Journal of Korea Multimedia Society Vol. 12, No. 9, September 2009(pp. 1242-1251)

데이터 스트림 환경에서 데이터 완전도 보장을 위한 과부하 예측 부하 분산 기법

김영기[†], 신숭선[††], 백성하[†††], 이동욱[††††], 김경배[†††††], 배해영[††††††]

요 약

유비쿼터스 환경에서 데이터 스트림 관리 시스템(Data Stream Management System: DSMS)은 수많은 센서로부터 생성되는 대량의 데이터 스트림을 처리한다. 기존의 시스템은 처리 능력 이상의 데이터 스트림이 입력되면 데이터의 일부를 제거하여 적정 부하를 유지하는 부하 제한 기법(Load Shedding)을 사용한다. 부하 제한 기법은 입력되는 데이터의 일부를 의도적으로 손실하여 데이터 완전도(Data Completeness)가 감소하기 때문에 처리 결과의 신뢰도 또한 감소한다. 따라서 본 논문에서는 시스템 처리 능력 이상의 데이터 스트림 입력 시 데이터 완전도 보장을 위한 과부하 예측 부하 분산 기법을 제안한다. 제안 기법은 데이터 손실이 예상되는 부하 시점을 미리 예측하고 예측된 부하 시점에 도달 시 부하를 분산하여 데이터 손실을 감소시킨다. 본 논문에서는 기존의 부하 제한 기법과의 비교 실험을 통해 제안 기법의 성능을 평가한다.

Load balancing method of overload prediction for guaranteeing the data completeness in data stream

Young-Ki Kim[†], Soong-Sun Shin[††], Sung-Ha Baek[†††], Dong-Wook Lee[††††], Gyoung-Bae Kim[†††††], Hae-Young Bae[††††††]

ABSTRACT

A DSMS(Data Stream Management System) in ubiquitous environment processes huge data that input from a number of sensor. The existed system is used with a load shedding method that is eliminated with a part of huge data stream when it doesn't process the huge data stream. The Load shedding method has to filter a part of input data. This is because, data completeness or reliability is decreased. In this paper, we proposed the overload prediction load balancing to maintain data completeness when the system has an overload. The proposed method predicts the overload time. and than it is decreased with data loss when achieves the prediction overload time. The performance evaluation shows that the proposed method performs better than the existed method.

Key words: Load Balancing(부하 분산), Data Stream(데이터 스트림), Overload Prediction(과부하 예측), DSMS(데이터 스트림 관리 시스템)

※ 교신저자(Corresponding Author) : 배해영, 주소 : 인천광역시 남구 용현4동(608-743), 전화 : 032)860-8712, FAX : 032)862-9845, E-mail : hybae@inha.ac.kr
접수일 : 2009년 3월 4일, 수정일 : 2009년 7월 1일
완료일 : 2009년 7월 1일
[†] 준회원, 인하대학교 정보공학과 석사과정
(E-mail : ykkim@dblab.inha.ac.kr)
[††] 준회원, 인하대학교 정보공학과 박사과정
(E-mail : hermit@dblab.inha.ac.kr)
[†††] 인하대학교 정보공학과 박사과정
(E-mail : shbaek@dblab.inha.ac.kr)
[††††] 준회원, 인하대학교 정보공학과 박사과정
(E-mail : dwlee@dblab.inha.ac.kr)
[†††††] 정회원, 서원대학교 컴퓨터교육과 조교수
(E-mail : gbkim@seowon.ac.kr)
[††††††] 정회원, 인하대학교 정보공학과 조교수
※ 본 연구는 건설교통부 첨단도시기술개발사업 - 지능형 국토정보기술혁신 사업과제의 연구비지원(07국토정보 C05)에 의해 수행되었습니다.

1. 서 론

수많은 센서와 그것을 처리하는 컴퓨터간의 유·무선 통신 기술의 발전은 유비쿼터스라는 새로운 시대를 열었다. 컴퓨터는 장소와 시간에 구애받지 않는 서비스를 제공할 수 있게 되었고, 사용자는 컴퓨터의 존재를 인지하지 않고 눈에 보이지 않는 정보의 흐름을 이용하여 서비스를 실시간으로 제공 받게 되었다. 서비스의 예로는 실시간 교통 정보 시스템, 건강 모니터링 시스템, 환경 모니터링 시스템, 실시간 금융 서비스, 주식 정보 서비스, 텔레매틱스 등을 들 수 있다. 이러한 서비스들은 고 수준의 컴퓨팅 능력을 가진 컴퓨터가 실세계를 모니터링하는 형태의 시스템으로 제공된다. 시스템들은 보통 센서와 같이 데이터를 지속적으로 생성하는 물리적 개체로부터 데이터를 전송받아 처리하게 되는데 이와 같이 지속적으로 생성되는 데이터를 데이터 스트림(Data Stream)이라 하며 데이터 스트림을 처리 및 관리하는 시스템을 데이터 스트림 관리 시스템(Data Stream Management System: DSMS)이라 한다[1,2]. DSMS는 지속적으로 발생하는 데이터 스트림을 실시간으로 처리하여 사용자에게 서비스를 제공한다. 하지만 눈에 보이지 않는 정보의 흐름, 즉 데이터 스트림의 지속적으로 빠르게 발생되며 그 경계가 불분명하다는 특징이 있기 때문에 DSMS는 데이터 스트림을 실시간으로 처리하기 위하여 고려해야할 문제점이 존재 한다. 만약 데이터 스트림의 전송률이 증가하거나 많은 질의가 등록 된다면 시스템은 제한적인 CPU 처리 능력과 메모리의 한계로 많은 양의 데이터 스트림을 처리하지 못하고 유실될 수 있다. 또한 데이터 스트림의 전송률뿐 아니라 데이터 스트림의 크기, 데이터 스트림의 종류, 질의의 종류 등에 따라 질의 처리 시간의 지연을 초래하는 경우도 과부하로 인한 데이터 손실을 초래할 수 있다[3-6]. 이러한 문제점을 해결하기 위해 기존의 DSMS는 시스템 처리 능력 이상의 데이터 스트림 입력 시 데이터의 일부를 제거하는 부하 제한 기법(Load Shedding)을 사용한다[6,7]. 하지만 부하 제한 기법을 통해 데이터의 일부가 제거될 경우 남아있는 데이터의 질의 처리 결과의 신뢰도 유지와 QoS 보장은 어렵다.

본 논문에서는 부하 제한 기법과 달리 데이터 손실의 감소와 신뢰도 증가를 위해 데이터 완전도 보장을 위한 과부하 예측 부하 분산 기법을 제안한다. 본 제안 기법은 입력되는 데이터의 증가 또는 질의의 증가로 인해 발생 되는 부하 시점을 미리 예측하고 예측 된 시점에 부하를 분산하여 입력되는 데이터의 손실을 감소 시켜 신뢰도를 보장한다. 제안 기법의 수행 과정은 3단계로 진행되며 첫 번째 단계에서는 센서, RFID, 엣지 서버(Edge Server) 등으로부터 입력 받은 데이터 스트림의 정보와 질의 처리 비용 정보를 통해 과부하 시점을 미리 예측한다. 두 번째 단계에서 예측된 시점을 기반으로 과부하 위험 구간을 설정하며 설정된 구간을 약 구간(Weakness Interval)이라 한다. 세 번째 단계에서는 설정된 약 구간에 데이터가 입력되면 서브 데이터 스트림을 생성하여 질의를 재분배하여 부하를 분산한다. 생성된 서브 데이터 스트림에 부하를 분산하게 되면 분산된 질의의 처리 능력을 증가 시킬 수 있기 때문에 데이터 손실이 감소되고 그에 비례하여 데이터의 완전도가 증가하기 때문에 처리 결과의 신뢰도가 높아진다.

본 제안기법은 성능 평가에서 기존의 DSMS에 사용하여 데이터 스트림을 처리한 결과, 기존의 부하 제한 기법에 비해 데이터 손실이 발생하지 않아 데이터 완전도를 보장하였고 처리 결과의 신뢰도가 증가했다.

본 논문의 구성은 2장에서 관련 연구로 데이터 스트림 관리시스템과 사용되는 부하 제한 기법에 대해 설명하며 3장에서는 본 논문이 제안하는 과부하 예측 제어기를 사용한 부하 분산 기법에 대하여 소개한다. 4장에서는 본 제안 기법과 부하 제한 기법의 비교를 통해 제안 기법의 우수성을 증명하고 제안 기법의 자체 성능 평가를 통해 과부하에 대한 대응력을 점검한다. 마지막으로 5장에서는 결론 및 앞으로의 연구 방향과 보완점을 논의한다.

2. 관련 연구

2.1 데이터 스트림 관리 시스템의 부하 제한 기법

데이터 스트림은 기존의 정적인 데이터와 달리 지속적으로 빠르게 발생하고 크기나 경계가 불규칙적이라는 특징을 갖고 있으며 생성된 시간 혹은 입력된 시간으로 정렬된다. 따라서 기존의 DBMS처럼 모든 데이터 스트림을 저장하거나 처리하는 것은 어렵다. 그렇기 때문에 데이터 스트림을 처리하기 위하여

DSMS가 연구되고 있다[2-5]. DSMS는 입력되는 데이터 스트림을 다수의 연속 질의를 통해 처리하고 그 결과를 사용자 또는 타 시스템에 전달하는 것을 주목적으로 한다. 연속 질의란 기존의 DBMS의 질의와 달리 시스템에 등록되어 일정 시간동안 존재하며 지속적으로 처리하는 질의이다. 연속 질의 처리는 지속적으로 입력되는 데이터 스트림을 질의 조건과 비교하여 부합하는 결과를 사용자에게 전달한다. DSMS의 대표적인 연구로 NiagaraCQ[8], PSoup[9], Aurora[1] 등이 있다.

앞서 말한 데이터 스트림의 특징들에 의해 DSMS는 많은 시스템 자원을 사용하게 되고 높은 컴퓨팅 시스템을 요구한다. 특히, 전송률이 높은 데이터 스트림에 한해서는 CPU 처리 능력이 한계를 보일 수 있기 때문에 이에 따른 부하 제한 기법(Load Shedding)을 사용한다[10-12]. 부하 제한 기법은 처리 되지 않은 데이터의 일부를 버리게 되는 것으로 크게 두 가지로 구분한다. 첫째는 데이터 스트림의 전송속도에 따라 데이터를 무작위로 제한하는 부하 제한 기법이고 둘째는 데이터 스트림의 중요도(Importance Level)에 따라 데이터를 제한하는 기법이다. 하지만 부하 제한을 통해 데이터 스트림 내의 일부 데이터를 버리게 되기 때문에 이때 남아있는 데이터 스트림의 결과가 신뢰도를 결정하므로 QoS를 보장하는 것은 어렵다.

2.2 데이터 완전도(Data Completeness)와 데이터 손실률(Miss Ratio)의 관계

DSMS는 빠르고 지속적으로 입력되는 데이터와 질의 처리기의 처리 지연으로 과부하가 발생 가능하다. 이러한 과부하로 인한 데이터 손실을 방지하기 위한 기법들은 데이터 손실률(Miss Ratio)과 데이터 완전도(Data Completeness)를 기반으로 한다. 데이터 완전도는 입력된 데이터 중에 질의 처리기로 전달된 데이터의 비율이다. 만약 모든 데이터가 질의 처리기로 전달된다면 데이터 완전도는 100%이고 신뢰도를 보장한다고 할 수 있다. 데이터 손실률은 입력된 데이터와 질의 처리 하지 못하고 버려진 데이터의 비율이다. 이는 데이터 완전도와 상충하는 관계를 갖는데, 데이터 손실률이 증가할수록 데이터 완전도가 감소하기 때문에 의도적이지 않은 데이터 손실률의 증가는 질의 처리 결과의 신뢰도를 보장하기 어렵다. 이를 해결하기 위해, 앞서 소개한 부하 제한 또는 본 절에서 소개할 데이터 샘플링(Data Sampling) 기법 등을 사용한다[6,7].

데이터 샘플링의 주요 목적은 데이터 스트림내의 유용한 데이터만으로 결과를 도출하는 것이다. 만약 10개의 데이터가 데이터 스트림 윈도우상에 존재할 때 시스템 자원의 부족으로 데이터를 모두 유지 및 처리하기 어려워질 수 있다. 따라서 윈도우 내의 10개의 데이터 중 유용한 데이터만을 유지한다면 시스템 자원 부족의 문제는 해결 가능하다. 이렇게 유용한 데이터만으로 추출된 데이터를 샘플링된 데이터라 한다. 앞서 가정한 10개의 데이터 중 4개가 유용한 데이터로 샘플링 되어 질의가 수행됐다면 데이터 완전도는 40%라 할 수 있다. 이러한 기법은 데이터 완전도를 감소시키지만 결과의 신뢰도는 높일 수 있다는 장점이 있다. 따라서 DSMS에서 사용되는 부하 제한이나 데이터 샘플링과 같은 기법들은 데이터 완전도 증가 및 질의 처리 결과의 신뢰도를 보장하기 위하여 연구되어야 한다.

3. 과부하 예측 부하 분산 기법

3.1 과부하 예측 제어기를 통해 부하 분산을 수행하는 확장된 DSMS

그림 1은 과부하 예측 제어기(Overload Prediction Controller: OPC)를 통해 부하 분산을 수행하는 확장된 DSMS의 시스템 흐름도이다. 데이터는 센서, RFID, 엣지 서버 등과 같은 소스(Source)로부터 발생되어 시스템으로 전달된다. 전달된 데이터는 시스템 내의 해당 데이터 스트림의 큐(Queue)에 임시로 저장되고 과부하 예측 제어기를 통해 약 구간을 설정하거나 해제한다. 만약 입력된 데이터가 약 구간에 속한다면 과부하 예측 제어기는 서브 데이터 스트림을 생성하고 원본 데이터 스트림에 배정된 질의의 일부를 서브 데이터 스트림으로 재분배하는 부하 분산을 수행한다. 데이터 스트림에 입력된 데이터는 입력된 순서대로 질의 처리기로 입력되어 연산자 박스를 따라 연산을 수행하고 수행된 데이터는 각 연산자 박스에 할당되어 있는 시놉시스(Synopsis)에 저장된다. 모든 연산을 수행한 데이터는 사용자나 다른 시스템 또는 어플리케이션 등으로 전달된다.

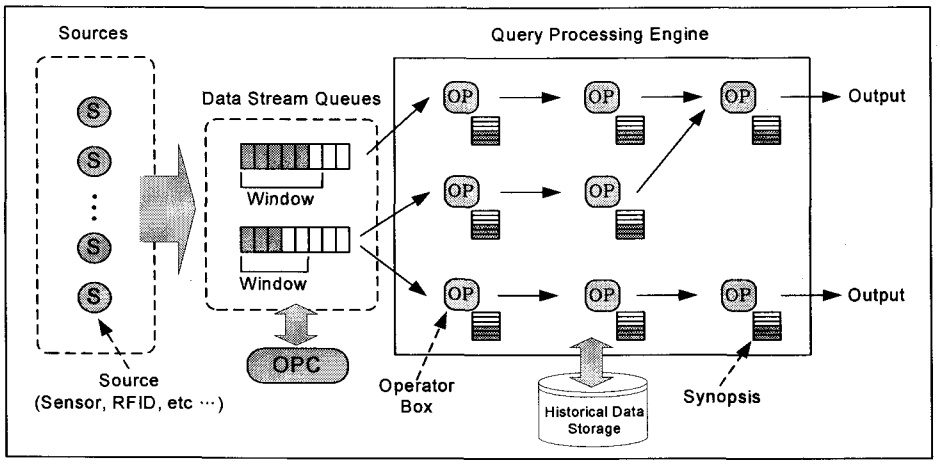


그림 1. JPEG 부호화 및 복호화 블록도

3.2 과부하 예측 제어기의 처리 과정

과부하 예측 제어기의 목적은 데이터 스트림의 과부하에 따른 데이터 완전도 보장과 데이터 손실률의 감소에 있다. 따라서 데이터 스트림에 데이터가 입력되는 시점에서 수행되며 과부하 시점을 미리 예측하여 위험 구간을 약 구간으로 설정하거나 해제한다. 만약 입력된 데이터가 약 구간에 도달 하면 해당 데이터 스트림을 복제하여 등록된 질의의 일부를 복제된 서브 데이터 스트림으로 재분배한다.

그림 2는 과부하 예측 제어기의 처리 과정을 나타낸다. 모니터(Monitor)는 과부하 예측 제어기에게 과부하 예측을 위한 질의 처리 정보를 제공하게 된다. 데이터 스트림 S에 관한 질의를 $q_1,...,q_n$이라고 가정할 때 각 질의 q_i는 연산자 $o_1,...,o_n$을 갖는다. 이에 관한 모니터가 제공하는 정보는 표 1과 같다.

$C(o_i)$ 는 연산자 o_i에 대한 처리 비용을 나타내며 ms(millisecond)로 표기한다. 각 질의 q_i의 처리 비용은 해당 질의의 전체 연산자의 비용으로 얻는다. 과부하 예측 제어기는 앞서 말한 질의 처리 정보와 각 데이터 스트림에 관한 정보를 이용하여 과부하 예측을 한다. 각 데이터 스트림에 관한 정보는 표 2와 같다.

I(t)는 각 데이터 스트림에 전송되는 데이터의 간격으로 ms로 표기한다. 각 데이터 스트림 S_i의 큐 전체 공간은 S_{Ei}로 표기하며 여유 공간은 S_{Ri}로 나타낸다. 또한 t는 전송되는 데이터의 크기를 의미한다. 과부하 예측은 과부하 발생 이전에 데이터 손실 없이 분배 가능한 시점을 예측하며 이를 위한 식은 표 3과 같다.

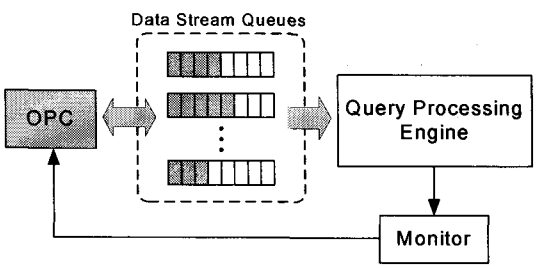


그림 2. 과부하 예측 제어기의 처리 과정

표 1. 질의 처리 정보

◦ $C(o_i)$: 연산자 o_i의 처리 비용(ms)
◦ $C(q_i)$: 질의 q_i의 처리 비용(ms)

표 2. 데이터 스트림 정보

◦ I(t): 데이터의 전송 간격 (ms)
◦ t: 데이터의 크기
◦ S_{Ei}: 데이터 스트림 S_i의 큐 전체 공간
◦ S_{Ri}: 데이터 스트림 S_i의 큐 여유 공간

표 3. 과부하 예측을 위한 식

(식1) $C(q_i) = \sum_{j=0}^{k} C(o_{ij})$ (k = 연산자 개수)
(식2) $P(S) = \frac{I(t)}{MAX(C(q))}$, $1 \le P(S)$
(식3) $I_{weak} = (S_R \bmod (\frac{MAX(C(q)) - I(t)}{I(t)} \times t)) / S_E$

식 (1)은 각 질의의 처리 비용을 계산하며 처리 비용이 가장 큰 질의는 MAX(C(q))로 나타낸다. 식 (2)의 결과 값은 데이터 스트림 S_i의 질의 처리 능력에 대한 값으로 데이터의 발생 빈도와 해당 데이터 스트림에 대한 질의 연산 비용으로 얻게 되며 $1 \leq P(S)$를 만족할 시 데이터 완전도는 100%가 되며 데이터 손실률은 0%가 된다. 만약 만족하지 못할 경우 약 구간(= I_{weak})을 설정하게 되는데 약 구간은 식 (3)을 이용하여 구하게 된다. 약 구간은 과부하 예측을 위한 구간이며 입력된 데이터가 약 구간에 속하게 되면 데이터 스트림을 복제하여 서브 데이터 스트림을 생성하게 된다. 그리고 원본 데이터 스트림이 $1 > P(S)$를 만족할 때까지 가장 비용이 큰 질의 순으로 서브 데이터 스트림에 분배하여 등록한다. 또한 약 구간의 갱신은 과부하의 불예측성에 의한 위험을 방지하기 위해 과거의 약 구간보다 현재의 약 구간이 더 클 경우 갱신하여 오차 범위를 넓히거나 최소 구간과 최대 구간의 평균 구간(= AVG(I_{weak})) 이하일 경우 좁혀서 갱신한다.

그림 3은 과부하에 따른 데이터 완전도 변화 그래프이다. 그래프 상의 good zone은 데이터 완전도를 보장하는 시점이며 데이터 손실률이 0%인 구간이다. 그래프 (a)는 부하 제한 기법의 데이터 완전도 그래프이다. good zone 이후의 r의 위치는 시스템의 처리 성능 이상의 부하 발생으로 인해 데이터 완전도가 감소하는 시점이며 r 이후의 구간은 식 (2)의 $1 \leq P(S)$를 만족하지 못하는 구간이다. r 이후 부하의 증가에 따라 데이터 완전도가 감소하는 것을 볼 수 있고 이는 처리 결과의 신뢰도 또한 감소한다고 할 수 있다. 그래프 (b)는 제안 기법 사용으로 인한 데이터 완전도 보장을 나타낸다. 빗금 상자는 과부하 예측 제어기에 의해 예측된 약 구간을 의미한다. 제안 기법은 그래프 (a)와 동일한 r의 위치를 예측하여

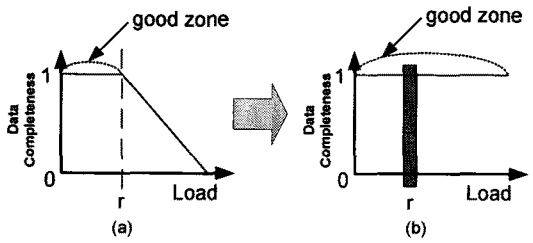


그림 3. 과부하에 따른 데이터 완전도 변화 그래프

약 구간으로 설정하고 부하 분산을 수행하여 데이터 완전도를 보장한다. 따라서 그래프 (a)와 달리 r 이후의 구간도 good zone을 유지한다.

3.3 과부하 예측 제어기의 알고리즘

과부하 예측 제어기의 알고리즘은 크게 두 가지로 분류할 수 있으며 표 4는 과부하가 발생할 가능성이 있는 지점을 예측하여 약 구간을 설정하는 알고리즘이다.

먼저 해당 데이터 스트림에 대한 정보를 얻는다 (01 - 04). 그 다음 데이터 발생 간격에 대한 질의 처리 능력을 측정(05)하고 질의 처리 능력이 데이터 발생 빈도보다 부족할 때 표 3의 식 (3)를 이용하여 약 구간을 얻는다(07). 예측된 약 구간이 현재 해당 데이터 스트림에 설정되어있는 약 구간보다 클 경우 또는 해당 데이터 스트림의 평균 약 구간보다 작을

표 4. 과부하 예측에 의한 약 구간 설정 알고리즘

```
Procedure Overload_Prediction(S)
Input
   S : 데이터 스트림 식별자
Output
   SET_WEAK_SIG : 약 구간 설정 수행
   DEL_WEAK_SIG : 약 구간 삭제 수행
Variables
   P : 데이터 발생 간격에 대한 질의 처리 능력
   I : 데이터 발생 간격
   SR : 데이터 스트림 큐의 여유 공간
   SE : 데이터 스트림 큐의 전체 공간
   QC : 최대 질의 처리 비용
   W : 약 구간
begin
01 I := GetDataInterval(S)
02 QC := MaxQueryCost(S)
03 SR := GetRemainSpace(S)
04 SE := GetEntireSpace(S)
05 P := I / QC
05 if( P < 1 ) {
07   W = ( SR mod ( ( ( QC - I ) / I ) * t ) ) / SE
08   if(GetWeak(S) < W) {
09      SetWeak(S, W)
10   }
11   else if(AvgWeak(S) > W) {
12      SetWeak(S, W)
13   }
14   return SET_WEAK_SIG
15 }
16 else {
17   DeleteWeak(S)
18   return DEL_WEAK_SIG
19 }
end
```

경우 갱신한다(08 - 15). 만약 현재 해당 데이터 스트림의 모든 데이터를 처리 가능하다면, 다시 말해 데이터 발생 간격에 대한 질의 처리 능력의 값이 1이상이라면 약 구간을 해제한다(16 - 19).

표 5는 데이터의 약 구간 진입 시 부하 분산 과정을 나타낸 알고리즘이다. 먼저 입력된 데이터의 위치와 현재 데이터 스트림에 설정되어있는 약 구간을 얻는다(01 - 02). 다음 데이터의 저장 위치가 약 구간에 포함된다면(03) 서브 데이터 스트림을 생성한다(04). 원본 데이터 스트림에 등록되어있는 질의 중 비용이 큰 질의를 서브 데이터 스트림에 등록하고 등록한 질의는 원본 데이터 스트림으로부터 삭제한다(06 - 07). 원본 데이터 스트림의 질의 처리 능력이 1 이상이 될 때까지(08 - 11) 반복한다(05 - 12).

표 5. 데이터의 약 구간 진입 시 부하 분산 알고리즘

```
Procedure Load_Control(S, T)
Input
   S : 데이터 스트림 식별자
   T : 현재 입력된 데이터의 식별자
Output
   OVER_WEAK_SIG : 입력된 데이터가 약 구간에
                   진입 하여 부하 분산을 수행
   UNDER_WEAK_SIG : 입력된 데이터가 약 구간
                    에 진입하지 않음
Variables
   Pos : 현재 데이터가 입력된 위치
   CW : 설정되어 있는 약 구간
   P : 데이터 발생 간격에 대한 질의 처리 능력
begin
01 Pos := GetPos(T)
02 CW := GetWeak(S)
03 if( Pos >= CW ) {
04   SubS = CloneStream(S)
05   while(true) {
06     RegQuery( SubS, GetMaxQuery(S) )
07     DeleteMaxQuery(S)
08     P = I / MaxQueryCost(S)
09     if( P >= 1 ) {
10       break
11     }
12   }
13   return OVER_WEAK_SIG
14 }
15 else {
16   return UNDER_WEAK_SIG
17 }
end
```

표 6. 과부하 예측 제어기의 수행 예시의 가정

- 데이터 소스 : 엣지 서버(Edge Server)
- Stream S_1(int ID, int type, int temp)
- $C(q_1)$: 1ms, $C(q_2)$: 2ms, $C(q_3)$: 8ms, $C(q_4)$: 2ms
- I(t): 2ms, t: 12 bytes
- S_{E1}: 4096bytes, S_{R1}: 1024bytes

3.4 과부하 예측 제어기의 알고리즘

본 절에서는 과부하 예측 제어기의 실제 수행과정의 예를 소개한다. 본 예시를 위한 데이터 스트림 S_1의 가정은 표 6과 같다.

데이터 스트림 S_1은 엣지 서버(Edge Server)로부터 데이터를 전송 받아 해당 데이터 스트림의 큐에 저장한다. 이때 제어기는 S_1의 정보(표 2 참조)를 갱신하게 되며 데이터는 해당 질의로 전달되어 처리된다. 질의가 처리되면 질의 처리 정보(표 1 참조)를 모니터로 전달하여 갱신하게 되며 제어기는 모니터로부터 질의 처리 정보를 전달 받아 표 3의 식을 이용하여 약 구간을 갱신하게 되고 서브 데이터 스트림 생성 및 질의 분배 여부를 결정한다. 표 6의 가정에 따라 데이터는 2ms 마다 입력되며 데이터 스트림 S_1에 대한 질의가 q_1,q_2,q_3,q_4일 때 가장 소비가 큰 질의는 q_3이다. 이를 식 (2)에 적용하게 되면 질의 처리 능력이 0.25이기 때문에 약 구간을 설정해야 한다. 따라서 과부하 예측을 위해 식 (3)를 이용하여 약 구간을 설정한다. 식 (3)의 결과로 약 구간은 0.03이 된다. 이와 같은 과정은 그림 4에서 볼 수 있다.

만약 새로 입력된 데이터가 약 구간에 도달한다면 서브 데이터 스트림 생성 후 질의 분배가 발생한다. 질의 분배 대상은 질의 처리 지연을 발생시키는 q_3이며 질의 분배로 인해 q_3의 질의 소비 비용이 감소한다. 이는 일시적인 부하이거나 DSMS의 질의 체인 구조 기반의 연산자 공유 정책에 의한 처리 부하일 수 있기 때문에 질의 분배와 동시에 연산자를 재구성

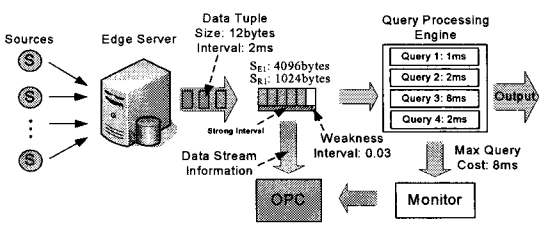


그림 4. 과부하 예측 제어기 수행 흐름도

하여 비용 감소를 얻었다. 연산자 재구성을 위한 DSMS의 질의 체인 구조와 연산자 공유 정책에 대한 스케줄링은 본 논문에서 깊게 다룰 부분이 아니므로 향후 연구로 남긴다.

4. 성능 평가

본 장에서는 제안 기법의 성능을 평가하기 위한 몇 가지 실험의 결과를 설명한다. 실험의 주요 목적은 제안 기법을 사용한 시스템에 과부하 예측으로 인한 데이터 완전도 보장과 데이터 손실률의 감소이며 실험을 위해 과부하 시뮬레이터(Overload Simulator)를 사용하였다. 과부하 시뮬레이터는 제안 기법의 성능을 평가하기 위하여 개발된 어플리케이션으로 시스템 관리 툴의 일종이다. 과부하 시뮬레이터를 이용하여 임의로 부하를 조절하고 시스템의 모니터로부터 시스템 정보를 전달받아 부하에 따른 변화를 점검한다. 실험은 Fedora Core 8.0을 기반으로 수행하였으며 시스템 장비는 Intel(R) Pentium (R) 4 CPU 3.00Ghz 칩셋, 4Gigabyte main memory를 사용하였다. DSMS는 C++로 개발되었으며 과부하 시뮬레이터는 자바로 개발하였다.

4.1 실험환경

실험은 크게 두 가지로 분류하며 하나는 기존의 부하 제한 기법과 본 논문에서 제안하는 과부하 예측을 통한 부하 분산 기법의 데이터 완전도와 데이터 손실률의 변화 비교이고, 또 다른 하나는 제안 알고리즘의 자체 성능 평가로 부하에 따른 메모리 사용량을 측정하였다. 먼저 데이터 완전도 및 데이터 손실도의 변화 비교는 기존의 부하 제한 기법으로 Aurora에서 사용한 데이터의 유용도에 따른 부하 제한 기법과 전송 속도에 따른 무작위 제한 기법을 이용하여 제안 기법과 비교한다. 실험을 위해 개발된 DSMS에 앞서 말한 두 가지 부하 제한기와 과부하 예측 제어기를 구현하여 원하는 기법을 사용 가능하도록 하였다. 실험에 사용된 시스템 설정은 표 7과 같다. 시스템에서 사용되는 메모리는 512Mbytes이고 데이터 스트림를 저장하는 큐는 각각 163kbytes이다. 또한 입력되는 데이터의 크기는 12bytes이다.

부하 조절은 질의의 개수 증가와 데이터 전송 빈도의 증가로 조절 되며 실험에 사용되는 부하 변화

표 7. 데이터 완전도 및 데이터 손실률 변화 실험을 위한 설정

속성	값
전체 사용 메모리	512Mbytes
데이터 스트림 큐 크기	163kbytes
데이터의 크기	12bytes
데이터 전송 빈도	1 - 20ms
질의의 개수	4 - 12
실험 시간	200sec

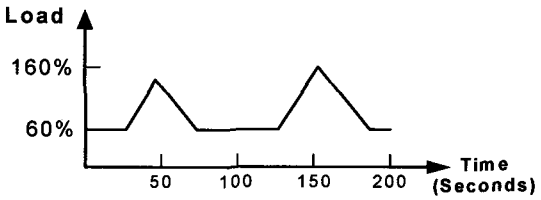


그림 5. 부하 변화 그래프

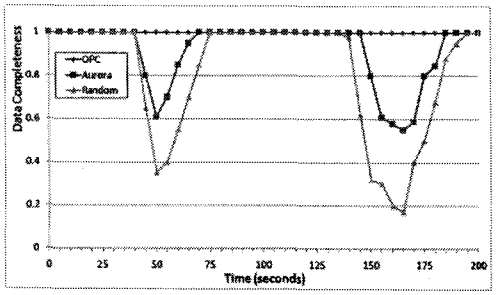
는 그림 5의 그래프와 같다. 30 - 70 구간에 부하를 점차 140%정도로 조절하고 다시 감소시킨 후 130 - 190 구간에 160%로 부하를 증가 시킨다.

4.2 성능 평가

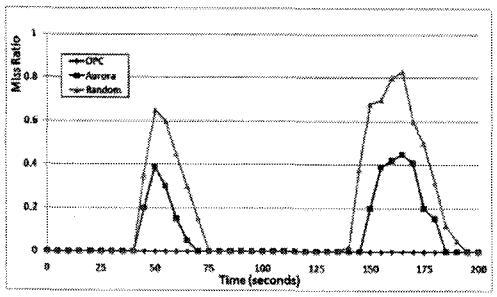
그림 6은 데이터 완전도와 데이터 손실률에 대한 실험 결과이다. 데이터 손실률이 증가하는 만큼 데이터 완전도가 감소하는 것을 볼 수 있으며 무작위 부하 제한은 최대 80% 이상의 데이터가 손실된 것을 볼 수 있었다. 그에 반해 Aurora에서 사용된 데이터 유용도에 따른 부하 제한 기법은 과부하에 대한 효율적인 스케줄링으로 손실률을 감소 시켰다. 하지만 데이터 유용도에 따른 부하 제한 기법과 무작위 부하 제한 기법은 원본 데이터의 일부를 제거하므로 데이터 신뢰도가 완벽하다고 볼 수 없다.

본 논문이 제안하는 기법은 과부하에 의한 데이터 손실이 없었으며 따라서 데이터 완전도도 100%를 보장하였다. 특히 부하손실을 예측하여 과부하 시점에 부하 분산을 수행하였기 때문에 점진적으로 증가하는 부하에 대처가 가능하였다. 그림 7은 부하에 따른 제안 기법의 약 구간 범위 변화를 나타냈다. 약 구간 범위는 부하의 증가 또는 감소에 따라 변화 하였다.

그림 8은 4.1절의 실험 중 측정한 메모리 사용량의 증가 그래프이다. 메모리는 실험에서 사용된 데이터 스트림이 부하의 증가에 따라 서브 데이터 스트림을



(a) 데이터 완전도 비교 그래프



(b) 데이터 손실률 비교 그래프

그림 6. 데이터 완전도와 데이터 손실률 실험 결과

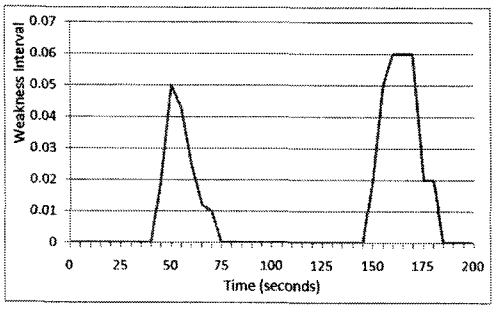


그림 7. 약 구간 범위의 변화 그래프

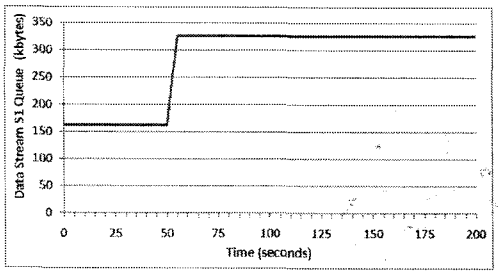


그림 8. 메모리 사용량 증가 그래프

생성할 때 증가하게 된다. 먼저 30 - 70초 구간에서 점진적으로 증가하는 부하에 따른 메모리의 증가 시점은 약 45초이며 그림 5의 부하량을 대조하면 메모리 증가 시점의 부하는 130%이다. 하지만 130 - 190초 구간에서는 이미 생성된 서브 데이터 스트림의 사용으로 인해 메모리가 증가하지 않았다.

5. 결론 및 향후 연구

본 논문에서는 과부하에 의한 데이터 손실 감소와 데이터 완전도 보장을 위하여 과부하 예측 부하 분산 기법을 제안하였다. 제안 기법은 과부하 발생으로 인한 데이터 손실이 가능한 시점을 미리 예측하여 약 구간으로 설정하고 약 구간에 도달 시 서브 데이터 스트림을 생성하여 부하를 분산한다. 기존의 부하 제한 기법과 비교하여 데이터 완전도와 데이터 손실률을 측정하였으며 자체 성능 평가도 수행하였다. 그 결과 제안 기법은 기존의 부하 제한 기법과 달리 데이터 손실 시점 이전에 부하 분산을 하여 데이터 손실률은 0%를 유지하였고 데이터 완전도는 100%를 유지하였다. 따라서 제안기법이 질의 처리 결과에 대한 신뢰도를 보장한다는 것을 알 수 있었다. 본 제안 기법은 증권 시세 모니터링이나 건강 모니터링 시스템 등과 같이 지속적으로 발생하는 데이터 스트림의 처리 결과가 신뢰도를 보장해야하는 환경에서 효율적으로 사용될 수 있다.

향후 연구방향은 두 가지로, 첫째는 과부하에 의해 생성된 서브 데이터 스트림의 결합 시점과 그에 따른 동시성 제어이다. 본 논문에서 제안한 과부하 예측 부하 분산 기법은 데이터 손실이 예측되는 약 구간에서 서브 데이터 스트림을 생성하기 때문에 사용되는 메모리가 증가한다. 따라서 부하가 정상화되면 원본 데이터 스트림과 서브 데이터 스트림을 결합해야할 필요성이 있다. 그렇기 때문에 결합 시점 정의에 대한 연구가 필요하며 결합 시 동시성을 보장해야 하므로 동시성 제어에 대한 연구도 필요하다. 둘째는 제안 기법과 부하 제한 기법의 결합이 있다. 실제 네트워크상의 변화는 불규칙적이기 때문에 고려하지 못한 상황이 발생 가능하다. 따라서 부득이하게 데이터 손실을 감당해야할 경우 질의 처리 결과의 신뢰도를 최대한 보장하는 부하 제한 기법과의 결합에 대한 연구가 필요하다.

참 고 문 헌

[1] D. Abadi, D. Carney, U. Centintemel, M.

Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul and S. Zdonik, "Aurora: A New Model and Architecture for Data Stream Management," *VLDB J.*, Vol.12, No.2, pp. 120-139, 2003.

[2] B. Babcock, S. Babu, M. Datar, R. Motwani and J. Widow, "Model and Issues in Data Stream System," *Proc. of ACM PODS*, pp. 1-16, 2002.

[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] Lukasz Golab and M. Tamer Ozsu, "Issues in Data Stream Management," *In SIGMOD Record*, Vol.32, No.2, June 2003.

[5] J. Gehrke (ed.), "Special Issue on Data Stream Processing," *IEEE Data Eng. Bull.*, 2003.

[6] B. Babcock, M. Datar and R. Motwani, "Load Shedding for Aggregation Queries over Data Streams," *Proc. of the 20th ICDE*, pp. 1-12, 2004.

[7] N. Tatbul, U. Cetintemel, S. Zdonik, M. Cherniack and M. Stonebraker, "Load Shedding in a Data Stream Manager," *Proc. of the 29th VLDB Conf.*, pp. 309-320, 2003.

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

[9] S. Chandrasekaran and M. Franklin, "PSoup: A System for Streaming Queries over Streaming Data," *VLDB J.*, Vol.12, No.2, pp. 140-156, 2003.

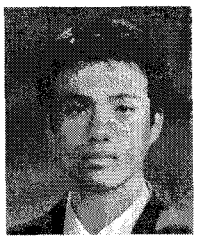
[10] J. Considine, F. Li, G. Kollios and J. Byers, "Approximate Aggregation Techniques for Sensor Databases," *ICDE*, 2004.

[11] G. S. Manku and R. Motwani, "Approximate frequency counts over data streams," *In Proc. of the 28th Conference on Very Large Databases*, 2002.

[12] 백성하, 이동욱, 김경배, 정원일 and 배해영, "공간 슬라이딩 윈도우 집계질의의 정확도 향상을 위한 그리드 해쉬 기반의 부하제한 기법," *한국공간정보시스템학회 논문지*, 제 11권 제 1호, 2009. 3.

김 영 기

2007년 서원대학교 컴퓨터교육과 학사
2008년~현재 인하대학교 정보공학과 석사과정
관심분야 : 데이터 스트림, 공간 데이터베이스, 위치 기반 서비스



신 승 선

2006년 서원대학교 컴퓨터교육과 학사
2008년 인하대학교 컴퓨터공학부 석사
2008년~현재 인하대학교 정보공학과 박사과정
관심분야 : 공간 데이터베이스, 그리드 데이터베이스, 위치 기반 서비스, u-GIS, 데이터 스트림 관리 시스템

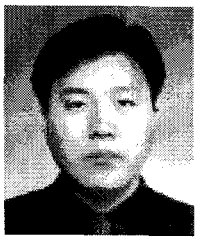
백 성 하

2005년 인하대학교 컴퓨터공학부 학사
2007년 인하대학교 컴퓨터 정보공학과 석사
2007년~현재 하대학교 정보공학과 박사과정
관심분야 : 데이터 스트림, 클러스터, 위치 기반 서비스

이 동 욱

2003년 상지대학교 전자계산공학과 학사
2005년 인하대학교 컴퓨터 정보공학과 석사
2005년~현재 인하대학교 정보공학과 박사과정

관심분야 : 공간데이터웨어하우스, 공간정보관리, 유비쿼터스 환경을 위한 SDBMS



김 경 배

1992년 인하대학교 전자계산공학과 학사
1994년 인하대학교 전자계산공학과 석사
2000년 인하대학교 전자계산공학과 박사
2000년~2004년 한국전자통신연구원 선임연구원
2004년~현재 서원대학교 컴퓨터교육과 조교수
관심분야 : 이동 실시간 데이터베이스, 스토리지 시스템

배 해 영

1974년 인하대학교 응용물리학과 학사
1978년 연세대학교 전자계산학과 석사
1989년 숭실대학교 전자계산학과 박사
1985년 Univ. of Houston 객원 교수
1992년~1994년 인하대학교 전자계산소 소장
1982년~현재 인하대학교 컴퓨터공학부 교수
1999년~현재 지능형 GIS연구센터 센터장
2000년~현재 중국 중경우전대학교 대학원 명예교수
2004년~2006년 인하대학교 정보통신대학원 원장
2006년~2009년 인하대학교 대학원장
관심분야 : 분산 데이터베이스, 공간 데이터베이스, 지리정보 시스템, 멀티미디어 데이터베이스