共同体的P2P全文検索システムの開発

1　背景

近年、電子文書の普及にしたがって、それに対して検索を行うシステムへの要求は高まってきている。個々の文書につけた属性情報をもとに検索するだけでは不十分であり、文書の本文の全文を対象とした検索システム、すなわち全文検索システムが求められる場合は多い。しかしながら、全文検索にかかる処理は大きな計算量を伴うため、大規模な検索システムを構築するにあたっては、高価なハードウェアやソフトウェアと、専門的な知識が必要とされるという問題がある。

既に多くの全文検索システムの製品が市場に存在するが、そのどれもがそれぞれの長所と短所を備えている。一口に全文検索システムといっても、要求される機能や性能は利用者によって様々であり、全ての要求に応えることは不可能である。したがって、どのようなユースケースを想定して製品開発を行うかが重要である。本プロジェクトでは、Web 上の文書や企業のファイルサーバ上の文書のような、比較的大規模で、かつ頻繁に更新される文書群を対象としたユースケースを想定し、そこで要求される機能や性能に焦点を絞って開発を行った。

2　目的

本プロジェクトでは、全文検索システム「Hyper Estraier」を開発した。その目的は、安価なハードウェアでも動作し、検索システムに対する専門的な知識がなくても容易に導入できるシステムを実現することであった。そのため、Peer-to-Peer 方式の分散処理によって一般的なパーソナルコンピュータを組み合わせて利用できるようにするとともに、ライブラリや各種のユーティリティを拡充することで、簡単に導入できるだけでなく、ユーザの要求に応じた高度なカスタマイズができるように設計した。

インデックスを構築する処理のパフォーマンスとスケーラビリティを従来のシステムに比べて劇的に向上させることも目的のひとつである。テキスト解析には検索漏れがないという利点を持つ N-gram 法を採用しているが、N-gram 法による従来の実装だと、インデックスのサイズが肥大化し、結果として構築にかかる速度が低下してしまう問題があった。

実用上、全文検索システムを実用する上でのスケールの限界は構築速度にかけられる時間の制約から来るため、インデックスの構築速度を向上させることは重要な課題である。本プロジェクトではこの課題に取り組み、一般的なパソコン 1 台で検索対象が 100 万ファイルを越える規模のインデックスを現実的な時間で構築できるようにし、また P2P の並列処理を可能とすることでさらに大規模な検索システムを構築できるようにした。

3　開発内容

3.1　アーキテクチャ

Hyper Estraier は転置インデックスと呼ばれるデータ構造のデータベースを用いて高速な検索を実現する。以後、これを単にインデックスと呼ぶ。一般的に、インデックス型の検索システムは、インデックスを構築する機能とインデックスの検索を行う機能を主体とし、その周辺にユーザの要求に応じた機能を組み合わせて構成される。Hyper Estraier もその構成を踏襲している。検索対象の文書を収集してインデックスに登録する機能を「ギャザラ（gatherer）」と呼ぶ。インデックスに対して検索を行ってその結果をユーザに提示する機能を「サーチャ（searcher）」と呼ぶ。Hyper Estraier は、ギャザラやサーチャを簡単に実装するためのライブラリと、実際にそれを実装したアプリケーションを組み合わせた製品である。

3.2 階層 API

Hyper Estraier のライブラリは複数の層からなる API を備える。プログラミング言語は C 言語である。カスタマイズを行うプログラマは、その目的や能力に応じた層の API を用いる。API の階層としては、コア API、スレッド API、ノード API および言語バインディングの 4 層がある。コア API を利用すると性能を追求したアプリケーションが開発できるが、同期制御に関してアプリケーションプログラマが責任を負わねばならない。スレッド API はスレッド間の同期制御をライブラリ内部に隠蔽したものである。

コア API もスレッド API も、アプリケーションが不具合などによって不慮の停止をするとインデックスに不整合が起きる可能性がある。これを防ぐためには、インデックスを管理するプロセスとアプリケーションのプロセスを分離する必要がある。そのため、Hyper Estraier では、インデックスの管理を行うノードサーバと呼ばれる機能を提供する。ノード API は、ノードサーバとの通信プロトコルを隠蔽するための API である。

言語バインディングとしては、Java 用と Ruby 用が提供されている。C 言語による開発はメモリ管理等の負担が大きいために敬遠されることが多いが、Java や Ruby といった言語を用いることによって市場価値を高めることが狙いである。なお、有志の手によってコア API の Python・Perl・Ruby 用のバインディングもリリースされている。

3.3 P2P 機能

複数のノードサーバが相互に通信することよって、P2P 方式の分散処理を実現している。サーバ同士は対等の関係であるため、クライアントはどのサーバにアクセスしてもサービスを享受することができる。どれかのサーバがダウンしてもシステム全体が停止することはない。

多数のインデックスを管理することを想定した場合、インデックス毎にサーバを起動するのは非効率である。そこで、単一のプロセスおよび単一のポートで複数のインデックスを管轄できるノードマスタというプログラムが提供される。ノードマスタ内の個々のインデックスは独立したサービスを提供するので、ノードマスタは複数のサーバの集合体とみなすこともできる。個々のインデックスを管轄する仮想的なサーバをノードサーバと呼ぶ。各ノードサーバには別々の URL が割り当てられる。ユーザが操作するアプリケーションはノードサーバのクライアントとして位置づけられるが、接続先のノードサーバについては URL だけ知っていればよく、ノードサーバがどのノードマスタ上に存在しているかをアプリケーション側で意識する必要はない。

ノードサーバは別のノードサーバに対して一方的にリンクを張ることができる。クライアントがノードサーバに検索要求を出した場合、依頼を受けたノードサーバは自分がリンクを張っているノードサーバにもクエリを中継し、返された結果を自分のインデックスの結果とマージしてクライアントに返す。すなわち、いわゆるメタ検索機能を全てのノードサーバが持つことによって、P2P 型の分散処理を実現している。

メタ検索は多階層的に行われる。経路の循環は自動的に検出されて抑止されるので、検索時には木構造のネットワークからデータを収集しているような挙動になる。この機構によって、検索対象となるノードの数を際限なく増やしていくことが可能になる。

ノード間の各リンクには、信頼度という数値が定義される。ノードがメタ検索の結果をマージする際には、スコアの重みづけに信頼度が利用される。信頼度が高いノードの結果は上位に来やすいということである。リンク作成や信頼度設定の指示はアプリケーションに任されるが、よく利用されるノードの信頼度を高めるように指示することによって、検索の精度が高めていくことができる。

4　従来の技術との相違

4.1　インターフェイスの簡潔性と多様性

アプリケーション（ユーザインターフェイス）とライブラリ（API）の双方を提供することにより、プログラマでなくても簡単に全文検索システムを構築できるようにするとともに、システムインテグレーションのビジネスで必要とされる高度なカスタマイズ性を確保している。

汎用の全文検索システムを考えると、文書の格納先、文書のサイズや数、文書のフォーマット、文書に付随する属性（メタデータ）、テキストの自然言語などに関するユーザの要求は多様なものである。実用的なシステムを作ろうとするとユーザ毎にインテグレーションや個別開発が必要となるが、その自由度や難易度は利用するツールやライブラリによって大きく異なる。Hyper Estraier は特にビジネスシーンでの適用を考えて、レガシシステムとの連携も容易に行えるようにインターフェイスが整備されている。また、言語バインディングを拡充させることによって、アプリケーションを新規開発する際のコストをできるだけ抑えられるようにしている。非常に簡潔な表現で全文検索システムのアプリケーションを記述できるため、法人ユーザだけでなく個人ユーザでも独自のシステムを手軽に構築できる。この特徴によって、既存の製品では扱えなかったニッチ市場への普及が期待できる。

4.2　N-gram 法の効率的な実装

Google のような大規模な検索システムでは分かち書きと呼ばれる方式でテキストを解析している。これは、テキストに対して語彙の辞書や自然言語の文法を用いた複雑な計算処理を行って、人間が識別できる単語もしくは形態素を抽出する手法である。この方式では、辞書にない語彙を含んだり、くだけた表現がなされていたり、古文であったりする場合に検索がうまくできないという問題がある。このような検索漏れは、法人や個人で特定の文書群を対象とした検索システムでは大きな問題となることがある。また、特定の言語に強く依存した処理を行うために、多言語化が難しいという問題もある。

一方で、N-gram 法は単純に文字単位でテキストを区切るために、いかなる言語のいかなる語彙を持つテキストでも、漏れの無い検索ができる。しかし、検索キー同士の連接判定を行うために、どのキーが何番目に出現したかという位置情報をインデックスに持つ必要があり、インデックスのサイズが非常に大きくなるという問題がある。インデックスのサイズの増大はインデックスの構築速度の低下と検索速度の低下を招き、実用上のスケーラビリティを低下させる。

Hyper Estraier では、位置情報を持たなくても N-gram のキーの連接が判定できる仕組みを考案し、実装した。間単に言えば、各キーに後続のキーのハッシュ値を持たせてインデックスに格納する方式である。通常の位置情報を使う方法だと、インデックスのサイズが対象文書のサイズの数倍にならざるを得ないが、Hyper Estraier では、インデックスのサイズを対象文書のサイズの半分以下に抑えることができる。

これによって、普通のパソコンでも 100 万件（10GB）以上の文書を対象とした検索システムが実用できるようになった。N-gram 法の実装でこのようなスケーラビリティを有するシステムは他に類を見ない。

4.3　P2P 型の全文検索システム

大規模な全文検索システムを実現するためにクラスタ構成を取ることが考えられるが、クラスタの管理システムや複雑なノウハウが必要となり、運用コストが大きくなるという問題がある。より安価に分散処理を実現するために、Hyper Estraier では P2P 型のアーキテクチャを採用した。また、インターネット環境で分散処理を行うことを考えると、不特定多数がクラスタ構成（クライアントサーバ）で連携することは難しい。Hyper Estraier のノード同士の連携は個々のノードの管理者が一方的に宣言したり拒絶したりできるため、特定の管理者がいなくてもネットワークを広げていくことができる。この構成はまさにインターネット環境に適したものであると言える。

Hyper Estraier の P2P 機構は単にスケーラビリティを増大させるだけではなく、検索精度の向上にも寄与する。クラスタ型の検索システムの場合、特定のクエリにたいする検索結果はネットワーク全体として生成されるが、Hyper Estraier の場合はユーザがどのノードに接続したかによって結果が変わる。したがって、各ユーザによって有用なノードにリンクすることは有利であるため、ソーシャルネットワークシステムのように、関連のあるノード同士が連携していくことで、ネットワーク全体としての利便性を高めていくことが期待できる。また、ノード間の信頼度をスコアの重みづけに利用することでその効果を活用することができる。

5　期待される効果

本プロジェクトの活動の結果として、高性能・高機能な全文検索システムを容易に構築するためのソフトウェアの開発は成功したと言える。各種のアプリケーションやサービスに容易に組み込むことができるようになるため、全文検索機能があれば便利であろう製品には順次導入されていくことが期待できる。

Hyper Estraier はオープンソースソフトウェアとして一般に公開している。各種の製品開発やシステムインテグレーションの機会に広く利用してもらうことによって、産業の発展に貢献するとともに、ユーザからのフィードバックを成果物に反映して機能や品質を高めていくことができる。本プロジェクトの活動は期間終了後も継続させ、より利用価値の高い製品を仕上げていく予定である。

6　普及の見通し

既に、Hyper Estraier は Web サイトの検索機能として広く使われてきている。それだけでなく、電子メールのアプリケーションに組み込んだり、いわゆるデスクトップ検索のアプリケーションにも組み込まれてきている。現状で報告を受けている適用例はオープンソースのものが多いが、パッケージ製品に組み込んだり、ASP のサービスで動作させるなどして、企業活動に利用することも可能である。

全文検索システムとしての中心的な機能は未踏プロジェクトの期間中に完成したが、今後はアプリケーションの開発に注力する予定である。具体的には、Web を対象としたいわゆるサーチエンジンの開発と、Windows 用のデスクトップ検索システムの開発である。それらの活動を通して普及を図るとともに、性能と機能の強化を継続していきたい。

7　開発者名

平林幹雄 （mikio@users.sourceforge.net）