협력시스템에서의 공유객체의 일관성 유지 방법에 관한 연구.

원보규[0] 정병수
경 희 대 학 교 전 자 계 산 공 학 과
wonbk@jupiter.kyunghee.ac.kr jeong@nms.kyunghee.ac.kr

A Study for Consistency maintenance of the Shared data in Collaborative System

Bo-Kyue Won[0] Byung-Soo Jong
Dept. of Computer Science, Kyunghee University

요 약

협력시스템(collaborative System or Computer-Supported Cooperative Work)은 여러 사용자가 공동 작업을 하기 위해서 참여자간의 정보의 공유와 공조, 의사소통을 지원하는 컴퓨터 기술을 말한다. 오늘날의 CSCW는 기업체내에서의 기업업무 처리 측면에서 고려되는 시스템과 산업 디자인, 협력 편집기와 같이 특정 목적 시스템의 분야로 발전 되어지는 경향이 있다. 산업 디자인과, 협력 편집기와 같은 시스템에서는 다수의 사용자가 동시에 하나의 공유객체를 제어하고, 처리하는 환경을 제공한다. 이러한 시스템은 세분화된 데이터의 공유방법과 제어 방법이 필요하다. 다수의 사용자에 의해서 공유객체에 대한 동시적 조작이 가할 때 공유객체는 일관적인 상태로 존재 시키기 위해서 동시성 제어와 같은 일관성 유지 정책이 요구되어 진다. 본 논문에서는 협력시스템에서의 공유객체에 관한 일관성 유지 정책을 소개하고 기존 정책의 문제점과 개선된 방법을 소개한다

1. 서론

협력 시스템의 목적은 여러 참여자들이 작업에 참여함에 있어서 혼돈하지 않고, 하나의 결과를 생성하게끔 하는 데에 있다. 이러한 협력시스템의 특성은 공간 또는 정보(객체)를 공유하는 환경에서 서로간의 작업이 이루어진다는 것이다. 따라서 다수의 참여자들이 공유객체에 발생시키는 Operation들에 대해서 제어 정책이 필요로 하고, 참여자들이 공유하는 객체는 일관된 상태를 유지하여야 한다. 산업 디자인과 그룹 저작도구와 같은 협력시스템에서는 참여자들이 공유하고 있는 객체에 대한 일관성 유지 정책은 시스템의 가장 중요한 부분 중에 하나이다. 일관성 유지 기법은 다음과 같은 요구 조건을 충족 시켜야 한다. 빠른 응답시간 보장하고 참여자들의 혼란을 최소화하고 참여자들의 의도가 충분히 반영되어야 한다.[1]

지금까지의 여러 협력시스템에서는 이러한 요구조건을 충족 시키는 일관성 유지 정책을 제공한다. 멀티 뷰(view)를 이용한 협력 시스템은 다른 참여자들의 작업 상태를 멀티 뷰(view)를 통해서 인지하기 때문에 어느 정도의 일관성을 유지 한다.[2] 실시간 처리를 요구하는 협력시스템에서는 이러한 view 를 이용한 방법으로는 한계가 있고, 많은 overhead를 발생시킬 수 있다. 따라서 보다 시스템적 내부 측면에서 많은 일관성 유지 방법이 연구되고 있다.

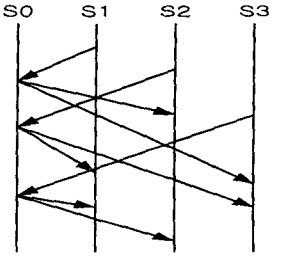
Lock과 Ordering 을 이용하는 방법이 가장 대표적이다. Lock은 공유객체의 작업 권한을 한 참여자에게만 줌으로써 일관성을 유지 하는 방법이며, 다중단위 Locking 과 같은 방법이 대표적인 예이다.[3] 허나 이러한 Locking 방법은 상호작용의 정도가 높은 협력시스템에서는 여러 참여자가 대기 되어야 하기 때문에 효율적이지 못하다. Ordering 방법은 여러 참여자가 대기 상태로 있게 되는 것을 피할 수 있는 측면에서 효율적이어서 많은 연구가 진행되고 있으며 State Vector 이용한 operation 들간의 관계를 검사하여 해당 Operation의 변형을 통해서 일관성을 유지 하는 방법이 활발하게 연구가 되고 있다.[4][5][6] 본 논문에서는 State vector를 이용한 ordering 정책을 중심적으로 소개하고 개선된 연구방향을 제시한다.

또한 협력시스템의 환경은 객체의 공유형태가 완전히 여러 사이트에 분산되어 협력작업이 이루어지는 환경[5]과 중앙이 서버가 공유객체에 대해서 Commit 단계에 저장시키는 환경(Star Like Schema)[6]으로 분류된다. 본 논문에서는 Star Like 환경기반으로 한 산업디자인 협력시스템, 그룹 저작도구와 같은 시스템에서의 일관성유지 정책측면을 고려 할 것이다. 본 논문의 2 장에서는 기존의 Star like 스키마 환경에서 고려된 관련연구와 State vector를 이용한 Ordering 정책을 소개하고 3 장에서는 기존 State vector 를 이용한 방법에 대한 문제점과 개선방향에 대해서 소개가 된다.

2. 관련 연구

Ordering을 고려한 Operation 변형 정책

[그림1]에서와 같이 Star-like 환경에서의 협력 시스템의 Operation 전달 과정에서는 참여자가 발생시킨 Operation은 중앙의 전달자 S0(Notifiler)로 전달되고 ,중앙의 전달자는 다른 모든 참여 사이트에 Operation을 전달하게 된다. 이때 다른 참여자 사이트에게서 Operation을 받았을 때 본인 사이트에서 실행되었던 Operation과의 Ordering과 Operation-relation을 검증한 후 동시성 여부를 판가름하게 되어 실행 여부를 결정한다.[6]



[그림 1] Star-like 환경에서의 Operation 들의 전달 과정

동시성 여부가 존재하면 Operation을 변형 작업을 거친 후 수행되게 되고 동시성이 없는 Operation이면 즉시 수행한다. 협력시스템에서는 Operation들간의 관계 정의를 다음과 같이 할 수 있다.[6][7]

Oa는 site i 에서 발생된 Operation, Ob 는 site j 에서 발생된 Operation이라고 가정

[정의1] Dependent , Independent

Ob는 Oa 에 dependent 하다 (Oa -> Ob)

iff Oa->Ob

Ob와 Oa 는 Independent 한다.(Oa || Ob)

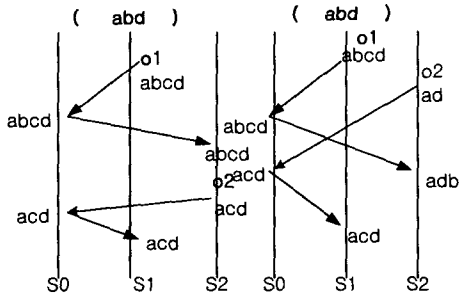
iff Oa -> Ob neither nor Ob -> Oa

[정의2] Causal Ordering

iff i = j, Oa' generation > Ob

iff i != j Oa' execution at site j > Ob'generation

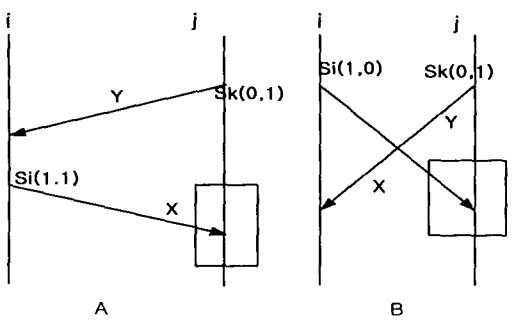
iff there i s exists an operation Ox, such that Oa ->Ox, Ox->Ob



A) Not Concurrent case B) Concurrent case

[그림 2] operation 의 관계

그림 2 에서 O1 은 insert(2,c) 라는 두 번째 포인터에 ' c' 를 삽입하는 operation 이고 O2 는 del(1) 라는 첫 번째 포인터에 문자를 삭제한다는 operation 이라고 가정한다면 A 의 경우를 볼 때 O1 과 O2 는 정의1, 정의2 모두를 만족하므로 O1과 O2 는 동시성이 없는 경우이다. 그러므로 모든 사이트에 최종 상태가 일관성이 유지됨을 볼 수 있다. 그러나 B 의 경우에서는 S0 가 O2 를 받았을 때 검증단계에서 이전에 실행됐던 O1 과 O2 를 비교하면 서로 동시성이 존재(independent 관계)하므로 모든 사이트의 공유객체의 최종 상태는 일관성을 보장하지 않는다. 이때 두 Operation 들간에 관계를 검사하는 데 있어서 State Vector 를 이용할 수 있다. State Vector(Lo gical Time stamp) 는 자신의 Site에서 수행된 Operation 들의 개수를 의미한다. 만약 두 개의 site a, b 가 있다고 가정하면 다음과 같을 수 있다. Site i의 State vector(i,j) : i 는 자기 자신에 의해서 발생된 operation이 실행된 개수를 의미하며 j 는 Site j 에 의해서 전달된 Operation 이 Site i 에서 실행된 operation 의 개수를 의미한다. 분산환경에서 각 Site 는 이러한 Sta te vector 를 가짐으로써 각기 다른 Site 의 상태를 알 수 있는 것이다



[그림 3] State vector 를 이용한 Operation 관계 검사

[그림 3] 에은 같이 Sate Vector 를 이용해서 Operation의 관계를 검사하는 것을 보여주고 있고 세부 관계식은 다음과 같다.[6]

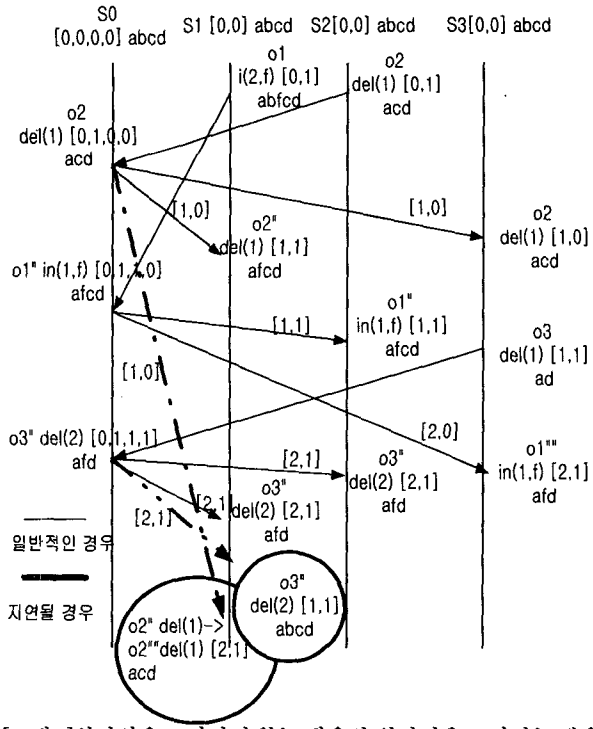
- A : Sk[i] > Si[i] ; Y -> X (dependent, not concurrent)
- B : Sk[i] < Si[i] and Sk[k] > Si[k] ;
  X|| Y(independent, concurrent)

Star like 환경에는 각 State vector 의 요소 값은 참고논문[6]에서와 같은 방법으로 결정되어 Operation과 함께 중앙의 Notifer 또는 각 사이트에 전달된다.

[그림4]은 그룹 에디터 시스템에서 각 사이트에 State vector 와 operation 들의 전달과정을 보여준다. [그림4]에서 o1 이 S0 로 전달 되어질 때 S0 에서는 이전에 실행했던 Operation o2 와 o1 의 관계를 비교 하게 된다. "Star-like환경에서의 State vector 의 비교"에 의해서 o2 와 o1 은 independent 관계에 있기 때문에 o1 의 원래의 Operation insert(2,f) 는 operation 변형 알고리즘에 의해 insert(1,f) 로 변형되어 실행된다. 이러한 과정을 거쳐서 각 사이트에서의 공유객체의 최종 상태는 일관성을 보장하게 된다.

3. State vector 를 이용한 Operation 관계 검사 방안의 문제점과 개선 방안

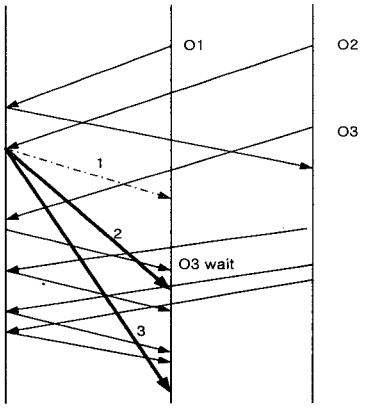
2장에서 소개된 State vector 를 이용한 Operation 의 관계 검사 방법을 소개하였다. 이 방법은 그룹 편집기와 산업 디자인협력시스템에서 작업에 참여하는 모든 사이트의 Operation 들의 관계를 검사하여 각 사이트에 공유된 객체의 최종상태가 모든 사이트에서 일관성을 만족 하게 한다.



[그림4]일관성을 보장하지 않는 경우와 일관성을 보장하는 경우

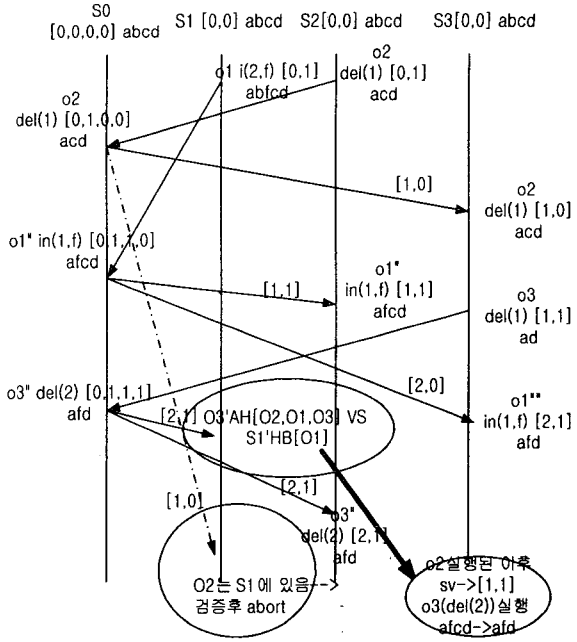
분산환경에서의 협력시스템은 무엇보다도 작업에 참여하는 참여자들이 발생시킨 Operation 들이 다른 각 참여자에게도 적용되어야 하며, 오류를 유발하는 상황에서도 그 해결책을 제공해야 한다. 즉, 시스템의 오류나 네트워크 지연으로 인해서 참여자들이 발생시킨 Operation 은 제때에 서로 전달되지 않고 다소 지연될 수 있는 문제가 발생된다는 것이다. 그렇게 되면 협력시스템에 참여하는 참여자들의 operation 들은 서로 뒤엉켜 잘못된 결과를 초래하며 결과적으로는 모든 사이트에서의 공유객체는 일관성을

보장하지 않게 된다. 그림 4 에서 보면 operation o2 가 사이트 s1 에 도착이 o3 의 도착이전에 도착되지 않고 이후에 도착한다면 각 사이트의 공유객체에 대한 최종 상태는 일관적이지 못하다. 사이트 s1 에서 o3 가 도착하면 o3 는 o1 과의 관계가 dependent . 한 관계가 되기 때문에 o3 는 변형 없이 즉시 실행되어 지고 이후에 도착된 o1 은 잘못되어 실행된 o3 와 관계를 검사하기 때문에 최종 결과는 일관성을 보장하지 않게 된다. 이러한 문제의 해결은 o3 는 o1 이 도착할 때 까지 기다림으로써 해결될 수 있다.



[그림 5] Operation 이 대기하는 경우

이것 또한 [그림 5] 에서 보듯이 문제점을 갖고 있다. 정상적으로는 o1 의 도착은 '1'과 같은 경우여야 한다. 그러나 문제가 발생되어 o2 의 도착이 o3 이후에 도착하게 된다면 o3 는 o2 가 도착할 때까지 대기하여야 한다. 이때 '2' 의 경우처럼 o2 의 도착이 o3 대기 한 후 바로 도착한다면 o3 의 대기 시간은 짧아 지게 된다. 그러나 '3' 의 경우처럼 o2 의 도착이 무작정 길어 진다면 o3 가 대기한 시간 이후에 도착하는 operation 모두는 대기 할 수 밖에 없고 그 만큼에 시스템의 신뢰도는 떨어 진다. 본 논문에서는 이러한 상황을 해결하기 위해서 Operation 들의 조상 히스토리 버퍼를 이용한 낙관적 실행 방법을 제시한다. 사이트 뿐만 아니라 Operation 들도 그 들이 생성되기까지의 자신들의 조상 히스토리를 가져야 한다. 사이트들은 Operation 전달 시 Operation 뿐만 아니라 그 Operation 의 조상 히스토리를 함께 보냄으로써 Operation 들의 관계를 검사하는 데 보다 낳은 효과를 볼 수 있다. O1 ->o2->o3->.... Oi 이면 oi 의 AH(Ancestor History)= [o1, o2, o3,o4 ....oi] 이다. 또한 각 사이트도 자신의 사이트에서 실행된 모든 operation 에 대한 history buffer 를 가지게 된다. 어떤 O peration 이 사이트에 도착하게 되면 자신의 사이트의 history buffer(HB) 와 operation 의 AH 를 비교 함으로써 아직 도착되지 않은 Operation 을 예측 할 수 있다. 또한 아직 도착되지 않은 operation 이 발견되면 operatio n 은 대기 되지 않고 우선적으로 실행된다. 그런 후에 미 도착 operation 이 도착하면 대기 되었어야 할 operation 의 실행 결과가 바른지를 검증하고 올바르다고 판별되면 미 도착 operation 을 abort 시키는 낙관적 수행 방법을 씀으로써 operation 이 무한 적 대기 함으로써 발생되는 문제점을 해결 할 수 있다.[그림 6] 은 [그림 4] 의 경우에 문제점을 해결하고 공유객체에 대한 최종 상태가 일관성을 보장하는 경우를 나타낸다. o3 의 A H 와 사이트 S1 의 HB 를 비교한 후 O2 가 아직 도착하지 않았다는 것을 알 수 있다. Operation o3 의 AH 를 탐색한 결과 아직 도착되지 않은 O2 가 어떤 Operation 인지를 알기 때문에 우선적으로 O2 를 실행시키고 그 후에 O2 가 도착되었을 경우 전에 실행



[그림 6] 일관성을 보장하는 경우

된 O2 의 결과가 바르다면 도착된 O2 는 실행시키지 않아도 된다.

4. 결론

협력시스템에서는 참여하는 모든 참여자들의 Operation 들이 서로 상호작용을 이루며 어떠한 작업을 행하게 된다. 이 때 참여자들이 발생된 operation 은 서로 상호작용을 통해서 공유객체에 최종적인 일관성이 보장하게 하여야 하는데 State Vector 를 이용한 operation 관계 검사 방법은 일관성을 보장하지 하지 않는다는 것을 보였다. 우리는 그 문제 해결 방안으로 Operation 들의 조상 히스토리와 참여 사이트에서 실행된 모든 Operation 들의 히스토리 버퍼를 이용하여 낙관적 수행 방법을 제안함으로써 각 참여자가 공유한 객체의 최종 상태를 보장 시킴을 보였다.

참고 문헌

[1] 김창한, 양재현, " Collaborative Virtual Environment' 정보과학회지 제 16권 7호,1998,7

[2] Ted O'Grady and Saul Greenberg ;" A groupware environment for complete meetings. " ; *Proceedings of the CHI '94 conference companion on Human factors in computing systems* , 1994

[3] Jonathan Munson and Prasun Dewan; " A concurrency control framework for collaborative systems " ;Proceedings of the ACM 1996 conference on on Computer supported cooperative work

[4] MaherSuleiman,Michele Cart and Jean Ferrie; " Serialization of concurrent operations in a distributed collaborative environment" ;Proceedings of the international ACM SIGGROUP conference on Supporting group work: the i ntegration challenge, 1997

[5] C. A. Ellis and S. J. Gibbs; " Concurrency control in groupware systems " ;Proceedings of the 1989 ACM SIGMOD international conference on Management of data, 1989

[6] Chengzheng Sun and Rok Sosic;"Consistency Maintenance in Web-Based Real-Time Group Editors"Proceedings of the 1999 ICDCS Workshop on Electronic Commerce and Web-Based Applications

[7] Chenegzheg Sun and Xiaohua jia ; " Archieving Convergence, Causality Preservation, and Intention Preservation in Real -Time Cooperative Edi ting Systems" ;Acm Transactions on Computer-Human Interactive, March 1998