Paul E. McKenney, Meta Platforms Kernel Team

Linux Plumbers Conference Refereed Track, November 14, 2023

Hunting Heisenbugs

Heisenbugs and impressionism: The closer you get, the less you see!

© 2023 Meta Platforms

Overview

- Heisenbugs, Then and Now
- How to Hunt Heisenbugs
- Heisenbugs: The Goal

Overview

- Heisenbugs, Then and Now
- How to Hunt Heisenbugs
- Heisenbugs: The Goal

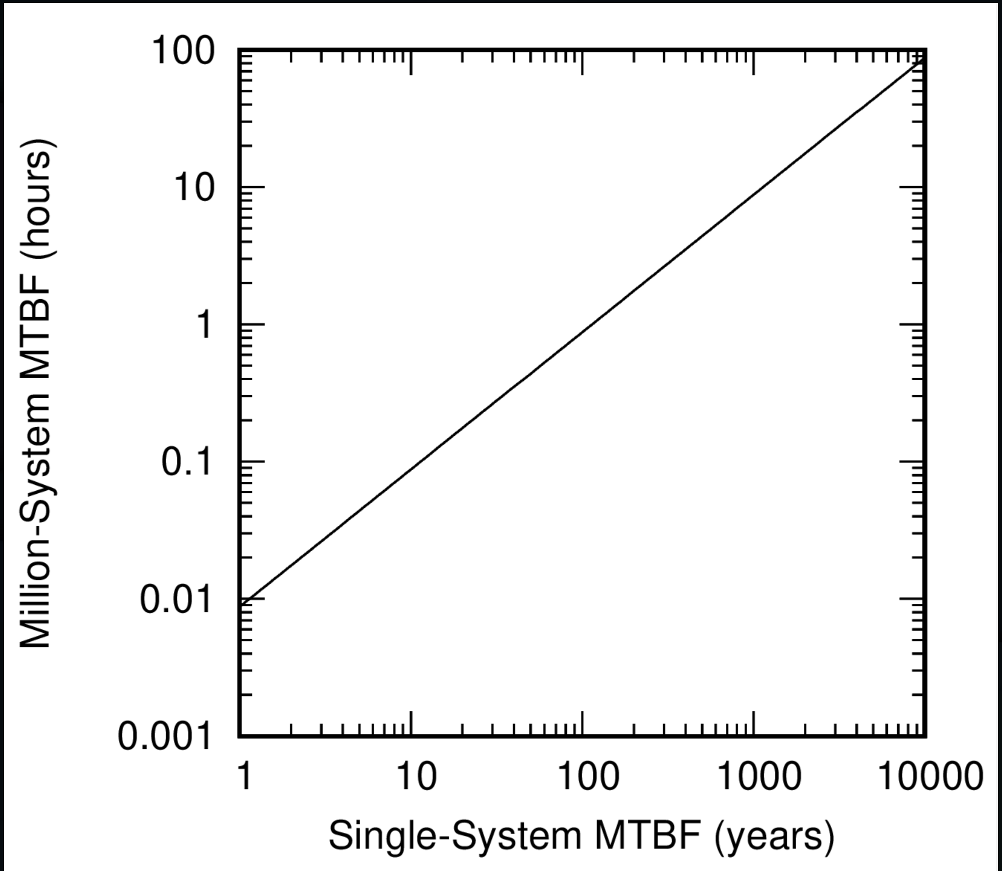
How to avoid hunting heisenbugs!!!

Heisenbugs, Then and Now

Heisenbugs, Then and Now

- Heisenbugs used to result in horrible life-changing experiences
- But they are increasingly just “Tuesday”

Fleets as Heisenbug Detectors



After Heisenbug Detected?

- Debug based on console output
- Collect debug information via BPF
- Collect debug information via kernel patch
- Set up kernel debugger

After Heisenbug Detected?

- **Debug based on console output**
- Collect debug information via BPF
- Collect debug information via kernel patch
- Set up kernel debugger

One failure per 4K systems per week (70-year single-system MTBF)

After Heisenbug Detected?

- Debug based on console output
- Collect debug information via BPF
- Collect debug information via kernel patch
- **Set up kernel debugger**

After Heisenbug Detected?

- Debug based on console output
- Collect debug information via BPF
- Collect debug information via kernel patch
- Set up kernel debugger

Except that 1M instances of kgdb is not fun...

After Heisenbug Detected?

- Debug based on console output
- Collect debug information via BPF
- Collect debug information via kernel patch
- Set up kernel debugger

Except that 1M instances of kgdb is not fun...

Nor is waiting for your alleged fix to deploy!!!

After Heisenbug Detected?

- D bug based on console output
- Collect ug information via BPF
- Collect debug mation kernel patch
- Set up ke

Nor is waiting for ged fix to deploy!!!

Except that 1M ins of kgdb is not fun...

Need a better way

But What If I Don’t Have A Big Fleet?

- I hunted heisenbugs long before “having” a fleet!!!

But What If I Don’t Have A Big Fleet?

- I hunted heisenbugs long before “having” a fleet!!!
- Tens of billions of Linux instances
  - Good to have fewer things going bump in the night
- Potential safety-critical benefit
  - Whether we like it or not, Linux kernel is already increasingly used in safety-critical applications
  - Acceptance test suffices in many cases

Key Heisenbug-Hunting Trick

Key Heisenbug-Hunting Trick

- Why is it a heisenbug?

Key Heisenbug-Hunting Trick

- Why is it a heisenbug?
- Because it occurs only rarely
- Any added debugging changes timing
- Slight changes in timing can reduce incidence

Key Heisenbug-Hunting Trick

- Why is it a heisenbug?
- Because it occurs only r
- Any added debug timing
- Slight chan an reduce incidence

Anti-Heisenbug: Reduce MTBF!!!

What If I Do Have a Fleet?

- One-week test on 50 systems to validate to run for one year on 1M systems?

What If I Do Have a Fleet?

- One-week test on 50 systems to validate to run for one year on 1M systems?
  - MTBF of test systems must be ***six orders of magnitude*** shorter than MTBF of fleet systems

What If I Do Have a Fleet?

- One-week test on 50 systems to validate to run for one year on 1M systems?
  - MTBF of test systems must be ***six orders of magnitude*** shorter than MTBF of fleet systems
  - **In many cases, this is eminently doable**

How to Hunt Heisenbugs

Increase Workload Intensity

- Leverage the philosophy of my high-school track and cross-country coach

Increase Workload Intensity

- Leverage the philosophy of m school track and cross-countr h

Race days were the easy days

Increase Workload Intensity: Kernel

- Most production systems major in userspace execution (if not idle!)
  - 10% kernel utilization is high
  - A few percent kernel utilization is commonplace

Increase Workload Intensity: Kernel

- Most production systems major in userspace execution (if not idle!)
  - 10% kernel utilization is high
  - A few percent kernel utilization is commonplace
- One-to-two orders of magnitude MTBF reduction just by focusing on kernel execution!

Increase Workload Intensity: Kernel

- Most production systems major i erspace execution (if not idle!)
  - 10% kernel utilization i
  - A few percent ker ilization is commonplace
- One-to-two s of magnitude MTBF reductio st by focusing on kernel execution!

rcutorture majors in this

Increase Workload Intensity: Kernel

- Most produc systems major i erspace executi l)
  - 10% kerne
  - A few percent ke commonplace
- One-to-two s of m BF reductio st by focusing o el execution!

Anti-Heisenbug: Increase Intensity!!!

rcutortu ors in this

Increase Workload Intensity: Kernel

- Special case of testing suspicious subsystems in isolation
- Configure application to beat up kernel
- Run kernel portion of workload from traces taken from application
- Increase CPUs, memory, I/O, ...

Increase Workload Intensity: Kernel

- Special case of testing suspicious subsyste in isolation
- Configure applicati ernel
- Run kernel po rkload from traces taken from ation
- In e CPUs, memory, I/O, ...

Ask yourself:

What has caused trouble in the past?

Increase Workload Intensity: Kernel

- pecial cas of testing suspicious subsyste
  i n
- Configure ernel
- Run kernel po traces
  taken from ation
- In e CPUs, memory, I/O, ...

Look for and promote trouble!!!

Anti-Heisenbug:

As elf:

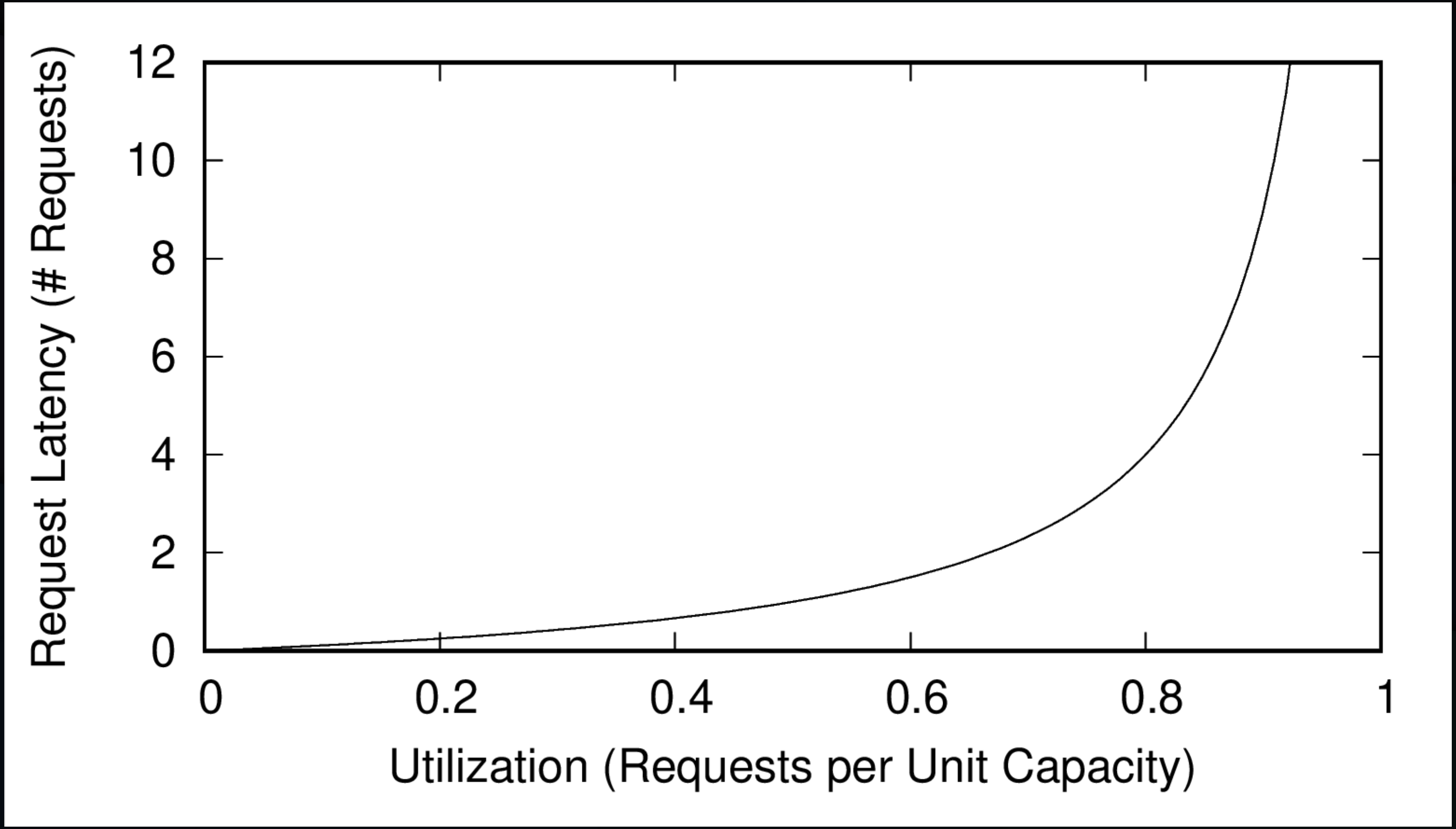
What has cause in the past?

Workload-Intensity Caution: Latency

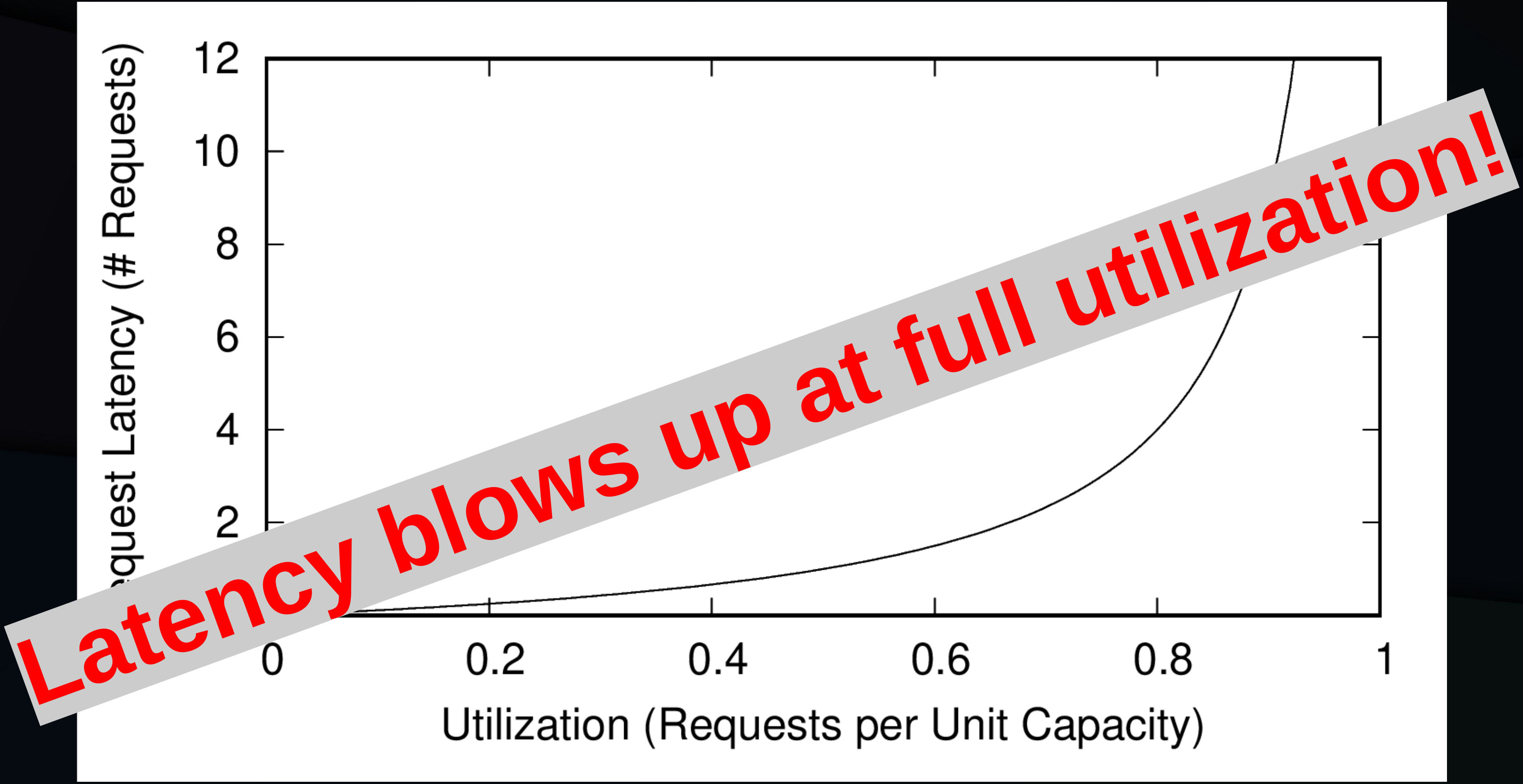
- Increasing load normally increases delays
  - Scheduler queueing
  - Lock contention
  - Memory contention

Theoretical Latency: M/M/1 Queue

Theoretical Latency: M/M/1 Queue

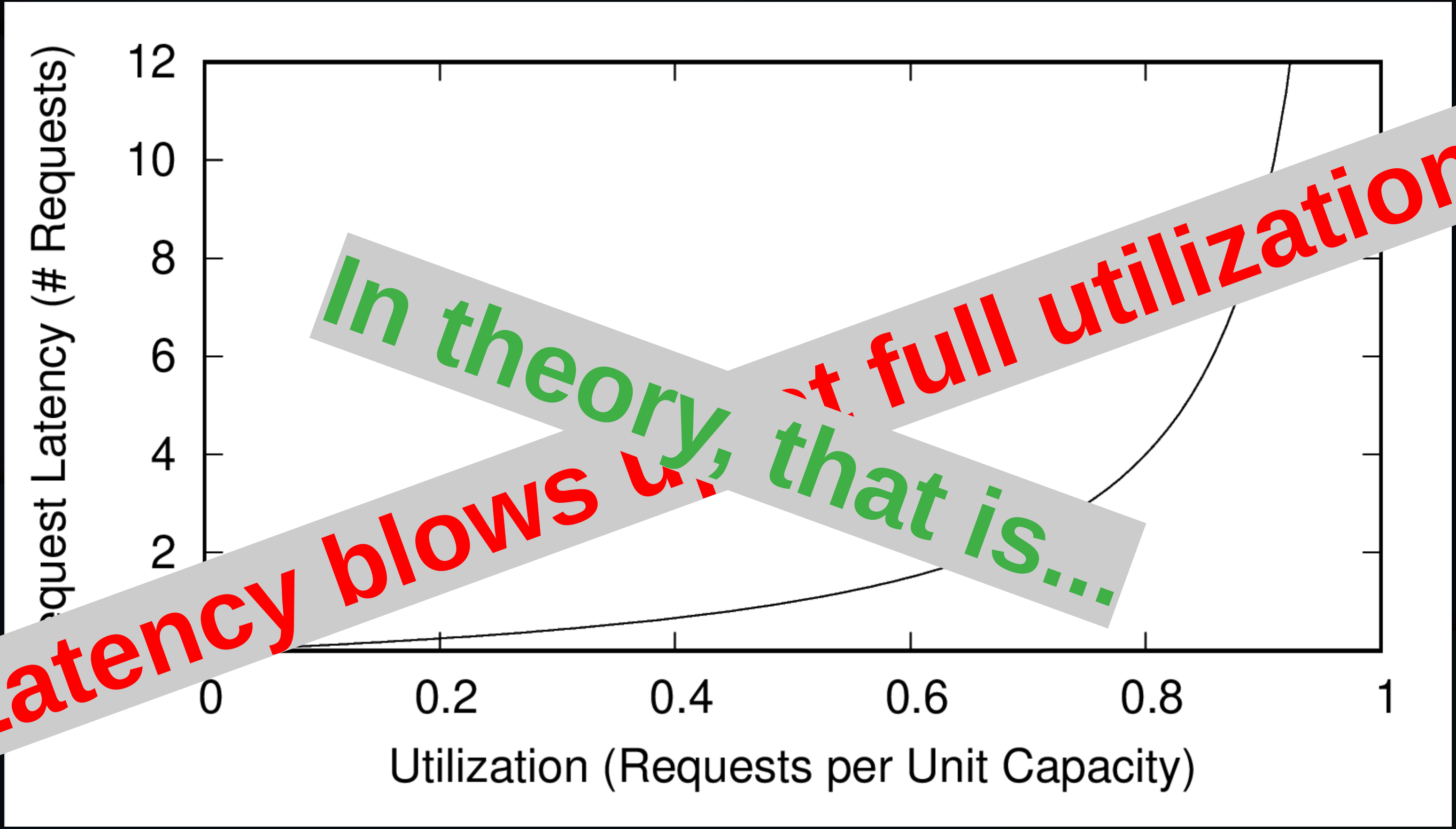


Theoretical Latency: M/M/1 Queue



Latency blows up at full utilization!

Theoretical Latency: M/M/1 Queue

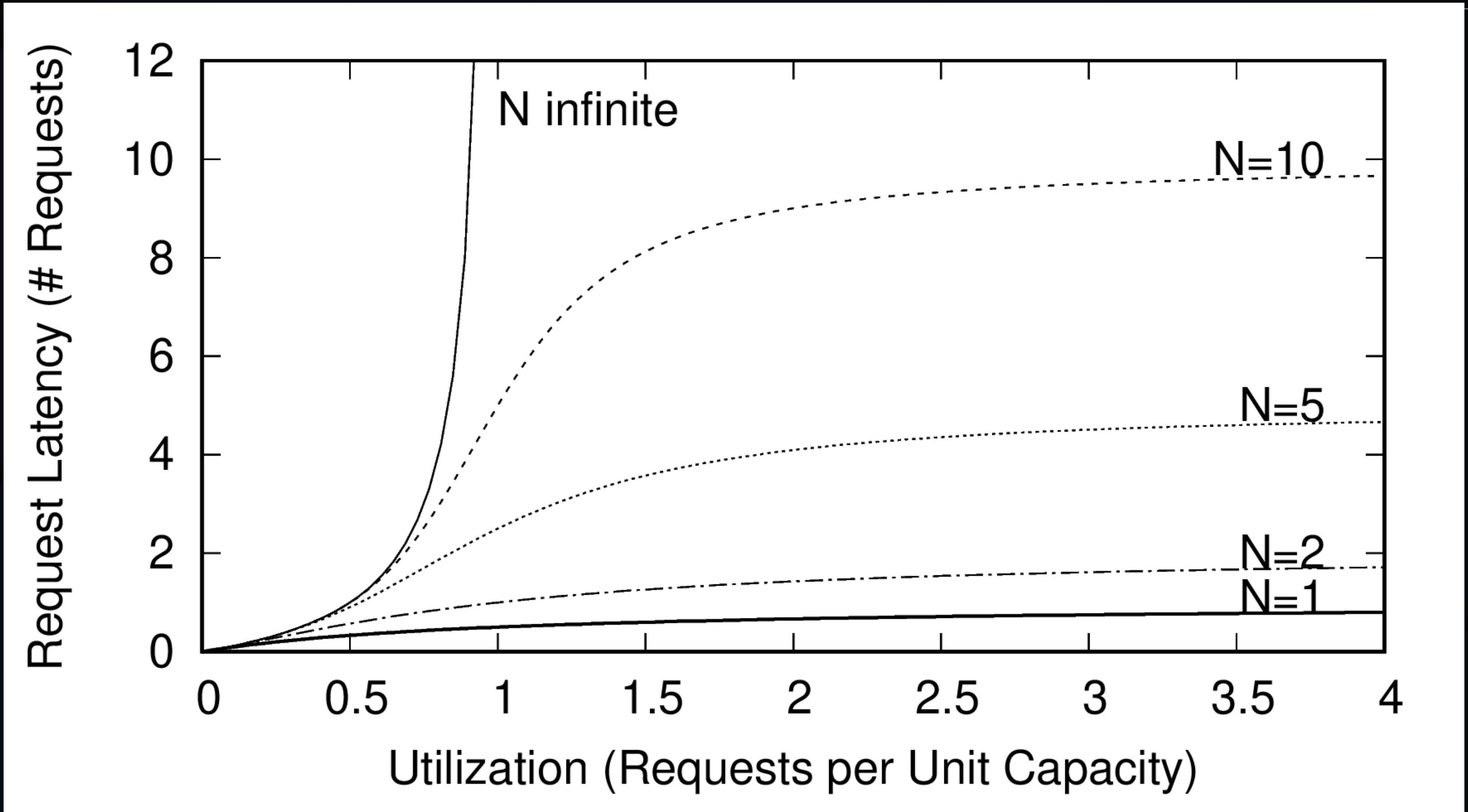


Latency blows up at full utilization!

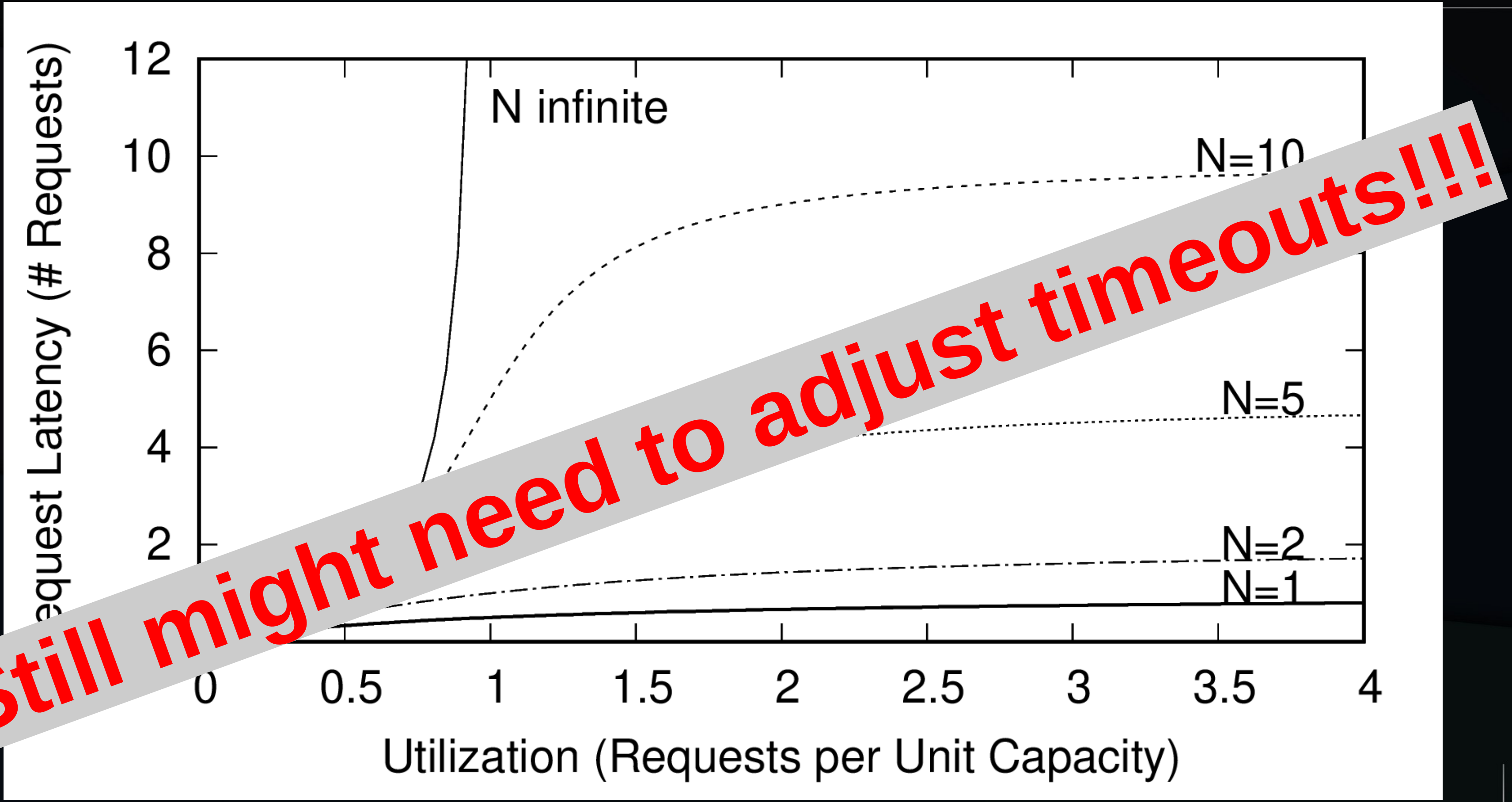
In theory, that is...

Queues Are Finite!!! M/M/1/k Queue

Latency From Finite Queueing



Latency From Finite Queueing



Still might need to adjust timeouts!!!

Inject Strategic Delays

Inject Strategic Delays

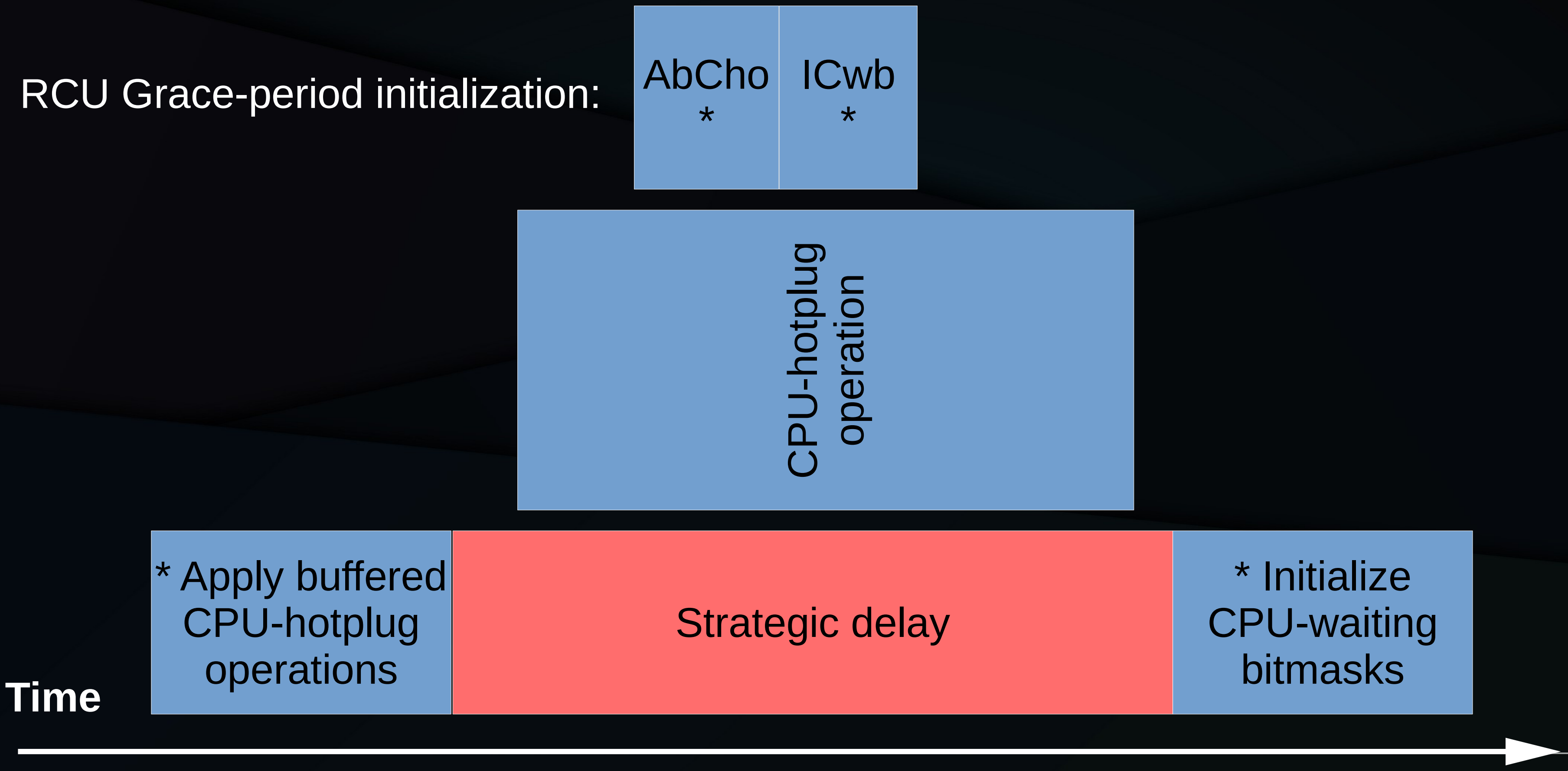
- Intensify one part of the workload by de-intensifying another
- Examples:
  - Running on multi-socket systems injects cache-miss latencies *
  - rcutorture injects delays during grace-period initialization to promote races with CPU-hotplug operations
  - Old days: Run CPUs at different speeds

* https://paulmck.livejournal.com/62071.html

RCU CPU-Hotplug Strategic Delays

- RCU need not wait on offline CPUs
  - Nor on CPUs that online after grace period start
    - Though it is OK to wait on them
  - But RCU does need to be clear on whether or not it needs to wait on a given CPU
  - And RCU does need to "keep its own books" on which CPUs are online

RCU CPU-Hotplug Strategic Delays


RCU Grace-period initialization:
AbCho *
ICwb *
CPU-hotplug operation
* Apply buffered CPU-hotplug operations
Strategic delay
* Initialize CPU-waiting bitmasks
Time

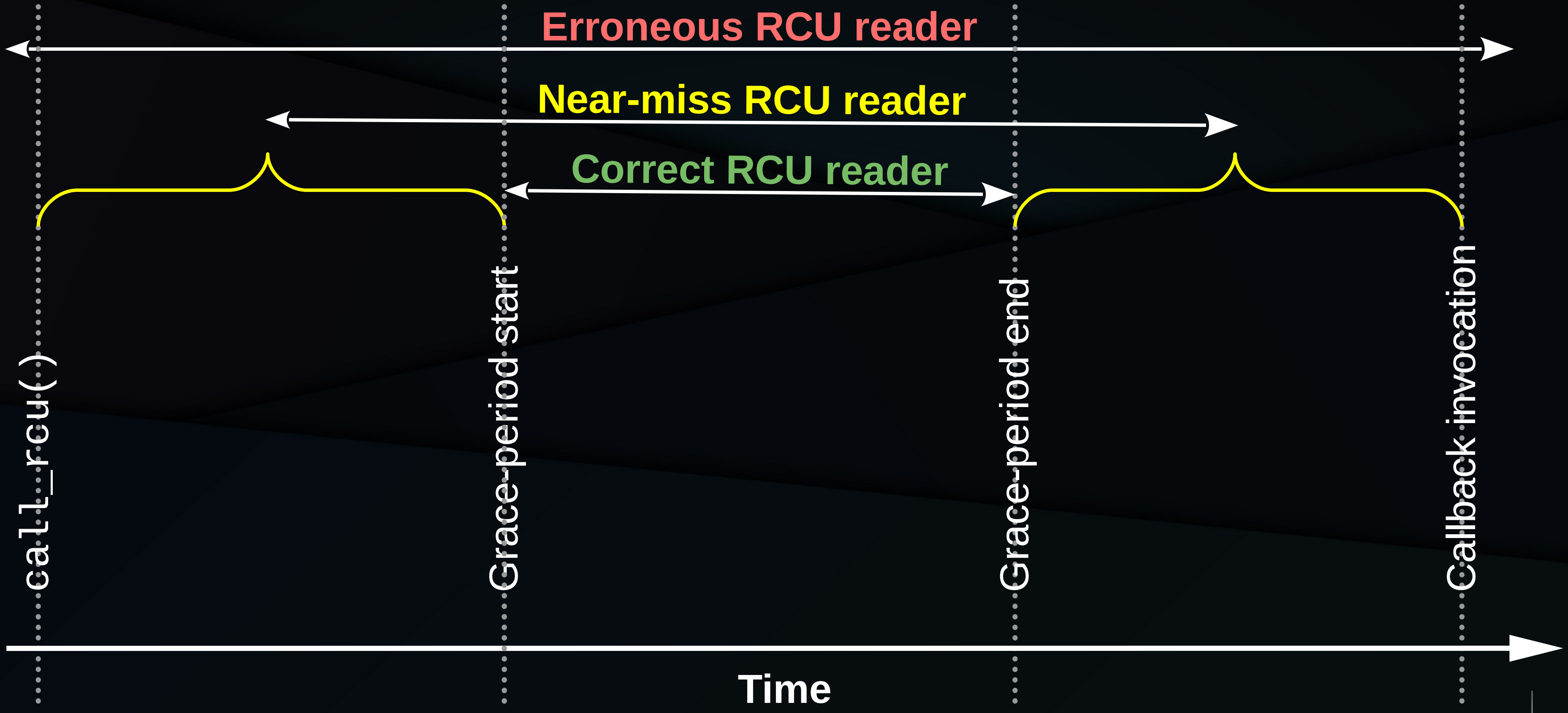
Count Near Misses

Count Near Misses

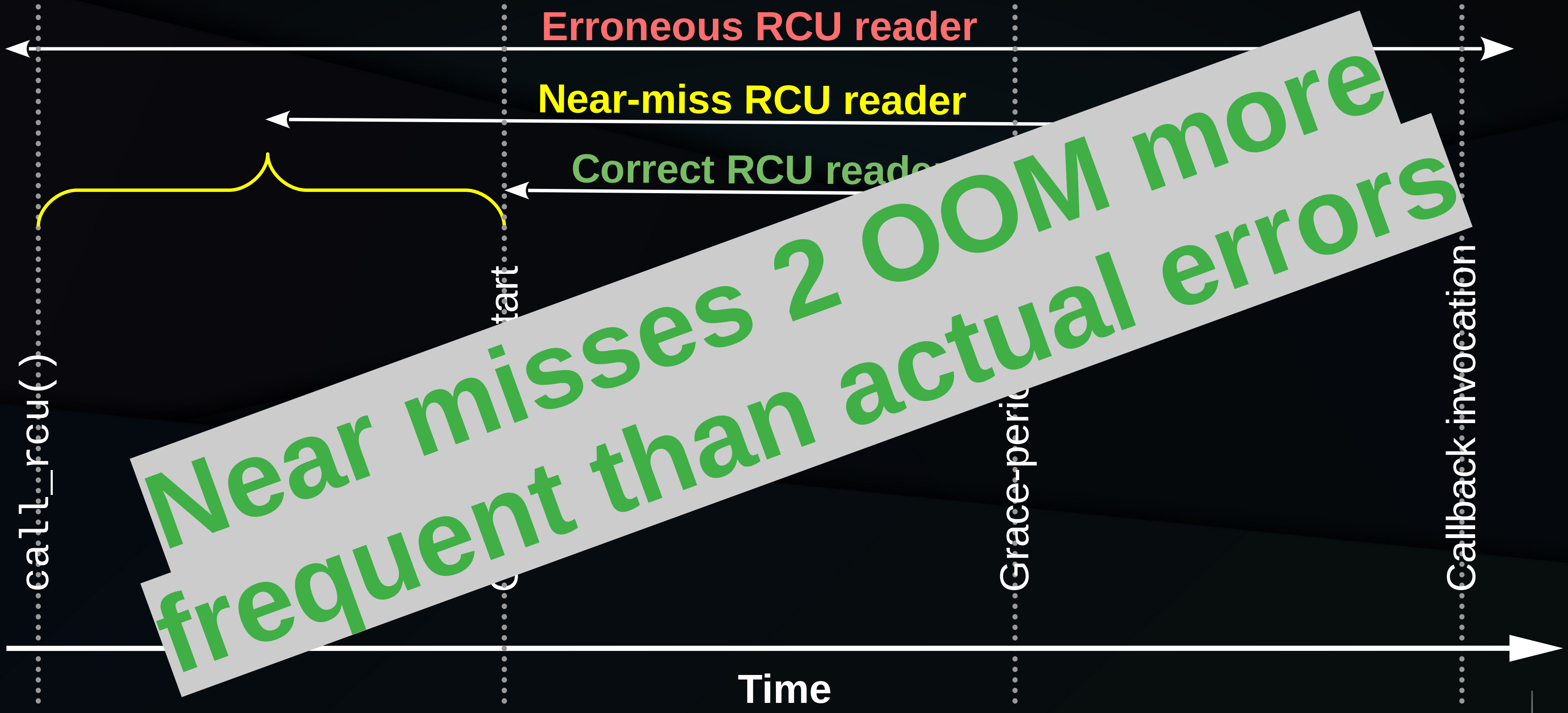
- USA FAA requires reporting of near misses *
  - Higher probability of near miss than of collision
- Near misses can help hunting heisenbugs
  - More quickly evaluate commits, configurations, and effectiveness of other anti-heisenbugs
  - Especially helpful when verifying fixes

* https://www.faa.gov/air_traffic/publications/atpubs/aim_html/chap7_section_7.html

Count Near Misses: RCU Example

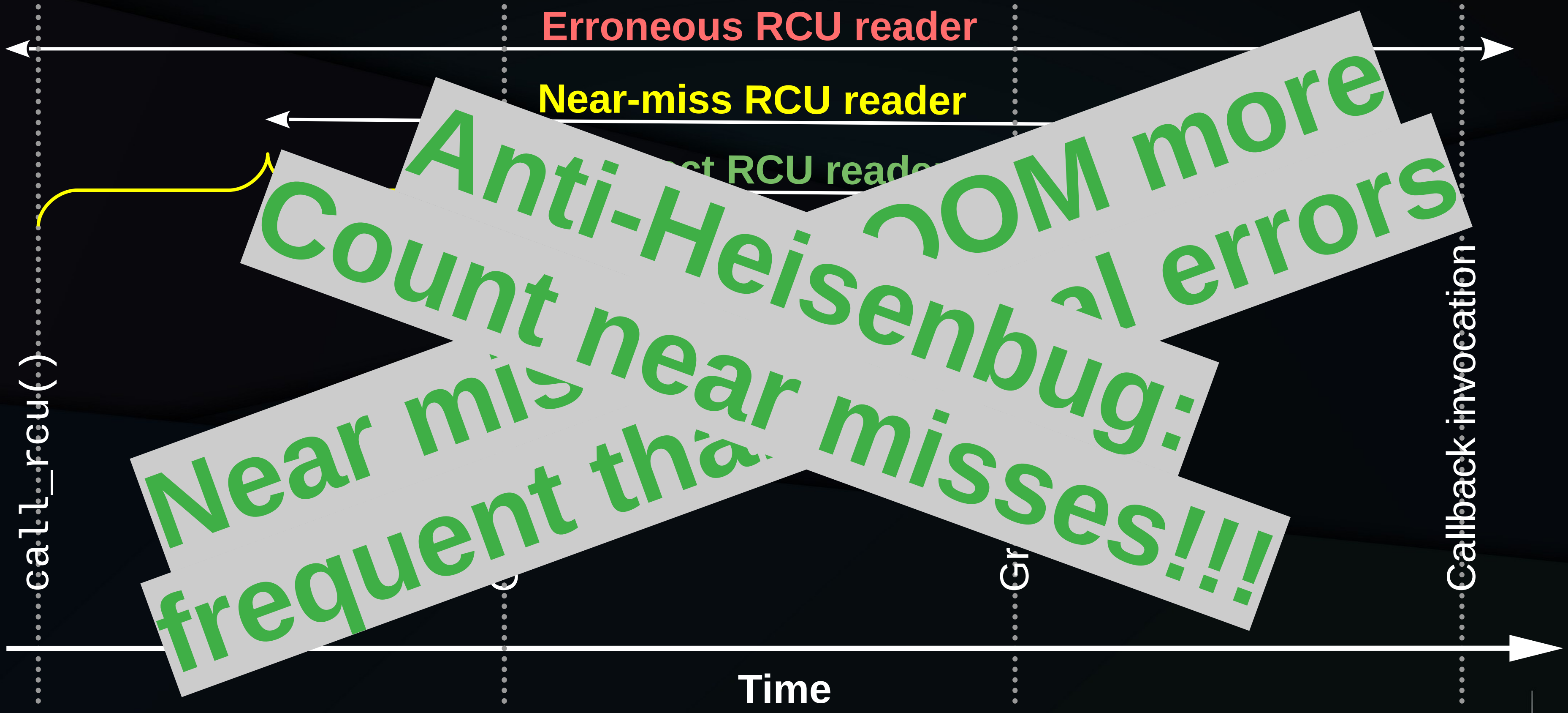


Count Near Misses: RCU Example



Near misses 2 OOM more frequent than actual errors

Count Near Misses: RCU Example



Near mis OOM more
frequent tha l errors

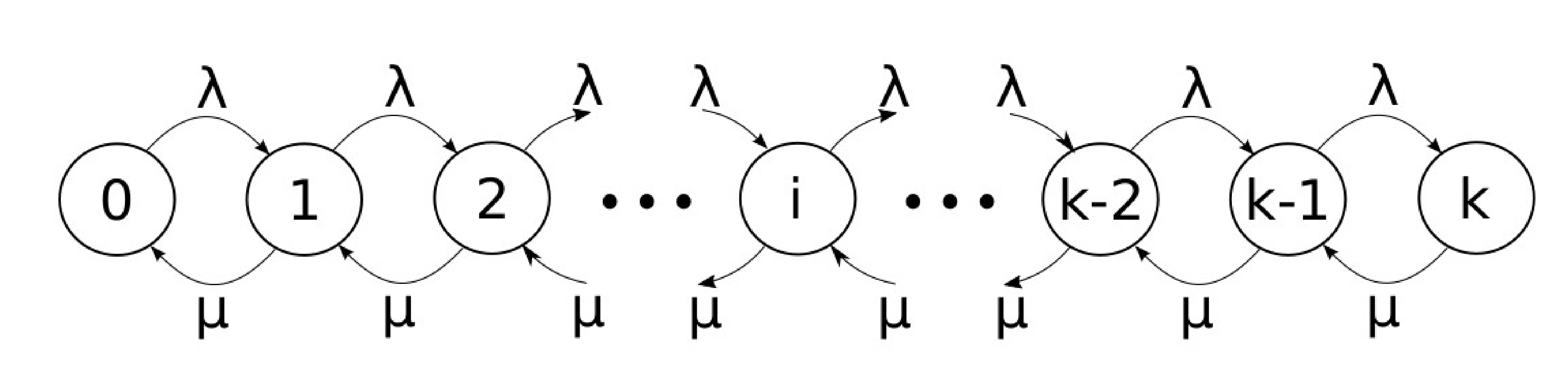
Anti-Heisenbug:
Count near misses!!!

Make Rare Events Happen Frequently

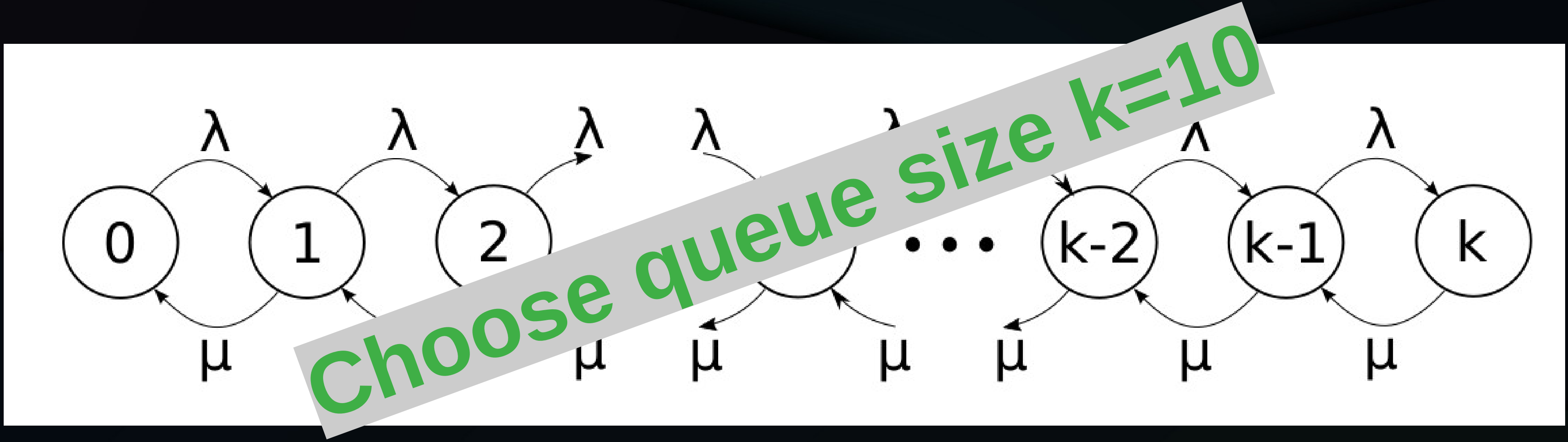
Make Rare Events Happen Frequently

- Utilization, redux
- Force rare error conditions
- Force rare slowpath execution
- Add delays to race-prone code
- “The nuclear option”

Utilization and Rare Events

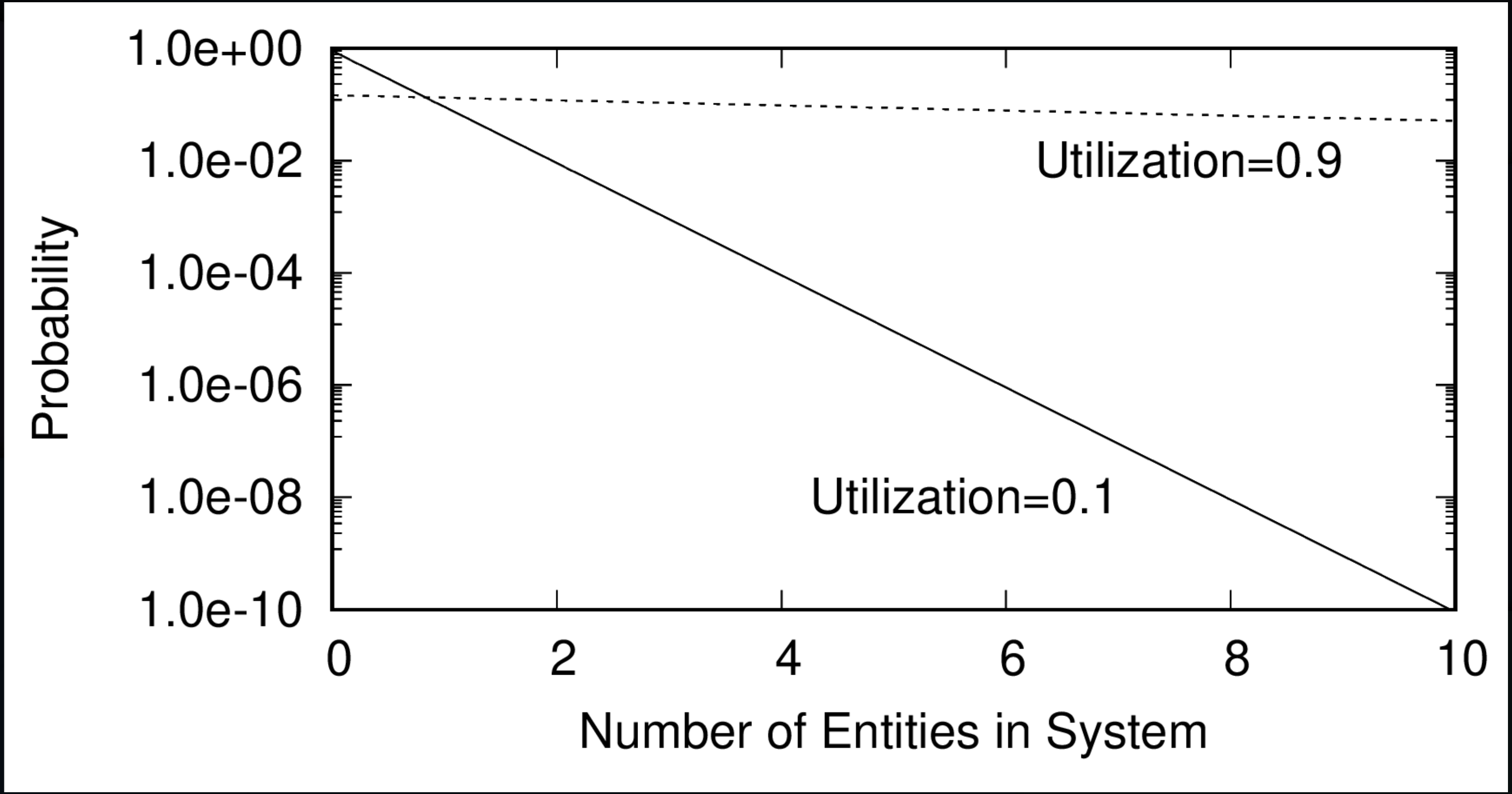


Utilization and Rare Events

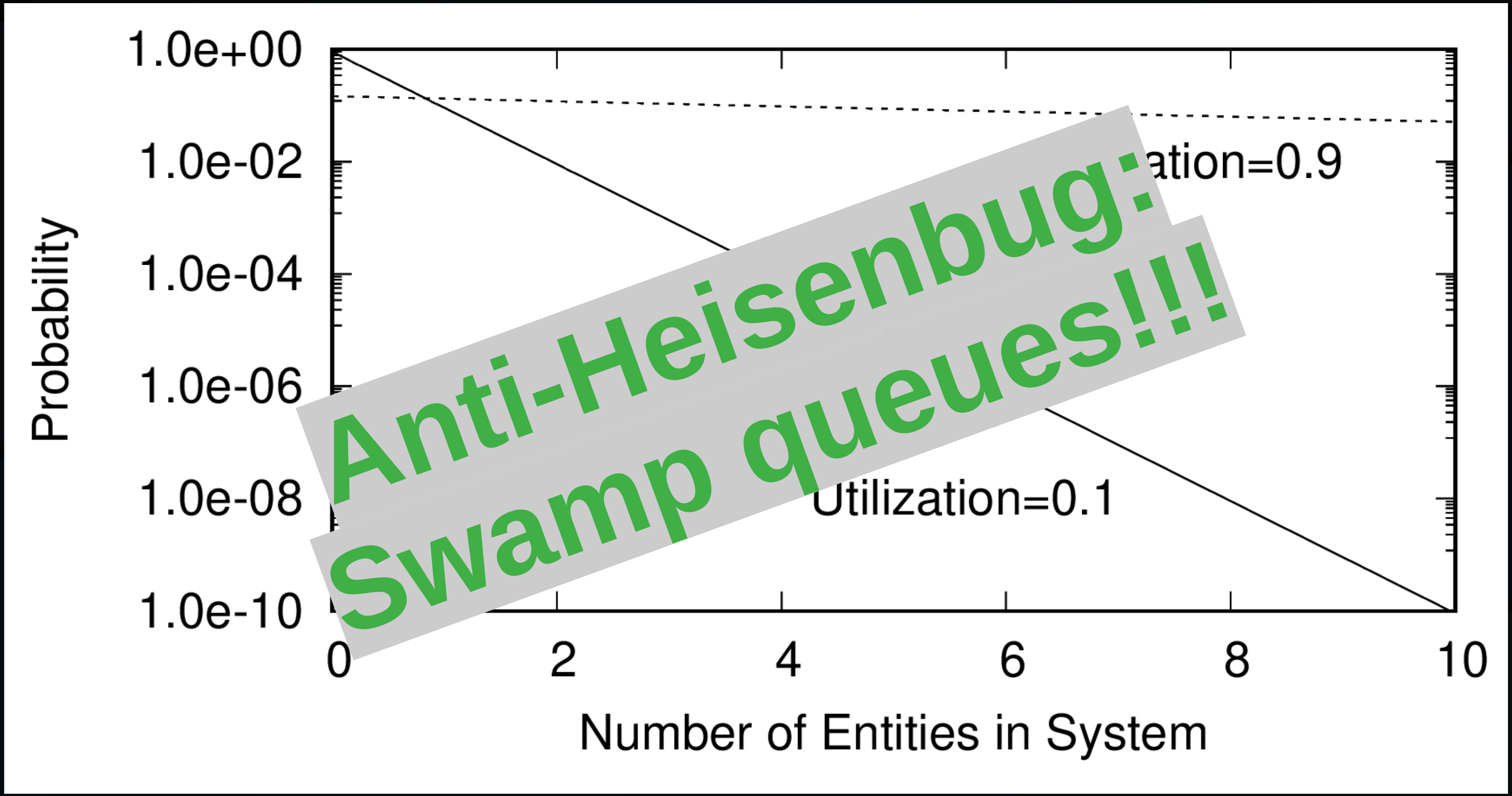


Choose queue size k=10

Utilization and Rarity of Events

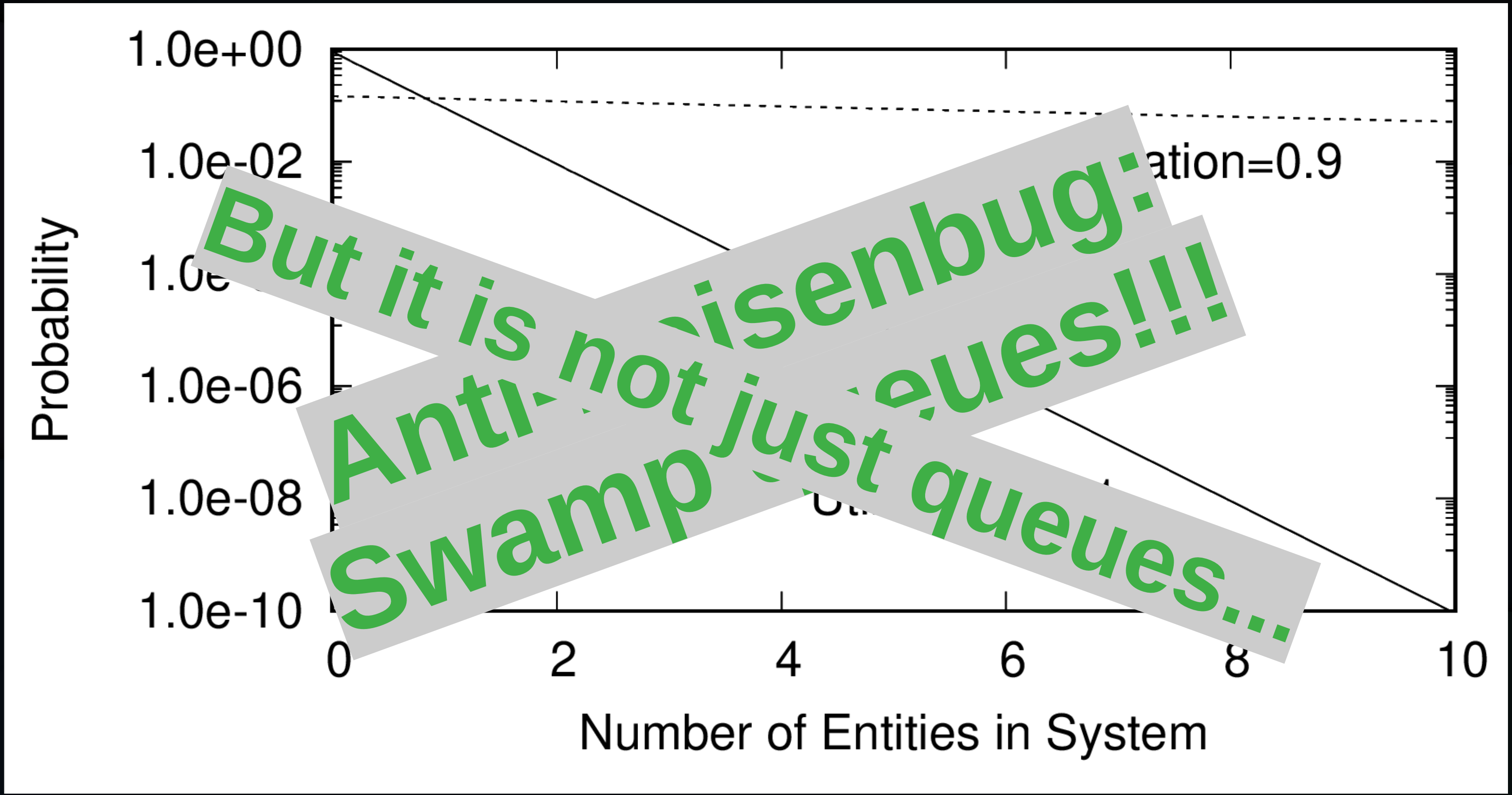


Utilization and Rarity of Events



Anti-Heisenbug: Swamp queues!!!

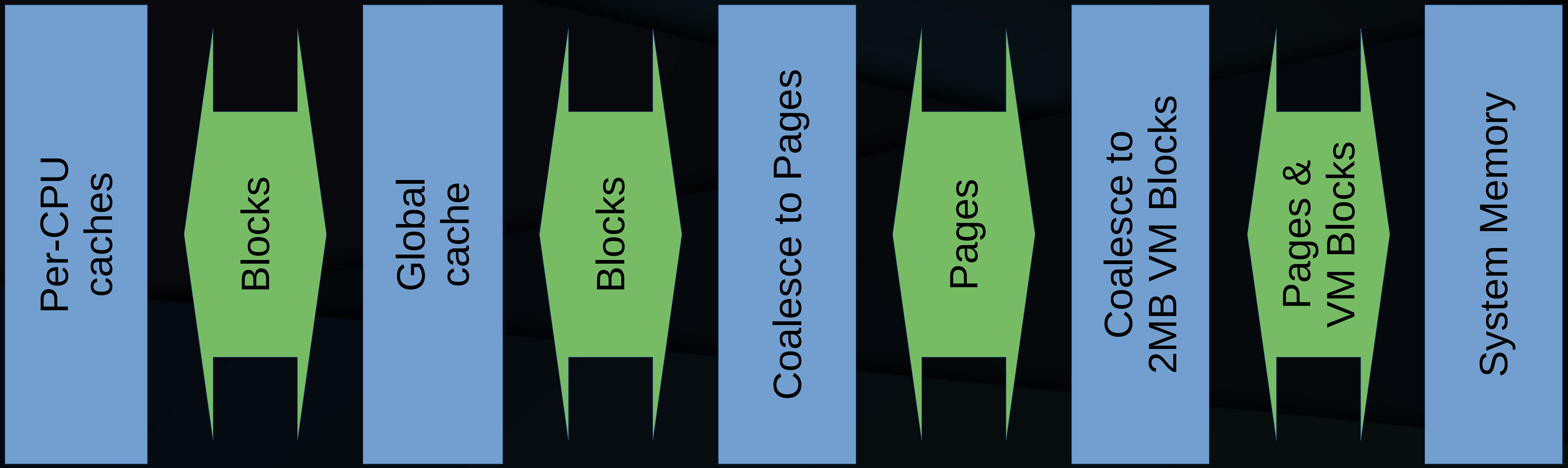
Utilization and Rarity of Events



But it is not just queues...

Anti-Heisenbug:
Swamp Queues!!!

DYNIX/ptx Memory Allocator ca. 1993



Per-CPU caches

Blocks

Global cache

Blocks

Coalesce to Pages

Pages

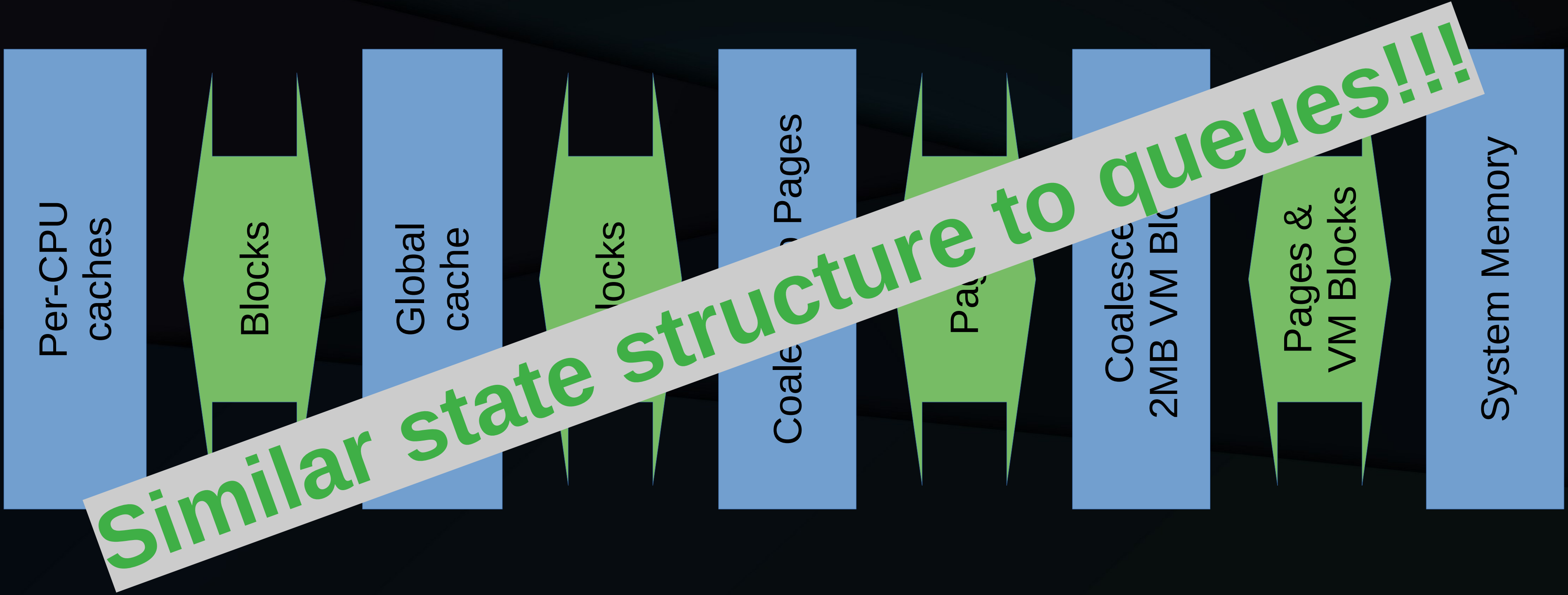
Coalesce to 2MB VM Blocks

Pages & VM Blocks

System Memory

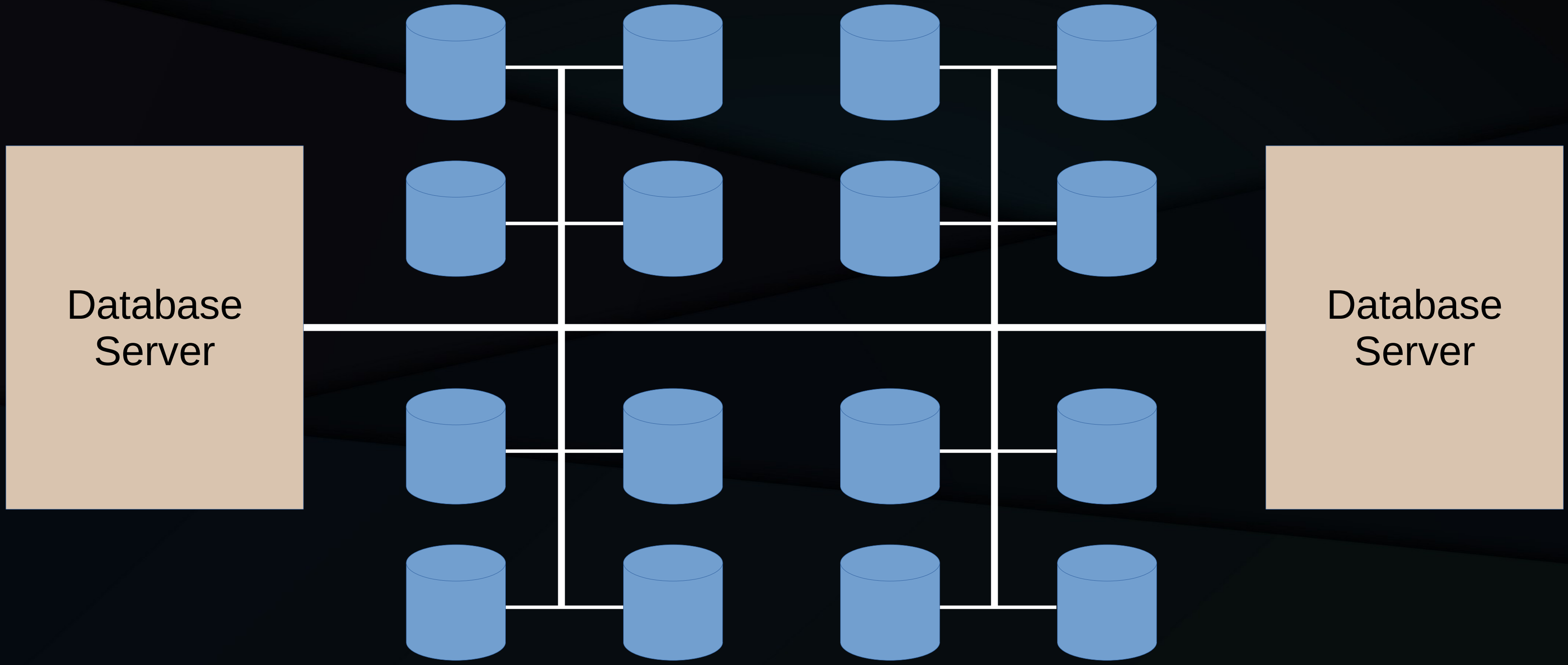
Later on added per-NUMA-node caches

DYNIX/ptx Memory Allocator ca. 1993

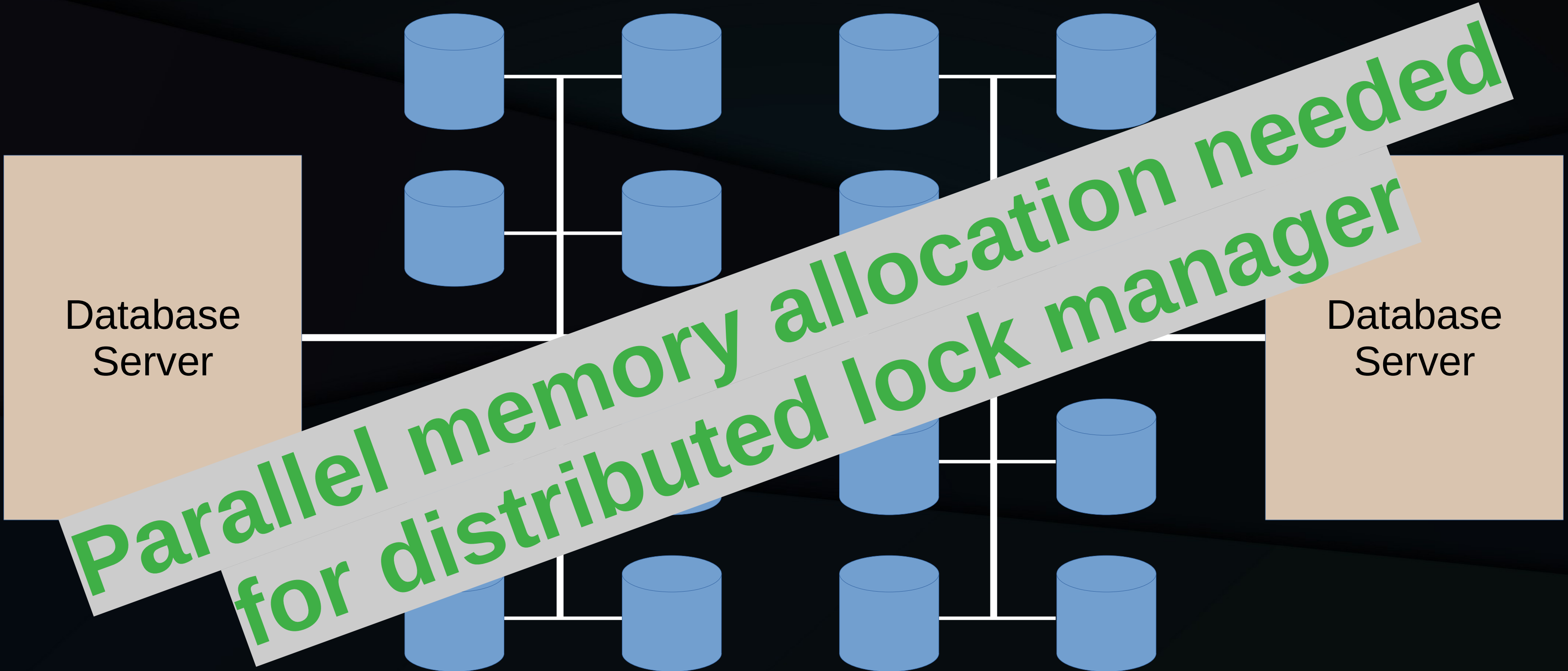


Later on added per-NUMA-node caches

Shared Disks For Availability Win!!!

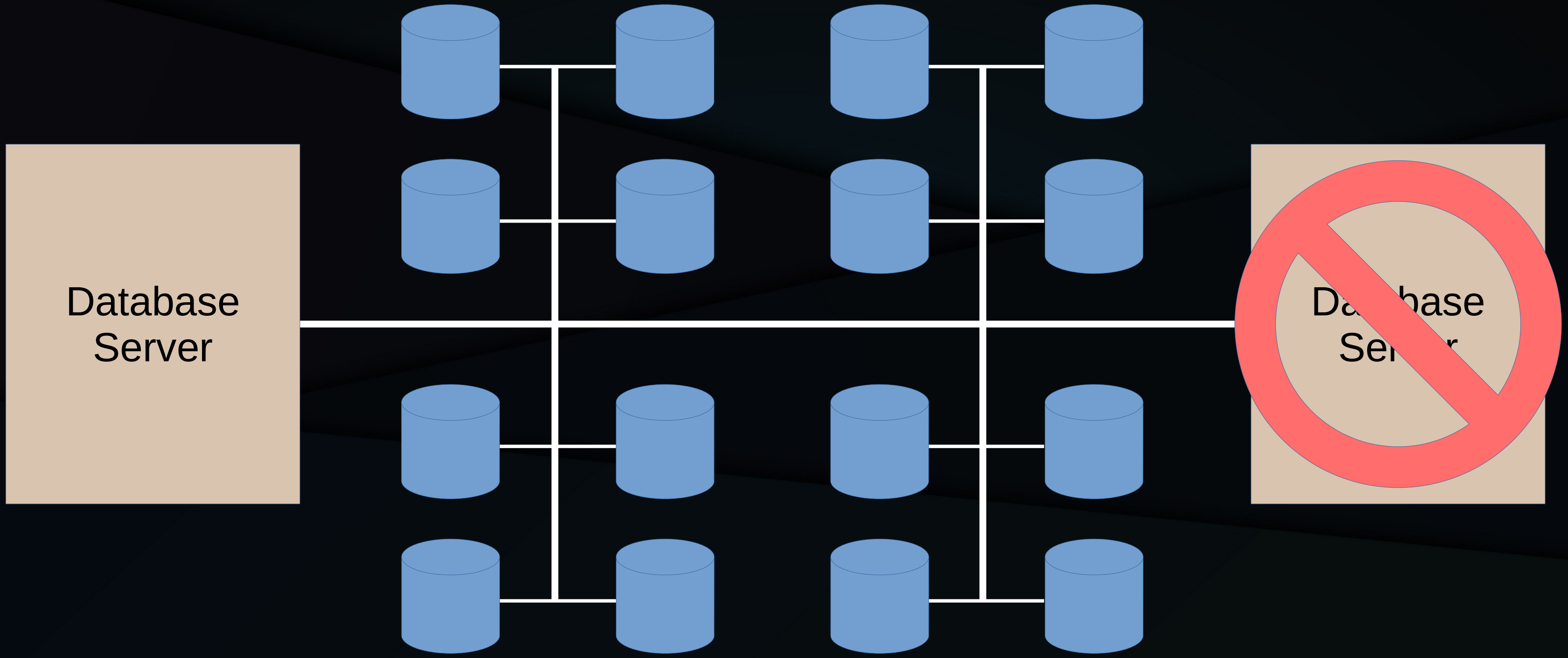


Shared Disks For Availability Win!!!



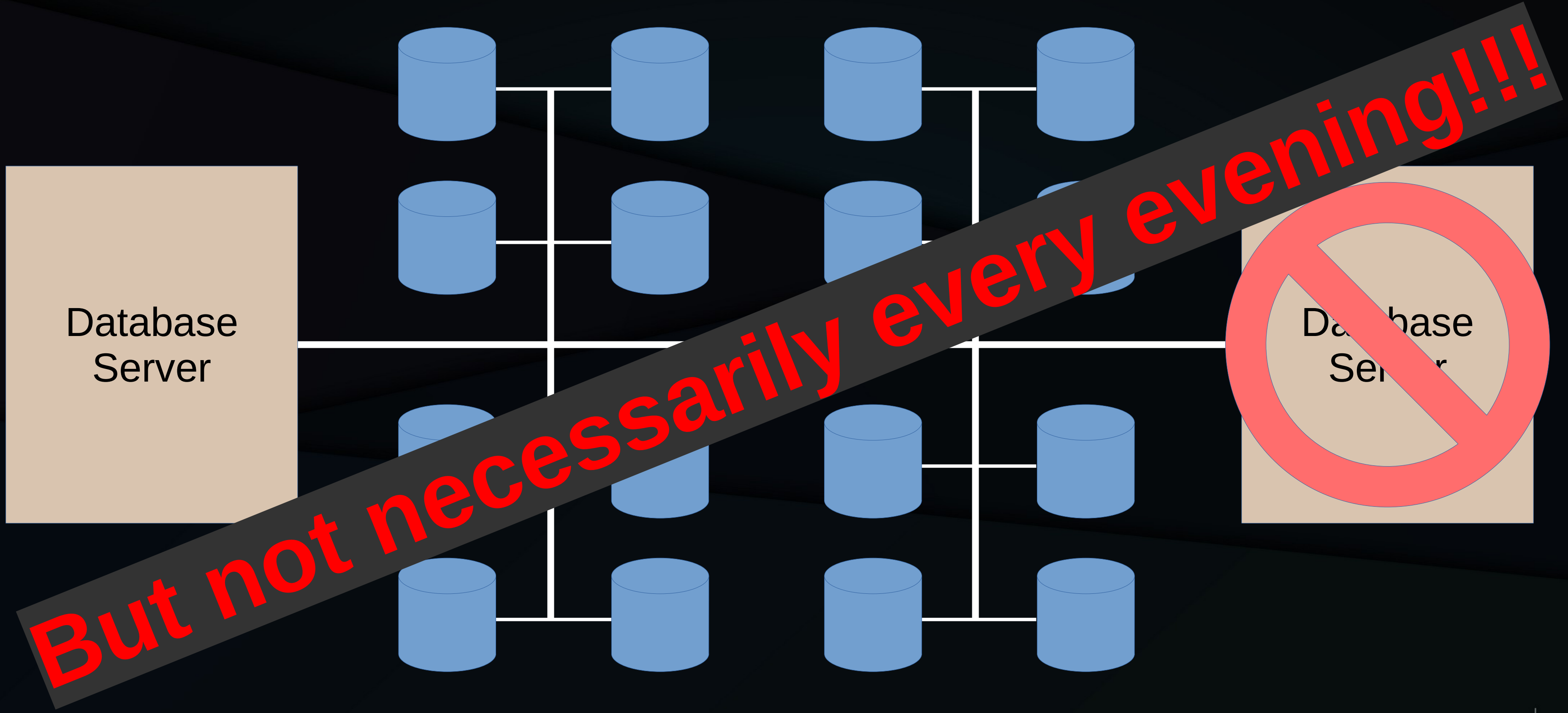
Parallel memory allocation needed for distributed lock manager

Shared Disks For Availability Win!!!



All data is still accessible!!! Of course, sites should test this frequently...

Shared Disks For Availability Win!!!

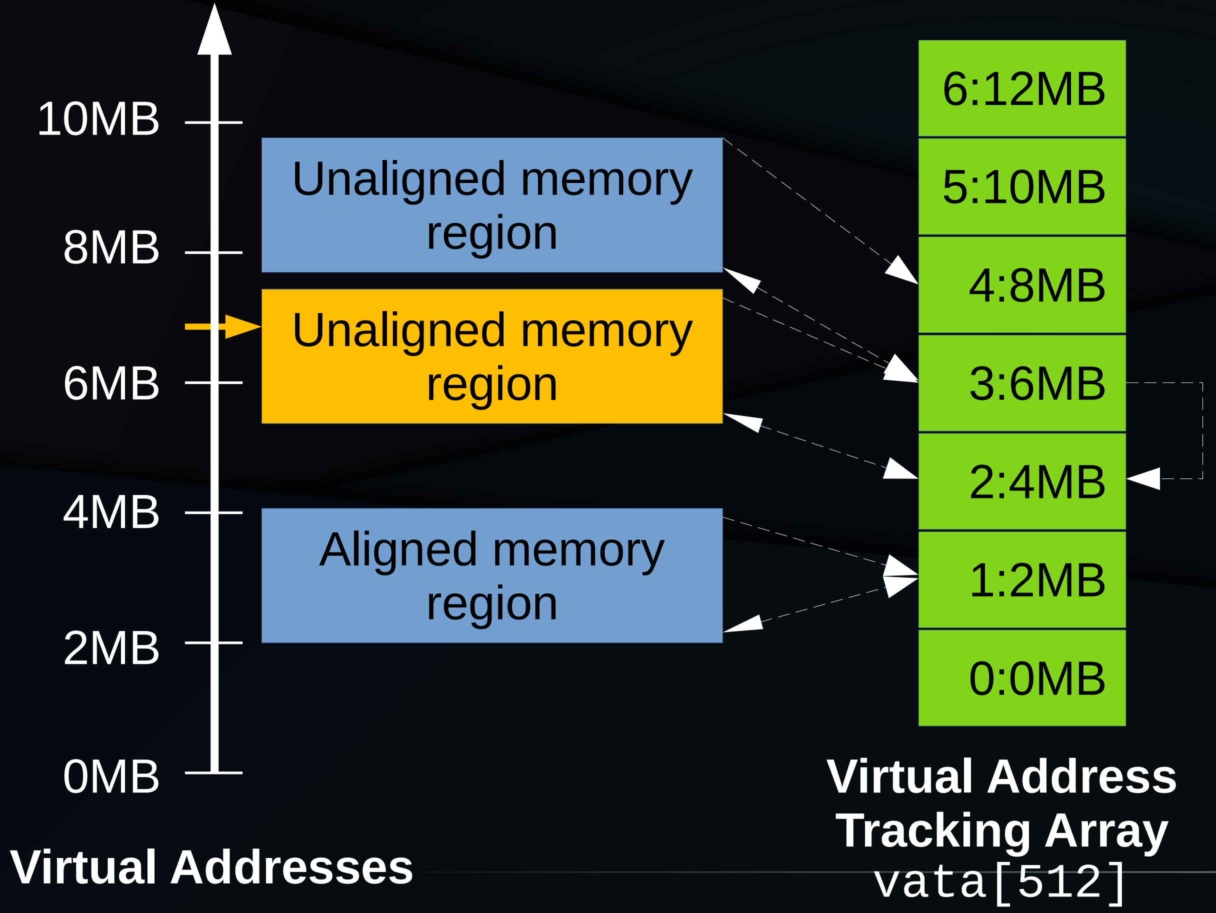


All data is still accessible!!! Of course, sites should test this frequently...

Chaos-Monkey Challenges

- Crash dump was a complete disaster area
  - No hints for on-site debugging instrumentation
- Eventually found test case: 5-27-hour MTBF
  - But need week-long test for any alleged fix!!!

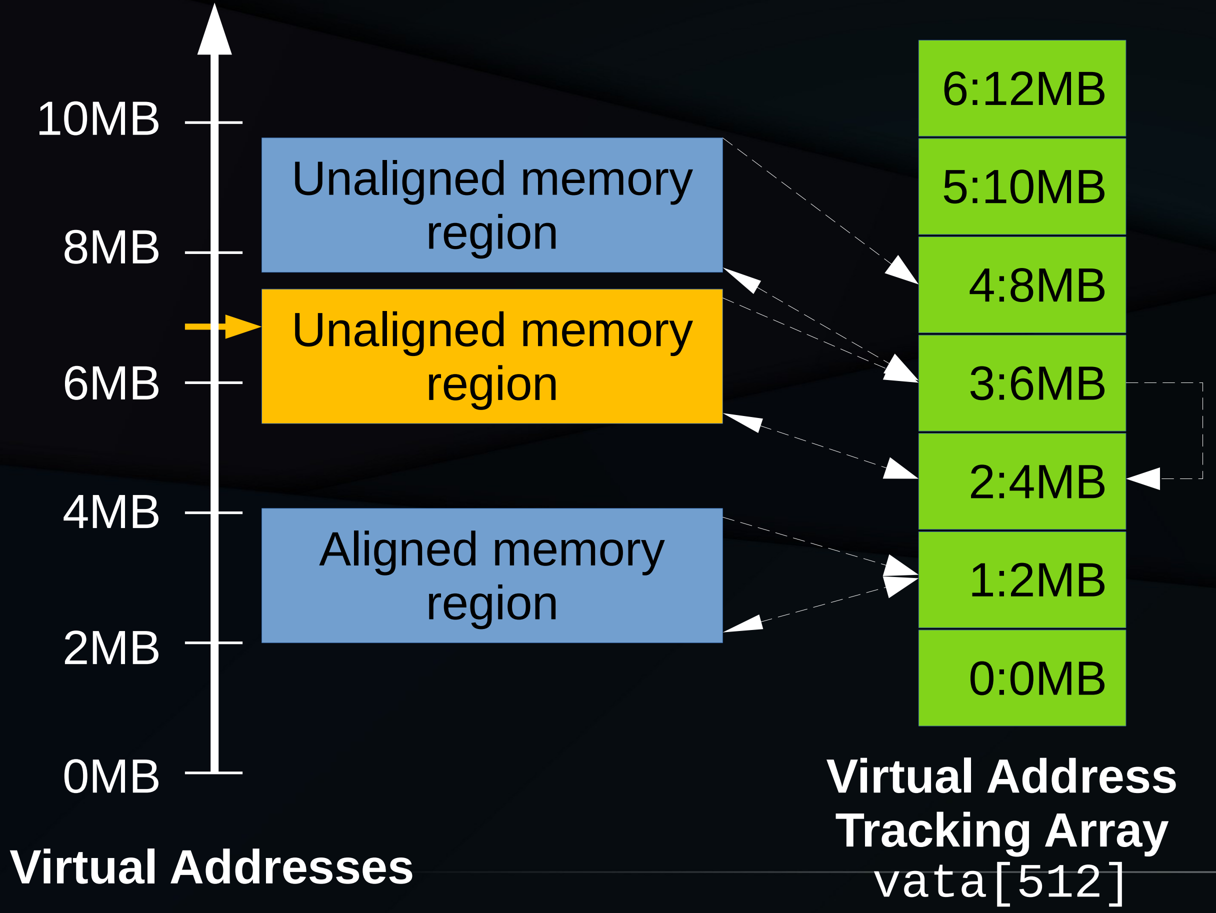
Hint From Stack Trace



```
int idx = vadr / (2 * MB);
void *vta;

vta = vata[idx];
if (!vta || vadr < vta)
    vta = vta[idx – 1];
```

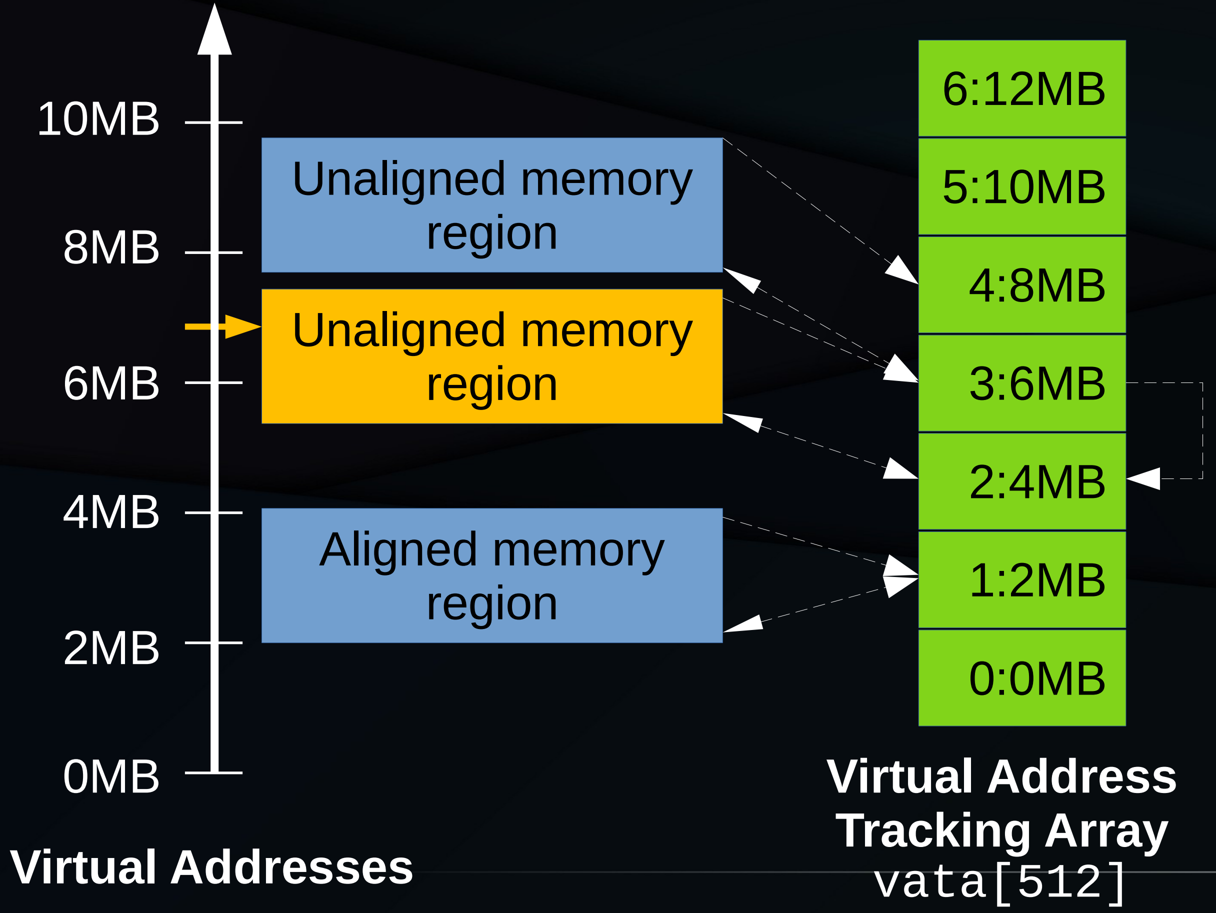
Hint From Stack Trace: Compiler Fun



```
int idx = vadr / (2 * MB);
void *vta;

vta = vata[idx];
if (!vata[idx] ||
     vadr < vata[idx])
    vta = vta[idx – 1];
```

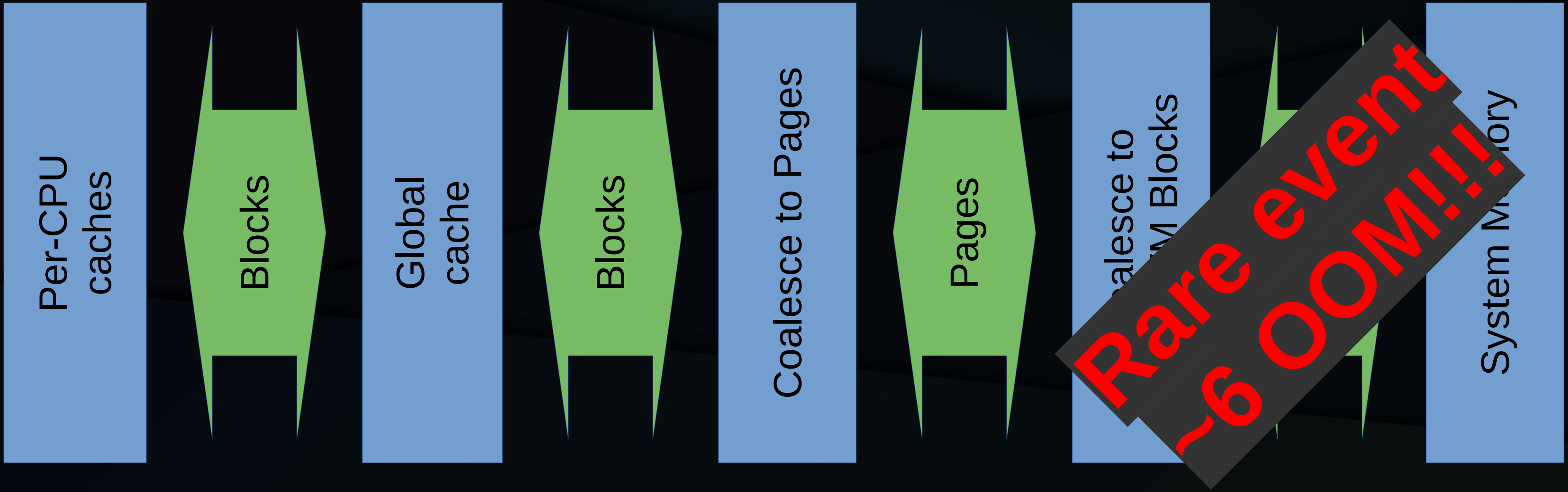
Hint From Stack Trace: Compiler Fun



```
int idx = vadr / (2 * MB);
void *vta;

vta = READ_ONCE(vata[idx]);
if (!vta || vadr < vta)
    vta = vta[idx – 1];
```

DYNIX/ptx Memory Allocator ca. 1993



Per-CPU caches

Blocks

Global cache

Blocks

Coalesce to Pages

Pages

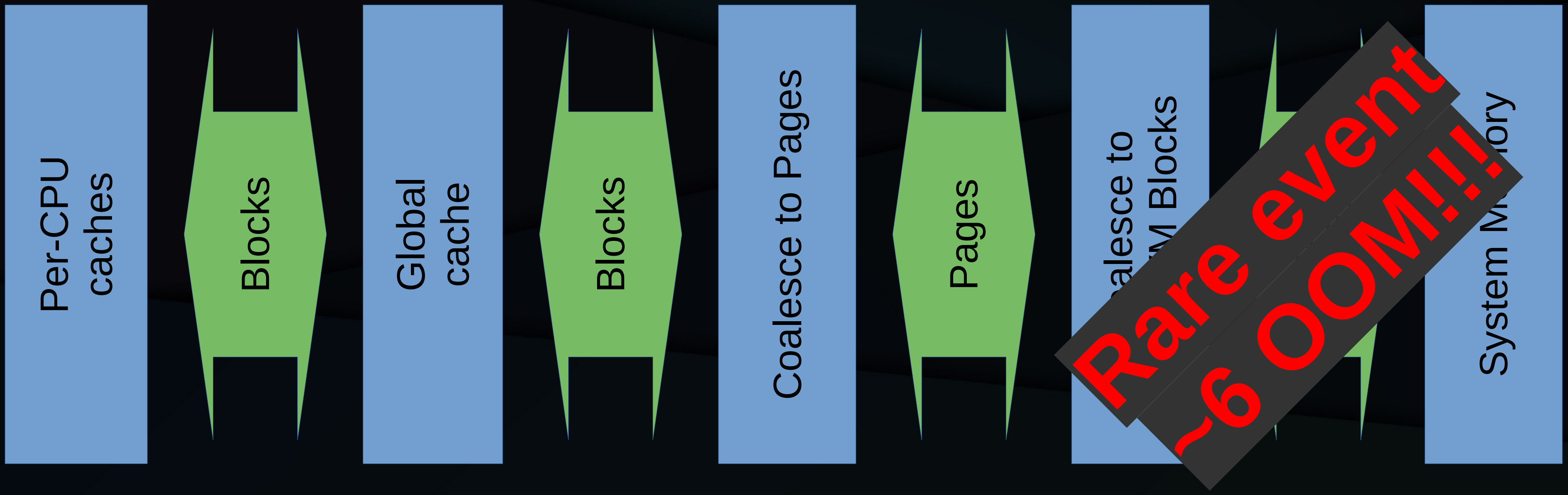
alesce to M Blocks

System M ory

Rare event

~6 OOM!!!

DYNIX/ptx Memory Allocator ca. 1993



Rare event
~6 OOM!!!

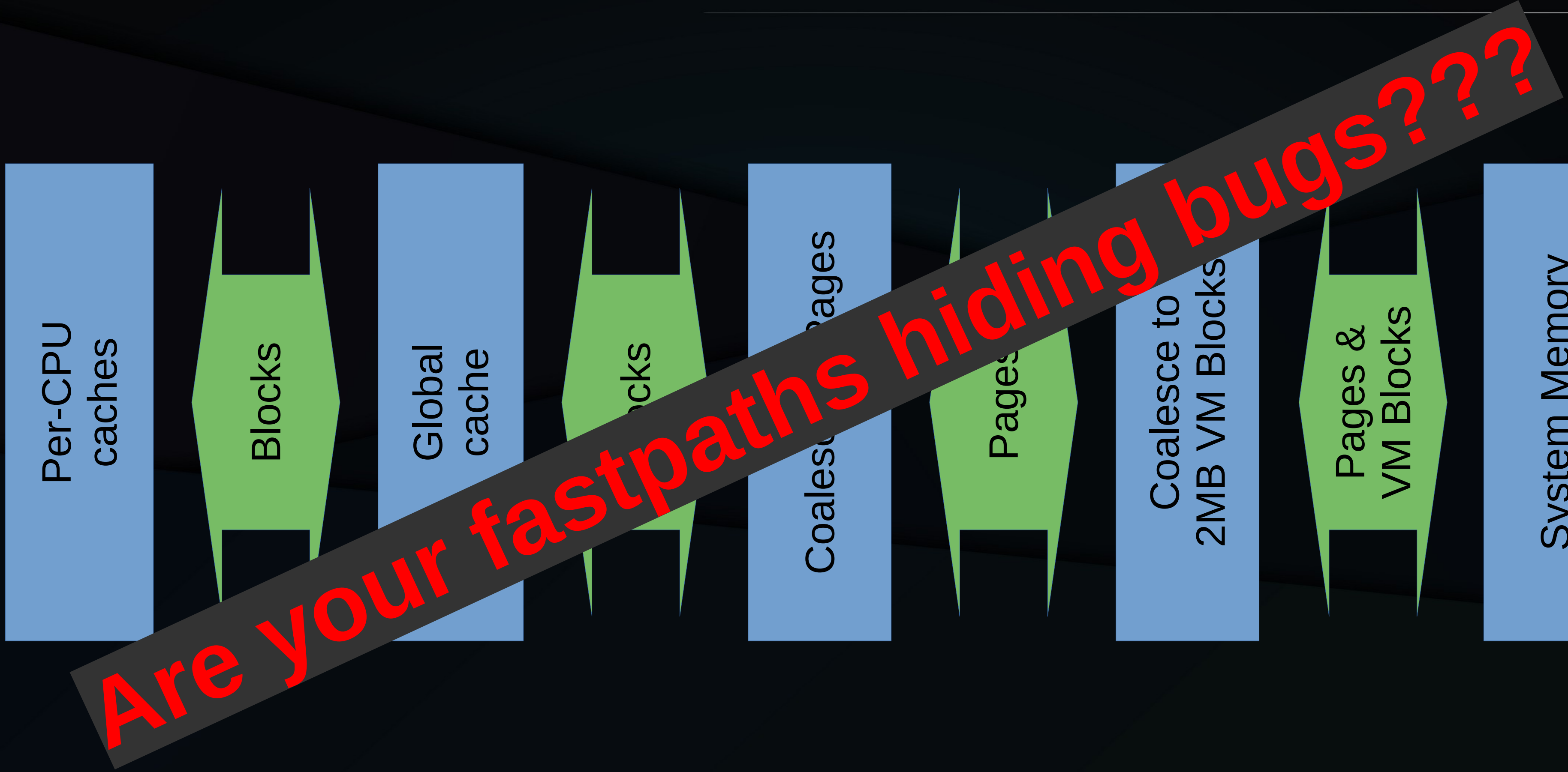
Focused test reduced MTBF to 12 minutes, 1-2 OOM better than stress test

Focused Test vs. Stress Test

	Focused Test	Stress test	Existing testing
MTBF	12 minutes	5 to 27 hours	Infinite?
Basis	Exact bug	Customer workload	Past experience
Hardware	Minimal	A few large systems	Many systems
Development	Day or two	Few person-weeks	Large over years
Applicability	Narrow *	Modest	Wide
Impact	Profound contention	Heavy load	Wide variation

* No I/O, few tasks, modest stress on scheduler, almost no userspace

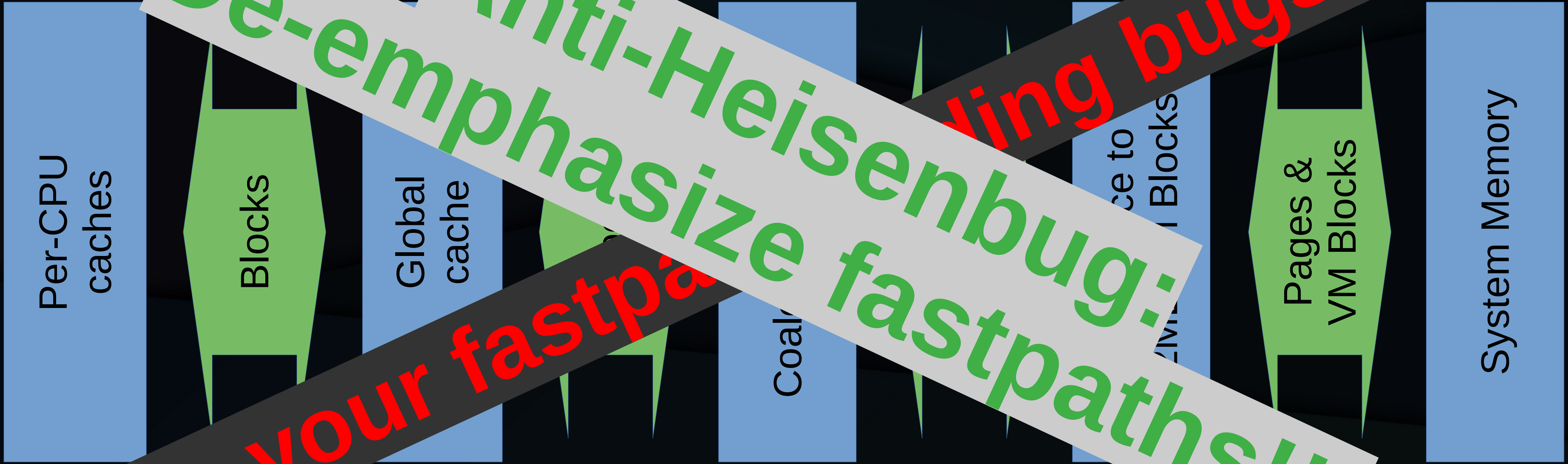
DYNIX/ptx Memory Allocator ca. 1993



Are your fastpaths hiding bugs???

Focused test reduced MTBF to 12 minutes, 1-2 OOM better than stress test

DYNIX/ptx Memory Allocator ca. 1993



Anti-Heisenbug:

De-emphasize fastpaths!!!

Are your fastpa

ding bugs???

Focused test reduced MTBF to 12 minutes, 1-2 OOM better than stress test

DYNIX/ptx Memory Allocator ca. 1993

Per-CPU
Coal
e to
locks
P &
VM ks
System Memory

De-emphas… fastpaths!!!

Anti-Heisen… g!

Are your fastp… ing bugs???

Performance and scalability???

Focused test reduced MTBF to 12 minutes, 1-2 OOM better than stress test

Safely Disabling Fastpaths: Options

- Run on small systems
  - Four-CPU guest OSes for the win!
- Accept massive contention
- Run code developed for old systems on newer highly integrated systems

Hardware Latency Trends

Year	Sockets	CPUs	CAS Latency (ns)
2008	4	16	95.9
2017	1	56	101.9
2017	4	224	442.9
2022	2	224	147.0

Hardware Latency Trends

Year	Sockets		Latency (ns)
2008	4	6	95.9
2017	1		101.9
201		224	442.9
		224	147.0

Newer systems handle memory contention better

Fewer fastpaths? Larger systems? Decisions, decisions...

Overlapping RCU Readers

```
rcu_read_lock();
preempt_disable();
rcu_read_unlock();
local_irq_disable();
preempt_enable();
local_bh_disable();
local_irq_enable();
local_bh_enable();
```

Overlapping RCU Readers

```
rcu_read_lock();
preempt_disable();
rcu_read_unlock();
local_irq_disable();
preempt_enable()
local_bh
loc      _enabl
local_bh_enable();
```

Rare combination unless you are running rcutorture

Other Rare Events

- Transitions to and from RCU idle
- CPU hotplug operations (boot and suspend)
- RCU callback flooding
- Memory near-exhaustion
- Transparent hugepage split/coalescing
- And many many more...

Other Rare Events

- Transitions to and from RCU idle
- CPU hotplug operations (uspend)
- RCU callback flo
- Memory
- Transp ent hu oalescing
- And many ma

Combine rare events: multiplicative decreases in MTBF

Other Rare Events

- Transitions to … nd from RCU idl…
- CPU … (… uspend)
- RCU callba…
- Memory m…
- Transp… nt hu…
- And many ma…

Anti-Heisenbug:

Combine rare events!!!

…e events:

…tive

Comb… mult… …TBF

decrease…

Other Rare Events

- Transitions to nd from RCU idl
- CPU h s (uspend)
- Transp nt h
- And many ma

Anti- e events:

Con

Co

mu

decrease events!!!

g:

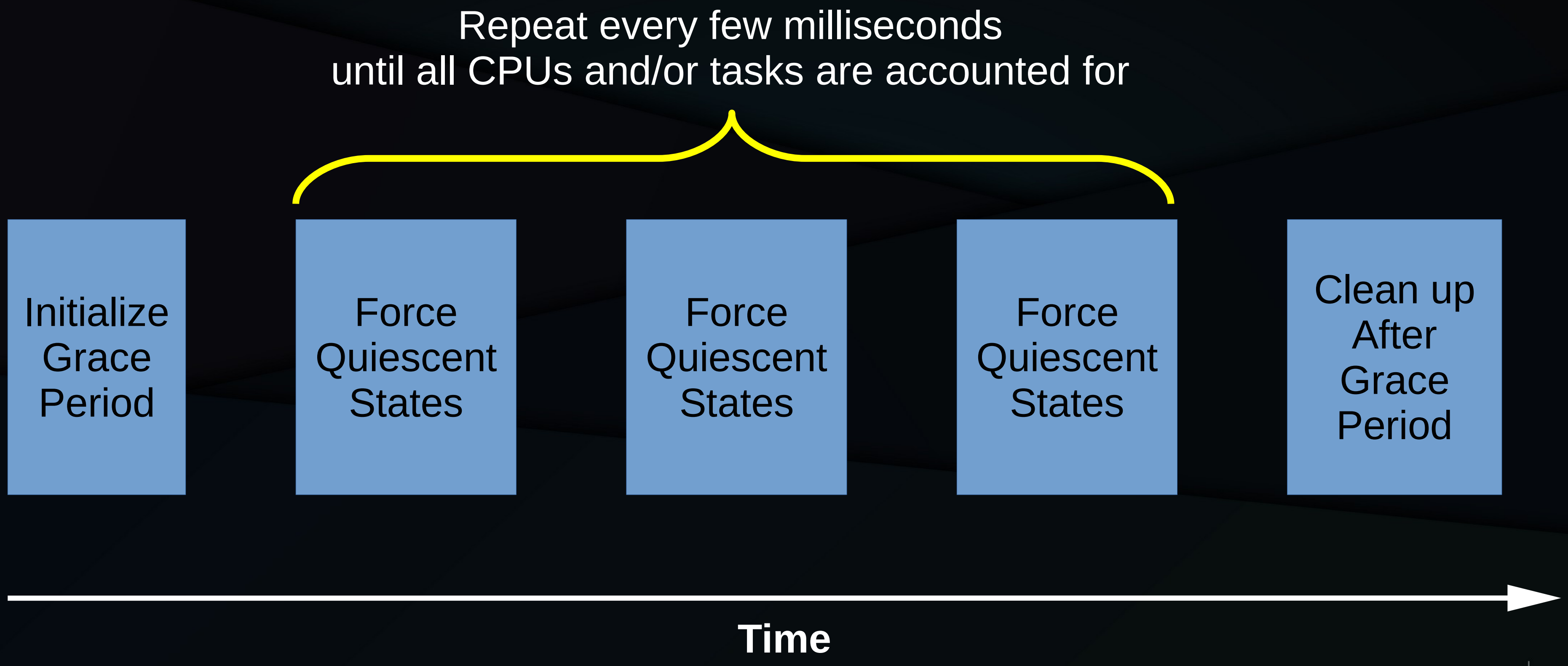
If you must choose, choose the events causing the most trouble

Detect, ***Then*** Instrument

Detect, ***Then*** Instrument

- In theory, code executed after the heisenbug occurs does not affect MTBF
  - In practice, code-size changes can affect MTBF, but this is relatively rare, at least until you count on it
- Two tales of timers…

Cartoon of RCU Grace Periods



Repeat every few milliseconds
until all CPUs and/or tasks are accounted for

Initialize Grace Period

Force Quiescent States

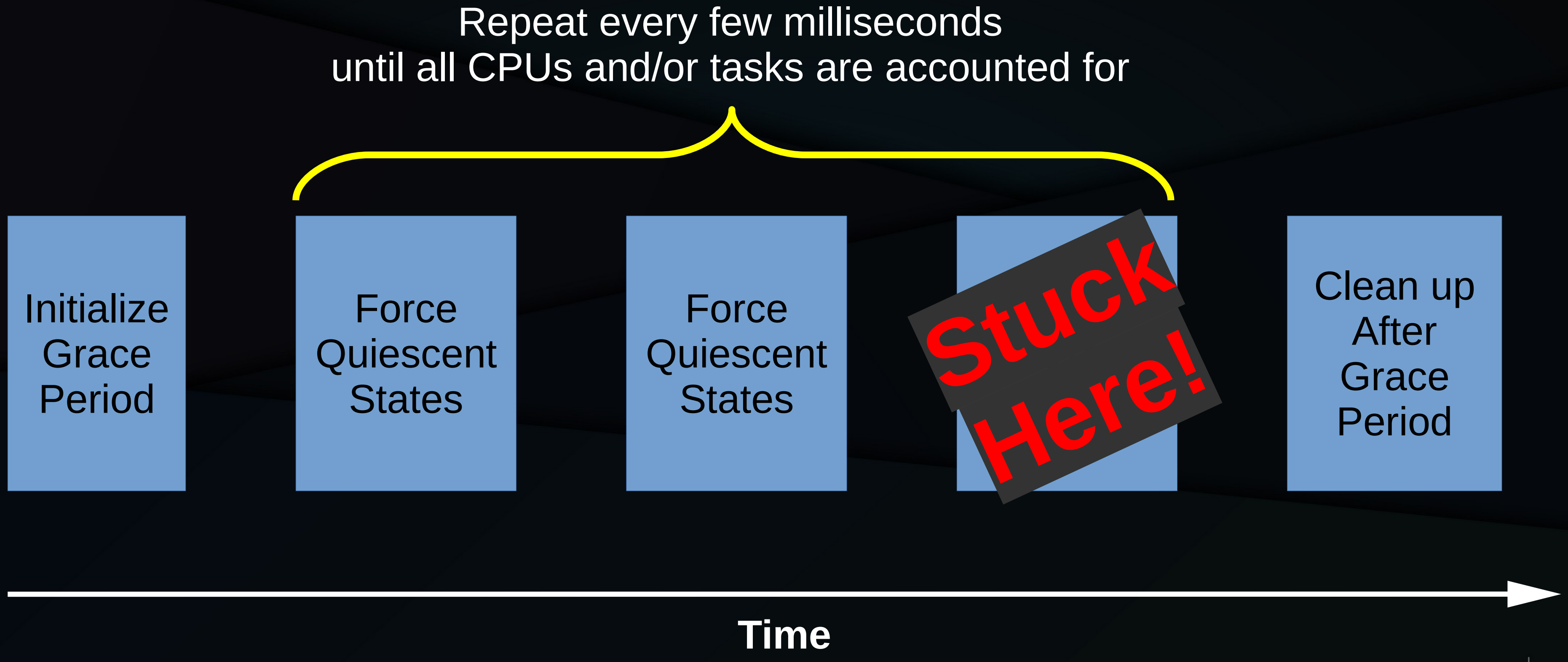
Force Quiescent States

Force Quiescent States

Clean up After Grace Period

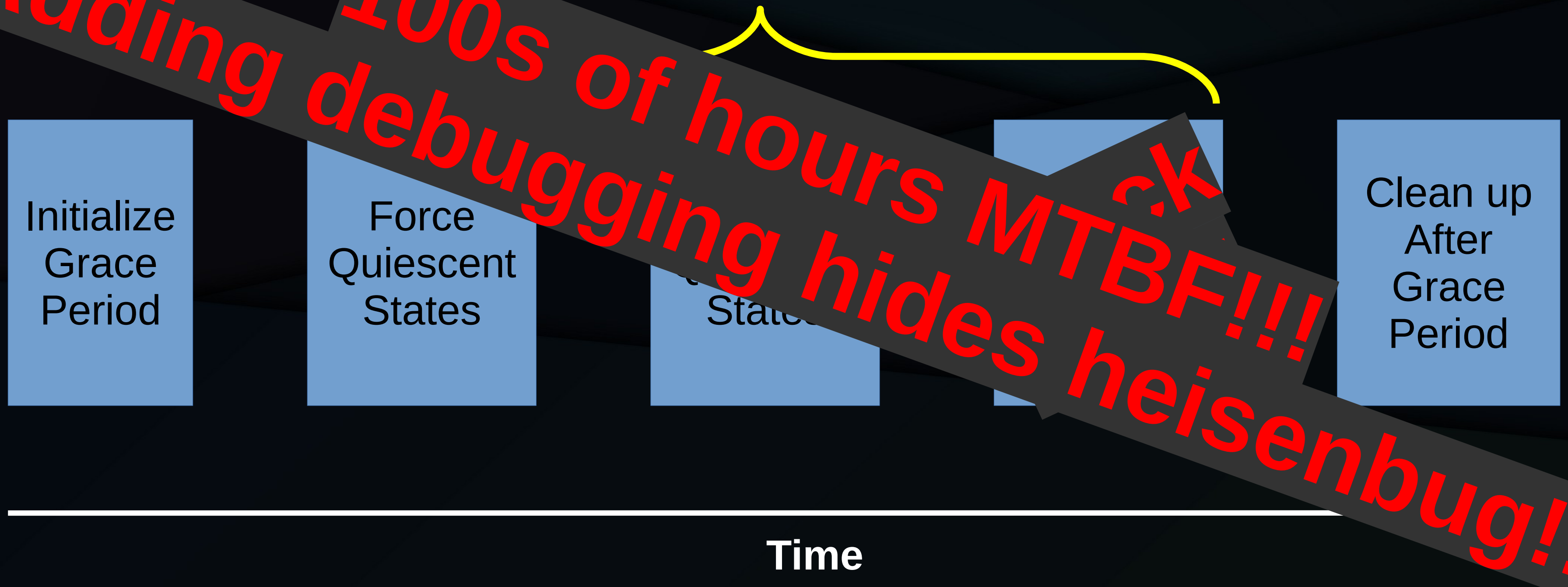
Time

Cartoon of RCU Grace Periods


Repeat every few milliseconds
until all CPUs and/or tasks are accounted for
Initialize Grace Period
Force Quiescent States
Force Quiescent States
Stuck Here!
Clean up After Grace Period
Time

Cartoon of RCU Grace Periods

Repeat every few milliseconds
until all CPUs and/or tasks are accounted for



100s of hours MTBF!!!

Adding debugging hides heisenbug!!!

Hunting Stuck-GP Heisenbug

- About a year to reduce MTBF to ~300 hours
  - Choose .config to increase MTBF
  - Increase rate of CPU-hotplug operations
  - Debug still hides heisenbug
  - RCU CPU stall warning restores forward progress

Hunting Stuck-GP Heisenbug

- About a year to reduce MTBF to ~3 urs
  - Choose .config to increase MT
  - Increase rate of CPU-h operations
  - Debug still hid nbug
  - RCU CP warning restores forward progress

Add debug after detection!!!

Adding Debug After Detection

- If timer took more than eight seconds and more than three times as long as was requested, dump debugging information
  - Heuristic, but good enough in this case
  - Bug was due to an interaction between timers, CPU hotplug, and RCU

Adding Debug After Detection

- If timer took more than eight seconds and more than three times as long as was requested, dump debugging information
  - Heuristic, but good enough in this case
  - Bug was due to an interaction between timers, CPU hotplug, and RCU
  - Recent bug between workqueues and RCU?

Adding Debug After Detection

- If timer took more than eight secon[illegible] more than three times as long as w[illegible]quested, dump debugging inform[illegible]
  - Heuristic, but goo[illegible] this case
  - Bug was [illegible]ction between timers, CPU hotpl[illegible]d RCU
  - [illegible]ent bug between workqueues and RCU?

Post-facto debug did not increase MTBF!!!

Adding Debug After Detection

- If tim… more than eight secon… more than thr… as long as w… quested, dump d…
  - Heuristic, but … this case
  - Bug was … timers, CPU hotp… RCU
- …ent bug between workqueues … RCU

Anti-heisenbug: Add debugging after bug is detected!!!

Post-facto de… id not increase M…

Taking This One Step Further...

- When rare combination of events takes system to a legal but vulnerable state, start the system in that state
  - The nuclear option: White-box testing
  - Exhaustive state testing of `rcu_segcblist`
    - Done in userspace

Taking This One Step Further...

- When rare combination of events system to a legal but vulnerable stat the system in that state
  - The nuclear opti esting
  - Exhausti of rcu_segcblist
    - ce

Anti-heisenbug: Force rare risky legal states!!!

Heisenbugs: The Goal

Heisenbugs: The Goal

- What is better than being proficient at hunting heisenbugs?

Heisenbugs: The Goal

- What is better than being proficient at hunting heisenbugs?
- **Not having heisenbugs in the first place!!!**

How to Avoid Hunting Heisenbugs

How to Avoid Hunting Heisenbugs

- No easy way out, but:
  - Careful concurrency-first design
  - Thorough unit testing, including stress testing
  - Thorough integration testing
  - Stringent code-review process
  - Verification, if applicable

How to Avoid Hunting Heisenbugs

- No easy way out, but:
  - Careful concurrency-first desi
  - Thorough unit testin esting
  - Thorough i
  - S w process
  - verific , if applicable

Anti-heisenbug: Don’t randomly hack concurrent code!!!

Summary

Summary: How to Hunt Heisenbugs

- Create anti-heisenbugs
  - Reduce MTBF
  - Increase workload intensity
  - Look for and promote trouble
  - Inject strategic delays
  - Count near misses
  - Swamp queues
  - De-emphasize fastpaths
  - Combine rare events
  - Add debugging ***after*** bug is detected
  - Force rare risky legal states (whitebox)

- Avoid (many) heisenbugs
  - Careful concurrency-first design
  - Thorough unit testing, including stress testing
  - Thorough integration testing
  - Stringent code-review process
  - Verification, if applicable
  - Don’t randomly hack concurrent code!!!

Summary: How to Hunt Heisenbugs

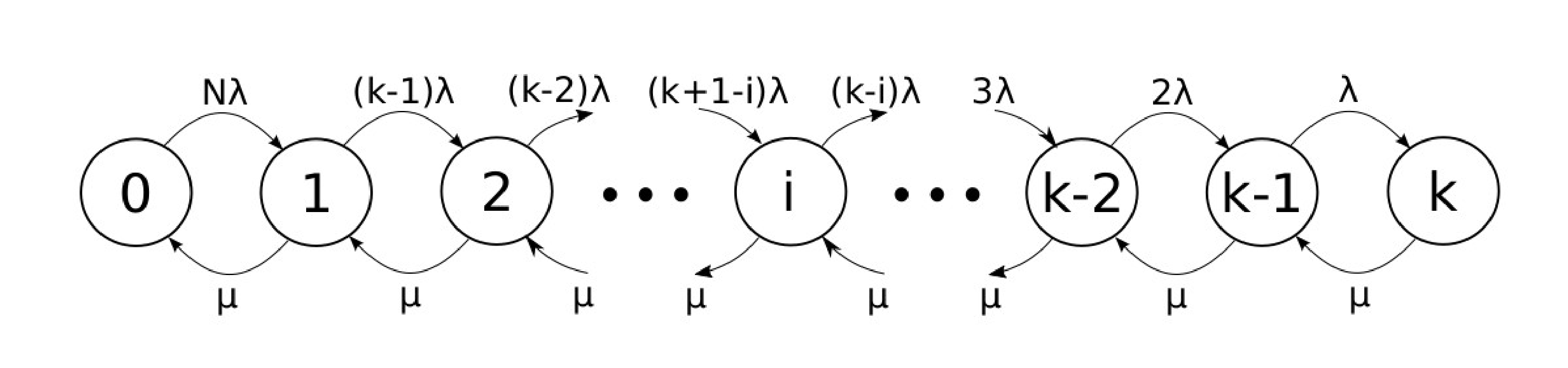
- Create anti-heisenbugs
  - Reduce MTBF
  - Increase workload intensity
  - Look for and promote trouble
  - Inject strategic delays
  - Count near misses
  - Swamp queues
  - De-empha[illegible]
  - Combin[illegible] ts
  - Add debugging ***after*** bug detected
  - Force rare risky legal states (whitebox)
- Avoid (many) heisenbugs
  - Careful [illegible]-first design
  - [illegible] it testing, including
  - [illegible]
  - [illegible] integration testing
  - Stringent code-review process
  - Verification, if applicable
  - Don’t randomly hack concurrent code!!!

No “silver bullet”, but many useful techniques

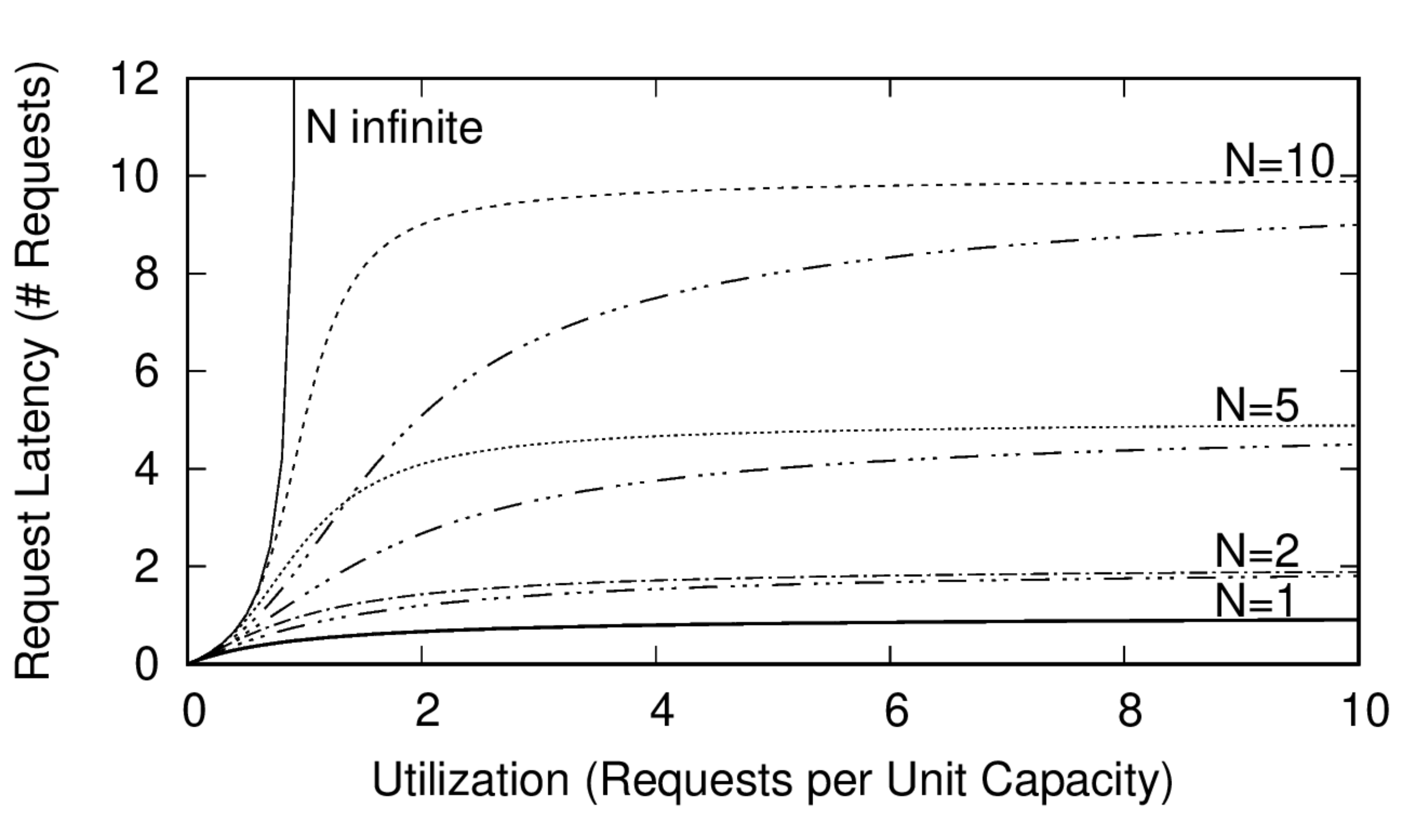
Questions?

Backup

Finite Requests into Finite Queue



Finite Requests into Finite Queue


Request Latency (# Requests)
N infinite
N=10
N=5
N=2
N=1
Utilization (Requests per Unit Capacity)