韓國컴퓨터情報學會 論文誌
第17卷 第1號, 2012. 1.

2012-17-1-1-4

클러스터 환경에서 GeoSensor 스트림 데이터의 집계질의의 정확도 향상을 위한 이중처리 부하제한 기법

지 민 섭*, 이 연*, 김 경 배**, 배 해 영*

A Dual Processing Load Shedding to Improve The Accuracy of Aggregate Queries on Clustering Environment of GeoSensor Data Stream

Min-Sub Ji*, Yeon Lee*, Gyeong-Bae Kim**, Hae-Young Bae*

요 약

인간의 삶을 돕는 유비쿼터스 환경에서 GeoSensor의 다양한 센서 데이터들을 다루는 u-GIS DSMS의 연구가 진행되고 있고 그에 따른 고가용성 서비스를 제공하기 위한 클러스터 시스템이 대두되고 있다. GeoSensor에 의해 수집되는 데이터는 폭발적으로 발생되는 특징을 가지고 있다. 이러한 특징은 서버의 제한된 메모리로 인하여 주어진 메모리를 초과하는 현상과 데이터가 손실되어 질의 정확도가 떨어지는 현상이 발생한다. 이를 해결하기 위해 부하제한 기법들이 활발히 연구되고 있다. 하지만 기존의 기법들은 단일 서버환경에서의 기법들로써 필터링을 통해 부하가 발생한 큐의 튜플들을 특별한 기준에 의해 드롭하는 방식이다. 그렇기 때문에 집계질의와 같은 튜플 삭제에 민감한 질의의 정확도를 만족시키기 어렵다. 본 논문에서는 GeoSensor 스트림 데이터의 클러스터링 환경에서 집계질의의 정확도 향상을 위한 이중처리 부하제한 기법을 제안한다. 본 기법은 두 노드가 고가용성을 위해 이중화 되어있는 스트림 데이터의 환경을 이용한다. 같은 스트림의 데이터를 공유하고 있는 특성을 이용해 두 노드에서 하나의 스트림의 데이터를 나누어 처리한다. 이때 슬라이딩 윈도우 단위로 두 노드 간 스트림 데이터를 동기화한다. 그리고 각 노드에서 처리된 결과를 다시 병합하는 방식이다. 성능평가를 통해 기존 기법들과 달리 튜플의 손실 없이 집계질의의 질의 정확도가 향상된 결과를 얻을 수 있었다.

▸ Keyword : GeoSensor환경, 데이터 스트림, 클러스터 시스템, 부하제한

Abstract

u-GIS DSMSs have been researched to deal with various sensor data from GeoSensors in ubiquitous environment. Also, they has been more important for high availability. The data from

• 제1저자 : 지민섭 • 교신저자 : 이연, 김경배 • 책임저자 : 배해영
• 투고일 : 2011. 08. 11, 심사일 : 2011. 08. 29, 게재확정일 : 2011. 09. 05.
* 인하대학교 컴퓨터정보공학과(Dept. of Computer & Information Engineering, Inha University)
** 서원대학교 컴퓨터교육과(Dept, of Computer Education, Seowon University)

GeoSensors have some characteristics that increase explosively. This characteristic could lead memory overflow and data loss. To solve the problem, various load shedding methods have been researched. Traditional methods drop the overloaded tuples according to a particular criteria in a single server. Tuple deletion sensitive queries such as aggregation is hard to satisfy accuracy. In this paper a dual processing load shedding method is suggested to improve the accuracy of aggregation in clustering environment. In this method two nodes use replicated stream data for high availability. They process a stream in two nodes by using a characteristic they share stream data. Stream data are synchronized between them with a window as a unit. Then, processed results are merged. We gain improved query accuracy without data loss.

▸ Keyword :GeoSensor Network, Data Stream, Cluster System, Load Sheddin

I. 서 론

인간의 삶을 돕는 유비쿼터스 환경에서 국토에 대한 공간 및 위치 정보를 제공하는 u-GIS 공간정보 기술이 미래 유비쿼터스 환경의 핵심 기반 기술로 대두되고 있다[1]. 이런 유비쿼터스 실연의 바탕이 되는 GeoSensor 환경은 u-GIS 공간정보 기술을 필수적인 요소로 삼는다. u-GIS 공간정보 기술은 기존 GIS인 건물, 도로, 지하시설물과 같은 2차원 또는 3차원상의 정적인 지형·지물 정보와 유비쿼터스 환경을 기반으로 시간에 따라 공간적인 위치가 포함된 동적인 GeoSensor 정보의 융합 처리를 요구한다[2].

여기서 GeoSensor는 넓은 지역에 산발적으로 분포하며, 고정성뿐만 아니라 자신의 위치 인식 장치를 이용하여 시간에 따라 장소를 이동할 수 있는 이동성도 가지고 있다. 넓은 지역에서 실시간으로 발생하는 대용량 정보를 처리하기 위해 데이터 스트림 관리시스템(Data Stream Management System)에 대한 연구가 활발히 진행되어 왔다. DSMS는 데이터 스트림을 입력받아 등록된 연속질의를 통해 질의를 처리하여 그 결과를 사용자 또는 다른 시스템이나 응용프로그램으로 제공하는 시스템으로 대표적인 연구로는 Aurora[3], STREAM[4], NiagaraCQ[5], TelegraphCQ[6]등이 존재하고, GeoSensor에 의해 수집되는 데이터는 데이터 스트림의 특성과 GIS의 특성을 동시에 가지고 있으며 국내 지능형국토정보기술혁신 사업과제의 일환으로 개발된 GeoSensor Data Stream Management System[2]이 있다.

DSMS는 최근 인터넷 환경에서 급속히 증대되는 24시간 무정지 서비스 요구되고 있고, 상업적으로 사용되는 시스템에서 서비스 중단은 심각한 손실을 초래할 수 있기 때문에 최근 시스템들은 고가용성(high availability)의 서비스를 제공한다. 고가용성이란 하드웨어, 소프트웨어 혹은 네트워크 등에 문제가 발생하여도 서비스를 지속적으로 제공할 수 있도록 해주는 기술을 말한다[7]. DSMS와 같은 실시간 데이터베이스 응용 시스템 분야나 임무 결정적(mission critical)인 응용 시스템을 다루는 산업 분야에서는 데이터의 성능이나 일관성보다 시스템의 가용성을 더욱 중요시 하는 경우가 많다[8]. 이와 같은 고가용성을 얻기 위하여 오류-극복(fail-over) 구조의 기본적인 방법인 이중화(replication) 기법은 DSMS의 클러스터 기법으로 가장 일반적이며 널리 사용되는 방식이다. 여러 노드로 구성된 클러스터 DSMS들 중 두 노드씩 쌍을 이루어 처리하고 있는 모든 데이터를 이중화 해두어 오류 발생 시 데이터 복구를 할 수 있는 방식이다.

그리고 GeoSensor에 의해 수집되는 데이터는 실시간으로 획득되며 수집되는 양은 항상 유동적이고 크기 역시 가변적이다. 또한 다양한 이벤트에 의해 폭발적으로 데이터가 발생되는 특징을 가지고 있다. 이러한 특징은 서버의 제한된 메모리로 인하여 주어진 메모리를 초과하는 현상과 데이터가 손실되거나 유실되는 현상을 발생시킨다. 이런 현상은 결과적으로 질의 처리 결과의 정확도를 감소시키는 문제점을 야기한다. 그러나 방재시스템, 교통관제시스템, 위치기반응용시스템 등과 같은 응용들은 DBMS에 등록된 질의를 통해 정확한 결과값을 전달 받길 기대한다[9]. 따라서 부하를 제한하기 위한 다양한 기법들의 연구가 활발히 진행되고 있다[9,10,11,12]. 단일 서버 환경에서의 부하제한 기법들에는 랜덤부하제한 기법, 의미적 부하제한 기법, 샘플링 기법 등의 방법이 존재한다[12]. 이는 모두 필터링 방식으로써 부하가 발생한 큐의 튜플들을 특별한 기준에 의해 드롭하는 방식이다. 그렇기 때문에 과부하 현상에서 어느 정도의 질의 정확도는 상승하지만 튜플들을 드롭하기 때문에 COUNT연산이나 SUM연산 등의 집계 질의를 수행하고자 할 때 정확한 질의

결과를 만족시키기 힘들다. 그리고 특정 중요도에 따라 필터링을 하기 때문에 추가적인 연산이 발생하는 특징이 있다.

본 논문에서는, GeoSensor환경에서 u-GIS DSMS인 GeoStream Server(GSS)의 클러스터 환경에서 집계 질의 결과의 정확도를 높이기 위해 이중처리 부하분산 기법을 제안한다. 부하제한을 위한 이중화가 아닌 고가용성을 위한 이중화 환경에서의 부하제한 기법을 제안한다. 제안 기법은 고가용성을 얻기 위해 사용하는 기본적인 방법인 이중화 기법의 환경을 이용한다. 즉 부하가 발생한 연산의 스트림은 페어노드에도 동일하게 가지고 있기 때문에 부하가 발생한 노드의 페어노드에서 동일한 연산을 처리 할 수 있다. 그리고 슬라이딩 윈도우 단위로 체크 포인트를 생성해 두 노드에서 번갈아 가며 처리해 중복적인 데이터 연산을 피한다. 이때 두 가지 슬라이딩 윈도우 종류에 따른 특성을 고려해서 체크 포인트를 나눈다. 그리고 시간 값을 가지고 있는 결과 튜플들의 특징에 따라 해당 질의가 등록된 노드로 결과를 보내서 병합하여 질의 결과를 만들어 낸다. 병합을 위해 노드 간 통신비용문제가 존재하지만 집계질의의 특성상 결과 데이터의 양이 굉장히 적은 특징을 가지고 있기 때문에 추가적인 통신비용의 오버헤드가 적다.

이렇게 함으로써 부하가 발생한 스트림을 처리하는 연산은 반으로 줄어든 튜플만 처리하게 되므로 입력 스트림을 필터링 할 필요가 없었고 기존 필터링 부하 제한 기법보다 질의 결과의 정확도를 높일 수 있었다.

본 논문의 구성은 다음과 같다. 2장은 관련연구로 GeoSensor스트림 데이터의 클러스터링 환경과 기존의 부하제한 기법을 소개한다. 3장에서는 본 논문의 제안 기법인 이중화 스트림 데이터를 이용한 집계질의의 부하제한 기법을 대하여 전체 구조와 슬라이딩 윈도우별로 수행되는 두 가지 방식을 각각 설명한다. 4장에서는 제안 기법에 대한 성능평가를 수행하며, 마지막 5장에서는 결론 및 향후 연구를 기술한다.

II. 관련 연구

2.1 GeoSensor스트림데이터의 클러스터링 환경

유비쿼터스 시대를 위해 새롭게 요구되는 u-GIS 환경은 데이터 스트림 처리 시스템(DSMS: Data Stream Management System)과 지리정보 시스템(GIS : Geography Information System)이 결합된 플랫폼인 u-GIS DSMS를 요구한다. 특히 이 환경은 GeoSensor를 비롯해 GES(GeoSensor Edge Server), GSS(GeoStream Server)로 구성된다[2]. 각각은 다음과 같은 역할과 관계로 형성된다.

GES는 GeoSensor로 구성된 네트워크와 연동하여 다양한 GeoSensor를 접근하고 제어하거나 센싱 정보를 수집하는 서버이다. 다수의 GES는 공간 질의 및 다양한 연산을 수행하는 하나의 GSS와 연동된다. GSS는 GeoSensor Network 또는 GES로부터 입력되는 스트림 정보에 대하여 다양한 연산자를 실행하여 의미 있는 정보를 추출하거나 이벤트를 생성하고, 이를 출력 스트림으로 제공하는 서버이다.

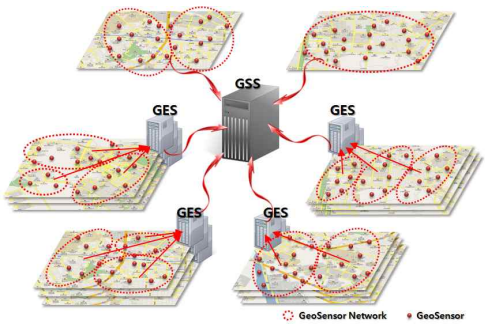

그림 1. GeoSensor, GES, GSS간의 관계
Fig. 1. Relationship among GeoSensor, GES, GSS

[그림 1]은 u-GIS환경에서 GeoSensor, GES, GSS 간의 관계를 나타내는 환경도이다. 이 흐름에서 볼 수 있듯이 GeoSensor, GES의 관계는 N:1 성립되며, GES와 GSS간의 관계 역시 N:1로 성립된다.

여러 노드로 구성된 클러스터 DSMS들 중 두 노드씩 쌍을 이루어 처리하고 있는 모든 데이터를 이중화 해두어 오류 발생 시 데이터 복구를 할 수 있는 방식이다. DSMS는 실시간 데이터를 빠르게 처리해야 하는 특성상 데이터를 스토리지에 저장하지 않고 메모리상에서 처리 및 관리를 하기 때문에 메모리상에 존재하는 스트림 큐의 데이터를 이중화해야 오류 극복시 데이터의 손실을 최소로 줄일 수 있다.

[그림 2]는 고가용성을 위해 클러스터로 확장된 GSS의 이중화 방식의 전체 구조도이다. 클러스터로 확장된 GSS는 여러 노드를 가지고 있고 각각의 노드들은 두 개의 노드가 하나의 쌍을 이루어 이중화 방식을 통해 처리중인 스트림과 질의들을 동기화한다. 노드i 와 노드j 는 페어노드로 쌍을 이루고 있다. 그림에서 RO는 Running Operation으로 현재 수행중인 질의를 말한다. SO는 Stand by Operation으로써 페어노드 RO의 데이터를 이중화하고 있고 실제로 동작하지 않으며 오류극복을 위해 대기상태의 질의이다. 만약 노드i 가 정지되었을 때 노드j 의 SO들이 동작한다. SO는 노드i 의

RO에서 사용되는 스트림 큐의 데이터를 미리 동기화한 상태이기 때문에 데이터 손실 없이 기존의 처리중인 데이터들까지도 복구가 가능하다.

클러스터 GSS환경에서 고가용성을 위해 이중화 되어 있는 스트림 데이터는 기존 단일 서버 환경의 DSMS에서는 존재 하지 않았던 요소이다. 이는 부하제한이나 질의처리 등에서 보다 효율적으로 사용될 수 있는 요소로 작용한다. 따라서 클러스터환경에서는 단일 서버의 부하제한 기법보다는 데이터 이중화의 요소를 고려한 부하제한 기법이 요구된다.

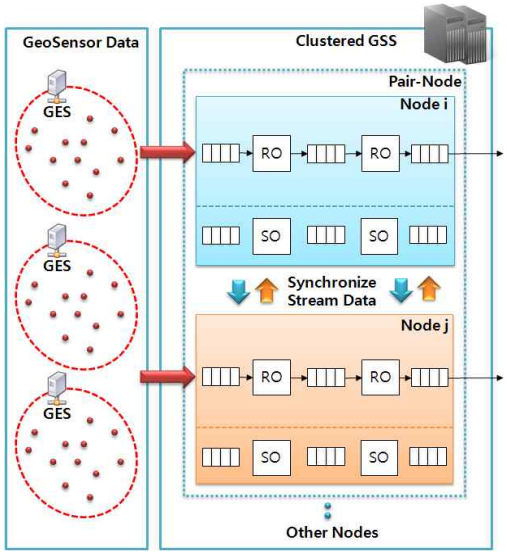

그림 2. 클러스터 GSS의 이중화
Fig. 2. Replication in the clustered GSS

2.2 단일 서버환경의 부하제한 기법

GeoSensor는 산발적으로 분포하고 수집되는 데이터는 실시간으로 획득되며 수집되는 양은 항상 유동적이고 크기 역시 매우 가변적이다. 이와 같은 특징 때문에 데이터를 최종적으로 수집하여 질의를 수행하는 GSS는 데이터 스트림의 저장 공간이 초과되는 경우가 발생하여 질의 정확도가 떨어지는 현상이 나타난다. 집계질의의 경우 이런 과부하 현상에 의해 발생하는 질의 정확도문제가 치명적으로 작용한다. 이런 문제를 해결하기 위하여 DSMS에서는 부하제한 기법을 이용한다. 부하제한 기법은 크게 랜덤부하제한 기법과 의미적 부하제한 기법, 샘플링 기법으로 분류된다. 기존에 연구된 기법들은 스트림 큐에 과부하가 발생할 경우 즉각적으로 부하제한을 수행시키는 방식을 따른다.

랜덤부하제한 기법은 DSMS에 등록된 질의나 데이터에 대해 통합적이거나 개별적인 유용도를 고려하지 않고 과부하가 발생한 스트림 큐의 데이터를 무작위로 선택하여 삭제한다. 따라서 복잡한 연산 과정을 생략하여 부하를 제한하는 연산 수행 속도가 여느 기법에 비하여 빠르며, 구현 역시 간단하다. 그러나 랜덤부하제한 기법은 질의 및 데이터에 대한 유용도를 고려하지 않기 때문에, 사용자가 중요하게 여기는 질의에 대해 높은 정확도를 보장하지 못할 수 있다[9].

의미적 부하제한 기법은 랜덤부하제한 기법이 갖는 정확도 저하 현상을 해결 및 향상시키기 위한 연구의 산출물로, 등록된 질의나 데이터의 유용도를 분석하여 우선순위를 부여하고 부여된 우선순위를 통해 데이터의 삭제 여부를 판단한다. 이 기법의 수행 방식은, 과부하가 발생한 스트림 큐의 데이터의 특정 값을 토대로 우선순위를 결정하고, 부여 받은 우선순위를 분석한다. 이때 우선순위가 낮은 질의를 처리하기 위한 데이터나 데이터 자체의 우선순위가 낮을 경우 삭제하여, 우선순위가 높은 질의를 처리하기 위한 데이터나 우선순위가 높은 데이터가 처리될 수 있도록 한다. 이 기법은 우선순위를 이용하여 사용자에게 QoS를 보장하면서, 질의 결과의 정확도를 향상시켰다. 특히, 의미 있는 질의나 데이터를 선별하기 위한 기준 설정, 삭제할 데이터의 양에 따라 정확도가 크게 달라지기 때문에 우선순위 부여 방법에 대한 많은 연구가 활발히 진행되고 있다[10].

샘플링 기법은 통계적인 수치나 비율을 가지고 필요 데이터를 감소시키는 것으로, 질의의 정확도를 최대한 보장하고 에러율을 최대한 균등하게 분산시킨다. 이 기법 역시 샘플링 비율을 설정하기 위한 기준과 그 비율에 따라 질의 결과의 정확도가 달라지기 때문에 샘플링 비율을 산정하기 위한 많은 연구가 진행 중이다. 예를 들면, Brian의 샘플링 기법은 연산마다 샘플링 비율을 가지고 있으며, 그 비율만큼 데이터를 뽑는다. 샘플링 비율을 산정하기 위하여 데이터 입력 속도와 튜플 당 평균 처리 속도 등을 고려한다[11,12].

단일 서버환경의 필터링 부하제한 기법들은 부하가 발생한 큐의 튜플들을 일정 기준에 따라 삭제하는 필터링 기법을 사용하여 질의 정확도를 높이는 방법을 사용한다. 하지만 COUNT연산이나 SUM연산 등의 집계 질의를 수행하고자 할 때 해당 연산들은 입력 튜플들의 삭제 여부는 굉장히 중요한 영향을 미치기 때문에 정확한 질의 결과를 만족시키기 힘들다. 그리고 특정 중요도에 따라 필터링을 하기 때문에 추가적인 연산의 오버헤드가 발생하는 단점이 존재한다. 따라서 클러스터 GSS환경에서 적합한 부하제한 기법의 연구가 필요하다.

III. 이중화 스트림 데이터를 이용한

집계질의의 부하제한 기법

본 장에서는 공간 질의를 비롯하여 비공간 질의를 처리하고, 동적인 센서 데이터와 정적인 데이터 간의 융합 처리하는 u-GIS DSMS인 GeoStream Server(GSS)의 클러스터 환경에서 수행하는 집계질의에 대한 부하제한 기법을 설명한다.

[그림 3]은 이중처리 부하제한 기법 구조도이다. GSS에 등록된 질의 중에서 부하가 예상되거나 발생한 입력 스트림을 가지는 집계연산에 대해 스트림을 동기화하고 있는 페어노드에 분산하여 처리한다. 노드i 에 있는 수행 질의 중 ROk와 ROg의 연산으로 이루어진 질의가 있다. 해당질의는 고가용성을 위해 노드j 에 이중화되어 있는 상태이다. 이 때 연산 ROk에 과부하가 발생했거나 예상되었을 때 페어노드에서 가지고 있는 SOk와 입력 스트림이 동작하게 된다. 이중화를 통해 ROk와 SOk의 입력 스트림의 데이터는 동기화 되어 있는 상태이기 때문에 추가적인 데이터 전송이나 작업이 요구되지 않는다. 그리고 노드i 의 ROg 연산에는 과부하가 발생하지 않았기 때문에 노드 j의 SOg는 동작하지 않고 이중화만 유지하고 SOk만 동작한다. 이 때, ROk와 SOk는 같은 입력 스트림 데이터를 가지고 있지만 두 연산이 입력스트림의 모든 데이터를 처리하는 것이 아니라 각각의 연산은 중복되지 않는 데이터를 따로 처리해야 한다. 이때 각각의 연산은 페어노드에 분산된 연산이 어떤 데이터를 사용하고 있는지를 알아야 한다. 본 논문에서는 슬라이딩 윈도우의 체크 포인트를 사용해 각각 다른 데이터를 처리한다. 질의분산을 위한 슬라이딩 윈도우의 체크포인트 방법은 다음절에서 자세히 설명할 것이다. 그리고 SOk에서 처리된 결과는 바로 노드i 로 다시 보내져 ROk의 결과 값과 합병 해 ROg의 입력스트림으로 보내진다. 이때 노드 간에 결과를 전송해야하기 때문에 추가적인 통신비용이 발생한다. 하지만 집계질의의 특성상 슬라이딩 윈도우 하나에 해당하는 평균값이나 카운트 값의 튜플 하나만 결과 값으로 나오기 때문에 추가적인 통신비용에 대한 부담이 적다. 이에 대한 자세한 설명은 뒷 절에서 설명하고 있다.

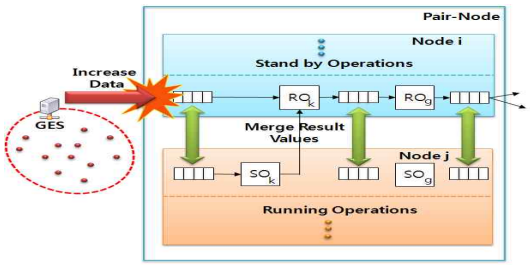

그림 3. 이중처리 부하제한 기법 구조도
Fig. 3. Dual processing load shedding Architecture

3.1 슬라이딩 윈도우 타입에 따른 분산

DSMS가 처리하는 스트림 데이터는 처음과 끝이 존재하지 않고 연속적으로 들어오는 대용량의 데이터이다. 하지만 조인(Join)연산이나 집계(Aggregate) 연산과 같이 blocking연산의 경우 데이터의 끝이 정해져 있어야만 가능한 연산이 있다. 그래서 DSMS에서는 주로 슬라이딩 윈도우 방식을 사용한다. 슬라이딩 윈도우는 일정시간 또는 지정된 개수의 튜플로 스트림 데이터를 분할하여 질의 처리에 이용한다. 그리고 슬라이딩 윈도우를 이동하면서 질의 처리를 계속적으로 진행한다.

따라서 DSMS가 슬라이딩 윈도우 단위로 입력 데이터를 사용하는 특징을 이용하여 분산된 질의의 입력 큐를 슬라이딩 윈도우 단위로 나누는 것이 가장 효과적이다. [그림 4]는 연산을 이중 처리하게 되는 각각의 페어노드에서 가지고 있는 동일한 입력 큐에서 중복 없이 각각의 연산에서 데이터를 처리하는 방법을 설명하고 있다. 노드i 의 RO연산에 대해서 과부하가 발생했고 해당 질의가 페어노드인 노드j 로 분산되어 처리하는 과정을 설명한다. 노드i 의 RO는 S1의 데이터를 처리한다. 그 다음에 처리해야할 S2는 드롭시켜 무시하고 S3의 데이터를 처리한다. 이렇게 되면 슬라이딩 윈도우 2개 단위의 큐가 비어 있게 되므로 입력되는 스트림 데이터를 부하 없이 저장할 공간이 확보하게 된다. 계속적으로 S4를 무시하고 다음 윈도우를 처리함으로써 계속적으로 과부하에 대비할 수 있다. 이렇게 드롭된 데이터는 페어노드인 노드j 에서 처리 한다. 노드j 의 SO는 노드i 의 RO에서 드롭된 슬라이딩 윈도우들을 처리한다. SO에서는 S1을 드롭하고 S2를 바로 처리한다. 그리고 S3을 무시하고 S4를 처리한다. 이렇게 슬라이딩 단위로 두 개의 노드에서 번갈아가면서 처리하게 되면 연산을 처리하고 버퍼를 비우는 시간이 2배로 빨라지게 된다. 따라서 과부하에 증가된 데이터를 받는 버퍼의 공간이 많이 확보될 수 있다.

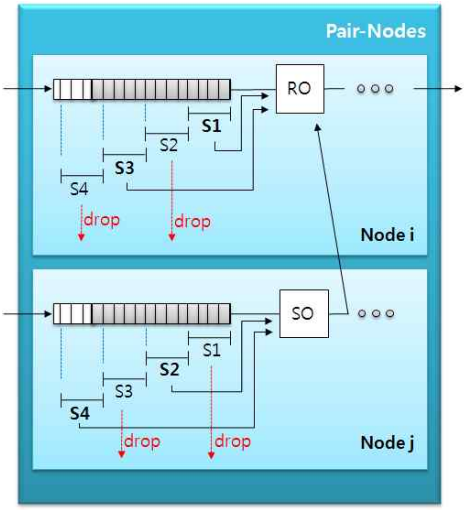

그림 4. 이중처리 부하제한 기법의
슬라이딩 윈도우 단위 처리
Fig. 4. Sliding window unit processing
for dual processing load shedding

슬라이딩 윈도우 단위로 입력 데이터를 나눠 처리하기 위해서는 각 노드들은 각자가 어떤 윈도우를 처리했는지에 대한 정보를 공유해야 한다. 공유에 있어서 고려해야 할 것은 슬라이딩 윈도우 타입이다. 슬라이딩 윈도우는 시간을 기반으로 윈도우 크기를 결정하는 시간 기반 슬라이딩 윈도우와 튜플을 기반으로 윈도우 크기를 결정하는 튜플 기반 슬라이딩 윈도우로 나뉜다.

튜플 기반 슬라이딩 윈도우는 윈도우의 크기를 튜플 수로 설정한다. 따라서 윈도우 크기가 항상 미리 설정한 튜플의 수로 고정적인 특성이 있다. 시간 기반 슬라이딩 윈도우는 설정한 시간에 포함되는 모든 튜플들을 윈도우 크기로 설정한다. 따라서 윈도우 크기는 가변적으로 변하는 특징이 존재한다. 튜플 기반 슬라이딩 윈도우로 설정한 큐에서의 부하가 발생하면 고정적인 윈도우 크기 때문에 연산 횟수가 늘어나게 된다. 하지만 시간 기반 슬라이딩 윈도우는 시간에 따라 윈도우 크기가 결정되기 때문에 연산 횟수는 같지만 하나의 윈도우 크기가 굉장히 늘어나게 된다. 따라서 질의분산 처리에 사용되는 입력큐의 처리 단위를 슬라이딩 윈도우 타입에 따라 다르게 처리해야 할 필요가 있다.

3.1.1 튜플기반 슬라이딩 윈도우

위에서 설명했듯이 튜플기반 슬라이딩 윈도우의 윈도우 크기는 고정적이다. 따라서 윈도우 크기가 가변적인 시간기반 슬라이딩 윈도우 보다 간단하게 처리해야 할 입력 데이터를 나눌 수 있다. 먼저 과부하가 발생하고 질의분산이 이루어 진 시점에서 Primary 노드에서 처리되고 있는 윈도우부터 카운팅을 한다. 그리고 페어노드에서 윈도우로 설정할 입력 큐의 포인트를 전송한다. 이렇게 되면 두 개의 노드에서 처리해야 입력 큐의 데이터를 나누는 슬라이딩 윈도우의 시작점을 확실히 나눈 상태가 된다. 이 후 부터는 고정적인 윈도우 값을 가지고 있기 때문에 고정 윈도우 크기만큼 건너뛰면서 윈도우를 설정한다.

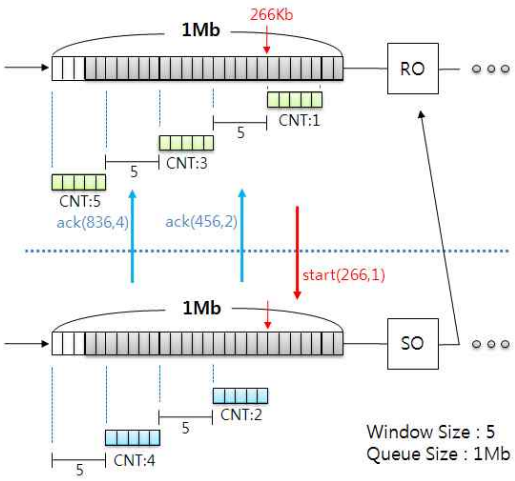

그림 5. 튜플기반 슬라이딩 윈도우의 처리단위 동기화
Fig. 5. Process unit synchronization of tuple based
sliding window

[그림 5]는 튜플기반 슬라이딩 윈도우의 처리단위 동기화이다. 상위노드가 질의가 등록된 Primary노드이고 하위노드가 페어노드이다. 가장 먼저 질의분산이 시작될 때 처리하고 있는 윈도우에 CNT1을 부여한다. 그리고 Primary노드는 페어노드에게 현재 처리하고 있는 윈도우의 카운트(CNT1)와 페어노드가 가장 먼저 윈도우를 시작할 입력 큐의 포인트(266)를 전송한다. 페어노드는 전송받은 포인트부터 고정적인 윈도우 크기 5만큼을 윈도우 크기로 설정하고 전송받은 카운트에 1을 더해 윈도우를 설정한다. 이렇게 되면 두 노드 간 처리해야 할 입력 큐의 윈도우가 확실히 정해진다. 그다음엔 고정적인 튜플 크기인 5만큼 건너뛰며 윈도우를 설정하며 데이터를 처리한다. 그리고 페어노드에서 처리해서 Primary노드로 결과 값이 보내진 윈도우의 카운트 정보와 포인트 정보를 Primary노드로 전송한다. 이는 기존 환경인 고가용성 서비스를 만족하기 위한 방법으로써 페어노드에서 처리 될 입력 데이터를 Primary노드에서 아무 제약 없이 삭제하게 되면

페어노드 오류시 이 데이터를 복구 할 수 없기 때문이다. 때문에 처리응답 메시지를 받았을 때 비로소 입력 큐의 데이터를 삭제할 수 있다.

3.1.2 시간기반 슬라이딩 윈도우

시간기반 슬라이딩 윈도우는 튜플기반과 다르게 가변적인 윈도우 크기를 가진다. 때문에 튜플기반 윈도우의 방식과는 다른 방법을 사용해야 한다.

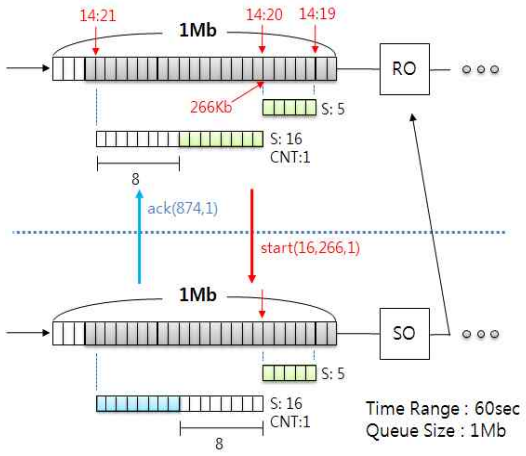

그림 6. 시간기반 슬라이딩 윈도우의 처리단위 동기화
Fig. 6. Process unit synchronization of time based sliding window

[그림 6]은 시간기반 슬라이딩 윈도우의 처리단위 동기화를 설명하고 있다. 그림에서는 큐 사이즈를 1Mb이고 윈도우 크기가 60초로 정해진 질의가 등록되어 있는 상태이다. 14:19분에서 14:20사이의 윈도우는 부하가 발생하지 않아 5의 윈도우 크기를 가지고 있지만 14:20분부터 14:21분까지는 과부하가 발생하여 16개의 윈도우 크기를 가지게 된다. 시간기반 슬라이딩 윈도우의 과부하는 하나의 윈도우 크기가 증가하기 때문에 하나의 윈도우를 분할해 각각의 노드에서 처리하는 방식을 사용한다. 먼저 튜플기반 슬라이딩 윈도우 방식과 동일하게 과부하가 시작된 시점에서 처리하고 있는 윈도우를 카운팅(CNT:1)한다. 그리고 페어노드에게 Primary노드가 처리하고 있는 윈도우의 사이즈(16), 시작 포인트(266)와 카운트(CNT:1)를 전송한다. 페어노드는 전송받은 포인트로 Primary노드와 동일한 윈도우와 카운트를 설정한다. 그리고 전송 받은 윈도우 크기의 상위 1/2만큼을 무시하고 하위 1/2의 윈도우를 처리하게 된다. Primary노드의 질의는 반대로 윈도우의 상위 1/2만큼만 처리한다. 본 방식도 튜플기반 방식과 마찬가지고 페어 노드에서 해당 윈도우를 처리할 때 마다 응답 메시지를 보낸다. 그에 따라 처리 된 윈도우를 확인한 Primary노드는 비로서 해당 윈도우를 비우게 된다. 따라서 본 기법의 사용중에 발생하는 서버 오류에도 고가용성을 보장 할 수 있다.

3.2 분산된 질의 결과의 병합

분산된 질의의 결과를 합병하기 위해서는 노드간의 통신비용이라는 오버헤드가 발생하는 단점이 존재한다. 하지만 집계질의의 특징을 살펴보면 입력 튜플삭제에 민감하게 반응하여 질의 정확도를 결정하는 특징이 있었다. 또 다른 특징 중 하나는 질의 결과가 윈도우마다 하나의 튜플만 생성되는 것이다. 때문에 합병을 위해 페어노드에서 Primary노드로 보내지는 데이터의 양은 굉장히 적은 특징을 가지고 있다. 이는 무시 될 수 있는 수준이다. 이런 집계질의의 특징 때문에 본 논문의 기법의 노드간 통신 오버헤드 단점이 극복될 수 있다.

분산된 질의의 합병결과는 단일 서버 처리의 질의 결과와 같아야 한다. 쉽게 말해서 처리된 슬라이딩 윈도우 순서대로 출력 결과가 나와야 한다. 따라서 슬라이딩 윈도우의 순서에 따라 결과 값을 정렬한다면 단일 서버의 질의 결과와 같은 결과를 만들 수 있다. 그리고 질의 결과의 합병과정도 두 가지 슬라이딩 윈도우 타입에 따라 달라질 수밖에 없다. 튜플기반 슬라이딩 윈도우 방식은 각각의 노드에서 다른 윈도우를 처리했지만 시간기반 슬라이딩 윈도우 방식은 같은 윈도우의 절반씩을 나누어 처리했기 때문에 같은 방식을 사용 할 수 없다.

분산된 질의 결과를 합병하기 위해 기본적인 정보들을 튜플기반과 시간기반 슬라이딩 윈도우의 처리 단위 동기화 작업 모두([그림5,6])에서 미리 만들어 두었다. 그것은 질의 분산이 되는 시점에서 윈도우에 카운팅을 한 것이다. 따라서 카운팅의 순서대로 질의 결과를 합병하면 단일 질의에서 처리된 것과 같은 결과를 얻을 수 있다. 먼저 튜플기반 슬라이딩 윈도우의 합병 과정을 설명하면 Primary노드의 연산에서 카운트1을 가지고 있다. 그리고 카운트 3,5,7... 순으로 가지고 있을 것이다. 반대로 페어노드는 카운트 2,4,6... 순으로 가지고 있다. 따라서 Primary노드의 시놉시스()에서 결과들을 종합한 후에 윈도우 카운트에 따라서 순차적으로 출력 스트림에 결과를 전송한다. 시간기반 슬라이딩 윈도우의 합병은 카운트가 같은 윈도우를 나눠서 연산하는 특성에 따라서 추가적인 연산이 한번 더 들어간다. 마찬가지로 페어노드의 질의 결과를 Primary노드의 시놉시스에 저장한다. 그리고 카운트가 같은 결과들을 재연산하는 작업을 한 후에 출력 스트림에 전송한다. 예를 들어 60초 마다 특정 센서의 온도의 평균을 구하는 질의가 등록이 되어 있고 해당질의가 분산이 되어있다고

가정하자. 60초에 해당하는 튜플들이 하나의 윈도우가 설정되고 약 1~30초 해당하는 평균값은 Primary노드에서 처리되고 31~60초에 해당하는 평균값은 페어노드에서 처리될 것이다. 이 두 결과의 평균이 시놉시스에 저장되어 있고 출력 스트림으로 전송하기 전에 이 두 값의 평균을 구해서 전송하는 것이다.

이렇게 각 노드에서 처리하는 윈도우를 카운팅 함에 따라 단일 서버에서 처리되는 질의의 결과 값과 같은 순서로 결과를 만들 수 있게 된다. 따라서 과부하에 따른 튜플 손실로 발생하는 질의 정확도 문제를 보완할 수 있다.

IV. 성능평가

4.1 평가환경

본 실험은 Fedora Core 12를 기반으로 수행했으며, 시스템 환경은 Intel(R) Pentium(R) 4 CPU 3.00GHz이고, 메인 메모리는 4GB인 서버 두 대의 클러스터 환경이다. GES와 GSS는 Java로 구현했으며, GeoSensor에서 발생하는 데이터 셋은 센서 시뮬레이터(Sensor Simulator)으로 대체하였다. 센서 시뮬레이터는 제안 기법의 성능을 평가하기 위하여 개발된 어플리케이션으로 시스템 관리 툴의 일종이다. 센서 시뮬레이터는 고정성·이동성 GeoSensor 선택하면, 데이터의 사이즈 및 발생 주기 등 몇 가지 옵션을 설정할 수 있다. GIS 데이터는 TIGER/Line 2007 데이터로, Oracle에 구축하여 본 실험에 사용한다.

표 1. 실험환경 기본 속성 값
Table 1. Evaluation Environment Property Values

속성	값
서버 할당 메모리	512 MBytes
스트림 큐 사이즈	5 MBytes
스트림 큐 생성 수	30개
등록된 질의	30개
시공간 데이터 스트림 사이즈	72 Bytes

4.2 성능평가

본 논문에서 제안하는 기법의 우수성을 증명하기 위해 기존의 부하제한기법인 랜덤부하제한 기법과 의미적 부하제한 기법, 샘플링 기법과 제안된 기법인 질의 분산기법을 비교한다. 먼저 각각의 부하제한 기법들의 질의 결과 정확도를 측적, 비교한다. 그리고 부하제한기법들을 사용 할 때의 질의 처리 속도를 비교한다.

4.2.1 질의 결과 정확도 비교

제한 기법을 사용 했을 때 질의 정확도를 측정한 것이다. GES가 수집해 GSS에 보내진 데이터를 100%의 기준으로 설정하고 부하 제한이 없는 상황의 결과와 비교한다. 기존 기법인 랜덤부하제한 기법과 의미적 부하제한 기법, 샘플링 기법과 제안된 기법인 질의 분산기법을 데이터 발생 수를 높이며 과부하 상황을 만들며 실험하였다.

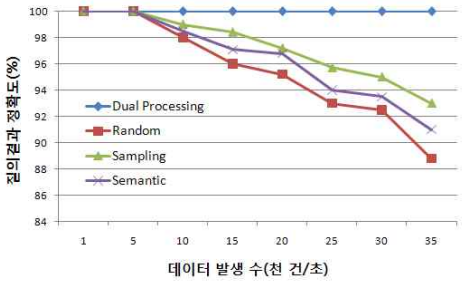

그림 7. 질의 정확도 비교
Fig. 7. Comparison of query accuracy

[그림 7]은 과부하 상황에서의 질의 정확도를 비교한 것이다. 그리고 데이터 발생 수가 5천건 이상일 때 부하가 발생하도록 하였다. 이 상황에서 랜덤부하제한 기법은 튜플들을 기준없이 랜덤으로 드롭함에 따라 가장 정확도가 떨어졌다. 의미적 부하제한 기법과 샘플링 기법은 듀플들을 기준에 따라 드롭하기 때문에 랜덤 부하제한 기법보다 높은 정확도를 가진다. 하지만 제한기법 외의 부하제한 기법들은 질의 처리에 필요한 튜플들을 삭제하기 때문에 어느 정도의 질의 정확도 손실을 가져왔다. 하지만 제안기법은 드롭하는 튜플이 하나도 없기 때문에 질의 정확도를 100%로 유지 할 수 있다.

4.2.2 질의처리 속도 비교

모든 부하제한기법들은 질의 정확도를 위해 약간의 추가적인 연산이 필요하다. 때문에 부하제한 기법들의 질의처리 속도 또한 중요한 요소이다. 이번 실험에서도 4.2.1과 같이 기존의 세 가지 기법과 제안기법의 질의처리 속도를 비교했다. 비교 방법은 부하제한을 사용하지 않는 경우의 질의처리 속도를 100%로 하였을 때 기존 기법들과 제안기법들이 질의처리에 걸리는 속도를 비율로 분석하였다.

[그림 8]은 과부하 상황에서 질의 처리 속도를 비교한 것이다. 랜덤부하제한 기법은 듀플들을 특별한 기준 없이 드롭시키기 때문에 그에 따른 부가적인 연산이 적다. 때문에 높은

처리 속도 성능을 나타냈다. 그 외의 의미적 부하제한 기법과 샘플링 기법들은 랜덤부하제한 보다는 추가적인 연산들이 많기 때문에 랜덤 부하제한기법보다 낮은 처리 속도를 보였다. 그리고 제안기법은 튜플들을 모두 확인하지 두 개의 노드에서 처리하고 있기 때문에 랜덤 부하제한 기법과 비슷한 수준의 질의 처리속도를 보였다. 따라서 제안 기법은 기존의 기법들과 유사하거나 높은 질의 처리속도를 가지고 있고 질의 정확도를 크게 상승시키는 결과를 보였다.

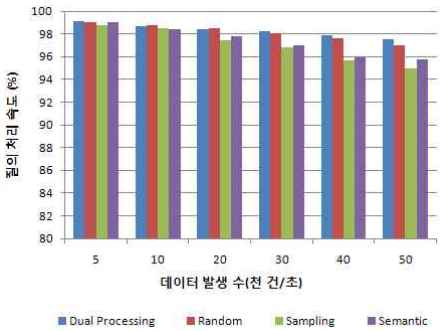

그림 8. 질의처리 속도 비교
Fig. 8. Query processing speed comparison

V. 결 론

본 논문에서는 GeoSensor환경에서 u-GIS DSMS의 클러스터 환경에서 집계 질의 결과의 정확도를 높이기 위해 질의 분산 기법을 제안하였다. 제안 기법은 고가용성을 위한 스트림 데이터의 이중화 방식을 이용하여 두 개의 노드에서 부하가 발생한 질의를 분산해서 처리한 후 질의 결과를 합병해 과부하 발생 시 집계질의의 질의정확도를 향상 시켰다. 또한 질의 처리 속도도 기존의 부하제한기법과 비슷한 수준과 질의 처리 성능저하에 영향을 주지 않는 속도를 보였다.

향후 연구는 u-GIS DSMS의 클러스터 환경에서 집계질의 뿐만 아니라 다양한 질의 특성에 맞는 부하제한 기법연구가 필요하다. 또한 공간 질의와 비공간 질의를 구분하여 각 질의에 효과적인 부하제한 기법의 연구가 필요하다.

Acknowledgment

본 연구는 건설교통부 첨단도시기술개발사업 - 지능형국토정보기술혁신 사업과제의 연구비지원(07국토정보C05)에 의해 수행되었습니다.

참고문헌

[1] C.H. Lee, K.W. An, M.S. Lee, J.W. Kim, "Trends of u-GIS Spatial Information Technology" Telectronics and Telecommunications Trends, ETRI, 2007.

[2] Won-il Jung, Sung-sun Sin, Sung-ha Baek, Yeon Lee, Hae-young Bae, "GeoSensor Data Stream Processing System for u-GIS Computing" KSISS journal, 11-1, 2009, pp. 9-16.

[3] D.J.Abadi, D. Carney, U. Cetintemel, 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.

[4] A. Arasu and et. al., "STREAM: The Stanford Da-ta Stream Management System" http://dbpubs.standford.edu/pub/2004-20, 2004.

[5] J. Chen, D.J. DeWitt, F. Tian and Y. Wang, "NiagaraCQ: a scalable continous query system for internet databases" Proc. of the ACM SIGMOD International Conference on Management of Data, pp. 379-390, 2000.

[6] F. Reiss and J.M. Hellerstein, "Data triage: an adaptive architecture for load shedding in TelegraphCQ" Proc. of the International Conference on Data Engineering, pp. 155-156, 2005.

[7] Gregory F. Pfister, In search of Clusters, Prentice Hall PTR, 1998.

[8] M. Wiesmann, F. Pedone,A. Shiper, B. Kemme, G. Alonso, "Understanding Replication in Databases and Distributed Systems", Proc. of the 20th International onference on Distributed Computing Systems, 2000.

[9] Yi-Cheng T., Song L., Sunil P., and Bin Y., "Load Shedding in Stream Databases: A Control-Based Approach," In VLDB Conference, 2006, pp. 787-798.

[10] Nesume T., Ugur C., Stan Z., Mitch C., and Michael S., "Load Shedding in a Data Stream Manager," VLDB, 2003, pp. 309-320.
[11] Brian B., Mayur D., and Rajeev M., "Load Shedding for Aggregation Queries over Data Stream," ICDE, 2004, pp. 350-361.
[12] Ho K., Sung-Ha B., Dong-Wook L., Gyoung-Bae K., Hae-Young B., "Load Shedding applying range overlap ratio of spatial query over Data Stream," ASGIS2009, 2009, pp. 49-55.

저 자 소 개

지 민 섭
2010 : 서원대학교 컴퓨터교육과 교육학사
2010~현재 : 인하대학교 컴퓨터정보공학과 공학석사과정
관심분야 : 공간 데이터베이스, 데이터스트림, 클러스터링 시스템, 부하제한
Email : msji@dblab.inha.ac.kr

이 연
2006 : 중국 중경우전대학교 지리정보시스템학과 이학사.
2008 : 인하대학교 컴퓨터정보공학과 공학석사.
현 재 : 인하대학교 컴퓨터정보공학과 박사과정
관심분야 : 공간 데이터베이스, 공간 데이터 웨어하우스, 지리정보 시스템, USN, 스트림 데이터 시스템
Email : leeyeon@dblab.inha.ac.kr

김 경 배
1992 : 인하대학교 전자계산학과 공학사
2000 : 인하대학교 컴퓨터정보공학과 공학박사
2000~2004 : 한국전자통신연구원 선임연구원
2004~현재 : 서원대학교 컴퓨터교육과 조교수
관심분야 : 이동실시간 데이터베이스, 스토리지 시스템, GIS, VOD
Email : gbkim@seowon.ac.kr

배 해 영
1974 : 인하대학교 응용물리학과 공학사.
1978 : 연세대학교 전자계산학과 공학석사.
1989 : 숭실대학교 전자계산학과 공학박사
1985 : Univ. of Houston 객원교수
1992~1994 : 인하대학교 전자계산소장
2004~2006 : 인하대학교 정보통신대학원장
2006~2009 : 인하대학교 대학원장
현 재 : 인하대학교 컴퓨터정보공학과 교수 지능형 GIS연구센터 센터장 중국 중경우전대학교 대학원 명예교수
관심분야 : 데이터베이스, 공간 데이터베이스, 지리정보 시스템, 멀티미디어 데이터베이스
Email : hybae@inha.ac.kr