



Parallel Data Laboratory
Carnegie Mellon University

Pacemaker

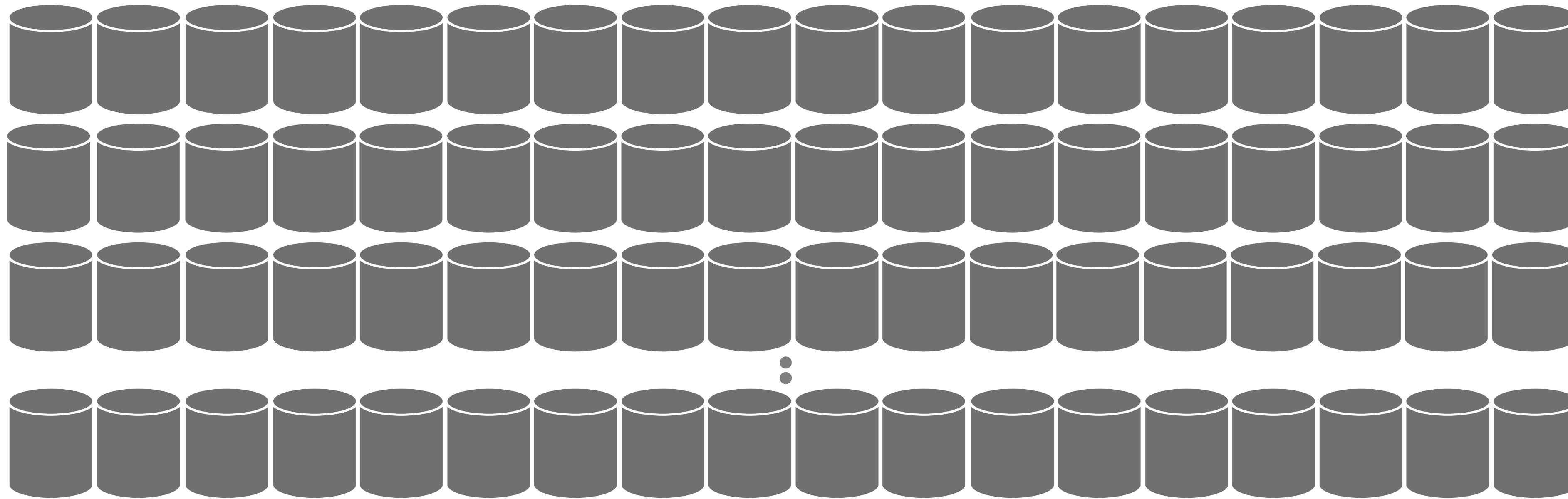
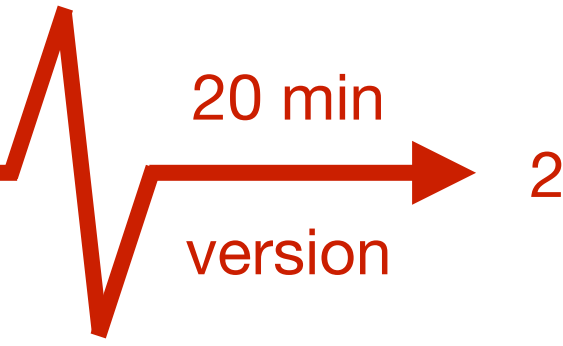
USENIX OSDI 2020

Avoiding HeART attacks in storage clusters
with disk-adaptive redundancy

20 min version

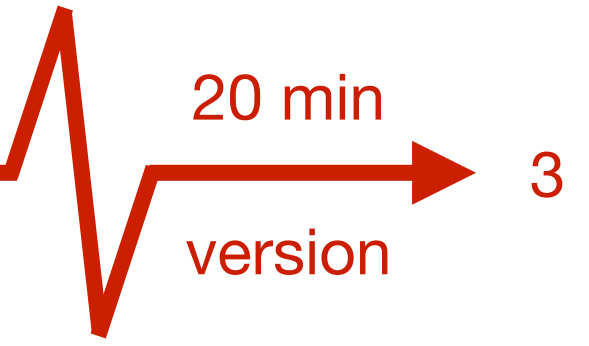
Saurabh Kadekodi, Francisco Maturana, Suhas Jayaram Subramanya, Juncheng Yang,
K. V. Rashmi, Greg Ganger

Today's storage clusters

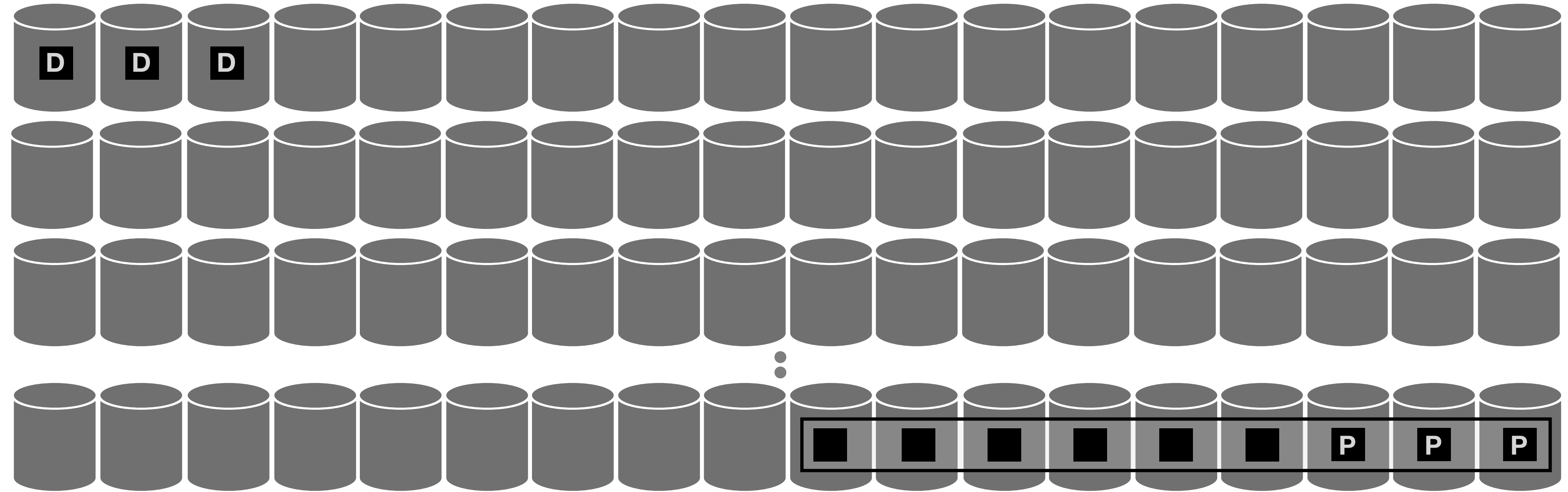


- Thousands to millions of disks in primary storage tier
- Failures common in today's cluster storage systems
 - Disk failures measured as **annualized failure rates (AFR)**
 - AFR = expected % of failures in a year

Data redundancy for fault tolerance



3-replication

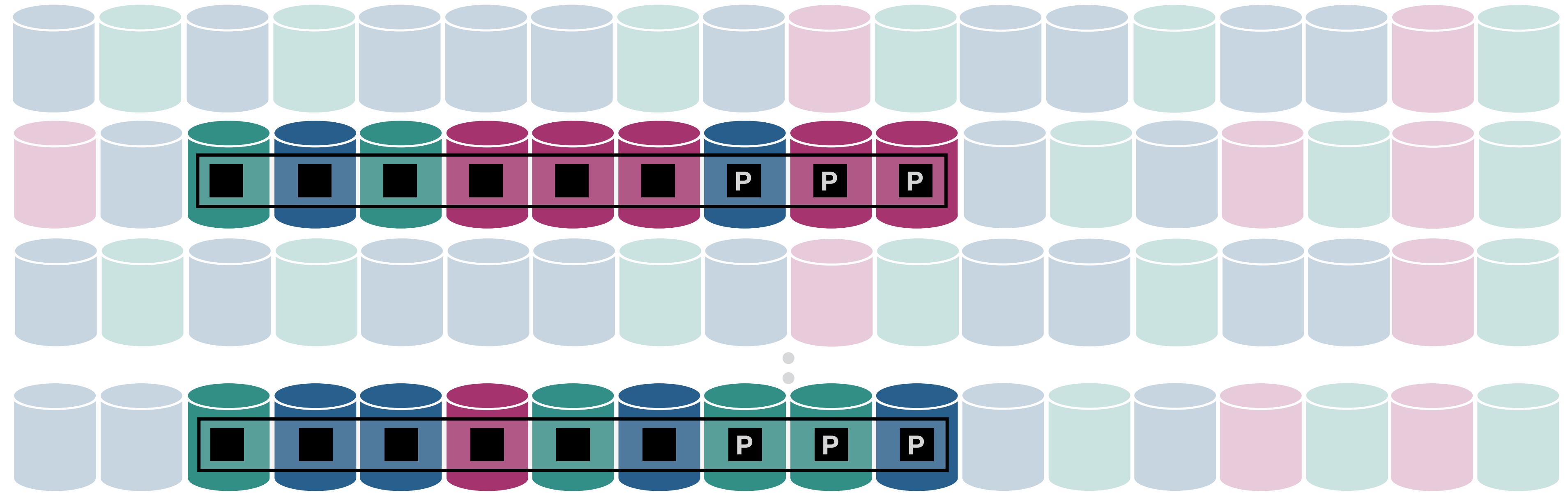
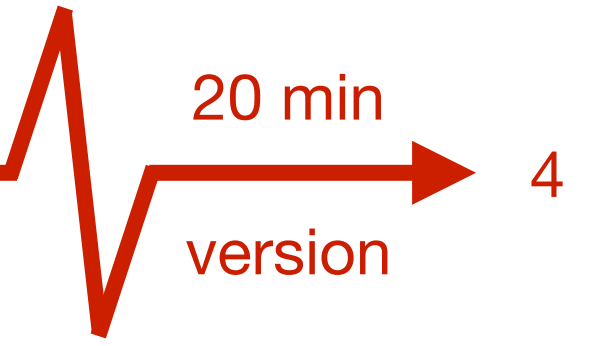


6-of-9 erasure code (6 data, 3 parities)

- Data redundancy is used to protect against data loss
- Multiple redundancy scheme may be used in the entire fleet

Redundancy scheme unaware of AFR differences among disks

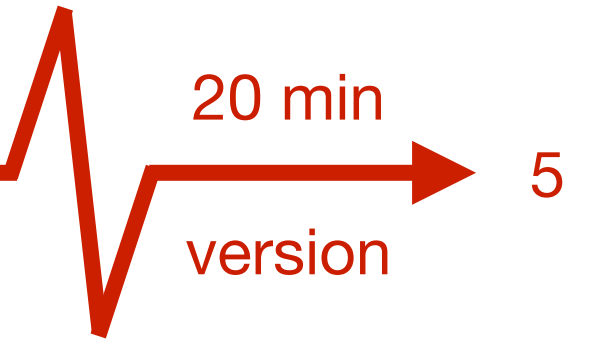
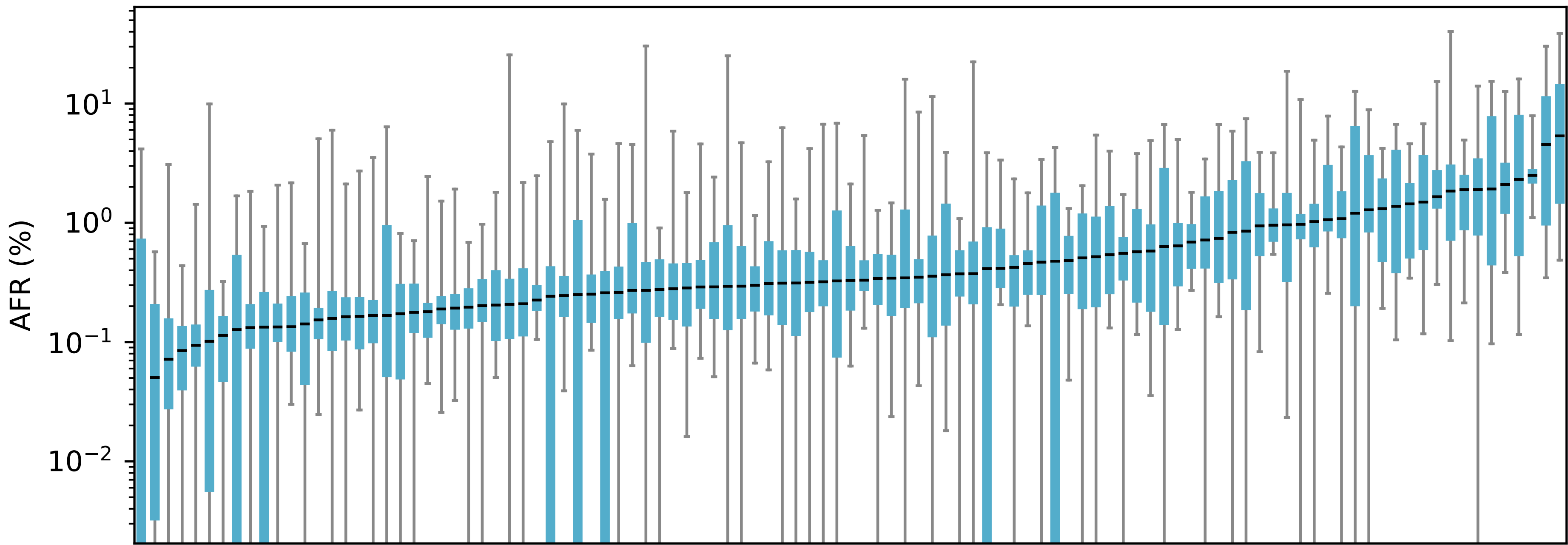
Reality: different disks fail differently



- Single storage cluster may have multiple makes/models
- Result: stripes (or replicas) may provide different reliability

Same redundancy is either insufficient or overly wasteful

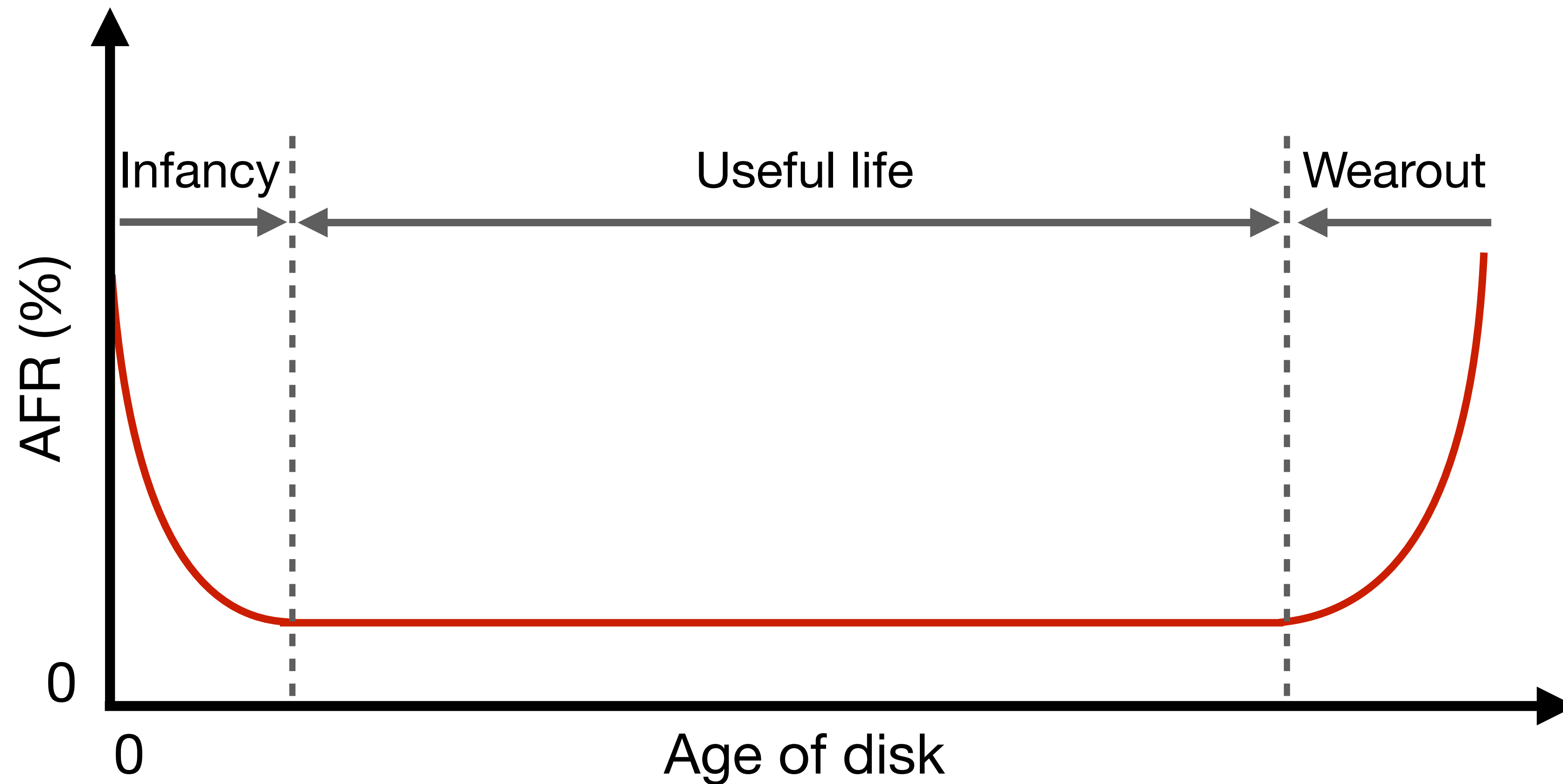
How much do failure rates vary?



- Totally over 5.3 million HDDs, across over 60 makes/models
- Deployed in production environments at NetApp, Google, Backblaze
- Each box represents a make/model with at least 10000 HDDs

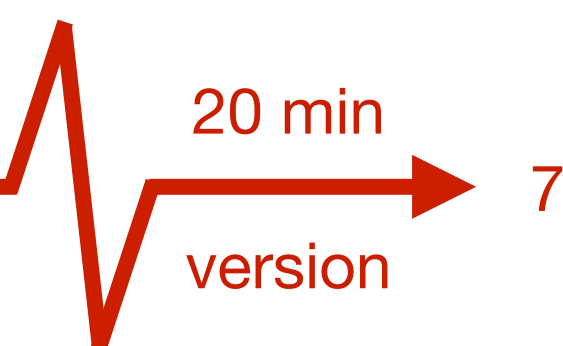
Over 10x difference in failure rates across makes/models

The disk hazard (bathtub) curve

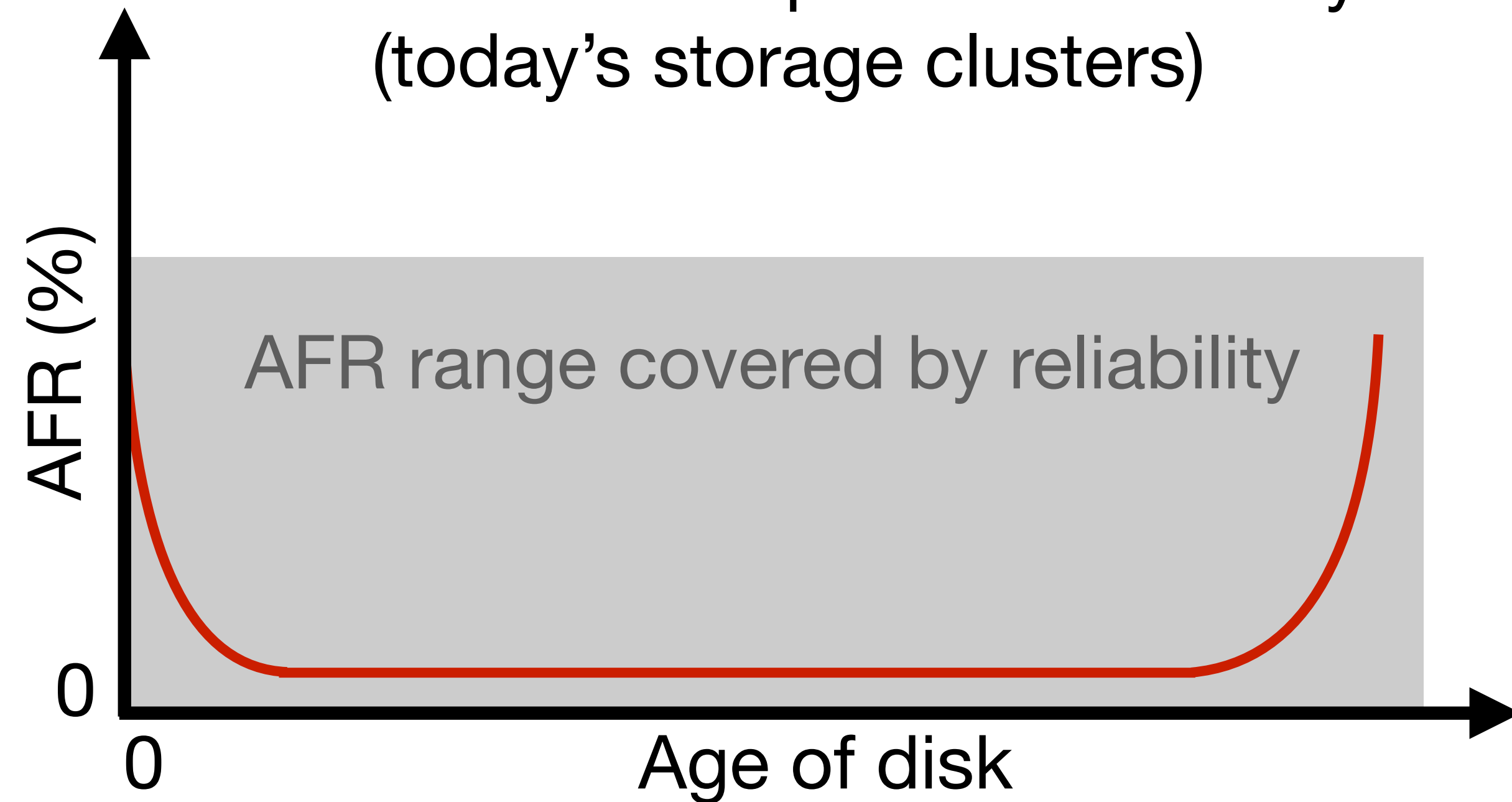


Failure rate varies over a disk's lifetime

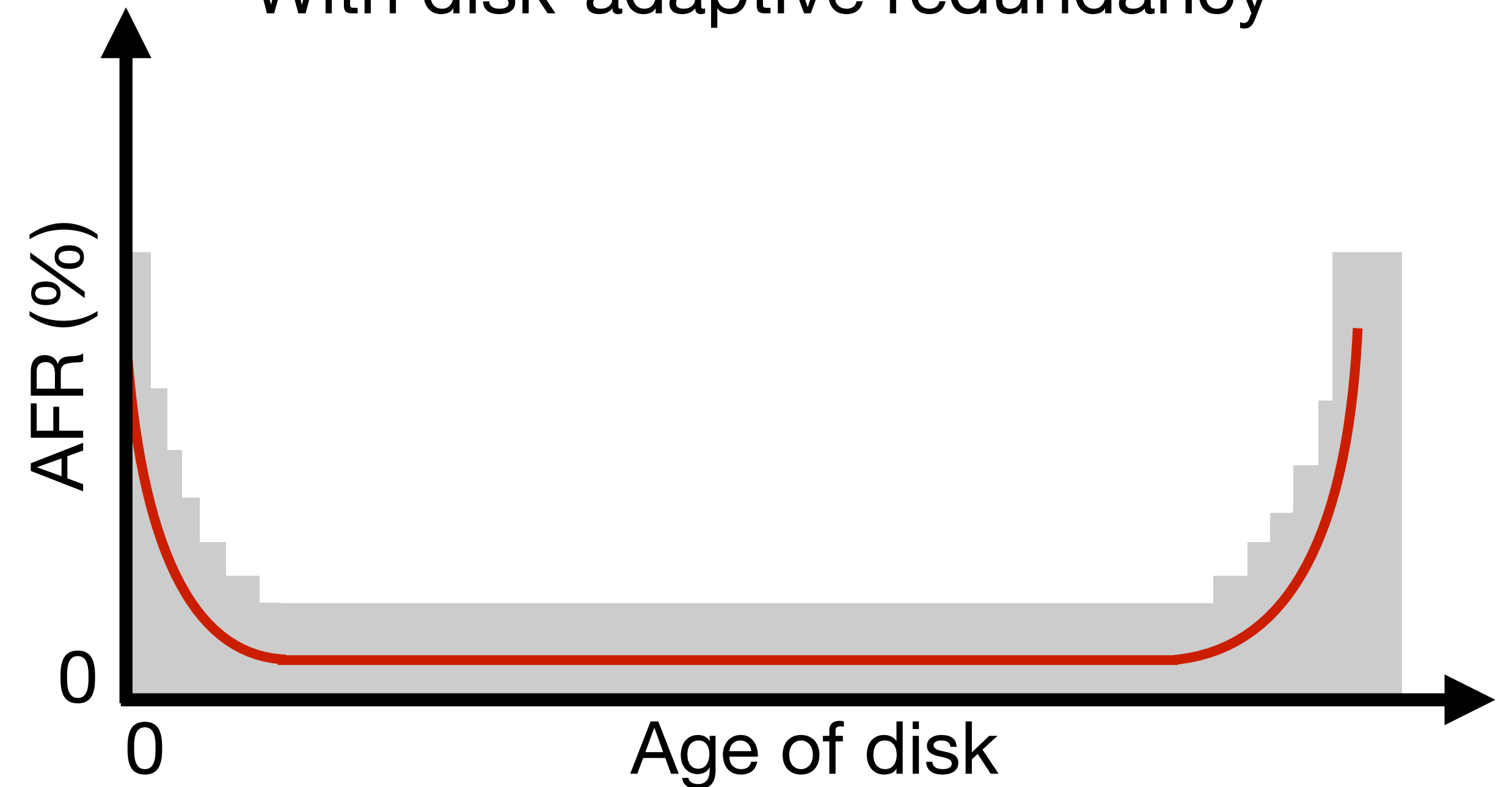
Tailoring data redundancy to disk failure rate



Without disk-adaptive redundancy
(today's storage clusters)

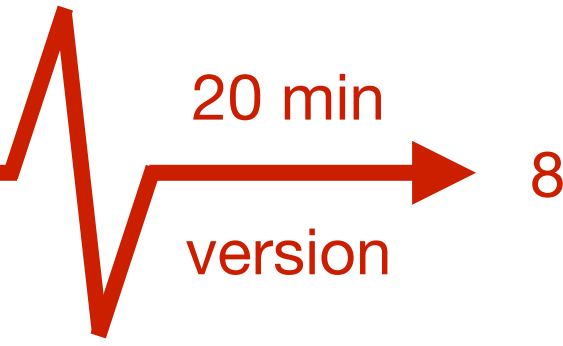


With disk-adaptive redundancy



Lower AFR → Lower redundancy → Lower storage cost

Disk-adaptive redundancy promises huge savings



- First proposed as the **Heterogeneity Aware Redundancy Tuner (HeART)**
 - Published in USENIX FAST 2019
- HeART simulated disk-adaptive redundancy in storage cluster with over 100K HDDs
- Promised substantial storage space-savings over one-scheme-fits-all approaches:
 - Up to **33% lesser space compared to 3-way replication**
 - **11–16% lesser space compared to popular erasure codes: 6-of-9 and 10-of-14**
- In modern storage clusters **>10% space-savings** → tens of thousands of fewer disks

Disk-adaptive redundancy can substantially reduce storage and energy costs

Cluster storage systems gotta have HeART:
improving storage efficiency by exploiting disk-reliability heterogeneity

Saurabh Kadekodi, K. V. Rashmi, Gregory R. Ganger
Carnegie Mellon University

Abstract

Large-scale cluster storage systems typically consist of a heterogeneous mix of storage devices with significantly varying failure rates. Despite such differences among devices, redundancy settings are generally configured in a one-scheme-for-all fashion. In this paper, we make a case for exploiting reliability heterogeneity to tailor redundancy settings to different device groups. We present HeART, an online tuning tool that guides selection of, and transitions between, redundancy settings, based on observed reliability properties of each disk group. By processing disk failure data over time, HeART identifies the boundaries and steady-state failure rate for each deployed disk group (e.g., by makemodel). Using this information, HeART suggests the most space-efficient redundancy option allowed that will achieve the specified target data reliability. Analysis of longitudinal failure data for a large production storage cluster shows the robustness of HeART's failure-rate determination algorithm. The same analysis shows that a storage system guided by HeART could provide target data reliability levels with fewer disks than one-scheme-for-all approaches: 11–16% fewer compared to erasure codes like 10-of-14 or 6-of-9 and 33% fewer compared to 3-way replication.

Figure 1: Simulated failure rate (AFR) for the six disk groups that make up >90% of the 100,000+ HDDs used for the Backblaze backup server [4]. Details of each disk group are given in [2].

make up more than 90% of the cluster storage system (with 100,000+ disks) used for the Backblaze backup server [4]. The highest failure rate is over 3.5x greater than the lowest, and no two are the same. Schneider et al. [3] recently showed that different Flash SSD makes/models similarly exhibit substantial failure rate differences.

Despite such differences, the degree of redundancy employed in cluster storage systems for the purpose of long term data durability (e.g., the degree of replication or erasure code parameters) are generally configured as if all of the devices have the same reliability. Unfortunately, this approach leads to configurations that are overly resource-consuming, overly risky, or a mix of the two. For example, if the redundancy settings are configured to achieve a given data reliability target (e.g., a specific mean time to data loss (MTTDL)) based on the highest AFR of any device makemodel (e.g., S-4 from Fig. 1), then too much space will be used for redundancy associated with data that is stored fully on lower AFR makemodels (e.g., H-4A). Continuing this example, our evaluations show that the overall wasted capacity can be up to 16% compared to uniform use of erasure code settings stated as being used in real large storage clusters [12, 24, 27] and up to 33% compared to 3-way replication. This is the direct consequence of not making more disks are needed. If redundancy failure rates (AFR) are based on lower AFRs, on the other hand, higher AFR devices is not

Introduction

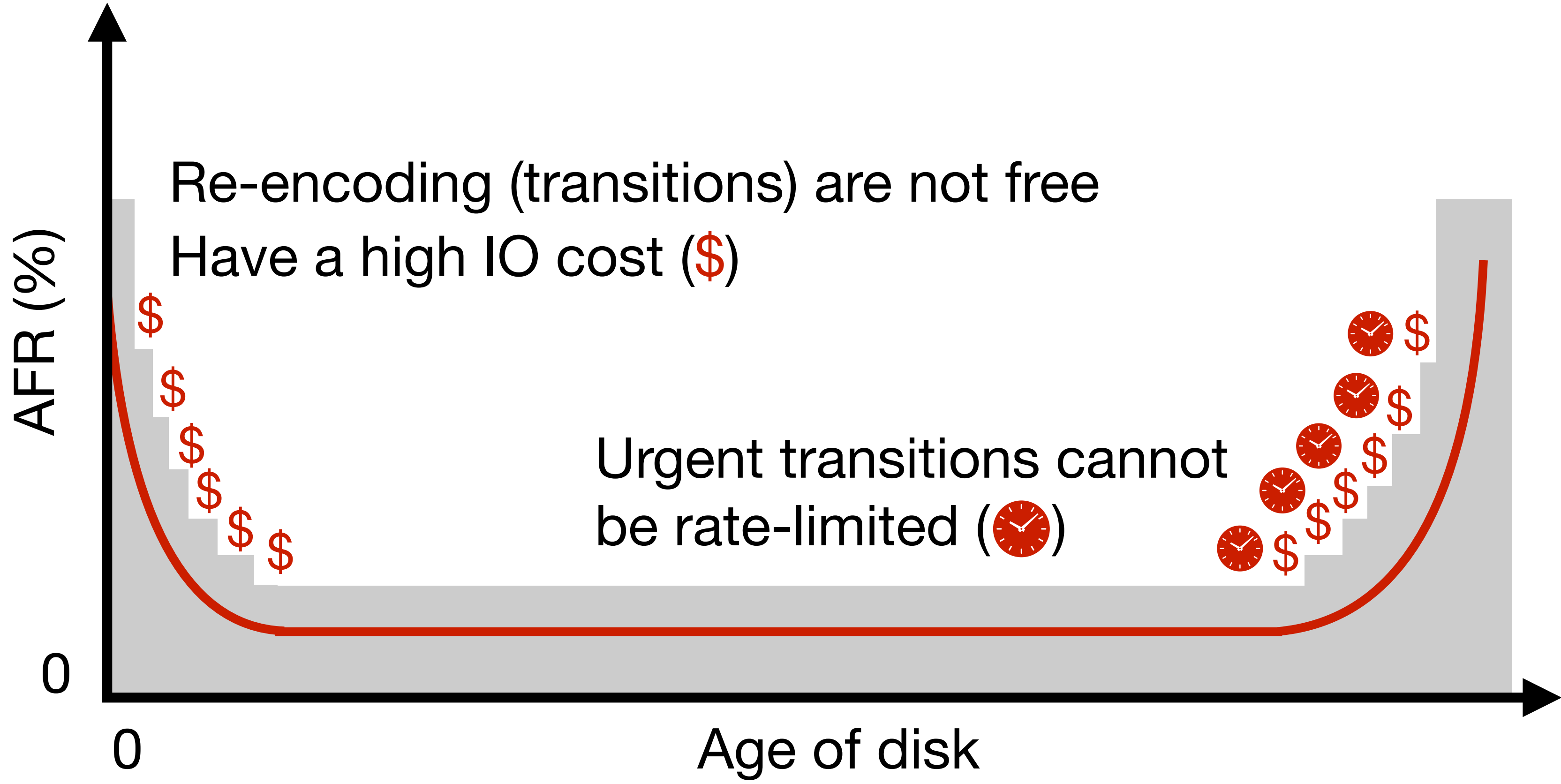
Cluster storage systems almost always include a heterogeneous mix of storage devices, even when using devices like Flash SSDs or mechanical hard drives. Reliability heterogeneity arises from increasing device diversity and from manufacturing variations. Exploiting this heterogeneity for optimization opportunities is a research goal. In this paper, we present HeART, a redundancy tuner that guides selection of, and transitions between, redundancy settings, based on observed reliability properties of each disk group. By processing disk failure data over time, HeART identifies the boundaries and steady-state failure rate for each deployed disk group (e.g., by makemodel). Using this information, HeART suggests the most space-efficient redundancy option allowed that will achieve the specified target data reliability. Analysis of longitudinal failure data for a large production storage cluster shows the robustness of HeART's failure-rate determination algorithm. The same analysis shows that a storage system guided by HeART could provide target data reliability levels with fewer disks than one-scheme-for-all approaches: 11–16% fewer compared to erasure codes like 10-of-14 or 6-of-9 and 33% fewer compared to 3-way replication.

By exploiting reliability heterogeneity to tailor redundancy settings to different device groups, we make a case for exploiting reliability heterogeneity to tailor redundancy settings to different device groups. We present HeART, an online tuning tool that guides selection of, and transitions between, redundancy settings, based on observed reliability properties of each disk group. By processing disk failure data over time, HeART identifies the boundaries and steady-state failure rate for each deployed disk group (e.g., by makemodel). Using this information, HeART suggests the most space-efficient redundancy option allowed that will achieve the specified target data reliability. Analysis of longitudinal failure data for a large production storage cluster shows the robustness of HeART's failure-rate determination algorithm. The same analysis shows that a storage system guided by HeART could provide target data reliability levels with fewer disks than one-scheme-for-all approaches: 11–16% fewer compared to erasure codes like 10-of-14 or 6-of-9 and 33% fewer compared to 3-way replication.

FAST 2019

Existing solutions suffer from transition overload

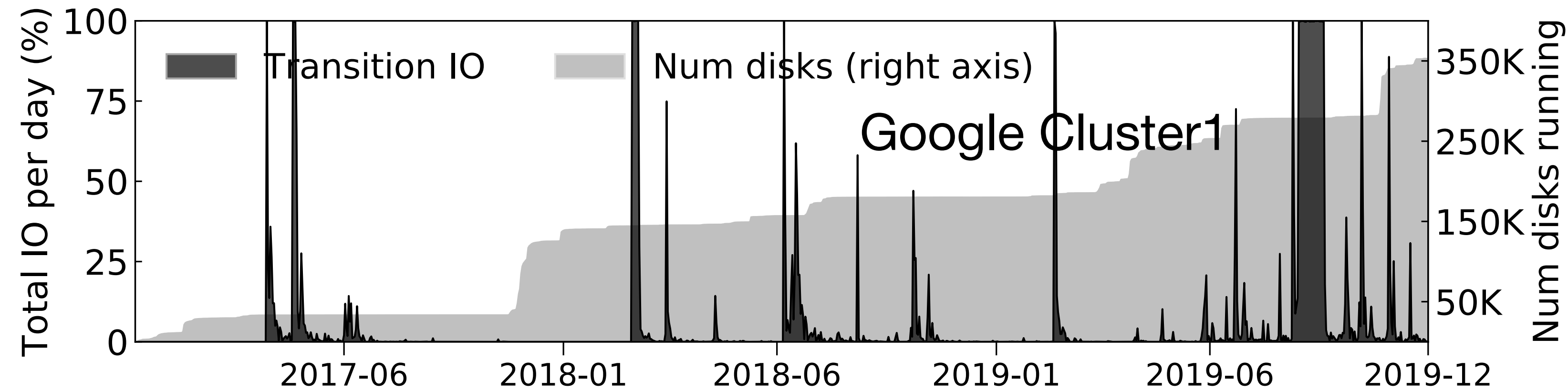
20 min
version → 9



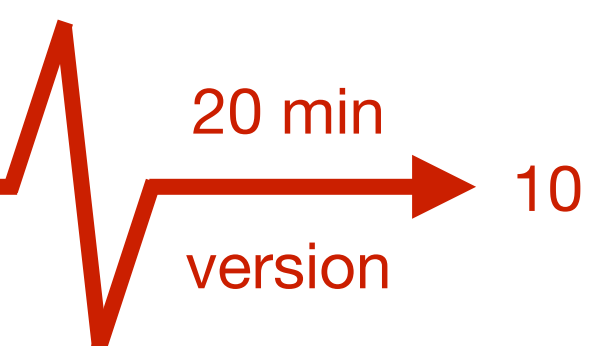
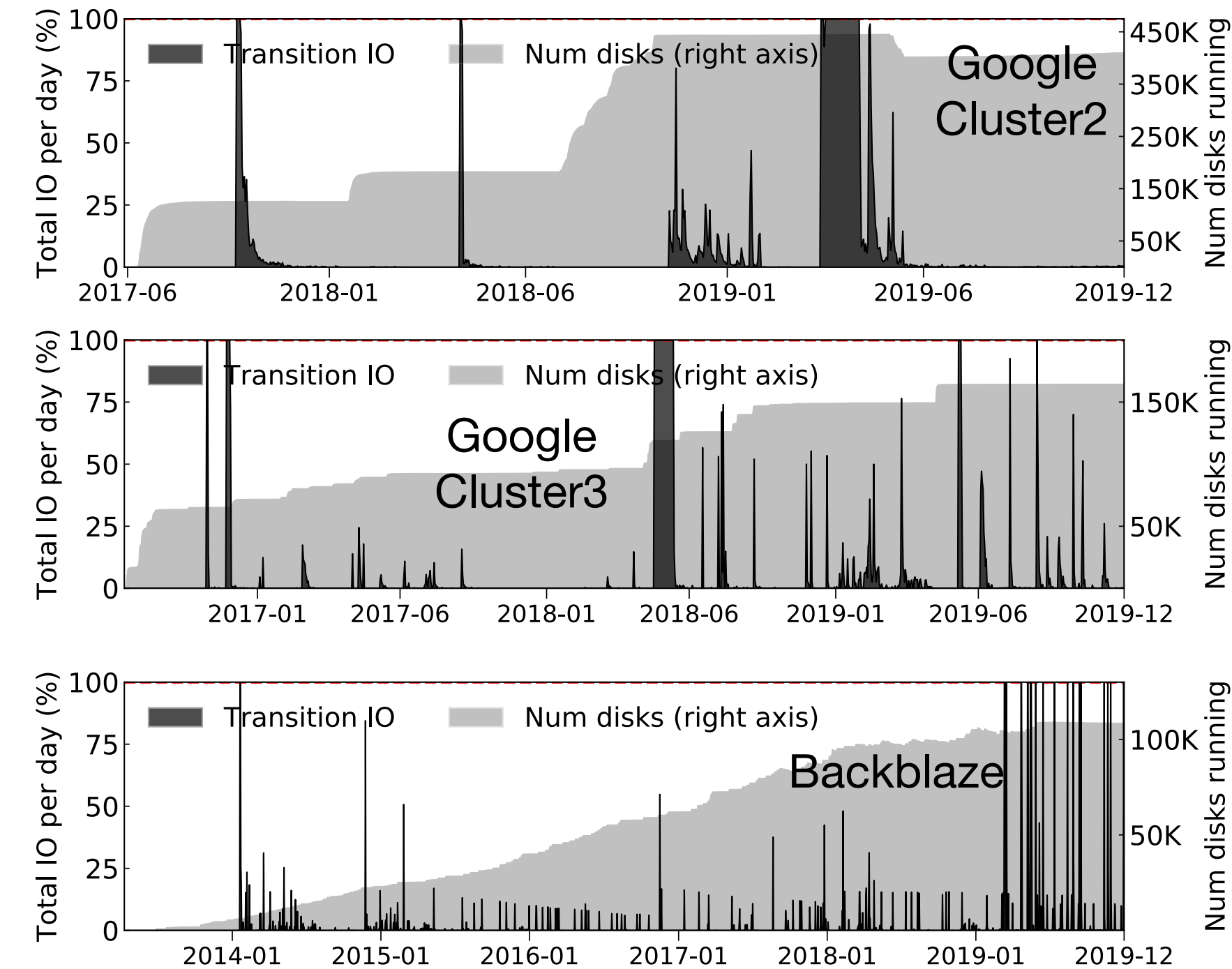
High IO cost and urgent transitions cause **transition overload**

Transition overload is a show-stopper

- Existing disk-adaptive redundancy simulated on clusters

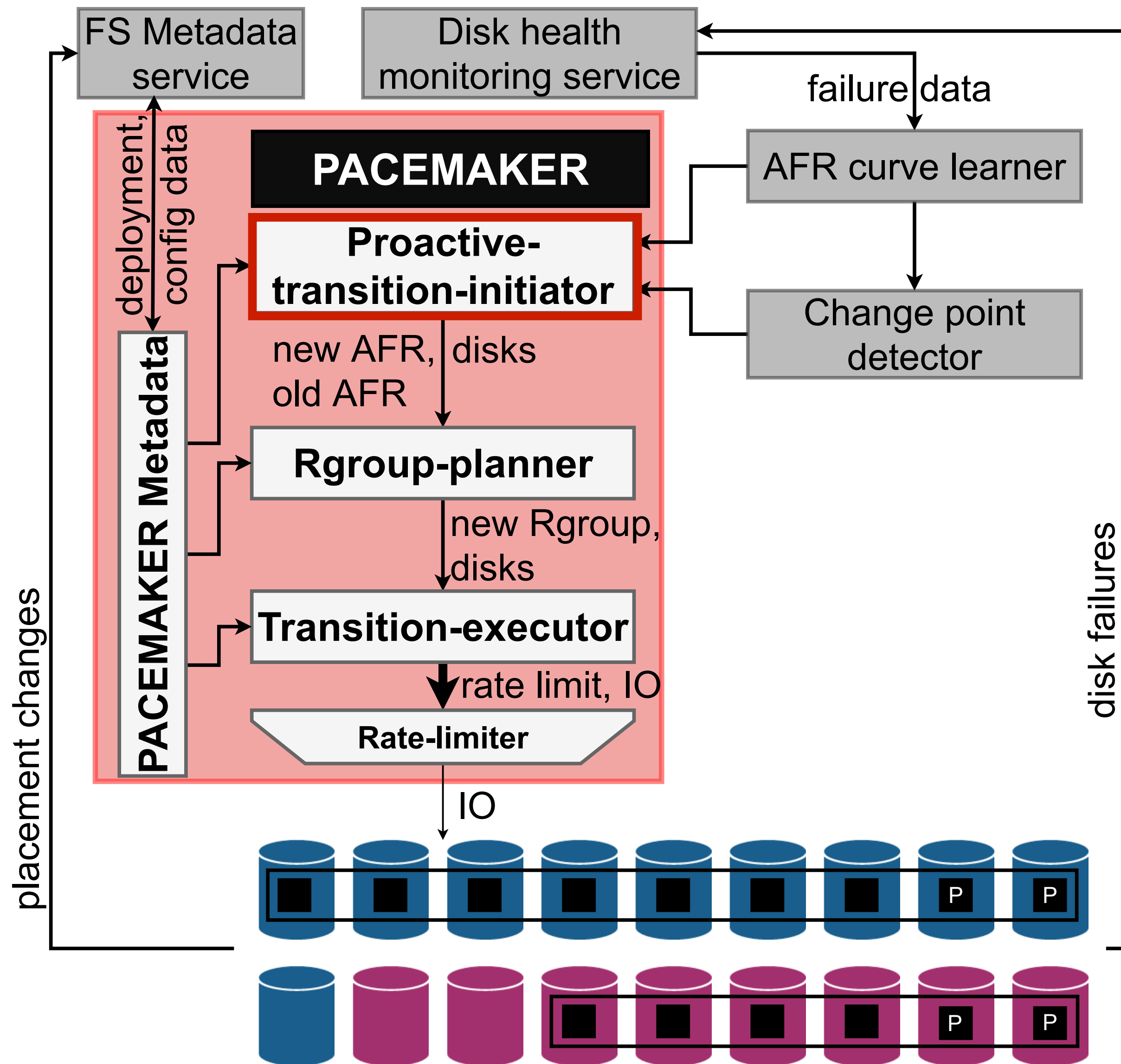


- Weeks of 100% cluster IO bandwidth spent in transitions
 - caused by costly transitions
 - in addition to too many disks transitioning together



Pacemaker built to overcome transition overload

20 min
version → 11



3 questions guide Pacemaker's approach:

1. When should disks transition?

 Issue proactive transitions that can be safely rate-limited

2. Which scheme should disks transition to?

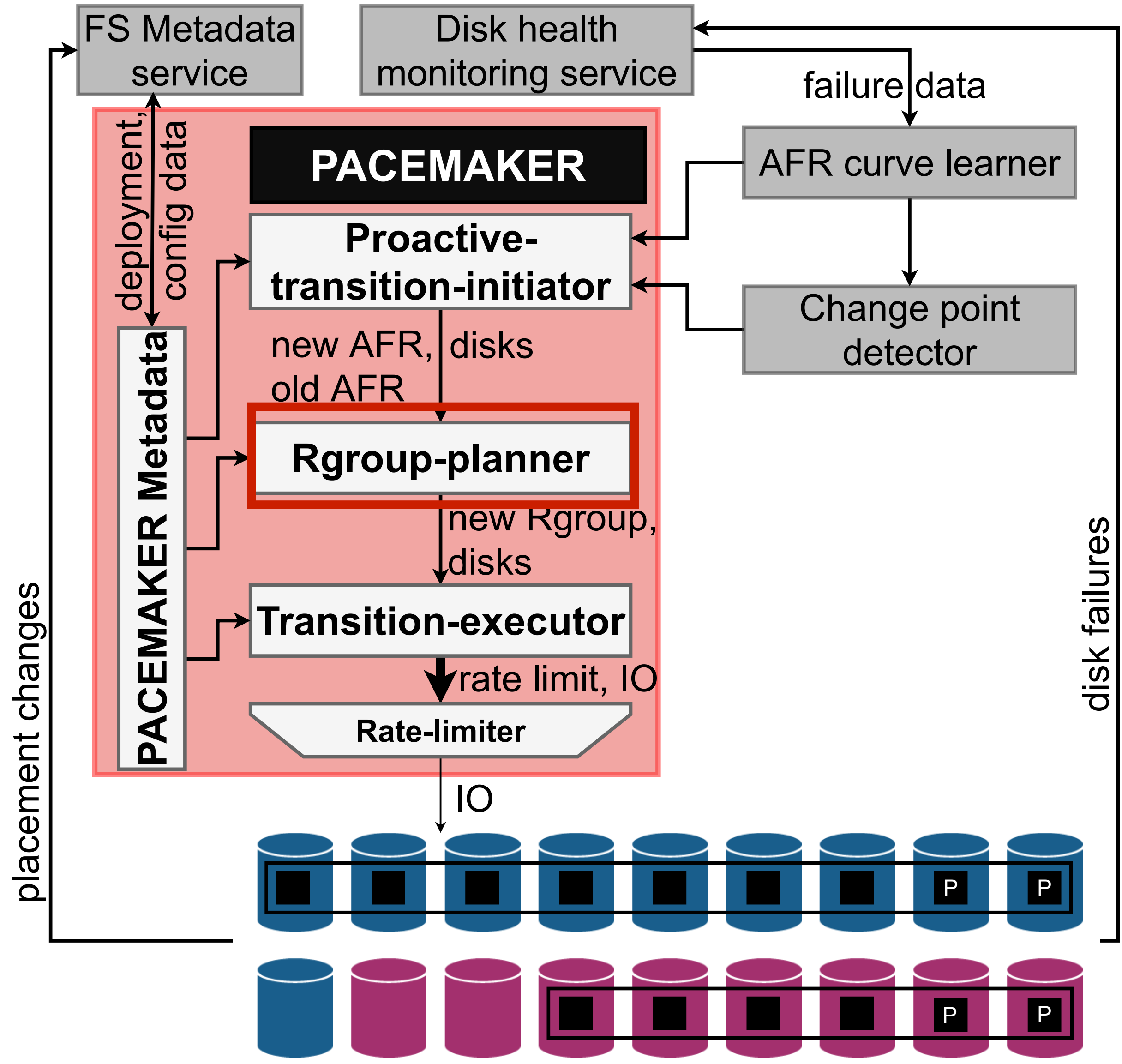
3. How should the disks transition?

Pacemaker built to overcome transition overload

20 min

version

11

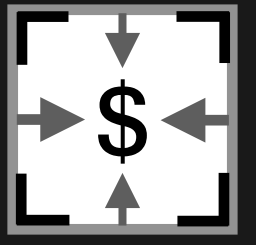


3 questions guide Pacemaker's approach:

1. When should disks transition?

 Issue proactive transitions that can be safely rate-limited

2. Which scheme should disks transition to?

 Most space-efficient scheme with constraints on IO

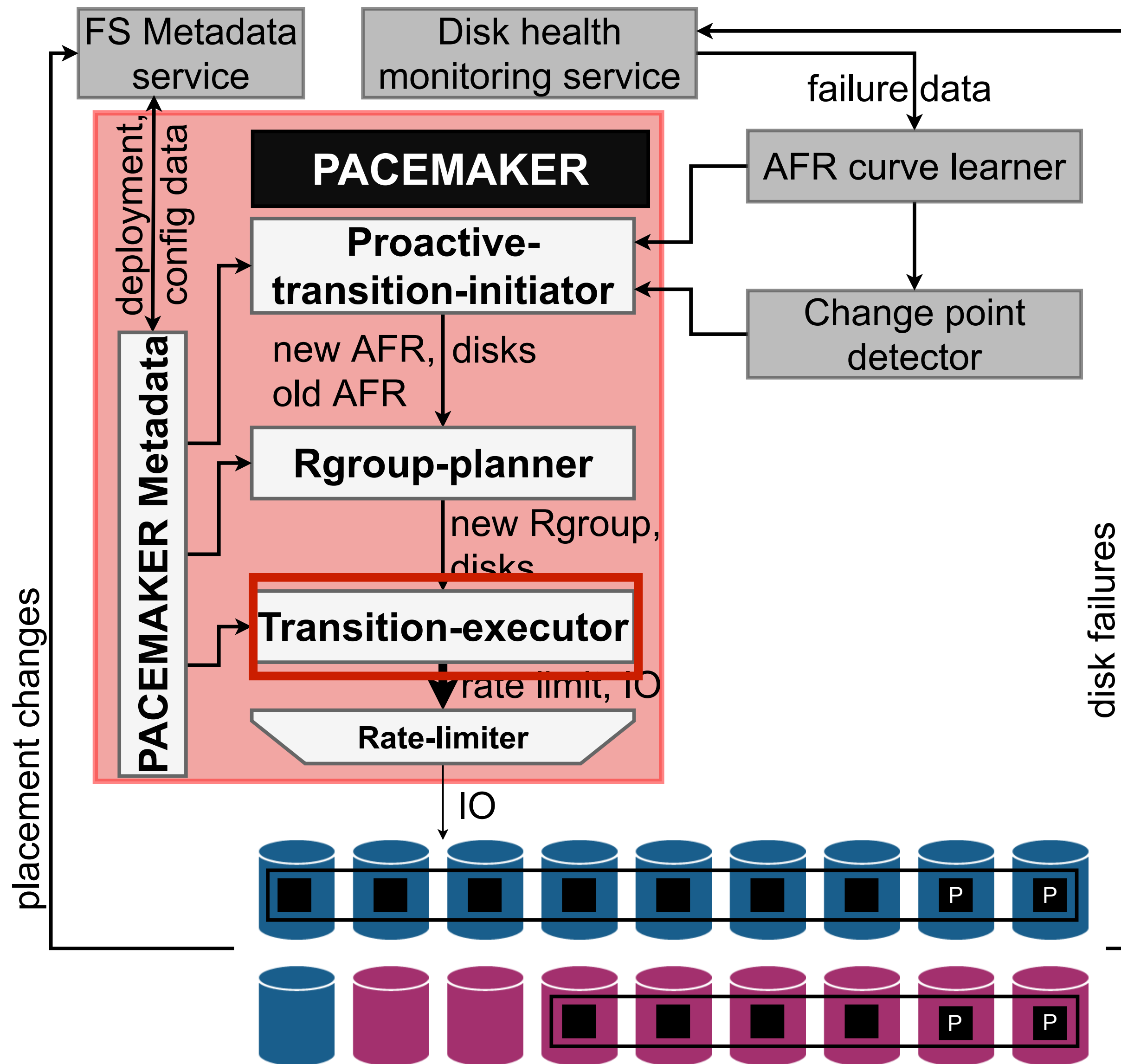
3. How should the disks transition?

Pacemaker built to overcome transition overload

20 min

version

11



3 questions guide Pacemaker's approach:

1. When should disks transition?

 Issue proactive transitions that can be safely rate-limited

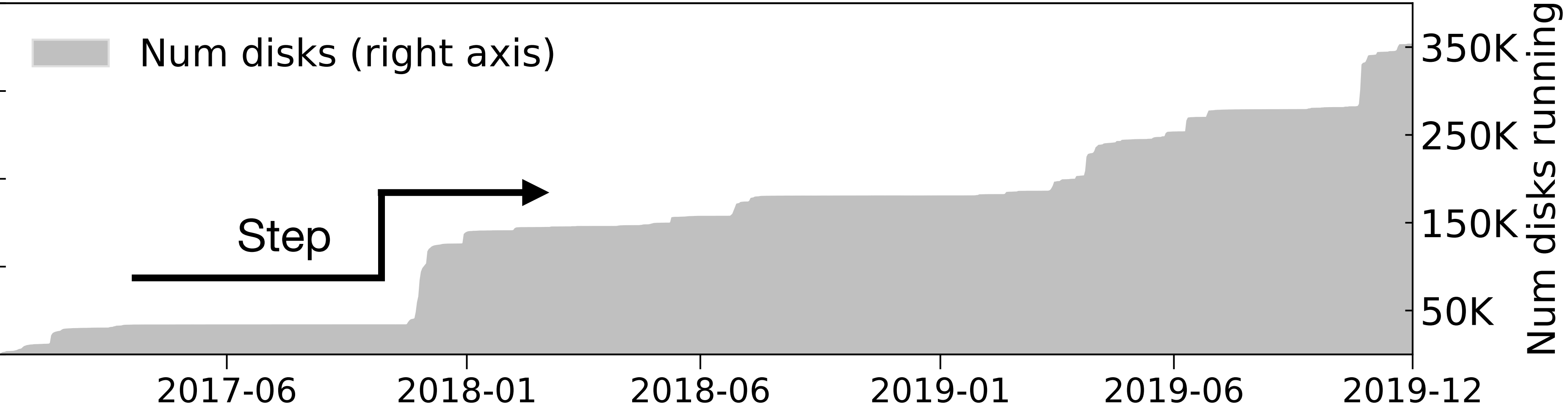
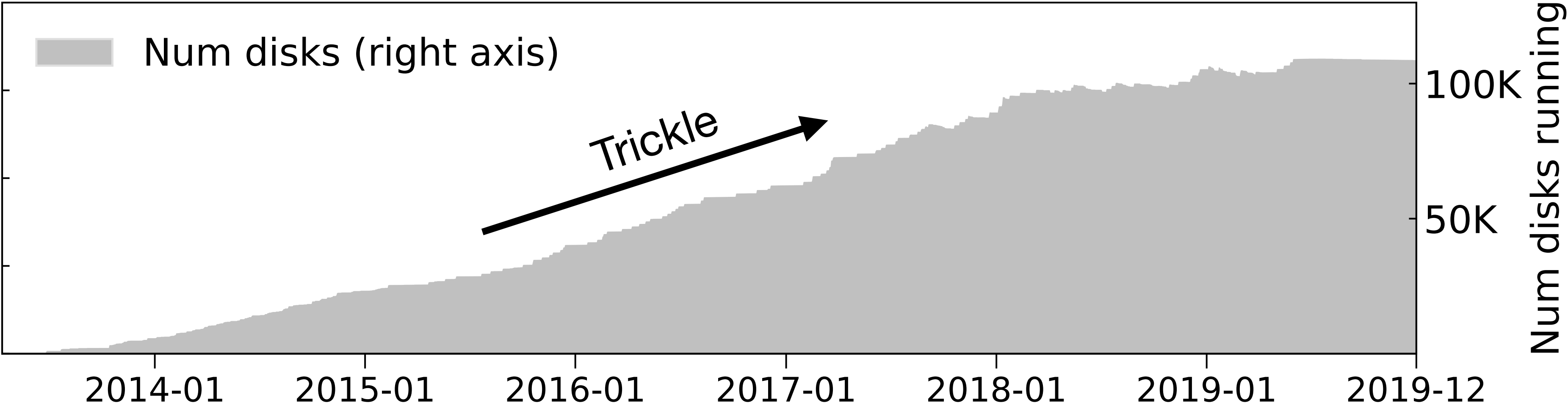
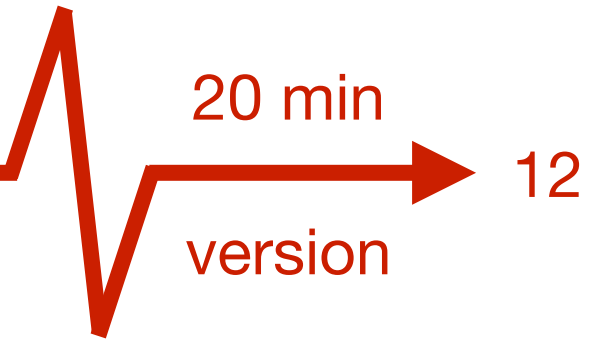
2. Which scheme should disks transition to?

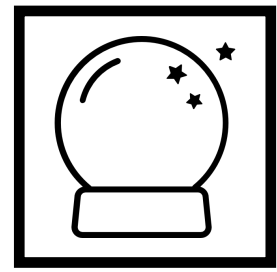
 Most space-efficient scheme with constraints on IO

3. How should the disks transition?

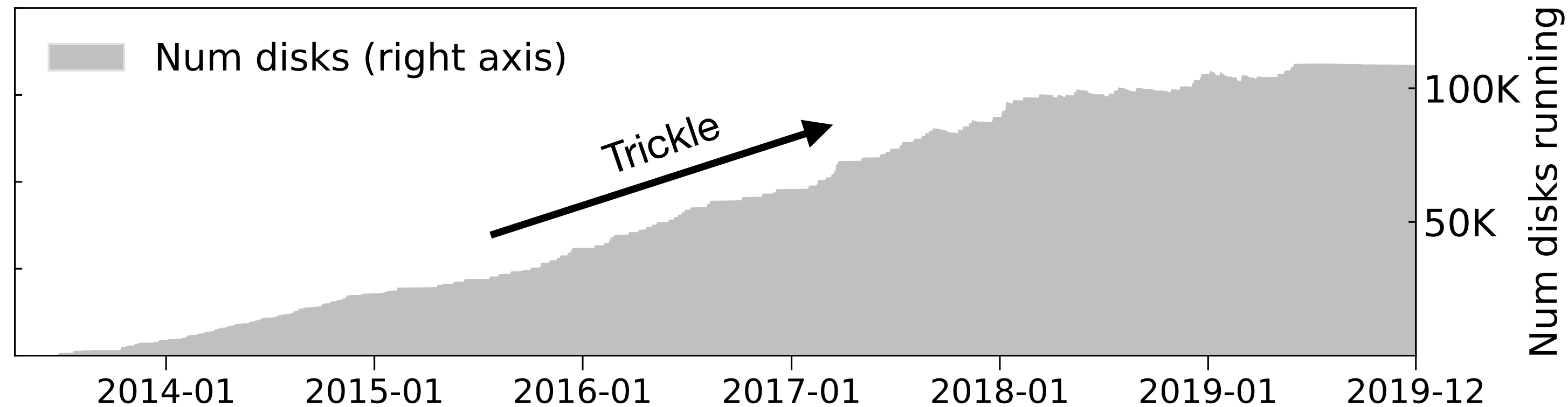
 New IO-efficient transitioning mechanisms

Disk deployment patterns: **trickle** and **step**





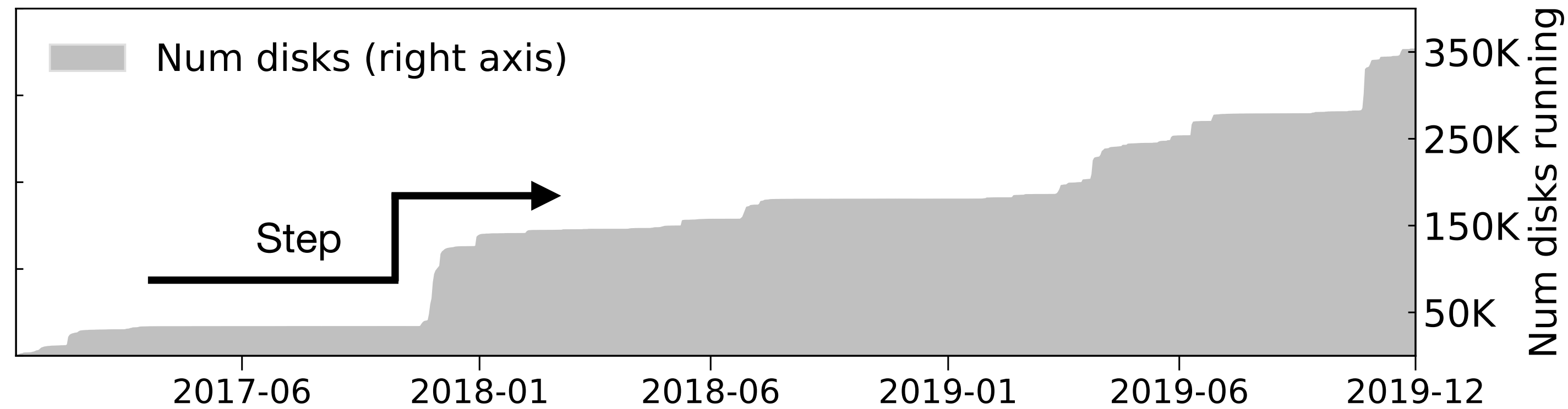
Proactive transitions for trickle deployments



- Trickle-deployed disks are deployed in tens and hundreds every few days
- Thousands of disks are needed for statistically accurate AFR estimation
 - AFR rise for a given age can't be known until few thousand disks cross that age
- Pacemaker marks **first C disks are canary disks** ($C = 3000$)
 - Pacemaker learns the AFR curve from canaries
 - Redundancy not optimized for canary disks
- Remaining trickle-deployed disks can be proactively transitioned
 - AFR curve learned from canaries educates Pacemaker of age when AFR rises



Proactive transitions for step deployments

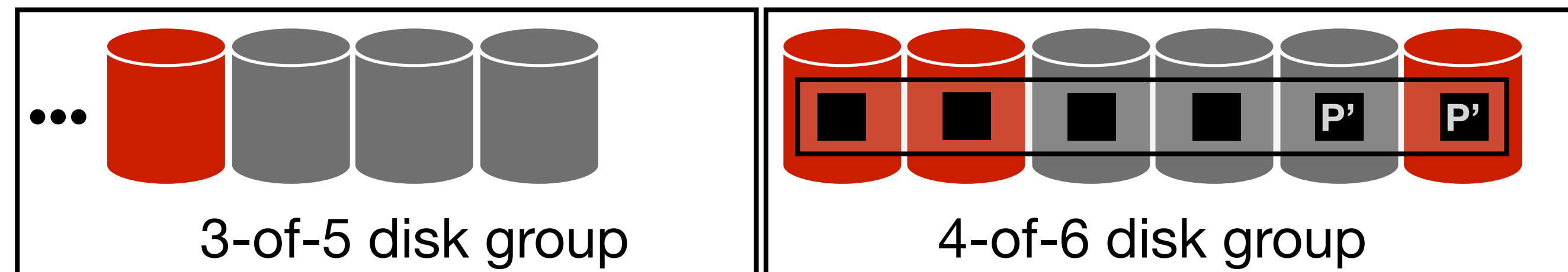


- Step-deployed disks deployed in several thousands over a few days
 - Canaries useless for step-deployed disks; almost all disks deployed together
 - Not optimizing for canaries implies not optimizing for most disks
- Step-deployed disks always provide high-confidence AFR estimate
 - Not true for trickle-deployed disks deployed a-few-at-a-time over long periods
- Pacemaker uses slowly rising AFR as an early-warning system
 - AFRs rise gradually towards wearout (refer paper)
- Early-warning triggers proactive transitions

Traditional re-encoding (transitioning) is costly

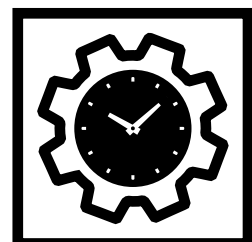
20 min
version

16



Disk transition IO $> 2 \times k_1 \times \text{disk-capacity}$

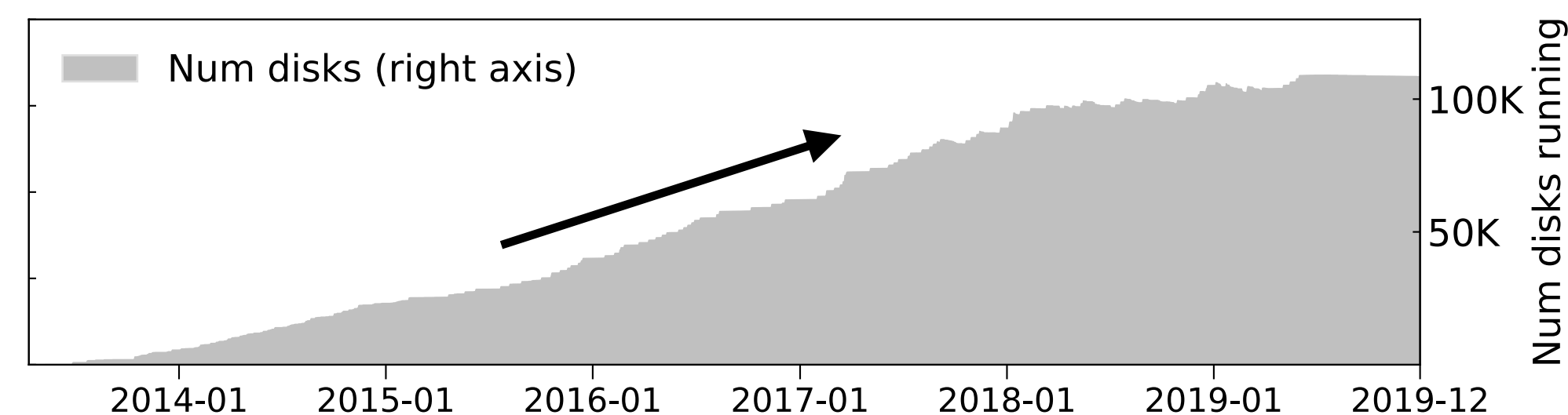
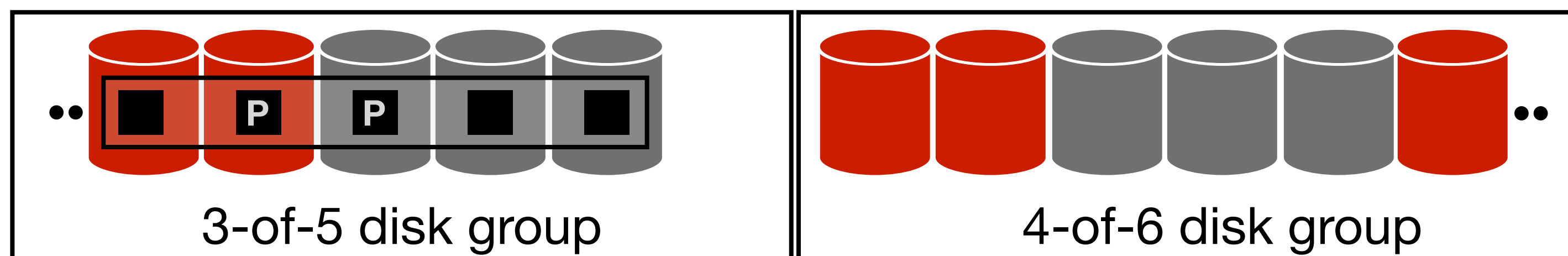
- Need to re-encode (transition) k_1 -of- n_1 to k_2 -of- n_2
- Read rest of the data chunks of stripe ($k_1 \times \text{disk-capacity}$)
- Write new stripe to new disk-group ($k_1 \times \text{disk-capacity}$)
- Create new parities
- Delete old parities



Transitioning by emptying disks

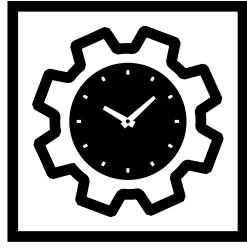
20 min
version

17



Disk transition IO = $2 \times$ disk-capacity
cheaper than traditional by $\approx k_1 \times$

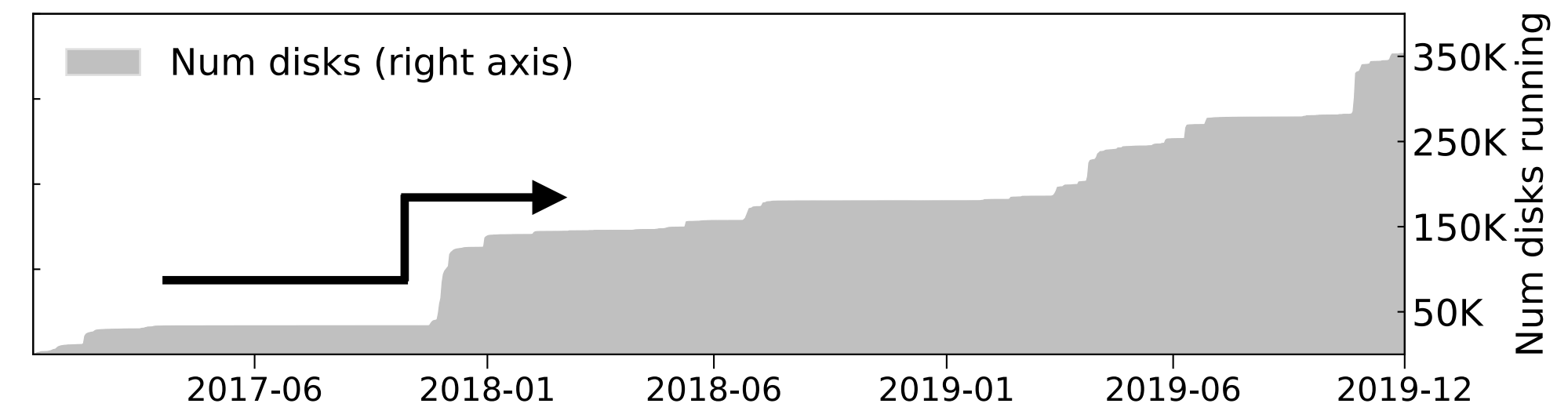
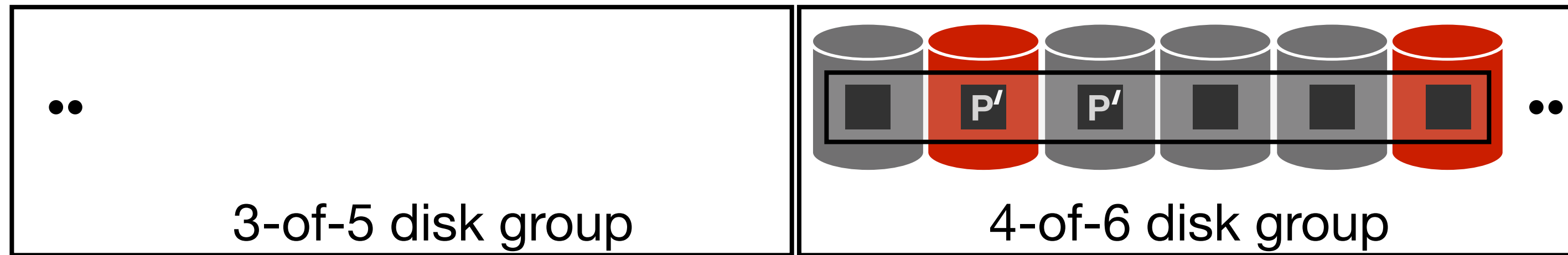
- Move data in same disk-group **before transition** ($2 \times$ disk-capacity)
- Transition an empty disk
- No data on disk \rightarrow no expensive re-encoding
- Apt when a few disks are transitioning at a time (trickle deployments)



Transitioning by bulk parity re-calculation

20 min
version

18



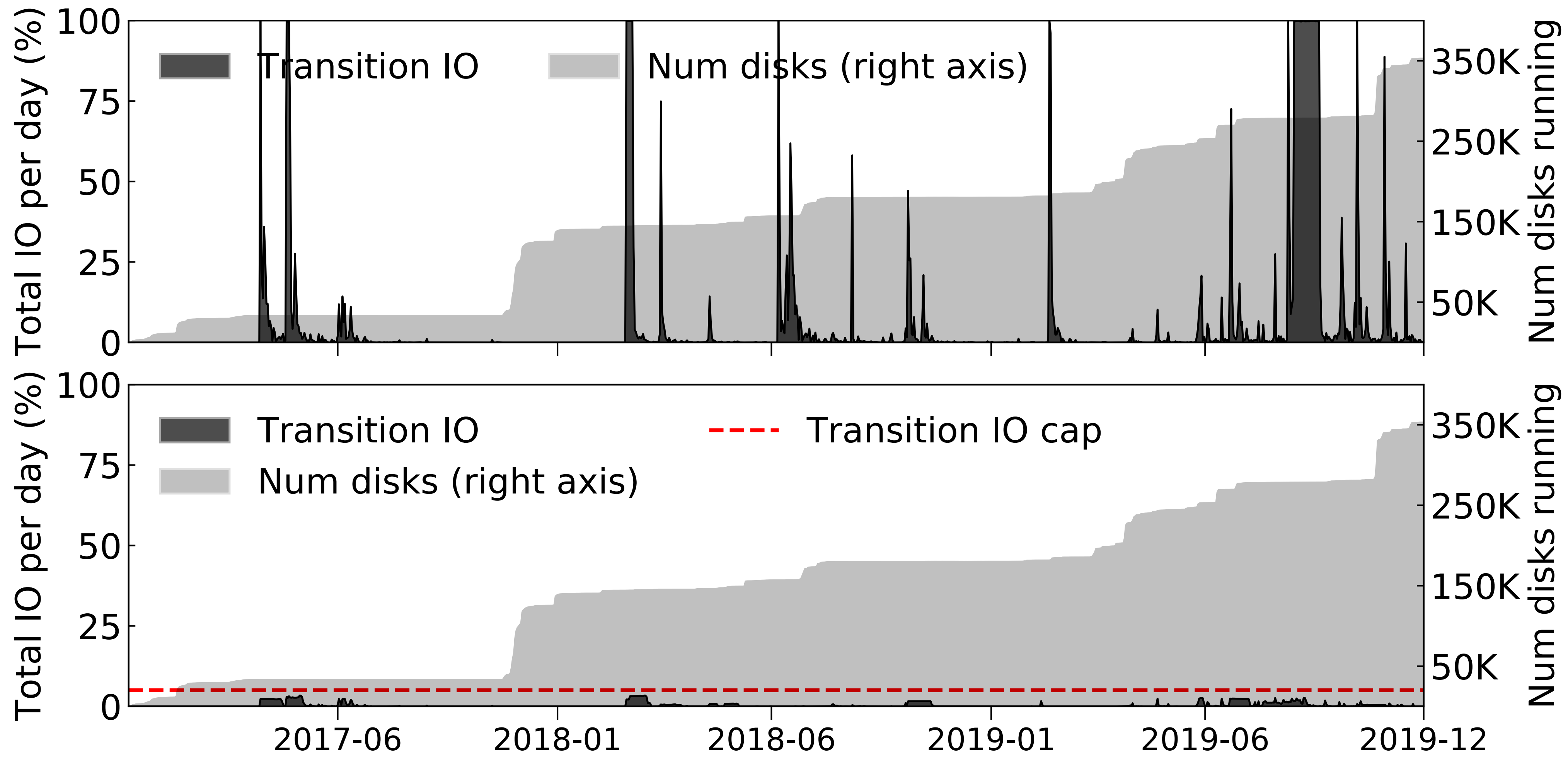
$$\text{Disk transition IO} = \left(1 + \frac{n_2 - k_2}{k_2}\right) \times \frac{k_1}{n_1} \times \text{disk-capacity}$$

cheaper by $\approx n_1 \times$

- Disks transition **without moving data**
- New parities calculated based on new scheme ($1 \times$ disk-capacity)
- New stripes formed (write only new parities)
- Apt for re-encoding large disk populations together (step deployments)

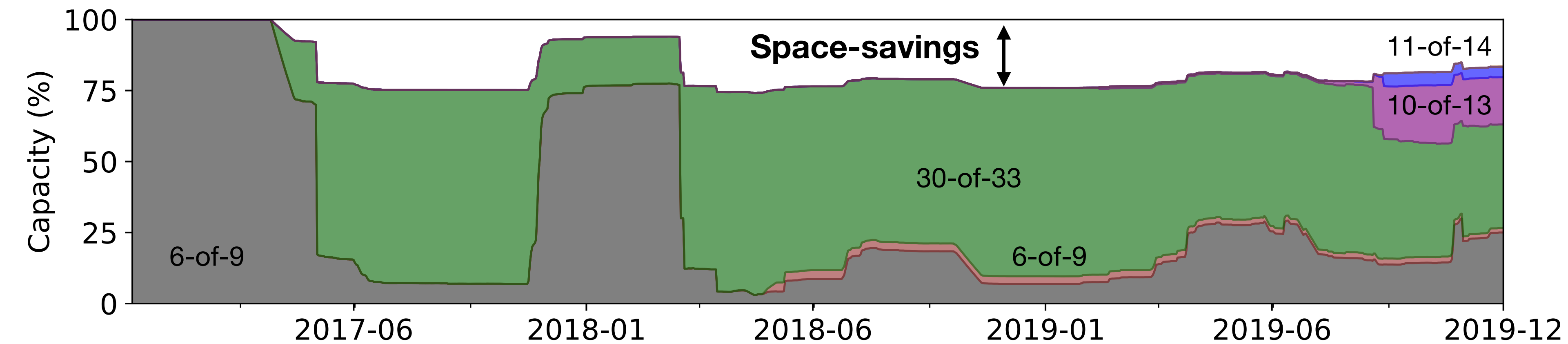
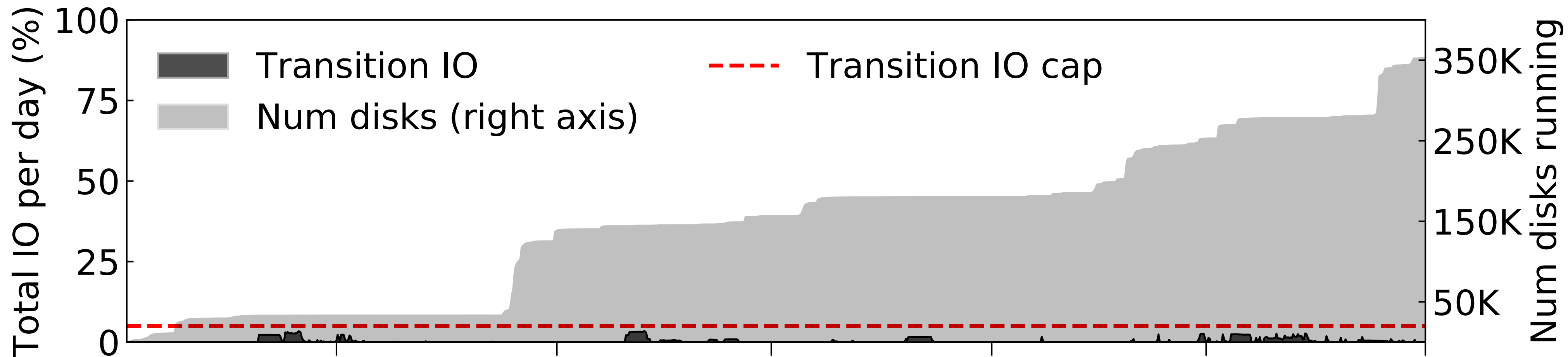
Evaluation: Google Cluster1 (transition overload)

20 min
version → 19



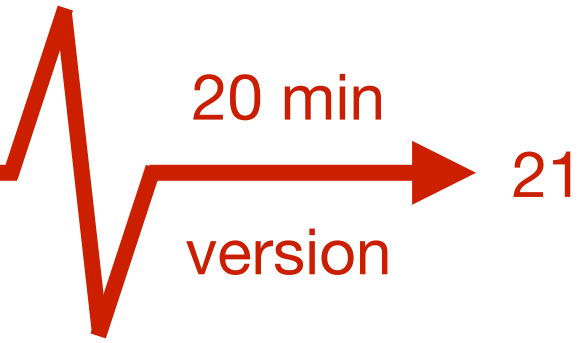
Avg. IO = 0.3%
Peak IO < 5%

Evaluation: Google Cluster1 (space-savings)



Avg. space-savings = 14%
Peak space-savings = 25%

Conclusion



- Disk-adaptive redundancy systems suffer from transition overload
- Pacemaker is an IO-efficient disk-adaptive redundancy orchestrator
 - Only uses 0.2–0.4% cluster IO bandwidth on average for redundancy transitions
 - All transition IO activity is safely capped to <5% cluster IO bandwidth
 - Provides between 14–20% average space-savings; tens of thousands of fewer disks
- Pacemaker’s design is informed by real-world disk failure analysis
 - 5.3 million disks spanning over 60 makes/models from Google, NetApp, Backblaze
 - Pacemaker’s design is based on insights from this analysis (refer paper for details)
- Working prototype of Pacemaker-equipped HDFS (refer paper)
 - Prototype reuses existing HDFS machinery to enable disk-adaptive redundancy
 - Open sourced at: <https://github.com/thesys-lab/pacemaker-hdfs>
- Contact saukad@cs.cmu.edu for questions or feedback