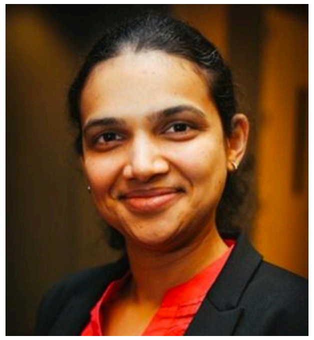
Gotta have HeART

Improving storage efficiency by exploiting disk-reliability heterogeneity

Saurabh Kadekodi

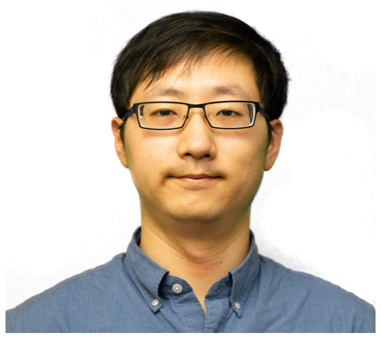
Rashmi Vinayak



Greg Ganger

Francisco Maturana

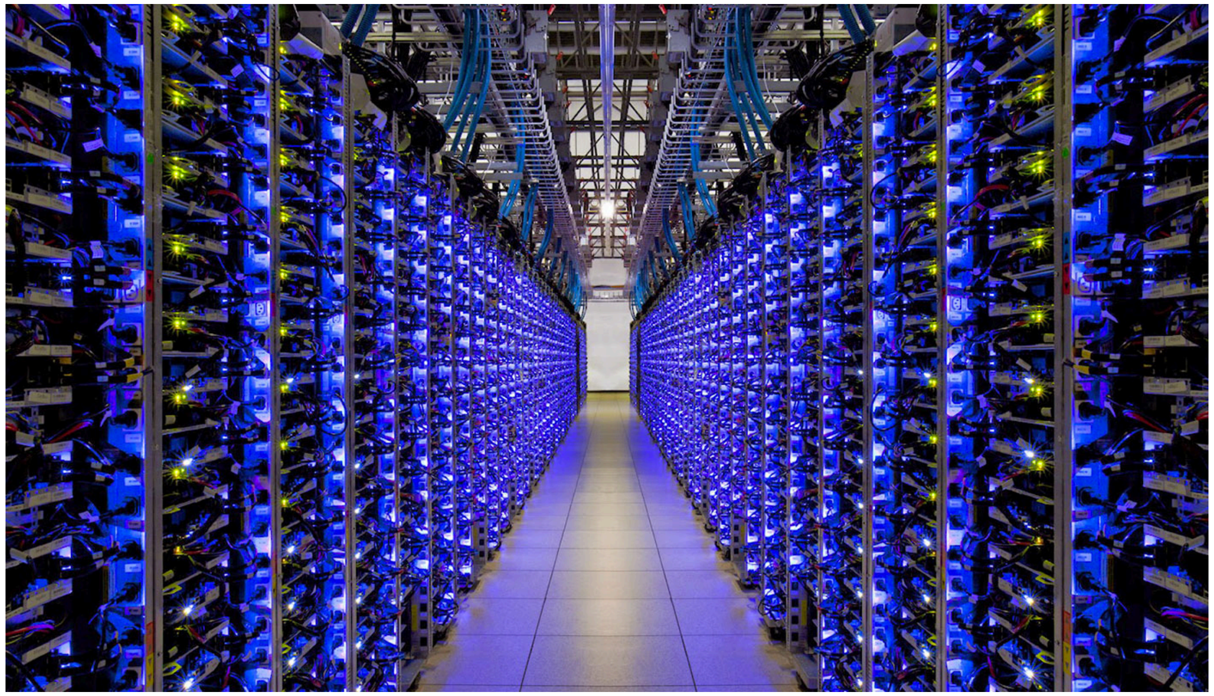
Jason Yang

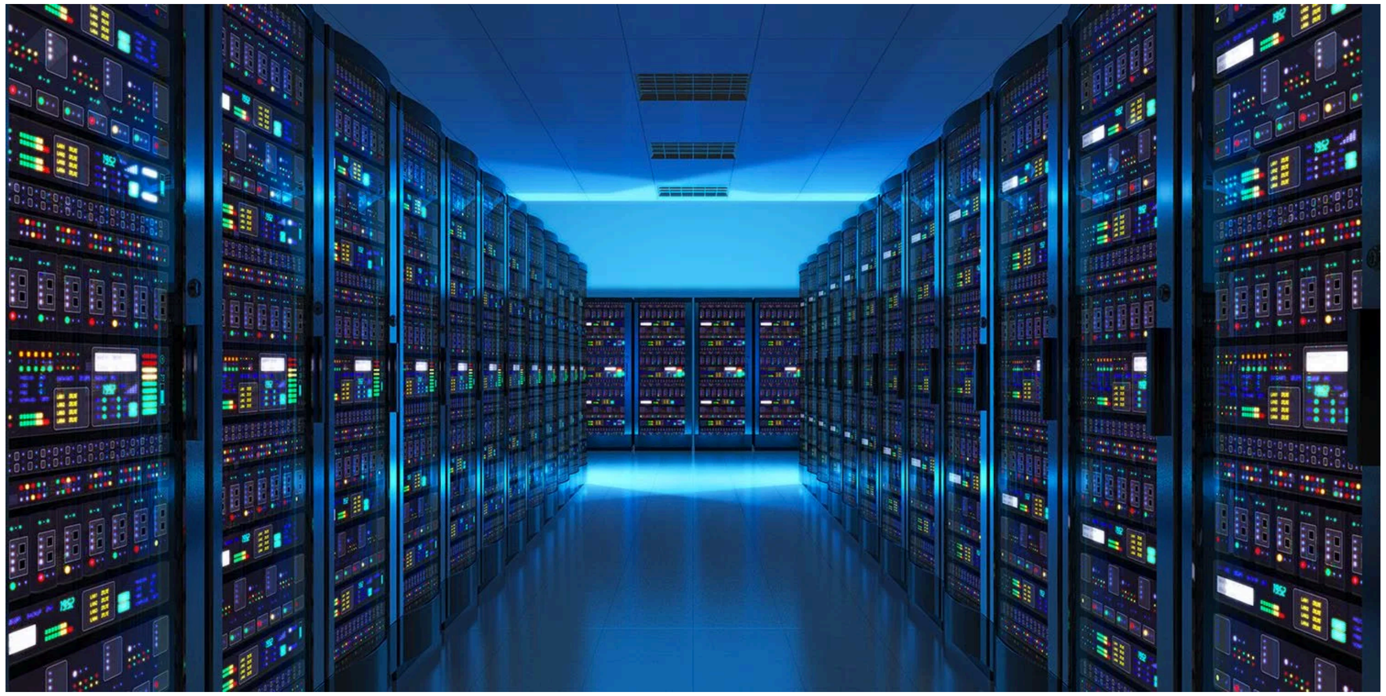


Suhas Jayaram

Cluster storage systems

- Storage subsystem of distributed systems





- Thousands to millions of disks in primary storage tier
- Built incrementally according to demand

Reliability heterogeneity in disks

- Disk fleet has heterogeneous collection of disks
- **Different in reliability**
  - Across disks:
    - Manufacturing differences across makes/models
    - Experiences: different vibration / temperature/ IO churn
  - For each disk:
    - 3 reliability phases throughout lifetime

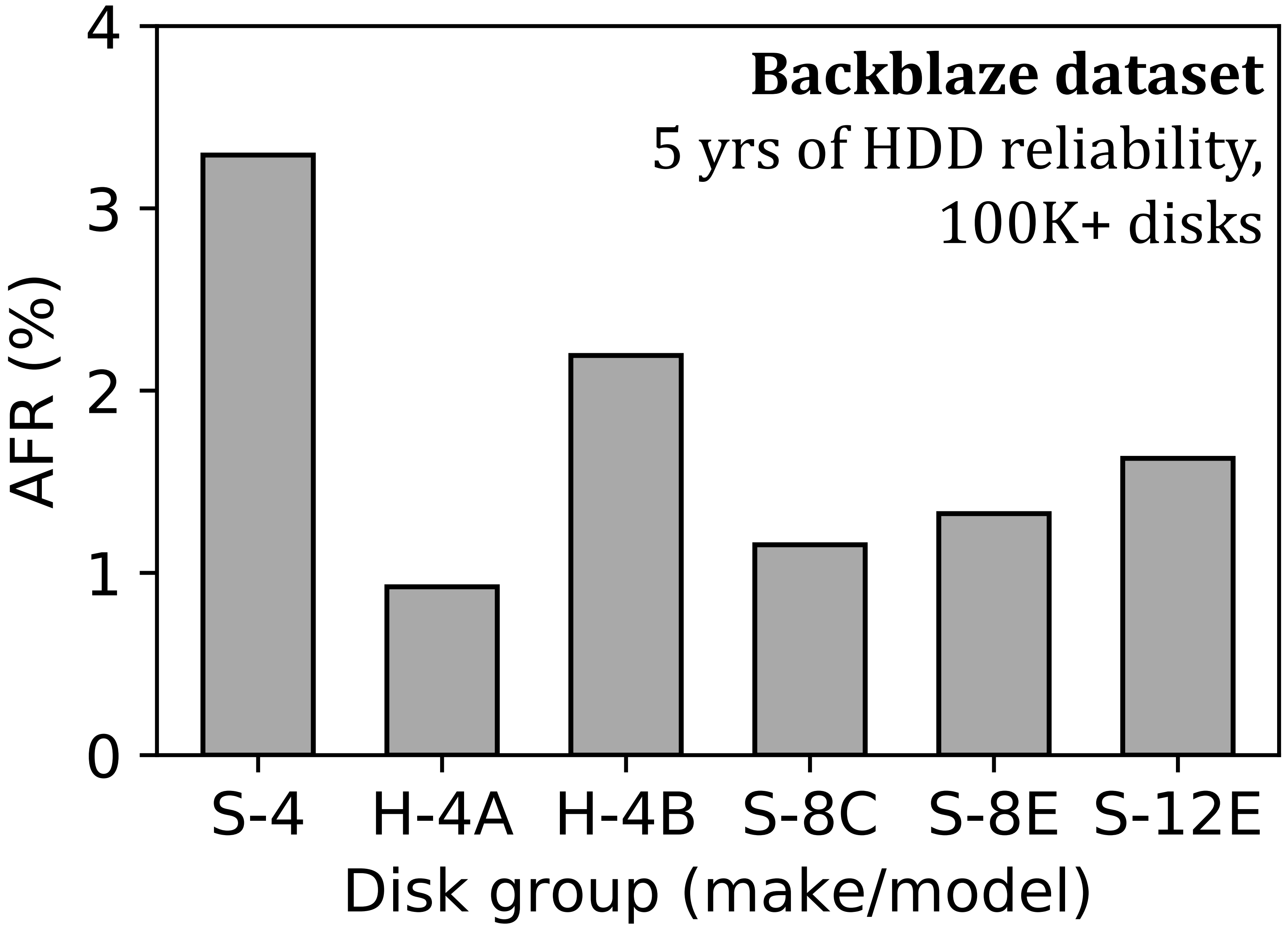
Overview of exploiting reliability heterogeneity

- Data redundancy typically same across disk fleet
  - E.g., 3-replication: 3 copies of data on independent devices
- Disks from same storage tier vary a lot in failure rates
  - E.g., HDDs from different makes/models fail differently
- **Explicitly consider reliability heterogeneity in deciding redundancy**
- **HeART**: **He**terogeneity **A**ware **R**edundancy **T**uner
  - Tailors redundancy to disk failure rate heterogeneity
  - A safe, accurate and online framework
  - Reduces storage overhead, and thus cost
- **Pacemaker**: regulating the HeART
  - Manages redundancy management overheads
  - Perform cheap re-encoding
  - Converts urgent re-encoding tasks into schedulable tasks
- **HeART + Pacemaker reduces overall storage space by > 20% [m(b)illions $]**

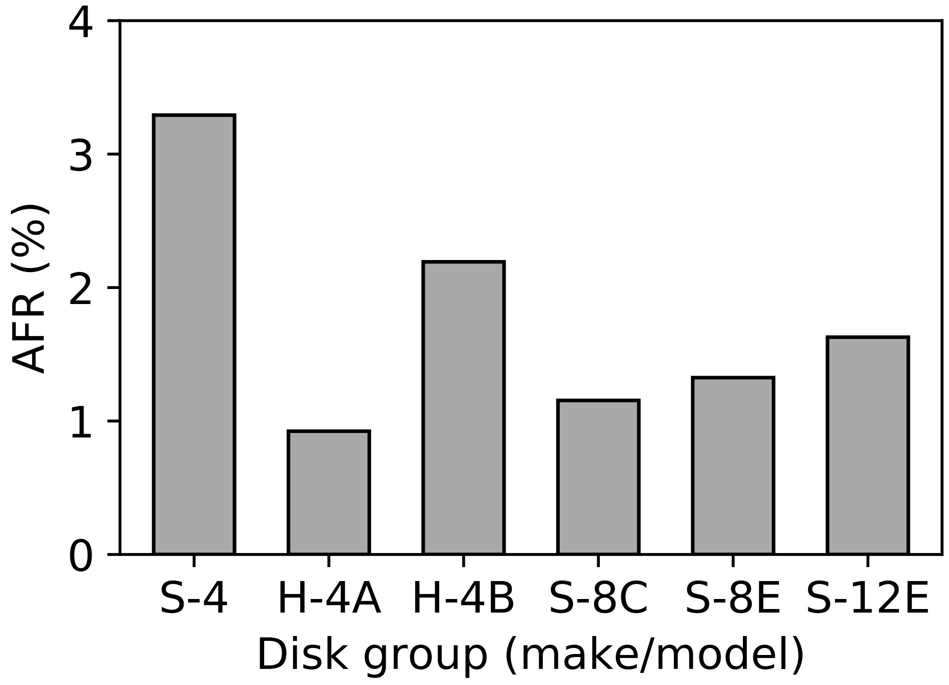
Cluster storage system reliability

- Failures common in today's cluster storage systems
  - Disk failures measured as annualized failure rates (AFR)
  - AFR → expected % of disk failures in a year
- Popular fault tolerance mechanism → redundancy
  - Full data replication (n-replication)
  - Erasure coding (**k-of-n**: k data chunks, n-k parity chunks)
- Reliability measured in mean-time-to-data-loss (MTTDL)
- Redundancy configurations ignore disk AFR differences

Reliability heterogeneity



Reliability heterogeneity

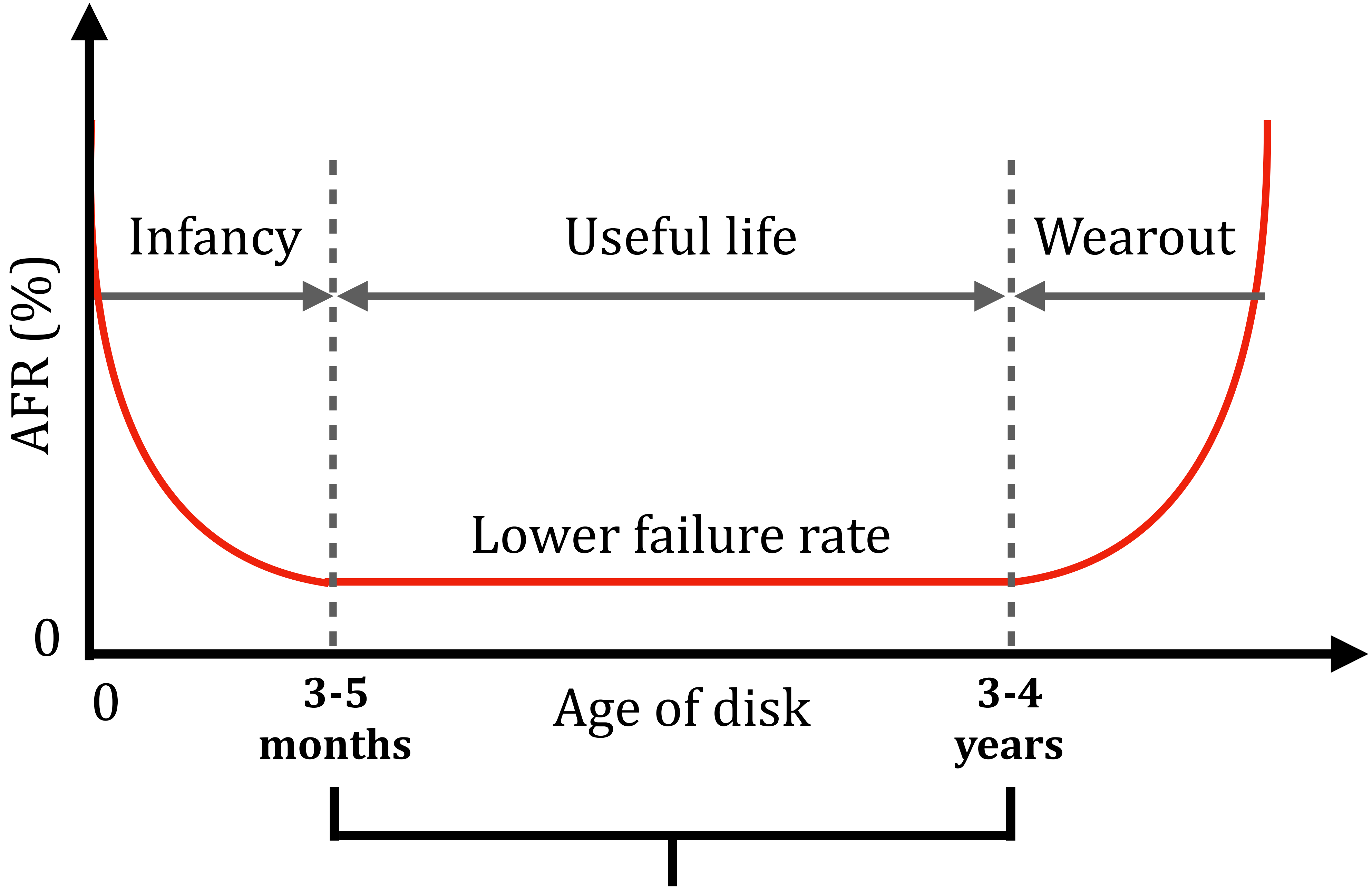


- HDD failure rates vary a lot in the field
- No single redundancy scheme is good enough for all disks
  - Conservative redundancy → overprotection for strong disk types
  - Lower redundancy → subset of disks risk data loss

Exploiting reliability heterogeneity

- Redundancy decisions informed by AFR differences
- **Challenges**
  1. Has to be **monitored in the field**
  2. Disk failure rate **varies over its lifetime**
- Redundancy tailoring mechanism needs to be:
  - **Safe**: prevent under-redundancy from causing data loss
  - **Accurate**: identify different reliability phases correctly
  - **Online**: benefits only realizable during disk's low failure rate

The bathtub curve (each disk group)

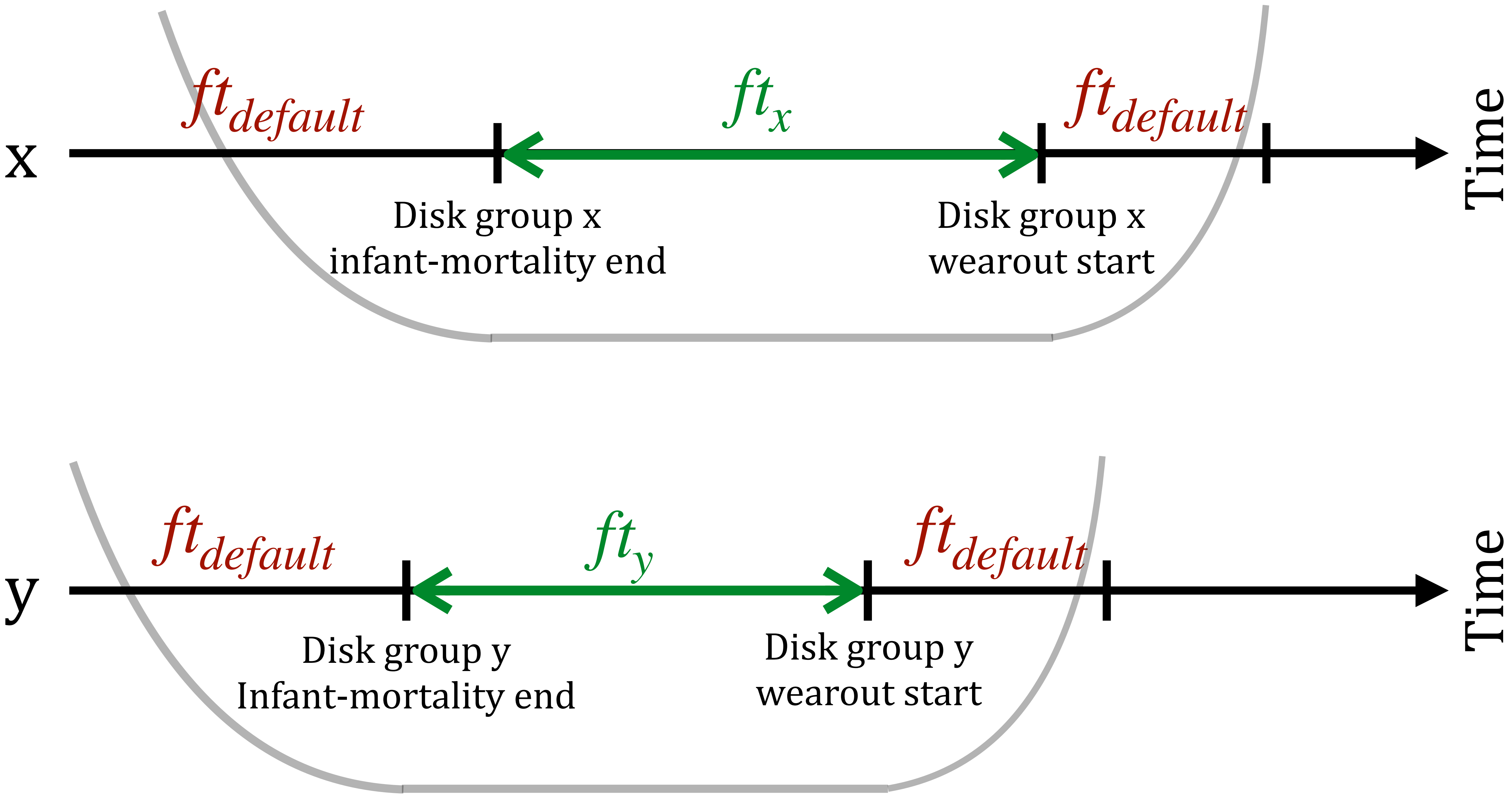


lower AFR → **lower redundancy** → **lower storage cost**

Two disk groups over time

Deployment
(start monitoring)

$ft_{default}$ = default fault tolerance scheme



x

$ft_{default}$

ft_x

$ft_{default}$

Time

Disk group x
infant-mortality end

Disk group x
wearout start

y

$ft_{default}$

ft_y

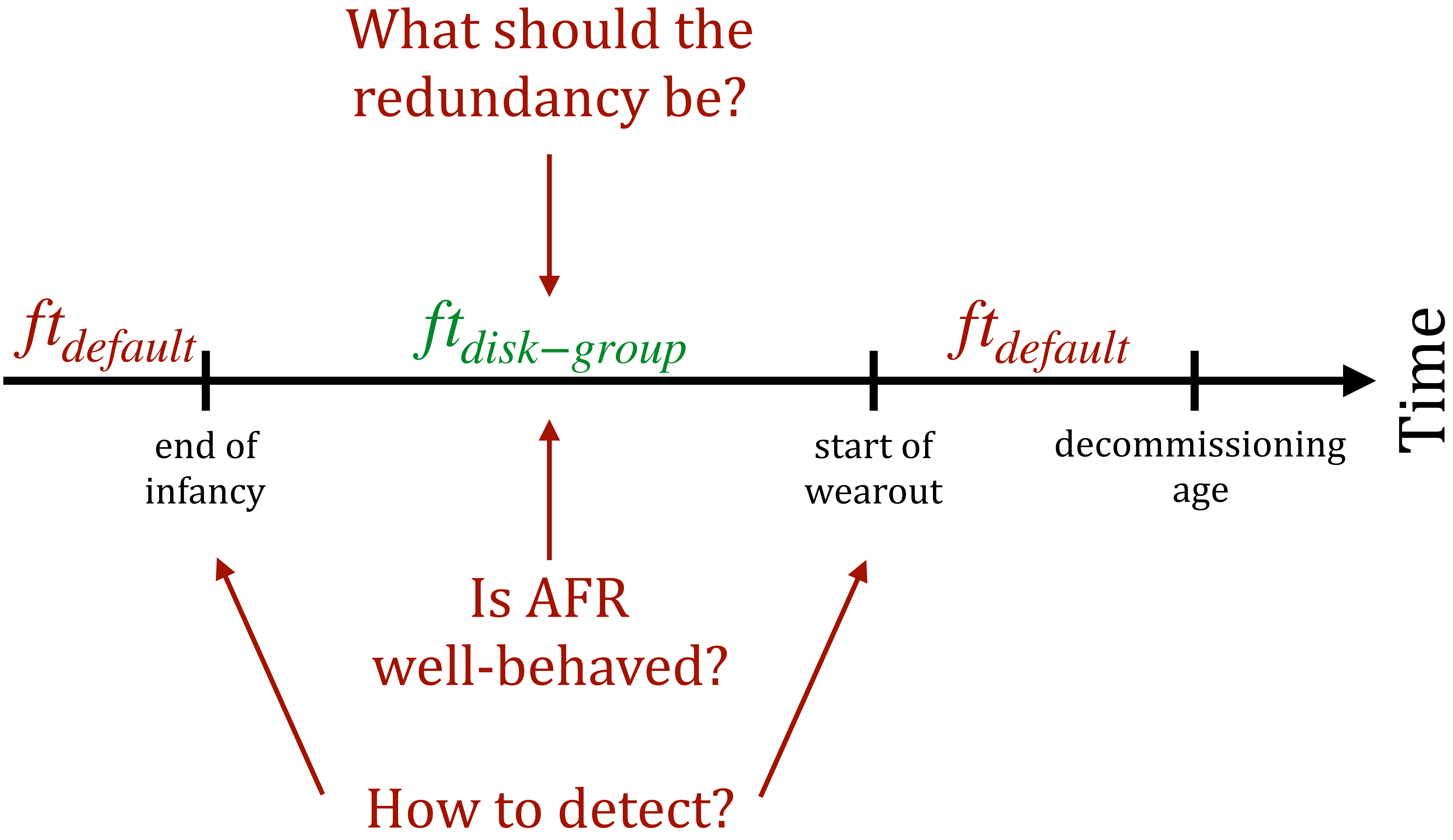
$ft_{default}$

Time

Disk group y
Infant-mortality end

Disk group y
wearout start

Disk-group reliability timeline



What should the redundancy be?

$ft_{default}$

$ft_{disk-group}$

$ft_{default}$

Time

end of infancy

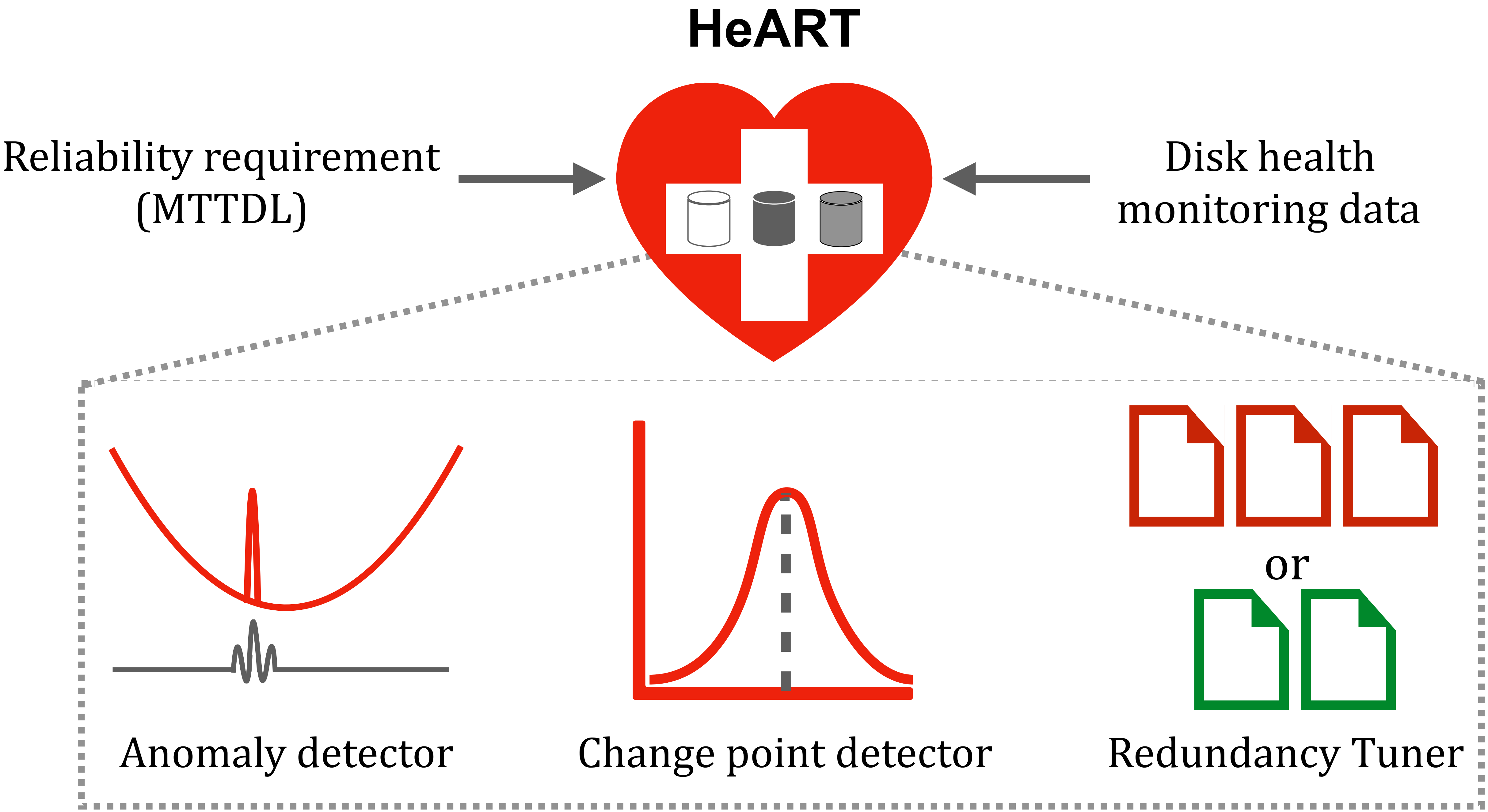
start of wearout

decommissioning age

Is AFR well-behaved?

How to detect?

Heteretogeneity-Aware Redundancy Tuner



HeART

Reliability requirement (MTTDL)

Disk health monitoring data

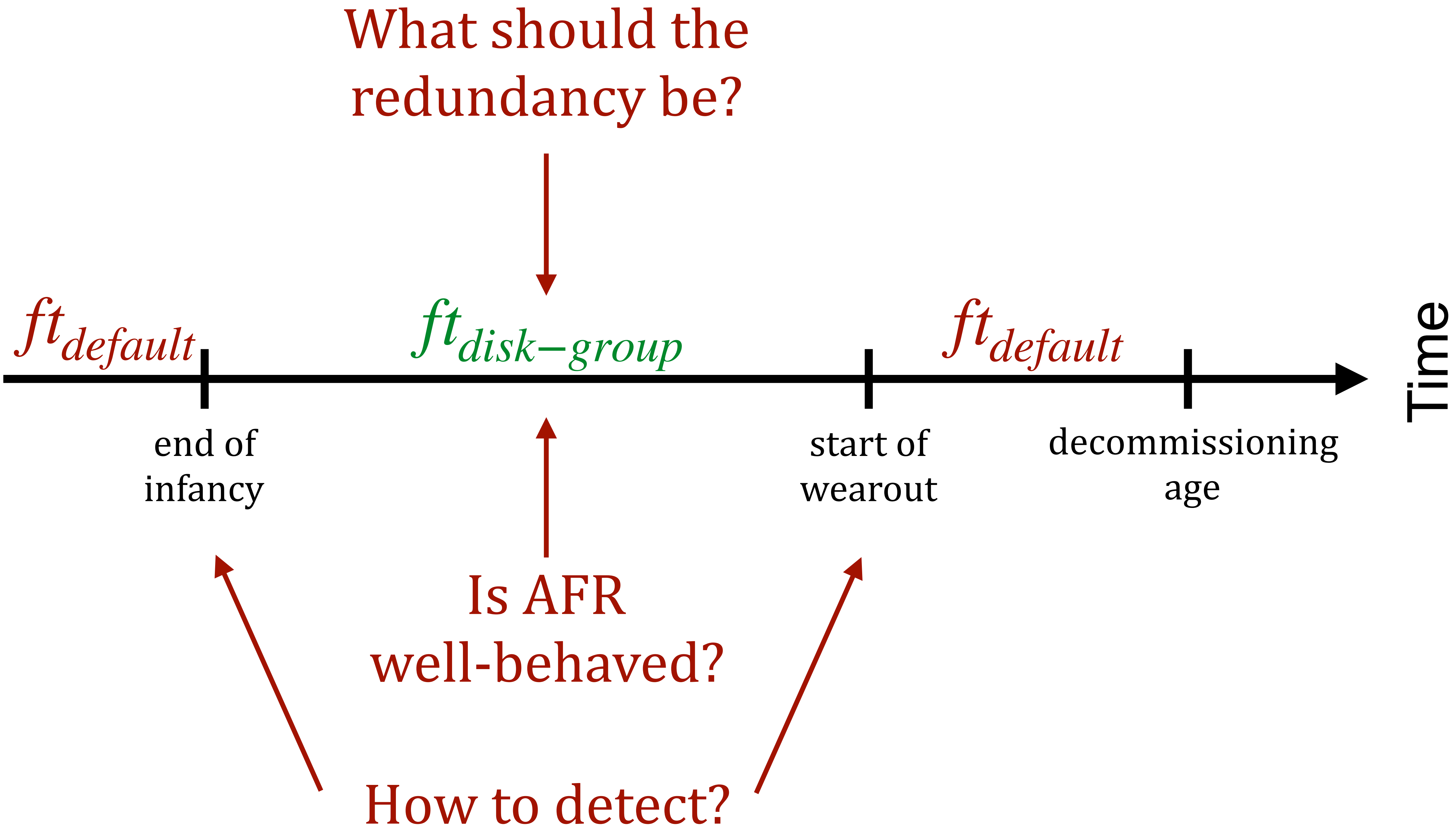
Anomaly detector

Change point detector

or

Redundancy Tuner

Disk-group reliability timeline



What should the redundancy be?

$ft_{default}$ $ft_{disk-group}$ $ft_{default}$

Time

end of infancy

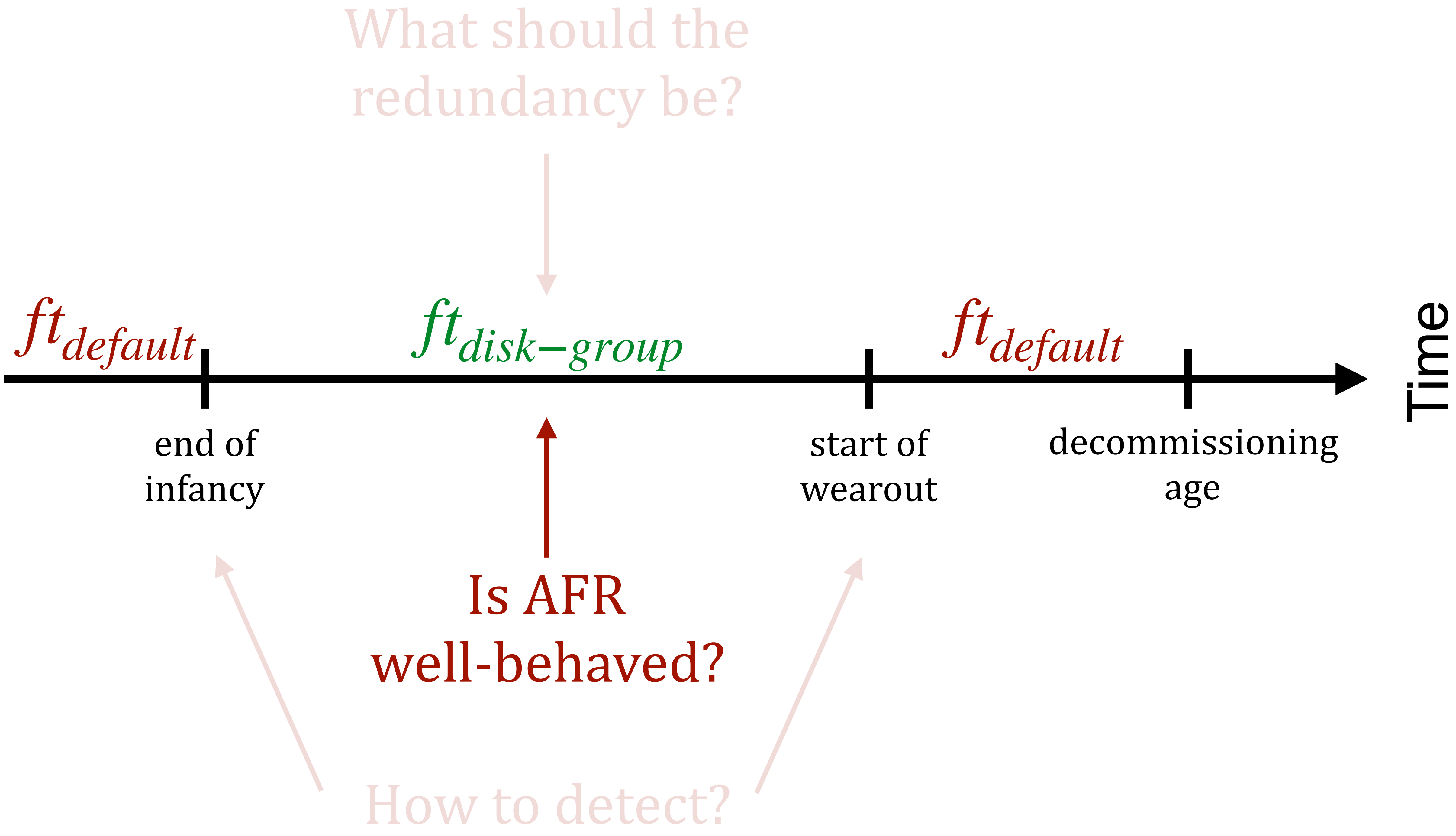
start of wearout

decommissioning age

Is AFR well-behaved?

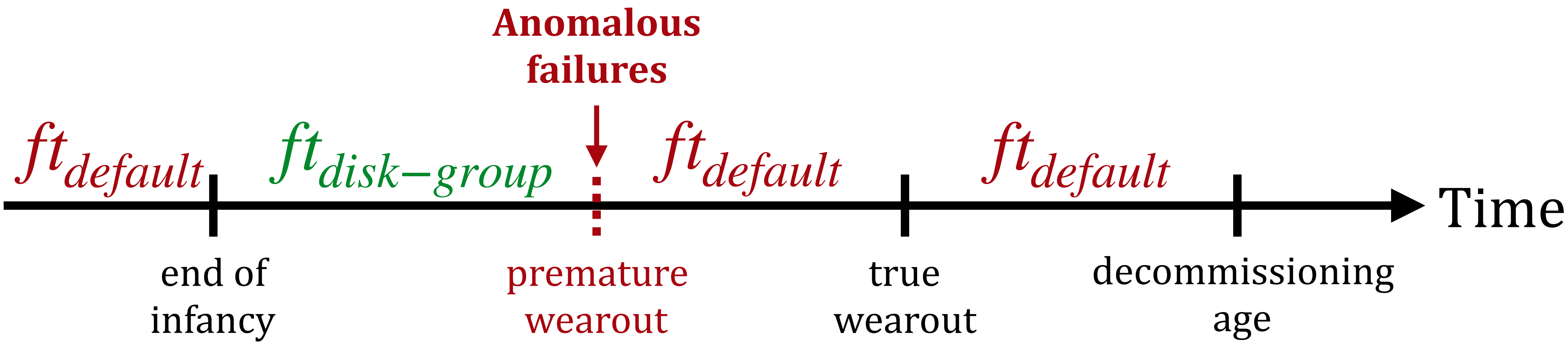
How to detect?

Disk-group reliability timeline

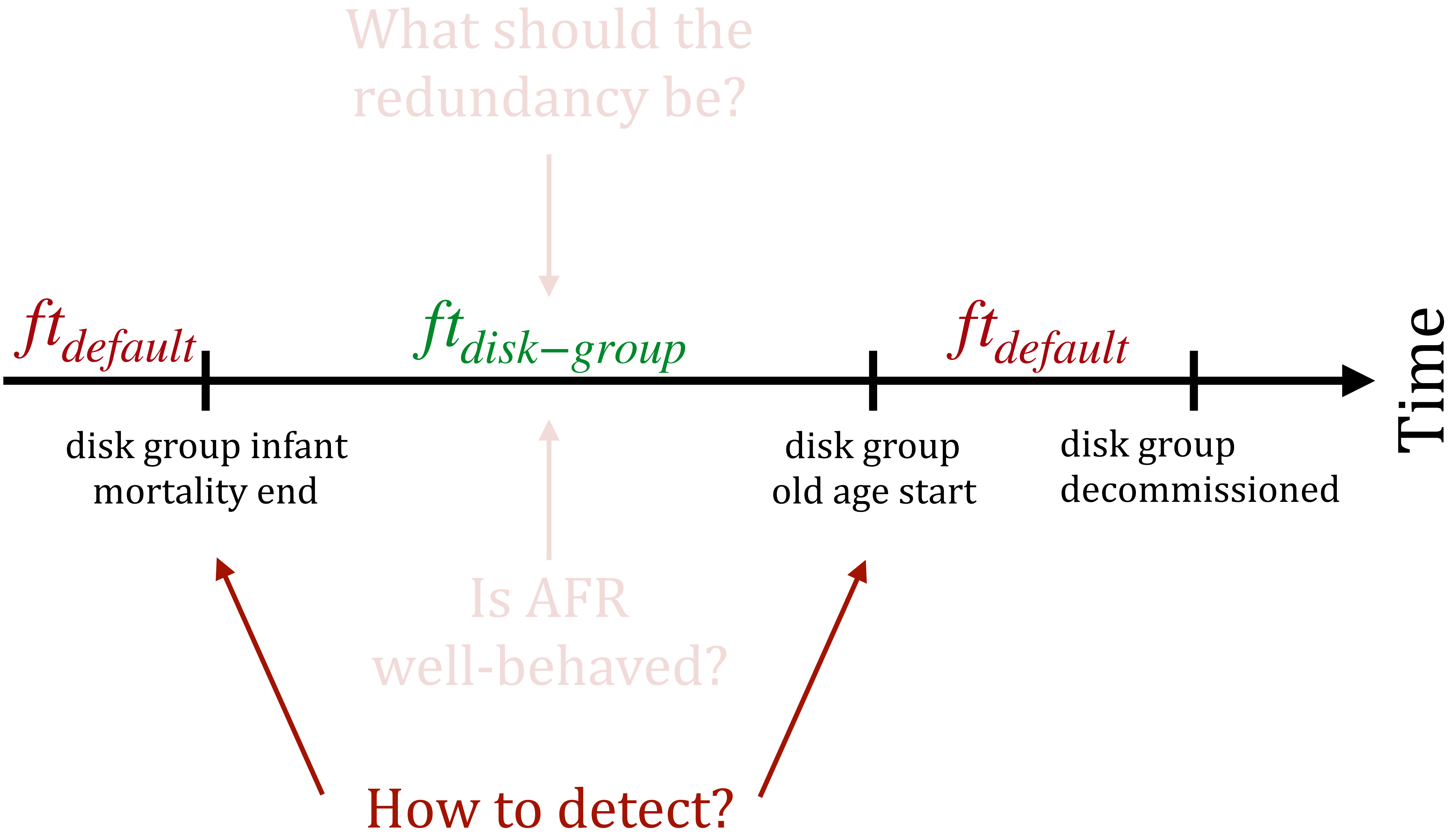


AFR in useful life: stability & anomalies

- Useful life AFR is typically stable, within reasonable bounds
- External factors can cause simultaneous bulk failures
  - Rack power failure, accidents, human error, etc.
- "Anomalies" appear like (premature) wearout
  - Benefits proportional to length of useful life
  - Bulk failures may not reflect true HDD failure rate



Disk-group reliability timeline



What should the redundancy be?

$ft_{default}$ $ft_{disk-group}$ $ft_{default}$

disk group infant mortality end

disk group old age start

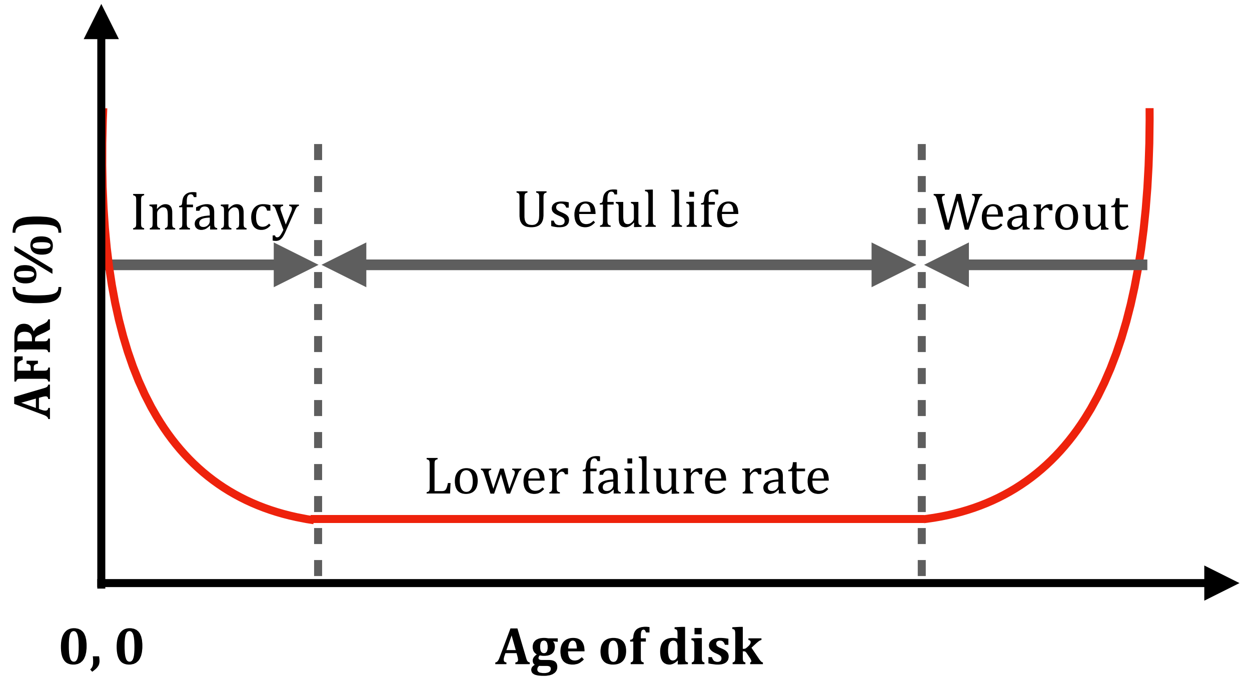
disk group decommissioned

Time

Is AFR well-behaved?

How to detect?

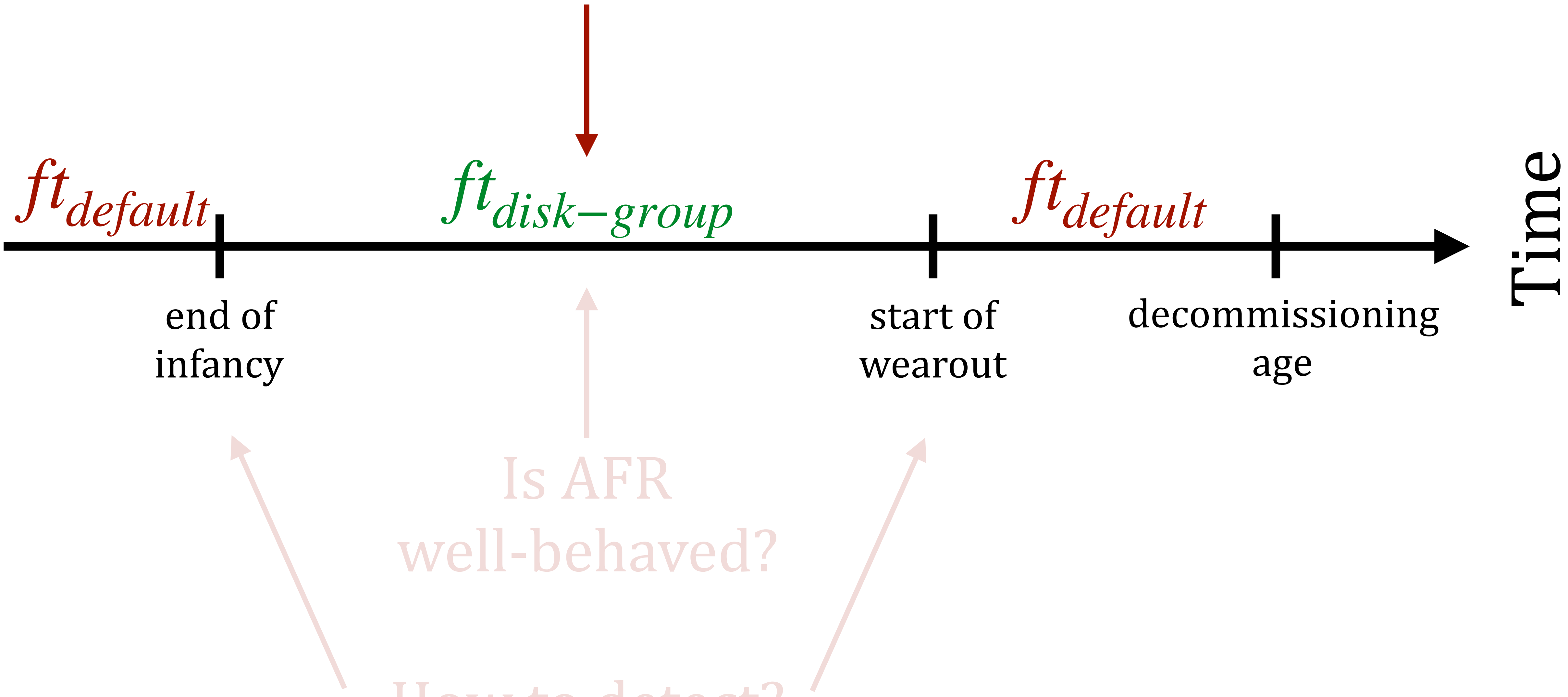
Change point detection



- Reliability target can be missed if:
  - Hasty declaration of end of infancy
  - Delayed declaration of onset of wearout
- Tradeoff between extracting benefits and safety
- Use online change point detectors to identify change points

Disk-group reliability timeline

What should the redundancy be?



Is AFR well-behaved?

How to detect?

Redundancy scheme selection

- $ft_{disk-group}$ MTTDL >= $ft_{default}$ MTTDL (default AFR = 16%)
  - MTTDL: mean time to irrecoverable data loss

Redundancy scheme selection

- $ft_{disk-group}$ MTTDL >= ft = 16%)
  - MT e to irrecoverable data loss

Target reliability constraint

Redundancy scheme selection

- $ft_{disk-group}$ MTTDL >= ft = 16%)
  - MT to irrecoverable data loss

Target reliability constraint

- Failures tolerated in $ft_{disk-group}$ >= failures tolerated in $ft_{default}$

Redundancy scheme selection

- $ft_{disk-group}$ MTTDL >= ft = 16%)

  - MT to irrecoverable data loss

Target reliability constraint

- Failures t $ft_{disk-group}$ ailures tolerated in $ft_{default}$

Min num failures constraint

Redundancy scheme selection

- $ft_{disk-group}$ MTTDL >= ft = 16%)
    - MT irrecoverable data loss

Target reliability constraint

- Failures $ft_{disk-group}$ failures tolerated in $ft_{default}$

Min num failures constraint

- $ft_{disk-group}$ dimension <= max dimension (max k = 30)

Redundancy scheme selection

- $ft_{disk-group}$ MTTDL >= ft = 16%)
    - MT to irrecoverable data loss
- Failures t $_{disk-group}$ failures tolerated in $ft_{default}$
- ft_{disk-g} (max k = 30)

Target reliability constraint

Min num failures constraint

Max code width constraint

Redundancy scheme selection

- $ft_{disk-group}$ MTTDL >= ft … = 16%)
  - MT… to irrecoverable data loss

Target reliability constraint

- Failures t… $_{disk-group}$ … failures tolerated in $ft_{default}$

Min num failures constraint

- ft_{disk-g} … sion (max k = 30)

Max code width constraint

- Default AFR x $ft_{default}$ >= Useful life AFR x $ft_{disk-group}$
  - Reconstruction IO: k x disk-capacity x AFR

Redundancy scheme selection

- $ft_{disk-group}$ MTTDL >= ft = 16%)
  - MT to irrecoverable data loss

Target reliability constraint

- Failure ailures tolerated in $ft_{default}$ $_{group}$

Min num failures constraint

- ft_{disk-g} ion (max k = 30)

Max code width constraint

- Default AFR x ft >= $_{group}$
  - Rec k x disk-capacity x AFR

Max reconstruction work constraint

Redundancy scheme selection

- $ft_{disk-group}$ MTTDL >= ft = 16%)

Target reliability constraint

  - MT irrecoverable data loss

- Failures failures tolerated in $ft_{default}$ $_{group}$

Min num failures constraint

- ft_{disk-g} (max k = 30)

Max code width constraint

- Default AFR x ft >= $_{group}$

Max reconstruction work constraint

  - Rec k x disk-capacity x AFR

- $ft_{disk-group}$ reconstr. time <= max reconstr. time (1.5 hrs)

Redundancy scheme selection

- $ft_{disk-group}$ MTTDL >= ft = 16%)
  - MT irrecoverable data loss

Target reliability constraint

- Failure tolerated in $ft_{default}$ $group$

Min num failures constraint

- ft_{disk-g} (max k = 30)

Max code width constraint

- Default AFR x ft >= $group$
  - Rec x disk-capacity x AFR

Max reconstruction work constraint

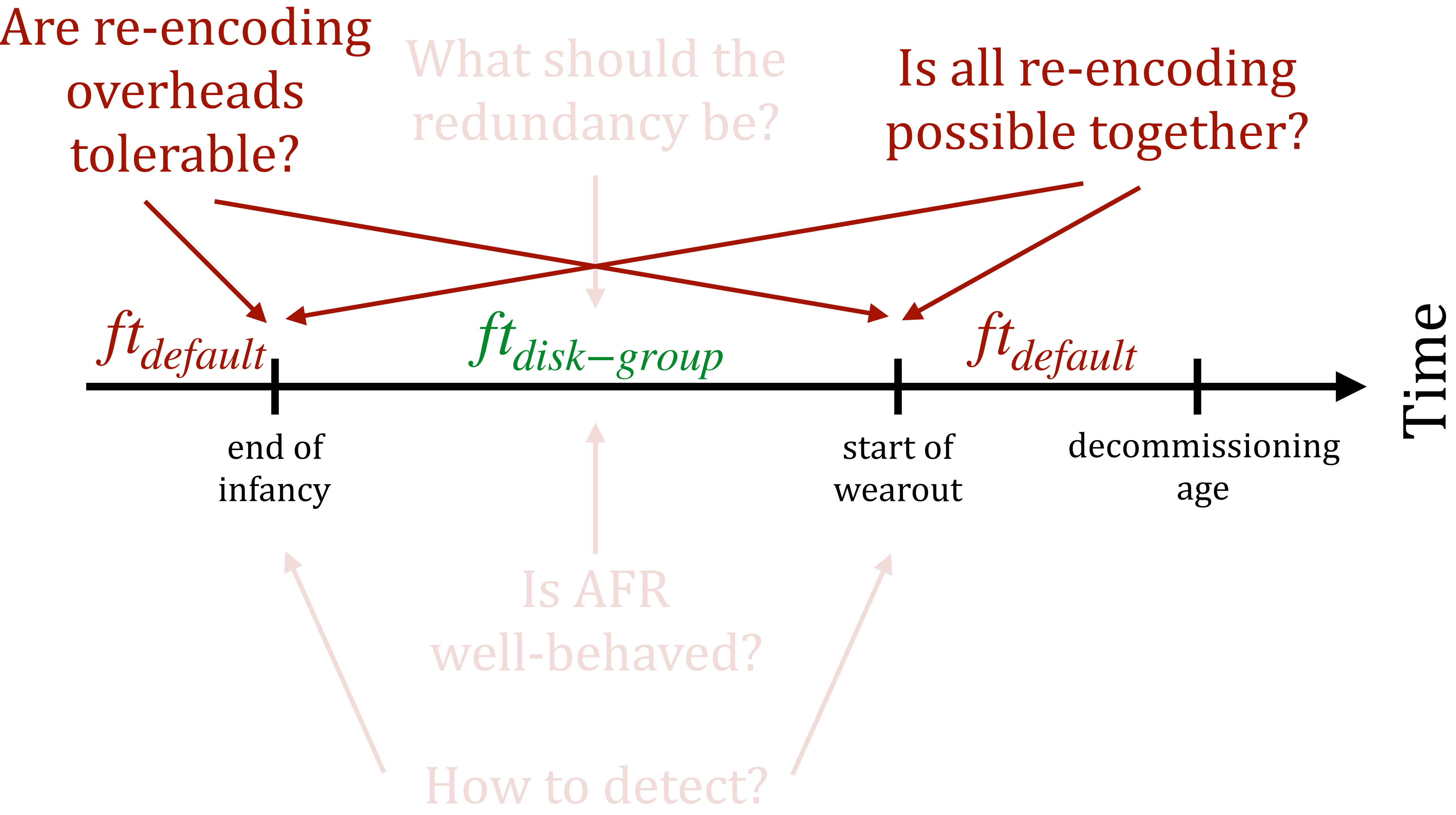
- ft_{disk-g} time (1.5 hrs)

Max reconstruction time constraint

HeART is possible, but is it feasible?

- Data gets re-encoded twice for each disk
    - Infancy —> useful life
    - Useful life —> wearout
- Read—re-encode—write cycle can be very expensive
    - Re-encoding 1TB disk from 30-of-33 to 6-of-9 is at least 75TB IO
- Re-encoding IO can hurt because of two main reasons:
    - Wide redundancy schemes used
    - Too many disks requiring re-encoding at the same time

Disk-group reliability timeline



Are re-encoding overheads tolerable?

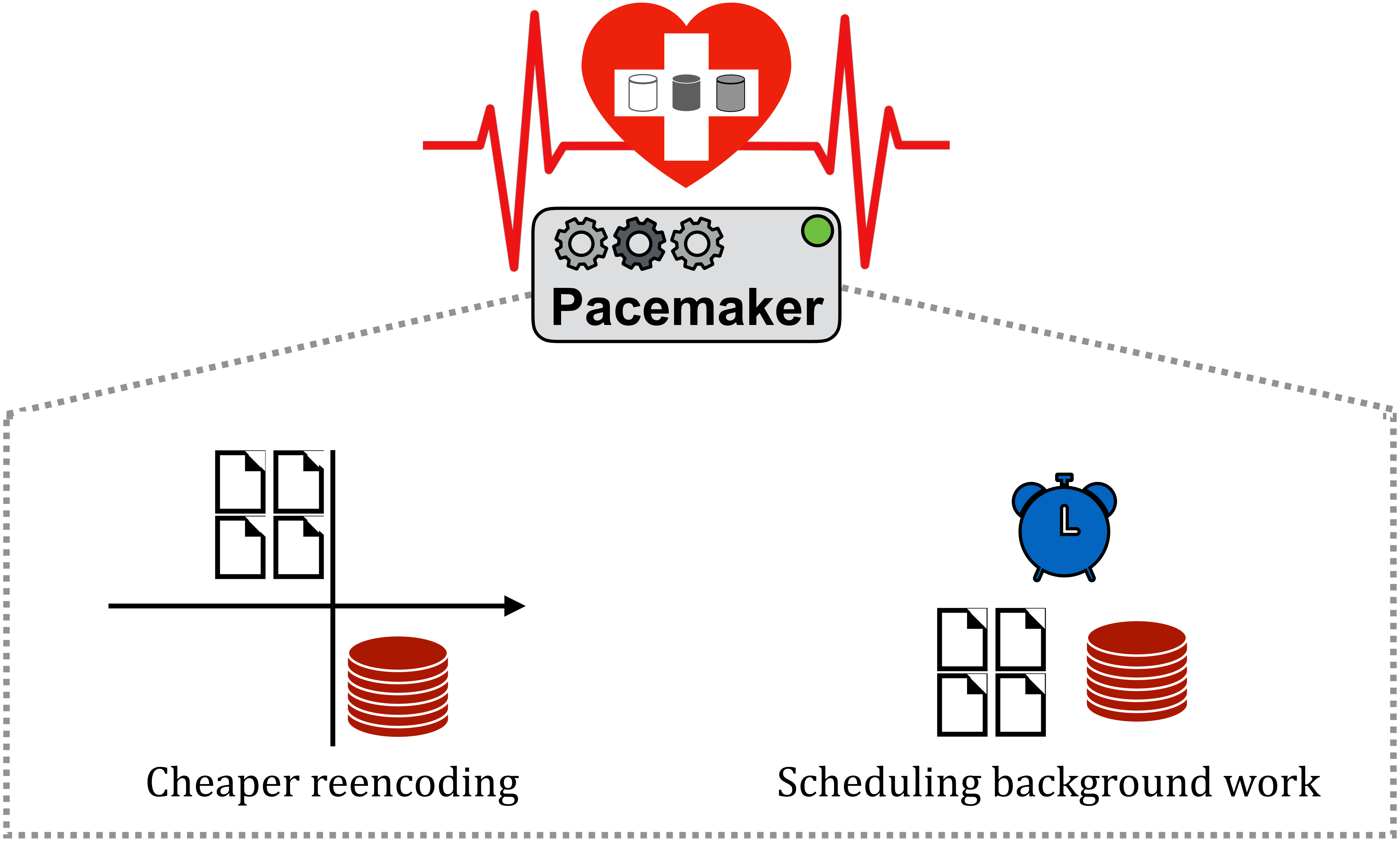
What should the redundancy be?

Is all re-encoding possible together?

Is AFR well-behaved?

How to detect?

Pacemaker: regulating the HeART



Pacemaker

Cheaper reencoding

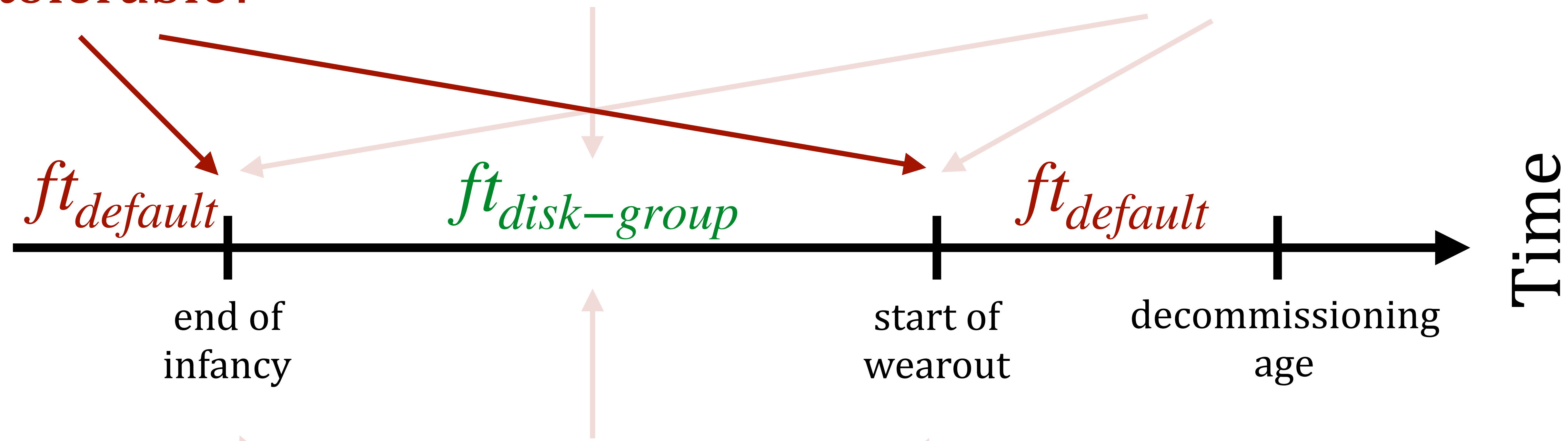
Scheduling background work

Disk-group reliability timeline

Are re-encoding overheads tolerable?

What should the redundancy be?

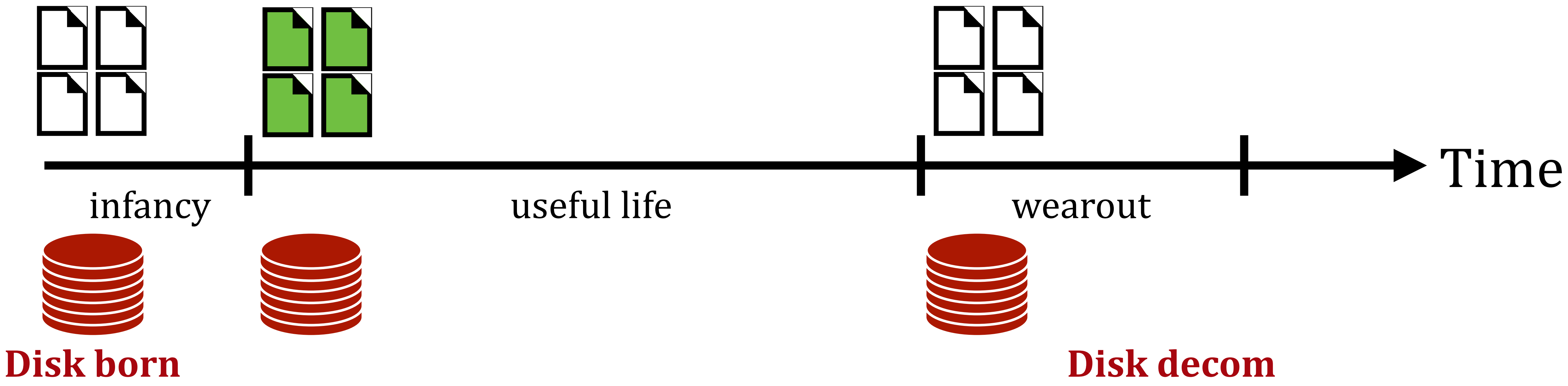
Is the re-encoding possible together?



Is AFR well-behaved?

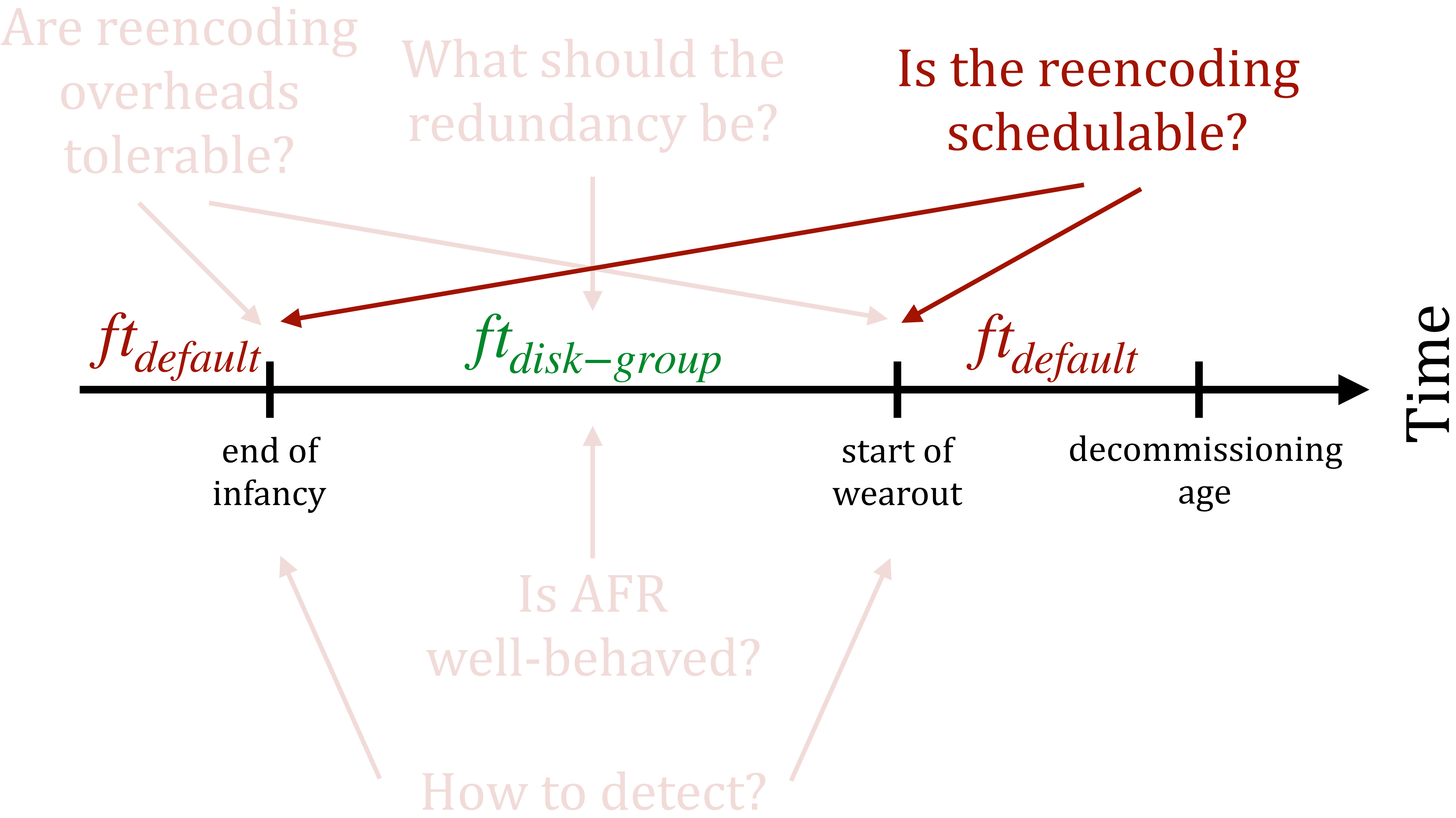
How to detect?

Data reencoding = data redistribution



- Recall that naive read—re-encode—write is very expensive
    - k x disk capacity needs to be read and written
- Key idea: **disks change failure families, it's data need not**
- Moving one stripe unit cheaper than reencoding entire stripe
    - Decouples reencoding IO from redundancy scheme used
    - Moving eliminates the computation overhead, only generates I/O

Disk-group reliability timeline



Schedulable background work

- Infancy to useful life transition is completely schedulable
  - Only impacts savings because of reduced useful life
- Useful life to wearout is urgent
  - But not all of it…
- Key observation: **not all disks enter wearout together**
- Incremental disk deployments help schedule urgent work
  - **Only the first disk batch used to detect wearout is urgent**
  - Subsequent disks wearout transitions can be scheduled

Other optimizations

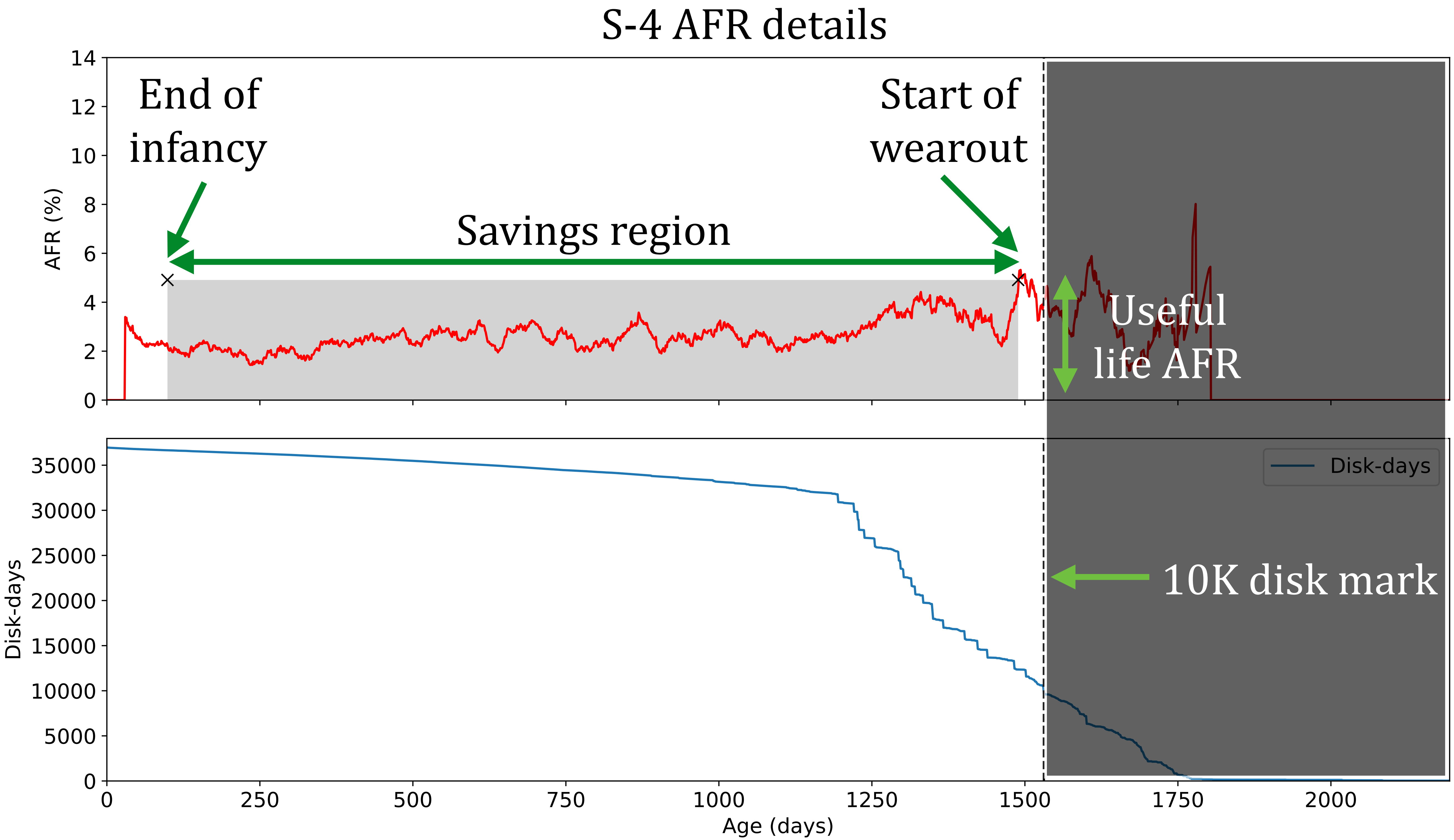
- **Canary disks**
  - Canaries can be encoded in conservative redundancy schemes
  - 2000 for detecting end of infancy & 1000 for detecting wearout
- **Useful life AFR buffer**
  - Buffer helps protect against jitter in AFR during useful life
  - Buffer also helps in exercising caution when tuning redundancy
- **Deciding wearout based on what $ft_{disk-group}$ can tolerate**
  - Useful life redundancy scheme chosen on basis of detected AFR
  - Transition to wearout based on what the scheme can tolerate
- **Iterative change point detection**
  - One-shot change point detection too conservative
  - More data => lower useful life AFR => greater savings

The Backblaze dataset

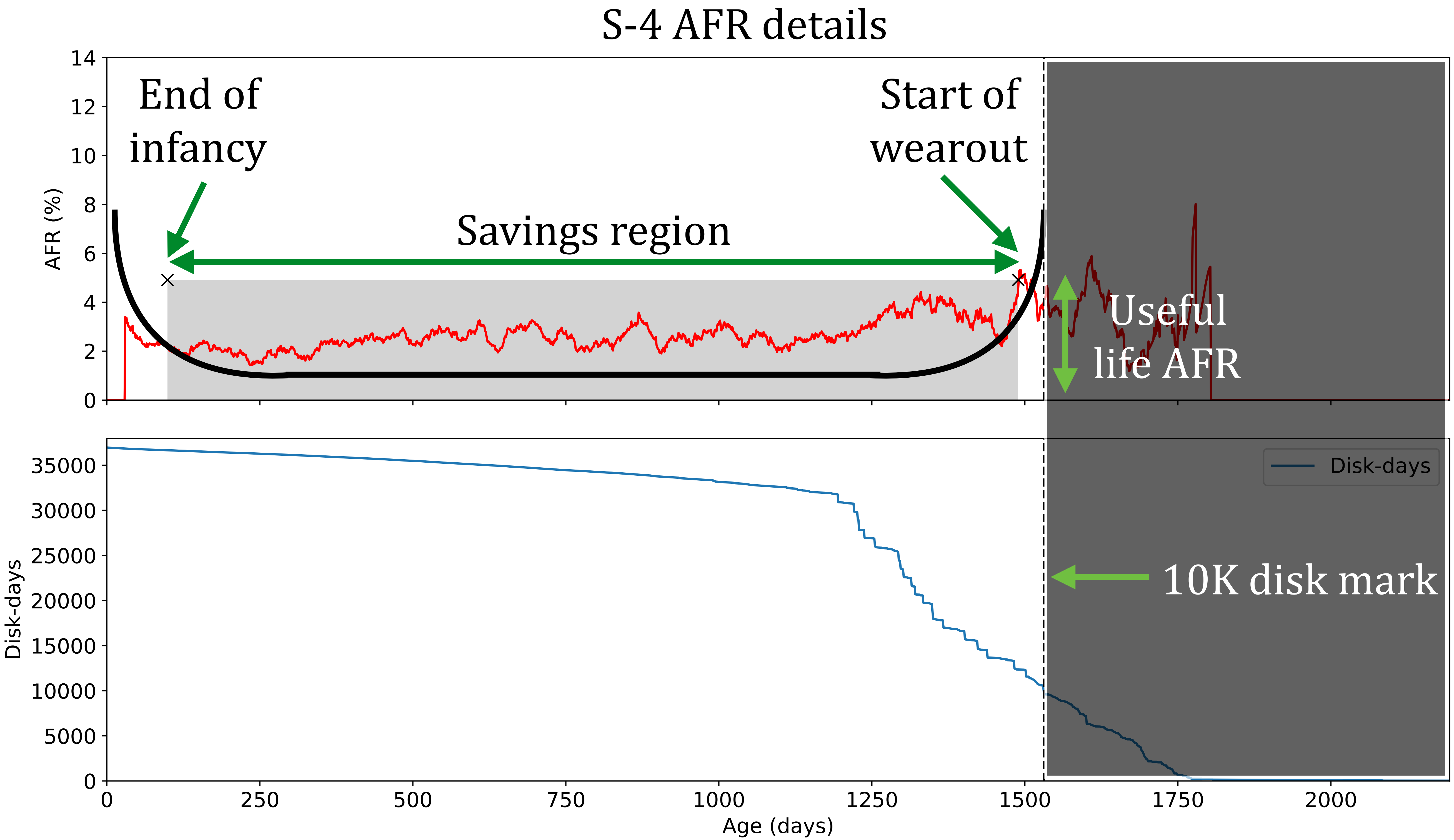
- 100K+ HDDs belonging to Backblaze: a backup company
  - Daily reliability statistics from mid 2013 - mid 2019
  - Open sourced
  - 7 drive makes/models with significant number of disks to test:

Disk Grp	Num Drives	Num Failed	Age so far (yrs)
S-4	36962	3535	6
H-4A	8708	137	6
H-4B	16316	207	5
S-8C	10150	275	3
S-8E	14716	331	2.5
S-12E	35435	735	1.5
H-12E	9680	10	0.5

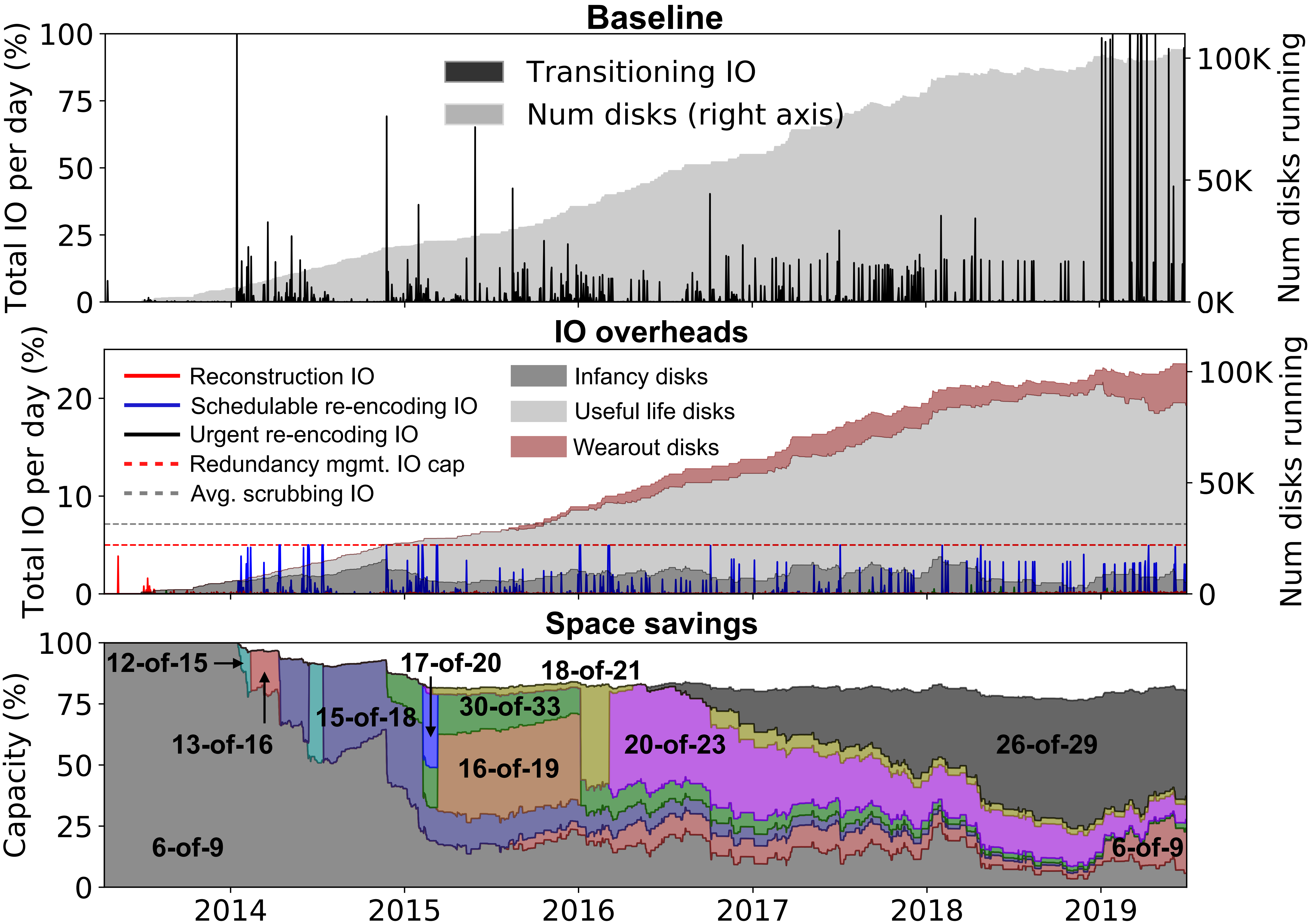
HeART in action on a disk-group



HeART in action on a disk-group



HeART + Pacemaker on Backblaze

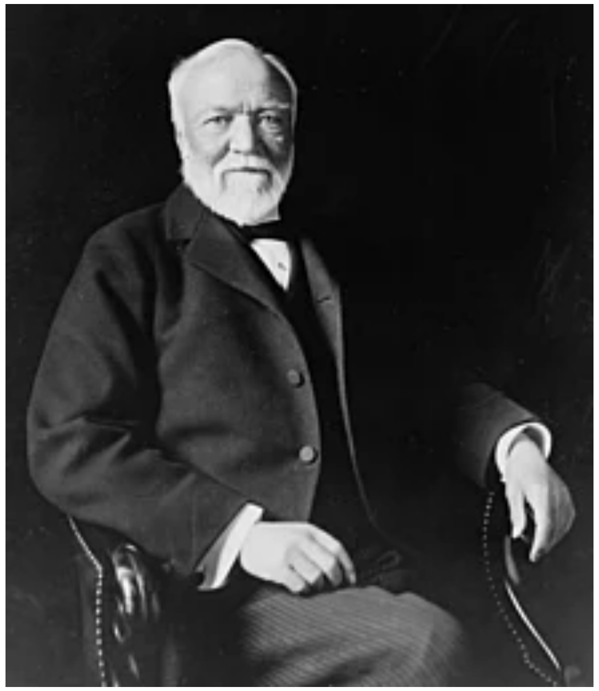


HeART summary

- Exploiting reliability heterogeneity reduces storage cost
- Overall >20% space savings observed on production dataset
- Less than 5% IO bandwidth spent in redundancy mgmt
- **HeART**: an online heterogeneity-aware redundancy tuner
  - actively engages with disk bathtub curves
  - built-in online anomaly and change point detector
- **Pacemaker**: performs efficient redundancy management
  - data redistribution instead of data reencoding
  - converts urgent redundancy mgmt IO into schedulable IO

References

1. Xia, Mingyuan, et al. "**A Tale of Two Erasure Codes in HDFS.**" *FAST*. 2015.
2. Sathiamoorthy, Maheswaran, et al. "**Xoring elephants: Novel erasure codes for big data.**" *VLDB.* 2013.
3. Guha, Sudipto, et al. "**Robust random cut forest based anomaly detection on streams.**" *ICML*. 2016.
4. Truong, Charles, Laurent Oudre, and Nicolas Vayatis. "**ruptures: change point detection in Python.**" *arXiv:1801.00826.* 2018.
5. Rashmi, K. V., et al. **"A hitchhiker's guide to fast and efficient data reconstruction in erasure-coded data centers."** *ACM SIGCOMM Computer Communication Review.* 2015.



“My heart is in the work”

“My work is in the HeART”