Petabytes Data Migration and Load Balancing

with Football + MARS on Enterprise-Critical Data

LCA 2019 Presentation by Thomas Schöbel-Theuer

Container Football: Agenda

New method for load balancing

- Motivation: data growth > 20% / year
- HOWTO Container Football = Background Migration of LVs

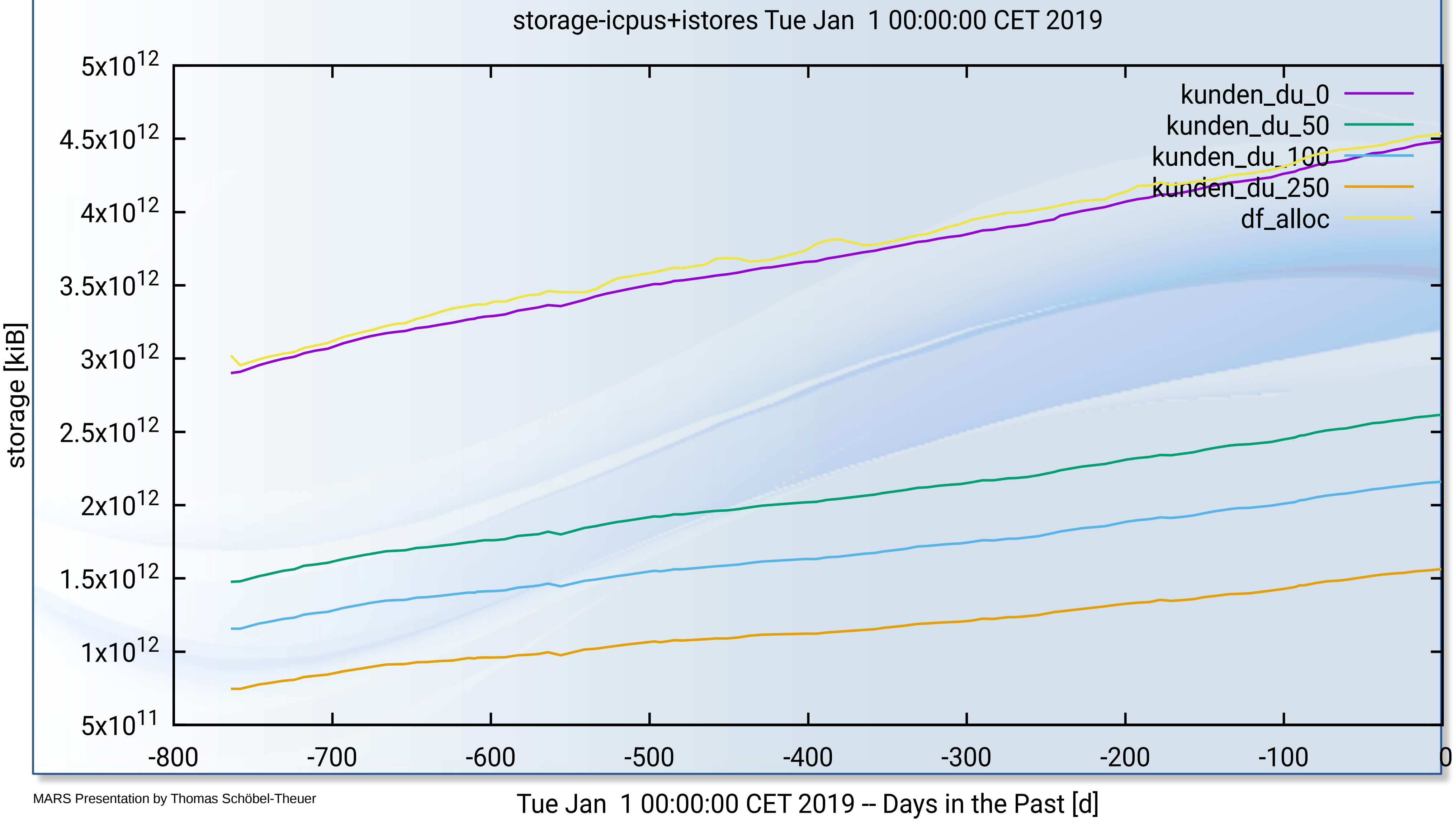
  e.g. for load balancing, HW lifecycle, etc
- The Football Automation Project
- Current Status / Future Plans

Motivation: Growth at 1&1 ShaHoLin = Shared Hosting Linux

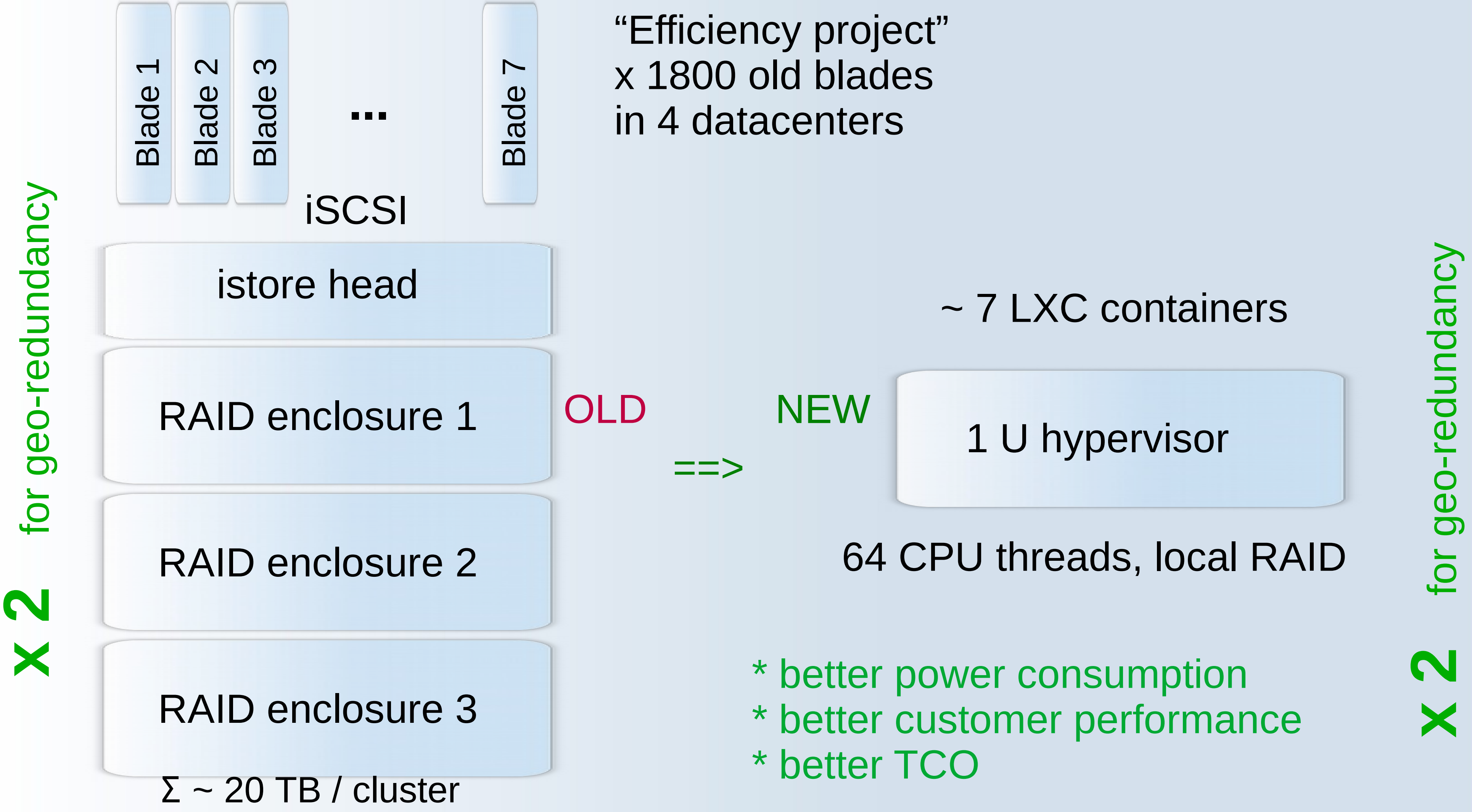
- ~ 9 millions of customer home directories
- ~ 10 billions of inodes
- > 4.5 petabytes *allocated* in ~ 2700 xfs instances,

  LVM ~ 8 PB x 2 for geo-redundancy via MARS
- Growth rate ~ 21 % / year

Motivation: Growth at 1&1 ShaHoLin = Shared Hosting Linux



Hardware Lifecycle



Blade 1

Blade 2

Blade 3

...

Blade 7

iSCSI

istore head

RAID enclosure 1

RAID enclosure 2

RAID enclosure 3

Σ ~ 20 TB / cluster

x 2 for geo-redundancy

“Efficiency project”
x 1800 old blades
in 4 datacenters

OLD ==> NEW

~ 7 LXC containers

1 U hypervisor

64 CPU threads, local RAID

x 2 for geo-redundancy

* better power consumption
* better customer performance
* better TCO

Football: Basic Operations

- **migrate**: move pairs of LVs to new pairs of hypervisors

```
./football.sh migrate infong4711 cluster1234
```

- **shrink**: use local rsync for in-place resizing (downtime)

```
./football.sh shrink infong4711 75%
```

- **expand**: online lvresize + marsadm resize + xfs_growfs

```
./football.sh expand infong4712 75%
```

- several combined operations, e.g. migrate+shrink (less network traffic)

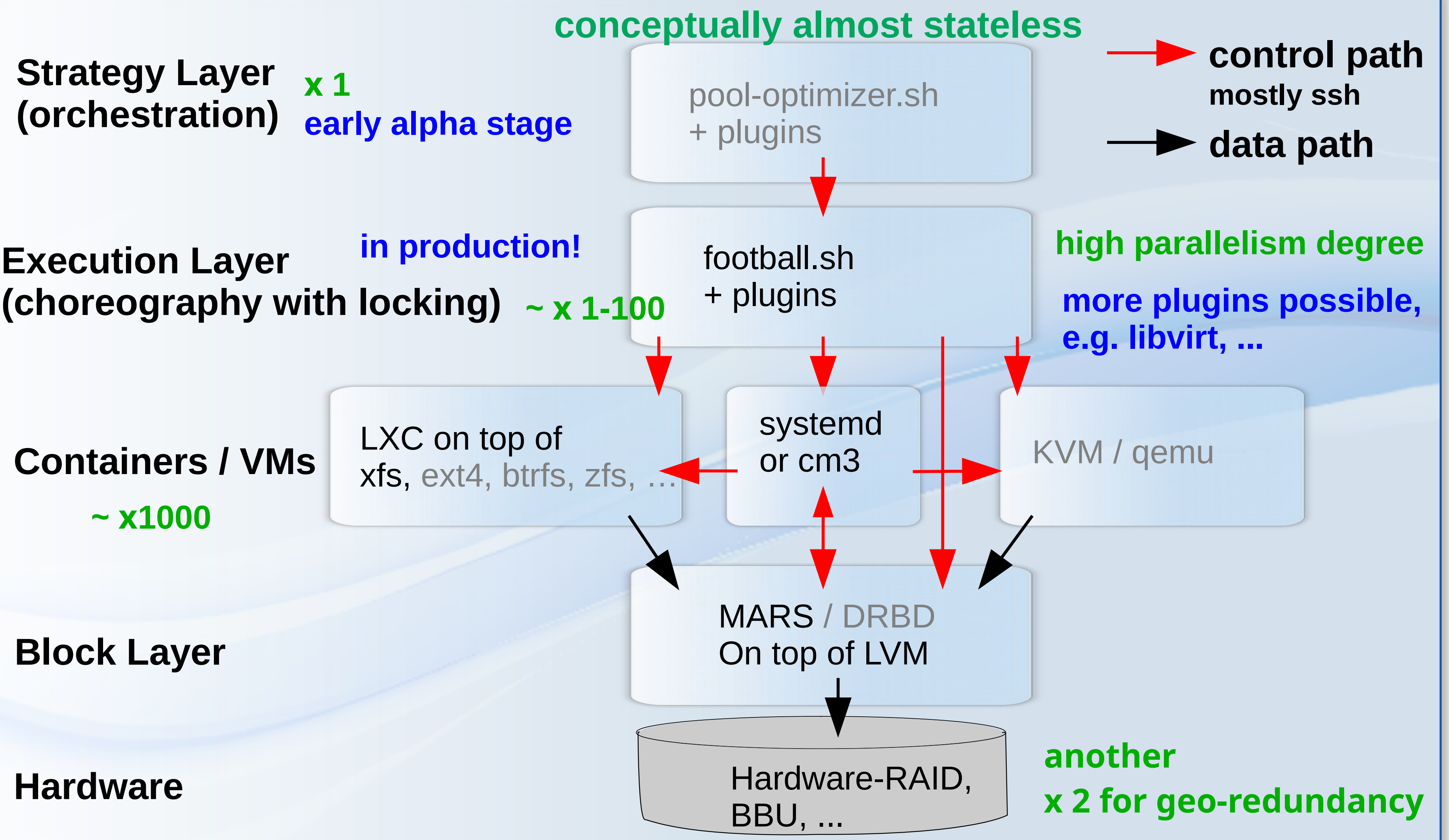
HOWTO Container Football = Background Migration of LVs

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

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

Toolset: football.sh in github.com/schoebel/football

Football Architecture (grey = not yet implemented)



Football Current Status

- GPL with lots of plugins, some generic, some 1&1-specific
  - about 2/3 of code is generic
  - plugins/football-basic.sh uses `systemd`
  - https://github.com/schoebel/football
  - https://github.com/schoebel/mars
- Multiple operations:
  - migrate $vm $target_cluster
    - low downtime (seconds to few minutes)
  - shrink $vm $target_percent
    - uses local incremental rsync, more downtime
  - expand $vm $target_percent
    - online, no downtime
- In production at internal Efficiency project
  - get rid of old hardware
  - Concentrate ~ 7 LXC containers on 1 hypervisor
  - currently >50 „kicks“ per week
    - limited by hardware deployment speed
    - Proprietary Planner (for HW lifecycle)
  - Almost finished: ~70% of ~1800 blades already migrated (mid of January 2019) and mostly shrunk

Sponsoring (MARS + Football)

- Best for > 1 PiB of enterprise-critical data
  - Example: ShaHoLin (slide3)
  - More plugins in future, e.g. for KVM, ...
- Future pool-optimizer will deliver similar functionality than **Kubernetes**
  - but on stateful storage + containers instead of stateless Docker containers
  - State is in the storage and in the machines, but not in orchestration
- Long-term perspective
  - MARS is largely complementary to DRBD
  - Geo-redundancy with OpenSource components
  - distances > 50km possible, tolerates flaky replication networks
  - **Price / performance** better than anything else (see mars-manual.pdf)
  - Architectural **reliability** better than BigCluster with cheaper hw + network!
- ask me: decades of experience with enterprise-critical data and long-distance replication

Appendix

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)

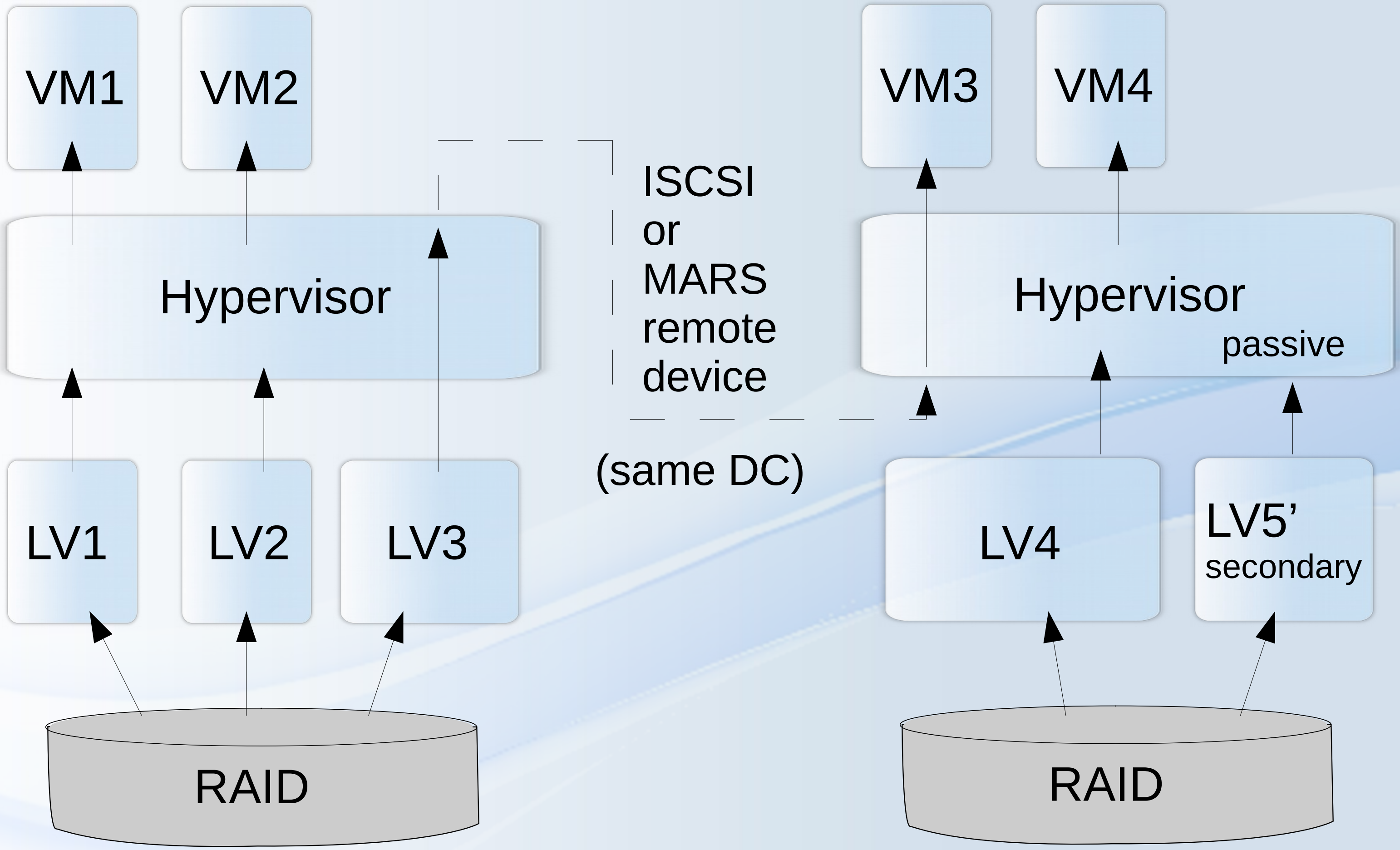
  > 2x12 petabyte total

  ~ 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

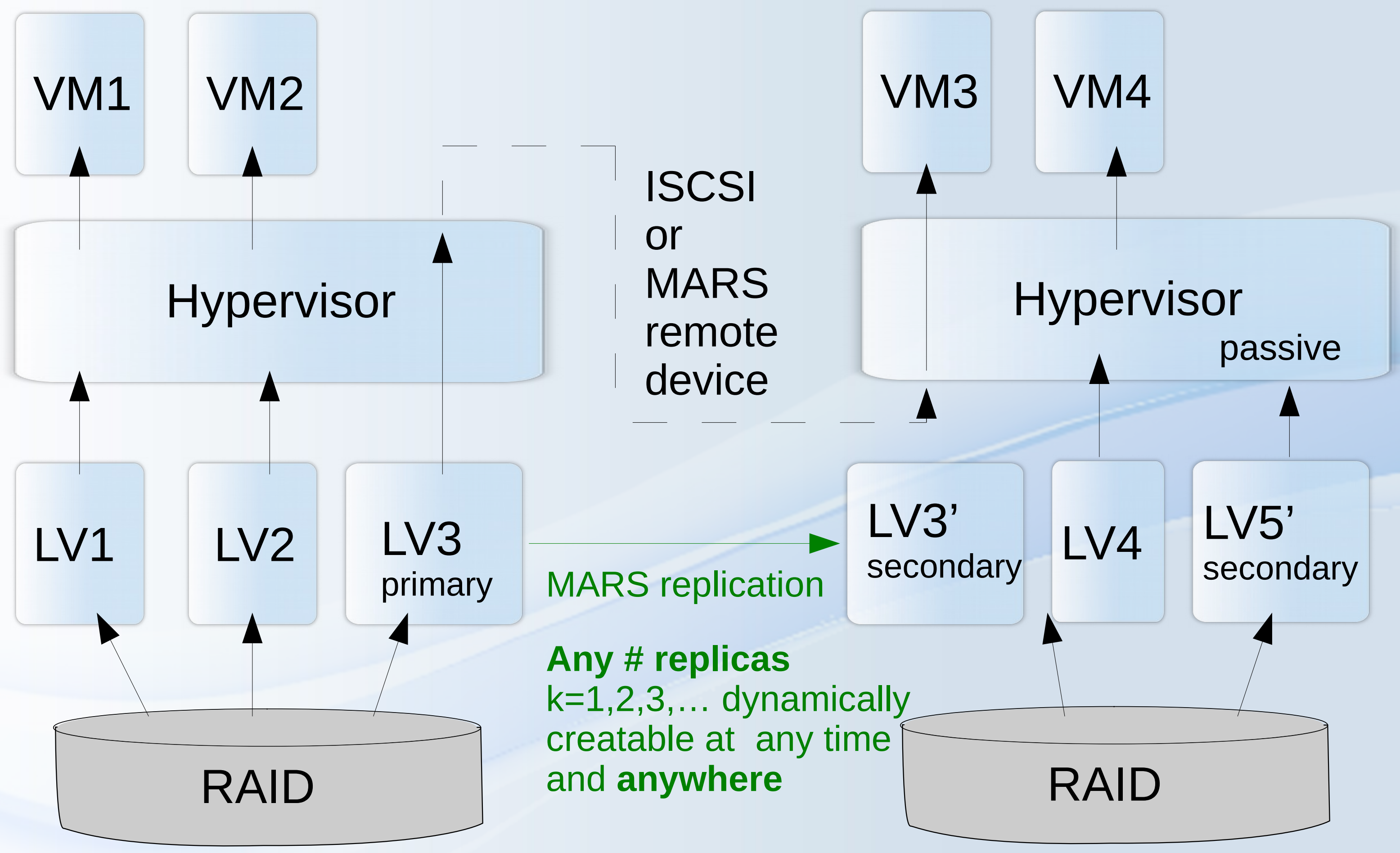
Flexible MARS Sharding + Cluster-on-Demand



any hypervisor works in client and/or server role

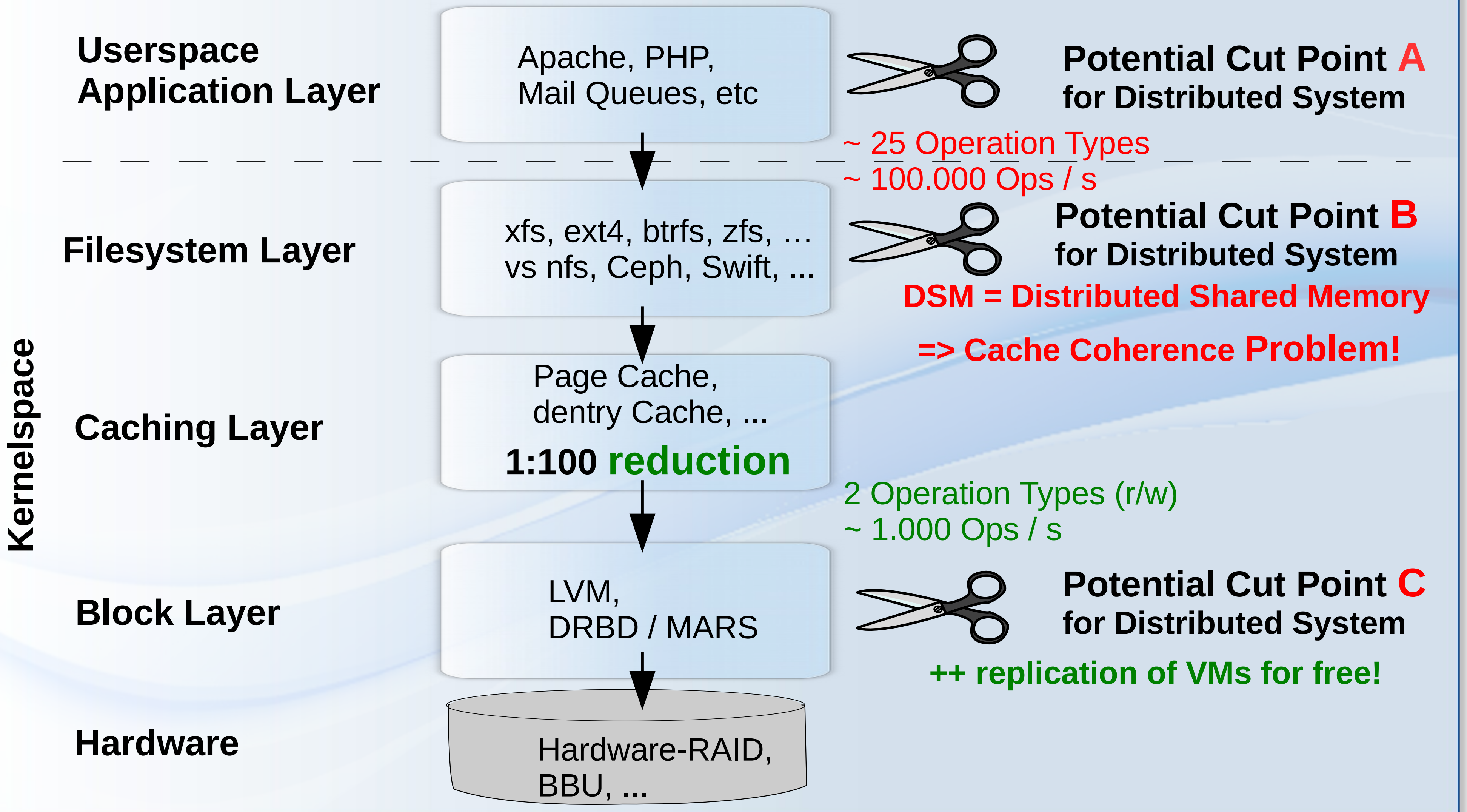
and preferably **locally** at the same time

Flexible MARS Background Migration



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

Replication at Block Level vs FS Level



Userspace
Application Layer

Apache, PHP,
Mail Queues, etc

Potential Cut Point A
for Distributed System

~ 25 Operation Types
~ 100.000 Ops / s

Kernelspace

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, ...

Use Cases DRBD+proxy vs MARS Light

DRBD+proxy
(proprietary)

Application area:

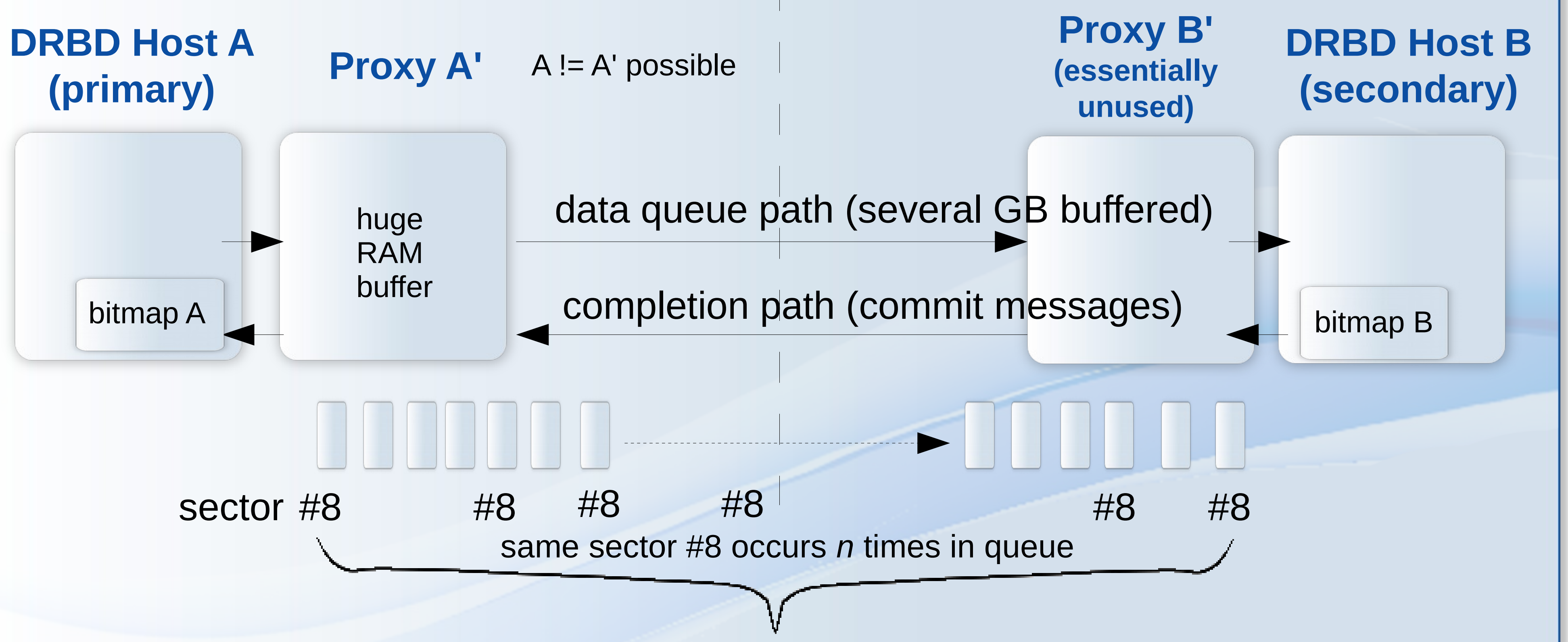
- Distances: any
- Aynchronously
  - **Buffering in RAM**
- Unreliable network leads to **frequent re-syncs**
  - RAM buffer gets lost
  - at cost of actuality
- **Long** inconsistencies during re-sync
- Under pressure: **permanent** inconsistency possible
- High memory overhead
- Difficult scaling to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

MARS Light
(GPL)

Application area:

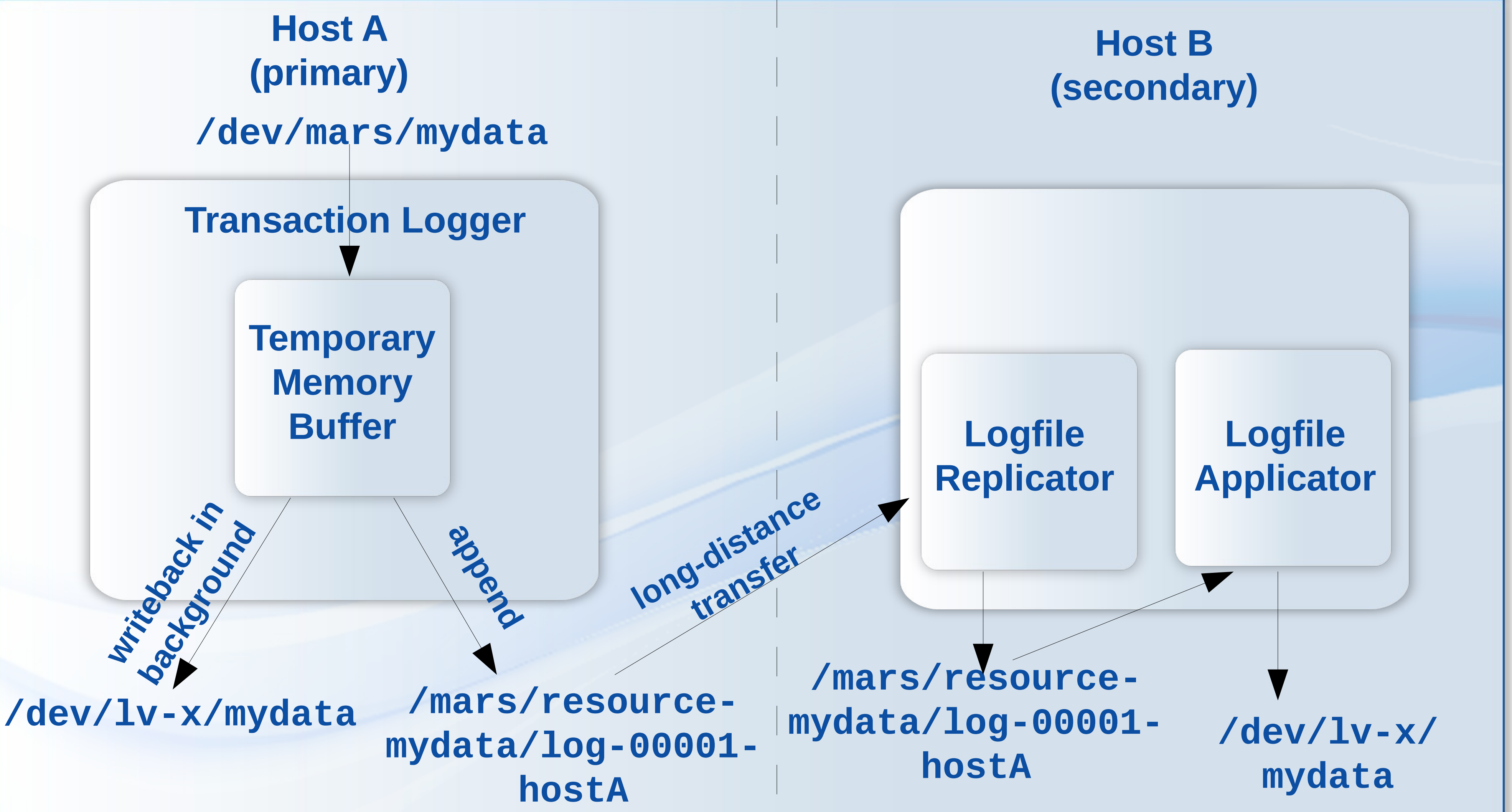
- Distances: **any** (>>50 km)
- Asynchronously
  - near-synchronous modes in preparation
- Tolerates **unreliable network**
- Anytime consistency
  - no re-sync
- Under pressure: no inconsistency
  - possibly at cost of actuality
- Needs >= 100GB in `/mars/` for transaction logfiles
  - dedicated spindle(s)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

1&1

Framework Application Layer

MARS Light, MARS Full, etc

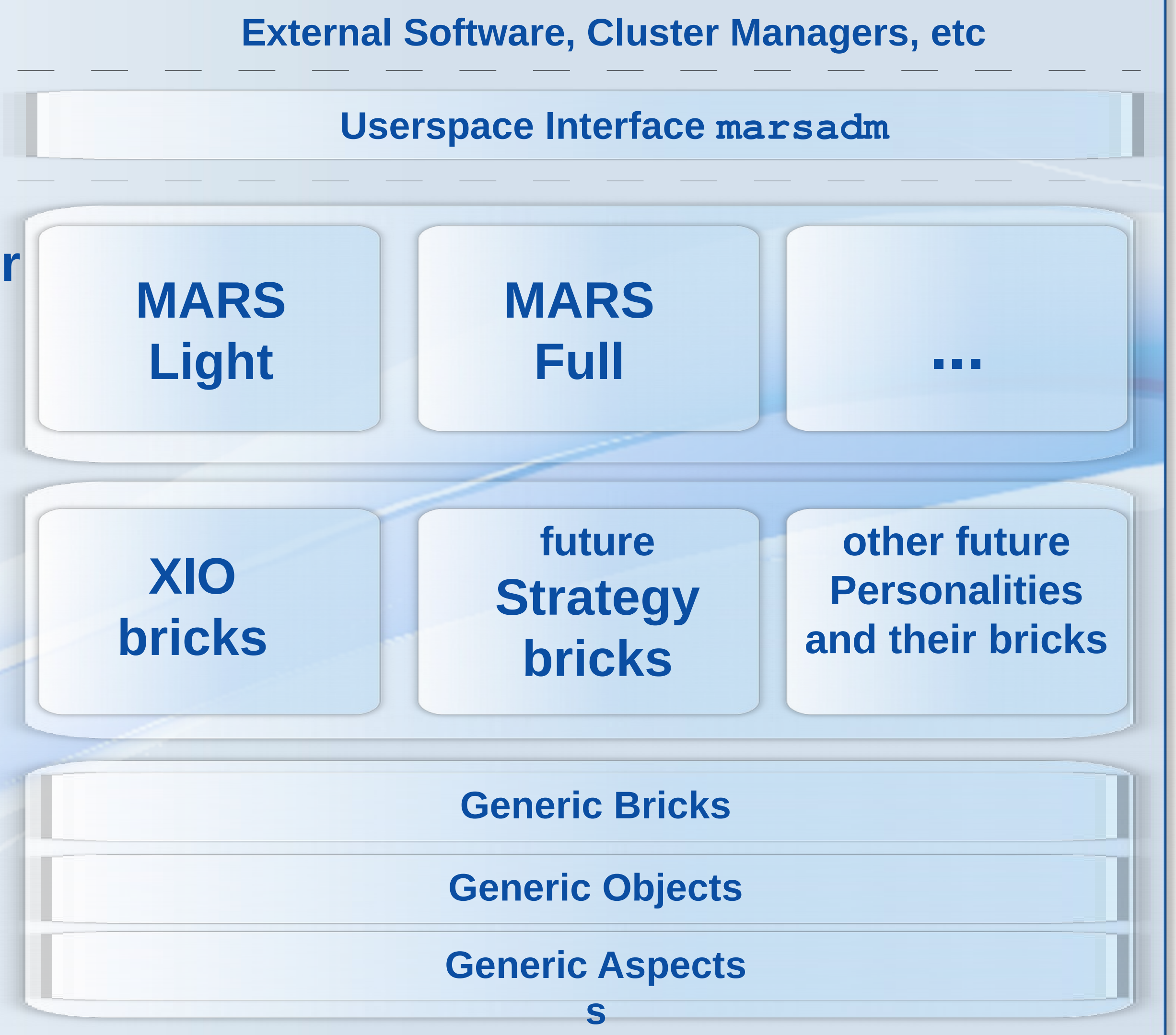
Framework Personalities

XIO = eXtended IO ≈ AIO

Generic Brick Layer

IOP = Instance Oriented Programming

+ AOP = Aspect Oriented Programming



External Software, Cluster Managers, etc

Userspace Interface `marsadm`

MARS Light

MARS Full

...

XIO bricks

future Strategy bricks

other future Personalities and their bricks

Generic Bricks

Generic Objects

Generic Aspects

s