Practical Scrubbing

Getting to the bad sector at the right time

George Amvrosiadis
Bianca Schroeder
University of Toronto

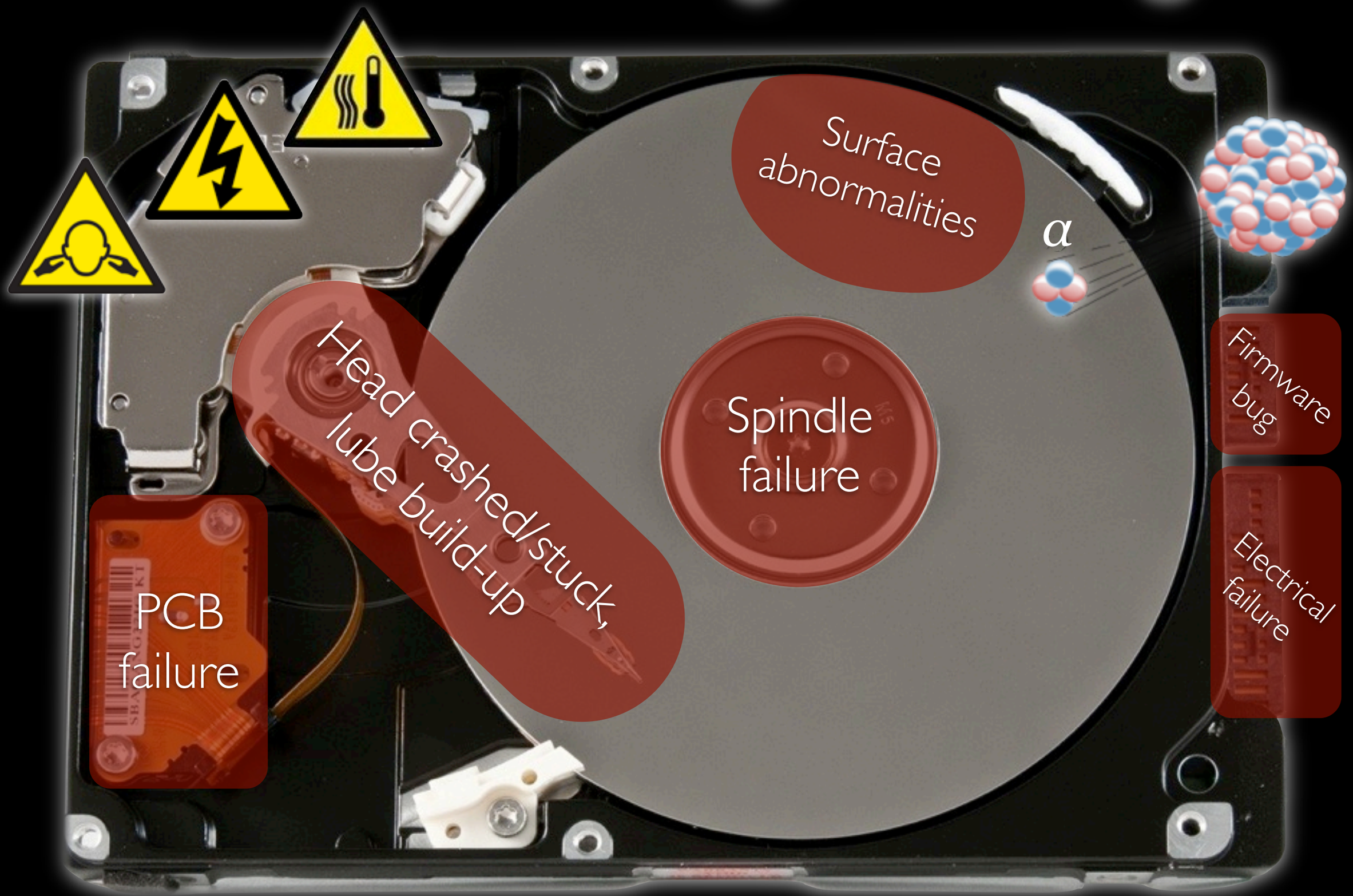
Alina Oprea
RSA Laboratories

The Security Division of EMC

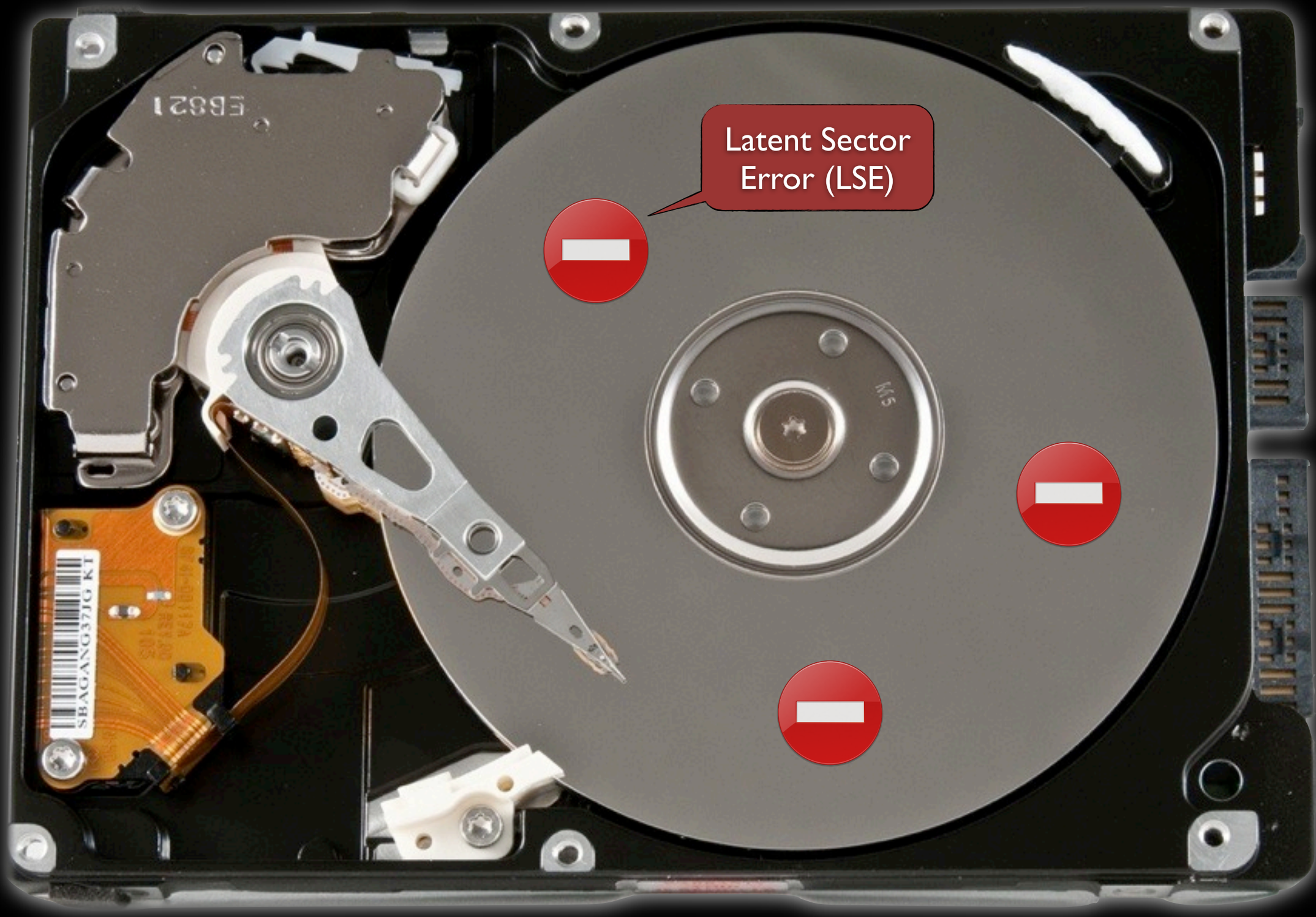
Hard disk errors & Scrubbing

What could go wrong?

What could go wrong?



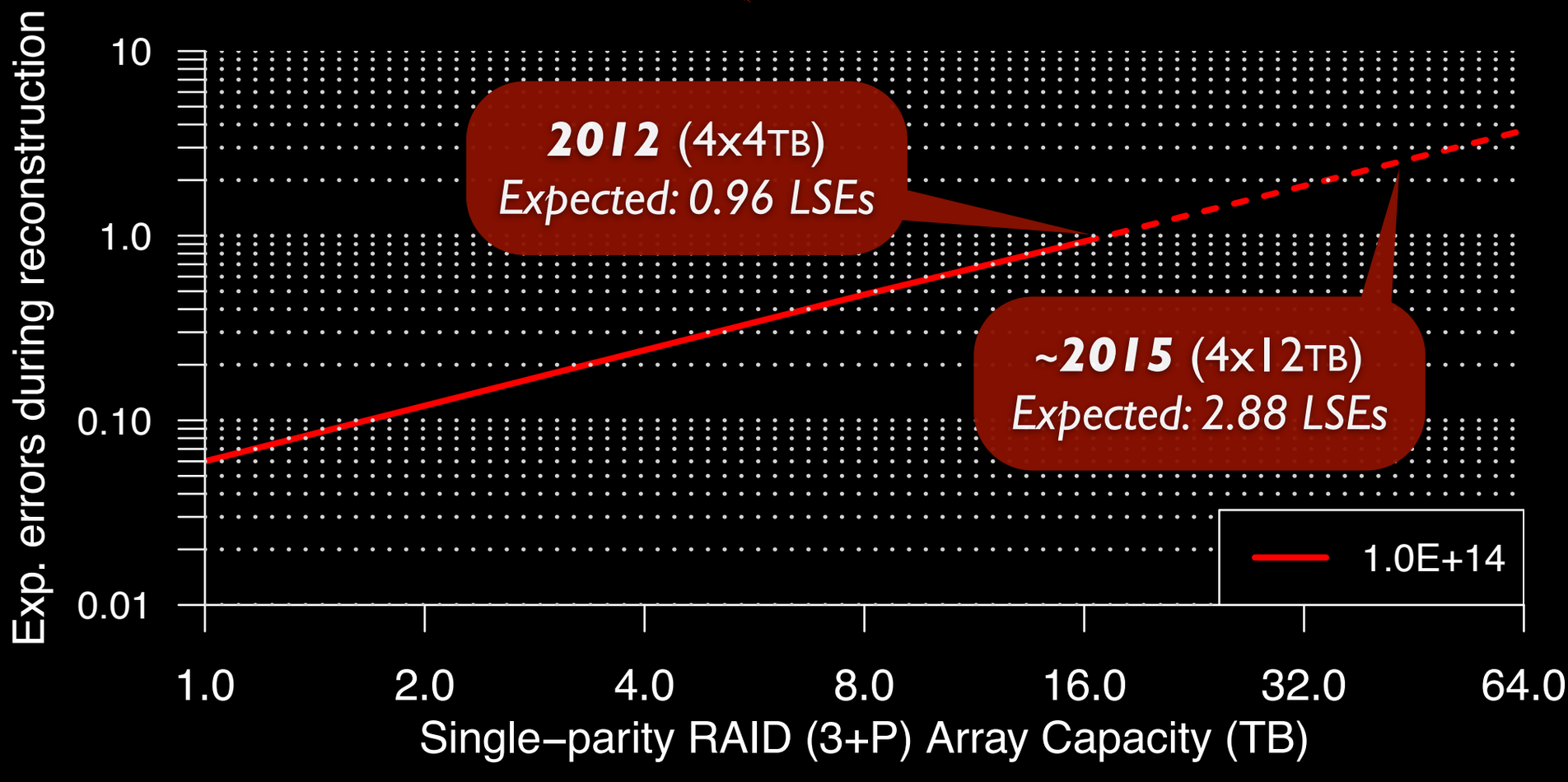
What could go wrong?



L. Bairavasundaram et al., “An analysis of latent sector errors in disk drives”, ACM SIGMETRICS 2007.

What could go wrong?

No Redundancy + LSE = Data Loss



Steven R. Hetzler, "*System Impacts of Storage Trends: Hard Errors and Testability*". In USENIX ;login:, v.36/3

Scrubbing

- *Goal:* Detect LSEs timely to enable recovery
- *How:* Background process verifying sector contents
  - *Detection speed:* verify sectors at high frequency
  - *Verification cost:* avoid delaying workload requests

- *Previous Work:* Focus on detecting LSEs fast
- ***Cost of scrubbing?*** Practical questions raised:

How do I *implement* a scrubber? How do I *configure* it to find LSEs fast?

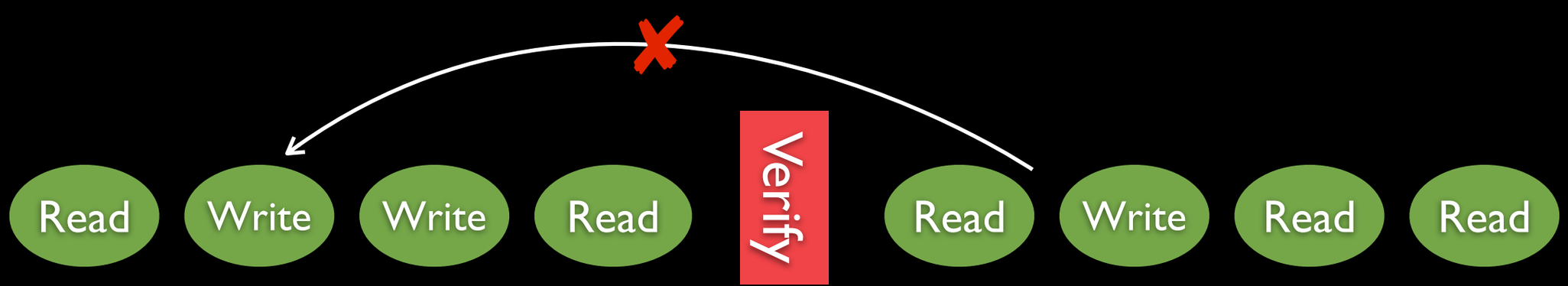
When should I scrub, to *minimize impact* on the system?

Implementation & Configuration

Data scrubbing

- ~~Option 1~~: Use READs to verify data integrity
  - Overhead: *Data transfer cost, Cache pollution*
  - Correctness: *Might not check medium surface*

- Option 2: Use VERIFY firmware command
  - Caveat: *Treated as scheduling barriers*
  - Solution: *Disguise scrubber's VERIFYs as READs*



Data scrubbing

- ~~Option 1~~: Use READs to verify data integrity
  - Overhead: *Data transfer cost, Cache pollution*
  - Correctness: *Might not check medium surface*

- Option 2: Use VERIFY firmware command
  - Caveat: *Treated as scheduling barriers*
  - Solution: *Disguise scrubber's VERIFYs as READs*

System Overview

Filesystem Layer

Generic Block Layer

Scrubbing Framework

?

Verify

Verify

Verify

Verify

Verify

Verify

I/O Scheduler

On-disk cache

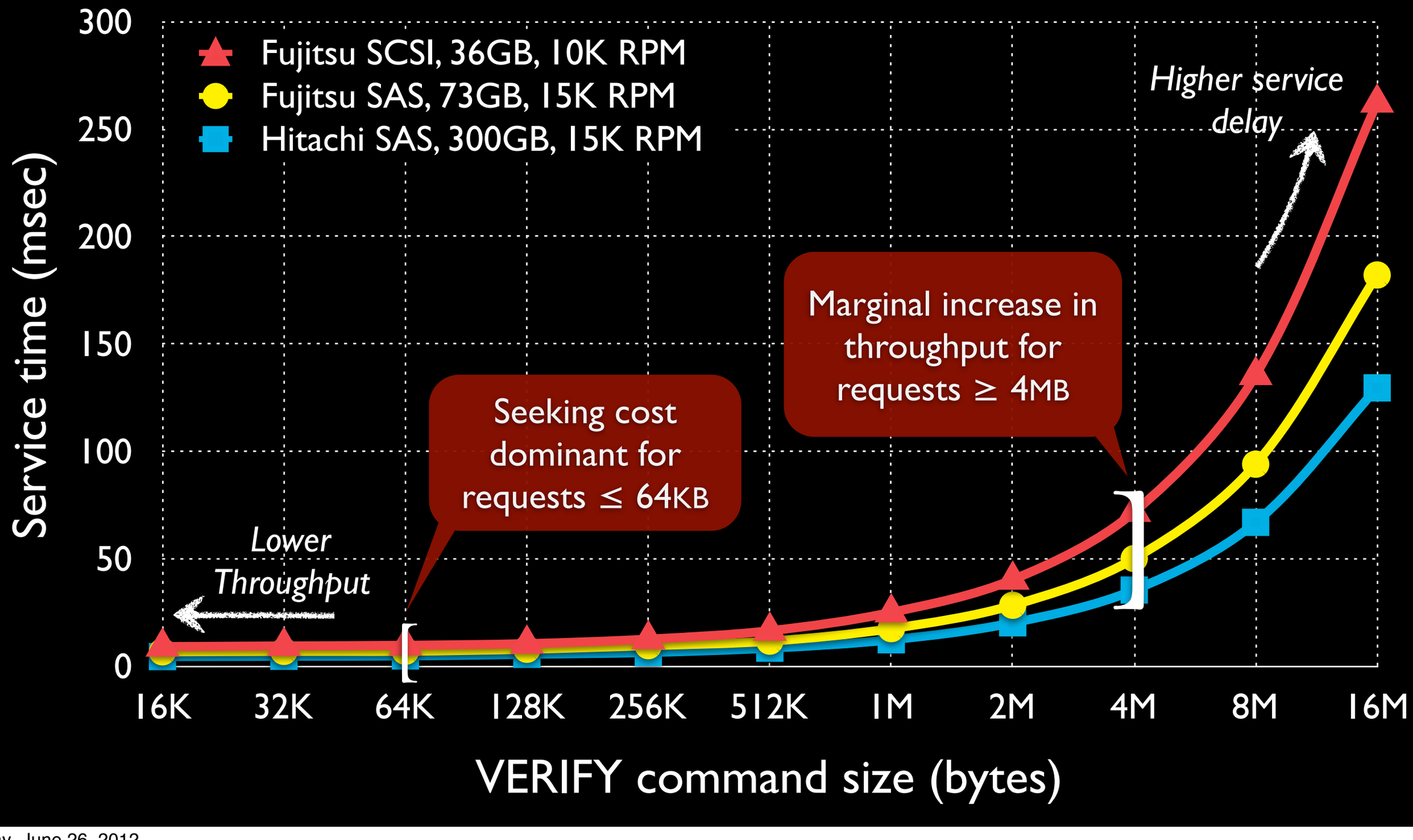
Hard disk

Parameter 1:

VERIFY request size

larger requests! *smaller requests?!*

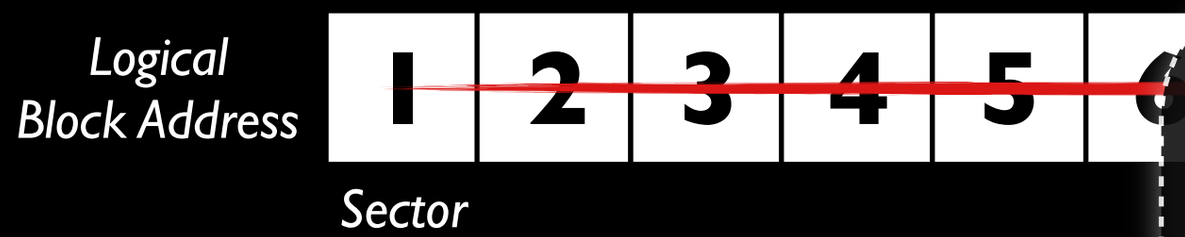
Finding the sweet spot between scrub throughput and service delay



Parameter 2:

Scrubbing Order

- ~~Sequential~~ scrubbing (used in the field today)

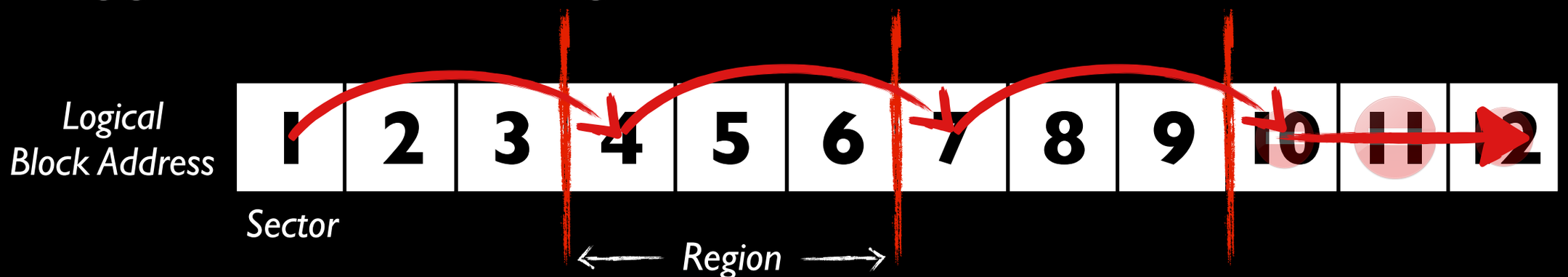


Can we afford this?

HDD capacities: CAGR +100%

Data transfer speed: CAGR +40%

- Staggered scrubbing (fast LSE detection) *[Oprea, Juels FAST'10]*



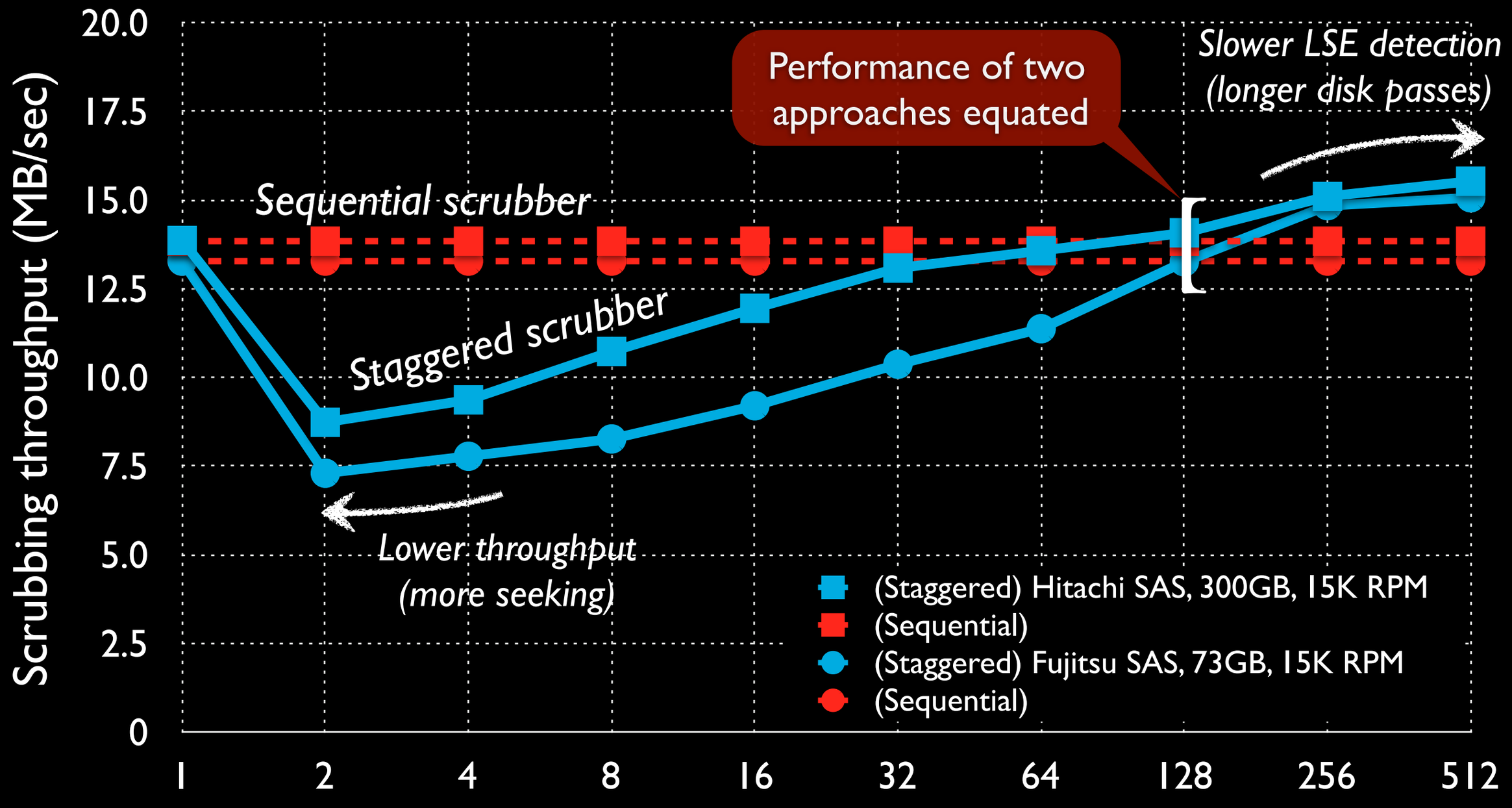
Staggered scrubbing guarantees fast LSE detection, but:
seeking overhead never evaluated in implementation

Parameter 3:

Number of Regions

more, smaller regions! *fewer, larger regions?!*

Limit seeking between regions, but retain frequent disk passes



System Overview

Filesystem Layer

Generic Block Layer

VERIFY size:
[64KB, 4MB]

Scrubbing Framework

Regions:
≥128

I/O Scheduler

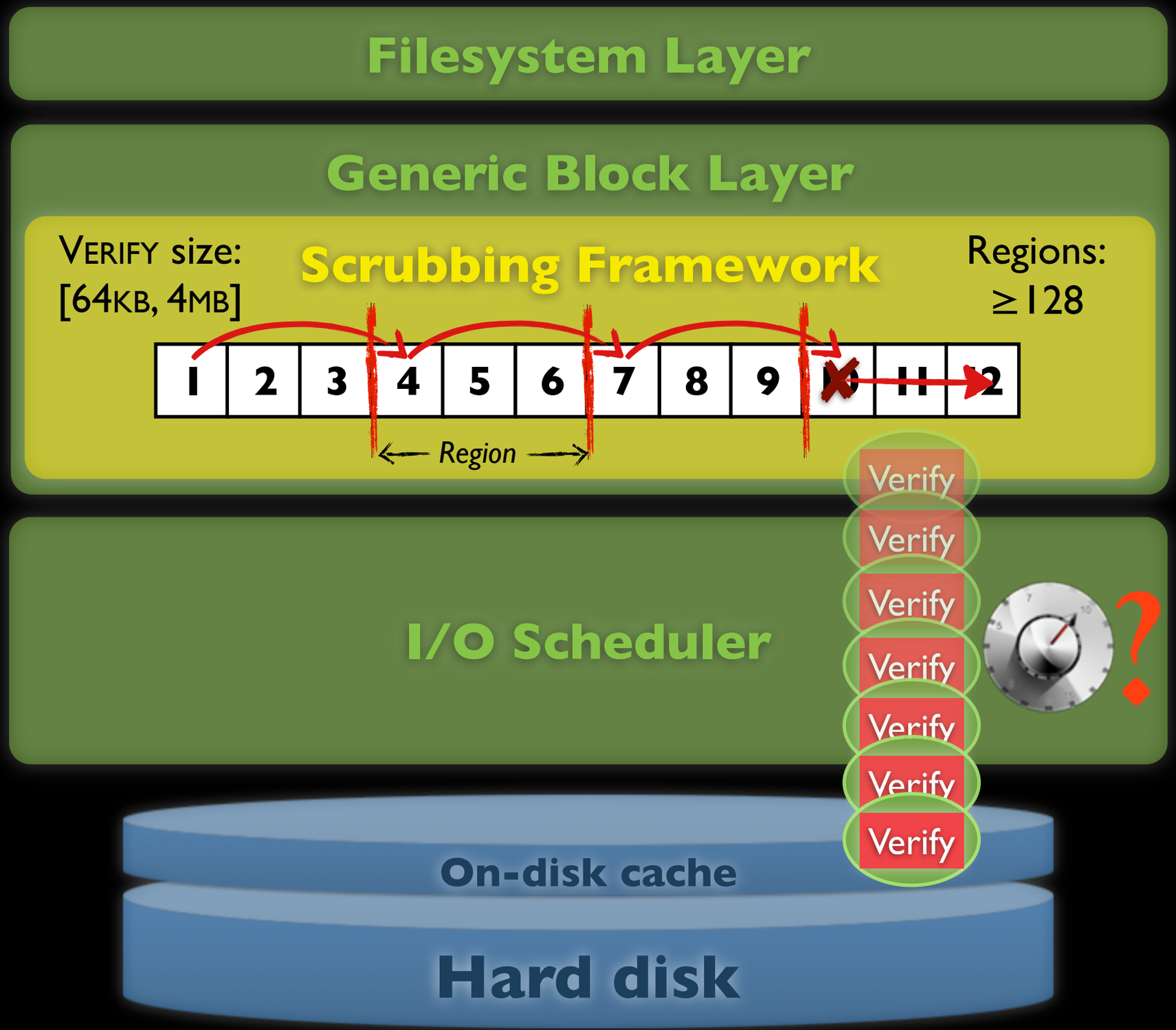
Verify
Verify
Verify
Verify
Verify
Verify
Verify

On-disk cache

Hard disk

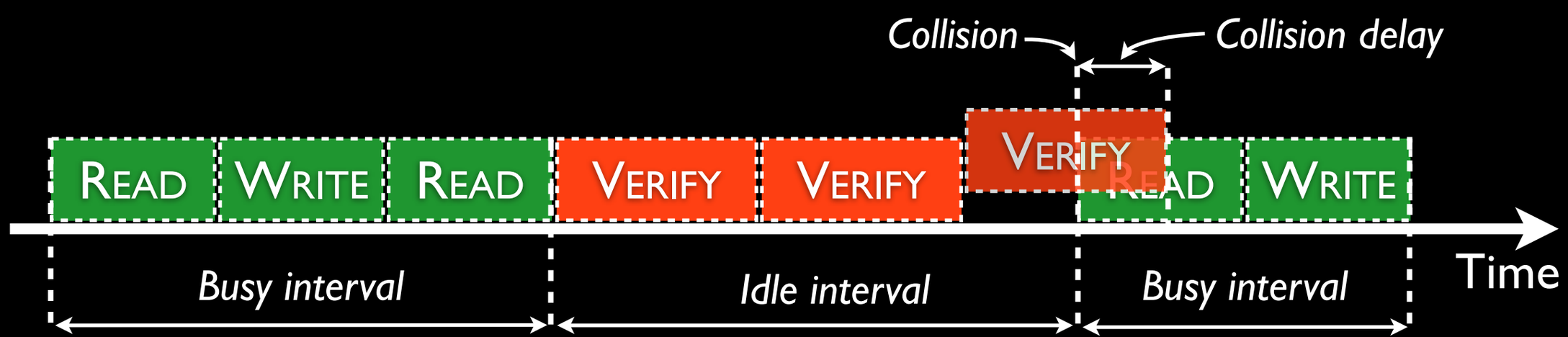
Minimizing Impact

System Overview



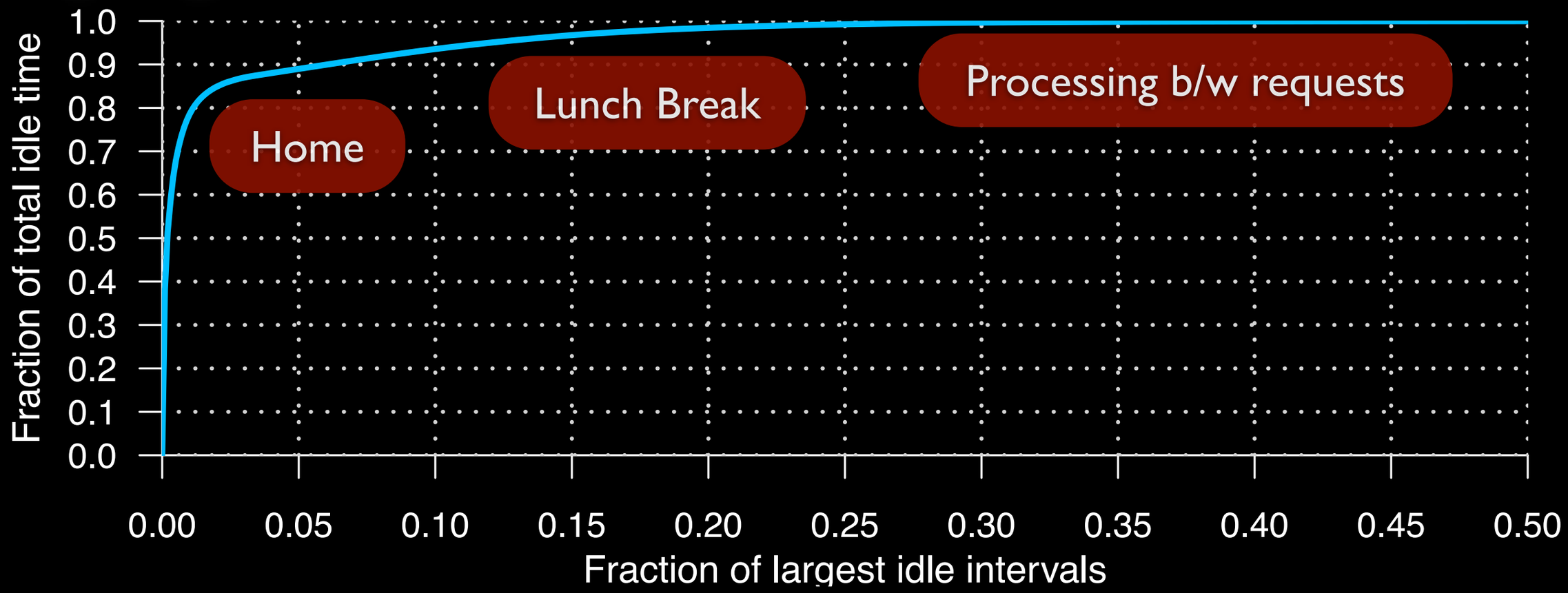
Background Scheduling

- Fire VERIFY requests only when disk otherwise idle
  - Previous work: *Focus on unobtrusive READS/WRITES [Lumb'02, Bachmat'02]*
- Avoid collisions with workload requests
  - Start time: *When should we start firing VERIFYS?*
  - Stop time: *When do we stop to avoid collision?*
- Statistical analysis of idleness
  - I/O traces: *2 systems, 77 disks, diverse workloads [SNIA, IOTTA repository]*



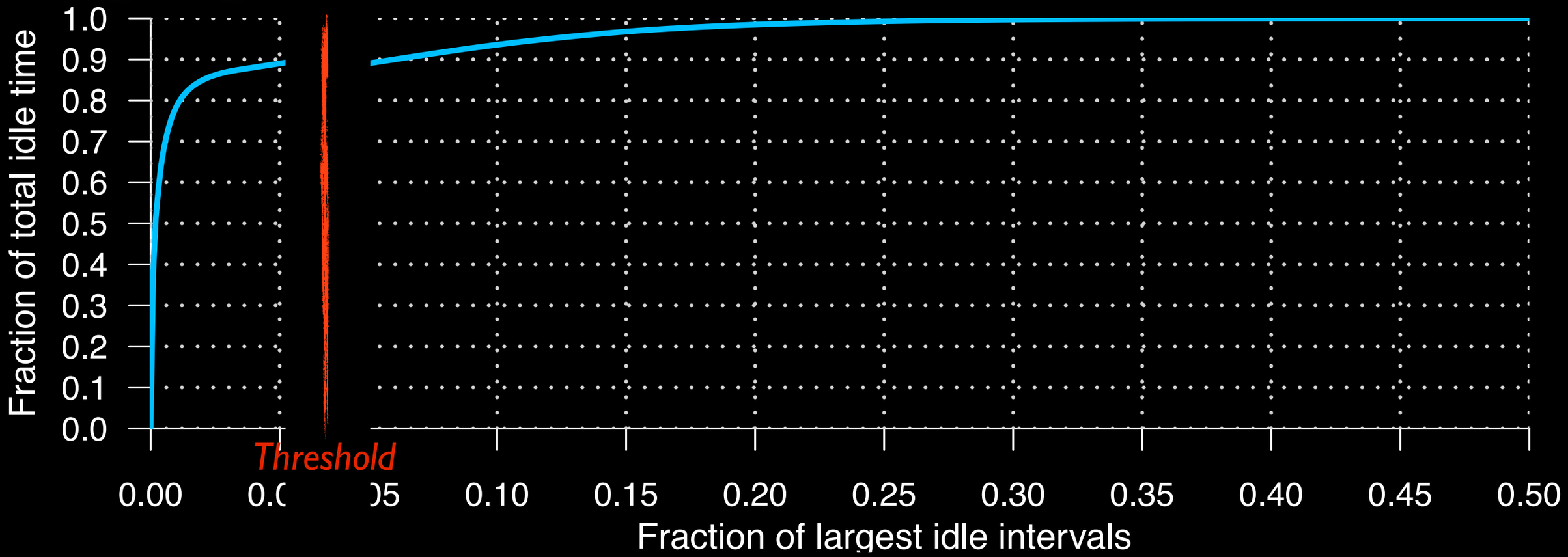
Idleness & Long Tails

Property: Long Tail - *Majority of idle time in few idle intervals* *[Riska '09]*

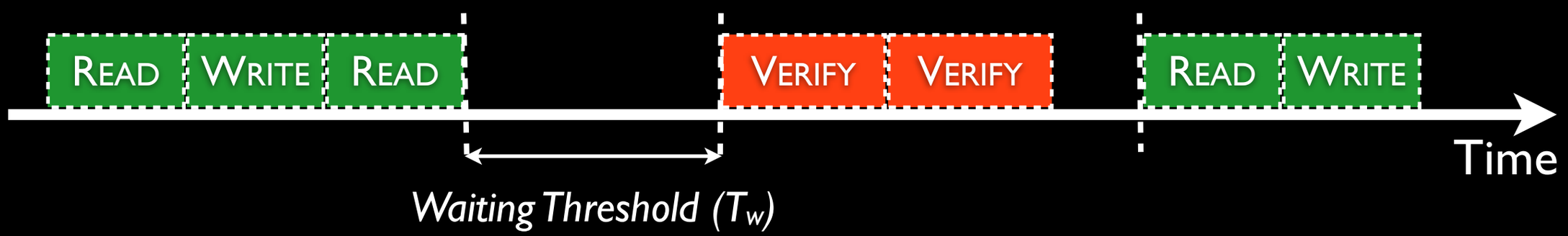


Idleness & Long Tails

Property: Long Tail - *Majority of idle time in few idle intervals* *[Riska '09]*

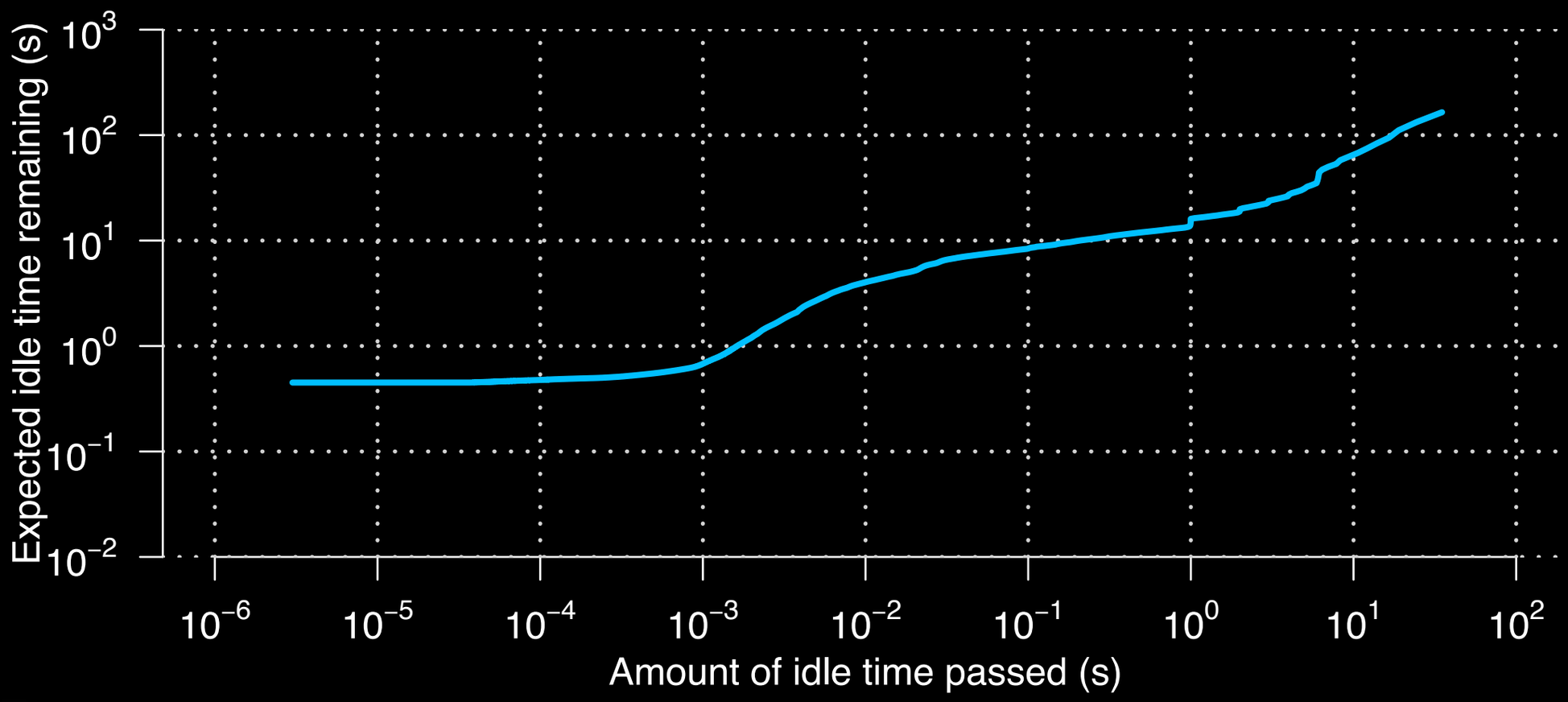


Predictor: Waiting - *Fire past threshold*

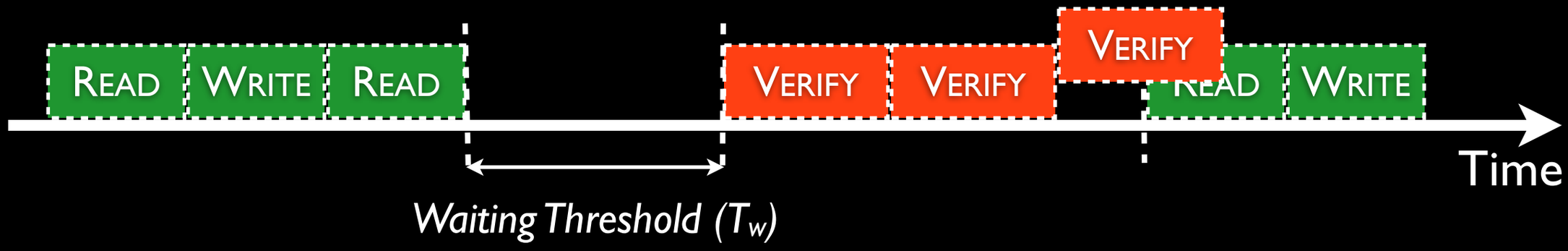


Idleness & Long Tails

Property: Long Tail - *Majority of idle time in few idle intervals* *[Riska '09]*

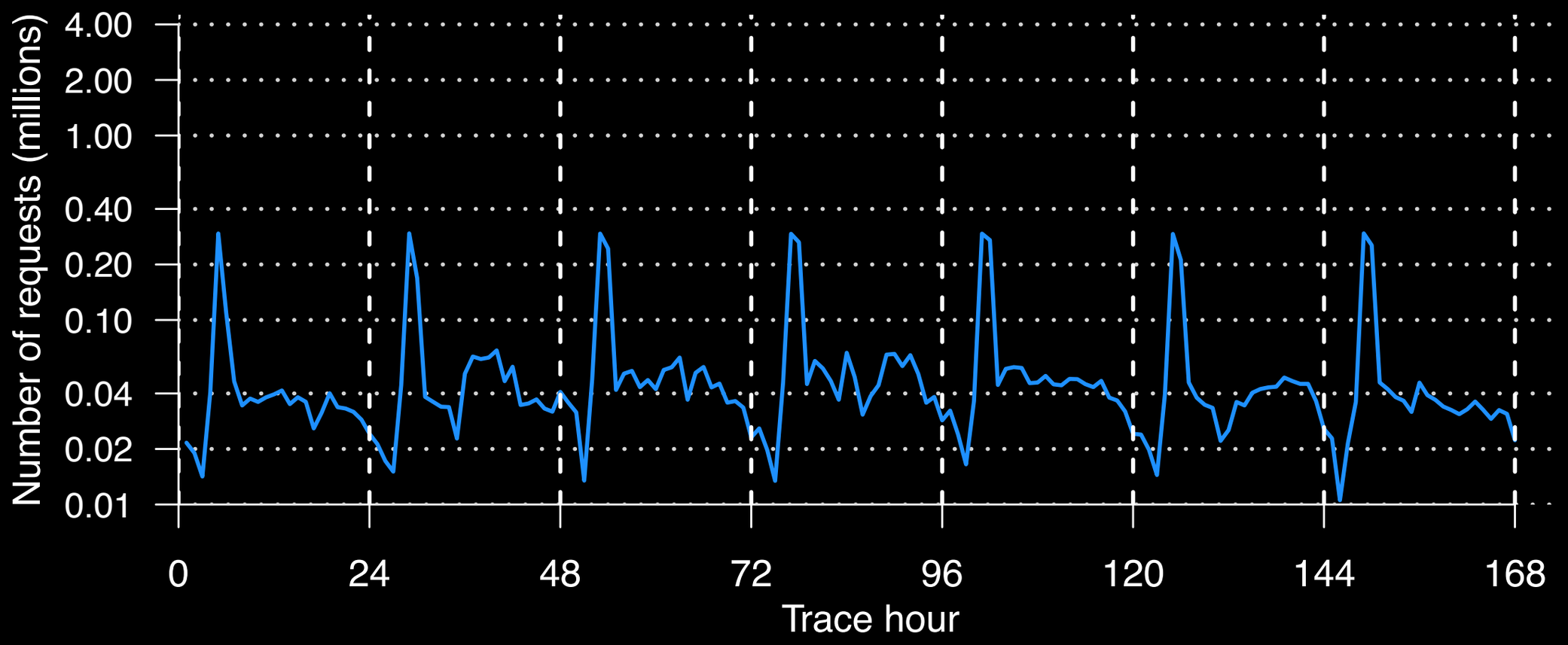


Predictor: Waiting - *Fire past threshold, stop only on collision*

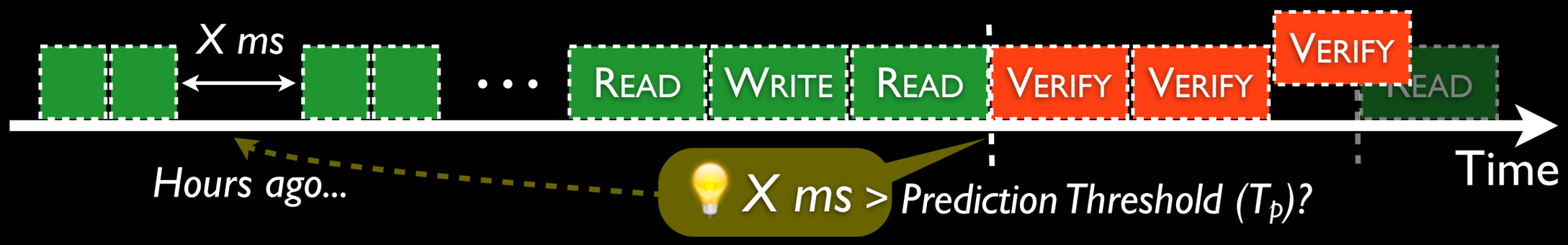


Idleness & Periodicity

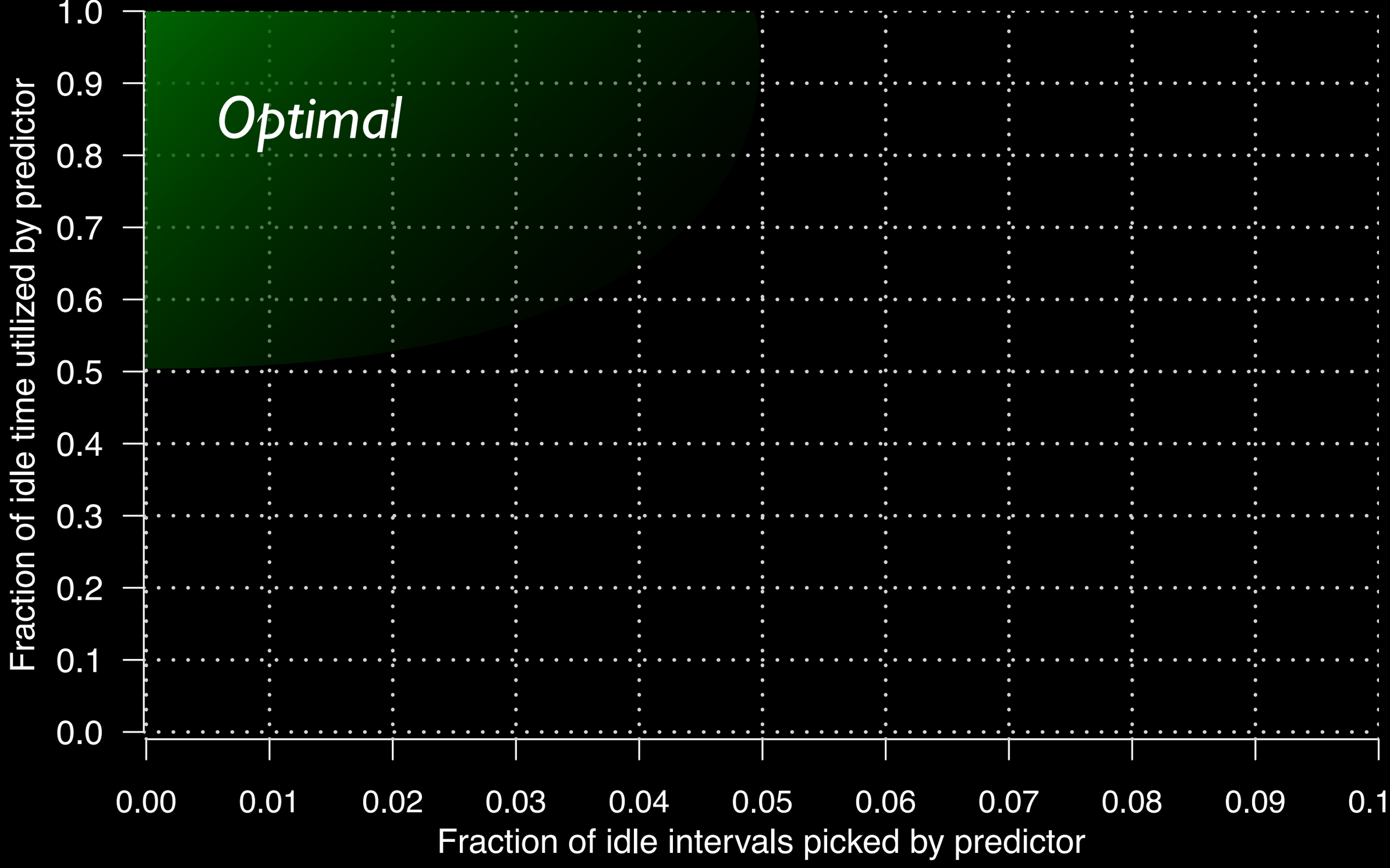
Property: Periodicity - *Repeating patterns in disk traffic*



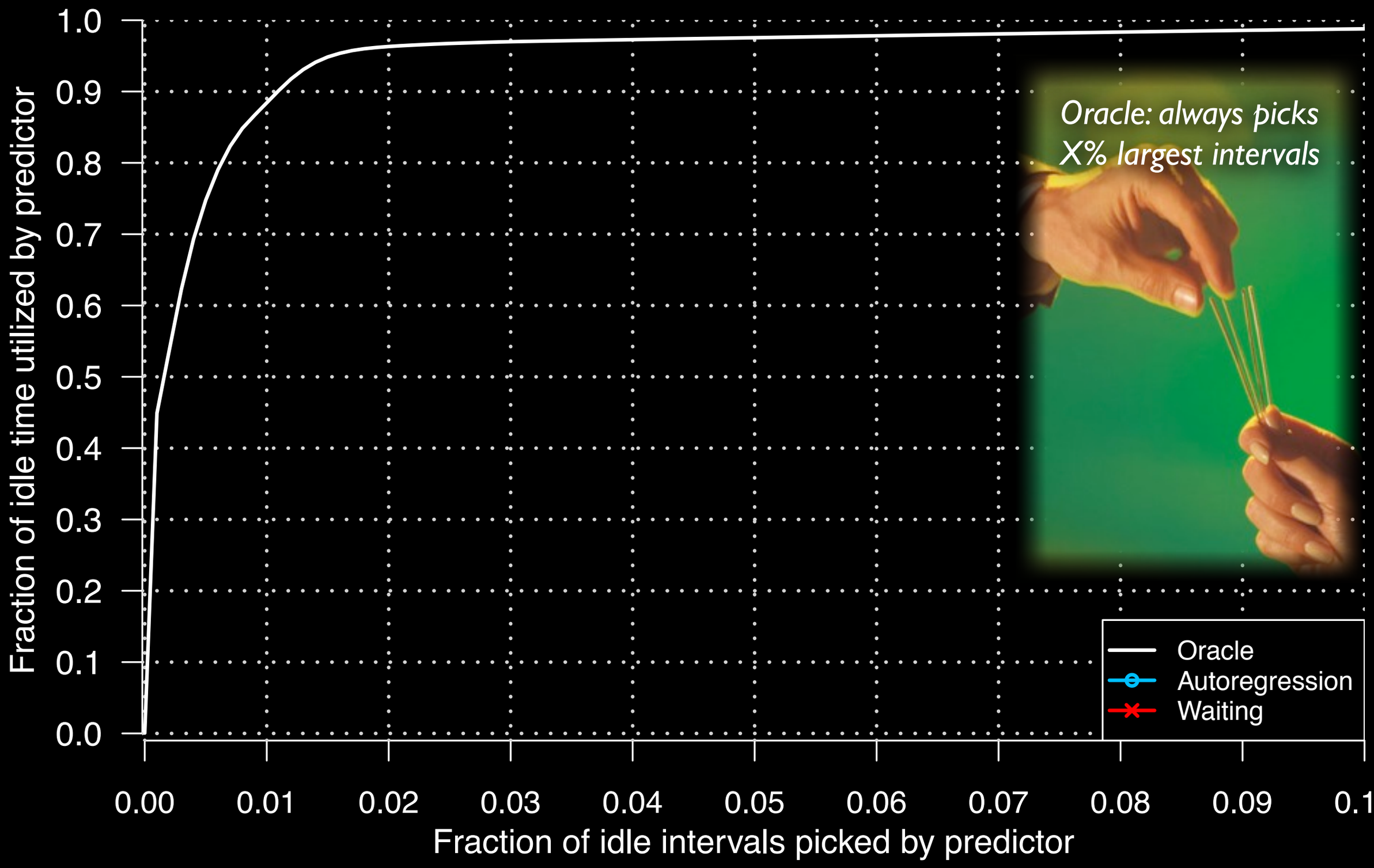
Predictor: Autoregression - *Fire if prediction > threshold, don't stop*



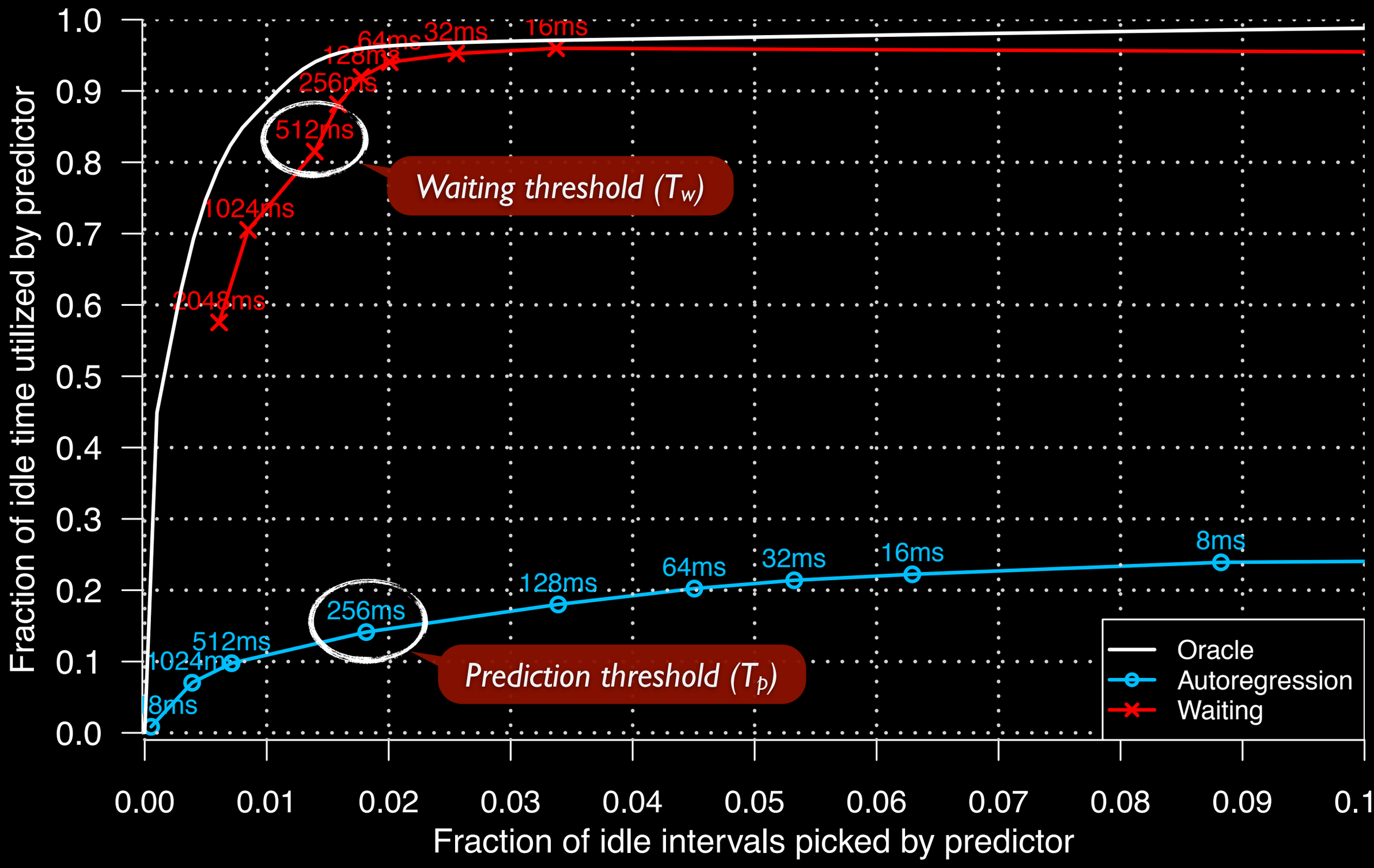
Predictor evaluation



Predictor evaluation



Predictor evaluation



Fine-tuning the wait

VERIFY
request size

Waiting
threshold

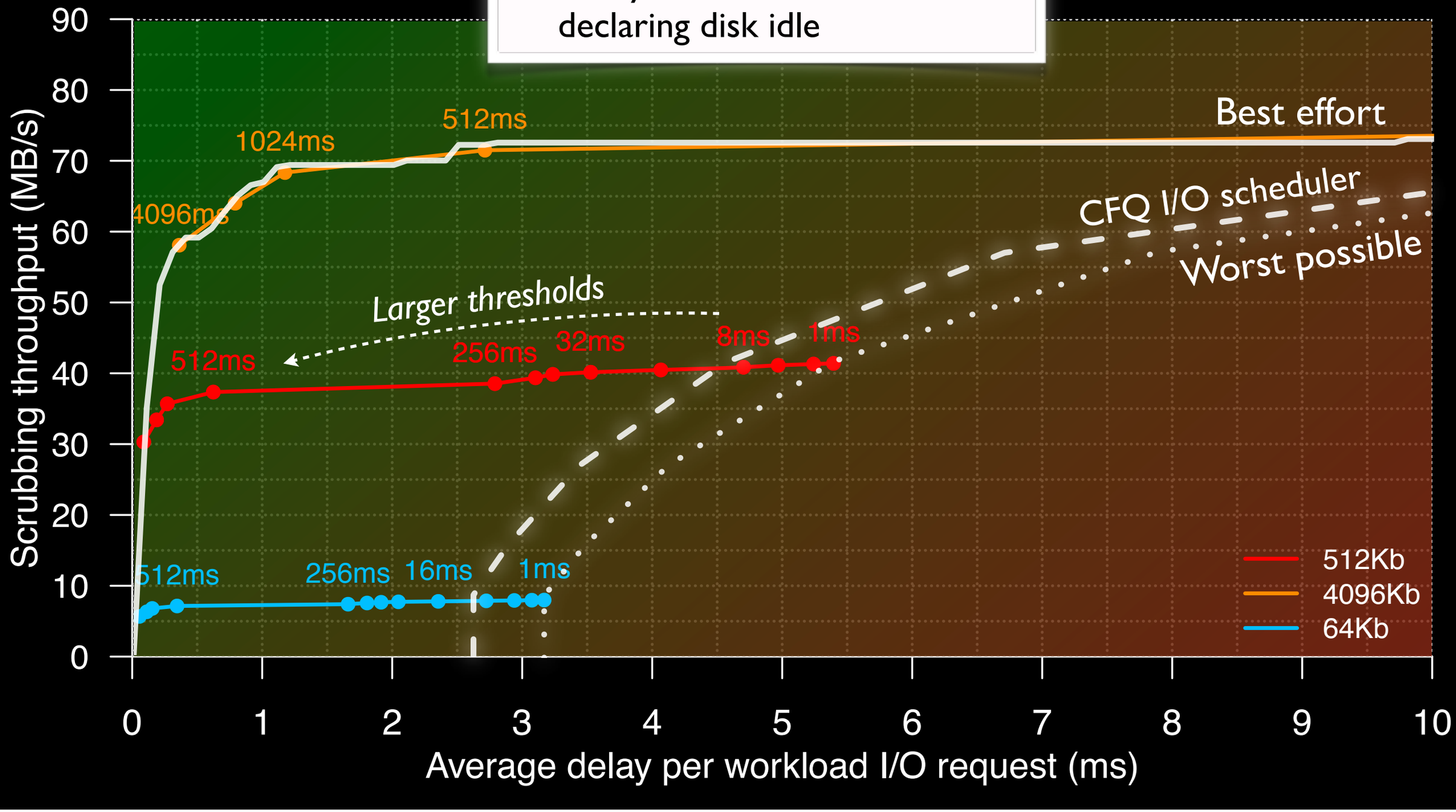
Fine-tuning the wait

VERIFY request size

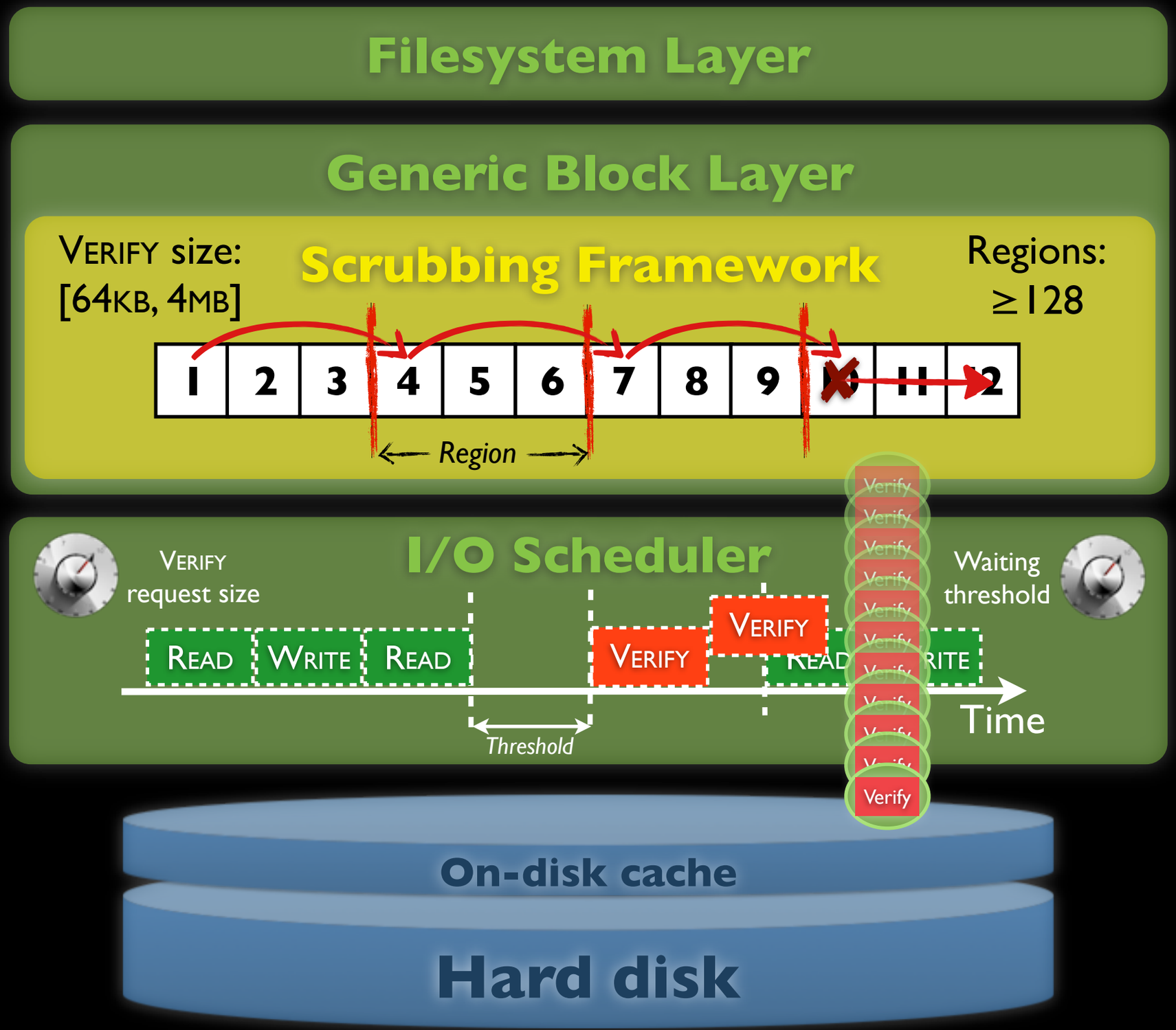
Complete Fair Queueing (CFQ):

- Default Linux I/O scheduler
- Always waits *10ms* before declaring disk idle

Waiting threshold

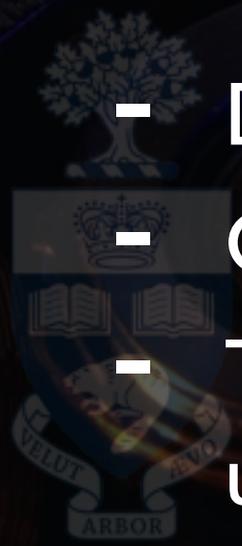


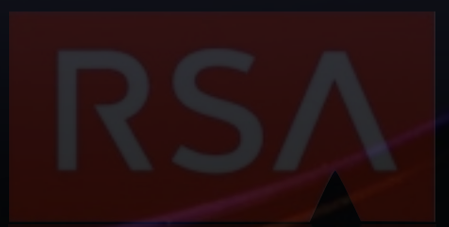
System Overview



Also in the paper

- Scrubbing framework implementation
  - Open-sourced* framework simplifying development of scrubbing algorithms
  - Why VERIFY is implemented incorrectly in ATA drives
  - Scrubbing impact evaluation in implementation using synthetic/realistic workloads
- Detailed statistical analysis
  - Detection of longer tails than reported in previous work
  - Characterization of periods in traces
  - TPC-C benchmark: idle time distribution unrepresentative of OLTP workloads





The Security Division of EMC

* Kernel/User-level sources available at http://www.cs.toronto.edu/~gamvrosi