Geo-Redundant Handover / Failover: Agenda

- Motivation: why GEO-redundancy
- Long-distance asynchronous replication
- Current OPs status: petabytes & co
- What is the future prosumer device?
  - Both local and remote storage location transparent
  - Planned handover without service interruption
  - Unplanned failover as best as possible
- Discussion
Growth at 1&1 Ionos ShaHoLin = Shared Hosting Linux

- 6 datacenters at 2 continents, pair distance > 50 km
- ~ 10 millions of customer home directories
- ~ 10 billions of inodes
- > 7 petabytes allocated in ~ 4000 xfs instances

LVM > 10 PB $\times$ 2 for geo-redundancy via MARS
https://github.com/schoebel/mars

- Growth rate ~ 20 % per year

Operational since 2014
Example: 2021 Ahrtal geo disaster
- Disaster = earthquake, flood, terrorist attack, full power outage, ...

BSI Papers
Kriterien für die Standortwahl höchstverfügbarer und georedundanter Rechenzentren

in English: Criteria for Locations of Highly Available and Geo-Redundant Datacenters

- Stimulated some controversial discussions, but see commentary
  https://www.it-finanzmagazin.de/bsi-rechenzentren-entfernung-bafin-84078/

Conclusions: distances > 200 km „recommended“

Influence future legislation (EU / international)

New: KRITIS
Long-Distance Asynchronous Replication

- Synchronous does not generally work over ≈50 km
  - like iSCSI over 50 km
- Need asynchronous Replication
  - Application specific, e.g. mySQL replication
  - Commercial appliances: $$$ €€€
  - OpenSource
    - plain DRBD is not asynchronous
      - commercial DRBD-Proxy: RAM buffering
    - MARS: truly asynchronous + persistent buffering
      + transaction logging + CRC || MD5 checksums
      + Anytime Consistency
Replication at Block Level vs FS Level

- **Userspace Application Layer**: Apache, PHP, mySQL, Mail Queues, etc

- **Filesystem Layer**: xfs, ext4, btrfs, zfs, ... vs nfs, Ceph, Swift, ...
  - Potential Cut Point A for Distributed System
  - ~25 Operation Types
  - ~100,000 Ops / s

- **Caching Layer**: Page Cache, dentry Cache, ...
  - **Potential Cut Point B for Distributed System**: DSM = Distributed Shared Memory
  - => Cache Coherence Problem!
  - Reduction: ~1:100
  - 2 Operation Types (r/w)
  - ~1,000 Ops / s

- **Block Layer**: LVM, DRBD / MARS
  - Potential Cut Point C for Distributed System
  - +++ replication of VMS

- **Hardware**: Hardware-RAID, BBU, ...
  - +++ LONG DISTANCES
  - +++ replication of VMS
MARS Current Status

- **Kernel module + marsadm tool**

- **MARS source under GPL + docs:**
  - [https://github.com/schoebel/mars/docu/](https://github.com/schoebel/mars/docu/)
  - `mars-user-manual.pdf` ~ 140 pages
  - `architecture-guide-geo-redundancy.pdf` ~ 180 pages

- **mars0.1stable** productive since 02/2014

- **Backbone of the 1&1 Ionos geo-redundancy feature**
MARS Future Plans in short

- LTS kernels >= 5.10  WIP-qio-for-*
- Prosumer device (WIP => next slides)
- Linux kernel upstream requires a lot of work!
- Backlog: more tooling, integration into other OpenSource projects

Collaboration sought
=> Opportunities for other OpenSource projects!
Any /dev/mars/$vmname can appear at any machine whether “storage” machine or “hypervisor” machine

*automatic* introduction of iSCSI-like network connections on port 7776

*backwards* compatible to classical MARS

LocalProsumer RemoteProsumer
Prosumer Device (2)  Hybrid Machines

... or “hybrid” machine ... or imagine ...

no service interruption
planned handover || unplanned failover
- only the storage
- only /dev/mars/$name
- both in parallel

HW lifecycle support
Prosumer Device (3)  Unplanned Failover Scenarios

- unplanned service interruption unavoidable
- only /dev/mars/$name

both in parallel

planned handover: similar to multipath
Surf to https://github.com/schoebel/mars

- Select the branch WIP-prosumer
- Click on docu/
- Download mars-user-manual.pdf
- Read new chapter 5: The MARS Prosumer Device p.61-79
  - Optionally: consult architecture-guide-geo-redundancy.pdf from branch master

Please contribute!
The CAP Theorem states that it is impossible to achieve all three properties simultaneously:

- **C** = Strict Consistency
- **A** = Availability
- **P** = Partitioning Tolerance

**Pick any 2**

- **C**: Strict Consistency over long distances is violated.
- **A**: Availability over long distances is violated.
- **P**: Partitioning Tolerance over long distances is violated.

Both possible? See architecture guide!
Network Bottlenecks (1) DRBD

- network throughput

- DRBD throughput

- (potential) incident → automatic disconnect

- automatic re-connect

- decreasing throughput limit

- mirror inconsistency ...

- additional throughput needed for re-sync, not possible

- Permanently inconsistent!
Network Bottlenecks (2) MARS

- **network throughput**
- **time**
- **MARS**
- **application throughput, recorded in transaction log**
- **replication network throughput**
- **Best possible throughput behaviour at information theoretic limit**
- **decreasing throughput limit**
Network Bottlenecks: MARS

TCP send buffer way too small

MARS application throughput

MARS network throughput

Best possible throughput behaviour

flaky throughput limit

corresponding DRBD inconsistency

network throughput

time
Use Cases DRBD+proxy vs MARS

**DRBD+proxy**  
(proprietary)

- **Application area:**
  - Distances: any
  - Aynchronously
    - Buffering in RAM
  - Unreliable network leads to **frequent re-syncs**
    - RAM buffer gets lost
    - at cost of actuality
  - **Long** inconsistencies during re-sync
  - Under pressure: **permanent** inconsistency possible
  - High memory overhead
  - Difficult scaling to \(k>2\) nodes

**MARS**  
(GPL)

- **Application area:**
  - Distances: **any** ( \(>>50\) km )
  - Asynchronously
    - near-synchronous modes in preparation
  - Tolerates **unreliable network**
  - Anytime consistency
    - no re-sync
  - Under pressure: no inconsistency
    - possibly at cost of actuality
  - Needs \(\geq 100\)GB in /\text{mars}/ for transaction logfiles
    - dedicated spindle(s) recommended
    - RAID with BBU recommended
  - Easy scaling to \(k>2\) nodes
DRBD+proxy Architectural Challenge

**DRBD Host A (primary)**
- Bitmap A

**Proxy A’**
- Huge RAM buffer
- A \( \neq \) A’ possible

**Proxy B’**
- (essentially unused)

**DRBD Host B (secondary)**
- Bitmap B

Data queue path (several GB buffered)

Completion path (commit messages)

- Same sector #8 occurs \( n \) times in queue

\( n \) times
- \( \Rightarrow \) need \( \log(n) \) bits for counter
- \( \Rightarrow \) but DRBD bitmap has only 1 bit/sector
- \( \Rightarrow \) workarounds exist, but complicated
  - (e.g. additional dynamic memory)
Badly Scaling Architecture: **Big Cluster**

- **Internet** access: \( O(n^2) \) REALTIME Access like cross-bar

  - **Internal Storage (or FS) Network**

    - **Frontend** access: \( O(nk) \)
      - **Storage** access: \( O(n^2) \)

  - **Frontend** for geo-redundancy

  - **Storage**: \( n \times 2 \) for redundancy
Well-Scaling Architecture: **Sharding**

**Internet**  \(O(n \times k)\)  \(\checkmark\)

++ local scalability: spare RAID slots, ...

**Smaller Replication Network** for Batch Migration  \(O(n)\)

+++ traffic shaping possible

=> method *really* scales to petabytes

---

MARS Presentation by Thomas Schöbel-Theuer
Reliability of Architectures: NODE failures

2 Node failure => ALL their disks are unreachable

DRBD or MARS simple pairs

=> no customer-visible incident

Low probability for hitting the same pair, even then: only 1 shard affected => low total downtime

Big Storage Cluster e.g. Ceph, Swift, ...

k=2 replicas not enough => INCIDENT because objects are randomly distributed across whole cluster

same n

O(n^2) network

Higher probability for hitting any 2 nodes, then O(n) clients affected => much higher total downtime

need k >= 3 replicas here
### Cost (1) non-georedundant, n>100 nodes

<table>
<thead>
<tr>
<th>Big Cluster:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typically ≈RAID-10 with k=3 replicas for failure compensation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Disks:</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 300%</td>
</tr>
</tbody>
</table>

| Additional CPU and RAM for storage nodes |

| Additional power |

| Additional HU |

<table>
<thead>
<tr>
<th>Simple Sharding:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Often local RAID-6 sufficient (plus external backup, no further redundancy)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Disks:</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 120%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Client == Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>no storage network</td>
</tr>
</tbody>
</table>

| Hardware RAID controllers with BBU cache on 1 card |

<p>| Less power, less HU |</p>
<table>
<thead>
<tr>
<th>Big Cluster:</th>
<th>Geo-redundant Sharding:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- 2X ≈ RAID-10 for failure compensation (k=6 replicas needed, smaller does not work in long-lasting DC failure scenarios)</td>
<td>- 2 x local RAID-6</td>
</tr>
<tr>
<td>- MARS for long distances or DRBD for room redundancy</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Disks:</th>
<th>Disks:</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 600%</td>
<td>&lt; 240%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Additional CPU and RAM for storage nodes</th>
<th>Hardware RAID controllers with BBU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Additional power</td>
<td>Less power</td>
</tr>
<tr>
<td>Additional HU</td>
<td>Less HU</td>
</tr>
</tbody>
</table>
### Cost (1+2): Geo-Redundancy Cheaper than Big Cluster

**Single Big Cluster:**
- ≈RAID-10 with \( k=3 \) replicas for failure compensation
- \( O(n) \) Clients
  + 3 \( O(n) \) storage servers
  + \( O(n^2) \) storage network
- Disks: > 300%
- Additional power
- Additional HU

**Geo-redundant sharding:**
- 2 x local RAID-6
- MARS for long distances or DRBD for room redundancy
- \( 2 \cdot O(n) \) clients = storage servers
- \( O(n) \) replication network
- Disks: < 240%
- Less total power
- Less total HU
  + ++ geo failure scenarios
Cost (3): Geo-Redundancy even Cheaper

Precondition: CPU must not be the bottleneck

Idea: passive LV roles get less CPU

1 datacenter out of 3 may fail

Total Storage: x 2
Total CPU: x 1.5
=> 1.5 \cdot O(n)

HOWTO flexible CPU assignment => next slide
any hypervisor works in client and/or server role and preferably **locally** at the same time
Flexible MARS Background Data Migration football sub-project

Any hypervisor may be source or destination of some LV replicas at the same time