Virtual Petabytes Storage Pools
using MARS

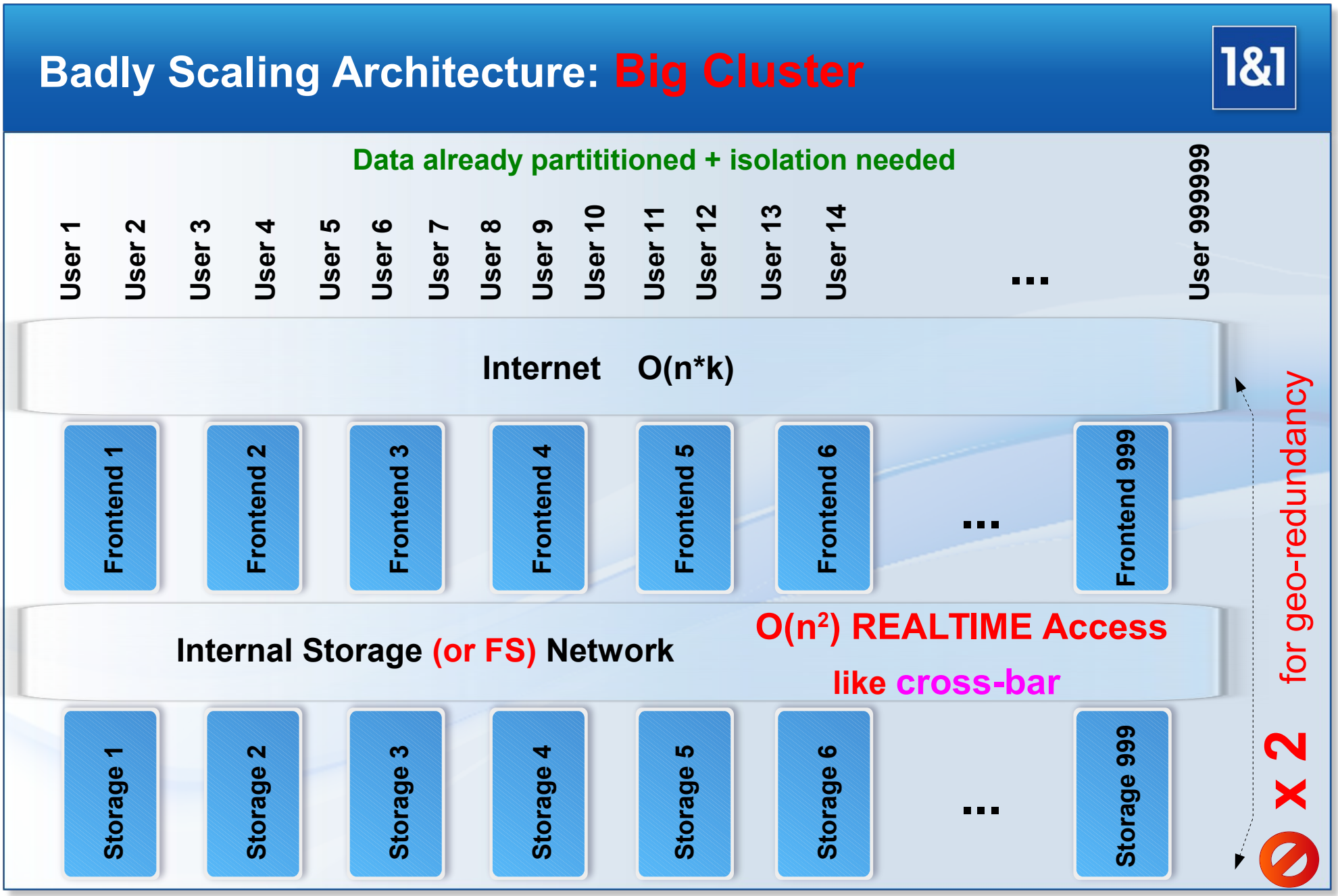
LCA 2018 Presentation by Thomas Schöbel-Theuer

Virtual Petabytes Storage Pools: Agenda

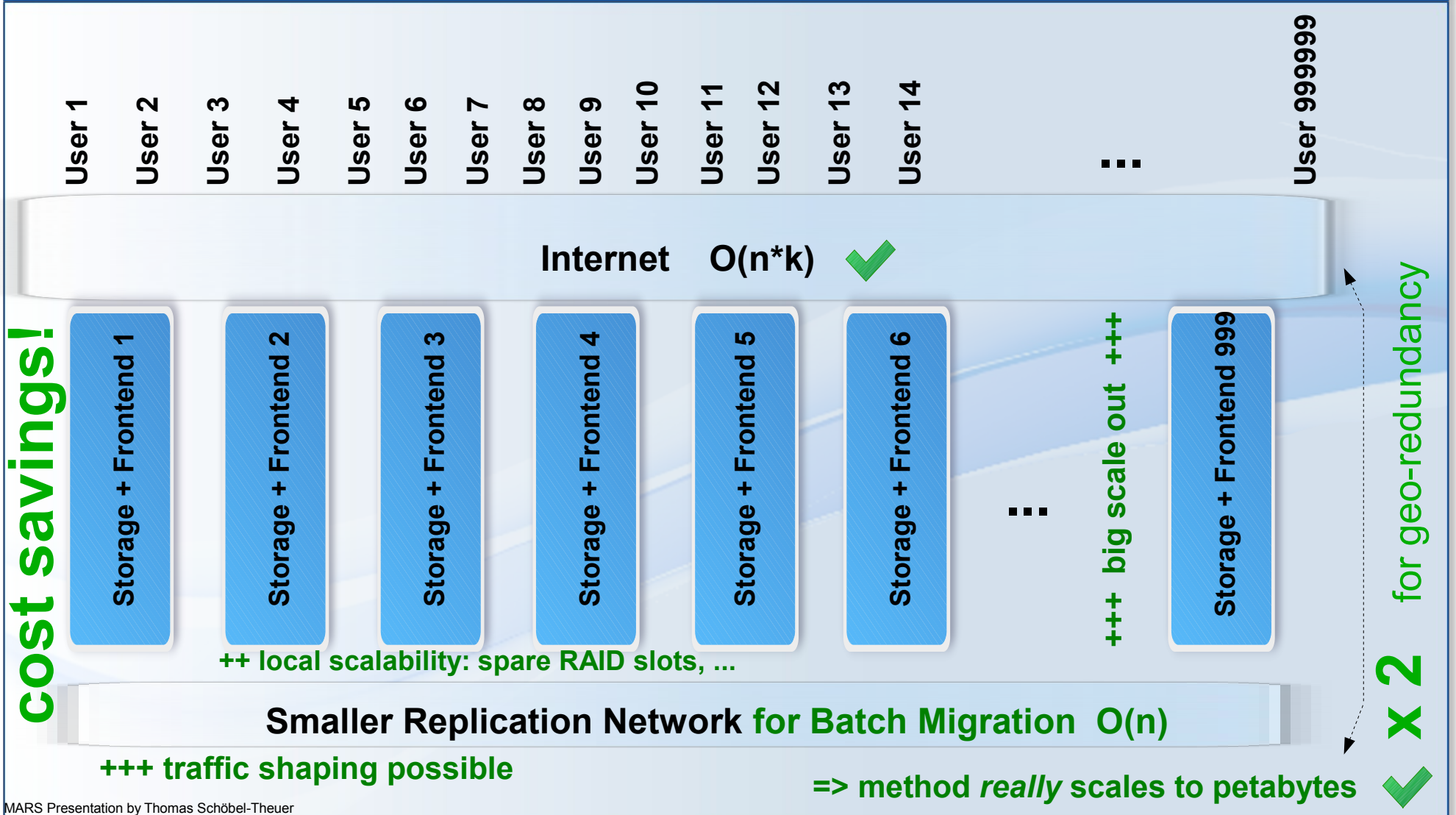
- Storage Architectures → Scalability
- HOWTO Background Migration of LVs

  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



Well-Scaling Architecture: Sharding



HOWTO Background Migration of LVs

HOST A (old) VM is running	→ HOST B (new) has spare space
– lvdisplay /dev/vg/$mydata	–
–	– lvcreate -L $size $mydata
–	– marsadm join-resource $mydata \ /dev/vg/$mydata
– (meanwhile VM is altering data)	– marsadm view: wait for UpToDate
– $vmmanager stop /dev/mars/$mydata	–
–	– marsadm primary $mydata
–	– $vmmanger start /dev/mars/$mydata
– marsadm leave-resource $mydata	–
– lvremove $mydata	–

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

Example: tetris.sh in github.com/schoebel/mars/contrib/

Guidelines: Use Cases

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-complicant FS needed

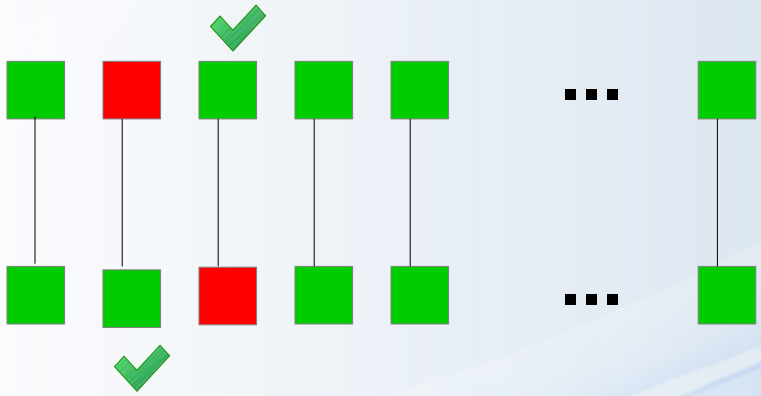
Grey Zones

- 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

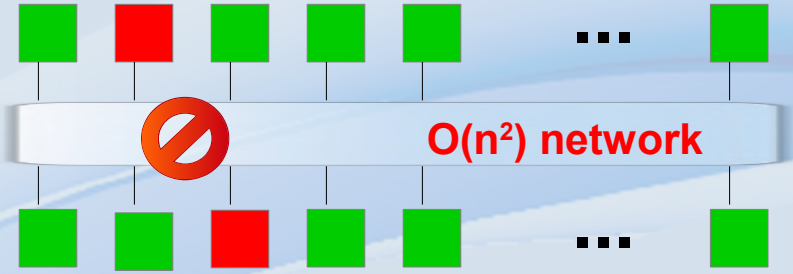


same n

=> 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, ...

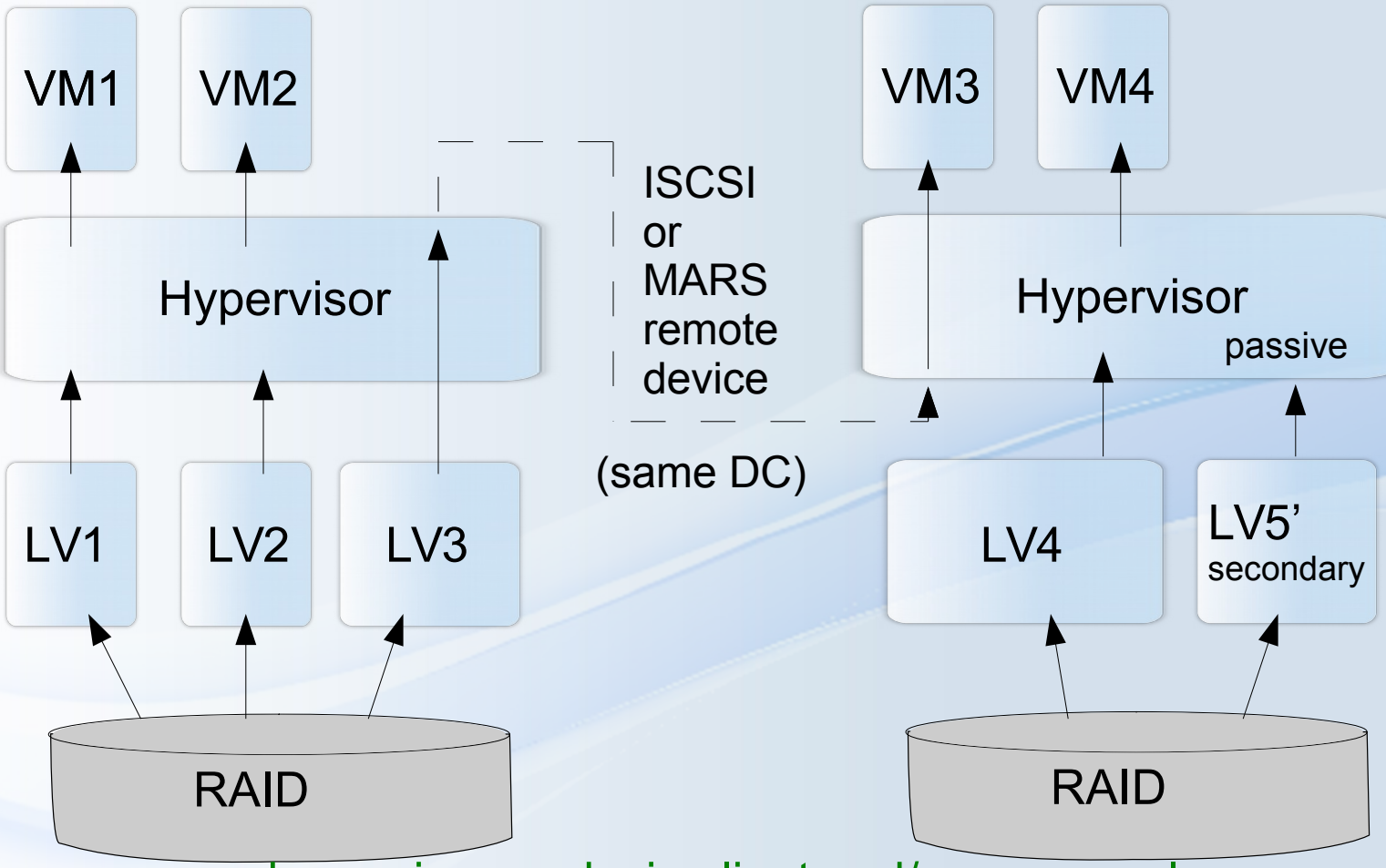


k=2 replicas not enough
=> INCIDENT because objects are randomly distributed across whole cluster

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

need k >= 3 replicas here

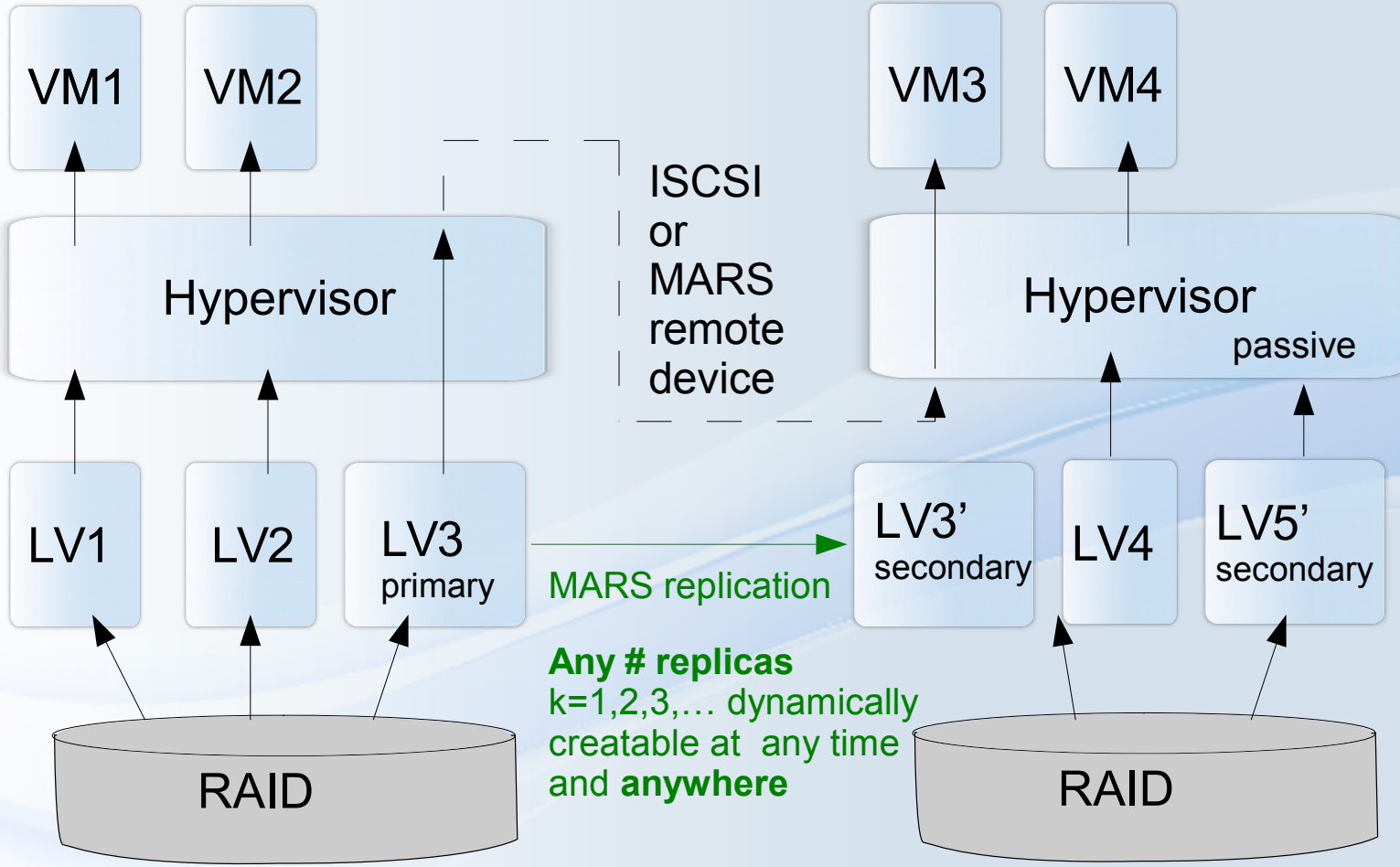
Flexible MARS Sharding + Cluster-on-Demand



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)

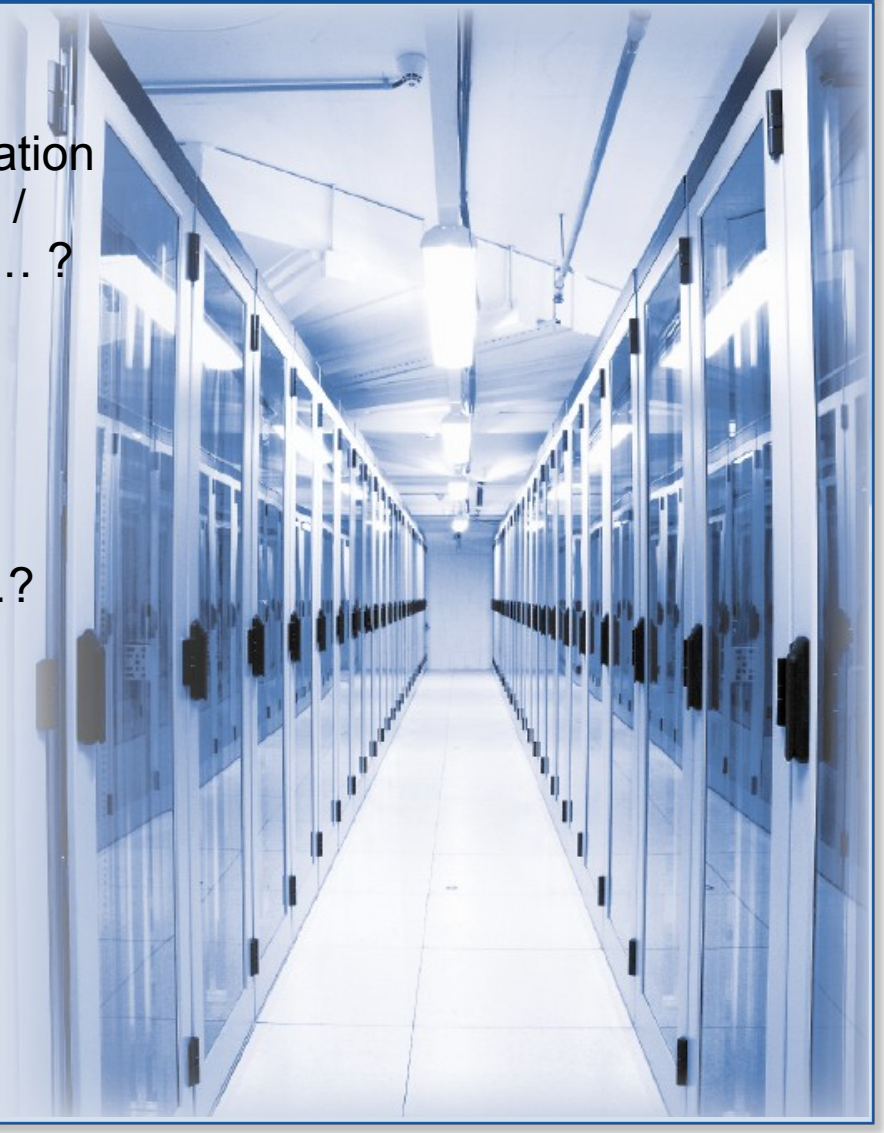
  > 2x12 petabyte total

  ~ 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

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

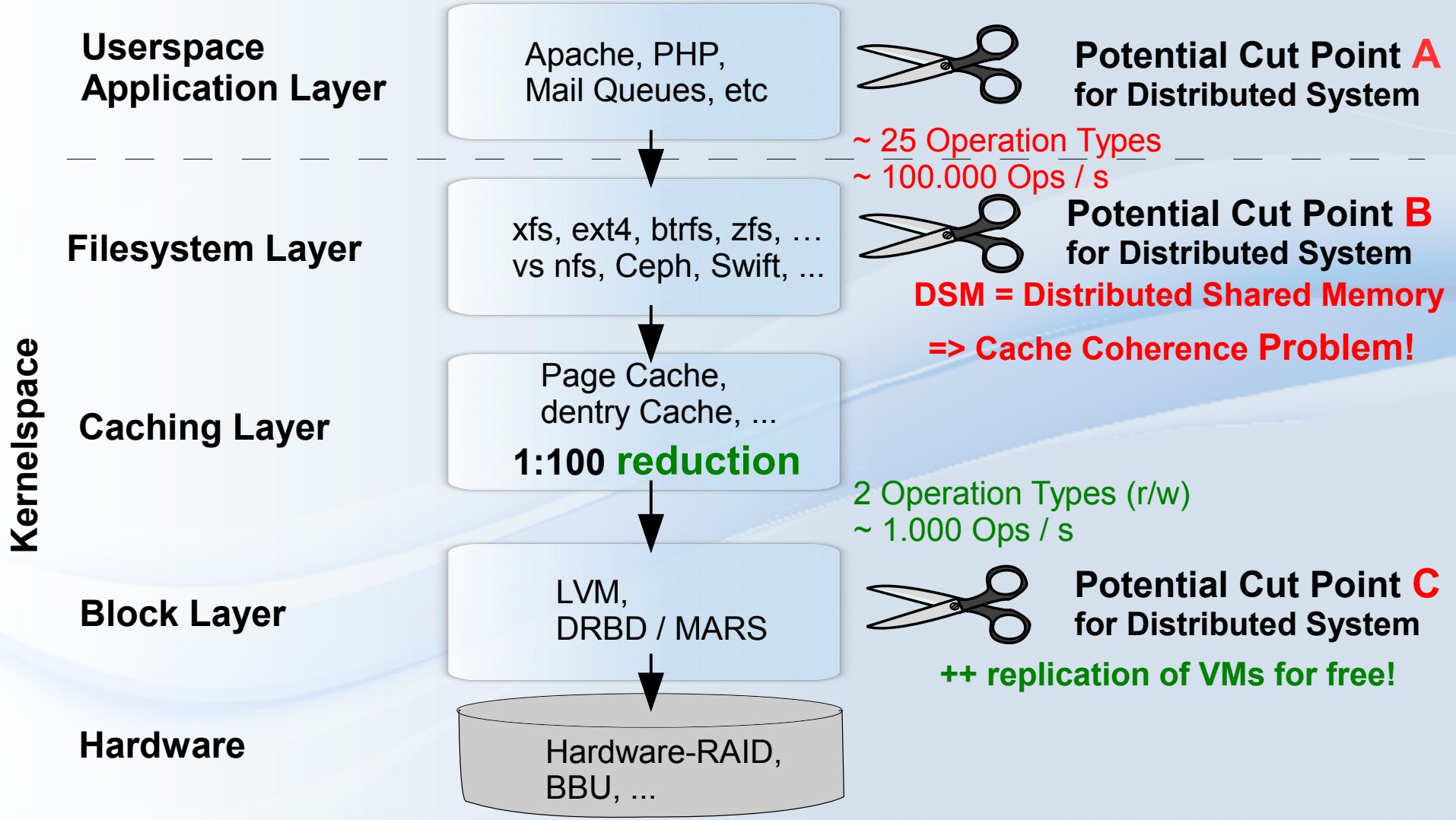


Collaboration sought

=> Opportunities for other OpenSource projects!

Appendix

Replication at Block Level vs FS Level



Use Cases DRBD+proxy vs MARS Light

DRBD+proxy
(proprietary)

Application area:

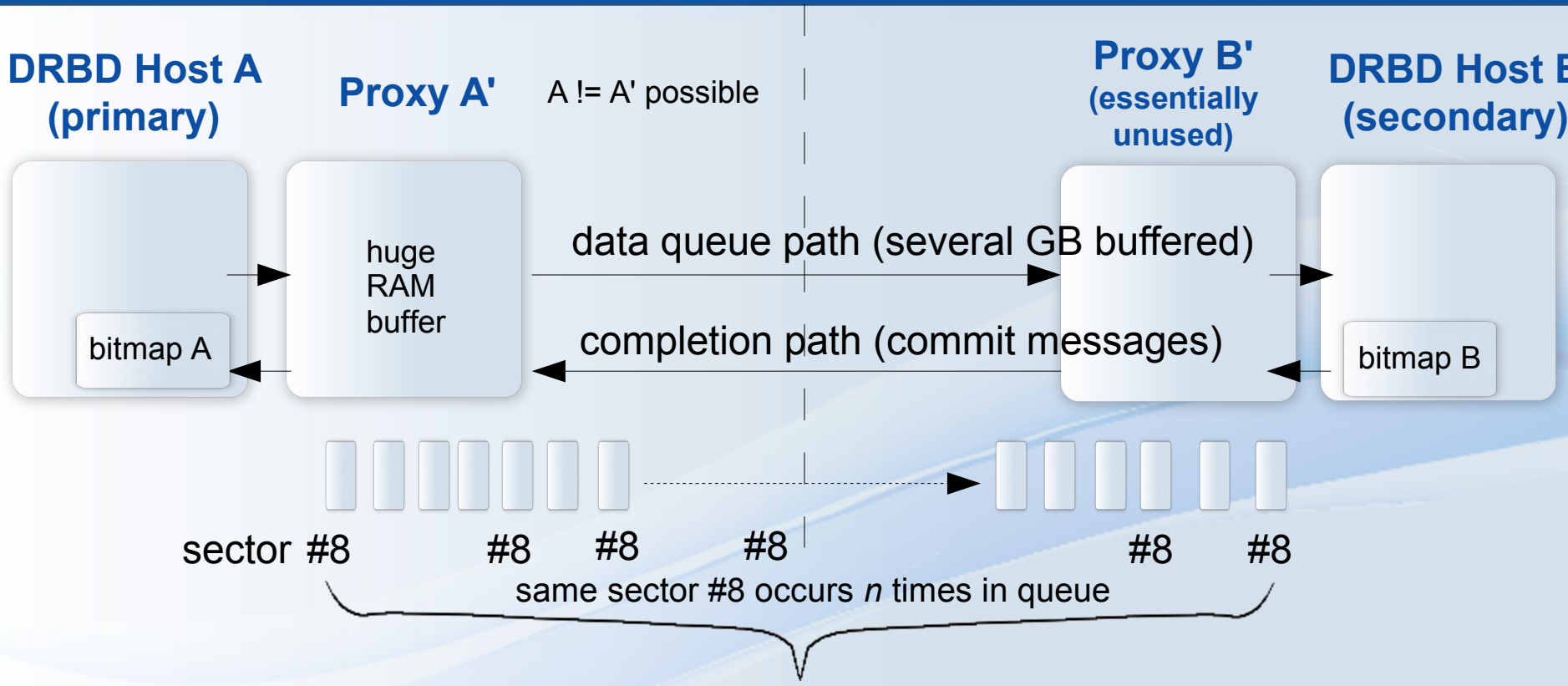
- Distances: any
- Aynchronously
  - **Buffering in RAM**
- Unreliable network leads to **frequent re-syncs**
  - RAM buffer gets lost
  - at cost of actuality
- **Long** inconsistencies during re-sync
- Under pressure: **permanent** inconsistency possible
- High memory overhead
- Difficult scaling to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

MARS Light
(GPL)

Application area:

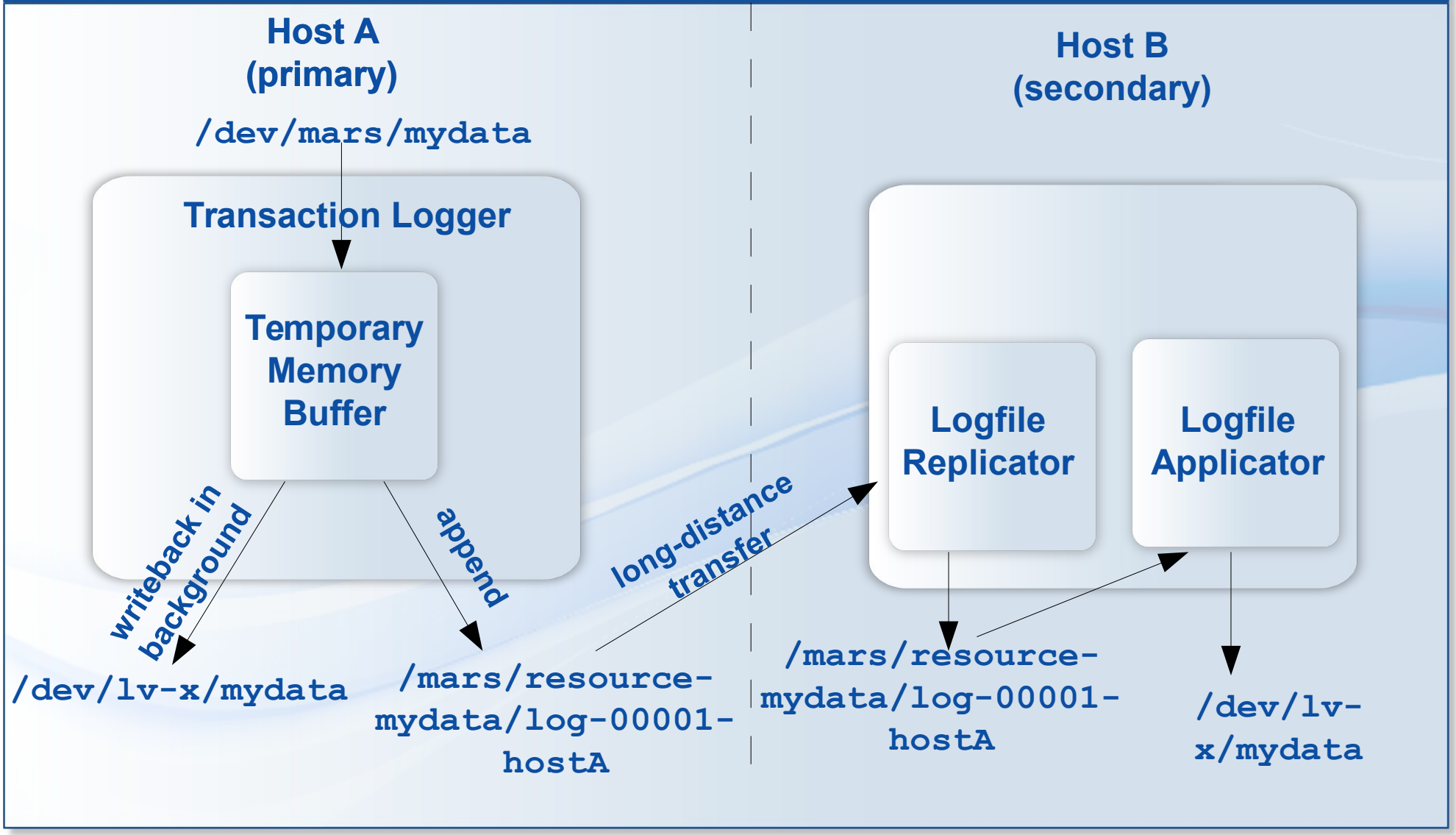
- Distances: **any** (>>50 km)
- Asynchronously
  - near-synchronous modes in preparation
- Tolerates **unreliable network**
- Anytime consistency
  - no re-sync
- Under pressure: no inconsistency
  - possibly at cost of actuality
- Needs >= 100GB in `/mars/` for transaction logfiles
  - dedicated spindle(s) 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