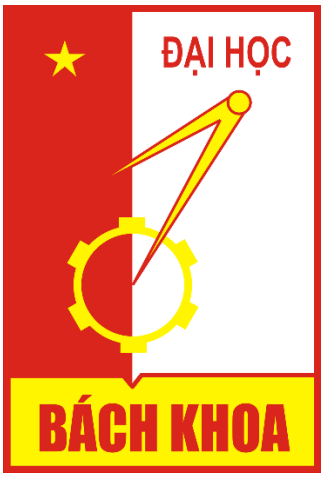
ĐẠI HỌC BÁCH KHOA HÀ NỘI

KHOA TOÁN – TIN



NHÓM 12 – BÁO CÁO TIẾN ĐỘ BÀI TẬP LỚN

HỌC PHẦN: HỆ THỐNG VÀ MẠNG MÁY TÍNH (MI4060)

CHỦ ĐỀ: Distributed Computing

ĐỀ TÀI: Thiết kế hệ thống hiệu suất cao dựa trên
trình thu thập dữ liệu thông tin web phân tán – *Distributed Web Crawler*

.

Giảng viên hướng dẫn: TS. Ngô Thị Hiền

Hà Nội, Tháng 10/2024

MỤC LỤC

PHẦN 1. GIỚI THIỆU CHUNG

1.1. Bối cảnh và lý do chọn đề tài

> *"Khi chúng ta lướt web mỗi ngày, chúng ta đang đắm chìm trong một đại dương thông tin khổng lồ."*

Theo một nghiên cứu gần đây, lượng dữ liệu được tạo ra mỗi ngày ước tính lên đến 2,5 triệu gigabyte (GB). Đó là một con số choáng ngợp, và trong số đó, nhiều thông tin quý giá vẫn chưa được khai thác triệt để.

Chúng ta đang sống trong kỷ nguyên của dữ liệu lớn (Big Data), nơi mà khả năng thu thập, phân tích và sử dụng thông tin là yếu tố then chốt cho thành công của bất kỳ doanh nghiệp hay tổ chức nào. Thông tin trên internet không chỉ đa dạng mà còn thường xuyên thay đổi, đòi hỏi các phương pháp thu thập và phân tích phải linh hoạt và hiệu quả.

1.2. Mục tiêu đề tài

Đề tài *"Thiết kế hệ thống dựa trên trình thu thập dữ liệu thông tin web phân tán"* ra đời nhằm giải quyết bài toán khai thác thông tin một cách hiệu quả và bền vững. Hệ thống này được thiết kế để không chỉ thu thập dữ liệu từ nhiều nguồn khác nhau mà còn tối ưu hóa quá trình này để đảm bảo tốc độ và độ tin cậy cao.

Với mục tiêu này, nhóm 12 chúng em sẽ triển khai một giải pháp sử dụng công nghệ thu thập dữ liệu phân tán, cho phép các nút thu thập và xử lý dữ liệu đồng thời, từ đó giảm tải cho từng nút và tăng cường hiệu suất tổng thể của hệ thống.

PHẦN 2. TỔNG QUAN VỀ TRÌNH THU THẬP DỮ LIỆU WEB

2.1. Khái niệm và nguyên lý hoạt động

Trình thu thập dữ liệu web, hay còn gọi là web crawler hoặc spider, là một chương trình tự động được thiết kế để duyệt qua các trang web trên internet và thu thập thông tin từ chúng. Nguyên lý hoạt động của một trình thu thập dữ liệu cơ bản bao gồm các bước sau:

Bước 1. **Khởi tạo danh sách URL:** Trình thu thập bắt đầu với một danh sách các URL (đường dẫn) cần truy cập. Những URL này có thể được cung cấp thủ công hoặc tự động lấy từ một nguồn dữ liệu nào đó.

Bước 2. **Truy cập và phân tích nội dung:** Khi truy cập vào một URL, trình thu thập sẽ tải nội dung của trang và phân tích cấu trúc của nó. Điều này có thể bao gồm việc xác định các thẻ HTML, hình ảnh, liên kết, và các phần tử khác có giá trị.

Bước 3. **Lưu trữ dữ liệu:** Sau khi phân tích, thông tin hữu ích sẽ được lưu trữ vào cơ sở dữ liệu để phục vụ cho các mục đích sau này, chẳng hạn như tìm kiếm, phân tích, hoặc thống kê.

Bước 4. **Theo dõi liên kết:** Trình thu thập sẽ tiếp tục theo dõi các liên kết trong trang để tìm các trang mới, từ đó mở rộng danh sách URL cần thu thập.

2.2. Phân loại Trình thu thập dữ liệu

Có 2 loại trình thu thập dữ liệu chính:

- **Trình thu thập tập trung (Centralized Crawler):** Loại trình thu thập này hoạt động từ một điểm duy nhất, tải xuống và xử lý tất cả các trang web từ một máy chủ. Mặc dù dễ dàng triển khai, nó có thể gặp phải các vấn đề về hiệu suất và khả năng mở rộng khi đối mặt với lượng dữ liệu lớn.
- **Trình thu thập phân tán (Distributed Crawler):** Như tên gọi, loại này sử dụng nhiều nút để thực hiện thu thập dữ liệu đồng thời. Mỗi nút có thể đảm nhiệm một phần công việc, giúp tăng tốc độ thu thập và giảm tải cho các nguồn dữ liệu. Phương pháp này rất hiệu quả trong việc xử lý dữ liệu lớn và thay đổi liên tục.

PHẦN 3. TRÌNH THU THẬP DỮ LIỆU WEB PHÂN TÁN (DISTRIBUTED WEB CRAWLER)

3.1. Khái niệm

Distributed Web Crawler là một hệ thống thu thập dữ liệu từ các trang web, nhưng thay vì hoạt động tập trung, nó sử dụng nhiều thành phần phân tán trên nhiều máy chủ hoặc hệ thống để thu thập dữ liệu đồng thời. Mục tiêu chính của Distributed Web Crawler là cải thiện hiệu quả thu thập thông tin, mở rộng quy mô để xử lý số lượng lớn dữ liệu web.

Distributed Web Crawler được sử dụng trong nhiều ứng dụng khác nhau, bao gồm:

- Công cụ tìm kiếm (Search Engines) như Google, Bing, để thu thập và lập chỉ mục nội dung trang web.
- Khai thác dữ liệu (Data Mining) từ các trang web cho các phân tích thông tin và nghiên cứu thị trường.
- Giám sát web (Web Monitoring) để theo dõi các thay đổi và xu hướng trên các trang web quan trọng.

Trong hệ thống crawl quy mô lớn, như thu thập dữ liệu hàng tỷ trang web, việc sử dụng Distributed Web Crawler là cần thiết vì:

- **Tăng khả năng mở rộng:** Thu thập dữ liệu từ một lượng lớn trang web đòi hỏi nhiều tài nguyên và băng thông. Phân tán công việc giúp chia tải cho nhiều máy, từ đó tăng khả năng thu thập đồng thời.
- **Hiệu suất cao:** Phân tán giúp tối ưu hóa tốc độ thu thập dữ liệu, giảm thời gian tổng thể để crawl một lượng lớn trang web.
- **Độ tin cậy:** Nếu một máy hoặc phần mềm crawler bị lỗi, các crawler khác vẫn có thể tiếp tục hoạt động, đảm bảo tính liên tục của hệ thống.
- **Giảm quá tải:** Phân tán các crawler trên nhiều địa điểm giúp giảm thiểu quá tải cho một khu vực hoặc máy chủ duy nhất.

3.2. Các hạn chế của hệ thống crawler đơn lẻ và lý do cần phân tán.

3.2.1. Hạn chế của hệ thống Crawler đơn lẻ

Hệ thống crawler đơn lẻ, mặc dù có thể thu thập dữ liệu web hiệu quả trên quy mô nhỏ, nhưng khi đối mặt với việc thu thập dữ liệu từ hàng tỷ trang web hoặc các trang có tốc độ thay đổi cao, nó gặp phải nhiều hạn chế, bao gồm:

a) Khả năng mở rộng hạn chế

Crawler đơn lẻ chỉ hoạt động trên một máy chủ hoặc một hệ thống duy nhất, do đó:

- Giới hạn về tài nguyên: CPU, RAM, và băng thông đều có giới hạn trên một máy chủ duy nhất. Khi số lượng trang web cần thu thập lớn, hệ thống sẽ nhanh chóng gặp phải giới hạn về tài nguyên.
- Tốc độ thu thập chậm: Một crawler đơn lẻ không thể xử lý cùng lúc nhiều yêu cầu đến từ các trang web khác nhau, làm giảm tốc độ thu thập dữ liệu. Điều này có thể khiến quá trình crawl trở nên rất chậm khi quy mô dữ liệu quá lớn.

b) Nguy cơ gặp "nút cổ chai" (Bottleneck)

Vì tất cả các tác vụ đều diễn ra trên cùng một hệ thống, nếu có một thành phần nào đó hoạt động kém hiệu quả hoặc bị quá tải, toàn bộ hệ thống sẽ bị ảnh hưởng. Hệ thống dễ bị tắc nghẽn khi gặp phải các trang web có dung lượng lớn, nhiều nội dung đa phương tiện, hoặc có phản hồi chậm.

c) Tính không ổn định và dễ bị lỗi

Một crawler đơn lẻ dễ bị gián đoạn nếu máy chủ gặp sự cố, mất kết nối, hoặc bị quá tải. Nếu hệ thống bị ngưng hoạt động, toàn bộ quá trình crawl sẽ bị dừng lại, dẫn đến việc không thu thập được dữ liệu kịp thời.

d) Không hiệu quả trong việc quản lý băng thông

Crawl các trang web từ một vị trí địa lý duy nhất có thể dẫn đến việc sử dụng băng thông không đồng đều, đặc biệt khi cần crawl các trang web ở các vùng địa lý khác nhau. Băng thông giữa vị trí crawler và các trang web quốc tế có thể bị hạn chế, khiến quá trình thu thập dữ liệu bị chậm đi đáng kể.

e) Không tối ưu cho các trang web có nội dung thay đổi liên tục

Nếu một trang web cập nhật thường xuyên hoặc có nội dung động (như các mạng xã hội hoặc trang tin tức), crawler đơn lẻ sẽ khó theo kịp tần suất thay đổi, khiến dữ liệu thu thập được trở nên lỗi thời.

f) Thiếu khả năng chịu lỗi (Fault Tolerance)

Nếu crawler gặp sự cố, toàn bộ quá trình thu thập dữ liệu có thể bị mất đi, hoặc phải bắt đầu lại từ đầu. Điều này dẫn đến tình trạng lãng phí tài nguyên và thời gian.

3.2.2. Lý do cần phân tán

a) Khả năng mở rộng (Scalability)

Distributed Web Crawler giải quyết hạn chế về tài nguyên của crawler đơn lẻ bằng cách phân chia công việc trên nhiều máy chủ (nodes). Mỗi node có thể đảm nhận một phần nhỏ của nhiệm vụ crawl, cho phép hệ thống thu thập dữ liệu từ hàng tỷ trang web mà không bị giới hạn bởi tài nguyên của một máy duy nhất.

b) Tăng tốc độ thu thập dữ liệu

Việc phân tán hệ thống crawl giúp tăng tốc độ thu thập dữ liệu bằng cách cho phép nhiều crawler hoạt động song song. Khi nhiều máy chủ cùng thu thập dữ liệu từ nhiều nguồn khác nhau, tổng thời gian để crawl toàn bộ web sẽ giảm đi đáng kể.

c) Tính chịu lỗi cao (Fault Tolerance)

Trong hệ thống phân tán, nếu một node (máy chủ) gặp sự cố, các node khác vẫn có thể tiếp tục hoạt động. Công việc của node bị lỗi có thể được phân chia lại cho các node còn lại mà không ảnh hưởng đến toàn bộ quá trình. Điều này giúp hệ thống có độ tin cậy cao hơn và ít bị gián đoạn.

d) Quản lý tài nguyên hiệu quả

Phân tán công việc trên nhiều máy chủ giúp sử dụng tài nguyên hiệu quả hơn. Các crawler có thể được phân bố ở nhiều vị trí địa lý khác nhau, từ đó tối ưu hóa băng thông và tránh quá tải trên một máy chủ duy nhất. Điều này cũng giúp tối ưu hóa việc crawl các trang web quốc tế.

e) Tối ưu hóa quá trình thu thập nội dung động

Distributed Web Crawler có thể được cấu hình để crawl các trang web động hoặc có nội dung thay đổi thường xuyên theo cách hiệu quả hơn. Các node có thể được phân công chỉ định thu thập dữ liệu từ các nguồn cụ thể, đảm bảo rằng dữ liệu luôn được cập nhật.

f) Tăng khả năng quản lý và sắp xếp

Trong hệ thống phân tán, có thể áp dụng nhiều chiến lược crawl khác nhau dựa trên loại nội dung hoặc cấu trúc của trang web. Các crawler có thể được điều chỉnh để ưu tiên crawl những trang quan trọng trước, hoặc phân chia công việc theo vùng địa lý hoặc ngôn ngữ, giúp quản lý hiệu quả hơn quá trình thu thập dữ liệu.

PHẦN 4. CHIẾN LƯỢC LƯU TRỮ DỮ LIỆU

4.1. Chiến lược lưu trữ dữ liệu crawl

4.1.1. MongoDB

Được sử dụng cho dữ liệu có cấu trúc hoặc nửa cấu trúc như JSON hoặc BSON. MongoDB lưu trữ dữ liệu dưới dạng tài liệu (document-based) giúp dễ dàng tìm kiếm, truy xuất, và lưu trữ dữ liệu crawl. Thích hợp cho việc lưu trữ dữ liệu không có cấu trúc cố định.

MongoDB giúp các tổ chức lưu trữ lượng lớn dữ liệu trong khi vẫn hoạt động nhanh chóng. Ngoài lưu trữ dữ liệu, MongoDB còn được sử dụng trong các trường hợp sau:

- Tích hợp một lượng lớn dữ liệu đa dạng
- Mô tả các cấu trúc dữ liệu phức tạp, biến hoá
- Cung cấp dữ liệu cho các ứng dụng hiệu suất cao
- Hỗ trợ các ứng dụng đám mây lai và đa đám mây
- Hỗ trợ phương pháp phát triển Agile

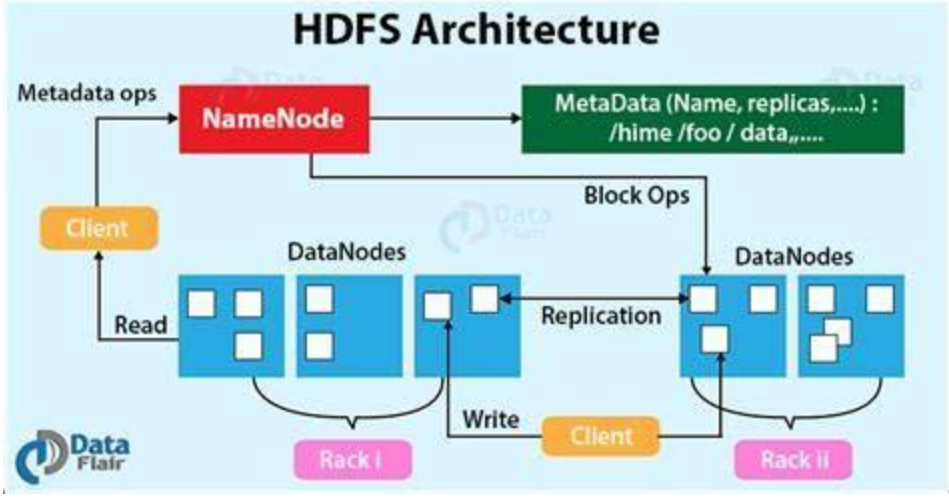
Thay vì sử dụng các table và row như trong cơ sở dữ liệu quan hệ, vì là cơ sở dữ liệu NoSQL, MongoDB được tạo thành từ collection và document. Document được tạo thành từ các cặp khóa-giá trị (là đơn vị dữ liệu cơ bản của MongoDB). Còn collection, tương đương với table trong SQL, là nơi chứa các bộ document.

Ưu điểm: Linh hoạt trong lưu trữ dữ liệu không có cấu trúc, có khả năng mở rộng ngang (horizontal scaling), hỗ trợ indexing mạnh mẽ.

Nhược điểm: Không tối ưu cho dữ liệu lớn hơn vài terabyte và truy vấn phức tạp.

4.1.2. Hadoop Distributed File System (HDFS)

Một hệ thống file phân tán, thường được dùng trong các dự án Big Data. Dữ liệu crawl có thể được lưu trữ dưới dạng file lớn (logs, media) và phân tán trên nhiều nodes.



Ưu điểm: Khả năng lưu trữ và xử lý dữ liệu lớn (petabyte). Tích hợp tốt với các công cụ xử lý dữ liệu khác như Apache Spark.

Nhược điểm: Truy cập và cập nhật dữ liệu thời gian thực không tốt bằng MongoDB hay Elasticsearch.

4.1.3. Elasticsearch

Elasticsearch là một dạng Search Engine (Công cụ tìm kiếm) phát triển dựa trên nền tảng Apache Lucene. Elasticsearch được ra mắt năm 2010 bởi Shay Banon, được xây dựng bằng ngôn ngữ Java, với giao diện web HTTP có hỗ trợ JSON. Nó cho phép ta chuyển dữ liệu vào và tìm kiếm trực tiếp, thay vì tìm kiếm trong một hệ quản lý cơ sở dữ liệu thông thường như Oracle, MySQL,…

Thích hợp cho việc lưu trữ và tìm kiếm dữ liệu văn bản hoặc logs crawl với khả năng tìm kiếm full-text mạnh mẽ.

Ưu điểm: Tìm kiếm nhanh chóng, khả năng lập chỉ mục (indexing) mạnh mẽ và truy vấn gần thời gian thực.

Nhược điểm: Khó tối ưu hóa với dữ liệu không phải là văn bản và logs, không phù hợp với dữ liệu quá lớn.

4.2. Kỹ thuật tối ưu hóa dữ liệu crawl

4.2.1. Nén dữ liệu (Data Compression)

Các thuật toán nén như GZIP, BZIP2 giúp giảm kích thước dữ liệu lưu trữ. Phù hợp cho dữ liệu logs, text crawl. Điều này giúp tiết kiệm không gian lưu trữ và cải thiện hiệu suất truyền tải mạng.

Ưu điểm: Giảm dung lượng dữ liệu, tiết kiệm không gian và băng thông.

Nhược điểm: Tốn tài nguyên tính toán để nén và giải nén dữ liệu.

4.2.2. Loại bỏ trùng lặp (Duplicate Removal)

Việc dữ liệu crawl có thể chứa nhiều bản sao gây lãng phí không gian lưu trữ. Sử dụng các thuật toán kiểm tra và loại bỏ trùng lặp, như hash-based deduplication (so sánh hash của dữ liệu).

Ưu điểm: Giảm tải lưu trữ không cần thiết, cải thiện hiệu suất xử lý.

Nhược điểm: Cần thêm bước tính toán và kiểm tra trùng lặp, có thể phức tạp đối với dữ liệu phi cấu trúc.

4.3. Đánh giá hiệu suất lưu trữ và xử lý dữ liệu

4.3.1. Hiệu suất lưu trữ

Đánh giá dựa trên các chỉ số như tốc độ ghi dữ liệu, dung lượng lưu trữ, khả năng mở rộng (scalability) và tính sẵn sàng (availability). Mỗi công nghệ có đặc điểm hiệu suất riêng:

- MongoDB: Lưu trữ tốt với dữ liệu nửa cấu trúc nhưng hiệu suất có thể giảm nếu không tối ưu index.
- HDFS: Lưu trữ dữ liệu lớn nhưng tốc độ truy cập và xử lý chậm với dữ liệu thời gian thực.
- Elasticsearch: Tìm kiếm rất nhanh với dữ liệu logs, văn bản nhưng không hiệu quả cho lưu trữ dữ liệu dạng khác.

4.3.2. Hiệu suất xử lý

Phụ thuộc vào khả năng tìm kiếm, truy vấn và phân tích dữ liệu.

Ví dụ:

- MongoDB: Thích hợp cho truy vấn phức tạp nhưng cần tối ưu các indexes để đảm bảo hiệu suất.
- HDFS: Tối ưu cho xử lý song song với dữ liệu lớn nhưng chậm với các tác vụ nhỏ lẻ.
- Elasticsearch: Tối ưu cho các tìm kiếm văn bản hoặc logs gần thời gian thực, nhưng việc xử lý dữ liệu lớn không phải logs sẽ không hiệu quả.

PHẦN 5. THIẾT KẾ HỆ THỐNG

5.1. Kiến trúc hệ thống

Hệ thống thu thập dữ liệu phân tán mà chúng em đề xuất thiết kế để tối ưu hóa cả tốc độ và độ tin cậy trong việc thu thập thông tin từ internet. Kiến trúc tổng thể của hệ thống bao gồm các thành phần chính sau:

- **Các Nút Thu Thập (Crawlers):** Mỗi nút thu thập hoạt động độc lập, chịu trách nhiệm thu thập dữ liệu từ một hoặc nhiều trang web cụ thể. Các nút này có khả năng hoạt động đồng thời, giúp tăng cường hiệu suất của hệ thống.
- **Bộ Điều Phối (Coordinator):** Bộ điều phối quản lý việc phân phối công việc giữa các nút thu thập. Nó xác định nhiệm vụ nào sẽ được gửi đến nút nào dựa trên khả năng và tình trạng hiện tại của từng nút.
- **Cơ sở Dữ liệu (Database):** Dữ liệu thu thập được sẽ được lưu trữ trong một cơ sở dữ liệu, cho phép truy xuất và phân tích sau này. Cơ sở dữ liệu này cần phải có khả năng mở rộng để xử lý khối lượng dữ liệu lớn.
- **Giao Diện Người Dùng (User Interface):** Giao diện cho phép người dùng tương tác với hệ thống, theo dõi quá trình thu thập dữ liệu và truy cập vào dữ liệu đã được thu thập.

5.2. Các thành phần chính

- **Thu thập Dữ liệu:** Mỗi nút thu thập được lập trình để thực hiện các truy vấn, phân tích nội dung và lưu trữ dữ liệu cần thiết.
- **Quản lý Tải:** Bộ điều phối sẽ theo dõi tình trạng của các nút và điều chỉnh khối lượng công việc để đảm bảo không có nút nào bị quá tải.
- **Đảm bảo Tính Toàn Vẹn Dữ liệu:** Hệ thống cần có cơ chế kiểm tra và xác thực để đảm bảo rằng dữ liệu thu thập được là chính xác và không bị lỗi.

5.3. Sơ đồ hệ thống

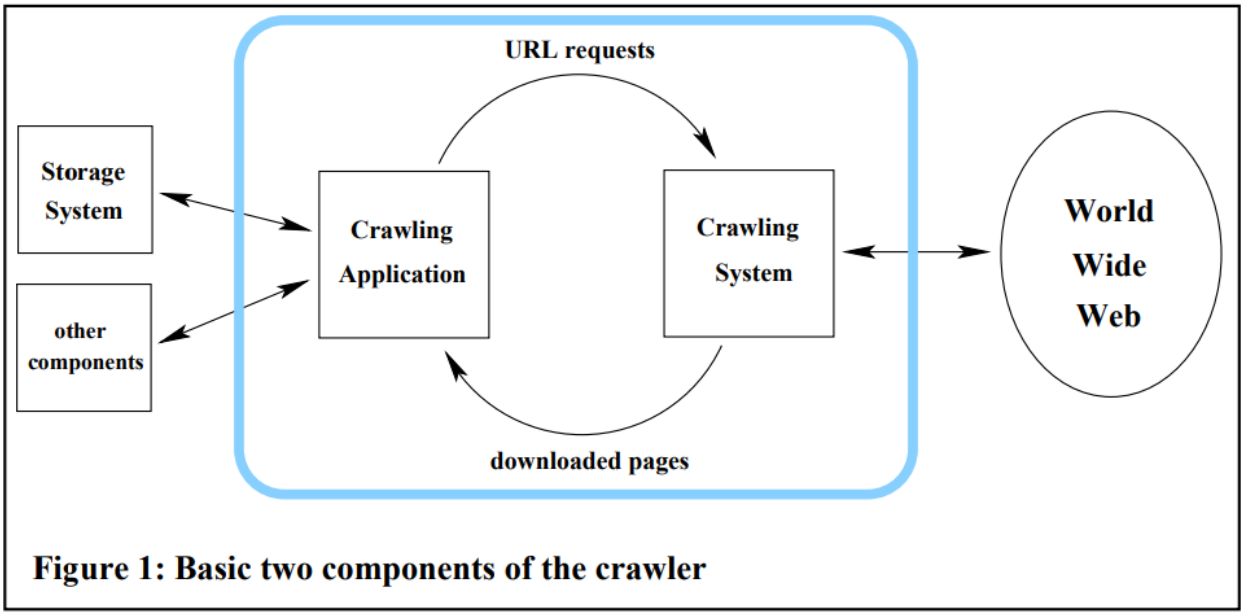


Figure 1: Basic two components of the crawler

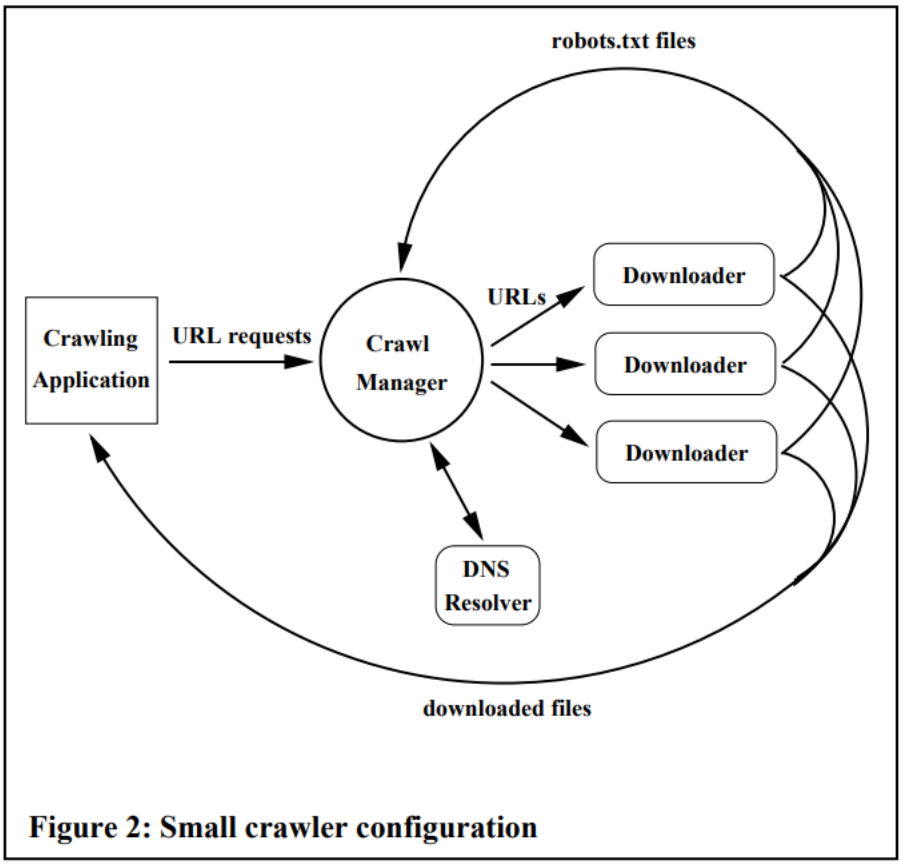


Figure 2: Small crawler configuration

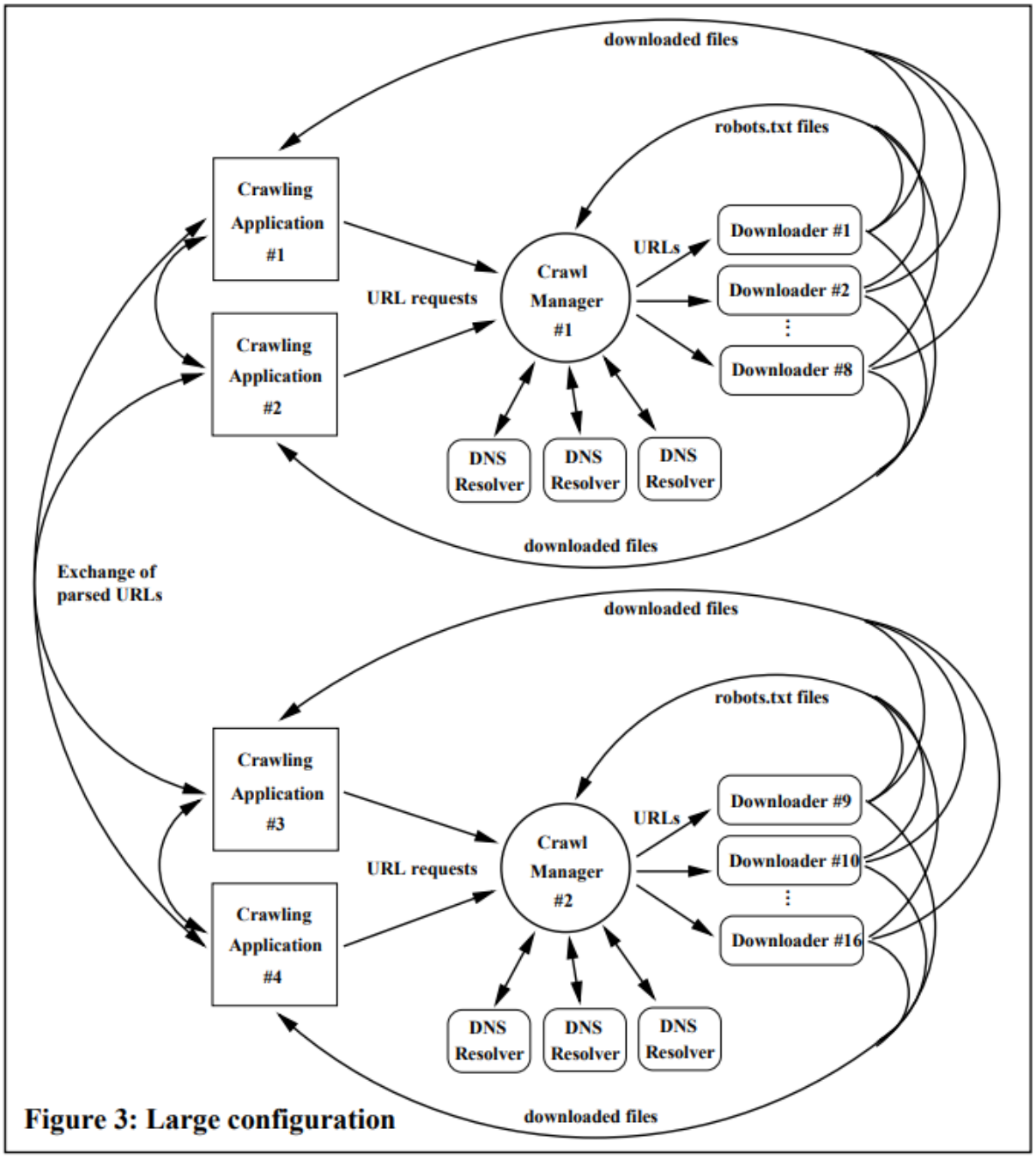


Figure 3: Large configuration

PHẦN 6. CÔNG CỤ THIẾT KẾ

6.1. Ngôn ngữ lập trình

Trong đề tài này, chúng em sẽ sử dụng Python là ngôn ngữ lập trình chính. Python không chỉ nổi bật với cú pháp đơn giản và dễ đọc mà còn có một hệ sinh thái phong phú với nhiều thư viện hỗ trợ việc thu thập và phân tích dữ liệu web.

Thư viện chính:

- **BeautifulSoup:** Dùng để phân tích cú pháp HTML và XML, giúp chúng ta dễ dàng truy cập và thao tác với nội dung của các trang web.
- **Scrapy:** Là một framework mạnh mẽ để xây dựng trình thu thập dữ liệu, hỗ trợ thu thập dữ liệu từ nhiều nguồn một cách đồng thời và hiệu quả.
- **Requests:** Thư viện cho phép gửi các yêu cầu HTTP dễ dàng để truy cập dữ liệu từ các trang web.

6.2. Công cụ và thư viện

Bên cạnh các ngôn ngữ lập trình và thư viện đã nêu, chúng em cũng sẽ sử dụng một số công cụ khác để tối ưu hóa quy trình thu thập dữ liệu:

- **Apache Kafka:** Một hệ thống phân phối cho phép xử lý dữ liệu theo thời gian thực, giúp quản lý luồng dữ liệu giữa các nút thu thập và bộ điều phối.
- **MongoDB:** Cơ sở dữ liệu NoSQL sẽ được sử dụng để lưu trữ dữ liệu thu thập được. MongoDB phù hợp cho việc lưu trữ dữ liệu phi cấu trúc và cho phép truy xuất dữ liệu linh hoạt.
- **Redis (Remote Dictionary Server):** Một cơ sở dữ liệu NoSQL lưu trữ dữ liệu dưới dạng key-value, được sử dụng chủ yếu như một bộ nhớ đệm (cache) hoặc một cơ sở dữ liệu in-memory (lưu trữ trong bộ nhớ RAM). Redis hỗ trợ quản lý hàng đợi công việc (Task Queue), đồng bộ trạng thái giữa các Crawler, phân chia công việc giữa các Crawler-nodes.

6.3. Mạng và giao thức

Hệ thống sẽ sử dụng các giao thức mạng tiêu chuẩn như **HTTP/HTTPS** để giao tiếp giữa các nút thu thập và máy chủ web.

Ngoài ra, chúng em sẽ áp dụng giao thức **RESTful** cho giao diện người dùng, giúp người dùng dễ dàng tương tác với hệ thống và truy xuất dữ liệu đã thu thập.

6.4. Cấu trúc dữ liệu

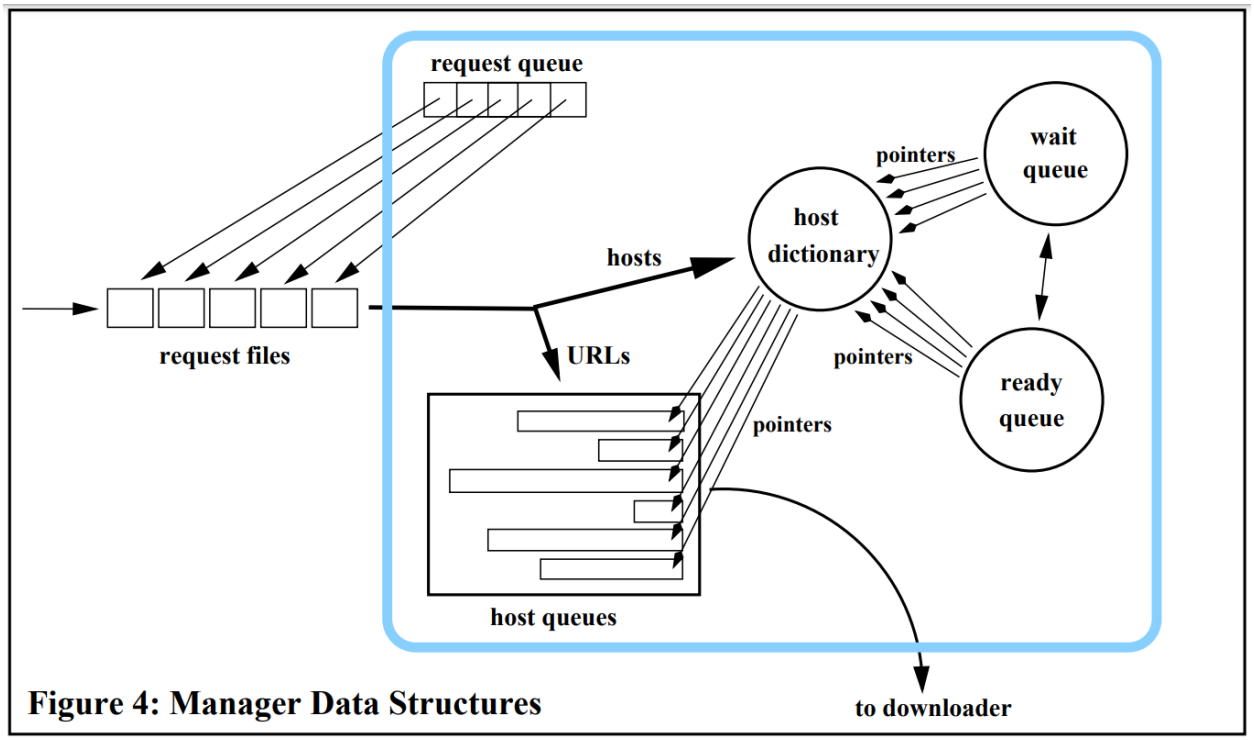


Figure 4: Manager Data Structures

PHẦN 7. KẾ HOẠCH THỰC HIỆN

7.1. Cài đặt môi trường

a) Cài đặt Scrapy và Scrapy Cluster

- Sử dụng lệnh sau để cài đặt Scrapy:

```
pip install scrapy
```

- Để cài đặt Scrapy Cluster, clone project từ GitHub và cài đặt các yêu cầu:

```
git clone https://github.com/istresearch/scrapy-cluster.git

cd scrapy-cluster

pip install -r requirements.txt
```

b) Cài đặt Redis và Kafka

- Redis và Kafka là hai thành phần quan trọng trong hệ thống phân tán. Ta có thể cài đặt Redis và Kafka bằng Docker Compose.

```
docker run -d --name redis -p 6379:6379 redis
```

- Cài đặt kafka:

```
docker run -d --name zookeeper -p 2181:2181 zookeeper:3.4.13

docker run -d --name kafka -p 9092:9092 --link zookeeper wurstmeister/kafka
```

- Sau đó, khởi chạy Docker Compose:

```
docker-compose up -d
```

c) Cài đặt MongoDB

- Để lưu trữ dữ liệu, ta cần MongoDB. Cài đặt MongoDB cũng có thể được thực hiện qua Docker:

```
docker run -d -p 27017:27017 --name mongodb mongo
```

7.2. Quy trình hoạt động

- **URL Frontier:** Danh sách URL đầu vào được đẩy vào Redis. Mỗi URL là một công việc crawl mà các node crawler sẽ thực hiện.

- **Crawler Nodes:** Mỗi node crawler lấy công việc từ Redis, thực hiện việc crawl dữ liệu từ URL được giao, và sau đó gửi dữ liệu crawl được về MongoDB.
- **Data Storage:** Dữ liệu được crawler thu thập (bao gồm nội dung trang web, tiêu đề, và các metadata khác) sẽ được lưu trữ trong MongoDB để xử lý tiếp theo.
- **Load Balancing:** Redis và Kafka đảm bảo rằng các công việc crawl được phân phối đều cho tất cả các node crawler, giúp hệ thống chịu tải tốt và dễ dàng mở rộng.

7.3. Cài đặt mã nguồn Scrapy Cluster

Dưới đây là mã Scrapy Spider để crawl các URL phân tán bằng Scrapy-Redis:

```
import scrapy

from scrapy_redis.spiders import RedisSpider

class DistributedCrawler(RedisSpider):

    name = "distributed_crawler"

    redis_key = "crawl:start_urls"

    def parse(self, response):

        page_title =
response.xpath('//title/text()').get()

        page_url = response.url

        yield {

            'url': page_url,

            'title': page_title

        }
```

7.4. Chạy hệ thống phân tán

Các bước để chạy hệ thống:

Bước 1. Khởi động Redis và MongoDB (sử dụng Docker hoặc cài đặt trực tiếp).
Bước 2. Khởi động Crawler Node: Mở terminal và chạy Scrapy spider. Scrapy sẽ kết nối với Redis để lấy URL:scrapy runspider distributed_crawler.py
Bước 3. Đẩy URL vào Redis: redis-cli lpush crawl:start_urls “https:….”
Bước 4. Khi spider crawl xong, kết quả sẽ được lưu vào MongoDB.

7.5. Phân chia công việc giữa các node crawler

7.5.1. Phân chia công việc với Kafka

Kafka được sử dụng để phân phối các URL cần crawl giữa nhiều node crawler. Kafka hoạt động dựa trên cơ chế Publish-Subscribe và Consumer Groups.

Cách thức hoạt động:

Bước 1. Producer (Scheduler) sẽ đẩy URL vào một Kafka topic.

Bước 2. Các node crawler (Consumer) sẽ tham gia vào một consumer group và mỗi node sẽ nhận một tập hợp URL riêng biệt để crawl.

7.5.2. Phân chia công việc với Redis

Redis với Scrapy-Redis có thể được sử dụng để quản lý và phân phối các URL giữa nhiều node crawler. Redis lưu trữ các URL cần crawl dưới dạng danh sách (list), và các node crawler có thể đồng thời lấy URL từ danh sách này mà không bị trùng lặp.

Cách thức hoạt động:

Bước 1. Redis đóng vai trò là message broker, lưu trữ các URL trong một danh sách (list).

Bước 2. Các node crawler sẽ kết nối đến Redis và lần lượt lấy các URL để crawl.

PHẦN 8. HIỆU SUẤT VÀ TỐI ƯU HOÁ

8.1. Các yếu tố ảnh hưởng đến hiệu suất

Trong việc thiết kế hệ thống thu thập dữ liệu phân tán, hiệu suất là yếu tố quyết định đến khả năng hoạt động của hệ thống. Một số yếu tố chính ảnh hưởng đến hiệu suất bao gồm:

- **Tốc độ truy cập dữ liệu:** Thời gian cần thiết để tải một trang web và xử lý dữ liệu từ nó. Sử dụng các kỹ thuật như Caching (lưu trữ tạm thời) và Pooling (tạo nhóm các kết nối) có thể giúp giảm thời gian truy cập.
- **Khả năng xử lý đồng thời:** Số lượng nút thu thập hoạt động đồng thời ảnh hưởng đến khả năng mở rộng của hệ thống. Hệ thống nên có khả năng mở rộng quy mô dễ dàng để xử lý nhiều yêu cầu đồng thời.
- **Tính khả dụng của dữ liệu:** Dữ liệu cần được thu thập và lưu trữ một cách nhất quán và tin cậy. Sử dụng các công nghệ như Distributed Database (cơ sở dữ liệu phân tán) giúp nâng cao tính khả dụng và độ bền vững của dữ liệu.

8.2. Kỹ thuật tối ưu hoá

- **Phân chia công việc:** Bộ điều phối sẽ chia nhỏ nhiệm vụ thu thập dữ liệu và phân bổ cho các nút thu thập khác nhau dựa trên khả năng và tải trọng hiện tại của từng nút. Điều này giúp tối ưu hóa thời gian xử lý.
- **Giảm thiểu yêu cầu không cần thiết:** Hệ thống sẽ được lập trình để kiểm tra dữ liệu đã thu thập trước đó và chỉ thu thập lại khi có sự thay đổi đáng kể. Điều này không chỉ tiết kiệm tài nguyên mà còn giảm tải cho các trang web mục tiêu.
- **Quản lý tài nguyên:** Sử dụng các công cụ giám sát để theo dõi hiệu suất của các nút thu thập và điều chỉnh tài nguyên một cách linh hoạt, đảm bảo không có nút nào bị quá tải hoặc ngừng hoạt động.

8.3. Phương thức đánh giá hiệu suất

Để đảm bảo hiệu suất của hệ thống, chúng em sẽ thực hiện các bước đánh giá định kỳ thông qua:

- **Kiểm tra tải (Load Testing):** Mô phỏng nhiều yêu cầu đồng thời để kiểm tra khả năng chịu tải của hệ thống.
- **Theo dõi hiệu suất thời gian thực:** Sử dụng các công cụ giám sát như Prometheus hoặc Grafana để theo dõi hiệu suất của các nút thu thập và phân tích dữ liệu trong thời gian thực.

PHẦN 9. TÍNH ỨNG DỤNG ĐỀ TÀI

- **Học máy và Trí tuệ nhân tạo:** Tích hợp các kỹ thuật học máy để cải thiện khả năng phân tích dữ liệu thu thập được. Hệ thống có thể tự động nhận diện các mẫu, xu hướng và đưa ra các dự đoán dựa trên dữ liệu lịch sử.
- **Mở rộng lĩnh vực áp dụng:** Hệ thống có thể được áp dụng cho nhiều lĩnh vực khác nhau, từ tiếp thị đến nghiên cứu khoa học, giúp các tổ chức tận dụng tối đa dữ liệu mà họ thu thập được.
- **Tối ưu hóa và Bảo mật:** Tiếp tục nghiên cứu các phương pháp bảo mật để bảo vệ dữ liệu thu thập được và đảm bảo tuân thủ các quy định về quyền riêng tư.
- **Hợp tác và chia sẻ dữ liệu:** Khả năng kết nối với các hệ thống và dịch vụ khác để chia sẻ dữ liệu và hợp tác trong việc phát triển các giải pháp sáng tạo hơn.

PHẦN 10. KẾT LUẬN

Đề tài "Thiết kế hệ thống dựa trên trình thu thập dữ liệu thông tin web phân tán" không chỉ mang lại một giải pháp hiệu quả cho việc thu thập dữ liệu từ internet mà còn đặt nền tảng cho những tiến bộ trong công nghệ khai thác dữ liệu. Với việc áp dụng các công nghệ hiện đại như Python, Scrapy và MongoDB, chúng em hy vọng sẽ tối ưu hóa quá trình thu thập thông tin, từ đó giúp tổ chức và doanh nghiệp khai thác được những giá trị quý báu từ dữ liệu lớn.

Nhờ vào kiến trúc phân tán, hệ thống sẽ có khả năng mở rộng tốt, cho phép xử lý nhiều nguồn dữ liệu đồng thời mà vẫn đảm bảo hiệu suất cao và độ tin cậy. Điều này là rất quan trọng trong bối cảnh hiện nay, khi mà thông tin trên internet đang phát triển và thay đổi một cách chóng mặt.

HẾT