HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Nguyễn Nhật Anh

NGHIÊN CỨU MÔ HÌNH MICROSERVICES VÀ ỨNG DỤNG VÀO HỆ THỐNG TÍNH CƯỚC TẠI VIETTEL

ĐỀ ÁN TỐT NGHIỆP THẠC SĨ

HÀ NỘI - NĂM 2024

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Nguyễn Nhật Anh

NGHIÊN CỨU MÔ HÌNH MICROSERVICES VÀ ỨNG DỤNG VÀO HỆ THỐNG TÍNH CƯỚC TẠI VIETTEL

Chuyên ngành: Kỹ thuật viễn thông

Mã số: 8.52.02.08

ĐỀ ÁN TỐT NGHIỆP THẠC SĨ

Người hướng dẫn khoa học: TS. PHẠM ANH THƯ

HÀ NỘI - NĂM 2024

LỜI CAM ĐOAN

Tôi xin cam đoan rằng, đây là công trình nghiên cứu của tôi, có sự giúp đỡ của các Giảng viên hướng dẫn là TS. Phạm Anh Thư. Các nội dung tôi nghiên cứu và kết quả trong đề tài này là hoàn toàn trung thực và chưa từng được ai công bố trong bất kỳ công trình nghiên cứu nào ở các đề án trước đây. Những số liệu thống kê được sử dụng trong đề tài được tôi thu thập từ nhiều nguồn khác nhau và có được ghi trong phần tài liệu tham khảo.

Nếu phát hiện có bất kỳ sự gian lận không đúng sự thật nào tôi xin hoàn toàn chịu trách nhiệm trước Hội đồng cũng như kết quả đề án của mình.

Hà Nội, ngày…tháng…năm 2024

Tác giả

Nguyễn Nhật Anh

MỤC LỤC

DANH TỪ VIẾT TẮT

Từ viết tắt	Nghĩa tiếng Anh	Nghĩa tiếng Việt
CBA	Customer Behavior Analysis	Thành phần phân tích hành vi khách hàng
CDR	Charging Detail Record	Bản ghi chi tiết cước
CGW	Charging Gateway	Cổng tính cước
DCC	Diameter Credit-Control	Giao thức tính cước Diameter
FTP	File Transfer Protocol	Giao thức truyền tập tin
GGSN	Gateway GPRS Support Node	Nút hỗ trợ cổng GPRS
LB	Load Balancer	Thành phần cân bằng tải
MML	Man-Machine Language	Giao thức ngôn ngữ người-máy
MS	Microservices	Vi dịch vụ
SMSC	Mobile Switching Center	Trung tâm chuyển mạch di động
OAM	Operation and Administration Management	Thành phần quản lý và vận hành
OCP	Online Charging Process	Thành phần xử lý tính cước trực tuyến
OCS	Online Charging System	Hệ thống tính cước trực tuyến
PCRF	Policy and Charging Rules Function	Chức năng quy tắc chính sách và tính phí
PGW	Provisioning Gateway	Cổng cung cấp
PRO	Provisioning	Thành phần cung cấp
REC	Recurring	Thành phần gia hạn
SCP	Service Control Point	Điểm điều khiển dịch vụ
SIU	Signaling Interface Unit	Đơn vị giao tiếp tín hiệu
SMPP	Short Message Peer-to-Peer	Giao thức tin nhắn ngắn ngang hàng
SMSC	Short Message Service Center	Trung tâm dịch vụ tin nhắn ngắn

TRIG	Trigger	Thành phần kích hoạt
VC	Voucher Card	Hệ thống thẻ cào
VGW	VAS Gateway	Cổng dịch vụ giá trị gia tăng

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

LỜI NÓI ĐẦU

Trong thời đại số hóa hiện nay, dịch vụ viễn thông đóng vai trò quan trọng trong cuộc sống hàng ngày của chúng ta. Từ việc liên lạc, truy cập internet, đến sử dụng các ứng dụng di động, chúng ta không thể phủ nhận tầm quan trọng của viễn thông và hệ thống tính cước OCS trong việc cung cấp dịch vụ này.

Để đáp ứng nhu cầu ngày càng cao của người dùng và tận dụng tối đa tiềm năng của ngành viễn thông, các nhà cung cấp dịch vụ đã không ngừng nghiên cứu và phát triển hệ thống tính cước OCS. Việc quản lý cước phí một cách hiệu quả, đáng tin cậy và linh hoạt là một thách thức không nhỏ đối với các nhà cung cấp dịch vụ viễn thông.

Bên cạnh đó, với sự phát triển mạnh mẽ của công nghệ phần mềm thì mô hình vi dịch vụ (Microservices) trở thành xu hướng tất yếu để triển khai các hệ thống viễn thông và công nghệ thông tin. Với những lợi ích mang lại như tính linh hoạt trong việc triển khai và mở rộng độc lập, khả năng dễ dàng thay thế và nâng cấp đã thúc đẩy việc phát triển, tích hợp triển khai hệ thống tại các doanh nghiệp theo mô hình Microservices.

Tôi chọn đề tài nghiên cứu "Mô hình Microservices và ứng dụng vào hệ thống tính cước tại Viettel" không chỉ nằm ở việc cải thiện quá trình tính toán cước phí, mà còn ở khả năng đáp ứng nhu cầu người dùng và tối ưu hóa quy trình kinh doanh của các doanh nghiệp trong lĩnh vực này. Những lợi ích mà mô hình Microservices mang lại khi được tích hợp vào hệ thống tính cước OCS sẽ mang đến khả năng mở rộng độc lập, có thể mở rộng chỉ các dịch vụ cần thiết thay vì phải mở rộng toàn bộ ứng dụng. Bằng cách tìm hiểu, nghiên cứu về hệ thống OCS cũng như khả năng phát triển hệ thống theo mô hình Microservices, chúng ta có thể đóng góp vào sự phát triển và tiến bộ của ngành viễn thông, từ đó tạo ra những trải nghiệm tốt hơn cho người dùng và thúc đẩy sự cạnh tranh trên thị trường.

Đề án mang tính ứng dụng thực tế, thông qua làm việc tại đơn vị và tham khảo tài liệu. Tuy nhiên, không tránh khỏi sai sót trong quá trình phát triển, tôi mong đóng góp từ các thầy cô và hội đồng để đề án được hoàn thiện hơn nữa.

Tôi xin chân thành cảm ơn!

CHƯƠNG 1: TỔNG QUAN VỀ MÔ HÌNH MICROSERVICES

1.1. Động lực thúc đẩy mô hình Microservices

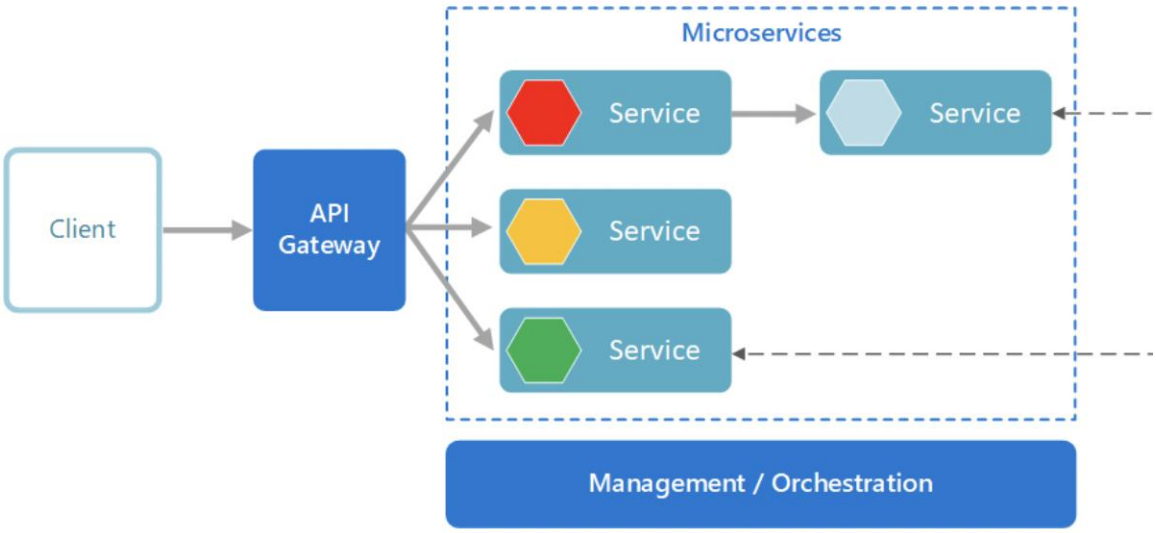
Các ứng dụng phần mềm doanh nghiệp được thiết kế để đáp ứng nhiều yêu cầu kinh doanh của doanh nghiệp. Ban đầu, các công ty thường xây dựng ứng dụng theo mô hình Monolith truyền thống. Trong mô hình này, toàn bộ ứng dụng được triển khai và chạy trên một máy chủ hoặc một nhóm máy chủ duy nhất. Tất cả các chức năng và quy trình kinh doanh được tích hợp trong một mã nguồn duy nhất.

Theo thời gian, ứng dụng Monolith có thể trở nên phức tạp và khó quản lý hơn khi kích thước và quy mô của nó tăng lên do sự phức tạp về mặt nghiệp vụ của các hệ thống phần mềm doanh nghiệp. Việc thay đổi và triển khai các tính năng mới trở nên khó khăn do sự phụ thuộc mạnh mẽ giữa các thành phần trong Monolith. Nhằm giải quyết những khó khăn này, hướng kiến trúc Microservices đã ra đời [8].

Microservices cung cấp cho các nhà phát triển một hướng tiếp cận mới trong việc xây dựng phần mềm theo hướng dịch vụ bằng cách tổ chức các thành phần phần mềm thành những module nhỏ, phân tách ứng dụng thành các dịch vụ nhỏ hơn. Các module này được coi là những ứng dụng riêng rẽ, được phát triển và triển khai độc lập, giao tiếp với nhau thông qua các API. Mỗi Microservice cung cấp một API để cho phép các microservices khác gọi và tương tác với nó. API định nghĩa các giao thức và các tập lệnh mà các microservices có thể sử dụng để trao đổi dữ liệu và thực hiện các chức năng cần thiết.

1.2. Kiến trúc Microservices

Kiến trúc Microservices là một kiến trúc phần mềm phân tán, trong đó ứng dụng được xây dựng dưới dạng một tập hợp các dịch vụ nhỏ, độc lập và phối hợp với nhau. Mỗi dịch vụ trong kiến trúc Microservices giải quyết một khía cạnh và chức năng cụ thể của ứng dụng như ghi log, tìm kiếm dữ liệu, đóng vai trò độc lập và có thể được triển khai, mở rộng và quản lý độc lập. Kiến trúc này được thể hiện chi tiết trong hình 1.1.



Hình 1.1. Mô hình Microservices

Các thành phần trong kiến trúc Microservices được mô tả chi tiết dưới đây:

Service: Kiến trúc microservices xây dựng ứng dụng dưới dạng một tập hợp các dịch vụ nhỏ, tồn tại độc lập và có khả năng thực hiện một chức năng cụ thể. Mỗi dịch vụ có thể được triển khai và quản lý độc lập, sử dụng ngôn ngữ lập trình và công nghệ phù hợp cho chức năng của mình. Các dịch vụ trong kiến trúc microservices giao tiếp với nhau thông qua các giao diện lập trình ứng dụng (API). Các API này định nghĩa các giao thức và tập lệnh mà các dịch vụ sử dụng để trao đổi dữ liệu và thực hiện các chức năng cần thiết. Giao tiếp qua API giúp các dịch vụ hoạt động độc lập và linh hoạt trong việc phát triển và triển khai.

Client: Khách hàng có thể sử dụng giao diện người dùng để tạo yêu cầu. Đồng thời, một hoặc nhiều vi dịch vụ được ủy quyền thông qua cổng API để thực hiện nhiệm vụ được yêu cầu. Kết quả là, ngay cả những vấn đề phức tạp lớn hơn đòi hỏi sự kết hợp của các vi dịch vụ cũng có thể được giải quyết tương đối dễ dàng.

API Gateway: Cổng API là điểm nhận các yêu cầu từ khách hàng. Thay vì yêu cầu trực tiếp các dịch vụ, khách hàng kết nối đến cổng API để chuyển tiếp các yêu cầu đến các dịch vụ thích hợp ở phía sau.

1.3. Những ưu điểm và nhược điểm của mô hình Microservices

1.3.1. Ưu điểm của mô hinh Microservices

Với những đặc điểm trong mô hình kiến trúc, hệ thống Microservices mang đến những những lợi ích cho các nhà phát triển và các kỹ sư trong việc phát triển và vận hành mà mô hình monolith không thể mang lại.

a) Giảm thiểu công sức phát triển

Kiến trúc Microservices cho phép phân tách ứng dụng thành các dịch vụ nhỏ độc lập. Mỗi dịch vụ chỉ cần tập trung vào một chức năng cụ thể. Điều này giảm thiểu phạm vi và phức tạp của mỗi dịch vụ, giúp đơn giản hóa quá trình phát triển và bảo trì.

Với kiến trúc Microservices, mỗi dịch vụ có thể được triển khai độc lập. Điều này đồng nghĩa với việc nhóm phát triển có thể làm việc độc lập trên từng dịch vụ mà không ảnh hưởng đến các dịch vụ khác trong hệ thống. Điều này giảm thiểu sự phụ thuộc và tăng tốc độ phát triển.

Mỗi dịch vụ có thể sử dụng công nghệ phù hợp cho mục tiêu và yêu cầu cụ thể của nó. Điều này cho phép nhóm phát triển lựa chọn công nghệ tốt nhất để giải quyết vấn đề cụ thể mà không bị ràng buộc bởi công nghệ của toàn bộ ứng dụng.

Kiến trúc Microservices khuyến khích việc tái sử dụng các dịch vụ đã phát triển trước đó. Các dịch vụ có thể được xây dựng và triển khai lại trong các ứng dụng khác nhau, giúp tiết kiệm thời gian và công sức phát triển. Ngoài ra, việc mở rộng hệ thống cũng trở nên dễ dàng hơn, vì mỗi dịch vụ có thể được mở rộng độc lập theo nhu cầu.

Với kiến trúc Microservices, các dịch vụ có thể được kiểm thử độc lập. Điều này giúp giảm thiểu sự phức tạp và thời gian kiểm thử toàn bộ hệ thống. Nhóm phát triển có thể tập trung vào việc kiểm thử từng dịch vụ một, đảm bảo tính ổn định và chất lượng của từng phần trong hệ thống.

b) Khả năng mở rộng linh hoạt

Kiến trúc Microservices cho phép khả năng mở rộng từng dịch vụ độc lập dựa trên yêu cầu tải và tài nguyên của từng dịch vụ. Nhà phát triển và vận hành có thể tập trung vào việc mở rộng chỉ những dịch vụ cần thiết mà không ảnh hưởng đến toàn bộ hệ thống. Điều này giúp tối ưu hóa việc sử dụng tài nguyên và đáp ứng linh hoạt với tải công việc biến đổi.

Microservices cho phép nhà phát triển và vận hành mở rộng hệ thống bằng cách thêm các phiên bản mới của cùng một dịch vụ và phân chia công việc giữa các phiên bản đó. Điều này giúp tăng khả năng xử lý số lượng yêu cầu lớn mà không ảnh hưởng đến hiệu suất hoặc sự ổn định của hệ thống.

Ngoài ra, kiến trúc này cho phép tối ưu hóa việc sử dụng tài nguyên bằng cách chỉ mở rộng những phần của hệ thống đòi hỏi nhiều tài nguyên hơn. Bằng cách phân tách ứng dụng thành các dịch vụ nhỏ, các kỹ sư có thể phân bổ tài nguyên dựa trên yêu cầu của từng dịch vụ, giúp tối ưu hóa việc sử dụng tài nguyên và giảm thiểu lãng phí.

Mỗi dịch vụ trong Microservices có thể được mở rộng độc lập với nhau. Điều này cho phép các kỹ sư tập trung mở rộng những phần của hệ thống cần thiết mà không ảnh hưởng đến các phần khác. Họ có thể áp dụng mức độ mở rộng khác nhau cho từng dịch vụ dựa trên yêu cầu và tiêu chí hiệu suất.

Kiến trúc Microservices cho phép nhà phát triển và vận hành linh hoạt đáp ứng với sự thay đổi trong yêu cầu và tải công việc, có thể thêm, loại bỏ hoặc thay đổi các dịch vụ một cách độc lập mà không ảnh hưởng đến toàn bộ hệ thống. Điều này giúp hệ thống dễ dàng thích ứng với môi trường kinh doanh thay đổi và tăng cường khả năng mở rộng.

c) Khả năng triển khai độc lập

Mỗi dịch vụ có thể được triển khai độc lập. Điều này cho phép nhóm phát triển triển khai các thay đổi và cập nhật chỉ trên các dịch vụ cần thiết mà không ảnh hưởng đến các phần khác trong hệ thống. Kết quả là, quá trình triển khai trở nên nhanh chóng và linh hoạt hơn, không cần chờ đợi cho toàn bộ ứng dụng được triển khai.

Với khả năng triển khai độc lập, các lỗi hoặc sự cố có thể được giới hạn chỉ trong phạm vi của một dịch vụ cụ thể. Khi có lỗi xảy ra, các kỹ sư có thể tạm dừng hoặc quay lại phiên bản trước của dịch vụ đó mà không ảnh hưởng đến các dịch vụ khác. Điều này giảm rủi ro và tác động tiêu cực lên toàn bộ hệ thống.

Kiến trúc Microservices thích hợp với các phương pháp tích hợp liên tục (CI) và triển khai liên tục (CD). Với triển khai độc lập, nhà phát triển có thể tự động hóa quá trình triển khai và tích hợp, giúp tăng nhanh tốc độ triển khai và đảm bảo tính nhất quán của hệ thống.

d) Khả năng cô lập lỗi

Trong kiến trúc Microservices, mỗi dịch vụ hoạt động độc lập với các dịch vụ khác. Điều này có nghĩa là nếu một dịch vụ gặp lỗi, lỗi chỉ ảnh hưởng đến dịch vụ đó mà không lan rộng sang các phần khác của hệ thống. Việc cô lập lỗi giúp hạn chế tác động của lỗi, giúp hệ thống tiếp tục hoạt động một cách bình thường trong khi vẫn có thể xử lý các yêu cầu từ các dịch vụ khác.

Khi một dịch vụ gặp lỗi, các dịch vụ khác trong hệ thống vẫn có thể tiếp tục hoạt động mà không bị ảnh hưởng. Điều này giúp tăng độ tin cậy của hệ thống và khả năng phục hồi sau lỗi. Nhà phát triển có thể xử lý lỗi trong dịch vụ bị lỗi mà không phải tắt toàn bộ hệ thống, đồng thời cho phép các dịch vụ khác vẫn tiếp tục phục vụ yêu cầu.

Với kiến trúc Microservices, việc xác định và sửa lỗi trở nên dễ dàng hơn. Nhà phát triển có thể xác định rõ ràng dịch vụ nào gặp lỗi và tập trung vào việc sửa lỗi trong dịch vụ đó mà không cần kiểm tra và sửa lỗi ở toàn bộ hệ thống. Điều này giúp giảm thời gian và công sức cần thiết để xử lý lỗi, đồng thời giảm thiểu rủi ro gây lỗi cho các phần khác của hệ thống.

Mỗi dịch vụ trong kiến trúc Microservices có thể được xây dựng bằng công nghệ và ngôn ngữ khác nhau. Điều này cho phép sử dụng các công nghệ tốt nhất và phù hợp nhất cho từng dịch vụ cụ thể. Đồng thời, nếu một dịch vụ gặp lỗi do sự cố công nghệ hoặc ngôn ngữ, chỉ dịch vụ đó bị ảnh hưởng và không ảnh hưởng đến các dịch vụ khác.

e) Khả năng tích hợp với các công nghệ và ngăn xếp công nghệ khác nhau

Kiến trúc Microservices cho phép sử dụng và tích hợp nhiều công nghệ khác nhau trong các dịch vụ riêng biệt. Mỗi dịch vụ có thể được xây dựng bằng ngôn ngữ lập trình và công nghệ phù hợp nhất cho nhiệm vụ cụ thể của nó. Việc này mang lại sự linh hoạt và khả năng tận dụng các công nghệ mới và phổ biến nhất cho từng thành phần của hệ thống.

Các dịch vụ trong kiến trúc Microservices hoạt động độc lập với nhau, điều này cho phép sử dụng các ngăn xếp công nghệ và phiên bản công nghệ khác nhau cho từng dịch vụ. Nhà phát triển có thể nâng cấp, cập nhật và thay đổi công nghệ trong một dịch vụ mà không ảnh hưởng đến các dịch vụ khác. Điều này giúp tối ưu hóa sự phát triển và quản lý công nghệ trong hệ thống.

Kiến trúc Microservices cung cấp các cơ chế và giao thức để tạo kết nối và trao đổi dữ liệu giữa các dịch vụ. Điều này cho phép tích hợp linh hoạt với các công nghệ và dịch vụ bên ngoài như cơ sở dữ liệu, hàng đợi tin nhắn, hệ thống xử lý sự kiện, và các dịch vụ cloud. Các kỹ sư có thể chọn công nghệ phù hợp và kết hợp chúng để xây dựng một hệ thống phức tạp và mạnh mẽ.

Ngoài ra, nó cho phép việc sử dụng và tích hợp các dịch vụ bên thứ ba một cách dễ dàng. Nhà phát triển có thể sử dụng các dịch vụ và công nghệ đã được xây dựng sẵn như thanh toán, xác thực người dùng, gửi email, và nhiều hơn nữa. Việc sử dụng các dịch vụ bên thứ ba giúp giảm thời gian và công sức phát triển, đồng thời tận dụng các chức năng đã được kiểm tra và thử nghiệm.

Mỗi dịch vụ trong kiến trúc Microservices có thể sử dụng cấu trúc dữ liệu riêng của nó. Điều này cho phép sử dụng cấu trúc dữ liệu phù hợp nhất cho nhiệm vụ cụ thể của dịch vụ đó. Người dùng có thể sử dụng cơ sở dữ liệu SQL, NoSQL hoặc các hệ thống lưu trữ dữ liệu khác cho từng dịch vụ mà không cần tuân theo một cấu trúc dữ liệu duy nhất cho toàn bộ hệ thống.

1.3.2. Nhược điểm của mô hinh Microservices

Bên cạnh những lợi ích mà mô hình kiến trúc Microservices mang lại thì kèm theo đó là những nhược điểm của kiến trúc này.

a) Sự phức tạp trong việc quản lý và triển khai

Trong kiến trúc Microservices, hệ thống được chia thành nhiều dịch vụ nhỏ độc lập, mỗi dịch vụ có quy mô nhỏ và chức năng riêng của nó. Khi số lượng dịch vụ tăng lên, việc quản lý và điều hành các dịch vụ này trở nên phức tạp. Cần phải có các công cụ và quy trình quản lý phù hợp để theo dõi và duy trì các dịch vụ này, bao gồm việc triển khai, cập nhật, và giám sát, đảm bảo rằng tài nguyên và khả năng mở rộng được phân phối đúng đắn và hiệu quả.

Với nhiều dịch vụ hoạt động độc lập, việc tích hợp và kiểm thử trở nên phức tạp hơn. Mỗi dịch vụ có thể sử dụng công nghệ và ngôn ngữ lập trình khác nhau, và cần phải đảm bảo tính tương thích và khả năng hoạt động tốt khi kết hợp với các dịch vụ khác. Việc kiểm thử phải được thực hiện cẩn thận để đảm bảo rằng các dịch vụ hoạt động đúng và tương thích với nhau. Bên cạnh đó, việc tìm kiếm và gỡ lỗi trở nên phức

tạp hơn. Khi có sự cố xảy ra hoặc khi cần tìm hiểu thông tin từ các dịch vụ cụ thể, cần phải có khả năng tìm kiếm và gỡ lỗi qua các dịch vụ khác nhau. Điều này đòi hỏi kiến thức và công cụ phù hợp để phân tích và theo dõi các bản ghi log từ các dịch vụ và tìm ra nguyên nhân gốc rễ của sự cố.

b) Chi phí phát sinh và vận hành

Xây dựng và triển khai kiến trúc Microservices đòi hỏi một công sức đáng kể và tài nguyên đầu tư. Mỗi dịch vụ trong kiến trúc phải được phát triển, kiểm thử và triển khai một cách độc lập. Điều này đòi hỏi sự hiểu biết về nhiều ngôn ngữ lập trình, công nghệ và phạm vi làm việc. Nhóm phát triển cần phải có kỹ năng và kiến thức đa dạng để làm việc trên các dịch vụ khác nhau và đảm bảo tính tương thích giữa chúng. Điều này có thể tăng chi phí phát triển và làm tăng thời gian cần thiết để hoàn thành dự án.

Kiến trúc Microservices đòi hỏi một môi trường vận hành phức tạp để quản lý và duy trì các dịch vụ. Mỗi dịch vụ cần được triển khai, cập nhật và giám sát độc lập. Điều này đòi hỏi khả năng quản lý hạ tầng và khả năng tự động hóa cao. Do đó, cần có công cụ và hệ thống quản lý phù hợp để giám sát và điều phối các dịch vụ, đảm bảo tính khả dụng và hiệu suất của hệ thống. Chi phí vận hành có thể tăng lên do việc triển khai và duy trì nhiều dịch vụ độc lập nhau.

Ngoài ra, để triển khai mô hình Microservices yêu cầu nhóm phát triển và quản lý có kiến thức sâu về các nguyên tắc và quy ước của kiến trúc này. Điều này có thể đòi hỏi đầu tư thêm vào việc đào tạo nhân viên hoặc thuê những người có kinh nghiệm trong lĩnh vực này. Ngoài ra, việc quản lý và điều phối nhiều dịch vụ độc lập cũng đòi hỏi kỹ năng quản lý cao và sự hiểu biết về các khía cạnh kỹ thuật và kinh doanh của các dịch vụ đó. Điều này cũng có thể tăng chi phí và đòi hỏi thêm công sức để quản lý và điều hành kiến trúc.

1.4. Khả năng ứng dụng của mô hình Microservices

1.4.1. Ứng dụng web và di động

Kiến trúc Microservices thích hợp cho việc xây dựng các ứng dụng web và di động phức tạp. Với việc phân tách các chức năng và logic kinh doanh thành các dịch vụ độc lập, các nhóm phát triển có thể làm việc độc lập trên từng dịch vụ. Điều này giúp tăng tốc độ phát triển và khả năng mở rộng của hệ thống. Ví dụ, một ứng dụng thương mại

điện tử có thể sử dụng các dịch vụ riêng biệt cho quản lý sản phẩm, quản lý đơn hàng, thanh toán và xác thực người dùng [7].

Một số khả năng ứng dụng của mô hình Microservices trong ứng dụng web và di động được mô tả chi tiết dưới đây:

Tính linh hoạt và mở rộng: cho phép chia nhỏ ứng dụng thành các dịch vụ độc lập mà có thể được phát triển và triển khai độc lập. Điều này mang lại tính linh hoạt cho việc phát triển và nâng cấp ứng dụng. Bằng cách tách biệt các chức năng và logic kinh doanh thành các dịch vụ riêng biệt, nhóm phát triển có thể làm việc độc lập trên từng dịch vụ mà không ảnh hưởng đến toàn bộ hệ thống. Điều này giúp tăng tốc độ phát triển và khả năng mở rộng của ứng dụng.

Tích hợp dễ dàng: cho phép tích hợp dễ dàng với các hệ thống và dịch vụ khác. Mỗi dịch vụ có thể sử dụng các giao thức mạng chuẩn để giao tiếp với các dịch vụ khác, bao gồm cả các dịch vụ bên ngoài như các API công khai hoặc dịch vụ của bên thứ ba. Điều này giúp tạo ra một hệ thống linh hoạt và có khả năng tích hợp với các thành phần bên ngoài.

Tăng khả năng phân phối và mở rộng: cho phép mở rộng dễ dàng các dịch vụ riêng lẻ mà không cần mở rộng toàn bộ hệ thống. Điều này cho phép tối ưu hóa sử dụng tài nguyên và tăng khả năng chịu tải của ứng dụng. Nếu một dịch vụ đang gặp vấn đề về hiệu suất hoặc tải, ta có thể mở rộng hoặc tăng cường dịch vụ đó mà không ảnh hưởng đến các dịch vụ khác.

Tính tin cậy và khả dụng cao: nếu một dịch vụ gặp sự cố, các dịch vụ khác vẫn có thể hoạt động mà không bị ảnh hưởng. Điều này giúp tăng tính tin cậy và khả dụng của ứng dụng. Ngoài ra, việc phân tách các chức năng giúp giảm thiểu sự phụ thuộc giữa các phần tử, giúp nâng cao khả năng chịu lỗi và khả năng khôi phục sau sự cố. Nếu một dịch vụ bị lỗi, chỉ cần tắt dịch vụ đó mà không ảnh hưởng đến các dịch vụ khác.

Quản lý và triển khai dễ dàng: giúp tách biệt quản lý và triển khai các dịch vụ riêng lẻ. Điều này giúp đơn giản hóa quy trình triển khai và quản lý hệ thống. Mỗi dịch vụ có thể được triển khai độc lập với các quy trình triển khai tự động và công cụ quản lý riêng. Điều này giúp nâng cao hiệu quả và độ tin cậy của quy trình triển khai.

Từ những khả năng ứng dụng của mô hình Microservices trong ứng dụng web và di động, các ứng dụng có thể được áp dụng bao gồm:

Hệ thống e-commerce: Trong một hệ thống e-commerce, mô hình Microservices có thể được sử dụng để phân chia các chức năng như quản lý sản phẩm, quản lý đơn hàng, thanh toán, xử lý vận chuyển thành các dịch vụ độc lập. Điều này giúp tăng khả năng mở rộng, quản lý dễ dàng và tích hợp linh hoạt với các dịch vụ bên ngoài như cổng thanh toán và dịch vụ vận chuyển.

Hệ thống xử lý dữ liệu lớn (Big Data): Trong một hệ thống xử lý dữ liệu lớn, mô hình Microservices có thể được sử dụng để phân chia các công việc xử lý dữ liệu thành các dịch vụ nhỏ độc lập. Ví dụ, một dịch vụ có thể được sử dụng để thu thập dữ liệu, một dịch vụ khác để xử lý và phân tích dữ liệu, và một dịch vụ khác để lưu trữ kết quả. Điều này giúp tăng tính linh hoạt và khả năng mở rộng của hệ thống xử lý dữ liệu lớn.

Hệ thống định vị và định tuyến: Trong một hệ thống định vị và định tuyến, mô hình Microservices có thể được sử dụng để xây dựng các dịch vụ cung cấp chức năng như tìm kiếm địa điểm, tính toán đường đi, thông báo lưu lượng giao thông. Các dịch vụ này có thể được triển khai độc lập và tương tác với nhau để tạo ra các dịch vụ định vị và định tuyến mạnh mẽ.

1.4.2. Hệ thống tài chính và ngân hàng

Các hệ thống tài chính và ngân hàng thường có nhiều chức năng và dịch vụ phức tạp. Kiến trúc Microservices cho phép phân tách các chức năng như quản lý tài khoản, quản lý giao dịch, xử lý thanh toán và phân tích dữ liệu thành các dịch vụ độc lập. Điều này giúp tăng tính linh hoạt và khả năng mở rộng của hệ thống, đồng thời giảm thiểu sự phụ thuộc giữa các phần tử [7].

Một số khả năng ứng dụng của mô hình Microservices trong hệ thống tài chính và ngân hàng bao gồm:

Quản lý người dùng và quyền truy cập: cho phép tách biệt quyền truy cập và quản lý người dùng trong hệ thống tài chính và ngân hàng. Các dịch vụ riêng biệt có thể được tạo ra để quản lý thông tin người dùng, xác thực và phân quyền. Điều này giúp đảm bảo tính bảo mật và kiểm soát quyền truy cập vào các dịch vụ và tài khoản khác nhau.

Quản lý tài khoản và thanh toán: cho phép xây dựng các dịch vụ độc lập để quản lý tài khoản ngân hàng và hoạt động thanh toán. Các dịch vụ này có thể bao gồm kiểm tra số dư, chuyển tiền, thanh toán hóa đơn và xử lý giao dịch. Bằng cách phân chia chức năng thành các dịch vụ riêng biệt, hệ thống trở nên linh hoạt và có khả năng mở rộng để đáp ứng nhu cầu người dùng.

Quản lý rủi ro và phân tích tín dụng: Trong hệ thống tài chính và ngân hàng, việc quản lý rủi ro và phân tích tín dụng là rất quan trọng. Mô hình Microservices cho phép xây dựng các dịch vụ độc lập để theo dõi và phân tích dữ liệu liên quan đến rủi ro tín dụng, danh sách đen, đánh giá khách hàng và các chỉ số tài chính. Nhờ đó, các dịch vụ này có thể cung cấp thông tin quan trọng để hỗ trợ quyết định về tín dụng và quản lý rủi ro trong hệ thống.

Giao tiếp với bên thứ ba: cung cấp khả năng tích hợp và giao tiếp với các bên thứ ba trong hệ thống tài chính và ngân hàng. Ví dụ, có thể có các dịch vụ riêng biệt để giao tiếp với các tổ chức thanh toán, hệ thống rút tiền tự động (ATM), hoặc các dịch vụ thanh toán điện tử khác. Điều này giúp tạo ra tính linh hoạt và khả năng tích hợp với hệ thống bên ngoài.

Bảo mật và tuân thủ quy định: Trong hệ thống tài chính và ngân hàng, bảo mật và tuân thủ quy định là yếu tố quan trọng. Mô hình Microservices cho phép áp dụng các biện pháp bảo mật như xác thực hai yếu tố, mã hóa dữ liệu và giám sát bảo mật. Ngoài ra, các dịch vụ riêng biệt cũng có thể được thiết kế để tuân thủ các quy định và chuẩn an ninh như PCI-DSS để đảm bảo tính an toàn và tuân thủ quy định trong ngành.

1.4.3. Hệ thống y tế

Trong lĩnh vực y tế, kiến trúc Microservices có thể được áp dụng để tách riêng các chức năng như quản lý bệnh nhân, quản lý lịch hẹn, xử lý hồ sơ y tế và gửi thông báo. Điều này giúp tăng tính linh hoạt và khả năng mở rộng của hệ thống y tế, đồng thời cho phép tích hợp dữ liệu và tương tác với các ứng dụng và thiết bị y tế khác [7].

Một số khả năng ứng dụng của mô hình Microservices trong hệ thống y tế được thể hiện như sau:

Quản lý hồ sơ bệnh nhân: cho phép xây dựng các dịch vụ độc lập để quản lý hồ sơ bệnh nhân. Các dịch vụ này có thể chịu trách nhiệm về đăng ký bệnh nhân, lưu

trữ thông tin y tế, quản lý lịch sử bệnh án và cung cấp thông tin y tế cho các dịch vụ khác. Việc phân chia chức năng này giúp tăng tính linh hoạt và khả năng mở rộng của hệ thống.

Quản lý lịch hẹn và đặt lịch khám: tạo ra các dịch vụ riêng biệt để quản lý lịch hẹn và đặt lịch khám. Các dịch vụ này có thể tổ chức và quản lý lịch trình của bác sĩ, cho phép bệnh nhân đặt lịch khám và nhận thông báo về lịch hẹn. Điều này giúp cải thiện quản lý thời gian và tiện lợi cho bệnh nhân và nhân viên y tế.

Hỗ trợ chẩn đoán và quản lý bệnh: xây dựng các dịch vụ độc lập để hỗ trợ chẩn đoán và quản lý bệnh. Các dịch vụ này có thể cung cấp công cụ phân tích dữ liệu y tế, hỗ trợ tư vấn y tế từ xa, cung cấp hướng dẫn điều trị và theo dõi sự tiến triển của bệnh nhân. Điều này giúp cải thiện chất lượng chăm sóc y tế và tăng khả năng chẩn đoán chính xác.

Quản lý dược phẩm và thuốc: quản lý dược phẩm và thuốc một cách hiệu quả. Các dịch vụ độc lập có thể quản lý quy trình đặt hàng, kiểm tra và lưu trữ thông tin về thuốc, quản lý kho thuốc và cung cấp thông tin về liều lượng và tương tác thuốc. Điều này giúp đảm bảo an toàn và chính xác trong việc sử dụng thuốc và quản lý nguồn lực dược phẩm.

1.5. Kết luận chương 1

Mô hình Microservices với những ưu điểm vượt trội trong việc phát triển phát mềm và vận hành hệ thống đang ngày càng trở nên phổ biến. Nó giúp tăng tính linh hoạt, khả năng mở rộng và sử dụng công nghệ đa dạng giúp các nhà phát triển và vận hành tiết kiệm thời gian và công sức hơn rất nhiều so với các hệ thống triển khai theo mô hình Monolith. Tuy nhiên, mô hình Microservices cũng có những thách thức như việc quản lý và triển khai nhiều dịch vụ độc lập mang lại nhiều sự phức tạp hơn do đó đòi hỏi sự quản lý và triển khai kỹ lưỡng của đội ngũ kỹ sư công nghệ thông tin. Dẫu vậy thì với những lợi ích lớn mà kiến trúc Microservices mang lại, nó vẫn hứa hẹn sẽ là kiểu mô hình được áp dụng rộng rãi trong kiến trúc phần mềm trong tương lai.

CHƯƠNG 2: HỆ THỐNG TÍNH CƯỚC TRỰC TUYẾN OCS

2.1. Tổng quan về hệ thống OCS

Hệ thống tính cước trực tuyến OCS là một hệ thống trong lĩnh vực viễn thông và dịch vụ di động, được sử dụng để tính toán và quản lý thông tin về cước phí và sử dụng dịch vụ của khách hàng.

Hệ thống OCS đóng vai trò là trái tim của mạng lưới và thực hiện một số chức năng như sau:

Quản lý cước phí: tính toán và quản lý cước phí dịch vụ. Khi khách hàng sử dụng dịch vụ, hệ thống OCS sẽ ghi nhận thông tin về việc sử dụng và áp dụng các quy tắc tính cước để tính toán số tiền phải trả. Các quy tắc tính cước có thể được cấu hình để phù hợp với các gói cước, dịch vụ và chính sách của nhà cung cấp.

Quản lý tài khoản và sử dụng dịch vụ: cung cấp chức năng quản lý tài khoản và sử dụng dịch vụ cho khách hàng. Nó có thể ghi nhận thông tin về tài khoản của khách hàng, bao gồm thông tin cá nhân, gói cước và lịch sử sử dụng dịch vụ. Hệ thống cũng có khả năng kiểm tra và xác minh quyền truy cập của khách hàng đối với các dịch vụ cụ thể.

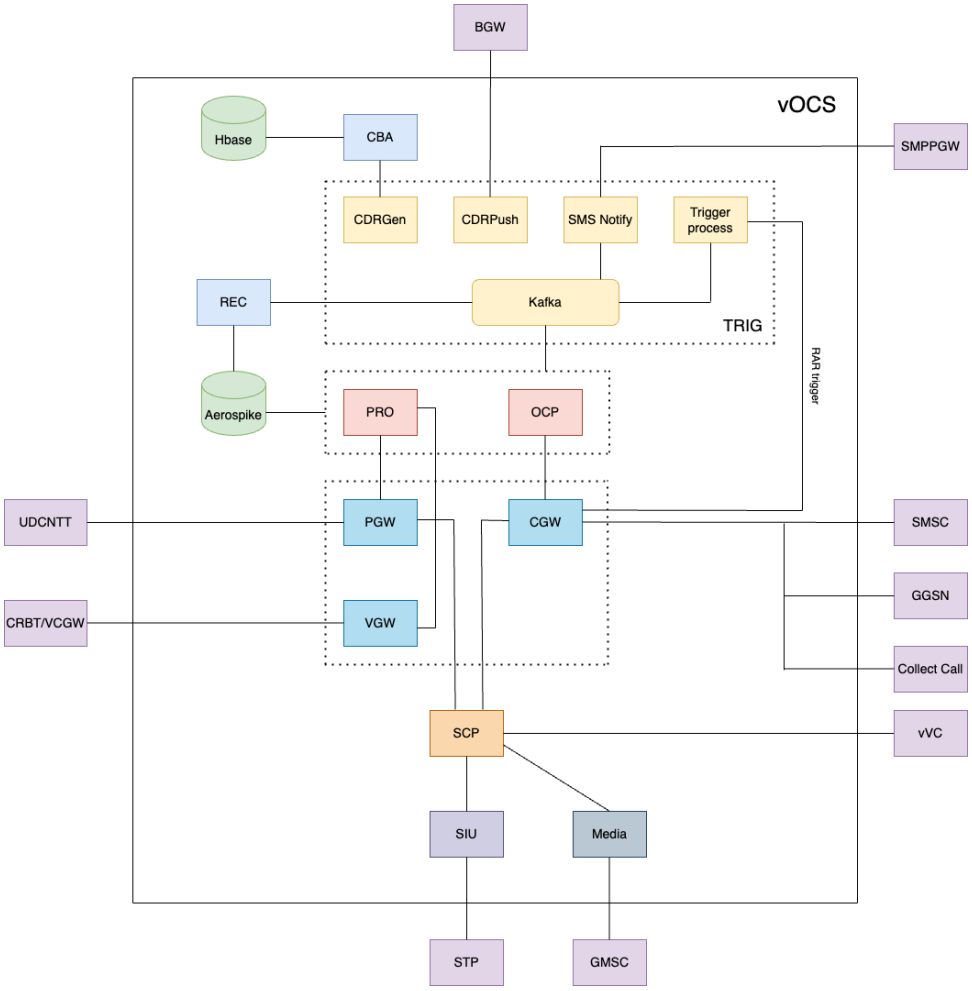
Quản lý thời gian và tương tác thời gian thực: xử lý và tính toán cước phí theo thời gian thực. Khi khách hàng sử dụng dịch vụ, hệ thống OCS sẽ theo dõi thời gian và tính toán cước phí dựa trên thời gian thực tại thời điểm sử dụng. Điều này cho phép nhà cung cấp dịch vụ áp dụng cước phí linh hoạt và chính xác theo từng thời điểm.

Tích hợp với hệ thống khác: có khả năng tích hợp với các hệ thống khác trong mạng viễn thông. Nó có thể kết nối và giao tiếp với các hệ thống như hệ thống quản lý thuê bao (BSS), hệ thống hóa đơn (Billing System), hệ thống xác thực (Authentication System) và hệ thống quản lý dịch vụ (Service Management System). Điều này giúp đảm bảo tính toàn vẹn và chính xác trong việc tính toán cước phí và quản lý thông tin khách hàng.

Quản lý sự kiện và báo cáo: ghi nhận và quản lý các sự kiện liên quan đến việc sử dụng dịch vụ của khách hàng. Nó có khả năng tạo ra báo cáo và thống kê về thông tin cước phí, sử dụng dịch vụ và các hoạt động khác của khách hàng. Báo cáo này

giúp nhà cung cấp dịch vụ có cái nhìn tổng quan về hoạt động kinh doanh và hỗ trợ quyết định về chiến lược cước phí và dịch vụ.

2.2. Kiến trúc hệ thống OCS truyền thống



Hình 2.1. Mô hình hệ thống OCS theo kiến trúc Monolith

Hình 2.1 mô tả mô hình hệ thống OCS theo kiến trúc Monolith. Hệ thống OCS theo kiến trúc Monolith tập trung tất cả các dịch vụ trong một khối kiến trúc duy nhất bao gồm các dịch vụ như thoại, data, sms, gia hạn hay quản lý thông tin thuê bao và giao tiếp với các hệ thống mạng lõi như GGSN, SMSC hay MSC để tính cước trong quá trình sử dụng dịch vụ của người dùng [3].

Các cơ sở dữ liệu sử dụng trong hệ thống OCS để lưu trữ lượng lớn thông tin thuê bao liên quan đến tài khoản, gói cước sử dụng, các bản ghi chi tiết cước CDR hay chính sách kinh doanh để vận hành hệ thống.

2.2.1. Các thành phần trong hệ thống

Hệ thống OCS Viettel bao gồm các thành phần chính như sau:

SIU: cung cấp kết nối giữa mạng lõi SS7 và các máy chủ ứng dụng phân tán.

SCP: là một tập hợp các cơ sở dữ liệu lưu giữ các thông tin cần thiết để cung cấp các dịch vụ phức tạp hơn so với điều khiển cuộc gọi thông thường (cung cấp VAS, các dịch vụ tổng đài 1800), kết nối mạng SS7 với hệ thống cơ sở dữ liệu.

Media: là thành phần lưu trữ và xử lý các file âm báo của tất cả luồng nghiệp vụ trên OCS.

vVC: hệ thống quản lý thẻ cào, chứa các thông tin khuyến mại, ưu đãi.

CGW: gateway kết nối tới các thành phần tính cước chính trong khối OCU, chuyển đổi các giao thức Diameter/SMPP+ sang giao thức nội bộ trong hệ thống OCS, chuyển các bản tin thành bản tin craMsg.

PGW: gateway kết nối tới các hệ thống TRAN, VAS, Provisioning, UDCNTT, chuyển đổi các giao thức thành giao thức nội bộ, chuyển các bản tin thành bản tin craMsg.

VGW: chuyển tiếp các tin nhắn dành cho các dịch vụ VAS.

OCP: chạy theo cơ chế Active-Active, xử lý các bản tin craMsg nhận được từ CGW, thực hiện chức năng tính cước thời gian thực cho tất cả các dịch vụ và điều khiển băng thông cho các dịch vụ data (Mobile data, FTTH, ADSL), tất cả các server có vai trò như nhau và được nhận roundrobin bản tin từ Gateway.

PRO: thực hiện các chức năng đấu nối, đăng ký/hủy thuê bao, cộng/trừ tiền tài khoản, xử lý các dịch vụ VAS, USSD, IVR, Web-Service.

REC: thực hiện gia hạn, chuyển đổi vòng đời, xuất dữ liệu thuê bao (trạng thái thuê bao, số dư tài khoản).

TRIG: xử lý các bản tin trigger, chuyển đổi thành CDR, craMsg, RAR, smssnotify.

PCRF: hỗ trợ detect luồng dịch vụ data, thực thi chính sách và tính cước dựa trên luồng (flow-based), cho phép kiểm soát dịch vụ tốt hơn phù hợp với nguồn lực tài chính.

CBA: quản lý và phân tích thông tin hành vi khách hàng, đưa ra các chính sách hoặc các gói cước phù hợp cho khách hàng.

OAM: phân hệ hỗ trợ vận hành, thống kê và giám sát toàn bộ tiến trình, node mạng của hệ thống OCS.

2.2.2. Các giao thức trong hệ thống

Các giao thức được hỗ trợ sử dụng trong hệ thống OCS bao gồm các giao thức sau:

SMPP: giao thức truyền một lượng lớn tin nhắn SMS, được sử dụng để kết nối các hệ thống bên ngoài với trung tâm xử lý lượng lớn SMS (cụ thể là SMSC).

SMPP+: là một phiên bản độc quyền của SMPP dành cho các dịch vụ USSD

MML (Man-Machine Language): hỗ trợ kết nối tới phân hệ SCU của nghiệp vụ USSD, IVR và hỗ trợ kết nối của các tổng đài TRAN/VAS và tổng đài thẻ nạp.

DCC: giao thức Diameter được sử dụng để tính cước thời gian thực cho các dịch vụ người dùng (truy nhập mạng, dịch vụ khởi tạo phiên, dịch vụ tin nhắn và các dịch vụ truy cập dữ liệu khác, điều khiển tín dụng và giá cước thời gian thực.

Sigtran: truyền các dữ liệu báo hiệu thời gian thực qua mạng IP, cho phép các nút phía mạng IP giao tiếp với các nút phía mạng SS7, cho phép các nút SS7 có thể giao tiếp với nhau qua các link IP.

Gx: kết nối với GGSN, hỗ trợ các nghiệp vụ PCRF.

Gy: kết nối với GGSN, hỗ trợ các nghiệp vụ liên quan đến tính cước nghiệp vụ Data.

FTP: trao đổi file CDR với các hệ thống khác.

2.2.3. Các tiến trình trong hệ thống

Hệ thống OCS bao gồm các tiến trình xử lý từng nghiệp vụ viễn thông, cụ thể như sau:

ProvisioningProcess: phân hệ thực hiện xử lý các nghiệp vụ đấu nối, kích hoạt/hủy thuê bao, đăng ký các dịch vụ VAS đăng ký, thay đổi, hủy gói cước, tạo/hủy nhóm, thêm thành viên, nạp tiền, thay đổi/truy vấn tài khoản, quản lý thông tin zone, các parameter của thuê bao và hệ thống, recurring.

OfflineProcess: thực hiện tính cước offline cho khách hàng hoặc refund bằng cách tạo ra các bản tin tính cước từ CDROffline hoặc CDRRefund.

TriggerProcess: phân loại trigger liên quan đến Notify tới khách hàng và các trigger liên quan đến tạo ra các event tác động tới profile khách hàng (công tích lũy, đăng

ký gói cước, đầu ra là các sự kiện thông báo (SMS_Notify, USSD_Notify) để gửi tới phân hệ SMS Notify (bổ sung các thông tin còn thiếu cho bản tin thông báo, phân loại bản tin thông báo theo thời gian gửi) hoặc các sự kiện tính cước để gửi đến phân hệ Gateway.

CDRGen: xử lý tin nhắn trong hàng đợi xử lý trigger Kafka liên quan đến các CDR Event, tạo CDR file và đẩy đến CBA.

CDRPush: đẩy các file CDR được tạo tới BGW qua giao thức FTP.

SMS Notify: đẩy SMS qua SMSC đến khách hàng đồng thời đẩy tin nhắn đến hàng đợi Kafka trong CBA để lưu trữ lịch sử tin nhắn.

2.2.4. Các cơ sở dữ liệu

Các hệ thống cơ sở dữ liệu có khả năng lưu trữ dữ liệu lớn được dùng để lưu trữ phục vụ việc tra cứu bao gồm các thành phần như sau:

MySQL: lưu trữ chính sách nghiệp vụ, cấu hình hệ thống, gói cước, thao tác liên quan đến lịch sử của thuê bao, ưu điểm là miễn phí và đảm bảo được hết các tính năng của 1 CSDL quan hệ.

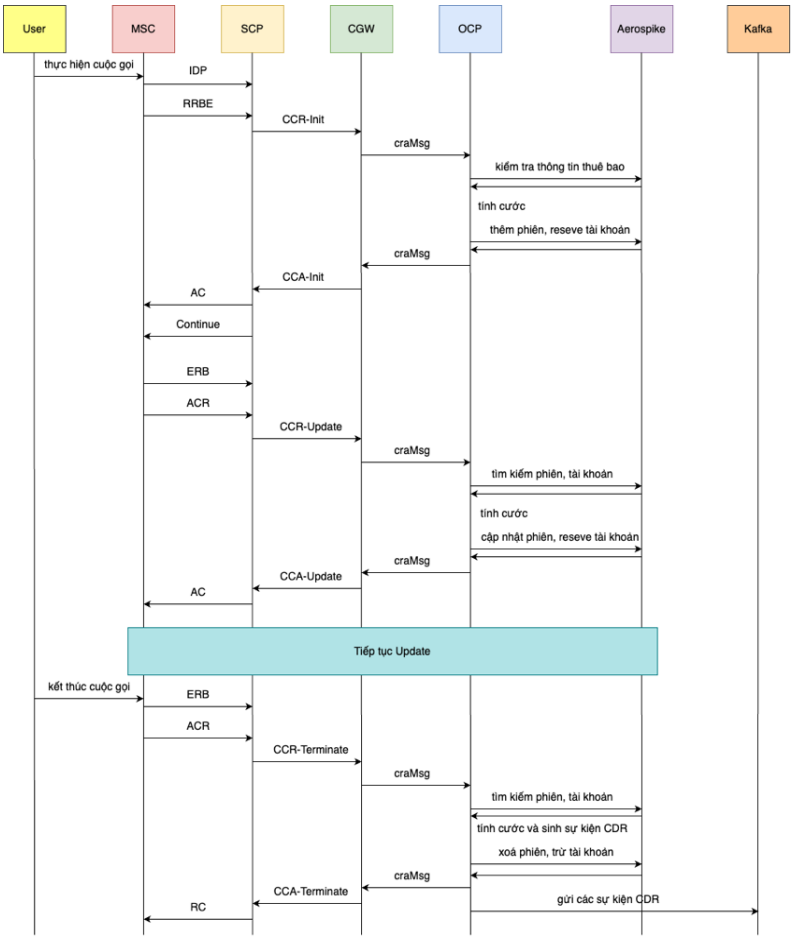
Aerospike (NoSQL): lưu trữ thông tin khách hàng (số thuê bao, trạng thái, gói cước, tài khoản, tham số khác).

Hbase: cơ sở dữ liệu có khả năng lưu trữ và truy xuất dữ liệu lớn với hiệu suất cao và khả năng mở rộng linh hoạt nên được dùng để lưu trữ các CDR sinh ra từ các nghiệp vụ tính cước của OCP, PRO, REC.

2.3. Các luồng xử lý tính cước trong hệ thống OCS

2.3.1. Luồng tính cước dịch vụ thoại

Hình 2.2 mô tả chi tiết luồng tính cước của dịch vụ thoại cơ bản và các bước thực hiện được phân chia thành ba giai đoạn bao gồm thiết lập cuộc gọi đầu cuối, theo dõi tình trạng cuộc gọi, kết thúc cuộc gọi [3].



Hình 2.2. Luồng tính cước dịch vụ thoại

- **Thiết lập cuộc gọi đầu cuối (Init, khi thuê bao thực hiện cuộc gọi)**

Bước 1: Khi MSC xác định cần phải có thêm hướng dẫn để định tuyến cuộc gọi thì nó sẽ gửi IDP đến SCP.

Bước 2: MSC gửi bản tin RRBE lên SCP để yêu cầu trạng thái cuộc gọi. Sau đó SCP gửi bản tin CCR-Init (gồm thông tin CC-Time, địa chỉ MSC, thông tin thuê bao). Khi CGW nhận được bản tin từ SCP, nó sẽ chuyển thành bản tin nội bộ craMsg, sau đó gửi đến OCP.

Bước 3: OCP giải mã bản tin, lấy thông tin MSISDN, thời gian bắt đầu cuộc gọi.

Bước 4: OCP gửi yêu cầu đến Aerospike để kiểm tra thông tin thuê bao.

Bước 5: OCP tính toán cước phí cho thuê bao.

Bước 6: nếu thuê bao thỏa mãn điều kiện để thực hiện cuộc gọi (active và đủ tiền) thì thực hiện create session và reserve bal vào trong các dữ liệu của IMDB AeroSpike.

Bước 7: OCP gửi bản tin phản hồi tới CGW, sau đó CGW gửi CCA-Init (gồm Initial, GSU (CC-Time), Return-Code) về SCP.

Bước 8: SCP bản tin AC, bản tin Continue đến MSC để thiết lập cuộc gọi, MSC tập hợp thông tin của called party, cuộc gọi thoại giữa thuê bao nhận và thuê bao gọi được kết nối.

- **Theo dõi tình trạng cuộc gọi (Update)**

Bước 9: MSC gửi ERB tới SCP để lấy thông tin trạng thái cuộc gọi. Trong quá trình đàm thoại, MSC gửi bản tin ACR tới SCP để yêu cầu cập nhật tính cước.

Bước 10: CGW chuyển lời gọi nội bộ và gửi CCR-Update (gồm RSU (CC-Time), MSC-Address, USU (CC-Time), thông tin thuê bao) tới OCP yêu cầu reserve thời gian gọi cho thuê bao.

Bước 11: OCP thực hiện chọn Session, Balance từ IMDB Aerospike (nếu tài khoản thuê bao đủ tiền thì reserve cho thuê bao).

Bước 12: OCP thực hiện tính cước, reserve cho thuê bao để tính thời gian thuê bao được phép gọi.

Bước 13: OCP thực hiện update Session và Reserve Bal trên IMDB Aerospike sau đó gửi bản tin CCA-Update tới CGW.

Bước 14: CGW gửi bản tin CCA (Update, GSU (CC-Time), Return-Code) tới SCP.

Bước 15: SCP chuyển bản tin đến báo hiệu SS7 và gửi bản tin AC (Apply Charging) tới MSC để yêu cầu thời gian reserve cho cuộc gọi.

Bước 16: quá trình cập nhật diễn ra như trên đến khi thuê bao kết thúc đàm thoại.

- **Kết thúc cuộc gọi (Terminate, thuê bao ngắt cuộc gọi)**

Bước 17: MSC gửi bản tin ERB và ACR đến SCP để giải phóng phiên làm việc.

Bước 18: CGW gửi bản tin CCR-Terminate (gồm USU (CC-Time), thông tin thuê bao) đến OCP để thực hiện tính cước lại cho thuê bao.

Bước 19: OCP thực hiện chọn Session, Balance từ IMDB Aerospike để lấy thông tin tài khoản thuê bao.

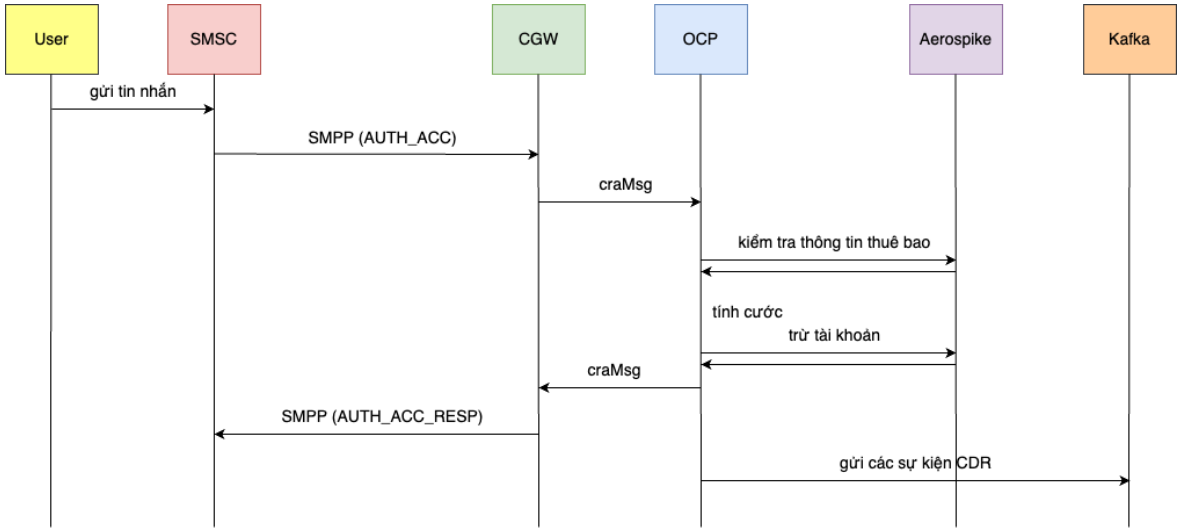
Bước 20: OCP thực hiện rating để tính cước lại cuộc gọi của thuê bao và sinh CDR Event.

Bước 21: OCP thực hiện trừ tiền tài khoản, xóa session trên IMDB Aerospike sau đó gửi CDR Event tới Kafka.

Bước 22: CGW nhận lời gọi nội bộ và gửi bản tin CCA-Terminate (gồm Return-Code) tới SCP.

Bước 23: SCP chuyển bản tin tới báo hiệu SS7 và gửi bản tin RC (Release Call) tới MSC, toàn bộ quá trình thoại kết thúc.

2.3.2. Luồng tính cước dịch vụ SMS



Hình 2.3. Luồng tính cước dịch vụ SMS

Hình 2.3 mô tả luồng tính cước dịch vụ SMS khi người dùng gửi tin nhắn. Khác với luồng thoại tính cước theo phiên, luồng SMS và MMS thì OCS giao tiếp với các phần tử mạng tương ứng để lấy các thông tin phục vụ việc tính cước, việc xử lý tính cước tuân theo phương thức tính cước theo sự kiện (event) [4].

Các bước tính cước luồng dịch vụ tin nhắn bao gồm:

Bước 1: Khi khách hàng gửi tin nhắn, MSC sẽ chuyển yêu cầu này tới SMSC.

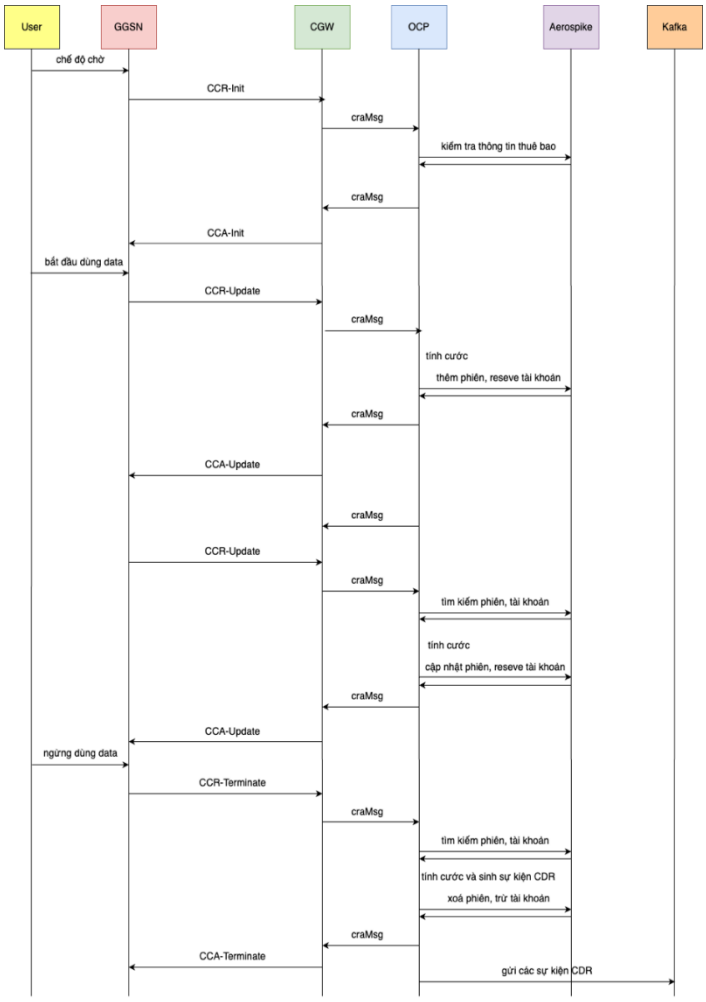
Bước 2: SMSC phân tích các thuộc tính trong yêu cầu dịch vụ nhận được (bảng giá, mã dịch vụ) sau đó chuyển thông tin tới CGW trong yêu cầu tính cước (SMPP).

Bước 3: CGW nhận bản tin và chuyển giao thức SMPP thành giao thức tính cước, sau đó gửi bản tin CCR (Request-Action = 0) tới OCP.

Bước 4: OCP kiểm tra thông tin thuê bao và tài khoản trong IMDB Aerospike. Nếu thuê bao thỏa mãn điều kiện hoạt động 2 chiều và đủ tiền thì sẽ trừ tiền tài khoản. Sau đó, OCP mã hóa bản tin và gửi bản tin CCA (deduct, Return-code) tới CGW.

Bước 5: CGW phản hồi lại kết quả tính cước cho SMSC. Sau khi nhận được kết quả tính cước từ CRA, SMSC phản hồi lại cho khách hàng và cung cấp dịch vụ.

2.3.3. Luồng tính cước dịch vụ Data



Hình 2.4. Luồng tính cước dịch vụ data

Hình 2.4 thể hiện luồng tính cước dịch vụ Data. Các bước trong luồng xử lý tính cước bao gồm từ khi thuê bao để ở chế độ chờ chưa sử dụng đến dịch vụ đến khi ngừng sử dụng data. Luồng tính cước data được tính cước trên giao diện mặt phẳng gy [3].

- **Khi thuê bao ở chế độ chờ (Init)**

Bước 1: Thuê bao nằm trong hệ thống OCS thực hiện truy cập data, GGSN sẽ kiểm tra các thuộc tính tính cước cho dịch vụ này và gửi một bản tin khởi tạo phiên lên OCS.

Bước 2: CGW nhận được yêu cầu và chuyển tới thành phần nội bộ, sau đó bản tin được gửi đến OCP.

Bước 3: OCP giải mã bản tin, lấy thông tin MSISDN, sau đó truy vấn IMDB AeroSpike để kiểm tra thông tin thuê bao và gửi bản tin CCA cho CGW.

Khác với các dịch vụ khi sử dụng là tính cước ngay (Voice, SMS) thì data sẽ chỉ thực hiện tính cước khi khách hàng sử dụng truy cập dữ liệu mà không phải để ở chế độ chờ.

Bước 4: CGW gửi bản tin CCA tới GGSN để trả lại thông tin xác thực thuê bao cho GGSN (khi khách hàng bật dịch vụ data nhưng mới chỉ để ở chế độ chờ chưa sử dụng để truy cập).

- **Khi khách hàng sử dụng dịch vụ (không ở chế độ chờ, Update)**

Bước 5: GGSN gửi bản tin CCR (Update –lần 1) cho CGW để yêu cầu reserve data.

Bước 6: CGW nhận được yêu cầu và chuyển đến thành phần nội bộ, sau đó bản tin được gửi đến OCP.

Bước 7: OCP phân tích thông qua luồng Pre-Processing để lấy được event tính cước. Sau đó OCP thực hiện luồng rating để tính toán cước phí cho thuê bao.

Bước 8: OCP thực hiện tạo session truy nhập và reserve tài khoản vào trong IMDB AeroSpike sau đó gửi bản tin phản hồi CCA cho CGW.

Bước 9: Sau khi OCP mã hóa bản tin gửi về CRA Server, CGW gửi CCA (Update, GSU (CC-Time), Return-code) tới GGSN.

Bước 10: Khi thuê bao sử dụng hết lượng reserve hoặc reserve hết hiệu lực, GGSN gửi bản tin CCR Update - lần tiếp theo tới CGW. Bản tin chứa thông tin về lưu lượng data thuê bao đã sử dụng sau bản tin (CCA-Update-1).

Bước 11: CGW nhận được yêu cầu và chuyển đến thành phần nội bộ, sau đó bản tin được gửi đến OCP.

Bước 12: OCP lấy thông tin phiên truy nhập và thông tin tài khoản thuê bao từ IMDB Aerospike để chuẩn bị thực hiện luồng rating.

Bước 13: OCP thực hiện luồng rating, dựa vào thông tin lưu lượng data thuê bao sử dụng ở bước 10 (nằm trong CCR-Update) để tính cước cho thuê bao. Sau khi hoàn thành tính cước, OCP dựa vào thông tin tài khoản hiện tại và cấu hình giá để thực hiện reserve thêm lưu lượng data cho thuê bao.

Bước 14: OCP thực hiện cập nhật lại session và reserve tài khoản trên IMDB AeroSpike, sau đó gửi bản tin CCA tới CGW.

Bước 15: CGW gửi bản tin CCA (Update, GSU, Return-Code) tới GGSN để thông báo lưu lượng data được cấp phát cho thuê bao.

- **Khi thuê bao ngừng sử dụng dịch vụ (Terminate)**

Bước 16: Khi khách hàng dừng sử dụng dịch vụ, GGSN gửi bản tin CCR (Terminate) tới CGW để yêu cầu tính lại cước cho thuê bao.

Bước 17: CGW nhận được yêu cầu và chuyển tới thành phần nội bộ, sau đó bản tin được gửi tới OCP.

Bước 18: OCP thực hiện lấy thông tin session và thông tin tài khoản của thuê bao từ IMDB AeroSpike để chuẩn bị cho luồng tính cước.

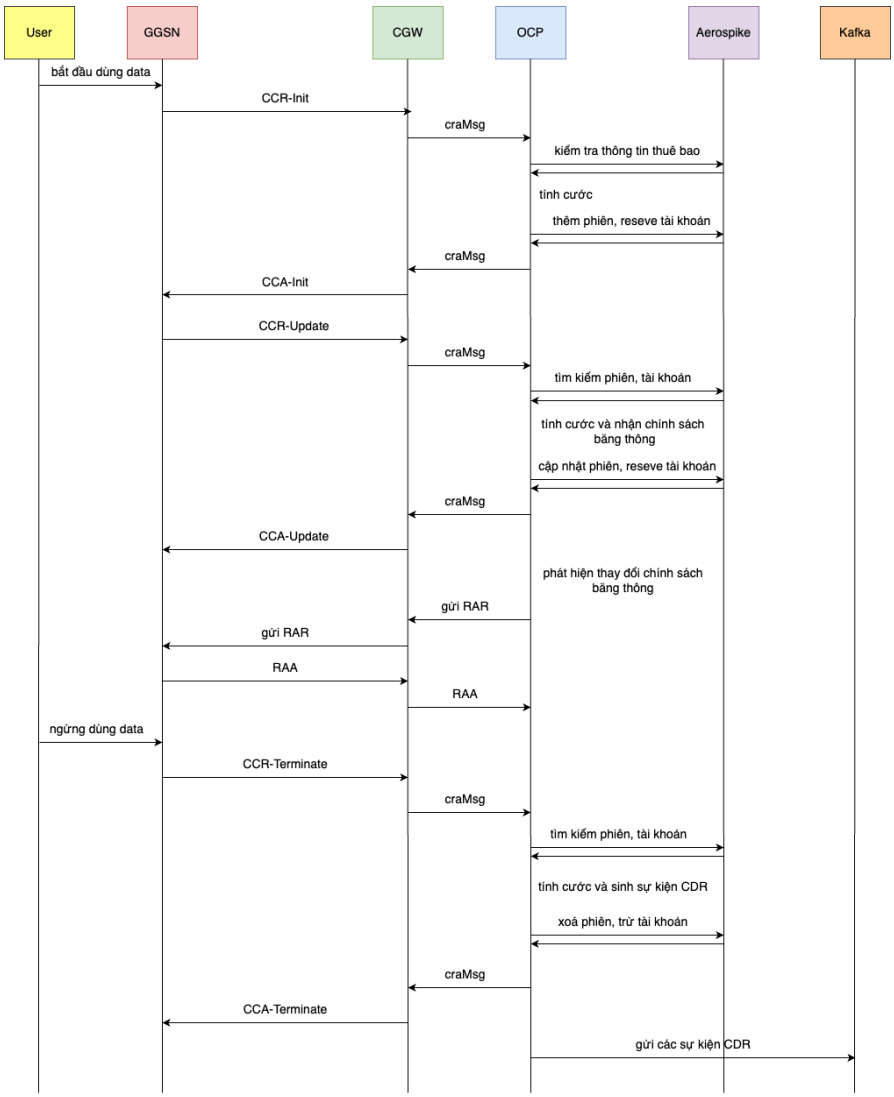
Bước 19: OCP thực hiện luồng rating để tính cước cho thuê bao và sinh CDR Event.

Bước 20: OCP thực hiện trừ tiền tài khoản, xóa session trên IMDB AeroSpike. Sau đó, OCP gửi CDR Event tới Kafka để sinh ra file CDR lưu thông tin về chi tiết cước và gửi bản tin CCA tới CGW.

Bước 21: CGW gửi bản tin CCA (Terminate, Return-Code) tới GGSN. Toàn bộ quá trình data kết thúc.

2.3.4. Luồng tính cước dịch vụ PCRF

Hình 2.5 thể hiện luồng tính cước dịch vụ PCRF, hoạt động song song với luồng tính cước dịch vụ data (khi sử dụng data thì chức năng PCRF cũng sẽ hoạt động để đưa ra các chính sách tính cước phù hợp) [3]. Luồng tinh cước dịch vụ PCRF được tính cước trên giao diện mặt phẳng gx. Các bước xử lý tính cước luồng PCRF được mô tả như bên dưới.



Hình 2.5. Luồng tính cước dịch vụ PCRF

- **Luồng tính cước PCRF (Init)**

Bước 1: Thuê bao nằm trong hệ thống OCS thực hiện truy cập data, GGSN gửi bản tin CCR-Init, thông tin thuê bao tới CGW.

Bước 2: CGW nhận được yêu cầu và chuyển tới server PCRF.

Bước 3: PCRF thực hiện giải mã bản tin, gọi IMDB AeroSpike để kiểm tra thông tin thuê bao và gói cước.

Bước 4: PCRF tính toán cước phí và ra chính sách policy băng thông cho thuê bao qua Rating.

Bước 5: Thực hiện create session và reserve balance vào trong các bảng của database IMDB AeroSpike.

Bước 6: PCRF gửi trả lại trả lời CCA-Init về CGW.

Bước 7: CGW gửi CCA-Init từ PCRF trả về cho GGSN.

- **Luồng tính cước PCRF (Update)**

Bước 8: Khi thuê bao sử dụng hết lượng reserve hoặc reserve hết hiệu lực, GGSN gửi bản bản tin CCR-Update tiếp theo tới CGW.

Bước 9: CGW nhận được yêu cầu và chuyển tới PCRF.

Bước 10: PCRF thực hiện giải mã bản tin, gọi IMDB Aerospike để kiểm tra thông tin thuê bao và gói cước.

Bước 11: PCRF thực hiện tìm chọn phiên, tài khoản từ IMDB Aerospike.

Bước 12: PCRF thực hiện rating, reserve và lấy chính sách băng thông policy cho thuê bao.

Bước 13: PCRF thực hiện cập nhật phiên, trừ và resere tài khoản trên IMDB Aerospike, sau đó gửi bản tin CCA-Update tới CGW.

Bước 14: PCRF gửi bản tin CCA-Update trả về tới GGSN.

Trong khi người dùng sử dụng data, sẽ có nhiều bản tin CCR update nên từ bước 8 đến bước 14, sẽ được lặp lại nhiều lần.

Bước 15: Trong quá trình sử dụng nếu PCRF phát hiện ra có thay đổi chính sách policy băng thông (detect policy change) thì gửi RAR để yêu cầu GGSN thay đổi chính sách.

Chính sách thay đổi có thể xảy ra trong trường hợp băng thông thay đổi (hết lưu lượng data gói cước, thay đổi gói cước) dẫn đến chính sách băng thông (bandwidth policy) cũng thay đổi.

Bước 16: PCRF gửi RAR cho CGW.

Bước 17: CGW gửi RAR từ PCRF cho GGSN.

Bước 18: GGSN gửi RAA trả lời yêu cầu chính sách của PCRF cho CGW.

Bước 19: CGW gửi RAA cho PCRF.

- **Luồng tính cước PCRF (Terminate)**

Bước 20: (CCR Terminate): Khi khách hàng dừng sử dụng dịch vụ, GGSN gửi bản tin CCR (Terminate) tới CGW.

Bước 21: CGW nhận được yêu cầu và chuyển tới PCRF.

Bước 22: PCRF thực hiện giải mã bản tin, gọi IMDB Aerospike để kiểm tra thông tin thuê bao và gói cước.

Bước 23: PCRF thực hiện chọn phiên, tài khoản từ IMDB Aerospike.

Bước 24: PCRF thực hiện rating và sinh CDR event.

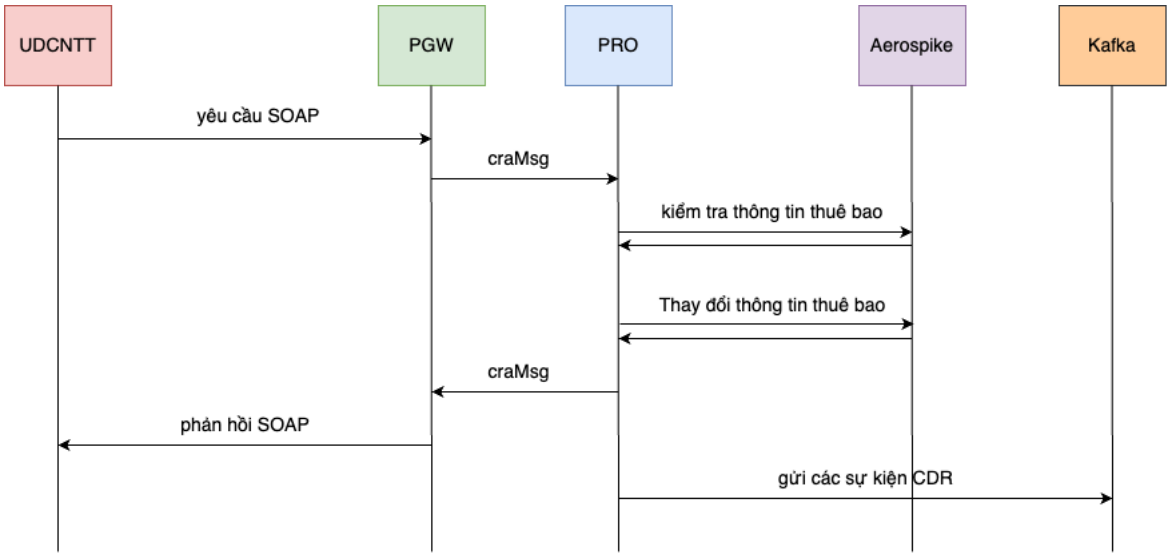
Bước 25: PCRF thực hiện trừ tài khoản, xoá phiên trên IMDB Aerospike.

Bước 26: PCRF trả bản tin CCA terminate cho CGW.

Bước 27: CGW trả bản tin CCA terminate cho GGSN thông báo ngắt phiên sử dụng.

Bước 28: PCRF gửi CDR Event tới CDRGen (Thuộc phân hệ CDR), CDRGen được sử dụng để đưa đến CBA phân tích hành vi khách hàng từ đó đưa ra các gói cước phù hợp.

2.3.5. Luồng tính cước dịch vụ Webservice



Hình 2.6. Luồng tính cước dịch vụ Webservice

Luồng xử lý tính cước dịch vụ Webservice được sử dụng trong các nghiệp vụ từ hệ thống ứng dụng CNTT như MyViettel [3] và được mô tả chi tiết trong hình 2.6. Các bước thực hiện tính cước được mô tả như bên dưới:

Bước 1: Hệ thống ứng dụng CNTT gửi sang OCS qua giao thức SOAP (yêu cầu SOAP) tới CGW rồi tới PRO.

Bước 2: PRO nhận được yêu cầu từ khách hàng thông qua các điểm đăng kí dịch vụ và thực hiện lấy thông tin về thuê bao, dịch vụ.

Tùy theo dịch vụ mà thuê bao muốn sử dụng hay đăng ký, thì trước tiên hệ thống sẽ gửi yêu cầu lên IMDB Aerospike để kiểm tra xem thuê bao có đang ở trạng thái hoạt động hay không? Nếu có thì thực hiện thêm hoặc update thông tin của thuê bao theo dịch vụ đã đăng ký.

Bước 3: Sau khi nhận được thông tin trả về từ IMDB Aerospike, tùy theo dịch vụ thuê bao đăng ký thì hệ thống sẽ gửi yêu cầu cộng tiền hay trừ tiền lên IMDB Aerospike để thực thi tác vụ đó.

Bước 4: PRO trả kết quả về cho thuê bao. Thông tin bản tin phản hồi SOAP trả về dựa trên dịch vụ mà thuê bao đăng ký, chẳng hạn: thuê bao muốn truy vấn thông tin tài khoản thì bản tin phản hồi SOAP sẽ bao gồm mã lỗi và danh sách các tài khoản cùng với lượng tiền ứng với mỗi tài khoản đó.

Bước 5: Đồng thời PRO gửi CDR Event đến CDRGen thực hiện sinh CDR File để lưu trữ thông tin dịch vụ mà khách hàng đã đăng ký.

2.4. Hạn chế của hệ thống OCS truyền thống

Hệ thống OCS tích hợp triển khai theo kiến trúc Monolith mang lại những ưu điểm như đơn giản để phát triển và triển khai cũng như dễ dàng quản lý. Chi tiết các ưu điểm của mô hình kiến trúc Monolith được thể hiện dưới đây:

Đơn giản: Mô hình monolith tổ chức ứng dụng thành một đơn vị duy nhất, giúp dễ dàng triển khai và vận hành. Việc phát triển và quản lý một ứng dụng monolith thường đơn giản hơn so với việc phát triển và quản lý các dịch vụ độc lập trong mô hình Microservices.

Hiệu suất: Vì ứng dụng monolith chạy trên một tiến trình duy nhất và chia sẻ cùng một bộ nhớ, nên có thể có hiệu suất cao hơn trong việc truy cập và xử lý dữ

liệu so với mô hình Microservices, đặc biệt là trong các ứng dụng có yêu cầu xử lý dữ liệu lớn.

Tính nhất quán: Vì toàn bộ mã nguồn và dữ liệu nằm trong một ứng dụng duy nhất, mô hình monolith đảm bảo tính nhất quán về dữ liệu và xử lý logic. Điều này có thể giảm bớt các vấn đề liên quan đến tính nhất quán và đồng bộ hóa dữ liệu giữa các dịch vụ độc lập trong mô hình Microservices.

Khả năng debug và kiểm tra: Trong mô hình monolith, việc debug và kiểm tra ứng dụng thường dễ dàng hơn do mã nguồn và dữ liệu nằm trong cùng một ứng dụng. Điều này giúp giảm thời gian và công sức trong việc tìm và sửa lỗi.

Tuy nhiên, những ưu điểm mà mô hình kiến trúc này mang lại cũng không thể bù đắp được cho những nhược điểm khi triển khai, Điều này đặt ra những thử thách rất lớn trong thời đại công nghệ phần mềm phát triển rất nhanh hiện nay [8].

Một số nhược điểm đáng kể của mô hình Monolith bao gồm:

Khả năng mở rộng hạn chế: Mô hình Monolith truyền thống thường xây dựng một ứng dụng lớn, không phân tách thành các dịch vụ nhỏ độc lập. Điều này làm giảm khả năng mở rộng của hệ thống, vì khi cần tăng cường khả năng xử lý, bạn phải tăng cường toàn bộ ứng dụng thay vì chỉ tập trung vào các phần cần thiết.

Độ linh hoạt thấp: Với mô hình Monolith, việc thay đổi hay cập nhật một phần của ứng dụng có thể phức tạp và rủi ro cao. Bạn cần triển khai toàn bộ ứng dụng lại để áp dụng các thay đổi, điều này làm giảm khả năng đáp ứng nhanh chóng với các yêu cầu và tình huống mới.

Khả năng phân phối tài nguyên không hiệu quả: Trong mô hình Monolith, tất cả các thành phần và chức năng của ứng dụng chia sẻ cùng một tài nguyên và không thể tận dụng tối đa khả năng phân phối tài nguyên của một hệ thống phân tán. Điều này có thể dẫn đến việc sử dụng không hiệu quả tài nguyên và giới hạn khả năng mở rộng.

Độ tin cậy và khả năng phục hồi kém: Một lợi ích của kiến trúc phân tán là khả năng chịu lỗi và khả năng phục hồi cao. Tuy nhiên, trong mô hình Monolith truyền thống, một lỗi trong một phần của ứng dụng có thể lan rộng và gây ra sự gián đoạn cho toàn bộ hệ thống. Đồng thời, khả năng khôi phục sau lỗi cũng có thể bị hạn chế.

2.5. Kết luận chương 2

Trong chương này đã nêu được tổng quan về hệ thống OCS theo mô hình Monolith bao gồm kiến trúc hệ thống, các thành phần trong hệ thống và luồng xử lý tính cước trực tuyến của các dịch vụ viễn thông đang triển khai tại Viettel hiện tại.

Tuy nhiên, với các lợi ích của mô hình Microservices đã đề cập trong chương 1 thì việc tích hợp triển khai hệ thống OCS theo kiến trúc Microservices thay thế cho mô hình Monolith truyền thống được xem là giải pháp triển khai phù hợp với xu hướng công nghệ tương lai và các đề xuất cũng như giải pháp tích hợp hệ thống OCS với mô hình Microservices sẽ được đề cập trong nội dung của chương 3.

CHƯƠNG 3: ÁP DỤNG MÔ HÌNH MICROSERVICES VÀO HỆ THỐNG TÍNH CƯỚC TẠI VIETTEL

3.1. Giới thiệu chung

Hệ thống tính cước trực tuyến OCS đóng vai trò đặc biệt quan trọng, góp phần nâng cao trải nghiệm của khách hàng và mang lại nguồn thu lớn cho các doanh nghiệp viễn thông, công nghệ thông tin. Bên cạnh đó, hệ thống OCS là một hệ thống bao gồm rất nhiều các luồng dịch vụ khác nhau và mỗi luồng dịch vụ lại có quá trình tính cước khác nhau nên nếu vẫn duy trì vận hành theo kiến trúc Monolith sẽ dẫn đến việc vận hành trở nên phức tạp. Mỗi khi lỗi xảy ra ảnh hưởng đến dịch vụ, kiến trúc một khối chung duy nhất này cũng dễ dẫn đến ảnh hưởng toàn bộ dịch vụ và rất khó để tìm nguyên nhân và xử lý lỗi. Vì vậy, kiến trúc hướng phân tách từng dịch vụ độc lập như Microservices sẽ trở nên hiệu quả hơn và nếu áp dụng được những ưu điểm của mô hình tích hợp vào hệ thống OCS sẽ mang lại những lợi ích vô cùng lớn [7]. Các lợi ích khi triển khai hệ thống OCS theo mô hình Microservices và quá trình triển khai hệ thống theo mô hình này được đề cập bên dưới.

3.1.1. Một số lợi ích khi triển khai hệ thống OCS theo mô hình Microservices

Những lợi ích khi triển khai hệ thống OCS theo mô hình Microservices có thể nhìn thấy dễ dàng nhất được mô tả như sau:

Phân tách chức năng: cho phép phân tách các chức năng của hệ thống OCS thành các dịch vụ nhỏ, độc lập và có khả năng mở rộng. Thay vì một hệ thống monolithic lớn, mỗi dịch vụ có thể chịu trách nhiệm về một phần cụ thể của quy trình tính cước, chẳng hạn như quản lý tài khoản khách hàng, tính toán cước phí, xác minh quyền truy cập và ghi nhận sự kiện. Điều này giúp tăng tính linh hoạt và khả năng mở rộng của hệ thống.

Độc lập triển khai và cập nhật: Với mô hình Microservices, mỗi dịch vụ có thể được triển khai và cập nhật độc lập. Điều này giúp giảm rủi ro và tác động của việc triển khai và cập nhật lên toàn bộ hệ thống. Nếu một dịch vụ cần được nâng cấp hoặc sửa lỗi, chỉ cần triển khai và cập nhật dịch vụ đó mà không ảnh hưởng đến các dịch

vụ khác. Điều này giúp giảm thời gian và công sức trong việc triển khai và cải thiện khả năng duy trì hệ thống.

Tích hợp linh hoạt: tích hợp linh hoạt với các hệ thống khác trong mạng viễn thông. Ví dụ, có thể có các dịch vụ riêng biệt để tích hợp với hệ thống quản lý thuê bao (BSS) để lấy thông tin tài khoản khách hàng, hoặc tích hợp với hệ thống thanh toán để xử lý thanh toán cước phí. Điều này giúp đảm bảo tính toàn vẹn và chính xác trong việc tính toán cước phí và quản lý thông tin khách hàng.

Tăng khả năng mở rộng: tăng khả năng mở rộng của hệ thống OCS. Với việc phân tách chức năng thành các dịch vụ độc lập, có thể dễ dàng mở rộng các dịch vụ đó để đáp ứng tải lớn hơn hoặc tăng cường khả năng xử lý. Nếu một dịch vụ gặp tải cao, có thể triển khai thêm bản sao của dịch vụ đó để chia sẻ công việc và tăng khả năng xử lý.

Tăng tính sẵn sàng và khả năng phục hồi: cung cấp khả năng sẵn sàng và khả năng phục hồi cao hơn. Nếu một dịch vụ gặp sự cố, các dịch vụ khác vẫn có thể hoạt động bình thường. Điều này giúp giảm thiểu sự gián đoạn và tăng tính sẵn sàng của hệ thống. Ngoài ra, có thể triển khai các giải pháp sao lưu và khôi phục để đảm bảo khả năng phục hồi sau sự cố.

3.1.2. Quá trình triển khai mô hình Microservices trong hệ thống OCS

Hệ thống OCS là một hệ thống lớn tích hợp xử lý tính cước rất nhiều dịch vụ viễn thông và công nghệ thông tin. Vì vậy, khi triển khai mô hình Microservices trong hệ thống OCS sẽ đối mặt với một số khó khăn như sau:

Phân cắt dịch vụ: Hệ thống OCS thường tích hợp nhiều dịch vụ và chức năng quan trọng. Phân cắt hệ thống thành các microservices độc lập và duy trì tính toàn vẹn của dữ liệu và tính nhất quán có thể là một thách thức.

Quản lý giao tiếp và đồng bộ: Các microservices trong hệ thống cần phải giao tiếp và đồng bộ dữ liệu với nhau. Điều này đòi hỏi sự quản lý cẩn thận để đảm bảo tính nhất quán và hiệu suất.

Quản lý tài nguyên và mở rộng: Hệ thống OCS thường xử lý khối lượng lớn giao dịch và yêu cầu tài nguyên mạnh mẽ. Triển khai microservices trong môi trường OCS đòi hỏi quản lý tài nguyên và mở rộng linh hoạt để đáp ứng nhu cầu tải cao và đảm bảo hiệu suất.

Quản lý lỗi và khắc phục sự cố: Với nhiều microservices hoạt động song song, quản lý lỗi và khắc phục sự cố có thể trở nên phức tạp. Đảm bảo khả năng theo dõi, ghi nhật ký và xử lý lỗi hiệu quả là quan trọng để duy trì sự ổn định và khả dụng của hệ thống.

Mô hình Microservices có rất nhiều những lợi ích lớn nhưng cũng tồn tại những khó khăn nhất định trong quá trình tích hợp. Để vượt qua và xử lý được các khó khăn này thì quá trình triển khai kiến trúc trong hệ thống OCS cần phải được đề ra và thực hiện theo quy tắc [7].

Các bước trong quá trình triển khai được đề cập cụ thể như bên dưới đây:

Phân tích chức năng: Tìm hiểu và phân tích chức năng của hệ thống OCS. Xác định các phần tử chức năng chính và xác định những phần tử này có thể được phân tách thành các dịch vụ độc lập (thoại, sms, data).

Thiết kế kiến trúc: Xác định các dịch vụ con và xác định cách chúng giao tiếp với nhau và với các hệ thống khác trong mạng viễn thông. Xác định giao diện API cho mỗi dịch vụ để cho phép việc giao tiếp và tích hợp.

Triển khai dịch vụ: Triển khai từng dịch vụ nhỏ thành một môi trường chạy độc lập. Có thể sử dụng công nghệ như Docker hoặc Kubernetes để triển khai và quản lý các dịch vụ này. Mỗi dịch vụ được triển khai trên một máy chủ hoặc một bộ vi xử lý độc lập để đảm bảo tính tách biệt và mở rộng.

Giao tiếp và tích hợp: Xác định cách các dịch vụ sẽ giao tiếp với nhau. Có thể sử dụng giao thức HTTP/REST hoặc giao thức thông điệp để gửi và nhận dữ liệu giữa các dịch vụ. Đồng thời, triển khai các cơ chế bảo mật và xác thực để đảm bảo tính an toàn và đáng tin cậy của hệ thống.

Quản lý và giám sát: Triển khai các công cụ và hệ thống giám sát để theo dõi hoạt động của các dịch vụ và tìm kiếm các vấn đề hoạt động. Sử dụng công cụ như Prometheus, Grafana hoặc ELK Stack để thu thập và hiển thị các thông số hoạt động và lỗi của các dịch vụ.

Quản lý phiên bản và cập nhật: Quản lý các phiên bản của từng dịch vụ để có thể triển khai các bản vá lỗi và cải tiến. Sử dụng các công cụ quản lý phiên bản như Git để theo dõi và quản lý mã nguồn của từng dịch vụ.

Kiểm thử và triển khai: Thực hiện kiểm thử đáng tin cậy cho từng dịch vụ và sau đó triển khai các dịch vụ đó vào môi trường hoạt động. Sử dụng các công cụ kiểm thử tự động và công cụ triển khai tự động để giảm thiểu sự cố và tối ưu hóa quy trình triển khai.

Theo dõi và tối ưu: Theo dõi hoạt động của các dịch vụ và tìm kiếm cơ hội để tối ưu hóa hiệu suất và khả năng mở rộng của hệ thống. Sử dụng các công cụ giám sát và thu thập phản hồi từ người dùng để cải thiện và tinh chỉnh các dịch vụ.

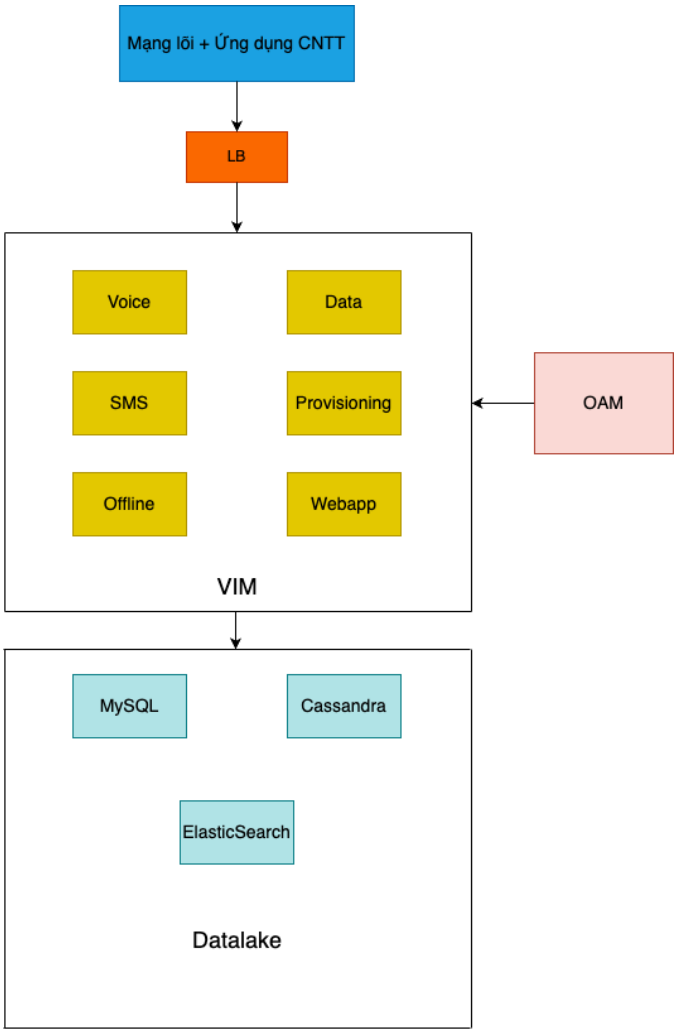
3.2. Đề xuất hệ thống OCS của Viettel theo mô hình Microservices

Hệ thống OCS tại Viettel đang triển khai theo mô hình Monolith, các dịch vụ viễn thông sẽ được gộp thành một khối duy nhất để tính cước. Với việc triển khai theo mô hình monolith truyền thống nên hệ thống OCS có kiến trúc đơn giản, không yêu cầu tính linh hoạt và mở rộng cao giúp đơn giản hoá quá trình triển khai và quản lý.

Tuy nhiên với nhu cầu sử dụng ngày càng cao và phức tạp của khách hàng, yêu cầu giao tiếp của hệ thống OCS với các hệ thống khác ngày càng lớn thì việc triển khai theo mô hình Monolith truyền thống sẽ gặp khó khăn trong việc mở rộng quy mô. Khi tải hệ thống lớn, việc mở rộng mô hình Monolith có thể đòi hỏi triển khai và quản lý nhiều bản sao của toàn bộ ứng dụng, không thể mở rộng chỉ một phần cụ thể. Ngoài ra, các yêu cầu sửa lỗi và nâng cấp liên tục có thể ảnh hưởng đến toàn bộ hệ thống OCS, làm tăng rủi ro và giới hạn tính linh hoạt trong việc triển khai các thay đổi và cải thiện.

Để đáp ứng được những yêu cầu thay đổi này thì việc định hướng phát triển hệ thống theo mô hình Microservices là đặc biệt quan trọng khi sẽ giải quyết được các nhược điểm của mô hình Monolith.

Hệ thống OCS là một trong những hệ thống đặc biệt quan trọng của Viettel. Vì vậy, các giải pháp được đưa ra để tích hợp mô hình Microservices vào hệ thống trở thành mục tiêu hàng đầu của doanh nghiệp.



Hình 3.1. Mô hình hệ thống OCS theo kiến trúc Microservices được triển khai tại Viettel

Hệ thống OCS triển khai theo mô hình Microservices tại Viettel được thể hiện chi tiết trong hình 3.1. Mô hình này chia các dịch vụ để tính cước thành từng khối dịch vụ nhỏ để xử lý, các quá trình xử lý tính cước, duy trì hoạt động bình thường của các tiến trình trong từng khối dịch vụ được quản lý chung bởi khối giám sát OAM. Ngoài ra, các cơ sở dữ liệu sử dụng để lưu trữ được tập trung trong khối Datalake [1].

Các thành phần trong mô hình OCS mới này được mô tả cụ thể như sau:

Khối dịch vụ Voice: xử lý các nghiệp vụ liên quan đến quá trình tính cước dịch vụ thoại. Khối này bao gồm các thành phần: SIU, SCP, CGW, OCP, ABM, TRIG.

Khối dịch vụ Data: xử lý các nghiệp vụ liên quan đến quá trình tính cước dịch vụ Data, PCRF. Khối này bao gồm các thành phần: CGW, OCP, ABM, TRIG.

Khối dịch vụ SMS: xử lý các nghiệp vụ liên quan đến quá trình tính cước dịch vụ SMS. Khối này bao gồm các thành phần xử lý như CGW, OCP, ABM, TRIG.

Khối dịch vụ Provisioning: xử lý các nghiệp vụ liên quan đến quá trình quản lý thông tin thuê bao, đăng ký, huỷ gói cước, cộng trừ tiền tài khoản sử dụng. Khối xử lý bao gồm các thành phần: SCP, PGW, PRO, ABM, TRIG.

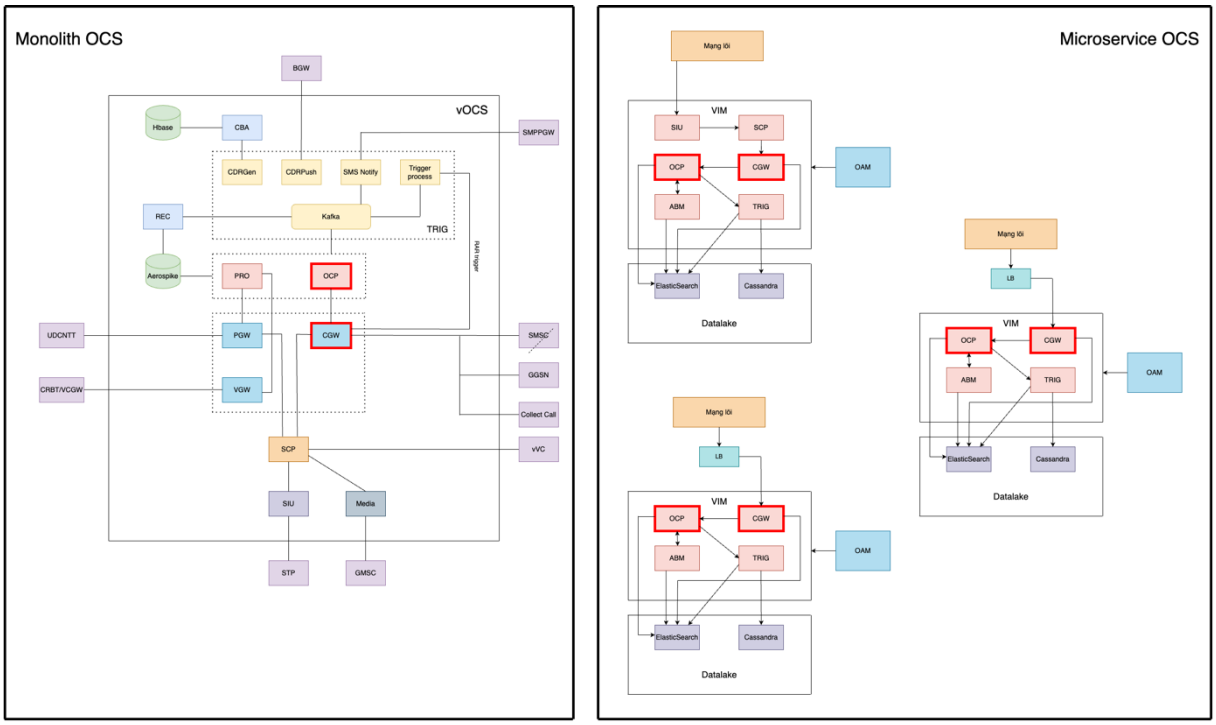
Khối dịch vụ Offline: xử lý các nghiệp vụ liên quan đến quá trình gia hạn gói cước, tài khoản sử dụng của thuê bao. Khối gia hạn bao gồm các thành phần như REC, ABM, TRIG.

Khối dịch vụ Webapp: xử lý các nghiệp vụ liên quan đến quá trình vận hành khai thác các dịch vụ trên Web như cấu hình chính sách kinh doanh (khai báo, thay đổi chính sách gói cước), tra cứu thông tin, lịch sử sử dụng dịch vụ của thuê bao. Khối webapp bao gồm các thành phần NGINX, WEBAPP, ABM.

Khối lưu trữ Datalake: xử lý các nghiệp vụ liên quan đến quá trình lưu trữ bản ghi cước dịch vụ (CDR), log tập trung các tiến trình xử lý của dịch vụ, chính sách kinh doanh của hệ thống.

Khối giám sát OAM: xử lý các nghiệp vụ liên quan đến quá trình giám sát tiến trình bao gồm CPU, RAM, tỷ lệ mã lỗi, tỷ lệ xử lý bản tin thành công, số lượng bản tin đầu vào.

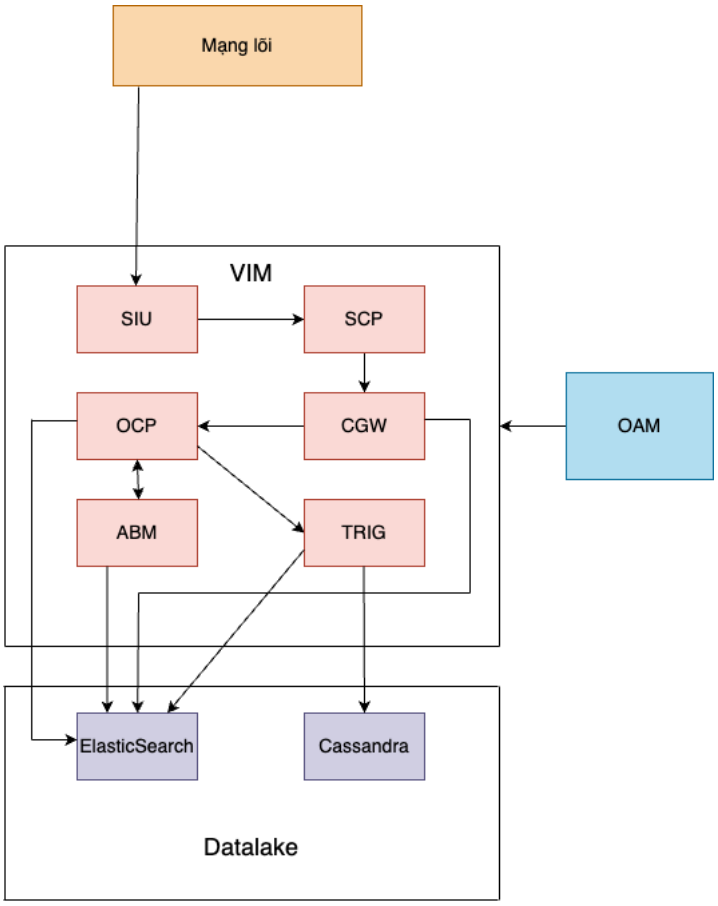
Các thành phần xử lý tính cước của mô hình mới vẫn giống với trong mô hình OCS truyền thống triển khai theo mô hình Monolith nhưng thay vì các thành phần xử lý CGW, OCP phải tham gia xử lý toàn bộ luồng tính cước trực tuyến dịch vụ viễn thông thì đã được phân tách ra sử dụng riêng cho từng dịch vụ, tức là các thành phần này khi nằm trong khối dịch vụ nào thì chỉ cần xử lý luồng tính cước dịch vụ đó. Điều này được thể hiện trong hình 3.2 khi so sánh chức năng xử lý tính cước của CGW và OCP trong hệ thống Monolith OCS và Microservice OCS.



Hình 3.2. Ánh xạ vai trò tính cước của CGW và OCP giữa hai hệ thống Monolith OCS và Microservice OCS

3.2.1. Khối dịch vụ Voice

Khối dịch vụ Voice trong hình 3.3 xử lý các nghiệp vụ liên quan đến quá trình tính cước dịch vụ thoại, sinh các bản ghi CDR thoại để lưu trữ vào trong cơ sở dữ liệu Cassandra phục vụ cho việc tra cứu các bản ghi CDR xử lý khiếu nại của khách hàng hay đối soát cước chênh lệch giữa hệ thống OCS với hệ thống khác. Ngoài ra, các log xử lý của từng thành phần trong khối sẽ được lưu trữ vào cơ sở dữ liệu ElasticSearch, sử dụng cho việc tra cứu quá trình xử lý trong trường hợp có lỗi xảy ra [1].



Hình 3.3. Mô hình khối dịch vụ Voice theo kiến trúc Microservices

Các thành phần trong khối dịch vụ Voice

SIU: cung cấp kết nối giữa báo hiệu SS7 và các máy chủ ứng dụng phân tán SCP, truyền tải các thông điệp và tín hiệu liên quan đến việc tính cước và quản lý cước phí trong thời gian thực, chuyển đổi giao thức, từ giao thức SS7 của mạng di động sang giao thức tương thích với OCS, đảm bảo tính tương thích và giao tiếp hiệu quả giữa hai hệ thống.

SCP: đóng vai trò quản lý trạng thái của cuộc gọi, xác định quá trình thiết lập, cập nhật và kết thúc cuộc gọi. Ngoài ra, SCP được sử dụng để lưu trữ các quy tắc xử lý và chức năng logic của dịch vụ thoại như chuyển tiếp cuộc gọi, cuộc gọi chờ, cung cấp các dịch vụ phức tạp hơn so với cuộc gọi thông thường, có thể kể đến như dịch vụ giá trị gia tăng VAS, các dịch vụ tổng đài 1800, 198. Khối SCP là thành phần trung gian giao tiếp giữa hệ thống SS7 và cổng tính cước thoại của hệ thống OCS.

CGW: nhận các bản tin từ SCP theo chuẩn giao thức diameter và chuyển đổi thành bản tin nội bộ CRA sau đó chuyển đến khối xử lý chính tính cước thoại online OCP.

Bên cạnh đó, đối với các cuộc gọi dịch vụ giá trị gia tăng hay cuộc gọi tổng đài 1800/198 thì CGW trực tiếp xử lý sinh ra các bản ghi CDR offline để sử dụng cho quá trình đối soát giữa hệ thống OCS và hệ thống khác cũng như xử lý các phản ánh của khách hàng trong quá trình sử dụng dịch vụ.

OCP: đóng vai trò là thành phần lõi của luồng tính cước, nhận bản tin CRA từ CGW gửi lên và thực hiện tính cước thoại theo thời gian thực, nhận các bản tin từ Gateway theo cơ chế roundrobin.

ABM: xử lý luồng truy vấn từ OCP tới database Aerospike lưu trữ thông tin thuê bao bao gồm số điện thoại, tài khoản, gói cước sử dụng, phiên truy cập thoại giúp OCP lấy được thông tin dữ liệu thuê bao để thực hiện tính cước thoại chính xác.

TRIG: xử lý nghiệp vụ liên quan đến luồng sinh bản ghi CDR sử dụng sau khi OCP thực hiện tính cước xong. Các bản ghi CDR này sẽ được dùng để giải quyết các khiếu nại của khách hàng trong quá trình sử dụng. Ngoài ra, CDR còn được dùng để đối soát giữa hệ thống OCS và hệ thống MSC, tránh thất thoát cước.

Các CDR sinh ra từ TRIG sẽ được thêm vào cơ sở dữ liệu Cassandra để lưu trữ và các log xử lý của tất cả các thành phần CGW, OCP, TRIG sẽ được lưu vào cơ sở dữ liệu ElasticSearch để quản lý log tập trung.

Tất cả quá trình hoạt động của luồng dịch vụ thoại đều được thành phần OAM giám sát. Khi các tiến trình thuộc các thành phần này ngừng hoạt động thì OAM sẽ thực hiện bật lại tiến trình. Bên cạnh đó, nó còn phát hiện khi quá trình xử lý gặp lỗi, ảnh hưởng đến tỷ lệ xử lý bản tin thành công thì sẽ gửi cảnh báo đến bộ phận giám sát để kịp thời phát hiện và tìm kiếm các phương pháp xử lý.

Luồng xử lý dịch vụ Voice

Bước 1: Khi thuê bao thực hiện cuộc gọi, MSC sẽ gửi thông tin đến OCS thông qua báo hiệu SS7 để điều khiển cuộc gọi. Các thông tin liên quan đến cuộc gọi như số gọi đi, số gọi đến hay vị trí thực hiện cuộc gọi được chuyển tiếp đến SCP để quản lý trạng thái cuộc gọi trước khi thực hiện quá trình tính cước.

Bước 2: SCP nhận các thông tin cuộc gọi và phân loại xử lý, các cuộc gọi thông thường sẽ được chuyển tiếp CGW và thực hiện quá trình tính cước online còn nếu là các cuộc gọi đến tổng đài miễn phí thì sẽ chuyển đến CGW và sinh ra CDR offline.

Bước 3: Với các cuộc gọi thông thường, CGW sẽ chuyển đổi bản tin gửi từ SCP thành bản tin nội bộ để xử lý trong hệ thống OCS. Sau khi quá trình chuyển đổi hoàn thành (bản tin craMsg được tạo ra), CGW sẽ gửi bản tin này lên OCP để tính cước thoại.

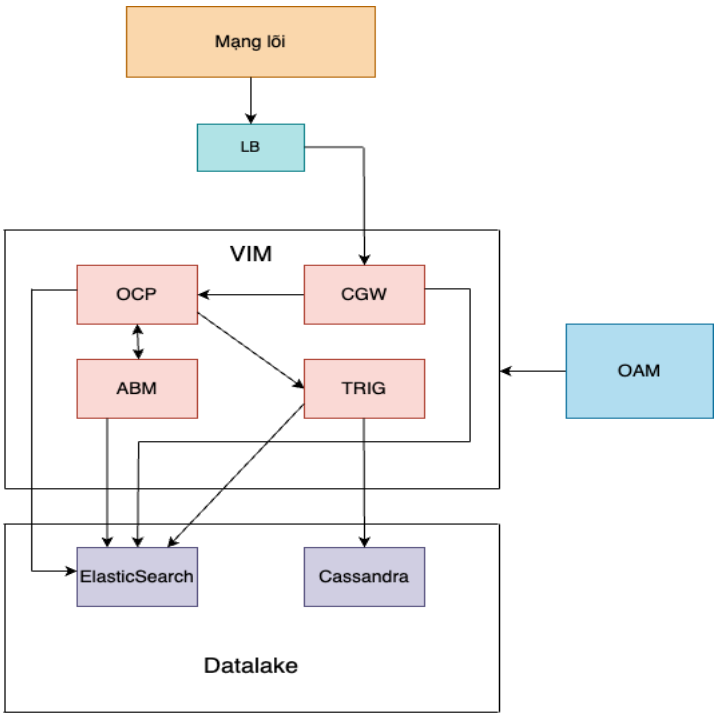
Bước 4: OCP sẽ đảm nhận chức năng chính trong quá trình tính cước thoại và giá cước sẽ được tính dựa trên các trường thông tin trong bản tin craMsg nhận được từ CGW.

Bước 5: OCP sẽ gửi truy vấn đến Aerospike để lấy thông tin thuê bao thông qua ABM. ABM chuyển tiếp yêu cầu của OCP đến cơ sở dữ liệu Aerospike để phản hồi thông tin thuê bao bao gồm trạng thái thuê bao, gói cước, tài khoản sử dụng, phiên thoại đang truy cập để OCP tính cước được chính xác.

Bước 6: Sau khi quá trình tính cước thực hiện xong, OCP sẽ gửi yêu cầu đến TRIG để sinh ra bản ghi CDR thoại phục vụ cho quá trình tra cứu và đối soát. Các CDR này sẽ được lưu trữ trong cơ sở dữ liệu Cassandra.

Bước 7: Các log xử lý của từng thành phần cũng được lưu trữ vào hệ thống quản lý log tập trung sử dụng cơ sở dữ liệu ElasticSearch để lưu trữ dài hạn phục vụ tìm kiếm và xử lý lỗi.

3.2.2. Khối dịch vụ Data



Hình 3.4. Mô hình khối dịch vụ Data theo kiến trúc Microservices

Khối dịch vụ Data mô tả trong hình 3.4 xử lý các nghiệp vụ liên quan đến quá trình tính cước dịch vụ Data và PCRF, sinh các bản ghi CDR data để lưu trữ vào trong cơ sở dữ liệu Cassandra phục vụ cho việc tra cứu các bản ghi CDR xử lý khiếu nại của khách hàng hay đối soát cước chênh lệch giữa hệ thống OCS với hệ thống khác tương tự như với chức năng của khối dịch vụ Voice. Các log xử lý của từng thành phần trong khối cũng sẽ được lưu trữ vào cơ sở dữ liệu ElasticSearch, sử dụng cho việc tra cứu quá trình xử lý trong trường hợp có lỗi xảy ra [1].

Các thành phần trong khối dịch vụ Data

CGW: nhận các bản tin từ GGSN theo chuẩn giao thức diameter và chuyển đổi thành bản tin nội bộ CRA sau đó chuyển đến khối xử lý chính tính cước data online OCP. Khác với luồng tính cước thoại, luồng tính cước data sẽ tính cước trên hai mặt phẳng giao diện gx và gy giống với hệ thống vOCS3.0 tạo thành hai luồng tính cước. Một luồng tính cước data, một luồng tính cước pcrf (điều khiển băng thông).

OCP: đóng vai trò là thành phần lõi của luồng tính cước, nhận bản tin CRA từ CGW gửi lên và thực hiện tính cước data và pcrf theo thời gian thực, nhận các bản tin từ Gateway theo cơ chế roundrobin.

ABM: xử lý luồng truy vấn từ OCP tới database Aerospike lưu trữ thông tin thuê bao bao gồm số điện thoại, tài khoản, gói cước sử dụng, phiên truy cập data (gy) và phiên truy cập pcrf (gx) giúp OCP lấy được thông tin dữ liệu thuê bao để thực hiện tính cước data chính xác.

TRIG: xử lý nghiệp vụ liên quan đến luồng sinh bản ghi CDR sử dụng sau khi OCP thực hiện tính cước xong. Các bản ghi CDR này sẽ được dùng để giải quyết các khiếu nại của khách hàng trong quá trình sử dụng. Ngoài ra, CDR còn được dùng để đối soát giữa hệ thống OCS và hệ thống GGSN, tránh thất thoát cước.

Các CDR sinh ra từ TRIG sẽ được thêm vào cơ sở dữ liệu Cassandra để lưu trữ và các log xử lý của tất cả các thành phần CGW, OCP, TRIG sẽ được lưu vào cơ sở dữ liệu ElasticSearch để quản lý log tập trung.

Tất cả quá trình hoạt động của luồng dịch vụ data đều được thành phần OAM giám sát. Cũng giống với vai trò trong luồng dịch vụ thoại, khi các tiến trình thuộc các thành phần này ngừng hoạt động thì OAM sẽ thực hiện bật lại tiến trình. Bên cạnh đó, nó còn

phát hiện khi quá trình xử lý gặp lỗi, ảnh hưởng đến tỷ lệ xử lý bản tin thành công thì sẽ gửi cảnh báo đến bộ phận giám sát để kịp thời phát hiện và tìm kiếm các phương pháp xử lý.

Luồng xử lý dịch vụ Data

Bước 1: Khi người dùng sử dụng data để truy cập mạng. các bản tin được GGSN gửi lên CGW theo hai luồng gx, gy. Mỗi luồng xử lý tính cước mang những thông tin khác nhau. Luồng gy mang những thông tin tính cước data còn luồng gx mang những thông tin để điều khiển băng thông.

Bước 2: CGW sau khi nhận được bản tin từ GGSN sẽ thực hiện chuyển đổi thành bản tin nội bộ craMsg để phục vụ xử lý tính cước trong hệ thống OCS.

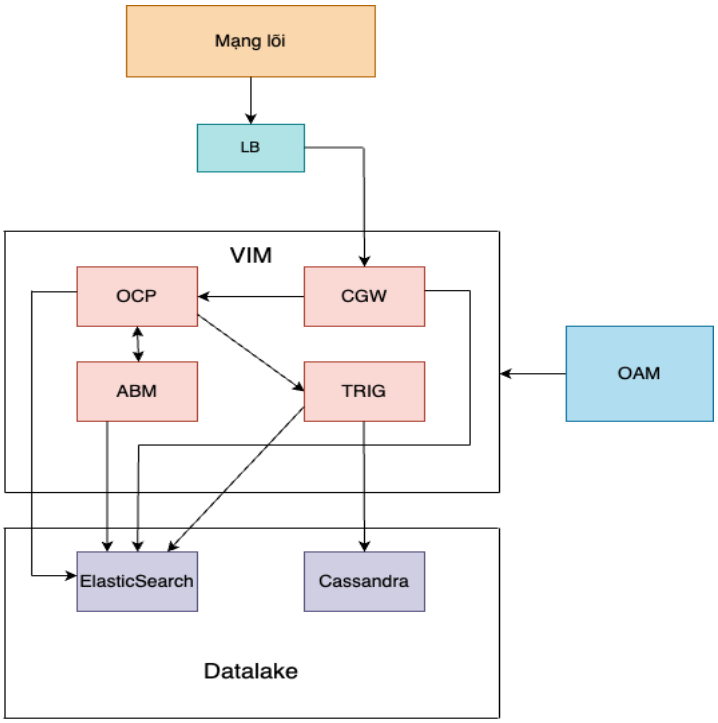
Bước 3: Các bản tin craMsg chuyển tiếp đến OCP được dùng để tính cước data và điều khiển băng thông (chặn truy cập, bóp băng thông).

Bước 4: OCP sẽ gửi truy vấn đến Aerospike để lấy thông tin thuê bao thông qua ABM. ABM chuyển tiếp yêu cầu của OCP đến cơ sở dữ liệu Aerospike để phản hồi thông tin thuê bao bao gồm trạng thái thuê bao, gói cước, tài khoản sử dụng, phiên data và pcrf đang truy cập để OCP tính cước được chính xác.

Bước 5: Sau khi quá trình tính cước thực hiện xong, OCP sẽ gửi yêu cầu đến TRIG để sinh ra bản ghi CDR data phục vụ cho quá trình tra cứu và đối soát. Các CDR này sẽ được lưu trữ trong cơ sở dữ liệu Cassandra.

Bước 6: Các log xử lý của từng thành phần cũng được lưu trữ vào hệ thống quản lý log tập trung sử dụng cơ sở dữ liệu ElasticSearch để lưu trữ dài hạn phục vụ tìm kiếm và xử lý lỗi.

3.2.3. Khối dịch vụ SMS



Hình 3.5. Mô hình khối dịch vụ SMS theo kiến trúc Microservices

Khối dịch vụ SMS xử lý các nghiệp vụ liên quan đến quá trình tính cước dịch vụ tin nhắn SMS, sinh các bản ghi CDR SMS để lưu trữ vào trong cơ sở dữ liệu Cassandra phục vụ cho việc tra cứu các bản ghi CDR xử lý khiếu nại của khách hàng hay đối soát cước chênh lệch giữa hệ thống OCS với hệ thống khác tương tự như với chức năng của khối dịch vụ Voice và Data. Các log xử lý của từng thành phần trong khối cũng sẽ được lưu trữ vào cơ sở dữ liệu ElasticSearch, sử dụng cho việc tra cứu quá trình xử lý trong trường hợp có lỗi xảy ra [1]. Chi tiết các thành phần trong khối và luồng xử lý tính cước được mô tả trong hình 3.5 và thể hiện bên dưới.

Các thành phần trong khối dịch vụ SMS

CGW: nhận các bản tin từ SMSC theo chuẩn giao thức SMPP và chuyển đổi thành bản tin nội bộ CRA sau đó chuyển đến khối xử lý chính tính cước SMS online OCP.

OCP: là thành phần xử lý chính của luồng tính cước SMS, nhận bản tin CRA từ CGW gửi lên và thực hiện tính cước SMS theo thời gian thực, nhận các bản tin từ Gateway theo cơ chế roundrobin.

ABM: xử lý luồng truy vấn từ OCP tới database Aerospike lưu trữ thông tin thuê bao bao gồm số điện thoại, tài khoản, gói cước sử dụng giúp OCP lấy được thông tin dữ liệu thuê bao để thực hiện tính cước data chính xác.

TRIG: xử lý nghiệp vụ liên quan đến luồng sinh bản ghi CDR sử dụng sau khi OCP thực hiện tính cước xong. Các bản ghi CDR này sẽ được dùng để giải quyết các khiếu nại của khách hàng trong quá trình sử dụng. Ngoài ra, CDR còn được dùng để đối soát giữa hệ thống OCS và hệ thống SMSC, tránh thất thoát cước.

Các CDR sinh ra từ TRIG sẽ được thêm vào cơ sở dữ liệu Cassandra để lưu trữ và các log xử lý của tất cả các thành phần CGW, OCP, TRIG sẽ được lưu vào cơ sở dữ liệu ElasticSearch để quản lý log tập trung.

Tất cả quá trình hoạt động của luồng dịch vụ data đều được thành phần OAM giám sát. Cũng giống với vai trò trong luồng dịch vụ thoại, khi các tiến trình thuộc các thành phần này ngừng hoạt động thì OAM sẽ thực hiện bật lại tiến trình. Bên cạnh đó, nó còn phát hiện khi quá trình xử lý gặp lỗi, ảnh hưởng đến tỷ lệ xử lý bản tin thành công thì sẽ gửi cảnh báo đến bộ phận giám sát để kịp thời phát hiện và tìm kiếm các phương pháp xử lý.

Luồng xử lý dịch vụ SMS

Bước 1: Khi người dùng gửi tin nhắn. các bản tin được SMSC gửi lên CGW, sau khi nhận được bản tin thì CGW sẽ thực hiện chuyển đổi thành bản tin nội bộ craMsg để phục vụ xử lý tính cước trong hệ thống OCS.

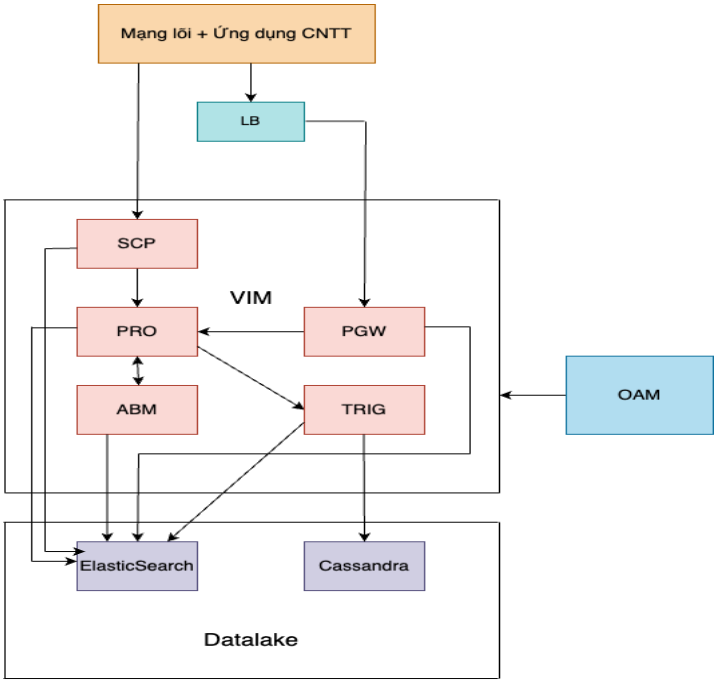
Bước 2: Các bản tin craMsg chuyển tiếp đến thành phần xử lý tinh cước chính OCP để tính cước SMS.

Bước 3: OCP sẽ gửi truy vấn đến Aerospike để lấy thông tin thuê bao thông qua ABM. ABM chuyển tiếp yêu cầu của OCP đến cơ sở dữ liệu Aerospike để phản hồi thông tin thuê bao bao gồm trạng thái thuê bao, gói cước, tài khoản sử dụng để OCP tính cước được chính xác.

Bước 4: Sau khi quá trình tính cước thực hiện xong, OCP sẽ gửi yêu cầu đến TRIG để sinh ra bản ghi CDR SMS phục vụ cho quá trình tra cứu và đối soát. Các CDR này sẽ được lưu trữ trong cơ sở dữ liệu Cassandra.

Bước 5: Các log xử lý của từng thành phần cũng được lưu trữ vào hệ thống quản lý log tập trung sử dụng cơ sở dữ liệu ElasticSearch để lưu trữ dài hạn phục vụ tìm kiếm và xử lý lỗi.

3.2.4. Khối dịch vụ Provisioning



Hình 3.6. Mô hình khối dịch vụ Provisioning theo kiến trúc Microservices

Khối dịch vụ Provisioning mô tả trong hình 3.6 xử lý các nghiệp vụ liên quan đến quá trình quản lý thông tin thuê bao bao gồm trạng thái thuê bao, các phiên truy cập đang tồn tại, các gói cước đang sử dụng và tài khoản tương ứng với từng gói. Cũng giống như các khối dịch vụ tính cước trực tuyến, khối Provisioning cũng sinh các bản ghi CDR để lưu trữ vào trong cơ sở dữ liệu Cassandra phục vụ cho việc tra cứu các bản ghi CDR xử lý khiếu nại của khách hàng hay đối soát cước chênh lệch giữa hệ thống OCS với hệ thống khác. Ngoài ra, các log xử lý của từng thành phần trong khối cũng sẽ được lưu trữ vào cơ sở dữ liệu ElasticSearch, sử dụng cho việc tra cứu quá trình xử lý trong trường hợp có lỗi xảy ra [1].

Các thành phần trong khối dịch vụ Provisioning

PGW: là cổng nhận các bản tin tính cước cho dịch vụ công nghệ thông tin từ LB, có khả năng bypass các bản tin theo yêu cầu trong các trường hợp ứng cứu thông tin.

Ngoài ra, việc đóng vai trò là cổng nhận bản tin thì nó cũng có tác dụng chuyển đổi các bản tin thành bản tin nội bộ CRA và gửi đến PRO xử lý tính cước.

SCP: đóng vai trò quản lý trạng thái của cuộc gọi, xác định quá trình thiết lập, cập nhật và kết thúc cuộc gọi IVR, USSD, chuyển tiếp các thông tin cuộc gọi lên PRO để tính cước. Ngoài ra, SCP cũng chuyển các thông tin về nghiệp vụ nạp thẻ thông thường không qua luồng webservice đến PRO để thay đổi thông tin thuê bao (hạn sử dụng thuê bao, lượng tiền tài khoản gốc).

PRO: là thành phần đóng vai trò chính trong quá trình quản lý thông tin thuê bao, nhận các bản tin CRA từ PGW và xử lý tính cước theo nghiệp vụ, thực hiện các chức năng đấu nối, đăng ký/hủy thuê bao, cộng/trừ tiền tài khoản, xử lý các dịch vụ USSD, IVR thông qua các bản tin gửi từ SCP. Đối với luồng nạp thẻ, khi nhận được thông tin về thẻ cào từ SCP, PRO sẽ kiểm tra thông tin của thẻ, nếu thẻ hợp lệ sẽ kiểm tra đến điều kiện nạp thẻ và gửi lại mã lỗi thành công hay thất bại về SCP.

ABM: xử lý luồng truy vấn từ PRO tới database Aerospike, cập nhật thông tin profile thuê bao từ PRO, cập nhật thông tin thuê bao bao gồm số điện thoại, tài khoản, gói cước sử dụng giúp PRO lấy được thông tin dữ liệu thuê bao để thực hiện tính cước chính xác.

TRIG: xử lý nghiệp vụ liên quan đến luồng sinh bản ghi CDR sử dụng sau khi PRO thực hiện tính cước xong. Các bản ghi CDR này sẽ được dùng để giải quyết các khiếu nại của khách hàng trong quá trình sử dụng. Ngoài ra, CDR còn được dùng để đối soát giữa hệ thống OCS và hệ thống ứng dụng công nghệ thông tin, tránh thất thoát cước.

Các CDR sinh ra từ TRIG sẽ được thêm vào cơ sở dữ liệu Cassandra để lưu trữ và các log xử lý của tất cả các thành phần PGW, PRO, TRIG sẽ được lưu vào cơ sở dữ liệu ElasticSearch để quản lý log tập trung.

Tất cả quá trình hoạt động của luồng dịch vụ provisioning đều được thành phần OAM giám sát. Cũng giống với vai trò trong luồng dịch vụ online, khi các tiến trình thuộc các thành phần này ngừng hoạt động thì OAM sẽ thực hiện bật lại tiến trình. Bên cạnh đó, nó còn phát hiện khi quá trình xử lý gặp lỗi, ảnh hưởng đến tỷ lệ xử lý bản tin thành công thì sẽ gửi cảnh báo đến bộ phận giám sát để kịp thời phát hiện và tìm kiếm các phương pháp xử lý.

Luồng xử lý dịch vụ Provisioning

Bước 1: Luồng dịch vụ Provisioning xử lý các nghiệp vụ liên quan đến nạp thẻ, IVR, USSD và Webservice.

Bước 2: Khi người dùng tra cứu thông tin thuê bao qua IVR, USSD, thành phần SCP sẽ nhận các bản tin và chuyển tiếp đến PRO để xử lý. Trong khi đó, với các tác vụ liên quan đến dịch vụ Webservice thì hệ thống ứng dụng CNTT sẽ gửi các yêu cầu lên OCS thông qua PGW.

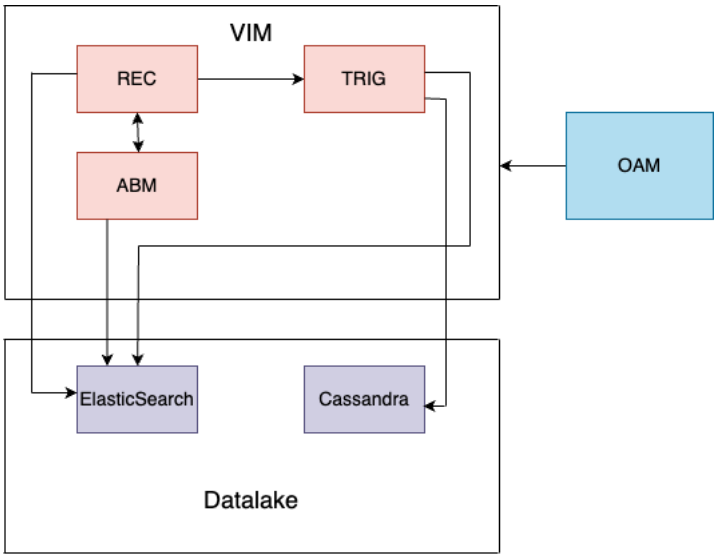
Bước 3: Tương tự như với chức năng của CGW, PGW cũng chuyển đổi các bản tin nhận được thành bản tin nội bộ craMicroservicesg rồi gửi đến PRO để xử lý tính cước trong hệ thống OCS.

Bước 4: PRO sẽ gửi truy vấn đến Aerospike để lấy thông tin thuê bao thông qua ABM. ABM chuyển tiếp yêu cầu của PRO đến cơ sở dữ liệu Aerospike để phản hồi thông tin thuê bao bao gồm trạng thái thuê bao, gói cước, tài khoản sử dụng để PRO tính cước được chính xác.

Bước 5: Sau khi quá trình tính cước thực hiện xong, PRO sẽ gửi yêu cầu đến TRIG để sinh ra bản ghi CDR Provisioning phục vụ cho quá trình tra cứu và đối soát. Các CDR này sẽ được lưu trữ trong cơ sở dữ liệu Cassandra.

Bước 6: Các log xử lý của từng thành phần cũng được lưu trữ vào hệ thống quản lý log tập trung sử dụng cơ sở dữ liệu ElasticSearch để lưu trữ dài hạn phục vụ tìm kiếm và xử lý lỗi.

3.2.5. Khối dịch vụ Offline



Hình 3.7. Mô hình khối dịch vụ Offline theo kiến trúc Microservice

Trong hình 3.7, khối dịch vụ Offline xử lý các nghiệp vụ liên quan đến quá trình gia hạn gói cước, tài khoản sử dụng của thuê bao theo các chu kỳ gia hạn đã được cấu hình trên web chính sách kinh doanh. Khối Offline sau khi hoàn thành quá trình gia hạn sẽ sinh ra các CDR để lưu trữ vào trong cơ sở dữ liệu Cassandra. Tương tự với các khối dịch vụ khác trong hệ thống OCS, các CDR này phục vụ cho việc tra cứu xử lý khiếu nại của khách hàng hay đối soát cước chênh lệch giữa hệ thống OCS với hệ thống khác. Với mỗi quá trình xử lý của từng thành phần trong khối sẽ tạo ra log và được lưu trữ vào cơ sở dữ liệu ElasticSearch, sử dụng cho việc tra cứu quá trình xử lý trong trường hợp có lỗi xảy ra [1].

Các thành phần trong khối dịch vụ Offline

REC: chịu trách nhiệm gia hạn gói cước, tài khoản sử dụng của thuê bao theo chu kỳ gia hạn đã được cấu hình trước của gói cước và tài khoản sử dụng trong gói cước đó. Nó sẽ kiểm tra các điều kiện gia hạn của thuê bao xem có thỏa mãn hay không (kiểm tra trạng thái của thuê bao có hoạt động bình thường hay đang bị chặn chiều, kiểm tra tài khoản thuê bao còn đủ để gia hạn không). Khi thuê bao thỏa mãn tất cả các điều kiện sẽ được gia hạn để sử dụng tiếp.

ABM: xử lý luồng truy vấn từ REC tới database Aerospike, cập nhật thông tin profile thuê bao từ REC, cập nhật thông tin thuê bao bao gồm số điện thoại, tài khoản,

gói cước sử dụng giúp REC lấy được thông tin dữ liệu thuê bao để thực hiện tính cước offline chính xác.

TRIG: nhận các bản tin gia hạn từ TRIG để sinh ra các bản ghi CDR gia hạn và được lưu trữ vào Cassandra. Các CDR này cũng sẽ được sử dụng để kiểm tra các phản ánh khách hàng giống như với các CDR dịch vụ online. Bên cạnh đó, việc sinh ra các CDR gia hạn cũng giúp rà soát cước với các hệ thống khác của Viettel giúp đảm bảo doanh thu.

Tất cả quá trình hoạt động của luồng dịch vụ offline đều được thành phần OAM giám sát. Khi các tiến trình thuộc các thành phần này ngừng hoạt động thì OAM sẽ thực hiện bật lại tiến trình. Bên cạnh đó, nó còn phát hiện khi quá trình xử lý gặp lỗi, ảnh hưởng đến tỷ lệ xử lý bản tin thành công thì sẽ gửi cảnh báo đến bộ phận giám sát để kịp thời phát hiện và tìm kiếm các phương pháp xử lý.

Luồng xử lý dịch vụ Offline

Bước 1: REC truy vấn đến ABM để lấy thông tin thuê bao của thuê bao từ Aerospike bao gồm các thông tin về trạng thái thuê bao, gói cước, tài khoản sử dụng để xác định quá trình gia hạn.

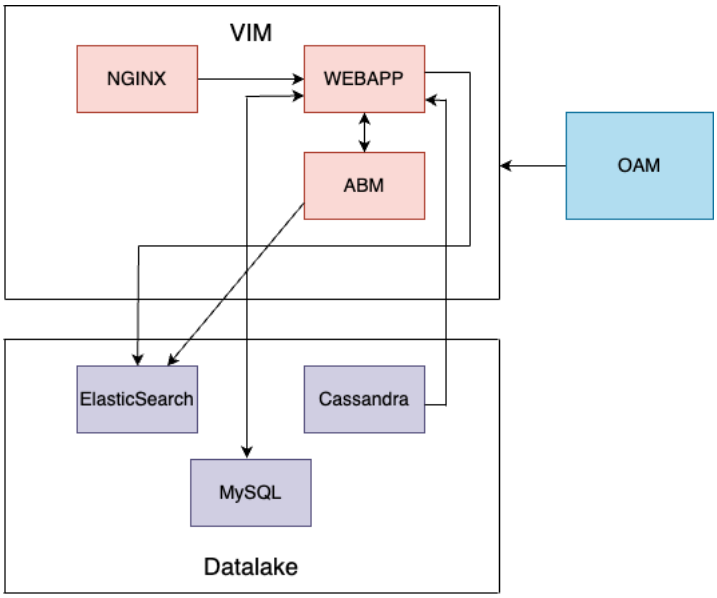
Bước 2: Trước khi thực hiện gia hạn, REC sẽ kiểm tra trạng thái thuê bao xem đang hoạt động bình thường hay bị chặn chiều cũng như thời điểm gia hạn của từng gói cước, từng tài khoản mà thuê bao đang sử dụng. Từ đó xác định thuê bao có đủ điều kiện để được gia hạn hay không.

Bước 3: Khi thuê bao đủ điều kiện gia hạn, REC sẽ thực hiện nghiệp vụ gia hạn cho gói cước, tài khoản để thuê bao sử dụng trong chu kỳ mới.

Bước 4: Sau khi quá trình gia hạn thực hiện xong, REC sẽ gửi yêu cầu đến TRIG để sinh ra bản ghi CDR gia hạn phục vụ cho quá trình tra cứu và đối soát. Các CDR này đặc biệt quan trọng để rà soát đảm bảo doanh thu và sẽ được lưu trữ trong cơ sở dữ liệu Cassandra.

Bước 5: Các log xử lý của từng thành phần cũng được lưu trữ vào hệ thống quản lý log tập trung sử dụng cơ sở dữ liệu ElasticSearch để lưu trữ dài hạn phục vụ tìm kiếm và xử lý lỗi.

3.2.6. Khối dịch vụ Webapp



Hình 3.8. Mô hình khối dịch vụ Webapp theo kiến trúc Microservice

Khối dịch vụ Webapp xử lý các nghiệp vụ liên quan đến quá trình vận hành khai thác các dịch vụ trên Web như cấu hình chính sách kinh doanh (khai báo, thay đổi chính sách gói cước), tra cứu thông tin, lịch sử sử dụng dịch vụ, tác động thay đổi thông tin của thuê bao [1]. Chi tiết các thành phần và luồng xử lý trong khối dịch vụ Webapp được thể hiện bên dưới.

Các thành phần trong khối dịch vụ Webapp

NGINX: xử lý các yêu cầu từ web đến các ứng dụng frontend hoặc backend thông qua các API thay vì như trên hệ thống OCS đang hoạt động sử dụng thành phần apache tomcat để quản lý server web, xử lý các yêu cầu từ web đến các ứng dụng thông qua các java servlet.

WEBAPP: website ứng dụng OCS có chức năng cấu hình chính sách kinh doanh liên quan đến gói cước, tài khoản, CDR. Ngoài ra, nó còn được sử dụng để tra cứu thông tin thuê bao, lịch sử sử dụng của thuê bao và tác động thay đổi thông tin của thuê bao theo yêu cầu.

ABM: tương tự như chức năng trong luồng dịch vụ online, quản lý thông tin thuê bao hay luồng gia hạn, thành phần này là khối xử lý trung gian giữa client và cơ sở dữ liệu aerospike. Khi nhận được yêu cầu truy vấn thông tin thuê bao từ webapp, ABM

chuyển tiếp đến Aerospike và gửi lại thông tin cho webapp hiển thị để người dùng sử dụng.

Tất cả quá trình hoạt động của luồng dịch vụ webapp đều được thành phần OAM giám sát. Khi các tiến trình thuộc các thành phần này ngừng hoạt động thì OAM sẽ thực hiện bật lại tiến trình. Bên cạnh đó, nó còn phát hiện khi quá trình xử lý gặp lỗi, ảnh hưởng đến tỷ lệ xử lý bản tin thành công thì sẽ gửi cảnh báo đến bộ phận giám sát để kịp thời phát hiện và tìm kiếm các phương pháp xử lý.

Luồng xử lý dịch vụ Webapp

Bước 1: Webapp trong hệ thống OCS mới tích hợp theo mô hình Microservice sẽ sử dụng Nginx để quản lý.

Bước 2: Khi người dùng sử dụng chức năng xem, tác động cấu hình chính sách gói cước, tài khoản, CDR thì Webapp sẽ gửi truy cập đến cơ sở dữ liệu MySQL để lấy thông tin hiển thị lên giao diện web.

Bước 3: Khi người dùng sử dụng chức năng tra cứu thông tin thuê bao, Webapp sẽ gửi yêu cầu đến Aerospike thông qua ABM tương tự như với các luồng dịch vụ khác để nhận lại thông tin hiển thị lên giao diện web cho người dùng tra cứu và kiểm tra.

Bước 4: Khi người dùng sử dụng chức năng tra cứu lịch sử sử dụng của thuê bao, Webapp sẽ truy vấn đến cơ sở dữ liệu Cassandra để lấy ra các CDR chức thông tin sử dụng cước dịch vụ online, cước provisioning hay cước gia hạn để tra cứu.

3.3. Đánh giá kết quả thử nghiệm hệ thống OCS theo mô hình Microservices

3.3.1. Những ưu điểm về chỉ tiêu kỹ thuật so với hệ thống cũ

Hệ thống OCS mới thử nghiệm đang có những ưu điểm đáng kể so với hệ thống chính đang triển khai như số lượng các server sử dụng, năng lực chịu tải của các node mạng (số lượng tải giao dịch TPS), thời gian phản hồi với mỗi giao dịch phải xử lý [2].

Các số liệu cụ thể được mô tả trong bảng 3.1 khi so sánh với hệ thống cũ đang triển khai.

Bảng 3.1. So sánh tham số giữa hệ thống OCS hiện tại và thử nghiệm

	Hệ thống OCS hiện tại	**Hệ thống OCS thử nghiệm**
Số lượng server	78	20
Năng lực chịu tải	800 TPS (OCP)	1200 TPS (OCP)
Thời gian phản hồi	20 ms (OCP-Voice)	3.22 ms (OCP-Voice)

Từ bảng 3.1, hiệu suất sử dụng và năng lực xử lý của hệ thống OCS thử nghiệm đang mang đến những kết quả đáng kể. Số lượng server trong kiến trúc mới giảm được hơn 1/4 so với mô hình cũ. Điều này giúp chi phí đầu tư, bảo dưỡng các server vật lý hàng năm giảm rất nhiều. Ngoài ra, việc số lượng server vật lý phải sử dụng ít hơn cũng đem lại hiệu quả về mặt kinh tế khi giảm được số lượng nhân sự trực vận hành tại các tổng trạm.

Năng lực chịu tải của các thành phần trong hệ thống cũng được tăng lên khi sử dụng hệ thống OCS tích hợp mô hình Microservices. Theo chỉ tiêu kỹ thuật và tham số đang cấu hình thực tế tại các node mạng OCP thì tỉ lệ chịu tải tối đa đang là 800 TPS, tức là với tất cả dịch vụ trực tuyến mà OCP tính cước thì tải tối đa có thể chịu là 800. Trong khi đó, với hệ thống OCS thử nghiệm tích hợp kiến trúc mới, năng lực chịu tải của OCP là 1200 TPS. Điểm khác biệt là số lượng tải giao dịch này là dành riêng cho từng dịch vụ, tức là với mỗi dịch vụ như thoại, data, sms thì năng lực chịu tải của OCP đều là 1200. Nếu so với số tải giao dịch của hệ thống OCS hiện tại thì lớn hơn rất nhiều, cho phép tăng năng lực xử lý, đáp ứng trải nghiệm dịch vụ của nhiều khách hàng hơn nữa.

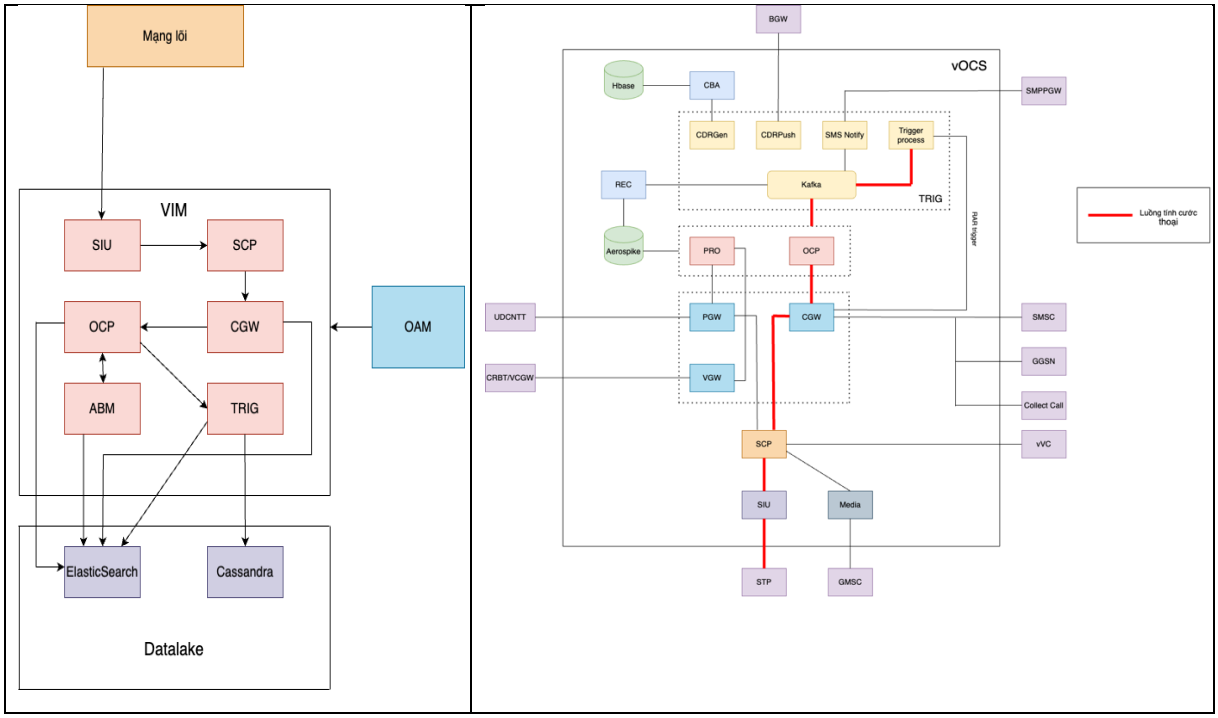
Một điểm đáng chú ý là thời gian phản hồi trong quá trình xử lý khi so sánh giữa hệ thống hiện tại và hệ thống thử nghiệm cũng rất khác biệt. Đối với hệ thống hiện tại, xét riêng trong trường hợp luồng dịch vụ thoại, thời gian phản hồi của OCP với mỗi tải giao dịch đang là 20 ms. Trong khi đó, thời gian phản hồi của OCP trong hệ thống OCS thử nghiệm là 3.22 ms, ít hơn rất nhiều lần. Với thời gian phản hồi ngắn hơn thì trong cùng một khoảng thời gian nhất định, OCP trong hệ thống mới theo kiến trúc Microservices sẽ xử lý được nhiều tải giao dịch hơn so với hệ thống hiện tại theo kiến trúc Monolith.

3.3.2. Kết quả đạt được trong quá trình thử nghiệm hệ thống

Ngoài những ưu điểm lớn về mặt chỉ tiêu kỹ thuật so với hệ thống cũ thì các luồng dịch vụ tính cước cũng được phân tách độc lập, cùng với đó là các luồng giám sát và lưu log tập trung được trực quan hoá để dễ dàng phát triển và vận hành khai thác hệ thống.

Hệ thống OCS mới tại Viettel tích hợp triển khai theo mô hình Microservices đang trong giai đoạn thử nghiệm và tiếp tục phát triển. Hiện tại, hệ thống mới đang cung cấp dịch vụ tính cước, điều khiển băng thông đáp ứng cho 02 triệu thuê bao cắt chuyển từ hệ thống cũ và phục vụ đánh giá các yêu cầu tính năng theo bộ chỉ tiêu kỹ thuật để làm căn cứ báo cáo đầu tư hệ thống có năng lực xử lý toàn bộ thuê bao trên hệ thống chính.

Các dịch vụ tính cước được phân tách ra thành các khối dịch vụ độc lập mang lại thuận lợi tạo ra luồng tính cước dịch vụ tường minh và thuận lợi cho công tác vận hành khai thác.



Hình 3.9. So sánh luồng dịch vụ thoại giữa hệ thống OCS Monolith (trái) và OCS Microservice (phải)

Trong hệ thống OCS triển khai theo mô hình Microservices, các phần tử như CGW, OCP, TRIG sẽ chỉ phải chịu trách nhiệm xử lý riêng phần tính cước dịch vụ cụ thể thay

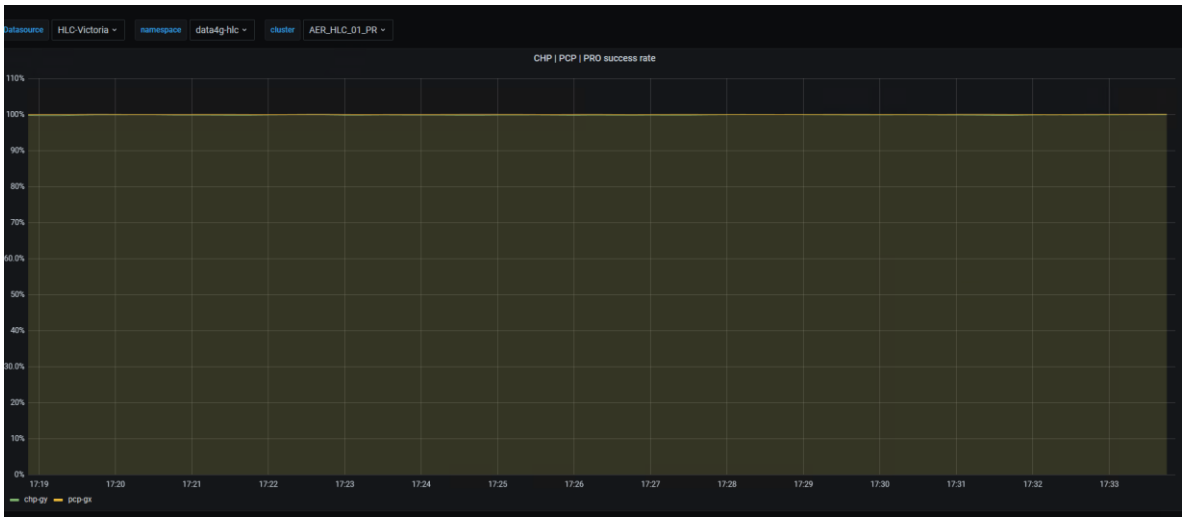
vì phải chịu trách nhiệm với tất cả phần tính cước dịch vụ trực tuyến như trong mô hình Monolith truyền thống. Điều này được thể hiện trong hình 3.9 bên trên, ví dụ đối với thành phần tính cước chính OCP thì sẽ phải tính cước trực tuyến cho tất cả các dịch vụ như thoại, data, sms trong khi đó OCP trong kiến trúc Microservices chỉ phải tính cước cho duy nhất dịch vụ thoại, góp phần giảm tải xử lý cho OCP. Từ đó giúp hạn chế lỗi đa dịch vụ hoặc lỗi toàn mạng. Ngoài ra cũng tạo thuận lợi khi xử lý lỗi, tìm nguyên nhân và biện pháp khắc phục nhanh hơn.

Các thông tin về phần giám sát từng khối dịch vụ cũng dễ dàng hơn cho các kỹ sư trong việc theo dõi và vận hành hệ thống. Các thành phần cần giám sát như TPS, Success Rate hay Result Code cũng được hiển thị cụ thể cho từng loại dịch vụ khác nhau để phát hiện lỗi kịp thời và đưa ra biện pháp xử lý phù hợp.

Hình 3.10. Các thành phần giám sát KPI cho dịch vụ Data

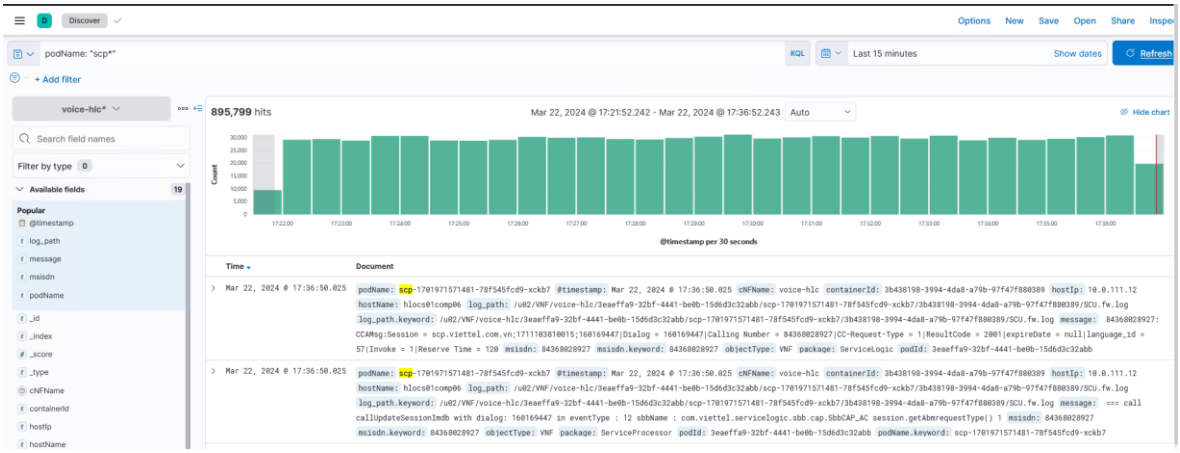
Với các biểu đồ giám sát của dịch vụ Data như trong hình 3.10, các kỹ sư vận hành có thể quan sát thấy những bất thường ngay lập tức khi TPS nhỏ hơn 0, tỉ lệ xử lý bản tin thành công Success Rate dưới mức 95% hoặc số bản tin bị timeout, drop tại CGW tăng. So với việc giám sát dịch vụ của hệ thống OCS hiện tại đang gộp chung với nhau thì hệ thống OCS mới thử nghiệm có thể giám sát cụ thể, chi tiết được từng tham số.

Các tiêu chí của từng thành phần KPI được chi tiết hoá theo thời gian để tạo điều kiện thuận lợi cho quá trình vận hành khai thác. Tham số xử lý bản tin thành công được giám sát chi tiết trong khoảng thời gian từ 17h20 đến 17h30 theo như hình 3.11.



Hình 3.11. Biểu đồ giám sát tỉ lệ xử lý bản tin thành công của OCP trong dịch vụ Data

Bên cạnh những giá trị về mặt giám sát KPI dịch vụ khi triển khai hệ thống OCS theo mô hình Microservices, các log xử lý của từng dịch vụ cũng được tách riêng để tìm kiếm và xử lý trong trường hợp lỗi xảy ra, phát hiện điểm bất thường như bản tin bị timeout không xử lý được hay xuất hiện nhiều exception của từng thành phần trong từng khối dịch vụ.



Hình 3.12. Các log xử lý của thành phần SCP trong luồng dịch vụ thoại

Trong hình 3.12, các log của module SCP xử lý nghiệp vụ thoại được tra cứu trong khoảng 15 phút gần nhất, các kỹ sư Viettel có thể tìm kiếm log xử lý nghiệp vụ hay các execption bất thường có thể xuất hiện như TimeOutException, DisconnectException. So với hệ thống lưu log tập trung hiện tại đang gộp chung log xử lý của tất cả dịch vụ

với nhau thì hệ thống lưu log của hệ thống OCS mới thử nghiệm có thể lưu log riêng của từng dịch vụ bao gồm tất cả các thành phần xử lý tính cước liên quan.

3.4. Kết luận chương 3

Hệ thống tính cước trực tuyến OCS là một trong những hệ thống đặc biệt quan trọng của mỗi doanh nghiệp viễn thông, công nghệ thông tin. Do vậy, việc triển khai thử nghiệm tích hợp hệ thống OCS với mô hình Microservices là xu hướng tất yếu trong lĩnh vực công nghệ phần mềm. Những ưu điểm mà mô hình Microservices mang lại như việc phân tách độc lập giữa các dịch vụ với nhau, tăng tính sẵn sàng trong việc cập nhật phiên bản mới hoặc xử lý lỗi đã giúp các dịch vụ viễn thông trong hệ thống OCS linh hoạt hơn, thuận lợi hơn cho công tác phát triển và vận hành khai thác tại Viettel.

KẾT LUẬN

Mô hình Microservices đã trở thành một xu hướng phổ biến trong việc phát triển các hệ thống phần mềm phức tạp. Bằng cách chia nhỏ ứng dụng thành các dịch vụ nhỏ, độc lập và có khả năng hoạt động độc lập, mô hình Microservices mang lại nhiều lợi ích quan trọng như tăng tính linh hoạt và khả năng mở rộng, cho phép sử dụng công nghệ và ngôn ngữ phù hợp nhất cho từng dịch vụ, tạo điều kiện cho việc phát triển phân tán và mở rộng ngang.

Trong khi đó, hệ thống tính cước trực tuyến OCS là một hệ thống đóng vai trò quan trọng trong việc tính toán cước phí dịch vụ viễn thông và quản lý thông tin thuê bao nên việc tận dụng những ưu điểm mà mô hình Microservices mang lại sẽ đem đến nhiều giá trị như hạn chế lỗi đa dịch vụ, dễ dàng khoanh vùng lỗi khi xảy ra và thuận lợi cho công cuộc vận hành khai thác tại Viettel.

Do thời gian nghiên cứu còn có hạn nên nhiều vấn đề trong đề án có thể chưa sâu sắc và đầy đủ, vì vậy tôi mong muốn nhận được sự thông cảm và đóng góp quý báu của các Thầy, Cô để bổ sung cho những nghiên cứu sau này.

Hướng phát triển tiếp theo của Đề án là hoàn thiện và phát triển hệ thống OCS tích hợp mô hình Microservices cho toàn bộ thuê bao trong mạng lưới của Viettel trong tương lai.

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1]. Trung tâm Kỹ thuật Toàn cầu, Tổng Công ty Mạng lưới Viettel, "*Tài liệu thiết kế chi tiết hệ thống vOCS4.0 PoC*", 2022.

[2]. Trung tâm nghiên cứu OCS, Tổng Công ty Công nghiệp Công nghệ cao, "Tài liệu quy hoạch định cỡ vOCS4.0 100M", 2022.

[3]. Trung tâm kỹ thuật toàn cầu, Tổng Công ty Mạng lưới Viettel, (2017), Hồ sơ thiết kế chi tiết hệ thống vOCS3.0.

[4]. Trung tâm nghiên cứu công nghệ mạng Viettel, Tập đoàn Viễn thông Quân đội, (2018), Nghiên cứu, xây dựng hệ thống tính cước online vOCS3.0.

[5]. Trung tâm nghiên cứu công nghệ mạng Viettel, Tập đoàn Viễn thông Quân đội, (2019), Tài Liệu Mô Tả Chi Tiết Thiết Kế Cấu Hình Gói Cước.

[6]. Nguyễn Thế Lộc, "*Triển khai ứng dụng theo mô hình Microservice*", Trường Đại học Mỏ - Địa chất, 2022.

Tiếng Anh.

[7]. Shahir Daya, Nguyen Van Duy, Kameswara Eati, ..., "Microservices from Theory to Practice", IBM International Technical Support Organization, August 2015.

[8]. Eivind Solberg, "The transition from monolithic architecture to microservice architecture", UNIVERSITY OF OSLO, Spring 2022.