Journal of the Korea Academia-Industrial cooperation Society
Vol. 14, No. 10 pp. 5156-5161, 2013

http://dx.doi.org/10.5762/KAIS.2013.14.10.5156

분산 공간 데이터 스트림 시스템에서 연산 처리율 기반의 적응적 업스트림 백업 기법

정원일[1*]
[1]호서대학교 정보보호학과

Adaptive Upstream Backup Scheme based on Throughput Rate in Distributed Spatial Data Stream System

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

요 약 분산 공간 데이터 스트림 처리에서는 분산 노드의 활용도를 높이고 고장이 발생한 경우 신속하게 시스템을 복구하기 위해 하위 노드에서 처리된 튜플에 대해 상위 노드로 데이터를 백업한다. 그러나 데이터의 유입량이 증가하고 노드의 연산 결과를 다수의 하위 노드들과 공유할 때 튜플 처리가 지연되면 상위 노드의 삭제 지연으로 인해 백업 데이터의 손실을 야기할 수 있다. 본 논문에서는 노드들의 데이터 유입량과 하위 노드의 연산 처리율을 분석하고 적응적 업스트림 백업 방법을 적용하여 노드의 평균 부하율을 감소시키고, 노드 연산 결과의 공유에 따른 데이터 손실을 최소화하는 방법을 제안한다. 그리고 실험에서는 제안 기법을 통해 데이터 손실을 방지하고, 노드 모니터링에 소요되는 CPU 사용률을 평균 20% 감소시키는 결과를 나타낸다.

Abstract In distributed spatial data stream processing, processed tuples of downstream nodes are replicated to the upstream node in order to increase the utilization of distributed nodes and to recover the whole system for the case of system failure. However, while the data input rate increases and multiple downstream nodes share the operation result of the upstream node, the data which stores to output queues as a backup can be lost since the deletion operation delay may be occurred by the delay of the tuple processing of upstream node. In this paper, the adaptive upstream backup scheme based on operation throughput in distributed spatial data stream system is proposed. This method can cut down the average load rate of nodes by efficient spatial operation migration as it processes spatial temporal data stream, and it can minimize the data loss by fluid change of backup mode. The experiments show the proposed approach can prevent data loss and can decrease, on average, 20% of CPU utilization by node monitoring.

Key Words : Fault Tolerance, Load Distribution, Spatial Data Stream, Upstream Backup

1. 서론

이동 기기들의 보급 확대로 시공간 데이터 스트림을 생성하는 이동 객체들이 급증하고 있다[1-4]. 이러한 이동 객체들의 응용 서비스에서는 공간 연산을 포함하는 공간 질의에 대한 요청률이 급격하게 증가한다[5-7]. 이러한 대량의 실시간 데이터와 공간 질의 요청은 큰 질의 처리 비용이 요구되므로 부하를 분산하여 처리하기 위한 분산 데이터 스트림 처리에 관한 연구들이 수행되고 있다[8,9].

분산 데이터 스트림 처리 시스템에서의 한 노드의 고장은 전체 스트림 처리의 정지와 데이터의 손실을 초래하므로 노드의 실패에도 데이터의 손실 없이 빠른 복구를 제공할 수 있는 고가용성을 제공해야 한다[10-12].

이 논문은 2012년도 호서대학교의 재원으로 학술연구비 지원을 받아 수행된 연구임(2012-0240)
*Corresponding Author : Weonil Jeong(Hoseo Univ.)
Tel: +82-41-540-5984 email: wnchung@hoseo.edu
Received July 4, 2013 Revised (1st July 31, 2013, 2nd August 6, 2013) Accepted October 10, 2013

Process pair 모델은 연산을 처리하는 일차 노드들과 고장이 발생할 경우 복구를 위한 이차 노드들로 구성되며, 쌍이 되는 노드들 간에 주기적으로 상태 정보를 생성하고 전송하기 때문에 통신 오버헤드와 처리 오버헤드가 큰 단점이 있다. 업스트림 백업(Upstream Backup) 기법은 상위 노드가 자신의 하위 노드들의 실패 시 복구를 위한 백업 역할을 한다. 하위 노드에서 상위 노드로부터 입력된 튜플을 모두 처리할 때까지 관련된 튜플을 상위 노드의 출력 큐에 저장하여 노드 실패를 위한 별도의 예비 노드가 불필요하여 운용비용을 감소시킨다. 그러나 업스트림 백업 기법은 데이터 스트림의 유입량이 증가하고 노드의 연산 결과를 다수의 하위노드들이 공유하는 경우에 하위 노드의 연산처리 지연이 발생할 경우 상위노드의 출력 큐의 여유 공간 부족으로 인한 데이터의 손실 문제가 발생할 수 있다.

본 논문에서는 분산 데이터 스트림 처리 시스템에서 고가용성을 제공하기 위해 적응적 업스트림 백업기법을 제안한다. 제안 기법은 한 노드가 다수의 하위 노드를 갖는 부분에 대해 노드의 데이터 스트림의 유입량과 하위 노드들의 연산 처리율을 모니터링하면서 특정 임계값을 벗어나면 복구를 위한 백업을 특정 서버가 수행하도록 한다. 그렇지 않은 노드들은 실패 시 복구를 위한 백업 방법으로 기존의 업스트림 백업 방법을 적용한다. 또한, 제안 기법은 노드의 상태 모니터링을 이용하여 복구를 위해 백업 방법을 변경할 때 기존의 업스트림 백업에서 부하가 발생할 경우 출력 큐의 오버플로우로 인해 발생할 수 있는 데이터의 손실을 최소화할 수 있다.

2. 관련 연구

2.1 분산 데이터 스트림 처리

분산 데이터 스트림 처리 시스템(DSMS: Distributed Data Stream Processing System)은 다수의 연산을 포함하여 처리할 수 있는 노드들로 구성된다. DSMS에서의 연산은 map, filter, join, aggregate 등이 이용된다. 그리고 각 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[8,9].

2.2 고가용성 기법

분산 데이터 스트림 처리 시스템에서의 고가용성 알고리즘[10-12]으로, Passive Standby 기법은 대기 노드에 각 활성 노드를 연결시키고 실패 시 대기 노드가 연산의 처리를 넘겨받는다. 활성화 상태의 일차 노드가 실패하면 대기 노드인 이차 노드가 처리를 넘겨받고 이차 노드의 출력 큐로부터 모든 튜플들을 일차 노드의 다운스트림(downstream) 노드들로 보낸다. 또한, 실패한 일차 노드의 업스트림(upstream) 노드들도 튜플들의 처리를 넘겨받은 이차 노드로 보내기 시작한다. 그리고 실패했던 일차 노드는 복구가 완료되면 새로운 이차 노드가 된다. Active Standby 기법은 실패 시의 복구를 위한 전용 이차 노드를 갖는 것은 Passive Standby와 유사하지만 Active Standby기법에서 이차 노드들은 일차 노드의 업스트림 노드로부터 입력 튜플들을 받아 이를 병행으로 처리한다. 이차 노드는 그들이 생산한 튜플들을 출력 큐에 저장해 놓고 다운 스트림 노드로 보내지는 않는다. 그리고 일차 노드가 실패하면 이차 노드가 처리를 넘겨받고 저장했던 튜플들을 다운스트림 노드들로 전송한다. 이러한 튜플들은 일차 노드가 모든 상응하는 튜플들을 다운스트림 노드로 보내면 이차 노드에서 버려진다. 이 방식의 주요 장점은 대기 노드에서 일차 노드에서의 처리와 똑같은 처리를 하다가 실패가 발생하면 즉시 Standby 노드가 병행처리 했던 결과로 바로 연산 처리를 이어갈 수 있으므로 Passive Standby보다 빠른 복구가 가능하다. 업스트림 백업 기법에서는 작은 복구시간 비용으로 복구 실행 오버헤드를 감소시키며, 이 접근에서 업스트림 노드들은 자신의 다운스트림 노드들에게 백업의 역할을 한다. 백업 역할을 수행하기 위해 각 노드는 자신의 모든 다운스트림 이웃 노드들로 보낸 튜플들이 해당 노드들에서 완전히 처리될 때까지 그것들을 출력 큐에 저장한다. 따라서 다운 스트림 노드가 실패해도 다운스트림 노드에 저장되어 있는 튜플을 재처리하여 실패 발생 이전의 잃어버린 상태를 복구할 수 있다. 노드는 튜플을 생산하고 그것을 이웃한 다운스트림 노드로 전송한다. 노드들은 받은 튜플들에 대한 응답으로 업스트림 이웃 노드들에게 주기적으로 레벨 0의 ack 메시지를 보낸다. 레벨 0에 해당하는 ack 메시지를 받은 노드는 해당 튜플들을 출력 큐에 저장하고 입력 튜플과 출력 튜플간의 관계를 매핑시킨다. 이 튜플간의 매핑 정보를 통해 자신의 다운스트림 노드의 출력 큐에 저장되어 있는 해당 튜플들을 삭제해도 됨을 알려주는 레벨 1의 ack 메시지를 작성해 전송하며 해당 메시지를 받은 노드는 레벨 1의 ack 메시지 중 제일 작은 튜플 번호에 해당하는 튜플까지 출력 큐에서 삭제한다. Passive Standby 기법과 Active Standby 기법은 일차 및 이차 노드 간의 빈번한 체크포인트 메시지 전송으로 인해 높은 통신 오버헤드와 처리 오버헤드가 발생한다.

Upstream Backup 기법은 통신 오버헤드와 처리 오버헤드는 감소시키지만 노드 연산의 결과를 다수의 다운스트림 노드들이 공유하므로 데이터 스트림의 유입량이 폭발적으로 증가할 경우 출력 큐의 오버플로우로 인해 데이터의 손실이 발생할 수 있다.

3. 적응적 노드 복제 기법

3.1 적응적 업스트림 백업

분산 스트림 처리 환경에서는 노드의 연산 결과를 다른 연산자들로 이루어진 다수의 노드들이 공유한다. 예를 들면, 노드 A에서 주식시장에서 금일 주가가 3%이상 오른 종목들을 걸러내는 연산을 수행했다면 이 결과를 다양한 목적으로 사용하기 위하여 다수의 노드들이 A 노드의 결과를 자신들의 입력으로 받아들인다. 이런 경우에 기존의 Upstream Backup에서는 노드의 실패 발생 시 복원을 위해 필요한 튜플들을 자신의 출력 큐에 저장한다. 그리고 다운스트림 노드들은 자신들이 처리를 완료한 튜플들에 대하여 A의 출력 큐에서 백업을 위해서 저장하고 있는 해당 튜플들을 삭제해도 좋다는 메시지들을 보낸다. 그러면 노드 A는 해당 삭제 메시지들 중 제일 낮은 번호의 튜플까지 자신의 출력 큐에서 삭제한다. 다운스트림 노드들 중 한 노드의 연산처리가 지연되고 데이터의 유입량이 빨라지면 노드 A의 출력 큐에 저장 공간이 없어서 데이터의 손실이 발생할 수 있다.

이런 문제를 해결하기 위하여 제안하는 적응적 Upstream Backup은 노드의 데이터 스트림의 유입률과 다운스트림 노드들의 연산 처리율을 모니터링 하여 노드의 실패 시 복구를 위한 백업의 방법을 유동적으로 변경할 수 있게 한다. 이를 위하여 분산 데이터 스트림 처리 시스템의 질의 네트워크를 이루고 있는 노드 중에서 다수의 다운스트림 노드를 가지고 있는 노드들은 자신의 입력 스트림으로 입력되는 데이터 스트림의 유입량을 지속적으로 계산한다. 또한, 다운스트림 노드들은 연산 처리율을 계산하여 자신의 업스트림 노드에게 주기적으로 전송한다. 이러한 정보들은 특정 테이블에 저장되어 관리된다.

$$I = \lceil N_{input} S_{input} / F_r \rceil \qquad \text{(Expr. 1)}$$

(수식 1)은 유입률을 구하는 식이다. 데이터 유입률 I는 등록된 갱신주기 F_r동안 입력 큐로 유입되는 데이터의 개수 N_{input}과 데이터의 크기 S_{input}에 의해 결정된다.

$$P^i = \left\lceil N^i_{output} \left(\sum_{j=1}^{n} O^i_j \right) / N^i_{input} \right\rceil \qquad \text{(Expr. 2)}$$

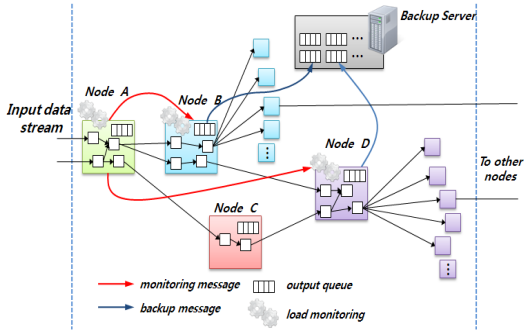
(수식 2)는 한 노드의 i번째 다운스트림 노드에서 계산되는 연산처리율 P^i를 구하는 식으로 갱신 주기마다 수행된다. 각각 N^i_{output}는 노드의 연산 처리한 튜플의 개수, N^i_{input}은 입력된 튜플의 개수, O^i_j는 연산자의 가중치를 나타낸다.

노드는 자신의 데이터 유입률 I를 계산하며 다운스트림 노드들은 각자의 연산처리율 P^i를 계산하여 자신의 업스트림 노드에게 보낸다. 노드는 데이터의 유입률과 다운스트림 노드들의 연산처리율을 바탕으로 정해진 부하 발생 기준치를 넘으면 출력 큐의 내용을 정해진 백업서버로 전송하게 된다. 백업을 위하여 다운스트림 노드로부터 오는 레벨 0의 ack 메시지 중 제일 큰 튜플의 번호를 확인하여 백업 메시지를 구성하고 백업을 위한 서버로 이를 전송한다. 백업 서버로부터 전송이 잘 되었다는 확인 메시지를 받으면 노드는 메시지를 보낸 튜플의 번호까지 큐를 비운다. 노드 부하 발생 시 이 같은 백업방법 변경으로 출력 큐의 빈 공간을 신속하게 만들 수 있고 백업의 내용은 백업서버에 저장함으로써 다운스트림 노드의 실패에도 견딜 수 있다.

[Algorithm 1] ChangeBackup

```
Input:  I  (date rate),
        P^i (downstream node's processing rate)
Output: SUCCESS or FAILURE
01 :  while receive exit message
02 :      dataInputRate = calculateIR();
03 :      processingRate = calculatePR();
04 :      if( dataInputRate * processingRate
              > StandardLoadofNode)
05 :          activateBSConnection();
06 :          makeACK0Msg();
07 :          sendMsg();
08 :          responseACK();
09 :          deleteTpFromQueue();
10 :      end if
11 :      else
12 :          deleteBackupTp();
13 :          sendResultTupleToDS();
14 :          receiveACK0Msg();
15 :          saveTupleToOutputQ();
16 :          receiveACK1Msg();
17 :          deleteTpFromQ();
18 :       end if
19 :  end while
```

[Algorithm 1]은 노드의 부하를 모니터링 하여 부하의 발생 여부에 따라 백업 방법을 변경하는 알고리즘이다.



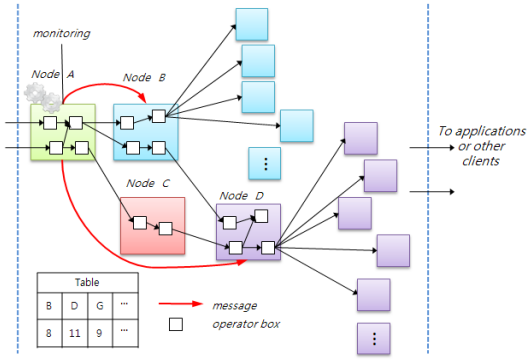
[Fig. 1] Change the backup path of output queues caused by the overloaded node

Fig. 1은 부하 발생 노드에서의 출력 큐의 백업 방식 변경에 대한 흐름을 나타낸다. 자신의 다운스트림 노드들이 다수인 노드 B와 D는 데이터의 유입량과 다운스트림 노드들의 연산 처리량을 계산하여 부하가 발생했음을 알고 출력 큐의 내용을 백업 서버에 저장한다.

3.2 모니터링 부하 제어

노드의 부하발생 여부를 모니터링하기 위해서는 다운스트림 노드들이 다수인 노드들에서는 지속적으로 데이터의 유입량과 다운스트림 노드들의 연산 처리율을 산출해야 한다. 이는 또 다른 연산 비용을 야기하므로 이를 줄이기 위하여 먼저 질의 네트워크를 이루는 노드들 중에서 제일 처음으로 데이터 스트림을 입력받는 노드에서 데이터의 유입률을 지속적으로 모니터링을 수행한다. 모니터링 중 계산 값이 특정한 임계값을 넘으면 질의 네트워크의 시작노드에서 테이블로 관리하는 다수의 노드들을 다운스트림으로 갖는 노드들에게 부하 모니터링을 시작하라는 메시지를 전송한다. 해당 메시지를 받은 노드들은 부하 모니터링을 시작하게 된다.

Fig. 2는 질의 네트워크의 첫 번째 노드에서 이루어지는 데이터 유입량 모니터링과 부하의 발생 시 해당 노드들에게 부하 모니터링 시작 메시지 전송과정을 보여준다. 노드 A는 질의 네트워크의 처음 노드로서 입력 데이터 스트림을 받아들이고 유입량을 계산하여 부하율을 계산한다. 계산 결과가 특정 임계값을 넘으면 노드 A는 다수의 다운스트림 노드들이 연산 결과를 공유하는 노드들의 정보를 관리하는 테이블에서 해당 노드들에게 부하 모니터링을 시작하라는 메시지를 보낸다. Fig. 2에서는 노드 A가 테이블을 보고 노드 B와 D, G에게 메시지를 보내게 된다. 메시지를 받은 노드들은 각각 부하 모니터링을 시작하고 부하 모니터링 알고리즘에 따라 작업을 수행한다.



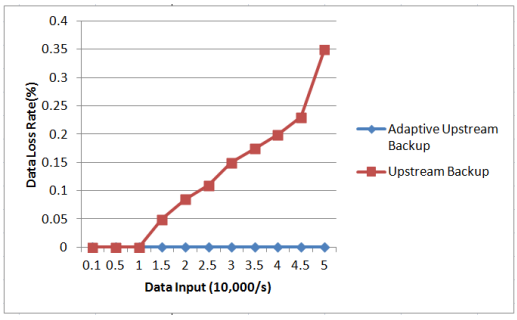
[Fig. 2] Monitor load status of nodes

4. 성능 평가

본 실험은 Fedora Core 8.0을 기반으로 수행했으며, 시스템 환경은 Intel(R) Pentium(R) 4 CPU 3.00GHz이고, 메인 메모리는 4GB이다. 실험에 사용된 데이터는 TIGER/Line 2007 데이터로, Oracle에 구축하여 본 실험에 사용한다[13]. 스트림 큐의 크기는 5MB, 스트림 큐의 개수는 15개, 등록된 공간 질의와 비공간 질의는 10개와 5개로 설정하고, 200초 동안 스트림 큐의 부하 임계값을 3.5MB로 지정하며 각 실험은 5회씩 수행하여 산출한 결과이다. 본 실험에서 사용된 부하 모니터링 알고리즘은 분산 노드의 부하 모니터링 방법[9]을 적용하였다.

4.1 부하로 인한 데이터 손실량

이 실험에서는 기존의 Upstream Backup 기법과 제안 기법의 데이터 손실률을 비교하기 위해 데이터의 발생 수를 증가시키면서 데이터의 손실률을 측정하였다.

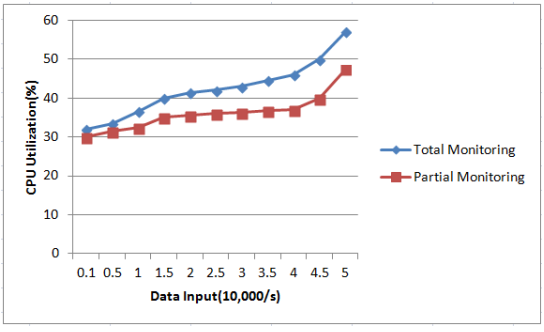


[Fig. 3] Evaluation of the data loss rate

Fig. 3은 두 기법의 데이터 손실률을 나타낸다. 기존의 Upstream Backup 기법은 초당 데이터가 1만 건씩 발생할 때부터 데이터의 손실이 발생한다. 또한, 데이터가 2만 건 씩 발생할 때 데이터의 손실률이 급격히 증가하였다. 제안 기법은 기존 Upstream Backup 기법의 단점이었던 출력 큐의 삭제로 인해 발생할 수 있는 데이터 손실 문제에 대해 모니터링을 통해 부하 발생이 감지될 경우 고가용성 제공을 위한 데이터의 백업 방법을 특정 백업서버로 변경하게 함으로써 출력 큐에서 발생했던 데이터의 손실 문제가 해결된 것을 확인할 수 있었다.

4.2 시스템 자원 소모량

이번 실험은 노드마다 모니터링을 실시했을 경우와 제안한 기법에서 사용한 부분적 노드 모니터링을 사용하였을 때의 성능을 측정한다. 제안하는 기법에서는 질의 네트워크를 이루는 모든 노드에서 모니터링을 하지 않고 한 노드의 연산 결과를 다수의 노드들이 공유하는 노드에 대해서만 데이터 유입율과 연산처리율을 모니터링 한다. 두 방법의 성능을 측정하기 위해 데이터 스트림의 유입량을 매초 마다증가 시키면서 두 방법에 따른 CPU의 사용률을 측정하였다.



[Fig. 4] CPU Utilization by the monitoring approaches

Fig. 4는 초당 1천 건의 데이터 생성에서부터 5만 건의 데이터 생성에 따른 각 기법의 CPU 사용률을 나타낸다. 이 결과에서 전체 노드 모니터링 방법은 부분 노드 모니터링 방법보다 실험 구간에서 평균 20% 정도의 높은 CPU 사용률을 보이며, 특히 데이터 발생 빈도가 2만 건을 넘으면서 그 격차는 더욱 커짐을 알 수 있다.

5. 결론

본 논문에서는 분산 데이터 스트림 처리 시스템에서 시스템의 노드가 고장을 일으키더라도 노드의 활용도와 백업 데이터의 손실 문제를 해결하기 위한 고가용성 기법을 제안하였다. 제안하는 고가용성 기법은 노드의 처리율과 부하율 모니터링을 통해 백업 방식을 변경함으로써 백업 데이터의 손실을 최소화하였다. 모니터링 비용의 감소를 위해서 한 노드가 여러 하위 노드와 연결된 경우 해당하는 노드에서 모니터링을 실시하였다. 이에 제안 기법은 부하가 발생할 경우 시스템 고장에 대비한 백업 데이터의 손실을 최소화할 수 있도록 하였다. 향후 연구로는 다양한 분산 환경에서의 모니터링에 적용할 수 있도록 노드 부하에 따른 임계치 산출 및 백업 등에 대한 다양한 실험과 평가 방안이 필요하다.

References

[1] S. R. Ahuja, K. D. Hong, K. S. Hong, "The Rapport Multimedia Conferencing System", *Proc. of 2nd IEEE Conference on Computer Workstations*, pp. 52-58, March, 1988.
DOI: http://dx.doi.org/10.1109/COMWOR.1988.4800

[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. 10.
DOI: http://dx.doi.org/10.1109/TC.2002.1039840

[2] C. Park, D. Hong, C. Park and K. Han, "Design and Implementation of a Spatial DSMS based on STREAM", Proc. of KSIS, pp. 131-136, 2006. 11.

[3] B. Gedik, L. Liu, "MobiEyes: Distributed processing of continuou sly moving queries on moving objects in a mobile system", Proc. of the International Conference on Extending Database Technology, 2004.
DOI: http://dx.doi.org/10.1007/978-3-540-24741-8_6

[4] C.S Jensen, D. Lin, B.C. Ooi, "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] Hu H, Xu J, Lee D.L, "A generic framework for monitoring continuous spatial queries over moving objects", Proc. of the ACM International Conference on Management of Data, SIGMOD, Baltimore, 2005.
DOI: http://dx.doi.org/10.1145/1066157.1066212

[6] Iwerks G.S, Samet H, Smith K, "Continuous K-nearest neighbor queries for continuously moving points with updates", Proc. of the International Conference on Very Large Data Bases, VLDB, 2003.

[7] Jensen C.S, Lin D, Ooi B.C, "Query and update efficient B+-tree based indexing of moving objects", Proc. of the ACM international Conference on Very Large Data Bases, VLDB, 2004.
[8] M. A. Shah, J. M. Hellerstein, and E. Brewer. Highly-available, fault-tolerant, parallel dataflows. In Proc. of the 2004 ACM SIGMOD, June 2004.
DOI: http://dx.doi.org/10.1145/1007568.1007662
[9] Weonil Chung, "Spatial Operation Allocation Scheme over Common Query Regions for Distributed Spatial Data Stream Processing," KAIS Journal, Vol. 13, No, 6, pp. 2713-2719, 2012.
DOI: http://dx.doi.org/10.5762/KAIS.2012.13.6.2713
[10] J.H. Hwang, M. Balazinsk, A. Rasin, U. Cetintem-el, M. Stonebraker, and S. Zdonik. High-availability algorithms for distributed stream processing. In Proc. of the 21st ICDE Conf., Apr. 2005.
DOI: http://dx.doi.org/10.1109/ICDE.2005.72
[11] J. Barlett, J. Gray, and B. Horst. Fault tolerance in Tandem computer systems. Technical Report 86.2, Tandem Computers, Mar. 1986.
[12] J. Gray. Why do computers stop and what can be done about it? Technical Report 85.7, Tandem Computers, 1985.
[13] "Tiger/Line Shapefiles.", www.census.gov/geo/www/tiger/tgrshp2007/tgrshp2007.html, 2007.

정 원 일(Weonil Jeong) **[정회원]**

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

<관심분야>
데이터스트림처리, 이동객체, 시스템보안