韓國 컴퓨터情報學會 2008 제38차 하계학술발표논문집
第16卷 第1號, 2008. 06.

네트워크 모니터링을 위한 OLAP 구현

양 우 석*, 이 원 석*

OLAP Implementation for Network Monitoring

Woo-Sock Yang*, Won-Suk Lee*

요 약

데이터스트림 환경에서 무한히 연속적으로 생성되는 데이터를 처리하고 분석하는 방법에 관한 많은 연구가 진행 중이다. 본 논문은 데이터스트림의 한 예인 네트워크 트래픽을 모니터링하기 위한 OLAP 구현에 대하여 기술한다. 제안하는 OLAP 시스템은 기존의 네트워크 모니터링 툴이 제공하지 못했던 다양한 연산을 지원하여 유연한 분석을 가능하게 하며, 정적인 데이터를 처리하는 데이터웨어하우스에서만 적용되던 OLAP을 데이터스트림 환경에 적용할 수 있게 한다.

▸ Keyword : 데이터스트림(Data Stream), OLAP, 네트워크 모니터링(Network Monitoring)

• 제1저자 : 양우석
* 연세대학교 컴퓨터과학과
※이 논문은 교육과학기술부, 지식경제부, 노동부의 출연금 및 보조금으로 수행한 최우수실험실지원사업과 2008년도 정부(교육과학기술부)의 재원으로 한국과학재단의 국가지정연구실사업(No.R0A-2006-000-10225-0)으로 수행된 연구입니다.

Ⅰ. 서 론

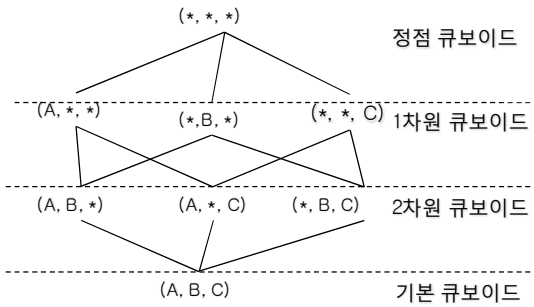
데이터의 양이 급격히 증가함에 따라 데이터로부터 유용한 정보를 관리 및 추출하기 위한 데이터웨어하우징 및 분석 기법에 대한 요구가 증대되고 있다. 특히 최근 유비쿼터스 시대로 진입함에 따라, 데이터스트림 환경에 대한 연구가 주목을 받고 있다. 데이터스트림은 기존의 데이터웨어하우스에 저장되었던 데이터들과는 그 특성이 매우 다르기 때문에 이를 처리하기 위해 새로운 기법들이 요구되고 있다. 데이터스트림은 무한히 연속적으로 생성되며, 이 데이터들의 처리 결과는 사용자에게 즉시 제공되어야 한다. 따라서 주기적으로 업데이트를 실행하는 데이터웨어하우스와 기존의 DBMS로는 데이터스트림을 처리하는 데에 한계가 있다.

데이터스트림의 대표적인 예로는 네트워크 트래픽을 들 수 있다. 기업은 자사 사이트의 네트워크 트래픽을 관찰하여 소비자의 방문 패턴을 분석하고 그 결과를 기업 개선 방향에 반영할 수 있다. 또한 네트워크 트래픽 관찰은 안정적인 서비스를 위해서 중요하며 침입탐지와 같은 보안영역에서도 유용하게 활용될 수 있다. 본 논문에서는 데이터스트림 환경에서의 네트워크 데이터를 처리 및 분석할 수 있는 OLAP의 구현에 대해 논의하고자 한다.

논문의 구성은 다음과 같다. 2장에서는 OLAP과 관련된 기존의 연구 및 개념들을 살펴본다. 3장에서는 네트워크 모니터링에서 OLAP이 적용되어야 하는 필요성에 대해 설명한다. 4장에서 제안하는 시스템을 적용하여 네트워크 모니터링을 수행하는 방법을 설명하고 결과를 분석한다. 5장에서는 결론을 기술한다.

Ⅱ. OLAP과 스트림큐브

OLAP은 데이터웨어하우스에 저장된 데이터의 요약된 정보를 사용자에게 효과적으로 보여주기 위한 도구이다.[1,2] 데이터웨어하우스의 데이터는 일반적으로 스타스키마 구조를 통해 관리되며, 스타스키마는 하나의 사실 테이블과 이와 연결된 다수의 차원 테이블로 이루어진다. 사실 테이블에는 가장 세밀한 상세도를 가지는 데이터들이 저장되며, 이 데이터들은 여러 차원들에 대하여 SUM, AVG, MIN, MAX등의 집계함수를 이용하여 요약될 수 있다. 일반적으로 OLAP은 내부적으로 데이터큐브 구조를 사용하여 구축된다.[3] 그림 1에서 A, B, C 세 개의 차원 테이블을 가진 데이터큐브의 예를 나타내었다. 각 정점은 요약된 데이터를 나타내며 이를 큐보이드라 부른다. 정점 큐보이드는 세 차원에 대하여 요약한 경우이고, 2차원 큐보이드의 세 정점은 각각 C, B, A에 대하여 집계한 결과를 표시한다. 차원의 수가 많고 각 차원이 가질 수 있는 값의 범위가 많아질수록 데이터큐브가 필요로 하는 저장 공간이 커지고 사용자 질의 응답시간도 늘어나게 된다. 특히 데이터를 물리적인 디스크에 저장함에 따라 많은 입출력 연산이 요구되므로 실시간으로 운영되어야 하는 데이터스트림환경에는 부적합하다.

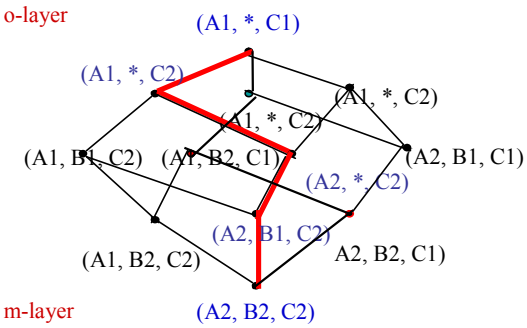


<그림 1> 데이터 큐브

데이터스트림 환경에서의 OLAP에 대한 연구로는 스트림 큐브가 있다.[4] 그림 2는 스트림 큐브 구조를 도시한 것이다. 데이터스트림 환경에서는 빠른 응답을 위해서 모든 연산이 메모리 공간에서 처리되어야 한다. 이를 위하여 스트림 큐브는 다음과 같은 특징을 가지고 있다.

i) 최근의 데이터는 그대로, 과거의 데이터는 요약하여 저장한다.
ii) 관찰 계층(o-layer)과 최소 관심 계층(m-layer)을 지정하여 관심 영역을 벗어나는 큐보이드는 저장하지 않는다.
iii) 그림 2의 굵게 표시된 경로와 같이, 통계적으로 가장 많은 질의가 발생하는 경로에 대해서만 상세 정보를 저장하게 된다.

이와 같은 방식으로 주어진 메모리 내에서 사용자가 필요로 하는 정보를 즉시 제공할 수 있다.



<그림 2> 스트림 큐브

III. 네트워크 모니터링

네트워크 전송량이 급증하면서 네트워크 데이터 관리의 중요성이 부각되었고 모니터링의 역할과 비중 또한 커져왔다. 네트워크 모니터링은 사용자에게 분석 및 관리에 필요한 다양한 정보들을 제공한다. 예를 들어, 전체 트래픽의 량을 관찰하게 되면 안정적인 서비스를 위하여 트래픽의 량을 얼마나 조절해야 하는지를 결정할 수 있다. 또한 과부하를 유발하는 특정 IP를 검출하거나, 전체 트래픽 중 정상적인 트래픽의 비율, 네트워크 통해 지나가는 입출력 패킷 수 등의 통계를 제공할 수 있다. 이러한 정보들은 서비스의 개선 및 보안 강화의 측면에서 중요한 의미를 가진다.

현재 네트워크 모니터링을 위한 다양한 기법과 도구가 개발되어왔으나 대부분 처리할 수 있는 기능에 한계가 있다. 보다 의미있는 정보를 얻기 위하여 대용량의 네트워크 트래픽 데이터를 대상으로 마이닝을 수행한 연구들[6, 7]이 있으나 이 역시 데이터스트림 환경에 적용하기에는 한계를 갖고 있다. 데이터 마이닝 같은 고차원적인 분석 결과를 얻기 위해서는 개개의 트래픽 정보를 데이터베이스 관리 시스템(DBMS)에 저장한 뒤 요청 시점에 다시 읽어들여 처리하는 방식을 취하기 때문이다. 네트워크 트래픽은 무한히 연속해서 들어오는 데이터로서 데이터스트림의 전형적인 예이다. 따라서 이를 모니터링하려는 관찰자는 입력에 따른 분석 결과를 즉시 받기를 원하게 된다. 그러나 DBMS의 방식은 응답시간이 오래 걸린다. 또한 데이터의 추가적인 입력 및 갱신이 일어나게 되므로 이에 따르는 부하 역시 무시할 수 없다. 한편 모든 입력 패킷의 정보를 디스크에 저장하게 되면서 엄청난 용량의 디스크 공간을 필요로 하게 된다.

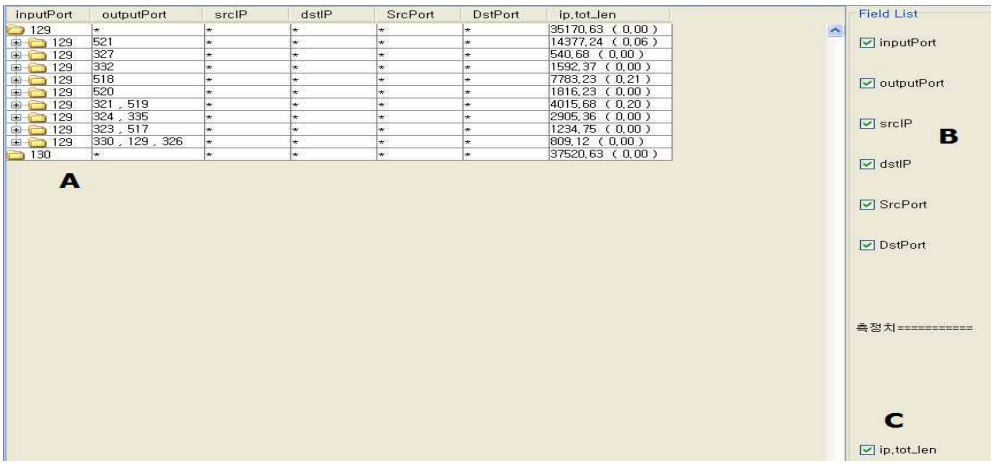
대부분 제공하는 기능이 단순히 현재 상황의 모니터링에 국한됨에 따라 다양한 측면의 분석을 수행하는 데에 어려움이 있다. 네트워크 모니터링에 OLAP을 적용하면 더욱 다양하고 유연한 분석이 가능하다. OLAP 시스템에서는 Roll-up, Drill-down, Slice, Dice, Pivot등의 연산이 사용된다.[5] Roll-up 연산은 차원의 계층 구조를 한 단계 상승시키거나 차원을 감소시키면서 데이터 큐브의 집계를 수행한다. Drill-down 연산은 Roll-up 연산의 역으로, 데이터의 상세 레벨을 탐색하는 것이다. Slice 연산은 주어진 데이터 큐브에서 한 개의 차원을 선택하는 것으로, 결과는 서브 큐브가 된다. Dice 연산은 두 개 이상의 차원을 선택하면서 서브 큐브를 생성하는 것이고, Pivot 연산은 데이터의 축을 회전하여 다른 관점에서 볼 수 있게 하는 시각화를 위한 연산이다. 이러한 연산을 적용함으로써 네트워크 데이터에 대한 다양한 분석 결과를 도출 및 제공할 수 있다.

IV. OLAP의 구현

본 구현 시스템에서는 스위치 단에서 sFlow[8] 형태의 네트워크 데이터를 수집하기 위하여 sflowtool[9]을 사용한다. sFlow는 샘플링 기반으로 작동하기 때문에 데이터 전체를 수집하는 것이 거의 불가능한 초고속 인터넷 환경에 적합하며 네트워크 플로우의 상세 정보들을 각각의 필드 형태로 제공한다. 표1은 네트워크 플로우 데이터의 한 예이다. sFlow는 이

<표 1> 네트워크 플로우 데이터의 예

src.IP	src.Port	dst.IP	dst.Port	InputPort	OutputPort	ip.tot_len
192.168.1.55	24452	yahoo.com	WWW	129	521	46
192.168.1.66	3611	213.1.2.1	12400	128	520	52
yahoo.com	WWW	192.168.1.55	3312	129	521	64
192.168.1.66	0	213.1.2.1	0	128	520	32



<그림 3> OLAP 메인 화면

inputPort	outputPort	srcIP	dstIP	SrcPort	ip.tot_len
130	*	*	*	*	29844.63 (...
130	518	*	*	*	4757.73 (...
130	520 , 521	*	*	*	2024.48 (...
130	520 , 521	211.55.94.1...	*	*	6.46 (0.05...
130	520 , 521	61.255.0.88...	*	*	6.46 (0.05...
130	520 , 521	58.143.59.1...	*	*	0.89 (0.01...
130	520 , 521	221.159.215...	*	*	8.07 (0.15...
130	520 , 521	122.128.196...	*	*	0.89 (0.01...
130	520 , 521	211.212.154...	*	*	0.89 (0.01...
130	520 , 521	211.207.224...	*	*	42.82 (0.3...
130	520 , 521	59.186.44.4...	*	*	56.00 (0.2...
130	520 , 521	222.100.157...	*	*	1.89 (0.60...
130	520 , 521	218.55.141....	*	*	146.38 (1....
130	520 , 521	221.160.112...	*	*	35.19 (0.0...
130	327 , 324 , ...	*	*	*	1395.68 (...
130	332 , 323 , ...	*	*	*	1099.87 (...
129	*	*	*	*	27260.63 (...

<그림 4> OLAP의 실행 예

보다 많은 필드를 제공하지만, 본 논문에서는 표1과 같이 7개의 필드만을 사용한다. ip.tot_len은 하나의 패킷 데이터의 크기를 의미하며 이것을 측정치로 사용하였다. 나머지 6개의 필드들은 차원 속성으로 사용된다. src.IP와 src.Port 는 각 제안하는 OLAP 시스템은 그림 3과 같이 구성된다. 그림 3의 B에서 각 차원을 선택하고 C에서 측정치 값을 선택한다. 이러한 선택 사항들이 A에서 각 컬럼 별로 나열된다. 각 차원들은 내부적으로는 트리 구조로 관리된다. 예를 들어 그림 3에서 inputPort는 트리의 첫 번째 레벨을 나타내며 그 옆의 outputPort는 두 번째 레벨을 나타낸다. 그림 3의 A에서 알 수 있듯이, 현재 입력된 데이터스트림의 장치 입력 포트 번호(InputPort)는 129와 130이다. 표시된 화면에서 노드 129를 클릭하면 그 하위 레벨인 outputPort 차원의 노드들이 나타난다. '*' 는 해당 차원에 대하여 요약되었음을 나타낸다. 그림 3의 A의 마지막 줄은 지금까지 들어온 전체 sFlow 데이터에 대하여 inputPort가 130인 경우의 ip.tot_len 값이 37520.63이라는 정보를 보여주고 있다.

데이터를 처리하는 서버 측의 내부 구조도 기존 OLAP에서 사용하는 큐브 형태가 아닌 트리 형태의 데이터 구조를 유지한다. 사용자 인터페이스에서 보이는 노드의 구조는 내부 트리의 구조와 유사하다. 이 구조는 어떤 노드가 많이 나왔을 경우 이를 개별 노드로 분리하는 특성을 갖는다. 상대적으로 적게 나왔을 경우는 합병해서 관리한다. 합병에서 관리하는 것의 가장 큰 이점은 노드를 적게 만듦으로써 메모리를 절약하게 해주기 때문이다. 또한 자주 들어온 값 일수록 개별 노드로 분리되기 때문에 OLAP이 표시해주는 정보에 대한 사용자의 이해를 높일 수 있다. 위의 그림 3에서 총 14개의 outputPort 값이 나왔다는 것을 알 수 있다. 이중에서 상위 5개의 값은 개별 노드로 분리되었으며, 두 개의 값을 갖는 노드가 세 개, 세 개의 값을 갖는 노드가 하나 있음을 볼 수 있다. 이러한 분리의 정도는 데이터스트림의 입력에 따라서 동적으로 바뀐다. 이를 통해 사용자는 어떤 값이 현재 빈발하게

출현하고 있는 지를 알 수 있다. 현재 outputPort에서 개별 노드로 분리된 다섯 개의 포트 값 521, 327, 332, 518, 520은 이러한 출력 포트로 값이 빈발하게 나갔다는 것을 알려준다.

그림 4에서는 연산의 수행 과정을 보이고 있다. Drill-down과 Roll-up은 첫 번째 노드를 클릭하면서 수행할 수 있다. 그림 4는 inputPort가 130인 상태에서 outputPort가 520이거나 outputPort가 521인 경우로 Drill-down한 예이다. Drill-down한 상태에서 반대로 outputPort가 520이거나 521인 노드를 클릭하여 트리를 수축시켰을 경우에는 Roll-up이 수행되는 것이다. 또한 간단한 통계 연산을 수행하여 이례적인 상황을 표시해 준다. 그림의 ip.tot_len 필드에서 음영으로 표시된 부분은 inputPort가 130이고 outputPort가 520 또는 521인 집합 중에서 가장 빈발하게 입력된 srcIP를 강조해서 표시한 것이다. 이는 일반적으로 OLAP에서 연산의 조합에 대하여 수많은 경우의 수가 존재하므로, 사용자가 OLAP을 용이하게 탐색할 수 있도록 돕는 역할을 한다.

이와 같은 OLAP은 다음과 같은 이유에서 네트워크 데이터를 비롯한 데이터스트림 환경에서 유용하게 쓰일 수 있다. 첫째, 유연한 분석을 가능하게 한다. 제안하는 OLAP 시스템은 네트워크 모니터링이 제공하지 못하는 기능을 지원 한다. 즉, 기존의 모니터링 툴을 통해 사용자는 프로그램이 나타내는 정해진 형식의 정보만을 살펴볼 수 있지만, 제안하는 OLAP 시스템은 현재의 입력 데이터에 대하여 사용자가 요구하는 다양한 연산들을 수행할 수 있다. 둘째, 데이터스트림이 입력됨에 따라 즉시 처리 및 분석 결과를 갱신시킨다. 이에 따라 사용자는 현재의 입력 데이터에 따른 변화 정도를 빠르게 인식할 수 있고, 현재 주목해서 보아야 할 부분을 OLAP을 통해서 탐색할 수 있다. 셋째, 제안한 시스템은 다양한 데이터스트림 환경에서의 활용 가능성을 가지고 있다. 입력 데이터의 형식 정보를 각 응용 환경 별로 변환함으로써, 네트워크 데이터뿐만 아니라 센서 데이터, 증권 정보, 날씨, 콜 센터, 웹 페이지 클릭 로그 등의 다양한 데이터스트림에서도 확장 및 적용이 가능하다.

V. 결론

네트워크 데이터와 같은 데이터스트림 환경에서의 데이터 분석에 관한 많은 연구가 진행 중에 있다. 본 논문에서는 기존의 정적인 데이터 관리를 위한 데이터웨어하우스에서의 OLAP을 데이터스트림 환경에서의 네트워크 모니터링에 적용하는 방법에 대해 기술하고, 이를 구현한 시스템의 특성을 설명하였다. 제안하는 OLAP 시스템은 무한히 연속적으로 들어오는 네트워크 데이터를 주어진 메모리 공간 내에서 처리함으로써 사용자의 질의에 즉시 응답할 수 있다. 또한 기존의 OLAP이 지원하는 다양한 연산들을 제공함으로써 사용자의 분석 활용도를 높였다.

참고문헌

[1] The OLAP Council., “MD-API the OLAP Application Program Interface Version 0.5 Specification”, 1996

[2] Rakesh Agrawal, Ashish Gupta, Sunita Sarawagi, “Modeling multidimensional database. In Proc.”, the 13th Intl conference on Data Engineering, Birmingham, U.K., 1997

[3] S. Chaudhuri and U. Dayal, “An overview of data warehousing and OLAP technology,” SIGMOD Record, vol. 26, pp. 65 - 74, 1997.

[4] Jiawei Han, Yixin Chen, Guozhu Dong, Jian Pei, Benjamin W. Wah, Jianyong Wang, Y. Dora Cai, “Stream Cube: An Architecture for Multi-Dimensional Analysis of Data Streams.”, Distributed and Parallel Databases 18(2): 173-197 (2005)

[5] J. Gray, S.Chaudhuri, A.Bosworth, A.Layman, D.Reichart, M.Venkatrao, “Data Cube: A Relational Aggregation Operator Generalizaing Group- By, Cross-Tab, and Sub-Totals”, Data Mining and Knowledge Discovery 1, pp. 29-53, 1997.

[6] Y. Hu and B. Panda, "A Data Mining Approach for Database Intrusion Detection," In Proceedings of the 2004 ACM Symposium on Applied Computing, pp. 711-716, 2004.

[7] J. Luo and S. Bridges, "Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection," International Journal of Intelligent Systems, Vol. 15, No. 8, pp. 687-704, 2000.

[8] http://www.sflow.org

[9] http://www.inmon.com/technology/sflowTools.php