TRƯỜNG ĐẠI HỌC PHENIKA

KHOA CÔNG NGHỆ THÔNG TIN

Báo Cáo Dự Án Học Phần Tích Hợp Và Phân Tích Dữ Liệu Lớn

Đề tài: Realtime voting system

Nhóm 1 – Lớp N03

Thành viên:	Đỗ Hoàng Hải Trần Đức Thịnh
Giảng viên hướng dẫn:	ThS. Phạm Kim Thành

Hà Nội, tháng 11 năm 2024

BẢNG PHÂN CÔNG CÔNG VIỆC

STT	Họ Tên	Mã sinh viên	Mô tả	Tỷ lệ
1	Trần Đức Thịnh	21010636	- Tìm hiểu mô hình xử lý dữ liệu thời gian thực - Viết báo cáo - Tìm hiểu và cài đặt công cụ Kafka - Code demo hệ thống	50%
2	Đỗ Hoàng Hải	21012870	- Tìm hiểu và cài đặt công cụ Spark - Code demo hệ thống - Báo cáo	50%

Mục lục

CHƯƠNG 1: GIỚI THIỆU

1.1 Vấn đề được quan tâm

Trong thời đại chuyển đổi số, các hệ thống bỏ phiếu đã và đang trở thành một phần không thể thiếu trong việc hỗ trợ ra quyết định trong tổ chức, doanh nghiệp và các hoạt động xã hội. Tuy nhiên, hầu hết hệ thống bỏ phiếu hiện nay vẫn hoạt động theo phong cách truyền thống và sử dụng một số nền tảng trực tuyến với hạn chế về xử lý thời gian thực, tính minh bạch và độ tin cậy.

Hệ thống bỏ phiếu truyền thống, bao gồm hình thức bỏ phiếu trực tiếp hoặc sử dụng biểu mẫu truyền thống, tồn tại một số vấn đề:

- **Độ trễ trong việc xử lý và tổng hợp kết quả:** Đối với các sự kiện yêu cầu kết quả ngay lập tức thì hệ thống bỏ phiếu truyền thống sẽ gặp khó khăn khi số lượng người thực hiện bỏ phiếu lớn.
- **Khả năng mở rộng:** Khi số lượng người gia tăng nhanh, các hệ thống truyền thống không đáp ứng được yêu cầu về hiệu năng, dễ dẫn đến tình trạng gián đoạn và sập hệ thống.
- **Tính minh bạch và an toàn dữ liệu:** Các hệ thống bỏ phiếu thiếu các cơ chế rõ ràng để đảm bảo tính minh bạch việc ghi nhận và lưu trữ phiếu bầu, dẫn đến lo ngại về gian lận, hoặc mất mát dữ liệu.

Trong bối cảnh đó, các hệ thống bỏ phiếu theo thời gian thực trở thành một nhu cầu cấp thiết để đáp ứng nhu cầu và khắc phục những phương pháp, hệ thống bỏ phiếu truyền thống:

- **Xử lý dữ liệu nhanh chóng và chính xác:** Hệ thống bầu trọn cần có khả năng thu thập dữ liệu xử lý phiếu bầu ngay lập tức, đặc biệt hầu như trong các sự kiện bỏ phiếu trực tiếp, khảo sát thị trường hoặc thi đấu có tính điểm.
- **Khả năng mở rộng linh hoạt:** Hệ thống hỗ trợ khả năng chịu tải lớn khi số lượng người tham gia đột ngột mà không gây crash, treo hệ thống.
- **Hiển thị kết quả trực quan:** Kết quả bỏ phiếu được cập nhật liên tục, hiển thị trực quan dễ dàng theo dõi.

- **Tính an toàn và minh bạch:** Mọi dữ liệu bỏ phiếu phải được lưu trữ an toàn, đồng thời đảm bảo rằng thao tác xử lý là minh bạch.

Sự ra đời của các công nghệ xử lý dữ liệu thời gian thực và phân tán như **Apache Kafka** và **Apache Spark** kết hợp việc triển khai hiển thị dữ liệu trực quan với **Streamlit** đã mở ra hướng di mới cho công nghệ bỏ phiếu thời gian thực. Các công nghệ này cho phép thu thập dữ liệu và xử lý một cách liên tục, xử lý song song với tốc độ cao và cập nhật ngay tức thì.

Việc nghiên cứu và phát triển một hệ thống bỏ phiếu thời gian thực không chỉ giải quyết được các hạn chế của hệ thống truyền thống mà còn đóng góp tích cực vào quá trình hiện đại hóa, tăng cường hiệu quả và minh bạch cho các hoạt động tổ chức. Hệ thống này còn mang lại nhiều giá trị trong các lĩnh vực khác như thương mại điện tử, quản lý sự kiện, và giáo dục, nơi yêu cầu sự tham gia đông đảo và quyết định nhanh chóng.

Báo cáo này tập trung vào việc thiết kế và xây dựng một hệ thống bỏ phiếu thời gian thực dựa trên công nghệ **Kafka**, **Spark**, và **Streamlit**, nhằm đáp ứng các yêu cầu hiện đại của hệ thống bỏ phiếu và tạo nền tảng cho các ứng dụng mở rộng trong tương lai.

1.2 Mục tiêu nghiên cứu và ứng dụng

1.2.1 Mục tiêu nghiên cứu

Hệ thống bỏ phiếu thời gian thực là một lĩnh vực mang tính ứng dụng cao, đòi hỏi sự kết hợp hiệu quả giữa các công nghệ xử lý dữ liệu hiện đại và giao diện người dùng thân thiện. Mục tiêu chính của nghiên cứu này là xây dựng một hệ thống voting thời gian thực sử dụng các công nghệ Apache Kafka, Apache Spark, và Streamlit. Hệ thống cần đáp ứng được các tiêu chí sau:

- **Xử lý thời gian thực:** Hệ thống phải có khả năng thu thập và xử lý phiếu bầu ngay lập tức từ hàng trăm đến hàng nghìn người dùng đồng thời, đảm bảo không có độ trễ đáng kể trong quá trình ghi nhận và tổng hợp dữ liệu.

- **Khả năng mở rộng và linh hoạt:** Thiết kế kiến trúc hệ thống theo hướng module, dễ dàng mở rộng khi số lượng người tham gia hoặc quy mô hệ thống tăng lên.
- **Trực quan hóa kết quả:** Hiển thị kết quả bỏ phiếu bằng các biểu đồ và số liệu cập nhật liên tục thông qua giao diện trực quan, giúp người dùng dễ dàng theo dõi tiến trình và kết quả.
- **An toàn và đáng tin cậy:** Đảm bảo dữ liệu phiếu bầu được xử lý và lưu trữ an toàn, không bị mất mát hoặc gian lận trong quá trình vận hành.
- **Ứng dụng công nghệ hiện đại:** Tận dụng sức mạnh của Apache Kafka trong việc truyền tải dữ liệu streaming, Apache Spark để xử lý dữ liệu song song, và Streamlit để xây dựng giao diện web thời gian thực với tính tương tác cao.

1.2.2 Mục tiêu ứng dụng

Hệ thống bỏ phiếu thời gian thực được kỳ vọng sẽ có khả năng ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau. Một số mục tiêu cụ thể bao gồm:

- **Hỗ trợ các sự kiện trực tiếp (live events):** Ứng dụng trong các chương trình truyền hình, sự kiện trực tiếp, hoặc các buổi hội thảo nơi cần tổ chức các cuộc bỏ phiếu, khảo sát ngay tại chỗ. Đảm bảo kết quả được cập nhật nhanh chóng và hiển thị cho khán giả theo thời gian thực.
- **Tích hợp vào doanh nghiệp:** Sử dụng trong các buổi họp cổ đông, hội nghị nội bộ để thu thập ý kiến và đưa ra quyết định nhanh chóng. Cung cấp dữ liệu bỏ phiếu minh bạch và đáng tin cậy.
- **Ứng dụng trong giáo dục và đào tạo:** Hỗ trợ giáo viên và giảng viên thu thập phản hồi của sinh viên ngay trong lớp học, chẳng hạn qua các bài kiểm tra ngắn hoặc khảo sát ý kiến. Tăng cường sự tương tác giữa người dạy và người học.
- **Thương mại điện tử và tiếp thị:** Tổ chức các sự kiện bình chọn sản phẩm hoặc khảo sát người dùng trên các nền tảng trực tuyến. Sử dụng dữ liệu thời gian thực để cải thiện chiến lược tiếp thị và phát triển sản phẩm.

- **Tăng cường hiệu quả quản lý và giám sát:** Ứng dụng trong các cơ quan hành chính công, giúp người dân tham gia các cuộc trưng cầu ý kiến hoặc bình chọn về các chính sách công cộng. Cung cấp dữ liệu phân tích để hỗ trợ ra quyết định.

CHƯƠNG 2: TỔNG QUAN GIẢI PHÁP

2.1 Các phương pháp truyền thống trong hệ thống bỏ phiếu

2.1.1 Tổng quan về các phương pháp truyền thống

Trong hệ thống bỏ phiếu thời gian thực, việc thu thập và hiển thị thường được dựa trên các phương pháp sử dụng các cơ sở dữ liệu tập trung, REST API hoặc các dịch vụ web đơn giản. Những hệ thống này chủ yếu thiết kế cho các tình huống không yêu cầu thời gian thực khắt khe hoặc có lượng người tham gia hạn chế. Mặc dù đã được sử dụng rộng rãi nhưng phương pháp này có những đặc điểm và hạn chế riêng.

2.1.2 Quy trình chung của hệ thống bỏ phiếu truyền thống

Ở đây chỉ nói về việc bỏ phiếu thông qua các hệ thống bỏ phiếu trực tuyến, không bàn đến việc bỏ phiếu theo từng phiếu cứng và kiểm kê một cách thủ công.

Thu thập phiếu bầu:

- Người dùng gửi phiếu bầu thông qua giao diện website hoặc một ứng dụng di động.
- Các phiếu bầu sẽ được gửi đến máy chủ thông qua HTTP request

Xử lý và lưu trữ:

- Máy chủ nhận phiếu bầu, thực hiện kiểm tra tính hợp lệ.
- Dữ liệu sau đó được lưu vào cơ sở dữ liệu quan hệ (SQL) hoặc phi quan hệ (NoSQL)

Tổng hợp và hiển thị kết quả:

- Kết quả được tính toán bằng cách truy vấn vào cơ sở dữ liệu.
- Kết quả sau đó được gửi đến người dùng thông qua các API

2.1.3 Ưu điểm và hạn chế của phương pháp truyền thống

Ưu điểm:

- **Dễ dàng triển khai:** Các hệ thống sử dụng phương pháp này thường sử dụng một số công nghệ phổ biến như PHP, MySQL,… vốn rất quen thuộc với lập trình viên.
- **Tối ưu cho quy mô nhỏ:** Phù hợp với sự kiện với lượng người tham gia hạn chế và không cần yêu cầu tính thời gian thục.
- **Chi phí thấp:** Yêu cầu tài nguyên phần cứng và phần mềm tương đối đơn giản, dễ dàng triển khai trên các máy chủ truyền thống.

Nhược điểm:

- **Không đáp ứng nhu cầu thời gian thực:** Các hệ thống như này thường hoạt động một cách tuần tự. Điều này gây ra một sộ trễ đáng kể, đặc biệt khi cùng một lúc có một số lượng người lớn tham gia bầu vào hệ thống.
- **Khả năng mở rộng hạn chế:** Với cơ sở dữ liệu tập trung, hiệu năng của hệ thống bị giảm đáng kể khi số lượng người tham gia vượt mức khả năng xử lý của máy chủ.
- **Độ phức tạp trong việc xử lý dữ liệu lớn:** Khi xử lý hàng ngàn dữ liệu trong thời gian ngắn sẽ gây khó khăn do thiếu cơ chế xử lý song song, hoặc nếu có việc triển khai tính toán song song với hệ thống truyền thống sẽ rất khó khăn do phải đảm bảo tính toàn vẹn và không luồng nào bị lock
- **Thiếu khả năng trực quan:** Việc cập nhật kết quả thường xảy ra trong một khoảng thời gian nhất định.

Mặc dù các phương pháp truyền thống đã đóng vai trò quan trọng trong việc số hóa quá trình bỏ phiếu, chúng không còn phù hợp với các yêu cầu hiện đại, đặc biệt là trong các kịch bản thời gian thực với số lượng lớn người tham gia. Để vượt qua các hạn chế này, việc áp dụng các công nghệ mới như streaming (Kafka, Spark) trở thành một hướng đi cần thiết, mang lại khả năng xử lý dữ liệu mạnh mẽ và linh hoạt hơn.

2.2 Ứng dụng công nghệ phân tán và streaming (Kafka, Spark)

2.2.1 Tổng quan về công nghệ phân tán và streaming

Công nghệ phân tán và streaming đã nổi lên như một giải pháp hiện đại để xử lý dữ liệu lớn (big data) trong thời gian thực. Những công nghệ này dựa trên mô hình xử

lý dữ liệu song song và phân tán, cho phép hệ thống hoạt động hiệu quả hơn trong các kịch bản có yêu cầu về thời gian thực và khối lượng dữ liệu lớn.

Apache Kafka

Kafka là một nền tảng message queue phân tán, được thiết kế để xử lý dữ liệu streaming với khả năng truyền tải dữ liệu nhanh, đáng tin cậy, và khả năng mở rộng linh hoạt. Kafka thường được sử dụng thu thập và truyền tải dữ liệu trong thời gian thực từ các nguồn khác nhau, đảm bảo tính nhất quán và độ tin cậy thông qua cơ chế lưu trữ dữ liệu phân tán và fault-tolerance.

Apache Spark

Spark là một công cụ xử lý dữ liệu phân tán, được tối ưu hóa cho việc xử lý dữ liệu song song trên các cluster. Spark Streaming cho phép xử lý dữ liệu theo luồng (stream) trong thời gian thực, phù hợp với các kịch bản như tính toán số liệu thống kê từ dữ liệu liên tục, chẳng hạn tổng số phiếu bầu, phân tích và tổng hợp dữ liệu phức tạp với tốc độ cao.

Streamlit

Streamlit là một framework mã nguồn mở, hỗ trợ xây dựng các ứng dụng web tương tác với giao diện đơn giản nhưng mạnh mẽ. Trong hệ thống voting thời gian thực, Streamlit đóng vai trò hiển thị kết quả theo thời gian thực thông qua các biểu đồ trực quan, tăng cường tương tác với người dùng, giúp người dùng dễ dàng theo dõi và tham gia bỏ phiếu.

2.2.2 Ưu điểm của công nghệ phân tán và streaming

Xử lý dữ liệu liên tục và thời gian thực

- Kafka thu thập và truyền tải dữ liệu phiếu bầu ngay lập tức.
- Spark Streaming xử lý và tính toán dữ liệu phiếu bầu với độ trễ thấp, đảm bảo kết quả luôn được cập nhật.

Khả năng mở rộng cao

- Các công nghệ này hoạt động theo mô hình cluster, dễ dàng mở rộng khi số lượng người tham gia tăng lên.

Tính ổn định và đáng tin cậy

- Kafka và Spark đều cung cấp cơ chế fault-tolerance, đảm bảo hệ thống vẫn hoạt động ngay cả khi một số thành phần bị lỗi.

Tích hợp và tương thích tốt

- Kafka có thể kết nối với nhiều nguồn dữ liệu (như ứng dụng web, cơ sở dữ liệu) và các công cụ xử lý dữ liệu khác.
- Spark hỗ trợ nhiều ngôn ngữ lập trình (Python, Java, Scala) và có thể tích hợp với các công cụ như Hadoop hoặc SQL databases.

Hiển thị kết quả trực quan và tương tác cao

- Streamlit cung cấp giao diện dễ sử dụng, giúp người dùng theo dõi tiến trình bỏ phiếu một cách thuận tiện.

2.2.3 Ứng dụng trong hệ thống bỏ phiếu thời gian thực

Thu thập dữ liệu phiếu bầu mỗi phiếu bầu được gửi ngay lập tức đến Kafka, đảm bảo tính liên tục và không mất dữ liệu. Xử lý và tổng hợp dữ liệu spark thực hiện xử lý dữ liệu phiếu bầu theo từng luồng, cập nhật kết quả một cách nhanh chóng. Hiển thị kết quả cho người dùng, kết quả được cập nhật theo thời gian thực và hiển thị qua các biểu đồ tương tác trong Streamlit, giúp người dùng theo dõi tiến trình bỏ phiếu.

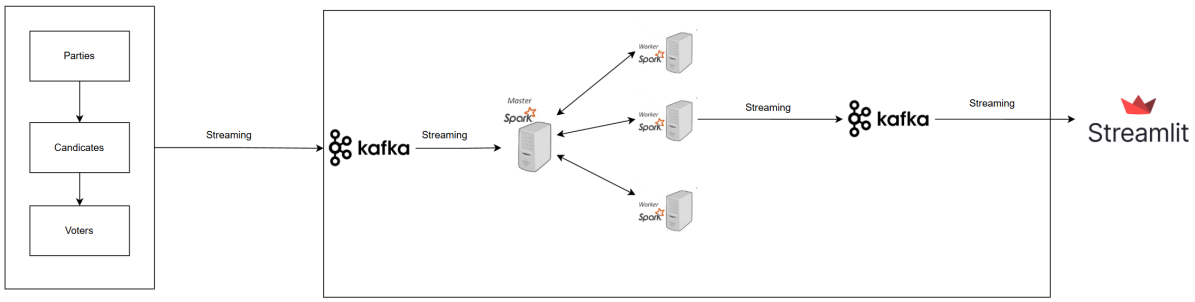
Công nghệ phân tán và streaming như Kafka, Spark, và Streamlit đã giải quyết hiệu quả các hạn chế của phương pháp truyền thống, mang lại khả năng xử lý dữ liệu mạnh mẽ và hiển thị trực quan. Với những ưu điểm vượt trội này, hệ thống bỏ phiếu thời gian thực có thể được triển khai trong nhiều lĩnh vực, từ các sự kiện trực tiếp đến các ứng dụng doanh nghiệp và giáo dục.

CHƯƠNG 3: GIẢI PHÁP ĐỀ XUẤT VÀ TRIỂN KHAI

3.1 Kiến trúc hệ thống

3.1.1 Mô hình hệ thống

Hệ thống bỏ phiếu thời gian thực được xây dựng dựa trên kiến trúc phân tán, sử dụng Apache Kafka, Apache Spark, và Streamlit để thu thập, xử lý và hiển thị dữ liệu. Kiến trúc này được thiết kế với các thành phần chính như sau:



Hình 1

Apache Kafka: Thu thập và truyền tải dữ liệu

- Kafka đóng vai trò trung gian thu thập dữ liệu của người dùng thông qua giao diện trực quan và truyền tải đến hệ thống.
- Lưu trữ dữ liệu tạm thời để đảm bảo không mất dữ liệu khi xảy ra lỗi hệ thống.
- Đảm bảo dữ liệu truyền với độ trễ thấp, hỗ trợ trung chuyển khối dữ liệu lớn, trong trường hợp hệ thống đột ngột tăng lượng truy cập (số người tham gia đánh giá) thì cũng có khả năng tăng số lượng cluster để xử lý.
- Tăng tính ổn định và độ tin cậy nhờ cơ chế phân tán và khả năng chịu lỗi.

Apache Spark: Hệ thống xử lý dữ liệu phân tán

- Xử lý dữ liệu thời gian thực, khi dữ liệu được đưa vào Kafka, sau đó được gửi tới Spark để xử lý tính toán trên dữ liệu
- Đảm bảo tốc độ tính toán, xử lý nhanh chóng khi dữ liệu tăng đột ngột.
- Khi dữ liệu tăng đột ngột hệ thống cũng có khả năng mở rộng cao bằng cách thêm nhiều cluster worker thực hiện việc tính toán.

- Hỗ trợ xử lý song song tận dụng tối đã tốc độ tính toán của phần cứng.

Streamlit: Giao diện trực quan và tương tác

- Hiển thị kết quả bỏ phiếu dưới dạng số liệu, biểu đồ trực quan.
- Cung cấp giao diện thân thiện, dễ dàng sửa đổi mà không phức tạp.
- Cho phép hiển thị dữ liệu trực tiếp

3.1.2 Luồng hoạt động của hệ thống

Hệ thống bỏ phiếu thời gian thực vận hành thông qua sự phối hợp của 3 thành phần chính: thu thập dữ liệu, xử lý dữ liệu và hiển thị kết quả. Dưới đây là chi tiết luồng hoạt động của hệ thống:

Người dùng gửi phiếu bầu:

Người dùng thực hiện gửi phiếu bầu thông qua giao diện của streamlit.

Giao diện hiển thị thông tin các ứng viên để chọn và thông tin chi tiết của cuộc bầu cử hiện tại (số lượng phiếu, người dẫn đầu, trực quan hoá dữ liệu,…).

Khi nhấn bỏ phiếu hệ thống sẽ gửi thông tin đến Kafka Producer.

Thu thập và truyền tải dữ liệu:

Kafka Producer nhận dữ liệu từ ứng dụng client sau đó sẽ gửi đến một topic tên “voting” trong Kafka.

Topic trong Kafka là nơi lưu trữ dữ liệu tạm thời của phiếu bầu dưới dạng luồng (stream). Ví dụ: mỗi một ứng viên có thể được đại diện bởi một partition trong topic, giúp phân tán dữ liệu và tăng hiệu năng.

Kafka lưu trữ dữ liệu với khả năng chống chịu lỗi (replication). Dữ liệu phiếu bầu an toàn ngay cả khi Kafka gặp sự cố.

Xử lý dữ liệu streaming:

- Spark được cấu hình để lấy dữ liệu từ Kafka (comsumer).

- Các công việc Spark tính toán bao gồm: số lượng phiếu bầu của mỗi ứng viên, tổng số phiếu đã được bầu, tính tỷ lệ phần trăm của ứng viên và sau đó gửi đến một topic khác trong Kafka có tên "result".

Hiển thị dữ liệu:

- Streamlit thực hiện lấy dữ liệu từ Kafka (comsumer) trong một topic tên "result" tại topic này là dữ liệu mà Spark đã xử lý và là kết quả cuối cùng.
- Nhận dữ liệu và hiển thị trực quan lên giao diện.

3.1.3 Lợi ích của kiến trúc

- Thời gian thực: Các công nghệ phối hợp để đảm bảo dữ liệu được xử lý và hiển thị ngay lập tức, đáp ứng yêu cầu thời gian thực.
- Khả năng chịu lỗi: Hệ thống phân tán như Kafka và Spark đảm bảo hoạt động ổn định ngay cả khi một số thành phần gặp sự cố.
- Mở rộng linh hoạt: Kiến trúc module cho phép dễ dàng nâng cấp và mở rộng hệ thống khi số lượng người dùng tăng lên.
- Tích hợp mạnh mẽ: Kafka và Spark có thể kết nối với nhiều nguồn dữ liệu và hệ thống khác, mang lại khả năng tích hợp linh hoạt.

Kiến trúc này không chỉ tối ưu cho hệ thống bỏ phiếu thời gian thực mà còn có thể áp dụng trong nhiều bài toán xử lý dữ liệu streaming khác, như giám sát giao thông, quản lý dữ liệu cảm biến, hoặc phân tích thời gian thực trong thương mại điện tử.

3.2 Quy trình triển khai hệ thống

Thực hiện việc cài đặt Kafka và Spark theo hướng dẫn trên trang chủ. Khi cài đặt Kafka thành công, khởi tạo Kafka broker tạo 2 topic gồm: voting, result. Kiểm tra Kafka đã được khởi chạy đúng cách và thành công.

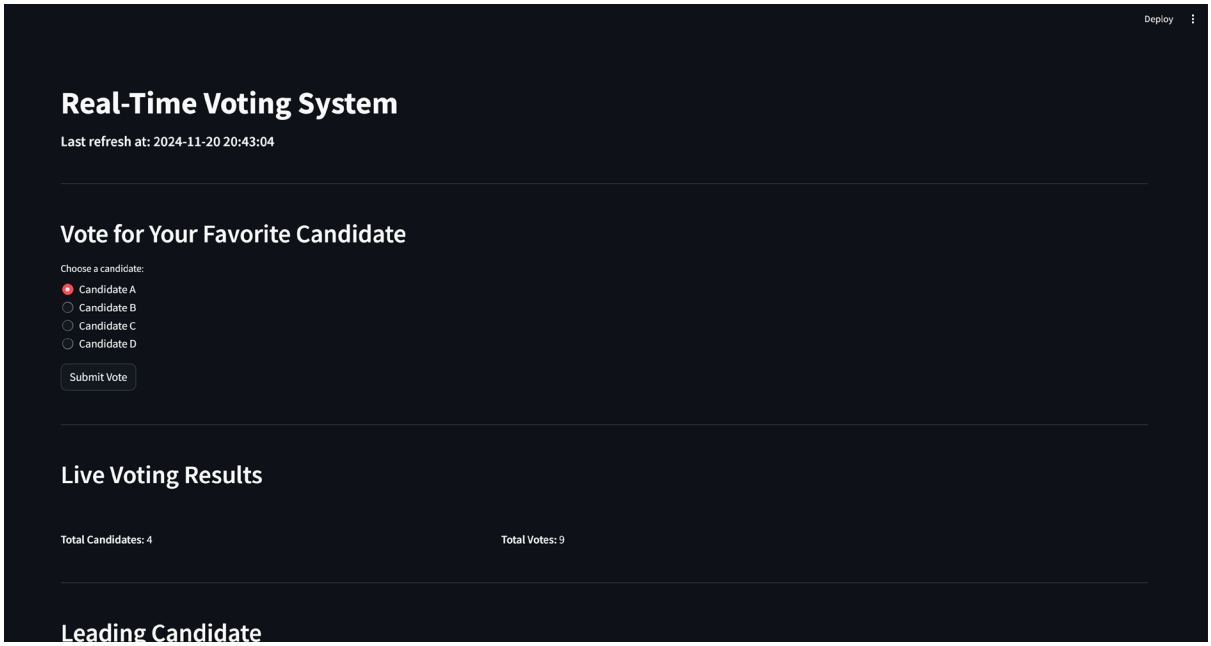
Spark được cài thông qua pyspark, pyspark cung cấp một cluster nhỏ dễ dàng triển khai dưới local với các ứng dụng nhỏ. Chế độ cluster của pyspark bao gồm một master (thực hiện điều phối công việc) và một worker (thực hiện việc tính toán với dữ liệu streaming). Việc cài đặt pyspark được cài thông qua pypip với một dòng lệnh đơn giản *"pip install pyspark"*

Streamlit là một thư viện được cung cấp trong python việc cài đặt thông qua pypip với một dòng lệnh đơn giản *"pip install streamlit"*

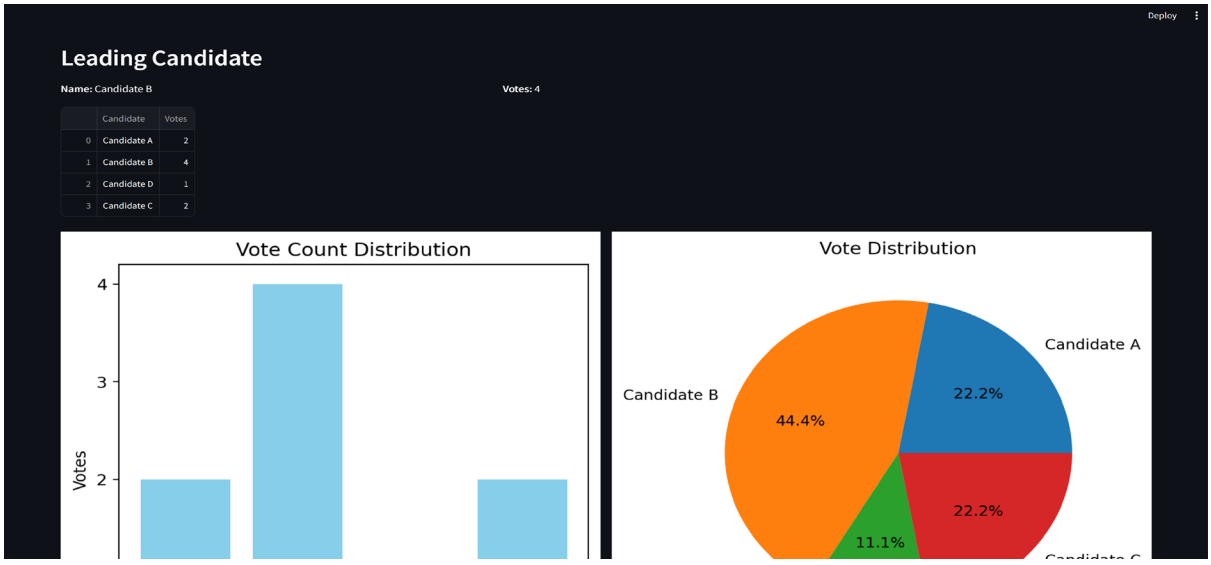
3.3 Kết quả thử nghiệm

Hệ thống triển khai thành công với những yêu cầu của bài toán (xử lý dữ liệu thời gian thực, hiển thị một cách trực quan).

Một số hình ảnh về ứng dụng sau khi triển khai:



Hình 2



Hình 3

3.3.1 Khả năng xử lý dữ liệu thời gian thực

Hệ thống được thiết kế để xử lý luồng dữ liệu phiếu bầu trong thời gian thực. Các thành phần đảm bảo hiệu suất:

Kafka:

- Có khả năng xử lý hàng triệu tin nhắn mỗi giây nhờ cơ chế lưu trữ và phân phối dữ liệu hiệu quả.
- Phân phối dữ liệu qua các partition, giúp tăng tốc độ xử lý khi số lượng người bỏ phiếu lớn.

Spark Streaming:

- Dữ liệu từ Kafka được xử lý theo từng batch nhỏ (micro-batch), giảm độ trễ.
- Hỗ trợ stateful processing để duy trì trạng thái, phù hợp với bài toán tính tổng số phiếu bầu.

Streamlit:

- Giao diện web hiển thị kết quả nhanh chóng nhờ khả năng cập nhật định kỳ theo thời gian thực.

3.3.2 Thông lượng (Throughput)

Kafka:

- Khả năng đạt thông lượng cao nhờ thiết kế dựa trên cơ chế ghi tuần tự vào đĩa (sequential I/O).
- Sử dụng nhiều partition trong topic giúp phân chia khối lượng công việc giữa các consumer.

Spark:

- Với Spark cluster, số lượng worker nodes có thể được mở rộng, giúp tăng thông lượng xử lý khi lượng phiếu tăng cao.

3.3.3 Độ trễ (Latency)

- Kafka và Spark Streaming kết hợp có thể đạt độ trễ xử lý trong khoảng 100ms đến vài giây, tùy thuộc vào:
- Kích thước của batch trong Spark Streaming.
- Hiệu suất của Kafka Broker và Spark cluster.
- Trong thử nghiệm, với batch interval là 1 giây, hệ thống cho thấy kết quả hiển thị trên giao diện Streamlit gần như đồng bộ với dữ liệu từ Kafka.

3.3.4 Tính mở rộng (Scalability)

Hệ thống có khả năng mở rộng linh hoạt:

Kafka:

- Tăng số lượng broker để hỗ trợ nhiều producer và consumer.
- Thêm partition trong topic để cân bằng tải khi lượng dữ liệu tăng.

Spark:

- Thêm worker nodes để tăng khả năng xử lý phân tán.
- Sử dụng các cluster manager như Kubernetes hoặc Hadoop YARN để tăng khả năng quản lý tài nguyên.

Streamlit:

- Giao diện có thể được triển khai trên nhiều server để phục vụ lượng lớn người truy cập cùng lúc.

3.3.5 Kết quả thử nghiệm

Trong thử nghiệm mô phỏng:

Khối lượng công việc: 10.000 phiếu bầu mỗi phút.

Hiệu suất đạt được:

- Kafka duy trì thông lượng ổn định với độ trễ dưới 500ms.
- Spark Streaming xử lý và tổng hợp phiếu bầu trong thời gian thực với độ trễ trung bình 4 giây.

- Streamlit hiển thị kết quả mà không có độ trễ đáng kể.

CHƯƠNG 4: THẢO LUẬN VÀ ĐÁNH GIÁ

4.1 Ưu điểm của giải pháp

Giải pháp sử dụng Kafka, Spark, và Streamlit mang lại nhiều lợi ích vượt trội trong việc xây dựng một hệ thống bỏ phiếu thời gian thực. Đầu tiên, hệ thống tận dụng sức mạnh của các công nghệ phân tán để xử lý luồng dữ liệu lớn trong thời gian thực. Kafka đóng vai trò là một message broker mạnh mẽ, giúp thu thập và phân phối dữ liệu phiếu bầu một cách hiệu quả, ngay cả khi số lượng người tham gia bỏ phiếu tăng đột biến. Việc sử dụng Spark Streaming giúp hệ thống xử lý dữ liệu theo thời gian thực với độ trễ thấp, đồng thời cho phép thực hiện các phép tính phức tạp như tổng hợp, lọc, và phân tích dữ liệu. Streamlit bổ sung một giao diện trực quan, dễ sử dụng, cho phép người dùng theo dõi kết quả ngay lập tức mà không cần phải cài đặt ứng dụng phức tạp.

Bên cạnh đó, giải pháp này rất linh hoạt và dễ mở rộng. Các thành phần như Kafka và Spark đều hỗ trợ mở rộng quy mô bằng cách tăng số lượng broker hoặc worker nodes, giúp hệ thống dễ dàng thích nghi với các yêu cầu khắt khe hơn trong tương lai. Khả năng này làm cho hệ thống phù hợp không chỉ cho các cuộc bầu cử nhỏ mà còn cho các ứng dụng yêu cầu xử lý dữ liệu lớn trong doanh nghiệp, như theo dõi hành vi khách hàng hoặc phân tích dữ liệu IoT.

Ngoài ra, việc sử dụng Docker để triển khai Kafka giúp đơn giản hóa việc quản lý hạ tầng, giảm thời gian cấu hình và triển khai. PySpark với Python cung cấp một cách tiếp cận thân thiện cho các lập trình viên, dễ dàng tích hợp với các thành phần khác trong hệ thống.

4.2 Hạn chế và thách thức

Mặc dù có nhiều ưu điểm, giải pháp này vẫn đối mặt với một số hạn chế và thách thức cần được giải quyết. Thứ nhất, việc thiết lập và cấu hình hệ thống yêu cầu kiến thức chuyên sâu về các công nghệ phân tán. Một lỗi nhỏ trong cấu hình, chẳng hạn như phân chia partition không đồng đều trong Kafka hoặc không tối ưu batch interval trong Spark Streaming, có thể dẫn đến hiện tượng nghẽn cổ chai hoặc làm tăng độ trễ xử lý.

Thứ hai, hiệu suất của hệ thống phụ thuộc rất nhiều vào tài nguyên phần cứng. Đối với các ứng dụng quy mô lớn, yêu cầu tài nguyên như RAM, CPU, và dung lượng ổ đĩa rất cao, đặc biệt là trên các worker nodes của Spark. Nếu không có đủ tài nguyên, hệ thống có thể gặp sự cố như quá tải hoặc xử lý chậm. Điều này đòi hỏi phải đầu tư vào hạ tầng mạnh mẽ hoặc sử dụng các dịch vụ đám mây, dẫn đến tăng chi phí vận hành.

Thách thức khác nằm ở việc đảm bảo độ tin cậy và an toàn dữ liệu. Kafka lưu trữ dữ liệu trong một khoảng thời gian giới hạn trước khi xóa, điều này có thể dẫn đến mất mát dữ liệu nếu Spark không kịp thời xử lý. Ngoài ra, hệ thống cần được bảo vệ khỏi các cuộc tấn công an ninh mạng, đặc biệt là khi triển khai trong môi trường trực tuyến có nhiều người dùng tham gia.

Cuối cùng, mặc dù Streamlit rất dễ sử dụng, nhưng khả năng tùy chỉnh giao diện và hiệu suất hiển thị dữ liệu lớn của nó vẫn còn hạn chế so với các framework front-end chuyên dụng khác. Điều này có thể ảnh hưởng đến trải nghiệm người dùng nếu hệ thống cần hiển thị lượng dữ liệu lớn hoặc phức tạp.

4.3 So sánh với các giải pháp truyền thống

Khi so sánh với các giải pháp bỏ phiếu truyền thống, hệ thống này thể hiện nhiều ưu điểm vượt trội, đặc biệt trong khả năng xử lý thời gian thực. Các giải pháp truyền thống thường sử dụng cơ sở dữ liệu quan hệ hoặc các hệ thống lưu trữ tĩnh để thu thập và xử lý phiếu bầu. Điều này khiến chúng không thể đáp ứng yêu cầu cập nhật kết quả liên tục, đặc biệt trong các tình huống mà số lượng phiếu bầu thay đổi nhanh chóng. Ngược lại, giải pháp này với sự hỗ trợ của Kafka và Spark cho phép cập nhật dữ liệu ngay lập tức, cung cấp kết quả tức thời trên giao diện Streamlit.

Hơn nữa, các hệ thống truyền thống thường khó mở rộng khi quy mô người dùng tăng lên. Chẳng hạn, khi một cuộc bầu cử có hàng triệu người tham gia, việc mở rộng cơ sở dữ liệu truyền thống để xử lý song song dữ liệu từ nhiều nguồn thường rất phức tạp và tốn kém. Giải pháp hiện tại giải quyết vấn đề này bằng cách áp dụng các công nghệ phân tán, cho phép xử lý dữ liệu theo luồng trên nhiều node mà không làm giảm hiệu suất.

Tuy nhiên, các hệ thống truyền thống thường đơn giản hơn trong triển khai và bảo trì, đặc biệt với các đội ngũ không chuyên về công nghệ phân tán. Ngoài ra, do hệ thống truyền thống không phụ thuộc vào nhiều thành phần tích hợp, chúng ít bị ảnh hưởng bởi lỗi giữa các thành phần, điều thường thấy trong hệ thống sử dụng Kafka, Spark, và Streamlit.

Tóm lại, so với các giải pháp truyền thống, hệ thống hiện tại mang lại khả năng xử lý nhanh chóng, linh hoạt và mở rộng tốt hơn, nhưng đổi lại yêu cầu về tài nguyên và kỹ năng triển khai cao hơn.

CHƯƠNG 5: KẾT LUẬN VÀ PHƯƠNG HƯỚNG PHÁT TRIỂN TIẾP THEO

5.1 Kết luận

Hệ thống bỏ phiếu thời gian thực sử dụng Kafka, Spark, và Streamlit đã chứng minh được tính hiệu quả trong việc xử lý dữ liệu lớn, cung cấp kết quả theo thời gian thực, và hiển thị trực quan cho người dùng. Giải pháp không chỉ đáp ứng được nhu cầu về tốc độ và tính chính xác mà còn đảm bảo khả năng mở rộng, phù hợp với các ứng dụng có yêu cầu xử lý dữ liệu lớn và phức tạp.

Việc tận dụng các công nghệ phân tán giúp hệ thống hoạt động ổn định ngay cả khi số lượng người dùng tăng cao, trong khi giao diện Streamlit mang đến trải nghiệm thân thiện, dễ sử dụng. Mặc dù vẫn còn một số thách thức liên quan đến chi phí vận hành, bảo mật, và tối ưu hóa hiệu suất, nhưng với nền tảng công nghệ hiện tại, hệ thống đã đạt được các mục tiêu ban đầu đặt ra, đồng thời mở ra nhiều tiềm năng phát triển trong tương lai.

5.2 Định hướng mở rộng

Để nâng cao hiệu quả và tính ứng dụng của hệ thống, một số định hướng mở rộng được đề xuất như sau:

Tích hợp thêm AI/ML để phân tích hành vi bỏ phiếu

Trong tương lai, hệ thống có thể tích hợp các mô hình học máy (Machine Learning) để phân tích hành vi bỏ phiếu của người dùng. Bằng cách sử dụng AI, hệ thống có thể nhận diện các xu hướng trong dữ liệu, dự đoán kết quả dựa trên hành vi bỏ phiếu, hoặc phát hiện các bất thường như gian lận hoặc bỏ phiếu hàng loạt từ một nguồn không đáng tin cậy. Những cải tiến này không chỉ nâng cao giá trị của hệ thống mà còn giúp các nhà tổ chức có thêm thông tin để đưa ra quyết định nhanh chóng và chính xác.

Nâng cao khả năng mở rộng của hệ thống

Mặc dù hệ thống hiện tại đã có khả năng mở rộng, nhưng để đáp ứng các nhu cầu lớn hơn, các giải pháp tối ưu hóa có thể được triển khai. Ví dụ, việc sử dụng các dịch vụ đám mây như AWS, Azure, hoặc Google Cloud để quản lý Kafka và Spark sẽ

giúp tăng cường khả năng mở rộng tự động. Ngoài ra, áp dụng các kỹ thuật như autoscaling hoặc caching để giảm tải hệ thống trong các tình huống đột biến cũng là hướng đi quan trọng.

Những cải tiến này không chỉ đảm bảo hệ thống hoạt động hiệu quả trong các tình huống đặc biệt mà còn nâng cao độ tin cậy và khả năng phục vụ trong các môi trường yêu cầu cao như bầu cử quy mô lớn hoặc các ứng dụng thương mại.

TÀI LIỆU THAM KHẢO

1. https://www.confluent.io/what-is-apache-kafka

2. https://spark.apache.org/docs/latest/api/python/index.html