THÔNG TIN CHUNG CỦA NHÓM

- Link YouTube video của báo cáo (tối đa 5 phút): https://youtu.be/wOxwvdY4x8A
- Link slides (dạng .pdf đặt trên Github của nhóm): *https://github.com/khoalhd18/CS2205.CH183/blob/main/Slide-FineTuningApacheSparkConfigurationsUsingLargeLanguageModelsADataDrivenApproach.pdf*
- *Mỗi thành viên của nhóm điền thông tin vào một dòng theo mẫu bên dưới*
- *Sau đó điền vào Đề cương nghiên cứu (tối đa 5 trang), rồi chọn Turn in*
- *Lớp Cao học, mỗi nhóm một thành viên*

- Họ và Tên: Lăng Huỳnh Đăng Khoa
- MSSV: 230101051

- Lớp: CS2205.CH183
- Tự đánh giá (điểm tổng kết môn): 8/10
- Số buổi vắng: 2
- Số câu hỏi QT cá nhân: 0
- Số câu hỏi QT của cả nhóm: 0
- Link Github: https://github.com/khoalhd18/CS2205.CH183/

ĐỀ CƯƠNG NGHIÊN CỨU

TÊN ĐỀ TÀI (IN HOA)

TỐI ƯU HÓA CẤU HÌNH APACHE SPARK BẰNG MÔ HÌNH NGÔN NGỮ LỚN: MỘT CÁCH TIẾP CẬN DỰA TRÊN DỮ LIỆU

TÊN ĐỀ TÀI TIẾNG ANH (IN HOA)

FINE TUNING APACHE SPARK CONFIGURATIONS USING LARGE LANGUAGE MODELS: A DATA-DRIVEN APPROACH

TÓM TẮT *(Tối đa 400 từ)*

Trong các hệ thống xử lý dữ liệu lớn ứng dụng Apache Spark thường gặp thách thức trong việc lựa chọn cấu hình phần cứng tối ưu để cân bằng giữa hiệu suất và chi phí tài nguyên sử dụng. Vấn đề này chịu ảnh hưởng bởi nhiều yếu tố khách quan khác nhau khiến người kỹ sư phải gặp rắc rối và mất rất nhiều thời gian kiểm tra từng yếu tố để có thể đúc kết được cấu hình hợp lý tại thời điểm đó. Tuy nhiên, nếu có sự thay đổi xảy ra sẽ phá hỏng thế cân bằng của hệ thống và người kỹ sư sẽ phải lặp lại các bước trên một lần nữa.

Nghiên cứu này đề xuất một phương pháp ứng dụng mô hình ngôn ngữ lớn (LLM) để phân tích dữ liệu lịch sử và tự động đề xuất cấu hình Spark phù hợp. Dữ liệu đầu vào bao gồm kích thước tập dữ liệu, số giờ tính toán lãng phí, kích thước trung bình của tệp và số lượng tệp mỗi lần xử lý. LLM sẽ sử dụng các cấu hình trước đó để suy luận và đưa ra thông số tối ưu nhằm giảm thời gian thực thi và tối ưu hóa tài nguyên. Hệ thống sẽ được triển khai trên đám mây, giúp người dùng dễ dàng nhận khuyến nghị và áp dụng vào thực tế. Đánh giá hiệu suất sẽ được thực hiện bằng cách so sánh hệ thống với các phương pháp tối ưu truyền thống như heuristic-based tuning và grid search.

GIỚI THIỆU *(Tối đa 1 trang A4)*

Xử lý dữ liệu lớn là một thử thách trong thời đại số hóa mới, đặc biệt với các hệ thống yêu cầu khả năng xử lý nhanh và tối ưu tài nguyên. Apache Spark là một trong những công cụ phổ biến cho tác vụ này, nhưng việc lựa chọn cấu hình phù hợp là một vấn đề phức tạp, ảnh hưởng trực tiếp đến hiệu suất hệ thống. Cấu hình không phù hợp có thể dẫn đến lãng phí tài nguyên, chi phí tăng cao, thời gian xử lý kéo dài hoặc làm gián đoạn các quy trình tính toán, huấn luyện mô hình AI phía sau,...

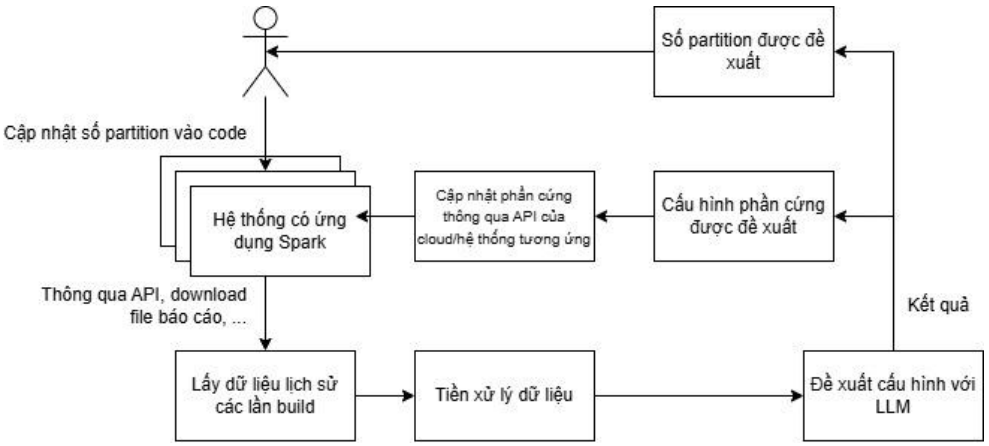
Hiện nay, việc tối ưu hóa cấu hình Spark thường dựa vào kinh nghiệm của kỹ sư dữ liệu hoặc sử dụng các phương pháp heuristic truyền thống. Tuy nhiên, với sự phát triển của mô hình ngôn ngữ lớn (LLM), việc áp dụng AI để phân tích dữ liệu lịch sử và đề xuất cấu hình tối ưu trở thành một hướng đi đầy triển vọng và nhanh chóng. Bằng cách tận dụng khả năng suy luận của LLM, chúng ta có thể yêu cầu nhận diện mẫu dữ liệu và tự động đề xuất các tham số tối ưu mà không cần can thiệp thủ công. Các tham số được nhắc đến bao gồm:

- Số executors
- Số nhân vi xử lý của một executor
- Dung lượng bộ nhớ trong của một executor
- Số nhân vi xử lý của driver (master)
- Dung lượng bộ nhớ trong của driver
- Số partition trong quá trình Spark xử lý dữ liệu

Nghiên cứu này tập trung vào việc áp dụng LLM để phân tích dữ liệu lịch sử, bao gồm: cấu hình phần cứng được sử dụng, dung lượng của input trong một lần build, số lượng file trung bình được đọc lên trong một lần build, số giờ sử dụng tài nguyên lãng phí (Nếu có) và số partition được cấu hình trong một lần build.

Từ đó, hệ thống có thể tự động đề xuất cấu hình Spark phù hợp, giúp giảm thời gian thực thi và tối ưu hóa tài nguyên. Để đảm bảo tính ứng dụng cao, hệ thống sẽ được triển khai trên đám mây, cho phép người dùng dễ dàng nhập thông tin và nhận kết quả

nhanh chóng.



MỤC TIÊU

- **Tối ưu cấu hình phần cứng Spark**: Hệ thống sẽ đề xuất cấu hình Spark tối ưu nhằm giảm lãng phí tài nguyên và tăng hiệu suất xử lý dữ liệu lớn.
- **Tối ưu số partition một lần build**: Hệ thống sẽ đề xuất số partition trong một lần build nhằm giúp spark engine có thể điều phối, sử dụng các executor một cách hiệu quả hơn.
- **Đánh giá hiệu suất**: Sử dụng các chỉ số như thời gian thực thi, tài nguyên tiêu thụ, số lần tái tính toán và mức độ cải thiện so với cấu hình mặc định để đo lường hiệu quả của hệ thống.

NỘI DUNG VÀ PHƯƠNG PHÁP

- **Nội dung 1**: Nghiên cứu cách thức hoạt động của Apache Spark và các cấu hình quan trọng.
    - **Mục tiêu**: Tìm hiểu các tham số quan trọng ảnh hưởng đến hiệu suất của Spark như số lượng executors, cores, memory allocation, shuffle partitions,... để có thể tối ưu hóa bằng LLM.
    - **Phương pháp**: Nghiên cứu tài liệu chính thức của Apache Spark, phân tích các nghiên cứu liên quan, đồng thời thực hiện nhiều thử nghiệm để đánh giá tác động của các tham số.

- **Nội dung 2**: Ứng dụng LLM trong tối ưu cấu hình Spark.
    - **Mục tiêu**: Từ dữ liệu lịch sử, sử dụng LLM để dự đoán cấu hình phần cứng cho Spark phù hợp nhất.
    - **Phương pháp**:
        - Sử dụng LLM để phân tích dữ liệu đầu vào, học từ các cấu hình trước đó để đề xuất cấu hình tối ưu.
        - So sánh kết quả với các phương pháp tối ưu truyền thống như heuristic-based tuning hoặc grid search.
- **Nội dung 3**: Triển khai hệ thống trên dịch vụ đám mây.
    - **Mục tiêu**: Xây dựng hệ thống chạy trên đám mây, có khả năng mở rộng và tương tác dễ dàng với người dùng để cung cấp khuyến nghị cấu hình Spark.
    - **Phương pháp**:
        - Sử dụng Amazon EC2 để triển khai môi trường Spark với các cấu hình linh hoạt.
        - Xây dựng API hoặc giao diện web để người dùng có thể nhập thông tin và nhận khuyến nghị từ LLM.
        - Đảm bảo hệ thống có khả năng chịu tải cao, phục hồi sau lỗi và bảo mật dữ liệu.
- **Nội dung 4**: Đánh giá hiệu suất của hệ thống.
    - **Mục tiêu**: Đánh giá hiệu quả của hệ thống trong việc giảm lãng phí tài nguyên và tối ưu thời gian xử lý so với cấu hình mặc định hoặc heuristic-based tuning.
    - **Phương pháp**:
        - Sử dụng các chỉ số như thời gian thực thi trung bình, mức giảm lãng phí tài nguyên, số lần tái tính toán và độ chính xác của dự đoán so với cấu hình tối ưu thực tế.
        - So sánh kết quả của hệ thống với các phương pháp truyền thống để

xác định mức độ cải thiện.

KẾT QUẢ MONG ĐỢI

- Hệ thống có thể đưa ra khuyến nghị cấu hình Spark chính xác dựa trên dữ liệu lịch sử, giúp giảm thời gian xử lý và tối ưu hóa tài nguyên.
- So sánh với các phương pháp tối ưu truyền thống cho thấy LLM có khả năng cải thiện hiệu suất đáng kể.
- Hệ thống triển khai trên đám mây, cho phép người dùng tương tác dễ dàng và áp dụng vào thực tế.

TÀI LIỆU THAM KHẢO

[1] J. Doe and A. Smith, "Auto-Tuning Apache Spark Parameters for Processing Large Datasets," KTH Royal Institute of Technology, 2024. Available: https://kth.diva-portal.org/smash/record.jsf?pid=diva2%3A1798104&dswid=-8982

[2] M. Johnson and L. Wang, "AI-Enhanced Compute Resource Management for Apache Spark," International Journal of Future Modern Research (IJFMR), vol. 6, no. 3, pp. 45–56, 2024. Available: https://www.ijfmr.com/research-paper.php?id=33716

[3] S. Lee and K. Patel, "Reinforcement Learning for Automatic Parameter Tuning in Apache Spark: A Q-Learning Approach," in Proceedings of IEEE International Conference on Big Data (IEEE BigData), 2024. Available: https://ieeexplore.ieee.org/document/10665567

[4] G. Wang, J. Xu, and B. He, "Towards Automatic Tuning of Apache Spark Configuration," in Proceedings of the ACM Symposium on Cloud Computing (SoCC), 2018. Available: https://www.researchgate.net/publication/327812093_Towards_Automatic_Tuning_of_Apache_Spark_Configuration

[5] Y. Zhang and H. Liu, "A Novel Reinforcement Learning Approach for Spark Configuration Optimization," Sensors, vol. 22, no. 15, p. 5930, 2022. Available: https://www.mdpi.com/1424-8220/22/15/5930