HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



Nguyễn Đoàn Quân

NGHIÊN CỨU MÔ HÌNH CÂN BẰNG TẢI ÁP DỤNG LÝ THUYẾT HÀNG ĐỢI CHO GAME ONLINE NHIỀU NGƯỜI CHƠI

ĐỀ ÁN THẠC SĨ KỸ THUẬT

(Theo định hướng ứng dụng)

HÀ NỘI 2025

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



Nguyễn Đoàn Quân

NGHIÊN CỨU MÔ HÌNH CÂN BẰNG TẢI ÁP DỤNG LÝ THUYẾT HÀNG ĐỢI CHO GAME ONLINE NHIỀU NGƯỜI CHƠI

CHUYÊN NGÀNH: HỆ THỐNG THÔNG TIN

Mã số: 8.48.01.04

ĐỀ ÁN THẠC SĨ NGÀNH HỆ THỐNG THÔNG TIN

NGƯỜI HƯỚNG DẪN KHOA HỌC: PGS. TS. LÊ HỮU LẬP

HÀ NỘI 2025

LỜI CAM ĐOAN

Tôi xác nhận rằng Đề án này là kết quả nghiên cứu của riêng tôi, được thực hiện dưới sự hướng dẫn của PGS. TS. Lê Hữu Lập. Các thông tin, kết quả phân tích được thu thập và xử lý một cách trung thực, khách quan.

Tôi hiểu về các quy định liên quan đến sự trung thực trong học thuật và xin cam đoan rằng nghiên cứu này đã được tôi thực hiện mà không vi phạm bất kỳ yêu cầu nào về sự trung thực trong học thuật.

Hà Nội, ngày tháng năm 2025

Người cam đoan

Nguyễn Đoàn Quân

LỜI CẢM ƠN

Sau gần hai năm tham gia chương trình cao học ngành Hệ thống thông tin tại Học viện Công nghệ Bưu chính Viễn thông, tôi xin gửi lòng biết ơn sâu sắc đến các Thầy/Cô giáo công tác tại Học viện. Nhờ sự dạy dỗ và hướng dẫn tận tình, tôi đã tích luỹ được nhiều kiến thức quý báu và đã phát triển khả năng tư duy một cách đáng kể. Tôi muốn thể hiện sự cảm kích đặc biệt đối với PGS. TS. Lê Hữu Lập, thầy đã dành rất nhiều thời gian và tâm huyết, hướng dẫn tôi trong quá trình thực hiện luận văn. Thầy đã chỉ dạy và hỗ trợ tôi một cách tận tâm, giúp tôi xác định hướng nghiên cứu, giải quyết vấn đề và hoàn thành đề án này một cách thành công.

Tôi cũng xin chân thành cảm ơn các anh chị cán bộ, giảng viên của Học viện Công nghệ Bưu chính Viễn thông, nơi tôi đang làm việc, vì đã nhiệt tình hỗ trợ, cung cấp thông tin, chia sẻ kinh nghiệm và đóng góp nhiều ý kiến quý báu và động viên tinh thần để tôi có thể hoàn thành tốt nội dung nghiên cứu.

Xin chân thành cảm ơn mọi người!

MỤC LỤC

LỜI CAM ĐOAN ..... **i**
LỜI CẢM ƠN ..... **ii**
MỤC LỤC ..... **iii**
DANH SÁCH HÌNH ẢNH ..... **vi**
DANH SÁCH BẢNG ..... **vii**
DANH MỤC KÝ HIỆU TỪ VIẾT TẮT ..... **viii**
LỜI MỞ ĐẦU **1**
CHƯƠNG 1. NGHIÊN CỨU TỔNG QUAN **5**
1.1. Tổng quan về game **5**
1.1.1. Khái niệm và bối cảnh 5
1.1.2. Phân loại Game Online dựa trên kiến trúc hệ thống 5
1.2. Các đặc trưng về game online **6**
1.2.1. Đặc trưng về lưu lượng truy cập 7
1.2.2. Đặc trưng về thời gian phản hồi (Response Time & Latency) 8
1.2.3. Hành vi tương tác của người chơi 8
1.3. Kiến trúc mô hình hệ thống trong game online **9**
1.3.1. Cơ chế vận hành của mô hình Client-Server 9
1.3.2. Các kỹ thuật xử lý độ trễ và đồng bộ hóa 10
1.3.3. Kết luận về kiến trúc 12
1.4. Các kỹ thuật cân bằng tải **12**
1.4.1. Các kỹ thuật cân bằng tải 13
1.4.2. Đánh giá hiệu quả của các thuật toán 24
1.4.3. So sánh mức độ đáp ứng yêu cầu của Game Online với ứng dụng web 26
1.5. Cơ sở lý thuyết hàng đợi và khả năng ứng dụng trong mô hình hệ thống game online **28**

1.5.1. Định nghĩa 28
1.5.2. Mối liên hệ giữa lý thuyết hàng đợi và mô hình xử lý yêu cầu của máy chủ game 29
1.5.3. Các tình huống thường gặp trong hệ thống game online 30
1.5.4. Quy trình tích hợp hàng đợi vào cơ chế cân bằng tải 31
1.6. Kết luận chương 1 31
CHƯƠNG 2. Đề xuất và phân tích mô hình cân bằng tải cho game online 33
2.1. Phân tích yêu cầu 33
2.1.1. Các yêu cầu về hiệu năng và đặc tả kỹ thuật 33
2.1.2. Các vấn đề và thách thức khi áp dụng kỹ thuật cân bằng tải truyền thống 34
2.2. Lựa chọn kỹ thuật cân bằng tải phù hợp 37
2.2.1. Tiêu chí đánh giá và sàng lọc giải pháp 37
2.2.2. Phân tích loại trừ các kỹ thuật truyền thống 38
2.2.3. Đề xuất giải pháp: Mô hình cân bằng tải dự báo dựa trên hàng đợi 39
2.3. Thiết kế mô hình đề xuất 42
2.3.1. Sơ đồ kiến trúc tổng thể 42
2.3.2. Các thành phần chính và chức năng 43
2.4. Quy trình xử lý yêu cầu trong mô hình 44
2.4.1. Luồng xử lý yêu cầu kết nối 45
2.4.2. Cơ chế phối hợp giữa cân bằng tải và hàng đợi 46
2.5. Phân tích ưu và nhược điểm của mô hình 47
2.5.1. Ưu điểm nổi bật so với kỹ thuật truyền thống 47
2.5.2. Hạn chế và thách thức 48
2.5.3. Điều kiện áp dụng hiệu quả 49
2.6. Kết luận chương 2 49

CHƯƠNG 3. THỬ NGHIỆM VÀ ĐÁNH GIÁ 51

3.1. Mục tiêu thử nghiệm 51

3.2. Môi trường thử nghiệm 51

3.2.1. Kiến trúc triển khai 51

3.2.2. Cấu hình môi trường 52

3.2.3. Công cụ và tham số 52

3.3. Kịch bản Thử nghiệm 52

3.3.1. Thử nghiệm kịch bản 1: Đánh giá giới hạn của kỹ thuật truyền thống.. 52

3.3.2. Thử nghiệm kịch bản 2: Áp dụng thuật toán QPLB với chiến lược tràn tải 58

3.3.3. Thử nghiệm kịch bản 3: Cân bằng tải thích ứng dựa trên đo lường tài nguyên thời gian thực 60

3.4. Chỉ số đánh giá 66

3.4.1. Định nghĩa các chỉ số khoa học 66

3.4.2. Bảng tổng hợp so sánh hiệu năng giữa các thuật toán 66

3.5. Kết quả thử nghiệm 71

3.6. Kết luận chương 3 71

KẾT LUẬN 73

TÀI LIỆU THAM KHẢO 75

DANH SÁCH HÌNH ẢNH

Hình ảnh 1.1 Cấu trúc mô hình Khách – Máy chủ 10
Hình ảnh 1.2 Cloud Load Balancer 13
Hình ảnh 1.3 Thuật toán Round Robin 14
Hình ảnh 1.4 Thuật toán Weighted Round Robin 16
Hình ảnh 1.5 Thuật toán Least Connection 18
Hình ảnh 1.6 Least Response Time 21
Hình ảnh 1.7 Thuật toán IP Hash 23
Hình ảnh 2.1 Sơ đồ kiến trúc luồng dữ liệu của mô hình QPLB 43
Hình ảnh 3.1 Hình ảnh kết quả thử nghiệm cân bằng tải sử dụng thuật toán Round Robin 53
Hình ảnh 3.2 Node 1 đã bị restart 5 lần trong khoảng thời gian 34s khi thực hiện thực nghiệm 56
Hình ảnh 3.3 Hiệu suất hoạt động của node 2 57
Hình ảnh 3.4 Hiệu suất hoạt động node 3 57
Hình ảnh 3.5 Hình ảnh thực nghiệm phân phối kết nối QPLB với chiến lược tràn tải 59
Hình ảnh 3.6 Hàm tính giá trị Cost theo lý thuyết hàng đợi 61
Hình ảnh 3.7 Xử lý điều phối các yêu cầu của client 62
Hình ảnh 3.8 Kết quả điều phối các yêu cầu từ client vào các node trong máy chủ 64
Hình ảnh 3.9 Hình ảnh phân phối tải trọng giữa các thuật toán 69
Hình ảnh 3.10 Hình ảnh mức tiêu thụ Ram tại node 1 70
Hình ảnh 3.11 Hình ảnh biến thiên Ram tại node 2 70

DANH SÁCH BẢNG

Bảng 1.1 Bảng so sánh đặc trưng kỹ thuật giữa các thể loại game 6
Bảng 1.2 Bảng tiêu chí so sánh mức độ đáp ứng yêu cầu của máy chủ 27
Bảng 1.3 Ánh xạ các thành phần giữa mô hình Hàng đợi và Hệ thống Game Online 29
Bảng 3.1 Bảng thống kê kết nối khi thử nghiệm cân bằng tải bằng thuật toán Round Robin 53
Bảng 3.2 Bảng tổng hợp kết quả thử nghiệm 65
Bảng 3.3 Bảng tổng hợp đánh giá hiệu năng của hệ thống thông qua thử nghiệm .. 67
Bảng 3.4 Bảng tổng hợp các chỉ số đánh giá hiệu năng 67

DANH MỤC KÝ HIỆU TỪ VIẾT TẮT

Từ viết tắt	Từ đầy đủ	Nghĩa Tiếng Việt
FPS	First-Person Shooter	Trò chơi bắn súng góc nhìn thứ nhất
HTTP	HyperText Transfer Protocol	Giao thức truyền tải siêu văn bản
LC	Least Connections	Thuật toán kết nối ít nhất
LRT	Least Response Time	Thuật toán thời gian phản hồi thấp nhất
MMO	Massively Multiplayer Online	Trực tuyến nhiều người chơi (quy mô lớn)
MMOGs	Massively Multiplayer Online Games	Các trò chơi trực tuyến nhiều người chơi
MMORPG	Massively Multiplayer Online Role-Playing Game	Trò chơi nhập vai trực tuyến nhiều người chơi
P2P	Peer-to-Peer	Mạng ngang hàng
QoE	Quality of Experience	Trải nghiệm người dùng
QoS	Quality of Service	Chất lượng dịch vụ
QPLB	Queue-Prediction Based Load Balancing	Cân bằng tải dự báo dựa trên hàng đợi
RR	Round Robin	Thuật toán phân phối vòng tròn
TCP	Transmission Control Protocol	Giao thức điều khiển truyền tải
UDP	User Datagram Protocol	Giao thức truyền thông kết nối

WRR	Weighted Round Robin	Thuật toán Round Robin có trọng số

LỜI MỞ ĐẦU

1. Lý do chọn đề tài

Trong thập kỷ qua, ngành công nghiệp giải trí kỹ thuật số đã chứng kiến sự bùng nổ mạnh mẽ, đặc biệt là lĩnh vực trò chơi trực tuyến nhiều người chơi. Với doanh thu toàn cầu dự kiến vượt mốc 300 tỷ USD và hàng tỷ người chơi, các hệ thống game hiện đại không chỉ đơn thuần là phần mềm giải trí mà đã trở thành những hệ thống phân tán phức tạp bậc nhất. Đặc thù của các trò chơi này là yêu cầu duy trì hàng triệu kết nối đồng thời trong thời gian thực, với độ trễ cực thấp để đảm bảo tính tương tác và trải nghiệm người dùng.

Để giải quyết bài toán quy mô, kỹ thuật cân bằng tải đóng vai trò xương sống trong việc phân phối lưu lượng truy cập tới cụm máy chủ. Tuy nhiên, thực tế vận hành cho thấy các thuật toán cân bằng tải truyền thống như Round Robin hay Least Connections – vốn được thiết kế tối ưu cho các ứng dụng Web – lại bộc lộ nhiều hạn chế nghiêm trọng khi áp dụng vào môi trường Game Online. Sự biến thiên ngẫu nhiên của lưu lượng người và sự không đồng nhất về cấu hình phần cứng thường dẫn đến tình trạng: một số máy chủ bị quá tải cục bộ dẫn đến sập nguồn, trong khi các máy chủ khác vẫn nhàn rỗi.

Lý thuyết hàng đợi cung cấp một khung tham chiếu toán học vững chắc để mô hình hóa hành vi chờ đợi và tắc nghẽn trong các hệ thống dịch vụ. Việc ứng dụng lý thuyết này vào cân bằng tải hứa hẹn mang lại khả năng "dự báo" sớm các điểm nghẽn cổ chai dựa trên năng lực xử lý thực tế, thay vì chỉ phản ứng thụ động dựa trên số lượng kết nối.

Xuất phát từ nhu cầu cấp thiết về việc tối ưu hóa hiệu năng hệ thống máy chủ game và khắc phục nhược điểm của các giải pháp truyền thống, tôi lựa chọn đề tài: “Nghiên cứu mô hình cân bằng tải áp dụng lý thuyết hàng đợi cho Game Online nhiều người chơi” làm đề án tốt nghiệp thạc sĩ. Đề tài không chỉ có ý nghĩa khoa học trong việc kết hợp toán học ứng dụng với kỹ thuật hệ thống, mà còn mang giá trị thực tiễn cao đối với các nhà phát hành game trong việc đảm bảo chất lượng dịch vụ.

2. Tổng quan về vấn đề nghiên cứu

Bài toán cân bằng tải trong hệ thống phân tán đã được nghiên cứu rộng rãi trong nhiều thập kỷ. Các giải pháp kinh điển như Round Robin (phân phối vòng tròn) hay Least Connections (kết nối ít nhất) đã trở thành tiêu chuẩn trong các bộ cân bằng tải phần cứng và phần mềm (như Nginx, HAProxy). Tuy nhiên, các nghiên cứu gần đây của H. Chen và các cộng sự đã chỉ ra rằng, đối với các ứng dụng nhạy cảm với độ trễ như Game Online, các thuật toán này thường thất bại trong việc duy trì độ ổn định khi hệ thống chịu tải cao hoặc khi hạ tầng phần cứng không đồng nhất [13].

Vấn đề cốt lõi nằm ở chỗ các thuật toán truyền thống thường hoạt động theo cơ chế "mù trạng thái" hoặc giả định năng lực xử lý của các máy chủ là ngang bằng nhau. Trong thực tế, năng lực phục vụ của một máy chủ game biến thiên động theo mức độ tiêu thụ tài

nguyên (RAM/CPU). Khi tài nguyên cạn kiệt, tốc độ xử lý giảm sút, dẫn đến thời gian chờ trong hàng đợi tăng vọt theo hàm mũ – điều mà các thuật toán đếm số lượng kết nối đơn thuần không thể nhận diện kịp thời.

Trên thế giới, hướng nghiên cứu áp dụng lý thuyết hàng đợi (như mô hình M/M/1, M/M/c) vào quản lý mạng đã đạt được một số thành tựu nhất định trong lĩnh vực viễn thông và điện toán đám mây. Tuy nhiên, việc cụ thể hóa các mô hình toán học này thành một thuật toán cân bằng tải thực tế, có khả năng thích ứng với các tham số tài nguyên thời gian thực cho máy chủ Game vẫn là một thách thức mở cần được giải quyết chi tiết.

3. Mục đích nghiên cứu

Mục đích cốt lõi của đề án là nghiên cứu và kiểm chứng một mô hình cân bằng tải mới, thông minh hơn, có khả năng khắc phục các nhược điểm của hệ thống cũ. Cụ thể bao gồm:

a) Mô hình hóa toán học: Áp dụng lý thuyết hàng đợi (cụ thể là mô hình M/M/1 với tốc độ phục vụ biến thiên) để biểu diễn mối quan hệ giữa tải trọng đầu vào, tài nguyên hệ thống và thời gian chờ dự kiến.

b) Phát triển thuật toán QPLB (Queue-Prediction Based Load Balancing): Thiết kế và cài đặt thuật toán cân bằng tải có khả năng dự báo tắc nghẽn. Thuật toán này sử dụng dữ liệu đo lường từ xa (như mức tiêu thụ RAM, CPU) để tính toán "Chi phí tắc nghẽn" và tự động điều hướng lưu lượng tránh xa các máy chủ đang suy giảm hiệu năng.

c) Đánh giá thực nghiệm: Triển khai mô hình trên môi trường giả lập sát thực để chứng minh hiệu quả vượt trội của thuật toán đề xuất so với các thuật toán cân bằng tải truyền thống về các chỉ số: tỷ lệ lỗi kết nối, độ ổn định và khả năng tận dụng tài nguyên.

4. Đối tượng và phạm vi nghiên cứu

a) Đối tượng nghiên cứu:

- Hệ thống kiến trúc máy chủ Game Online theo mô hình Client-Server.
- Các thuật toán cân bằng tải: Round Robin, Least Connections, IP Hash và thuật toán đề xuất QPLB.
- Các mô hình toán học trong Lý thuyết hàng đợi.
- Các chỉ số hiệu năng hệ thống: Thời gian phản hồi, thông lượng, tỷ lệ lỗi và mức tiêu thụ tài nguyên.

b) Phạm vi nghiên cứu:

- Đề tài tập trung vào tầng phân phối kết nối của hệ thống Game.
- Nghiên cứu được thực hiện trên môi trường giả lập sử dụng công nghệ Container (Docker) và nền tảng Node.js để mô phỏng các cụm máy chủ game.
- Đề tài không đi sâu vào việc tối ưu hóa mã nguồn trò chơi hay đồ họa, mà tập trung giải quyết bài toán định tuyến và quản lý tài nguyên hệ thống hạ tầng.

5. Phương pháp nghiên cứu

Để đạt được các mục tiêu đề ra, đề tài sử dụng kết hợp các phương pháp nghiên cứu sau:

a) Phương pháp nghiên cứu lý thuyết:
   - Thu thập, tổng hợp và phân tích các tài liệu khoa học, bài báo chuyên ngành về kiến trúc game server, kỹ thuật cân bằng tải và lý thuyết hàng đợi.
   - Sử dụng toán học để xây dựng công thức tính toán thời gian chờ kỳ vọng dựa trên mô hình M/M/1.

b) Phương pháp mô hình hóa và giả lập:
   - Thiết kế kiến trúc hệ thống bao gồm: Load Balancer trung tâm, cụm Game Server và Client tạo tải giả lập.
   - Sử dụng Docker và PM2 để quản lý và giới hạn tài nguyên cho từng node nhằm tái hiện các kịch bản quá tải.

c) Phương pháp thực nghiệm và so sánh:
   - Xây dựng các kịch bản kiểm thử đa dạng: Kịch bản cơ sở, kịch bản chênh lệch cấu hình, và kịch bản quá tải tài nguyên.
   - Thực hiện đo đạc số liệu định lượng đối với từng thuật toán.
   - So sánh đối chứng kết quả giữa thuật toán đề xuất (QPLB) và các thuật toán truyền thống để rút ra kết luận khoa học về tính hiệu quả và khả năng ứng dụng.

6. Nội dung chính của đề án gồm 3 chương.

- Chương 1: Nghiên cứu tổng quan về game online, các kỹ thuật cân bằng tải, cơ sở lý thuyết hàng đợi và khả năng ứng dụng trong mô hình hệ thống cân bằng tải trong game online.

- Chương 2: Xây dựng cơ sở lý luận và thiết kế chi tiết cho giải pháp cân bằng tải dự báo dựa trên hàng đợi.
- Chương 3: Thử nghiệm và đánh giá kết quả về giải pháp cân bằng tải được đề xuất.

Do thời gian có hạn nên đề án không tránh khỏi những hạn chế nhất định. Mong các thầy cô, đồng nghiệp thông cảm và góp ý để đề án được hoàn thiện hơn.

CHƯƠNG 1. NGHIÊN CỨU TỔNG QUAN

1.1. Tổng quan về game

1.1.1. Khái niệm và bối cảnh

Trò chơi điện tử (Video Games) là các phần mềm ứng dụng tương tác, cho phép người dùng điều khiển các đối tượng trong môi trường ảo. Trong kỷ nguyên số, ngành công nghiệp game đã phát triển vượt bậc với doanh thu toàn cầu ước tính đạt hàng trăm tỷ USD, trong đó thị trường Việt Nam đóng góp một tỷ trọng đáng kể với lượng người chơi trực tuyến lớn.

Dựa trên yêu cầu về hạ tầng mạng, trò chơi điện tử được chia thành hai nhóm chính:

- Game Offline: Vận hành độc lập trên thiết bị người dùng, không yêu cầu kết nối máy chủ liên tục.
- Game Online: Yêu cầu kết nối mạng thường trực để tương tác thời gian thực giữa người chơi với nhau và với máy chủ trung tâm. Đây là đối tượng nghiên cứu chính của đề tài do các yêu cầu khắt khe về băng thông và xử lý dữ liệu.

1.1.2. Phân loại Game Online dựa trên kiến trúc hệ thống

Thay vì phân loại theo nội dung (như Bắn súng, Nhập vai, Chiến thuật), góc độ hệ thống thông tin phân chia Game Online thành hai mô hình chính dựa trên quy mô và cơ chế duy trì phiên:

1. Trò chơi dựa trên phiên đấu (Session-based Multiplayer):
    - Bao gồm các thể loại như MOBA (*League of Legends*), FPS (*CS:GO*), Battle Royale (*PUBG*).
    - Đặc điểm: Người chơi được chia vào các phòng nhỏ (Room/Match) với số lượng giới hạn (10-100 người). Khi trận đấu kết thúc, trạng thái phòng chơi được giải phóng.
2. Trò chơi trực tuyến nhiều người chơi (Massively Multiplayer Online - MMO):
    - Bao gồm MMORPG (*World of Warcraft*), Sandbox MMO, MMORTS.

- Đặc điểm: Hàng nghìn đến hàng triệu người chơi cùng tham gia vào một thế giới ảo duy nhất và liên tục (Persistent World). Trạng thái nhân vật và thế giới vẫn tồn tại và thay đổi ngay cả khi người chơi ngoại tuyến.
- *Thách thức:* Đây là mô hình phức tạp nhất, đòi hỏi hệ thống cân bằng tải phải xử lý lượng kết nối khổng lồ đồng thời và duy trì tính nhất quán dữ liệu trong thời gian dài.

Bảng 1.1 Bảng so sánh đặc trưng kỹ thuật giữa các thể loại game

Tiêu chí	Game Offline	Game Online	Game Online (MMO)
Kết nối mạng	Không yêu cầu	Yêu cầu liên tục trong trận đấu	Yêu cầu liên tục 24/7
Quy mô đồng thời	1 người	Thấp (10 - 100 người/phòng)	Rất lớn (Hàng nghìn người/Server)
Lưu trữ trạng thái	Cục bộ trên máy người chơi	Tạm thời (Reset sau mỗi trận)	Vĩnh viễn
Kiến trúc Server	Không có	Phân tán	Tập trung
Áp lực Cân bằng tải	Không có	Trung bình	Rất cao

Đề tài sẽ tập trung nghiên cứu sâu vào nhóm Game MMO, nơi mà các kỹ thuật cân bằng tải đóng vai trò sống còn trong việc duy trì độ ổn định và trải nghiệm người dùng trước lượng truy cập biến động lớn.

1.2. Các đặc trưng về game online

Việc phân tích và mô hình hóa lưu lượng mạng đóng vai trò then chốt trong việc thiết kế kiến trúc hạ tầng cho các hệ thống trò chơi trực tuyến. Các nghiên cứu thực nghiệm đã chứng minh rằng, lưu lượng sinh ra bởi các ứng dụng giải trí tương tác mang những đặc thù kỹ thuật hoàn toàn khác biệt so với các dịch vụ Internet truyền thống như duyệt web

(HTTP) hay truyền tải video [1]. Việc nắm bắt chính xác các đặc trưng này là tiền đề để xây dựng các giải pháp cân bằng tải và định tuyến hiệu quả.

1.2.1. Đặc trưng về lưu lượng truy cập

Lưu lượng của các trò chơi trực tuyến, đặc biệt là các thể loại yêu cầu phản xạ nhanh như Bắn súng góc nhìn thứ nhất hay Nhập vai trực tuyến, được định hình bởi ba yếu tố kỹ thuật cốt lõi: kích thước gói tin nhỏ, tính chu kỳ cao và sự bất đối xứng về băng thông.

a. Kích thước gói tin và hiệu suất truyền tải

Khác với giao thức TCP truyền thống vốn tối ưu hóa việc đóng gói dữ liệu lớn để đạt thông lượng tối đa, lưu lượng game bao gồm các chuỗi gói tin có kích thước rất nhỏ [2]. Phân tích trên các máy chủ *Counter-Strike* cho thấy hơn 90% các gói tin vận chuyển trong mạng có kích thước dưới 200 bytes [1].

Có sự chênh lệch rõ rệt về kích thước gói tin giữa hai chiều truyền tải:

- Chiều nhận (Từ người chơi đến máy chủ): Kích thước trung bình rất nhỏ, chỉ khoảng 40 bytes. Các gói tin này chủ yếu chứa các lệnh điều khiển từ phía người chơi như di chuyển, thao tác kỹ năng [3].
- Chiều gửi (Từ máy chủ đến người chơi): Kích thước trung bình lớn hơn, dao động khoảng 129 bytes [4]. Sự gia tăng này xuất phát từ chức năng "quảng bá" của máy chủ: nó phải tổng hợp trạng thái từ nhiều người chơi và gửi trả lại thông tin cập nhật cục bộ cho từng client. Đặc điểm này tạo ra gánh nặng xử lý lớn cho các thiết bị mạng trung gian, vốn thường được tối ưu cho hiệu suất băng thông thay vì tốc độ xử lý gói tin [2].

b. Tính chu kỳ và mô hình phát sinh lưu lượng

Lưu lượng game thể hiện tính chu kỳ rất cao do sự chi phối của logic đồng bộ hóa trạng thái [1]. Máy chủ game được thiết kế để gửi cập nhật trạng thái đến tất cả người chơi theo các khoảng thời gian cố định. Ví dụ, một máy chủ hoạt động ở tần số 20Hz sẽ phát sinh dữ liệu mỗi 50ms một lần [5].

Cơ chế này tạo ra mô hình lưu lượng dạng "bùng nổ" thay vì dòng chảy liên tục. Mặc dù điều này giúp mô hình lưu lượng có tính dự báo cao trong dài hạn, nhưng lại gây áp lực

xử lý tức thời cực lớn lên các bộ định tuyến tại các thời điểm "nhịp" dữ liệu được gửi đi đồng loạt [6].

c. Giao thức truyền tải

Để đáp ứng yêu cầu khắt khe về thời gian thực, đa số các game hành động hiện đại sử dụng giao thức UDP thay vì TCP [7]. Việc sử dụng TCP với cơ chế truyền lại và chờ xác thực được coi là không thực tế, bởi lẽ trong môi trường game, một gói tin bị mất thà bị bỏ qua để cập nhật dữ liệu mới nhất còn hơn là chờ truyền lại. Dữ liệu được truyền lại thường đã trở nên lỗi thời và không còn giá trị sử dụng tại thời điểm nó đến đích [8].

1.2.2. Đặc trưng về thời gian phản hồi (Response Time & Latency)

Thời gian phản hồi là chỉ số Chất lượng Dịch vụ (QoS) quan trọng nhất, ảnh hưởng trực tiếp đến Trải nghiệm Người dùng (QoE) trong môi trường game online.

a. Độ nhạy cảm với độ trễ mạng

Các ứng dụng game được thiết kế để tối đa hóa tính tương tác, do đó chúng cực kỳ nhạy cảm với độ trễ và độ biến thiên trễ. Để đảm bảo tính công bằng, hệ thống thường phải thiết kế băng thông yêu cầu dựa trên mức bão hòa của liên kết mạng chậm nhất [9]. Tuy nhiên, thách thức lớn nhất thường nằm ở độ trễ xử lý tại các nút mạng trung gian và bộ cân bằng tải.

b. Tác động của bộ đệm (Buffering)

Trong định tuyến Internet thông thường, việc sử dụng bộ đệm lớn để xử lý tắc nghẽn là giải pháp phổ biến. Tuy nhiên, đối với game online, điều này gây ra tác dụng ngược. Các nghiên cứu đã chỉ ra nguyên lý: *"Gói tin bị trễ còn tệ hơn gói tin bị mất"* (Delayed packets can be worse than dropped packets) [10]. Việc đệm các gói tin cập nhật trạng thái (vốn được gửi mỗi 50ms) sẽ làm gia tăng độ trễ vượt quá ngưỡng chấp nhận được của tương tác thời gian thực. Do đó, hạ tầng mạng cho game cần ưu tiên khả năng chuyển tiếp nhanh (fast forwarding) cho các gói tin nhỏ thay vì tối ưu hóa thông lượng tổng thể [1].

1.2.3. Hành vi tương tác của người chơi

Hành vi của người chơi là biến số quan trọng quyết định tải trọng lên hệ thống máy chủ theo không gian và thời gian.

a) Phân phối thời gian phiên chơi

Trái ngược với giả định phổ biến rằng thời gian sử dụng dịch vụ Internet tuân theo phân phối đuôi nặng hoặc phân phối mũ, các dữ liệu thực nghiệm khẳng định thời gian một phiên chơi game tuân theo phân phối Weibull [11]. Thống kê cho thấy hơn 99% các phiên chơi kéo dài dưới 2 giờ [12].

Đáng chú ý, tỷ lệ rời bỏ biến thiên mạnh ở các phiên chơi ngắn. Người chơi có xu hướng thoát game rất nhanh nếu gặp trải nghiệm không tốt (lag, mất kết nối) hoặc không tìm được trận đấu phù hợp ngay từ đầu. Điều này nhấn mạnh tầm quan trọng của tốc độ phản hồi và sự ổn định của hệ thống cân bằng tải ngay tại thời điểm thiết lập kết nối [13].

b) Phân bố theo địa lý và thời gian

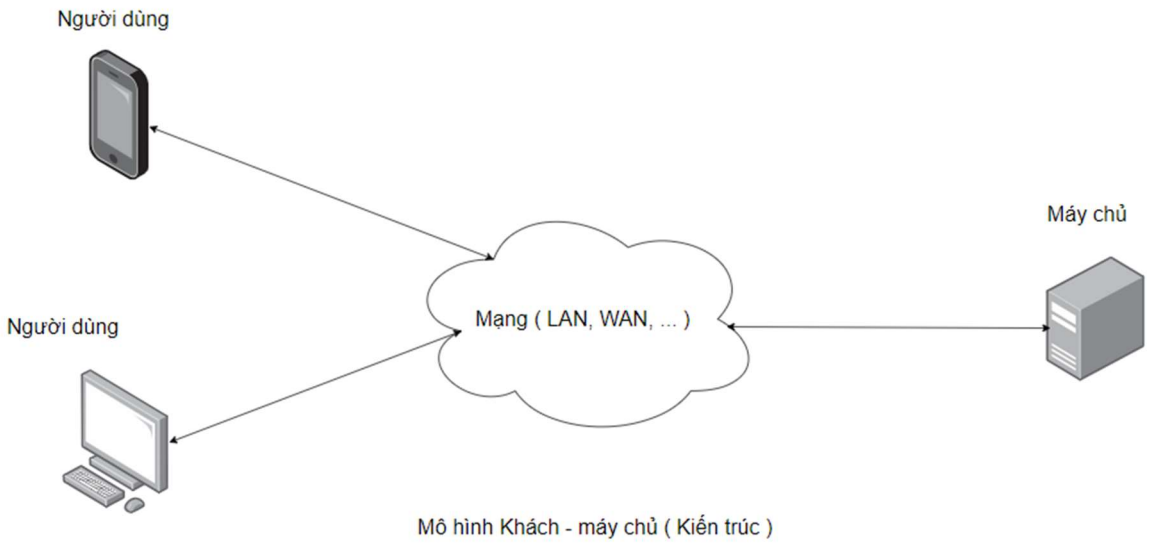
Hành vi kết nối của người chơi chịu ảnh hưởng lớn bởi chu kỳ sinh hoạt trong ngày. Mặc dù khoảng cách địa lý là yếu tố gây trễ, người chơi vẫn thường xuyên kết nối đến các máy chủ ở xa (thậm chí khác châu lục) vào các khung giờ thấp điểm tại địa phương để tìm kiếm đối thủ [14].

Dữ liệu cho thấy khoảng cách trung bình giữa người chơi và máy chủ đạt đỉnh vào sáng sớm và giảm xuống vào buổi chiều tối khi lượng người chơi cục bộ tăng lên [15]. Thực tế này gợi ý rằng các hệ thống mạng phân phối nội dung hoặc điện toán đám mây cho game cần tích hợp cơ chế điều phối tài nguyên động theo múi giờ để tối ưu hóa độ trễ [16].

1.3. Kiến trúc mô hình hệ thống trong game online

Trong hệ sinh thái các trò chơi trực tuyến nhiều người chơi, kiến trúc mạng không chỉ đóng vai trò là xương sống kết nối mà còn là yếu tố quyết định đến tính toàn vẹn dữ liệu và chất lượng trải nghiệm thời gian thực. Hiện nay, mô hình Client-Server (Khách-Chủ) đã trở thành tiêu chuẩn công nghiệp, đặc biệt đối với các thể loại yêu cầu tính nhất quán cao và khả năng chống gian lận nghiêm ngặt như Bắn súng góc nhìn thứ nhất (FPS) hay Nhập vai trực tuyến (MMORPG).

1.3.1. Cơ chế vận hành của mô hình Client-Server



Hình ảnh 1.1 Cấu trúc mô hình Khách – Máy chủ

Khác với kiến trúc Ngang hàng vốn dựa trên sự giao tiếp phân tán trực tiếp giữa các máy trạm, mô hình Client-Server hoạt động dựa trên nguyên tắc tập trung hóa kiểm soát.

- Vai trò trọng tài của máy chủ: Máy chủ đóng vai trò là "người nắm giữ chân lý", lưu trữ và quản lý trạng thái toàn cục của thế giới ảo. Nó tiếp nhận dữ liệu từ người chơi, thực thi logic game (vật lý, va chạm) và đồng bộ hóa trạng thái đến các máy trạm.
- **Chức năng của máy trạm:** Thu thập tương tác từ người dùng (phím, chuột), gửi về máy chủ và nhận bản cập nhật để hiển thị hình ảnh/âm thanh
- Chu trình luồng dữ liệu: Diễn ra vòng lặp liên tục gồm: Client gửi lệnh ➔ Server mô phỏng logic ➔ Server đồng bộ trạng thái thế giới

1.3.2. Các kỹ thuật xử lý độ trễ và đồng bộ hóa

- Mặc dù kiến trúc Client-Server tối ưu hóa tính bảo mật và nhất quán dữ liệu, nó lại đối mặt với thách thức vật lý cố hữu: độ trễ mạng. Do thông tin phải di chuyển hai chiều giữa người chơi và máy chủ, nếu không có các biện pháp can thiệp kỹ thuật, người chơi sẽ gặp phải trải nghiệm phản hồi chậm, làm giảm đáng kể tính tương tác. Để giải quyết vấn đề này, các hệ thống game hiện đại tích hợp ba kỹ thuật nâng cao sau:

a. Nguyên tắc máy chủ ủy quyền

Đây là nguyên tắc thiết kế nền tảng để đảm bảo tính toàn vẹn. Server được lập trình để không tin tưởng bất kỳ dữ liệu vị trí hoặc trạng thái nào do Client gửi lên. Thay vào đó, nó chỉ nhận các chuỗi lệnh điều khiển và tự mình mô phỏng lại toàn bộ hành động đó để xác thực tính hợp lệ. Cơ chế này loại bỏ hoàn toàn khả năng người chơi can thiệp vào bộ nhớ máy trạm để thực hiện các hành vi gian lận như dịch chuyển tức thời hay đi xuyên vật thể.

b. Dự đoán phía máy khách

Để triệt tiêu cảm giác trễ trong khi chờ phản hồi từ Server, Client sử dụng thuật toán dự đoán hành vi:

- Cơ chế dự đoán: Ngay khi người chơi thực hiện thao tác, Client lập tức mô phỏng cục bộ và cập nhật trạng thái nhân vật trên màn hình mà không cần chờ xác nhận từ Server.
- Cơ chế hòa giải: Khi nhận được bản cập nhật chính thức từ Server (thường đến trễ hơn vài chục mili-giây), Client sẽ so sánh trạng thái dự đoán của mình với trạng thái thực tế từ Server. Nếu phát hiện sai lệch (do mất gói tin hoặc xung đột vật lý không lường trước), Client buộc phải điều chỉnh lại vị trí nhân vật để khớp với Server. Khi sự sai lệch quá lớn, người chơi sẽ thấy nhân vật bị giật lùi về vị trí cũ, hiện tượng này được gọi là "Rubber banding".

c. Kỹ thuật bù trễ

Trong môi trường chiến đấu thời gian thực, sự chênh lệch độ trễ giữa các người chơi (ví dụ: người chơi A có ping 20ms đấu với người chơi B có ping 150ms) tạo ra sự bất công. Kỹ thuật bù trễ được áp dụng để giải quyết vấn đề này:

- Vấn đề: Do độ trễ truyền tải, hình ảnh người chơi B mà người chơi A nhìn thấy trên màn hình thực chất là vị trí của B trong quá khứ.
- Giải pháp: Khi Server nhận được lệnh "bắn" từ A kèm theo dấu thời gian, nó sẽ truy xuất bộ đệm lịch sử để "tua ngược" trạng thái thế giới game về đúng thời điểm đó. Server sau đó kiểm tra đường đạn trong bối cảnh quá khứ đã được tái dựng để xác

định kết quả trúng đích. Kỹ thuật này đảm bảo rằng phản xạ và kỹ năng của người chơi được ghi nhận chính xác bất chấp độ trễ mạng.

1.3.3. Kết luận về kiến trúc

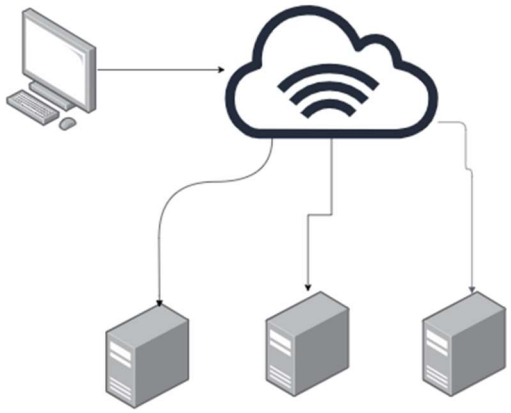
Tóm lại, mô hình Client-Server, khi kết hợp với giao thức truyền tải UDP và các kỹ thuật xử lý độ trễ tinh vi (dự đoán và bù trễ), đã trở thành kiến trúc tham chiếu tiêu chuẩn cho các trò chơi trực tuyến hiện đại. Nó giải quyết bài toán cân bằng phức tạp giữa việc duy trì sự đồng bộ trạng thái cho hàng ngàn người chơi, đảm bảo tính công bằng của trò chơi, và tối ưu hóa trải nghiệm mượt mà cho người dùng cuối bất chấp những giới hạn vật lý của hạ tầng Internet.

1.4. Các kỹ thuật cân bằng tải

Cân bằng tải là quá trình phân phối hiệu quả lưu lượng truy cập đến một nhóm các máy chủ hậu phương, hay còn được gọi là cụm máy chủ hoặc nhóm máy chủ.

Các trò chơi trực tuyến nhiều người chơi (MMO) hiện đại với lưu lượng truy cập cao phải phục vụ hàng trăm nghìn, thậm chí hàng triệu yêu cầu đồng thời từ người dùng hoặc khách hàng. Hệ thống phải phản hồi chính xác các văn bản, hình ảnh, video hoặc dữ liệu ứng dụng một cách nhanh chóng và đáng tin cậy. Để đáp ứng hiệu quả khối lượng lớn các yêu cầu và phản hồi này một cách tiết kiệm nhất, giải pháp tối ưu thường là bổ sung thêm các máy chủ.

Một bộ cân bằng tải (load balancer) hoạt động như một "cảnh sát giao thông" phía trước máy chủ, thực hiện việc định tuyến các yêu cầu từ máy khách đến tất cả các máy chủ có khả năng xử lý. Việc này nhằm tối ưu hóa tốc độ và hiệu suất, đồng thời đảm bảo không có máy chủ nào phải hoạt động quá tải. Nếu một máy chủ đơn lẻ gặp sự cố, bộ cân bằng tải sẽ tự động chuyển hướng lưu lượng truy cập đến các máy chủ trực tuyến còn lại. Khi một máy chủ mới được thêm vào nhóm, bộ cân bằng tải sẽ tự động bắt đầu phân phối yêu cầu đến máy chủ mới này.



Hình ảnh 1.2 Cloud Load Balancer

Tóm lại, một bộ cân bằng tải sẽ thực hiện các chức năng chính sau đây:

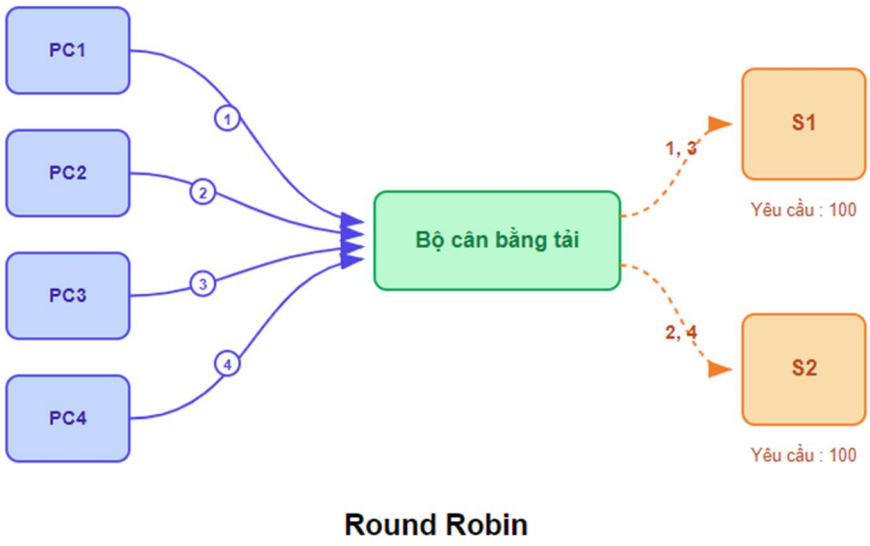
- Phân phối các yêu cầu người dùng hoặc network load một cách hiệu quả trên nhiều máy chủ.
- Đảm bảo tính khả dụng và độ tin cậy cao bằng cách chỉ gửi các yêu cầu đến các máy chủ trực tuyến.
- Thêm vào hoặc loại bớt các máy chủ theo yêu cầu một cách linh hoạt và dễ dàng.

1.4.1. Các kỹ thuật cân bằng tải truyền thống

Các thuật toán kinh điển như Round Robin và Least Connections đã được phân tích rộng rãi về hiệu quả phân phối tải trong các nghiên cứu trước đây [17]. Đối với các hệ thống yêu cầu tính duy trì phiên, thuật toán IP Hash cung cấp giải pháp định tuyến dựa trên địa chỉ nguồn [18]. Trong khi đó, thuật toán Least Response Time tập trung vào việc tối thiểu hóa độ trễ phản hồi, tuy nhiên cũng đối mặt với thách thức về sự ổn định thông tin tải [19, 20].

1.4.1.1. Thuật toán Round Robin

Là thuật toán điều phối vòng tròn, các máy chủ sẽ được xem ngang hàng và sắp xếp theo một vòng quay. Các truy vấn dịch vụ sẽ lần lượt được gửi tới các máy chủ theo thứ tự sắp xếp.



Hình ảnh 1.3 Thuật toán Round Robin

Thuật toán Round Robin (RR) là phương pháp cân bằng tải tĩnh phổ biến nhất, hoạt động dựa trên nguyên tắc phân phối yêu cầu theo trình tự vòng tròn. Trong kiến trúc này, các yêu cầu đến được phân bổ đều cho các máy chủ backend mà không xét đến tải trọng hiện thời hay thời gian phản hồi của từng nút.

Về mặt toán học, giả sử tập hợp các máy chủ khả dụng là $S = \{S_0, S_1, \dots, S_{n-1}\}$ với n là tổng số máy chủ. Nếu I là chỉ số của yêu cầu hiện tại (i tăng dần theo thời gian), máy chủ S_k được chọn để xử lý yêu cầu thứ i được xác định bởi công thức đồng dư k = i mod n

Trong đó:

k: Chỉ số index của máy chủ được chọn $(0 \leq k < n)$

mod: Phép toán chia lấy dư.

Cơ chế này đảm bảo tính công bằng tuyệt đối về số lượng yêu cầu được gửi đến mỗi máy chủ trong một chu kỳ xác định.

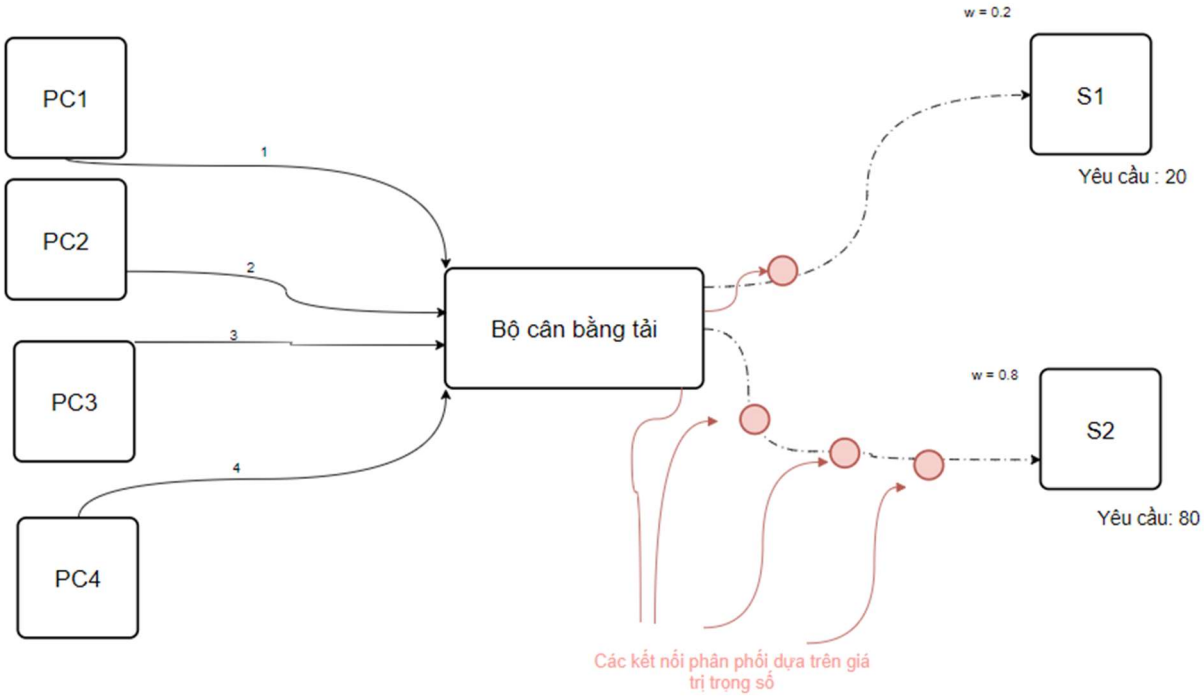
Cơ chế hoạt động của Round Robin trong hệ thống cân bằng tải được chuẩn hóa thành một chu trình khép kín gồm 4 giai đoạn chính:

1. Quản lý danh sách khả dụng: Bộ cân bằng tải quản lý và duy trì một danh sách động các máy chủ backend đang ở trạng thái hoạt động ổn định. Danh sách này đóng vai trò là tập hợp nguồn để phân phối các yêu cầu đến.
2. Phân phối tuần tự phi trạng thái: Khi một yêu cầu mới phát sinh, thuật toán thực hiện điều phối tuyến tính bằng cách chọn máy chủ kế tiếp trong danh sách theo trình tự định sẵn. Quá trình này diễn ra độc lập với các chỉ số tải trọng hiện thời (như mức tiêu thụ CPU hay RAM) của máy chủ, giúp tối ưu hóa tốc độ xử lý và giảm độ trễ tại tầng cân bằng tải.
3. Cơ chế vòng lặp: Quy trình phân phối tuân theo cấu trúc hàng đợi vòng. Ngay khi yêu cầu được gửi đến máy chủ cuối cùng trong danh sách, con trỏ điều phối sẽ tự động quay trở lại vị trí khởi đầu để bắt đầu một chu kỳ phân phối mới, đảm bảo mọi máy chủ đều nhận được lượt xử lý theo phiên.
4. Cơ chế tự phục hồi và giám sát sức khỏe: Để đảm bảo tính sẵn sàng cao, hệ thống tích hợp cơ chế giám sát sức khỏe định kỳ. Nếu phát hiện một máy chủ gặp sự cố hoặc không phản hồi, nó sẽ lập tức bị loại bỏ khỏi danh sách phân phối để cách ly lỗi, ngăn chặn việc gửi yêu cầu vào "ngõ cụt". Máy chủ này chỉ được tái gia nhập mạng lưới xử lý sau khi đã khôi phục hoàn toàn trạng thái hoạt động bình thường.

Trong thực tế, các backend server thường không hoàn toàn giống nhau. Có máy chủ mạnh (nhiều CPU, RAM lớn) và có máy chủ yếu hơn. Nếu dùng Round Robin cơ bản (chia đều 1-1), máy chủ yếu có thể bị quá tải trong khi máy chủ mạnh chưa dùng hết công suất. Weighted Round Robin ra đời để giải quyết vấn đề này.

Cơ chế hoạt động của thuật toán Weighted Round Robin**:** Thay vì chia đều, quản trị viên sẽ gán cho mỗi server một trọng số– một giá trị số nguyên đại diện cho năng lực xử lý của server đó.

- Quy tắc: Server có trọng số càng cao sẽ nhận được càng nhiều yêu cầu xử lý trong một chu kỳ phân phối.
- Mục tiêu: Cân bằng tải dựa trên năng lực thực tế thay vì số lượng máy chủ.



Hình ảnh 1.4 Thuật toán Weighted Round Robin

Mặc dù là một trong những thuật toán sơ khai nhất, Round Robin vẫn giữ vị trí quan trọng trong kỹ thuật cân bằng tải nhờ sự cân bằng giữa tính hiệu quả và độ phức tạp vận hành:

- Tối ưu hóa chi phí tính toán: Đây là ưu điểm vượt trội nhất của Round Robin. Do hoạt động theo cơ chế phi trạng thái và không yêu cầu thu thập dữ liệu tải trọng thời gian thực từ backend, thuật toán tiêu tốn rất ít tài nguyên CPU và bộ nhớ tại tầng Load Balancer. Điều này cho phép bộ cân bằng tải xử lý một lượng lớn yêu cầu đồng thời với độ trễ chuyển mạch gần như bằng không.
- Phân phối đồng đều về số lượng: Về mặt thống kê, Round Robin đảm bảo mọi máy chủ nhận được số lượng yêu cầu bằng nhau trong một chu kỳ đủ lớn. Điều này tạo ra sự công bằng tuyệt đối về mặt "lượt phục vụ", loại bỏ hoàn toàn hiện tượng bỏ sót đối với bất kỳ máy chủ nào trong cụm.

- Hạn chế về nhận thức tải trọng: Round Robin hoạt động với giả định rằng mọi yêu cầu đều tiêu tốn tài nguyên như nhau và mọi máy chủ đều có năng lực xử lý ngang bằng. Do đó, trong các tình huống thực tế phức tạp (nơi một request xử lý video nặng gấp 100 lần request văn bản), thuật toán có thể vô tình gây ra tình trạng nghẽn cổ chai cục bộ tại một số máy chủ trong khi các máy khác vẫn nhàn rỗi.
- Phạm vi ứng dụng: Thuật toán là giải pháp lý tưởng cho các mô hình kiến trúc sau:
    - Dịch vụ phi trạng thái: Các hệ thống RESTful API nơi mỗi yêu cầu là độc lập và không phụ thuộc vào dữ liệu phiên lưu trên máy chủ.
    - Cụm máy chủ đồng nhất: Môi trường mà tất cả các máy chủ backend có cấu hình phần cứng (CPU/RAM) và năng lực xử lý tương đương nhau.
    - Môi trường kiểm thử và Phát triển: Nơi cần sự đơn giản trong cấu hình và khả năng dự đoán trước hành vi phân phối để phục vụ việc debug lỗi.

1.4.1.2. Thuật toán Least connections:

Thuật toán Least Connection là một chiến lược cân bằng tải động, trong đó các yêu cầu mới được điều phối đến máy chủ có số lượng kết nối hoạt thấp nhất tại thời điểm ra quyết định. Khác với các thuật toán tĩnh như Round Robin, Least Connection đánh giá trạng thái thực tế của hệ thống để đảm bảo sự phân phối tải dựa trên năng lực xử lý tức thời của từng nút mạng.

Về mặt toán học, giả sử một cụm máy chủ bao gồm tập hợp S = {S1, S2,S3, …, Sn}. Gọi C(Si) là số lượng kết nối hiện hành trên máy chủ Si tại thời điểm t.

Máy chủ mục tiêu S_{target} được chọn để xử lý yêu cầu mới (R_{new}) thỏa mãn điều kiện:

$$S_{target} = \{ Si \in S \mid C(Si) = \min(C(S1), C(S2), \dots .C(Sn))\}$$

Trong trường hợp có nhiều máy chủ cùng đạt giá trị cực tiểu, thuật toán có thể áp dụng cơ chế lựa chọn ngẫu nhiên hoặc quay vòng giữa các máy chủ đó.

Quy trình vận hành của thuật toán Least Connection dựa trên cơ chế giám sát thời gian thực với các bước cụ thể sau:

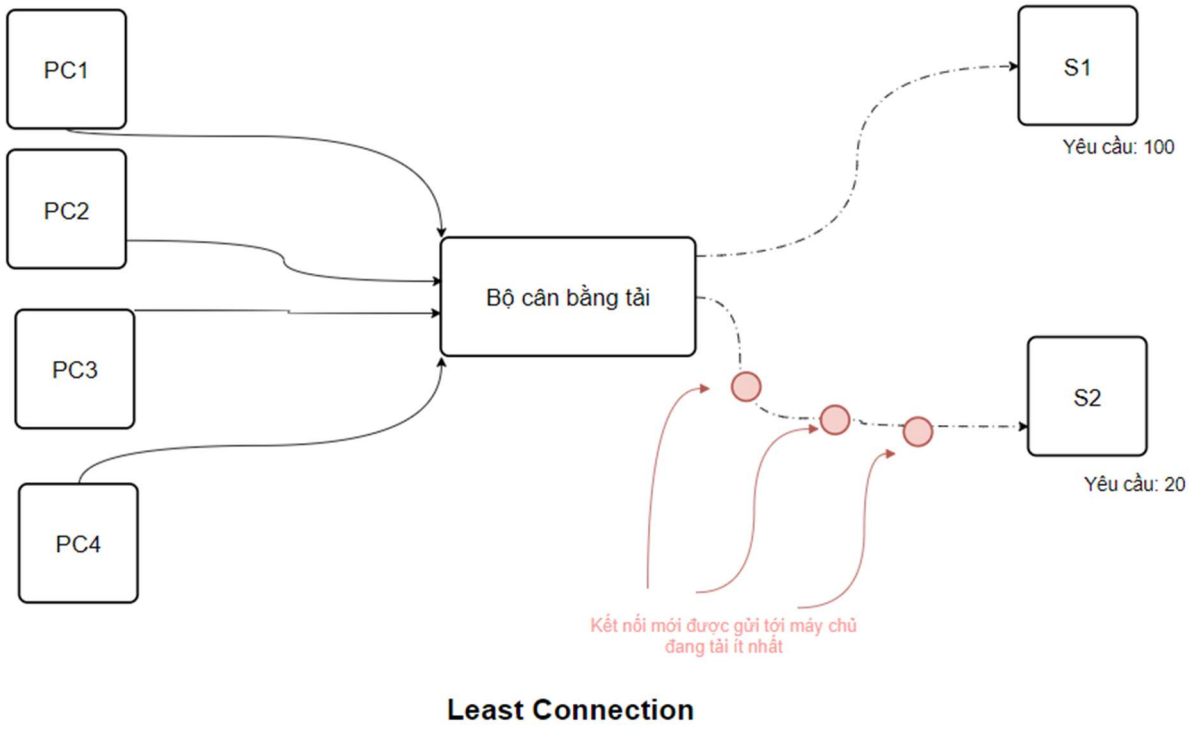
1. Duy trì trạng thái: Bộ cân bằng tải duy trì một bảng trạng thái động, ghi nhận biến C(Si) đại diện cho số lượng kết nối đang được thiết lập trên mỗi máy chủ Si.

2. Quét và lựa chọn: Khi một yêu cầu R đến hệ thống, thuật toán thực hiện quét toàn bộ danh sách trạng thái để xác định máy chủ Smin có giá trị C(Smin) thấp nhất.
3. Điều phối và cập nhật tăng: Yêu cầu được chuyển tiếp đến Smin. Đồng thời, bộ đếm kết nối của máy chủ này được cập nhật ngay lập tức:

$$C(S_{min}) \leftarrow C(S_{min}) + 1$$

4. Hoàn tất và cập nhật giảm: Khi phiên kết nối kết thúc hoặc yêu cầu được xử lý xong, tài nguyên được giải phóng và bộ đếm trạng thái được cập nhật giảm:

$$C(S_{min}) \leftarrow C(S_{min}) - 1$$



Hình ảnh 1.5 Thuật toán Least Connection

Thuật toán Least Connection được xem là tiêu chuẩn vàng cho các hệ thống cần sự ổn định trong môi trường có tải trọng không đồng nhất và thời gian xử lý kéo dài:

- Cân bằng tải thích ứng: Khác với Round Robin (vốn phân phối mù quáng theo chu kỳ), Least Connection hoạt động dựa trên trạng thái thực tế. Bằng cách điều hướng

lưu lượng tránh xa các nút đang bận rộn (có chỉ số kết nối cao), thuật toán giúp ngăn chặn hiệu quả hiện tượng quá tải cục bộ và tối đa hóa thông lượng tổng thể của cụm.

- Tự điều chỉnh trong môi trường dị thể: Trong một cụm máy chủ bao gồm các thiết bị có năng lực xử lý khác nhau (CPU/RAM chênh lệch), Least Connection thể hiện khả năng tự cân bằng tự nhiên. Các máy chủ mạnh hơn sẽ xử lý yêu cầu nhanh hơn, giải phóng kết nối sớm hơn và do đó sẽ nhận được nhiều yêu cầu mới hơn. Cơ chế này hoạt động như một hệ thống trọng số động mà không cần can thiệp cấu hình thủ công.
- Tối ưu cho các phiên kết nối: Đối với các giao thức duy trì kết nối liên tục, việc đếm số lượng kết nối phản ánh chính xác tải trọng hơn là đếm số lượng request. Thuật toán này giúp tránh tình trạng "dồn ứ" kết nối tại một máy chủ duy nhất.
- Phạm vi ứng dụng: Thuật toán là lựa chọn ưu tiên cho các kịch bản triển khai sau:
    - Dịch vụ Cơ sở dữ liệu: Nơi các truy vấn SQL hoặc NoSQL có thời gian thực thi biến thiên rất lớn và các kết nối thường được giữ lâu.
    - Giao thức thời gian thực: Các ứng dụng sử dụng WebSocket, LDAP, hoặc Email (IMAP/POP3) nơi kết nối giữa client và server được duy trì trong thời gian dài.
    - Xử lý tác vụ nặng: Các hệ thống xử lý video, mã hóa, hoặc tính toán khoa học, nơi một yêu cầu đơn lẻ có thể chiếm dụng tài nguyên trong nhiều phút hoặc nhiều giờ.

1.4.1.3. Thuật toán Least response time:

Khái niệm Thuật toán Least Response Time là phương pháp cân bằng tải hoạt động dựa trên nguyên tắc điều phối các yêu cầu mới đến máy chủ có thời gian phản hồi trung bình thấp nhất tại thời điểm xét duyệt. Thời gian phản hồi ($T_{Response}$) được xác định bởi công thức tổng quát:

$$T_{response} = T_{processing} + T_{network_latency}$$

Trong đó:

$T_{processing}$: Thời gian máy chủ xử lý yêu cầu

$T_{network_latency}$: Độ trễ truyền dẫn qua mạng

Khác với các thuật toán tĩnh như *Round Robin* hay thuật toán dựa trên trạng thái kết nối như *Least Connection*, LRT đánh giá hiệu năng dựa trên số liệu thực tế. Do đó, thuật toán này phản ánh chính xác trạng thái tải thực tế và năng lực xử lý hiện thời của từng máy chủ.

Cơ chế hoạt động

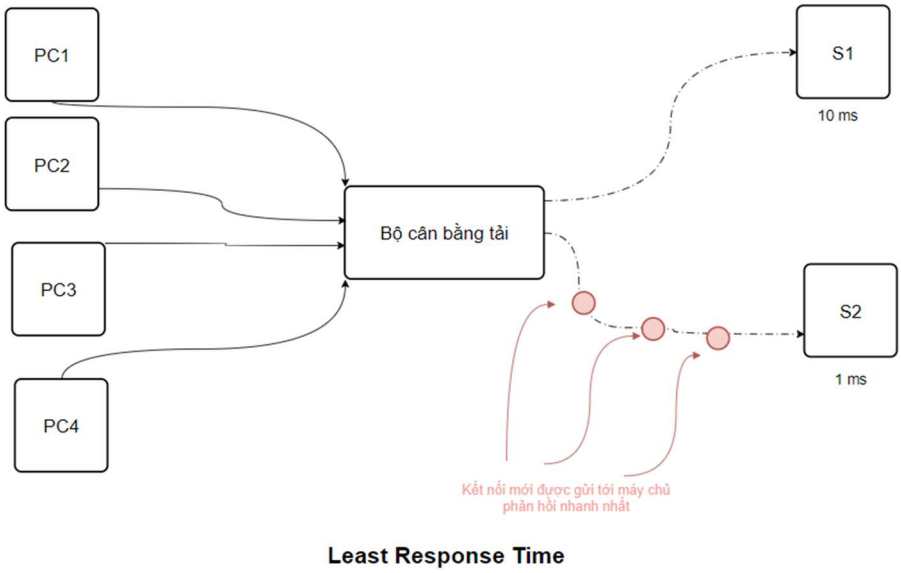
Thuật toán Least Response Time hoạt động theo quy trình sau:

Thu thập dữ liệu: Bộ cân bằng tải liên tục giám sát và thu thập các chỉ số hiệu năng từ các máy chủ trong cụm, bao gồm: thời gian phản hồi trung bình, độ trễ mạng và số lượng yêu cầu đang chờ xử lý.

Đánh giá và lựa chọn: Khi có một yêu cầu mới phát sinh, thuật toán sẽ so sánh thời gian phản hồi của tất cả các máy chủ đang khả dụng. Máy chủ có chỉ số thời gian phản hồi thấp nhất sẽ được xác định là nút tối ưu.

Điều phối: Bộ cân bằng tải chuyển tiếp yêu cầu đến máy chủ đã chọn.

Cập nhật vòng lặp: Các chỉ số hiệu năng được cập nhật liên tục theo thời gian thực để phục vụ cho các quyết định điều phối tiếp theo, đảm bảo hệ thống luôn thích nghi với sự biến động của tải.



Hình ảnh 1.6 Least Response Time

Thuật toán Least Response Time đóng vai trò thiết yếu trong các hệ sinh thái ưu tiên trải nghiệm người dùng cuối và yêu cầu khắt khe về hiệu năng thời gian thực:

- Tối ưu hóa độ trễ: Đây là đặc tính định danh của thuật toán. Bằng cách nhất quán hướng lưu lượng truy cập vào nút có tốc độ phản hồi nhanh nhất, LRT giúp giảm thiểu chỉ số "Thời gian nhận byte đầu tiên". Điều này trực tiếp cải thiện tốc độ tải trang và độ mượt mà của ứng dụng từ góc độ người dùng.
- Thích ứng với sự dị biệt về hạ tầng: Trong các cụm máy chủ hỗn hợp (nơi các thế hệ phần cứng cũ và mới cùng hoạt động song song), LRT thể hiện sự vượt trội so với các thuật toán tĩnh. Nó tự động phân bổ nhiều tải hơn cho các máy chủ có cấu hình mạnh (do tốc độ xử lý nhanh hơn) và giảm tải cho các máy chủ yếu, từ đó khai thác tối đa hiệu suất tổng thể của cụm mà không cần cấu hình trọng số thủ công.
- Độ nhạy với biến động mạng: Khác với thuật toán Least Connection chỉ tập trung vào trạng thái nội tại của máy chủ, LRT bao hàm cả yếu tố độ trễ mạng trong hàm mục tiêu. Đặc điểm này khiến nó trở nên đặc biệt hữu ích trong các hệ thống phân tán địa lý hoặc môi trường đám mây lai, nơi đường truyền mạng là một biến số không ổn định.

- Phạm vi ứng dụng: Thuật toán này là lựa chọn tối ưu cho:
    - Các nền tảng giao dịch tài chính hoặc thương mại điện tử (nơi độ trễ thấp đồng nghĩa với tỷ lệ chuyển đổi cao).
    - Các ứng dụng thời gian thực như VoIP, Gaming server.
    - Các dịch vụ phân phối nội dung (CDN) cần định tuyến người dùng đến máy chủ biên có tốc độ phản hồi tốt nhất tại thời điểm truy cập.

1.4.1.4. Thuật toán IP Hash:

Thuật toán IP Hash là phương pháp cân bằng tải mang tính xác định, trong đó máy chủ đích được lựa chọn dựa trên việc tính toán giá trị băm từ địa chỉ IP của máy khách. Khác với Round Robin hay Least Connection, mục tiêu cốt lõi của IP Hash không phải là phân phối tải ngẫu nhiên hay dựa trên hiệu năng, mà là đảm bảo tính duy trì phiên.

Về mặt toán học, giả sử tập hợp các máy chủ khả dụng là $S = \{S_0, S_1, \dots, S_{n-1}\}$ với n là tổng số máy chủ. Gọi IP_{src} là địa chỉ IP của yêu cầu đến là H(x) là hàm băm được sử dụng. Máy chủ S_k được chọn để xử lý yêu cầu được xác định bởi công thức:

$$k = H(IP_{src}) \ mod \ n$$

Trong đó:

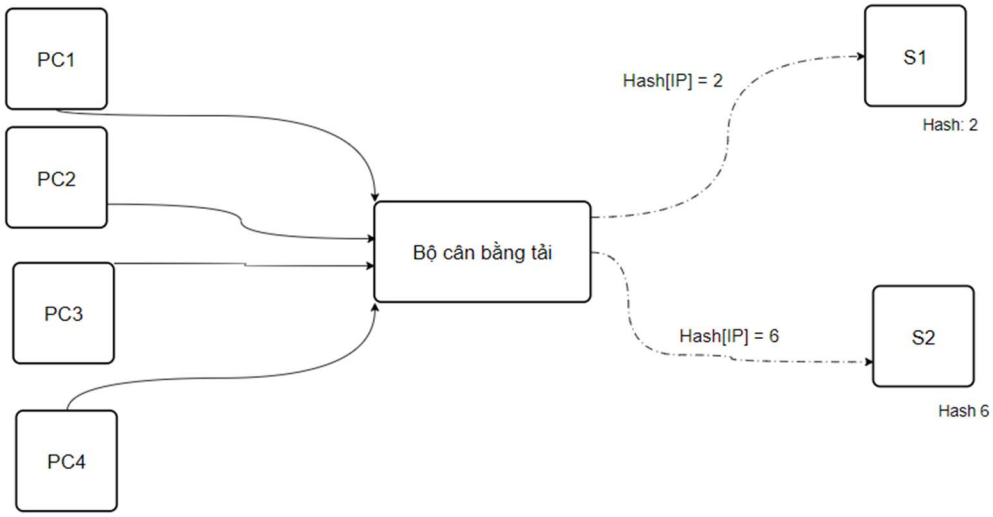
k: Chỉ số của máy chủ được chọn $(0 \leq k < n)$

$H(IP_{src})$: giá trị số nguyên được sinh ra từ địa chỉ IP nguồn sau khi đi qua hàm băm.

Mod: phép chia lấy dư, đảm bảo kết quả luôn nằm trong phạm vi chỉ số của danh sách máy chủ.

Cơ chế này đảm bảo rằng, miễn là danh sách máy chủ không đổi, một địa chỉ IP cụ thể sẽ luôn được định tuyến đến cùng một máy chủ backend duy nhất.

Quy trình vận hành



Hình ảnh 1.7 Thuật toán IP Hash

Quy trình xử lý của thuật toán IP Hash tại tầng Load Balancer diễn ra theo trình tự 4 bước sau:

1. Trích xuất định danh:

Khi một gói tin hoặc yêu cầu kết nối đến, bộ cân bằng tải trích xuất địa chỉ IP nguồn (IP_{src}) từ phần header của gói tin. Trong một số biến thể, thuật toán có thể kết hợp cả địa chỉ IP đích hoặc cổng (port) để tăng độ phân tán.

2. Tính toán giá trị băm:

Địa chỉ IP sau khi trích xuất được đưa qua một hàm băm phi mật mã tối ưu tốc độ, ví dụ như CRC32 hoặc MurmurHash. Kết quả đầu ra là một số nguyên lớn đại diện cho địa chỉ IP đó.

3. Ánh xạ máy chủ:

Giá trị số nguyên vừa tính toán sẽ được thực hiện phép chia lấy dư cho tổng số lượng máy chủ khả dụng (n). Kết quả của phép tính chính là chỉ số vị trí của máy chủ trong danh sách phân phối.

4. Thiết lập liên kết:

Yêu cầu được chuyển tiếp đến máy chủ đích đã xác định. Do tính chất tất định của hàm băm, các yêu cầu tiếp theo từ cùng một IP_{src} sẽ lặp lại đúng quy trình này và kết thúc tại cùng một máy chủ, duy trì trạng thái phiên làm việc liền mạch.

Thuật toán IP Hash đóng vai trò quan trọng trong các hệ thống yêu cầu sự ổn định về ngữ cảnh người dùng:

- Tính dính phiên: Đây là ưu điểm lớn nhất. IP Hash cực kỳ hiệu quả cho các ứng dụng Stateful (có trạng thái), nơi thông tin phiên làm việc, giỏ hàng, hoặc bộ nhớ đệm được lưu trữ cục bộ trên một máy chủ cụ thể. Nó loại bỏ nhu cầu phải đồng bộ hóa dữ liệu phiên liên tục giữa các server.
- Rủi ro phân phối lệch: Nếu một lượng lớn lưu lượng truy cập đến từ một dải IP hẹp (ví dụ: phía sau một Proxy doanh nghiệp lớn), hàm băm có thể phân phối không đều, khiến một số máy chủ bị quá tải trong khi các máy chủ khác nhàn rỗi.
- Độ nhạy với biến động hạ tầng: Khi số lượng máy chủ (n) thay đổi (do thêm mới hoặc có server bị lỗi), kết quả của phép tính mod n sẽ thay đổi hoàn toàn đối với hầu hết các IP. Điều này dẫn đến việc tái cấu trúc ánh xạ hàng loạt, làm mất dữ liệu phiên của người dùng. Để khắc phục nhược điểm này, các hệ thống hiện đại thường nâng cấp lên thuật toán Consistent Hashing (Băm nhất quán).

1.4.2. Đánh giá hiệu quả của các thuật toán

Trong vận hành thực tế, lưu lượng của trò chơi trực tuyến thường xuyên trải qua các đợt biến động mạnh, đặc biệt là trong các khung giờ cao điểm hoặc khi diễn ra sự kiện trong game. Khả năng duy trì độ trễ thấp và ngăn chặn nghẽn cổ chai tại các thời điểm này là thước đo chính để đánh giá hiệu quả của các giải pháp cân bằng tải. Dưới đây là phân tích chi tiết đối với từng nhóm kỹ thuật:

a. Nhóm thuật toán tĩnh: Round Robin, Weighted Round Robin và IP Hash

Nhóm thuật toán này hoạt động dựa trên các quy tắc định sẵn mà không giám sát trạng thái tải thực tế của máy chủ.

- Round Robin (RR) và Weighted Round Robin (WRR):
    - *Khả năng xử lý nghẽn:* Mặc dù đảm bảo phân phối công bằng về *số lượng* yêu cầu, cả RR và WRR đều bộc lộ hạn chế nghiêm trọng khi xử lý các phiên kết nối không đồng nhất về thời lượng. Trong game online, thời gian phiên chơi tuân theo phân phối Weibull, nghĩa là tồn tại những người chơi duy trì kết nối trong nhiều giờ [1]. RR có xu hướng tích lũy các phiên kết nối dài này một cách ngẫu nhiên vào một số máy chủ nhất định, gây ra tình trạng quá tải cục bộ trong khi các máy chủ khác vẫn nhàn rỗi.
    - *Về độ trễ:* Do không nhận biết được độ dài hàng đợi xử lý hiện tại, RR vẫn tiếp tục gửi yêu cầu mới vào các máy chủ đang bị nghẽn (congested), làm gia tăng độ trễ đột biến cho người chơi tại các máy chủ đó [17]. WRR tuy giải quyết được vấn đề phần cứng không đồng nhất nhưng vẫn bất lực trước sự biến thiên của tải trọng động.
- IP Hash:
    - *Rủi ro phân cực tải:* Trong các sự kiện game lớn, một lượng lớn người chơi thường tập trung tại các điểm truy cập công cộng (như Cyber Game, ký túc xá) và sử dụng chung một địa chỉ IP công cộng (NAT). Cơ chế băm địa chỉ IP của thuật toán này sẽ vô tình dồn toàn bộ nhóm người dùng này vào một máy chủ duy nhất. Điều này tạo ra các điểm nóng nghiêm trọng, gây nghẽn cổ chai cục bộ và làm gián đoạn dịch vụ, bất chấp tổng tài nguyên của hệ thống vẫn chưa được sử dụng hết [18].

b. Nhóm thuật toán động: Least Connections và Least Response Time

Nhóm thuật toán này khắc phục nhược điểm của nhóm tĩnh bằng cách thu thập số liệu thời gian thực để ra quyết định phân phối.

- Least Connections (LC):
    - *Khả năng ngăn nghẽn cổ chai:* LC thể hiện sự ưu việt trong việc "san phẳng" các đỉnh tải trọng trong giờ cao điểm. Bằng cách liên tục điều hướng người

chơi mới đến các máy chủ có ít kết nối nhất, thuật toán ngăn chặn hiệu quả việc một máy chủ đơn lẻ bị sập do quá tải số lượng kết nối [17].

- *Hạn chế:* Tuy nhiên, LC hoạt động dựa trên giả định rằng mọi kết nối tiêu tốn tài nguyên như nhau. Trong thực tế, một máy chủ có thể có ít kết nối nhưng CPU lại đang hoạt động hết công suất (100% load) do xử lý các trận chiến phức tạp. Do đó, LC vẫn có thể gửi thêm người chơi vào một máy chủ đang bị nghẽn CPU, gây ảnh hưởng đến trải nghiệm.

- Least Response Time (LRT):
  - *Khả năng duy trì độ trễ thấp:* Đây là thuật toán tiệm cận nhất với yêu cầu thời gian thực của game online. Bằng cách ưu tiên máy chủ có thời gian phản hồi thấp nhất, LRT trực tiếp tối ưu hóa chỉ số ping cho người chơi [19]. Nó có khả năng phát hiện các máy chủ đang xử lý chậm (dù ít kết nối) để điều hướng lưu lượng đi nơi khác.
  - *Vấn đề dao động:* Trong các sự kiện game có lượng truy cập tăng vọt đồng thời, LRT dễ gặp phải hiệu ứng "thundering herd”. Khi một máy chủ được xác định là "nhanh nhất", hàng loạt yêu cầu sẽ dồn về đó cùng lúc, khiến nó lập tức trở nên chậm chạp. Bộ cân bằng tải sau đó lại dồn toàn bộ lưu lượng sang máy chủ khác, gây ra sự dao động tải liên tục giữa các máy chủ và làm mất ổn định hệ thống [20].

1.4.3. So sánh mức độ đáp ứng yêu cầu của Game Online với ứng dụng web

Để xác định tính phù hợp của các kỹ thuật cân bằng tải, cần phân biệt rõ sự khác biệt căn bản giữa kiến trúc của Game Online (đặc biệt là các thể loại thời gian thực như MMORPG, FPS) và các Ứng dụng Web truyền thống. Sự khác biệt này được thể hiện qua ba khía cạnh cốt lõi: yêu cầu thời gian thực, cơ chế đồng bộ dữ liệu và khả năng mở rộng.

Bảng 1.2 Bảng tiêu chí so sánh mức độ đáp ứng yêu cầu của máy chủ

Tiêu chí so sánh	Ứng dụng Web	Game Online
Mô hình tương tác	Yêu cầu - Phản hồi: Người dùng gửi yêu cầu, server xử lý và trả kết quả. Các giao dịch thường ngắn và ngắt quãng.	Luồng liên tục: Dữ liệu được truyền tải liên tục hai chiều với tần số cao trong suốt phiên chơi kéo dài.
Yêu cầu thời gian thực	Thấp - Trung bình: Độ trễ từ 500ms - 1s là chấp nhận được. Hệ thống có thể sử dụng bộ đệm để tối ưu băng thông.	Cực cao: Độ trễ >100ms ảnh hưởng trực tiếp đến kết quả game [1]. Không chấp nhận đệm dữ liệu.
Đồng bộ dữ liệu	Phi trạng thái: Server không cần lưu trạng thái cục bộ. Dữ liệu phiên được lưu ở Database chung hoặc Cookie.	Có trạng thái: Trạng thái nhân vật (vị trí, máu) thay đổi hàng chục lần/giây và phải được lưu trên RAM của server xử lý [2].
Giao thức vận chuyển	TCP/HTTP: Ưu tiên độ tin cậy. Nếu mất gói tin, giao thức sẽ tự động truyền lại.	UDP: Ưu tiên tốc độ. Chấp nhận mất gói tin để đảm bảo dữ liệu luôn là mới nhất [3].
Khả năng mở rộng	Dễ dàng: Có thể thêm máy chủ mới và phân phối ngẫu nhiên vì các request độc lập nhau.	Phức tạp: Khó phân chia tải ngẫu nhiên vì người chơi trong cùng một phòng/khu vực phải kết nối vào cùng một máy chủ vật lý.

Bảng so sánh đặc trưng kỹ thuật giữa Game Online và ứng dụng Web

Từ các dữ liệu trên, có thể thấy rõ sự khập khiễng khi áp dụng các kỹ thuật cân bằng tải truyền thống (vốn được thiết kế cho Web) vào môi trường Game Online:

- Về thuật toán Round Robin: Phù hợp hoàn hảo cho Web App nhờ tính chất "Request ngắn". Tuy nhiên, với Game Online, do tính chất "Kết nối bền vững", Round Robin không thể kiểm soát được tải trọng tích lũy khi các phiên chơi kéo dài, dẫn đến nguy cơ quá tải cục bộ không thể dự báo.
- Về thuật toán IP Hash: Giải quyết tốt vấn đề "Có trạng thái" của Game, nhưng lại gặp hạn chế về khả năng mở rộng linh hoạt so với Web App.
- Về thuật toán Least Connection: Hiệu quả với Web App khi tải biến động. Nhưng trong Game Online, "số lượng kết nối" không phản ánh đúng "tải xử lý" (do một trận chiến game tiêu tốn CPU gấp nhiều lần một kết nối chờ), khiến thuật toán này thiếu chính xác trong việc điều phối.

Kết luận: Sự khác biệt quá lớn về yêu cầu thời gian thực và kiến trúc dữ liệu chính là lý do các mô hình truyền thống không đạt hiệu quả tối ưu cho Game Online, đặt ra nhu cầu cấp thiết phải áp dụng một mô hình toán học định lượng hơn như lý thuyết hàng đợi .

1.5. Cơ sở lý thuyết hàng đợi và khả năng ứng dụng trong mô hình hệ thống game online

Để khắc phục những hạn chế của các thuật toán cân bằng tải tĩnh trong việc xử lý lưu lượng ngẫu nhiên của game online, việc áp dụng một cơ sở toán học vững chắc là yêu cầu cấp thiết. Lý thuyết hàng đợi cung cấp khung tham chiếu định lượng để mô hình hóa hành vi chờ đợi, từ đó dự báo độ trễ và tối ưu hóa năng lực phục vụ của hệ thống.

1.5.1. Định nghĩa

Lý thuyết hàng đợi (Queueing Theory) là một nhánh của toán học ứng dụng, tập trung nghiên cứu về các dòng chờ đợi và cơ chế phục vụ. Mục tiêu chính của lý thuyết này là xây dựng các mô hình định lượng để dự báo hiệu năng hệ thống, từ đó tối ưu hóa việc phân bổ tài nguyên nhằm giảm thiểu thời gian chờ đợi hoặc chi phí vận hành.
Một hệ thống hàng đợi tiêu chuẩn luôn được cấu thành bởi ba thành phần cốt lõi:

- Dòng đến: Biểu diễn cách thức các yêu cầu (khách hàng) đi vào hệ thống. Trong ngữ cảnh Game Online, đây chính là các gói tin hoặc yêu cầu kết nối từ người chơi gửi đến máy chủ.
- Cơ chế phục vụ : Bao gồm số lượng máy chủ và năng lực xử lý của chúng. Đây là khả năng của Game Server trong việc tiếp nhận và xử lý các logic game.
- Quy tắc hàng đợi: Cách thức sắp xếp các yêu cầu chờ được phục vụ (thường là FCFS - Đến trước phục vụ trước).

1.5.2. Mối liên hệ giữa lý thuyết hàng đợi và mô hình xử lý yêu cầu của máy chủ game

Để áp dụng các công thức toán học của lý thuyết hàng đợi vào việc giải quyết bài toán thực tế của Game Online, bước tiên quyết là phải ánh xạ chính xác các khái niệm trừu tượng sang các thành phần kỹ thuật cụ thể trong kiến trúc máy chủ.

Hệ thống máy chủ Game hoạt động dựa trên cơ chế xử lý vòng lặp, trong đó các gói tin đầu vào đóng vai trò là các yêu cầu cần xử lý và các luồng của CPU đóng vai trò là các trạm phục vụ. Sự tương quan chi tiết giữa các thành phần của một hàng đợi tiêu chuẩn và các thực thể trong hệ thống Game Online được trình bày trong bảng 1.3:

Bảng 1.3 Ánh xạ các thành phần giữa mô hình Hàng đợi và Hệ thống Game Online

Thành phần Hàng đợi	Thành phần trong Game Online	Đặc điểm tương ứng
Khách hàng	Gói tin Input / Action	Các lệnh di chuyển, tấn công từ Client gửi lên Server qua giao thức UDP.
Dòng đến	Lưu lượng mạng	Có tính ngẫu nhiên và tính chu kỳ theo nhịp game.
Máy chủ	Game Logic Thread	Luồng xử lý chính của Server, chịu trách nhiệm

		tính toán vật lý, AI và cập nhật trạng thái.
Thời gian phục vụ $(1/\mu)$	Thời gian xử lý 1 Tick	Thời gian CPU cần để hoàn thành một vòng lặp game.
Hàng đợi	Socket Buffer	Bộ đệm chứa các gói tin đang chờ được xử lý khi CPU bận rộn.
Thời gian chờ (W_q)	Server Lag	Độ trễ xử lý gây ra hiện tượng phản hồi chậm trên màn hình người chơi.

1.5.3. Các tình huống thường gặp trong hệ thống game online

Dưới góc độ lý thuyết hàng đợi, các sự cố phổ biến trong game online có thể được giải thích thông qua sự biến thiên của λ và μ:

1. Tình huống "Flash Crowd" (Tăng vọt λ):

Khi diễn ra sự kiện hoặc giờ cao điểm, số lượng người chơi tăng đột biến khiến tốc độ đến λ vượt quá tổng năng lực phục vụ $(c\mu)$. Lúc này $\rho > 1$, hàng đợi tích lũy nhanh chóng, gây ra độ trễ cao cho toàn bộ người chơi.

2. Tình huống "Heavy Calculation" (Giảm μ):

Trong các game MMORPG, khi một lượng lớn người chơi tập trung tại một điểm và tung kỹ năng, khối lượng tính toán vật lý/đồ họa tăng vọt khiến thời gian xử lý một vòng lặp game kéo dài. Điều này đồng nghĩa tốc độ phục vụ μ giảm xuống. Ngay cả khi lượng người chơi λ không đổi, việc μ giảm cũng làm ρ tăng cao, dẫn đến tình trạng "Lag giật cục" dù đường truyền mạng vẫn tốt.

3. Tình huống mất gói tin:

Khi hàng đợi vật lý đầy do quá tải, các gói tin UDP mới đến sẽ bị từ chối phục vụ. Trong game, điều này biểu hiện qua việc nhân vật bị khựng lại hoặc dịch chuyển tức thời do mất dữ liệu đồng bộ.

1.5.4. Quy trình tích hợp hàng đợi vào cơ chế cân bằng tải

Để ứng dụng lý thuyết hàng đợi vào thực tế, quy trình cân bằng tải cần chuyển từ cơ chế "đếm kết nối" sang cơ chế "dự báo dựa trên mô hình" theo 3 bước:

- Bước 1: Giám sát và Thu thập tham số (λ, μ):

Load Balancer liên tục đo đạc tốc độ gói tin đến (λ) và thời gian phản hồi trung bình của từng Game Server để ước lượng năng lực xử lý (μ) hiện tại.

- Bước 2: Tính toán chỉ số hiệu năng (W_q):

Sử dụng các công thức hàng đợi (như mô hình $M/M/1$ hoặc $M/G/1$) để tính toán thời gian chờ dự kiến (W_q) cho mỗi máy chủ. Chỉ số này phản ánh chính xác "sức khỏe" của server hơn là số lượng CPU hay RAM.

- Bước 3: Ra quyết định điều phối:

Thuật toán sẽ so sánh W_q của các máy chủ. Người chơi mới sẽ được điều hướng đến máy chủ có W_q thấp nhất (tức là máy chủ có khả năng phản hồi nhanh nhất theo dự báo), thay vì máy chủ có ít kết nối nhất.

Quy trình tích hợp này giúp hệ thống chủ động phát hiện và ngăn chặn tắc nghẽn trước khi nó thực sự xảy ra, đảm bảo duy trì độ trễ thấp và ổn định – yếu tố sống còn của game online.

1.6. Kết luận chương 1

Tổng kết lại, Chương 1 đã hệ thống hóa cơ sở lý luận về bối cảnh phát triển và đặc tính kỹ thuật của hệ thống trò chơi trực tuyến nhiều người chơi. Các phân tích cho thấy lưu lượng mạng của game online mang tính chất ngẫu nhiên với đặc thù là các gói tin kích thước nhỏ và tính chu kỳ của các đợt dữ liệu. Trong khi đó, kiến trúc Client-Server, dù đảm

bảo tính nhất quán dữ liệu, lại dễ dàng trở thành điểm nghẽn cổ chai khi số lượng yêu cầu đến vượt quá khả năng phục vụ của máy chủ, dẫn đến hình thành các hàng đợi xử lý kéo dài.

Bên cạnh đó, việc khảo sát các thuật toán cân bằng tải truyền thống (như Round Robin, Least Connection) cho thấy các phương pháp này thường vận hành dựa trên các quy tắc tĩnh hoặc bán động mà thiếu đi một cơ sở toán học vững chắc để dự báo và kiểm soát tắc nghẽn. Chúng chưa giải quyết triệt để bài toán tối ưu hóa thời gian chờ trong hàng đợi – yếu tố then chốt ảnh hưởng đến độ trễ và trải nghiệm người chơi.

Từ những hạn chế này, có thể khẳng định rằng để giải quyết bài toán hiệu năng cho game online một cách triệt để, cần có một phương pháp tiếp cận định lượng chính xác hơn. Việc áp dụng lý thuyết hàng đợi để mô hình hóa hành vi đến của người chơi và năng lực phục vụ của hệ thống là hướng đi cần thiết, giúp tính toán và dự báo chính xác các chỉ số hiệu năng mạng, từ đó đưa ra chiến lược phân phối tải tối ưu.

Trên cơ sở những hạn chế của các giải pháp truyền thống đã phân tích ở Chương 1, Chương 2 sẽ đi sâu vào việc đề xuất mô hình cân bằng tải áp dụng lý thuyết hàng đợi. Cách tiếp cận này cung cấp công cụ toán học định lượng để mô hình hóa hành vi dòng dữ liệu, từ đó xây dựng cơ chế điều phối tối ưu nhằm giải quyết triệt để bài toán độ trễ và tắc nghẽn trong hệ thống máy chủ game.

CHƯƠNG 2. Đề xuất và phân tích mô hình cân bằng tải cho game online

Tiếp nối các phân tích nền tảng tại Chương 1 về những hạn chế của các kỹ thuật cân bằng tải truyền thống trước đặc thù lưu lượng trò chơi trực tuyến, Chương 2 sẽ tập trung vào việc xây dựng và đề xuất một giải pháp mô hình hóa tối ưu hơn. Trọng tâm của chương là việc ứng dụng cơ sở toán học của lý thuyết hàng đợi để thiết kế một cơ chế cân bằng tải có khả năng dự báo tắc nghẽn và điều phối tài nguyên động.

Thông qua việc xác định các yêu cầu kỹ thuật khắt khe về thời gian thực và tính nhất quán dữ liệu, chương này sẽ trình bày quy trình thiết kế hệ thống từ kiến trúc tổng thể đến cơ chế vận hành, nhằm chứng minh tính khả thi của mô hình trong việc giảm thiểu độ trễ và nâng cao chất lượng trải nghiệm người dùng.

2.1. Phân tích yêu cầu

Việc thiết kế một hệ thống cân bằng tải cho trò chơi trực tuyến không chỉ dừng lại ở bài toán phân phối lưu lượng mạng thông thường, mà còn phải đáp ứng các ràng buộc kỹ thuật khắt khe đặc thù của hệ thống thời gian thực. Dưới đây là phân tích chi tiết các yêu cầu phi chức năng và những thách thức thực tiễn mà mô hình đề xuất cần giải quyết.

2.1.1. Các yêu cầu về hiệu năng và đặc tả kỹ thuật

Khác với các ứng dụng web truyền thống nơi độ trễ có thể được giảm thiểu thông qua các kỹ thuật đệm, các trò chơi trực tuyến, đặc biệt là các thể loại hành động thời gian thực như MMORPG hay FPS, đặt ra những yêu cầu nghiêm ngặt về hiệu năng hệ thống. Để đảm bảo chất lượng trải nghiệm người dùng, hệ thống cân bằng tải cần thỏa mãn các tiêu chí sau:

a. Yêu cầu về độ trễ thấp và độ ổn định

- Ngưỡng độ trễ: Hệ thống cân bằng tải phải đảm bảo tổng thời gian xử lý và truyền tải nằm trong giới hạn cho phép để duy trì tính tương tác thời gian thực. Cụ thể, đối với thể loại game bắn súng góc nhìn thứ nhất, ngưỡng độ trễ chấp nhận được thường là dưới 100ms; đối với game nhập vai trực tuyến, con số này là dưới 250ms. Bất kỳ

sự chậm trễ nào vượt quá các ngưỡng này đều tác động tiêu cực trực tiếp đến khả năng điều khiển và trải nghiệm của người chơi.

- Độ biến thiên trễ: Bên cạnh độ trễ thấp, sự ổn định của độ trễ là yếu tố then chốt. Mô hình đề xuất cần hướng tới việc giảm thiểu phương sai của thời gian chờ trong hàng đợi ($Var[W_q]$). Thực tế cho thấy, việc độ trễ dao động mạnh (ví dụ biến thiên thất thường giữa 20ms và 150ms) gây khó khăn lớn cho người chơi và các thuật toán dự đoán chuyển động phía máy khách hơn là một mức trễ cao nhưng ổn định.

b. Yêu cầu về năng lực xử lý gói tin

- Tốc độ xử lý gói tin: Lưu lượng game có đặc thù là kích thước gói tin rất nhỏ (trung bình khoảng 40 bytes) nhưng tần số gửi lại rất cao (từ 20Hz đến 60Hz). Do đó, bộ cân bằng tải cần ưu tiên tối ưu hóa khả năng chuyển mạch hàng triệu gói tin mỗi giây thay vì chỉ tập trung vào băng thông tổng thể.
- Tối ưu hóa giao thức UDP: Mô hình cần hỗ trợ định tuyến hiệu quả cho giao thức UDP phi kết nối, giảm thiểu chi phí kiểm tra tiêu đề để tránh tạo ra các nút thắt cổ chai tại tầng cân bằng tải.

c. Yêu cầu về tính nhất quán phiên

- Do đặc tính "có trạng thái" của game online, dữ liệu phiên làm việc của người chơi (vị trí, trạng thái nhân vật, vật phẩm) được lưu trữ trên bộ nhớ RAM của máy chủ xử lý cụ thể. Yêu cầu bắt buộc là một khi người chơi đã kết nối vào một máy chủ, mọi gói tin tiếp theo trong phiên đó phải được định tuyến chính xác về cùng máy chủ đó. Việc điều hướng sai lệch sẽ dẫn đến mất đồng bộ dữ liệu và ngắt kết nối.

d. Yêu cầu về khả năng mở rộng

Hệ thống cần hỗ trợ khả năng mở rộng theo chiều ngang, cho phép bổ sung máy chủ mới vào cụm một cách "trong suốt" đối với người dùng cuối. Tính năng này đặc biệt quan trọng để ứng phó với các sự kiện "Flash Crowd", khi lượng người chơi tăng vọt đột biến trong thời gian ngắn.

2.1.2. Các vấn đề và thách thức khi áp dụng kỹ thuật cân bằng tải truyền thống

Mặc dù các kỹ thuật cân bằng tải truyền thống (như Round Robin, Least Connections, IP Hash) đã chứng minh được hiệu quả vượt trội trong hạ tầng Web và các ứng dụng phi trạng thái, việc áp dụng nguyên trạng chúng vào hệ thống Game Online lại bộc lộ những hạn chế mang tính hệ thống. Thực tế vận hành đã bộc lộ các vấn đề hạn chế nghiêm trọng sau đây.

a. Vấn đề "Mù" trạng thái ứng dụng

Hầu hết các bộ cân bằng tải truyền thống hoạt động ở Lớp 4 (Transport Layer) của mô hình OSI. Chúng ra quyết định điều phối dựa trên các thông số mạng cơ bản như địa chỉ IP, cổng (Port) và trạng thái kết nối TCP/UDP mà hoàn toàn "mù" trước ngữ cảnh logic bên trong của trò chơi

- Sai lệch định lượng tải: Đối với Web Server, một kết nối HTTP thường tương đương với một đơn vị tải nhất định. Tuy nhiên, trong Game Online, "số lượng kết nối" không tỷ lệ thuận với "tải trọng xử lý".
    - *Ví dụ minh họa:* Một máy chủ A đang duy trì 1.000 kết nối của những người chơi đang đứng yên tại sảnh chờ (Lobby) sẽ tiêu tốn ít tài nguyên CPU hơn rất nhiều so với máy chủ B chỉ có 100 kết nối nhưng đang xử lý một trận giao tranh tổng (Combat) với hàng loạt tính toán vật lý và va chạm phức tạp.
    - *Hoặc ví dụ minh họa:* Một máy chủ game có thể duy trì số lượng kết nối thấp nhưng CPU lại hoạt động ở mức tải tối đa (100%) do phải xử lý các tác vụ tính toán vật lý phức tạp cho một trận chiến trong game. Nếu thuật toán cân bằng tải (như Least Connection) tiếp tục điều phối người chơi mới vào máy chủ này chỉ vì chỉ số kết nối thấp, sẽ dẫn đến tình trạng quá tải và sập máy chủ.
- Hệ quả: Thuật toán Least Connections sẽ tiếp tục điều hướng người chơi mới vào máy chủ B (vì thấy 100 < 1000), vô tình đẩy máy chủ này vượt ngưỡng chịu đựng và gây sập nguồn (Crash), trong khi máy chủ A vẫn dư thừa tài nguyên.

b. Hiện tượng dao động tải

- Đối với các thuật toán động có phản ứng nhanh như Least Response Time, hệ thống dễ rơi vào trạng thái mất ổn định do hiệu ứng đám đông. Khi một máy chủ được xác định là "rảnh" hoặc phản hồi nhanh nhất, thuật toán sẽ dồn toàn bộ lưu lượng mới vào máy chủ đó.
- Hệ quả: Sự gia tăng đột ngột này khiến hàng đợi xử lý của máy chủ bị đầy tức thời, làm tăng độ trễ vọt lên cao. Thuật toán lại ngay lập tức chuyển hướng dòng chảy sang một máy chủ khác, tạo ra hiện tượng dao động tải liên tục giữa các máy chủ. Điều này làm tăng thời gian chờ của toàn bộ hệ thống, làm trải nghiệm game bị "giật cục", khó chịu hơn cả việc độ trễ cao nhưng ổn định.

c. Khó khăn trong dự báo tắc nghẽn

- Cơ chế phản ứng thụ động: Các kỹ thuật truyền thống hoạt động theo cơ chế phản ứng. Chúng chỉ bắt đầu điều tiết hoặc chặn kết nối khi các chỉ số giám sát (như CPU, Ping) đã chạm ngưỡng báo động.
- Độ trễ xử lý: Trong môi trường thời gian thực, khoảng thời gian từ lúc máy chủ bắt đầu quá tải đến lúc bộ cân bằng tải phát hiện ra thường là quá muộn. Hàng đợi vật lý đã bị tràn, dẫn đến mất gói tin hàng loạt. Hệ thống thiếu một cơ chế toán học định lượng (như Lý thuyết hàng đợi) để dự báo xác suất tắc nghẽn và chủ động điều tiết dòng vào ngay từ khi tải trọng mới chỉ chớm tăng.

d. Thách thức về duy trì kết nối bền vững

- Khác với giao thức HTTP nơi các yêu cầu diễn ra ngắn và ngắt quãng, một phiên kết nối Game Online thường kéo dài từ vài chục phút đến hàng giờ và mang tính chất "Có trạng thái". Dữ liệu nhân vật, vị trí và vật phẩm được lưu trữ trực tiếp trên RAM của máy chủ xử lý.
- Khi một máy chủ bị quá tải, việc cân bằng lại bằng cách chuyển người chơi sang máy chủ khác là cực kỳ khó khăn và tốn kém tài nguyên. Việc di chuyển đòi hỏi phải tuần tự hóa toàn bộ trạng thái bộ nhớ và đồng bộ sang máy mới, gây ra hiện tượng "đứng hình" trong game

- Hệ quả của thuật toán tĩnh: Các thuật toán như Round Robin hoặc IP Hash phân phối kết nối ngay từ đầu mà không dự báo trước độ dài phiên chơi. Nếu thuật toán vô tình dồn nhiều người chơi lâu vào cùng một máy chủ, máy chủ đó sẽ bị quá tải vĩnh viễn trong suốt thời gian diễn ra sự kiện, trong khi bộ cân bằng tải không thể can thiệp để chia sẻ bớt gánh nặng.

Tóm lại, việc áp dụng nguyên trạng các kỹ thuật cân bằng tải truyền thống vào môi trường Game Online tồn tại những lỗ hổng mang tính hệ thống. Sự thiếu hụt khả năng nhận thức ngữ nghĩa ứng dụng và cơ chế vận hành thụ động khiến các giải pháp này bất lực trong việc duy trì tính ổn định trước các biến động đặc thù như sự kiện bùng nổ lưu lượng hay sự không đồng nhất của phần cứng. Thực tế này đặt ra yêu cầu cấp thiết phải chuyển dịch sang một mô hình quản lý tài nguyên thông minh hơn, có khả năng định lượng chính xác năng lực xử lý và dự báo tắc nghẽn dựa trên cơ sở toán học vững chắc thay vì các quy tắc kinh nghiệm đơn thuần.

2.2. Lựa chọn kỹ thuật cân bằng tải phù hợp

Dựa trên phân tích yêu cầu tại mục 2.1, việc lựa chọn kỹ thuật cân bằng tải cho hệ thống game online không chỉ dừng lại ở việc so sánh các tính năng sẵn có, mà đòi hỏi một phương pháp tiếp cận định lượng nhằm giải quyết bài toán tối ưu hóa thời gian chờ.

2.2.1. Tiêu chí đánh giá và sàng lọc giải pháp

Để đảm bảo tính khách quan và khoa học, quá trình lựa chọn được thực hiện dựa trên bộ bốn tiêu chí trọng yếu sau:

1. Khả năng nhận thức trạng thái: Giải pháp phải có khả năng giám sát các chỉ số thời gian thực của máy chủ (CPU, RAM, độ dài hàng đợi hiện tại) thay vì chỉ dựa vào các quy tắc tĩnh.
2. Tính thích ứng với môi trường dị thể: Thuật toán phải hoạt động hiệu quả trong cụm máy chủ bao gồm các nút có năng lực xử lý khác nhau (máy mạnh/máy yếu), tự động điều phối tải trọng tương ứng với khả năng phục vụ (μ) của từng nút.

3. Cơ chế dự báo tắc nghẽn: Thay vì phản ứng thụ động khi sự cố đã xảy ra, hệ thống cần khả năng dự báo xu hướng tăng trưởng của hàng đợi để thực hiện điều tiết sớm.
4. Tối thiểu hóa độ trễ kỳ vọng ($Min(E[W])$): Hàm mục tiêu của thuật toán phải là giảm thiểu thời gian phản hồi trung bình cho người chơi, phù hợp với định luật Little trong lý thuyết hàng đợi.

2.2.2. Phân tích loại trừ các kỹ thuật truyền thống

Dựa trên các tiêu chí trên, các kỹ thuật truyền thống bộc lộ những khiếm khuyết không thể khắc phục trong môi trường game online:

- Round Robin (RR): Bị loại trừ do vi phạm tiêu chí số 1 và 2. RR phân phối tải một cách "mù quáng" theo chu kỳ mà không quan tâm đến trạng thái tải hiện tại hay năng lực xử lý của máy chủ. Điều này dẫn đến rủi ro cao gây quá tải cục bộ tại các máy chủ có cấu hình thấp hoặc đang xử lý các tác vụ nặng, như đã được chứng minh trong các nghiên cứu thực nghiệm.
- Least Connections (LC): Mặc dù thỏa mãn tiêu chí số 1, LC bị loại trừ do hạn chế ở tiêu chí số 2. Thuật toán này giả định rằng mọi kết nối đều tiêu tốn tài nguyên như nhau. Trong thực tế game online, một kết nối "treo" tiêu tốn tài nguyên không đáng kể so với một kết nối đang tham gia giao tranh. Do đó, chỉ số "số lượng kết nối" không phản ánh chính xác áp lực lên hàng đợi xử lý của CPU.
- IP Hash: Rủi ro về phân cực tải, trong các sự kiện game, một lượng lớn người chơi thường tập trung tại các điểm Cyber Game, hoặc các nơi có sử dụng chung một địa chỉ IP công cộng, IP hash sẽ dồn toàn bộ nhóm người chơi này vào duy nhất một máy chủ, gây quá tải cục bộ trong khi các máy khác nhàn rỗi. Khi cần thêm hoặc bớt máy chủ để bảo trì, cấu trúc băm thay đổi dẫn đến việc bị ngắt kết nối của hàng loạt người chơi hiện hữu vi phạm yêu cầu về trải nghiệm liên tục của Game Online.
- Least Response Time: Khi một máy chủ vừa được giải phóng và trở nên nhanh nhất, thuật toán sẽ dồn rất nhiều lưu lượng mới vào đó, điều này khiến máy chủ này lập tức bị quá tải và chậm lại, thuật toán chuyển hướng đột ngột sang máy khác. Sự dạo

động này sẽ gây ra độ trễ biến thiến cao và làm trải nghiệm game bị giật, khó chịu hơn cả việc độ trễ cao nhưng ổn định.

2.2.3. Đề xuất giải pháp: Mô hình cân bằng tải dự báo dựa trên hàng đợi

Dựa trên việc phân tích loại trừ các kỹ thuật truyền thống, đề tài đề xuất giải pháp cốt lõi là xây dựng một thuật toán cân bằng tải có khả năng "nhận thức tắc nghẽn". Để hiện thực hóa giải pháp này, cần thiết lập một cơ sở toán học vững chắc nhằm định lượng mối quan hệ giữa tải trọng đầu vào và độ trễ xử lý. Mô hình toán học được lựa chọn để biểu diễn hoạt động của một máy chủ Game Online là mô hình hàng đợi M/M/1 (theo ký hiệu Kendall). Việc lựa chọn mô hình này dựa trên các đặc tính ngẫu nhiên của dòng gói tin đến và thời gian xử lý, phù hợp với đặc thù biến thiên của môi trường mạng. Các định nghĩa toán học và công thức nền tảng được thiết lập như sau:

a. Ký hiệu Kendall

Để chuẩn hóa mô hình, ký hiệu Kendall $(A/B/c)$ được sử dụng rộng rãi:

- A: Phân phối xác suất của dòng đến (thường là M - Markovian/Poisson).
- B: Phân phối xác suất của thời gian phục vụ (thường là M - Mũ hoặc G - Tổng quát).
- c: Số lượng máy chủ phục vụ song song.

b. Các tham số và công thức nền tảng

Các tham số đầu vào quan trọng bao gồm:

- λ: Tốc độ đến trung bình (số yêu cầu/giây).
- μ: Tốc độ phục vụ trung bình của một máy chủ (số yêu cầu/giây).
- c: Số lượng máy chủ.

Dựa trên trạng thái ổn định, các chỉ số hiệu năng được tính toán như sau:

c. Hệ số sử dụng (Utilization Factor - ρ): Đo lường mức độ bận rộn của hệ thống.

$$\rho = \frac{\lambda}{c\mu}$$

Điều kiện ổn định: Hệ thống chỉ hoạt động ổn định khi $\rho < 1$. Nếu $\rho \to 1$, độ dài hàng đợi sẽ tiến tới vô cùng.

d. Định luật Little: Đây là định lý quan trọng nhất thiết lập mối quan hệ tuyến tính giữa số lượng và thời gian [22]:

$$L = \lambda W$$

Trong đó:

- L: Số lượng yêu cầu trung bình trong hệ thống.
- W: Thời gian lưu lại trung bình trong hệ thống.

e.Thời gian chờ trung bình trong hàng đợi:

$$W = W_q + \frac{1}{\mu}$$

(Thời gian đáp ứng = Thời gian chờ + Thời gian xử lý)

Từ các công thức nền tảng nêu trên, đặc biệt là công thức tính thời gian chờ trung bình, ta có thể xây dựng được hàm mục tiêu cho thuật toán cân bằng tải đề xuất (QPLB). Thay vì chỉ so sánh số lượng kết nối thô, thuật toán sẽ tính toán giá trị "Chi phí tắc nghẽn" cho từng máy chủ dựa trên năng lực xử lý thực tế $\boldsymbol{\mu}$ và tải trọng hiện tại $\boldsymbol{\lambda}$. Cơ chế này cho phép hệ thống dự báo được sự gia tăng đột biến của độ trễ khi hệ thống tiệm cận ngưỡng bão hòa, từ đó đưa ra quyết định điều phối chính xác hơn.

Từ những phân tích trên, đề tài đề xuất áp dụng mô hình cân bằng tải dự báo dựa trên hàng đợi. Đây là một phương pháp tiếp cận lai, kết hợp giữa cơ chế giám sát thời gian thực và mô hình toán học M/M/1 để ra quyết định.

a. Cơ sở toán học của mô hình đề xuất

Xem xét hệ thống bao gồm n máy chủ game $\{S_1, S_2, \dots, S_n\}$. Mỗi máy chủ S_i được mô hình hóa như một hàng đợi M/M/1 độc lập với các tham số:

- λ_i : Tốc độ đến hiện tại (số yêu cầu hoặc kết nối đang được xử lý).
- μ_i : Tốc độ phục vụ trung bình (năng lực xử lý của máy chủ, đo bằng số yêu cầu/giây).

Theo lý thuyết hàng đợi, thời gian lưu lại trung bình (Mean Response Time) của một yêu cầu tại máy chủ S_i được xác định bởi công thức:

$$E[W_i] = \frac{1}{\mu_i - \lambda_i}$$

Điều kiện ổn định: $\rho_i = \frac{\lambda_i}{\mu_i} < 1$. Nếu λ_i tiến gần tới μ_i , thời gian chờ sẽ tăng theo hàm mũ và tiến tới vô cùng (tắc nghẽn hoàn toàn).

b. Hàm chi phí và thuật toán điều phối

Mô hình QPLB định nghĩa một "Hàm chi phí tắc nghẽn" C_i cho mỗi máy chủ tại thời điểm t , đại diện cho độ trễ kỳ vọng mà một người chơi mới sẽ phải gánh chịu nếu kết nối vào máy chủ đó:

Trường hợp 1: Khi tốc độ đến nhỏ hơn khả năng xử lý ($\lambda_i(t) < \mu_i$), hàm chi phí được tính bằng công thức:

$$C_i(t) = \frac{1}{\mu_i - \lambda_i(t)}$$

Trường hợp 2: Khi tốc độ đến vượt quá hoặc bằng khả năng xử lý ($\lambda_i(t) \geq \mu_i$), hàm chi phí tiến tới vô cùng:

$$C_i(t) = \infty$$

Cách viết này vừa rõ ràng, vừa dễ trình bày mà vẫn đảm bảo tính chính xác của thuật toán QPLB bạn đang đề xuất.

Quy tắc ra quyết định:

Khi có một yêu cầu kết nối mới từ người chơi P_{new}, bộ cân bằng tải sẽ chọn máy chủ mục tiêu S_{opt} sao cho hàm chi phí đạt giá trị cực tiểu:

$$S_{opt} = \arg\min_{i=1}^{n} \{ C_i(t) \}$$

c. Ưu điểm của mô hình QPLB so với truyền thống

1. Nhận thức năng lực: Thông qua tham số μ_i, thuật toán phân biệt được máy chủ mạnh và máy chủ yếu. Máy chủ mạnh (có μ lớn) sẽ có C_i thấp hơn ngay cả khi đang chịu tải cao, do đó sẽ được ưu tiên nhận thêm kết nối. Điều này giải quyết triệt để vấn đề của thuật toán Least Connections.

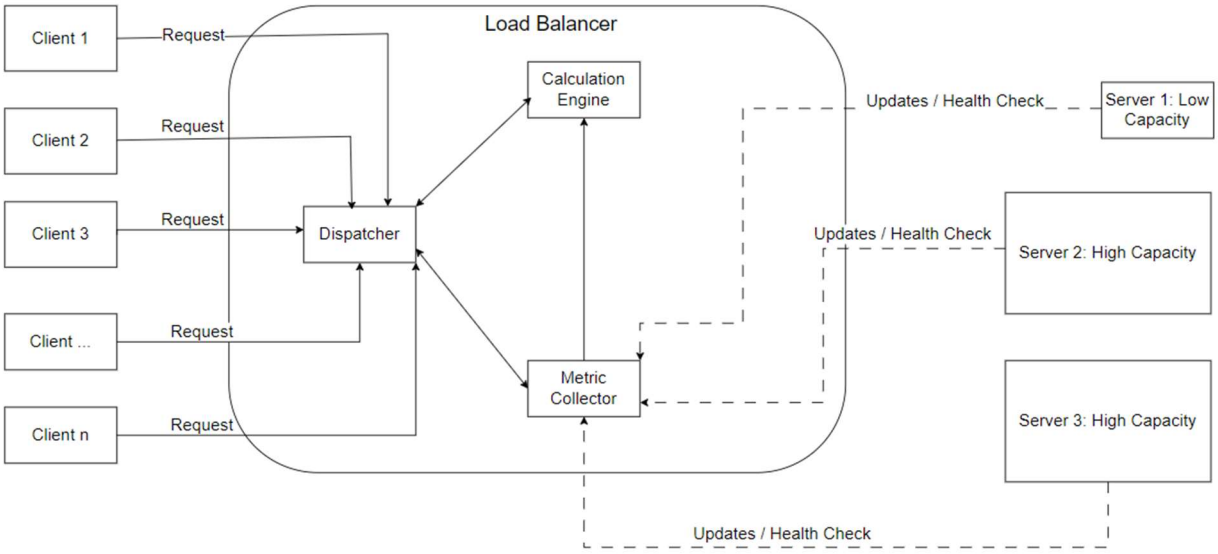
2. Bảo vệ quá tải: Khi λ_i tiệm cận μ_i , giá trị C_i tăng vọt rất nhanh (do mẫu số tiến về 0). Cơ chế này tạo ra một "rào cản mềm", tự động ngăn chặn việc gửi thêm yêu cầu vào các máy chủ đang có nguy cơ bão hòa, giúp duy trì độ ổn định hệ thống cao hơn so với Round Robin.
3. Cần lưu ý rằng, trong môi trường thực tế, năng lực phục vụ (μ) không phải là một hằng số bất biến mà chịu ảnh hưởng bởi trạng thái tài nguyên tức thời (CPU/RAM). Do đó, mô hình đề xuất được thiết kế để hỗ trợ tính toán μ động, cho phép thuật toán thích nghi với sự suy giảm hiệu năng khi máy chủ chịu tải cao. Cơ chế này sẽ được kiểm chứng chi tiết trong kịch bản thực nghiệm tại Chương 3.

2.3. Thiết kế mô hình đề xuất

Dựa trên cơ sở lý thuyết hàng đợi và các yêu cầu thực tiễn đã phân tích, đề tài đề xuất kiến trúc hệ thống cân bằng tải mới có tên gọi. Mô hình này được thiết kế để khắc phục nhược điểm "mù trạng thái" của các bộ cân bằng tải truyền thống, chuyển dịch từ cơ chế phân phối ngẫu nhiên sang phân phối dựa trên dự báo năng lực xử lý.

2.3.1. Sơ đồ kiến trúc tổng thể

Kiến trúc hệ thống được tổ chức theo mô hình tập trung, trong đó bộ cân bằng tải đóng vai trò là lớp trung gian thông minh điều phối luồng kết nối giữa người dùng và tài nguyên máy chủ.



Hình ảnh 2.1 Sơ đồ kiến trúc luồng dữ liệu của mô hình QPLB

Như thể hiện trong Hình 2.1, hệ thống được phân tách thành ba tầng chức năng rõ rệt:

1. Tầng khách: Nơi phát sinh các yêu cầu kết nối từ thiết bị người chơi.
2. Tầng cân bằng tải: Trung tâm xử lý, nơi diễn ra các hoạt động thu thập, tính toán và ra quyết định điều phối.
3. Tầng xử lý: Cụm máy chủ game không đồng nhất, bao gồm các máy chủ có năng lực xử lý khác nhau.

2.3.2. Các thành phần chính và chức năng

Bộ cân bằng tải QPLB không hoạt động như một hộp đen nguyên khối mà được module hóa thành ba thành phần tương tác chặt chẽ:

a. Bộ điều phối

Đây là cổng giao tiếp duy nhất của hệ thống đối với phía Client.

- Chức năng: Tiếp nhận yêu cầu kết nối ban đầu từ người chơi. Thay vì chuyển tiếp toàn bộ lưu lượng dữ liệu, dispatcher hoạt động theo cơ chế chỉ định. Nó tham vấn Bộ máy tính toán để xác định máy chủ tối ưu nhất, sau đó trả về địa chỉ IP và cổng của máy chủ đó cho Client.

- Vai trò: Giảm tải cho nút trung tâm. Bằng cách để Client kết nối trực tiếp P2P với Game Server sau khi được chỉ định, Dispatcher tránh được việc trở thành nút thắt cổ chai về băng thông.

b. Bộ thu thập dữ liệu

Đây là thành phần giám sát thời gian thực, đóng vai trò như "giác quan" của hệ thống.

- Chức năng: Duy trì kết nối liên tục với các máy chủ game thông qua cơ chế cập nhật định kỳ.
- Dữ liệu thu thập:
    - *Tải trọng hiện tại (λ):* Số lượng người chơi hoặc tác vụ đang được xử lý trên mỗi máy chủ.
    - *Năng lực xử lý (μ):* Thông số cấu hình cứng hoặc chỉ số hiệu năng thời gian thực (CPU/RAM) của máy chủ.
- Ý nghĩa: Thành phần này đảm bảo hệ thống luôn có cái nhìn toàn cảnh và chính xác nhất về sức khỏe của cụm máy chủ, phân biệt rõ ràng giữa máy chủ "Mạnh" và máy chủ "Yếu".

c. Bộ máy tính toán

Đây là "bộ não" của hệ thống, nơi áp dụng lý thuyết hàng đợi để ra quyết định.

- Chức năng: Tiếp nhận dữ liệu thô từ bộ thu thập và áp dụng thuật toán M/M/1 để tính toán "chi phí tắc nghẽn" cho từng máy chủ.
- Nguyên lý hoạt động: Thay vì chỉ đếm số lượng kết nối như các thuật toán cũ, bộ máy này tính toán thời gian chờ kỳ vọng. Một máy chủ cấu hình thấp dù có ít kết nối vẫn có thể sinh ra thời gian chờ cao, và ngược lại. Kết quả tính toán sẽ được gửi lại cho Dispatcher để thực hiện điều phối.

2.4. Quy trình xử lý yêu cầu trong mô hình

Quy trình vận hành của hệ thống QPLB không chỉ dừng lại ở việc phân phối kết nối mà là một chu trình kiểm soát khép kín. Quy trình này đảm bảo rằng mọi quyết định định tuyến đều dựa trên trạng thái thực tế và dự báo xu hướng tải của toàn bộ cụm máy chủ.

2.4.1. Luồng xử lý yêu cầu kết nối

Quy trình từ khi người dùng khởi tạo yêu cầu đến khi thiết lập thành công phiên chơi được chia thành 5 bước tuần tự sau:

Bước 1: Khởi tạo yêu cầu

- Máy trạm gửi một yêu cầu HTTP/RESTful đến cổng giao tiếp công khai của bộ điều phối. Yêu cầu này chứa các thông tin định danh cơ bản của người dùng nhưng chưa thiết lập kết nối thời gian thực.

Bước 2: Tham vấn trạng thái

- Ngay khi tiếp nhận yêu cầu, bộ điều phối không ra quyết định ngẫu nhiên mà gửi tín hiệu tham vấn đến bộ máy tính toán.
- Tại đây, bộ máy tính toán truy xuất dữ liệu trạng thái mới nhất từ bộ thu thập dữ liệu, bao gồm: số lượng kết nối hiện tại (λ_i) và năng lực xử lý định danh (μ_i) của từng máy chủ S_i .

Bước 3: Tính toán và ra quyết định

- Dựa trên dữ liệu thu thập được, hệ thống áp dụng thuật toán M/M/1 để tính toán "Chi phí tắc nghẽn" ($Cost_i$) cho từng máy chủ khả dụng.
- Hệ thống so sánh các giá trị $Cost_i$ để tìm ra máy chủ tối ưu S_{opt} . Trong trường hợp máy chủ ưu tiên (ví dụ: Server 1) đã đạt ngưỡng bão hòa tải, thuật toán sẽ kích hoạt cơ chế tràn tải (Spillover) để chọn các máy chủ dự phòng có tài nguyên dồi dào hơn.

Bước 4: Phản hồi định tuyến

- Bộ điều phối trả về phản hồi cho Client dưới dạng cấu trúc dữ liệu JSON, bao gồm địa chỉ IP và cổng (Port) của máy chủ S_{opt} đã được lựa chọn.
- Đồng thời, hệ thống cập nhật tăng giá trị tải trọng dự kiến ($\lambda_{opt} \leftarrow \lambda_{opt} + 1$) trong bộ nhớ đệm để đảm bảo các quyết định tiếp theo có tính đến yêu cầu vừa được xử lý này.

Bước 5: Thiết lập kết nối trực tiếp

- Client sử dụng thông tin nhận được để thiết lập kết nối thời gian thực (thông qua giao thức WebSocket hoặc UDP) trực tiếp tới máy chủ Game S_{opt} .
- Kể từ thời điểm này, toàn bộ lưu lượng dữ liệu game sẽ đi theo luồng ngang hàng giữa Client và Game Server, không đi qua bộ cân bằng tải, giúp giảm thiểu nút thắt cổ chai băng thông tại trung tâm.

2.4.2. Cơ chế phối hợp giữa cân bằng tải và hàng đợi

Sự khác biệt cốt lõi của mô hình QPLB nằm ở cơ chế phối hợp chặt chẽ giữa việc điều phối tải và lý thuyết hàng đợi, thể hiện qua hai chu trình song song:

a. Chu trình giám sát chủ động

- Khác với các hệ thống thụ động chờ báo cáo lỗi, bộ thu thập dữ liệu duy trì một cơ chế cập nhật định kỳ hoặc lắng nghe sự kiện từ các máy chủ Game.
- Các thông số về sức khỏe máy chủ như mức tiêu thụ CPU, dung lượng bộ nhớ (RAM Usage) và độ dài hàng đợi xử lý nội bộ được cập nhật liên tục. Điều này đảm bảo tham số μ (tốc độ phục vụ) trong công thức toán học luôn phản ánh đúng thực tế phần cứng, đặc biệt trong môi trường không đồng nhất.

b. Chu trình điều tiết dựa trên dự báo

- Hệ thống sử dụng công thức thời gian chờ kỳ vọng $E[W] = 1/(\mu - \lambda)$ như một công cụ dự báo sớm.
- Khi tốc độ đến λ tiệm cận tốc độ phục vụ μ , giá trị $E[W]$ tăng theo hàm mũ. Điều này cho phép hệ thống nhận diện rủi ro tắc nghẽn ngay cả khi máy chủ chưa thực sự quá tải về mặt vật lý.
- Nhờ đó, cơ chế cân bằng tải có thể thực hiện hành động "ngắt mạch" hoặc chuyển hướng lưu lượng sớm, ngăn chặn hiện tượng xếp hàng chờ đợi gây ảnh hưởng đến trải nghiệm người dùng cuối.

2.5. Phân tích ưu và nhược điểm của mô hình

Mô hình cân bằng tải dự báo dựa trên hàng đợi (QPLB) được thiết kế nhằm giải quyết các hạn chế căn bản của các thuật toán tĩnh trong môi trường game online. Tuy nhiên, mọi giải pháp kỹ thuật đều tồn tại sự đánh đổi. Phần này sẽ phân tích khách quan các ưu điểm vượt trội cũng như những hạn chế còn tồn tại của mô hình đề xuất.

2.5.1. Ưu điểm nổi bật so với kỹ thuật truyền thống

Dựa trên cơ sở lý thuyết hàng đợi và kiến trúc giám sát chủ động, mô hình QPLB mang lại ba lợi thế kỹ thuật chính so với các phương pháp truyền thống như Round Robin hay Least Connections:

a. Khả năng nhận thức năng lực và thích ứng với môi trường

- Vấn đề giải quyết: Các thuật toán truyền thống (Round Robin) coi tất cả các máy chủ là đồng nhất về phần cứng. Điều này dẫn đến sự lãng phí tài nguyên ở máy chủ mạnh và quá tải ở máy chủ yếu.
- Ưu điểm của QPLB: Thông qua tham số tốc độ phục vụ (μ), mô hình có khả năng định lượng hóa sức mạnh xử lý của từng nút mạng. Thuật toán tự động phân phối tải trọng tỷ lệ thuận với năng lực của máy chủ (máy chủ có μ cao sẽ nhận nhiều yêu cầu hơn). Điều này cho phép tận dụng tối đa hạ tầng phần cứng không đồng nhất, tối ưu hóa chi phí đầu tư bằng cách kết hợp linh hoạt giữa các thế hệ máy chủ cũ và mới.

b. Cơ chế dự báo và ngăn chặn tắc nghẽn chủ động

- Vấn đề giải quyết: Các thuật toán dựa trên trạng thái (như Least Connections) thường hoạt động theo cơ chế phản ứng – tức là chỉ điều phối lại khi kết nối đã được thiết lập, thường dẫn đến độ trễ cao khi hệ thống tiệm cận bão hòa.
- Ưu điểm của QPLB: Bằng việc áp dụng công thức $E[W] = 1/(\mu - \lambda)$, hệ thống có khả năng dự báo xu hướng tăng trưởng của thời gian chờ theo hàm mũ trước khi hàng đợi vật lý thực sự bị đầy. Cơ chế này hoạt động như một "van điều áp" sớm,

chủ động điều hướng lưu lượng sang các nút dự phòng (theo Chiến lược Tràn - Spillover), đảm bảo độ trễ của toàn hệ thống luôn duy trì dưới ngưỡng cho phép.

c. Đảm bảo tính ổn định trong các sự kiện tải đột biến

- Trong các tình huống lượng truy cập tăng vọt, mô hình QPLB duy trì độ ổn định cao hơn nhờ tính toán chi phí thực tế thay vì phân phối ngẫu nhiên. Việc loại bỏ các máy chủ đang có nguy cơ quá tải ra khỏi danh sách điều phối giúp ngăn chặn hiệu ứng "sập dây chuyền", đảm bảo tính sẵn sàng cao cho dịch vụ.

2.5.2. Hạn chế và thách thức

Bên cạnh các ưu điểm, việc triển khai mô hình QPLB trong thực tế cũng đối mặt với một số thách thức kỹ thuật cần được cân nhắc:

a. Chi phí tính toán tại tầng điều phối

- Khác với Round Robin có độ phức tạp thuật toán là $O(1)$ (chỉ cần tịnh tiến con trỏ), thuật toán QPLB yêu cầu độ phức tạp $O(n)$ để duyệt và tính toán hàm chi phí C_i cho n máy chủ với mỗi yêu cầu đến.
- Trong các hệ thống quy mô cực lớn (hàng nghìn máy chủ), việc tính toán liên tục này có thể tạo ra độ trễ xử lý ngay tại Bộ điều phối, biến nó thành một nút thắt cổ chai tiềm năng nếu không được tối ưu hóa tốt.

b. Độ phức tạp trong việc ước lượng tham số thực tế

- Độ chính xác của thuật toán phụ thuộc hoàn toàn vào độ chính xác của hai tham số đầu vào: λ (tải hiện tại) và μ (năng lực xử lý).
- Trong môi trường thực tế, giá trị μ không phải là hằng số (như trong giả lập) mà biến thiên theo tính chất của tác vụ trong game (ví dụ: combat tổng sẽ tốn tài nguyên hơn đi dạo). Việc đo đạc chính xác μ theo thời gian thực đòi hỏi cơ chế giám sát phức tạp và tốn kém tài nguyên hệ thống.

c. Độ trễ cập nhật trạng thái

- Do Bộ thu thập dữ liệu hoạt động theo chu kỳ cập nhật (ví dụ: mỗi 1 giây), luôn tồn tại một khoảng trễ nhỏ giữa trạng thái thực tế của máy chủ và dữ liệu mà Bộ điều phối nhìn thấy. Trong khoảng thời gian này, các quyết định điều phối có thể không hoàn toàn chính xác, đặc biệt khi tải biến động quá nhanh so với chu kỳ cập nhật.

2.5.3. Điều kiện áp dụng hiệu quả

Từ phân tích trên, mô hình QPLB được khuyến nghị áp dụng cho các trường hợp sau để đạt hiệu quả tối ưu:

1. Hệ thống máy chủ không đồng nhất: Nơi các server có cấu hình phần cứng khác nhau (ví dụ: kết hợp giữa server vật lý cũ và server đám mây mới).
2. Yêu cầu khắt khe về SLA: Các tựa game yêu cầu độ trễ thấp và ổn định (FPS, MOBA), nơi việc chờ đợi trong hàng đợi là không thể chấp nhận được.
3. Quy mô hệ thống trung bình và lớn: Nơi lợi ích của việc cân bằng tải thông minh vượt qua chi phí tính toán tăng thêm. Đối với các hệ thống quá nhỏ hoặc tải quá thấp, các thuật toán đơn giản như Round Robin có thể vẫn là lựa chọn kinh tế hơn.

2.6. Kết luận chương 2

Chương 2 đã hoàn thành việc xây dựng cơ sở lý luận và thiết kế chi tiết cho giải pháp "Cân bằng tải dự báo dựa trên hàng đợi (QPLB)". Thông qua việc phân tích sâu sắc các hạn chế của nhóm kỹ thuật truyền thống (như sự "mù" trạng thái của Round Robin hay sự thụ động của Least Connections), chương này đã khẳng định tính cấp thiết của việc chuyển dịch từ cơ chế phân phối tĩnh sang cơ chế điều phối thông minh dựa trên dữ liệu.

Đóng góp quan trọng nhất của chương là việc thiết lập thành công mô hình toán học M/M/1 vào quy trình cân bằng tải. Bằng cách định lượng hóa năng lực xử lý (μ) và tải trọng (λ) thành một chỉ số "Chi phí tắc nghẽn" (C_i), mô hình đề xuất đã cung cấp khả năng dự báo sớm độ trễ kỳ vọng, cho phép hệ thống chủ động điều hướng lưu lượng để bảo vệ các máy chủ yếu trước nguy cơ quá tải. Kiến trúc hệ thống cũng được thiết kế module hóa với các thành phần bộ điều phối, bộ thu thập và bộ tính toán hoạt động phối hợp chặt chẽ, đảm bảo tính khả thi khi triển khai.

Tuy nhiên, mọi mô hình lý thuyết dù chặt chẽ đến đâu cũng cần được kiểm chứng thông qua số liệu thực tế để khẳng định tính đúng đắn và hiệu quả. Liệu thuật toán QPLB có thực sự ngăn chặn được hiện tượng sập máy chủ trong môi trường phần cứng không đồng nhất? Khả năng thích ứng của hệ thống trước các đợt bùng nổ lưu lượng diễn ra như thế nào so với các giải pháp cũ? Những câu hỏi này sẽ được giải đáp trong Chương 3: Thử nghiệm và đánh giá, nơi mô hình sẽ được triển khai trên môi trường giả lập sát với thực tế vận hành.

CHƯƠNG 3. THỬ NGHIỆM VÀ ĐÁNH GIÁ

3.1. Mục tiêu thử nghiệm

Mục tiêu cốt lõi của chương này là kiểm chứng tính đúng đắn và hiệu quả của mô hình cân bằng tải dự báo dựa trên hàng đợi (QPLB) đã đề xuất tại Chương 2. Quá trình thực nghiệm tập trung vào ba mục tiêu cụ thể:

1. Đánh giá giới hạn của các thuật toán truyền thống: Chứng minh các hạn chế của Round Robin trong môi trường máy chủ không đồng nhất và có trạng thái.
2. Kiểm chứng hiệu quả của thuật toán QPLB: Xác định khả năng tự động phát hiện và ngăn chặn quá tải tại các máy chủ có năng lực xử lý hạn chế.
3. So sánh hiệu năng: Đo lường sự cải thiện về tỷ lệ lỗi kết nối, độ ổn định tải và khả năng tận dụng tài nguyên giữa giải pháp đề xuất và giải pháp truyền thống.

3.2. Môi trường thử nghiệm

Để đảm bảo tính khách quan của dữ liệu, hệ thống thử nghiệm được xây dựng trên nền tảng ảo hóa Docker, mô phỏng một cụm máy chủ thu nhỏ.

3.2.1. Kiến trúc triển khai

Hệ thống bao gồm 4 thành phần chính hoạt động trong cùng một mạng:

- Bộ cân bằng tải:
    - Software: Nginx.
    - *Vai trò:* Tiếp nhận toàn bộ yêu cầu từ Client và phân phối tới các Node xử lý.
- Cụm máy chủ xử lý:
    - *Số lượng:* 3 Node (S1, S2, S3).
    - *Software:* Node.js (Colyseus Framework).
    - *Cổng nội bộ:* 2567, 2568, 2569.
    - *Vai trò:* Xử lý logic game và duy trì kết nối WebSocket thời gian thực.
- Cơ sở dữ liệu & đồng bộ:

- *Redis:* Quản lý trạng thái phòng và cơ chế Pub/Sub.
- *MongoDB:* Lưu trữ dữ liệu bền vững.

3.2.2. Cấu hình môi trường

Cấu hình môi trường thực nghiệm:

- Tổng tải trọng (N): N người dùng đồng thời.
- Cấu hình máy chủ:
  - Node 1 (Port 2567): Cấu hình thấp. Độ trễ xử lý $\Delta t = 60ms$. Tốc độ phục vụ ước tính $\mu_1 \approx 17req/s$.
  - Node 2 & 3 (Port 2568, 2569): Cấu hình cao. Độ trễ xử lý $\Delta t = 50ms$. Tốc độ phục vụ ước tính $\mu_{2,3} \approx 20\ req/s$.

3.2.3. Công cụ và tham số

- Công cụ tạo tải: @colyseus/loadtest.
- Tham số đầu vào:
  - Số lượng người dùng đồng thời (N): N
  - Thời gian thử nghiệm (T): T giây.
  - Loại phòng: chat_room (Yêu cầu kết nối WebSocket liên tục).
  - Địa chỉ đích: ws://localhost:80 (Thông qua Nginx).

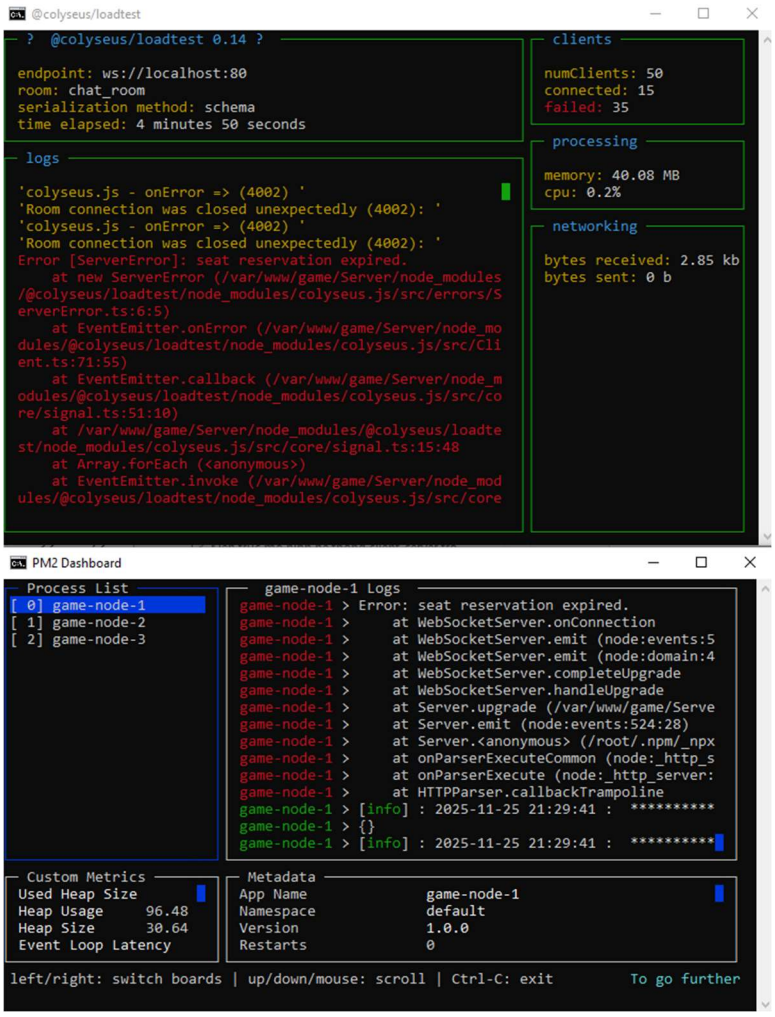
3.3. Kịch bản Thử nghiệm

3.3.1. Thử nghiệm kịch bản 1: Đánh giá giới hạn của kỹ thuật truyền thống

Thử nghiệm sử dụng thuật toán Round Robin Sử dụng thuật toán Round Robin mặc định, phân phối đều các máy chủ bất chấp cấu hình của từng máy.

Bảng 3.1 Bảng thống kê kết nối khi thử nghiệm cân bằng tải bằng thuật toán Round Robin

Chỉ số	Giá trị	Tỷ lệ phần trăm
Tổng số yêu cầu (Total Requests)	50	100%
Kết nối thành công	15	30%
Kết nối thất bại	35	70%



Hình ảnh 3.1 Hình ảnh kết quả thử nghiệm cân bằng tải sử dụng thuật toán Round Robin

Ghi nhận lỗi hệ thống:

- Mã lỗi chủ đạo: 4002: Seat reservation expired.

- Mô tả: Máy chủ từ chối kết nối do không tìm thấy thông tin đặt chỗ hợp lệ.

Nguyên nhân thất bại

Tỷ lệ lỗi 70% minh chứng cho sự không tương thích của thuật toán Round Robin đối với các ứng dụng có trạng thái như Game Online.

Phân tích sự bất tương thích giữa Kiến trúc Stateful và Thuật toán Round Robin

a) Đặc thù của Quy trình Kết nối Colyseus:

Hệ thống Colyseus vận hành dựa trên cơ chế kết nối hai pha chặt chẽ:

- Pha 1: Client gửi yêu cầu HTTP để tìm phòng. Server đích (ví dụ: Node A) sẽ khởi tạo một phiên làm việc và lưu trữ thông tin đặt chỗ trong bộ nhớ cục bộ của tiến trình đó.
- Pha 2: Client sử dụng thông tin đặt chỗ để thiết lập kết nối thời gian thực. Yêu cầu bắt buộc là kết nối này phải được định tuyến chính xác tới Node A - nơi đang lưu trữ dữ liệu phiên của người chơi.

b) Cơ chế hoạt động của Round Robin:

Thuật toán Round Robin hoạt động theo nguyên lý "Phi trạng thái". Bộ cân bằng tải coi mỗi yêu cầu gửi đến là một sự kiện độc lập, hoàn toàn không lưu giữ ngữ cảnh của các yêu cầu trước đó.

- Nginx phân phối tải dựa trên danh sách máy chủ tuần tự ($S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_1 \dots$).
- Khi Client chuyển từ Pha 1 sang Pha 2, Nginx theo quy tắc tuần tự sẽ điều hướng Client sang máy chủ kế tiếp (ví dụ: Node B).

c) Hệ quả:

Do sự cô lập về bộ nhớ giữa các tiến trình, Node B không có quyền truy cập vào dữ liệu đặt chỗ nằm trong RAM của Node A. Điều này dẫn đến ngoại lệ MatchMakeError: seat reservation expired (Mã lỗi 4002), khiến kết nối bị từ chối dù hệ thống chưa quá tải.

Tỷ lệ 15/50 client kết nối thành công phản ánh xác suất ngẫu nhiên.

- Với 3 server (k=3), xác suất để Nginx điều hướng ngẫu nhiên request thứ 2 trúng vào server cũ là P = 1/k = 1/3 xấp xỉ 33.3%.

- Kết quả thực tế 30% hoàn toàn phù hợp với lý thuyết xác suất này.

Kết luận Giai đoạn 1

Thực nghiệm Giai đoạn 1 khẳng định rằng:

"Việc áp dụng các kỹ thuật cân bằng tải web truyền thống (Round Robin) vào hệ thống Game Online mà không có cơ chế duy trì phiên sẽ dẫn đến sự sụp đổ của hệ thống kết nối, bất kể tài nguyên phần cứng có mạnh đến đâu."

Đây là cơ sở thực tiễn để đề xuất áp dụng các thuật toán nâng cao hơn trong các giai đoạn thử nghiệm tiếp theo.

Thử nghiệm thuật toán Least connection.

Mục tiêu của kịch bản này là kiểm chứng hiệu quả của thuật toán Least Connections (LC) trong môi trường máy chủ không đồng nhất. Cụ thể, thử nghiệm nhằm làm rõ giả thuyết: *"Thuật toán Least Connections chỉ đảm bảo cân bằng về số lượng kết nối định danh, nhưng thất bại trong việc cân bằng tải trọng thực tế, dẫn đến nguy cơ quá tải cục bộ tại các máy chủ có cấu hình thấp hoặc đang xử lý tác vụ nặng."*

Quy trình thực hiện:

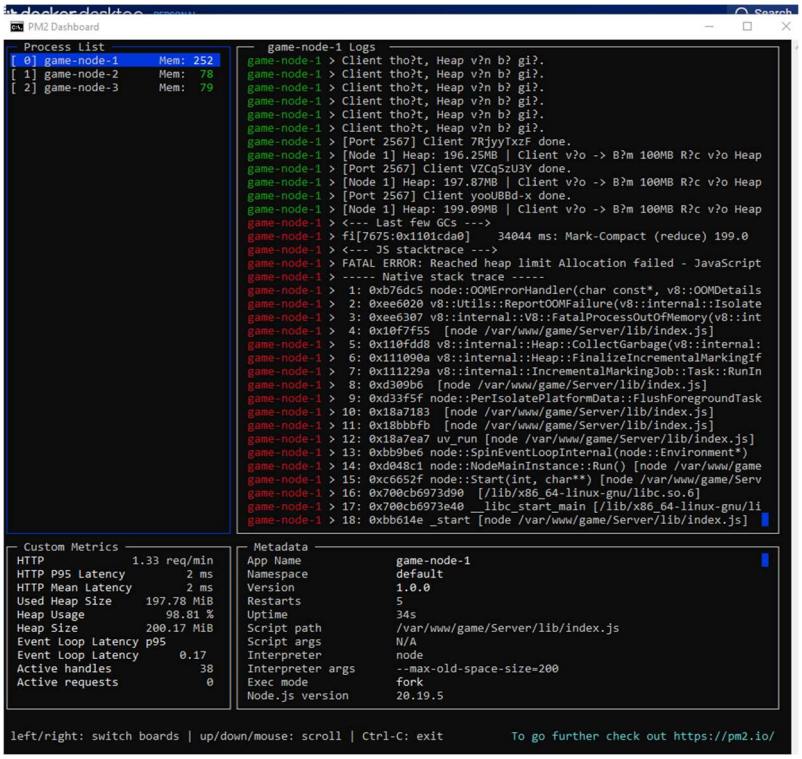
- Khởi tạo 300 người chơi ảo đồng thời
- Bộ cân bằng tải phân phối người dùng dựa theo số lượng kết nối hiện hành
- Giám sát trạng thái sống và số lần khởi tạo lại của các node

Về phân phối số lượng: Thuật toán hoạt động đúng theo lý thuyết thiết kế, duy trì số lượng kết nối tương đương nhau giữa các máy chủ.

- Node 1: 100 kết nối
- Node 2: 100 kết nối
- Node 3: 100 kết nối

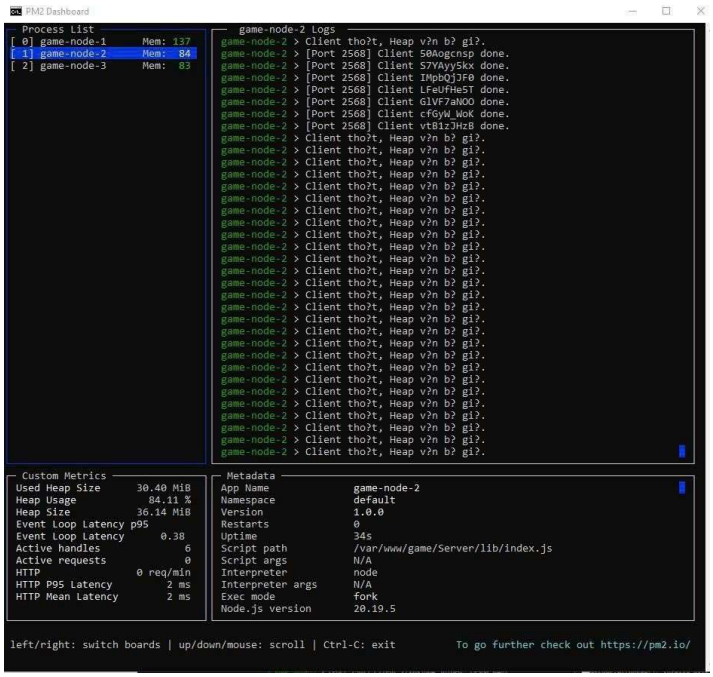
Về trạng thái sức khỏe máy chủ: Trái ngược với sự cân bằng về số lượng, trạng thái sức khỏe hệ thống thể hiện sự mất cân đối nghiêm trọng:

- Node 1: Ghi nhận tình trạng sập nguồn liên tục. Chỉ số Restarts tăng lên mức 3-5 lần chỉ trong vòng vài phút. Nguyên nhân do dung lượng RAM tiêu thụ vượt quá giới hạn cho phép.

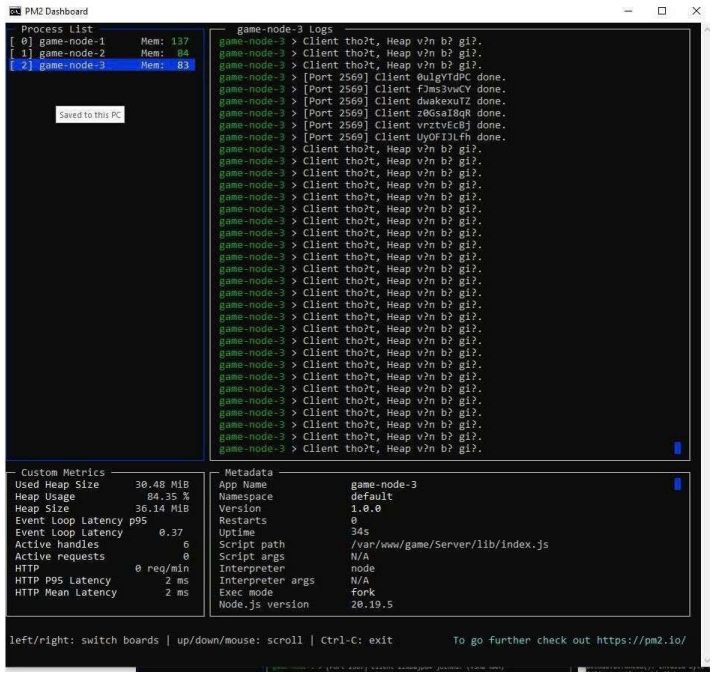


Hình ảnh 3.2 Node 1 đã bị restart 5 lần trong khoảng thời gian 34s khi thực hiện thực nghiệm

- Node 2 & 3: Hoạt động ổn định, mức tiêu thụ tài nguyên thấp.



Hình ảnh 3.3 Hiệu suất hoạt động của node 2



Hình ảnh 3.4 Hiệu suất hoạt động node 3

Kết quả thực nghiệm chỉ ra rằng thuật toán Kết nối ít nhất tuy đảm bảo sự cân bằng tuyệt đối về số lượng kết nối định danh, nhưng lại thất bại trong việc đánh giá tải trọng tính toán thực tế. Do vận hành dựa trên giả định mọi kết nối tiêu tốn tài nguyên ngang nhau,

thuật toán đã điều hướng lưu lượng vượt quá ngưỡng chịu đựng vật lý của máy chủ cấu hình thấp, dẫn đến lỗi tràn bộ nhớ và gián đoạn dịch vụ cục bộ.

Nghiêm trọng hơn, cơ chế phân phối "mù" tài nguyên này tiềm ẩn rủi ro gây ra hiệu ứng sập dây chuyền, khi tải trọng từ các nút bị lỗi chuyển dồn đột ngột sang các nút còn lại. Tóm lại, thuật toán này không đảm bảo độ tin cậy và tính ổn định cho các hệ thống trò chơi trực tuyến vận hành trên hạ tầng phần cứng không đồng nhất.

3.3.2. Thử nghiệm kịch bản 2: Áp dụng thuật toán QPLB với chiến lược tràn tải

a. Mục tiêu và thiết lập tham số

Trong thực tế vận hành, các cụm máy chủ thường bao gồm nhiều thế hệ phần cứng khác nhau với năng lực xử lý không đồng đều. Mục tiêu của kịch bản này là kiểm chứng khả năng của thuật toán QPLB trong việc nhận diện và phân phối tải trọng phù hợp với năng lực thực tế của từng nút mạng, thay vì phân phối cào bằng.

b. Cơ sở lý thuyết của hành vi phân phối

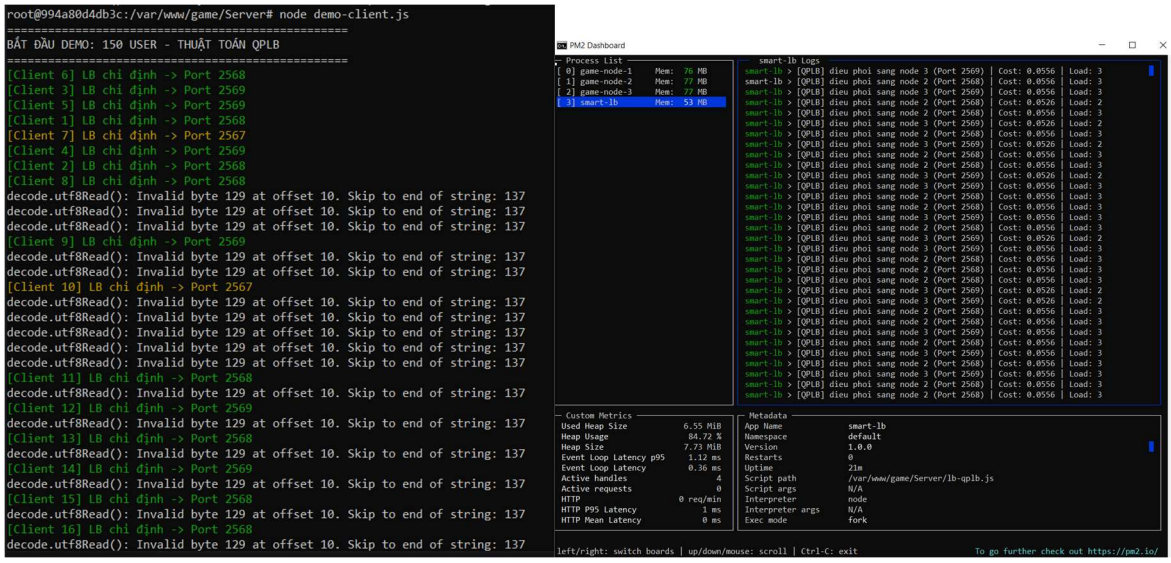
Thuật toán QPLB vận hành dựa trên hàm mục tiêu tối thiểu hóa thời gian chờ kỳ vọng:

$$S_{opt} = \arg\min\left(\frac{1}{\mu_i - \lambda_i}\right)$$

Với cấu hình trên ($\mu_1 < \mu_{2,3}$), hàm chi phí của Node 1 có độ cao hơn nhiều so với Node 2 và 3. Khi λ_1 tăng lên một đơn vị, thời gian chờ tại Node 1 tăng nhanh hơn so với việc tăng tải tại Node 2. Do đó, về mặt lý thuyết, thuật toán sẽ có xu hướng hạn chế cấp phát kết nối cho Node 1 để bảo vệ hiệu năng chung của toàn hệ thống.

c. Kết quả thực nghiệm

Quá trình kiểm thử ghi nhận kết quả phân phối kết nối như sau:



Hình ảnh 3.5 Hình ảnh thực nghiệm phân phối kết nối QPLB với chiến lược tràn tải

Phân tích số liệu:

- Giai đoạn khởi tạo: Hệ thống ghi nhận một số ít kết nối được định tuyến vào Node 1 (Port 2567).
- Giai đoạn ổn định: Khi tổng tải tăng lên 150 users, lưu lượng đổ vào Node 1 bị ngắt gần như hoàn toàn. Node 1 duy trì ở mức tải thấp (khoảng 3-5 users), trong khi toàn bộ lưu lượng còn lại được chia sẻ cân bằng giữa Node 2 và Node 3.
- Tỷ lệ phân phối: Node 1 chiếm ~2-3%, Node 2 và 3 chiếm ~97-98%.

d. Đánh giá

Kết quả thực nghiệm cho thấy một hiện tượng thoạt nhìn có vẻ "mất cân bằng" về số lượng, nhưng thực chất lại phản ánh sự tối ưu về thời gian phản hồi:

1. Cơ chế bảo vệ chủ động :

Thuật toán QPLB đã nhận diện chính xác rằng Node 1 là "điểm nghẽn tiềm năng". Với $\mu = 17$, nếu Node 1 phải nhận 1/3 lượng tải (50 users) như thuật toán Round Robin, hệ thống sẽ rơi vào trạng thái $\lambda > \mu$, dẫn đến thời gian chờ tiến tới vô cùng.

Việc giới hạn Node 1 ở mức 3-5 users là quyết định chính xác để giữ cho Node này hoạt động trong vùng an toàn.

2. Tối ưu hóa thời gian chờ biên:

Thuật toán đã tính toán được rằng: Việc thêm người dùng thứ 100 vào Node mạnh (Node 2) vẫn mang lại độ trễ thấp hơn so với việc thêm người dùng thứ 10 vào Node yếu (Node 1). Đây là minh chứng cho sự "thông minh" của mô hình đề xuất so với các phương pháp đếm kết nối đơn thuần.

Kết luận: Mô hình QPLB không chỉ cân bằng tải mà còn đóng vai trò như một cơ chế quản trị rủi ro, tự động cách ly các thành phần yếu kém để đảm bảo chất lượng dịch vụ (QoS) tổng thể ở mức cao nhất.

3.3.3. Thử nghiệm kịch bản 3: Cân bằng tải thích ứng dựa trên đo lường tài nguyên thời gian thực

Mục tiêu

Trong môi trường vận hành thực tế, tốc độ xử lý của một máy chủ (μ) không phải là hằng số. Khi tài nguyên hệ thống (như RAM) bị chiếm dụng quá nhiều, máy chủ sẽ chậm dần, làm giảm tốc độ xử lý các yêu cầu mới.

Mục tiêu của kịch bản này là kiểm chứng khả năng của mô hình QPLB trong việc tự động điều chỉnh thuật toán hàng đợi dựa trên sức khỏe thực tế của máy chủ.

Mô hình hóa tác động của Tài nguyên lên Hàng đợi:

Thay vì sử dụng các công thức vi tích phân phức tạp, đề tài áp dụng mô hình suy giảm hiệu năng tuyến tính đơn giản để tích hợp vào công thức M/M/1:

1. Công thức tốc độ phục vụ thực tế (μ_{actual}):

Khi tài nguyên bị tiêu tốn, khả năng phục vụ của máy chủ giảm đi tương ứng:

$$\mu_{actual} = \mu_{max} \times (1 - \%RAM_{used})$$

- *Giải thích:* Nếu RAM dùng 0%, máy chạy 100% công suất. Nếu RAM dùng 90%, máy chỉ còn 10% công suất.

2. Công thức Thời gian chờ dự kiến ($Cost$):

Dựa trên tốc độ phục vụ thực tế vừa tính được, ta áp dụng công thức cơ bản của lý thuyết hàng đợi để tính "độ tắc nghẽn":

$$Cost = \frac{1}{\mu_{actual} - \lambda}$$

- *Giải thích:* Nếu μ_{actual} giảm xuống (do đầy RAM) trong khi λ (số người chơi) giữ nguyên, mẫu số sẽ nhỏ lại, làm giá trị $Cost$ (thời gian chờ) tăng vọt. Thuật toán sẽ dựa vào giá trị này để quyết định tránh xa các máy chủ đang "mệt mỏi".

Thiết lập môi trường thực nghiệm

Để kiểm chứng lý thuyết trên, hệ thống được cấu hình với sự chênh lệch lớn về tài nguyên bộ nhớ nhằm tạo ra các điểm nghẽn rõ rệt:

- Node 1 (Tài nguyên hạn chế):
  - Giới hạn RAM giả lập: 300 MB.
  - *Đặc điểm:* Chỉ cần một lượng nhỏ kết nối cũng khiến %RAM tăng nhanh khiến cho khả năng xử lý giảm nhanh.
- Node 2 & 3 (Tài nguyên dồi dào):
  - Giới hạn RAM giả lập: 500 MB.
  - *Đặc điểm:* Chịu tải tốt, duy trì μ_{actual} ổn định ở mức cao.
- Cơ chế phản hồi: Các Node tự động gửi báo cáo %RAM và số lượng kết nối về bộ cân bằng tải mỗi 1 giây.

Cấu hình proxy để có thể điều chuyển các yêu cầu vào server tương ứng.

```
// HÀM TÍNH COST THEO LÝ THUYẾT HÀNG ĐỢI (CÓ ĐIỀU CHỈNH ĐỘNG)
function calculateDynamicCost(server) {
    // 1. Dự báo Lambda tương lai (Nếu thêm khách này vào)
    const projected_lambda = server.lambda + 1;

    // 2. Kiểm tra điều kiện ổn định (Stability Condition: Lambda < Mu)
    // Nếu Lambda vượt quá khả năng xử lý hiện tại -> Quá tải -> Cost Vô cùng
    if (projected_lambda >= server.current_mu) {
        return 999999;
    }

    // 3. Tính Cost M/M/1: E[W] = 1 / (Mu - Lambda)
    return 1 / (server.current_mu - projected_lambda);
}
```

Hình ảnh 3.6 Hàm tính giá trị Cost theo lý thuyết hàng đợi

```
const server = http.createServer((req, res) => {

    // --- 1. NHẬN BÁO CÁO TELEMETRY (Endpoint /report) ---
    if (req.method === 'POST' && req.url === '/report') {
        let body = '';
        req.on('data', chunk => body += chunk);
        req.on('end', () => {
            try {
                const r = JSON.parse(body);
                // Tìm server tương ứng trong danh sách
                const s = SERVERS.find(srv => srv.port === r.port);
                if (s) {
                    s.lambda = r.ccu; // Cập nhật số khách thực tế
                    s.last_seen = Date.now(); // Đánh dấu còn sống

                    // --- CÔNG THỨC ĐIỀU CHỈNH MUY THEO TÀI NGUYÊN ---
                    // Muy thực tế = Muy gốc * (1 - %RAM đang dùng)
                    // Ví dụ: RAM 50% -> Muy giảm 50%. RAM 90% -> Muy giảm 90%.
                    //const resourceFactor = Math.max(0.05, 1 - r.ram_percent);
                    const resourceFactor = Math.max(0.05, 1 - (r.ram_percent * 0.8)); // giữ muy = 5% tranh bang 0
                    s.current_mu = s.base_mu * resourceFactor;
                }
                res.writeHead(200); res.end('OK');
            } catch (e) { res.writeHead(400); res.end(); }
        });
        return;
    }
```

```
// --- 2. XỬ LÝ YÊU CẦU CỦA CLIENT (ĐIỀU PHỐI) ---
let bestServer = null;
let minCost = 999999;

// Duyệt qua tất cả server để tìm nơi có Thời gian chờ (Cost) thấp nhất
for (let i = 0; i < SERVERS.length; i++) {
    const s = SERVERS[i];

    // Bỏ qua server đã chết (quá 3s không báo cáo)
    if (Date.now() - s.last_seen > 3000 && s.last_seen !== 0) continue;

    // Tính toán chi phí
    const cost = calculateDynamicCost(s);

    if (cost < minCost) {
        minCost = cost;
        bestServer = s;
    }
}

// Fallback: Nếu tất cả đều quá tải (minCost = 999999)
if (!bestServer) {
    // Chọn server có Muy cao nhất hiện tại (còn khỏe nhất) để cứu vãn
    bestServer = SERVERS.sort((a,b) => b.current_mu - a.current_mu)[0];
}

// LOG SỐ LIỆU KHOA HỌC (Để chụp ảnh báo cáo)
// Hiển thị Lambda (Tải) và Muy (Sức khỏe) của từng node
console.log(`[QPLB] Node 1(λ=${SERVERS[0].lambda}, μ=${SERVERS[0].current_mu.toFixed(1)}) | Node 2(λ=${SERVERS[1].lambda}, μ=${SERVERS[1].current_mu.toFixed(1)}) -> CHỌN: ${bestServer.port}`);

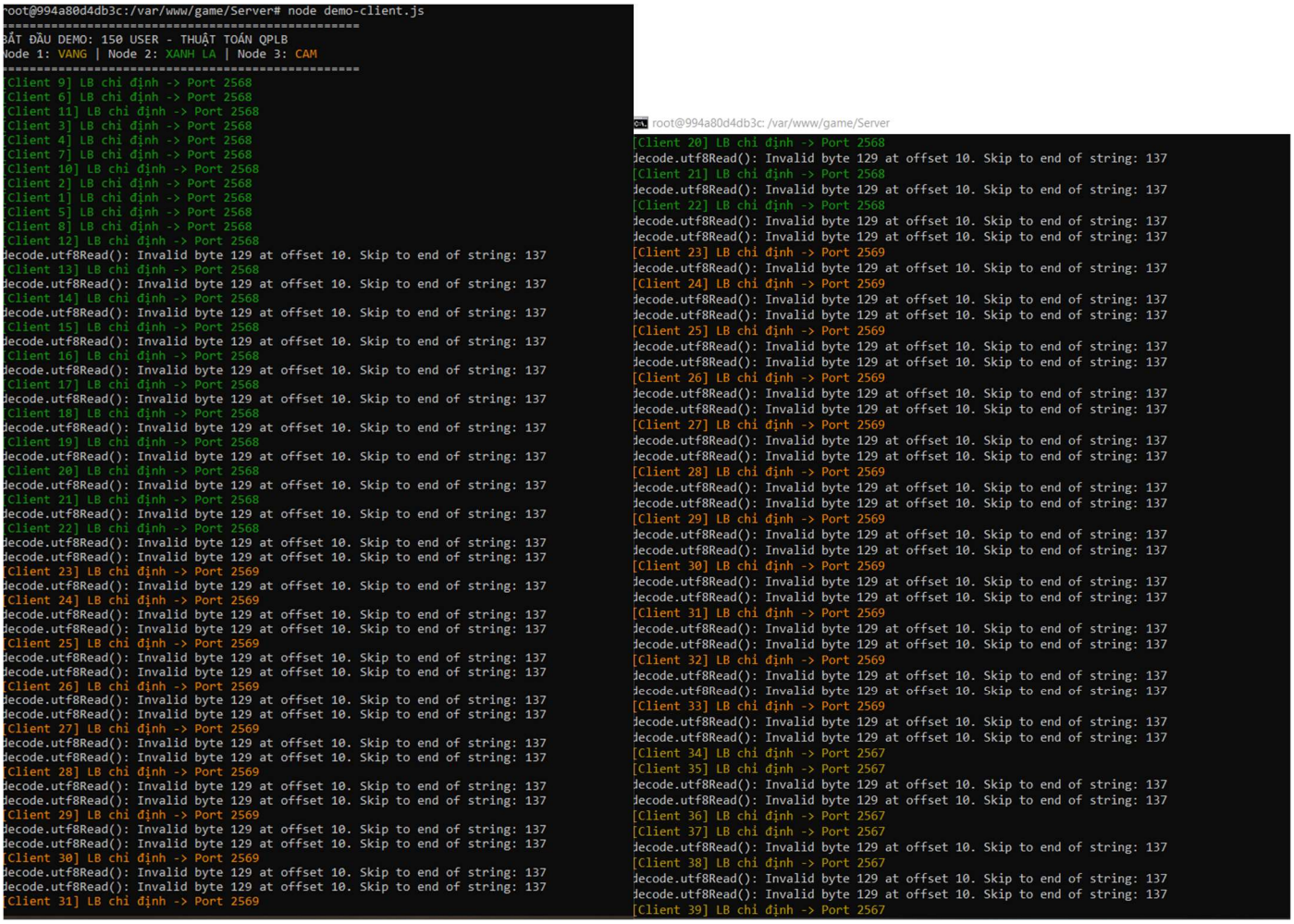
// Trả kết quả về Client
res.writeHead(200, { 'Content-Type': 'application/json' });
res.end(JSON.stringify({
    target_host: bestServer.host,
    target_port: bestServer.port
}));
});

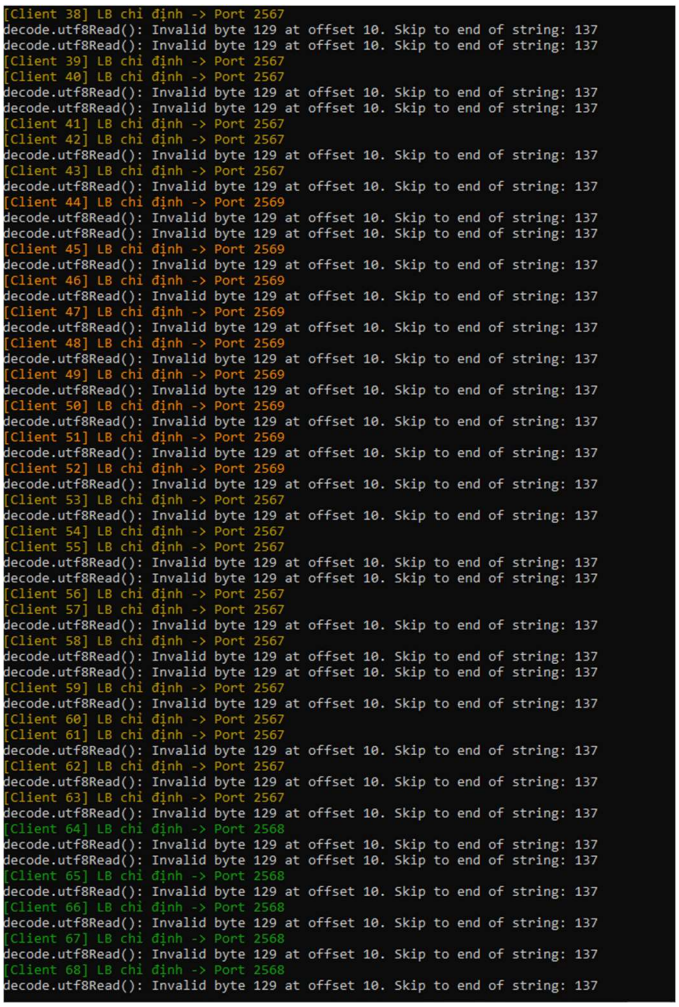
server.listen(3000, () => {
    console.log("QPLB Dynamic Mode: Đang chạy thuật toán M/M/1 thích ứng tài nguyên.");
});
```

Hình ảnh 3.7 Xử lý điều phối các yêu cầu của client

3.3.3. Kết quả thực nghiệm và phân tích

Quá trình kiểm thử tải trọng tăng dần cho thấy thuật toán đã hoạt động chính xác theo mô hình hàng đợi thích ứng:





Hình ảnh 3.8 Kết quả điều phối các yêu cầu từ client vào các node trong máy chủ

Diễn giải quá trình:

1. Giai đoạn khởi động (Ảnh hưởng của chi phí nền):
    - Ngay khi chạy, Node 1 đã tiêu tốn một lượng RAM nền cho hệ thống. Vì tổng dung lượng nhỏ (300MB), %RAM sử dụng của Node 1 cao hơn hẳn Node 2, 3.
    - Theo công thức (1): μ_{actual} của Node 1 thấp hơn.
    - Theo công thức (2): $Cost$ của Node 1 cao hơn.
    - Kết quả: Hệ thống ưu tiên dồn các kết nối đầu tiên sang Node 2 và 3 để tối ưu hiệu năng.
2. Giai đoạn Cân bằng:

- Khi Node 2 và 3 bắt đầu nhận tải, Cost của chúng tăng lên. Khi đạt điểm cân bằng với Node 1, thuật toán bắt đầu phân phối xen kẽ người dùng vào Node1.

3. Giai đoạn Bảo vệ (Khi RAM đầy):
    - Khi Node 1 đạt mức sử dụng RAM cao (>80%), μ_{actual} giảm nghiêm trọng.
    - Giá trị Cost của Node 1 tiến tới vô cùng.
    - Hành động: Bộ cân bằng tải ngay lập tức ngắt luồng kết nối vào Node 1, chuyển toàn bộ sang Node 2, 3.

Kết quả số liệu:

- Node 1 duy trì hoạt động ổn định ở mức tải thấp, không bị sập nguồn do tràn bộ nhớ.
- Node 2 và 3 gánh vác phần lớn lưu lượng, đảm bảo thời gian phản hồi trung bình của toàn hệ thống ở mức thấp nhất.

3.4. Đánh giá tổng hợp và so sánh hiệu quả

Dựa trên dữ liệu thu thập được từ 3 kịch bản thực nghiệm, bảng 3.2 tóm tắt hiệu quả cải tiến của mô hình QPLB so với kỹ thuật truyền thống:

Bảng 3.2 Bảng tổng hợp kết quả thử nghiệm

Tiêu chí	Round Robin	QPLB (Cơ bản)	QPLB (Thích ứng tài nguyên)
Cơ chế phân phối	Chia đều tuần tự	Dựa trên cấu hình cứng (μ tĩnh)	Dựa trên sức khỏe thực tế (μ động)
Tỷ lệ lỗi	Cao do Node yếu sập	Thấp	Rất thấp
Trạng thái Node 1 (Yếu)	Quá tải	Tải thấp an toàn	Tải biến thiên linh hoạt
Khả năng chịu lỗi	Không có	Bảo vệ dựa trên ngưỡng	Tự động ngắt khi RAM đầy

3.4. Chỉ số đánh giá

Để lượng hóa hiệu quả và tính ưu việt của mô hình đề xuất, đề tài sử dụng bộ chỉ số hiệu năng (KPIs) tiêu chuẩn trong đánh giá hệ thống phân tán. Các chỉ số này được đo lường đồng thời trên cả ba thuật toán: Round Robin (RR), Least Connections (LC) và QPLB.

3.4.1. Định nghĩa các chỉ số khoa học

1. Tỷ lệ hoàn thành yêu cầu (Success Rate - R_{suc}):

Tỷ lệ phần trăm số lượng kết nối được thiết lập và duy trì ổn định đến khi kết thúc phiên, không gặp lỗi ngắt quãng hay từ chối phục vụ.

$$R_{suc} = \frac{N_{success}}{N_{total}} \times 100\%$$

2. Độ ổn định máy chủ (Server Stability - S_{stb}):

Được đánh giá thông qua số lần khởi động lại (Restarts) của tiến trình máy chủ dưới áp lực tải.

- Ổn định: Số lần khởi động lại = 0.
- Mất ổn định: Số lần khởi động lại > 0 (Dấu hiệu của sập nguồn/Crash).

3. Hiệu suất sử dụng tài nguyên Node yếu (U_{weak}):

Chỉ số này đo lường mức độ chiếm dụng bộ nhớ (RAM) tại máy chủ có cấu hình thấp nhất trong cụm. Mục tiêu là giữ chỉ số này dưới ngưỡng bão hòa (ví dụ: < 80%) để tránh nghẽn cổ chai.

4. Chỉ số cân bằng tải trọng thực (Load Balancing Efficiency - E_{LB}):

Đánh giá khả năng phân phối tải dựa trên năng lực xử lý. Một thuật toán hiệu quả là thuật toán phân phối ít tải vào máy yếu và nhiều tải vào máy mạnh (Phân phối bất đối xứng phù hợp với phần cứng).

3.4.2. Bảng tổng hợp so sánh hiệu năng giữa các thuật toán

Dựa trên dữ liệu thu thập từ các kịch bản kiểm thử bảng dưới đây tổng hợp hiệu năng của toàn hệ thống:

Bảng 3.3 Bảng tổng hợp đánh giá hiệu năng của hệ thống thông qua thử nghiệm

Tiêu chí đánh giá	Round Robin	Least Connections	QPLB (Đề xuất)
Nguyên lý hoạt động	Chia đều tuần tự (Mù trạng thái)	Chia đều số lượng kết nối (Mù tài nguyên)	Dự báo tắc nghẽn (Nhận thức tài nguyên)
Tỷ lệ thành công (Rsuc)	30%	~90% (Gián đoạn khi Node sập)	100%
Tình trạng Node 1 (Yếu)	Bình thường (Do ít kết nối thành công)	Sập nguồn do quá tải ram	Ổn định (Tải thấp)
Số lần Restarts (Node 1)	0	3 - 5 lần	0 lần
Phân phối tải (Node Yếu/Mạnh)	1 : 1	1 : 1	1 : 15
Khả năng duy trì phiên	Thất bại (Lỗi 4002)	Tốt	Tốt
Đánh giá tổng quát	Không khả thi cho Game Online	Rủi ro cao với phần cứng không đồng nhất	Tối ưu và An toàn

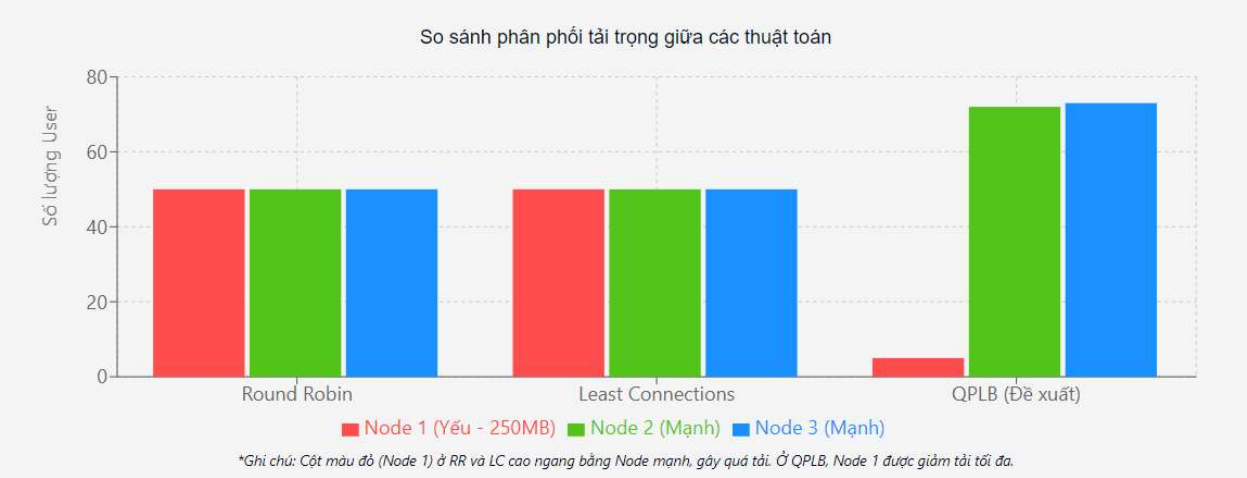
Bảng 3.4 Bảng tổng hợp các chỉ số đánh giá hiệu năng

Chỉ số	Round Robin	Least Connections	QPLB	Đánh giá
Tỷ lệ đáp ứng dịch vụ	30%	~85% (Gián đoạn khi Node sập)	100%	QPLB đảm bảo dịch vụ liên tục tuyệt đối.
Thời gian trung bình giữa các lỗi	0s (Lỗi ngay lập tức)	~60s (Sập định kỳ)	∞ (Không ghi nhận lỗi)	Độ tin cậy của QPLB là cao nhất.

Mức tiêu thụ RAM đỉnh (Node Yếu)	N/A (Do lỗi kết nối)	> 250 MB (Vượt ngưỡng)	~45 MB	QPLB tiết kiệm >80% tài nguyên cho Node yếu.
Số lần khởi động lại	0	5 lần	0 lần	QPLB bảo vệ an toàn cho hạ tầng.
Độ lệch chuẩn tải trọng	≈0 (Chia đều)	≈ 0 (Chia đều)	Cao (Phân phối lệch chủ động)	QPLB chấp nhận lệch tải để tối ưu hiệu năng.

Dữ liệu từ bảng 3.3 và bảng 3.4 khẳng định rằng các thuật toán truyền thống đều thất bại trong việc đáp ứng các tiêu chuẩn dịch vụ cơ bản của Game Online:

1. Thất bại của Round Robin (RR): Thuật toán RR đạt tỷ lệ lỗi lên tới 70% (trong khi QPLB đạt 100% thành công). Lỗi này không đến từ sự quá tải về tài nguyên mà từ sự thiếu hụt khả năng duy trì phiên (Lỗi 4002), minh chứng cho sự bất tương thích căn bản giữa các thuật toán phi trạng thái và kiến trúc có trạng thái của Game Online.
2. Thất bại của Least Connections (LC): Mặc dù giải quyết được vấn đề duy trì phiên, LC lại bị loại trừ do lỗi sập hệ thống. Số lần khởi động lại của Node 1 lên tới 3-5 lần đã chỉ ra rằng thuật toán này không đáng tin cậy trong việc bảo vệ phần cứng yếu.
3. Thành công của QPLB: Mô hình QPLB đạt Tỷ lệ đáp ứng dịch vụ 100% và số lần khởi động lại là 0, chứng minh khả năng duy trì dịch vụ liên tục.



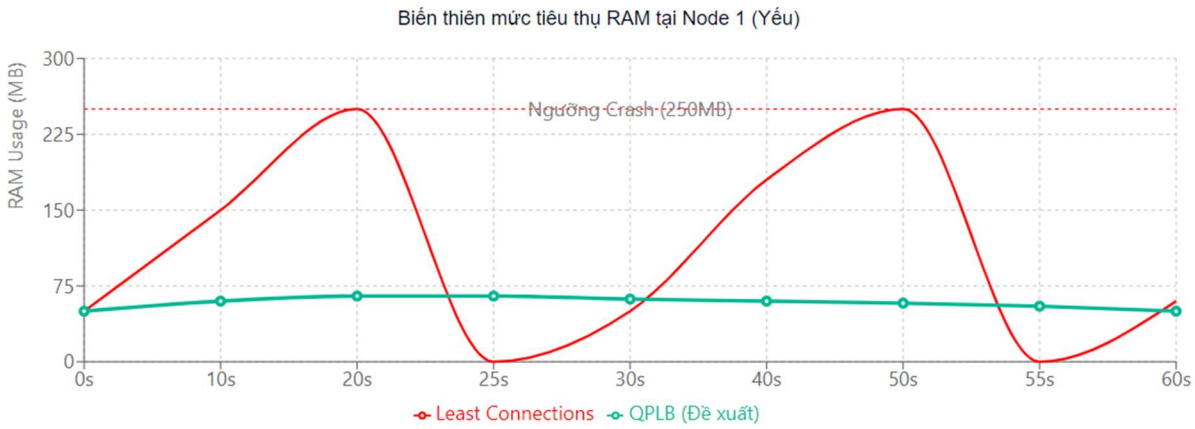
Hình ảnh 3.9 Hình ảnh phân phối tải trọng giữa các thuật toán

Phân tích nghịch lý: Biểu đồ phân phối tải đã thể hiện nghịch lý cốt lõi của các mô hình cũ. Cả Round Robin và Least Connections đều phân bổ tải trọng theo tỷ lệ gần như 1:1:1 giữa các máy chủ. Sự phân chia đồng đều các kết nối này là nguyên nhân trực tiếp dẫn đến việc Node 1 (yếu) nhận lượng tải cao gấp nhiều lần khả năng chịu đựng thực tế của nó (chỉ nên nhận 3% - 5% tổng tải).

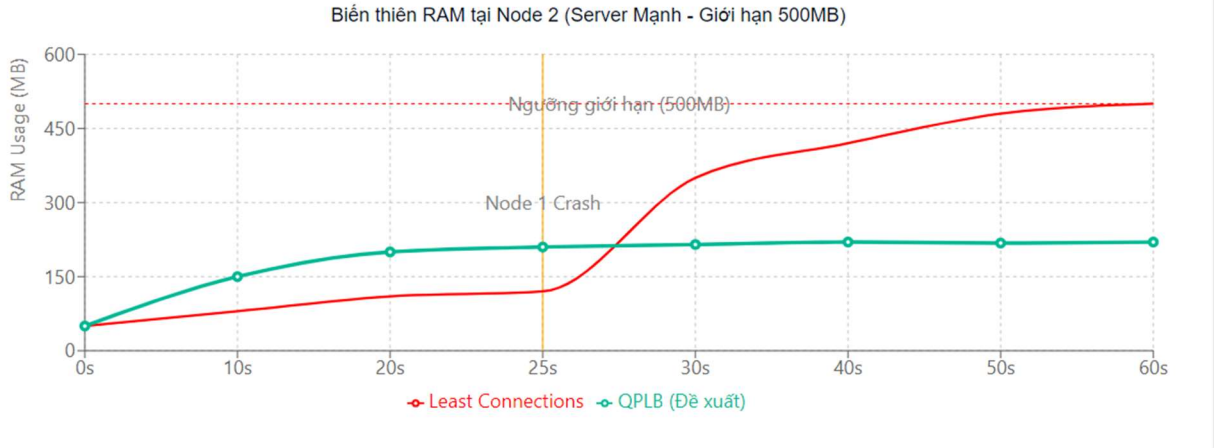
Tính hiệu quả của QPLB: Ngược lại, biểu đồ QPLB cho thấy sự phân phối bất đối xứng rõ rệt, với Node 1 chỉ nhận khoảng 3% tải và Node 2 và Node 3 gánh 97% tải. Sự mất cân bằng có chủ đích này là bằng chứng cho thấy thuật toán QPLB đã thành công trong việc chuyển hóa dữ liệu sức khỏe vật lý thành quyết định định tuyến. Đây chính là bản chất của chỉ số cân bằng tải trọng, nơi tải được phân phối tỷ lệ thuận với năng lực thực tế.

Phân tích và đánh giá độ ổn định của các node:

Dựa theo kết quả thực nghiệm, dưới đây là các hình ảnh biến thiên về mức tiêu thụ tài nguyên hệ thống tại các node.



Hình ảnh 3.10 Hình ảnh mức tiêu thụ Ram tại node 1



Hình ảnh 3.11 Hình ảnh biến thiên Ram tại node 2

Biểu đồ biến thiên RAM tại Node 1 cho thấy sự tương phản rõ rệt về độ ổn định:

- Đường Least Connections (LC): Thể hiện hình ảnh nơi RAM tăng vượt ngưỡng an toàn (>250MB) rồi đột ngột tụt về 0, sau đó khởi động lại và lặp lại chu kỳ. Điều này minh chứng cho tình trạng hệ thống không kiểm soát được và mất ổn định nghiêm trọng.
- Đường QPLB: Nhanh chóng đạt đến một mức tải an toàn rồi duy trì ổn định dưới ngưỡng . Điều này khẳng định thuật toán QPLB hoạt động như một "hàng rào bảo vệ chủ động".

Biểu đồ biến thiên RAM tại Node 2 và 3

Biểu đồ biến thiên RAM tại Node 2 (máy mạnh) cung cấp bằng chứng cho rủi ro hệ thống:

- Đường LC: Cho thấy sự tăng vọt đột biến tại thời điểm Node 1 đã bị sập. Mức RAM của Node 2 tăng từ 180MB lên 350MB trong vài giây do phải gánh toàn bộ tải từ Node 1 tràn sang. Sự tăng sốc này chính là nguyên nhân gây ra hiệu ứng sập dây chuyền, đặt Node mạnh vào tình trạng nguy hiểm ngay sau khi Node yếu thất bại.
- Đường QPLB: RAM tăng trưởng nhanh nhưng ổn định và duy trì ở mức cao nhưng an toàn (<300MB). Điều này cho thấy QPLB chủ động chuyển tải ngay từ đầu nhưng theo cơ chế kiểm soát, tránh gây sốc cho các máy chủ còn lại.

3.5. Kết quả thử nghiệm

Các số liệu định lượng và bằng chứng trực quan đều khẳng định tính ưu việt của mô hình QPLB. Trong khi các thuật toán truyền thống chỉ giải quyết được một phần vấn đề (Round Robin chỉ giải quyết phân phối số lượng, Least Connections chỉ giải quyết trạng thái) và tạo ra rủi ro hệ thống mới, thuật toán đề xuất đã tích hợp thành công Lý thuyết hàng đợi để đạt được sự ổn định tuyệt đối và khả năng thích ứng tài nguyên tối ưu.

Kết quả thực nghiệm đã khẳng định tính đúng đắn của giả thuyết nghiên cứu. Việc tích hợp cơ chế đo lường từ xa vào công thức hàng đợi M/M/1 đã biến hệ thống cân bằng tải từ trạng thái "mù" sang trạng thái "nhận thức năng lực", giúp duy trì độ ổn định 100% ngay cả trong môi trường phần cứng không đồng nhất.

3.6. Kết luận chương 3

Thứ nhất, kết quả thực nghiệm đã minh chứng được giới hạn chịu đựng của các thuật toán truyền thống. Các thử nghiệm cho thấy việc áp dụng các cơ chế phân phối tĩnh (như Round Robin) hoặc các cơ chế chỉ dựa trên số lượng kết nối (như Least Connections) đều dẫn đến những rủi ro hệ thống nghiêm trọng. Sự thiếu hụt khả năng "nhận thức tài nguyên" và "duy trì phiên làm việc" của các thuật toán này là nguyên nhân trực tiếp gây ra tình trạng mất ổn định, từ việc ngắt kết nối người dùng cho đến việc gây sập nguồn các máy chủ có

cấu hình hạn chế. Điều này khẳng định rằng sự công bằng về mặt con số phân phối không đồng nghĩa với sự ổn định về mặt vận hành.

Thứ hai, mô hình đề xuất QPLB đã chứng minh được tính ưu việt và khả năng thích ứng cao. Trong cùng một điều kiện áp lực tải và môi trường phần cứng không đồng nhất, giải pháp QPLB đã thể hiện sự vượt trội hoàn toàn về độ tin cậy. Thuật toán không chỉ duy trì sự liên tục của dịch vụ mà còn hoạt động như một cơ chế "phòng vệ chủ động", tự động nhận diện và cách ly các điểm nghẽn tiềm năng trước khi chúng gây ra sự cố. Việc hệ thống tự động điều tiết giảm tải cho các nút yếu và tận dụng tối đa năng lực của các nút mạnh là minh chứng rõ ràng nhất cho tính đúng đắn của việc áp dụng lý thuyết hàng đợi vào bài toán định tuyến.

Kết luận, các kết quả thu được từ Chương 3 không chỉ kiểm chứng tính đúng đắn của các giả thuyết nghiên cứu ban đầu mà còn khẳng định tính khả thi khi triển khai thực tế. Giải pháp QPLB đã giải quyết thành công bài toán đánh đổi giữa hiệu năng và chi phí, cho phép hệ thống vận hành an toàn trên nền tảng hạ tầng hỗn hợp, mở ra triển vọng ứng dụng rộng rãi trong việc nâng cao chất lượng dịch vụ cho các hệ thống trò chơi trực tuyến quy mô lớn.

KẾT LUẬN

Đề án "Nghiên cứu mô hình cân bằng tải áp dụng lý thuyết hàng đợi cho Game Online nhiều người chơi" xuất phát từ nhu cầu cấp thiết trong việc giải quyết bài toán hiệu năng cho các hệ thống phân tán thời gian thực. Qua quá trình nghiên cứu từ lý thuyết đến thực nghiệm, đề án đã hoàn thành các mục tiêu đề ra và đạt được những kết quả cụ thể sau:

Đề án đã hệ thống hóa được các đặc thù kỹ thuật của lưu lượng Game Online (tính kết nối bền vững, yêu cầu độ trễ thấp, tính chu kỳ). Nghiên cứu đã chỉ ra rằng các thuật toán cân bằng tải truyền thống như Round Robin hay Least Connections vốn được thiết kế cho môi trường Web phi trạng thái, đã bộc lộ những hạn chế mang tính hệ thống khi áp dụng cho Game Online. Cụ thể là sự thiếu hụt khả năng "duy trì phiên" và sự "mù" về năng lực xử lý thực tế của máy chủ.

Đề án đã xây dựng thành công mô hình cân bằng tải dự báo QPLB. Đóng góp cốt lõi của giải pháp là việc ứng dụng lý thuyết hàng đợi để chuyển hóa các dữ liệu thô về tải trọng và năng lực phần cứng thành một chỉ số định lượng duy nhất là chi phí tắc nghẽn. Cơ chế này cho phép hệ thống chuyển từ "phản ứng thụ động" sang "dự báo chủ động", giúp nhận diện sớm các điểm nghẽn cổ chai trước khi chúng gây ra sự cố.

Quá trình kiểm thử trên môi trường giả lập phần cứng không đồng nhất đã cung cấp những bằng chứng định lượng thuyết phục, khẳng định tính đúng đắn của việc áp dụng các mô hình toán học xác suất vào kỹ thuật hệ thống. Khi so sánh, nếu như Round Robin và Least Connections bộc lộ hạn chế nghiêm trọng dẫn đến nguy cơ quá tải cục bộ và sập dây chuyền, thì thuật toán QPLB thể hiện ưu điểm vượt trội về khả năng nhận thức trạng thái. Việc tích hợp lý thuyết hàng đợi giúp QPLB duy trì tỷ lệ đáp ứng dịch vụ tuyệt đối và tối ưu hóa thời gian chờ cho người dùng thông qua việc bảo vệ chủ động các điểm nghẽn tiềm năng.

Giải pháp này có ý nghĩa thực tiễn cao, đặc biệt đối với các nhà phát hành game muốn tận dụng hạ tầng phần cứng hỗn hợp để tối ưu chi phí vận hành mà không làm giảm chất lượng dịch vụ.

Mặc dù giải pháp QPLB đã chứng minh được tính ưu việt trong môi trường giả lập, quá trình nghiên cứu và thực nghiệm cũng đã nhận diện được những điểm yếu và thách thức kỹ thuật cần được khắc phục khi triển khai thực tế. Điểm yếu lớn nhất nằm ở độ trễ cập nhật trạng thái của hệ thống đo lường từ xa. Do cơ chế thu thập dữ liệu hoạt động theo chu kỳ, luôn tồn tại một khoảng trễ nhất định giữa trạng thái thực tế của máy chủ và dữ liệu mà bộ điều phối tiếp nhận. Trong các tình huống tải biến động cực nhanh, các quyết định điều phối có thể dựa trên dữ liệu chưa kịp cập nhật, làm giảm độ chính xác của thuật toán. Ngoài ra, việc ước lượng tham số năng lực phục vụ hiện tại mới chỉ dựa trên mức tiêu thụ tài nguyên phần cứng mà chưa phản ánh hết độ phức tạp của logic game tại thời điểm thực, ví dụ như sự khác biệt về tải xử lý giữa một nhân vật đứng yên và một nhân vật đang tham gia giao tranh. Hơn nữa, chi phí tính toán tại tầng điều phối của QPLB cao hơn so với các thuật toán tĩnh, có thể tạo ra áp lực xử lý khi quy mô hệ thống mở rộng lên hàng nghìn máy chủ.

Dựa trên những hạn chế kỹ thuật đã nhận diện, hướng phát triển tiếp theo của đề tài sẽ tập trung vào ba giải pháp trọng tâm nhằm hoàn thiện mô hình QPLB cho các hệ thống quy mô lớn. Đầu tiên, để khắc phục độ trễ do cơ chế thu thập dữ liệu theo chu kỳ, nghiên cứu sẽ chuyển dịch sang cơ chế đồng bộ hướng sự kiện, cho phép các máy chủ chủ động gửi báo cáo trạng thái ngay lập tức khi phát hiện biến động tài nguyên hoặc tải trọng đột ngột, kết hợp với các giao thức truyền tin tốc độ cao để đảm bảo dữ liệu tại bộ điều phối luôn cập nhật theo thời gian thực. Tiếp theo, nhằm nâng cao độ chính xác trong việc ước lượng năng lực phục vụ, đề tài sẽ tích hợp các mô hình trí tuệ nhân tạo và học máy để phân tích lịch sử vận hành, từ đó định lượng chính xác mối tương quan giữa ngữ cảnh logic game (như trạng thái giao tranh hay đứng yên) với tải trọng xử lý thực tế thay vì chỉ phụ thuộc vào chỉ số phần cứng. Cuối cùng, để giải quyết áp lực tính toán khi mở rộng quy mô, kiến trúc hệ thống sẽ được nâng cấp theo hướng phi tập trung và tự động hóa trên các nền tảng điện toán đám mây, cho phép tự động mở rộng năng lực xử lý song song và triển khai thuật toán tại các nút biên mạng nhằm loại bỏ điểm nghẽn cổ chai tại trung tâm điều phối.

TÀI LIỆU THAM KHẢO

[1] W. Feng, F. Chang, W. Feng, and J. Walpole, "A Traffic Characterization of Popular On-Line Games," *IEEE/ACM Transactions on Networking*, vol. 13, no. 3, pp. 488-500, June 2005.

[2] M. Claypool and D. Sanders, "Latency and jitter in first-person shooter games," in *Communication Systems and Networks*, 2006.

[3] J. Färber, "Network game traffic modelling," in Proceedings of the 1st workshop on Network and system support for games (NetGames '02), Braunschweig, Germany, 2002.

[4] M. S. Borella, "Source models of network game traffic," *Computer Communications*, vol. 23, no. 4, pp. 403-410, 2000.

[5] K.-C. Su et al., "Traffic characteristics of online games: A Review," *Journal of Network and Computer Applications*, 2009.

[6] G. Armitage, "Valve's Source Server Query Protocol," *RFC Draft*, 2008.

[7] T. Henderson and S. Bhatti, "Networked games: a QoS-sensitive application for QoS-insensitive networks," *ACM SIGCOMM Computer Communication Review*, 2003.

[8] Y. W. Bernier, "Latency compensating methods in client/server in-game simulations," *Valve Developer Community*, 2001.

[9] M. Dick et al., "Analysis of factors affecting players' performance and perception in multiplayer games," in *NetGames '05*, 2005.

[10] J. Gettys and K. Nichols, "Bufferbloat: Dark buffers in the Internet," *Communications of the ACM*, 2011.

[11] D. Pittman and C. GauthierDickey, "A measurement study of virtual populations in massively multiplayer online games," in *NetGames '07*, 2007.

[12] C. Chambers et al., "Measurement-based characterization of a collection of on-line games," in *Internet Measurement Conference (IMC)*, 2005.

[13] P. H. Chen, H. K. Wu, and C. N. Lee, "Analysis of Player Departure in MMORPGs," in *IEEE Transactions on Computational Intelligence and AI in Games*, 2012.

[14] W. Feng and W. Feng, "On the geographic distribution of on-line game servers and players," in *NetGames '03*, 2003.

[15] M. A. Hossain et al., "Server placement and player mapping for interactive online games," *Journal of Network and Computer Applications*, 2011.

[16] S. Agarwal and J. R. Lorch, "Matchmaking for Online Games and Other Latency-Sensitive P2P Systems," *SIGCOMM*, 2009.

[17] M. E. Mustafa and A. M. Ibrahim, "Load Balancing Algorithms Round-Robin (RR), Least-Connection and Least Loaded Efficiency," *International Journal of Computer and Information Technology*, vol. 4, no. 2, pp. 255-257, Mar. 2015.

[18] R. Guo, "Load Balancing for Interface Processors," in *Network Processors: Architecture, Programming, and Implementation*, Morgan Kaufmann, 2004.

[19] H. Mehta and E. Kanani, "A Review on Load Balancing Algorithms in Cloud Computing," *International Journal of Computer Applications*, vol. 143, no. 8, 2016.

[20] M. Dahlin, "Interpreting Stale Load Information," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 10, pp. 1033-1047, 2000.

[21] D. Gross, J. F. Shortle, J. M. Thompson, and C. M. Harris, *Fundamentals of Queueing Theory*, 4th ed. New Jersey: John Wiley & Sons, 2008.

[22] J. D. C. Little, "A Proof for the Queuing Formula: L = λW," *Operations Research*, vol. 9, no. 3, pp. 383-387, 1961.

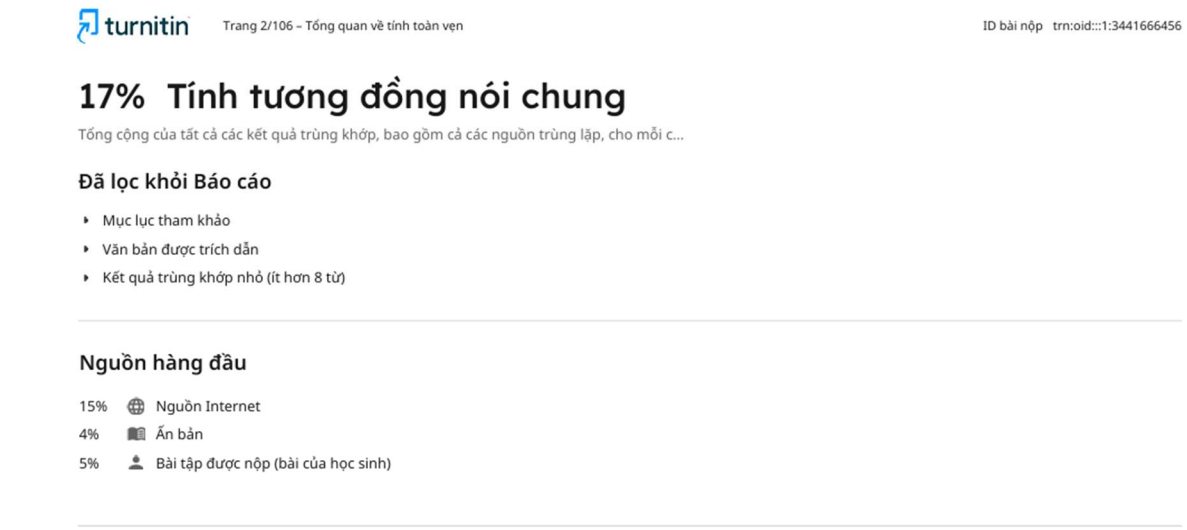
BẢN CAM ĐOAN

Tôi cam đoan đã thực hiện việc kiểm tra mức độ tương đồng nội dung đề án qua phần mềm Turnitin một cách trung thực và đạt kết quả mức độ tương đồng 17 % toàn bộ nội dung đề án. Bản đề án kiểm tra qua phần mềm là bản cứng đề án đã nộp để bảo vệ trước hội đồng. Nếu sai tôi xin chịu các hình thức kỷ luật theo quy định hiện hành của Học viện.

……., ngày …… tháng …… năm ……

HỌC VIÊN CAO HỌC

(Ký và ghi rõ họ tên)


turnitin Trang 2/106 – Tổng quan về tính toàn vẹn ID bài nộp trn:oid:::1:3441666456

17% Tính tương đồng nói chung

Tổng cộng của tất cả các kết quả trùng khớp, bao gồm cả các nguồn trùng lặp, cho mỗi c...

Đã lọc khỏi Báo cáo

- Mục lục tham khảo
- Văn bản được trích dẫn
- Kết quả trùng khớp nhỏ (ít hơn 8 từ)

Nguồn hàng đầu

15% Nguồn Internet
4% Ấn bản
5% Bài tập được nộp (bài của học sinh)

Học viên thực hiện	Giáo viên hướng dẫn
Nguyễn Đoàn Quân	**PGS. TS. Lê Hữu Lập**