이동컴퓨터 상에서의 공동작업을 위한 자동저장 방식 설계 및 구현*

이 근 영, 김 남 광, 박 승 규
아주대학교 정보 및 컴퓨터공학부

A Hoarding Policy for Collaborative Computing in Mobile Environment : Design and Implementation.

Keun-Young Lee, Nam-Kwang Kim, Seung-Kyu Park
Division of Information and Computer Engineering, Ajou University
Tel: 0331-219-2532, E-mail: sparky@madang.ajou.ac.kr

Abstract

This paper describes the design and implementation of file system which allows the collaborative computing in mobile environment. The design goal is to make a logically one file system in the distributed computer systems. The characteristics of frequent, foreseeable and variable disconnections in a mobile environment were taken into consideration. We introduce an auto-hoarding system that provides the availability of large number of nodes which are weakly and intermittently connected. The data consistency problems in distributed or replicated mobile data are also discussed.

1. 서 론

이동 컴퓨터(mobile computer)의 개발과 유/무선 통신 기술의 발달, 그에 따른 웹 사용의 급속한 확산에 따라 이동중에서의 원거리 작업과 함께 동기적 그룹웨이(synchronous groupware)에 대한 요구가 급속도로 증가되고 있다. 새로운 이동 컴퓨터는 사용자의 편의성에 중점을 두는 현재의 컴퓨터 환경 추세에 발맞추어 빠른 속도로 개발되고 있다.

아침에 잠에서 깨워주고 하루의 스케줄을 알려주고 메모를 정리해주며 인터넷의 정보를 얻어서 보고서를 작성하고 지하철에서 음악을 들으며 게임을 하면서 퇴근을 하는 생활의 중심에서 필수품으로 자리잡을 것이다. 또한 무선 통신 기술의 발달과 접목하여 거리에서 상대방 얼굴을 보며 통화할 날이 멀지 않았다. 하지만, 아직까지 제한된 시스템 및 빈약한 응용 프로그램등 선결되어야할 많은 문제들이 남아 있다.

본 논문에서는 이전에 이동 컴퓨터 환경을 위해 제안되었던 이동 분산 파일 시스템[1]을 근간으로 협동작업에 보다 적응적인 시스템 모델을 설계한다. 최근 collaborative computing을 통한 협동작업을 지원하는 그룹웨어에 대한 관심이 높아지고 있지만, 아직까지 구체적인 시스템이 체계적으로 정립되었기보다는 한창 개발이 진행단계에 있다고 볼 수 있다.

특히 이것은 이동 컴퓨터를 이용하여 보다 손쉽게 작업을 처리할 수 있음에도 불구하고 뒷받침할 수 있는 시스템의 부재로 그 활용이 빈약했었다.

2. 전체적인 구성도

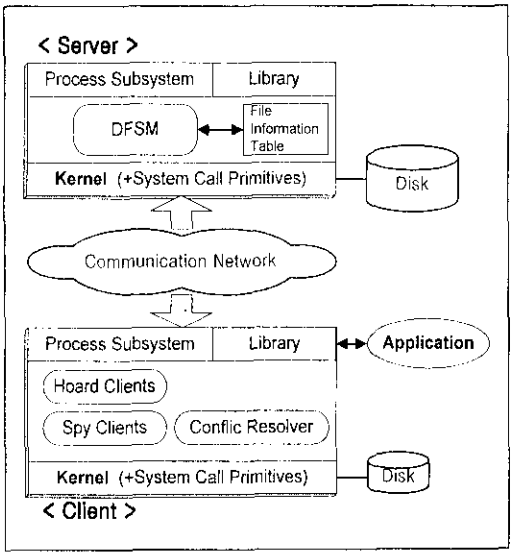
<그림 1>과 같은 형태를 갖는 본 분산 파일 시스템 모델의 서버측은 Windows NT를 기반으로 하였으며, 클라이언트측은 Windows 95/CE를 기반으로 설계를 적용시켰다. 이와 같은 적용은 최근 많은 사용자 및 개발자가 이용하는 시스템을 채택하고, GUI를 기반으로 한 파일 시스템 관리자를 제공함으로써 보다 손쉽게 접근할 수 있도록 하였다.

2.1 서버 모듈

다수의 클라이언트에서 요청되고 갱신되는 파일들에 대한 관리를 담당하게 된다. DFSM(분산 파일 시스템 관리자)는 본 분산 시스템에서의 사용자를 위해 파일을 등록시키며, 등록된 파일을 관리하는 역할을 담당

* 본 연구는 한국과학재단 목정 기초연구비(961-0100-001-2) 지원으로 수행되었으며 지원에 감사드립니다.

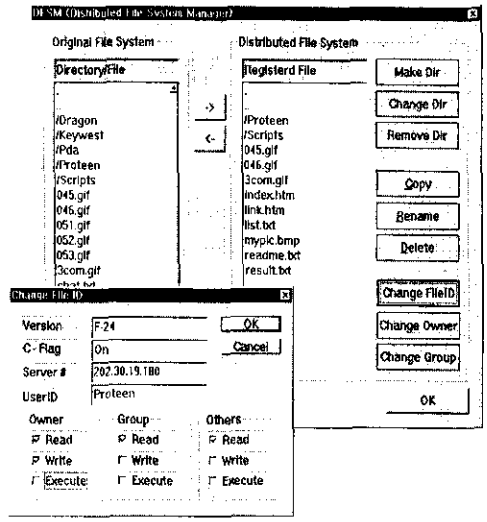
하는 관리자로써 별도의 'File Information Table'을 두어 파일의 사용 내역에 대한 자세한 정보들을 저장하고, 이것을 자동저장(hoarding)시 활용하게 된다. 특히 분산되고 복제될 파일에 대한 접근 제어와 데이터 재공 등에 대한 역할을 담당하는데, 이것은 기존 시스템에서 제공되는 명령어에 추가적인 기능을 첨가하여 구현하고 있다.



<그림 1> 전체 시스템 구성도

본 구조에서는 기존 시스템에서 볼 수 있는 파일의 소유권외에 부가적으로 파일 ID를 부여하는데, 이것 또한 'File Information Table'에서 관리하게 된다.

<그림 2>에서 보는 것처럼 파일 ID는 네 가지 필드로 구성된다. 버전(Version)은 캐쉬된 파일의 버전과 서버의 원 파일간의 버전을 확인함으로써 일관성을 유지시키도록 한다. 일관성 유지를 위해서는 C-Flag을 두고, 이에 따라 낙관적 일관성(optimistic consistency)유지 정책을 사용할 것인가 엄격한 일관성(strictly consistency)유지 정책을 사용할 것인가에 대한 결정을 내리게 된다. 다음의 서버#는 중복 복제된 파일에서 원시 파일을 가진 서버를 찾기 위한 것으로 네트워크상의 IP address를 이용한다. 마지막의 User ID를 추가시킴으로써 각 파일의 사용권에 대한 제한적 사용을 가능하게 하고, 일관성 정책 수행시 우선 순위를 부여한다. 또한 내부적으로 디렉토리 정보 베이스(DIB)를 구축하여, 디렉토리 정보 트리(DIT)와 함께 계층적 이름 공간(hierarchical name space)을 설정하게 되며, 이것을 통해 새로운 파일서버의 디렉토리 구조를 통합하는 역할을 수행하고, 'File Information Table'과 연결되어 보다 효과적인 파일을 관리할 수 있게 한다.



<그림 2> 분산 파일 관리자에 대한 실행 화면

2.2 클라이언트

기본 시스템 콜을 지원하는 커널과 함께 크게 세 가지 부분으로 구성된다. 자동저장과 그에 따라 중복복제된 파일들을 일관성있게 관리하는 'hoard client'와 자동저장된 데이터가 얼마나 효율적으로 캐쉬되고 사용되는지에 대한 정보를 추적하는 'spy client', 그리고 재접속시 수정된 파일들을 서버로 갱신할 때 발생하는 충돌(conflict)을 해결하는 'conflict resolver'등이 있다.

일관성에 관한 문제는 앞선 논문에서 제안된 방법과 같이, 낙관적 일관성 수행시에는 'conflict resolver'가 함께 수행되어 발생하는 충돌을 제거하도록 하였으며, 엄격한 일관성을 요구할 경우에는 사용자 권한에 의존하여 부가적인 토큰을 통하여 제한적 파일 사용 허가를 부여하는 방법을 사용하였다.[1] 이중 낙관적 일관성 유지에 사용되는 'conflict resolver'는 선택적으로 두 가지 방법을 채택하였다. 첫번째는 확장자를 이용한 방법으로 파일간 버전을 구분시키고 차후에 해당 파일의 소유자로 하여금 수동적으로 파일을 통합하도록 하는 방식이고, 두번째는 작업중 파일을 직접 수정하는 것이 아니라 작업 절차를 타임 스탬프와 함께 log형태로 저장하였다가 차후 서버에 갱신할 때 통합적인 log에 의해 갱신이 가능하도록 하는 방식을 사용하였다.

자동저장 방법에 관해서는 다음 절에서 보다 구체적으로 설명되고 있다.

3. 파일의 자동저장(Hoarding) 모듈

이동 환경에서의 원활한 작업을 위해서 가장 효율적인 방법중 하나로 자동저장(Hoarding)의 방법을 많이 사용하고 있다. 중복 복제(replication)의 한 형태를 유지하면서 자료를 캐쉬하고 단절된 상태에서의 작업을 수행시킨다. 이때 발생하는 가장 큰 문제점으로는 캐쉬된 데이터에 대한 캐쉬 미스를 최소화하는 것과 어떻게 일관성을 유지시킬 것인가에 있다.

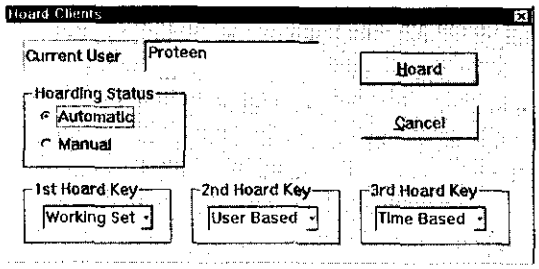
3.1 기존의 자동 저장 방법

CODA[2], AFS(andrew file system)[3], Ficus[4]등 기존의 시스템에서는 캐쉬를 이용하여 단절 상황에서의 작업 수행을 통하여 보다 효율적인 활용 방안들을 제시하기도 하였다. 특히 네트워크가 급속도로 발달되고 이에 따른 응용 프로그램들이 많아진 현재의 상황에 반하여, 빠르지 못하고 낮은 수준의 통신을 사용하는 이동 컴퓨터 환경에서 단절 상태의 작업 수행은 더욱 절실한 요구 사항이 되고 있다. 원격 파일에 대한 접근 제한이나 연결상태의 단절은 특정 작업을 중단케 만들 것이며, 심지어는 컴퓨터 자체를 무용지물로 만들게 될 것이다.[1]

단절 상태의 작업 수행에 대한 유용한 방법중의 하나는 저장(hoarding)과정을 들 수 있다. AFS에서는 단절된 사용자가 단절을 대비하여 응용(Application)에 적절한 파일을 지역 디스크에 캐쉬하는 방법을 취한다. 또한 AFS의 방법을 계승한 CODA에서는 사용자가 hoard profile내에 자동저장될 파일과 디렉토리를 직접 지정해 주는 방법을 취하지만 이것도 부하가 심하고 신뢰도가 높지 못하다는 단점을 갖는다. Kenning의 Seer 시스템에서는 의미론적인 거리(Semantic distance)라는 개념을 이용하여 파일간의 상관관계를 통해 예측적 캐싱을 하는 정책을 사용한다.[5] 이 시스템은 사용자의 유형과 파일 억세스 형태를 관찰하고 그에 따른 정보를 수집하는 observer와 참조된 파일 사이의 거리를 구하는 correlator 두 가지 요소로 분리된다. 마지막으로 transparent analytical spying[6]이라고 불리는 방법으로 사용자의 파일 억세스 패턴을 관찰하여 작업 그룹을 만들어 사용하는 것과, 사용자 데이터와 응용 프로그램에 대한 데이터를 확장자나 디렉토리 이름으로 구분짓는 것, 그리고 타임 스탬프를 사용하는 세 가지의 방법을 사용하고 있다.

3.2 제안된 자동저장 모듈

다음 <그림 3>은 구현중에 있는 자동저장 프로세서(hoard client)의 초기 실행 화면이다.



<그림 3> hoard client의 초기 시작 화면

본 시스템의 가장 큰 특징으로는 단절중 작업시 캐쉬된 데이터에 대한 통계적 자료가 'spy client'에 의해서 기록되고 보관된다는 점을 들 수 있다. 즉 각각의 사용자 특성 또는 여러 가지 작업을 수행하는 한 사용자에 대한 작업 유형에 따라 저장키(hoard key)를 분류하고, 어떠한 저장키를 사용하는 것이 유리한 것인가를 판단할 수 있는 것이다. 이 자료는 사용자가 바뀜에 따라 적절한 저장키를 'Hoard Client'가 자동 설정하여 단절전 자동저장 작업을 수행하게 된다.

<표 1>은 제안된 시스템에서의 주요 저장키에 대한 리스트이다.

<표 1> 주요 저장키 리스트

Main Key	분류 기준
Working Set	사용자의 파일 억세스 형태에 따른 log 기록 및 tree형성
Content Based	파일 성격에 따른 보조키 설정과 그에 따른 파일 분류
Extension	확장자에 따른 파일 분류
User Based	파일 소유권(owner, group, others)을 통한 분류
Time Based	최근 또는 특정 기간을 설정하여 분류
Application	각 응용 프로그램에 필요한 파일들의 그룹 형성

표에서 보듯이, 주요 키중에서 'Working Set'키는 전적으로 'spy client'에 의해 생성된 트리의 패턴에 의존한다. 따라서 이 트리의 패턴 정보가 완전히 획득되기 전까지는 'Content Based'키나 'Application'키에 의존하여 자동저장된 파일들과, 이때 캐쉬 미스되어 요구된 파일 정보를 통해 작업을 수행한다. 이렇게 생성된 트리의 패턴은 서버의 지역 디스크에 기록되어 차후 발생하는 자동저장 과정에서 유용한 정보로 이용

된다. 반면에 'Content Based'키는 파일의 성격이나 내용에 따라서 파일 생성시 사용자에 의해 정의된 몇 가지 보조키에 따라 자동저장 여부를 결정짓게 된다. 예를 들어, 전기 점검을 담당하는 회사의 DB정보를 생각해보자. 이 DB에는 각각의 가구에 대한 정보(현 상태, 위치, 최근점검일 등)를 담은 파일들이 있을 것이고, 수원 지역에 대한 점검을 담당하는 사람은 바로 'Suwon'이라는 키를 갖는 파일만 자동저장(hoarding)하여 작업에 나서게 된다. 또한 최근 한달 이내에 점검된 가구는 작업을 하지 않는다면, 그에 해당되는 파일은 자동저장될 필요가 없다. 지상키에 대한 개별적인 관리와 함께, 'spy client'는 해당 사용자가 사용하는 저장키에 따른 캐쉬 미스 확률도 별도로 계산하여 사용자에게 최적화된 Working Set Tree를 제공하기도 하며, 보다 적응적인 저장키를 선택할 수 있는 정보를 제공하게 된다. 그리고 단절 중에 관찰되었던 정보들은 서버와 접속시 서버의 'File Information Table'에 기록이 되어 다음 번 자동저장을 위한 정보로 활용된다.

4. 결론 및 향후연구방향

본 연구는 이동 컴퓨터에서의 단절된 동안 작업 수행 및 공동 작업 수행을 위한 자동저장방법과 일관성 정책을 고려한 시스템을 설계하고, 간단한 구현을 통해 본 시스템을 테스트한 것이었다. 이 시스템을 통해 이동 환경을 위한 기존의 분산 시스템과의 접목은 물론, Collaborative Computing을 위한 응용 프로그램을 개발하는데 손쉬운 환경을 제공하게 되었다. 향후 제안된 방법의 효과와 성능 향상의 가장 중요한 요소인 지능적인 자동저장(Intelligent File Hoarding)에 대한 완벽한 구현과 함께 이를 뒷받침할 수 있는 시스템 콜 인터페이스에 대한 개선이 첫번째 선결과제로 대두되었다. 또한 여전히 발생하는 낙관적 일관성 유지와 엄격한 일관성 유지 사이의 tradeoff 문제를 보다 적절히 수용하여 일관성 유지를 위해 발생하는 시스템 낭비 및 네트워크 부하를 최소화할 수 있는 방안을 모색 중에 있다.

참 고 문 헌

[1] 이근영, 남동훈, 박승규, "이동컴퓨팅 환경에서 캐쉬 정책과 일관성 유지방법", 정보과학과 98년 춘계 학술발표회, 아주대학교, 1998

[2] J. J. Kistler and M. Satyanarayanan, "Disconnected Operation in the Coda File System", ACM Trans. on Computer Systems, Vol. 10, No. 1, Feb. 1992.

[3] John H. Howard, "An overview of the Andrew File System", pp23-36 in Proc. of the winter USENIX Conference, Dallas, Feb. 1988

[4] R. G. Guy, J. S. Heidemann, W. Mak, T. W. Page Jr., G. J. Popek, and D. Rothmeier, "Implementation of the Ficus Replicated File System", In Proc. 1990 USENIX Summer Conf., pages 63-71, June 1990.

[5] G. H. Kuenning, "The Design of the Seer Predictive Caching System", In IEEE Workshop on Mobile Computing Systems and Applications, Dec. 1994.

[6] Carl Tait, Hui Lei, Swarup Acharya, Henry Chang, "Intelligent File Hoarding for Mobile Computers", ACM conference on Mobile Computing and Networking(Mobicom '95), Berkeley, CA, Nov. 1995