Easy Geo-Redundancy with MARS

LCA2020 Kernel Miniconf Presentation

by Thomas Schöbel-Theuer

Easy Geo-Redundancy with MARS: Agenda

- **Background: MARS devel under downstream conditions**
- **Architectural Differences MARS vs DRBD, dm vs Bricks**
- **Current Status / Future Plans**
- **Discussion: how to bring MARS upstream**

Background: Kernel Downstream

- **1&1 Ionos ShaHoLin** = Shared Hosting Linux
  + managed root servers
- **Customized kernel for > 10.000 servers**
  > 300 patches on top of LTS upstream
  - prio #1: maintain the SLA (next slide)
  - small parts from grsecurity
  - special security „features“
  - daily backup of 10 billion inodes: persistent filemonitor2
  - most frequent operation: `git rebase`
  - highly customized .config
- **Non-public, eats > 100% of my capacity**

Background: SLA + Growth

- SLA: 99.98% end-to-end measured from Frankfurt
  - Including WAN outages, PHP problems, HumanError™

    => MARS geo-redundancy is compensating much better!
- 4 datacenters at 2 continents, pair distance > 50 km
- ~ 9 millions of customer home directories
- ~ 10 billions of inodes + daily incremental backup
- > 4.7 petabytes *allocated* in ~ 3800 xfs instances

  LocalStorage LVM ~ 8 PB x 2 for geo-redundancy via MARS

  `https://github.com/schoebel/mars`
- Data growth rate ~ 21 % / year
- Solution: Container Football on top of MARS

  `https://github.com/schoebel/football`

History of MARS

1&1

- **2009: geo-redundancy introduced, via DRBD ~50km**
  - DRBD sync speed: >50 MB/s in *lab testing* over 1GBit uplinks
  - 2012: single sync dropped to 5 MB/s. Highly congested cross-DC lines by ordinary repl traffic of >3000 DRBD resources
  - Asked Linbit for architectural change: separate TCP connections for sync traffic. Answer: NO
- **2010 personal initiative: MARS started during my spare time**

  `marsadm` similar to `drbdadm`
- **2014: mass rollout (replacement of DRBD)**
- **Today's cross-DC sync speed: 800 MB/s** over 10GBit uplinks

  port 7777: metadata traffic symlink tree

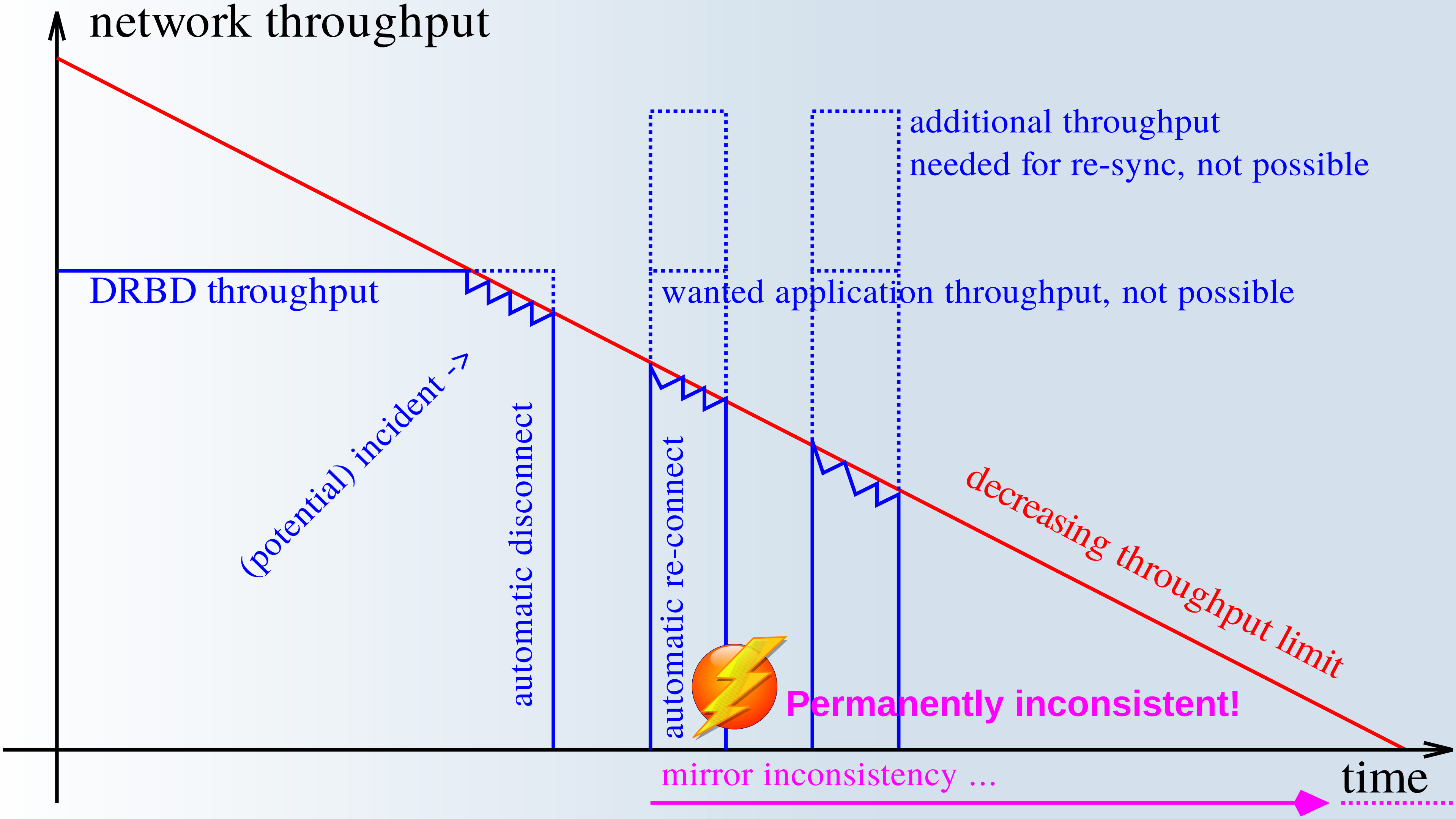
  port 7778: replication traffic transaction logs

  port 7779: sync traffic

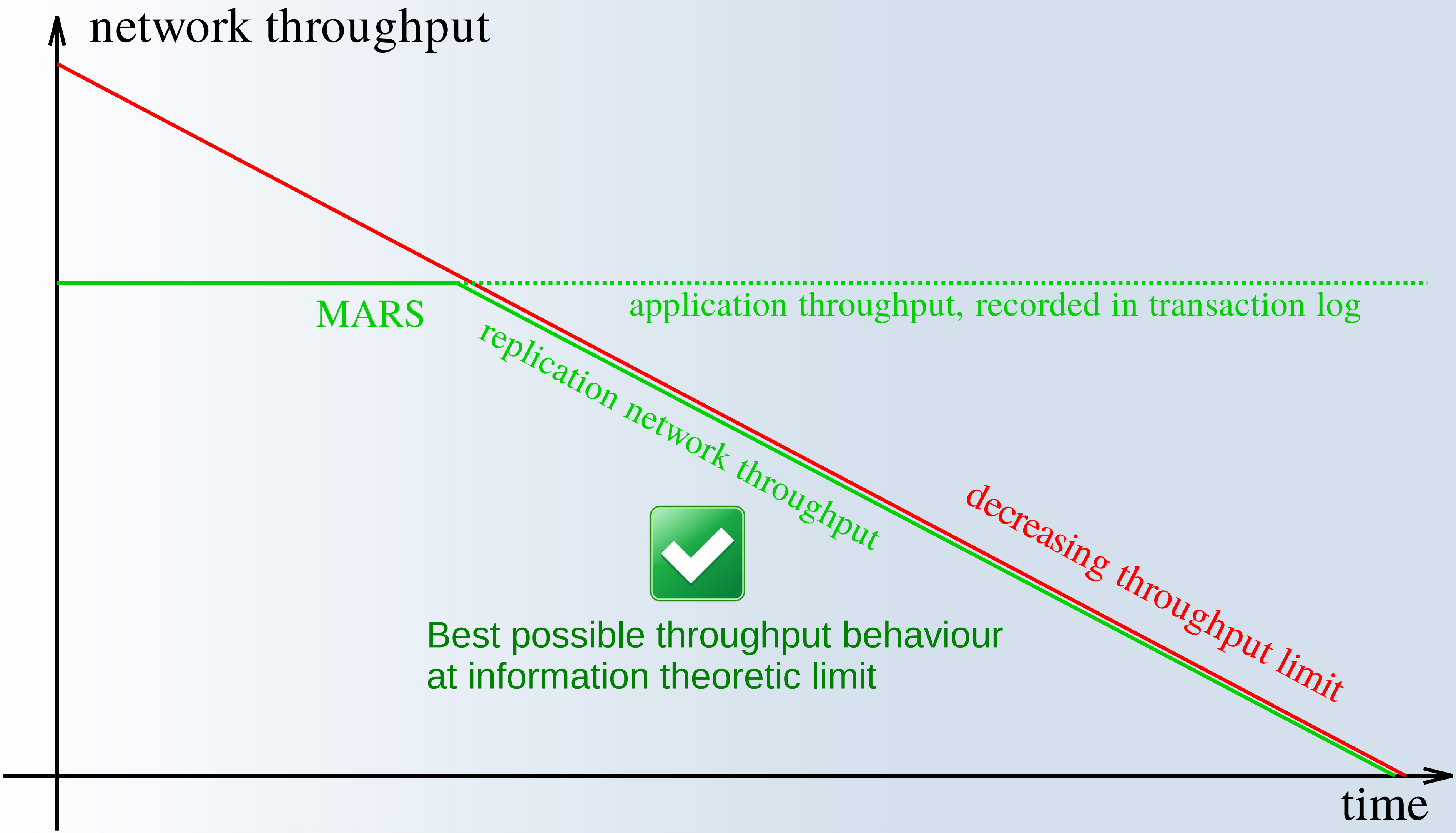
  Traffic Shaping possible by port / DSCP marking

  Socket Bundling: default 2 parallel TCP connections / resource / port

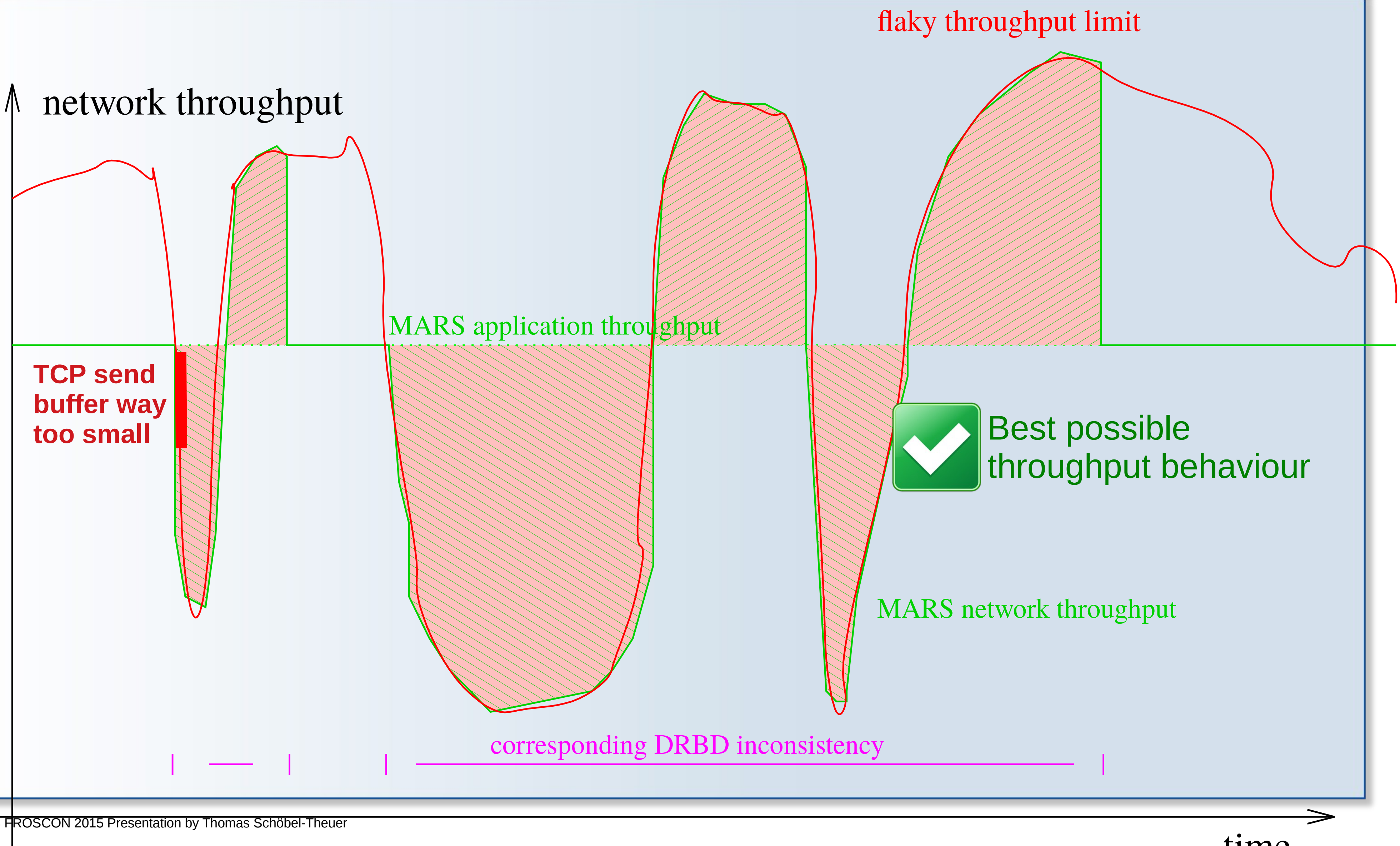
Network Bottlenecks (1) DRBD over long distances



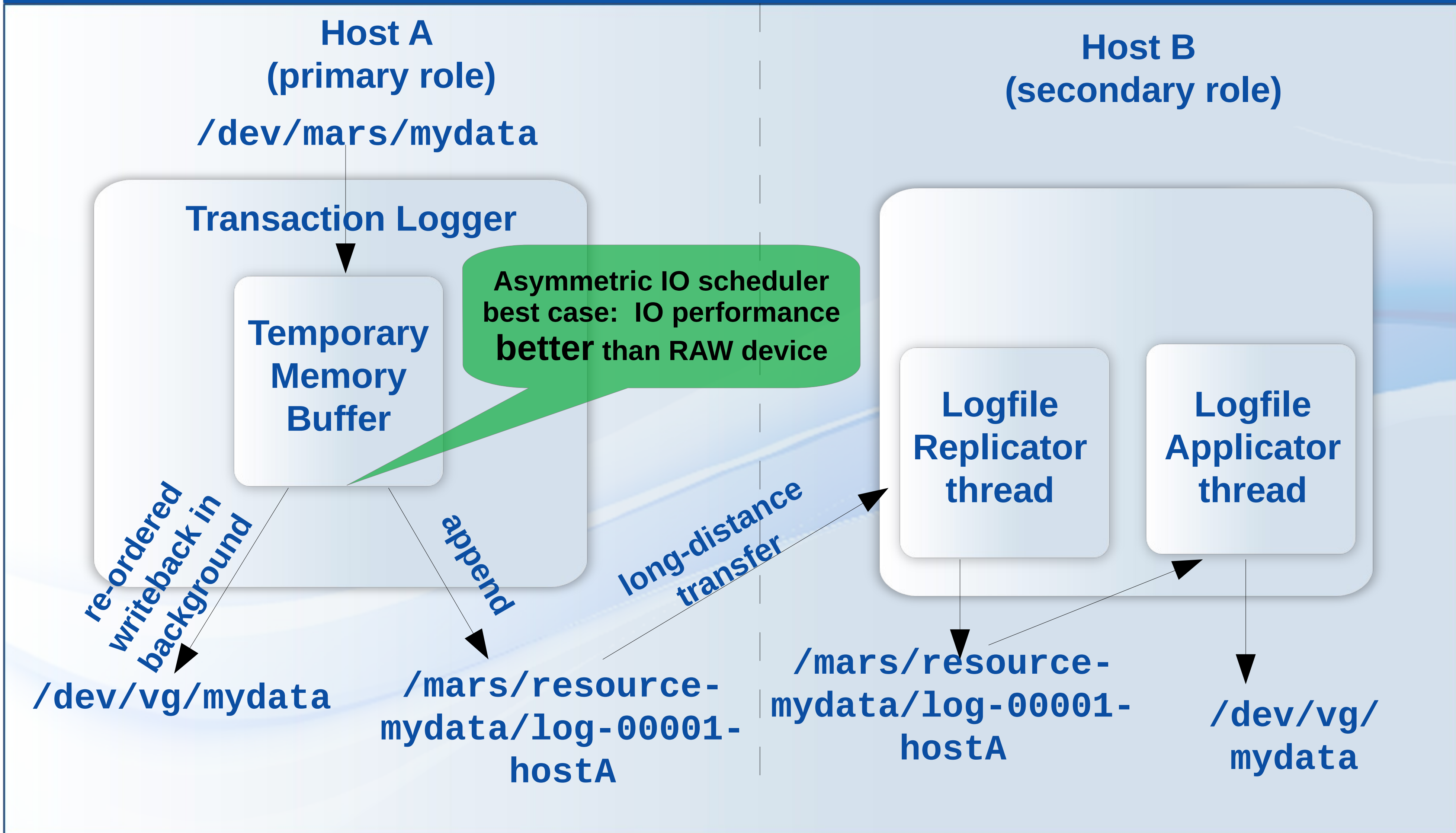
Network Bottlenecks (2) MARS



Network Bottlenecks: MARS



MARS Data Flow Principle



MARS Current Status

- MARS source under GPL + docs:

  `github.com/schoebel/mars`

  `docu/mars-architecture-guide.pdf`

  `docu/mars-user-manual.pdf`

READ THIS
for deeper background

- mars0.1stable productive since 02/2014
- Backbone of the 1&1 Ionos geo-redundancy feature
- Currently supports kernels 3.2 to 4.14
  - see kernel/compat.h
  - pre-patch only EXPORT_SYMBOL()
  - non-breaking backwards compatibility between MARS versions since 2014
- ShaHoLin: up to 14 LXC containers on 1 hypervisor
  - Efficiency project using Football:
  - TCO has **halved!**

Architectural Differences DRBD vs MARS

- DRBD: structured like classical Linux driver
- activity log + bitmap
- data + metadata intermixed
- Strict Consistency

- MARS: instance-oriented brick architecture
- Data structures implemented *inside* of bricks
- *Any* wire replacable with client-server network connection

  supports Location Transparency

- Separation of data vs metadata
- Eventually Consistent via Lamport Clock

Architectural Differences dm vs Bricks

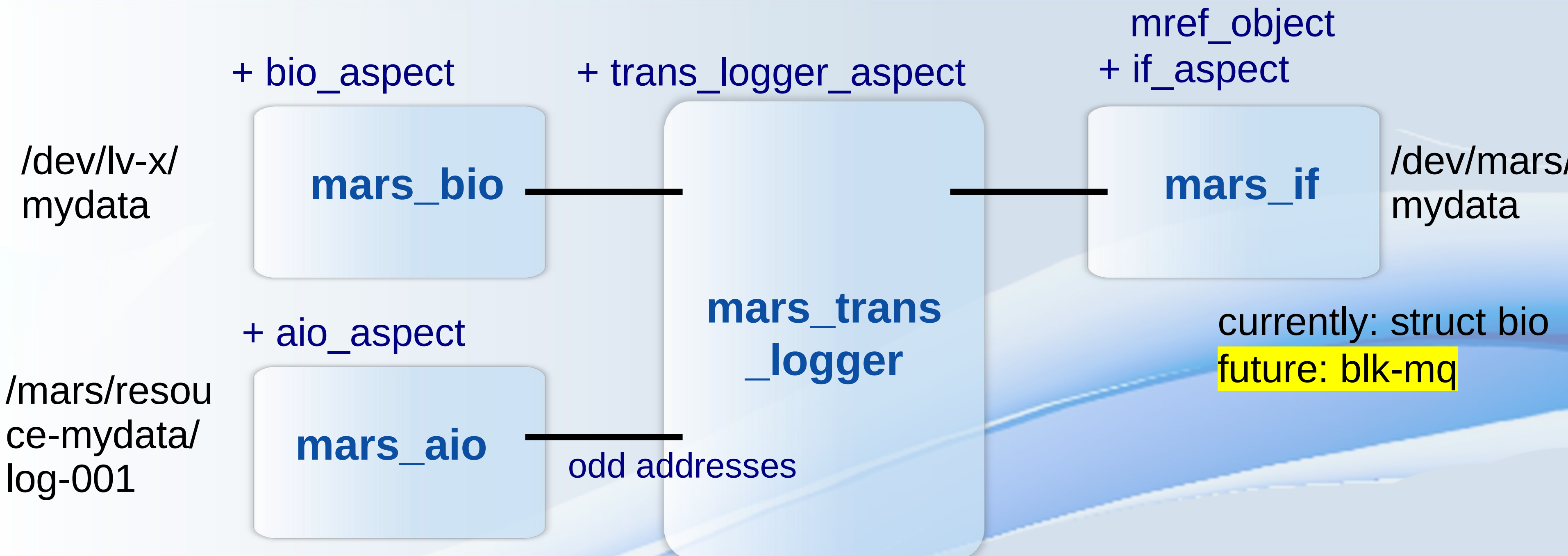
- device mapper: firmly bound to `struct bio`
- stacking: *tree* structure
- static stacking order

- MARS: generic brick infrastruct can have *multiple* personalities
  - MARS `struct mref_object` for *both* block and file IO (odd addresses)
  - future personalities: e.g. stacked asynchronous filesystems
- wiring: DAG structure, and on multiple inputs / outputs per brick
- dynamic *re*-wire during runtime

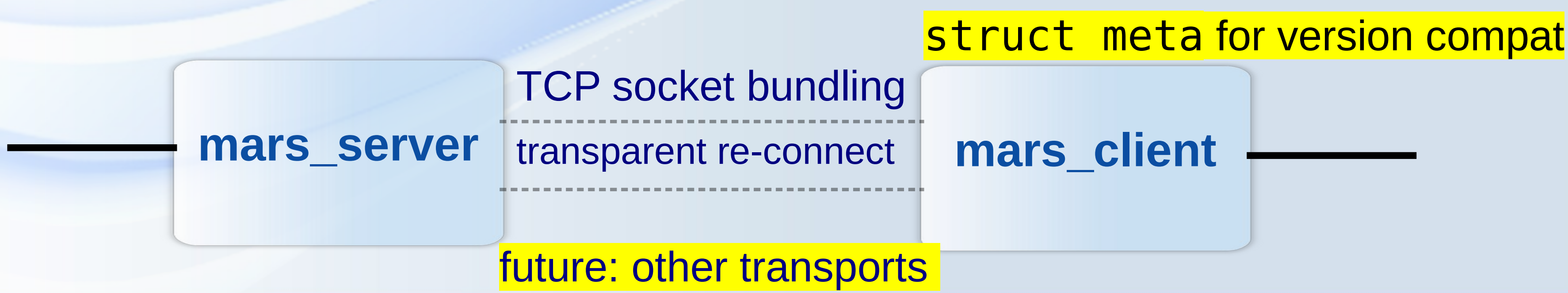
  needed for log-rotate!

  future: handover of client-server connections without IO stalls
- Interfaces: object-oriented
- Bricks: aspect-oriented

Bricks, Objects + Aspects



Aspects are automatically attached on the fly **Necessary for runtime re-wiring**



Generic Symlink Tree for Metadata Exchange

- „Misuse“ of tree-structured symlinks as a persistent key → value store
  - history: fear at 1&1 in 2011 that MARS could be canceled when taking too long
  - may be replaced by another representation `(lamport_stamp,key,value)`
- Whole tree is replicated throughout the cluster
- Name clash avoidance: **origin hostname** is encoded into key

No hostname encoded into key
=> common for all cluster hosts
mtime = Lamport Stamp => Eventually Consistent

- Examples:
  - /mars/resource-mydata/primary -> hostA
  - /mars/resource-mydata/actual-**hostA**/is-primary -> 1

MARS Future Plans

Kernel part almost done
Mixed operations of old/new MARS protocol versions via `struct meta`

- Faster checksumming (CRC32c | CRC32 | SHA1 | MD5)
- Logfile compression (LZO | LZ4 | ZLIB)
- Optional network transport compression
  - may help for some very slow networks

IO data paths already scaling well

- TODO: better *metadata* scalability needed!
  - single `mars_main` control thread (non-blocking)
  - TODO: more resources per host (max. 24 in prod at 1&1)
  - TODO: more hosts per cluster
    - requires slight restructuring of symlink tree
- TODO: Linux kernel upstream
  - get rid of downstream version
  - requires a *lot* of work!
  - see next slide

Discussion: howto get MARS upstream?

- 0) Keep current setup
  - won‘t work because of my >100% downstream workqueue.
- 1) First help me preparing the out-of-tree version for upstream submission
  - Upstream senior, please help + coach me
  - Side conditions:
    - „symlink forest“ could/should be replaced by better metadata representation

      e.g. list of tuples `(lamport_stamp,key,value)`
    - Existing user base expects compatibility, at least a migration script.

      better on secondary: `rmmod mars; depmod -a; modprobe mars`
    - Out-of-tree MARS needs to be maintained for LTS kernels *until* upstream version is LTS && meets SLA => ultimately I want to get rid of non-upstream version.
- 2) Get rid of downstream kernel work, e.g. by changing employer.
  - Should be a company commited to OpenSource
  - Best for a distributor, or a major user of MARS (on enterprise-critical data)

Appendix

Why GEO-Redundancy

DR = Disaster Recovery
CDP = Continuous Data Protection

- **Example: GALILEO incident (DR / CDP did not work)**
  - Disaster = earthquake, flood, terrorist attack, mass power outage, ...
- **BSI Paper 12/2018:**

  Kriterien für die Standortwahl höchstverfügbarer und georedundanter Rechenzentren

  https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Sicherheitsberatung/Standort-Kriterien_HV-RZ/Standort-Kriterien_HV-RZ.pdf?__blob=publicationFile&v=5

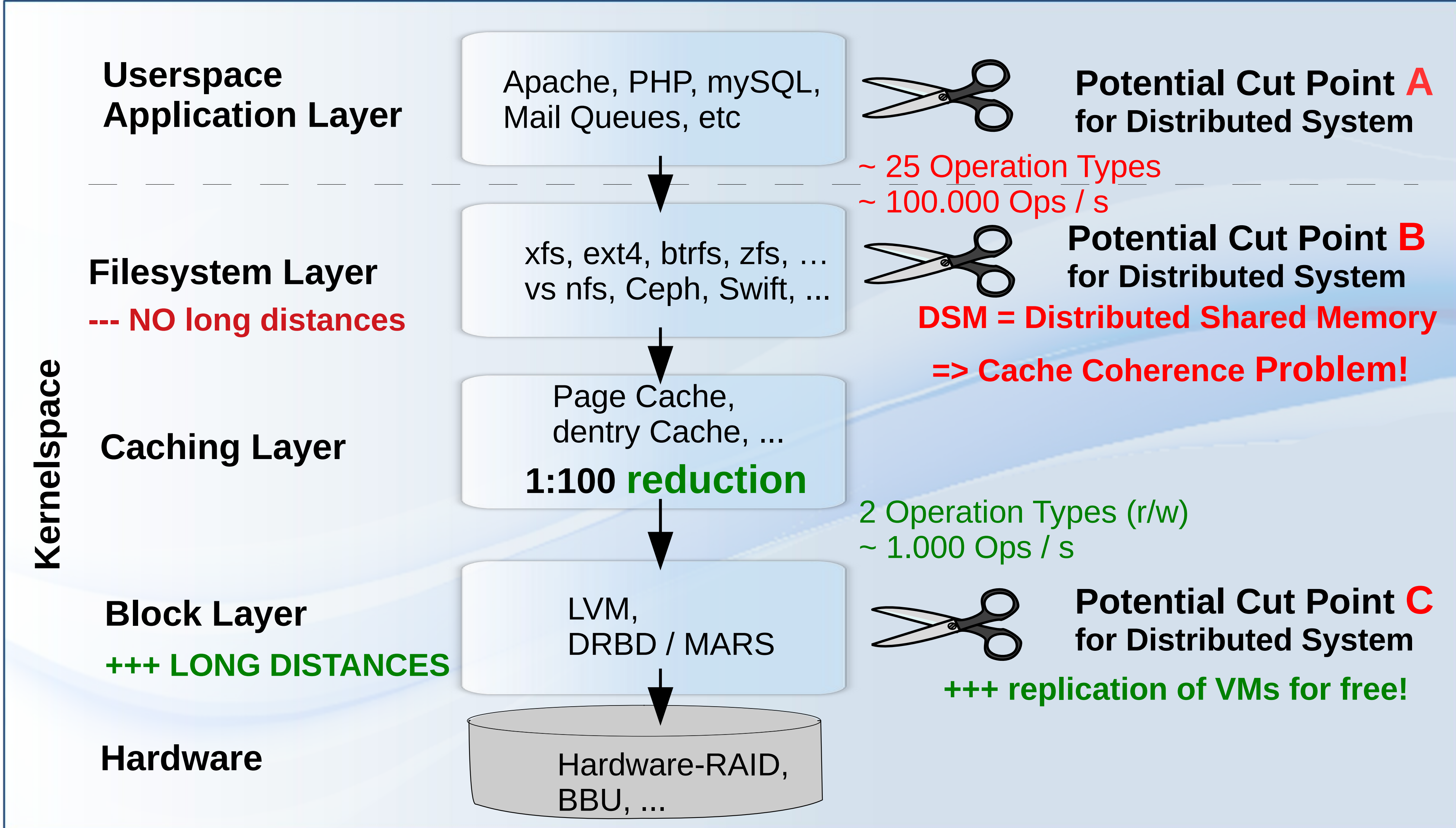
  in English: Criteria for Locations of Highly Available and Geo-Redundant Datacenters

  - Stimulated some controversial discussions, but see commentary
    https://www.it-finanzmagazin.de/bsi-rechenzentren-entfernung-bafin-84078/
- **Conclusions: distances > 200 km „recommended"**
- **Might influence future legislation** (EU / international)
- **„Critical Infrastructures" more important!**

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 + MD5 checksums + Anytime Consistency**

Replication at Block Level vs FS Level



CAP Theorem

C = Strict Consistency

Partitioning unavoidable at
- disasters
- LONG distances

A = Availability

P = Partitioning Tolerance

= the network can have its own outages

Use Cases DRBD+proxy vs MARS

DRBD+proxy
(proprietary)

Application area:

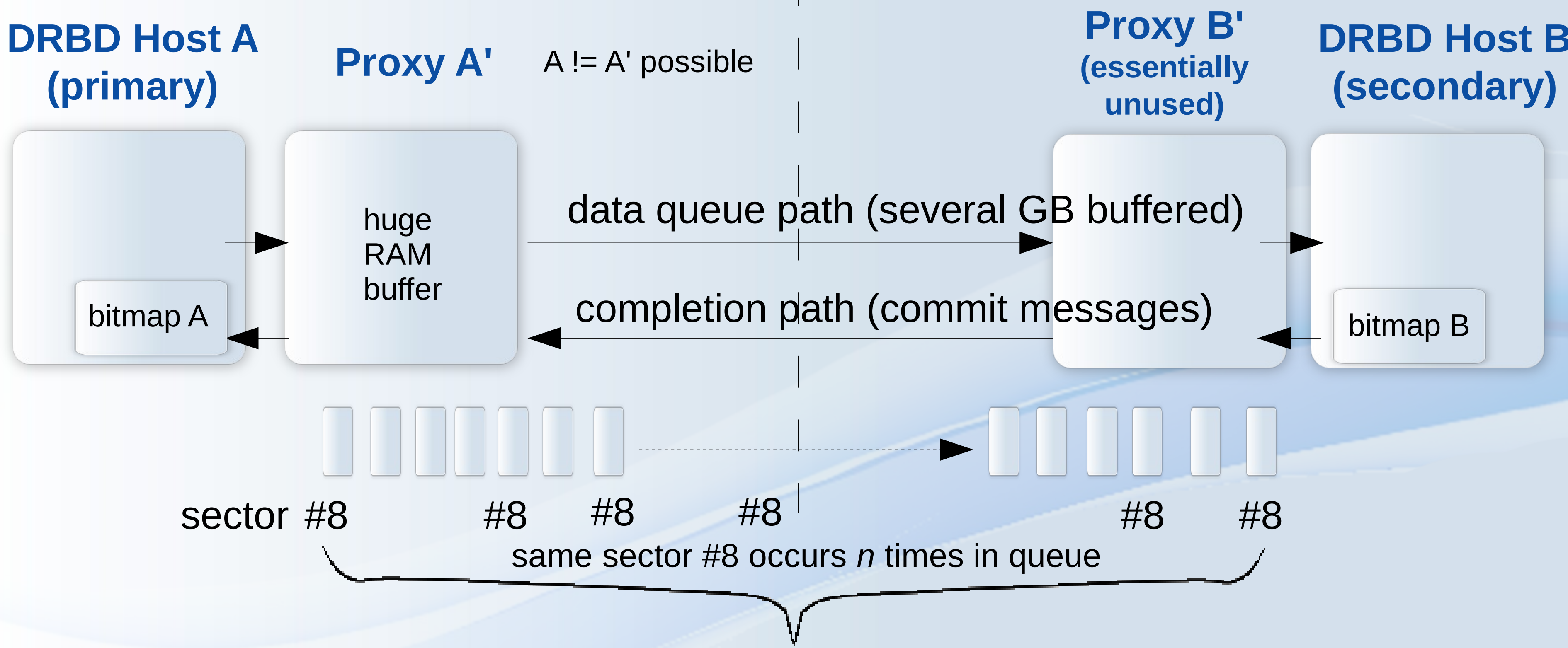
- Distances: any
- Aynchronously
  - **Buffering in RAM**
- Unreliable network leads to **frequent re-syncs**
  - RAM buffer gets lost
  - at cost of actuality
- **Long** inconsistencies during re-sync
- Under pressure: **permanent** inconsistency possible
- High memory overhead
- Difficult scaling to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

MARS
(GPL)

Application area:

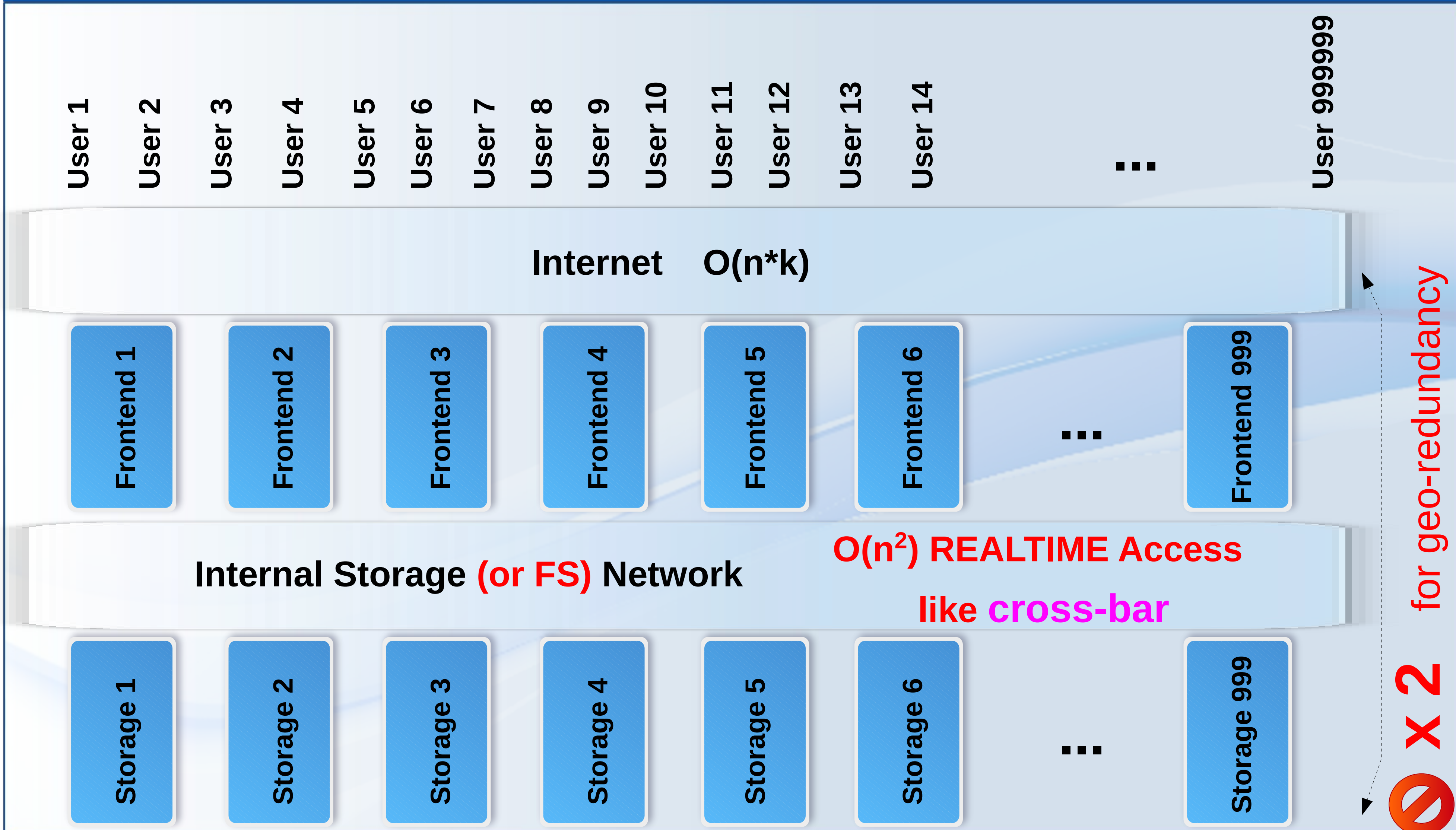
- Distances: **any** (>>50 km)
- Asynchronously
  - near-synchronous modes in preparation
- Tolerates **unreliable network**
- Anytime consistency
  - no re-sync
- Under pressure: no inconsistency
  - possibly at cost of actuality
- Needs >= 100GB in `/mars/` for transaction logfiles
  - dedicated spindle(s)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

DRBD+proxy Architectural Challenge

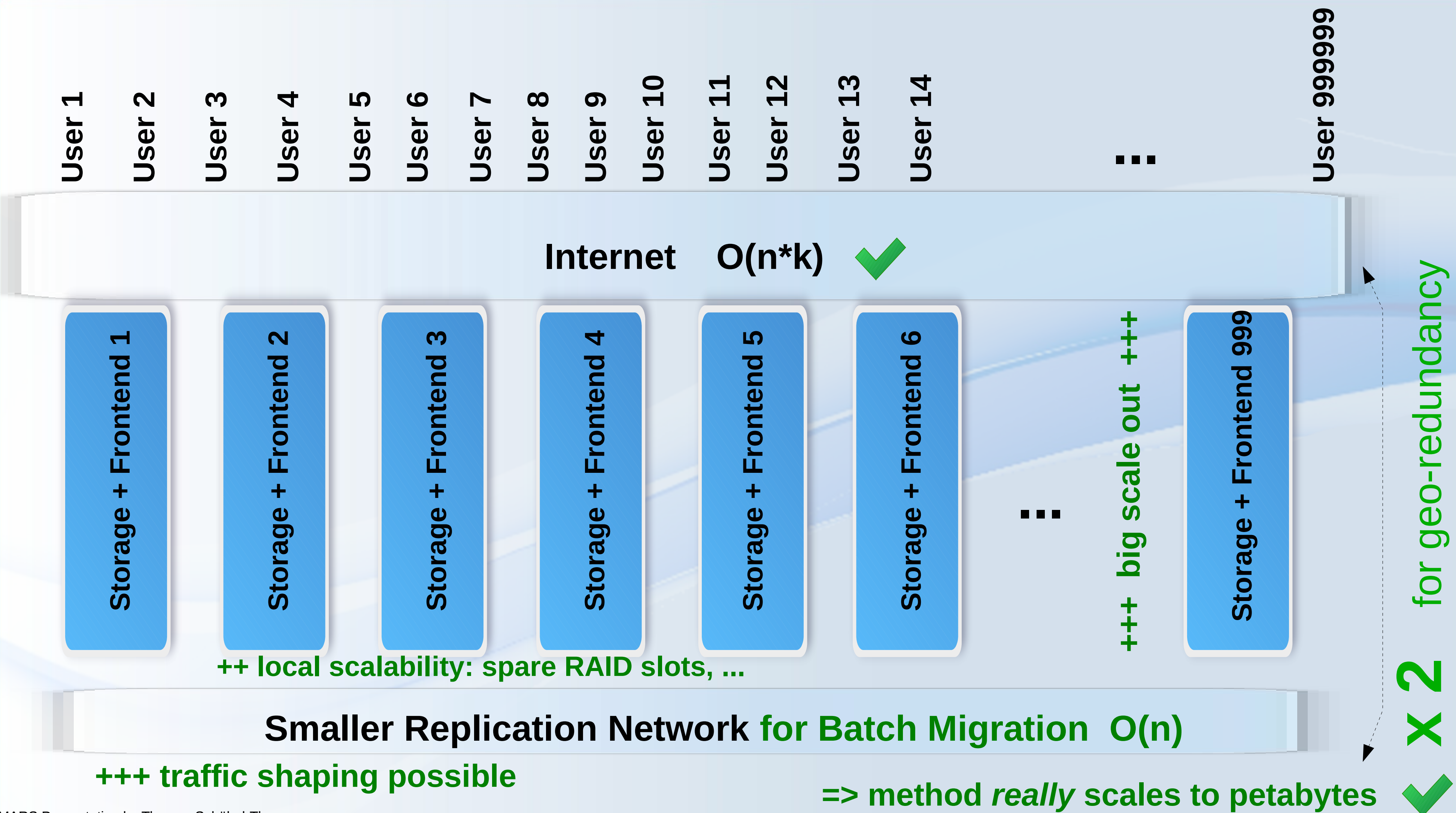


n times
=> need *log*(n) bits for counter
=> but DRBD bitmap has only 1 bit/sector
=> workarounds exist, but complicated
(e.g. additional dynamic memory)

Badly Scaling Architecture: Big Cluster



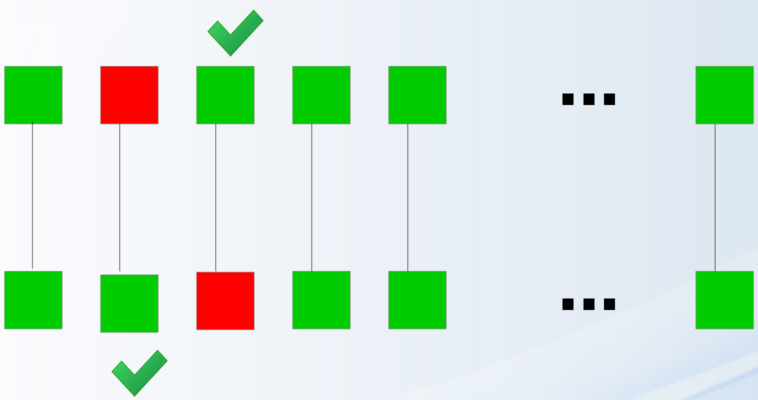
Well-Scaling Architecture: Sharding



Reliability of Architectures: NODE failures

2 Node failure => ALL their disks are unreachable

DRBD or MARS simple pairs

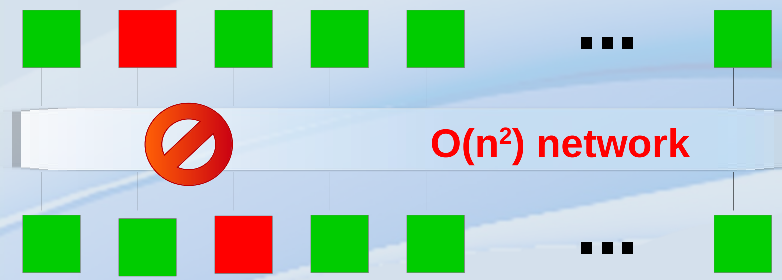


=> no customer-visible incident

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

same n

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

Cost (1) non-georedundant, n>100 nodes

1&1

- **Big Cluster:**
  Typically ≈RAID-10 with
  k=3 replicas for failure
  compensation
- Disks: > 300%
- Additional CPU and RAM

  for storage nodes
- Additional power
- Additional HU

- **Simple Sharding:**
  Often local RAID-6
  sufficient (plus external backup,
  no further redundancy)
- Disks: < 120%
- Client == Server
  no storage network

  MARS for LV background migration
- Hardware RAID controllers

  with BBU cache on 1 card
- Less power, less HU

Cost (2) geo-redundant => LONG Distances

- **Big Cluster:**
  - 2X ≈ RAID-10 for failure compensation
    (k=6 replicas needed, smaller does not work in long-lasting DC failure scenarios)
- **Disks: > 600%**
- **Additional CPU and RAM for storage nodes**
- **Additional power**
- **Additional HU**

- **Geo-redundant Sharding:**
  - 2 x local RAID-6
  - MARS for long distances

    or DRBD for room redundancy
- **Disks: < 240%**
- **Hardware RAID controllers with BBU**
- **Less power**
- **Less HU**

Cost (1+2): Geo-Redundancy Cheaper than Big Cluster

1&1

- **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 • O(n) clients = storage servers

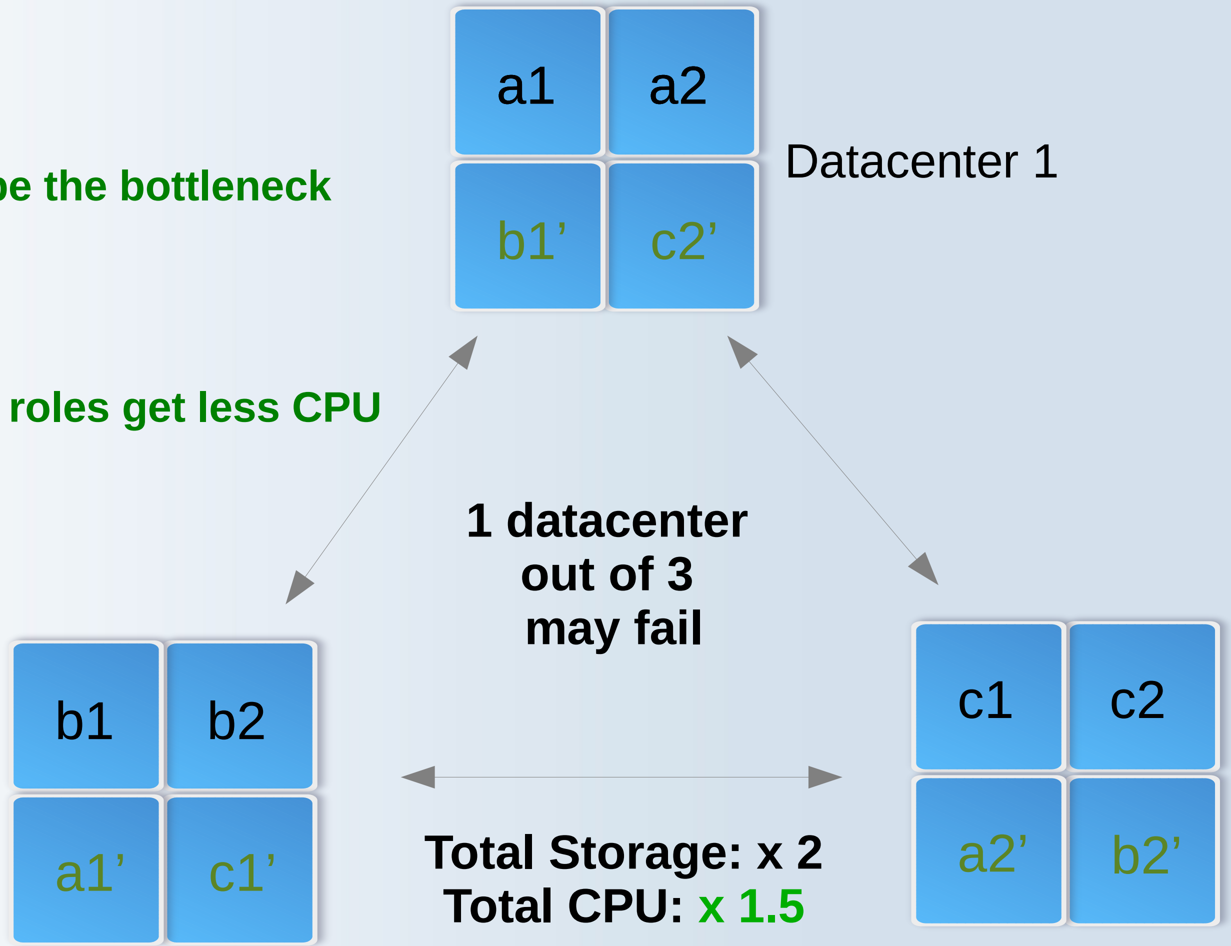
  + O(n) replication network
- Disks: < 240%
- Less total power
- Less total HU

  +++ geo failure scenarios

Cost (3): Geo-Redundancy even Cheaper

Precondition:
CPU must not be the bottleneck

Idea: passive LV roles get less CPU

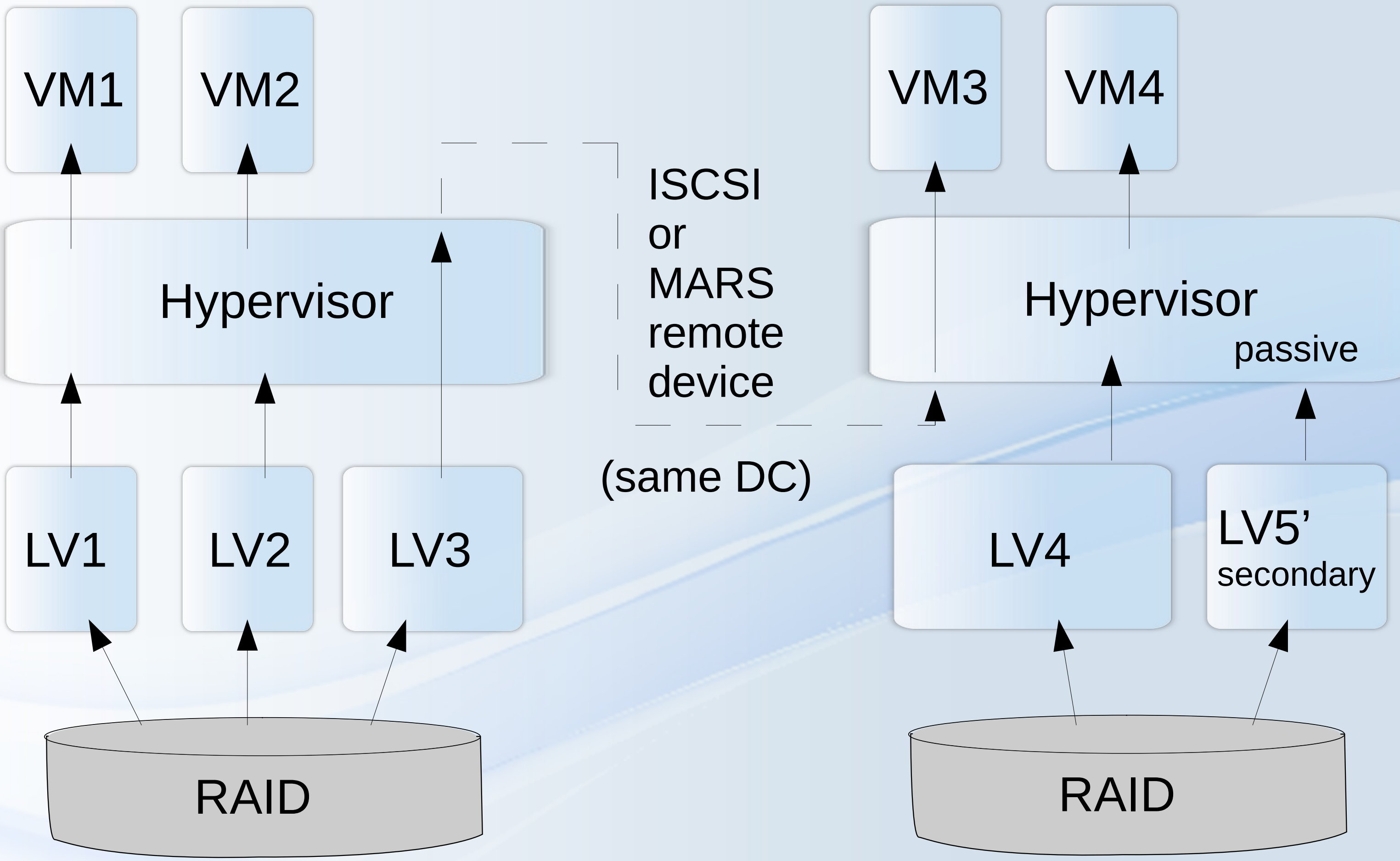


1 datacenter
out of 3
may fail

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

HOWTO flexible CPU assignment => next slide

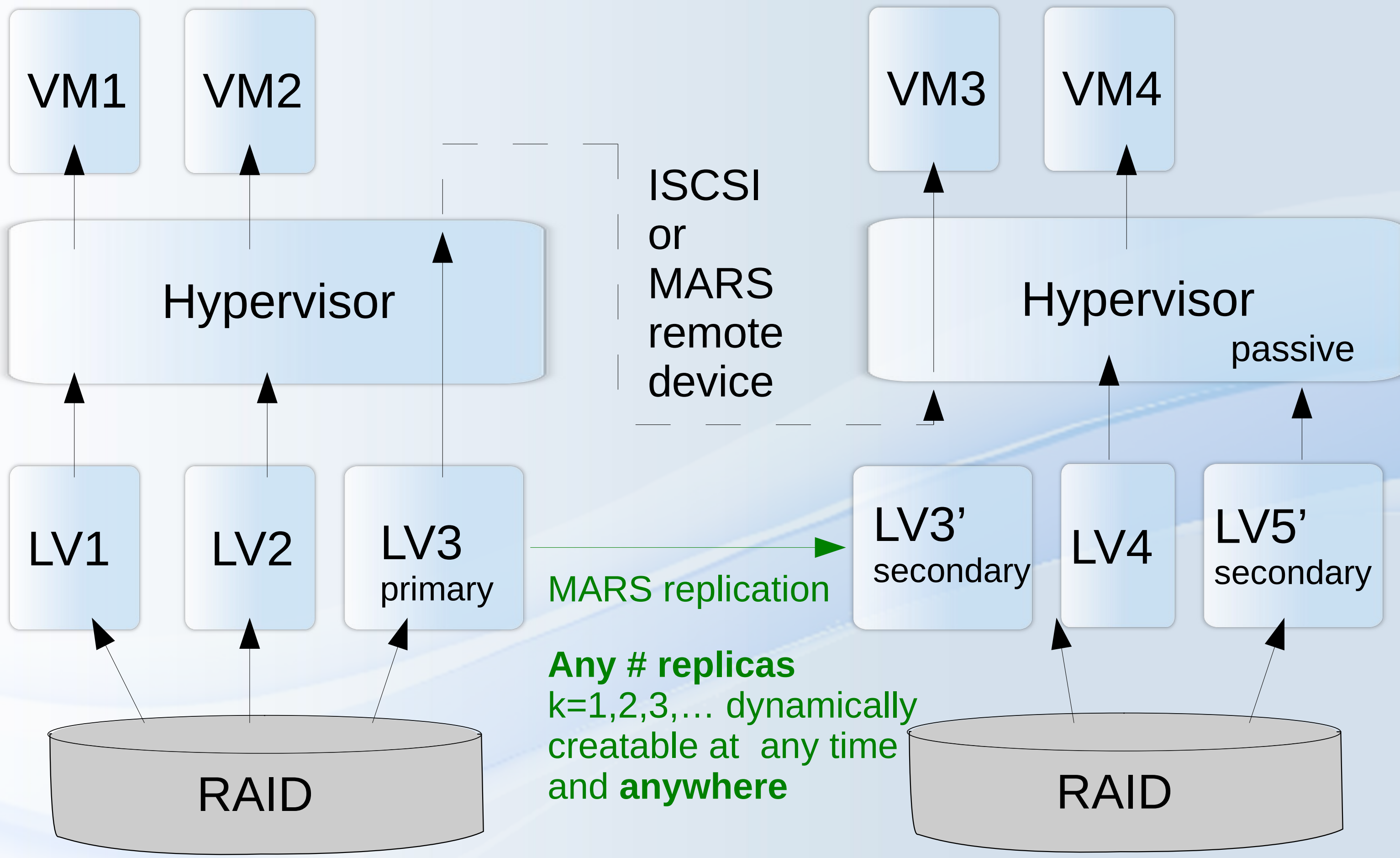
Flexible MARS Sharding + Cluster-on-Demand



any hypervisor works in client and/or server role

and preferably **locally** at the same time

Flexible MARS Background Data Migration



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

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` as cluster manager
  - 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 1&1 Ionos
  - get rid of old hardware (project successfully finished, **TCO is now halved**)
  - load balancing
  - >50 „kicks“ per week
    - limited by hardware deployment speed
    - Proprietary Planner (for HW lifecycle)