Journal of the Korea Academia-Industrial cooperation Society
Vol. 13, No. 1 pp. 370-381, 2012

http://dx.doi.org/10.5762/KAIS.2012.13.1.370

유효시간 데이터 스트림에서의 스카이라인 질의 알고리즘

박남훈[1], 장중혁[2*]
[1]안양대학교 컴퓨터학과, [2]대구대학교 컴퓨터IT공학부

Efficient Skyline Computation on Time-Interval Data Streams

Nam Hun Park[1] and Joong-Hyuk Chang[2*]
[1]**Dept. of Computer Science, Anyang University**
[2]**Dept. of Computer & Information Technology, Daegu University**

요 약 다기준 의사결정 연구는 평가기준이 상이한 다수의 선호도로부터 최선의 대안을 찾는 방법으로 실시간 재난 탐지, 센서를 이용한 서식 모니터링 등의 응용환경에서 활용되어 왔다. 최근 유효시간 데이터 스트림 응용환경에서 각 객체들이 개개의 유효시간을 가지므로, 기존의 슬라이딩 윈도우보다 다기준 의사결정 방법, 즉 스카이라인 질의 수행에 더 많은 연산이 필요한다. 본 연구에서는 유효시간 데이터 스트림에서 스카이라인 질의를 수행하는 TI-Sky 알고리즘을 제시한다. 실시간 환경에서 새로운 객체가 생성되고 소멸되기까지 유효한 객체들을 관리하고 스카이라인 질의를 수행하기 위해 파티션단위의 시간 지배관계를 제시한다. 객체의 생성과 유효시간, 지배관계에 따라 시간지배관계를 갱신하며 다양한 방법으로 사멸객체를 제거하여 수행성능을 향상 시켰다. 실험을 통해 TI-Sky가 다양한 데이터 상에서 기존 연구보다 뛰어난 성능으로 스카이라인 질의를 수행하는 것을 증명하였다.

Abstract Multi-criteria result extraction is crucial in many scientific applications that support real-time stream processing, such as habitat research and disaster monitoring. Skyline evaluation is computational intensive especially over continuous time-interval data streams where each object has its own customized expiration time. In this work, we propose TI-Sky – a continuous skyline evaluation framework. To ensure correctness, the result space needs to be continuously maintained as new objects arrive and older objects expire. TI-Sky strikes a perfect balance between the costs of continuously maintaining the result space and the costs of computing the final skyline result from this space whenever a pull-based user query is received. Our key principle is to incrementally maintain a partially precomputed skyline result space – however doing so efficiently by working at a higher level of abstraction. TI-Sky's algorithms for insertion, deletion, purging and result retrieval exploit both layers of granularity. Our experimental study demonstrates the superiority of TI-Sky over existing techniques to handle a wide variety of data sets.

Key Words : Skyline, Time-interval Data Stream, Multi-Criteria Decision Support

1. 서론

최근 야생동물 서식지 연구, 외래환자 건강관리 연구, 그리고 환경재해 감지 등의 많은 과학응용분야에서 실시간 모니터링, 이상탐지, 변화분석 등을 위해 다기준 의사결정(MCDS: Multi- Criteria Decision Support) 질의를 활용하고 있다. 스카이라인(Skyline) 질의는 다기준 의사결정 연구들 중 유용한 기법으로 특히, 사용자 선호기반 기준으로 의사결정을 지원할 수 있어 가장 많이 활용되고 있다[2,6]. 스카이라인 질의는 다수의 기준으로 분석이 필요하거나, 또는 선호 기준간의 상호관계를 결정하는데 유용한 역할을 한다. 하지만, 이러한 다기준으로 분석한 해

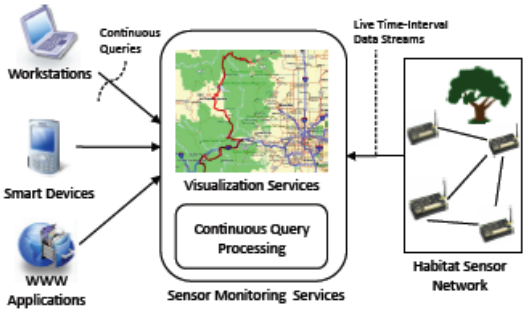
본 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임 (2011-0025300).
*교신저자 : 장중혁(jhchang@daegu.ac.kr)
접수일 11년 11월 08일 수정일 11년 12월 27일 게재확정일 12년 01월 05일

당 데이터 간의 우선순위는 다기준간 상호관계를 분석하는데 유용하지 않다. 오히려 스카이라인 질의 결과로부터 상호관계를 명확히 분석할 수 있다.

최근 유비쿼터스 스마트 센서, RFID 네트워크 등이 다양한 실세계 응용환경에서 활용됨에 따라 이러한 기기들에서 생성되는 데이터스트림이 크게 증가하였다[1,8]. 데이터 스트림 질의 처리는 기존 데이터베이스 질의처리와 여러가지 면에서 차이가 많다. 첫째, 응용환경에서 생성되는 데이터 스트림이 지속적으로 무한하다. 둘째, 데이터는 시간에 따라 변화하며 각 데이터 객체는 생성시간과 소멸시간으로 구성된 유효주기를 가질 수 있다. 셋째, 데이터 스트림 질의의 수행 범위는 무한할 수 있다. 넷째, 스마트 센서 모니터링과 같은 실시간성이 부각되는 응용환경에서는 질의결과를 제공하기 위해서 질의수행이 실시간으로 이루어져야 한다.

기존에 발표되어온 정적 데이터베이스에서의 스카이라인 연구[2,6]들은 한정된 데이터 집합에 대해 인덱스를 미리 구축하고 이를 반복 탐색하여 스카이라인 질의를 수행하였다. 또한, 유효시간을 가진 데이터 스트림과 같은 실시간 응용환경을 고려하지 않아 최근 응용환경에 있어서 유용한 다기준 의사결정을 지원하지 못한다. 기존의 데이터베이스 시스템에서는 한번의 수행결과를 요청하는데 비해 데이터 스트림 응용환경에서 질의는 지속적으로 결과를 생성하는 연속질의의 형태로 수행된다. 그림 1과 같은 야생동물 서식지 연구를 위한 센서 모니터링 환경을 예로 들 수 있겠다.

최근 고성능의 휴대용 컴퓨팅 기기들이 널리 사용되고 중저가의 규격센서, 즉 GPS센서와 수치측정센서 등이 널리 보급됨에 따라 기존의 초고속 유/무선 네트워크를 활용한 센서 네트워크가 관련분야에서 활용되고 있다. 그림 1은 야생 서식지 모니터링에 응용되는 센서 네트워크의 예이다. 야생 서식지의 무선 센서로부터 수집된 센서 데이터는 실시간 감시와 분석을 통해 기후변화에 따른 패턴 변화나 이주 분석 연구에 사용된다. 하지만, 휴대용 센서 기기에서의 데이터 전송은 에너지 효율면에서 논의가 필요하다. 기기에서는 가용자원용량과 측정빈도에 연관하여 각각 다른 주기로 측정치를 전송한다. 따라서, 각 센서 데이터는 해당 전송 시점의 타임스탬프를 동반하며 또한, 다음 센서 데이터가 어느 시간간격으로 전송될 것인지, 즉 유효주기 데이터 스트림(time-interval data stream)의 형태로 전송된다. 연구자들은 다기준 기반의 비정상 측정치를 보이는 센서 측정값을 실시간 감시한다.



[그림 1] 야생 서식지 및 환경 모니터 응용환경
[Fig. 1] Habitat and Environmental Monitoring System

최근 데이터 스트림에서의 스카이라인 연산은 크게 슬라이딩 윈도우와 유효시간 데이터 스트림 모델로 분류할 수 있다. Stabbing-the -Sky[3]에서는 슬라이딩 윈도우 크기 내의 최근 n개의 객체만을 저장하며, 이들 객체간의 지배관계를 갱신한다. 본 논문의 비교실험에서는 동등한 조건의 비교를 위해 기존 Stabbing-the-Sky을 유효시간 개념을 적용하여 Stabbing-the- Sky+를 별도로 구현하여 비교하였다. 객체 단위로 모든 지배관계가 관리되므로, 남아있는 객체의 수에 따라 매 객체가 갱신될 때의 지배관계 연산 비용이 증가한다.

유효시간 데이터 스트림 모델에서의 스카이라인 연산은 슬라이딩 윈도우를 포함하는 보다 넓은 범위의 데이터 스트림을 다룬다. LookOut[4] 연구에서 유효시간 스트림 스카이라인 탐사방법을 제시하였다. 유효시간 내의 객체에 대한 다차원 인덱스를 생성하여, 객체의 추가/소멸 연산마다 인덱스를 반복 검색하여 스카이라인을 탐색한다. 반면에 스카이라인과 유사한 다기준 의사결정 지원 연구로 순위기반 질의를 꼽을 수 있다. Top-K 질의와 같이 순위기반 질의에서는 사용자가 정의한 평가함수(scoring function)[5]의 비용을 최소로 하는 객체들을 우선순위를 두어 탐사한다.

본 논문에서는 유효시간 데이터 스트림에서의 스카이라인 연속질의 수행 방법을 제시한다. 각 데이터 객체는 객체의 유효시간을 표시하기위한 객체 생성시간과 소멸시간을 동반한다. 이러한 유효시간 데이터 스트림 상에서 주어진 시간에서 유효한 객체들로 구성된 스카이라인을 구하고 관리하는 것이 본 논문의 주제이다.

기존의 스카이라인 연구들은 유효시간 데이터 스트림 환경에서 다음과 같은 문제점을 가진다.

A. 유효시간 처리의 어려움 : 기존의 데이터 스트림 스카이라인 알고리즘[3,8,10]에서는 각 객체들의 유효시간을 정의하지 않는다. 대신, 슬라이딩 윈도우 모델

(sliding window model)을 사용하여 모든 데이터 객체들은 정해진 윈도우의 주기에 따라 동일한 시간뒤에 소멸한다. 따라서, 새롭게 생성된 객체 o_{new}는 항상 윈도우에 이미 존재하고 있는 모든 객체가 소멸된 후에 소멸된다. 이러한 이유로, [3]에서 o_{new}는 지배관계(domination relationship)에 있는 모든 데이터 객체를 삭제시킬 수 있어 슬라이딩 윈도우에서 효율적으로 스카이라인을 수행할 수 있다. 반면에, 시간간격 데이터 스트림에서는 먼저 생성된 객체가 o_{new}보다 긴 유효시간을 가질 수 있다. o_{new}가 이전 생성된 객체 o_{old}를 지배하더라도 o_{new}가 소멸된 후에, o_{new}가 지배된 모든 객체를 제거하는 슬라이딩 윈도우와 달리 유효시간 스트림에서는 o_{old}가 다시 스카이라인이 될 수 있다. 이러한 경우를 고려해서 유효시간 데이터 스트림에서는 기존 연구보다 정확하게 객체를 유지하고 정확하게 소멸해야 한다. 최악의 경우 어떤 객체도 삭제되지 않고 존재할 수도 있다. 따라서, 유효시간 스트림 모델은 슬라이딩 윈도우 보다 포괄적인 데이터 스트림 응용환경이라 하겠다. 본 논문에서 이러한 특성을 반영하여 슬라이딩 윈도우에서의 스카이라인 수행이 가능하면서 동시에 유효시간 스트림에서도 수행이 가능한 방법을 제시한다.

B. 객체 단위의 연산수행 최소화 : [4]에서는 유효시간 데이터 스트림에서의 스카이라인 질의를 다루었다. 하지만, 모든 생성된 객체에 대한 인덱스를 구축하여 객체 단위의 비교연산을 반복수행하므로 효율성을 갖지 못한다. 기존 대부분의 스카이라인 알고리즘[3,4,8]들은 객체단위의 일대일 비교를 수행하는데, 이는 무한히 생성되는 데이터 스트림의 규모가 커지고 많은 객체가 유효할수록 수행시간이 길어진다. 보다 효율적인 스카이라인 질의처리를 위해서는 데이터 객체에서의 비교연산 이전에, 보다 상위레벨에서 데이터 공간정보를 활용하면 객체단위의 연산을 최소화할 수 있다. 본 연구에서는 다레벨의 함축정보, 즉 파티션지배관계를 활용하여 스카이라인 질의를 수행하고자 한다.

이러한 문제들을 극복하고자 본 논문에서는 유효시간 스트림에서 연속 스카이라인 질의를 수행하는 TI-Sky방법을 제시한다. 실시간 스카이라인 결과유지와 임의 질의 수행 비용간의 균형을 맞추기 위해 파티션단위의 연산을 수행한다. 기존 연구들의 객체간 지배관계 개념을 확장하여 유효시간을 반영한 객체시간지배(Micro-TDominance)와 파티션시간지배(Macro- TDomination)관계를 제시한다. 본 논문은 객체레벨에서 그리고 파티션레벨에서 추가, 삭제, 갱신 및 결과탐색 방법을 기술하며, 다음과 같은 장점을 가진다.

A. 유효시간 데이터 스트림을 대상으로 다기준 의사결정 질의 수행을 위한 TI-Sky 알고리즘을 제시한다.
B. 기존 스카이라인 알고리즘에서의 객체의 값에 의한 지배관계에 비해, 본 논문에서는 각 객체 소멸시간을 고려한 시간지배 관계를 제시한다.
C. 데이터 객체의 실시간 추가, 삭제, 탐색을 수행하여 객체시간지배, 파티션시간지배 관계를 실시간 갱신한다.
D. 실험을 통해 다양한 데이터를 대상으로 기존 연구보다 TI-Sky가 효율적인 것을 증명한다.

2. 유효시간 지배관계

기존 데이터 스트림 모델에서는 각 객체는 생성시간이나 질의 수행기에 입력된 시간을 표시하는 타임스탬프를 동반한다[7]. 또한, 많은 스트림 응용환경에서는 객체가 생성된 후 유효한 시간만큼을 소멸시간으로 객체와 함께 정의하기도 한다[1]. 유효시간 데이터 스트림을 정의하면 아래와 같다.

정의 1. d차원 데이터 객체로 구성된 유효시간.
데이터 스트림 S는 객체의 집합 $\{o_1,\cdots,o_n\}$으로 나타낸다. 객체 o_i와 동반된 객체의 생성시간과소멸시간은 각각 $(o_i.t_{arr},\ o_i.t_{exp})$로 표시한다. 생성시간과 소멸시간 사이의 데이터 객체가 유효한 간격으로 유효시간으로 정의한다.

주어진 다차원 속성 E⊆A에 대해서 객체의 집합 R상의 선호도 P는 P:=(E,$>_p$) 로 나타내며, $>_p$는 속성 E의 도메인에서의 값의 순서를 의미한다.

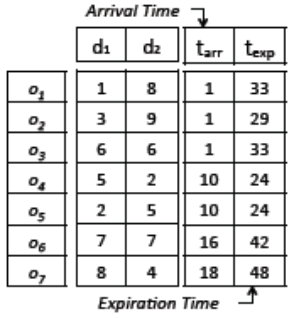
정의 2. 데이터 객체의 집합 R과 도메인 선호도.
P:=(E,$>_p$)로 구성된 정적 데이터베이스에서 선호도 P를 기반으로 $\forall(a_k\in E)\ (o_i[a_k]\geq_p o_j[a_k])\wedge\exists(a_l\in E)\ (o_i[a_l]>_p o_i[a_l])$가 성립하는 경우, 투플 $o_i\in R$ 은 $o_j\in R$를 지배한다.($o_i>_p o_j$)

예제 1. 그림 2와 같이 유효시간 스트림 S가 존재한다. 데이터 객체의 생성시간과 소멸시간을 무시하고 정의 2의 지배관계를 적용할 경우 $\{o_1,o_4,o_5\}$의 스카이라인을 구할 수 있다.

유효시간 스트림에서는 시간에 따라 기존의 객체가 소멸되어 스카이라인이 변화한다. [3]에서 제시된 기존의 객체간 지배관계를 확장하여 객체의 유효주기를 고려한 지배관계를 정의한다.

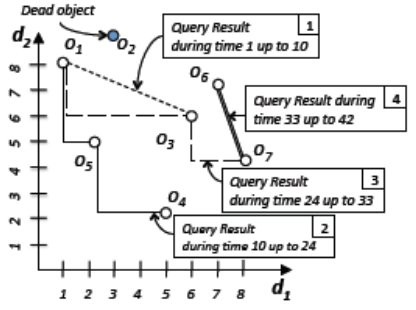
정의 3. 유효시간기반 지배관계.

주어진 선호도 P와 시간간격 데이터 스트림 S상에서, 객체 $o_i,o_j \in S$ 가 $o_i >_p o_j$ 이며, $o_j.t_{exp} > o_i.t_{exp}$인 경우 o_i는 o_j를 시간간격기반 지배(time-based domination)라하고, $o_i \rightarrow_T o_j$로 표시한다. 이 때, o_j를 시간지배하는 다음과 같은 o_k는 존재하지 않아야 한다. $\exists o_k \in S$, $o_k >_p o_j$ and $o_i.t_{exp} < o_k.t_{exp} < o_j.t_{exp}$.



	d_1	d_2	t_{arr}	t_{exp}
o_1	1	8	1	33
o_2	3	9	1	29
o_3	6	6	1	33
o_4	5	2	10	24
o_5	2	5	10	24
o_6	7	7	16	42
o_7	8	4	18	48

(a) 입력 유효시간 스트림 예



(b) 유효시간 스트림의 스카이라인

[그림 2] 유효시간 스카이라인
[Fig. 2] Motivating Example

그림 2의 시간간격 $1 \le t < 10$에서 o_1과 o_3는 스카이라인 객체가 된다. t=10의 시점에서 새로 생성된 투플 $o_5(2,5)$는 기존의 $o_3(6,6)$을 지배한다. 하지만, o_5가 단기간 유효한데 비해($o_5.t_{exp}=24$) o_3는 o_5가 소멸된 이후에도 유효하다($o_3.t_{exp}=33$). 그래서, 그림 2.b의 경우와 같이 o_5가 소멸된 $24 \le t < 33$의 시점에서는 o_3가 다시 스카이라인 객체에 해당된다.

연속 스카이라인 연산에 있어, t_i 시점에서 존재하는 객체들은 다음과 같이 분류할 수 있다. (1) 사멸객체 – t_i 이후의 시간에 결코 스카이라인 객체가 되지 않을 객체들의 집합. (2) 스카이라인 객체 – t_i 시점에 스카이라인에 존재하는 객체들의 집합. (3) 잠재적 스카이라인 – t_i 시점에서는 스카이라인에 속해있지 않지만, t_i 이후의 시점에 스카이라인이 될 수도 있는 객체들의 집합. 사멸객체로 판정되는 객체들은 미리 제거할 수 있다. 유효시간 데이터 스트림에서 스카이라인 탐사가 수행되는 동안 객체들은 스카이라인 객체에 속하거나 잠재적 스카이라인에 속하면서 객체의 유효시간동안 이를 반복한다.

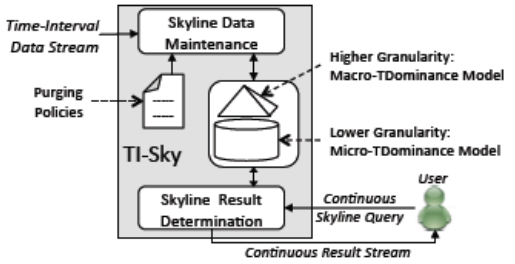
유효시간 데이터 스트림에서 스카이라인과 슬라이딩 윈도우에서 스카이라인의 차이점은 다음과 같다. 유효시간 데이터 스트림에서는 현재 지배되고 있는 객체도 이후에 스카이라인 결과가 될 수 있다. 하지만, 슬라이딩 윈도우 모델에서는 기존의 객체가 새로 생성된 객체에 의해 지배된다면 기존의 객체는 이후 스카이라인 결과가 될 수 없어 모두 일찍 제거할 수 있다. 따라서, 유효시간 스트림에서는 객체를 제거하기에 앞서 객체의 소멸시간을 비교, 연산해야한다. 즉, 슬라이딩 윈도우는 유효시간 스트림에서 모든 객체의 소멸시간이 생성된 시간으로부터 주어진 윈도우 크기만큼 고정된 보다 한정된 스트림이라 하겠다. 본 논문에서 제안하는 방법은 보다 포괄적인 유효시간 기반 스트림 환경 뿐 아니라, 기존의 슬라이딩 윈도우 스트림에서도 활용할 수 있다.

3. TI-Sky 스카이라인 탐사

본 논문에서는 그림 3과 같은 유효시간 스트림에서의 스카이라인 탐사를 위한 TI-Sky 방법을 제시한다. 기존 정적 데이터베이스에서의 객체단위 연산을 통한 스카이라인 탐사는 객체 수에 따라 과도한 비교연산을 반복하며, 각 객체가 추가되고 삭제될 때마다 지배관계를 갱신해야한다. TI-Sky에서는 객체수준의 비교연산을 지연시킨다. TI-Sky에서는 상위수준의 파티션단위의 데이터 공간을 구성하여 스카이라인을 상위레벨에서 우선적으로 수행할 수 있다.

스카이라인 관리 단계(Skyline Data Maintenance)에서는 스카이라인을 연산을 두 레벨로 관리한다. 상위레벨에서는 파티션 단위의 요약정보를 다루며, 하위레벨에서는 각 파티션 내의 객체들에서 지역 스카이라인을 다룬다. 생성된 스트림은 다차원 격자구조의 공간인 파티션으로 나누어 진다. 각 객체는 해당 범위의 파티션으로 분류되어 필요시 각 파티션의 상위레벨 요약정보를 갱신한다. 파티션의 요약정보로부터 객체 탐색없이 해당 파티션이

스카이라인 결과에 포함이 되는지 결정할 수 있다. 이러한 요약정보를 이용하여 각 파티션에서는 새로 생성된 객체에 대해 다음과 같이 수행한다. 1. 스카이라인 결과에 포함되지 않는 파티션에 생성된 모든 객체는 연산없이 제거할 수 있다. 2. 스카이라인 결과에 해당되는 파티션에 생성된 객체는 우선적으로 연산되어야 한다. 3. 객체 단위 연산은 현재 스카이라인 결과에 반영되지 않는 범위에서 연기할 수 있다.



[그림 3] TI-Sky 구성도
[Fig. 3] Overview of the proposed method

객체단위의 연산에서는 스카이라인 연산을 위한 비교연산의 수를 최소화한다. 즉, 상위레벨에서 파티션 단위의 지배관계를 비교하여 잠재적 스카이라인에 해당되는 파티션에서의 객체연산을 최소화한다. 객체의 생성과 소멸시간 모두 상위레벨 요약정보에 영향을 미치므로 파티션지배관계의 갱신이 필요하며, 사멸객체 집합에 속하는 객체들과 파티션에 대해서 수행을 미루거나 삭제연산을 연기할 수 있다.

스카이라인 결과 탐사 단계에서는 스카이라인 결과 $S_p(S,t_i)$를 제공한다. 상위레벨 요약정보로부터 스카이라인에 참여하는 파티션들을 탐색한 뒤, 각 파티션에서 객체단위로 지배관계를 비교하여 스카이라인 객체들을 찾는다. 먼저, 스트림 데이터 객체들은 값에 따라 정의 6과 같이 해당범위 파티션으로 분류된다. 정의 4의 객체단위의 시간지배관계와 같이 파티션 간의 시간지배관계를 표시하는 상위레벨 시간지배관계는 정의7과 같다.

정의 6. 파티션.

파티션 P_k(RBoundary, SkyHeap, RestHeap, VBoundary, TBoundary)는 P_k.RBoundary 범위 내의 객체들에 대해 아래와 같은 요약정보들을 관리한다.

RBoundary : d차원의 사각형 범위로 좌측하단경계점 Lower(P_k.RBoundary)와 우측상단경계점 Upper(P_k.Rboundary)로 정의함.

SkyHeap : 파티션 P_k에서 지배되지 않은 객체들의 힙 구조.

RestHeap : P_k.SkyHeap 상의 객체들에 대해 지배관계가 되는 P_k상의 모든 객체들의 힙구조.

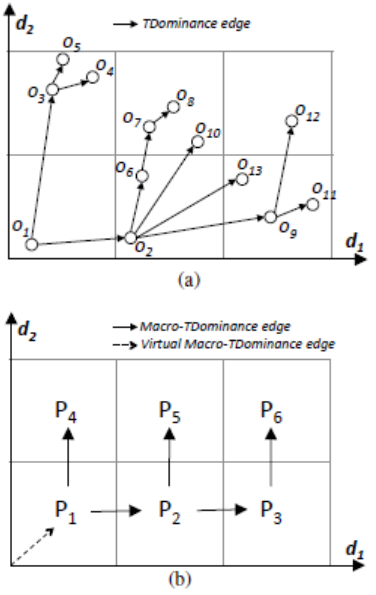
VBoundary : 파티션 P_k내의 객체들의 최소값과 최대값범위를 나타내는 다차원 벡터.

TBoundary : P_k내의 객체의 최소 및 최대 소멸시간.

파티션 시간지배관계는 파티션 내의 객체의 소멸시간을 반영한 지배관계로서 해당 파티션이 어느 시점에 스카이라인 결과에 위치하는지 보여준다. 파티션 지배관계에서 지배하는 파티션은 스카이라인 결과에 항상 포함된다. 따라서, 그러한 파티션을 우선적으로 갱신하고 결과단계에서 처리할 수 있다.

정의 7. 파티션 시간지배관계.

파티션 P_f와 파티션 P_g에 있어서, $\exists o_i \in P_f$, $\exists o_j \in P_g$인 $o_i \rightarrow_T o_j$ 일 경우, P_f가 P_g를 시간지배한다 $P_f \rightarrow_T P_g$.



[그림 4] (a) 객체 시간지배관계
(b) 파티션 시간지배관계 (Macro-TDominance)
[Fig. 4] (a) Object-Level TDominance
(b) Partition-Level TDominance

예제 2. 그림 4에서 $o_3 \in P_4$와 $o_6 \in P_2$의 객체에서 $o_1 \rightarrow_T o_3$와 $o_1 \rightarrow_T o_6$이 성립하므로 파티션 시간지배관계 $P_1 \rightarrow_T P_4$와 $P_1 \rightarrow_T P_2$가 존재한다.

새로 생성된 o_{new}가 파티션 지배관계 중 루트에 해당되

는 파티션에 위치하거나, 루트 파티션에 속하는 기존 객체가 소멸하는 경우, 스카이라인이 갱신될 수 있어 우선적으로 수행되어야 한다. 반면에 루트가 아닌 파티션에서의 연산은 현재의 스카이라인 결과에 바로 영향을 주지 않으므로 수행을 연기할 수 있다. 즉, 스카이라인 결과를 찾는 과정에서 이러한 루트 파티션만을 우선적으로 탐색하면 모든 객체들간의 지배관계를 비교하는 것에 비해 효율적으로 스카이라인 탐색 연산을 수행할 수 있다.

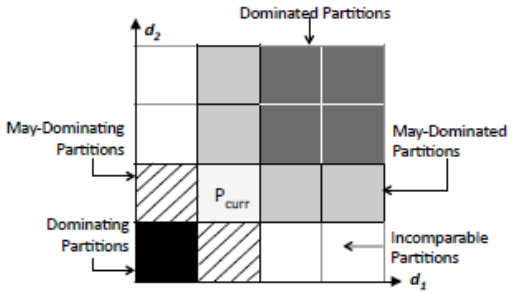
정의 8. 객체시간지배관계.

유효시간 데이터스트림 S와 파티션 P_k상에서, 객체간 시간지배관계는 S에 포함된 모든 객체 간의 관계가 아닌 P_k내에 객체들 간의 시간지배관계로 정의한다.

데이터 스트림에서 실시간으로 생성된 객체 o_{new}에 대해 객체단위에서 그 후 파티션단위에서 갱신이 수행된다. 먼저, o_{new}에 대해 해당범위의 파티션이 탐색된다. 그림 6에서 o_{new}는 P_{curr}에 생성된 후, 파티션내의 객체단위에서 갱신이 수행된다. (a) o_{new}에 대해 P_{curr}에 이미 존재하는 객체들과 비교연산을 수행하여 P_{curr}의 SkyHeap 또는 RestHeap을 갱신한다. (b) 정의 6에서의 P_{curr}의 VBoundary, TBoundary를 갱신한다.

다음으로 o_{new}가 현재 결과 스카이라인을 갱신하는지 또는 앞으로 미래에 스카이라인이 될 것인지 결정한다. [3,4,8]에서와 같이 모든 데이터공간을 탐색하는 대신, 본 논문에서는 상위레벨인 파티션단위의 탐색만 수행한다. 파티션 시간지배관계의 루트로부터 너비우선탐색을 수행하여, P_{curr}로 연결되는 새롭게 갱신되는 시간지배관계를 찾는다. $P_i \rightarrow_T P_{curr}$과 $P_{curr} \rightarrow_T P_j$가 성립되는 P_i와 P_j를 탐색한다. 기존의 객체기반의 연구에서는 o_{new}와 P_i의 객체간의 시간지배관계를 소모적으로 검사를 했지만, TI-Sky에서는 파티션단위의 시간지배관계를 비교한다. 파티션 P_i를 P_{curr}에 따라 그림과 같이 네가지로 분류할 수 있다.

1. 지배파티션 : P_{curr}을 지배하는 파티션 P_i목록
2. 피지배파티션 : P_{curr}에 의해 피지배되는 파티션 P_i목록
3. 잠재적지배파티션 : P_{curr}내에 o_{new}를 지배하는 객체를 포함할 수 있는 P_i목록
4. 잠재적피지배파티션 : P_{curr}내의 o_{new}에 의해 피지배되는 객체를 포함할 수 있는 P_i목록



[그림 5] P_{curr}에 대한 파티션 분류
[Fig. 5] Value-Based Categorization w.r.t. P_{curr}

먼저 파티션시간지배관계 $P_i \rightarrow_T P_{curr}$를 탐색한다. 그림 5의 분류로부터 P_{curr}에 대해 지배파티션 또는 잠재적지배파티션에 해당되는 파티션들이 P_{curr}을 파티션시간지배관계가 될 수 있다.

파티션 P_i가 P_{curr}에 대해 지배파티션 관계에 있는 경우, P_i를 P_{curr}과 파티션시간지배관계의 후보파티션으로 분류한다. 만약 $P_i \rightarrow_T P_{curr}$관계가 이미 존재하는 경우, $P_i \rightarrow_T P_{curr}$에서 o_{new}에 의한 P_{curr}의 소멸시간을 갱신하고 탐색을 종료한다. 반면에, $P_i \rightarrow_T P_{curr}$관계가 존재하지 않는다면, (a) P_{curr}은 객체가 처음 생성된 파티션이거나, 또는 (b) 다른 파티션 P_j가 존재하여 P_j로부터 파티션시간지배관계가 존재하는 경우이다. 이 두 경우에 있어, 파티션 P_i는 P_{curr}과의 파티션시간지배관계가 될 수 있는 후보파티션이 된다. 이러한 후보파티션들을 모두 탐색한 후, 이들 중 P_{curr}과 파티션시간지배관계를 결정한다. 만약 P_i가 P_{curr}의 잠재적지배파티션에 속한다면 P_i.VBoundary로부터 P_i가 P_{curr}를 지배할 수 있는지 비교하고, 아니라면 후보에서 제거할 수 있다.

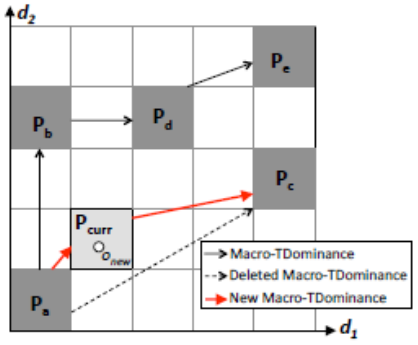
Theorem 1. 새로 생성된 객체 $o_{new} \in P_{curr}$과 P_{curr}에 대해 잠재적지배파티션으로 분류되는 P_i가 존재할 때, $\forall j(o_{new}[j] \leq P_i.VBoundary[j]) \wedge \exists j(o_{new}[j] < P_i.VBoundary[j])$가 성립하는 경우 P_i는 P_{curr}의 파티션시간지배관계 후보파티션에서 제외할 수 있다. 만약 UPPER(P_i.VBoundary) $\rightarrow_T o_{new}$라면 $P_i \rightarrow_T P_{curr}$이다.

Theorem 1의 조건이 성립되지 않는 경우, o_{new}를 시간지배하는 $o_k \in P_i$가 존재하기 때문에 $P_i \rightarrow_T P_{curr}$이 성립된다. 다음으로 다른 후보파티션을 탐사한다. P_{curr}에 대한 후보파티션으로 P_i가 존재할 경우($\exists o_k \in P_i$ $o_k \rightarrow_T o_{new}$), 파티션 P_h ($\exists o_l \in P_h$, $(o_l \rightarrow_T o_{new})) \wedge (o_l.t_{exp} > o_k.t_{exp})$가 존재한다면 $o_l \rightarrow_T o_{new}$가 존재한다. o_{new}를 시간지배하는 o_l에

의해 P_i대신 P_h가 P_{curr}의 파티션지배관계 후보파티션이 된다.

예제 3. 그림 6에서 파티션 지배관계가 $P_a \rightarrow_T P_b$, $P_a \rightarrow_T P_c$, $P_b \rightarrow_T P_d$, $P_d \rightarrow_T P_e$가 존재한다. o_{new}가 P_{curr}에 새로 생성되고, 이를 반영하기 위해서 파티션시간지배관계를 루트인 P_a로부터 탐색된다. P_a는 P_{curr}의 지배파티션에 분류되기 때문에 파티션지배관계 후보파티션이 된다. P_a 이외의 지배파티션 또는 잠재적지배파티션이 없으므로 P_a에서의 P_{curr}로의 시간지배관계 $P_a \rightarrow_T P_{curr}$을 생성한다.

다음으로, P_{curr}에 대해 피지배관계 또는 잠재적피지배관계에 있는 파티션들을 탐색한다. 즉, $P_{curr} \rightarrow_T P_i$ 에 해당되는 P_i를 탐색한다. 앞서 지배파티션과 잠재적지배파티션을 탐색했던 과정과 유사하게 피지배파티션, 잠재적피지배파티션을 결정할 수 있다.



[그림 6] $o_{new} \in P_{curr}$에 의한 파티션 탐색
[Fig. 6] Inserting o_{new} in P_{curr} and Traversing Partitions

예제 4. 그림 6에서 $o_{new}.t_{exp} > P_a.TBoundary$ 이므로 $P_a \rightarrow_T P_c$ 관계는 삭제되고 새로운 파티션시간지배관계 $P_{curr} \rightarrow_T P_c$로 갱신된다.

파티션 탐색에 있어서 불필요한 파티션시간지배관계 탐색을 피하기 위해 Theorem 2를 적용한다. 두 개의 지배관계 후보파티션 P_l, P_{curr}에 대해서 다음 조건을 만족할 경우, 더 이상의 탐색없이 파티션지배관계 갱신과정을 멈출 수 있다.

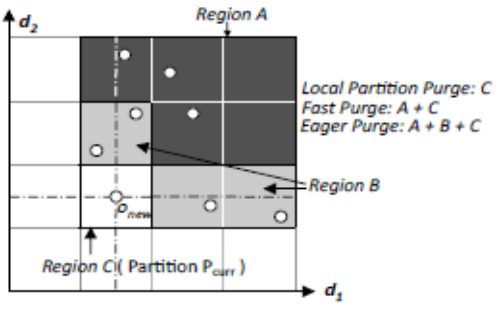
Theorem 2. 파티션지배관계 $P_i \rightarrow_T P_j$가 존재하고 새로 생성된 객체 $o_{new} \in P_{curr}$에 대해서, P_j가 P_i와 P_{curr}의 피지배파티션 또는 잠재적피지배파티션인 경우를 가정한다. $\exists o_k \in P_i$에 대해서 $\exists o_m, o_n \in P_j$ $(o_k \rightarrow_T o_m) \wedge (o_{new} \rightarrow_T o_n)$이고, $(o_k.t_{exp} < o_{new}.t_{exp})$이라면, P_j를 루트로하는 파티션지배관계는 더 이상 탐색할 필요가 없다.

즉, 파티션 P_i 내에 존재하는 객체 o_k가 파티션 P_j 내의 객체에 대해서 시간지배관계가 성립한다면, o_j가 o_{new}보다 소멸시간이 길기 때문에 P_j로부터 연결되는 파티션지배관계탐색은 불필요하다.

객체의 소멸시간이 현재 타임스탬프보다 작을 경우, 해당 객체는 더 이상 유효하지 않다. 최신 스카이라인 결과를 유지하기 위해서는 소멸시간이 지난 객체들은 모두 제거되어야 한다. 예를 들어, $o_a(2,3)$, $o_b(4,3)$, $o_c(3,5)$가 순차적으로 생성되었다고 가정해보자($o_a.t_{exp} < o_b.t_{exp} < o_c.t_{exp}$). o_a가 소멸시간이 지난 경우, o_a가 반드시 제거되어야 o_b와 o_c가 스카이라인 결과가 된다. 이러한 소멸과정을 위해 최소힙(min-heap){h(e):e∈D}을 사용하여 객체의 소멸시간에 따라 정렬한다. o_a가 소멸될 때, 기본적 방법으로는 o_a에 지배되는 모든 객체들을 탐색하여 새로운 스카이라인 객체를 찾는다[3]. 기존의 연구[3,4,8]에서는 각 소멸 때마다 비스카이라인 객체들을 탐색하는 방법을 사용하였다. 하지만, 이러한 방법은 객체의 소멸시간이 고정된 슬라이딩 윈도우 모델[3,8]과 달리 다수의 객체가 동시에 소멸될 수 있는 시간간격 데이터 스트림에서는 비현실적인 방법이다.

본 연구에서는 객체 소멸을 효율적으로 처리하기 위해, 파티션시간지배관계를 사용한다. 파티션시간지배관계 $P_i \rightarrow_T P_j$는 P_j내의 객체가 P_i내의 객체에 의해 시간지배 되었음을 의미한다. 따라서, P_i내의 객체가 소멸되면 P_j의 객체가 스카이라인 결과가 될 수 있다. 파티션시간지배관계는 이와 같이 객체의 소멸순서를 안내할 수 있어, 기존의 객체 소멸 방법에 비해 적은 수의 객체들만을 대상으로 탐색하여 객체들을 소멸시킬 수 있다.

한편, 스카이라인 탐색과정에서 그림 2.b의 객체 o_2와 같은 객체들은 소멸되기 전까지 스카이라인 결과가 되지 못하므로 제거할 수 있다. 이러한 객체들을 사멸객체(dead objects)라 정의한다. 사멸객체는 앞으로 생성된 새로운 객체관리를 위한 메모리공간 활용의 측면에서 미리 제거될 수 있다. 사멸객체를 식별하기 위해서 스카이라인에 존재하는 객체들의 소멸시간과 지배된 객체들의 소멸시간을 비교해야 한다. 본 논문에서는 기존의 단순한 객체 비교 외에 객체들의 위치와 제거 간격에 따라 다음과 같이 방법을 제시한다.



[그림 7] 강제제거 방법 분류
[Fig. 7] Purging Regions

그림 7에서는 P_{curr}에 생성된 o_{new}에 대해서 사멸객체 제거를 위한 파티션을 분류하였다. 컴퓨팅 자원을 소비하지 않는 기준에서 지역객체제거(local purging)를 사용할 수 있고, 반대의 관점에서는 전역객체제거(eager purging)를 사용할 수 있다. 지역객체 제거는 o_{new}를 포함하는 P_{curr} 내의 사멸객체만을 삭제한다. P_{curr}내의 적은 수의 지배관계만을 검사하므로 지역객체 제거 비용은 적다. 그러나, 다른 파티션에 사멸객체들이 누적될 수 있으므로 효율적이라 말하기는 어렵다. 반면에 빠른객체제거(fast purging)에서는 그림7의 지역A와 같이 파티션 범위가 o_{new}에 의해 지배관계에 있는 파티션만 탐색하여 삭제한다. 전역사멸객체 제거에서는 그림 7의 지역 A,B,C와 같이 피지배파티션과 잠재적피지배파티션 모두를 탐색한다. 먼저 빠른객체제거와 같이 지역 A,C의 파티션에 대한 사멸객체 제거를 수행한 뒤 지역 B에 있어서 o_{new}와 객체 단위의 비교를 통해 사멸객체를 판별해서 제거한다.

TI-Sky의 스카이라인 연산은 데이터 객체의 생성, 소멸, 그리고 제거 단계로 수행이 된다. 객체의 생성 단계에서는 새로운 데이터 객체 o_{new}가 해당 파티션에 추가되고 파티션시간지배관계를 갱신한다. 객체 o_i의 소멸시간에 해당될 경우 소멸단계가 수행된다. 파티션시간지배관계에서 o_i가 소멸된 후 어느 파티션의 객체가 스카이라인 결과에 해당되는지 결정해서 객체소멸 연산을 수행할 수 있다. 마지막으로 사멸객체 제거 단계는 수행되는 방법에 따라 o_{new}에 의해 해당 파티션 탐색시, P_{curr}를 기준으로 수행할 수 있다.

스카이라인 결과 연산 단계에서는 주어진 t_i시간에서의 파티션시간지배관계를 탐색한다. 파티션시간지배관계의 루트들은 현재 스카이라인 결과 객체들을 포함하고 있는 파티션들로써 해당 파티션들의 스카이라인 힙 구조 내의 객체들을 탐색한다.

TI-Sky의 복잡도는 새로운 객체 $o_{new} \in P_{curr}$에 의한 시간지배관계 갱신과 객체 소멸에 의한 시간지배관계 갱신의 복잡도가 큰 비중을 차지한다. 이 두 과정에서의 복잡도를 분석해본다. 파티션 내 연산에서는, 새로운 객체에 의해 파티션 내의 스카이라인 객체 수 n_{sky}만큼 비교를 수행하므로 $O(n_{sky})$로 정의할 수 있다. 파티션 내 객체비교 연산이 끝난 후에는 파티션 수 N_p에 대한 $O(N_p)$의 파티션시간지배관계를 탐색한다. o_{new}이 P_{curr}내의 스카이라인 객체가 되었다면, P_{curr}의 파티션지배관계 또한 o_{new}에 의해 갱신될 수 있다. 따라서, 파티션 중 잠재적피지배파티션에 해당되는 파티션 $N_{may\text{-}tdominating}$들을 탐색하여 P_{curr}로부터 새롭게 갱신된 파티션시간지배관계를 생성한다. Theorem 1에 해당되지 않는 $N_{may\text{-}tdominating}$과 파티션 내에 평균 객체 수 n_{obj}에 대해 $O(N_{may\text{-}tdominating} * n_{obj})$의 비교연산을 수행하여 파티션지배관계를 갱신한다. 새로운 객체에 의한 탐색 및 시간지배관계 갱신연산의 복잡도는 $O(n_{sky}+N_p+ (N_{may\text{-}tdominating} * n_{obj}))$가 된다. 다음으로 객체 소멸과정에서는 스카이라인에 해당되는 객체가 소멸될 경우 연산이 이뤄진다. 이 경우 파티션 내에 지배된 객체들로부터 새로운 스카이라인을 연산해야하므로 $O(n_{obj}^2)$의 복잡도를 가진다. 마지막으로 사멸객체 제거 단계에서는 지역사멸객체 제거의 경우, 새로운 객체 추가단계 연산과 동시에 수행할 수 있으므로 그 복잡도는 객체 추가에 의한 복잡도를 넘지 않는다. 따라서 TI-Sky의 복잡도는 $O(N_{may\text{-}tdominatiog} * n_{obj})$이라 할 수 있다.

4. 실험

4.1 실험환경

비교 실험 : 실험에서는 본 논문의 TI-Sky와 데이터스트림 스카이라인 탐사기법인 LookOut, 그리고 Stabbing-the-Sky+와 비교를 수행한다.

실험 세팅 : 모든 알고리즘은 C++로 구현이 되어 윈도우 운영체제를 가진 Intel 2.8GHz Dual Core, 2GB memory 컴퓨팅환경에서 수행하였다.

성능 평가 척도 : 다양한 환경에서의 TI-Sky의 성능을 평가하고 기존의 알고리즘과 비교하기 위해 다음과 같은 비교 척도를 사용한다. (1) 스카이라인 관리 및 결과까지의 평균 객체 처리 비용. (2) 파티션시간지배관계와 파티션 내 객체관리에 사용되는 노드의 수.

가상 실험 데이터 : [2]의 연구에서 활용된 스카이라인 가상데이터를 사용한다. 세가지 데이터 분포, independent, correlated, anti-correlated를 사용하였다. 객체들은 1-100의 범위의 값들이 2-6차원까지 존재한다. [2]의 실험데이터에는 객체의 생성/소멸시간은 고려되지 않았으므로 본

실험을 위해 가상으로 생성하였다. 객체의 유효시간은 100k를 평균으로 분포했다. 즉, 평균적으로 100k개의 객체가 생성된 후에 이전 객체들은 소멸되기 시작한다. 1000k개의 객체로 구성된 데이터 스트림을 대상으로 실험을 수행하였다.

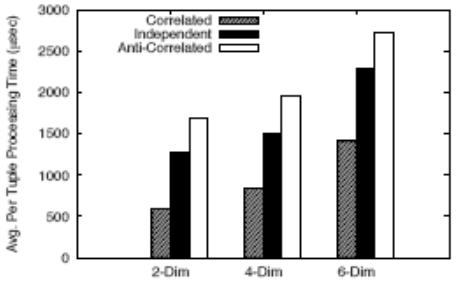
실세계 응용환경 데이터 : Lawrence Berkeley Lab.[9]에서 연구용으로 수집한 TCP 통신 데이터를 사용한다. 로컬네트웍과 인터넷간에 통신으로 발생한 30일간의 TCP 연결에 대한 기록으로, 각 데이터는 로컬/원격 호스트 IP, 프로토콜 타입, 전송용량, 연결 시간 간격 정보를 포함한다. 본 실험에서는 로컬/원격에서의 각 전송용량과 연결 시간 간격 정보를 스카이라인의 분석 차원으로 사용한다.

4.2 TI-Sky의 실험 분석

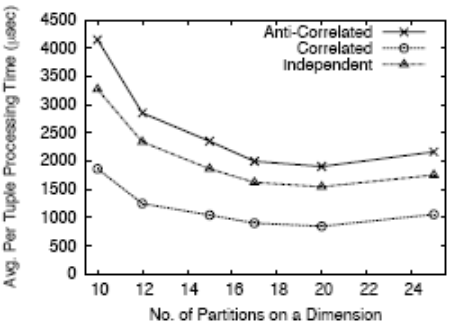
TI-Sky의 성능평가를 위해 다음과 같은 환경으로 실험을 수행한다. (1) 각 차원에서의 파티션의 수, (2) 사멸객체 제거 방법, (3) 차원의 수, (4) 데이터 분포.

스카이라인 데이터 차원의 수 d : 그림 8.a는 차원 수 *d*=2에서 6에 대한 세가지 데이터 분포의 실험을 보여준다. 차원 수 *d*가 증가할수록 많은 수의 파티션이 구성되어 스카이라인을 수행한다.

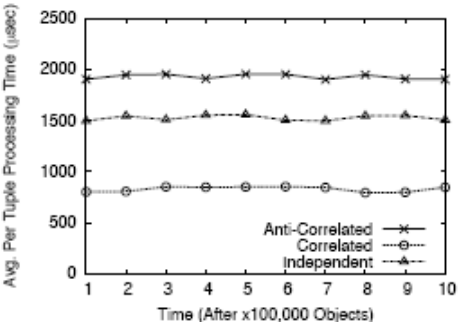
파티션의 수 k : 각 차원에서의 파티션의 수는 각 파티션 내에 저장되는 객체 수에 영향을 미친다. 파티션 수 *k*가 작을수록 많은 수의 객체들이 하나의 파티션 내에 존재하며, 파티션 내의 객체연산에서의 메모리사용량과 CPU사용량이 증가한다. 반면에 *k*가 증가할수록 파티션 내의 객체수가 줄어들어 객체비교연산의 비용이 감소한다. 하지만, 파티션 수가 증가하므로 결과적으로 파티션 지배관계의 수가 증가하여 상위레벨에서의 연산이 많이 발생한다. 그림 8.(a)는 차원축을 10에서 25까지 분할했을 경우 TI-Sky의 평균 투플 처리시간을 나타낸다. 기존의 세 가지 데이터 분포를 대상으로 실행하였다. 그림 8.(b)에서 k=20 부근에서 그래프가 반전하는 것을 볼 수 있다.



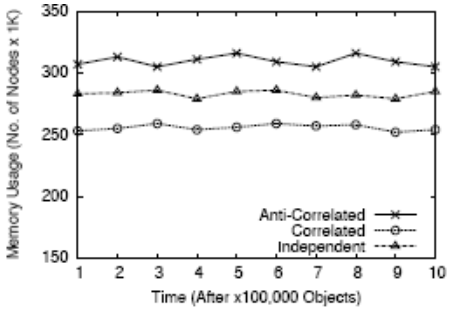
(a) 차원 수에 따른 수행시간



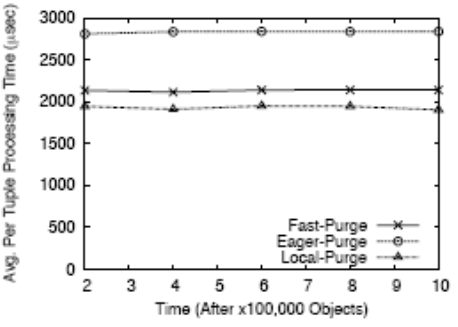
(b) 파티션 수에 따른 수행시간



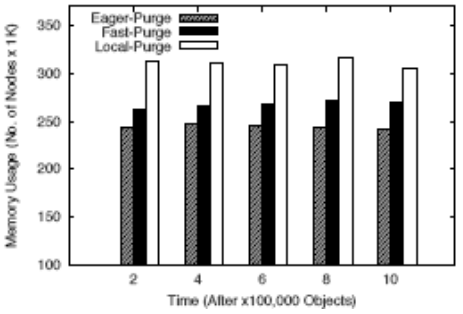
(c) 데이터 분포에 따른 수행시간



(d) 데이터 분포에 따른 메모리 사용량



(e) 강제제거에 의한 수행시간



(f) 강제제거에 의한 메모리 사용량

[그림 8] TI-Sky 성능 실험

[Fig. 8] Performance Evaluation of TI-Sky

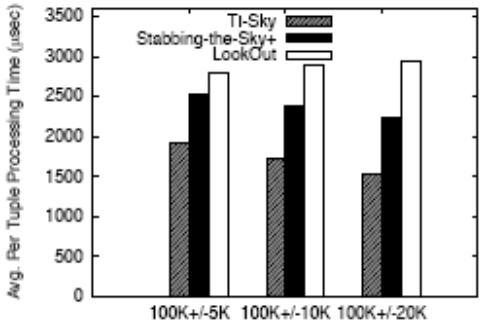
데이터분포 : 그림 8.(c-d)에서는 데이터 분포에 따른 투플 처리시간과 메모리 사용량을 실험하였다. correlated 분포에서는 다수의 객체들이 소수의 스카이라인 객체에 의해 지배된다. 따라서, 다수의 객체들이 사멸객체로 제거되고 적은 컴퓨팅 자원으로 스카이라인 탐사를 수행할 수 있다. 하지만, anti-correlated 분포에서는 스카이라인 객체가 다수 분포하므로 CPU와 메모리자원의 사용량이 증가한다.

사멸객체 제거 : 그림 8.(e-f)에서는 사멸객체 제거 방법에 따른 투플 처리시간과 메모리 사용량의 변화를 보여준다. 전역제거 방법에서는 가능한 모든 파티션을 탐색하여 제거하므로 가장 적은 메모리 사용량을 보여준다. 반대로, 투플 처리시간에 있어서는 지역제거 방법이 가장 효율적인 수행시간을 보여준다. 새로운 객체가 생성되었을 때, 해당 파티션 내의 객체만 제거하는 것이 다른 파티션에 대한 별도의 탐색이나 연산없이 효율적으로 수행할 수 있기 때문이다.

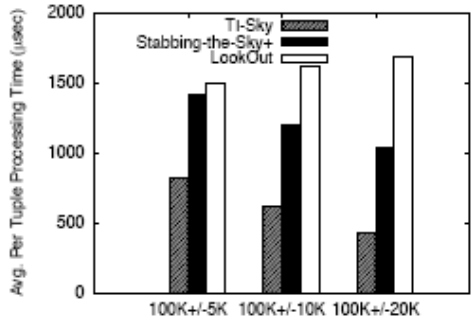
4.3 성능 비교 실험

다음과 같은 변수들의 변화에 따른 투플 처리시간을 비교한다. (1) 각 객체의 유효시간 크기의 분포, (2) 스카이라인 질의 빈도, (3) 데이터 공간의 차원 크기

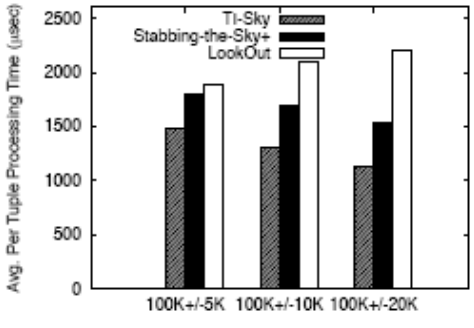
스트림 객체의 시간간격 : 그림 9에서는 객체들의 유효시간 간격을 평균 100K, 표준편차 5-20K으로 실험을 하였다. 즉, 모든 객체들은 100K를 기준으로 표준편차 범위내의 유효시간 간격을 가진다. anti-correlated 분포 데이터에서 TI-Sky는 Stabbing-the-sky+, LookOut과 비교해서 38%, 69%빠른 수행결과를 보인다. 그리고 correlated 분포에서는 TI-Sky는 2배까지 빠른 수행속도를 보이며, inpedendent 분포 데이터에서 TI-Sky는 각각 29%, 60% 빠른 결과를 보여준다. 객체 유효시간의 편차가 증가할수록 TI-Sky와 Stabbing-the-sky+의 객체 처리 시간은 빨라진다. LookOut[4]에서는 사멸객체 강제제거 과정 없이 모든 객체들을 유지하기 때문에 보다 많은 처리시간이 요구된다.



(a) anti-correlated 분포에서의 성능비교



(b) correlated 분포에서의 성능비교



(c) independent 분포에서의 성능비교

[그림 9] 성능 비교 실험

[Fig. 9] Performance Comparison

질의 빈도에 의한 성능 변화 : 그림 10.(a)에서는 스카이라인 질의 빈도에 따른 성능변화 결과를 보여준다. 즉, "x"개의 객체가 처리될 때마다 주기적으로 스카이라인 질의를 수행하였다. 실험에서의 질의 빈도는 1000개의

객체부터 0으로 변화하였으며, 0일 때는 스카이라인 질의를 수행하지 않았다. 차원의 수 d=4일 때, 스카이라인 객체의 수가 가장 많은 anti-correlated 분포에서 실험을 수행하였다. 그림 10.(a)와 같이 매번 객체마다 질의를 수행할 때 가장 수행비용이 증가한다. 반면에, 질의 수행 빈도 변화에 비해 처리시간이 일정하게 유지된다. 즉, 스카이라인 질의 처리 시간은 스카이라인 알고리즘의 데이터 관리 비용에 비해 미소하여 큰 영향을 주지 않는다.

실세계 응용 데이터 환경 : TCP 데이터에서 TI-Sky는 기존의 방법들보다 향상된 성능을 보여준다. 특히, 그림 10.(b)에서 TI-Sky는 Stabbing-the-Sky+와 LookOut과 비교해서 각각 16%, 20% 빠른 성능을 보인다.

5. 결론

기존의 스카이라인 기법들은 데이터 스트림을 대상으로 슬라이딩 윈도우 내의 객체에 대한 스카이라인 탐사를 수행하였다. 하지만, 슬라이딩 윈도우는 유효시간 데이터 스트림에 비해 객체들의 소멸시간이 고정된 보다 단순화된 응용환경이다. 유효시간 데이터 스트림 연구에서 객체마다 다른 소멸시간을 관리하기 위해 다수의 다차원 인덱스를 동시에 갱신하지만, 빈번한 객체간의 비교연산으로 효율적이지 못하다. 본 연구에서는 유효시간 데이터 스트림에서 파티션단위의 지배관계를 이용한 스카이라인 탐사 방법을 제시한다. 기존 연구들이 객체 기반의 지배관계만을 고려하였지만, 본 연구에서는 스트림 상에서의 소멸시간과 다차원값을 동시에 반영하여, 객체간의 연산보다 상위레벨에서 파티션 간의 시간지배관계를 구축하여 스카이라인을 탐사한다. TI-Sky는 기존의 연구보다 데이터 스트림에서 효율적으로 스카이라인을 갱신하고 실시간 탐사를 실행하며, 이를 다양한 데이터 상에서 실험으로 증명하였다.

References

[1] R. S. Barga, J. Goldstein, M. H. Ali, and M. Hong. “Consistent streaming through time: A vision for event stream processing”, In CIDR, pages 363-374, 2007.

[2] S. B¨orzs¨onyi, D. Kossmann, and K. Stocker. “The skyline operator”, In ICDE, pages 421-430, 2001.

[3] X. Lin, Y. Yuan,W.Wang, and H. Lu. “tabbing the sky: Efficient skyline computation over sliding windows”, n ICDE, pages 502-513, 2005.

[4] M. Morse, J. Patel, and W. Grosky. “Efficient continuous skyline computation", Inf. Sci., pages 3411-3437, 2007.

[5] K. Mouratidis, S. Bakiras, and D. Papadias. "Continuous monitoring of top-k queries over sliding windows", In SIGMOD, pages 635-646, 2006.

[6] D. Papadias, Y. Tao, G. Fu, and B. Seeger. "An optimal and progressive algorithm for skyline queries", In SIGMOD, pages 467-478, 2003.

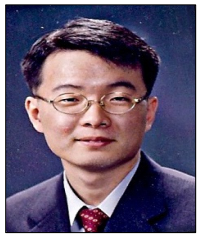
[7] U. Srivastava and J. Widom. "Flexible time management in data stream systems", In PODS, pages 263-274, 2004.

[8] Y. Tao and D. Papadias. "Maintaining sliding window skylines on data streams", TKDE (18:2), pages 377-391, 2006.

[9] Z. Zhang, R. Cheng, D. Papadias, and A. K. H. Tung. "Minimizing the communication cost for continuous skyline maintenance", In SIGMOD, pages 495-508, 2009.

[10] Shiming Zhang, N. Mamoulis, and D. W. Cheung. "Scalable skyline computation using object-based space partitioning", In Proc. SIGMOD, pp. 483-494, 2009.

박 남 훈(Nam Hun Park) **[정회원]**



- 2002년 2월 : 연세대학교 대학원 컴퓨터과학과 (공학석사)
- 2007년 8월 : 연세대학교 대학원 컴퓨터과학과 (공학박사)
- 2007년 9월 ~ 2008년 8월 : 연세대학교 데이터베이스 국가지정 연구실 연구원
- 2008년 9월 ~ 2010년 2월 : Worcester Polytechnic Institute Research Associate
- 2010년 3월 ~ 현재 : 안양대학교 컴퓨터학과 교수

<관심분야>
데이터베이스, 스카이라인 질의처리, 실시간 질의처리

장 중 혁(Joong-Hyuk Chang) **[정회원]**

- 1996년 2월 : 연세대학교 컴퓨터과학과 (이학사)
- 1998년 8월 : 연세대학교 컴퓨터과학과 (공학석사)
- 2005년 8월 : 연세대학교 컴퓨터과학과 (공학박사)
- 2006년 1월 ~ 2008년 7월 : UIUC, Wright State University 박사후 연구원
- 2008년 9월 ~ 현재 : 대구대학교 컴퓨터IT공학부 교수

<관심분야>
데이터 스트림, 데이터 마이닝, 데이타베이스