데이터 스트림 처리를 위한 윈도우 메모리 재배치의 비용 분석

Cost Analysis of Window Memory Relocation for Data Stream Processing

이상돈
국립목포대학교 공과대학 정보공학부
Sangdon Lee(sdlee@mokpo.ac.kr)

요약

본 논문에서는 데이터 스트림 환경에서 윈도우 기반 연산자를 대상으로 메모리와 연산 비용의 상대적인 이해득실 관계를 분석한다. 이를 위하여 기본적인 연산자 네트워크 구성 요소를 식별하고, 윈도우 메모리의 재배치를 통한 메모리 소요량의 감소 효과와, 이로 인한 추가적인 연산 비용의 규모를 산정하는 비용 모델을 수립한다. 이러한 비용 모델을 통해 윈도우 메모리의 재배치의 효용성을 확인하고, 이러한 접근 방법을 데이터 스트림 질의의 실행 계획 개선을 위해 효과적으로 활용할 수 있는 방법을 모색한다.

이를 통해 데이터 스트림 환경에서 질의 처리 및 최적화의 적용 영역을 확장시키고, 윈도우 메모리 재배치를 통한 질의최적화를 위한 비용 산정 모델의 토대를 제공한다.

■ 중심어 : | 데이터 스트림 | 윈도우 연산자 | 메모리 재배치 | 비용 모델 |

Abstract

This paper analyzes cost tradeoffs between memory usage and computation for window-based operators in data stream environments. It identifies generic operator network constructs, and sets up a cost model for the estimation of the expected memory reduction and the computation overheads when window memory relocations are applied to each operator network construct. This cost model helps to identify the utility of window memory relocations. It also helps to apply window memory relocation to improve a query execution plan to save memory usage.

The proposed approach contributes to expand the scope of query processing and optimization in data stream environments. It also provides a basis to develop a cost estimation model for the query optimization using window memory relocations.

■ keyword : | Data Stream | Window Operator | Memory Relocation| Cost Model |

I. 서 론

수많은 센서로 부터의 데이터 값을 모니터링하는 등 최근 들어 일련의 연속적인 데이터 입력을 대상으로 지속적인 질의의 처리를 필요로 하는 데이터 스트림 응용의 필요성 및 적용 분야의 확장이 요구되고 있다[1][2]. 이러한 데이터 스트림 환경에서 질의의 지속적인 처리를 위해서는 윈도우 연산의 적용이 필수적이다[3][5][9]. 윈도우 연산이란 일정 기간(또는 일정량의 데이터 입력)을 대상으로 수행되는 연산을 의미하며, 예를 들어 "최근 5

* 본 논문은 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구입니다.(R05-2004-000-11785-0)

접수번호 : #080328-002
접수일자 : 2008년 03월 28일
심사완료일 : 2008년 04월 08일
교신저자 : 이상돈, e-mail : sdlee@mokpo.ac.kr

분 동안 측정된 온도의 평균값을 센서별로 출력"하거나, 또는 "두 센서로부터의 최근 100개의 관찰 값 중 같은 온도 측정 값을 모두 출력" 하는 등의 질의 예를 들 수 있다. 윈도우 연산에서는 요구되는 연산의 처리를 위해 요구되는 데이터의 보관을 위한 메모리 공간을 필요로 한다[5][9].

최근 들어 메모리의 용량이 비약적으로 발전하고 있으며, 점진적으로 단위 메모리 당 비용이 하락하고 있다. 하지만 메모리에 비해서 CPU 속도의 발전은 비약적이며, 연산능력의 확보는 상대적으로 메모리 보다 비용 측면에서 경제적이다. 데이터 스트림을 대상으로 하는 모니터링 환경과 같은 데이터 스트림 응용에서는 매우 많은(수만개 또는 수십만개등) 질의가 동시에 수행되는 것이 가능하며[8], 이러한 복잡한 데이터 스트림 응용의 경우처럼 메모리에 대한 경쟁이 데이터 처리 환경에서 상존한다. 메모리에 대한 질의간의 경쟁이 심하고, 계산 능력의 비용이 상대적으로 저렴한 환경에서 메모리를 필요로 하는 윈도우 연산자에서의 윈도우 메모리를 연산자 네트워크 상에서 재배치하면 질의에서의 메모리의 요구량이 변화하게 된다. 한편 윈도우 메모리 재배치는 질의의 수행을 위해 요구되는 상대적인 연산 양의 변화를 가져올 수 있다.

본 논문에서는 윈도우 연산자를 대상으로 윈도우 메모리의 재배치를 통한 메모리 소요량의 감소 효과와, 이로 인한 추가적인 연산 비용 측면에서의 비용 발생 요소 및 규모를 파악하므로써 데이터 스트림 환경에서 메모리와 연산 량의 상대적인 이해득실 관계를 식별하는 것을 목표로 한다. 이를 위해 윈도우 메모리 재배치 비용을 분석적으로 예측 가능한 모델을 수립하고, 이를 통해 윈도우 메모리 재배치가 데이터 스트림 응용을 위한 질의 실행 계획의 개선에 매우 효과적인 경우를 식별하므로써 적용성을 증대시키도록 한다. 이를 통해 데이터 스트림 환경에서 질의 처리 및 최적화의 범위를 확장시키고, 윈도우 메모리 재배치를 통한 질의 최적화를 위한 비용 산정 모델 개발의 토대를 제공한다.

II. 기본 개념

데이터 스트림 응용에서 사용자의 데이터 접근 요구는 질의 인터페이스를 통해 이루어지며, SQL을 확장하거나[6][9] 또는 작업 흐름 형식의 연산자 네트워크를 사용하여 표현[7][8]된다. 이러한 질의 인터페이스는 사용자의 요구사항을 표현하는 표현력에 있어서는 큰 차이가 없으며 데이터 스트림 관리 시스템 내부적으로는 일련의 연산자의 네트워크로 표현된다. 데이터 스트림이 갖는 무한성으로 인한 연산자의 블록킹 현상을 해결해야 하며[4], 이를 해결하는 기본적인 접근 방법은 전체 데이터 스트림을 대상으로 연산을 수행하지 않고 데이터 스트림의 일부만을 대상으로 연산을 수행하도록 하는 것이다. 이를 위한 방법으로서 데이터 스트림 관리 시스템[4-6][8-9]들은 데이터 스트림을 위한 윈도우 개념을 적용하고 있다. 윈도우는 끝점의 이동 방향에 따라 고정 윈도우, 이동 윈도우, 랜드마크 윈도우 등으로 구분되며, 윈도우의 설정 기준에 따라 시간 기반 윈도우, 또는 투플 기반 윈도우로 구분된다[5][9].

윈도우를 사용하는 연산들은 연산의 의미있는 수행을 위해 윈도우의 크기에 따라 데이터 스트림 값을 유지해야만 한다. 만일 투플 기반의 윈도우를 사용한다고 가정하고 윈도우의 크기가 100이라면, 100개의 투플을 해당 윈도우 연산을 위해 유지해야 한다.

윈도우 메모리 재배치는 윈도우 연산을 위해 유지되어야 하는 투플의 수를 감소시킬 목적으로 윈도우 연산자의 윈도우 메모리를 연산자 네트워크 상에서 앞쪽으로 이동시키는 것을 말한다. 물론 이러한 윈도우 메모리의 이동을 통해 감소되는 보관될 투플들은 연관된 연산들을 한번 또는 여러번 부가적으로 실행시킴으로써 다시 생성되어야만 한다. [그림 1]은 이러한 윈도우 메모리 재배치의 개념을 보여준다.

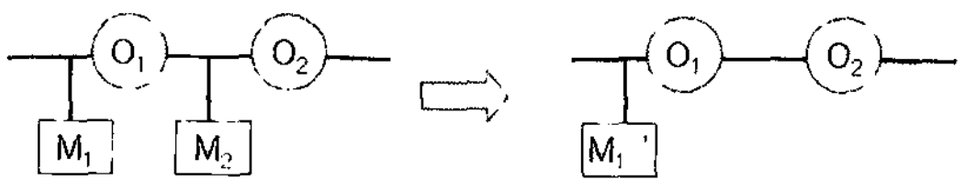


그림 1. 윈도우 메모리 재배치의 예

[그림 1]에서 O_1 과 O_2는 윈도우 연산(예를 들면 각 윈도우에 대해서 평균을 구하는)을 의미한다. 또한 M_1, M_2는 O_1 과 O_2를 위한 각각의 윈도우 메모리를 나타낸다. M_1'는 윈도우 메모리가 재배치된 후 O_1을 위해 필요한 윈도우 메모리를 나타낸다. 여기서 만일 O_1의 윈도우 크기가 2투플이고, O_2의 윈도우 크기가 3투플이라면, 이 네트워크의 전체 윈도우 메모리 소요량은 5 투플이 된다. 그러나 O_2 앞의 윈도우 메모리를 O_1 앞으로 이동시킬 수도 있으며, 이 경우 새로운 윈도우 메모리의 크기는 4가 된다. 윈도우 메모리가 재배치된 후 O_1 연산자로부터의 결과를 사용하여 O_2 연산을 수행하기 위해서는, 2개의 과거 투플이 다시 생성되어야만 하며, 이를 위해서는 O_1이 추가적으로 2번 더 호출되어야 한다. 결국 O_2의 윈도우 메모리를 O_1의 윈도우 메모리와 합쳐서 얻는 1 투플의 윈도우 메모리 감소량은 2번의 추가적인 O_2 연산자 실행을 통해서 얻어진 것이다.

데이터 스트림 질의를 연산자 네트워크로 표현하는 Aurora 데이터 스트림 관리 시스템[7][8]에서는 고유한 질의 대수인 SQuAl에서 정의한 스트림 연산자들을 대상으로 연산자들을 결합하거나, 순서를 바꾸는 등의 정적인 질의 최적화 방법, 그리고 동적인 질의 최적화 방법으로서 부하 감쇄, 연산자의 스케쥴링 방법 등 다양한 데이터 스트림 질의 처리 방법을 제시하였다. STREAM[5][6]이나 TelegraphCQ와 같은 다른 데이터 스트림 관리 시스템에서도 다양한 데이터 스트림 질의 처리 방법의 연구가 진행되었으나, 데이터 스트림 질의 최적화의 대상으로서 윈도우 메모리의 재배치를 다루고 있지는 않다.

본 논문에서 제안하고 있는 연산자 메모리의 재배치를 위한 접근 방법은 다음과 같은 가정을 기반으로 한다. 첫째, 윈도우 메모리 재배치는 네트워크를 통하지 않는 단일 시스템을 대상으로 한다. 둘째, 데이터가 연산자 네트워크에 전달될 때 요구되는 큐로부터의 입출력 비용은 고려하지 않는다. 셋째, 연산자의 구현 시 피연산자는 연산자의 요청에 의해 얻어지는 요구 기반 방식을 사용한다.

III. 윈도우 메모리 재배치의 비용 분석

이제 데이터 스트림을 위한 질의가 연산자의 네트워크로 표현되고, 투플 기반의 윈도우를 사용하는 것으로 가정하여 윈도우 메모리 재배치의 비용을 분석한다. 이러한 가정은 궁극적으로 모든 데이터 스트림 질의를 의미상으로 동등한 내부적 연산자의 네트워크로 표현하는 것이 가능하고, 시간 기반 윈도우 역시 투플 기반 윈도우의 확장으로 볼 수 있기 때문에 유의미한 가정이라 판단된다. 윈도우 메모리 재배치의 효과적인 비용 분석을 위해서 연산자 네트워크를 구성하는 기본 구성 요소를 대역폭의 변화가 없는 경우를 대상으로 직렬형, 분기확대형, 분기축소형의 세가지, 그리고 대역폭변화형 등 총 네가지로 나누었으며, 이들을 대상으로 윈도우 메모리 재배치의 비용을 분석하였다. 또한 각 경우에 대해서 다음의 두가지로 구분하여 비용을 분석하였다. 첫번째는 오른쪽 연산자가 윈도우 연산자이고 왼쪽 연산자는 윈도우 연산자가 아닌 경우이며, 두번째는 모든 연산자가 윈도우 기반 연산자인 경우이다. 비용의 분석 및 비교를 위해 각 경우에 대해서 초기 상태와 윈도우 메모리 재배치의 실행 후 상태, 각각에 대해서 메모리 비용과 연산 수행 비용을 도출하였다. 비용 분석을 위해 사용된 주요 기호들의 의미는 [표 1]과 같다.

표 1. 주요 기호 요약

기호	의미
M_i	윈도우 메모리 i의 구분자
$W(M_i)$	윈도우 메모리 i 의 크기
O_i	연산자 i
$C(O_i)$	연산자 i를 실행시키기 위한 계산 비용

1. 직렬형 네트워크

연산자들이 직렬로 연결된 경우로서 유형 I과 II에 대한 윈도우 메모리 재배치 이전과 이후의 연산자 네트워

크 상태는 [그림 2]와 같으며, 연산자 O_2의 앞에 위치하던 윈도우 메모리 M_2가 O_1의 앞으로 이동하는 경우를 나타낸다.

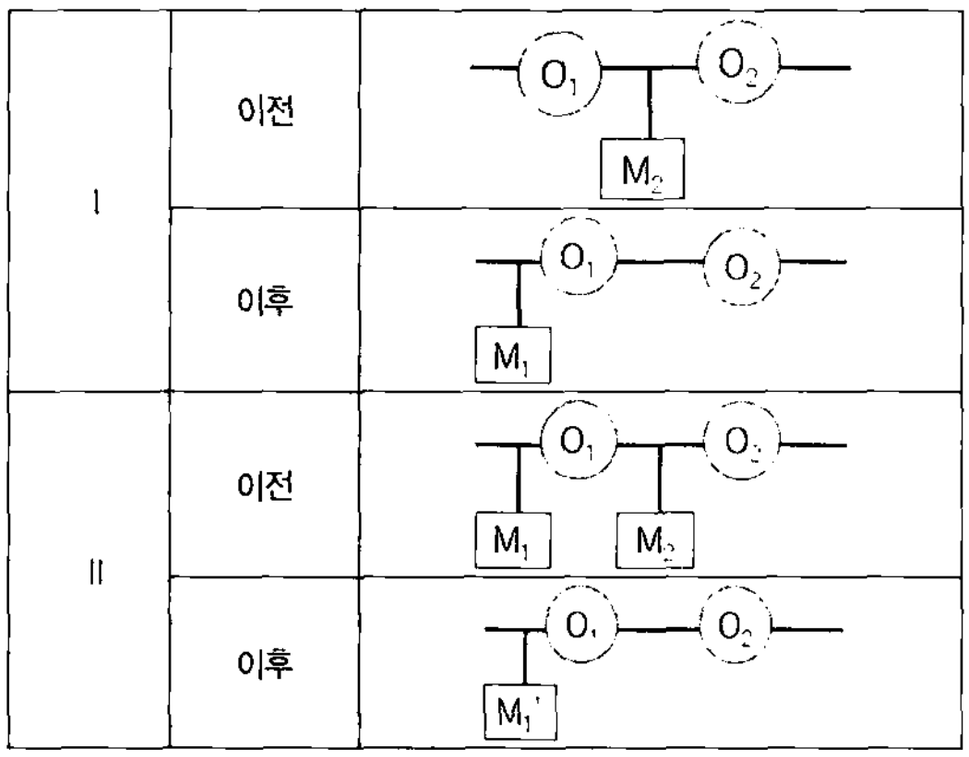


그림 2. 직렬형 네트워크의 변환

각 경우의 메모리 비용과 연산 비용은 다음 [표 2]와 같다. O_2 앞의 메모리를 O_1 앞으로 재배치 시킴에 따라 연산을 수행하기 위해서는 M_2의 크기 만큼 O_1 연산을 반복 실행해야 함을 알 수 있으며, 메모리 비용은 II번 유형에서 소량 감소함을 알 수 있다.

표 2. 직렬형의 연산 및 메모리 비용 분석

유형	재배치	연산 비용	메모리 비용
I	이전	$C(O_1)+C(O_2)$	$W(M_2)$
	이후	$W(M_2)*C(O_1)+C(O_2)$	$W(M_1)=W(M_2)$
II	이전	$C(O_1)+C(O_2)$	$W(M_1)+W(M_2)$
	이후	$W(M_2)*C(O_1)+C(O_2)$	$W(M_1')=W(M_2)+W(M_1)-1$

2. 분기확대형 네트워크

한 연산자의 실행 결과가 다른 여러 연산자로 동시에 분배되는 경우로서 윈도우 메모리 재배치 이전과 이후의 연산자 네트워크 상태는 [그림 3]과 같다. 연산자 O_2, O_3의 앞에 위치하던 윈도우 메모리 M_2, M_3가 각각 O_1의 앞으로 이동하는 경우를 나타낸다.

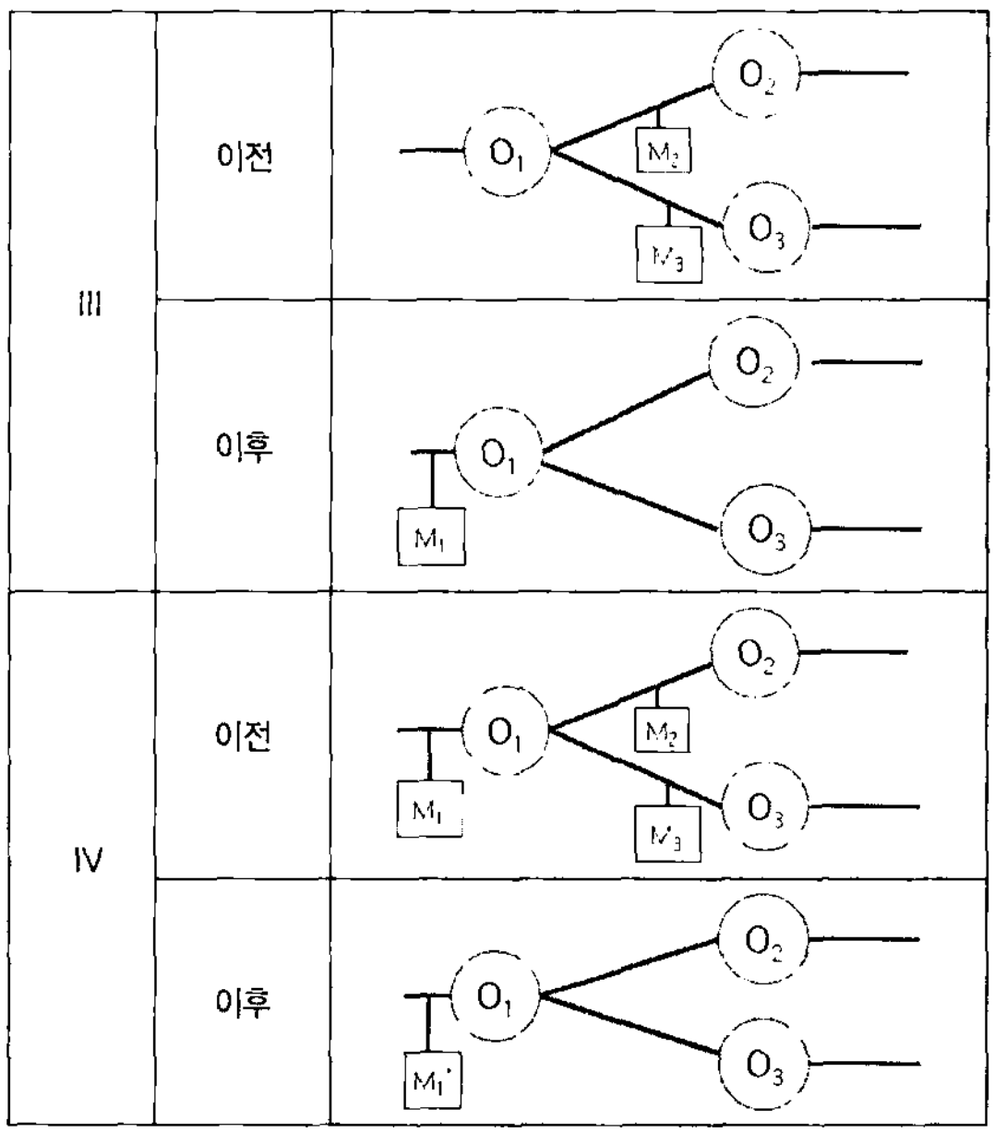


그림 3. 분기확대형 네트워크의 변환

각 경우의 메모리 비용과 연산 비용은 다음 [표 3]과 같다. 흥미로운 사실은 분기확대형의 경우 O_2와 O_3의 윈도우 메모리를 연산자 네트워크의 앞단으로 재배치 할 때 두 윈도우 메모리의 공통 부분 만큼 메모리 소비가 감소한다는 것이다. 분기확대형의 경우 분기 차수가 2보다 클 수 있으며, 분기 차수가 클수록 메모리 감소 효과는 커진다. 연산 비용에서 O_1이 한번 수행되면 O_2와 O_3에 결과가 동시에 전달되므로 각 분기점에서 재배치 이전의 비용은 분기 차수만큼 O_1의 연산 비용을 분담하는 것을 알 수 있다.

표 3. 분기확대형의 연산 및 메모리 비용 분석

유형	재배치	연산 비용	메모리 비용
III	이전	$C(O_1)+C(O_2)+C(O_3)$	$W(M_2)+W(M_3)$
	이후	$(W(M_2)+W(M_3))*C(O_1)+C(O_2)+C(O_3)$	$W(M_1)=MAX(W(M_2),W(M_3))$
IV	이전	$C(O_1)+C(O_2)+C(O_3)$	$W(M_1)+W(M_2)+W(M_3)$
	이후	$(W(M_2)+W(M_3))*C(O_1)+C(O_2)+C(O_3)$	$W(M_1')=MAX(W(M_2),W(M_3))+W(M_1)-1$

3. 분기축소형 네트워크

한 연산자의 실행 결과가 다른 여러 연산자로 동시에 분배되는 경우로서 윈도우 메모리 재배치 이전과 이후의 연산자 네트워크 상태는 [그림 4]와 같다. 연산자 O_4의 앞에 위치하던 윈도우 메모리 M_4가 각각 O_3의 앞으로 이동하여 별개의 윈도우 메모리로 분리되는 경우를 나타낸다.

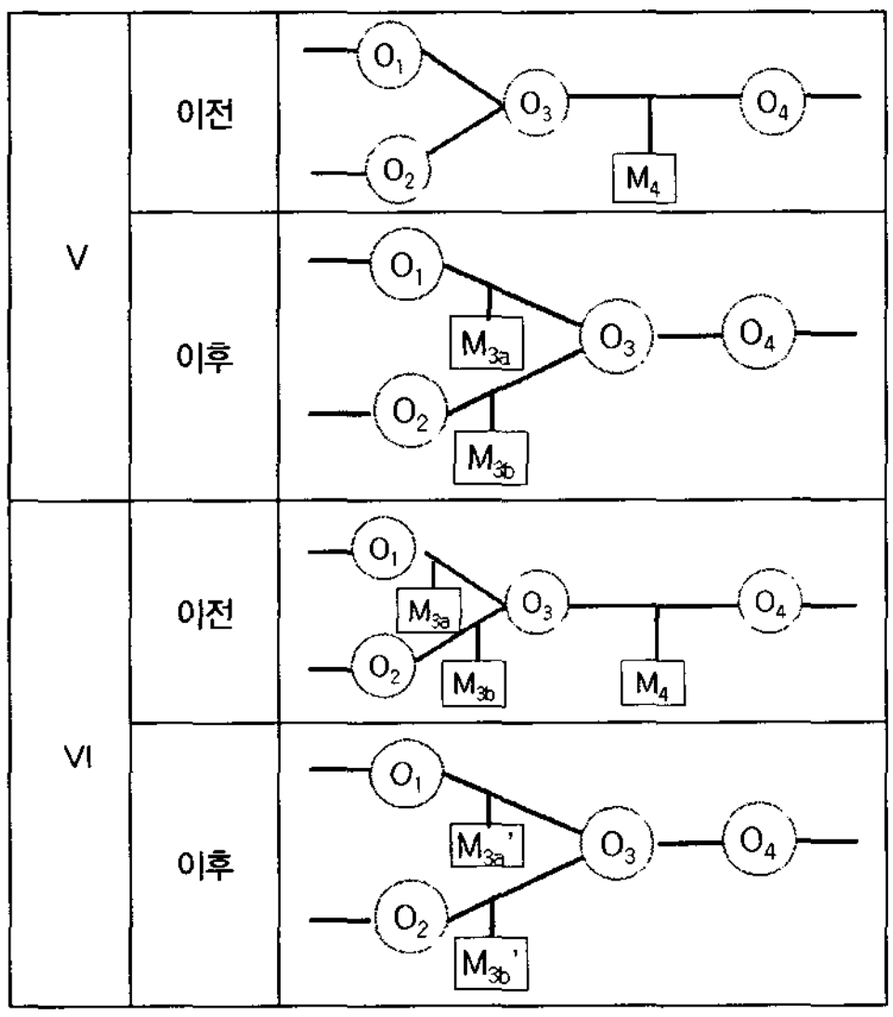


그림 4. 분기축소형 네트워크의 변환

각 경우의 메모리 비용과 연산 비용은 다음 [표 4]와 같다. 분기축소형 단위 네트워크 요소만으로는 윈도우 메모리 재배치를 통해 연산과 메모리의 비용이 증가하는 것을 알 수 있다.

표 4. 분기축소형의 연산 및 메모리 비용 분석

유형	재배치	연산 비용	메모리 비용
V	이전	$C(O_1)+C(O_2)$ $+C(O_3)+C(O_4)$	$W(M_4)$
	이후	$C(O_1)+C(O_2)$ $+W(M_4)*C(O_3)+C(O_4)$	$W(M_{3a})=W(M_4)$ $W(M_{3b})=W(M_4)$
VI	이전	$C(O_1)+C(O_2)$ $+C(O_3)+C(O_4)$	$W(M_{3a})+W(M_{3b})+W(M_4)$
	이후	$C(O_1)+C(O_2)$ $+W(M_4)*C(O_3)+C(O_4)$	$W(M_{3a}')=$ $W(M_4)+W(M_{3a})-1$ $W(M_{3b}')=$ $W(M_4)+W(M_{3b})-1$

4. 대역폭변화형 네트워크

직렬형 네트워크와 유사하나 연산자의 수행 결과 연산자의 입력 대역폭(b_1)과 출력 대역폭(b_2)의 차이가 발생하는 경우를 의미한다. 연산자가 입력보다 많은 투플을 발생시키면 $b_1 < b_2$이며, 출력이 감소되는 연산자인 경우에는 $b_1 > b_2$가 된다. 이러한 대역폭 변화형 네트워크는 지금까지 기술하였던 직렬형, 분기확대형 및 분기 축소형 네트워크와 결합되어 다양한 네트워크 유형을 표현하는데 사용될 수 있다. 윈도우 메모리 재배치 이전과 이후의 연산자 네트워크 상태는 [그림 5]와 같으며, 연산자 O_2의 앞에 위치하던 윈도우 메모리 M_2가 O_1의 앞으로 이동하는 경우를 나타낸다.

각 경우의 메모리 비용과 연산 비용은 다음 표 5와 같다. 분석 결과에서 보듯이 메모리 비용은 입력 대역폭과 출력대역폭의 비율(즉, b_1 / b_2)에 의존적이며, 출력대역폭이 입력대역폭보다 클수록 윈도우 메모리 재배치 후 메모리 비용의 감소 효과가 증가함을 알 수 있다. 이러한 현상은 입력 데이터 사이의 값을 보간법(interpolation) 등을 사용하여 추가적으로 생성하는 경우, 또는 두 입력 스트림의 조인 등으로 발생되는 결과 투플의 양이 증가하는 경우 등에 발생이 가능하다.

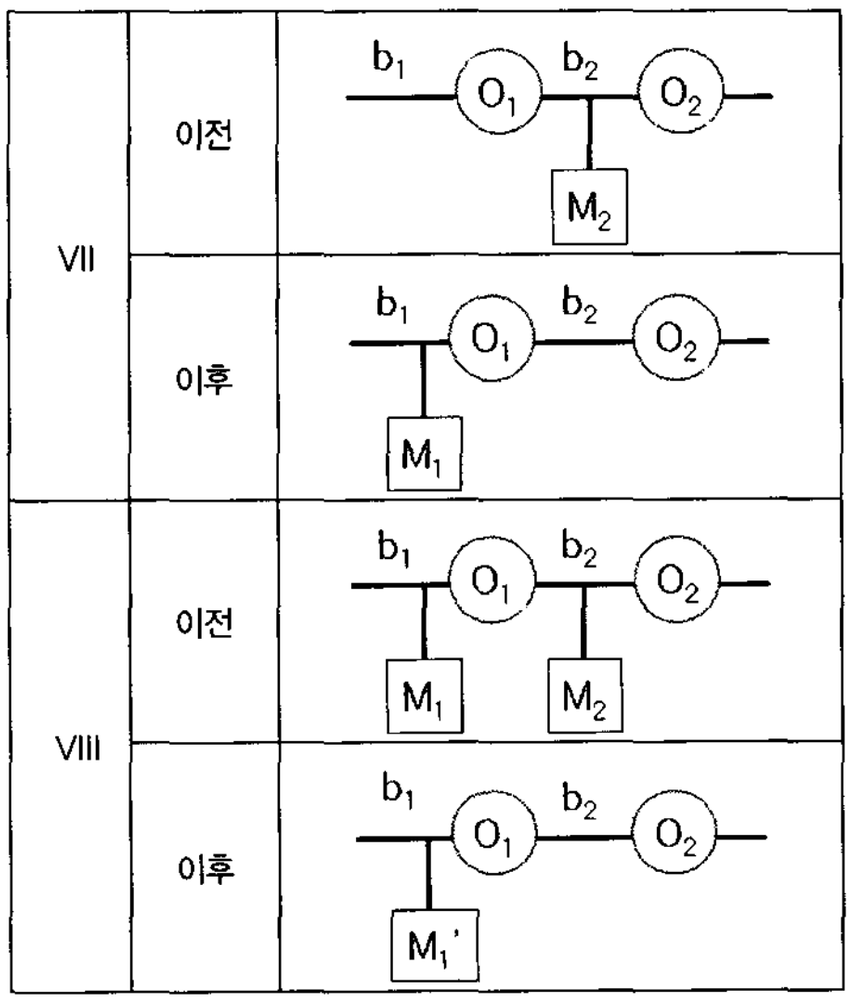


그림 5. 대역폭변화형 네트워크의 변환

표 5. 대역폭변화형의 연산 및 메모리 비용 분석

유형	재배치	연산 비용	메모리 비용
VII	이전	$C(O_1)+C(O_2)$	$W(M_2)$
	이후	$W(M_2)*C(O_1)+C(O_2)$	$W(M_1)=\frac{b_1}{b_2}*W(M_2)$
VIII	이전	$C(O_1)+C(O_2)$	$W(M_1)+W(M_2)$
	이후	$W(M_2)*C(O_1)+C(O_2)$	$W(M_1')-\frac{b_1}{b_2}*(W(M_2)+W(M_1)-1)$

IV. 윈도우 메모리 재배치의 활용 방안

이상에서 기술한 윈도우 메모리 재배치의 비용 분석을 바탕으로 데이터 스트림 관리 시스템의 메모리 비용을 감소시키기 위해 연산자 네트워크의 윈도우 메모리 할당을 제어할 수 있다. 윈도우 메모리 재배치는 연산 비용과 메모리 소요량의 이해득실을 바탕으로 연산자 네트워크 상에서 임의로 이루어질 수 있으므로, 재배치를 위한 경우의 수가 매우 많을 수 있다. 그러므로 윈도우 메모리 재배치를 효과적으로 실행할 수 있도록 하는 전략이 요구되며, 가장 효과적인 윈도우 메모리 재배치 경우부터 점진적으로 재배치를 실행해나가는 탐욕적 방법을 적용할 수 있다.

윈도우 메모리를 재배치할 경우 얻는 메모리 소비량의 감소와 연산 증가량의 비율을 α 라 하면,(즉 α = 메모리감소량/연산증가량), α의 감소 순으로 윈도우 메모리 재배치를 실행하며, 전체 메모리 소요량이 목표 수준 이하일 때까지 반복하는 전략을 적용할 수 있다.

V. 결론

본 논문에서는 데이터 스트림 환경에서 메모리와 연산의 비용간 이해득실 관계를 윈도우 메모리 재배치의 측면에서 식별하고 분석하였다. 이를 위해 윈도우 메모리 재배치를 위한 기본적인 연산자 네트워크 구성 요소를 식별하고, 이 구성 요소를 대상으로 윈도우 메모리 재배치 비용을 분석하는 비용 모델을 수립하였다. 분석 결과 비용의 증가를 감당할 수 있는 환경에서는 윈도우 메모리의 재배치를 통해 메모리 소비량의 감소를 기할 수 있음을 확인하였다.

이러한 윈도우 메모리 재배치에 대한 이해득실의 체계적인 분석을 통해 데이터 스트림 응용을 위한 질의 실행 계획의 개선을 위한 접근 영역을 확장하였으며, 또한 윈도우 메모리 재배치를 활용한 질의 최적화를 위한 비용 산정 모델의 개발 토대를 제공하였다.

본 논문에서는 비용 분석의 편의를 위해 데이터 스트림 투플의 크기 및 윈도우 메모리의 크기가 고정된 것으로 가정하였으나 현실적으로 이들의 크기는 가변적인 경우가 많다. 또한 비용 요소로서 메모리 소요량과 연산의 수행 비용만을 고려하였으나 윈도우 메모리 재배치를 통한 연산 수행 시간의 지연 역시 고려해야 할 비용 요인 중의 하나이다. 앞으로 보다 다양하고 가변적인 환경을 고려한 비용 모델의 확장이 요구되며, 또한 실험을 통한 정량적인 성능 분석의 시도가 필요할 것이다.

참 고 문 헌

[1] C. Aggarwal, *Data Streams: Models and Algorithms (Advances in Database Systems)*, 2006.

[2] N. Chaudhry, K. Shaw, and M. Abdelguerfi, *Stream Data Management (Advances in Database Systems)*, 2005.

[3] J. Kang, J. F. Naughton, and S. D. Viglas: "Evaluating Window Joins over Unbounded Streams," In Proc. of the International Conference on Data Engineering (ICDE), Bangalore, India, 2003(3).

[4] B. Babcock, S. Badu, M. Datar, R. V. Motwani, and J. Widom, "Models and Issues in Data Stream Systems," In Proc. of the 2002 ACM Symp. on Principles of Database Systems (PODS), 2002(6).

[5] 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 the 2003 Conference on Innovative Data Systems Research (CIDR), 2003(1).

[6] A. Arasu, B. Babcock. S. Babu, M. Datar, K. Ito, I. Nishizawa, J. Rosenstein, and J. Widom. "STREAM: The Stanford Stream Data Manager," In Proc. of the ACM Intl Conf. on Management of Data (SIGMOD 2003), 2003(6).

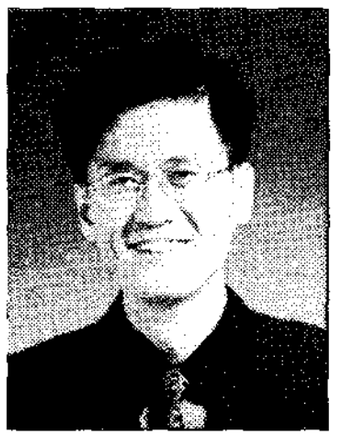
[7] D. Abadi, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. "Aurora: A New Model and Architecture for Data Stream Management," the VLDB Journal, Vol.12, No.2, 2003(8).

[8] D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul and S. Zdonik. "Monitoring Streams: A New Class of Data Management Applications," In Proc. of the 28th International Conference on Very Large Data Bases (VLDB'02), August 20-23, Hong Kong, China.

[9] S. Chandrasekaran, O. Cooper, A. Deshpande, M.Franklin, J.Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M. Shah, "TelegraphCQ: Continuous Dataflow Processing for an Uncertain World," In Proc. of the 2003 Conference on Innovative Data Systems Research (CIDR), 2003(1).

저 자 소 개

이 상 돈(Sangdon Lee) 정회원



- 1984년 2월 : 서울대학교 전자계산기공학과(공학사)
- 1986년 2월 : 서울대학교 전자계산기공학과(공학석사)
- 1991년 2월 : 서울대학교 컴퓨터공학과(공학박사)
- 1987년 3월 ~ 1997년 8월 : 한국통신 연구개발원 선임연구원
- 2001년 1월 ~ 2002년 8월 : 미국 Brown대학교 객원교수
- 1997년 9월 ~ 현재 : 국립목포대학교 정보공학부 부교수

<관심분야> : 멀티미디어 정보관리, 멀티미디어 응용