Cost-Effective Virtual Petabytes Storage Pools

using MARS

1&1

LCA 2018 Presentation by Thomas Schöbel-Theuer

Virtual Petabytes Storage Pools: Agenda

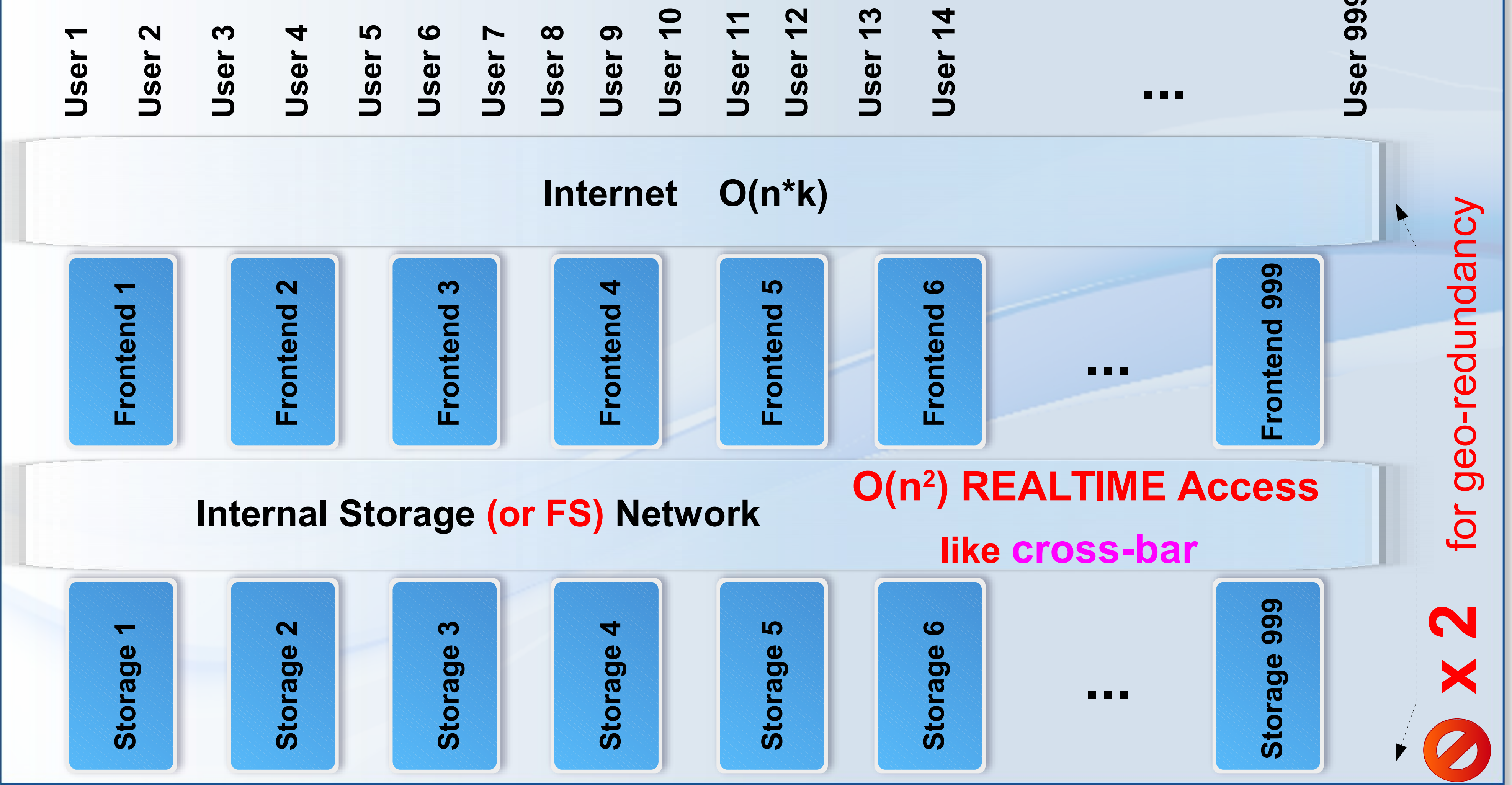
1&1

- Storage Architectures → Scalability && Costs
- 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 partititioned + 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)

Frontend 1 Frontend 2 Frontend 3 Frontend 4 Frontend 5 Frontend 6 ... Frontend 999

Internal Storage (or FS) Network

$O(n^2)$ REALTIME Access

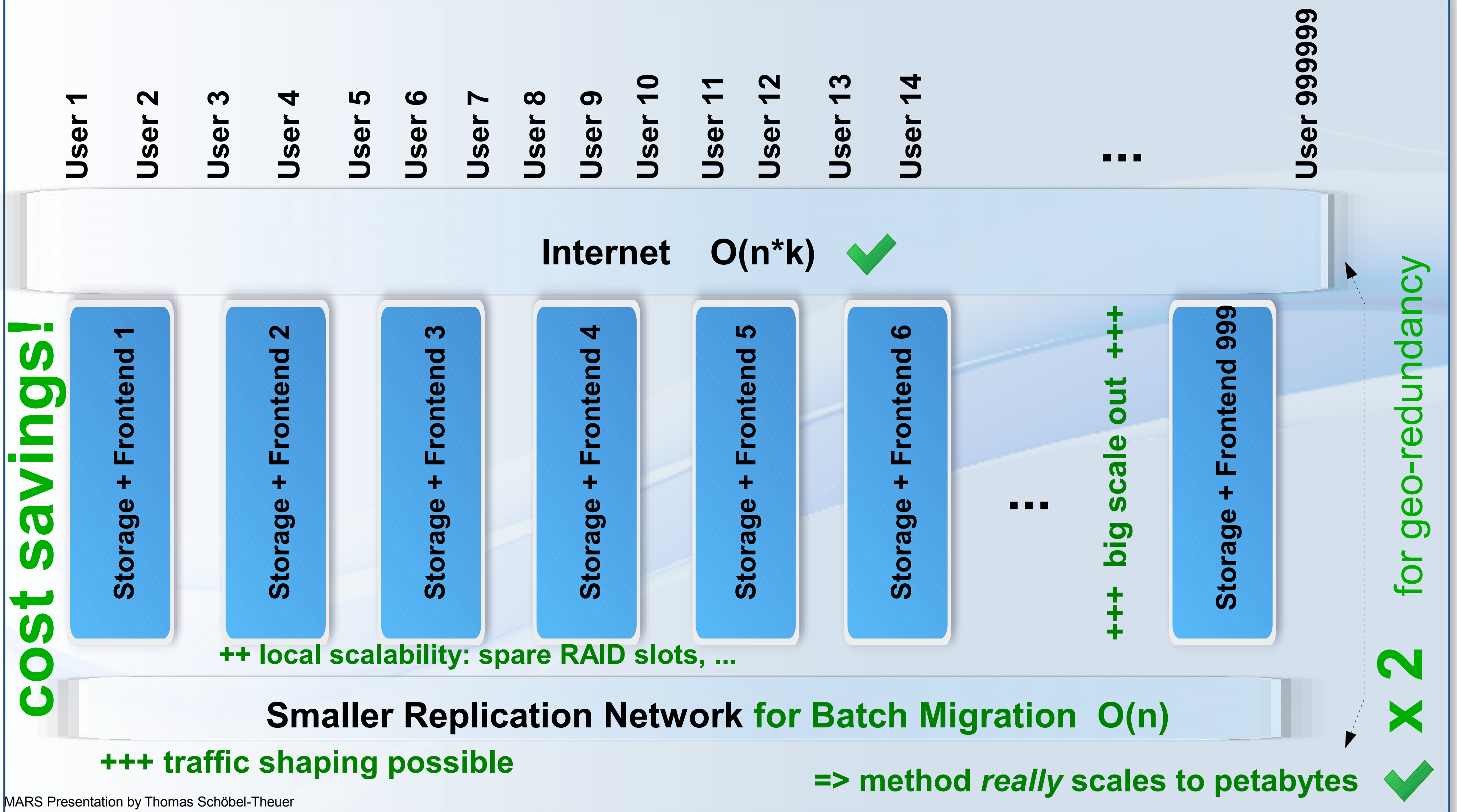
like cross-bar

Storage 1 Storage 2 Storage 3 Storage 4 Storage 5 Storage 6 ... Storage 999

for geo-redundancy

x 2

Well-Scaling Architecture: Sharding



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) ✓

cost savings!

Storage + Frontend 1

Storage + Frontend 2

Storage + Frontend 3

Storage + Frontend 4

Storage + Frontend 5

Storage + Frontend 6

...

+++ big scale out +++

Storage + Frontend 999

for geo-redundancy

x 2

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

Smaller Replication Network for Batch Migration O(n)

+++ traffic shaping possible

=> method *really* scales to petabytes ✓

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

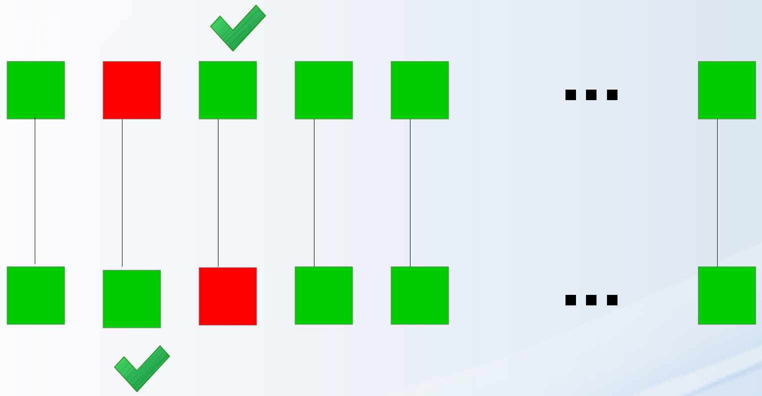
Grey Zones

- 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

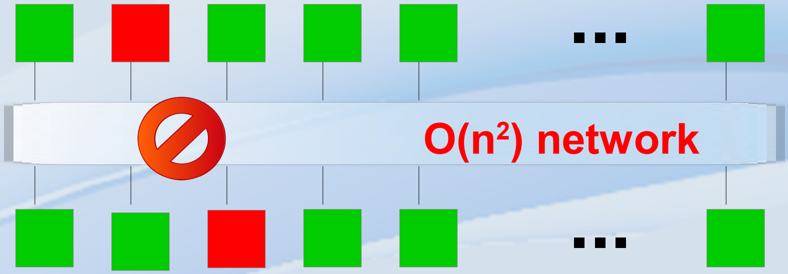


same n

=> 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

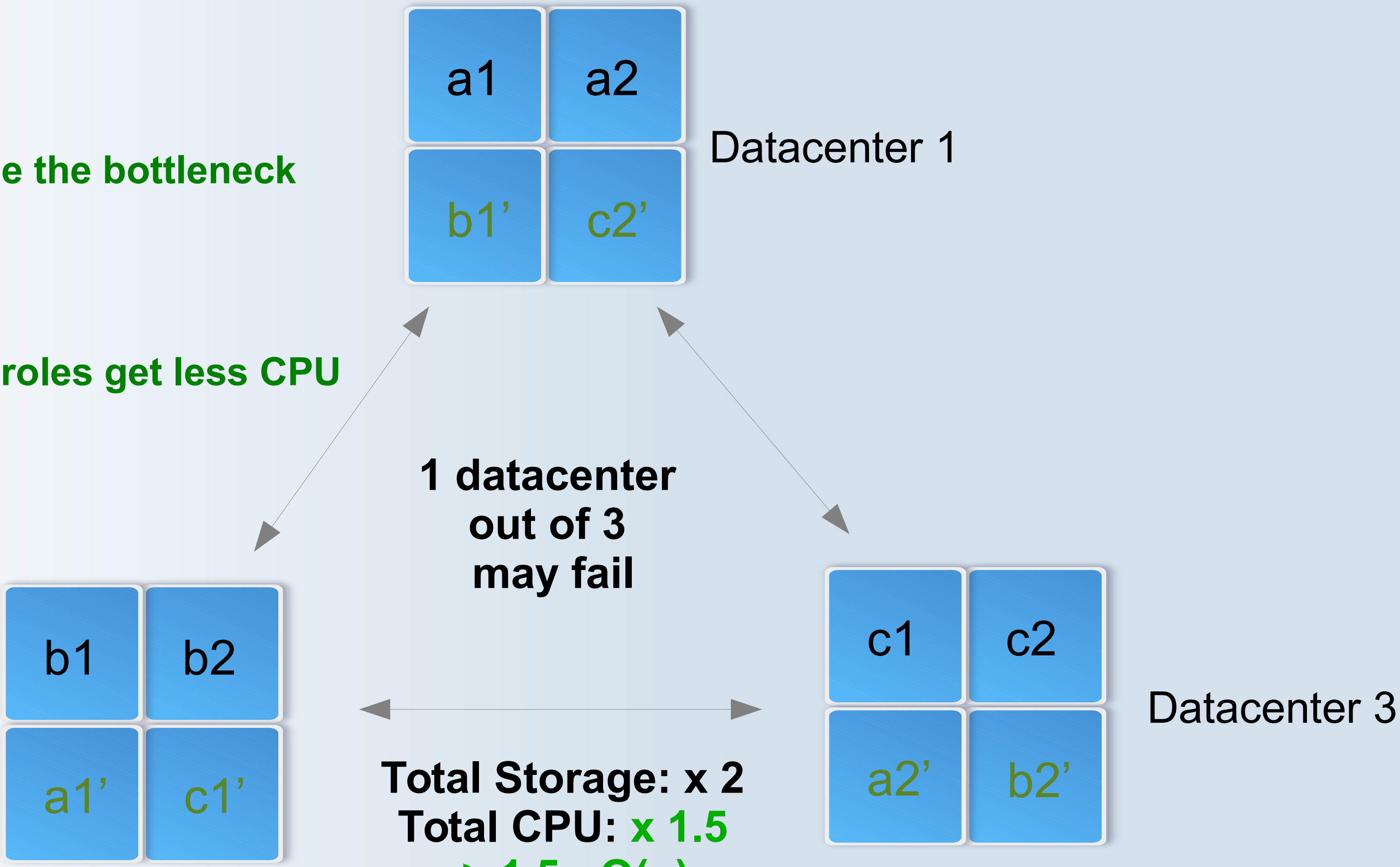
Higher probability for hitting *any* 2 nodes, then O(n) clients affected
=> much higher total downtime

need k >= 3 replicas here

Costs: Geo-Redundancy even Cheaper

Precondition:
CPU must not be the bottleneck

Idea: passive LV roles get less CPU



a1	a2
b1’	c2’

Datacenter 1

Datacenter 2

b1	b2
a1’	c1’

c1	c2
a2’	b2’

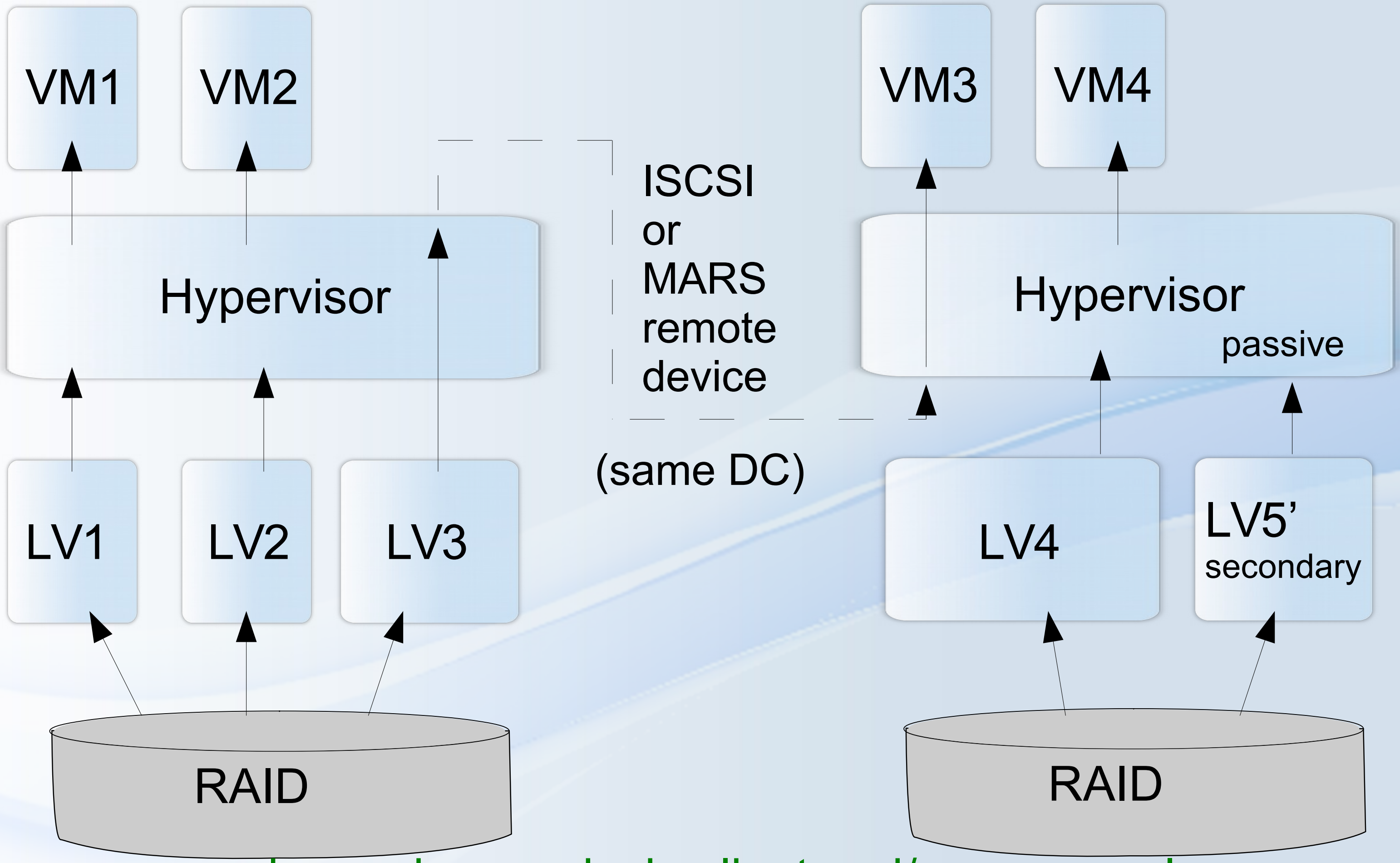
Datacenter 3

1 datacenter
out of 3
may fail

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

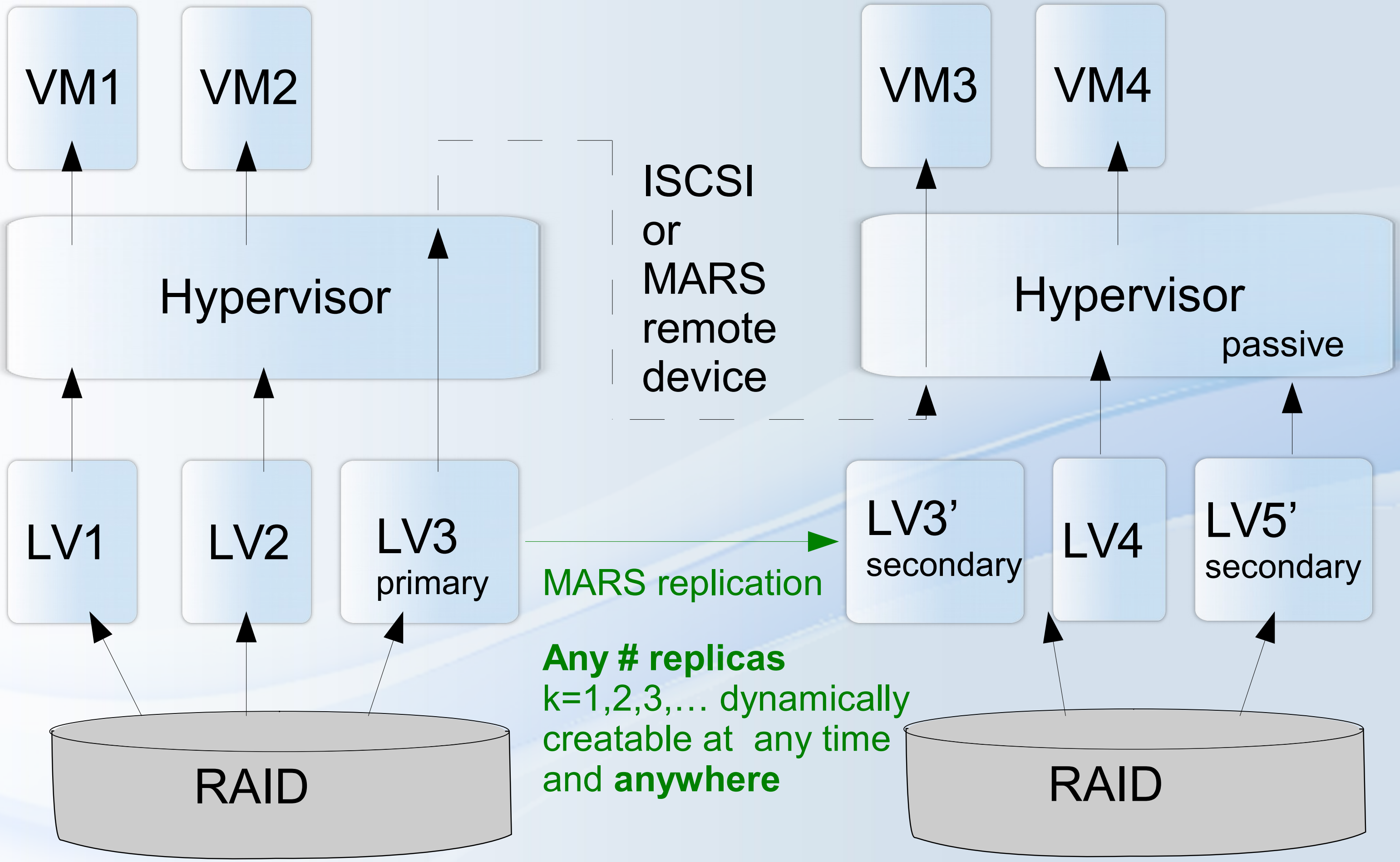
HOWTO flexible CPU assignment => next slide

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



MARS replication

Any # replicas
k=1,2,3,… dynamically
creatable at any time
and **anywhere**

=> 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)

  > 2x12 petabyte total

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

  up to 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

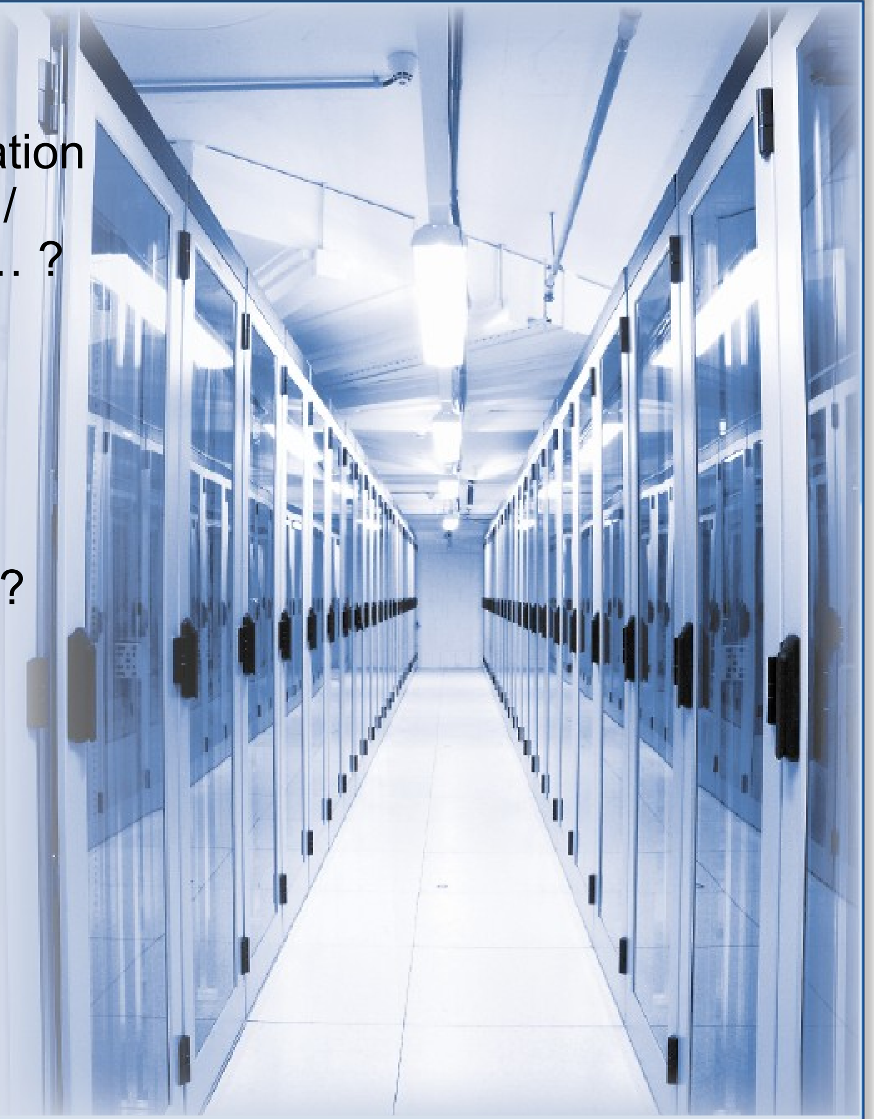
TBD
Separate implementation
or libvirt / Openstack /
Kubernetes plugins … ?

Virtual LVM-like
Storage + VM pools

WIP `tetris.sh` +
1&1 clustermanager
`cm3` and/or systemd
and/or libvirt plugin …?

Physically
sharded pools

Done
MARS instead
of DRBD

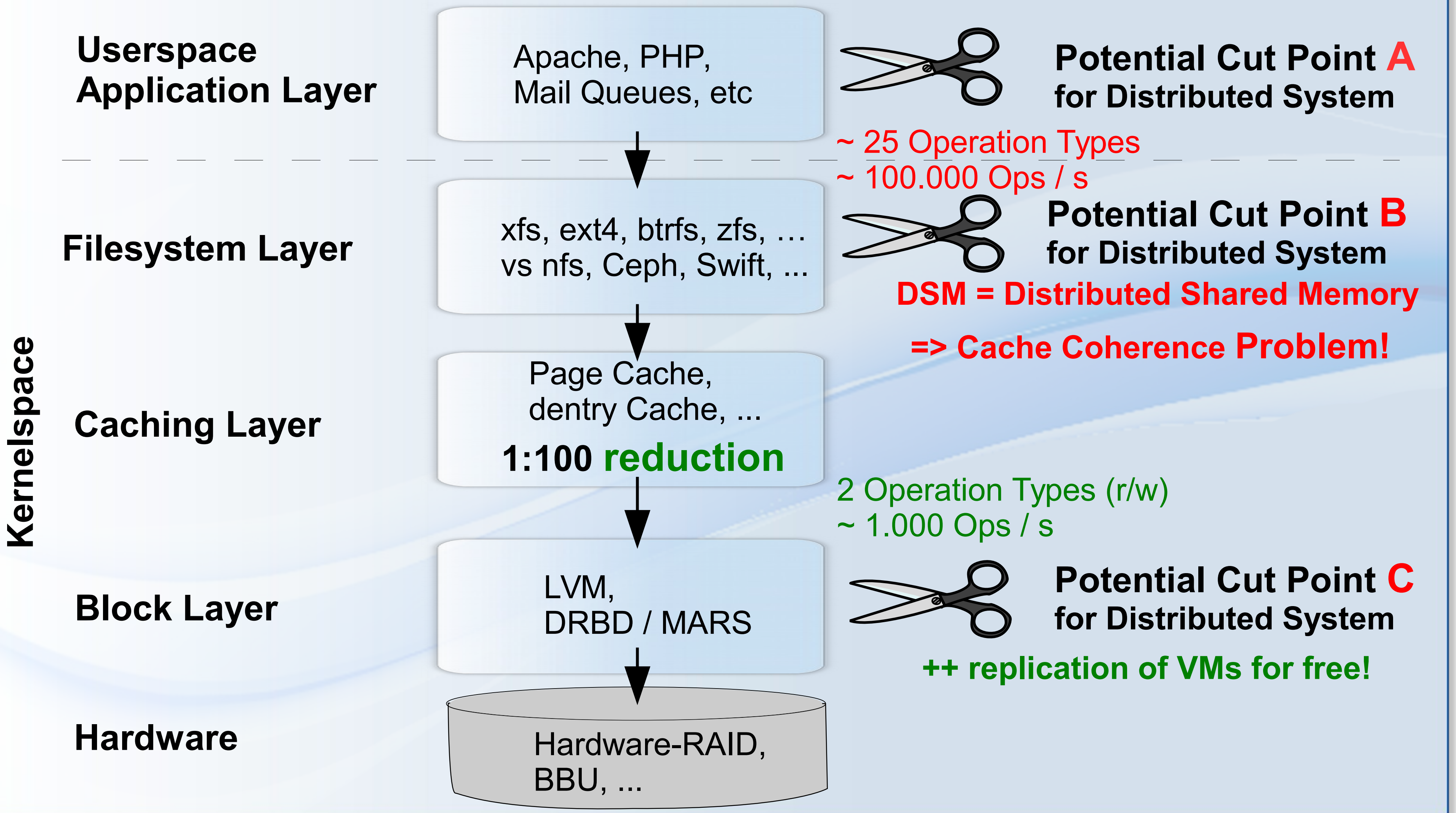


Collaboration sought

=> 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, ...

Use Cases DRBD+proxy vs MARS Light

DRBD+proxy
(proprietary)

Application area:

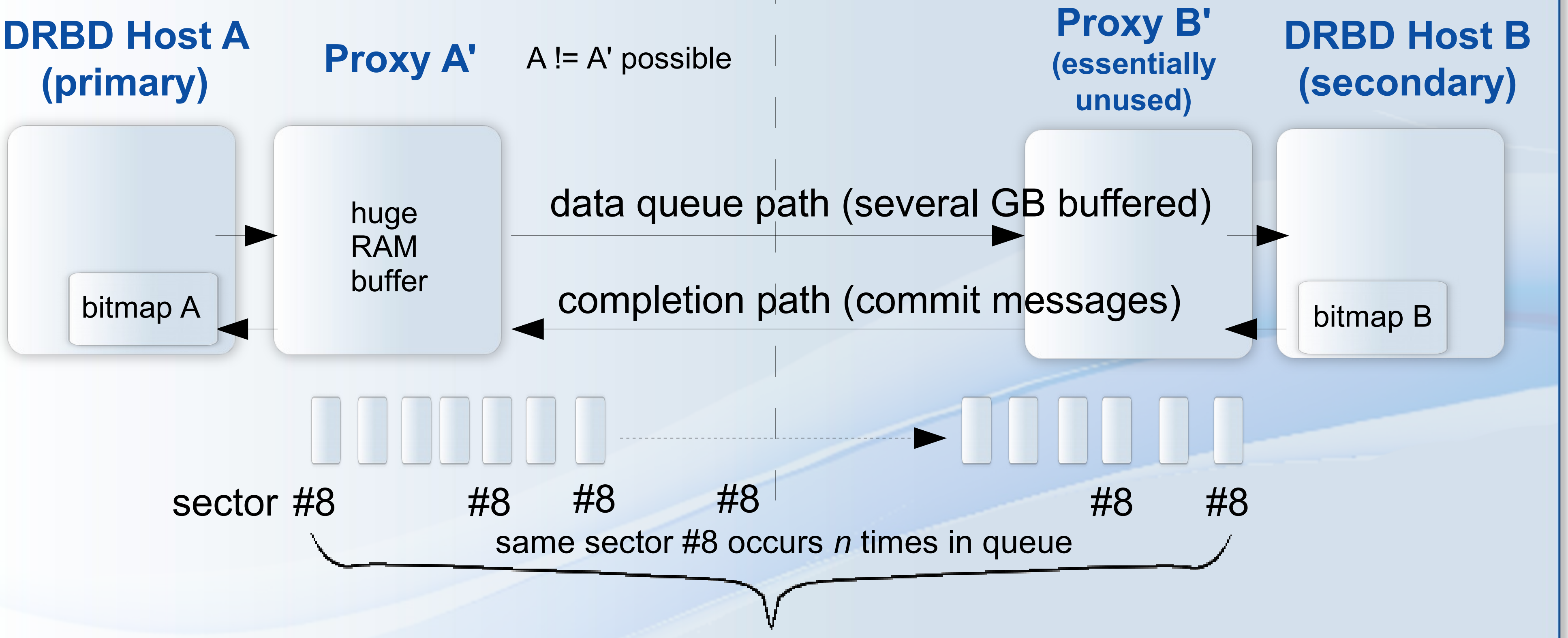
- Distances: any
- Aynchronously
  - **Buffering in RAM**
- Unreliable network leads to **frequent re-syncs**
  - RAM buffer gets lost
  - at cost of actuality
- **Long** inconsistencies during re-sync
- Under pressure: **permanent** inconsistency possible
- High memory overhead
- Difficult scaling to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

MARS Light
(GPL)

Application area:

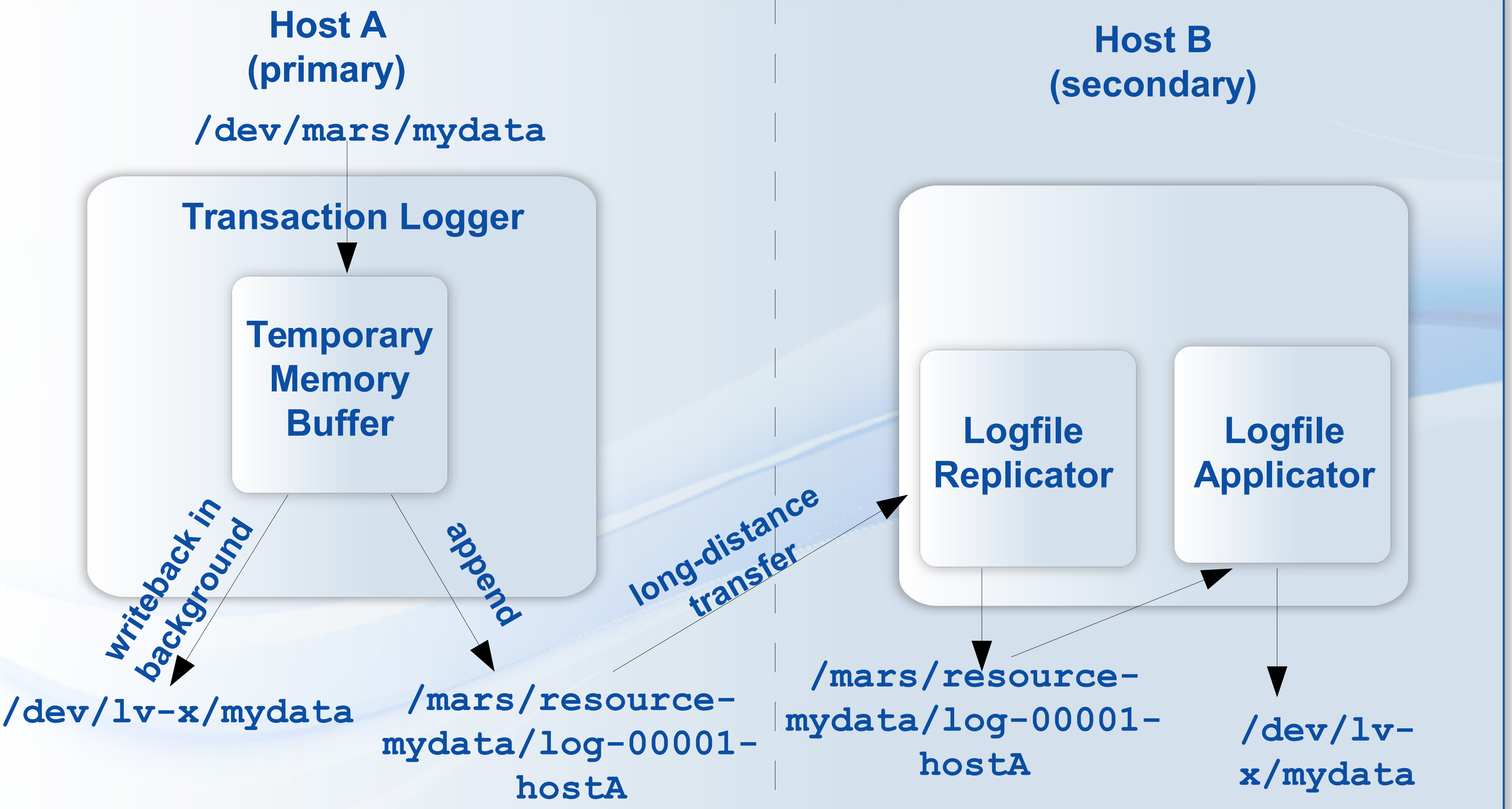
- Distances: **any** (>>50 km)
- Asynchronously
  - near-synchronous modes in preparation
- Tolerates **unreliable network**
- Anytime consistency
  - no re-sync
- Under pressure: no inconsistency
  - possibly at cost of actuality
- Needs >= 100GB in `/mars/` for transaction logfiles
  - dedicated spindle(s) recommended
  - RAID with BBU recommended
- Easy scaling to k>2 nodes

DRBD+proxy Architectural Challenge


DRBD Host A (primary)
Proxy A'
A != A' possible
Proxy B' (essentially unused)
DRBD Host B (secondary)
huge RAM buffer
bitmap A
bitmap B
data queue path (several GB buffered)
completion path (commit messages)
sector #8
#8
#8
#8
#8
#8
same sector #8 occurs *n* times in queue

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



Host A
(primary)

/dev/mars/mydata

Transaction Logger

Temporary Memory Buffer

writeback in background

append

/dev/lv-x/mydata

/mars/resource-mydata/log-00001-hostA

long-distance transfer

Host B
(secondary)

Logfile Replicator

Logfile Applicator

/mars/resource-mydata/log-00001-hostA

/dev/lv-x/mydata

Framework Architecture

for MARS + future projects

Framework Application Layer

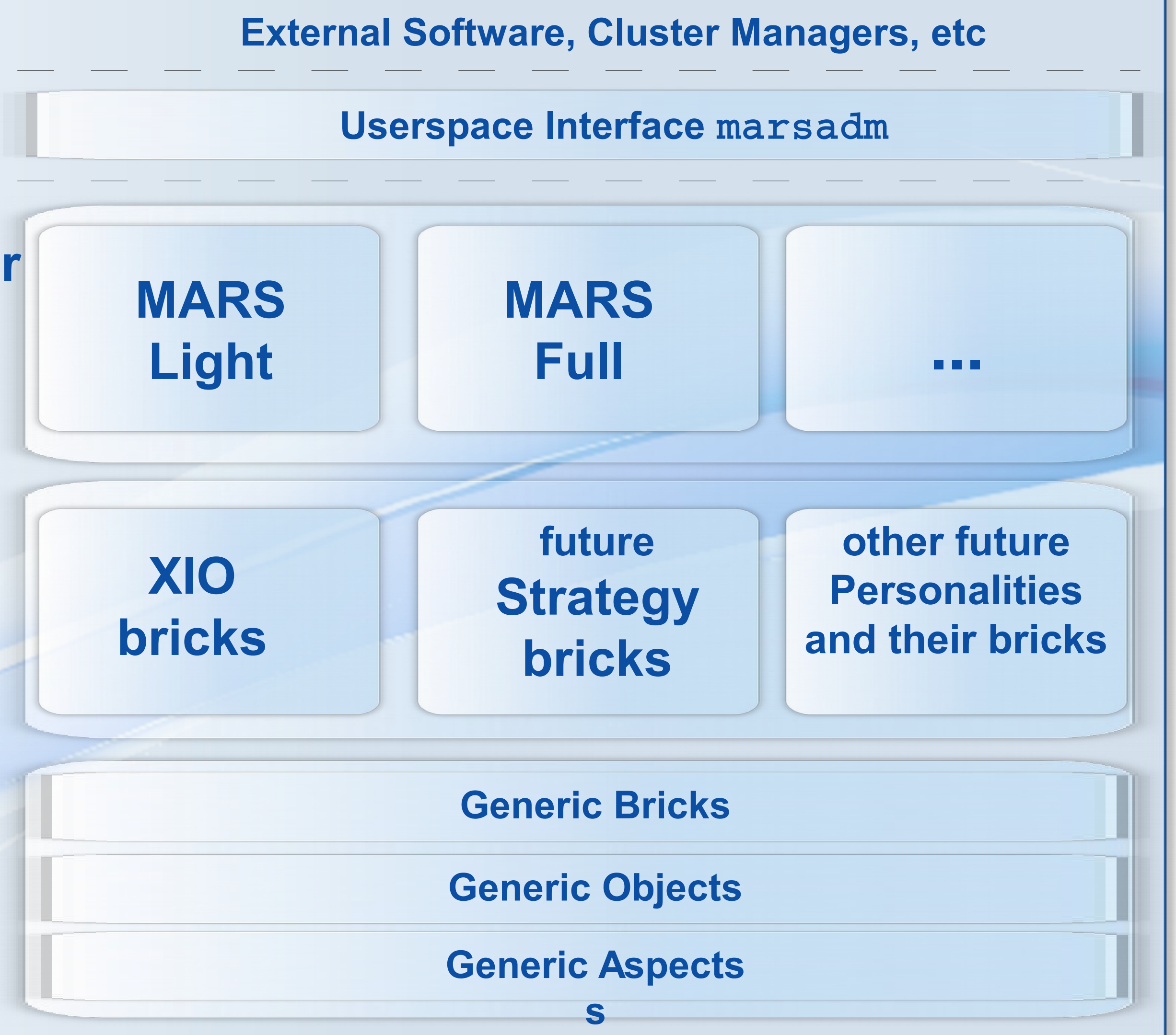
MARS Light, MARS Full, etc

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