3T-08

協調サーチエンジンにおけるキャッシュサーバの改善と評価

西田 喜裕 † 山本 崇 † 佐藤永欣 † 上原 稔 † 森 秀樹 † 酒井 義文 †

† 東洋大学工学部情報工学科

1 はじめに

我々はWeb検索のためのインデックス更新時間を短縮化できる分散型のサーチエンジンである協調サーチエンジン (Cooperative Search Engine、以下CSE) を開発している。CSEは複数の局所的なサーチエンジン (Local Meta Search Engine、以下LMSE) を構築し、これらを協調動作させ、検索やインデックスファイルの更新作業を行なう。これらの協調作業にはLocation Server(以下LS) という、LMSEの管理を行なうサーバが必要であるが、CSEではこれが単一であるために検索ユーザやLMSEの増加への対応が困難であった。また、以前のCSEでは検索結果をCGIで操作していたために、検索結果を効率的に扱えないという問題があった。そこでCSEにキャッシュサーバ (以下CS) という、LSの応答や検索結果を管理するサーバを導入した。

本稿では、このCSの検索時の動作や、問題点、及び問題点の改善について述べる。

2 関連研究

本研究は分散型のサーチエンジンであるCSEの検索結果のキャッシュについてであるが、『On Caching Search Engine Query Results』[1]で、サーチエンジンであるexcite[2]の検索のログを調査した結果、参照の局所性がみられ、多くのクエリーが、同じ、あるいは異なったユーザによって二度以上使用されていることなどから検索結果のキャッシュは有効であると述べられている。さらに、検索サービスサイトaltavista[3]の調査でも同様の結果がみられたと述べている。しかしこれは規模が大きい程有効であるとも述べられているので、中小規模を対象としたCSEとは少し異なる結果である。

また、ポピュラーなWebのキャッシュサーバであるsquid[4]は、ICP(Internet Cache Protocol)[5][6]によってキャッシュの分散、階層化を実現している。これは現在CSEの中で単一であるCSの将来の分散、階層化の参考になるだろう。

Performance Improvement and Evaluation of Cache Server in Cooperative Search Engine
Yoshihiro Nishida
Department of Information and Computer Sciences, Faculty of Engineering,Toyo University

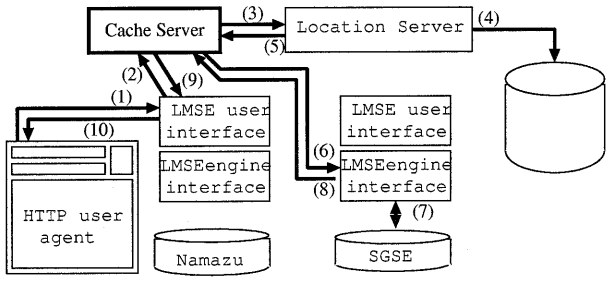

図 1: CSEの検索時の動作

3 CSEの検索時の動作

3.1 CSEの検索時の動作

CSEの検索時の動作を説明する。まず、キャッシュにヒットしなかった場合は図.1のようになる。

1. (1)(2)LMSEのユーザインターフェース部はユーザからの検索要求を受け取ると、CSに検索式を送信する。
2. (3)(4)(5) 与えられた検索式がキャッシュに無かった場合、つまり、その検索式が最近問合わされたものでなければ、CSは、与えられた検索式に適切なLMSEのURL(集合) をLSに問い合わせる。
3. (6)(7)(8) 次に、そのURLであらわされるLMSEに検索を行わせ、結果を受け取る。
4. (9)(10) そして、元のLMSEに検索結果を送信し、そのLMSEは結果を整形してユーザに送信する。同時に次のページを作成するために必要な数の検索結果をLMSEに要求し、LSから受け取ったLMSEのURL、それぞれのLMSEから読出した検索結果の数、検索結果をキャッシュする。

次に、与えられた検索式がキャッシュにヒットした場合は、図.2のようになる。

1. (1)(2) ユーザから検索式を与えられたLMSEのユーザインターフェース部は、キャッシュサーバに問い合わせる。
2. (3)(4) その検索式に対応するキャッシュファイルから検索結果を得て、指示された範囲の検索結果を返す。

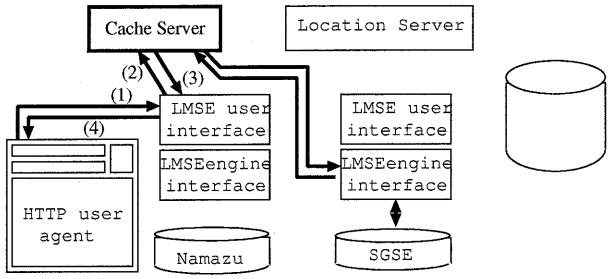

図 2: キャッシュにヒットした場合の検索動作

表 1: 要求された検索結果ページの数

1	2	3	4	5 ページ目以降
1954	119	45	36	28

もし要求されたページの次のページに必要な分の検索結果がキャッシュに無ければ、足りない分を継続検索に備えて既にあるキャッシュから LMSE の URL と読み出した検索結果の数と、ユーザに指定された検索結果数をもとに LMSE に問い合わせを行ない、検索結果をキャッシュしておく。

4 継続検索問題

表.1 は、本学で運用している squid のログからサーチエンジン (excite、fresheye、goo、google、infoseek、lycos) への検索リクエストを取り出し、ユーザが検索結果の結果の何ページ目までを参照したかをあらわすものである。ログに日時が記録されていないために期間は不明であるが、このログの総 HTTP リクエスト数は 1,986,040 件、その内、サーチエンジンへの総検索リクエスト数は 2713 件であった。表.1 より、検索結果の 1 ページ目で満足したユーザは、およそ 72%であった。以前の CS は、ユーザの要求した検索結果数にかかわらず全ての検索結果をキャッシュとして保存していた。これは継続検索を容易に行なえたが、表.1 に示されているように、後の検索結果ははほとんど使われず無駄が多い上に、CSE の特徴である短期間でのインデックスの更新がなされた場合、それが反映されないということもあった。そこで、CS のキャッシュ方法を検索結果を全件保存するのではなく、要求されたページよりも 1 ページ先までキャッシュする方式に改めた。これにより、最新の検索結果を反映しながらより有効にキャッシュすることができるようになった。表.2 は異なる 100 のクエリーを用いて CS にキャッシュを行なわせた結果である。今回の実験環境ではそれ程多くの検索結果が

表 2: キャッシュされたクエリーの数

	100	500(kbyte)
旧 CS	26	62
現在の CS	33	100

表 3: 旧 CS と現 CS の検索時間の比較

検索式	旧 CS 方式	現 CS 方式
mail(1 ページ目)	1.45 秒	0.80 秒
link(1 ページ目)	1.17 秒	0.66 秒
mail(2 ページ目)	0.37 秒	0.31 秒
link(2 ページ目)	0.28 秒	0.32 秒

得られなかったためにキャッシュされたクエリー数の差は大きくは無いが、より大規模になればキャッシュヒット率の向上がみられると思われる。表.3 は以前の CS と新しいキャッシュ方針を導入した CS の検索にかかった時間を示している。与えられたクエリーがキャッシュに無い検索結果の 1 ページ目を比較すると、以前の CS は大きなキャッシュを準備するために現在の CS よりも時間がかかっていた。2 ページ目以降についてはそれぞれキャッシュがあるためにほとんど時間はかわらない結果となった。

5 まとめ

以前は継続検索を矛盾無く行なうためと、他のユーザにも使用されることを考慮して全ての検索結果を保存していたが、要求されたページの次のページまでのキャッシュを作成しておくという方式に改めた結果、1 ページ目の結果を返す時間を短縮し、さらに限られたキャッシュを有効に使えるようになった。また継続検索の時に、CSE の特長である短期間のインデックス更新が行なわれると検索結果の順位等に影響が出てしまうが、最新の検索結果を反映できるようになった。

References

[1] Evangelos P. Markatos 『*On Caching Search Engine Query Results*』 ,http://archvlsi.ics.forth.gr/html_papers/TR241/

[2] 『*excite*』 ,http://www.excite.com

[3] 『*altavista*』 ,http://www.altavista.com

[4] 『*squid*』 ,http://www.squid-cache.org

[5] D. Wessels、K. Claffyp 『*Internet Cache Protocol (ICP), version 2*』 ,RFC2186

[6] D. Wessels、K. Claffy 『*Application of Internet Cache Protocol (ICP), version 2*』 ,RFC2187)