타입 정보 추출을 통한 질의 가능 XML 압축[1]

박명제[0] 민준기 정진완
한국과학기술원 전자전산학과 전산학전공
(jpark[0], jkmin, chungcw)@islab.kaist.ac.kr

A Queriable XML Compression Through An Extraction of Type Information

Myung-Jae Park[0] Jun-Ki Min Chin-Wan Chung
Division of Computer Science, Department of Electrical Engineering & Computer Science,
Korea Advanced Institute of Science and Technology

요 약

인터넷에서 널리 사용되는 HTML은 현재 데이터베이스 시스템과 같은 저장소 대신, 전형적인 파일 시스템에 저장되는 경우가 대부분이다. 마찬가지로, 최근에 인터넷 상에서의 데이터 교환 및 표현의 표준으로 부각되는 XML 역시 파일 시스템에 저장되는 경우가 많다. 하지만, XML 문서의 비정규적인 구조와 장황성 때문에, 디스크 공간이나 네트워크 대역폭이 정규적인 구조의 데이터에 비해 비효율적이다. 따라서, 이를 해결하고자, XML 문서의 압축에 관한 연구가 진행되었다. 하지만, 최근에 연구된 XML 압축 기법들은 압축한 XML 문서에 대한 질의를 지원하지 않거나, 질의를 지원하더라도 XML 문서의 데이터 값들의 특성을 고려하지 않고 단순히 기존의 압축 방법을 통해 XML 문서를 압축한다. 그러므로, 본 연구에서는 압축한 XML 문서에 대한 질의를 효율적으로 지원하는 XML 압축 기법을 제안한다. 본 연구에서는 태그를 Dictionary 압축으로 압축하며, 태그 별로 데이터 값들의 타입을 추출하여 추출한 타입에 적절한 압축 방법으로 데이터 값을 압축한다. 또한, 제안하는 압축 기법의 구현 및 성능 평가를 통하여, 구현한 시스템이 실생활에 사용되는 XML 문서들을 효율적으로 압축하며 향상된 질의 성능을 제공하는 것을 보인다.

1. 서론

인터넷의 확산은 다양한 형태의 전자 문서 양을 증가시켰다. 데이터를 전자 문서 형태로 표현하는 다양한 방법들 가운데 XML (eXtensible Markup Language) [1] 은 XML이 지니는 정보 표현의 유연성 때문에, 인터넷에서의 데이터 교환 (exchange) 및 표현 (representation) 의 표준으로 부각되고 있다. 그 결과, 인터넷에서의 XML 문서 사용은 나날이 증가하고 있으며, 이러한 XML 문서의 저장 [2, 3, 4] 및 검색 [5, 6] 에 관한 다양한 연구가 진행되었다. 하지만, 현재 대부분의 XML 문서는 HTML 문서처럼 전형적인 파일 시스템에 저장되는 경우가 대부분이다. 그러므로, XML이 데이터를 표현하는 진정한 표준으로 자리잡기 위해서는 파일 형태로 저장되는 XML 문서를 효율적으로 관리하는 기술에 관한 연구의 필요성이 대두되어 XML 문서의 압축 기술에 관한 연구가 진행되었다.

XMill [7] 은 압축한 XML 문서의 크기를 최소화하기 위한 압축 기법이다. 태그 (tag) 들은 Dictionary 압축을 통해 압축하며, 의미상 관련 (semantically related) 있는 데이터 값 (data value) 은 컨테이너 (container) 별로 분류한다. 또한, 범용 압축 라이브러리인 zlib [8] 을 통해 압축한 태그와 분류된 데이터 값을 압축한다. 하지만, 압축한 XML 문서의 구조와 본래 XML 문서의 구조가 다르므로, 압축한 XML 문서에 대한 질의는 처리하지 못한다. 즉, 질의 처리를 위해서는 압축한 XML 문서를 본래 XML 문서로 먼저 복원해야만 한다.

반면, 최근 연구된 XGrind [9] 는 압축한 XML 문서에 대한 질의를 처리하기 위한 압축 기법이다. 하지만, XGrind는 XML 문서의 특성을 고려하지 않고, 태그는 Dictionary 압축으로, 데이터 값은 Huffman 압축으로 압축한다. 압축한 XML 문서에 대한 질의를 처리하는 질의 처리기는 먼저 질의에서 주어진 태그들과 데이터 값들을 압축에 사용한 압축 방법들을 통해 압축한 다음, 압축한 XML 문서를 탐색하면서 해당 결과를 찾는다. 하지만, 레인지 질의 (range query) 의 경우, 레인지 내에 포함된 데이터 값인지의 여부를 판단하기 위해서는 본래 데이터 값으로의 복원이 항상 필요하다.

따라서, 본 연구에서는 압축한 XML 문서에 대한 질의를 효율적으로 처리하는 XML 압축 기법을 제안한다. 본 연구에서는 태그 별로 데이터 값들의 타입을 추출하여 추출한 타입에 적절한 압축 방법으로 압축한다. 또한, 본 연구에서 제안하는 XML 압축 기법은 압축에 사용한 정보가 변경되지 않는 Semi-adaptive 방식에 의해 압축하는 데이터의 위치와 관계없이 항상 동일한 값으로 압축하고, 본래 XML 문서의 구조와 압축한 XML 문서의 구조를 같도록 압축하므로 압축한 XML 문서에 대한 질의 처리는 본래 XML 문서로의 복원을 필요로 하지 않는다.

본 연구에서 제안하는 압축 기법을 기반으로 XML 압축 시스템을 구현하였으며, 실생활에서 사용되는 XML 문서와 다양한 종류의 질의를 통해 XML 압축 시스템의 성능을 평가하였다. 그 결과, XML 압축 시스템은 기존의 XML 압축 기법보다 향상된 질의 수행 시간과 적절한 압축비 (compression ratio) 를 보여준다.

2. 관련 연구

2.1 텍스트 압축 기법

일반적으로, 압축 기법들은 데이터 복원력에 따라, Lossy 압축과 Lossless 압축으로 구분된다. Lossy 압축은 데이터의 일부 정보를 제거하여 압축하므로, 압축한 데이터를 복원한 결과는 본래 데이터와 다르다. 따라서, 본 연구에서는 Lossy 압축을 고려하지 않는다. 반면, Lossless 압축에는 정해 놓은 통계치 (statistic) 를 사용하거나 전혀 사용하지 않는 Static 방식, 데이터의 통계치를 먼저 추출하고 추출한 통계치를 사용하여 압축하는 Semi-adaptive 방식, 그리고, 데이터를 압축하면서 통계치를 추출하는 Adaptive 방식이 있다. 특히, Semi-adaptive 방식에서 추출한 통계치는 어떠한 변경도 발생하지 않지만,

[1] 본 연구는 정보통신부의 대학 IT연구센터(ITRC) 지원을 받아 수행되었습니다.

Adaptive 방식의 통계치는 계속해서 변경이 발생한다.

Static 방식에는 데이터를 2진수로 변환하는 Binary 압축, 새로운 단어마다 다른 정수를 부여하여 부여한 정수들로 데이터를 변환하는 Dictionary 압축, 그리고, 선택한 기준 값과 다른 값들간의 차이들로 데이터를 변환하는 Differential 압축이 있다. 특히, Dictionary 압축에서 부여한 정수들간의 크기는 본래 데이터의 크기와 다르므로 본래 데이터들로의 복원이 필요하다. 반면, Differential 압축은 복원이 불필요하다. Semi-adaptive 방식에는 발생 빈도가 높은 문자를 짧은 값으로, 발생 빈도가 낮은 문자를 긴 값으로 변환하는 Huffman 압축이 있다. 이것은 추출한 통계치를 기반으로 생성한 Huffman Tree를 사용하여 변환 값을 제공하므로, 변환 값들간의 크기 비교는 복원을 통해서만 가능하다. Adaptive 방식에는 압축하면서 추출한 통계치를 기반으로 Huffman Tree를 계속해서 변경하면서 변환 값을 부여하는 Adaptive Huffman 압축이 있다.

2.2 XML 압축 기법

XML 압축에 관한 연구로는 XMill과 XGrind가 있다. XMill은 데이터 값을 태그로부터 구분하여, 의미상 관련 있는 데이터 값을 컨테이너 별로 분류한다. 또한, 태그는 Dictionary 압축을 통해 압축하며, 컨테이너에 포함된 데이터 값은 사용자가 제공하는 압축 방법으로 압축한다. 즉, 사용자가 압축 방법을 제공하지 않으면, 데이터 값을 압축하지 않는다. 마지막으로, XMill은 압축한 태그와 분류한 데이터 값을 zlib으로 압축한다. 하지만, XMill에 의해서 압축한 XML 문서는 데이터 값을 컨테이너 별로 분류했으므로 본래 XML 문서의 구조와 다른 구조를 지닌다. 따라서, XMill은 본래 XML 문서로의 복원 없이 압축한 XML 문서에 대한 질의를 처리하는 것은 불가능하다. 반면, 본래 XML 문서의 구조와 압축한 XML 문서의 구조를 동일하게 유지하는 XGrind는 압축한 XML 문서에 대한 질의 처리가 가능하다. XGrind는 DTD를 통해 데이터 값을 Huffman 압축이나 Dictionary 압축으로 압축하며, 태그는 Dictionary 압축으로 압축한다. 또한, 질의 처리기는 엘리먼트 (element) 를 방문할 때마다 루트 (root) 로부터 현재 방문한 엘리먼트까지의 압축된 경로 (path) 를 추출하여, 이것이 질의에서 주어진 경로를 압축한 경로와 동일한지를 확인한다. 또한, Huffman 압축이나 Dictionary 압축으로 압축한 데이터 값들간의 크기 비교는 불가능하므로, 레인지 질의의 경우, 압축한 데이터 값들의 복원이 반드시 필요하다.

3. 타입 추출을 통한 XML 압축

3.1 제안하는 XML 압축의 특징

본 연구에서 제안하는 XML 압축은 Semi-adaptive 방식으로, XML 문서 압축에 필요한 통계치를 미리 추출한다. 그 결과, 동일한 태그 및 데이터 값은 해당 태그와 데이터 값의 위치와는 관계없이 항상 같은 값으로 압축한다. 또한, 본 XML 압축은 XGrind처럼 압축한 XML 문서의 구조를 본래 XML 문서의 구조와 동일하게 유지하는 Homomorphism [9] 을 기반으로 효율적인 질의 처리가 가능하다. 만약, Adaptive 방식을 적용한다면, 통계치를 미리 추출하는데 소요되는 시간을 절약하여 압축 시간이 단축되지만 태그나 데이터 값의 위치에 따라 다른 값으로 압축되므로 본래 XML 문서로의 복원이 필요하다. 그리고, XMill처럼 데이터 값을 컨테이너 별로 분류한다면, 해당 컨테이너마다 데이터 값을 탐색하는 파일 포인터가 질의 처리를 위해 필요하다. 따라서, 이런 문제들은 질의 성능을 저하시킨다.

3.2 태그의 압축

본 연구에서는 Dictionary 압축으로 태그를 압축한다. 새로운 태그마다 0보다 큰 정수를 부여하고, 부여한 정수들로 태그들을 변환한다. 또한, XML 문서의 유일한 태그 수에 따라 다른 바이트를 사용하여 정수들을 표현함으로써 바이트의 사용을 최소화한다. 표 1은 태그 수에 따라 선택한 바이트를 보여준다. 일반적으로, 7비트로 표현 가능한 수가 127이므로 태그 수가 127 까지는 1바이트를 사용한다. (2, 4바이트의 경우도 마찬가지이다)

표 1. 선택된 바이트

유일한 태그 수	바이트
~ 127	1
128 ~ 32766	2
32767 ~	4

압축한 값이 태그임을 구분하고자, MSB (Most Significant Bit) 를 항상 1로 변환한다. 이것은 1바이트에서 7비트로 정수를 표현하여 MSB가 항상 0이므로 가능하다. 또한, 시작 태그와 끝 태그를 구분하고자, 0을 끝 태그에 할당하고, MSB를 1로 변환한다.

3.3 데이터 값의 압축

데이터 값을 압축하려면, 먼저 태그 별로 데이터 값들의 타입을 간단한 귀납적 추론으로 추출한다. 표 2는 추출 가능한 타입을 보여준다. 특히, 정수는 표 1에서 설명한 것처럼 세 종류로 구분하며, n은 임의의 바이트를 의미한다. 이처럼, 데이터 값의 타입이 추출되면, 타입에 따른 압축 방법으로 데이터 값들을 압축한다. Huffman 압축 (huff) 과 Dictionary 압축 (dict) 을 통해 압축한 데이터 값들간의 크기 비교는 불가능하므로, 레인지 질의의 경우, 데이터 값에 대한 복원이 필요하지만, int와 float의 경우에는 Binary 압축을 한 후 태그 별 데이터 값 중에서 가장 작은 값을 기준으로 Differential 압축을 하므로 데이터 값에 대한 복원이 불필요하다.

표 2. 추출 가능한 타입

타입	바이트	설명
int1	1	최대값 – 최소값 <127 인 정수들
int2	2	128< 최대값 – 최소값 <32766 인 정수들
int4	4	32767< 최대값 – 최소값 인 정수들
float	4	4 바이트를 사용하여 표현한 소수들
dict	1	유일한 단어의 수가 127개 이하인 텍스트
huff	n	128개 이상인 텍스트

태그의 경우처럼, 압축한 값이 데이터 값임을 구별하고자, MSB는 항상 0이다. 이것은 int1, int2, 그리고, int4는 각각 7비트, 15비트, 그리고, 31비트로 표현하고, float은 음수로 표현될 수 없으며, dict은 1바이트로 유일한 단어 수가 127개 이하의 경우를 표현하므로, MSB는 항상 0이다. 하지만, huff에 의해 압축한 데이터 값의 길이는 가변적이므로, 그 길이를 127로 나누어 나온 값 앞에 길이를 표현하는 1바이트를 추가하여 MSB를 항상 0이 되도록 한다.

3.4 XML 압축 시스템 구조도

그림 1은 XML 분석기와 XML 압축기로 구성된 시스템의 구조도이다. XML 분석기는 주어진 XML 문서에 대한 통계치 및 데이터 값의 타입을 추출하고, XML 압축기는 이를 기반으로 XML 문서를 압축하여 질의 가능하도록 압축한 XML 문서를 생성한다.

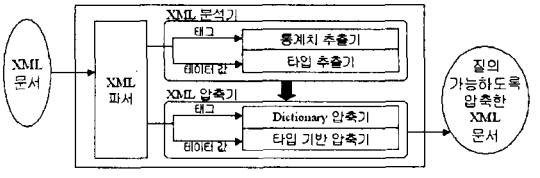

그림 1. XML 압축 시스템 구조도

XML 분석기는 태그 별로 정수를 부여하고 XML 문서의 유일한 태그 수를 계산하며 필요한 통계치를 추출하는 통계치 추출기와 태그 별로 데이터 값의 타입을 추출하는 타입 추론기로 구성된다. XML 압축기는 정수로 태그를 변환하고 유일한 태그의 수를 기반으로 선택한 최소한의 바이트로 변환한 태그들을 표현하는 Dictionary 압축기와 추출한 타입에 따라 데이터 값을 압축하는 타입 기반 압축기로 구성된다.

3.5 질의 처리기

질의 처리기는 기본적으로 루트부터 해당 엘리먼트까지의 모든 경로를 포함하는 단순경로 표현식 (simple path expression) 과 일부 경로만 포함하는 부분경로 표현식 (partial matching path expression), 그리고, 특정 값을 찾는 정확 질의 (exact matching query) 와 특정 범위 내의 값들을 찾는 레인지 질의를 지원한다.

먼저, 질의 처리기는 질의에 사용된 경로 내의 엘리먼트들을 통계치 추출기가 부여한 정수들로 변환하고, 압축한 XML 문서를 깊이 우선 검색 (Depth First Search) 으로 탐색하면서 방문하는 엘리먼트의 단순경로가 정수로 변환된 경로와 동일한가를 확인한다. 만약, 주어진 질의가 정확 질의라면, 질의 조건에 사용된 데이터 값을 타입 기반 압축기로 압축하고, 정수로 변환된 경로와 동일한 경로를 가지는 엘리먼트의 압축된 데이터 값이 질의 조건의 데이터 값을 압축한 값과 같은지를 비교한다. 그리고, 레인지 질의의 경우, 해당 데이터 값의 타입이 int1, int2, int4, 또는 float면, 압축한 데이터 값이 레인지 내에 포함하는지를 비교한다. 하지만, 해당 데이터 값의 타입이 dict이나 huff면, 데이터 값들에 대한 복원을 한 뒤 크기 비교를 한다.

4. 성능 평가

실험은 메인 메모리 384MB에 Solaris 2.5.1이 탑재된 Sun UltraSparc-II를 사용하였고, XML 문서는 로컬 디스크에 저장하였다. 실험에서, XMill은 사용자로부터 압축 방법이 제공되지 않으며, XGrind의 질의 처리기는 부분경로 표현식을 지원하지 않으므로, 이 기능을 추가적으로 구현하였다.

4.1 XML 문서

실생활에서 사용되는 네 개의 XML 문서를 사용하였으며, 표 3은 사용된 XML 문서들의 정보를 보여준다. 크기는 XML 문서의 크기 (MB) 를, 길이는 가장 긴 단순경로의 길이를, 태그는 유일한 태그 수를, int/float은 추출한 타입이 int1, int2, int4, 또는, float인 엘리먼트 수를, 그리고, dict는 타입이 dict인 엘리먼트 수를 의미한다.

표 3. XML 문서 종류

XML 문서	크기	길이	태그	int/float	dict
Auction	8.53	5	30	4	19
Baseball	17.06	6	46	19	5
Course	12.28	6	18	5	4
Shakespeare	15.30	5	21	0	0

4.2 XML 질의

XML 문서마다 다섯 개의 질의를 선택하였다. 질의 1은 단순경로 표현식을, 질의 2, 3은 부분경로 표현식을 통한 정확 질의와 레인지 질의를, 질의 4, 5는 복잡한 구조의 정확 질의와 레인지 질의를 의미한다. 또한, Auction과 Baseball은 dict 타입을, Course는 int 타입을, 그리고, Shakespeare는 huff 타입에 대한 정확 및 레인지 질의를 선택하였다. (공간 부족에 따라 질의 예는 생략한다.)

4.3 실험 결과

먼저, 그림 2는 압축 시간의 결과로써, XGrind가 가장 많이 소요되었다. 일반적으로, Huffman 압축이 Huffman Tree를 탐색하면서 압축값을 부여하므로 Differential 압축에 비해 비효율적이다. 그림 3은 압축비의 결과이다. 여기서, 압축비는 1-압축된 XML 문서/본래 XML 문서를 의미한다. 예상대로, XMill이 92%로 가장 좋았고, 본 시스템은 80%로써 XML 문서가 int, float, 그리고, dict 타입을 많이 포함하면, 더 나은 성능을 보일 것이다. 그림 4는 zlib의 성능을 확인하고자, XGrind와 본 시스템으로 압축한 XML 문서를 zlib으로 구성된 gzip으로 압축한 결과로써, 두 방법 모두 거의 비슷한 압축비를 보였다. 마지막으로, 그림 5는 질의 수행 시간의 결과로, 본 시스템이 모든 질의에 대해서 XGrind보다 나은 성능을 보였다. 질의 1, 2, 4는 최소한의 바이트로 정수를 표현하여 항상 2바이트를 사용하는 XGrind보다 나은 성능을 보였고, 질의 3, 5는 항상 복원하는 XGrind의 경우를 Differential 압축으로 최소화 하여, 1.4 배 향상된 성능을 보였다.

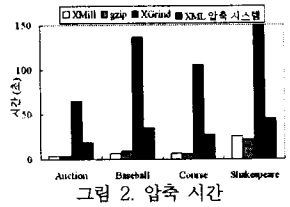

그림 2. 압축 시간

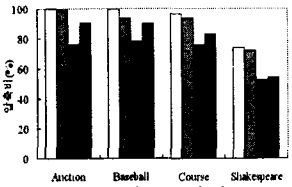

그림 3. 압축비

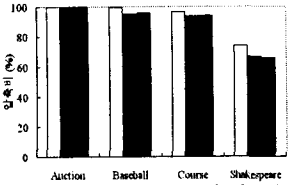

그림 4. gzip을 적용한 압축비

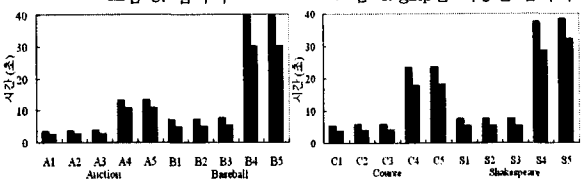

그림 5. 질의 수행 시간

5. 결론

본 연구에서는 압축한 XML 문서에 대한 질의를 효율적으로 지원하는 XML 압축 기법을 제안하였다. 태그는 Dictionary 압축을 기반으로 정수를 표현하는데 필요한 최소한의 바이트를 사용하여 표현하였다. 또한, 데이터 값은 타입 추출을 통해 추출한 타입에 적절한 압축 방법으로 압축하였다. 특히, 압축한 값들간의 크기 비교가 가능한 Differential 압축의 사용으로 압축한 데이터 값에 대한 복원이 필요한 경우를 최소화 하였다.

그리고, 실생활에 사용되는 XML 문서들과 다양한 종류의 질의들을 선택하여 구현한 시스템의 성능 평가를 한 결과, 구현한 시스템은 1.4 배 향상된 질의 처리 성능을 보였고, 압축비는 80%였다.

6. 참고 문헌

[1] T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler, Extensible Markup Language (XML) 1.0 W3C Recommendation, http://www.w3c.org/TR/REC-XML, 1998

[2] D. Florescu and D. Kossman, Storing and Querying XML Data using an RDBMS, IEEE Data Engineering Bulletin, 22(3):27-34, 1999

[3] T. Shimura, M. Yoshikawa, and S. Uemura, Storing and Retrieval of XML Documents using Object-Relational Databases, Proc. of 10th International Conference, DEXA, pp. 206-217, 1999

[4] I. Tatarinov, S. D. Viglas, K. Beyer, J. Shanmugasundaram, E. Shekita, and C. Zhang, Storing and Querying Ordered XML Using a Relational Database System, Proc. of the 2002 ACM SIGMOD International Conference on Management of Data, pp. 204-215, 2002

[5] M. F. Fernandez and D. Suciu, Optimizing Regular Path Expressions Using Graph Schemas, Proc. of the 14th International Conference on Data Engineering, pp. 14-23, 1998

[6] R. Goldman and J. Widom, DataGuides: Enable Query Formulation and Optimization in Semistructured DataBases, Proc. of 23rd International Conference on Very Large Data Bases, pp. 436-445, 1997

[7] H. Liefke and D. Suciu, XMill: An Efficient Compressor for XML Data, Proc. of the 2000 ACM SIGMOD International Conference on Management of Data, pp. 153-164, 2000

[8] J. Gailly and M. Adler, zlib 1.1.4, http://www.gzip.org/zlib/, 2002.

[9] P. M. Tolani and J. R. Haritsa, XGRIND: A Query-friendly XML Compressor, Proc. of 18th International Conference on Database Engineering, 2002