2000년도 대한전기학회 하계학술대회 논문집 2000.7.17-20

웹 페이지 저장공간 및 전송시간 축소를 위한 시스템 설계

정옥란, 김혜연, 이은영, 조동섭
이화여자대학교 과학기술대학원 컴퓨터학과

An algorithm for reduction of WEB Page Access Complecity

Ok-Ran chung, Hye-Yeon Kim, Eun-Young Lee, Dong-Sub Cho
Dept. of Computer Science and Engineering Ewha Womans University

Abstract - 전자상거래의 활성화는 HTML 문서나 Javascript와 같은 웹 문서의 빈번한 전송을 요구할 것이며 이는 향후 인터넷 전송 트래픽을 야기하는 주요 요인이 될 전망이다. 웹 페이지는 비슷한 문장열이 인수에 해당하는 부분만이 변화되면서 반복하는 특징을 갖고 있다. 본 연구에서는 웹 페이지의 이러한 특징을 이용하여 매크로 기법을 사용한 웹 문서 압축 알고리즘을 제안한다. 우리는 실험을 통해 본 알고리즘이 웹 페이지의 저장공간 압축에 좋은 성능을 가짐을 보여줌으로써 전송시간의 축소의 부가적인 효과를 거둘 수 있었다.

1. 서 론

현재 인터넷 상에서 전송 트래픽을 야기하는 주된 요인은 사운드나 그래픽, 영상 파일과 같은 멀티미디어 파일의 전송이었다. 따라서, 현재의 압축 기술은 이러한 멀티미디어 파일의 압축에 중점을 두어 왔다. 그러나, 전자 상거래의 활성화는 HTML, XML 문서나 Javascript와 같은 웹 문서에 대한 전송이 빈번해질 전망이다. 따라서, 향후 방대한 웹 문서의 전송은 인터넷 전송 트래픽을 야기하는 주요 요인이 될 것으로 예상된다.

이전의 발표한 압축내용으로는 저장된 또는 통신 된 데이터의 여분을 줄이는 보편적인 방법[1],한 파일을 다른 것으로 encoding하는 방법으로 데이터를 압축하는 "delta algorithm"[2], 웹 페이지 압축과 관련이 있는 delta 인코딩과 데이터 압축을 위한 HTTP 프로토콜의 명확한 확장[3] 등이 발표되었다. 이처럼 다각도에서 접근이 시도되었으나, 전송 시 유용한 바이트 압축에 대한 시도는 없었다. 또한 웹 페이지는 비슷한 문자열이 특정 부분만 바뀌면서 반복하는 특징을 가지고 있다. 표 1은 이러한 웹 페이지의 특징을 잘 보여 주고 있다.
즉, HTML 태그나 Javasript의 코드 부분은 반복하면서 인수에 해당하는 값만 변화하는 모습을 보여준다.

```
<table border="0" cellpadding="0" cellspacing="0" width="573">
    <tr>
        <td width="30"><p><img src="src/b.jpg" border="0"></td>
        <td width="42" height="29" cellpadding="0" cellspacing="0
            onmouseover="img_act('a1')" onmouseout="img_inac
            border="0" name="a1"></a></td>
        <td width="44" height="29" cellpadding="0" cellspacing="0
            onmouseover="img_act('a2')" onmouseout="img_inac
            border="0" name="a2"></a></td>
        <td width="120" height="29" cellpadding="0" cellspacing=
```

표1. 웹 페이지 소스

본 논문은 이러한 웹 페이지의 특징을 이용하여 매크로 기법을 사용한 압축 알고리즘을 제안한다. 따라서, 웹 문서의 압축은 웹 문서 내에서 반복하는 비슷한 부분을 찾아 이를 매크로로 정의하고 웹 문서의 내용을 매크로로 치환함으로써 이루어진다. 그리고, 웹 문서의 복원은 치환된 매크로를 매크로 확장함으로써 수행된다. 이러한 압축 기법은 웹 문서의 특징을 잘 반영하고 있기 때문에 높은 압축률을 기대할 수 있다. 기존의 연구에서는 웹 페이지 자체 특성을 이용하여 구체적으로는 HTML 자체 특성을 이용한 압축방법은 나와있지 않았다[4][5]. 이러한 시도는 본 연구가 처음이며, 또한 성능평가 면에서도 긍정적인 결과를 보여주었다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 연구에 대해 설명하고, 3장에서는 본 논문에서 제시한 압축 알고리즘을 제시한다. 4장에서는 다양한 웹 문서를 가지고 압축을 수행함으로써 본 알고리즘이 웹 문서의 압축에 좋은 성능을 가짐을 실험적으로 보인다. 마지막으로, 5장에서는 결론을 맺는다.

2. 기존 연구

잘 알려진 Huffman coding은 파일에 있는 각 symbol이나 문자들의 발생 빈도와 관련되는 방식으로 data 압축을 한다. 즉 자주 발생하는 것들은 짧은 길이의 bit를 차지하고, 드물게 발생하는 것들을 긴 길이의 bit를 차지하도록 한다. 실용적인 압축 시스템을 설계할 때에는, ① 압축될 하나의 파일 ② 같은 종류의 파일의 모임 ③ 파일의 종류를 고려하지 않는 전형적인 파일들 중 어느 것의 빈도 수를 기반 하여 coding 할 것인지 결정한다.
첫 번째 방식은 decoding에 사용되기 위해 coding table이 압축된 파일과 함께 저장되어야 한다.
반면에 두 번째 방식은 여러 coding table을 사용하고 세 번째 방식은 하나의 coding table을 사용하는데 이 방식들은 광범위하고 파일 당 오버헤드를 부과하지 않는다. 그러나 첫 번째 방식의 더 정확한 통계는 두 번째 세 번째 방식보다 코딩 효율을 향상시켜준다.[1]
다음으로 Delta algorithm은 두 파일이나 string 간의 차이를 계산하는 알고리즘이다. 즉 차이(difference)는 두 스트링에 포함되어 있는 가장 긴 연속된 문자인 Longest Common Subsequence(LCS)을 찾아, 두 파일 크기의 평균에서 이 LCS들를 빼는 방식으로 계산된다.
이 알고리즘은 데이터 object의 다양한 버전이 저장되고, 전송되고 처리될 때 사용된다. 이러한 압축은 공간을 절약해줄 수 있을 뿐만 아니라 I/O를 줄여 프로그램의 시간을 단축해준다. 즉 디스크에서 delta를 읽는 것은

delta를 사용하여 원하는 파일을 만드는 시간을 고려하더라도 전체 파일을 읽는 것보다 빠르다. 또한 새로운 버전을 만들기 위해서는 이전 버전을 가지고 있어야 하기 때문에 효율적인 암호화 형태로 사용될 수도 있다.

delta를 만들어 내기 위한 전통적인 프로그램은 Unix의 diff 이다. 그러나 이것은 text 파일들에만 적용되기 때문에 binary 파일을 처리하기 위해서는 binary code를 text 로 매핑시키는 과정을 거친 후 diff를 적용해야 한다. 그러나 최근에 나온 bdiff 나 vdelta 는 이러한 문제를 해결했다. [2]

또한, delta encoding and data compression for HTTP에서는 존재하는 구현 및 명세와 호환이 가능하고 암호화 기술을 효과적으로 다양하게 사용할 수 있도록 delta 인코딩과 데이터 압축을 위한 HTTP 프로토콜의 확장을 제안하였다.

HTTP/1.1은 조건적인 retrieval을 위한 응답을 위해서 all-or-none 모델을 유지하는데 이것은 resource가 조금이라도 변하면 현재 값 전체를 전송하고, 변하지 않았다면 전혀 변화하지 않았다는 것을 클라이언트한테 알려주는 방식이다[6]. 그러나 일반적으로 Web 자원이 변할 때 새로운 instance가 예전 것과 대체로 비슷하다는 것을 고안하여 모든 새로운 instance를 보내는 것 대신 두 instance 사이의 차이(혹은 delta)만 클라이언트에게 보내는 방식을 사용한다. 이것은 예전 instance의 복사 본을 저장하고 있는 클라이언트가 delta를 사용하여 새로운 instance를 생성하는 방식이다[7][8].

delta encoding을 사용한 HTTP는 response-size와 response-delay에서 주목할만한 향상을 제공한다. 제한된 대역폭 내에서의 response 크기와 지연의 감소는 성능향상에 중요한 영향을 미칠 수 있다.

3. 웹 페이지의 압축 및 복원 알고리즘

그림 1은 웹 문서의 압축과 복원 과정을 보여준다. 웹 서버는 일반 웹 문서를 압축하여 이를 데이터 베이스에 저장한 후 클라이언트의 요청이 있다. 클라이언트는 압축된 웹 문서를 복원한 후 이를 웹 브라우저에 전달하여 사용자가 해당 웹 문서를 볼 수 있도록 한다.

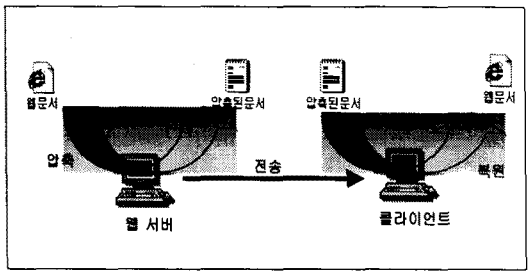

그림 1 웹 페이지의 압축과 복원

3.1 웹 문서의 압축

웹 문서에 적용할 매크로를 찾기 위해서는 먼저 비슷한 패턴을 가진 문자열들의 집합을 구할 필요가 있다. 이는 웹 문서를 정렬하고 각 패턴마다 적절한 매크로를 생성할 수 있는지 조사함으로써 해결할 수 있다.

```
<td width="250" align="center" valign="bottom" rowspan="6"
<td width="30" bgcolor="white"><p><img src="src/b.jpg" bor
<td width="30" bgcolor="white"><p><img src="src/b.jpg" bor
<td width="30" bgcolor="white"><p><img src="src/b.jpg" bor
<td width="30" bgcolor="white"><p><img src="src/b.jpg" bor
<td width="30" bgcolor="white"><p><img src="src/b.jpg" bor
<td width="30"></td>
<td width="30"><p><img src="src/b.jpg" border="0"></td>
<td width="30"><p><img src="src/b.jpg" border="0"></td>
<td width="360"><p><img src="src/b.jpg" border="0"></td>
<td width="376" height="76"><p><font face="Arial" size="1">
```

표 2 정렬된 웹 페이지 소스

표 2 는 정렬된 웹 문서의 일부를 보여준다.

매크로는 두 패턴에서 공통되는 부분을 찾음으로써 생성될 수 있다. 이 때, 두 문자열의 공통되는 부분이 원래 문자열 길이의 2/3 이상일 때에만 생성되도록 하였다. 길이가 짧은 매크로는 압축 효과를 거의 나타내지 않기 때문이다. 이렇게 생성된 매크로는 보다 많은 문자열에 적용될 수 있도록 조정된다. 일례로, 표2에서 패턴 1과 패턴 2에 의해 매크로가 생성되면, 이러한 매크로 다음과 같이 조정 과정을 거친다.

String adjust_macro(int b, String old_m) 매크로를 조정한다.

예1) imgOff = eval(imgName + "on.src");
imgOff = eval(imgName + "on.src");
imgOn = eval(imgName + "off.src");
imgOn = eval(imgName + "off.src")
예2) target="new"><img src="src/mov.jpg">
target="new"><img src="src/search.jpg">
target="_top"><img src="src/map.jpg">
최종적으로 생성되는 매크로는 다음과 같다

변경 후 ▼
String adjust_macro(int b, String old_m) 매크로를 조정한다.
예로 다음과 같은 패턴이 입력된다고 가정하자
(p1) target="new"><img src="src/mov.jpg">
(p2) target="new"><img src="src/search.jpg">
(p3) target="_top"><img src="src/map.jpg">

패턴 (p1)과 패턴 (p2)에 의해 맨 처음 아래와 같은 매크로가 생성한다.
(m1) target="new"><img src="src/~.jpg">
adjust_macro 함수는 새로운 패턴 (p3)에 의해 매크로 (m1)을 다음과 같이 조정한다.
(m2) target="~"><img src="src/~.jpg">
이렇게 생성된 매크로는 웹 문서를 압축하는 데 이용된다. 일례로, 위 매크로를 표 2에 적용한 결과는 표3과 같다. 압축 웹 문서에는 위 매크로와 표3의 데이터가 저장된다.

```
-1:<td width="30"></td>
8:`in/in`1`1`in
16:`b1`/td>
-1:<td width="250"align="center"valign="bottom"rowspan="6">
  <img src="src/ani.gif"
-1:border="0"><br>
  <a href="http://www.ewha.ac.kr/search97/samples/forms/s_de
-1:target="new"><img src="src/search.jpg"border="0"></a><a h
-1:target="_top"><img src="src/map.jpg"border="0"></a>
  <a href="http://www.ewha.ac.kr/~w3master/new/vivo/parent.h
-1:target="new"><img src="src/mov.jpg"border="0"></a>
  <a href="ewhaeng/index.html"><img src="src/eng.gif"
-1:width="50"height="46"border="0"></a></td>
```

표 3. 매크로를 적용한 웹 페이지 소스

다음 *Algorithm1* 은 웹 문서의 압축 과정을 보여준다.

```
Algorithm1 compress(FILE webDocument)
begin
   sort webDocumenet;
   foreach line1 in webDocument
   begin
      set macro to line1;
      do begin
         read line2 from webDocument;
         generate m from macro and line2;
         if length of m < 2/3*max   (lengh of macro,
   then                             length of line2)
            set line1 to line2;
            if macro is generated then
               store macro;
            break;
         endif;
         set macro to m;
      end do;
   end foreach;
   foreach line1 in webDocument
   begin
      apply a macro to line1 and store it;
   end foreach;
end.
```

3.2 압축 웹 문서의 복원

압축된 웹 문서의 복원은 각 라인에 해당 매크로를 적용하는 것이다. 일례로 표3의 패턴 1을 매크로를 적용하면 원래의 웹 문서 내용을 복원할 수 있다.

4. 실험 및 평가

본 알고리즘은 Java 언어를 사용하여 PC 상에서 구현되었다. 실험을 위한 웹 문서는 Javascript를 포함한 HTML 문서로 하였다.

	문서명	원래크기	압축후크기	압축률(%)
1	Archi	2kb	1kb	50%
2	Computer	2kb	1kb	50%
3	doobob	3kb	2kb	34%
4	Kiee	2kb	1kb	50%
5	Kyungin	4kb	3kb	25%
6	ewha	12kb	6kb	50%
7	opendir	17kb	9kb	47%
평균압축률				44%

표 4 . 실험 데이터

각 HTML 문서의 압축률은 표4 와 같다. 무작위 데이터 추출 실험결과 44%정도의 압축률을 보여주었다.

5. 결 론

본 논문에서는 기존에 압축방법과는 달리 웹 페이지 자체 HTML의 반복되는 태그를 매크로로 치환하는 바이트 압축방법을 이용하여 저장공간 및 전송시간을 줄이고자 하였다. 물론 기존의 압축에 대한 방법은 많이 제안이 됐지만, 바이트 압축은 새로운 방향 제시이며, 또한 긍정적인 결과를 볼 수 있었다. 전자상거래의 활성화에 맞춰 웹 상에서의 무한한 정보교환은 본 연구에서 제안한 내용이 유용하게 활용될 수 있다.

[참 고 문 헌]

[1] David R. McIntyre and Michael A. Pechura. Data Compression using static Huffman code-decode tables. Digital Library is published by the Association for Computin Computing Machinery. Copyright 1999, 2000 ACM, Inc. pp.612-616 . Feb. 2000

[2] James J. Hunt, Kiem-phong Vo Walter F. Tichy. Delta Algorithms : An Empirical Analysis. ACM Transactions on software Engineering and Methodology V.7 N.2 . April, 1998

[3] Jeffrey C. Mogul, Fred Douglis, and Balachander Krishnamurthy. Potential Benefits of delta-encoding and data compresson for HTTP. Computer Communication Review V.27 N.4 . October, 1997

[4] MAXIME CROCHEMORE. Pattern-Matching and Text-Compression Algorithms ACM Computing Surveys V.28 N.1. March, 1996

[5] Debra A. Lelewer and Daniel S. Hirschberg. Data compression. ACM Comput.Surv 19,3. pp.261 - 296. Sep.1987

[6] Barron C. Housel and David B. Lindquist. WebExpress: a system for optimizing Web browsing in a wireless environment. Proceedings of the second annual international conference on Mobile computing and networking. pp.108 - 116. 1996,

[7] Henrik Frystyk Nielsen, Jim Gettys, Anaselm Baird-Smith, Eric Prud'hommeaux, Chris Lilley, Hakon Lie. Network Performance Effects of HTTP/1.1, CSS1, and PNG. Computer Communication Review V.27 N.4 . October, 1997

[8] Fox A, Gribble SD, Chawathe Y, Brewer EA. Adapting To Network And Client Variation Using Infrastructural Proxies - Lessons And Perspectives. IEEE Personal Communications V.5 N.4. pp.10-19. August, 1998