**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN   
KHOA CÔNG NGHỆ PHẦN MỀM**

**Trần Hoàng Ân – 16520021**

**KHÓA LUẬN TỐT NGHIỆP**

**HỆ THỐNG DỮ LIỆU LỚN**

**HỖ TRỢ NHẬN DIỆN SỰ BUỒN NGỦ**

**Big Data System supports Drowsiness Detection as a Service**

**KỸ SƯ NGÀNH KỸ THUẬT PHẦN MỀM**

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN   
KHOA CÔNG NGHỆ PHẦN MỀM**

**Trần Hoàng Ân – 16520021**

**KHÓA LUẬN TỐT NGHIỆP**

**HỆ THỐNG DỮ LIỆU LỚN**

**HỖ TRỢ NHẬN DIỆN SỰ BUỒN NGỦ**

**Big Data System supports Drowsiness Detection as a Service**

**KỸ SƯ NGÀNH KỸ THUẬT PHẦN MỀM**

ĐHQG TP. HỒ CHÍ MINH. **CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**TRƯỜNG ĐẠI HỌC Độc Lập – Tự Do – Hạnh Phúc**

**CÔNG NGHỆ THÔNG TIN**

# ĐỀ CƯƠNG CHI TIẾT

|  |
| --- |
| **TÊN ĐỀ TÀI: HỆ THỐNG DỮ LIỆU LỚNHỖ TRỢ NHẬN DIỆN KHUÔN MẶT BUỒN NGỦ** |
| **Cán bộ hướng dẫn: TS. Đỗ Thị Thanh Tuyền** |
| **Thời gian thực hiện:** Từ ngày 07/09/2020 đến ngày 27/12/2020 |
| **Sinh viên thực hiện:**  **Trần Hoàng Ân - 16520021** |
| **Nội dung đề tài:**  ***Mục tiêu:***   * Tìm hiểu các ưu điểm, nhược điểm của các hệ thống dữ liệu lớn hiện tại. * Nghiên cứu các công cụ hỗ trợ làm data warehouse, data processing, ETL hiện tại có trên thị trường. * Nghiên cứu các kiến trúc hệ thống dữ liệu lớn hiện tại. * Thiết kế hệ thống hỗ trợ dữ liệu lớn phù hợp với yêu cầu phục vụ hệ thống máy học. |

|  |
| --- |
| ***Phạm vi:***  Các hệ thống dữ liệu lớn đám mây (cloud), các hệ thống dữ liệu lớn tại chỗ (on- premise) của các công ty vừa và nhỏ, các nền tảng mã nguồn mỡ hỗ trợ dữ liệu lớn.  ***Đối tượng nghiên cứu:***   * Hadoop ecosystem. * Các nền tảng mã nguồn mở hỗ trợ dữ liệu lớn.   ***Phương pháp thực hiện:***  Thực hiện đề tài theo các bước:   * Phân tích đề tài. * Nghiên cứu các dịch vụ / nền tảng đang hiện hữu, kết hợp với hệ thống đang được triển khai tại công ty đã, đang công tác là VNG, Sendo. * Thiết kế cơ bản các yêu cầu. * Nghiên cứu công nghệ để áp dụng. * Phát triển sản phẩm qua các giai đoạn. * Kiểm thử và hoàn thiện.   ***Kết quả mong đợi:***   * Làm rõ định nghĩa và xác định đúng Dữ liệu lớn là gì? Các khái niệm chuyên ngành cần biết đến. * Đem đến cái hình từ tổng quát đến chi tiết các hệ thống dữ liệu lớn hiện có |

trên thị trường.

* Phân tích các phần cốt lõi để tạo nên một hệ thống dữ liệu lớn tại chỗ.
* Thiết kế chi tiết hệ thống dữ liệu lớn (mức độ có thể triển khai) bao gồm: Apache Hadoop, Apache Spark, Cassandra, Jupyter notebook, Airflow.
* Thiết kế thành công hệ thống dữ liệu lớn làm nền tảng hỗ trợ cho máy học

|  |  |  |
| --- | --- | --- |
| **Kế hoạch thực hiện:** | | |
| **STT** | **Công việc** | **Thời gian thực hiện** |
| **Phase 1: Tìm hiểu các hệ thống Big Data hiện có trên thị trường, xây dựng hệ thống** | | **12 tuần**  **(07/09/2020 – 02/11/2020)** |
| **1** | **Sprint 1: Tìm hiểu các định nghĩa về Big data** |  |
| 1.1 | * Big data là gì? * Các thuộc tính của Big data? * Big data với các CSDL hiện tại.. * Chi phí, bảo mật, tính mở rộng,.. * Các định nghĩa về Batch processing,   cluster computing, data storage,  ochestration, data mining,HDFS, data lake  và data warehouse.   * Hadoop eco system. * Kiến trúc của 1 big data system * Lựa chọn vấn đề để giải quyết | Tuần 1 - 2  (07/09/2020 - 20/09/2020) |
| **2** | **Sprint 2: Tìm hiểu các mô hình thực tế,**  **xác định các thành phần trong hệ thống** |  |
| 2.1 | * Tìm hiểu: Cloudera, IBM, Teradata,   Oracle,… | Tuần 3  (21/09/2020 - 27/09/2020) |
| **3** | **Sprint 3: Xây dựng prototype trên Apache**  **Hadoop Ecosystem** |  |
| 3.1 | * Cài đặt môi trường trên máy ảo VM * Deploy Hadoop, Flume, sqoop, Pig,   Airflow, Kafka.   * Deploy database Cassandra, Mongodb,… * Deploy Analysis tool: D3.JS, Dygaphs, … | Tuần 4 - 8  (28/09/202 – 01/11/2020) |
| **4** | **Sprint 4: Xây dựng thuật toán Máy học** |  |

|  |  |  |
| --- | --- | --- |
| 4.1 | * **Xây dựng thuật toán máy học** | Tuần 9 – 12  (02/11/2020 – 29/11/2020) |
| **Phase 2: Kết hợp hệ thống Data & Machine learning** | | **4 tuần**  **(30/11/2020 – 27/12/2020)** |
| **5** | **Sprint 5: Kết hợp Data warehouse và ML system** |  |
| 5.1 | * Đổ data vào Data warehouse * Kết nối ML system * Chạy train model | Tuần 13 - 14  (30/11/2020 - 13/12/2020) |
| **6** | **Sprint 6: Kiểm thử** |  |
| 6.1 | * Kiểm thử * Đánh giá | Tuần 15-16  (14/12/2020 - 27/12/2020) |

|  |  |
| --- | --- |
| **Xác nhận của CBHD**  (Ký và ghi rõ họ tên) | **TP.HCM, ngày 12 tháng 12 năm 2020**  (Ký tên và ghi rõ họ tên)  **Trần Hoàng Ân** |

# MỤC LỤC

Contents

[ĐỀ CƯƠNG CHI TIẾT 4](#_Toc59480897)

[MỤC LỤC 16](#_Toc59480898)

[DANH MỤC HÌNH VẼ 22](#_Toc59480899)

[DANH MỤC BẢNG 24](#_Toc59480900)

[DANH MỤC TỪ VIẾT TẮT 25](#_Toc59480901)

[TÓM TẮT ĐỒ ÁN 24](#_Toc59480902)

[MỞ ĐẦU 25](#_Toc59480903)

[Lý do chọn đề tài: 25](#_Toc59480904)

[Chương 1. TỔNG QUAN ĐỀ TÀI 28](#_Toc59480905)

[1.1 Mục tiêu và phạm vi 28](#_Toc59480906)

[**1.1.1** **Mục tiêu** 28](#_Toc59480907)

[**1.1.2** **Phạm vi** 28](#_Toc59480908)

[1.2 Nội dung thực hiện 28](#_Toc59480909)

[Chương 2. DỮ LIỆU LỚN LÀ GÌ, CÁC KHÁI NIỆM CHUYÊN NGÀNH CẦN LÀM RÕ 29](#_Toc59480910)

[2.1. Khảo sát thực trạng 29](#_Toc59480911)

[2.2. Dữ liệu lớn là gì? 30](#_Toc59480912)

[2.3. Đặc điểm của dữ liệu lớn 33](#_Toc59480913)

[**2.3.1. Dung lượng** 35](#_Toc59480914)

[**2.3.2. Vận tốc** 37](#_Toc59480915)

[**2.3.3.** **Đa dạng** 38](#_Toc59480916)

[**2.3.4.** **Độ chính xác** 39](#_Toc59480917)

[**2.3.5.** **Tính biến thiên** 40](#_Toc59480918)

[**2.3.6.** **Giá trị** 41](#_Toc59480919)

[2.4. Cách tiếp cận dựa trên giải pháp cho dữ liệu 42](#_Toc59480920)

[**2.4.1. Dữ liệu - tài sản quý giá nhất** 42](#_Toc59480921)

[**2.4.2. Phương pháp truyền thống để lưu trữ dữ liệu** 44](#_Toc59480922)

[**2.4.3. Điện toán cụm** 47](#_Toc59480923)

[**2.4.4.** **Giải pháp dữ liệu lớn - cơ sở hạ tầng đám mây so với tại chỗ** 48](#_Toc59480924)

[**2.4.5.** **Các thuật ngữ** 52](#_Toc59480925)

[Chương 3. HỆ SINH THÁI HADOOP – HỆ THỐNG DỮ LIỆU LỚN TẠI CHỖ 55](#_Toc59480926)

[3.1. Apache Hadoop 56](#_Toc59480927)

[**3.1.1. Hadoop Distributed File System (HDFS)** 57](#_Toc59480928)

[**3.1.2. MapReduce** 61](#_Toc59480929)

[**3.1.3. YARN (Yet Another Resource Negotiation)** 62](#_Toc59480930)

[**3.1.4. Lợi thế và khó khăn khi dùng hadoop** 64](#_Toc59480931)

[3.2. Apache Zookeeper 65](#_Toc59480932)

[**3.2.1. Kiến trúc** 66](#_Toc59480933)

[**3.2.2. Data Model** 67](#_Toc59480934)

[3.3. Apache Kafka 69](#_Toc59480935)

[**3.3.1. Kiến trúc API** 69](#_Toc59480936)

[**3.3.2. Kiến trúc kafka cluster** 70](#_Toc59480937)

[**3.3.4. Các khái niệm cơ bản** 71](#_Toc59480938)

[3.4. Apache Cassandra 73](#_Toc59480939)

[**3.4.1. Các định nghĩa cơ bản** 73](#_Toc59480940)

[**3.4.2. Nhân bản dữ liệu (Data Replication)** 74](#_Toc59480941)

[3.5. Apache Spark 76](#_Toc59480942)

[**3.5.1. Kiến trúc và thành phần của Spark** 77](#_Toc59480943)

[**3.5.2. Những ưu điểm nổi bật của Spark** 78](#_Toc59480944)

[**3.5.3. So sánh giữa Spark và Hadoop MapReduce** 79](#_Toc59480945)

[3.6. Apache Airflow 81](#_Toc59480946)

[**3.6.1. Các component của Airflow** 81](#_Toc59480947)

[**3.6.2. Kiến trúc đơn node.** 82](#_Toc59480948)

[**3.6.3. Kiến trúc đa node** 83](#_Toc59480949)

[Chương 4. TỔNG QUAN VỀ CÁC HỆ THỐNG BIG DATA TRÊN CLOUD - AMAZON 85](#_Toc59480950)

[4.1. Amazon S3 86](#_Toc59480951)

[**4.1.1. Kiến trúc và các khái niệm cơ bản** 86](#_Toc59480952)

[**4.1.2. Quy trình xử dụng Amazon S3 cơ bản** 89](#_Toc59480953)

[**4.1.3. Ưu điểm của Amazon** 89](#_Toc59480954)

[**4.1.4. Lợi ích của amazon S3** 90](#_Toc59480955)

[**4.1.5. Các trường hợp nên dùng Amazon S3** 92](#_Toc59480956)

[**4.1.6. Các công ty điển hình sử dụng AWS S3** 94](#_Toc59480957)

[**4.1.7. Các loại dữ liệu amazon có thể lưu trữ** 95](#_Toc59480958)

[4.2. Amazon EC2 95](#_Toc59480959)

[**4.2.1. Tổng quan về EC 2** 95](#_Toc59480960)

[**4.2.2. Các tính năng nổi bật EC2** 96](#_Toc59480961)

[4.3. Amazon EMR 100](#_Toc59480962)

[**4.3.1. Lợi ích** 100](#_Toc59480963)

[**4.3.2. Các Use Case** 102](#_Toc59480964)

[Chương 5. CONTAINER HOÁ 104](#_Toc59480965)

[5.1. Ảo hoá 104](#_Toc59480966)

[5.2. Hypervisor 105](#_Toc59480967)

[**5.2.1. Các siêu giám sát dựa trên phần cứng** 105](#_Toc59480968)

[**5.2.2. Các siêu giám sát dựa trên phần mềm** 105](#_Toc59480969)

[5.3. Container hóa là gì? 106](#_Toc59480970)

[5.4. Lợi ích của container 107](#_Toc59480971)

[5.5. Docker 108](#_Toc59480972)

[5.6. Quy trình làm việc của Docker 108](#_Toc59480973)

[5.7. Docker Swarm 109](#_Toc59480974)

[5.8. Kubernetes 110](#_Toc59480975)

[**5.8.1. Thành phần chính** 110](#_Toc59480976)

[Chương 6. NỀN TẢNG CHO KHOA HỌC DỮ LIỆU 113](#_Toc59480977)

[6.1 Khung phân tích dữ liệu PaaS 114](#_Toc59480978)

[6.2 Khung phân tích dữ liệu SaaS 120](#_Toc59480979)

[Chương 7. XÂY DỰNG HỆ THỐNG DỮ LIỆU LỚN HỖ TRỢ MÁY HỌC VÀ ÁP DỤNG GIẢI QUYẾT BÀI TOÁN DROWSINESS DETECTION 132](#_Toc59480980)

[7.1 Trình bày ý tưởng 132](#_Toc59480981)

[7.2. Khái quát, tổng quan hóa ý tưởng 132](#_Toc59480982)

[7.3. Các Use case của Machine Learning Toolbox 135](#_Toc59480983)

[**7.3.1. Lưu trữ dữ liệu** 135](#_Toc59480984)

[**7.3.2. Phân tích dữ liệu** 137](#_Toc59480985)

[**7.3.3. Training model máy học** 137](#_Toc59480986)

[**7.3.4. Trực quan hóa dữ liệu ( tạo báo cáo)** 138](#_Toc59480987)

[**7.3.5. Kế hoạch hóa từ process** 139](#_Toc59480988)

[7.4. Machine learning toolbox 139](#_Toc59480989)

[**7.4.1. Chi tiết hệ thống** 139](#_Toc59480990)

[**4.4.2. Mô hình chi tiết hệ thống sau khi chọn công cụ** 147](#_Toc59480991)

[7.5. Áp dụng hệ thống giải quyết bài toán nhận diện buồn ngủ 148](#_Toc59480992)

[**7.5.1. Đặt vấn đề** 148](#_Toc59480993)

[**7.5.2. Nguồn dữ liệu và các nghiên cứu ban đầu** 149](#_Toc59480994)

[**7.5.3. Bóc tách dữ liệu** 152](#_Toc59480995)

[**7.5.4. Normalize các feature** 156](#_Toc59480996)

[**7.5.5. Đánh label lại ban đầu** 157](#_Toc59480997)

[**7.5.6. Kết quả với các model phân loại cơ bản** 157](#_Toc59480998)

[**7.5.7. Độ quan trọng từng thuộc tính** 160](#_Toc59480999)

[Chương 8. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 160](#_Toc59481000)

[8.1 Kết quả đạt được 160](#_Toc59481001)

[8.2 Nhận xét 161](#_Toc59481002)

[**8.2.1. Ưu điểm** 161](#_Toc59481003)

[**8.2.2. Khuyết điểm** 161](#_Toc59481004)

[8.3 Hướng phát triển 161](#_Toc59481005)

[TÀI LIỆU THAM KHẢO 167](#_Toc59481006)

# DANH MỤC HÌNH VẼ

Hình 1 Tăng trưởng dữ liệu trong quá khứ, hiện tại và tương lai 13

Hình 2 Vận tốc mà dữ liệu được tạo ra 15

Hình 3 Sự đa dạng của dữ liệu 16

Hình 4 Mô tả sự không rõ ràng trong ví dụ của táo 16

Hình 5 Tính biến thiên của dữ liệu 17

Hình 6 Kiến trúc hệ thống kinh doanh thông minh 21

Hình 7 Minh họa của một môi trường cụm máy tính 24

Hình 8 Các loai cơ sở hạ tầng dữ liệu lớn khác nhau 26

Hình 9 Mô hình hệ sinh thái Hadoop 32

Hình 10 Mô hình kiến trúc Apache Hadoop 34

Hình 11 Kiến trúc master-slaves của HDFS (Name node – Data nodes) 35

Hình 12 Ví dụ chia block trên 1 file 36

Hình 13 Mô phỏng replication trên các data nodes. 37

Hình 14 Mô hình thực tế kiến trúc HDFS trên từng Rack 38

Hình 15 Sơ đồ hoạt động MapReduce. 39

Hình 16 Sơ đồ hoạt động của YARN. 40

Hình 17 Kiến trúc ZooKeeper 43

Hình 18 Mô hình dữ liệu của ZooKeeper 45

Hình 19 Kiến trúc API 46

Hình 20 Kiến trúc của Kafka cluster 47

Hình 21 Mối quan hệ giữa partition và topic 48

Hình 22 Quy tắc replicate của kafka partition. 49

Hình 23 Kiến trúc tổng quan Cassandra 50

Hình 24 Kiến trúc Cassandra cluster 51

Hình 25 Replicate theo chiều kim đồng hồ 52

Hình 26 Cơ chế replicate theo Network topology 52

Hình 27 Kiến trúc và thành phần của Spark 54

Hình 28 Mô hình workflow bao gồm 6 nút DAGs 58

Hình 29 Kiến trúc đơn node 59

Hình 30 Kiến trúc đa node 60

Hình 31 Tổng thể kiến trúc hệ thống và các thành phần dịch vụ amazon 62

Hình 32 Kiến trúc Amazon s3 63

Hình 33 Quy trình sử dụng amazon s3 cơ bản 66

Hình 34. Virtual machine và container-based architecture 84

Hình 35. Quy trình làm việc của Docker 86

Hình 37. Minh hoạ ReplicaSets 89

Hình 38. Khoa học dữ liệu lớn như một chồng phần mềm dịch vụ 91

Hình 39. Kiến trúc hệ thống mô hình 109

Hình 40 Các use case của machine learning toolbox 112

Hình 41 Benchmarking giữa kafka, rabbitmq và apache pulsar 120

Hình 42 Bảng thống kê dựa trên hình 41 120

Hình 43 Các ngôn ngữ phổ biến cho data scientists 123

Hình 44 Chi tiết sơ đồ hệ thống 124

Hình 45 Thống kê tai nạn giao thông và những tổn thất do ngủ gật gây ra 125

Hình 46 Các điểm trên khuôn mặt 128

Hình 47 Tỉ lệ EAR 130

Hình 48 Tỉ lệ MAR 131

Hình 49 Độ tròn đồng tử 131

Hình 50 Phương pháp predict tuần tự cho model phân loại cơ bản 135

Hình 51. Đồ thi đường độ chính xác của từng model 136

Hình 52. ROC curve 136

Hình 53. Độ quan trong từng feature 137

# DANH MỤC BẢNG

Bảng 1. So sánh giữa Spark và MapReduce 57

Bảng 2. Tóm tắt tính năng chính của các framework tích dữ liệu lớn 107

Bảng 3. So sánh giữa NoSQL và RDBMS 122

# DANH MỤC TỪ VIẾT TẮT

# TÓM TẮT KHÓA LUẬN

* Tổng quan đề tài: đặt vấn đề, lý do chọn đề tài, mục tiêu và phạm vi tiếp cận  
   đề tài.
* Tìm hiểu thực trạng, thị trường, khả năng phát triển.
* Nghiên cứu các công nghệ, cấu trúc, cơ chế hoạt động của Hệ thống dữ liệu  
   lớn. Thêm vào đó là các công nghệ mới và xu hướng như Cơ sở dữ liệu đồ   
   thị và phân tích dữ liệu tức thời.
* Phân tích các dịch vụ hệ thống dữ liệu lớn trả phí trên thị trường.
* Tìm hiểu các cách để xây dựng hệ thống dữ liệu lớn tại chỗ.
* Phát triển hệ thống dữ liệu lớn dựa trên các yêu cầu của hệ thống máy học.
* Thiết kế hệ thống máy học hỗ trợ các dịch vụ máy học.
* Trình bày kết luận và hướng phát triển đồ án.

# MỞ ĐẦU

## Lý do chọn đề tài:

Trong thời đại ngày nay, nhiều ứng dụng xã hội đang được phát triển, dẫn đến dữ liệu được tạo ra ồ ạt mỗi ngày. Ví dụ như các nền tảng truyền thông xã hội, hàng triệu người dùng kết nối hàng ngày, thông tin được chia sẻ bất cứ khi nào người dùng sử dụng nền tảng truyền thông xã hội hoặc bất kỳ trang web nào khác. Chính vì thế, câu hỏi được đặt ra là lượng dữ liệu này khổng lồ này được xử lý như thế nào và dữ liệu được xử lý, lưu trữ qua phương tiên hay công cụ nào. Đây là lúc Dữ liệu lớn bước vào cuộc chơi.

Sự phát triển và tích hợp liên tục của lưu trữ dữ liệu, tính toán, thiết bị kỹ thuật số và mạng đã tạo ra một môi trường phong phú cho sự phát triển bùng nổ của Dữ liệu lớn cũng như các công cụ mà qua đó dữ liệu được tạo ra, chia sẻ, và phân tích. Theo SAS, Dữ liệu lớn được định nghĩa với 5V (Volume, Velocity, Variety, Veracity and Variability) (Dung lượng, Vận tốc, Độ đa dạng, Độ chính xác và Độ biến thiên), rồi khi dữ liệu bắt đầu thành “mỏ vàng”, thành nguồn doanh thu mới của mỗi doanh nghiệp thì xuất hiện V thứ 6 trong định nghĩa Dữ liệu lớn là Giá trị (Value). Giá trị thu được bằng cách phân tích Dữ liệu lớn và trích xuất từ chúng các mô hình, xu hướng và mô hình kiến thức ẩn bằng cách sử dụng các thuật toán và kỹ thuật phân tích dữ liệu thông minh. Phương pháp khoa học dữ liệu phải có khả năng phân tích Dữ liệu lớn và trích xuất các tính năng mà chúng ta không biết. Những tính năng đã học này cải thiện giá trị của dữ liệu sẽ giúp hiểu rõ hơn về hiện tượng và hành vi, tối ưu hóa các quy trình và cải thiện máy móc, kinh doanh và khám phá khoa học. Do đó, chúng ta không thể xem Khoa học dữ liệu lớn mà không xem phân tích dữ liệu và học máy là các bước chính để bao gồm giá trị như một chiến lược Khoa học dữ liệu lớn.

Trong thực tế, các công cụ phân tích dữ liệu lớn cho phép các nhà khoa học dữ liệu khám phá các mối tương quan và các mẫu thông qua việc phân tích lượng dữ liệu khổng lồ từ nhiều nguồn khác nhau thuộc các loại khác nhau. Gần đây, khoa học dữ liệu lớn đã nổi lên như một môn học phân tích dữ liệu hiện đại và quan trọng. Nó được coi là sự hợp nhất của các ngành cổ điển như thống kê, trí tuệ nhân tạo, toán học và khoa học máy tính với các phân ngành của nó bao gồm hệ thống cơ sở dữ liệu, máy học và hệ thống phân tán. Nó kết hợp các phương pháp tiếp cận hiện tại với mục đích biến dữ liệu dồi dào thành giá trị cho các cá nhân, tổ chức và xã hội. Mục tiêu cuối cùng của kỹ thuật khoa học dữ liệu là chuyển đổi dữ liệu thành thông tin có ý nghĩa. Cả trong kinh doanh và trong khoa học, các phương pháp khoa học dữ liệu đã cho thấy khả năng ra quyết định mạnh mẽ hơn.

Trong vài năm qua, chúng ta đã chứng kiến sự xuất hiện lớn của Khoa học dữ liệu lớn trong các ứng dụng thực tế khác nhau như tối ưu hóa kinh doanh, giao dịch tài chính, phân tích dữ liệu chăm sóc sức khỏe và phân tích mạng xã hội. Cụ thể, chúng ta có thể nghĩ về mối quan hệ giữa Dữ liệu lớn và Khoa học dữ liệu giống như mối quan hệ giữa dầu thô và nhà máy lọc dầu.

Dưới sự thúc đẩy bởi những tiến bộ liên tục và mạnh mẽ về sức mạnh xử lý, bộ nhớ, lưu trữ và vô số dữ liệu chưa từng có, nền tảng xử lý Dữ liệu lớn đã được phát triển để giải quyết các công việc khoa học dữ liệu ngày càng phức tạp. Được dẫn dắt bởi bộ Nền tảng Hadoop và hệ sinh thái của nó, các hệ thống xử lý Dữ liệu lớn đang cho thấy thành công đáng kể trong một số lĩnh vực kinh doanh và nghiên cứu. Đặc biệt, trong khoảng một thập kỷ, nền tảng Hadoop đại diện cho tiêu chuẩn defacto của thế giới phân tích Dữ liệu lớn. Tuy nhiên, gần đây chúng tôi đã chứng kiến một làn sóng nền tảng xử lý Big Data 2.0 mới dành riêng cho các ngành cụ thể như xử lý dữ liệu SQL có cấu trúc (ví dụ: Hive, Impala, Presto), xử lý đồ thị quy mô lớn (ví dụ: Neo4j, Giraph, Graphlab, GraphX), dữ liệu xử lý luồng quy mô lớn (ví dụ: Storm, Heron, Flink, Samza, Kafka) và học máy và phân tích dữ liệu (Pig, Mahout, Spark MLib, Azure ML).

Các kỹ thuật và công nghệ của Khoa học dữ liệu lớn đã có thể thâm nhập vào tất cả các khía cạnh của lĩnh vực kinh doanh và nghiên cứu. Từ doanh nghiệp kinh doanh hiện đại đến lựa chọn lối sống của công dân kỹ thuật số ngày nay, những hiểu biết về phân tích Dữ liệu lớn đang thúc đẩy những thay đổi và cải tiến trong mọi lĩnh vực. Một số khảo sát dữ liệu lớn đã được trình bày trong tài liệu bên dưới (xem mục tham khảo). Để bắt đầu, chúng tôi bao quát toàn diện về phối cảnh hệ thống và nền tảng cho các lớp khác nhau của các kỹ thuật phân tích dữ liệu (ví dụ: học máy, học sâu). Chúng tôi tóm tắt những nội dung chính của quá trình nghiên cứu này như sau:

* Chúng tôi phân tích các tính năng chính và các khối xây dựng của ngăn xếp phần mềm để hỗ trợ khoa học Dữ liệu lớn trên Đám mây như một dịch vụ hàng hóa cho các nhà khoa học dữ liệu.
* Chúng tôi xác định một tập hợp các yêu cầu chính để đạt được hiệu quả tầm nhìn về việc cung cấp phân tích Dữ liệu lớn dưới dạng dịch vụ.
* Chúng tôi phân tích và phân loại các khung phân tích Dữ liệu lớn tiên tiến dựa trên các mô hình dịch vụ được hỗ trợ của họ.
* Chúng tôi cung cấp những hiểu biết khác nhau về những phát triển mới nhất đang diễn ra và những thách thức mở trong lĩnh vực này.
* Chúng tôi thực hiện các ví dụ thực tế về việc áp dụng Hệ thống dữ liệu lớn kết hợp với Hệ thống máy học.

# Chương 1. TỔNG QUAN ĐỀ TÀI

## Mục tiêu và phạm vi

### **Mục tiêu**

* + - * Tìm hiểu các ưu điểm, nhược điểm của các hệ thống dữ liệu lớn hiện tại.
      * Nghiên cứu các công cụ hỗ trợ làm data warehouse, data processing, ETL hiện tại có trên thị trường.
      * Thiết kế hệ thống hỗ trợ dữ liệu lớn phù hợp với yêu cầu phục vụ hệ thống máy học.

### **Phạm vi**

Tìm ra các ưu điểm, khuyết điểm của các hệ thống dữ liệu lớn hiện tại và cải thiện và tối ưu chúng

## Nội dung thực hiện

Mục tiêu của đề tài “Nghiên cứu hệ thống dữ liệu lớn hỗ trợ hệ thống nhận diện sự buồn ngủ”:

* Tìm hiểu về Dữ liệu lớn, các khái niệm chuyên ngành.
* Tìm hiểu về Hệ thống dữ liệu lớn, các dịch vụ thương mại có trên thị trường.
* Nghiên cứu các yếu tố, thành phần cấu thành Hệ thống dữ liệu lớn
* Thiết kế Hệ thống dữ liệu lớn hỗ trợ các dịch vụ máy học.
* Tích hợp và triển khai.
* Viết báo cáo.

# Chương 2. DỮ LIỆU LỚN LÀ GÌ, CÁC KHÁI NIỆM CHUYÊN NGÀNH CẦN LÀM RÕ

## Khảo sát thực trạng

Ngày nay, dường như mọi người đang nói về thuật ngữ dữ liệu lớn. Tuy nhiên, phần lớn trong số họ không chắc chắn đó là gì và họ sẽ tận dụng tối đa nó như thế nào. Ngoài một vài công ty, hầu hết trong số họ vẫn còn bối rối về khái niệm này và chưa sẵn sàng áp dụng ý tưởng này. Ngay cả khi chúng ta nghe thấy thuật ngữ dữ liệu lớn, rất nhiều câu hỏi xuất hiện trong đầu chúng ta. Điều rất quan trọng để hiểu những khái niệm này. Những câu hỏi này bao gồm:

* Dữ liệu lớn là gì?
* Tại sao lại phải sử dụng Dữ liệu lớn?
* Liệu dữ liệu lớn có bất kỳ đặc điểm nào và đây là những gì?
* Làm sao để lưu trữ dữ liệu khổng lồ như vậy?

Trong chương này, chúng tôi sẽ tập trung vào việc trả lời những câu hỏi này và xây dựng một nền tảng vững chắc để tìm hiểu thế giới của thế giới dữ liệu lớn. Chủ yếu, chúng tôi sẽ đề cập đến các chủ đề sau:

* Dữ liệu lớn
* Các đặc điểm của dữ liệu lớn
* Các cân nhắc thiết kế khác nhau cho các giải pháp dữ liệu lớn
* Thuật ngữ chính được sử dụng trong thế giới dữ liệu lớn

## Dữ liệu lớn là gì?

Dữ liệu lớn có thể hiểu một cách đơn giản như sau:

*Dữ liệu lớn thể hiện một lượng lớn dữ liệu – có cấu trúc hay không có cấu trúc, được sinh ra từ các công việc hằng ngày.*

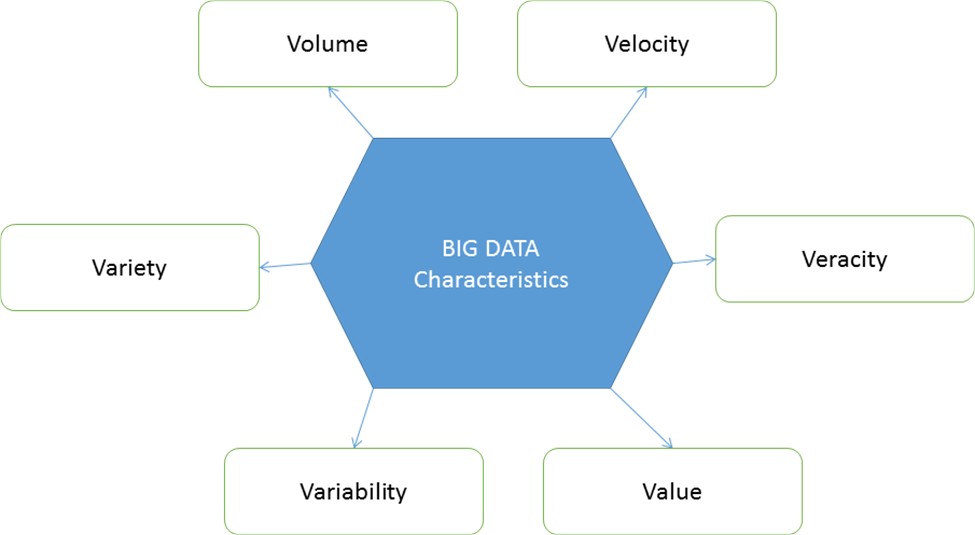
Sinh ra từ các công việc hằng ngày như từ việc truy cập, đăng tải trên Facebook, số ảnh được đăng trên Twitter trong một ngày đến hồ sơ bệnh lý, bệnh nhân, dữ liệu được ghi lại, lưu trữ và phân tích để cho phép công nghệ và dịch vụ tạo ra thống kê, sản phẩm để phục vụ cho nhiều nhu cầu khác nhau. Một điều quan trọng trong khái niệm của Dữ liệu lớn mà chúng ta hay nhầm lẫn là không phải khối lượng dữ liệu quan trọng mà cách ta xử lý và sử dụng những dữ liệu, thông tin ấy. Chính vì thế, dữ liệu lớn được lưu trữ để nghiên cứu và phận tích để chúng ta có thể đưa ra kết luận, hành động tốt hơn

Bạn đã bao giờ xem YouTube đang gợi ý cho bạn những video mà bạn có khả năng xem nhiều nhất chưa? Google đang phân phát quảng cáo địa phương cho bạn như thế nào, nhắm mục tiêu cụ thể đến bạn như những quảng cáo bạn sẽ mở hoặc sản phẩm bạn đang tìm kiếm? Các công ty này đang giữ tất cả các hoạt động bạn làm trên trang web của họ và sử dụng chúng cho trải nghiệm người dùng tốt hơn, cũng như vì lợi ích của họ, để tạo doanh thu. Có rất nhiều ví dụ có sẵn của loại hành vi này và nó đang gia tăng khi ngày càng nhiều công ty nhận ra sức mạnh của dữ liệu. Điều này đặt ra một thách thức cho các nhà nghiên cứu công nghệ: đưa ra các giải pháp mạnh mẽ và hiệu quả hơn có thể đáp ứng các thách thức và yêu cầu mới.

Bây giờ, khi chúng ta có một số hiểu biết về dữ liệu lớn là gì, chúng ta sẽ tiến lên phía trước và thảo luận về các đặc điểm khác nhau của nó.

## Đặc điểm của dữ liệu lớn

Năm 2001, Doug Laney lần đầu tiên trình bày cái được gọi là ba Vs của dữ liệu lớn để mô tả một số đặc điểm làm cho dữ liệu lớn khác với xử lý dữ liệu khác. Ba Vs này là khối lượng, vận tốc và sự đa dạng. Đây là thời đại của sự tiến bộ công nghệ và vô số nghiên cứu đang diễn ra. Kết quả của việc đạt được và tiến bộ này, ba Vs này đã trở thành sáu Vs của dữ liệu lớn như bây giờ. Nó cũng có thể tăng trong tương lai. Cho đến nay, sáu Vs của dữ liệu lớn là khối lượng, vận tốc, sự đa dạng, tính xác thực, tính biến đổi và giá trị, như được minh họa trong sơ đồ sau. Những đặc điểm này sẽ được thảo luận chi tiết sau trong chương này:



Các kích thước bộ nhớ máy tính khác nhau được liệt kê trong bảng sau để cho bạn ý tưởng về các chuyển đổi giữa các đơn vị khác nhau. Nó sẽ cho bạn hiểu kích thước của dữ liệu trong các ví dụ sắp tới trong chương này:

|  |  |
| --- | --- |
| **1 Bit** | **Binary digit** |
| 8 Bits | 1 byte |
| 1,024 Bytes | 1 KB (kilobyte) |
| 1,024 KB | 1 MB (megabyte) |
| 1,024 MB | 1 GB (gigabyte) |
| 1,024 GB | 1 TB (terabyte) |
| 1,024 TB | 1 PB (petabyte) |
| 1,024 PB | 1 EB (exabyte) |
| 1,024 EB | 1 ZB (zettabyte) |
| 1,024 ZB | 1 YB (yottabyte) |
| 1,024 YB | 1 brontobyte |
| 1,024 brontobyte | 1. geopbyte |

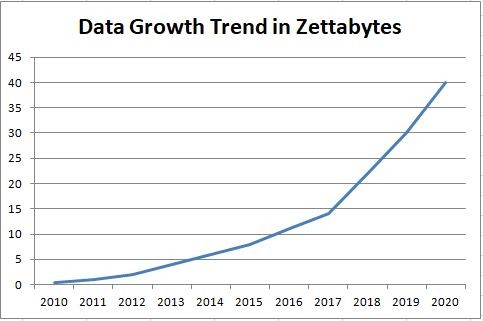
### **2.3.1. Dung lượng**

Là khối lượng dữ liệu được tạo ra, lưu trữ và xử lý. Với sự phát triển của phương tiện truyền thông xã hội và các tài nguyên internet khác, mọi người đang đăng và tải lên rất nhiều nội dung, video, hình ảnh, tweet, v.v. Cứ thử hình dung:

7 tỷ người và gần 6 tỷ trong số họ có điện thoại di động. Một ngày, mỗi cá thể sử dụng điện thoại với vô vàng mục đích từ gọi điện, nhắn tin, chụp hình, giải trí. Mỗi hành động của mỗi cá thể như thế chỉ từ một thiết bị điện tử đã sinh ra khối lượng dữ liệu cực lớn.

Khi chúng ta nói về khối lượng trong bối cảnh dữ liệu lớn, đó là một lượng dữ liệu khổng lồ đối với hệ thống xử lý truyền thống không thể đáp ứng và hoàn thành các nhu cầu của chúng ta trong thời gian yêu cầu nữa.

Lấy ví dụ về Facebook. Họ tạo ra khoảng 600 TB dữ liệu mỗi ngày từ hơn 2 tỷ người dùng hoạt động liên tục sử dụng trang mạng xã hội này để chia sẻ trạng thái, ảnh, video, nhận xét về bài đăng, lượt thích, không thích và nhiều hoạt động khác của họ. Biểu đồ sau đây biểu thị khối lượng dữ liệu được sinh ra đã có trong những năm trước, tình hình hiện tại và nơi nó sẽ đến trong tương lai:

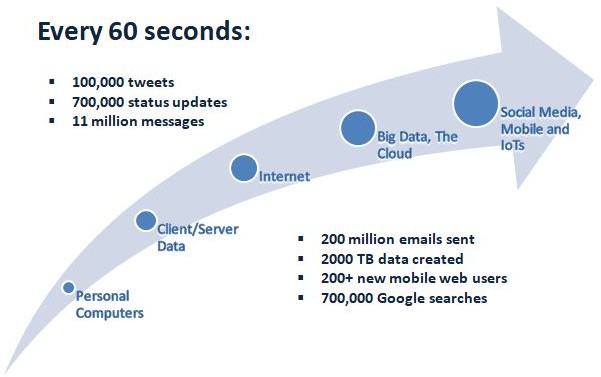


Hình 1 Tăng trưởng dữ liệu trong quá khứ, hiện tại và tương lai

### **2.3.2. Vận tốc**

Là tốc độ tạo ra dữ liệu hoặc tốc độ dữ liệu đến. Hãy tưởng tượng lượng dữ liệu Facebook, YouTube hoặc bất kỳ trang web mạng xã hội nào đang nhận được mỗi ngày. Họ phải lưu trữ nó, xử lý nó và bằng cách nào đó sau đó có thể lấy nó. Dưới đây là một vài ví dụ về việc dữ liệu tăng nhanh như thế nào:

* + - * Sàn giao dịch chứng khoán New York thu được 1 TB dữ liệu trong mỗi phiên giao dịch.
      * 0.4 triệu giờ video đang được tải lên YouTube.
      * 3.5 tỷ lượt tìm kiếm được thực hiện mỗi ngày trên Google.
      * 200 triệu email được gửi mỗi phút.



Hình 2. Vận tốc mà dữ liệu được tạo ra

Biểu đồ trước cho thấy lượng thời gian người dùng đang dành cho các trang web mạng xã hội phổ biến. Hãy tưởng tượng tần suất dữ liệu được tạo ra dựa trên các hoạt động của người dùng này. Trong lúc ta đang xem hình ảnh này có thể những con số này đã tăng vụt và liên tục thay đổi.

### **Đa dạng**

Là tính đa dạng của dữ liệu được lưu trữ. Nó có thể có cấu trúc hoặc không có cấu trúc. Dữ liệu có cấu trúc được ưu tiên cho thông tin có lược đồ được xác định trước hoặc có mô hình dữ liệu với các cột, loại dữ liệu được xác định trước, v.v., trong khi dữ liệu phi cấu trúc không có bất kỳ đặc điểm nào trong số này. Chúng bao gồm một danh sách dài các dữ liệu, chẳng hạn như tài liệu, email, tin nhắn văn bản phương tiện truyền thông xã hội, video, hình ảnh tĩnh, âm thanh, đồ thị, đầu ra từ tất cả các loại dữ liệu do máy tạo từ cảm biến, thiết bị, thẻ RFID, nhật ký

máy và tín hiệu GPS điện thoại di động, và nhiều hơn nữa. Ngoài ra, còn có kiểu bán cấu trúc như file json hay file xml.

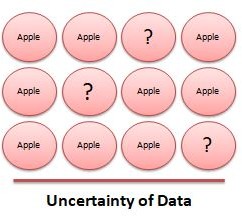


Hình 3 Sự đa dạng của dữ liệu

### **Độ chính xác**

Là mức độ tin cậy của dữ liệu. Khi khối lượng dữ liệu ngày càng lớn, tính đa dạng của dữ liệu ngày càng phong phú thì mức độ tin cậy của dữ liệu ngày càng giảm xuống.

Sơ đồ sau đây là một minh họa về ví dụ của táo:

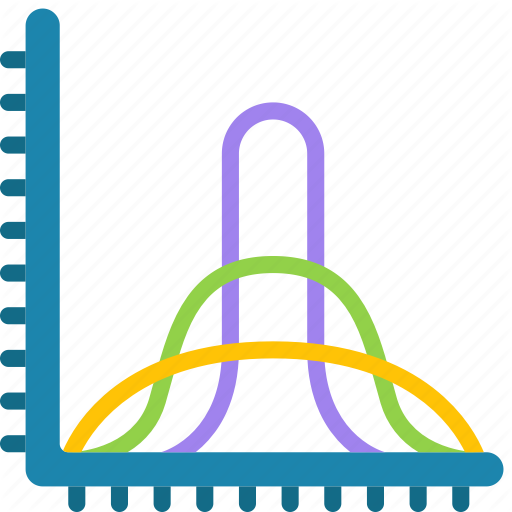


Hình 4 Mô tả sự không rõ ràng trong ví dụ của táo

Độ chính xác cũng nói lên chất lượng của dữ liệu. Chúng ta không thể nào sử dụng những thông tin, dữ liệu cho một bảng báo cáo từ một nguồn thông tin không rõ nguồn gốc hay tải những sản phẩm lạ không có kiểm chứng.

### **Tính biến thiên**

Tính biến thiên của dữ liệu phản ánh sự không nhất quán trong dữ liệu.



Hình 5. Tính biến thiên của dữ liệu

Ví dụ như một từ có thể mang nhiều nghĩa khác nhau tùy vào hoàn cảnh, trường hợp mà ta sử dụng nó. Ngoài ra, tính biến thiên còn thể hiện qua sự chênh lệch về kích thước và tốc độ dữ liệu được sinh ra, do dữ liệu được thu thập từ nhiều nguồn khác nhau.

### **Giá trị**

Tuy là một đặc điểm mới nhưng lại mang tầm quan trọng hết sức đặc biệt khi Dữ liệu lớn được xem là “mỏ vàng” tuy mang ý nghĩa về mặt dữ liệu. Sau khi giải quyết tất cả các Vs, khối lượng, vận tốc, sự đa dạng, tính biến đổi và tính xác thực khác, việc này tốn rất nhiều thời gian, công sức và tài nguyên, giờ là lúc quyết định xem có nên lưu trữ dữ liệu đó và đầu tư vào cơ sở hạ tầng hay không, tại cơ sở hoặc trong đám mây. Một khía cạnh của giá trị là bạn phải lưu trữ một lượng dữ liệu khổng lồ trước khi bạn có thể sử dụng nó để cung cấp thông tin có giá trị. Trước đây, việc lưu trữ khối lượng dữ liệu này khiến bạn mất chi phí rất lớn, nhưng bây giờ công nghệ lưu trữ và truy xuất lại ít tốn kém hơn nhiều. Bạn muốn chắc chắn rằng tổ chức của bạn nhận được giá trị từ dữ liệu. Việc phân tích cần phải được thực hiện để đáp ứng các cân nhắc về đạo đức.

Bây giờ chúng ta đã thảo luận và hiểu sáu Vs của dữ liệu lớn, đã đến lúc mở rộng phạm vi hiểu biết của chúng ta và tìm hiểu phải làm gì với dữ liệu có các đặc điểm này. Các công ty vẫn có thể nghĩ rằng các hệ thống truyền thống của họ là đủ cho dữ liệu có các đặc điểm này, nhưng nếu chúng vẫn nằm dưới ảnh hưởng này, họ có thể mất trong thời gian dài. Bây giờ chúng ta đã hiểu tầm quan trọng của dữ liệu và đặc điểm của dữ liệu, trọng tâm chính là cách lưu trữ dữ liệu, cách xử lý dữ liệu, lưu trữ dữ liệu nào, kết quả phân tích dự kiến sẽ nhanh như thế nào. Các giải pháp khác nhau để xử lý loại dữ liệu này, mỗi loại có ưu và nhược điểm riêng, có sẵn trên thị trường, trong khi các giải pháp mới liên tục được phát triển.

## Cách tiếp cận dựa trên giải pháp cho dữ liệu

Tăng doanh thu và lợi nhuận là trọng tâm chính của bất kỳ tổ chức nào. Nhắm mục tiêu này đòi hỏi hiệu quả và hiệu quả từ nhân viên, đồng thời giảm thiểu rủi ro ảnh hưởng đến tăng trưởng chung. Mọi tổ chức đều có đối thủ cạnh tranh và để cạnh tranh với họ, bạn phải suy nghĩ và hành động nhanh chóng và hiệu quả trước đối thủ cạnh tranh. Hầu hết những người ra quyết định phụ thuộc vào số liệu thống kê có sẵn cho họ và nêu ra các vấn đề sau:

* + - 1. Điều gì nếu bạn nhận được báo cáo phân tích nhanh hơn so với các hệ thống truyền thống?
      2. Điều gì sẽ xảy ra nếu bạn có thể dự đoán cách khách hàng cư xử, các xu hướng khác nhau và các cơ hội khác nhau để phát triển doanh nghiệp của bạn, gần với thời gian thực?
      3. Điều gì nếu bạn có hệ thống tự động có thể bắt đầu các nhiệm vụ quan trọng tự động?
      4. Điều gì xảy ra nếu các hoạt động tự động làm sạch dữ liệu của bạn và bạn có thể đưa ra quyết định dựa trên dữ liệu đáng tin cậy?
      5. Điều gì nếu bạn có thể dự đoán rủi ro và định lượng chúng?

Bất kỳ người quản lý nào, nếu họ có thể có câu trả lời cho những câu hỏi này, có thể hành động hiệu quả để tăng doanh thu và tăng trưởng của bất kỳ tổ chức nào, nhưng nhận được tất cả những câu trả lời này chỉ là một kịch bản lý tưởng.

### **2.4.1. Dữ liệu - tài sản quý giá nhất**

Gần một thập kỷ trước, mọi người bắt đầu nhận ra sức mạnh của dữ liệu: nó quan trọng như thế nào và nó có thể giúp các tổ chức phát triển như thế nào. Nó có thể giúp họ cải thiện công việc kinh doanh dựa trên thực tế chứ không phải bản năng của họ. Có rất ít nguồn dữ liệu để thu thập và phân tích dữ liệu này ở dạng thô không phải là một nhiệm vụ dễ dàng.

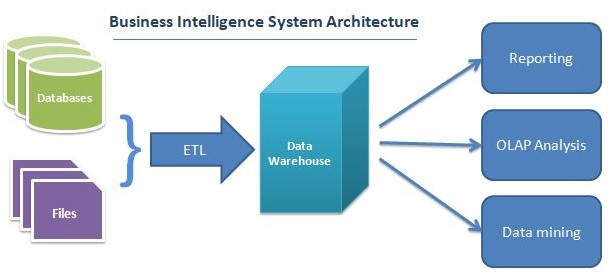
Bây giờ, lượng dữ liệu đang tăng theo cấp số nhân, ít nhất là gấp đôi mỗi năm, các giải pháp dữ liệu lớn được yêu cầu để tận dụng tối đa tài sản của bạn. Nghiên cứu

liên tục đang được thực hiện để đưa ra các giải pháp mới và các cải tiến thường xuyên đang diễn ra để phục vụ các yêu cầu, sau khi nhận ra thực tế quan trọng rằng dữ liệu là tất cả.

### **2.4.2. Phương pháp truyền thống để lưu trữ dữ liệu**

Các quy trình chuyên sâu của con người chỉ hoạt động để hiểu được dữ liệu không mở rộng khi khối lượng dữ liệu tăng. Ví dụ, mọi người thường đặt từng bản ghi của hồ sơ công ty vào một loại bảng tính nào đó, và sau đó rất khó tìm và phân tích thông tin đó một khi khối lượng hoặc vận tốc thông tin tăng lên.

Các hệ thống truyền thống sử dụng các công việc hàng loạt, lên lịch cho chúng trên cơ sở hàng ngày, hàng tuần hoặc hàng tháng để di chuyển dữ liệu vào các máy chủ riêng biệt hoặc vào kho dữ liệu. Dữ liệu này có lược đồ và được phân loại là dữ liệu có cấu trúc. Sau đó, nó sẽ trải qua chu trình xử lý và phân tích để tạo bộ dữ liệu và trích xuất thông tin có ý nghĩa. Các kho dữ liệu này được tối ưu hóa cho mục đích báo cáo và phân tích. Đây là khái niệm cốt lõi của hệ thống kinh doanh thông minh (BI). Hệ thống BI lưu trữ dữ liệu này trong các hệ thống cơ sở dữ liệu quan hệ. Sơ đồ sau minh họa một kiến trúc của hệ thống BI truyền thống:



Hình 6. Kiến trúc hệ thống kinh doanh thông minh

Vấn đề chính với phương pháp này là tốc độ trễ. Các báo cáo có sẵn cho những người ra quyết định không phải là thời gian thực và chủ yếu là vài ngày hoặc vài tuần, phụ thuộc vào ít nguồn hơn có mô hình dữ liệu tĩnh.

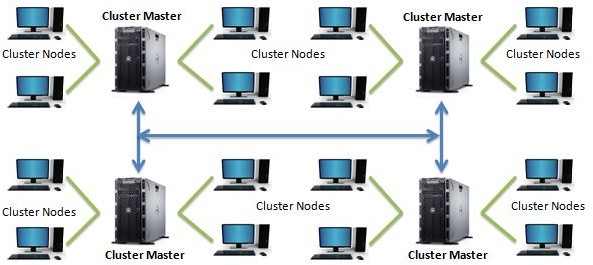
Đây là một kỷ nguyên của sự tiến bộ công nghệ và mọi thứ đang chuyển động rất nhanh. Bạn trả lời khách hàng càng nhanh, khách hàng của bạn sẽ càng hạnh phúc và điều đó sẽ giúp doanh nghiệp của bạn phát triển. Ngày nay, nguồn thông tin không chỉ là cơ sở dữ liệu giao dịch của bạn hoặc một vài mô hình dữ liệu khác; có nhiều nguồn dữ liệu khác có thể ảnh hưởng trực tiếp đến doanh nghiệp của bạn và nếu bạn không nắm bắt chúng và đưa chúng vào phân tích của mình, nó sẽ ảnh hưởng lớn đến bạn. Những nguồn này bao gồm bài đăng trên blog và đánh giá web, nguồn phương tiện truyền thông xã hội, bài đăng, tweet, hình ảnh, video và âm thanh. Và nó không chỉ là những nguồn này; Nhật ký được tạo bởi các cảm biến, thường được gọi là IoT (Internet of Things), trong điện thoại thông minh, thiết bị, robot và hệ thống tự trị, lưới điện thông minh và bất kỳ thiết bị nào đang thu thập dữ liệu, giờ đây có thể được sử dụng để nghiên cứu các hành vi và kiểu khác nhau, thứ gì đó mà là không thể tưởng tượng trong quá khứ.

Có một thực tế là mọi loại dữ liệu đang gia tăng, nhưng đặc biệt là các IoT, tạo ra nhật ký của từng sự kiện một cách tự động và liên tục. Ví dụ, một báo cáo được Intel chia sẻ nói rằng một chiếc xe tự trị tạo ra và tiêu thụ 4 TB dữ liệu mỗi ngày và đây chỉ là một giờ lái xe. Đây chỉ là một nguồn thông tin; nếu chúng tôi xem xét tất cả các nguồn được đề cập trước đây, chẳng hạn như blog và phương tiện truyền thông xã hội, nó sẽ không chỉ tạo ra một vài terabyte. Ở đây, chúng ta đang nói về exabyte, zettabyte hoặc thậm chí là yottabyte dữ liệu.

Chúng tôi đã nói về dữ liệu đang gia tăng, nhưng nó không chỉ là như vậy; các loại dữ liệu cũng đang phát triển. Ít hơn 20% các loại có lược đồ xác định. 80% khác chỉ là dữ liệu thô mà không có bất kỳ mẫu cụ thể nào không thể nằm trong các hệ thống cơ sở dữ liệu quan hệ truyền thống. 80% dữ liệu này bao gồm video, ảnh và dữ liệu văn bản bao gồm bài viết, bài viết và email.

Sau khi xem xét tất cả các đặc điểm này của dữ liệu và cố gắng thực hiện điều này trong giải pháp BI truyền thống, chúng ta thấy chỉ có thể sử dụng 20% dữ liệu của bạn, vì 80% khác chỉ là dữ liệu thô, làm cho nó vượt ra ngoài cơ sở dữ liệu quan hệ hệ thống. Trong thế giới ngày nay, mọi người đã nhận ra rằng dữ liệu mà họ cho là không có ích thực sự có thể tạo ra sự khác biệt lớn đối với việc ra quyết định và hiểu các hành vi khác nhau. Tuy nhiên, các giải pháp kinh doanh truyền thống không phải là cách tiếp cận chính xác để phân tích dữ liệu với các đặc điểm này, vì chúng chủ yếu hoạt động với lược đồ xác định và trên các công việc hàng loạt tạo ra kết quả sau một ngày, tuần hoặc tháng.

### **2.4.3. Điện toán cụm**

Trước khi chúng ta đi sâu hơn vào dữ liệu lớn, hãy tìm hiểu tính toán phân cụm một cách đơn giản nhất. Đây là một tập hợp các máy tính được kết nối với nhau theo cách chúng hoạt động như một máy chủ duy nhất cho người dùng cuối. Nó có thể được cấu hình để hoạt động với các đặc tính khác nhau cho phép tính sẵn sàng cao, cân bằng tải và xử lý song song. Mỗi máy tính trong các cấu hình này được gọi là một nút. Họ làm việc cùng nhau để thực hiện bất kỳ nhiệm vụ hoặc ứng dụng nào và hoạt động như một máy duy nhất. Sơ đồ sau minh họa một môi trường cụm máy tính:

Hình 7. Minh họa của một môi trường cụm máy tính

Khối lượng dữ liệu đang tăng lên, như chúng tôi đã nêu; bây giờ vượt quá khả năng của một máy tính để tự mình phân tích. Điện toán phân cụm kết hợp tài nguyên của nhiều máy giá rẻ nhỏ hơn, để đạt được nhiều lợi ích lớn hơn. Một số trong số họ được liệt kê ở đây trong các phần sau.

#### **2.4.3.1. Tính sẵn sàng cao**

Điều rất quan trọng đối với tất cả các công ty là dữ liệu và nội dung của họ phải luôn có sẵn và khi xảy ra bất kỳ lỗi phần cứng hoặc phần mềm nào, nó không phải là một thảm họa đối với họ. Điện toán cụm cung cấp các công cụ và cơ chế chịu lỗi để cung cấp thời gian hoạt động tối đa mà không ảnh hưởng đến hiệu suất, để mọi người đều có sẵn dữ liệu để phân tích và xử lý.

#### **2.4.3.2. Nguồn tổng hợp (Resource pooling)**

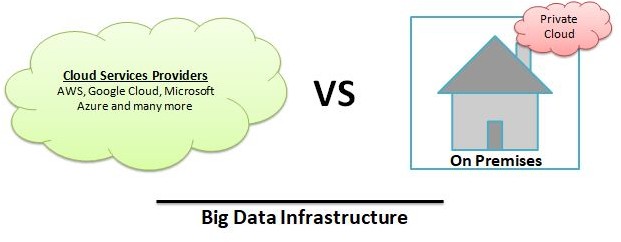
Trong điện toán cụm, nhiều máy tính được kết nối với nhau để hoạt động như một máy tính. Không chỉ là khả năng lưu trữ dữ liệu của họ được chia sẻ; Nhóm CPU và bộ nhớ cũng có thể được sử dụng trong các máy tính riêng lẻ để xử lý các tác vụ khác nhau một cách độc lập và sau đó hợp nhất các đầu ra để tạo ra kết quả. Để thực thi các bộ dữ liệu lớn, thiết lập này cung cấp sức mạnh xử lý hiệu quả hơn.

#### **2.4.3.3. Khả năng mở rộng dễ dàng**

Thu nhỏ là rất đơn giản trong điện toán cụm. Để thêm dung lượng lưu trữ bổ sung hoặc sức mạnh tính toán, chỉ cần thêm các máy mới có phần cứng cần thiết vào nhóm. Nó sẽ bắt đầu sử dụng các tài nguyên bổ sung với thiết lập tối thiểu, không cần phải mở rộng vật lý trong bất kỳ máy nào hiện có.

### **Giải pháp dữ liệu lớn - cơ sở hạ tầng đám mây so với tại chỗ**

Kể từ khi mọi người bắt đầu nhận ra sức mạnh của dữ liệu, các nhà nghiên cứu đã làm việc để sử dụng nó để trích xuất thông tin có ý nghĩa và xác định các mẫu khác nhau. Với những cải tiến công nghệ dữ liệu lớn, ngày càng có nhiều công ty bắt đầu sử dụng dữ liệu lớn và nhiều người hiện đang trên đà sử dụng các giải pháp dữ liệu lớn. Có nhiều cách cơ sở hạ tầng khác nhau để triển khai một thiết lập dữ liệu lớn. Cho đến một thời gian trước, lựa chọn duy nhất cho các công ty là thiết lập thiết lập trên trang web. Nhưng bây giờ họ có một tùy chọn khác: thiết lập đám mây. Nhiều công ty lớn, như Microsoft, Google và Amazon, hiện đang cung cấp một lượng lớn dịch vụ dựa trên yêu cầu của công ty. Nó có thể dựa trên cấu hình phần cứng của máy chủ và có thể để sử dụng năng lượng tính toán hoặc chỉ là không gian lưu trữ.



Hình 8. Các loại cơ sở hạ tầng dữ liệu lớn khác nhau

Yêu cầu của mỗi công ty là khác nhau; họ có những cách tiếp cận khác nhau cho những thứ khác nhau. Họ phân tích và khả thi trước khi áp dụng bất kỳ thay đổi lớn nào, đặc biệt là trong bộ phận công nghệ. Nếu công ty của bạn là một trong số họ

và bạn đang làm việc để áp dụng bất kỳ giải pháp dữ liệu lớn nào, hãy đảm bảo ghi nhớ những điều sau đây.

#### **Chi phí**

Đây là một yếu tố rất quan trọng, không chỉ đối với các công ty nhỏ mà còn đối với các công ty lớn. Đôi khi, nó là yếu tố quyết định duy nhất trong việc hoàn thiện bất kỳ giải pháp nào.

Thiết lập cơ sở hạ tầng trên trang web có một chi phí khởi đầu lớn. Nó liên quan đến các máy chủ cao cấp và thiết lập mạng để lưu trữ và phân tích thông tin. Thông thường, đó là một thiết lập hàng triệu đô la, rất khó cho các công ty nhỏ chịu; điều đó sẽ dẫn họ đến một cách tiếp cận truyền thống. Giờ đây, với sự xuất hiện của công nghệ, các công ty này có tùy chọn để lựa chọn thiết lập đám mây. Theo cách này, họ có tùy chọn để có một lối thoát từ chi phí khởi động, trong khi sử dụng toàn bộ tiềm năng của các phân tích dữ liệu lớn. Các công ty cung cấp thiết lập đám mây có các kế hoạch rất linh hoạt, chẳng hạn như người dùng có tùy chọn chỉ cần có máy chủ và tăng chúng theo yêu cầu. Mặt khác, thiết lập trên trang web yêu cầu các kỹ sư và chuyên gia mạng giám sát hoạt động toàn thời gian của thiết lập và duy trì nó. Trong một thiết lập đám mây, có ít rắc rối hơn cho các công ty; họ chỉ cần lo lắng về việc họ cần bao nhiêu dung lượng lưu trữ và cần bao nhiêu sức mạnh tính toán cho mục đích phân tích.

#### **Bảo mật**

Đây là một trong những mối quan tâm chính cho các công ty, bởi vì cơ sở của họ phụ thuộc vào nó. Thiết lập cơ sở hạ tầng tại cơ sở mang lại cho các công ty cảm giác an toàn hơn. Nó cũng cho phép họ kiểm soát ai đang truy cập dữ liệu của họ, khi nào nó được sử dụng và cho mục đích gì nó đang được sử dụng. Họ có thể thực hiện trách nhiệm của mình để đảm bảo dữ liệu được an toàn.

Mặt khác, dữ liệu trong đám mây có những rủi ro cố hữu. Nhiều câu hỏi phát sinh liên quan đến bảo mật dữ liệu khi bạn không biết nơi ở của dữ liệu của mình. Làm thế nào nó được quản lý? Những thành viên trong nhóm từ nhà cung cấp cơ sở hạ tầng đám mây có thể truy cập dữ liệu? Điều gì, nếu có, truy cập trái phép đã được

thực hiện để sao chép dữ liệu đó? Điều đó đang được nói, các nhà cung cấp cơ sở hạ tầng đám mây có uy tín đang thực hiện các biện pháp nghiêm túc để đảm bảo rằng mọi thông tin bạn đưa lên đám mây đều an toàn và bảo mật. Nhiều cơ chế mã hóa đang được triển khai để ngay cả khi có bất kỳ truy cập trái phép nào được thực hiện, dữ liệu sẽ vô dụng đối với người đó. Thứ hai, các bản sao bổ sung của dữ liệu của bạn đang được sao lưu trên các cơ sở hoàn toàn khác nhau để đảm bảo rằng bạn không bị mất dữ liệu của mình. Các biện pháp như thế này đang làm cho cơ sở hạ tầng đám mây gần như an toàn và bảo mật như thiết lập tại chỗ.

#### **Khả năng hiện tại**

Một yếu tố quan trọng khác cần xem xét trong khi triển khai các giải pháp dữ liệu lớn về mặt thiết lập tại chỗ so với thiết lập đám mây là liệu bạn hiện có nhân viên dữ liệu lớn để quản lý việc triển khai của mình trên trang web hay không. Bạn đã có một nhóm để hỗ trợ và giám sát tất cả các khía cạnh của dữ liệu lớn? Là nó với ngân sách của bạn, hoặc bạn có thể đủ khả năng để thuê chúng? Nếu bạn đang bắt đầu từ đầu, đội ngũ nhân viên cần thiết để thiết lập tại chỗ sẽ rất đáng kể, từ các kiến trúc sư dữ liệu lớn đến các kỹ sư hỗ trợ mạng. Điều đó không có nghĩa là bạn không cần một nhóm nếu bạn chọn thiết lập đám mây. Bạn vẫn sẽ cần các kiến trúc sư dữ liệu lớn để thực hiện giải pháp của mình để các công ty có thể tập trung vào những gì quan trọng, có ý nghĩa về thông tin được thu thập và thực hiện triển khai để cải thiện doanh nghiệp.

#### **Khả năng mở rộng**

Khi bạn cài đặt cơ sở hạ tầng cho dữ liệu lớn trên trang web, bạn phải thực hiện một số phân tích về việc sẽ thu thập bao nhiêu dữ liệu, dung lượng lưu trữ cần thiết để lưu trữ dữ liệu đó và cần bao nhiêu công suất tính toán cho mục đích phân tích. Theo đó, bạn phải quyết định phần cứng cần thiết trong thiết lập này. Bây giờ trong tương lai, nếu yêu cầu phân tích dữ liệu của bạn thay đổi, bạn có thể bắt đầu nhận dữ liệu từ nhiều nguồn hơn hoặc cần nhiều sức mạnh tính toán hơn để thực hiện phân tích về nó. Để đáp ứng yêu cầu này, bạn cần thêm máy chủ hoặc cài đặt phần cứng bổ sung trong thiết lập tại chỗ của bạn.

Một khía cạnh khác sẽ là: giả sử bạn đã phân tích rằng bạn sẽ thu thập được nhiều dữ liệu này, nhưng sau khi thực hiện, bạn sẽ không nhận được nhiều như vậy, hoặc trước đó, bạn cho rằng bạn sẽ cần sức mạnh tính toán nhiều này nhưng bây giờ, sau khi thực hiện, bạn nhận ra rằng bạn không. Trong cả hai trường hợp, bạn sẽ kết thúc với phần cứng máy chủ đắt tiền không sử dụng để quản lý. Trong thiết lập đám mây, bạn sẽ có tùy chọn mở rộng phần cứng lên xuống mà không phải lo lắng về các tác động tài chính tiêu cực và làm như vậy cực kỳ dễ dàng.

Bây giờ chúng ta sẽ chuyển sang thảo luận ngắn gọn về một số khái niệm và thuật ngữ chính mà bạn sẽ gặp trong cuộc sống hàng ngày của mình khi làm việc trong thế giới dữ liệu lớn.

### **Các thuật ngữ**

#### **Kho dữ liệu (Data warehouse)**

Một kho lớn dữ liệu có cấu trúc cho mục đích phân tích và báo cáo. Nó bao gồm các dữ liệu đã được làm sạch, với một lược đồ xác định và được tích hợp tốt với các nguồn. Nó thường được đề cập trong bối cảnh của các hệ thống truyền thống, chẳng hạn như BI.

#### **Hồ dữ liệu (Data lake)**

Tương tự như kho dữ liệu, để lưu trữ các bộ dữ liệu lớn, nhưng nó bao gồm dữ liệu phi cấu trúc. Đây là một thuật ngữ thường được sử dụng trong bối cảnh các giải pháp dữ liệu lớn lưu trữ thông tin như blog, bài đăng, video, hình ảnh, v.v.

#### **Khai thác dữ liệu (Data mining)**

Khai thác dữ liệu là quá trình cố gắng xử lý một khối dữ liệu thành một phương tiện trực quan và dễ hiểu hơn. Đây là một thuật ngữ rộng để thực hành cố gắng tìm các mẫu ẩn trong các bộ dữ liệu lớn.

#### **ETL**

ETL là viết tắt của trích xuất, chuyển đổi và tải (**extract, transform, and load**). Điều này chủ yếu đề cập đến các hệ thống truyền thống như BI, lấy dữ liệu thô và xử lý nó cho mục đích phân tích và báo cáo. Nó chủ yếu được liên kết với kho dữ

liệu, nhưng đặc điểm của quá trình này cũng được tìm thấy trong các đường ống ăn vào của các hệ thống dữ liệu lớn.

#### **Hadoop**

Một nền tảng phần mềm nguồn mở để xử lý các bộ dữ liệu rất lớn trong môi trường phân tán liên quan đến khả năng lưu trữ và tính toán, chủ yếu được xây dựng trên phần cứng hàng hóa chi phí thấp. Nó được thiết kế để dễ dàng mở rộng quy mô từ vài đến hàng ngàn máy chủ. Nó sẽ giúp xử lý dữ liệu được lưu trữ cục bộ trong một thiết lập xử lý song song tổng thể. Nó bao gồm các mô-đun khác nhau Hadoop Distributed File System (HDFS), Hadoop MapReduce và Hadoop YARN (Yet Another Resource Negotiator).

#### **Điện toán trong bộ nhớ (In-memory computing)**

Chiến lược liên quan đến việc di chuyển các bộ dữ liệu làm việc hoàn toàn trong bộ nhớ tập thể của cụm thay vì đọc nó từ đĩa cứng, để giảm thời gian xử lý trong khi bỏ qua các hoạt động ràng buộc I / O. Tính toán trung gian không được ghi vào đĩa và thay vào đó được giữ trong bộ nhớ. Đây là ý tưởng cơ bản của các dự án như Apache Spark. Do đó, nó có lợi thế rất lớn về tốc độ so với các hệ thống ràng buộc I

/ O như MapReduce.

#### **Học máy**

Nghiên cứu liên quan đến việc thiết kế một hệ thống có thể học mà không được lập trình rõ ràng. Nó có thể có khả năng tự điều chỉnh và cải thiện dựa trên dữ liệu được cung cấp cho nó. Nó liên quan đến việc thực hiện các thuật toán dự đoán và thống kê có thể liên tục bằng không về hành vi chính xác và hiểu biết khi có nhiều dữ liệu chảy qua hệ thống.

#### **MapReduce**

MapReduce là một khung để xử lý bất kỳ tác vụ nào trong môi trường phân tán. Nó hoạt động trên nguyên tắc Master / Slave tương tự HDFS. Nó liên quan đến việc phân tách một vấn đề được đặt thành các nút khác nhau có sẵn trong môi trường

điện toán phân cụm và tạo ra kết quả trung gian. Sau đó, nó xáo trộn các kết quả để căn chỉnh như các bộ và sau đó giảm chúng bằng cách tạo ra một giá trị duy nhất cho mỗi bộ.

#### **NoQuery**

NoQuery cung cấp một cơ chế lưu trữ và truy xuất dữ liệu không ở dạng bảng, chẳng hạn như những gì chúng ta lưu trữ trong các hệ thống cơ sở dữ liệu quan hệ. Nó cũng được gọi là không phải SQL hoặc cơ sở dữ liệu không liên quan. Nó rất phù hợp với dữ liệu lớn vì nó chủ yếu được sử dụng với dữ liệu phi cấu trúc và có thể hoạt động trong môi trường phân tán.

#### **Xử lý luồng (Stream processing)**

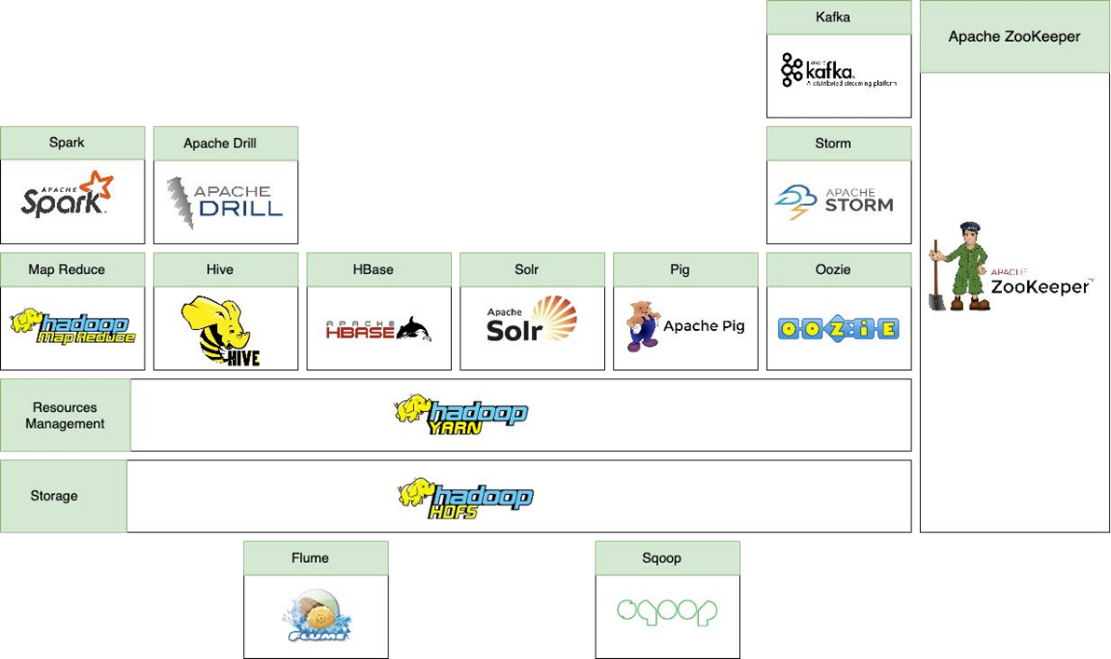
Việc thực hành xử lý các mục dữ liệu riêng lẻ khi chúng di chuyển qua một hệ thống. Điều này sẽ giúp phân tích dữ liệu theo thời gian thực khi nó được đưa vào hệ thống. Nó rất hữu ích cho các hoạt động nhạy cảm với thời gian bằng cách sử dụng các số liệu vận tốc cao.

# Chương 3. HỆ SINH THÁI HADOOP – HỆ THỐNG DỮ LIỆU LỚN TẠI CHỖ

Trong chương này, chúng ta sẽ xem xét chi tiết Dự án hệ sinh thái Hadoop của Apache, lý do tại sao nó được phát triển và những lợi thế và bất lợi của việc sử dụng hệ sinh thái Hadoop là gì. Chúng ta sẽ đi qua các thành phần thiết yếu của hệ sinh thái – những thành phần chính để có thể thiết kế một hệ thống dữ liệu lớn. chương này bao gồm:

* Apache Hadoop
* Apache kafka
* Apache Spark
* Apache Airflow
* Apache Cassandra

Từ đó chúng ta sẽ thảo luận ngắn gọn về các mục đích của các dự án này và cách chúng sẽ tác động đến việc ra quyết định của bạn trong thiết kế hệ thống dữ liệu lớn.



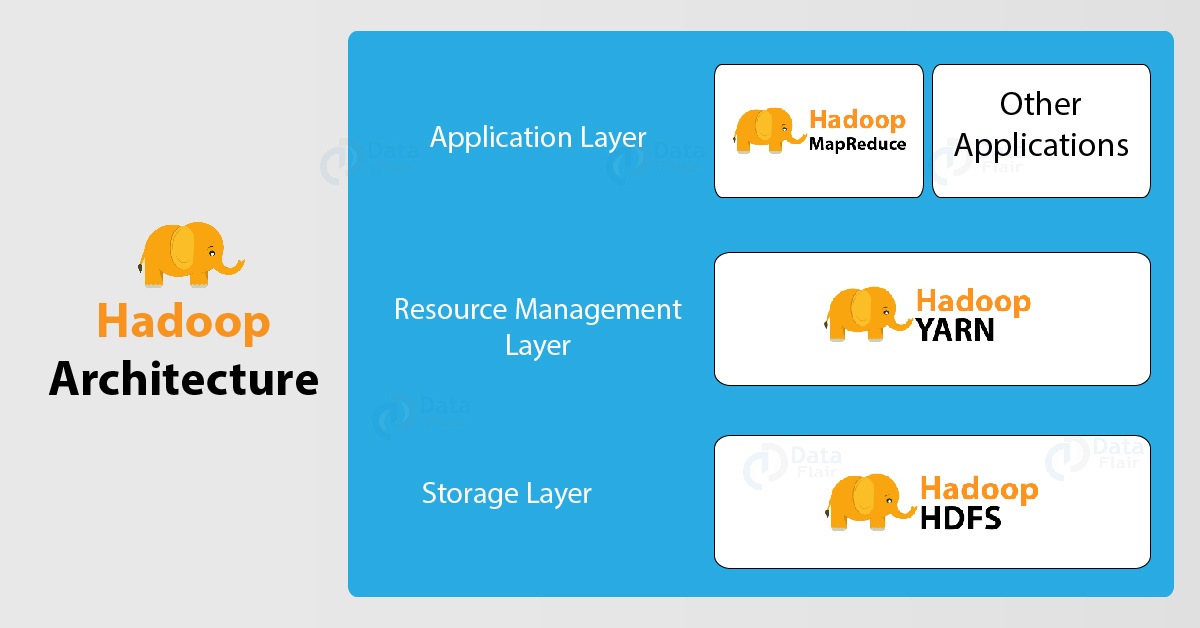
Hình 9. Tổng quan hệ sinh thái Hadoop

## Apache Hadoop

Apache Hadoop là một framework phần mềm mã nguồn mở cho phép lưu trữ và xử lý big data trên hệ thống phân tán. Hadoop được phát triển bởi Apache Software Foundation và được thiết kế bằng Java. Hiện nay, Hadoop đã trở thành một giải pháp phổ biến cho nhu cầu của thế giới. Kiến trúc của Hadoop luôn tuân thủ các mục tiêu: khả năng chịu lỗi, xử lý tập dữ liệu lớn, dữ liệu cục bộ, v.v.  
Sơ lược lịch sử phát triển của Hadoop:

* 2002: Được bắt đầu với dự án Apache Nutch
* 2007: Yahoo sử dụng Hadoop với 1000 node cluster
* 2008: Hadoop trở thành dự án top-level trên Apache
* 2011: Hadoop phiên bản 1.0 ra đời
* 2012: Hadoop phiên bản 2.0 ra đời, có chứa YARN
* 2017: Hadoop phiên bản 3.0 ra đời
* 2020: Hadoop phiên bản 3.2.1 ra đời

Apache Hadoop được thiết kế để dễ dàng mở rộng quy mô từ vài đến hàng nghìn máy chủ. Nó giúp bạn xử lý dữ liệu được lưu trữ cục bộ trong một thiết lập xử lý song song tổng thể. Hình dưới đây minh họa kiến trúc tổng thể của Apache Hadoop :



Hình 10. Mô hình kiến trúc Apache Hadoop

Apache Hadoop bao gồm bốn mô-đun chính. Các mô-đun này là Hệ thống tệp phân tán Hadoop (HDFS), Hadoop MapReduce (hoạt động với cả quản lý cụm và lớp xử lý), Yet Another Resource Negotiator (YARN, lớp quản lý cụm). Sau đây, chúng ta đến với thành phần đâu tiên trong Hadoop. [1]

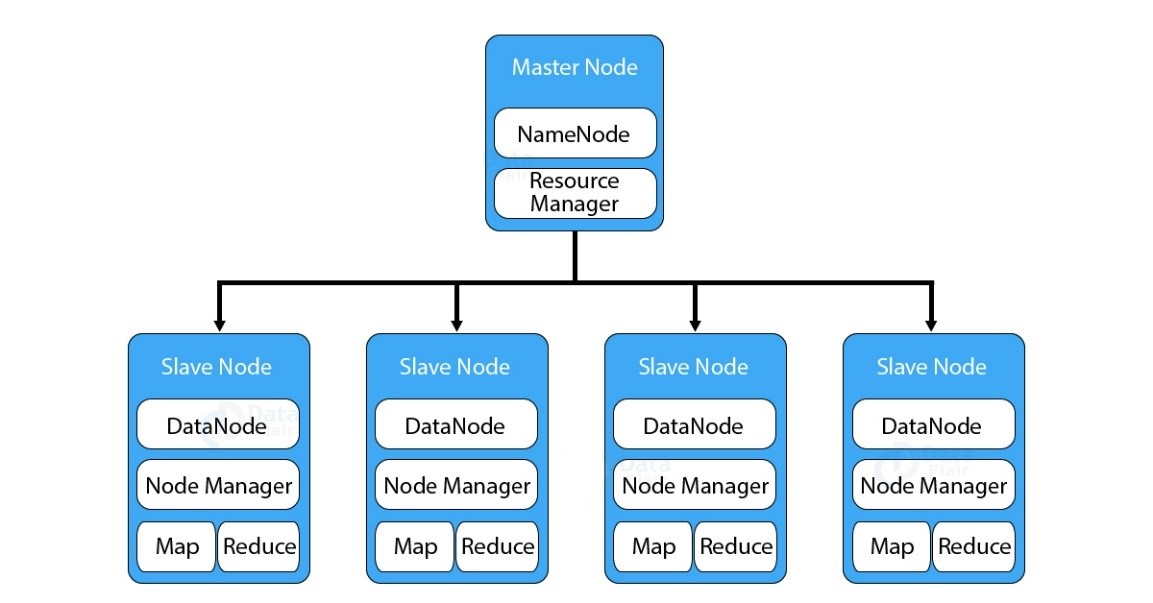
### **3.1.1. Hadoop Distributed File System (HDFS)**

HDFS là viết tắt của Hadoop Distributed File System. Nó cung cấp nơi lưu trữ dữ liệu cho Hadoop. HDFS chia dữ liệu thành các đơn vị nhỏ hơn được gọi là khối và lưu trữ chúng phân tán. Hệ phân tán của HDFS có cấu trúc gồm 2 thành phần: nút chính – Name Node và các nút phụ - Data Node.

#### **3.1.1.1. Name Node và Data Node**

Kiến trúc của HDFS (hay gọi là quan hệ giữa Name node và Data node) là chủ - tớ (Master – Slaves Architecture). Tương ứng, Name node sẽ đóng vai trò là Master – chủ, chịu trách nhiệm trong quản lý các Namespace, thay đổi tên thư mục, tệp và quyền truy cập. Các Data nodes sẽ đóng vai trò là Slaves – tôi tớ, dùng để chứa các dữ liệu từ người dùng. Dữ liệu bên trong các Data nodes sẽ được chia thành các khối (Block) và được nhân bản (replicate) sang các Data nodes khác. Cơ chế Replicate sẽ được làm rõ hơn ở phần tiếp theo.

Tóm lại, để thao tác với dữ liệu, cụ thể là truy cập, thêm, xóa, cập nhật vào hệ thống HDFS thì buộc phải thông qua Name node. Để rõ hơn thì chúng ta hãy quan sát hình vẽ bên dưới:

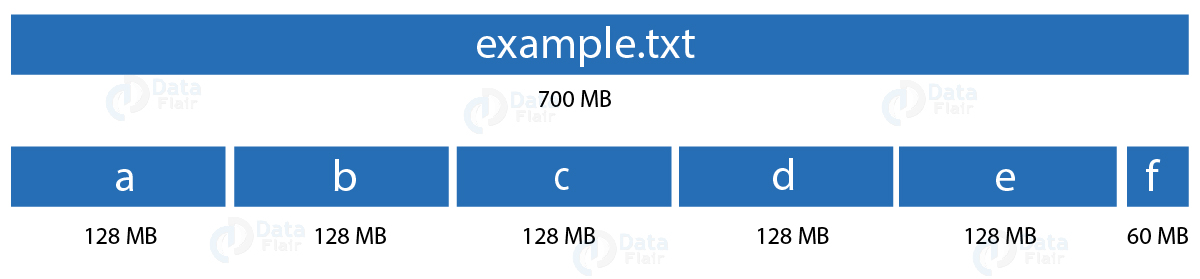


Hình 11. Kiến trúc master-slaves của HDFS (Name node – Data nodes)

Java là ngôn ngữ gốc của HDFS. Với đặc trưng là Write one – Run everywhere thì điều này khiến hệ thống HDFS hầu như không phụ thuộc vào môi trường cài đặt. Tất nhiên là vẫn cần phải cài đặt Java. Trong quá trình tìm hiểu triển khai thực tế, sẽ có 1 máy tính riêng biệt để cài đặt Name node. Các máy còn lại sẽ đóng vai trò là data node trong cụm máy (Cluster). Name node sẽ chứa các dữ liệu mô tả các data node (metadata) như: vị trí của các khối dữ liệu (block of data) trên các Data nodes. Tuy nhiên lưu ý, qua quá trình tự cài đặt, cấu hình trên docker, chúng tôi thấy việc này rất phức tạp và tốn nhiều tài nguyên thời gian, trí tuệ.

#### **3.1.1.2. Cách chia dữ liệu theo từng khối (Block)**

Khối (Block) là đơn vị lưu trữ nhỏ nhất trên hệ thống HDFS. Đây là bộ nhớ liền kề nhỏ nhất được phân bổ cho một tệp. Trong Hadoop, kích thước khối mặc định là 128MB hoặc 256 MB. Trong lập trình, ta có thể ví 1 Block đóng vai trò như 1 ô nhớ dung lượng là 1 byte. Như vậy, để lưu 1 biến int (4 bytes) thì ta cần 4 ô nhớ liền kề (1 byte mỗi ô). Để lấy dữ liệu ra lại chúng ta chỉ cần gộp 4 ô lại là có được giá trị, tương tự như Block trong HDFS.

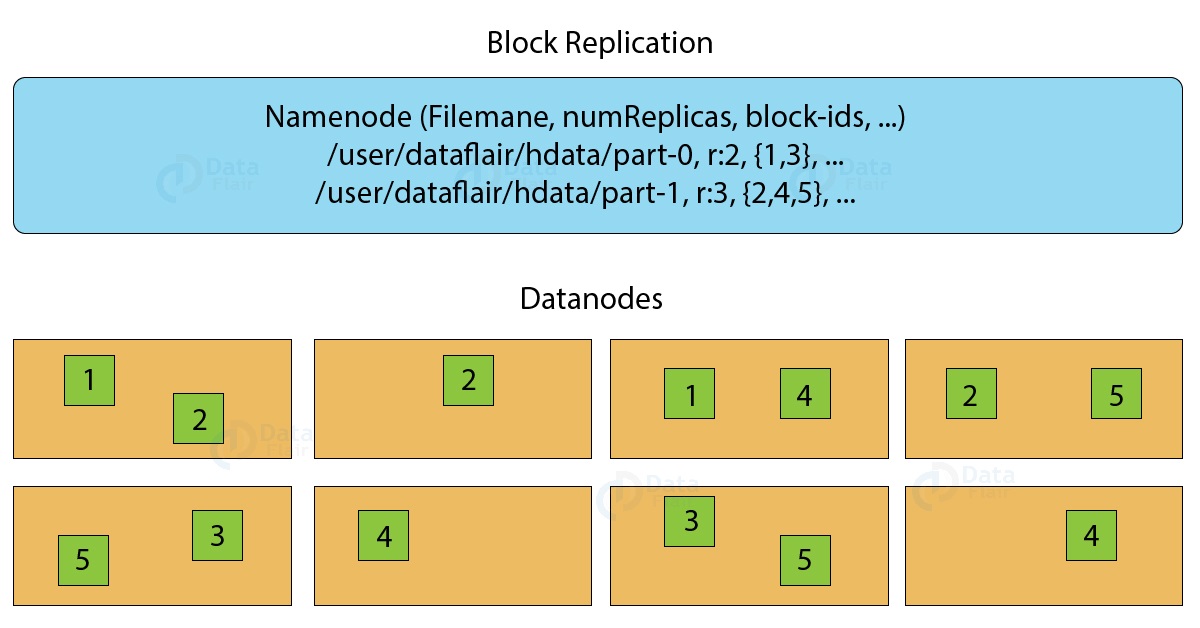


Hình 12. Ví dụ chia block trên 1 file.

Việc chọn kích thước khối rất quan trọng. Để giải thích lý do tại sao, chúng ta hãy nhìn hình 12 - một tệp có kích thước 700MB. Nếu kích thước khối của chúng ta là 128MB thì HDFS sẽ chia tệp thành 6 khối. Năm khối 128MB và một khối 60MB. Điều gì sẽ xảy ra nếu chúng ta cấu hình dung lượng khối (block size) là 4KB? Với kích thước khối 4KB, chúng ta sẽ có nhiều khối. Thực tế thì các tệp này thường có dung lượng vài Terabyte đến Petabyte. Điều này sẽ tạo ra một lượng lớn metadata trên Name node gây hiện tượng quá tải, lúc này các thao tác với dữ liệu sẽ cực kỳ chậm. Do đó, chúng ta phải chọn kích thước khối HDFS một cách thận trọng.

#### **3.1.1.3. Quản lý sao chép (Replication) phục vụ cơ chế chịu lỗi (Fault tolerance)**

Để cung cấp khả năng chịu lỗi, HDFS sử dụng kỹ thuật sao chép. Trong đó, nó tạo bản sao của các khối và lưu trữ trên các Data nodes khác nhau. Replicate property (Thuộc tính tạo bản sao) quyết định có bao nhiêu bản sao của các khối được lưu trữ. Mặc định là 3 nhưng chúng ta có thể thay đổi replicate property theo ý muốn.



Hình 13. Mô phỏng replication trên các data nodes.

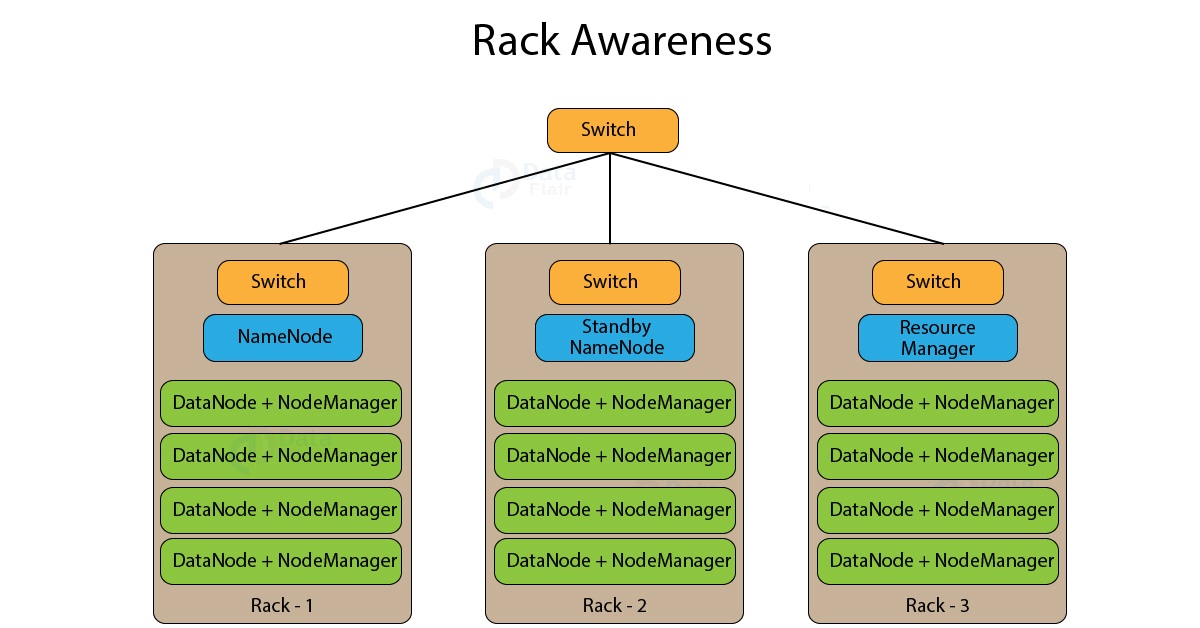
Hình trên cho thấy cách kỹ thuật nhân bản hoạt động. Giả sử chúng ta có một tệp 1GB thì với hệ số nhân bản là 3, nó sẽ yêu cầu tổng dung lượng lưu trữ là 3GB ( 1 bản gốc và 2 bản sao).

Để duy trì replicate propery, Name Node thu thập báo cáo về thông tin khối (vị trí, số thứ tự, dữ liệu) từ mọi Data Node. Bất cứ khi nào một khối bị sao chép thiếu hoặc thừ hơn so với replicate property, Name Node sẽ thêm hoặc xóa các bản sao tương ứng.

Bên cạnh ưu điểm là cơ chế chịu lỗi, trong quá trình nghiên cứu, chúng tôi cũng nhận ra các khuyết điểm của việc tạo nhiều bản sao:

* Tốn nhiều tiền hơn: việc tạo bản sao chắc chắn sẽ gây ra sự bành trướng về kho lưu trữ vật lý.
* Băng thông: việc duy trì tính nhất quán của dữ liệu có thể làm tăng giao thông trên mạng cục bộ.

#### **3.1.1.4. Rack Awareness phục vụ cơ chế hồi phục sau lỗi (Fault tolerance)**



Hình 14. Mô hình thực tế kiến trúc HDFS trên từng Rack

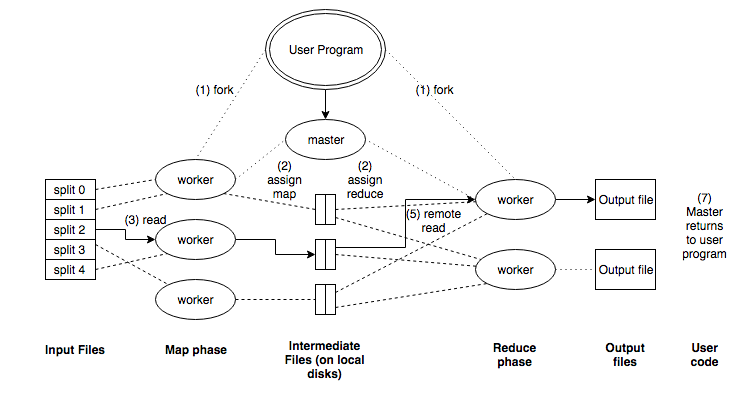
Một Rack chứa nhiều máy DataNode và nhiều hơn một rack như vậy trong môi trường production. HDFS tuân theo một thuật toán ( nhận biết rack – Rack awareness algorithm) để đặt các bản sao của các khối theo kiểu phân tán. Thuật toán này cung cấp khả năng chịu lỗi và độ trễ thấp. Giả sử hệ số nhân bản được cấu hình là 3. Thuật toán sẽ đặt khối đầu tiên trên một rack cục bộ của nó. Sau đó sẽ sao chép hai khối còn lại trên một rack khác. Đặc biệt, thuật toán ưu tiên không lưu trữ nhiều hơn hai khối bản sao cùng 1 bản chính trong cùng một rack.

### **3.1.2. MapReduce**

MapReduce là lớp xử lý dữ liệu của Hadoop. Nó là một framework cho phép bạn viết các ứng dụng để xử lý một lượng lớn dữ liệu. Các ứng dụng được MapReduce chạy song song trên cụm máy với khả năng chịu lỗi và độ tin cậy cao.

Công việc MapReduce bao gồm phân bổ tasks vào từng máy trên cluster để tính toán sau (Map tasks) đó gộp kết quả các task lại (Reduce Tasks) và đưa ra kết quả cuối cùng. Mỗi task hoạt động trên một phần dữ liệu. Điều này phân phối tải trên toàn bộ cụm. Chức năng của Map Tasks là tải, phân tích, biến đổi và lọc dữ liệu. Reduce Tasks tổng hợp dữ liệu từ các kết quả ở giai đoạn Map Tasks.

Data cho công việc MapReduce phải được lưu trên HDFS. Định dạng của dữ liệu đầu vào quyết định cách chia nhỏ dữ liệu cho từng Map task. Sau đó Map task thực thi code với dữ liệu liên quan. Dữ liệu không được di chuyển qua mạng và được xử lý cục bộ.

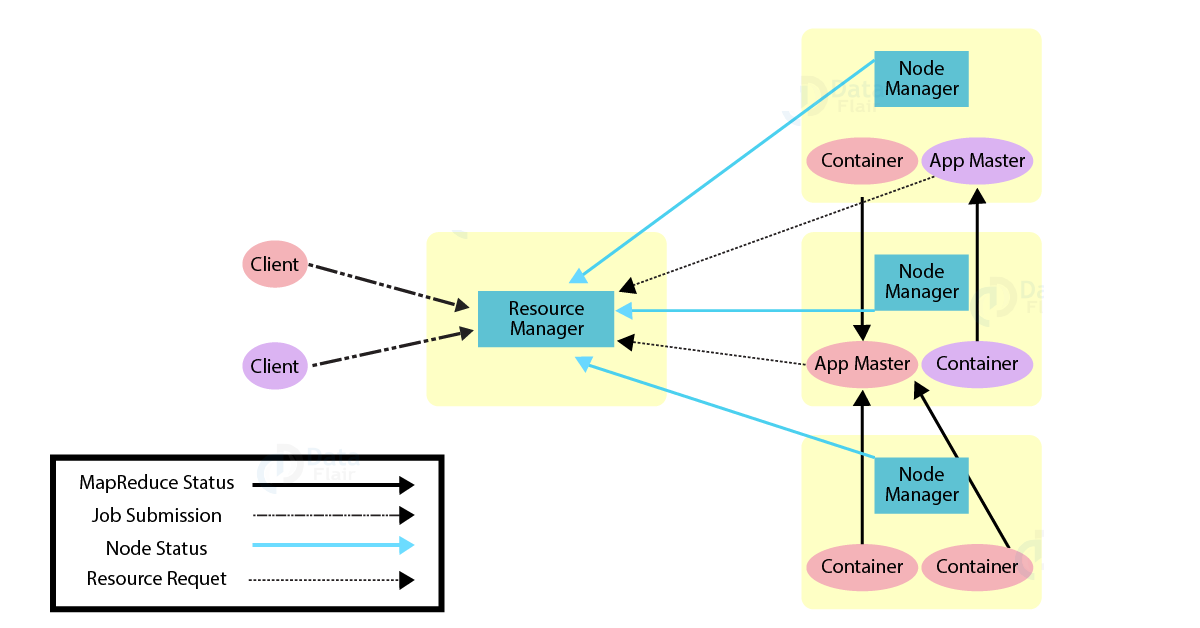


Hình 15. Sơ đồ hoạt động MapReduce.

### **3.1.3. YARN (Yet Another Resource Negotiation)**

Nói ngắn gọn, ta hiểu YARN là bộ phận quản lý tài nguyên của Hadoop. Qui tắc thiết kế của YARN rất đơn giản đó là tách quản lý tài nguyên và giám sát thành 2 chức năng riêng biệt nhưng bổ trợ cho nhau. Kiến trúc sẽ bao gồm 1 Resource Manager cho tất cả (global) và Node Manager cho từng Application Master của ứng dụng đang hoạt động.

Resource Manager có nhiệm vụ phân chia tài nguyên cho các ứng dụng. Tài nguyên ở đây là CPU, memory, disk, network,… .Node Manager sẽ quan sát tài nguyên tiêu tốn của từng ứng dụng sau đó thông báo lại cho Resource Manager. Application Master sẽ thương lượng (negotiate) với Resource Manager để lấy tài nguyên về dùng, đồng thời kết hợp với Node Manager để thực thi và theo dõi công việc.



Hình 16. Sơ đồ hoạt động của YARN.

Các đặc trưng của YARN:

* Multi – tenancy(Đa loại hệ thống đi kèm): Cho phép nhiều loại hệ thống xử lý truy cập vào dataset trên Hadoop. Các hệ thống xử lý này gồm các loại: xử lý theo lô (Batch processing), xử lý luồng (Stream processing), xử lý lặp (Iterative processing),… .
* Cluster Utilization (sử dụng với cluster): với việc phân bỏ tài nguyên động thì đây là một đặc trưng quan trọng và thiết yếu khi triển khai Hadoop ở cluster mode.
* Scalability (Khả năng mở rộng): YARN được thiết kế để tập trung vào việc phân bổ tài nguyên, nên ngay cả khi bạn mở rộng cluster – thêm 1 node dữ liệu, thực thi petabytes dữ liệu thì nó vẫn hoạt động tốt.
* Compatibility (Khả năng tương thích): MapReduce phên bản 1.x vẫn hoạt động tốt trên YARN.

### **3.1.4. Lợi thế và khó khăn khi dùng hadoop**

Lợi thế:

* Đa dạng dữ liệu: Hadoop có thể lưu nhiều loãi dữ liệu. Hay nói cách khác, bất cứ dữ liệu nào có thể chuyển đổi sang bytes thì Hadoop đều lưu được.
* Cơ chế chịu lỗi (Fault - telerance): như đã trình bày ở mục 3.1.1.3 và 3.1.1.4. Việc tạo bản sao (Replication) giúp cho phục hồi tệp bị mất dễ dàng.
* Tính sẵn sàng cao (High Availability): bên cạnh Name node sẽ luôn có các node khác như Secondary node và Standby Name node. Secondary node được xem như 1 checkpoint node, lưu lại các phiên bản của Name node theo chu kỳ. Khi Name node xảy ra lỗi thì có thể quay lại phiên bản trước (rollback) được lưu ở Secondary node. Còn khi Name node chết và không thể phục hồi, lúc này Standby Name node sẽ được sử dụng như Name node cũ với các thông tin đã được backup tại Secondary node.
* Lưu lượng thông tin trên network thấp (Low network traffic): Việc phân chia 1 task lớn thành các sub tasks nhỏ hơn cho các Data nodes khiến cho lưu lượng thông tin trên network luôn giữ ở mức thấp.
* Mã nguồn mở (Open source): miễn phí toàn quyền sử dụng.
* Tính mở rộng (Scalability): Khi lượng dữ liệu quá lớn, chúng ta có thể mở rộng dễ dàng bằng cách thêm 1 node vật lý vào cluster (được gọi là Vertical Scalability).
* Dễ sử dụng: Hadoop đã lo liệu việc sử lý song song, các nhà phát triển (developer) không cần quan tâm làm thế nào để hệ thống xử lý phân tán nữa.
* Tính tương thích (Compatibility): Hầu hết các công cụ hỗ trợ dữ liệu lớn đều tương thích với Hadoop như: Spark, Kafka, Flink,… .
* Hỗ trợ lập trình với nhiều ngôn ngữ : Các nhà phát triển phần mềm có thể lập trình với : C, C++, Python, Ruby,Groovy.

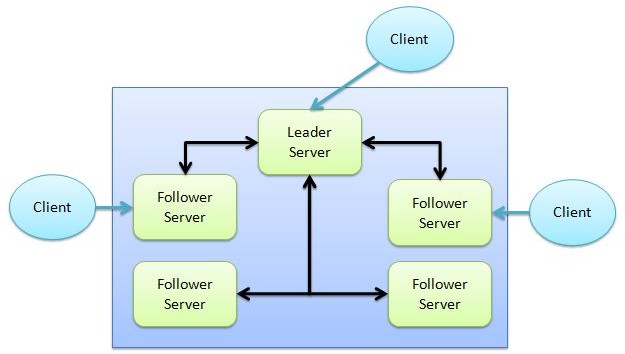
Bên cạnh đó, các khó khăn, thách thức khi dùng Hadoop cũng nhiều không kém:

* Gặp khó khăn với các dữ liệu nhỏ: Khi tệp dữ liệu quá nhỏ, bé hơn rất nhiều so với dung lượng khối (Block size) mặc định là 128MB/256MB, và số lượng quá nhiều sẽ gây ra quá tải cho Name node giống với hiện tượng đã đề cập tại mục 3.1.1.2.
* Bị ảnh hưởng bởi tác động bên ngoài của môi trường: Các điều kiện môi trường sẽ làm ảnh hưởng đến phần cứng vật lý bên ngoài. Do vậy cần có 1 bộ phận chuyên bảo trì.
* Qúa trình thực thi tốn nhiều tài nguyên: Dữ liệu trong Hadoop được thao tác trên disk, nên rất tốn tài nguyên máy khi dung lượng lên đến vài petabytes.
* Chỉ hỗ trợ thực thi lô (Batch processing): Tức chỉ có thể thao tác với dữ liệu khi chúng đã được lưu tại HDFS.

## Apache Zookeeper

Zookeeper là một dự án Apache nguồn mở cung cấp cơ sở hạ tầng và dịch vụ tập trung cho phép đồng bộ hóa trên một cụm. Nó chịu trách nhiệm duy trì đồng bộ hóa giữa các node khác nhau như cấu hình, phân cấp thư mục,… .ZooKeeper đơn giản, phân phối, đáng tin cậy và nhanh chóng.. Hãy tưởng tượng rằng bạn cần xây dựng một ứng dụng cho môi trường phân tán cho 10 đến 20 máy chủ yêu cầu dịch vụ đồng bộ hóa để bảo trì tính nhất quán dữ liệu, đó là lúc Zookeeper vào cuộc. [2]

### **3.2.1. Kiến trúc**



Hình 17. Kiến trúc ZooKeeper

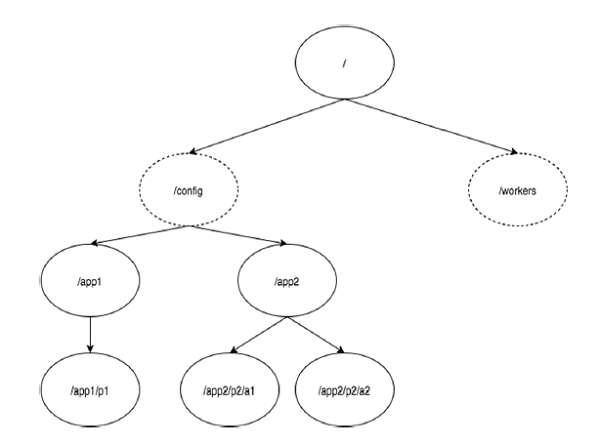
Gỉai thích các thành phần:

|  |  |
| --- | --- |
| **Part** | **Description** |
| Client | Client - ở đây là 1 node trong cụm, truy cập vào server để lấy thông tin. Client sẽ gửi thông tin lên server chỉ để báo với server rằng nó vẫn đag hoạt động lặp đi lặp lại nhiều lần theo chu kỳ.  Tương tự, khi client kết nối vào Server, server sẽ gửi lại 1 gói ACK để xác nhận server sẵn sàng nhận yêu cầu. Nếu không có phản hồi từ server hiện tại, Client sẽ được điều sang server khác. |
| Server | Server – 1 node trong cụm zooKeeper, cung cấp toàn bộ dịch vụ cho node client. Server gửi gói ACK hồi đáp cho client để xác nhận mình đang hoặc động. |
| Leader | Node chủ trong cụm ZooKeeper, có nhiệm vụ hồi phục các node Follower khi nó gặp lỗi. |
| Follower | Node tôi tớ, làm theo chỉ thị của node leader. |

### **3.2.2. Data Model**

Sơ đồ sau đây mô tả cấu trúc cây của hệ thống tệp ZooKeeper được sử dụng để biểu diễn bộ nhớ. Nút ZooKeeper được gọi là znode. Mọi znode được xác định bằng một tên và được phân tách bằng một chuỗi đường dẫn (/).

* Trong sơ đồ, đầu tiên bạn có một znode gốc được phân tách bằng “/”. Dưới gốc, bạn có hai namespace: config và worker.
* Namespace config được sử dụng để quản lý cấu hình và worker được sử dụng để đặt tên.
* Trong config node, mỗi znode có thể lưu trữ tối đa 1MB dữ liệu. Điều này tương tự với hệ thống tệp UNIX. Mục đích chính của cấu trúc này là lưu trữ dữ liệu được đồng bộ hóa và mô tả metadata của znode. Cấu trúc này được gọi là Mô hình Dữ liệu ZooKeeper.



Hình 18. Mô hình dữ liệu của ZooKeeper

Mọi znode trong mô hình dữ liệu ZooKeeper đều chứa các thông số về node đó. Nó bao gồm Version number (phiên bản), Action Control List (thao tác được cho phép trên node đó), Timestamp(mốc thay đổi) và Data length (độ lớn dữ liệu).

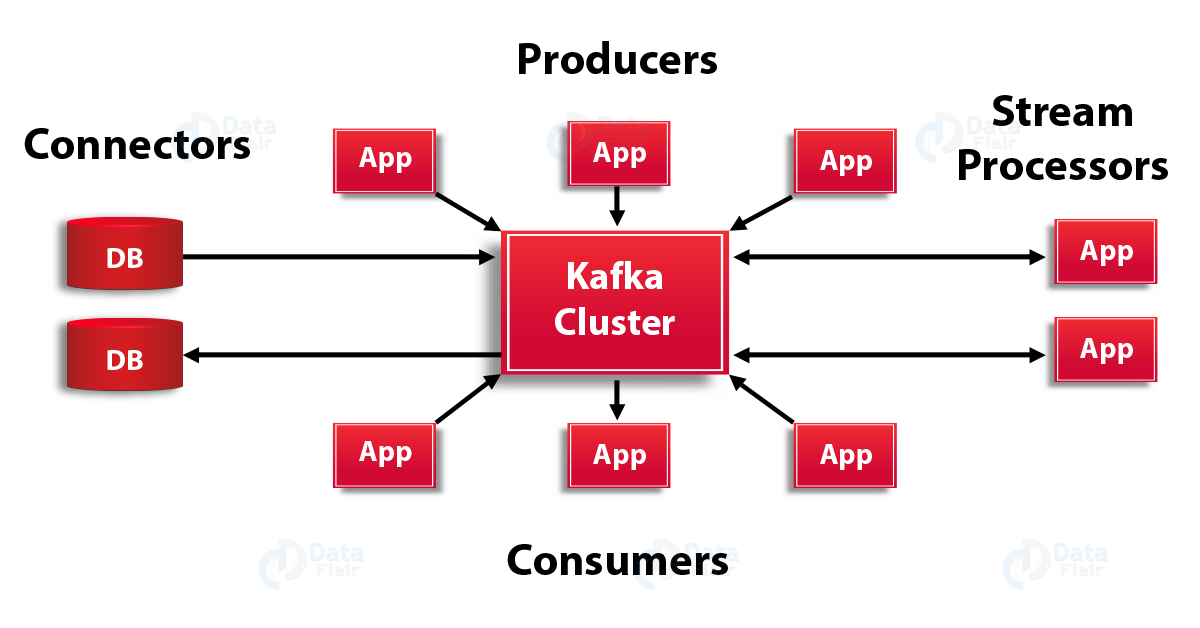
* Version number - Mỗi znode đều có một version number, có nghĩa là mỗi khi dữ liệu liên quan đến znode thay đổi, version number tương ứng của nó cũng sẽ tăng lên. Việc sử dụng version number là quan trọng khi nhiều client node đang cố gắng thực hiện các hoạt động trên cùng một znode.
* Action control List (ACL) - ACL về cơ bản là một cơ chế xác thực để truy cập znode. Nó chi phối tất cả các hoạt động đọc và ghi của znode.
* Timestamp - biểu thị các mốc thời gian từ lúc khởi tạo và sửa đổi znode. Nó thường được biểu thị bằng mili giây. ZooKeeper xác định mọi thay đổi đối với znodes từ “transaction ID” (zxid).Mỗi zxid là duy nhất và được gắn mốc thời gian cho mỗi giao dịch để có thể dễ dàng xác định thời gian trôi qua từ yêu cầu này đến yêu cầu khác.
* Data Length - Tổng lượng dữ liệu được lưu trữ trong một znode. Bạn có thể lưu trữ tối đa 1MB dữ liệu.

## Apache Kafka

Apache Kafka là hệ thống xử lý nguồn cấp dữ liệu thời gian thực có thể nhận dữ liệu từ nhiều nguồn được gọi là Producer và có thể cung cấp dữ liệu cho nhiều điểm đến được gọi là Consumer theo cách thức liên tục. [3]

### **3.3.1. Kiến trúc API**

Gồm 4 API chính: producer API, consumer API, Stream API, Connector API.



Hình 19. Kiến trúc API

Producer API: dùng để đẩy dữ liệu vào kafka topic.

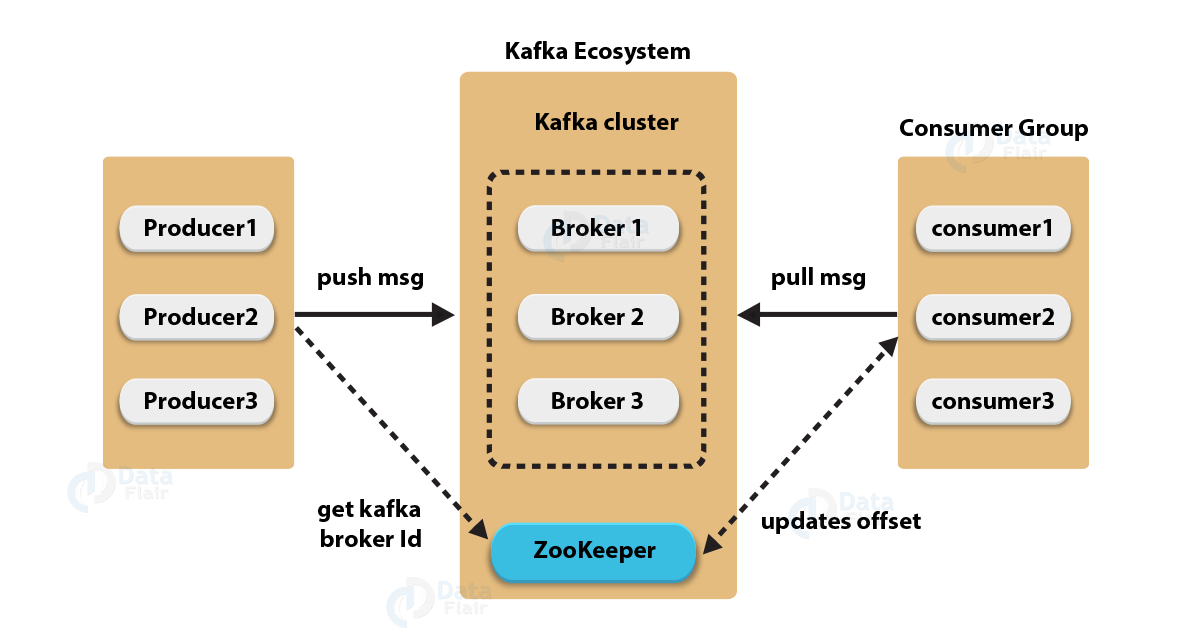
Consumer API: Cho phép ứng dụng đăng ký vào 1 hay nhiều topics và lấy dữ liệu về thao tác.

Strean API: Chi phép 1 ứng dụng có thể tiêu thụ hoặc đẩy ra dữ liệu dạng stream vào topic.

Connector API: liên kết kafka với một nguồn dữ liệu bên ngoài và quan sát sự thay đổi của nguồn dữ liệu đó.

### **3.3.2. Kiến trúc kafka cluster**

Hình bên dưới thể hiện chi tiết cấu trúc của 1 kafka cluster:



Hình 20. Kiến trúc của Kafka cluster.

Kafka Broker: Về cơ bản, để duy trì cân bằng tải, Kafka clsuter thường bao gồm nhiều broker. Tuy nhiên, chúng là không trạng thái (stateless), do đó để duy trì trạng thái cụm, kafka phải sử dụng ZooKeeper. Một Kafka Broker có thể xử lý hàng trăm nghìn lượt đọc và ghi mỗi giây. Trong khi đó mỗi broker có thể xử lý hàng TB tin nhắn mà không ảnh hưởng đế hiệu suất.

Kafka Zookeeper: Với mục đích quản lý và điều phối, Kafka broker sử dụng ZooKeeper. Ngoài ra, sử dụng Zookeeper còn để thông báo cho producer và consumer nếu có một broker mới được thêm vào hoặc có 1 hay nhiều broker nào đó gặp sự cố trong hệ thống Kafka. Ngay sau khi nhận được thông báo của Zookeeper, các producer và consumer sẽ được điều chỉnh lại kết nối.

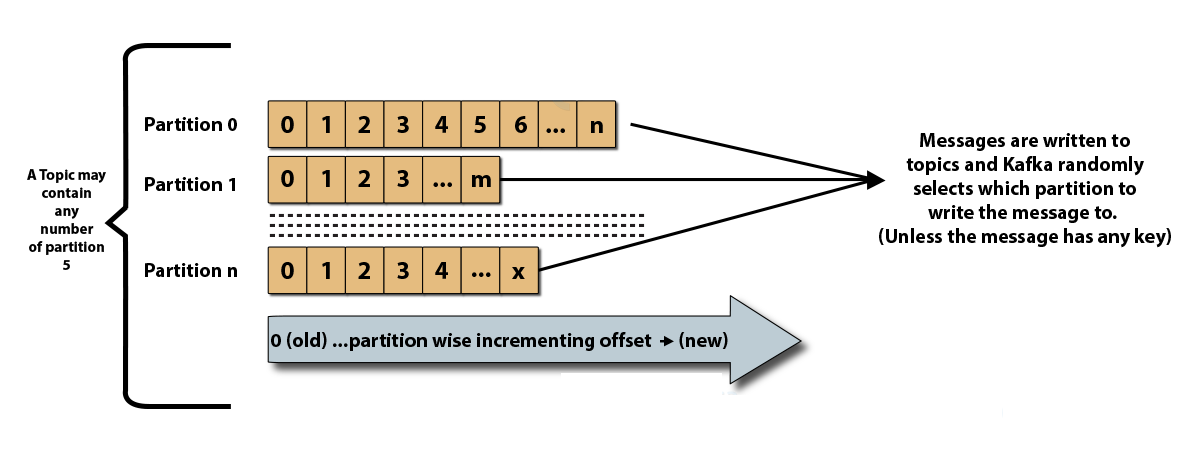
Kafka Producer: đẩy dữ liệu vào topic chứa bên trong các broker. Ngoài ra, nếu như có 1 broker mới vừa dc tạo thì các producer sẽ tìm các broker đó và tự động thêm dữ liệu của mình vào.

Kafka Consumer: đăng ký vào 1 hay nhiều topics trên các broker và tiêu thụ dữ liệu từ topic ấy.

### **3.3.4. Các khái niệm cơ bản**

**Topic**: Là một kên logic mà tại đó các producer đẩy dữ liệu vào và các consumer tiêu thụ dữ liệu.

* Một topic xác định luồng của một loại dữ liệu trong Kafka.
* Dữ liệu trong topic có cấu trúc và được tổ chức ngăn nắp. 1 loại dữ liệu chỉ được đẩy vào 1 topic có cùng loại dữ liệu đó.
* Topic được định danh bằng 1 cái tên và nó phải là độc nhất.
* Không có giới hạn cho lượng topic được tạo.
* Chỉ có thể đọc dữ liệu chứ không thể thay đổi dữ liệu một khi nó đã được đẩy vào topic

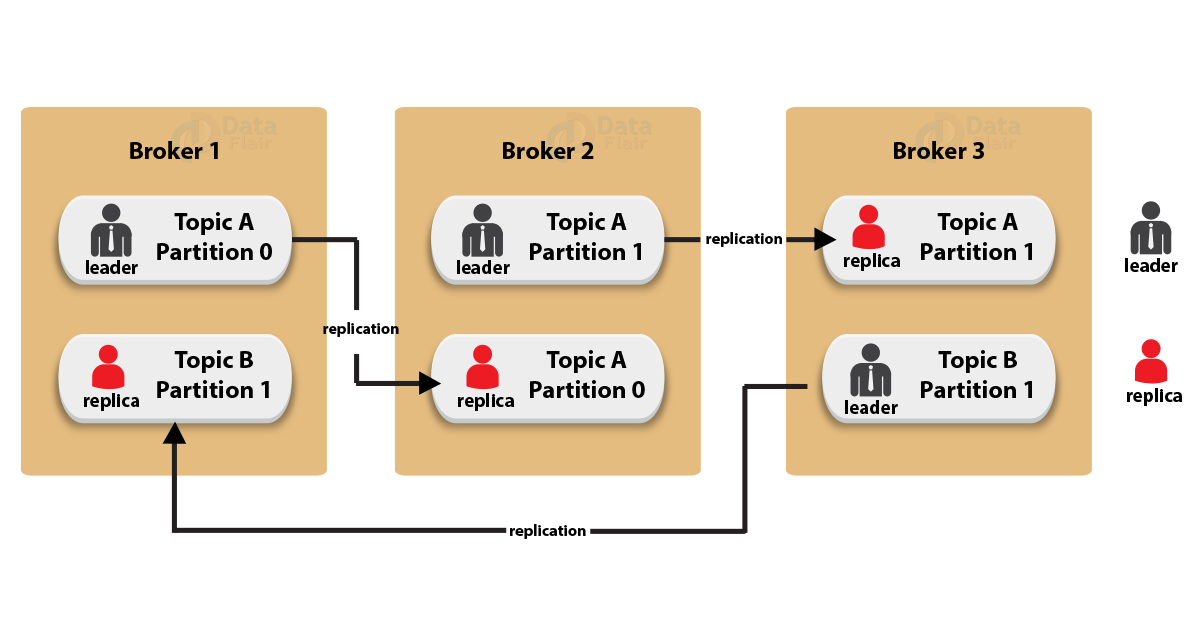


Hình 21. Mối quan hệ giữa partition và topic.

**Partitions**: Các topic được chia thành các partitions (khối) nhỏ hơn và cũng nhân bản (replicate) trên các broker khác nhau (như Hadoop Replication).

* Việc các producer đẩy data vào partition nào thì không hề biết được.
* Tuy nhiên chúng ta có thể tạo khóa cho dữ liệu đó để đảm bảo rằng các dữ liệu mà có cùng khóa sẽ được producer đẩy vào cùng một partition trong topic.
* Trong một partition, các dữ liệu sẽ được lưu theo thứ tự (như queue trong lập trình).
* Mỗi dữ liệu sẽ được đánh 1 ID, hay còn gọi là offset, theo thứ tự tăng dần.

**Topic replication factor**: Thông số điều chỉnh số lượng bản sao. Ví dụ,   
  
chúng ta có 3 broker, 3 topic và replication factor là 2. Broker1 có topic A và partition 0 vậy thỉ bản sao của nó sẽ trong Broker2. Hệ số nhân bản là 2 có nghĩa là nó sẽ có thêm một bản sao khác với bản chính. Lưu ý: chúng ta không thể điều chỉnh replication factor lớn hơn số lượng broker.

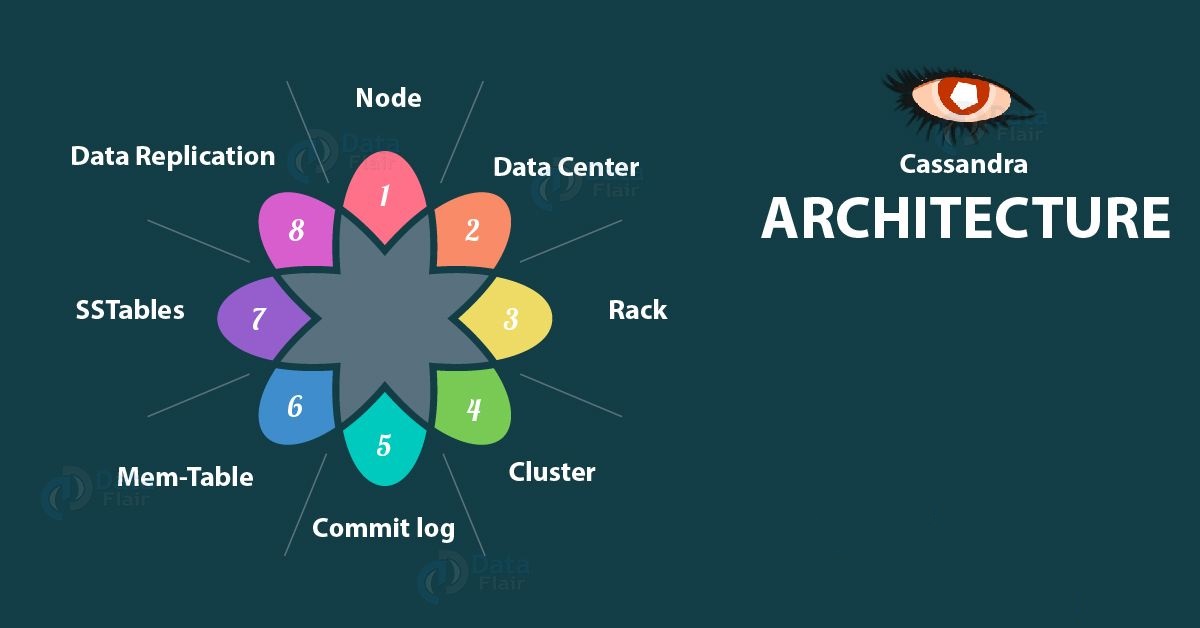


Hình 22. Quy tắc replicate của kafka partition.

## Apache Cassandra

Cassandra là một Dự án Apache mã nguồn mở được thiết kế để quản lý cơ sở dữ liệu NoQuery. Ban đầu nó được nhóm Facebook thiết kế để sử dụng nội bộ, đặc biệt là để tăng cường công cụ tìm kiếm của họ cho dữ liệu phi cấu trúc. Sau đó, nó đã được Facebook phát hành dưới dạng Dự án Apache.

Cassandra chạy trên một môi trường phân tán. Nếu bất cứ lỗi nào xảy ra trong hệ thống gì thất bại không có nghĩa là toàn bộ hoạt động cũng phải thất bại. Tổng quan kiến trúc của Cassandra: [4]



Hình 23. Kiến trúc tổng quan Cassandra.

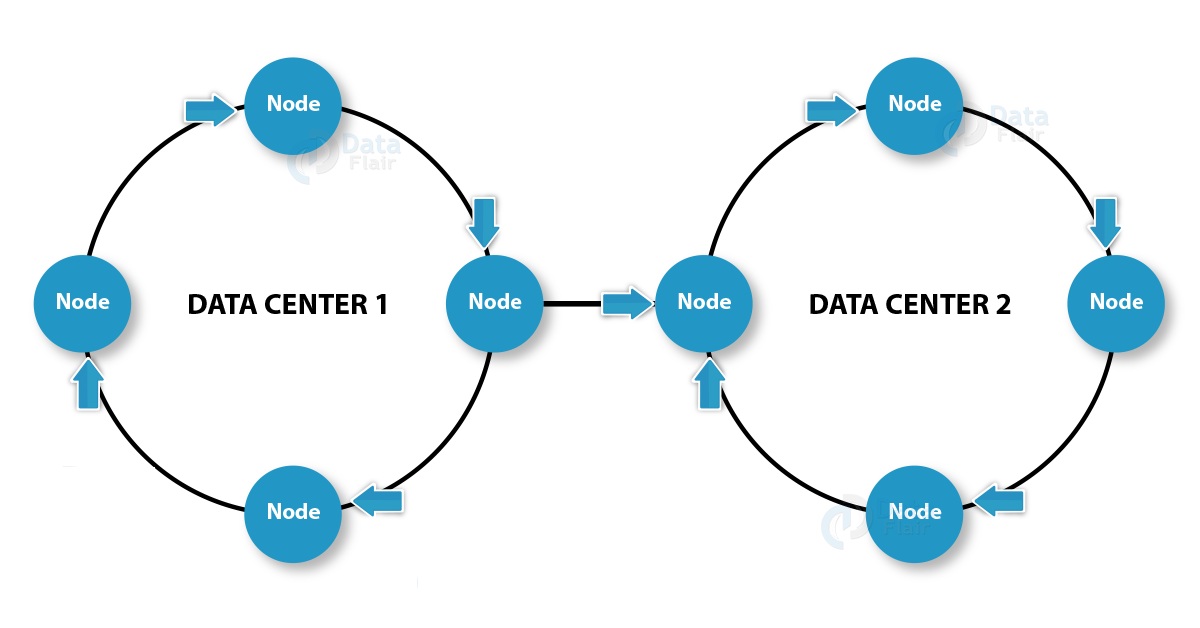
### **3.4.1. Các định nghĩa cơ bản**

**Cassandra Node**: Các node lưu trữ dữ liệu.

**Cassandra Data Center**: là một tập hợp các Cassandra node.

**Cassandra Rack**: Là khe chứa 1 tập hợp các server vật lý. Một node đượcxem là một server trong 1 Rack.

**Cassandra Cluster**: 1 cụm các Cassandra data center tạo thành 1 cassandra cluster.



Hình 24. Kiến trúc Cassandra cluster.

**Cassandra commit log**: Mọi thao tác thay đổi dữ liệu đều được ghi trong commit log để đảm bảo độ bền của dữ liệu. Nó giống như một cơ chế khôi phục sự cố vậy.

**MemTables**: Là ô nhớ tạm thời dùng để chứa dữ liệu khi cập nhật dữ liệu hoặc xóa. Dữ liệu sẽ được ghi trong memtables sau khi nó đã được ghi trong commit log. Khi dữ liệu trong memtables đầy, chúng tôi sẽ chuyển chúng vào đĩa từ SSTables

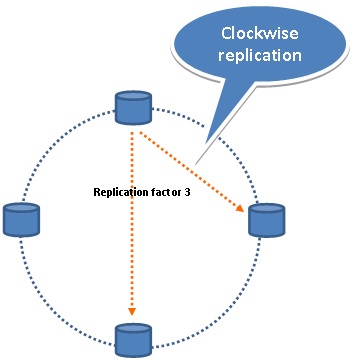
**SSTables**: Tập hợp cố định các tệp mà tại đó dữ liệu từ memtables được ghi vào theo chu kỳ

### **3.4.2. Nhân bản dữ liệu (Data Replication)**

Như chúng ta đều biết rằng để tránh single point failure, Cassandra tạo các bản sao dữ liệu trên một số nút. Ở đây, có hai điều quan trọng cần lưu ý:

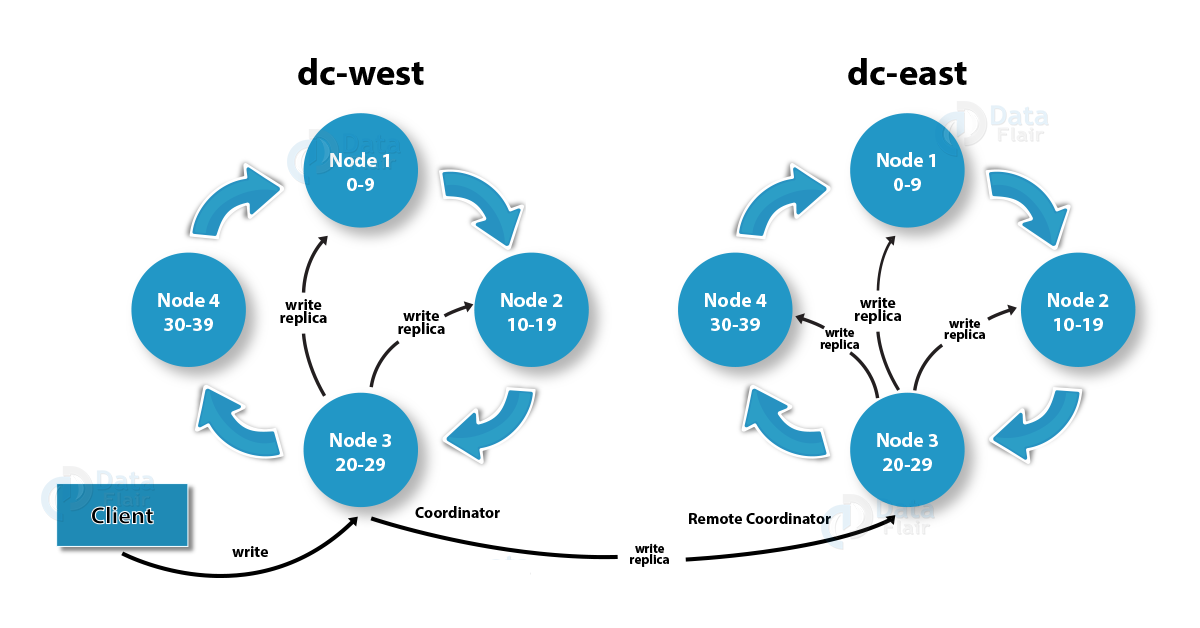
**Replication Factor**: giống với của Hadoop, là số nhân bản của dữ liệu và các nhân bản này được lưu trên node khác.

**Simple Replication Strategy**: Dữ liệu nhân bản được lưu trữ trên từng node theo chiều kim đồng hồ. Chỉ dùng khi chỉ có 1 data center.



Hình 25. Replicate theo chiều kim đồng hồ

**Network topology strategy:**Chiến thuật này dùng khi bạn cần mở rộng nhiều hơn 1 data center trong tương lai. Cassandra sẽ thực hiện replicate theo clockwise hết data center thứ nhất và sau đó chuyển sang tương tự cho data center thứ 2.



Hình 26. Cơ chế replicate theo Network topology.

## Apache Spark

Ngày nay có rất nhiều hệ thống đang sử dụng Hadoop để phân tích và xử lý dữ liệu lớn. Ưu điểm lớn nhất của Hadoop là được dựa trên một mô hình lập trình song song với xử lý dữ liệu lớn là MapReduce, mô hình này cho phép khả năng tính toán có thể mở rộng, linh hoạt, khả năng chịu lỗi, chi phí rẻ. Điều này cho phép tăng tốc thời gian xử lý các dữ liệu lớn nhằm duy trì tốc độ, giảm thời gian chờ đợi khi dữ liệu ngày càng lớn.

Dù có rất nhiều điểm mạnh về khả năng tính toán song song và khả năng chịu lỗi cao nhưng Apache Haddop có một nhược điểm là tất cả các thao tác đều phải thực hiện trên ổ đĩa cứng điều này đã làm giảm tốc độ tính toán đi gấp nhiều lần.

Để khắc phục được nhược điểm này thì Apache Spark được ra đời. Apache Spark có thể chạy nhanh hơn 10 lần so với Hadoop ở trên đĩa cứng và 100 lần khi chạy trên bộ nhớ RAM.

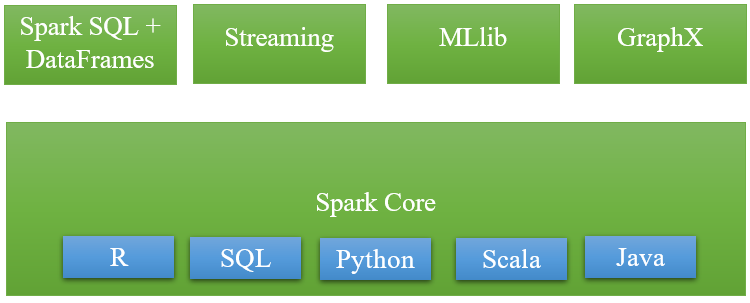
Apache Spark là một framework mã nguồn mở tính toán cụm, được phát triển sơ khởi vào năm 2009 bởi AMPLab. Sau này, Spark đã được trao cho Apache Software Foundation vào năm 2013 và được phát triển cho đến nay.

Tốc độ xử lý của Spark có được do việc tính toán được thực hiện cùng lúc trên nhiều máy khác nhau. Đồng thời việc tính toán được thực hiện ở bộ nhớ trong (in-memories) hay thực hiện hoàn toàn trên RAM.

Spark cho phép xử lý dữ liệu theo thời gian thực, vừa nhận dữ liệu từ các nguồn khác nhau đồng thời thực hiện ngay việc xử lý trên dữ liệu vừa nhận được ( Spark Streaming).

Spark không có hệ thống file của riêng mình, nó sử dụng hệ thống file khác như: HDFS, Cassandra, S3,…. Spark hỗ trợ nhiều kiểu định dạng file khác nhau (text, csv, json…) đồng thời nó hoàn toàn không phụ thuộc vào bất cứ một hệ thống file nào. [5]

### **3.5.1. Kiến trúc và thành phần của Spark**



Hình 27. Kiến trúc và thành phần của Spark.

Apache Spark gồm có 5 thành phần chính : Spark Core, Spark Streaming, Spark SQL, MLlib và GraphX, trong đó:

**Spark Core** là nền tảng cho các thành phần còn lại và các thành phần này muốn khởi chạy được thì đều phải thông qua Spark Core do Spark Core đảm nhận vai trò thực hiện công việc tính toán và xử lý trong bộ nhớ (In-memory computing) đồng thời nó cũng tham chiếu các dữ liệu được lưu trữ tại các hệ thống lưu trữ bên ngoài.

**Spark SQL** cung cấp một kiểu data abstraction mới (SchemaRDD) nhằm hỗ trợ cho cả kiểu dữ liệu có cấu trúc (structured data) và dữ liệu nửa cấu trúc (semi-structured data – thường là dữ liệu dữ liệu có cấu trúc nhưng không đồng nhất và cấu trúc của dữ liệu phụ thuộc vào chính nội dung của dữ liệu ấy). Spark SQL hỗ trợ DSL (Domain-specific language) để thực hiện các thao tác trên DataFrames bằng ngôn ngữ Scala, Java hoặc Python và nó cũng hỗ trợ cả ngôn ngữ SQL với giao diện command-line và ODBC/JDBC server.

**Spark Streaming** được sử dụng để thực hiện việc phân tích stream bằng việc coi stream là các mini-batches và thực hiệc kỹ thuật RDD transformation đối với các dữ liệu mini-batches này. Qua đó cho phép các đoạn code được viết cho xử lý batch có thể được tận dụng lại vào trong việc xử lý stream, làm cho việc phát triển lambda architecture được dễ dàng hơn. Tuy nhiên điều này lại tạo ra độ trễ trong xử lý dữ liệu (độ trễ chính bằng mini-batch duration) và do đó nhiều chuyên gia cho rằng Spark Streaming không thực sự là công cụ xử lý streaming giống như Storm hoặc Flink.

**MLlib (Machine Learning Library):** MLlib là một nền tảng học máy phân tán bên trên Spark do kiến trúc phân tán dựa trên bộ nhớ. Theo các so sánh benchmark Spark MLlib nhanh hơn 9 lần so với phiên bản chạy trên Hadoop (Apache Mahout).

**GrapX**: Grapx là nền tảng xử lý đồ thị dựa trên Spark. Nó cung cấp các Api để diễn tảcác tính toán trong đồ thị bằng cách sử dụng Pregel Api.

### **3.5.2. Những ưu điểm nổi bật của Spark**

**Xử lý dữ liệu**: Spark xử lý dữ liệu theo lô và thời gian thực

**Tính tương thích**: Có thể tích hợp với tất cả các nguồn dữ liệu và định dạng tệp được hỗ trợ bởi cụm Hadoop.

**Hỗ trợ ngôn ngữ**: hỗ trợ Java, Scala, Python và R.

**Phân tích thời gian thực**: Apache Spark có thể xử lý dữ liệu thời gian thực tức là dữ liệu đến từ các luồng sự kiện thời gian thực với tốc độ hàng triệu sự kiện mỗi giây. Ví dụ: Data Twitter chẳng hạn hoặc luợt chia sẻ, đăng bài trên Facebook. Sức mạnh Spark là khả năng xử lý luồng trực tiếp hiệu quả.

Apache Spark có thể được sử dụng để xử lý phát hiện gian lận trong khi thực hiện các giao dịch ngân hàng. Đó là bởi vì, tất cả các khoản thanh toán trực tuyến được thực hiện trong thời gian thực và chúng ta cần ngừng giao dịch gian lận trong khi quá trình thanh toán đang diễn ra.

**Mục tiêu sử dụng**:

* Xử lý dữ liệu nhanh và tương tác
* Xử lý đồ thị
* Công việc lặp đi lặp lại
* Xử lý thời gian thực
* joining Dataset
* Machine Learning

Apache Spark là Framework thực thi dữ liệu dựa trên Hadoop HDFS. Apache Spark không thay thế cho Hadoop nhưng nó là một framework ứng dụng. Apache Spark tuy ra đời sau nhưng được nhiều người biết đến hơn Apache Hadoop vì khả năng xử lý hàng loạt và thời gian thực.

### **3.5.3. So sánh giữa Spark và Hadoop MapReduce**

|  |  |  |
| --- | --- | --- |
|  | **Spark** | **MapReduce** |
| **Tốc độ** | Nhanh hơn Hadoop gấp 100 lần | Nhanh hơn các hệ thống truyền thống |
| **Sự đơn giản** | Dễ dàng sử dụng hơn hadoop | Độ phức tạp cao. |
| **Lưu trữ dữ liệu** | Trong memoy (RAM) | Trong ổ cứng (disk) |
| **Ngôn ngữ** | Scala | Java |
| **Xử lý dữ liệu** | Có thể xử lý dữ liệu theo lô / luồng / lặp / máy học | Chỉ hỗ trợ xử lý theo lô. |
| **Chịu lỗi** | Công nghệ RDD của Spark có khả năng chịu lỗi | Hadoop cũng có khả năng chịu lỗi. |
| **Máy học** | Có hỗ trợ máy học qua thư viên MLlib | Không hỗ trợ. |
| **Khả năng mở rộng** | Đã được test thử trên cụm gồm 8000 nodes. | Đã từng được test trên cụm 15000 nodes. |
| **Độ trễ** | Độ trễ thấp | Độ trễ cao. |
| **Luồng dữ liệu** | Có khả năng xử lý luồng theo thời gian thực | Chỉ xử lý dữ liệu theo lô. |
| **Lưu tạm thời** | Có khả năng lưu dữ liệu tạm thời vào bộ nhớ đệm, tăng hiệu năng. | Không hỗ trợ lưu tạm thời |
| **Chi phí** | Chi phí cao hơn Hadoop do cần nhiều RAM. | Thấp hơn do chỉ cần ổ cứng là đủ. |

Bảng 1. So sánh giữa Spark và MapReduce

## Apache Airflow

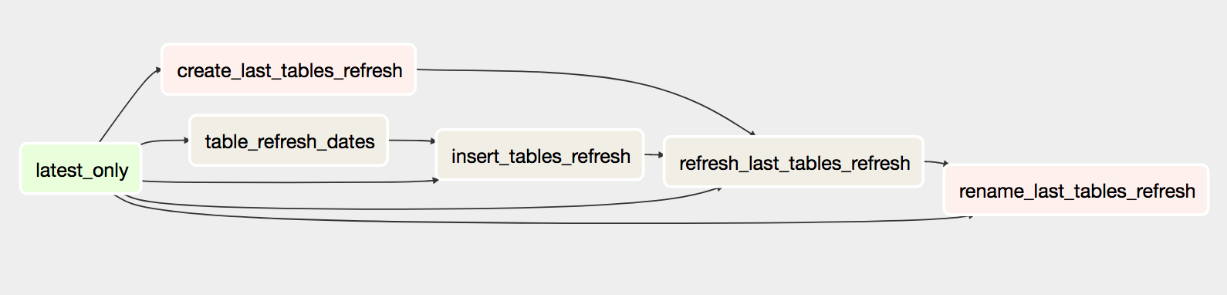
Airflow là đứa con của Airbnb. Đây là một dự án mã nguồn mở lên lịch cho các DAG. Dag là viết tắt của Directed Acyclic Graph. Về cơ bản, chúng là một tập hợp các task có tổ chức. Nhờ có giao diện người dùng đẹp mắt của Airflow, chúng ta có thể xem các DAG hiện đang hoạt động như thế nào và hoạt động của chúng ra sao. Nếu một DAG không thành công, một email sẽ được gửi cho bạn cùng với log của nó. Airflow có thể được kích hoạt lại theo cách thủ công thông qua giao diện người dùng. DAGS có thể kết hợp nhiều loại tác vụ khác nhau (bash, python, sql…) và tương tác với các nguồn dữ liệu khác nhau. [6]

### **3.6.1. Các component của Airflow**

**Webserver**: Cho phép người dùng tương tác với các workflow và xem trạng thái của các workflow đó. Mỗi workflow được tạo từ nhiều DAGs. Webserver còn cho phép chúng ta xóa, tạm ngưng hay tiếp tục một DAG. Khi webserver được khởi động, nó sẽ khởi động các worker để xử lý các yêu cầu khác nhau song song.

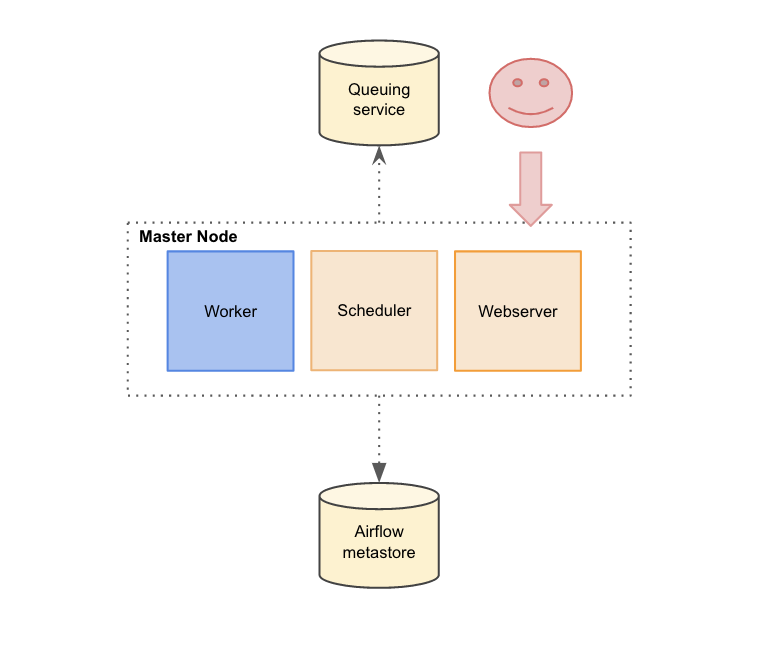
**Scheduler**: được gọi là bộ lập lịch, giám sát các DAG. scheduler giám sát và đồng bộ hóa với một thư mục cho tất cả các đối tượng DAG, đồng thời kiểm tra định kỳ các tác vụ để xem liệu chúng có thể được kích hoạt hay không.

**Worker**: Các đơn vị để thực thi task.



Hình 28. Mô hình workflow bao gồm 6 nút DAGs.

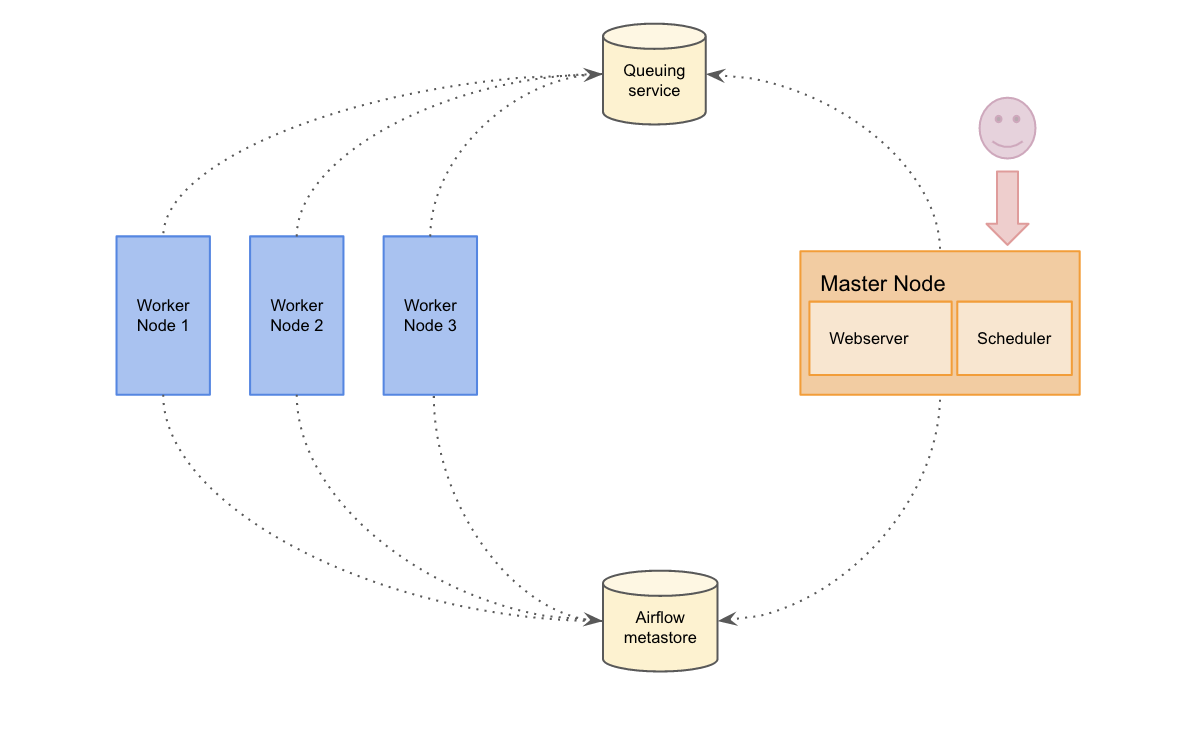
### **3.6.2. Kiến trúc đơn node.**



Hình 29. Kiến trúc đơn node.

Kiến trúc đơn node được người dùng sử dụng rộng rãi trong trường hợp họ có một lượng DAGs vừa phải. Trong chế độ này, worker kéo các tác vụ để chạy từ hàng đợi IPC (Inter Process Communication). Chế độ này không cần bất kỳ phụ thuộc bên ngoài nào. Giải pháp này hoạt động khá tốt. Tuy nhiên, để mở rộng quy mô đến nhiều máy chủ, chế độ thực thi Celery phải được sử dụng. Người thực thi Celery sử dụng Celery (và một máy chủ xếp hàng đợi tin nhắn) để phân phối tải cho một nhóm worker.

### **3.6.3. Kiến trúc đa node**



Hình 30. Kiến trúc đa node

Ở kiến trúc này, các worker sẽ được tách biệt ra nằm trên 1 máy riêng tương ứng. Để triển hay ở chế độ này, Airflow phải được cấu hình với Celery (một hàng đợi không đồngbộ).

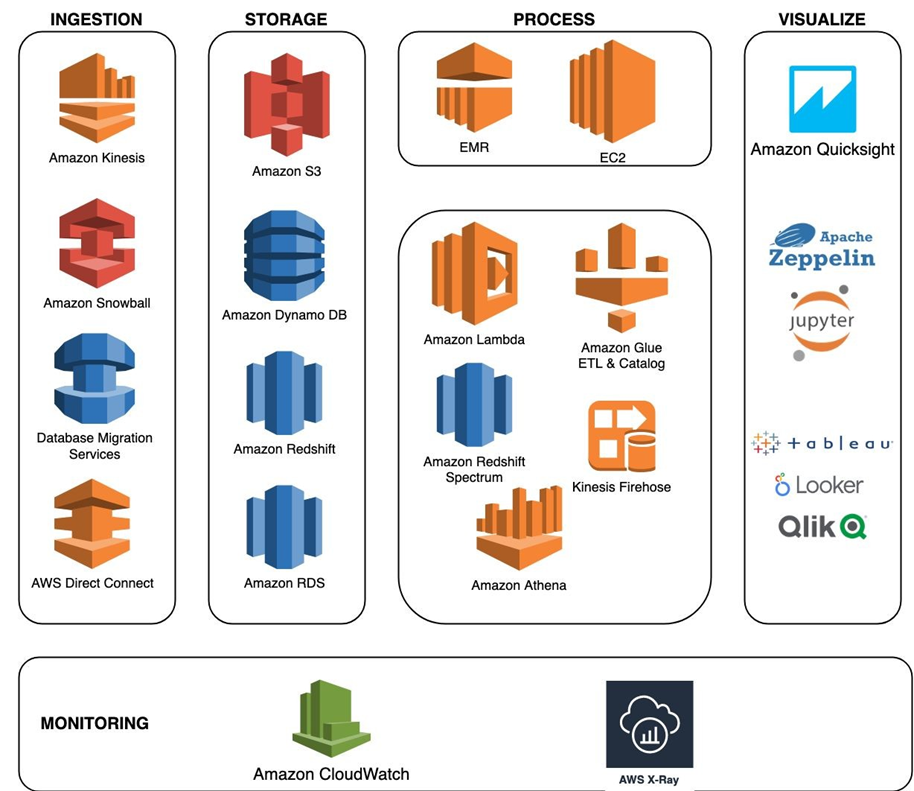
Trong chế độ này, phần backend Celery phải được cài đặt. Celery là một hàng đợi không đồng bộ dựa trên việc truyền thông điệp phân tán. Airflow sử dụng nó để thực hiện một số tác vụ đồng thời trên các worker bằng cách sử dụng đa xử lý. Chế độ này cho phép mở rộng cụm dễ dàng.

Các lợi ích khi triển khai ở đa node thay vì đơn node:

* Tính khả dụng cao hơn: nếu một trong các worker gặp sự cố, cụm sẽ vẫn hoạt động và các DAG sẽ vẫn chạy.
* Worker chuyên dụng cho các workflow cụ thể: Có thể có 1 workflow máy học cần nhiều về GPU.
* Mở lệ theo chiều ngang: Mở rộng quy mô cụm dễ dàng bằng cách thêm worker node. Các worker có thể được bật và tắt mà không có bất kỳ thời gian chết nào trên cụm.

# Chương 4. TỔNG QUAN VỀ CÁC HỆ THỐNG BIG DATA TRÊN CLOUD - AMAZON

Amazon Web Services cung cấp hàng loạt sản phẩm dựa trên nền tảng đám mây trên phạm vi toàn cầu gồm có điện toán, lưu trữ, cơ sở dữ liệu, phân tích, kết nối mạng, di động, công cụ dành cho nhà phát triển, công cụ quản lý, IoT, bảo mật và ứng dụng doanh nghiệp. Những dịch vụ này giúp các doanh nghiệp phát triển nhanh hơn, giảm chi phí CNTT và mở rộng quy mô. Những sản phẩm AWS mà các doanh nghiệp lớn nhất cũng như các công ty khởi nghiệp nổi tiếng nhất tin dùng để hỗ trợ khối lượng công việc đa dạng của họ bao gồm: các ứng dụng trên web và thiết bị di động, phát triển trò chơi, xử lý và lưu kho dữ liệu, lưu trữ, sao lưu và nhiều dịch vụ khác.



Hình 31. Tổng thể kiến trúc hệ thống và các thành phần dịch vụ amazon.

## Amazon S3

Amazon Simple Storage Service (Amazon S3) là một dịch vụ lưu trữ đối tượng cung cấp khả năng thay đổi theo quy mô, tính khả dụng của dữ liệu, bảo mật và hiệu năng hàng đầu trong lĩnh vực. Điều này có nghĩa là khách hàng thuộc mọi quy mô và lĩnh vực có thể sử dụng dịch vụ này để lưu trữ và bảo vệ bất kỳ lượng dữ liệu nào cho nhiều trường hợp sử dụng khác nhau, chẳng hạn như trang web, ứng dụng di động, sao lưu và khôi phục, lưu trữ, ứng dụng doanh nghiệp, thiết bị IoT và phân tích dữ liệu hớn.

Amazon S3 cung cấp các tính năng quản lý dễ sử dụng, nhờ đó, bạn có thể tổ chức dữ liệu và cấu hình các kiểm soát truy cập được tinh chỉnh để đáp ứng yêu cầu cụ thể của doanh nghiệp, tổ chức và yêu cầu về tuân thủ. Amazon S3 được thiết kế để đảm bảo độ bền 99,999999999% (11,9’s downtime) và lưu trữ dữ liệu của hàng triệu ứng dụng cho các công ty trên toàn thế giới. [7]

### **4.1.1. Kiến trúc và các khái niệm cơ bản**

#### **4.1.2.1. Bucket**



Hình 32. Kiến trúc Amazon s3

Dữ liệu trên Amazon S3 được hệ thống hoá theo khái niệm **Bucket**:

Amazon S3 lưu trữ dữ liệu như một Object.

* Một Bucket là một đơn vị lưu trữ logical trên S3.
* Một Bucket sẽ chứa các object, trong đó Object sẽ chứa dữ liệu (data) và metadata miêu tả về dữ liệu đó.
* Bạn có thể chỉ định vị trí địa lý khu vực mà Bucket của bạn tạo ra sẽ được lưu trữ ở Amazon S3 khu vực đó.
* Mỗi bucket sẽ có một cái tên toàn cục (global) độc nhất. Điều đó có nghĩa nếu tên bucket đã được xài thì không tài khoản AWS nào ở bất kì khu vực nào có thể đặt lại tên đó.

Như thế thì một bucket giống như một container cho các object lưu trữ trên Amazon S3. Mỗi object sẽ được chứa trong một bucket.

**Ví dụ** :

– Nếu object tên ‘photos/puppy.jpg’ được chứa trong bucket tên ‘16520021’, thì lúc này dữ liệu của bạn có thể được truy xuất qua URL :

http://16520021.s3.amazonaws.com/photos/puppy.jpg

Vậy bucket sinh ra dùng cho một số mục đích như:

* Quản lý Amazon S3 namepsace.
* Xác định được tài khoản nào chịu trách nhiệm lưu trữ dữ liệu và chuyển dữ liệu để tính phí sử dụng
* Xác định được các quyền kiểm soát truy cập.
* Tiện lợi trong việc báo cáo sử dụng.

#### **4.1.2.2. Key**

Một **key** là một chuỗi định danh ID cho một Object trong một Bucket. Mỗi Object trong bucket sẽ có chính xác một key. Mỗi Object trong Amazon S3 sẽ có một địa chỉ truy cập độc nhất bằng cách kết hợp giữa các thông tin như web service enpoint, tên bucket, key, version.

**Ví dụ**

* **URL truy cập Object ‘AmazonS3.wsdl’** : http://doc.s3.amazonaws.com/2020-09-08/AmazonS3.wsdl
* **Tên bucket** : doc
* **Key** : 2020-09-08/AmazonS3.wsdl

#### **4.1.2.3. Object**

Object (đối tượng) là khái niệm cơ bản về một thực thể dữ liệu được lưu trữ trong Amazon S3. Các object sẽ bao gồm object dữ liệu (object data) và metadata. Về phần object dữ liệu thì S3 chỉ lưu trữ không quan tâm dữ liệu đó là gì. Còn lại thì phải có “metadata” là dữ liệu thông tin theo kiểu ‘name-value’ dùng để miêu tả về object ví dụ như : ngày chỉnh sửa, loại dữ liệu,…. Mỗi một Object sẽ có một ID độc nhất nằm trong một bucket.

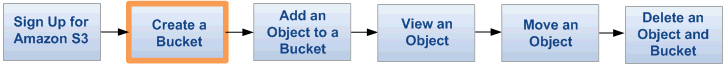
#### **4.1.2.4. Region**

Bạn có thể chọn khu vực địa lý mà Amazon S3 sẽ chứa các bucket mà bạn khởi tạo. Điều này có ích cho việc tối ưu hoá về độ trễ, chi phí … Object khi lưu ở một khu vực cụ thể sẽ không bao mất cho đến khi bạn xoá hoặc chuyển các Object đó sang khu vực khác.

Bạn cần cân nhắc 4 yếu tố sẽ giúp bạn chọn ra được khu vực (region) Amazon S3 sẽ lưu trữ giúp tối ưu và hiệu quả, đó là:

* Pricing (Giá)
* User/Customer Location (Nơi khách hàng sử dụng dịch vụ)
* Latency (Độ trễ)
* Service Availability (Tính khả dụng dịch vụ)

### **4.1.2. Quy trình xử dụng Amazon S3 cơ bản**



Hình 33. Quy trình sử dụng amazon s3 cơ bản

* **Bước 1**: tạo một bucket
* **Bước 2**: thêm một object vào bucket
* **Bước 3**: xem thông tin về object
* **Bước 4**: nếu muốn di chuyển object thì thực hiện thao tác object
* **Bước 5**: nếu muốn xoá object và bucket thì thực hiện thao tác xoá object/bucket.

### **4.1.3. Ưu điểm của Amazon**

**Amazon S3** được thiết kế các tính năng tối giản nhằm đạt được mục tiêu đơn giản và nhanh gọn. Sau đây là một số ưu điểm của Amazon S3:

* **Tạo Buckets**: tạo và đặt tên một Bucket sẽ chứa dữ liệu. Bucket là khái niệm cơ bản trong Amazon S3 dành cho việc lưu trữ dữ liệu.
* **Lưu trữ dữ liệu trong Bucket** : bạn có thể lưu trữ vô hạn các loại dữ liệu khác nhau trong một bucket. Mỗi dữ liệu của bạn sẽ như là một Object và muốn up nhiêu cũng được. Mỗi object có thể chứa tới 5TB dữ liệu.
* **Tải dữ liệu :** bạn có thể tải dữ liệu của bạn về bất cứ lúc nào, thậm chí là cho phép người khác tải dữ liệu của bạn đang nằm trong Amazon S3 Bucket nữa cơ.
* **Phân quyền** : phân quyền cho phép hoặc từ chối quyền hạn upload/download đối với dữ liệu nằm trong Amazon S3 Bucket của bạn.
* **Giao diện tương tác tiêu chuẩn** : bạn có thể sử dụng REST hoặc SOAP để thiết kế tương tác từ ứng dụng của bạn đến Amazon S3 qua các công cụ lập trình phát triển.
* **Tính ổn định**: Nó được thiết kế để chịu được các hỏng hóc và phục hồi hệ thống rất nhanh với thời gian tối thiểu. Amazon cung cấp một thỏa thuận cấp dịch vụ (service-level agreement – SLA) để duy trì tính sẵn sàng ở mức 99.99 phần trăm.
* **Đơn giản**: Dễ dùng S3 được xây dụng trên các khái niệm đơn giản và cung cấp tính mềm dẻo cao cho việc phát triển các ứng dụng của bạn. Bạn có thể xây dựng các lược đồ lưu trữ phức tạp hơn, nếu cần, bằng cách thêm các hàm vào các thành phần của S3.
* **Tính mở rộng**: Thiết kế của S3 cung cấp một cấp độ cao về tính mở rộng và cho phép sự điều chỉnh dễ dàng trong dịch vụ khi lượng truy cập vào ứng dụng web của bạn tăng đột biến với lưu lượng khổng lồ.
* **Rất Rẻ**: Chi phí sử dụng S3 rất cạnh tranh với các giải pháp của công ty và cá nhân khác trên thị trường.

### **4.1.4. Lợi ích của amazon S3**

* **Hiệu năng, khả năng thay đổi theo quy mô, tính khả dụng và độ bền hàng đầu trong lĩnh vực**

Tăng và giảm quy mô tài nguyên lưu trữ của bạn để đáp ứng nhu cầu hay thay đổi, mà không cần các khoản đầu tư trả trước hay chu kỳ thu mua tài nguyên. Amazon S3 được thiết kế để đảm bảo độ bền dữ liệu 99,999999999% (11 9’s) vì dịch vụ này tự động tạo và lưu trữ bản sao của tất cả đối tượng S3 trên nhiều hệ thống. Điều này có nghĩa là dữ liệu của bạn khả dụng khi cần và được bảo vệ khỏi các sự cố, lỗi và mối đe dọa.

* **Nhiều lớp lưu trữ tiết kiệm chi phí**

Tiết kiệm chi phí mà không phải hi sinh hiệu suất bằng cách lưu trữ dữ liệu trên Các lớp lưu trữ S3 hỗ trợ các cấp độ truy cập dữ liệu khác nhau ở mức giá tương ứng. Bạn có thể sử dụng Phân tích lớp lưu trữ S3 để khám phá dữ liệu cần di chuyển sang lớp lưu trữ có chi phí thấp hơn dựa trên các cấu trúc truy cập và cấu hình một chính sách Vòng đời S3 để thực thi việc truyền dữ liệu. Bạn cũng có thể lưu trữ dữ liệu với các cấu trúc truy cập không xác định hoặc hay thay đổi trong S3 Thông minh-Phân bậc, dịch vụ này phân bậc đối tượng dựa trên các cấu trúc truy cập hay thay đổi và tự động tiết kiệm chi phí.

* **Bảo mật, tính tuân thủ và khả năng kiểm tra chưa từng có**

Lưu trữ dữ liệu của bạn trong Amazon S3 và ngăn chặn truy cập dữ liệu trái phép với các tính năng mã hóa và công cụ quản lý truy cập. Bạn cũng có thể sử dụng Amazon Macie để xác định dữ liệu nhạy cảm trong bộ chứa S3 của mình và phát hiện yêu cầu truy cập bất thường. Amazon S3 duy trì các chương trình tuân thủ, chẳng hạn như PCI-DSS, HIPAA/HITECH, FedRAMP, Chỉ thị bảo vệ dữ liệu của Liên minh châu Âu và FISMA, để giúp bạn đáp ứng các yêu cầu về quy định. AWS cũng hỗ trợ nhiều khả năng kiểm tra để giám sát các yêu cầu truy cập vào tài nguyên S3 của bạn.

* **Các công cụ quản lý dùng để kiểm soát dữ liệu chi tiết**

Phân loại, quản lý và báo cáo về dữ liệu của bạn bằng các tính năng như: Phân tích lớp lưu trữ S3 để phân tích các cấu trúc truy cập; các chính sách Vòng đời S3 để chuyển đối tượng sang lớp lưu trữ có chi phí thấp hơn; Sao chép liên khu vực S3 để sao chép dữ liệu vào khu vực khác; Khóa đối tượng S3 để áp dụng ngày duy trì đối với ứng dụng và bảo vệ để ứng dụng không bị xóa; và Kho S3 để tìm hiểu về đối tượng được lưu trữ của bạn, siêu dữ liệu và trạng thái mã hóa của các đối tượng đó. Bạn cũng có thể sử dụng S3 Batch Operations để thay đổi thuộc tính đối tượng và thực hiện tác vụ quản lý lưu trữ cho hàng tỷ đối tượng. Do Amazon S3 có liên kết với AWS Lambda, bạn có thể ghi nhật ký hoạt động, định nghĩa cảnh báo và tự động hóa luồng công việc mà không phải quản lý thêm cơ sở hạ tầng nào.

* **Dịch vụ truy vấn tại chỗ cho phân tích**

Chạy phân tích dữ liệu lớn trên các đối tượng S3 của bạn (và các tập hợp dữ liệu khác trong AWS) với dịch vụ truy vấn tại chỗ của chúng tôi. Sử dụng Amazon Athena để truy vấn dữ liệu S3 với các biểu thức SQL tiêu chuẩn và Amazon Redshift Spectrum để phân tích dữ liệu được lưu trữ trên kho lưu trữ dữ liệu AWS và tài nguyên S3 của bạn. Bạn cũng có thể sử dụng S3 Select để truy xuất các tập hợp nhỏ siêu dữ liệu đối tượng thay vì toàn bộ đối tượng và tăng tới 400% hiệu suất truy vấn.

* **Dịch vụ lưu trữ đám mây được hỗ trợ nhiều nhất**

Lưu trữ và bảo vệ dữ liệu của bạn trong Amazon S3 bằng cách làm việc với đối tác từ Mạng lưới đối tác AWS (APN) — cộng đồng nhà cung cấp dịch vụ tư vấn và công nghệ lớn nhất. APN nhận diện các đối tác di chuyển truyền dữ liệu sang Amazon S3 và đối tác lưu trữ cung cấp giải pháp tích hợp S3 để phục vụ lưu trữ chính, sao lưu và khôi phục, lưu trữ và khôi phục sau thảm họa. Bạn cũng có thể mua giải pháp tích hợp AWS trực tiếp từ AWS Marketplace, tại đây có hơn 250 dịch vụ dành riêng cho lưu trữ.

### **4.1.5. Các trường hợp nên dùng Amazon S3**

#### **4.1.5.1. Sao lưu và phục hồi**

Xây dựng các giải pháp sao lưu và khôi phục có thể thay đổi theo quy mô, có độ bền và bảo mật với **Amazon S3** và các dịch vụ AWS khác, chẳng hạn như **S3 Glacier**, Amazon EFS và Amazon EBS, để tăng cường hoặc thay thế các khả năng tại chỗ hiện có. Các đối tác AWS và APN có thể giúp bạn đáp ứng Thời gian khôi phục (Recovery Time Objectives – RTO), Thời điểm khôi phục (Recovery Point Objectives – RPO) và các yêu cầu về tuân thủ. Với AWS, bạn có thể sao lưu dữ liệu đã có trong Đám mây AWS hoặc sử dụng **AWS Storage Gateway**, một dịch vụ lưu trữ kết hợp để gửi các bản sao lưu dữ liệu tại chỗ sang AWS.

#### **4.1.5.2. Khôi phục sau thảm họa (DR)**

Bảo vệ dữ liệu, ứng dụng và hệ thống CNTT quan trọng đang chạy trong Đám mây AWS hoặc trong môi trường tại chỗ của bạn mà không phải chịu phí tổn của một cơ sở vật lý phụ. Với lưu trữ Amazon S3, **Sao chép liên khu vực S3**, và các dịch vụ cơ sở dữ liệu, mạng và điện toán AWS khác, bạn có thể tạo các kiến trúc DR để khôi phục nhanh chóng và dễ dàng sau khi mất điện do thảm họa thiên nhiên, sự cố hệ thống và lỗi của con người.

#### **4.1.5.3. Lưu trữ**

Giải phóng cơ sở hạ tầng vật lý và lưu trữ dữ liệu với **S3 Glacier** và **S3 Glacier Deep Archive**. Các lớp lưu trữ S3 này duy trì các đối tượng lâu dài với mức giá thấp nhất. Chỉ cần tạo một chính sách S3 Vòng đời để lưu trữ đối tượng trong toàn bộ vòng đời hoặc tải đối tượng trực tiếp lên các lớp lưu trữ. Với **S3 Object Lock**, bạn có thể áp dụng ngày duy trì với đối tượng để ngăn đối tượng bị xóa và đáp ứng các yêu cầu về tuân thủ. Không giống các thư viện băng từ, S3 Glacier cho phép bạn khôi phục đối tượng đã lưu trữ chỉ trong một phút đối với truy xuất nhanh và 3-5 giờ đối với truy xuất tiêu chuẩn. Quá trình khôi phục khối lượng dữ liệu lớn từ S3 Glacier và khôi phục toàn bộ từ**S3 Glacier Deep Archive** hoàn tất trong vòng 12 giờ.

#### **4.1.5.4. Kho dữ liệu và phân tích dữ liệu lớn**

Đẩy nhanh quá trình đổi mới bằng cách tạo một kho dữ liệu trong Amazon S3 và trích xuất thông tin chi tiết quan trọng bằng các công cụ machine learning, phân tích và truy vấn tại chỗ. Bạn cũng có thể sử dụng**AWS Lake Formation** để tạo nhanh một kho dữ liệu và tập trung xác định cũng như thực thi các chính sách bảo mật, quản trị và kiểm tra. Dịch vụ này thu thập dữ liệu ở các cơ sở dữ liệu và tài nguyên S3 của bạn, di chuyển dữ liệu vào một kho dữ liệu mới trong Amazon S3 và dọn dẹp cũng như phân loại dữ liệu bằng các thuật toán machine learning. Mọi tài nguyên AWS đều có thể được tăng quy mô để đáp ứng các kho dữ liệu mở rộng của bạn mà không cần đầu tư trước.

#### **4.1.5.5. Lưu trữ đám mây lai**

Tạo một kết nối liền mạch giữa ứng dụng tại chỗ và Amazon S3 với **AWS Storage Gateway** để giảm phạm vi của trung tâm dữ liệu và sử dụng quy mô, độ tin cậy và độ bền của AWS, cũng như các khả năng phân tích và machine learning đổi mới của AWS. Bạn cũng có thể tự động hóa quá trình truyền dữ liệu giữa lưu trữ tại chỗ và Amazon S3 bằng **AWS DataSync**, dịch vụ này có thể truyền dữ liệu ở tốc độ nhanh hơn tới 10 lần so với các công cụ nguồn mở. Một cách khác để hỗ trợ môi trường lưu trữ đám mây lai là làm việc với nhà cung cấp cổng từ APN. Bạn cũng có thể truyền tệp trực tiếp vào hoặc ra khỏi Amazon S3 với**AWS Transfer for SFTP** — một dịch vụ được quản lý đầy đủ, cho phép trao đổi tệp an toàn với các bên thứ ba.

#### **4.1.5.6. Dữ liệu ứng dụng dành cho đám mây**

Xây dựng các ứng dụng trên Internet và di động hiệu quả về chi phí, có tốc độ nhanh bằng cách sử dụng Dịch vụ AWS và Amazon S3 để lưu trữ dữ liệu sản xuất. Với Amazon S3, bạn có thể tải lên bất kỳ khối lượng dữ liệu nào và truy cập dữ liệu ở mọi nơi để triển khai ứng dụng nhanh hơn và tiếp cận nhiều người dùng cuối hơn. Việc lưu trữ dữ liệu trong Amazon S3 cũng có nghĩa là bạn có quyền truy cập các công cụ, dịch vụ mới nhất về machine learning và phân tích dành cho nhà phát triển AWS để đổi mới và tối ưu hóa ứng dụng dành cho đám mây của bạn.

### **4.1.6. Các công ty điển hình sử dụng AWS S3**

* **Netflix**

Netflix phân phối hàng tỷ giờ nội dung đến khách hàng trên khắp thế giới từ Amazon S3. Amazon S3 cũng đóng vai trò là kho dữ liệu cho giải pháp phân tích dữ liệu lớn của họ.

* **Finra**

FINRA sử dụng Amazon S3 để thu nạp và lưu trữ dữ liệu cho hơn 75 tỷ sự kiện thị trường hàng ngày và các hàm AWS Lambda để định dạng và xác thực dữ liệu dựa trên 200 quy tắc.

* **Airbnb**

Airbnb để dữ liệu sao lưu và tập tin tĩnh trên Amazon S3, trong đó có trên 10 petabyte hình ảnh người dùng. Là giải pháp được sinh ra trên đám mây, họ liên tục sáng tạo ra nhiều cách thức mới để phân tích dữ liệu được lưu trữ trên Amazon S3.

* **GE Healthcare**

GE sử dụng Amazon S3 để lưu trữ và bảo vệ một petabyte dữ liệu hình ảnh y tế quan trọng cho dịch vụ Đám mây sức khỏe GE, kết nối hàng trăm nghìn máy chụp và các thiết bị y tế khác.

### **4.1.7. Các loại dữ liệu amazon có thể lưu trữ**

Với bất kì loại dữ liệu nào và format như thế nào đi chăng nữa, Amazon S3 cũng có thể lưu trữ được hết. Số lượng đối tượng (object) mà Amazon S3 có thể lưu trữ gần như là không có giới hạn, bao gồm dữ liệu, key và metadata.

Tuy không phân biệt kiểu dữ liệu, nhưng Amazon S3 sẽ phân loại các dữ liệu như sau :

* Dữ liệu được truy cập thường xuyên, tần suất cao.
* Dữ liệu không được truy cập thường xuyên, tần suất thấp.

Khi dựa vào cách phân loại dữ liệu như trên thì Amazon sẽ cung cấp nhiều loại hình lưu trữ (AWS S3 Storage Classes) phù hợp cho bạn . Tiêu biểu thì có 3 loại hình lưu trữ :

* **Standard** : dành cho dữ liệu thường xuyên truy cập, mang lại hiệu suất truy cập tốt hơn và độ trễ thấp hơn. Đây là loại hình lưu trữ mặc định của Amazon S3 khi bạn sử dụng.
* **Standard\_IA** : dành cho dữ liệu ít truy cập. Thường dùng cho các hoạt động backup/restore dữ liệu ứng dụng.
* **Glacier** : Glacier là một dịch vụ lưu trữ chi phí cực thấp, cung cấp lưu trữ lâu bền với các tính năng bảo mật để lưu trữ và sao lưu dữ liệu. Với Glacier, khách hàng có thể lưu trữ chi phí dữ liệu của họ một cách hiệu quả trong nhiều tháng, nhiều năm hoặc thậm chí nhiều thập kỷ.

## Amazon EC2

### **4.2.1. Tổng quan về EC 2**

Điện toán Đám mây Linh hoạt của Amazon (**Amazon Elastic Compute Cloud** –**Amazon EC2**) là một dịch vụ cung cấp khả năng tính toán có thể mở rộng trong dịch vụ đám mây của Amazon Web Services (AWS). Sử dụng Amazon EC2 giúp doanh nghiệp không cần phải đầu tư vào hệ thống phần cứng. Do đó, doanh nghiệp có thể phát triển và triển khai các ứng dụng nhanh hơn.

Nó cho phép bạn sử dụng môi trường Amazon để tính toán và kiểm soát hoàn toàn tài nguyên máy tính của bạn. Khi nhu cầu điện toán của bạn thay đổi, bạn có thể nhanh chóng mở rộng công suất vì bạn có thể khởi động và tạo lập các máy chủ mới trong vài phút thông qua Amazon EC2. Nó cũng bảo vệ dữ liệu, chống lại các sự cố ngoài ý muốn và cung cấp cho các nhà phát triển các công cụ để phát triển các ứng dụng có khả năng phục hồi những sự cố.

Sau khi đã biết được Amazon EC2 là gì, chúng ta cùng tìm hiểu về các tính năng vượt trội của Amazon EC2. [8]

### **4.2.2. Các tính năng nổi bật EC2**

Amazon EC2 có nhiều tính năng khác nhau giúp doanh nghiệp phát triển các ứng dụng, với khả năng phục hồi dữ liệu. Các tính năng đó bao gồm:

#### **4.2.2.1 Mạng mở rộng**

Tính năng này cho phép bạn giảm độ trễ, giảm tải cho mạng và đem đến hiệu suất hoạt động cao hơn đáng kể mỗi giây. So với cách triển khai máy chủ truyền thống, tính năng này làm giảm việc sử dụng CPU và tăng cường hiệu năng đầu vào và đầu ra thông qua việc sử dụng ngăn xếp ảo hóa mạng mới.

Cài đặt trình điều khiển phù hợp và khởi chạy AMI – máy ảo hỗ trợ phần cứng (HVM) trong VPC để hưởng lợi từ mạng tăng cường. Hiện tại I2, C3 và R3 hỗ trợ kết nối mạng tăng cường.

#### **4.2.2.2 Lưu trữ dạng khối lâu dài – Amazon Elastic Block Store (EBS)**

Tính năng này cung cấp sự lưu trữ liên tục cho các phiên bản của Amazon EC2. Khối lượng Amazon EBS được gắn vào mạng và tồn tại mà không bị phụ thuộc vào tuổi thọ của máy chủ. Chúng có thể được gắn dưới dạng một thiết bị lưu trữ khối tiêu chuẩn với phiên bản Amazon EC2 đang chạy hoặc được sử dụng làm phân vùng khởi động của phiên bản Amazon EC2 với có độ tin cậy và tính khả dụng cao.

Bạn có thể dừng và khởi động lại các phiên bản Amazon EC2 theo nhu cầu khi bạn sử dụng Amazon EC2 làm phân vùng khởi động. Điểm hay của việc định giá AWS EBS là bạn chỉ phải trả tiền cho các tài nguyên lưu trữ được sử dụng để duy trì trạng thái của máy chủ. Các tài nguyên Amazon EBS được sao chép tự động. Vì vậy, so với Amazon EC2, Amazon EBS cung cấp độ bền tốt hơn.

Bạn có thể lưu dữ liệu của mình để đảm bảo độ bền trong thời gian dài bằng cách sử dụng ảnh chụp nhanh làm điểm bắt đầu cho các tài nguyên Amazon EBS mới. Những ảnh chụp nhanh này có thể dễ dàng chia sẻ với các nhà phát triển AWS và đồng nghiệp.

#### **4.2.2.3 Cửa hàng AWS**

Bạn có thể nhanh chóng đưa phần mềm chạy trên AWS để sử dụng sau khi bạn tìm và mua nó từ AWS Marketplace, một cửa hàng trực tuyến. Bạn phải trả tiền theo tháng hoặc theo giờ khi bạn khởi chạy phần mềm được cấu hình sẵn từ AWS Marketplace. Thanh toán và hoá đơn được xử lý bởi AWS và các khoản phí cho phần mềm hiển thị trên hóa đơn AWS của bạn.

#### **4.2.2.4 Dịch vụ giám sát và quản lý – Amazon CloudWatch**

Amazon CloudWatch giám sát các ứng dụng và tài nguyên trên đám mây của Amazon và là một dịch vụ web. Lưu lượng truy cập mạng, tốc độ đọc và ghi đĩa, công suất sử dụng CPU là một số số liệu mà Amazon CloudWatch cung cấp cho bạn về khả năng hiển thị các mẫu nhu cầu tổng thể, hiệu suất hoạt động và sử dụng tài nguyên.

Bạn có thể cài đặt thông báo, xem biểu đồ và lấy số liệu thống kê cho dữ liệu của mình. Bạn chỉ cần chọn các phiên bản Amazon EC2 mà bạn muốn được theo dõi để sử dụng Amazon CloudWatch. Bạn cũng có thể cung cấp ứng dụng hoặc số liệu kinh doanh của riêng bạn. Amazon CloudWatch sẽ thu thập và lưu trữ dữ liệu giám sát có thể được truy cập bằng công cụ dòng lệnh hoặc dịch vụ web API.

#### **4.2.2.5 Các phiên bản EBS được tối ưu**

Bạn có thể khởi chạy một vài đối tượng Amazon EC2 được chọn làm phiên bản được tối ưu hóa EBS để chỉ phải trả một khoản phí thấp theo giờ. Các phiên bản tối ưu hóa EBS cho phép các phiên bản EC2 sử dụng hoàn toàn IOPS (hoạt động đầu vào và đầu ra mỗi giây) được cung cấp trên một khối lượng EBS.

Tùy thuộc vào phiên bản hiện được sử dụng, thông lượng ở bất cứ đâu giữa 425 Mbps và 14.000 Mbps có thể được phân phối bởi các phiên bản được tối ưu hóa EBS giữa Amazon EC2 và Amazon EBS. Sự tham gia giữa I/O (Hoạt động đầu vào và đầu ra) của Amazon EBS và lưu lượng truy cập khác từ phiên bản EC2 của bạn được giảm thiểu thông qua thông lượng chuyên dụng và điều này cung cấp hiệu suất tốt nhất cho khối lượng EBS của bạn.

EBS được tối ưu có thể được sử dụng với cả khối lượng EBS được cung cấp và loại tiêu chuẩn. Khối lượng IOPS được cung cấp có thể phân phối trong khoảng 10% hiệu suất IOPS được cung cấp trong 99,9% thời gian và có thể đạt được độ trễ lên đến mili giây khi được gắn vào các phiên bản được tối ưu hóa EBS.

#### **4.2.2.6 Địa chỉ IP linh hoạt**

Địa chỉ IP tĩnh có thể được sử dụng cho điện toán đám mây động được gọi là địa chỉ IP linh hoạt. Một địa chỉ IP linh hoạt không được liên kết với một phiên bản máy chủ cụ thể nhưng lại được liên kết với tài khoản của bạn và nếu bạn giải phóng nó, bạn sẽ kiểm soát địa chỉ đó.

Amazon EC2 cho phép bạn giải quyết các vấn đề với phần mềm hoặc máy chủ bằng cách nhanh chóng tinh chỉnh lại địa chỉ IP linh hoạt của bạn sang một máy chủ khác thay thế. Vì vậy, bạn không phải chờ DNS để truyền tải đến cho tất cả khách hàng hay đợi kỹ thuật viên thay thế hoặc cấu hình lại máy chủ của bạn.

#### **4.2.2.7 Nhập/Xuất máy ảo (VM) – Virtual Machine (VM) Import/Export**

Sử dụng tính năng này để nhập hình ảnh máy ảo từ môi trường hiện tại của bạn vào các phiên bản Amazon EC2. Bạn có thể xuất chúng trở lại bất cứ khi nào bạn muốn. Tận dụng những tài nguyên mà bạn đã thực hiện trong các máy ảo, quản lý cấu hình và bảo mật CNTT của bạn bằng cách nhập các máy ảo khi các phiên bản EC2 sẵn sàng để sử dụng. Tất cả những gì bạn phải trả là phí sử dụng tiêu chuẩn cho Amazon S3 và Amazon EC2.

#### **4.2.2.8 Tự động điều chỉnh quy mô – Auto Scaling**

Tuỳ theo kế hoạch bạn đã lên, tính năng tự động điều chỉnh quy mô cho phép bạn tăng hoặc giảm công suất của Amazon EC2. Bạn có thể giảm thiểu chi phí trong thời gian nhu cầu sử dụng tăng đột biến hoặc giảm đi theo yêu cầu thông qua quy mô tự động điều chỉnh – tự động tăng hoặc giảm tỷ lệ và cấu hình các phiên bản Amazon EC2 bạn đang sử dụng.

Đối với các ứng dụng trải qua sự thay đổi sử dụng hàng tuần, hàng ngày hoặc hàng giờ, tính năng này rất phù hợp. Amazon CloudWatch cho phép tự động mở rộng quy mô và bạn chỉ phải trả phí Amazon CloudWatch để sử dụng tính năng tự động điều chỉnh này.

#### **4.2.2.9 Đám mây riêng ảo – Amazon VPC**

Bạn có thể xây dựng một lớp mạng ảo trong Amazon EC2, nơi bạn có thể khởi chạy các tài nguyên AWS. Điều này có thể được thực hiện thông qua một phần của đám mây AWS bị cô lập một cách hợp lý với sự trợ giúp của Amazon VPC. Bạn có thể xác định cấu hình cổng mạng và bảng tuyến, tạo mạng con và chọn dải địa chỉ IP của riêng bạn vì bạn kiểm soát hoàn toàn môi trường mạng ảo của mình thông qua Amazon VPC.

Tạo kết nối mạng riêng ảo (VPN) phần cứng giữa VPC và trung tâm dữ liệu công ty của bạn. Chúng ta có thể sử dụng đám mây AWS như một phần mở rộng của trung tâm dữ liệu công ty của bạn theo cách này.

#### **4.2.2.10 Máy chủ đặt tại nhiều vị trí**

Bạn có thể đặt các máy chủ ở một số vị trí thông qua Amazon EC2. Các khu vực bao gồm các khu vực và các khu vực khả dụng (khu vực sẵn có). Các khu vực sẵn có được cách ly khỏi các sự cố trong các khu vực sẵn có khác và là các địa điểm khác biệt.

Chúng cung cấp kết nối mạng chi phí thấp và độ trễ thấp cho các vùng khả dụng trong cùng khu vực. Trong trường hợp bất kỳ vị trí nào bị lỗi, bạn có thể bảo vệ ứng dụng của mình bằng cách khởi chạy các phiên bản trong các vùng khả dụng khác.

Các khu vực nằm trong các quốc gia hoặc khu vực địa lý riêng biệt và có một hoặc nhiều khu vực sẵn có. Amazon EC2 cam kết cung cấp dịch vụ khả dụng 99,95% cho mỗi khu vực.

## Amazon EMR

Amazon EMR là nền tảng dữ liệu lớn trên nền tảng đám mây hàng đầu ngành để xử lý lượng lớn dữ liệu bằng các công cụ nguồn mở như Apache Spark, Apache Hive, Apache HBase, Apache Flink, Apache Hudi và Presto. Với EMR bạn có thể chạy phân tích ở cấp độ Petabyte với chi phí ít hơn một nửa so với các giải pháp tại chỗ truyền thống và nhanh gấp hơn 3 lần so với Apache Spark tiêu chuẩn. Đối với các tác vụ ngắn hạn, bạn có thể tăng tốc và giảm tốc các cụm và thanh toán mỗi giây cho các phiên bản được sử dụng. Đối với khối lượng công việc dài hạn, bạn có thể tạo các cụm có độ sẵn sàng cao tự động điều chỉnh quy mô để đáp ứng nhu cầu. Nếu bạn đã triển khai tại chỗ các công cụ nguồn mở như Apache Spark và Apache Hive, bạn cũng có thể chạy các cụm EMR trên AWS Outposts. [9]

### **4.3.1. Lợi ích**

#### **4.3.1.1. Dễ sử dụng**

Các nhà phân tích, kỹ sư dữ liệu và nhà khoa học dữ liệu có thể sử dụng EMR Notebook, cho phép các cá nhân và nhóm dễ dàng cộng tác và khám phá tương tác, xử lý và trực quan hóa dữ liệu. Bạn chỉ có thể chỉ định phiên bản của ứng dụng EMR và loại điện toán bạn muốn sử dụng. EMR đảm nhiệm việc cung cấp, cấu hình và điều chỉnh các cụm sao cho bạn có thể tập trung vào việc chạy phân tích.

#### **4.3.1.2. Chi phí thấp**

Mức giá của EMR đơn giản và dự báo được: Bạn trả mức phí theo từng phiên bản cho mỗi giây sử dụng, trong đó tiền phí tối thiểu là một phút. Bạn có thể chạy cụm máy chủ EMR gồm 10 nút mạng với mức phí thấp chỉ bằng 0.15 USD mỗi giờ. Bạn cũng có thể tiết kiệm 50-80% chi phí cho các phiên bản bằng cách chọn Amazon EC2 Spot cho khối lượng công việc nhất thời và Phiên bản đặt trước cho khối lượng công việc dài hạn. Bạn cũng có thể sử dụng Savings Plans.

#### **4.3.1.3. Linh hoạt**

Không giống như cơ sở hạ tầng khuôn mẫu của các cụm tại chỗ, EMR phân tách điện toán và lưu trữ, cho bạn khả năng thay đổi quy mô một cách độc lập và tận dụng lưu trữ được phân bậc của Amazon S3. Với EMR, bạn có thể cung cấp một, hàng trăm hoặc hàng nghìn phiên bản điện toán để xử lý dữ liệu ở bất kỳ quy mô nào. Số lượng phiên bản có thể tự động tăng hoặc giảm bằng cách sử dụng Auto Scaling (quản lý kích thước cụm dựa trên mức độ sử dụng) và bạn chỉ trả tiền cho những gì mình sử dụng.

#### **4.3.1.4. Độ tin cậy**

Thời gian cho việc tinh chỉnh và giám sát cụm máy chủ của bạn sẽ được rút ngắn. EMR được tinh chỉnh cho đám mây và lập tức giám sát cụm máy chủ của bạn – thử lại các tác vụ thất bại và tự động thay thế các phiên bản có hiệu năng thấp. Với nhiều nút chính, các cụm máy chủ có mức độ khả dụng cao và chuyển đổi dự phòng tự động trong trường hợp xảy ra sự cố nút. EMR cung cấp các bản phát hành phần mềm nguồn mở ổn định mới nhất, do đó, bạn không phải quản lý các bản cập nhật và sửa lỗi, nhờ đó gặp ít sự cố hơn và không phải tốn nhiều công sức để duy trì môi trường.

#### **4.3.1.5. Bảo mật**

EMR tự động cấu hình thiết lập tường lửa EC2 kiểm soát quyền truy cập mạng đến các phiên bản và khởi chạy các cụm máy chủ trên Amazon Virtual Private Cloud (VPC). Mã hóa phía máy chủ hoặc mã hóa phía máy khách có thể được sử dụng với AWS Key Management Service hoặc các khóa do chính khách hàng quản lý. EMR giúp dễ dàng thực hiện các tùy chọn mã hóa, như mã hóa dữ liệu truyền và lưu trữ và phương thức xác thực mạnh với Kerberos. Bạn có thể sử dụng AWS Lake Formation hoặc Apache Ranger để áp dụng các kiểm soát truy cập dữ liệu chi tiết cho cơ sở dữ liệu, bảng và cột.

#### **4.3.1.6. Linh hoạt**

Bạn có toàn quyền kiểm soát cụm của mình với quyền truy cập gốc vào mọi phiên bản. Bạn có thể khởi chạy các cụm EMR với Amazon Linux AMI tùy chỉnh và dễ dàng cài đặt các ứng dụng bổ sung với các thao tác bootstrap. EMR cho phép bạn cấu hình lại các ứng dụng trên các cụm đang chạy một cách nhanh chóng mà không cần phải khởi chạy lại các cụm. Ngoài ra, sử dụng Hadoop 3.0, bạn có thể gói các tác nhân phụ thuộc của thư viện trong các bộ chứa Docker và gửi cùng các tác vụ của bạn để đơn giản hóa các tác nhân phụ thuộc của môi trường.

### **4.3.2. Các Use Case**

**Machine learning**

Sử dụng các công cụ machine learning tích hợp của EMR, bao gồm Apache Spark MLlib, TensorFlow và Apache MXNet cho các thuật toán machine learning có thể thay đổi quy mô và sử dụng các thao tác của các AMI tùy chỉnh và bootstrap để dễ dàng thêm các thư viện và công cụ phân tích dự đoán của riêng bạn.

**Extract Transform Load (ETL)**

Có thể sử dụng EMR để thực hiện các công việc chuyển đổi dữ liệu (ETL) một cách nhanh chóng và tiết kiệm chi phí chẳng hạn như việc phân loại, tổng hợp và nối trên các tập dữ liệu lớn.

**Phân tích luồng dữ liệu nhấp chuột**

Phân tích dữ liệu nhấp chuột từ Amazon S3 bằng Apache Spark và Apache Hive cho người dùng theo phân khúc, hiểu sở thích của người dùng và phân phối quảng cáo hiệu quả hơn.

**Truyền dòng dữ liệu thời gian thực**

Phân tích các sự kiện từ Apache Kafka, Amazon Kinesis hoặc các nguồn dữ liệu phát trực tuyến khác trong thời gian thực với Apache Spark Streaming và Apache Flink để tạo ra các đường truyền dữ liệu thời gian dài, khả năng sử dụng cao và dung sai cao trên EMR. Chuyển đổi liên tục bộ dữ liệu sang S3 hoặc HDFS và hiểu hơn về Amazon Elasticsearch Service.

**Phân tích tương tác**

EMR Notebooks cung cấp một môi trường phân tích được quản lý dựa trên Jupyter mã nguồn mở cho phép các nhà khoa học, nhà phân tích và nhà phát triển dữ liệu chuẩn bị và trực quan hóa dữ liệu, cộng tác với các đồng nghiệp, xây dựng ứng dụng và thực hiện phân tích tương tác.

**Nghiên cứu cấu trúc gen**

Có thể sử dụng EMR để xử lý khối lượng dữ liệu gen cực lớn và các bộ dữ liệu khoa học lớn khác một cách nhanh chóng và hiệu quả. Các nhà nghiên cứu có thể truy cập dữ liệu gien được lưu trữ miễn phí trên AWS.

# Chương 5. CONTAINER HOÁ

Trong một hệ thống truyền thống, một ứng dụng thường triển khai trong một máy chủ chuyên dụng. Những máy chủ này không phải là phần cứng giá rẻ và thường có giá lên tới hàng ngàn đô la. Việc triển khai vào một máy chủ chuyên dụng chủ yếu được thực hiện để cung cấp quyền truy cập đầy đủ vào tài nguyên phần cứng và để tránh mọi xung đột cấu hình. Tuy nhiên, điều này có thể khiến các công ty tốn một khoản tiền rất lớn mỗi lần họ được yêu cầu triển khai một ứng dụng mới. Từ đây, xuất hiện ảo hóa Mặc dù có cả ưu điểm và nhược điểm đối với phương pháp của nó. Để khắc phục những tiêu cực của ảo hóa, container hóa ra đời. Trong chương này, chúng ta sẽ tìm hiểu các chủ đề sau:

* Ảo hóa và các loại khác nhau của siêu giám sát
* Container hóa và lợi ích của nó
* Docker
* Docker Swarm
* Kubernetes

## Ảo hoá

Trong thế giới ngày nay, máy ảo (VM) rất phổ biến và hữu ích trong nhiều tình huống khác nhau. Máy ảo giúp tạo ra một máy cô lập trong máy vật lý của bạn, với một hệ điều hành bạn chọn. Không có yêu cầu của một hệ điều hành cụ thể từ một bộ nhất định. Một trong những lợi thế của VM là bạn có thể định cấu hình phần cứng cơ bản theo yêu cầu của bạn; nó có thể là bất kỳ kiến trúc nào và bạn có thể thêm bao nhiêu thiết bị ảo tùy thích. Bạn có thể tạo và chạy nhiều máy ảo dựa trên phần cứng vật lý có sẵn. Mặc dù phần cứng vật lý được chia sẻ giữa nhiều máy ảo, về mặt phần mềm và cấu hình, chúng hoàn toàn tách biệt với nhau.

## Hypervisor

Trong thế giới ảo hóa, có một thành phần quan trọng được gọi là **hypervisor**. Bạn có thể nghĩ về nó như một trình giám sát lưu lượng phần cứng và phân bổ tài nguyên.

Nó chia sẻ tài nguyên của phần cứng máy chủ giữa các VM khách khác nhau, chẳng hạn như CPU, bộ nhớ và mạng. Hypervisor giúp đảm bảo sử dụng tốt hơn các tài nguyên phần cứng bằng cách cho phép lưu trữ nhiều phiên bản trên HĐH khách.

Một trình ảo hóa cô lập tất cả các VM khách với nhau, vì vậy chúng không có bất kỳ kiến thức nào về nhau. Nếu bất kỳ máy ảo khách nào bị ảnh hưởng bởi phần mềm độc hại hoặc trở thành nạn nhân của một cuộc tấn công bên ngoài, các máy ảo khác sẽ không bị ảnh hưởng. Điều này làm cho loại công nghệ này cực kỳ an toàn.

Một trong những lợi ích chính của một trình ảo hóa là nó làm cho các chi tiết phần cứng cơ bản không liên quan đến VM khách, điều này làm cho việc di chuyển VM rất dễ dàng. Cho dù bạn yêu cầu VM của mình di chuyển cục bộ hoặc đến một địa điểm từ xa, sẽ không có trục trặc miễn là máy đích có tài nguyên phần cứng cần thiết. Tại đây, quá trình di chuyển sẽ diễn ra suôn sẻ và các máy ảo di chuyển có thể hoạt động ngay lập tức.

### **5.2.1. Các siêu giám sát dựa trên phần cứng**

Điều này cũng được gọi là một siêu giám sát loại 1. Điều này chạy trực tiếp trên phần cứng máy chủ, và còn được gọi là một trình ảo hóa kim loại nguyên gốc, nhúng hoặc trần. Nó là một phần của phần cứng và hoạt động chặt chẽ với nó trong việc phân bổ tài nguyên phần cứng cho các HĐH khác nhau. Ví dụ về các loại siêu giám sát này bao gồm Hyper-V, Oracle OVM, VMWare và ESXi.

### **5.2.2. Các siêu giám sát dựa trên phần mềm**

Điều này cũng được gọi là một siêu giám sát loại 2. Điều này chạy trên hệ điều hành máy chủ như một gói phần mềm. Nó hoạt động như một Trình quản lý máy ảo và phân bổ các tài nguyên phần cứng khác nhau bằng cách sử dụng các cuộc gọi được thực hiện cho hệ điều hành máy chủ. Ví dụ về các loại trình ảo hóa này là Oracle VirtualBox và VMware Fusion.

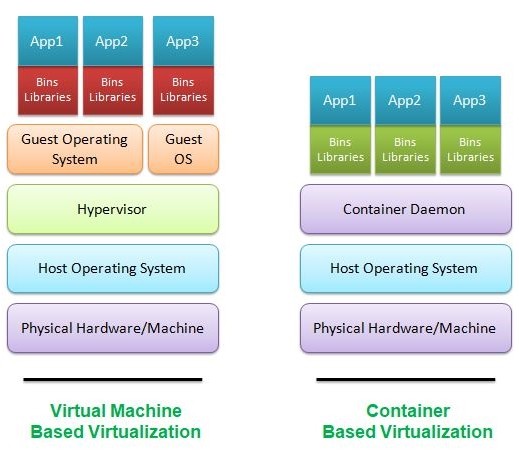
## Container hóa là gì?

Máy ảo có tầm quan trọng rất lớn và mặc dù chúng ta không thể làm việc mà không có chúng, có một số lĩnh vực mà máy ảo bị tụt lại phía sau. Ví dụ: nếu bạn

muốn chạy một ứng dụng trong một môi trường biệt lập, bạn phải thiết lập và chạy một hệ điều hành đầy đủ trước khi bạn có thể làm như vậy. Điều này sẽ có thời gian khởi động riêng và sẽ tiêu tốn rất nhiều tài nguyên khi tải các tệp nhị phân và thư viện của nó. Đổi lại, điều này sẽ giới hạn số lượng VM bạn có thể chạy trên một máy vật lý.

Vậy, container là gì? Containerization tương tự như một máy ảo chạy trong một môi trường biệt lập mà không tiêu tốn một lượng lớn tài nguyên hệ thống. Thay vì tải một hệ điều hành đầy đủ, container hóa sử dụng nhân cơ sở của một hệ điều hành cần thiết và chỉ tải các thư viện theo yêu cầu của ứng dụng chúng tôi. Điều này sẽ mất vài giây để tải, trong khi VM đôi khi có thể mất vài phút để bắt đầu. Do tính chất nhẹ của chúng, các thùng chứa cho phép bạn bắt đầu các ứng dụng biệt lập hơn trong một máy vật lý.

Sơ đồ sau minh họa so sánh kiến trúc của các ứng dụng chạy trong máy ảo và trong thùng chứa:



Hình 34. Virtual machine and container-based architecture

## Lợi ích của container

Hãy khám phá một vài lợi ích chính của việc sử dụng container. Chúng là như sau:

* Các container sử dụng ít tài nguyên hơn, chẳng hạn như CPU và bộ nhớ, so với VM, vì chúng không yêu cầu hệ điều hành đầy đủ để tải
* Bạn có thể bắt đầu, dừng và phá hủy container trong vài giây
* Container rất di động; Khi bạn thiết lập vùng chứa, nó có thể chạy trên mọi môi trường mà không cần cấu hình hoặc cài đặt bổ sung
* Bạn có thể dễ dàng thực hiện chia tỷ lệ ngang của các container trong một môi trường cụm
* Các container tiêu thụ ít không gian lưu trữ hơn so với VM, vì chúng không yêu cầu các nhị phân và thư viện của một hệ điều hành đầy đủ.

Có nhiều trình tiện ích và ứng dụng container khác nhau có sẵn trên thị trường để bạn áp dụng, chẳng hạn như Docker, Kubernetes, CoreOS rkt, Apache Mesos và Canonical LXD. Chúng ta sẽ thảo luận về Docker và Kubernetes trong các phần sau.

## Docker

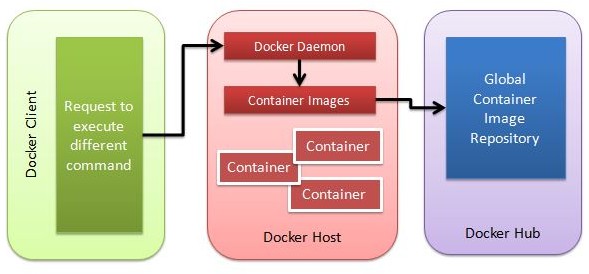
Docker là một công nghệ phần mềm nguồn mở được quản lý bởi Docker Inc. Nó được phát hành lần đầu vào tháng 3 năm 2013. Docker giúp bạn dễ dàng tạo, chạy và triển khai các thùng chứa. Nó được thiết kế để trở thành một phần trong vòng đời phát triển và triển khai của ứng dụng, khiến các nhà phát triển và quản trị viên hệ thống yên tâm rằng nó sẽ chạy trơn tru trên bất kỳ máy nào mà không bỏ qua bất kỳ sự phụ thuộc hoặc cấu hình nào. Bây giờ, hãy khám phá những gì xảy ra khi bạn cố chạy một hình ảnh Docker. [10]

## Quy trình làm việc của Docker

Có ba thành phần trong một động cơ Docker; chúng như sau:

* + - Docker Client: Như tên trạng thái, đây là một máy khách ở mặt trước được sử dụng để đưa ra lệnh cho máy chủ Docker; các lệnh này có thể là bất kỳ loại nào.
    - Docker Host: Đây là công cụ chính lưu trữ hình ảnh container và chạy chúng. Nó thực thi các lệnh khác nhau theo yêu cầu của máy khách Docker.
    - Docker Hub: Đây là một kho chứa hình ảnh container. Khi khách hàng yêu cầu chạy một hình ảnh, Docker Host sẽ kiểm tra bên trong xem hình ảnh có tồn tại cục bộ hay không. Nếu hình ảnh không tồn tại, nó sẽ kéo hình ảnh từ Docker Hub.

Sơ đồ sau minh họa quy trình làm việc của Docker:



Hình 35. Quy trình làm việc của Docker

Có hai phiên bản có sẵn cho Docker và chúng như sau:

* + - Docker Enterprise Edition (EE): Điều này dựa trên cơ chế dựa trên đăng ký và bao gồm hỗ trợ với SLA. Nó có các tầng đăng ký khác nhau, bao gồm Trung tâm dữ liệu Docker tích hợp, bảo mật và cụm.
    - Docker Community Edition (CE): Đây là bản miễn phí cho mọi người sử dụng. Docker phát hành một phiên bản ổn định của phiên bản này trên cơ sở hàng quý và phát hành cạnh trên cơ sở hàng tháng.

## Docker Swarm

Nếu bạn tra từ này trong từ điển, 'swarm' sẽ có nghĩa là một số thứ. Docker Swarm, tuy nhiên, về cơ bản là một cụm máy móc, với mỗi máy chạy một daemon Docker. Mục đích chính là cung cấp một nền tảng có thể mở rộng cho một ứng dụng được đóng gói có thể chạy trong một môi trường nhóm. Trong các phiên bản trước của Docker, Swarm đã có sẵn dưới dạng một sản phẩm riêng biệt, nhưng từ phiên bản 1.12, nó không còn yêu cầu một ứng dụng bổ sung.

Docker Swarm hoạt động bằng cách sử dụng khái niệm manager-worker, trong đó một trong các nút trở thành người quản lý và quản lý các nút worker khác. Điều kiện tiên quyết của việc chạy Docker Swarm là tất cả các máy phải chạy Docker v1.12 trở lên và phải có sẵn trong cùng một mạng con. Master và slave Docker giao tiếp với nhau bằng cổng. Các cổng mặc định là 2377 cho quản lý cụm, 7946 cho nhịp tim và khám phá và 4789 cho lưu lượng mạng lớp phủ. Đảm bảo rằng các cổng này không bị chặn bởi bất kỳ máy nào của bạn có tường lửa để đảm bảo liên lạc Docker Swarm tiếp tục.

## Kubernetes

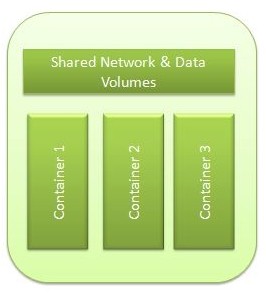
Kubernetes là một công cụ mã nguồn mở để sắp xếp các thùng chứa trong một cụm. Nó hỗ trợ các công nghệ container khác nhau bao gồm Docker và rkt. Ban đầu nó được thiết kế bởi Google và dựa trên dự án quản lý cụm nội bộ của nó và được đặt tên là Project Seven. Sau đó, nó đã được chuyển cho Tổ chức Điện toán đám mây như một công nghệ ban đầu vào tháng 7 năm 2015.

### **5.8.1. Thành phần chính**

Có một số thành phần và thuật ngữ chính bạn sẽ gặp trong khi sử dụng Kubernetes. Đây là như sau.

#### **5.8.1.1. Pods**

Một nhóm là một hoặc nhiều nhóm container được liên kết chặt chẽ với nhau. Container có thể dựa trên bất kỳ khung nào, chẳng hạn như Docker, rkt, v.v. Nó là đơn vị triển khai nhỏ nhất. Nó cũng có mạng chia sẻ và khối lượng dữ liệu.

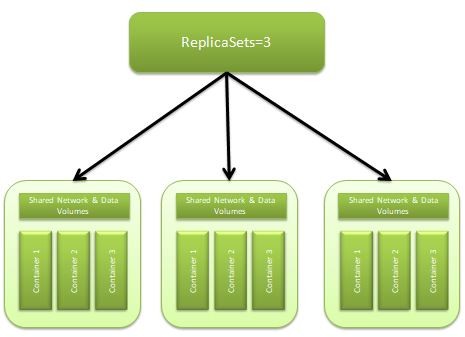


Hình 36. Minh hoạ Pod

Một pod có địa chỉ IP có thể định tuyến riêng của mình, nhưng không phải là nó luôn có cùng một địa chỉ. Ví dụ: nếu xảy ra lỗi và pod phải khởi động lại, nó có thể khởi động lại trong cùng một nút cụm hoặc trong một nút khác, điều này có thể dẫn đến một địa chỉ IP khác.

#### **5.8.1.2. ReplicaSets**

Mục đích của ReplicaSets là tạo ra bất kỳ số lượng bản sao nào của pod. Nó có trách nhiệm bắt đầu và dừng một pod. Nó cũng thường xuyên kiểm tra trạng thái của một nhóm. Nếu bất kỳ pod phải đối mặt với bất kỳ vấn đề hoặc thất bại, nó sẽ được khởi động lại. Hình dưới đây minh họa các bản sao:



Hình 37. Minh hoạ ReplicaSets

#### **5.8.1.3. Deployments**

Một deployments là một tệp cấu hình có chứa tất cả các khai báo của ứng dụng của bạn. Nó bao gồm các khai báo cho các biến môi trường, hình ảnh chứa, khối lượng dữ liệu và nhiều hơn nữa. Nó cũng chứa thông tin liên quan đến bản sao. Khi bạn thực hiện triển khai, nó sẽ tạo các bản sao, từ đó tạo ra các nhóm.

Khi bạn cập nhật ứng dụng của mình, nó sẽ tạo một bản sao riêng biệt do kết quả của việc triển khai mới của bạn. Nó bỏ một nhóm từ phiên bản cũ của ứng dụng của bạn và tạo một nhóm mới theo phiên bản mới của ứng dụng - và chu kỳ tiếp tục. Nếu có bất cứ điều gì sai do kết quả của việc triển khai mới, nó sẽ tự động quay trở lại phiên bản trước của ứng dụng của bạn.

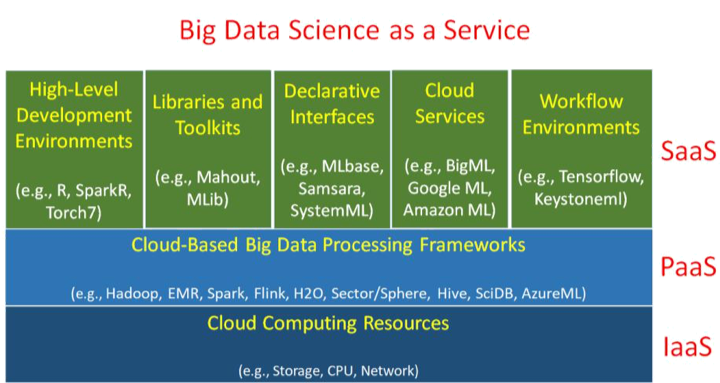
#### **5.8.1.4. PetSets**

Một Pet là một Pod có một trạng thái. Điều đó có nghĩa là khối lượng dữ liệu được tạo động, theo yêu cầu của một container, sẽ không bao giờ bị xóa tự động. Nó sẽ một lần nữa liên kết lại với cùng một khối lượng dữ liệu mà nó được liên kết lần đầu tiên khi khởi động lại. Một bộ sưu tập những Pet này được gọi là Pets

# Chương 6. NỀN TẢNG CHO KHOA HỌC DỮ LIỆU

Hiện tượng Dữ liệu lớn đã tạo ra áp lực ngày càng tăng đối với các giải pháp chấm dứt dữ liệu có thể mở rộng. Một số hệ thống quản lý và xử lý dữ liệu đã được thực hiện gần đây. Đặc biệt, cách tiếp cận cơ sở dữ liệu NoQuery đã trở nên phổ biến trong những năm qua như là một giải pháp thay thế hoặc bổ sung cho cơ sở dữ liệu quan hệ, để đảm bảo khả năng mở rộng theo chiều ngang của các hoạt động cơ sở dữ liệu đọc / ghi đơn giản được phân phối trên nhiều máy chủ. Các hệ thống NoQuery như HBase, Cassandra, MongoDB, Couchbase và DynamoDB ngày nay được sử dụng hiệu quả để hỗ trợ triển khai các công việc và ứng dụng khung phân tích Dữ liệu lớn. Thật vậy, cơ sở dữ liệu NoQuery cung cấp các cơ chế và kỹ thuật hiệu quả để lưu trữ và truy cập các giá trị vô hướng, các đối tượng nhị phân và dữ liệu phức tạp hơn.

Các yêu cầu phân tích dữ liệu ngày càng tăng của hầu hết tất cả các lĩnh vực ứng dụng đã tạo ra một nhu cầu thiết yếu trong việc thiết kế và xây dựng một thế hệ công cụ khoa học Big Data mới có thể phân tích hiệu quả và hiệu quả lượng lớn dữ liệu để gợi ra thông tin đáng giá, phát hiện những hiểu biết thú vị và khám phá ý nghĩa mô hình và kiến thức. Theo ba mô hình dịch vụ đám mây chính (Phần 2), các công cụ Khoa học dữ liệu lớn hỗ trợ phân tích và học hỏi từ dữ liệu có thể được phân loại là Paas hoặc SaaS. Phân loại này được minh họa trong Hình 1 tóm tắt Khoa học dữ liệu lớn dưới dạng ngăn xếp phần mềm Dịch vụ. Thông thường, lớp PaaS bao gồm các khung phân tích dữ liệu dựa trên đám mây được sử dụng để triển khai phần mềm khoa học dữ liệu như một dịch vụ trong khi lớp SaaS bao gồm các phần mềm hoặc thư viện máy học dựa trên đám mây được phát triển để trích xuất mô hình kiến thức, học từ dữ liệu và tạo dự đoán. Trong các phần phụ sau, chúng tôi phân loại các khung phân tích dữ liệu theo hai mô hình dịch vụ chính này: PaaS và SaaS.



Hình 38. Khoa học dữ liệu lớn như một chồng phần mềm dịch vụ

## Khung phân tích dữ liệu PaaS

Các khung PaaS cho phép người dùng tập trung vào việc tạo và chạy các ứng dụng thay vì xây dựng và duy trì cơ sở hạ tầng và dịch vụ cơ bản. Các khung PaaS phân tích dữ liệu cung cấp các dịch vụ tính toán và lưu trữ cũng như các dịch vụ phân tích dữ liệu và máy học giúp các nhà phát triển tạo ra các ứng dụng nhanh chóng và hiệu quả hơn. Chúng tôi sẽ thảo luận về một số khung được sử dụng nhiều nhất sau đây.

MapReduce là một mô hình lập trình được Google phát triển để xử lý dữ liệu quy mô lớn nhằm đối phó hiệu quả với thách thức xử lý lượng dữ liệu khổng lồ được tạo bởi các ứng dụng dựa trên Internet. Kể từ khi được giới thiệu, MapReduce đã được chứng minh là có thể áp dụng cho nhiều lĩnh vực, bao gồm học máy và khai thác dữ liệu và phân tích dữ liệu xã hội. Ngày nay, MapReduce được công nhận rộng rãi là một trong những mô hình lập trình quan trọng nhất cho môi trường điện toán đám mây và được Google và các nhà cung cấp đám mây hàng đầu khác như Amazon hỗ trợ, với dịch vụ đàn hồi MapReduce và Microsoft, với HDInsight hoặc các cơ sở

hạ tầng đám mây riêng tư khác chẳng hạn như OpenStack, với dịch vụ Sahara. Dự án Hadoop được công nhận là ứng dụng MapReduce mã nguồn mở phổ biến nhất để phát triển các ứng dụng song song phân tích lượng lớn dữ liệu. Nó có thể được chấp nhận để phát triển các ứng dụng phân tán và song song bằng nhiều ngôn ngữ pro- gramming khác nhau (ví dụ: Java, Ruby, Python, C ++). Hadoop giúp các nhà phát triển không phải đối phó với các vấn đề điện toán phân tán cổ điển, chẳng hạn như cân bằng tải, khả năng chịu lỗi, địa phương dữ liệu và tiết kiệm băng thông mạng. Kết quả là, Hadoop đã trở thành một tài liệu tham khảo cho một số khung công tác khác, chẳng hạn như: Giraph cho phân tích biểu đồ; Bão để phân tích dữ liệu trực tuyến; Hive, một phần mềm kho dữ liệu để truy vấn và quản lý các bộ dữ liệu lớn; Pig, một ngôn ngữ dataflow để khám phá các bộ dữ liệu lớn; Tez để thực hiện đồ thị theo hướng phức tạp của các tác vụ xử lý dữ liệu; Oozie, một hệ thống lập lịch quy trình công việc để quản lý các công việc Hadoop.

Apache Spark là một nền tảng phần mềm để phân tích Dữ liệu lớn dựa trên mô hình xử lý trong bộ nhớ. Khác với Hadoop trong đó dữ liệu trung gian luôn được lưu trữ trong các hệ thống tệp phân tán, Spark lưu trữ dữ liệu trong RAM và truy vấn nó một cách liên tục để có được hiệu suất tốt hơn cho một số loại ứng dụng (ví dụ: thuật toán học máy lặp). Trong nhiều năm, Hadoop đã được coi là khung Big Data mã nguồn mở dẫn đầu, nhưng gần đây Spark đã trở nên phổ biến hơn để được mọi nhà cung cấp Hadoop hỗ trợ. Trên thực tế, đối với các tác vụ cụ thể, Spark nhanh hơn tới 100 lần so với Hadoop trong bộ nhớ và nhanh hơn 10 lần trên đĩa. Một số thư viện khác đã được xây dựng trên Spark: Spark SQL để xử lý SQL và DataFrames, MLib cho máy học, GraphX cho đồ thị và tính toán song song đồ thị và Spark Streaming để xây dựng các ứng dụng phát trực tuyến có khả năng chịu lỗi. Vì những lý do này, Spark đang trở thành công cụ thực thi chính để xử lý dữ liệu và nói chung, cần phải có cho các ứng dụng Dữ liệu lớn. Nhưng mặc dù trong một số ứng dụng, Spark có thể được coi là sự thay thế tốt hơn cho Hadoop, nhưng trong nhiều ứng dụng khác, nó có một số hạn chế khiến nó bổ sung cho Hadoop. Hạn chế chính của Spark là nó

không cung cấp hệ thống lưu trữ phân tán và có khả năng mở rộng của riêng nó, đây là một yêu cầu cơ bản đối với các ứng dụng Dữ liệu lớn sử dụng khối lượng dữ liệu khổng lồ và liên tục được lưu trữ trên một số lượng rất lớn các nút. Để vượt qua giới hạn này, Spark đã được thiết kế để chạy trên một số nguồn dữ liệu, chẳng hạn như lưu trữ đối tượng Đám mây (ví dụ: Lưu trữ Amazon S3, Lưu trữ đối tượng Swift), hệ thống tệp phân tán (ví dụ: HDFS), cơ sở dữ liệu NoQuery (ví dụ: HBase, Cassandra), và những phần mềm khác.

Sector / Sphere là một khung công tác Đám mây được thiết kế để triển khai các ứng dụng phân tích dữ liệu liên quan đến các bộ dữ liệu lớn, được phân phối theo địa lý trong đó dữ liệu có thể được xử lý song song. Khung này bao gồm hai thành phần: dịch vụ lưu trữ có tên là ngành, quản lý các bộ dữ liệu phân tán lớn với độ tin cậy cao, IO hiệu suất cao và có quyền truy cập thống nhất và dịch vụ tính toán có tên Sphere, sử dụng dịch vụ của ngành để đơn giản hóa việc truy cập dữ liệu, tăng dữ liệu IO băng rộng và khai thác các mạng hiệu suất cao diện rộng. Cả hai đều có sẵn dưới dạng phần mềm nguồn mở. Sphere là một dịch vụ tính toán được xây dựng trên đỉnh của ngành và cung cấp một bộ giao diện lập trình để viết các ứng dụng phân tích dữ liệu phân tán. Sphere lấy luồng làm đầu vào và tạo luồng làm đầu ra. Luồng bao gồm các phân đoạn dữ liệu nhiều ple được xử lý bởi Công cụ xử lý hình cầu (SPE) bằng các nút tớ. Thông thường có nhiều phân khúc hơn TỐC ĐỘ. Mỗi SPE lấy một phân đoạn từ một luồng làm đầu vào và tạo ra một phân đoạn của luồng làm đầu ra. Các phân đoạn ngoài này có thể lần lượt là các phân đoạn đầu vào của một quá trình Sphere khác. Các nhà phát triển có thể sử dụng API máy khách Sphere để khởi tạo luồng đầu vào, tải lên các thư viện chức năng xử lý, bắt đầu các quy trình Sphere và đọc kết quả xử lý.

H2O là một khung công tác mã nguồn mở cung cấp một công cụ xử lý song song được trang bị các thư viện toán học và máy học. Nó cung cấp hỗ trợ cho các ngôn ngữ pro-gramming khác nhau bao gồm Java, R, Python và Scala. Các thuật toán

học máy được triển khai trên khung công tác MapReduce phân tán của H2O và ngoại trừ khung Java Fork / Tham gia để thực hiện đa luồng. H2O thực hiện nhiều thuật toán học máy, chẳng hạn như mô hình tuyến tính tổng quát (ví dụ: hồi quy tuyến tính, hồi quy logistic), Nave Bayes, phân tích thành phần chính (PCA), chuỗi thời gian, phân cụm K-nghĩa, mạng thần kinh và các loại khác. H2O cũng thực hiện các chiến lược khai thác dữ liệu phức tạp như Random Forest, Gradient Boosting và Deep Learning. Người dùng trên H2O có thể xây dựng hàng ngàn mô hình và so sánh chúng để có kết quả dự đoán tốt nhất. H2O chạy trên một số platoforms trên đám mây, bao gồm Amazon EC2 và S3 Storage, Microsoft Azure và IBM DSX.

Microsoft đã giới thiệu AzureML như một giải pháp khung máy học cung cấp môi trường trực quan dựa trên đám mây để xây dựng các luồng công việc phân tích dữ liệu. Azure ML thường được mô tả là SaaS, tuy nhiên cũng có thể thấy PaaS vì nó có thể được sử dụng để phát triển các giải pháp SaaS trên đầu trang. Nó được cung cấp như một dịch vụ được quản lý hoàn toàn bởi Microsoft, nơi người dùng không cần phải mua bất kỳ phần cứng / phần mềm nào cũng như tự điều khiển bất kỳ máy ảo nào. AzureML cung cấp cho các nhà khoa học dữ liệu một IDE học máy dựa trên Web để tạo và tự động hóa quy trình công việc học máy. Ngoài ra, nó cung cấp các triển khai song song và có thể mở rộng của các kỹ thuật máy học phổ biến cũng như khả năng xử lý dữ liệu bằng giao diện kéo và thả. AzureML có thể đọc và nhập dữ liệu từ nhiều nguồn khác nhau bao gồm URL HTTP, Azure Blob Storage, Azure Table và Azure SQL Database. Nó cũng cho phép các nhà khoa học dữ liệu nhập các tập lệnh phân tích dữ liệu tùy chỉnh của riêng họ (ví dụ: bằng R hoặc Python). Cumulon đã có mặt như một hệ thống được thiết kế để giúp người dùng nhanh chóng phát triển và triển khai các chương trình phân tích dữ liệu lớn dựa trên ma trận trong đám mây. Nó cung cấp một sự trừu tượng cho việc lưu trữ phân tán các ma trận trên đỉnh HDFS. Đặc biệt, ma trận được lưu trữ và truy cập bằng gạch. Một chương trình Cumulon thực thi như một quy trình công việc. Mỗi công việc đọc một số ma trận đầu vào và viết một số ma trận đầu ra; ma trận đầu vào và đầu ra phải rời rạc. Sự phụ

thuộc giữa các công việc được ngụ ý bởi các truy cập phụ thuộc vào cùng một ma- trice. Công việc phụ thuộc thực hiện theo thứ tự nối tiếp. Mỗi công việc thực thi nhiều nhiệm vụ độc lập không giao tiếp với nhau. Cumulon dựa trên Hadoop thừa hưởng các tính năng quan trọng của Hadoop như xử lý lỗi và có thể tận dụng hệ sinh thái Hadoop sôi động. Trong khi nhắm mục tiêu các hoạt động ma trận, Cumulon có thể hỗ trợ các chương trình cũng chứa các công việc Hadoop truyền thống, không ma trận.

SciDB đã được giới thiệu như một cơ sở dữ liệu phân tích được định hướng theo nhu cầu quản lý dữ liệu của quy trình công việc khoa học. Cụ thể, nó trộn lẫn các hoạt động đại số thống kê và tuyến tính với các hoạt động quản lý dữ liệu bằng mô hình dữ liệu mảng đa chiều. SciDB hỗ trợ cả chức năng (AFL) và truy vấn truy vấn giống như SQL (AQL) trong đó AQL được biên dịch thành AFL.

MADlib đã cung cấp một bộ triển khai dựa trên SQL để khai thác dữ liệu và thuật toán học máy được thiết kế để được cài đặt và chạy ở quy mô trong bất kỳ công cụ cơ sở dữ liệu quan hệ nào hỗ trợ SQL mở rộng, không cần nhập / xuất dữ liệu các công cụ bên ngoài khác. Các phương pháp phân tích trong MADlib được thiết kế cho cả thực thi trong hoặc ngoài lõi và cho song song không chia sẻ quy mô không có gì được cung cấp bởi các công cụ cơ sở dữ liệu song song hiện đại, đảm bảo tính toán được thực hiện gần dữ liệu. Chức năng cốt lõi của MADlib được viết bằng các câu lệnh SQL khai báo, giúp phối hợp di chuyển dữ liệu đến và từ đĩa và trên các máy tính nối mạng.

MLog đã được trình bày như một ngôn ngữ cấp cao tích hợp học máy vào các hệ thống quản lý dữ liệu. Nó mở rộng ngôn ngữ truy vấn qua mô hình dữ liệu SciDB để cho phép người dùng chỉ định các mô hình học máy theo cách tương tự như các quan điểm quan hệ truyền thống và các truy vấn quan hệ. Nó được thiết kế để tự động quản lý tất cả các chuyển động dữ liệu, lưu giữ dữ liệu và tối ưu hóa liên quan đến

máy học. Mô hình dữ liệu của MLog dựa trên các thang đo thay vì quan hệ. Trong thực tế, tất cả dữ liệu trong MLog là tenxơ và tất cả các phép toán là một tập hợp con của đại số tuyến tính trên các tenxơ.

## Khung phân tích dữ liệu SaaS

Với nhu cầu ngày càng tăng đối với các yêu cầu phân tích dữ liệu trong một số miền, một bộ khung đã được triển khai để đơn giản hóa và đẩy nhanh quá trình thực hiện các công việc phân tích Dữ liệu lớn của người dùng cuối. Các khung lập trình đó được thiết kế để triển khai các ứng dụng phân tích dữ liệu và máy học như là các phần kết hợp của các dịch vụ cấp cao với mục tiêu giảm gánh nặng và độ phức tạp của lập trình.

Môi trường phát triển cấp cao Có tính đến các khung mức cao trừu tượng với kiến trúc xử lý, chúng ta phải trình bày hệ thống R. R hiện được coi là tiêu chuẩn thực tế trong nghiên cứu phân tích dữ liệu và thống kê. Đây là phần mềm đa nền tảng và mã nguồn mở phổ biến nhất có hỗ trợ cộng đồng rất rộng. Nó linh hoạt, mở rộng và toàn diện cho năng suất. R cung cấp một ngôn ngữ lập trình được sử dụng bởi các nhà thống kê và nhà khoa học dữ liệu để thực hiện các nhiệm vụ phân tích dữ liệu và khám phá những hiểu biết mới về dữ liệu bằng cách khai thác các công nghệ như phân cụm, hồi quy, phân loại và phân tích văn bản. Nó được trang bị thư viện gói rất phong phú và mạnh mẽ. Đặc biệt, R cung cấp một tập hợp phong phú các hàm tích hợp cũng như mở rộng để trích xuất dữ liệu, làm sạch dữ liệu, tải dữ liệu, chuyển đổi dữ liệu, phân tích thống kê, học máy và trực quan hóa. Ngoài ra, nó cung cấp khả năng kết nối với các ngôn ngữ và hệ thống khác (ví dụ: Python). Trong thực tế, một nhược điểm chính với R là hầu hết các gói của nó được phát triển hàng đầu để sử dụng trong bộ nhớ và tương tác, tức là, cho các tình huống trong đó dữ liệu phù hợp với bộ nhớ. Với mục đích giải quyết thách thức này và cung cấp khả năng xử lý các bộ dữ liệu khổng lồ, một số hệ thống đã được phát triển để hỗ trợ thực hiện các chương trình R trên các nền tảng xử lý Dữ liệu lớn phân tán và có thể mở rộng như Hadoop (ví dụ: Ricardo , RHadoop và RHIPE, Segue) và Spark (ví dụ: SparkR). Ví dụ, RHIPE là gói R mang khung MapReduce cho người dùng R và cho phép họ truy cập cụm

Hadoop từ trong môi trường R. Cụ thể, bằng cách sử dụng các chức năng R cụ thể, người dùng có thể khởi chạy các công việc MapReduce trên cụm Hadoop nơi kết quả có thể dễ dàng truy xuất từ HDFS. Segue cho phép người dùng thực thi các công việc MapReduce từ bên trong môi trường R trên các nền tảng MapReduce của Amazon. SparkR đã trở thành một gói R phổ biến, hỗ trợ giao diện trọng lượng nhẹ để thực thi các chương trình R trên công cụ tính toán phân tán Apache Spark và cho phép thực hiện các tác vụ hậu môn dữ liệu quy mô lớn từ vỏ R. Pydoop là gói Python cung cấp API cho cả khung Hadoop và HDFS. Torch7 đã được trình bày như một môi trường toán học và khung tính toán số linh hoạt cho các thuật toán học máy tích hợp. Theano đã được trình bày như một trình biên dịch đại số tuyến tính nhằm tối ưu hóa các tính toán toán học và tạo ra các triển khai cấp thấp hiệu quả.

Bộ công cụ và thư viện Trong số các bộ công cụ khác nhau, Apache Mahout là bộ công cụ nguồn mở được thiết kế để giải quyết các vấn đề học máy rất thực tế và có thể mở rộng trên nền tảng Hadoop. Do đó, Mahout chủ yếu có nghĩa là để phân phối và xử lý hàng loạt các kích thước dữ liệu khổng lồ trên một cụm. Cụ thể, Mahout về cơ bản là một tập hợp các thư viện Java được tích hợp tốt với Apache Hadoop và được thiết kế để làm cho các ứng dụng học máy dễ dàng hơn để xây dựng. Gần đây, Mahout đã được mở rộng để cung cấp hỗ trợ cho các thuật toán học máy để lọc và phân loại cộng tác trên các nền tảng Spark và H2O. MLib đã được trình bày dưới dạng thư viện máy học phân tán Spark Spark rất phù hợp cho các nhiệm vụ học máy lặp đi lặp lại. Nó cung cấp các triển khai có thể mở rộng của các thuật toán học tập tiêu chuẩn cho các cài đặt học tập phổ biến bao gồm phân loại, hồi quy, lọc cộng tác, phân cụm và giảm kích thước. MLlib hỗ trợ các ngôn ngữ thứ bảy (ví dụ: Java, Scala và Python) và cung cấp API cấp cao tận dụng hệ sinh thái giàu Spark Spark để đơn giản hóa việc phát triển các đường ống học máy từ đầu đến cuối.

Giao diện / Ngôn ngữ khai báo Một số triển khai máy học khai báo đã được triển khai trên các hệ thống xử lý Dữ liệu lớn. Đối với kỳ thi, Samsara, đã được giới thiệu như một môi trường toán học hỗ trợ thực hiện khai báo cho đại số tuyến tính nói chung và các hoạt động thống kê như là một phần của thư viện Apache Mahout.

Nó cho phép người dùng chỉ định các chương trình theo kiểu giống như R bằng cách sử dụng một tập hợp trừu tượng ma trận phổ biến và các phép toán đại số tuyến tính. Samsara biên dịch, tối ưu hóa và thực hiện các chương trình của nó trên các hệ thống dữ liệu phân tán (ví dụ: Apache Spark, Apache Flink, H2O). MLbase đã được triển khai để cung cấp thư viện máy học đa năng với mục tiêu tương tự như mục tiêu Mahout, để cung cấp giải pháp khả thi để xử lý các tác vụ học máy quy mô lớn trên khung Spark. Nó hỗ trợ ngôn ngữ khai báo giống như tiếng Latin Latin để chỉ định các tác vụ và thực hiện máy học và cung cấp một bộ các trình điều khiển cấp độ cao cho phép người dùng thực hiện một loạt các phương pháp học máy mà không cần kiến thức hệ thống sâu. Ngoài ra, nó thực hiện một trình tối ưu hóa để chọn và tự động điều chỉnh sự lựa chọn của thuật toán học tập.

Apache SystemML cung cấp một khung công tác máy học khai báo được phát triển để chạy trên Apache Spark. Nó hỗ trợ cú pháp giống R và Python bao gồm các hàm thống kê, nguyên hàm đại số tuyến tính và các cấu trúc đặc thù ML. Nó áp dụng các kỹ thuật biên dịch dựa trên chi phí để tạo ra các kế hoạch thực hiện ở mức độ thấp, hiệu quả với các hoạt động phân tán đơn nút và quy mô lớn trong bộ nhớ.

ScalOps đã được trình bày dưới dạng ngôn ngữ dành riêng cho miền (DSL) vượt ra ngoài các phân tích dữ liệu vượt qua một lần (tức là, MapReduce) để bao gồm khối lượng công việc nhiều vượt qua, hỗ trợ lặp lại các thuật toán được thể hiện dưới dạng truy vấn quan hệ trên dữ liệu đào tạo và mô hình. Các kế hoạch thực hiện vật lý của ScalOps bao gồm các toán tử dataflow được thực thi bằng công cụ tính toán chuyên sâu dữ liệu Hyracks.

Mxnet là một thư viện đã được thiết kế để dễ dàng phát triển các thuật toán học máy. Nó pha trộn biểu thức khai báo với tính toán tenxơ bắt buộc và cung cấp sự khác biệt tự động để lấy ra độ dốc. MXNet được thiết kế để chạy trên các hệ thống không đồng nhất khác nhau, từ các thiết bị di động đến các cụm GPU phân tán.

Các dịch vụ học máy trên đám mây Nhìn chung, một trong những lợi thế chính của công việc với các công cụ SaaS dựa trên đám mây là người dùng không phải lo

lắng về việc mở rộng giải pháp của họ. Thay vào đó, lý tưởng nhất, dịch vụ được cung cấp sẽ có thể tự động mở rộng quy mô nếu mức tiêu thụ tài nguyên máy tính cho các mô hình phân tích được xác định đã tăng và theo các yêu cầu và cấu hình do người dùng xác định. Google đã cung cấp một nền tảng học máy SaaS dựa trên nền tảng đám mây, được trang bị các mô hình được đào tạo trước cùng với một nền tảng để tạo ra các mô hình người dùng. Dịch vụ này được tích hợp với các dịch vụ khác của Google như Google Cloud Storage và Google Cloud Dataflow. Nó đóng gói các mô hình học máy mạnh mẽ hỗ trợ các ứng dụng phân tích khác nhau (ví dụ: phân tích hình ảnh, nhận dạng giọng nói, phân tích văn bản và dịch tự động) thông qua các lệnh gọi API REST. Tương tự, Amazon cung cấp máy học như một giải pháp dịch vụ (AML) hướng dẫn người dùng của mình thông qua quá trình tạo mô hình phân tích dữ liệu mà không cần phải học các thuật toán hoặc công nghệ phức tạp. Khi các mô hình được tạo, dịch vụ giúp bạn dễ dàng thực hiện dự đoán thông qua API đơn giản mà không cần phải viết bất kỳ mã người dùng nào hoặc quản lý bất kỳ cơ sở hạ tầng phần cứng hoặc phần mềm nào. AML hoạt động với dữ liệu được lưu trữ trong Amazon S3, RDS hoặc Redshift. Nó cung cấp một bộ API để kết nối và thao tác với các nguồn dữ liệu khác. AML dựa trên nền tảng Amazon SageMaker cho phép người dùng xây dựng, đào tạo và triển khai các mô hình học máy của họ. IBM Watson Analytics41 là một khung phân tích dự đoán SaaS khác cho phép người dùng thể hiện công việc phân tích của họ bằng ngôn ngữ tiếng Anh tự nhiên. Dịch vụ cố gắng tự động phát hiện các mối tương quan và ngoại lệ thú vị trong dữ liệu đầu vào. Nó cũng cung cấp các đề xuất về các bước làm sạch dữ liệu khác nhau và kỹ thuật hiển thị dữ liệu đầy đủ để sử dụng cho các tình huống phân tích khác nhau.

Khung công tác BigML SaaS hỗ trợ khám phá các mô hình dự đoán từ dữ liệu đưa vào bằng thuật toán phân loại và hồi quy dữ liệu. Trong BigML, các mod dự đoán được trình bày cho người dùng dưới dạng cây quyết định tương tác được tự động hóa và khám phá trong giao diện BigML. BigML cũng cung cấp giải pháp PaaS, BigML PredictServer, có thể được tích hợp với các ứng dụng, dịch vụ và các công cụ phân tích dữ liệu khác. Hunk là một nền tảng phân tích dữ liệu thương mại được phát triển

để nhanh chóng khám phá, phân tích và trực quan hóa dữ liệu trong các cửa hàng dữ liệu Hadoop và NoQuery. Hunk sử dụng một bộ giao diện lập trình và người dùng cấp cao để cải thiện tốc độ và sự đơn giản để có được những hiểu biết sâu sắc từ các tập dữ liệu lớn không có cấu trúc và có cấu trúc. Một trong những thành phần chính của kiến trúc Hunk là Chỉ số ảo Splunk. Hệ thống này tách lớp lưu trữ khỏi tầng truy cập và phân tích dữ liệu, do đó cho phép Hunk định tuyến các yêu cầu đến các kho lưu trữ dữ liệu khác nhau. Tầng phân tích dựa trên Ngôn ngữ xử lý tìm kiếm Splunks (SPL) được thiết kế để khám phá dữ liệu trên các tập dữ liệu lớn, khác nhau. Khung web Hunk cho phép xây dựng các ứng dụng trên hệ thống tệp phân tán Hadoop (HDFS) và / hoặc kho lưu trữ dữ liệu NoQuery. Các nhà phát triển có thể sử dụng Hunk để xây dựng các ứng dụng Dữ liệu lớn của họ trên đầu dữ liệu trong Hadoop bằng cách sử dụng một bộ ngôn ngữ và khung nổi tiếng. Thật vậy, khung cho phép các nhà phát triển tích hợp dữ liệu và chức năng từ Hunk vào các ứng dụng Dữ liệu lớn dành cho doanh nghiệp bằng cách sử dụng khung web, API REST và bộ công cụ phát triển phần mềm cho C #, Java, JavaScript, PHP và Ruby.

Nền tảng phân tích Kognitio được thiết kế dưới dạng dịch vụ Đám mây hoặc được cung cấp dưới dạng thiết bị tích hợp sẵn cho phép người dùng kéo một lượng rất lớn dữ liệu từ các hệ thống lưu trữ dữ liệu hiện có vào bộ nhớ máy tính tốc độ cao, cho phép trả lời các câu hỏi phân tích phức tạp. Mặc dù Kognitio có hệ thống con đĩa bên trong riêng, nhưng nó chủ yếu được sử dụng làm lớp phân tích trên các hệ thống lưu trữ / xử lý dữ liệu cũ, ví dụ, cụm Hadoop và / hoặc các sản phẩm kho dữ liệu dựa trên đĩa truyền thống hiện có, lưu trữ đám mây, v.v. Một tính năng được gọi là Tables bên ngoài cho phép dữ liệu liên tục cư trú trên các hệ thống bên ngoài. Sử dụng tính năng này, quản trị viên hệ thống hoặc người dùng đặc quyền, có thể dễ dàng thiết lập quyền truy cập vào dữ liệu cư trú trong môi trường khác, điển hình là cửa hàng đĩa như cụm Hadoop và hệ thống kho dữ liệu được đề cập ở trên. Đối với người dùng cuối cùng, Nền tảng phân tích Kognitio trông giống như một hệ thống quản lý cơ sở dữ liệu quan hệ (RDBMS) tương tự như nhiều cơ sở dữ liệu thương mại. Tuy nhiên, không giống như các cơ sở dữ liệu này, Kognitio đã được thiết kế đặc biệt để xử lý

khối lượng công việc truy vấn phân tích, trái ngược với khối lượng công việc xử lý giao dịch trực tuyến (OLTP) truyền thống hơn.

Nubytics là một hệ thống Phần mềm dưới dạng Dịch vụ (SaaS) khai thác các cơ sở Đám mây để cung cấp các dịch vụ hiệu quả để phân tích các bộ dữ liệu lớn. Hệ thống cho phép người dùng nhập dữ liệu của họ vào Đám mây, trích xuất các mô hình kiến thức bằng cách sử dụng các dịch vụ khai thác dữ liệu hiệu suất cao và khai thác kiến thức được suy luận để dự đoán dữ liệu mới và dữ liệu mới. Cụ thể, Nubytics cung cấp dịch vụ phân loại và hồi quy dữ liệu có thể được sử dụng trong nhiều ứng dụng khoa học và kinh doanh. Khả năng mở rộng được cung cấp bởi một phương pháp tính toán song song khai thác triệt để các tài nguyên có sẵn trên Đám mây. Nubytics khác với các khung phân tích dữ liệu mục đích chung như Azure ML, Hadoop và Sparks hoặc các hệ thống quản lý luồng công việc theo định hướng dữ liệu như DMCF, vì nó cung cấp các dịch vụ chuyên biệt để phân loại và dự đoán dữ liệu. Các dịch vụ này được cung cấp bởi giao diện Web cho phép các nhà phân tích dữ liệu tập trung vào quá trình phân tích dữ liệu mà không phải lo lắng về các chi tiết lập trình cấp thấp. Cách tiếp cận này tương tự như phương pháp được BigML áp dụng. Tuy nhiên, Nubytics cũng tập trung vào khả năng mở rộng, bằng cách triển khai phương pháp tính toán song song ad hoc khai thác triệt để các tài nguyên phân tán của nền tảng điện toán đám mây.

Môi trường dòng công việc Về nguyên tắc, quy trình công việc được sử dụng như một mô hình hiệu quả cho lập trình phân tích dữ liệu. Một số khung sử dụng mô hình này để tích hợp các phương pháp phân tích dữ liệu. Họ chủ yếu tập trung vào việc đơn giản hóa quá trình phối hợp các thành phần khác nhau và giảm thời gian sản xuất. Ở đây, chúng tôi chỉ thảo luận về một vài hệ thống đại diện cho lớp framework này.

Tensorflow cung cấp giao diện để thiết kế các thuật toán học máy và triển khai để thực hiện các thuật toán đó. Cụ thể, Tensorflow lấy các tính toán được mô tả bằng mô hình giống như dataflow và biên dịch chúng trên một số nền tảng phần cứng, từ

chạy suy luận trên nền tảng thiết bị di động (ví dụ: Android và iOS) đến các hệ thống phân tán quy mô lớn của hàng trăm máy và cát của các thiết bị tính toán như thẻ GPU. Trọng tâm chính của Tensorflow là đơn giản hóa việc sử dụng hệ thống máy học trong thế giới thực và giảm đáng kể gánh nặng bảo trì. TFX đã được Google trình bày dưới dạng nền tảng máy học đa năng dựa trên TensorFlow tích hợp các thành phần khác nhau bao gồm người học để tạo mô hình dựa trên dữ liệu đào tạo, mô-đun để phân tích và xác thực cả dữ liệu cũng như mô hình. đến cơ sở hạ tầng để phục vụ các mô hình trong sản xuất. Keras46 là một thư viện học sâu của Python, có khả năng chạy trên đỉnh của TensorFlow hoặc Theano. Nó cho phép tạo mẫu dễ dàng và nhanh chóng thông qua sự thân thiện, tính mô đun và khả năng mở rộng của người dùng. Bose¨ et al. [68] đã trình bày một nền tảng cho các phương pháp tiếp cận máy học quy mô lớn (ML) cho phép đào tạo và áp dụng các mô hình dự báo nhu cầu xác suất. Nền tảng này bao gồm một hệ thống máy học từ đầu đến cuối, được xây dựng trên Apache Spark, bao gồm tiền xử lý dữ liệu, kỹ thuật fea-ture, học tập phân tán, đánh giá, thử nghiệm và tập hợp.

Khung phân tích F2 đã được thiết kế để tách biệt thực thi khỏi quản lý dữ liệu và xử lý tính toán và dữ liệu như các công dân hạng nhất như nhau. Cụ thể, trong khung này, dữ liệu được quản lý riêng trong khi các quyết định xác định cách phân vùng dữ liệu hoặc khi nào nó được xử lý được thực hiện trong thời gian chạy. Việc tính toán xử lý dữ liệu có thể bị mất ngữ nghĩa và chạy bất kỳ hoạt động có sẵn nào trên bất kỳ dữ liệu nào đã sẵn sàng. Một trong những ưu điểm chính của thiết kế khung này là nó cung cấp sự linh hoạt hơn trong việc thể hiện các công việc phân tích bằng cách loại bỏ các mối quan tâm về phân vùng dữ liệu, định tuyến và logic nào để chỉ định trong thời gian chạy.

Khung khai thác dữ liệu đám mây (DMCF) là một hệ thống phần mềm được triển khai để thiết kế và thực hiện quy trình phân tích dữ liệu trên Đám mây. Giao diện người dùng dựa trên Web cho phép người dùng soạn ứng dụng của họ và gửi chúng để thực thi trên tài nguyên Đám mây, theo cách tiếp cận Phần mềm dưới dạng dịch vụ (SaaS). Kiến trúc DMCF đã được thiết kế để triển khai trên các cài đặt Đám mây khác nhau. Hiện tại, có hai triển khai khác nhau của DMCF: i) trên nền tảng đám mây Dịch vụ nền tảng (PaaS), tức là sử dụng API lưu trữ, tính toán và API mạng ẩn lớp cơ sở hạ tầng không liên quan; ii) trên đỉnh của đám mây Cơ sở hạ tầng (IaaS), tức là sử dụng hình ảnh máy ảo (VM) được triển khai trên lớp cơ sở hạ tầng. Trong cả hai kịch bản triển khai, DMCF sử dụng Microsoft Azure làm nhà cung cấp đám mây. Các mô-đun phần mềm DMCF có thể được nhóm thành các thành phần web và tính toán các tính toán. DMCF cho phép người dùng soạn thảo, kiểm tra và chạy các quy trình phân tích dữ liệu thông qua trình soạn thảo web HTML5. Quy trình công việc có thể được xác định bằng hai ngôn ngữ: VL4Cloud (Visual Language for Cloud) và JS4Cloud (JavaScript for Cloud). Cả hai cách sử dụng lan truyền đều sử dụng ba tóm tắt chính: Các phần tử dữ liệu, biểu thị các tệp đầu vào (ví dụ: tập dữ liệu cần phân tích) hoặc các tệp đầu ra (ví dụ: mô hình khai thác dữ liệu); Các phần tử công cụ, đại diện cho các công cụ phần mềm được sử dụng để thực hiện các thao tác trên các phần tử dữ liệu (phân vùng, lọc, khai thác, v.v.); và Nhiệm vụ, đại diện cho việc thực thi các phần tử Công cụ trên các phần tử Dữ liệu đầu vào đã cho để tạo một số phần tử Dữ liệu đầu ra.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Framework** | **Model** | **Abstraction** | **Supported Languages** | **Underlying Engine** | **Availability** |
| Hadoop | PaaS | MapReduce | Java, Python | Hadoop | Open source |
| Spark | PaaS | RDD | Scala, Python, Java, R | Spark, Yarn | Open source |
| H2O | PaaS | Procedural + Library | REST, R,Python | H2O | Open source |
| SciDB | PaaS | Declarative | SQL | SiDB | Open source |
| AzureML | SaaS/PaaS | Visual User Interface | REST | Microsoft Azure | Proprietary/ Microsoft |
| R | SaaS/  PaaS | Procedural | R | R | Open source |
| SparkR | SaaS/  PaaS | Procedural | R | Spark | Open source |
| Mahout | SaaS/PaaS | Toolkit | Java, Scala | Hadoop, Spark | Open source |
| Spark MLib | SaaS/  PaaS | Library | Scala, Python | Spark | Open Source |
| Samsara | SaaS/  PaaS | Declarative | Java, Scala | Spark, Flink, H2O | Open Source |
| Apache SystemML | SaaS/  PaaS | Declarative | R, Python | Spark | Open Source |
| Google ML | SaaS | Visual User Interface | Python | Google Cloud Dataflow | Proprietary/ Google |
| Amazon ML | SaaS | Visual User Interface | N/A | Apache MXNet, TensorFlow, PyTorch | Proprietary/ Amazon |
| BigML | SaaS/  Paas | Visual User Interface | Python | BigML PredictServer | Proprietary/ BigML |
| Tensorflow | SaaS | Visual User Interface/Lib rary | Python, Haskell, Java, Go Julia, R, Scala | CUDA, TPU | Open Source |
| KeystoneML | SaaS/  PaaS | Procedural + Library | Scala | Spark | Open Source |

Bảng 2. Tóm tắt tính năng chính của Khung phân tích dữ liệu lớn

Khung Keystoneml đã được thiết kế để hỗ trợ xây dựng các đường ống phức tạp và nhiều giai đoạn bao gồm khai thác tính năng, giảm kích thước, biến đổi dữ liệu và mô hình học tập có giám sát. Nó cung cấp API cấp độ cao, an toàn loại được xây dựng xung quanh các toán tử logic để nắm bắt các ứng dụng học máy từ đầu đến cuối. Để tối ưu hóa các đường ống học máy, Keystoneml áp dụng các công nghệ để thực hiện cả tối ưu hóa cho mỗi nhà khai thác và tối ưu hóa đường ống đầu cuối. Nó sử dụng một trình tối ưu hóa dựa trên chi phí chiếm cả chi phí tính toán và truyền thông. Trình tối ưu hóa cũng có thể xác định trạng thái trung gian nào sẽ được vật chất hóa trong bộ nhớ chính trong quá trình thực hiện lặp lại trên dữ liệu thô. MBDAaaS đã được đề xuất như là một khung cho Dịch vụ phân tích dữ liệu lớn dựa trên mô hình hỗ trợ người dùng có chuyên môn về Dữ liệu lớn hạn chế trong việc triển khai các đường ống dữ liệu. Nó cung cấp một mô hình khai báo để chỉ định các mục tiêu của một phân tích nhất định dưới dạng các chỉ số / mục tiêu cặp được sử dụng để tinh chỉnh dần dần một mô hình thủ tục độc lập với nền tảng để chỉ định cách phân tích nên được thực hiện theo quy trình công việc trừu tượng. Các mô hình thủ tục sau đó được xếp chồng lên nhau trong một mô hình triển khai sẵn sàng thực hiện, nó tự động đề xuất các cấu hình thành phần phụ thuộc nền tảng và hỗ trợ cung cấp tự động các thành phần và tài nguyên tính toán. Để hoàn thành cuộc thảo luận của chúng tôi và đưa ra cái nhìn so sánh, Bảng 2 tóm tắt các tính năng chính của khung phân tích Dữ liệu lớn mà chúng tôi đã trình bày.

# Chương 7. XÂY DỰNG HỆ THỐNG DỮ LIỆU LỚN HỖ TRỢ MÁY HỌC VÀ ÁP DỤNG GIẢI QUYẾT BÀI TOÁN DROWSINESS DETECTION

## 7.1 Trình bày ý tưởng

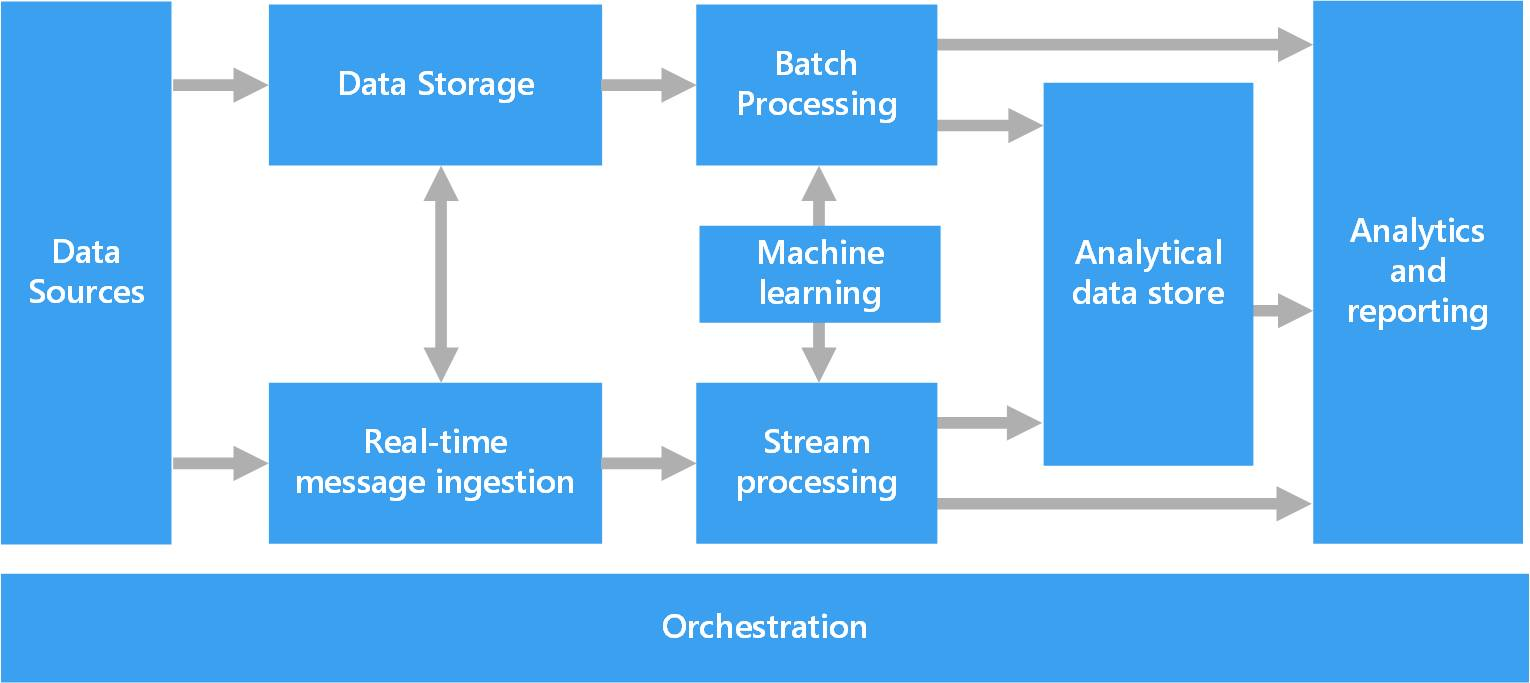
Hiện nay, việc cài đặt một hệ thống dữ liệu lớn rất phức tạp:

* Conflict giữa các component
* Khó khăn trong việc tái cài đặt hệ thống ở một cluster khác.
* Chi phí, giá thành cài đặt rất cao.
* Tính tương thích với phần mềm của công ty.

Do vậy việc thiết kế hệ thống dữ liệu lớn để giải quyết các khó khan trên là mục tiêu mà tôi đã đề ra.

## 7.2. Khái quát, tổng quan hóa ý tưởng

Từ 2 mô hình kiến trức dữ liệu lớn đã trình bày ở chương 2 và chương 3, chúng ta đúc kết được mô hình tổng quát của các kiến trúc dữ liệu lớn hiện nay:

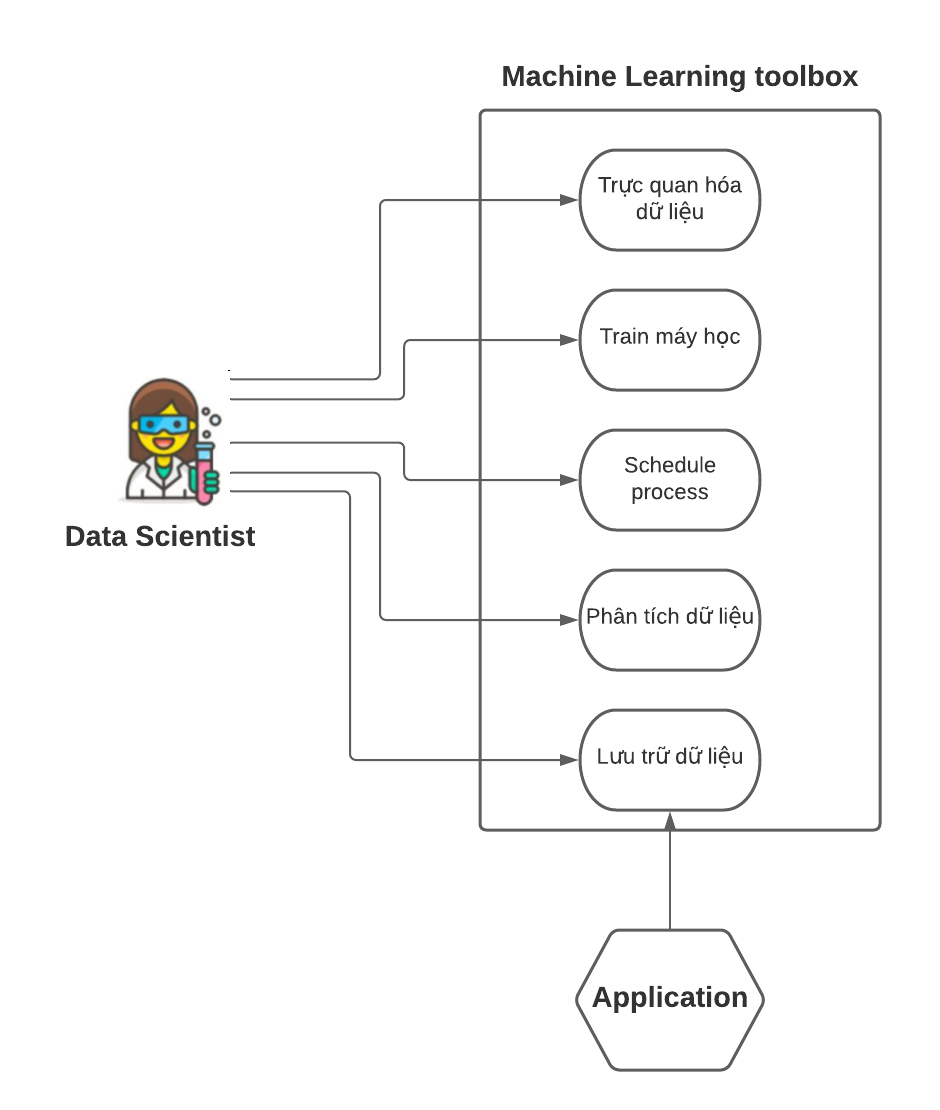


Hình 39. Kiến trúc hệ thống mô hình

* **Nguồn dữ liệu (Data Sources)**: nơi dữ liệu được sinh ra, bao gồm dữ liệu có cấu trúc (structure), dữ liệu phi cấu trúc (un-structure) cũng như dữ liệu bán cấu trúc (semi-structure). Dữ liệu có thể đến từ rất nhiều nguồn khác nhau như dữ liệu từ các ứng dụng, cơ sở dữ liệu quan hệ (giao dịch mua/ bán hàng từ một hệ thống bán lẻ, giao dịch gửi tiền vào ngân hàng, …), hoặc dữ liệu file được tạo ra bởi các log của ứng dụng (dữ liệu log ghi lại thời gian xử lý của hệ thống, …), hay dữ liệu thời gian thực từ các thiết bị IoT (hình ảnh theo dõi từ camera, cảm biến nhiệt độ, độ ẩm, ….)
* **Lưu trữ dữ liệu (Data Storage)**: thành phần này được thiết kế để lưu trữ lại khối lượng rất lớn các loại dữ liệu với các định dạng khác nhau được sinh ra bởi nguồn dữ liệu (Data Source) trong mô hình xử lý dữ liệu theo lô (Batch Processing). Mô hình phổ biến cho thành phần này là các hệ thống lưu trữ file phân tán trên nhiều node khác nhau trong 1 cụm (cluster), đảm bảo cho khả năng xử lý lượng rất lớn các file dữ liệu cũng như tính an toàn của dữ liệu (replica). Apache Hadoop HDFS đang được sử dụng phổ biến để implement thành phần này trong các hệ thống Big Data.
* **Xử lý dữ liệu theo lô (Batch Processing)**: thành phần này cho phép xử lý một lượng lớn dữ liệu thông qua việc đọc dữ liệu từ các file nguồn, lọc dữ liệu theo các điều kiện nhất định, tính toán trên dữ liệu, và ghi kết quả xuống 1 file đích. Trong thành phần này bạn có thể sử dụng Spark, Hive, MapReduce, … với nhiều ngôn ngữ lập trình khác nhau như Java, Scala hoặc Python.
* **Thu thập dữ liệu thời gian thực (Real-time Message Ingestion):** như đã nói ở trên, dữ liệu được sinh ra từ nguồn (Data Source) có thể bao gồm dữ liệu thời gian thực (ví dụ từ các thiết bị IoT) do đó thành phần này cho phép một hệ thống Big Data có thể thu thập và lưu trữ các loại dữ liệu trong thời gian thực phục vụ cho việc xử lý dữ liệu theo luồng (Streaming Processing). Công nghệ phổ biến nhất chắc các bạn hay nghe đến Kafka, ngoài ra còn có những cái tên khác như RabbitMQ, ActiveMQ, …
* **Lưu trữ dữ liệu phân tích (Analytical Data Store)**: chịu trách nhiệm lưu trữ dữ liệu đã được xử lý theo định dạng có cấu trúc để phục vụ cho các công cụ phân tích dữ liệu (BI Tools). Dữ liệu có thể được lưu trữ dưới dạng OLAP trong thiết kế Kimball (cho bạn nào chưa biết thì Kimball là một trong 3 phương pháp luận khi thiết kế 1 data warehouse: Inmon, Kimball và Data Vault) hoặc dữ liệu có thể lưu trữ bằng các công nghệ NoQuery như HBase, Cassandra, …
* **Lớp phân tích và báo cáo (Analysis and Reporting)**: thành phần này đáp ứng việc tự khai thác dữ liệu data self-service. Cho phép người dùng cuối trực quan hóa dữ liệu (data visualization), phân tích dữ liệu, cũng như kết xuất các báo cáo khác nhau. Công nghệ được sử dụng ở tầng này khá đa dạng, có thể là các open source tool như D3.JS, Dygaphs, … đến các công cụ commercial như Tableau, Power BI, …, hay bạn có thể tự code bằng các ngôn ngữ Python, R, …
* **Điều phối (Orchestration)**: thành phần này có nhiệm vụ điều phối các công việc trong một hệ thống Big Data để đảm bảo luồng xử lý dữ liệu được thông suốt, từ việc thu thập dữ liệu, lưu trữ dữ liệu đến lọc, tính toán trên dữ liệu. Apache Oozie, Airflow, …

## 7.3. Các Use case của Machine Learning Toolbox

Use case diagram:



Hình 40. Các use case của machine learning toolbox.

### **7.3.1. Lưu trữ dữ liệu**

**Mục đích**: lưu trữ tất cả các loại dữ liệu từ mọi phía.

**Tác nhân**: Data Scientist, Ứng dụng phần mềm, các nguồn sản sinh dữ liệu (IOT).

**Mô tả**: Data scientist hay các nguồn dữ liệu muốn thao tác với dữ liệu của mình phải đẩy lên nơi lưu trữ.

**Luồng sự kiện chính (Actor là nguồn dữ liệu)**

* User đăng nhập vào Azure để lấy client id, client secret, tenant ID.
* Gắn các thông tin trên vào mã nguồn của ứng dụng.
* Gọi hàm để đẩy dữ liệu lên azure.
* Azure kiểm tra quyền truy cập của nơi dữ liệu được đẩy lên.
  + Nếu nơi lưu trữ đã được cấp phép, tiến hành lưu dữ liệu.
  + Nếu chưa, thì dữ liệu không được lưu và đòi quyền truy cập.

**Luồng thay thế (Actor là nguồn dữ liệu)** : Nếu dữ liệu là stream real time theo thời gian thực thì đẩy vào kafka cluster.

**Các yêu cầu cụ thể (Actor là nguồn dữ liệu)**: thông tin key đã nhập ở mã nguồn phải khớp mới thông tin key từ phía Azure cloud.

**Luồng sự kiện chính (Actor là Data scientist)**

* User đăng nhập vào Azure cloud.
* Vào cloud storage trên Azure (ở đây là Datalake gen 1).
* Chọn upload file.
* Chọn file để upload.
* Azure kiểm tra quyền truy cập của user ở nơi dữ liệu được đẩy lên.
  + Nếu nơi lưu trữ đã được cấp phép, tiến hành lưu dữ liệu.
  + Nếu chưa, thì dữ liệu không được lưu và đòi quyền truy cập.

**Luồng thay thế (Actor là Data scientist)** : Không có.

**Các yêu cầu cụ thể (Actor là data scientist)**: Không có.

**Điều kiện trước**: người dùng đã có tài khoản trên Azure cloud.

**Điều kiện sau**: thông báo đã lưu dữ liệu thành công hay thất bại.

### **7.3.2. Phân tích dữ liệu**

**Mục đích**: Tìm ra những giá trị mà dữ liệu thô mang lại.

**Tác nhân**: Data Scientist.

**Mô tả**: Data scientist sử dụng spark và các framework liên quan để bóc tách dữ liệu từ azure hoặc kafka.

**Luồng sự kiện chính**

* User đăng nhập vào jupyter.
* Tạo new 1 notebook.
* Tạo spark session, context.
* Lấy data từ azure hay kafka.
* Phân tích dữ liệu.

**Luồng thay thế**: Lưu dữ liệu đã phân tích vào Cassandra.

**Các yêu cầu cụ thể :** Không có.

**Điều kiện trước**: người dùng đã có tài khoản trên Jupyter và Azure.

**Điều kiện sau**: Không có.

### **7.3.3. Training model máy học**

**Mục đích**: Feed những giá trị từ dữ liệu cho model.

**Tác nhân**: Data Scientist.

**Mô tả**: Data scientist sử dụng các giá trị đã bóc tách để feed cho model máy học.

**Luồng sự kiện chính**

* User đăng nhập vào jupyter.
* Tạo new 1 notebook.
* Tạo model máy học từ spark, tensorflow hay keras,...
* Lấy data từ azure hay Cassandra.
* Feed cho model máy học.
* Lưu model vào Azure.

**Luồng thay thế**: Không có.

**Các yêu cầu cụ thể :** Không có.

**Điều kiện trước**: người dùng đã có tài khoản trên Jupyter và Azure.

**Điều kiện sau**: Thông báo khi train model thành công, hoặc thất bại.

### **7.3.4. Trực quan hóa dữ liệu ( tạo báo cáo)**

**Mục đích**: Tạo báo cáo từ những dữ liệu đã phân tích.

**Tác nhân**: Data Scientist.

**Mô tả**: Data scientist sử dụng các giá trị đã bóc tách để feed cho model máy học.

**Luồng sự kiện chính**

* User đăng nhập vào jupyter.
* Tạo new 1 notebook.
* Lấy data từ azure hay Cassandra.
* Dùng matplotlib, seaborn trực quan hóa dữ liệu

**Luồng thay thế**: Có thể lưu báo cáo vào Azure.

**Các yêu cầu cụ thể :** Không có.

**Điều kiện trước**: người dùng đã có tài khoản trên Jupyter và Azure.

**Điều kiện sau**: Không có.

### **7.3.5. Kế hoạch hóa từ process**

**Mục đích**: Đặt thời cho từng process tự chạy.

**Tác nhân**: Data Scientist.

**Mô tả**: Data scientist đặt thời gian cho các process chạy.

**Luồng sự kiện chính**

* User truy cập airflow thông qua browser.
* Tạo new 1 process.
* Cấu hình process theo nhu cầu và đặt thời gian.
* Lưu process.

**Luồng thay thế**: Không có.

**Các yêu cầu cụ thể :** Không có.

**Điều kiện trước**: người dùng đã có tài khoản trên Jupyter và Azure.

**Điều kiện sau**: Thông báo lưu process thành công, hoặc thất bại.

## 7.4. Machine learning toolbox

### **7.4.1. Chi tiết hệ thống**

Sau khi đã nắm được mô hình tổng quát, chúng ta bắt đầu chọn công cụ để áp dụng nhằm hạn chế tối đa các khuyến điểm đã nêu ra ở phần 7.1. trình bày ý tưởng.

#### **7.4.1.1. Data Storage**

Data Storage, chúng tôi chọn dịch vụ cloud của các bên phát triển như amazon, Microsoft hay google. So với lưu trữ tại chỗ bằng hệ thống Hadoop, dịch vụ trên cloud có các ưu điểm sau (cụ thể là AWS S3):

* **Cost**

Ở đây, chúng ta xét theo công thức: Cost = Storage Cost + Maintainance cost. Tức chi phí ở đây sẽ bao gồm chi phí cho việc lưu trữ và bảo trì cơ sở hạ tầng.

Trước tiên, hãy ước tính chi phí lưu trữ 1 terabyte dữ liệu mỗi tháng.

Gía bộ nhớ tiêu chuẩn của S3 cho 1TB dữ liệu đầu tiên là **23 đô la / tháng**. Đối với dữ liệu sử dụng bộ nhớ thường xuyên truy cập sẽ chỉ có giá một nửa, ở mức **12,5 đô la / tháng**. Chúng ta sẽ sử dụng giá cơ bản là $ 23 / tháng để ước tính chi phí. S3 không đi kèm với dung lượng tính toán nhưng nó cho phép bạn tự do tận dụng các cụm tạm thời và chọn các loại phiên bản phù hợp nhất cho khối lượng công việc.

Đối với HDFS, do chưa tiếp xúc thực tế với các máy chủ bên ngoài, nên chúng ta sẽ ước tính storage cost dựa trên máy ảo dịch vụ của Amazon là EC2. Các phiên bản lưu trữ tiết kiệm chi phí nhất trên EC2 là d2 server. Để tìm ra trường hợp tốt nhất cho HDFS, chúng tôi sử dụng các giả định sau mà hầu như không thể đạt được trong thực tế:

* Gỉa sử khối lượng data sinh ra mỗi năm luôn ổn định, khối lượng công vệc (workload) không biến động. Điều này có nghĩa là hệ thống lưu trữ không cần phải co giãn (Elasticity).
* Hiệu suất sử dụng bộ nhớ là 70% và hệ số sao chép HDFS tiêu chuẩn đặt ở mức 3.

Với các giả định trên, sử dụng loại phiên bản d2,8xl (5,52 đô la / giờ giảm giá 71%, ổ cứng 48TB), chi phí **5,52 x 0,29 x 24 x 30/48 x 3 / 0,7 = 103 đô la / tháng cho 1TB dữ liệu.**

Vì vậy, chỉ tính riêng về chi phí lưu trữ (Storage cost), **S3 rẻ hơn HDFS 5 lần.**

Chi phí nhân lực (Maintainance cost) của S3 hầu như bằng không, trong khi thường phải cần đến một nhóm kỹ sư Hadoop hoặc bộ phận hỗ trợ của nhà cung cấp để duy trì HDFS. Khi chúng tôi tính đến **chi phí con người, S3 rẻ hơn 10 lần so với các cụm HDFS trên EC2 với dung lượng tương đương**.

* **Độ co giãn (Elasticity)**

Việc ước lượng sức chứa cho storage rất khó thực hiện đúng và rất ít tổ chức có thể ước tính trước chính xác. Trong các hệ thống tại chỗ, điều này dẫn đến sự khó khan rất lớn. Khi dữ liệu phình nở trong tương lai biến động rất nhiều khiến bạn phải thêm dung lượng cho storage nhưng bạn không thể biết chính xác mình cần thêm bao nhiêu sẽ gây ra sự lãng phí lớn về tài chính cũng như trên hệ thống.

Một trong những lợi ích tuyệt vời nhất của S3, hay lưu trữ đám mây nói chung, là tính đàn hồi và mô hình định giá trả (pay as you go) khi bạn sử dụng: bạn chỉ bị tính tiền khi bạn đưa thêm dữ liệu vào. Về cơ bản, nhà cung cấp đám mây tự động cung cấp tài nguyên theo yêu cầu.

Nói một cách đơn giản, **S3 có tính đàn hồi, HDFS thì không.**

* **Tính sẵn sàng và khả năng chịu đựng (Availability - Durability)**

Trong quá trình tìm hiểu,dựa trên các báo cáo của các kỹ sư trong Databrick, chỉ hai lần trong sáu năm qua, họ đã trải qua thời gian ngừng hoạt động của S3 và chưa bao giờ có thông báo bị mất dữ liệu từ S3.

Amazon tuyên bố 99,999999999% độ bền và 99,99% tính khả dụng. Lưu ý rằng con số này cao hơn so với phần lớn các dịch vụ nội bộ của các tổ chức. Các bạn có thể tìm thấy thông tin về tuyên bố trên trong mục thỏa thuận trang Amazon s3 Overview.

Ngược lại, đối với HDFS, rất khó để ước tính tính khả dụng và độ bền. Về mặt lý thuyết, người ta có thể tính toán dựa trên thời gian trung bình giữa các lần hỏng hóc của EC2, cộng với thời gian ngừng nâng cấp và bảo trì. Trong thực tế, rất khó để định lượng. Các kỹ sư bên databrick làm việc với khách hàng cho biết phần lớn các cụm Hadoop có tính khả dụng thấp hơn 99,9%, tức là thời gian ngừng hoạt động ít nhất 9 giờ mỗi năm.

Với tính năng sao chép xuyên A tới Z tự động sao chép qua các trung tâm dữ liệu khác nhau, tính khả dụng và độ bền của S3 vượt trội hơn nhiều so với HDFS.

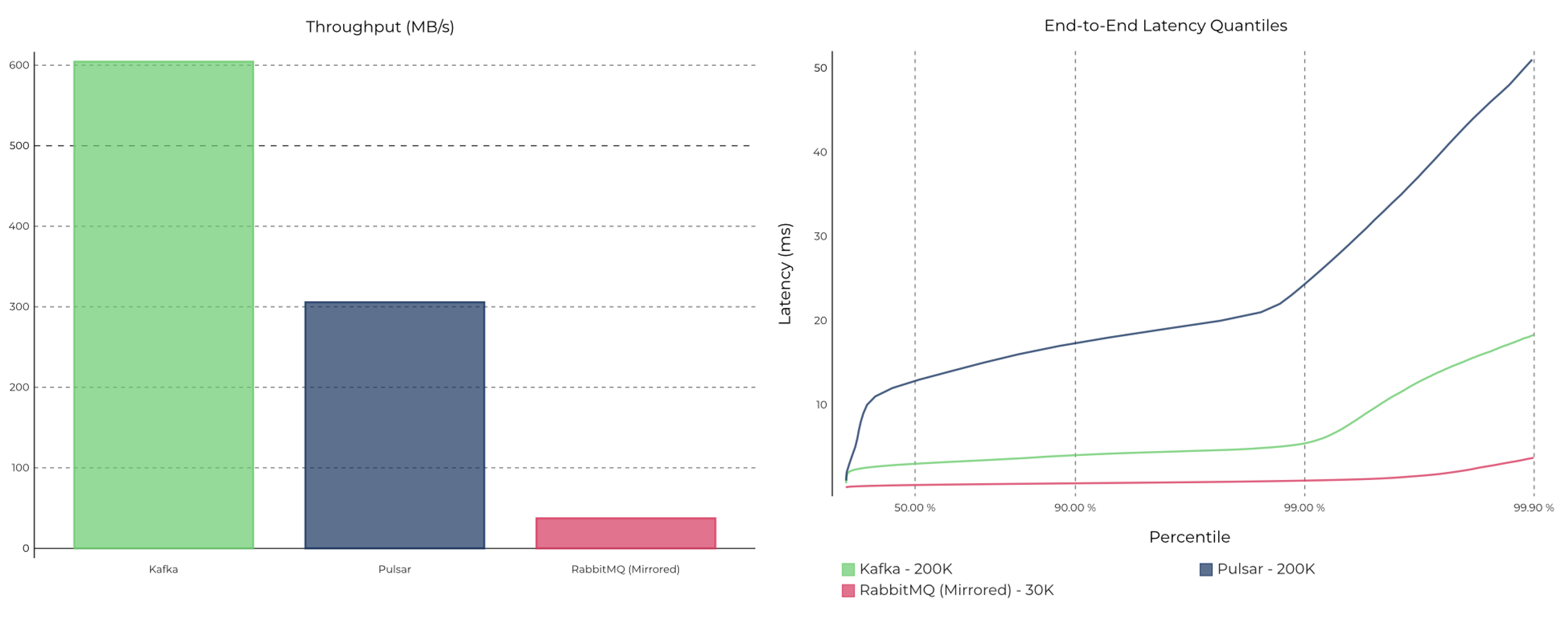
Như vậy, công nghệ lựa chọn cho thành phần này không còn gì khác là dịch vụ **Cloud Storage** hơn là Hadoop với các ưu điểm vượt trội trên.

#### **7.4.1.2. Real-time Message Ingestion**

Chúng tôi chọn Kafka là thành phần của hệ thống. Hiện tại trên thị trường, Kafka đang là phần mềm mã nguồn mở đứng đầu trong lĩnh vực real-time streaming với các ưu điểm:

* Độ trễ thấp: Apache Kafka cung cấp giá trị độ trễ thấp, tức là tối đa 10 mili giây. Đó là bởi vì nó tách rời thông điệp cho phép người tiêu dùng xem thông điệp đó bất cứ lúc nào.
* Thông lượng cao: Do ​​độ trễ thấp, Kafka có thể xử lý nhiều tin nhắn có khối lượng lớn và tốc độ cao. Kafka có thể hỗ trợ hàng nghìn tin nhắn trong một giây. Nhiều công ty như Uber sử dụng Kafka để tải một lượng lớn dữ liệu.
* Khả năng chịu lỗi: Kafka có tính năng cần thiết để cung cấp khả năng chống lỗi nút / máy trong cụm.
* Độ bền: Kafka cung cấp tính năng sao chép, giúp dữ liệu hoặc tin nhắn tồn tại nhiều hơn trên cụm trên đĩa. Điều này làm cho nó bền.
* Dễ dàng truy cập: Vì tất cả dữ liệu của chúng tôi được lưu trữ trong Kafka, nên bất kỳ ai trong hệ thống cũng có thể dễ dàng truy cập.
* Hệ thống phân tán: Apache Kafka chứa một kiến ​​trúc phân tán giúp nó có thể mở rộng. Phân vùng và sao chép là hai khả năng trong hệ thống phân tán.
* Xử lý thời gian thực: Apache Kafka có thể xử lý đường ống dẫn dữ liệu thời gian thực. Xây dựng đường ống dữ liệu thời gian thực bao gồm bộ xử lý, phân tích, lưu trữ, v.v.
* Phương pháp tiếp cận hàng loạt: Kafka sử dụng các trường hợp sử dụng giống như hàng loạt. Nó cũng có thể hoạt động giống như một công cụ ETL vì khả năng lưu trữ dữ liệu của nó.
* Khả năng mở rộng: Chất lượng của Kafka để xử lý một lượng lớn tin nhắn đồng thời khiến nó trở thành một sản phẩm phần mềm có khả năng mở rộng.

benchmark với các tool real-time streaming (RabbitMQ, Apche Pulsar) trên thị trường với kafka:



Hình 41. Benchmarking giữa kafka, rabbitmq và apache pulsar.



Hình 42. Bảng thống kê dựa trên hình 41.

Kết quả cho thấy **Kafka** cho thông lượng cao nhất với độ trễ tương đối thấp.

#### **7.4.1.3. Batch Processing & Stream Processing**

Chúng tôi chọn **Spark** là thành phần của hệ thống để phục vụ cho batch lẫn streaming process. Hiện tại trên thị trường, Spark là mã nguồn mỡ duy nhất hỗ trợ tối đa việc phân tích dữ liệu lớn. So với EMR, Spark tiết kiệm hơn nhiều và loại bỏ các chức năng không cần thiết, chỉ dùng để phục vụ phân tích, làm sạch dữ liệu:

Các lý do lựa chọn Spark cho hệ thống tại chỗ:

* Tiết kiệm chi phí.
* Công cụ process dữ liệu lớn tốt nhất trên hiện trường.
* Hỗ trợ máy học.
* Mã nguồn mở.
* Dễ thích ứng.

Chúng tôi không chọn EMR bởi vì:

* Gía thành rất cao, và tăng theo thời gian.
* Giao diện đẹp, đơn giản là ít cần thiết hơn nhu cầu chính là process dữ liệu khi đối tượng nhắm tới là data scientists.
* Không phù hợp khi chạy một công cụ riêng lẻ.
* Thiếu tính thích ứng. Mỗi khi cần nâng cấp hệ thống thì chúng ta phải đợi Amazon nâng cấp.
* Đối với Engineer thì không nâng cao được tay nghề.

Kết luận**, Spark là công cụ riêng lẻ tối ưu nhất** để phân tích dữ liệu cho máy học.

#### **7.4.1.4. Analytical Data Store**

Ở thành phần này, chúng tôi chọn **Cassandra** để lưu trữ dữ liệu có cấu trúc đã được làm sạch. Vậy tại sao phải là Cassandra mà không phải là cơ sở dữ liệu truyền thống (RDBMS) như SQL ?

Xét về khía cạnh NoSQL (Not only SQL) và RDBMS:

|  |  |  |
| --- | --- | --- |
|  | **NoSQL** | **RDBMS** |
| **Điểm quan trọng** | Mở rộng quy mô, tính khả dụng cao | Tính nhất quán |
| **Hiệu năng** | Mở rộng quy mô bằng cách xếp thêm máy chủ cạnh nhau. | Chia tỷ lệ hoặc chia nhỏ dữ liệu theo chiều ngang. |
| **Query** | Query với khóa đơn gỉan | Query bằng SQL |
| **Tính nhất quán** | Lỏng lẻo | Mạnh mẽ |
| **Data model** | Nhiều mô hình như định hướng cột, khóa, giá trị thuần túy. | Mô hình quan hệ |

Bảng 3. So sánh giữa MoSQL vs2 RDBMS.

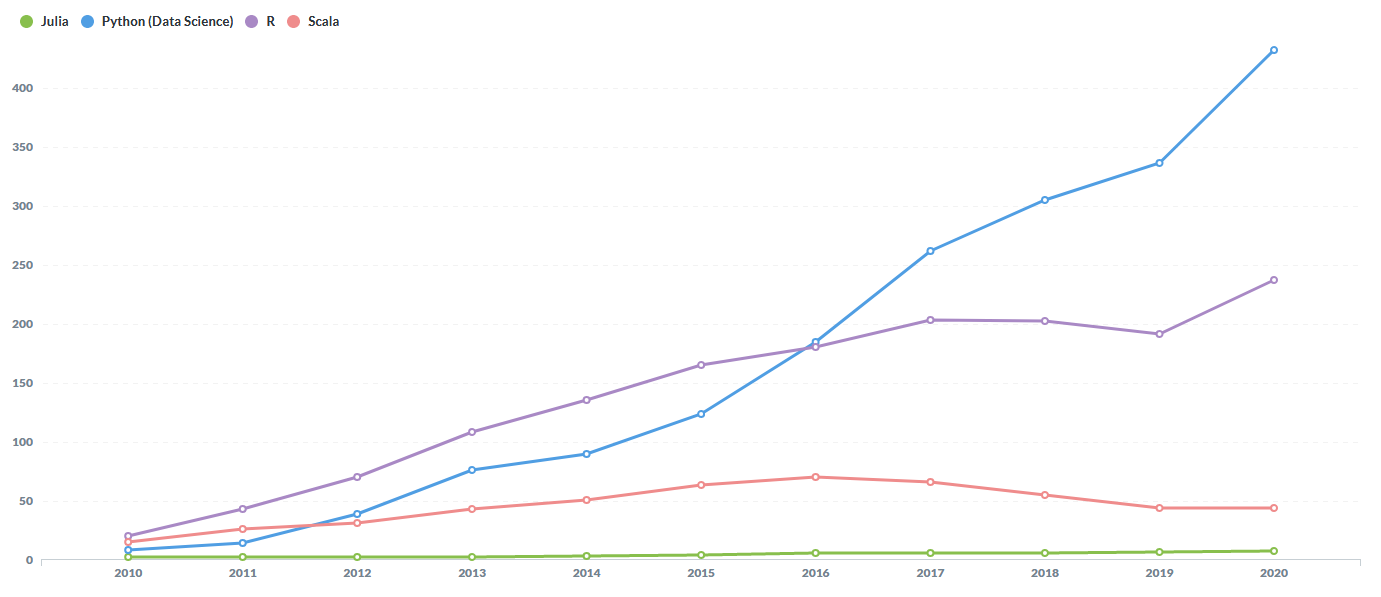
Ngoài các đặc điểm trên Cassandra còn mang những ưu việt:

* Mở rộng dễ dàng
* Khả năng chịu lỗi cao
* Kiến trúc không có SPOF (một điểm gây tổn hại)
* Mức độ tự do kiểm soát nhất quán
* Mô hình dữ liệu phong phú
* Có thể tăng cường cải thiện thông lượng cho tuyến tính
* Tính khả dụng cao
* Hỗ trợ các ngôn ngữ khác nhau dưới dạng client code
* Dễ dàng nắm bắt trạng thái bên trong của máy chủ bằng JMX/Dễ giám sát.

Quan trọng chính là: “Kiến trúc không có SPOF", "Mở rộng dễ dàng", “Khả năng chịu lỗi cao” và “Mô hình dữ liệu phong phú”. Chính 4 điều trên đã làm cho **Cassandra** có mặt trong hệ thống này.

#### **7.4.1.5. Analytical and Reporting**

Về component này, Jupyter notebook đã dành được vị trí.

Theo khảo sát tại trang <https://towardsdatascience.com/popularity-ranking-of-programming-languages-72bcf697ea20>, báo cáo từng năm cho thấy số lượng các data scientists sử dụng python gia tăng rất nhanh qua hằng năm và đứng đầu trong các năm từ 2016 đến nay: 

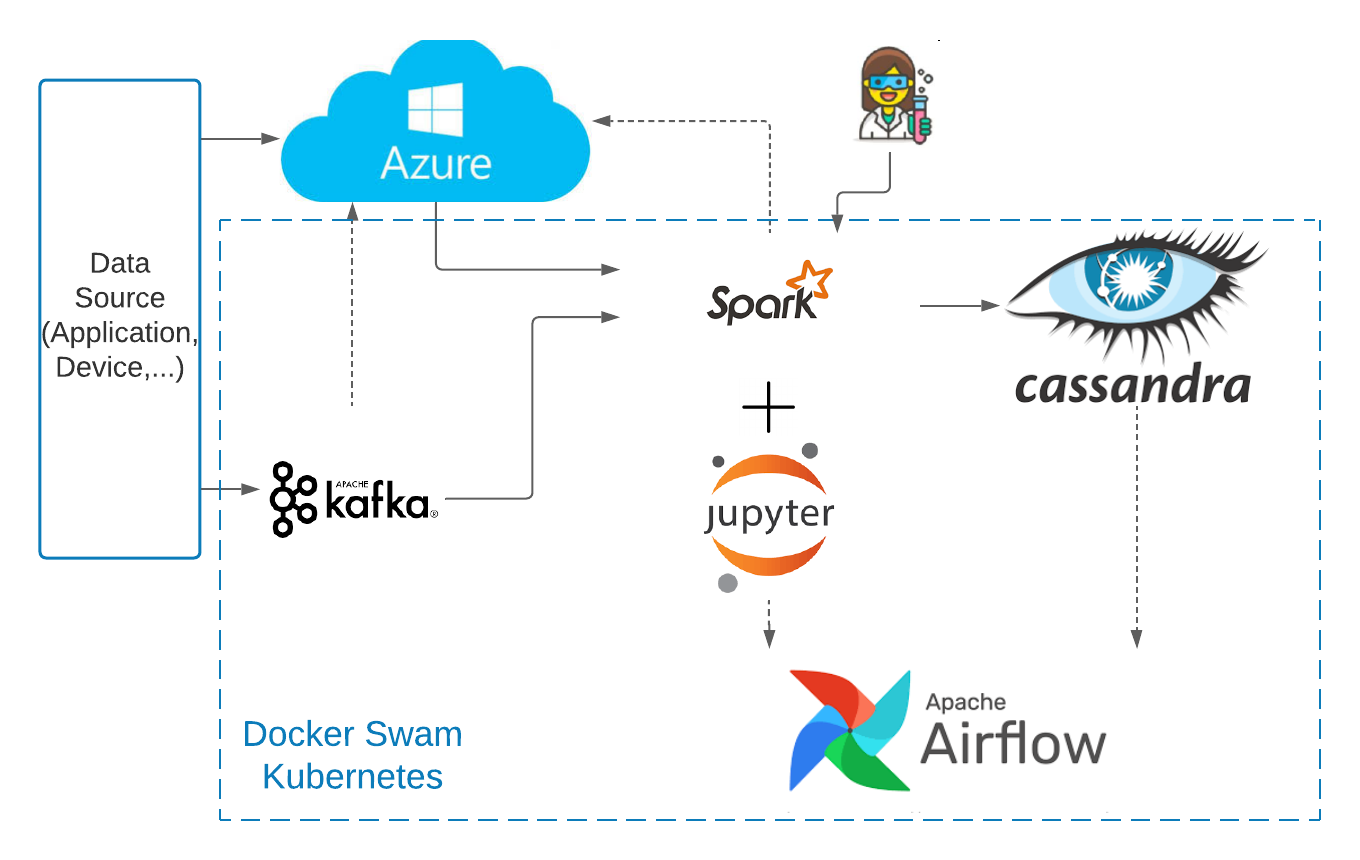
Hình 43. Các ngôn ngữ phổ biến cho data scientists.

Lý do thứ hai, jupyter notebook có thể được xây dựng on-top với apache spark, cung cấp giao diện làm việc dễ nhìn hơn là phải làm việc với môi trường console: pySpark. Mô hình hóa dữ liệu là điều mà console không thể làm được hoặc rất khó khăn.

Lý do cuối cùng, với jupyter notebook, ngoài phân tích dữ liệu, train model chúng ta còn có thể tạo báo cáo với các thư viện như Matplotlib, seaborn,… .

Với 3 lý do chính trên, chúng tôi đã chọn python cùng với **Jupyter Hub** làm hướng phát triển chính trên hệ thống.

### **4.4.2. Mô hình chi tiết hệ thống sau khi chọn công cụ**



Hình 44. Chi tiết sơ đồ hệ thống

## 7.5. Áp dụng hệ thống giải quyết bài toán nhận diện buồn ngủ

### **7.5.1. Đặt vấn đề**

Theo thống kê của Ủy ban An toàn giao thông quốc gia, những vụ tai nạn giao thông liên quan tới giấc ngủ chiếm tới 30% tổng các vụ giao thông trong một năm. Chỉ trong năm 2019, nhiều vụ tai nạn nghiêm trọng đã xảy ra do tài xế ngủ gật.

Ngoài ra, tại hội nghị Driver Care Day 2019, đã đưa ra các con số thống kê như sau:



Hình 45. Thống kê tai nạn giao thông và những tổn thất do ngủ gật gây ra.

Nghiên cứu chuyên sâu từ Hiệp hội An toàn giao thông AAA của Hoa Kỳ cho rằng việc lái xe khi ít ngủ hơn 5 tiếng một ngày cũng nguy hiểm không khác gì uống rượu khi lái xe. Nghiên cứu cũng cho rằng khá khó khăn trong việc phát hiện buồn ngủ khi lái xe, khiến đây trở thành vấn đề an toàn giao thông cấp bách hiện nay.

Ngoài ra, chúng tôi tin rằng buồn ngủ cũng có thể tác động tiêu cực đến mọi người trong môi trường làm việc và lớp học. Mặc dù thiếu ngủ và học đại học đi đôi với nhau, nhưng buồn ngủ ở nơi làm việc, đặc biệt là khi làm việc với máy móc nặng có thể dẫn đến chấn thương nghiêm trọng tương tự như những trường hợp xảy ra khi lái xe buồn ngủ.

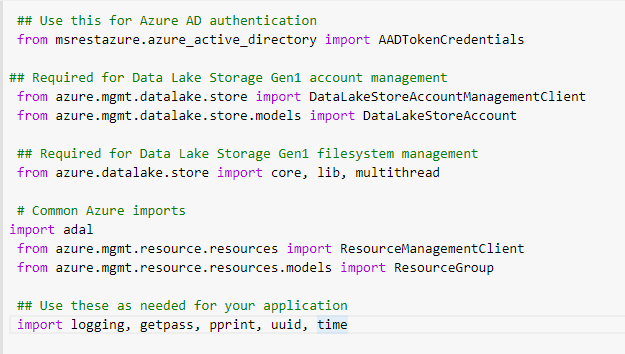
Giải pháp của chúng tôi cho vấn đề này là xây dựng một hệ thống phát hiện xác định các thuộc tính chính của buồn ngủ và kích hoạt cảnh báo khi ai đó buồn ngủ trước khi quá muộn.

### **7.5.2. Nguồn dữ liệu và các nghiên cứu ban đầu**

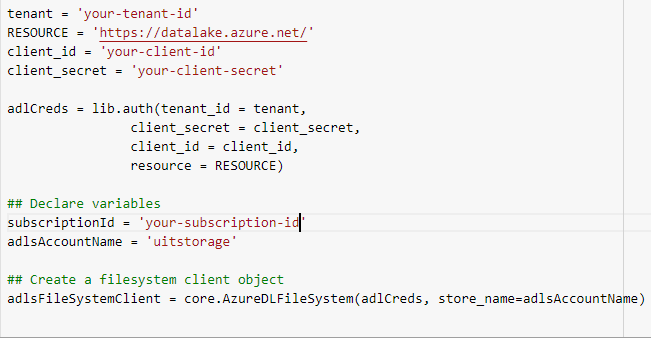
Đối với dữ liệu, chúng tôi đã sử dụng Bộ dữ liệu về cơn buồn ngủ trong đời thực do một nhóm nghiên cứu từ Đại học Texas tại Arlington tạo ra để phát hiện tình trạng buồn ngủ nhiều giai đoạn [11]. Mục tiêu cuối cùng là không chỉ phát hiện các trường hợp buồn ngủ cực độ mà còn cho phép hệ thống của chúng tôi phát hiện các tín hiệu buồn ngủ nhẹ nhàng hơn. Bộ dữ liệu bao gồm khoảng 30 giờ video của 60 người tham gia khác nhau. Từ tập dữ liệu, chúng tôi có thể trích xuất các điểm mốc trên khuôn mặt từ 69 video của 23 người tham gia. Điều này cho phép chúng tôi có đủ lượng dữ liệu cho cả trạng thái cảnh báo và buồn ngủ.

Đầu tiên chúng ta cần phải đăng nhập vào azure để lấy dữ liệu:

Import các gói từ azure:

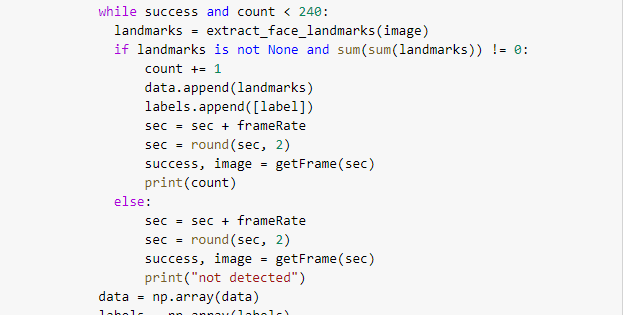


Lấy các thông tin cơ bản như sau từ azure cloud về nhập vào:

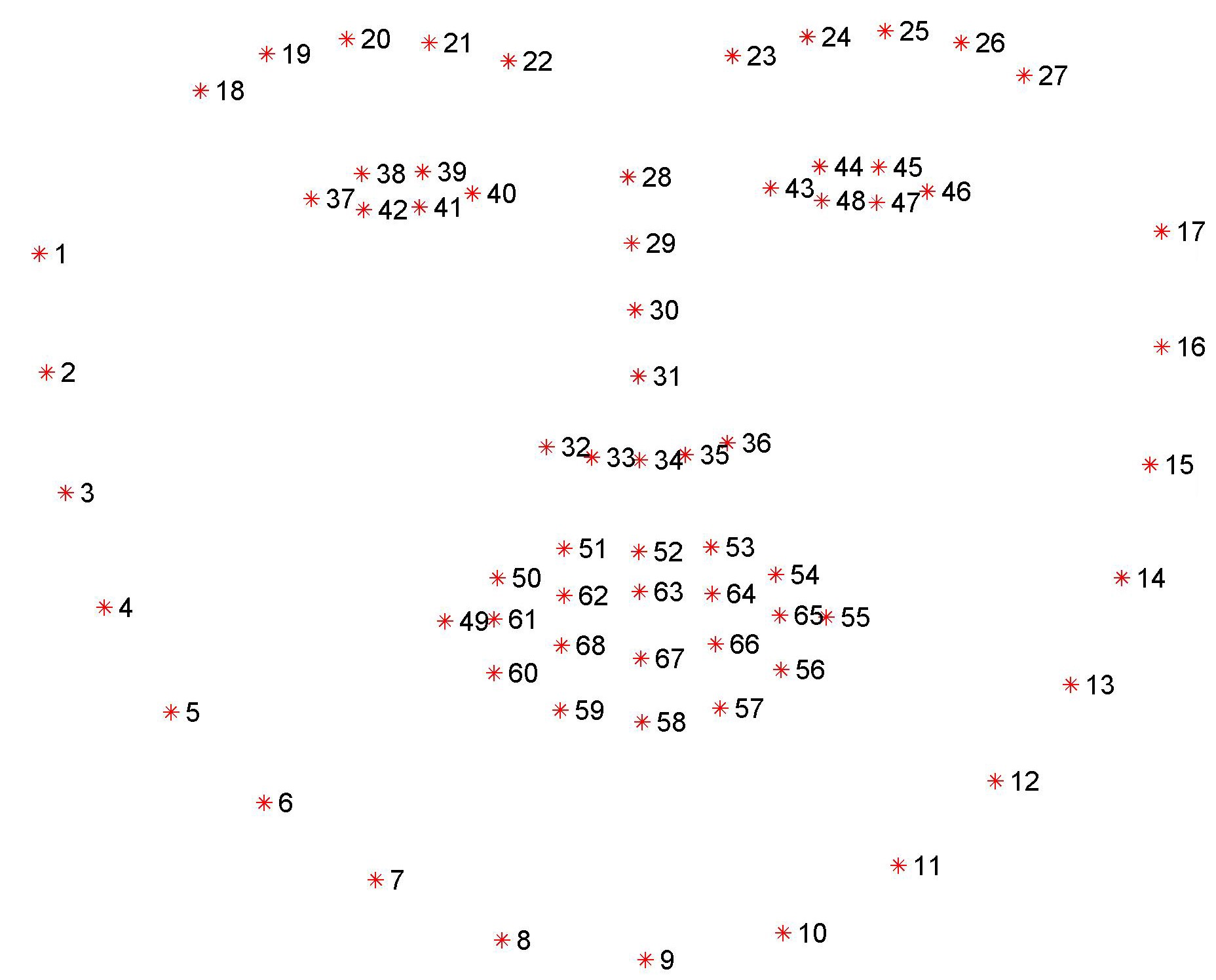


Đối với mỗi video, chúng tôi đã sử dụng OpenCV để trích xuất 1 khung hình mỗi giây bắt đầu từ mốc 3 phút cho đến khi kết thúc video:





Mỗi video dài khoảng 10 phút, vì vậy chúng tôi trích xuất khoảng 240 khung hình cho mỗi video, tổng cộng có 16560 khung hình cho toàn bộ tập dữ liệu.



Hình 46. Các điểm trên khuôn mặt. [12]

Có tổng cộng 68 điểm mốc trên mỗi khung hình nhưng chúng tôi quyết định chỉ giữ lại các điểm mốc cho mắt và miệng (Điểm 37–68). Đây là những điểm dữ liệu quan trọng mà chúng tôi đã sử dụng để trích xuất các tính năng cho model của mình.

### **7.5.3. Bóc tách dữ liệu**

Như đã đề cập ngắn gọn trước đó, dựa trên các điểm mốc khuôn mặt mà chúng tôi trích xuất từ khung hình của video, chúng tôi đã phát triển các thuộc tính phù hợp cho mô hình phân loại của mình.

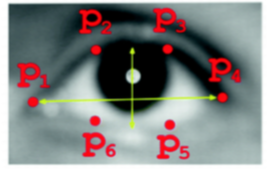
Trong quá trình tìm hiểu các bài viết khoa học liên quan, chúng tôi đúc kết được 2 thuộc tính cơ bản đó là tỉ lệ mắt và tỉ lệ miệng.Tuy nhiên để cải tiến, sau nhiều ngày suy nghĩ chúng tôi cũng thêm vào 2 thuộc tính khác mà chúng tôi nghĩ chúng đều có ý nghĩa đó là: độ tròn đồng tử mắt và kết quả giữa tỉ lệ miệng so với tỉ lệ mắt.

Như vậy, đã có tất cả 4 thuộc tính sẽ được bóc ra từ 1 khung hình đó là:

* Tỉ lệ mắt
* Tỉ lệ miệng
* Độ tròn đồng tử ở mắt
* Tỉ số giữa tỉ lệ miệng so với tỉ lệ mắt.

#### **7.5.3.1. Tỉ lệ mắt (Eye Aspect Ratio - EAR)**

EAR là tỷ lệ giữa chiều dài của mắt với chiều rộng của mắt. Chiều dài của đôi mắt được tính bằng khoảng cách từ khóe mắt trong cho tới đuôi bên ngoài. Chiều rộng của mắt thì được tính bằng tổng 2 khoảng cách giữa mí mắt trên so với mí mắt dưới. Hình sau đây sẽ cho các bạn thấy trực quan hơn:

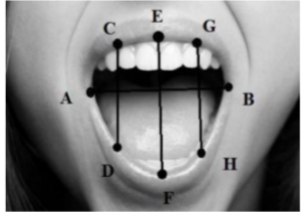


Hình 47. Tỉ lệ EAR. [13]

Giả thuyết của chúng tôi là khi một người buồn ngủ, mắt của họ có sẽ khép lại và có khả năng chớp mắt nhiều hơn. Dựa trên giả thuyết này, chúng tôi mong đợi mô hình của mình dự đoán là buồn ngủ nếu tỷ lệ khung hình của mắt trên các khung hình liên tiếp bắt đầu giảm, tức là mắt của họ bắt đầu nhắm nhiều hơn hoặc họ chớp nhanh hơn.

#### **7.5.3.2. Tỉ lệ miệng (Mouth Aspect Ratio - MAR)**

Về mặt tính toán tương tự như EAR, MAR là tỷ lệ giữa chiều dài của miệng với chiều rộng của miệng. Chiều dài của miệng sẽ bằng từ khóe miệng bên trái đến khóe miệng bên phải. Và chiều rộng của miệng sẽ bằng khoảng cách từ điểm cao nhất ở môi trên tới điểm thấm nhất môi dưới. Hình minh họa:

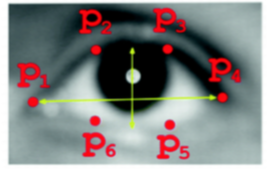


Hình 48. Tỉ lệ MAR. [13]

Giả thuyết của chúng tôi là khi một cá nhân trở nên buồn ngủ, họ có khả năng ngáp và mất kiểm soát miệng, khiến MAR của họ cao hơn bình thường trong trạng thái này.

#### **7.5.3.3. Độ tròn đồng tử (Pupil Circularity - PUC)**

PUC là một thuộc tính bổ sung cho EAR, nhưng nó tập trung nhiều hơn vào đồng tử thay vì toàn bộ mắt. Công thức này tham khảo từ các research đã có trước, nguồn sẽ được trích dẫn ở chương cuối.



Hình 49. Độ tròn đồng tử [14]

Ví dụ, một người nhắm mắt hoặc gần như nhắm lại sẽ có độ tròn đồng tử thấp hơn nhiều so với người mở mắt hoàn toàn do số hạng bình phương trong mẫu số. Tương tự như EAR, kỳ vọng rằng khi một cá nhân buồn ngủ, khả năng tròn của đồng tử của họ có thể giảm.

#### **7.5.3.4. Tỉ số giữa tỉ lện miệng so với tỉ lệ mắt (Mouth aspect ratio Over Eye aspect ratio - MOE)**

Cuối cùng, chúng tôi quyết định thêm MOE làm một thuộc tính khác. MOE đơn giản là tỷ số giữa MAR so với EAR.

Lợi ích của việc sử dụng tính năng này là EAR và MAR dự kiến sẽ di chuyển ngược chiều nhau nếu trạng thái của cá nhân thay đổi. Trái ngược với cả EAR và MAR, MOE như một thước đo phản ứng sẽ nhanh hơn với những thay đổi này vì nó sẽ nắm bắt những thay đổi tinh tế trong cả EAR lẫn MAR và sẽ phóng đại những thay đổi khi mẫu số và tử số di chuyển theo hướng ngược nhau. Lý thuyết của chúng tôi là khi cá nhân buồn ngủ, MOE sẽ tăng lên.

#### **7.5.3.5. Kết quả**

Mặc dù tất cả các tính năng này có ý nghĩa trực quan, nhưng khi được thử nghiệm với các mô hình phân loại, chúng mang lại kết quả kém trong phạm vi độ chính xác từ 40% đến 55%. Tuy nhiên, sự thất vọng này đã đưa chúng tôi đến khám phá sâu hơn. Các tính năng không sai, chỉ là chúng không được xem xét chính xác.

### **7.5.4. Normalize các feature**

Sao một tuần quan sát tập dữ liệu, thử nghiệm thì chúng tôi nhận ra rằng khi mà dữ liệu tách ra tập test và tập train theo 1 cách ngẫu nhiên thì độ chính xác vào khoảng hơn 60%. Tuy nhiên nếu chia tập dữ liệu theo từng cá nhân ( ví dụ: người trong tập test sẽ không có trong tập train) thì độ chính xác chỉ từ 40-50%. Từ đó chúng tôi kết luận model hiện tại đang gặp khó khan với những gương mặt không có trong tập train.

Từ kết luận trên, chúng tôi phải bỏ 1 tuần xem lại tập dữ liệu và nhận ra điều cơ bản rằng đặc điểm khuôn mặt của mỗi người là khác nhau vậy nên dẫn đến các thông số cũng khác nhau. Ví dụ người A bình thường có đôi mắt híp so với người B mắt to thì nếu dữ liệu train có B nhưng không có A điều đó sẽ dẫn đến model dự đoán người A trong trạng thái buồn ngủ vì các thông số EAR và PUC sẽ thấp còn MOE cao mặc dù họ vẫn tỉnh. Dựa trên khám phá này chúng tôi giả định rằng việc normalize các thuộc tính cho từng cá nhân có thể sẽ giúp cải thiện kết quả.

Để normalize các thuộc tính của từng cá nhân, chúng tôi đã lấy ba khung hình đầu tiên của mỗi cá nhân và sử dụng chúng làm cơ sở để normalize. Có nghĩa là với 3 khung hình đó, chúng tôi sẽ tính toán giá trị trung bình và độ lệch chuẩn của từng thuộc tính và xem đó là tiêu chuẩn để áp dụng vào các khung hình sau của đối tượng, cá nhân đó.Về mặt toán học, đây là phương trình chuẩn hóa trông giống như sau: [15]

Trong đó:

* a: Là tên thuộc tính
* b: Là tên cá nhân
* : Là giá trị trung bình thuộc tính a của cá nhân b trong 3 khung hình đầu tiên.
* : Là độ lệch chuẩn thuộc tính a của cá nhân b trong 3 khung hình đầu tiên
* : Gía trị thuộc tính a của đối tượng b.

Như vậy, sau khi chuẩn hóa, chúng tôi thu được 8 thuộc tính bao gồm 4 thuộc tính cơ bản ban đầu và 4 thuộc tính sau khi chuẩn hóa.

Tuy nhiên chưa dừng lại ở đó, trong quá trình tìm hiểu ở giai đoạn này chúng tôi còn nhận ra được 1 điều khác. Đó là các label được gắn không chính xác.

### **7.5.5. Đánh label lại ban đầu**

Theo tập dữ liệu ban đầu, các cá nhân sẽ có 3 video, tên của video sẽ được gắn nhãn sẵn ứng với mức độ buồn ngủ của người trong video đó:

* 0: tỉnh táo.
* 5: lờ đờ.
* 10: ngủ.

Ban đầu, chúng tôi mặc định nhãn dán mà họ đã gắn sẵn làm label cho từng khung hình cũng như từng hàng giá trị mà đã được bóc tách ra. Tuy nhiên, thực chất không đơn giản như vậy, khoảng 50-60% các video gắn nhãn là 5-10, các cá nhân vẫn cho thấy sự tỉnh táo trong 1 giai đoạn nào đó, có người còn thậm chí tỉnh táo hẳn. Điều này làm sai lệch độ chính xác đi rất nhiều. Bởi vì như đã nói ban đầu, mặc định gắn nhãn cho từng khung hình tách ra từ video bằng chính tên của nó. Ví dụ như các khung hình tách từ video tên 10.mp4 thì sẽ đều gắn nhãn là 10 (tức đã ngủ). Vì vậy, chúng tôi đã bỏ một khoản lớn thời gian (~2 tuần) để gắn nhãn lại từ đầu. Thay vì nhãn là 0-5-10 giờ chỉ còn lại 0-1.

Kết quả sau khi chuẩn hóa kết hợp với gắn nhãn lại, chúng tôi đã đạt được độ chính xác khoảng 89 ~ 93%.

### **7.5.6. Kết quả với các model phân loại cơ bản**

Sau khi trích xuất và chuẩn hóa các tính năng của mình, chúng tôi muốn thử nghiệm trên nhiều model khác nhau, bắt đầu với các mô hình phân loại cơ bản nhất như hồi quy logistic và Naive Bayes rồi sau đó chuyển sang các mô hình phức tạp hơn chứa mạng nơ-ron và các phương pháp học sâu khác. Để đào tạo và kiểm tra các mô hình của mình, chúng tôi chia bộ dữ liệu của mình theo quy tắc cơ bản: 80% dữ liệu train và 20% dữ liệu test. Như vậy tập dữ liệu train bao gồm: 19 video với 13680 khung hình và tập dữ liệu test 4 video với 2880 khung hình.

Một thách thức mà chúng tôi phải đối mặt trong dự án đó là làm thế nào để phân biệt giữa nháy mắt và nhắm mắt. Để giải quyết vấn đề trên thì chúng ta cần dự đoán chuỗi hành động (gồm nhiều frame). Trong khi các mô hình phức tạp như LSTM và RNN có thể làm điều này, thì các mô hình phân loại cơ bản không thể.

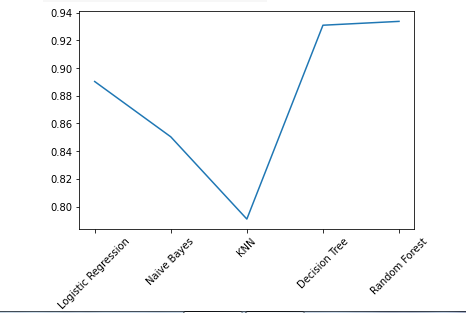
Cách chúng tôi xử lý vấn đề này là lấy trung bình cộng kết quả dự đoán ban đầu với kết quả dự đoán từ hai khung trước. Tính trung bình có ý nghĩa trong trường hợp này và cho phép chúng tôi đưa ra các dự đoán chính xác hơn bởi vì tập dữ liệu của chúng tôi được sắp xếp theo từng cá nhân và theo thứ tự thời gian trong video.



Hình 50. Phương pháp predict tuần tự cho model phân loại cơ bản.

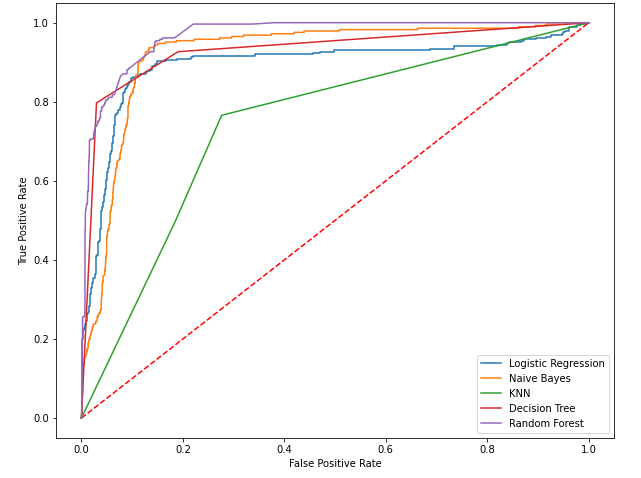
**Kết quả trên các model:**

Độ chính xác (Accuracy):



Hình 51. Đồ thi đường độ chính xác của từng model.

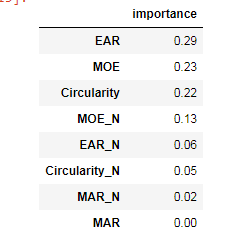
ROC (Receiver Operating Characteristic curve):



Hình 52. ROC curve.

### **7.5.7. Độ quan trọng từng thuộc tính**

Sau khi kiểm thử, chúng tôi muốn tổng kết lại để đánh giá từng thuộc tính (feature). Chúng tôi chọn ngẫu nhiên model random forest làm thí nghiệm.



Hình 53. Độ quan trong từng feature.

EAR (Eye Aspect Ratio) có vẻ là thuộc tính quan trong nhất trong cả 8 thuộc tính. Điều này hoàn toàn có lý khi chúng ta buồn ngủ, đều có xu hướng híp mắt lại. Tuy nhiên MAR (Mouth Aspect Ratio) không được xét là bởi vì, sau khi xem từng video, các đối tượng đều che miệng khi ngáp. Mặc dù vậy model vẫn có khả năng nhận biết vì mắt họ sẽ hẹp lại khi ngáp (thói quen tự nhiên của khuôn mặt).

# Chương 8. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết quả đạt được

* Làm rõ định nghĩa và xác định đúng Dữ liệu lớn là gì? Các khái niệm chuyên ngành cần biết đến.
* Phân tích các phần cốt lõi để tạo nên một hệ thống Dữ liệu lớn.
* Thiết kế chi tiết hệ thống dữ liệu lớn (mức độ có thể triển khai) bao gồm: Microsoft Azure DataLake Gen1, Apache Kafka, Apache Spark, Apache Cassandra, Apache Airflow và Jupyter Hub.
* Áp dụng thuật toán máy học nhận diện sự buồn ngủ và tích hợp vào hệ thống dữ liệu lớn trên.
* Độ chính xác của model đạt được sau khi train lên đến 93%.

## Nhận xét

### **8.2.1. Ưu điểm**

#### **8.2.1.1. Hệ thống dữ liệu lớn**

* Nghiên cứu nhiều khía cạnh của ngành Dữ liệu lớn.
* Đem lại góc nhìn tổng qua và hướng đi trong ngành Khoa học dữ liệu.
* Triểm khai được prototype Dữ liệu lớn vào bài toán thực tế trong hệ thống máy học.
* Hệ thống dữ liệu gồm nhiều module có thể bổ sung hoặc tháo gỡ dễ dàng.

#### **8.2.1.2. Model máy học**

- Đạt được độ chính xác cao.

- Gỉai quyết vấn đề thực tế.

### **8.2.2. Khuyết điểm**

#### **8.2.2.1. Hệ thống dữ liệu lớn**

* Chưa có kết quả benchmark giữa các framework hỗ trợ dữ liệu với nhau.
* Chưa tích hợp thử vào nguồn dữ liệu: ứng dụng, camera,….

#### **8.2.2.2. Model máy học**

* Thử nghiệm dataset trên các model nâng cao hơn như: LSTM, VGG16.

## Hướng phát triển

* Phát triển ứng dụng để tích hợp vào hệ thống dữ liệu.
* Tiếp tục phát triển tập dữ liệu trên các model hiện đại.
* Triển khai thử hệ thống trên Azure Kubernetes.

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | Apache Software Foundation, “Apache Hadoop,” Apache Software Foundation, [Trực tuyến]. Available: https://hadoop.apache.org/. |
| [2] | Apache Software Foundation, “https://zookeeper.apache.org/,” Apache Software Foundation, [Trực tuyến]. Available: https://zookeeper.apache.org/. |
| [3] | Apache Software Foundation, “Apache Kafka,” Apache Software Foundation, [Trực tuyến]. Available: https://kafka.apache.org/. |
| [4] | Apache Software Foundation, “Apache Cassandra,” Apache Software Foundation, [Trực tuyến]. Available: https://cassandra.apache.org/. |
| [5] | Apache Software Foundation, “Apache Spark,” Apache Software Foundation, [Trực tuyến]. Available: https://spark.apache.org/. |
| [6] | Apache Software Foundation, “Apache Airflow,” Apache Software Foundation, [Trực tuyến]. Available: https://airflow.apache.org/. |
| [7] | Amazon Web Service, Inc., “Amazon S3,” Amazon Web Service, Inc., [Trực tuyến]. Available: https://aws.amazon.com/vi/s3/. |
| [8] | Amazon Web Service, Inc., “Amazon EC2,” Amazon Web Service, Inc., [Trực tuyến]. Available: https://aws.amazon.com/vi/ec2. |
| [9] | Amazon Web Services, Inc, “Amazon EMR,” Amazon Web Services, Inc, [Trực tuyến]. Available: https://aws.amazon.com/vi/emr/. |
| [10] | Docker, Inc., “Docker,” Docker, Inc., [Trực tuyến]. Available: https://docs.docker.com/. |
| [11] | The University of Texas at Arlington, “UTA Real-Life Drowsiness Dataset,” [Trực tuyến]. Available: https://sites.google.com/view/utarldd/home. |
| [12] | C. Sagonas, “Facial point annotations,” Intelligent Behaviour Understanding Group (iBUG), [Trực tuyến]. Available: https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/. |
| [13] | N. G. A. A. N. N. Isha Gupta, “Real-Time Driver's Drowsiness Monitoring Based on Dynamically Varying Threshold,” trong *Eleventh International Conference on Contemporary Computing (IC3)*, 2018. |
| [14] | M. J. Bottema, “Circularity Of Objects In Images,” Flinders University of South Australia, Adelaide. |
| [15] | Wikipedia, “Wikipedia - Standard score,” [Trực tuyến]. Available: https://en.wikipedia.org/wiki/Standard\_score. |