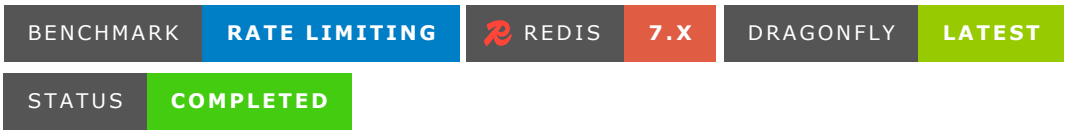
📊 Báo cáo Benchmark Toàn diện: Redis vs Dragonfly

Hệ thống Rate Limiting dựa trên In-Memory Store



Ngày thực hiện: Tháng 1-2, 2026
Tác giả: PhongHV **Phiên bản**: 1.0

📑 Mục lục

1. 🎯 Tổng quan dự án
2. 🏗️ Kiến trúc hệ thống & Testbed
3. 🔬 Phương pháp Benchmark
4. 📈 Kết quả chi tiết: Redis Single Node
5. 📈 Kết quả chi tiết: Dragonfly Single Node
6. ⚔️ So sánh trực tiếp Redis vs Dragonfly
7. 🔍 Phân tích chuyên sâu
8. 🚀 Khuyến nghị Production
9. ✅ Kết luận

📌 Executive Summary

🏆 Kết quả chính

Metric	Redis	Dragonfly	Cải thiện
Max RPS (Uniform)	20,000	27,500	🔺 **+37.5%**
Max RPS (Hot Key)	17,500	22,500	🔺 **+28.6%**
P99 Latency @17.5k	160ms	24ms	🔺 **6.6x faster**
Failure Pattern	Collapse đột ngột	Graceful degradation	✅ **Resilient**

```
╔═══════════════════════════════════════════════════════════════════════════════╗
                                🎯 BOTTOM LINE
╠═══════════════════════════════════════════════════════════════════════════════╣

   Dragonfly cho thấy hiệu năng vượt trội so với Redis trong mọi kịch bản:
```

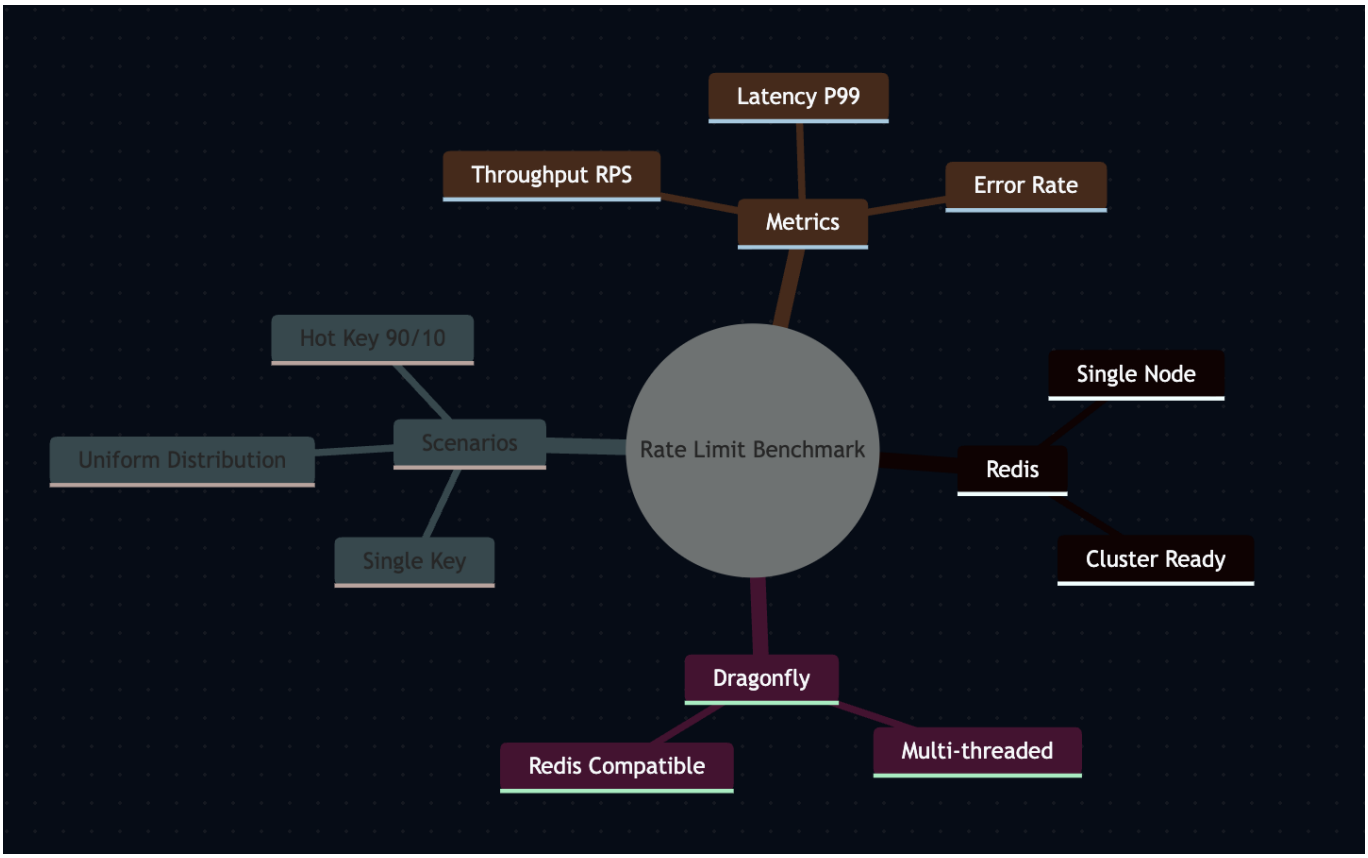
```
╠═══════════════════════════════════════════════════════════════════════════════╣
║                                                                               ║
║   ✅ Throughput cao hơn 28–37%                                                ║
║                                                                               ║
║   ✅ Latency thấp hơn 2–7 lần                                                 ║
║                                                                               ║
║   ✅ Failure pattern "mềm" hơn, cho phép hệ thống phản ứng kịp thời           ║
║                                                                               ║
║                                                                               ║
║   👉 KHUYẾN NGHỊ: Nếu đang dùng Redis với >15k RPS, hãy cân nhắc Dragonfly    ║
║                                                                               ║
╚═══════════════════════════════════════════════════════════════════════════════╝
```

---

1. 🎯 Tổng quan dự án

1.1 Mục tiêu nghiên cứu

Dự án này nhằm **đánh giá toàn diện năng lực** của hai hệ thống in-memory store phổ biến — **Redis** và **Dragonfly** — khi được sử dụng làm backend cho hệ thống Rate Limiting trong môi trường microservices.



Câu hỏi nghiên cứu chính:

#	Câu hỏi	Đã trả lời

#	Câu hỏi	Đã trả lời
1	Throughput tối đa mà mỗi hệ thống có thể xử lý là bao nhiêu?	✅
2	Hệ thống phản ứng như thế nào với các pattern traffic khác nhau?	✅
3	Đâu là ngưỡng an toàn cho môi trường production?	✅
4	Dragonfly có phải là giải pháp thay thế tốt hơn Redis không?	✅

1.2 Phạm vi benchmark

Khía cạnh	Phạm vi
Backends được test	Redis 7.x (Single Node), Dragonfly (Single Node)
Dải RPS	10,000 - 30,000 RPS
Kịch bản phân phối key	Single Key, 100 Keys (Uniform), 100 Keys (Hot Key 90/10)
Thời gian mỗi step	30-60 giây
Metrics thu thập	Throughput, Latency (Avg, P95, P99), Error Rate, Resource Usage

1.3 Tại sao Rate Limiting quan trọng?

Rate Limiting là tuyến phòng thủ đầu tiên bảo vệ hệ thống khỏi:

Mối đe dọa	Mô tả	Impact nếu không có Rate Limit
🔴 **DDoS/Abuse**	Traffic độc hại làm quá tải	Service down, revenue loss
🟠 **Unfair Usage**	Một tenant chiếm hết tài nguyên	Poor UX cho users khác
🟡 **Cost Overrun**	Vượt quota third-party APIs	Chi phí phát sinh không kiểm soát
🟢 **Cascade Failure**	Downstream quá tải	Toàn hệ thống sụp đổ

---

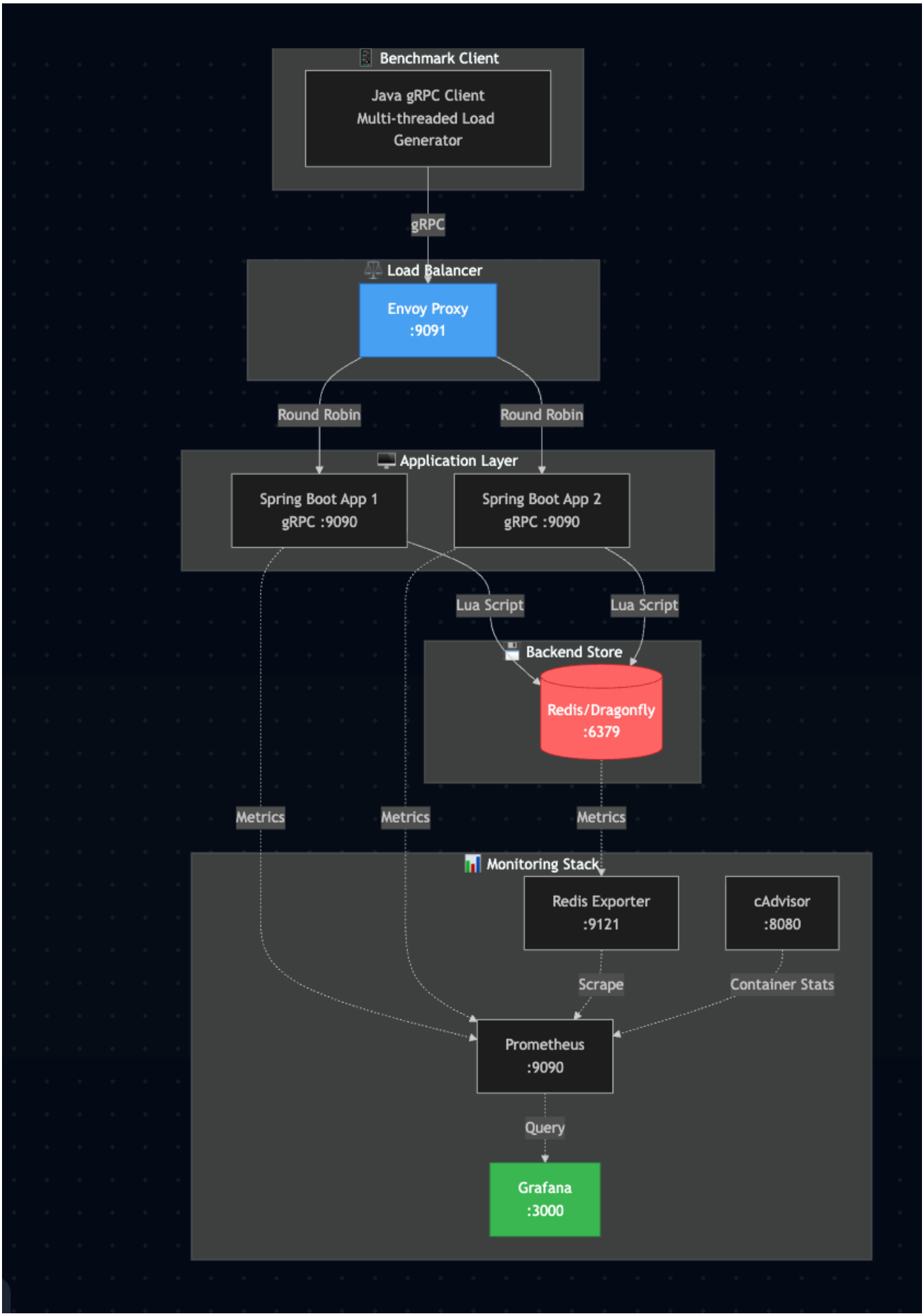
2. 🏗️ Kiến trúc hệ thống & Testbed

2.1 Cấu hình phần cứng

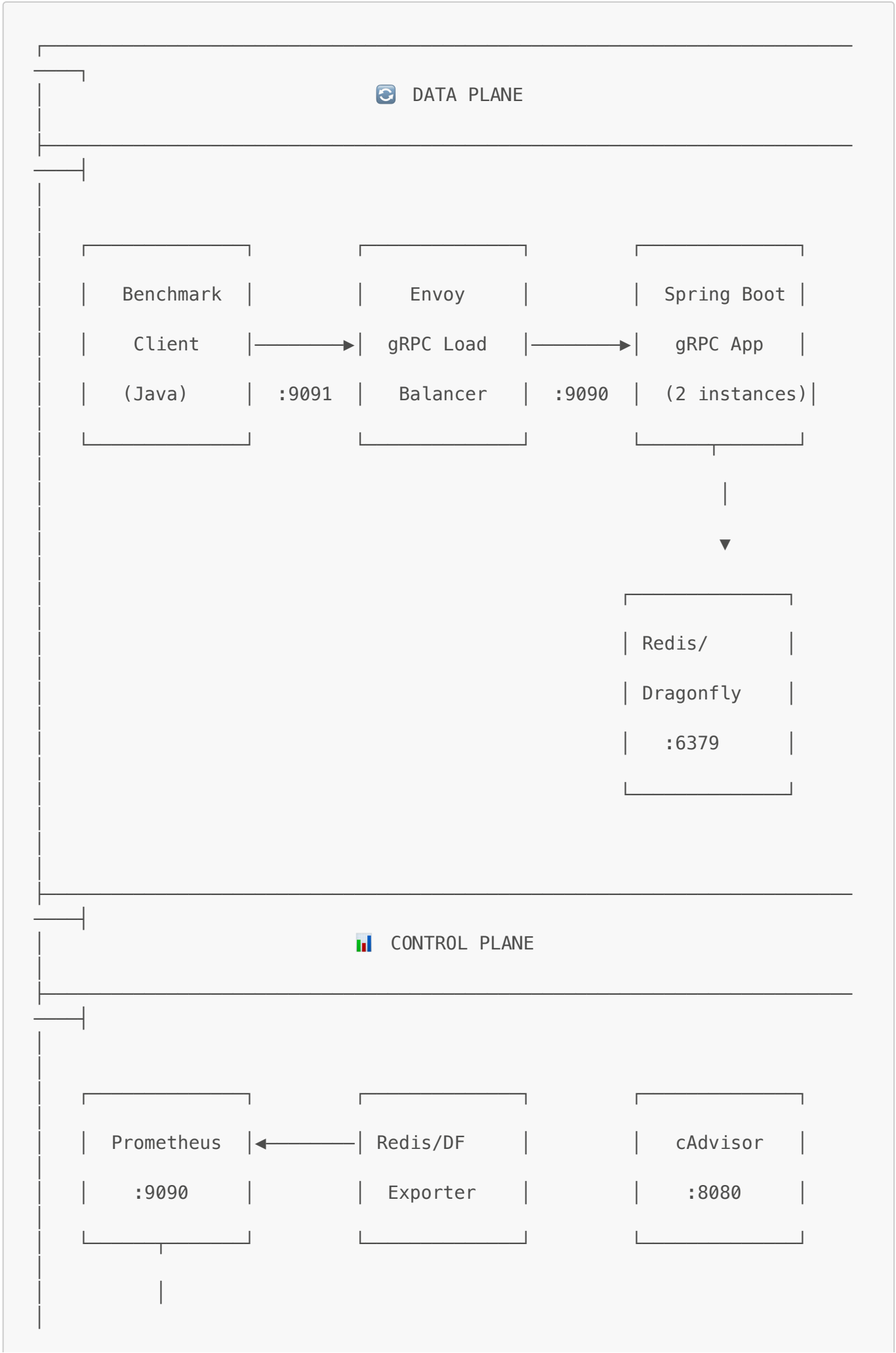
🖥️ Thông số	Giá trị
Hệ điều hành	macOS 26.2 (Build 25C56)
CPU	Apple M4
Số core	10 cores (logic)
RAM	24 GB
Docker Engine	28.0.4

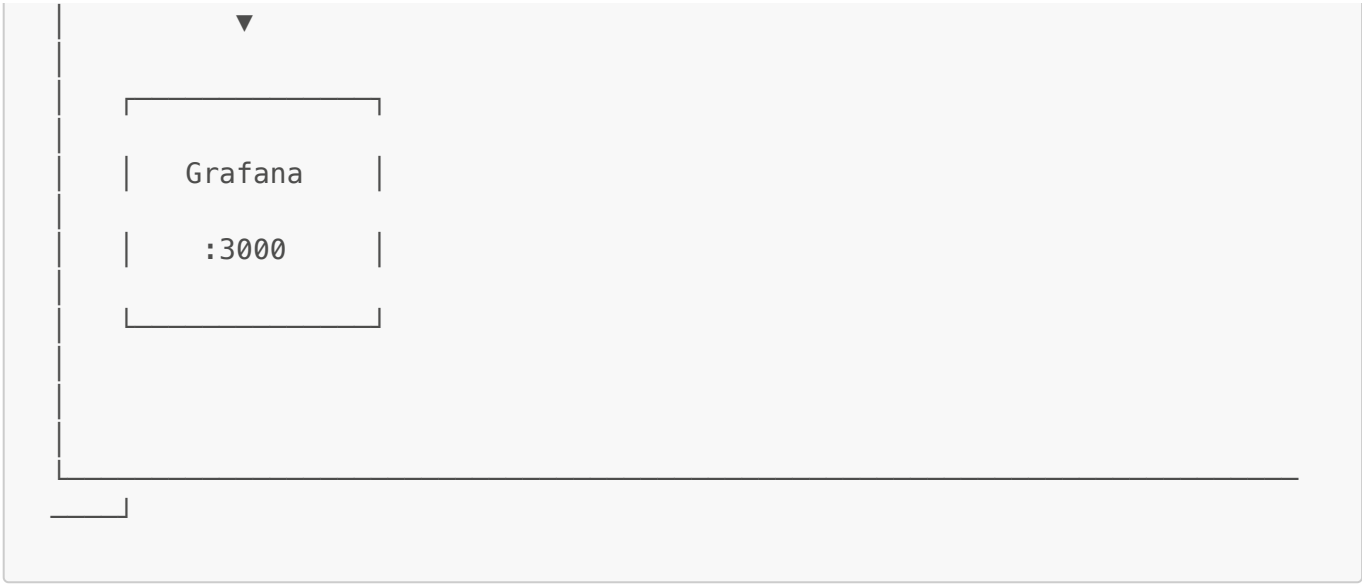
> ⚠ **Lưu ý**: Tất cả thành phần (Backend, Proxy, App, Monitoring) đều chạy trên cùng một máy qua Docker Compose. Đây là môi trường "best-case" về network latency.

2.2 Kiến trúc tổng quan



2.3 Chi tiết Data Flow

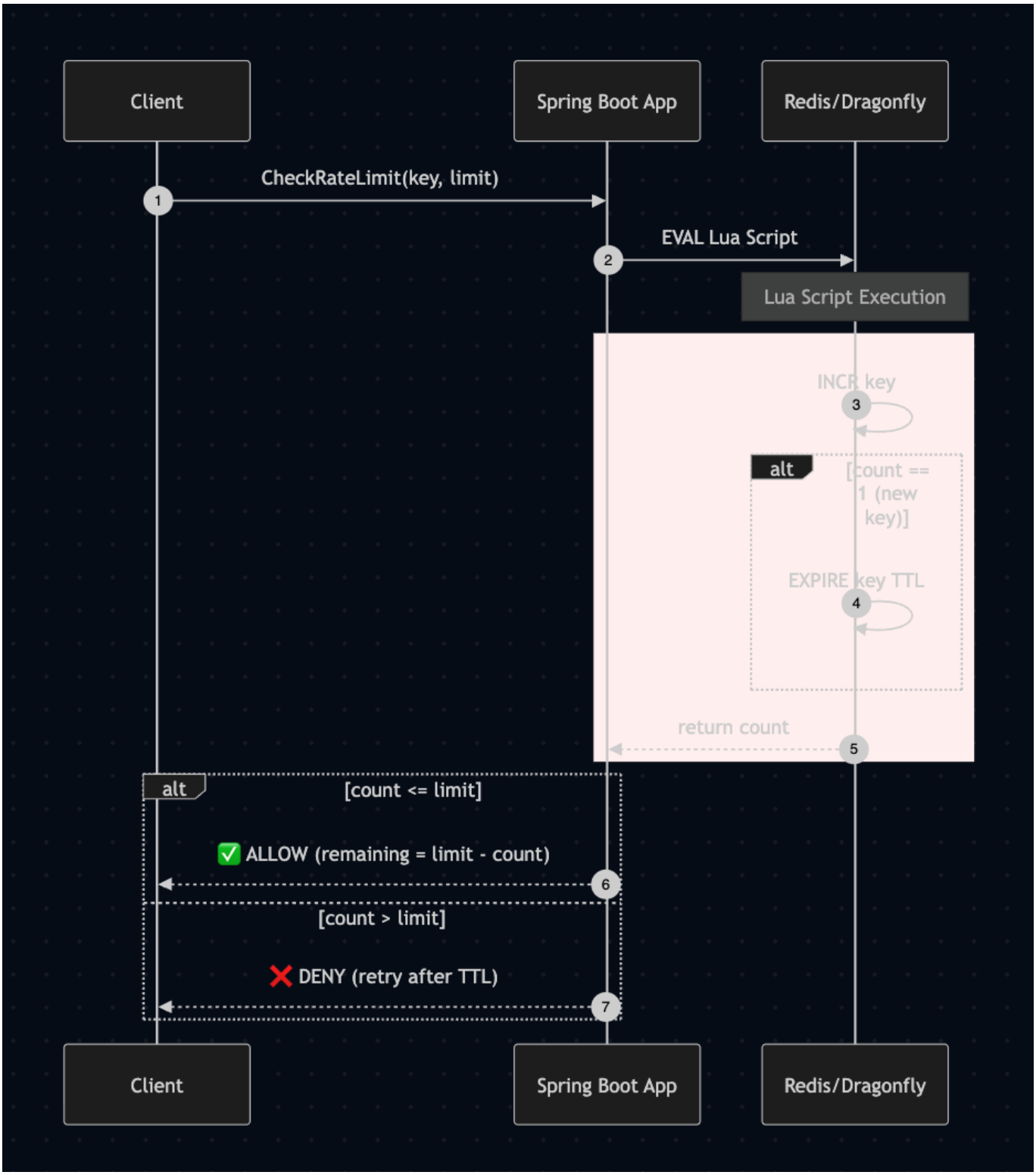

DATA PLANE
Benchmark Client (Java)
:9091
Envoy gRPC Load Balancer
:9090
Spring Boot gRPC App (2 instances)
Redis/ Dragonfly :6379
CONTROL PLANE
Prometheus :9090
Redis/DF Exporter
cAdvisor :8080



2.4 Chi tiết các thành phần

Thành phần	Công nghệ	Vai trò	Port
📱 **Benchmark Client**	Java 21 + gRPC	Sinh tải theo target RPS	-
⚖️ **Load Balancer**	Envoy	Round-robin gRPC requests	9091
🖥️ **Application**	Spring Boot 3 + WebFlux + gRPC	Xử lý logic rate limit	9090
📜 **Rate Limit Logic**	Lua Script (INCR + EXPIRE)	Atomic counter	-
💾 **Backend Store**	Redis 7.x / Dragonfly	Lưu trữ counter	6379
📊 **Monitoring**	Prometheus + Grafana	Thu thập & visualize metrics	9090, 3000

2.5 Rate Limit Algorithm (Fixed Window Counter)

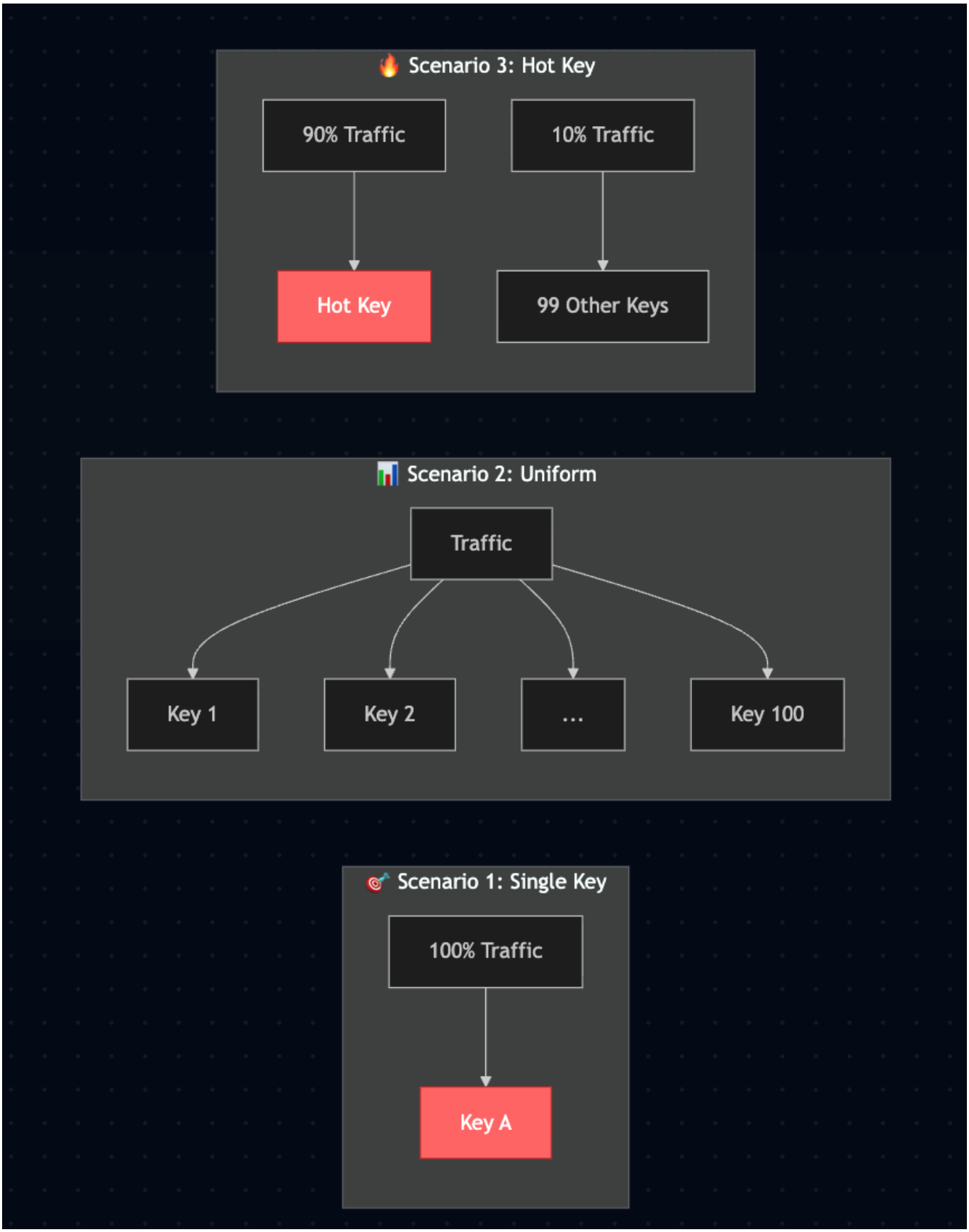


---

3. 🔬 Phương pháp Benchmark

3.1 Các kịch bản test

Chúng tôi thiết kế **3 kịch bản** mô phỏng các pattern traffic thực tế:



Kịch bản 1: Single Key (Best Case)

Đặc điểm	Mô tả
Pattern	100% traffic → 1 key duy nhất
Mô phỏng	Extreme cache locality, minimal memory access

Đặc điểm	Mô tả
Kỳ vọng	Throughput cao nhất (CPU cache hit rate tối ưu)
Thực tế	Hiếm gặp trong production, để đo ceiling

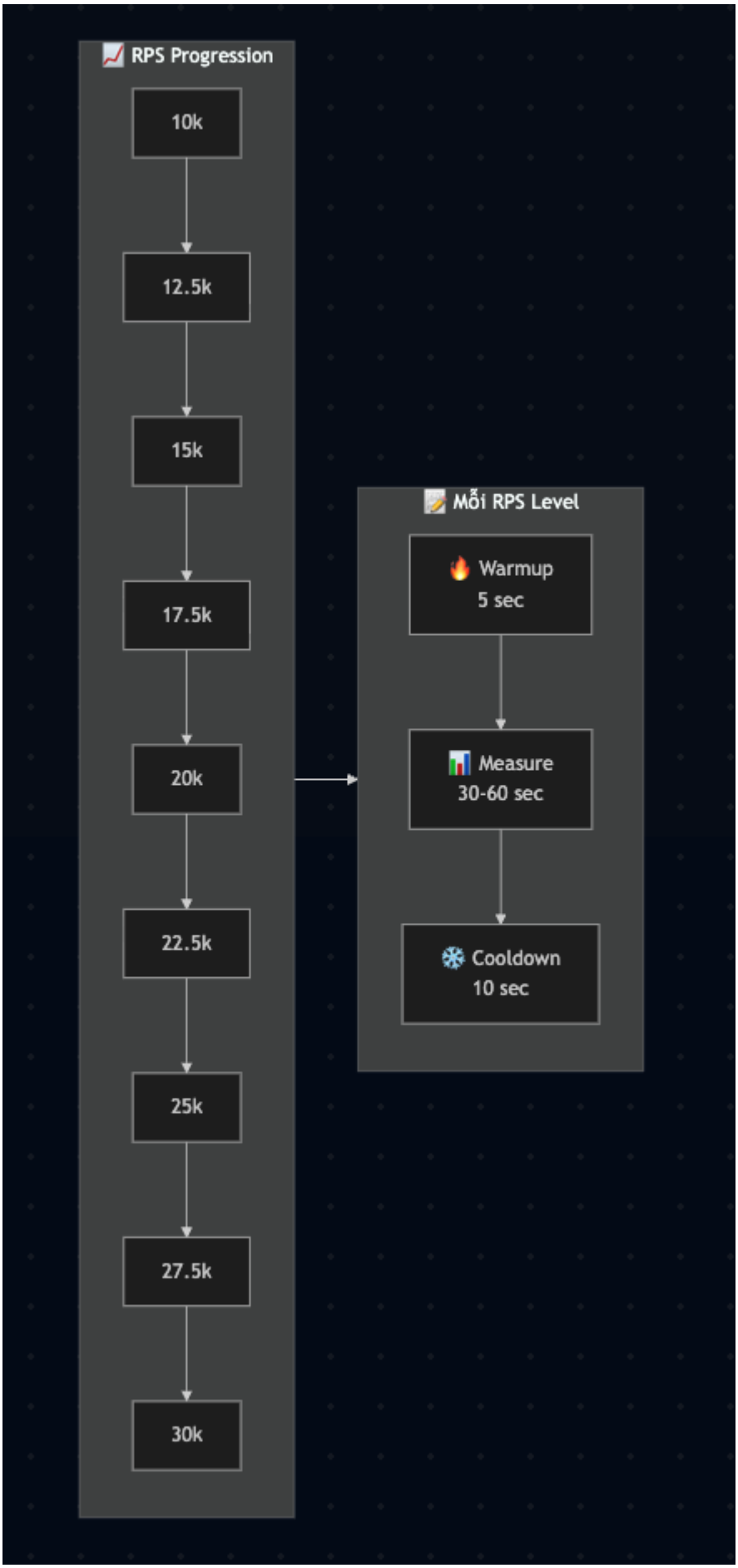
Kịch bản 2: Hundred Keys - Uniform Distribution

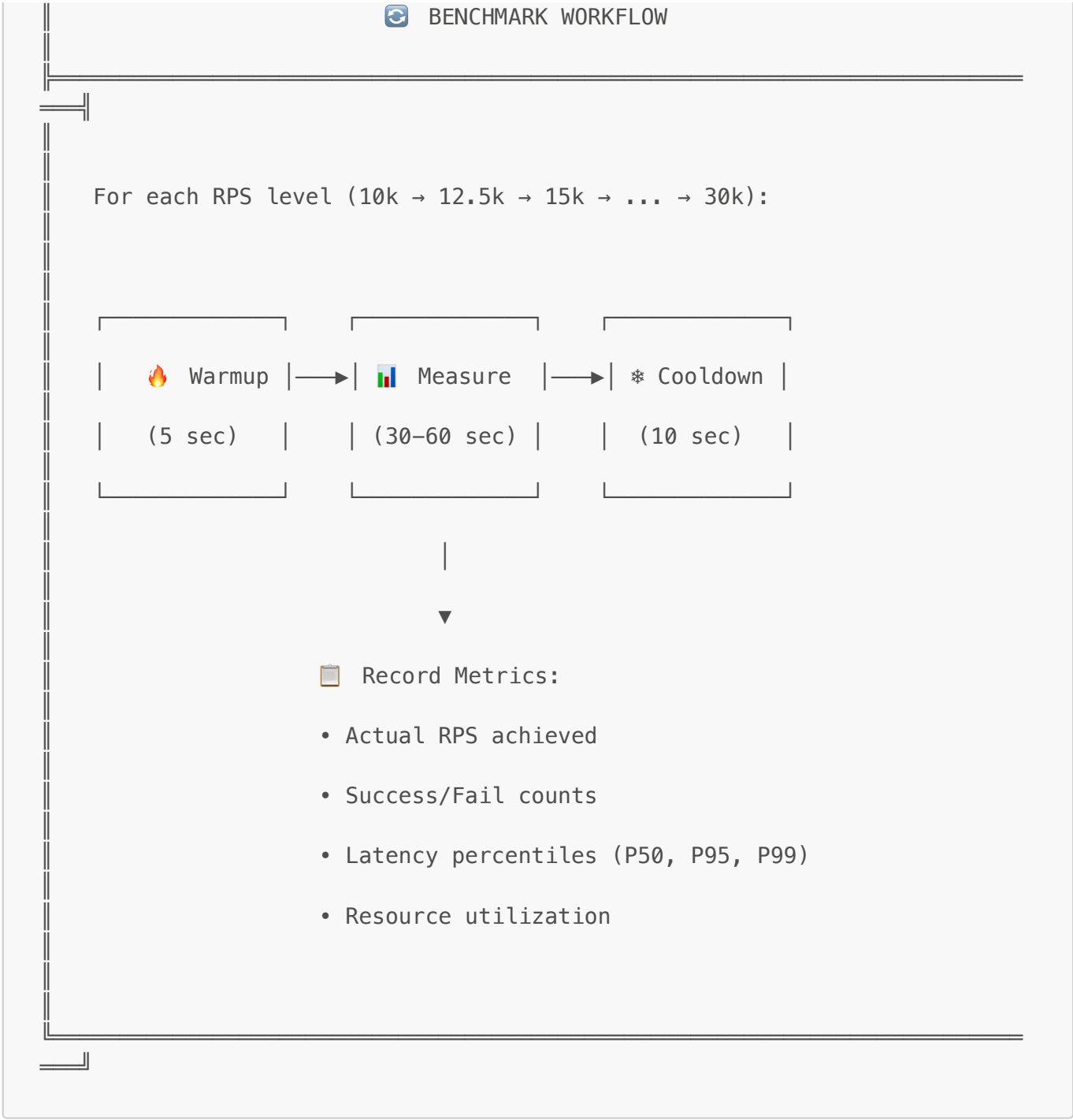
Đặc điểm	Mô tả
Pattern	Traffic phân bổ đều cho 100 keys
Mô phỏng	Multi-tenant API với traffic cân bằng
Kỳ vọng	Throughput thấp hơn do context switching
Thực tế	Gần với production của hệ thống well-designed

Kịch bản 3: Hundred Keys - Hot Key (90/10)

Đặc điểm	Mô tả
Pattern	90% traffic → 1 key "hot", 10% → 99 keys còn lại
Mô phỏng	Viral event, popular user, API abuse
Kỳ vọng	Throughput thấp nhất, failure nguy hiểm
Thực tế	Worst-case scenario cần plan for

3.2 Quy trình benchmark


RPS Progression
10k
12.5k
15k
17.5k
20k
22.5k
25k
27.5k
30k
Mỗi RPS Level
Warmup
5 sec
Measure
30-60 sec
Cooldown
10 sec



3.3 Metrics thu thập & ngưỡng cảnh báo

Metric	Ý nghĩa	🟢 Good	🟡 Warning	🔴 Critical
Actual RPS	Throughput thực tế	≥ 98% target	95-98%	< 95%
Fail %	Tỷ lệ request thất bại	< 0.1%	0.1-1%	> 1%
Avg Latency	Độ trễ trung bình	< 10ms	10-50ms	> 50ms
P95 Latency	95th percentile	< 50ms	50-100ms	> 100ms
P99 Latency	99th percentile	< 100ms	100-200ms	> 200ms

---

4. 📈 Kết quả chi tiết: Redis Single Node

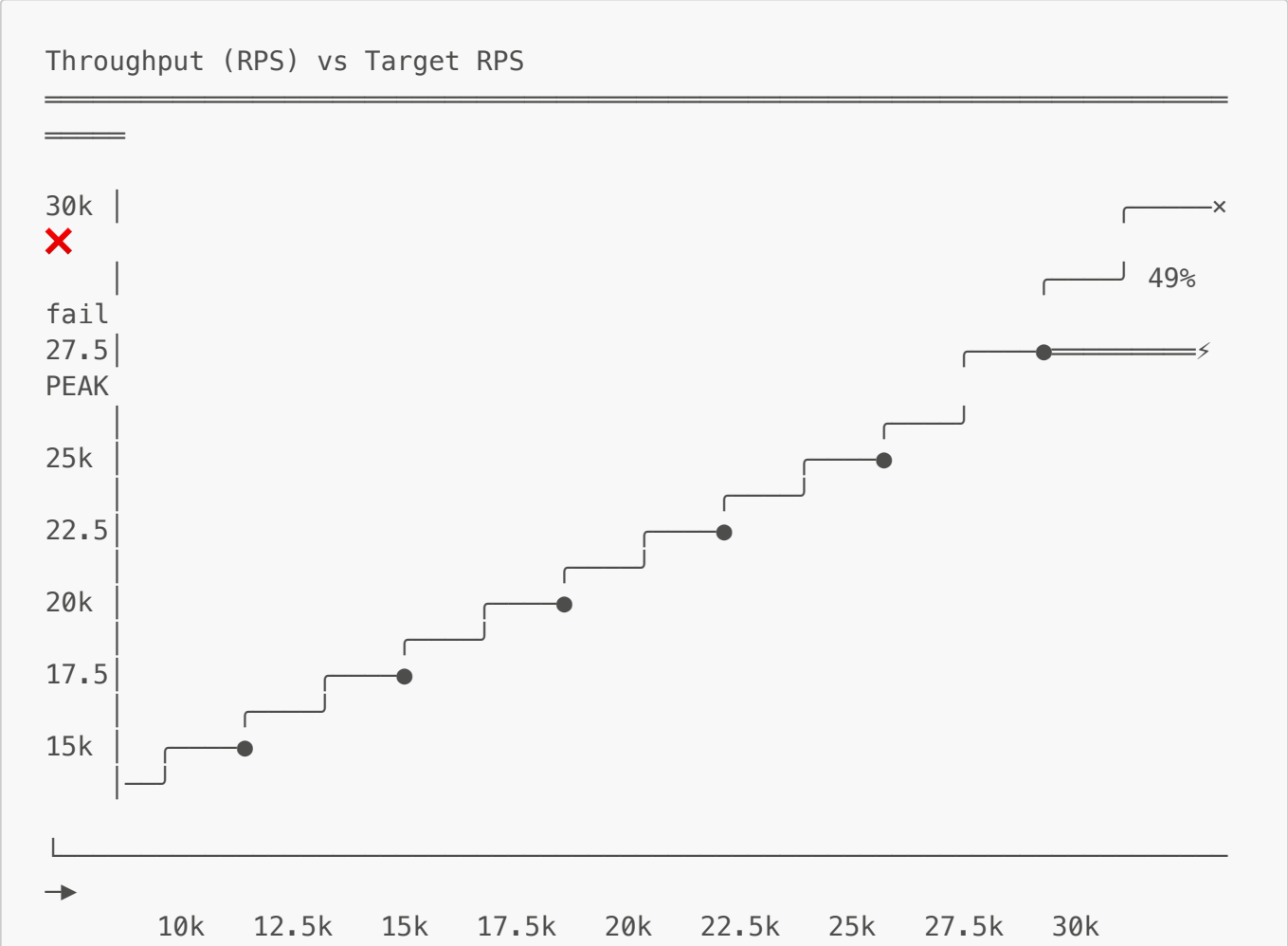
4.1 Kịch bản Single Key

> 📅 **Test Date**: January 27, 2026
> ⏱️ **Duration**: 30 giây/step

▶ 📊 **Bảng kết quả chi tiết (Click để mở)**

Target RPS	Actual RPS	Success	Fail	Fail %	Avg Latency	P95	P99	Status
10,000	10,000	300,000	0	0%	0.94ms	1.74ms	**6.26ms**	✅
12,500	12,500	374,987	0	0%	0.98ms	1.54ms	**3.71ms**	✅
15,000	14,999	436,764	13,225	2.9%	22.5ms	19.9ms	**702ms**	⚠
17,500	17,499	524,986	0	0%	3.17ms	11.0ms	**42.1ms**	✅
20,000	19,999	564,317	35,664	5.9%	67.5ms	390ms	**667ms**	⚠
22,500	22,499	646,903	28,081	4.2%	42.6ms	190ms	**873ms**	⚠
25,000	24,997	749,964	0	0%	19.5ms	107ms	**234ms**	⚡
27,500	**27,456**	**824,930**	**0**	**0%**	**19.4ms**	**92.3ms**	**137ms**	⚡
30,000	28,718	458,166	441,809	49.1%	751ms	2.83s	**3.83s**	❌

📈 **Biểu đồ Throughput & Latency**



```
Target

Legend: ● Achieved RPS    × Failed    ═══ Optimal zone    ⚡ Peak
performance
```

```
P99 Latency (ms) - Log Scale
═══════════════════════════════════════════════════════════════════════
═══

4000ms┤
      ▓▓▓▓▓▓▓▓
            │
      ▓▓▓▓▓▓▓▓
            │
      ▓▓▓▓▓▓▓▓
1000ms┤
      ▓▓▓▓▓▓▓▓
            │                  ▒▒▒▒
 500ms┤                  ▒▒▒▒  ▒▒▒▒  ▒▒▒▒
            │             ▒▒▒▒  ▒▒▒▒  ▒▒▒▒  ▒▒▒▒
 200ms┤                  ▒▒▒▒  ▒▒▒▒  ▒▒▒▒  ▒▒▒▒  ▒▒▒▒  ▒▒▒▒
 100ms┤                  ▒▒▒▒  ▒▒▒▒  ▒▒▒▒  ▒▒▒▒  ▒▒▒▒  ▒▒▒▒
  50ms┤       ▒▒▒▒       ▒▒▒▒  ▒▒▒▒  ▒▒▒▒  ▒▒▒▒  ▒▒▒▒  ▒▒▒▒
  10ms┤ ▒▒▒▒  ▒▒▒▒                                  ▒▒▒▒  ▒▒▒▒
      └──────────────────────────────────────────────────────────►
         10k  12.5k  15k  17.5k  20k  22.5k  25k  27.5k  30k

Legend: ▒▒▒▒ Healthy (<100ms)   ▒▒▒▒ Degraded (100-1000ms)   ▓▓▓▓ Critical
(>1s)
```

📸 Grafana Dashboard Screenshots

Overview - Throughput, Latency & Container Resources

Figure 4.1: Clear ramp-up pattern từ 10k đến 30k RPS. Dramatic spike visible tại 30k RPS step.

JVM & GC Pressure + Redis Performance

Figure 4.2: Redis latency spike tại saturation point confirms Redis là bottleneck.

Redis Resources & Envoy Load Balancer

Figure 4.3: Envoy RPS by Status cho thấy error rate spikes rõ ràng.

🔍 Nhận xét

Observation	Detail
🏆 **Peak Performance**	27,500 RPS với P99 < 140ms
⚠ **Anomaly**	15k, 20k, 22.5k có failure tạm thời → JVM warmup/GC
💥 **Breaking Point**	30,000 RPS → 49% failure, hệ thống sụp đổ

4.2 Kịch bản Hundred Keys - Uniform

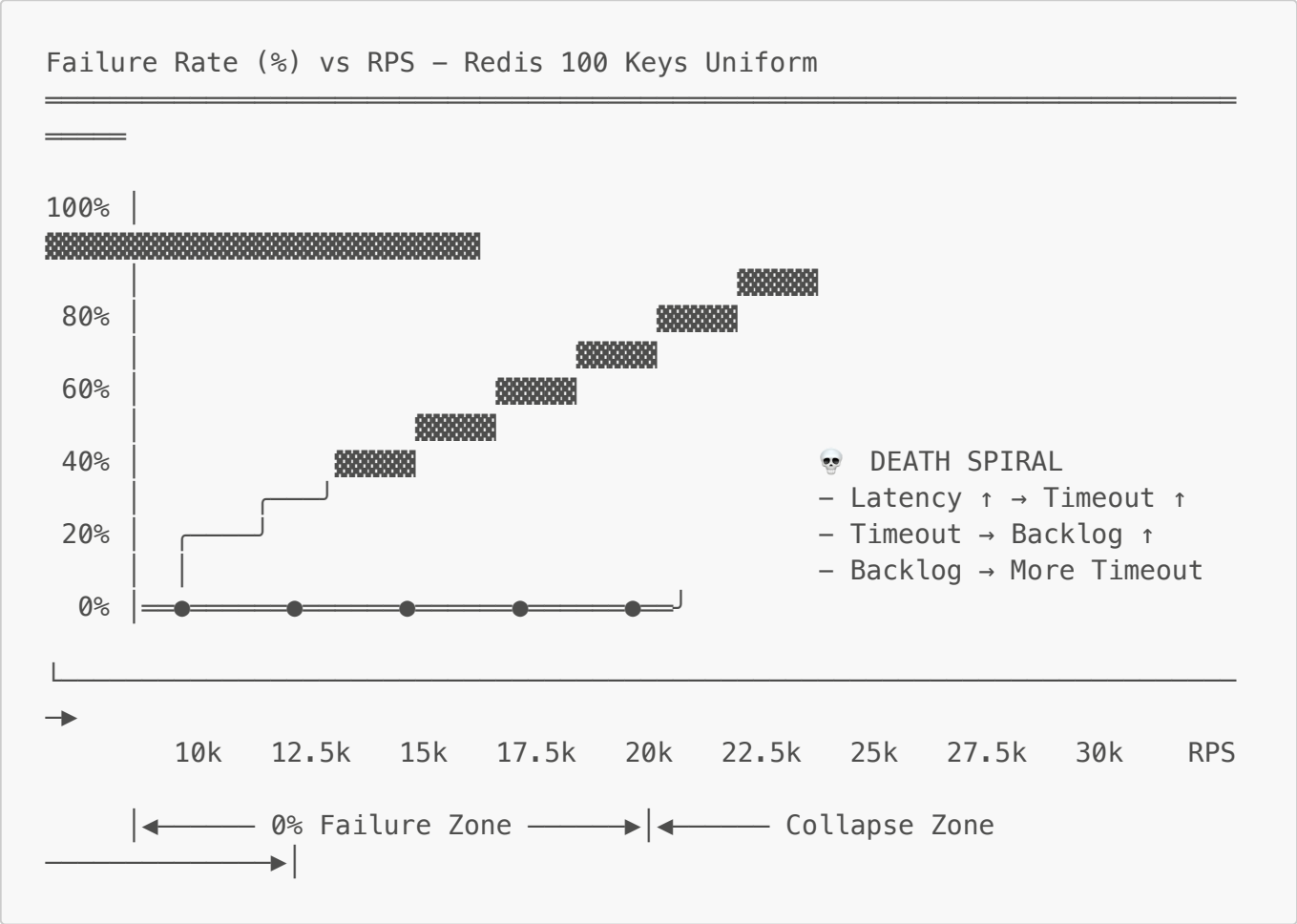
> 📅 **Test Date**: January 28, 2026
> ⏱ **Duration**: 60 giây/step (extended)

▶ 📊 **Bảng kết quả chi tiết (Click để mở)**

Target RPS	Actual RPS	Success	Fail	Fail %	Avg Latency	P95	P99	Status
10,000	10,000	600,000	0	0%	2.37ms	12.9ms	**29.8ms**	✅
12,500	12,500	749,988	0	0%	6.20ms	30.5ms	**41.5ms**	✅
15,000	15,000	899,998	0	0%	8.82ms	37.3ms	**51.1ms**	✅
17,500	17,499	1,049,975	0	0%	12.3ms	45.1ms	**60.0ms**	✅
20,000	**19,990**	**1,199,995**	**0**	**0%**	**23.5ms**	**77.1ms**	**110ms**	✅
22,500	22,159	990,129	359,867	**26.6%**	418ms	1.38s	**1.62s**	❌

Target RPS	Actual RPS	Success	Fail	Fail %	Avg Latency	P95	P99	Status
25,000	16,077	273,861	1,222,710	**81.7%**	18.1s	52.2s	**63.8s**	💀
27,500	9,980	240,142	1,409,832	**85.4%**	45.1s	119s	**134s**	💀
30,000	4,755	182,209	1,617,780	**89.9%**	173s	340s	**356s**	💀

📈 Failure Rate Progression



📸 Grafana Dashboard Screenshots

Overview - Throughput & Latency

Figure 4.4: Throughput stable đến 20k RPS, sau đó cliff drop.

JVM & Redis Performance

Figure 4.5: Redis latency tăng exponential sau 20k RPS.

Envoy & Network Stats

Figure 4.6: Error rate visualization trên Envoy dashboard.

🔍 Nhận xét

Observation	Detail
🏆 **Stable Limit**	20,000 RPS với 0% failure
🧱 **Hard Wall Effect**	Vượt 22.5k → collapse không hồi phục
📊 **Latency**	Cao hơn Single Key (30ms vs 6ms tại 10k)

4.3 Kịch bản Hundred Keys - Hot Key (90/10)

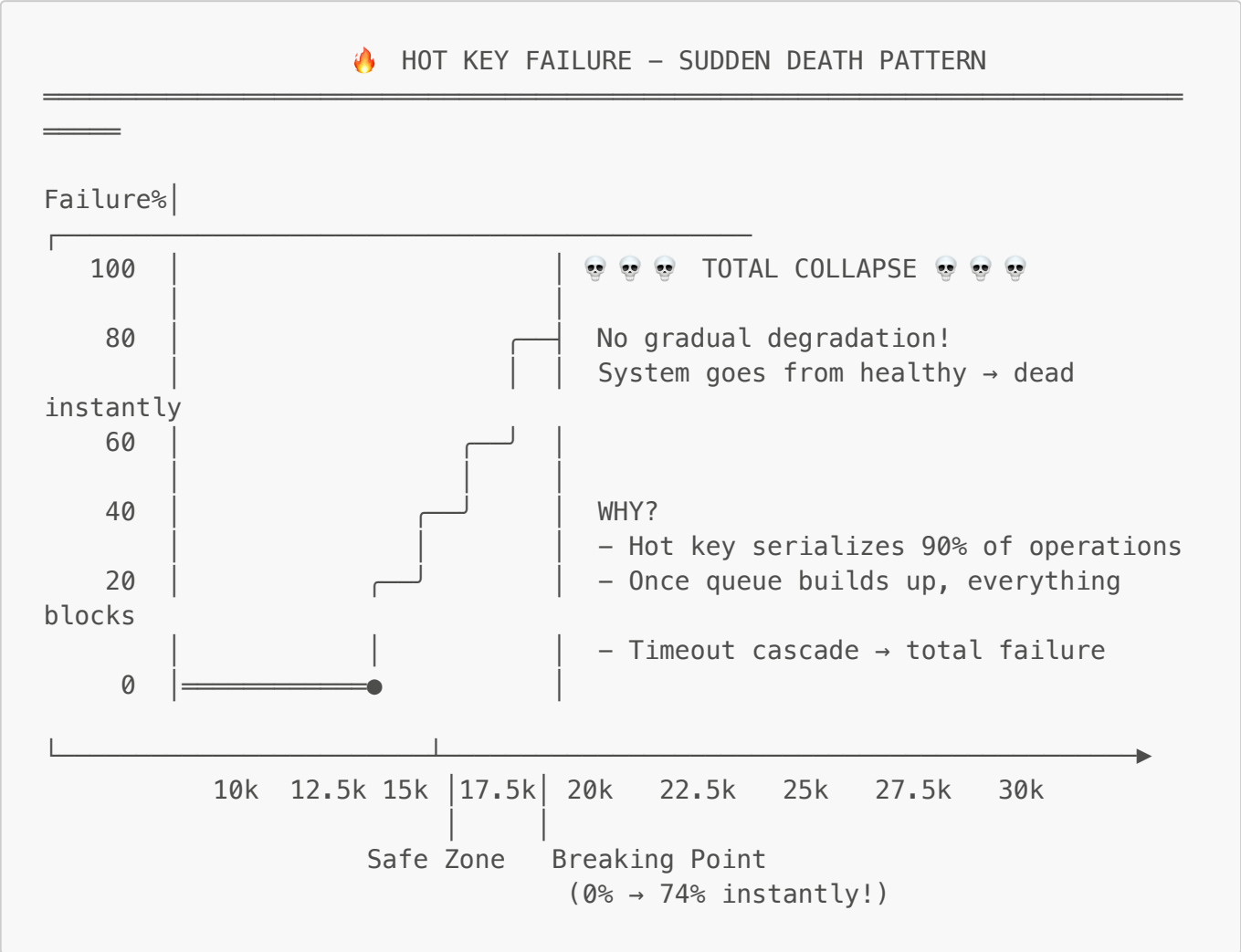
> 📅 **Test Date**: February 1, 2026
> ⏱️ **Duration**: 60 giây/step

▶ 📊 **Bảng kết quả chi tiết (Click để mở)**

Target RPS	Actual RPS	Success	Fail	Fail %	Avg Latency	P95	P99	Status
10,000	9,997	600,000	0	0%	3.08ms	19.7ms	**36.4ms**	✅
12,500	12,493	749,989	0	0%	12.7ms	45.4ms	**66.4ms**	✅
15,000	15,000	900,000	0	0%	18.1ms	59.0ms	**83.6ms**	✅
17,500	**17,494**	**1,049,995**	**0**	**0%**	**36.3ms**	**109ms**	**160ms**	⚡
20,000	13,873	306,231	893,757	**74.5%**	12.4s	38.0s	**49.1s**	💀
22,500	11,835	117,865	1,231,898	**91.3%**	30.9s	76.3s	**83.4s**	💀

Target RPS	Actual RPS	Success	Fail	Fail %	Avg Latency	P95	P99	Status
25,000	6,031	207,837	1,248,614	**85.7%**	110s	215s	**228s**	💀

📉 Catastrophic Failure Pattern



📸 Grafana Dashboard Screenshots

Overview - The Cliff at 20k

Figure 4.7: Sharp drop in throughput và massive spike in latency tại 20k RPS step.

Resource Utilization

Figure 4.8: JVM Memory và CPU plateau khi application trở thành I/O bound.

Redis Performance

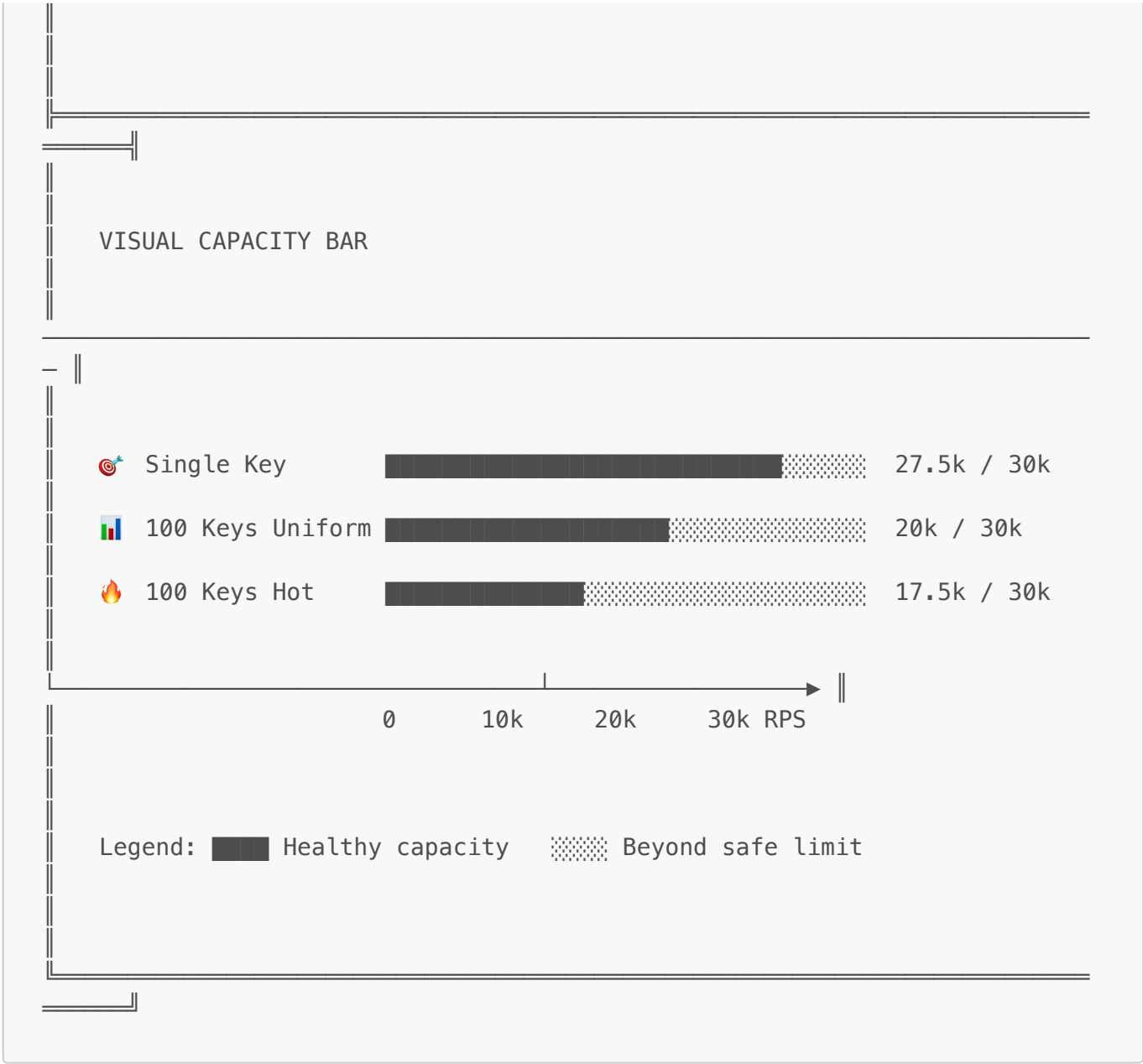
Figure 4.9: Redis latency spike correlate với application failure.

🔍 Nhận xét

Observation	Detail
🔴 **Safe Limit**	Chỉ 17,500 RPS
💀 **Catastrophic**	Tại 20k → 74.5% failure ngay lập tức
⚠ **Worst-Case**	Hot key là pattern nguy hiểm nhất

4.4 📊 Tổng hợp kết quả Redis

```
╔═══════════════════════════════════════════════════════════════════════════╗
║                                                                           ║
║                  🔴 REDIS SINGLE NODE — CAPACITY SUMMARY                  ║
║                                                                           ║
╚═══════════════════════════════════════════════════════════════════════════╝

  Scenario            | Safe Limit | Max Limit   | Breaking Point
  ────────────────────|────────────|─────────────|─────────────────
  🎯 Single Key       |  25,000    |  27,500     |  30,000 RPS
  📊 100 Keys Uniform |  18,000    |  20,000     |  22,500 RPS
  🔥 100 Keys Hot     |  15,000    |  17,500     |  20,000 RPS
```



5. 📈 Kết quả chi tiết: Dragonfly Single Node

5.1 Kịch bản Hundred Keys - Uniform

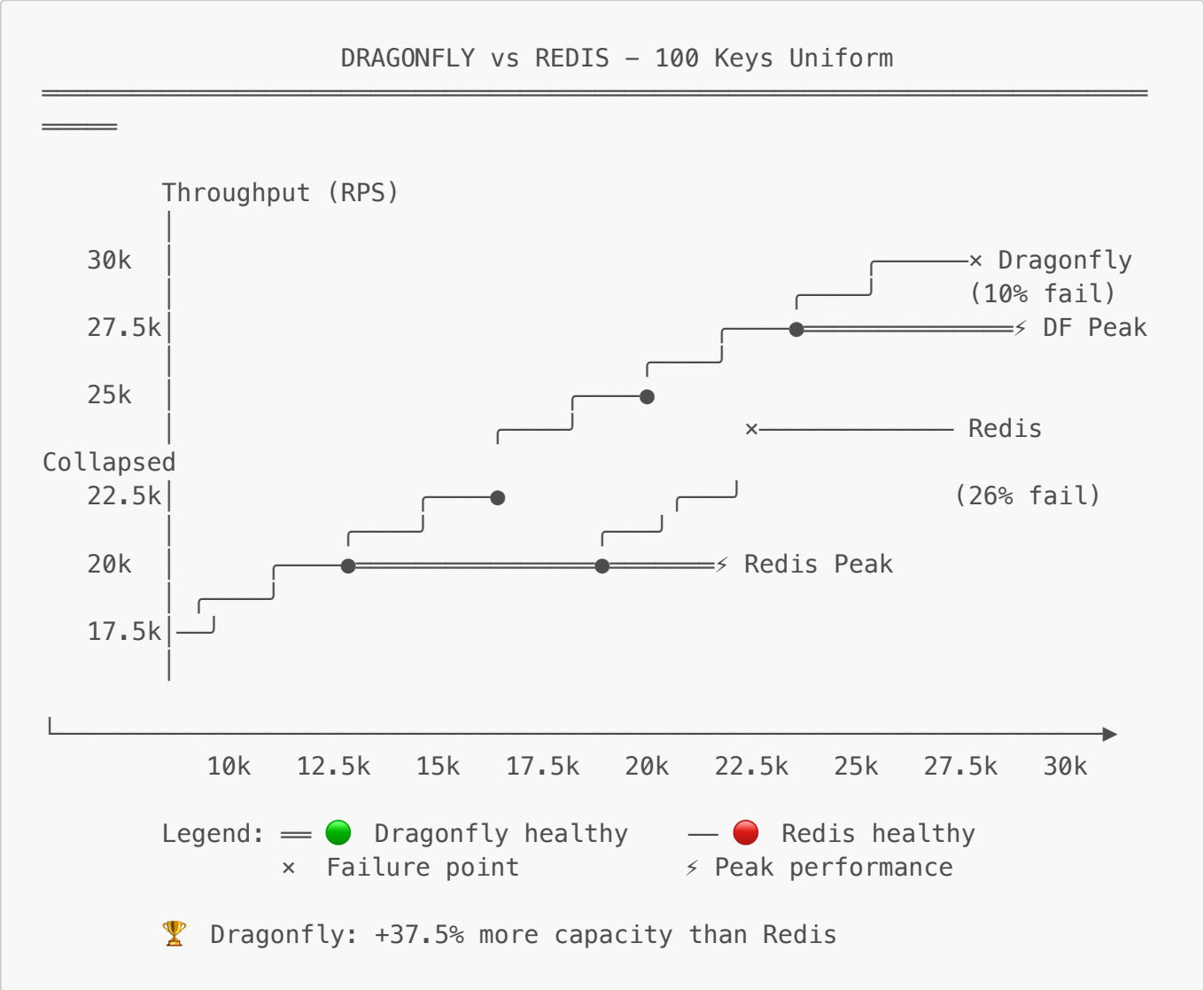
> 📅 **Test Date**: February 4, 2026
> ⏱️ **Duration**: 1 giây/step (short burst test)

▶ 📊 **Bảng kết quả chi tiết (Click để mở)**

Target RPS	Actual RPS	Success	Fail	Fail %	Avg Latency	P95	P99	Status
10,000	9,988	9,998	0	0%	3.28ms	17.9ms	**30.4ms**	✅
12,500	12,482	12,494	0	0%	2.22ms	8.01ms	**22.7ms**	✅
15,000	14,980	14,995	0	0%	1.55ms	2.70ms	**12.4ms**	✅
17,500	17,476	17,493	0	0%	2.10ms	5.92ms	**20.7ms**	✅

Target RPS	Actual RPS	Success	Fail	Fail %	Avg Latency	P95	P99	Status
20,000	19,947	19,987	0	0%	10.2ms	43.5ms	**59.4ms**	✅
22,500	22,471	22,493	0	0%	2.65ms	7.62ms	**22.4ms**	✅
25,000	24,722	24,994	0	0%	82.8ms	146ms	**154ms**	⚠
27,500	**27,456**	**27,483**	**0**	**0%**	**36.2ms**	**70.8ms**	**83.2ms**	✅
30,000	26,376	26,889	3,100	**10.3%**	164ms	283ms	**304ms**	❌

📈 Dragonfly vs Redis - Uniform Load Comparison



🔍 Nhận xét

Observation	Detail
🏆 **Stable Limit**	27,500 RPS với 0% failure (Redis: 20,000 RPS)
📊 **Latency Excellence**	P99 < 100ms tại 27.5k RPS

Observation	Detail
📈 **Capacity Gain**	+37.5% so với Redis

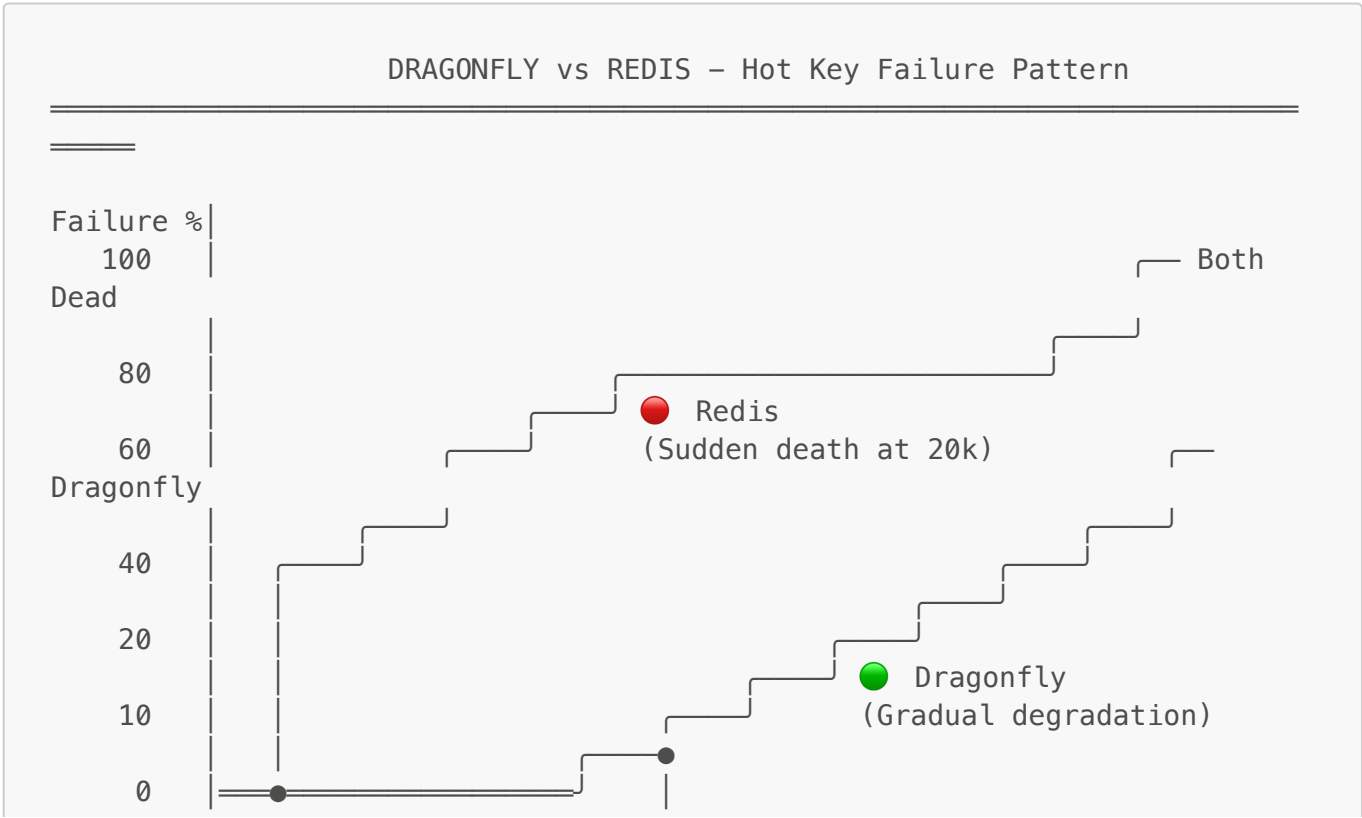
5.2 Kịch bản Hundred Keys - Hot Key (90/10)

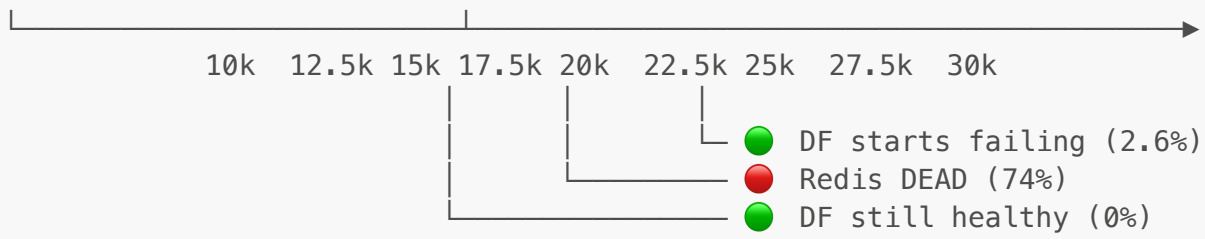
> 📅 **Test Date**: February 5, 2026
> ⏱️ **Duration**: 60 giây/step

▶ 📊 **Bảng kết quả chi tiết (Click để mở)**

Target RPS	Actual RPS	Success	Fail	Fail %	Avg Latency	P95	P99	Status
10,000	10,000	600,000	0	0%	0.96ms	1.16ms	**4.98ms**	✅
12,500	12,500	749,991	0	0%	1.17ms	1.82ms	**6.12ms**	✅
15,000	15,000	899,987	0	0%	1.54ms	2.60ms	**10.18ms**	✅
17,500	17,499	1,049,986	0	0%	2.34ms	4.60ms	**24.27ms**	✅
20,000	19,999	1,199,982	0	0%	4.29ms	14.37ms	**42.05ms**	✅
22,500	**22,499**	**1,349,989**	**0**	**0%**	**7.51ms**	**28.98ms**	**54.62ms**	⚡
25,000	24,998	1,460,145	39,831	**2.65%**	94.98ms	276ms	**358ms**	⚠
27,500	27,478	1,505,433	144,533	**8.76%**	106.9ms	324ms	**420ms**	❌
30,000	28,880	704,778	1,095,190	**60.8%**	1.26s	3.30s	**4.66s**	💀

📈 **Graceful Degradation Pattern**





KEY INSIGHT: Dragonfly provides "Golden Time" for reaction
- Redis: 0% → 74% failure (instant death)
- Dragonfly: 0% → 2.6% → 8.7% → 60% (gradual, actionable)

📸 Grafana Dashboard Screenshot

Overview - Hot Key Test

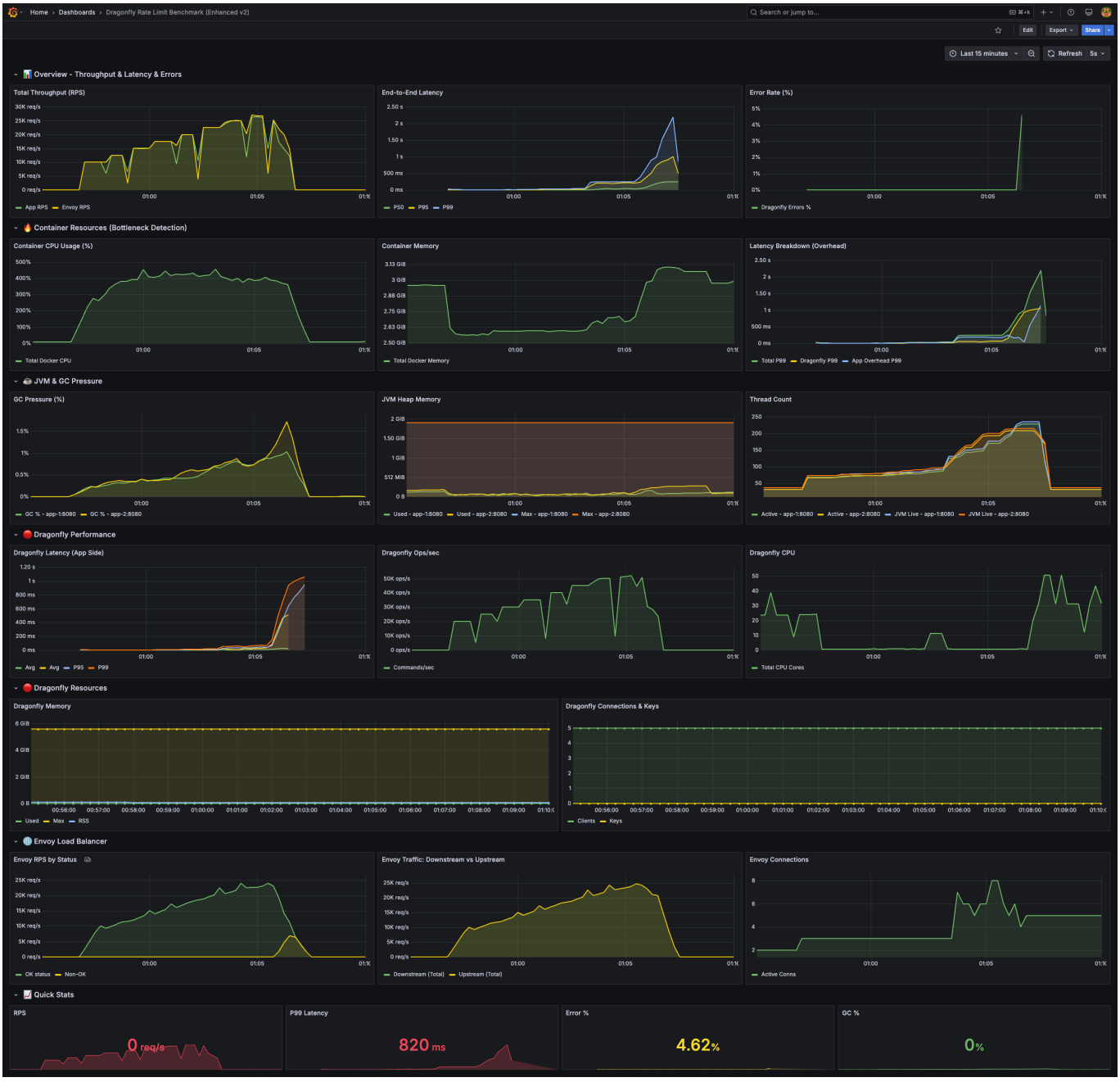
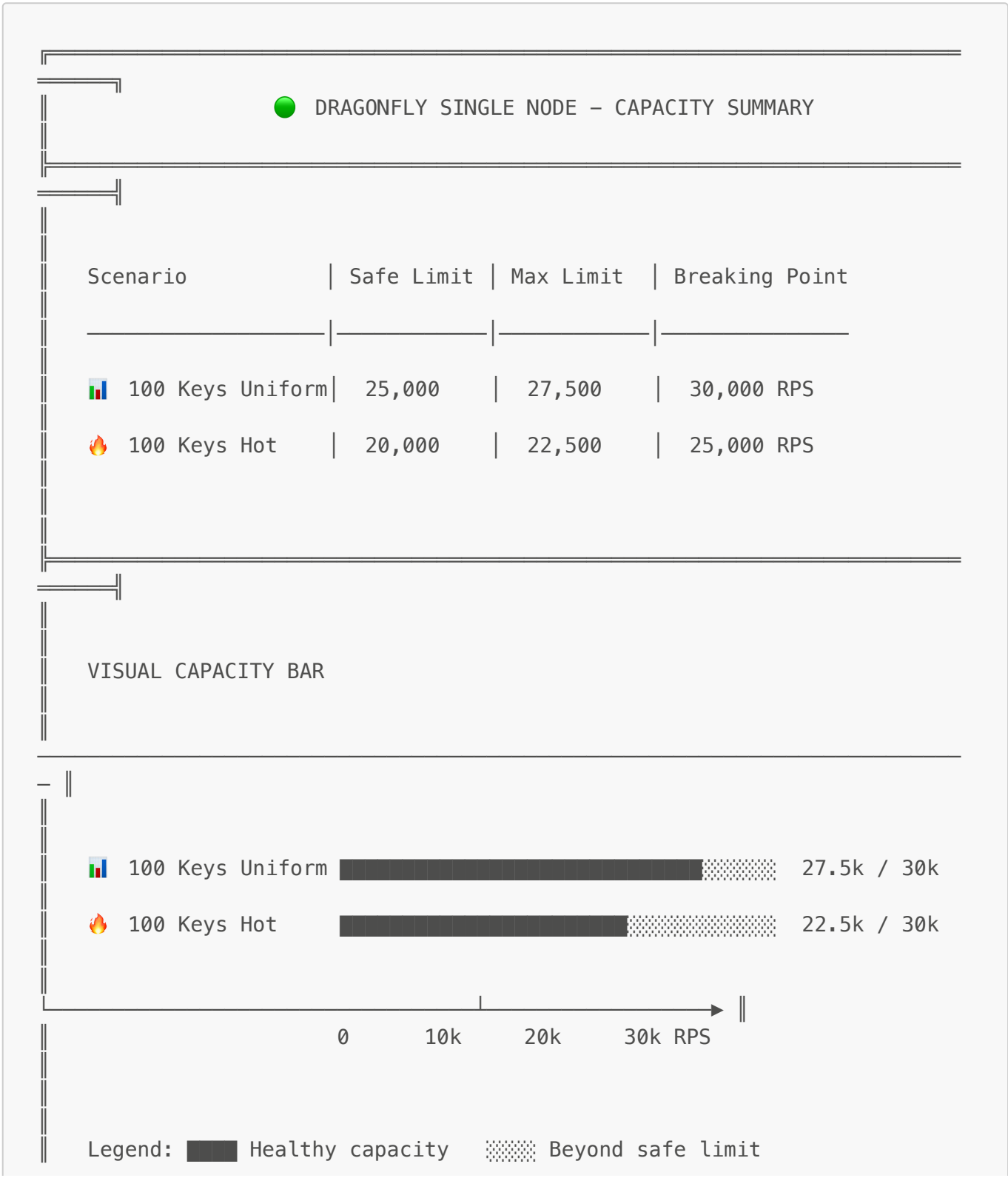


Figure 5.1: Dragonfly maintains stability up to 22.5k RPS under hot key load.

🔍 **Nhận xét**

Observation	Detail
🏆 **Stable Limit**	22,500 RPS với 0% failure (Redis: 17,500 RPS)
📈 **Graceful Degradation**	Failure tăng từ từ (2.6% → 8.7% → 60.8%)
⚡ **Latency Superiority**	Tại 17.5k: 24ms vs Redis 160ms (**6.6x faster**)

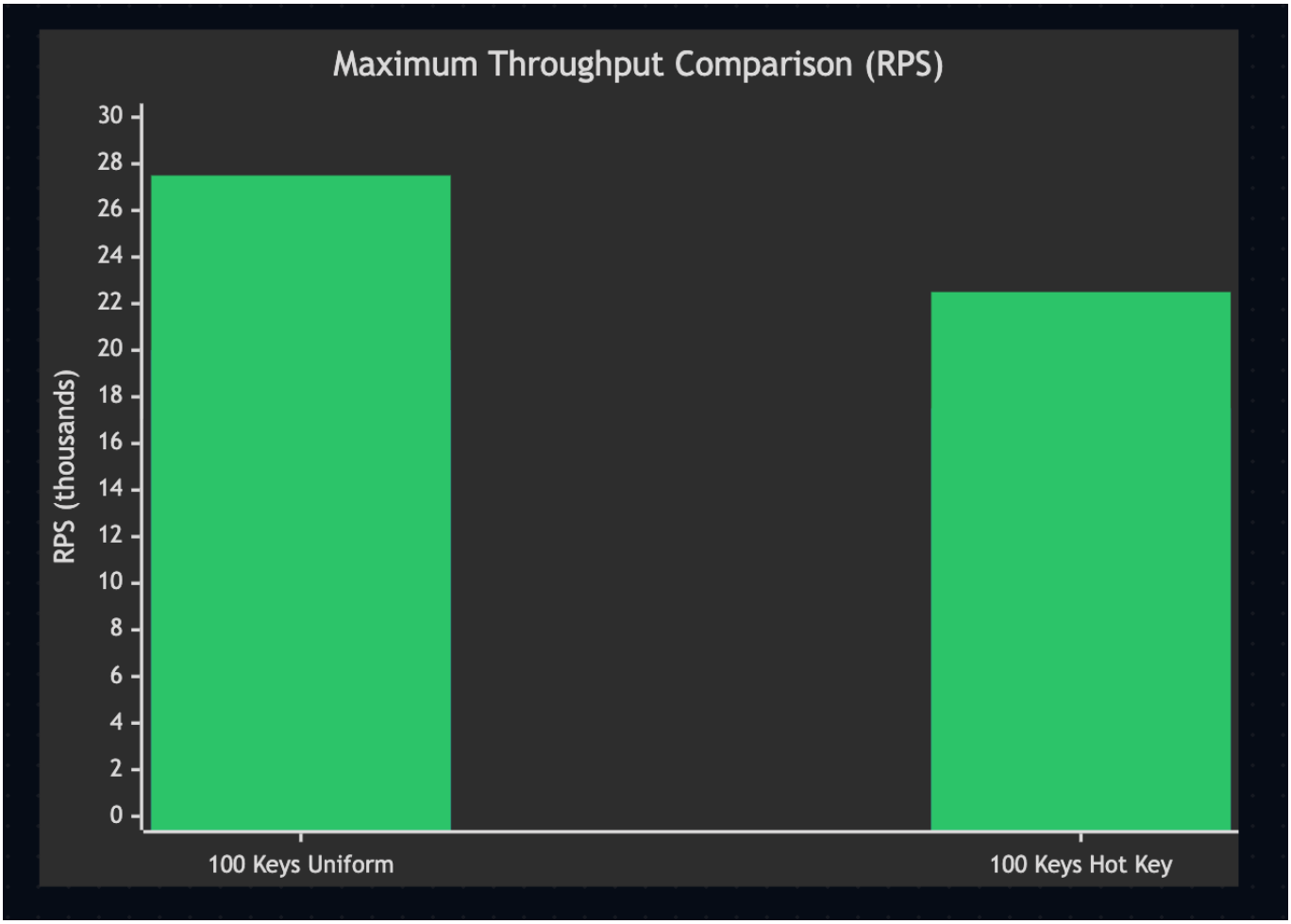
5.3 📊 Tổng hợp kết quả Dragonfly



---

6. ⚔️ So sánh trực tiếp Redis vs Dragonfly

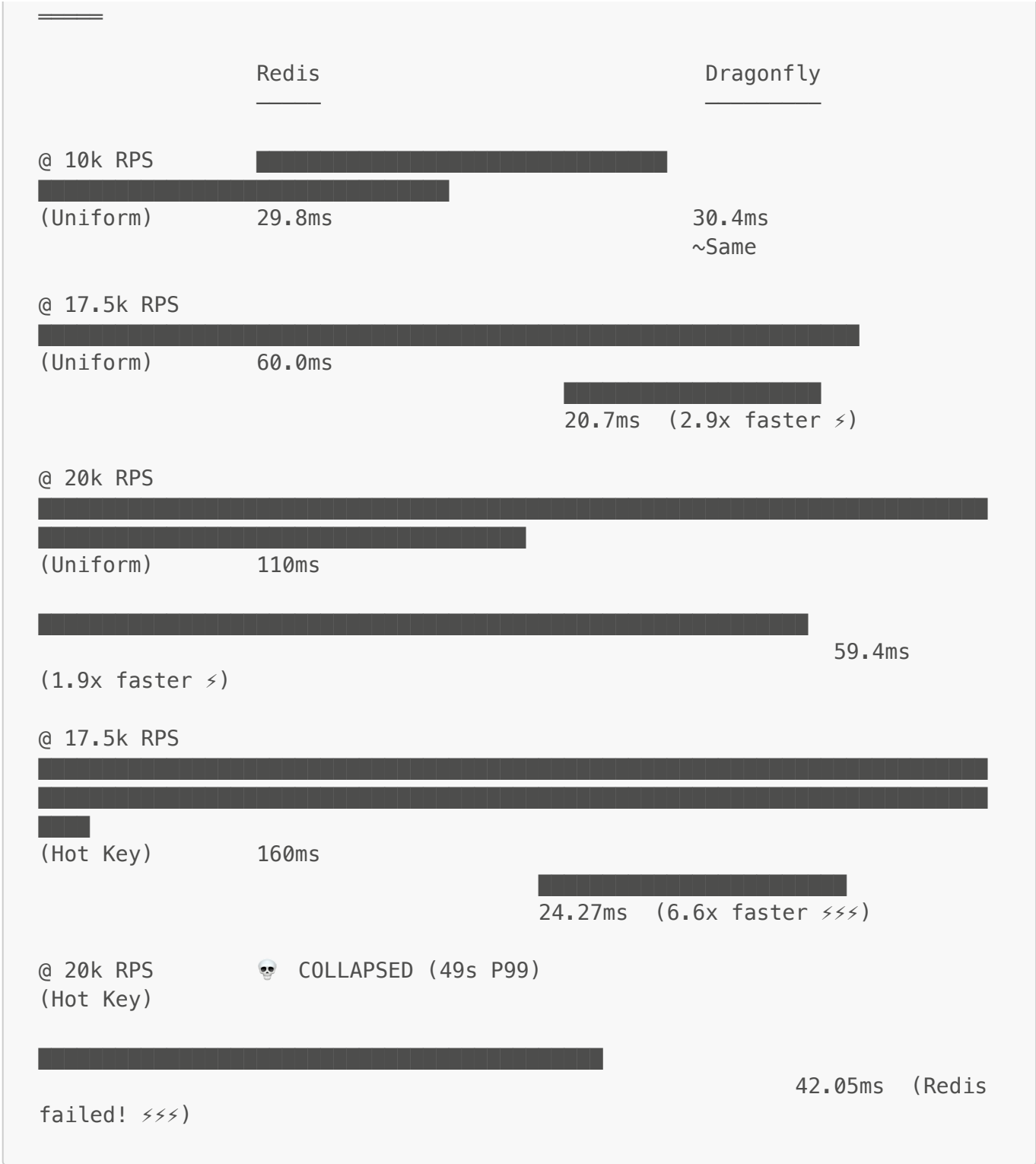
6.1 📊 Throughput Comparison



Kịch bản	Redis Safe	Dragonfly Safe	📈 Improvement	Redis Max	Dragonfly Max	📈 Improvement
100 Keys (Uniform)	18,000	25,000	**+38.9%**	20,000	27,500	**+37.5%**
100 Keys (Hot Key)	15,000	20,000	**+33.3%**	17,500	22,500	**+28.6%**

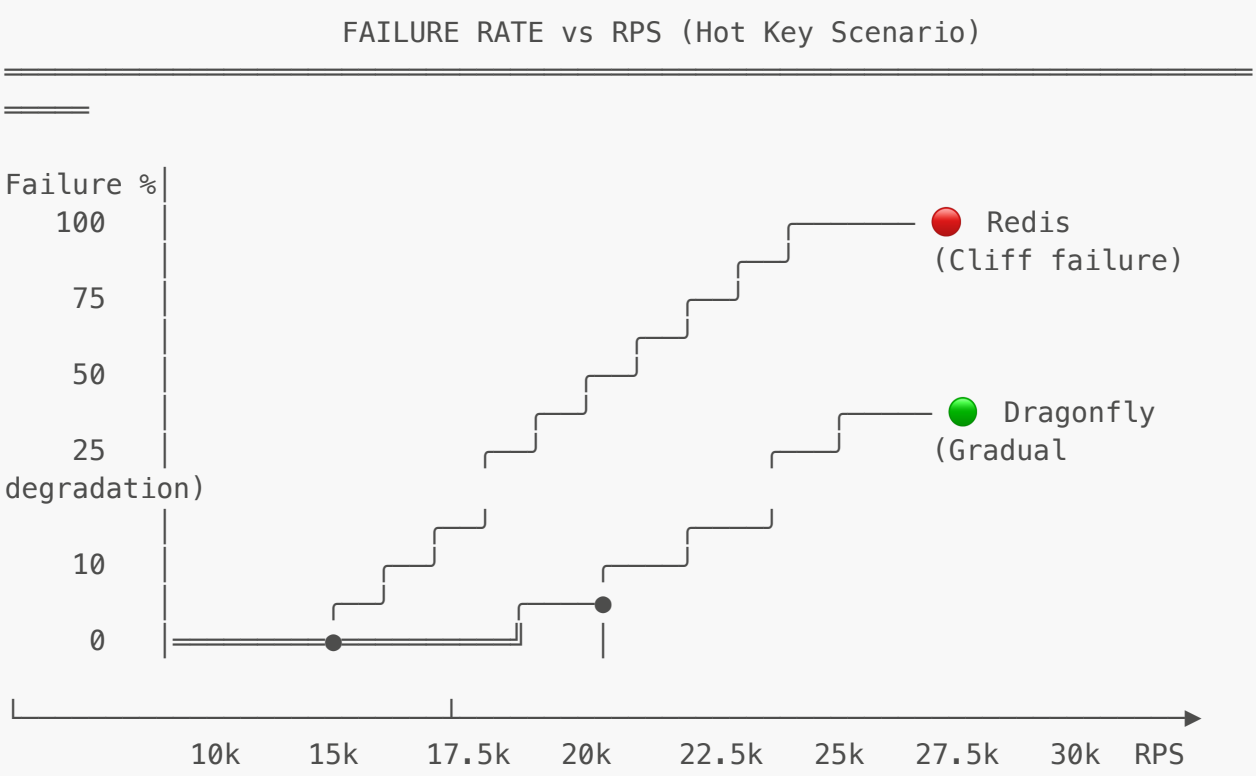
6.2 ⚡ Latency Comparison (P99 at Comparable Load)

```
                    P99 LATENCY COMPARISON (ms) - Lower is Better
═══════════════════════════════════════════════════════════════════
```



Load Level	Redis P99	Dragonfly P99	📈 Improvement
10,000 RPS (Uniform)	29.8ms	30.4ms	~Same
17,500 RPS (Uniform)	60.0ms	20.7ms	🔺 **2.9x faster**
20,000 RPS (Uniform)	110ms	59.4ms	🔺 **1.9x faster**
10,000 RPS (Hot Key)	36.4ms	4.98ms	🔺 **7.3x faster**
17,500 RPS (Hot Key)	160ms	24.27ms	🔺 **6.6x faster**
20,000 RPS (Hot Key)	💀 collapsed	42.05ms	🔺 **Redis failed!**

6.3 Failure Pattern Comparison



📊 KEY INSIGHT: Failure Pattern Difference

🔴 REDIS: 0% ⟶ 74% (instant death, no warning)

↑

No time to react!

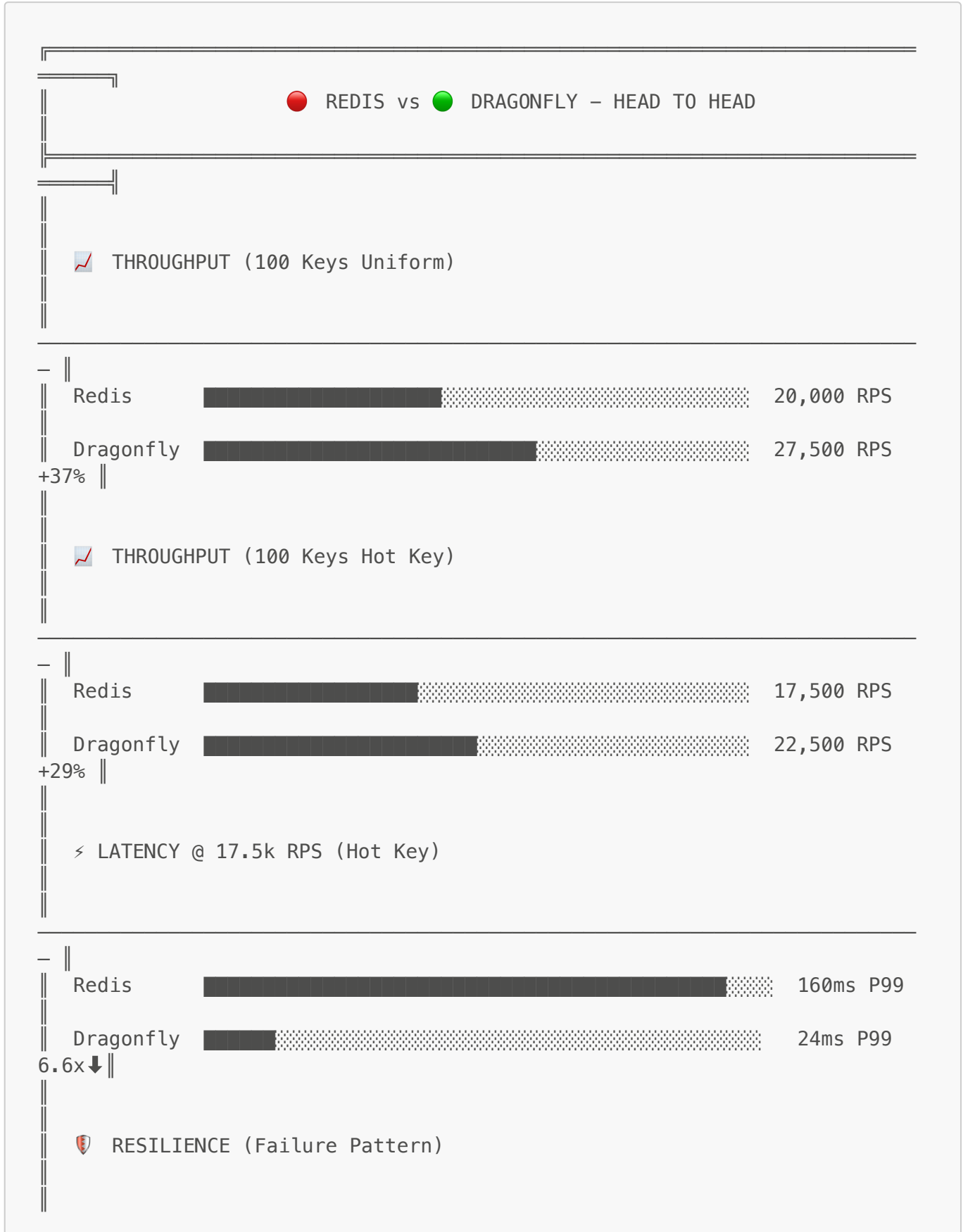
🟢 DRAGONFLY: 0% ⟶ 2.6% ⟶ 8.7% ⟶ 60% (gradual, actionable)

↑ ↑ ↑

Alert! Scale! Circuit Break!

👉 Dragonfly cho "thời gian vàng" để hệ thống phản ứng

6.4 🏆 Visual Summary - Head to Head



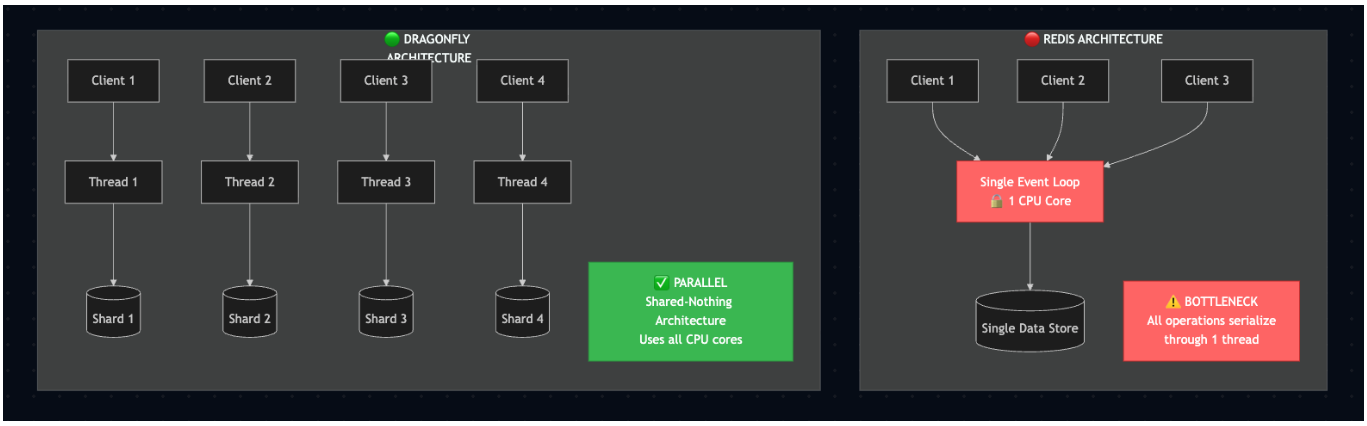
```
═══════════════════════════════════════════════════════════════════
─ ║
║  Redis      ⚠ Catastrophic collapse (0% → 74% instantly)     ❌
DANGEROUS   ║
║  Dragonfly ✅  Graceful degradation (gradual increase)        ✅  SAFE
║
║
║
╠═══════════════════════════════════════════════════════════════════
═════╣
║
║
║                         🏆  WINNER: DRAGONFLY
║
║
║
║     "37% more throughput, 6.6x lower latency, graceful failure"
║
║
║
╚═══════════════════════════════════════════════════════════════════
═════╝
```

---

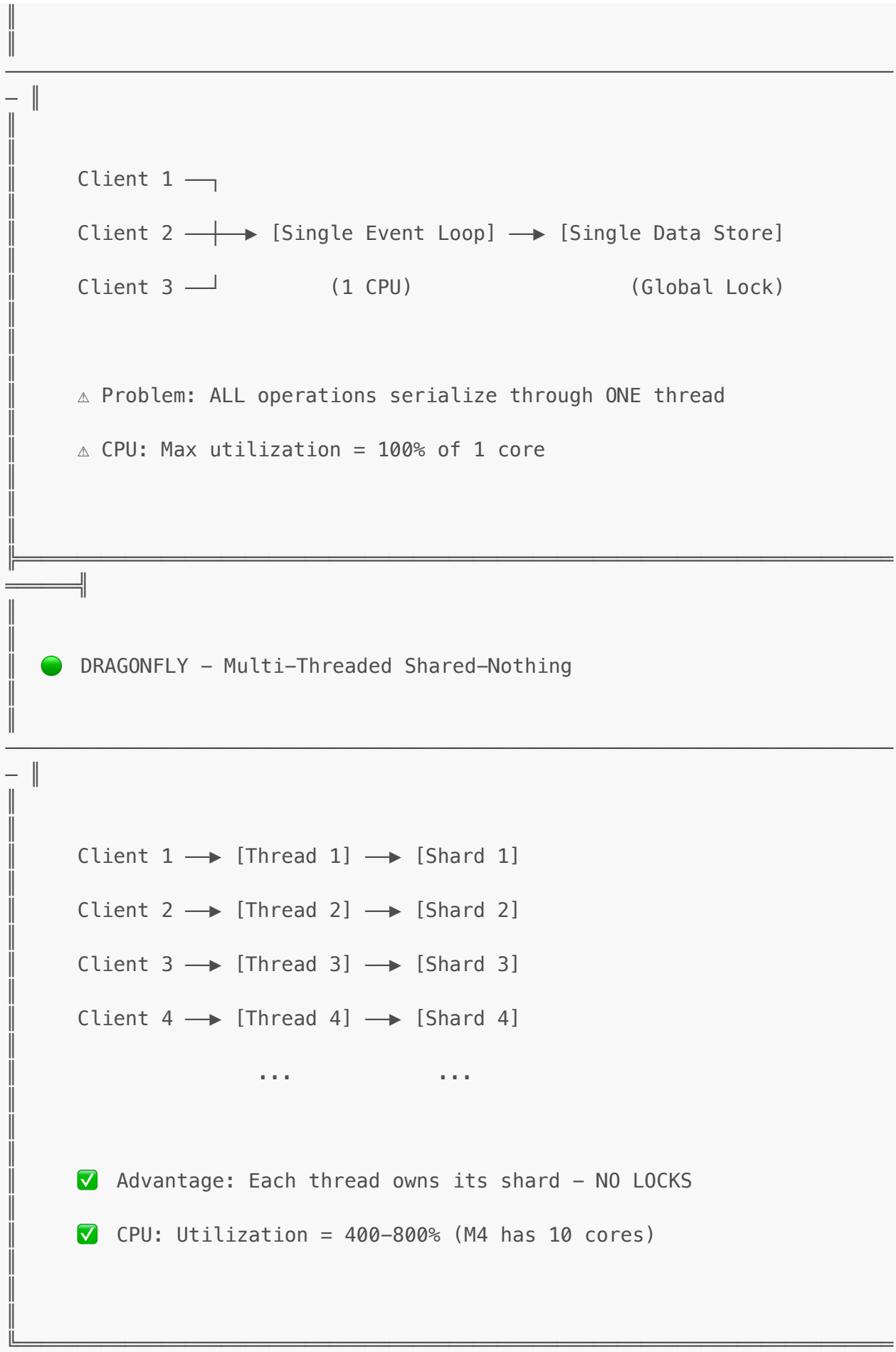
7. 🔍 Phân tích chuyên sâu

7.1 Tại sao Dragonfly nhanh hơn?

🏗️ Kiến trúc Single-threaded vs Multi-threaded




```
╔═══════════════════════════════════════════════════════════════════
═════╗
║                    🔬  ARCHITECTURE DEEP DIVE
║
╠═══════════════════════════════════════════════════════════════════
═════╣
║
║  🔴  REDIS - Single-Threaded Event Loop
```

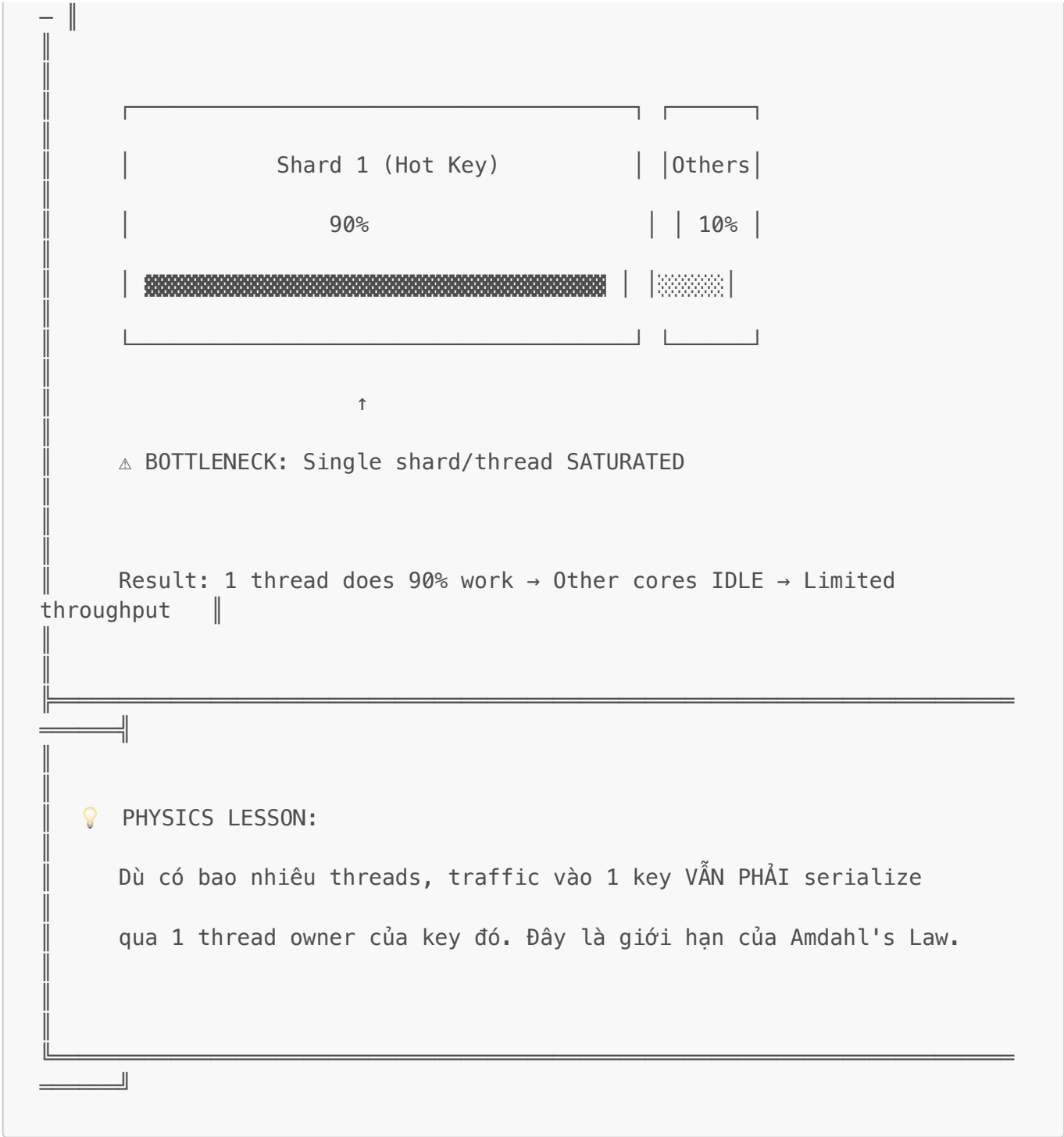

Client 1
Client 2
Client 3
[Single Event Loop]
[Single Data Store]
(1 CPU)
(Global Lock)
⚠ Problem: ALL operations serialize through ONE thread
⚠ CPU: Max utilization = 100% of 1 core
DRAGONFLY - Multi-Threaded Shared-Nothing
Client 1
[Thread 1]
[Shard 1]
Client 2
[Thread 2]
[Shard 2]
Client 3
[Thread 3]
[Shard 3]
Client 4
[Thread 4]
[Shard 4]
...
...
✅ Advantage: Each thread owns its shard - NO LOCKS
✅ CPU: Utilization = 400-800% (M4 has 10 cores)

📊 **Lý do cụ thể:**

Factor	🔴 Redis	🟢 Dragonfly	Impact
CPU Utilization	Max 1 core (~100%)	Multi-core (400-800%)	**4-8x potential**
Memory Access	Single-threaded GC	Per-shard memory	Less contention
Lock Contention	Global dict lock	Lock-free per shard	Better parallelism
I/O Handling	epoll/kqueue single	io_uring multi-thread	Higher throughput

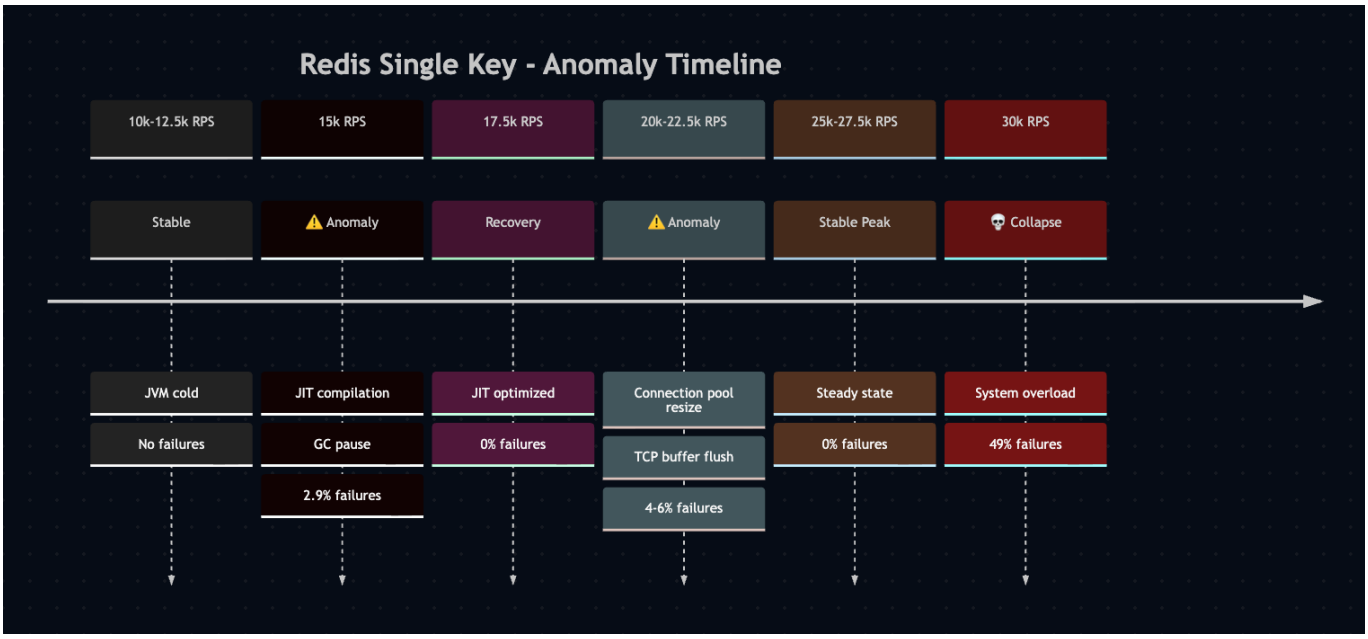
7.2 Tại sao Hot Key vẫn là vấn đề?

Dù Dragonfly nhanh hơn, **Hot Key vẫn là bottleneck** vì quy luật vật lý:



7.3 Giải thích các Anomaly

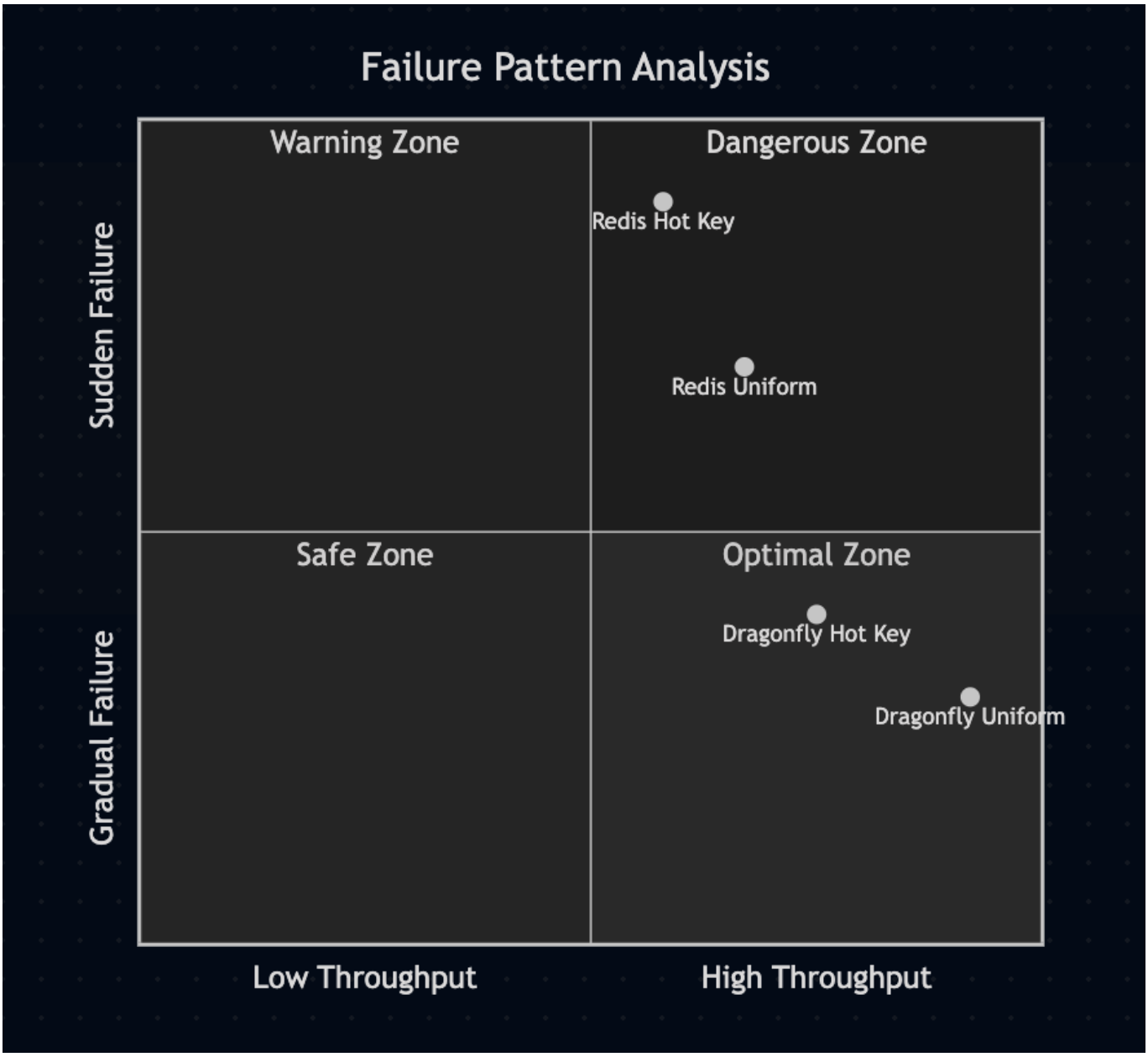
🔍 **Redis Single Key Test - Tại sao 15k, 20k, 22.5k RPS có failure tạm thời?**



Nguyên nhân có thể:

Anomaly	Nguyên nhân	Tại sao hồi phục?
15k spike	JVM JIT Compilation	HotSpot optimize xong code path
20k spike	Connection Pool resize	Pool đã scale up đủ
22.5k spike	GC pause + TCP buffer	GC frequency ổn định
25k-27.5k stable	System đạt steady state	Tất cả đã warmup xong

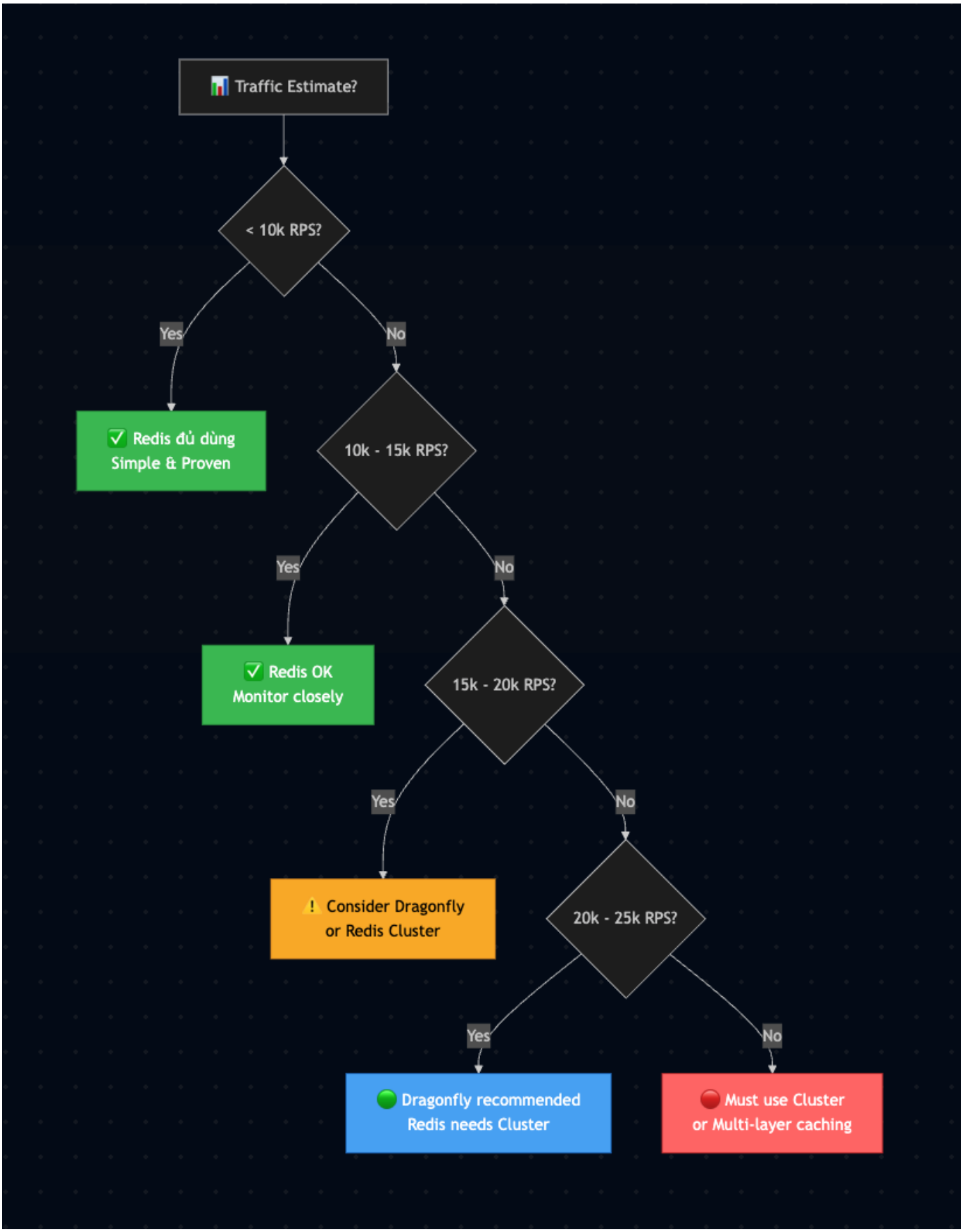
7.4 📚 Bài học từ Failure Patterns



Pattern	🔴 Redis Behavior	🟢 Dragonfly Behavior	Implication
Threshold Breach	Sudden collapse	Gradual degradation	DF cho thêm thời gian react
Recovery	Không tự hồi phục	Có thể recover	DF resilient hơn
Cascading	Hot key → toàn bộ client bị ảnh hưởng	Isolate tốt hơn	DF giảm blast radius

8. 🚀 Khuyến nghị Production

8.1 📊 Capacity Planning Matrix

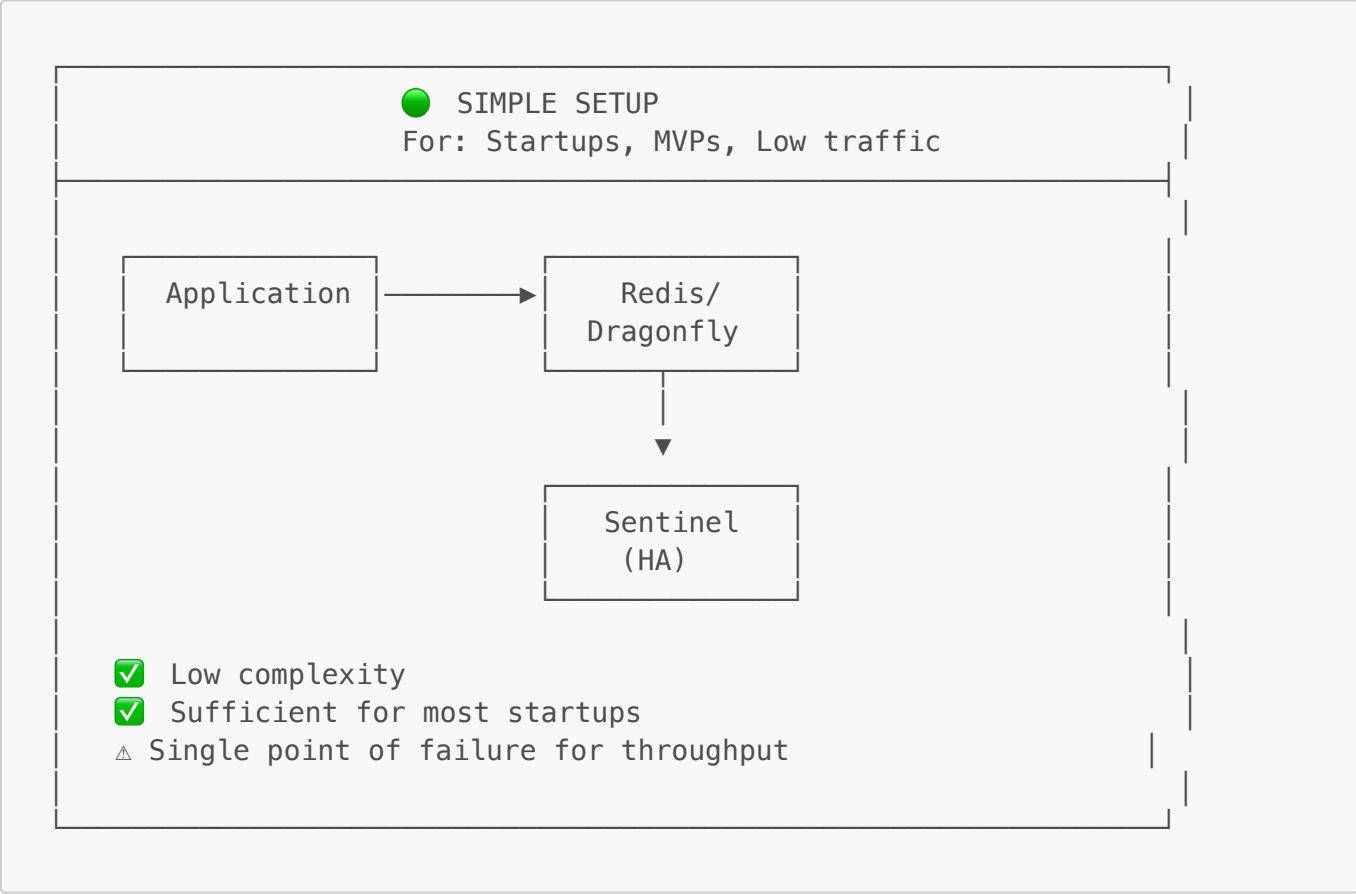


Use Case	🔴 Redis Single Node	🟢 Dragonfly Single Node	👉 Recommendation
< 10,000 RPS	✅ Comfortable	✅ Overkill	Redis đủ dùng

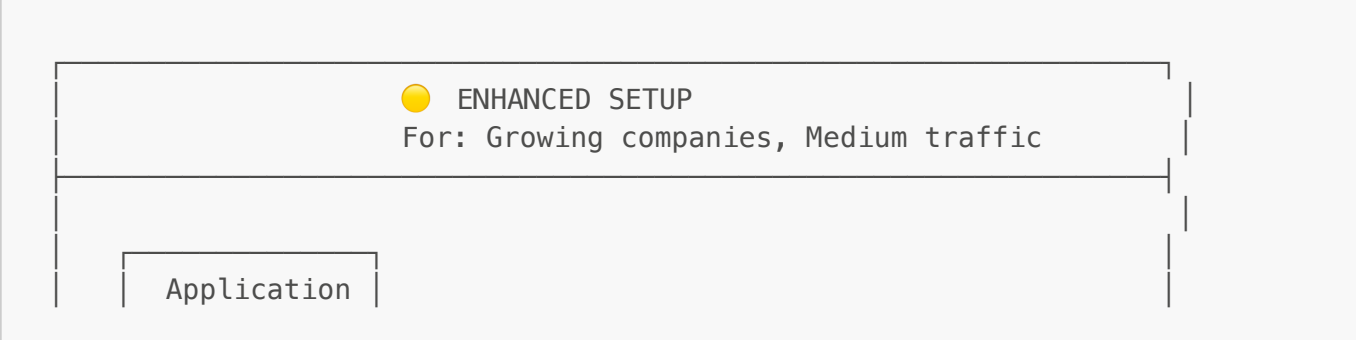
Use Case	🔴 Redis Single Node	🟢 Dragonfly Single Node	👉 Recommendation
10,000 - 15,000 RPS	✅ Safe	✅ Easy	Redis + Monitor
15,000 - 20,000 RPS	⚠ Near limit	✅ Safe	**Migrate to Dragonfly**
20,000 - 25,000 RPS	❌ Need Cluster	✅ Safe	Dragonfly recommended
> 25,000 RPS	❌ Must Cluster	⚠ Near limit	Cluster required

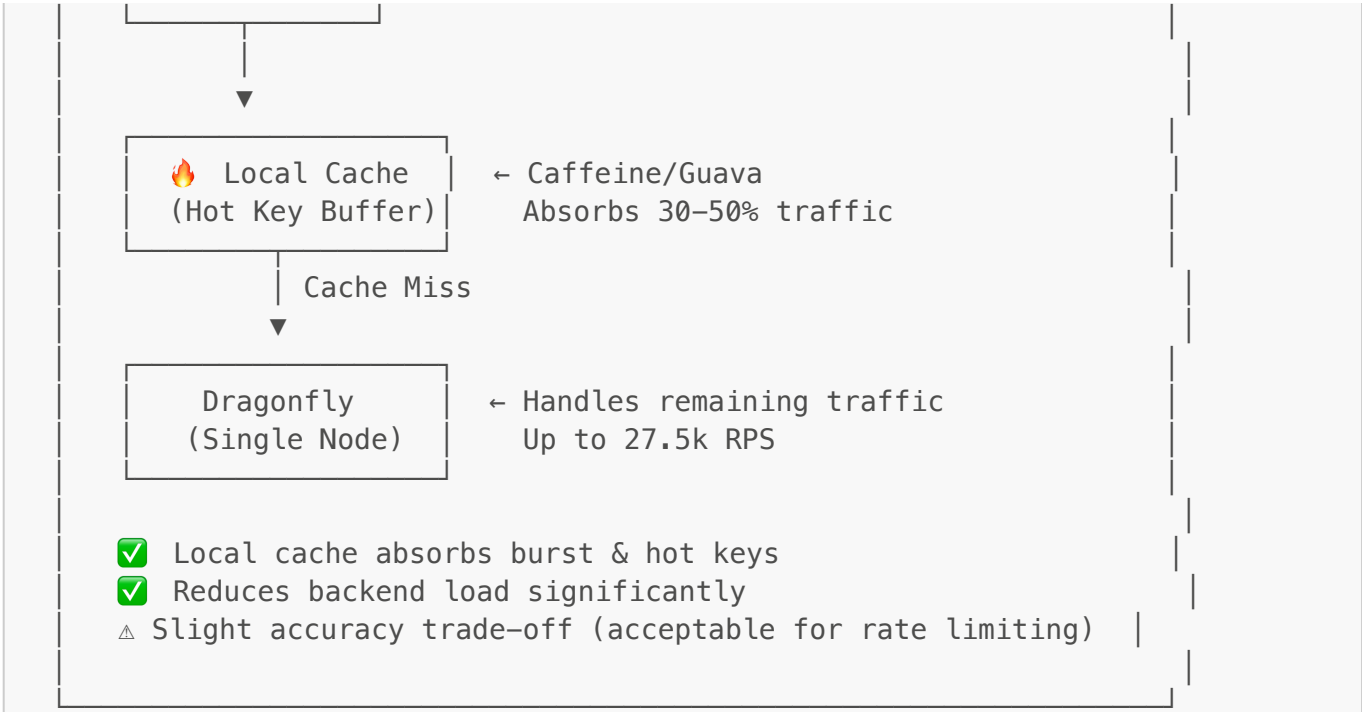
8.2 🏗️ Architecture Recommendations

Tier 1: Simple Setup (< 15k RPS)

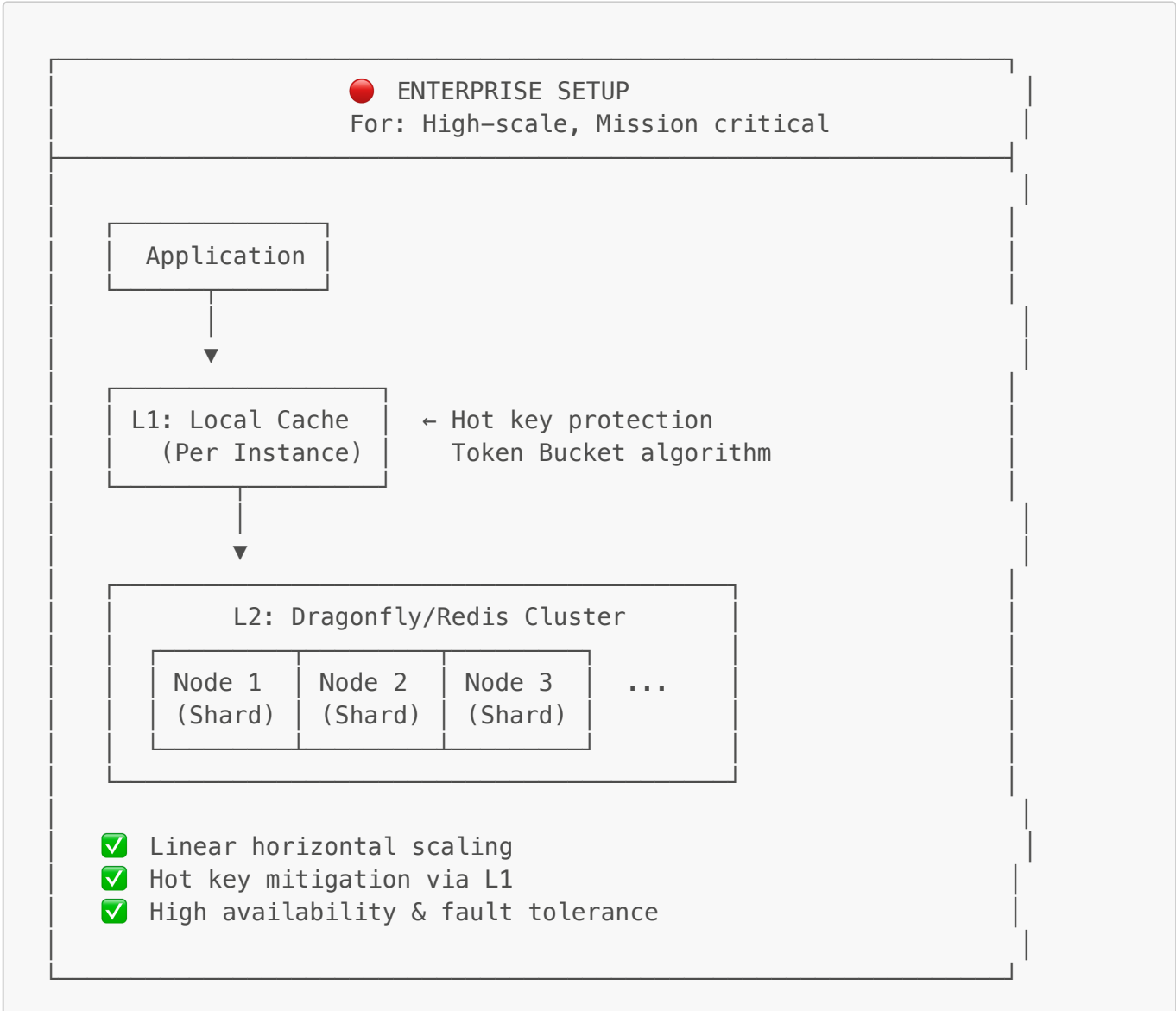


Tier 2: Enhanced Setup (15k - 30k RPS)

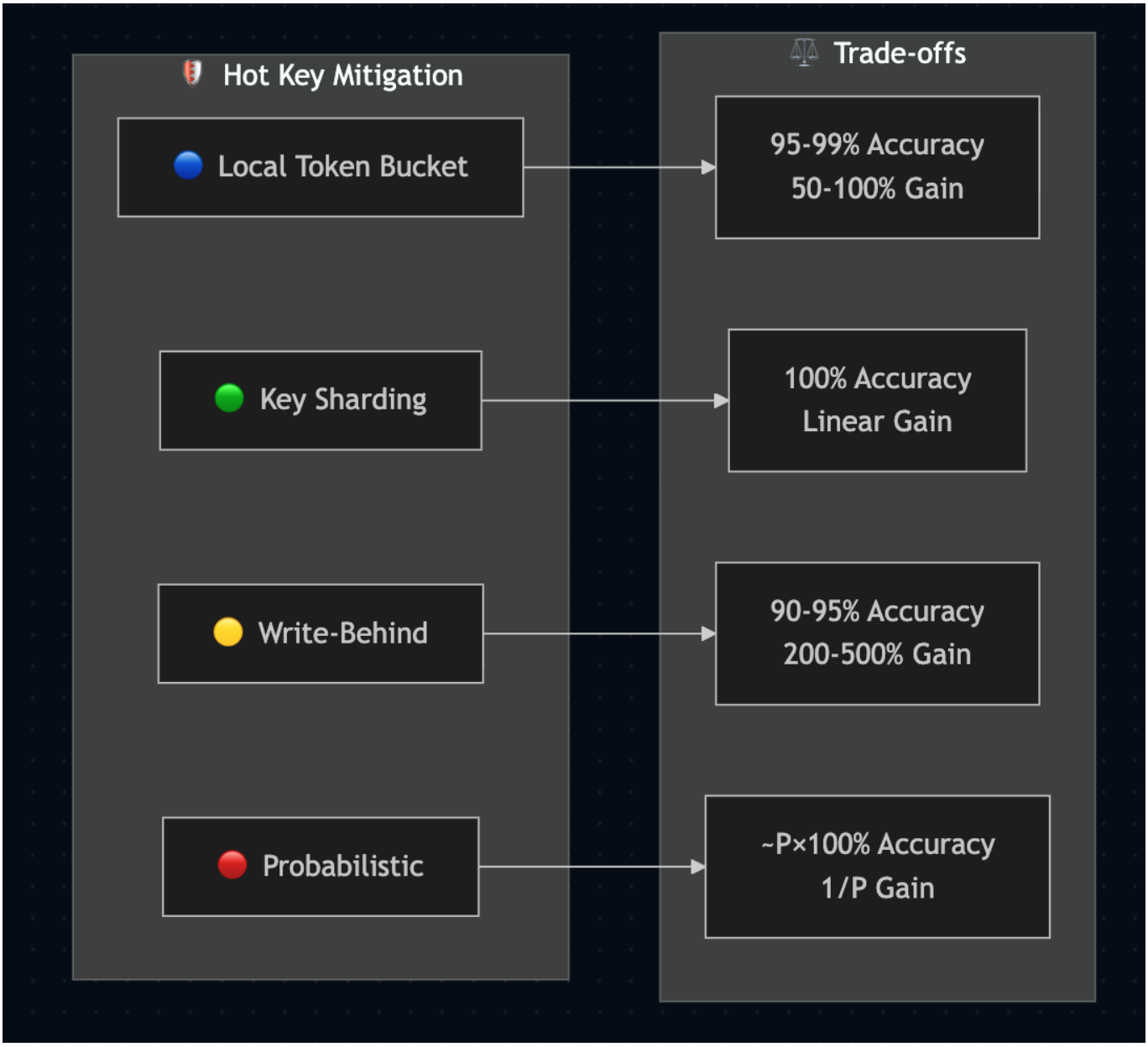




Tier 3: Enterprise Setup (> 30k RPS)



8.3 🔥 Hot Key Mitigation Strategies



Strategy	Description	Accuracy	Throughput Gain	Complexity
🔵 **Local Token Bucket**	Cho phép N req/s locally trước khi check remote	95-99%	50-100%	Low
🟢 **Key Sharding**	Thêm random suffix: `key:0`, `key:1`, ...	100%	Linear	Medium
🟡 **Write-Behind**	Batch local increments, flush periodically	90-95%	200-500%	Medium
🔴 **Probabilistic**	Chỉ check rate limit với probability P	~P×100%	1/P	Low

8.4 📊 Monitoring & Alerting

```
╔═══════════════════════════════════════════════════════════════════════════════════════════════════════
═══════╗
║                                    📊 MONITORING THRESHOLDS
                                                                                                       ║
╠═══════════════════════════════════════════════════════════════════════════════════════════════════════
═══════╣
║                                                                                                       
                                                                                                       ║
║   Metric            | 🟢 Good       | 🟡 Warning     | 🔴 Critical    |
Action    ║
║
──────────────────|─────────────────|─────────────────|─────────────────|────────
──╫
║   RPS               | < 70% cap     | 70–85% cap     | > 85% cap      |
Scale     ║
║   P99 Latency       | < 50ms        | 50–100ms       | > 100ms        |
Investigate╫
║   Error Rate        | < 0.1%        | 0.1–1%         | > 1%           |
Alert     ║
║   CPU (Backend)     | < 60%         | 60–80%         | > 80%          |
Scale     ║
║   Memory            | < 70%         | 70–85%         | > 85%          | Tune
TTL   ╫
║   Connections       | < 80% pool    | 80–95% pool    | > 95% pool     |
Increase  ╫
║
                                                                                                       ║
╚═══════════════════════════════════════════════════════════════════════════════════════════════════════
═══════╝
```

8.5 🔄 Migration Checklist: Redis → Dragonfly

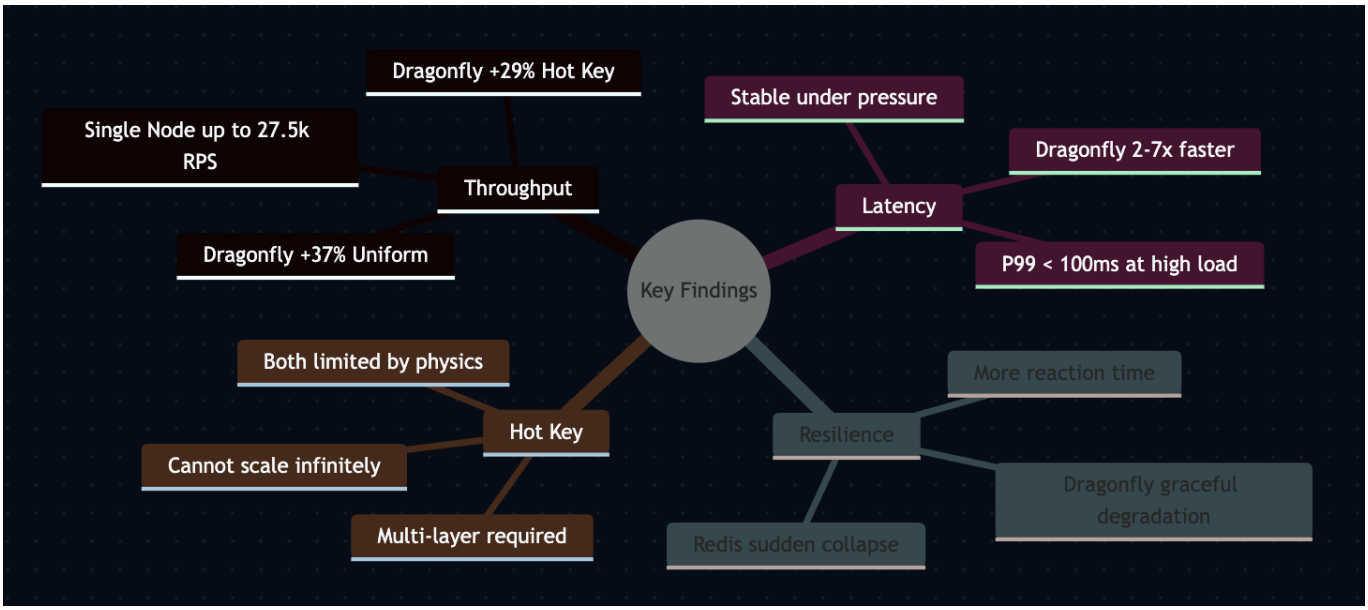
```
┌──────────────────────────────────────────────────────────────────────────────────────────────────────
──────┐
│                                    📋 MIGRATION CHECKLIST
                                                                                                       │
├──────────────────────────────────────────────────────────────────────────────────────────────────────
──────┤
│                                                                                                       
                                                                                                       │
│  PREPARATION
                                                                                                       │
│                                                                                                       
                                                                                                       │
└──────────────────────────────────────────────────────────────────────────────────────────────────────
──────┘
┌──────────────────────────────────────────────────────────────────────────────────────────────────────
──────┐
│                                                                                                       
                                                                                                       │
│  ☐ Compatibility Test: Chạy test suite với Dragonfly backend
                                                                                                       │
│                                                                                                       
                                                                                                       │
│  ☐ Benchmark: So sánh performance trên staging environment
                                                                                                       │
│                                                                                                       
                                                                                                       │
│  ☐ Documentation: Review Dragonfly-specific limitations
```

```
│
│
│  CONFIGURATION
│
├──────────────────────────────────────────────────────────────────────────────
│
│  ☐ Connection Pool: Có thể tăng pool size (DF xử lý nhanh hơn)
│
│  ☐ Timeout Settings: Có thể giảm timeout (latency thấp hơn)
│
│  ☐ Persistence: Review RDB/AOF settings nếu cần
│
│
│  DEPLOYMENT
│
├──────────────────────────────────────────────────────────────────────────────
│
│  ☐ Monitoring: Cập nhật dashboard cho Dragonfly metrics
│
│  ☐ Rollback Plan: Giữ Redis config để rollback nếu cần
│
│  ☐ Canary Deploy: Chuyển 5% traffic trước, monitor 24h
│
│  ☐ Gradual Rollout: 5% → 25% → 50% → 100%
│
│
│  VALIDATION
│
├──────────────────────────────────────────────────────────────────────────────
│
│  ☐ Performance: Verify throughput & latency meet expectations
│
│  ☐ Correctness: Rate limit accuracy unchanged
│
│  ☐ Stability: No memory leaks, connection issues after 48h
│
│
└──────────────────────────────────────────────────────────────────────────────
┘
```

---

9. ✅ Kết luận

9.1 🔑 Key Findings



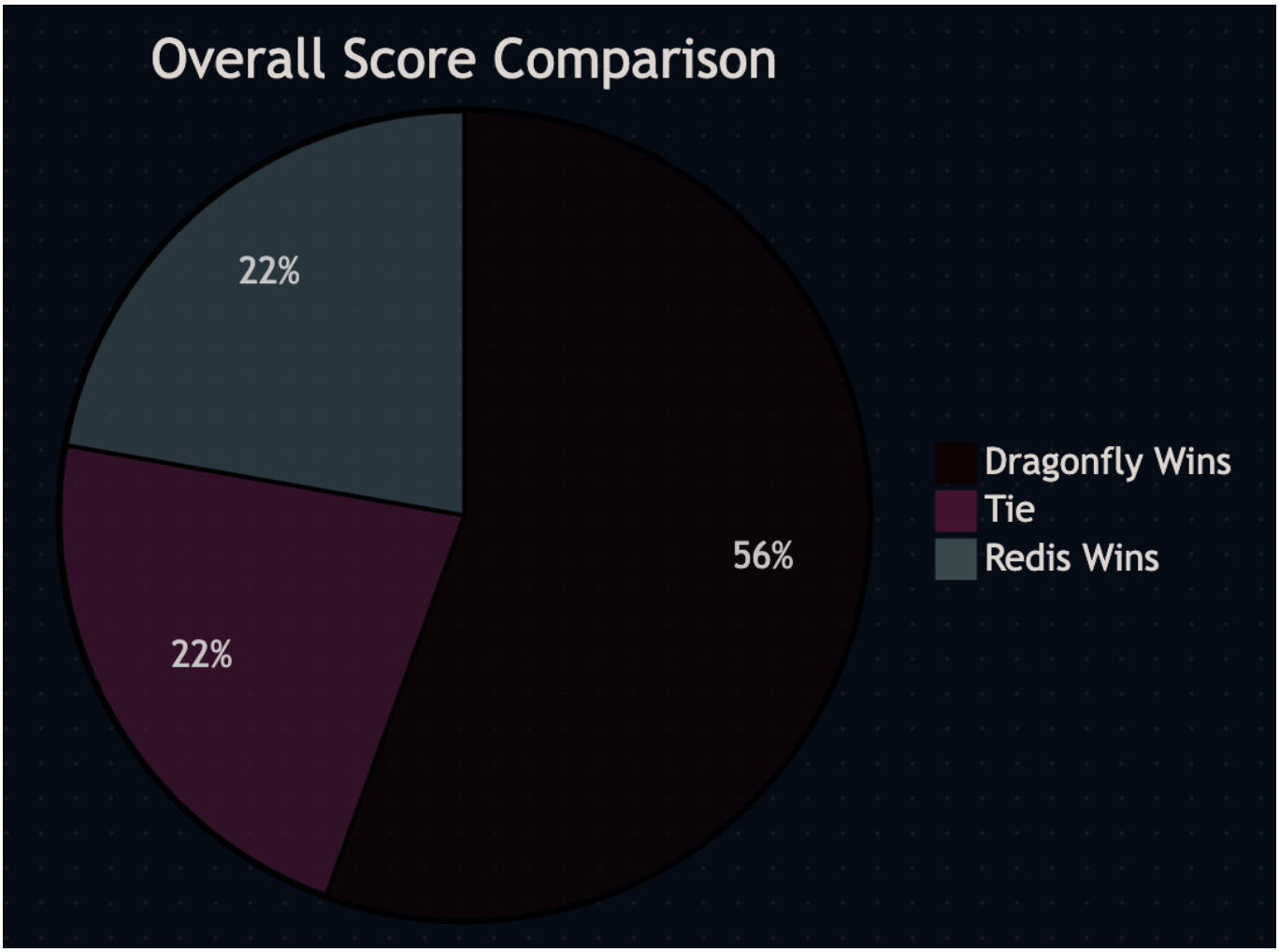
#	Finding	Impact
1️⃣	**Dragonfly vượt trội về Throughput**	+37% capacity cho Uniform, +29% cho Hot Key
2️⃣	**Dragonfly vượt trội về Latency**	2-7x faster P99 latency ở cùng mức load
3️⃣	**Dragonfly Resilient hơn**	Graceful degradation vs catastrophic collapse
4️⃣	**Hot Key là kẻ thù chung**	Multi-layer rate limiting là bắt buộc

9.2 📋 Decision Matrix

```
╔═══════════════════════════════════════════════════════════════════════════╗
║                                                                           ║
║                    🎯 FINAL RECOMMENDATION                                ║
║                                                                           ║
╚═══════════════════════════════════════════════════════════════════════════╝

  💬 "Tôi đang dùng Redis và throughput < 15k RPS"
  → ✅ Giữ nguyên Redis, không cần thay đổi

  💬 "Tôi đang dùng Redis và throughput 15k–20k RPS"
  → 🟡 Cân nhắc migrate sang Dragonfly để có headroom

  💬 "Tôi đang dùng Redis và throughput > 20k RPS"
  → 🔴 KHUYẾN NGHỊ MẠNH: Migrate sang Dragonfly hoặc Redis Cluster
```

```
╔═══════════════════════════════════════════════════════════════════════════╗
║                                                                           ║
║  💬 "Tôi đang setup mới và expect traffic cao"                            ║
║  → 🟢 Bắt đầu với Dragonfly từ đầu - drop-in replacement, no regrets      ║
║                                                                           ║
║  💬 "Tôi có Hot Key problem"                                              ║
║  → ⚠️ Dragonfly giúp, nhưng PHẢI có Local Cache layer                     ║
║                                                                           ║
╚═══════════════════════════════════════════════════════════════════════════╝
```

9.3 🏆 Final Verdict



Criteria	Winner	Score/Detail
📈 **Raw Throughput**	🏆 Dragonfly	37% higher
⚡ **Latency**	🏆 Dragonfly	2-7x better

Criteria	Winner	Score/Detail
🛡️ **Resilience**	🏆 Dragonfly	Graceful degradation
🔄 **Compatibility**	🤝 Tie	Both Redis protocol
📚 **Ecosystem**	🏆 Redis	More mature, more tools
👥 **Community**	🏆 Redis	Larger, more resources
🔧 **Operational**	🤝 Tie	Both simple single-node
💰 **Cost**	🤝 Tie	Both open source
🔥 **Hot Key Handling**	🏆 Dragonfly	Better isolation

🎖️ Overall Winner: **DRAGONFLY**

cho use-case Rate Limiting high-throughput

```
╔═══════════════════════════════════════════════════════════════════════════════
═══════╗
║                                                                               
       ║
║    💡 BOTTOM LINE                                                             
       ║
║                                                                               
       ║
╠═══════════════════════════════════════════════════════════════════════════════
═══════╣
║                                                                               
       ║
║   "Nếu bạn đang chạm ngưỡng 15-20k RPS trên Redis và đang cân nhắc             
       ║
║    Redis Cluster, hãy thử Dragonfly trước.                                    
       ║
║                                                                               
       ║
║    Nó có thể giải quyết vấn đề performance ngay lập tức (Vertical
Scaling)    ║
║    mà không cần thay đổi topology phức tạp (Horizontal Scaling)."             
       ║
║                                                                               
       ║
║                                                                               
       ║
║                                                  - Rate Limit Benchmark
Team ║
║                                                                               
       ║
╚═══════════════════════════════════════════════════════════════════════════════
═══════╝
```

📎 Appendix

A. Test Artifacts

Artifact	Location
Redis Single Key Report	`single-node-redis/single_key_20260127_214303/`
Redis 100 Keys Uniform Report	`single-node-redis/hundred_keys_20260128_071830/`
Redis 100 Keys Hot Report	`single-node-redis/hundred_keys_hot_20260201_184238/`
Dragonfly 100 Keys Uniform Report	`single-node-dragonfly/hundred_keys_20260204_12345/`
Dragonfly 100 Keys Hot Report	`single-node-dragonfly/hundred_keys_hot_20260204_184238/`

B. Infrastructure Configuration

File	Purpose
`single-node-benchmark/docker-compose.yml`	Redis test environment
`dragonfly-single-node-benchmark/docker-compose.yml`	Dragonfly test environment
`*/envoy.yaml`	Load balancer config
`*/prometheus.yml`	Metrics collection
`*/grafana/dashboards/benchmark.json`	Visualization

C. Reproducing Results

```
# 1. Start Redis benchmark environment
cd single-node-benchmark
docker compose up -d

# 2. Run benchmark (example: 100 keys uniform)
cd client
./gradlew run --args="localhost:9091 key 8 60 5" \
  -PmainClass=com.example.ratelimit.client.HundredKeyBenchmark

# 3. Start Dragonfly benchmark environment
cd ../../dragonfly-single-node-benchmark
docker compose up -d

# 4. Run same benchmark against Dragonfly
cd client
./gradlew run --args="localhost:9091 key 8 60 5" \
  -PmainClass=com.example.ratelimit.client.HundredKeyBenchmark

# 5. View results in Grafana
open http://localhost:3000
```

D. Glossary

Term	Definition
RPS	Requests Per Second
P99 Latency	99th percentile latency (99% of requests complete faster)
Hot Key	A single key receiving disproportionate traffic
Graceful Degradation	System performance declines gradually under load
Circuit Breaker	Pattern to prevent cascade failures
Token Bucket	Rate limiting algorithm using tokens

📊 **Rate Limit Benchmark Report**

Comparing Redis vs Dragonfly for High-Throughput Rate Limiting

---

Document generated: February 5, 2026