클러스터를 위한 소프트웨어 도구†

숭실대학교 최재영* · 이준호* · 황석찬

1. 서 론

컴퓨터 기술이 발달하면서 고성능 마이크로 프로세서, 그리고 높은 대역폭과 낮은 지연 시간을 가지는 네트워크를 저렴하고 손쉽게 구할 수 있게 되었다. 컴퓨터 성능이 향상되고 가격이 하락하면서 PC 또는 워크스테이션들을 네트워크로 연결하여 대용량 컴퓨터의 성능을 가질 수 있는 클러스터 시스템에 많은 관심이 집중되고 있다.

저가격으로 효과적인 성능을 낼 수 있도록 클러스터 시스템을 구현하기 위해 쉽게 구할 수 있는 하드웨어와 무료로 배포되는 Linux 운영체제를 비롯한 공개용 소프트웨어를 사용한다. Windows NT를 이용하기도 하지만, 운영체제 자체의 가격이 비싸고 NT를 지원하는 소프트웨어 도구들을 쉽게 구할 수 없으므로(무료가 아님) 대부분의 클러스터 시스템의 운영체제는 Linux를 사용하고 있다.

클러스터 시스템을 효과적으로 사용하기 위해서는 운영체제뿐만 아니라 여러 가지 소프트웨어 도구들을 갖추어야 한다. 클러스터 시스템의 하드웨어는 비교적 쉽게 구성할 수 있지만, 일반 사용자의 경우 클러스터를 위한 소프트웨어를 설치하기가 쉽지 않다. 운영체제인 리눅스를 설치하고 사용자 작업을 실행시키는 MPI와 같은 병렬 소프트웨어 패키지, 화일 공유와 사용자 인증 문제를 위한 NFS(Network File System), NIS (Network Information System)를 설치하려고 할 때 숙련된 사용자가 아니라면 여러 번의 시행착오를 겪기도 한다. 이와 같은 기본적인 운영 소프트웨어를 설치하려고 할 때, 예전에 비해서 쉽게 설치할 수 있지만 아직도 까다로운 설정 부분이 있다. 더구나 설치하는 노드의 수가 많아질수록 그 어려움은 더욱 늘어날 것이다.

비록 기본적인 설정이 되었더라도 클러스터 시스템을 원래 목적대로 사용하기 위해서는 다른 소프트웨어 도구들이 필요하다. 전통적인 유닉스 명령에 기반한 관리는 숙련된 사용자가 아니면 매우 어려워서 GUI 기반의 사용자 인터페이스를 지원하는 관련 소프트웨어 도구들을 필요로 한다. 그리고, 클러스터 시스템을 최대로 활용하기 위한 작업 관리 도구의 사용도 필요하다.

지금까지 일반 사용자들은 클러스터 시스템을 사용하기 위해서 기본 운영 소프트웨어(운영체제, 병렬 소프트웨어 패키지) 정도만을 설치하여 사용하는 경우가 많다. 개인용 컴퓨터와 달리 클러스터 시스템은 여러 컴퓨터를 동시에 운영하는 시스템이기 때문에 필요로 하는 소프트웨어가 다르다. 클러스터 컴퓨터의 각 노드를 총괄하여 설치/관리하는 작업부터 사용에 필요한 소프트웨어와 시스템 튜닝을 위한 소프트웨어 등이 필요하다.

본 논문에서는 기본적인 소프트웨어 도구 이외에 클러스터를 위한 소프트웨어 도구로 많은 클러스터 노드들을 간편하게 설치하는 도구와 설치된 클러스터의 환경을 설정하고 운영하는 관리도구, 작업에 필요한 프로그래밍을 위한 프로그래밍 도구(성능 감시기 및 디버거), 사용자의 작업

† 본 연구는 1999년도 과학기술부의 국책연구개발사업(98-NF-03-01-A-03)과 1999년도 교육부의 BK21 핵심분야사업(E-0075)의 지원을 받아 수행되었습니다

* 종신회원

처리를 위한 작업 관리 도구들을 살펴보도록 한다.

2. 클러스터 설치 도구

실제 클러스터 시스템의 연결 방법이나 이용되는 기술은 클러스터만을 위해 새롭게 개발된 것이 아니다. 물리적인 연결 관점으로 보면 기존의 NFS로 화일을 공유하고 PVM 혹은 MPI를 이용해서 워크스테이션을 연결한 COW(Cluster Of Workstation)와 크게 다르지 않다. 클러스터 시스템의 경우에서 각각의 노드들을 설정하는 과정은 각각의 단일한 워크스테이션으로 이루어진 COW와는 다른 설치(OS 및 Application Software의 Install) 방법이 필요하며 전체 클러스터 시스템의 노드 숫자가 많아질 경우 각각의 노드를 빠른 시간 내에 설치할 수 있는 방법이 더욱 필요하다.

이에 편리한 노드 설정 및 빠르고 자동화된 클러스터 기반의 OS와 응용 소프트웨어 설치 방법 및 각 노드들의 네트워크 설정, 클러스터 시스템에서 기본적으로 사용되는 데몬들의 쉬운 설정 기능 자체가 클러스터 관리 시스템에서 필요하다.

클러스터 시스템에서 각각의 노드를 구성하는 방법에는 물리적으로 두 가지 방법이 있다. Diskless Computer로 Slave 노드를 구성하고 Master 노드에 대용량의 하드디스크를 설치해서 NFS로 연결하는 방법과 각각의 노드를 각각 구성해서 모든 노드에 운영체제를 설치해서 NFS로 특정 부분만 공유하는 것이다. 첫 번째 방법의 경우 각각의 Slave 노드들은 하드디스크가 없기 때문에 NFS에 부하가 많다. 두 번째 경우 각각의 노드가 모두 하드디스크를 가지고 있으므로 운영체제나 기본적인 시스템에 관련된 라이브러리를 NFS로 공유하지 않아도 되므로 그 만큼의 NFS 부하가 감소한다. 하지만 운영체제 설치나 유지 보수가 복잡하다.

대표적인 상업용 리눅스 클러스터 관리 프로그램으로는 ALINKA사의 LCM(GPL로 공개된 Text 기반 프로그램)과 RAISIN(Web 기반 프로그램)이 있으며 PVM과 MPI 기반의 어플리케이션을 설치할 수 있는 Office Module을 제공하고 있다. 또한 캘리포니아 공대에서 운영하는 CACR(Center for Advanced Computing Research)의 beowulf Project에서는 nodecloner라는 텍스트 기반 운영체제 설치 스크립트 프로그램을 공개하고 있다. 이것은 운영체제 설치 부분에 국한된 것이지만 안정성과 쉬운 설치 방법을 제시해 준다. 미국의 National Institutes of Health에서 가동하고 있는 LoBos라는 클러스터 시스템에서도 하나의 플로피 디스크로 클러스터 노드에 운영체제 설치할 수 있는 기능을 LoBos Software Pack에서 제공하고 있다.

그림 1은 Nodecloner를 이용하여 각 노드의 운영체제를 설치하는 과정을 도식적으로 표현한 것이다. 클러스터 구축 자체는 많은 연구가 진행되고 있지만 클러스터 관리, 특히 운영체제나 응용 소프트웨어 설치와 같은 초기화 과정의 관리 도구들의 연구는 상대적으로 미비하다.

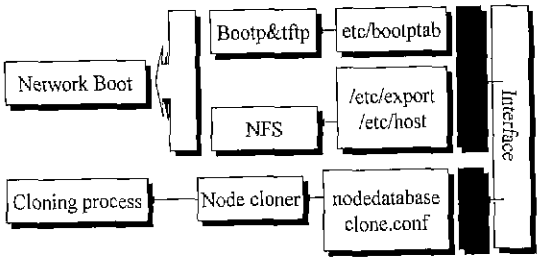

그림 1 운영체제 설치 프로세스

3 클러스터 관리 도구

클러스터 컴퓨터는 시스템의 규모가 커질수록 전체 시스템을 관리하기가 더욱 복잡해진다. 클러스터 환경에서는 여러대의 컴퓨터에서 개별적으로 운영체제가 실행되더라도 마치 하나의 운영체계가 있는 것처럼 시스템 관리자가 시스템을 운영할 수 있도록 단일 인터페이스를 제공하는 단일 시스템 이미지 기능을 제공하여야 한다. 클러스터 관리 시스템은 다음의 기본 요건을 갖추어야 한다.

· 가상의 단일 시스템 환경

다수의 노드로 구성된 클러스터 시스템을 단일 시스템처럼 인식할 수 있는 환경이 제공되어야 한다 즉 시스템 내의 어떠한 한 노드에서든지 클러스터내의 모든 시스템 자원과 행위에 대하여

간단하고 직관적인 관점을 제공하여야 한다. 따라서 시스템은 응용의 수행이 사용자에게 투명하게 진행될 수 있도록 하여야 하며, 어디서 작업이 수행되어지는가에 대해서 사용자가 구애받지 않아야 한다. 뿐만 아니라 시스템 관리자는 매우 간단한 작업으로 클러스터 노드들을 관리할 수 있어야 하며, 다양한 여러 자원들이 서로 다른 컴퓨터에 위치하더라도 하나의 명령으로 이를 관리할 수 있어야 한다. 단일 시스템 이미지는 모든 자원들에 대한 지속적인 탐색을 통해 시스템 관리자가 작업을 수행함에 있어서 각 자원들에 대한 실제위치에 구애받지 않도록 하여야 한다

· 사용자에게 친숙한 사용자 인터페이스

사용자의 응용 프로그램이 인터렉티브 또는 배치 모드로 수행될 수 있도록 사용하기 편리하고 친숙한 사용자 인터페이스가 제공되어야 하며 이것은 X 윈도우즈 인터페이스 또는 웹을 기반으로 수행될 수 있어야 한다. 배치 작업을 수행할 때 사용자가 시스템에 데이터의 집합, 실행 화일의 위치, 수행될 최대 시간, 작업이 순차적 혹은 병렬적 수행 여부를 쉽게 기술할 수 있어야 하고, 시스템은 이를 반영하여 스케쥴링을 통해 수행될 수 있도록 한다.

· 전체 시스템 파악의 용이

전체 시스템 구성과 각 노드의 부하 및 CPU, 메모리, 디스크 등 자원의 활용 상태의 파악이 용이해야 한다. 이를 위해 시스템 자원 모니터링 기능과 문제가 발생할 때 이를 시스템 관리자에게 알려줄 수 있는 알람 기능이 요구되며, 최종적으로 신뢰성 있는 시스템을 통해 사용자의 작업이 수행될 수 있어야 한다.

클러스터 관리 소프트웨어는 최근의 클러스터 열기로 인하여 많은 연구 개발이 이루어지고 있다. 다음은 기존의 개발된 클러스터 관리도구이다.

Berkeley에서 개발한 NOW 시스템 관리도구는 데이터베이스에 각종 정보를 모아서 저장하고, GLUnix라는 단일 시스템 이미지를 제공하고 있다. 이 시스템은 자바 애플릿을 사용하여 사용자가 브라우저를 통하여 모니터링할 수 있다.

MAT(Monitoring & Administration Tool)은 캐나다의 Ryerson 대학에서 개발한 관리도구로서 네트워크 환경하의 유닉스를 모니터링, 설정 관리하는 도구이다. MAT은 개별적인 노드의 화일시스템 백업, 복사 등의 관리와 자원의 모니터링 작업을 수행할 수 있다. 그리고, DNS(Domain Name Service)와 NFS, NIS를 원격으로 설정 관리하는 기능도 포함하고 있다.

SMILE은 K-CAP이라고 불리는 모니터링 시스템으로, 계산노드와 관리노드, 클러스터를 모니터하고 관리할 수 있는 클라이언트로 구성된 SMILE 클러스터 시스템의 관리도구이다 K-CAP은 자바 애플릿을 사용하여 클러스터의 미리 정의된 URL 주소를 통하여 관리한다.

VACM은 VA Linux에서 개발한 관리도구로 인텔보드의 IPMI(Intelligent Platform Management Interface)를 이용하여 각 장치의 전원관리부터 운영체제를 비롯한 각종 소프트웨어 제어 기능을 제공하고 있다. VACM은 사용자 인터페이스와 노드 제어기(클러스터 Front-end)에서 중앙 관리를 하는 Mayor Daemon, EMP 프로토콜을 이용하여 각 노드를 제어하는 agent로 구성되어 있다 VACM의 기능을 모두 활용하기 위해서는 전용의 메인보드(IPMI를 지원하는)를 사용해야 하는 제약이 있다.

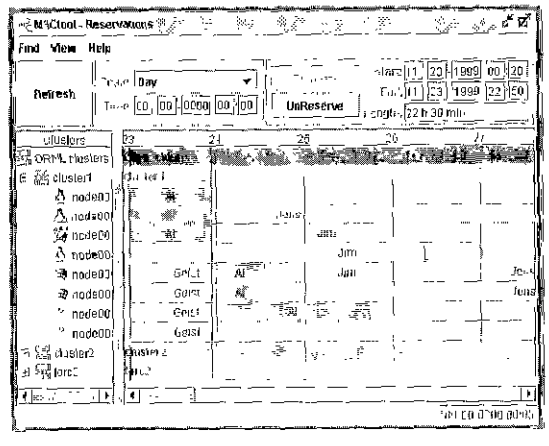

그림 2 M3C Reservations View

M3C(Managing Multiple Multi-user PC Clusters)는 HighTORC 클러스터의 관리용 소프트웨어로서 응용 소프트웨어의 설치에서부터

작업 스케줄링까지의 기능 등을 통합해 놓은 관리 시스템이다. 그리고, 인터넷에서의 사용자 접근을 가능하도록 Web 기반의 인터페이스를 제공한다. 다음 그림 2는 M3C의 작업 예약 화면을 보여주고 있다.

이 외에도 PARMON, MAT, WebRaT 등의 많은 관리도구가 개발되고 있다.

4. 프로그래밍 환경 및 도구

4.1 프로그래밍 도구

클러스터에서 작업할 수 있는 표준화된 병렬 프로그래밍 도구와 유틸리티는 문제 해결의 효율적인 처리를 도와준다. 병렬 프로그래밍 방식은 메시지 전송 방식과 공유 메모리 방식으로 구분된다. 메시지 전송 방식은 분산 메모리 시스템에서 가장 효율적인 병렬 프로그램을 작성할 수 있다. 대표적인 시스템으로는 PVM(Parallel Virtual Machine)과 MPI(Message Passing Interface)가 있다. 대부분의 클러스터 시스템에서는 MPI를 사용하고 있으며 각 통신 메커니즘에 최적화되어 있는 패키지도 개발되고 있다.

PVM은 오크리지 국립 연구소와 테네시 대학에서 공동으로 개발한 병렬 도구로 이기종의 컴퓨터에서도 동작할 수 있는 환경을 제공한다. 메시지 전송 방식을 기본으로 제공하고 있으며, 다른 병렬 컴퓨터에도 이식할 수 있도록 공유메모리에서의 프로세스간의 통신 방법도 같이 제공하고 있다. 이기종간의 네트워크 컴퓨터에서 주로 사용되었으나 클러스터 시스템에서는 많이 사용되고 있지 않다.

MPI는 1992년에 구성된 MPI 포럼에서 여러 형태의 병렬 컴퓨터를 대상으로 이식성 및 호환성, 성능, 구현의 용이성과 확장성 등의 주요 목표를 두고 만들어진 표준화 인터페이스이다. MPI는 병행 프로그램에서 프로세스들 집합 사이에서의 통신에 관한 기술에 이용된다. MPI는 인터페이스만을 정의하였으며 클러스터를 위한 대표적인 소프트웨어로는 MPICH와 LAM이 있다.

MPICH는 미국 아르곤 국립 연구소에서 개발되었으며 MPI에서 가장 필요한 기본 기능들만 추출하여 ADI(Abstract Device Interface) 인터페이스를 만들고, 이 ADI에서 더욱 기본적인 기능들만 추출하여 Channel Interface를 정의하는 계층적인 구조로 구현되었다. 이러한 계층적 구조는 Channel Interface의 적은 수의 루틴들만 컴퓨터 종속적인 루틴들을 이용하여 구현함으로서, 비교적 쉽게 다른 구조를 가진 컴퓨터로 이식될 수 있다.

LAM은 미국 오하이오 수퍼컴퓨터 센터에서 이기종간의 컴퓨터를 네트워크로 연결하여 MPI 프로그램을 개발하는 환경으로 개발되었다. 이것은 워크스테이션 클러스터 또는 네트워크 컴퓨팅의 기본 시설이 되어있는 곳에서 병렬 문제를 해결하는데 적합하도록 구현되었다. LAM은 각각의 컴퓨터에서 나노커널과 같은 유일한 구조를 갖으며, 유닉스 데몬처럼 실행된다.

메시지 전송 방식은 분산 메모리 환경에서 효율적이고 많이 쓰이는 방식이지만 프로그래밍이 복잡하고 어려운 단점이 있다. 이러한 문제를 해결하는 방법으로 공유 메모리 시스템이 있다. 그러나, 클러스터 시스템은 일반 컴퓨터를 네트워크로 연결한 구조이기 때문에 물리적인 공유 메모리를 가질 수 없다. 그래서, 물리적으로는 분산 메모리 구조를 갖지만 사용자 수준의 프로그래밍에서는 공유 메모리 형태의 구조를 제공하는 분산 공유 메모리 방식의 프로그래밍 방식을 지원하는 연구가 있다.

분산 공유 메모리는 네트워크를 통해 연결되어지는 컴퓨터 사이에 공유되는 메모리의 주소 공간을 가상적으로 만들어주는 것이다. 이러한 가상화는 복잡한 데이터 구조를 단순화하고 병렬 프로그래밍을 하는데 필요한 노력을 줄여 준다. 대표적 소프트웨어로는 Linda, CVM, Treadmarks 등이 있다.

Linda는 미국 예일대에서 개발한 병렬 프로그래밍 모델이다. Linda는 프로세스간의 통신을 조정해주는 추상적 개념의 튜플 스페이스(Tuple-space)를 제공해준다. 이것은 병렬 프로세싱에서 다루어지는 공유 메모리와 메시지 전송(분산메모리)의 두 가지 방법을 선별적으로 지원한다. 응용 프로그램에서 보는 Linda의 관점은 병렬 프로그래밍을 쉽게 해주는 확장된 프로그래밍 언어의 집합이다.

4.2 성능 감시기 및 디버거

병렬 프로그램은 보통 그 구조가 복잡하고 작업내의 각 태스크의 동작을 정확하게 파악할 수 없기 때문에, 사용자가 병렬 프로그램의 정확성과 성능을 분석하기 위해서 성능 감시기와 프로그램 작성에 있어서 오류 탐색과 수정을 위한 병렬 디버거를 필요로 한다. 대표적인 성능 감시기로는 PVM 프로그램을 추적하는 XPVM과 Pablo 등이 있으며, 병렬 디버거로는 UniView와 LCB(Lightweight Corefile Browser), TotalView 등이 있다.

XPVM은 PVM을 위한 모니터링 도구이며 PVM 콘솔의 함수들과 실시간 성능 감시, 디버거, 재실행 분석 도구, 그리고 "point-and-click" 엑세스로 PVM 명령어를 제공한다. XPVM은 "Utilization"과 "space-time"과 같은 뷰로 구성되어 있고, 원거리에 있는 태스크에 대하여 간단하게 디버깅할 수 있다.

Pablo는 일리노이 주립대학교에서 개발한 성능 감시기이며 프로그램 편집 및 실행과 프로그램 추적 및 분석 부분 등으로 이루어졌다. Pablo 구성 요소들 사이의 인터렉션은 SDDF(Self Definition Data Format) 추적 화일을 바탕으로 이루어진다. SDDF는 추적 화일 형태 중 사실상의 표준으로 자리 매김하고 있으며 이 후에 연구되는 거의 모든 성능 감시기는 SDDF를 기본 형태로 사용하고 있다. 이외에도 LAM의 성능을 감시하기 위한 XMPI와 위스콘신 대학교의 Paradyn, 조지아 공대의 PVaniM 등이 개발되고 있다.

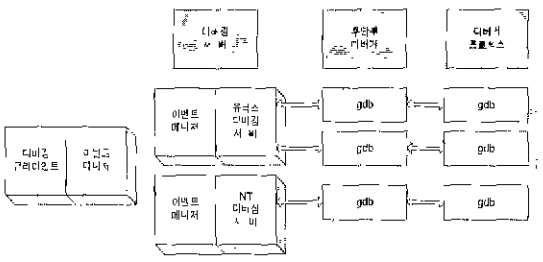

그림 3 유니뷰 시스템의 구조

UniView는 GUI를 제공하는 심볼릭 분산 디버거로서 UNIX와 NT 플랫폼을 포함한 이기종 환경에서 분산 시스템을 디버깅할 수 있다. UniView는 디버깅 클라이언트와 디버깅 서버의 두 개의 하위 시스템으로 구현되었다. 디버깅 서버는 디버깅 호스트에 각각 하나씩 존재하며 호스트의 후단부 디버거를 제어하는 역할을 한다. 그림 3은 UniView 시스템의 전체구조를 도식화한 것으로 후단부 디버거로 gdb를 사용한다.

LCB는 Ptools 컨소시엄에서 개발한 병렬 디버깅을 위한 X 윈도우 기반의 디버거이다. LCB는 병렬 프로그램에서의 동적 호출 그래프의 검사에 간단하고 알맞은 방법을 제공한다. 프로그램이 비정상적으로 종료되었을 때 발생하는 core image를 분석하여 전체적인 고수준의 뷰를 제공한다. LCB는 Lightweight corefile와 command-line lightweight corefile browser, graphical lightweight corefile browser 기능을 이용하여 병렬 프로그램의 동적인 상태를 알아내고 표현해 내는 기능을 제공한다.

TotalView는 Dolphin에서 개발한 병렬 디버거로서 HPC 플랫폼에 적용할 수 있고, C와 C++, Fortran, assembler를 위한 소스 레벨의 GUI를 제공한다. TotalView는 다중 프로세스와 한 프로세스 내의 다중 쓰레드를 관리할 수 있다. 그리고, 분산 구조를 갖는 형태의 프로그램과 원격지의 프로그램 등을 제어하여 디버깅할 수 있는 구조를 갖는다.

5. 작업 관리 소프트웨어

클러스터 컴퓨터에서는 분산된 노드들 간의 컴퓨팅 파워의 효과적 관리를 통해 작업 처리량을 극대화하고 사용자 작업 실행의 편의를 제공하기 위한 작업 관리 시스템이 필요하다. 지금까지 개발된 상업적 및 비상업적인 많은 JMS(Job Management System)들은 큐잉 배치 시스템에서 부하 균등 및 사이클 스틸링까지 폭넓게 지원하고 있다. JMS를 사용하기 위한 선택 요건으로는 다음과 같은 요구 사항을 필요로 한다.

- 사용의 편의성
- 안정성 및 결함 허용
- 순차, 병렬, 인터액티브, 배치 작업의 관리
- 로컬 및 분산 화일 시스템의 지원
- 작업 및 시스템 자원에 대한 사용자의 감시 허용
- 다양한 자원에 대한 할당 및 제한을 통한 관리

· 사용자에 의한 작업 상호간에 대한 의존성 명시 허용

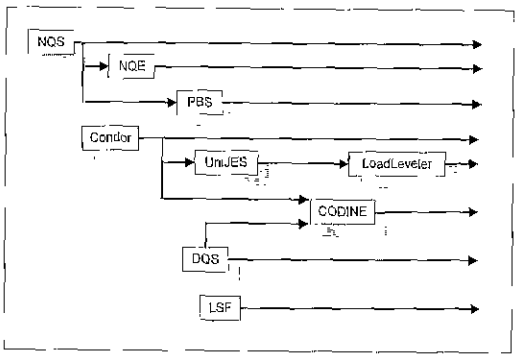

그림 4 작업 관리 시스템 개발 관계

다음 그림 4는 기존의 작업 관리 시스템들의 개발 관계를 그래프로 보여주고 있다.

NQS(Network Queueing System)는 NASA에서 배치 작업을 위한 도구로서 개발되었다. 작업에 대한 CPU time, 데이터, 스택, resident set, 화일 크기 등에 제한 가능한 다중 배치 큐를 정의할 수 있다. 초기 NQS 버전에 의해 많은 비상업적 및 상업적 버전(sterling software, Monsanto Company, CERN 등의 NQS)들이 나오게 되었으며, 이러한 것들은 NQS의 기능에 부하 균등 등 효율적인 자원 관리 기능들이 추가되었다.

PBS(Portable Batch System)는 NASA에서 고성능 컴퓨팅을 위한 프로젝트로 개발되었다. PBS의 모든 사용자 명령어는 문법과 기능에 있어서 배치 큐잉을 위한 Posix 1003.2d 표준과 모두 호환된다. 작업의 실행, 멈춤, 재실행, 감시, 삭제에 관한 명령어뿐만 아니라 시그널을 통하여 선택, 재실행, 변경이 가능하다. PBS는 일관된 사용자 영역에서 수행되며 큐와 서버에 대한 접근의 제어를 보통의 유닉스 보안 및 인증 방법을 확장에서 사용한다. 부하 균등을 위해 LSF와 유사하게 특정 노드에 관련된 자원들의 수치를 감시 데몬이 스케줄러에게 제공하게 되며 스케줄러는 이러한 로드 수치들을 고려하여 노드를 선택하여 작업을 수행할 수 있도록 한다. 작업 관리를 지원하는 클러스터 컴퓨터 중에서 많은 시스템이 PBS를 이용한 작업 처리 방법을 제공하고 있다.

LSF(Load Sharing Facility)는 로드의 공유 및 배치 큐잉 컴포넌트로 구성된 workload 관리 도구로 이루어져 있으며 Toronto의 Platform Computing에 의해 개발되었다. LSF는 시스템의 구성에 관계없이 모든 하드웨어와 소프트웨어 자원에 대한 일관되고 투명한 처리를 제공함으로서 작업들을 관리한다. LSF는 NFS, AFS 및 DCE/DFS를 지원하며 file-stage in/out도 지원하고 있다. 그리고, 배치처리와 대화식, 병렬 작업을 지원하고, 유휴 상태의 워크스테이션이나 서버의 클러스터에서 이러한 작업들을 관리한다.

Condor는 위스콘신 대학에서 오랜 시간동안 수행하는 배치 작업을 위한 워크스테이션들에서 유휴 상태에 있는 CPU 시간을 이용하기 위해 개발되었다. 주요 특징으로는 자동으로 유휴한 컴퓨터의 자원을 할당하고 위치시키며, 프로세스의 체크포인트와 이동을 제공한다. Condor는 로컬 네트워크에서 참여하고 있는 워크스테이션의 모든 행동을 모니터링 할 수 있다. 이러한 모든 기능은 UNIX 커널의 특별한 수정 없이 제공된다. 위스콘신 대학은 현재 Condor를 지속적으로 개발 및 지원하고 있으며 많은 대학의 클러스터 관련 연구에서 이용되고 있다.

DQS(Distributed Queueing System)는 Florida 주립대학의 SCRI에서 개발된 UNIX 기반의 큐잉 시스템이다. 전체 시스템 자원들을 각각의 사용자들이 공평하게 접근할 수 있도록 하고 가장 최적의 방법으로 자원들을 이용할 목적으로 개발되었다. DQS는 CPU time, Wall-clock time, 데이터, 스택 등의 하드 및 소프트 제한을 줄 수 있는 큐를 정의하고 있다. 현재 DQS의 확장된 버전은 병렬 작업의 모든 프로세스들에 대해 추적할 수 있으며 병렬 make 유틸리티를 제공한다.

다음의 표 1은 클러스터를 사용하는데 필요한 기본적인 소프트웨어 도구를 구할 수 있는 사이트이다.

6. 결 론

이상에서 우리는 클러스터 컴퓨터에서 필요한 소프트웨어를 살펴보았다 클러스터 시스템의 사용 목적에 따라 별도의 소프트웨어가 필요할 수

표 1 클러스터 소프트웨어 도구 사이트

프로그래밍 도구	
PVM	www epm.ornl gov/pvm/
MPICH	www mcs.anl gov/mpi/mpich/
LAM	www.mpi nd.edu/lam/
클러스터 관리 도구	
SMILE	smile cpe.ku.ac th/~ppe/
M3C	www.epm ornl gov/torc/
MAT	www.ee rverson ca 8080/~sblack/mat/
VACM	www valinux.com/projects/vacm/
성능 감시기 및 디버기	
XPVM	www.netlib org/utk/icl/xpvm/
Pablo	www-pablo cs uiuc edu/
TotalView	www.etnus com/
LCB	www ptools.org/projects/lcb/
작업 관리 도구	
LSF	www platform com/
NQS	www gnqs.org/
PBS	pbs mrj.com/
Condor	www.cs wisc.edu/condor/
DQS	www scri fsu edu/pasko/dqs html

도 있다. 위에서 살펴본 예들은 주로 연구 목적으로 개발된 소프트웨어들이기 때문에 일반 사용자가 곧바로 사용하기 어려운 경우도 있을 수 있다 그러나 연구 개발 경험이 있는 대부분의 사용자들은 큰 어려움 없이 사용할 수 있다.

근래에는 클러스터 시스템의 개발이 가속화되어 상업적으로도 이용 가능한 수준의 소프트웨어들이 발표되고 있다. 또한 쉽게 구할 수 있고 확장(사용자 입맛에 맞게 변경할 수 있는)이 자유로운 리눅스를 이용하여 클러스터를 사용/개발하고 있고, 많은 소프트웨어들이 계속 개발되고 있다. Beowulf project에는 세계의 수많은 곳에서 연구 개발한 최신의 소프트웨어, 구현 사례 등을 발표하고 있어 마음만 먹는다면 언제든지 자신의 환경에 맞는 클러스터 시스템을 만들 수 있을 것이다.

클러스터 시스템은 하드웨어의 발전으로 슈퍼컴퓨터와 비슷한 성능을 낼 수 있는 하드웨어적 성능을 갖추고 있어서 클러스터 시스템에 대한 연구 개발과 사용이 빠르게 늘고 있다. 클러스터 시스템의 사용이 늘어갈수록 클러스터 시스템을 효율적으로 운영할 수 있는 소프트웨어 도구의 역할이 더욱 증대될 것이다.

사용자의 입장에서 좀 더 사용하기 쉽고 강력한 기능을 갖는 운영 도구의 개발을 필요로 한다. 그리고, 고속 통신을 지원하는 네트워크, 자원을 통합 관리하는 클러스터를 위한 미들웨어, 입출력 속도를 향상시킬 수 있는 병렬 I/O, 사용자 중심의 간편한 프로그래밍 환경, 저장 장치, 소프트웨어 공학 등의 기술 지원들이 앞으로 더욱 연구되어야 할 것이다.

참고문헌

[1] Mark Baker, Rajkumar Buyya, High Performance Cluster Computing: Architectures and Systems, Prentice Hall, 1999.

[2] Luis M. Silva, Rajkumar Buyya, High Performance Cluster Computing: Programming and Applications, Prentice Hall, 1999.

[3] IEEE CS Task Force on Cluster Computing, at http://www.dgs.monash.edu.au/~rajkumar/tfcc/

[4] Gory F. Pfister, In search of Clusters, 2nd Edition, Prentice Hall, 1998.

[5] 최재영, 황석찬, "클러스터 컴퓨터의 개요," 병렬처리시스템 연구회지, 제10권 제2호, 1999.

[6] James P. Jones, Cristy Brickell, "Second Evaluation of Job Queuing/Scheduling Software: Phase 1 Report," NAS Technical Report NAS-97-013, June, 1997.

[7] Mary Papakhian, "Comparing Job-Management Systems: The User's Perspecive," IEEE Computationial Science & Engineering April-June 1998.

[8] T. Anderson, D. Culler, D. Patterson, "A Case for Network of Workstation," IEEE Micro, Feb. 1995.

[9] C. E Mcdoucll, D. P. Helmbold "Debugging Concurrent Programs," ACM Computing Surveys, Vol. 21, No. 4. 1989

[10] L. Beguelin, J. Dongarra, A Geist, Sunderam, "Heterogeneous Network Computing," In Sixth SIAM Conference on Parallel Processing, SIAM, 1993.

[11] Mark Baker, "Custer Computing Reveiw," NPAC Technical Report SCCS-748, Nov. 16 1995.

[12] MPI Forum, at http://www.mpi-forum.org/docs/docs.html

[13] The beowulf project at CACR, at http://www.cacr.caltech.edu/beowulf/index.html

[14] Beowulf class Project, at http://www.beowulf.org/

최 재 영

1984 서울대학교 제어계측공학과 학사
1986 미국 남가주대학교 전기공학과 석사 (컴퓨터 공학)
1991 미국 코넬대학교 전기공학부 박사 (컴퓨터 공학)
1992.1~1994.2 미국 오크리지 연구소 연구원
1994.3~1995.2 미국 네바시 주립대학교 연구교수
1995.3~현재 숭실대학교 컴퓨터학부 조교수
관심분야 병렬/분산 처리, 클러스터시스템, 시스템 소프트웨어
E-mail choi@comp.soongsil.ac.kr

이 준 호

1987 서울대학교 컴퓨터공학과(학사)
1989 한국과학기술원 전산학과(석사)
1993 한국과학기술원 전산학과(박사)
1993~1994 한국과학기술원 인공지능연구센터 연구원
1994~1995 코넬대학교 전산학과 방문연구원
1997~현재 숭실대학교 컴퓨터학부 조교수
관심분야 정보검색, 클러스터시스템, 데이터베이스
E-mail joonho@comp.soongsil.ac.kr

황 석 찬

1996 청주대학교 건지계산학과 학사
1998 숭실대학교 컴퓨터학과 석사
1998~현재 숭실대학교 컴퓨터학과 박사과정
관심분야 병렬/분산 처리, 고성능 컴퓨팅, 클러스터링, 시스템 소프트웨어
E-mail:schwang@ss.soongsil.ac.kr