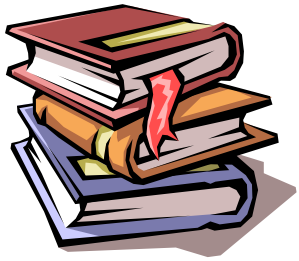
Luận văn tốt nghiệp

Khai phá dữ liệu Web và máy tìm kiếm



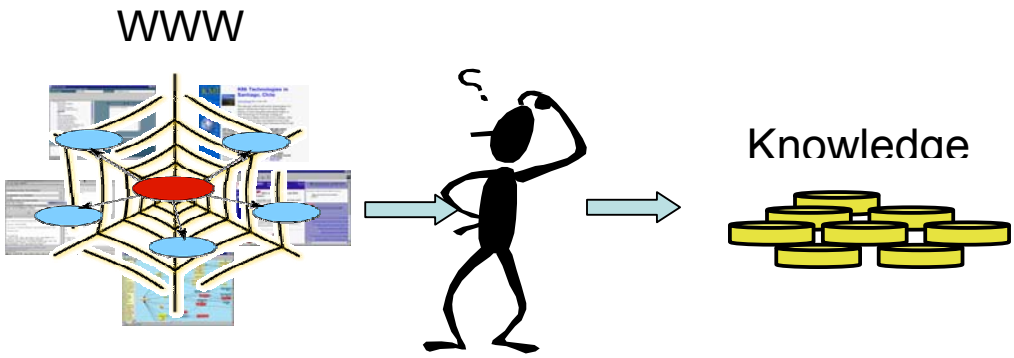
Mục lục

Chương 1. Tổng quan về khai phá dữ liệu Web và máy tìm kiếm

1.1. Khai phá dữ liệu Web

1.1.1. Tổng quan về khai phá dữ liệu Web

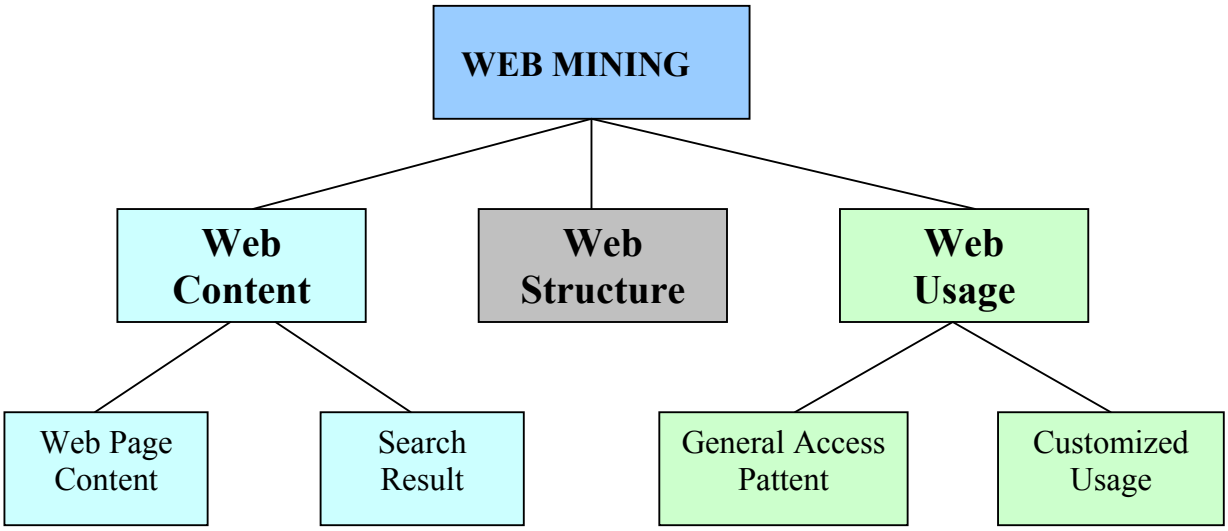
Ngày nay, sự phát triển nhanh chóng của mạng Internet và Intranet đã sinh ra một khối lượng khổng lồ các dữ liệu dạng siêu văn bản (dữ liệu Web). Trong những năm gần đây Intrnet đã trở thành một trong những kênh về khoa học, thông tin kinh tế, thương mại và quảng cáo. Một trong những lý do cho sự phát triển này là chi phí thấp để duy trì một trang Web trên Internet. So sánh với những dịch vụ khác như đăng tin hay quảng cáo trên một tờ báo hay tạp chí, thì một trang Web "đòi" rẻ hơn rất nhiều và cập nhật nhanh chóng hơn tới hàng triệu người dùng khắp mọi nơi trên thế giới. Có thể nói Internet như là cuốn từ điển Bách khoa toàn thư với nội dung và hình thức đa dạng. Nó như một xã hội ảo, nó bao gồm các thông tin về mọi mặt của đời sống kinh tế, xã hội được trình bày dưới dạng văn bản, hình ảnh, âm thanh ...



Hình 1.1: *Khai phá web, công việc không dễ dàng*

Tuy nhiên, Internet là một môi trường đa phương tiện động bao gồm sự kết hợp của các cơ sở dữ liệu không đồng nhất, các chương trình và các giao tiếp người dùng. Rõ ràng, khai phá dữ liệu text chỉ là một lĩnh vực nhỏ trong môi trường này. Khai phá dữ liệu trên Internet, hay thường được gọi là khai phá web ngoài việc cần khai phá được nội dung các trang văn bản, còn phải khai thác được các nguồn lực này cũng như mối quan hệ giữa chúng. Khai phá Web, sự giao thoa giữa khai phá dữ liệu và Word-Wide-Web, đang phát triển mạnh mẽ và bao gồm rất nhiều lĩnh vực nghiên

cứu như trí tuệ nhân tạo, truy xuất thông tin (information retrival) hay các lĩnh vực khác. Các công nghệ Agent-base, truy xuất thông tin dựa trên khái niệm (concept-based), truy xuất thông tin sử dụng case-base reasoning và tính hạng văn bản dựa trên các đặc trưng (features) siêu liên kết... thường được xem là các lĩnh vực nhỏ trong khai phá web. Khai phá Web vẫn chưa được định nghĩa một cách rõ ràng và các chủ đề trong đó vẫn tiếp tục được mở rộng. Tuy vậy, chúng ta có thể hiểu *khai phá web như việc trích ra các thành phần được quan tâm hay được đánh giá là có ích cùng các thông tin tiềm năng từ các tài nguyên hoặc các hoạt động liên quan tới World-Wide Web*[]. Hình 1.2 thể hiện một sự phân loại các lĩnh vực nghiên cứu quen thuộc trong khai phá Web. Người ta thường phân khai phá web thành 3 lĩnh vực chính: khai phá nội dung web (web content mining), khai phá cấu trúc web (web structure mining) và khai phá việc sử dụng web (web usage mining).



Hình 1.2: *Các nội dung trong khai phá Web.*

1.1.2 Các bài toán được đặt ra trong khai phá Web

- Tìm kiếm các thông tin cần thiết: Web quá lớn và quá đa dạng, vì vậy việc tìm được thông tin cần thiết là không đơn giản. Công việc này được giải quyết bởi các máy tìm kiếm.

- Tạo ra các tri thức mới từ các thông tin có sẵn trên Web: Vấn đề này có thể được coi như một vấn đề con của bài toán trên. Ở đây ta mặc định đã có một tập các dữ liệu Web, và ta cần lấy ra được các thông tin hữu ích từ những dữ liệu này.

- Cá nhân hóa các thông tin: Mỗi người dùng thường có các mối quan tâm khác nhau cũng như thích các cách biểu diễn thông tin khác nhau khi tương tác với thế giới Web. Các nghiên cứu về lĩnh vực này sẽ cung cấp các thông tin hữu ích cho những nhà cung cấp thông tin trên Web để họ có thể đạt được mục đích của mình.

- Tìm hiểu về những người tiêu thụ sản phẩm cũng như về cá nhân người dùng: Các nghiên cứu này phục vụ đắc lực để giải quyết vấn đề ở trên. Nó tìm hiểu những điều mà người tiêu dùng muốn và làm. Điều đó sẽ giúp chuyên biệt hóa thông tin cho từng người dùng, giúp thiết kế và quản lý web site một cách hiệu quả, cũng như các vấn đề liên quan tới maketing.

1.1.3 Các lĩnh vực của khai phá dữ liệu Web

1.1.3.1 Khai phá nội dung Web (Web content mining):

Phần lớn các tri thức của World-Wide Web được chứa trong nội dung văn bản. Khai phá nội dung web là các quá trình xử lý để lấy ra các tri thức từ nội dung các trang văn bản hoặc mô tả của chúng. Có hai chiến lược khai phá nội dung web: một là khai phá trực tiếp nội dung của trang web, và một là nâng cao khả năng tìm kiếm nội dung của các công cụ khác như máy tìm kiếm.

- Web Page summarization: liên quan tới việc truy xuất các thông tin từ các văn bản có cấu trúc, văn bản siêu liên kết, hay các văn bản bán cấu trúc. Lĩnh vực này liên quan chủ yếu tới việc khai phá bản thân nội dung các văn bản.

- Search engine result summarization: Tìm kiếm trong kết quả. Trong các máy tìm kiếm, sau khi đã tìm ra những trang Web thoả mãn yêu cầu người dùng, còn một công việc không kém phần quan trọng, đó là phải sắp xếp, chọn lọc kết quả theo mức độ hợp lệ với yêu cầu người dùng. Quá trình này thường sử dụng các thông tin như tiêu đề trang, URL, content-type, các liên kết trong trang web... để tiến hành phân lớp và đưa ra tập con các kết quả tốt nhất cho người dùng.

1.1.3.2. Khai phá cấu trúc web (web structure mining):

Nhờ vào các kết nối giữa các văn bản siêu liên kết, World-Wide Web có thể chứa đựng nhiều thông tin hơn là chỉ các thông tin ở bên trong văn bản. Ví dụ, các liên kết trỏ tới một trang web chỉ ra mức độ quan trọng của trang web đó, trong khi các liên kết đi ra từ một trang web thể hiện các trang có liên quan tới chủ đề đề cập trong trang hiện tại. Và nội dung của khai phá cấu trúc Web là các quá trình xử lý nhằm rút ra các tri thức từ cách tổ chức và liên kết giữa các tham chiếu của các trang web.

1.1.3.3 Khai phá sử dụng web (web usage mining).

Khai phá sử dụng web (web usage mining) hay khai phá hồ sơ web (web log mining) là việc xử lý để lấy ra các thông tin hữu ích trong các hồ sơ truy cập Web. Thông thường các web server thường ghi lại và tích lũy các dữ liệu về các tương tác của người dùng mỗi khi nó nhận được một yêu cầu truy cập. Việc phân tích các hồ sơ truy cập web của các web site khác nhau sẽ dự đoán các tương tác của người dùng khi họ tương tác với Web cũng như tìm hiểu cấu trúc của Web, từ đó cải thiện các thiết kế của các hệ thống liên quan. Có hai xu hướng chính trong khai phá sử dụng web là General Access Pattern Tracking và Customizied Usage tracking.

- General Access Pattern tracking: phân tích các hồ sơ web để biết được các mẫu và các xu hướng truy cập. Các phân tích này có thể giúp cấu trúc lại các site trong các phân nhóm hiệu quả hơn, hay xác định các vị trí quảng cáo hiệu quả nhất, cũng như gắn các quảng cáo sản phẩm nhất định cho những người dùng nhất định để đạt được hiệu quả cao nhất...

- Cusomized Usage tracking: phân tích các xu hướng cá nhân. Mục đích là để chuyên biệt hóa các web site cho các lớp đối tượng người dùng. Các thông tin được hiển thị, độ sâu của cấu trúc site và định dạng của các tài nguyên, tất cả đều có thể chuyên biệt hóa một cách tự động cho mỗi người dùng theo thời gian dựa trên các mẫu truy cập của họ.

1.1.4. Khó khăn

World Wide Web là một hệ thống rất lớn phân bố rộng khắp, cung cấp thông tin trên mọi lĩnh vực khoa học, xã hội, thương mại, văn hóa,... Web là một nguồn tài nguyên giàu có cho Khai phá dữ liệu. Những quan sát sau đây cho thấy Web đã đưa ra những thách thức lớn cho công nghệ Khai phá dữ liệu [1].

1.1.4.1 Web dường như quá lớn để tổ chức thành kho dữ liệu phục vụ Dataming

Các CSDL truyền thống thì có kích thước không lớn lắm và thường được lưu trữ ở một nơi, trong khi đó kích thước Web rất lớn, tới hàng terabytes và thay đổi liên tục, không những thế còn phân tán trên rất nhiều máy tính khắp nơi trên thế giới. Một vài nghiên cứu về kích thước của Web đã đưa ra các số liệu như sau: Hiện nay trên Internet có khoảng hơn một tỷ các trang Web được cung cấp cho người sử dụng., giả sử kích thước trung bình của mỗi trang là 5-10Kb thì tổng kích thước của nó ít nhất là

khoảng 10 terabyte. Còn tỷ lệ tăng của các trang Web thì thật sự gây ấn tượng. Hai năm gần đây số các trang Web tăng gấp đôi và còng tiếp tục tăng trong hai năm tới. Nhiều tổ chức và xã hội đặt hầu hết những thông tin công cộng của họ lên Web. Như vậy việc xây dựng một kho dữ liệu (datawarehouse) để lưu trữ, sao chép hay tích hợp các dữ liệu trên Web là gần như không thể.

1.1.4.2. Độ phức tạp của trang Web lớn hơn rất nhiều so với những tài liệu văn bản truyền thống khác

Các dữ liệu trong các CSDL truyền thống thì thường là loại dữ liệu đồng nhất (về ngôn ngữ, định dạng,…), còn dữ liệu Web thì hoàn toàn không đồng nhất. Ví dụ về ngôn ngữ dữ liệu Web bao gồm rất nhiều loại ngôn ngữ khác nhau (Cả ngôn ngữ diễn tả nội dung lẫn ngôn ngữ lập trình), nhiều loại định dạng khác nhau (Text, HTML, PDF, hình ảnh âm thanh,…), nhiều loại từ vựng khác nhau (Địa chỉ Email, các liên kết (links), các mã nén (zipcode), số điện thoại).

Nói cách khác, trang Web thiếu một cấu trúc thống nhất. Chúng được coi như một thư viện kỹ thuật số rộng lớn, tuy nhiên con số khổng lồ các tài liệu trong thư viện thì không được sắp xếp tuân theo một tiêu chuẩn đặc biệt nào, không theo phạm trù, tiêu đề, tác giả, số trang hay nội dung,... Điều này là một thử thách rất lớn cho việc tìm kiếm thông tin cần thiết trong một thư viện như thế.

1.1.4.3. Web là một nguồn tài nguyên thông tin có độ thay đổi cao

Web không chỉ có thay đổi về độ lớn mà thông tin trong chính các trang Web cũng được cập nhật liên tục. Theo kết quả nghiên cứu [], hơn 500.000 trang Web trong hơn 4 tháng thì 23% các trang thay đổi hàng ngày, và khoảng hơn 10 ngày thì 50% các trang trong tên miền đó biến mất, nghĩa là địa chỉ URL của nó không còn tồn tại nữa. Tin tức, thị trường chứng khoán, các công ty quản cáo và trung tâm phục vụ Web thường xuyên cập nhật trang Web của họ. Thêm vào đó sự kết nối thông tin và sự truy cập bản ghi cũng được cập nhật.

1.1.4.4. Web phục vụ một cộng đồng người dùng rộng lớn và đa dạng

Internet hiện nay nối với khoảng 50 triệu trạm làm việc [1], và cộng đồng người dùng vẫn đang nhanh chóng lan rộng. Mỗi người dùng có một kiến thức, mối quan tâm, sở thích khác nhau. Nhưng hầu hết người dùng không có kiến thức tốt về cấu trúc mạng thông tin, hoặc không có ý thức cho những tìm kiếm, rất dễ bị "lạc" khi

đang "mò mẫm" trong "bóng tối" của mạng hoặc sẽ chán khi tìm kiếm mà chỉ nhận những mảng thông tin không mấy hữu ích.

1.1.4.5. Chỉ một phần rất nhỏ của thông tin trên Web là thực sự hữu ích.

Theo thống kê, 99% của thông tin Web là vô ích với 99% người dùng Web. Trong khi những phần Web không được quan tâm lại bị búi vào kết quả nhận được trong khi tìm kiếm. Vậy thì ta cần phải khai phá Web như thế nào để nhận được trang web chất lượng cao nhất theo tiêu chuẩn của người dùng?

Như vậy chúng ta có thể thấy các điểm khác nhau giữa việc tìm kiếm trong một CSDL truyền thống với vviệc tìm kiếm trên Internet. Những thách thức trên đã đẩy mạnh việc nghiên cứu khai phá và sử dụng tài nguyên trên Internet

1.1.5. Thuận lợi

Bên cạnh những thử thách trên, công việc khai phá Web cũng có những thuận lợi:

1. Web bao gồm không chỉ có các trang mà còn có cả các hyperlink trỏ từ trang này tới trang khác. Khi một tác giả tạo một hyperlink từ trang của ông ta tới một trang A có nghĩa là A là trang có hữu ích với vấn đề đang bàn luận. Nếu trang A càng nhiều Hyperlink từ trang khác trỏ đến chứng tỏ trang A quan trọng. Vì vậy số lượng lớn các thông tin liên kết trang sẽ cung cấp một lượng thông tin giàu có về mối liên quan, chất lượng, và cấu trúc của nội dung trang Web, và vì thế là một nguồn tài nguyên lớn cho khai phá Web.

2. Một máy chủ Web thường đăng ký một bản ghi đầu vào (Weblog entry) cho mọi lần truy cập trang Web. Nó bao gồm địa chỉ URL, địa chỉ IP, timestamp. Dữ liệu Weblog cung cấp lượng thông tin giàu có về những trang Web động. Với những thông tin về địa chỉ URL, địa chỉ IP,… một cách hiển thị đa chiều có thể được cấu trúc nên dựa trên CSDL Weblog. Thực hiện phân tích OLAP đa chiều có thể đưa ra N người dùng cao nhất, N trang Web truy cập nhiều nhất, và khoảng thời gian nhiều người truy cập nhất, xu hướng truy cập Web.

1.2 Tổng quan về máy tìm kiếm

1.2.1 Nhu cầu

Như đã đề cập ở phần trên, Internet là một kho thông tin khổng lồ và phức tạp. Thông tin trên các trang Web đa dạng về mặt nội dung cũng như hình thức. Tuy nhiên

cùng với sự đa dạng và số lượng lớn thông tin như vậy đã nảy sinh vấn đề quá tải thông tin. Cùng với sự thay đổi và phát triển hàng ngày hàng giờ về nội dung cũng như số lượng của các trang Web trên Internet thì vấn đề tìm kiếm thông tin đối với người sử dụng lại ngày càng khó khăn. Đối với mỗi người dùng chỉ một phần rất nhỏ thông tin là có ích, chẳng hạn có người chỉ quan tâm đến trang Thể thao, Văn hóa mà không mấy khi quan tâm đến Kinh tế. Người ta không thể tìm tự kiếm địa chỉ trang Web chứa thông tin mà mình cần, do vậy đòi hỏi cần phải có một trình tiện ích quản lý nội dung của các trang Web và cho phép tìm thấy các địa chỉ trang Web có nội dung giống với yêu cầu của người tìm kiếm.

Định nghĩa []:Máy tìm kiếm (search engine) là một hệ thống được xây dựng nhằm tiếp nhận các yêu cầu tìm kiếm của người dùng (thường là một tập các từ khóa), sau đó phân tích yêu cầu này và tìm kiếm thông tin trong cơ sở dữ liệu được tải xuống từ Web và đưa ra kết quả là các trang web có liên quan cho người dùng.

Cụ thể, người dùng gửi một truy vấn, dạng đơn giản nhất là một danh sách các từ khóa, và máy tìm kiếm sẽ làm việc để trả lại một danh sách các trang Web có liên quan hoặc có chứa các từ khóa đó. Phức tạp hơn, thì truy vấn là cả một văn bản hoặc một đoạn văn bản hoặc nội dung tóm tắt của văn bản. Một số máy tìm kiếm điển hình hiện nay: Yahoo, Google, Alvista,...

1.2.2 Cơ chế hoạt động của máy tìm kiếm.

Một máy tìm kiếm có thể được xem như là một ví dụ của hệ thống truy xuất thông tin Information Retrival (IR). Một hệ thống truy xuất thông tin IR thường tập trung vào việc cải thiện hiệu quả thông tin được lấy ra bằng cách sử dụng việc đánh chỉ số dựa trên các từ khóa (term-base indexing)[8,11] và kỹ thuật tổ chức lại các câu truy vấn (query refomulation technique)[32]. Quá trình xử lý các văn bản dựa trên từ khóa ban đầu trích ra các từ khóa trong văn bản sử dụng một từ điển được xây dựng trước, một tập các từ dừng, và các qui tắc (stemming rule)[10] để chuyển các hình thái của từ về dạng từ gốc. Sau khi các từ khóa đã được lấy ra, và thường sử dụng phương pháp TF-IDF (hoặc biến thể của nó) [31,33] để xác định mức độ quan trọng của các từ khóa. Do đó, một văn bản có thể được biểu diễn bởi một tập các từ khóa và độ quan trọng của chúng. Mức độ tương tự đo được giữa một câu truy vấn và một văn bản chính bằng tích trực tiếp tích direct product giữa hai vector các từ khóa tương ứng. Để thể hiện mức độ hợp lệ của các văn bản và câu truy vấn, các văn bản được lấy ra được

biểu diễn dưới dạng một danh sách được xếp hạng dựa trên độ đo mức độ tương tự giữa chúng và câu truy vấn. Intelligent Internet Doc Organization

Các máy tìm kiếm hiện nay sử dụng các công nghệ IR rất đa dạng. Sự khác nhau giữa chúng liên quan tới vấn đề đánh chỉ số, cách biểu diễn văn bản, cách thức truy vấn và thực thi.

Quá trình đánh chỉ số: Các máy tìm kiếm thu thập các trang văn bản HTML trên Internet theo yêu cầu của người dùng hoặc một cách tự động sử dụng các Internet robot (hay còn gọi là spider hoặc crawler). Giống như một hệ thống IR điển hình, các máy tìm kiếm sẽ đánh chỉ số các văn bản này theo từ hoặc cụm từ theo cách ta có thể dễ dàng truy xuất thông tin. Dựa vào định dạng bán cấu trúc của trang HTML, các máy tìm kiếm có thể xác định trọng số cho các từ khóa này dựa vào ý nghĩa của các thẻ.

Cách thức biểu diễn (representation): Phần lớn các máy tìm kiếm sử dụng cách đánh chỉ số full text để nhanh chóng đo mức độ tương tự giữa câu truy vấn và trang web, trong đó các văn bản được biểu diễn bởi một tập các cặp từ khóa – trọng số giống như trong các hệ thống IR điển hình.

Cách truy vấn (querying): Các công cụ tìm kiếm sử dụng một số hàm số để tinh lọc trong số rất lớn các kết quả tìm kiếm. Ví dụ phần lớn các máy tìm kiếm cung cấp các toán tử Boolean để đưa ra các kết quả chính xác hơn. Các hàm số khác chẳng hạn tìm kiếm chính xác theo cụm từ, sắp xếp các trang web theo các site, hay hạn chế tìm kiếm theo các site nhất định cũng rất hiệu quả trong việc tinh lọc các kết quả tìm kiếm.

Thực thi (implementation): Các máy tìm kiếm cũng như các hệ thống thư mục chủ đề (topic directory) đều phải đương đầu với bản chất động của môi trường Internet ngược hẳn với bản chất tĩnh của các hệ thống truy xuất thông tin IR. Các trang web được tạo ra, sửa đổi và xóa bỏ một cách thường xuyên, điều này đòi hỏi các hệ thống phải được trang bị một cấu trúc lưu trữ động và một cơ chế đánh chỉ số hiệu quả. Việc thực thi các Internet robot thông minh cũng là một thử thách khác trong việc thu thập các trang web từ Internet.

1.2.3 Cấu trúc điển hình của một máy tìm kiếm.

Một máy tìm kiếm điển hình thường gồm các thành phần:

- **Module crawler**: đi theo các liên kết trên các trên Web để thu thập nội dung các trang Web một cách tự động và lưu vào các kho chứa cục bộ.

- **Module index (đánh chỉ mục):** module này có nhiệm vụ duyệt nội dung các trang web đã được tải về, phân lớp, tính hạng cho các trang này lưu trữ trong các cấu trúc thuận tiện cho quá trình tìm kiếm.

- **Module tìm kiếm**: truy xuất cơ sở dữ liệu để trả về danh sách các tài liệu thỏa mãn một yêu cầu của người dùng, đồng thời sắp xếp các tài liệu này theo mức độ hợp lệ so với câu truy vấn.

- **Module giao diện người máy:** liên quan tới việc giao tiếp với người dùng. Nhiệm vụ module này là nhận câu truy vấn của người dùng, gửi cho module tìm kiếm, đồng thời nhận kết quả trả về của quá trình tìm kiếm và hiển thị cho người sử dụng.

Chương 2. Module Crawler trong các máy tìm kiếm

2.1 Tổng quan

Kích thước quá lớn và bản chất thay đổi không ngừng của Web đã đặt ra nhu cầu to lớn trong việc hỗ trợ và cập nhật một cách không ngừng các hệ thống trích chọn các thông tin dựa trên nền Web. Crawler đáp ứng được nhu cầu này bằng cách đi theo các siêu liên kết trên các trang Web để download một cách tự động nội dung các trang Web.

Định nghĩa[]: Web crawler là các chương trình khai thác sơ đồ cấu trúc của Web bằng cách chuyển từ trang web này sang trang web khác.

Ban đầu, động cơ chủ yếu thúc đẩy việc thiết kế các web crawler là việc lấy ra nội dung các trang web và thêm chúng hoặc thể hiện của chúng vào các kho chứa cục bộ. Các kho chứa này, sau đó sẽ đáp ứng các ứng dụng cụ thể chẳng hạn một hệ thống tìm kiếm trên Web. Ở dạng đơn giản nhất, một chương trình crawler sẽ bắt đầu từ một địa chỉ nguồn khởi đầu nào đó và sử dụng các liê n kết ngoài trong trang web đó để mở rộng ra các trang tiếp theo. Quá trình này tiếp tục với các trang web mới, các trang này lại cung cấp các liên kết ngoài khác để đi theo. Cứ như vậy cho tới khi đạt tới một số lượng trang web xác định hoặc một mục tiêu nào đó đạt được. Phía sau sự mô tả một cách đơn giản này là một mảng các vấn đề phức tạp có liên quan như việc kết nối mạng, các tiêu chuẩn về một URL, việc duyệt các trang HTML và cách thức để giao tiếp với các Server ở xa. Trên thực tế, các thế hệ web crawler gần đây, có thể coi là một trong những phần phức tạp nhất của hệ thống mà nó đi kèm.

Nếu môi trường Web là một tập các trang web tĩnh cố định, thì chúng ta sẽ ít khi phải sử dụng các chương trình Crawler. Một khi các trang web đã được lưu vào kho chứa (ví dụ như một cơ sở dữ liệu của hệ thống tìm kiếm), ta sẽ chẳng còn lý do nào để sử dụng modul crawler. Tuy nhiên, môi trường Web là một thực thể động, với các không gian con thay đổi theo các xu hướng khác nhau và thường là với tốc độ rất nhanh. Do đó chúng ta luôn cần sử dụng các crawler để giúp các ứng dụng được cập nhật bằng cách cập nhật nội dung mới của các trang web, xóa bỏ hoặc sửa đổi nội dung cũ. Các hệ thống tìm kiếm thường cố gắng thu thập được càng nhiều trang web càng tốt. Các hệ thống này thường sử dụng Web crawler để bảo trì cơ sở dữ liệu được đánh chỉ mục của chúng, cân bằng cái giá của quá trình crawling và đánh chỉ mục với hàng triệu truy vấn mà hệ thống nhận được. Module crawler của các hệ thống này

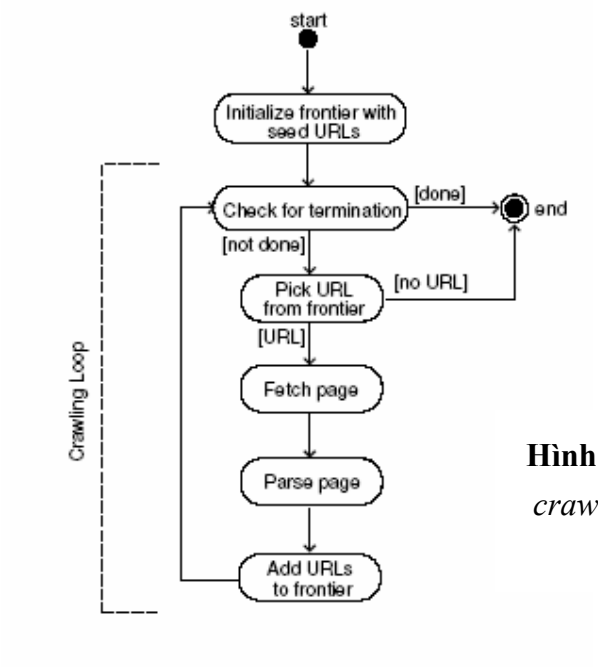
thường có xu hướng và mục tiêu chính là download hết các trang web mà nó gặp. Ngược lại, các crawler khác lại chỉ chọn một số trang web để tải và duyệt trong số rất nhiều các trang web nó gặp, các crawler này được gọi là các crawler có lựa chọn *preferential crawler* hoặc crawler dựa trên kinh nghiệm. Chúng được sử dụng để xây dựng các kho dữ liệu có chủ điểm, tự động hóa các nguồn lực khai phá và đáp ứng cho các đại lý phần mềm. Các crawler có lựa chọn được xây dựng để lấy ra các trang web theo một chủ đề xác định được gọi là các crawler theo chủ đề *topic crawler* hoặc crawler tập trung *focused crawler*.

Có một số khía cạnh của các topic crawler, đang được tập trung nghiên cứu. Một câu hỏi then chốt đã đang thu hút sự quan tâm của các nhà nghiên cứu là: Làm thế nào để đạt được tính chất lựa chọn của crawler. Các vấn đề phụ thuộc nhiều vào ngữ cảnh như mục tiêu của ứng dụng cha (mà crawler là một thành phần) hoặc các tín hiệu ngữ nghĩa trong trang web cũng như những đặc trưng (features) của các lược đồ được xây dựng từ các trang web cũng đã được xem xét. Thêm vào đó, các crawler cũng sử dụng các cơ chế khác nhau trong việc xử lý các yếu tố này.

Một khía cạnh quan trọng thứ hai cần xem xét khi nghiên cứu các crawler, đặc biệt là các topical crawler, đó là bản chất của nhiệm vụ crawl (duyệt web). Các tính chất của việc crawl như là các truy vấn hay là các từ khóa được cung cấp như là các đầu vào cho các crawler, các hồ sơ người dùng user-profile, hay các thuộc tính của trang web cần tải (các trang tương tự, các trang web phổ biến ....) có thể dẫn tới các thay đổi đáng kể trong việc thiết kế và thực thi các crawler. Các tác vụ có thể bị ràng buộc bởi các tham số như số lượng cực đại các trang web cần nạp hay dung lượng bộ nhớ có thể... Do đó, một nhiệm vụ crawling có thể được xem như một bài toán tìm kiếm bị ràng buộc bởi nhiều mục tiêu (multi-objective). Tuy nhiên, do sự đa dạng của các hàm mục tiêu cộng với sự thiếu các hiểu biết chính xác về không gian tìm kiếm làm cho vấn đề càng trở nên phức tạp. Hơn nữa, một chương trình crawler có thể sẽ phải giải quyết các vấn đề về tối ưu hóa như tối ưu toàn cục và tối ưu cục bộ.

Phần đầu của chương này giới thiệu cấu trúc cơ bản của một chương trình crawler và từ đó giới thiệu những khái niệm cơ bản về Web crawling. Tiếp đó, chúng tôi giới thiệu một số thuật toán crawling phổ biến. Phần tiếp theo nữa đề cập tới các phương pháp hiện tại được sử dụng để đánh giá và so sánh việc thực thi của các crawler khác nhau.

2.2 Cấu trúc cơ bản của một crawler



Hình 2.1: *sơ đồ của một crawler tuần tự cơ bản.*

Hình 2.1 biểu diễn đồ thị của một crawler tuần tự cơ bản. Một chương trình crawler bao gồm một danh sách các URL chưa được thăm gọi là frontier. Danh sách này được khởi tạo bởi các URL hạt nhân đã được cung cấp bởi người dùng hoặc các chương trình khác. Mỗi vòng lặp crawling bao gồm: lấy ra URL cần được index tiếp theo từ frontier, nạp trang web tương ứng với URL đó bằng giao thức HTTP, duyệt trang web vừa tải về để lấy ra các từ URL và các thông tin mà ứng dụng cần, và cuối cùng là thêm các trang URL chưa được thăm vào frontier. Trước khi các URL được thêm vào frontier chúng sẽ được gán cho một độ đo thể hiện đánh giá hiệu quả khi thăm trang web tương ứng với URL đó. Quá trình crawling có thể kết thúc khi một số lượng nhất định các trang web đã được tải. Nếu chương trình crawler đã sẵn sàng để duyệt một trang web khác và trạng thái của frontier là rỗng, một tín hiệu trạng thái kết thúc (dead-end) sẽ được gửi cho crawler. Chương trình crawler sẽ không có trang web mới để tải và dừng lại.

Công việc crawling có thể được xem như một bài toán duyệt đồ thị. Toàn bộ thế giới Web được xem như một đồ thị lớn với các nút là các trang web và các liên kết là các đường đi (cạnh). Một crawler bắt đầu tại một vài nút hạt nhân và sau đó đi theo các cạnh để tới các nút khác. Quá trình tải một trang web và trích ra các liên kết trong nó tương tự như việc mở rộng một nút trong bài toán tìm kiếm trên đồ thị. Một crawler

có chủ điểm cố gắng đi theo các cạnh mà được kỳ vọng là dẫn tới các vị trí trong đồ thị là hợp lệ với chủ điểm đó.

2.2.1 Frontier

Phần frontier là danh sách các công việc cần làm của một crawler, nó chứa các URL của các trang web chưa được thăm. Trong thuật ngữ tìm kiếm đồ thị, frontier là một danh sách mở các nút chưa được mở rộng (chưa được thăm). Mặc dù có thể có nhu cầu phải lưu các frontier lên đĩa đối với các crawler rất lớn, ở đây để đơn giản hóa chúng tôi chỉ giới thiệu frontier như là các cấu trúc dữ liệu trong bộ nhớ trong. Dựa trên dung lượng bộ nhớ có thể, ta có thể quyết định kích thước cực đại của frontier. Dựa vào dung lượng lớn của bộ nhớ máy tính ngày nay, kích thước một frontier vào khoảng 100,000 URL không phải là hiếm. Do các frontier chỉ có kích thước giới hạn ta cần có một cơ chế để quyết định URL nào cần bị bỏ qua khi số lượng url trên frontier đạt tới giới hạn đó. Cần phải lưu ý rằng frontier có thể bị đầy nhanh hơn nhiều so với số lượng trang web được duyệt. Ta có thể có tới khoảng 60,000 URL trong frontier khi mới duyệt được khoảng 10,000 trang web do trung bình có khoảng 7 liên kết trong một trang web[29].

Frontier có thể được thực thi như một hàng đợi FIFO nếu ta muốn xây dựng một crawler theo duyệt chiều rộng (breadth-first) để duyệt Web theo chiến lược mù (blindly). URL cần được duyệt tiếp theo được lấy từ đỉnh của hàng đợi và các URL mới được thêm vào cuối hàng đợi. Do kích thước hạn chế của frontier, chúng ta cần phải đảm bảo là không thêm các url lặp lại vào hàng đợi. Do vậy một cơ chế tìm kiếm tuyến tính để tìm ra một URL mới được trích ra từ nội dung của URL đang được duyệt có nằm trên frontier chưa là rất cần thiết. Một giải pháp được đưa ra là định vị một lượng bộ nhớ cần thiết để duy trì một bảng băm riêng (với khóa là URL) để lưu giữ mỗi một URL của frontier để thuận lợi cho việc tìm kiếm. Bảng băm này phải được giữ đồng bộ với frontier thực sự. Một giải pháp khác tốn nhiều thời gian hơn là duy trì bản thân hàng đợi đó như một bảng băm (cũng với khóa là URL). Điều này cung cấp một cách tìm kiếm nhanh chóng để tránh việc lưu lặp lại các URLs. Tuy nhiên, mỗi lần crawler cần một URL để duyệt, nó cần phải tìm kiếm và lấy ra URL mới được đưa vào frontier gần đây nhất. Nếu bộ nhớ không phải là vấn đề nổi cộm bằng tốc độ, giải pháp thứ nhất có thể sẽ tốt hơn. Một khi frontier đạt tới kích thước tối đa, thì crawler theo chiều rộng chỉ có thể thêm duy nhất một URL chưa được thăm từ mỗi trang web đã được duyệt.

Nếu phần frontier được thực thi như một hàng đợi ưu tiên chúng ta có một crawler ưu tiên hay còn gọi là crawler tốt nhất đầu tiên (best-first crawler). Hàng đợi ưu tiên có thể là một mảng động luôn được sắp xếp theo độ đo được đánh giá của các URL chưa được thăm. Tại mỗi bước, URL tốt nhất được lấy ra ở đầu hàng đợi. Một khi trang web tương ứng được nạp, các URL outgoing từ nó được lấy ra và được đánh giá dựa trên một số kinh nghiệm. Sau đó chúng lại được thêm vào frontier tại các vị trí phụ thuộc vào độ đo đó. Chúng ta có thể tránh việc thêm một cách lặp lại các URL trong frontier bằng cách giữ một bảng băm riêng biệt để tìm kiếm. Khi frontier đạt tới kích thước tối đa là MAX, chỉ có MAX URL tốt nhất được giữ lại trong frontier.

Nếu chương trình crawler nhận thấy frontier là rỗng trong khi nó cần URL tiếp theo để duyệt, quá trình crawling sẽ ngừng lại. Với một giá trị MAX lớn và một vài URL hạt nhân thường frontier rất hiếm khi đạt tới trạng thái rỗng.

Tại một số thời điểm, một crawler có thể gặp một bẫy nhện (spider trap) mà dẫn nó tới một số lượng lớn các URL khác nhau cùng trỏ tới một trang web. Ta có thể hạn chế điều này bằng cách hạn chế số lượng các trang web mà crawler truy cập tới từ một domain xác định. Đoạn mã lệnh liên quan tới frontier có thể đảm bảo rằng mọi chuỗi k URL liên tiếp (thường k=100), được lấy ra bởi crawler, chỉ chứa duy nhất một địa chỉ URL chuẩn hóa. Điều này sẽ tránh được việc crawler phải truy cập cùng một web site quá nhiều lần, và nội dung các trang web tải được sẽ có xu hướng khác biệt nhiều hơn.

2.2.2 History và kho chứa trang web

Phần history của crawler là một danh sách động các URL đã được nạp bởi crawler. Nó chứa các đường dẫn mà crawler đã đi qua bắt đầu từ trang hạt nhân. Một URL đầu vào chỉ được tạo trong phần history sau khi trang web tương ứng đã được nạp. Phần này được sử dụng cho việc phân tích và đánh giá các trang web sau này. Ví dụ, chúng ta có thể gắn cho mỗi trang web một giá trị trên đường dẫn và xác định các sự kiện có ý nghĩa (ví dụ như việc khám phá ra một nguồn lực quan trọng). Trong một số trường hợp phần history được lưu trữ ở bộ nhớ ngoài, nhưng nó cũng có thể được duy trì như một cấu trúc dữ liệu trong bộ nhớ trong. Điều này cho phép tìm kiếm nhanh chóng để kiểm tra xem liệu một trang web đã được duyệt hay chưa. Việc kiểm tra này là rất quan trọng để tránh đi thăm lại các trang web, và do đó tránh việc thêm các URL đã được duyệt vào trong frontier có kích thước giới hạn. Cũng với lý do

tương tự, việc chuẩn hóa các URL (2.2.4.1) trước khi thêm chúng vào history cũng rất quan trọng.

Khi trang web đã được tải, nó có thể được lưu trữ, đánh chỉ số để phục vụ cho ứng dụng chính (ví dụ một máy tìm kiếm). Ở dạng đơn giản nhất, một kho chứa các trang web có thể có thể lưu các trang web đã được crawl như các file riêng biệt. Trong trường hợp đó, mỗi trang phải được ánh xạ tới một tên file duy nhất. Một cách để thực hiện điều này là ánh xạ URL của mỗi trang tới một chuỗi nén bằng cách sử dụng một dạng hàm băm với xác xuất xung đột thấp (để đảm bảo tính duy nhất của tên file). Các giá trị băm được sử dụng làm các tên file. Chúng tôi sử dụng hàm băm một chiều MD5 để cung cấp mã băm 128 bit cho mỗi URL. Giá trị băm 128 bit sau đó được chuyển thành 32 ký tự ở dạng cơ số 16 tương ứng. Theo cách này ta sẽ có các tên file có chiều dài cố định cho các URL có độ dài bất kỳ. Các kho chứa nội dung trang web có thể được sử dụng để kiểm tra liệu một URL đã được crawl trước đó hay chưa bằng cách chuyển URL đó sang 32 ký tự thập lục phân và kiểm tra sự tồn tại của nó trong kho chứa. Trong một số trường hợp, điều này có thể dẫn tới sự không cần thiết của cấu trúc dữ liệu history trong bộ nhớ trong.

2.2.3 Tải các trang web (fetching)

Để nạp một trang web, ta cần một HTTP client để gửi một yêu cầu HTTP tới một trang web và đọc các đáp ứng. Phía client cần có một thời gian timeout để đảm bảo rằng nó không lãng phí quá nhiều thời gian để giao tiếp với một server quá chậm hoặc đọc một trang web quá lớn. Trên thực tế, chúng tôi thường giới hạn client chỉ download các trang web có kích thước nhỏ hơn 10-20KB. Phía client cần duyệt các đáp ứng header để lấy các mã trạng thái và các sự định hướng lại (redirection). Chúng cũng duyệt header và lưu thời gian sửa đổi (last-modify) để xác định độ cập nhật của trang web. Việc kiểm tra các lỗi và ngoại lệ là rất quan trọng trong quá trình tải trang web do chúng ta sẽ phải liên hệ tới hàng triệu server ở xa bằng cùng một đoạn mã lệnh. Thêm vào đó, việc thu thập các thống kê về thời gian timeout và các mã trạng thái cũng rất hữu ích trong việc xác định các vấn đề nảy sinh hoặc để thay đổi tự động giá trị timeout. Các ngôn ngữ lập trình hiện đại như Java hoặc Perl cung cấp các cơ chế đơn giản cùng nhiều giao diện lập trình để tải các trang web. Tuy nhiên, ta cũng cần phải cẩn thận trong việc sử dụng các giao diện bậc cao do có thể sẽ khó tìm ra các lỗi ở bậc thấp.

Chúng ta không thể kết thúc việc nói về quá trình crawling mà không đề cập tới giao thức loại trừ robot Robot Exclusion Protocol. Giao thức này cung cấp một cơ chế cho các nhà quản trị Web server để thông báo về các quyền truy nhập file trên server, đặc biệt là để chỉ định các file không được truy cập bởi một crawler. Điều này được thực hiện bằng cách lưu một file có tên robots.txt dưới thư mục chủ của Web server (chẳng hạn http://www.biz.uiowa.edu/robots.txt). File này cung cấp các chính sách truy cập các cho User-agents khác nhau (robots hoặc crawler). Một giá trị User-agent '*' biểu diễn một chính sách mặc định cho bất kỳ crawler nào không khớp (match) các giá trị User-agent khác trong file. Một số các đầu vào bị cấm Disallow được đề ra cho một User-agent. Bất kỳ URL nào bắt đầu với giá trị trường disallow được đặt sẽ không được truy xuất bởi các crawler ứng với các giá trị User-agent đã chỉ định. Khi một crawler muốn lấy ra một trang web từ web server, đầu tiên nó phải nạp file robots.txt và đảm bảo rằng URL cần nạp là không bị cấm. Các thông tin chi tiết về giao thức loại trừ này có thể tìm thấy ở http://www.robotstxt.org/wc/norobots.html. Việc lưu cache các chính sách truy nhập của một số server được truy nhập gần đây là khá hiệu quả. Điều này cho phép tránh phải truy nhập một file robots.txt mỗi khi cần nạp một URL. Tuy nhiên, ta cần phải đảm bảo rằng các thông tin trong cache là đủ cập nhật.

Ví dụ	**Ý nghĩa**
User-agent: * Disallow: /cgi-bin/ Disallow: /tmp/	Tất cả các máy tìm kiếm có thể thăm tất cả các thư mục ngoại trừ hai thư mục đề cập ở đây
User-agent: BadBot Disallow: /	Máy tìm kiếm BadBot không được phép thăm bất cứ thư mục nào.
User-agent: BadBot Disallow: / User-agent:* Disallow : /private/	Riêng máy tìm kiếm BadBot không được phép thăm bất cứ thư mục nào còn tất cả các máy tìm kiếm còn lại đều có quyền thăm tất cả các thư mục ngoại trừ thư mục "private"

2.2.4 Duyệt và phân tích nội dung (parsing)

Sau khi trang web đã được tải về, chúng ta cần duyệt nội dung của nó để lấy ra các thông tin sẽ được nạp trở lại và giúp định hướng việc đi theo các đường dẫn tiếp theo của crawler. Việc duyệt nội dung có thể đơn giản chỉ bao hàm việc trích ra các

URL/liên kết mà trang web link tới hay nó có thể bao hàm các xử lý phức tạp như làm sạch các nội dung HTML để phân tích cấu trúc cây của các thẻ. Việc duyệt có thể bao gồm các bước để chuẩn hóa các URL được lấy ra, loại bỏ các từ dừng khỏi nội dung trang web ... Các thành phần của bộ duyệt được mô tả ở phần sau.

2.2.4.1. Quá trình lấy ra và chuẩn hóa các URL.

Bộ duyệt HTML đã được xây dựng sẵn trong rất nhiều ngôn ngữ. Chúng cung cấp các tính năng để dễ dàng xác định các thẻ HTML và liên kết giá trị các cặp thuộc tính trong một văn bản HTML cho trước. Để lấy ra được các URL hyperlink từ một trang web, ta có thể sử dụng các bộ duyệt ở trên để tìm các thẻ anchor và lấy ra các giá trị của thuộc tính href tương ứng. Tuy nhiên, chúng ta cần chuyển các URL tương đối sang các địa chỉ URL tuyệt đối sử dụng URL cơ sở của trang web nơi chúng được trích ra.

Các URL khác nhau tương ứng với cùng một trang web có thể được ánh xạ vào một dạng chuẩn đơn nhất. Điều này rất quan trọng nhằm tránh được việc nạp cùng một trang web nhiều lần. Sau đây là các bước được sử dụng trong các hàm chuẩn hóa thông dụng:

- Chuyển giao thức và tên máy chủ sang dạng chữ thường.

Ví dụ: HTTP://www.UIOWA.edu được chuyển thành http://www.uiowa.edu.

- Loại bỏ phần anchor hoặc reference của URL. Do đó: http://spiders.uiowa.edu/faq.htm# được thu gọn thành http://spiders.uiowa.edu/faq.htm

- Thực hiện việc mã hóa URL bằng các ký tự thông dụng như ‘~’. Điều này sẽ ngăn chặn các crawler khỏi bị đánh lừa http://dollar.biz.uiowa.edu/~pant/ là một URL khác với http://dollar.biz.uiowa.edu/%7Epant/.

- Đối với một số URL, thêm vào dấu ‘/’. http://dollar.biz.uiowa.edu và http://dollar.biz.uiowa.edu/ phải cùng ánh xạ vào cùng một dạng chuẩn. Việc thêm vào dấu ‘/’ hay không trong nhiều trường hợp đòi hỏi kinh nghiệm.

- Sử dụng các kinh nghiệm để nhận ra các trang web mặc định. Các tên file như index.html hay index.htm có thể bị loại khỏi URL bởi chúng vẫn được coi là các file mặc định. Nếu điều này là đúng, chúng có thể được lấy ra bằng cách chỉ sử dụng URL cơ sở.

- Loại bỏ ký tự '..' và thư mục cha khỏi đường dẫn URL. Do đó, địa chỉ URL /%7epant/Bizlntel/Seeds/../ODPSeeds.dat được đơn giản hóa thành /%7Epant/Bizlnel/ODPSeeds.dat.

- Để lại các số hiệu cổng trong các URL ngoại trừ đó là cổng 80. Một cách khác là để lại các số hiệu cổng trong URL và thêm cổng 80 nếu số hiệu cổng không được chỉ định.

2.2.4.2 Loại bỏ các từ dừng và chuyển các dạng thức của từ sang dạng gốc.

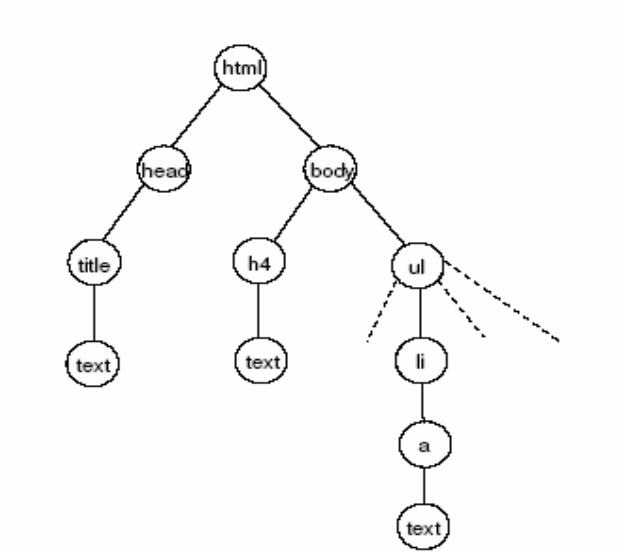
Khi duyệt một trang web để trích ra các thông tin nội dung hoặc để tính điểm các URL mà trang đó trỏ tới, thông thường ta nên loại bỏ các từ được dùng thường xuyên hay từ dừng (stopwords) như 'it' hay 'can' trong Tiếng Anh... Tiến trình xử lý việc loại bỏ các từ dừng khỏi văn bản được gọi là stoplisting. Ngoài việc xử lý các từ dừng, ta cũng cần lấy ra từ gốc của các từ có trong văn bản. Quá trình stemming chuẩn hóa các từ bằng cách đúc kết các hình thái của các từ thành một từ ở dạng gốc hay stem. Ví dụ: từ connect, connected hay connection đều được đưa về dạng connect.

2.2.4.3 Xây dựng cây các thẻ HTML

Các chương trình crawler có thể đánh giá giá trị của một URL hoặc một từ trong nội dung trang web bằng cách xem xét ngữ cảnh của các thẻ HTML mà nó thuộc vào. Để làm được điều này, một crawler cần sử dụng cây các thẻ hoặc cấu trúc DOM của trang HTML [8,23, 25]. Hình 2 chỉ ra cấu trúc cây của các thẻ tương ứng với văn bản HTML nguồn. Thẻ <html> được lấy làm gốc của cây các thẻ khác và text tạo thành các nút của cây. Đáng tiếc là, rất nhiều trang web có cấu trúc HTML không chuẩn. Ví dụ, một thẻ bắt đầu có thể không có thẻ đóng, hoặc các thẻ không được lồng nhau một cách hợp lý. Trong nhiều trường hợp, thẻ <html> hoặc <body> đều bị thiếu trong trang HTML. Do đó các tiêu chuẩn dựa trên cấu trúc (structure-based criteria) thường cần có một bước tiền xử lý để chuẩn hóa một văn bản HTML có cấu trúc không chuẩn, quá trình xử lý này gọi là làm sạch (tidying) các trang HTML. Nó bao gồm cả việc chèn thêm các thẻ bị thiếu và sắp xếp lại thứ tự các thẻ trong trang. Việc làm sạch một trang HTML là cần thiết để ánh xạ nội dung của trang vào trong một cấu trúc cây để đảm bảo tính toàn vẹn, mỗi nút có một cha duy nhất, từ đó phân tích nên cấu trúc cây của các thẻ. Chú ý rằng việc phân tích cấu trúc DOM chỉ cần thiết nếu crawler theo chủ điểm có ý định sử dụng cấu trúc của trang HTML cho những phân tích phức tạp. Còn nếu crawler chỉ cần các liên kết trong một trang, các từ khóa và vị

trí xuất hiện của chúng trong trang web thì chỉ cần sử dụng các bộ duyệt HTML thông thường. Các bộ duyệt này rất sẵn trong nhiều ngôn ngữ lập trình.

```
<html>
<head>
<title>Projects</title>
</head>
<body>
<h4>Projects</h4>
<ul>
<li> <a href="blink.html">LAMP</a> Linkage analysis with multiple processors.</li>
<li> <a href="nice.html">NICE</a> The network infrastructure for combinatorial   exploration.</li>
<li> <a href="amass.html">AMASS</a> A DNA sequence assembly algorithm.</li>
<li> <a href="dali.html">DALI</a> A distributed, adaptive, first-order logic theorem prover.</li>
</ul>
</body>
</html>
```

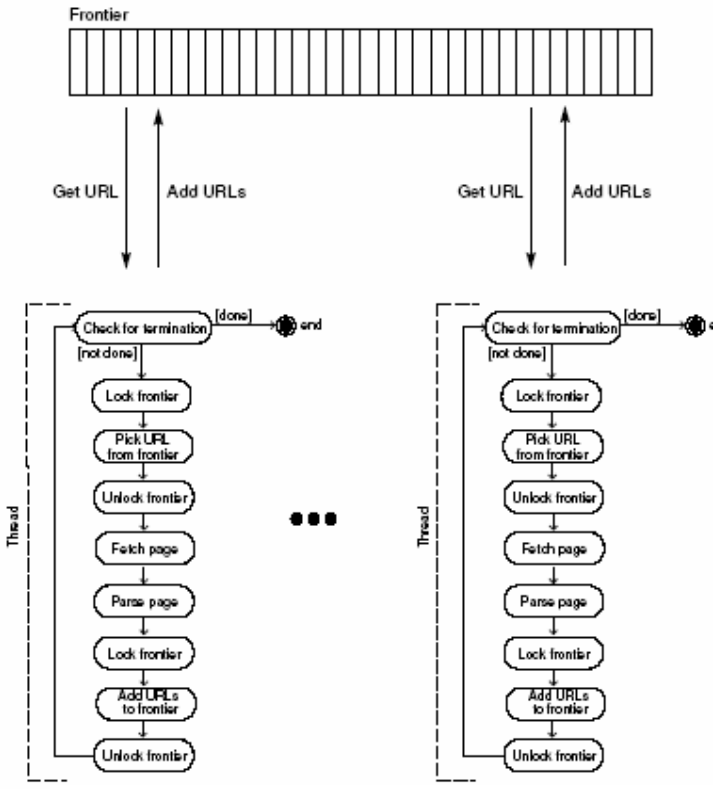


Hình 2.2: *Một trang HTML và cấu trúc cây của thẻ tương ứng.*

2.3 Crawler đa luồng (Multi-threaded crawler)

Trong một vòng lặp crawling tuần tự có một lượng lớn thời gian trong đó hoặc CPU là rỗi (quá trình truy cập mạng và đĩa) hoặc giao thức mạng là rỗi (trong khi CPU đang xử lý). Việc sử dụng đa luồng trong đó mỗi luồng xử lý một vòng lặp crawling có thể cải thiện được đáng kể tốc độ và hiệu quả sử dụng băng thông mạng. Hình 2.3 thể hiện một phiên bản đa luồng của một crawler cơ bản trong hình 2.1. Chú ý rằng mỗi luồng bắt đầu bằng việc khóa frontier để lấy ra URL tiếp theo để crawl. Sau khi lấy được URL nó mở khóa frontier để cho phép các luồng khác truy cập vào. Khi các URL được thêm vào, frontier lại bị khóa một lần nữa. Các bước khóa này là cần thiết để

đồng bộ hóa việc sử dụng frontier giờ đây được chia sẻ giữa rất nhiều vòng lặp crawling (các luồng). Mô hình crawler đa luồng trong hình 2.3 tuân theo tiêu chuẩn của mô hình máy tính song song. Cần chú ý rằng một crawler điển hình cũng cần duy trì một cấu trúc history chung để tìm kiếm nhanh các URL đã được crawl. Do đó, ngoài frontier, các crawler cũng cần đồng bộ các truy cập vào history.



Hình 2.3: *Một mô hình crawler đa luồng.*

Mô hình crawler đa luồng cũng cần phải giải quyết trường hợp frontier bị rỗng giống như mô hình crawler tuần tự. Tuy nhiên, vấn đề ở đây không đơn giản như vậy. Nếu một luồng phát hiện ra frontier là rỗng, nó không tự cho rằng toàn bộ crawler đã đạt tới trạng thái dead-end. Hoàn toàn có khả năng rằng các luồng khác đang nạp các trang và có thể thêm các URL mới ngay sau đó. Một cách để giải quyết tình huống này là gửi cho một luồng một trạng thái nghỉ (sleep state) khi nó gặp một frontier rỗng. Khi luồng thức dậy, nó cố gắng lấy các URL một lần nữa. Một bộ phận quản lý toàn cục sẽ đếm số lần các luồng phải nghỉ trong thời gian gần đây. Chỉ khi tất cả các luồng đều đã ở trạng thái nghỉ thì quá trình crawler mới dừng lại.

Phần này đã mô tả các bộ phận tổng quát cấu thành một crawler. Có những crawler sẽ hỗ trợ thuật toán mở rộng theo chiều rộng đơn giản, cũng có những thuật

toán crawler liên quan tới cơ chế lựa chọn các URL rất phức tạp. Các vấn đề như kích thước của frontier, chiến lược duyệt trang, crawler history và kho chứa được xác định như là các thông số quan trọng và hấp dẫn trong khái niệm về crawler.

2.4. Các thuật toán crawling

Chúng ta cùng xem xét một số thuật toán crawling điển hình. Rất nhiều thuật toán trong số đó là các biến thể của mô hình tốt nhất đầu tiên (best-first scheme). Sự khác biệt giữa chúng chính là các kinh nghiệm mà chúng sử dụng để tính điểm cho các URL chưa được thăm, một số thuật toán còn có thể chỉnh lại giá trị các tham số trước hoặc trong quá trình crawl.

2.4.1 Thuật toán Naïve tốt nhất đầu tiên

Các crawler áp dụng thuật toán này biểu diễn một trang web đã được tải như là một vector các từ được đánh trọng số dựa trên số lần xuất hiện của từ đó. Chương trình crawler sau đó tính toán mức độ tương tự của trang web với câu truy vấn hoặc câu mô tả của người dùng, và tính điểm các URL chưa được viếng thăm trong trang web đó bằng giá trị tương tự đó. Sau đó các URL được thêm vào frontier được quản lý dưới dạng một hàng đợi ưu tiên dựa trên các điểm được tính đó. Trong các vòng lặp tiếp theo, mỗi luồng của crawler lấy ra URL tốt nhất trong frontier để crawl, và trả về các URL chưa được thăm mới, và chúng lại tiếp tục được thêm vào hàng đợi ưu tiên sau khi đã được tính điểm dựa trên mức độ tương tự của trang web cha. Mức độ tương tự giữa trang p và câu truy vấn q được tính bởi.

$$sim\,(q, p) = \frac{v_q . v_p}{\| v_q \| . \| v_p \|} \qquad (1)$$

Trong đó *vq* và *vp* tương ứng là các vector biểu diễn số lần xuất hiện hay mức độ thường xuyên (term frequency) *TF* của các từ trong câu truy vấn và trong trang web. *vq.vp* là tích vô hướng của hai vector và ||*v*|| là chuẩn Euclidean của vector *v*. Các biểu diễn vector phức tạp hơn của trang web như mô hình TF-IDF thường được sử dụng trong lĩnh vực information retrieve, là rất khó áp dụng trong các crawler do nó đòi hỏi hiểu biết trước về việc phân phối các từ khóa (term) trong toàn trang web được tải về. Trong việc thực thi đa luồng, các crawler thường hoạt động như các crawler N tốt nhất đầu tiên (best-N-first) trong đó N là một hàm số của số lượng các luồng chạy đồng thời. Do đó N tốt nhất đầu tiên là phiên bản tổng quát hóa của crawler tốt nhất

đầu tiên, nó lấy ra N URL tốt nhất để duyệt trong một thời điểm. Theo nghiên cứu trong [] crawler N tốt nhất đầu tiên (với N=256) là một giải pháp rất tốt, thể hiện được ưu điểm vượt trội hơn hẳn trong việc lấy ra được các trang web hợp lệ. Chú ý rằng crawler tốt nhất đầu tiên giữ cho kích thước của frontier trong giới hạn trên bằng cách chỉ giữ lại các URL tốt nhất dựa vào các điểm mà chúng được gán.

2.4.2 Thuật toán SharkSearch

SharkSearch [15] là một phiên bản của FishSearch [12] với một số cải tiến. Nó sử dụng một độ đo mức độ tương tự giống như trong crawler Naïve tốt nhất đầu tiên để đánh giá tính hợp lệ của URL chưa được viếng thăm. Tuy nhiên, SharkSearch có một cơ chế tính điểm tinh tế hơn cho các liên kết trong frontier. Các từ thể hiện liên kết (anchor text), các từ xung quanh liên kết hoặc ngữ cảnh của liên kết, và điểm được thừa kế từ URL cha (ancestor) đều ảnh hưởng tới việc tính điểm của liên kết. Các URL phía trước của một URL là các trang web mà xuất hiện trong đường dẫn crawl để tới URL đó. SharkSearch giống như phiên bản xuất phát FishSearch, lưu giữ một giới hạn độ sâu. Nghĩa là, nếu chương trình crawler tìm thấy các trang web không quan trọng trên đường dẫn khi crawl, nó sẽ dừng việc duyệt đi xa hơn trên đường dẫn đó. Để có thể theo dõi tất cả các thông tin, mỗi Url trong frontier được liên kết với một độ sâu và một điểm số. Giới hạn độ sâu (d) được cung cấp bởi người dùng trong khi điểm số của một URL chưa được viếng thăm được tính bởi công thức:

$$score(url) = \gamma.inherited(url) + (1-\gamma).neighborhood(url) \quad (2)$$

Trong đó $\gamma < 1$ là một tham số, điểm số lân cận neighborhood score biểu thị các dấu hiệu ngữ cảnh tìm thấy trong trang web chứa liên kết tới URL đó, và điểm số được thừa kế inherited score nhận được từ điểm số của các URL cha của URL hiện tại. Một cách chính xác hơn, inherited score được tính bởi:

$$inherited(url) = \begin{cases} \delta.sim(q,p) & sim(q,p) > 0 \\ \delta.inherited(p) & otherwise \end{cases} \quad (3)$$

Trong đó: $\delta < 1$ là một tham số khác, q là câu truy vấn, và p là trang web mà từ đó URL được trích ra.

Một neighborhood score sử dụng các từ biểu diễn liên kết (anchor text) và các từ lân cận của liên kết nhằm cải tiến tổng điểm của một URL bằng cách chú ý đến sự khác nhau giữa các liên kết được tìm thấy trong cùng một trang. Để phục vụ mục đích này, các crawler áp dụng SharkSearch gán một điểm số liên kết anchor score và điểm

số ngữ cảnh context score cho mỗi URL. Trong khi anchor score chỉ đơn giản là mức độ tương tự giữa các từ khóa dùng để biểu diễn liên kết tới URL với câu truy vấn q. ví dụ sim(q, anchor_text). Thì context score mở rộng ngữ cảnh của liên kết tới cả các từ khóa gần đó. Kết quả là một ngữ cảnh mở rộng, aug_context, được sử dụng để tính giá trị của context score như sau:

$$context(url) = \begin{cases} 1 & anchor(url) > 0 \\ sim(q, aug\_context) & otherwise \end{cases} \quad (4)$$

Cuối cùng chúng ta nhận được neighborhood score từ anchor score và context score bằng cách:

$$neighborhood(url) = \beta.anchor(url) + (1-\beta).contex(url) \quad (5)$$

Trong đó $\beta<1$ là một tham số khác. Cần chú ý rằng để thực thi được thuật toán SharkSearch ta cần phải đặt trước 4 tham số khác nhau: d, γ, δ và β. Một số giá trị tham số được đề xuất tại [15].

2.4.3 Crawler hướng tâm (focused crawler)

Charkrabarti [9,6] đã phát triển một focused crawler dựa trên một bộ phân lớp siêu liên kết. Ý tưởng cơ bản của crawler này là phân lớp các trang web được tải về vào các lớp theo cấu trúc các lớp chủ đề có sẵn. Để bắt đầu, bộ crawler cần có một cấu trúc cây các chủ đề để phân lớp như trong Yahoo hoặc ODP (Open Directory Project). Thêm vào đó, người dùng cần cung cấp các URL mẫu để phân lớp. Các URL mẫu được phân loại một cách tự động vào các lớp khác nhau trong cây chủ đề. Thông qua một quá trình tương tác, người dùng có thể sửa chữa việc phân lớp tự động đó, thêm các chủ đề mới và đánh dấu một số lớp/chủ đề là tốt (dựa theo mức độ quan tâm của người dùng). Bộ crawler sử dụng các URL mẫu để xây dựng một bộ phân lớp Bayesian để tìm ra xác xuất *(Pr(c|p))* mà một trang web đã được duyệt p thuộc vào lớp c trong cây chủ đề. Chú ý rằng theo định nghĩa *Pr(r|p)=1* với *r* là lớp gốc của cây chủ đề. Một độ đo tính hợp lệ được gán cho mỗi trang web được tải về theo công thức:

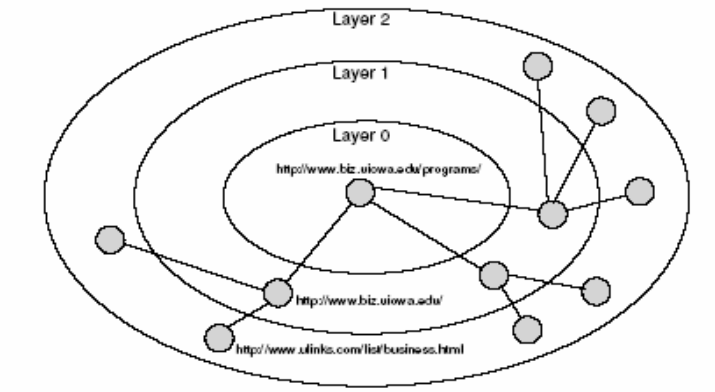
$$R(p) = \sum_{c \in good} \Pr(c \mid p) \quad (6)$$

Nếu một crawler tuân theo mô hình trọng tâm mềm "soft" focused, nó sử dụng độ đo tính hợp lệ của trang web được tải để tính điểm cho các URL chưa được viếng thăm trích ra từ trang đó. Các URL đã được tính điểm sau đó sẽ được thêm vào frontier. Tiếp theo, chương trình crawler đó sẽ lấy ra các URL tốt nhất tiếp theo theo

cách tương tự như trong thuật toán Naïve tốt nhất đầu tiên. Còn trong mô hình trọng tâm "cứng" "hard" focused, đối với mỗi trang web đã được tải p, đầu tiên bộ phân lớp sẽ tìm các nút lá c* trong cấu trúc lớp các chủ đề có xác xuất trang p thuộc vào lớp đó là lớn nhất. Nếu bất kỳ chủ đề cha nào của c* được người dùng đánh dấu là tốt, thì các URL được liên kết tới trong trang p sẽ được trích ra và thêm vào frontier.

2.3.4 Các crawler tập trung theo ngữ cảnh (context focused crawler)

Các context focused crawler [13] sử dụng một bộ phân lớp Bayesian để hướng dẫn quá trình crawl. Tuy nhiên, không giống các focus crawler phía trên, các bộ phân lớp này được huấn luyện để đánh giá khoảng cách liên kết giữa một trang được tải và một trang web hợp lệ. Chúng ta có thể nhận thức được giá trị của bộ đánh giá này từ chính kinh nghiệm duyệt web của mình. Nếu chúng ta đang tìm kiếm các trang về "phân tích số học" đầu tiên ta có thể tới các trang chủ về toán học hoặc khoa học máy tính và sau đó chuyển tới các phân trang nhỏ hơn mà có thể dẫn ta tới các trang hợp lệ. Một web site chuyên về toán học thường sẽ không có cụm từ "phân tích số học" ở trong trang chủ của nó. Một crawler sử dụng thuật toán naïve tốt nhất đầu tiên có thể sẽ gán cho các trang đó một độ ưu tiên thấp và có thể sẽ chẳng bao giờ thăm chúng. Tuy nhiên, nếu crawler có thể đánh giá rằng được khoảng cách giữa một trang hợp lệ về chủ đề "phân tích số học" với trang đang được duyệt, ta sẽ có một cách thức để cấp cho trang chủ của khoa toán độ ưu tiên cao hơn trang chủ của một trường luật.



Hình 3.1 *Một sơ đồ ngữ cảnh*.

Các contex focused crawler được huấn luyện bằng cách sử dụng các sơ đồ ngữ cảnh contex graph L tầng tương ứng với mỗi trang web hạt nhân. Các trang web hạt nhân tạo thành tầng thứ 0 của đồ thị. Các trang web chứa các liên kết tới trang hạt nhân (in-link) tạo thành tầng 1. Các trang chứa liên kết tới các trang thuộc tầng 1 tạo

thành tầng thứ 2 và cứ như vậy. Chúng ta có thể đi theo các liên kết vào in-link để tới các trang thuộc tầng bất kỳ bằng cách sử dụng một bộ tìm kiếm. Hình 4 mô tả một sơ đồ ngữ cảnh với trang http://www.biz.uiowa.edu/programs làm hạt nhân. Khi có được sơ đồ ngữ cảnh của tất cả các hạt nhân, các trang web ở cùng một tầng từ mỗi đồ thị được kết hợp vào một tầng đơn. Như vậy ta tạo được tập các tầng mới gọi là sơ đồ ngữ cảnh tổng hợp merged contex graph. Sơ đồ này được đi theo bởi một bộ lựa chọn đặc trưng trong đó các trang hạt nhân (hoặc có thể cả các trang ở tầng thứ nhất) được nối lại tạo thành một văn bản lớn. Sử dụng cách thức tính điểm TF-IDF [31], một số từ có điểm cao nhất trong văn bản này sẽ được sử dụng để xây dựng nên bộ từ điển (không gian các đặc trưng) được dùng để phân lớp.

Một tập các bộ phân lớp naïve Bayes được xây dựng, mỗi tầng trong sơ đồ ngữ cảnh tổng hợp có một bộ phân lớp riêng. Tất cả các trang trong một tầng được sử dụng để tính giá trị Pr(t|cl), là xác xuất xuất hiện từ t trong lớp cl tương ứng với tầng thứ l. Một xác xuất ưu tiên, Pr(cl)=1/L được gán cho mỗi lớp, trong đó L là số lượng các tầng. Xác xuất của một trang web cần xét p thuộc vào một lớp cl được tính bởi Pr(cl|p). Các xác xuất này được tính cho tất cả các lớp. Lớp mà có xác xuất lớn nhất được coi là lớp (tầng) thắng cuộc. Tuy nhiên, nếu xác xuất của lớp thắng cuộc vẫn nhỏ hơn một giá trị ngưỡng, thì trang web đang xét được phân vào lớp “other”. Lớp “other” này chứa các trang web mà không phù hợp với bất kỳ lớp nào trong sơ đồ ngữ cảnh. Nếu xác xuất của lớp thắng cuộc lớn hơn giá trị ngưỡng, trang web đó sẽ được phân lớp vào lớp thắng cuộc.

Tập các bộ phân lớp tương ứng với sơ đồ ngữ cảnh cung cấp cho chúng ta một cơ chế để đánh giá khoảng cách liên kết giữa một trang web đang được duyệt và một trang web hợp lệ. Nếu sử dụng cơ chế này, trang chủ của một khoa Toán có thể sẽ được phân lớp vào tầng thứ 2 trong khi trang chủ của một trường Luật sẽ được phân lớp vào lớp “other”. Chương trình crawler cần lưu một hàng đợi cho mỗi lớp, hàng đợi này sẽ chứa các trang web đã được duyệt và phân vào trong lớp đó. Mỗi hàng đợi được sắp xếp bởi một điểm xác xuất (Pr(cl|p)). Khi chương trình crawler cần một URL để tải, nó sẽ lấy ra trang web ở đỉnh của một hàng đợi không rỗng có giá trị l là nhỏ nhất. Do đó nó sẽ khuynh hướng lấy ra được các trang có khoảng cách gần với các trang hợp lệ nhất trước hết. Các liên kết ra khỏi các trang này sẽ được duyệt trước các liên kết ra từ các trang được đánh giá là có khoảng cách xa so với các trang hợp lệ.

2.4. Các tiêu chuẩn đánh giá các crawler

Theo nhận định thông thường, một crawler (đặc biệt là một topic crawler) có thể được đánh giá dựa trên khả năng lấy được các trang web "tốt". Tuy nhiên, vấn đề mấu chốt ở đây chính là làm thế nào để nhận ra một trang web "tốt". Trong môi trường tương tác, một người dùng thực có thể xác định được tính hợp lệ của các trang được tải về và cho phép chúng ta xác định liệu quá trình crawl có thành công hay không. Nhưng không may là việc thực hiện các thí nghiệm hiệu quả có sự tham gia của những người dùng thực để đánh giá chất lượng của một crawler là cực kỳ khó khăn. Do kích thước khổng lồ của Web cho thấy, để có thể đạt được những đánh giá hợp lý về mức độ hiệu quả của quá trình crawl chúng ta cần phải xem xét một số lượng lớn các cách thức crawl, do đó cần liên quan tới một số lượng lớn người dùng.

Thứ hai, quá trình crawl phải thỏa mãn các ràng buộc nghiêm ngặt về thời gian. Do đó quá trình crawl, nếu không được thực hiện trong một thời gian ngắn sẽ trở nên rất phiền toái cho người dùng. Nếu chúng ta có giảm thời gian tải thì lại sẽ hạn chế qui mô của quá trình crawl, và việc đánh giá lại không chuẩn xác.

Trong một tương lai không xa, những người thu thập thông tin trực tiếp sẽ là các Web agent đại diện cho người dùng hoặc các Web agent khác hơn là bản thân người dùng. Do đó, việc khảo sát các crawler là khá hợp lý trong một ngữ cảnh khi mà các tham số về thời gian crawl và khoảng cách crawl có thể vượt xa khỏi các hạn chế bị áp đặt trong các thử nghiệm với người dùng thực.

Thông thường, việc so sánh các topic crawler theo một lượng lớn các chủ đề và nhiệm vụ là rất quan trọng. Điều này cho phép chúng ta biết được một cách chắc chắn ý nghĩa thống kê của mỗi một cải tiến được đề xuất cho crawler. Các nghiên cứu về đánh giá các crawler đòi hỏi một tập các độ đo thích hợp. Đầu tiên chúng ta sẽ xem xét hai khía cạnh cơ bản trong quá trình đánh giá. Chúng ta cần một *độ đo về độ quan trọng* của các trang web và sau đó là một *phương pháp để tổng hợp các hiệu năng* thông qua một tập các trang web được crawl.

2.4.1 Độ quan trọng của trang web

Chúng ta cùng liệt kê một số phương pháp đã được sử dụng để đo độ quan trọng của các trang web.

Các từ khóa trong văn bản: Một trang web được coi là hợp lệ nếu nó có chứa một số hoặc tất cả các từ khóa trong câu truy vấn. Cũng như vậy, tần số xuất hiện của từ khóa trong trang cũng được xem xét.

Mức độ tương tự với câu truy vấn: Thông thường một người dùng chỉ định một thông tin cần tìm bởi một câu truy vấn ngắn. Trong một số trường hợp người dùng có thể có một mô tả về điều cần biết bằng các cụm từ dài hơn. Mức độ tương tự giữa các mô tả ngắn hay dài của người dùng với mỗi trang web được tải về có thể sử dụng để xác định tính hợp lệ của trang.

Mức độ tương tự với trang hạt nhân: Các trang tương ứng với các URL hạt nhân được sử dụng để đo mức độ hợp lệ của mỗi trang được tải. Trang web hạt nhân được kết hợp với nhau thành một văn bản lớn duy nhất và mức độ gần nhau của trang văn bản này với các trang web đang được duyệt được sử dụng làm điểm số của trang web đó.

Điểm số phân lớp: một bộ phân lớp có thể được huấn luyện để xác đinh các trang phù hợp với thông tin hoặc nhiệm vụ cần làm. Việc huấn luyện được tiến hành sử dụng các trang hạt nhân (hoặc các trang web hợp lệ được chỉ định trước) như là các ví dụ dương. Các bộ phân lớp được huấn luyện sau đó sẽ gán các điểm số nhị phân (0,1) hoặc liên tiếp cho các trang web được duyệt [9].

Tính hạng cho hệ thống các trang lấy được: N crawler khác nhau cùng bắt đầu bởi cùng một tập các trang hạt nhân và được chạy cho tới khi mỗi crawler lấy được P trang web. Tất cả N.P trang tập hợp được từ các crawler được tính hạng dựa trên câu truy vấn ban đầu hoặc mô tả bằng cách sử dụng một hệ thống phục hồi (truy xuất thông tin) retrieval system chẳng hạn SMART. Các thứ hạng được cung cấp bởi hệ thống này được sử dụng như là mức độ hợp lệ của trang web [21].

Tính phổ biến dựa trên liên kết (link-based popularity): Một crawler có thể sử dụng các thuật toán như PageRank hoặc HITS [16], để cung cấp một sự đánh giá tính phổ cập của mỗi trang web được duyệt. Một phương pháp đơn giản hơn là chỉ sử dụng số lượng các liên kết tới trang web đó để xác định thông tin đó. Rất nhiều biến thể của các phương pháp dựa trên các liên kết sử dụng các trọng số của chủ đề được sử dụng để đo tính phổ biến về chủ đề đó của trang web [4, 7].

2.4.2 Các phân tích tổng hợp

Sau khi được cung cấp một cách thức xác định để đo độ quan trọng của trang web, ta có thể tổng kết hiệu năng của các crawler với các đơn vị đo tương tự như độ đo độ chính xác precision và độ hồi tưởng recall trong information retrieval (IR). Độ chính xác precision là tỉ lệ các trang web hợp lệ trên tổng số các trang web được duyệt, còn độ hồi tưởng recall là tỉ lệ các trang web hợp lệ đã được lấy về trên tổng số tất cả các trang hợp lệ. Trong một nhiệm vụ IR thông thường khái niệm một tập hợp lệ để tính recall thường chỉ hạn chế trong một tập hợp dữ liệu xác định hoặc cơ sở dữ liệu. Nếu ta xem toàn bộ Web là một tập hợp lớn, thì tập các dữ liệu hợp lệ thường là không được biết trước cho phần lớn các nhiệm vụ IR trên Web. Do đó, giá trị recall chính xác rất khó xác định. Rất nhiều học giả đã đề xuất độ đo giống precision (precision-like) dễ tính toán hơn để từ đó đánh giá hiệu năng của các crawler. Chúng ta cùng xem xét một số độ đo precision-like này.

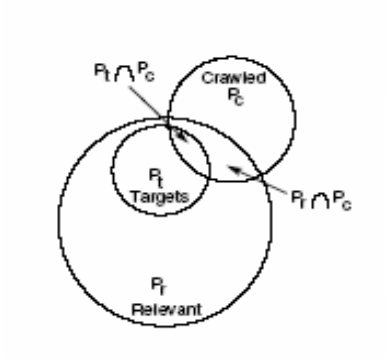
Tỉ lệ thu được (acquisition rate): Trong trường hợp ta gán cho mỗi trang một trọng số ở dạng nhị phân (0,1) ta có thể dễ dàng tính được tỉ lệ các trang web "tốt" được tìm thấy. Chẳng hạn, nếu ta tìm được 50 trang hợp lệ (có trọng số là 1) trong số 500 trang web được tải về thì ta sẽ có acquisition rate là 10% của 500 trang.

Tỉ lệ hợp lệ trung bình: nếu điểm số hợp lệ là giá trị liên tục, ta có thể được tính giá trị trung bình trên tổng số các trang được crawl. Đây là trường hợp tổng quát hơn của tỉ lệ thu hoạch ở trên. Điểm số của chúng có thể được cung cấp thông qua một quá trình đo mức độ tương tự hoặc bằng một bộ phân lớp đã được huấn luyện. Tỉ lệ trung bình này (hình 6a) có thể được tính trên toàn bộ kết quả của quá trình crawl (100 trang đầu tiên, 200 trang đầu tiên và cứ như vậy) [21]. Đôi khi giá trị trung bình trong thời gian chạy được tính trên một cửa sổ hoặc một vài trang (ví dụ 50 trang cuối cùng từ điểm crawl hiện tại) [9]. Do các đơn vị đo tương tự (measures analogs) như recall là rất khó tính toán trong môi trường Web, các tác giả đã sử dụng tới các bộ chỉ đường gián tiếp (indirect indicator) để đánh giá độ hồi tưởng. Một số bộ chỉ đường đó là:

Độ hồi tưởng đích (target recall): Một tập các trang URL hợp lệ đã biết được chia thành 2 tập không giao nhau: các trang đích (target) và các trang hạt nhân (seed). Bộ crawler bắt đầu từ các trang hạt nhân và độ hồi tưởng của các trang đích được tính. Độ hồi tưởng đích được tính bằng:

$$target\_recall = \frac{|Pt \cap Pc|}{|Pt|} \qquad (7)$$

Trong đó: Pt là tập các trang web đích, còn Pc là tập các trang web được tải về. Độ hồi tưởng của tập đích được sử dụng như một tiêu chuẩn đánh giá độ hồi tưởng của các trang web hợp lệ. Hình 5 thể hiện một lược đồ chứng minh cho độ đo này. Chú ý rằng giả thiết được gạch chân là các trang đích chính là một tập con ngẫu nhiên của tập các trang web hợp lệ.

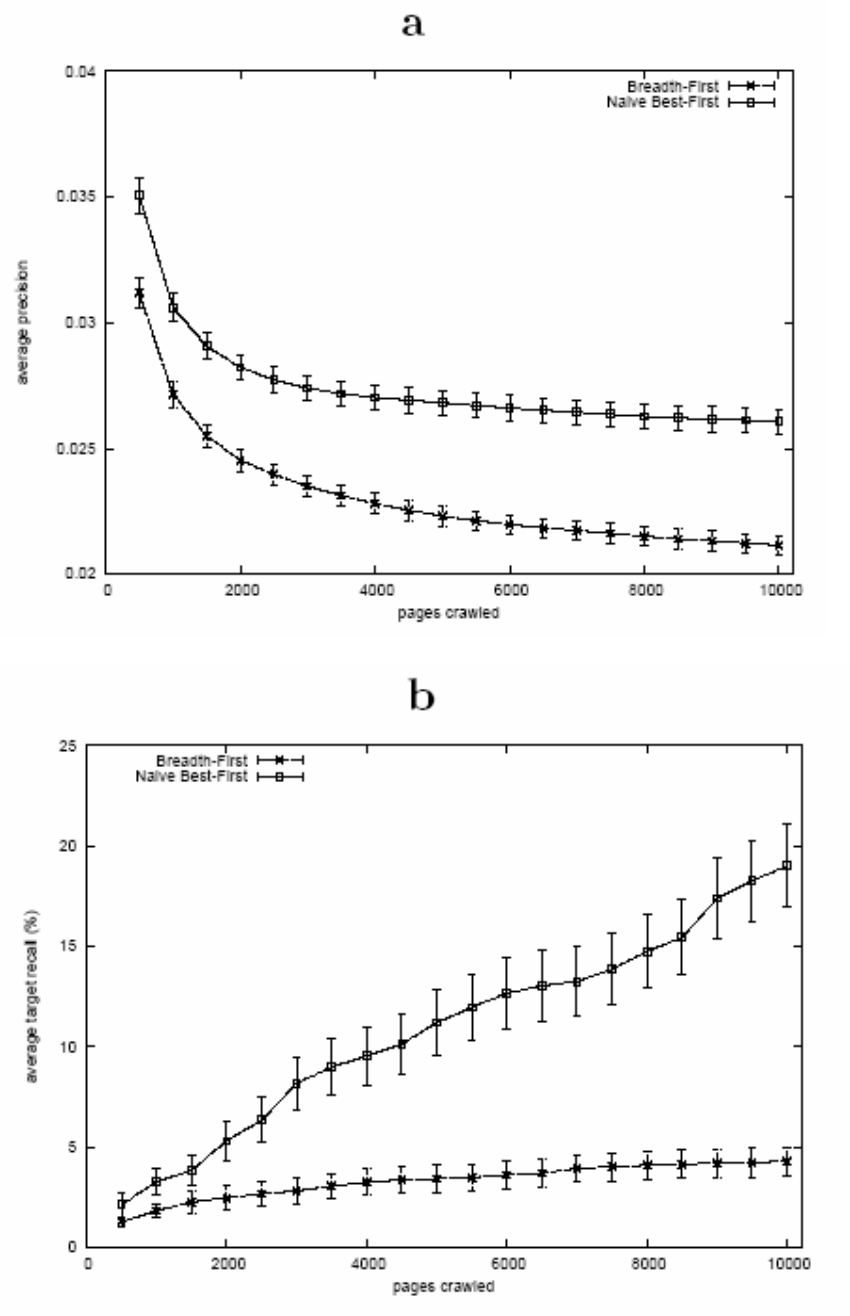


Hình 5: *Phương pháp đo hiệu năng sử dụng $|P_t \cap P_c|/P_t$ như là giá trị xấp xỉ của $|P_r \cap P_c|/P_r$*

Robustness: Tập các trang URL hạt nhân được chia thành hai tập không giao nhau Sa và Sb. Mỗi tập được sử dụng để khởi tạo một trường hợp của cùng một crawler. Sự gối nhau (overlap) của các trang đã được duyệt bắt đầu từ hai tập không giao nhau này được ước lượng. Một lượng lớn các trang web gối nhau thể hiện như là sự hiệu quả của crawler trong việc định vị các trang hợp lệ trên Web.

Ngoài ra còn có các cách thức khác để đo hiệu năng của crawler bằng cách kết hợp cả độ chính xác và độ hồi tưởng. Ví dụ, phương thức đánh giá search length [20] đo số lượng các trang đã được duyệt trước khi lấy được một tỉ lệ phần trăm nhất định các trang web hợp lệ.

Hình 6 biểu diễn một ví dụ về đồ thị thực thi của 2 crawler khác nhau []. Quá trình thực thi của các crawler được mô tả như một quĩ đạo theo thời gian (được xấp xỉ bởi các trang web đã được tải). Bộ crawler sử dụng thuật toán naïve tốt nhất đầu tiên cho ra tốt hơn hẳn so với crawler sử dụng thuật toán tốt nhất đầu tiên trong việc đánh giá khoảng 159 chủ đề và khoảng 10000 trang web được duyệt bởi mỗi crawler trong mỗi chủ đề (do đó việc đánh giá này tiến hành trên hàng triệu trang web).

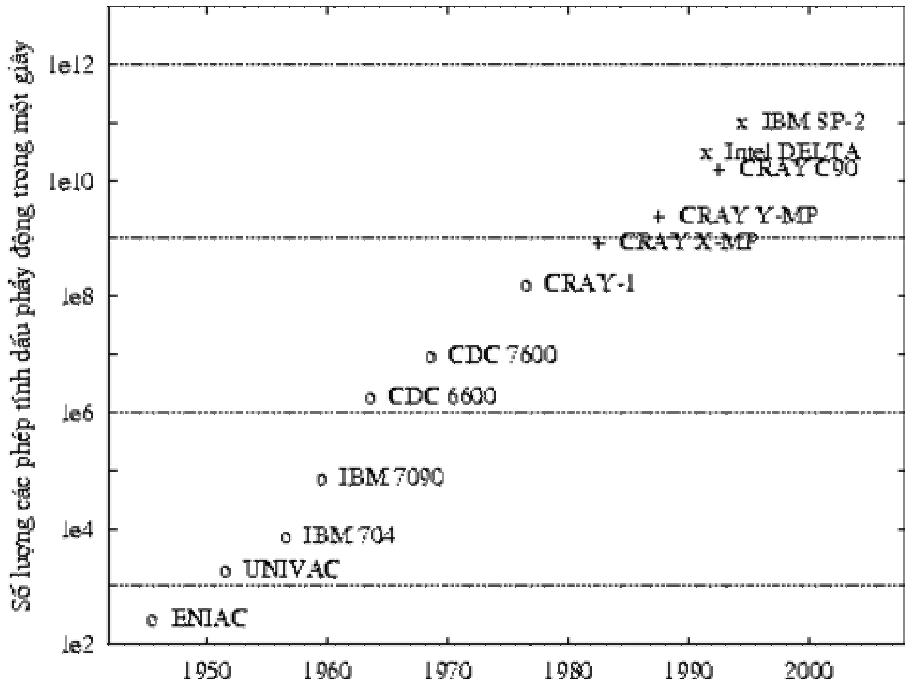


Hình 6: Sơ đồ thực thi của các crawler. (a) độ chính xác trung bình (b) độ hồi tưởng đích trung bình. Các giá trị trung bình được tính trên 159 chủ đề, sai số chuẩn ở đây là ±1, và tổng số trang tải về là khoảng 10000 trang.

Chương 3. Tổng quan về xử lý song song

3.1 Máy tính song song

Tốc độ của chiếc máy tính nhanh nhất đã tăng theo hàm mũ kể từ năm 1945 cho đến nay với tỉ lệ tăng trung bình là 10 lần trong 5 năm. Trong khi chiếc máy tính đầu tiên chỉ có thể tính toán được vài chục phép tính dấu phẩy động trong một giây, các máy tính song song ở giữa thập niên 90 đã có thể tính toán được hàng chục tỉ phép tính trong một giây. Tốc độ phát triển nhanh chóng đó cũng có thể nhận thấy trong các hệ máy tính cá nhân và các workstation. Sự tăng trưởng này sẽ vẫn còn tiếp tục, tuy nhiên đã có sự chuyển đổi to lớn trong kiến trúc của máy tính, từ kiến trúc tuần tự sang song song.



Hình 3.1: *Tốc độ của các máy tính nhanh nhất từ 1945-1995. Ký hiệu "o" thể hiện máy tính với duy nhất một bộ vi xử lý, ký hiệu "+" thể hiện các máy tính vector song song có từ 4-16 bộ xử lý, ký hiệu "x" thể hiện các máy tính song song khổng lồ với hàng trăm hoặc hàng nghìn bộ xử lý.*

Tốc độ của máy tính phụ thuộc vào thời gian cần thiết để thực hiện một thao tác cơ bản và số lượng các thao tác cơ bản có thể thực hiện được đồng thời. Rõ ràng là thời gian thực hiện một thao tác cơ bản sẽ bị giới hạn bởi chu kỳ đồng hồ của bộ xử lý,

nghĩa là thời gian để thực hiện một thao tác nguyên tố nhất. Tuy nhiên, thời gian của một chu kỳ đồng hồ đang giảm đi rất chậm và dường như sắp tiếp cận tới giới hạn vật lý. Do vậy chúng ta không thể dựa trên những bộ xử lý nhanh hơn để làm tăng tốc độ tính toán.

Nhằm vượt qua những giới hạn này, các nhà thiết kế đã sử dụng khả năng tính toán song song trong một con chip, chẳng hạn tiến hành tính toán đồng thời trên cả 64 bit trong thao tác nhân hai số. Tuy nhiên việc xây dựng các thành phần (component) riêng rẽ chạy nhanh hơn không những là rất khó khăn mà việc thực hiện điều đó cũng không kinh tế. Thay vào đó việc sử dụng đồng thời nhiều thành phần có tốc độ chậm hơn sẽ rẻ hơn rất nhiều.

Các nhà thiết kế máy tính đã sử dụng nhiều kỹ thuật khác nhau để làm tăng tốc độ của một máy tính đơn như cơ chế pipeline, hay sử dụng nhiều đơn vị tính toán (multi function units). Một xu hướng nữa là sử dụng nhiều máy tính mà mỗi máy tính trong số đó có bộ xử lý, bộ nhớ riêng rẽ và được kết nối theo một logic nào đó.

Cách tiếp cận này ngày càng trở nên phổ biến hơn do các tiến bộ của kỹ thuật VLSI []cho phép giảm số lượng các thành phần cần thiết cho một máy tính. Do giá thành của máy tính tỉ lệ với số lượng thành phần mà nó có nên việc tăng cường tích hợp cũng làm tăng số lượng các bộ xử lý trong một máy tính mà vẫn giữ được giá cả hợp lý.

Số lượng bộ xử lý trên một máy tính đang tiếp tục tăng và tỉ lệ tăng trong một số môi trường là gấp đôi trong vòng một hoặc hai năm.

3.1.2 Phân loại máy tính song song

Có một số tiêu chuẩn phân loại các máy tính song song như: phân loại dựa trên cơ chế điều khiển chung, dựa trên cơ chế tương tác giữa các BXL, kiểu và số lượng các BXL và việc thực hiện xử lý đồng bộ hay không đồng bộ.

3.1.2.1 Phân loại dựa trên cơ chế điều khiển chung.

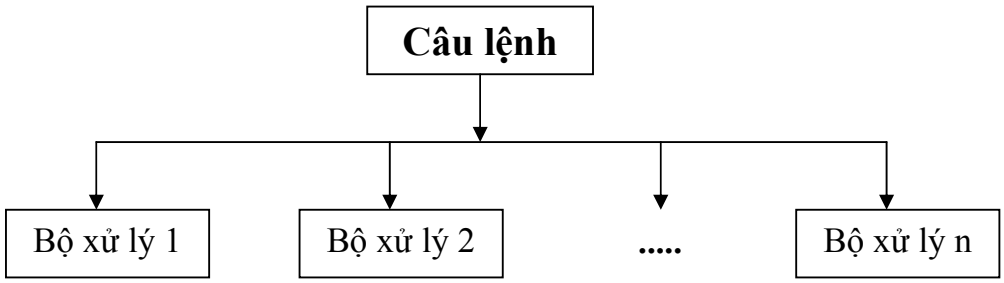
Phần lớn các hệ máy tính song song thường có một cơ chế điều khiển chung, có hai loại cơ chế điều khiển.

- Cơ chế điều khiển chung chỉ được sử dụng để nạp chương trình và dữ liệu vào các BXL còn sau đó các BXL hoạt động độc lập.
- Cơ chế điều khiển được sử dụng để hướng dẫn cho các BXL các công việc phải làm tại mỗi bước.

Hai loại cơ chế điều khiển phổ biến nhất là

a. Hệ thống đa xử lý một dòng lệnh, nhiều dòng dữ liệu.

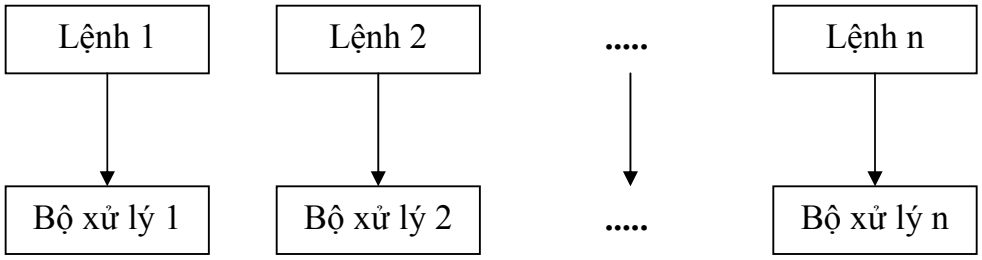
Các máy tính vector thuộc vào dạng này. Mỗi máy tính vector có thể thực hiện một dòng lệnh, tuy nhiên nó có nhiều BXL số học khác nhau mà mỗi BXL này có khả năng nạp và xử lý dữ liệu riêng của nó. Bởi vậy, trong bất kỳ thời điểm nào, một thao tác luôn ở cùng trạng thái thực thi trên nhiều đơn vị xử lý, mà mỗi đơn vị này có thể xử lý dữ liệu riêng rẽ.



Hình 3.2: *Hệ thống một dòng lệnh, nhiều dòng dữ liệu.*

b. Hệ thống đa xử lý nhiều dòng lệnh, nhiều dòng dữ liệu.

Phần lớn các máy tính đa xử lý hiện nay đều thuộc vào loại này. Trong các máy tính loại này, nhiều dòng lệnh có thể thực hiện cùng một lúc và mỗi dòng lệnh có thể xử lý dữ liệu riêng biệt. Các máy tính MIMD ban đầu có rất ít tương tác giữa các CPU song hiện nay phần lớn các máy tính đều được thiết kế cho phép tương tác giữa các CPU được thực hiện một cách hiệu quả.



Hình 3.3:*Hệ thống nhiều dòng lệnh, nhiều dòng dữ liệu*

3.1.2.2 Cách phân loại dựa trên sự tương tác giữa các BXL.

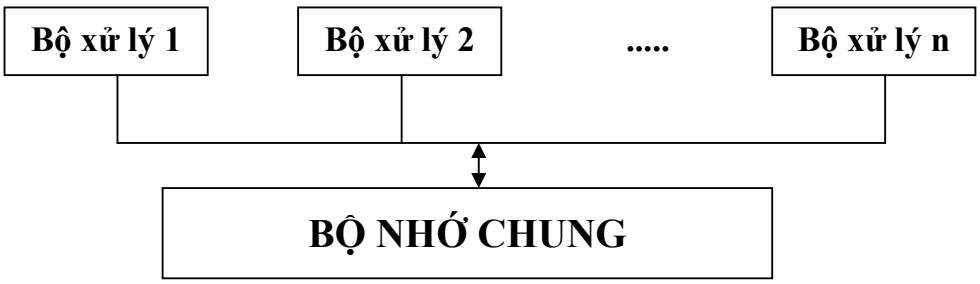
Một trong những khía cạnh quan trọng của các máy tính song song là cơ chế trao đổi thông tin giữa các BXL. Hai kiến trúc phổ biến là kiến trúc chia sẻ bộ nhớ (shared memory) và kiến trúc truyền thông điệp (message passing).

a. Chia sẻ bộ nhớ chung

Các hệ máy tính dạng này sử dụng một bộ nhớ chia sẻ toàn cục (global shared memory) mà tất cả các BXL đều có thể truy cập đến. Một BXL có thể trao đổi với một BXL khác bằng cách ghi vào bộ nhớ toàn cục và BXL thứ hai sẽ đọc tại cùng vị trí đó trong bộ nhớ. Điều này cho phép giải quyết vấn đề trao đổi thông tin giữa các BXL, tuy nhiên nó dẫn tới vấn đề truy nhập đồng thời các vị trí khác nhau trong bộ nhớ bởi nhiều BXL. Có hai cách tiếp cận chủ yếu để xử lý vấn đề truy nhập bộ nhớ là sử dụng hệ thống chuyển mạch (switching systems) hoặc các BXL truy nhập bộ nhớ thông qua bus hệ thống.

Đối với các hệ thống truy nhập bộ nhớ thông qua bus chung, việc thiết lập tương đối đơn giản, song nếu có nhiều BXL cùng truy nhập bộ nhớ thì bus có thể trở thành nút cổ chai. Bởi vậy số lượng BXL trong các hệ thống này thường tương đối nhỏ và cao nhất chỉ khoảng vài chục BXL.

Một khó khăn khác của các hệ thống này là thời gian truy nhập bộ nhớ sẽ không cao và không đồng bộ. Tuy nhiên việc sử dụng kiến trúc này khiến cho việc thiết kế giải thuật trở nên đơn giản bởi hệ thống được xử lý như là tất cả các BXL đều được nối trực tiếp với nhau.

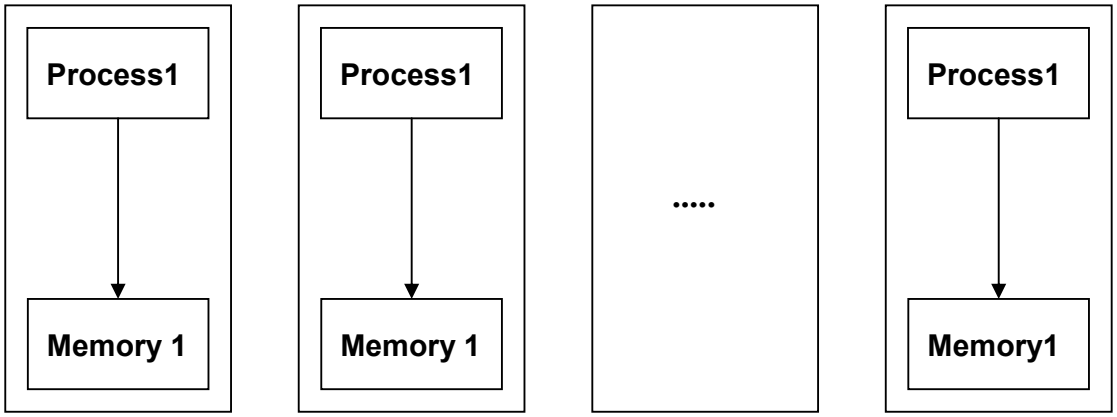


Hình 3.4: *Máy tính song song chia sẻ bộ nhớ chung*

b. Truyền thông điệp

Ngược lại với các máy tính chia sẻ bộ nhớ chung là các máy tính song song có bộ nhớ phân tán, mỗi BXL có bộ nhớ cục bộ riêng. Với kiến trúc này, việc mở rộng các

máy tính trong hệ thống là khá dễ dàng, các nhà thiết kế có thể đưa ra các hệ thống với hàng nghìn BXL mà không phải thay đổi nhiều trong cấu trúc thiết kế. Trong các hệ máy tính song song có bộ nhớ phân tán, các BXL liên lạc với nhau bằng các thông điệp (message) qua một mạng liên kết (interconnection network) gồm các liên kết truyền thông trực tiếp giữa một số cặp BXL.



Hình 3.5: *Máy tính song song có bộ nhớ phân tán*

3.2 Mô hình lập trình song song

3.2.1 Mô hình nhiệm vụ - kênh liên lạc

3.2.1.1 Đặc điểm mô hình nhiệm vụ-kênh liên lạc.

Mô hình nhiệm vụ – kênh liên lạc có thể được tóm tắt như sau:

- Một công việc tính toán song song bao gồm một hoặc nhiều nhiệm vụ. Các nhiệm vụ có thể được thực hiện đồng thời. Số lượng nhiệm vụ có thể thay đổi trong thời gian thực thi chương trình.

- Mỗi nhiệm vụ là một chương trình tuần tự và có bộ nhớ cục bộ. Một tập các cổng vào và cổng ra (inport, outport) được định nghĩa như là giao diện của nó với môi trường.

- Một nhiệm vụ có thể thực hiện 4 thao tác cơ bản ngoài việc đọc và ghi vào bộ nhớ cục bộ, đó là: gửi thông điệp tới các cổng ra, nhận thông điệp từ các cổng vào, tạo ra nhiệm vụ mới và kết thúc việc thực thi chương trình.

- Thao tác gửi dữ liệu là không đồng bộ (nó được hoàn thành ngay lập tức). Thao tác nhận dữ liệu là đồng bộ theo nghĩa nó bắt việc thực thi nhiệm vụ phải dừng lại cho tới khi nhận được thông điệp.

- Các cặp cổng vào/ cổng ra có thể được nối bởi một hàng thông điệp được gọi là một kênh liên lạc. Các kênh liên lạc có thể được tạo ra hoặc xóa đi.

- Các nhiệm vụ được ánh xạ vào các BXL vật lý theo nhiều cơ chế khác nhau, cơ chế ánh xạ được sử dụng không làm ảnh hưởng tới ngữ nghĩa của chương trình. Có thể có nhiều nhiệm vụ được ánh xạ lên một BXL.

3.2.1.2 Đặc điểm của mô hình nhiệm vụ - kênh liên lạc.

a. Tốc độ

Các khái niệm về nhiệm vụ và kênh liên lạc có thể được ánh xạ trực tiếp lên mô hình máy tính song song. Một nhiệm vụ đại diện cho một đoạn mã được thực hiện tuần tự trên một BXL. Nếu hai nhiệm vụ liên lạc với nhau qua một kênh chung được ánh xạ lên các BXL khác nhau thì kênh liên lạc giữa chúng được thể hiện như truyền thông liên BXL, nếu chúng được ánh xạ lên cùng một BXL thì một cơ chế hiệu quả hơn có thể được sử dụng.

b. Tính độc lập ánh xạ.

Trong mô hình trên, các nhiệm vụ tương tác với nhau sử dụng cùng một cơ chế là kênh liên lạc không tính đến vị trí của nhiệm vụ. Do đó, kết quả đưa ra bởi chương trình không phụ thuộc vào vị trí nhiệm vụ thực thi. Bởi vậy, việc thiết kế các giải thuật song song có thể được thực hiện mà không cần tính đến số lượng BXL trong thực tế. Thông thường, các thiết kế đều đưa đến số lượng nhiệm vụ tạo ra lớn hơn số lượng BXL có, khi đó việc mở rộng (scalability) là khá dễ dàng. Việc tạo ra nhiều nhiệm vụ hơn số BXL cũng cho phép giảm bớt chi phí truyền thông bởi khi một nhiệm vụ đang truy nhập dữ liệu ở xa thì một nhiệm vụ khác có thể thực hiện công việc tính toán.

c. Tính module

Khi thiết kế chương trình theo hướng module, các thành phần của chương trình có thể được phát triển như những module độc lập và sau đó được kết hợp lại để tạo thành chương trình. Các module có thể được thay đổi mà không cần thay đổi các thành phần khác. Các đặc tính của chương trình có thể được xác định từ đặc tả về các module và mã lệnh nối các module. Việc áp dụng có hiệu quả các thiết kế module sẽ giúp giảm bớt độ phức tạp của chương trình cũng như cho phép tái sử dụng mã lệnh.

Khái niệm một nhiệm vụ trong mô hình nhiệm vụ- kênh liên lạc phù hợp một cách tự nhiên với một module trong quá trình thiết kế. Một nhiệm vụ ở đây bao gồm

cả mã lệnh và dữ liệu cần thao tác, các cổng mà nó gửi và nhận thông điệp tạo nên giao diện của nó. Bởi vậy các ưu điểm của thiết kế module có thể được áp dụng trực tiếp trong mô hình này.

d. Tính xác định

Một giải thuật hay một chương trình được coi là xác định nếu như sự thực thi với một dữ liệu vào riêng biệt luôn đưa ra cùng một kết quả. Nếu giải thuật là không xác định thì các lần thực thi khác nhau của chương trình sẽ cho ra những kết quả khác nhau. Trong mô hình nhiệm vụ - kênh liên lạc mỗi kênh có một nhiệm vụ gửi và một nhiệm vụ nhận, khi có yêu cầu nhận dữ liệu, các xử lý khác ở nhiệm vụ nhận phải ngừng thực thi cho tới khi nhận được dữ liệu. Điều này đảm bảo cho tính xác định của chương trình.

3.2.2 Mô hình chia sẻ bộ nhớ chung.

Trong mô hình bộ nhớ chung, các nhiệm vụ cùng chia xẻ một không gian địa chỉ chung có thể được truy cập đọc ghi theo phương thức không đồng bộ. Các cơ chế khác nhau như khóa (lock), semaphore được sử dụng để điều khiển việc truy nhập tới bộ nhớ chung. Xét theo quan điểm của lập trình viên thì mô hình này có ưu điểm là không có khái niệm sở hữu dữ liệu, nghĩa là không phải chỉ định rõ ràng quá trình truyền dữ liệu giữa các nhiệm vụ. Tính chất này làm cho việc phát triển chương trình đơn giản hơn. Tuy nhiên khi đó việc hiểu và đảm bảo tính cục bộ trở nên khó khăn, đây cũng là vấn đề được chú ý nhiều nhất trong hầu hết các kiến trúc chia xẻ bộ nhớ chung. Điều này làm cho việc viết các chương trình xác định cũng trở nên khó khăn.

3.3. Hiệu năng của xử lý song song

Phần này đề cập tới một số vấn đề liên quan tới hiệu năng của xử lý song song bao gồm: khả năng tăng tốc độ tính toán, các chiến lược làm tăng tốc độ tính toán, sự cân bằng tải và sự bế tắc.

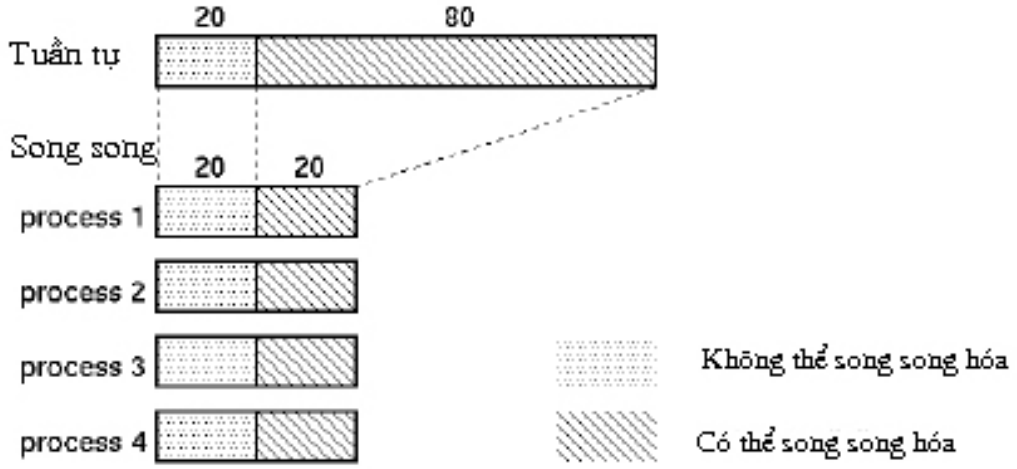
3.3.1 Khả năng tăng tốc độ tính toán:

Việc song song hóa một chương trình nhằm làm cho chương trình đó chạy nhanh hơn, tuy nhiên chương trình đó sẽ chạy nhanh hơn bao nhiêu lần? Định luật Amdahl’s [] cho phép ta xác định điều này. Giả sử xét về khía cạnh thời gian chạy chương trình, một phần p của chương trình có thể song song hóa và phần 1-p còn lại buộc phải chạy

tuần tự. Trong trường hợp lý tưởng, nếu thực thi chương trình sử dụng n bộ xử lý, thời gian chạy chương trình sẽ là:

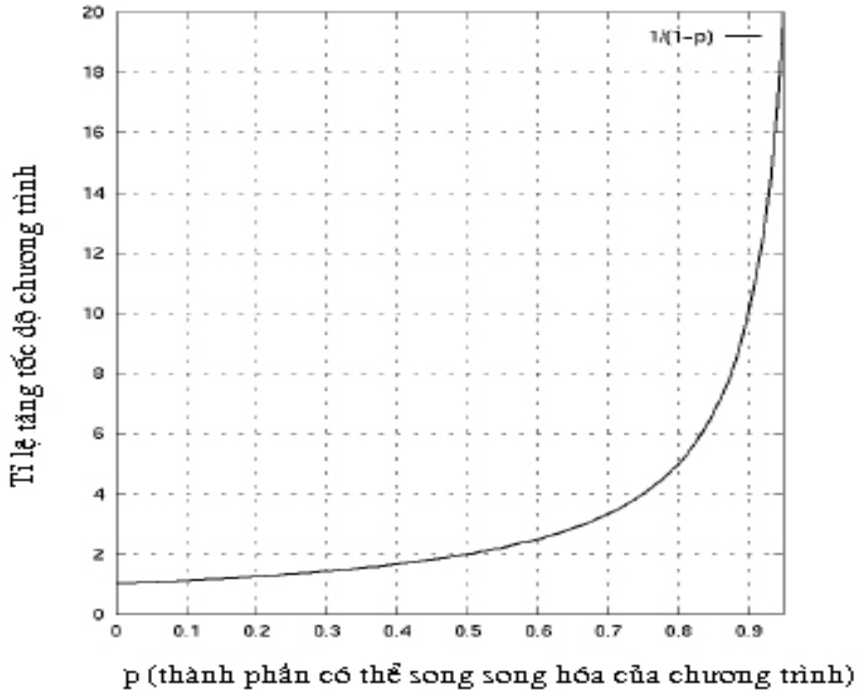
1-p + p/n

của thời gian chạy chương trình một cách tuần tự. Đây là hệ quả trực tiếp của định luật Amdahl áp dụng cho trường hợp thực thi lý tưởng. Ví dụ: nếu 80% chương trình có thể được song song hóa, và ta có 4 bộ xử lý, thời gian chạy song song sẽ là: 1 - 0.8 + 0.8/4 = 0.4 tức là bằng 40% thời gian chạy tuần tự.



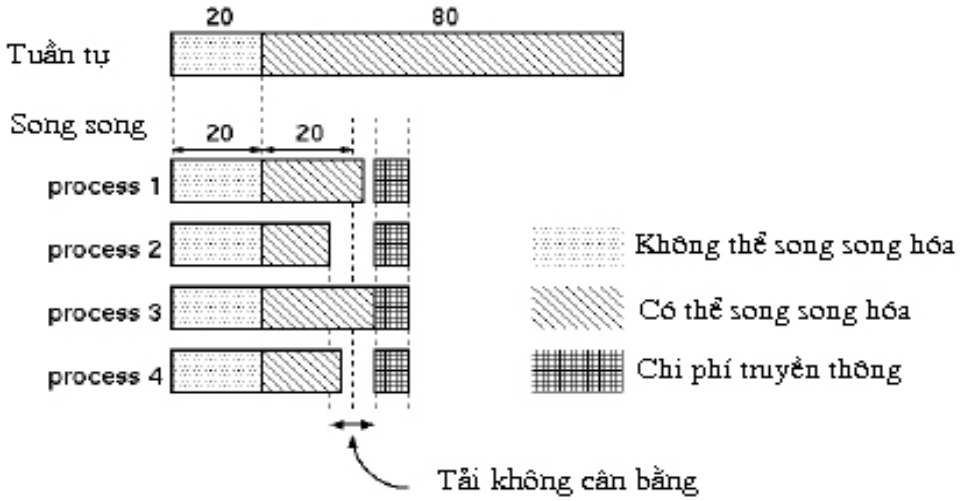
Hình 3.6: *Khả năng tăng tốc độ tính toán, trường hợp lý tưởng.*

Đối với chương trình trên, thời gian chạy song song sẽ không thể nào nhỏ hơn 20% thời gian chạy tuần tự cho dù ta sử dụng số lượng vô cùng lớn các bộ xử lý. Định luật Amdahl đã nêu lên tầm quan trọng của việc xác định các thành phần của chương trình có thể được song song hóa để cực đại chúng. Hình xx thể hiện giới hạn trên của tỉ lệ tăng tốc độ chương trình (1/(1-p)) với các giá trị khác nhau của p.



Hình 3.7: *Giới hạn trên của khả năng tăng tốc độ chạy chương trình.*

Tuy nhiên ví dụ trên chỉ là trường hợp lý tưởng hóa. Trên thực tế, khi chạy một chương trình song song, thường xuất hiện các chi phí truyền thông và việc phân công công việc không cân bằng giữa các bộ xử lý. Do đó thời gian chạy chương trình sẽ là:



Hình 3.8: *Khả năng tăng tốc độ: trường hợp thực tế.*

Do vậy để tăng tốc độ của chương trình ta cần:

- Tăng tỉ lệ (thành phần) được song song hóa của chương trình.
- Phân công công việc một cách công bằng cho các bộ xử lý.
- Giảm tới mức tối thiểu thời gian truyền thông.

3.3.3 Cân bằng tải

Giả sử rằng nếu dữ liệu được phân tán trên các bộ nhớ địa phương của các bộ xử lý trong hệ thống nhiều máy tính, khi đó khối lượng công việc của các bộ xử lý cần phải được phân phối hợp lý trong suốt quá trình tính toán. Trong nhiều trường hợp, giả sử này là đúng, tuy nhiên trong thực tế điều này không phải lúc nào cũng thực hiện được. Giải pháp được đưa ra ở đây là cân bằng tải động nhằm mục đích làm thay đổi sự phân phối khối lượng công viêc giữa các bộ xử lý trong quá trình thực hiện tính toán.

Thông thường sau khi phân phối khối lượng công việc cho các bộ xử lý, quá trình cân bằng tải động thực hiện bốn bước cơ bản sau:

- Giám sát hiệu năng của các bộ xử lý.
- Trao đổi thông tin trạng thái giữa các BXL
- Tính toán và ra quyết định phân phối lại khối lượng công việc
- Thực hiện việc chuyển đổi dữ liệu thực sự.

Để thực hiện được điều này, rất nhiều thuật toán đã được đề xuất. Người ta phân lớp các thuật toán này theo các chiến lược: tập trung, phân tán hoàn toàn (fully distributed) và phân tán một nửa (semi distributed).

a. Các thuật toán cân bằng tải tập trung.

Các thuật toán này thường đưa ra quyết định có tính chất tổng thể trong việc phân phối lại khối lượng công việc cho các bộ xử lý. Một vài thuật toán trong lớp này sử dụng thông tin hệ thống có tính toàn cục để lưu trạng thái các máy tính riêng lẻ. Thông tin này sẽ giúp thuật toán phân phối công việc một cách dễ dàng. Tuy nhiên, khối lượng thông tin tăng theo tỉ lệ thuận với số lượng các BXL, do đó nó đòi hỏi khối lượng lớn bộ nhớ trên một bộ xử lý để lưu thông tin trạng thái. Vì vậy thuật toán thuộc lớp này không được tiếp cận một cách rộng rãi.

b. Các thuật toán cân bằng tải phân tán hoàn toàn

Trong các thuật toán dạng này, mỗi bộ xử lý có một bản sao về thông tin trạng thái của hệ thống. Các bộ xử lý trao đổi thông tin trạng thái với nhau và sử dụng các thông tin này để làm thay đổi một cách cục bộ việc phân chia công việc. Tuy nhiên các bộ xử lý chỉ có thông tin trạng thái cục bộ nên việc cân bằng tải không tốt bằng các thuật toán cân bằng tải tập trung.

c. Các thuật toán cân bằng tải phân tán một nửa.

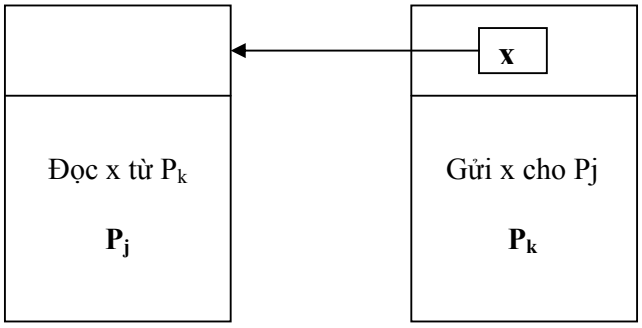
Các thuật toán thuộc lớp này chia các bộ xử lý thành từng miền. Trong mỗi miền sử dụng thuật toán cân bằng tải tập trung để phân phối công việc cho các bộ xử lý thuộc miền đó.

3.3.4 Sự bế tắc.

Các tiến trình xử lý bị rơi vào tình trạng bế tắc nếu mỗi tiến trình đó nắm giữ tài nguyên mà một vài tiến trình khác đang yêu cầu để xử lý. Lý do tiềm ẩn của sự bế tắc là do nhiều tiến trình cùng sử dụng nguồn tài nguyên chung mà không có sự kiểm soát tốt.

Đối với các hệ thống đa máy tính, một trong những sự bế tắc phổ biến nhất là bế tắc vùng đệm (buffer deadlock) xảy ra khi một tiến trình đợi một thông điệp mà thông điệp này có thể không bao giờ nhận được do vùng đệm đã đầy.

Ví dụ xét hệ thống đa máy tính với các bộ xử lý không đồng bộ. Bộ xử lý p_i gửi thông điệp cho bộ xử lý khác p_j không kết khối cho tới khi có thao tác đọc thông điệp đó. Mặt khác, khi bộ xử lý p_i gửi thông điệp cho bộ xử lý p_j, nội dung của thông điệp được lưu trong vùng đệm của hệ thống cho tới khi p_j nhận và đọc thông điệp. Giả sử rằng trong cùng một thời điểm có nhiều bộ xử lý cùng gửi thông điệp tới p_j và làm cho vùng đệm bị đầy. Việc gửi thông điệp tiếp theo chỉ thực hiện khi bộ xử lý p_j đọc một hay nhiều thông điệp.



Hình 3.9: *Bộ xử lý P_k kết khối để gửi x cho P_j, nhưng do vùng đệm P_j đầy, P_j không thể nhận được x. P_j và P_k rơi vào bế tắc.*

Giả sử bộ xử lý p_k là một trong các bộ xử lý có khả năng gửi thông điệp đến bộ xử lý p_j. Nếu p_j cố gắng đọc thông điệp do p_k gửi đến, nó sẽ bị kết khối cho tới khi nội dung thông điệp có trong vùng đệm. Rõ ràng bộ xử lý p_k bị kết khối cho tới khi p_j loại bỏ một hay nhiều thông điệp từ vùng đệm, và như vậy p_j và p_k rơi vào bế tắc.

Theo kết quả nghiên cứu của Coffman và Denning, bốn điều kiện sau là nguyên nhân gây ra bế tắc.

1. Sự loại trừ lẫn nhau: mỗi tiến trình có sự độc quyền trong việc sử dụng tài nguyên của nó.

2. Không có sự ưu tiên: Mỗi tiến trình không bao giờ giải phóng tài nguyên mà nó đang chiếm giữ cho tới tận khi không còn sử dụng chúng nữa.

3. Sự chờ đợi tài nguyên: mỗi tiến trình đang chiếm giữ tài nguyên trong khi lại chờ đợi các tiến trình khác giải phóng chúng.

4. Sự chờ đợi giữa các tiến trình: tiến trình chờ đợi tài nguyên mà tiến trình kế tiếp đang chiếm giữ mà tài nguyên đó không được giải phóng.

Một số giải pháp khắc phục sự bế tắc.

Cách tiếp cận thứ nhất là dò tìm sự bế tắc khi chúng xảy ra và cố gắng khôi phục lại. Một cách khác để tránh bế tắc là sử dụng các thông tin yêu cầu tài nguyên của các tiến trình để điều khiển sự phân phối để khi tiếp tục phân phối các tài nguyên không là nguyên nhân để các tiến trình rơi vào bế tắc. Cách tiếp cận thứ ba là ngăn cấm không để xảy ra điều kiện thứ 4 trong các điều kiện trên.

3.4 Môi trường lập trình song song.

Do yêu cầu ngày càng nhiều ứng dụng đòi hỏi phải xử lý nhanh và tiện lợi đã thúc đẩy việc xây dựng các mô hình lập trình song song mới. Các môi trường mới này sử dụng các ngôn ngữ bậc cao như Fortran, C, C++ tạo điều kiện thuận tiện cho người lập trình điều khiển việc thực hiện các nhiệm vụ trong chương trình, đa dạng hóa cách thức truyền thông giữa các nhiệm vụ.

Trong luận văn này, tôi xin trình bày một số môi trường lập trình phổ biến dựa trên việc sử dụng bộ nhớ phân tán. Trong mô hình này, mỗi bộ xử lý có bộ nhớ cục bộ riêng và không có sự chia sẻ. Kiến trúc này có khả năng mở rộng rất cao.

3.4.1 Mô hình MPI (Message Passing Interface).

MPI là chuẩn truyền thông điệp được phát triển bởi nhóm các nhà nghiên cứu từ các phòng thí nghiệm, và các trường đại học. MPI sử dụng mô hình truyền thông giữa các nhiệm vụ với nhau thông qua việc gửi các thông điệp. Mục tiêu của MPI là xây dựng các môi trường lập trình có tính khả chuyển và hiệu năng cao.

Thư viện các hàm của MPI bao gồm rất nhiều hàm hỗ trợ cho việc truyền thông giữa các nhiệm vụ trong chương trình song song.

Một chương trình song song viết trên môi trường MPI gồm nhiều nhiệm vu, các nhiệm vụ này có thể giống hoặc khác nhau trong việc thực thi. Mỗi nhiệm vụ cần khai báo trong môi trường MPI trước khi sử dụng các hàm thư viện. Khi đã được khai báo trong môi trường,các nhiệm vụ có thể bắt đầu truyền thông với các nhiệm vụ khác một cách trực tiếp thông qua mô hình truyền thông điểm–điểm hoặc với các thành viên trong nhóm sử dụng các hàm hỗ trợ truyền thông.

Ví dụ, thành phần crawler trong ở chương sau ...

3.4.2 PVM (Parallel Virtual Machine)

PVM sử dụng môi trường truyền thông điệp, có khả năng kết hợp một mạng các máy tính không thuần nhất thành một tài nguyên tính toán, do đó được gọi là môi trường máy ảo song song.

Môi trường PVM gồm 3 thành phần chính: thứ nhất là chương trình thường trú chạy trên tất cả các máy trong hệ thống máy ảo song song, thành phần thứ hai là thư viện PVM, thành phần thứ ba là PVM console cho phép người dùng dễ dàng khởi tạo, cấu hình hệ thống máy ảo.

Ưu điểm lớn nhất của PVM là tính khả chuyển cao và khả năng tích hợp các hệ thống không thuần nhất. Không những các chương trình PVM chạy được trên bất cứ hệ thống nào mà nó hỗ trợ mà các nhiệm vụ trong một chương trình còn có khả năng chạy trên các hệ thống khác nhau tại cùng một thời điểm.

3.4.3 So sánh giữa MPI và PVM.

Tính năng	PVM	MPI
Hỗ trợ đa nền – portbility	Có	Có

Mạng không đồng nhất	Có	Không
Ngôn ngữ không đồng nhất	Có	Không
Truyền thông điệp	Có	Xuất sắc
Điểm – điểm Nhóm Ngữ cảnh	Có Có Đơn giản	Xuất sắc Có Có
Tốc độ truyền thông điệp	Tốt	Xuất sắc
Topology truyền thông logic	Không	Có
Phát hiện và phục hồi lỗi	Có	Kém hơn
Khả năng debug	Tốt	Kém
Quản trị tài nguyên, điều khiển tiến trình	Tốt	Trung bình
Message Handler	Có	Có

Bảng 3.1: *so sánh một số tính năng giữa PVM và MPI*

3.5 Giao thức truyền thông điệp MPI

Giao thức truyền thông điệp MPI là một thư viện các hàm và macro có thể được gọi từ các chương trình sử dụng ngôn ngữ C, Fortran, và C++. Như tên gọi của nó MPI được xây dựng nhằm sử dụng trong các chương trình để khai thác hệ thống các bộ xử lý bằng cách truyền thông điệp.

MPI được phát triển trong 1993-1994 bởi Message Passing Interface Forum (MPIF). Nó là tiêu chuẩn đầu tiên cho việc lập trình trên các bộ xử lý song song. Mục đích là tạo ra một giao thức khả chuyển để viết các chương trình sử dụng truyền thông điệp, và hướng tới tính thực thi, tính hiệu quả, và tính linh hoạt cùng một lúc. MPIF với sự tham gia của hơn 40 tổ chức hoạt động trong công nghiệp, các tổ chức chính phủ và hàn lâm đã cùng làm việc và đưa ra phiên bản đầu tiên vào 1994.

Mục tiêu thiết kế của MPI bao gồm:

- Thiết kế một giao diện lập trình ứng dụng. Mặc dù MPI gần đây được sử dụng như một chương trình dịch và một thư viện ở thời gian chạy, nhưng thiết kế của MPI chủ yếu phản ánh nhu cầu nhận thức của những người lập trình ứng dụng.

- Cho phép truyền thông một cách hiệu quả: tránh việc copy dữ liệu từ bộ nhớ sang bộ nhớ và cho phép gối chồng (overlap) giữa các tính toán và truyền thông và offload để truyền thông đồng xử lý khi có thể.

- Cho phép thực thi trên một môi trường không đồng nhất.

- Có thể được gắn kết dễ dàng vào trong các chương trình ngôn ngữ C và Fortran.

- Cung cấp một giao thức truyền thông tin cậy: người dùng không cần phải lo lắng tới thất bại trong truyền thông. Các thất bại này được xử lý bởi các hệ thống truyền thông cơ sở phía sau.

- Định nghĩa một giao thức không quá khác biệt so với các môi trường song song hiện tại như PVM, NX, Express..... và cung cấp các mở rộng cho phép độ linh hoạt cao hơn.

- Định nghĩa một giao thức có thể được thực thi trên các môi trường (flatform) của nhiều nhà cung cấp mà không cần thay đổi nào đáng kể trong truyền thông cơ sở và phần mềm hệ thống.

- Ngữ nghĩa của giao thức là độc lập với ngôn ngữ.

- Giao thức được thiết kế cho phép sử dụng luồng một cách an toàn.

Các tính năng chủ yếu của MPI[2]

- Một lượng lớn các hàm truyền thông điểm – điểm (phong phú hơn rất nhiều so với các môi trường lập trình song song khác).

- Một lượng lớn các thủ tục truyền thông chọn lọc, được sử dụng để giao tiếp giữa một nhóm các bộ xử lý trong hệ thống.

- Một ngữ cảnh truyền thông hỗ trợ cho việc thiết kế các thư viện phần mềm song song.

- Có khả năng xác định các topology truyền thông.

- Có khả năng định nghĩa các kiểu dữ liệu mới để mô tả các thông báo dựa trên các dữ liệu không liên tục.

Các tiến trình

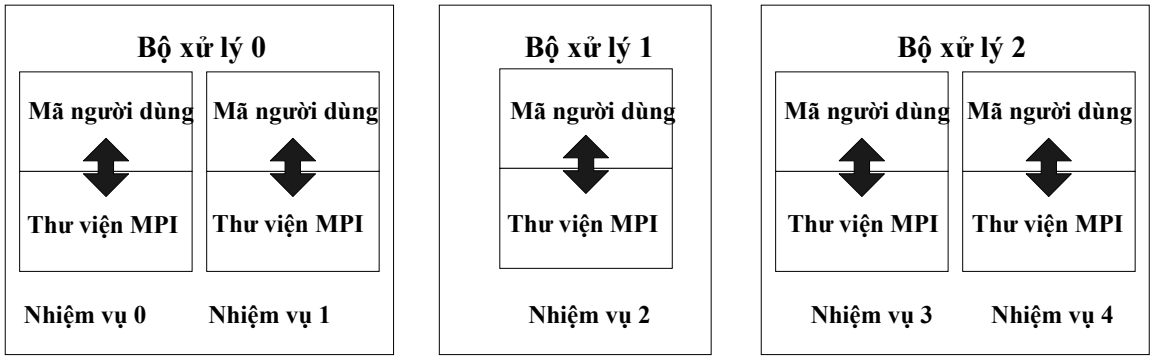
Một chương trình MPI bao gồm các bộ xử lý độc lập, thực thi các mã lệnh riêng của chúng trong một mô hình MIMD. Các tập lệnh được thực thi bởi các bộ xử lý không

nhất thiết phải giống nhau, việc truyền thông giữa các bộ xử lý được thực hiện thông qua việc gọi các hàm truyền thông của MPI.

MPI không định rõ kiểu thực thi cho mỗi bộ xử lý. Bộ xử lý A có thể chạy tuần tự, hoặc có thể thực thi ở dạng đa luồng với các luồng được kích hoạt đồng thời. Điều này thực hiện được do tính chất luồng an toàn "thread-safe" của MPI bằng cách tránh sử dụng các trạng thái tuyệt đối.

Ứng dụng MPI.

Một ứng dụng MPI có thể được thực thi như là một tập các nhiệm vụ truyền thông đồng thời. Một chương trình bao gồm các đoạn mã của người lập trình được liên kết với các hàm thư viện được cung cấp bởi phần mềm MPI. Mỗi nhiệm vụ được chỉ định một thứ hạng (rank) duy nhất trong khoảng 1-> n-1 với các ứng dụng có n nhiệm vụ. Các hạng này được sử dụng để xác định các nhiệm vụ MPI khác nhau trong việc gửi và nhận tin cũng như thực hiện các thao tác truyền thông nói chung. Nhiệm vụ MPI có thể chạy trên cùng bộ xử lý hoặc các bộ xử lý khác nhau một cách đồng thời. Lợi ích của các rank là làm cho thao tác phối hợp độc lập với vị trí vật lý của các thành phần.



Hình 3. : *Ví dụ về 5 nhiệm vụ MPI chạy trên 3 bộ xử lý.*

Chương 4. Giới thiệu về máy tìm kiếm ASPseek và đề xuất giải pháp song song hóa

4.1 Giới thiệu chung về máy tìm kiếm ASPseek

Aspseek là một công cụ tìm kiếm mã nguồn mở trên Internet với rất nhiều đặc tính ưu việt mà người dùng mong đợi ở một công cụ tìm kiếm. Aspseek được viết bằng ngôn ngữ C++, sử dụng thư viên STL chuẩn, và sử dụng phương pháp lưu trữ pha trộn giữa các bảng cơ sở dữ liệu SQL và các file nhị phân. Aspseek bao gồm các thành phần: chương trình đánh chỉ mục index, module tìm kiếm searchd, và module tìm kiếm front-end s.cgi.

4.1.1 Một số tính năng của ASPseek.

a. Có khả năng đánh chỉ mục và tìm kiếm trong vài triệu tài liệu: Sử dụng Aspseek, ta có thể xây dựng một cơ sở dữ liệu và tìm kiếm trong rất nhiều site, và kết quả trả về cho mỗi câu truy vấn là rất nhanh ngay cả khi ta có hàng triệu trang web đã được đánh chỉ mục.

b. Tối ưu các kết quả trả về: Mục đích của một công cụ tìm kiếm là tìm được những gì mà người dùng yêu cầu. Các kết quả trả về của Aspseek được sắp xếp theo mức độ hợp lệ của trang web so với câu truy vấn của người dùng.

c. Khả năng tìm kiếm nâng cao: Hỗ trợ việc tìm kiếm theo cụm từ, để tìm kiếm một cụm từ, người dùng chỉ việc bao cụm từ bởi các dấu ngoặc kép (“”), chẳng hạn “many years ago”. Ngoài ra người dùng có thể thực hiện tìm kiếm theo các ký tự đại diện. Ví dụ nếu chúng ta biết chính xác cụm từ nhưng lại quên một từ ở giữa, chúng ta có thể thay thế từ đó bởi ký hiệu (*). Do đó, câu truy vấn “many * ago” sẽ trả lại các trang web có các cụm từ như “many years ago”, “many days ago”.

Hỗ trợ tìm kiếm sử dụng các biểu thức logic. Các biểu thức có thể kết hợp với nhau sử dụng các toán tử AND và OR. Các biểu thức con có thể được nhóm lại sử dụng các dấu ngoặc đơn:

Ví dụ: (some OR any) AND (days OR months OR years)

d. Hỗ trợ việc lưu trữ theo định dạng Unicode: Aspseek lưu trữ thông tin các trang web ở dạng Unicode, do đó ta có thể tìm kiếm và đánh chỉ mục các văn bản thuộc nhiều ngôn ngữ như tiếng Anh, tiếng Nga, tiếng Trung Quốc... trong cùng một cơ sở dữ liệu.

e. Hỗ trợ các giao thức HTTP, HTTPS, HTTP proxy và FPT proxy đồng thời có khả năng nhận dạng các văn bản ở định dạng HTML và plain text. Các định dạng văn bản khác có thể được hỗ trợ thông qua các chương trình mở rộng chuyển sang định dạng HTML hoặc plain text.

f. Được thiết kế chạy đa luồng: ASPseek sử dụng đa luồng POSIX, mỗi tiến trình có rất nhiều luồng chạy song song. Điều này không chỉ giúp cải thiện rất lớn tốc độ của quá trình đánh chỉ mục, do nếu chỉ chạy một luồng thì phần lớn thời gian là chờ tiếp nhận dữ liệu từ mạng.

g. Hỗ trợ việc đánh chỉ mục không đồng bộ theo thời gian thực: Một số máy tìm kiếm yêu cầu việc tìm kiếm phải dừng lại trong suốt thời gian cập nhật cơ sở dữ liệu. ASPseek không đòi hỏi điều này bằng cách hỗ trợ chế độ thời gian thực cho modul đánh chỉ mục. Tính năng này sẽ rất có ích khi chúng ta đang xây dựng một máy tìm kiếm chuyên biệt cho các trang Web có nội dung thay đổi liên tục ví dụ như các trang tin trực tuyến. Tuy nhiên số lượng tài liệu trong cơ sở dữ liệu thời gian thực bị giới hạn vào khoảng 1000 tài liệu. Nếu có càng nhiều tài liệu trong cơ sở dữ liệu thời gian thực thì tốc độ index vào cơ sở dữ liệu chính sẽ càng bị chậm.

h. Xử lý các từ dừng và đoán nhận mã chữ cái: Từ dừng là các từ mà bản thân nó không có ý nghĩa. Ví dụ các từ dừng trong tiếng Anh: “is, are, at, this”...Việc tìm kiếm trên các từ dừng là hoàn toàn vô nghĩa, bởi vậy các từ dừng sẽ bị loại bỏ khỏi câu truy vấn. Các từ dừng cũng bị loại bỏ ra khỏi cơ sở dữ liệu trong suốt quá trình đánh chỉ mục bởi vậy cơ sỡ dữ liệu sẽ nhỏ hơn và nhanh hơn.

Một số server cấu hình không đúng sẽ không cho phía client biết tập mã ký tự của nội dung mà nó cung cấp. Trong trường hợp này, ASPseek sẽ sử dụng bộ đoán nhận mã ký tự để xác định tập ký tự đúng của văn bản.

4.1.2 Các thành phần của ASPseek

a. Module đánh chỉ số (indexing).

Index là một thành phần của công cụ tìm kiếm ASPseek có nhiệm vụ thực hiện việc duyệt Web, tải nội dung các trang Web, phân tích nội dung các trang này và lưu trữ vào trong cơ sở dữ liệu. Module này cũng được sử dụng để quản lý cơ sở dữ liệu của ASPseek. Như vậy trong ASPseek, quá trình crawler và quá trình index được tích hợp trong cùng một module, và được thực hiện kế tiếp nhau.

Trong suốt quá trình crawler, module index duyệt lần lượt các trang Web, tải và lưu trữ các văn bản tìm được trong một cấu trúc dữ liệu đặc biệt gọi là các file delta, cũng như lưu vào trong cơ sở dữ liệu.

Khi quá trình duyệt web kết thúc, module này sẽ sắp xếp nội dung các file delta, và trộn nội dung các file này vào các file nhị phân có cấu trúc tiện lợi cho việc tìm kiếm (quá trình index ngược), cũng như tính hạng (PageRank)của các trang Web trong cơ sở dữ liệu.

Các thao tác trong quá trình index đều có thể điều khiển thông qua file cấu hình aspseek.conf, file này sẽ được đọc trong quá trình khởi động.

b. Module tìm kiếm (searchd).

Module searchd lắng nghe và xử lý các câu truy vấn tìm kiếm được cung cấp bởi module giao diện tìm kiếm front-end s.cgi. Module front-end sau đó sẽ định dạng lại các kết quả tìm kiếm được bởi module searchd thành các trang HTML và trả về cho người dùng.

Aspseek được tối ưu hóa cho cho việc tìm kiếm trên nhiều site với tốc độ tải trung bình, và có thể được sử dụng để tìm kiếm trên vài triệu trang web (Urls). Người dùng có thể tìm kiếm theo các từ hoặc cụm từ, sử dụng các ký tự đại diện và thực hiện các câu tìm kiếm với các mệnh đề điều kiện.

Phạm vi tìm kiếm có thể được hạn chế trong một khoảng thời gian xác định, hạn chế theo site, hoặc một phần nhỏ của site, hạn chế theo không gian web (một tập các site) hoặc một phần cụ thể của tập các trang HTML phân loại theo chủ đề, theo body, theo phần mô tả (description), hoặc theo từ khóa. Kết quả tìm kiếm có thể được sắp xếp theo thuộc tính phân loại hoặc theo ngày tháng.

c. Module tìm kiếm s.cgi.

Module này có nhiệm vụ nhận các câu truy vấn của người dùng, chuyển chúng cho module chạy ngầm searchd. Sau đó nhận các kết quả tìm kiếm trả về từ module searchd và định dạng lại các kết quả này thành các trang HTML và trả về cho người dùng.

4.2 Cấu trúc cơ sở dữ liệu trong máy tìm kiếm ASPseek.

Nhiệm vụ của các máy tìm kiếm là tải nội dung các trang Web trên Internet sau đó lưu vào các cơ sở dữ liệu cục bộ để phục vụ các truy vấn của người dùng. Do đó

cách thức biểu diễn trang web cũng như cách tổ chức lưu trữ dữ liệu đóng vai trò rất quan trọng trong việc nâng cao chất lượng của máy tìm kiếm. Ở đây chúng tôi xin giới thiệu cách thức tổ chức dữ liệu của máy tìm kiếm ASPseek.

Aspseek lưu trữ các thông tin về các trang web sử dụng phương pháp lưu trữ pha trộn giữa các bảng cơ sở dữ liệu SQL và các file nhị phân. Mục đích của việc lưu thông tin trong các file nhị phân là giúp giảm bớt gánh nặng về kích thước cho cơ sở dữ liệu MySQL (do mỗi bảng trong cơ sở dữ liệu MySQL chỉ có kích thước lớn nhất là 4GB) đồng thời nó cũng giúp nâng cao tốc độ tìm kiếm các trang web [](Aspseek manual). Nội dung lưu trong các file nhị phân được sử dụng chủ yếu cho quá trình tìm kiếm, các file này được lưu trong thư mục /usr/local/aspseek/var/aspseek12/.

4.2.1 Cấu trúc một số bảng chính trong cơ sở dữ liệu của ASPseek.

Bảng urlword: bảng này chứa thông tin tổng quan về tất cả các URL đã hoặc chưa được đánh chỉ số bởi máy tìm kiếm ASPseek, thỏa mãn một điều kiện đặc biệt nào đó do người dùng chỉ định. Các thông tin chi tiết hơn sẽ được lưu trữ trong các bảng urlwordsNN.

Tên trường	Miêu tả
url_id	Số định danh của URL
site_id	Số định danh của site
deleted	=1 nếu máy chủ trả về lỗi 404 hay do file "robots.txt" không cho phép được đánh chỉ số trang Web này
url	Nội dung của chính URL
next_index_time	Thời điểm tiếp theo cần index, tính theo giây
status	Trạng thái HTTP trả về bởi máy chủ hoặc 0 nếu trang này chưa được đánh chỉ số
crc	chuỗi đại diện MD5 của tài liệu
last_modified	Thời gian thay đổi nội dung gần đây nhất, được trả về từ server.
etag	tiêu đề "Etag" được trả về bởi máy chủ
last_index_time	thời điểm tiến hành đánh chỉ số cuối cùng
referre	Số định danh của URL tham chiếu đầu tiên đến trang Web này
hops	độ sâu của URL trong cây siêu liên kết

redir	=URLID mới nếu trang Web này bị chuyển hướng nếu không sẽ bằng 0
origin	=URLID của trang Web ban đầu nếu trang Web này là một bản sao, nếu không có giá trị bằng 0.

Bảng UrlwordNN (NN là các số từ 00 – 15): Các bảng này chứa các thông tin chi tiết về nội dung các Url đã được đánh chỉ số trong cơ sở dữ liệu. Việc một url được ghi vào bảng nào trong 16 bảng này phụ thuộc vào giá trị url_id mod 16.

Tên trường	**Miêu tả**
url_id	Số định danh của URL
deleted	Được đặt bằng 1 nếu máy chủ trả về lỗi, hoặc do file "robots.txt" không cho phép đánh chỉ số trang Web này.
wordcount	Số lượng các từ khác nhau trong nội dung đã được index của trang
totalcount	Tổng tất cả các từ trong nội dung đã được đánh chỉ số của trang
content-type	Tiêu đề "Content-Type" được trả về bởi máy chủ
charset	Bộ chữ cái được sử dụng trong nội dung tài liệu, thông tin này được lấy từ thẻ META
title	128 ký tự đầu tiên trong tiêu đề của trang Web
txt	255 ký tự đầu tiên, không tính các thẻ HTML, trong nội dung của trang Web.
docsize	Kích thước của trang Web.
keywords	255 ký tự đầu tiên từ các từ khóa của trang Web.
description	100 ký tự đầu tiên trong phần mô tả trang Web
words	Nội dung đã được nén của các URL
hrefs	Danh sách đã sắp xếp các URL liên kết ra (outgoing) từ trang này

Bảng wordurl: chứa thông tin về mỗi từ khóa (không phải từ dừng) xuất hiện trong các trang Web được tải.

Tên trường	**Miêu tả**

word	bản thân các từ khóa, không phải từ dừng
word_id	Số định danh của từ(khóa chính)
urls	Thông tin về các site và các url mà từ khóa này xuất hiện.Trường này sẽ rỗng nếu như kích thước của nó lớn hơn 1000 byte, trong trường hợp này thông tin sẽ được lưu trữ trong các file nhị phân.
urlcount	Số lượng các url có chứa từ khóa này
totalcount	Tổng số lần xuất hiện của từ khóa này trong tất cả các tài liệu.

Bảng wordurl1: chứa thông tin các từ khóa trong cơ sở dữ liệu thời gian thực

Tên trường	Miêu tả
word	Nội dung các từ khóa (không phải từ dừng)
word_id	Số định danh của từ (khóa chính)
urls	Thông tin về các site và các url mà từ khóa này xuất hiện.Trường này luôn luôn khác rỗng, bất kể kích thước của nó.
urlcount	Số lượng các url có chứa từ khóa này
totalcount	Tổng số lần xuất hiện của từ này trong tất cả các tài liệu đã index.

Bảng Stat: chứa thông tin thống kê về các câu truy vấn của người dùng.

Tên trường	**Miêu tả**
addr	Địa chỉ IP của máy tính có câu truy vấn tới máy tìm kiếm ASPSeek
proxy	Địa chỉ IP của máy chủ proxy
query	Nội dung câu truy vấn
ul	Giới hạn về URL được sử dụng để áp đặt lên câu truy vấn
sp	Không gian Web được áp đặt lên câu truy vấn
site	SiteID dùng để hạn chế không gian tìm kiếm
sites	Số lượng các site tìm thấy thỏa mãn câu truy vấn
urls	Số lượng các Url tìm thấy thỏa mãn câu truy vấn
referer	URLID của các trang Web có các yêu cầu truy vấn

4.2.2 Cấu trúc một số file nhị phân trong cơ sở dữ liệu của ASPseek

Cấu trúc các file trong thư mục này như sau:

- 100 thư mục 00w -> 99w:các thư mục chứa nội dung đã được index ngược của các trang web, phục vụ cho việc ánh xạ từ các từ khóa sang các địa chỉ URL.

- Thư mục citations: chứa các file nhị phân phục vụ cho quá trình tính hạng (ranking) của các trang web.

- Thư mục deltas: chứa các file nhị phân trung gian trong quá trình index, sau khi quá trình index kết thúc nộidung các file này sẽ bị xóa bỏ.

4.2.2.1 Cấu trúc các file nhị phân trong thư mục xxw:

Các file nhị phân trong thư mục xxw có nhiệm vụ lưu nội dung đã được index ngược của các trang web đã được index. Nội dung các file này chính là giá trị của trường urls trong bảng wordurl trong trường hợp kích thước của trường lớn hơn 1000 bytes. Mục đích của nó là phục vụ cho quá trình tìm kiếm các trang web theo từ khóa của người dùng. Các file này được cấu trúc theo cách thức có thể dễ dàng tìm ra các url_id có chứa từ khóa word_id, đồng thời ta cũng có thể dễ dàng tìm được số lượng cùng vị trí xuất hiện của các word_id đó trong từng url_id.

Aspseek có 2 cơ chế để lưu các file nhị phân, đó là các cơ chế lưu sử dụng CompactStorage và không sử dụng CompactStorage. Chế độ mặc định là có sử dụng. Người dùng có thể bật tắt chế độ này bằng việc điều chỉnh tham số trong file cấu hình hệ thống là aspseek.conf và search.conf. Cơ chế không CompactStorage được giữ lại để tương thích với các phiên bản cũ của Aspseek, còn các phiên bản mới đều mặc định sử dụng chế độ CompactStorage do chế độ này ngoài việc làm giảm lượng bộ nhớ cần để lưu trữ các thông tin, nó còn giúp làm tăng đáng kể tốc độ tìm kiếm các trang web. Sau đây tôi xin giới thiệu cả hai cơ chế này:

a. Cách lưu các file nhị phân theo cơ chế thông thường.

Thông tin index ngược về mỗi từ khóa trong cơ sở dữ liệu được lưu trong một file nhị phân riêng biệt có tên trùng với word_id của từ đó. Nội dung file này được trong thư mục nnw với nn= word_id mod 100. Thông tin về một word_id chỉ được lưu trong file nhị phân khi nội dung của trường urls trong bảng wordurl lớn hơn 1000 byte.

Cấu trúc của các file này là:

Các thông tin về site, được sắp xếp theo site_id		
Offset	**Độ dài**	**Miêu tả chi tiết**
0	4	Giá trị offset bắt đầu thông tin về site thứ nhất nơi từ xuất hiện
4	4	Mã nhận dạng url_id của site thứ nhất nơi từ xuất hiện
8	4	Giá trị offset bắt đầu thông tin về site thứ hai nơi từ xuất hiện
12	4	Mã nhận dạng của site thứ 2 nơi từ xuất hiện
.....		
(N-1)*8+4	4	Giá trị offset bắt đầu thông tin về site thứ N, N là tổng số các site mà từ xuất hiện
(N-1)*8+8	4	Mã nhận dạng của site thứ N nơi từ xuất hiện
Thông tin về các URL, được lưu trữ tiếp ngay sau thông tin về site. Giá trị offset được tính từ 0.		
0	4	url_id của trang thứ nhất trong site thứ nhất trong phần thông tin về các site.
4	2	Tổng số từ trong URL này
6	2	Vị trí thứ nhất
8	2	Vị trí thứ hai
.....		
6+(N-1)*2	2	Vị trí thứ N, N là tổng số từ xuất hiện trong URL
Lặp lại với các thông tin cho các URL của cùng site, nhưng có url_id lớn hơn		
.....		
Lặp lại với các thông tin về URL của site tiếp theo trong phần thông tin về site		

b. Cách lưu các file nhị phân theo cơ chế CompactStorage.

Thay vì lưu thông tin về mỗi từ trong một site như trên, nội dung index ngược của tất cả các từ có cùng giá word_id mod 100 được lưu chung trong 3 file trong cùng một thư mục file nhị phân có chỉ số bằng word_id mod 100. Đó là các file: ind, sites, urls. Nội dung của 3 file như sau:

i. File ind: chứa thông tin về các word có cùng giá trị word_id mod 100. Nội dung là một dãy liên tiếp các phần tử, mỗi phần tử mô tả thông tin về một word và có kiểu struct WordInd, kiểu này có dạng:

```
Struct {
     ULONG m_offset;
     ULONG m_siteCount;
     ULONG m_urlCount;
     ULONG m_totalCount;
}WordInd;
```

Trong đó: m_offset: vị trí kết thúc nội dung về word_id đó trong bảng sites.

m_siteCount: số lượng các site có chứa các url chứa từ khóa đó.

m_urlCount: tổng số các url có chứa từ khóa đó trong tất cả các site.

m_totalCount: tổng số lần xuất hiện của word_id đó trong tất cả các url, trong tất cả các sites.

ii. File sites:chứa thông tin về các site mà site này có chứa các url chứa một word nào đó ở file ind ở trên, nội dung là một dãy liên tiếp các phần tử, mỗi phần tử mô tả thông tin một site và có kiểu SiteInd, kiểu này có dạng:

```
Struct{
     ULONG m_siteID;
     ULONG m_offset;
}SiteInd;
```

Trong đó: m_siteID: site_id của site chứa từ khóa.

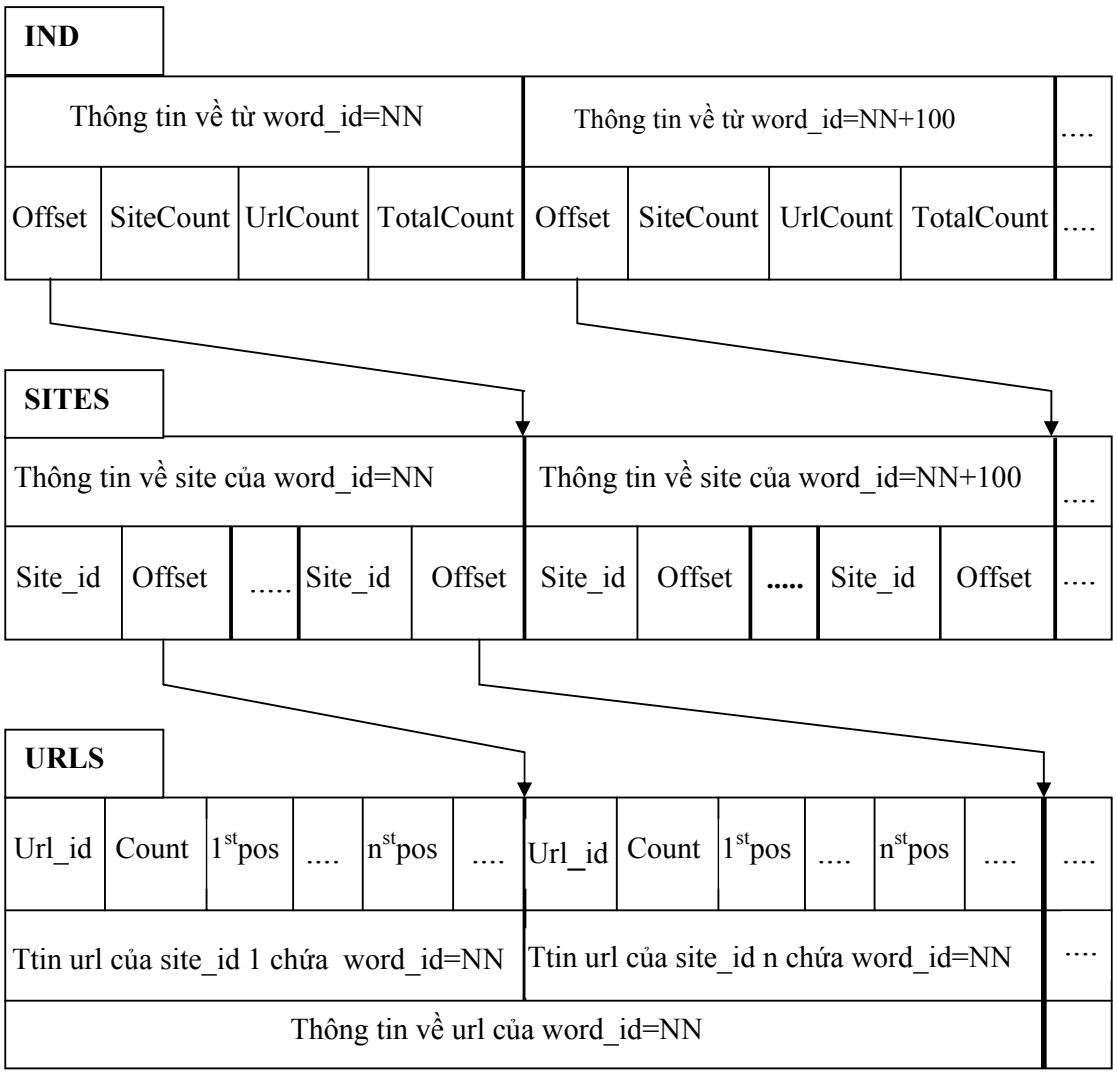
m_offset: vị trí kết thúc các thông tin về url trong site đó trong file urls.

iii. File urls: file này chứa nội dung các url thuộc về các site trong file sites nói trên và có chứa từ được nêu trong bảng ind. Cấu trúc file như sau:

Thông tin về mỗi từ (đã được mô tả trong file ind) trong các url, các url này thuộc vào các site được mô tả trong file sites ở trên		
Vị trí	**Độ rộng**	**Mô tả**
0	4	url_id của url đó
4	4	Count: số lần xuất hiện của từ đó trong url
6	2	Vị trí xuất hiện lần thứ nhất của từ trong url
.....		

N*2+4	2	Vị trí xuất hiện lần thứ N của từ trong url, N là tổng số lần xuất hiện của từ đó trong url
Lặp lại với các url khác có chứa từ đó trong cùng một site		
Lặp lại với các url khác có chứa từ đó thuộc các site khác		
Lặp lại với các từ khóa khác trong file ind.		

Ba file ind, sites, urls ở trên liên hệ và phụ thuộc chặt chẽ vào nhau. Khi cần lưu thêm thông tin về từ hoặc lấy ra thông tin ta cần phải truy cập vào cả 3 file cùng một lúc. Mối liên hệ giữa chúng được thể hiện trong sơ đồ sau:



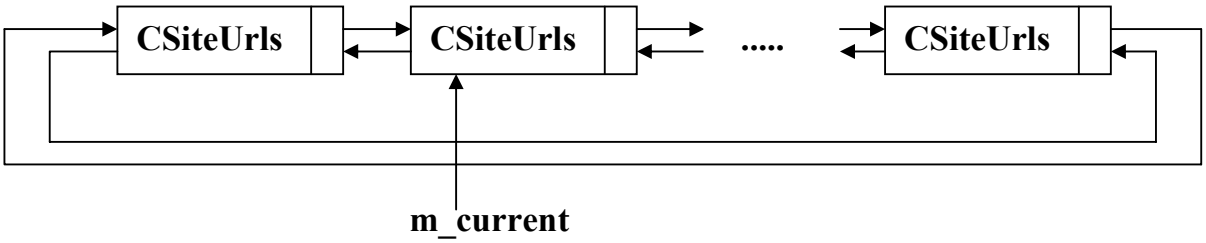
Hình 4.: *Sơ đồ liên hệ giữa các file nhị phân theo cơ chế CompactStorage.*

Chẳng hạn cần tìm kiếm các urls có chứa từ khóa word với mã là word_id, ta cần tìm trong thư mục word_id mod 100. Địa chỉ bắt đầu thông tin về từ word này trong file ind sẽ là (word_id div 100)*sizeof(WordInd) (do các từ trong file ind được viết kế tiếp nhau, mỗi từ được viết trong một WordInd và word_id của từ lưu tiếp sau bằng word_id của từ phía trước cộng 100). Từ đó ta sẽ xác định được giá trị của trường m_offset tức là địa chỉ offset bắt đầu và kết thúc lưu thông tin về các site trong file sites. Từ đó ta sẽ xác định được địa chỉ offset trên file urls bắt đầu thông tin về các url thuộc site này có chứa từ khóa đó. Cứ như vậy với các url thuộc các site khác.

4.3 Tìm hiểu về việc thực thi quá trình crawler trong module index của máy tìm kiếm VietSeek.

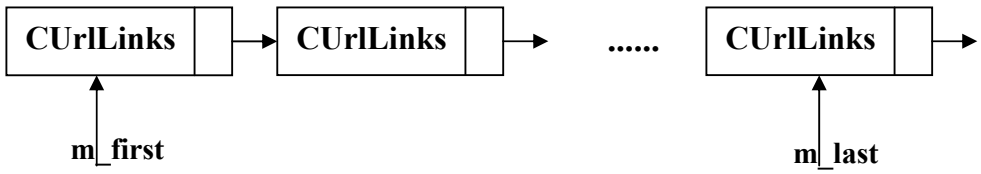
4.3.1 Quá trình crawler trong ASPseek và nhu cầu song song hóa

Aspseek sử dụng một cấu trúc dữ liệu bảng băm để làm hàng đợi lưu các url cần được index. Các URL trong hàng đợi được nhóm theo site, các url thuộc cùng một site được nhóm vào một danh sách FIFO gọi là CSiteUrls. Khi có một url thuộc vào một site cần đưa vào hàng đợi, url đó được thêm vào cuối danh sách url của site nó thuộc vào. Toàn bộ hàng đợi là một bảng băm các CsiteUrls và có một con trỏ trỏ tới site hiện tại đang được duyệt. Khi cần lấy ra một url để duyệt tiếp, url ở đỉnh danh sách của site hiện tại sẽ được trích ra. Cấu trúc của hàng đợi này như sau:

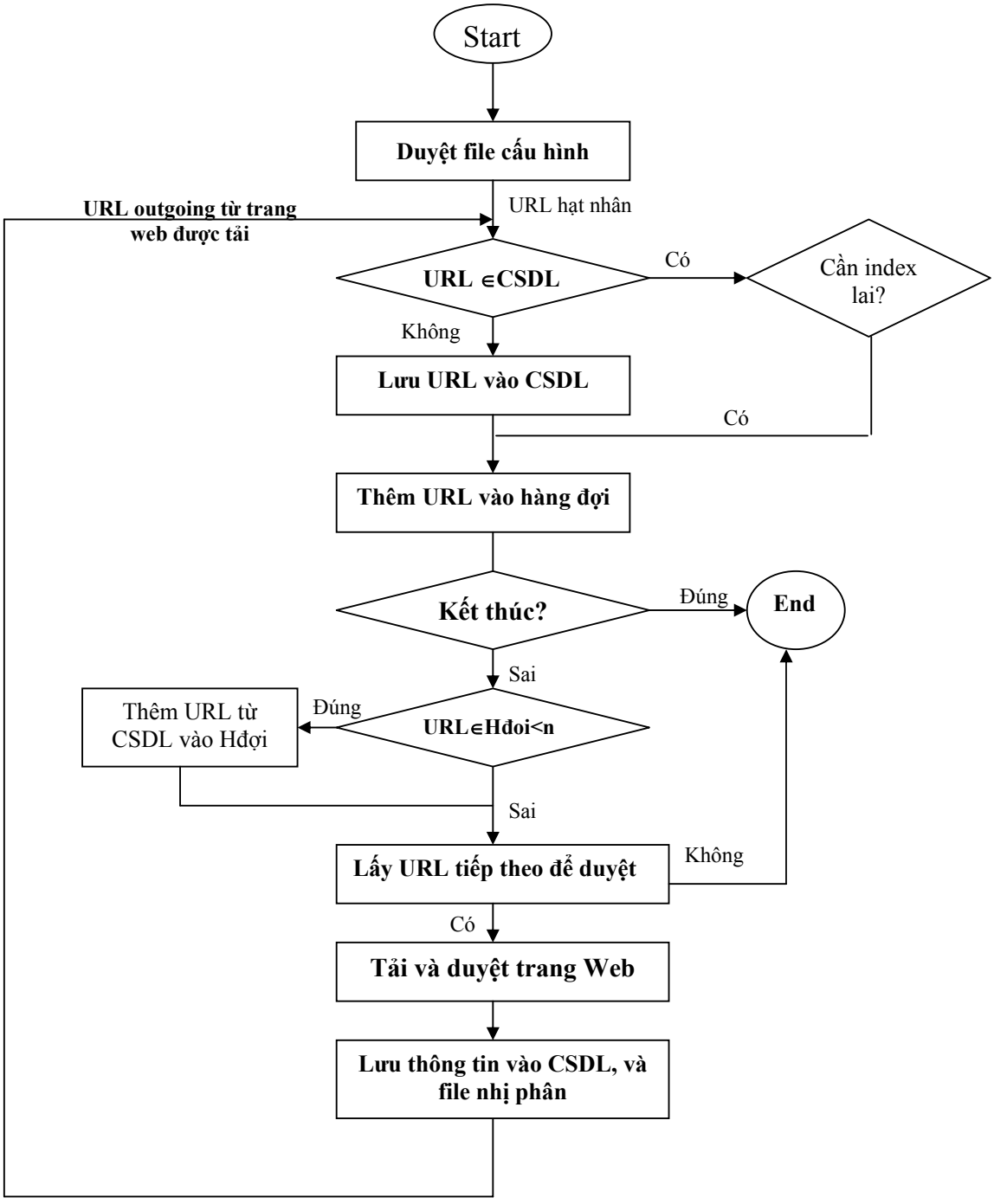


Hình 4.a:*Cấu trúc của hàng đợi frontierCSiteQueue trong ASPseek*

Trong đó: mỗi CsiteUrls là một danh sách một chiều các mảng chứa url thuộc về cùng một site. Và CurlLinks là một mảng gồm 100 url liên tiếp.



Hình 4.b:*Cấu trúc một phần tử CSiteUrls*



Hình 1: *Cơ chế chọn các trang web để tải và duyệt nội dung trong ASPseek*

- ASPseek thực hiện việc duyệt Web theo chiều rộng theo chiến lược mù, nó cố gắng tải nội dung tất cả url nó gặp trong quá trình duyệt mà không thực hiện việc lựa chọn hay tính điểm cho các URL cần duyệt.

- Đầu tiên chương trình sẽ tải file cấu hình aspseek.conf, lấy ra các URL bắt đầu (URL hạt nhân) được chỉ định trong lệnh Server. Sau đó module index sẽ kiểm tra xem các URL khởi đầu này đã tồn tại trong cơ sở dữ liệu hay chưa, nếu chưa thông tin về chúng sẽ được lưu vào các bảng site và urlword trong cơ sở dữ liệu MySQL đồng thời được thêm vào trong hàng đợi. Trong trường hợp URL đã có mặt trong cơ sở dữ liệu, chương trình sẽ kiểm tra xem nó có cần phải index lại hay không, nếu cần URL đó cũng được thêm vào trong hàng đợi.

- Trong quá trình duyệt nội dung một url, các địa chỉ url được trỏ tới từ url đó đầu tiên sẽ được thêm vào trong cơ sở dữ liệu (bảng urlword, sites) nếu nó chưa có trong đó và được thêm hàng đợi nếu nó thỏa mãn các điều kiện hạn chế do người dùng chỉ định (không quá sâu so với URL bắt đầu, không vượt khỏi server hiện tại, không nằm trong các url bị cấm...). Đồng thời trong quá trình duyệt, nội dung url sẽ được lưu vào trong cơ sở dữ liệu (bảng urlwordNN, wordurl) cũng như lưu vào các file nhị phân trung gian.

- Tại một thời điểm, nếu số lượng inactiveSite (số lượng các CsiteUrls có một hoặc nhiều hơn các url chưa được index) trong hàng đợi nhỏ hơn một số lượng xác định, aspseek sẽ truy vấn cơ sở dữ liệu và thêm vào trong hàng đợi các url đã tới thời hạn index lại (các url có giá trị trường next_index_time nhỏ hơn thời điểm hiện tại).

- Kết quả thu được của quá trình crawler trong ASPseek bao gồm:

- Thông tin về các site, url và từ khóa được lưu trong các bảng của cơ sở dữ liệu.

- Thông tin nội dung của các url sau khi đã loại bỏ các từ dừng (stopword) được lưu trong các file nhị phân có cấu trúc đặc biệt gọi là các file delta.

- Sau khi quá trình crawler kết thúc, nội dung các file delta sẽ được sắp xếp và trộn với nội dung các file nhị phân cũ trong cơ sở dữ liệu trong một quá trình gọi là index ngược. Qúa trình này sẽ chuyển cấu trúc các file nhị phân từ dạng định dạng theo url sang định dạng theo từ khóa để phục vụ cho nhu cầu tìm kiếm url theo từ khóa của người dùng. Sau khi trộn, nội dung của các file delta bị cắt bỏ.

Nhu cầu song song hóa

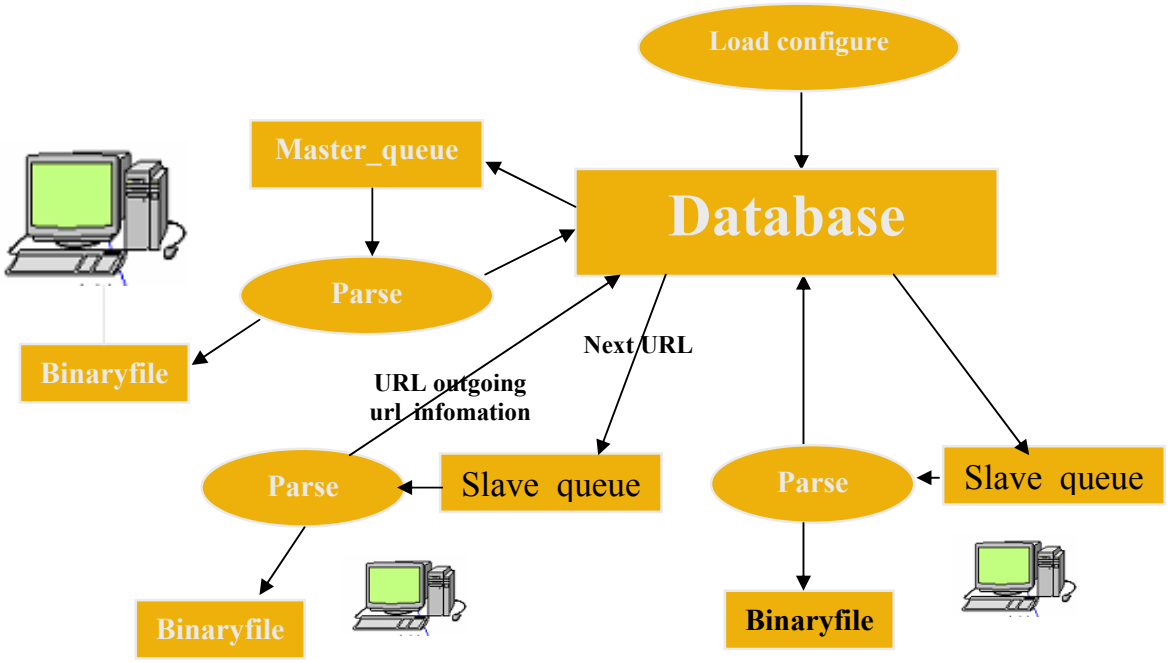
4.3.2 Đề xuất giải pháp song song hóa

Công việc tải và duyệt các trang web là công việc tốn rất nhiều thời gian, chiếm tới hơn 80% thời gian thực hiện của module index. Vì vậy việc song song hóa quá trình này sẽ có ý nghĩa rất lớn trong việc nâng cao chất lượng của bộ tìm kiếm. Ngoài việc giúp rút ngắn thời gian thực hiện index, việc song song hóa sẽ cho phép tăng thêm số lượng các trang web được đánh chỉ số, cũng như tăng mức độ cập nhật của các trang web đã có trong cơ sở dữ liệu.

4.3.2.1 Giải pháp song song hóa

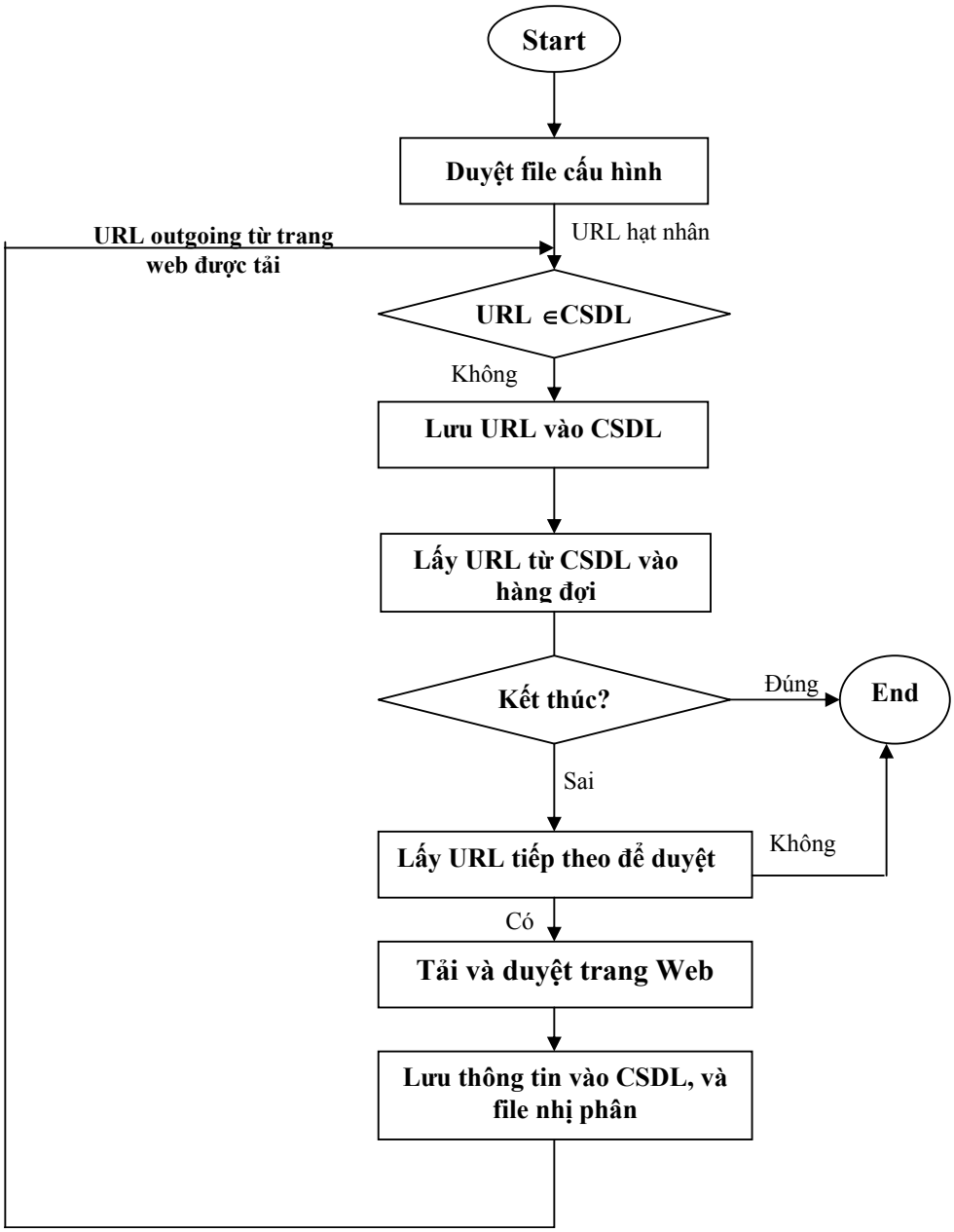
Giải pháp chúng tôi đưa ra ở đây là sử dụng cơ sở dữ liệu, các bảng trong MySQL và các file nhị phân, tập trung trên một bộ xử lý chính, và sử dụng cơ sở dữ liệu MySQL làm kênh giao tiếp trung gian chính giữa các bộ xử lý trong hệ thống. Ở đây hệ quản trị cơ sở dữ liệu sẽ đóng vai trò chính trong việc phân chia công việc cho các bộ xử lý. Quá trình tải và chọn các url để index tiếp theo trong cơ chế này như sau:

Trong quá trình duyệt file cấu hình, các địa chỉ url được đặt làm địa chỉ xuất phát của quá trình index sẽ chỉ được thêm vào các bảng trong cơ sở dữ liệu MySQL mà không được thêm vào trong hàng đợi của bất kỳ bộ xử lý nào. Tương tự như vậy, trong quá trình duyệt nội dung một trang Web, các địa chỉ url mà trang này liên kết tới cũng chỉ được thêm vào cơ sở dữ liệu mà không thêm vào trong hàng đợi của các bộ xử lý. Các bộ xử lý sẽ lấy ra các url để thêm vào hàng đợi bằng các truy vấn cơ sở dữ liệu. Như vậy quá các bảng trong cơ sở dữ liệu sẽ là nguồn cung cấp các địa chỉ url để index duy nhất cho các bộ xử lý.



Hình 3: *Giải pháp song song hóa quá trình crawler trong module index của ASPseek.*

Khi đó quá trình crawler trên mỗi bộ xử lý được minh họa như hình 4



Hình 4: *Quá trình crawler ở mỗi bộ xử lý sau khi đã tiến hành song song hóa.*

- Đầu tiên bộ xử lý chính sẽ tiến hành duyệt file cấu hình, lưu các url bắt đầu vào trong cơ sở dữ liệu MySQL nếu nó chưa có mặt trong đó. Tiếp theo các bộ xử lý sẽ thực hiện các vòng lặp crawler như sau:
- Tiến hành truy vấn cơ sở dữ liệu để lấy ra các url để thêm vào hàng đợi nếu số lượng url trong hàng đợi nhỏ hơn một số lượng xác định.
- Kiểm tra điều kiện kết thúc, nếu đúng dừng lại
- Lấy url tiếp theo từ hàng đợi để tải và duyệt, nếu không lấy thêm được các url mới, quá trình crawler cũng kết thúc.
- Duyệt nội dung các trang web ứng với các url, lưu thông tin của trang vào trong các bảng trong cơ sở dữ liệu cũng như các file nhị phân. Các url được trích ra từ trang web sẽ được lưu vào trong bảng urlword trong cơ sở dữ liệu (nếu chưa tồn tại).
- Lặp lại quá trình trên cho tới khi thỏa mãn điều kiện dừng.

Thi hành cụ thể

4.3.2.2 Cơ chế phân công công việc giữa các bộ xử lý.

Mỗi lần một bộ xử lý truy vấn cơ sở dữ liệu để lấy ra các URL tiếp theo để index, chương trình sẽ cân đối giữa số lượng các URL đang có mặt trong hàng đợi của bộ xử lý đó mà chưa được index với số lượng các URL trong cơ sở dữ liệu cần được đánh chỉ số cũng như số lượng bộ xử lý đang chạy song song để tính toán số lượng URL được lấy ra. Như vậy ta có thể phân phối tương đối đồng đều công việc cho các BXL làm việc.

Thi hành cụ thể

4.3.2.3 Tổng hợp kết quả sau quá trình song song:

Sau khi quá trình crawler trên các bộ xử lý kết thúc, bộ xử lý chính sẽ tiến hành kết hợp các kết quả crawler trên các máy sau đó tiếp tục tiến hành các công việc còn lại của quá trình index. Việc xử lý ở đây chỉ bao gồm xử lý các file nhị phân delta do các bộ xử lý tạo ra. Do trong quá trình thực hiện, mỗi bộ xử lý sẽ lưu kết quả vào một thư mục file nhị phân riêng biệt, ta cần kết hợp nội dung các file này thành một file chung duy nhất. Các file đã kết hợp này sẽ được sử dụng làm đầu vào cho quá trình index ngược tiếp theo.

4.3.2.4 Vấn đề tương tranh giữa các bộ xử lý:

Để giải quyết vấn đề tương tranh giữa các bộ xử lý để tránh duyệt lặp lại một URL nhiều lần, trong bảng urlword (đã được trình bày ở trên) ta thêm vào một trường index_status để lưu giữ trạng thái index của một url.

Giá trị	Ý nghĩa
0	URL tương ứng chưa được index
1	URL tương ứng đã được index, việc có index lại URL này hay không phụ thuộc vào giá trị của trường next_index_time.
2	URL đang được index, dù giá trị của trường next_index_time là gì cũng không được index URL này.

Giải pháp này đảm bảo việc các bộ xử lý sẽ không duyệt lặp lại các url mà các bộ xử lý khác đã xử lý trong cùng một lần index.

4.3.2.5 Đánh giá giải pháp song song hóa.

Ưu điểm:

Giải pháp tương đối đơn giản, khả năng tận dụng lại mã nguồn cao

- Các bộ xử lý có thể hoạt động tương đối độc lập, nhu cầu truyền thông giữa các bộ xử lý là rất ít, các bộ xử lý không cần sử dụng các kết quả do bộ xử lý khác tạo ra.

- Hệ thống dễ dàng được mở rộng thêm trên nhiều bộ xử lý.

Nhược điểm:

- Việc phân chia công việc phụ thuộc nhiều vào hệ quản trị cơ sở dữ liệu, do đó có thể xuất hiện sự phân phối công việc không được cân bằng.

- Giải pháp trên mới chỉ song song hóa được quá trình crawler trong module index, cần nghiên cứu các giải pháp song song hóa được nhiều công việc hơn nữa.

Tài liệu tham khảo:

[1] Đỗ thị Diệu Ngọc (2003). *Một số vấn đề về phân lớp cho ....* Luận văn đại học khoa Công Nghệ Đại học Quốc Gia Hà Nội 2003.

[2] G.A.Geist, J.A.Kolh, P.M.Papadopoulos, *PVM and MPI: a comparison of features.* Applied Mathematical Sciences subprogram of the Office of Energy Reaseach, US Department of Energy. May 30 1996.

[x] Gautam Pant, Padmini Srinivasan, Fillipo Menczer. *Crawling the Web.* The University of Iowa, Iowa City IA 52242, USA

[x] Jack Dongarra. *MPI: the complete Reference.* Cung cấp tại http://rsusu1.rnd.runnet.ru/parallel/mpi/mpibook/. 1995.

[x] Ian Foster. *Designing and Building Parallel Programs.* Cung cấp tại http://www-unix.mcs.anl.gov/dbpp/ 1995.

[x] Li wang, Edward A.Fox, *Crawling on the World Wide Web*.Virginia Tech 2001

[x] Kir Kolyshkin. *Aspseek Manual.* Cung cấp tại http://www.aspseek.org. 2002.

[x] Osmar R.Zaiane, *From Resource Discovery to Knowledge Discovery in the Internet.* School of Computing Science, Simon Fracer University, Burnaby , BC Canada V5A 1S6.

[x] Sergey Brin and Lawrence Page, *The Anatomy of large scale Hypertextual Web Search Engine.* Computer Science Department, Standford University, Standford CA 94305, USA.

[x] Shian - Hua Lin, Meng Chang Chen, Jan-Ming Ho, *ACIRD: Intelligent Internet Document Organization and Retrival.* IEEE transaction on knowledge and data engineering VOL 14, NO 3 May/June 2002.

[x] Yukiya Aoyama, Jun Nakano. *RS/6000 SP: Practical MPI Programming.* IBM International Technical Support Organization 1999.