# DARE  Disk-Adaptive REdundancy

Tailoring data redundancy to disk-reliability heterogeneity in cluster storage systems

Saurabh Kadekodi

Parallel Data Laboratory, Carnegie Mellon University

Carnegie Mellon University Computer Science Department

# Outline

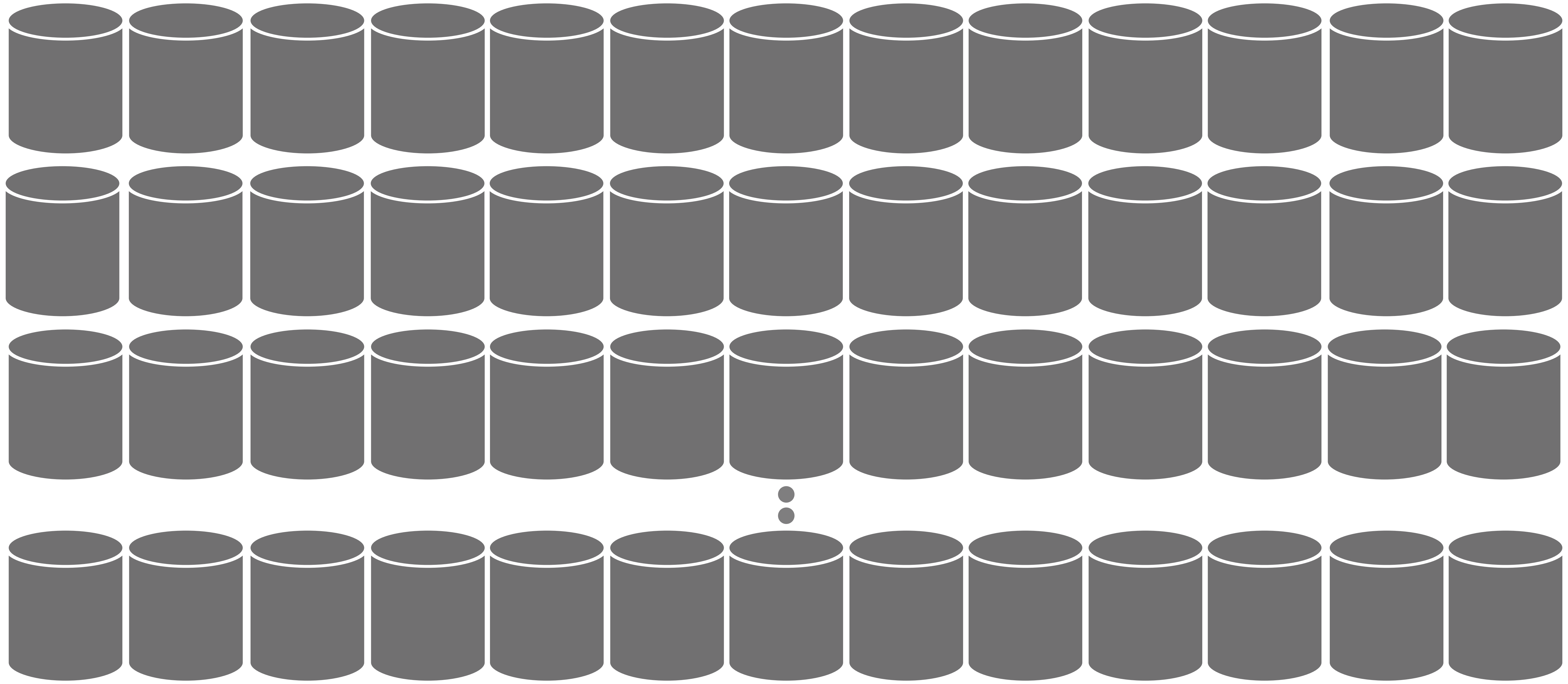
- Background and motivation
- Disk failure rate heterogeneity
- Making a case for disk-adaptive redundancy (DARE)
- Overcoming transition overload
- Evaluation on real-world cluster traces
- Enabling HDFS to DARE

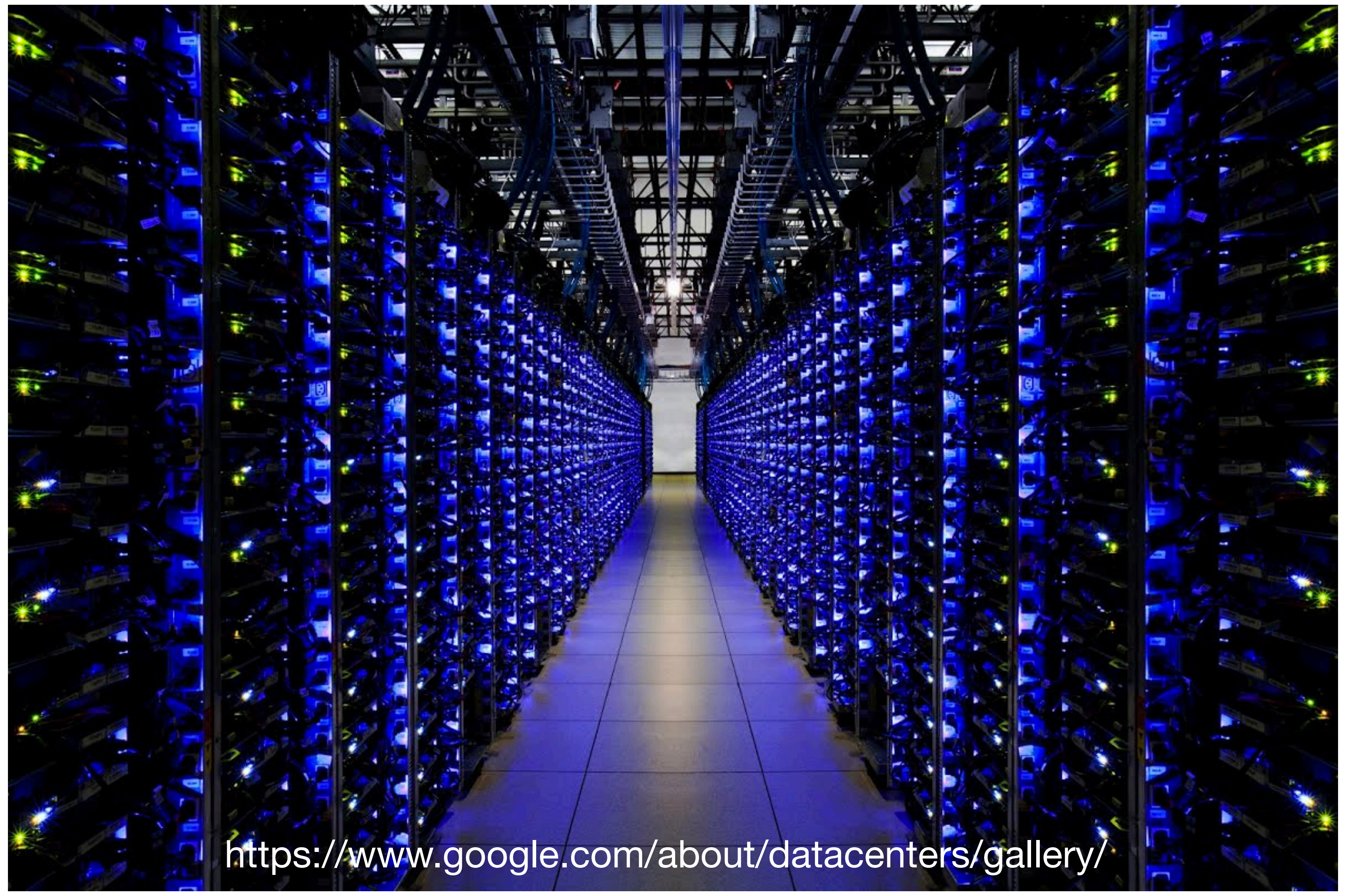
DARE

# Outline

- Background and motivation
- Disk failure rate heterogeneity
- Making a case for disk-adaptive redundancy (DARE)
- Overcoming transition overload
- Evaluation on real-world cluster traces
- Enabling HDFS to DARE

# Today's storage clusters

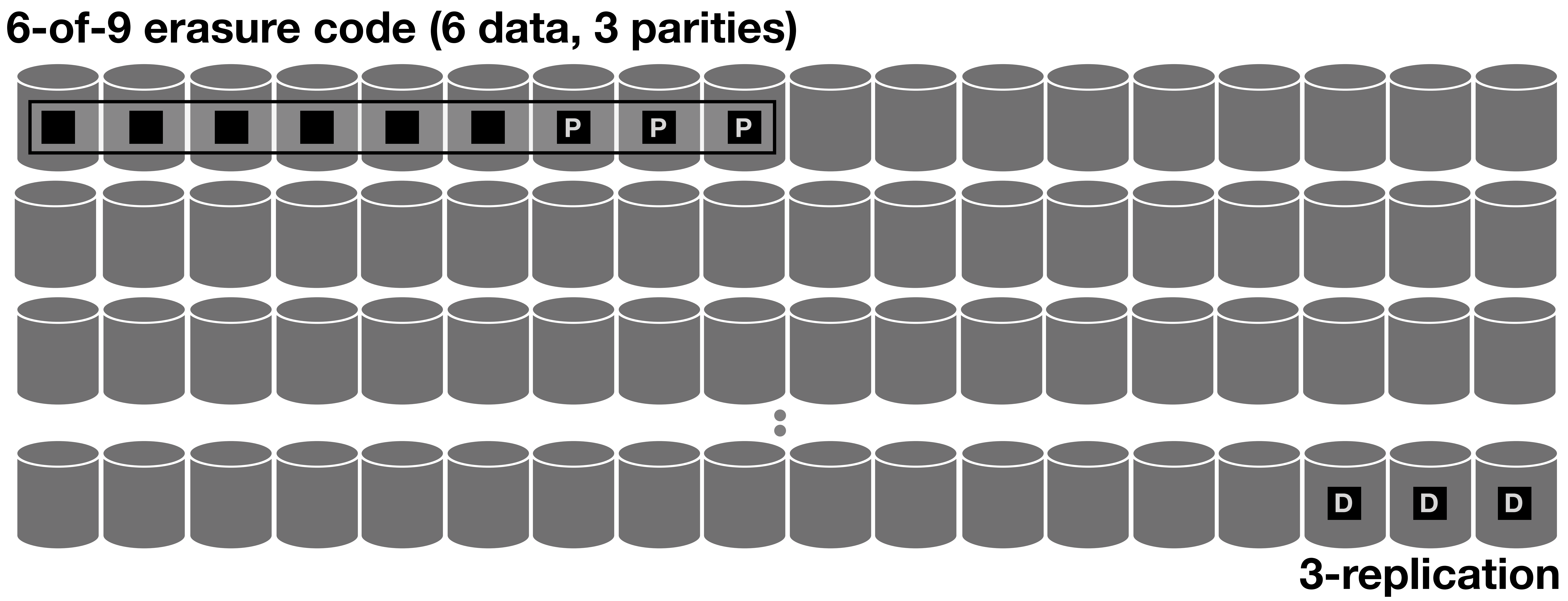





- Storage subsystem of distributed systems
- 1000s to millions of hard-disk drives (HDD) in primary storage tier
- Failures common in today's cluster storage systems
  - Disk failures measured as **annualized failure rates (AFR)**

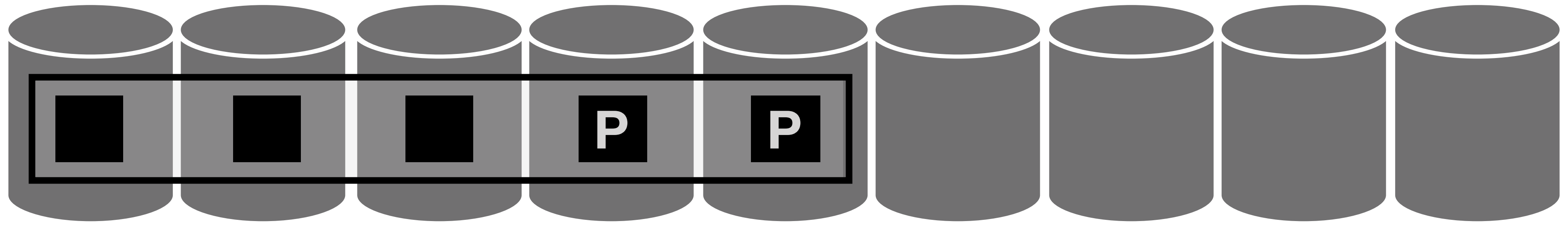
AFR = expected % of disk failures in a given year

# Data redundancy prevents data loss

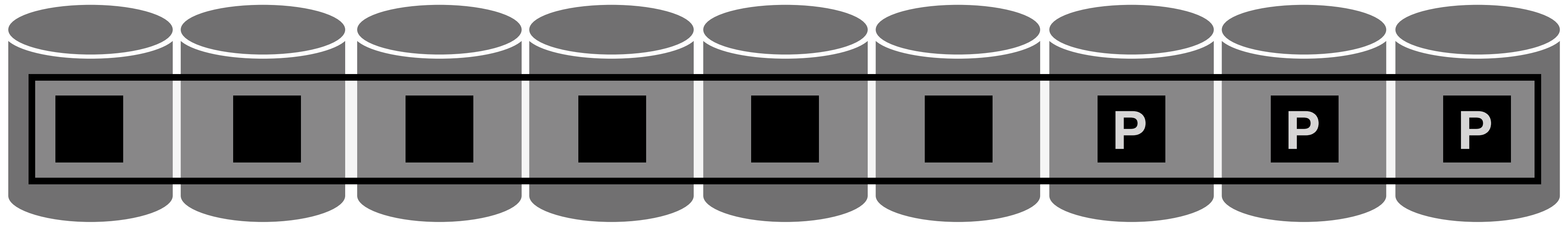


# Erasure coding primer

- Erasure coding is space-efficient redundancy
- $k$—of—$n$ scheme: $k$ data chunks (■ ■ ■), $n - k$ parity chunks (P P)
  - $n$ chunks form a stripe: ■ ■ ■ P P
  - All chunks are of the same size (typically few MBs per chunk)
  - Failed chunk reconstructed using any $k$—of—$n$ chunks
- Storage overhead: $\frac{n}{k}$
  - Reliability (typically) directly proportional to overhead




3-of-5 erasure code (overhead = 1.66)

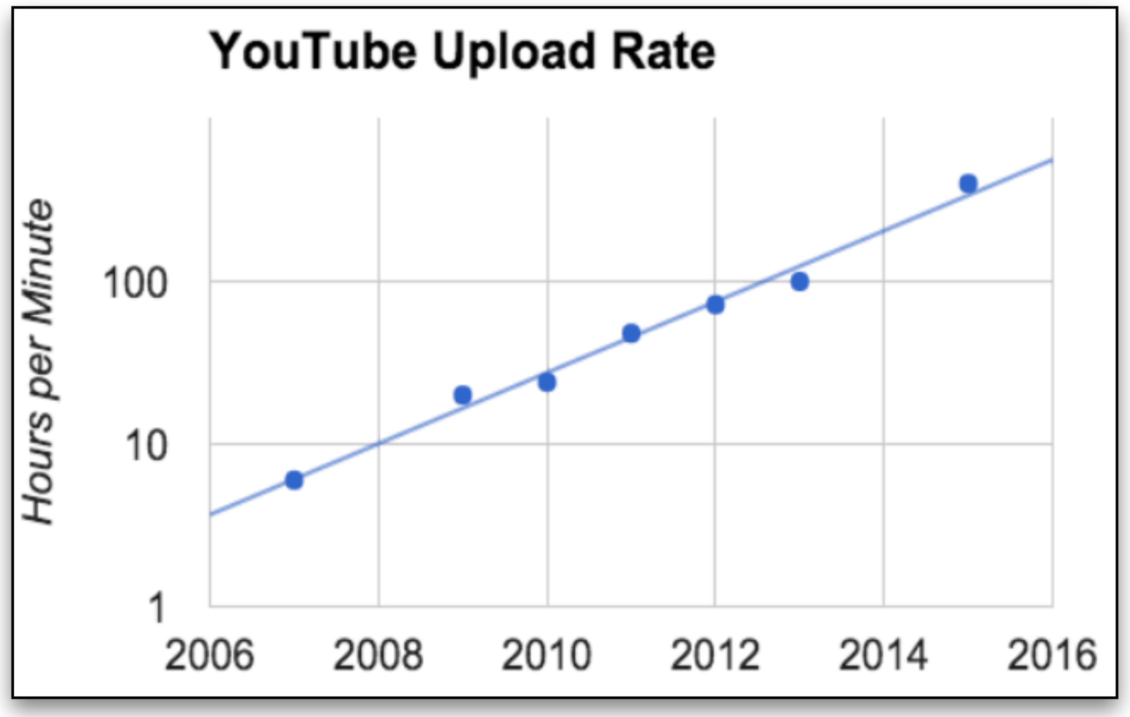



6-of-9 erasure code (overhead = 1.5)

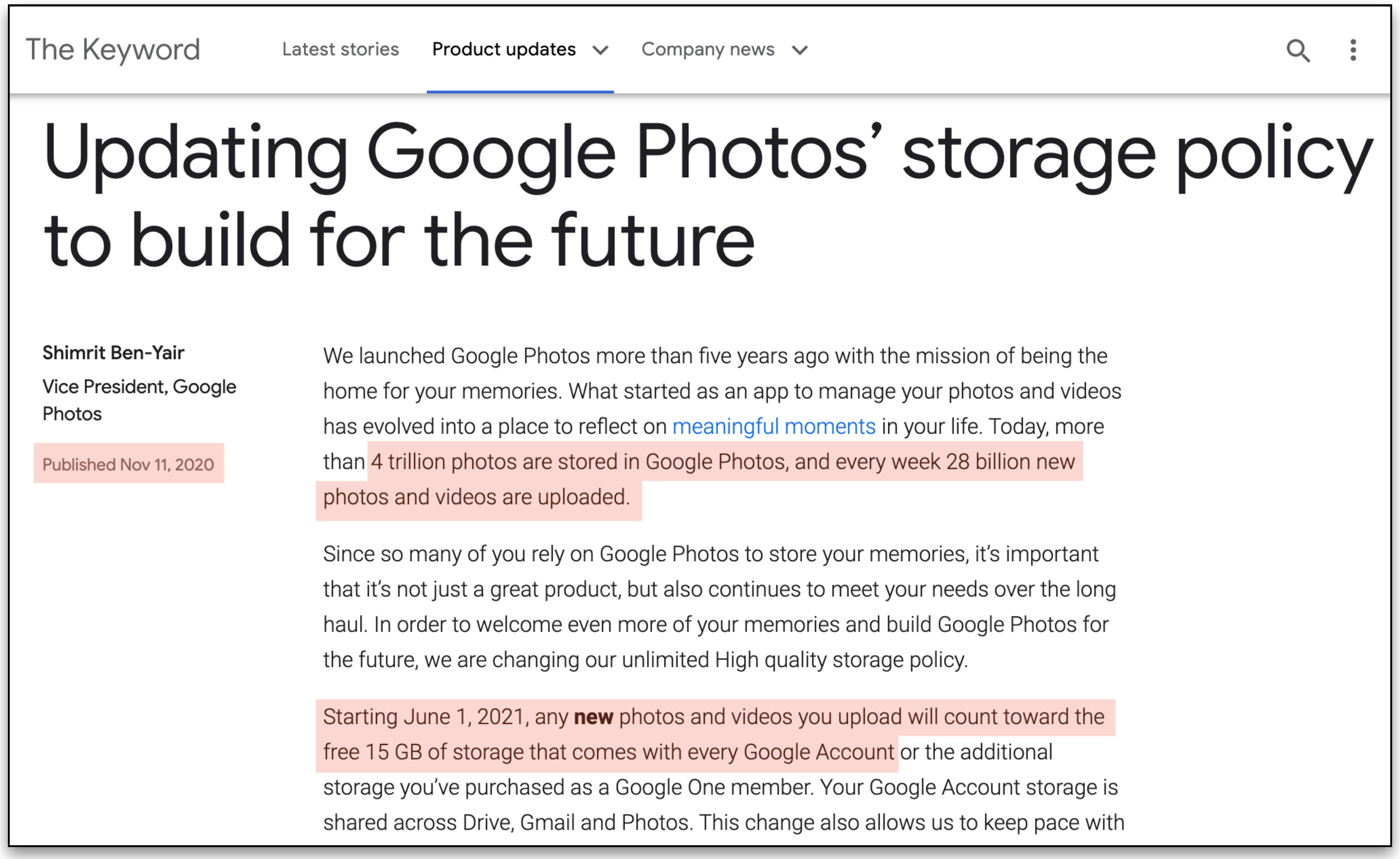
**Bulk of the data in large-scale storage clusters is erasure encoded**

DARE

# Data grows exponentially




Disks for Data Centers White paper for FAST 2016



The Keyword | Latest stories | Product updates | Company news

## Updating Google Photos’ storage policy to build for the future

Shimrit Ben-Yair
Vice President, Google Photos

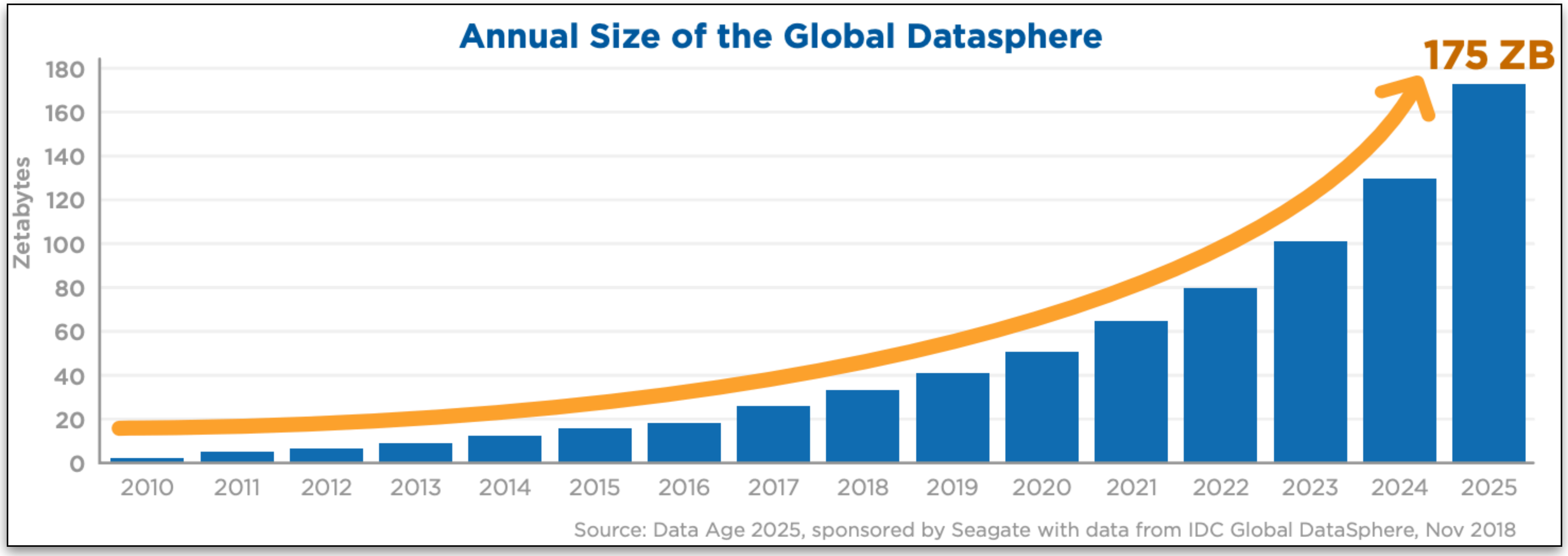
Published Nov 11, 2020

We launched Google Photos more than five years ago with the mission of being the home for your memories. What started as an app to manage your photos and videos has evolved into a place to reflect on meaningful moments in your life. Today, more than 4 trillion photos are stored in Google Photos, and every week 28 billion new photos and videos are uploaded.

Since so many of you rely on Google Photos to store your memories, it’s important that it’s not just a great product, but also continues to meet your needs over the long haul. In order to welcome even more of your memories and build Google Photos for the future, we are changing our unlimited High quality storage policy.

Starting June 1, 2021, any **new** photos and videos you upload will count toward the free 15 GB of storage that comes with every Google Account or the additional storage you’ve purchased as a Google One member. Your Google Account storage is shared across Drive, Gmail and Photos. This change also allows us to keep pace with

https://blog.google/products/photos/storage-changes/



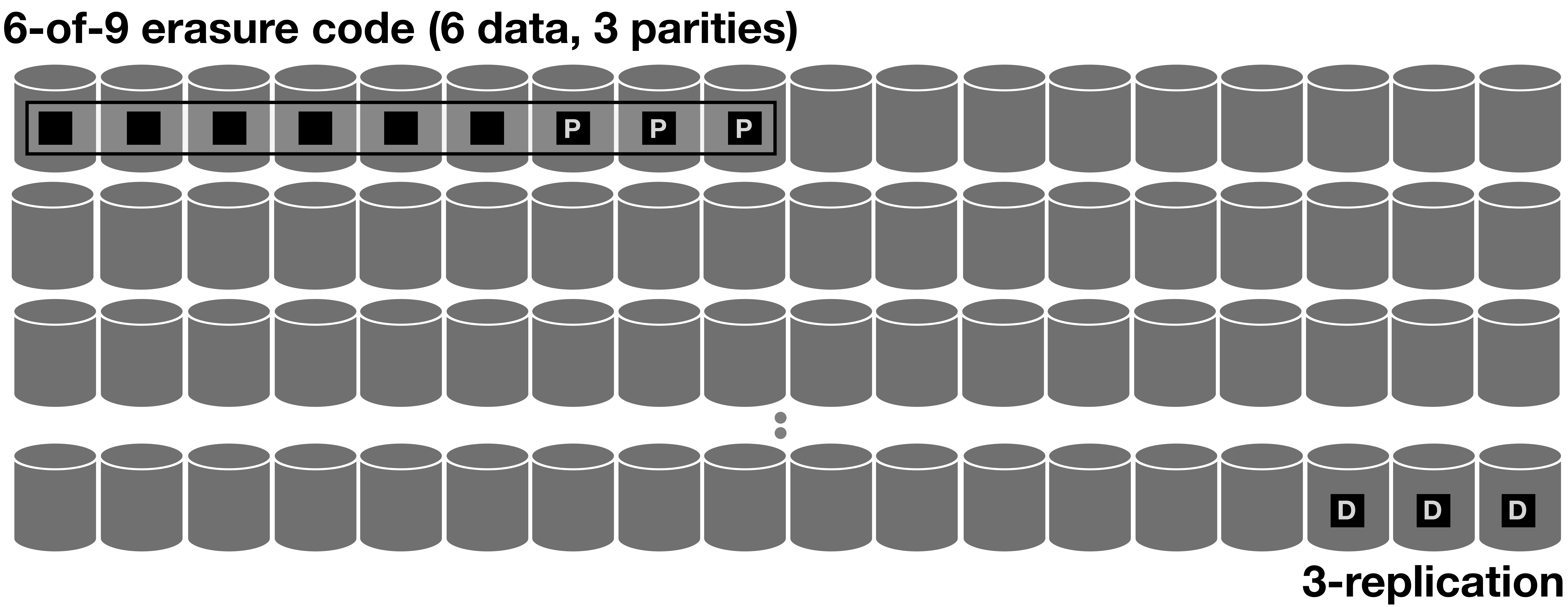

By the end of 2025, over **80%** of the enterprise bytes shipped into the core and edge will continue to be HDD bytes.

In 2025 IDC predicts that **49%** of the world’s stored data will reside in public cloud environments

https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf

**Even single digit % improvements in storage efficiency ⟶ massive savings**

# Current assumption: *all* disks fail similarly




- Multiple redundancy schemes may be used in the entire fleet

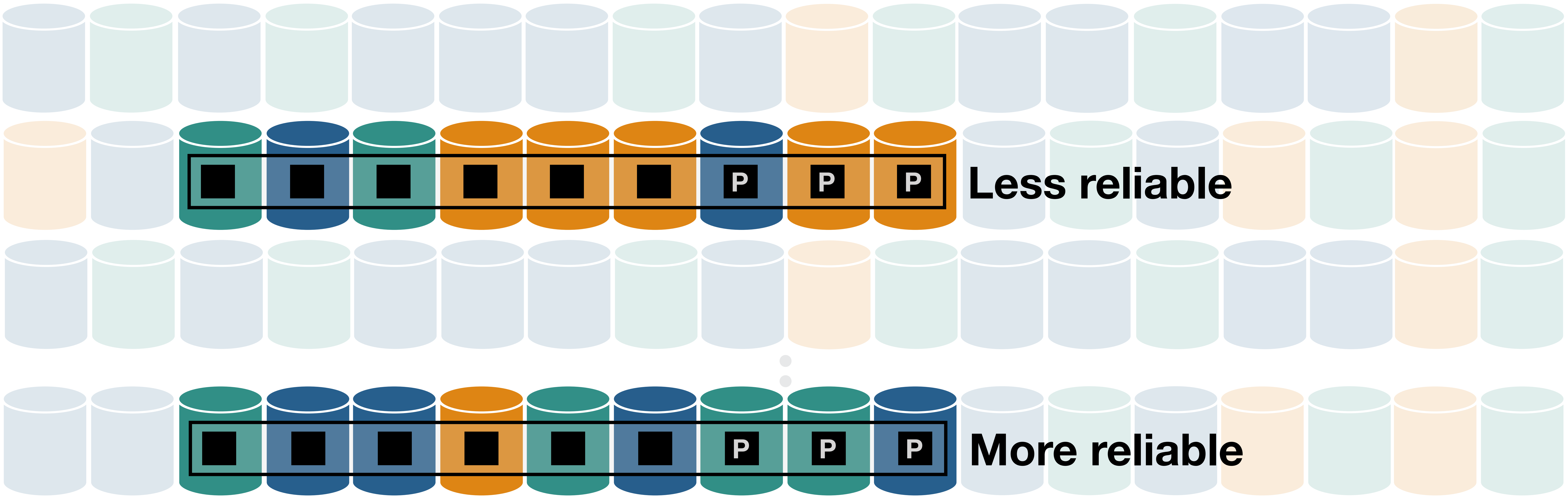
**Redundancy scheme unaware of AFR differences among disks**

# Outline

- Background and motivation
- Disk failure rate heterogeneity
- Making a case for disk-adaptive redundancy (DARE)
- Overcoming transition overload
- Evaluation on real-world cluster traces
- Enabling HDFS to DARE

# Reality: different disks fail differently

Order of reliability: 

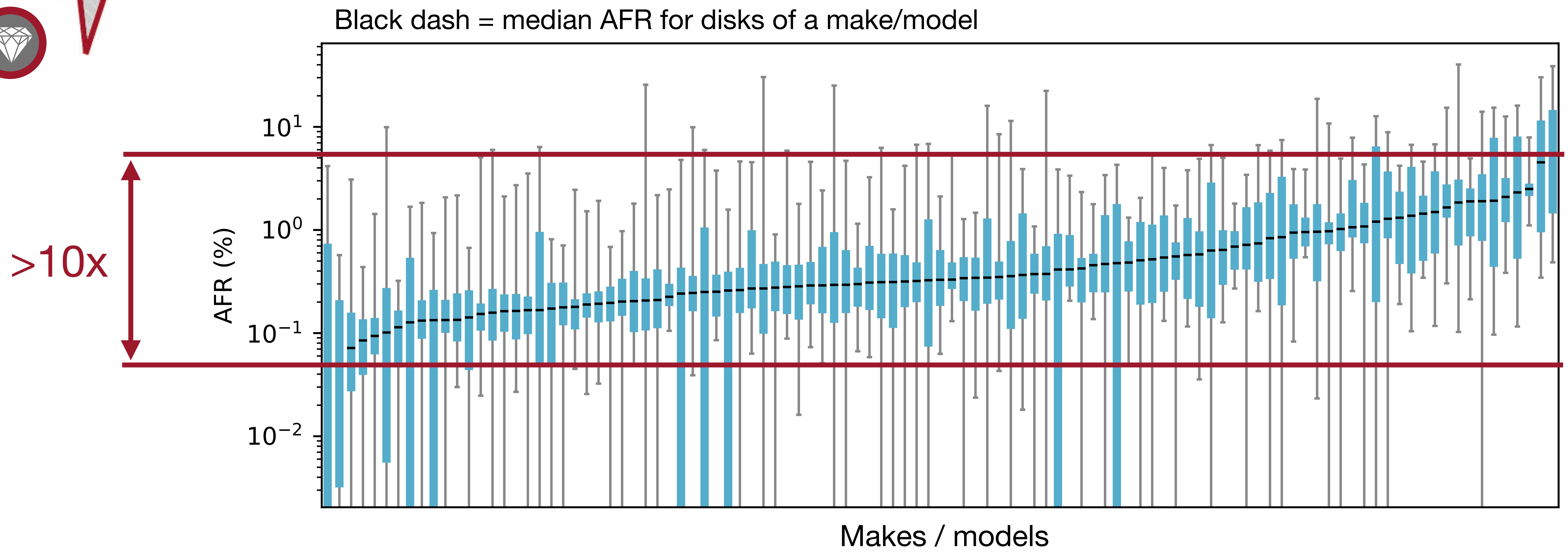



- Single storage cluster typically has multiple makes/models
- Result: stripes (or replicas) may provide different reliability

**Same redundancy is either insufficient or wasteful, mostly the latter**

DARE

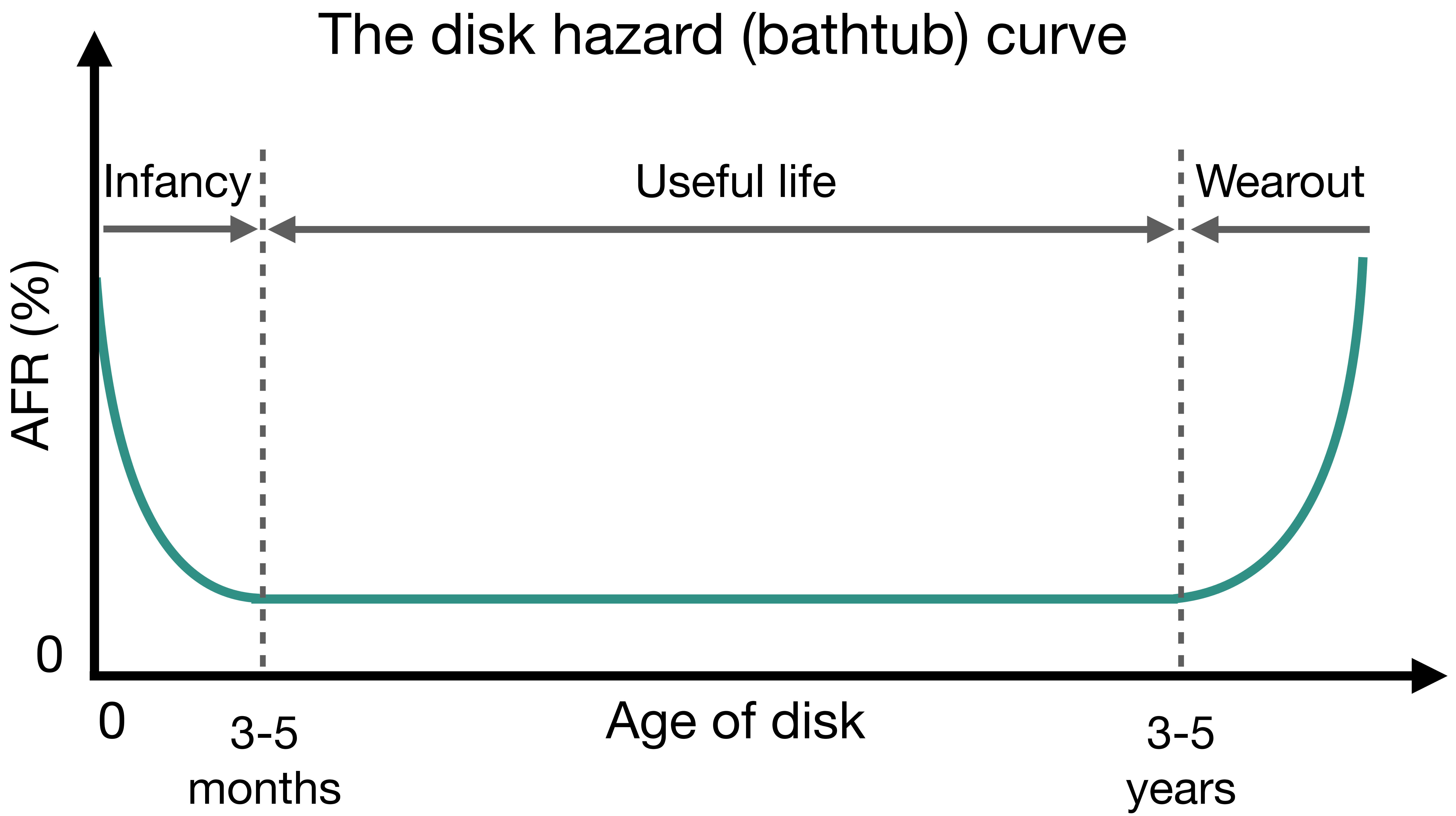
# AFR varies across makes/models




- 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**

# AFR varies across age

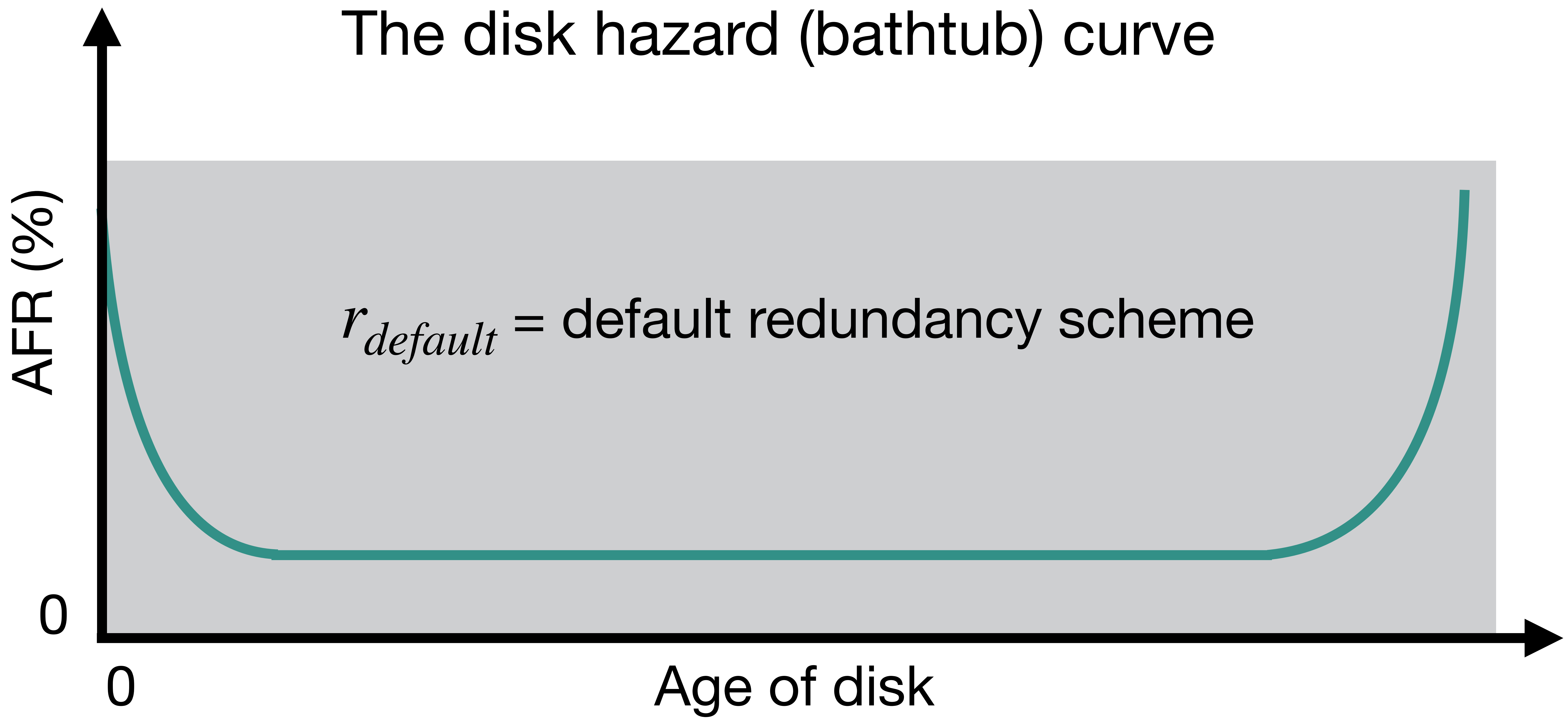



Failure rate varies over a disk's lifetime

# Default redundancy used throughout life

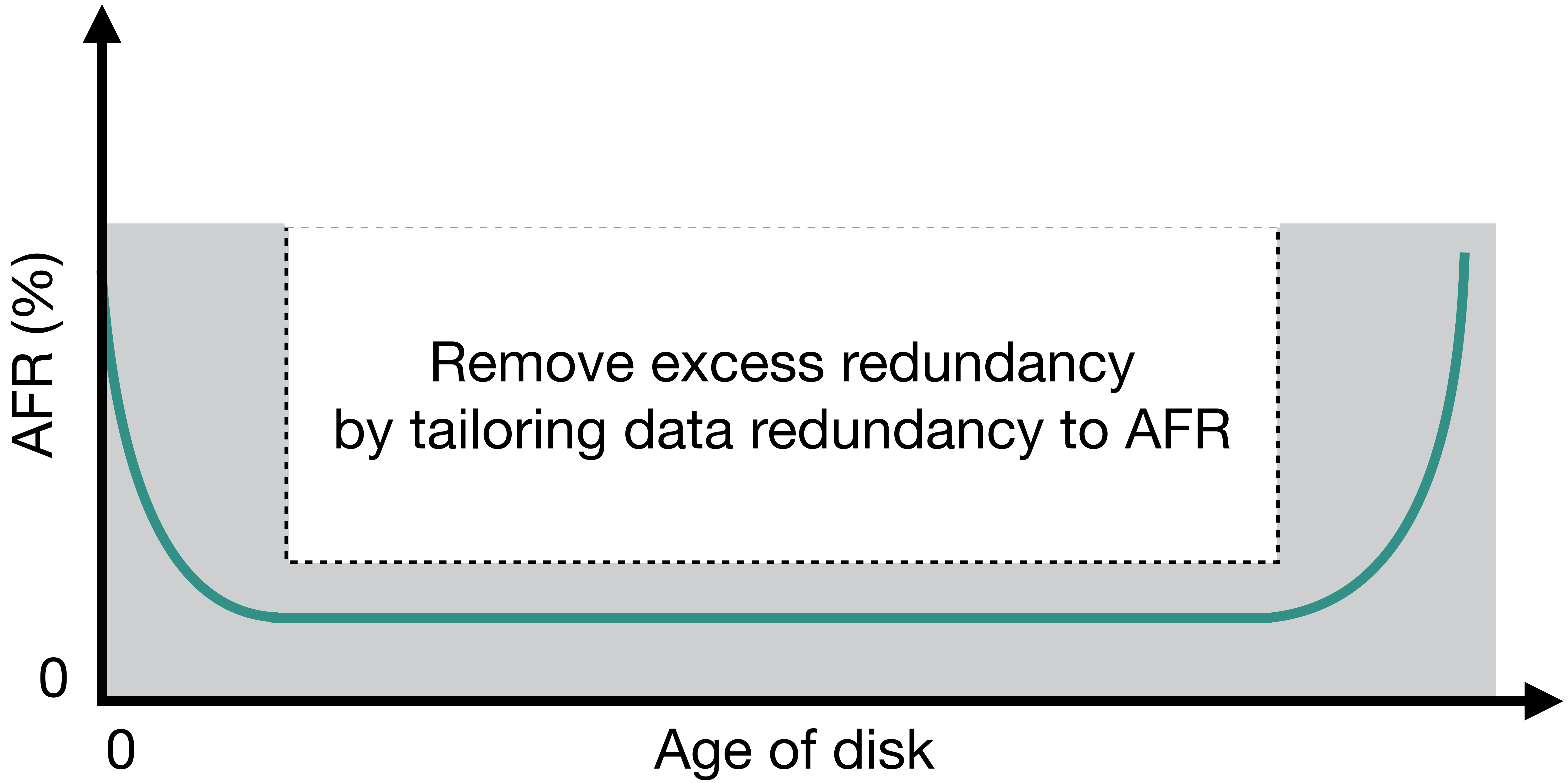
DARE

- Redundancy scheme chosen on some default AFR value
  - Default AFR is typically high enough to be higher than any observed AFR




$r_{default}$ used on all disks throughout disk's life

# Dare to DARE: tailor redundancy to AFR




Lower AFR ⟶ Lower redundancy ⟶ Lower storage cost

# Exploiting AFR heterogeneity

- **Challenges** for DARE systems:
  1. Need to **monitor AFRs in the field**
  2. Need to **handle AFR heterogeneity across makes/models**
  3. Need to **handle AFR heterogeneity across age**

- **Goals** of DARE systems:
  1. **Safe**: protect data sufficiently
  2. **Accurate**: identify different reliability phases, redundancy transitions correctly
  3. **Online**: realize low-AFR opportunities to optimize redundancy on-the-fly
  4. **Efficient**: perform redundancy transitions with minimal interference

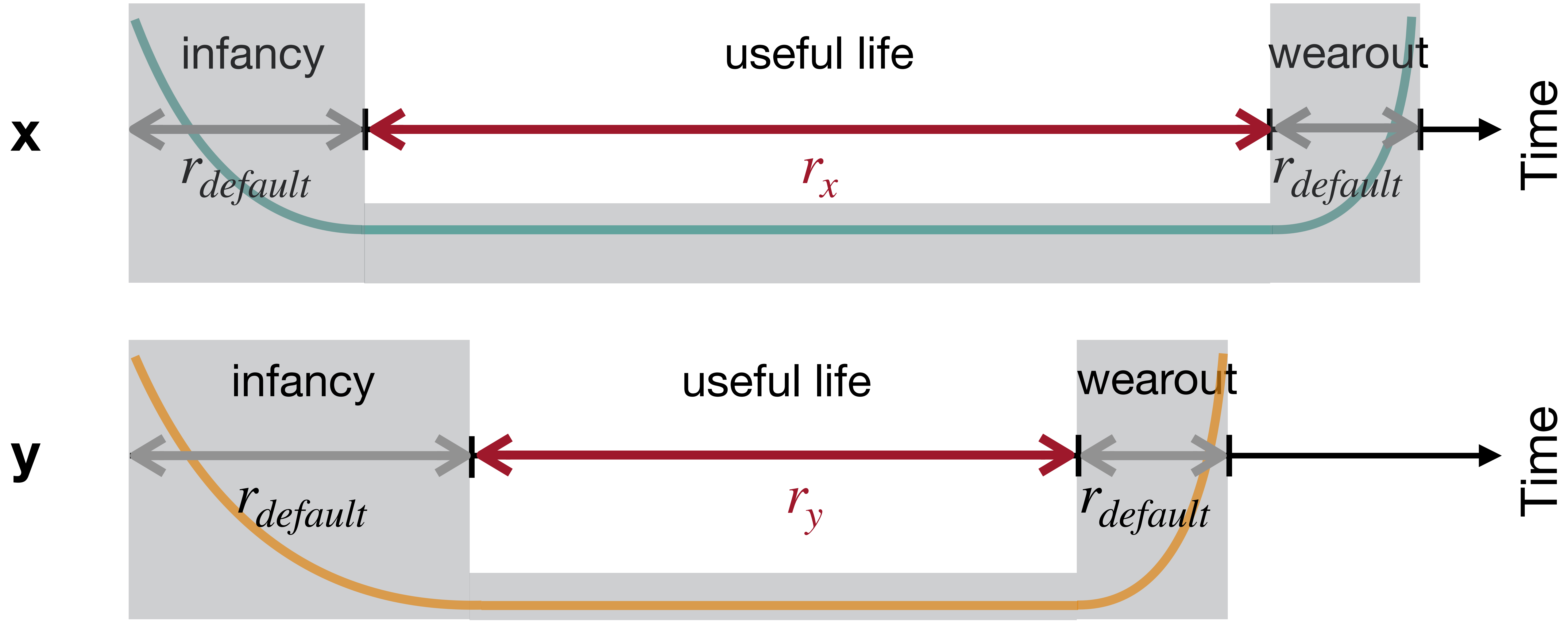
- **Benefits** of DARE systems:
  1. **Safer redundancy**: system dynamically adapts redundancy to AFR changes
  2. **Cost-effective**: provides reduced storage, operational and energy cost

DARE

# DARE for multiple makes/models

$r_{default}$ = default (existing) fault tolerance scheme

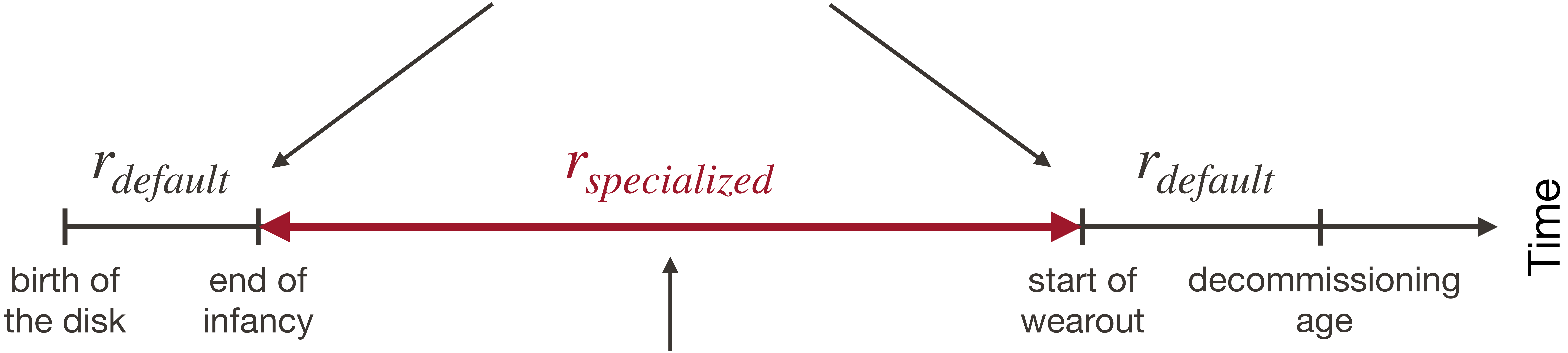
$r_{specialized}$ = tailored redundancy scheme




$r_{specialized}$ defined per make/model's useful life

# DAREing disk timeline

When should disks transition?



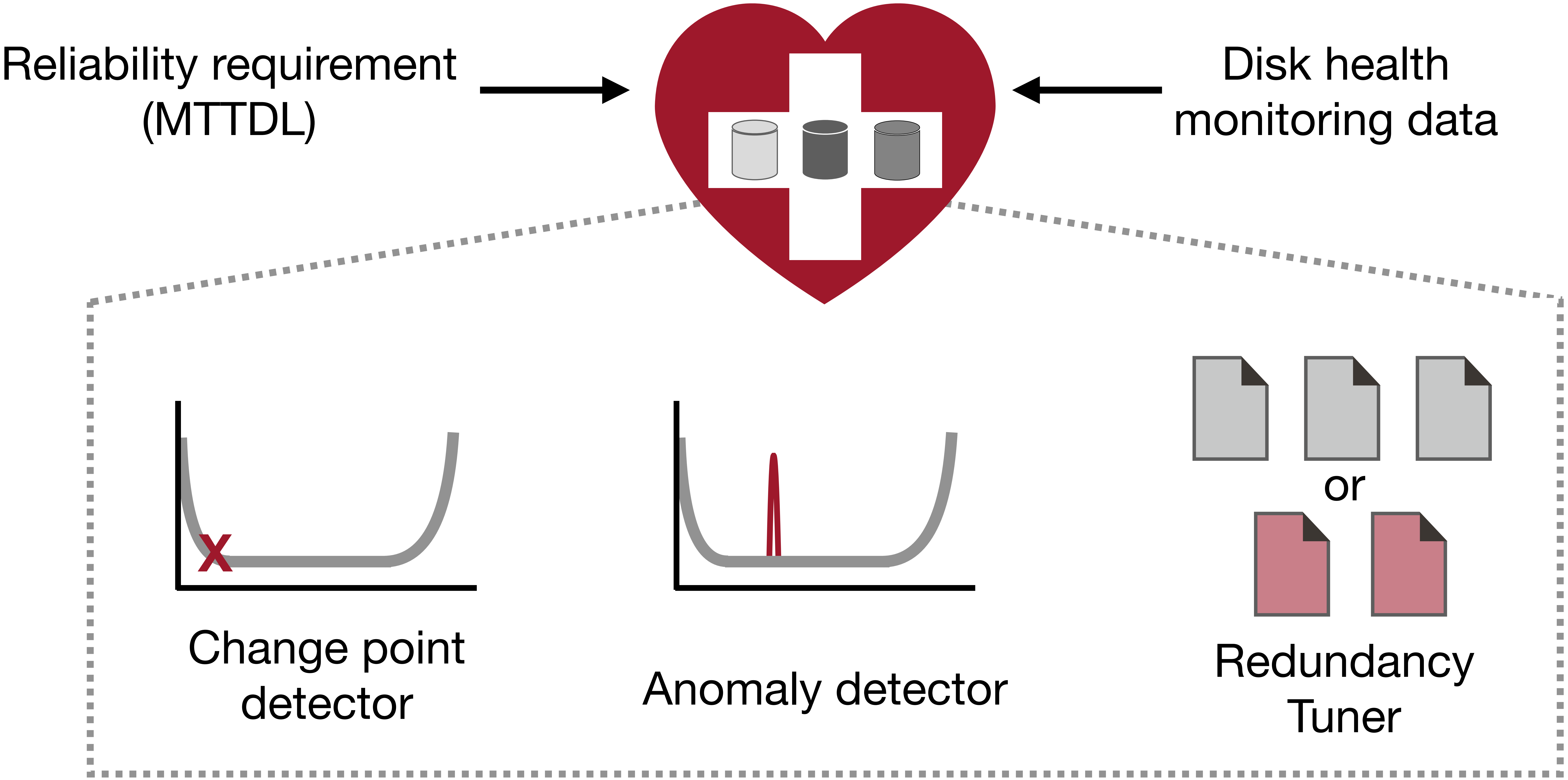

What scheme should disks transition to?

# Outline

- Background and motivation
- Disk failure rate heterogeneity
- Making a case for disk-adaptive redundancy (DARE)
- Overcoming transition overload
- Evaluation on real-world cluster traces
- Enabling HDFS to DARE

# First DARE system: **HeART**

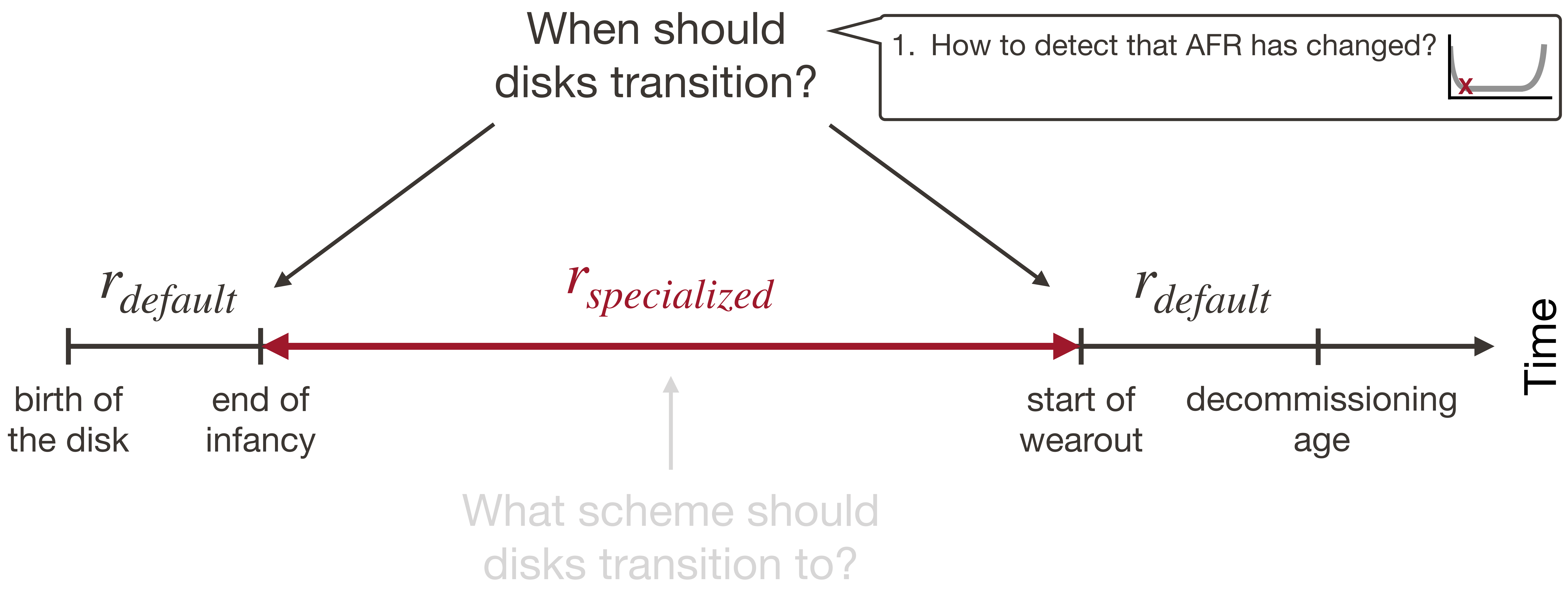
**He**tereogeneity-**A**ware **R**edundancy **T**uner (HeART)



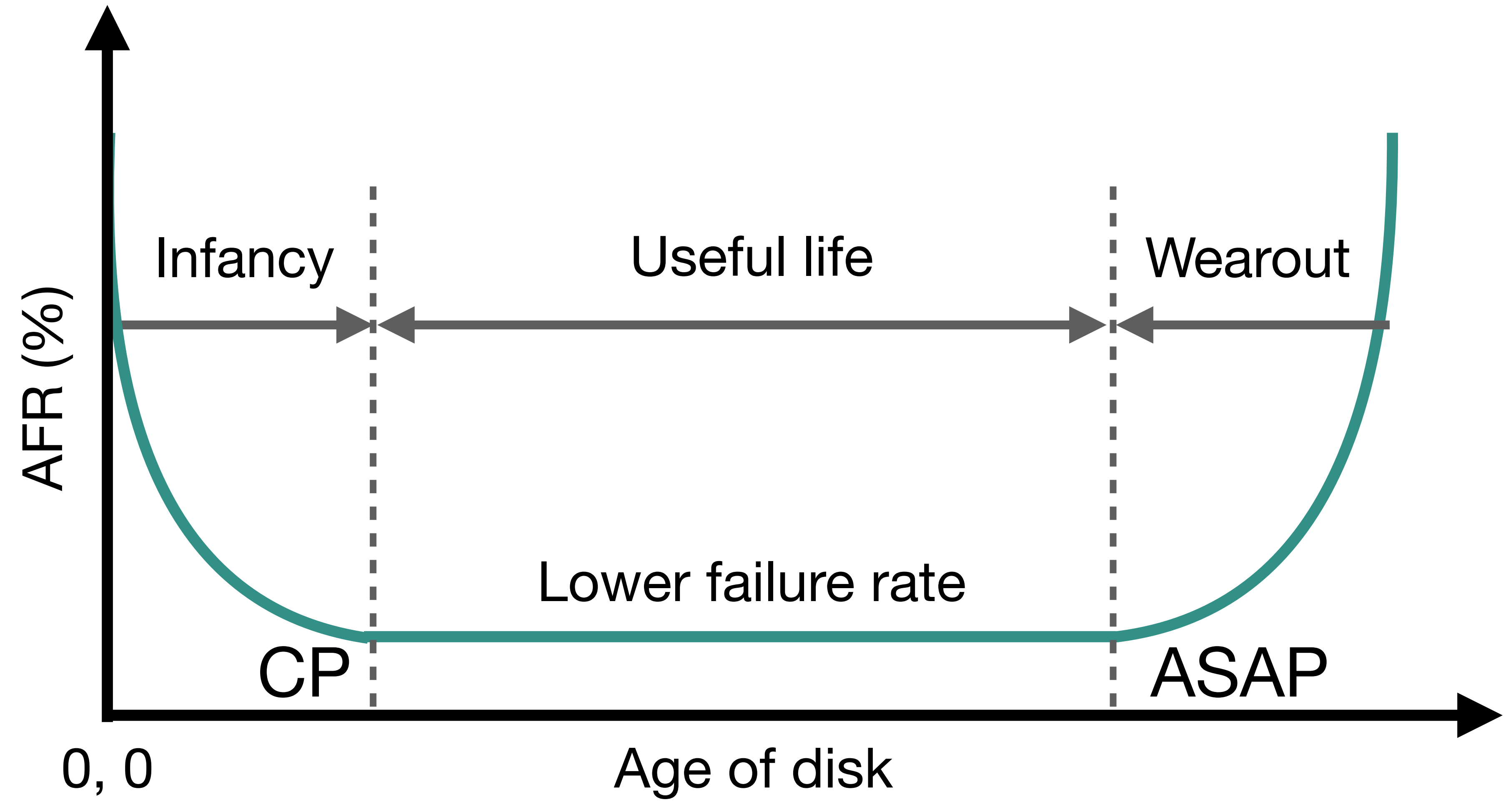

Published in USENIX FAST 2019

**Saurabh Kadekodi**, K. V. Rashmi, and Gregory R. Ganger

# DAREing disk timeline (when?)



# When to transition impacts data reliability

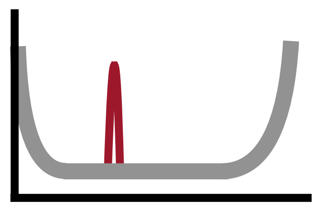


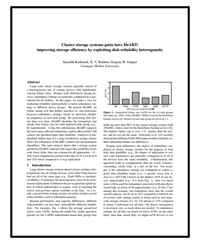

- Data can be under-reliable if:
  - End of infancy is declared too early
  - Onset of wearout is declared too late
- HeART uses a **change point detector** to identify end of infancy
- Change to wearout happens as soon as AFR nears $r_{specialized}$ threshold

# DAREing disk timeline (when?)

When should
disks transition?

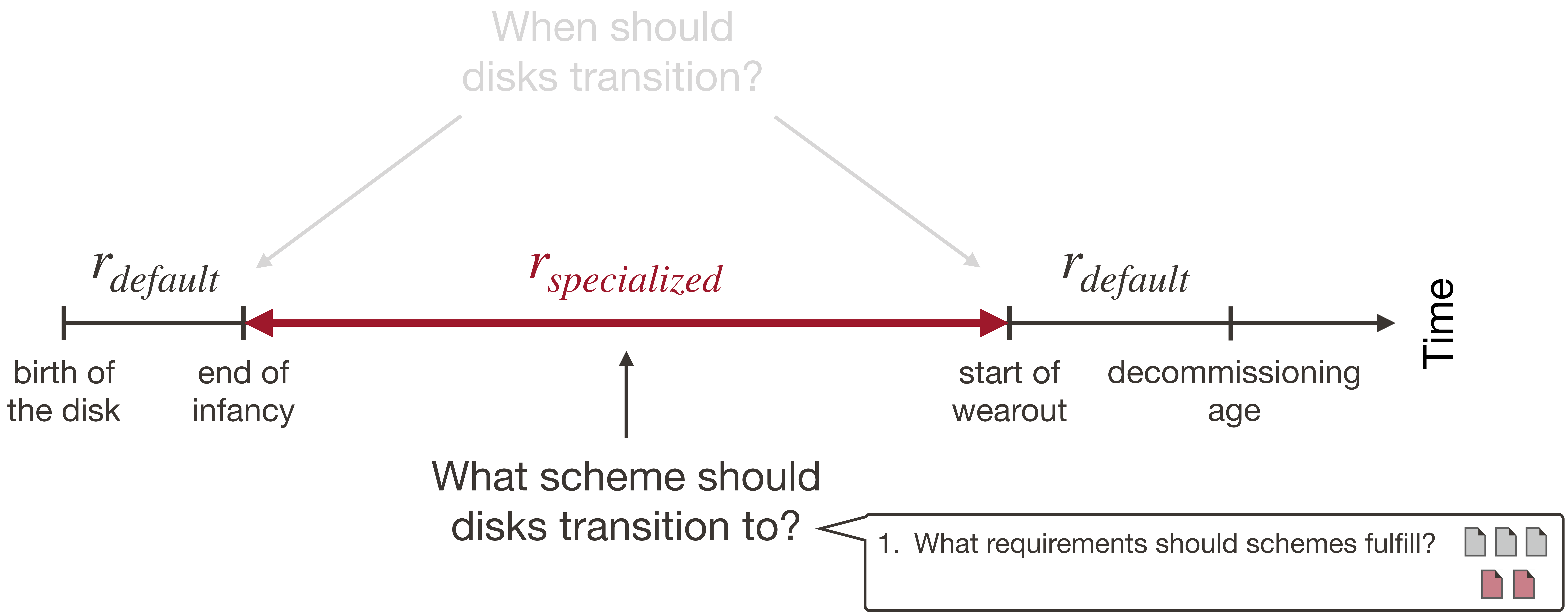
1. How to detect that AFR has changed?
2. How can we trust the observed curve?



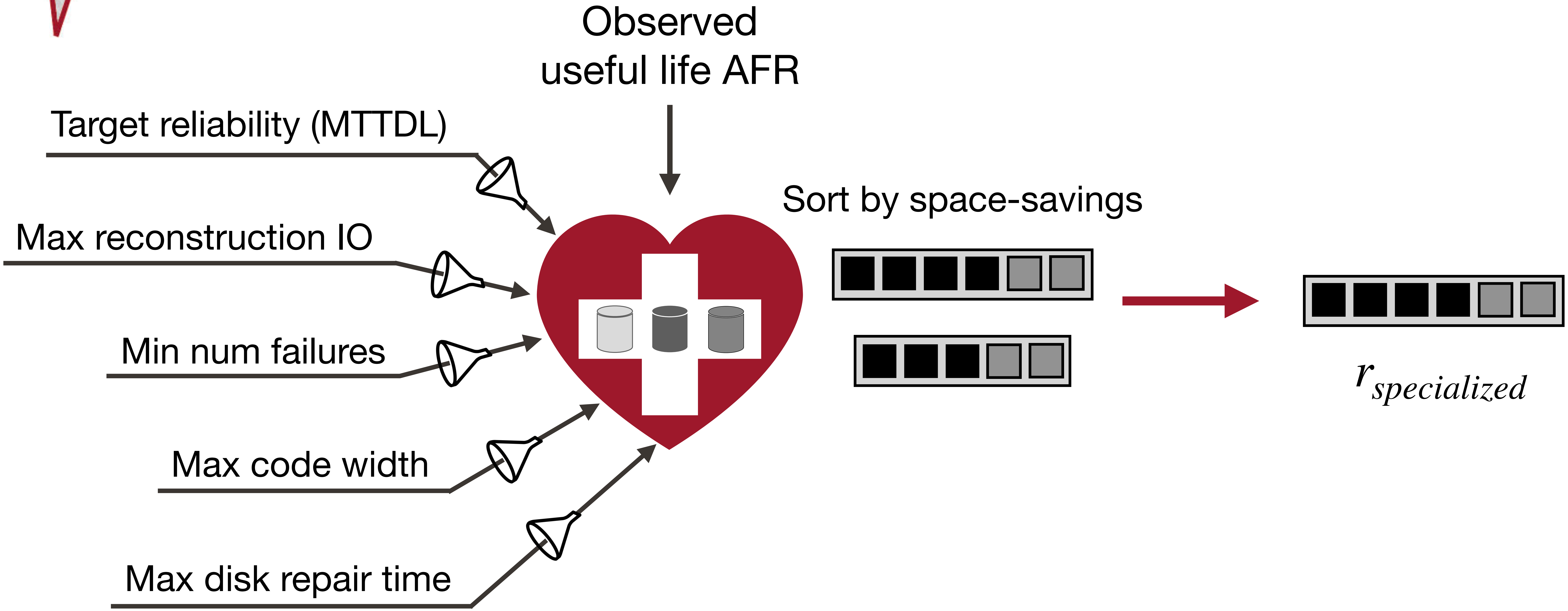



What scheme should
disks transition to?

# DAREing disk timeline (what?)



# Constraint based $r_{specialized}$ selection



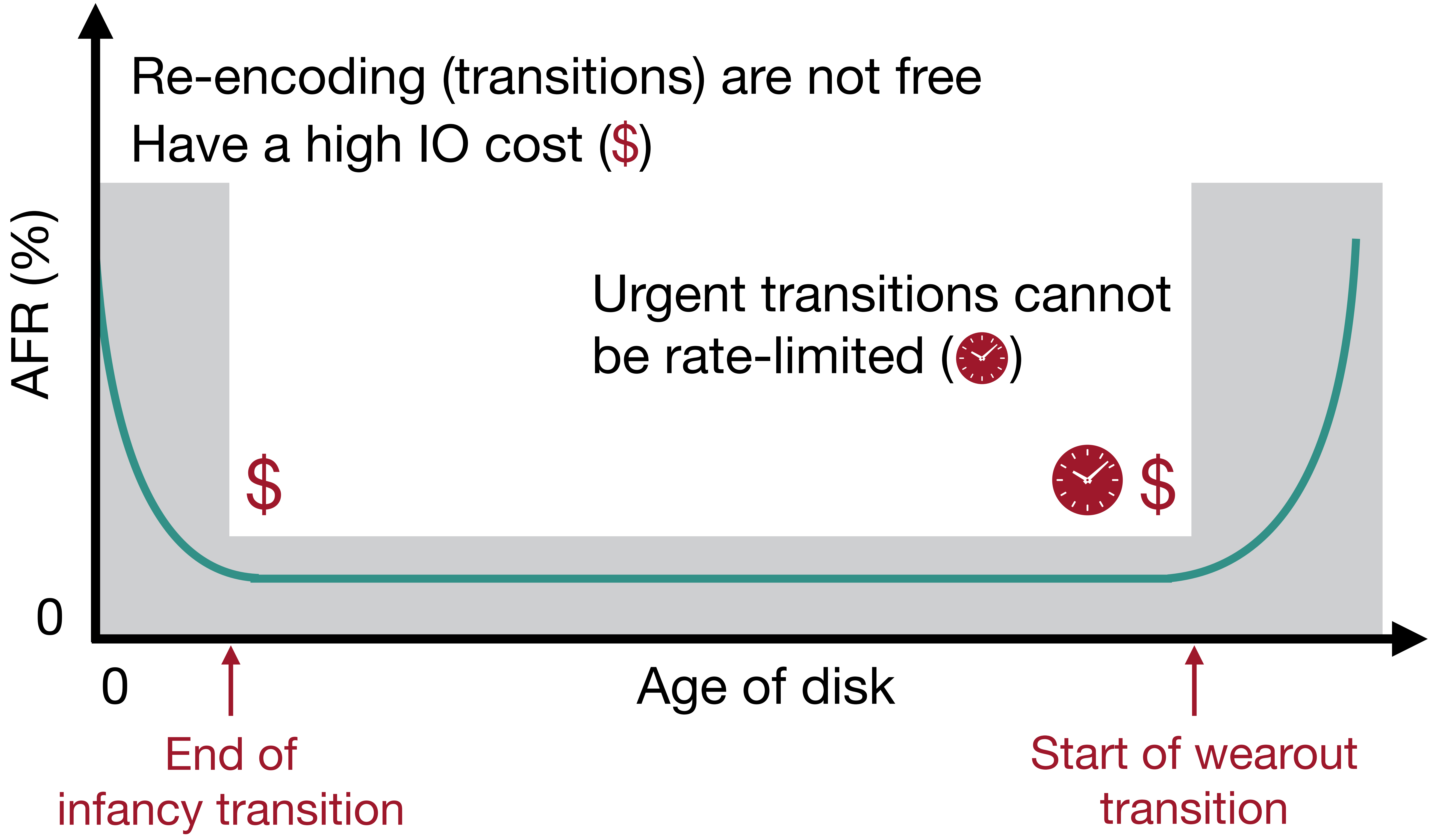

Constraint-driven redundancy scheme selection

# HeART provides huge benefits

- HeART evaluated on reliability trace of storage cluster with over 100K HDDs
- Promised substantial storage space-savings over “one-scheme-fits-all” redundancy:
  - 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 ⟶ 1000s of fewer disks
  - Much lower storage cost
  - Significantly lower carbon footprint

however…

# HeART suffers from transition overload




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

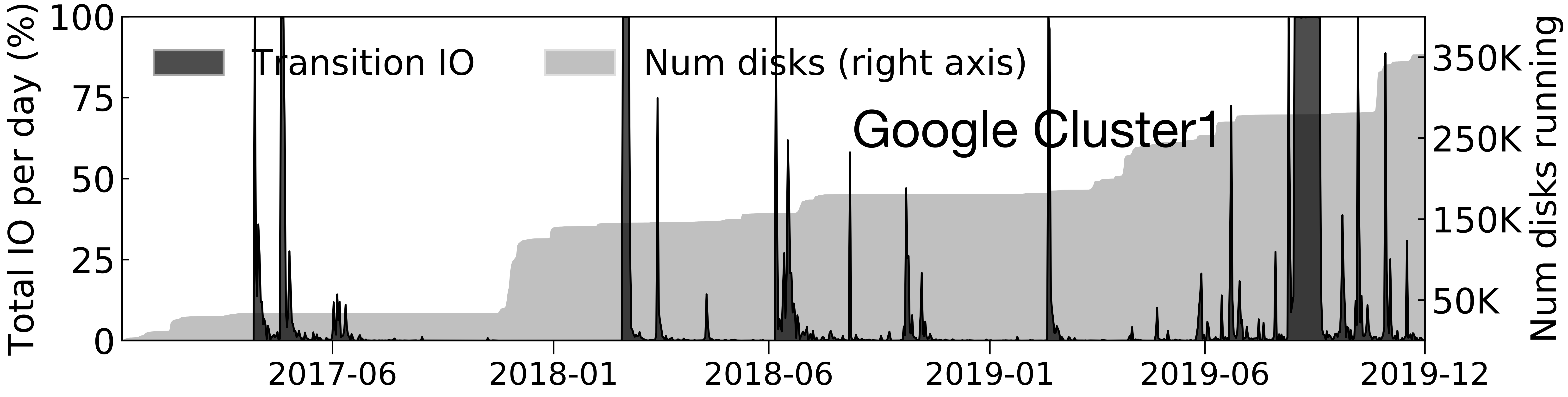
# Outline

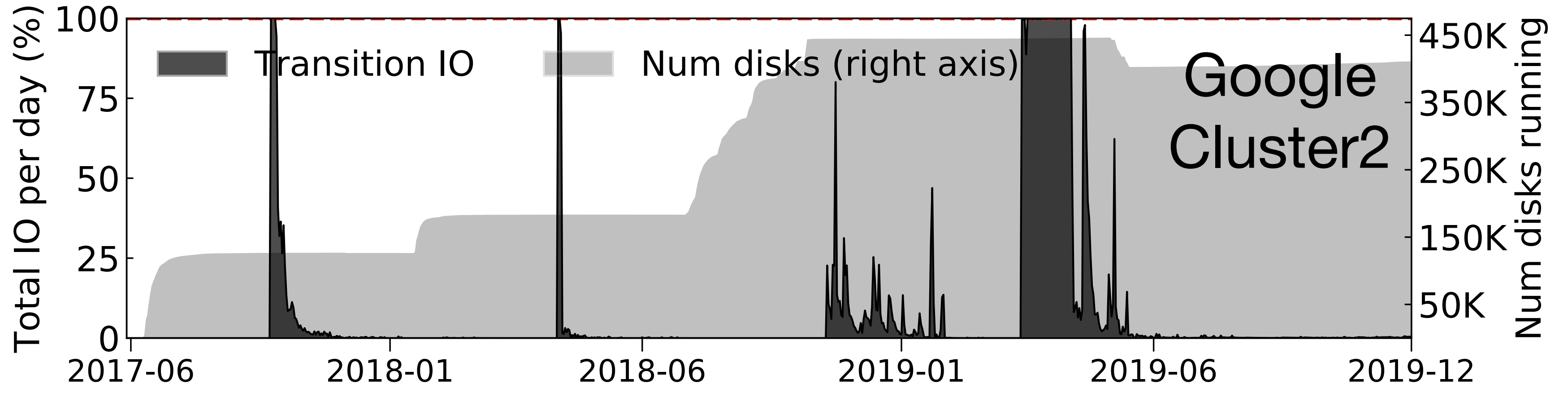
- Background and motivation
- Disk failure rate heterogeneity
- Making a case for disk-adaptive redundancy (DARE)
- Overcoming transition overload
- Evaluation on real-world cluster traces
- Enabling HDFS to DARE

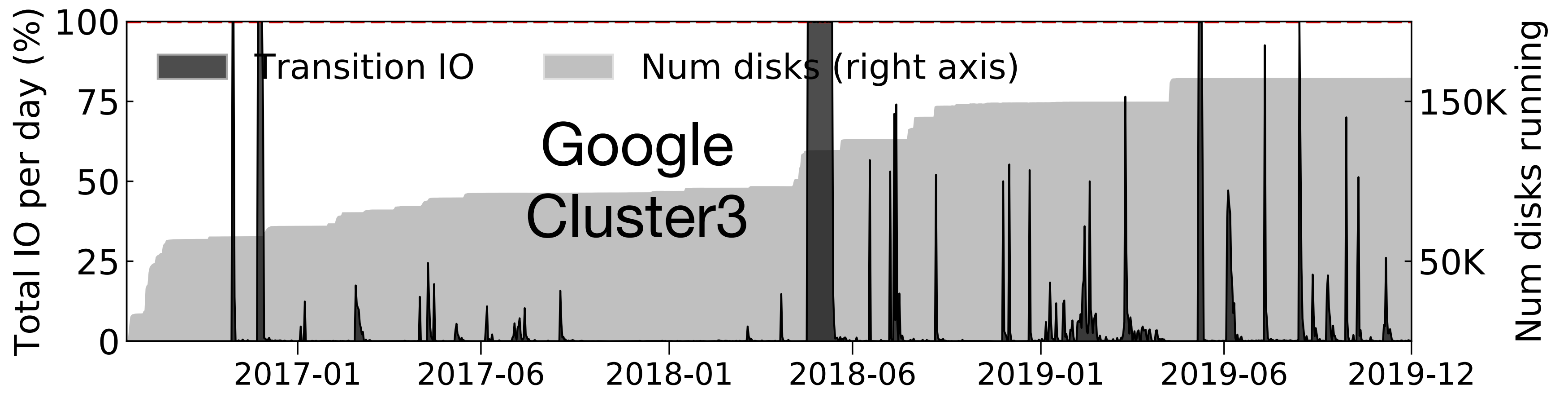
DARE

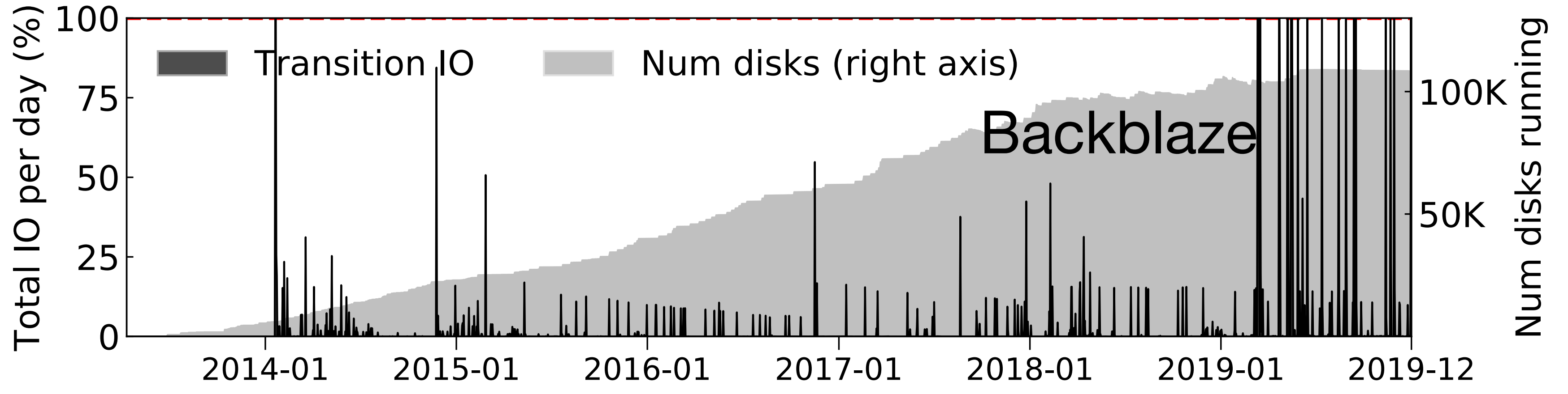
# Transition overload causes HeART attacks

- HeART simulated on clusters



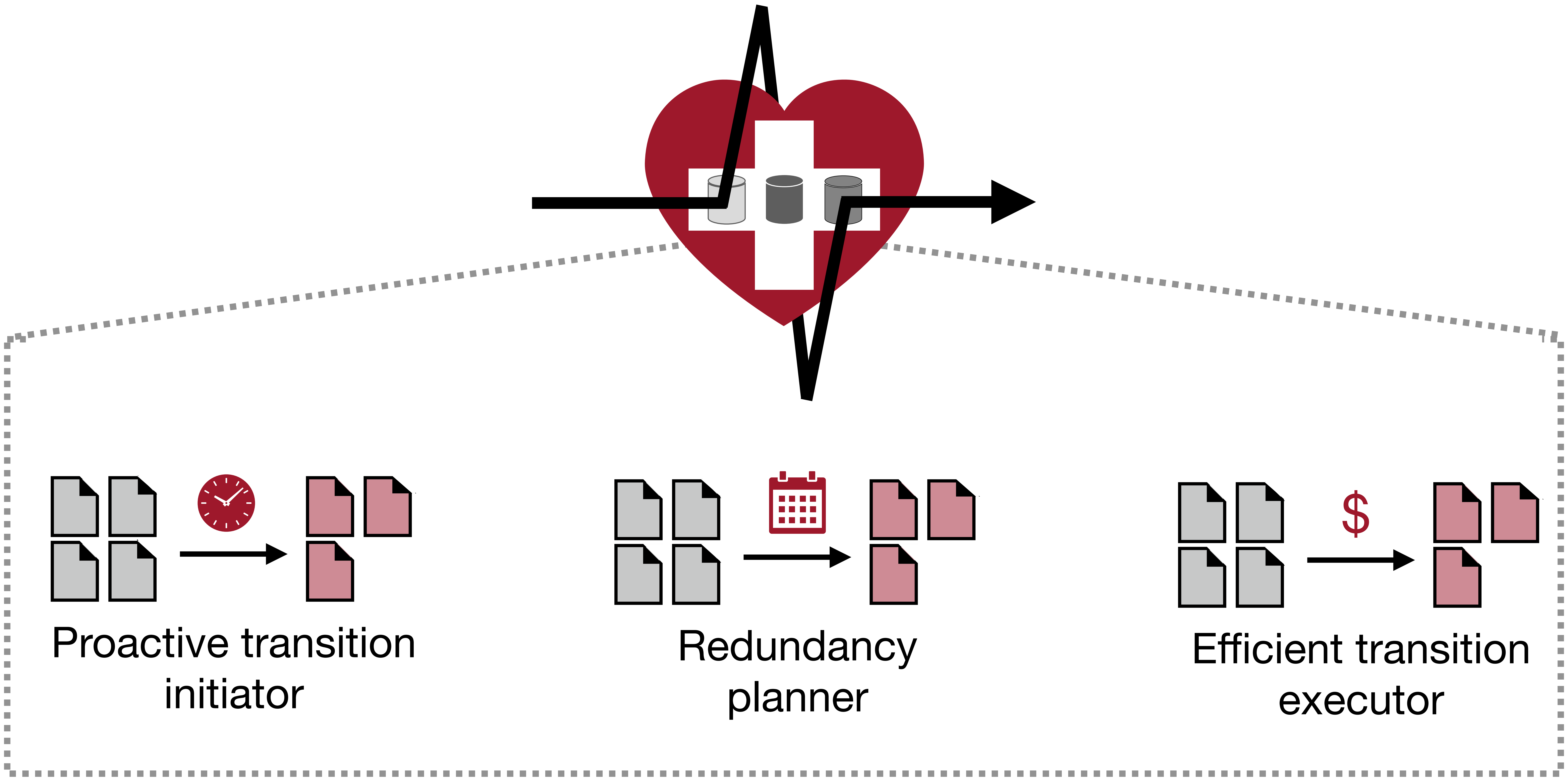




- Weeks of 100% cluster IO bandwidth spent in transitions
  - caused by costly transitions
  - in addition to too many disks transitioning together

# **Pacemaker:** regulating the HeART

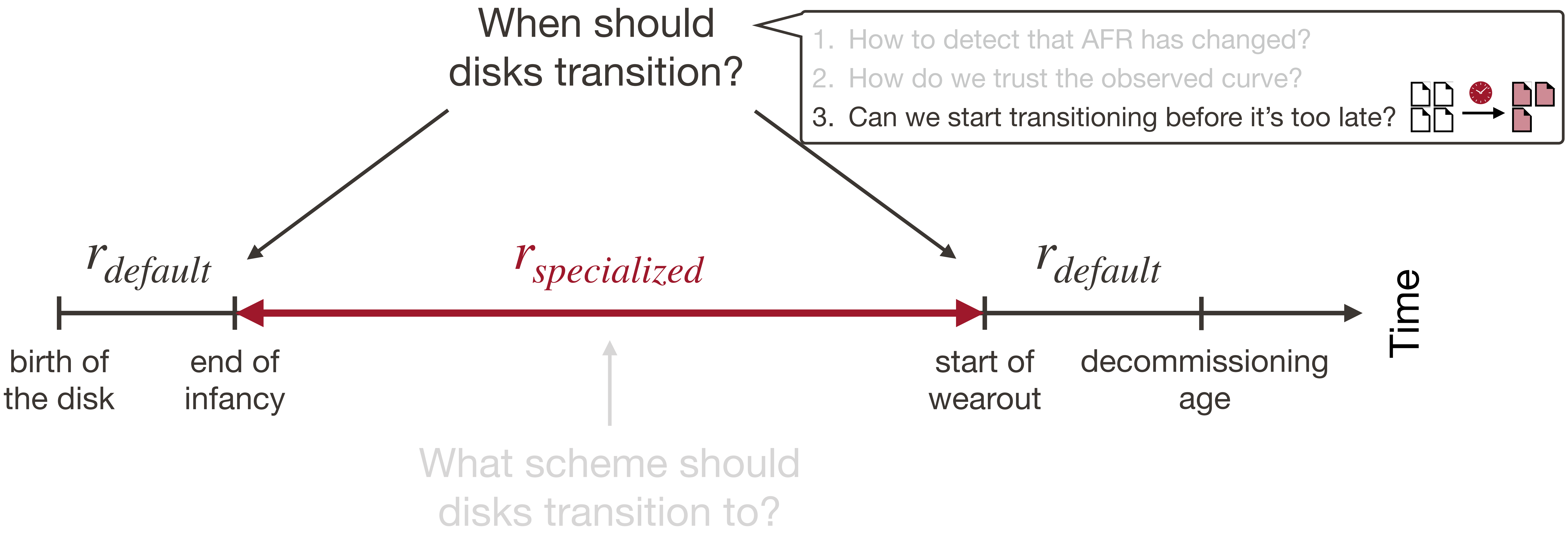



Published in USENIX OSDI 2020

**Saurabh Kadekodi**, Francisco Maturana, Suhas Subramanya Jayaram, Juncheng Yang, K. V. Rashmi, and Gregory R. Ganger

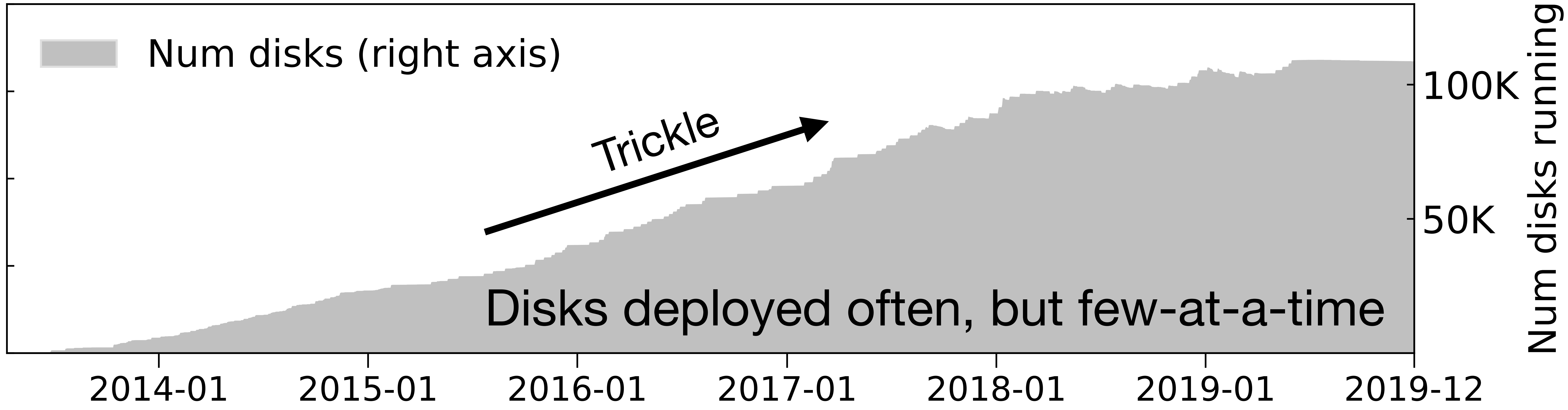
DARE

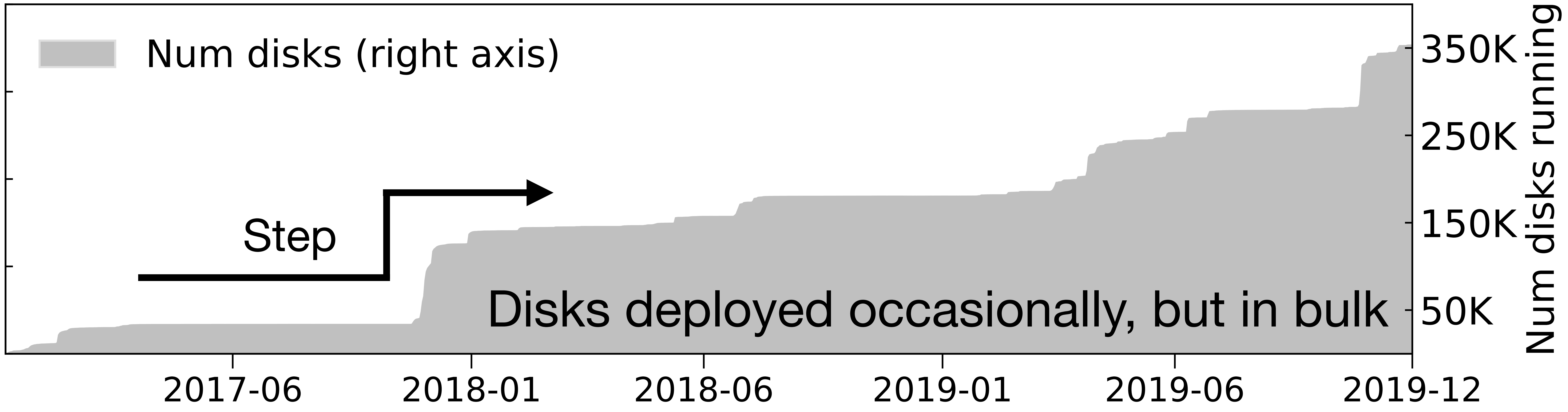
# DAREing disk timeline (when?)



# Two disk deployment patterns

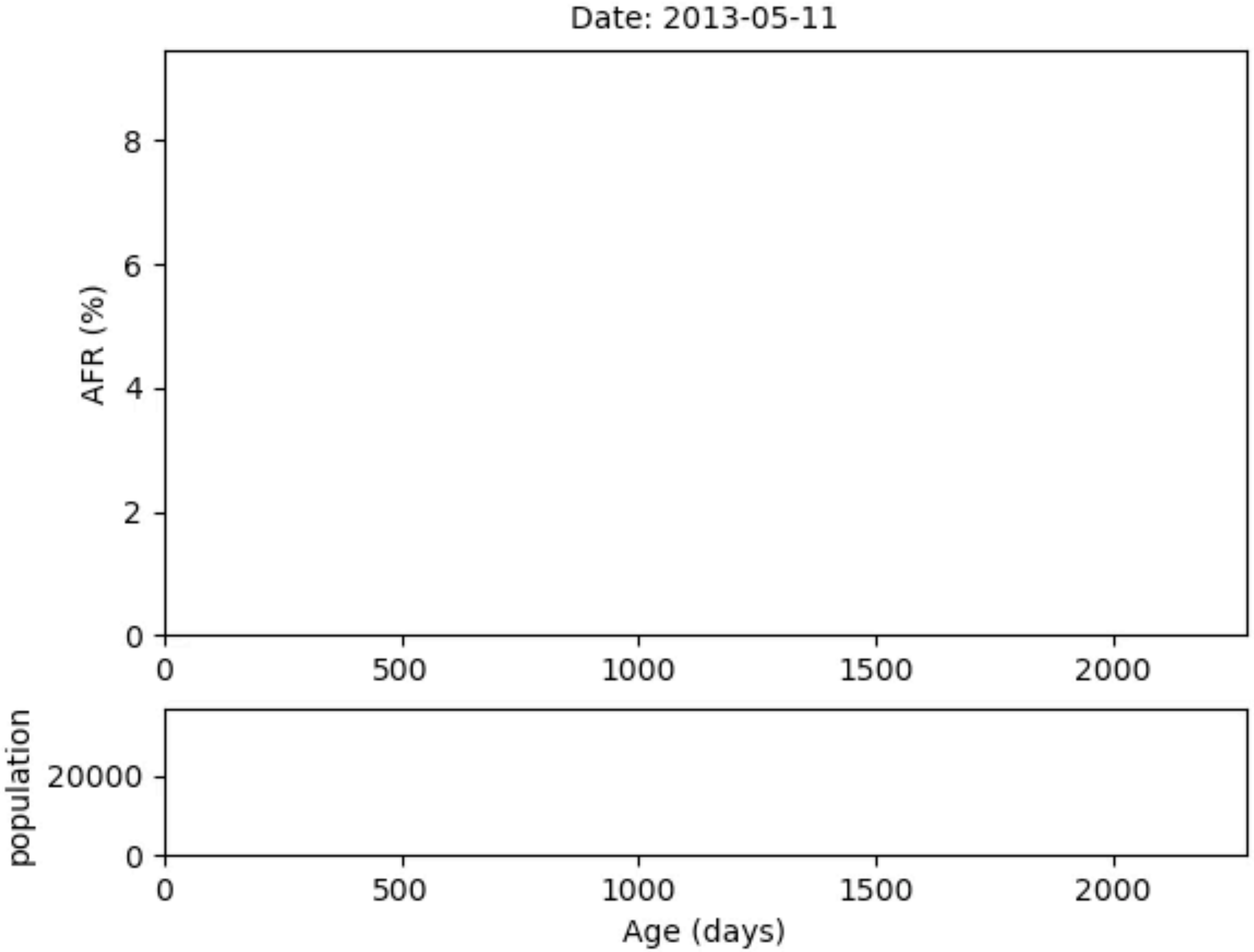
DARE



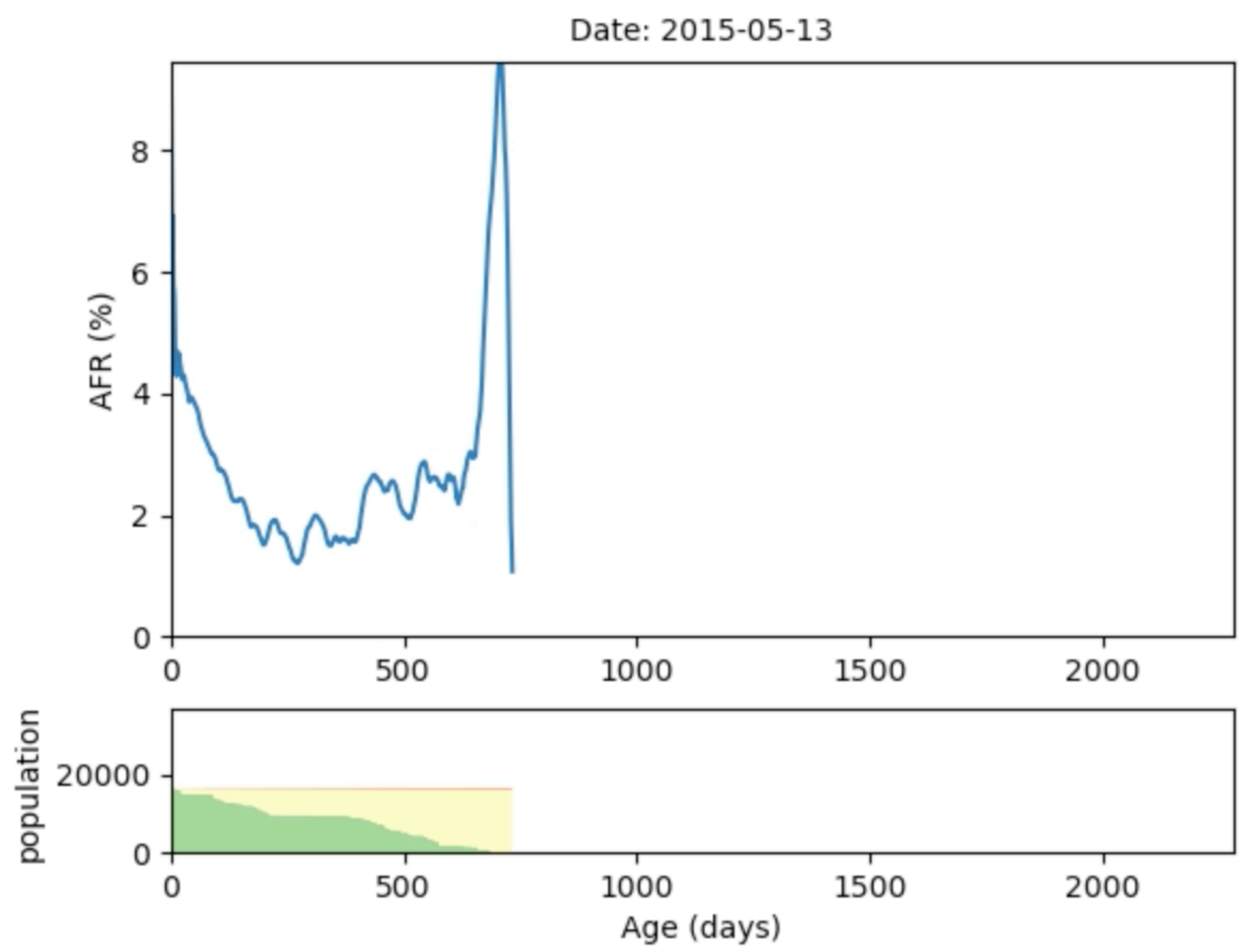



# Trickle-deployed disks have jittery AFR

- AFR for any age known only after few 1000 disks cross that age



# Canaries help proactive trickle transitions

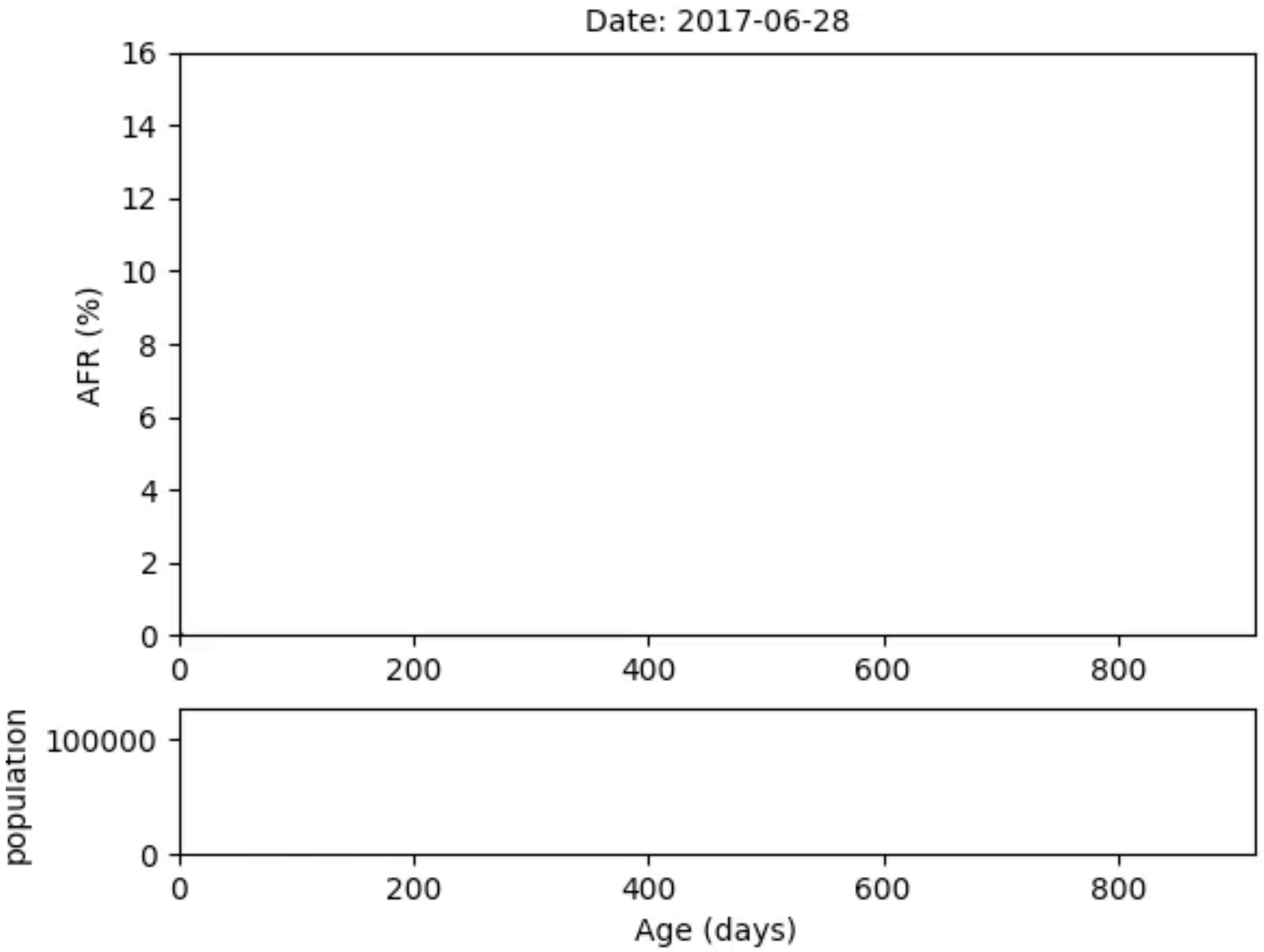



- Pacemaker marks **first *C* disks as canary disks**
  - Learns the AFR curve from canaries
  - Does not optimize redundancy for canary disks
- Remaining trickle-deployed disks can be proactively transitioned
  - Canaries educates Pacemaker of age when AFR rises

Canary disks help in proactively transitioning trickle-deployed disks

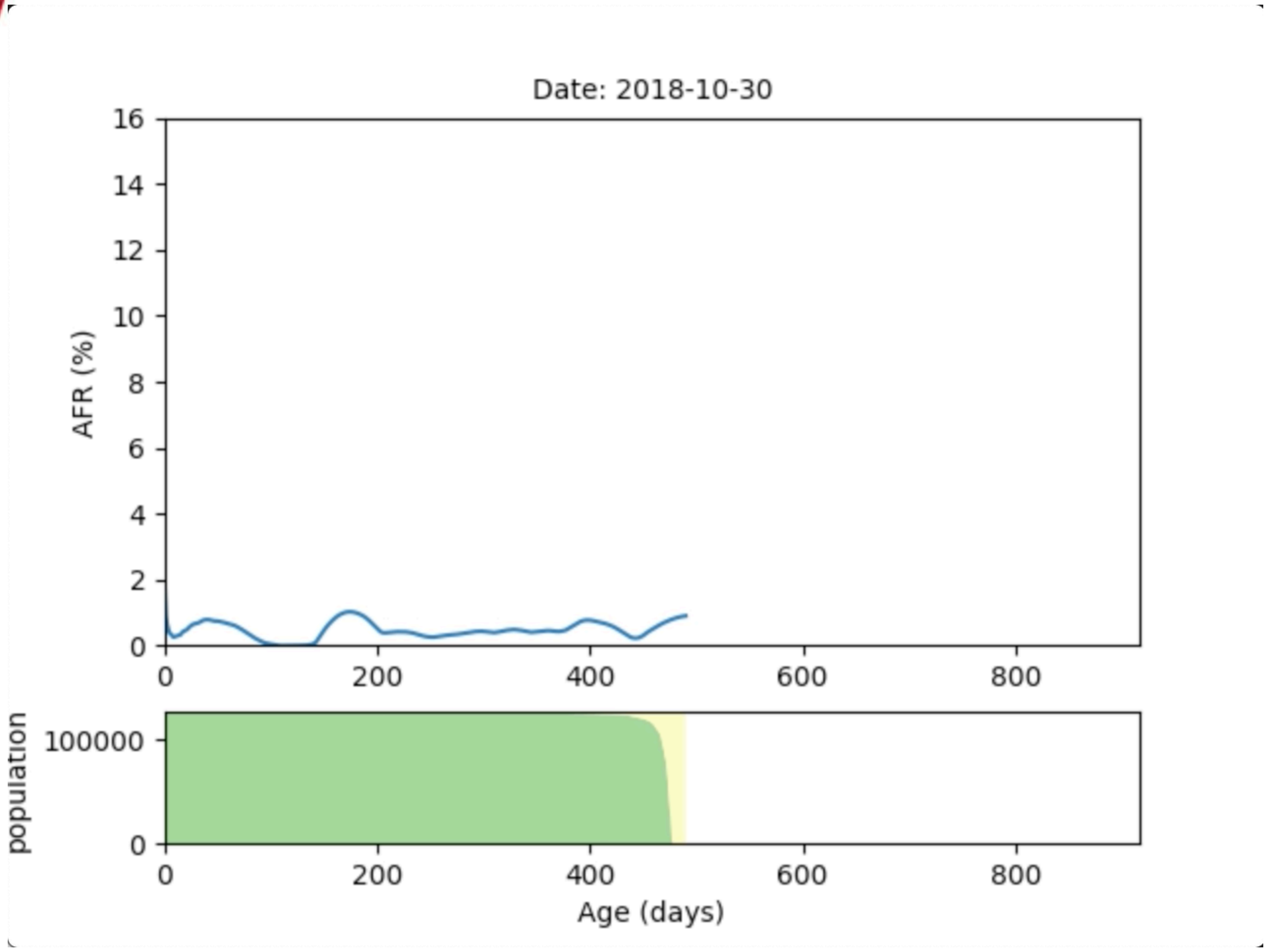
# Confidence in AFR of step-deployment

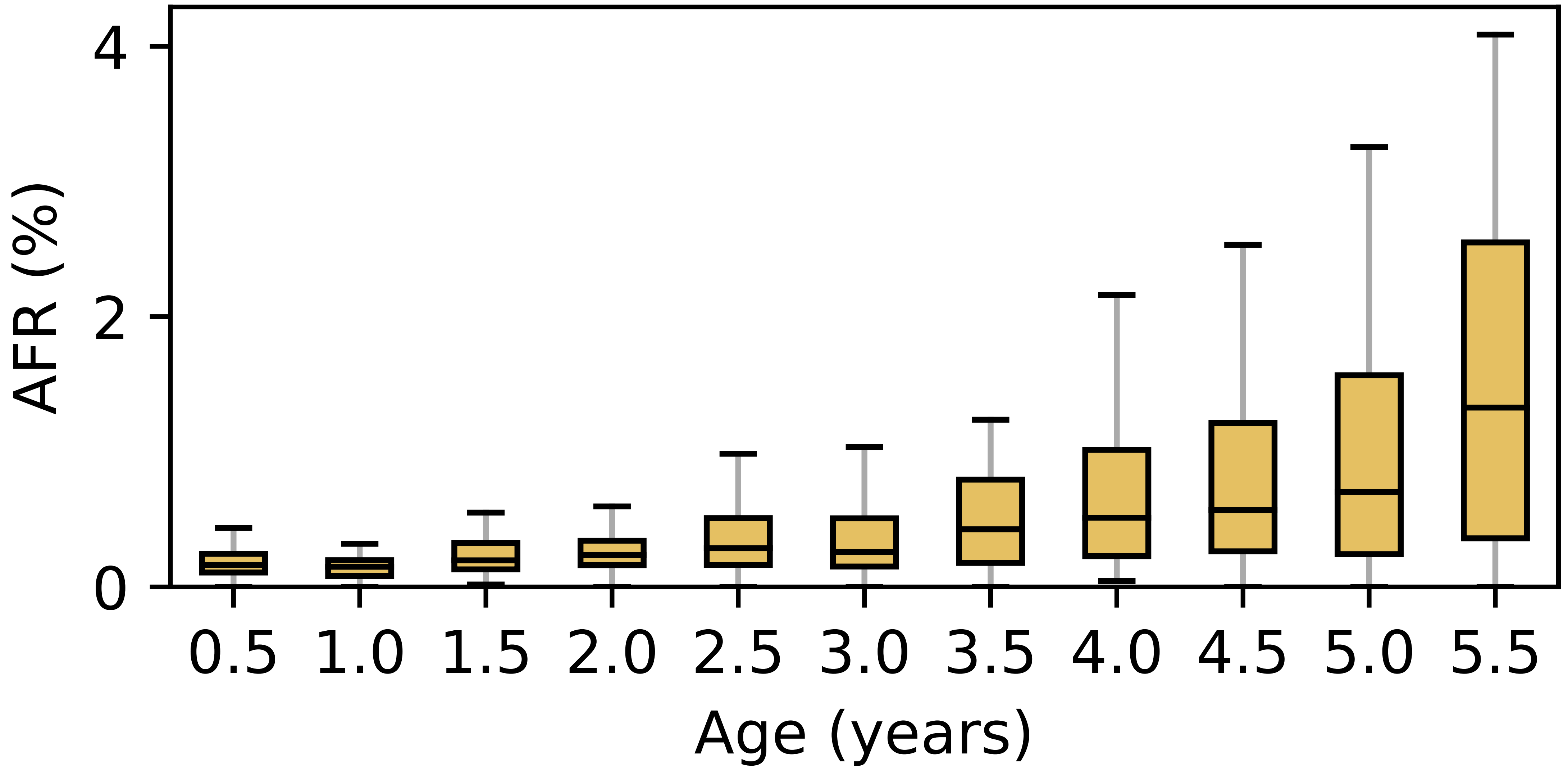
- Step-deployed disks deployed together (canaries ⟶ most disks unspecialized)
- Step-deployed disks gives high-confidence AFR (most disks have the same age)



# Early warning proactively transitions step

DARE




- AFRs rise gradually through useful life phases towards wearout
- Pacemaker uses stable AFR + gradual AFR rise as “early-warning”
  - Threshold AFR for each $r_{specialized}$ when crossed triggers transition

“Early warning” triggers transitions for step-deployed disks

# DAREing disk timeline (how?)

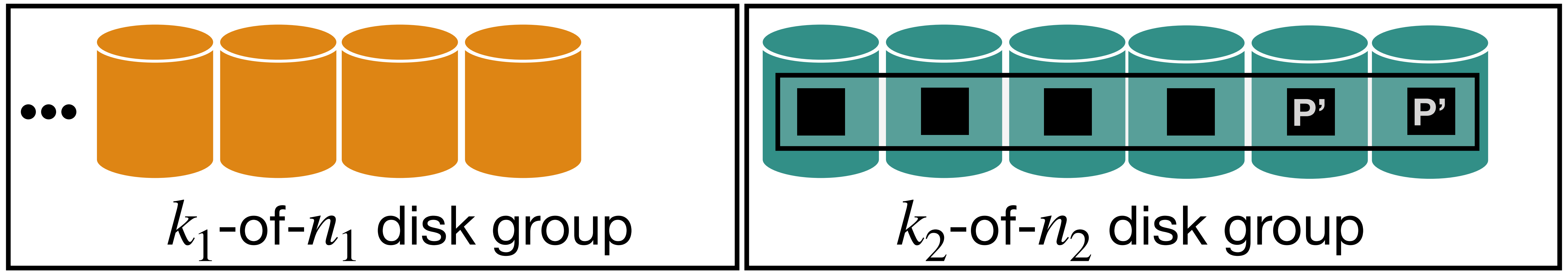
When should disks transition?


What scheme should disks transition to?

How should the disks transitions?

Can transition IO itself be reduced?

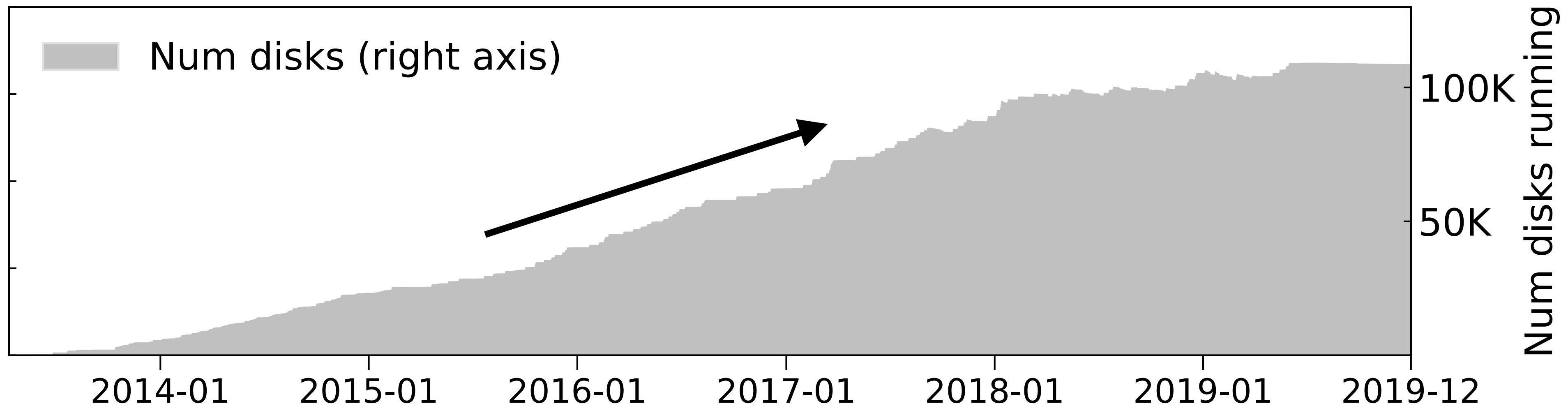
# "Read—re-encode—write" is costly

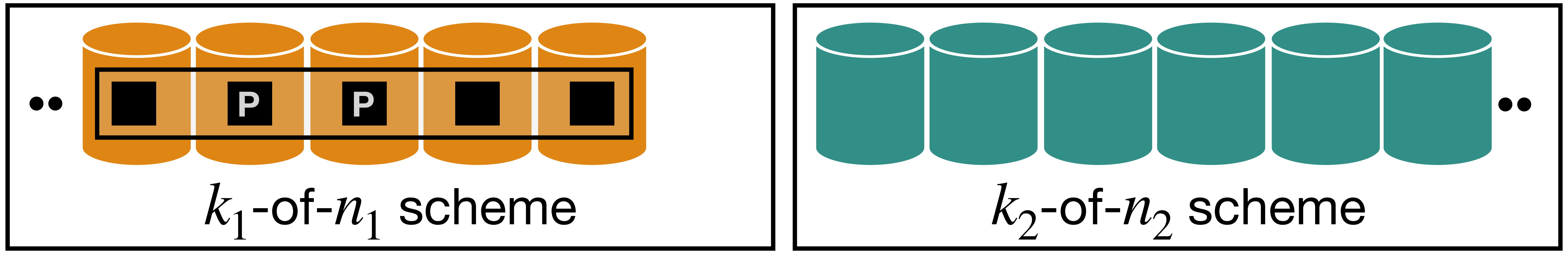



- Need to re-encode (transition) $k_1$-of-$n_1$ to $k_2$-of-$n_2$
- Read rest of the data chunks of stripe ($k_1$ × disk-capacity)
- Write new stripe to new disk-group ($k_1$ × disk-capacity)
- Create new parities
- Delete old parities

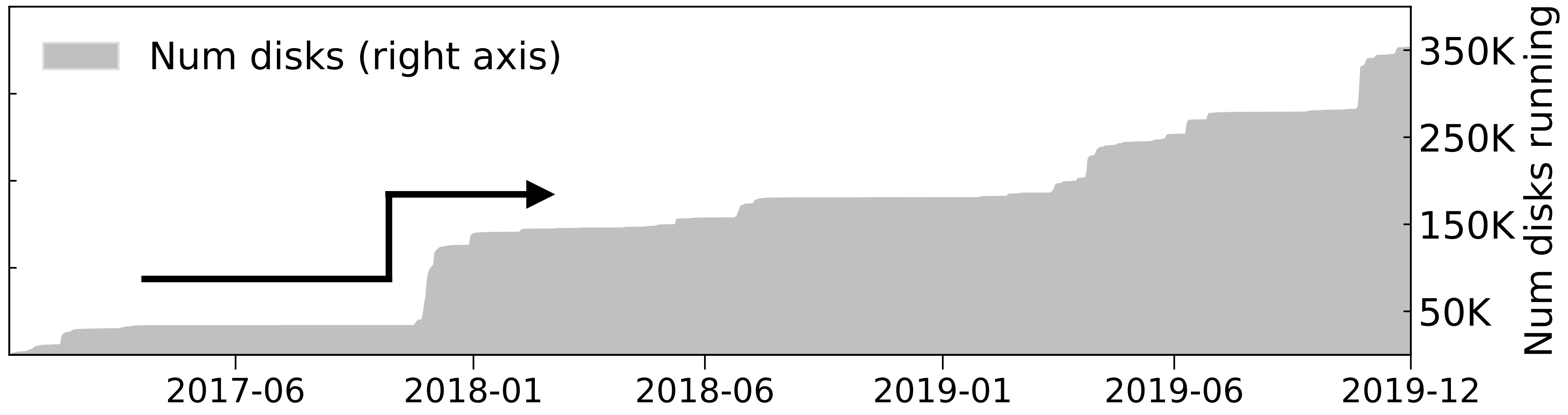
**Disk transition IO > $2 \times k_1 \times$ disk-capacity**

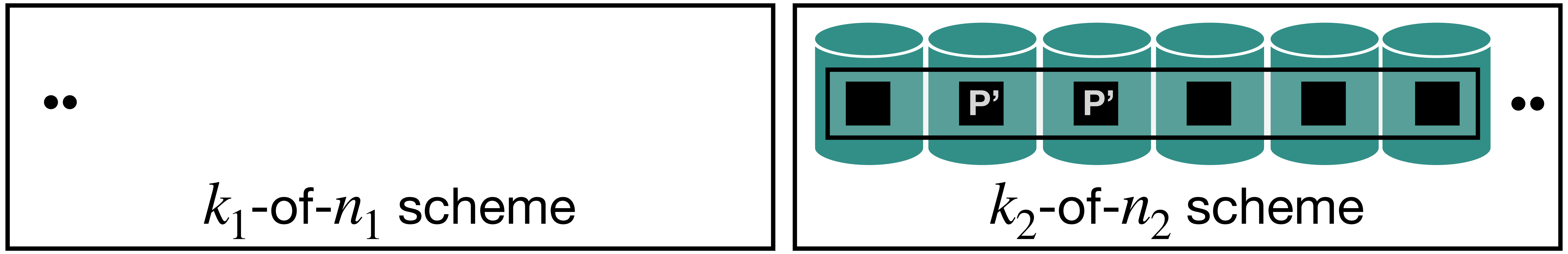
# Transition executor reduces transition IO




Moving data is ($k_1 \times$) cheaper than re-encoding

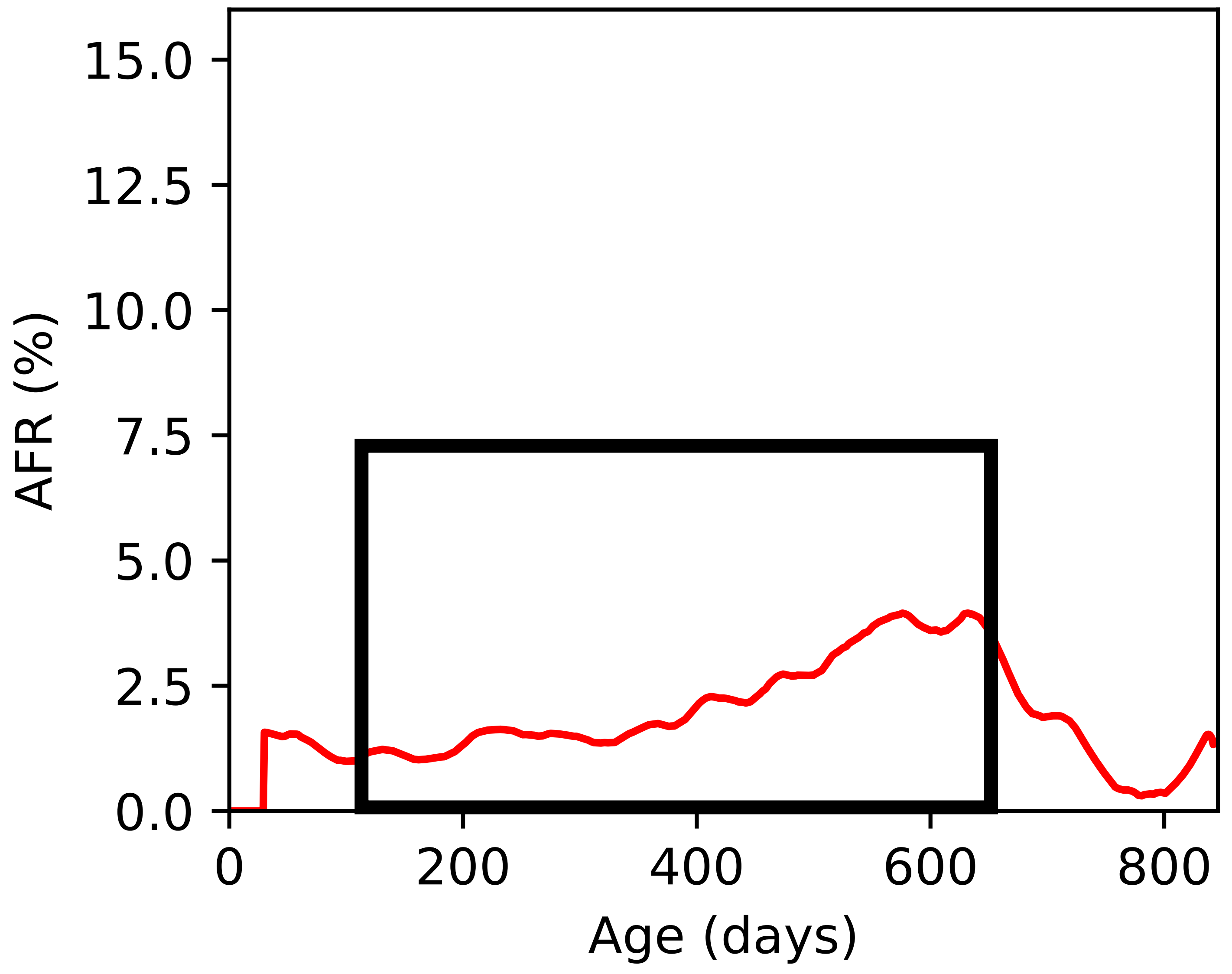


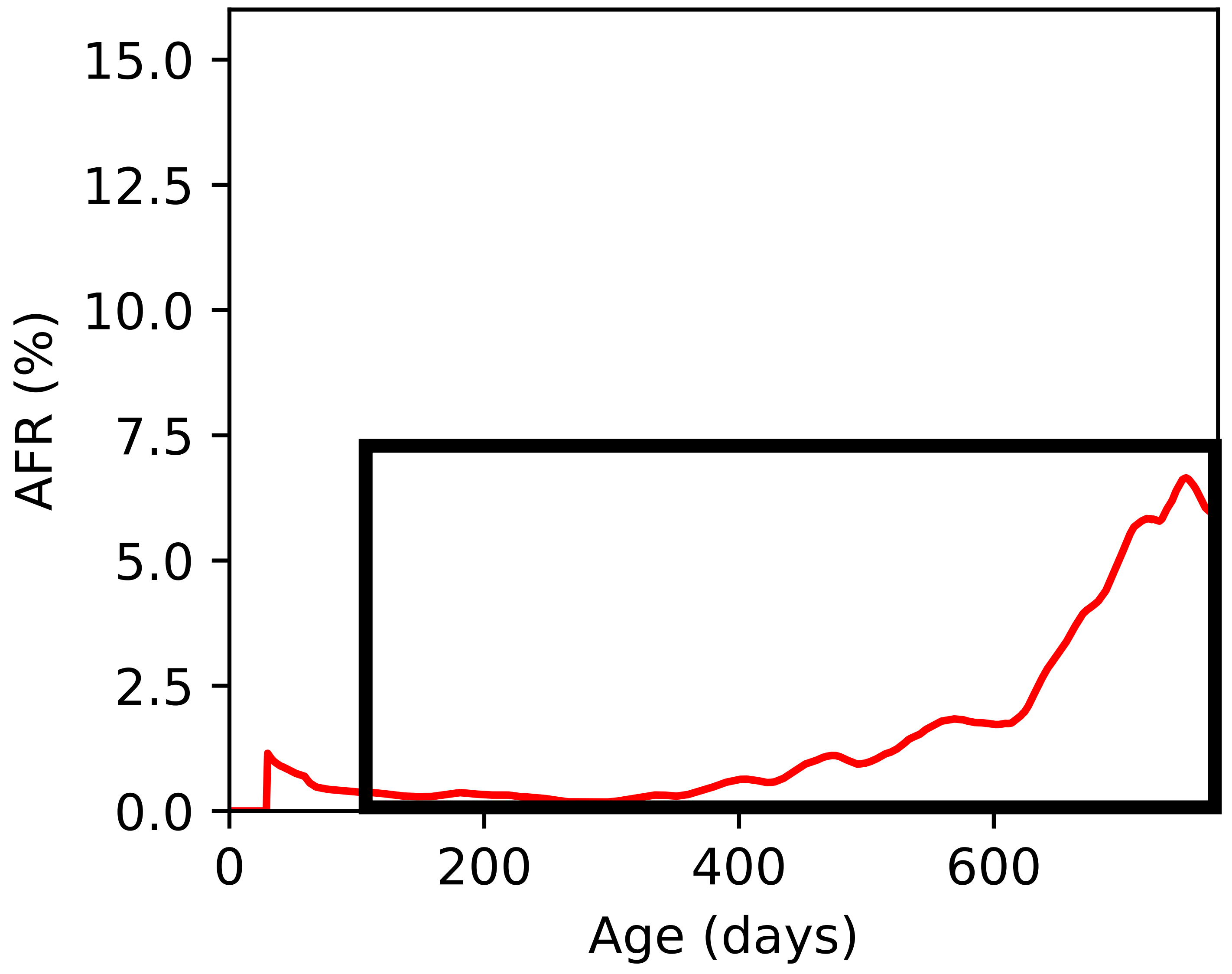




Bulk parity re-calculation is ($n_1 \times$) cheaper than re-encoding

**Transition executor performs deployment specific transitions**

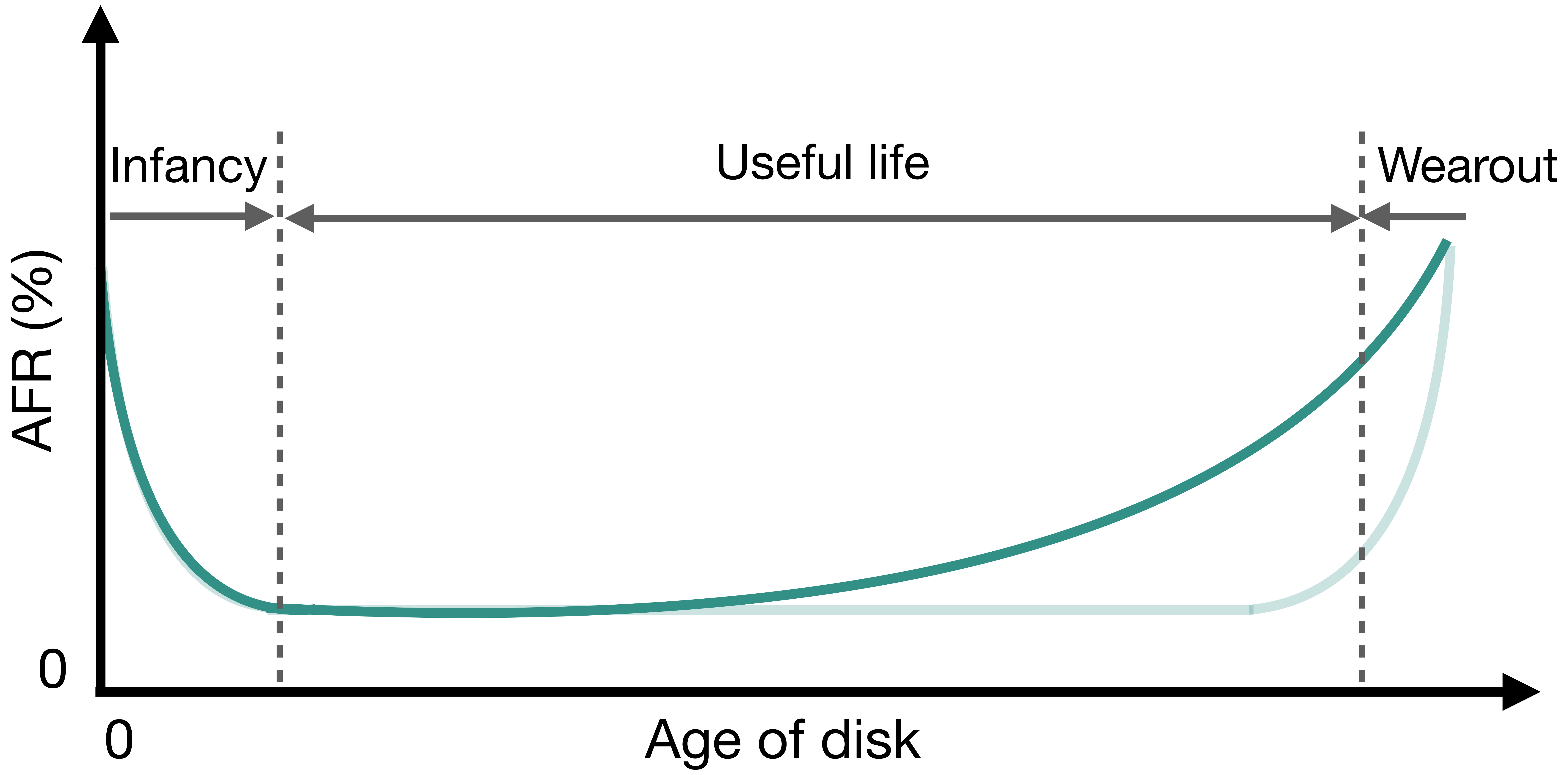
# Practical bathtubs unlike idealized ones



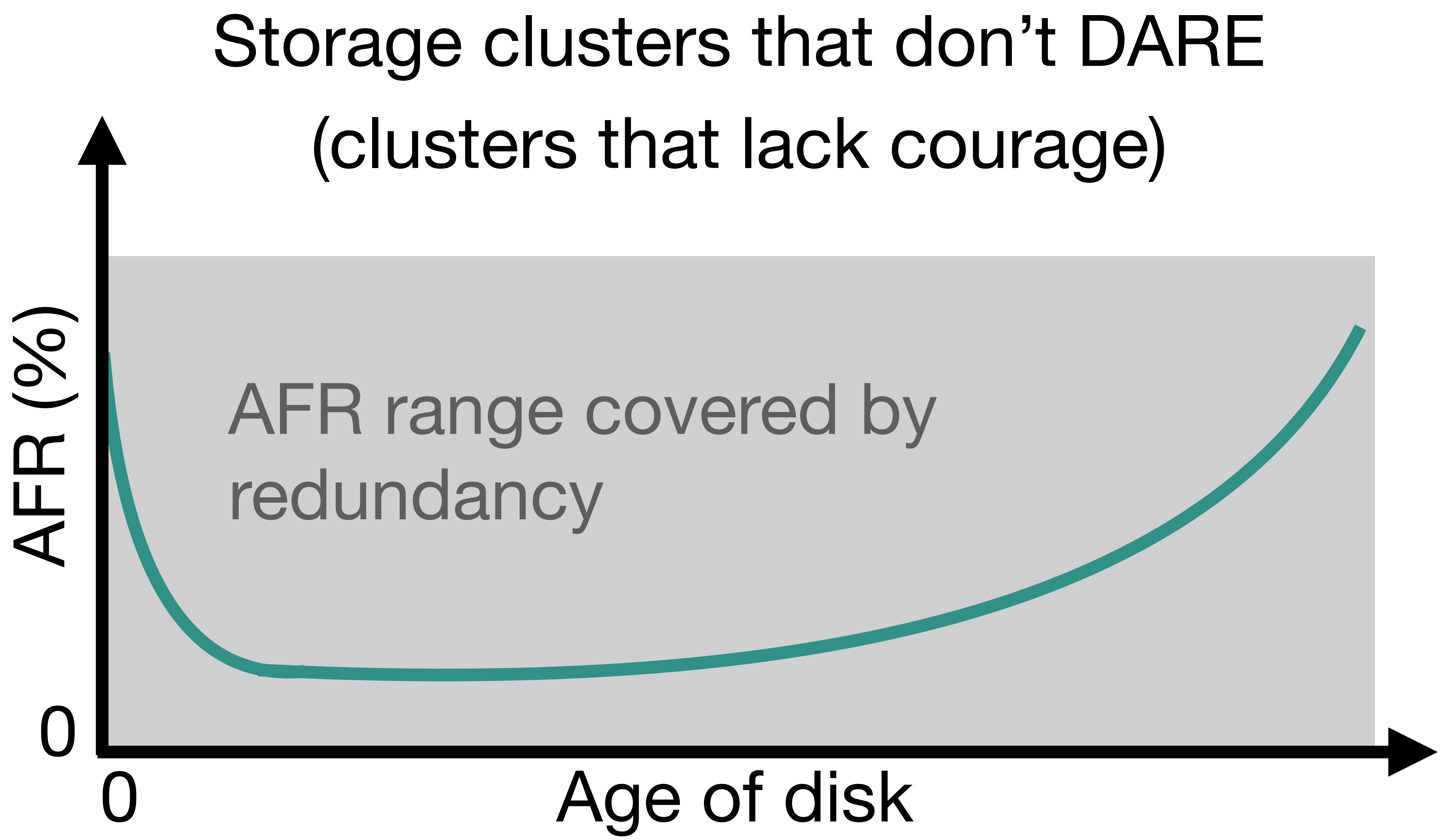



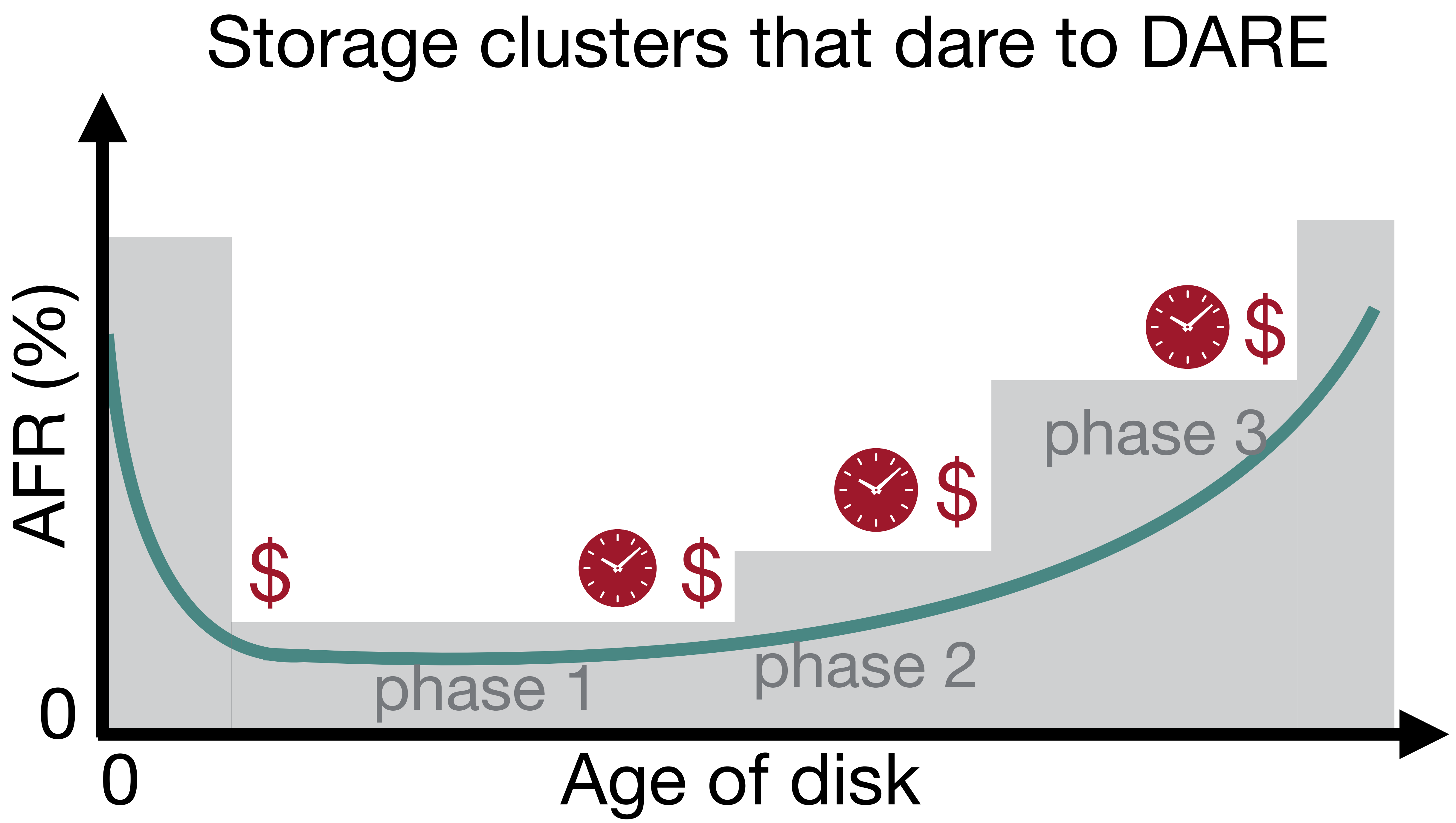

- Useful life isn't flat
    - Gradually increasing useful life curve instead of flat and stable throughout

# Revised idealized disk hazard curve



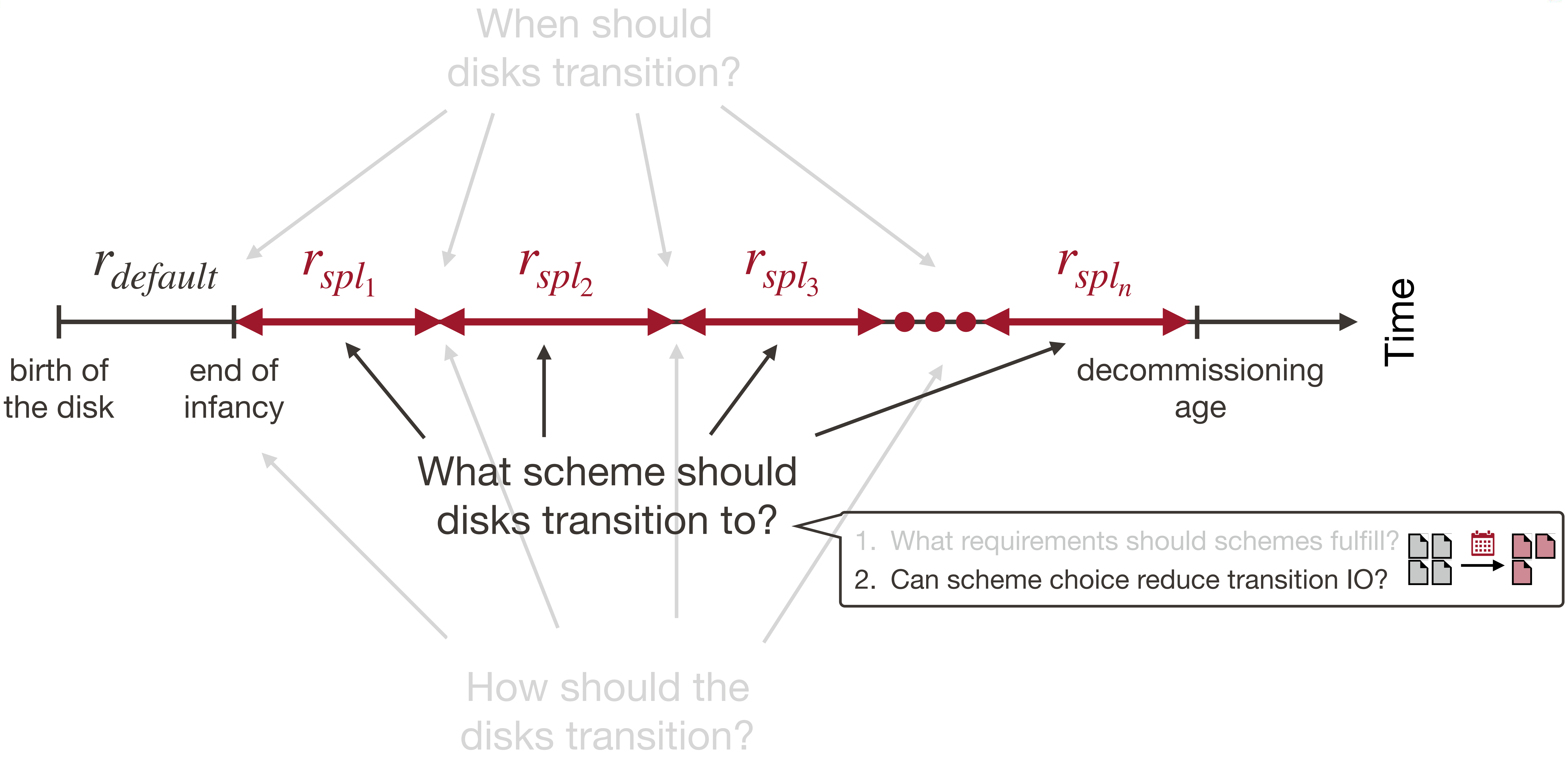
# DARE with multiple useful life phases




**Pacemaker enables multiple redundancy transitions**

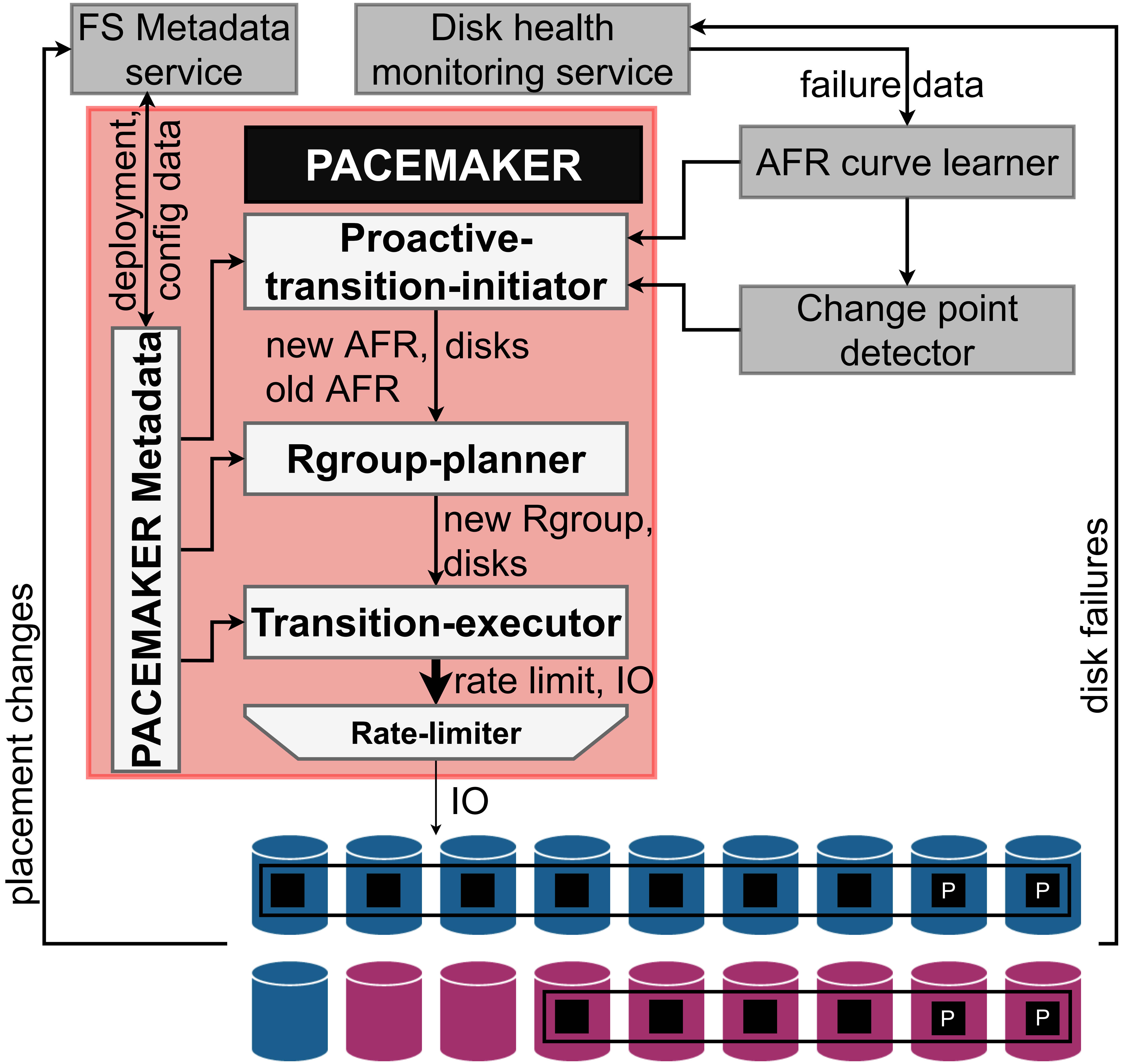
# DAREing disk timeline (what?)



# Redundancy planner adds IO constraints


IO constraints allow IO-friendly redundancy scheme selection

# Architecture of Pacemaker

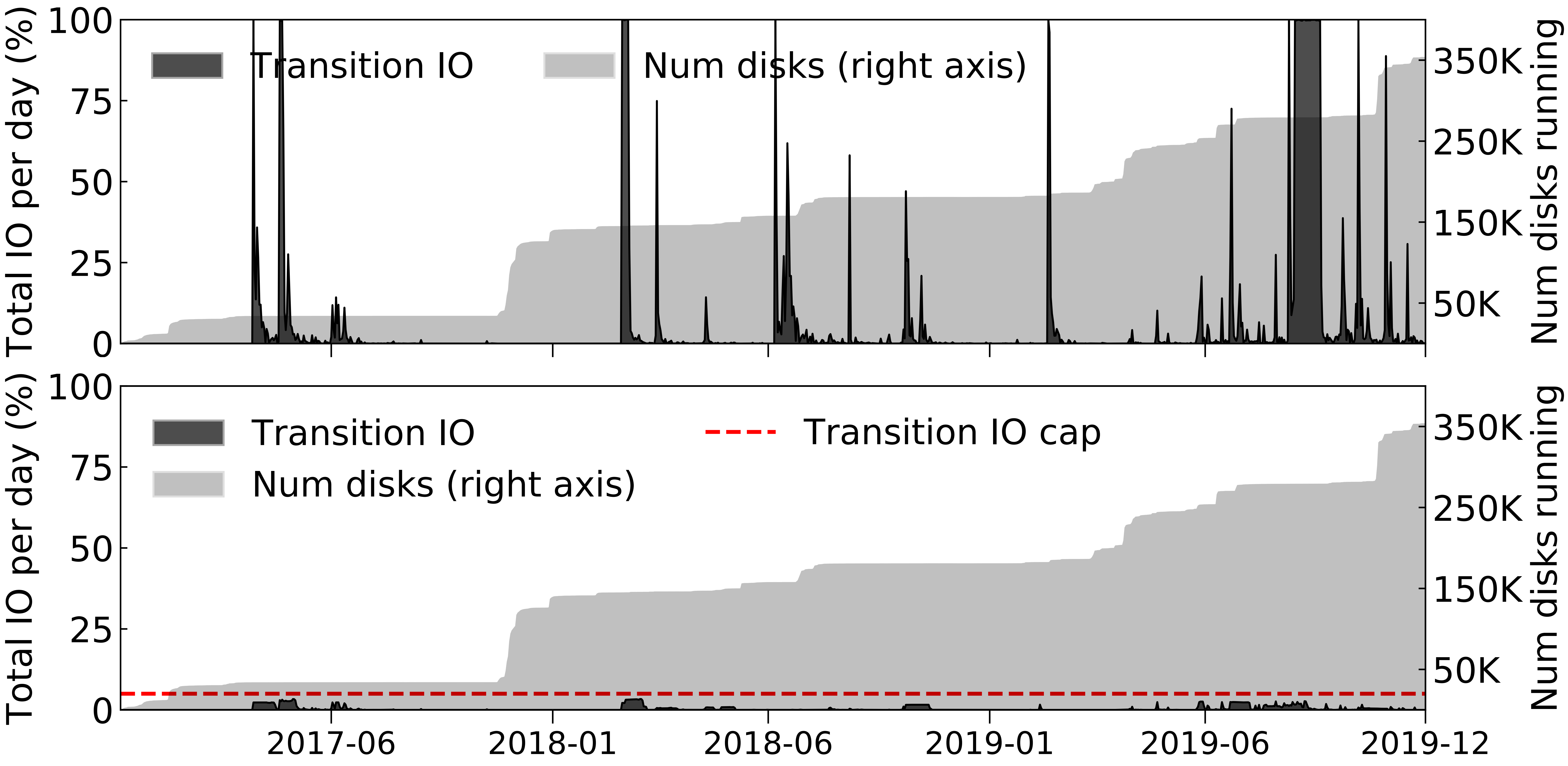


# Production storage cluster reliability traces

- Total disks analyzed:
  - Over **5.3 million** disks
  - **60+ makes/models**
  - **Google, NetApp, Backblaze**
- Daily vitals captured of each disk
- Pacemaker evaluated on four large-scale disaggregated storage clusters:
  - **Google clusters**:
    - Cluster1: 7 makes/models, 350K+ disks, trickle + step
    - Cluster2: 4 makes/models, 450K+ disks, step
    - Cluster3: 3 makes/models, 160K+ disks, step
  - **Backblaze cluster**:
    - 7 makes/models, 120K+ disks, trickle

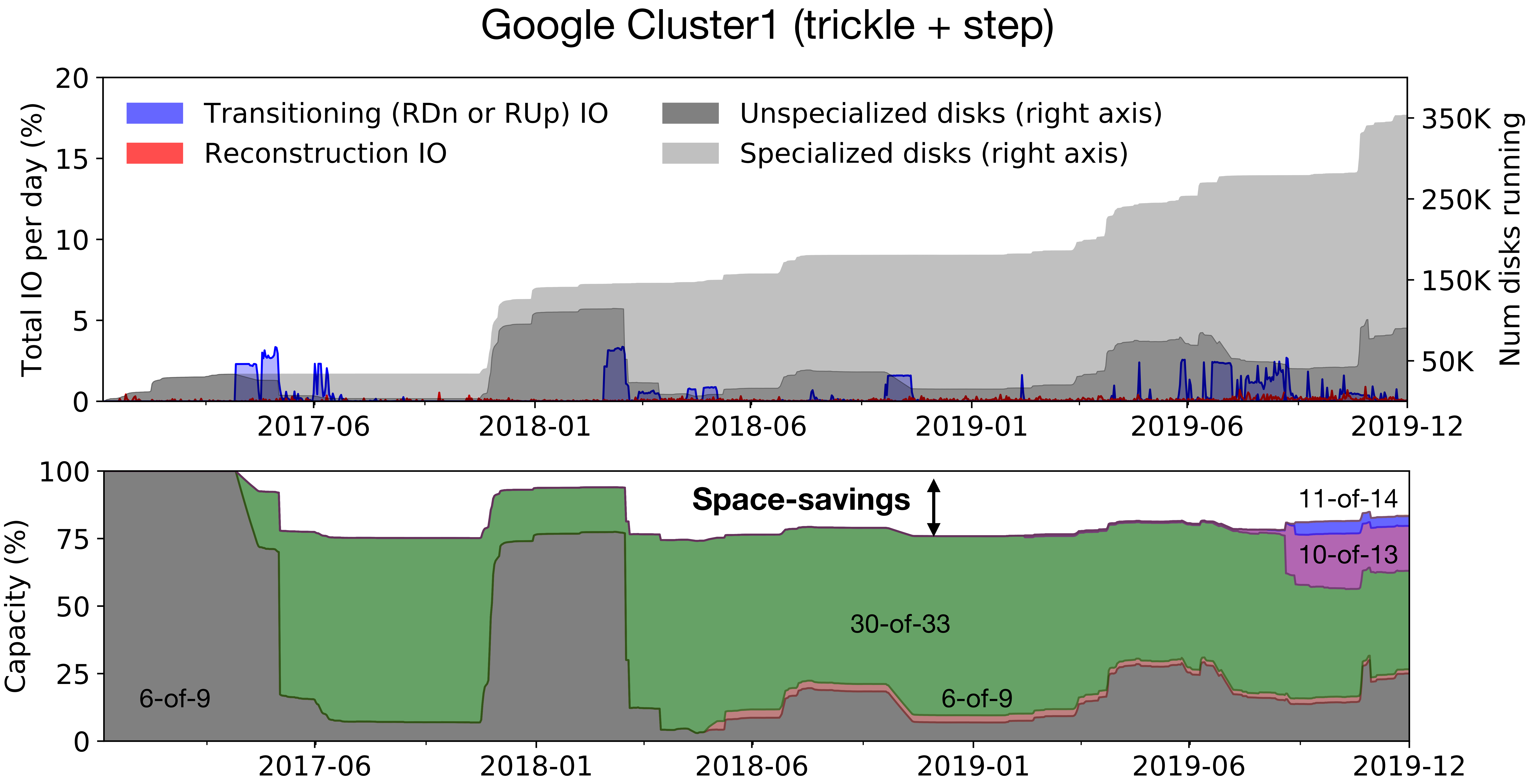
# Enabling a Google Cluster to DARE

DARE




IO reduced by > 90%, Avg. IO = 0.3%, Peak IO < 5%

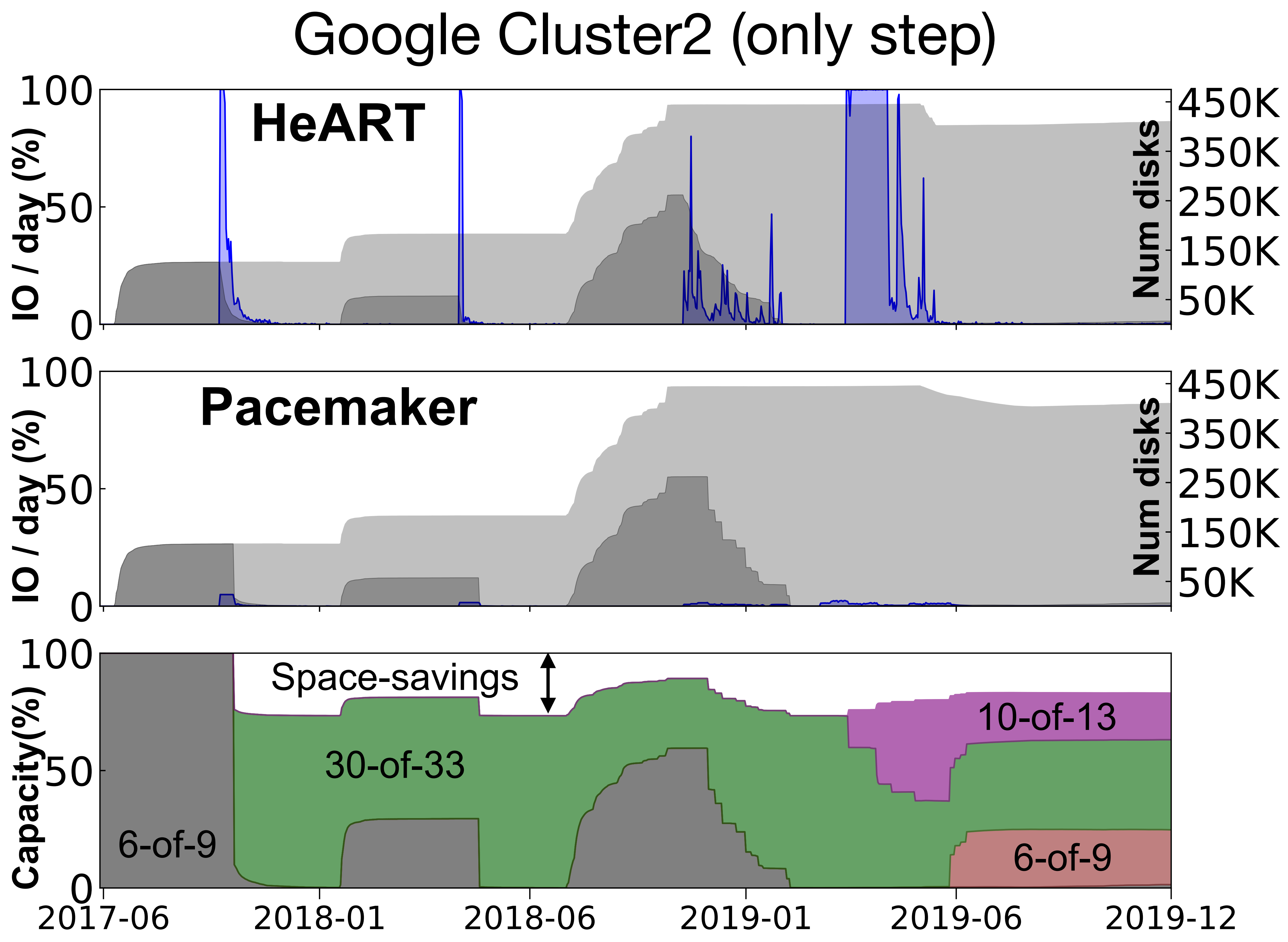
# Space-savings achieved by Pacemaker

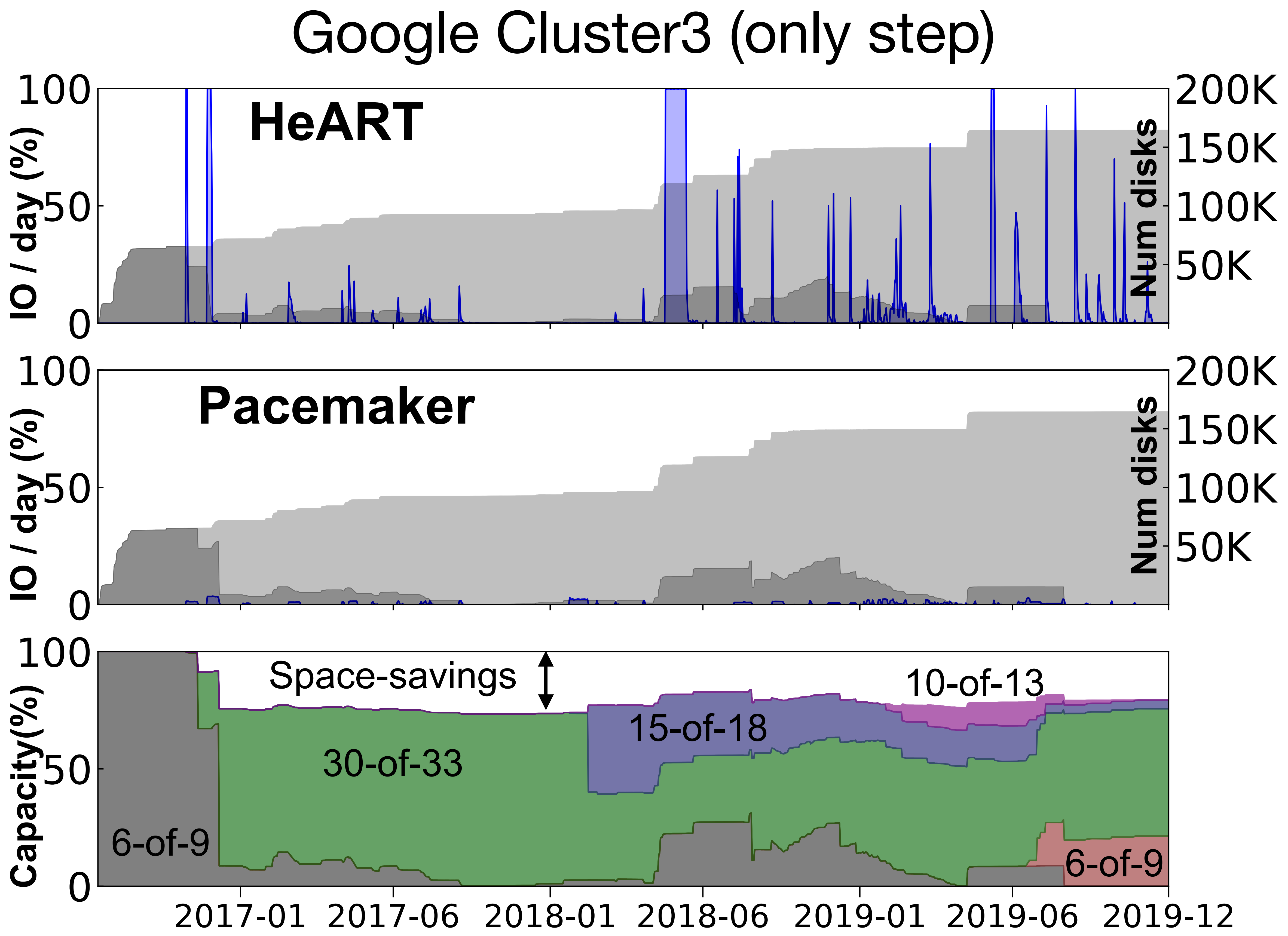



Avg. space-savings = 14%, Peak space-savings = 25%, up to 75000 fewer disks

DARE

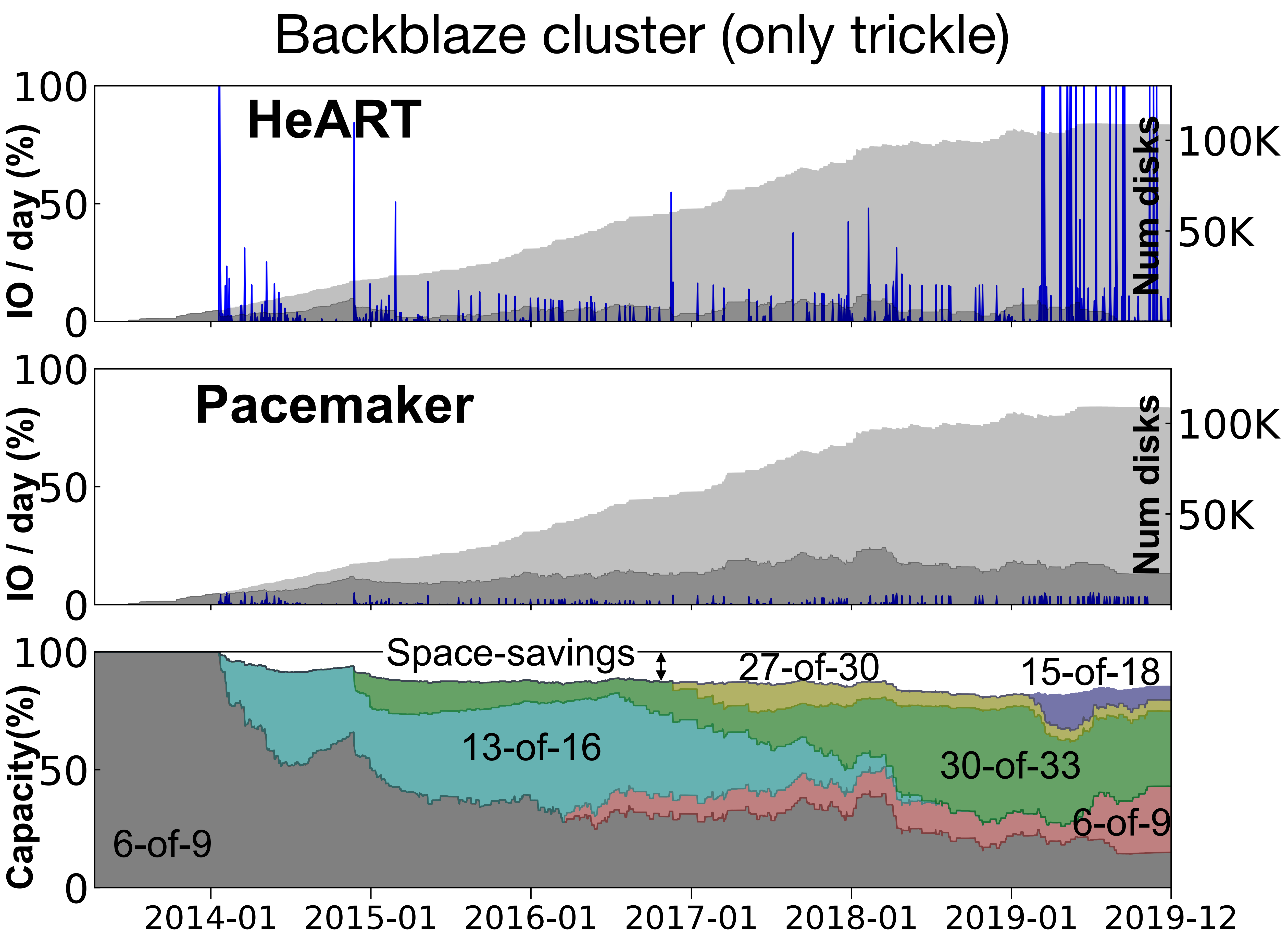
# Other Google clusters



DARE

# Backblaze cluster



# Outline

- Background and motivation
- Disk failure rate heterogeneity
- Making a case for disk-adaptive redundancy (DARE)
- Overcoming transition overload
- Evaluation on real-world cluster traces
- Enabling HDFS to DARE

# Pacemaker in HDFS


Pacemaker incorporated at level of abstraction that leaves files, blocks unaffected

# Conclusion

- **Thesis based on data driven research**
  - 5.3 million disks (HDDs), over 60 makes/models
  - Production environments of Google, NetApp, Backblaze
- **Key insight: high AFR heterogeneity in same storage tier**
  - Over 10x difference in AFRs among disks in the same cluster
- **Invented disk-adaptive redundancy (DARE)**
  - "One-scheme-fits-all" approach mostly overprotects data
  - DARE tailors redundancy to observed AFR for apt redundancy
- **Designed two DARE systems driven by real-world data**
  - Online techniques to tailor redundancy dynamically, yet safely
  - Up to 20% space-savings in clusters with 100–450K disks
- **Built DARE in HDFS as a proof-of-concept**
  - Added DARE at right abstraction (transparent to clients)
  - Reused existing functionality to realize DARE optimizations