

Virtual Petabytes Storage Pools using MARS



LCA 2018 Presentation by Thomas Schöbel-Theuer

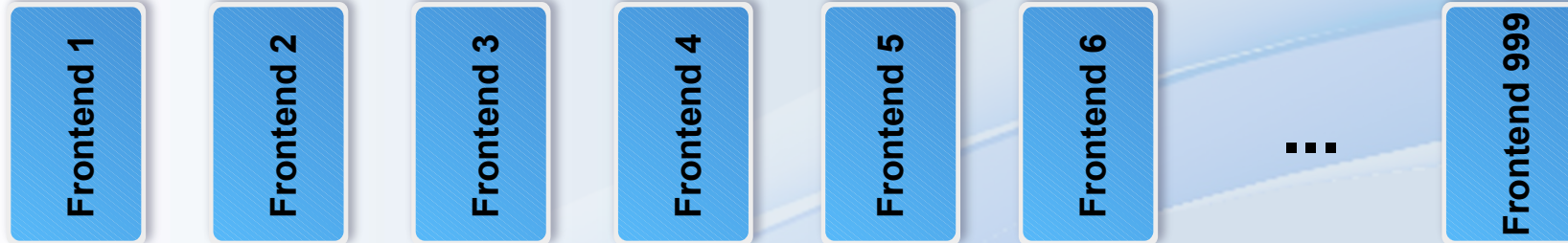
- **Storage Architectures → Scalability**
- **HOWTO Background Migration of LVs**
e.g. for load balancing, HW lifecycle, etc
- **Use Cases for Storage Architectures**
- **Reliability of Storage Architectures**
- **Flexible MARS Sharding + Cluster-on-Demand**
- **Current Status / Future Plans**

Badly Scaling Architecture: **Big Cluster**

Data already partitioned + isolation needed

User 1
User 2
User 3
User 4
User 5
User 6
User 7
User 8
User 9
User 10
User 11
User 12
User 13
User 14
⋮
User 999999

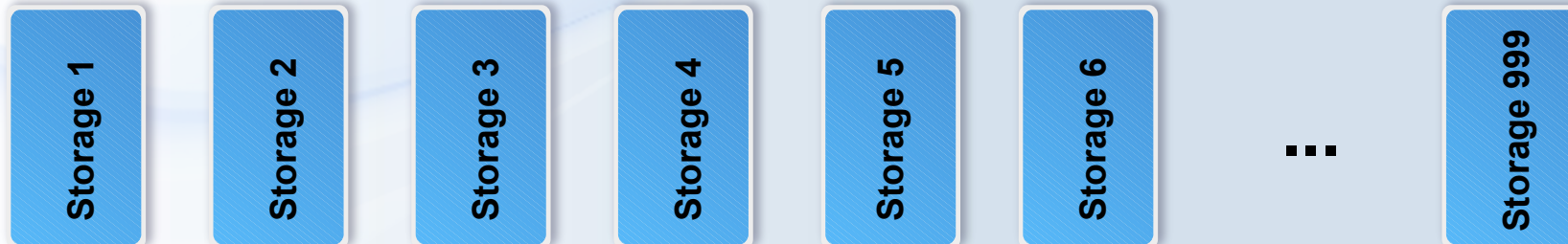
Internet $O(n*k)$



Internal Storage (or FS) Network

$O(n^2)$ REALTIME Access

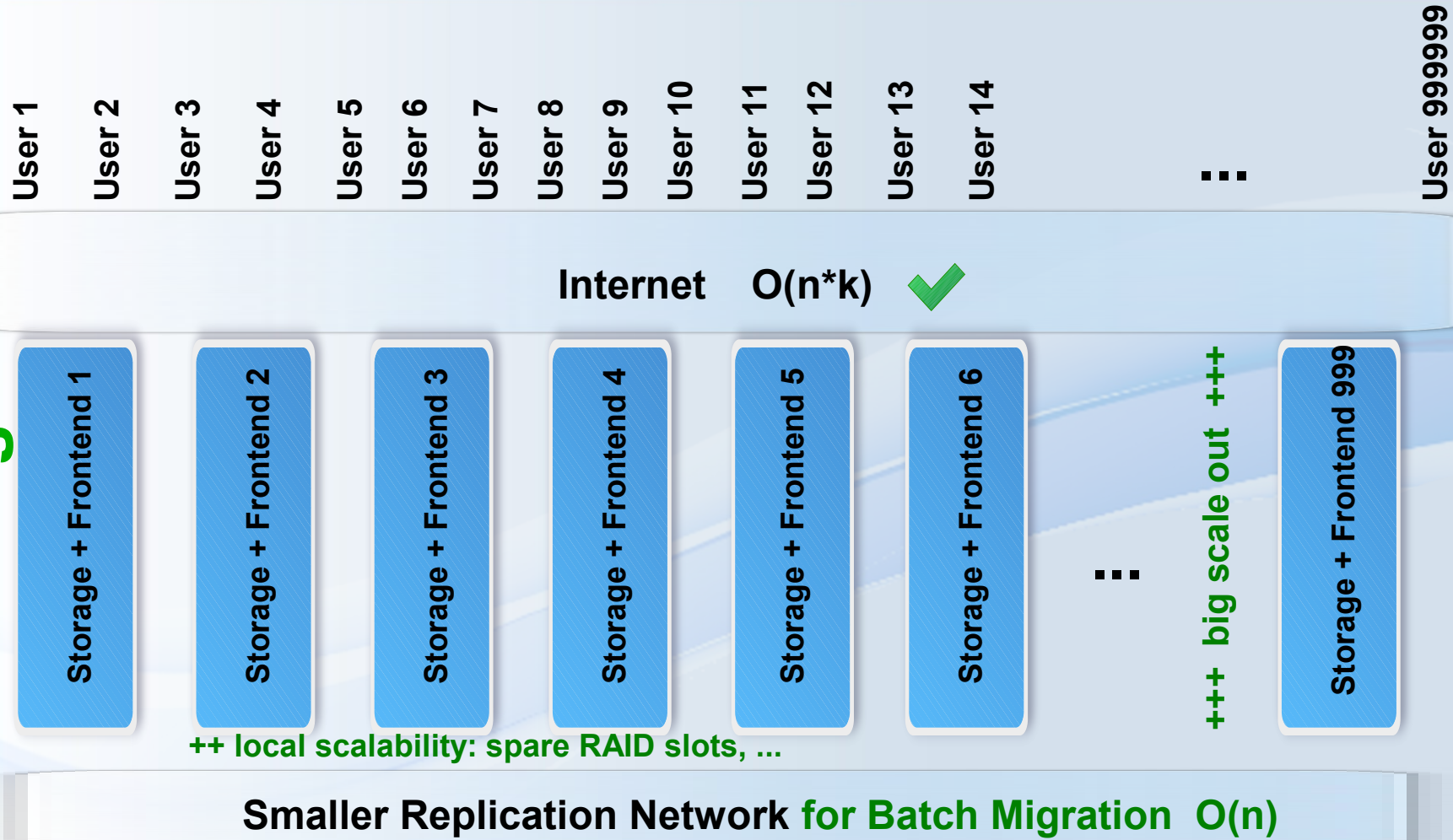
like cross-bar



 **x 2** for geo-redundancy

Well-Scaling Architecture: **Sharding**

cost savings!



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

+++ big scale out +++

+++ traffic shaping possible

=> method *really* scales to petabytes

X 2 for geo-redundancy

HOWTO Background Migration of LVs



HOST A (old) VM is running → **HOST B (new) has spare space**

- | | |
|---|---|
| <ul style="list-style-type: none">- <code>lvdisplay /dev/vg/\$mydata</code>--- (meanwhile VM is altering data)- <code>\$vmmanager stop /dev/mars/\$mydata</code>--- <code>marsadm leave-resource \$mydata</code>- <code>lvremove \$mydata</code> | <ul style="list-style-type: none">-- <code>lvcreate -L \$size \$mydata</code>- <code>marsadm join-resource \$mydata \</code>
<code> /dev/vg/\$mydata</code>- <code>marsadm view: wait for UpToDate</code>-- <code>marsadm primary \$mydata</code>- <code>\$vmmanager start /dev/mars/\$mydata</code>-- |
|---|---|



=> also works with 2 old replicas → 2 new replicas

Example: [tetris.sh](https://github.com/schoebel/mars/contrib/tetris.sh) in github.com/schoebel/mars/contrib/

Big Cluster

- Objects with **non-meaningful** keys
- No logical dependencies between objects → failures should not propagate
- No **data** partitioning possible

Beware

-  **Filesystems on top of spreaded unreliable objects**
-  **Block devices on top of spreaded unreliable objects**

Sharding on top of RAID

- Legal requirements (know where the data is)
- Data is already partitioned
- Structured keys (pathnames)
- Recursively structured data with **in-place** updates, e.g. Block Devices, VMs, ...
- **POSIX-compliant FS needed**

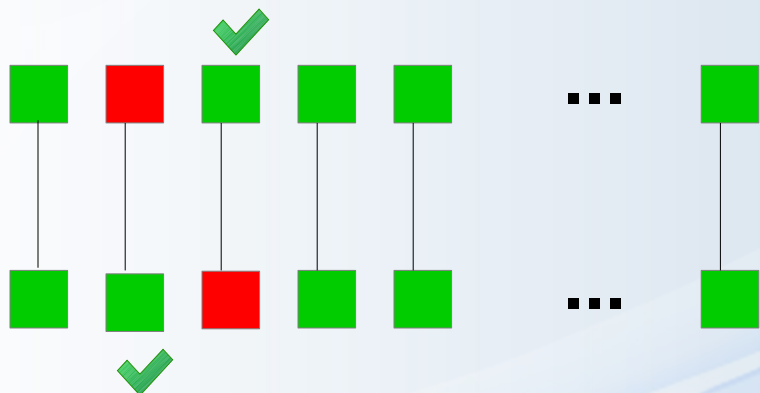
Grey Zones

- when artificial partitioning is possible...
- when data is highly volatile

Reliability of Architectures: NODE failures

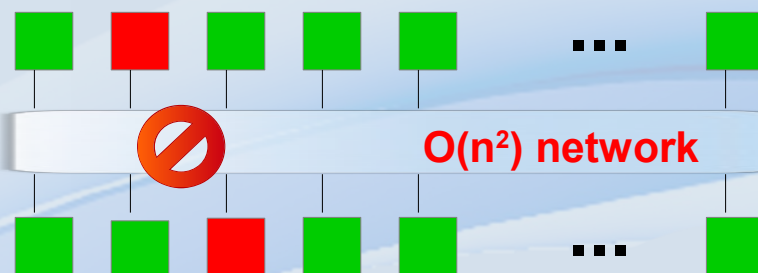
2 Node failure => ALL their disks are unreachable

DRBD or MARS
simple pairs



same n

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



=> no customer-visible incident

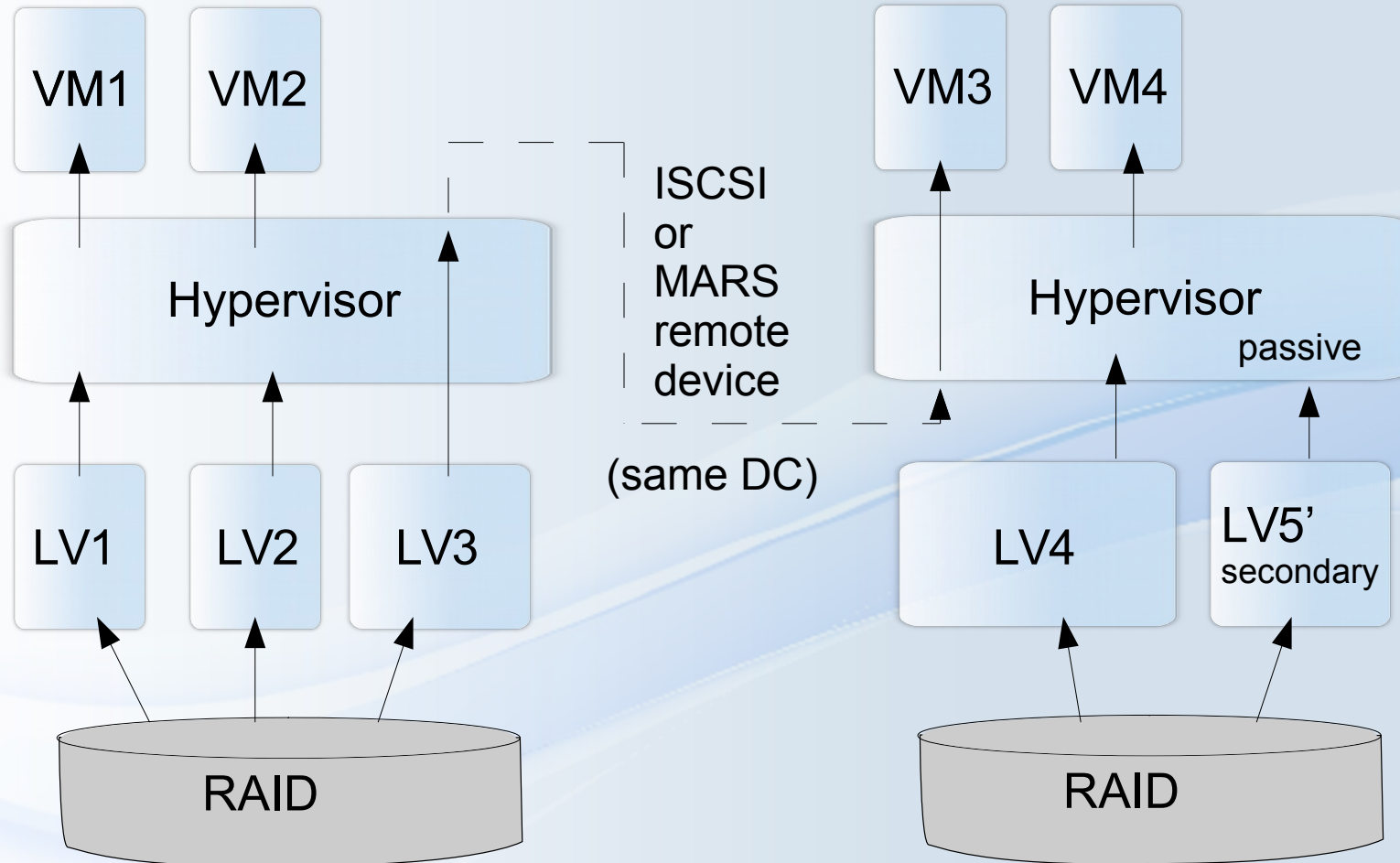
k=2 replicas not enough
=> INCIDENT because objects are randomly distributed across whole cluster

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

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

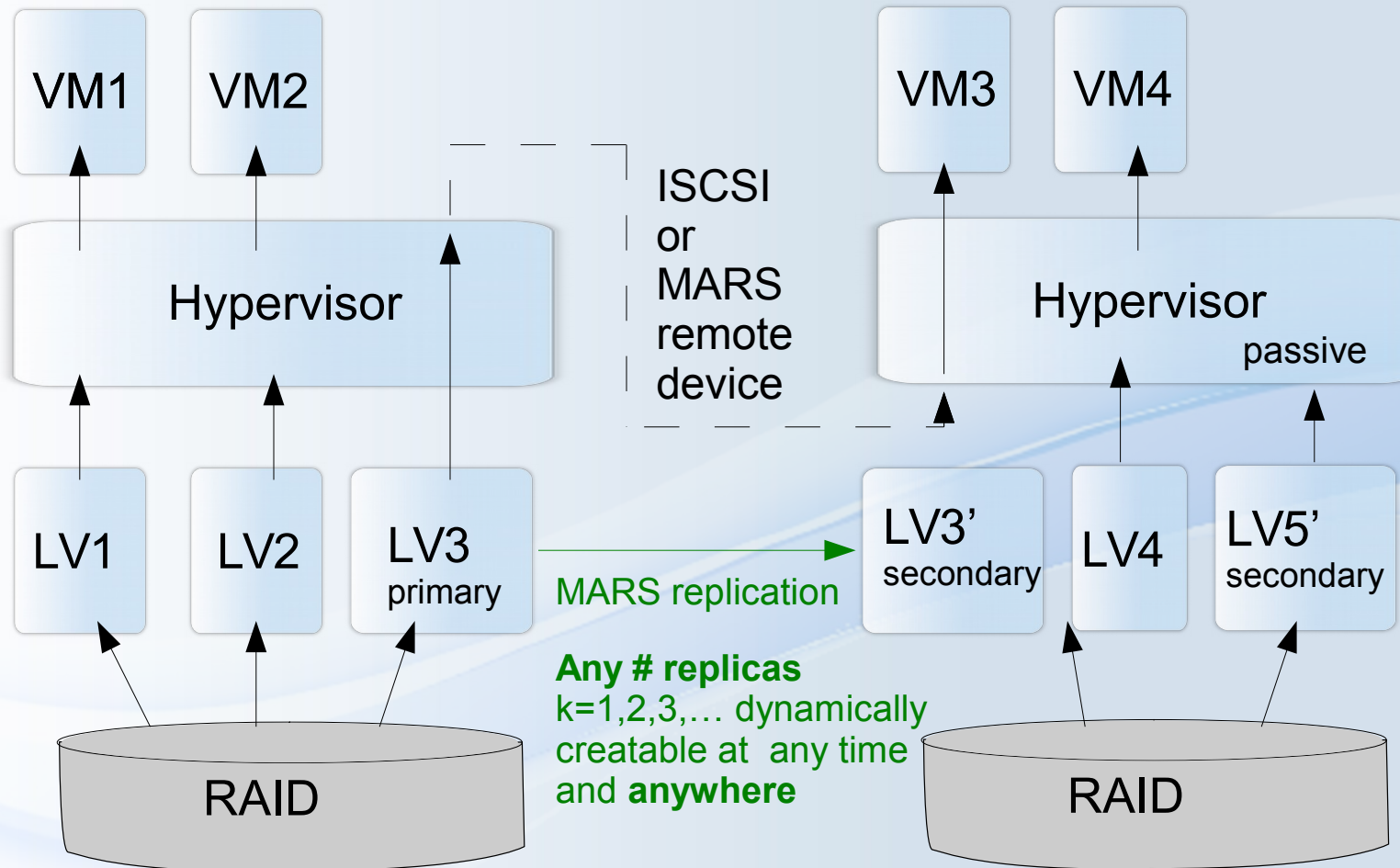
need $k \geq 3$ replicas here

Flexible MARS Sharding + Cluster-on-Demand



any hypervisor works in client and/or server role
and preferably **locally** at the same time

Flexible MARS Background Migration



=> any hypervisor may be source or destination of some LV replicas at the same time

MARS Current Status

MARS source under GPL + docs:

`github.com/schoebel/mars`
`mars-manual.pdf` ~ 100 pages

mars0.1stable productive since 02/2014

Backbone of the 1&1 geo-redundancy feature

MARS status January 2018:

> 5800 servers (shared hosting + databases)

> 2x12 petabyte total

~ 10 billions of inodes in > 2500 xfs instances,
biggest ~ 40 TB

<= 10 LXC Containers on 1 Hypervisor

New internal Efficiency project

- Concentrate more LXC containers on 1 hypervisor
- New public branch mars0.1b with many new features, e.g. mass-scale clustering, socket bundling, remote device, etc
- mars0.1b currently in ALPHA stage



MARS Future Plans

1&1

Automatic
load balancing

TBD
Separate implementation
or libvirt / Openstack /
Kubernetes plugins ... ?

Virtual LVM-like
Storage + VM pools

WIP
1&1 clustermanager
cm3 and/or systemd
and/or libvirt plugin ...?

Physically
sharded pools

Done
MARS instead
of DRBD

Collaboration sought

=> Opportunities for other OpenSource projects!



Appendix

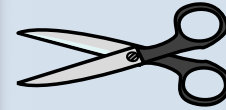


Replication at Block Level vs FS Level

Kernelspace

Userspace
Application Layer

Apache, PHP,
Mail Queues, etc

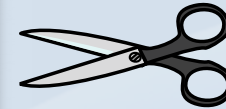


Potential Cut Point **A**
for Distributed System

~ 25 Operation Types
~ 100.000 Ops / s

Filesystem Layer

xfs, ext4, btrfs, zfs, ...
vs nfs, Ceph, Swift, ...



Potential Cut Point **B**
for Distributed System

DSM = Distributed Shared Memory
=> Cache Coherence Problem!

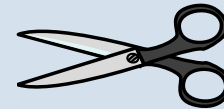
Caching Layer

Page Cache,
dentry Cache, ...
1:100 reduction

2 Operation Types (r/w)
~ 1.000 Ops / s

Block Layer

LVM,
DRBD / MARS



Potential Cut Point **C**
for Distributed System

++ replication of VMs for free!

Hardware

Hardware-RAID,
BBU, ...

DRBD+proxy (proprietary)

Application area:

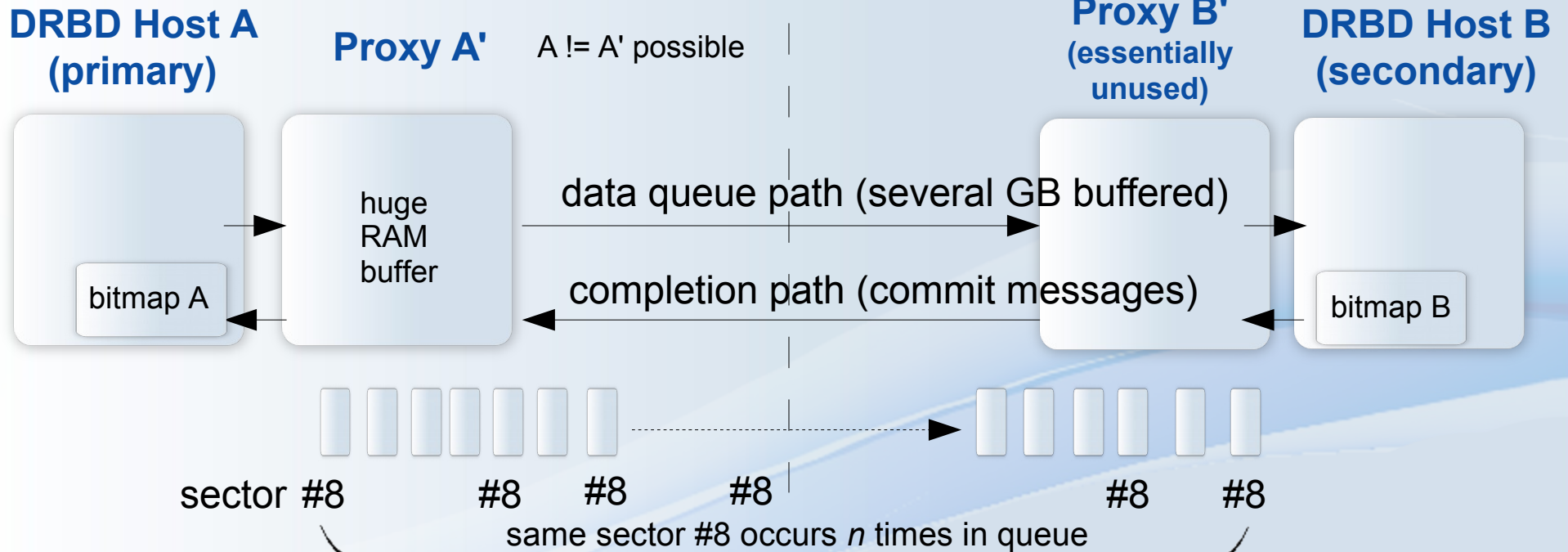
- Distances: any
- Asynchronously
 - **Buffering in RAM**
- Unreliable network leads to **frequent re-syncs**
 - RAM buffer gets lost
 - at cost of actuality
- **Long** inconsistencies during re-sync
- Under pressure: **permanent** inconsistency possible
- High memory overhead
- Difficult scaling to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

MARS Light (GPL)

Application area:

- Distances: **any** ($\gg 50$ 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 ≥ 100 GB in `/mars/` for transaction logfiles
 - dedicated spindle(s) recommended
 - RAID with BBU recommended
- Easy scaling to $k > 2$ nodes

DRBD+proxy Architectural Challenge



n times

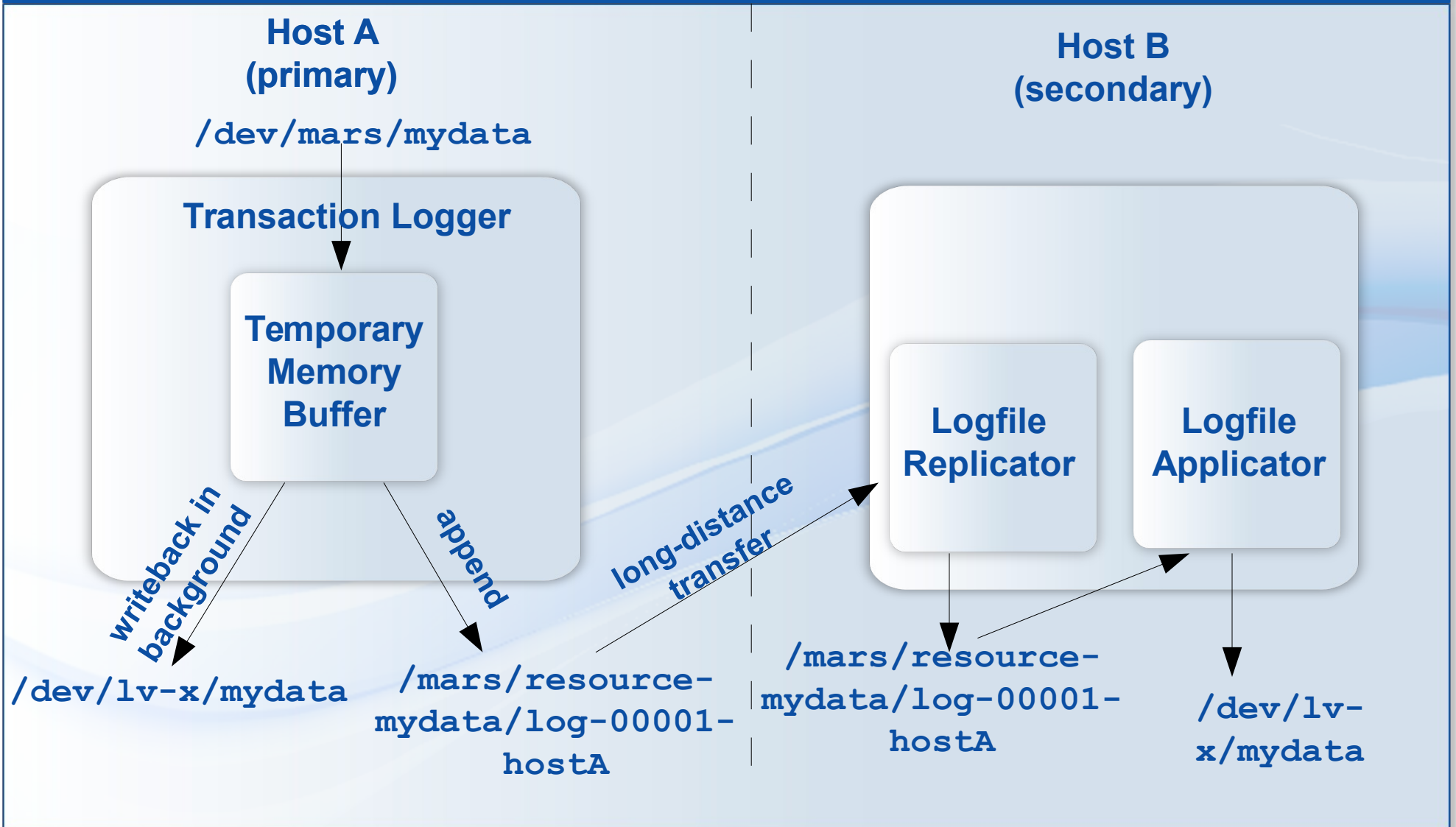
=> need $\log(n)$ bits for counter

=> but DRBD bitmap has only 1 bit/sector

=> workarounds exist, but complicated

(e.g. additional dynamic memory)

MARS Data Flow Principle



Framework Architecture

for MARS + future projects



External Software, Cluster Managers, etc

Userspace Interface `marsadm`

Framework Application Layer
MARS Light, MARS Full, etc

**MARS
Light**

**MARS
Full**

...

Framework Personalities
XIO = eXtended IO \approx AIO

**XIO
bricks**

**future
Strategy
bricks**

**other future
Personalities
and their bricks**

Generic Brick Layer
IOP = Instance Oriented Programming
+ AOP = Aspect Oriented Programming

Generic Bricks

Generic Objects

Generic Aspects

S