□ 기술해설 □

이동 컴퓨팅 환경의 분산 화일 시스템

성균관대학교 박재원·김문정·엄영익*

1. 서 론

분산 화일 시스템이란 실제로는 여러 호스트에 개별적으로 구성되어 있는 각 화일 시스템들을 통합하여 궁극적으로 사용자들에게는 하나의 화일 시스템으로 보일 수 있도록 구성된 화일 시스템을 말하며, 이와 같은 환경에서는 사용자들이 원격 화일(remote file)에 접근하고자 할 때 자신이 사용하고 있는 호스트의 지역 화일(local file)에 접근하는 것과 같은 방법으로 접근할 수 있도록 지원한다[1]. 이를 위해 분산 화일 시스템은 위치 투명성(location transparency)과 접근 투명성(access transparency) 등을 제공하여야 한다.

기존의 분산 시스템이 유선망(wired network)을 기반으로 구성되어 왔으나, 최근에는 무선통신 기술의 발달과 휴대용 컴퓨터의 보편화에 따라 이동하는 호스트들에게도 네트워크 연결 및 서비스 제공을 지속시킬 수 있는 환경에 대한 연구가 이루어지고 있다. 이러한 환경을 이동 컴퓨팅 환경(mobile computing environment)이라 하며(그림 1) 이는 호스트의 이동성(mobility)을 추가한 분산 컴퓨팅 환경의 확장된 개념으로 볼 수 있다[2].

이와 같은 이동 컴퓨팅 환경은 고정 호스트(fixed host)들 및 유선망으로 구성되는 고정 네트워크(fixed network)와 위치를 이동하는 중에 무선통신 매체에 의하여 고정 네트워크와 통신하는 이동 호스트(mobile host), 그리고 일종의 고정 호스트로서 이동 호스트와의 통신을 담당하는 이동지원국(MSS : Mobile Support Station) 등으로 구성된다.

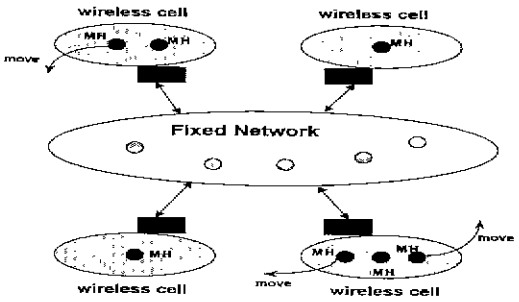

그림 1 이동 컴퓨팅 환경

앞에서 언급하였듯이 이동 컴퓨팅 환경은 분산 컴퓨팅 환경의 확장된 개념으로 볼 수 있으며 분산 컴퓨팅 환경에서의 각종 문제점이나 해결책, 설계 기법 등이 이동 컴퓨팅 환경의 설계 및 구축에 이용될 수는 있으나, 이동 컴퓨팅 환경에서는 무선통신 및 호스트의 휴대성, 그리고 이동성에 따르는 추가의 문제점들이 발생하며 이에 따라 분산 컴퓨팅 환경에서의 설계 기법들을 변경하여야 하는 경우도 많이 발생한다[3].

화일 서비스(file service)를 제공하기 위한 화일 시스템의 구성에서도 이동 컴퓨팅 환경의 특성에 따라 고려해야 할 요소들이 나타나게 되는데 이는 우선적으로 이동 호스트와 고정 네트워크측과의 연결상태가 변화할 수 있음에 따라 발생한다. 이동 호스트의 상태는 고정 네트워크와의 연결 정도에 따라 완전연결(fully connected) 상태와 부분연결(weakly connected) 상태, 그리고 단절(disconnected) 상태 등으로 구분할 수 있다(그림 2).

각 상태들은 이동 호스트와 고정 네트워크측

*종신회원

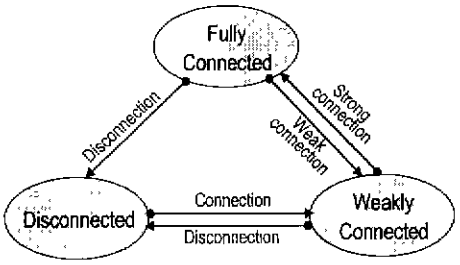

그림 2 이동 호스트의 연결상태

간의 통신 대역폭(bandwidth)이 어느정도 제공되고 있는가에 따르는 구분이다. 일반적으로 이동 호스트가 유선망으로 연결되는 경우나 무선 LAN(wireless LAN) 환경에 있어서 빠르고 신뢰성 있는 통신이 가능한 경우를 완전연결 상태로 구분하며, 보다 저속의 cellular telephony나 CDPD 등의 환경에 있어서 느리고 비신뢰적이거나 통신비용이 높은 경우를 부분연결 상태로 구분하고, 연결이 완전히 끊어져서 서버로부터 어떠한 형태의 서비스도 받을 수 없는 경우를 단절상태로 구분하는데, 이는 경우에 따라 달리 구분될 수도 있다.

현재까지 이동 컴퓨팅 환경을 위하여 화일 시스템 측면에서 고려하고 있는 사항은 이동 클라이언트가 단절상태로 전이될 수 있다는 사실과 이에 대한 준비가 미리 이루어져야 한다는 점이며, 이를 위한 정책 및 기법들이 몇가지 제시되어 있는 정도이다[4]. 이동 클라이언트가 고정 네트워크에 있는 서버측의 화일을 요청하는 환경에서 이를 효과적으로 서비스하기 위해서는 단절상태에 대한 고려뿐만 아니라 현재 이동 호스트의 위치나 각 연결상태에 따라 서버측의 서비스 제공 방법이 달라져야 할 것이다. 이중 특히 캐싱 기법과 관련된 사항들에 대해서는 3절에서 언급한다. 기본적으로 이동 컴퓨팅 환경에서의 화일 서비스와 관련한 각종 기법들은 가능한한 사용자들에게 투명성을 제공한다는 원칙하에서 이루어져야 할 것이다.

본 고에서는 이동 컴퓨팅 환경을 위한 분산 화일 시스템 설계 기술, 특히, 캐싱 기법에 서 고려하여야 할 사항들에 대해 기술한다. 구체적으로 2절에서는 분산 화일 시스템의 구조에 대해 기술하며, 3절에서는 캐싱 기법에서 고려할 사항들을 언급하고, 4절에서는 기존의 대표적인 분산 화일 시스템들에 대한 소개와 이들이 최근 이동 컴퓨팅 환경을 지원하기 위하여 이떠한 형태로 변화되고 있는지에 대한 사례들을 보인다.

2. 분산 화일 시스템 구조

분산 컴퓨팅 환경에서의 화일 서비스를 위해서는 클라이언트측과 서버측에 각각 필요한 모듈들이 구성되어야 한다. 클라이언트측에는 응용 프로그램의 화일 서비스 요청을 받아 원격 화일에 대한 요청일 경우 이를 해당 서버로 전달하고, 필요한 화일을 캐싱하여 관리하는 모듈이 필요하며, 서버측에는 이러한 서비스 요청을 받아 이에 대한 서비스를 지원하는 디렉토리 서비스 모듈과 화일 서비스 모듈등이 필요하다(그림 3).

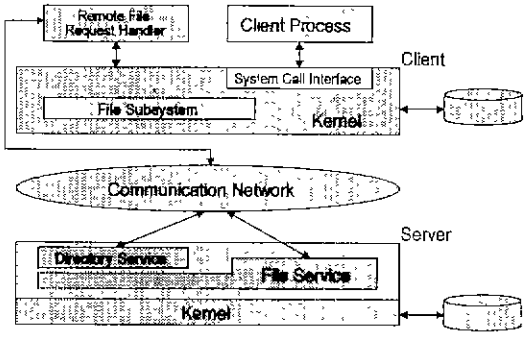

그림 3 화일 서비스 구성요소

각 서버에 존재하는 화일들을 전체적으로 관리하기 위해서는 일반적으로 모든 화일들에 대해 고유번호(UFID : Universal File Identifier)를 부여하게 되며, 특히 화일들의 관리의 용이성을 위해 관련된 화일들을 그룹 단위로 구분하기도 하는데 이 경우에는 각 화일들의 고유번호를 그림 4에서와 같이 구성한다. UFID에는 화일 서비스 인터페이스에 따라 접근 권한 등에 관련된 정보가 추가될 수도 있다[1].

File Group ID	File ID	Other Fieds

그림 4 UFID 구조

3. 캐싱기법 및 설계사항

이동 컴퓨팅 환경에서는 사용자가 이동하는 중에도 자신이 원하는 작업을 계속할 수 있도록 지원해야 한다. 이는 다시 말해서 위치-독립적 작업(location-independent work)을 지원해야 한다는 의미이며, 사용자가 이동하는 도중에도 자신이 원하는 서비스를 지원받을 수 있어야 하고 필요한 자원에의 접근이 가능해야 한다는 의미이다. 또한 이러한 모든 기능은 사용자들에게 추가의 행위를 요구하지 않으면서 기능면이나 성능면에서 투명하게(transparently) 이루어져야 한다.

시스템 환경에 심한 오버헤드를 주지 않으면서 이와 같은 기능을 지원하기 위해서 기본적으로 사용할 수 있는 기법이 클라이언트측에 필요한 정보 또는 화일들을 캐싱(caching)하는 것이다. 분산 환경에서와 달리 이동 컴퓨팅 환경에서는 이동 호스트의 연결상태가 다양하게 나타날 수 있고 1절에서 언급한 상태외에도 구체적으로 간헐적 단절(intermittent disconnection)이 이루어지는 상태, 비신뢰적 상태(unreliable state) 등이 있을 수 있다. 이에 따라 서비스를 제공하는 환경이 달라질 수 있으므로 캐싱과 관련된 각종 세부 기법들이 각 연결 상태에 적응성(adaptability)을 갖도록 변형되어야 한다.

본 절에서는 이동 컴퓨팅 환경의 성능 극대화를 위하여 고려되어야 할 구체적인 화일 캐쉬 관리 기법들, 특히, 캐싱의 단위 설정 문제, 캐쉬 교체 기법, write-back 기법, prefetch 기법, hoarding 기법, adaptation 기법 등에 대해 언급한다.

3.1 캐싱 단위

캐싱 단위는 클라이언트가 서버측의 정보를 보관하는 단위를 의미하는데 보통은 클라이언트와 서버간의 자료 전송 단위를 포함하여 의미한다. 캐싱 단위은 전형적으로 블럭단위(block basis)와 화일단위(file basis)로 구분된다. 전송속도가 매우 빠르고 신뢰성이 높은 통신 환경에서는 블럭단위 캐싱 기법을 사용할 수 있으나, 일반적으로 분산 환경에서는 클라이언트가 서버측과 단절되는 경우를 고려하여 전체 화일을 전송하고(whole file transfer) 전체 화일 단위로 캐싱하는(whole file caching) 화일단위 캐싱 기법을 사용한다.

이동 컴퓨팅 환경과 같이 제한된 대역폭만이 제공되는 부분연결 상태가 가능한 경우에는 전송량과 전송횟수를 줄일수 있는 기법이 필요하게 된다. 따라서 LAN 기반의 기존의 분산 환경에서와는 달리 이동 컴퓨팅 환경에서의 캐싱 단위 설정문제는 보다 복잡해질 수 있다. 현재 이동 컴퓨팅 환경을 위한 캐싱 단위의 설정과 관련해서는 다음과 같은 두 가지 기법이 제시되어 있다.

첫번째로 캐싱 단위를 가변적으로 변화시키는 방법이 있을 수 있다[4]. 이는 네트워크의 접속 상태에 따라 유동적으로 캐싱 단위를 변화시키는 기법으로, 낮은 대역폭 상태에서는 캐싱 단위를 크게 하고, 높은 대역폭 상태에서는 캐싱 단위를 작게 한다. 즉, 완전연결 상태에서는 캐싱 단위를 LAN 환경에서와 같은 수준으로 하고, 단절 상태에서는 화일 전체를 캐싱 단위로 하며, 부분연결 상태에서는 캐싱 단위를 블럭 크기보다는 크고 화일 전체의 크기보다는 작은 중간정도의 크기로 설정하는 것이다. 이러한 기법은 적응성의 면에서는 장점을 가질 수 있으나 시스템 운영의 면에서는 복잡도가 크게 증가하는 단점을 갖는다.

또 다른 방법으로는 캐싱의 단위로 화일의 그룹을 사용하는 Seer 예측 캐싱 기법이 제시되어 있다[5]. 이는 관련성이 있는 여러 화일들을 같은 그룹으로 묶고 이를 캐싱 단위로 사용하는 기법이다. 이 기법은 클라이언트의 접속상태를 고려하지 않고 있으며, 낮은 대역폭 환경에서 비실용적일 수 있다는 단점을 갖는다.

3.2 캐쉬 교체 기법

클라이언트의 캐쉬가 더 이상의 가용영역이 없는 경우 새로운 데이타를 캐싱하기 위해서는 필요한 공간의 확보를 위하여 기존에 캐싱되어 있는 데이타의 일부를 교체해야 한다. 이를 위하여 기존의 LAN 환경에서는 LRU(Least Recently Used) 기반의 기법들을 사용해 왔으며, LRU 기반의 기법들은 시스템 전체의 성능

면에서 적절한 기법으로 알려져 있다[6, 14]. 그러나, 이동 컴퓨팅 환경에서는 캐쉬 교체를 위해 몇가지 요소를 추가로 고려해야 한다.

우선 이동 클라이언트는 접속상태의 변화 가능성으로 인하여 기존의 분산 환경에서보다 화일 캐쉬의 내용에 대한 의존도가 높다는 사실이다. 오랜 기간동안 참조되지 않았던 화일에 대한 접근 가능성을 대비하여 더욱 긴 기간동안의 화일 참조 패턴을 고려하여야 할 것이며 캐쉬의 크기가 크다는 사실도 염두에 두어야 한다. 또한 사용자의 작업 형태가 변화되는 상황, 즉, 사용자가 완전연결 상태에서 참조하던 화일과 부분연결 상태 또는 단절상태에서 참조하는 화일이 다를 수 있음도 고려되어야 하며, 캐쉬 단위의 변화에 따라 교체 기법이 영향을 받을 수 있음도 고려되어야 한다. 이동 클라이언트의 접속상태에 따라 캐쉬 미스(cache miss)의 영향이 달라진다는 사실도 고려하여 교체 기법이 설계되어야 할 것이다.

이동 클라이언트의 캐쉬 교체 기법중 유용성이 인정된 기법으로는 FBR(Frequency Based Replacement) 기법과 pinning 기법이 제시되어 있다. FBR 기법은 데이타의 참조 시점뿐만 아니라 참조 빈도를 같이 고려하는 교체 기법이며[7], pinning 기법은 특별히 지정된 화일들에 대해 교체 대상에서 제외시킬 수 있도록 지원하는 기법이다[8]. 이 외에도 뒤에서 언급할 prefetching 기법과 hoarding 기법을 연계하여 교체 기법이 설계되어야 할 것이다.

3.3 Write-back 기법

Write-back이란 클라이언트에 의해 변경된 데이타 또는 화일을 서버에게 되돌려 전송함으로써 변경 내용을 서버측에 반영하는 일을 말하며, 캐쉬 관리 시스템의 write-back 기법은 이와같은 변경내용을 언제 어떠한 방법으로 서버측에 전송할 것인지를 결정하는 것이다. Andrew 화일 시스템이나 Coda 화일 시스템을 비롯한 대부분의 기존 화일 시스템들에서는 클라이언트측에서 화일이 close될때 그 변경 내용을 서버에 전송하는 write-on-close 기법을 사용하고 있다[1].

데이타 일치성 문제를 고려한다면 화일의 변경 내용을 변경 즉시 서버에 전송하는 것이 효과적이겠으나 이는 통신 오버헤드를 증가시키는 문제점을 갖고 있으며, 또한 이동 클라이언트의 경우에는 접속상태에 따라 전송이 불가능한 경우도 있을 수 있으며 또는 전송 소요시간이 매우 길어지는 상황도 발생할 수 있으므로 이러한 상황들을 모두 고려해야 한다. 또한 데이타의 가용성(availability)을 염두에 두고 낙관적 일치성 유지기법(optimistic consistency control scheme)을 사용할 경우 통신 오버헤드는 감소시킬 수 있으나 갱신 충돌(update conflict)의 가능성이 증가한다는 사실과 데이타의 변경과 관련한 클라이언트측 로그(log)의 크기가 증가한다는 사실을 고려해야 한다.

이동 컴퓨팅 환경에서는 데이타의 가용성을 우선 고려하여 일반적으로 낙관적인 일치성 유지 기법을 사용하기 때문에 delayed write-back 기반의 기법들을 선호할 것으로 보인다. 실제로 Andrew 화일 시스템에서는 통신 부류별로 서로 다른 네트워크 큐들을 두고 이들에 대해 lottery 스케줄링을 사용하는 형태로 delayed write-back 기법을 사용하고 있으며[8], Coda 화일 시스템에서도 이러한 delayed write-back 기법의 일종으로 대역폭이 허락하는 범위내에서 조금씩 write-back을 진행하는 trickle reintegration 기법을 사용하고 있다[9]. 이는 분산 환경의 대부분의 화일들이 한 사용자에 의해 갱신되고, 따라서 갱신 충돌 현상은 매우 드물게 발생한다는 사실을 근거로 한 것이기도 하며, write-back을 지연시킴으로써 발생하는 성능 향상을 목적으로 한 것으로 보여진다.

또 다른 기법으로는 클라이언트가 변경한 화일의 내용 전송은 유보하되 특정 화일의 내용이 변경되었다는 write notification만을 서버에게 전달하도록 함으로서 대역폭 사용의 낭비를 줄이는 기법이 제시되어 있다[10].

3.4 Prefetching 기법

클라이언트측에서 접근할 가능성이 있는 화일들을 미리 캐쉬하는 것은 예측이 정확했을 경우 성능의 향상에 크게 도움이 될 것이다. 더욱이 이동 컴퓨팅 환경에서는 이동 클라이언

트가 접속상태가 좋지 않은 네트워크 환경으로 이동할 가능성이 있으므로 이를 대비하여 필요한 데이타들을 미리 캐싱하는 일은 매우 중요하다.

기존의 전통적인 기법에서는 접근하고자 하는 화일에 대한 사용자의 요구가 발생하면 클라이언트가 곧 서버에 저장된 화일의 전송을 요구하고 이를 전송받아 캐싱한 후 사용하는 정책을 사용했다. Prefetching이란 클라이언트가 특정 화일에 접근할 때 화일의 전송으로 인한 지연을 사용자가 느낄 수 없도록 하기 위하여 예측된 화일들을 미리 캐싱하는 일을 의미한다. 그러나 prefetching 판단이 적절하지 못하여 접근하고자 하는 화일을 미리 캐쉬하지 못하는 경우 부분연결 상태에서는 전송지연 시간이 매우 길어지게 될 것이고, 따라서 전송하는 동안의 지연을 사용자가 느낄 수도 있게 될 것이다. 현재까지는 prefetching에 대한 연구가 많지 않으나 앞으로 이동 컴퓨팅 환경과 관련해서는 이 부분에 대해서도 많은 연구가 진행되어야 할 것이다.

Columbia 대학에서는 화일 참조 패턴을 트리 구조로 표현하고 현재 참조된 화일을 저장된 패턴과 비교하여 해당 패턴에 명시된 화일들을 미리 캐싱하는 화일 단위의 prefetching 기법을 제안하고 있다[11]. 이 외에도 단절상태를 고려하여 제시된 Seer 예측 캐싱 기법[5]이 있으며, 이는 부분접속 상태에서 화일들을 그룹화하고 이를 미리 캐싱하도록 확장하는 기법이다.

3.5 Hoarding 기법

화일의 prefetching이 앞으로 필요할 것으로 예측되는 화일을 미리 캐싱함으로써 화일 전송 지연시간을 줄이는 기법임에 반하여, hoarding이란 접속상태가 정상적일 때 앞으로 네트워크 접속이 단절될 경우를 대비하여 필요한 화일들을 클라이언트 캐쉬에 미리 축적하는 작업을 의미한다. Hoarding의 궁극적인 목적은 앞으로 발생할 가능성이 있는 오랜 기간동안의 단절상태에 대비해 화일들의 가용성을 높이기 위한 것이다. 즉, 이후 오랜 기간동안의 단절상태에서 사용자가 필요로하게 될 화일들을 예측하고 이들을 미리 캐싱하는 것이 hoarding 기법의 가장 핵심적인 사항이다.

이 기법은 클라이언트측의 화일 참조 과정을 조사하여 기록해두고 이를 이용하여 hoarding의 대상 화일들을 선정하는 암시적인(implicit) 방법과, 사용자가 미리 지정해 놓은 화일들을 캐쉬하도록 하는 명시적인(explicit) 방법으로 분류된다. 암시적인 방법에서는 각 사용자의 화일 참조 기록을 계속 보존하는 작업이 필요할 것이며, 명시적인 방법에서는 사용자가 지정하는 화일들을 기록해 둘 hint 화일 등의 구성이 필요하다.

Coda 화일 시스템에서는 앞에서 언급한 두 가지 기법을 조합하여 사용하고 있으며, 여기에 우선순위 개념을 도입하여 두 가지 방법에 의해 선정된 화일들에 각각 우선순위를 부여하고 이 우선 순위에 따라 화일들을 캐싱하는 기법을 사용하고 있다[12]. 사용자가 제공하는 정보는 HDB(Hoard Database)에 기록하여 보존한다. 이 시스템에서는 주기적으로 또는 사용자의 요청이 있을 때 hoard walking이라는 과정을 수행함으로써 우선순위가 높은 화일들이 캐싱될 수 있도록 한다.

이외에도 Ficus 화일 시스템에서처럼 사용자의 개입 없이 시스템이 완전히 자동적으로 hoarding을 수행하도록 하는 경우도 있다[5]. 이 시스템에서는 상호 관련성이 있는 화일들을 그룹화하여 하나의 화일이 캐싱될 때 이와 관련된 화일들이 같이 캐싱되도록 함으로서 클라이언트가 단절되는 상황을 대비하고 있는 것이다.

3.6 적응성 지원 기법

이동 컴퓨팅 환경에서는 클라이언트나 서버가 적응성(adaptive)을 가지도록 하는 일이 필요하다. 특히, 이동 클라이언트는 수시로 위치를 이동할 수 있고 이에 따라 다양한 네트워크 접속 환경에 놓이게 된다. 이동 클라이언트가 완전연결 상태에 있을 경우와 부분연결 상태에 있을 경우, 그리고 단절상태에 있을 경우 각각에 대해 캐싱과 관련된 각 기법들은 가능하면 전체 시스템의 효과적 운영을 위해 동적으로 동작해야 할 것이며, 또한 서버측의 서비

스의 질(quality of service)도 이러한 동작 환경에 따라 변화될 수 있도록 해야 할 것이다.

앞에서도 이미 언급하였듯이 캐쉬의 단위나 prefetching 기법, 그리고 write-back 기법 등이 통신 환경에 따라 가변적으로 변화할 수 있는 요소들이며, 사용자의 작업 패턴등에 따라 prefetching 기법이나 hoarding 기법 등이 적응성있게 변화할 수도 있을 것이다. 또한 통신 환경에 따라 서버측의 서비스의 질을 변화시키는 목적, 즉, application-aware adaptation을 위해 현재 Carnegie-Mellon 대학에서는 Odyssey라는 프로젝트가 진행중에 있다[13].

캐쉬 관리자의 입장에서 변화하는 환경에 적절히 적응하기 위하여 수행해야할 일들을 세가지로 구분해 볼 수 있다. 첫째는 현재의 네트워크 접속 환경을 알아야 한다는 것이고, 둘째는 현재의 작업부하를 알아야 한다는 것이다. 작업부하 추적(workload tracking) 기법은 작업부하와 관련한 앞으로의 경향을 추측하는데 도움을 줄 수 있다. 마지막으로 캐쉬 관리자에게는 스스로 취득할 수 없는 정보를 사용자에게서 직접 얻어내는 기능이 필요할 수 있다. 이동성의 궁극적인 목적은 사용자에게 투명성을 제공하는 것이지만 때에 따라서는 시스템 환경에의 적절한 적응을 위해 사용자에게 어느 정도의 부담을 주는 것이 보다 효과적일 수도 있는 것이다.

4. 분산 화일 시스템 사례연구

초기에 분산 컴퓨팅 환경에서의 효과적인 정보 공유의 수단으로 구성되었던 분산 화일 시스템이 최근에는 이동 컴퓨팅 환경을 지원할 수 있는 분산 화일 시스템으로 발전되어가고 있다. 대표적으로 Sun Microsystems의 NFS (Network File System)에서 시작하여 Carnegie-Mellon 대학의 AFS(Andrew File System), 그리고 뒤를 이어 단절상태를 고려하여 제시된 CFS(Coda File System) 등은 초기의 분산 화일 시스템으로부터 이동 컴퓨팅 환경을 고려한 화일 시스템으로의 변화과정을 보여주고 있다(그림 5). 본 절에서는 3절에서 언급한 이동 컴퓨팅 환경의 화일 시스템 설계사항들과

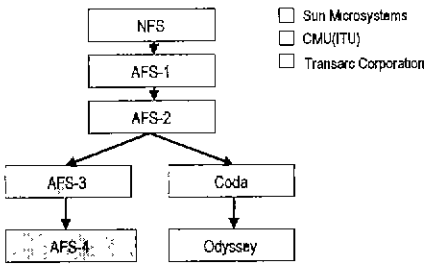

그림 5 분산 화일 시스템 발달 과정

관련하여 몇몇 실제 시스템들의 특성들을 언급한다.

4.1 Coda 화일 시스템(CFS)

CFS는 AFS(Andrew File System)의 많은 특성들을 유지하면서 단절상태에서도 자원에 대한 높은 가용성을 제공할 수 있도록 Carnegie-Mellon 대학에서 설계된 화일 시스템이다 [1, 12]. 즉, CFS은 기존의 분산 화일 시스템 골격을 유지하면서 클라이언트의 이동성을 어느 정도 지원할 수 있도록 설계된 화일 시스템이다.

Carnegie-Mellon 대학에서는 최근 CFS외에도 이동 컴퓨팅 환경에서의 응용-인식 적응성(application-aware adaptation)을 지원하기 위한 오디세이(Odyssey) 프로젝트를 진행하고 있다[13]. 이 프로젝트에서는 응용 프로그램의 특성과 현재 네트워크 접속상태를 고려하여 서버측의 서비스의 질을 조절하는 기법에 대한 연구가 주로 진행되고 있다.

4.2 Ficus 화일 시스템

UCLA에서 개발된 Ficus 화일 시스템은 초대형 규모의 분산 컴퓨팅 환경을 위해 만들어졌으며 같은 대학에서 개발되었던 Locus의 특성들을 많이 전수받고 있다[5]. 이 시스템의 특성은 앞에서 언급한 CFS에서와 같이 네트워크 분할(network partition)이 있을 경우에도 화일에 대한 갱신을 허용하며 갱신 충돌이 발생했을 때 이를 탐지하여 사용자에게 알려줄 뿐만 아니라 디렉토리의 경우처럼 충돌 해결이 가능한 경우에는 이를 자동으로 해결하는 역할도 수행한다.

5. 결 론

이동 컴퓨팅 환경에서는 호스트가 위치를 이동하고 있는 도중에도 서버측과의 연결이 계속 이루어지도록 지원한다. 다만, 이러한 환경에서는 이동 호스트가 무선매체를 이용하여 서버측의 지원을 받아야 하므로 접속 상태가 다양하게 나타날 수 있으며 사용자에게는 이와 같은 접속 상태에 관계없이 최대한의 서비스를 지원하는 기법이 필요한 것이다.

현재까지 연구되고 개발되어 온 대부분의 분산 화일 시스템에서는 이동 컴퓨팅 환경에 대한 고려가 그리 많지 않았으며, Carnegie-Mellon 대학의 CFS와 UCLA의 Ficus 화일 시스템, Columbia 대학의 이동 컴퓨팅 환경을 위한 분산 화일 시스템 등을 비롯한 몇몇 시스템에서 클라이언트와 서버가 단절되는 상황을 고려하여 이를 대비한 기법들을 설계·구현하고 있는 정도이다. 이동 컴퓨팅 환경에서는 이와 같은 단절 상태를 대비한 기법들 뿐만 아니라 앞으로는 부분연결 상태 등에 대비한 화일 시스템에서의 지원 기법등과 관련하여 더욱 많은 연구가 진행되어야 할 것이고, 또, 그러한 연구가 진행될 것으로 기대된다.

참고문헌

[1] Coulouris, G., Dollimore, J., and Kindberg, T., Distributed Systems : Concepts and Design, 2-ed., Addison-Wesley, 1994.

[2] Badrinath, B. R., Acharya, A., and Imielinski T., "Impact of Mobility on Distributed Computations," ACM Operating Systems Review, Vol. 27, No. 2, Apr. 1993.

[3] Forman, G. H., Zahorjan, J., "The Challenges of Mobile Computing," IEEE Computer, Vol. 27, No. 4. Apr. 1994.

[4] Froese, K. W., "File Cache Management for Mobile Computing," MS Thesis, Dept. of Computer Science, Univ. of Saskatchewan, Canada.

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

[6] Willick, D., Eager, D., and Bunt, R., "Disk Cache Replacement Policies for Network File Servers," In Proc. 13th International Conference on Distributed Computing Systems, Pittsburgh, PA, May 1993.

[7] Robinson, J., Devarakonda, M., "Data Cache Management using Frequency-based Replacement." In Proc. ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, Boulder, CO, May 1990.

[8] Huston, L., Honeyman, P., "Disconnected Operation for AFS," In Proc. 1st USENIX Symp. on Mobile and Location-Independent Computing, Cambridge, MA, Apr. 1993.

[9] Mummert, L., Ebling, M., and Satyanarayanan, M., "Exploiting Weak Connectivity for Mobile File Access," In Proc. 15th ACM Symposium on Operating Systems Principles, Copper Mountain Resort, CO, Dec. 1995.

[10] Tait. C., Duchamp, D., "An Efficient Variable Consistency Replicated File Service," In Proc. USENIX File Systems Workshop, Ann Abor, MI, May 1992.

[11] Lei, H., Duchamp, D., "Transparent File Prefetching," Technical Report, Dept. of Computer Science, Columbia Univ., New York, NY, 1995.

[12] Kistler, J., Satyanarayanan, M., "Disconnected Operation in the Coda File System," ACM Transactions on Computer Systems, Vol. 10, No. 1, Jan. 1992.

[13] Noble, B. D., et. al., "Agile Application-Aware Adaptation for Mobility," In Proc. 16th ACM Symposium on Operating System Principles, Oct. 1994.

[14] Baker, M., Hartman, J., Kupfer, M., Shirriff, K., and Ousterhout, J., "Measurements of a Distributed File System," In Proc. of the 13th ACM Symposium on Operating System Principles, Pacific Grove, CA, Oct. 1991.

박 재 원

1990 성균관대학교 정보공학과 학사
1996~현재 성균관대학교 정보공학과 컴퓨터공학 전공 석사과정
관심분야 : 분산 시스템, 이동 컴퓨팅 시스템

김 문 정

1997 성균관대학교 정보공학과 학사
관심분야 : 분산 시스템, 이동 컴퓨팅 시스템

엄 영 익

1979 서울대학교 계산통계학과 학사
1983 서울대학교 계산통계학과 전산과학전공 석사
1985 서울대학교 계산통계학과 전산과학전공 박사
1986 서울대학교 도서관 조교
1993 단국대학교 전자계산학과 부교수
1993~현재 성균관대학교 정보공학과 교수
관련분야 : 분산 시스템, 분산 객체시스템, 이동컴퓨팅 시스템, 운영체제