<td>325000</td>
</tr>
<tr>
<td>65536</td>
<td>350000</td>
</tr>
<tr>
<td>131072</td>
<td>375000</td>
</tr>
<tr>
<td>262144</td>
<td>400000</td>
</tr>
<tr>
<td>524288</td>
<td>425000</td>
</tr>
<tr>
<td>1048576</td>
<td>450000</td>
</tr>
<tr>
<td>2097152</td>
<td>475000</td>
</tr>
<tr>
<td>4194304</td>
<td>500000</td>
</tr>
</tbody>
</table>

- Random read
- Random write
Case study: Lustre scaling

Lustre - 8 Stripes, 2 OSS, IB

agg. bandwidth (kB/s)

Write
Rewrite
Read
Reread

0
100000
200000
300000
400000
500000
600000
700000
800000
900000
1000000

1 clients
2 clients
3 clients
4 clients
Case study: Lustre scaling

32 nodes, 1GB/client, 16MB regions, fixed OSTs

test

kB/s

write rewrite read re-read random read random write Backward read Record Stride read fwrite freewrite fread Freead

0 100000 200000 300000 400000 500000 600000 700000 800000 900000 1000000 1100000 1200000 1300000
Case study: Current status

- > 1 year in production
- Ca. 150 clients access filesystem
- > 1 GB/s aggregate bandwidth
- 33 TB net capacity
- Combination of open-source software (Lustre, heartbeat, Linux...) and (more or less) „standard“ hardware components!
- Central storage for all project data within workgroup
- Additional storage servers planned in the near future
- Similar installations in several neighbouring groups have followed
Case study: Problem areas

- Storage servers: redundant hardware components, but system-level failure stalls filesystem
- No (trivially) scalable backup concept
  - Backup/restore via multiple filesystem clients possible, but requires manual tuning
  - Full/differential backups undesirable
    - Backup software needs to support incremental-only schemes
    - Integration with HSM systems desirable
- Client-side modifications required
„Does it get any more convenient?“
Scalable filesystems + NAS

- **Idea:**
  - Install scalable filesystem on cluster of NAS servers
  - Re-export filesystem from all server nodes simultaneously to many clients
- **Requirement:**
  - Cluster-aware NFS/CIFS servers to ensure lock consistency
- **Benefits:**
  - Easy access from clients via native protocol to whole namespace
  - Scalable bandwidth and capacity
  - High availability
- **Implementations:**
  - CTDB (in Samba 3.2) with GPFS, GFS, Lustre...
  - Alternative: pNFS/NFSv4.1 (draft standard)
Wrapping it up
Scalable filesystems in a Linux world

- For numerous scalable filesystems, Linux is the primary platform
- Linux-based scalable filesystems allow to build **fast, capable, reliable** and **affordable, custom-tailored** storage clusters from **commodity hardware**, and proprietary or open-source software components
- We've had this before: Beowulf clusters revolutionized high-performance computing, and made Linux the pre-dominant supercomputing platform
- Available software provides the potential for Linux to assume a similar role for scalable storage solutions in the near future
Thank you for your attention.

Daniel Kobras
science + computing ag
www.science-computing.de

Telefon 07071 9457-493
d.kobras@science-computing.de