분산시스템 OS의 향방

고속 컴퓨터 네트워크 기술이 발전됨에 따라 물리적으로 분산된 컴퓨터 시스템들도 하나의 컴퓨터 시스템 처럼 통제가 가능하게 되었다. 이를 가능케하는 분산시스템이 갖추어야 할 조건, 구현상의 문제점 그리고 해결책에 대해 살펴본다. -편집자주-

고속 컴퓨터 네트워크 기술의 발전으로 등장

컴퓨터와 통신 관련 기술이 발전함에 따라 고속 컴퓨터 네트워크 기술은 눈부신 발전을 거듭하게 되었다. 하지만 기존의 중앙집중형 컴퓨터 시스템에서 제공되던 온라인서비스나 대화식서비스는 장시간 또는 예측하기 어려운 응답시간을 갖거나 제한된 사용자 인터페이스만을 제공해 사용자의 욕구를 충족시키지 못했다. 이에 관련 데이터를 빠르게 액서스할 수 있는 컴퓨터시스템에 대한 관심이 고조되기 시작했다. 이러한 사용자의 새로운 욕구에 의해 출현한 것이 분산시스템이다.

물리적 분산 불구 단일 시스템으로 운용

분산시스템은 중앙집중형 컴퓨터시스템으로 보이는 여러개의 독립적인 CPU들로 구성된 시스템이다. 일반적으로 네트워크를 구성하는 각각의 컴퓨터들이 동일 네트워크상에 있는 공유자원들을 이용할 수 있도록 기존의 중앙집중형 컴퓨터시스템에서 실행하던 기능들을 제공하며 물리적으로는 분산되지만 기능면에서는 마치 단일적이고 통합된 컴퓨터(Single and Integrated Computer)에서 제공하는 것처럼 컴퓨터를 사용할 수 있다.

분산시스템이 가지는 특성에는 분리(Seperation)와 투명성(Transparency)이 있다. 분리특성은 컴퓨터 통신을 이용해 시스템의 관리와 통합을 구현하는 것이다. 이에 따라 프로그램의 병행수행을 보장하고 한 시스템의 파괴가 네트워크 내의 또 다른 시스템들에 영향을 주지 않는다. 이 특성에 따라 시스템의 보안성, 구성요소 등의 점진적 확장, 축소 또는 하나의 시스템을 전체시스템으로 부터 분리 등의 작업이 가능하다. 또 다른 특성인 투명성은 크게 ▲접근투명성 ▲위치투명성 ▲동시투명성 ▲반복투명성 ▲실패투명성 ▲이주투명성 ▲수행투명성 ▲규모투명성 등으로 정의된다. 접근투명성은 동

일한 작동으로 로컬 및 리모트 파일과 객체에 접근한다. 위치투명성은 각 객체에 대한 위치정보 없이 객체 접근이 가능하다. 동시투명성은 사용자와 응용 프로그램이 공유 데이터상에서 서로 간섭없이 일정하게 동작한다. 반복투명성은 반복된 동일한 작업의 결과도 동일하다. 실패투명성은 한 시스템의 작업실패는 다른 시스템의 작업과 무관하다. 이중투명성은 객체의 움직임이 다른 응용 프로그램이나 사용자에게 영향을 주지 않는다. 수행투명성은 수행기능을 개선할 수 있도록 시스템을 재구성한다. 규모투명성은 시스템 구조 또는 응용 알고리즘의 변화없이 규모면에서 시스템과 응용을 확장한다.

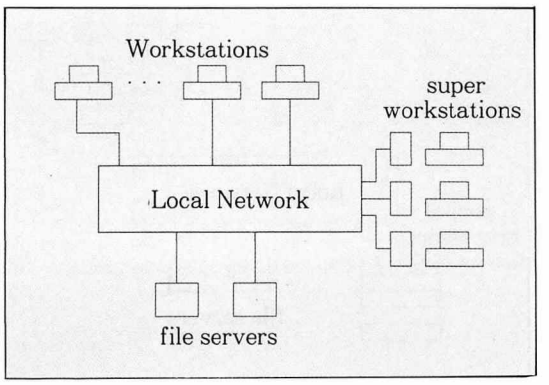
분산시스템은 중앙집중형 컴퓨터시스템에 비해 몇가지 장점을 가지고 있다. 첫째, 시분할시스템에 비해 신뢰성, 유연성 및 수행능력이 좋다. 둘째, 사용자의 작업이 다양하고 대화적이며 막대한 양의 용량을 요구할수록 효과적이다. 셋째, 시스템의 확장 또는 축소 요구에 유연하다. 넷째, 네트워크상의 모든 자원을 공유할 수 있다. 다섯째, 저장정보의 신뢰성과 빠른 접근이 가능하다. 여섯째, 한 작업이 중단되더라도 다른 작업은 계속될 수 있다. 반면 단점도 가지고 있다. 첫째, 투명성 확보를 위한 추가적인 프로토콜이 필요하다. 둘째, 네트워크의 수행능력과 신뢰도에 의존도가 높다. 셋째, 네트워크 상태에 대한 전체적인 정보의 유지 관리가 필수적이다. 넷째, 시스템이 모두 개방돼 시스템 보안측면에서 매우 불리하다.

네트워크 연결형태에 따라 유형과 시스템 모델 결정

분산시스템은 어떤 네트워크에 연결되어 있는가에 따라 세가지 유형으로 범위가 결정된다. 첫째, Weakly-coupled System은 WAN에 연결된 시스템들의 집합을 의미하며 광역에 연결된 모든 시스템들을 포함한다. 둘째, Strongly-coupled System은 LAN에 연결된 시스템들을 의미하며 특정의 제한된 구역내에서 통합된 시스템들만을 일컫는다. 셋째, Very Strongly-coupled System은 한 시스템내에 다양한 여러개의 프로세서들로 구성된 컴퓨터를 의미하며 단순히 시스템내 프로세서간의 연결외에는 자원공유 등의 의미에서는 매우 좁은 범위이다.

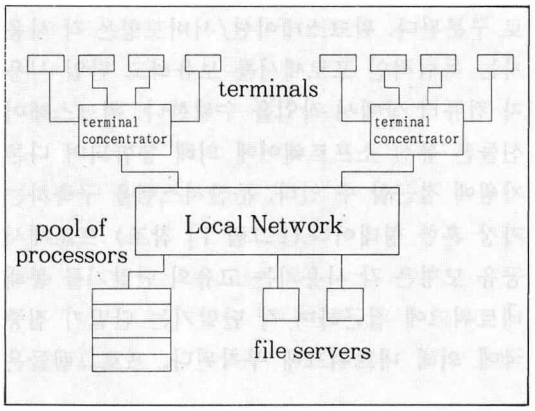
또한 시스템 모델은 네트워크에 시스템들이 어떻게 연결되어 있는가에 따라 세가지 모형으로 구분된다. 워크스테이션/서버모형은 각 사용자는 독립적인 프로세서를 보유하고 단일 사용자 컴퓨터 상에서 작업을 수행한다. 워크스테이션들은 통신 소프트웨어에 의해 통합되며 다른 자원에 접근할 수 있다. 분산시스템을 구축하는 가장 흔한 형태이다.([그림 1] 참조) 프로세서 공유 모형은 각 사용자는 고유의 단말기를 통해 네트워크에 접근하며 각 단말기는 단말기 집중국에 의해 네트워크에 부착된다. 프로그램들은

[그림 1] 워크스테이션/ 서버모형(Workstation /Server Model)

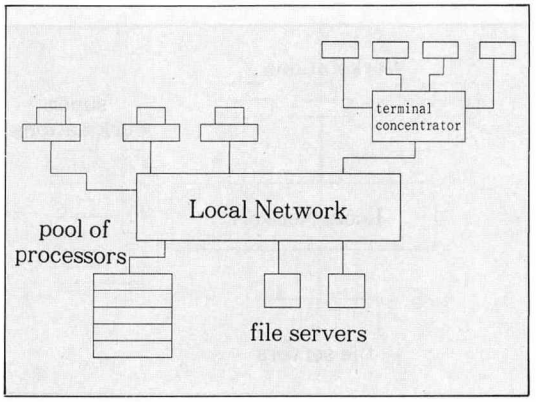


프로세서 공유영역(processor pool)에 있는 프로세서를 이용하여 작업을 수행한다. 워크스테이션/서버모델에 비해 응용 프로그램과 터미널간의 통신속도가 느려 그래픽과 같은 고성능의 작업이 요구될 때 부적당하다. 그러나 자원이용 효용도, 유연성, 호환성 등에서 효과적이다.([그림 2] 참조) 하이브리드 모형은 워크스테이션/서버 모형과 프로세서 공유모형을 혼합한 형태로서 이 응용은 두 모형의 장점들을 제공한다. 따라서 고성능을 원하는 경우에는 워크스테이션 상에서 수행하고 하나 이상의 프로세서가 필요할 경우에는 다양한 형태의 고유영역 프로세서를 이용하여 수행할 수 있다. 사용자 요구에 적응하기 쉽고 작업을 병행할 수 있으나 서로 상이한 운영체제 구현이나 터미널을 통해 각 시스템에 접근할 때의 차이점들을 극복해야 하므로 사용이 어렵고 시스템 보안이 불리하다.([그림 3] 참조)

[그림 2] 프로세서 공유 모형(Processor Pool Model)



[그림 3] 하이브리드 모형(Hybrid Model)



네트워크와 관련된 통신이 주요 이슈

분산시스템에 관련된 주요이슈를 분산 프로세서 부문과 자원관리 부문으로 나누어 살펴보면 다음과 같다. 분산 프로세서 부문에서의 이슈는 원격 프로세서간 통신, 프로세서간 동기화, 프로세서관리 및 명명화 등이다. 자원관리 부분에서의 이슈는 자원할당, 비밀보호와 보안, 제공되는 서비스의 형태 또는 공급 등이다. 이중 가장 중요한 이슈는 네트워크와 관련된 통신이다. 이와 관련돼 NOS(Network Operation System)가 등장하게 됐다.

NOS는 원격자원을 접근할 수 있는 모듈을 포함하는 네트워크에 연결된 컴퓨터 운영체제들의 집합이다. DOS(Distributed Operating System)와 비교해 NOS는 분산자원들로 구성된 시스템의 보안, 신뢰도, 간명화, 유연성들을 위해 등장했다는 점과 프로세서 관리, 통신관리, 장치관리, 기억장소 또는 메모리의 관리 등을 위한 응용 프로그램을 제공한다는 점이 비슷하다. 그러나 NOS는 DOS와는 달리 각 컴퓨터가 별도의 운영체제를 갖는다. 이에 따라 NOS는 주로 자신의 컴퓨터를 이용해 작업하기 때문에 수행될 CPU는 전체 네트워크 운용시스템에 의해 결정되지 않고 사용자에 의해 결정된다. 또한 파일

을 전송하는 경우 전체 네트워크 운영시스템이 아닌 사용자에 의존하기 때문에 파일위치 관리도 사용자가 한다. 또한 NOS는 결함 허용성(faulttolerance)이 매우 낮거나 없다.

다양한 방향으로 추진된 NOS 개발

결국 DOS와 NOS의 차이는 지역 또는 전체의 자원을 보는 방법과 관리하는 방법에 따라 정의된다. DOS의 가장 큰 관심사는 네트워크 운영시스템이다. 그동안 이를 해결하기 위한 대규모 프로젝트들이 수행되었다.

첫번째가 1970년경에 개발된 최초의 NOS로서 NSWs(National Software Works)프로젝트이다. 이 프로젝트는 여러 종류의 CPU 형태와 이종 OS를 통합하는 것이었는데, 부분적으로 구현된 시스템이 수용하기 어려운 수행능력을 보여 결국 파기되었다.

또 다른 시도는 Simple Network Operating System으로 Goscinski 등에 의해 제시됐다. 이 프로젝트 역시 서로 다른 운영체제와 서로 다른 컴퓨터 자원에 접근할 수 있는 NOS를 개발하는 것이었다. 이 프로젝트에서는 RAS(Remote Access System)라는 접근방식이 적용됐다.

이밖에 유닉스 베이스 운용시스템들이 있는데 이는 Newcastle 접속과 같이 C의 라이브러리를 확장하여 구현하는 것이다. 이 개념은 이론적 측면이나 구현면에서 초보단계에 있으며 완전하게 정립이 되지 않는 상태이다. 그러나 분산시스템에서의 구역과 지역간의자원을 어떻게 공유하느냐와 네트워크 관리를 어떻게 할 것인가는 매우 중요하다.

효율적 네트워크 관리는 통신 프로토콜 선택이 관건

지금까지 분산시스템이 갖추어야 할 조건과 장단점을 통해 현재 분산시스템이 안고 있는 문제점이 무엇인지 알 수 있었다. 결국 효율적인 네트워크 관리를 위해 통신 프로토콜을 어떻게 선택하느냐가 주요한 관건임을 확인할 수 있었다.

일반적으로 통신 프로토콜은 GPT(General-Purpose Trotocols)라 불리우는 광범위한 응용을 지원하는 방식과 SPT(Sepecial-Purpose Trotocols)라 불리우는 일종의 프리미티브로서 통신을 약속해 송수신하는 방식이 있다.

두가지 방식 모두 장·단점을 가지고 있지만 상호 보완적이다. 분산시스템의 경우 출발은 GPT 방식으로 부터 시작됐지만 현재에는 실행가능성이나 응용 가능성 측면에서 SPT가 더 많이 고려되고 있다. 그 이유는 SPT 방식이 특정 목적을 위해 구현됨에 따라 비효율적인 오버헤드들을 줄여줄 수 있기 때문이다. 하지만 응용범위의 한계로 GPT를 선호하게 될 것으로 보인다. 이는 분산된 많은 자원의 공유라는 측면에서 고려될 경우 또 다른 양상의 운영체제 개발향방이 이루어질 수도 있겠지만, SPT 방식으로 네트워크 관리 프로토콜을 구현하면서 범위와 분야를 넓혀 GPT를 이용한 광범위한 응용을 가능하게 하는 통신 프로토콜 연구와 구현으로 이어질 것으로 보이기 때문이다.