APACHE SPARK

Dữ liệu lớn

ThS. Nguyễn Hồ Duy Trí
trinhd@uit.edu.vn

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA HỆ THỐNG THÔNG TIN

2020

NỘI DUNG

Mở đầu
Xử lý dữ liệu lớn

3

Lambda
Kiến trúc hệ thống dữ liệu lớn

λ

9

Kappa
Kiến trúc hệ thống dữ liệu lớn

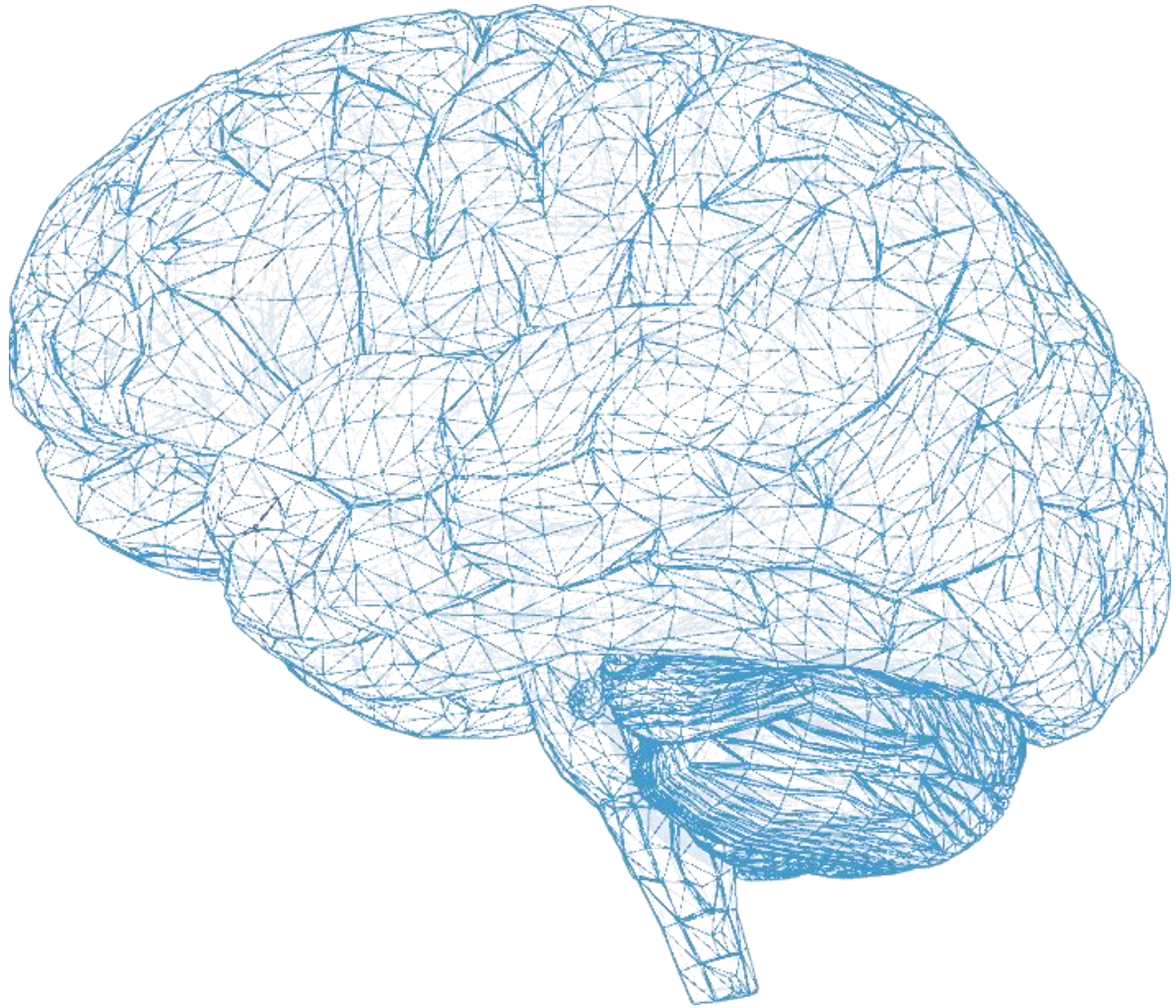
κ

22

Q & A

Mở đầu

Xử lý dữ liệu lớn



Các tổ chức xử lý dữ liệu lớn với tính chất khác nhau

- Về khối lượng: có thể là vài trăm GB hoặc vài trăm TB
- Về tốc độ: dữ liệu có thể đến nhanh, liên tục và cũng có thể chậm hơn nhưng với mỗi lần đều có khối lượng rất lớn.
- Về ý nghĩa: giá trị trích xuất ra được từ dữ liệu tùy thuộc vào từng tổ chức
- Về phương pháp xử lý: có thể áp dụng những thuật toán xử lý từ đơn giản đến phức tạp tùy theo bài toán, tổ chức
- Về cách thức thu thập và lưu trữ: phương tiện thu thập ngày càng đa dạng trong khi chi phí lưu trữ ngày càng giảm đáng kể
- Phụ thuộc vào khả năng của người xử lý và các công cụ của họ

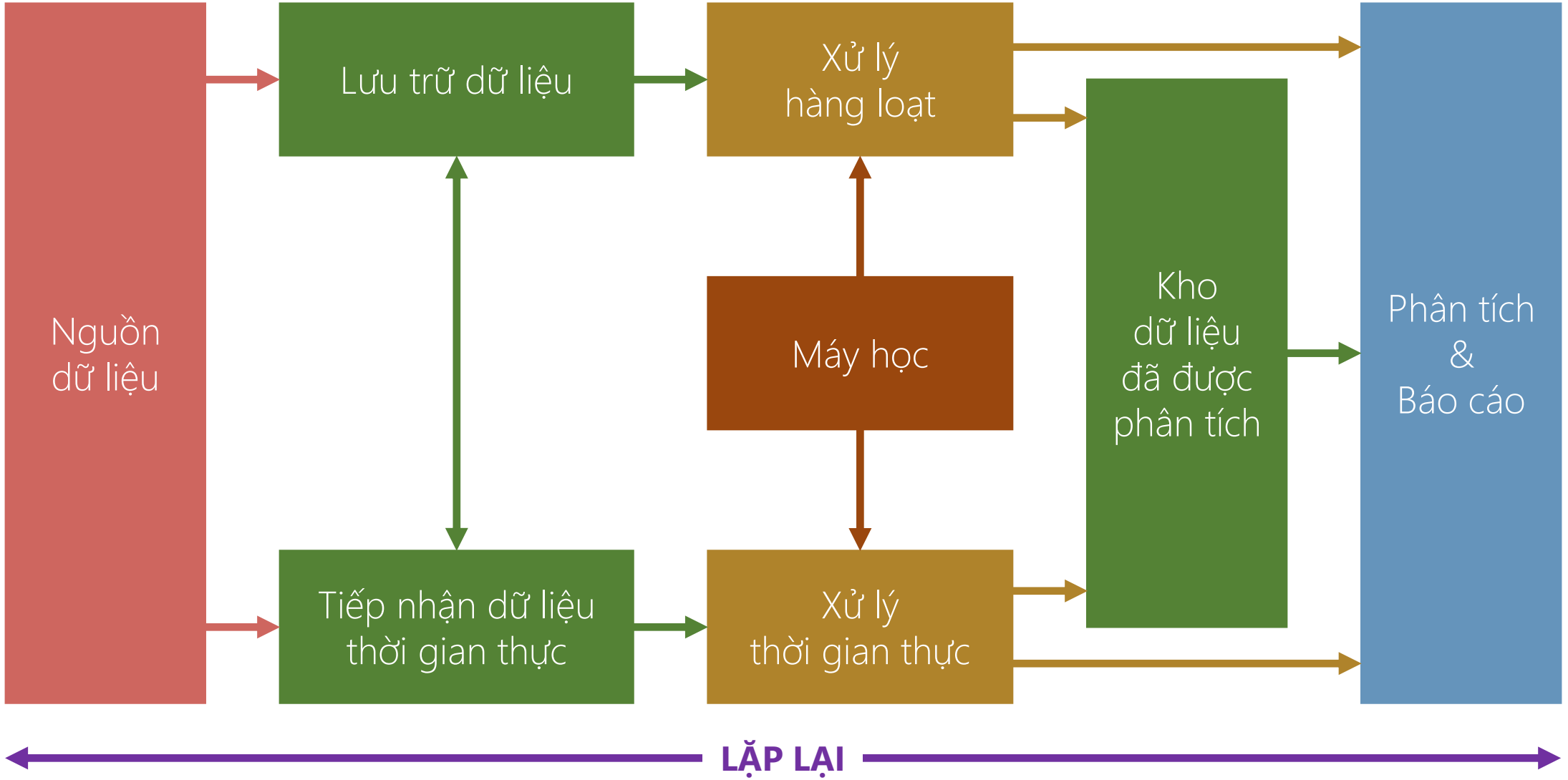
Các bài toán xử lý dữ liệu lớn thường thuộc một hoặc nhiều loại sau:

- Xử lý hàng loạt (batch processing) các nguồn dữ liệu lớn ở quy mô từng phần hoặc toàn bộ.
- Xử lý thời gian thực (real-time processing) các dữ liệu lớn đang chuyển động.
- Khám phá tương tác trong dữ liệu lớn.
- Phân tích dự đoán và học máy.

Sử dụng hệ thống xử lý dữ liệu lớn khi cần:

- Lưu trữ và xử lý dữ liệu với khối lượng quá lớn đối với cơ sở dữ liệu truyền thống.
- Chuyển đổi dữ liệu phi cấu trúc để phân tích và báo cáo.
- Thu thập, xử lý và phân tích các luồng dữ liệu không bị ràng buộc theo thời gian thực hoặc với độ trễ thấp.

Các thành phần trong hệ thống xử lý dữ liệu lớn



Khi làm việc với các tập dữ liệu rất lớn, có thể mất nhiều thời gian để chạy các truy vấn mà người dùng cần. Những truy vấn này không thể thực hiện theo thời gian thực và thường yêu cầu các thuật toán như MapReduce thực hiện song song trên toàn bộ tập dữ liệu.

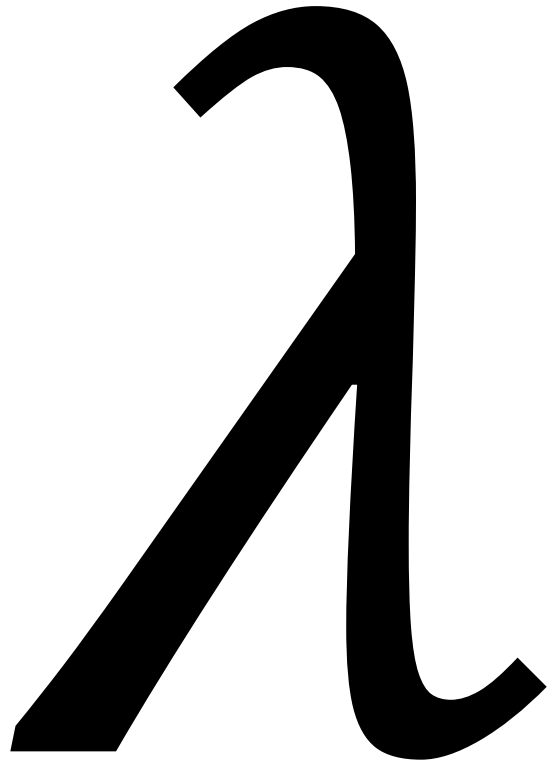
Kết quả sau đó được lưu trữ riêng biệt với dữ liệu thô và được sử dụng để truy vấn.

⇨ Hạn chế: độ trễ lớn nếu quá trình xử lý mất vài giờ, một truy vấn có thể trả về kết quả cũ trước đó vài giờ.

⇨ Lý tưởng nhất: nhận được kết quả theo thời gian thực (có thể làm độ chính xác thấp hơn) và kết hợp những kết quả này với kết quả từ phân tích hàng loạt.

Lambda

Kiến trúc hệ thống dữ liệu lớn



Kiến trúc Lambda được đề xuất bởi Nathan Marz.

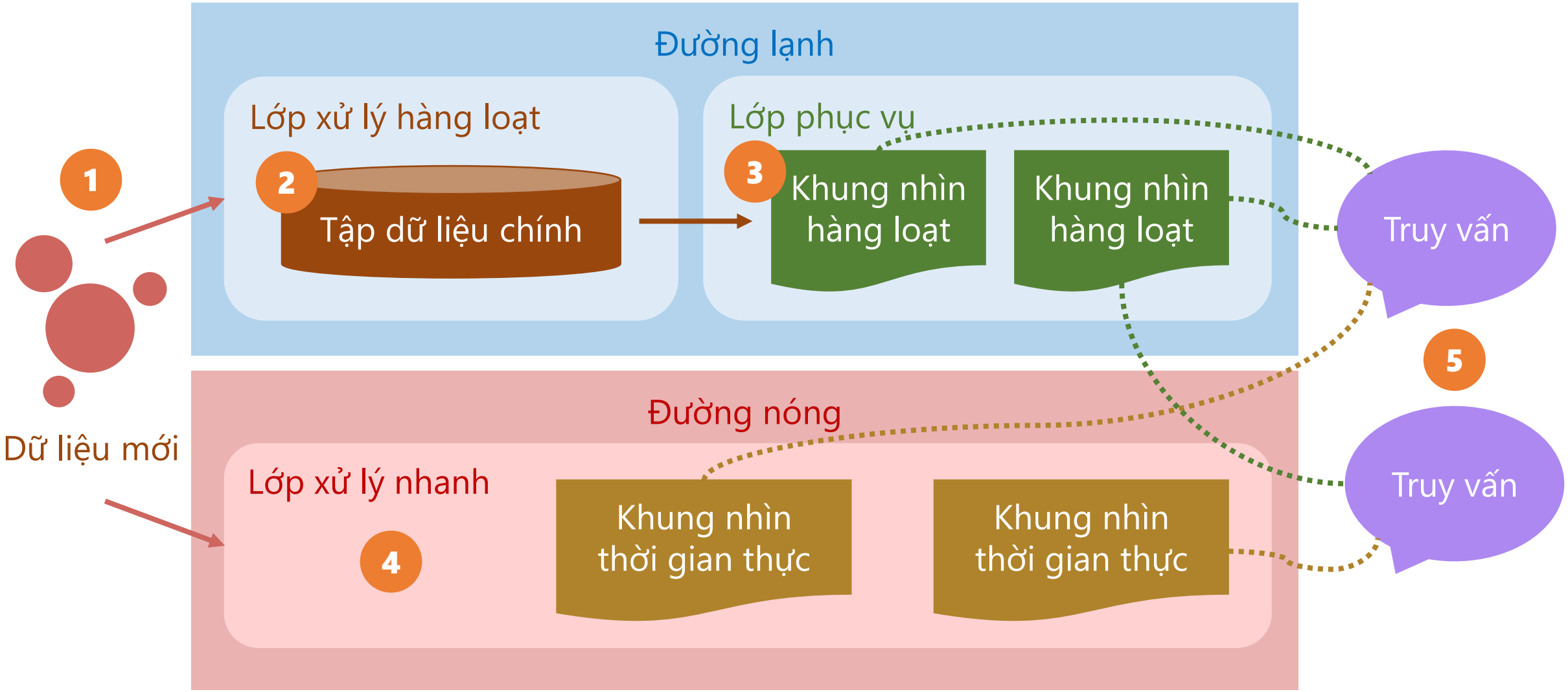
Kiến trúc Lambda (Lambda Architecture) là một kiến trúc xử lý dữ liệu chung, có thể mở rộng và chịu được lỗi.

Kiến trúc Lambda đáp ứng nhu cầu về một hệ thống mạnh mẽ có khả năng chịu lỗi, cả lỗi phần cứng và lỗi của con người, có khả năng phục vụ khối lượng công việc lớn và nhiều trường hợp cụ thể. Trong đó yêu cầu cập nhật và đọc phải có độ trễ thấp. Hệ thống sẽ có khả năng mở rộng tuyến tính và thích hợp để mở rộng hơn là nâng cấp.

Mô hình kiến trúc

Trong kiến trúc Lambda, tất cả dữ liệu vào hệ thống đi qua hai con đường sau:

- Lớp xử lý hàng loạt (batch layer) thuộc đường lạnh (cold path) lưu trữ tất cả dữ liệu đến ở dạng thô và thực hiện xử lý hàng loạt. Kết quả của quá trình xử lý được lưu trữ dưới dạng khung nhìn hàng loạt (batch view).
- Lớp xử lý nhanh (speed layer) thuộc đường nóng (hot path) phân tích dữ liệu theo thời gian thực. Lớp này được thiết kế để có độ trễ thấp nhưng độ chính xác có thể giảm đi.



Kiến trúc Lambda

Nguyên lý hoạt động

1. Tất cả các dữ liệu mới tiếp nhận đều được đưa vào cả Lớp xử lý hàng loạt và Lớp xử lý nhanh
2. Lớp xử lý hàng loạt thực hiện hai chức năng: lưu trữ dữ liệu và tiến hành xử lý dữ liệu với đầu ra là các Khung nhìn hàng loạt

3. Lớp phục vụ (serving layer) lập chỉ mục cho các Khung nhìn hàng loạt để có thể truy cập các kết quả này một cách nhanh chóng
4. Lớp xử lý nhanh chỉ xử lý các thông tin mới nhất trong thời gian thực để tạo ra các Khung nhìn thời gian thực (real-time view)
5. Hệ thống trả về kết quả cho các truy vấn của người dùng bằng cách kết hợp cả Khung nhìn hàng loạt và Khung nhìn thời gian thực.

Các lớp trong mô hình

Lớp xử lý hàng loạt đóng vai trò như một kho chứa dữ liệu để lưu trữ toàn bộ dữ liệu thu thập được và xử lý dữ liệu này hàng loạt.

Việc áp dụng xử lý hàng loạt đem lại ưu điểm cho kiến trúc Lambda vì dữ liệu thu về có thể bị trùng lặp hoặc có thể chứa nhiều thông tin rác do đó cần phải có các bước xử lý và làm sạch dữ liệu. Bên cạnh đó dữ liệu cũng có thể đến trễ do sự cố đường truyền hoặc được thu thập muộn hơn so với thời gian đăng tải.

Ngoài ra, kết quả mỗi lần xử lý sẽ được cập nhật thường xuyên, từ đó giúp nâng cao độ chính xác và sửa lỗi kết quả của việc xử lý theo thời gian thực ở các lần chạy trước.

Lớp xử lý nhanh có nhiệm vụ xử lý thông tin theo thời gian thực, đóng vai trò hỗ trợ cho Lớp xử lý hàng loạt. Do Lớp xử lý hàng loạt có độ trễ lớn và không đáp ứng được việc đưa ra kết quả xử lý với dữ liệu mới nhận về. Tuy nhiên kết quả của Lớp xử lý nhanh thường không có độ chính xác như ở Lớp xử lý hàng loạt do hạn chế về thời gian thao tác. Có thể cài đặt bằng các công cụ như Apache Storm, Apache Spark hoặc Apache Flume...

Lớp phục vụ có vai trò lưu trữ các kết quả đầu ra của Lớp xử lý hàng loạt và Lớp xử lý nhanh. Có thể cài đặt bằng một số công cụ như Apache Cassandra, MongoDB hoặc ElasticSearch...

Áp dụng

Có thể áp dụng kiến trúc này khi có các yêu cầu sau:

- Các truy vấn của người dùng đòi hỏi phải sử dụng dữ liệu lưu trữ bất biến.
- Bắt buộc phải phản hồi nhanh và có khả năng xử lý các thay đổi khác nhau khi xuất hiện dữ liệu mới.
- Dữ liệu được lưu trữ sẽ không bị xóa, cho phép bổ sung các bản cập nhật và dữ liệu mới vào cơ sở dữ liệu.

Kiến trúc Lambda có thể được coi là kiến trúc xử lý dữ liệu gần thời gian thực.

Ưu điểm

- Lớp xử lý hàng loạt của kiến trúc Lambda quản lý dữ liệu lịch sử với bộ lưu trữ phân tán đảm bảo khả năng xảy ra lỗi thấp ngay cả khi hệ thống gặp sự cố.
- Cân bằng tốt giữa yêu cầu tốc độ và độ tin cậy.
- Kiến trúc có khả năng chịu lỗi và có thể mở rộng dễ dàng để xử lý dữ liệu.

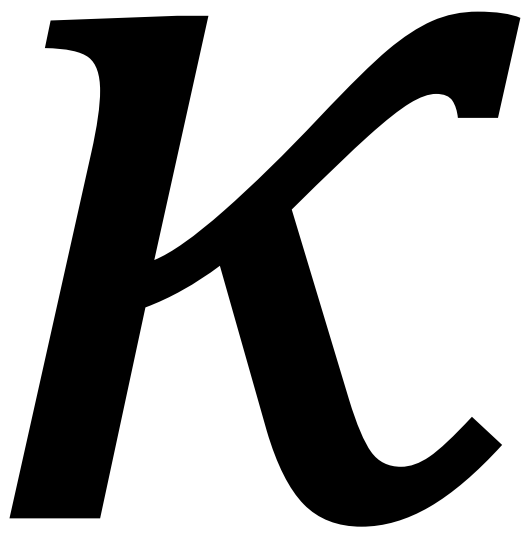
Khuyết điểm

- Là kiến trúc chung nên đối với từng trường hợp cụ thể sẽ phải xem xét loại bỏ một số thành phần.
- Do kết hợp từ nhiều thành phần khác nhau nên chi phí lập trình, bảo trì cho toàn bộ hệ thống sẽ tốn kém, khó triển khai.
- Do không có dữ liệu nào bị loại bỏ nên việc mở rộng theo thời gian sẽ gây tốn kém chi phí.
- Chạy lại mọi chu trình xử lý hàng loạt không có lợi trong một số trường hợp nhất định.

- Dữ liệu được mô hình hóa bằng kiến trúc Lambda rất khó tích hợp hoặc sắp xếp lại.
- Logic xử lý xuất hiện ở hai nơi khác nhau - các đường lạnh và nóng - sử dụng các khung nhìn khác nhau. Điều này dẫn đến logic tính toán trùng lặp và sự phức tạp trong việc quản lý kiến trúc, đồng bộ xử lý cho cả hai đường.

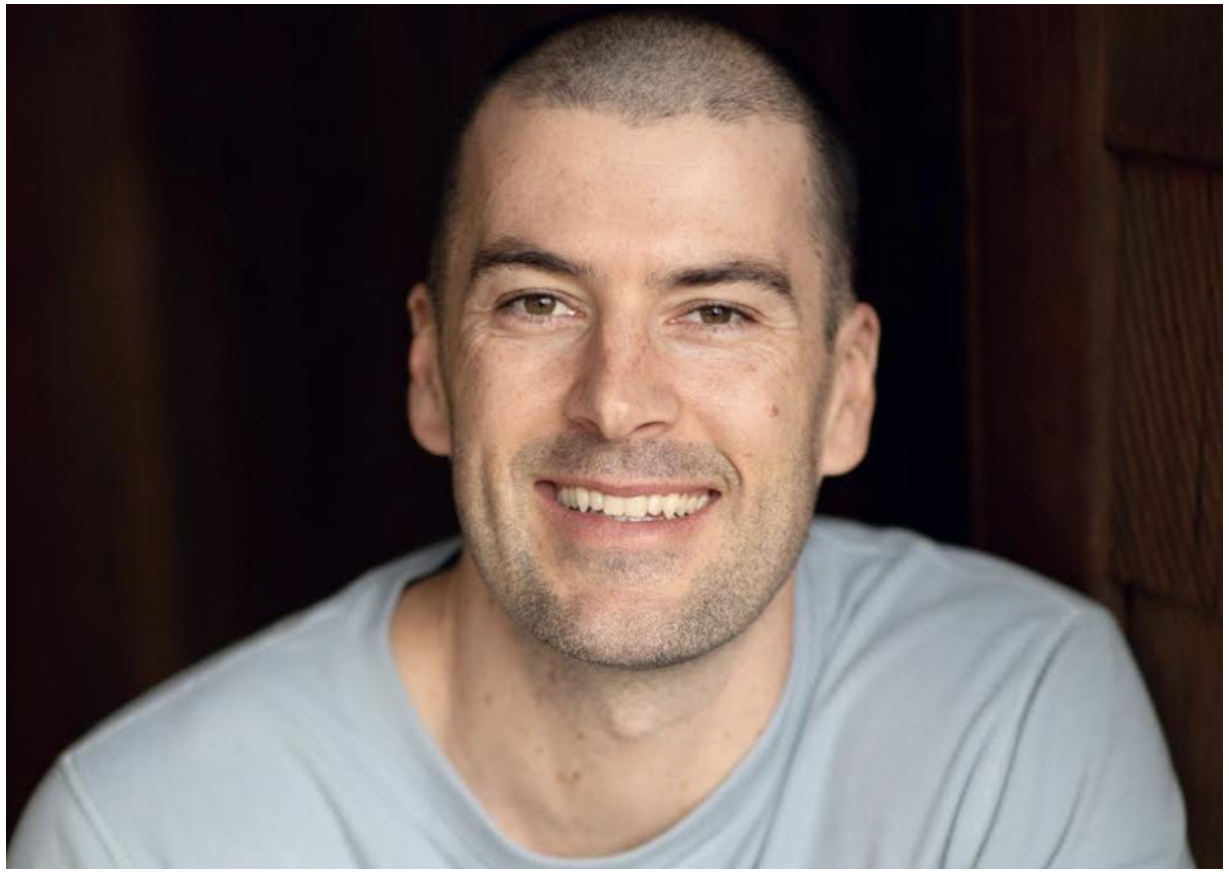
Kappa

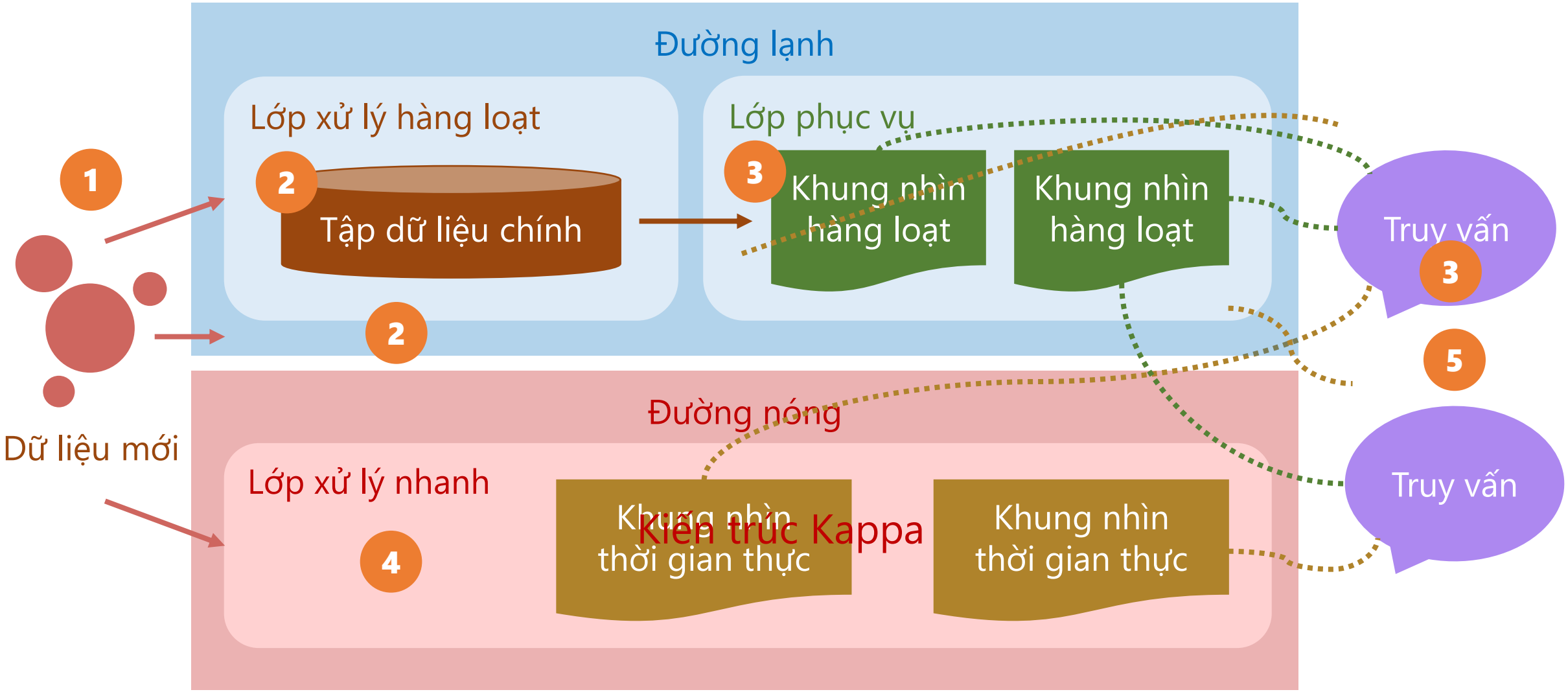
Kiến trúc hệ thống dữ liệu lớn



Kiến trúc Kappa được Jay Kreps đề xuất vào năm 2014 từ sự *đơn giản hóa* kiến trúc Lambda.

Nó có các mục tiêu cơ bản giống như kiến trúc Lambda, nhưng có một điểm khác biệt quan trọng: *Tất cả dữ liệu đều đi qua một đường duy nhất, sử dụng Lớp xử lý nhanh*.





Kiến trúc Lambda

Đặc điểm

- Có một số điểm tương đồng với Lớp xử lý hàng loạt của kiến trúc Lambda
  - Dữ liệu sự kiện là bất biến
  - Tất cả dữ liệu được thu thập, thay vì chỉ một phần.
- Dữ liệu được nhập dưới dạng một chuỗi sự kiện đi vào quá trình tính toán có khả năng chịu lỗi và phân tán. Các sự kiện này được sắp xếp theo thứ tự và trạng thái hiện tại của một sự kiện chỉ được thay đổi khi một sự kiện mới được thêm vào.

- Tương tự như Lớp xử lý nhanh của kiến trúc Lambda, tất cả quá trình xử lý sự kiện được thực hiện trên chuỗi dữ liệu đầu vào và tồn tại ở dạng Khung nhìn thời gian thực.
- Nếu cần tính toán lại toàn bộ tập dữ liệu (tương đương với những gì Lớp xử lý hàng loạt thực hiện trong kiến trúc Lambda), chỉ cần đưa vào chuỗi dữ liệu từ ban đầu.
    - Thường sử dụng các kỹ thuật song song hóa để hoàn thành việc tính toán kịp thời, nhanh chóng.

Lambda vs. Kappa

Kiến trúc Lambda là một mô hình triển khai kiến trúc phần mềm trong đó:

- Dữ liệu đến được cung cấp song song cho cả hai lớp: Lớp xử lý hàng loạt và Lớp xử lý nhanh.
- Lớp xử lý hàng loạt đưa dữ liệu vào kho chứa dữ liệu, áp dụng logic tính toán và phân phối kết quả đến Lớp phục vụ để người dùng sử dụng.
- Lớp xử lý nhanh sử dụng thông tin chi tiết trước đó có được trong Lớp xử lý hàng loạt để xử lý dữ liệu mới đến.

- Điều quan trọng cần lưu ý là kiến trúc Lambda yêu cầu một Lớp xử lý hàng loạt riêng biệt cùng với Lớp xử lý nhanh trước khi dữ liệu được phân phối đến Lớp phục vụ.

Kiến trúc Kappa không phải mô hình thay thế cho kiến trúc Lambda.

- Là một cách tiếp cận chuyên biệt.
- Có thể được sử dụng trong các kiến trúc mà Lớp xử lý hàng loạt không cần thiết, các yêu cầu có thể thay thế bằng Lớp xử lý nhanh.
- Đang được sử dụng trong các mô hình triển khai ưu tiên luồng dữ liệu trực tuyến, nơi có yêu cầu về độ trễ đầu cuối là rất nghiêm ngặt.

Áp dụng

Hầu hết các trường hợp áp dụng đều có nhu cầu truy cập dữ liệu có độ trễ rất thấp:

- Báo cáo và điều khiển thời gian thực
- Xử lý các quy tắc và đưa ra cảnh báo thời gian thực
- Vận hành mô hình học máy thời gian thực
- Thực hiện các tính năng thông minh theo thời gian thực

Ưu điểm

- Kiến trúc Kappa có thể được sử dụng để phát triển hệ thống dữ liệu dành cho việc học trực tuyến.
- Chỉ cần xử lý lại khi chương trình/bài toán thay đổi.
- Có thể được triển khai với bộ nhớ cố định.
- Có thể được sử dụng cho các hệ thống có khả năng mở rộng theo chiều ngang.

- Yêu cầu ít tài nguyên hơn vì các thuật toán máy học đang được thực hiện ở mức độ thời gian thực.
- Việc phát triển, triển khai, gỡ lỗi và bảo trì đơn giản hơn.
- Kiến trúc được đơn giản và không cần các yêu cầu đồng bộ giữa các thành phần xử lý.

Khuyết điểm

- Kiến trúc Kappa không có khả năng quản lý các ứng dụng tính toán chuyên sâu đối với các trường hợp sử dụng máy học trên quy mô lớn.
- Do không có Lớp xử lý hàng loạt, các tác vụ xử lý hàng loạt hầu như sẽ bị ảnh hưởng.

Simplicity *is the ultimate sophistication*

*- **Leonardo Da Vinci***

TÀI LIỆU THAM KHẢO

1. Marz, Nathan and Warren, James. 2015. **Big Data: Principles and Best Practices of Scalable Realtime Data Systems (1st ed.)**. Manning Publications Co., USA.
2. **Lambda Architecture official website**, URL: http://lambda-architecture.net/
3. **Big data architectures**, Microsoft Docs, URL: https://docs.microsoft.com/en-us/azure/architecture/data-guide/big-data/

Q & A