Operating System (MS)

Essay #1: Architectural Support for OS in Cloud

모바일시스템공학과 김태경 (32211203)

목차

2024.12.22

1. 서론

1.1 배경

클라우드 컴퓨팅은 사용자에게 물리적 장비에 의존하지 않고도 데이터를 저장하고 앱을 실행할 수 있도록 유연성과 확장성을 제공한다. 이 기술은 최근 생활에서도 느낄 수 있는 AI(인공지능), ML(머신러닝), 빅데이터 분석, IoT(사물인터넷)와 같은 첨단기술들이 상업에 이용되기 시작하면서 필수 불가결한 핵심 인프라로 자리 잡으며, 현대 사회의 전반에 큰 영향을 미치고 있다. 운영체제는 이 클라우드를 효율적이고 안정적인 서비스를 제공하는 역할을 한다.

기존의 운영체제는 단일 컴퓨터 또는 네트워크 환경을 중심으로 설계되어 개별적인 하드웨어와 소프트웨어 간의 인터페이스를 제공하지만, 클라우드 컴퓨팅 환경에서 운영체제는 수많은 사용자가 동일한 물리적 자원을 공유하기에, 다수의 가상 머신(VM)과 컨테이너가 동시에 실행되어, 자원의 효율적인 관리와 데이터의 빠른 접근성, 안정적인 동시성 제어가 요구되어 새로운 유형의 운영체제가 필요하다.

1.2 목적

이 보고서는 클라우드 컴퓨팅에 대한 전반적인 이해와, 일반적인 컴퓨터와는 다른 환경인 클라우드 환경에서 운영체제가 어떻게 구현되어 있는지를 구체적으로 분석하고자 한다. 주요 분석 주제로는 다음과 같다.

1) 가상 메모리는 다수의 가상머신과 컨테이너를 어떻게 효율적으로 지원하는가
2) 분산 파일 시스템의 대규모 데이터의 저장과 접근 속도 최적화는 어떻게 이루어지는가

위와 같이 가상환경에서의 메모리 관리, 파일 시스템의 설계를 중심으로 운영체제의 기능과 역할을 분석하고, 현대 클라우드 컴퓨팅 환경에서 문제점을 찾아 개선안을 제시하여 운영체제의 설계 방향을 제안한다.

1.3 보고서 구성

* 2장에서는 클라우드 컴퓨팅의 개념과 특징을 분석.
* 3장에서는 클라우드 환경에 필요한 가상 메모리 관리 전략을 정리.
* 4장에서는 클라우드 환경 조성에 필요한 분산 파일 시스템에 대해 정리.
* 5장에서는 클라우드의 운영체제에서의 개선 아이디어 제안 및 구체화
* 6장에서는 결론 및 느낀점
* 7장에서는 참고문헌이 수록되어 있습니다.

2. 클라우드 컴퓨팅 개요

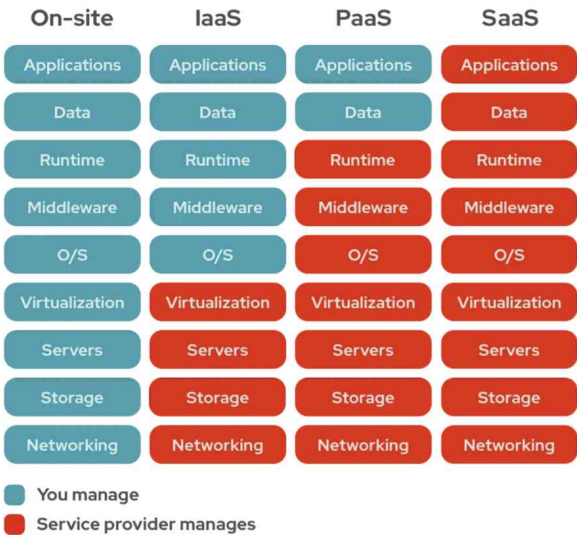
2.1 클라우드 컴퓨팅의 정의 및 개념

2,1,1 클라우드 컴퓨팅의 정의

: 클라우드 컴퓨팅은 컴퓨팅 자원(예: 서버, 스토리지, 네트워크, 소프트웨어)을 필요에 따라 제공하는 IT 서비스 모델로 인터넷 기반의 기술입니다. 이 서비스를 이용하는 사용자는 물리적인 장비나 인프라를 직접 소유, 관리할 필요 없이 서비스로 더 많은 자원을 활용할 수 있습니다. 이러한 IT 서비스 모델은 컴퓨팅 자원을 이용한 산업에서 초기 투자 비용을 낮추고, 효율적이고 경제적인 컴퓨팅 환경을 제공합니다.

2.1.2 클라우드 서비스 모델

: 클라우드 컴퓨팅은 제공되는 서비스의 유형에 따라 크게 세 가지 주요 모델로 분류됩니다. 서비스로서의 인프라, 플랫폼, 소프트웨어로 구분되며 각각의 모델들은 클라우드 환경에서 관리와 책임이 각기 달라 사용자의 상황에 맞춰 선택 및 결합하여 사용할 수 있습니다.



1) On-Site(온프레미스)

: 기업이 자체적으로 데이터센터, 서버, 네트워크 장비 등 IT 인프라를 보유하고, 이를 직접 설치, 운영, 유지보수하는 환경을 의미합니다. 모든 자원이 기업의 물리적 위치에 저장되며, 기업 내부 IT 부서가 이를 전적으로 관리합니다.

2) IaaS (Infrastructure as a Service)

: 하드웨어 인프라를 클라우드 서비스 제공업체가 관리하며, 네트워크, 스토리지, 가상화 계층을 포함한 기본 인프라의 유지보수를 담당합니다. 사용자는 운영체제, 애플리케이션, 데이터 등을 API 또는 대시보드를 통해 사용자는 자원을 직접 제어하고 관리합니다.

3) **PaaS** (Platform as a Service)
: 개발자들이 애플리케이션을 구축, 테스트, 배포, 관리할 수 있도록 플랫폼과 도구를 제공하는 클라우드 서비스 모델입니다. 하드웨어와 운영체제 등 기본 인프라는 클라우드 제공업체가 관리하며, 미들웨어, 데이터베이스, 프레임워크 등을 포함한 통합 솔루션을 제공합니다.

4) **SaaS** (Software as a Service)
: 소프트웨어 애플리케이션을 인터넷을 통해 서비스 형태로 제공하는 클라우드 모델입니다. 사용자는 소프트웨어를 설치하거나 유지보수할 필요 없이 웹 기반으로 애플리케이션에 접근하여 사용할 수 있습니다.

2.1.3 클라우드 배포 모델

: 클라우드 배포 모델은 인프라가 있는 위치, 인프라가 소유하고 관리하는 사람, 사용자가 클라우드 리소스 및 서비스를 제공하는 방법에 따라 각각 사용자의 요구사항에 따라 선택됩니다.

1) **Public Cloud**
: 여러 사용자가 동일한 물리적 인프라를 공유하는 클라우드 환경으로, 서비스 제공업체가 데이터 센터, 서버, 네트워크, 스토리지 등의 물리자원을 소유하고 관리하며, 사용자는 이를 임대하여 사용합니다.

2) **Private Cloud**
: 특정 조직에서만 접근할 수 있도록 설계된 독립적인 클라우드 환경으로, 조직 내부 또는 외부에서 운영될 수 있으며, On-site 환경에서도 구현 가능합니다. 높은 수준의 보안과 맞춤화 서비스를 제공합니다.

3) **Hybrid Cloud**
: Public Cloud와 Private Cloud를 결합하여 두 환경의 장점을 동시에 활용하는 클라우드 환경으로, 민감한 데이터와 일반 작업을 분리하여 처리합니다. 데이터의 이동에 따른 복잡성이 증가하지만, 유연성과 보안성을 동시에 제공합니다.

2.2 클라우드 컴퓨팅의 주요 특징

1) **확장성** (Scalability) : 사용자의 필요에 따라 자원을 동적으로 확장 또는 축소할 수 있습니다. 기존의 인프라에 CPU, 메모리 등 자원을 추가하여 성능을 높이는 수직적 확장(Verticla Scaling) 방식과, 추가적인 가상 머신이나 컨테이너를 생성하여 작업의 부하를 분산하는 수평적 확장(Horizontal Scaling) 방식이 있습니다.

2) **가용성** (Availability) : 클라우드 컴퓨팅은 고가용성을 통해 연중무휴 서비스를 제공합니다. 전 세계적인 사용자들에게 서비스가 지연되지 않도록 여러지역에 분산된 데이터센터를 운영하며, 데이터센터의 장애 발생 시 자동으로 복구할 수 있습니다.

3) **비용 효율성** (Cost Efficiency) : 자원을 사용한 만큼만 비용을 지불한 Pay-as-you-go 모델을 채택하여 운영하며, 이는 스타트업 같은 기업이 초기 하드웨어 투자 비용없이 클라우드 플랫폼을 사용하여 초기 비용을 줄이고, 불필요한 자원 낭비를 방지합니다.

4) **유연성** (Flexibility) : 클라우드 컴퓨팅은 다양한 플랫폼과 애플리케이션에서 사용가능하도록 멀티 플랫폼을 지원하기 위해 서로 다른 기술 스택 간의 호환성을 제공합니다. 이는 변화하는 비즈니스 요구사항에 빠른 대처가 가능하도록 보조합니다.

5) **자원 풀링** (Resource Pooling) : 물리적 자원을 가상화하여 여러 사용자가 동시에 자원을 공유할수 있는 Multi-Tenant 구조를 제공하며, 사용자의 요청에 따라 자원이 동적으로 할당됩니다.

2.3 클라우드 컴퓨팅과 운영체제의 관계

2.3.1 작업 환경 변화

클라우드 컴퓨팅은 물리적 자원을 가상화하여 유연한 서비르를 사용자에게 제공하며, 운영체제는 자원관리와 작업처리의 핵심적인 역할을 수행합니다. 전통적인 운영체제는 단일 컴퓨터의 자원을 관리하는데 초점을 두지만, 클라우드 환경에서는 다중 사용자와 가상화된 자원들을 지원하기 위해 추가적인 기능을 제공해야 합니다.

2.3.2 운영체제의 기본 역할

하드웨어와 소프트웨어 간의 중재자로 작동하여, 사용자와 앱이 하드웨어 자원을 효율적으로 사용할 수 있도록 지원합니다.

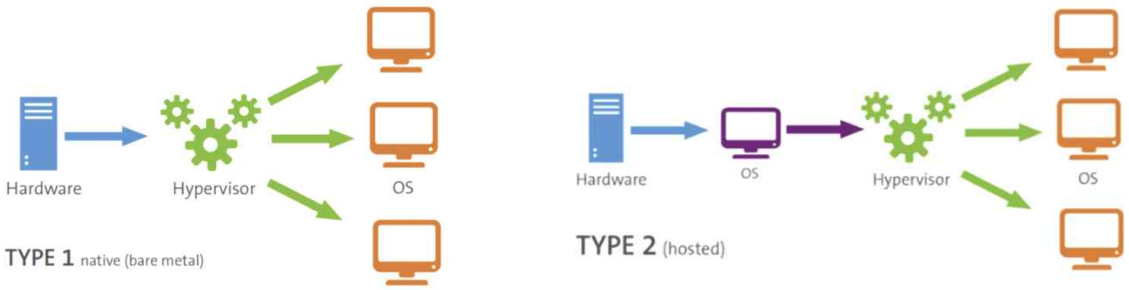
- 프로세스 관리: CPU 스케줄링을 통해 여러 작업을 동시에 실행한다.
- 메모리 관리: 프로그램 실행에 필요한 메모리를 동적으로 할당 및 해제한다.
- 파일 시스템 관리: 데이터 저장 및 접근을 관리하여 파일의 무결성과 효율성을 보장한다.
- 네트워크 관리: 데이터 패킷의 송수신 및 연결 상태를 유지하여 통신 안정성 제공한다.

2.3.3 클라우드 환경에서 확장된 역할

1) **가상 머신**

: 하드웨어 자원을 논리적으로 분리하여 물리적 자원을 가상화하며, 각각의 가상환경에서 자원 격리 및 효율적인 자원분배를 지원합니다. 각 자원들은 동적 자원 할당을 통해 워크로드에 적용됩니다. 이 기술 지원은 하이퍼바이저(Hypervisor) 소프트웨어를 통해

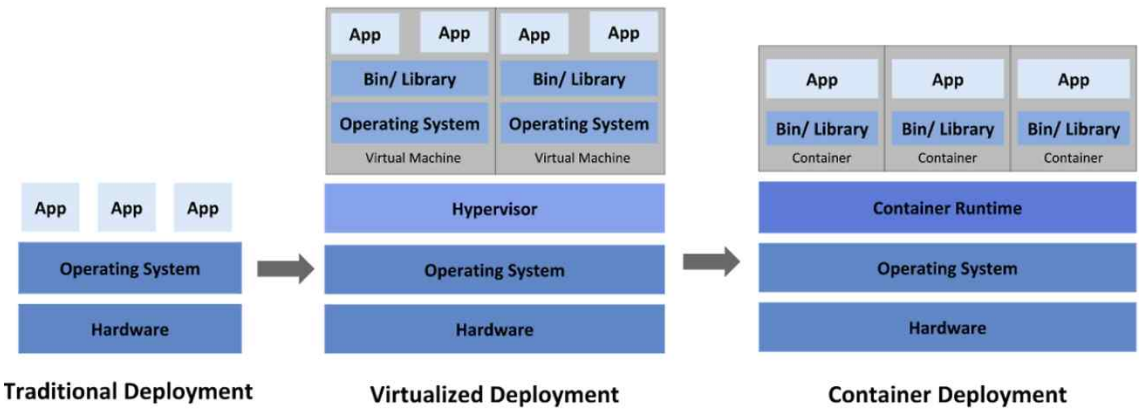
지원되며, 하드웨어 위에서 직접 실행되는 베어메탈(Bare-Metal) 하이퍼바이저와 기존 운영체제 위에서 실행되는 소프트웨어 하이퍼바이저(Hosted Hypervisor)로 나뉜다.



2) **컨테이너 기술 지원**

: 가상화된 환경에서 애플리케이션과 필요한 파일을 하나의 런타임 환경으로 묶어 격리하고 실행할 수 있는 기술로, 운영체제는 이를 지원하기 위한 커널 기능을 제공합니다. 가상화 기술에는 각각의 독립된 OS가 존재하지만, 컨테이너는 호스트 OS의 커널을 공유하여 각각의 컨테이너를 격리된 프로세르로 실행합니다. 이는 앱을 패키징하여 환경 간의 이동에 영향받지 않도록 의존성 문제를 해결하여 개발과 운영의 일관성을 보장하는데 중요한 역할을 합니다.

.



2.4 클라우드 기술의 응용 사례

1) AI/ML(인공지능/머신러닝) 워크로드

2) 클라우드 기반 빅데이터 분석

3) 웹 애플리케이션 호스팅

4) 백업 및 복구작업

3. 가상 메모리 관리

3.1 클라우드 환경에서의 가상 메모리

3.1.1 **메모리 오버커밋 문제**

클라우드 환경은 다수의 사용자와 워크로드를 수용하기 위해 다중 가상 머신(VM)과 컨테이너를 실행하는데, 이러한 환경에서는 물리적 메모리보다 더 많은 가상 메모리를 할당하는 메모리 오버커밋(Memory Overcommit)이 일반적으로 사용됩니다.

메모리 오버커밋은 클라우드 서비스의 제공자가 비용 효율성을 높이기 위해, 동일한 물리적 메모리에서 더 많은 워크로드를 실행하게 해주지만, 아래와 같은 문제를 일으킬 수 있습니다.

1) 서비스 품질(QoS) 저하 : 다중 워크로드는 메모리를 경쟁적으로 사용하는데, 특정 VM이나 컨테이너에서 메모리를 과도하게 사용하는 경우 메모리 부족 상태가 발생하여 성능이 저하됩니다.

2) 성능 병목 현상 : 메모리 초과 사용으로 인해 디스크 메모리 등으로 swap이 발생하는 경우, RAM에 비해 접근 속도가 느려지므로 성능 병목 현상이 발생합니다.

3) 시스템 충돌 : 메모리 오버커밋이 과도하게 발생할 경우, 워크로드의 작업이 불안정하거나, 메모리 부족으로 인해 가상 머신이나 컨테이너가 비정상적으로 종료될 수 있습니다.

따라서 클라우드 환경에서 발생하는 메모리 오버커밋 문제를 해결하기 위한 기술을 필요로 합니다.

3.1.2 **해결 메커니즘**

메모리 오버커밋으로 인해 발생하는 문제를 해결하는 방법으로 클라우드 서비스는 아래와 같은 메모리 관리 메커니즘을 사용하고 있습니다.

1) **메모리 스왑**(Swap)

RAM의 용량이 부족할 경우 디스크 공간을 가상 메모리로서 사용하여 물리적 메모리를 보완하는 기술입니다. 메모리가 부족하여도 적재적소에 메모리를 활용하여 시스템은 종료되지 않고 실행되는 서비스의 지속성을 보장합니다. 다만 디스크 I/O의 성능이 RAM보다 느리기에 Swap이 많이 발생할 경우 성능 정하를 초래할 수 있습니다.

2) **메모리 압축**(Memory Compression)

메모리의 페이지를 압축하여 사용 가능한 메모리 용량을 증가시키는 기술입니다. 이 기술은 디스크로의 swap 횟수를 줄여 성능을 유지시키며, 메모리 공간의 효율성을 극대화합니다. 다만 메모리 압축과 해제의 과정에서 CPU 오버헤드가 발생할 가능성이 있습니다.

3) **캐시 메모리의 최적화**

I/O 병목 현상을 줄이기 위해 자주 접근하는 데이터를 RAM에 우선 저장하여 성능을 개선하는 기술입니다. 메모리 보다 CPU에 더 가까운 RAM에 자주 사용되는 데이터를 임시 저장하여 접근 속도를 향상 시켜 전체적인 성능을 개선합니다.

3.1.3 메모리 격리를 통한 보안 및 성능 보장

여러 사용자가 동일한 물리적 자원을 공유하는 클라우드 환경에서는 메모리 격리를 통하여 보안 및 성능을 보장합니다. 이를 구현하기 위하여 아래와 같은 기술을 필요로 합니다.

1) **하드웨어 기반 메모리 격리** : 각 VM이 자신의 메모리 공간만 접근하도록 보장하여, VM 간 메모리 접근을 하드웨어 수준에서 차단하여 데이터 유출을 방지합니다. 이는 보안이 강회되어 악의적인 가상머신이 다른 가상머신의 데이터를 읽거나 수정할 수 없도록하여, 가상머신의 메모리 성능을 보장합니다.

ex) Intel VT-x(Intel Virtualization Technology), AMD SVM(Secure Virtual Machine)

2) **컨테이너 환경에서의 네임스페이스와 cgroups**

네임스페이스(Namespce)는 각 컨데이너에 고유한 네임 스페이스를 부여하여, 프로세스와 자원이 서로 격리되로록 설정하고, cgroups(Control Groups)은 컨테이너별로 메모리 사용량 제한을 설정하고, 과도한 자원 사용을 방지합니다. 이로서 컨테이너 간의 자원 경쟁 문제를 최소화 할 수 있으며, 과도한 메모리 사용으로 인한 성능 저하를 방지합니다.

3) **운영체제 기반 격리 기술**

운영체제 기반 격리 기술은 물리적 또는 가상화된 환경에서 다중 프로세스와 사용자 간의 자원 접근을 제한하고, 보안을 보장하기 위한 기술입니다. 이 기술은 가상 머신과 컨테이너의 단점을 보완하며, 다중 사용자 환경에서의 데이터 보호와 프로세스 격리를 강화합니다.
아래는 주요 운영체제 기반 격리 기술입니다.

- 접근 제어와 보안 정책 강화: SELinux, AppArmor
- 네임스페이스를 통한 격리 : 프로세스 그룹이 고유한 자원 공간을 가져, 격리를 제공.
- Cgroups(Control Groups) : 프로세스나 컨테이너에 할당할 CPU. 메모리 양을 제한.
- 프로세스 격리를 통한 보안 : Chroot Jai, Sandboxing

3,2 페이지 교체 알고리즘

클라우드 환경에서는 사용자의 요청과 워크로드가 시간에 따라 급격하게 변화하는데, 이러한 동적 관리를 처리하기 위한 효율적인 메모리 관리가 필요합니다. 사용자에게 항상 높은 성능과 낮은 지연 시간을 서비스로 제공해야하는 클라우드는 이를 위해 캐시 적중률을 높이고, 페이지 폴트를 줄이는 효율적인 알고리즘이 필요합니다.

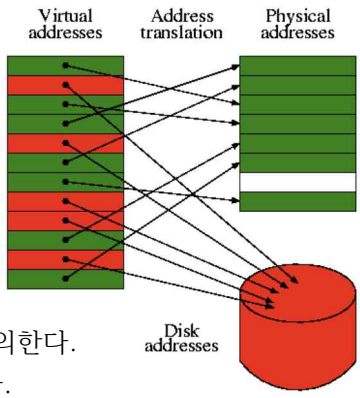
3.2.1 LRU(Least Recently Used) 알고리즘

가장 오랫동안 사용되지 않은 페이지를 교체하는 알고리즘으로, 메모리 내 페이지의 참조 패턴을 기반으로 작동하며, 최근에 자주 사용된 페이지는 유지하고 덜 사용된 페이지를 교체합니다.

클라우드에서는 클라우드 기반 메모리 캐싱 시스템에서 자주 사용되며, 이는 사용자의 요청 패턴을 분석하여 데이터를 효율적으로 캐싱하도록 구현됩니다. 이는 데이터베이스나 웹 애플리케이션에서 LRU 알고리즘을 사용하여 응답시간 최소화하는데 기여합니다. LRU는 페이지 사용 이력을 추적하는데 높은 오버헤드가 발생하며, 대규모 클라우드 환경에서는 구현 복잡성이 증가하기에 효과적으로 실행하기 위한 데이터 구조가 마련되어야 합니다.

3.2.2 Working Set 알고리즘

활성화된 작업 집합(프로세스에서 사용하는 물리적 메모리)을 기준으로 메모리 페이지를 관리하는 기법입니다. 이 알고리즘은 페이지 교체가 과도하게 발생하는 Thrashing 문제를 방지하는데 유용합니다.



작동원리

- 프로세스의 작업 집합을 일정 시간동안 참조된 페이지로 정의한다.
- 메모리에 유지해야할 페이지 집합을 작업 집합으로 제한한다.
- 작업 집합에 포함되지 않은 페이지는 교체작업으 진행한다.

클라우드 워크로드에서는 프로세스의 메모리 요구량의 변동이 적기에, 메모리를 효율적으로 사용하고 불필요한 페이지 교체 작업을 최소화 할 수 있습니다. 다만 작업 집합을 추적하기 위한 추가적인 계산 및 메타 데이터의 관리가 요구됩니다.

3.2.3 클라우드 워크로드 최적화 전략

자원 효율성, 비용 절감, 안정성을 위한 클라우드의 필수적인 요소로, 서비스 제공자의 경쟁력을 강화하고, 사용자에게는 안정적이고 신뢰할 수 있는 서비스를 제공하는 데 핵심적인 역할을 합니다.

1) Dynamic Page Balancing

가상 머신 간의 메모리 사용량의 균형을 조정하여 전체 시스템의 메모리 사용률을 최적화하는 전략입니다. 이는 각 가상 머신의 메모리 사용량을 모니터링하면서 메모리 과잉 또는 부족 상태에 따라 메모리를 동적으로 재배분합니다.

이는 클라우드 환경에서 메모리 부족으로 인한 가상 머신의 성능 저하를 방지하며, 불필요한 swap이 감소하여 클라우드 워크로드 변동에 유연하게 대응할 수 있습니다.

2) **Adaptive Replacement Cache (ARC)**

ARC는 LRU(Least Recently Used)와 LFU(Least Frequently Used)를 결합한 교체 알고리즘으로, 클라우드 워크로드의 다양성에 최적화된 페이지 교체 방식을 제공하는 전략입니다. 이는 LFU 캐시와 LRU 캐시를 병렬도 관리하며, 자주 사용되는 페이지와 최근의 사용된 페이지 간의 균형을 페이지 참조 패턴에 따라 캐시의 비율을 조정합니다. 복잡한 구조로 인해 LRU보다는 높은 오버해드가 발생하며, 메모리 제한이 엄격한 환경에서의 구현은 힘들지만, 워크로드 패턴이 빠르게 변화하는 클라우드 App에서 효과적입니다.

3.3 NUMA 구조 최적화

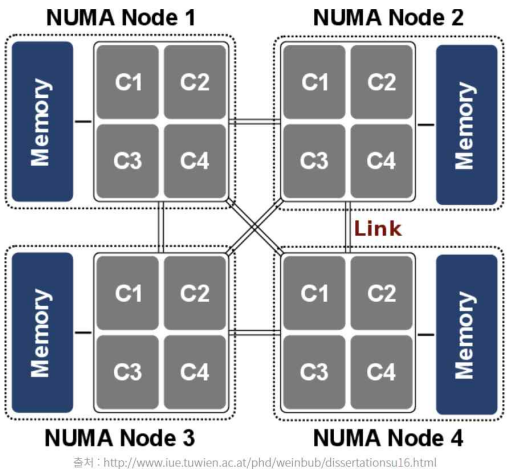
3.3.1 **NUMA의 개념** (Non-Uniform Memory Access)

다중 프로세서 시스템에서 메모리에 접근할 때의 물리적 거리 차이에 따라 메모리 액세스 시간이 달라지는 구조를 뜻합니다. local 메모리를 통해 CPU와 물리적으로 가까운 메모리 영역. 접근 속도가 빠릅니다. 이는 대규모 병렬 작업에서의 성능은 향상시키며, 지역메모리를 통해 메모리 병목 현상을 완화합니다.

클라우드 데이터 센터는 대규모 병렬 처리가 필요한 AI/ML 작업, 데이터 분석, 고성능 컴퓨팅(HPC)을 실행합니다. NUMA 구조는 작업과 메모리 간의 물리적 거리 최적화는 작업 지연을 줄이고 처리량(Throughput)을 증가시켜 작업의 성능을 향상시킵니다.

3.3.2 **NUMA 환경에서 메모리 액세스 속도 최적화**

메모리 액세스 지연을 최소화하고 성능을 극대화하기 위해 메모리 지역성과 NUMA-aware 스케줄링이 필요합니다.



출처 : http://www.iue.tuwien.ac.at/phd/weinbub/dissertationsu16.html

메모리 지역성(Locality)은 작업이 실행되는 CPU에 가까운 지역 메모리를 우선적으로 사용하도록 설계하며, 데이터와 작업이 동일한 노드에 배치되면 원격 메모리 접근의 지연 시간을 줄일 수 있습니다. 데이터 지역성을 유지하기 위한 방법으로는 CPU가 속한 노드의 메모리를 먼저 할당하는 메모리 할당 우선순위 설정하는 방법과, NUMA-aware 라이브러리를 활용하는 방법이 있습니다.

NUMA-aware 스케줄링은 작업 스레드가 실행되는 CPU와 메모리가 물리적으로 가까운 위치에 배치되도록 스케줄링합니다. NUMA 스케줄링의 전략으로는 CPU와 메모리의 물리적 거리를 고려하여 가까운 노드의 자원을 우선적으로 사용하며, 특정 노드에 과도하게 작업이 몰리지 않도록 워크로드의 균형을 유지합니다.

3.4 VM 메모리 관리 전략

3.4.1 **메모리 공유 기법**

KVM과 Xen과 같은 가상화 플랫폼에서는 Transparent Page Sharing (TPS) 기술을 통해 여러 가상 머신(VM)이 동일한 메모리 페이지를 공유할 수 있습니다. 이는 메모리 페이지의 hash값을 계산하여 중복된 페이지를 탐지하고, 중복된 페이지를 단일 페이지로 병합 후 병합된 페이지를 공유 페이지로 설정하여 여러 가상머신이 참조되도록 작동합니다.

이는 메모리 사용량을 줄여 물리저 자원의 효율성을 증가시키며, 가상화 환경에서 더 많은 가상 머신을 실행할 수 있게 만들어줍니다.

3.4.2 **메모리 격리 기법**

1) **Xen의 Dom0** : Xen 하이퍼바이저(가상화 프로그램)는 Dom0라는 관리 도메인을 사용하여 메모리 격리를 구현하는데, 이 도메인은 하이퍼바이저와 직접 통신하고 다른 VM 간의 메모리 접근을 제한하여 보안을 강화합니다. 이는 가상 머신 간의 메모리 간섭을 방지하며, 하이퍼 바이저가 메모리 할당과 접근을 완전히 통제합니다.

2) **KSM** (Kernel SamePage Merging) : KVM은 KSM 기술을 사용하여 중복된 메모리 페이지를 탐지하고 단일 페이지로 병합합니다. 가상 머신의 메모리 페이지르 주기적으로 스캔하여 중복된 페이지를 탐지한 후 병합 가능한 페이지들은 하나로 병합합니다. 이는 메모리의 사용량을 감소시키며, 페이지 복사를 통하여 무결성을 유지하고, 가상 머신 간의 격리를 유지하면서 성능과 보안을 보장합니다.

3.4.3 **Ballooning 기법**

Ballooning 기법은 가상화 환경에서 메모리의 동적 재분배를 가능하게 하는 기법으로, 특정 VM에서 불필요한 메모리를 회수하여 다른 VM에 동적으로 할당합니다. 하이퍼바이저가 메모리 부족 상황을 감지하면, Balloon 드라이버가 실행 중인 VM 내에서 메모리를 요청하고 회수된 메모리를 다른 VM으로 재할당됩니다.

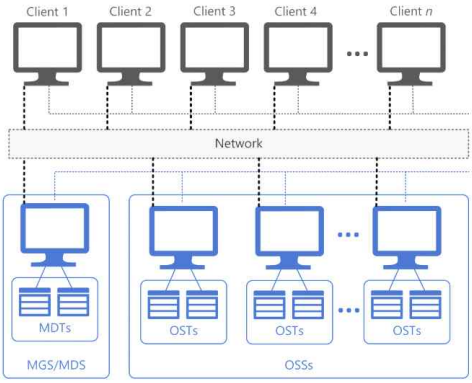
가상화 환경에서의 가상 메모리 관리 전략은 자원의 효율적 사용과 안정적인 서비스 제공을 가능하게 합니다. KSM과 같은 기술은 메모리 사용량을 줄이고 자원을 공유하며, Dom0와 같은 격리 기법은 보안을 강화합니다. Ballooning 기법은 유연한 자원 분배를 통해 클라우드 환경의 변화에 적응할 수 있도록 합니다.

4. 분산 파일 시스템

4.1 클라우드 환경에서의 분산 파일 시스템(DFS)

4.1.1 정의 및 개념

중앙 집중식 시스템에서 발생할 수 있는 병목 현상을 방지하기 위해 분산 파일 시스템((Distributed File System, DFS)은 데이터를 여러 물리적 노드에 저장하고, 데이터의 저장 및 접근 효율성을 극대화 하는 기술입니다. DFS는 네트워크로 연결된 여러 서버의 클러스터로 데이터에 대한 빠른 액세스를 지원합니다.



4.1.2 주요 특징

1) 확장성(Scalability) : 데이터를 분산하여 저장하여 시스템 확장이 용이하며, 새로운 노드를 추가하는 방식으로 저장용량과 처리 성능을 사용자의 요구조건에 따라 증가 또는 감소시킬 수 있습니다.

2) 가용성(Availability) : 데이터 복제를 통해 서버에서 장애가 발생하더라도 서비스를 지속적으로 제공할 수 있습니다.

3) 신뢰성(Reliability) : 서버 장애나 네트워크 문제 발생 시 데이터 손실을 방지하기 위해 복제본을 유지합니다. 이는 다중 사용자와 애플리케이션의 안정적인 데이터 접근을 지원합니다.

4.1.3 분산 파일 시스템의 아키텍처

1) **Master-Slave 구조**

slave들은 master에게 서비스를 복제하여 제공하며, master는 복제한 서비스의 호출을 구성하고 slave의 전략에 따라 하나의 slave(결과)를 선택합니다. slave들은 각각 같은 기능을 다른 알고리즘으로 수행하기도 하며, 또는 완전히 다른 기능을 수행하기도 합니다.

이 구조는 병렬 컴퓨팅을 가능하게 하여, 모든 slave가 독립적으로 병렬으로 실행될 수 있습니다. 동일한 작업을 여러 slave에 분배하여 실행하므로, 대규모 데이터 처리에서 다양한 방법을 이용해 작업을 수행하여 부정확한 결과를 방지할 수 있습니다. 또한 여러 서버에 나뉘어 작업을 수행하는 만큼 클라우드 시스템 내의 장애 발생 시에도 계속 운영할 수 있도록 합니다.

2) **P2P 구조**

P2P(peer-to-peer) 구조는 DFS에서 노드 간의 동등한 관계를 기반으로 데이터를 저장 및 복제하는 방식으로, 중앙으로 집중화 된 제어노드 없이 각 노드가 자율적인 작업을 수행하고, 데이터를 복제하는 것이 핵심입니다.

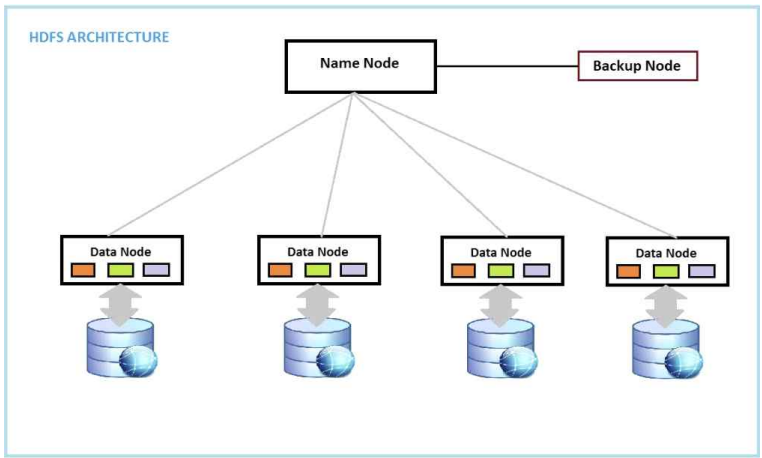
작동 원리

- 데이터 분산 및 복제 : 데이터가 네트워크에 연결된 여러 노드에 분산되어 저장된다.
- 자율적 작업 수행 : 각 노드는 자체적으로 데이터들의 저장, 관리, 복제하여 중앙 제어노드 없이도 작업을 수행한다.
- 부하 분산 : 다수의 사용자가 동시에 데이터를 요청할 경우, 트레픽을 여러 노드로 분산시킨다.
- 자체 복구능력 : 장애가 발생한 노드를 감지하면, 자동으로 복구작업을 진행하여 데이터 복제본을 다른 노드에 새로이 생성한다.

이 구조는 복잡성과 네트워크 비용의 증가한다는 단점이 존재하지만, 클라우드 환경에서는 대규모 데이터 처리가 필요하기에 필수적인 역할을 합니다.

4.2 Hadoop Distributed File System (HDFS)

대규모 데이터를 분산, 저장 및 처리하기 위해 설계된 분산 파일 시스템으로, 데이터를 물리적으로 분산하여 저장하면서도 사용자에게 단일 파일 시스템으로 보이도록 설계되었습니다.



4.2.1 HDFS의 구조

1) **Name_node** : HDFS의 중심 관리노드로 파일시스템의 메타 데이터를 관리합니다. 관리 항목으로는 파일과 디렉토리의 이름과 계층구조, 파일의 블록매핑 정보, 블록 복제본의 위치 등으로, 메모리에 메타 데이터를 유지하여 빠른 접근이 가능합니다.

2) **Data_node** : 실제 데이터를 물리적으로 저장하며, 파일은 블록 단위로 나뉘어 여러 데이터 노드에 저장합니다. data_node는 주기적으로 자신의 보유 데이터 블록의 상태를 name_node에게 보고하며, name_node의 지시에 따른 읽기 및 쓰기 작업을 수행합니다.

4.2.2 **데이터 읽기/쓰기 흐름**

1) **읽기 프로세스** : 데이터 접근이 병렬로 이루어져 네트워크 대역폭을 효율적으로 사용하며 사용자가 읽고자 하는 파일의 정보를 name_node에 요청한 후, name_node는 파이르이 데이터 블록 위치를 사용자에게 반환한다. 그 후 사용자는 지정된 데이터 노드에서 직업 데이터를 읽어들인다.

2) **쓰기 프로세스** : 사용자가 name_node에 파일 저장을 요청하면, name_noode는 파일을 여러 블록으로 분할한 후 블록들의 저장 위치를 지정한다. 사용자는 지정된 data_node로 데이터를 전송하며, 복제가 필요할 경우 블록 복제가 수행된다.

4.2.3 **HDFS의 장점과 한계**

1) **장점** :
- 대규모 데이터 처리에 적합 : 데이터를 블록단위로 분산 저장하여, 병렬처리를 통한 대규모 데이터를 효율적으로 관리한다.
- 데이터 지역성 지원 : 데이터를 계산 작업을 수행하는 곳과 가까운 위치에 저장하여 네트워크 대역폭의 사용을 최소화 하고 작업 속도를 향상시킨다.
- 고 가용성과 신뢰성 제공 : 데이터를 복제하여 백업을 진행해두어 데이터로의 접근을 보장한다.

2) **단점** :
- 단일 장애점 : name_node가 작업의 수행을 지시하는 노드로서, name_node가 다운되면 전체 시스템이 작동하라 수 없다.
- 높은 메모리 요구 : name_node는 파일 시스템의 모든 메타 데이터를 메모리에 저장하기에, 메타 데이터의 양이 증가하면 메모리 요구량이 증가한다.
- 실시간 데이터 불가 : 블록단위로 분리되어 저장되는 데이터로 인해, 실시간 처리보다는 Batch처리에 더 적합하다.

4.3. 데이터 지역성과 스토리지 최적화

4.3.1 **데이터 지역성의 중요성**

데이터 지역성(locality)는 DFS에서 계산작업을 수행하는 노드와 데이터를 저장하는 노드 간의 물리적 거리를 최소화 하는 개념입니다. 이를 통해 네트워크를 통해 데이터를 전송하는 시간을 줄일 수 있어 작업 처리속도를 향상시킬 수 있습니다.

특히 빅데이터 분석과 머신러닝 학습 같은 데이터 집약적 작업은 클라우드 환경에서 대규모 데이터의 이동이 이루어지는데 데이터의 저장위치와 작업 노드 간의 거리가 성능에 영향을 미치게 되는데. 이 개념으로 네트워크 병목현상 완화와 자원의 활용도를 극대화합니다.

4.3.2 **데이터 지역성 구현 기술**

1) **스케줄링 기반 지역성 보장**

DFS에서 각 데이터는 클러스터 내의 여러 노드에 분산 저장되며, 동일한 데이터의 복제본이 여러 노드에 존재합니다. 스케줄러는 데이터가 저장된 노드에서 작업을 실행하거나, 해당 데이터의 복제본이 있는 가장 가까운 노드에서 실행되도록 작업을 배치합니다.

작동 원리

- 데이터 위치 정보 수집 : name_node를 통해 데이터가 저장된 노드의 위치들을 추적한다.
- 작업 데이터 매핑 : 작업 스케줄러는 처리해야 할 원본 데이터의 위치를 확인하고, 작업을 실행하려 시도한다.
- 지역성 수준 결정 : 작업이 데이터 저장 노드에서 실행되지 않을 경우, 스케줄러는 데이터의 복제본이 있는 다른 가까운 노드에서 작업을 실행 시도한다.
- 작업 실행 : 작업이 data_node에서 실행되도록 배치한 후, 네트워크로 데이터 이동없이 작업을 수행한다.

이 기술은 복잡한 스케줄링 알고리즘을 필요로 하고, 데이터가 일부 노드에 집중적으로 저장되어 있거나 특정 데이터로의 접근이 많을 경우 워크로드 불균형과 노드의 가용성 문제가 있습니다. 다만 네트워크 대역폭의 절감, 작업처리 속도 향상, 시스템 자원의 최적화를 이루어내어 클라우드 컴퓨팅의 최적화를 일부 이루어냈습니다.

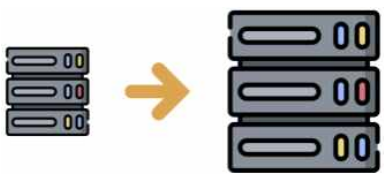
2) **클라우드 서비스 사례**

- Google Cloud Storage & Amazom S3 : 데이터를 물리적으로 분산 저장하면서 사용자가 가장 가까운 데이터 센터에서 복제된 데이터를 읽어오도록 설계되어 네트워크 지연을 최소화한다.

4.4 파일 시스템 확장성과 장애 복구

4.4.1 **파일 시스템 확장성**

DFS는 클라우드 환경에서 저장소의 크기와 처리능력을 유연하게 증가 감소시킬 수 있는 서비스를 제공합니다. 기존의 성능 향상은 수직확장으로 보유하고 있던 단일 장비의 성능을 증가시는데 반해, DFS의 주요 특징 중 하나로 수평 확장은 시스템에 여러 장비를 추가하여 저장용량과 성능을 개선할 수 있습니다.



(Verticla Scaling)

(Horizontal Scaling)

1) **수직적 확장**(Verticla Scaling)

: 기존에 사용하던 장비에서 높은 사양으로 업그레드 하는 것을 얘기합니다. 하드웨어 관점에서는 컴퓨터에 cpu, 메모리와 같은 자원을 추가하는 작업이 있으며, 하나의 서버가 수행할수 있는 능력을 증가시키는 방식입니다. 이 방식은 서버 성능을 개선하는데 비용증가의 폭이 크며, 하나의 장비로 운영되기에 성능의 확장에 한계가 존재합니다. 또한 한 대의 서버에 부하가 집중되기에 장애가 발생할 수 있으며, 다른 작업들이 동시에 수행될 경우 서로에게 영향을 주기도 합니다.

2) **수평적 확장**(Horizontal Scaling)

: 기존의 수직적 확장과는 달리 동일한 서버를 증가시키는 것으로, 하나의 서버에서 처리하던 작업을 여러 서버에게 나누어 수행할 수 있습니다. 이는 장비 자체를 추가하는 방식으로 지속적인 확장이 가능하며, 장비의 성능 개선 대비 비용부담이 적습니다. 각각의 작업들을 여러 장비들에서 운영되어 독립성을 유지할 수 있으며, 여러 서버에 분산되어 일부 서버에서 장애가 발생하더라도 지속적인 작업수행이 가능합니다.

5. 개선 아이디어 제안

클라우드 컴퓨팅에서 현대의 AI/ML 및 빅데이터 애플리케이션은 데이터 중심의 작업을 수행하며, 실시간 데이터 접근이 필수적입니다. 이러한 워크로드는 기존의 분산 파일 시스템(DFS)이 제공하는 Batch 처리 방식으로는 충분히 지원되지 않습니다. 실시간 데이터를 효과적으로 관리하고 처리할 수 있는 새로운 DFS 설계가 필요합니다.

5.1 AI/ML 및 빅데이터 애플리케이션의 요구사항

클라우드 환경에서 AI/ML 및 빅데이터 애플리케이션을 효과적으로 지원하기 위해서는, 작업의 고유한 요구사항을 이해하고 이를 충족하는 분산 파일 시스템을 설계해야 합니다.

1) **데이터 유형**

- AI/ML : 이미지, 텍스트, 음성 데이터와 같은 구조화되지 않은 데이터가 주를 이룬다.
- 빅데이터 : 실시간 로그, 센서 데이터, 스트리밍 데이터 등 대규모 데이터를 빠르게 처리해야 한다.

2) **작업 방식**

- AI/ML : 데이터셋을 훈련, 테스트, 검증 단계로 분리하고 병렬처리로 효율성을 극대화. 모델 훈련 과정에서 반복적으로 동일한 데이터에 접근.
- 빅데이터: 실시간 데이터 스트림의 처리 및 분석을 진행하며, 대규모 데이터 집계를 위한 고속 데이터 접근 필요.

3) **저장 방식**

- 데이터 지역성: 데이터, 작업이 물리적으로 가까운 위치에 저장되어 네트워크 지연을 줄임.
- 캐시 최적화: 빈번히 접근하는 데이터를 고속 캐싱으로 처리하여 성능을 향상시켜야 한다.

5.2 클라우드 환경에서 발생하는 문제유형

클라우드 환경을 활용하는 AI/ML 및 빅데이터 애플리케이션은 데이터 접근 속도와 복제 관리, 캐싱 효율성이 시스템 성능의 핵심 요소로 작용하지만, 기존 DFS의 설계 한계로 인해 다음과 같은 문제가 발생합니다.

1) **데이터 접근 지연** : 다수의 요청이 특정 노드로 집중되면 네트워크 대역폭 한계로 병목 현상이 발생하여 데이터 접근 시간이 증가하고, 응답 속도가 느려집니다.

2) **비효율적인 데이터 복제 관리** : 고정된 복제본 수와 비효율적인 위치 설정으로 인해 복제본 생성 및 재배치 속도가 느리고, 데이터 접근 시간이 길어집니다.

3) **데이터 저장 방식의 비효율성** : AI/ML 및 빅데이터 애플리케이션의 데이터 유형(예: 이미지, 텍스트, 로그 데이터)을 적절히 지원하지 못하여 저장 및 접근 효율성이 낮습니다.

5.3 분산 파일 시스템 개선 방안

1) 워크로드 맞춤형 데이터 복제 및 배치

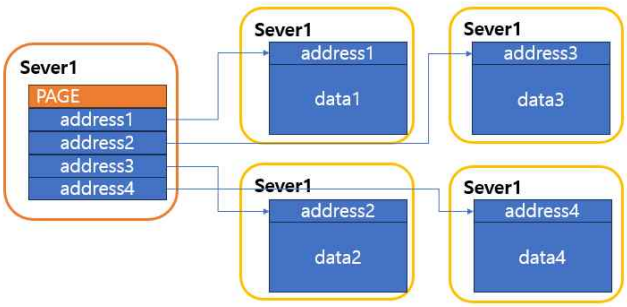
AI/ML 및 빅데이터 워크로드의 핵심적인 요구사항은 데이터 접근 속도를 극대화하고 네트워크 병목을 방지하는 것입니다. AI/ML 작업에서는 데이터셋이 반복적으로 접근되며, 빅데이터 작업에서는 실시간 스트리밍 데이터의 처리가 주로 요구됩니다.

두 작업 모두 특정 워커 노드에서 연산이 집중적으로 이루어지기에, 자주 접근되는 데이터를 '핫 데이터'로 식별하고 작업 노드와 물리적으로 가까운 노드에 복제하여 처리 속도를 향상시킬 수 있습니다. 이 과정에서 가까운 노드들을 캐시 메모리처럼 작업의 우선순위와 접근 빈도를 활용한 알고리즘을 사용하여 노드에는 작업에 필요한 데이터들의 유형과 실행할 작업의 행동패턴에 적절한 알고리즘을 실시간으로 적용하여 데이터들을 보관한다면 더욱 성능을 향상시킬 수 있습니다.

2) 데이터 유형 맞춤 저장

AI/ML 및 빅데이터 애플리케이션은 다양한 데이터 유형을 처리합니다. 이를 효과적으로 관리하기 위해, 데이터 유형에 따라 저장 방식을 차별화해야 합니다. AI/ML 작업에서는 비구조화 데이터인 이미지, 동영상과 같은 대용량 데이터들도 다루는데, 이러한 데이터 유형은 압축하여 저장공간 사용량을 줄이도록 하며, 고속 저장장치(SSD)에 저장하여 빠른 저장과 접근을 가능하게 조치하면 성능을 향상시킬 수 있습니다.

반면에 빅데이터 작업에서는 스트리밍 데이터와 생성되는 로그 기존의 블록으로 데이터를 나누어 저장하는 방식을 무분별하게 저장하는 것이 아닌 페이징 기법과 같이 데이터의 순서를 세우고 각 저장되는 블록 단위의 데이터들의 위치가 별도의 페이지에 기록시키도록 구현하여 대용량 데이터에 대한 저장공간의 효율성과 접근 속도를 향상시킬 수 있습니다.



3) 실시간 데이터 접근 최적화

실시간 데이터를 효율적으로 관리하려면 데이터 접근 경로의 설계가 중요합니다. 초고속 접근이 필요한 데이터는 RAM을 통해 접근하도록 하여 실시간 응답을 보장하며, 대용량 데이터를 이동하는 방법으로는 네트워크를 우회하여 메모리 간의 직접 데이터 전송을 가능하게 하는 RDMA 기술을 적용하여 데이터의 접근 속도를 향상 시킬 수 있습니다.

뿐만 아니라, 빈번한 접근이 발생하는 데이터를 효과적으로 처리하기 위해 계층적 캐싱 구조를 설계해야 합니다. RAM을 최상위 캐시로 활용하고, NVMe SSD와 같은 고속 저장 장치를 하위 계층 캐시로 구성하여 자주 호출되는 데이터에 대한 접근 속도를 데이터 유형별로 최적화할 수 있습니다.

4) **전용 클라우드 서비스 API 제공**

클라우드 서비스는 주로 제공업체가 인프라와 설계를 주도하기 때문에 사용자가 원하는 작업 목적에 맞춰 클라우드 환경을 세팅할 수 있도록 전용 API를 제공하는 것도 중요한 개선 방안입니다. 클라우드 서비스 API로 제공할 수 있는 설정으로는 아래와 같은 기능들이 있습니다.

- 사용자 맞춤형 환경 구성 : 사용자가 선택한 작업 목적에 따라 클라우드 환경을 재구성한다. 예를 들어 AI모델 훈련을 위한 GPU 최적화나, 빅데이터 분석을 위한 spark 클러스터를 생성하도록 환경을 재구성한다.
- 워크로드 기반 설정 최적화 : 사용자가 데이터의 유형을 지정하면, 데이터의 저장 방식과 복제 전략을 데이터 유형에 알맞게 자동으로 설정한다.
- 인프라 자동 스케일링 : 사용자가 작업 외의 클라우드에서 발생하는 작업부하를 신경 쓰지 않도록, 워크로드가 증가할 경우 자동으로 인프라를 확장 및 축소시켜 작업의 진행에 문제가 생기지 않게하며, 자원의 불필요한 소모를 줄인다.

이러한 API는 클라우드 사용자의 생산성을 높이고, 작업 환경을 단순화하며, 다양한 작업 목적에 맞는 클라우드 환경을 손쉽게 구성할 수 있도록 돕습니다. 이를 통해 클라우드 사용자는 운영 부담을 줄이고, 본연의 작업에 집중할 수 있습니다.

5.4 현대 사례와의 비교

1) **워크로드 맞춤형 데이터 복제**

- AWS 사례: Amazon Redshift는 데이터 쿼리 시 병렬 처리를 통해 빠른 성능을 제공하지만, 쿼리가 빈번히 발생하는 데이터의 동적복제 및 재배치 기능은 제공하지 않는다.
- Google 사례: Bigtable은 분산 데이터베이스로 높은 처리 성능을 제공하지만, 데이터 복제는 주로 수동 설정에 의존한다.
- Google Cloud 사례: Persistent Disk는 데이터 복제를 통해 가용성을 확보하지만, 데이터 접근 빈도에 따라 동적으로 복제본을 재배치하거나 삭제하는 기능은 지원하지 않는다.

제안된 개선 방안

AI/ML 및 빅데이터 워크로드 특성을 실시간으로 분석하여 다음과 같은 이점을 제공합니다.

- 핫 데이터 관리: AI/ML 작업에서 반복적으로 호출되는 데이터는 "핫 데이터"로 식별되고 작업 노드 근처로 동적으로 복제된다.
- 동적 복제 최적화: 빅데이터 애플리케이션에서는 실시간 스트리밍 데이터 요청을 분석하여 데이터 복제본을 자동으로 생성 및 재배치한다.

제안된 구조는 이러한 데이터를 워커 노드 근처로 복제하여 데이터 접근 시간을 단축하고, 모델 훈련 속도를 향상시킵니다. 기존의 정적인 복제 정책에 비해 훨씬 유연한 방식입니다.

2) **데이터 유형 맞춤 저장**

- Google Cloud Storage는 데이터를 Standard, Nearline, Coldline, Archive와 같은 스토리지 클래스로 분류하여 저장 비용을 최적화한다.
- AWS S3는 S3 Intelligent-Tiering을 통해 데이터 접근 빈도에 따라 스토리지 계층을 자동으로 변경합니다. 그러나 이러한 시스템은 데이터 유형에 따른 저장 최적화를 세분화하지는 않는다.

제안된 개선 방안
데이터 유형에 따라 저장 방식을 차별화하여 저장공간과 접근속도를 동시에 최적화합니다.
- AI/ML 데이터: 이미지 및 동영상과 같은 비구조화 데이터는 NVMe SSD와 같은 고속 저장 장치에 저장하고, 효율적인 압축 알고리즘을 적용하여 저장공간을 절약한다.
- 빅데이터 스트리밍 로그: 기존 블록 저장방식을 개선하여 페이징 기법을 적용, 데이터 순서를 유지하며 저장 효율성을 높인다.

Google Cloud의 Coldline Storage는 장기 저장 데이터에 적합하지만, AI 모델 훈련 중 빈번히 호출되는 이미지 데이터를 고속 NVMe SSD로 저장하는 제안된 설계가 데이터 접근 속도에서 더 우수합니다.

3) **실시간 데이터 접근 최적화**

- AWS Elastic File System(EFS)와 Google Filestore는 네트워크를 통해 데이터를 공유하고 고속 데이터 접근을 지원하지만, 네트워크 병목 현상이나 대규모 데이터 전송 시 발생하는 지연 문제를 해결하지 못한다.
- Google 사례: Filestore는 NFS 기반의 파일 공유를 지원하지만, 대규모 병렬 데이터 요청 처리에는 한계가 있다.

제안된 개선 방안
RDMA를 도입하여 메모리 간 직접 데이터 전송을 가능하게 하며, 대규모 데이터 전송의 지연 시간을 최소화합니다.
- 계층적 캐싱 구조: RAM과 NVMe SSD를 활용하여 빈번히 호출되는 데이터를 최상위 계층(RAM)에 저장하고, 덜 자주 접근되는 데이터를 하위 계층(NVMe SSD)에 배치한다.
- 데이터 전송 최적화: RDMA 기술을 통해 데이터 접근 속도를 최대화하며, 특히 AI/ML 모델 훈련과 같은 실시간 워크로드에 최적화된다.

Google Filestore는 병렬 요청 처리 시 속도 저하를 경험할 수 있지만, 제안된 설계는 RDMA 기반 데이터 전송은 대규모 데이터를 실시간으로 처리하여 AI/ML 및 빅데이터 작업에서 탁월한 성능을 제공합니다.

5.5. 향후 발전 가능성

1) **사용자 작업 기반 설정 학습 및 맞춤 제공**

클라우드 사용자는 일반적으로 자신에게 최적화된 환경을 직접 구성해야 하며, 이는 시간과 비용을 소모하고 작업 효율성을 저하시킬 수 있습니다. 이러한 문제를 해결하기 위해 클라우드 서비스 API가 다양한 사용자의 작업을 학습하여, 최적의 설정을 자동으로 제공하는 기능이 필요합니다.

2) **AI 모델의 복잡성 증가와 DFS의 대응**

AI/ML 애플리케이션의 발전은 점점 더 큰 규모와 복잡성을 요구합니다. 대규모 언어 모델(예: GPT-4, GPT-5)은 수십억에서 수백억 개의 매개변수를 학습하며, 이러한 모델을 훈련하고 배포하는 데 DFS의 역할은 더욱 중요해집니다.

대규모 언어 모델의 훈련에 필요한 데이터셋의 크기가 수십 테라바이트에서 페타바이트 단위로 증가함에 따라 기존보다 더 대용량의 데이터를 수용할 수 있어야 하며, 이 대용량 데이터에 대해 수백 번의 반복적인 접근이 필요하기에, 단순한 데이터 복제로는 감당하기 힘들어 질 것입니다.

3) **엣지 컴퓨팅과의 통합**

엣지 컴퓨팅은 데이터가 생성되는 위치에서 가까운 노드에서 데이터를 처리하는 방식으로, 실시간 데이터 처리를 요구하는 애플리케이션에 적합합니다. 클라우드 환경에서 중앙 집중식 DFS와 엣지 컴퓨팅의 분산 처리 방식을 통합하면 데이터 처리 속도와 효율성을 크게 향상시킬 수 있습니다. 데이터를 엣지에서 1차적으로 처리한 후, 중요 데이터를 중앙 클라우드로 전송하여 후속 분석을 수행하는 협력구조를 가지는 형태로 발전할 수 있습니다.

5.6 제안된 개선 아이디어의 한계

제안된 아이디어를 실현하려면 RDMA 기반 네트워크를 도입하려면 전용 네트워크 인터페이스 카드(NIC), 스위치, 및 네트워크 설계 변경 등이 필요하며, 이는 초기 도입 비용을 증가시키게 됩니다. 특히, 클라우드 환경에서 대규모 분산 파일 시스템(DFS)을 지원하는 인프라에 RDMA를 적용하려면 모든 노드에 RDMA 지원 하드웨어를 설치해야 하므로 초기 투자 비용과 운영 복잡도가 급격히 증가할 수 있습니다.

해결 방법으로는 모든 노드에 RDMA를 즉시 적용하기보다는, 초기에는 데이터 접근이 잦은 워크로드 또는 고성능이 필요한 노드에만 우선적으로 RDMA를 도입하여 점진적으로 확장하는 방법이 있습니다.

6. 결론 및 느낀점

6.1 결론

클라우드 컴퓨팅은 AI/ML 및 빅데이터 애플리케이션의 확산과 함께 그 중요성이 점차 커지고 있습니다. 그러나 기존의 분산 파일 시스템(DFS)은 Batch 처리 중심으로 설계되어 있어 실시간 데이터 접근 및 고성능 데이터 처리를 요구하는 현대 워크로드를 충분히 지원하지 못하고 있습니다. 이러한 한계를 극복하기 위한 개선된 DFS 설계를 제안하였으며, 이를 통해 클라우드 환경에서 AI/ML 및 빅데이터 애플리케이션의 요구를 효과적으로 충족시킬 수 있는 방안을 모색했습니다.

제안된 DFS는 워크로드 맞춤형 데이터 복제 및 배치, 데이터 유형에 따른 저장 방식 최적화, 실시간 데이터 접근 최적화, 그리고 사용자 맞춤형 클라우드 서비스 API 제공을 주요 개선점으로 합니다. 현대 사례와의 비교한 결과 기존 시스템은 데이터 접근 속도와 복제 정책의 유연성에서 제안된 DFS에 비해 제한적인 성능을 보입니다.
물론 제안된 설계는 RDMA와 같은 고급 기술의 초기 도입 비용 증가와 같은 한계를 가지고 있습니다. 그러나 단계적 도입 방식을 통해 이러한 문제를 완화할 수 있으며, 새로운 기술의 효과를 검증하고 안정적으로 확장할 수 있는 현실적인 방안을 제공합니다.

결론적으로, 제안한 DFS 설계는 현재와 미래의 클라우드 환경에서 실시간 데이터 처리와 효율성을 극대화할 수 있는 강력한 기반을 제공합니다. 이는 AI/ML 및 빅데이터 애플리케이션의 발전에 기여할 뿐만 아니라, 클라우드 서비스의 성능, 확장성, 사용자 경험을 전반적으로 향상시킬 것입니다.

6.2 느낀점

한 학기간 동안 운영체제에 대해 공부하며, 수업에서 주어진 프로젝트 과제를 수행하기 위해서는 강의 내용을 깊이 이해하고 전반적인 개념을 완벽히 소화해야 가능하다는 것을 느꼈다. 이 에세이 또한 학기중 배운 내용들을 바탕으로 작성하는 보고서였기에, 가장 미흡했던 파일시스템을 주제로 하여 작성하기로 마음먹었다. 다만 파일 시스템과 관련된 내용을 잘 알지 못했던 탓에, 어디에 가장 잘 활용되고 있는지 고민하던 중 클라우드 컴퓨팅이라는 분야가 떠올랐다. 최근 들어 클라우드가 최근에는 기본적으로 요구된다는 현직자 분의 말을 들었던 것도 관심을 끌기에 충분했고, 다음 학기에 학과에서 개설되는 클라우드 컴퓨팅 과목을 수강해볼까 고민 중이던 터라, 예습한다는 마음으로 가볍게 주제를 정하게 되었다.

클라우드 컴퓨팅을 주제로 정한 후 기본적인 개념부터 차근차근 정리하며 보고서를 작성하기 시작했는데, 생각보다 다룰 내용이 너무 많아 처음에는 막막함을 느꼈다. 클라우드 환경의 운영체제에서 가장 중요한 요소 중 하나인 가상 메모리와 분산 파일 시스템에 집중하기로 결정했다. 두 개념을 구조적으로만 파악하고 이를 중심으로 보고서를 작성하면서,

보고서의 목표인 클라우드를 개선하기 위한 아이디어를 제안하고 구체화 하였다.

보고서의 주제를 '실시간 분산 파일 시스템 설계'로 정하게 된 계기로는 기본적인 개념들을 정리하던 중, 개념 정리 중 DFS가 Batch 시스템에 더 적합하다고 하였기 때문이다. 몇 년 전에 한창 게임이 요구하는 컴퓨터 성능이 한창 올라갈 때 'NVIDIA GeForce NOW'라는 서비스를 접하게 되었는데, 이 서비스는 개인 컴퓨터의 성능을 개선하지 않고도 고사양 게임을 가능하게 해주는 기술로, 클라우드 서버에서 게임을 실행하고 사용자는 단순히 명령어와 화면을 주고받는 방식이라 클라우드 컴퓨팅과 같은 개념으로 실행되던 서비스라 게임플레이를 지원할 정도면 실시간 분산 파일 시스템이 가능해야 한다고 생각했기 때문이다.

설계를 구체화하면서 느낀 점으로는 기본적인 클라우드 컴퓨팅에서의 설계는 나날이 증가하는 데이터양을 감당하기 위해 대규모 데이터를 저장하는 것에 집중적으로 치우쳐져 있다고 느꼈고, 데이터를 사용하기 위한 실시간 처리 능력은 다소 떨어진다고 느껴져 적절한 주제를 고른 것 같다고 생각이 들었다. 개인적인 생각으로는 클라우드 서비스를 제공하는 업체들의 미래 주요 고객들은 AI/ML과 빅데이터 작업자들이기에 작업별 API 세팅으로 사용자들의 편의성을 극대화하여 산업에서 경쟁우위를 가질 것으로 생각된다.

구현하면서 아쉬운 것은 내가 제안한 아이디어는 수업 중 배운 페이징 기법이나, 캐시 메모리의 관리, 스케줄링 알고리즘 같은 것들을 기반으로 활용하는 구획만 바꾼 느낌이라 이미 구현이 되어있을거 같다는 생각이 들었는데 이미 존재하는지 확인해보지 못한 것이 아쉬웠다. 그리고 현재 클라우드를 운영하고 있는 업체들이 미래에 주 고객이 될 AI와 빅데이터 처리에 효율적인 기능들이 일반 운영체제에 대비해 제한적으로 구현되어있어, 아직은 발전 가능성이 높은 산업이라는 생각이 들었다.

7. 참고문헌

* 2.1 클라우드 서비스 모델 비교
https://www.redhat.com/ko/topics/cloud-computing/iaas-vs-paas-vs-saas

* 2.1 클라우드 배포 모델 비교
https://velog.io/@hidaehyunlee/IBM-Cloud-%ED%81%B4%EB%9D%BC%EC%9A%B0%EB%93%9C-%EC%84%9C%EB%B9%84%EC%8A%A4-%EB%AA%A8%EB%8D%B8-%EB%B0%8F-%EB%B0%B0%ED%8F%AC-%EB%AA%A8%EB%8D%B8

* 2.3 하이퍼바이저 사진 및 개념
https://selog.tistory.com/entry/%EA%B0%80%EC%83%81%ED%99%94-%EA%B0%80%EC%83%81%EB%A8%B8%EC%8B%A0VM%EA%B3%BC-%ED%95%98%EC%9D%B4%ED%8D%BC%EB%B0%94%EC%9D%B4%EC%A0%80-%EC%89%BD%EA%B2%8C-%EC%9D%B4%ED%95%B4%ED%95%98%EA%B8%B0

* 2.3 컨테이너 사진
https://nearhome.tistory.com/83

* 3.3 NUMA 개념 및 사진
https://brunch.co.kr/@dreaminz/4

* 3.4 Balloning
https://velog.io/@pmw1130/ballooning

* 4. 분산 데이터 저장 기술
https://sinpong.tistory.com/134

* 4.1 분산 파일 시스템
https://www.nutanix.com/kr/info/distributed-file-systems

* 4.4 수직/수평 확장
https://btcd.tistory.com/16