CS 6650 Midterm Mastery

—

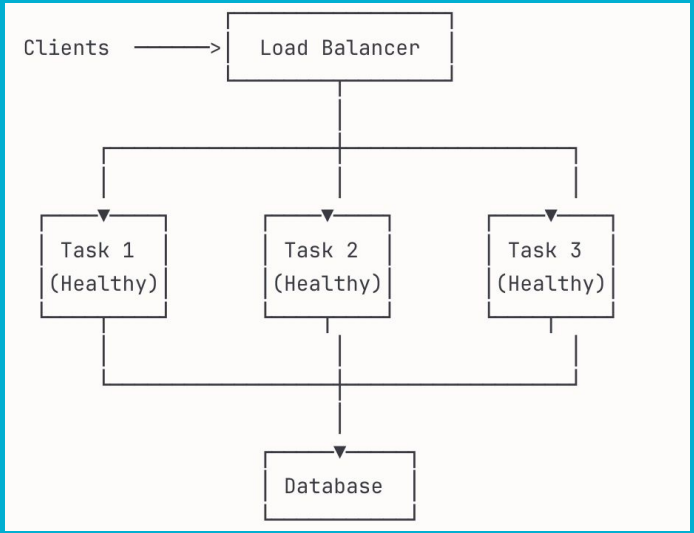
Meihao Cheng

Oct.20, 2025

Part 1: Learnings

1. **Horizontal Scaling = Redundancy + Load Balancing**

- **Key point: Stateless services** allow any request to go to any task
- **Why it matters:** No single point of failure



Part 1: Learnings

2. Three Resilience Patterns for Microservices

- **Timeout / Fail Fast**
  - Don't wait forever for slow services
- **Circuit Breaker**
  - Stop calling broken services temporarily
- **Bulkhead**
  - Isolate failures with separate thread pools

Part 1: Learnings

3. **Mindset Shift: From Preventing Failures → Working Despite Failures**

- **Before:** "Make systems that never fail"
- **After:** "Make systems that survive failures"
- **Multiple layers:** Infrastructure + Application code

Part 2: Experiment

- **Experiment goal:** To test system availability and recovery behavior during task failure and validate a resilience solution
- **System:** Reused HW6 product search service on AWS ECS
  - 1 ECR repository containing the Docker image
  - ECS Fargate tasks for container orchestration
  - 1 Application Load Balancer (ALB) for traffic distribution
  - Auto-scaling group for automatic capacity management
- **Load test:** 50 concurrent users, spawn rate 10, 5 minutes

The Problem - Minimal Redundancy

- **Configuration:** 2 tasks, 100% minimum healthy (default)
- **Test 1: Simultaneous failure**
  - Stopped 2 tasks simultaneously → 35 seconds downtime, 7.9% failures
- **Test 2: Cascading failures**
  - Stopped 1 task, and waited for the replacement task to be healthy, stopped another task → 0 downtime, but…
    - 60-90 seconds vulnerability window
    - System survival depends on timing (lucky!)

Test 1: Simultaneous Failure

Locust Test Report

Download the Report

During: 10/19/2025, 1:05:55 PM - 10/19/2025, 1:10:55 PM (5 minutes)
Target Host: http://product-search-service-alb-201966303.us-west-2.elb.amazonaws.com
Script: locustfile.py

Request Statistics

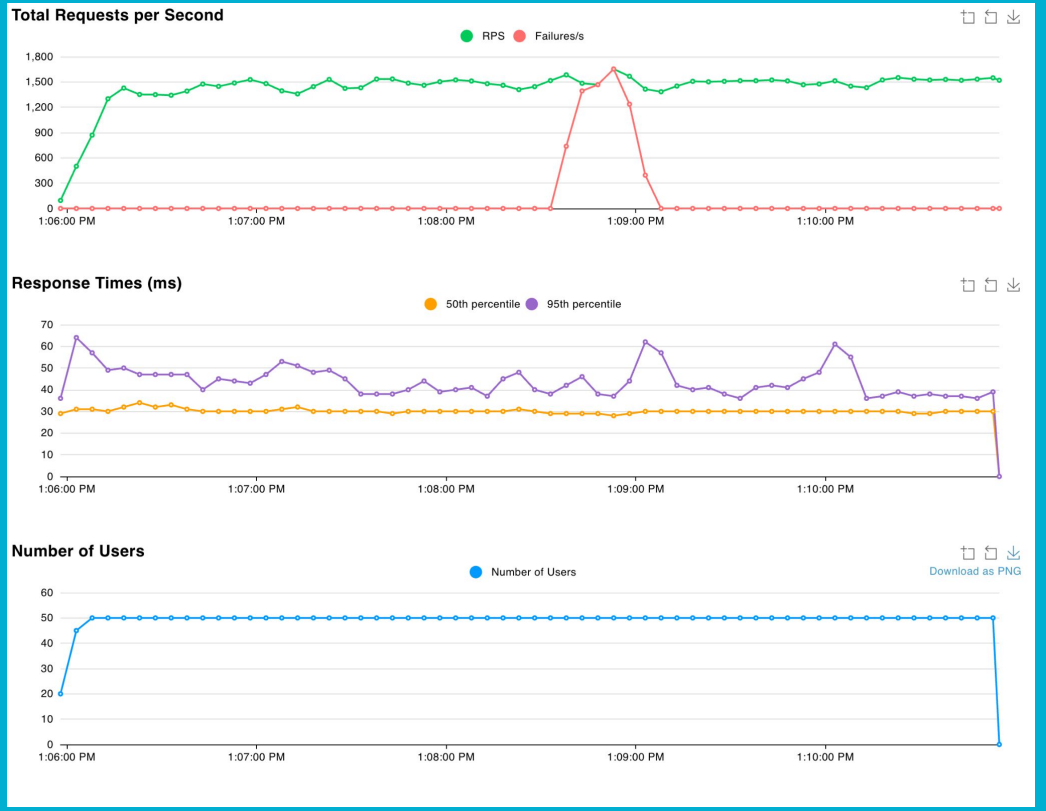
Type	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	/products/search	437130	34402	32.67	15	329	1394.18	1456.87	114.65
	Aggregated	437130	34402	32.67	15	329	1394.18	1456.87	114.65

Response Time Statistics

Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	/products/search	30	31	32	34	37	44	100	330
	Aggregated	30	31	32	34	37	44	100	330

Failures Statistics

# Failures	Method	Name	Message
34402	GET	/products/search	CatchResponseError('Got status code 503')



Test 2: Cascading Failure

Locust Test Report

Download the Report

During: 10/19/2025, 3:47:32 PM - 10/19/2025, 3:52:32 PM (5 minutes)
Target Host: http://product-search-service-alb-201966303.us-west-2.elb.amazonaws.com
Script: locustfile.py

Request Statistics

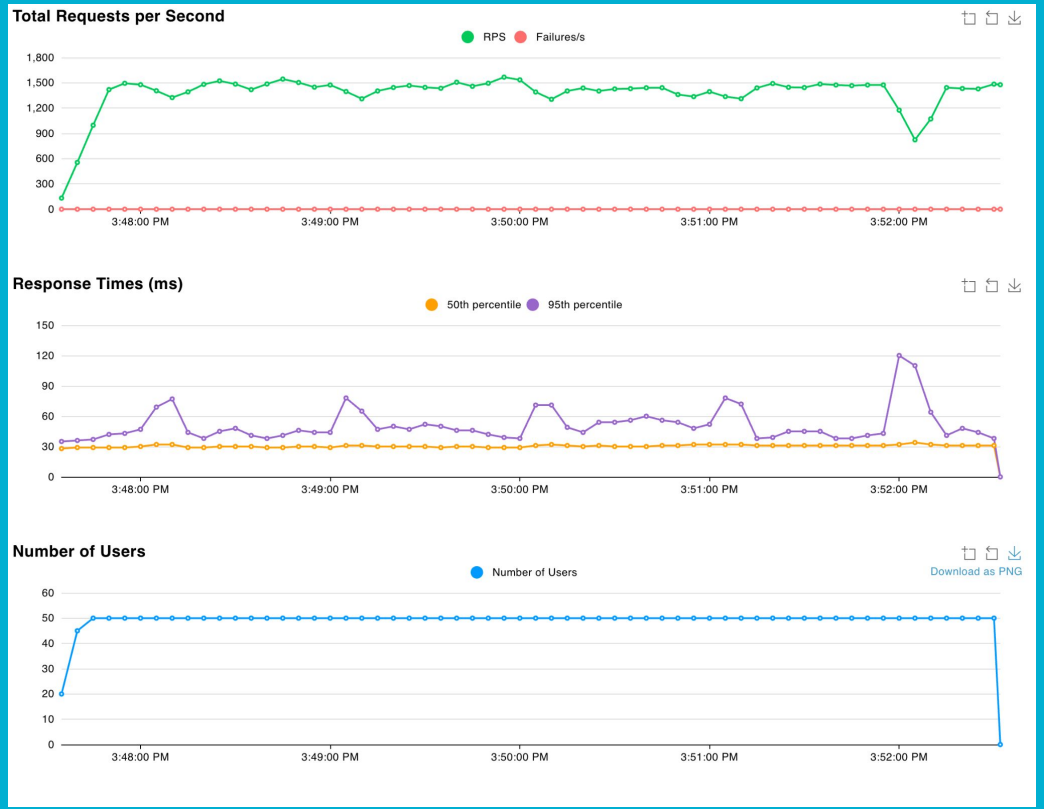
Type	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	/products/search	420729	0	33.91	17	951	1488.49	1402.18	0
	Aggregated	420729	0	33.91	17	951	1488.49	1402.18	0

Response Time Statistics

Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	/products/search	30	31	33	34	39	50	110	950
	Aggregated	30	31	33	34	39	50	110	950

Failures Statistics

# Failures	Method	Name	Message



The Solution - Over-Provisioning

- **Configuration:** 3 tasks, 66% minimum healthy (at least 2 tasks healthy)
- **Test 1: Rapid sequential failures**
  - Stopped 2 tasks rapidly within 5 seconds → 0 downtime
- **Test 2: Rolling sequential failures**
  - Stopped all 3 tasks sequentially with 20-30s interval → 0 downtime, 30s vulnerability window
- **Test 3: Simultaneous failures**
  - Stopped all 3 tasks simultaneously → 35s downtime (unrealistic scenario)

Test 1: Rapid Sequential Failure

Locust Test Report

Download the Report

During: 10/19/2025, 1:55:23 PM - 10/19/2025, 2:00:23 PM (5 minutes)
Target Host: http://product-search-service-alb-201966303.us-west-2.elb.amazonaws.com
Script: locustfile.py

Request Statistics

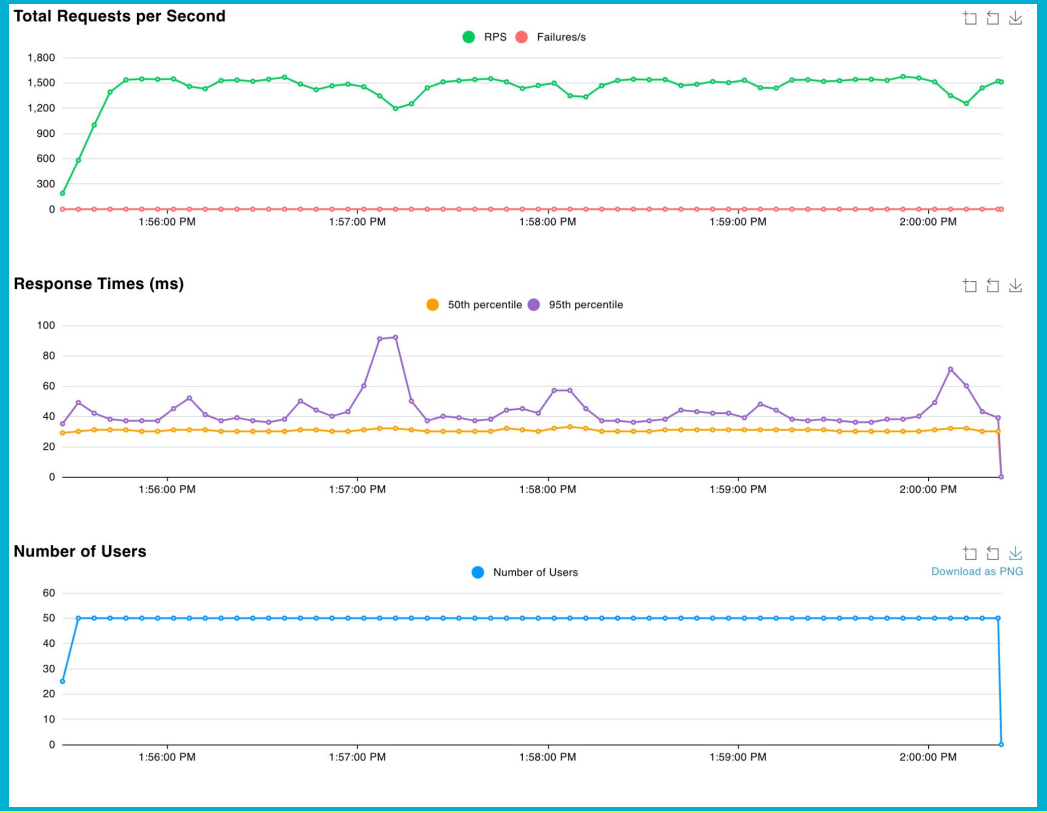
Type	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	/products/search	437400	0	32.85	17	567	1516.56	1457.72	0
	Aggregated	437400	0	32.85	17	567	1516.56	1457.72	0

Response Time Statistics

Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	/products/search	31	32	33	34	36	43	98	570
	Aggregated	31	32	33	34	36	43	98	570

Failures Statistics

# Failures	Method	Name	Message



Test 2: Rolling Sequential Failure

Locust Test Report

Download the Report

During: 10/19/2025, 2:15:39 PM - 10/19/2025, 2:20:39 PM (5 minutes)
Target Host: http://product-search-service-alb-201966303.us-west-2.elb.amazonaws.com
Script: locustfile.py

Request Statistics

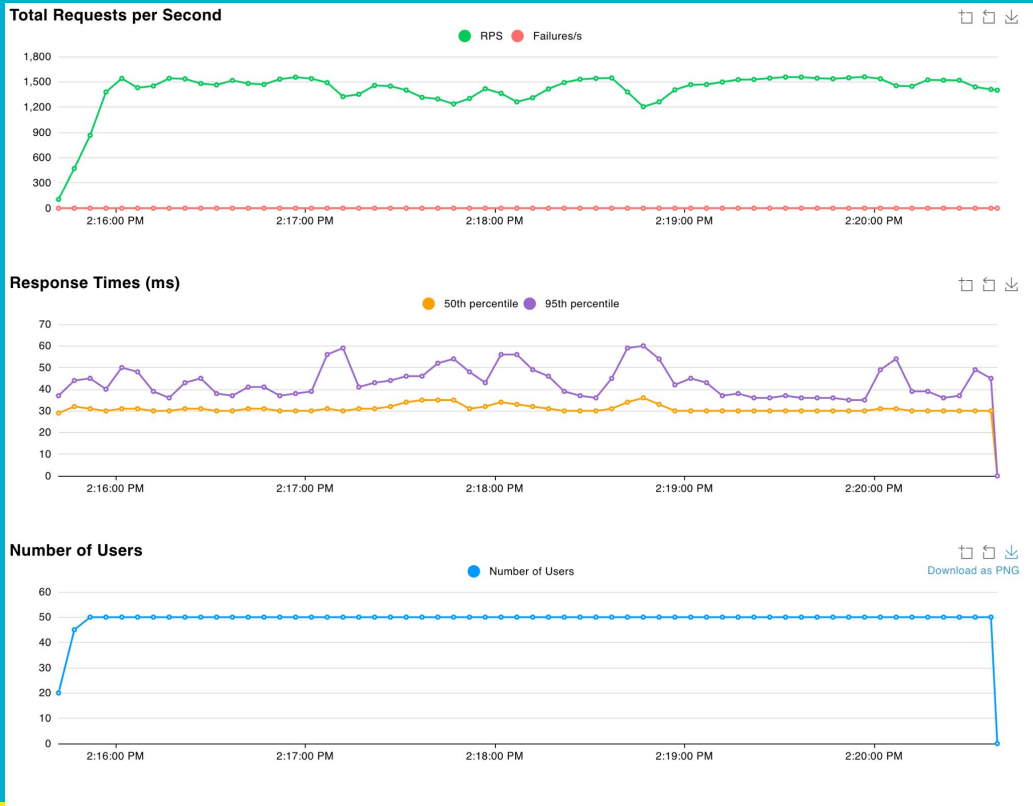
Type	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	/products/search	430334	0	33.23	17	393	1490.78	1434.19	0
	Aggregated	430334	0	33.23	17	393	1490.78	1434.19	0

Response Time Statistics

Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	/products/search	31	32	33	34	38	44	100	390
	Aggregated	31	32	33	34	38	44	100	390

Failures Statistics

# Failures	Method	Name	Message



Test 3: Simultaneous Failure

Locust Test Report

Download the Report

During: 10/19/2025, 2:35:42 PM - 10/19/2025, 2:40:42 PM (5 minutes)
Target Host: http://product-search-service-alb-201966303.us-west-2.elb.amazonaws.com
Script: locustfile.py

Request Statistics

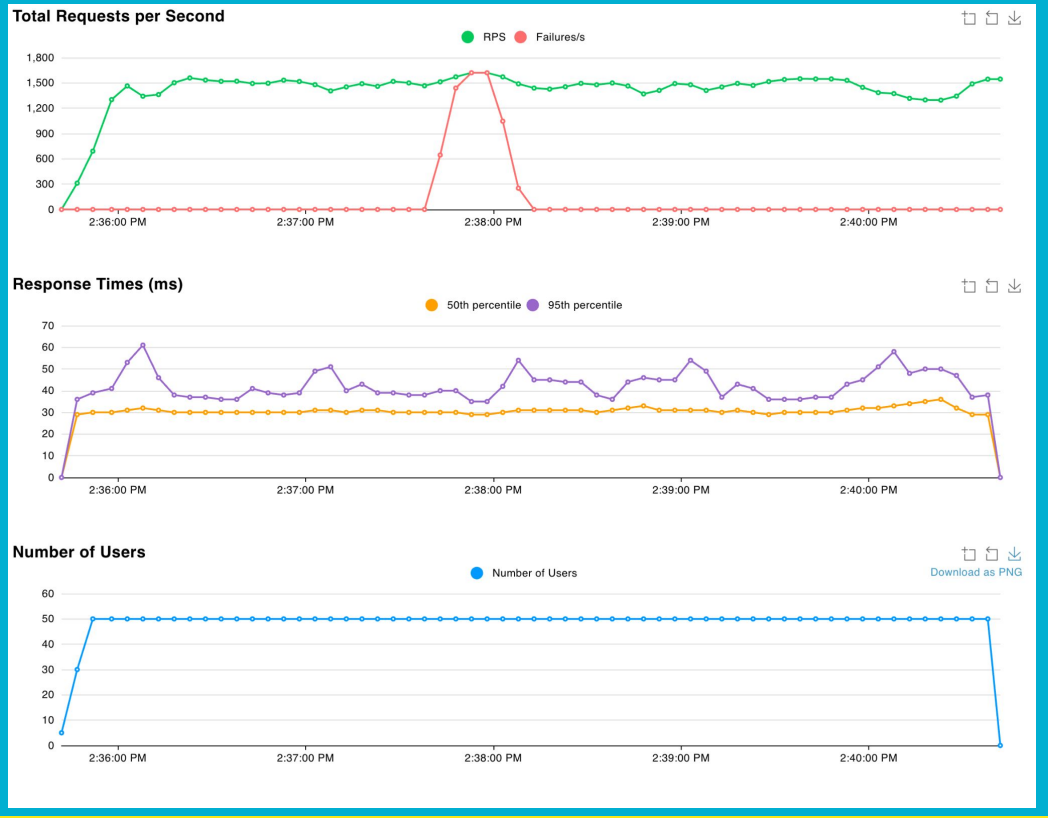
Type	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	/products/search	436954	33116	32.83	17	413	1385.06	1456.26	110.37
	Aggregated	436954	33116	32.83	17	413	1385.06	1456.26	110.37

Response Time Statistics

Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	/products/search	31	32	33	34	37	43	95	410
	Aggregated	31	32	33	34	37	43	95	410

Failures Statistics

# Failures	Method	Name	Message
33116	GET	/products/search	CatchResponseError('Got status code 503')



Results Comparison

Test Scenario	Configuration	Fail Requests	Downtime	Key Insights
Problem: Simultaneous	2 tasks, 100% min	7.9%	35s	No redundancy buffer
Problem: Cascading	**2 tasks, 100% min**	**0%**	**0s**	**60-90s vulnerability window**
Solution: Rapid Failure	3 tasks, 66% min	0%	0s	Proactive capacity maintenance (66%min)
Solution:Rolling Failure	**3 tasks, 66% min**	**0%**	**0s**	**30s vulnerability window**
Solution: Simultaneous	3 tasks, 66% min	7.6%	35s	Unrealistic edge case

Cost-Benefit & Takeaways

- Cost: +1 task = +$15/month (50% increase)
- Benefits:
  - Tolerates failures with less vulnerability
  - Shorter vulnerability window (30s vs 60-90s)
  - More resilient to realistic failure patterns
- Key Learnings:
  - Over-provisioning (N+1) reduces vulnerability windows
  - Lower minimum healthy % with more tasks = better buffer
  - Proactive capacity > reactive auto-scaling
  - Real reliability needs multiple layers of protection

Future Improvements - Zero Downtime

1. Multi-AZ Deployment → Distribute tasks across availability zones → Survives entire zone failures
2. Pre-Warmed Standby Tasks → Keep warm tasks ready (no cold start) → Instant replacement (0s instead of 60-90s)
3. Blue-Green Deployment → Run 2 parallel environments → Instant failover between them
4. Cross-Region Redundancy → Deploy in multiple AWS regions → Survives regional disasters

Trade-off: Higher cost + complexity, but zero vulnerability windows

Thank you