RELIABILITY IN MODERN CLOUD SYSTEMS

Summer 2025

LOGISTICS

ASSIGNMENT 2

- Assignment 2 was due last week
- Grades posted next week

ASSIGNMENT 3

- Assignment 3 was released on Monday June 2nd
- Due Date: Wednesday, June 18th 5pm
- Description: Reproducing a retry storm metastability failure
  - Not using the luggage sharing application
  - Instead using Hotel Reservation from DeathStarBench benchmark suite

LOCATION CHANGE (NEXT WEEK)

- Next week's lecture will be in room 105 (not 005)
- 105 is the room directly above our current room

EMERGENT MISBEHAVIORS DISCUSSION

PAPER #1 SUMMARY

PAPER #1 SUMMARY

- Study of open-source CSI failures
- ~20% of cloud incidents caused by CSI failures
- Data-plane failures account for half of all CSI failures
  - More complicated the data abstraction, higher the chance of a CSI failure
- Management-plane failures can impact observability
- Most CSI failure fixes are in dedicated connector modules

PAPER #2 SUMMARY

PAPER #2 SUMMARY

- Open-source study of metastability failures
- Framework and characterization of different types of metastability
- These failures lead to outages (4-10hr common)
- >50% caused by retries
- Metastability is directly dependent on workload, trigger, and current system state

DISCUSSION THEMES

- What are other examples of emergent misbehaviors in systems?
- What is the right strategy for dealing with metastable failures?
- How do you know if a system is vulnerable to a metastable failure?
- What is the right strategy for preventing metastable failures?

DISCUSSION THEMES

- What are other examples of emergent misbehaviors in systems?

DISCUSSION THEMES

- What are other examples of emergent misbehaviors in systems?

Laser of Death

- Load Balancer acts like a laser of death
- If 1 replica fails due to overload, balancer moves work to healthy replicas
- If failure detected due to health checks then it's a "killer health check"
- Healthy replicas now get overworked
- Replicas eventually fail!

Cross-System Inconsistency

- Write some value to a replicated database
- Reading that value returns the stale value instead of the newly written value
- Caused due to new writes getting to the replicas before the read request
- Common in geo-distributed datastores

DISCUSSION THEMES

- What is the right strategy for dealing with metastable failures?

DISCUSSION THEMES

- What is the right strategy for dealing with metastable failures?

There are 2 strategies: increasing capacity or load-shedding (decreasing the load). Increasing capacity is limited but load-shedding hurts availability! Need both ☺

DISCUSSION THEMES

- What is the right strategy for preventing metastable failures?

DISCUSSION THEMES

- What is the right strategy for preventing metastable failures?

Choosing a design that limits the artificial workload the system can generate, preventive load shedding strategies, aggressive capacity increasing strategies

DISCUSSION THEMES

- How do you know if a system is vulnerable to a metastable failure?

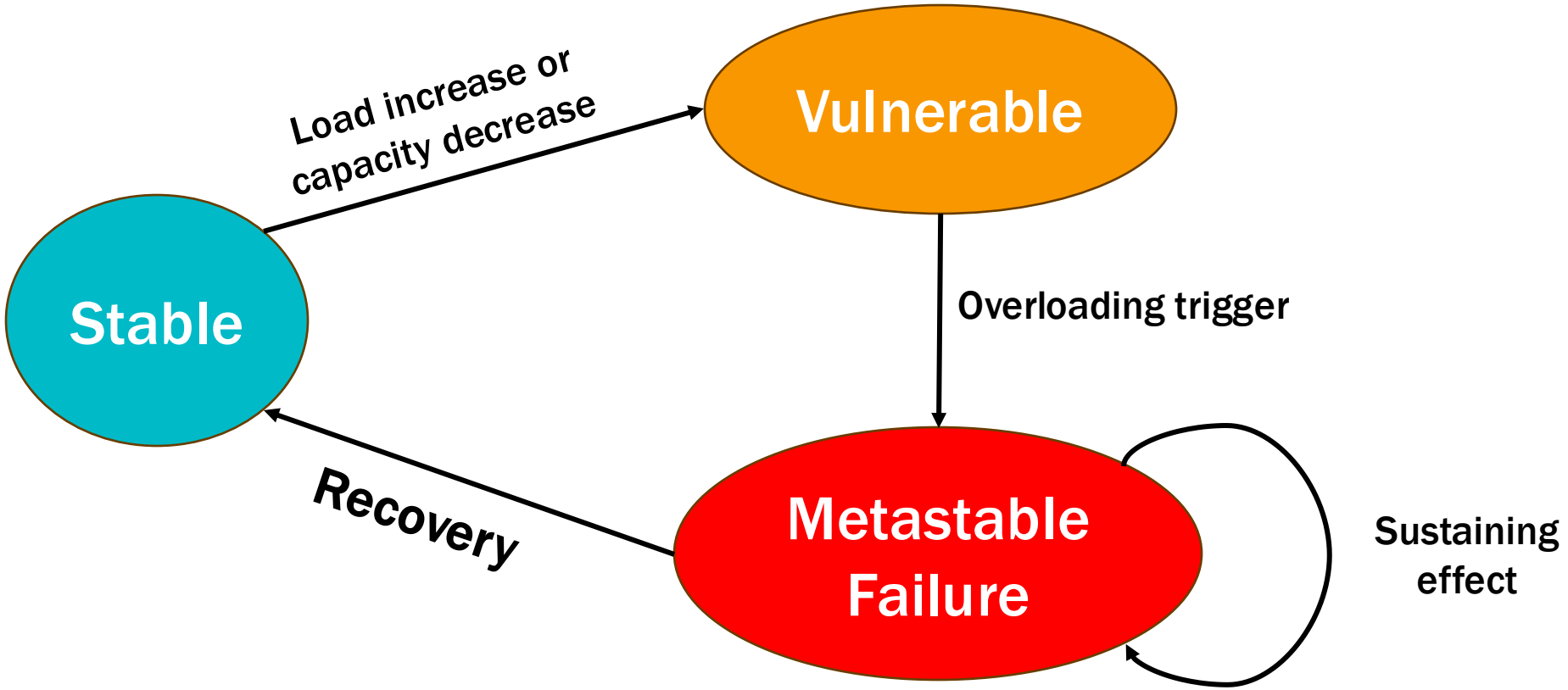
DISCUSSION THEMES

- How do you know if a system is vulnerable to a metastable failure?

Requires a lot of manual analysis currently. This is an open problem that researchers are working on right now.

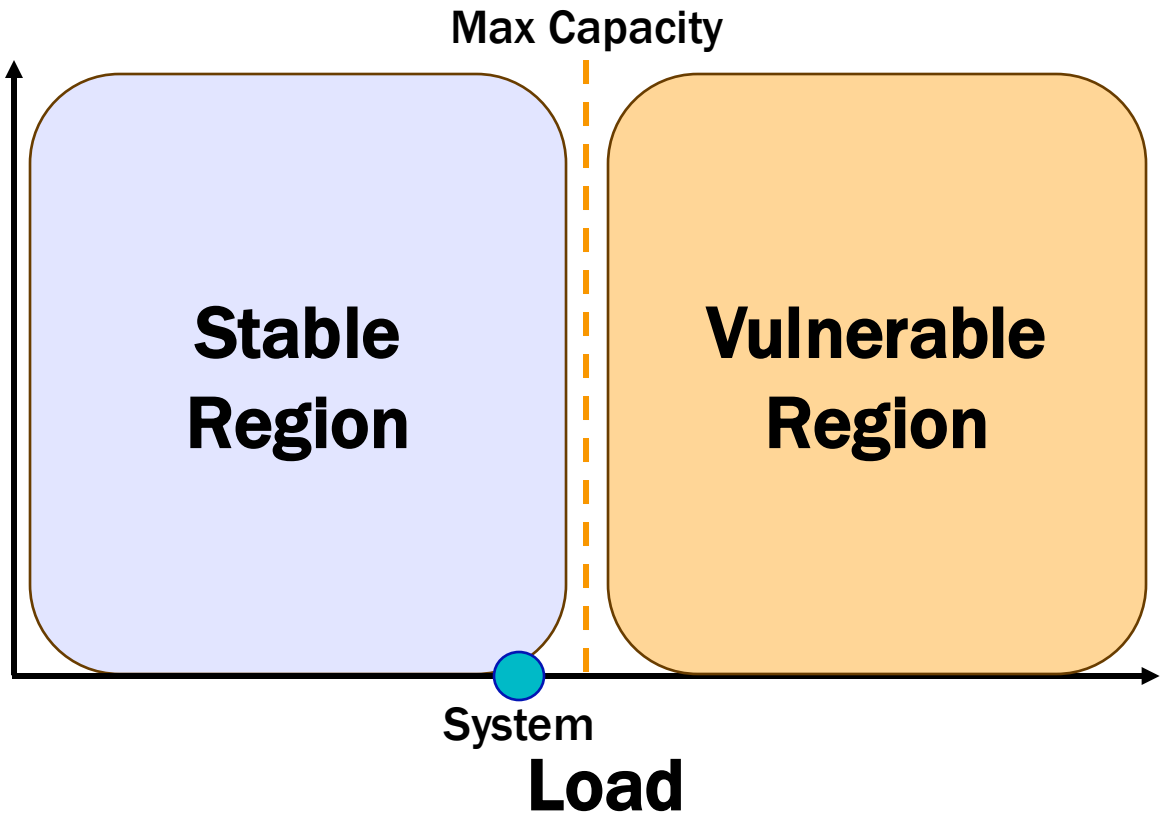
DEALING WITH METASTABILITY

SYSTEM STATES



ANALYZING SYSTEMS FOR METASTABILITY

ANALYZING SYSTEMS FOR METASTABILITY

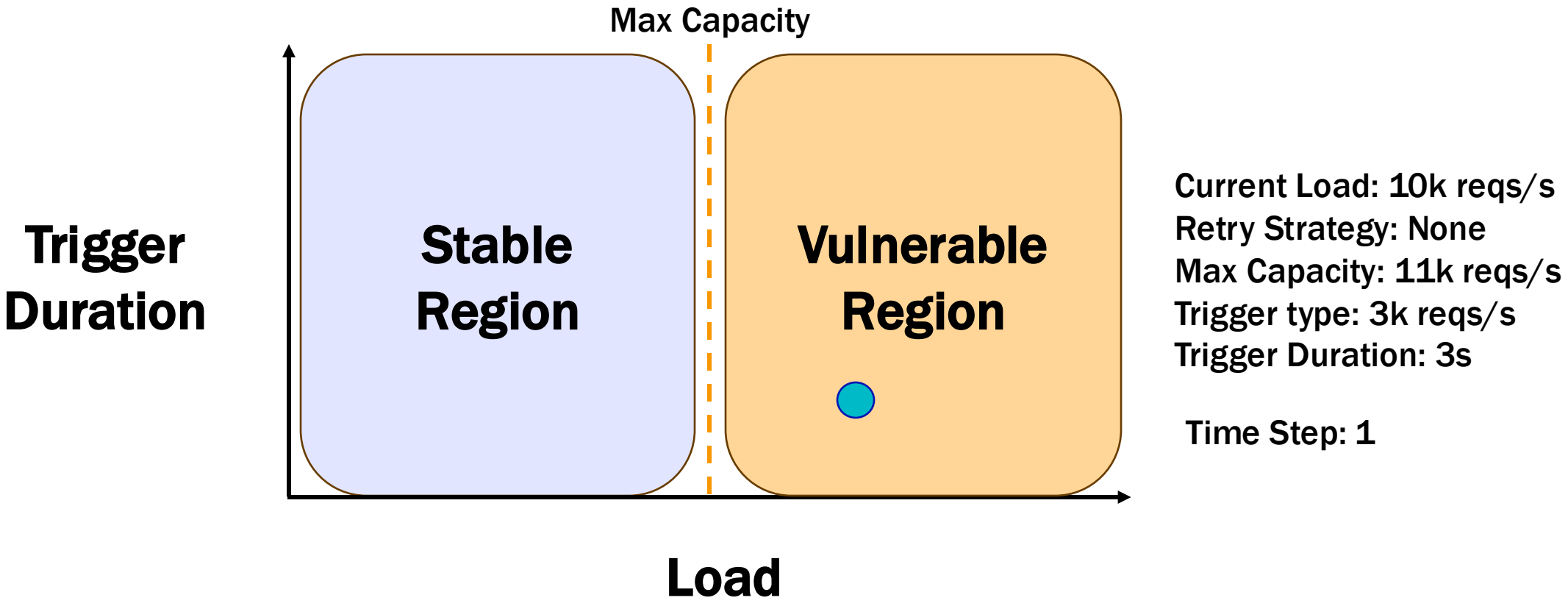


Trigger Duration

Current Load: 10k reqs/s
Retry Strategy: None
Max Capacity: 11k reqs/s
Trigger type: 3k reqs/s
Trigger Duration: 3s

Time Step: 0

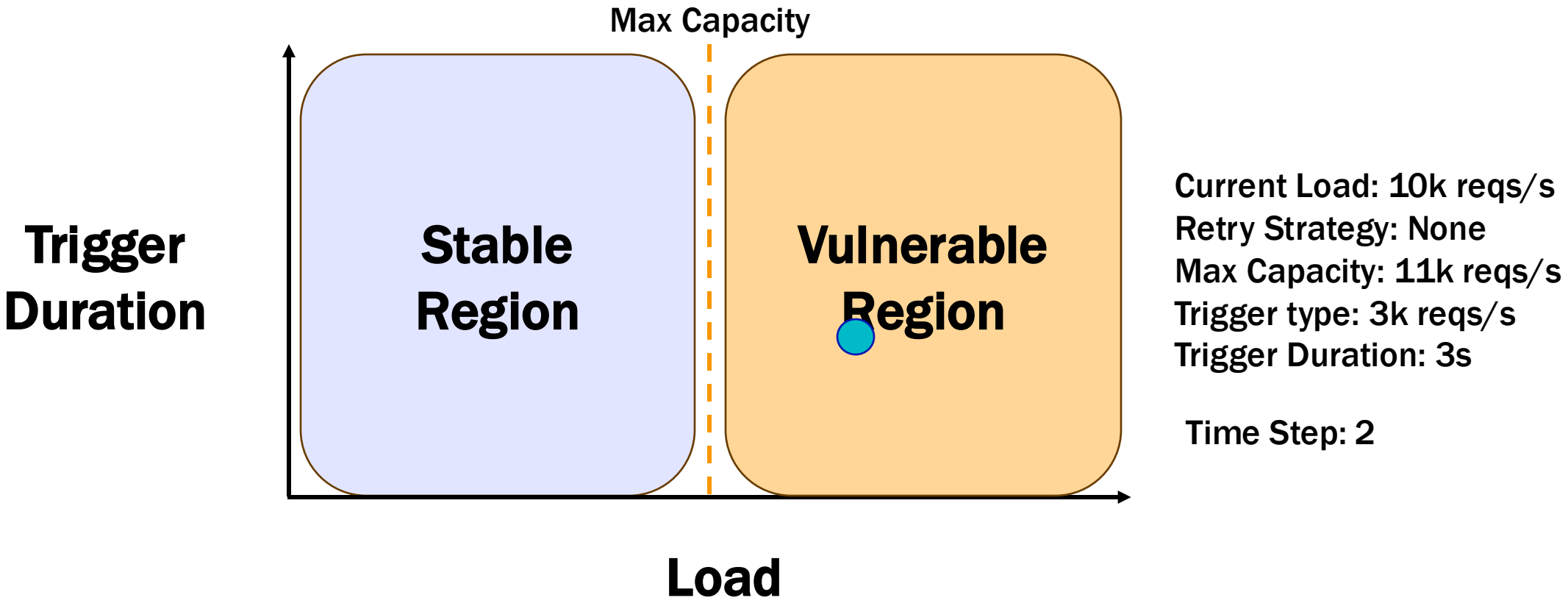
ANALYZING SYSTEMS FOR METASTABILITY



Current Load: 10k reqs/s
Retry Strategy: None
Max Capacity: 11k reqs/s
Trigger type: 3k reqs/s
Trigger Duration: 3s

Time Step: 1

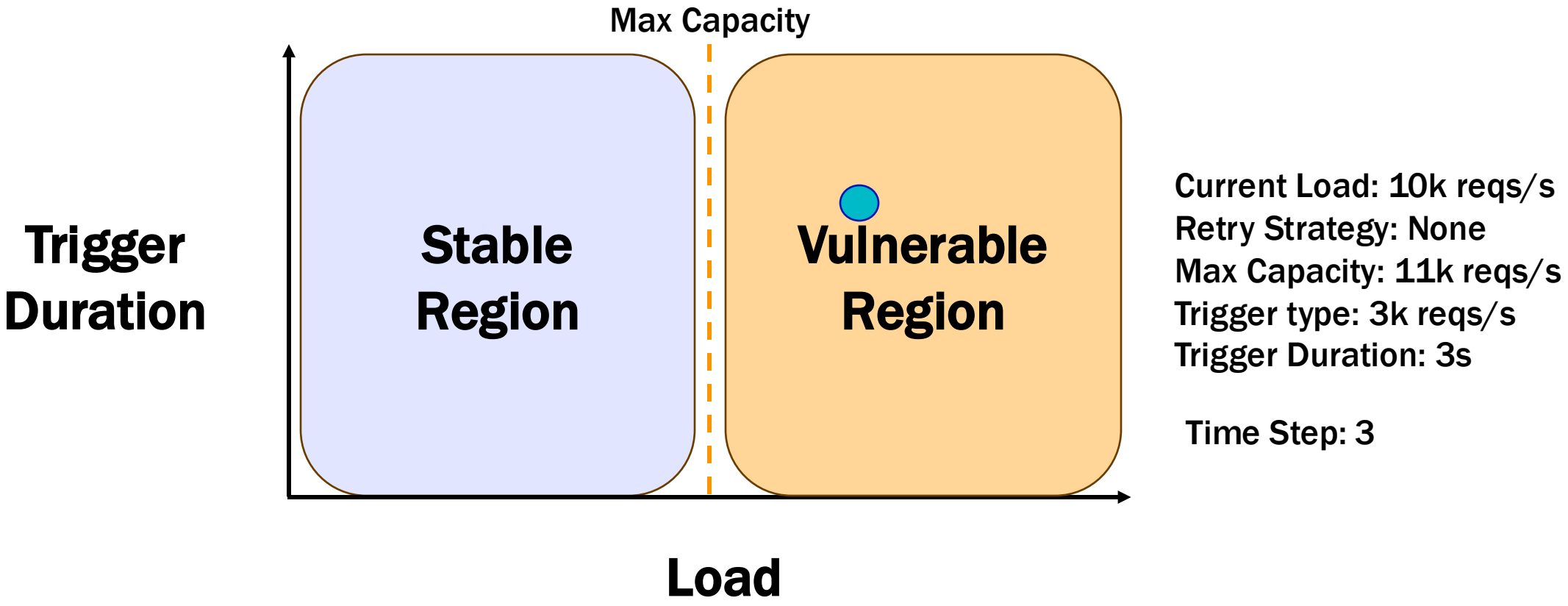
ANALYZING SYSTEMS FOR METASTABILITY



Current Load: 10k reqs/s
Retry Strategy: None
Max Capacity: 11k reqs/s
Trigger type: 3k reqs/s
Trigger Duration: 3s

Time Step: 2

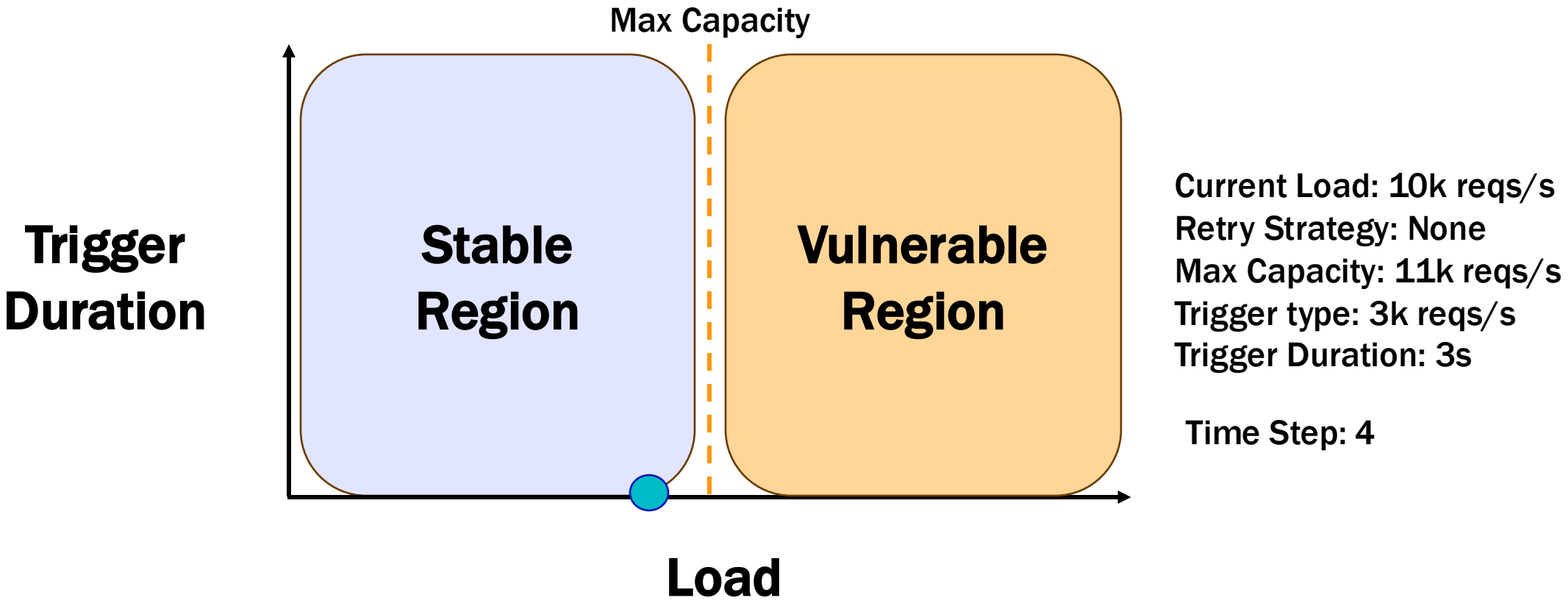
ANALYZING SYSTEMS FOR METASTABILITY



Current Load: 10k reqs/s
Retry Strategy: None
Max Capacity: 11k reqs/s
Trigger type: 3k reqs/s
Trigger Duration: 3s

Time Step: 3

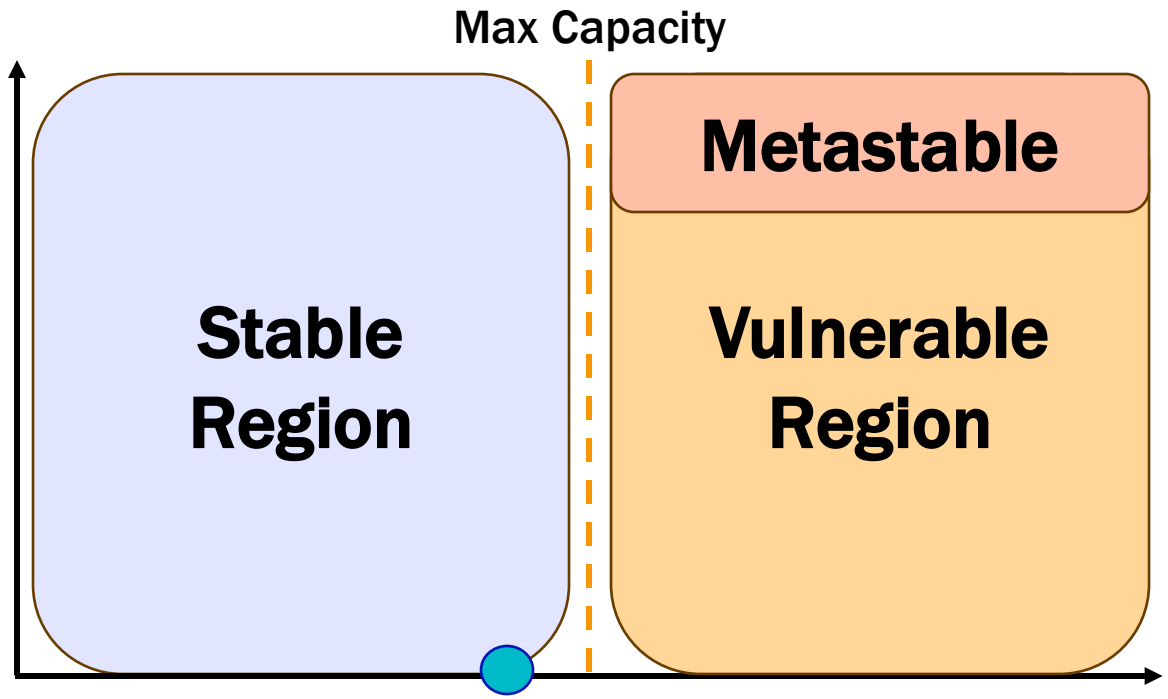
ANALYZING SYSTEMS FOR METASTABILITY



Current Load: 10k reqs/s
Retry Strategy: None
Max Capacity: 11k reqs/s
Trigger type: 3k reqs/s
Trigger Duration: 3s

Time Step: 4

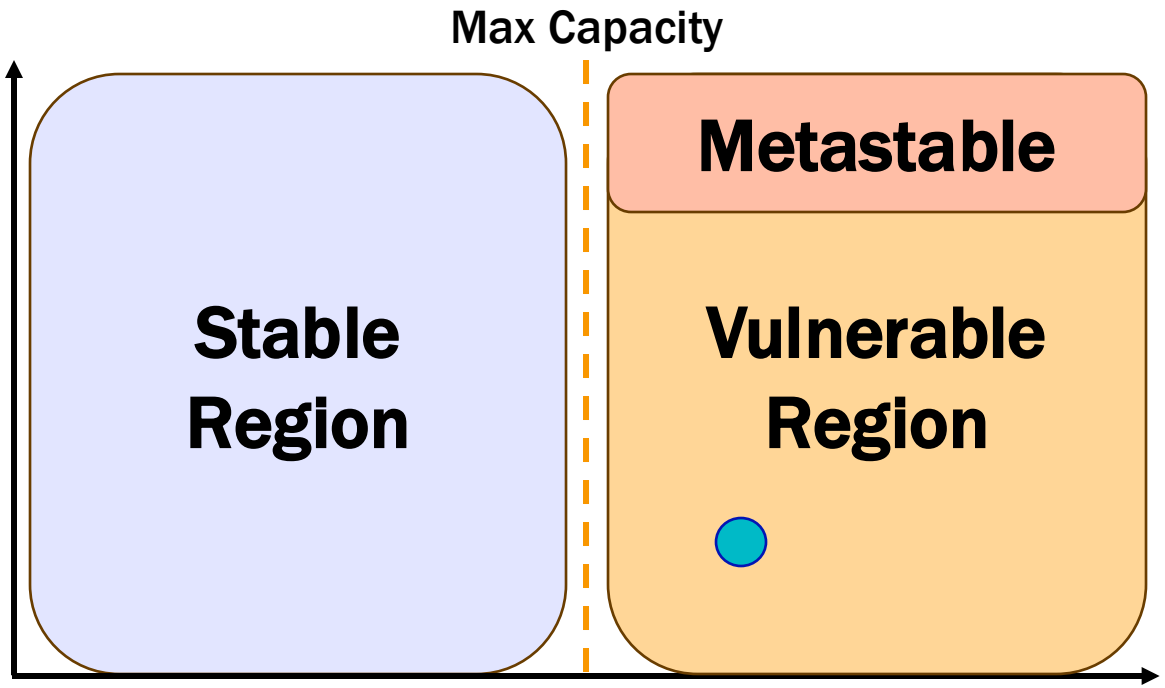
RETRY STORM



Current Load: 10k reqs/s
Retry Strategy: Try 3 times
Max Capacity: 11k reqs/s
Trigger type: 3k reqs/s
Trigger Duration: 3s

Time Step: 0

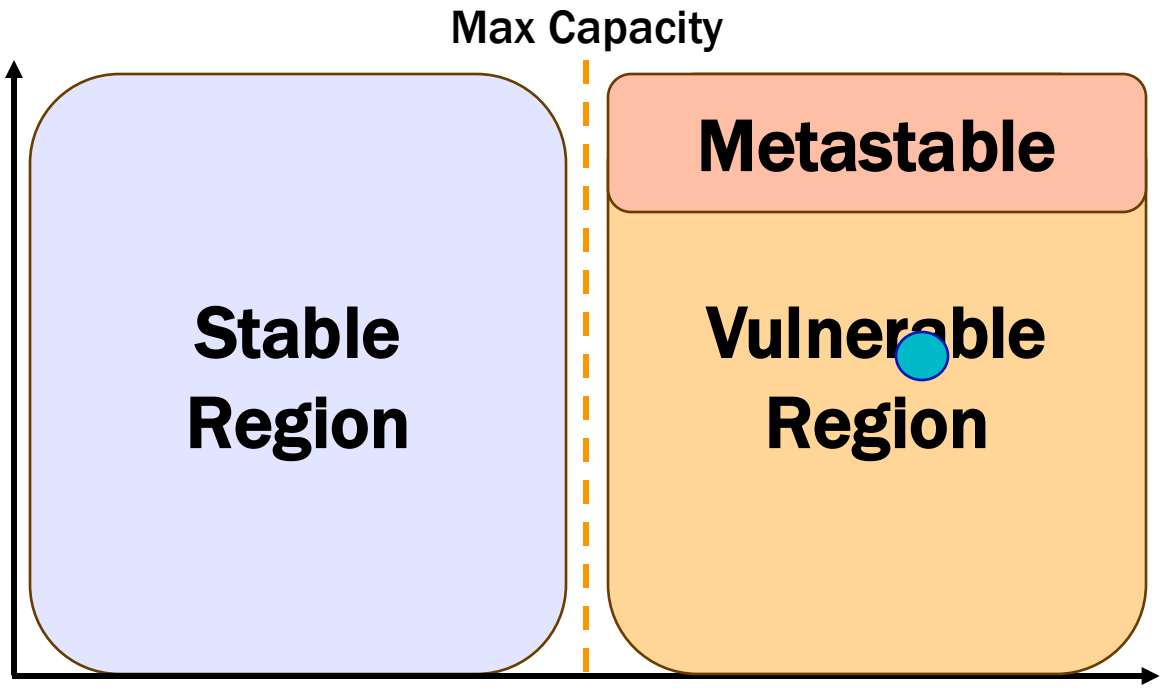
RETRY STORM



Current Load: 10k reqs/s
Retry Strategy: Try 3 times
Max Capacity: 11k reqs/s
Trigger type: 3k reqs/s
Trigger Duration: 3s

Time Step: 1

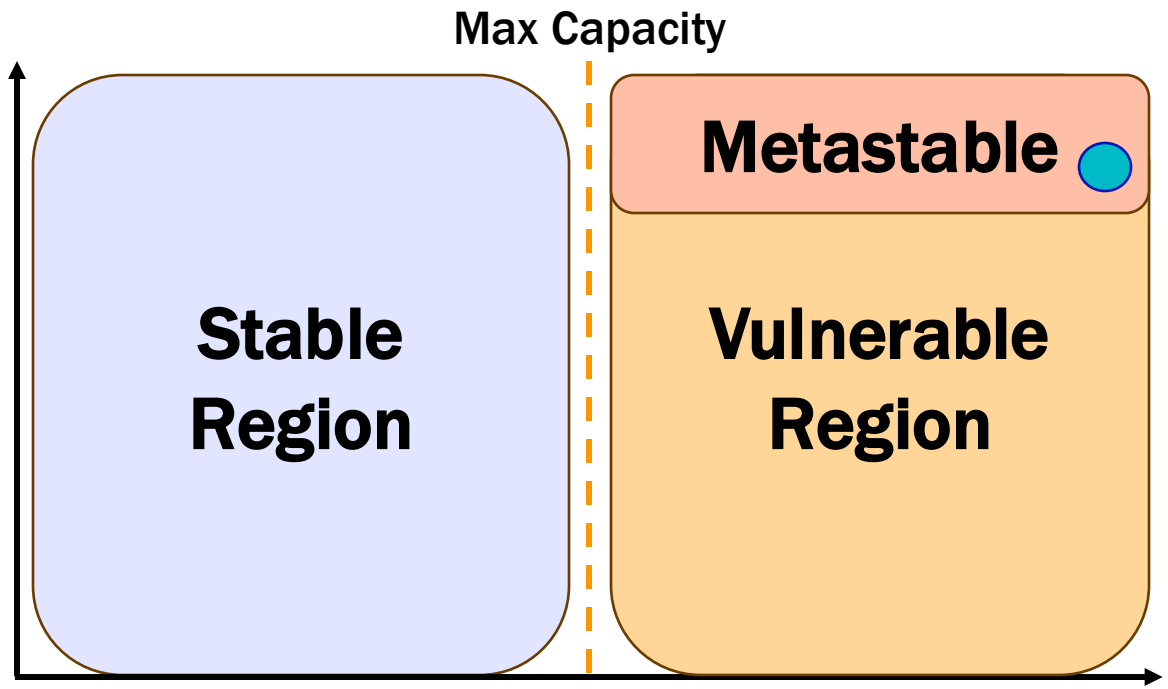
RETRY STORM



Current Load: 10k reqs/s
Retry Strategy: Try 3 times
Max Capacity: 11k reqs/s
Trigger type: 3k reqs/s
Trigger Duration: 3s

Time Step: 2

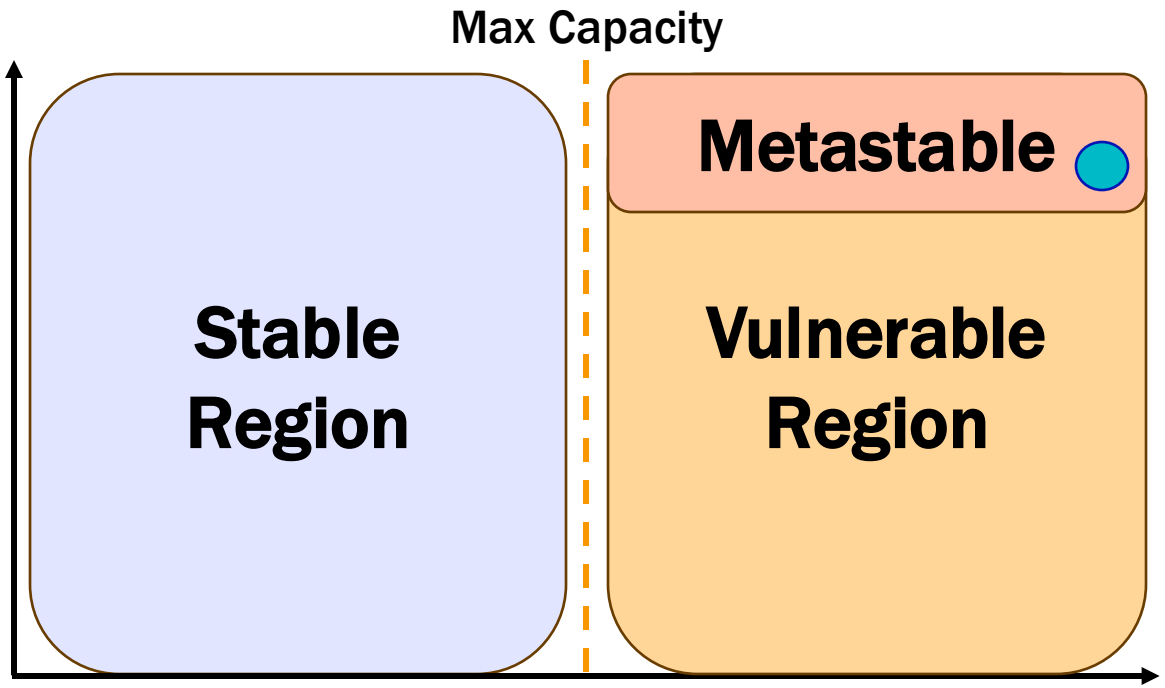
RETRY STORM



Current Load: 10k reqs/s
Retry Strategy: Try 3 times
Max Capacity: 11k reqs/s
Trigger type: 3k reqs/s
Trigger Duration: 3s

Time Step: 3

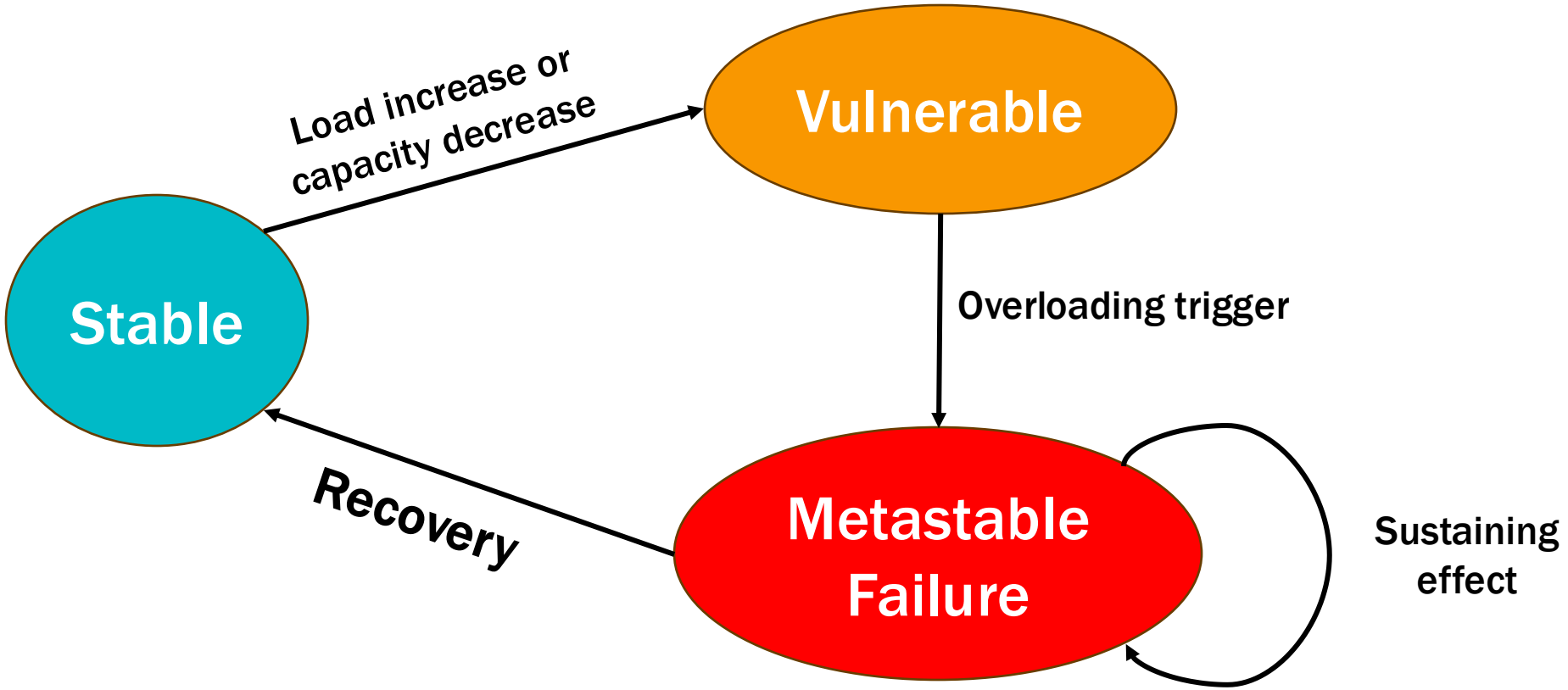
RETRY STORM



Current Load: 10k reqs/s
Retry Strategy: Try 3 times
Max Capacity: 11k reqs/s
Trigger type: 3k reqs/s
Trigger Duration: 3s

Time Step: 4

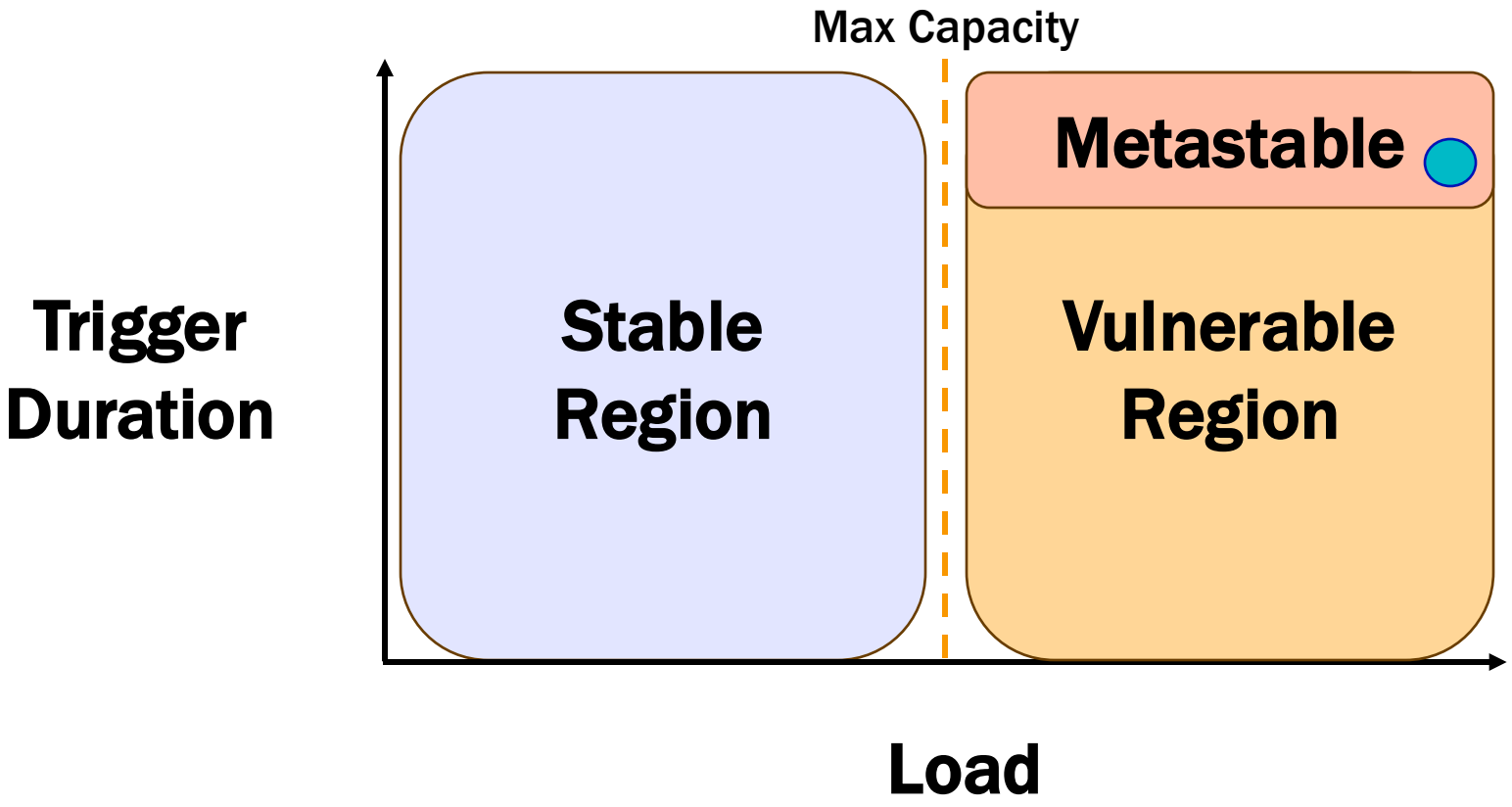
SYSTEM STATES



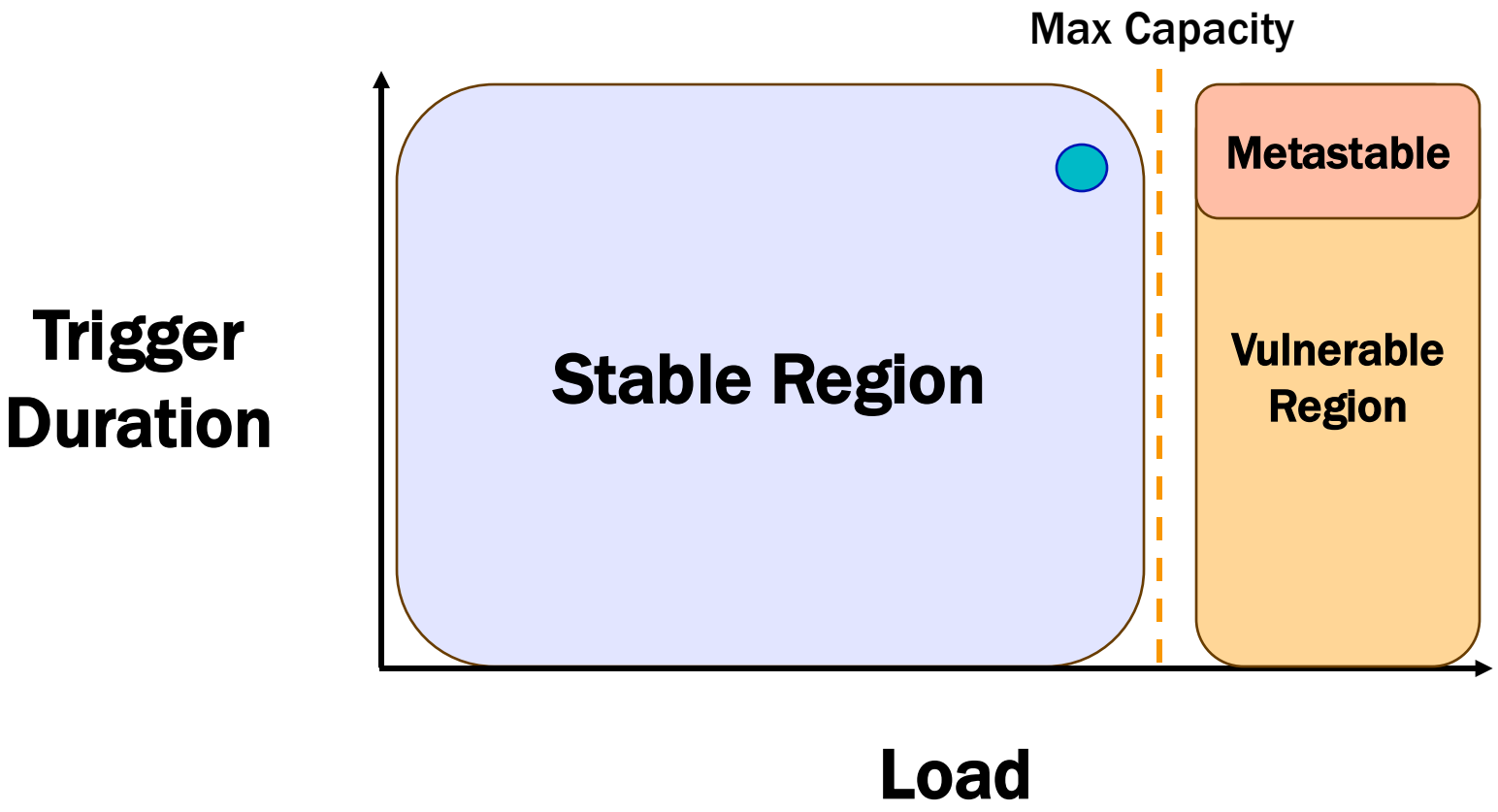
CURING METASTABILITY

- Increasing Capacity
  - Adding more instances
  - Adding more resources

AUTOSCALING



AUTOSCALING



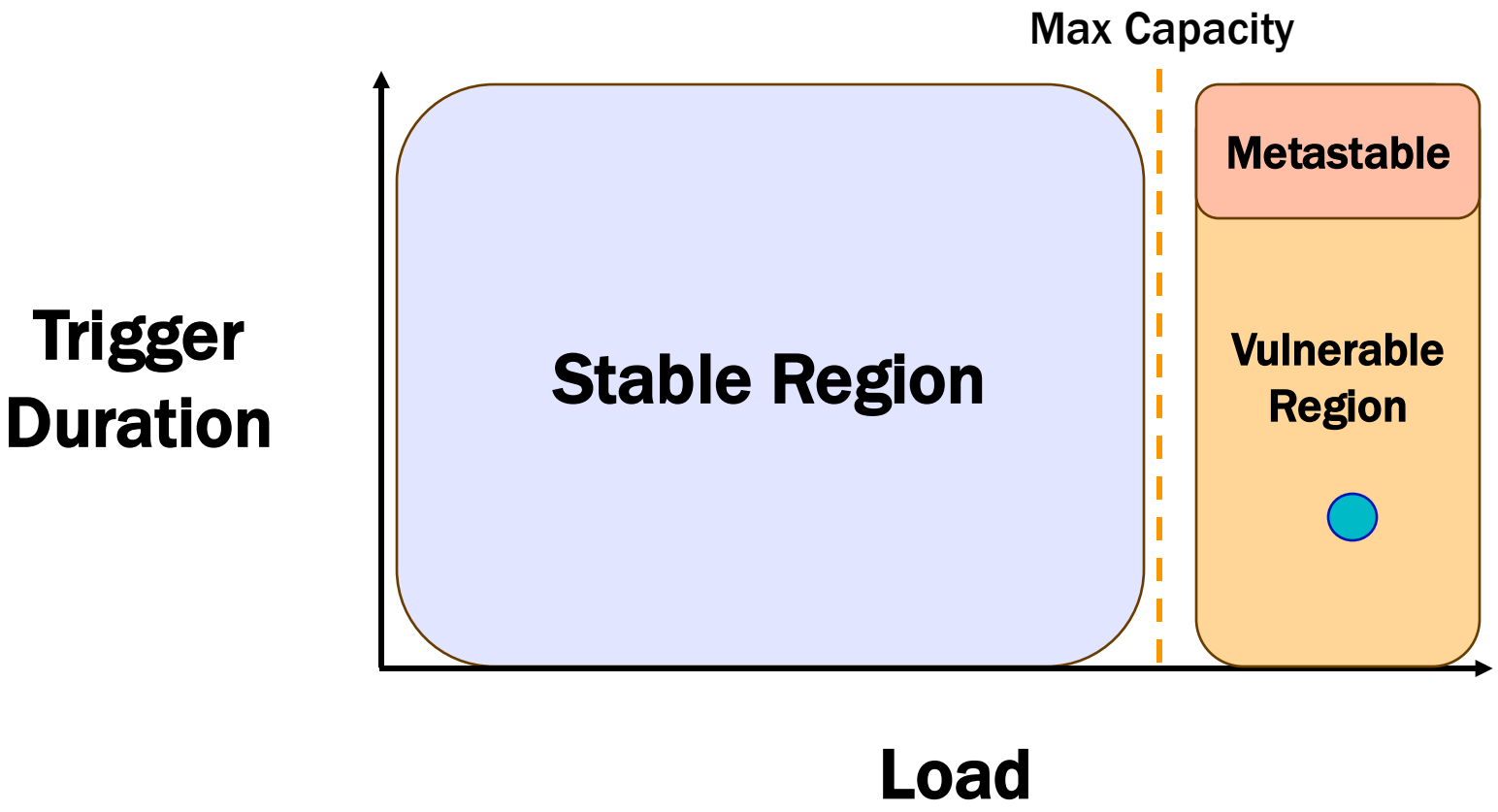
CURING METASTABILITY

- Load Shedding
- Increasing Capacity
  - Adding more instances
  - Adding more resources

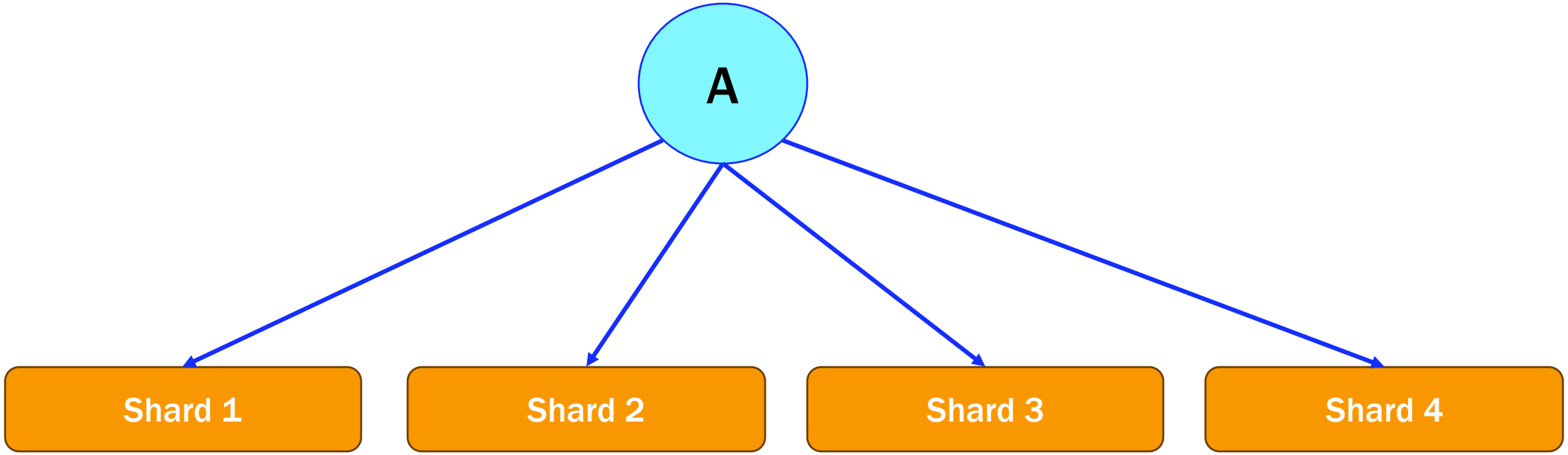
CIRCUITBREAKERS

- Prevents new requests from entering the system
- Rejects any new load until the downstream “heals”

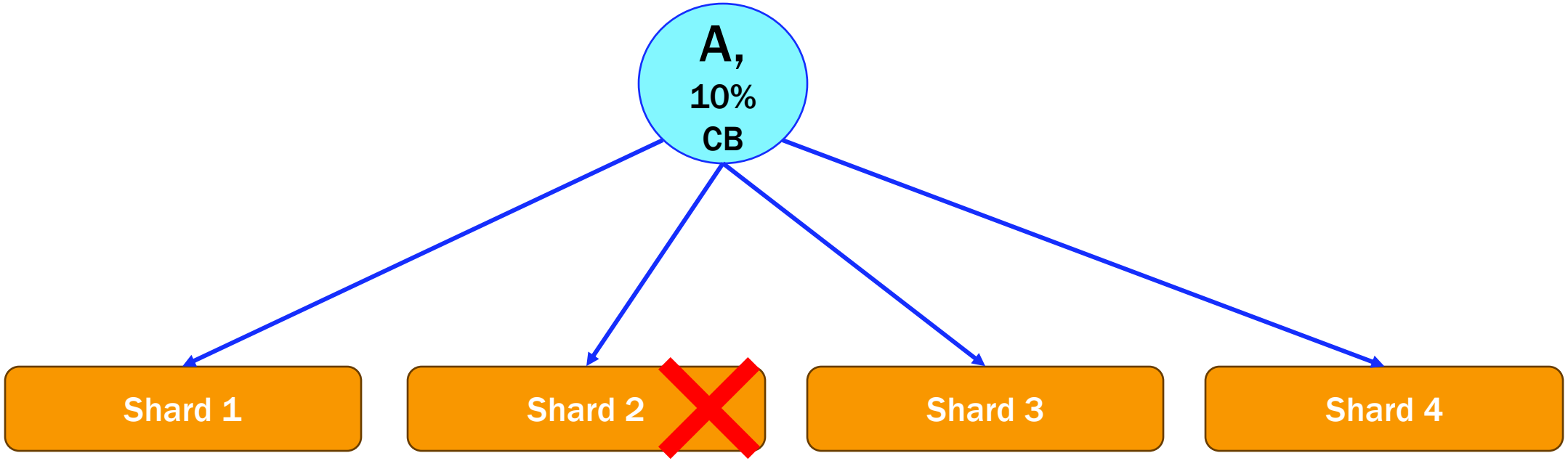
CIRCUITBREAKERS



CIRCUITBREAKING SHARDED SERVICE



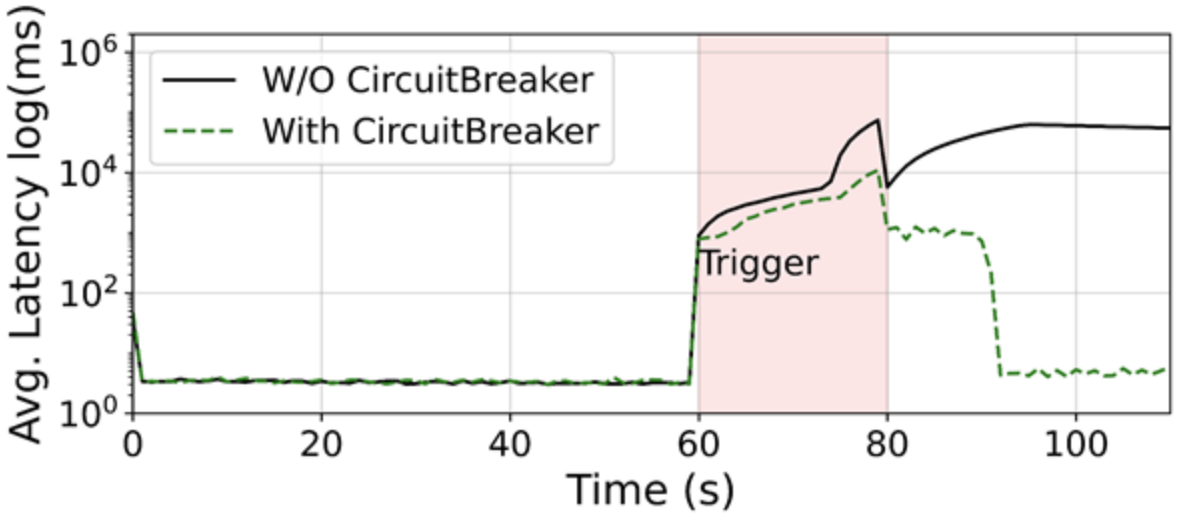
CIRCUITBREAKING SHARDED SERVICE



CIRCUITBREAKERS: KEY ISSUE

- Prevents new requests from entering the system
- Rejects any new load until the downstream “heals”

ISSUE: SACRIFICES AVAILABILITY!



LOAD-SHEDDING WHILE PRESERVING AVAILABILITY

LOAD-SHEDDING WHILE PRESERVING AVAILABILITY

Key Idea: Remove non-essential functionality while preserving the main features of the application

Eg: Remove nice-to-have UI elements and focus on the important requests!

LOAD-SHEDDING WHILE PRESERVING AVAILABILITY

Key Idea: Remove non-essential functionality while preserving the main features of the application

Eg: Remove nice-to-have UI elements and focus on the important requests!

REQUEST-BASED LOAD SHEDDING

- Drop/Accept request based on some intrinsic characteristic of the request
- Eg: Rate-limiting based on the IP address or user-id

PRIORITIZED LOAD SHEDDING

- For every generated request, attach a priority level
- At the entrypoint, filter out any request that has a priority level lower than the current acceptable request level
- Ensures that the high priority requests are still being serviced whereas low priority requests get dropped so that they don't disrupt the system

Keeping Netflix Reliable Using Prioritized Load Shedding, Netflix Tech Blog Nov 2020, Correa et al

SERVICE LEVEL LOAD SHEDDING

- Different services may have different priorities!
- Instead of load-shedding with a one-for-all policy, allow each service to define their own policy for load shedding

  - **CRITICAL:** Affect core functionality — These will never be shed if we are not in complete failure.
  - **DEGRADED:** Affect user experience — These will be progressively shed as the load increases.
  - **BEST_EFFORT:** Do not affect the user — These will be responded to in a best effort fashion and may be shed progressively in normal operation.
  - **BULK:** Background work, expect these to be routinely shed.

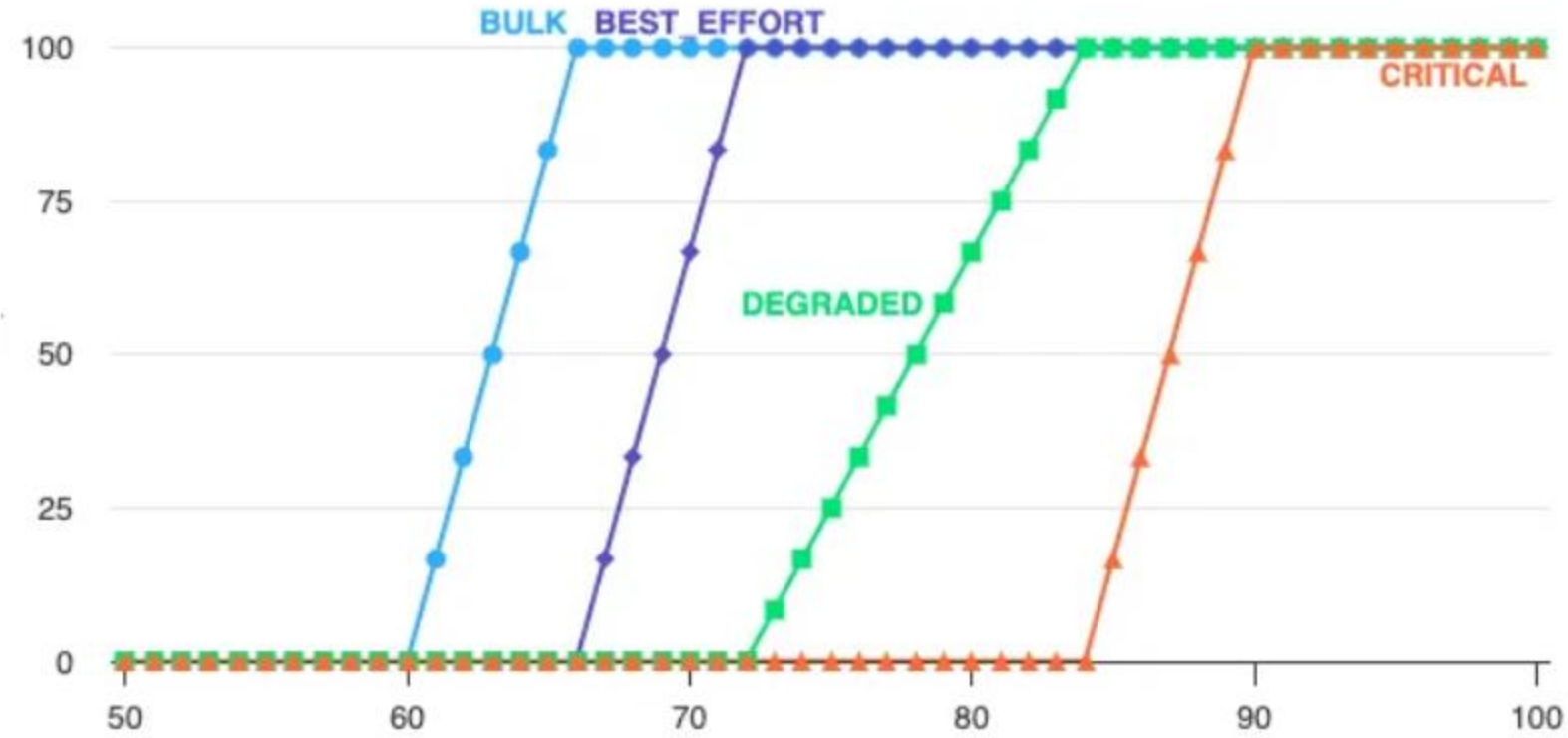
Enhancing Netlfix Reliability with Service level prioritized load shedding, Netflix Tech Blog Jun 2024, Mendiratta et al

EXAMPLE: NETFLIX PLAY API

- **CRITICAL:** Affect core functionality — These will never be shed if we are not in complete failure.
- **DEGRADED:** Affect user experience — These will be progressively shed as the load increases.
- **BEST_EFFORT:** Do not affect the user — These will be responded to in a best effort fashion and may be shed progressively in normal operation.
- **BULK:** Background work, expect these to be routinely shed.

Enhancing Netlfix Reliability with Service level prioritized load shedding, Netflix Tech Blog Jun 2024, Mendiratta et al

EXAMPLE: NETFLIX PLAY API



Percentage of requests (Y-axis) being load-shed based on CPU utilization (X-axis) for different priority buckets

Enhancing Netlfix Reliability with Service level prioritized load shedding, Netflix Tech Blog Jun 2024, Mendiratta et al

DISCUSSION THEMES

- How to analyze systems for metastability without actually executing the system?