데이터 스트림 처리에 관한 연구 동향[1)]

안동찬[O] 박석
서강대학교 컴퓨터학과
{channy[O], spark}@dblab.sogang.ac.kr

Research Directions for Data Stream Processing

Dong-Chan An[O] Seog Park
Department of Computer Science, Sogang University

요 약

데이터 스트림 처리에 관한 연구들은 최근 컴퓨터 과학 분야에서 가장 많은 관심을 가지고 있고 흥미 있는 이슈 중 하나이다. 이러한 경향은 다양한 어플리케이션의 출현으로 질의 처리에 대한 효율적인 방법과 새로운 데이터 스트림의 모델을 필요로 하기 때문일 것 이다. 본 논문은 그 중에서도 데이터 스트림 분야의 가장 중요한 부분으로 생각되는 스케줄링, 적절한 질의 처리, 부하 분산, 근사화, 분산 데이터 모니터링에 대한 연구 분석을 도모하였다.

1. 서 론

전통적인 데이터베이스 관리시스템은 영속적인(persistent) 데이터와 일시적인(transient) 질의에 대한 응답을 다루기 위해 설계되었다. 그러나, 최근 네트워크와 통신기술의 발달 및 센서(sensor) 기술의 발달로 새로운 데이터의 처리 모델이 요구되어져 왔다. 새롭게 요구되는 데이터 처리 모델은 일시적인 스트리밍 데이터에 대해 연속적(continuous)이고 장기간 실행되는 형태의 질의를 처리해야 한다[1,2]. 예를 들면, 증권 시세 표시기, 네트워크 트래픽 모니터링, 웹 로그 분석, 트랜잭션 로그 분석 및 센서 네트워크 등이 될 것이다. 이러한 환경의 특징은 온라인으로 처리되어지는 다중 고속 데이터 스트림의 형태로 데이터가 도착하며, 질의에 대한 응답은 거의 실시간으로 제공되는 것을 요구한다.

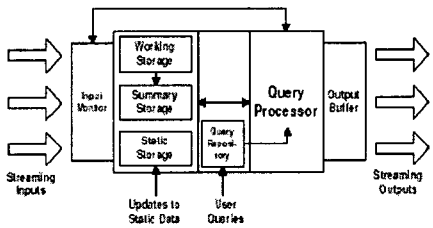
데이터 스트림은 연속성, 시변성(time-varying), 데이터 아이템의 무한 순서의 특징을 가지고 있다. 스트림 아이템은 일반적으로 처리 과정을 거친 후 관계형 튜플의 형태로 저장되며, 온라인 스트림 알고리즘은 데이터에 대해 한 번의 처리를 의미한다. 데이터의 도착율(arrival rate)은 정해지지 않았으며 처리 시스템에 의해 조절할 수 도 없다. 또한, 예측이 불가능하고 시간당 엄청난 양의 변동폭을 가지고 있다. 더욱이, 무한 속성을 가진 데이터를 전부 저장한다는 것은 불가능하다.

전형적인 데이터 스트림에서 질의 처리의 가장 중요한 부분은 연속질의(continuous query)에 대한 부분이다. 새롭게 도착하는 데이터에 대해 연속적인 처리를 하는, 장기간에 걸쳐 수행되는 질의에서 처럼 새로운 결과를 계속해서 생성해야 한다.

2. 데이터 스트림 관리 시스템 (DSMS:Data Stream Management System)

[그림1]은 데이터 스트림 환경에서의 질의 처리에 대한 설명을 단계적으로 나타낸 데이터 스트림 관리 시스템(DSMS)의 구조도이다[2].

DSMS[3]는 다중 데이터 스트림에 대하여 다중 연속 질의 처리를 수행한다. 여기서 입력 데이터의 과부하로 인하여 시스템의 처리 용량을 초과하는 현상이 발생 할 수 도 있다. 따라서, 입력 모니터(input monitor)는 데이터 입력의 양을 조절하는 기능을 하며, 유지가 어려울 경우 패킷을 일부 버리기도 한다.



[그림1] 데이터 스트림 관리시스템(DSMS)의 구조

[그림1]에서 데이터의 저장은 세 부분으로 나눌 수 있다. 첫 번째, 임시 작업 저장소(temporary working storage)와 두 번째, 스트림 시놉시스 처리를 위한 요약 저장소(summary storage) 세 번째, 메타 데이터 처리를 위한 정적 저장소(static storage)이다. 장기간 실행되는 질의는 질의 저장소(query repository)에 저장되고 질의 처리기(query processor)는 입력 모니터와 통신을 하며 입력 데이터양의 상황에 따라 질의 처리에 대한 최적화 작업을 시도한다.

현재 진행중인 DSMS 프로젝트는 NiagaraCQ[4], TelegraphCQ[5], Stream[3], 그리고 Aurora[6] 등이 있다. NiagaraCQ는 인터넷 상에서 분산 XML 파일에 대한 연속 질의에 대한 처리를 위한 초기의 접근이었고, TelegraphCQ는 다중 데이터 스트림에 대한 다중 연속 질의 처리에 목적을 두고 있다. Stream은 자원 관리와 근사(approximate) 연속 질의 처리에 중점을 두고 있고, Aurora는 주로 온라인 스케줄링과 부하 분산(load shedding)을 만족시키는 어플리케이션 모니터링 쪽으로 연구가 진행 중이다.

3. 스케줄링(Scheduling)

단일 또는 다중 연속 질의에 대한 DSMS의 대부분의 관심사항은 준비단계의 연산자들 사이에서 프로세서 할당 문제이다. 스케줄링에서 가장 중요한 것은 정상 상태에서 성능의 향

1) 본 연구는 한국과학재단 목적기초연구(R01-2003-000-10395-0) 지원으로 수행되었음.

상이 아니라, 매우 심각한 과부하 상태에서 부하 분산을 사전에 예방하는데 그 목적이 있다. 데이터 스트림 어플리케이션의 데이터의 도착 속도가 매우 빠르기 때문에 대부분 시스템의 한계 상황에 도달하게 된다. 효율적인 스케줄링은 시스템 자원의 효율을 높이고, 높은 잠재주기(latency)를 최소화하며, 정확도(accuracy)를 높이는 것이 중요한 문제이다.

스케줄링과 관련한 연구로는 [7]에서 우선순위에 의한 할당과 QoS를 이용한 동적 스케줄링 계획을 Aurora System[6]에 적용한 스케줄링 알고리즘을 제안하였으며 스케줄의 과부하(overhead)와 품질(quality)에 대한 트레이드오프(trade-off)를 버킷(bucket) 단위를 이용하여 간략화 하는 기법을 설명하고 있다. [8]에서 데이터 스트림 환경에서 메모리 사용의 최소화를 위한 전략을 제안하였다. 또한, 제안된 방법이 단일 스트림 질의에서 최적화된 것도 증명하였다. 그러나 잠재주기(latency)의 최소화와 기아현상(starvation)의 문제는 완벽하게 해결하지 못하였다.

4. 연속 질의 처리(Continuous Query Processing)

데이터 스트림 처리 환경은 매우 다양하게 그리고 광대한 변동요인이 산재해 있다. 우선, 이러한 비안정적인 요인(factor)을 극복하기 위한 노력이 필요하다.

연산자(operator)의 교환작용(commutativity)을 기초로한 eddy[9] 개념은 여전히 TelegraphCQ 시스템의 주된 질의 처리 근사 메카니즘이며, 최근에는 조인(join) 질의와 관련한 다중 질의에 대한 연구가 활발히 진행 중이다.

Language/ system	Motivating applications	Allowed inputs	Basic operators	Supported windows			Custom operators?
				type	base	execution	
AQuery	stock quotes, network traffic analysis	sorted relations	relational, "each", order-dependent (first, next, etc.)	fixed, landmark, sliding,	time and count	not discussed in [4]	via "each" operator
Aurora	sensor data	streams only	σ, π, ⋈, sort, group-by, resample, drop, map, window sort	fixed, landmark, sliding	time and count	streaming	via map operator
CQL/ STREAM	all-purpose	streams and relations	relational, relation-to-stream, sample	currently only sliding	time and count	streaming	allowed
StreaQuel/ TelegraphCQ	sensor data	streams and relations	relational	all types	time and count	streaming or periodic	allowed
Tribeca	network traffic analysis	single input stream	σ, π, group-by, union aggregates	fixed, landmark, sliding,	time and count	streaming	allows custom aggregates

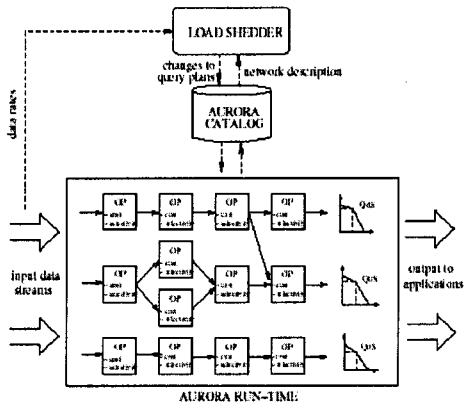
[그림2] 데이터 스트림 질의어 기능 요약

[그림2]는 데이터 스트림 질의어에 대한 기능 요약이다[2]. 데이터 스트림 질의어는 모두 사용자 정의 집계(aggregate)함수를 사용 가능하게 하며, 향후 스트리밍 어플리케이션에서 사용하게 될 질의어의 확장과 패턴매칭(pattern-matching) 기능을 정의하고 있다.

5. 부하 분산(Load Shedding)

푸시 기반(push-based) 데이터 스트림 환경에서 입력 데이터 스트림의 데이터 도착율은 매우 다양하다. 따라서, 높은 데이터 도착율 즉, 과부하의 경우 이용 가능한 자원들은 모든 입력 데이터 튜플의 처리를 수행하기에는 충분하지 못하다. 여기서 두 가지 시스템의 한계에 도달하게 된다. 첫 번째, CPU가 시스템 내에 도착하는 모든 데이터를 처리하기에 충분히 빠르지 못하다(slow CPU 문제). 두 번째, 시스템의 메인 메모리가 데이터 튜플에 상응하는 만큼 충분하지 못하다는 것이다. 그러나 최악의 상황에는 이러한 두 가지 문제가 서로 공존하는 것이 일반 적이다.

이러한 환경을 극복하기 위해 부하를 초과하는 단계의 데이터를 버리는 식의 적절한 방법을 통한 부하 분산이 필요하다. 여기서 두 가지 의문점은 얼마나 많은 튜플을 버릴 것인지와 질의 처리의 어느 단계에서 버릴 것 인지를 결정하는 문제이다. 이에 대한 방법론으로 랜덤(random) 부하분산과 의미적(semantic) 부하분산이 있다.



[그림3] Aurora 시스템의 부하분산

[그림3]은 QoS를 이용한 Aurora 시스템의 부하 분산을 나타낸 것이다[16]. 부하 분산의 양을 결정하기 위해 그림에서처럼 별도의 카탈로그 정보를 관리한다.

[6]에서는 slow CPU 문제 해결을 위해 과부하 문제가 해결될 때 까지 입력 데이터를 버리는 방법을 제안하고 있다. 또한, [10]에서 이동 윈도우 조인을 이용한 부하분산에 대해 제안하였으며, 향후 전체 질의에 대한 단일 조인 사이의 비용(cost) 모델의 확장과 적절한(adaptive) 질의 최적화에 대해 연구과제로 제한하고 있다.

6. 근사화(Approximation)

근사화에 대해서는 데이터 스트림의 요약(summaries)을 통한 근사화가 대부분이며, 현재까지 상황에 따른 세 가지 방법이 제안되었다. 첫 번째 방법은 이전 연구에서는 삽입과 합집합 같은 특정 연산만 가능했었는데, 삭제연산을 가능하게 해주는 방법을 2-레벨 해시 스케치 시놉시스(2-level hash sketch synopses)를 기초로 해서, 합집합 뿐만 아니라, 교집합, 차집합을 포함한 업데이트 스트림 환경에서 집합 표현 처리를 위한 공간 효율 (space-efficient)알고리즘을 제안하고 있다[11]. 센서 네트워크의 최대 단점인 제한된 통신과 제한된 전력(배터리) 환경에서 질의 처리를 위한 데이터의 양을 최소화하는 것이 센서 네트워크 시스템의 설계의 기본적인 고려사항이다. 두 번째 방법은 이러한 상황을 극복하기위한 센서 네트워크 환경에서 근사 집계 질의 방법을 제안하고 있다[12]. 마지막으로 데이터 스트림 환경에서 연속 질의의 흥미로운 분야 중의 하나인 RNNA(Reverse Nearest Neighbor Aggregates)가 제안되었다[13].

7. 분산 데이터 스트림 모니터링

분산 모니터링은 데이터 스트림 환경에서 또 하나의 흥미로운

분야이다. 많은 스트림 기반 어플리케이션들은 분산 데이터의 연속적인 모니터링을 요구한다. 대개의 경우, 수집된 데이터 전부를 중앙 처리 장치로의 연속적인 전송은 많은 비용이 들거나 불필요한 경우가 많다. 고려해 볼 수 있는 다른 방안은 데이터의 필터링 방법과 같이 질의 모니터링을 통해 스트림의 변경 부분만을 업데이트 하는 것이다. 이러한 모니터링 시스템 설계의 기본 목적은 통신비용을 최소화 하자는 것이다.

[14]에서는 네트워크 모니터링과 같이 정확한 데이터가 불필요하면서 많은 스트림 데이터의 효율적인 관리를 위한 분산 데이터 스트림 환경에서 근사 필터를 이용한 방법을 제안하였다. 분산 스트림에서 가장 큰 값 k를 포함하는 연속적인 리포트에 대한 질의를 top-k 모니터링 질의라 하는데 [15]에서는 분산 top-k 모니터링 기법을 제안하였다.

8. 향후 연구방향 및 결론

이제까지 살펴본 데이터 스트림 처리에 관한 연구들은 최근 컴퓨터 과학 분야에서 가장 많은 관심을 가지고 있고 흥미 있는 이슈 중 하나이다. 이러한 경향은 다양한 어플리케이션의 출현으로 질의 처리에 대한 효율적인 방법과 새로운 데이터 스트림의 모델을 필요로 하기 때문일 것 이다. 데이터 스트림 모델 분야에서 가장 중요한 부분은 앞에서도 언급한 스케줄링, 적절한 질의 처리, 부하 분산, 근사화, 분산 데이터 모니터링이라고 생각한다.

데이터 스트림 분야의 향후 연구방향은 새로운 어플리케이션 도메인과의 접목이 주를 이룰 것이라고 생각된다. 데이터 스트림 관리 시스템(DSMS)에 대해서도 많은 부분 연구가 진행되었으며, 그 구조에 대해서도 완성단계에 이르렀다.

스케줄링은 기존 운영체제 분야에서 핵심적으로 다루었던 CPU와 메모리의 효율성을 유지하면서 최적화하려는 시도가 있었던 것과 마찬가지로 데이터 스트림 분야에서도 이 같은 연구가 진행되었다. 그러나 트레이드오프 문제로 데이터 스트림을 간략화 하는 기법으로 발전 되었으며, 여전히 운영체제 분야에서 그랬던 것처럼 기아현상과 잠재주기 문제는 완전히 해결하지 못하였다.

연속 질의 처리는 지금도 많은 연구가 진행되고 있으며, 특히, 관계형 데이터베이스의 질의어인 SQL을 이용해서 기능을 추가한 형태의 질의를 가능하게 하고 있다. 그러나 SQL 수준의 모든 질의가 여전히 가능하지 않다. 이러한 현상은 데이터 스트림의 데이터 특성 때문이기도 하다. 따라서, 향후 질의 성향이 변해가고 또 다른 형태의 어플리케이션이 나올 경우 질의 패턴에 맞추어 연속 질의에 대한 많은 연구를 필요로 하게 된다고 본다. 또한, 준비된 질의뿐만 아니라 ad hoc 질의에 대한 연구도 더욱 필요하다.

부하 분산은 운영체제와도 연결시켜 생각해 볼 수 있지만 데이터 스트림의 데이터 특성을 고려한다면 일회성의 데이터를 적당한 규칙에 의해 버림으로써 해결하고자 한다. 그러나 일부 어플리케이션 도메인에서는 요약에 의한 결과를 허용하지 않으려하는 부분도 있다. 이런 경우에는 100% 정확한 결과를 어떠한 상황에서도 나타내 주어야 한다. 이 부분에 대한 연구가 더욱 필요하리라고 본다.

근사화는 데이터 스트림의 데이터 특성을 잘 반영한 처리 방법의 하나이다. 일회성이고, 센서 환경처럼 대략적인 결과만으로도 충분한 경우에는 아주 유용한 방법이다. 그러나 부하분산에서 와도 마찬가지로 근사화를 허용하지 않는 어플리케이션 도메인에서는 의미를 잃게 된다.

분산 데이터 스트림 모니터링은 새로운 도메인의 하나로 역시 간략화의 방법으로 필터링 기법을 사용하는데 XML과 같은 데이터 타입이 많이 사용되는 환경에서 더 많은 연구 대상이 존재한다.

전반적인 데이터 스트림 기술의 발전을 통하여 전통적인 데이터베이스 관리시스템에서 새로운 데이터 처리 모델을 통합하기 위한 시도가 가능할 것이다. 데이터 스트림 처리에 있어서도 이전에는 중요하게 다루어 지지 않았던 과거 데이터에 대한 처리가 강하게 요구되고 있다. 또한, 데이터 스트림 처리와 아주 밀접한 관련이 있는 센서 네트워크, 분산처리 그리고 전통적인 데이터베이스 시스템과 같은 분야에서도 아직 해결해야 할 많은 연구 과제들이 있다.

참고문헌

[1] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. "Models and Issues in Data Stream Systems." Invited paper in Proc. of PODS, 2002.

[2] Lukasz Golab and M. Tamer Ozsu. "Issues in Data Stream Management." In SIGMOD Record, Volume 32, Number 2, June 2003.

[3] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and R. Varma. "Query Processing, Resource Management, and Approximation in a Data Stream Management System." In Proc. of CIDR, 2003

[4] Chen, J., DeWitt, D. J., Tian, F., Wang, Y. "NiagaraCQ: A Scalable Continuous Query System for Internet Databases." SIGMOD, 2000.

[5] Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M. J., Hellerstein, J. M., Hong, W., Krishnamurthy, S., Madden, S., Raman, V., Reiss, F., Shah, M. "TelegraphCQ: Continuous Dataflow Processing for an Uncertain World." CIDR, 2003.

[6] Abadi, D. J., Carney, D., Cetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N., Zdonik, S. "Aurora: A New Model and Architecture for Data Stream Management." VLDB Journal, 2003.

[7] Carney, D., Cetintemel, U., Rasin, A., Zdonik, S., Cherniack, M., Stonebraker, M. "Operator Scheduling in a Data Stream Manager." VLDB, 2003.

[8] Babcock, B., Babu, S., Datar, M., Motwani, R. "Chain: Operator Scheduling for Memory Minimization in Data Stream Systems." SIGMOD, 2003.

[9] Avnur, R., Hellerstein, J. M. "Eddies: Continuously Adaptive Query Processing." SIGMOD, 2000.

[10] Kang, J., Naughton, J. F., Viglas, S. D. "Evaluating Window Joins over Unbounded Streams." ICDE, 2003.

[11] Ganguly, S., Garofalakis, M., Rastogi, R. "Processing Set Expressions over Continuous Update Streams." SIGMOD, 2003.

[12] Considine, J., Li, F., Kollios, G., Byers, J. "Approximate Aggregation Techniques for Sensor Databases." ICDE, 2004.

[13] Korn, F., Muthukrishnan, S., Srivastava, D. "Reverse Nearest Neighbor Aggregates over Data Streams." VLDB, 2002.

[14] Olston, C., Jiang, J., Widom, J. "Adaptive Filters for Continuous Queries over Distributed Data Streams." SIGMOD, 2003.

[15] Babcock, B., Olston, C. "Distributed Top-K Monitoring." SIGMOD, 2003.

[16] N. Tatbul, U. Çetintemel, S. Zdonik, M. Cherniack, M. Stonebraker. "Load Shedding in a Data Stream Manager." VLDB, 2003