**CÔNG TY CỔ PHẦN MISA JSC**

**---🙠**🕮**🙢---**



**NGÀY HỘI SẢN XUẤT**

**NGHIÊN CỨU GIẢI PHÁP ĐỒNG BỘ DỮ LIỆU REAL-TIME GIỮA SQL VÀ ELASTICSEARCH ĐÁP ỨNG TÌM KIẾM DỮ LIỆU LỚN**

**CHO CÁC SẢN PHẨM CỦA MISA**

BÁO CÁO NGHIÊN CỨU

Nhóm: Dream Chaser From CRM

**HÀ NỘI, 08-2022**

**CÔNG TY CỔ PHẦN MISA JSC**

**---🙠**🕮**🙢---**



**NGÀY HỘI SẢN XUẤT**

**NGHIÊN CỨU GIẢI PHÁP ĐỒNG BỘ DỮ LIỆU REAL-TIME GIỮA SQL VÀ ELASTICSEARCH ĐÁP ỨNG TÌM KIẾM DỮ LIỆU LỚN**

**CHO CÁC SẢN PHẨM CỦA MISA**

BÁO CÁO NGHIÊN CỨU

Nhóm: Dream Chaser From CRM

**Cán bộ hướng dẫn: Nguyễn Quang Hoàng**

**HÀ NỘI, 08-2022**

# **NHẬN XÉT CỦA BAN GIÁM KHẢO**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….………………………………………………………….**

**……………………………………….……………………………………………………………………………………………………………………………………………………………**

|  |
| --- |
| Hà Nội, ngày …. Tháng …. Năm 2022 |
| Ký tên |
|  |

**Tóm Tắt:** Với sự phát triển mạnh mẽ của công ty cổ phầm MISA JSC, lượng sản phầm của MISA ngày càng phát triển cả về số lượng cũng như chất lượng, cùng với đó là tập dữ liệu khách hàng khổng lồ từ các phần mềm. MISA luôn mong muốn đem lại cho người dung trải nghiệm dịch vụ tốt nhất nhằm đem lại hiệu quả về năng suất cũng như chất lượng công việc lên hàng đầu, luôn đề cao tinh thần phụ sự xã hội. Tuy nhiên, cùng với việc phát triển lớn mạnh của các phần mềm nhà MISA thì dữ liệu từ khách hàng là vô cùng lớn, việc dữ liệu lớn được lưu trữ trên cơ sở dữ liệu quan hệ cho ra một hiệu suất truy vấn không được tốt, MISA cũng đã có những sự cô liên quan đến việc tím kiếm dữ liệu lớn. Các công cụ tìm kiếm ra đời nhằm giải quyết tình trạng này như: Apache Solr, MeiliSearch, Sphinx, Manticore và Elasticsearch là một trong những công cụ tìm kiếm giúp tôi ưu việc tìm kiếm nhanh dữ liệu lớn. Tuy nhiên, việc phát sinh vấn đề đồng bộ dữ liệu giữa cơ sở dữ liệu quan hệ sang cơ sở dữ liệu tìm kiếm là vấn đề lớn nhất, Apache Kafka là một nền tảng theo kiến trúc phân tán cho phép lưu trữ sự kiện và xử lý dữ liệu luồng mã nguồn mở được phát triển bởi Apache Software, với khả năng chịu lỗi cao và là hệ thống nhắn tin nhanh, kafka đang dần được thay thế cho hệ thống nhắn tin truyền thống. Nó được sử dụng cho các hệ thống nhắn tin thông thường trong các ngữ cảnh khác nhau. Trong nghiên cứu này, việc sử dụng Elasticsearch giải quyết bài toán tìm kiếm dữ liệu lớn và quan trọng hơn cả là tìm ra phương pháp đồng bộ dữ liệu real-time giữa cơ sở dữ liệu quan hệ và cơ sở dữ liệu tìm kiếm sẽ được chúng tôi ứng dụng thực tế tại dự án CRM.

**MỤC LỤC**

[NHẬN XÉT CỦA BAN GIÁM KHẢO 3](#_Toc112272886)

[MỤC LỤC 5](#_Toc112272887)

[DANH MỤC HÌNH ẢNH 6](#_Toc112272888)

[DANH MỤC BẢNG BIỂU 7](#_Toc112272889)

[CHƯƠNG 1: GIỚI THIỆU CHUNG 8](#_Toc112272890)

[1.1 Lý do chọn đề tài 8](#_Toc112272891)

[1.2 Giới thiệu 8](#_Toc112272892)

[1.1.1 Bài toán tìm kiếm nhanh với dữ liệu lớn 8](#_Toc112272893)

[1.1.2 Tìm kiếm nhanh trên các công cụ tìm kiếm 8](#_Toc112272894)

[1.1.3 Cơ chế đồng bộ dữ liệu 8](#_Toc112272895)

[1.3 Tiềm năng ứng dụng của bài toán tìm kiếm dữ liệu lớn và đồng bộ dữ liệu real-time tại công ty cổ phần MISA JSC 8](#_Toc112272896)

[1.4 Mục tiêu của nghiên cứu 8](#_Toc112272897)

[1.5 Cấu trúc bài nghiên cứu 8](#_Toc112272898)

[CHƯƠNG 2. ĐỒNG BỘ DỮ LIỆU REAL-TIME GIỮA CSDL GỐC VÀ CSDL TÌM KIẾM 9](#_Toc112272899)

[2.1 Cơ chế đồng bộ CDC 9](#_Toc112272900)

[2.2 Đồng bộ theo cơ chế Binlog Mysql 9](#_Toc112272901)

[2.3 Message Queue 10](#_Toc112272902)

[2.4. Các nền tảng và website hiện nay 13](#_Toc112272903)

[2.5 Kết luận 13](#_Toc112272904)

[CHƯƠNG 3. KẾT QUẢ THỰC NGHIỆM 14](#_Toc112272905)

[CHƯƠNG 4. KẾT LUẬN 15](#_Toc112272906)

[TÀI LIỆU THAM KHẢO 16](#_Toc112272907)

# **DANH MỤC HÌNH ẢNH**

**DANH MỤC BẢNG BIỂU**

# **CHƯƠNG 1: GIỚI THIỆU CHUNG**

## Lý do chọn đề tài

Hiện tại, với gần 30 sản phẩm khách nhau, cùng với đó là một tập dữ liệu khách hàng lớn, việc dữ liệu lớn được lưu trữ trên cơ sở dữ liệu quan hệ cho ra một hiệu suất truy vấn không được tốt, MISA cũng đã có những sự cô liên quan đến việc tím kiếm dữ liệu lớn. Tìm kiếm nhanh với dữ liệu lớn là một trong những vấn đề nhức nhối tại hầu hết các công ty trong và ngoài nước, mỗi đơn vị, tổ chức đều có cách giải quyết tương đồng hoặc khác nhau, tuy nhiên theo khảo sát tại các dự án tại công ty cổ phần MISA JSC, có tới hơn một nửa trong số đó đang gặp vấn đề về bài toán tìm kiếm dữ liệu lớn. Trong số đố, một vài dự án đã có cách giải quyết, tuy nhiên những giải pháp đồng bộ dữ liệu giữa cơ sở dữ liệu quan hệ với cơ sở dữ liệu tìm kiếm đều gặp nhiều vấn đề về hiệu năng đồng bộ và hơn thế nữa là ảnh hưởng đến tính đúng đắn của dữ liệu, đây là thách thức rõ nhất, vấn đề này sẽ được tìm hiểu và giải quyết trong bài nghiên cứu này.

## Giới thiệu

Cùng với việc phát triển lớn mạnh của các phần mềm nhà MISA thì dữ liệu từ khách hàng là vô cùng lớn, việc dữ liệu lớn được lưu trữ trên cơ sở dữ liệu quan hệ cho ra một hiệu suất truy vấn không được tốt, MISA cũng đã có những sự cố liên quan đến việc tím kiếm dữ liệu lớn. Các công cụ tìm kiếm ra đời nhằm giải quyết tình trạng này như: Apache Solr, MeiliSearch, Sphinx, Manticore và Elasticsearch là một trong những công cụ tìm kiếm giúp tôi ưu việc tìm kiếm nhanh dữ liệu lớn.

Theo một khảo sát nhanh tại MISA hiện có hơn 5/30 phiếu khảo sát tại các dự án là sử dụng Elasticsearch, Elasticsearch sẽ cung cấp cho người dùng log đầy đủ để có thể giúp chúng ta có thể tìm và phân tích ra được trend dựa theo partern ví dụ như Show data với value cụ thể, Search data theo vị trí địa lý, Tổng hợp thông tin theo ngày. Vì vậy Elasticsearch được xem như là đã phổ biến tại MISA và trong nghiên cứu lần này việc giải quyết bài toán tìm kiếm nhanh bằng công cụ tìm kiếm được coi như đã được giải quyết tại các dự án.

Tuy nhiên, việc phát sinh vấn đề đồng bộ dữ liệu giữa cơ sở dữ liệu quan hệ sang cơ sở dữ liệu tìm kiếm là vấn đề lớn nhất, qua tìm hiểu sơ bộ cho thấy có rất nhiều phương pháp thủ công lẫn tự động nhằm giải quyết tình trạng này như: lưu dữ liệu đồng thời tại hai cơ sở dữ liệu, sử dụng một woker đồng bộ. Tuy nhiên những phương pháp trên không đem lại hiệu quả cao và tiềm ẩn nhiều rủi ro, trong nghiên cứu này chúng tôi sẽ đưa ra một bức tranh tổng quan về những phương pháp đồng bộ dữ liệu thời gian thực hiện có và đưa ra lời giải tốt nhất.

### **Bài toán tìm kiếm nhanh với dữ liệu lớn**

Tìm kiếm là một trong những chức năng gần như bắt buộc có đối với các sản phẩm nói chung và sản phẩm của MISA nói riêng, tìm kiếm trên cơ sở dữ liệu quan hệ luôn là lựa chọn hàng đầu cho những dự án nhỏ bởi những ưu điểm bởi chỉ mục trong SQL, Các chỉ mục trong SQL giúp chúng ta tìm một bản ghi hoặc danh sách các bản ghi dữ liệu bằng cách khớp với các điều kiện của mệnh đề WHERE . Nó có thể giúp các truy vấn tìm kiếm một giá trị cụ thể hoặc các giá trị bên trong một dải giá trị. Nó làm cho việc tìm kiếm nhanh hơn, điều này cuối cùng dẫn đến việc nâng cao hiệu suất của truy vấn. Các câu lệnh như SELECT , UPDATE và DELETE tận dụng tối đa các chỉ mục để tăng khả năng thực thi tìm kiếm

Tuy nhiên, Chỉ mục nhóm được sử dụng trong SQL không chiếm thêm bất kỳ không gian nào vì nó lưu trữ thứ tự vật lý của các bản ghi bảng trong cơ sở dữ liệu. Vì chỉ có một thứ tự vật lý của bảng, nên chỉ có một chỉ mục được nhóm. Nhưng đối với các chỉ mục không phân cụm cần thêm dung lượng đĩa thì khác. Chỉ mục không phân cụm là cấu trúc riêng biệt với các hàng dữ liệu trong bảng. Nó sắp xếp lại một hoặc nhiều cột vì thứ tự logic khác với thứ tự vật lý. Thứ tự vật lý của dữ liệu là thứ tự mà nó được lưu trữ trong cơ sở dữ liệu. Các chỉ mục có hậu quả kém về hiệu suất của các câu lệnh sửa đổi dữ liệu như INSERT , UPDATE hoặc DELETE . Mỗi khi truy vấn yêu cầu sửa đổi dữ liệu trong bảng, cơ sở dữ liệu sẽ tự cập nhật chỉ mục mới nơi dữ liệu thay đổi. Như đã thảo luận trước đó, các chỉ mục giúp chúng tôi xác định vị trí các bản ghi nhanh hơn, dẫn đến hiệu suất sắp xếp và tìm kiếm nhanh hơn. Do đó, có quá nhiều chỉ mục có thể giúp chúng tôi tìm thấy các bản ghi nhanh hơn nhưng lại có tác động kém đến tốc độ sửa đổi dữ liệu . Vì vậy, chúng ta cần có một số lượng tốt chỉ số cân bằng hiệu suất hệ thống. Hơn thế nữa, đến khi dữ liệu đủ lớn, những truy vấn tìm kiếm sẽ gây ra hậu quả nghiêm trọng khi phải chời đợi một thời gian đủ dài để làm time out một request

### **Tìm kiếm nhanh trên các công cụ tìm kiếm**

Elasticsearch đã quá phổ biến tại MISA, với những lý do khiến cho elasticsearch trở nên phổ biến như sau: Là phần mềm mã nguồn mở, hoàn toàn miễn phí, cộng đồng phát triển lớn. Tốc độ nhanh: Elasticsearch có khả năng thực hiện những câu truy vấn phức tập một cách nhanh chóng và cũng có thể lưu lại hầu hết cấu trúc truy vấn vào bộ nhớ đệm để sử dụng cho việc filter kết quả… Hỗ trợ Full-text search: với các tính năng như tách từ, tách câu, tạo chỉ mục cho dữ liệu Hỗ trợ tìm kiếm mờ, tự động hoàn thành (autocomplete): giúp bạn có thể tìm ra kết quả kể cả khi bạn viết sai chính tả. Cung cấp Restful API cho phép xử lý các yêu cầu với các API Restful request. Dữ liệu lưu dưới dạng document oriented, free schema nên rất linh hoạt cho những trường hợp dữ liệu thường xuyên thay đổi cấu trúc. Khả năng mở rộng và tính sẵn dùng cao: với việc sử dụng mô hình cluster với nhiều node cùng tham gia phục vụ việc sử lý dữ liệu, khi một node chết thì vẫn không ảnh hưởng tới luồng xử lý, ngược lại khi muốn mở rộng ta chỉ cần thêm node mới vào hệ thống.

Sẽ có nhiều thắc mắc giữa Elasticsearch và MongoDB vì chúng khá giống nhau. Mặc dù Elasticsearch và MongoDB đều hỗ trợ cơ chế đánh index cho dữ liệu. Và khi các field của MongoDB được đánh index đầy đủ thì khả năng tìm kiếm của MongoDB cũng không kém cạnh so với Elasticsearch. Nhưng Elasticsearch vẫn có những điểm riêng biệt như: Cung cấp Restful API cho phép xử lý các yêu cầu dưới dạng các request API Restful. Tự động đánh index khi insert dữ liệu Việc đánh index ở mức độ chuyên sâu hơn so với MongoDB, trong khi MongoDB chỉ đánh index ở mức độ từ /words thì Elasticsearch chia nhỏ hơn nên MongoDB sẽ không phù hợp cho trường hợp tìm kiếm mờ (ví dụ bạn có 1 đoạn text ‘tran van b’ khi đánh index trong MongoDB nó sẽ tách ra làm 3 từ là [‘tran’, ‘van’, ‘b’] nếu bạn search từ ‘a’ thì nó sẽ không tìm thấy).

### **Cơ chế đồng bộ dữ liệu**

Bài toán tìm kiếm với dữ liệu lớn chỉ được giải quyết một nửa nếu như không giải quyết bài toán thư hai là đồng bộ dữ liệu giữa cơ sở dữ liệu quan hệ và cơ sở dữ liệu tìm kiếm, đây là bài toán cốt lõi làm nên thành công của bài toán tìm kiếm nhanh với dữ liệu lớn, không những thế, đặc thù của một vài phần mềm tại MISA là khách hàng luôn mong muốn tìm kiếm thời gian thực. Tại MISA cũng đã giải quyết bài toán này bằng những phương pháp đơn giản như thêm dữ liệu tại cả hai cơ sở dữ liệu mỗi khi có dữ liệu mới phát sinh, hoặc đồng bộ dữ liệu sau một khoảng thời gian cố định.

Ưu điểm của những phương pháp được coi là truyền thống này là đơn giản và dễ hiểu. Tuy nhiên, tồn tại trong đó là những rủi ro như, dữ liệu không đồng nhất do vấn đề coding, vấn đề hiệu năng khi phải thêm cùng lúc tại hai cơ sử dữ liệu.

Hiện nay, đã có rất nhiều những phương pháp đơn giản và hiệu quả hơn, cùng với đó là nhưng công cụ có sẵn hỗ trợ giải quyết bài toán này, một trong những phương pháp đó là CDC hoặc cơ chế đồng bộ do Apache Kafka, … cung cấp.

Mỗi một phương pháp hay công cụ hỗ trợ điều có những điểm mạnh riêng, tuy nhiên để tìm ra một phương pháp hiệu quả nhất và phù hợp nhất với MISA thì cần có sự nghiên cứu và thực nghiệm với những số liệu cụ thể.

## Tiềm năng ứng dụng của bài toán tìm kiếm dữ liệu lớn và đồng bộ dữ liệu real-time tại công ty cổ phần MISA JSC

Số lượng sản phầm của MISA ngày càng phát triển cả về số lượng cũng như chất lượng, cùng với đó là tập dữ liệu khách hàng khổng lồ từ các phần mềm, việc dữ liệu lớn được lưu trữ trên cơ sở dữ liệu quan hệ cho ra một hiệu suất truy vấn không được tốt, MISA cũng đã có những sự cô liên quan đến việc tím kiếm dữ liệu lớn, có thể nói bài toán tìm kiếm với dữ liệu lớn được coi là “gãi đúng chỗ ngứa” của MISA bấy lâu nay, việc hiện thực hóa bài toán lúc này là cần thiết hơn bao giờ hết nhằm đem lại trải nghiệm tốt cho người dùng và nâng cao giá trị sản phẩm MISA.

## 1.4 Mục tiêu của nghiên cứu

## 1.5 Cấu trúc bài nghiên cứu

Dựa trên mục tiêu cụ thể đã trình bày, bài nghiên cứu được chia ra làm bốn chương với nội dung cụ thể như sau:

Chương 1: Giới thiệu chung

Chương 2: Đồng bộ dữ liệu real-time giữa CSDL gốc và CSDL tìm kiếm

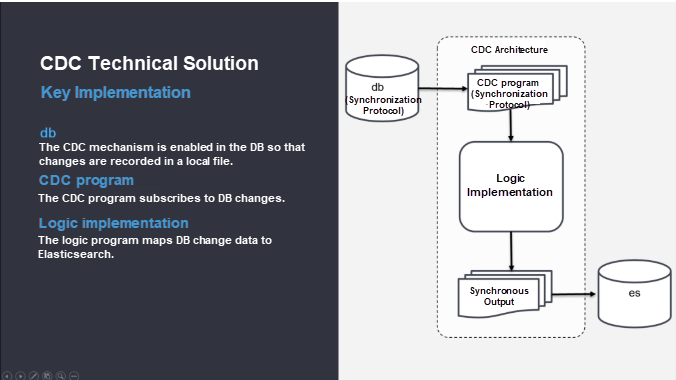
Chương 3: Kết quả thực nghiệm

Chương 4: Kết luận

**CHƯƠNG 2. ĐỒNG BỘ DỮ LIỆU REAL-TIME GIỮA CSDL GỐC VÀ CSDL TÌM KIẾM**

**2.1 Cơ chế đồng bộ CDC**

Dữ liệu thay đổi được ghi lại từ DB, được đẩy sang một chương trình trung gian và được đẩy đồng bộ đến Elaticsearch bằng cách sử dụng logic của chương trình trung gian. Dựa trên cơ chế CDC, dữ liệu chính xác được trả về ở tốc độ cực nhanh để đáp ứng với các truy vấn. Giải pháp này ít ảnh hưởng đến luồng chạy chính của các ứng dụng do đó, nó có thể được trừu tượng hóa và tách biệt với các hệ thống, làm cho nó phù hợp cho việc sử dụng quy mô lớn. Điều này được minh họa trong hình sau.

**

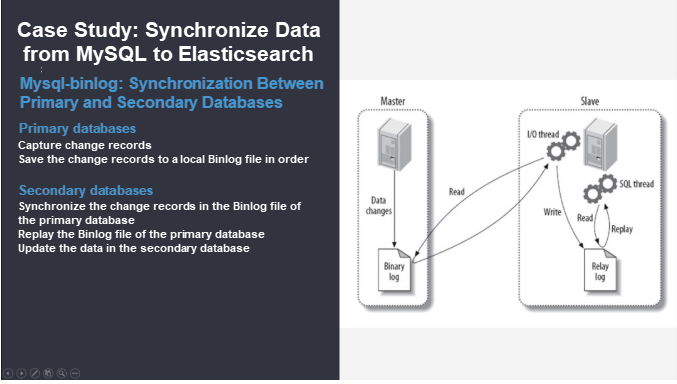
## 2.2 Đồng bộ theo cơ chế Binlog Mysql

Đồng bộ dữ liệu sử dụng cơ chế Binlog Mysql Binlog là một cơ chế chức năng được cung cấp bởi MySQL. BinLog là một tệp nhị phân trên đĩa, chứa tất cả các sự kiện thay đổi nội dung hoặc cấu trúc của cơ sở dữ liệu MySQL, ví dụ: chèn, cập nhật, xóa. Tất cả các thay đổi sẽ được lưu thêm vào (append only) vào một file log được đánh số thứ tự. Mỗi thay đổi được đặc trưng bởi hai tham số: file log được ghi vào vị trí offset trên file log. MySql Binlog có ba loại:

* Statement Based: lưu trữ các câu lệnh làm thay đổi dữ liệu
* Row based: lưu trữ các dữ liệu thay đổi
* Mixed: lưu trữ cả statement và data thay đổi của từng row

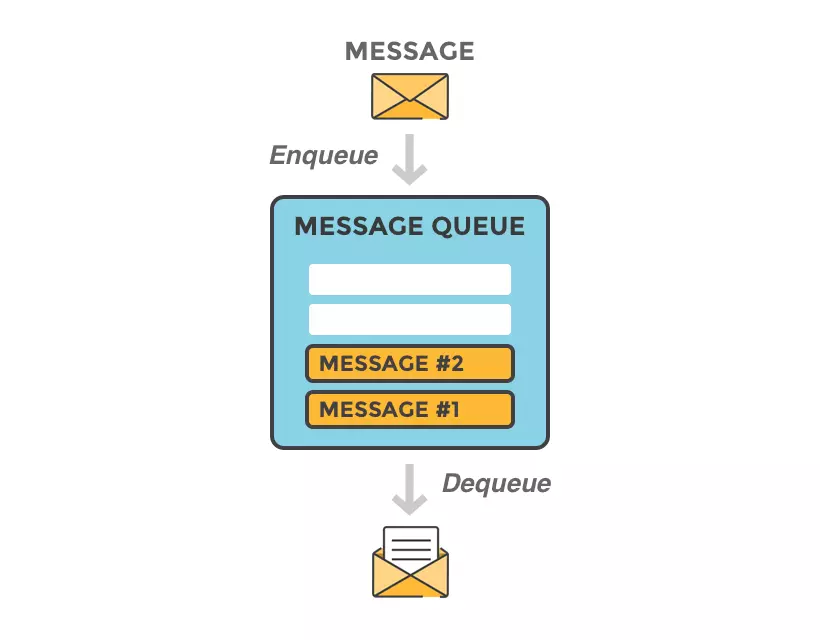
Mysql hỗ trợ cơ chế replicate binary log. MySql sẽ streamming tất cả các thay đổi của database từ master tới slave. Toàn bộ binary log sẽ được replicate realtime tới slave. Nó ban đầu được thiết kế để đồng bộ hóa giữa cơ sở dữ liệu chính và phụ:

* Cơ sở dữ liệu chính ghi dữ liệu vào tệp binlog khi cơ chế BinLog được bật.
* Cơ sở dữ liệu phụ lấy dữ liệu binlog từ cơ sở dữ liệu chính, phát lại dữ liệu binlog và cập nhật dữ liệu trong cơ sở dữ liệu phụ.
* Khi bạn bật Binlog, lưu ý rằng: Một cơ sở dữ liệu chính thường được liên kết bởi nhiều cơ sở dữ liệu phụ và phải cập nhật các thay đổi thời gian thực cho hệ thống, dẫn đến tắc nghẽn tài nguyên máy chủ. Các bảng được đồng bộ hóa phải có các khóa chính.



## 2.3 Message Queue

Định nghĩa: Message queue là một kiến trúc cung cấp giao tiếp không đồng bộ. Ý nghĩa của queue ở đây chính là 1 hàng đợi chứa message chờ để được xử lý tuần tự theo cơ chế vào trước thì ra trước (FIFO - First In First Out).

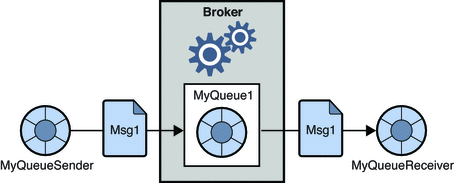
Một message là các dữ liệu cần vận chuyển giữa người gửi và người nhận. Vậy có thể hiểu đơn giản, message queue giống như một hòm thư email của chúng ta. Email có lẽ là ví dụ tốt nhất về giao tiếp không đồng bộ. Khi một email được gửi đi, người gửi tiếp tục xử lý những thứ khác mà không cần phản hồi ngay lập tức từ người nhận. Cách xử lý tin nhắn này tách người gửi khỏi người nhận để họ không cần phải tương tác với hàng đợi tin nhắn cùng một lúc.

Kiến trúc cơ bản của message queue rất đơn giản, bao gồm các thành phần như sau:

* Message: Thông tin được gửi (có thể là text, binary hoặc JSON)
* Producer: Service tạo ra thông tin, đưa thông tin vào message queue.
* Message Queue: Nơi chứa những message này, cho phép producer và consumer có thể trao đổi với nhau
* Consumer: Service nhận message từ message queue và xử lý
* Một service có thể vừa làm producer, vừa làm consumer

Một số message queue phổ biển hiện nay: Kafka, RamitMQ, Pulsar, ZeroMQ, IronMQ, RocketMQ, ..

Tại sao nên sử dụng Message queue.

* Ưu điểm:
  + Dễ scaling hệ thống: Vào giờ cao điểm, nhiều truy vấn, ta có thể tăng số lượng consumer lên để xử lý được nhiều messege hơn. Khi không cần ta có thể giảm lại.
  + Phân tán hệ thống: Giúp phân tách hệ thống thành nhiều service nhỏ hơn, mỗi service chỉ xử lý 1 chức năng nhất định theo cấu trúc [microservice](https://lcdung.top/phat-trien-phan-mem-theo-kien-truc-microservice/).
  + Đảm bảo duration/recovery: Do message đã được lưu trong queue, khi 1 service đang xử lý nhưng bị crash, ta không lo bị mất data vì có thể lấy message từ trong queue ra và retry. Trong 1 hệ thống có nhiều consumer, nếu vài consume crash cũng không làm crash cả hệ thống
  + Hỗ trợ rate limit, batch process: Trong trường hợp khả năng xử lý của hệ thống có hạn (chỉ có thể xử lý 100 lượt release/s) mà phải xử lý 10000 lượt release. Với message queue, ta có thể lấy từng lượt release chưa xử lý trong queue ra xử lý từ từ, không sợ bị mất.
* Nhược điểm:
  + Làm phức tạp hệ thống
  + Phải có message format: Từ 2 phía producer và consumer cần phải thống nhất format message để có thể gửi và nhận message.
  + Monitor Queue là cần thiết: Bạn cần phải theo dõi queue của mình để đảm bảo queue không quá nhiều hay đầy queue.
  + Khó xử lý đồng bộ: Không phải lúc nào message queue cũng là lựa chọn hàng đầu khi chúng ta xây dựng hệ thống. Sẽ có nhiều trường hợp hệ thống bắt buộc cần phải xử lý đồng bộ giữa các service, khi đó ta sẽ cần lựa chọn những cơ chế phù hợp hơn như Remote Procedure Invocation (RPI).
* Các loại message queue.
  + Point-to-point: Message queue có thể là kiểu point-to-point, tức là khi đó ta chỉ có một hàng đợi và một consumer duy nhất dể xử lý các tin nhắn trong hàng đợi:
  + Diagram

    Description automatically generatedPublisher-Subscriber. message queue có thể sử dụng định dạng Publisher-Subscriber, trong đó publisher (nhà sản xuất) gửi tin nhắn đến hàng đợi (trong trường hợp này được gọi là Topic) và tất cả subscriber (người đăng ký) vào cùng 1 Topic đều sẽ nhận được tin nhắn trong Topic đó:

## 2.4. Các nền tảng và website hiện nay

Magento là một nền tảng thương mại điện tử mã nguồn mở được viết bằng PHP. Nó sử dụng các framework PHP khác như Laminas và Symfony. Mã nguồn Magento được phân phối theo Bản quyền Phần mềm Mở v3.0. Magento được mua bởi Adobe Inc vào tháng 5/2018. Công cụ tìm kiếm được tíc hợp sẵn trong mangento cloud: Elasticsearch là chức năng mới được thêm vào ở các phiên bản Magento 2.4. Magento Cloud giải quyết bài toán đồng bộ bằng cách sử dụng zookeeper và kafka.

## 2.4 Tool đồng bộ bên thứ ba

**2.5 Kết luận**

**CHƯƠNG 3. KẾT QUẢ THỰC NGHIỆM**

# **CHƯƠNG 4. KẾT LUẬN**

# **TÀI LIỆU THAM KHẢO**