Geo-Redundant Failover with MARS

Now and in Future

FrOSCon 2022 Presentation by Thomas Schöbel-Theuer

Geo-Redundant Handover / Failover: Agenda

- Motivation: why GEO-redundancy
- Long-distance asynchronous replication
- Current OPs status: petabytes & co
- What is the future prosumer device?
  - Both local *and* remote storage location transparent
  - Planned handover without service interruption
  - Unplanned failover *as best as possible*
- Discussion

Growth at 1&1 Ionos ShaHoLin = Shared Hosting Linux

- 6 datacenters at 2 continents, pair distance > 50 km
- ~ 10 millions of customer home directories
- ~ 10 billions of inodes
- > 7 petabytes *allocated* in ~ 4000 xfs instances

  LVM > 10 PB x 2 for geo-redundancy via MARS

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

- Growth rate ~ 20 % per year

Operational since 2014

Why GEO-Redundancy

Full Loss of Datacenters
or *arbitrary* parts

- **Example: 2021 Ahrtal geo disaster**
  - Disaster = earthquake, flood, terrorist attack, full power outage, ...
- **BSI Papers**

  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“**
- **Influence future legislation** (EU / international)

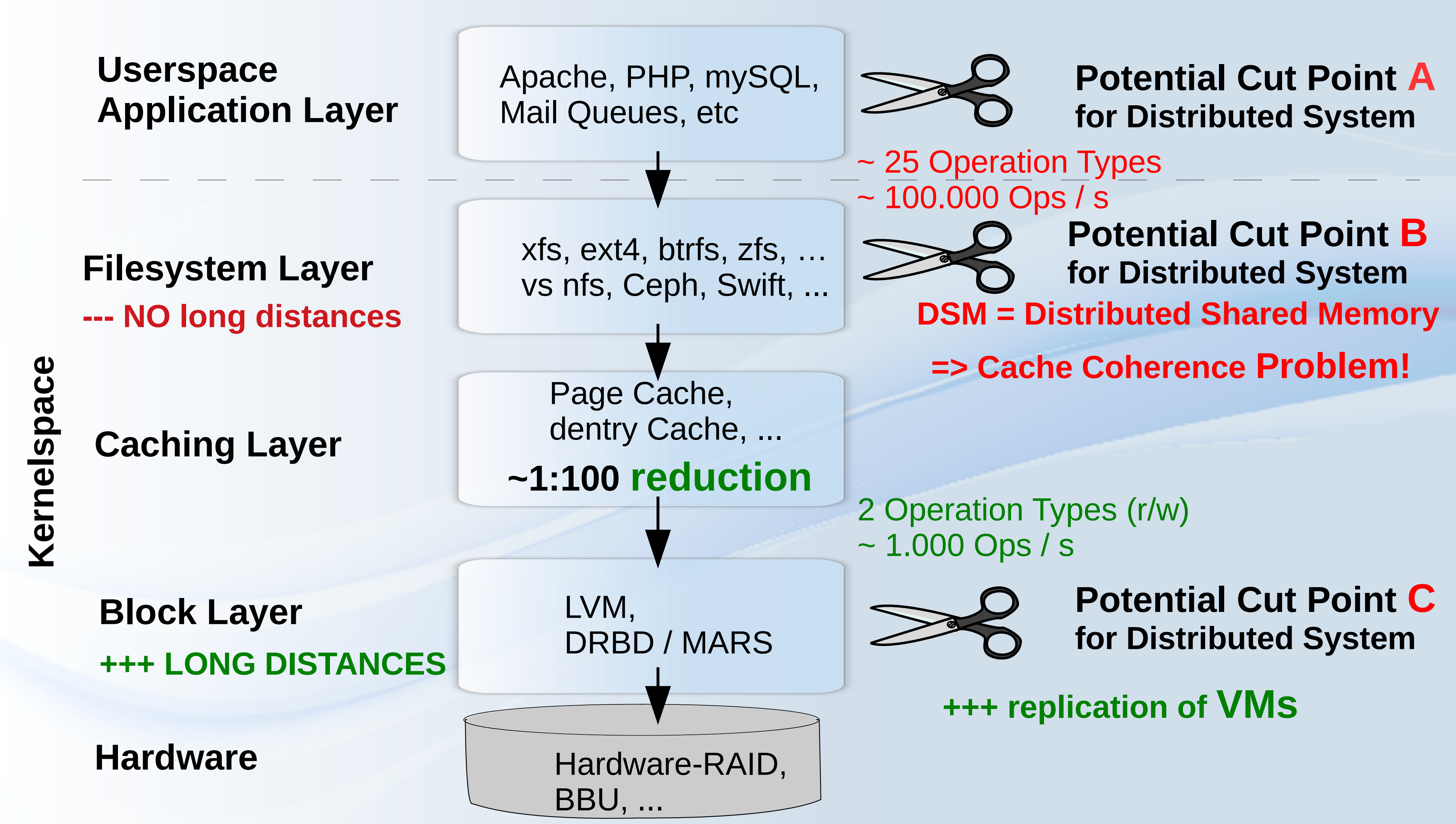
New: KRITIS

Long-Distance *Asynchronous* Replication

network latencies

- **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 + CRC || MD5 checksums + Anytime Consistency

Replication at Block Level vs FS Level



MARS Current Status

kernel module + `marsadm` tool

- MARS source under GPL + docs:

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

  `docu/`
  `mars-user-manual.pdf` ~ 140 pages

  `architecture-guide-geo-redundancy.pdf` ~ 180 pages

- mars0.1stable* productive since 02/2014

- Backbone of the 1&1 Ionos geo-redundancy feature

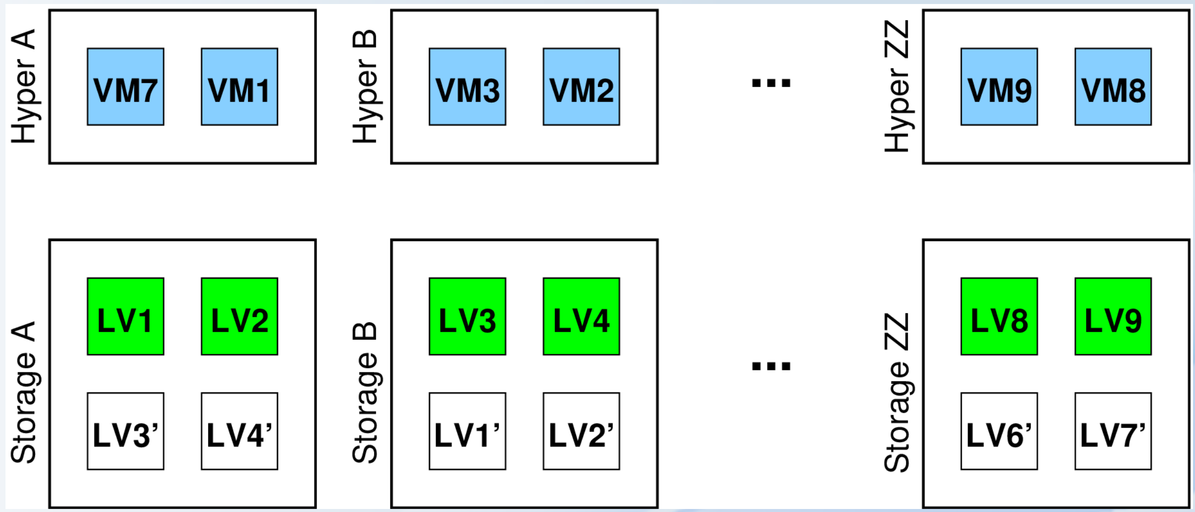
MARS Future Plans in short

- LTS kernels >= 5.10 WIP-qio-for-*
- Prosumer device (WIP => next slides)
- Linux kernel upstream
  requires a *lot* of work!
- Backlog: more tooling, integration into other OpenSource projects

Collaboration sought

=> Opportunities for other OpenSource projects!

Prosumer Device (1) FlexibleSharding



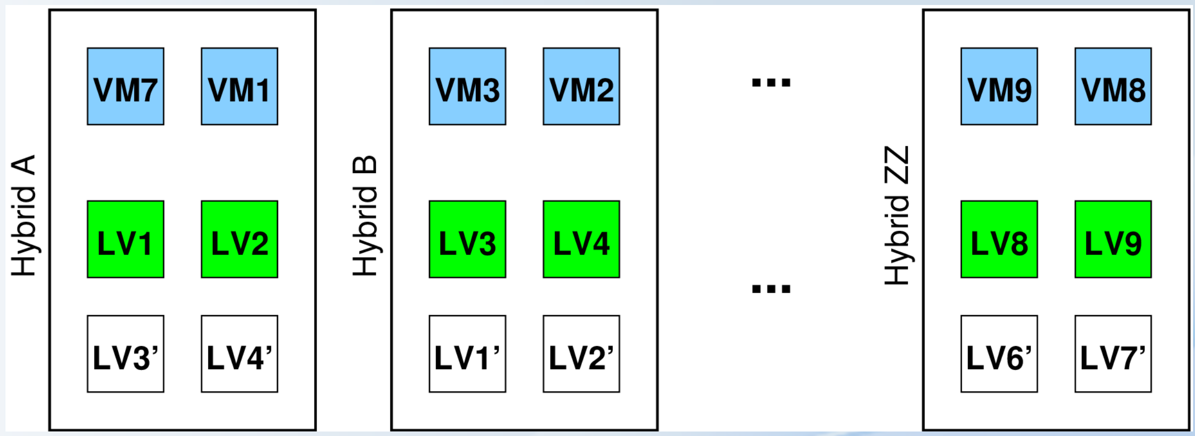
Any `/dev/mars/$vmname` can appear at any machine
whether "storage" machine or "hypervisor" machine

***automatic* introduction of iSCSI-like network connections on port 7776**

***backwards* compatible to classical MARS**

`LocalProsumer` **`RemoteProsumer`**

Prosumer Device (2) Hybrid Machines



… or "hybrid" machine … or imagine ...

no service interruption

HW lifecycle support

planned handover || unplanned failover

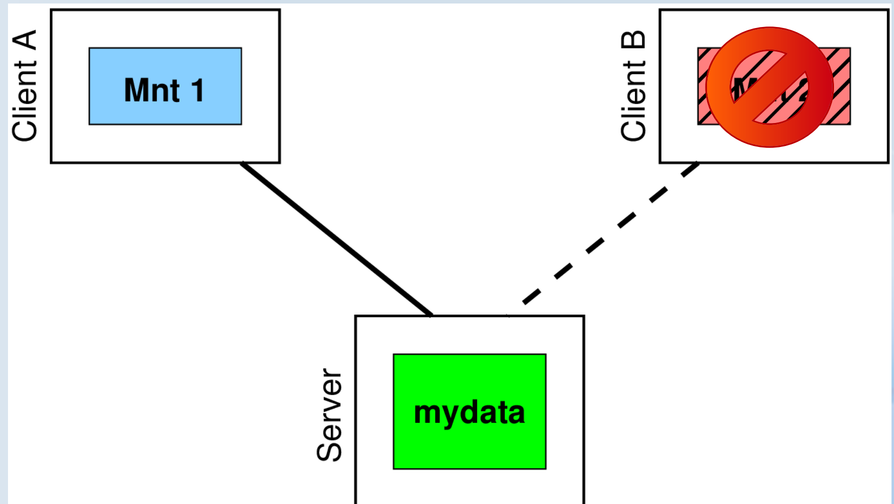
- only the storage
- only `/dev/mars/$name`

- *both* in parallel

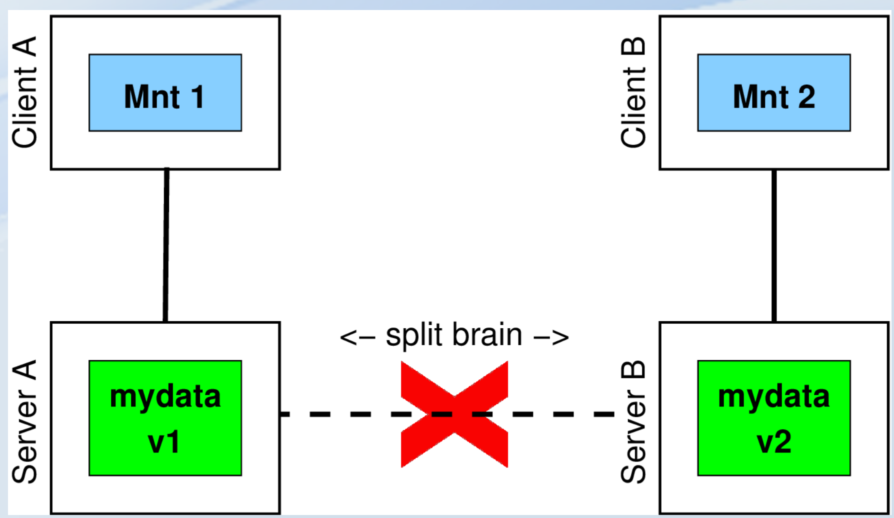
Prosumer Device (3) Unplanned Failover Scenarios

unplanned service interruption unavoidable

only **`/dev/mars/$name`**



both **in parallel**



planned handover: similar to multipath

Prosumer Device (4) Preliminary Documentation

- Surf to `https://github.com/schoebel/mars`
  - Select the branch **`WIP-prosumer`**
  - Click on `docu/`
  - Download `mars-user-manual.pdf`
  - Read new chapter 5: **The MARS Prosumer Device** **p.61-79**
    - Optionally: consult `architecture-guide-geo-redundancy.pdf` from branch `master`
- Please contribute!

Discussion

Appendix

CAP Theorem

globally vs locally
over long distances?

C = Strict Consistency

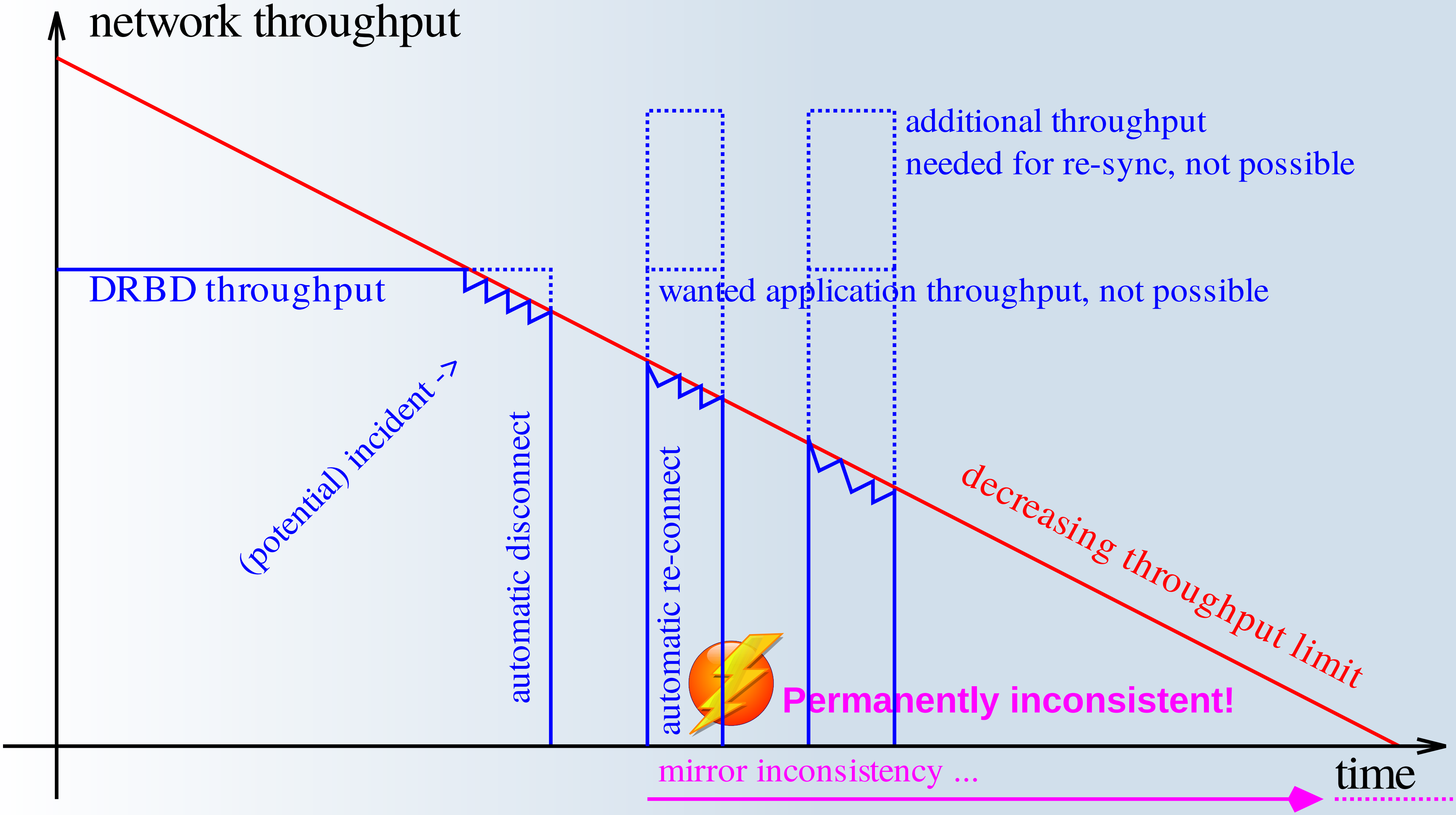
violated at
- disasters
- LONG distances

A = Availability

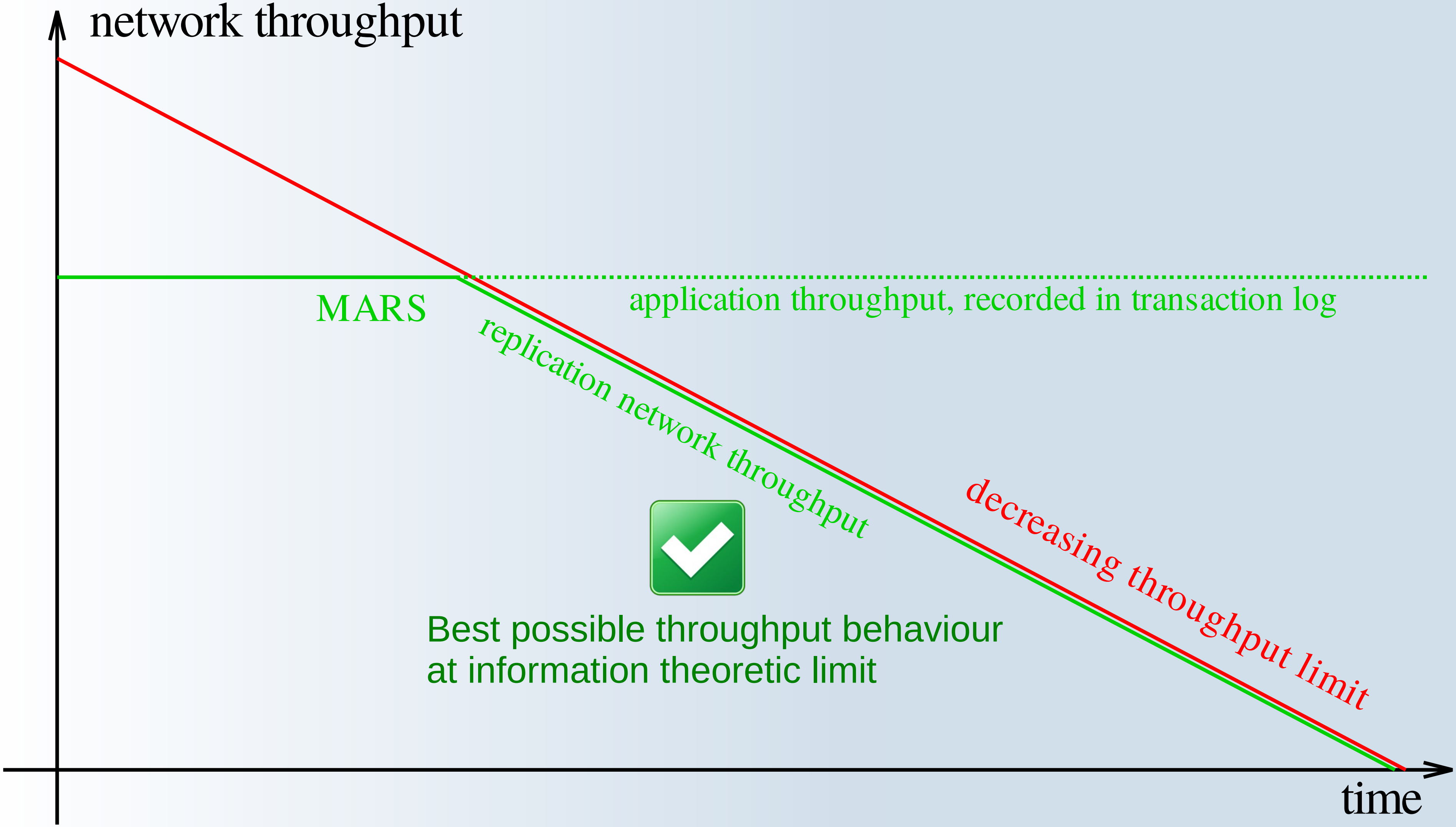
P = Partitioning Tolerance

= the network can have its own outages

Network Bottlenecks (1) DRBD

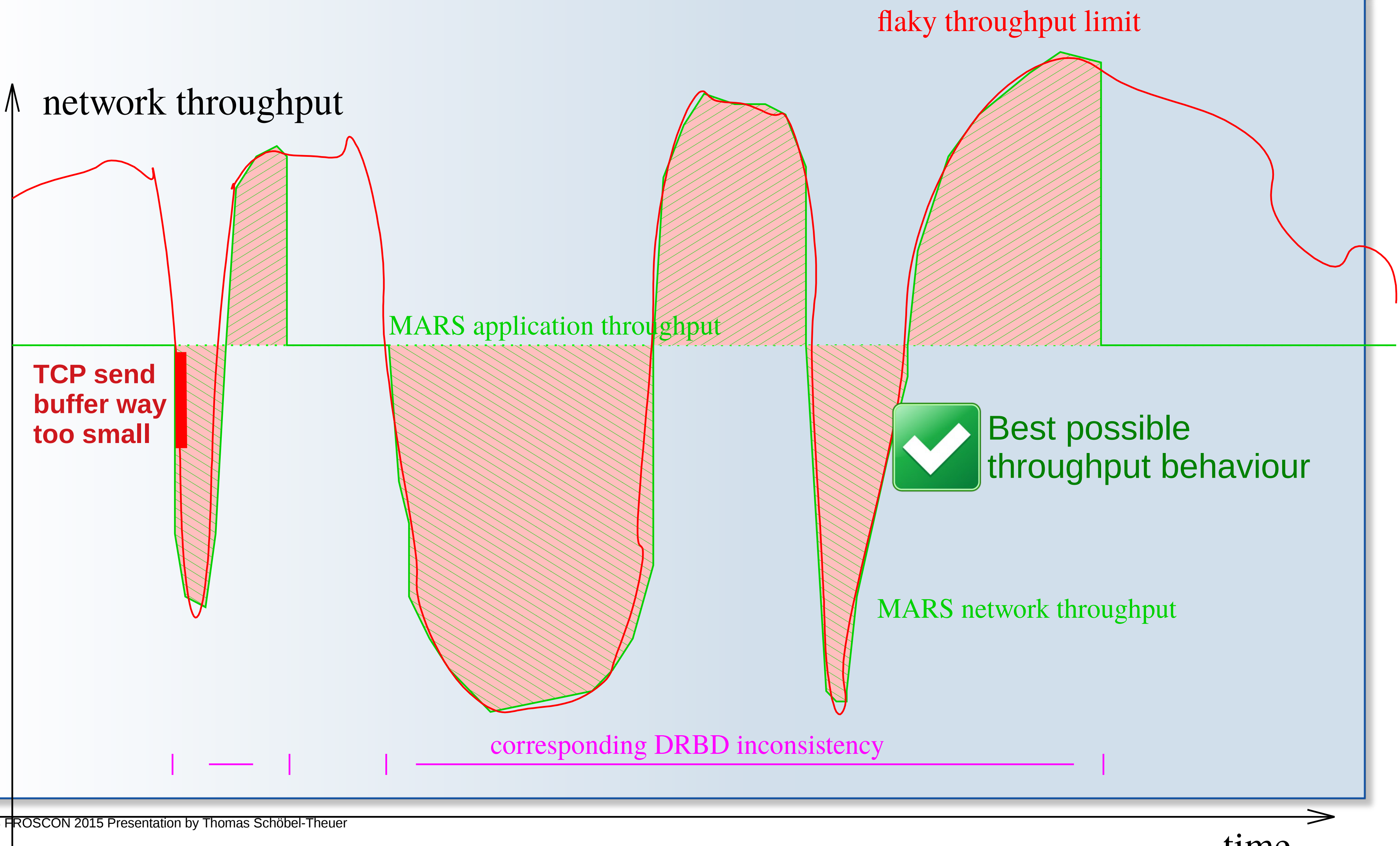


Network Bottlenecks (2) MARS

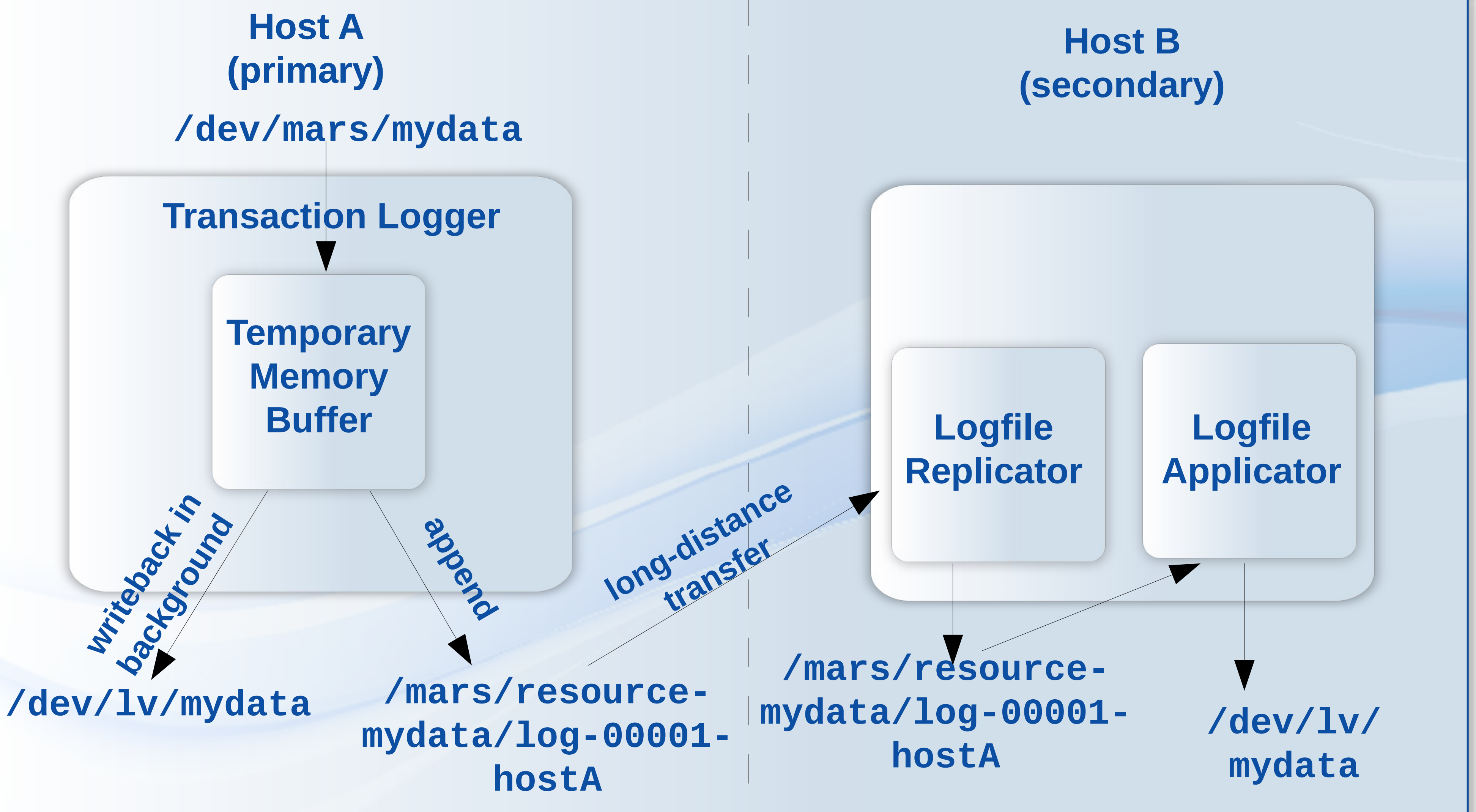


Best possible throughput behaviour
at information theoretic limit

Network Bottlenecks: MARS



MARS Data Flow Principle



Use Cases DRBD+proxy vs MARS

DRBD+proxy
(proprietary)

Application area:

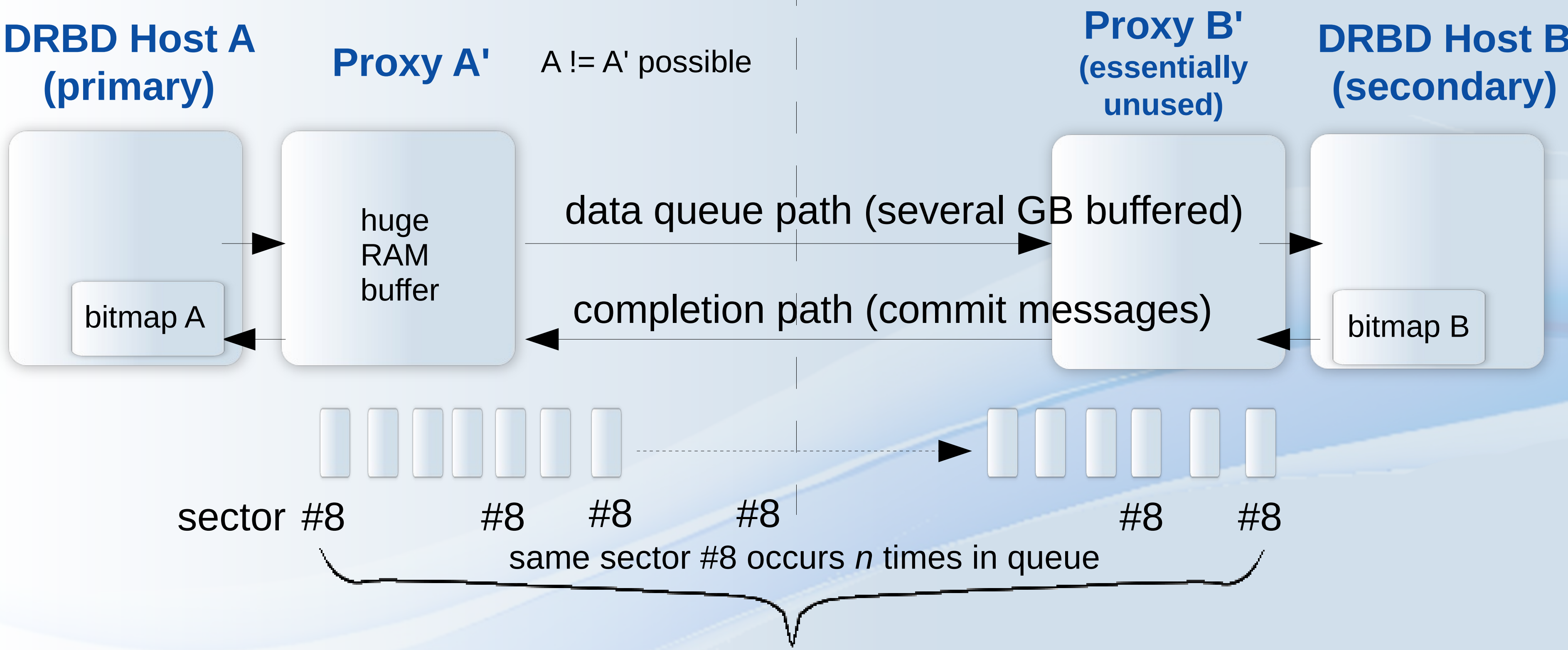
- Distances: any
- Aynchronously
  - **Buffering in RAM**
- Unreliable network leads to **frequent re-syncs**
  - RAM buffer gets lost
  - at cost of actuality
- **Long** inconsistencies during re-sync
- Under pressure: **permanent** inconsistency possible
- High memory overhead
- Difficult scaling to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

MARS
(GPL)

Application area:

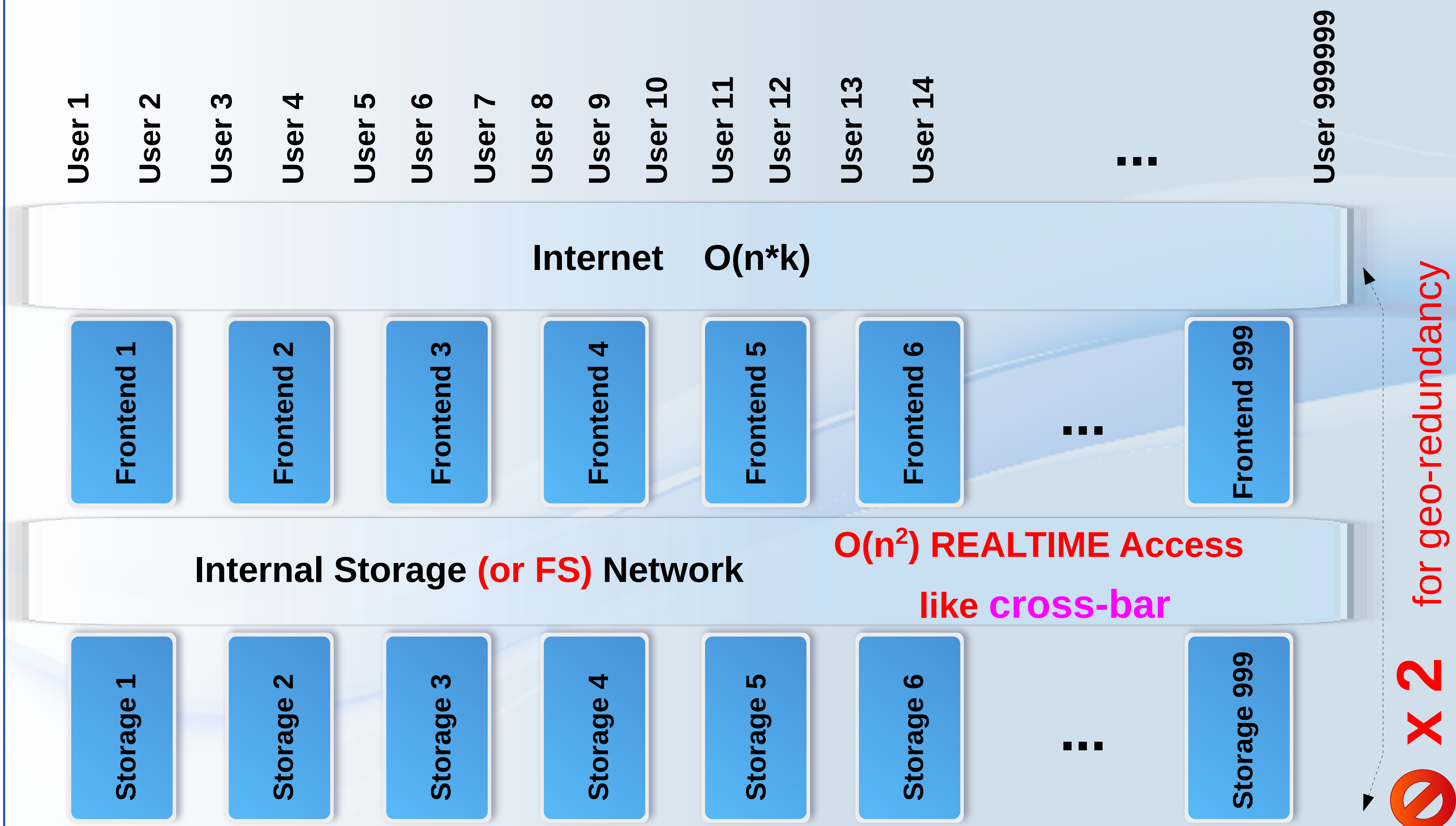
- Distances: **any** (>>50 km)
- Asynchronously
  - near-synchronous modes in preparation
- Tolerates **unreliable network**
- Anytime consistency
  - no re-sync
- Under pressure: no inconsistency
  - possibly at cost of actuality
- Needs >= 100GB in `/mars/` for transaction logfiles
  - dedicated spindle(s)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

DRBD+proxy Architectural Challenge

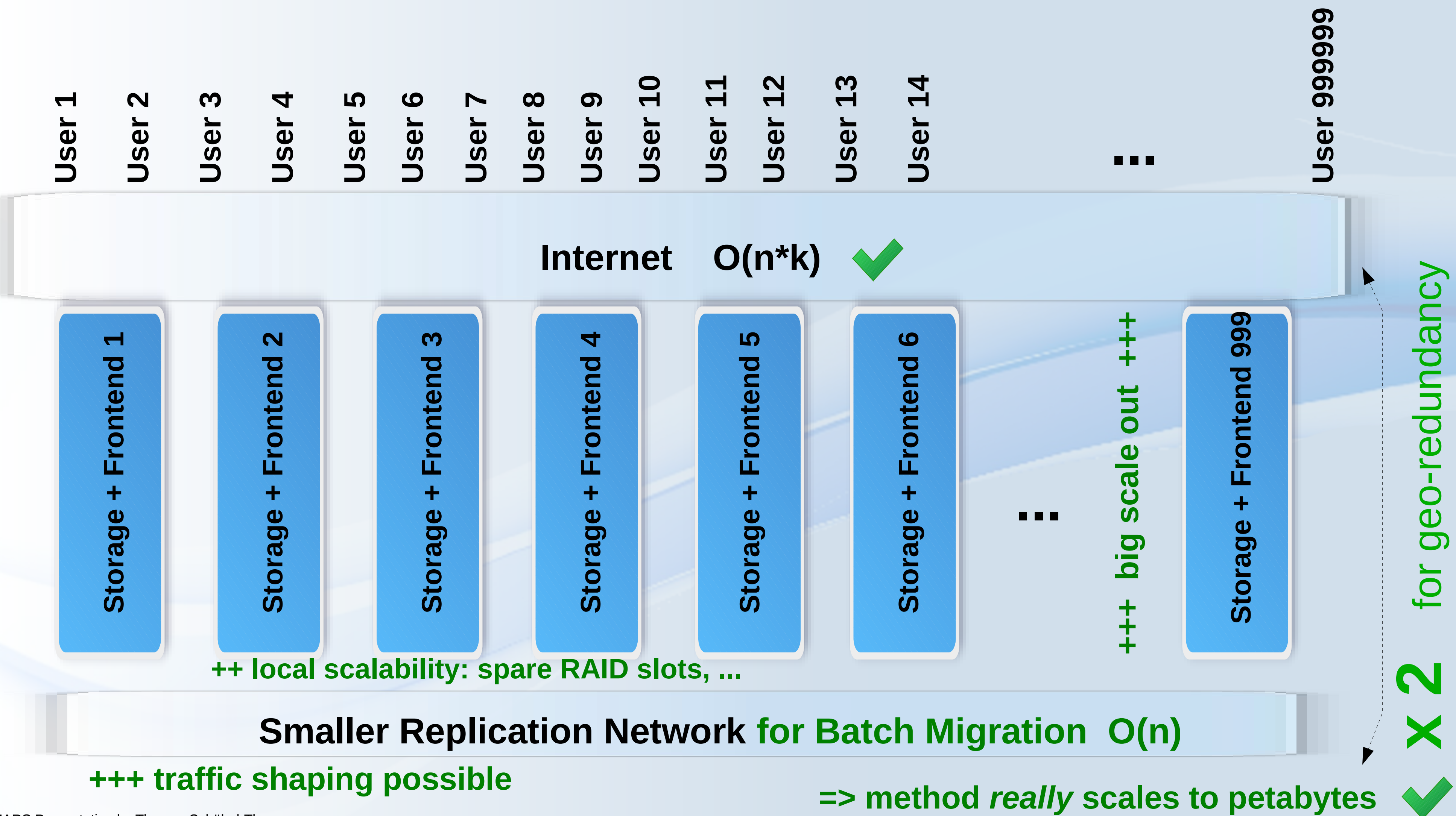


n times
=> need *log*(n) bits for counter
=> but DRBD bitmap has only 1 bit/sector
=> workarounds exist, but complicated
(e.g. additional dynamic memory)

Badly Scaling Architecture: Big Cluster



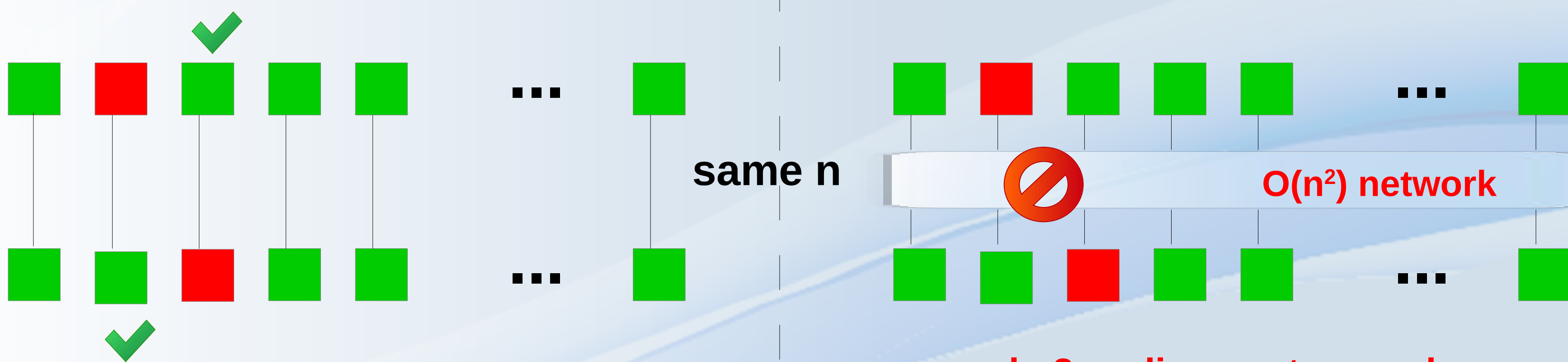
Well-Scaling Architecture: Sharding



Reliability of Architectures: NODE failures

2 Node failure => ALL their disks are unreachable

DRBD or MARS simple pairs | **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 >= 3 replicas here

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

- **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) georedundant => 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

- 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

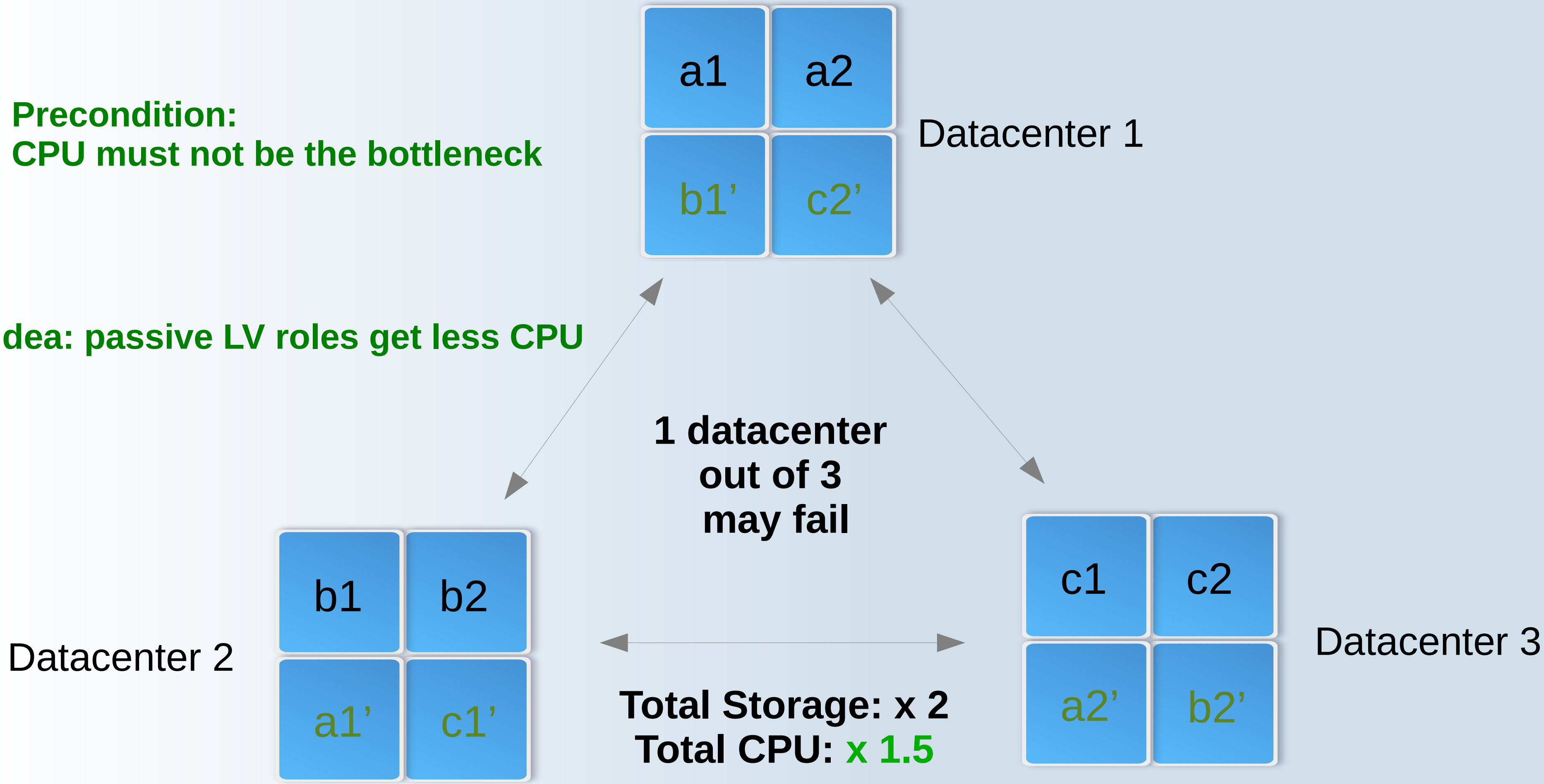
  + O(n) replication network
- Disks: < 240%
- Less total power
- Less total HU

  +++ geo failure scenarios

Cost (3): Geo-Redundancy even Cheaper

Precondition:
CPU must not be the bottleneck

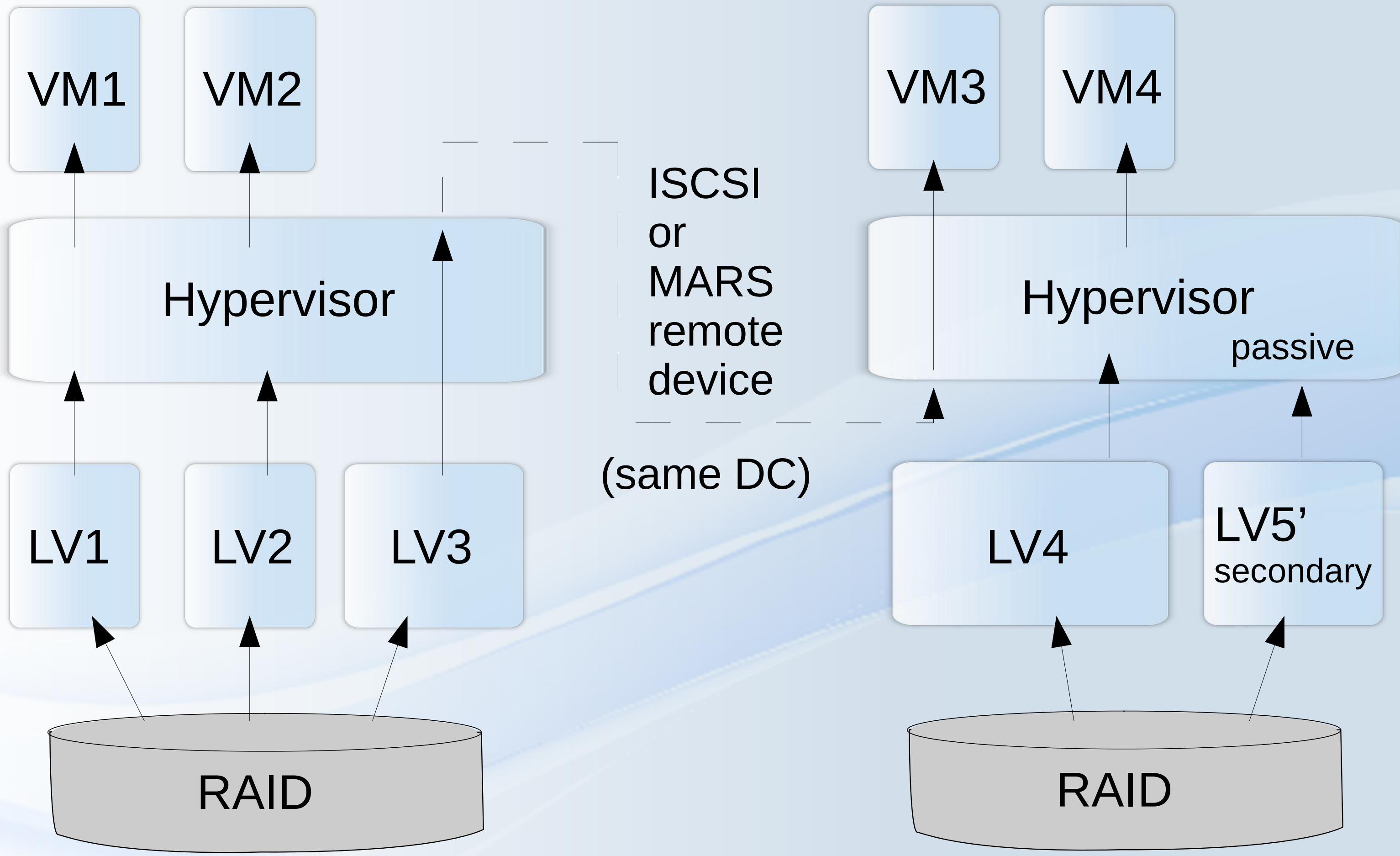
Idea: passive LV roles get less CPU



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

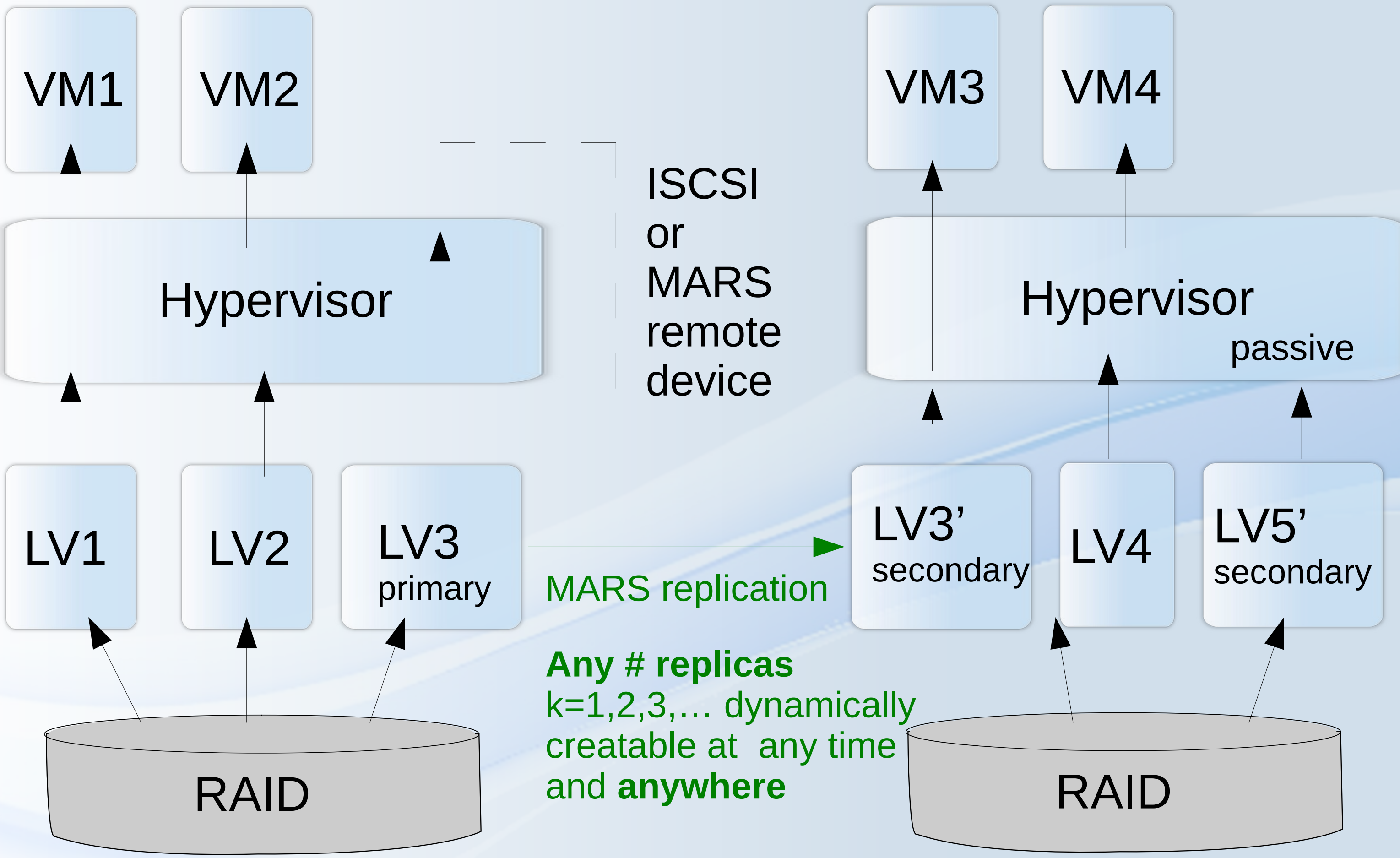
HOWTO flexible CPU assignment => next slide

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 football sub-project



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