제21회 한국정보처리학회 춘계학술발표대회 논문집 제11권 제1호 (2004. 5)

XML 스트리밍 데이터에 대한 시구간 질의 처리 시스템 모델†

한승철 O 강현철
중앙대학교 컴퓨터공학부
schan@dblab.cse.cau.ac.kr hckang@cau.ac.kr

Models of Time-based Query Processing System for XML Streaming Data

Seungchul Han[0] Hyunchul Kang
School of Computer Science and Engineering, Chung-Ang University

요 약

스트리밍 데이터 처리는 여러 분야에서 많은 관심을 가지고 활발한 연구가 수행되고 있다. 특히 모니터링, 센서 네트워크등의 응용 분야에서 끊임없이 생성되는 대량의 스트리밍 데이터 처리를 위한 요구가 높아지고 있다. 본 논문에서는 XML 스트리밍 데이터에 대한 시구간 질의 처리 시스템 모델을 제시한다. 스트리밍 데이터 모델로 웹상의 데이터 교환 표준으로 자리잡은 XML 을 사용하였고 연속적인 질의 처리를 위해 시구간이 명시된 XQuery 를 질의 모델로 사용하였다. 제시된 시스템에서는 질의 처리의 성능 향상을 기하기 위해 질의 결과 값을 백그라운드 프로세싱으로 생성하고 그것을 캐싱하여 후속 질의의 결과 값에 반영하는 캐싱 기법을 제공한다.

1. 서론

모니터링, 센서네트워크등 연속적인 스트리밍 데이터를 생성하는 응용 분야가 증가함에 따라 스트리밍 데이터의 처리 요구 또한 높아지고 있다. 특히 웹상의 데이터 교환의 표준으로 XML 이 부각된 이래 XML 스트리밍 데이터에 대한 처리 시스템의 연구가 활발히 수행되고 있다.

스트리밍 데이터의 특징은 실시간 처리를 요구하며, 연속적이고, 데이터 크기의 한계가 없다는 것이다. 이런 연속적이고 빠르게 발생하는 스트리밍 데이터 처리를 위해 질의 결과를 캐싱하여 성능 향상을 기할 수 있다. 특히, 스트림에 대한 시구간 질의가 주어질 경우 끊임없이 발생하는 XML 스트리밍 데이터를 저장하기 전에 각각의 질의에 대한 결과 값을 캐싱하여 해당 질의가 호출 되었을 때 해당 질의의 캐싱된 결과 값들에 시구간만을 적용하여 최종 결과를 산출함으로써 빠른 응답시간을 보장할 수 있다. 또한, 질의 결과의 캐싱은 백그라운드 프로세싱으로 수행할 수 있다.

본 논문에서는 XML 스트리밍 데이터에 대한 시구간 질의 처리 시스템 모델을 제시한다. 스트리밍 데이터 모델로 웹상의 데이터 교환 표준으로 자리잡은 XML 을 사용하였고 질의 모델로는 XML 질의의 유력한 표준으로 부각된 XQuery 에 시구간을 덧붙인 형태의 Continuous Query(CQ)를 시구간 질의 모델로 사용하였다. CQ 는 기존의 RDBMS 기반의 Standing Query 와는 많은 차이점이 있다. Standing Query 는 현재 저장된 데이터에 대한 One-Time Query 인 반면 본 논문의 시스템에서 지원하는 질의 모델은 일정 기간동안 지속적으로 사용될 수 있는 CQ 를 뜻한다. 시구간은 질의 대상인 스트리밍 데이터가 발생한 시간을 구간으로 정의함으로써 원하는 시간 구간의 데이터만을 질의 대상으로 선정한다. 스트리밍 데이터에 대한 CQ 처리는 많은 스트리밍 시스템에서 활발히 연구 되었다. 이를 위한 여러 가지 접근방법이 여러 대학과 연구기관에서 연구되었는데 본 논문에서는 이를 위한 방법으로 XML 엘리먼트를 라우팅하는 방법을 제안한다. 끊임없이 발생하는 연속적인 XML 스트리밍 데이

† 본 논문은 한국과학재단 특정기초연구사업(R01-2003-000-10395-0) 지원으로 수행 되었음.

터를 SAX Parser 를 이용하여 파싱하고 이때 발생한 이벤트를 이용하여 XML 엘리먼트를 각각의 Query Structure (QS)로 라우팅하여 현재 등록된 질의에 해당하는 엘리먼트들을 결과로 캐싱하는 방법이다.

2 절에서는 스트리밍 데이터 시스템에 관한 관련연구를 기술한다. 3 절에서는 본 논문에서 제안하는 XSDMS(XML Streaming Data Management System)의 전체적인 프레임워크에 대해서 기술한다. 4 절에서는 결론을 맺고 향후 연구 내용을 기술한다.

2. 관련 연구

기존의 스트리밍 데이터 처리 시스템의 데이터 모델은 대부분 관계 데이터 모델을 기반으로 한 것이다[1-5]. 본 논문에서는 이를 기반으로 관계형 데이터를 XML 데이터 모델로 확장하여 질의 스트림에 적용시키는 데 있어 해결해야 할 기술 요소들을 파악하고 시스템을 모델링한다.

최근 스트리밍 데이터를 처리하기 위한 다양한 CQ Engine (Continuous Query Engine)이 연구되고 있다. 그 중 본 연구의 기반이 되는 시스템들에는 Telegraph CQ, Stream, Tukwila, Niagara CQ 등이 있다[1-6]. Telegraph CQ 는 스트리밍 데이터 모델로는 관계형 데이터 모델을 사용하였고 질의 모델로는 SQL 에 시구간을 추가하여 확장한 형태의 질의 모델을 사용하였다[1-2]. 또한 데이터와 질의를 모두 스트리밍 데이터의 형태로 처리하기 위한 Data Structure 인 SteMs(State Module)를 제안하고 있다. 각각의 SteMs 에 데이터 또는 질의 스트림을 저장하기 위해 연속적으로 들어오는 관계형 스트리밍 데이터를 라우팅하기 위한 Eddy System 또한 Telegraph CQ 의 핵심 기술이다[6].

여러 시스템들 중 Tukwila 는 XML 스트리밍 데이터를 기반으로하는 스트리밍 데이터 처리 시스템이다[7]. Tukwila 는 Automata 를 이용한 X-Scan operator 를 이용하여 on-line 질의를 처리한다. Telegraph 와 Tukwila 는 모두 Memory based system 으로 과거의 데이터에 대한 질의 처리를 지원하지 않는다. Tukwila 의 경우는 과거의 질의 처리는 불가능하며, Telegraph 의 경우는 최근의 데이터에 대한 질의 처리만 가능하다.

본 논문에서 제안하는 시스템과 기존의 시스템의 가장 큰 차이점은 첫째, 디스크 기반의 시스템이라는 점이다. 기존의 시스템들은 메모리 기반으로 스트리밍 데이터를 일정 기간 동안 메모리에 저장하고 버리기 때문에 과거 데이터에 대한 질의가 용이하지 않으나 본 논문에서 제안하는 시스템은 스트리밍 데이터를 디스크에 저장함으로써 과거 데이터에 대한 질의 또한 지원한다. 둘째, XML 을 데이터 모델로 사용한다. 기존의 시스템들이 주로 관계형 데이터를 데이터 모델로 사용한 데 반해 본 시스템은 웹 교환의 표준으로 자리잡은 XML 을 데이터 모델로 사용하여 모니터링, 센서 네트워크등 여러 가지 XML 데이터 처리를 가능하게 한다. 마지막으로 질의 처리의 성능 향상을 기하기 위해 결과 값을 미리 실체화 한다. 이때 SAX 파서를 이용한 엘리먼트 라우팅 기법을 사용한다[8-9]. 질의 처리 기법은 연속적으로 발생하는 스트리밍 데이터를 각각의 미리 정의된 질의(pre-defined query)에 적용해 미리 질의 결과 값을 실체화 하여 캐싱하는 기법이다. 이 때 질의 결과 값은 시구간이 적용되지 않은 상태로 유지되고 해당 질의에 대한 호출이 발생할 때에 현재 시스템 타임 스탬프를 적용하여 시구간을 적용한 적절한 질의 결과 값을 반환한다. 즉, 질의에 대한 결과 값을 메모리에 실체화하여 캐싱함으로써 시스템의 성능향상을 기한다.

3. XSDMS(XML Streaming Data Management System) 모델

3.1 개요

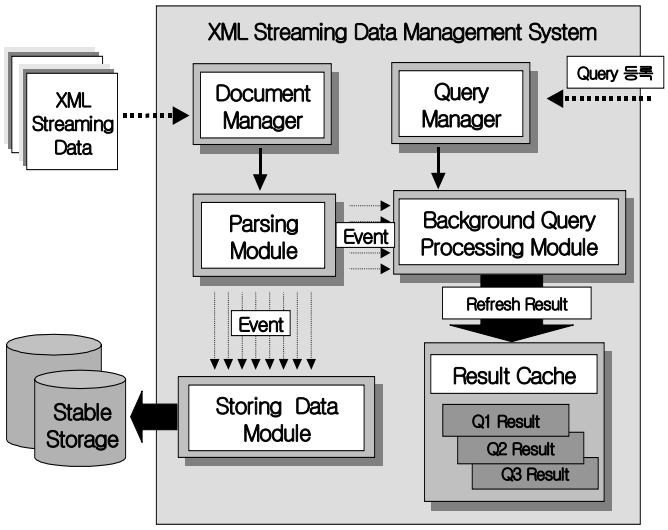

그림 1. XSDMS

XSDMS 는 XML Streaming Data Management System 으로서 XML 스트리밍 데이터를 실시간으로 처리하는 시스템이다. <그림 1>과 같이 XML 스트리밍 데이터 관리 시스템은 Document Manager(DM)를 통해 XML 문서를 스트리밍 데이터로 입력받고 Query Manager(QM)를 통해 질의를 등록 받는다. 이때 DM 은 각 문서의 입력 순서와 입력 시각을 표시하기 위해 각 문서 마다 시스템 타임 스탬프를 엘리먼트로 부여한다. 시스템 타임 스탬프가 부여된 문서는 파싱 모듈로 넘겨지고 파싱 모듈에서 SAX 파서를 이용해 스트리밍 데이터를 파싱하여 이벤트를 생성한다.

파싱 모듈에는 두개의 큐가 존재하는데 하나는 백그라운드 프로세싱을 위한 큐이고 다른 하나는 XML 스트리밍 데이터를 RDB 에 백업하기 위한 큐이다. SAX 파서를 통해 생성된 이벤트는 백그라운드 쿼리 프로세싱을 위한 큐에 우선 저장되고 백그라운드 쿼리 프로세싱이 끝난 이벤트는 데이터 저장을 위한 큐로 옮겨진다. 큐에 저장된 이벤트를 이용하여 백그라운드 쿼리 프로세싱 모듈에서는 XML 엘리먼트를 QM 에서 관리하는 각각의 Query Structure(QS)로 라우팅하여 시

구간을 적용하지 않은 XQuery 를 만족하는 결과 값을 산출한다. 이 때 생성된 결과 값은 질의 호출 시에 빠른 응답속도를 위해서 메모리에 캐쉉된다.

Query Manager 는 미리 등록된 질의 (pre-defined query)의 개수 만큼의 QS 를 생성하고 이를 유지 관리하는 역할을 한다. 백그라운드 프로세싱을 거친 이벤트는 데이터 저장 큐에 저장되었다가 스트리밍 데이터에 대한 처리 요구가 낮을 때에 저장 모듈에서 RDB 에 [10]의 XRel 시스템과 유사한 4 Table 에 저장한다. XML 문서를 4 Table 에 분할 저장하는 것은 본 논문의 주제가 아니므로 생략한다.

3.2 질의 모델

본 논문의 질의 모델은 Continuous Query(CQ)를 사용한다. CQ 는 일정 생명 주기동안 연속해서 질의를 수행할 수 있는 질의를 뜻한다. 또한 CQ 는 Pre-defined Query 로서 미리 정의하여 시스템에 등록한 후 사용한다. CQ 는 landmark 나 sliding window 와 같은 구간 질의를 지원한다. 본 논문에서 제안하는 CQ 모델은 <그림 2>와 같이 XQuery 에 시구간을 추가한 형태이다.

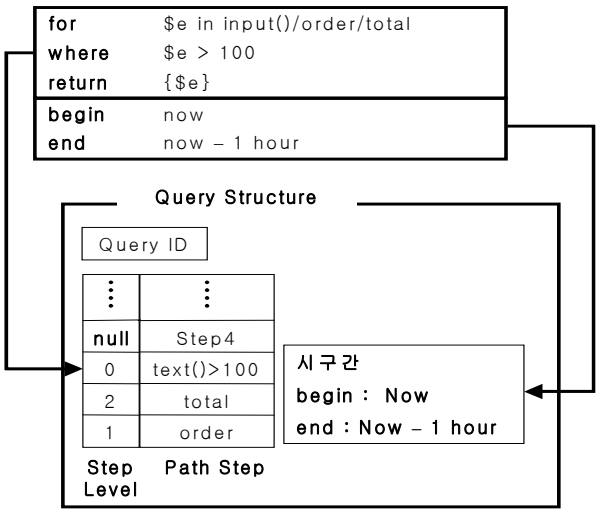

그림 2. Query Structure

시구간 질의를 등록하면 <그림 2>와 같이 Query Structure(QS)에 시구간 질의가 Path 정보에 따라 적절하게 분할되어 저장된다. QS 는 Path Step, Step Level, 시구간을 이용하여 CQ 를 나타낸다. Path Step 은 XQuery 의 path 정보를 Step 별로 나누어 저장한다. 이것은 XML 문서 내의 엘리먼트의 Path 정보와 매칭되어 각각의 XML 스트리밍 데이터가 질의를 만족하는가를 쉽게 비교할 수 있게 한다. Step Level 은 각각의 Path Step 이 문서 내에서 root 로부터의 몇 번째 자식인지를 나타낸다. 단 텍스트 엘리먼트는 0 으로 표시한다. Step Level 은 3.3 절의 백그라운드 쿼리 프로세싱에서 라우팅 마스크를 셋팅하는 데 이용된다. 마지막으로 시구간은 별도로 저장하여 질의가 호출될 때 시구간을 쉽게 적용할 수 있도록 한다. 시구간은 고정된 타임 스탬프를 정의할 경우 기존의 Standing Query 와 같이 등록되며 현재 시간에 따라 유동적인 시구간을 정의할 경우 Landmark 또는 Sliding Window Query 와 같은 CQ 로 등록할 수 있다. CQ 는 호출된 시점의 타임 스탬프를 적용하여 현재 시간이 반영된 시구간을 갖게 되므로 호출 시기에 따라 매번 다른 데이터를 대상으로 질의가 수행된다.

QS 는 단순히 Query 를 저장하는 자료 구조일 뿐 아니라 XML 문서의 엘리먼트와 질의의 매칭 여부를 쉽게 알 수 있도록 질의를 유지 관리하는 자료 구조이다. 미리 정의된 질의는 각각 QS 에 저장되고 이를 이용하여 하나의 XML 엘리먼트가 현재 등록된 모든 질의의 결과 값으로 타당한지를 하나씩 체크할 수 있다.

3.3 Background Query Processing

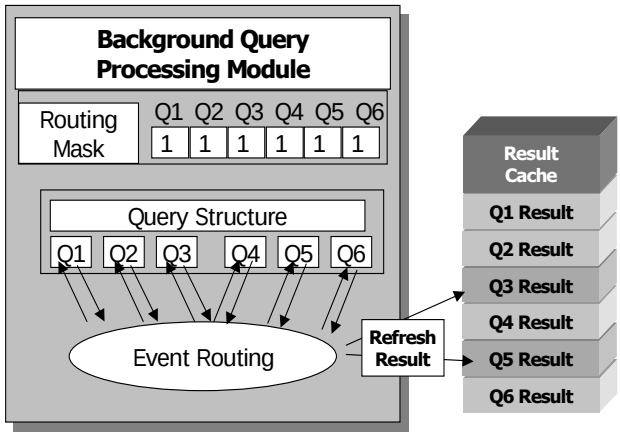

그림 3. Back ground Query Processing

<그림 3>은 Background Query Processing Module 을 나타낸다. 쿼리 처리 모듈은 연속적으로 발생하는 XML 스트리밍 데이터를 원활하게 처리하기 위해서 백그라운드 프로세싱으로 이루어진다. 파싱 모듈에서 생성된 이벤트를 이용해 엘리먼트를 QS 로 라우팅하여 해당 엘리먼트가 각각의 질의 결과 값으로 타당한지를 검사한다. 타당할 경우 해당 엘리먼트를 문서의 타임 스탬프로 묶어 결과 값으로 캐쉉한다. 이 때 불필요한 엘리먼트의 라우팅을 막기 위해 라우팅 마스크를 사용한다.

라우팅 마스크는 등록된 질의의 수 만큼의 마스크를 유지한다. 각각의 마스크는 다음 질의가 라우팅을 기다리고 있는 엘리먼트의 Step Level 값을 나타낸다. 이를 이용하여 Step Level 과 현재 라우팅하고자 하는 엘리먼트의 Level 이 같지 않을 경우 불필요한 라우팅을 피하여 성능향상을 기할 수 있다. 라우팅 마스크는 Start Document 이벤트가 들어오면 모두 1 로 셋팅되어 첫번째 자식을 기다린다. QS 의 모든 step 이 매칭 되었을 경우는 질의를 만족하는 엘리먼트이므로 결과 엘리먼트로 캐쉉한다. 질의를 만족하는 결과 값을 찾았을 경우에는 더 이상의 엘리먼트 라우팅이 필요하지 않으므로 라우팅 마스크를 NULL 로 셋팅하고 다른 이벤트들을 필터링하여 불필요한 라우팅을 피한다.

3.4 질의 결과의 점진적 갱신

새로 들어온 문서에서 각각의 질의 결과에 해당하는 엘리먼트들은 캐싱된 결과 문서의 마지막에 삽입된다. 이때 각각의 엘리먼트 들은 Document Manager가 부여한 문서의 타임 스탬프를 속성으로 갖는 결과 엘리먼트의 자식 엘리먼트로 추가된다. 이렇게 캐싱된 결과 값은 질의가 호출될 때 현재 시스템 타임 스탬프를 시구간에 적용하여 캐싱된 엘리먼트들 중에 시구간 안에 해당하는 엘리먼트만을 결과 값으로 반환한다. <그림 4>와 같이 시구간을 벗어나는 엘리먼트들은 모두 삭제되고 다음 번 질의의 호출을 위해 반환된 결과 값을 캐싱하여 결과 값으로 유지한다. 결

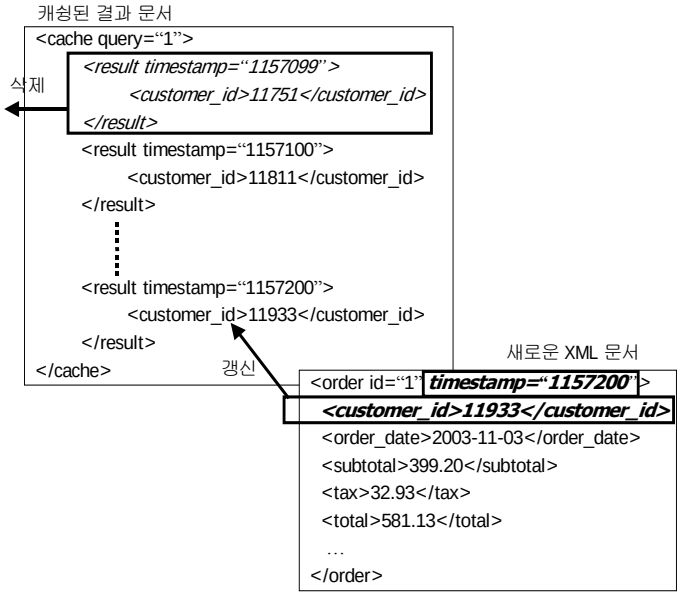

그림 4. 질의 결과의 점진적 갱신

과 값의 캐싱 기법은 질의 호출 시 별도의 결과 값을 산출하는 오버헤드 없이 시구간만을 적용하여 결과를 산출할 수 있게 해주어 빠른 응답시간을 보장한다.

4. 결론

본 논문에서는 XML 스트리밍 데이터를 처리하기 위한 XML 스트리밍 데이터 처리 시스템인 XSDMS의 시스템 모델을 제안하였다. 또한 스트리밍 데이터에 대한 시구간 질의를 처리하기 위한 기법으로 SAX 파서를 이용한 엘리먼트 라우팅을 사용하였고 질의 처리의 성능 향상을 위하여 질의 결과의 실체화를 이용한 캐싱 기법을 제안하였다.

향후 연구 과제는 다음과 같다. XML 스트리밍 데이터에 대한 시구간 처리 시스템의 구현과 구현에 따른 성능 평가를 수행하는 것이다. 구현은 본 논문에서 제안한 시스템 모델을 바탕으로 수행하며 성능 평가는 등록된 질의의 개수 당 질의 처리 응답 속도와 질의의 복잡도에 따른 질의 응답 속도에 대하여 평가한다. 그리고 질의 처리에 있어서 질의 결과의 캐싱이 성능 향상에 미치는 영향을 알아보기 위해 결과 값을 백그라운드 프로세싱으로 산출하여 유지하는 경우와 그렇지 않은 경우의 성능 비교가 필요하다.

참고 문헌

[1] S. Madden, M. Shah, J. Hellerstein and V. Raman "Continuously Adaptive Continuous Queries over Streams," Proc. ACM SIGMOD Int'l Conf. on Management of Data, 2002, pp. 49-60.

[2] S. Chandrasekaran and M. J. Franklin, "Streaming Queries Over Streaming Data," Proc. of Int'l Conf. on VLDB, 2002.

[3] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. widom, "Models and Issues in Data Stream Systems," Proc. of Symp. on PODS, 2002.

[4] D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. lee, G. Seidman., M. Stonebraker, N. Tatbul and S. zdonik, "Monitoring Streams - A New Class of Data Management Applications," Proc. of Int'l Conf. on VLDB, 2002.

[5] J. Chen, D. J. DeWitt, F. Tian, Y. Wang, "NiagraCQ: A Scalable Continuous query system for internet database," Proc. ACM SIGMOD Int'l Conf. on Management of Data, 2002, pp. 379-390.

[6] R. Avnur and J. Hellerstein, "Eddies: Continuously Adaptive Query Processing," Proc. ACM SIGMOD Int'l Conf. on Management of Data, 2000, pp. 261-272.

[7] Z. G. Ives, D. Florescu, M. Friedman, A. Levy and D. S. Weld, "An Adaptive Query Execution System for Data Integration," Proc. ACM SIGMOD Int'l Conf. on Management of Data, 1999.

[8] A. K. Gupta and D. Suciu, "Stream Processing of XPath Queries with Predicates," Proc. ACM SIGMOD Int'l Conf. on Management of Data, 2003, pp. 419-430.

[9] F. Peng and S. S. Chawathe, "XPath Queries on Streaming Data," Proc. ACM SIGMOD Int'l Conf. on Management of Data, 2003, pp. 431-442.

[10] M. Yoshikawa, T. Amagasa, T. Shimura and S. Uemura "XRel: A Path-Based Approach to Storage and Retrieval of XML Documents Using Relational Databases," ACM TOIT, 2001, Vol.1, No.1, pp. 110-141.