조선정보관리시스템에서의 갱신의 지역편중성을 갖는 XML 문서를 위한 XML 프록시 캐쉬 시스템

(An XML Proxy Cache System for XML Documents with Update Locality in Shipbuilding Information Management System)

김 낙 현 [†] 이 동 호 [††] 최 일 환 [†††] 김 형 주 [††††]
(Nak Hyun Kim) (Dong-Ho Lee) (Il-Hwan Choi) (Hyoung-Joo Kim)

요 약 XML의 등장에 따라 기존에 서로 상이한 응용에 의해 처리되어 질의 처리가 어려웠던 많은 정보들이 XML로 새롭게 기술되어 정보들과 정보들 사이의 연관정보에 대한 질의 처리가 가능하게 되었다. 조선 정보 관리 시스템을 개발하여 운영한 결과, 기존의 응용들로써는 어려웠던 다양한 질의 처리가 가능하게 되었는데 하나의 선박을 구성하는 커다란 XML 문서를 XDBox에 넣어 처리함에 있어 여러 가지 비효율적인 부분이 들어나게 되었다. 이러한 비효율성을 개선하기 위해서 빠른 질의처리를 위한 XML 프록시 캐쉬 시스템을 도입하게 되었다. 본 논문에서는 작업 흐름을 기술하고 있는 문서에서는 실제 작업에 따라 문서에 대한 갱신이 일어나기 때문에 갱신의 지역편중성이라는 특징을 고려하여 갱신블럭을 유지하는 방법으로 보다 효율적인 XML 프록시 캐쉬 시스템을 제안하였다.

키워드 : XML, 프록시, 캐쉬, 조선정보관리시스템, XDBox

Abstract XML makes it possible to query the information created and managed different applications, which is impossible if expressed in other structure or language. In using shipbuilding information management system, there is inefficiency in storing and querying such a large XML document in XDBox. XML proxy cache system is suggested to improve that. In this paper, we suggests XML proxy cache system with thought of update locality found in using shipbuilding information management system.

Key words : XML, Proxy, Cache, Shipbuilding, XDBox

1. 서 론

서로 다른 응용에 의해서 처리되던 정보들이 XML[1]이라는 공통된 언어를 가지고 처리되므로써 기존에는 불가능 했던 질의를 포함한 여러 가지 처리가 가능해지고 있다.

조선분야와 같은 제조업에서는 도면과 부품 목록, 작업지시서와 같은 많은 정보들이 존재한다. 기존에 전산화되지 않은 채로 처리되었던 정보들이 이제는 전산화되어서 처리되기는 하지만 작업지시서와 도면, 작업지시서와 부품 목록과 같은 각기 다른 응용에 의해서 유지, 관리되는 정보들 사이의 연관 정보는 통합되어 처리되지 못하고 사람에 의존하게 된다. 이에 이와 같은 연관 정보에 대한 질의를 원할 경우에 처리할 방법이 없게 되고 따라서 기존에 존재하는 정보들의 재활용성에 제한이 있을 수 밖에 없다.

이에 조선분야에서 사용되는 여러 정보들을 XML로 표현할 수 있는 DTD를 제안하고 이를 이용한 조선 정보 관리 시스템을 개발하여 운영하였다. 이용되는 사례를 분석한 결과 다음과 같은 특징을 가지고 있다. 우선 처리해야하는 XML 문서의 크기가 상당히 크다는 것이다. 10,000 개의 작업지시서만을 가지고 있는 선박의 경우에도 DOM[2]으로 표현했을 경우에 300메가 바이트 이상의 크기를 나타내었으며 이는 선박 생산 자체가 워낙 거대하고 필요한 작업이 많기 때문이다. 또한 조선

· 본 연구는 BK-21 정보기술 사업단과 정보통신부 ITRC(e-Biz 기술연구센터)에 의해 지원되었음
† 비 회 원 : 웹게이트
yusbha@hotmail.com
†† 비 회 원 : 한양대학교 컴퓨터공학과 교수
dhlee72@cse.hanyang.ac.kr
††† 학생회원 : 서울대학교 컴퓨터공학과
ihchoi@oopsla.snu.ac.kr
†††† 종신회원 : 서울대학교 전기전자컴퓨터공학부 교수
hjk@oopsla.snu.ac.kr
논문접수 : 2003년 7월 21일
심사완료 : 2005년 5월 16일

정보 관리 시스템에서 대상으로 하는 정보는 실제 선박을 만들어 나가는데 필요한 정보이기 때문에 한번에 모든 정보가 사용이 되는 것이 아니라 진행되는 과정에 따라서 몇 개의 작업지시서만이 사용이 된다. 결과적으로 갱신작업도 부분적으로 나타날 뿐 전체 문서에서 임의적으로 발생하는 것은 아니다.

이러한 갱신의 지역 편중성을 고려하면 기존에 제안되었던 XML 캐쉬에 있어서 원본 문서와의 일관성을 위해서 모든 갱신에 대해서 캐쉬의 적중률이 떨어지는 문제나 적중률을 높이기 위해서 과도한 정보를 유지하고 있어야 하는 문제를 해결할 수 있는 방안을 도출할 수 있다.

본 논문에서는 이러한 갱신의 지역편중성을 고려한 XML 프록시 캐쉬 시스템을 제안하여 이러한 큰 문서를 처리함에 있어서 보다 효율적인 처리가 가능하도록 하였다.

본 논문에서는 2장에서 조선 정보 관리 시스템과 그의 기반이 되는 XDBox에 대해서 소개를 하고 나타난 특징들을 살펴보고 3장에서는 관련 연구를 살펴보며 4장에서는 본 논문에서 제시하는 XML 프록시 캐쉬 시스템의 구조와 기법에 대해서 논하고 5장에서는 제시된 XML 프록시 캐쉬 시스템이 어떤 특징을 가지고 있는가에 대한 실험 결과를 제시하며 6장에서는 전체적으로 결론을 맺는 것으로 구성된다.

2. 배경지식

2.1 조선 정보 관리 시스템

2.1.1 조선 정보 관리 시스템의 소개

하나의 선박을 생산하는데 필요한 자료는 크게 도면과 작업지시서로 구분할 수 있다. 도면의 경우에 현재 CAD와 같은 도면 관리 시스템으로 생성되고 관리되고 있는데 여기서 나타나는 정보는 XML로 구성하여 처리할 수 있지만 실제 XML로 처리되었을 때 그 효용성에 대해서는 추가적인 연구가 필요하다. 다른 정보와의 연계에 있어서는 도면상에서 추출할 수 있는 메타정보만으로도 충분하다고 본다. 그림 1에서 볼 수 있는 작업지시서의 경우에는 작업지시서의 업무 처리에 필요한 부분과 실제 생산정보를 담고 있는 부분으로 나눌 수 있다. 이 작업지시서들은 서로 생산 순서에 따른 연관된 정보를 서로 가지고 있다. 또한 작업지시서에는 대상이 되는 부품의 도면과의 연관성을 나타내는 도면의 메타정보가 필요하게 된다.

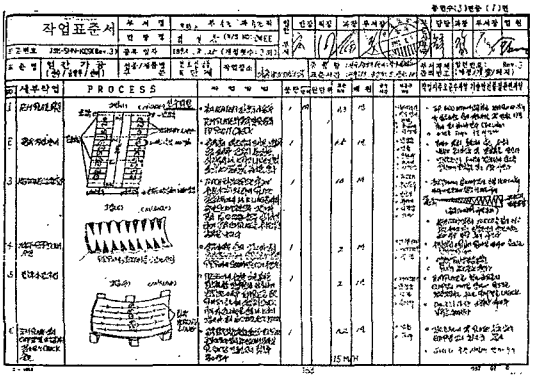


그림 1 실제 작업지시서

```
<!-- ShipbuildingInformation.DTD -->
<!-- Copyright (c) 2002 OOPSLA SNU All Rights Reserved. -->
<!-- Email comments to nhkim@oopsla.snu.ac.kr -->

<!-- 선박 -->
<!ELEMENT Ship (Work, DrawingMeta) >
<!ATTLIST Ship
        Id              ID              #REQUIRED
        Name            CDATA           #REQUIRED>

<!-- 작업지시서들 -->
<!ELEMENT Works (Works*, Work*) >
<!ATTLIST Works
        Name            CDATA           #REQUIRED>

<!-- 작업지시서 -->
<!ELEMENT Work (Format, Signature*, Content*) >

<!-- 형식 -->
<!ELEMENT Format>
<!ATTLIST Format
        Id              ID              #REQUIRED
        Name            CDATA           #REQUIRED
        RegiDate        CDATA           #REQUIRED
        Place           CDATA           #REQUIRED
        Part            CDATA           #REQUIRED
        LineForeman     CDATA           #REQUIRED
        Quantity        CDATA           #REQUIRED
        Time            CDATA           #REQUIRED>

<!-- 서명 -->
<!ELEMENT Signature>
<!ATTLIST Signature
        Position        CDATA   #REQUIRED
        Id              IDREF   #REQUIRED>

<!-- 작업내용 -->
<!ELEMENT Content (Process, Tool*, Notice*) >
<!ATTLIST Content
        Id              ID      #REQUIRED
        Name            CDATA   #REQUIRED
        DrawingMeta     CDATA   #IMPLIED
        Quantity        CDATA   #REQUIRED
        ManHour         CDATA   #REQUIRED
        Assigned        CDATA   #REQUIRED>

<!-- 작업방법 -->
<!ELEMENT Process (#PCDATA | DrawingMetaContent)+ >

<!-- 작업방법그림 -->
<!ELEMENT DrawingMetaContent>
<!ATTLIST DrawingMetaContent
        Id              IDREF   #REQUIRED>

<!-- 도구 -->
<!ELEMENT Tool (#PCDATA) >

<!-- 작업시 주요준수사항 -->
<!ELEMENT Notice (#PCDATA) >
<!-- 도면 메타정보들 -->
<!ELEMENT       DrawingMetas (DrawingMetas*, DrawingMeta) >
<!ATTLIST       DrawingMetas
        Name            CDATA   #REQUIRED>

<!-- 도면 메타정보 내용 -->
<!ELEMENT DrawingMetaContent>
<!ATTLIST DrawingMetaContent
        Id              ID      #REQUIRED
        PosX            CDATA   #REQUIRED
        PosY            CDATA   #REQUIRED>
```

그림 2 조선 정보 DTD

그림 2의 조선정보 DTD는 조선정보 관리 시스템내에서 각 작업지시서들을 XML로 표현하기 위한 DTD이다.

이렇게 구성된 DTD에 기반하여 그림 3에서와 같이 정보를 처리, 관리할 수 있는 조선 정보 관리 시스템을 운영하여 다음 절과 같은 결과를 얻을 수 있었다.

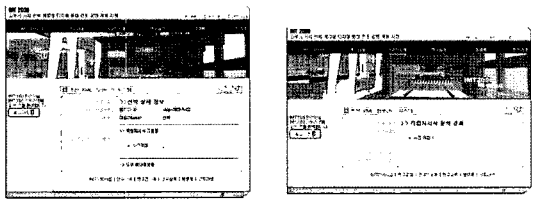


그림 3 조선 정보 관리 시스템

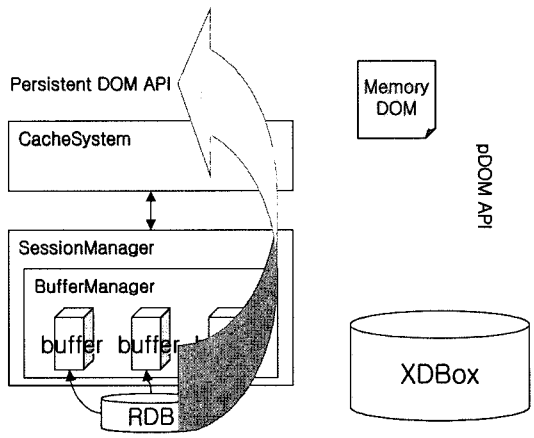


그림 4 XDBox 구조

2.1.2 조선 정보 관리 시스템내의 정보들의 특징

조선 정보 관리 시스템을 운영하면서 볼 수 있었던 정보들의 특징은 다음과 같다.

첫째로 선박 설계 자료는 단 하나의 선박만을 생산하는데 사용이 된다는 것이다. 다른 기계분야 등과 다르게 하나의 설계 자료가 일회성 자료가 되는 것이다. 따라서 하나의 선박을 이루는 자료들은 다음의 선박을 설계, 생산하는데 있어서 재활용이 가능하도록 구성이 되는 것이 필수적이다.

둘째로 조선 정보 전체는 하나의 큰 트리 구조를 이루게 되지만 작업지시서와 같은 부분은 독립적으로 하나의 문서를 이룰 수 있는 정보 단위가 된다는 것이다. 실제로 적용이 될 때는 작업이 진행될 때 하나의 작업지시서 단위로 진행이 되기 때문에 시스템을 구현할 때도 이와 같은 부분이 중요한 정보가 되며 실제 전체에 대한 갱신작업은 주로 이와 같은 작은 정보 단위로 일어나며 그것도 보다 반복적으로 일어남을 알 수 있다. 여기서 갱신이 지역 편중성을 가지고 있는 것을 알 수 있다.

세째로 작업지시서, 작업내용, 도면 메타정보의 연관정보들이 상당히 중요하게 된다. 동일한 작업내용이더라도 다른 작업내용들과 어떤 순서에 따라서 배열이 되어 있는가, 혹은 어떤 작업지시서에 속해있는가도 상당히 중요한 정보이다.

2.2 XDBox

2.2.1 XDBox의 소개

XDBox[3]는 XML 문서를 자바 DOM 객체로 표현하고 그 DOM 객체를 관계형 데이타베이스의 LOB(Large Object Block)을 이용해서 저장하는 구조를 갖는 시스템이다. 저장된 DOM 객체를 이용하기 위해서 영속 DOM API를 제공하여 메모리 상에 전체 DOM 객체를 운용하지 아니하고도 DOM에 접근할 수 있도록 하고 있다.

또한 효율적인 입출력을 위해서 버퍼 시스템과 클라이언트를 위한 캐쉬 시스템을 제공하고 있다.

2.2.2 XDBox의 문제점과 해결방안

저장된 DOM 객체에 접근하는 방법으로 영속 DOM API를 제공하고 있지만 버퍼 시스템을 이용하더라도 질의 처리와 같은 작업을 수행함에 있어서 속도의 저하가 심각하다. 보다 효율적인 처리를 위해서 단순 질의 작업시에는 메모리 상에 한번에 DOM을 다시 생성해서 접근하는 것이 보다 빠른 처리 결과를 얻을 수 있다.

하지만 반복되는 동일한 질의를 처리함에 있어서는 직접 영속 DOM API를 접근하는 방식이나 한번에 모든 노드를 DOM으로 구성한 다음에 접근하는 방식이나 모두 비효율적인 요소를 가지고 있으며 이 경우에 질의에 대한 캐쉬 시스템을 이용하여 동일한 질의 처리 작업 시의 중복되는 부하를 줄이는 것이 보다 효율적이다.

3. 관련 연구

XML이나 기존의 반구조적자료(semistructured data)의 경우에 캐쉬는 단순히 접근 빈도에 의해 캐쉬의 내용을 결정하는 것이 아니라 서로 연관된 정보를 바탕으로 결정되어야 한다는 연구가 있어왔다. 기존의 관계형 데이타베이스에서의 질의에 대한 캐쉬에 대한 연구에서 질의에 따른 정보에 의해 캐쉬를 저장할 경우에 더 효율적인 캐쉬 운용이 가능하다는 연구 결과를 보였다[4].

반구조적자료(semistructured data)에 있어서 실체뷰(materialized view)를 운용하는 방식으로 캐쉬를 유지하면서 캐쉬의 내용의 유효성을 더 오래 지속시키기 위해서 질의를 수행하는 과정에서 필요로 했던 모든 노드들을 저장하는 방식이 제안되기도 하였고[5], XML로 넘어오면서 보다 정교해진 형태로 트리의 마지막 노드

들에 있는 값에 대한 비교를 노드들에 대한 비교와 구별하는 두단계 정책으로 보다 더 오래 유효성을 지속하려는 노력과 그 결과를 XQL에 기반한 웹캐쉬에 적용하려는 노력[6-8]이 있었다.

또한 캐쉬를 가능한한 시스템의 말단부에 위치시키고 스스로 정해진 리소스에 따라서 관리되는 시스템을 구현하고자 하는 시도도 있었다.[9,10]

또한 캐쉬 시스템의 교환 알고리즘으로써 가장 간단하고 효율적인 알고리즘으로 LFU와 LRU를 생각할 수 있는데 이를 동시에 고려하는 알고리즘을 만들고자하는 노력이 계속되어 왔다.[11-13]

과거의 접근중 일정 단계 이전의 접근을 가지고 고려하는 경우[12]와 LFU와 LRU에 대해 동시에 고려를 하면서도 과거의 접근 정보를 기록하지 않는 구조를 가지면서 LFU와 LRU에 대한 비중을 선택할 수 있는 방법을 사용하기도 한다.[11]

4. XML 프록시 캐쉬 시스템

4.1 시스템 구조

XML 프록시 캐쉬 시스템은 다음 그림과 같이 이루어져 있다.

그림에서 질의파서(Query Parser)는 들어온 질의를 분석하는 모듈이다. 들어온 질의는 트리의 형태(Q-Tree)로 구성이 된다.

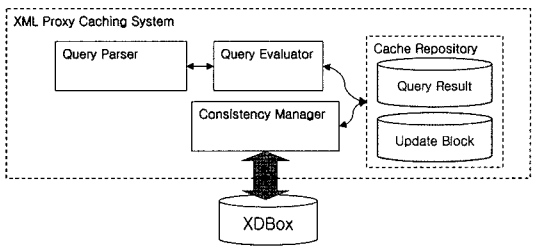


그림 5 XML 프록시 캐쉬 시스템 구조

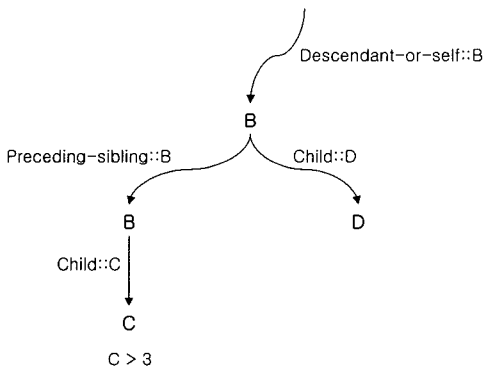


그림 6 질의 트리

예를 들어 //B[preceding-sibling::B/C > 3]/D과 같은 형태의 질의는 그림 6과 같은 형태의 트리로 재구성된다.

질의처리기(Query Evaluator)는 질의 트리를 가지고 캐쉬저장소(Cache Repository)안에 결과가 있을 경우에는 그 결과를 제공하고 결과가 없을 경우에는 하부의 XDBox에 보내어 질의를 처리한다.

새로 질의가 처리되었을 경우에 캐쉬저장소(Cache Repository)안에 처리된 질의를 넣는다.

XDBox안에 들어있는 XML 문서에 갱신이 이루어질 경우에는 캐쉬저장소(Cache Repository)내의 정보들과 원 XML 문서사이에 일관성 문제가 발생하게 된다.

질의의 결과만을 저장하였을 경우에는 어떤 부분에 갱신이 발생하여도 그 결과를 신뢰할 수 없기 때문에 모든 결과를 무효화하여야 한다. 이럴 경우에는 갱신이 일어날 때마다 무효화가 일어나기 때문에 프록시의 적중률이 매우 낮아지게 된다.

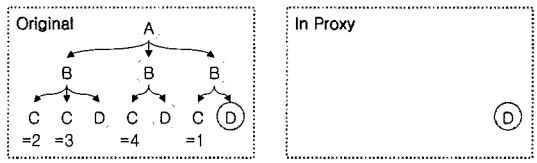


그림 7 결과 노드만을 기록

기존의 방법에 의해서 질의를 처리하는데 관여한 모든 노드들을 기록하고 있다면 저장된 질의의 결과에 대해서도 동일하게 갱신을 수행해줄 수 있기 때문에 결과를 무효화할 필요가 없게 된다. 하지만 이럴 경우에는 기존 질의의 결과에 비해서 훨씬 많은 중간 노드들을 기록하고 있어야 하는 문제점이 있을 수 있다. 최악의 경우에 원 문서와 동일한 크기의 정보를 유지하여야 할 수도 있다.

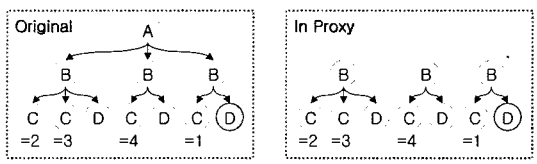


그림 8 모든 과정 노드들을 기록

하지만 조선 정보 관리 시스템에서 작업 흐름의 정보를 기록할 때에 실제 상황에서는 모든 문서가 동일하게 갱신이 되는 것이 아니라 현재 작업이 이루어지고 있는 부분에서만 갱신이 일어나는 경향이 있다.

이러한 경향을 갱신의 지역편중성(Update Locality)라고 하고 이러한 경향이 나타날 때에는 모든 과정 노

드들을 기록할 필요가 없음을 알 수 있다. 실제로 갱신이 주로 나타나는 부분(갱신 블럭)을 따로 분리해서 생각한다면 갱신 블럭에 있어서의 노드들과 그로 인해 영향받는 노드들, 그리고 결과 노드들만을 저장하면 갱신이 갱신 블럭에서만 일어났을 경우에는 위의 방법과 동일한 결과를 보다 적은 노드들을 저장하면서도 얻을 수 있다는 것을 알 수 있다.

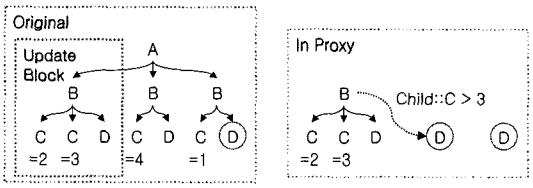


그림 9 갱신블럭(Update Block)내의 노드만을 추가로 기록

그림에서 갱신 블록(Update Block)내에서 갱신이 일어날 경우에는 그림 7과 같은 결과를, 갱신 블럭외에서 갱신이 일어날 경우에는 그림 8과 같은 결과를 가져오는 것을 알 수 있다.

XML 프록시 캐쉬 시스템내의 자원이 충분하다면 원문서의 모든 부분을 갱신 블럭으로 삼는다면 최고의 적중률을 유지할 수 있지만 일반적인 경우에 있어서 원문서와 같은 크기의 정보를 유지한다는 것은 불가능한 일이다.

4.2 교환 정책

제한된 용량의 캐쉬 시스템에서 필요한 공간을 확보하기 위해서 교환의 대상을 찾는 알고리즘들로써 LRU(Least Recently Used)와 LFU(Least Frequently Used)가 있다. LRU는 캐쉬 내용의 최신성에 따라서 교환의 대상을 정하고 LFU는 캐쉬 내용의 빈도성에 따라서 교환의 대상을 정하게 된다.

LFU와 LRU의 특징을 동시에 고려할 수 있는 몇가지 방법들이 제시되었는데 고려해야 할 점은 LRU와 LFU중 어느 쪽에 비중을 더 둘 것인지를 선택할 수 있는 형태가 되어야 하며 접근 시간을 모두 기록할 필요없이 최종적인 값만으로 지속적인 정보의 유지가 될 수 있어야 한다는 것이다.

본 논문에서는 여러 제안들[11-13] 중에 과거 접근에 대한 정보 저장이 필요없고 LRU와 LFU를 보다 유연하게 선택할 수 있는 방법[11]을 사용할 것이다.

이 방법에서는 우선 현재 시각에 대한 접근 시간의 가중함수로 $F=\left(\frac{1}{2}\right)^{(\lambda\times\Delta T)}$라는 함수를 제시하고 있고 여기서 λ는 0에서 1까지 변하는 값으로 0일 경우에는 LFU를 의미하게 되고 1일 경우에는 LRU를 의미하게 된다. ΔT는 현재 시각부터 접근 시각까지의 시간차이다.

접근시간들의 가중함수의 값들의 합을 CRF(Combined Recency Frequency)라는 값이라고 하고 이 값으로 캐쉬에서 버릴지 말지에 대한 기준을 삼게 된다.

본 논문에서 제안하고 있는 XML 프록시 캐쉬 시스템에서는 제한된 메모리내에서 각 질의의 결과들과 갱신 블록 양쪽을 다 유지하여야 한다. 각각의 내용들을 유지할지 버릴지에 대한 기준이 되는 값을 만들기 위해서 위에서 언급한 LRFU를 차용하여 값을 도출하도록 하겠다.

LRFU를 다시 생각해보면 LRU와 LFU를 동시에 고려한 CRF라는 값이 교환여부에 대한 결정 인자로 사용되고 있다.

$$CRF=\sum F=\sum\left(\frac{1}{2}\right)^{(\lambda\times\Delta T)}=\left(\frac{1}{2}\right)^{(\lambda\times\Delta T_{estimated})}$$

위 식에서 구해진 $\Delta T_{estimated}$를 통해서 LRU와 LFU가 모두 고려된 하나의 동등한 빈도값을 얻는다.

하나의 질의에 대해 절충된 빈도값과 그 질의를 XDBox에 수행할 때에 걸리는 시간, 그 질의에 대한 결과를 저장하고자 할 때 필요한 메모리의 값을 각각 eQF(Estimated Query Frequency), QPT(Query Process Time), RM(Result Memory)라고 정의한다.

하나의 갱신 블록에 대해 갱신이 일어나는 절충된 빈도값과 갱신 블록을 저장하는데 필요한 메모리의 값을 각각 eUF(Estimated Update Frequency), UM(Update Block Memory)이라고 정의한다.

갱신 블록이 존재하면 위의 그림에서와 같이 갱신 블록이 질의의 결과에 영향을 미치는 연결고리를 저장하고 있어야 하고 또한 갱신 블록내의 갱신으로 인해서 질의의 결과가 될 수 있는 예비 결과들도 또한 저장하여야 한다. 이를 URM(Update block and Result Memory)이라고 정의한다.

절충된 전체 갱신 빈도값을 eTUF(estimated Total Update Frequency)라고 정의한다.

이럴 때 프록시 캐쉬 시스템을 유지하므로써 얻는 이득을 계산하면 다음과 같다.

$$benefit=\frac{\sum eQF\times QPT}{\sum(eTUF-eUF)}$$

이 값은 줄일 수 있는 질의 수행 시간이 되고 RM과 UM과 URM이 미리 지정된 값이하로 유지되면서 benefit이 최대값이 되는 지점을 찾게 되면 제한된 메모리내에서의 최적화된 프록시 캐쉬 시스템을 유지할 수 있음을 알 수 있다.

조선정보관리시스템에서는 나타나는 문서의 유형이 특정한 패턴을 따르고 있음을 2.1.2절에서 보였으며 이러한 패턴의 문서에 있어서 benefit이 최대값이 되는 부분은 실험을 통해서 얻을 수 있다.

5. 실험 및 분석

5.1 실험 환경

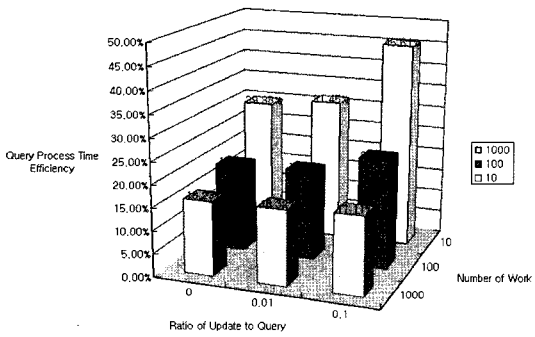
실험은 프록시 캐쉬 시스템이 질의의 중복성, 질의에 대해 갱신이 일어나는 빈도와 갱신의 지역편중성 정도에 따라서 영향을 받기 때문에 이 값들에 대해서 실험을 진행하였다.

전체 자료를 가지고 실험을 진행하는데 무리가 있기 때문에 작업지시서는 10,000개로 제한하였고 지역편중성이 많이 나타나는 곳과 적게 나타나는 곳이 모두 포함될 수 있도록 재구성하였고 질의도 그러한 상황을 나타낼 수 있도록 작성하였다.

작업지시서가 10,000개가 있는 상황에서 하나의 질의에 대해서 갱신의 지역 편중성이 있는 경우와 없는 경우, 일정 정도의 지역 편중성이 있는 경우에 대해서 질의의 중복성이 나타나는 경우와 아닌 경우에 대해서 실험을 수행하였다.

5.2 실험 결과

첫 번째 경우로 하나의 동일한 질의가 100번 주어지는 경우에 대해서 갱신의 지역 편중성이 아주 높아서 하나의 갱신블럭(Update Block)내에서만 계속 갱신이 일어나는 경우에 질의에 대한 갱신의 빈도가 어떤 영향을 주는지에 대해서 실험하였다.



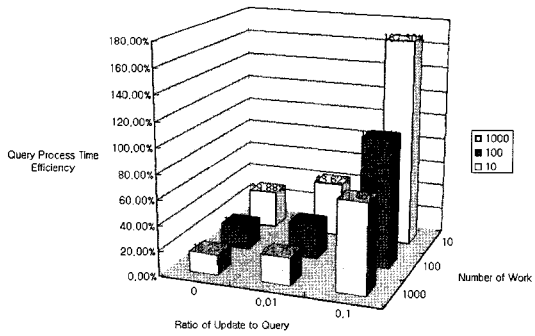
	0	0.01	0.1
1000	16.23%	16.32%	17.11%
100	19.39%	19.85%	24.05%
10	29.88%	31.45%	45.61%

그림 10 하나의 질의에 대한 지역 편중성이 심할 때의 응답시간

갱신의 지역편중성은 동일한 갱신블럭(Update Block)내에 평균적으로 몇 번이나 갱신이 발생하였는가로 나타낸다.

이 경우는 가장 이상적인 형태로 모든 질의가 중복이 되어 항상 프록시내에서 처리가 가능하며 갱신이 항상 갱신블럭(Update Block)내에서만 일어나기 때문에 처리 비용이 상당히 제한적이다. 가장 효율이 나쁜 경우인 질의 한번에 갱신 한번이 일어나는 경우에도 어느정도 높은 효율을 보여주게 된다.

두 번째 경우로 하나의 동일한 질의가 100번 주어지는 경우에 대해서 갱신의 지역 편중성이 없어서 갱신이 매번 다른 갱신블럭(Update Block)내에서만 계속 갱신이 일어나는 경우에 질의에 대한 갱신의 빈도가 어떤 영향을 주는지에 대해서 실험하였다.

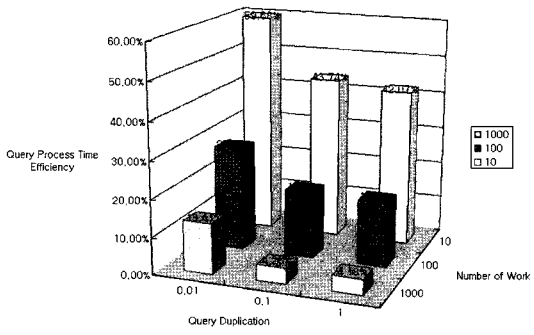


	0	0.01	0.1
1000	16.23%	21.75%	71.45%
100	19.39%	27.67%	102.21%
10	29.88%	43.62%	167.30%

그림 11 하나의 질의에 대한 지역 편중성이 없을 때의 응답시간

위의 실험결과에 비해서 전반적으로 효율이 나빠졌는데 갱신 빈도가 높아질수록 갱신블럭(Update Block)을 구성하는 것이 아무런 이득을 주지 못하기 때문에 이런 형태의 경우에는 본 논문에서 제안한 프록시 캐쉬 시스템은 의미가 없어지게 된다.

세 번째 경우로 1000개의 질의를 수행할 때 질의에 대한 갱신의 비율을 0.1로 하고 지역 편중성을 4로 잡고 질의의 중복성을 변화해 가면서 실험을 하였다.



	0.01	0.1	1
1000	13.41%	4.26%	4.18%
100	27.82%	18.04%	17.59%
10	59.68%	43.74%	42.07%

그림 12 질의의 중복성에 따른 응답시간

질의의 중복성은 실험이 모두 수행된 후에 전체 질의에 대한 평균적으로 중복된 질의의 비율로 나타낸다. 질의에 대한 갱신의 비율은 질의 한번당 몇 번의 갱신이 발생하는 가로 나타낸다.

지역 편중성이 전혀 없는 경우에는 매번 프록시의 내용을 새로 구성하고 처리를 해야 하기 때문에 원문서에 직접 질의하는 경우에 비해서 효율이 나빠지게 된다.

위의 실험의 결과로 현재 구성이 된 프록시 캐쉬 시스템은 갱신의 지역 편중성이 높을 수록 효율이 좋아짐을 보이고 모든 캐쉬 시스템과 동일하게 질의에 비해서 갱신의 비율이 낮을 수록, 질의가 중복될 수록 효율이 좋아짐을 보이고 있다.

6. 결 론

최근에 보다 유연한 정보의 표현을 위해서 XML의 사용이 늘어가고 있는 추세이다. 조선 분야에 있어서의 자료는 자료사이의 연관성이나 자료 자체를 나타내는데 있어서 현재 XML로 표현하는 것이 가장 타당하다. 본 논문에서는 실제 조선 분야의 자료에 대한 분석을 실시하고 그 자료를 XML로 모델링하여 DTD를 제시하고 이를 관리하는 조선 정보 관리 시스템을 개발하여 운영하였다. 그 결과로 나타난 특징이 갱신의 지역편중성이다.

XDBox에 기반하여 조선 정보 관리 시스템을 구현함에 있어서 조선 정보의 갱신 지역편중성이 나타남에 주목하여 이에 최적화된 XML 프록시 캐쉬 시스템을 제안하였다.

지역 편중성이라는 특징에 따라 결과만을 저장하여 트리 형태로 표현된 XML 문서에 하나의 노드에 대한 변경에도 캐쉬에 저장된 정보가 의미없어지는 것에 대한 단점과 질의를 처리할 때 그 결과와 그 결과를 도출하기 위해 접근된 모든 노드를 저장하여 갱신이 일어날 때에도 원 문서와의 일관성을 유지하지만 캐쉬내에 보다 많은 공간을 필요로 하는 단점을 개선하는 방법으로 갱신 블록을 지정하고 그 내부의 노드들에 대해서는 저장 공간을 할애하고 다른 부분에 있어서는 질의의 결과와 예비 결과 노드만을 저장하여 적은 공간에서도 갱신에 덜 민감한 캐쉬 구조를 제안하였다.

본 논문에서는 XML의 구조를 트리 모델로 보고 갱신 블록을 서브 트리로 한정하여 처리하였는데 ID, IDREF에 의한 참조를 고려하면 XML의 구조를 그래프 모델로 볼 수 있고 이런 경우에 갱신 블록의 선정과 그 처리에 대한 추가 연구가 필요하다.

참 고 문 헌

[1] T. Bray, J. Paoli, and C. M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0. W3C Recommendation, Feb. 1998. http://www.w3.org/TR/REC-xml/

[2] A. L. Hors, P. L. Hegaret, L. Wood, G. Nicol, J. Robie, M. Champion, S. Byrne. Document Object Model (DOM) Level 2 Core Specification. 13 November 2000. http://www.w3.org/TR/DOM-Level-2-Core/

[3] ITcamp, XDBox Technical Report, http://itcamp.co.kr/TR/xdbox.doc, 2002.

[4] S. Dar and M. J. Franklin and B. Jonsson. Semantic Data Caching and Replacement. In 22nd VLDB, pp. 330-341, 1996.

[5] S. Abiteboul, J. McHugh, M. Rys, V. Vassalos, and J. Winner. Incremental Maintenance for Materialized Views over Semistructured Data. In Int. Conference on Very Large Data Bases, pp. 38-40, August 1998.

[6] L. Chen and E. A. Rundensteiner. Aggregate Path Index fo Incremental Web View Maintenance. In *The 2nd Int. Workshop on Advanced Issues of E-Commerce and Web-based Information Systems*, San Jose, to appear, June 2000.

[7] L. Chen, E. A. Rundensteiner and S. Wong. XCache-A Semantic Caching System for XML Queries. In ACM SIGMOD June 2002.

[8] L. P. Quan, L. Chen and E. A. Rundensteiner. Argos: Efficient Refresh in an XQL-Based Web Caching System. In WebDB 2000, pp. 23-28 May 2000.

[9] K. Amiri, S. Park, R. Tewari, and S. Padmanabhan. DBProxy: A self-managing edge-of-network data cache. IBM Research Report, 2002.

[10] K. Amiri, R. Tewari, S. Park, and S. Padmanabhan. On Space management in a dynamic edge data cache. In WebDB 2002.

[11] D. Lee, J. Choi, J. Kim, S. Noh, S. Min, Y. Cho and C. Kim. LRFU(Least Recently/Frequently Used) Replacement Policy: A Spectrum of Block Replacement Policies. in Proceedings of the 1999 ACM SIGMETRICS Conference, May 2-4, 1999.

[12] E. J. O'Neil, P. E. O'Neil, and G. Weikum, The LRU-K Page Replacement Algorithm For Database Disk Buffering. in Proceedings of the 1993 ACM SIGMOD Conference, pp. 297-306, 1993.

[13] J. T. Robinson and N. V. Devarakonda, Data Cache Management Using Frequently-Based Replacement. in Proceedings of the 1990 ACM SIGMETRICS Conference, pp. 134-142, 1990.

김 낙 현

1996년 2월 서울대학교 기계설계학과 학사. 2005년 8월 서울대학교 컴퓨터공학과 석사

이 동 호

1995년 2월 홍익대학교 컴퓨터공학과(학사). 1997년 2월 서울대학교 컴퓨터공학과(석사). 2001년 2월 서울대학교 전기·컴퓨터공학부(박사). 2001년 3월~2004년 2월 삼성전자 소프트웨어센터(CTO) 책임연구원. 2004년 3월~현재 한양대학교 컴퓨터공학과 조교수. 관심분야는 멀티미디어 데이타베이스, 멀티미디어 정보검색, 내장형 데이타베이스

최 일 환

1996년 서울대학교 컴퓨터공학과 학사
1998년 서울대학교 컴퓨터공학과 석사
1998년~현재 서울대학교 컴퓨터공학과 박사과정 재학중. 관심분야는 XML, 데이타베이스

김 형 주

정보과학회논문지 : 컴퓨팅의 실제
제 11 권 제 3 호 참조