BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC ĐÀ NẴNG

LÊ VĂN TIÊN

ỨNG DỤNG HỆ PHÂN TÁN ĐỂ TỐI ƯU THỜI GIAN XỬ LÝ CHO MÁY TÌM KIẾM

LUẬN VĂN THẠC SĨ KỸ THUẬT

ĐÀ NẴNG – Năm 2011

LỜI CAM ĐOAN

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi dưới sự hướng dẫn khoa học của PGS. TS. Lê Văn Sơn. Các số liệu và kết quả nêu trong luận là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác.

Người cam đoan

Lê Văn Tiên

MỤC LỤC

DANH MỤC CÁC TỪ VIẾT TẮT

SE	Máy tìm kiếm
DS	Hệ phân tán
DNS	Hệ thống tên miền
MON	Truyền thông hướng thông điệp
SOM	Truyền thông thướng dòng
RPC	Gọi thủ tục từ xa
MDR	Nhịp trôi lớn nhất của đồng hồ
WWV	Thời gian quốc tế
UTC	Giờ phối hợp quốc tế
P	Tiến trình

DANH MỤC CÁC BẢNG

DANH MỤC CÁC HÌNH

MỞ ĐẦU

1. Lý do chọn đề tài

Hơn 40 năm kể từ khi internet ra đời cho đến nay, nó mang lại rất nhiều tiện ích hữu dụng cho người sử dụng điển hình như hệ thống thư điện tử (*email*), trò chuyện trực tuyến (*chat*), máy truy tìm dữ liệu (*search engine*), các dịch vụ thương mại, chuyển ngân và các dịch vụ về y tế giáo dục...Đi kèm với sự bùng nổ các dịch vụ trên internet là sự dùng nổ về số lượng website trên internet, hiện tại số lượng website đã lên con số hàng tỉ và không ngừng tăng lên theo thời gian, đứng đầu là tên miền có đuôi *.com*, theo thống kê mới nhất đã lên tới 84.000.000 tên miền. Tên miền có đuôi *.vn* cũng đã lên tới 140.000 tên miền. Chính sự bùng nổ về số lượng website trên internet đã bổ sung cho kho thông tin càng ngày càng khổng lồ hơn và ngày nay hầu như mọi kiến thức của mọi lĩnh vực đều có thể tìm thấy trên internet.

Vấn đề đặt ra ở đây là làm thế nào để tìm kiếm một mẫu thông tin trong kho tàng thông tin khổng lồ như vậy một cách chính xác và nhanh nhất, lời giải cho câu hỏi đó là sử dụng máy tìm kiếm (***search engine)*** và hiện nay nhiều nhà dịch vụ đã sử dụng nó rất thành công, điển hình như: Google, Yahoo, Mirosoft…

Máy tìm kiếm đã xuất hiện và được đưa vào sử dụng từ rất sớm, nhưng để tối ưu hóa sao cho thời gian trả lời kết quả tìm kiếm nhanh nhất và chính xác nhất thì các chuyên gia cũng đang ngày càng hoàn thiện.

Trong thời gian gần đây nhờ sự phát triển vượt bậc của lĩnh vực phần cứng CNTT và truyền thông, nhờ vậy mà một giải pháp mới cho các ứng dụng CNTT được ra đời và đang được các chuyên gia đánh giá cao về lợi ích mà mó mang lại đó là "**Hệ phân tán - Distributed Systems**".

Hệ phân tán là hệ thống xử lý thông tin bao gồm nhiều bộ xử lý hoặc bộ vi xử lý nằm tại các vị trí khác nhau được liên kết với nhau thông qua phương tiện viễn thông dưới sự điều khiển thống nhất của một hệ điều hành nhằm tăng tốc độ

bình quân trong tính toán xử lý, cải thiện tình trạng luôn sẵn sàng của các loại tài nguyên, tăng độ an toàn cho dữ liệu, đa dạng hóa các loại hình dịch vụ tin học, bảo đảm tính toàn vẹn của thông tin.

Xuất phát từ nhu cầu và các tiền đề trên, việc tối ưu hóa máy tìm kiếm thông tin, mà đặc biệt là tối ưu thời gian tìm kiếm thông tin của máy tìm kiếm là vấn đề rất có ý nghĩa trong giai đoạn CNTT hiện nay và tương lai. Chính vì vậy tôi chọn hướng nghiên cứu này và áp dụng "*hệ phân tán*" để tối ưu thời gian xử lý cho máy tìm kiếm và lấy tên đề tài là ***"ứng dụng hệ phân tán để tối ưu thời gian xử lý cho máy tìm kiếm"***.

2. Mục đích và nghiệm vụ nghiên cứu của đề tài

Mục đích của đề tài là nghiên cứu áp dụng hệ phân tán vào máy tìm kiếm nhằm giải quyết 3 yêu cầu đặt ra như sau:

Một: *Giảm thời gian tìm kiếm cho máy tìm kiếm:* có 3 nguyên nhân chính

+ Giảm tải lượng truy cập vào tài nguyên chung
+ Rút ngắn khoảng cách vật lý giữa người dùng và server
+ Tăng tốc độ tính toán – xử lý

Hai: *Tăng độ an toàn cho dữ liệu cho máy tìm kiếm**:*** có 3 nguyên nhân chính

+ Dữ liệu được đặt tại nhiều server khác nhau và có khả năng phục hồi
+ Đảm bảo tính đồng bộ dữ liệu giữa các server
+ Đảm bảo được tính toàn vẹn của dữ liệu

Ba: *Đảm bảo hệ thống luôn hoạt động thông suốt**:*** có 3 nguyên nhân chính

+ Tính co giãn của hệ thống cao
+ Tính chịu lỗi của hệ thống cao
+ Tính mở của hệ thống cao

3. Đối tượng và phạm vi nghiên cứu

- Nghiên cứu mô hình hoạt động tổng thể của máy tìm kiếm và một số giải pháp tìm kiếm thông dụng
- Nghiên cứu hệ phân tán đa server
  + Xây dựng hệ phân tán đa server
  + Lưu trữ, truy xuất dữ liệu trên hệ phân tán đa server
- Nghiên cứu, ứng dụng hệ phân tán vào máy tìm kiếm
- Nghiên cứu và áp dụng bộ định tuyến ưu tiên yêu cầu (*Request*) người dùng
- Ngôn ngữ lập trình Java, Lucene
- Hệ quản trị cơ sở dữ liệu My SQL

4. Giả thiết nghiên cứu

- Hiểu được quá trình hoạt động và một số giải pháp xây dựng máy SE
- Hiểu được bản chất của hệ phân tán và quá trình trao đổi thông tin giữa các thành phần trong hệ
- Hiểu thêm ngôn ngữ lập trình Java, Lucene và hệ quản trị cơ sở dữ liệu My SQL
- Hiểu và vận dụng được giải pháp ứng dụng hệ phân tán để tối ưu thời gian tìm kiếm cho máy SE

5. Phương pháp nghiên cứu

- Thu thập, tìm hiểu, phân tích các tài liệu và thông tin có liên quan đến luận văn
- Phân tích, nắm rõ quá trình hoạt động của máy tìm kiếm
- Nắm rõ cách xây dựng, truy xuất và lưu trữ dữ liệu trên hệ phân tán

- Phân tích, tìm hướng giải quyết cho các vấn đề nảy sinh khi áp dụng hệ phân tán vào máy SE
- Triển khai xây dựng chương trình chạy trên hệ phân tán
- Triển khai xây dựng chương trình chạy trên hệ tập trung
- Kiểm thử, đánh giá kết quả và rút ra kết luận

6. Ý nghĩa khoa học và thực tiễn của đề tài

- Nghiên cứu, nắm vững phương pháp thực hiện của máy tìm kiếm
- Nghiên cứu, nắm vững bản chất và phương pháp hoạt động của hệ phân tán đa server
- Nghiên cứu, xây dựng một mô hình lưu trữ thông tin mới cho máy tìm kiếm
- Giảm đáng kể thời gian thực hiện cho máy tìm kiếm
- Tăng độ an toàn cho dữ liệu
- Đảm bảo hệ thống luôn thông suốt
- Mang lại lợi ích ứng dụng rất lớn

CHƯƠNG 1: TỔNG QUAN VỀ MÁY TÌM KIẾM

Máy tìm kiếm (*tiếng Anh: search engine*), hay còn được gọi với nghĩa rộng hơn là công cụ tìm kiếm (*search tool*), nguyên thuỷ là một phần mềm nhằm tìm ra các trang web trên mạng Internet có nội dung theo yêu cầu người dùng dựa vào các thông tin mà chúng có. Trữ lượng thông tin này của công cụ tìm kiếm thực chất là một loại cơ sở dữ liệu (*database*) cực lớn. Việc tìm các tài liệu sẽ dựa trên các từ khóa (*keyword*) được người dùng gõ vào và trả về một danh mục của các trang Web có nội dung chứa từ khóa mà nó tìm được.

Máy tìm kiếm hoạt động dựa vào 3 bộ chính:

- Bộ thu thập thông tin – Robot
- Bộ lập chỉ mục – Index
- Bộ tìm kiếm thông tin – Search Engine

1.1 Giới thiệu một số máy tìm kiếm thông dụng

Bảng 1.2. Bảng xếp hạng search engine năm 2009

Top Search Engines for 2009

* Search.MSN.com 302 redirects to Live.com

2009	Google	Yahoo!	MSN/Live	Ask	Total
2009 05	72.92%	16.14%	5.68%	3.95%	98.69%
2009 04	72.68%	16.29%	5.67%	3.96%	98.60%
2009 03	72.13%	16.56%	5.50%	4.02%	98.21%
2009 02	72.11%	17.52%	5.55%	3.47%	98.65%

Thế giới

google.com

Hình 1.1 Giao diện của google search engine

Google là bộ máy tìm kiếm (Search Engine) hiện đang được đánh giá là "vô địch" trên Internet, với trên 4,2 tỷ trang Web đã được lập chỉ mục và có tốc độ tìm kiếm cực nhanh. Google không chỉ là công cụ tìm kiếm được hầu hết những người lướt Web sử dụng do hỗ trợ tới 97 ngôn ngữ, đây còn là tiện ích tìm kiếm được nhúng vào rất nhiều website (*một dịch vụ được Google cung cấp dưới nhiều hình thức và cho những đối tượng khác nhau*).

Các bộ tìm kiếm của google

Google không ngừng tìm kiếm và cập nhật các trang mới để thêm vào chỉ mục của bạn. Có chương trình phụ trách vấn đề này được gọi là các robot hay bọ tìm kiếm (Googlebot). Các Googlebot được gọi chương trình tìm kiếm có nhiệm vụ duy

nhất là để thu thập tài liệu web để xây dựng một cơ sở dữ liệu được sử dụng bởi các công cụ tìm kiếm của nó.

Các Googlebot sử dụng một quy trình dựa trên thuật toán xác định các trang web để thu thập dữ liệu, tần số và số lượng trang để tìm nạp từ mỗi trang web. Danh sách này các trang web toàn diện để xác định các liên kết đến các trang khác.

Bộ lập chỉ mục của google

Đánh chỉ mục là một quá trình quét qua các trang web và tạo ra chỉ số có sử dụng Google để cho kết quả khi bạn tìm kiếm. Thực tế, các robot các phân tích và đưa ra một chỉ mục của tất cả các từ họ xem và vị trí của họ. Và việc trang web có được Google đánh chỉ hay không luôn là mối quan tâm hàng đầu của các nhà thiết kế web hiện nay.

Các loại dữ liệu google có thể tìm kiếm

Không hẳn vậy, Google cũng trích xuất thông tin chỉ mục hoặc nhiều loại tập tin khác nhau: PDF, PS (Adobe PostScript), Excel (xls), tài liệu, văn bản MW, DOC, WRI, RTF, ANS, TXT, thuyết trình PowerPoint (ppt) các tập tin, Microsoft Works (wks, wps, Wdb) và swf.

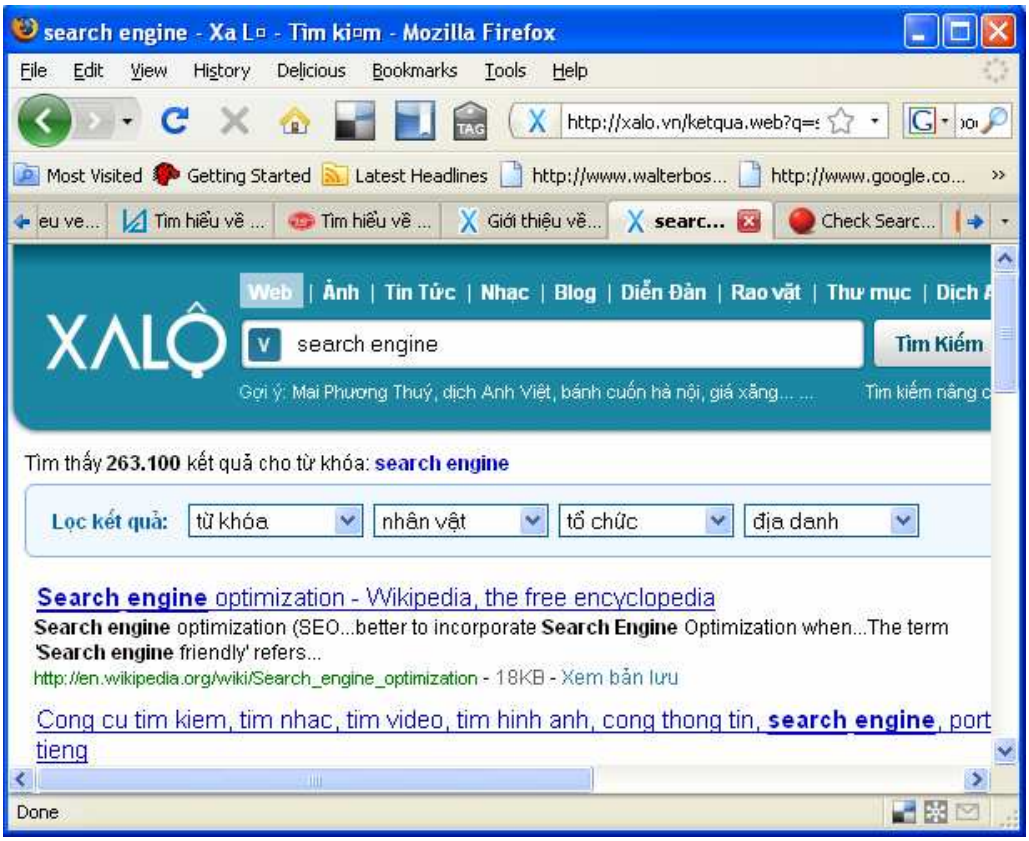
Điều này được thực hiện để cung cấp cho Google nhiều kết quả hơn, trên thực tế, trong quá trình thực hiện tìm kiếm bạn cũng có thể thấy hiển thị một số loại tập tin khác html, ví dụ: file .doc hay .pdf

Bộ pageRank của google

Google PageRank là một hệ thống có nhiệm vụ xếp hạng các trang web, được phát triển bởi Larry Page và Sergey Brin thuộc Đại học Stanford. Trong khi hiện nay Google có rất nhiều kỹ sư làm việc để cải thiện về mọi mặt của Google hàng ngày, PageRank tiếp tục đóng một vai trò trung tâm trong nhiều công cụ tìm kiếm web của Google.

Việt Nam

xalo.vn



Hình 1.2 Giao diện của xalo.vn search engine

Xalo.vn là một Máy tìm kiếm (search engine) được Tinhvân Media phát triển với tham vọng Xalo.vn sẽ trở thành công cụ tìm kiếm tiếng Việt hàng đầu của Việt Nam.

Xalo.vn hiện tại đang cung cấp 7 dịch vụ tìm kiếm bao gồm:

- ***Tìm kiếm Web:*** dịch vụ tìm kiếm thông tin tổng hợp trên dữ liệu gần 100 triệu trang văn bản tiếng Việt hiện có trên các Website của Việt Nam

- ***Tìm kiếm Tin tức:*** dịch vụ tổng hợp tin tức và tìm kiếm thông tin trên dữ liệu dạng tin tức được tổng hợp từ gần 70 trang tin điện tử hàng đầu của Việt Nam

- ***Tìm kiếm Diễn đàn:*** dịch vụ tìm kiếm cho phép người dùng tìm kiếm thông tin từ hơn 100 diễn đàn lớn nhất của Việt Nam hiện tại.

- ***Tìm kiếm Ảnh:*** dịch vụ tìm kiếm hình ảnh trên số lượng hơn 20 triệu hình ảnh được người dùng Việt Nam đưa lên Internet.

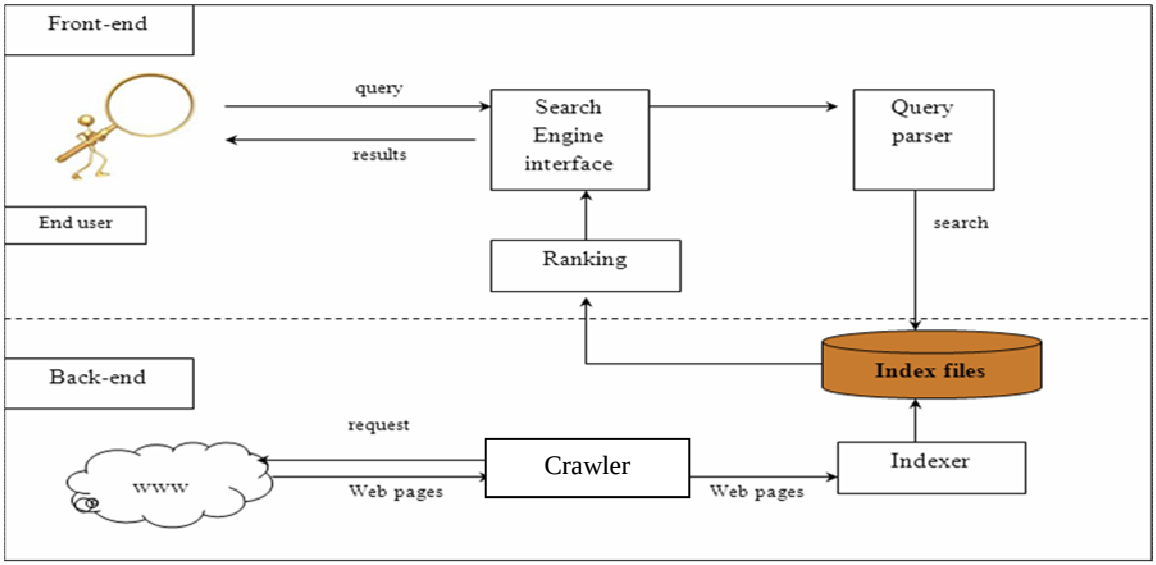
- ***Tìm kiếm Blog:*** dịch vụ tìm kiếm cho phép người dùng tìm kiếm thông tin trên hầu hết các mạng xã hội được cung cấp bởi Việt Nam cũng như trên thế giới mà người Việt Nam hay sử dụng

- ***Tìm kiếm Nhạc:*** dịch vụ tìm kiếm dữ liệu Nhạc từ các Website nghe nhạc trực tuyến lớn nhất Việt Nam hiện tại.

- ***Tìm kiếm Rao vặt:*** dịch vụ tổng hợp và tìm kiếm thông tin rao vặt từ hơn 20 Website mua bán rao vặt lớn nhất Việt Nam

Với các dịch vụ cung cấp và tính năng khác biệt cho từng dịch vụ, Xa Lộ đang không ngừng được hoàn thiện để có thể phục vụ tốt nhất nhu cầu tìm kiếm của người dùng Internet Việt Nam và trở thành máy tìm kiếm tiếng Việt hàng đầu của Việt Nam.

1.2 Kiến trúc và cơ chế hoạt động của máy tìm kiếm



Hình 1.3 Mô hình hoạt động của máy tìm kiếm

Máy tìm kiếm chi thành 2 phần chính **Front-end** và phần **Back-end**

- **Front- end:** Bao gồm giao diện người sử dụng (*Search engine interface*); bộ sắp xếp (*ranking*) và bộ xử lý yêu cầu người dùng (*query parser*)

Khi người sử dụng gửi một yêu cầu tìm kiếm một mẫu thông tin, máy tìm kiếm sẽ phân tích yêu cầu và gửi đến server, server thực hiện so khớp yêu cầu với dữ liệu trong kho index files và sắp xếp kết quả tìm được theo thứ tự từ cao đến của độ chính xác, cuối cùng là hiển thị kết quả cho người dùng.

- **Back-end:** Bao gồm bộ thu thập thông tin (*Crawler*) và bộ lập chỉ mục (*indexer*)

Bộ Crawler dựa vào các robot tìm kiếm sẽ tự động tìm kiếm thông tin trên internet và chuyển thông tin qua bộ indexer lập chỉ mục và lưu vào kho dữ liệu index files.

Các thành phần này sẽ được phân tích cụ thể ở phần sau.

1.3 Bộ thu thập thông tin – Crawler

Từ một hay nhiều các liên kết ban đầu, Crawler lên đường thực hiện công việc "lùng sục" Internet của mình. Crawler tải về nội dung các trang web từ các liên kết đã nhận ban đầu và truy xuất các liên kết mới nằm trong nội dung của các trang này. Các liên kết mới này sẽ được nạp vào một trình điều khiển (Crawler Manager). Crawler Manager sẽ quyết định các liên kết nào sẽ được viếng thăm kế tiếp, Crawler Manager sẽ nạp chúng vào hàng đợi để chờ xử lý. Các liên kết này sẽ được quản lý trong cơ sở dữ liệu để thuận tiện cho công việc cập nhật thông tin mới. Trong một lần thực hiện thì các liên kết phải chỉ được truy cập một lần để tăng khả năng hoạt động và tránh trùng lặp nội dung. Một crawler đi qua bốn bước cơ bản:

- Bắt đầu từ một hay nhiều liên kết
- Tải nội dung

- Phân tích nội dung, tìm liên kết, đi theo các liên kết
- Theo dõi liên kết, tránh trùng lặp

Có nhiều chế độ làm việc cho crawler thực hiện nhiệm vụ truy tìm thông tin. Các chế độ được phân biệt theo nhiều cách. Các đặc điểm phân biệt có thể là:

- Batch Mode
- Incremental Mode

Batch mode Crawler sẽ đánh chỉ mục liên tục các trang web và không tải nội dung về để lưu trữ. Cách này nội dung luôn được cập nhật nhưng chỉ phù hợp cho lượng trang web nhỏ có giới hạn. Chẳng hạn như mục tiêu của crawler được định ra là thực hiện trên một số website cụ thể nào đấy. Crawler chỉ có nhiệm vụ liên tục chạy qua các wesiste này để cập nhật các nội dung mới.

Incremental Mode hoạt động ở chế độ này crawler sẽ không bao giờ xóa các nội dung lưu trữ. Khi gặp một tài liệu được cho là đã viếng thăm thì crawler sẽ tuân theo chiến lược cập nhật nội dung đã được cài đặt. Ở chế độ này thì crawler cần phải có kho lưu trữ tài liệu thật lớn.

- Breadth-first(Tìm kiếm theo chiều rộng)
- Depth-first(Tìm kiếm theo chiều sâu)

1.3.1 Các thủ thuật tìm kiếm của Crawler

1.3.1.1Chiến thuật tìm kiếm theo chiều sâu (Depth-first)

Từ một danh sách chứa các liên kết cần duyệt, thực hiện các bước sau :

(1) Cho danh sách = {trang đầu tiên}
(2) Lấy trang đầu tiên trong danh sách.
* Nếu có qua (3)
* Nếu không qua (5)

(3) Trang này đã xét tới chưa ?

* Nếu rồi, quay lại (2)

* Nếu chưa, qua (4)

(4) Đánh dấu đã tới rồi. Phân tích và tìm xem liên kết có trong trang đó không?

(4a) Nếu có, thêm liên kết này vào đầu danh sách. Quay lại (4)

(4b) Nếu không, quay lại (2).

(5) Kết thúc.

1.3.1.2Chiến thuật tìm kiếm theo chiều rộng

Từ một danh sách chứa các liên kết cần duyệt, thực hiện các bước sau :

(1) Cho danh sách = {trang đầu tiên}

(2) Lấy trang đầu tiên trong danh sách.

* Nếu có qua (3)

* Nếu không qua (5)

(3) Trang này đã xét tới chưa ?

* Nếu rồi, quay lại (2)

* Nếu chưa, qua (4)

(4) Đánh dấu đã tới rồi. Phân tích và tìm xem liên kết có trong trang đó không?

* (4a) Nếu có, thêm liên kết này vào cuối danh sách. Quay lại (4)

* (4b) Nếu không, quay lại (2).

(5) Kết thúc.

1.3.1.3Chiến thuật tìm kiếm theo ngẫu nhiên

Từ một danh sách chứa các liên kết cần duyệt, thực hiện các bước sau : (1) Cho danh sách = {trang đầu tiên}

(2) Lấy ngẫu nhiên một trang trong danh sách.

* Nếu có qua (3)

* Nếu không qua (5)

(3) Trang này đã xét tới chưa ?

* Nếu rồi, quay lại (2)

* Nếu chưa, qua (4)

(4) Đánh dấu đã tới rồi. Phân tích và tìm xem liên kết có trong trang đó không?

* (4a) Nếu có, thêm liên kết này vào cuối danh sách. Quay lại (4)

* (4b) Nếu không, quay lại (2).

(5) Kết thúc.

1.3.2 Tính năng bắt buộc crawler phải tuân theo

- **Robustness:** Các crawler phải được thiết kế chính xác và hoạt động ổn định. Nhiều website có tạo ra chức năng đánh lừa các crawler. Tức là dẫn các crawler vào các vòng lặp không có kết thúc. Crawler cần phải được thiết kế sao cho có thể nhận ra "bẫy" và quay trở ra. Không phải "cái bẫy" nào cũng tạo ra nhằm đánh lừa các crawler mà đôi khi là vô tình bởi trang web thiết kế bị lỗi.

- **Politeness**: Các website sẽ có các chính sách cho phép các crawler truy cập vào trang web mình theo một cấp độ nào đó. Crawler cần phải tuân thủ các chính sách này.

1.3.3 Tính năng crawler nên tuân theo

Distributed: Các crawler cần được thiết kế theo mô hình phân tán để có thể thực thi trên nhiều máy tính.

Scalable: Crawler cần được thiết kế có khả năng tăng tần xuất hoạt động bằng nhiều cách. Chẳng hạn như thêm nhiều máy tính hoặc tăng dung lượng băng thông.

Performance and efficiency: Crawler cần được thiết kế hiệu thật hiệu quả trong việc sử dụng các tài nguyên hệ thống như bộ xử lý, dung lượng lưu trữ và băng thông mạng.

Freshness: Crawler cần phải được cập nhật nội dung mới một cách liên tục. Crawler cần phải có tần xuất truy cập trang web xấp xĩ với tần xuất thay đổi nội dung của trang web.

Extensible: Crawler cần được thiết kế sao cho dễ dàng mở rộng theo nhiều hướng. Chẳng hạn như tăng định dạng cho nội dung truy cập hay thêm giao thức truy cập Internet. Để được như vậy crawler cần được chia thành các phần nhỏ (các mudole) để tiện cho việc duy trì và nâng cấp.

1.3.4 Vấn đề cơ bản cần giải quyết của Crawler

Những trang web nào nên được tải về? Tải về tất cả các trang web đó là một việc làm không tưởng vì vậy cần phải có chiến lược lựa chọn các trang web quan trọng để tải về. Các trang web được nhiều truy cập, các trang web cung cấp nội dung có giá trị và phổ biến thì nên được vị trí ưu tiên trong hàng đợi.

Làm thế nào để cập nhật nội dung? Trên Internet các trang web cập nhật thường xuyên nội dung của nó. Có trang cập nhật liên tục có trang cập nhật trong thời gian lâu hơn. Làm thế nào để quyết định các trang web nào nên được truy cập lại và những trang web nào cần bỏ qua. Cũng như công việc trên ta không thể cập nhật lại toàn bộ các trang web một cách thường xuyên. Cần phải có chiến lược lựa chọn.

Làm thế nào để tải nội dung trang web tối ưu nhất. Trong khi crawler thực hiện công việc thu thập tài liệu sẽ tiêu tốn tài nguyên như CPU hay tài nguyên mạng. Nếu crawler chiếm quá nhiều tài nguyên mạng thì nó có thể bị người quản trị các website loại trừ. Cần phải có chiến lược để nâng cao khả năng hoạt động sao cho ít tốn tài nguyên nhất.

Làm thế nào để xử lý song song. Tài liệu trên Internet là vô cùng lớn cần phải có nhiều crawler hoạt động đồng thời. Làm thế nào để các crawler khác nhau sẽ không truy cập cùng một website ở các thời điểm khác nhau.

Khả năng lưu trữ: Yêu cầu tổng quan của một công cụ tìm kiếm và sao chép nội dung web bao hàm tính năng tải file, giao diện trực quan dễ mở rộng, bảo mật giữa server và trình duyệt web, trình diễn các thành phần động, và phản ảnh giao diện web một cách chính xác. Mục đích trong việc sao chép một trang web là phải phản ảnh lại chính xác giao diện của trang web, tải từng hình ảnh, liên kết cũng như thành phần động. Đây là một nhiệm vụ khó hầu hết bởi vì khả năng phân tích mã JavaScript của crawler còn thấp. Phạm vi hoạt động của crawler phụ thuộc vào cách trình bày của các trang web nguồn. Ví dụ cần phải có thuật toán để crawler truy lục hết các tài liệu trên cùng một website trước khi đi theo các liên kết đi ra các website khác. Hình ảnh nằm ở các website bên ngoài được tham chiếu tới bởi website này thì lại khác, tất cả các hình ảnh phải được lưu trữ để phục vụ cho việc lưu các trang web trong cache. HTTP header chuyển hướng (HTTP header redirect) nằm ở vị trí Location của HTTP header phải được lần theo. Các liên kết nằm hai server nhưng trên cùng một miền thì cần phải theo. Ví dụ như *mail.yahoo.com và quote.yahoo.com.* Nhiều liên kết và các dòng chuyển hướng thường được viết dưới mã JavaScript, các liên kết này cần phải lần theo. Cũng cần phải định nghĩa trước giới hạn độ sâu của liên kết.

1.3.5 Xây dựng Crawler

Có hai cấu trúc mà crawler có thể được xây dựng. Thứ nhất là xây dựng crawler như là một chương trình đệ quy. Thứ hai là xây dựng crawler theo cấu trúc dữ liệu sử dụng hàng đợi.

Chương trình đệ quy là một kỹ thuật lập trình mà các phương thức tự gọi lại chính nó. Chương trình đệ quy phù hợp cho các dự án mà các công việc được thực thi lặp lại một cách hệ thống và thông tin mà công việc trong tương lai cũng được biết trước sẽ giống như công việc đang được thực hiện hiện tại. Chương trình đệ quy chỉ nên dùng cho crawler mà biết trước sẽ có số ít các trang web vì mỗi khi chương trình chạy sẽ lưu dữ liệu vào ngăn xếp với số lượng trang web lớn thì ngăn xếp sẽ rất lớn và có thể gây tràn ngăn xếp nguyên nhân ngăn chặn không cho

chương trình có thể hoạt động. Lý do nữa là ta sẽ không thể thực hiện khả năng đa luồng cho crawler đệ quy, bởi khi ta đệ quy sẽ tạo ra nhiều tiến trình. Mỗi tiến trình đóng vai trò là một crawler, mỗi tiến tình lại có một ngăn xếp riêng mỗi khi chương trình gọi lại chính nó thì sẽ tạo ra một ngăn xếp mới điều này không phù hợp vì cần phải dùng một chung một ngăn xếp cho một chương trình crawler.

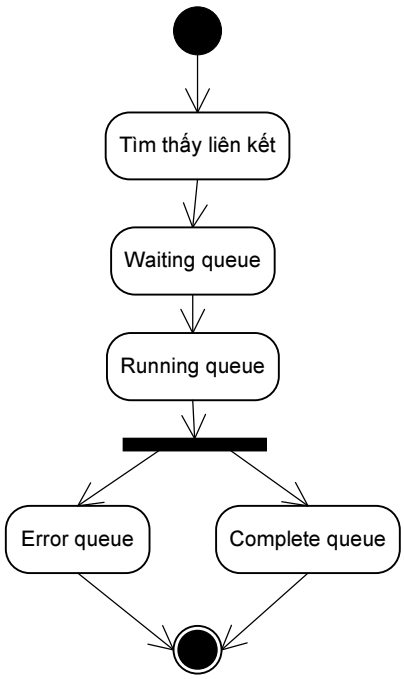
Khi xây dựng crawler sử dụng cấu trúc dữ liệu hàng đợi, đầu tiên crawler sẽ nhận được liên kết của trang web gốc. Crawler sẽ nạp liên kết này vào hàng đợi. Khi crawler tìm thấy các liên kết mới thì nó cũng đưa các liên kết này vào hàng đợi. Khi xử lý xong một liên kết crawler sẽ lấy một liên kết trong hàng đợi ra và tiếp tục truy cập. Công việc sẽ kết thúc khi không còn một liên kết nào trong hàng đợi.

Như mô tả thì một crawler chỉ cần một hàng đợi là có thể thực thi, Nhưng thông thường ta dùng bốn hàng đợi để lưu trữ các liên kết để theo dõi các trạng thái của nó.

- **Waiting Queue** trong hàng đợi này thì các liên kết đang chờ để được xử lý và liên kết mới sẽ được nạp vào hàng đợi này khi chúng được tìm thấy.
- **Running Queue** liên kết được chuyển tới hàng đợi này khi crawler bắt đầu xử lý nó. Một điều quan trọng cần tránh là một liên kết phải chỉ được xử lý một lần để tránh lãng phí tài nguyên. Khi một liên kết đã được xử lý thì hoặc nó được chuyển tới Error Queue hoặc Complete Queue.
- **Error Queue** khi có lỗi xảy ra trong quá trình xử lý liên kết thì liên kết này sẽ được chuyển tới Error Queue. Một khi đã vào đây thì liên kết sẽ không chuyển đi đâu nữa và cũng sẽ không được xử lý lần nào nữa.
- **Complete Queue** khi xử lý thành công thì các liên kết sẽ được chuyển tới đây và cũng sẽ không chuyển tới đâu nữa.

Một liên kết sẽ chỉ ở một hàng đợi tại một thời điểm. Vì vậy mỗi thời điểm có thể được coi là mỗi trạng thái của liên kết. Chương trình máy tính thường được

mô tả theo biểu đồ trạng thái trong đó sẽ biểu diễn luồng di chuyển của liên kết từ trạng thái này sang trạng thái khác.



Hình 1.4 Biểu đồ trạng thái của một liên kết

1.3.6 Vấn đề cần tránh

Quá tải mạng và server: Sử dụng robot, một điều tiên quyết phải chú ý đó là tài nguyên mạng phải lớn. Nhiều robot hoạt động sẽ gây tốn một lượng lớn băng thông cũng như các tài nguyên mạng khác, tốc độ xử lý, dung lượng bộ nhớ v.v...

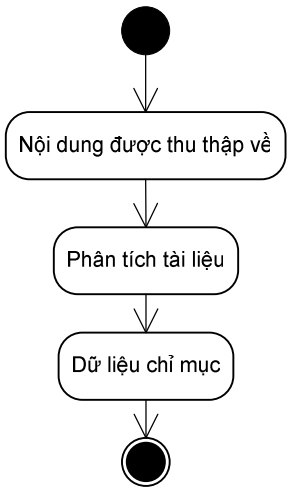
Cập nhật quá mức: Khó có thể kiểm soát được tốc độ cập nhật mới của trang web. Cập nhật thông tin là quan trọng. Nhưng phải có chiến lược cập nhật phù hợp sao cho tốc độ cập nhật xấp xỉ với tốc độ thay đổi nội dung của một trang web sao cho không quá thường xuyên gây tốn kém tài nguyên, hay quá rời rạc dữ liệu không được cập nhật cho người dùng.

Những tình huống không mong đợi khác: Robot không thể nhận ra các url khác nhau về tên nhưng cùng dẫn về một địa chỉ. Chẳng hạn như DSN và IP. Robot cũng có thể đi vào những vòng lặp vô hạng khi các trang có sử dụng sessionId. Mỗi phiên

làm việc sẽ tạo ra một Id, id này có thể là tham số của liên kết. Các liên kết cứ tạo ra liên tục như vậy. Robot theo các liên kết này sẽ không có điểm dừng. Một tình huống không mong đợi nữa là trong các website có sử dụng lịch (calendars) thành phần lịch này sử dụng các trang web động cùng với các liên kết cứ liên tục chỉ đến những ngày, hay năm trong tương lai. Spider đi theo các liên kết này cũng sẽ không có điểm dừng.

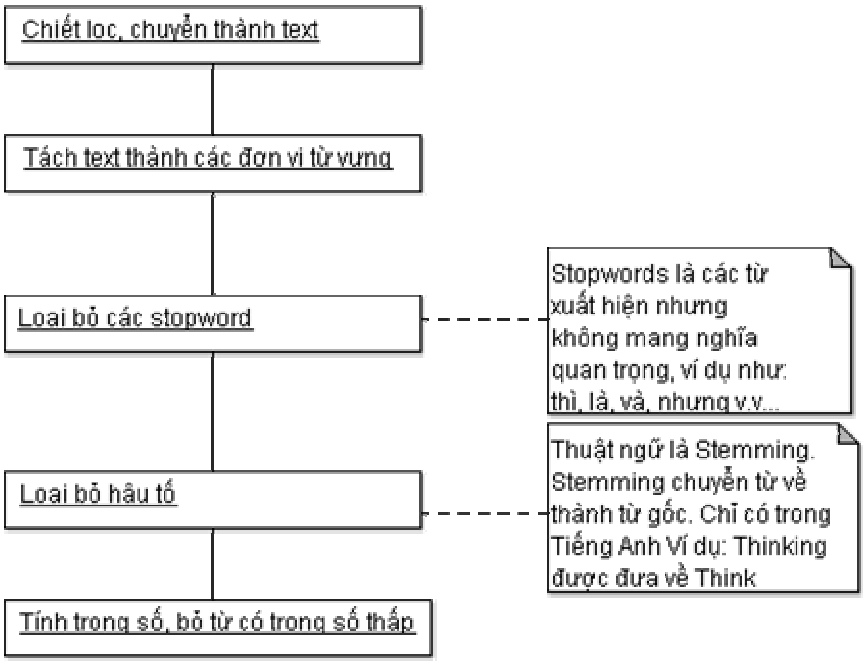
1.4 Bộ lập chỉ mục – Index

Các trang Web sau khi thu thập về sẽ được phân tích, trích chọn những thông tin cần thiết để lưu trữ trong cơ sở dữ liệu nhằm phục vụ cho nhu cầu tìm kiếm sau này. Ba bước cơ bản trong quá trình lập chỉ mục:



Hình 1.5 Quá trình đánh chỉ mục

Lập chỉ mục là quá trình phân tích tài liệu cần đánh chỉ mục thành các từ, cụm từ thích hợp quan trọng có khả năng phản ánh đúng nội dung của tài liệu. Bởi vậy cần phải có giải pháp để chiết lọc thông tin một cách chính xác và phải tự động vì lượng tài liệu là vô cùng lớn không thể làm thủ công. Thông tin được chiết lọc phải đủ nội dung để trả về kết quả tìm kiếm cho người dùng và cũng không được dư thừa gây tốn kém dung lượng lưu trữ cũng như trả về kết quả không chính xác cho người dùng. Quá trình phân tích tài liệu được chia thành các bước nhỏ hơn:



Hình 1.6 Các bước phân tích tài liệu

Bước thứ nhất chuyển các dạng tài liệu thành text hay còn gọi là plaintext tạm dịch là văn bản thuần túy. Vì tài liệu ngày nay tồn tại ở nhiều định dạng khác nhau như pdf, doc, xml, rtf, html, v.v...Không thể phân tích ngay trên các định dạng này được mà cần phải chuyển tất cả thành văn bản thuần túy sau đó mới tiếp tục các bước phân tích tiếp theo.

Bước thứ hai chuyển văn bản thành đơn vị từ vựng (Tokenization). Tức là tách văn bản thành các đơn vị từ. Trong Tiếng Anh thì cần dựa vào các dấu khoảng trắng ta có thể phân tích văn bản thành các token. Còn trong Tiếng Việt thì khó hơn. Ví dụ như ta có câu *Thầy giáo này rất gương mẫu* nếu dựa vào khoảng trắng ta phân tích sẽ thành 6 token sau*: thầy, giáo, này, rất, gương, mẫu* như thế sẽ không thích hợp mà kết quả ta cần phải được là 4 token *thầy giáo, này, rất, gương mẫu.*

Bước thứ hai là loại bỏ các stopword. Stopwords là các từ mà nó xuất hiện trong văn bản nhưng không mang nghĩa chẳng hạn trong Tiếng Việt các stopword như: thì, là, và, nhưng v.v...còn trong Tiếng Anh các stopwords như: a, the, but, and v.v...Các stopword thì không cần thiết trong việc đánh chỉ mục cho tài liệu, nó

không phản ánh nội dung của văn bản và hầu như trong văn bản nào cũng xuất hiện chúng vì thế chúng cần phải được loại bỏ.

Bước thứ ba là loại bỏ hậu tố. Phần này trong Tiếng Việt không có vì Tiếng Việt là ngôn ngữ đơn thể. Trong Tiếng Anh một từ ngoài từ gốc còn có thể có tiền tố và hậu tố. Tiền tố có làm thay đổi nghĩa của từ nên được giữ nguyên. Hậu tố chỉ thay đổi từ loại của từ mà không thay đổi nghĩa nên cần phải được đưa về thành từ gốc.

Bước cuối cùng là tính trọng số của các từ có trong tài liệu. Trong văn bản thì có nhiều từ sẽ xuất hiện thường xuyên, các từ đó phản ảnh nội dung của tài liệu. Cần phải tính trọng số của các từ để loại bỏ các từ mang trọng số thấp.[1]

1.5 Bộ tìm kiếm thông tin – Search Engine

Tìm sách trong thư viện là một ví dụ điển hình cho tầm quan trọng của công việc tìm kiếm. Khi ta cần tìm một quyển sách, ta vào thư viện tìm trong danh sách chỉ mục để tìm ra vị trí của sách. Nếu không có chỉ mục, ta không thể tìm ra vị trí của sách trong một thư viện lớn hay có tìm ra thì tốn thời gian rất nhiều. Tương tự như vậy đối với tài liệu trên Internet, ta cần phải lập một kho chỉ mục trước khi tìm kiếm. Ứng dụng web là bước cuối dùng trong xây dựng một máy tìm kiếm. Nhiệm vụ của ứng dụng web là tạo giao diện giao tiếp giữa người dùng và máy tìm kiếm. Nhận câu truy vấn từ người dùng, sau đó trả về kết quả truy vấn cho người dùng.

1.5.1 Tìm kiếm theo từ khóa

Tìm kiếm theo từ khóa là cách tìm kiếm mà dựa trên các từ được cho là quan trọng nhất trong tài liệu. Như đã đề cập các từ có trọng số cao nhất trong tài liệu là các từ xuất hiện thường xuyên và có khả năng phản ảnh một phần nội dung mà tài liệu đề cập tới. Ví dụ các từ khóa, cụm từ khóa hữu dụng trong chủ để máy tìm kiếm như "tìm kiếm", "công cụ tìm kiếm", "phương pháp tìm kiếm", "thuật toán tìm kiếm", "xếp hạng" "độ tương đồng", "kết quả tìm kiếm", v.v… Những từ khóa, cụm từ khóa trên phản ảnh được nội dung của chủ đề này. Các trang web ngày nay

thường liệt kê tất cả các từ khóa của website mình để tăng tính chính xác khi các máy tìm kiếm đánh chỉ mục cho website cũng như phục vụ cho công việc tìm kiếm của người dùng dễ dàng hơn. Trong tài liệu tác giả cũng thường định nghĩa các từ khóa cho tài liệu mình nhằm phục vụ cho máy tìm kiếm đánh chỉ mục dễ dàng và chính xác hơn. Trong các trang html thì các từ khóa được định nghĩa trong thẻ <Meta…/> ở đầu trang trong phần <head> </head>. Ví dụ: <Meta name=”keywords” content=”…”/> Các từ khóa được liệt kê trong thuộc tính content. Ngoại trừ những tài liệu trên, các từ khóa hoàn toàn phụ thuộc vào cách phân tích tài liệu của máy tìm kiếm. Chẳng hạn như các máy tìm kiếm thường để ý đến tựa đề của trang web, nơi mà sẽ chứa nội dung liên quan đến chủ đề mà trang web muốn đề cập đến nhiều nhất. Hay các máy tìm kiếm cũng thường phân tích các câu đầu của một tài liệu, đây là các câu mang nội dung trọng tâm mà tài liệu muốn nói đến. Các câu này được xem như là câu chính trong một đoạn văn mang nội dung cốt lõi. Ngoài ra các máy tìm kiếm còn phân tích các từ được lặp đi lặp lại nhiều lần trong tài liệu. Các từ đó cũng phản ảnh nội dung của tài liệu.

Các khó khăn của chiến lược tìm kiếm theo từ khóa là làm sao phải xử lý các từ đồng âm khác nghĩa. Vì các từ đồng âm khác nghĩa mà các máy tìm kiếm có khi sẽ trả về kết quả chẳng liên quan gì đến mục đích của ta. Khó khăn thứ hai là mà các máy tìm kiếm phải giải quyết là vấn đề đưa các từ về thành từ gốc (*stemming, vấn đề này chỉ có trong Tiếng Anh*). Chẳng hạng khi ta nhập vào từ “big” thì kết quả trả về có thể là “bigger”. Một khó khăn nữa là máy tìm kiếm chưa trả về được các kết quả theo từ đồng nghĩa. Chẳng hạn khi ta truy vấn với từ “giáo viên” thì kết quả trả về sẽ không trả về các tài liệu có các từ “thầy giáo” hay “cô giáo”.[1]

1.5.2 Tìm theo ngữ nghĩa

Thông tin về tìm kiếm theo ngữ nghĩa đã cũ và hiện nay nó cũng không còn tồn tại. Nhưng thông tin về cách tìm kiếm theo ngữ nghĩa sẽ có giá trị cho các nhà nghiên cứu.

Tìm kiếm theo ngữ nghĩa là cách tìm kiếm mà đoán ý của người sử dụng thông qua câu truy vấn. Cách này phải gom nhóm tài liệu thành chủ đề, dựa vào lý thuyết trí tuệ nhân tạo để phân tích câu truy vấn của người dùng. Chẳng hạn như trong tài liệu có từ trái tim. Có hai lĩnh vực chính mà từ trái tim thuộc đó là y học và tình yêu. Chỉ với từ trái tim thì chưa đủ tri thức để máy tìm kiếm lựa chọn chủ đề để sắp xếp cho tài liệu này. Máy tìm kiếm dựa vào các từ đi cùng từ trái tim chẳng hạn như nhịp đập, huyết áp v.v...thì có thể biết tài liệu hướng đến chủ đề y học. Còn từ trái tim đi với các từ như thổn thức, tan vỡ, ngọt ngào v.v...thì máy tìm kiếm có thể đoán nhận người dùng đang hướng đến chủ đề tình yêu.[1]

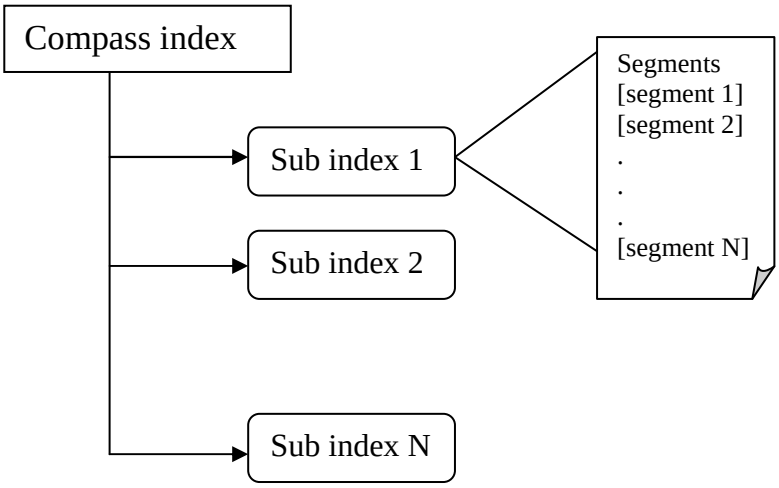
1.6 Cấu trúc lưu trữ dữ liệu index files

Bộ crawler căn cứ vào các liên kết, chúng tải tất cả các thông tin mà chúng bắt gặp về máy, một khối thông tin hỗn độn không tuân theo bất kỳ một quy tắc nào.

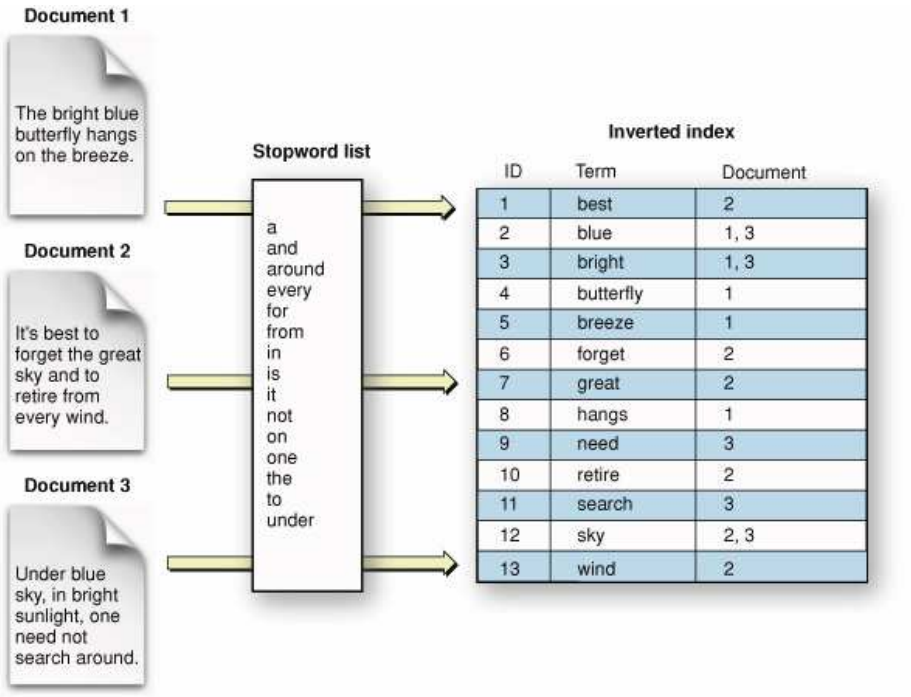
Như đã phân tích tại mục 1.4, bộ indexer có nhiệm vụ trích lọc các thông tin của bộ clawler tải về thành các đơn vị từ vựng, chúng tiếp tục loại bỏ các stopword, loại bỏ hậu tố và các từ có trọng số thấp. Cuối cùng chúng lưu các đơn vị từ vựng vào hệ thống index file.

Một bộ index được chia thành nhiều sub index. Mỗi sub index có cấu trúc và chức năng chỉ mục riêng của nó.

Mỗi Sub index được chia ra nhiều segments, các segments có thể là một hoặc nhiều tập tin. Một segment được xem như một chức năng chỉ mục đầy đủ chứa dữ liệu inverted index.



Hình 1.7 Cấu trúc lưu trữ files index [12]



Hình 1.8 Cấu trúc dữ liệu inverted index [11]

1.7 Kết luận

Qua quá trình tìm hiểu cấu trúc và mô hình hoạt động của máy tìm kiếm cho ta thấy việc truy vấn của người dùng chỉ thực hiện trên files index, như vậy việc tổ

chức lưu trữ hệ thống files index ảnh hưởng trực tiếp đến số lượng, chất lượng và thời gian trả về của kết quả truy vấn.

- Nếu hệ thống index file đặt tập trung tại một server

  + Việc lưu trữ hệ thống index file khổng lồ là một vấn đề không thể thực hiện được.

  + Số lượng người dùng lớn cùng truy vấn vào một segment của hệ thống index files dẫn đến việc truy xuất thông tin trở nên quá tải một server không thể đáp ứng được.

  + Việc xử lý một lượng thông tin quá lớn củng là vấn đề không thể triển khai được

  ➔Máy tìm kiếm triển khai trên hệ tập trung chỉ có thể dùng để nghiên cứu, nó không triển khai ứng dụng ra công chúng được.

- Nếu hệ thống index file phân tán tại nhiều server khác nhau

  + Phân tải ra nhiều server tránh được việc truy cập tập trung.

  + Tăng tốc độ xử lý cho máy tìm kiếm (*vì tất cả các server trong hệ thống đều làm việc phục vụ cho máy SE*)

  ➔Phân tán hệ thống index files sẽ giảm được thời gian xử lý cho máy SE

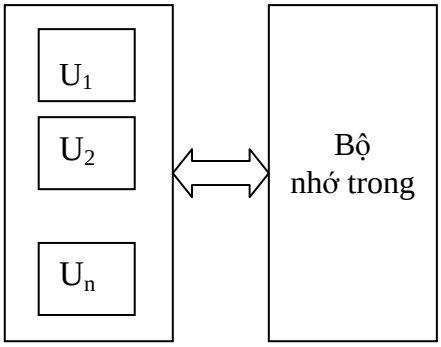
CHƯƠNG 2: HỆ PHÂN TÁN CHO MÁY TÌM KIẾM

2.1 Định nghĩa và các tính chất hệ phân tán

2.1.1 Định nghĩa

- *Hệ tập trung:*

Tiêu biểu là hệ thống máy đơn, là máy không kết nối vật lý và logic với các máy khác như hình vẽ sau:



Hình 2.1 Hệ thống máy đơn

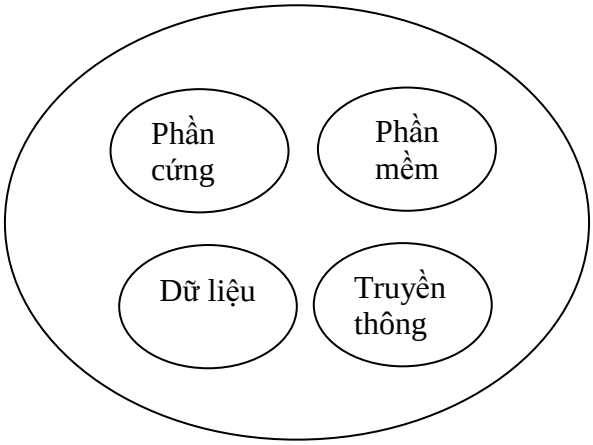
Ở một thời điểm nhất định, máy đơn được điều hành bởi một hệ điều hành duy nhất. Hệ thống như vậy được gọi là hệ tin học tập trung, thích hợp với các máy tính loại trung và loại lớn.

Tóm lại, hệ tin học tập trung bao gồm một hệ thống máy đơn được điều khiển bởi một hệ điều hành duy nhất và quản lý toàn bộ thông tin trên thiết bị nhớ cục bộ của mình.

- *Hệ phân tán:*

Hệ tin học phân tán *(Distributed System)* là hệ thống không chia sẻ bộ nhớ và đồng hồ, khác với xu hướng phân tán các tính toán trên nhiều bộ xử lý của hệ thống đa xử lý. Như vậy, hệ tin học phân tán đòi hỏi hệ thống phần cứng của mình phải trang bị bộ nhớ cục bộ, các bộ xử lý trao đổi thông tin với nhau thông qua các hệ thống đường truyền như cáp chuyên dụng, đường điện thoại, cáp quang. . .

Như vậy, hệ tin học phân tán có thể bao gồm bốn thực thể như sau:



Hình 2.2 Các thực thể của hệ phân tán

Một tư tưởng lớn của hệ tin học phân tán là phân tán hoá các quá trình xử lý thông tin và thực hiện công việc đó trên các trạm khác nhau. Đó là cơ sở căn bản cho việc xây dựng các ứng dụng lớn như thương mại điện tử, giáo dục điện tử, chính phủ điện tử, thư viện điện tử, . . .

Hiện nay, đứng trên những phương diện khác nhau, có thể có các định nghĩa khác nhau về hệ tin học phân tán, nhưng phổ biến hơn cả là định nghĩa sau:

Hệ tin học phân tán (hệ phân tán) là hệ thống xử lý thông tin bao gồm nhiều bộ xử lý hay vi xử lý nằm tại các vị trí khác nhau và được liên kết với nhau thông qua phương tiện viễn thông dưới sự điều khiển thống nhất của một hệ điều hành[2].

Từ định nghĩa trên, hệ phân tán có các ưu điểm căn bản so với hệ tập trung, như sau:

- Tăng tốc độ bình quân trong tính toán, xử lý.
- Cải thiện tình trạng luôn sẵn sàng của các loại tài nguyên.
- Tăng độ an toàn cho dữ liệu.
- Đa dạng hoá các loại hình dịch vụ tin học.
- Đảm báo tính toàn vẹn của thông tin.

2.1.2 Tính chất

Thông thường hệ phân tán bao gồm các đặc tính cơ bản sau đây:

2.1.2.1Tính chia sẽ tài nguyên (Resource Sharing):

Các tài nguyên có thể chia sẻ trong hệ phân tán là: Tài nguyên vật lý, tài nguyên logic.

Đối với hệ phân tán, bài toán chia sẻ tài nguyên phức tạp hơn vì bản thân các máy tính tự trị cũng có tài nguyên riêng của nó. Tuy nhiên, ta sẽ không đề cập đến vấn đề chia sẻ các tài nguyên dùng riêng này mà chủ yếu đi vào chia sẻ tài nguyên dùng chung.

Cụ thể hơn, bài toán chia sẻ tài nguyên trong hệ phân tán có hai đối tượng chính:

- Tập các tài nguyên dùng chung: Các tài nguyên này là phân tán, chúng là hữu hạn và có khả năng bổ sung được (*ví dụ như hiệu năng của CPU*). Tuy nhiên, chúng ta không thể bổ sung các tài nguyên này một cách tùy ý, tùy tiện được.
- Tập các người sử dụng (*user*): Tập này có đặc điểm là phân tán, hữu hạn và có tốc độ tăng trưởng rất nhanh.

Từ đó, chúng ta dễ dàng nhận thấy, lượng tài nguyên trong hệ phân tán thì hữu hạn mà số lượng người sử dụng lại càng nhiều. Điều đó dẫn tới vấn đề chia sẻ tài nguyên ngày càng trở nên căng thẳng, nó có thể gây ra xung đột, tắc nghẽn trong hệ phân tán. Và hệ phân tán sẽ phải đảm bảo làm thế nào để việc

chia sẻ tài nguyên trở nênhiệu quả nhất. Như vậy, một bài toán chia sẻ tài nguyên cần đảm bảo giải quyết các yêu cầu sau:

- Mức một: Tránh các hiện tượng xấu (tắc nghẽn...) xảy ra
- Mức hai: Đảm bảo việc chia sẻ tài nguyên một cách hiệu quả.

2.1.2.2Tính mở (Openness):

Thông thường, một hệ thống nào đó thường cung cấp nhiều dịch vụ khác nhau cho các đối tượng người dùng khác nhau. Đối với hệ phân tán, nó được gọi là có tính mở nếu như khi ta bổ sung một dịch vụ mới, thì dịch vụ này có khả năng chung sống bình thường với các dịch vụ trước đó. Và bài toán chia sẻ tài nguyên vẫn được giải quyế tmột cách hợp lý.

Một số đặc trưng:

- Interoperability: Các thành phần khác nhau trên các hệ thống khác nhau có thể cùng tồn tại và làm việc với nhau.
- Portability: Các ứng dụng được triển khai trên hệ phân tán A cũng có thể được thực thi mà ko cần chỉnh sửa trên một hệ phân tán B khác có cùng giao diện như hệ phân tán A.
- Extensible: Dễ dàng thêm các thành phần mới, thay thế các thành phần cũ mà ko hề ảnh hưởng đến phần còn lại của hệ phân tán.

2.1.2.3Tính đồng thời (Concurrency):

Theo định nghĩa thì hệ phân tán bao gồm nhiều bộ xử lý hoặc vi xử lý nằm tại nhiều trí khác nhau được liên kết với nhau thông qua đường truyền viễn thông, do đó các tiến trình xử lý được thực hiện một cách độc lập và đồng thời. Điều này giúp cho làm tăng tốc độ xử lý công việc lên đáng kể.

Việc xử lý đồng thời diễn ra khi một yêu cầu (*Request*) của người dùng (*user*) được gửi đến hệ yêu cầu xử lý một vấn đề cụ thể. Ngay lập tức, hệ phân tán request người dùng đến xử lý tại các bộ xử lý riêng biệt trong hệ và kết quả

(*result*) được tập hợp từ tất cả các kết quả của từng bộ xử lý riêng biệt trong hệ.

2.1.2.4Tính co giãn (Scalability):

Tính co dãn là một tính chất quan trọng trong hệ phân tán. Một hệ thống được gọi là có thể co dãn (*scalable*) nếu nó có thể kiểm soát được việc gia tăng của tài nguyên, cũng như người sử dụng mà không làm ảnh hưởng tới hiệu năng, cũng như làm tăng độ phức tạp của hệ thống. Hay nói cách khác, điều này liên quan đến việc xây dựng mộthệ phân tán sao cho trong bất cứ trường hợp thay đổi nào, thì phần mềm hệ phân tán của chúng ta phải có khả năng đáp ứng được.

Thông thường, việc co dãn thường liên quan đến sự gia tăng kích thước của hệ phân tán theo ba khía cạnh chính sau:

- Size: Số lượng người sử dụng và số lượng tài nguyên, ứng với sự gia tăng này, hệ thống có thể đối mặt với nguy cơ bị quá tải (*do nó phải xử lý nhiều user request hơn*). Tương tự như vậy, do hệ thống phải quản lý một số lượng tài nguyên lớn hơn, nó cũng có thể bị quá tải.
- Geography: Hệ phân tán cũng có thể phát triển theo khoảng cách địa lý, đó là khoảng cách vật lý thực tế giữa những người sử dụng, những tài nguyên với nhau. Vấn đề liên quan đến sự mở rộng về địa lý, chính là vấn đề về truyền thông (*khoảng cách càng xa thì độ trễ truyền thông càng lớn, và nguy cơ xảy ra lỗi cũng càng cao*)
- Administration: Hệ phân tán được phát triển và mở rộng theo nhiều lĩnh vực hành chính khác nhau. Vấn đề liên quan đến việc mở rộng này, đó là sự xung đột các chính sách giữa các tổ chức trong quá trình đảm bảo việc sử dụng, quản lý và bảo vệ tài nguyên một cách đúng đắn.

2.1.2.5Tính chịu lỗi (Fault Tolerance):

Hệ bao gồm nhiều bộ xử lý kết nối với nhau. Nếu có một bộ xử lý trong hệ phát sinh lỗi, ngay lập tức công việc sẽ được chuyển cho các bộ xử lý khác trong hệ sẽ đảm nhận, đảm bảo hệ thống luôn hoạt động bình thường.

Thông thường hệ thống phân tán sẽ phát sinh các lỗi sau đây:

- Lỗi sụp đổ (*crash failure*): khi server gặp lỗi này thì nó sẽ bị treo, trước đó server vẫn hoạt động tốt cho đến khi ngừng hoạt động. Khi server gặp lỗi này, nó sẽ không thể làm gì được nữa. Một ví dụ hay gặp lỗi này là hệ điều hành của các máy cá nhân. Khi hệ điều hành ngừng hoạt động thì chỉ còn cách duy nhất là khởi động lại.
- Lỗi bỏ sót (*omission failure*): là lỗi mà một server không thể đáp ứng được yêu cầu gửi tới nó. Người ta chia nó thành hai loại:
  + Lỗi khi nhận thông điệp gửi tới: gặp lỗi này, server không nhận được yêu cầu ngay cả từ client gần nó nhất và mặc dù kết nối giữa server với client đã được thiết lập. Lỗi khi nhận thông điệp chỉ làm cho server không nhận biết được các thông điệp gửi tới nó mà không hề ảnh hưởng đến trạng thái của server.
  + Lỗi khi gửi thông điệp: server vẫn nhận được các yêu cầu, vẫn hoàn thành yêu cầu đó nhưng vì một lý do nào đó lại không thể gửi kết quả tới máy đã yêu cầu. Một trong những lý do thường gặp là do bộ nhớ đệm gửi đầy. Trong trường hợp gặp lỗi này, server cần chuẩn bị tình huống clien sẽ gửi lại yêu cầu đã gửi đó .
- Lỗi thời gian (*timing failure*): là lỗi xảy ra khi server phản ứng lại quá chậm, sau cả thời gian cho phép. Trong một hệ thống luôn có các ràng buộc về mặt thời gian. Nếu bên gửi gửi đến bên nhận nhanh quá, bộ nhớ đệm của bên nhận không đủ để chứa thì sẽ gây ra lỗi. Tương tự, server phản ứng lại chậm quá, vượt quá khoảng timeout quy định sẵn cũng sẽ gây ra lỗi, ảnh hưởng đến hiệu năng chung của hệ thống.

- Lỗi đáp ứng (*Response failure*): là lỗi khi server trả lời không đúng. Đây là một kiểu lỗi rất ngiêm trọng và được phân chia thành hai loại:

  + Lỗi về mặt giá trị: là lỗi khi server trả lời lại yêu cầu của client với giá trị không chính xác. Ví dụ khi sử dụng các máy tìm kiếm, kết quả trả về không hề liên quan gì tới yêu cầu của người sử dụng.

  + Lỗi về chuyển trạng thái: là lỗi khi server hoạt động trệch hướng khỏi luồng điều khiển. Có nghĩa là server trả lời các yêu cầu được gửi tới một cách không theo như mong đợi.

- Lỗi bất kì (*Arbitrary failure*): một server có thể tạo ra một lỗi bất kì ở bất kì thời gian nào. Đây là loại lỗi nguy hiểm nhất. Có thể có hai khả năng xảy ra:

  + Thứ nhất: một server tạo ra một kết quả sai mà không thể phát hiện ra được.

  + Thứ hai: server bị lỗi có liên kết với các server khác tạo ra một kết quả sai.

2.1.2.6 Tính trong suốt (Transparency):

Hệ phân tán dù có hoàn hảo, tốt đẹp đến bao nhiêu thì bản chất của nó vẫn là rời rạc, và người thiết kể phải làm thế nào để che giấu, làm giảm ảnh hưởng, khiếm khuyết của hệ phân tán đối với người sử dụng.

Như vậy, tính trong suốt được hiểu là sự che giấu sự phân tách, rời rạc của các thành phân trong hệ phân tán đối với người sử dụng. Qua đó, người sử dụng sẽ coi hệ thống như là một hệ thống thống nhất. Tính trong suốt là mộttính chất rất mạnh, và cũng rất khó để đạt được. Thông thường, có một vài dạng tính trong suốt chính sau:

- Access Transparency: Tài nguyên toàn cục và cục bộ cùng được truy cập theo cách giống hệt nhau.
- Location Transparency: Người sử dụng sẽ không nhận biết được vị trí vật lý thực tế của tài nguyên mà họ đang dùng

- Migration Transparency : Tài nguyên có thể được di chuyển từ nơi này sang nơi khác mà người sử dụng không nhận ra sự thay đổi
- Replication Transparency: Người sử dụng không nhận ra sự tồn tại của nhiều bản sao của tài nguyên trong hệ thống Failure Transparency: Người sử dụng không nhận ra hệ thống mà họ đang sử dụng đang có lỗi ở một thành phần nào đó.
- Concurrency Transparency: Người sử dụng không hề biết họ đang chia sẻ tài nguyên dùng chung với rất nhiều người sử dụng khác.

2.2 Truyền thông trong hệ phân tán

Truyền thông làm một yếu tố tối quan trọng trong hệ phân tán. Sẽ là vô ích khi chúng ta tìm hiểu về hệ phân tán mà lại không quan tâm đến cách thức các tiến trình trong các máy khác nhau trao đổi thông tin. Các thành phần của một hệ phân tán có thể phân chia thành 2 nhóm: nhóm các đối tượng vật lý và nhóm các đối tượng logic, để có thể tương tác lẫn nhau, các thành phần này phải được nối kết với nhau thông qua các kênh truyền thông. Hệ thống phân tán và các ứng dụng bao gồm các thành phần phần mềm tương tác với nhau để thực hiện các tác vụ. Các thành phần yêu cầu hoặc cung cấp sự truy cập tài nguyên trong hệ phân tán thì được thực hiện như những trạm. Trong hệ thống client/server, một trạm client phải tương tác với một máy server khi có nhu cầu truy cập đến một tài nguyên không thuộc quyền quản lý của nó (*có thể là phần cứng, phần mềm hay dữ liệu*).

Truyền thông giữa hai trạm bao gồm hai thao tác gửi và nhận các gói tin nhằm đạt được hai kết quả:

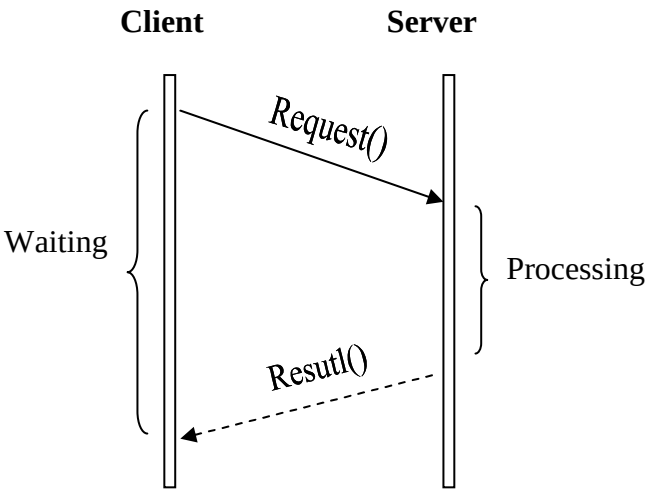
- Truyền dữ liệu từ nơi phát đến nơi thu.
- Trong một vài thao tác truyền thông, có sự đồng bộ giữa thu và phát, quá trình thu hoặc phát sẽ được tiếp diễn cho đến khi có sự can thiệp của một tiến trình khác nhằm giải phóng nó.

Để điều thứ nhất được thực hiện, các tiến trình phải chia sẻ kênh truyền thông để dữ liệu có thể truyền dẫn giữa điểm phát và thu.

Điều thứ hai là một ngầm định trong việc lập trình truyền thông nguyên mẫu.

Khuôn dạng của gói tin gửi và nhận được quy định trong lập trình, nhứng khuôn dạng này tạo ra các thông điệp (*message*) hành động giữa các trạm thu và phát. Mối thông điệp bao gồm một tập các đơn vị dữ liệu được trạm phát gủi thông qua các cơ cấu truyền thông (*cổng, kênh truyền*) để đến trạm thu. Các cơ chế truyền thông có thể là đồng bộ hoặc blocking, tức là phía gửi sau khi gửi một thông điệp sẽ chờ cho đến khi phía thu nhận được và thực hiện thao tác trả lời, hay cũng có thể là không đồng bộ, tức là các thông điệp được xếp vào một hàng đợi và chờ đến khi phía thu đồng ý nhận thì lập tức được gửi đi. Thông thường người ta sử dụng bốn mô hình phổ biến để truyền thông tin, đó là mô hình client-server, mô hình RPC (*Remote Procedure Call*), mô hình MOM (*Message – Oriented Middleware*) và mô hình SOM (*Stream – Oriented Middleware*).

2.2.1 Mô hình client – server



Hình 2.3 Mô hình Client – Server

Một trao đổi trong mô hình khách chủ bao gồm ba bước:

Bước 1: Client gửi một yêu cầu đến Server.

Bước 2: Server thực hiện yêu cầu

Bước 3: Server gửi thông điệp trả lời đến Client.

Mỗi phiên giao dịch như vậy cần phải thực hiện một gửi một thông điệp, nhận thông điệp và đòi hỏi phải có sự đồng bộ giữa Client và Server. Khi client gửi một thông điệp yêu cầu đến server, server tiếp nhận và xử lý thông điệp, sau đó gởi kết quả lại cho client. Trong khi server xử lý kết quả client phải ngưng tất cả các giao dịch và chờ cho đến khi nhận được kêt quả từ server mới thực hiện các giao dịch khác.

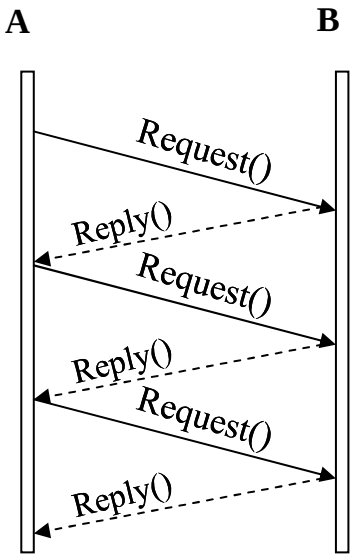
2.2.2 Mô hình RPC(Remote Procedure Call: gọi thủ tục từ xa)

Thực hiện chủ yếu theo mô hình Client/Server. Khi một tiến trình trên máy client (A) gọi một thủ tục trên máy server (B), tiến trình gọi trên A sẽ bị treo, và quá trình thực thi thủ tục được gọi diễn ra trên máy B. Thông tin được truyền từ phía gọi A sang phía nhận B trong các tham số, và được trả về dưới dạng kết quả của thủ tục. Phương pháp này được gọi là lời gọi thủ tục từ xa (*RPC – Remote Procedure call*). Trong khi ý tướng của phương pháp là đơn giản như vậy, thì trong thực tế, vẫn còn tồn tại nhiều vấn đề liên quan đến nó.

- Thứ nhất, do phía gọi và phía bị gọi nằm ở hai máy tính khác nhau, nên chúng sẽ được thực thi trên những không gian địa chỉ khác nhau, và điều này gây ra sự phức tạp không nhỏ.
- Thứ hai, tham số và kết quả trả về cũng phải được truyền đi, điều này cũng gây ra sự phức tạp nếu như hai máy là không tương tự nhau.
- Cuối cùng, trong quá trình thực hiện, có thể một trong hai máy xảy ra lỗi, và lỗi này có thể gây ra một số vấn đề khác nữa. Tuy nhiên, các khó khăn này đều đã được khắc phục phần nào, và RPC vẫn đang là một trong những kĩ thuật phổ biến được sử dụng bởi nhiều hệ phân tán.

Người ta phân RPC ra làm hai loại : Synchronous RPC (*RPC đồng bộ*) và Asynchronos RPC (*RPC Không đồng bộ*).

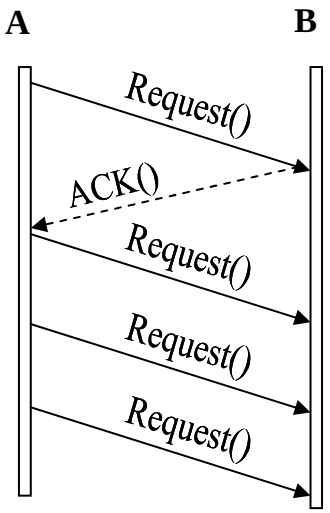
Synchronous RPC:



Hình 2.4 Mô hình Synchronous RPC

Với truyền thông đồng bộ, sau khi client đưa ra request, nó sẽ phải chờ phản hồi từ phía server. Trong thời gian đó, nó không làm gì cả, và đó cũng chính là một nhược điểm của việc truyền đồng bộ. Điều này sẽ dẫn tới việc chờ đợi vô ích của tiến trình gọi trên client, nhất là khi lời gọi không có kết quả trả về, trong khi nó có thể thực hiện một số công việc hữu ích khác. Một số ví dụ về việc tiến trình gọi không cần phải đợi kết quả trả về, đó là việc thêm một entry vào cơ sở dữ liệu, khởi động một dịch vụ từ xa, xử lý hàng loạt (*batch processing*)…

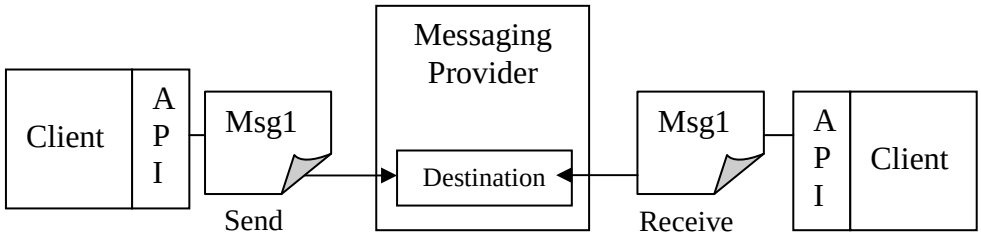
Asynchronos RPC



Hình 2.5 Mô hình Asynchronos RPC

Với truyền thông không đồng bộ, sau khi server nhận được request, nó sẽ gửi một tín hiệu ACK về cho client để báo nhận. Khi client nhận được tín hiệu ACK này, nó có thể thực hiện các công việc khác trong thời gian server xử lý request kia.

2.2.3 Truyền thông điệp (MOM)



Hình 2.6 Mô hình MOM

Trong nhiều trường hợp, RPC tỏ ra không phù hợp, nó đòi hỏi cả bên gửi và bên nhận cùng phải sẵn sàng (*active*) tại thời điểm diễn ra truyền thông. Tuy nhiên, đôi khi chúng ta không thể biết được liệu bên nhận yêu cầu có đang sẵn sàng (*active*) hay không. Chúng ta có thể giải quyết vấn đề này thông qua hệ thống messages hàng đợi, hoặc Messages – Oriented Middleware (*MOM*). Hệ thống

Messages hàng đợi cung cấp hỗ trợ rộng rãi cho các giao tiếp không đồng bộ liên tục. Bản chất của các hệ thống này là chúng cung cấp dung lượng lưu trữ cho, mà không đòi hỏi một trong hai người gửi hoặc nhận được hoạt động trong quá trình truyền messages.

2.2.4 Truyền thông hướng dòng (SOM)

Với các hình thức truyền thông MOM, chúng ta chỉ quan tâm đến việc trao đổi thông tin, mà không hề quan tâm cụ thể rằng việc trao đổi thông tin sẽ diễn ra tại một thời điểm cụ thể nào và cũng không hề quan tâm đến việc thời gian truyền là nhanh hay chậm. Hay nói khác đi, trong các mô hình truyền thông bên trên, thời gian không hề ảnh hưởng tới tính đúng đắn của việc truyền tin.

Với hình thức truyền thông hướng dòng (*Stream - Oriented Middleware*), thời gian đóng một vai trò cự kì quan trọng. Ví dụ, nếu một đoạn âm thanh được phát lại với tần số khác với tần số lấy mẫu của đoạn âm thanh ban đầu, nó có thể cho ra kết quả khácvới đoạn âm thanh gốc. Tương tự như vậy, nếu chúng ta muốn xem một bộ phim trực tuyến trên mạng, nếu thời gian truyền của dữ liệu là quá chậm, bộ phim có thể bị gián đoạn. Như vậy, hình thức truyền thông này được cung cấp nhằm tạo điều kiện thuận lợi cho việc trao đổi các thông tin phụ thuộc vào thời gian, tức là, thời gian sẽ ảnh hưởng đến tính đúng đắn của thông tin được trao đổi.

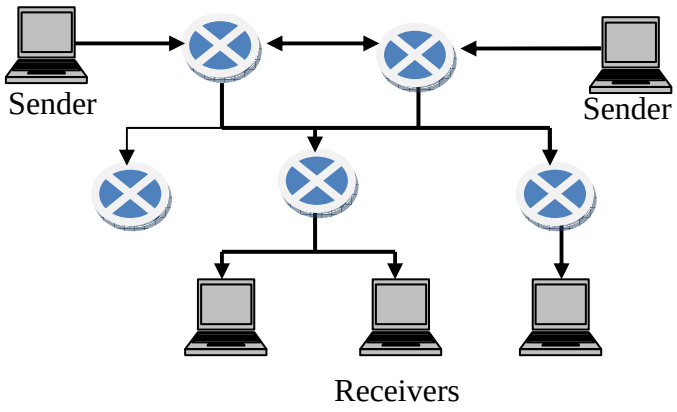
2.2.5 Truyền thông đa điểm (MultiCast)

Truyền thông đa điểm chính là khả năng gửi dữ liệu từ một điểm tới nhiều điểm nhận. Với mô hình này, chúng ta có hai trường hợp:

- Từ một điểm truyền đến nhiều điểm: Từ một điểm, chúng ta sẽ truyền đồng thời đến một nhóm các điểm thỏa mãn một tiêu chí nào đó.
- Từ một điểm truyền đến tất cả các điểm: broadcast (*quảng bá*). Từ một điểm, chúng ta sẽ truyền đến tất cả các điểm còn lại có khả năng tiếp nhận, điều này là rất có ích và nó được thực hiện khá nhiều trong bối cảnh hệ phân tán. Ví dụ, nếu chúng ta có một tiến trình, nếu ta muốn đưa ra một thống nhất, một chu trình

nào đó mà phải tham khảo tất cả các tiến trình còn lại. Trong trường hợp này chúng ta sẽ sử dụng broadcast.

- Từ nhiều điểm truyền đến nhiều điểm: bao gồm tương tác học tập từ xa, chơi game nhiều người, hội nghị đa phương tiện, chat nhóm, chỉnh sửa và chia sẻ các công cụ cộng tác, tính toán song song, cũng như phân phối mô phỏng tương tác.



Hình 2.7 Mô hình multicast many-to-many

2.3 Đồng bộ hóa tiến trình

2.3.1 Đặt vấn đề

+ Nhìn chung, các tiến trình kể các tiến trình xuất phát từ các ứng dụng độc lập muốn truy cập vào các tài nguyên với số lượng vốn rất hạn chế hay truy cập vào thông tin dùng chung cùng một lúc. Trường hợp này gọi là truy cập tương tranh.

Vì vậy, tương tranh là nguyên nhân chính của các xung đột giữa các tiến trình muốn truy cập vào tài nguyên dùng chung đây là một trong những nguyên nhân phải thực hiện cơ chế đồng bộ hoá các tiến trình.

+ Các tiến trình của cùng một hệ ứng dụng hoạt động theo kiểu hợp lực để giải quyết các bài toán đặt ra và cho kết quả nhanh chóng nhất. Điều này cho phép tăng hiệu năng sử dụng thiết bị và hiệu quả hoạt động của chương trình. Đây là một trong những nguyên nhân phải thực hiện cơ chế đồng bộ hoá các tiến trình.

Điều này, được minh họa cụ thể qua bài toán bãi để xe [2, tr 157], bài toán người sản xuất – người tiêu thụ [2, tr 162] và các vấn đề không gắn bó dữ liệu phát sinh khi vận hành hệ thống, làm cho kết quả bị sai lệch mà nguyên nhân chính đó là đồng hồ thời gian giữa các máy trong hệ không đồng bộ với nhau và *độ trễ* của việc truyền tin cũng không nhất quán.

Trong các hệ thống phân tán, việc đồng bộ hoá chỉ đặt ra duy nhất vấn đề thiết lập một trật tự giữa các sự kiện. Giữa các trạm khác nhau, trật tự đó chỉ có thể hiện được thông qua việc trao đổi các thông điệp với nhau.

2.3.2 Các giải pháp đồng bộ tiến trình

2.3.2.1Đồng bộ hóa theo thời gian vật lý

Trong hệ phân tán, mỗi bộ xử lý có một bộ định thời riêng (một đồng hồ riêng). Các bộ định thời trôi theo thời gian thực trong nhịp khác nhau. Nhịp trôi lớn nhất (MDR – Max Drift Rate) của một đồng hồ phụ thuộc vào đặc tính của đồng hồ và môi trường xung quanh. Sự khác nhau lớn nhất của hai đồng hồ với MDR tương tự là: 2*MDR.

$C_i(t)$: là thời gian đọc được từ phần mềm đồng hồ i khi thời gian thực là t.

Đồng bộ hóa ngoài: Cho một giới hạn đồng bộ hóa D>0, và nguồn S của thời gian UTC, $| S(t) – C_i(t) | < D$, với i=1, 2, ... , n và với tất cả thời gian thực t trong I.

Các đồng hồ Ci là chính xác trong giới hạn D.

Đồng bộ hóa trong: cho một giới hạn đồng bộ hóa D>0, $| C_i(t) – C_j(t) | < D$ với i, j=1, 2,...,n và với tất cả thời gian thực t trong I.

Các đồng hồ C_i đồng ý trong ranh giới D.

- **Giải thuật NTP(Network Time Protocol): giải thuật Cristian**

Giả sử trong hệ phân tán có một máy có WWV (*gọi là Time server*) và chúng ta sẽ tiến hành đồng bộ các máy khác với máy này.Trong khoảng thời gian δ/2p mỗi máy sẽ gửi một thông điệp đến máy chủ hỏi thời gian hiện tại.

Máy chủ nhanh sẽ phản hồi bằng một thông điệp mang giá trị thời gian C(utc). Bên gửi nhận được phản hồi nó sẽ thiết lập lại clock thành C(uct). →Máy chủ ở trạng thái bị động (*chờ hỏi*)

Đánh giá: Giải thuật này có 2 vấn đề

- Một là nếu clock bên gửi chạy nhanh thì lúc này C(uct) sẽ nhỏ hơn thời gian hiên tại C của bên gửi…Có thể giải quyết bằng cách thay đổi nhịp ngắt lại nhanh hơn hoặc chậm hơn cho đến lúc khớp nhau.
- Hai là sự chênh lệch từ lúc C(uct) được gửi cho đến lúc nhận được có thể gây lỗi. Giải quyết bằng cách ghi nhận khoản thời gian giữa lúc gửi và nhận

- **Giải thuật Berkeley**

Nếu như trong giải thuật Cristian, server thời gian đóng vai trò bị động, chỉ trả lời yêu cầu của client khi được hỏi thì trong giải thuật Berkeley, nó lại đóng vai trò chủ động: server thời gian (*time deamon*) chủ động hỏi thời gian hiện tại trên các máy khác, tính toán độ chênh lệch so với thời gian hiện tại trên server. Sau đó nó tính trung bình độ chênh lệch về thời gian để điều chỉnh đồng hồ trên máy mình, cũng như gửi đến các máy client yêu cầu chỉnh nhanh hoặc chậm đồng hồ cho phù hợp với máy server.

Ở đây, không cần thiết phải có bất kỳ máy nhận UTC (*Universal Coordinated Time: Giờ phối hợp quốc tế*) nào: thời gian trên máy server được thiết lập bằng các thao tác theo chu kỳ. Một điểm cần lưu ý khác là các máy trong hệ thống chỉ có thời gian giống nhau, chứ không nhất thiết đều phải có thời gian giống như thời gian trong thực tế.

1. Time deamon hỏi các máy khác về giá trị đồng hồ của chúng

2. Các máy này trả lời

3. Time deamon chỉ cho các máy biết cách chỉnh lại đồng hồ cho phù hợp

- **Giải thuật RBS** *(rate- based sequential)*

Giải thuật này khôngyêu cầu trên hệ thống có node chứa thời gian chuẩn (*khi so sánh với thời gian chuẩn thực tế*). Do đó, thay vì hướng tới đồng bộ hóa với thời gian UTC, nó chỉ đồng bộ trong nội bộ hệ thống, như với giải thuật Berkeley. Mặt khác, trong các giải thuật trước đó, chúng ta cố gắng đồng bộ hóa cả phía gửi và phía nhận, nhưng RBS chỉ để phía nhận đồng bộ hóa.

Cụ thể hơn, trong RBS, phía gửi quảng bá một thông điệp tham chiếu, thông điệp này cho phép phía nhận có thể chỉnh đồng hồ của nó. Một đặc điểm quan trọng, đó là trong mạng sensor, thời gian để lan truyền tín hiệu tới các node khác gần như là hằng số, Thời gian lan truyền trong trường hợp này được tính bắt đầu khi thông điệp rời khỏi giao diện mạng (*network interface*) của phía gửi. Chính vì thế, hai nguyên nhân chính gây ra chênh lệnh thời gian đã bị loại bỏ, đó là, khoảng thời gian tạo ra thông điệp và khoảng thời gian tiêu tốn để có thể truy cập vào mạng (*time spent to access the network*)

2.3.2.2Đồng bộ hóa theo thời gian logic

- **Khái niệm tem thời gian:**

Định nghĩa quan hệ “xảy ra trước” happens – before. Kí hiệu : a→b

Một quan hệ được gọi là “xảy ra trước khi nó thỏa mãn hai tình huống:

1. Nếu a và b là các sự kiện trong cùng một tiến trình và a xảy ra trước b thì quan hệ a→b là đúng.

2. Nếu a và b là hai sự kiện thuộc hai tiến trình khác nhau, trong đó a là một sự kiện gửi thông điệp đi à b là sựkiện nhật thông điệp đó thì quan hệ a→b là đúng.

Quan hệ a→b có tính chất bắc cầu, tức là nếu a→b; b→c thì a→c. Trên cơ sở quan hệ xảy ra trước đó, chúng ta định nghĩa khái niệm tem thời gian

Với mỗi sự kiện x, tem thời gian của x, kí hiệu là C(x) được định nghĩa thỏa mãn các điều kiện sau:

1. Nếu a và b là hai sự kiện xảy ra trong cùng một tiến trình và a→b thì C(a) < C(b)
2. Nếu a là sự kiện gửi một thông điệp của một tiến trình nào đó, b là sự kiện nhận thông điệp đó của tiến trình khác thì C(a) < C(b)

Chú ý: Quan hệ này mới chỉ là một chiều, tức là nếu a→b thì C(a) < C(b) nhưng điều ngược lại chưa chắc đãđúng, tức là nếu C(a) < C(b) thì chưa chắc chúng ta có a→b

- **Khái niệm tem thời gian vector**

Trong khái niệm về tem thời gian, nếu sự kiện a xảy ra trước sự kiện b thì C(a) < C(b). Tuy nhiên, chúng ta không thể nói trước điều gì về quan hệ giữa a và b nếu chỉ biết C(a) < C(b), hay nói cách khác, nếu C(a) < C(b), chúng ta sẽ không thể nói rằng a xảy ra trước b. Như vậy, vấn đề ở đây là khái niệm về tem thời gian không có tính chất nhân quả (*causality*). Do đó, khái niệm tem thời gian vector (*vector clock*) đã được đưa ra.

Với khái niệm này, một vector thời gian VC(a) được gán cho một sự kiện a có tính chất sao cho nếu VC(a) <VC(b) thì sự kiện a sẽ xảy ra trước sự kiện b.

- **Giải pháp trật tự từng phần**

Giả sử rằng ta có thể xác định một trật tự giữa các sự kiện của hệ phân tán nhờ vào quan hệ được ký hiệu là → và gọi là "có trước" hay "ở ngay trước". Quan hệ này tối thiểu phải thoã mãn được ràng buộc thể hiện trong bảng sau đây :

C_1 : Nếu A và B là hai sự kiện của cùng một trạm và nếu A được thực hiện trước B thì theo trật tự cục bộ của trạm ta có A → B.

C_2 : Nếu A là phát thông điệp bởi một trạm nào đó và nếu B là thu của thông điệp này thì ta có A → B.

Hình 2.8 sau đây cho ta ví dụ về trật tự hóa từng phần của các sự kiện trong hệ thống.

Theo hình vẽ ta có thể biểu trật tự từng phần như sau:

- Trật tự của các sự kiện

  A1→ A2→ A3→ A4→ A5

  B1→ B2→ B3→ B4

- Trao đổi thông điệp

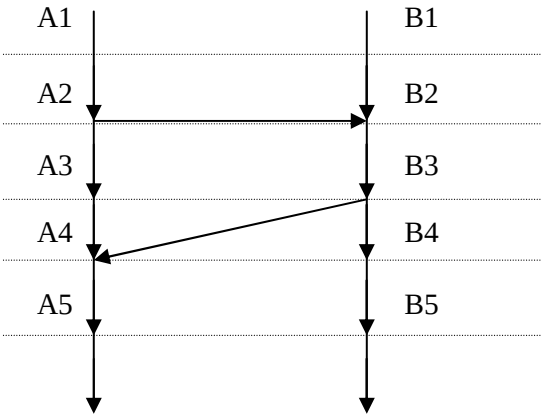
  A1→B2; B3→A4

- Chuyển qua

  A1→A2→B2→B3→B4

  B1→B2→B3→A4→A5

  A1→A2→B2→B3→A4→A5



Hình 2.8 Mô hình trật tự từng phần

- **Thuật toán Lamport - đồng bộ hóa đồng hồ logic:**

Các ký hiệu:

Một chương trình phân tán được tạo thành bởi tập hợp n tiến trình độc lập và không đồng bộ P1, P2,...,Pn. Các tiến trình này không chia xẻ một đồng hồ chung.

Mỗi tiến trình có thể xử lý một sự kiện tự động; khi gửi một thông điệp, nó không phải đợi việc phân phát hoàn thành.

Việc xử lý của mỗi tiến trình Pi, sản sinh ra một dãy sự kiện e_i^0, e_i^1,..., e_i^x, e_i^{x+1},...

Tập hợp các sự kiện được sản sinh ra bởi Pi có một tổng thứ tự được xác định bởi dãy các sự kiện $e_i^x \rightarrow e_i^{x+1}$

Chúng ta nói rằng e_i^x **xảy ra trước** e_i^{x+1}

Quan hệ xảy ra trước → có tính bắc cầu: $e_i^i \rightarrow e_i^j$ với mọi i<j.

Các sự kiện xảy ra giữa hai tiến trình đồng thời nói chung không quan hệ, ngoại trừ hai tiến trình đó có liên quan theo quan hệ như sau:

Đối với mỗi thông điệp m trao đổi giữa hai tiến trình Pi và Pj, chúng ta có e_i^x = send(m), e_j^y = receive(m) và $e_i^x \rightarrow e_j^y$.

Các sự kiện trong một sự xử lý phân tán là được sắp xếp riêng biệt.

- Các sự kiện cục bộ là tổng thể được sắp đặt.
- Các sự kiện nguyên nhân là tổng thể được sắp đặt.
- Tất cả các sự kiện khác là không được sắp đặt.

Cho bất kỳ hai sự kiện e_1 và e_2 trong một sự xử lý phân tán, có thể là:

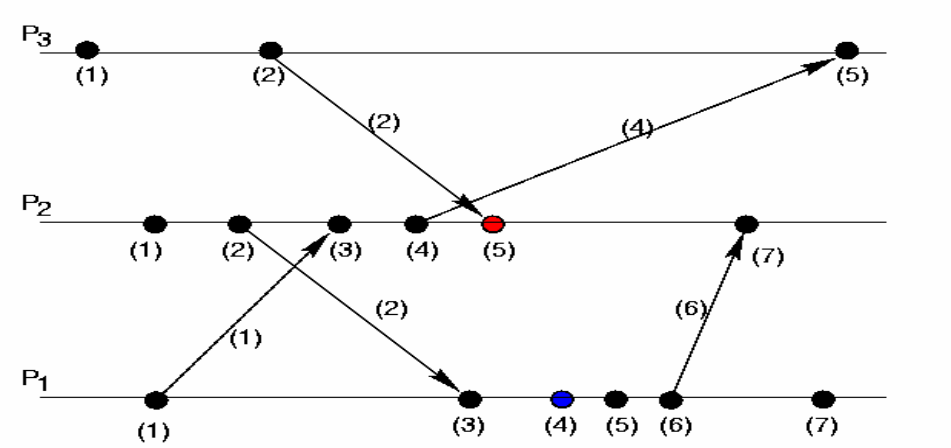
(i) $e_1 \rightarrow e_2$

(ii) $e_2 \rightarrow e_1$

(iii) $e_1 \parallel e_2$ (e_1 và e_2 là đồng thời).

❖ **Ví dụ**:

Những sự kiện nào là quan hệ → ? Những sự kiện nào là đồng thời ?



Hình 2. 9 Thứ tự các sự kiện tại của các tiến trình tại các trạm phát nhận

1. Những điều kiện đồng hồ:

Trong một hệ thống các đồng hồ logic, các tiến trình riêng biệt có một đồng hồ logic mà được áp dụng theo một giao thức.

Mỗi sự kiện được gán một timestamp (thời gian đánh dấu) trong cách thức mà thõa mãn điều kiện bền chặt đồng hồ: nếu $e_1 \rightarrow e_2$ thì $C(e_1) < C(e_2)$.

Trong đó: $C(e_i)$ là timestamp (thời gian đánh dấu) được gán cho sự kiện e_i.

Nếu giao thức thõa mãn các điều kiện theo sau nữa, thì đồng hồ được nói rằng bền chặt mạnh: nếu $C(e_1) < C(e_2)$ thì $e_1 \rightarrow e_2$.

2. Thuật toán Lamport – Tính giá trị đồng hồ logic

R1:Tất cả các máy (tiến trình - Pi) sử dụng một bộ đếm (đồng hồ - Ci) với giá trị khởi tạo là 0.

R2: Trước khi xử lý một sự kiện (gửi, nhận hoặc ngắt), Pi xử lý như sau: tăng bộ đếm và gán cho mỗi sự kiện, như là timestamp (thời gian đánh dấu) của nó.

Ci = Ci + d (d>0, thường d=1)

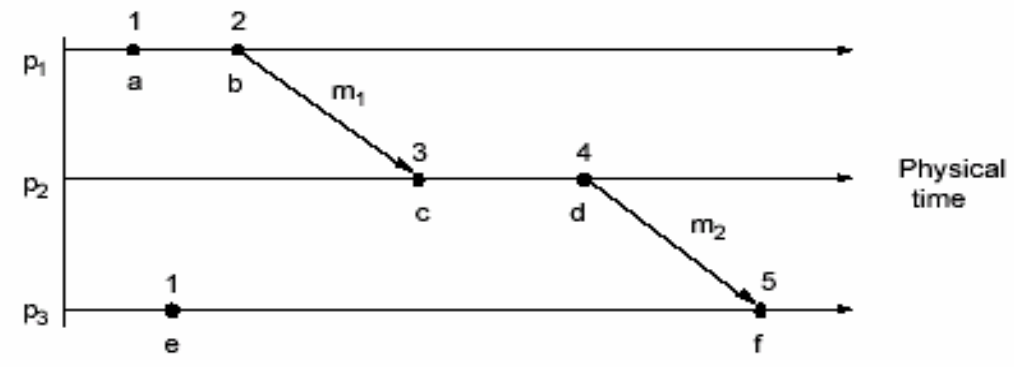
R3: Mỗi thông điệp mang giá trị đồng hồ của người gửi nó tại thời điểm gửi. Khi Pi nhận một thông điệp với timestamp (thời gian đánh dấu) C_{msg}, nó xử lý như sau:

1.Ci = Max(C_i, C_{msg})

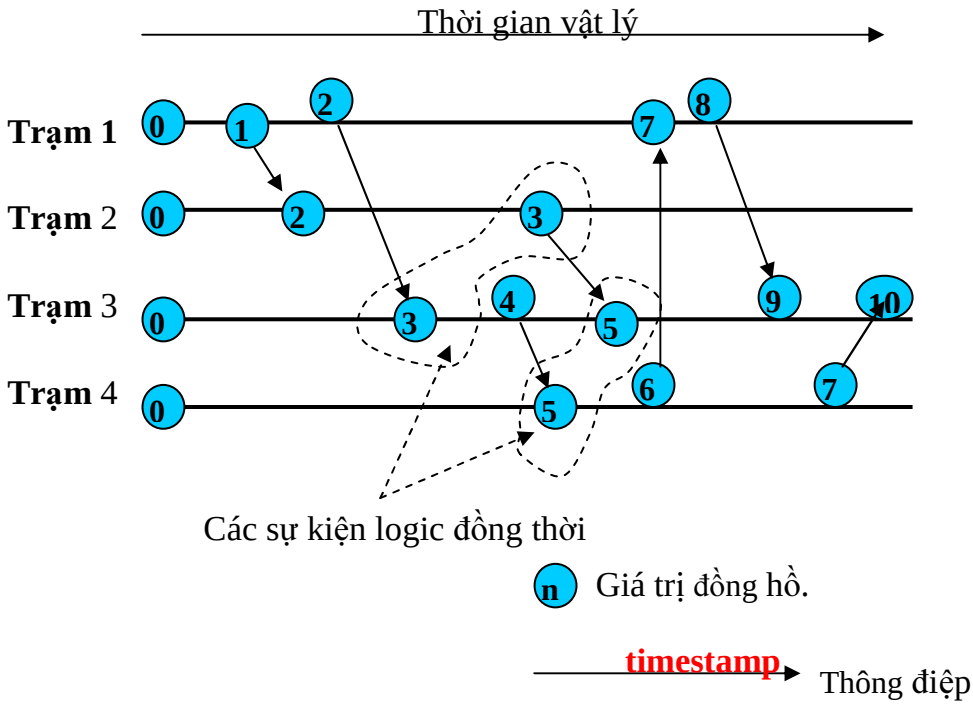
2.Xử lý R2.

3.Phát thông điệp.

Đồng hồ logic tại bất kỳ tiến trình nào là ngày càng tăng đơn giản.



Hình 2. 10 Các thời gian đánh dấu Lamport (Lamport timestamps)



Hình 2. 11 Ví dụ thời gian logic Lamport

2.3.3 Kết luận

Trong hệ thống phân tán, thời gian truyền thông tin của các thông điệp là không cố định, bên cạnh đó số lượng của các trạm ra quyết định lớn, điều này dẫn đến hệ thống phân tán có thể bị sự cố bất cứ lúc nào.

Các biện pháp đồng bộ hóa các tiến trình, mà cốt lõi là đồng bộ thời gian xử lý của các tiến trình trong hệ giúp các tiến trình được xử lý theo trật tự cố định. Đây là điều kiện giúp hệ thống tồn tại và có thể đưa vào sử dụng được.

CHƯƠNG 3: ỨNG DỤNG HỆ PHÂN TÁN TỐI ƯU THỜI GIAN XỬ LÝ CHO MÁY TÌM KIẾM

3.1 Phân tích máy tìm kiếm trên hệ tập trung

3.1.1 Phân tích hoạt động của máy tìm kiếm trên hệ tập trung

Máy tìm kiếm hoạt động dựa trên 3 bộ phận chính đó là bộ Crawler, bộ index và bộ Search engine (*xem mục 1.2*).

- Bộ Crawler có nhiệm vụ như một robot tự động tìm thông tin trên internet và lưu trữ về máy theo một cấu trúc nhất định (*thông tin thô*).
- Bộ Index có nhiệm vụ lập chỉ mục các file thông tin của bộ Crawler tải về thành hệ thống file index (*thông tin tinh lọc*) và tập trung tại một kho dữ liệu (*dữ liệu index file*).
- Bộ Search engine có nhiệm vụ truy vấn dữ liệu trong kho dữ liệu index file, sắp xếp kết quả và trả kết quả cho người dùng.

3.1.2 Một số hạn chế của máy tìm kiếm trên hệ tập trung

Theo như mô hình trình bày tại mục 1.2 và theo phân tích hoạt động của máy tìm kiếm trình bày ở trên thì tất cả các xử lý của hệ thống máy tìm kiếm được tập trung tại 1 server, do đó khi hệ thống vận hành sẽ phát sinh các hạn chế sau:

1. Dung lượng của hệ thống file index quá khổng lồ. Trên thế giới hiện tại ước tính khoảng 200 triệu tên miền được đăng ký và mỗi năm tăng thêm 19%. Mỗi tên niềm trung bình có khoảng 800 liên kết chứa dữ liệu. Như vậy số thông tin cần lưu trữ của quá lớn không thể triển khai trên server đơn được.
2. Số lượng người truy cập tập trung tại một server quá lớn.
3. Số lượng thông tin xử lý quá lớn.
4. Băng thông đường truyền quá tải.

➔ *Máy tìm kiếm không thể triển khai được trên hệ tập trung.*

3.1.3 Các yếu tố ảnh hưởng đến thời gian xử lý của máy tìm kiếm

Ta xét quá trình thực hiện và xử lý của hệ thống từ khi yêu cầu người dùng được gởi đi đến khi người dùng nhận được kết quả (*hình 3.1*) gồm các công đoạn như sau:

i. Máy cá nhân người dùng (*máy local*) xử lý
ii. Chuyển thông tin yêu cầu người dùng đến hệ thống
iii. Hệ thống xử lý yêu cầu người dùng
iv. Hệ thống chuyển kết quả đếm máy local của người dùng
v. Máy local của người dùng xử lý và hiển thị kết quả

Như vậy ta tạm chia thời gian xử lý của quá trình thành hai phần chính: Tổng thời gian xử lý, truyền thông của các công đoạn nằm ngoài hệ thống (T_{nht}) và tổng thời gian xử lý, truyền thông của các công đoạn nằm trong hệ thống (T_{tht}).

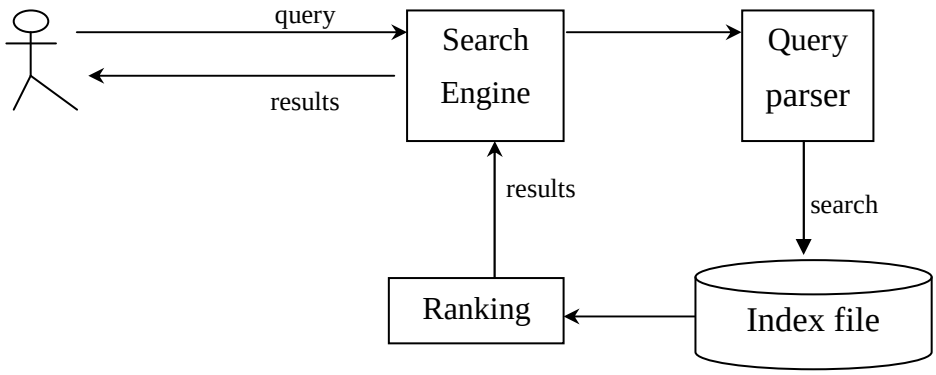
Ta có công thức như sau:

$$T = T_{nht} + T_{tht}$$

Các công đoạn xử lý nằm ngoài hệ thống chúng ta không thể can thiệp hoặc cải thiệp, do đó ta không xét. Như vậy để tối ưu thời gian xử lý của máy tìm kiếm chính là tối ưu thời gian xử lý T_{tht}

Các yếu tố ảnh hưởng đến thời gian xử lý T_{tht}

- Thời gian truyền thông điệp: Phụ thuộc số lượng, dung lượng gói tin và băng thông, khoảng cách đường truyền.
- Thời gian xử lý thông tin:
  + Cấu hình máy server
  + Số lượng thông tin cần xử lý
  + Dung lượng và cấu trúc kho dữ liệu
  + Độ phức tạp thuật toán



Hình 3. 1 Mô hình hoạt động của pha xử lý yêu cầu người dùng

Như vậy, để tối ưu thời gian xử lý thông tin của máy tìm kiếm chúng ta thực hiện tối ưu sáu tiêu chí sau:

Bảng 3.1. Bảng tiêu chí tối ưu máy tìm kiếm

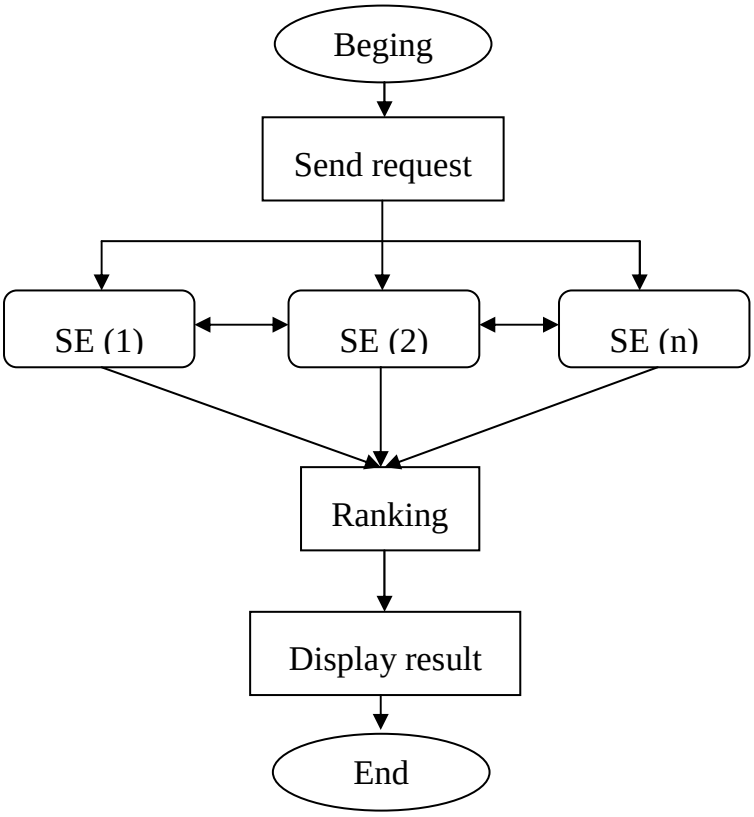
STT	**Tiêu chí tối ưu**
1	Số lượng và dung lượng gói tin cần truyền
2	Băng thông và khoảng cách đường truyền
3	Cấu hình server
4	Số lượng thông tin cần xử lý
5	Dung lượng và cấu trúc kho dữ liệu
6	Độ phức tạp thuật toán

3.1.4 Hướng giải quyết vấn đề

3.1.4.1Đặt vấn đề

Theo định nghĩa và các tính chất của hệ phân tán (*Trình bày tại chương 2*), hệ phân tán gồm nhiều server kết nối với nhau thông qua đường truyền viễn thông. Ưu điểm lớn nhất của hệ phân tán đa server đó là ta có thể chia sẽ tài nguyên giữa các trạm và tăng khả năng xử lý đồng thời. Điều này sẽ làm giảm thời gian xử lý thông tin rất nhiều.

Giả sử ta chia máy tìm kiếm ra thành nhiều máy tìm kiếm nhỏ đặt tại các server trong hệ thống phân tán. Kho dữ liệu index file cũng được chia nhỏ và phân tán tại các trạm. Khi một yêu cầu người dùng được chuyển tới, hệ thống sẽ thông báo cho tất cả các trạm thực hiện truy vấn vào kho index file riêng của mình và gửi kết quả lại cho hệ thống, hệ thống sẽ tổng hợp, sắp xếp kết quả và chuyển cho người dùng.



Hình 3. 2 Các bước hoạt động của máy tìm kiếm ứng dụng hệ phân tán

3.1.4.2Kết luận

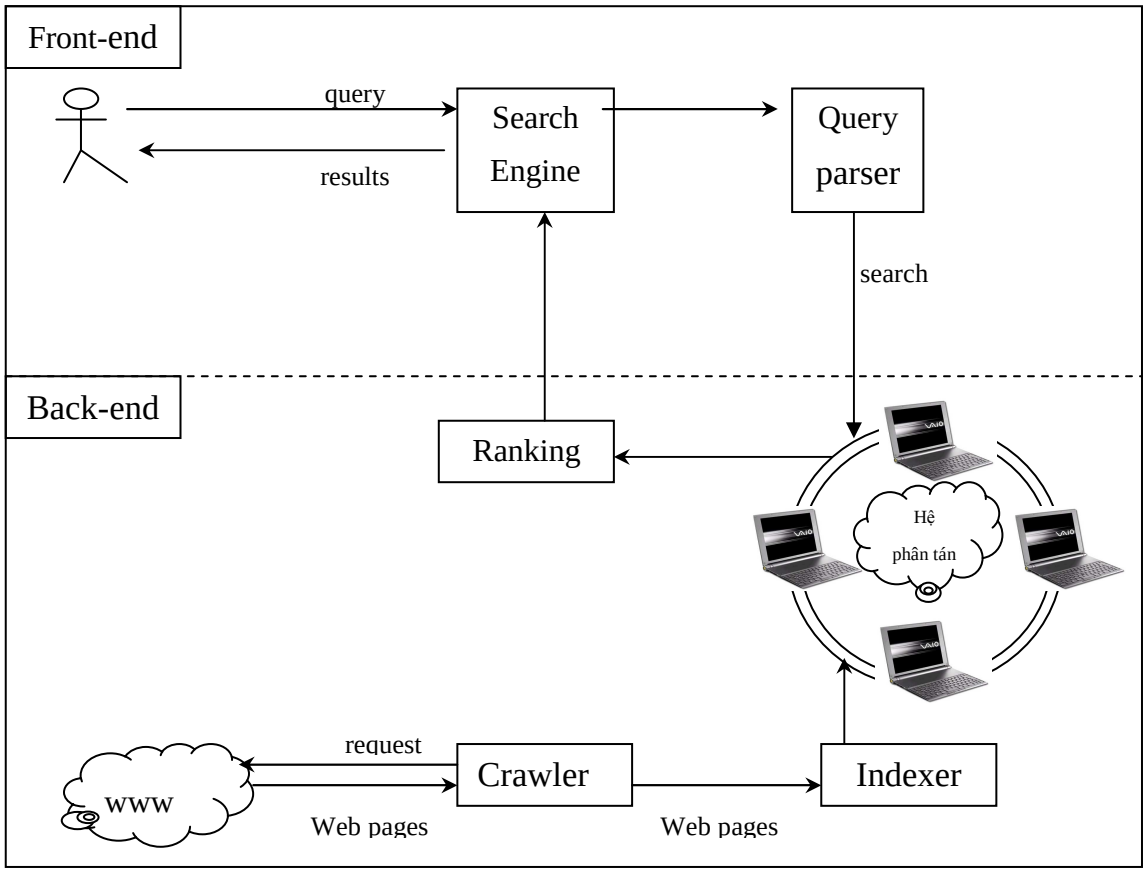
Như vậy, thời gian xử lý của hệ thống sẽ tỉ lệ nghịch với số lượng các trạm trong hệ thống, số lượng các trạm càng tăng thì thời gian xử lý càng giảm. Ứng dụng hệ phân tán đa server vào máy tìm kiếm ta có thể giải quyết được các vấn đề sau:

i. Kho dữ liệu index file được phân tán tại tất cả các trạm trong hệ → dung lượng kho dữ liệu index file tại mỗi trạm giảm xuống tương ứng với số lượng trạm trong hệ.
ii. Tất cả các trạm đồng thời cùng tham gia vào việc xử lý thông tin → Tăng cấu hình của hệ thống lên tương ứng với số lượng trạm trong hệ.
iii. Tăng băng thông, tối ưu khoảng cách đường truyền của hệ thống
iv. Giảm số lượng và dung lượng gói tin cần truyền tại mỗi trạm.

Như vậy ứng dụng hệ phân tán đa server sẽ giảm thời gian xử lý của máy tìm kiếm một cách đáng kể.

3.2 Đề xuất phương thức hoạt động của máy tìm kiếm trên hệ phân tán

3.2.1 Phương thức hoạt động tổng thể của hệ thống



Hình 3.3 Mô hình hoạt động tổng thể máy tìm kiếm ứng dụng hệ phân tán

Trên hệ thống tập trung, mọi xử lý của máy tìm kiếm được tập trung thực hiện tại một server, do đó thời gian xử lý yêu cầu người dùng quá lớn, thậm chí quá tải không thực hiện được mọi hoạt động. Do vậy, chúng ta phải thực hiện phân tán máy tìm kiếm ra thành nhiều máy tìm kiếm nhỏ và các máy tìm kiếm nhỏ này hoạt động như một máy tìm kiếm thực thụ với đầy đủ các chức năng.

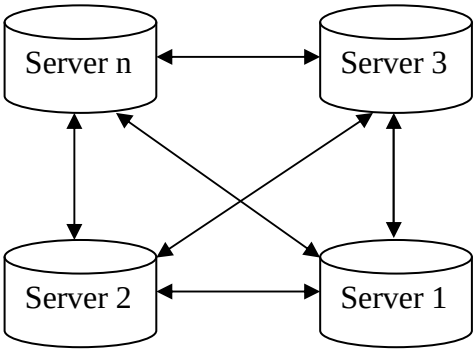
Trong hệ thống tập trung, mọi quá trình xử lý được thực hiện tại một server. Trong hệ thống ứng dụng phân tán, quá trình xử lý được chia nhỏ và được thực hiện xử lý đồng thời tại nhiều server khác nhau và kết quả được tập hợp từ các trạm trong hệ thống.

Xử lý yêu cầu người dùng: Khi một yêu cầu (*request*) của người dùng gửi tới hệ thống, hệ thống sẽ tiếp nhận yêu cầu và dựa trên một số tiêu chí (*các tiêu chí sẽ được trình bày ở phần sau*) nó ủy quyền cho một trạm cụ thể chịu trách nhiệm xử lý chính. Trạm được ủy quyền này sẽ gởi yêu cầu đến các trạm khác trong hệ. Tại mỗi trạm sẽ xử lý riêng biệt và gởi kết quả lại cho trạm được ủy quyền. Trạm này có nhiệm vụ tập hợp kết quả, sắp xếp theo thứ tự giảm dần của độ chính xác với từ khóa người dùng và hiển thị kết quả cho người sử dụng.

Thu thập thông tin: Tất cả các trạm xử lý trong hệ thống đều giống nhau. Tại mỗi trạm đều có bộ crawler và bộ indexer riêng biệt, mỗi trạm tự động lấy thông tin và lập chỉ mục, lưu trữ hệ thống index file riêng và luôn đảm bảo sự duy nhất của thông tin trong hệ thống.

3.2.2 Phương thức liên kết các trạm trong hệ thống

Hệ thống gồm n server trạm tương tự nhau, chúng liên kết với nhau qua đường truyền viễn thông và giao tiếp với nhau bằng thông điệp. Mỗi server là một máy tìm kiếm và có khả năng liên kết với tất cả các server còn lại trong hệ thống.



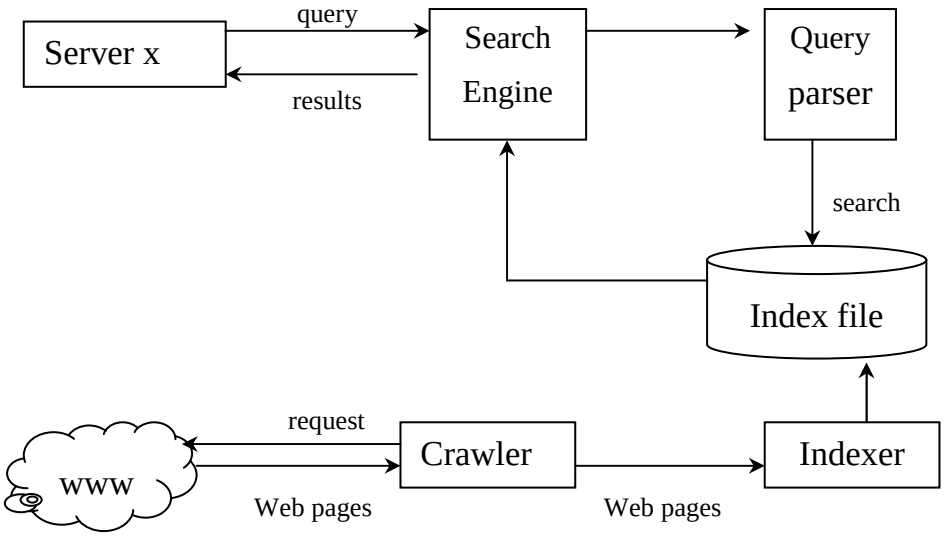
Hình 3. 4 Mô hình liên kết các trạm trong hệ thống

Mỗi server có một hệ thống thu thập thông tin và kho dữ liệu index file riêng. Dữ liệu của kho index file tại mỗi server là duy nhất.

Khi một server trong hệ thống nhận được thông điệp yêu cầu truy xuất dữ liệu, nó sẽ thông báo cho các server còn lại. Và quá trình xử lý sẽ được chia nhỏ tại tất cả các server trong hệ.

Các trạm trong hệ thống là một máy tìm kiếm, chúng cũng có đầy đủ 4 bộ phận chính như một máy tìm kiếm thông thường (*hình 3.3*), chúng hoạt động độc lập với nhau.

3.2.3 Phương thức hoạt động tại các trạm của hệ thống

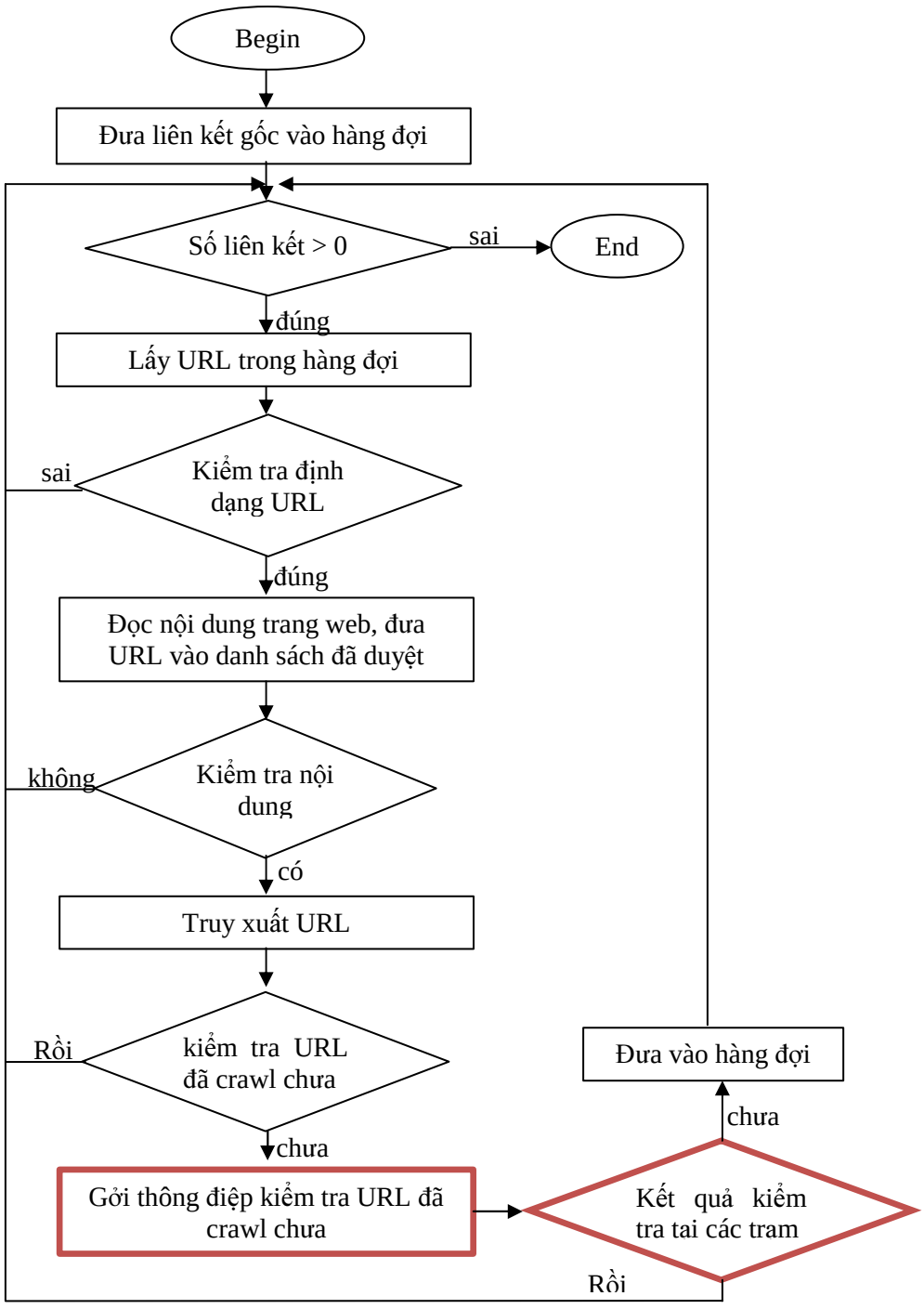


Hình 3. 5 Mô hình hoạt động của trạm các trạm con trong hệ thống

Như đã trình bày tại mục 3.1.2.3, tại mỗi trạm là một máy tìm kiếm thông thường, chúng tự động tìm thông tin trên internet, tự động lập chỉ mục và lưu trữ vào hệ thống index file.

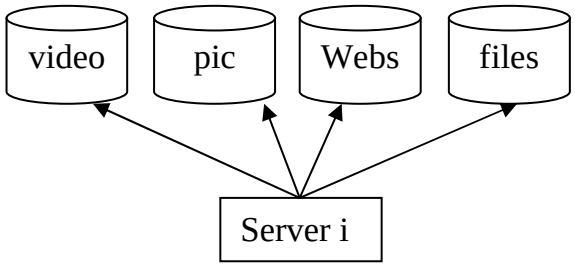
Khi server x (*server được quyền xử lý chính*) gửi thông điệp request tới các trạm yêu cầu truy xuất thông tin, bộ query parser tại các trạm phân tích câu truy vấn và truy xuất tới nơi chứa thông tin cần tìm.

Bộ phận crawler hoạt động hơi khác so với bộ phận crawler của máy tìm kiếm trên hệ tập trung. Do mỗi trạm có một bộ crawler hoạt động độc lập, điều này dẫn đến sự trùng lặp thông tin giữa các trạm. Để tránh trường hợp các crawler của các trạm tải thông tin trùng nhau, sau khi truy xuất các URL trong nội dung trang web, crawler ngoài việc kiểm tra các URL đó đã crawl chưa còn phải gởi thông điệp nhờ tất cả các trạm khác trong hệ thống kiểm tra xem URL đó đã có trạm mào kiểm tra hay chưa, nếu có bất kỳ trạm nào crawl rồi thì crawler sẽ loại bỏ URL đó ra không xử lý.



Hình 3. 6 Thuật toán xử lý của crawler

3.2.4 Phương thức lưu trữ file index của hệ thống



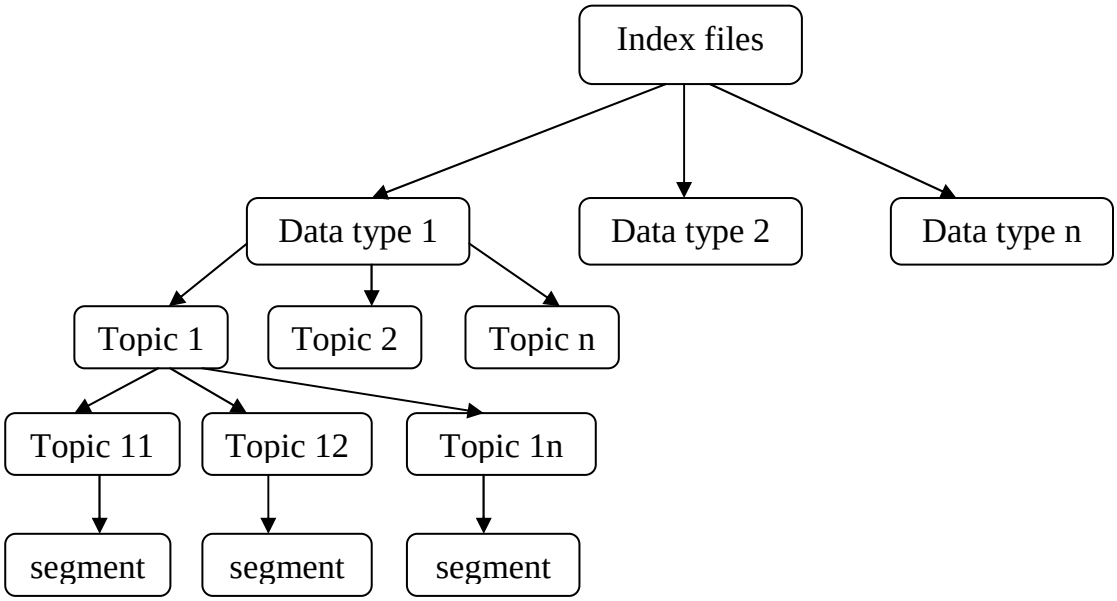
Hình 3. 7 Mô hình lưu trữ hệ thống files index tại mỗi trạm

Tại mỗi trạm hệ thống files index được lưu trữ theo mô hình như đã trình bày tại mục 1.6, đồng thời hệ thống file index được phân loại theo nhiều loại dữ liệu khác nhau như webs, videos, files, picture… Và tại các loại dữ liệu ta tiếp tục phân loại theo các chủ đề khác nhau để tiện cho việc truy xuất và tìm kiếm thông tin theo từng loại dữ liệu.

Mục đích của việc chia nhỏ thông tin thành từng loại dữ liệu và từng chủ đề cụ thể giúp việc truy vấn dữ liệu được chính xác và nhanh chóng hơn.

Ví dụ, tại kho dữ liệu webs ta có thể chia ra thành các chủ để như giáo dục, văn hóa, xã hội, kinh tế, chính trị…Tại các chủ đề này ta có thể tiếp tục chia nhỏ thành các chủ đề con như bộ giáo dục, mầm non, tiểu học, trung học, đại học, cao đẳng … và cứ thế chia nhỏ theo mô hình cây quan hệ.

Các nút lá của cây quan hệ là các segments chứa thông tin tinh lọc của bộ indexer trích lọc được từ thông tin thô của bộ crawler tải về. Mỗi segments là một hệ thống các từ vựng và các mã của url chứa từ vựng đó. URL gồm có hai loại url trên web và url trên máy local. Url trên máy local là địa chỉ các file chứa các từ vựng đó. Mục đích của url trên máy local giúp cho người dùng truy xuất được thông tin của các url trên web đã bị ngưng kết nối vì lý do gì đó.



Hình 3. 8 Hệ thống index file theo mô hình cây

Dựa vào cách lưu trữ này, bộ query parser sẽ phân tích thông tin người dùng và thực hiện truy vấn trực tiếp và các segment có liên quan. Do vậy, kết quả tìm kiếm chính xác và nhanh chóng hơn.

3.3 Các vấn đề phát sinh và cách giải quyết

3.3.1 Chọn lựa server xử lý chính

3.3.1.1Đặt vấn đề

Hệ thống gồm nhiều server tương tự nhau, có chức năng giống nhau, tại mỗi thời điểm mỗi server có độ "rỗi" khác nhau. Khi một yêu cầu người dùng được gởi đến hệ thống, hệ thống sẽ lựa chọn server nào tối ưu nhất để giao quyền xử lý chính nhằm tối ưu thời gian xử lý cho máy tìm kiếm là vấn đề cần thiết.

Độ "rỗi" của server tại một thời điểm được định nghĩa dựa trên thời gian xử lý một đơn vị thông tin của server tại thời điểm đó. Một server A có độ rỗi cao hơn server B, điều này có nghĩa server A có tốc độ xử lý trên một đơn vị thông tin cao hơn tốc độ xử lý trên một đơn vị thông tin của server B.

3.3.1.2Giải quyết vấn đề

Căn cứ bảng tiêu chí tối ưu (*xem tại mục 3.2.1*), gồm có sáu tiêu chí để tối ưu thời gian xử lý cho máy tìm kiếm. Trong hệ thống, các server được cài đặt chung một chương trình xử lý giống nhau, do đó độ phức tạp của thuật toán của các server là như nhau. Tổng quá lên ta có hai tiêu chí chính để chọn một server tối ưu tại một thời điểm T là *thời gian xử lý một đơn vị thông tin của server đó và thời gian truyền một đơn vị thông tin từ server đó đến client tại thời điểm T* .

Như vậy, để chọn server tối ưu ta dựa vào hai tiêu chí chính đó là:

Bảng 3.2. Bảng tiêu chí chọn server tối ưu

STT	**Tiêu chí**	**ĐV tính**
1	Thời gian xử lý một đơn vị thông tin	ms
2	Thời gian truyền một đơn vị thông tin từ server đến client	ms

Giả sử hệ thống gồm bốn server kết nối với nhau. Ta xét trong khoảng thời gian t, giả sử thông số của các yếu tố ảnh hưởng đến thời gian xử lý tại các trạm như sau.

Bảng 3.3. Bảng phân tích độ rỗi khác nhau của các server trong hệ

Server	Thời gian xử lý	Thời gian truyền thông tin	Tổng thời xử lý
1	0,005 ms	0,02 ms	0.025 ms
2	0,003 ms	0,05 ms	0.053 ms
3	0,004 ms	0,1 ms	0.104 ms
4	0,0025 ms	0,015 ms	0.0175 ms

Dựa vào tổng thời gian xử lý của từng server ta có thể xác định được server tối ưu (*server có tổng thời gian xử lý nhỏ nhất*) để giao quyền xử lý chính.

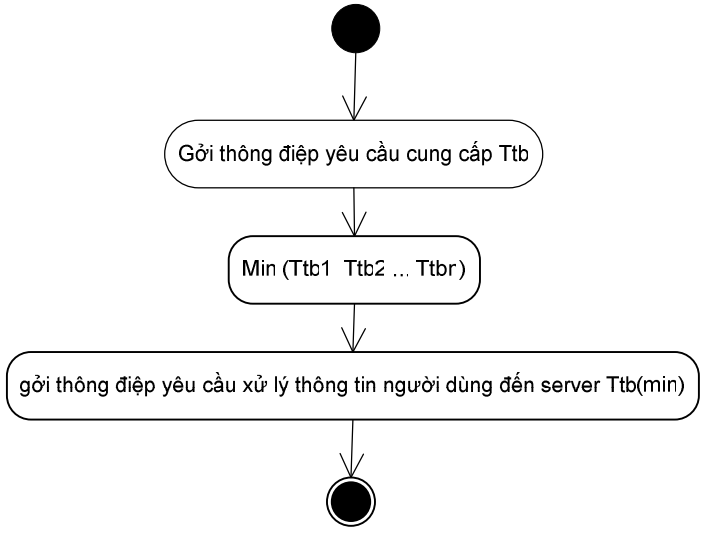
Các yếu tố ảnh hưởng đến thời gian xử lý của một server:

- Tốc độ xử lý của CPU.
- Dung lượng bộ nhớ tạm RAM và Bus của RAM.
- Tốc độ quay và chất lượng của đĩa cứng.
- FSB (*Front Side Bus*) của Main (xa lộ truyền dữ liệu).

Các yếu tố ảnh hưởng đến thời gian truyền thông tin:

- Tốc độ đường truyền.
- Khoảng cách điểm nguồn và điểm đích.
- Tốc độ các thiết bị trung gian.

Trước khi submit câu truy vấn của người dùng đến hệ thống, client gởi một thông điệp đến tất cả các trạm yêu cầu các trạm trả lời tổng thời gian xử lý (T) của mình. Sau khi nhận được thời gian T của tất cả các trạm, client thực hiện chọn server có T nhỏ nhất làm server xử lý chính và gởi câu truy vấn của người đến server đó yêu cầu xử lý.



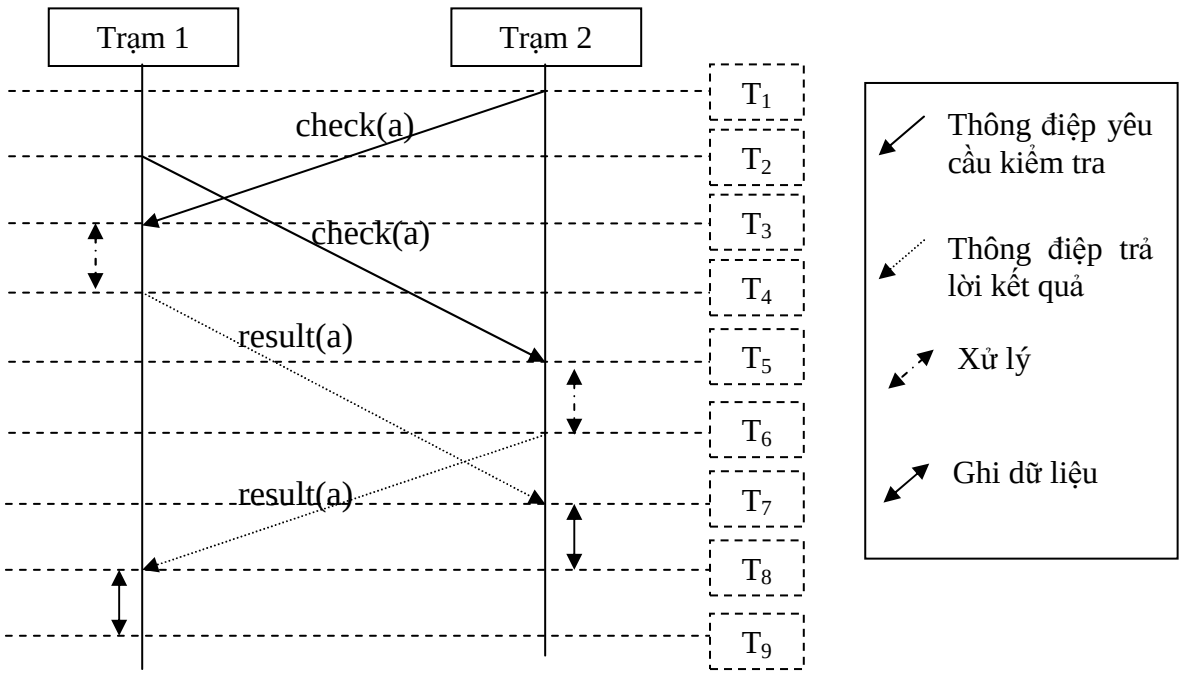
Hình 3. 9. Sơ đồ chọn server tối ưu

3.3.2 Vấn đề đồng bộ các tiến trình

3.3.2.1Đặt vấn đề

Giả thiết:

- Giả sử hệ thống gồm có 2 server trạm được liên kết với nhau thông qua đường truyền viễn thông và giao tiếp với nhau bằng hệ thống thông điệp. Các trạm cùng tiến hành công việc thu thập thông tin.
- Giả sử hệ thống thực hiện giao tiếp với nhau bằng 2 loại thông điệp: check(a) dùng để kiểm tra URL a có crawl chưa và result(a) dùng trả kết quả kiểm tra của URL a.



Hình 3. 10 Mô hình không đồng bộ của hai tiến trình giữa hai trạm

- Giả sử tại thời điểm t_1 trạm 2 gửi một thông điệp yêu cầu trạm 1 kiểm tra url a đã được trạm 1 crawl chưa và đến thời điểm t_3 trạm 1 mới nhận được thông điệp. Trong khi đó tại thời điểm t_2 trạm 1 cũng gửi thông điệp yêu cầu trạm 2 kiểm tra url a và đến thời điểm t_5 trạm 2 mới nhận được thông điệp. Trong khoảng thời gian t_3 đến t_4, tại trạm 1 url a chưa có trong cơ sở dữ liệu, do đó kết quả result(a) = NO và được gửi qua trạm 2. Tại thời điểm t_5, t_6 trạm 2 chưa nhận được kết quả từ

trạm 1 gửi đến, do đó url a cũng chưa được ghi vào cơ sở dữ liệu của trạm 2 và kết quả result(a) =NO. Điều này dẫn tới 2 trạm đều ghi url a vào cơ sở dữ liệu của mình.

Kết luận: *Dữ liệu tại các trạm sẽ bị trùng, không nhất quán. Điều này, dẫn tới dữ liệu bị dư thừa. Nguyên nhân của điều này chính là sự không đồng bộ giữa các tiến trình của các trạm.*

3.3.2.2Giải quyết vấn đề

Phương pháp đồng bộ hóa tiến trình

Vấn đề được đề cập bên trên tương tự như bài toán bãi để xe [2, tr 157] và bài toán người sản xuất – người tiêu thụ [2, tr 162]. Việc không đồng bộ giữa các tiến trình tại các trạm trong hệ thống dẫn đến các vấn đề sai lệch kết quả trong quá trình vận hành, mà nguyên nhân chính đó là thứ tự thực hiện của các tiến trình không đồng bộ do độ trễ của các thông điệp (*trình bày tại mục 2.3.2*).

Giải quyết vấn đề này chính là giải quyết đồng bộ hóa tiến trình (*trình bày tại mục 2.3.2*). Trong nội dung thông điệp ta đính kèm thêm nhãn thời gian logic, địa chỉ nguồn của thông điệp và dựa vào đồng hồ logic này ta xác định thông điệp của trạm nào được ưu tiên xử lý. Thông điệp nào có đồng hồ logic nhỏ hơn thì thông điệp đó được ưu tiên xử lý, thông điệp còn lại bị hủy.

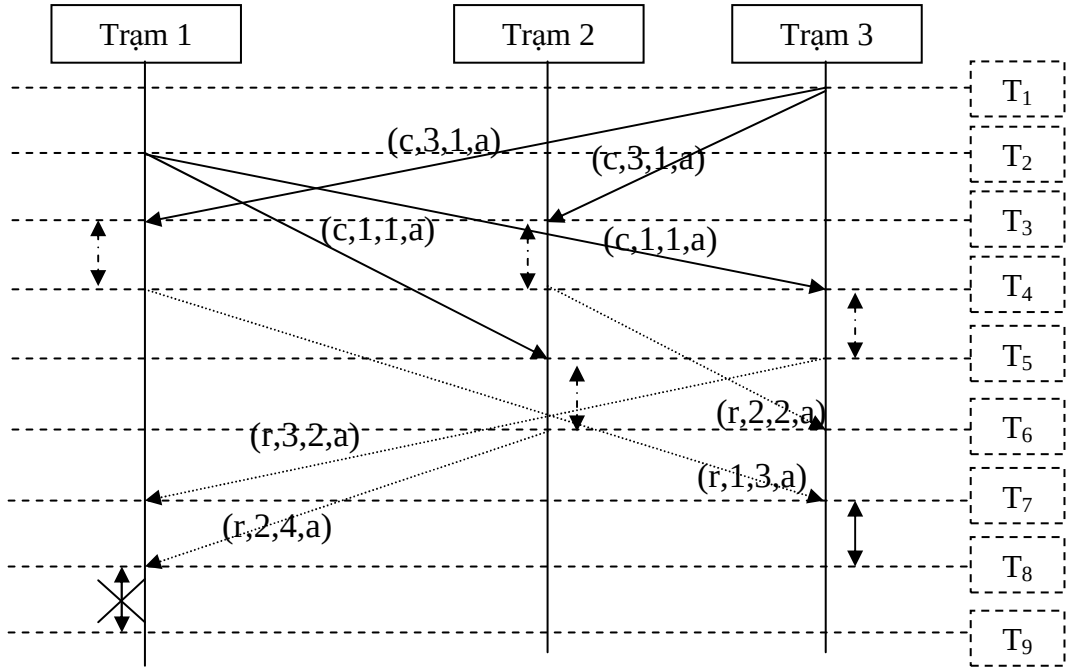
Thuật toán được thực hiện như sau:

Gán đồng hồ logic $T_i = 0$ cho tất cả các trạm

Khi một trạm thực hiện gởi thông điệp, trạm đó tự động tăng đồng hồ logic của mình lên một đơn vị $T_i=T_i + 1$ rồi gắn số hiệu đồng hồ logic C_i của mình vào nội dung thông điệp và gởi cho trạm đích.

Khi nhận được một thông điệp, trạm cập nhật số hiệu đồng hồ logic bằng cách lấy giá trị lớn nhất của số hiệu đồng hồ logic trạm gởi và số hiệu đồng hồ logic của mình $T_i=max(T_i , C_k)$.

Khi trạm nhận được đầy đủ thông điệp trả lời của các trạm, ngay lập tức trạm so sánh đồng hồ logic của mình với các trạm khác, nếu nhỏ hơn thì trạm thực hiện xử lý tiếp và gởi thông báo cho các trạm còn lại hủy việc xử lý.



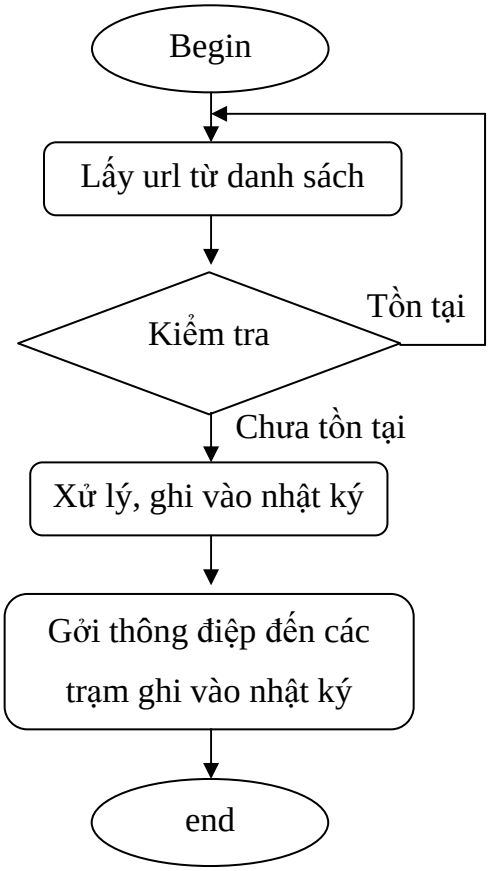
Hình 3. 11.Kết quả sau khi đồng bộ tiến trình theo thuật toán lamport

Nhược điểm của phương pháp này là lượng thông điệp cần gởi đi xử lý tăng lên rất nhiều, mỗi lần kiểm tra hệ thống phải gởi lượng thông điệp là (n-1)*2 trong đó n là số trạm trong hệ thống, do đó ảnh hưởng nhiều đến thời gian xử lý.

Phương pháp lưu nhật ký

Tại mỗi trạm chúng ta tổ chức một hệ thống lưu dữ tất cả các URL của tất cả các trạm đã được crawler. Khi một URL được lấy trong danh sách ra kiểm tra, thay vì gởi thông điệp đi yêu cầu từng trạm một kiểm tra tình trạng của URL thì trạm đó kiểm tra trực tiếp tại hệ thống nhật ký đã được lưu trữ của mình.

Phương pháp này không cần phải gởi thông điệp, nhưng thay vào đó tại tất cả các trạm phải thực hiện lưu nhật ký của tất cả URL đã được crawler.



Hình 3. 12 Thuật toán kiểm tra tình trạng URL

3.3.3 Vấn đề sự cố đường truyền

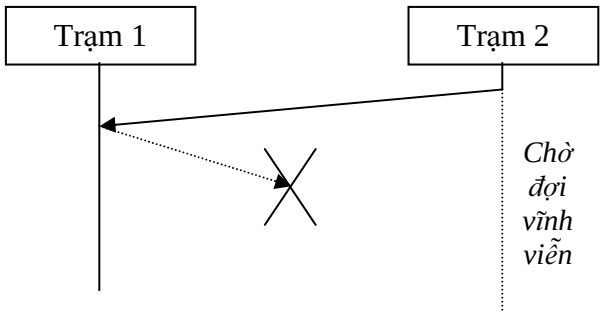
3.3.3.1Đặt vấn đề

Như đã trình tại mục 2.2, truyền thông là yếu tố tối quan trọng trong hệ phân tán, hệ phân tán sẽ không tồn tại nếu không có truyền thông. Thế nhưng trong thực tế truyền thông rất không ổn định, có thể mất kết nối bất cứ lúc nào, thông điệp cũng có thể thất lạc không đến được nơi nhận.

Nếu đường truyền bị hỏng, ngoài vấn đề làm mất các thông báo tuyền qua, nó cũng có thể phân cắt mạng thành hai hoặc nhiều nhóm tách rời. Tình huống này

được gọi là phân hoạch mạng, lúc đó các vị trí trong mỗi phân hoạch có thể vẫn tiếp tục hoạt động. Khi đó việc thực hiện các giao dịch cần truy xuất đến nhiều phân hoạch trở thành một vấn đề quan trọng.

Giả sử trạm 1 gởi thông điệp yêu cầu kiểm tra (C,1,2,“url a”) và thông điệp này được ưu tiên xử lý, nhưng thông điệp trả lời kết quả của một trạm trong hệ thống không gởi đến được trạm 1 *(do đường truyền bị gián đoạn)*. Điều này làm cho tiến trình crawl url a của trạm 1 ở trạng thái chờ vĩnh viễn *(tiến trình chết)*. Tương tự như vậy các trạm sẽ tồn tại rất nhiều tiến trình chết.



Hình 3. 13 Mô hình sự cố đường truyền

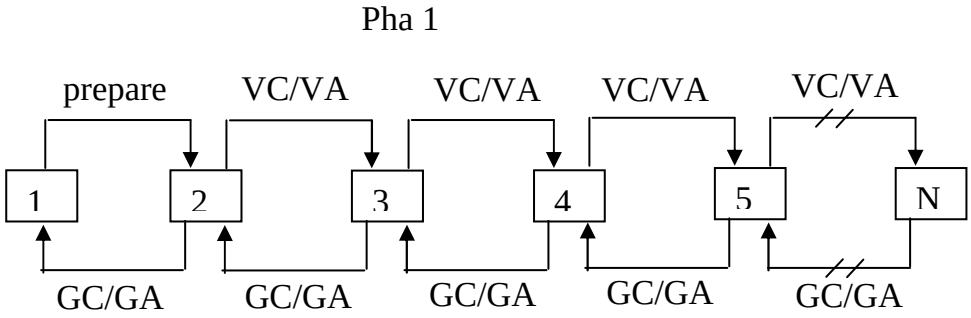
3.3.3.2Giải quyết vấn đề

Xét trong khoảng thời gian α *(α: là hằng số)*, kết quả của sự cố đường truyền tạm chia ra hai loại: Thất lạc thông điệp và phân hoạch mạng

Ở đây ta đưa ra giải thuật hai pha tuyến tính (Linear two phase commit - 2PC). Trong đó các thành viên có thể trao đổi với nhau. Chúng ta giả thiết rằng thứ tự giữa các vị trí có tham gia vào việc thực hiện một giao dịch là 1, 2,…,N với điều phối viên là vị trí đầu tiên giải thuật này hoạt động như sau:

Điều phối viên gửi thông điệp prepare đến thành viên 2. Nếu thành viên 2 chưa sẵn sàng ủy thác giao dịch, nó gửi thông điệp biểu quyết hủy bỏ Vote-abort (VA) đến thành viên 3 và giao dịch bị hủy tại thời điểm này (hủy bỏ đơn phương của 2). Ngược lại nếu thành viên 2 đồng ý ủy thác, nó gửi thông điệp vote-commit (VC) cho thành viên 3 rồi chuyển sang trạng thái READY. Quá trình này tiếp tục cho đến

khi một biểu quyết uỷ thác đến được thành viên N. Đến đây kết thúc pha đầu tiên. Nếu N quyết định ủy thác nó gửi trở lại cho thành viên N-1 thông báo global-commit (GC); bằng không, nó gửi một thông điệp hủy bỏ toàn cục global-abort (GA). Theo đó các thành viên chuyển sang trạng thái thích hợp (COMMIT hoặc ABORT) và làm lan truyền thông điệp trở về điều phối viên



Hình 3. 14 Cấu trúc giao tiếp 2PC tuyến tính

Như vậy theo giải thuật 2PC tuyến tính, giả sử có một trạm trong hệ thống bị sự cố không tiếp nhận được thông điệp, ngay lập tức hệ thống gởi thông điệp thông báo cho các trạm còn lại để các trạm xác định lại trạm "*hàng xóm*" của mình. Mặt khác hệ thống gửi thông điệp thông báo việc gia nhập trở lại của các trạm bị sự cố cho các trạm được biết.

3.3.4 Vấn add, remove các trạm

3.3.4.1Đặt vấn đề

Theo quan điểm trình bày tại mục 3.3,2, nếu các trạm trong hệ thống gởi thông điệp kiểm tra lần thứ hai và đã chờ trong khoảng thời gian α mà vẫn không có phản hồi, điều này, có nghĩa trạm đích đó đã bị loại bỏ khỏi hệ thống (*remove*). Sau khi khắc phục sự cố ta phải thực hiện add trạm này vào lại hệ thống.

Vì lý do đường truyền và sự cố tại các trạm nên hệ thống luôn có tình trạng remove hoặc add *(thêm trạm vào hệ thông)* của các trạm trong hệ thống.

3.3.4.2Giải quyết vấn đề

Giải quyết vấn đề add – remove các trạm trong hệ thống phân tán, ta tập trung giải quyết 3 vấn đề chính đó là:

i. Thông báo cho hệ thống biết việc add – remove của mình.
ii. Cập nhật lại đồng hồ logic.
iii. Giải quyết việc nhất quán dữ liệu.

Dữ liệu của hệ thống được phân tán tại mỗi trạm, khi một trạm bị đứt ra khỏi hệ thống đồng nghĩa với việc một phần dữ liệu của hệ bị mất theo, nhưng khi vận hành các trạm trong hệ phải luôn kiểm tra dữ liệu của nhau để đảm bảo dữ liệu của hệ luôn được nhất quán. Như vậy, chúng ta phải làm thế nào để không ảnh hưởng đến hoạt của hệ khi một hay nhiều trạm bị đứt ra khỏi hệ.

Để giải quyết vấn đề, ta đưa ra hai phương án như sau:

i. Phục hồi nhờ hệ thống backup

Tại mỗi trạm ta xây dựng thêm một trạm backup (*là bản sao của trạm chính*) hai trạm này hoạt động đồng thời với nhau. Nếu một trong hai trạm bị sự cố thì trạm còn lại vẫn hoạt động bình thường. Nếu trạm bị sự cố sau khi khắc phục và gia nhập lại hệ thống thì chúng thực hiện sao chép lại toàn bộ dữ liệu của trạm còn lại.

Để thực hiện phương án này, chúng ta phải xây dựng thêm một hệ thống tương tự phục vụ việc backup. Như vậy, đòi hỏi một khoản chi phí gấp đôi để xây dựng hệ thống. Tính về mặt kinh tế thì phương án này không khả thi.

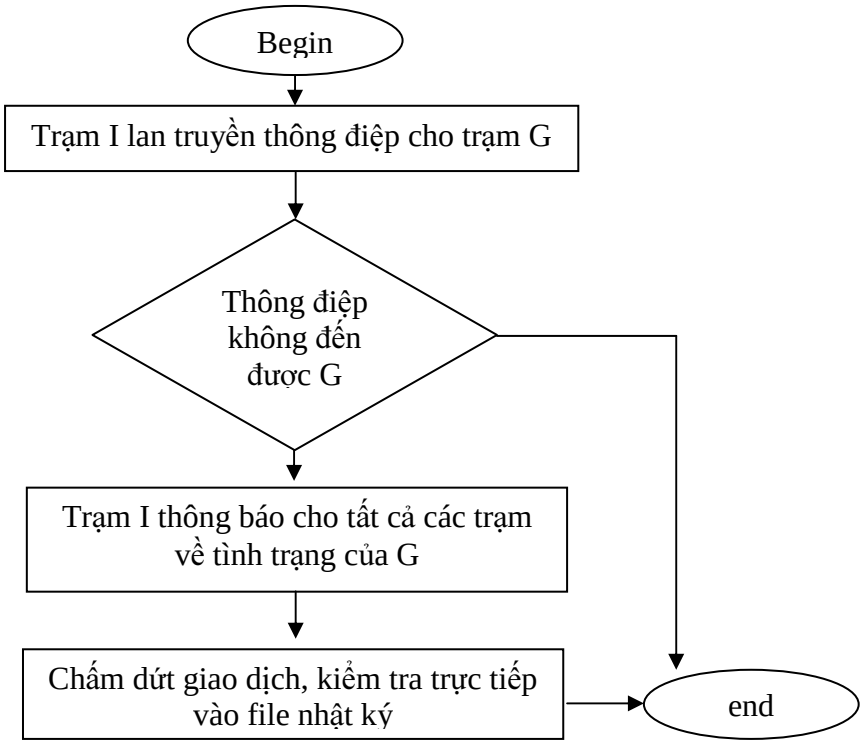
ii. Phục hồi nhờ nhật ký

Mọi hoạt động của từng trạm trong hệ thống được bản thân mỗi trạm ghi chép lại thành file nhật ký chứa thông tin của tất cả các URL đã được crawler.

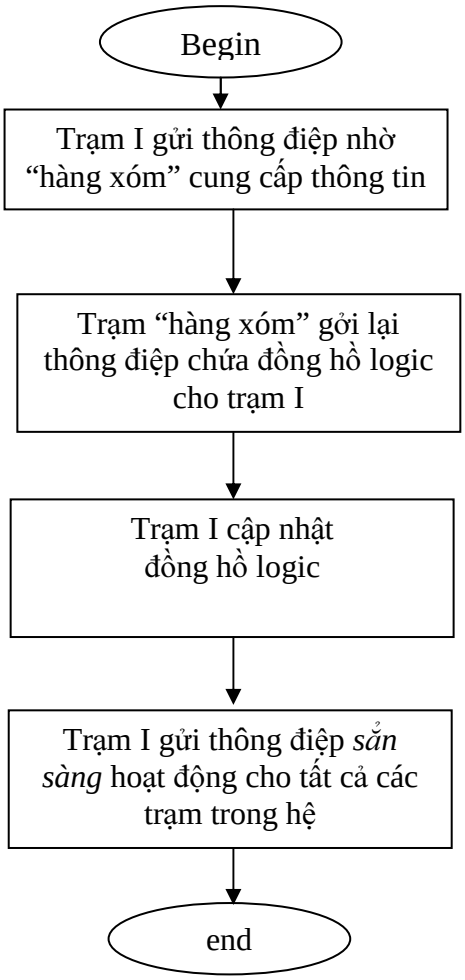
Hệ thống file nhật ký gồm tập hợp các file với tên file trùng với tên các trạm trong hệ thống. Mỗi file lưu trữ thông tin url của mỗi trạm đã clawler được và đồng hồ logic của từng trạm. Hệ thống file nhật ký của các trạm giống nhau hoàn toàn cả về cấu trúc và thông tin. Khi một url được crawler và mỗi lần cập nhật đồng logic chúng phải thực hiện ghi vào file nhật ký của tất cả các trạm.

Khi một trạm bị sự cố, "*hàng xóm*" của trạm đó sẽ thông báo cho tất cả các trạm trong hệ được biết. Các trạm sẽ chấm dứt mọi giao dịch với trạm bị sự cố ngay tức khắc. Việc kiểm tra url sẽ đươc thực hiện trực tiếp trên file nhật ký.

Khi một trạm I muốn gia nhập vào lại hệ thống, nó gửi một thông điệp nhờ trạm "*hàng xóm*" gởi lại đồng hồ logic cho trạm I, Trạm I thực hiện cập nhật đồng hồ logic, đồng thời gửi một thông điệp *sẳn sàng* thông báo cho các trạm trong hệ biết.



Hình 3. 15 Thuật toán xử lý trạm remove khỏi hệ



Hình 3. 16 Thuật toán xử lý việc add các trạm

3.4 Phân tích hệ thống

3.4.1 Danh sách các tác nhân hệ thống

3.4.1.1Người sử dụng (user)

Người sử dụng thao tác với hệ thống thông qua giao diện người dùng (*user interface*). User nhập câu thông tin cần tìm kiếm vào ô nhập hiệu của giao diện người sử dụng. User chọn các tùy chọn tìm kiếm (*kiểu dữ liệu cần tìm, phạm vi tìm kiếm, thể loại* …). User chọn nút submit để gởi thông tin cần tìm kiếm đến hệ thống xử lý. Hệ thống sẽ xử lý và trả lại kết quả cho User.

3.4.1.2Quản trị (admin)

Quản trị thao tác hệ thống thông qua giao diện admin (*admin interface*). Admin khởi động các trạm trong hệ thống, kiểm tra sự kết nối giữa các trạm, khắc phục sự cố nếu có. Khởi động bộ truy tìm thông tin tự động và bộ lập chỉ mục tự động cho từng trạm.

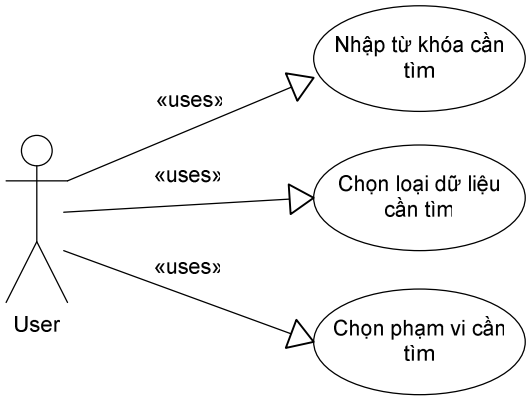
Quản trị thực hiện cân bằng tải cho các trạm trong hệ thống theo định kỳ hoặc ngẫu nhiên, giúp cho dung lượng lưu trữ thông tin của các trạm tương đương nhau.

Quản trị thực hiện khai báo thêm, bớt các trạm trong hệ thống, sao lưu dữ liệu thường xuyên và thực hiện các công việc bảo trì hệ thống.

Quản trị thực hiện sao lưu dự phòng dữ liệu

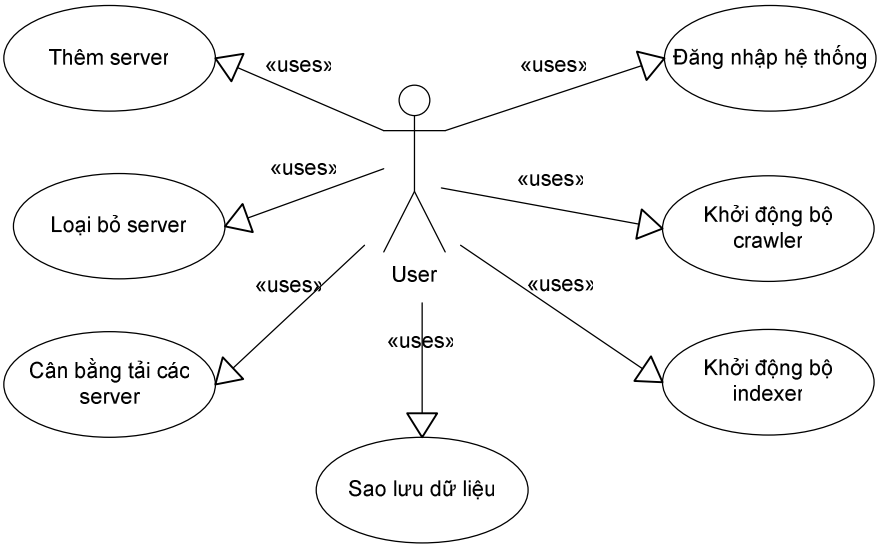
3.4.2 Sơ đồ tác nhân (UC)

3.4.2.1Biểu đồ tác nhân (UC) của người sử dụng



Hình 3. 17 biểu đồ UC của người sử dụng

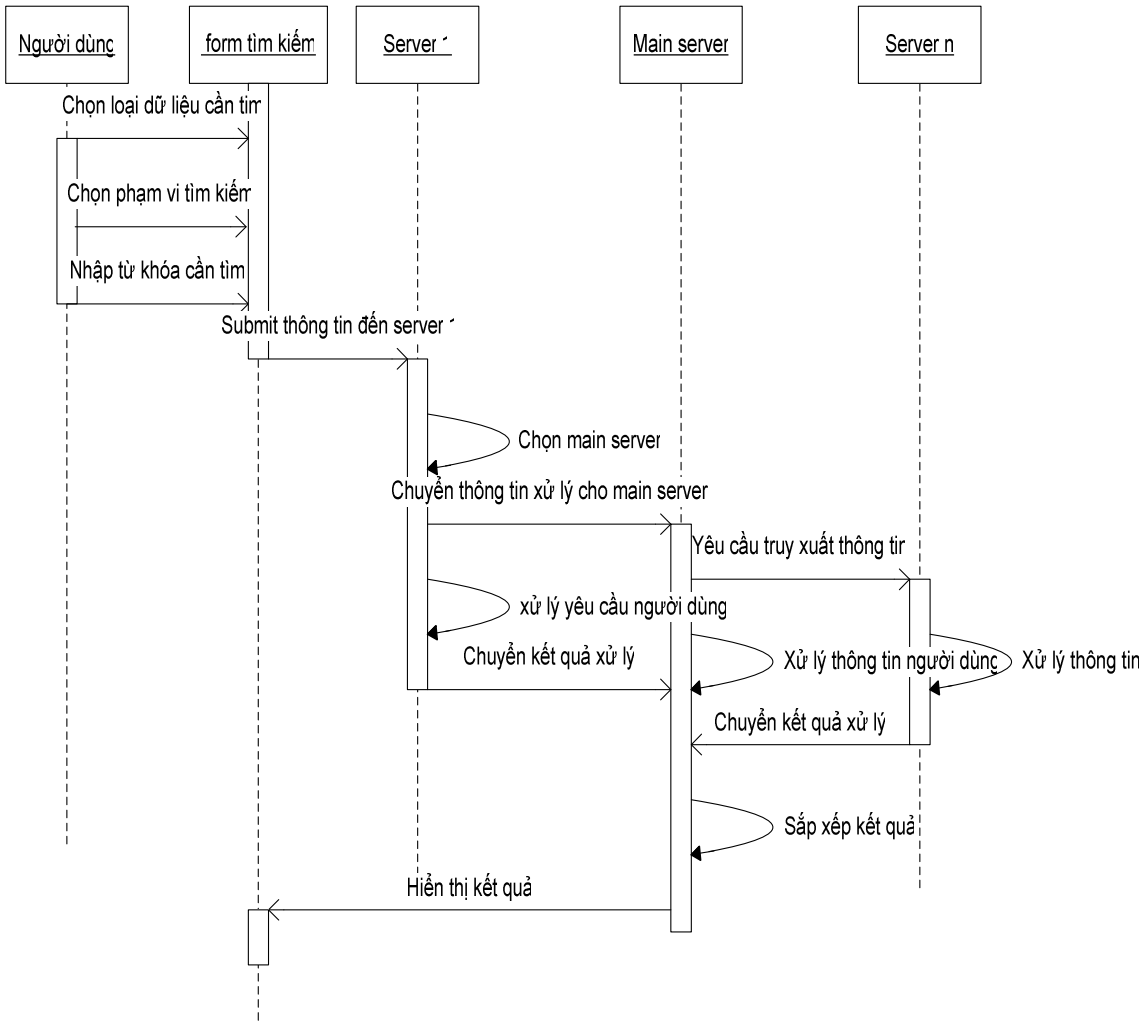
3.4.2.2Biểu đồ tác nhân (UC) của quản trị (admin)



Hình 3. 18 Biểu đồ UC của admin

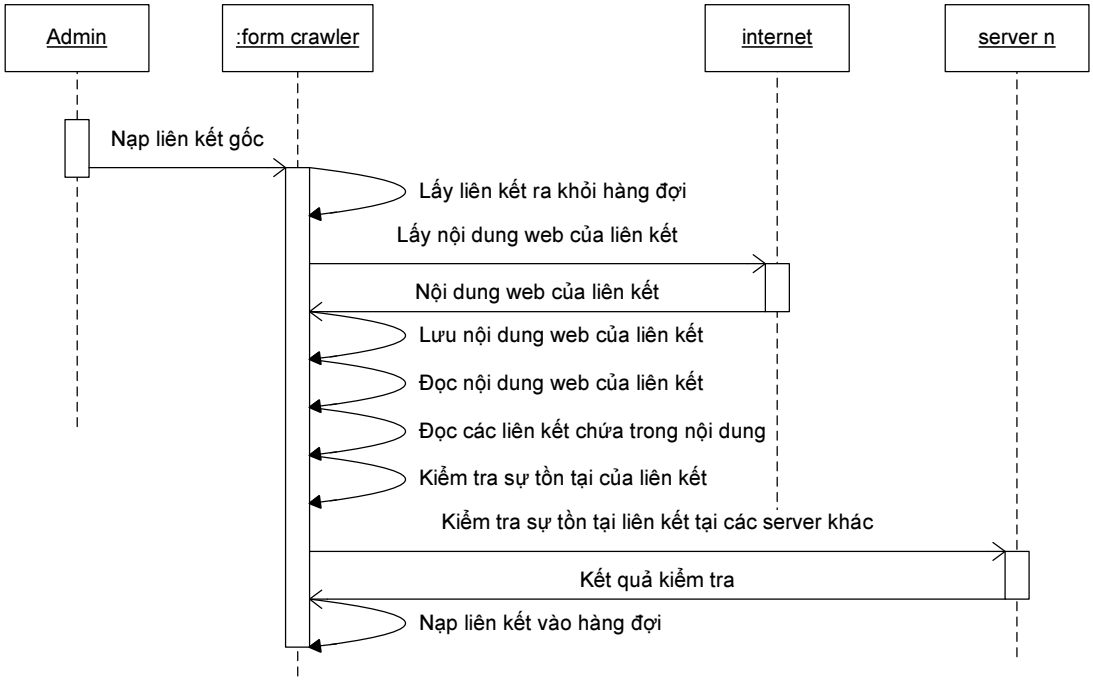
3.4.3 Biểu đồ tuần tự

3.4.3.1Xử lý yêu cầu người dùng



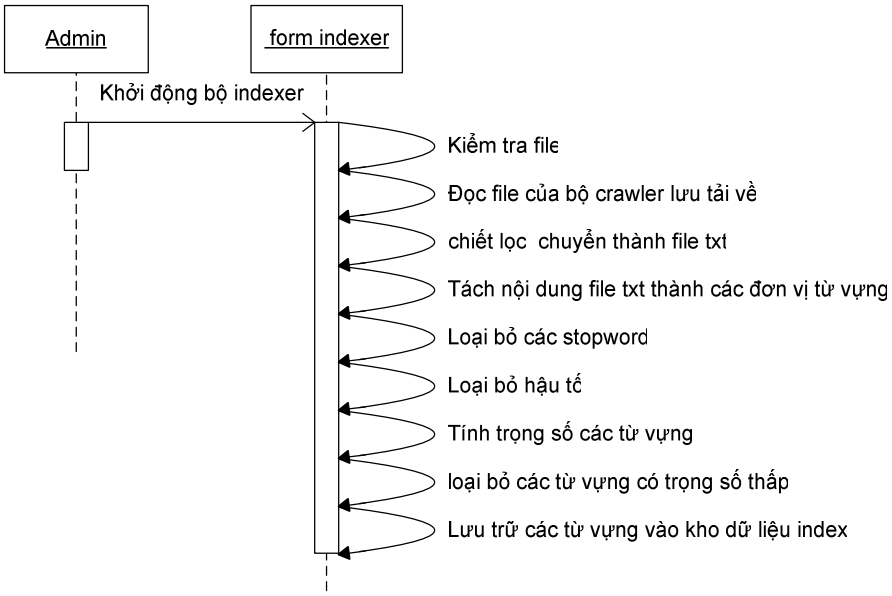
Hình 3. 19 Biểu đồ tuần tự xử lý yêu cầu người dùng

3.4.3.2Truy tìm thông tin tự động (bộ crawler)



Hình 3. 20 Biểu đồ tuần tự truy tìm thông tin tự động

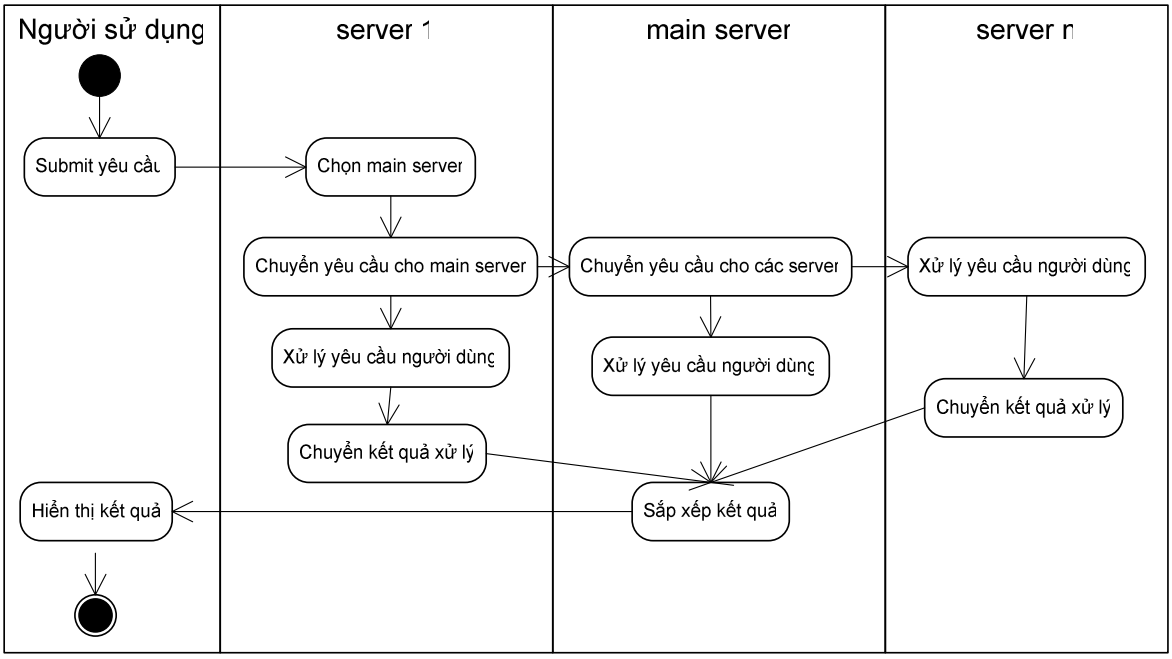
3.4.3.3Lập chỉ mục (bộ indexer)



Hình 3. 21 Biểu đồ tuần tự lập chỉ mục tự động

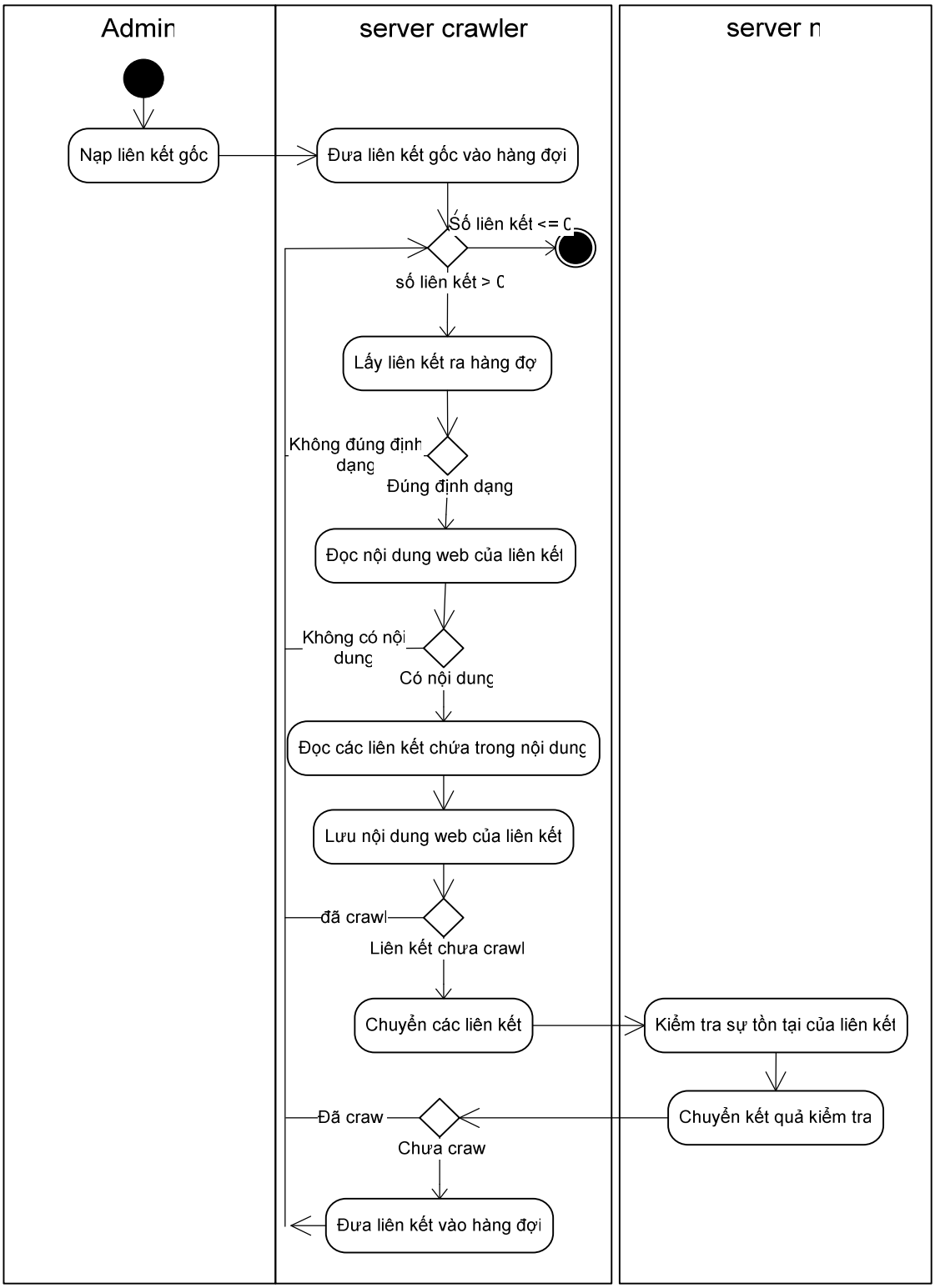
3.4.4 Biểu đồ hoạt động (activity)

3.4.4.1Xử lý yêu cầu người dùng



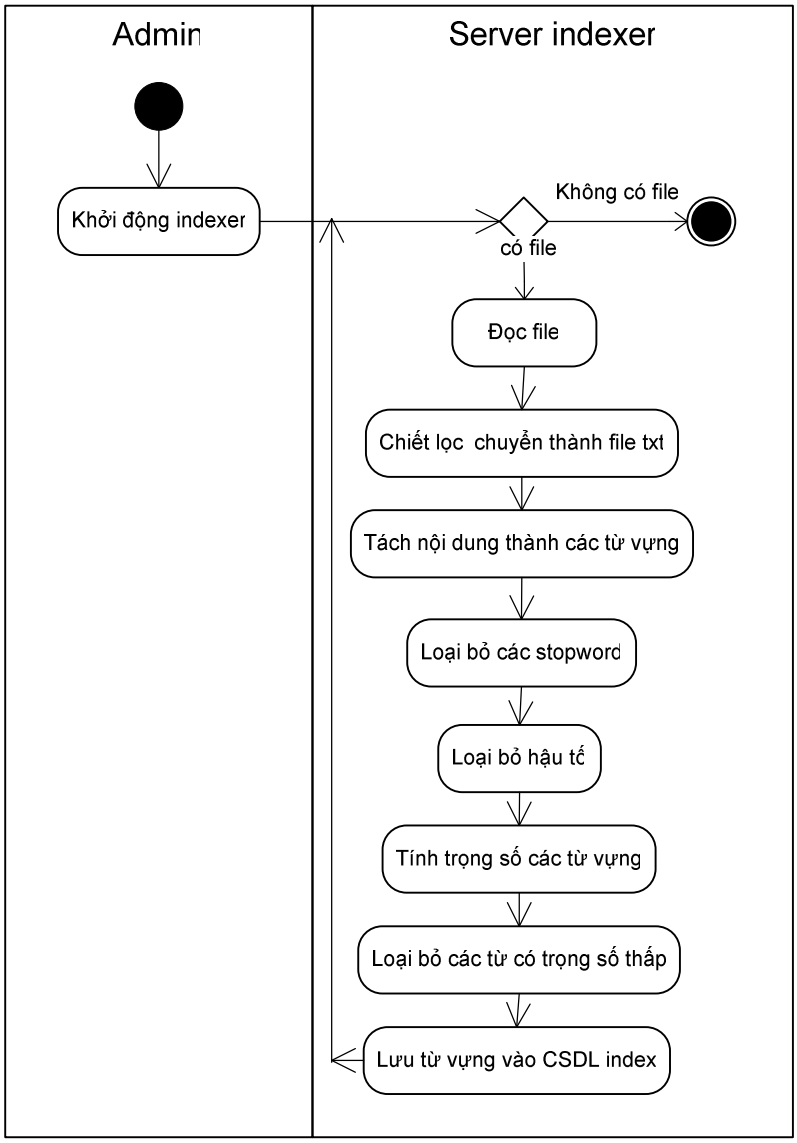
Hình 3. 22 Biểu bồ hoạt động xử lý yêu cầu người dùng

3.4.4.2Truy tìm thông tin tự động (bộ crawler)



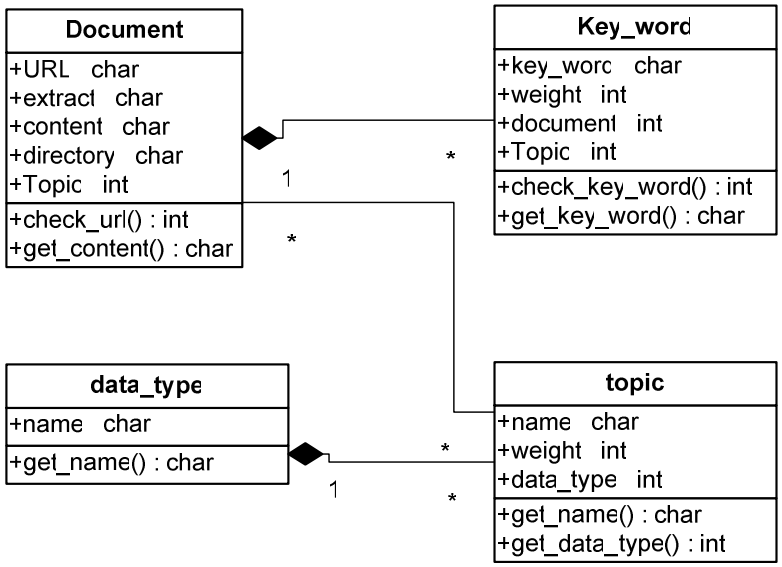
Hình 3. 23 Biểu đồ hoạt động truy tìm thông tin tự động

3.4.4.3Lập chỉ mục tự động (bộ indexer)



Hình 3. 24 Biểu đồ hoạt động lập chỉ mục tự động

3.4.5 Sơ đồ lớp



3.4.6 Các bảng dữ liệu của hệ thống file index

Bảng 3.4. Bảng dữ liệu tbl_document

Field	Data type	Description
ID	Number	Khóa chính
Url	Char(50)	địa chỉ web của document
extract	Char(128)	phần trích đoạn của document
Doc	Char(1024)	thông tin của document
Directory	Char(50)	đường dẫn document
Id_topic	Number	

Bảng 3.5. Bảng từ khóa tbl_key_word

Field	Data type	Description
ID	Number	Khóa chính
Id_doc	Char(128)	Lưu địa chỉ web của document
Key_word	Char(128)	Lưu từ các từ khóa
weight	Number	Trọng số của từ khóa

Bảng 3.6. Bảng chủ đề tbl_topics

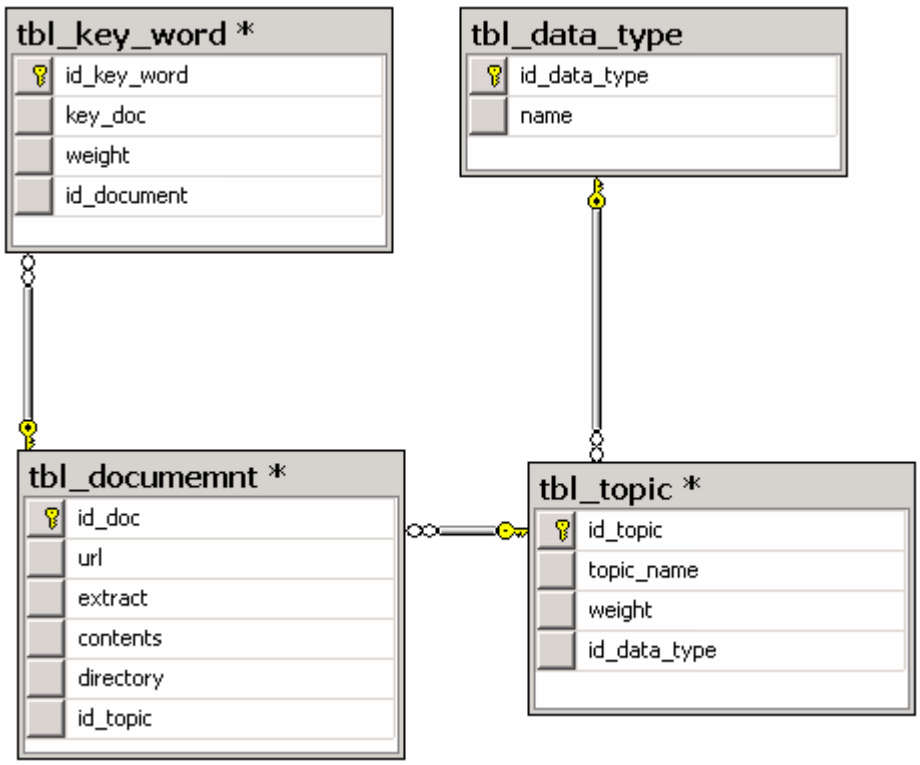
Field	Data type	Description
ID	Number	Khóa chính
Topics_name	Char(128)	Tên chủ đề
weight	Char(128)	Trọng số của chủ đề
Id_data_type	Number	

Bảng 3.7. Bảng loại dữ liệu tbl_data_type

Field	Data type	Description
ID	Number	Khóa chính
Data_type	Char(128)	Tên chủ đề

3.4.7 Xây dựng hệ thống

3.4.7.1Mô hình quan hệ của các bảng dữ liệu



Hình 3. 25 Mô hình quan hệ giữa các bảng dữ liệu

3.4.7.2form submit yêu cầu tìm kiếm

```
<FORM id=searchForm action=SearchController>
<TABLE>
  <TBODY>
    <TR>
      <TD colspan="3">
        <INPUT name=searchWord id=searchWord type=text size="40">
        <INPUT id=doSearch type=submit value=search>
      </TD>
    </TR>
  </TBODY>
</TABLE>
</FORM>
```

3.4.7.3Hiển thị kết quả tìm kiếm

```
<TABLE class="result">
    <TBODY>
    <%
      List searchResult = (List)request.getAttribute("searchResult");
      int resultCount = 0;
      if(null != searchResult){
          resultCount = searchResult.size();
      }
      for(int i = 0; i < resultCount; i++){
          SearchResultBean resultBean = (SearchResultBean)searchResult.get(i);
          String title = resultBean.getHtmlTitle();
          String path  = resultBean.getHtmlPath();
    %>
    <TR>
      <TD class="title"><h3><A href="<%=path %>"><%=title %></A></h3></TD>
    </TR>
    <tr><td><hr /></td></tr>
    <%
      }
    %>

   </TBODY>
</TABLE>
```

3.4.7.4Bộ crawler

```
public class Crawler{
	String url;
	public Crawler(String s){
			url = s;
	}
	public void getDocument(){
	  try{
			URL url = new URL(this.url);
			//String filename = url.
			HttpURLConnection con = (HttpURLConnection) url.openConnection();
			InputStream inputs = con.getInputStream();
			InputStreamReader r = new InputStreamReader(inputs);
			BufferedReader br = new BufferedReader(r);
			String line = null;
			while ((line = br.readLine()) != null) {
			 System.out.println(line);
			}
			System.out.println("TEST: header field = " + con.getHeaderField(2));
			con.disconnect();
	  }catch(MalformedURLException e){
			e.printStackTrace();
	  }catch(IOException e){
			e.printStackTrace();
	  }
	}

	public static void main(String[] args) {
			Crawler c = new Crawler("http://www.yahoo.com");
			c.getDocument();
	}
}
```

3.4.7.5Bộ indexer

Tạo chỉ mục

```
public boolean createIndex() throws IOException{
   if(true == ifIndexExist()){
      return true;
   }
   File dir = new File(dataDir);
   if(!dir.exists()){
      return false;
   }
   File[] htmls = dir.listFiles();
   Directory fsDirectory = FSDirectory.getDirectory(indexDir, true);
   Analyzer  analyzer    = new StandardAnalyzer();
   IndexWriter indexWriter = new IndexWriter(fsDirectory, analyzer, true);
   for(int i = 0; i < htmls.length; i++){
      String htmlPath = htmls[i].getAbsolutePath();

      if(htmlPath.endsWith(".html") || htmlPath.endsWith(".htm")){
               addDocument(htmlPath, indexWriter);
    }
   }
   indexWriter.optimize();
   indexWriter.close();
   return true;

}
```

Thêm từ vựng vào kho index file

```
public void addDocument(String htmlPath, IndexWriter indexWriter){

   HTMLDocParser htmlParser = new HTMLDocParser(htmlPath);
   String path    = htmlParser.getPath();
   String title   = htmlParser.getTitle();
   Reader content = htmlParser.getContent();
   Document document = new Document();
   document.add(new Field("path",path,Field.Store.YES,Field.Index.NO));
   document.add(new Field("title",title,Field.Store.YES,Field.Index.TOKENIZED));
   document.add(new Field("content",content));
   try {
        indexWriter.addDocument(document);
} catch (IOException e) {
        e.printStackTrace();
     }
}
```

Kiểm tra sự tồn tại của từ vựng

```
public boolean ifIndexExist(){
    File directory = new File(indexDir);
    if(0 < directory.listFiles().length){
       return true;
    }else{
       return false;  }
```

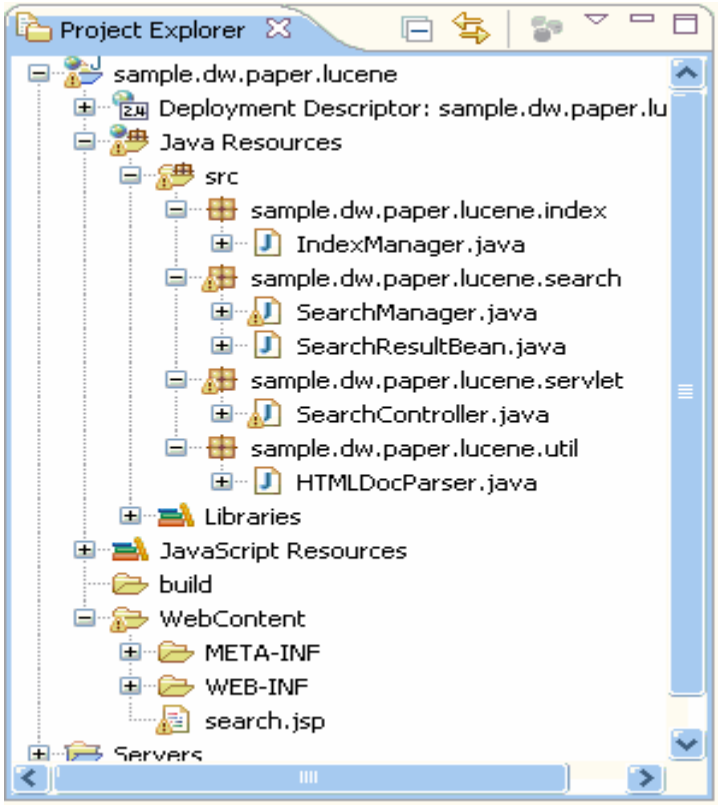
3.4.7.6Bộ search engine

```
 public List search(){
    List searchResult = new ArrayList();
    if(false == indexManager.ifIndexExist()){
    try {
       if(false == indexManager.createIndex()){
          return searchResult;
       }
    } catch (IOException e) {
      e.printStackTrace();
      return searchResult;
    } }
   IndexSearcher indexSearcher = null;
    try{
       indexSearcher = new IndexSearcher(indexManager.getIndexDir());
    }catch(IOException ioe){
       ioe.printStackTrace();
    }
    QueryParser queryParser = new QueryParser("content",analyzer);
    Query query = null;
    try {
       query = queryParser.parse(searchWord);
    } catch (ParseException e) {
      e.printStackTrace();
    }
    if(null != query >> null != indexSearcher){
       try {
          Hits hits = indexSearcher.search(query);
          for(int i = 0; i < hits.length(); i ++){
             SearchResultBean resultBean = new SearchResultBean();
             resultBean.setHtmlPath(hits.doc(i).get("path"));
             resultBean.setHtmlTitle(hits.doc(i).get("title"));
             searchResult.add(resultBean);
          }
       } catch (IOException e) {
          e.printStackTrace();
       } }  return searchResult;   }
```

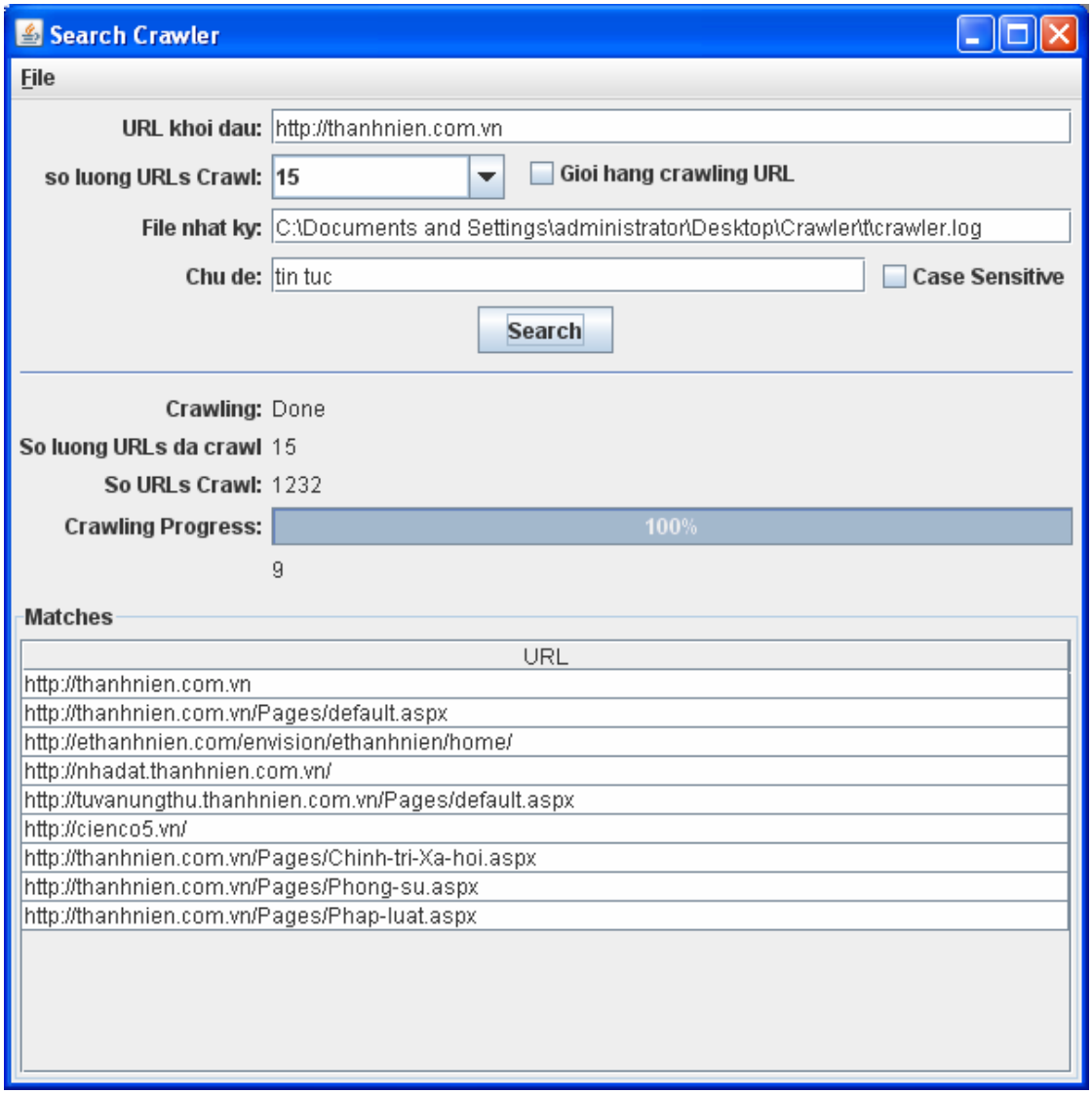
3.4.8 Đề mô chương trình

3.4.8.1 Giao diện tìm kiếm

3.4.8.2 Cấu trúc chương trình của bộ search engine và bộ indexer



3.4.8.3 Bộ crawler



3.4.8.4 Kết quả chương trình

KẾT LUẬN

Qua quá trình nghiên cứu, đề tài đã đạt được một số kết quả nhất định. Bên cạnh đó cũng tồn tại những hạn chế ở một số mặt nào đó. Phần này sẽ đánh giá lại những kết quả trên đồng thời phân tích các khả năng ứng dụng của đề tài để từ đó có hướng phát triển cao hơn.

**** Về kết quả đạt được***

Nội dung chính của đề tài là nghiên cứu về máy tìm kiếm thông tin ứng dụng hệ phân tán đa server để phân tán máy tìm kiếm nhằm tối ưu thời gian xử lý thông tin.

Đối với máy tìm kiếm, đề tài nghiên cứu các bộ phần cấu thành của máy tìm kiếm bộ clawler, bộ indexer, bộ searcher và cấu trúc lưu trữ kho dữ liệu index.

Đối với hệ phân tán, đề tài nghiêm cứu các tính chất của hệ phân tán, các mô hình truyền thông trong hệ phân tán và các giải thuật đồng bộ hóa các tiến trình xử lý trong hệ phân tán.

Đối với việc ứng dụng hệ phân tán để tối ưu thời gian xử lý của máy tìm kiếm, đề tài nghiên cứu, phân tích, đánh giá các nhược điểm của máy tìm kiếm triển khai trên hệ tập trung, đề xuất đưa ra các mô hình hoạt động của máy tìm kiếm triển khai trên hệ phân tán, phân tích hệ thống máy tìm kiếm mới, nghiên cứu, nêu ra một số vấn đề phát sinh và hướng giải quyết khi triển khai máy tìm kiếm mới, xây dựng cơ sở dữ liệu mới cho kho dữ liệu index và chương trình ứng dụng cho máy tìm kiếm mới.

**** Về ưu điểm và nhược điểm của đề tài***

Ưu điểm:

Đề tài đã nghiên cứu xây dựng thành công máy tìm kiếm phân tán, làm giảm thời gian xử lý đán kể cho máy tìm kiếm, giúp máy tìm kiếm cho kết quả nhanh và chính xác hơn.

Phân tích thành lập bảng tiêu chí tối ưu cho máy tìm kiếm.

Phân tích, đưa ra các vấn đề phát sinh dẫn đến sự cố cho máy tìm kiếm và hướng giải quyết các vấn đề đó.

Nhược điểm:

Bên cạnh đó đề tài vẫn còn nhiều hạn chế sau:

Đề tài chưa nghiêm cứu tiêu chí tối ưu thuật toán của máy tìm kiếm.

Đề tài chỉ nghiên cứu ở mức độ tổng quát, chưa đi sâu.

Các lập luận chưa có tính thuyết phục cao.

**** Hướng phát triển***

Tiếp tục nghiên cứu đề xuất thuật toán xử lý tối ưu hơn.

Tiếp tục hoàn chỉnh các mô hình hoạt động của máy tìm kiếm một cách có hiệu quả nhât.

Tiếp hoàn thiện các tiêu chí tối ưu cho máy tìm kiếm.

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1] Phan Tấn Luận (2007), *Nghiên cứu máy tìm kiếm và xây dựng mô phỏng máy tìm kiếm*, Luận văn tốt nghiệp, Đại học Đà Nẵng.

[2] PGS. TS. Lê Văn Sơn (2004), *Hệ tin học phân tán*, Đại học Đà Nẵng.

[3] GS.TS. Nguyễn Thúc Hải (2007), *Giáo trình hệ phân tán*, Đại học Đà Nẵng.

Tiếng nước ngoài

[4] S. Mullender ed (1993), *Distributed Systems*, Addison-Wesley, 2nd ed.,

[5] G. Coulouris, J. Dollimore, T. Kinberg (1994), *Distributed systems : Concept and Design*, Addison-Wesley

[6] A. S. Tanenbaum, M. V. Steen (2002), *Distributed Systems: Principles and Paradigms*, Prentice-Hall.

Trang web

[7] Tài liệu hệ phân tán http://www.wattpad.com/72987-h%E1%BB%87-ph%C3%A2n-t%C3%A1n?p=1

[8] Ví dụ tạo máy tìm kiếm. http://www.ibm.com/developerworks/web/library/wa-lucene2/

[9] Tìm hiểu máy tìm kiếm. http://www.vietseo.net/articles/search-engine/

[10] Định nghĩa máy tìm kiếm. http://vi.wikipedia.org/wiki/M%C3%A1y_truy_t%C3%ACm_d%E1%BB%AF_li%E1%BB%87u

[11] Building a Desktop Search Engine - Inverted Index
http://untiluknow.blogspot.com/2010/04/building-search-engine-inverted-index.html

[12] Search engine: http://www.compass-project.org/docs/2.0.2/reference/html/core-searchengine.html

[13] Ngôn ngữ lập trình Lucene.
http://www.webreference.com/programming/lucene/2.html