Container Football

using MARS on Enterprise-Critical Data

FrOSCon 2018 Presentation by Thomas Schöbel-Theuer

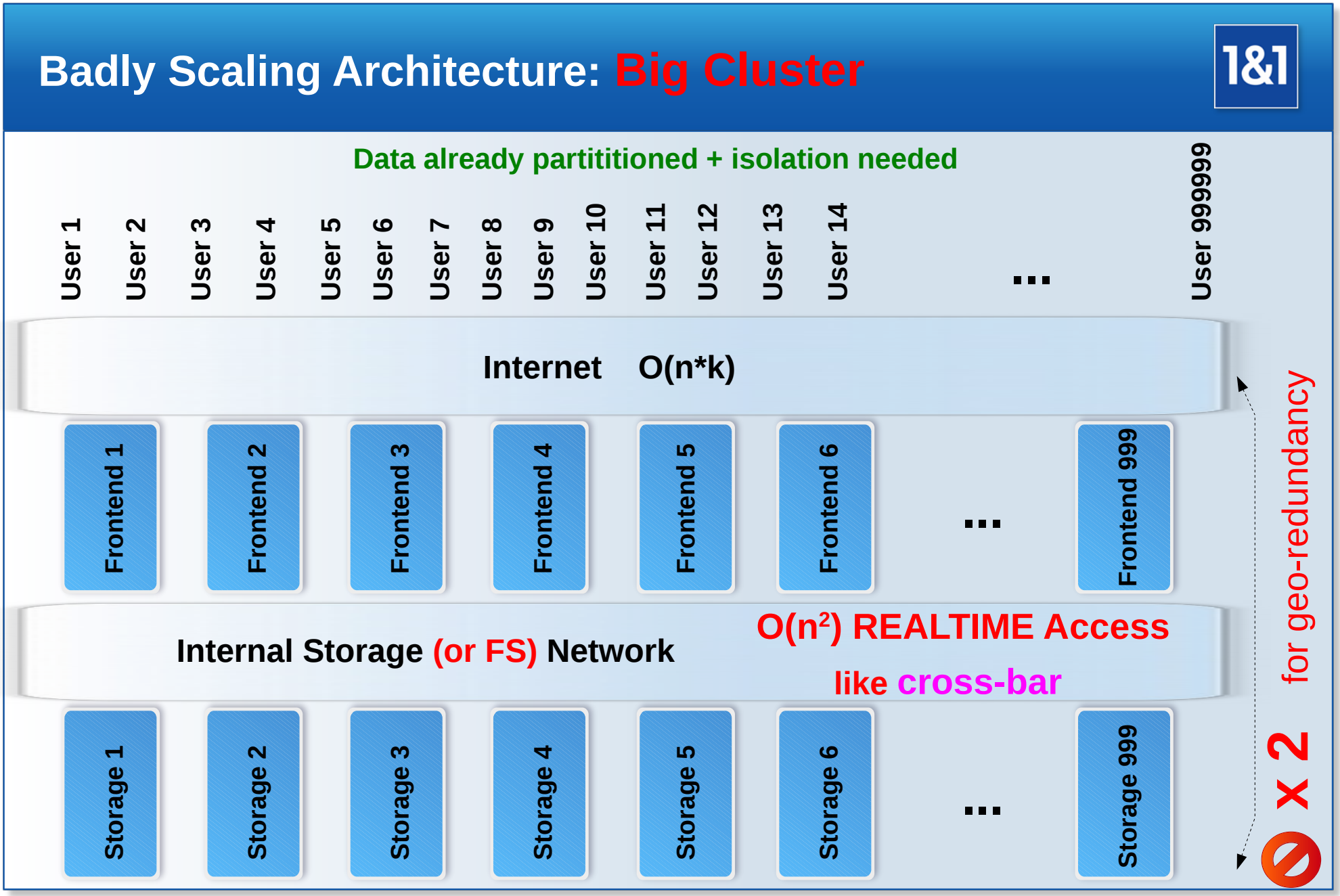
Container Football: Agenda

New method for load balancing

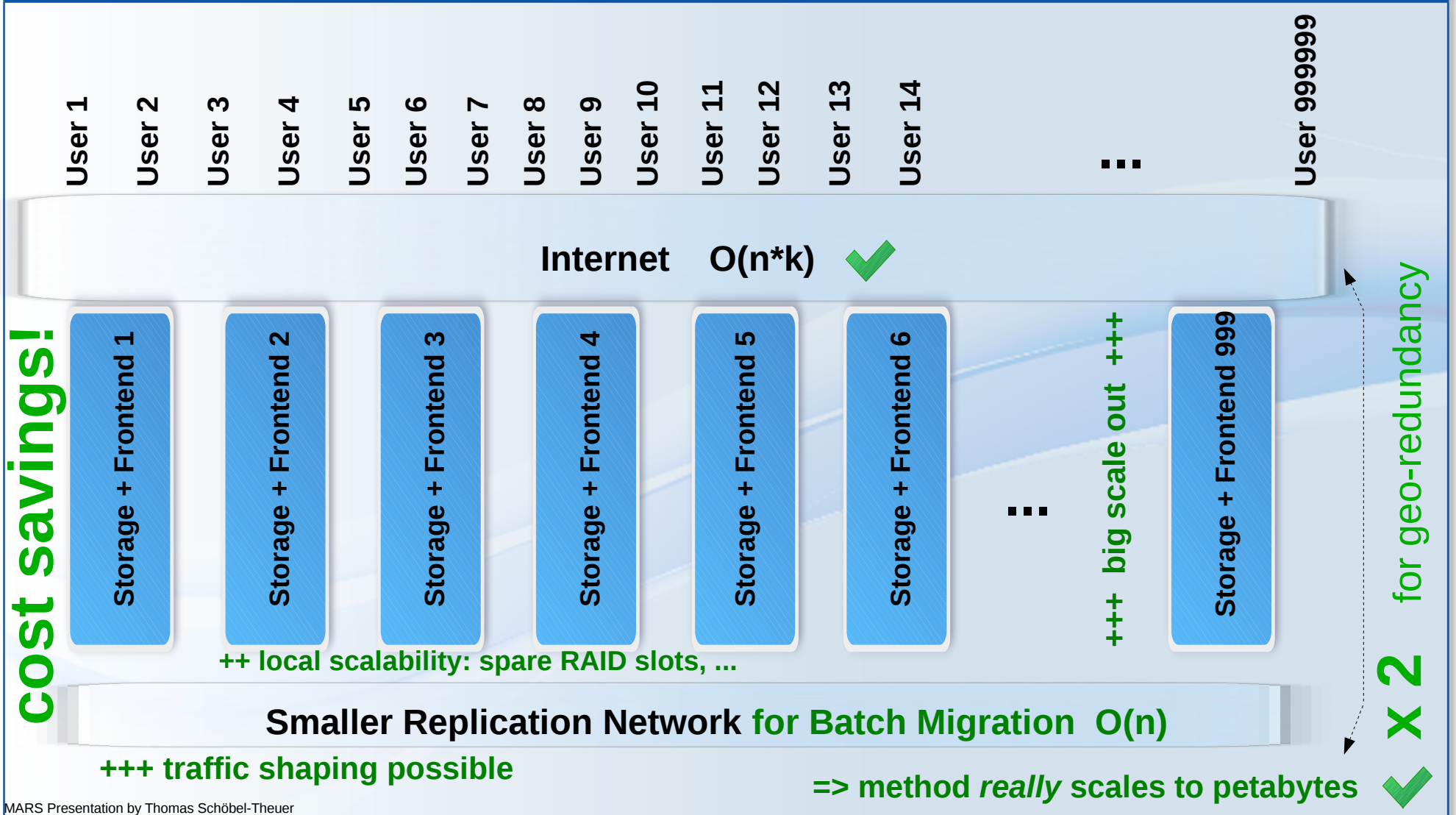
- Motivation: Scalability of Storage Architectures
- Motivation: Reliability Unexpected properties!
- HOWTO Container Football = Background Migration of LVs

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

Badly Scaling Architecture: Big Cluster



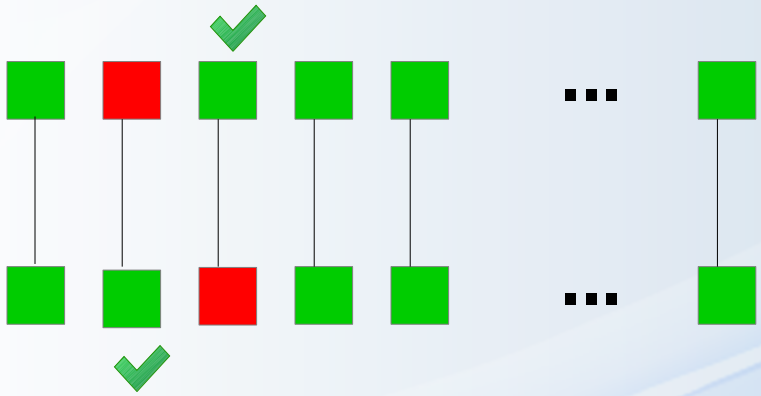
Well-Scaling Architecture: Sharding



Reliability of Architectures: NODE failures

2 Node failure => ALL their disks are unreachable

DRBD or MARS simple pairs

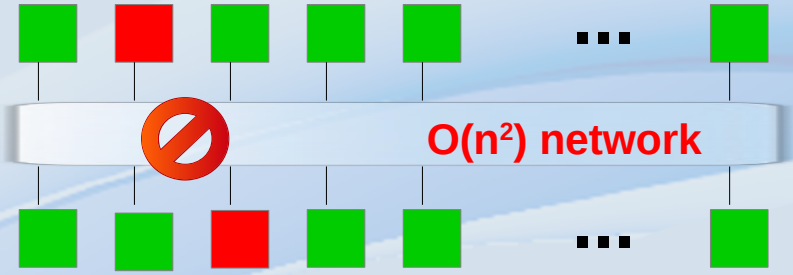


=> no customer-visible incident

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

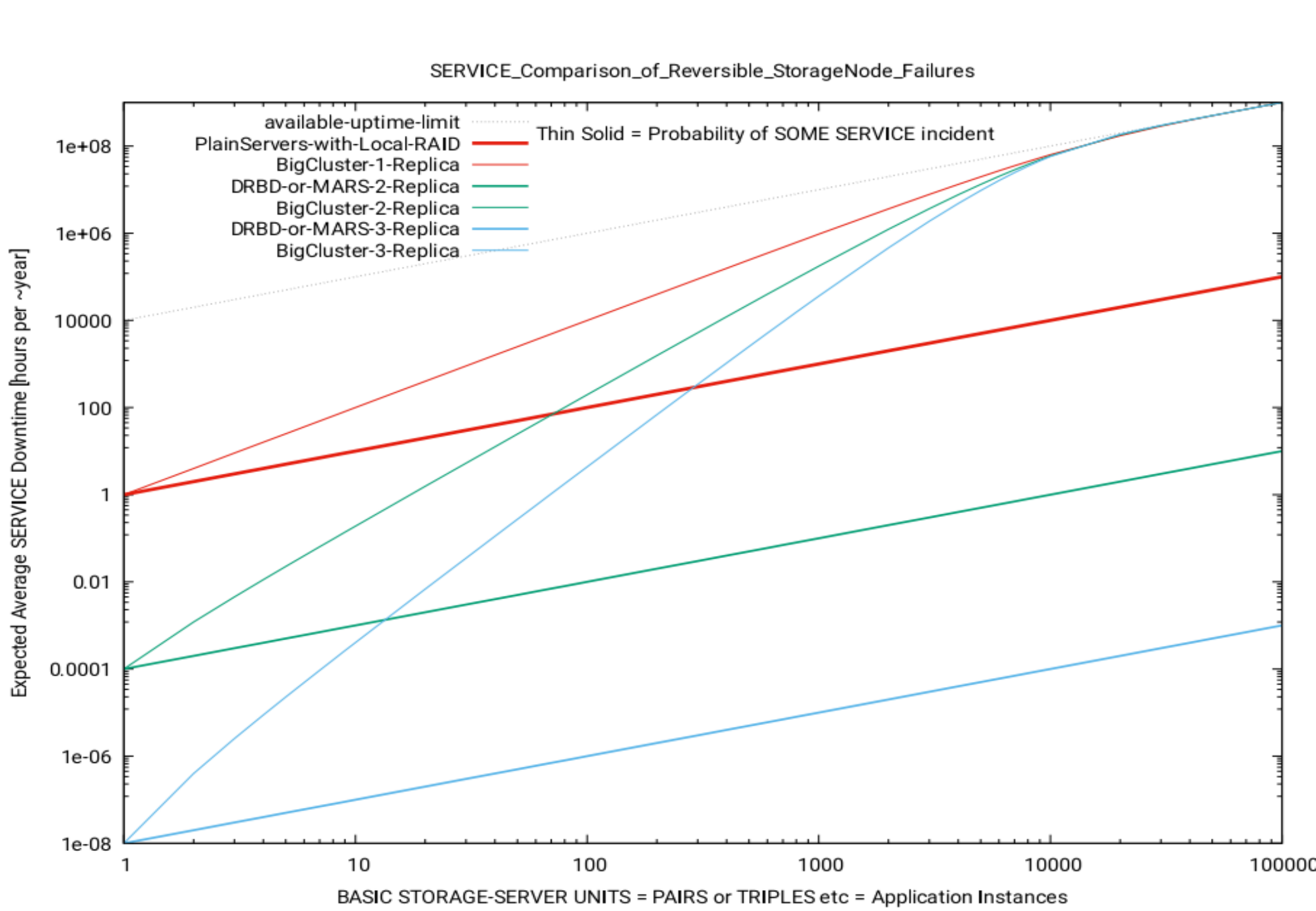
Example: Reliability Scenario

- Assumptions:
  - 1 Server has 99.99 % uptime

    => 1 hour downtime per 10.000 operating hours

    ≈ 13 months ≈ 1 year
  - Only temporary failures
  - No dependencies between servers
  - BigCluster: all objects spread to all servers
  - Sharding (DRBDorMARS): simple pairs / triples / ...
- 10000 servers => always 1 of 10000 is down in average

  Comparion => next slide


SERVICE_Comparison_of_Reversible_StorageNode_Failures
available-uptime-limit
PlainServers-with-Local-RAID
BigCluster-1-Replica
DRBD-or-MARS-2-Replica
BigCluster-2-Replica
DRBD-or-MARS-3-Replica
BigCluster-3-Replica
Thin Solid = Probability of SOME SERVICE incident
Expected Average SERVICE Downtime [hours per ~year]
1e+08
1e+06
10000
100
1
0.01
0.0001
1e-06
1e-08
1
10
100
1000
10000
100000
BASIC STORAGE-SERVER UNITS = PAIRS or TRIPLES etc = Application Instances

Fundamental Law

- Mathematical proof at `mars-manual.pdf`
  - motivated by practical experiences with 1&1 Ceph clusters
- Sharding with pairs / triples / etc has the

  BEST POSSIBLE RELIABILITY.
- BigCluster is **never better.**
  - BigCluster is not usable in important dimensioning cases
    - even worse when adding storage network outages, frontend node failures, permanent failures / disasters, etc.
      - Workaround: spread objects to O(k) instead of to O(n) storage nodes but even worse than Sharding
      - See also USENIX paper on Copysets
- Contrary to some internet propaganda / common belief

Common Belief

- Sharding is inflexible / no load balancing possible???
  - therefore storage networks a „must“???
- Yes, maybe in the past
- NO LONGER in future => see new Football method
  - VM Football / Container Football / LV Football / ...

Common belief changes only slowly
But fundamental laws of physics / mathematics are stronger

HOWTO Container Football = Background Migration of LVs

HOST A (old) VM is running → HOST B (new) has spare space

- lvdisplay /dev/vg/$mydata
-
-
- (meanwhile VM is altering data)
- $vmmanager stop /dev/mars/$mydata
-
-
- marsadm leave-resource $mydata
- lvremove /dev/vg/$mydata

-
- lvcreate -L $size -n $mydata vg
- marsadm join-resource $mydata \
  /dev/vg/$mydata
- marsadm view: wait for UpToDate
-
- marsadm primary $mydata
- $vmmanger start /dev/mars/$mydata
-

start kick

done kick

cleanup kick

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

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

Highlevel: Planner vs Optimizer

Dependencies

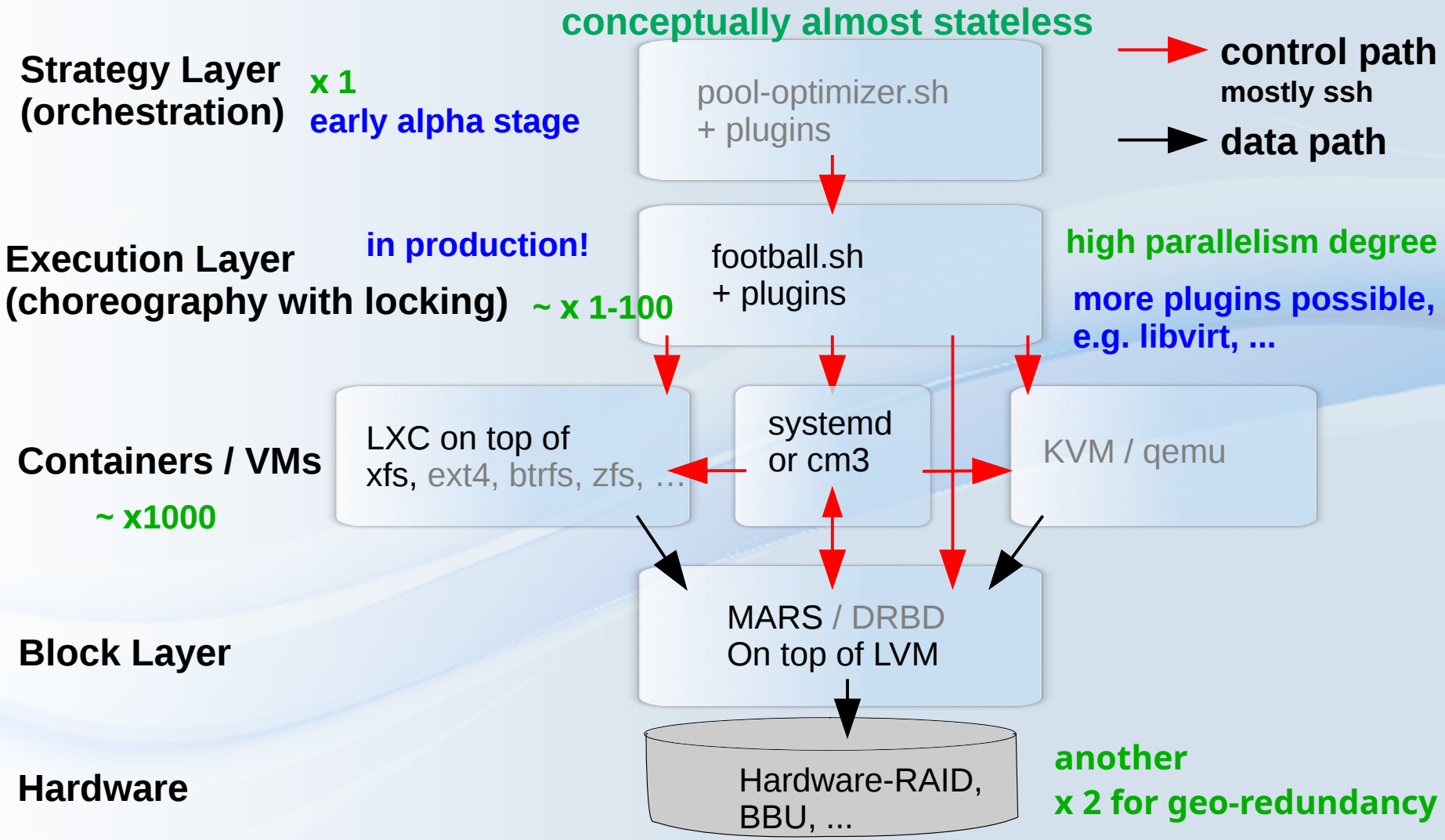
- **Planner: produces stateful plan**

Complexity = O(|state|^2)

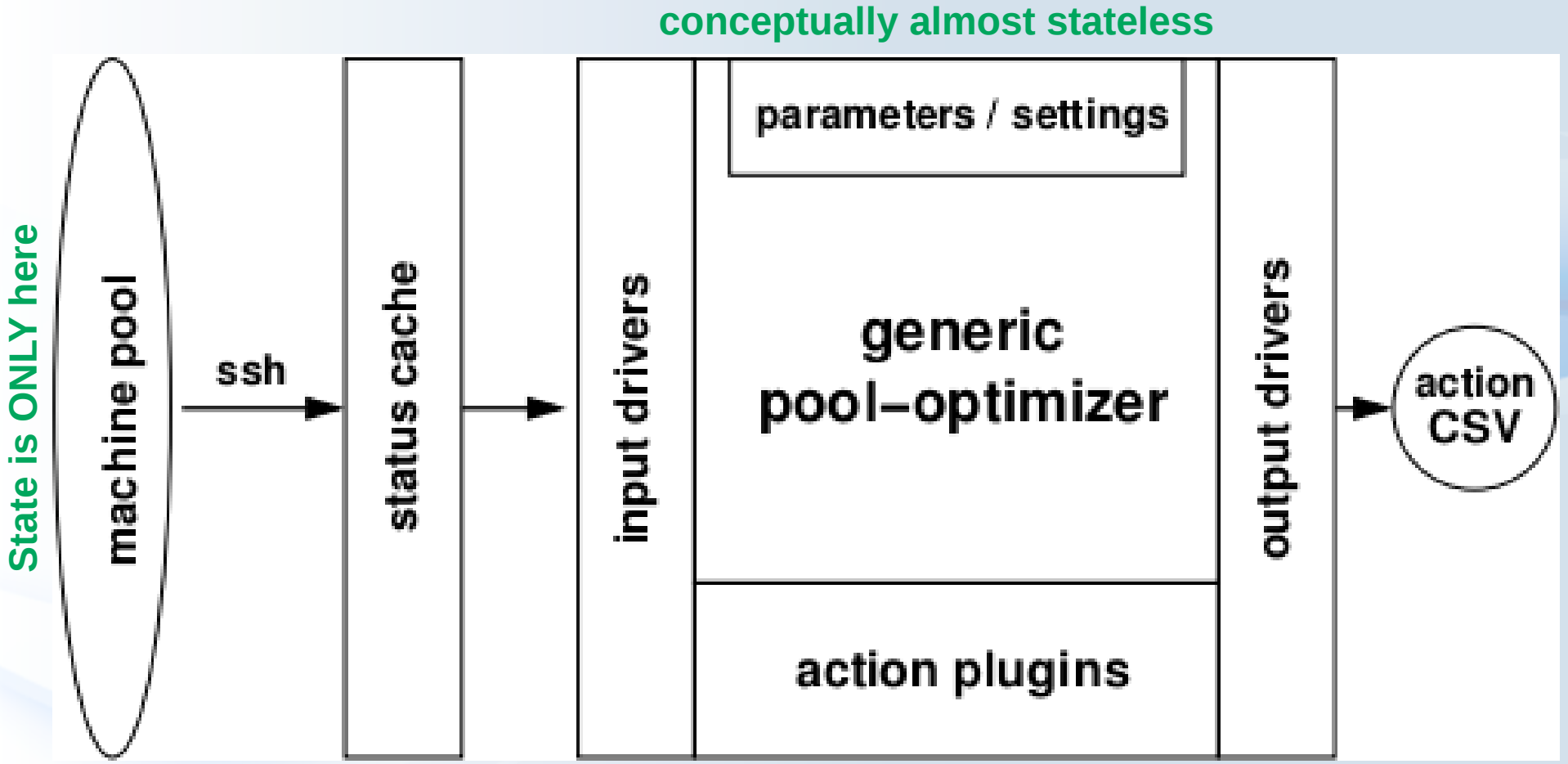
NO dependencies

- **Optimizer: produces stateless actions**
  - **works like a CONTROLLER LOOP**
  - **similar to Kubernetes**

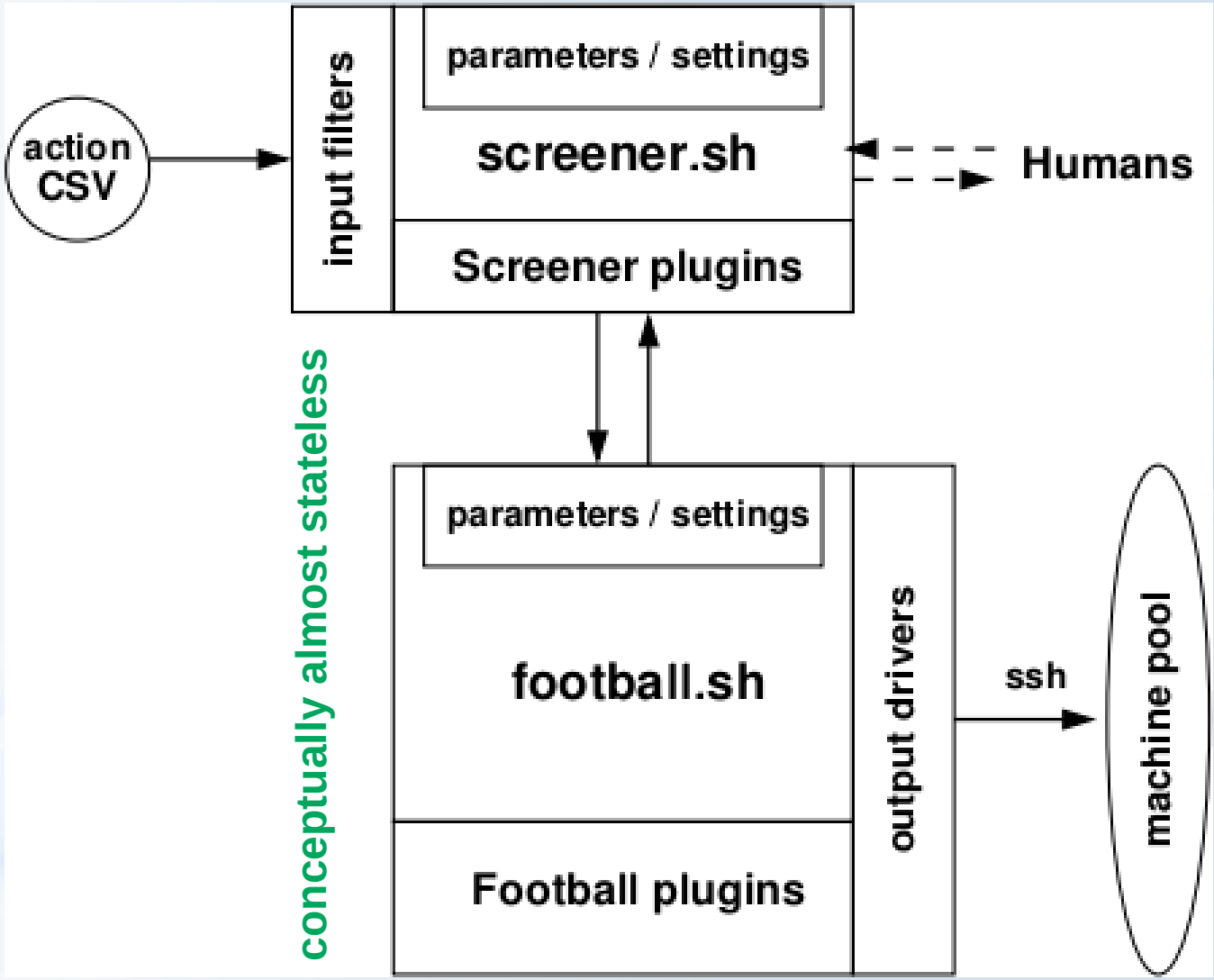
Football Architecture (not yet completed)



Pool-optimizer (early alpha stage)



football.sh (in production with cm3 plugin)



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 (few minutes)
  - shrink $vm $target_percent
    - uses local rsync, minimizes downtime
  - expand $vm $target_percent
    - online, no downtime
  - migrate+shrink
    - consumes less network traffic
- In production at internal Efficiency project
  - get rid of old hardware
  - Concentrate ~ 7 LXC containers on 1 hypervisor
  - currently >40 „kicks“ per week
    - limited by hardware deployment speed
    - Proprietary Planner (for HW lifecycle)

Sponsoring (MARS + Football)

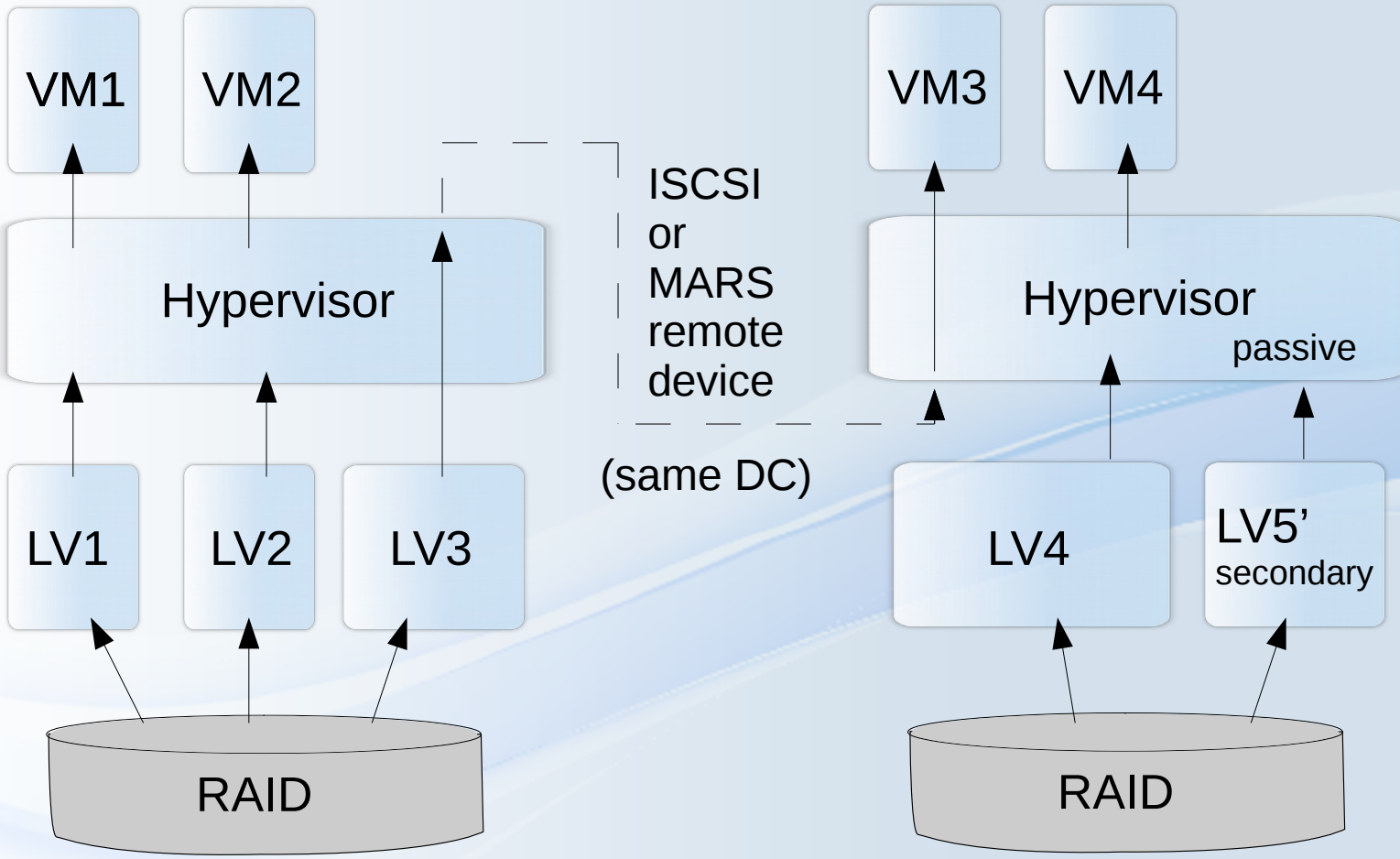
- Best for > 1 PiB of enterprise-critical data
  - Example: currently ShaHoLin has > 4PiB total allocated (df)
    - + much more at LVM layer
    - thousands of LXC instances => also KVM possible in future
- Pool-optimizer will deliver similar functionality than **Kubernetes**
  - but on stateful storage 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
- ask me: decades of experience with enterprise-critical data and their 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

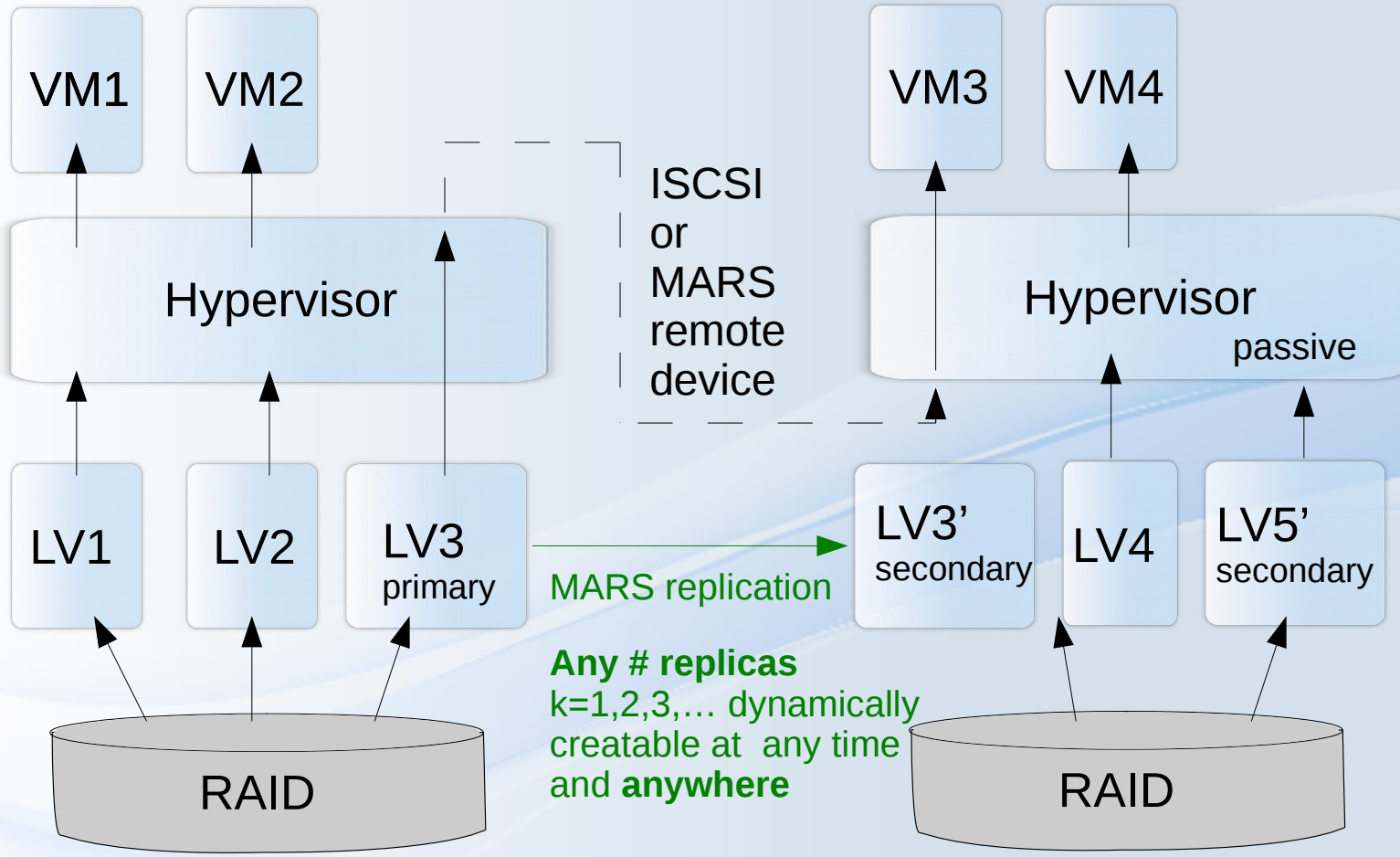
Flexible MARS Sharding + Cluster-on-Demand



any hypervisor works in client and/or server role

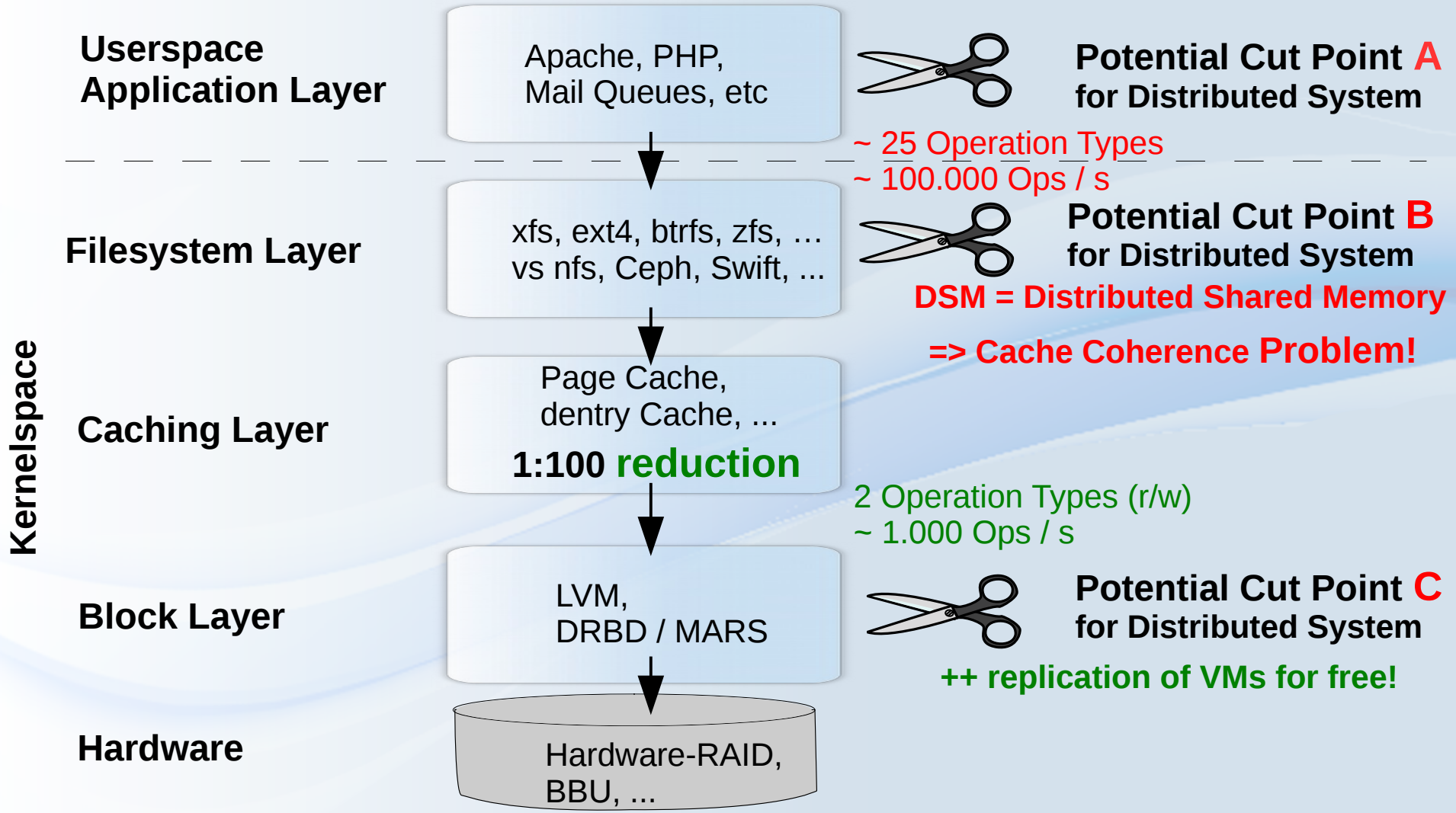
and preferably **locally** at the same time

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



Use Cases DRBD+proxy vs MARS Light

DRBD+proxy
(proprietary)

Application area:

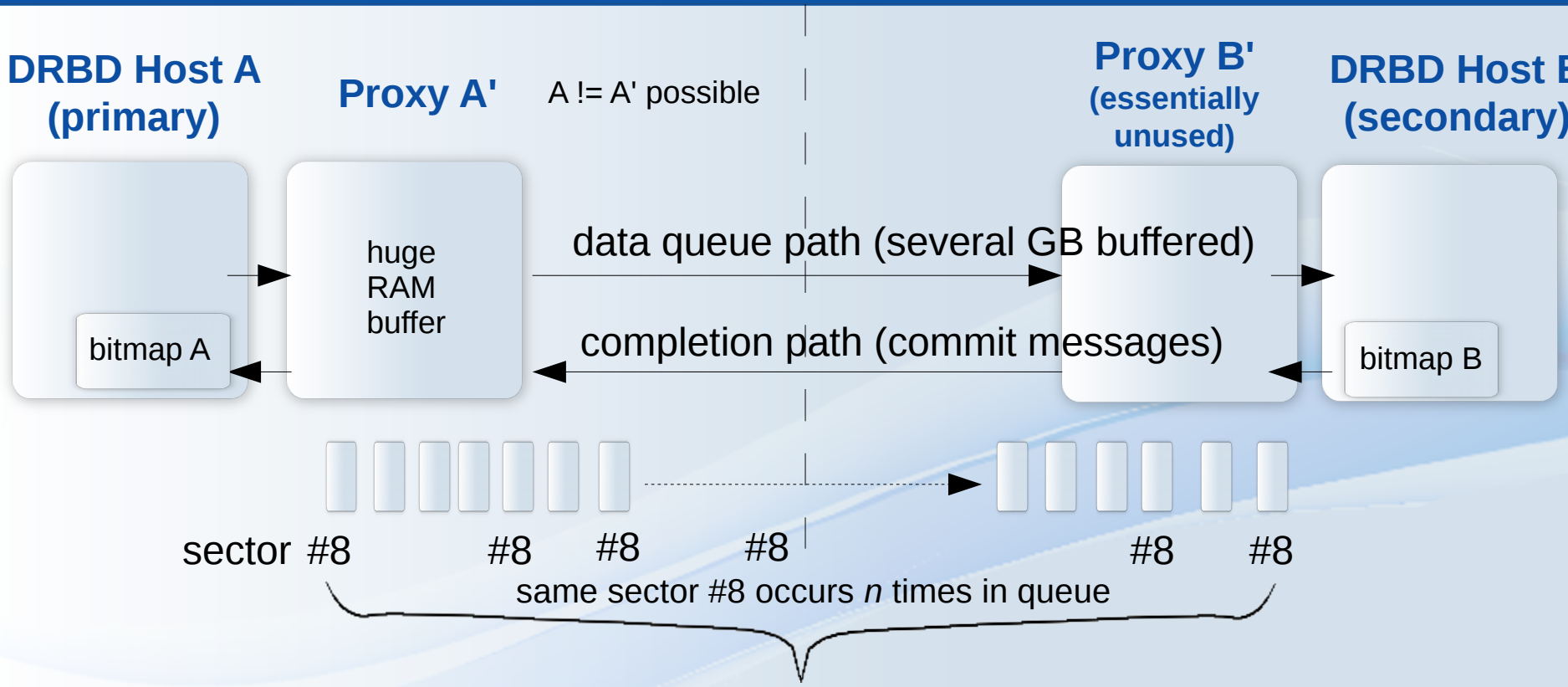
- Distances: any
- Aynchronously
  - **Buffering in RAM**
- Unreliable network leads to **frequent re-syncs**
  - RAM buffer gets lost
  - at cost of actuality
- **Long** inconsistencies during re-sync
- Under pressure: **permanent** inconsistency possible
- High memory overhead
- Difficult scaling to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

MARS Light
(GPL)

Application area:

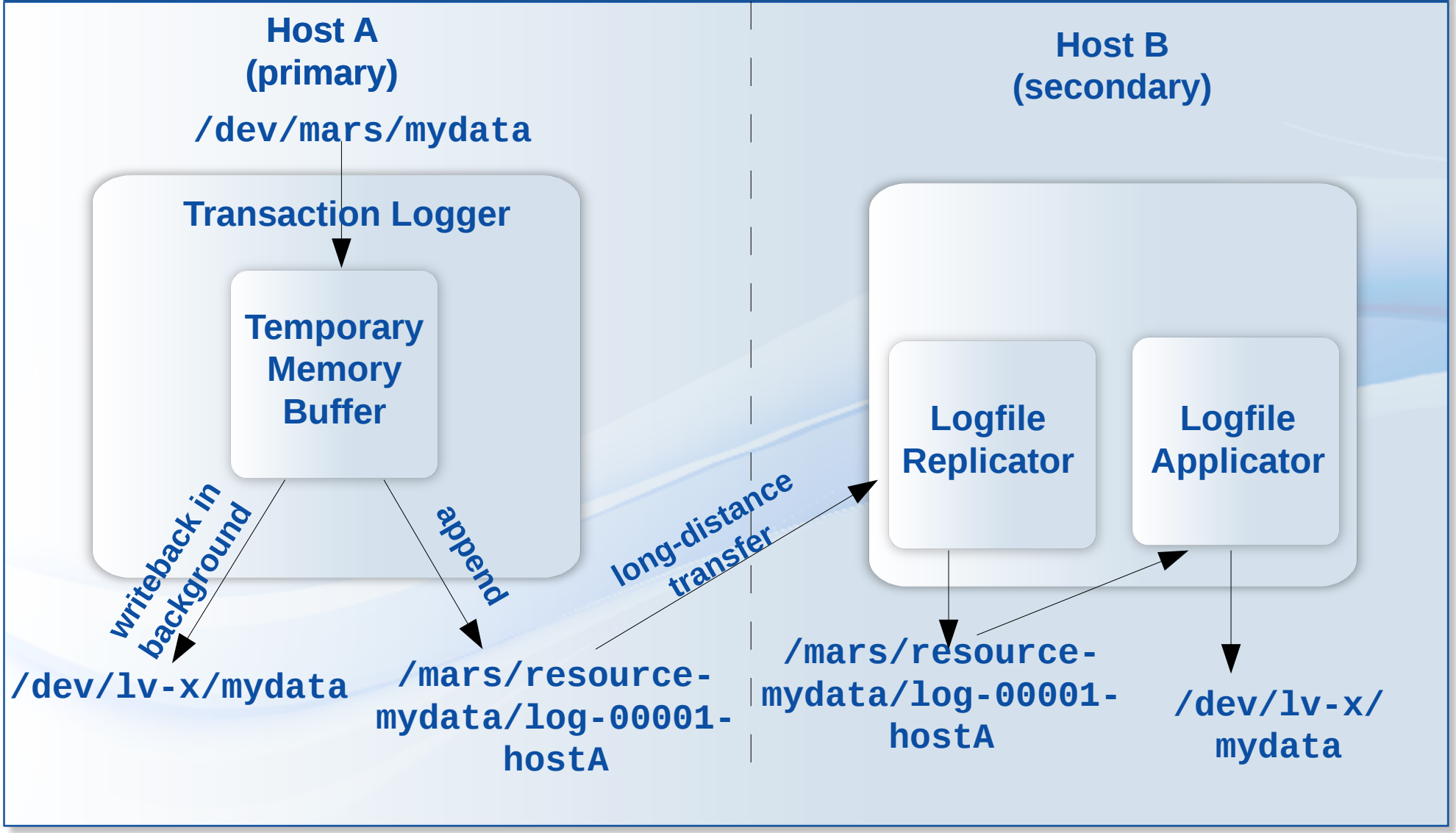
- Distances: **any** (>>50 km)
- Asynchronously
  - near-synchronous modes in preparation
- Tolerates **unreliable network**
- Anytime consistency
  - no re-sync
- Under pressure: no inconsistency
  - possibly at cost of actuality
- Needs >= 100GB in `/mars/` for transaction logfiles
  - dedicated spindle(s)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