Journal of the Korea Academia-Industrial cooperation Society
Vol. 13, No. 6 pp. 2713-2719, 2012

http://dx.doi.org/10.5762/KAIS.2012.13.6.2713

분산 공간 데이터 스트림 처리에서 질의 영역의 겹침을 고려한 공간 연산 배치 기법

정원일[1*]
[1]호서대학교 정보보호학과

Spatial Operation Allocation Scheme over Common Query Regions for Distributed Spatial Data Stream Processing

Weonil Chung[1*]
[1]**Dept. of Information Security Engineering, Hoseo University**

요 약 위치를 기반으로 하는 서비스가 다양해짐에 따라 고가용성과 고확장성을 제공하기 위한 분산 데이터 스트림 처리 기법에 대한 연구가 널리 수행되고 있다. 기존 연구는 분산된 노드들에서 부하의 균형을 유지하기 위해 공간 데이터 스트림의 지리적인 특성을 고려하지 않고 있어 공간적으로 인접한 연산을 수행함에 있어 전체 시스템의 부하를 증가시키고 있다. 본 논문에서는 분산 환경의 공간 데이터 스트림을 처리하기 위해 공간 영역의 겹침을 고려한 연산 배치 기법을 제안한다. 제안 기법에서는 인접한 공간 영역을 대상으로 하는 연산을 효율적으로 분리하기 위해 질의 영역이 겹치는 부분의 연산을 우선적으로 동일 노드에 분배하여 중복 영역에 대한 공유의 최대화를 보장한다.

Abstract According to increasing of various location-based services, distributed data stream processing techniques have been widely studied to provide high scalability and availability. In previous researches, in order to balance the load of distributed nodes, the geographic characteristics of spatial data stream are not considered. For this reason, distributed operations for adjacent spatial regions increases the overall system load. We propose a operation allocation scheme considering the characteristics of spatial operations to effectively processing spatial data stream in distributed computing environments. The proposed method presents the efficient share maximizing approach that preferentially distributes spatial operations sharing the common query regions to the same node in order to separate the adjacent spatial operations on overlapped regions.

Key Words : Distributed Processing, Spatial Data Stream, Operation Allocation

1. 서론

최근 다양한 모바일 기기들이 널리 활용됨에 따라 위치 정보를 기반으로 하는 시공간 이동 객체들이 급증하고 있다. 이들 시공간 이동 객체들은 시간과 공간 정보를 포함하는 시공간 데이터 스트림을 생성한다[1-4]. 이러한 시공간 정보를 응용한 서비스들이 증가하면서 공간 연산을 포함하는 공간질의에 대한 요청률 또한 급격하게 증가하고 있다[5-7]. 이와 같은 대용량 실시간 데이터의 양과 공간질의 요청의 처리는 서비스 제공을 위한 시스템에 큰 부하를 발생시키므로 이러한 부하를 분산시켜 처리할 수 있는 기법들이 연구되고 있다[8].

분산 환경에서 데이터 스트림 처리는 다수의 연산을 포함하여 처리할 수 있는 노드들로 구성되고, 각 노드들은 질의 네트워크를 구성하는 연산들을 분산하여 처리한다. 이러한 분산 데이터 스트림 처리 시스템은 노드의 평균 부하율을 낮추고 데이터 스트림 처리 성능 향상을 위해서는 노드들 간에 연산자를 배분하는 연산 분배 알고

이 논문은 2011년도 호서대학교 재원으로 학술연구비 지원을 받아 수행된 연구임(2011-0250)
*Corresponding Author : Weonil Chung
Tel: +82-41-540-5984 email: wnchung@hoseo.edu
접수일 12년 04월 04일 수정일 (1차 12년 04월 30일, 2차 12년 05월 09일) 게재확정일 12년 06월 07일

리즘이 요구된다[9-10]. 기존의 분산 스트림 처리 시스템을 위한 연산자 분배 알고리즘으로는 pair-wise알고리즘이 있다[9-10]. pair-wise는 노드들의 평균 부하율에 따라 노드들을 정렬해 노드 부하율이 제일 큰 노드와 제일 작은 노드의 순으로 노드의 쌍을 만들고 연산을 서로 분배한다. pair-wise는 오직 데이터 스트림의 유입률만을 고려해 노드 쌍들 사이에 연산을 무조건 교환하므로 시공간 데이터 스트림을 처리할 때 연산 처리의 결과를 공유할 수 있는 공간적으로 인접한 연산이 분산되어 메모리 사용량과 연산이 늘어나 전체 시스템의 부하가 증가할 수 있는 문제가 있다.

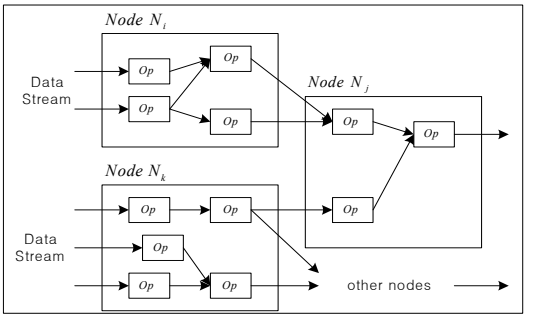
이에 본 논문에서는 시공간 데이터 스트림을 처리하는 분산 데이터 스트림 처리를 위해 공간 연산의 중첩 영역을 고려한 연산 분배 알고리즘을 제안한다. 제안 기법에서 연산 분배 알고리즘은 노드별로 공간 연산을 분배할 때 각 노드로부터 수집된 부하율 및 노드별로 할당된 공간 연산의 중첩 영역에 대한 정보를 이용한다. 이를 위해 R-색인[12]을 이용해 분산 데이터 스트림 처리 시스템에 등록된 공간 연산들의 질의 영역이 겹치는 부분을 그룹화 하여 해당 정보를 테이블에 저장한다[11]. 이어 노드들의 데이터 유입률과 연산의 처리량을 통해 부하율을 계산한다. 공간 연산이 중첩되는 노드의 결과를 최대한 공유하기 위해 공간 연산을 포함한 노드와 그 결과를 입력으로 받는 노드들을 쌍으로 생성한다. 부하율이 임계값을 넘지 않으면 중첩 영역의 공유를 위해 공간 연산은 우선적으로 기존 노드에 배치시키고 그 외의 연산들을 노드 간에 이동시킨다. 그리고 공간 연산이 포함되지 않은 노드들은 부하율에 따라 정렬시켜 부하율이 제일 큰 노드와 작은 노드의 쌍을 구성하여 연산을 분배한다. 제안 기법은 공간 연산의 중첩영역을 고려하여 연산을 분배하여 중첩된 영역의 결과를 공유함으로써 노드의 부하율을 낮출 수 있고 전체 노드들의 평균 부하율도 낮출 수 있다.

2. 관련 연구

2.1 분산 데이터 스트림 처리

분산 데이터 스트림 처리 시스템(DSMS: Distributed Data Stream Processing System)[8-9]은 다수의 연산을 포함하여 처리할 수 있는 노드들로 구성된다. 각 연산은 DSMS에서 데이터 스트림의 처리 유닛이며 map, filter, join, aggregate와 같은 것들이 있다. 그리고 연산은 데이터 스트림의 입력을 담당하는 입력 큐와 연산 결과를 저장할 출력 큐를 갖는다. 이 연산자들의 체인으로 구성된 연속 질의를 질의 네트워크라 부른다. 분산 데이터 스트림 처리 시스템의 각 노드들은 네트워크를 구성하는 연산들을 노드별로 분산하여 처리한다.

그림 1에서는 3개의 노드들에 걸친 분산 데이터 스트림 처리의 흐름에 대한 예를 나타내고 있다. 각 노드는 다양한 데이터 스트림을 입력으로 연산을 처리하고 그 결과는 다른 노드의 입력으로 사용될 수 있다. N_i, N_j, N_k 노드는 분산 데이터 스트림 처리 시스템의 노드들을 나타내며 노드 N_i, N_k는 노드 N_j의 upstream neighbor 노드, N_j 노드는 N_i, N_k 노드들의 downstream neighbor 노드이다. 다수의 데이터 스트림이 노드들의 입력으로서 들어오고 또한 각 노드들은 다수의 출력 스트림을 갖는다. 그림 1에서 노드 N_j는 자신의 upstream 노드들인 N_i와 N_k에서 입력되는 튜플들을 바탕으로 연산을 처리하고 그 결과를 응용으로 전송한다.



[그림 1] 분산 데이터 스트림 처리
[Fig. 1] Distributed Data Stream Processing

2.2 분산 환경에서의 연산 배치

분산 데이터 스트림 처리 시스템은 전체 노드의 평균 부하율을 낮추며 신속한 데이터 스트림 처리를 위해 노드들 간에 연산자를 배분해 주는 연산자 분배기법이 요구된다. 분산 데이터 스트림 처리 시스템과 같은 푸시 기반 시스템에서는 부하의 변동이 스트림의 유입률에 의해 발생한다. 노드의 부하가 매우 높지 않을 경우라도 노드들은 일시적인 부하의 급증을 경험할 수 있고 데이터 처리의 지연은 부하가 급증하는 동안 심각한 영향을 줄 수 있다. 그러므로 데이터 처리의 지연을 최소화하기 위해 가능한 일시적인 부하의 급증을 피할 수 있는 방법이 필요하다[9].

pair-wise 알고리즘은 노드에 유입되는 데이터 스트림의 입력 비율과 연산 비용의 연산자 매핑 관계를 동적으로 조정한다[9-10]. pair-wise 알고리즘에서 조정자가 주

기적으로 모든 노드들로부터 부하의 통계를 수집하며, 노드들과 연산자들의 부하는 주기적으로 고정 길이의 시간 구간에 걸쳐서 측정된다. 특정 시간 구간 동안에 연산자 o에 대한 데이터 스트림의 유입률이 $\lambda(o)$이고 연산자 o의 평균 처리 시간이 $p(o)$인 경우 특정 시간 구간에서 연산자 o의 부하는 $\lambda(o) \bullet p(o)$이 된다. 그리고 주어진 시간 구간에서 한 노드의 부하는 해당 시간 구간 동안에 노드에 속한 모든 연산자들의 부하의 합으로 정의된다. 그리고 그들의 평균 부하에 따라서 노드들을 정렬한다. 분산 데이터 스트림 처리 시스템의 노드들의 수가 n이라고 가정하면, 정렬된 리스트에서 i번째 노드와 n-i+1번째 노드를 함께 그룹화하여 두 노드를 쌍으로 만든다. 노드 쌍 사이에서 부하의 차이가 미리 정의된 임계값보다 크면, 연산자는 노드간의 평균 부하의 균형을 유지시키기 위해 노드들 사이에서 옮겨진다. 그러나 pair-wise 알고리즘은 노드사이에서의 연산자 분배를 입력되는 데이터 스트림의 유입률만을 고려해서 노드 쌍 간의 연산을 무조건 교환하므로 시공간 데이터 스트림을 처리할 때 연산 처리의 결과를 공유할 수 있는 공간적으로 인접한 연산이 분산되어 메모리 사용량과 연산이 늘어나 전체 시스템의 부하가 증가할 수 있는 문제가 있다.

분산 환경에서 다중 공간 집계 연산의 중복 수행을 감소시키기 위한 자원 공유 기법에서는 부하 분산을 위해 각 노드에 유입되는 데이터스트림의 비율과 연산자 평균 처리시간을 고려하고 있다. 제안 기법에서는 부하 분산의 효율성을 증대시키기 위해 데이터 스트림의 생성 빈도에 대한 변화율과 질의 처리 비용이 공간 연산에 대한 선택도를 추가로 고려한다[11].

3. 공유 질의영역에 대한 공간연산 배치

3.1 시공간 연속 질의

분산 데이터 스트림 처리 환경에서 서로 다른 노드에서 공유되는 질의 영역의 겹침을 최소화하기 위해 이 절에서는 기존의 연속 질의와 공간 연산이 결합된 시공간 연속 질의를 정의한다. 시공간 연속 질의는 기존의 SQL을 확장한 ST-CQL를 사용한다. ST-CQL의 연산은 크게 Compare Filtering과 Spatial Filtering으로 나뉜다. Compare Filtering은 일반적인 비교 연산(>, <, =, >=, <=, <>) 으로 구성되며 결과 값은 boolean으로 반환한다. Spatial Filtering은 CONTAINS 연산을 사용한다.

CONTAINS 연산은 질의 영역에 대한 시공간 객체의 포함 관계를 나타내며 boolean 결과를 반환한다. 또한 CONTAINS 연산의 질의 영역이 이동 객체를 중심으로 구성되었을 경우 이동 질의로 수행된다. ST-CQL의 기본 형태는 아래와 같으며 기존의 SQL-Like한 구문에 질의 처리 결과 출력을 위한 스트림을 지정하는 INTO 구문과 이동 및 고정 공간 질의 영역 정의를 위한 DEFINE 구문, 영역에 대한 연산 지정을 위한 CONTAINS 구문 그리고 질의의 효력 시간을 설정하기 위한 Life Time 구문이 있다.

질의 1은 8시간 동안 13세 이하의 피보호자가 ID가 5인 보호자(Guardian)로부터 가로와 세로가 50인 공간 영역에서 벗어나는 경우를 탐색하는 질의다.

[질의 1] 시공간 연속 질의 예
[Query 1] An Example of Spatio-temporal Continuous Query

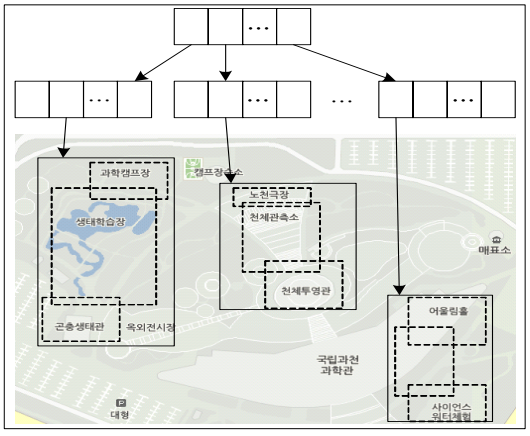
```
INTO OUT_OF_GUARD_5
SELECT MP.ID, MP.NAME
FROM MOVING_PERSON[PERSON_INFO] as MP
WHERE MP.GID = 5 and MP.Age <= 13
DEFINE Guardian(MP.ID = 5, 50, 50)
CONTAINS Guardian
LIFETIME 8h
```

공간 연산은 특정 영역에 대해 집중적으로 발생하며 등록된 두 개 이상의 연속 질의의 공간 범위 검색 영역이 거의 유사하며 미세한 차이를 갖거나 거의 다르지만 약간의 유사성으로 인해 공간 연산의 영역적으로 겹치는 등의 유사한 조건을 갖는 공간 연산을 포함하기 때문에 유사 질의의 등록이 잦아질 수 있다.

3.2 공유 영역 그룹화

공간 연산자의 중첩 영역을 고려해 노드별로 연산을 분배하기 위해 공간 영역을 그룹화하는 R-색인을 이용하여 공간 연산자 간의 중복 영역을 찾아 중복영역의 범위, 중복되는 연산자 ID 정보를 관리한다. R-색인을 이용한 그룹화시에 필요한 연산자 정보는 연속 질의 테이블에서 참조한다.

R-색인은 각 공간을 최소 경계 사각형으로 구성 및 분할하여 그룹화 함으로써 효율적으로 중복영역을 관리할 수 있다. 또한 R-색인 노드의 최대 차수는 3-4 레벨을 유지하며 좌우노드의 균형을 유지하기 때문에 검색시간이 빠른 특징을 갖는다. 그림 2는 R-색인을 이용하여 공간 연산을 그룹화한 예를 보여준다.



[그림 2] 공간 연산 영역의 그룹화
[Fig. 2] A Region Grouping of Spatial Operation

R-색인을 이용하여 그룹화 된 영역에서 각 공간연산의 영역과 중복 영역에 대해 ID를 부여하여 그 정보를 테이블로 관리한다. 해당 테이블의 공간 연산의 중복영역에 대한 정보를 바탕으로 노드간의 연산자 분배를 고려하게 된다.

3.3 분산 노드의 부하 모니터링

분산 데이터 스트림 환경에서 각 노드에 대한 부하 정도를 산정하기 위해서는 시간 변화에 따라 유동적으로 시스템에 입력되는 데이터 스트림의 생성 빈도와 변화율의 계산이 필요하다. 입력 데이터 스트림 D_x에 대해 데이터 스트림 생성 빈도의 변화율 $Freq_v(D_x)$는 아래와 같이 표현할 수 있다.

$$Freq_v(D_x) = Avg(Freq^{p's}(D_x)) + (|D_x^p - D_x^c|) \quad \text{(수식1)}$$

수식 1에서 $Avg(Freq^{p's}(D_x))$는 입력 데이터 스트림에 대한 변화량을 산출에 있어 예측 값의 보정을 위해 과거 입력된 데이터 스트림의 변화량을 평균값으로 계산하여 적용한 보정값이며, $(|D_x^p - D_x^c|)$는 데이터 스트림의 직전 입력량과 현재 입력량에 대한 절대값을 나타내고 있다. 이로부터 입력 데이터 스트림의 생성 빈도 $Freq(D_x)$는 아래와 같이 유도할 수 있다.

$$Freq(D_x) = Freq_v(D_x) + D_x^c \quad \text{(수식2)}$$

수식 2에서 입력 데이터 스트림 D_x에 대한 생성 빈도 $Freq(D_x)$는 D_x의 생성 빈도 변화율 $Freq_v(D_x)$와 현재 입력량 D_x^c의 합으로 유도된다.

입력 데이터 스트림은 분산 데이터 스트림 처리 시스템에서 다중 연속 질의의 연산자에 의해 처리되어 중간 결과로 산출된다. 연산자의 선택도는 입력 데이터 스트림에 대해 연산자에 의해 처리된 중간 결과물의 비율을 나타낸다. 연산자에 입력되는 데이터 스트림도 시간에 따라 변화하며, 연산자 Op_x에 대한 선택도 변화율 $S_v(Op_x)$은 아래와 같다.

$$S_v(Op_x) = |S^p(Op_x) - S^c(Op_x)| + Avg(S^{p's}(Op_x)) \quad \text{(수식3)}$$

수식 3에서 $|S^p(Op_x) - S^c(Op_x)|$는 연산자의 과거 선택도와 현재 선택도의 차이를 나타내며, $Avg(S^{p's}(Op_x))$는 최근의 선택도 변화를 반영하는 변화율의 보정값을 의미한다.

$$S(Op_x) = S_v(Op_x) + S^c(Op_x) \quad \text{(수식4)}$$

수식 4에서는 연산자 Op_x에 대한 선택도 $S(Op_x)$는 해당 연산자에 대한 선택도 변화율과 현재 선택도를 합산하여 유도된다.

노드들에 대한 부하 정도는 데이터 스트림 입력 빈도 $Freq(D_x)$와 연산자에 대한 선택도 $S(Op_x)$를 주기적으로 측정하여 산출한다.

$$Node_{load} = \sum_{i=1}^{n} Freq(D_x) \cdot S(Op_x) \quad \text{(수식5)}$$

수식 5에서 노드 N의 부하율 N_{load}는 노드에 할당된 연산자의 수가 n개일 때, 각 연산자의 선택도과 데이터 스트림 입력 빈도에 대한 곱의 합으로 계산된다.

분산 노드에 대해 산출된 부하율은 전체 노드의 부하율을 관리하는 전역 모니터 모듈로 전송되고 이 부하율 정보를 모니터링하여 그 값이 임계치를 넘어가면 질의 처리기는 spatial pair-wise 기법에 따라 질의 플랜을 다시 구성하고 노드들로 연산을 재분배한다.

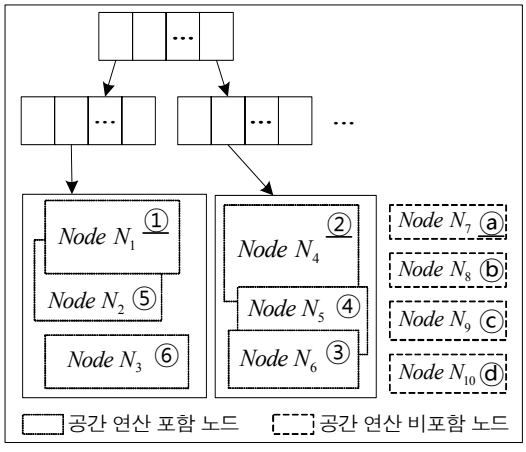
3.4 중첩을 고려한 노드별 연산자 분배

공간 정보를 포함하는 데이터 스트림은 데이터베이스에 저장된 속성 정보와의 연계 처리가 요구되므로 디스크 접근에 대한 접근이 필요하게 된다. 이에 질의 영역이 중첩되는 노드들을 대상으로 하는 연산의 재분배는 해당 노드의 디스크 접근 비용을 감소시킬 뿐 아니라 전체 시스템의 처리율도 향상시키게 된다.

spatial pair-wise 기법은 공간 연산에 중첩되는 영역을 최대한 공유시키기 위하여 R-색인으로 그룹화하여 테이블에 저장하는 공간 연산의 중첩 정보를 사용해 노드 간의 연산 분배를 조절한다. 이때 사용하는 중첩정보 테이블은 해당 공간연산 영역과 중첩되는 공간연산에 대한 ID 리스트 및 공간 연산 개수를 저장한다.

공간 연산이 포함되어 있는 노드들과 공간 연산이 포함되어 있지 않은 노드들로 분류한다. 각 노드들의 부하

율을 산출하고 부하 정도에 따라 정렬한다. 노드의 부하율에 따라 정렬된 순서에서 부하율이 제일 큰 노드와 제일 작은 노드를 쌍으로 만든다. 즉, 정렬된 전체 노드들의 수가 n 이라고 가정하면, 정렬된 리스트에서 i번째 노드와 n-i+1번째 노드를 함께 그룹화하여 그 두 개의 노드를 쌍으로 만든다. 이때, 공간 연산이 포함된 노드들의 쌍을 생성할 경우 공간 연산의 중첩 정보와 노드별 연산자 정보를 바탕으로 쌍을 선정한다. 이러한 과정을 거치면 공간 연산이 포함된 노드들의 쌍 그룹과 공간 연산이 포함되지 않은 노드들의 쌍 그룹으로 나누어진다.



[그림 3] 공유 영역 기반 공간 연산 배치
[Fig. 3] Spatial Operation Allocation based on shared regions

그림 3에서 $Node\,N_1$부터 $Node\,N_6$은 공간 연산이 포함된 노드들이며, $Node\,N_7$부터 $Node\,N_{10}$은 공간 연산이 포함되지 않은 노드들이다. 공간 연산이 포함된 노드들은 높은 부하율부터 ①에서 ⑥까지 정렬 순서가 표기되며, 공간 연산이 포함되지 않은 노드들에 대해서는 부하율이 높은 노드부터 ⓐ에서 ⓓ까지 정렬된다. 그림 3과 같이 $Node\,N_1$과 $Node\,N_4$가 부하의 임계치를 초과한 상황을 가정할 경우, $Node\,N_1$은 공간 연산의 중첩 영역을 고려하여 $Node\,N_6$이 아닌 $Node\,N_5$가 쌍으로 결정되고, $Node\,N_2$도 $Node\,N_5$를 대신하여 $Node\,N_4$가 쌍이 된다. 이로부터 부하 노드의 공간 연산은 쌍으로 선정된 노드로 옮겨지게 된다. 부하가 임계치를 넘어선 $Node\,N_7$은 공간 연산을 포함하지 않으므로 정렬 순서에서 대응하는 $Node\,N_{10}$이 쌍으로 선정되어 연산 분배가 수행된다.

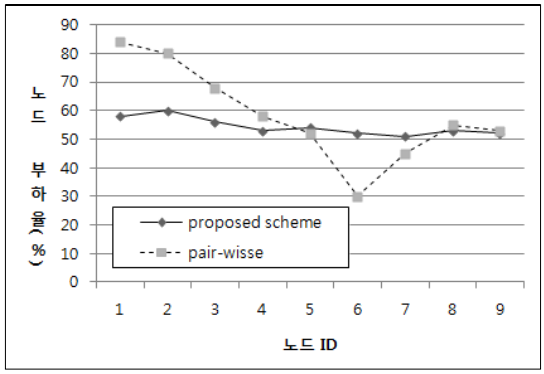
4. 실험

본 실험은 Fedora Core 8.0을 기반으로 수행했으며, 시스템 환경은 Intel(R) Pentium(R) 4 CPU 3.00GHz이고, 메인 메모리는 4GB이다. 분산 노드를 위해 GSS[13]를 활용하였으며, 공간 데이터 스트림을 위한 데이터 집합은 센서 시뮬레이터를 활용하였다. 센서 시뮬레이터는 고정체와 이동체에 대해 데이터의 사이즈 및 발생 주기 등의 옵션을 설정할 수 있다. 지리정보 데이터는 TIGER/Line 2007 데이터로, Oracle에 구축하여 실험에 사용한다[14].

실험을 위한 기본 속성으로는 서버 할당 메모리 512MB, 스트림 큐 사이즈 5MB, 스트림 큐 생성수 15개, 등록된 공간 질의 10개, 등록된 비공간 질의 5개, 시공간 데이터 스트림 크기 36B, 실험 수행 시간 200sec, 스트림 큐의 부하 임계치 3.5MB(70%), 실험군 당 실험 횟수 5회를 설정하였다.

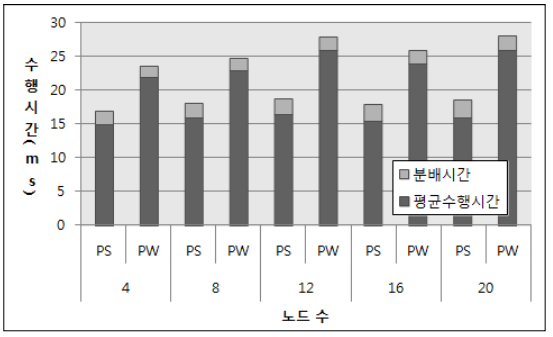
첫 번째 실험은 중첩 영역을 고려하여 공간 연산을 배치하는 제안 기법의 효용성을 검증하기 위해 시공간 데이터 스트림을 입력으로 연산 분배를 수행할 때 pair-wise 기법과 제안 기법간의 노드별 부하율을 비교하였다.

그림 4는 제안 기법과 pair-wise 기법에 대해 연산 분배의 수행에 따른 노드별 부하율을 측정한 결과이다. pair-wise 기법에서는 공간 연산의 중첩 영역을 고려하는 분배를 수행하지 않으므로 노드간의 부하 편차가 두 배까지도 차이가 나타나지만, 제안 기법은 중첩 영역을 고려하여 공간 연산을 분배하므로 노드간의 부하 편차가 9% 이내로 일정하게 나타나고 있다. 이러한 결과는 제안 기법이 pair-wise 기법에 비해 분산 환경에서의 시공간 데이터 스트림 처리에 있어 부하 분산을 효과적으로 수행하고 있음을 의미한다.



[그림 4] 노드 부하율 비교
[Fig. 4] Load Factor Comparison of Nodes

두 번째 실험은 부하 노드간의 연산 분배에 있어 제안 기법에서 추가적으로 요구되는 공간 연산에 중첩되는 영역을 선정하기 위한 과정이 전체 시스템 성능에 미치는 영향을 분석하기 위해 제안 기법과 pair-wise 기법간의 연산 분배에 소요되는 비용을 비교하였다.



[그림 5] 연산 분배 소요 시간 비교
[Fig. 5] Time cost comparison of the operations distribution

그림 5에서는 분산 환경에서 노드 수를 증가시키면서 제안 기법(PS)와 Pair-wise(PW) 기법에서의 연산 분배 시간 및 노드의 수행 시간의 평균을 비교하였다. 연산 분배를 위한 소요 시간은 제안 기법이 Pair-wise 기법에 비해 평균적으로 23%가 더 많은 시간이 소요되었으나, 각 노드에서의 평균 질의 처리 시간은 Pair-wise 기법에 비해 제안 기법이 최소 27%에서 최대 44%의 수행 시간이 절약됨을 알 수 있다. 이러한 결과는 제안 기법에서 영역 겹침을 고려한 연산 분배 비용이 요구되지만 분산 시스템의 성능은 향상됨을 알 수 있다.

5. 결론

본 논문에서는 시공간 데이터 스트림 처리를 위한 분산 데이터 스트림 처리 시스템에서 공간 연산의 특징을 고려한 부하 발생 시의 노드 간의 연산자 분배 기법을 제안하였다.

제안 기법에서의 연산자 분배는 먼저 노드들을 공간 연산이 포함된 노드와 포함되지 않은 노드로 분류하고 각각을 노드의 부하율을 계산하고 정렬해 공간 연산이 포함되지 않은 쌍은 부하율의 평균을 최소화 하는 방식으로 연산자 분배를 진행하고 공간 연산이 포함되어 있는 쌍의 경우에는 R-색인으로 그룹화된 공간 연산의 중첩정보를 토대로 최대한 중첩되는 연산이 같은 노드에 속하도록 연산 분배를 수행하였다.

실험에서는 공간 연산이 포함된 시공간 데이터 스트림의 처리시 분산 데이터 스트림 처리 시스템을 이루는 노드들의 평균 부하율을 효과적으로 제어할 수 있음을 보였다. 그리고 Pair-wise 기법에 비해 제안 기법에서질의 처리를 위해 추가로 요구되는 연산 분배 처리 비용으로 인한 시스템의 성능 저하가 발생하지 않음을 확인하였다.

향후 연구로는 질의 패턴을 다양화하여 연산 분배를 최적화할 수 있는 연구와 함께 분산 데이터스트림 처리 시스템의 고확장성을 제공하기 위한 연구가 필요하다.

References

[1] S. Prabhakar, et al., “Query indexing and velocity constrained indexing: scalable techniques for continuous queries on moving objects”, IEEE Transactions on Computers, Vol. 51, No. 10, pp. 1124-1140, 2002.

[2] Chi-Min Park, et al., “Design and Implementation of a Spatial DSMS based on STREAM”, Proc. of KSIS Fall Conference, pp. 131-136, 2006.

[3] B. Gedik, et al., “MobiEyes: Distributed processing of continuously moving queries on moving objects in a mobile system”, Proc. of the International Conference on Extending Database Technology, 2004.

[4] C. S. Jensen, et al., “Query and update efficient B+-tree based indexing of moving objects”, Proc. of the International Conference on Very Large Data Bases, pp. 768-779, 2004.

[5] H. Hu, et al., “A generic framework for monitoring continuous spatial queries over moving objects”, Proc. of the ACM International Conference on Management of Data, SIGMOD, 2005.

[6] G. S. Iwerks, et al., “Continuous K-nearest neighbor queries for continuously moving points with updates”, Proc. of the International Conference on Very Large Data Bases, 2003.

[7] C. S. Jensen, et al., “Query and update efficient B+-tree based indexing of moving objects”, Proc. of the ACM international Conference on Very Large Data Bases, 2004.

[8] M. A. Shah, et al., “Highly-available, fault-tolerant, parallel dataflows”, Proc. of the ACM SIGMOD, 2004.

[9] Y. Xing, et al., “Dynamic Load Distribution in the Borealis Stream Processor”, Proc. of IEEE ICDE Conference, 2005.

[10] N. Tatbul, et al., “Load management and high availability in the borealis dis-tributed stream processing

engine", Technical Report, ETHZurich, 2006.
[11] Min-ho Seo, et al., "Resource Sharing Method to Reduce Duplicate Operation Cost of Multiple Spatial Aggregates in u-GIS Environment", Proc. of the 31th KIPS Spring Conference, pp. 344-347, 2009.
[12] A. Guttman, "R-tree: A dynamic index structure for spatial searching", Proc. of International Conference on Management of Data, ACM SIGMOD, 1984.
[13] W. Chung, et al., "GeoSensor Data Stream Processing System for u-GIS Computing", The Journal of KSIS , Vol. 11, No. 1, pp. 9-16, 2009
[14] "Tiger/Line Shapefiles", www.census.gov/geo /www/tiger/tgrshp2007/tgrshp2007.html, 2007.

정 원 일(Weonil Chung) **[정회원]**

- 1998년 2월 : 인하대학교 전자계산공학과(공학사)
- 2004년 8월 : 인하대학교 컴퓨터정보공학과(공학박사)
- 2004년 7월 ~ 2006년 7월 : 한국전자통신연구원 선임연구원
- 2007년 3월 ~ 현재 : 호서대학교 정보보호학과 교수

<관심분야>
데이터스트림, 이동객체, 시스템보안