1. **분산 운영체제 개관**

분산 시스템

크기나 성능이 다를 수 있는 여러 대의 컴퓨터가 네트워크로 연결되어 이루어지는 시스템 학습목표

* 1. **분산 시스템의 구조와 특징을 이해한다.**
  2. **분산 시스템에서 파일, 메모리 등을 공유하는 방법을 이해한다.**
  3. **분산 시스템에서 원격 프로시저 호출을 이해한다. 주요 용어**

분산 시스템 분산 파일 시스템 분산 메모리 원격 호출

**13.1.1. 분산 시스템 [203]**

배경

빅데이터 응용이나 인공지능의 경우에 거대한 데이터에 대해 복잡한 연산을 오랫동안 수행 강력한 프로세서(CPU)

여러 개의 프로세서를 연결해 작업을 분담하는 방식

많은 경우 문제를 나누어 동시에 풀 수 있는 '병렬성'을 이용할 수 있기 때문에 적절한 접근방법이다. 병렬로 문제를 풀 수 있는 컴퓨터를 만드는 방법

강결합 시스템

한 대의 컴퓨터에 여러 개의 프로세서를 넣어서 모든 프로세서가 클럭과 메모리를 공유하는 시스템 약결합 시스템

**복수의 컴퓨터가 각각의 프로세서를 가지고 네트워크를 통해 결합 로컬(local)과 원격(remote)의 구분이 생김**

로컬 자원

각 프로세서 입장에서 자신이 속한 컴퓨터에 속하는 자원 원격 자원

(각 프로세서 입장에서) 다른 컴퓨터에 속해서 네트워크를 통해야 이용할 수 있는 자원

효율적인 분산 시스템의 설계를 위해서는 원격 자원도 로컬 자원처럼 쉽게 이용할 수 있는 방법을 운영체제가 제공 분산 시스템에 속하는 프로세서

강결합 시스템과는 달리 크기와 기능이 다양한 프로세서가 결합되어 한 시스템을 구축할 수 있다. 사이트

기계의 위치를 나타냄 호스트

사이트에 있는 특정 시스템 분산 시스템의 개략적 구축 모델

서버 / 클라이언트 서버 : 자원을 제공

클라이언트 : 자원을 이용 분산 시스템의 목표

자원을 공유하는 것이 마치 한 대의 컴퓨터의 자원을 이용하는 것처럼 구분하지 않고 효율적이며 편리하게 이용할 수 있는 환 경을 제공하는 것

분산 시스템의 장점 자원 공유

**예1) 원격 프린터를 공유 예2) 1,000대의 컴퓨터 256GB -> 256TB 각 컴퓨터의 자원을 연결된 다른 컴퓨터가 공유하여 사 용**

성능 향상

1) 다른 작업과 상호작용이 없거나 적어서 따로 분리할 수 잇고, 2) 동시에 많이 처리하는 병렬성이 존재한다면 여러 대의 프로세서에 작업을 분할하여 병렬적으로 동시에 수행하는 방식으로 성능을 향상시킬 수 있다.

신뢰성 향상

FAILOVER 하나의 프로세서 또는 자원에 문제가 생긴다 해도 이를 대신할 수 있는 다른 프로세서 자원이 있다면 역할을 대 신시켜서 작업을 진행

통신의 편리성

분산 시스템은 단일 시스템 내부에서 동작하는 서비스와 같은 서비스 제공 예) A와 B는 서로 다른 컴퓨터이지만 파일을 옮길 수 잇다.

A와 B가 서로 정보를 주고 받을 방법이 있어야 하며, "원격 프로시저 호출" 등의 방법으로 이를 달성한다.

* + 1. **분산 시스템 (2) [205]**

분산 시스템의 다양한 형태 개념

**노드: 사이트**

엣지 : 두 노드가 물리적으로 통신 링크에 의해 연결되어 있음 분산 시스템 구축 판단의 근거

망 구축 비용

통신 링크를 통해 사이트들을 물리적으로 연결해서 분산 시스템을 구축하는 데 드는 비용 통신비용

사이트 A에서 사이트 B로 메시지를 보내는 데 필요한 비용 신뢰성

일부 링크나 사이트가 고장났을 때, 시스템이 얼마나 정상적으로 동작할 수 잇는가 측정하는 정도 링크를 많이 연결할수록

망 구축비용은 늘어남 통신비용과 신뢰성이 높아짐

여러 가지 네트워크의 구성 형태 완전연결 네트워크

모든 사이트는 다른 사이트와 직접 연결 특징

단점

망 구축비용은 가장 크게 들게 된다.

노드가 하나 추가될 때마다 이미 네트워크에 있는 노드의 개수만큼의 엣지가 필요하기 때문에 노드의 개수가 많은 경우 적합하지 못하다

장점

통신 비용은 최소 : 어떤 두 사이트를 고르더라도 둘 사이가 연결되어 있음

신뢰성 높음 : 한 링크가 고장나더라도, 부분연결 네트워크

직접 연결되지 않은 노드의 쌍이 존재 장점

다른 사이트와 링크를 이용하여 통신 가능

망 구축비용은 (완전연결 네트워크에 비해)저렴 단점

연결되지 않은 노드끼리는 다른 노드와 링크를 통해 메시지를 전달

만약 이 노드나 링크가 고장나면, 두 노드는 연결 주고 받지 못해 신뢰성이 떨어질 수 있음 트리 구조 네트워크

어떤 두 노드도 연결되도록 할 때 링크가 가장 적은 형태 특징

N개의 노드가 있을 때 N-1개의 링크가 존재함.

직접 연결된 두 노드 외에는 반드시 다른 노드를 거쳐서 메시지를 주고 받아야 함 노드와 링크가 고장날 경우 전체 시스템이 둘 이상으로 분리될 수 있음

스타형 네트워크

하나의 노드가 중심을 이루고, 다른 노드들은 모두 중심 노드에 연결된 형태 특징

N개의 노드가 있을 때 N-1개의 링크가 필요

어떤 두 노드를 고르더라도 최대 2개의 링크를 거쳐 메시지를 주고받을 수 있음 중심 노드가 고장날 경우 전체 네트워크는 연결이 끊어지게 됨

링형 네트워크

전체 노드가 큰 고리를 이루는 형태 특징

N개의 노드가 있을 때 N개의 링크가 필요 두 노드 사이의 링크의 개수

**최소 1, 최대 N/2**

이유 모ㅗ름

어떤 하나의 노드나 링크가 고장난다 해도 연결이 끊어지지 않음 버스형 네트워크

모든 노드가 버스라고 부르는 빠른 링크에 연결된 형태 특징

한 노드가 버스에 메시지를 보내면 모든 노드가 이를 읽을 수 있음 버스가 고장나면 전체 네트워크가 문제 발생

* + 1. **분산 시스템 (3) [207]**

네트워크의 여러 가지 연결방법

**근거리 통신망(Local-Area Network: LAN)**

예시

일반적인 가정에서, 집으로 인터넷 회선이 들어오면 공유기에 연결하고, 유선 및 무선으로 TV, 컴퓨터, 태블릿, 전화기 등 에 연결하여 사용

**학교, 회사, 병원 등과 같은 건물**

**원거리 통신망(Wide-Area Network: WAN)**

예시

전국에 가설된 인터넷 회선 넓은 지역을 잇는 네트워크

**LAN과 WAN의 차이점 이더넷**

LAN을 구성하는 주요 기술

가격이 저렴하고, 에러가 적으며, 매우 빠른 전송속도를 제공

**but 거리 제한이 있음**

근접한 공간에 있는 서버, 워크스테이션, 랩톱, 프린터, 게이트웨이 등을 연결

# WAN

광케이블, 위성 네트워크 등으로 넓은 영역을 지원하는 것이 주된 목적 네트워크들의 네트워크

참여 노드가 LAN보다 훨씬 많기 때문에 많은 데이터가 공유되고 충돌할 수 있음.

전송 효율 낮음: 많은 데이터가 공유되고 충돌, 네트워크 자체 속도뿐 아니라 전송과정에 혼잡과 지연이 발생

* + 1. **분산 운영체제 [209]**

분산운영체제 원칙

사용자가 원격 자원을 로컬 자원을 사용하는 것처럼 쉽게 사용

둘 사이의 구별 없이 투명성(transparency)를 제공할 수 있어야 함 사용자 입장에서는 어떤 자원이 로컬인지 원격인지 구별할 필요X

원격 자원에 대한 접근에 필요한 일을 분산 운영체제가 대신 해 주는 것

**(분산) 운영체제에 적용할 수 있는 기본적인 방법 데이터 이주(data migration)**

원격 데이터를 로컬로 전송해 와서 사용

**필요한 만큼의 데이터를 우선 가져오고, 데이터를 처리하는 동안 백그라운드에서 계속 데이터를 가져올 수 있음 계산 이주(computation migration)**

원격 프로시저 호출(RPC)을 통해 계산을 원격지에서 처리한 후 결과를 전송받는 방식 데이터 이주를 고려하기에는 데이터 양이 너무 많은 경우 사용

**프로세스 이주(process migration)**

프로세스 자체를 원격지로 이주

프로세스의 코드, 실행 정보 등도 네트워크로 전송이 가능하기 때문에 사용 가능한 방법

한 코어 또는 프로세서에 할당된 프로세스가 다른 코어 또는 프로세서로 이동ㅎ할 수 있음 프로세스 이주의 주된 목적

작업량 분산 및 목적에 부합하는 하드웨어/소프트웨어에서 프로세스를 실행시킴으로써 성능을 향상

* 1. **분산 파일 시스템 [210]**

**분산 파일 시스템(Distributed File System : DFS)**

네트워크를 통해 물리적으로 분리되어 있는 원격 파일 시스템들을 하나의 파일 시스템처럼 이용할 수 있게 해 주는 것 클라이언트가 원격 파일을 로컬 파일처럼 사용할 수 있게 해 준다.

사용자는 자신이 사용하는 파일이 로컬인지 원격인지 구별할 필요X 다른 파일 시스템에 저장된 같은 이름을 가진 파일들을 구별하는 방법

파일에 호스트 이름과 그 호스트에서의 로컬 이름을 합쳐 새로운 이름을 주는 방법 원격 디렉터리를 로컬 디렉터리에 마운트(mount)하는 방식

**\* 마운트: mount는 파일 시스템을 특정 디렉토리에 연결(mounting) 하는 명령어 로컬**

원격

**확장하여 원격 파일 시스템을 디렉터리와 연결할 수 있음 예) NFS(Network File System)**

**IP 주소가 10.10.10.10인 호스트의 /backups 디렉터리를 /var/backups 디렉터리를 연결**

**사용자 입장에서 두 디렉터리(/usr/local , /var/backups) 동등하게 보이지만 전자는 로컬이고 후자는 원격이다. 원격 파일을 처리하는 과정**

절차

* + 1. **클라이언트가 파일 서버에 요청**
    2. **요청은 RPC를 통해 전달됨**
    3. **파일 서버는 클라이언트를 대신하여 디스크에 접근**
    4. **그 결과를 네트워크를 통해 전달 장점**

네트워크를 통해 파일을 접근할 수 있게 해줌 한계

파일에 대한 요청이 커지면 네트워크를 자주 이용해야 하므로 효율성에 문제가 생김 캐시를 이용하여 네트워크 사용량을 줄이고 전체 시스템의 성능을 높일 수 있음

캐시를 이용한 분산 파일 시스템의 구현 예 캐시의 기본 아이디어

국부성: 1) 한 번 읽은 데이터는 가까운 장래에 다시 사용될 가능성이 크며, 2) 데이터의 어떤 부분을 읽었다면 그 이웃한 부분 도 가까운 장래에 다시 사용될 가능성이 큼

캐시

클라이언트가 파일을 요청하면, 서버에서 전송된 데이터를 한 번 사용하고 버리는 것이 아니라 클라이언트의 캐시에 데이터 의 복사본이 저장됨

다음에 클라이언트가 해당 데이터를 다시 요청한다면, 파일 서버에 다시 요청하는 대신 자신이 가지고 있는 복사본을 이용한 다.

네트워크로 연결된 클라잉너트뿐 아니라, 서버와 디스크에서도 사용할 수 있음 캐시관련 고려해야 할 점

캐시 공간이 가득 찼을 때 어떤 것을 버리고 새 것을 넣을 것인가? 캐시 일관성

캐시에 저장된 데이터와 실제 데이터가 다를 수 있음

캐시에 복사된 데이터와 원래 데이터 사이의 불일치를 해소하고 일관성을 갖게 하는 문제

* 1. **분산 메모리 [213]**

원격 시스템의 메모리를 효율적으로 접근하는 방법 원격 메모리

분산 공유 메모리

* + 1. **원격 메모리**

로컬 메모리는 로컬 메모리와 보조기억장치를 합친 가상 메모리 형태로 접근 원격 메모리는 원격 메모리 API를 이용하여 클라이언트/서버 형태로 구성

방법

프로세스는 전역 주소공간에 매핑된 로컬 주소공간의 이름을 통해 원격 메모리를 참조한다. 예시

**원격 메모리에 전송 계층 주소(net#, host#, port#)가 할당되면 프로세스는 해당 서버 주소상의 "블록"과 "오프셋"**

을 참조할 수 있어야 한다.

컴파일 시간, 적재시간, 런타임 시 로컬 주소공간과 원격 메모리의 바인딩을 하도록 시스템을 설계

* + - 1. **컴파일 시간에 바인딩**

두 주소공간에 대한 정보를 정확히 알고 있어야 하는데 거의 불가능

* + - 1. **적재시간 바인딩**

프로그램이 링크되고 적재될 때 사용자가 원격 메모리의 위치를 제공함 링커에서 외부 참조를 해결할 때 링크 에디터에 의해 네트워크 위치가 정의됨

* + - 1. **런타임 바인딩**

프로그램은 메모리가 처음 참조될 때, 원격 메모리의 위치를 알아내고 이를 로컬 주소공간에 바인딩하기 위해 이름 서버를 사용한다.

가장 유연하며, 그 중 가장 많이 사용됨

* + 1. **분산 공유 메모리**

**분산 공유 메모리(Distributed Shared Memory : DSM)**

물리적으로 분리된 메모리를 하나의 주소공간을 통해 접근할 수 있게 해줌 특징

분산 공유 메모리에서 가상 메모리 관리자의 역할 메인 메모리와 보조기억장치 사이의 페이지 관리 네트워크로 연결된 원격 메모리 서버도 관리

가상 메모리 참조

(참조가 발생하기 전에 다음 수행) 가상주소에 실제 로컬 메모리, 보조기억장치, 원격 메모리의 물리주소가 대응됨

장점

노드의 개수가 늘어나더라도 잘 확장됨

실제로 메모리를 공유하기 위해 해야 할 일들을 프로그래머가 신경 쓸 필요X 복잡하고 큰 데이터를 처리하는 데 유리

멀티프로세서 시스템에 비해 저렴 큰 가상 메모리 공간을 제공함

단점

분산되지 않은 공유 메모리에 비해 접근속도가 느림

공유 메모리에 저장된 데이터에 대해 둘 이상의 접근이 발생할 때 추가적인 보호 메커니즘이 필요함 성능이 떨어질 수 있음

프로그래머가 분산 공유 메모리를 직접 제어하는 것이 쉽지 않음

# NUMA(Non-Uniform Memory Access)

각 프로세서가 로컬 메모리를 갖는 분산 공유 메모리 구조 특징

가상 메모리에서 통일된 주소공간에 대응되어 있다 하더라도 실제 데이터를 접근할 때 로컬 메모리와 네트워크로 연결된 메모리는 속도에 차이가 생길 수 밖에 없음

로컬 메모리는 매우 빠르지만, 현대의 CPU는 로컬 메모리의 접근속도에 비해서도 데이터를 매우 빠르게 처리하기 때문에

**"메모리"와 "CPU" 사이에 "캐시 메모리"를 두고 있음**

캐시 메모리는 원격 메모리에 대해서도 사용될 수 있음 프로그래머 입장에서 분산 공유 메모리의 장점

로컬 메모리와 동일한 방법으로 사용할 수 있음

네트워크의 여러 프로세스에서 복잡한 형태의 데이터를 공유할 수 있음 메시지 패싱의 경우

데이터를 문자열의 형태로 변환하여 전달 후 다시 원래 형태의 데이터로 복원해야 하고, 명시적으로 메시지를 주고받아야 하는 불편한 점이 있음

* 1. **원격 프로시저 호출 [217]**

**원격 프로시저 호출(Remote Procedure Call: RPC)**

프로세스가 다른 주소공간(네트워크로 연결된 다른 컴퓨터)에 있는 프로시저를 실행시키는 것

운영체제는 프로그래머가 RPC를 쉽게 이용할 수 있도록 해당 프로시저가 마치 같은 컴퓨터에 있는 것처럼 이용할 수 있게 하고, 이를 위해 필요한 일을 수행

* + 1. **원격 프로시저 호출의 동작 전통적인 프로시저의 호출**

매개변수를 스택에 넣고, 호출할 프로시저가 있는 위치로 점프

RPC의 경우 호출할 프로시저는 다른 주소공간에 있기 때문에 이 방법을 이용할 수 없음.

RPC 절차: 분산 운영체제는 RPC를 일반적인 함수 호출과 구별하지 않고 처리할 수 있도록 하려고 함

1. **RPC를 사용하려는 클라이언트는 같은 주소공간에 있는 프로시저(스텁 루틴)를 일단 호출한다. 스텁(stub) 루틴**

중간 대리인

1. **전달받은 매개변수를 메시지로 포장하여 네트워크를 이용하여 대기하고 있는 특정 서버에 있는 프로세스에 전달**
2. **스텁 루틴은 답을 기다리게 되며, 메시지를 받은 프로세스는 이를 해당 프로시저에 전달하여 실행시킨다.**
3. **그 실행 결과는 다시 메시지로 포장되어 네트워크를 통해 기다리고 있는 스텀 루틴에 전달됨**
4. **스텁 루틴은 이를 다시 풀어서 자신의 실행결과처럼 RPC를 호출한 함수에 전달함. 프로그래머 입장에서는 호출한 함수가 로컬인지 원격인지 여부를 구별할 필요가 없게 됨.**
   * 1. **원격 프로시저 호출의 구현**

**RPC 구현 상 고려해야할 사항**

이상적으로는 고급언어에서 RPC를 사용하는 것과 로컬 프로시저를 호출하는 것은 구별되지 않아야 한다.

but, RPC의 경우 네트워크 문제로 함수 호출에 실패할 수도 있고, 한 번 호출했지만 실제로는 두 번 이상 실행되는 경우 도 가능함. → 호출하는 프로시저나 호출되는 프로시저는 이를 감안 해야 한다.

네트워크로 메시지를 주고받기 때문에 RPC의 수행시간이 훨씬 느리다.

호출하는 프로시저와 호출되는 프로시저는 서로 다른 주소공간에 속하기 때문에 단순히 메모리 주소를 리턴하는 참조 호출은 의미가 없을 수 있음.

참조를 하려면 네트워크에서 해당하는 자원의 이름을 이용해야 한다. RPC의 수신자는 호출이 생성된 곳과 유사한 환경에서 실행해야 한다.

두 프로시저가 같은 주소공간에 있다면, 프로시저 사이에 정보를 전달하는 방법은 매개변수 외에도 전역변수 등 다른 방법 이 존재함.

RPC의 경우 전역변수를 통해 정보를 전달하려면 클라이언트-서버 사이에 별도 통신이 있어야 한다.

**RPC 구현의 예**

**클라이언트의 프로세스 theClient**

**클라이언트 응용 프로그램 코드, 클라이언트 스텁, 전송 메커니즘으로 이루어짐 서버의 프로세스 rpcServer**

원격 프로시저의 서버 구현, 서버 스텁, 전송 메커니즘으로 이루어져 있음. 코드 분석

# theClient main 함수

**로컬 프로시저 localF를 먼저 실행하고, RPC인 remoteF를 실행.**

**remoteF에 대응되는 클라이언트 스텁**

**Lookup을 이용하여 이름 서버에 remoteF가 어느 서버에서 동작하는 프로세스인지를 찾는다.**

**이러려면 rpcServer 프로세스는 remoteF()가 자신에 속하는 프로시저라는 것을 이름 서버의 Register 함수를 이용해 알려서 이 질의에 대한 답을 할 수 있도록 해야 한다.**

검색이 끝나면 클라이언트와 서버는 연결되어 두 번째 호출부터는 이 연결을 이용한다. 클라이언트 스텁

**Pack 함수**

자신에게 주어진 매개변수를 Pack 함수를 이용하여 네트워크로 전달될 메시지로 변환 메시지

기다란 문자열 하나, 첫번째 매개변수의 위치, 매개변수의 타입 등 정보를 넣는 것

**send 함수**

**rpcServer 프로세스 내의**

# rpcServer

서버 스텁

**"서버 스텁"으로 '메시지' 전송**

여러 원격 프로시저를 제공할 수 있는데, 이전 단계의 연결 정보와 입력 메시지를 이용하면 이 메시지가 어느 프로시 저에 가야 할지 알 수 있음.

**1) 호출을 수행 2) 실행결과를 메시지로 변환 3) 네트워크를 통해 theClient 프로세스에 전달 그동안 theClient 프로세스는 일시정지되며, 서버가 전달하는 결과가 올 때까지 기다리게 된다.**

**theClient 프로세스**

Unpack 함수를 이용하여 받은 '메시지'를 분리 결과를 자신을 호출한 함수에 전달

장단점

장점

다른 컴퓨터에서 동작하는 프로그램들 사이의 분산처리에 유용

분산 시스템에 대한 지식을 상대적으로 덜 요구하면서 분산 시스템 위에서 일반적인 프로그래밍과 유사하게 프로그래밍을 진행할 수 있음.

단점

고성능을 요구하는 병렬처리에 적합한 방법은 아님.

이유: 1) 결과가 올 때까지 프로시저는 동작을 멈추고 대기하기 때문 2) 네트워크 간 통신은 신뢰가 떨어지고 시간이 많 이 걸릴 수 있기 때문

요약

1. **분산 시스템은 네트워크를 통해 약하게 결합된 다양한 성능의 프로세서의 집합이며, 각 프로세서는 자신의 메모리와 클럭을 사용 한다.**
2. **분산 시스템의 장점은 자원 공유, 성능 향상, 신뢰성 향상, 통신의 편리성이다.**
3. **분산 시스템을 관리하기 위한 운영체제가 분산 운영체제이며, 사용자가 원격 자원을 로컬 자원을 사용하는 것처럼 쉽게 사용할 수 있고, 둘 사이의 구별이 없이 투명성을 제공함.**
4. **분산 파일 시스템은 클라이언트가 원격 파일과 로컬 파일을 구별하지 않고 처리할 수 있게 한다.**
5. **분산 메모리는 분산 시스템에 속한 컴퓨터들이 메모리를 공유할 수 있게 한다.**
6. **원격 메모리는 논리적 메모리를 정의하고 공유하도록 하는 원격 메모리 API를 통해 분산 메모리를 구현함.**
7. **분산 공유 메모리는 가상 메모리 기능을 확장하여, 한 주소공간에 로컬 메모리, 보조기억장치, 원격 메모리를 대응시켜 분산 메모 리를 구현함.**
8. **원격 프로시저 호출(RPC)을 통해 한 컴퓨터에서 동작하는 프로그램이 다른 컴퓨터에 있는 프로시저를 호출할 수 있다.**