OSF/1 AD 운영체제의 프로세스 이전 기법

Process Migration Facility in OSF/1 AD

조일연(I. Y. Cho) 시스템S/W연구실 연구원
이재경(J. K. Lee) 시스템S/W연구실 책임연구원
김해진(H. J. Kim) 시스템S/W연구실 책임연구원, 실장

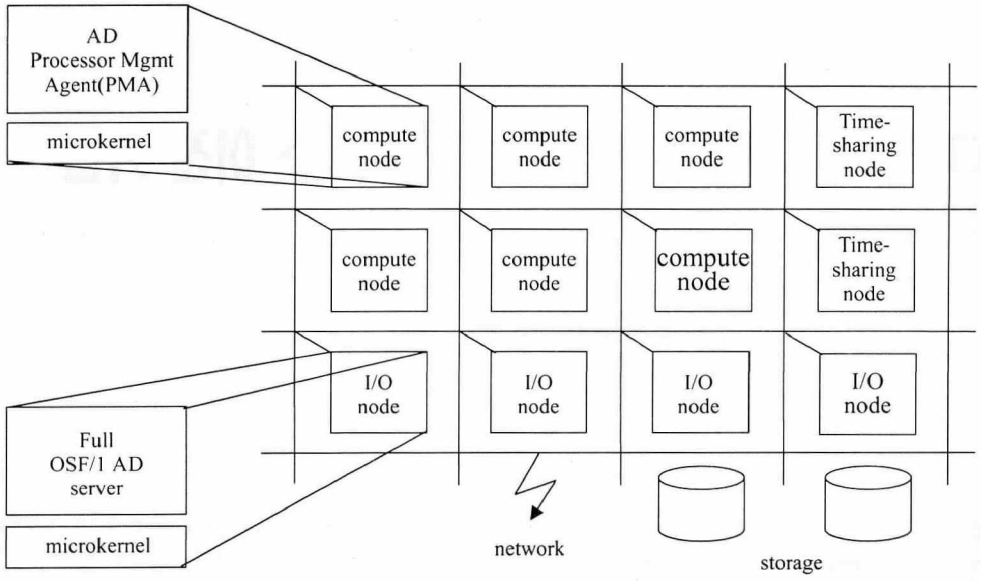
OSF/1 AD는 여러 대의 워크스테이션을 연결한 클러스터링 시스템에서부터 인텔 파라곤(Paragon)과 같은 massively parallel processing (MPP) 시스템에 이르는 다양한 시스템 상에서 단일시스템 이미지를 효과적으로 제공하는 다중컴퓨터용 운영체제이다. 본 고에서는 OSF/1 AD 운영체제의 기본구조와 분산 시스템의 부하 분산 기능 구현의 핵심 요소인 프로세스 이전 기법의 구현 방법에 대하여 살펴보았다.

I. 서 론

최근 들어 분산 시스템이 일반화되면서 시스템의 각 노드에 고르게 부하가 분산될 수 있도록 하는 부하 분산 기능에 대한 요구가 증가하고있다. 부하 분산을 구현하기 위한 기본 요건이면서 가장 중요한 구성 요소는 프로세스 이전이다. 프로세스 이전은 현재 수행중인 프로세스를 시스템내의 다른 노드로 동적으로 이동시켜주는 기능이다. 이러한 프로세스 이전 기법은 부하 분산 분야 외에도 다음과 같은 곳에서 유용하게 사용된다.

- 자원의 지역성 활용: 필요한 자원이 원격 노드에 있는 경우에 실행중인 프로세스를 자원이 위치하고 있는 노드로 이주시킴으로써 보다 효율적으로 자원에 접근할 수 있도록 한다.
- 신뢰성: 부분적으로 고장이 발생한 노드에서 실행중인 프로세스를 다른 노드로 이주 시킴으로써 시스템의 신뢰성을 높일 수 있다.
- 시스템 관리: 긴 실행 시간을 요구하는 프로세스가 실행 중인 경우에 시스템 보수 등 의 목적으로 시스템을 재 시작하려면 이러한 프로세스를 다른 노드로 이주시켜 작업을 수행할 수 있다.

프로세스의 이전은 다양한 수준에서 구현 가능하다. 운영체제 커널내에 구현된 예로는 MOS(IX)[1], Locus[2], Sprite[3]등이 있으며, 마이크로커널 구조를 기반으로 하는 운영체제에 구현된 예로는 V 커널[4], Mach[5] 등이 있다. Condor[6]는 사용자 수준의 라이브러리로 프로세스 이전을 구현하였다. 본 고에서는 이렇게 다양하게 구



(그림 1) MPP 상의 OSF/1 AD 운영체제 구성 예

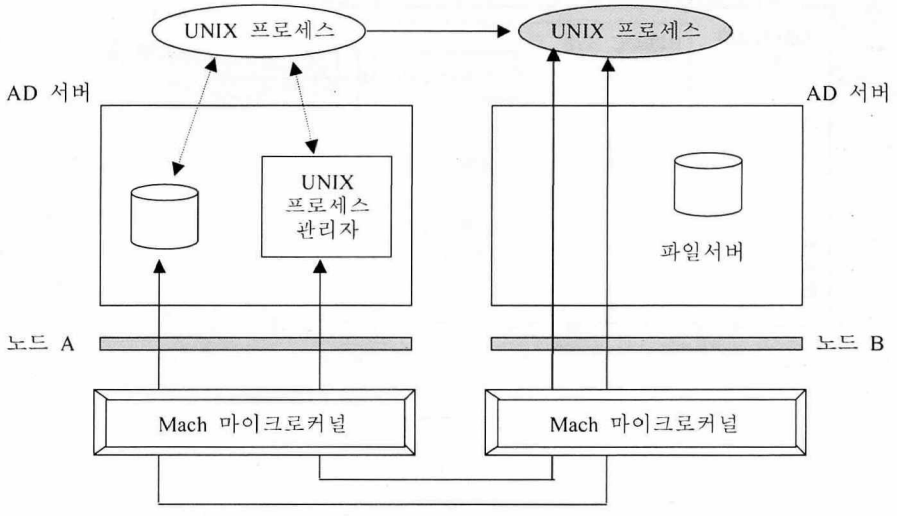
현된 프로세스 이전 기법 중 에서 마이크로커널을 기반으로 하고 다중컴퓨터 환경을 지원하는 운영체제인 OSF/1 AD[7] 상에 구현된 사례에 대하여 자세히 살펴보았다.

본 고의 구성은 다음과 같다. 2장에서는 Mach 마이크로커널과 OSF/1 AD 운영체제의 기본 구조와 기능에 대하여 살펴보고, 3장에서 프로세스의 분산관리에 대하여 설명한다. 4장에서 현재 Mach에 구현되어 있는 프로세스 이전기법에 대하여 설명하고, 5장에서 결론을 맺는다.

II. Mach와 OSF/1 AD

Mach 3.0 마이크로커널[8]은 시스템내의 모든 노드에서 실행되며, 태스크 및 쓰레드 관리, 프로세스간 통신(IPC), 메모리 및 장치 관리 등의 기능을 제공한다. 시스템 내의 가용한 모든 서비스와 자원은 마이크로커널 또는 태스크에 의해 capability[1]로 표현된다. Mach와 같이 마이크로커널 구조를 가지는 분산운영체제의 동작 환경 중에서 가장 중요한 요소는 단일 시스템 이미지를 제공한다는 것이다. 즉, 시스템의 자원이 어느 노드에 존재하는지에 상관없이 접근이 가능하도록 운영체제가 기능을 제공한다는 것이다. 이는 또한 투명한 프로세스 이전 기능을 제공할 수 있는 기반이 된다. 마이크로커널

1) Mach 마이크로커널에서는 커널내의 객체들에 대한 사용권한을 얻어야만 서비스를 제공 받을 수 있다. 이러한 권한을 capability라 한다.



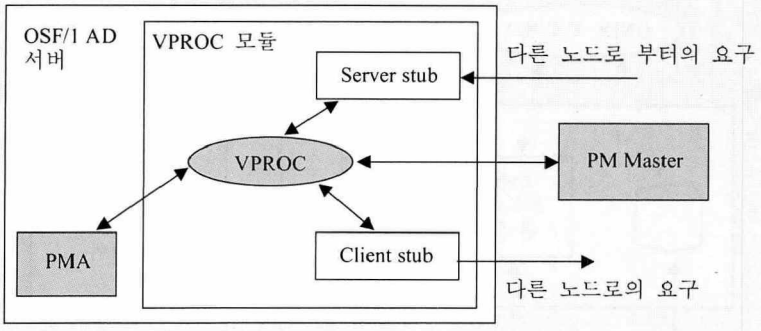
(그림 2) OSF/1 AD와 원격 프로세스

구조의 대표 격인 Mach, Chorus[9], Spring[10] 등의 운영체제는 모두 투명한 네트웍 IPC를 제공한다.

Mach의 가장 큰 장점으로는 메시지 전달 기법을 사용하여 투명한 노드간 통신 시스템을 제공하는 DIPC(Distributed InterProcess Communication)와 분산 메모리 관리 기법을 제공하는 XMM(eXtended Memory Management)을 들 수 있다. DIPC는 기존에 사용되어온 IPC를 응용프로그램이 어느 노드에서 실행이 되는 지에 관계없이 투명하게 태스크, 쓰레드, 메모리 객체, 포트 등과 같은 Mach의 객체에 접근할 수 있도록 개선한 것이다. XMM은 DSM(Distributed Shared Memory)을 지원 할 수 있도록 개선되었다.

OSF/1 AD1은 확장성과 고성능을 제공하는 OSF/1의 첫 번째 다중컴퓨터용 프로토타입 UNIX로 단일 시스템 이미지를 제공한다. OSF/1 AD1의 개발 목적은 인텔 파라곤 같은 MPP(Massively parallel processing) 시스템에서부터 여러 대의 워크스테이션을 연결하여 운용하는 클러스터링 시스템에서 MPP환경을 제공하기 위함이다. OSF/1 AD1 운영체제는 특수한 목적을 가지는 시스템서버들(화일서버, 프로세스 관리 서버 등) 이 노드에 분산되어 실행된다. 이들 시스템 서버의 가장 중요한 기능은 전체 시스템에 걸쳐 단일 이름 공간을 가지도록 하는 분산 파일시스템과 프로세스 관리 서버이다. (그림 1)에 OSF/1 AD 운영체제가 MPP 시스템 상에 구성된 한 예를 나타내었다.

기존의 UNIX 프로세스는 Mach의 태스크와 쓰레드로 연결되어 마이크로커널에 의해 관리된다. OSF/1 AD에서는 쓰레드가 프로세스의 코드를 실



(그림 3) 분산 프로세스 관리

행한다. 즉, 실행의 단위가 쓰레드이다. Mach 마이크로커널에서는 기존 UNIX와의 호환성을 제공하기 위해서 각 태스크에 애뮬레이션 라이브러리가 존재하는데, 이는 모든 시스템 호출을 받아들여서 이를 UNIX 서버를 위한 서비스 요구 메시지로 변화하여 주는 기능을 담당한다.

프로세스의 실제 위치에 상관없이 원하는 프로세스에게 서비스를 요구할 수 있는 기능을 제공하는 가상 프로세스(Virtual Process: *vproc*) 모듈을 운영체제 서버가 제공한다. *vproc* 이 프로세스와 관련된 투명성을 제공해주는 것과 같이 vnodes는 파일에 대한 투명성을 제공한다. OSF/1 AD TNC(Locus 사에서 개발)의 프로세스 관리 서브시스템은 시스템의 모든 노드에 분산되어 있다. 그러나 OSF/1 AD는 이러한 분산 개념을 가지고 있지 않기 때문에 Mach 태스크를 원격 노드에 생성하고 관리하는 하나의 노드(마스터 노드)를 두어 시스템내의 모든 태스크를 관리하도록 하였다. 이러한 경우에((그림 1)참조) 프로세스 관리 서브시스템은 노드 A에만 존재하며, 원격 노드에 실행되는 모든 태스크들을 생성하고 관리한다.

III. 분산 프로세스 관리

다중 컴퓨터 환경에서 분산 프로세스 관리의 가장 중요한 기능은 모든 프로세스에 대한 투명한 접근과 원격 작업의 지원이다. OSF/1 AD1의 후속 버전인 AD2와 AD3에서 이러한 기능이 구현되었다. (그림 2)에 분산 프로세스 관리에 필요한 구성요소와 요소들간의 상호 관계를 나타내었다. 새로이 구현된 vproc 모듈은 다음과 같이 단순하면서도 개방적인 구조를 가지고 있다.

- **PMA: Process Management Agent (또는 System Call Agent)**

시스템 내의 모든 노드에 실행되면서 지역노드에 실행중인 프로세스들에 대한 프로세스록, 시그널 등의 자료구조를 유지한다. 자신 외에 다른 프로세스(원격 노드에 있는 프로세스이건 지역노드에 있는 프로세스이든지에 상관없이)에 대한 작업을 하는 모든 경우에 vproc 모듈을 사용하게 된다.

- **가상 프로세스 인터페이스 (Virtual Process Interface)**

가상 프로세스 인터페이스는 *vproc* 구조체와 이와 관련된 운영 테이블로 구성되어 있다. 이 인터페이스는 프로세스의 위치에 상관없이 투명하게 접근할 수 있도록 하는 기능을 제공한다. 운영 테이블은 프로세스의 위치에 따라서 PMA 운영 또는 *client stub*를 세팅 한다. *server stub*는 원격요구를 받아서 이를 지역에서 실행할 수 있도록 한다. *vproc* 모듈은 UNIX 프로세스가 다른 노드에 투명하게 분산되어 실행될 수 있도록 하는 기능과 시스템 전체에 걸쳐 단일시스템 이미지를 제공할 수 있게 하여 주는 핵심요소이다.

- **PMM: Process Management Master (PM Master)**

하나의 노드에서 실행되면서 프로세스간의 관계 관리, 프로세스 세션과 프로세스 그룹 관리, 프로세스 ID 관리, 프로세스 생성시 PID 부여 등의 기능을 제공한다. PMM은 앞서 언급한 관리나 프로세스의 위치와 관련된 작업(fork(), exit(), wait(), killpg() 등과 같은 시스템 호출)이 이루어지면 *vproc*과 연계하여 작업을 수행한다.

IV. 프로세스 이전

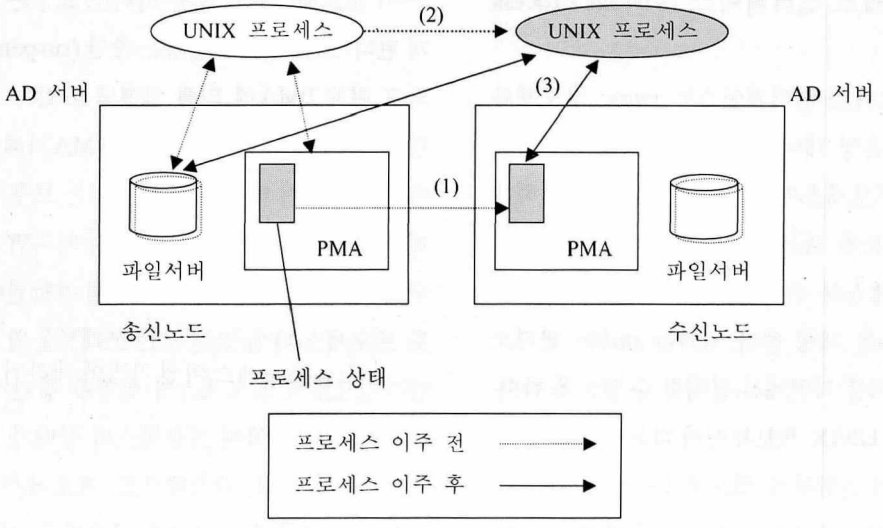
본 절에서는 OSF/1 AD에 구현되어 있는 프로세스 이전 기법을 단계별로 자세히 설명한다.

1) 이전 대상 프로세스의 상태를 고정시킨다.

대상 프로세스의 상태가 stable, coherent, resumable이 되었을 때 고정시킨다. 이후에 발생되는 이 프로세스로의 모든 서비스 요구는 큐에 쌓이게 된다. 프로세스의 실행은 중단(suspend) 상태가 되고 현재 PMA에 의해 실행되고 있는 모든 시스템 호출은 대기 상태가 된다. PMA의해 접수되어 아직 처리되지 않은 서비스 요구는 모두 큐에 쌓이게 된다. 이전이 성공적으로 이루어지면 이러한 요구들은 이전된 노드의 PMA에게 전달된다. 다음으로 프로세스 내에 있는 모든 쓰레드들의 실행을 중단시킴으로써 프로세스의 실행을 중단시킨다. 이러한 작업을 통하여 프로세스의 상태가 바뀌는 것을 막을 수 있다. 마지막으로, 프로세스가 실행한 모든 시스템 호출의 종료를 기다린다. 시스템 호출의 종료를 기다리는 것은 프로세스 이전시간이 늘어나는 요소중의 하나다. 이러한 지연을 없애기 위한 대안으로는 현재 수행중인 시스템 호출 중 프로세스의 상태를 변화시키지 않는 시스템 호출(예를 들어 wait(), sigsuspend() 등)을 중단시켰다가 이전이 완료되면 이전된 노드에서 이를 다시 호출하도록 하는 방법이 있다.

2) PMA에서 대상 프로세스를 떼어내고 이전을 시작한다.

이 단계에서는 프로세스의 상태가 반드시 stable, coherent 이어야 하며 이전되기 전에 수행된 시스템 호출로 인하여 종료되지 않아야 한다. 대상 프로세스의 상태를 PMA에서 떼어낸다. 이 프로세스를 active process 목록에서 제거하여 migrating process 목록에 넣는다. 이전대상 프로세스의 자료구조 즉, 자원의 사용량과 credential entry등을 전송을 위하여 복사한다. 이전대상 프로세스의 복사본은 이제 이전을 위한 준비가 끝난 상태이며, (그



(그림 4) 프로세스 이전 단계

림 4)의 (1)에 나타낸 것과 같이 UNIX 프로세스의 이전을 시도하기 위하여 수신노드로 보내어진다. 이 단계에서 이전 요구가 수신노드의 상태에 따라서 거절될 수도 있다.

3) 수신노드에 이전 대상 프로세스를 설치한다.

수신노드가 이전을 받아들이고 대상 프로세스의 상태를 수신함에 따라서 수신노드의 PMA는 새로운 자료 구조체와 가상 프로세스를 대상 프로세스의 ID와 똑 같이 할당한다. 필요한 자원이 할당되면 PMA는 태스크 이전 기법[5]을 사용하여 UNIX 프로세스와 "mapping"되어 있는 Mach 태스크를 알아낸다. Mach 태스크의 상태를 알아낸 PMA는 이를 참조하여 대상 프로세스의 나머지 상태를 설정한다. 이전된 프로세스의 스케쥴링 우선 순위에 대해서는 아무런 준비를 하지 않는다. 그 이유는 송신노드에서 이 프로세스의 우선 순위가 노드의 부하 정도에 따라서 바뀌었을 지도 모르기 때문이다. 이 시점에서부터 프로세스는 수신노드의 지역 프로세스가 된다. 그러나 만일에 프로세스가 바로 재시작되면 대기하고 있던 시스템 호출이 바로 송신노드의 PMA에게 요구 될 것이다. 이러한 문제를 해결하기 위해서 프로세스의 송신노드를 가리키고 있는 프로세스의 *capability*를 수신노드를 가리키게 바꾸어 주는 작업을 한다(그림 4)의 (3). 다른 *capability*와 파일 서버에 대한 종속성은 그대로 남아있게 된다.

이 시점에서 송신노드는 이전대상 프로세스의 이전을 포기하고 프로세스를 재시작 시킬 수 있다. 만일에 프로세스 이전 뒤에 수신노드에 고장이 발생한다면 이전된 프로세스는 다른 보통의 지역 프로세스처럼 죽게될 것이다. 송신노드에 고장이 발

생하는 경우에 송신노드의 큐에 쌓여있는 시스템 호출이 없는 경우에 바로 수신노드에서 이전된 프로세스를 재시작하면 아무런 문제없이 프로세스가 실행되어야 할 것이다. 그러나 가상 메모리의 늦은 전달 속도 때문에 이전된 프로세스의 주소공간은 아직까지도 송신노드와 연관을 가지게 된다. 따라서 이러한 경우에는 이전된 프로세스도 송신노드 고장의 영향을 받아 죽게 된다. 수신노드로부터 이전이 거부되는 경우에 PMA는 수신한 모든 상태와 *capability*를 클리어 시킨다. 이러한 이전 거부는 보통 자원 사용의 한계를 넘거나 자원이 고갈된 상태(주로 메모리)에서 발생된다. 이러한 경우에 이전은 단계 4)를 생략하고 단계 5)를 수행한다.

4) 수신노드는 PMM에게 프로세스의 이전을 알리고 프로세스를 재시작 시킨다.

PMM은 이전된 프로세스가 실행되고 있는 노드를 알게되고, 이전된 프로세스는 *active process* 목록에 넣어져 이전되기 바로 전의 상태에 따라서 재 시작된다. 이 시점이 되어서야 수신노드의 PMA는 송신노드로부터 프로세스에 대한 제어권을 완전하게 넘겨받게 된다.

5) 송신노드의 정리와 계류중(pending)인 서비스 요구의 전달

프로세스 이전이 성공적으로 완료되면 송신노드의 PMA는 이전이 시작된 이후로 요구된 시스템 호출을 수신노드에게 전달한다. 그리고 나서 송신노드의 PMA는 이전 전에 저장되어 있던 모든 자료구조와 *capability*를 소멸시킨다. 이전된 프로세스의 자료구조를 *migrating process* 목록에서 제거하고 *free process* 목록에 반환한다. *vproc* 또한 이 노드에서 더 이상 사용되지 않으면 반환한다. 이렇게 함으로써 이전된 프로세스와 송신노드의 PMA와는 아무런 종속관계도 가지지 않게 된다.

V. 결 론

본 고에서는 단일시스템 이미지를 제공하는 확장성 있는 다중컴퓨터용 운영체제인 OSF/1 AD의 기본구조와 프로세스 이전 기법에 대하여 자세하게 살펴보았다. 현재 OSF/1 AD를 기본으로 하는 운영체제가 탑재되어 있는 대표적인 MAP 시스템으로는 인텔사의 파라곤 시스템을 들 수 있다. OSF/1 AD 개발 프로젝트는 현재 OSF RI에서 계속 수행중에 있으며, 앞으로의 개발방향은 인터넷 사용자들의 요구를 만족시켜줄 수 있는 확장성과 보안성 있는 운영체제의 개발에 중점을 두고 있다.

참 고 문 헌

[1] A. Barak and A. Shiloh, "A distrbuted load-balancing policy for a multicomputer," *Software-Pratices and Experience*, vol. 5, no. 9, pp. 901-913, Sep. 1985.

[2] B. Walker and R. M. Mathew, "Process migration in AIX's transparent computing facility," *IEEE TCOS Newsletter*, vol. 3(1), pp. 5-7, Winter 1989.

[3] F. Douglis and J. Ousterhout, "Transparent process migration: Design alternatives and the sprite implementation," *Software Pratices and Experience*, vol. 2, no. 8, pp. 757-785. Aug. 1991.

[4] M. Theimer, K. Lantz, and D. Cheriton, "Preemtable remote execution facilities for the V system," *Proc. of the 10th ACM Symposium on OS Principles*, Dec. 1985, pp. 2-12.

[5] D. Milojicic, W. Dangel, and P. Giese, "Task migration on the top of the Mach Microkernel," *Proceedings of the*

third USENIX Mach Symposium, Santa Fe, New Mexico, Apr. 1993, pp. 273-290.

[6] M. Litzkow and M, Solomon, "Supporting checkpointing and process migration outside the UNIX kernel," *Proc. of the USENIX Winter Conference*, San Francisco, Jan. 1992, pp. 283-290.

[7] R. Zajcew *et al.*, "An OSF/1 UNIX for massively parallel multicomputers," *Proc. of the Winter USENIX Conference*, Jan. 1993, pp. 449-468.

[8] M. Accetta *et al.*, "Mach: A new kernel foundation for UNIX development," *Proc. of the Summer USENIX Conference*, Atlanta, GA, 1986, pp. 93-112.

[9] M. Rozier, V. Abrossimov, F. Armand, I. Boule, M. Gien, M. Guillemont, F. Herrmann, C. Kaiser, S. Langlois, and P. Leonard, "CHORUS distrbuted operating systems," *USERNIX Computing Systems*, vol. 1, no. 4, pp. 305-370, Fall 1998.

[10] G. Hamilton and P. Kougiouris, "The Spring nucleus: A microkernel for objects," Sun Microsystems Lab., Inc., Technical Report SMLI TR-93-14, Apr. 1993.