TRƯỜNG ĐẠI HỌC SƯ PHẠM HÀ NỘI
KHOA CÔNG NGHỆ THÔNG TIN
---*---**

THỰC HÀNH DỰ ÁN

Đề tài: Nghiên cứu phương pháp xây dựng môi trường co giãn cho hệ thống có luồng dữ liệu lớn trên môi trường điện toán đám mây

Giảng viên hướng dẫn: **VŨ THÁI GIANG**
Nhóm sinh viên thực hiện:
715105097 - Lê Huy Hoàng
715105074 - Mai Lý Hải

Hà Nội, 2024

Mục Lục

MỞ ĐẦU

Tại Việt Nam, khi dân số ngày càng tăng nhanh, nhu cầu sử dụng các công nghệ thông minh và tiện ích số trong đời sống cũng không ngừng gia tăng. Sự phát triển mạnh mẽ của các thiết bị điện tử, từ điện thoại thông minh cho đến các thiết bị kết nối Internet vạn vật (IoT), đã góp phần làm cho khối lượng dữ liệu được tạo ra và lưu thông ngày càng lớn. Chính điều này dẫn đến sự bùng nổ của các dịch vụ và ứng dụng công nghệ, đòi hỏi một hệ thống hạ tầng mạng phải đủ mạnh để đáp ứng nhu cầu phát triển không ngừng. Tuy nhiên, thực tế cho thấy hạ tầng mạng hiện tại vẫn đang phải đối mặt với nhiều thách thức, đặc biệt là việc không thể theo kịp tốc độ phát triển nhanh chóng của công nghệ. Việc xử lý các luồng dữ liệu lớn từ các hệ thống công nghệ hiện đại như dữ liệu từ các ứng dụng trực tuyến, hệ thống IoT, hay thậm chí là các dịch vụ giải trí trực tuyến, đang đặt ra yêu cầu cấp thiết về việc phát triển các giải pháp công nghệ tiên tiến.

Một trong những thách thức lớn nhất hiện nay chính là khả năng xử lý và lưu trữ khối lượng dữ liệu khổng lồ một cách hiệu quả, nhất là khi các dữ liệu này ngày càng trở nên phức tạp và đa dạng hơn. Điều này đòi hỏi các hệ thống xử lý luồng dữ liệu cần có khả năng co giãn và linh hoạt, không chỉ về mặt tài nguyên tính toán mà còn về việc quản lý tài nguyên mạng, đảm bảo hệ thống có thể mở rộng theo nhu cầu mà không gây ảnh hưởng đến hiệu suất. Môi trường điện toán đám mây, với ưu điểm nổi bật về sự linh động, khả năng mở rộng và tiết kiệm chi phí, đang trở thành lựa chọn hàng đầu cho việc xây dựng các hệ thống này. Tuy nhiên, để có thể tận dụng hết tiềm năng của điện toán đám mây, cần phải phát triển các giải pháp tối ưu hóa trong việc sử dụng hạ tầng mạng, tài nguyên tính toán và lưu trữ.

Trong đề tài nghiên cứu này, chúng tôi sẽ tập trung vào việc khảo sát và phát triển các phương pháp xây dựng môi trường co giãn cho hệ thống xử lý luồng dữ liệu lớn dựa trên nền tảng điện toán đám mây. Nghiên cứu sẽ không chỉ dừng lại ở việc tìm hiểu các kỹ thuật và thuật toán tối ưu hóa để cải thiện hiệu suất hệ thống, mà còn xem xét các vấn đề như khả năng bảo mật, độ tin cậy và hiệu quả sử dụng tài nguyên. Đặc biệt, nghiên cứu cũng sẽ khai thác tiềm năng của các thuật toán nén dữ liệu tiên tiến như Deflate, LZMA, LZ4, và Bzip2. Những thuật toán này có thể giúp tối ưu hóa quá trình truyền tải và lưu trữ dữ liệu trên đám mây, từ đó đảm bảo rằng hệ thống có khả năng mở rộng và thích ứng linh hoạt với sự gia tăng không ngừng của khối lượng dữ liệu. Nghiên cứu này không chỉ góp phần nâng cao hiệu quả sử dụng tài nguyên mạng, mà còn mở ra những hướng đi mới trong việc phát triển các hệ thống xử lý dữ liệu thông minh, bền vững trong tương lai.

A. TỔNG QUAN VỀ DỰ ÁN

1. Mục tiêu và phạm vi dự án

Mục tiêu dự án

Mục đích nghiên cứu của dự án này là khám phá và phát triển các phương pháp tối ưu nhằm xây dựng một môi trường co giãn (scalable) cho hệ thống xử lý và truyền tải luồng dữ liệu lớn trên nền tảng điện toán đám mây. Trong bối cảnh công nghệ số đang phát triển với tốc độ nhanh chóng, các ứng dụng công nghệ cao ngày càng chiếm lĩnh cuộc sống hiện đại. Điều này kéo theo nhu cầu xử lý và phân tích khối lượng dữ liệu khổng lồ từ nhiều nguồn khác nhau như các thiết bị IoT, hệ thống truyền thông, và mạng xã hội. Các nguồn dữ liệu này không ngừng gia tăng cả về quy mô lẫn độ phức tạp, tạo ra một thách thức lớn đối với các hệ thống công nghệ hiện đại khi phải xử lý luồng dữ liệu liên tục với yêu cầu cao về tốc độ và độ chính xác. Trong bối cảnh đó, điện toán đám mây đã nổi lên như một giải pháp tiềm năng, cung cấp tài nguyên tính toán, lưu trữ linh hoạt và khả năng mở rộng gần như vô hạn, tạo điều kiện thuận lợi cho các hệ thống công nghệ đáp ứng được nhu cầu xử lý dữ liệu lớn một cách hiệu quả và tối ưu.

Mục tiêu chính của nghiên cứu này là xây dựng và đánh giá các phương pháp giúp hệ thống có khả năng tự động mở rộng (scale out) hoặc thu hẹp (scale in) tài nguyên dựa trên nhu cầu xử lý dữ liệu thực tế, từ đó đảm bảo rằng hệ thống luôn hoạt động ổn định với hiệu suất cao nhất mà không làm lãng phí tài nguyên. Khả năng co giãn linh hoạt sẽ giúp hệ thống xử lý dữ liệu có thể dễ dàng thích nghi với sự biến động về khối lượng công việc, đảm bảo rằng trong thời điểm yêu cầu xử lý tăng cao, hệ thống sẽ tự động mở rộng quy mô để đáp ứng nhu cầu, và khi khối lượng công việc giảm, hệ thống sẽ tự động thu hẹp để tiết kiệm chi phí. Điều này đặc biệt có ý nghĩa đối với các hệ thống phân tán quy mô lớn, nơi mà việc tối ưu hóa tài nguyên tính toán và lưu trữ trở thành một thách thức quan trọng.

Thông qua dự án này, chúng tôi sẽ tập trung nghiên cứu các khía cạnh kỹ thuật sau để đạt được mục tiêu:

- Xây dựng kiến trúc hệ thống co giãn: Trong quá trình nghiên cứu, chúng tôi sẽ khám phá và phát triển các mô hình kiến trúc hiện đại như kiến trúc microservices và serverless trên các nền tảng điện toán đám mây phổ biến như AWS, Google Cloud, và Microsoft Azure. Các mô hình này không chỉ mang lại tính linh hoạt trong việc triển khai và quản lý ứng dụng mà còn giúp hệ thống tự động điều chỉnh quy mô theo khối lượng công việc thực tế. Cụ thể, chúng tôi sẽ phát triển một kiến trúc hệ thống có khả năng tự động mở rộng hoặc thu hẹp dựa trên nhu cầu thực tế về xử lý dữ liệu. Việc này đảm bảo rằng

hệ thống luôn hoạt động hiệu quả và linh hoạt mà không lãng phí tài nguyên không cần thiết, đồng thời nâng cao khả năng xử lý và truyền tải luồng dữ liệu lớn một cách hiệu quả.

- Quản lý luồng dữ liệu lớn: Nghiên cứu sẽ tìm hiểu và áp dụng các công nghệ tiên tiến nhất hiện nay trong việc quản lý và xử lý luồng dữ liệu lớn. Đặc biệt, chúng tôi sẽ khai thác tiềm năng của các công cụ và framework như Apache Kafka, Apache Flink, và nhiều công nghệ tương tự, giúp hệ thống có thể xử lý luồng dữ liệu trong thời gian thực với hiệu suất cao. Quản lý luồng dữ liệu lớn là một yếu tố then chốt trong hệ thống xử lý hiện đại, nhất là trong bối cảnh dữ liệu không ngừng thay đổi và tăng trưởng nhanh chóng. Việc áp dụng các công nghệ mới sẽ giúp hệ thống duy trì được hiệu suất và độ chính xác cao trong khi xử lý dữ liệu lớn từ nhiều nguồn khác nhau.

- Tối ưu hóa tài nguyên đám mây: Một phần quan trọng của nghiên cứu là đánh giá hiệu suất và chi phí của các giải pháp co giãn trên môi trường điện toán đám mây. Chúng tôi sẽ nghiên cứu và phát triển các chiến lược tối ưu hóa việc sử dụng tài nguyên, bao gồm autoscaling (tự động mở rộng), load balancing (cân bằng tải), và các kỹ thuật tối ưu khác để đảm bảo rằng các tài nguyên tính toán, bộ nhớ và lưu trữ được sử dụng một cách hiệu quả nhất. Việc này không chỉ giúp cải thiện hiệu suất của hệ thống mà còn góp phần giảm chi phí vận hành, đặc biệt là khi áp dụng trong các môi trường phân tán lớn, nơi mà tài nguyên mạng và tính toán đóng vai trò then chốt.

- Bảo mật và độ tin cậy: Để đảm bảo rằng hệ thống hoạt động ổn định và an toàn trên nền tảng điện toán đám mây, chúng tôi sẽ nghiên cứu và phát triển các biện pháp bảo mật và quản lý rủi ro. Đặc biệt, các biện pháp bảo vệ dữ liệu, kiểm soát truy cập, và khôi phục dữ liệu trong trường hợp xảy ra sự cố sẽ được triển khai nhằm đảm bảo hệ thống luôn duy trì được tính bảo mật và độ tin cậy cao, ngay cả trong những điều kiện hoạt động phức tạp. Các giải pháp này sẽ không chỉ tập trung vào việc bảo vệ dữ liệu mà còn giúp hệ thống chống lại các mối đe dọa an ninh mạng ngày càng tinh vi.

- Ứng dụng trong thực tiễn: Cuối cùng, nghiên cứu sẽ đề xuất các phương pháp và giải pháp cụ thể để áp dụng vào các lĩnh vực như tài chính, chăm sóc sức khỏe, bán lẻ, và nhiều lĩnh vực khác. Trong các lĩnh vực này, việc xử lý và phân tích dữ liệu lớn không chỉ là

yêu cầu cấp thiết mà còn có tiềm năng mang lại những giá trị kinh tế to lớn. Các ứng dụng thực tiễn từ nghiên cứu sẽ giúp minh họa cách mà hệ thống co giãn có thể được triển khai và mang lại hiệu quả vượt trội trong thực tế.

Kết quả của nghiên cứu này không chỉ đóng góp vào việc phát triển cơ sở lý thuyết cho các hệ thống xử lý dữ liệu lớn mà còn cung cấp các giải pháp thực tiễn, giúp các tổ chức và doanh nghiệp tối ưu hóa tài nguyên và chi phí trên nền tảng điện toán đám mây. Nhờ vào khả năng co giãn linh hoạt và hiệu quả cao, các hệ thống này sẽ giúp các tổ chức đối phó với những thách thức của kỷ nguyên dữ liệu lớn, đồng thời tận dụng được tiềm năng mà công nghệ đám mây mang lại trong việc quản lý và phân tích dữ liệu.

Phạm vi dự án

Phạm vi của dự án này được xác định rõ ràng trong việc nghiên cứu và triển khai các giải pháp xây dựng hệ thống co giãn trên nền tảng điện toán đám mây, đặc biệt trong việc xử lý luồng dữ liệu lớn (Big Data). Cụ thể, phạm vi dự án sẽ bao gồm các khía cạnh chính sau:

- Nền tảng điện toán đám mây: Dự án sẽ tập trung vào các nền tảng điện toán đám mây phổ biến như Amazon Web Services (AWS), Microsoft Azure, và Google Cloud Platform (GCP). Các tính năng cơ bản của điện toán đám mây như tính năng tự động mở rộng (autoscaling), cân bằng tải (load balancing), và quản lý tài nguyên linh hoạt sẽ là trọng tâm của nghiên cứu.

- Công nghệ xử lý dữ liệu lớn: Phạm vi nghiên cứu sẽ bao gồm các công cụ và nền tảng xử lý dữ liệu lớn như Apache Kafka, Apache Flink, Apache Spark, và các hệ thống tương tự khác có khả năng xử lý luồng dữ liệu liên tục và khối lượng lớn. Việc tích hợp các công nghệ này với các nền tảng đám mây sẽ được xem xét để đảm bảo khả năng mở rộng và tính hiệu quả của hệ thống.

- Kiến trúc hệ thống co giãn: Dự án sẽ xây dựng và thử nghiệm kiến trúc microservices hoặc serverless để đạt được tính linh hoạt và khả năng mở rộng theo nhu cầu. Chúng tôi sẽ tập trung vào các giải pháp có khả năng mở rộng tự động dựa trên khối lượng công việc, đồng thời nghiên cứu cách quản lý dữ liệu một cách hiệu quả.

– Bảo mật và quản lý dữ liệu: Dự án sẽ xem xét các giải pháp bảo mật khi triển khai hệ thống trên môi trường đám mây, bao gồm mã hóa dữ liệu, kiểm soát truy cập, và bảo vệ dữ liệu người dùng. Các yêu cầu pháp lý về bảo mật dữ liệu và quyền riêng tư sẽ được xem xét trong việc lưu trữ và xử lý dữ liệu lớn trên môi trường đám mây.

- Tối ưu hóa chi phí và tài nguyên: Phạm vi dự án bao gồm việc phân tích và tối ưu hóa việc sử dụng tài nguyên đám mây như bộ nhớ, CPU, và lưu trữ. Dự án sẽ nghiên cứu các mô hình thanh toán trên đám mây và đề xuất giải pháp tiết kiệm chi phí mà không làm giảm hiệu suất của hệ thống.

Thử nghiệm và đánh giá: Các thử nghiệm sẽ được thực hiện trong môi trường giả lập để đánh giá khả năng co giãn của hệ thống. Chúng tôi sẽ đo lường hiệu suất hệ thống thông qua các chỉ số như tốc độ xử lý dữ liệu, độ trễ, và khả năng phục hồi khi hệ thống gặp sự cố. Dự án cũng sẽ đưa ra các bài kiểm tra hiệu suất dưới điều kiện tải cao để đánh giá khả năng đáp ứng của hệ thống.

Ứng dụng trong thực tế: Cuối cùng, dự án sẽ giới hạn phạm vi vào một số ngành công nghiệp cụ thể như tài chính, bán lẻ, hoặc chăm sóc sức khỏe, nơi mà nhu cầu xử lý dữ liệu lớn và mở rộng hệ thống là cần thiết. Điều này nhằm minh họa khả năng áp dụng thực tế của các giải pháp và kiến trúc được phát triển trong dự án.

Tổng hợp lại, phạm vi của dự án bao gồm việc nghiên cứu từ lý thuyết đến thực nghiệm các giải pháp xử lý dữ liệu lớn trên nền tảng điện toán đám mây, tập trung vào tính co giãn, bảo mật, tối ưu hóa chi phí, và hiệu suất hệ thống trong các điều kiện thực tế.

Ngữ cảnh hệ thống

Ngữ cảnh hệ thống của dự án này được đặt trong bối cảnh các hệ thống xử lý dữ liệu lớn hiện đại, hoạt động trên nền tảng điện toán đám mây, nhằm giải quyết các thách thức về hiệu suất cao, khả năng mở rộng linh hoạt, và đáp ứng nhu cầu xử lý khối lượng dữ liệu không ngừng gia tăng. Các yếu tố chính của ngữ cảnh này bao gồm:

- Hệ thống xử lý dữ liệu lớn: Hệ thống phải hoạt động trong một môi trường nơi dữ liệu được thu thập liên tục từ nhiều nguồn khác nhau. Các nguồn dữ liệu phổ biến bao gồm thiết bị IoT (Internet of Things), mạng xã hội,

ứng dụng thương mại điện tử, hệ thống chăm sóc sức khỏe, và dịch vụ tài chính. Đặc điểm chung của các loại dữ liệu này là khối lượng và tốc độ thay đổi liên tục, đặc biệt là trong các đợt đột biến về lưu lượng dữ liệu. Điều này đòi hỏi hệ thống phải có khả năng co giãn tốt, tức là mở rộng hoặc thu hẹp tài nguyên một cách linh hoạt để đáp ứng sự biến động về khối lượng công việc. Bất kỳ sự chậm trễ nào trong quá trình xử lý có thể dẫn đến mất dữ liệu hoặc gián đoạn dịch vụ, do đó khả năng xử lý dữ liệu theo thời gian thực và hiệu suất cao là yếu tố then chốt.

- Môi trường điện toán đám mây: Hệ thống được triển khai trên các nền tảng điện toán đám mây hàng đầu như Amazon Web Services (AWS), Google Cloud, hoặc Microsoft Azure. Các nền tảng này cung cấp tài nguyên tính toán, lưu trữ và quản lý cơ sở dữ liệu với khả năng mở rộng linh hoạt, cho phép hệ thống điều chỉnh quy mô theo nhu cầu sử dụng. Bằng cách tận dụng các tính năng như autoscaling (tự động mở rộng), load balancing (cân bằng tải), và cơ chế quản lý tài nguyên theo nhu cầu, hệ thống có thể đảm bảo rằng các tài nguyên đám mây được sử dụng hiệu quả, không lãng phí, đồng thời đáp ứng được những thay đổi đột ngột trong khối lượng dữ liệu.

- Xử lý luồng dữ liệu thời gian thực: Một trong những yêu cầu quan trọng nhất của hệ thống là khả năng xử lý luồng dữ liệu theo thời gian thực. Việc sử dụng các công nghệ như Apache Kafka và Apache Flink giúp hệ thống có thể xử lý và phân tích dữ liệu ngay khi dữ liệu được thu thập. Điều này đặc biệt quan trọng trong các ứng dụng đòi hỏi phản hồi tức thời, chẳng hạn như hệ thống giám sát tài chính nhằm phát hiện gian lận, hoặc điều khiển thiết bị IoT trong các quy trình công nghiệp. Việc xử lý luồng dữ liệu thời gian thực không chỉ nâng cao khả năng phản ứng nhanh của hệ thống mà còn giúp đưa ra các quyết định nhanh chóng và chính xác, từ đó tối ưu hóa hoạt động kinh doanh.

- Cấu trúc microservices và serverless: Hệ thống sẽ được xây dựng dựa trên kiến trúc microservices hoặc serverless. Với kiến trúc microservices, mỗi thành phần của hệ thống chịu trách nhiệm cho một chức năng cụ thể, như thu thập dữ liệu, xử lý, lưu trữ, phân tích, và hiển thị kết quả. Việc phân chia chức năng thành các microservices giúp hệ thống có thể dễ dàng bảo trì, nâng cấp, và mở rộng từng phần riêng lẻ mà không ảnh hưởng đến toàn bộ hệ thống. Kiến trúc serverless, ngược lại, cho phép hệ thống không cần quản lý hạ tầng máy chủ, mà chỉ cần quan tâm đến logic xử lý, từ đó giảm thiểu chi phí và tăng cường tính linh hoạt trong việc mở rộng quy mô.

- Yêu cầu về hiệu suất cao và tính sẵn sàng: Để đảm bảo rằng hệ thống có thể đáp ứng nhu cầu xử lý dữ liệu lớn một cách liên tục và ổn định, hệ thống phải đảm bảo hiệu suất cao và độ trễ thấp ngay cả khi khối lượng dữ liệu đột ngột tăng mạnh. Điều này đòi hỏi hệ thống phải hỗ trợ xử lý song song và phân phối khối lượng công việc giữa các tài nguyên tính toán một cách hiệu quả. Đồng thời, tính sẵn sàng của hệ thống phải cao, đảm bảo rằng hệ thống không bị gián đoạn khi xảy ra sự cố, thông qua các cơ chế như dự phòng dữ liệu và khôi phục từ xa.

- Bảo mật và tuân thủ: Một yếu tố quan trọng khác trong ngữ cảnh này là bảo mật dữ liệu. Hệ thống phải xử lý các loại dữ liệu nhạy cảm, như dữ liệu tài chính, thông tin cá nhân, do đó, yêu cầu về bảo mật rất cao. Hệ thống cần áp dụng các biện pháp mã hóa dữ liệu trong quá trình truyền tải và lưu trữ, kiểm soát truy cập nghiêm ngặt thông qua cơ chế phân quyền người dùng, và tuân thủ các quy định bảo vệ dữ liệu như GDPR (General Data Protection Regulation) của Châu Âu và HIPAA (Health Insurance Portability and Accountability Act) của Hoa Kỳ. Việc tuân thủ các quy định này không chỉ đảm bảo sự an toàn của dữ liệu mà còn bảo vệ hệ thống trước các mối đe dọa từ bên ngoài.

- Tối ưu hóa tài nguyên và chi phí: Một trong những mục tiêu chính của hệ thống là tối ưu hóa việc sử dụng tài nguyên trên môi trường đám mây, từ đó giảm thiểu chi phí vận hành. Hệ thống sẽ sử dụng các chiến lược tối ưu tài nguyên, như chỉ kích hoạt tài nguyên khi cần thiết (on-demand), tự động giảm quy mô khi khối lượng công việc giảm, và tận dụng các dịch vụ serverless để tránh việc phải duy trì các hạ tầng cố định. Điều này giúp hệ thống vừa đảm bảo hiệu năng hoạt động, vừa tối ưu hóa được chi phí, nhất là khi triển khai ở quy mô lớn.

- Tích hợp và mở rộng: Cuối cùng, hệ thống phải có khả năng tích hợp dễ dàng với các hệ thống khác hoặc mở rộng để bổ sung các chức năng mới. Ví dụ, hệ thống có thể tích hợp với các nền tảng phân tích dữ liệu nâng cao hoặc các công cụ AI để dự đoán xu hướng, phân tích dữ liệu chuyên sâu, hoặc tự động hóa các quyết định kinh doanh. Điều này không chỉ giúp nâng cao giá trị mà còn tạo điều kiện cho việc mở rộng hệ thống trong tương lai mà không gặp phải giới hạn kỹ thuật.

Tổng quan, ngữ cảnh của hệ thống này thể hiện rõ nhu cầu xử lý dữ liệu lớn, sự linh hoạt trong việc mở rộng, và các yêu cầu khắt khe về bảo mật, hiệu suất và tối ưu hóa chi phí khi triển khai trên các nền tảng điện toán đám mây hiện đại. Hệ thống này sẽ đóng vai trò quan trọng trong việc đáp ứng nhu cầu

xử lý dữ liệu của nhiều ngành công nghiệp khác nhau, từ tài chính, chăm sóc sức khỏe, đến thương mại điện tử và mạng xã hội.

2. *Kế hoạch xây dựng và phát triển*

Các ràng buộc dự án

- Ràng buộc về tài nguyên hạ tầng
    - Tài nguyên tính toán: Hệ thống cần quản lý và phân bổ tài nguyên như số lượng máy chủ (instances), CPU, và bộ nhớ một cách tối ưu. Các nền tảng đám mây như AWS, Google Cloud, và Azure có giới hạn về số lượng tài nguyên được phép sử dụng dựa trên gói dịch vụ. Việc triển khai hệ thống phải đảm bảo không vượt quá tài nguyên được cấp phát, đồng thời tối ưu để tránh chi phí vận hành tăng quá mức. Ví dụ: chỉ mở rộng tài nguyên khi cần thiết và giải phóng khi nhu cầu giảm xuống.

    - Băng thông mạng: Khả năng truyền tải dữ liệu của hệ thống bị giới hạn bởi tốc độ mạng. Trong bối cảnh xử lý dữ liệu thời gian thực, việc truyền tải dữ liệu lớn có thể gây tắc nghẽn và ảnh hưởng đến hiệu suất hệ thống. Việc tối ưu hóa lưu lượng mạng thông qua nén dữ liệu hoặc chọn kênh truyền thông hiệu quả là cần thiết để đảm bảo tính liên tục.

    - Lưu trữ: Lưu trữ dữ liệu lớn yêu cầu dung lượng lưu trữ cao và chi phí tương ứng. Các dịch vụ lưu trữ đám mây như S3 của AWS có chi phí tăng dần theo dung lượng sử dụng. Việc tối ưu hóa dung lượng lưu trữ bằng cách áp dụng chiến lược lưu trữ dữ liệu lâu dài với chi phí thấp (cold storage) hoặc xóa dữ liệu không cần thiết là điều cần thiết để tiết kiệm chi phí.

- Ràng buộc về hiệu suất
    - Thời gian phản hồi: Hệ thống cần xử lý dữ liệu theo thời gian thực hoặc gần thời gian thực với độ trễ thấp nhất. Điều này yêu cầu việc tối ưu hóa từ việc xử lý dữ liệu đến phân bổ tài nguyên. Ví dụ: sử dụng các thuật toán xử lý dữ liệu phân tán hoặc song song để tăng tốc độ xử lý, đồng thời đảm bảo rằng không có tắc nghẽn ở bất kỳ điểm nào trong quy trình.

- Hiệu suất mở rộng (Scalability): Hệ thống phải có khả năng mở rộng nhanh chóng khi khối lượng dữ liệu tăng mà không ảnh hưởng đến hiệu suất. Điều này yêu cầu các thuật toán và kiến trúc hệ thống phải hỗ trợ co giãn tài nguyên đồng bộ, như tự động mở rộng số lượng máy chủ khi có nhu cầu và thu hẹp khi khối lượng công việc giảm.

- Ràng buộc về thuật toán và kỹ thuật nén
  - Chọn thuật toán nén phù hợp: Các thuật toán nén như Deflate, LZMA, LZ4, và Bzip2 có những ưu và nhược điểm khác nhau về tốc độ và mức độ nén. Lựa chọn thuật toán phải phù hợp với yêu cầu của hệ thống về tốc độ xử lý, khả năng truyền tải dữ liệu và lưu trữ. Chẳng hạn, LZ4 có tốc độ nén cao nhưng mức độ nén thấp, trong khi Bzip2 có mức độ nén cao nhưng tiêu tốn nhiều tài nguyên hơn.

  - Mức độ nén và tài nguyên: Quá trình nén và giải nén dữ liệu đòi hỏi tài nguyên CPU và bộ nhớ, điều này cần phải được cân bằng để không làm giảm hiệu suất chung của hệ thống. Hệ thống cần điều chỉnh mức độ nén sao cho phù hợp với khả năng xử lý tài nguyên hiện có, đặc biệt trong các tình huống yêu cầu tốc độ xử lý cao.

- Ràng buộc về chi phí
  - Chi phí sử dụng tài nguyên đám mây: Sử dụng đám mây đi kèm với chi phí dựa trên mức độ tiêu thụ tài nguyên, như tài nguyên tính toán, lưu trữ, và băng thông. Chi phí có thể tăng nhanh nếu không được tối ưu hóa, vì vậy hệ thống cần áp dụng các chiến lược như autoscaling để sử dụng tài nguyên hiệu quả và chỉ mở rộng khi cần thiết.
  - Chi phí lưu trữ và truyền tải dữ liệu: Lưu trữ và truyền tải dữ liệu lớn có thể phát sinh chi phí đáng kể. Cần tối ưu hóa bằng cách nén dữ liệu trước khi truyền tải hoặc sử dụng các dịch vụ lưu trữ chi phí thấp cho các dữ liệu ít truy cập (cold storage). Đồng thời, sử dụng các kỹ thuật như "chunking" (chia nhỏ dữ liệu) để truyền tải theo từng phần nhỏ, giảm bớt băng thông cần thiết.

- Ràng buộc về quản lý và giám sát

- Giám sát tài nguyên: Hệ thống cần cơ chế giám sát hiệu quả để theo dõi việc sử dụng tài nguyên và phát hiện sớm các vấn đề như tắc nghẽn, tài nguyên quá tải hoặc hiệu suất giảm. Các dịch vụ như AWS CloudWatch, Google Stackdriver có thể giúp giám sát tự động và gửi cảnh báo khi cần thiết.

- Tự động hóa trong quản lý: Tự động hóa là yếu tố quan trọng trong môi trường xử lý dữ liệu lớn, đặc biệt khi hệ thống phải co giãn liên tục. Hệ thống cần có khả năng tự động quản lý và phân bổ tài nguyên, tự động mở rộng khi nhu cầu tăng và giảm khi lưu lượng giảm mà không cần sự can thiệp của con người. Điều này không chỉ giúp hệ thống hoạt động hiệu quả hơn mà còn giảm thiểu rủi ro từ lỗi con người.

- Kết luận:
  - Các ràng buộc này đóng vai trò quan trọng trong việc định hình cách xây dựng và vận hành hệ thống xử lý luồng dữ liệu lớn trên nền tảng điện toán đám mây. Cần có sự cân nhắc và tối ưu hóa từng yếu tố để đảm bảo hiệu suất, khả năng mở rộng, và tối thiểu hóa chi phí trong suốt quá trình triển khai và vận hành hệ thống.

Kế hoạch triển khai dự án

Kế hoạch triển khai dự án bao gồm các giai đoạn từ thiết lập môi trường đám mây, phát triển hệ thống xử lý dữ liệu lớn, đến kiểm thử và tối ưu hóa. Dưới đây là mô tả chi tiết từng bước:

- Thiết lập môi trường đám mây
  - Cấu hình môi trường đám mây:
    - Chọn nền tảng đám mây phù hợp (AWS, Google Cloud, hoặc Azure) dựa trên yêu cầu về tài nguyên, dịch vụ hỗ trợ và chi phí.
    - Thiết lập tài khoản, cấu hình các vùng (regions), và triển khai các dịch vụ cơ bản như máy chủ ảo (EC2/Azure VM/Google Compute Engine), các dịch vụ lưu trữ (S3/Google Cloud Storage/Azure Blob Storage).

        - Cấu hình mạng ảo (VPC, Subnets), bảo mật (Security Groups, IAM Roles), và các dịch vụ mạng (Load Balancers, Auto Scaling Groups).
    - Cấu hình dịch vụ lưu trữ và cơ sở dữ liệu:
        - Chọn và cấu hình cơ sở dữ liệu phù hợp (RDS, DynamoDB, BigQuery, hoặc Cosmos DB).
        - Thiết lập các dịch vụ lưu trữ dữ liệu lớn (S3, GCS, Blob Storage) để lưu trữ dữ liệu đầu vào/ra hoặc kết quả xử lý.
        - Đảm bảo các yếu tố bảo mật như mã hóa dữ liệu, kiểm soát truy cập và tuân thủ các quy định bảo mật như GDPR.
- Phát triển hệ thống xử lý dữ liệu
    - Sử dụng Apache Kafka hoặc Spark:
        - Thiết lập và triển khai các hệ thống Apache Kafka hoặc Apache Spark để xử lý dữ liệu luồng (streaming data).
        - Phát triển pipeline xử lý dữ liệu cho các luồng dữ liệu thời gian thực từ nhiều nguồn như IoT, hệ thống thương mại điện tử, hay ứng dụng tài chính.
        - Thiết lập cơ chế đọc và ghi dữ liệu từ Kafka hoặc Spark sang dịch vụ lưu trữ đám mây hoặc cơ sở dữ liệu.
    - Tích hợp thuật toán nén:
        - Chọn các thuật toán nén như Deflate, LZMA, LZ4, và Bzip2 phù hợp với khối lượng và tốc độ dữ liệu.
        - Tích hợp các thuật toán này vào pipeline truyền tải và lưu trữ dữ liệu.
        - Đảm bảo rằng quá trình nén không ảnh hưởng đáng kể đến hiệu suất xử lý.
- Phát triển cơ chế co giãn tự động
    - Xây dựng cơ chế Auto Scaling:
        - Sử dụng Kubernetes (K8s) hoặc Docker Swarm để triển khai các dịch vụ microservices trong container.

        - Cấu hình Auto Scaling dựa trên các thông số như CPU, RAM hoặc lưu lượng công việc. Tận dụng Horizontal Pod Autoscaling (HPA) trong Kubernetes để tự động mở rộng hoặc giảm bớt số lượng pods dựa trên tải công việc thực tế.
        - Cấu hình Load Balancer để đảm bảo tải được phân bổ đồng đều giữa các instance hoặc container.
    - Phát triển công cụ quản lý và giám sát:
        - Tích hợp các công cụ như AWS CloudWatch, Google Cloud Monitoring, hoặc Azure Monitor để giám sát việc sử dụng tài nguyên.
        - Cài đặt Prometheus và Grafana để giám sát và hiển thị thông tin về tình trạng hệ thống theo thời gian thực.
        - Cấu hình các cảnh báo (alert) khi có sự cố như tài nguyên quá tải, hiệu suất giảm, hoặc sự cố mạng.
- Kiểm thử tính năng co giãn
    - Thực hiện stress test:
        - Sử dụng các công cụ kiểm thử tải như Apache JMeter hoặc Locust để thực hiện các bài kiểm tra tải (stress test) trên hệ thống.
        - Kiểm thử khả năng co giãn của hệ thống khi có đột biến về lưu lượng truy cập hoặc dữ liệu đầu vào lớn, đánh giá xem hệ thống có thể tự động mở rộng tài nguyên đúng lúc và giảm tài nguyên khi nhu cầu giảm.
    - Kiểm thử nén và giải nén dữ liệu:
        - Chạy thử các thuật toán nén/giải nén trên một khối lượng dữ liệu lớn để đảm bảo chúng hoạt động mượt mà, hiệu quả.
        - Đánh giá thời gian nén và giải nén, mức độ nén đạt được, và ảnh hưởng của quá trình này đến hệ thống (tài nguyên CPU, bộ nhớ).

Kết quả mong đợi

- Hệ thống ổn định: Môi trường đám mây được cấu hình đúng cách với các dịch vụ mở rộng linh hoạt, đảm bảo khả năng co giãn và xử lý khối lượng dữ liệu lớn.
- Hiệu suất cao: Hệ thống xử lý luồng dữ liệu lớn với độ trễ thấp, khả năng đáp ứng nhanh ngay cả khi khối lượng công việc tăng đột ngột.
- Tối ưu chi phí: Tài nguyên được phân bổ và sử dụng hợp lý nhờ cơ chế Auto Scaling và giám sát chặt chẽ, giúp tối ưu hóa chi phí.
- Đảm bảo bảo mật và tuân thủ: Dữ liệu được bảo vệ và quản lý theo các quy định bảo mật hiện hành như GDPR.
- Việc thực hiện đầy đủ các bước này sẽ giúp triển khai hệ thống một cách hiệu quả và đảm bảo tính linh hoạt, khả năng mở rộng trong tương lai.

B. CƠ SỞ NGHIÊN CỨU

1. Cơ sở lý thuyết

Điện toán đám mây (Cloud Computing)

Điện toán đám mây (Cloud Computing) là một mô hình cung cấp tài nguyên máy tính qua internet, cho phép người dùng truy cập và sử dụng các dịch vụ lưu trữ, xử lý và mạng mà không cần phải đầu tư vào phần cứng hoặc phần mềm tại chỗ. Mô hình này đã trở thành một phần không thể thiếu trong chiến lược công nghệ của nhiều tổ chức và doanh nghiệp, nhờ vào những lợi ích vượt trội mà nó mang lại. Dưới đây là một số khái niệm cơ bản và đặc điểm chính của điện toán đám mây.

Đặc điểm của điện toán đám mây:

- Khả năng mở rộng linh hoạt:

  Một trong những lợi ích lớn nhất của điện toán đám mây là khả năng mở rộng tài nguyên một cách linh hoạt. Người dùng có thể dễ dàng tăng hoặc giảm tài nguyên như CPU, bộ nhớ và dung lượng lưu trữ tùy theo nhu cầu thực tế. Điều này đặc biệt hữu ích trong các tình huống mà khối lượng công việc có sự thay đổi đột

ngột, chẳng hạn như khi có một sự kiện lớn diễn ra hoặc khi tổ chức cần xử lý một lượng dữ liệu lớn trong thời gian ngắn.

- Truy cập qua mạng:

  Tất cả các tài nguyên trong điện toán đám mây được cung cấp thông qua internet, cho phép người dùng truy cập từ bất kỳ đâu và từ nhiều loại thiết bị khác nhau, bao gồm máy tính để bàn, laptop, điện thoại thông minh và máy tính bảng. Điều này giúp tăng tính linh hoạt cho người dùng, cho phép họ làm việc từ xa hoặc di động mà không bị ràng buộc bởi vị trí vật lý.

- Tự động hóa việc cung cấp và thu hồi tài nguyên:

  Các dịch vụ điện toán đám mây cho phép tự động cung cấp và thu hồi tài nguyên một cách nhanh chóng và hiệu quả. Khi nhu cầu tăng lên, hệ thống có thể tự động mở rộng tài nguyên mà không cần sự can thiệp của con người. Ngược lại, khi nhu cầu giảm xuống, hệ thống cũng có thể thu hồi tài nguyên để tiết kiệm chi phí. Điều này giúp tổ chức tối ưu hóa chi phí và tài nguyên.

- Chi phí theo mức sử dụng:

  Điện toán đám mây hoạt động theo mô hình "pay-as-you-go", nghĩa là người dùng chỉ phải trả tiền cho những tài nguyên mà họ sử dụng thực sự. Điều này giúp tổ chức tiết kiệm chi phí và giảm thiểu rủi ro đầu tư vào cơ sở hạ tầng mà có thể không sử dụng hết.

- Bảo mật và quản lý dữ liệu:

  Mặc dù có nhiều lo ngại về bảo mật khi sử dụng dịch vụ điện toán đám mây, các nhà cung cấp dịch vụ lớn thường có các biện pháp bảo vệ dữ liệu mạnh mẽ, bao gồm mã hóa dữ liệu, kiểm soát truy cập và các giải pháp bảo mật khác. Họ cũng cung cấp khả năng sao lưu và phục hồi dữ liệu, giúp người dùng giảm thiểu nguy cơ mất mát dữ liệu.

- Dịch vụ đa dạng:

  Điện toán đám mây cung cấp nhiều loại dịch vụ khác nhau, bao gồm dịch vụ hạ tầng (IaaS), nền tảng (PaaS), và phần mềm (SaaS). Mỗi loại dịch vụ mang đến những lợi ích riêng biệt, cho phép tổ chức lựa chọn giải pháp phù hợp với nhu cầu và mục tiêu kinh doanh của họ.

Kết luận:

Điện toán đám mây không chỉ đơn thuần là một công nghệ; nó là một phương pháp tiếp cận mới trong việc quản lý và cung cấp tài nguyên công nghệ thông tin. Với khả năng mở rộng linh hoạt, chi phí tối ưu, và tính khả dụng cao, điện toán đám mây đã và đang giúp các tổ chức hiện đại tăng cường hiệu suất làm việc, nâng cao khả năng cạnh tranh và đáp ứng nhanh chóng các thay đổi trong nhu cầu của thị trường. Bằng cách tận dụng những lợi ích của điện toán đám mây, các doanh nghiệp có thể chuyển đổi cách thức hoạt động và mở ra những cơ hội mới trong kỷ nguyên số.

1.2. Luồng dữ liệu (Data Flow)

Định nghĩa: Luồng dữ liệu (Data Flow) là quá trình di chuyển dữ liệu từ một nguồn đến một đích trong một hệ thống. Khái niệm này mô tả cách dữ liệu được truyền tải, biến đổi và xử lý giữa các thành phần trong hệ thống. Luồng dữ liệu không chỉ là một phần quan trọng trong thiết kế hệ thống thông tin, hệ thống phần mềm mà còn là yếu tố quyết định trong việc phân tích nghiệp vụ, giúp hiểu rõ cách thức mà dữ liệu di chuyển và được xử lý trong toàn bộ quy trình.

- Các Thành Phần Chính của Luồng Dữ Liệu:
  - Nguồn Dữ Liệu (Data Source):
    Đây là nơi dữ liệu được tạo ra hoặc thu thập. Nguồn dữ liệu có thể là thiết bị IoT, cơ sở dữ liệu, hệ thống quản lý, hoặc thậm chí là người dùng cuối nhập dữ liệu.

  - Điểm Đích (Data Destination):
    Điểm đích là nơi dữ liệu được gửi đến sau khi đã được xử lý. Nó có thể là một cơ sở dữ liệu, hệ thống phân tích, hoặc một giao diện người dùng.

  - Biến Đổi Dữ Liệu (Data Transformation):
    Quá trình biến đổi dữ liệu liên quan đến việc chuyển đổi hoặc định dạng dữ liệu để nó có thể được sử dụng trong các ứng dụng khác nhau. Điều này có thể bao gồm các bước như làm sạch dữ liệu, chuẩn hóa, hoặc tổng hợp thông tin.

  - Quá Trình Xử Lý Dữ Liệu (Data Processing):
    Đây là các hoạt động mà dữ liệu trải qua trong suốt quá trình di chuyển từ nguồn đến đích. Điều này có thể bao gồm các phép

toán, phân tích, hoặc các thuật toán học máy nhằm rút ra thông tin hữu ích từ dữ liệu.

- Luồng Dữ Liệu (Data Flow):
  Luồng dữ liệu mô tả hướng di chuyển của dữ liệu trong hệ thống. Nó thể hiện cách mà dữ liệu chảy từ nguồn đến đích qua các quá trình xử lý và biến đổi.

- Mô Hình Luồng Dữ Liệu:
  Mô hình luồng dữ liệu có thể được biểu diễn bằng sơ đồ luồng dữ liệu (Data Flow Diagram - DFD), trong đó các thành phần được thể hiện bằng các hình khối, với các mũi tên chỉ hướng di chuyển của dữ liệu. Sơ đồ này giúp trực quan hóa các bước trong quy trình xử lý dữ liệu, làm rõ cách mà các thành phần tương tác với nhau.

- Lợi Ích của Luồng Dữ Liệu:
  - Tối ưu hóa quy trình:
    Hiểu rõ luồng dữ liệu giúp phát hiện và loại bỏ các điểm nghẽn trong quy trình, từ đó tối ưu hóa hiệu suất hoạt động của hệ thống.

  - Cải thiện khả năng bảo trì:
    Khi luồng dữ liệu được thiết kế rõ ràng, việc bảo trì và nâng cấp hệ thống sẽ trở nên dễ dàng hơn, vì các mối liên kết giữa các thành phần đã được xác định.

  - Hỗ trợ phân tích dữ liệu:
    Việc nắm bắt rõ luồng dữ liệu giúp các nhà phân tích dễ dàng t heo dõi, đánh giá và đưa ra quyết định dựa trên dữ liệu.

  - Tăng cường tính linh hoạt:

    Hệ thống có thể dễ dàng điều chỉnh khi có sự thay đổi trong yêu cầu của người dùng hoặc môi trường kinh doanh, nhờ vào sự hiểu biết sâu sắc về luồng dữ liệu.

Kết Luận:
Luồng dữ liệu là một khái niệm cơ bản nhưng cực kỳ quan trọng trong thiết kế và phát triển hệ thống thông tin hiện đại. Bằng cách phân tích

và tối ưu hóa luồng dữ liệu, các tổ chức có thể nâng cao hiệu quả hoạt động, giảm thiểu chi phí và cải thiện trải nghiệm người dùng. Do đó, việc xây dựng một mô hình luồng dữ liệu rõ ràng và chi tiết không chỉ giúp cải thiện khả năng quản lý dữ liệu mà còn tạo điều kiện thuận lợi cho sự phát triển bền vững của tổ chức trong thời đại số.

3. *Cân bằng tải (Load Balancing)*

Định nghĩa: Cân bằng tải (Load Balancing) là một kỹ thuật quan trọng trong quản lý lưu lượng truy cập và khối lượng công việc của hệ thống, nhằm phân phối đều đặn tải giữa nhiều máy chủ hoặc tài nguyên khác nhau. Kỹ thuật này giúp đảm bảo hiệu suất tối ưu, tính sẵn sàng cao và khả năng mở rộng linh hoạt cho hệ thống. Mục tiêu chính của cân bằng tải là tối ưu hóa việc sử dụng tài nguyên, ngăn chặn tình trạng quá tải ở các máy chủ, đồng thời bảo đảm rằng mọi máy chủ trong hệ thống đều hoạt động hiệu quả.

- Vai Trò của Cân Bằng Tải:
    - Giảm Tải cho Các Máy Chủ Đơn Lẻ:
        - Cân bằng tải giúp phân phối lưu lượng truy cập đều giữa các máy chủ, tránh việc một máy chủ nào đó bị quá tải trong khi các máy chủ khác vẫn còn khả năng xử lý. Điều này không chỉ giúp cải thiện hiệu suất tổng thể mà còn kéo dài tuổi thọ của phần cứng máy chủ.
    - Tăng Cường Khả Năng Chịu Lỗi (Fault Tolerance) và Dự Phòng (Redundancy):
        - Khi một máy chủ gặp sự cố hoặc không khả dụng, cân bằng tải có thể tự động chuyển hướng lưu lượng đến các máy chủ khác đang hoạt động. Điều này giúp hệ thống duy trì hoạt động liên tục, nâng cao độ tin cậy và khả năng phục hồi sau sự cố.
    - Tối Ưu Hóa Thời Gian Phản Hồi (Response Time):
        - Bằng cách phân phối công việc đồng đều, cân bằng tải giúp giảm thời gian phản hồi của hệ thống, cải thiện trải nghiệm người dùng. Người dùng sẽ nhận được dịch vụ nhanh chóng và ổn định hơn.

- Hỗ Trợ Khả Năng Mở Rộng (Scalability):
  - Cân bằng tải cho phép hệ thống dễ dàng thêm hoặc bớt tài nguyên mà không gây ra gián đoạn trong dịch vụ. Khi lưu lượng tăng lên, có thể thêm máy chủ mới vào cụm máy chủ mà không làm ảnh hưởng đến hiệu suất hoạt động.
- Các Phương Pháp Cân Bằng Tải:
  - Cân Bằng Tải Dựa Trên Phần Cứng (Hardware Load Balancing):

    Sử dụng thiết bị phần cứng chuyên dụng để quản lý lưu lượng truy cập. Các thiết bị này thường rất mạnh mẽ và có khả năng xử lý một lượng lớn yêu cầu một cách nhanh chóng.
  - Cân Bằng Tải Dựa Trên Phần Mềm (Software Load Balancing):

    Sử dụng phần mềm để phân phối lưu lượng truy cập giữa các máy chủ. Phương pháp này thường linh hoạt hơn và dễ dàng cấu hình, với chi phí thấp hơn so với giải pháp phần cứng.
  - Cân Bằng Tải Dựa Trên DNS (DNS Load Balancing):

    Sử dụng máy chủ DNS để phân phối lưu lượng đến nhiều máy chủ khác nhau. Mặc dù phương pháp này có thể không chính xác như các phương pháp khác, nhưng nó là một giải pháp đơn giản và hiệu quả cho các ứng dụng không yêu cầu quá nhiều tài nguyên.
  - Cân Bằng Tải Cấp 7 (Application Layer Load Balancing):

    Thực hiện cân bằng tải ở lớp ứng dụng, nơi có thể phân phối lưu lượng dựa trên thông tin ứng dụng như URL, cookie hoặc các tiêu chí khác.
- Lợi Ích Của Cân Bằng Tải:
  - Tăng Cường Hiệu Suất: Bằng cách tối ưu hóa việc sử dụng tài nguyên, cân bằng tải giúp nâng cao hiệu suất của hệ thống.
  - Cải Thiện Độ Tin Cậy: Hệ thống có thể tiếp tục hoạt động ngay cả khi một hoặc nhiều máy chủ gặp sự cố.

- Tăng Cường Trải Nghiệm Người Dùng: Thời gian phản hồi nhanh và khả năng tiếp cận liên tục giúp nâng cao sự hài lòng của khách hàng.
- Tiết Kiệm Chi Phí: Việc sử dụng tài nguyên hiệu quả giúp giảm chi phí vận hành tổng thể.

Kết Luận:

Cân bằng tải là một kỹ thuật thiết yếu trong việc xây dựng các hệ thống thông tin hiện đại, giúp đảm bảo rằng các ứng dụng và dịch vụ luôn sẵn sàng và hoạt động hiệu quả. Bằng cách tối ưu hóa lưu lượng truy cập và phân phối khối lượng công việc, cân bằng tải không chỉ cải thiện hiệu suất mà còn tạo ra một hệ sinh thái hoạt động ổn định và đáng tin cậy cho người dùng cuối.

1.1. Môi trường co giãn (Elastic Environment)

Định nghĩa: Môi trường co giãn (Elastic Environment) là một hệ thống máy tính hoặc môi trường điện toán có khả năng tự động mở rộng hoặc thu hẹp tài nguyên dựa trên nhu cầu thực tế của người dùng hoặc ứng dụng. Khả năng này chủ yếu được áp dụng trong các dịch vụ điện toán đám mây, giúp tối ưu hóa hiệu suất sử dụng tài nguyên, đảm bảo dịch vụ liên tục và tiết kiệm chi phí cho các doanh nghiệp và tổ chức.

- Đặc Điểm của Môi Trường Co Giãn:
  - Tự Động Điều Chỉnh:

Một trong những đặc điểm nổi bật của môi trường co giãn là khả năng tự động điều chỉnh tài nguyên. Hệ thống có thể thêm hoặc gỡ bỏ tài nguyên máy tính mà không cần sự can thiệp thủ công của quản trị viên, giúp tiết kiệm thời gian và nguồn lực cho đội ngũ IT.

  - Mở Rộng Theo Chiều Ngang (Horizontal Scaling):

Mở rộng theo chiều ngang liên quan đến việc thêm các máy chủ hoặc node vào hệ thống khi nhu cầu tăng lên. Ví dụ, nếu một ứng dụng nhận được nhiều lưu lượng truy cập hơn, hệ thống có thể tự động thêm các máy chủ mới để xử lý lưu lượng này. Khi nhu cầu giảm, số lượng máy chủ cũng sẽ được giảm bớt, từ đó tối ưu hóa chi phí.

- Mở Rộng Theo Chiều Dọc (Vertical Scaling):

  Mở rộng theo chiều dọc đề cập đến việc tăng hoặc giảm sức mạnh của một máy chủ cụ thể bằng cách tăng cường các tài nguyên như CPU và RAM cho máy đó. Phương pháp này có thể hữu ích trong những tình huống mà việc tăng cường hiệu suất của một máy chủ duy nhất là cần thiết.
- Khả Năng Đáp Ứng Theo Nhu Cầu:

  Hệ thống co giãn có khả năng giám sát và phân tích khối lượng công việc hiện tại để quyết định điều chỉnh tài nguyên sao cho phù hợp. Việc này thường được thực hiện thông qua các thuật toán và công cụ giám sát thông minh, đảm bảo rằng tài nguyên luôn được phân bổ một cách hợp lý.

- Lợi Ích của Môi Trường Co Giãn:
  - Tối Ưu Hóa Hiệu Suất:

    Khả năng tự động điều chỉnh tài nguyên giúp đảm bảo rằng ứng dụng luôn hoạt động ở hiệu suất tối ưu, bất kể khối lượng công việc có thay đổi ra sao.

  - Tiết Kiệm Chi Phí:

    Môi trường co giãn giúp doanh nghiệp chỉ phải trả tiền cho những tài nguyên mà họ thực sự sử dụng, giảm thiểu chi phí cho tài nguyên không cần thiết.
  - Đảm Bảo Tính Liên Tục:

    Môi trường co giãn giúp duy trì tính liên tục của dịch vụ, ngay cả trong các tình huống lưu lượng đột biến, bảo đảm trải nghiệm người dùng không bị gián đoạn.
  - Khả Năng Linh Hoạt:

    Doanh nghiệp có thể dễ dàng thay đổi quy mô tài nguyên theo nhu cầu kinh doanh mà không cần phải thực hiện các

thủ tục phức tạp, giúp họ linh hoạt hơn trong việc đáp ứng các thay đổi trong môi trường kinh doanh.

Kết Luận:

Môi trường co giãn là một yếu tố thiết yếu trong việc tối ưu hóa hiệu suất và tiết kiệm chi phí trong các hệ thống điện toán đám mây hiện đại. Bằng cách cho phép các doanh nghiệp tự động điều chỉnh tài nguyên dựa trên nhu cầu thực tế, môi trường co giãn giúp đảm bảo rằng các ứng dụng và dịch vụ luôn sẵn sàng và hoạt động hiệu quả. Điều này không chỉ cải thiện hiệu suất tổng thể mà còn tạo ra một trải nghiệm người dùng tốt hơn và tăng cường khả năng cạnh tranh trên thị trường.

2. Tổng quan tình hình nghiên cứu

❖ *Điện toán đám mây và xử lý luồng dữ liệu lớn*

Điện toán đám mây đã trở thành một giải pháp quan trọng trong việc lưu trữ và xử lý dữ liệu lớn, nhờ khả năng cung cấp tài nguyên linh hoạt, mở rộng dễ dàng và hiệu suất cao. Các nhà cung cấp dịch vụ đám mây như Amazon Web Services (AWS), Google Cloud, và Microsoft Azure đã cung cấp nhiều giải pháp đa dạng để xử lý dữ liệu theo thời gian thực, chẳng hạn như Apache Kafka, Apache Flink, và AWS Lambda. Việc xử lý luồng dữ liệu lớn (big data stream processing) là một thách thức quan trọng trong nhiều lĩnh vực như tài chính, y tế, viễn thông, và thương mại điện tử.

Tình hình nghiên cứu hiện tại

Nghiên cứu trong lĩnh vực này đã tập trung vào việc tối ưu hóa kiến trúc và phương pháp quản lý tài nguyên để đạt hiệu suất tối đa. Một số hướng nghiên cứu chính bao gồm:

- **Tối ưu hóa hiệu suất tính toán**: Nghiên cứu các mô hình tính toán phân tán và song song để tối ưu hóa quá trình xử lý dữ liệu lớn trong thời gian thực, như Apache Storm, Apache Spark Streaming, và Flink.

- **Quản lý tài nguyên co giãn**: Xây dựng các thuật toán quản lý tài nguyên tự động (Auto Scaling) để điều chỉnh tài nguyên máy chủ và lưu trữ theo yêu cầu của khối lượng dữ liệu và ứng dụng.
- **Tối ưu hóa truyền tải dữ liệu**: Áp dụng các kỹ thuật nén và mã hóa để giảm băng thông sử dụng khi truyền dữ liệu qua mạng.

❖ ***Cơ chế co giãn trong môi trường đám mây***

Môi trường co giãn là một trong những yếu tố quan trọng của điện toán đám mây. Nó cho phép các hệ thống tự động điều chỉnh tài nguyên dựa trên khối lượng công việc thay đổi. Điều này đảm bảo rằng các ứng dụng chỉ sử dụng đúng số lượng tài nguyên cần thiết, giúp tối ưu hóa chi phí và hiệu suất.

Tình hình nghiên cứu hiện tại

Nhiều nghiên cứu tập trung vào việc phát triển các thuật toán điều khiển co giãn tự động nhằm tăng cường hiệu suất và giảm thiểu chi phí:

- **Co giãn dọc (Vertical Scaling)**: Tăng hoặc giảm tài nguyên trên một máy chủ (CPU, RAM) khi khối lượng công việc thay đổi.
- **Co giãn ngang (Horizontal Scaling)**: Thêm hoặc xóa các máy chủ ảo (instances) để đáp ứng nhu cầu tài nguyên.
- **Cân bằng tải (Load Balancing)**: Các hệ thống cân bằng tải giúp phân phối công việc đồng đều trên các tài nguyên có sẵn, giảm thiểu nguy cơ quá tải và nâng cao độ tin cậy của hệ thống.

❖ ***Nén và lưu trữ dữ liệu trên đám mây***

Việc nén dữ liệu là một khía cạnh quan trọng trong việc xử lý luồng dữ liệu lớn, giúp giảm kích thước dữ liệu khi truyền tải qua mạng và tiết kiệm không gian lưu trữ. Các thuật toán nén dữ liệu như **Deflate**, **LZMA**, **LZ4**, và **Bzip2** đã được áp dụng trong nhiều trường hợp để tối ưu hóa luồng dữ liệu và nâng cao hiệu suất.

Tình hình nghiên cứu hiện tại

- **Thuật toán nén dữ liệu**: Các thuật toán nén khác nhau đã được nghiên cứu và tối ưu hóa cho các mục đích cụ thể. Ví dụ, LZ4 được ưa chuộng

nhờ tốc độ nén nhanh, trong khi Bzip2 cung cấp mức nén cao hơn nhưng tốc độ chậm hơn.

- **Tối ưu hóa nén trong môi trường phân tán**: Nghiên cứu tập trung vào việc tối ưu hóa nén và giải nén dữ liệu trong các môi trường phân tán, nơi tài nguyên tính toán và băng thông mạng là yếu tố hạn chế.

3. **Đặt vấn đề**
   - Dựa trên các nghiên cứu, vấn đề đặt ra của nghiên cứu này là thiết kế hệ thống xử lý luồng dữ liệu dựa trên mô hình điện toán đám mây. Trong đó, hệ thống có các tính năng cụ thể như sau:
     - Hệ thống phải xử lý được dữ liệu từ nhiều nguồn dữ liệu khác nhau.
     - Hệ thống sử dụng tối ưu ba khía cạnh tài nguyên là hạ tầng mạng (network), hiệu năng tính toán (computing), và khả năng lưu trữ (storage).
4. **Cấu trúc bài nghiên cứu**

C. CÁC PHƯƠNG PHÁP NGHIÊN CỨU

Trên thế giới hiện nay, các phương pháp xử lý luồng dữ liệu phụ thuộc nhiều vào hạ tầng phần cứng, hệ quả là xu hướng sử dụng các giải pháp do các nhà cung cấp Cloud cung cấp như Amazon, Google, hay Microsoft. Nghiên cứu này thử nghiệm và khảo sát phương pháp do Amazon Kinestic và Google Dataflow. Từ đó, chương này sẽ đề xuất mô hình xử lý luồng dữ liệu cho các ứng dụng IoT dựa trên mô hình kiến trúc điện toán đám biên và điện toán đám mây.

4. Các phương pháp xử lý luồng dữ liệu

2.1.1 Amazon Kinesis

Amazon Kinesis Data Streams là một dịch vụ cung cấp bởi Amazon nhằm đơn giản hóa quá trình tiêu thụ, xử lý, và lưu trữ luồng dữ liệu cho bất kì ứng dụng nào với phạm vi khác nhau. Dịch vụ này cho phép người dùng xây dựng các ứng dụng có thể đáp ứng nhu cầu thời gian thực nhờ sử dụng các framework xử lý luồng và lưu trữ các luồng này vào các nguồn dữ liệu khác nhau (data stores). Các ứng dụng phổ biến khi sử dụng dịch vụ này bao gồm:

- Giám sát và báo cáo các dịch vụ yêu cầu thời gian thực.
- Phân tích dữ liệu thời gian thực.
- Các dịch vụ xử lý luồng dữ liệu phực tạp

Hình .. - Kiến trúc dịch vụ Amazon Kinesis

Hình cho thấy kiến trúc của các dự án sử dụng dịch vụ này. Điều đáng chú ý trong kiến trúc là thành phần dùng để xử lý luồng dữ liệu chính là Amazon Elastic

Cloud Computing (Amazon EC2). Đây là một dịch vụ khác của Amazon cung cấp các tài nguyên tính toán như CPU, RAM và bộ nhớ; tổng hợp các tài nguyên này gọi chung là instance, có thể hiểu là một loại phần cứng cụ thể để triển khai dịch vụ người dùng. Amazon EC2 sẽ triển khai qua việc khởi tạo và chạy các instance; để tăng tính ổn đinh (stability) và tính khả dụng (availability), các dự án trong triển khai thực tế sẽ được triển khai nhiều instance cho một dịch vụ và dùng LoadBalancing5 để phân phối yêu cầu người dùng đến các instance này. Điều này nhằm mục đích đáp ứng nhu

cầu thay đổi phía tải, đặc biệt là trong các ứng dụng xử lý luồng dữ liệu có nhu cầu thay đổi thất thường.

Để tự động hóa việc quản lý các instance của Amazon EC2, một dịch vụ khác của Amazon là AutoScaling sẽ tăng giảm các tài nguyên một cách tự động. Nguyên lý hoạt động dựa trên các cài đặt của người dùng như tăng giảm theo lịch trình (scheduled scaling), tăng giảm theo yêu cầu phía tải (dynamic scaing), và tăng giảm được dự báo trước (predictive scaling). Nhìn chung, cơ chế hoạt động dựa trên việc quan sát các thông số của instance thông qua CloudWatch, nếu vi phạm các cài đặt của người dùng thì các instance sẽ đuược triển khai thêm.

2.1.2 Google Dataflow

Google Dataflow là một dịch vụ của Google Cloud cung cấp xử lý đồng thời dữ liệu luồng (strem processing) và xử lý theo mẻ (batch processing). Nguyên lý hoạt
động của dịch vụ này dựa trên một pipeline bao gồm bốn bước:

- Planning: Dựa trên yêu cầu từ phía người dùng, các ảnh hưởng từ nguồn dữ liệu (data sources), bước này sẽ quyết định thay đổi pipeline theo hướng tăng hoặc giảm cả về một phạm vi và hiệu năng để đáp ứng yêu cầu đó. Nhờ việc theo dõi này, dịch vụ này đạt được tính availability cao, có thể nhanh chóng khắc phục lỗi, và đánh giá tác động của hạ tầng mạng nơi hệ thống được triển khai.
- Developing and testing: Bước này sẽ triển khai môi trường phát triển theo các phương pháp như (1) Sử dụng queues để tránh việc mất mát dữ liệu, (2) Sử dụng operators để giảm độ trễ mạng và kinh phí sử dụng, (3) Sử dụng xử lý theo mẻ để tránh việc quá tải hệ thống.
- Deploying: Bước này phục vụ ý tưởng triển khai CI/CD hiện nay cho các ứng dụng xử lý luồng và đồng thời tối ưu tài nguyên sử dụng.
- Monitoring: Giám sát hiệu năng hoạt động của pipeline để đảm bảo đáp ứng nhu cầu của người dùng.

Công nghệ mà dịch vụ này sử dụng chính là Kubernetes để hiện thực hóa pipeline xử lý luồng dữ liệu. Kubernetes là một dịch vụ quản lý triển khai container với container là các môi trường cần thiết để triển khai một tác vụ nào đó. Lý do sử dụng Kubernetes vì công nghệ này quản lý các container, mà các container có khả năng gia tăng và giảm đi nhanh chóng nhằm đáp ứng nhu cầu thất thường của các ứng dụng xử lý luồng dữ liệu.

5. *Phát triển mô hình*

1.1. Lý thuyết áp dụng

Trong quá trình phát triển ứng dụng xử lý lượng dữ liệu lớn, việc mô phỏng các luồng dữ liệu này là rất quan trọng để kiểm tra khả năng chịu tải và hiệu suất của hệ thống. **JMeter** là một công cụ phổ biến được sử dụng để mô phỏng lượng dữ liệu lớn. Nó cho phép người dùng tạo và thực hiện các bài kiểm tra tải, giúp xác định hiệu suất của ứng dụng dưới áp lực cao. JMeter có khả năng mô phỏng hàng nghìn yêu cầu từ nhiều client khác nhau, giúp các nhà phát

triển và quản trị viên đánh giá khả năng xử lý của hệ thống trong các điều kiện thực tế.

Khi đã mô phỏng lượng dữ liệu lớn và xác định được các điểm yếu của hệ thống, việc mở rộng (scale) hệ thống để đáp ứng nhu cầu ngày càng cao là bước tiếp theo quan trọng. **Docker** và **Kubernetes** là hai công cụ hỗ trợ hiệu quả cho việc này.

JMeter là một công cụ mã nguồn mở được phát triển bởi Apache, chủ yếu được sử dụng để kiểm thử hiệu suất và tải của các ứng dụng web và dịch vụ. Nó được thiết kế để mô phỏng nhiều người dùng truy cập vào một ứng dụng hoặc dịch vụ cùng một lúc, giúp đánh giá khả năng chịu tải và hiệu suất của hệ thống trong các điều kiện thực tế.

Docker là một nền tảng phần mềm cho việc phát triển, đóng gói và chạy ứng dụng trong các môi trường cô lập gọi là containers. Docker hỗ trợ cho các dịch vụ khác có thể phát triển và triển khai một cách nhanh chóng vì nó cho phép tách biệt giữa phần cơ sở hạ tầng cần triển khai lên và dịch vụ chạy trên cơ sở hạ tầng đó. Để sử dụng được Docker, có hai thứ người dùng cần biết là containers và images

Containers là môi trường chạy ứng dụng đóng gói. Chúng bao gồm mã ứng dụng, thư viện và các phụ thuộc cần thiết vì vậy nếu một dịch vụ sử dụng containers để chạy thì dịch vụ đó hoàn toàn không phải phụ thuộc vào nơi mà nó được chạy hoặc triển khai

Images là các gói đóng gói chứa tất cả các thành phần cần thiết để chạy một container. Một image bao gồm các tệp cấu hình, mã ứng dụng và tất cả các thư viện cần thiết, gọi chung là filesystem. Image được sử dụng để tạo ra các container chạy ứng dụng. Chúng có thể được chia sẻ và triển khai trên nhiều máy chủ và môi trường. Sử dụng công cụ Docker sẽ nhanh chóng mô phỏng được một client trong thực tế mà không cần quan tâm đến nơi triển khai client đó.

Kubernetes Đây là một dịch vụ được Google phát triển và tung ra thị trường dưới dạng mã nguồn mở vào năm 2014. Dịch vụ này dùng để quản lý các dự án, các dịch vụ được triển khai dưới dạng các containers. Cách triển khai dạng containers trở nên phổ biến hiện nay vì yêu cầu phát triển nhanh và nhanh chóng triển khai đến tay người dùng (CI/CD). Để hiểu rõ hơn điều này hãy cùng thử so sánh với các cách triển khai khác.

Triển khai truyền thống	Triển khai bằng máy ảo	Triển khai bằng container
Một dịch vụ trên cùng một máy	Nhiều dịch vụ trên cùng một máy vật lý	Nhiều dịch vụ trên cùng một máy vật lý
	FileSystem bao gồm cả OS	FileSystem không bao gồm OS
		Nhanh chóng triển khai và thu hồi dịch vụ

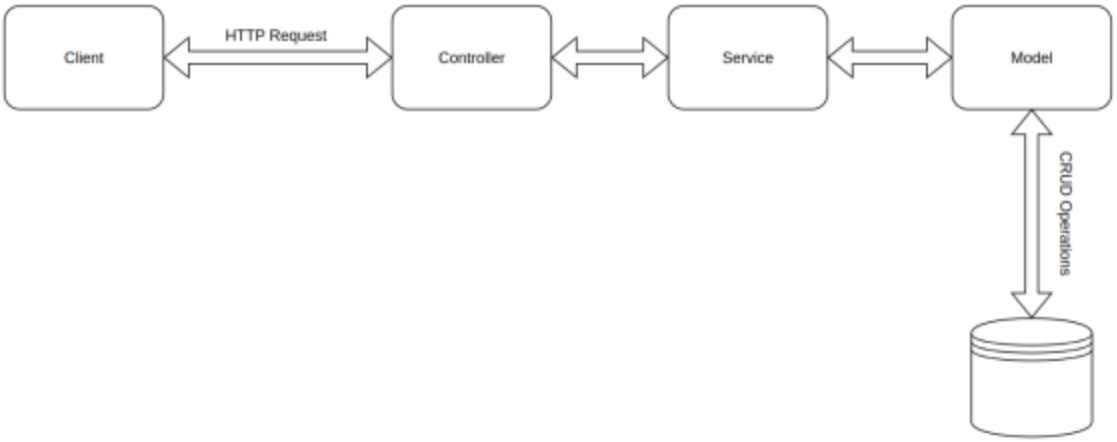
Khi sử dụng cách triển khai bằng các containers, kubernetes sẽ đảm bảo việc hệ thống triển khai sẽ không bao giờ ở trạng thái bảo trì vì bất cứ khi nào một container bị hỏng thì ngay lập tức kubernetes sẽ thực hiện thay thế bằng một container khác. Thêm vào đó, kubernetes giúp có thể mở rộng hệ thống theo chiều ngang để tận dụng tối đa phần cứng đang có và chức năng cân bằng tải khi có rất nhiều yêu cầu truy cập đến dịch vụ của bạn. Tuy vậy kubernetes có một điểm yếu là các dịch vụ chạy trên nó sẽ không theo dõi được log, ngoài ra cách kubernetes hoạt động cũng không theo một trình tự nhất định vì mục tiêu chính của kubernetes là làm sao đưa hệ thống đến một trạng thái mong muốn

Khi kết hợp JMeter để mô phỏng lượng dữ liệu lớn với Docker và Kubernetes để mở rộng hệ thống, ta có thể đạt được một quy trình phát triển và triển khai linh hoạt và mạnh mẽ. Bằng cách mô phỏng tải với JMeter, ta có thể xác định các yếu tố cần tối ưu trong hệ thống và sử dụng Docker/Kubernetes để triển khai các giải pháp mở rộng phù hợp, đảm bảo hệ thống có thể xử lý lượng dữ liệu lớn một cách hiệu quả và ổn định.

1.2. Xây dựng mô hình điện toán đám mây

2.2.1 Thiết kế phần mềm

Để phục vụ cho việc tiếp nhận dữ liệu từ phía Edge, Cloud sử dụng một service có tên là Data Receiver được xây dựng bằng Java Spring Boot cho phép xây dựng ra các cổng API có công việc chính là giao tiếp với cơ sở dữ liệu. Hình 2-15 là kiến trúc phần được xây dựng.



Hình .. - : Kiến trúc phần mềm trên đám mây

Thiết lập kết nối cơ sở dữ liệu Để có thể giao tiếp được với cơ sở dữ liệu, trước hết cần thiết lập một số biến môi trường liên quan đến cơ sở dữ liệu được sử dụng là Postgresql, thông qua file có tên application.yml

```yaml
spring:
  application:
    name: logger
  datasource:
    driver-class-name: org.postgresql.Driver
    url: jdbc:postgresql://db-thda-do-user-15365662-0.i.db.ondigitalocean.com:25060/data_logger
    username: doadmin
    password: AVNS_cYmLIioICOF3iou_v1P
  jpa:
    show-sql: true
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        dialect: org.hibernate.dialect.PostgreSQLDialect
server:
  port: 8080
```

Sau đó để nhận dữ liệu từ phía edge và có thể lưu được, cần một định dạng về thực thể có các thuộc tính được định nghĩa trước, thực thể sẽ được lưu trong cơ sở dữ liệu có tên là Data, gồm các thuộc tính như sau:

```java
package com.log.logger.logger.entity;

import jakarta.persistence.*;
import lombok.*;

import java.util.Date;

@AllArgsConstructor  12 usages
@NoArgsConstructor
@Getter
@Setter
@Builder
@Entity
@Table(name = "data")
public class DataEntity {
    @Id
    private String id;
    private String fileName;
    private String fileType;
    private int fileSize;
    private Date timeSent;
    private Date timeReceived;
    @Lob
    @Column(length = 500*1024)
    private byte[] content;
}
```

- Id: Một giá trị tự tăng xác định chỉ mục của dữ liệu được truyền vào
- fileName: tên tệp dữ liệu truyền đi
- fileType: loại tệp dữ liệu được truyền đi
- fileSize: Kích cỡ dữ liệu được truyền đi
- timeReceived: Thời gian nhận được dữ liệu trên
- timeSent: Thời gian được gửi từ phía Edge
- content: Nội dung của dữ liệu trên (tối đa 500MB)

Sau khi thiết lập xong kết nối với cơ sở dữ liệu, khi chạy service, thư viện Spring Boot Data JPA sẽ tự tạo thực thể Data trong cơ sở dữ liệu

Nhận dữ liệu được truyền từ phía Edge

Một đối tượng cần được định nghĩa phục vụ cho việc nhận dữ liệu từ phía Edge. Về cơ bản, các thuộc tính của DataFromClient và Data là giống nhau,

sau khi phía 41 cloud nhận được tín hiệu HTTP từ Edge, các thuộc tính của DataFromClient sẽ được ánh xạ với các thuộc tính trong thực thể Data.

```java
package com.log.logger.logger.request;

import ...

@AllArgsConstructor  5 usages
@NoArgsConstructor
@Getter
@Setter
public class CreateDataRequest {
    private String fileName;
    private String fileType;
    private int fileSize;
    private Date timeSent;
    private Date timeReceived;
    private byte[] content;
}
```

Thiết lập cổng API

Để truyền dữ liệu vào cơ sở dữ liệu, cần sử dụng đường dẫn http://[IP]/api/data với phương thức POST và dữ liệu là một định dạng JSON được định nghĩa trong Sơ đồ giao tiếp. Sau đó, Data Receiver sẽ gọi đến phương thức uploadData, phương thức này sẽ gọi đến đối tượng IDataService để thực hiện việc lưu dữ liệu

```java
1  package com.log.logger.logger.repository;
2
3  import com.log.logger.logger.entity.DataEntity;
4  import org.springframework.data.jpa.repository.JpaRepository;
5
6  public interface DataRepository extends JpaRepository<DataEntity, String> {  2 usages
7  }
8
```

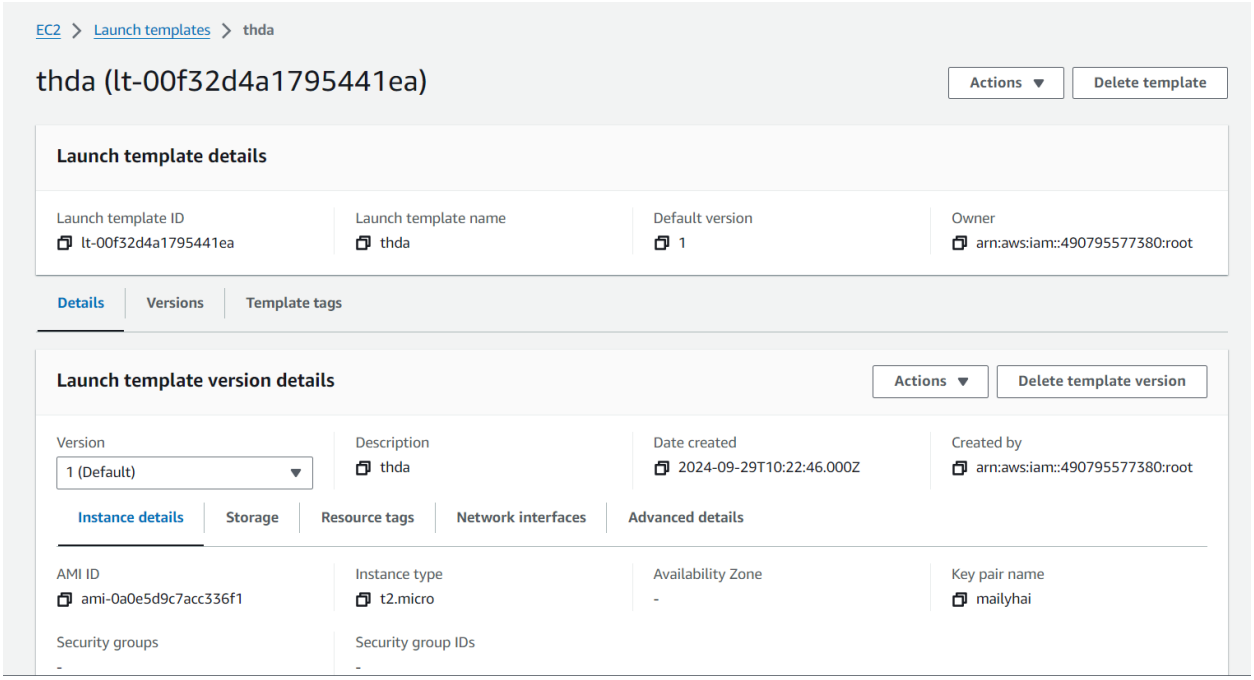
Tạo API để thao tác database từ client

```java
import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.*;

import java.io.IOException;
import java.util.List;
import java.util.concurrent.CompletableFuture;

@RequiredArgsConstructor
@RestController
@RequestMapping("/api/data")
public class DataController {

    private final DataService dataService;

    @PostMapping
    public CompletableFuture<String> create(@RequestBody CreateDataRequest request) {
        return CompletableFuture.supplyAsync(() -> {
            try {
                dataService.create(request);
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
            return "Data created successfully";
        });
    }

    @DeleteMapping("/{id}")
    public CompletableFuture<String> delete(@PathVariable String id) {
        return CompletableFuture.supplyAsync(() -> {
            dataService.delete(id);
            return "Data deleted successfully";
        });
    }

    @GetMapping("/{id}")
    public CompletableFuture<DataEntity> getById(@PathVariable String id) {
        return CompletableFuture.supplyAsync(() -> dataService.get(id));
    }

    @GetMapping
    public CompletableFuture<List<DataEntity>> getAll() { return CompletableFuture.supplyAsync(dataService::getAll); }
}
```

2.2.2 Thiết kế cơ chế co giãn

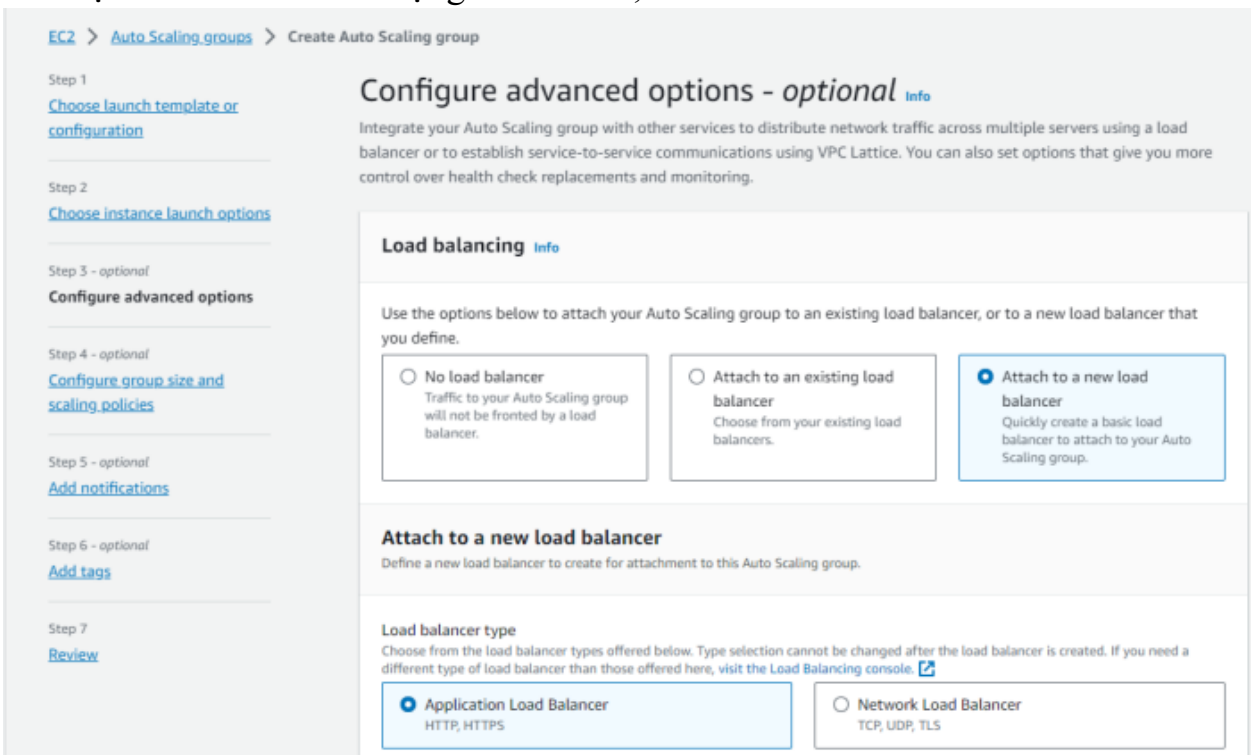
Khởi tạo Template cho Amazon EC2

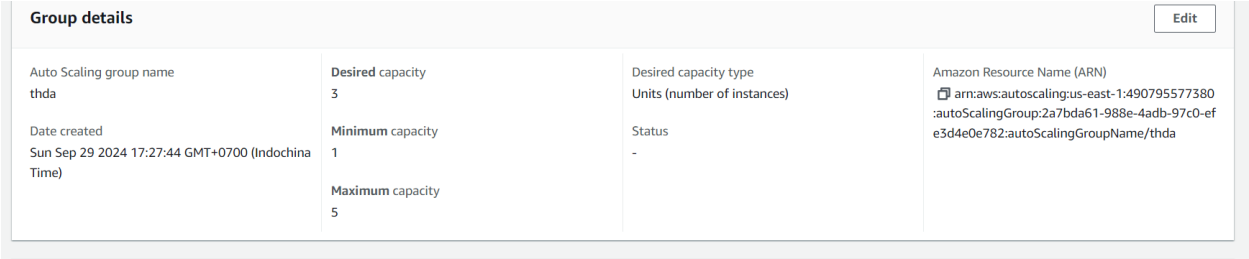
Các instance được khởi tạo sẽ có chung yêu cầu về CPU, RAM nên được gọi chung là template. Template được khởi tạo sử dụng OS là Ubuntu Server 22.04 LTS, 64-bit; loại instance là t2.micro (free tier eligible); có dung lượng bộ nhớ là 8GB.



Khởi tạo Auto Scaling groups

Lần lượt thực hiện theo hướng dẫn mà Amazon cung cấp. Điểm đáng chú ý là bước 3 và bước 4. Tại bước 3, LoadBalancer được khởi tạo để phân chia yêu cầu từ phía tải với các instance được khởi tạo. Số lượng instance được khởi tạo ở bước 4 với số lượng ít nhất là 1, nhiều nhất là 5.





Thiết lập Dynamic Scaling Policy

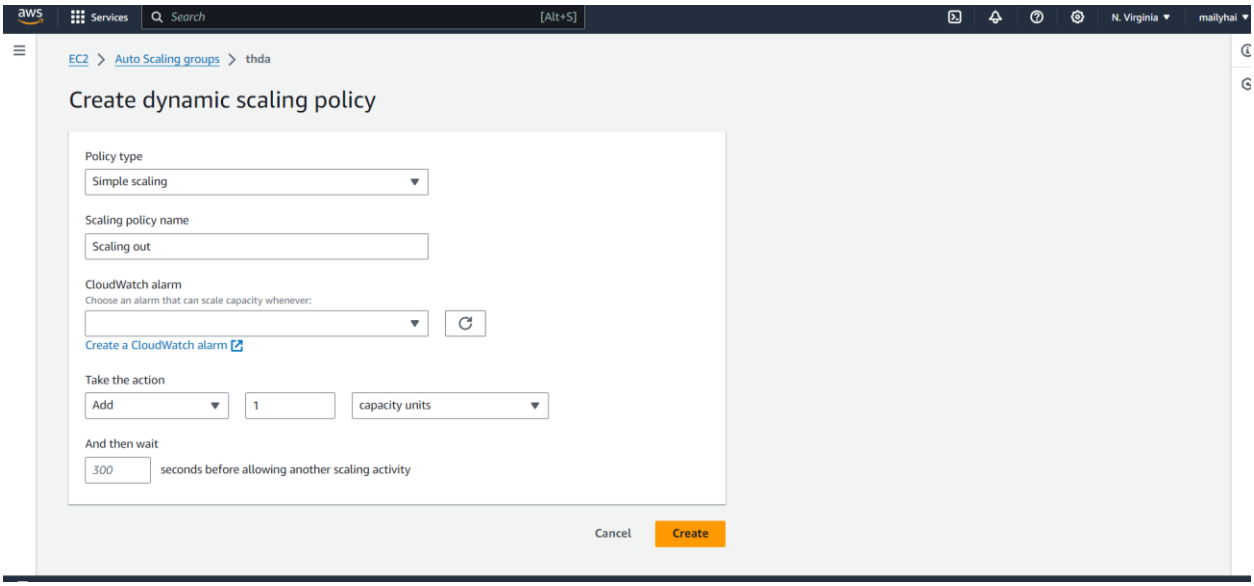
Yêu cầu phía tải được chia nhỏ thành ba cơ chế là

(1) Target tracking scaling theo dõi và đảm bảo Metrics bám theo một giá trị đặt có sẵn, ví dụ như tỷ lệ sử dụng CPU trung bình (Average CPU utilization) luôn đạt 30%

(2) Simple scaling là cơ chế tăng giảm instance dựa trên theo dõi Metrics vượt quá giá trị đặt (threshold value) và

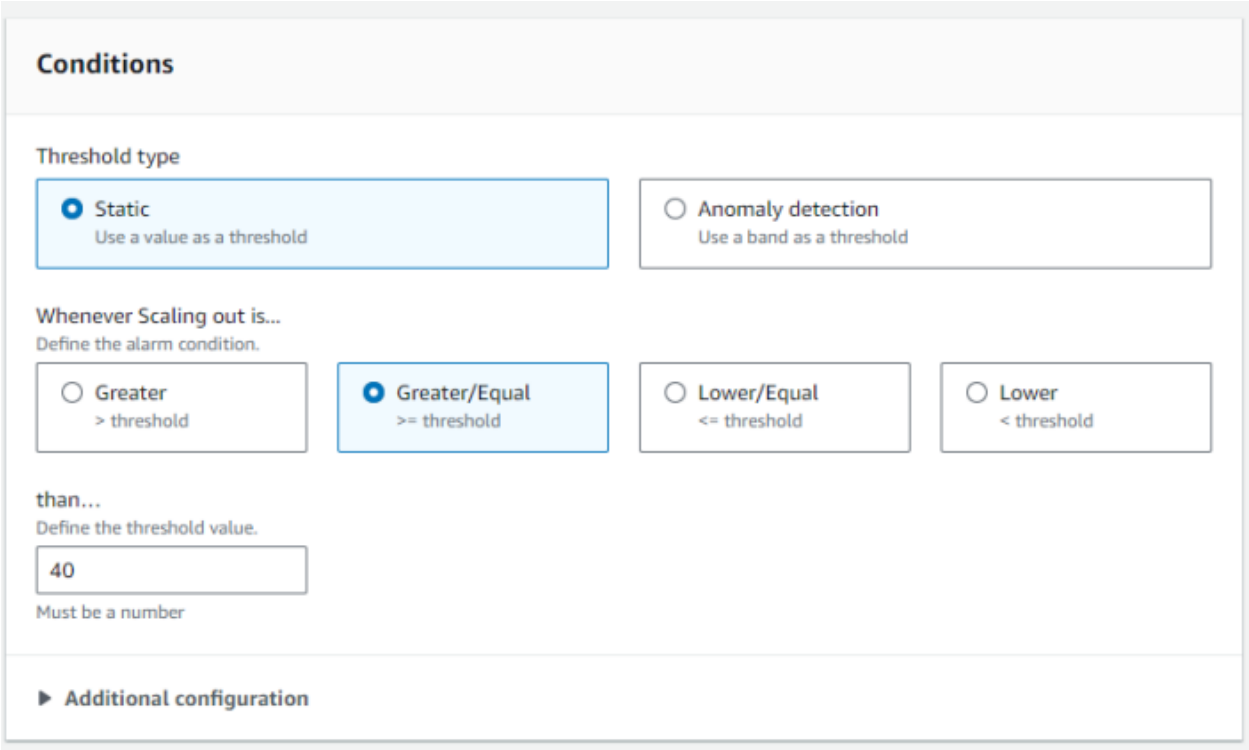
(3) Step scaling có thể hiểu là tổng hợp của nhiều simple scaling và các cơ chế co dãn được kích hoạt khi nhiều Metrics vượt quá giá trị đặt. Nghiên cứu này thử nghiên với cơ chế co dãn đơn giản nhất là Simple scaling và lựa chọn theo dõi phần trăm CPU sử dụng không quá 40%.

Trong bảng theo dõi các EC2, lựa chọn thda được khởi tạo ở trên và vào tab Automatic scaling. Trong tab này, có ba policy như đã trình bày bao gồm Dynamic scaling dựa trên vào yêu cầu phía tải, Predictive scaling dự báo trước yêu cầu và Scheduled tăng giảm theo lịch trình. Tại đây, policy dựa trên theo dõi CPU được khởi tạo bằng cách ấn nút Create dynamic scaling policy.

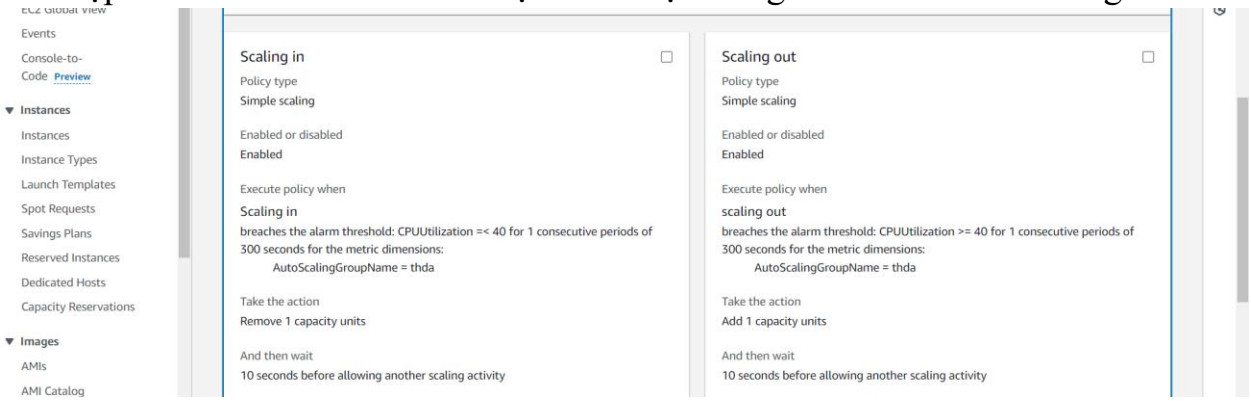


Hai cơ chế co dãn được thiết lập, cơ chế tăng thêm instance khi phần trăm CPU lớn hơn hoặc bằng 40% và cơ chế giảm instance khi phần trăm CPU

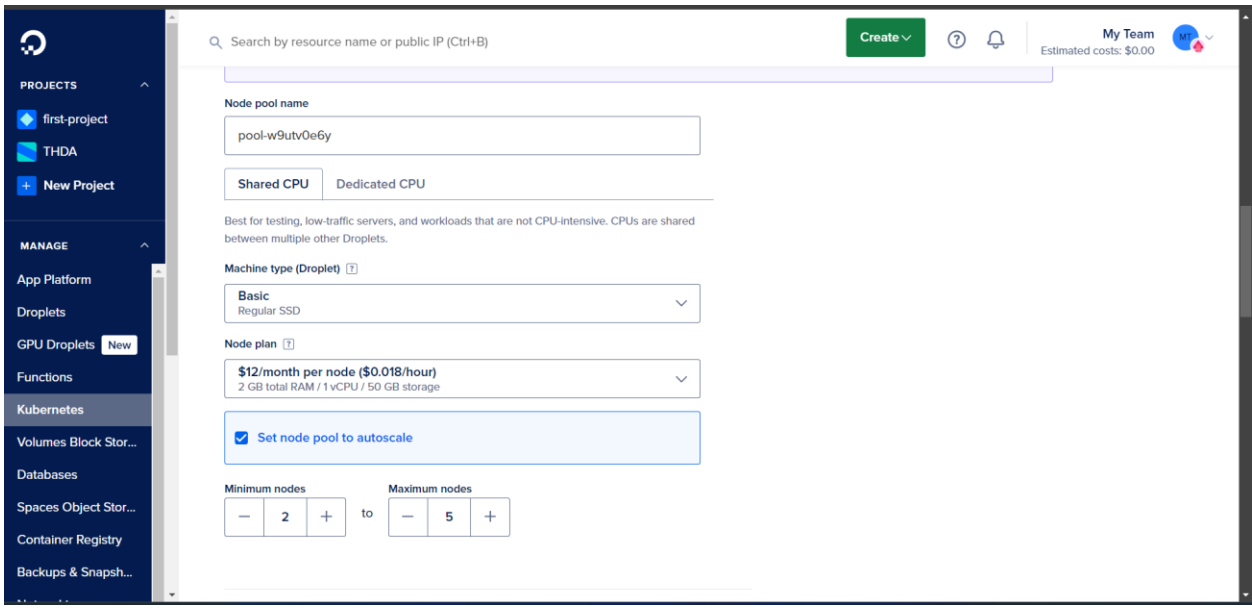
dưới 40%. Trước tiên là thiết lập cơ chế tăng instance dựa trên giám sát phần trăm CPU sử dụng trong CloudWatch. Trong khi khởi tạo Policy, lựa chọn Create a CloudWatch alarm và thiết lập thông số tương ứng.



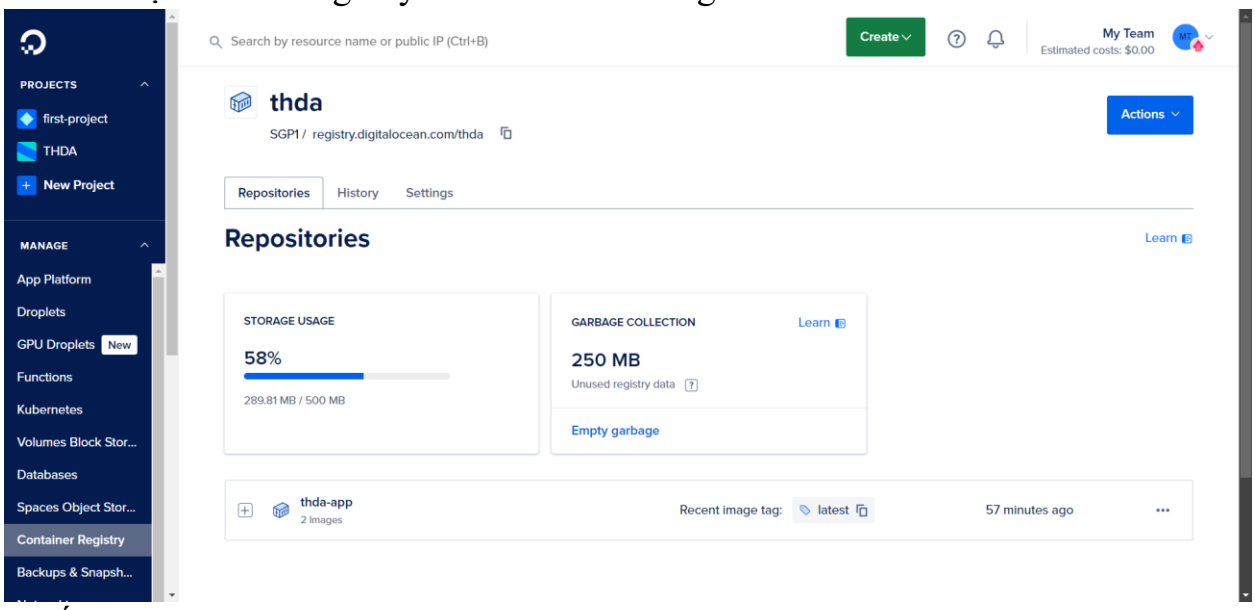
Thực hiện tương tự, cơ chế Scaling giảm instance khi CPU dưới 40% được thiết lập. Hai cơ chế sau khi khởi tạo xuất hiện trong tab Automatic scaling.



- Thiết kế cơ chế co giãn với k8s cluster DigitalOcean cloud
  - Khởi tạo cụm k8s



- Khởi tạo docker registry để lưu trữ các image docker



- Thiết lập kuberconfig

  doctl kubernetes cluster kubeconfig save 63ccde8a-b80c-40b5-aea7-ca718f11bb90

- Tạo Secret cho Docker Registry (DigitalOcean)

  kubectl create secret docker-registry regcred --docker-server=registry.digitalocean.com --docker-username=mailyhai --docker-password=dop_v1_c44b7f792639fde55d2586fff1dde5da929b593eff56a205ad188046e91d163a --docker-email=mailyhai814@gmail.com

- Cài đặt metric server để lấy thông tin tài nguyên phục vụ cho autoscale

kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml

- Tạo các file cấu hình yaml

deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: thda-app-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: thda-app
  template:
    metadata:
      labels:
        app: thda-app
    spec:
      containers:
        - name: thda-app
          image: registry.digitalocean.com/thda/thda-app:latest
          resources:
            requests:
              cpu: "1"
              memory: "256Mi"
            limits:
              cpu: "2"
              memory: "512Mi"
          ports:
            - containerPort: 80
          env:
            - name: SPRING_DATASOURCE_URL
              valueFrom:
                configMapKeyRef:
                  name: thda-app-configmap
                  key: SPRING_DATASOURCE_URL
            - name: SPRING_DATASOURCE_USERNAME
```

configmap.yaml

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: thda-app-configmap
data:
  SPRING_DATASOURCE_URL: jdbc:postgresql://db-thda-do-user-15365662-0.i.db.ondigitalocean.com:25060/data_logger
  SPRING_DATASOURCE_USERNAME: doadmin
  SPRING_DATASOURCE_PASSWORD: AVNS_cYmLIioICOF3iou_v1P

```

hpa.yaml

```yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: thda-app-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: thda-app-deployment # Đảm bảo tên này đúng
  minReplicas: 1
  maxReplicas: 10
  metrics:
    - type: Resource
      resource:
        name: cpu
        target:
          type: Utilization
          averageUtilization: 80

```

services.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: thda-app-service
spec:
  type: LoadBalancer
  selector:
    app: thda-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080

```

- Triển khai các file yaml lên k8s cluster
  kubectl apply -f configmap.yaml
  kubectl apply -f deployment.yaml
  kubectl apply -f services.yaml
  kubectl apply -f hpa.yaml
- Kiểm tra trạng thái
  kubectl get pods
  kubectl get services
  kubectl get hpa

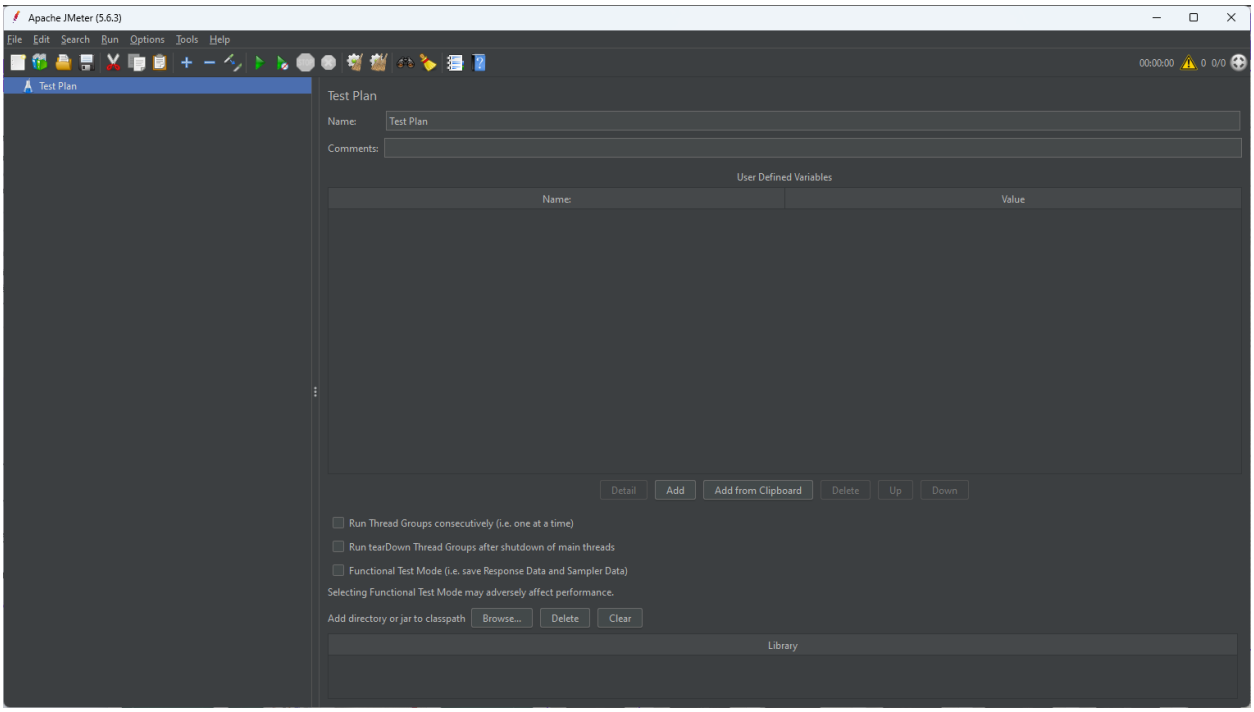
```
(base) PS D:\WorkSpace\Kubernetes\thda> kubectl get pods
>>    kubectl get services
>>    kubectl get hpa
NAME                                  READY   STATUS             RESTARTS       AGE
thda-app-deployment-7554bf6864-57zdb  0/1     Pending            0              18s
thda-app-deployment-7554bf6864-vxc7t  0/1     Pending            0              18s
thda-app-deployment-7554bf6864-w27kc  0/1     Pending            0              18s
thda-app-deployment-7554bf6864-xjgl2  0/1     Pending            0              18s
thda-app-deployment-7554bf6864-znflg  0/1     Pending            0              18s
thda-app-deployment-769cdd5856-6fgzn  0/1     Pending            0              26m
thda-app-deployment-769cdd5856-hhv7r  0/1     Pending            0              26m
thda-app-deployment-769cdd5856-pdt7q  0/1     Pending            0              26m
thda-app-deployment-769cdd5856-qhrmf  0/1     Pending            0              29m
thda-app-deployment-7c9cf4745-b6fwp   1/1     Running            0              26m
thda-app-deployment-7c9cf4745-qdhfv   1/1     Running            1 (27m ago)    28m
thda-app-deployment-7c9cf4745-qsl2v   0/1     CrashLoopBackOff   8 (17s ago)    26m
thda-app-deployment-7c9cf4745-w8nlb   0/1     CrashLoopBackOff   8 (18s ago)    26m
NAME               TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)        AGE
kubernetes         ClusterIP      10.245.0.1      <none>           443/TCP        14h
thda-app-service   LoadBalancer   10.245.194.18   144.126.243.188  80:32544/TCP   116m
NAME           REFERENCE                          TARGETS       MINPODS   MAXPODS   REPLICAS   AGE
thda-app-hpa   Deployment/thda-app-deployment     cpu: 3%/80%   1         10        10         53m
```

CÁC THỰC NGHIỆM VÀ KẾT QUẢ

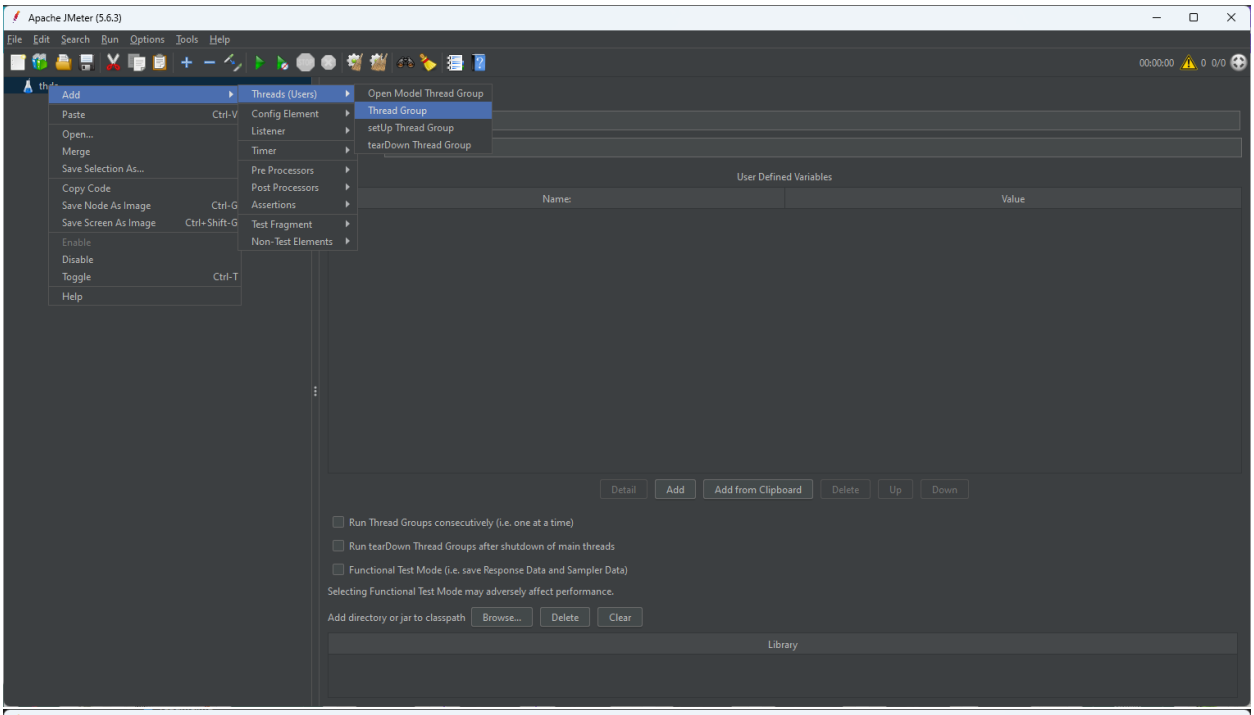
6. *Quá trình thực nghiệm*

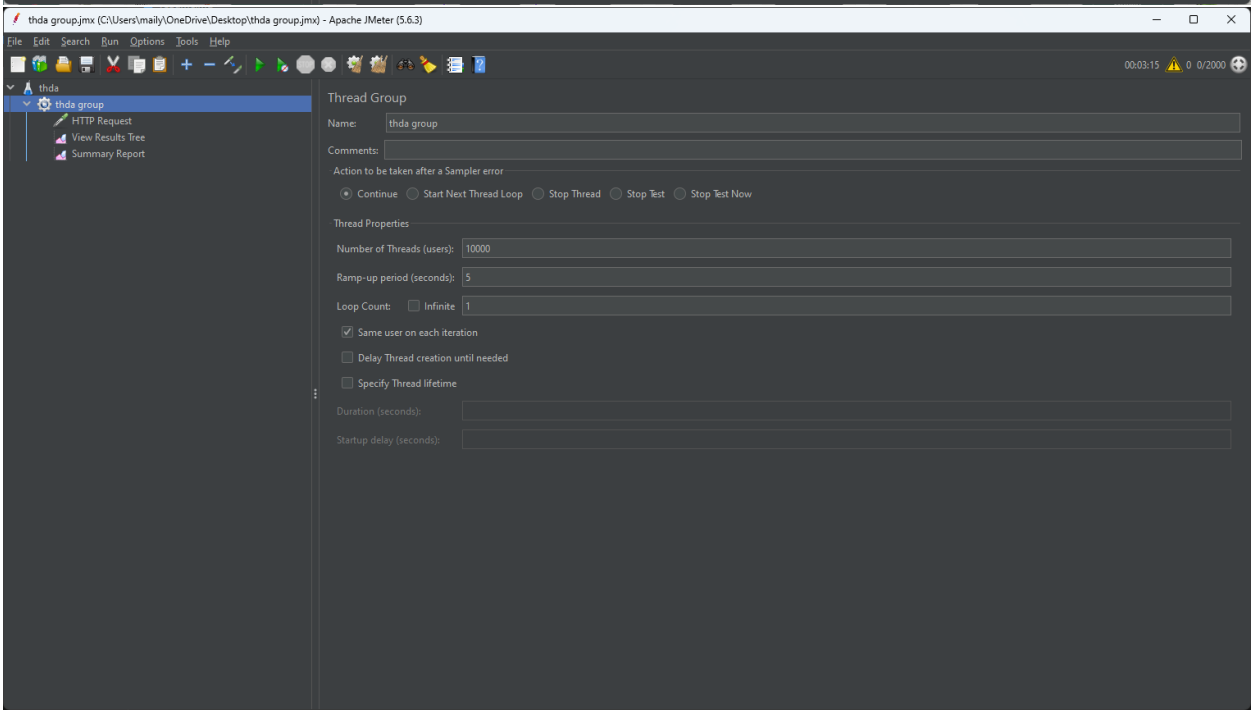
Thực hiện stress test với ApacheJmeter:

- Tạo test plan



- Thêm và cấu hình thread group

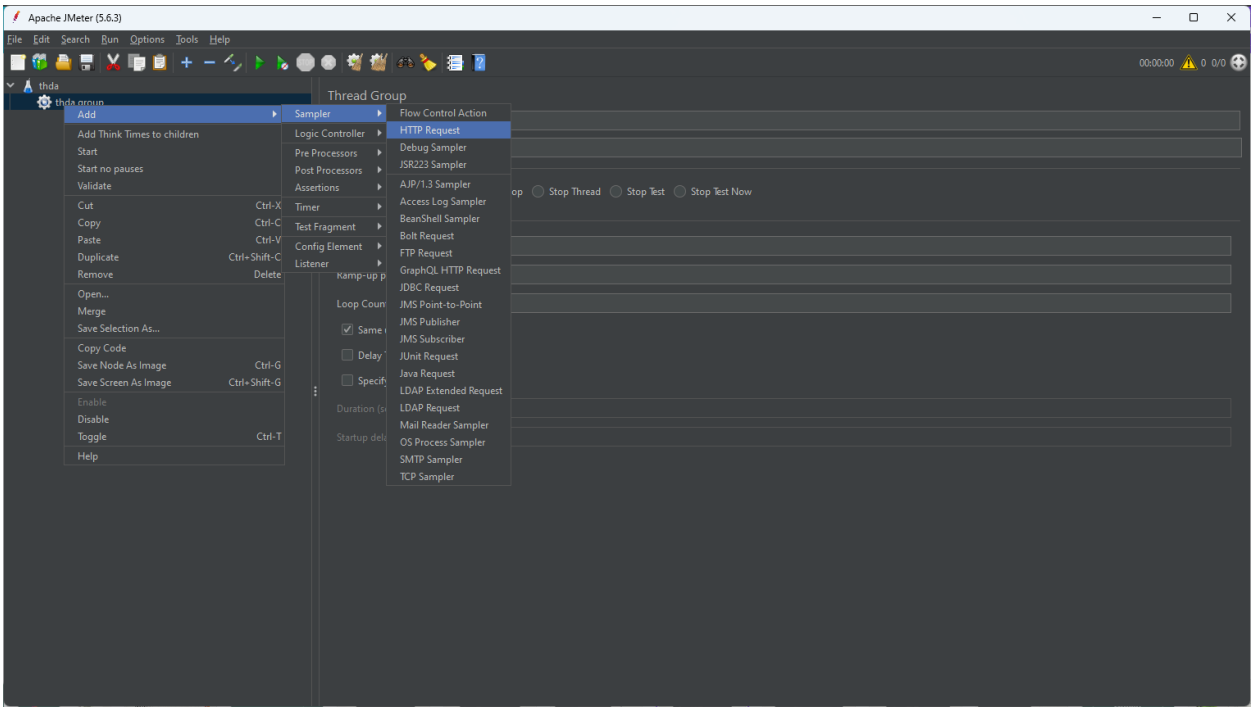




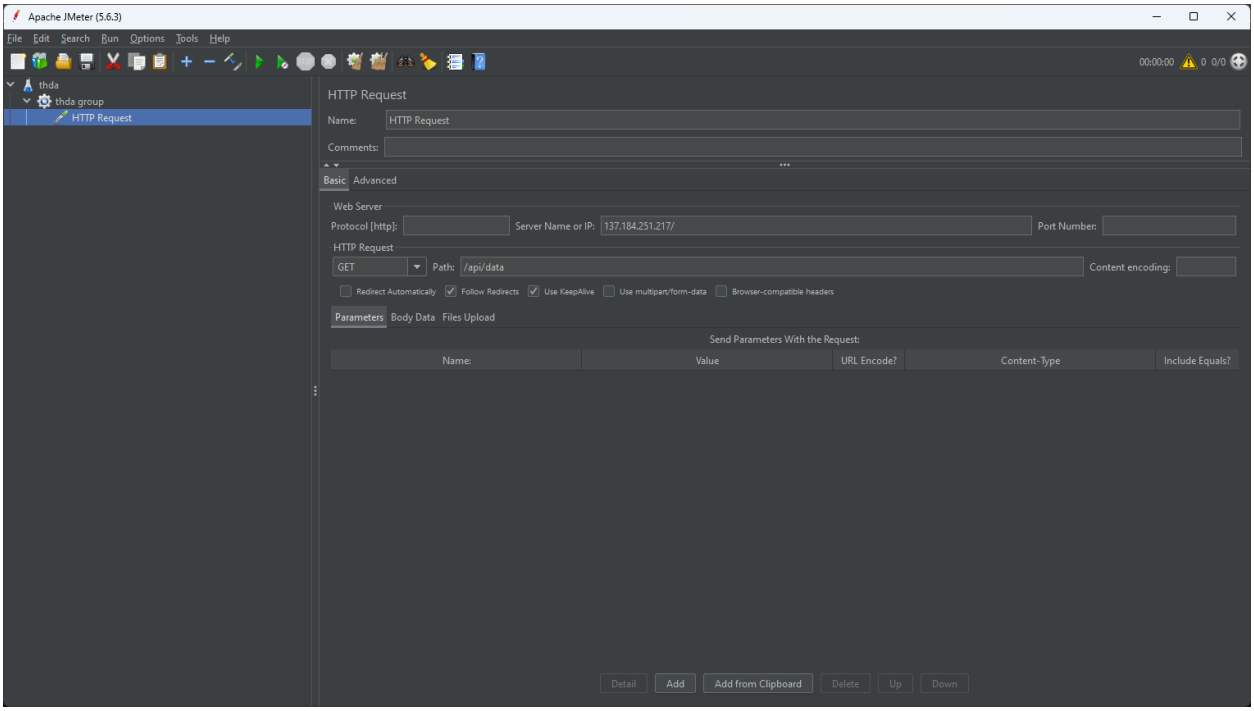
Trong đó number of thread là số lượng user đồng thời ta muốn mô phỏng, Ramp-up period là thời gian tăng từ 0 đến số lượng user ta muốn mô phỏng có nghĩa là thời gian tăng từ 0 user đến 10000 user sẽ mất 5 giây, Loop count là số lần lặp lại hành động của 1 user, ở đây mỗi user sẽ thực hiện 1 lần, nếu chọn infinite thì user sẽ thực hiện hành động đó liên tục cho đến khi nào ta dừng test.

- Thêm action

  Action là hành động mà mỗi user sẽ thực hiện trong khi test, có nhiều phương thức khác nhau, ta chọn phương thức phù hợp với server của mình



Ở đây chọn action là http request vì server đang được viết để nhận request dạng hypertext transfer protocol



Tiến hành điền các cấu hình cần thiết cho request:

Method: GET

Server name : domain của server nếu không có thì điền ip v4 public của server

Port: Cổng mà service đang chạy, mặc định sẽ là cổng 80 cho http và 443 cho https

Path: đường dẫn của router

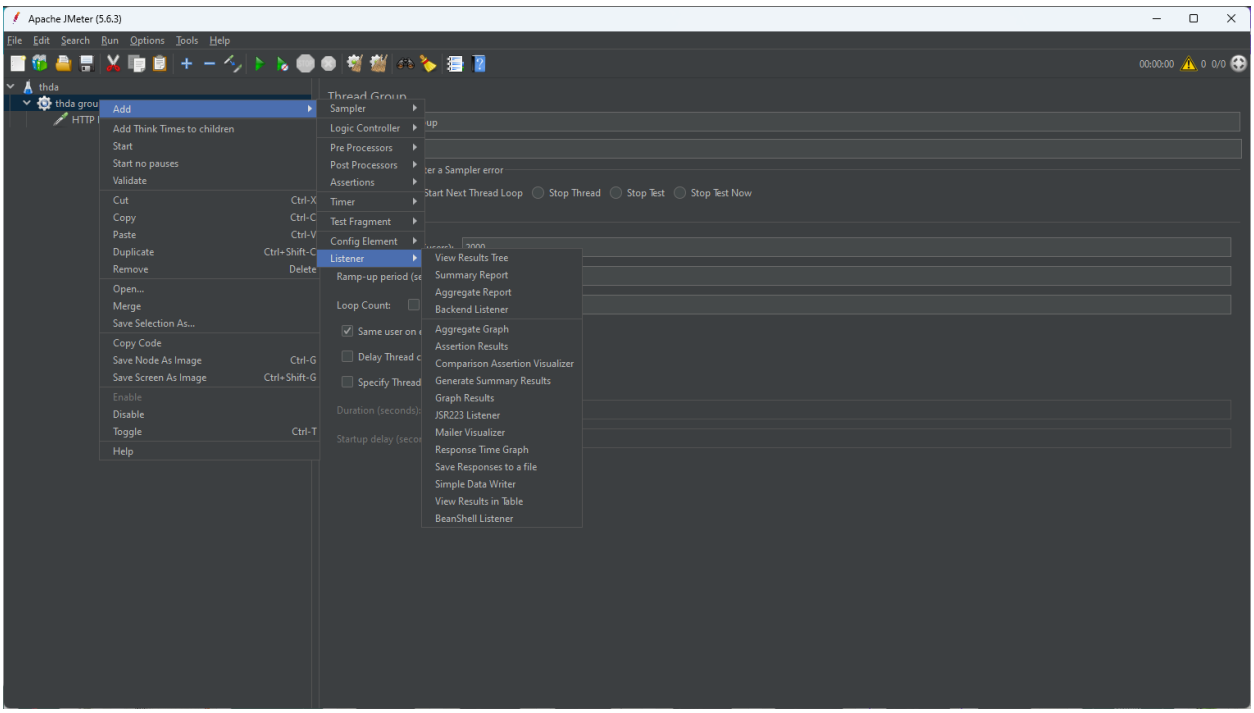
Parameter: Các biến được truyền vào trên thanh url

Body data: Các biến được truyền và trong body

Files upload: Dùng cho các trường hợp muốn uploadfile

- Thêm listener

Listener là các thành phần dùng để thu thập và xử lý kết quả của các test case. Dữ liệu thu thập được có thể làm dưới dạng bảng, biểu đồ hoặc file dùng để phân tích sau.



Có nhiều loại listener khác nhau, ở đây ta sẽ sử dụng 2 listener View result tree và Summary report

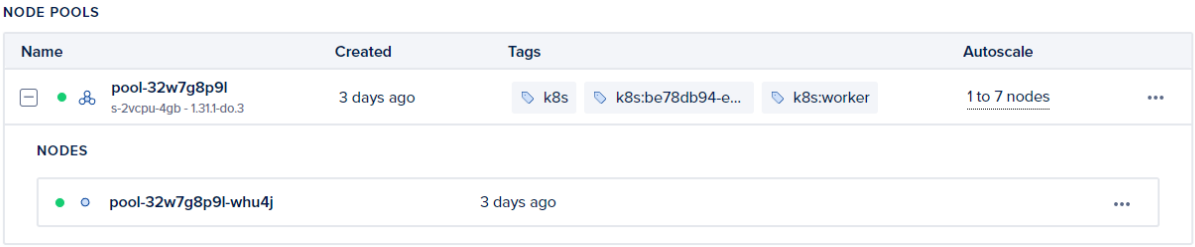
Sau khi đã thực hiện xong tất cả các bước trên, ta nhấn tổ hợp ctrl + R để bắt đầu quá trình test

7. *Kết quả thực nghiệm*

1.1. Hiệu quả tính năng co dãn

Khởi tạo ban đầu của dịch vụ trên Digital Ocean là 1 node và 2 pod, điều này được theo dõi qua control panel cung cấp. Hình sau cho thấy, các node này được phân bố cùng trong một node pools nhưng lại có địa chỉ Public IPv4 khác nhau. Điều này không ảnh hưởng tới truy cập do người dùng sẽ truy cập đến địa chỉ DNS cung cấp bởi LoadBalancer, và dịch vụ LoadBalancer sẽ tự động

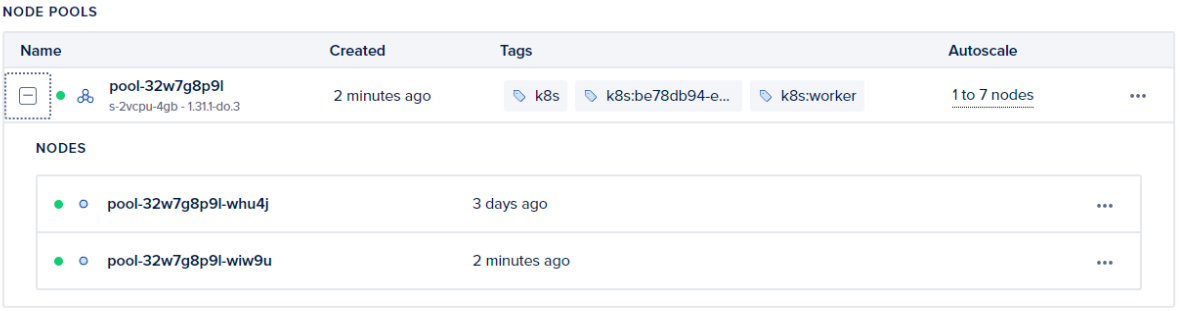
phân bố yêu cầu truy cập đó vào các node trong node pools này. Điều này đảm bảo tính ổn định và khả dụng khi có sự thay đổi từ phía yêu cầu người dùng



Tự động tăng số lượng pod và node

```
(base) PS D:\WorkSpace\kubernetes\thda> kubectl get hpa --watch
NAME          REFERENCE                                 TARGETS                          MINPODS   MAXPODS   REPLICAS   AGE
thda-app-hpa  Deployment/thda-app-deployment  cpu: 1%/50%, memory: 37%/80%   1         20        1          160m
thda-app-hpa  Deployment/thda-app-deployment  cpu: 10%/50%, memory: 38%/80%  1         20        1          160m
thda-app-hpa  Deployment/thda-app-deployment  cpu: 1%/50%, memory: 38%/80%   1         20        1          160m
thda-app-hpa  Deployment/thda-app-deployment  cpu: 1%/50%, memory: 38%/80%   1         20        1          161m
thda-app-hpa  Deployment/thda-app-deployment  cpu: 1%/50%, memory: 38%/80%   1         20        1          161m
thda-app-hpa  Deployment/thda-app-deployment  cpu: 1%/50%, memory: 38%/80%   1         20        1          162m
thda-app-hpa  Deployment/thda-app-deployment  cpu: 78%/50%, memory: 52%/80%  1         20        1          163m
thda-app-hpa  Deployment/thda-app-deployment  cpu: 200%/50%, memory: 114%/80%  1         20        2          163m
thda-app-hpa  Deployment/thda-app-deployment  cpu: 200%/50%, memory: 65%/80%  1         20        3          163m
thda-app-hpa  Deployment/thda-app-deployment  cpu: 200%/50%, memory: 50%/80%  1         20        4          163m
thda-app-hpa  Deployment/thda-app-deployment  cpu: 199%/50%, memory: 47%/80%  1         20        5          164m
thda-app-hpa  Deployment/thda-app-deployment  cpu: 174%/50%, memory: 52%/80%  1         20        6          164m
thda-app-hpa  Deployment/thda-app-deployment  cpu: 121%/50%, memory: 53%/80%  1         20        7          164m
thda-app-hpa  Deployment/thda-app-deployment  cpu: 75%/50%, memory: 55%/80%   1         20        8          164m
```

Số lượng pod đã tăng từ 1-8



Số lượng node đã tăng từ 1-2

Hiện tại cơ chế tự động tăng này dựa trên giám CPUUtilization Metrics, tức phần trăm CPU sử dụng và MemoryUtilization Metrics tức phần trăm ram sử dụng

Tự động giảm số lượng pod và node

```
thda-app-hpa  Deployment/thda-app-deployment  cpu: 50%/50%, memory: 34%/80%   1         20        8          167m
thda-app-hpa  Deployment/thda-app-deployment  cpu: 125%/50%, memory: 38%/80%  1         20        8          167m
thda-app-hpa  Deployment/thda-app-deployment  cpu: 125%/50%, memory: 40%/80%  1         20        9          167m
thda-app-hpa  Deployment/thda-app-deployment  cpu: 121%/50%, memory: 44%/80%  1         20        10         167m
thda-app-hpa  Deployment/thda-app-deployment  cpu: 39%/50%, memory: 44%/80%   1         20        11         168m
thda-app-hpa  Deployment/thda-app-deployment  cpu: 25%/50%, memory: 44%/80%   1         20        10         168m
thda-app-hpa  Deployment/thda-app-deployment  cpu: 25%/50%, memory: 44%/80%   1         20        9          168m
thda-app-hpa  Deployment/thda-app-deployment  cpu: 26%/50%, memory: 44%/80%   1         20        8          168m
thda-app-hpa  Deployment/thda-app-deployment  cpu: 29%/50%, memory: 46%/80%   1         20        7          169m
thda-app-hpa  Deployment/thda-app-deployment  cpu: 34%/50%, memory: 48%/80%   1         20        6          169m
thda-app-hpa  Deployment/thda-app-deployment  cpu: 40%/50%, memory: 51%/80%   1         20        5          169m
```

Số lượng pod bắt đầu giảm xuống khi phầm trăm sử dụng cpu dưới 50% và phần trăm sử dụng ram dưới 80%.

B. KẾT LUẬN

Trong bối cảnh phát triển nhanh chóng của công nghệ thông tin và sự gia tăng không ngừng của khối lượng dữ liệu, việc xây dựng một môi trường co giãn cho hệ thống có luồng dữ liệu lớn trên nền tảng điện toán đám mây trở nên vô cùng quan trọng. Đề tài nghiên cứu này đã chỉ ra rằng, để đáp ứng nhu cầu lưu trữ và xử lý dữ liệu một cách hiệu quả, việc thiết kế và triển khai một hệ thống có khả năng mở rộng tự động là rất cần thiết.

Thông qua việc áp dụng các phương pháp co giãn như **Auto Scaling**, **Load Balancing**, chúng ta có thể tối ưu hóa hiệu suất và tính sẵn sàng của hệ thống. Sự kết hợp giữa các công nghệ hiện đại như **Kubernetes** không chỉ giúp quản lý luồng dữ liệu lớn một cách hiệu quả mà còn nâng cao khả năng phục hồi của hệ thống trước các sự cố bất ngờ.

Cuối cùng, nghiên cứu này không chỉ mở ra hướng đi mới cho việc xây dựng môi trường co giãn cho các hệ thống có luồng dữ liệu lớn mà còn góp phần nâng cao nhận thức về tầm quan trọng của điện toán đám mây trong việc giải quyết các thách thức trong quản lý dữ liệu hiện nay. Hy vọng rằng những kết quả đạt được từ nghiên cứu này sẽ là cơ sở vững chắc cho các nghiên cứu và ứng dụng trong tương lai, giúp các tổ chức tối ưu hóa quy trình xử lý dữ liệu và đáp ứng nhanh chóng với các yêu cầu ngày càng cao của thị trường.

D. LỜI CẢM ƠN