Kiến trúc Lambda : Thiết kế đơn giản, có khả năng phục hồi, duy trì và mở rộng các giải pháp Big Data

Một hệ thống dữ liệu được thiết kế để lưu trữ dữ liệu. Nó cũng được thiết kế để lấy được thông tin từ các dữ liệu được lưu trữ. Các thông tin mong muốn có thể là dữ liệu của chính nó, hoặc các dữ liệu được tính toán từ dữ liệu được lưu trữ. Các hệ thống dữ liệu rất quan trọng và thường tồn tại lâu hơn các ứng dụng được xây dựng xung quanh chúng. Các hệ thống không thay đổi thường xuyên theo những ứng dụng xung quanh. Vì vậy thiết kế các hệ thống dài hạn rất quan trọng .

Gần đây các hệ thống dữ liệu là chủ đề được chú trọng nhiều hơn và đang ngày càng trở nên phức tạp hơn. Kiến trúc Lambda đề xuất đơn giản hơn, mô hình được thiết kế để làm giảm sự phức tạp trong khi vẫn có khả năng lưu trữ và xử lý lượng lớn dữ liệu một cách hiệu quả . Kiến trúc Lambda ban đầu được trình bày bởi Nathan Marz, người nổi tiếng trong cộng đồng big data trong dự án Storm của ông.

### Những khó khăn của hệ thống dữ liệu hiện tại ?

**Hệ thống hiện tại là không có khả năng phục hồi** : Hệ thống cơ sở dữ liệu hiện tại (quan hệ và NoSQL) không được thiết kế để khôi phục lại. Hầu hết các hệ thống dữ liệu hiện tại hỗ trợ create, read , update và delete (CRUD). Trong số này, update và delete có khả năng lớn để gây ra hỏng dữ liệu. Với các hệ thống dữ liệu hiện tại, điều đó quá dễ dàng để xóa một bảng toàn bộ dữ liệu khi chúng ta có ý định chỉ đơn giản là xóa một hàng duy nhất. Một lỗi phần mềm hoặc phần cứng đều cũng dễ dàng làm hỏng dữ liệu.

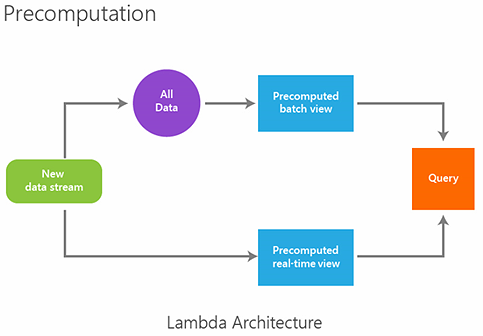
**Sự kết hợp của các truy vấn và dữ liệu** : Với các hệ thống cơ sở dữ liệu hiện tại, xử lý truy vấn liên quan chặt chẽ đến dữ liệu lưu trữ. Trong khi lưu trữ dữ liệu được thực hiện một cách đơn giản, còn truy vấn thường sử dụng với sự phức tạp hóa. Điều này dẫn tới sự khó khăn trong việc cân bằng phải thực hiện trong mỗi hệ thống – nên cần một sự tối ưu hóa cho hiệu suất truy vấn hoặc lưu trữ dữ liệu.

**Kết quả của việc mở rộng – định luật CAP** : Định luật CAP nói rằng hệ thống máy tính không thể phân bố để đáp ứng đồng thời cả 3 sự bảo đảm sau :

* Tính nhất quán : tất cả các node xem cùng dữ liệu trong cùng thời gian
* Tính sẵn sàng : đảm bảo rằng mọi yêu cầu đều nhận được một phản hồi về việc yêu cầu đó có thành công hay không.
* Phân vùng sự cho phép : Hệ thống tiếp tục hoạt động mặc dù có thể mất gói tin nào đó hoặc toàn bộ hệ thống bị lỗi.

### Kiến trúc Lambda

Kiến trúc Lam được thiết kế để thực hiện tốt hơn những bất cập mà chúng ta đã chỉ ra. Kiến trúc Lambda chỉ rõ dữ liệu lưu trữ đó là bất biến. Về khía cạnh của CRUP dữ liệu lưu trữ bất biến cơ bản loại bỏ update và delete, chỉ cho phép read và create các bản ghi dữ liệu.



Hình 1 : Kiến trúc Lambda

**Truy vấn (query)** : được xem là một chức năng của tất cả các dữ liệu . Một dữ liệu lưu trữ bất biến ghi lại các sự việc có thể cung cấp các câu truy vấn phản hồi giống như một cơ sở dữ liệu quan hệ, chỉ là các chi tiết xử lý khác nhau . Nói chung kết quả truy vấn có thể xem như một số chức năng của tất cả dữ liệu được lưu trữ trong hệ thống. *Query results => function (all data stored)*

**Tầng bacth** : có chứa các giá trị tính toán trước được sinh ra bởi từng đợt xử lý .Các ứng dụng cần khả năng truy cập dữ liệu một cách nhanh chóng. Chúng không thể đợi từng đợt xử lý hoàn thành. Với tầng batch chúng ta có thể phục vụ hầu hết các nhu cầu ứng dụng.

Tuy nhiên từng đợt xử lý sinh ra các view này cần một khoảng thời gian để chạy. Trong khoảng thời gian đó dữ diệu bổ sung có thể được thêm vào. Để tính toán cho những dữ liệu này chúng ta cần một tầng song song có thể xử lý dữ liệu bổ sung khi nó đi vào . Kiến trúc Lambda cung cấp cho khía cạnh này và gọi đó là tầng real-time.

**Tầng real-time**: được thiết kế để tính toán các kết quả truy vấn trên cùng của một luồng dữ liệu đến. Các kết quả , một khi tính toán, nên được lưu trữ trong phương pháp nào đó mà có thể được truy vấn bởi các ứng dụng. Cũng như tầng batch, tầng real-time cũng lưu trữ các kết quả khi chúng được tính toán vào một view.

Kết quả cuối cùng : Ứng dụng sẽ truy vấn cả real-time và batch của các view , cuối cùng tập hợp lại các kết quả.