HealthDL - Một hệ thống thu thập, lưu trữ dữ liệu y tế lớn

Phan Tân[1*], Trần Việt Trung[1], Nguyễn Hữu Đức[1], Nguyễn Thanh Tùng[2,3]

[1]Viện Công nghệ thông tin và Truyền thông, Trường Đại học Bách khoa Hà Nội
[2]Khoa Quốc tế, Đại học Quốc gia Hà Nội
[3]Trường Đại học Nguyễn Tất Thành

Ngày nhận bài 29/5/2017; ngày chuyển phản biện 1/6/2017; ngày nhận phản biện 26/6/2017; ngày chấp nhận đăng 30/6/2017

Tóm tắt:

Hiện nay, ứng dụng công nghệ lưu trữ lớn, khai phá dữ liệu trong lĩnh vực y tế để chẩn đoán, phòng ngừa và điều trị bệnh nhằm can thiệp nâng cao tình trạng sức khỏe con người là hướng nghiên cứu có nhu cầu thực tiễn. Trong nghiên cứu này, các tác giả giới thiệu HealthDL - Một hệ thống thu thập và lưu trữ dữ liệu y tế. HealthDL có kiến trúc phân tán, xây dựng từ các thành phần khả mở cho dữ liệu lớn, gia tăng liên tục theo thời gian thực. Hệ thống tối ưu cho dữ liệu nhận về từ các thiết bị y sinh và dữ liệu thông tin lịch sử bệnh án đến từ hàng triệu thiết bị phân tán về mặt địa lý. Các thử nghiệm, đánh giá trên hệ thống thực bằng các công cụ đánh giá tiêu chuẩn với dữ liệu mô phỏng cho kết quả tốt.

Từ khóa: *Cơ sở dữ liệu phân tán, dịch tễ học, dữ liệu lớn, thời gian thực.*

Chỉ số phân loại: *2.2*

HealthDL - A system for collecting and storing big data in the medical field

Tan Phan[1*], Viet Trung Tran[1], Huu Duc Nguyen[1], Thanh Tung Nguyen[2,3]

[1]School of Information and Communication Technology, HUST
[2]VNU - International School (VNU-IS)
[3]Nguyen Tat Thanh University

Received 29 May 2017; accepted 30 June 2017

Abstract:

At present, applications of large storage technology and data mining in the medical field to diagnose, prevent, and treat diseases with the purpose of improving human health are preferred the practical researches. In this study, we would like to introduce HealthDL: A system for collecting and storing medical data. HealthDL has a dispersal architecture, is built from open components for large data, and continuously increases in real time. An optimal system retrieves database from biomedical devices and medical record database coming from millions of geographically dispersed devices. Tests and evaluations on the real system using standard assessment tools with simulated database show good results.

Keywords: *Big data, dispersed database, epidemiology, real time.*

Classification number: *2.2*

Mở đầu

Hiện nay, ứng dụng công nghệ lưu trữ lớn, khai phá dữ liệu trong lĩnh vực y tế để chẩn đoán, phòng ngừa và điều trị bệnh nhằm can thiệp nâng cao sức khỏe con người là hướng nghiên cứu có nhu cầu thực tiễn, được quan tâm tích cực bởi cộng đồng các nhà nghiên cứu. Tại Việt Nam, các thiết bị y sinh (thiết bị đo thông số tim mạch, huyết áp…) được sử dụng rộng rãi để giám sát người bệnh theo thời gian thực. Các thiết bị này đưa ra các thông số đo có tính liên tục, cập nhật theo chu kỳ 1 lần/s. Hiện nay, nguồn dữ liệu này chưa được lưu trữ tập trung vì những thách thức liên quan đến độ lớn của dữ liệu đến từ hàng triệu thiết bị đo, liên tục và phân tán rộng về mặt địa lý.

Bên cạnh nguồn dữ liệu từ các thiết bị y sinh, hiện trạng các hồ sơ bệnh án cũng đang được lưu trữ một cách rải rác, không có sự chia sẻ. Do đó, các bác sỹ, trung tâm y tế khác nhau gặp nhiều khó khăn để tìm kiếm thông tin lịch sử bệnh tật của bệnh nhân để đưa ra phác đồ điều trị hợp lý. Vì vậy, việc lưu trữ các bệnh án tập trung cũng là một nhu cầu cấp thiết để phục vụ công tác phân tích, chẩn đoán.

Theo Ercan và Lane (2014) [1], trong những hệ thống bệnh án điện tử (Electronic Health Record - EHR) truyền

[*]Tác giả liên hệ: Email: phantan@gmail.com

thống, data được lưu trữ dưới dạng bản ghi trong bảng cơ sở dữ liệu quan hệ. Nghiên cứu cũng chỉ ra rằng, sự mở rộng ngày càng tăng của lượng thông tin trong các dịch vụ chăm sóc sức khỏe đã dẫn đến một nút thắt nghẽn cổ chai trong việc lưu trữ, truy xuất và đặt ra những thách thức về tính sẵn sàng cao đối với việc sử dụng mô hình cơ sở dữ liệu truyền thống. Hơn nữa trong nghiên cứu của Andreu-Perez và cs (2015) [2], các tác giả đã chỉ ra rằng sự đa dạng của dữ liệu y tế ngày càng tăng với sự phát triển của công nghệ, dữ liệu từ sensor, mobile, các dữ liệu ảnh... đòi hỏi phải nghiên cứu tìm kiếm một cách thức tổ chức lưu trữ dữ liệu y tế phù hợp. Nghiên cứu của Ercan và Lane (2014), Dobre và Xhafa (2015) [1, 3] đã chỉ ra những yêu cầu thiết yếu của EHR và đề xuất sử dụng mô hình dữ liệu phi quan hệ (NoSQL [4]) cho bài toán lưu trữ và xử lý dữ liệu lớn về y tế. Tuy nhiên những nghiên cứu này chỉ đưa ra gợi ý hướng tiếp cận chung, không đưa ra thiết kế tổng thể về thu thập, lưu trữ EHR, không thử nghiệm, cài đặt và đánh giá hiệu năng của hệ thống. Nghiên cứu cũng thống nhất rằng cơ sở dữ liệu khoá - giá trị (một dạng của NoSQL) có thể là một giải pháp tốt cho dữ liệu đơn giản, truy xuất thông qua khoá, các giá trị là đồng nhất về cấu trúc, không có cập nhật hay truy xuất dựa trên cấu trúc của giá trị dữ liệu. Ví dụ như thông số an sinh xã hội của khách hàng, thông tin số bảo hiểm y tế. Cơ sở dữ liệu hướng văn bản (một dạng của NoSQL) là một giải pháp cho việc lưu trữ hồ sơ bệnh án y tế, bao gồm hồ sơ bệnh nhân, báo cáo nghiên cứu, báo cáo phòng thí nghiệm, hồ sơ bệnh viện, các báo cáo bản chụp X quang, bản chụp cắt lớp… Các tác giả Ercan và Lane (2014) [1] đề xuất sử dụng Dynamo Amazon [5], một dịch vụ lưu trữ dữ liệu điện toán đám mây của Amazon để lưu trữ luồng dữ liệu liên tục từ các thiết bị y sinh. Kiến trúc Dynamo Amazon dựa trên hàm băm nhất quán cho cơ chế mở rộng, sử dụng nút ảo để phân tán dữ liệu đồng đều trên các nút máy chủ vật lý, sử dụng vector clock [6] để giải quyết xung đột giữa các phiên bản dữ liệu sau quá trình ghi tương tranh đồng thời.

HealthDL là một hệ thống tổng thể, ngoài thành phần lưu trữ dữ liệu theo mô hình NoSQL như các nghiên cứu liên quan, HealthDL tích hợp hệ thống hàng chờ thông điệp phân tán để thu thập luồng dữ liệu đến từ các thiết bị y sinh phân tán về mặt địa lý. HealthDL được xây dựng và triển khai thực nghiệm, đánh giá hiệu năng lưu trữ trên môi trường phân tán.

Trong nghiên cứu này, chúng tôi giới thiệu HealthDL - Một hệ thống phân tán, thu thập và lưu trữ dữ liệu y tế tối ưu cho dữ liệu nhận về từ các thiết bị y sinh và dữ liệu thông tin lịch sử bệnh án. Xây dựng HealthDL là một bài toán nghiên cứu về dữ liệu lớn, gia tăng liên tục theo thời gian thực và đến từ hàng triệu thiết bị phân tán về mặt địa lý. HealthDL cho phép thống kê, tìm kiếm và phân tích dữ liệu. HealthDL hỗ trợ công việc của một chuyên viên dịch tễ học trong công tác thu thập, phân tích dữ liệu và phỏng đoán bệnh.

Đặc trưng dữ liệu y tế

Dữ liệu y tế trong nghiên cứu này được phân thành 2 nhóm: Dữ liệu thu thập từ các thiết bị y sinh và dữ liệu thông tin bệnh án. Phần dưới đây mô tả đặc trưng nguồn dữ liệu đầu vào cho hệ thống HealthDL.

Dữ liệu y sinh

Các thiết bị y sinh như máy đo nhịp tim, huyết áp gửi thông số đo khoảng 540 bytes/s. Trung bình mỗi bệnh nhân đo 2 tiếng/ngày, như vậy số lượng gói dữ liệu sinh ra trong 1 tháng ứng với 1.000 bệnh nhân là 216.000.000 gói dữ liệu. Mỗi gói trung bình 540 bytes, xét trên 1.000 bệnh nhân với các máy đo độc lập, lượng dữ liệu thu thập được tương ứng là 116 Gigabytes về thông số bệnh nhân trong một tháng.

Dữ liệu bệnh án

Xét dữ liệu bệnh án tìm hiểu trên 4 nhóm bệnh, gồm: Bệnh tăng huyết áp (kích thước bản ghi 800-1.000 bytes); bệnh lao phổi (kích thước bản ghi 400-600 bytes); bệnh hen phế quản (kích thước bản ghi 500-700 bytes); bệnh đái tháo đường (kích thước bản ghi 800-1.000 bytes). Đặc điểm của dữ liệu bệnh án là cấu trúc dữ liệu rất linh hoạt. Đối với bệnh nhân đái tháo đường, một văn bản bệnh án sẽ có khoảng 150 trường dữ liệu riêng biệt với cấu trúc chia nhỏ từ 4 đến 5 tầng (bảng 1). Đối với bệnh nhân tăng huyết áp, một văn bản bệnh án sẽ có khoảng 75 trường dữ liệu riêng biệt với cấu trúc chia nhỏ đến 4-5 tầng.

Bảng 1. Một phần bệnh án đái tháo đường (ĐTĐ).

II. HỎI BỆNH

A. Bệnh sử: Lưu ý: Tuổi xuất hiện < 30 tuổi → ĐTĐ typ 2; Nếu là nữ có thai hay không?

B. Tiền sử:

❖ Tiền sử bản thân

1. Bản thân có mắc bệnh ĐTĐ, hoặc tăng huyết áp (THA), huyết áp ≥ 140/90 mmHg hoặc đang điều trị THA?
   Không ☐ **Có** ☐ Bệnh gì:.....
2. Có thừa cân béo phì (chiều cao, cân nặng, BMI) kèm theo lối sống ít vận động
   Không ☐ **Có** ☐
3. Bố hoặc mẹ có mắc ĐTĐ?
   Không ☐ **Có** ☐
4. Phụ nữ có hội chứng buồng chứng đa nang
   Không ☐ **Có** ☐
5. Phụ nữ đã sinh con ≥ 4 kg hoặc được chẩn đoán ĐTĐ thai kỳ?
   Không ☐ **Có** ☐
6. Có rối loạn Lipit HDL-C < 0,9 mmol/lít hoặc Triglycerid ≥ 2,82 mmol/lít
   Không ☐ **Có** ☐
7. Người có rối loạn đường máu lúc đói hoặc rối loạn dung nạp glucose hoặc HBA,C ≥ 5,7%
   Không ☐ **Có** ☐
8. Có gai đen ở nếp cổ (gợi ý tình trạng kháng insulin)
   Không ☐ **Có** ☐
9. Người trên 45 tuổi nên sàng lọc ĐTĐ (nên xét nghiệm 3 năm 1 lần)
   ≥ 45 tuổi ☐ **< 45 tuổi** ☐
10. Có mắc bệnh tuyến tụy: Viêm tụy, sỏi tụy, ung thư tụy, bệnh cushing?
    Không ☐ **Có** ☐
11. Có dùng thuốc hoặc hóa chất gì không? Corticoid, lợi tiểu, chen beta giao cảm
    Không ☐ **Có** ☐

Như vậy có thể thấy, nguồn dữ liệu y tế trong hệ thống HealthDL mang đặc trưng của dữ liệu lớn. Cụ thể:

Kích thước lớn: Xét trên 1.000 bệnh nhân với các máy đo độc lập, lượng dữ liệu thu thập được tương ứng là 116 Gigabytes về thông số bệnh nhân trong một tháng. Như vậy, khi số bệnh nhân tăng lên và thời gian đo kéo dài, lượng dữ liệu cần thu thập là vô cùng lớn.

Tốc độ lớn: Các thiết bị y sinh tạo ra dữ liệu liên tục với tốc độ cao (tần suất 1 bản ghi dữ liệu/s) đòi hỏi hệ thống lưu trữ cần đảm bảo tính sẵn sàng cao, đáp ứng tương tranh đồng thời giữa các tiến trình ghi dữ liệu và các tiến trình đọc, xử lý dữ liệu theo thời gian thực.

Tính đa dạng: Các bệnh án là dữ liệu bán cấu trúc có lược đồ không đồng nhất. Việc lưu trữ dữ liệu này đòi hỏi hệ quản trị cơ sở dữ liệu phải có lược đồ dữ liệu linh hoạt, điều mà các hệ quản trị cơ sở dữ liệu quan hệ truyền thống không đáp ứng được.

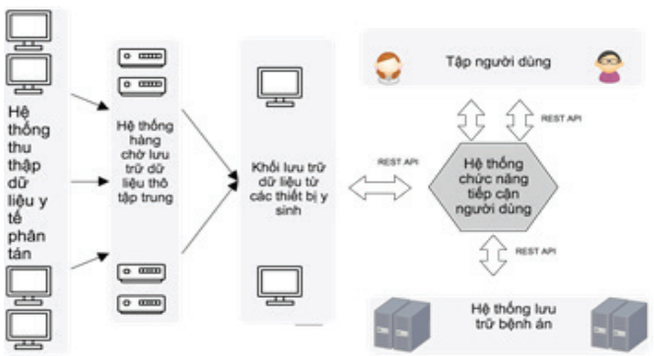
Mang lại giá trị lớn: Dữ liệu y tế được lưu trữ và khai thác mang lại hiệu quả cao trong việc chẩn đoán, điều trị bệnh, góp phần cải thiện chất lượng khám chữa bệnh và giảm chi phí xét nghiệm.

Mô hình hệ thống

Chúng tôi xây dựng HealthDL là hệ thống có kiến trúc tổng thể chia làm 4 phần chính, gồm: 1) Khối các thiết bị y sinh đo các thông số cần thiết từ người bệnh; 2) Khối tiếp nhận và chuyển tiếp dữ liệu; 3) Khối lưu trữ dữ liệu từ các thiết bị y sinh; 4) Khối lưu trữ bệnh án. So với các mô hình truyền thống, HealthDL xây dựng trên các thành phần phân tán, có tính khả mở, tối ưu hoá cho luồng dữ liệu lớn.

Đầu vào của hệ thống có 2 luồng chính: Dữ liệu đầu vào từ các thiết bị y sinh (dữ liệu này sẽ được đưa qua một hàng chờ rồi lưu trữ vào cơ sở dữ liệu); dữ liệu đầu vào bệnh án, lưu trữ vào cơ sở dữ liệu chuyên biệt tối ưu hoá cho dữ liệu bệnh án có cấu trúc linh hoạt.

Mô hình tổng thể của hệ thống được mô tả như hình 1. Trong bài báo này, chúng tôi không trình bày về các thiết bị y sinh mà chỉ đi vào các thành phần chính của 3 khối còn lại với nhiệm vụ thu thập và tổ chức lưu trữ dữ liệu phân tán. Mục tiêu của hệ thống là đảm bảo hiệu năng, khả năng mở rộng cao với dữ liệu vào theo luồng và dữ liệu có cấu trúc linh hoạt, nguồn dữ liệu phân tán rộng về mặt địa lý.



Hình 1. Thiết kế tổng quan HealthDL.

Công nghệ đề xuất

Apache Kafka cho hàng chờ thông điệp

Apache Kafka [7] là hệ thống truyền thông điệp phân tán, độ tin cậy cao, dễ dàng mở rộng và có thông lượng cao. Apache Kafka cung cấp cơ chế offset (có thể hiểu tương tự như chỉ số của một mảng) để lấy thông điệp một cách linh hoạt, cho phép các ứng dụng xử lý có thể xử lý lại dữ liệu nếu việc xử lý trước đó bị lỗi. Ngoài ra, cơ chế "đăng ký" theo dõi cho phép việc lấy thông điệp ra gần như tức thời ngay khi dữ liệu đi vào hàng chờ. Apache Kafka được thiết kế hỗ trợ tốt cho việc thu thập dữ liệu theo thời gian thực. Với các tính chất này, Apache Kafka là lựa chọn phù hợp cho khối thu thập dữ liệu trong HealthDL.

Apache Kafka là hệ thống lưu trữ thông điệp được phát triển tại LinkedIn với những đặc điểm chính sau:

Tốc độ nhanh: Với một máy đơn cài đặt Apache Kafka có thể xử lý số lượng dữ liệu từ việc đọc và ghi lên tới

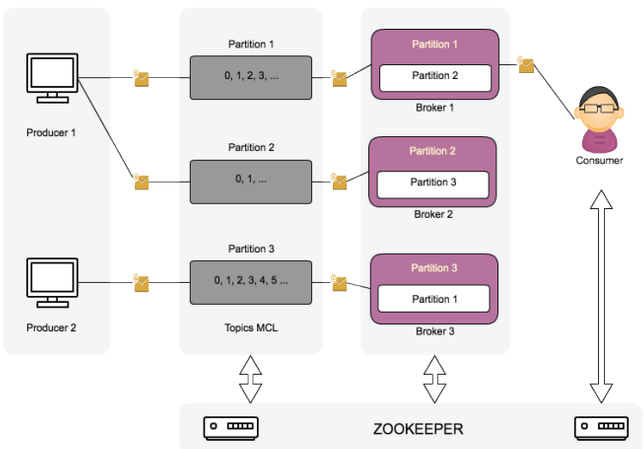
hàng trăm megabytes/s từ hàng ngàn máy khách.

Khả năng mở rộng: Apache Kafka được thiết kế cho phép dễ dàng được mở rộng và trong suốt với người dùng (nghĩa là không có thời gian chết - ngừng hoạt động trong khi thêm một nút mới vào cụm). Khi Apache Kafka chạy trên một cụm, luồng dữ liệu sẽ được phân chia và được vận chuyển tới các nút trong cụm, do đó cho phép trung chuyển các dữ liệu có khối lượng lớn hơn nhiều so với sức chứa của một máy đơn.

Độ tin cậy: Dữ liệu vào hàng chờ sẽ được lưu trữ trên ổ đĩa và được sao chép tới các nút khác trong cụm để ngăn ngừa việc mất dữ liệu, như vậy Apache Kafka đảm bảo tính chịu lỗi cao.

Khi so sánh với các hệ thống truyền thông điệp truyền thống lâu đời như RabbitMQ [8] cho thấy Apache Kafka có lượng dữ liệu đọc và ghi cao hơn nhiều so với RabbitMQ. Ngược lại, lượng tài nguyên mà Apache Kafka chiếm dụng lại ít hơn nhiều. Do đó, Apache Kafka thích hợp hơn cho các ứng dụng xử lý theo thời gian thực với lượng dữ liệu lớn.

Một số khái niệm cần nắm rõ khi vận hành Apache Kafka: 1) Topic: Là "chủ đề", thông thường các dữ liệu liên quan hoặc tương tự được nhóm trong các chủ đề. Mỗi chủ đề có thể được coi là một nguồn dữ liệu riêng biệt; 2) Broker: Apache Kafka chạy trên một cụm bao gồm một hoặc nhiều máy (nút), mỗi nút được gọi là một broker; 3) Partition: Mỗi topic sẽ được phân chia thành nhiều phân mảnh (partition), các partition sẽ là đơn vị cho phép sao lưu để phục hồi đến các broker khác; 4) Producer: Thành phần sinh dữ liệu, ghi dữ liệu tới broker; 5) Consumer: Thành phần "tiêu thụ" dữ liệu, đọc dữ liệu từ các broker.



Hình 2. Mô hình truyền thông điệp phân tán Apache Kafka.

Hình 2 mô tả kiến trúc của Apache Kafka khi chạy trên một cụm với 3 broker, với một topic tên là MCL được chia thành 3 partition, mỗi partition sẽ có một bản sao chép ở một broker khác để phục vụ cơ chế sao lưu, khắc phục lỗi.

Mỗi partition sẽ có 1 broker làm vai trò trưởng nhóm (leader), những broker còn lại có lưu trữ partition đó được gọi là nút theo dõi (follower), chỉ có nhiệm vụ sao lưu dữ liệu. Mỗi partition được cấu hình tham số nhân bản (replication factor). Trong hình 2, replication factor = 2, nghĩa là mỗi partition sẽ được lưu trữ trên 2 broker.

Zookeeper [9] là thành phần cung cấp các dịch vụ lõi cho các hệ thống phân tán như: Dịch vụ quản lý cấu hình hệ thống, bầu trưởng nhóm (leader election), định danh (naming). Zookeeper được Apache Kafka bầu tự động leader cho các partition, quản lý danh sách các nút máy chủ đang hoạt động, quản lý danh sách các topic.

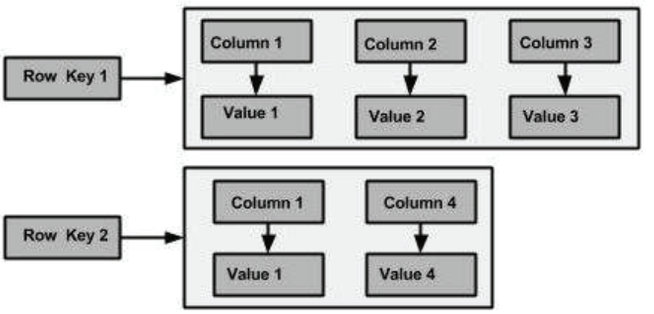
Cơ sở dữ liệu Cassandra cho lưu trữ dữ liệu từ thiết bị y sinh

Cassandra [10] là một cơ sở dữ liệu hướng cột, phân tán (một mô hình dữ liệu NoSQL) có khả năng mở rộng cao (highly scalable), được thiết kế để có thể quản lý một lượng rất lớn dữ liệu có cấu trúc. Cassandra đảm bảo tính sẵn sàng cao với thiết kế có khả năng chịu lỗi, các nút máy chủ trong cụm Cassandra là đồng nhất theo thiết kế ngang hàng (peer-to-peer [11]), không có bất cứ thành phần nào trong hệ thống là điểm hỏng thắt cổ chai (bottle-neck). Cassandra có một số đặc điểm chính sau:

Tính khả mở và có thể co dãn được: Cassandra có tính khả mở cao, nó cho phép thêm vào hệ thống các máy chủ để đáp ứng với nhu cầu tải từ phía ứng dụng, đồng thời cũng cho phép rút bớt ra, tháo khỏi hệ thống các máy chủ để giảm điện năng tiêu thụ, thay thế phục hồi, sửa lỗi mà không phải tạm dừng và khởi động lại hệ thống.

Kiến trúc luôn sẵn sàng: Cassandra không có điểm hỏng hóc đơn lẻ, có cơ chế cho phép hệ thống hoạt động liên tục, đáp ứng các ứng dụng thương mại quan trọng mà không chấp nhận hỏng hóc.

Mô hình dữ liệu linh hoạt: Cassandra là một hệ cơ sở dữ liệu hướng cột với mô hình dữ liệu linh hoạt cho phép lưu trữ các dữ liệu có cấu trúc, bán cấu trúc và phi cấu trúc (hình 3). Cassandra có thể tiếp nhận dữ liệu với cấu trúc động mà không cần định nghĩa trước lược đồ dữ liệu như trong cơ sở dữ liệu quan hệ.



Hình 3. Mô hình dữ liệu hướng cột của Cassandra (Row key 1: Bản ghi dữ liệu với khoá là "key 1"; Column 1: Định nghĩa thuộc tính của cột thứ nhất; Value 1: Giá trị của thuộc tính thứ nhất tương ứng với mỗi bản ghi dữ liệu).

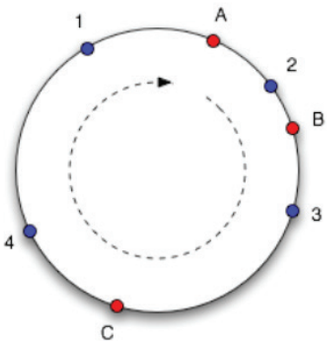
Dễ dàng phân tán dữ liệu: Cassandra thiết kế theo mô hình phân tán dữ liệu của Dynamo Amazon [5] sử dụng hàm băm nhất quán (consistent-hashing [12]) để phân tán dữ liệu. Điều này cho phép tối ưu quá trình di chuyển dữ liệu khi cấu hình hệ thống thay đổi. Các nút máy chủ thêm vào hay rút ra không làm ảnh hưởng đến việc phân bố lại toàn bộ không gian dữ liệu.

Cho phép ghi dữ liệu nhanh, thông lượng lớn: Cassandra mặc dù được thiết kế chạy trên các máy tính phổ thông có cấu hình thấp nhưng vẫn cho phép đáp ứng hiệu năng cao với các thao tác đọc, ghi dữ liệu thông lượng lớn, có thể lưu trữ hàng trăm terabyte dữ liệu.

Thiết kế của Cassandra không có điểm chết tập trung nào. Cassandra có thiết kế dựa trên kiến trúc mạng ngang hàng, tất cả các nút máy chủ trong hệ thống đều có vai trò như nhau và không có nút máy chủ nào đóng vai trò là máy chủ trung tâm mà việc hỏng hóc của máy chủ này có thể kéo theo đánh sập hoàn toàn hệ thống như các kiến trúc chủ - khách truyền thống. Các nút máy chủ của Cassandra là độc lập. Mỗi nút đều có thể xử lý các thao tác ghi và đọc dữ liệu, không phân biệt là dữ liệu được lưu trữ một cách vật lý trên máy chủ nào trong hệ thống. Khi một nút trong hệ thống bị hỏng hóc và dừng hoạt động, các thao tác đọc ghi dữ liệu có thể được xử lý bởi các nút khác trong hệ thống. Quá trình này hoàn toàn trong suốt với ứng dụng, cho phép ẩn đi hỏng hóc của hệ thống đối với các ứng dụng đó.

Trong Cassandra, mỗi đối tượng dữ liệu có thể được nhân bản và lưu giữ trên nhiều máy chủ. Nếu một trong các máy chủ lưu một phiên bản dữ liệu bị lỗi hoặc không phải là phiên bản được cập nhật dữ liệu mới nhất, Cassandra có cơ chế đồng bộ để luôn đảm báo các thao tác đọc sẽ luôn trả về dữ liệu mới nhất. Cơ chế này được thực thi trong quá trình đọc dữ liệu (read repair) thay vì đồng bộ ngay trong thao tác ghi dữ liệu, điều này cho phép tăng hiệu năng đối với thao tác ghi dữ liệu.

Cassandra sử dụng cơ chế hàm băm nhất quán phân tán (distributed consistent hashing [12]) tổ chức các nút máy chủ thành cụm theo định dạng vòng tròn và dữ liệu được phân tán theo vòng tròn này theo hàm băm nhất quán (hình 4).



Hình 4. Bảng băm nhất quán biểu diễn dưới dạng vòng tròn địa chỉ.

Cơ sở dữ liệu MongoDB để lưu trữ dữ liệu bệnh án

MongoDB [13] là cơ sở dữ liệu hướng văn bản, dưới đây là các khái niệm khi làm việc với MongoDB:

Document: Document hay văn bản là đơn vị dữ liệu trong MongoDB. Document có cấu trúc tương tự như kiểu dữ liệu JSON [14], bao gồm các cặp khóa - giá trị, các giá trị có thể bao hàm các trường khoá - giá trị con. Document có thể hiểu giống như các bản ghi dữ liệu trong cơ sở dữ liệu quan hệ, tuy nhiên sự khác biệt là các cặp khoá - giá trị trong các văn bản có thể không giống nhau ở mỗi document. Do vậy, các document là linh hoạt về mặt cấu trúc. Mỗi văn bản có một định danh id là duy nhất trong mỗi collection.

Collection: Collection là nhóm các tài liệu (document), tương đương với một bảng (table) trong cơ sở dữ liệu quan hệ. Các collection không có ràng buộc quan hệ tham chiếu lẫn nhau như các bảng trong hệ quản trị cơ sở dữ liệu quan hệ.

Database: Database là cơ sở dữ liệu, mỗi database gồm nhiều collection. Một máy chủ MongoDB có thể tạo nhiều cơ sở dữ liệu.

Các tính năng quan trọng của MongoDB:

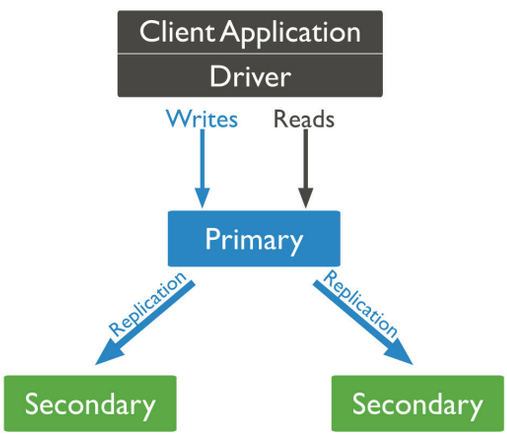
Mô hình dữ liệu linh hoạt: Mô hình dữ liệu hướng văn bản của MongoDB cho phép dễ dàng lưu trữ và kết hợp dữ liệu có cấu trúc động mà không làm giảm đi việc kiểm soát ràng buộc về kiểu, miền giá trị của dữ liệu.

Tính khả mở cao, cho phép triển khai trên nhiều trung tâm dữ liệu: MongoDB có thể mở rộng trên một trung tâm dữ liệu hoặc triển khai phân tán trên nhiều trung tâm dữ liệu phân tán về mặt địa lý.

Tính sẵn sàng cao: Người quản trị hệ thống có thể dễ dàng tăng hoặc giảm số lượng máy chủ tham gia vào hệ cơ sở dữ liệu ở bất cứ thời điểm nào, không cần phải tạm dừng và khởi động lại hệ thống.

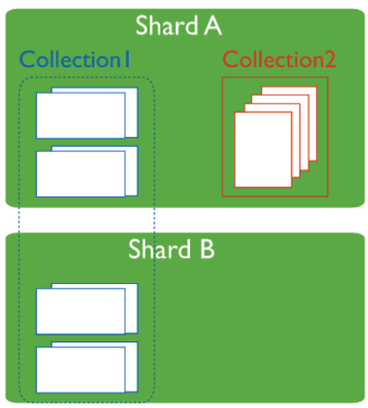
Nhiều tính năng tích hợp đi kèm hệ sinh thái sử dụng MongoDB: Các tính năng phân tích dữ liệu, trình diễn dữ liệu, đánh chỉ mục, xử lý dữ liệu đồ thị, dữ liệu không gian đều có thể dễ dàng tích hợp với cơ sở dữ liệu MongoDB thông qua các trình điều khiển kết nối chuẩn hoá và được hỗ trợ mạnh mẽ.

Nhân bản: Nhân bản là một tính năng vô cùng quan trọng của MongoDB và được khuyến cáo sử dụng khi cài đặt MongoDB cho môi trường thực tế (ứng dụng thực, dữ liệu thực). Kỹ thuật nhân bản tức là sao một đối tượng dữ liệu ra một nhóm các máy chủ khác nhau. Trong nhóm các máy chủ này sẽ có một máy chủ là máy chủ nhân bản chính và các máy còn lại là máy chủ nhân bản thứ cấp (phụ). Máy chủ nhân bản chính đóng vai trò là tổng quản, nơi mà mọi thao tác ghi, cập nhật đối tượng dữ liệu cần phải thông qua máy chủ đó. Các máy chủ thứ cấp có thể được sử dụng cho việc đọc dữ liệu để cân bằng tải. MongoDB có chế độ chuyển đổi dự phòng (failover) tự động nếu máy chủ nhân bản chính bị lỗi thì một trong các máy chủ thứ cấp sẽ được bầu làm máy chủ chính thay thế để đảm bảo các thao tác ghi dữ liệu luôn được thực thi thành công (hình 5).



Hình 5. Nhân bản trong MongoDB (Client Application: Ứng dụng phía người dùng; Writes: Tác vụ ghi dữ liệu; Reads: Tác vụ đọc dữ liệu; Primary: Máy chủ nhân bản chính; Secondary: Máy chủ nhân bản thứ cấp; Replication: Quá trình nhân bản dữ liệu).

Sharding: Khi kích thước của dữ liệu gia tăng, một máy tính đơn lẻ không thể đủ để lưu dữ liệu cũng như cung cấp các xử lý đọc và ghi thông thường. Sharding là một kỹ thuật cho phép các văn bản trong MongoDB được phân tán ra trên nhiều máy chủ. Việc phân tán dữ liệu này có ý nghĩa quan trọng góp phần mở rộng khả năng lưu trữ và xử lý của hệ thống, đồng thời tăng khả năng chịu lỗi. Song song với việc tự động hoá phân tán dữ liệu, nếu một máy chủ dữ liệu nào đó bị gặp lỗi thì hệ thống có cơ chế tự động định tuyến lại các thao tác đọc ghi trên máy chủ bị lỗi qua các máy chủ khác. Hình 6 giải thích việc collection A được chia ra ở 2 shard A và B trong khi colllection B chỉ nằm ở 1 shard A.



Hình 6. Kỹ thuật phân mảnh dữ liệu sharding trong MongoDB (Shard: Một khối lưu trữ dữ liệu độc lập; Collection: Tập hợp những văn bản có tính tương đồng về cấu trúc).

Với thiết kế dạng dữ liệu hướng văn bản, MongoDB là cơ sở dữ liệu thích hợp nhất để lưu trữ thông tin bệnh án gồm nhiều trường thông tin, các trường không đồng nhất giữa các bệnh án khác nhau hoặc các bệnh án của các bệnh khác nhau. Cấu trúc hướng văn bản của MongoDB cho phép người dùng có thể tạo các chỉ mục trên các thuộc tính của văn bản để tìm kiếm nhanh, điều này là khác biệt của MongoDB so với Cassandra. Đây là lý do để chúng tôi lựa chọn MongoDB thay cho Cassandra trong lưu trữ thông tin bệnh án.

Đánh giá thực nghiệm

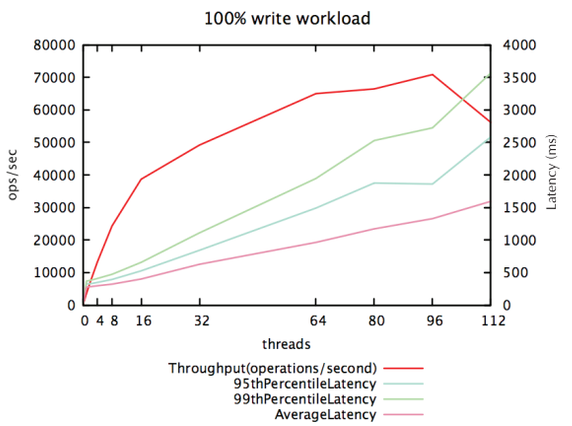
Trong phần này, chúng tôi đánh giá hiệu năng của hệ thống HealthDL trong các thao tác đọc, ghi dữ liệu trên môi trường phân tán, khi gia tăng các kết nối tương tranh đồng thời. Chúng tôi đã cài đặt và tiến hành chạy đánh giá MongoDB và Cassandra với 2 công cụ đánh giá tiêu chuẩn là YCSB [15] và Cassandra-stress [16]. Hai công cụ này

đo số thao tác vào ra trung bình có thể thực thi trong 1 s với khối dữ liệu vào ra được cung cấp sẵn.

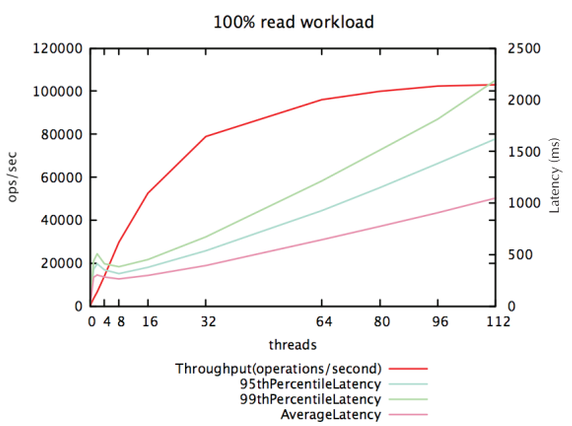
Đánh giá thành phần lưu trữ thông tin bệnh án MongoDB

MongoDB được cài đặt trên môi trường ảo hoá sử dụng docker-compose, cụm máy tính gồm 30 nút ảo chia sẻ cấu hình: 2 CPU (Haswell 2.3 G), 1 SSD (Intel 800 GB SATA 6 Gb/s), RAM (128 GB).

Kết quả đối với kịch bản chỉ đọc và chỉ ghi cho hiệu năng cao với tốc độ đạt từ 70.000 đến 100.000 bản ghi/s, độ trễ là 1 đến 1,5 s với số lượng client kết nối tới lên đến 100 client tương tranh đồng thời (hình 7A và 7B).



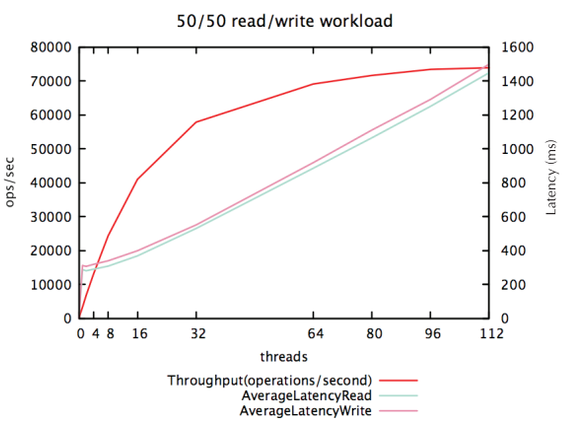
Hình 7A. Kịch bản ghi dữ liệu MongoDB.



Hình 7B. Kịch bản đọc dữ liệu MongoDB.
Trong đó: 100% write workload: Kịch bản chỉ ghi; 100% read workload: Kịch bản chỉ đọc; Ops/sec: Số tác vụ thực thi thành công trong 1 s; Theads: Số lượng kết nối đến hệ thống; Throughput: Giải nghĩa như ops/sec; 95thPercentileLatency: Độ trễ của 5% tác vụ có độ trễ cao nhất; 99thPercentileLatency: Độ trễ của 1% tác vụ có độ trễ cao nhất; AverageLatency: Độ trễ trung bình.

Trong hình 7A và 7B, đường kẻ màu tím mô tả độ trễ trung bình, đường kẻ màu xanh dương mô tả độ trễ trung bình của 5% các tác vụ có độ trễ cao nhất, đường kẻ màu xanh lá mô tả độ trễ trung bình của 1% các tác vụ có độ trễ cao nhất. Từ biểu đồ kết quả cho thấy, chỉ 1% các tác vụ có độ trễ cao nhất là 3,5 s.

Với kịch bản các tác vụ đọc và ghi là tỷ lệ 50/50 (vừa đọc vừa ghi) cũng cho kết quả tốt, với tốc độ trung bình 70.000 bản ghi/s, độ trễ trung bình là 1,4 s (hình 8). Trong hình 8, đường kẻ màu xanh dương mô tả độ trễ trung bình của các tác vụ đọc, đường kẻ màu tím mô tả độ trễ trung bình của các tác vụ ghi. Từ kết quả này cho thấy rằng, độ trễ của hai dạng tác vụ này không quá chênh lệch với nhau.



Hình 8. Kịch bản đọc ghi đồng thời MongoDB (50/50 read/write workload: Kịch bản đọc ghi đồng thời; Ops/sec: Số tác vụ thực thi thành công trong 1 s; Theads: Số lượng kết nối đến hệ thống; Throughput: Giải nghĩa như ops/sec; AverageLatencyRead: Độ trễ trung bình của tác vụ đọc dữ liệu; AverageLatencyWrite: Độ trễ trung bình của tác vụ ghi dữ liệu).

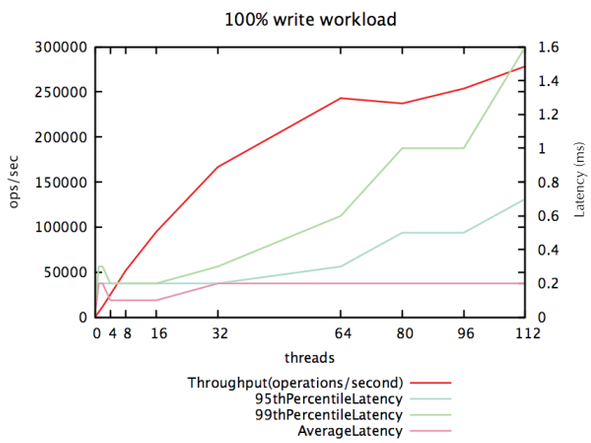
Mặc dù khi xây dựng một cụm MongoDB phân tán ảo bằng docker-compose, các nút sẽ không sử dụng được trọn vẹn tài nguyên phần cứng tốt nhất, nhưng kết quả trả về rất khả quan đối với yêu cầu của việc nhập liệu, lưu trữ và chia sẻ bệnh án.

Đánh giá thành phần lưu trữ dữ liệu từ thiết bị y sinh Cassandra

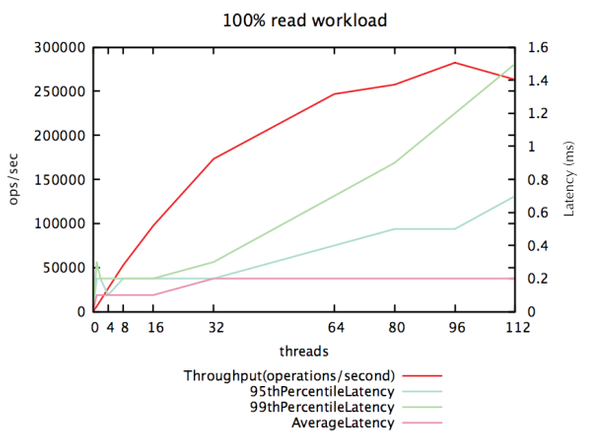
Cassandra được cài đặt trên 3 máy chủ riêng biệt với cấu hình mỗi máy như sau: 2 CPU (Haswell 2.3 G), 1 SSD (Intel 800 GB SATA 6 Gb/s), RAM (128 GB). Kịch bản thử nghiệm đo số lượng các thao tác đọc, ghi/s và độ trễ trung bình. Các thử nghiệm qua gia tăng số lượng các tiến trình client thực thi các thao tác đọc, ghi dữ liệu tương tranh đồng thời.

Trong thử nghiệm các tiến trình tương tranh chỉ thực thi các thao tác ghi (hình 9A), các tiến trình tương tranh chỉ thực thi các thao tác đọc (hình 9B), Cassandra đáp ứng hiệu năng cao với số thao tác xử lý từ 250.000 đến 300.000 bản ghi/s. Độ trễ trung bình từ 0,2 đến 0,3 ms. Với kịch bản đọc, ghi tương tranh (hình 10) Cassandra vẫn đáp ứng được số thao tác từ 250.000 đến 300.000 bản ghi/s.

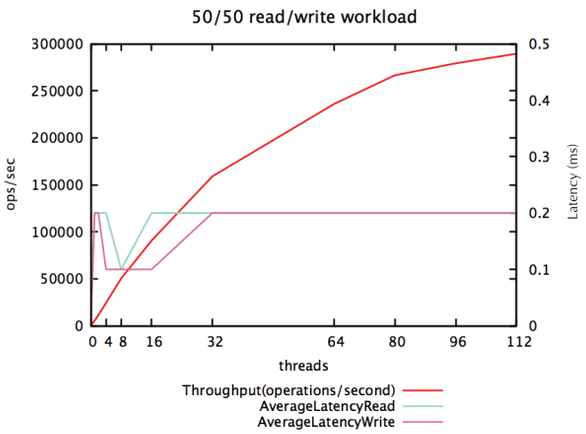
Đường kẻ màu tím mô tả độ trễ trung bình, đường kẻ màu xanh dương mô tả độ trễ trung bình của 5% các tác vụ có độ trễ cao nhất, đường kẻ màu xanh lá mô tả độ trễ trung bình của 1% các tác vụ có độ trễ cao nhất. Từ kết quả nhận thấy rằng, chỉ có 1% các tác vụ có độ trễ cao nhất là 1,6 ms.



Hình 9A. Gia tăng thao tác tương tranh chỉ ghi.



Hình 9B. Gia tăng thao tác tương tranh chỉ đọc.
Trong đó: 100% write workload: Kịch bản chỉ ghi; 100% read workload: Kịch bản chỉ đọc; Ops/sec: Số tác vụ thực thi thành công trong 1 s; Theads: Số lượng kết nối đến hệ thống; Throughput: Giải nghĩa như ops/sec; 95thPercentileLatency: Độ trễ của 5% tác vụ có độ trễ cao nhất; 99thPercentileLatency: Độ trễ của 1% tác vụ có độ trễ cao nhất; AverageLatency: Độ trễ trung bình.



Hình 10. Đọc ghi tương tranh đồng thời trên Cassandra (50/50 read/write workload: Kịch bản đọc ghi đồng thời; Ops/sec: Số tác vụ thực thi thành công trong 1 s; Theads: Số lượng kết nối đến hệ thống; Throughput: Giải nghĩa như ops/sec; AverageLatencyRead: Độ trễ trung bình của tác vụ đọc dữ liệu; AverageLatencyWrite: Độ trễ trung bình của tác vụ ghi dữ liệu).

Các kết quả thử nghiệm với MongoDB và Cassandra trên môi trường tương tranh cho thấy, các thành phần đáp ứng hiệu năng cao ngay cả trong điều kiện đọc ghi tương tranh đồng thời. Về hiệu năng, Cassandra cho hiệu năng cao hơn về số thao tác/s, phù hợp với lưu trữ các dữ liệu đến từ thiết bị y sinh thời gian thực.

Kết luận

Trong bài báo này, chúng tôi giới thiệu HealthDL, hệ thống thu thập và lưu trữ dữ liệu lớn cho y tế. Kết quả đánh giá hiệu năng của thành phần lưu trữ trên môi trường thử nghiệm chứng minh khả năng đáp ứng tốt với các yêu cầu nghiệp vụ đọc ghi dữ liệu tương tranh đồng thời. Về thiết kế tổng thể, hệ thống cấu thành từ các thành phần phân tán, có tính khả mở cao, hỗ trợ mô hình dữ liệu linh hoạt. Trong tương lai, chúng tôi sẽ triển khai hệ thống trong thực tiễn và tiến hành tích hợp với thành phần phân tích dữ liệu y tế phân tán.

LỜI CẢM ƠN

Các tác giả xin cảm ơn Chương trình KH&CN cấp nhà nước phục vụ phát triển bền vững vùng Tây Bắc đã tài trợ nghiên cứu thông qua đề tài "Ứng dụng và triển khai hệ thống phần mềm tích hợp và kết nối các thiết bị điện tử y sinh và mạng truyền thông hỗ trợ theo dõi sức khoẻ và dịch tễ cộng đồng khu vực Tây Bắc" (mã số KHCN-TB.06C/13-18).

TÀI LIỆU THAM KHẢO

[1] M. Ercan, M. Lane (2014), "An evaluation of NoSQL databases for EHR systems", *Proceedings of the 25th Australasian Conference on Information Systems*, pp.8-10.

[2] J. Andreu-Perez, C.C.Y. Poon, R.D. Merrifield, S.T.C. Wong, G.Z. Yang (2015), "Big data for health", *IEEE J. Biomed. Heal. informatics*, **19(4)**, pp.1193-1208.

[3] C. Dobre, F. Xhafa (2015), "NoSQL technologies for real time (patient) monitoring", *Advanced Technological Solutions for E-Health and Dementia Patient Monitoring*, pp.183-210.

[4] K. Grolinger, W.A. Higashino, A. Tiwari, M.A. Capretz (2013), "Data management in cloud environments: NoSQL and NewSQL data stores", *J. of Cloud Comput.: Adv. Syst. Appl.*, **2**, pp.2-22.

[5] G. Decandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, W. Vogels (2007), "Dynamo: Amazon's highly available key-value store", *ACM SIGOPS Oper. Syst. Rev.*, **41(6)**, pp.205-220.

[6] D.S. Parker, G.J. Popek, G. Rudisin, A. Stoughton, B.J. Walker, E. Walton, J.M. Chow, D. Edwards, S. Kiser, C. Kline (1983), "Detection of Mutual Inconsistency in Distributed Systems", *IEEE Trans. Softw. Eng.*, **SE-9(3)**, pp.240-247.

[7] N. Garg (2013), *Apache Kafka*, Packt Publishing Ltd.

[8] A. Videla, J.J.W. Williams (2012), *RabbitMQ in action*, Manning.

[9] P. Hunt, M. Konar, F.P. Junqueira, B. Reed (2010), "ZooKeeper: Wait-free coordination for Internet-scale systems", *USENIX Annual Technical Conference*, **8**, p.9.

[10] A. Lakshman, P. Malik (2010), "Cassandra: A decentralized structured storage system", *ACM SIGOPS Oper. Syst. Rev.*, **44(2)**, pp.35-40.

[11] S. Androutsellis-Theotokis, D. Spinellis (2004), "A survey of peer-to-peer content distribution technologies", *ACM Comput. Surv.*, **36(4)**, pp.335-371.

[12] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, R. Panigrahy (1997), "Consistent Hashing and Random Trees: Distributed caching protocols for Relieving hot Spots on the world wide web", *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pp.654-663.

[13] K. Chodorow (2013), *MongoDB: The definitive guide*, O'Reilly Media, Inc.

[14] T. Bray (2014), "The javascript object notation (JSON) data interchange format", *Standards Track*, pp.1-16.

[15] B.F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, R. Sears (2010), "Benchmarking Cloud Serving Systems with YCSB", *Proceeding of the 1st ACM symposium on cloud computing*.

[16] Cassandra Stress, Available, http://docs.datastax.com/en/cassandra/2.1/cassandra/tools/toolsCStress_t.html.