특집 03

고성능 컴퓨팅 환경 구축을 위한 클라우드 기반 단일 시스템 이미지 기술

송충건 (이노그리드)

목 차

1. 서 론

최근 학계에서 인공 신경망 기반 AI(Artificial Intelligence) 기술의 성능이 검증되고 Google DeepMind의 AlphaGo를 통해 실효성이 알려지면서 AI에 대한 관심이 폭발적으로 증가하였다[1]. 이에 따라 AI를 다양한 분야에 활용하기 위한 시도들이 가속화되고 있으며, TensorFlow와 같은 SW 개발 라이브러리, AI 연산 속도 향상을 위한 H/W인 TPU(Tensor Processing Unit), NPU(Neural Processing Unit) 등 다양한 레벨의 기술들이 등장하고 있다. 또한 AI 연산이 요구하는 대규모 병렬처리 시스템을 신속하고 손쉽게 구축하기 위한 HPC(High-Performance Computing) 목적의 클라우드 기반 IaaS 서비스가 각광받고 있다. 클라우드 서비스 제공자 입장에서 HPC 목적의 IaaS 서비스를 제공하기 위해선 서비스 단위인 VM에 할당하는 컴퓨팅 자원의 양을 대규모로 확장해야 한다. 여기서 일반적인 HPC 시스템의 자원 확장과 동일하게 자원 확장의 유형을 선택하는 이슈가 발생한다. 이러한 컴퓨팅 자원 확장은 일반적으로 Scale-Up 방식과 Scale-Out 방식으로 구분된다.

Scale-Up 방식은 단일 머신의 하드웨어를 고사양 시스템으로 교체하거나 추가 장치를 장착하는 방식을 말한다. 이는 기본 자원 확장 작업에 추가적으로 발열이나 데이터 접근 속도 문제를 고려해야 한다. 최근 OCP(Open Compute Project)와 같이 서버 설계에 대한 노하우를 오픈소스 형태로 공유하는 프로젝트가 생기면서 Scale-Up 방식에 대한 안정성이 높아지고 있다. Scale-Up 방식은 하나의 머신에 하나의 운영체제가 동작하기 때문에 확장된 자원을 활용하기 위한 복잡한 라이브러리나 프로그래밍 기법이 요구되지 않는다. 따라서 기존 환경에 맞게 작성된 어플리케이션을 소스코드 레벨에서의 수정

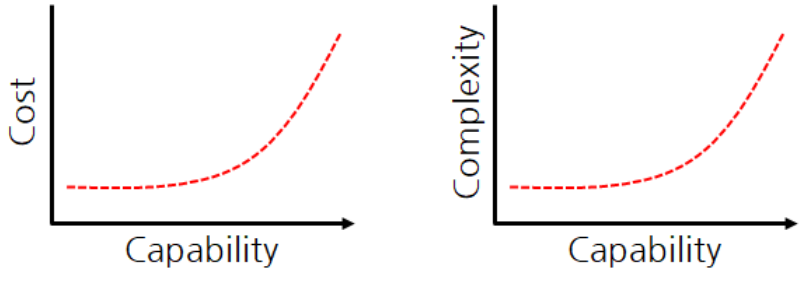
없이 운영 가능한 장점이 있다. 그러나 Scale-Up 방식은 단일 머신을 확장하면서 생기는 다양한 하드웨어 이슈를 처리하기 위해 추가적인 장치나 어플리케이션이 요구되면서 가격이 기하급수적으로 증가한다.

Scale-Out 방식은 LAN(Local Area Network) 환경에서 고속 네트워크로 다수의 물리 머신을 연결하여 자원을 확장하는 방식이다. 이는 저사양의 안정적인 물리 머신들을 연결하면서 자원 확장으로 야기되는 이슈가 상대적으로 적다. 그리고 기존 유휴 시스템을 연결하면서 시스템 구축비용도 절약된다. 또한 물리 머신들의 역할을 분리하고 유동적으로 자원을 On/Off 하면서 대규모 자원 풀 관리의 편의성을 제공한다. 그러나 물리 머신의 수가 증가하면서 시스템의 복잡도와 관리에 대한 오버헤드가 증가한다. 또한 이를 활용하는 사용자 입장에서 추가된 자원을 효율적으로 활용하기 위해 소스코드 레벨에서 비종속적 병렬연산을 분리하고 클러스터에 적절하게 분배해야하는 추가 작업이 요구된다. 이러한 작업은 시스템의 구성과 단일 머신이 보유한 하드웨어 유형에 따라 성능의 좌우되므로 분산 시스템에 대한 전문 지식이 요구된다. (그림 1)에서는 컴퓨팅 자원 확장 시 각 유형에 따라 나타나는 단점을 그래프로 나타내고 있다.

앞서 살펴본 두 가지 컴퓨팅 자원 확장 유형의 단점들을 해결하기 위해 시스템 복잡도를 최소화 하고 이용에 대한 편의성을 제공하면서 동시에 시스템 구축비용을 절약할 수 있는 단일 시스템 이미지 기술이 있다. 본 고에서는 클라우드 환경에서 HPC 목적의 IaaS 서비스 제공에 적합한 단일 시스템 이미지 기술을 소개하고 이를 클라우드 컴퓨팅 환경에 도입한 시스템 구조에 대하여 소개한다.

2. 단일 시스템 이미지

단일 시스템 이미지(Single System Image) 기술은 이종 분산 시스템의 리소스들을 투명하게 숨기고 통합된 하나의 컴퓨팅 리소스 형태로 사용자나 어플리케이션에게 보여주는 시스템의 특징으로 정의된다[2]. LAN 환경에서 고속 네트워크를 기반 클러스터로 구축된 시스템에서 분산 운영체제나 미들웨어 형태로 구현된다. 최근 기가비트 이더넷과 인피니밴드의 기술의 발전으로 단일 시스템 이미지의 단점인 네트워크 오버헤드가 획기적으로 개선되었고 이로 인해 단일 시스템 이미지는 새로운 국면을 맞이하고 있다. 커널 레벨의 단일 시스템 이미지를 기준으로 핵심 기능은 아래와 같이 3가지로 요약된다.



(a) Scale-Up 성능 확장 이슈 (b) Scale-Out 성능 확장 이슈

(그림 1) HPC를 위한 자원 확장에서 발생하는 이슈

○ 편리한 사용 : 일반적인 응용 어플리케이션을 소스코드 수정 없이 실행하는 것이 가능하다. 유닉스 계열 시스템에서 사용하는 프로세스 관리 기법들도 동일하게 적용되며, ps, top 같은 사용자 영역 명령어도 기존과 동일하게 사용 가능하다. 이를 통해서 사용자는 클러스터 기반 병렬처리 기법에 대한 전문 지식 없이 익숙하게 사용해온 방식으로 프로세스를 실행하여 HPC 수행이 가능하다.

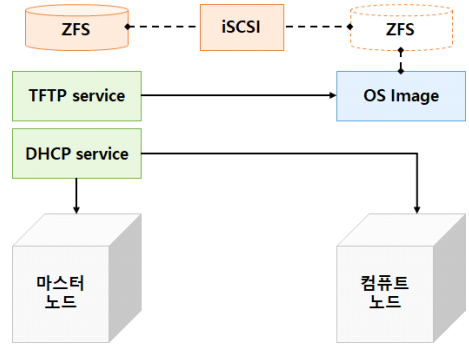
○ 확장성 : 물리 머신을 스위치에 연결하는 방식으로 손쉽게 확장이 가능하다. 단일 시스템 이미지 기술이 구현되는 커널 레벨에서 네트워크 프로토콜 기반 hot plug 기능을 이용하여 물리적 환경 변화를 인지하고 자원을 병합한다. 그리고 IPMI 시스템의 물리 머신 제어 기능을 이용하여 자동화 수준을 높일 수 있다.

○ 고가용성 : 단일 시스템 이미지는 물리 머신 단위로 자원을 탄력적으로 관리하는 기술이 가능하며, 하나의 머신에서 장애가 발생할 경우 자원 병합이 자동적으로 해제되고 나머지 머신의 컴퓨팅 자원을 기반으로 Fail-stop 없이 서비스를 정상적으로 운영 가능하다.

아래 하위 섹션에서는 먼저 단일 시스템이 구현되는 시스템 구조인 Beowulf Cluster에 대하여 설명하며 단일 시스템 기술을 구성하는 세부 기술인 Process Migration과 Checkpoint/Restart 기법에 대하여 소개한다.

2.1 Beowulf Cluster

Beowulf Cluster는 미국 NASA의 고다드 우주비행센터(GSFC)에서 기성 PC들을 결합한 병렬 컴퓨터를 만드는 프로젝트에서 탄생하였다[3]. 클러스터 환경에서 과학적 연산을 수행하는



(그림 2) Beowulf Cluster 예시

멀티 컴퓨터 구조를 가지며, 컴퓨팅 노드들이 보조기억장치 없이 CPU와 Memory를 기반으로 마스터 노드에서 제공하는 운영체제 이미지로 부팅하는 구조이다. (그림 2)에서는 Beowulf Cluster의 예시 시스템을 나타내고 있다.

이러한 클러스터는 기본적으로 PXE 부팅을 기반으로 한다. 마스터 노드가 PXE 부팅의 서버 역할을 수행하고 컴퓨트 노드들이 PXE 부팅의 클라이언트가 된다. PXE 부팅을 위해선 클라이언트의 NIC 카드와 바이오스에서 PXE 기능을 지원해야 한다. 그리고 마스터 노드에서 사전에 PXE 부팅에 활용되는 운영체제 이미지가 준비되어 있고 DHCP 서버를 통해 IP 할당을 수행해야 한다. 또한 운영체제 이미지 전송을 위해 TFTP 서비스가 동작하고 있어야 한다.

그리고 클라이언트는 부팅 시 자체 보드에 연결된 저장소 대신 NFS나 iSCSI를 통해 제공되는 원격 저장소에 저장된 운영체제 이미지를 우선적으로 활용한다. 운영체제를 메모리에 올려놓은 후 루트 파일 시스템을 원격 저장소에 구성된 파일 시스템을 참조한다. 이는 마스터 노드에서 사전에 설정된 부팅 파라미터를 통해 설정된다. 이러한 구조를 통해 컴퓨트 노드가 HPC 작업 수행 도중에 특정 물리 머신에서 Fail-Stop 에러가 발생할 경우 데이터 손실 없이 동일한 운영체제

이미지와 파일 시스템을 다른 컴퓨트 노드에 부팅하여 신속하게 복구할 수 있다.

2.2 Process Migration

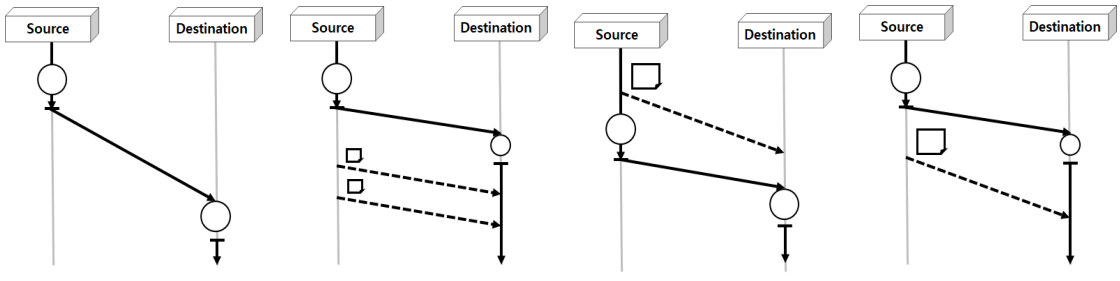
단일 시스템 이미지 기술은 원격에 위치한 다른 컴퓨팅 노드에 새로운 프로세스를 생성하거나, 이미 실행중인 프로세스를 이주하여 부하를 분산한다. 여기서 프로세스 이주는 동일한 네트워크 내 물리적으로 떨어진 다른 노드에 프로세스를 이동시키는 기법을 말한다. 서비스 워크로드를 가지는 실행중인 프로세스도 종료 없이 실시간 프로세스 이주를 통해 부하 분산이 가능하다[4]. 컴퓨트 노드들은 주기적으로 컴퓨팅 자원 사용량 정보를 측정하여 공유 파일 시스템에 저장하고 로드 밸런싱 기법을 통해 부하를 분산하여 클러스터 전체 자원 사용률을 높인다. 프로세스 이주는 이주 대상 선정, 중단 정책, 재시작 정책에 따라 다양한 알고리즘이 존재한다[5]. 이러한 알고리즘은 다음 <표 1>에서 간략하게 설명하며, (그림 3)에서 이주 절차에 대한 개념도를 나타내고 있다.

2.3 Checkpoint/Restart

프로세스 Checkpoint/Restart 기법은 프로세스의 실행 상태를 영구적 저장소에 주기적으로 저장한 후 장애 발생 시 이를 기반으로 프로세스를

〈표 1〉 프로세스 이주 알고리즘

알고리즘	설명	이주 절차
Eager Copy	프로세스 전체를 중지하고 프로세스 데이터와 커널 데이터를 모두 전송한 후에 프로세스를 실행하는 기법이다. 가장 단순하여 구현이 쉬우나 전체 이주 시간이 가장 길다.	· 소스 프로세스 suspend 상태로 변경 · 프로세스 전체와 커널 데이터를 목적지로 전송 · 목적지에서 프로세스 생성 · 목적지 프로세스 resume 상태로 변경
Lazy Copy	프로세스 실행에 필요한 최소한의 데이터를 전송하고 목적지에서 프로세스를 신속하게 시작하고, 나머지 데이터들은 필요한 시점에 마저 전송하는 기법이다. 고용량의 데이터를 메모리에 올린 프로세스에 대하여 이주 시 발생하는 프로세스 중지 시간을 최소화 할 수 있다.	· 소스 프로세스를 suspend 상태로 변경 · 프로세스와 커널 데이터 중 실행에 필요한 최소한의 데이터를 목적지로 전송 · 이전 단계에서 전송한 데이터를 기반으로 목적지에서 프로세스를 생성 · 목적지 프로세스 resume 상태로 변경 · 전송되지 않은 데이터들은 참조되는 시점에 전송
Pre-Copy	커널 데이터와 프로세스의 메모리 주소 영역이 이동되기 전까지 소스 프로세스가 suspend 되지 않는 기법이다. 프로세스중지 시간을 최소화 할 수 있으나 전체 이주 시간이 상대적으로 길어진다.	· 소스 프로세스의 모든 메모리 주소 영역이 목적지로 전송 · 전송이 진행되는 중에도 소스에서는 프로세스가 실행됨 · 전송 완료 시 소스 프로세스를 suspened 상태로 변경 · 소스에서 프로세스와 연관된 모든 커널 데이터 정보와 전송 과정 중 변경된 프로세스 데이터를 목적지에 전송 · 목적지에서 프로세스 생성 · 프로세스를 resumed 상태로 변경 시키고 실행을 시작
Post-Copy	프로세스 실행에 필요한 최소한의 데이터만 전송하고 목적지에서 프로세스를 신속하게 시작한다. Lazy Copy와 다르게 나머지 데이터가 전송되기 전에 데이터 조회가 발생할 경우 원격에 있는 데이터를 참조하는 기법을 사용한다.	· 소스 프로세스를 suspend 상태로 변경 · 실행에 필요한 최소한의 데이터를 목적지로 전송 · 이전 단계에서 전송된 데이터를 기반으로 목적지에서 프로세스 생성 · 목적지에서 프로세스를 resumed 상태로 변경 · 실행과 동시에 병렬적으로 소스에 있는 나머지 데이터를 전송 · 나머지 데이터가 전송되지 전에 요청이 발생한 경우 소스에 있는 데이터를 참조



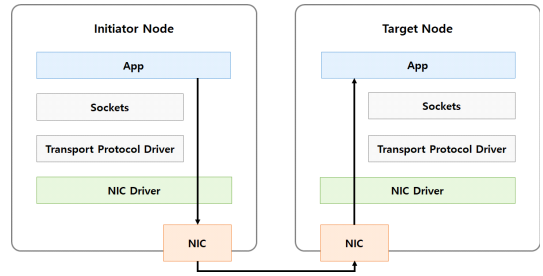
(a) Eager Copy (b) Lazy Copy (c) Pre-Copy (d) Post-Copy

(그림 3) Process Migration 절차

복구하는 기법이다. 오랜 시간 실행되는 프로세스를 대상으로 결함 복구, 부하 분산 목적에 유용한 기술이다[6]. HPC 작업의 방대한 연산에서는 SDC(Silent Data Corruption)이 발생하기 때문에 주기적으로 복제된 N개의 프로세스의 연산 결과를 비교하고 결함 발생 여부를 검증하고 복구해야한다. 이러한 SDC 결함 발생 시 정상적인 실행의 마지막 Checkpoint 단계로 이동하여 Restart를 수행한다. 단일 시스템 이미지에서는 HPC의 기본적인 결함을 복구하는 수단과 함께 프로세스 이주의 보조적인 기능으로도 활용된다. Checkpoint 기법은 프로세스의 실행 상태를 저장하기 때문에 이주할 프로세스를 구성하는 단위로 Checkpoint가 활용된다. 이러한 작업에서는 주소 영역, 레지스터 집합, 오픈된 파일/파이프/소켓, IPC, 워킹 디렉토리, 시그널 핸들러, 타이머, 터미널 셋팅, 사용자 식별자, pid 등이 저장된다.

2.4 RDMA

단일 시스템 이미지에서 원격 메모리 접근 기술 구현 시 RDMA(Remote Direct Memory Access) 프로토콜 활용이 가능하다. CPU를 사용하지 않고 메모리에 상주한 프로세스가 다른 원격 노드의 메모리를 직접 참조하는 기능이다. 이를 통해 CPU 연산이 최소화되고 원격 노드의 호스트 메모리 및 I/O 버스에 대한 경합도 줄여준다. (그림 4)에서는 RDMA를 통한 원격 프로세스의 데이터를 참조하는 구조는 나타내고 있다.



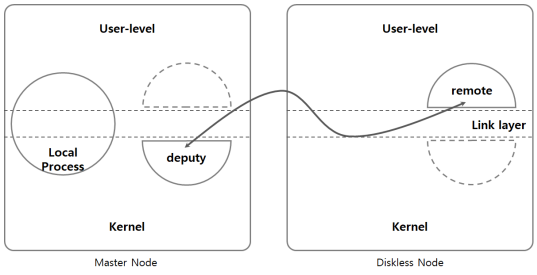
(그림 4) RDMA 동작 구조

3. 단일 시스템 이미지 프로젝트

단일 시스템 이미지 기술이 구현된 오픈소스 프로젝트를 소개한다. 미들웨어 레벨의 단일 시스템 이미지를 대표하는 MOSIX와 커널 레벨의 단일 시스템 이미지를 대표하는 Kerrighed를 기본 구조와 함께 설명한다.

3.1 MOSIX

리눅스 환경에서 적응적 자원 공유를 목적으로 만들어진 오픈소스 단일 시스템 이미지 프로



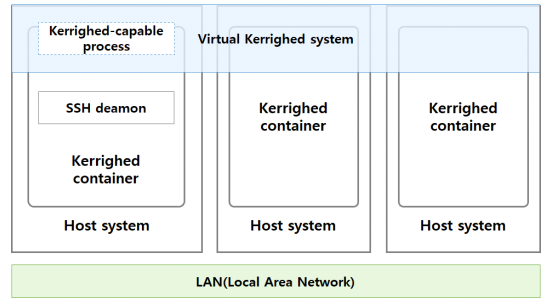
(그림 5) MOSIX 프로세스 이주 구조

젝트이다[7]. 자동적 자원 발견, 선점형 프로세스 이주를 이용한 동적 워크로드 분산, 프로세스 사이의 직접 통신 기술이 적용되어 있다. (그림 5)에서는 MOSIX의 대표적인 기능인 프로세스 이주 기능의 구조를 나타내고 있다. 이주 동작 이후 프로세스는 시작 노드의 deputy와 목적지 노드의 remote 구성요소로 이루어진다. 그리고 클러스터를 구성하는 노드 사이의 통신은 최적화된 TCP/IP을 사용한다. 프로세스 이주 이후에 파일 캐시를 참조하지 못하여 파일을 접근 시 성능이 저하되는 단점을 가지고 있다.

3.2 Kerrighed

Kerrighed는 표준 PC들로 구성된 클러스터 환경에서 SMP 머신에 대한 단일 시스템 이미지 기술을 구현하는 오픈소스 프로젝트이다[8]. 1999년 프랑스 국가 산하 연구소인 INRIA에서 Christine Morin에 의해 시작되었다. 그 후 2006년 IRISA 연구소의 PARIS 프로젝트에서 다양한 연구기관과 함께 발전시켰다. 2006년 이후에는 Kerlabs, INRIA, XtreemOS에 의해 지속적으로 발전하고 있다. Kerrighed의 핵심 기능으로는 ConfigFS를 통해 커스터마이징이 가능한 프로세스 스케줄링 기능이 있다. Kerrighed는 (그림 6)과 같은 구조를 가진다.

오픈소스 단일 시스템 이미지 중 가장 완성도

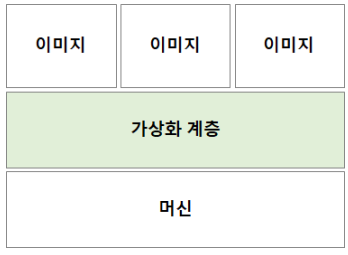


(그림 6) Kerrighed 시스템 구조

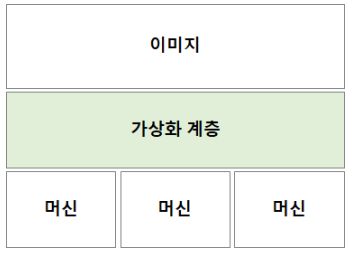
가 높다. 그러나 이는 상용 솔루션으로 발전시키기 위해 많은 기술적 해결과제가 있다. 특히 부하 분산을 위해 원격에 생성한 프로세스가 좀비 프로세스로 변할 경우 이를 처리할 로직이 없어 커널 패닉이 발생한다. 또한 물리 노드를 추가하면서 Scalability가 큰 폭으로 감소하는 성능 측면의 단점을 가지고 있다[9][10].

4. 단일 시스템 이미지와 클라우드 서비스

3장에서는 단일 시스템 이미지 기술의 기본 개념과 해당 기술을 구성하는 세부 기술들을 소개하였다. 이번 장에서는 단일 시스템 이미지 기술을 도입한 클라우드 시스템에 대하여 설명한다. 클라우드 IaaS 서비스는 대규모 컴퓨팅 자원 풀의 유연한 관리를 위해 가상화 기술을 기반으로 한다. 이러한 가상화 기술은 각 단일 머신의 운영체제 커널 내부에 하이퍼바이저를 구축하여 다수의 컴퓨팅 환경을 제공하는 형태가 일반적이다. 그러나 단일 시스템 이미지는 일반적인 가상화 기술의 형태와 반대로 다수의 물리 머신들을 하나로 묶어 하나의 컴퓨팅 환경을 제공한다. 이러한 가상화 형태를 역가상화라고 하며, 일반적인 가상화 기술을 정가상화라 명시하여 구분하였다. 이러한 2가지 가상화 형태를 (그림 7)에서 비교하여 나타내고 있다.



(a) 정가상화



(b) 역가상화

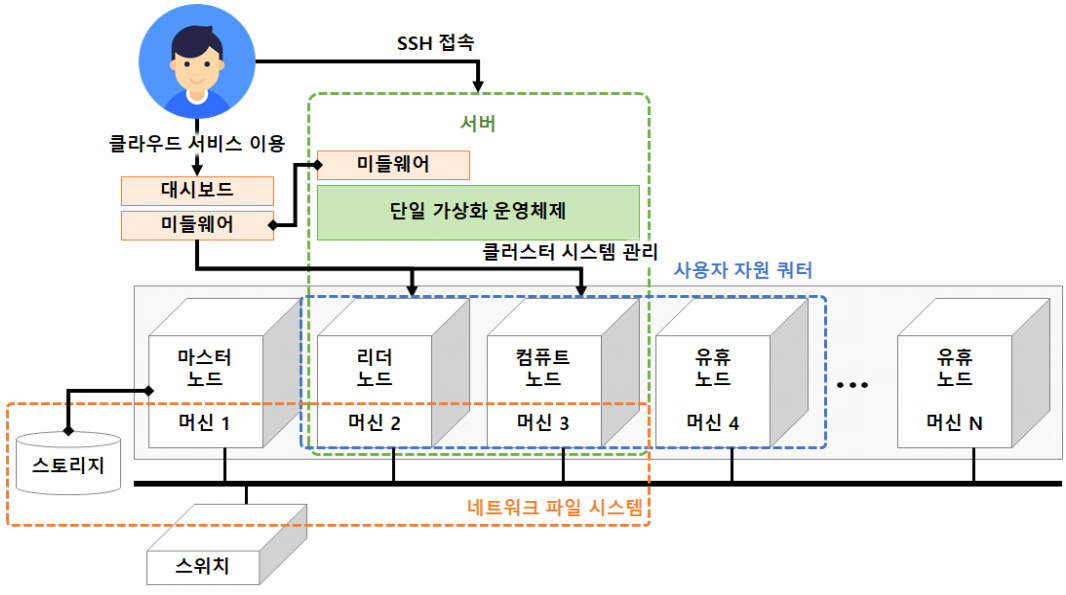
(그림 7) 가상화 유형

단일 시스템 이미지 기술은 기존 가상화 기술과 융합하여 여러 계층으로 구축이 가능하다. 기존 정가상화 계층에서 지원하는 하드웨어 에뮬레이팅 기반 호환성 제공 기능과 VCPU 단위 동적 자원 관리 기법을 활용함과 동시에 단일 시스템 이미지가 가지는 역가상화 개념을 상위 계층에 구성하여 VM의 물리 자원의 벽을 넘어 HPC용 VM을 구성할 수 있다. 아래에서는 이러한 역가상화 개념을 적용한 적용한 클라우드 기반 단일 시스템 이미지 구조와 모니터링 구조를 설명한다.

4.1 클라우드 기반 단일 시스템 이미지 구조

클라우드 기반 단일 시스템은 HPC 연산을 위한 Beowulf Cluster 구조로 동작하기 때문에 모든 물리 머신들이 고속 LAN 환경에서 클러스터로 연결되어 있다. 그리고 PXE 부팅을 위한 다양한 서비스와 설정, 운영체제 이미지가 템플릿 형태로 구성되어 있다. 클라우드 기반 단일 시스템 구조에서 물리 머신의 유형으로 마스터 노드, 리더 노드, 컴퓨트 노드 이렇게 3가지가 있으며, 노드들 사이의 동작 관계와 전체 시스템 구조를 (그림 8)에서 나타내고 있다.

마스터 노드는 클라우드 시스템을 관리하고



(그림 8) 클라우드 기반 단일 시스템 이미지 구조

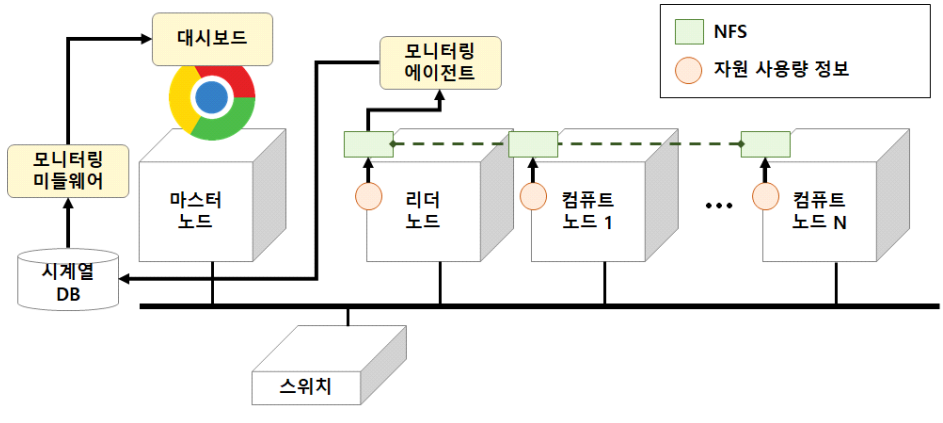
중앙 집중화된 스토리지를 관리하는 역할을 담당한다. 마스터 노드에는 클라우드 서비스 이용을 위한 대시보드와 네트워크, 스토리지, 컴퓨팅 노드의 전원 관리, 모니터링 등 클라우드 서비스 구현을 위한 다양한 서비스가 미들웨어 계층에서 동작하고 있다. 다음으로 리더 노드는 컴퓨트 노드를 대표하는 노드로 연결된 컴퓨트 노드의 컴퓨팅 자원이 병합된 전역 컴퓨팅 환경을 제공한다. 운영체제 내부에서 수행할 명령어를 마스터 노드의 미들웨어 계층의 서비스로부터 전달받아 수행하는 서비스가 동작한다. 마지막으로 컴퓨트 노드는 사용자에게 컴퓨팅 환경을 제공하는 것이 아닌 서버가 지시한 작업을 처리하는 역할만 수행하는 머신이다. GUI 기반 터미널을 제공하지 않고 리더 노드의 명령에 따라 시스템이 활성화되고 리더 노드가 보내는 작업을 처리하여 결과를 네트워크 기반 파일 시스템에 반영하는 동작을 수행한다.

4.2 단일 시스템 이미지 모니터링 시스템 구조

앞서 설명한 클라우드 기반 단일 시스템 이미지 환경을 대상으로 설계한 모니터링 시스템 구조를 (그림 9)에서 나타내고 있다. 컴퓨트 노드들은 네트워크 기반 파일 시스템을 통해 공유되는 파일 시스템에 커널 오브젝트 형태로 시스템 사용량 정보를 저장하고 리더 노드에는 통합된 자원 정보가 보관된다. 리더 노드의 사용자 영역에 운영하는 모니터링 에이전트가 범용적인 모니터링 데이터 수집 기법을 통해 시계열 데이터 저장 목적의 DB에 모니터링 데이터를 저장한다. 저장된 모니터링 데이터는 최종적으로 클라우드 대시보드를 통해 가시화 되어 사용자에게 출력된다.

5. 결 론

본 고에서는 커널 레벨에서 클러스터로 연결된 이종 자원을 투명하게 병합하고 유닉스, 리눅스 계열 응용 S/W를 수정 없이 실행 가능한 환경을 구성하여 Scale-Up 방식과 Scale-Oup 방식의 단점을 모두 해결한 단일 시스템 이미지 기술을 소개하였다. 그리고 단일 시스템을 이루는 세부 기술들과 단일 시스템 이미지가 적용된 완성도 높은 오픈소스 프로젝트에 대하여 알아봤으며, 단일 시스템 이미지를 클라우드 서비스 형태로 구성한 시스템과 모니터링 시스템 구조에 대



(그림 9) 단일 시스템 이미지 환경에서 모니터링 시스템 구조

하여 설명하였다. 이러한 단일 시스템 이미지 기술은 프라이빗 클라우드 산업에서는 저비용으로 손쉽게 HPC용 IaaS 구축을 가능하게 하여 과학적 연산 및 AI 분야의 발전을 도모할 것으로 기대된다. 그리고 퍼블릭 클라우드 산업 분야에서는 NVIDIA와 같이 폐쇄적인 고성능 병렬 컴퓨팅 장치에 대한 오픈소스화가 활성화되고 AI 가속화 목적의 다양한 고성능 병렬 컴퓨팅 장치도 단일 시스템 이미지 형태로 자원 병합이 가능할 것으로 기대된다.

참 고 문 헌

[1] Mikolov, Tomáš, et al. “Recurrent neural network based language model.” Eleventh annual conference of the international speech communication association. 2010.

[2] Buyya, Rajkumar, Toni Cortes, and Hai Jin. “Single system image.” The International Journal of High Performance Computing Applications 15.2 (2001): 124-135.

[3] Becker, Donald J., et al. “BEOWULF: A parallel workstation for scientific computation.” Proceedings, International Conference on Parallel Processing. Vol. 95. 1995.

[4] Shivaratri, Niranjan G., Phillip Krueger, and Mukesh Singhal. “Load distributing for locally distributed systems.” Computer 25.12 (1992): 33-44.

[5] Richmond, Michael, and Michael Hitchens. “A new process migration algorithm.” ACM SIGOPS Operating Systems Review 31.1 (1997): 31-42.

[6] Roman, Eric. “A survey of checkpoint/restart implementations.” Lawrence Berkeley National Laboratory, Tech. 2002.

[7] Barak, Amnon, and Oren La'adan. “The MOSIX multicomputer operating system for high performance cluster computing.” Future Generation Computer Systems 13.4-5 (1998): 361-372.

[8] Morin, Christine, et al. “Kerrighed: a single system image cluster operating system for high performance computing.” European Conference on Parallel Processing. Springer, Berlin, Heidelberg, 2003.

[9] Setiawan, Iwan, and Eko Murdyantoro. “Commodity cluster using single system image based on Linux/Kerrighed for high-performance computing.” 2016 3rd International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE). IEEE, 2016.

[10] Sandhya, K. V., and G. Raju. “Single System Image clustering using Kerrighed.” 2011 Third International Conference on Advanced Computing. IEEE, 2011.

저 자 약 력

송 충 건

이메일 : security0730@korea.ac.kr

- 2015년~현재 고려대학교 일반대학원 컴퓨터학과 박사 수료
- 2015년-현재 고려대학교 분산 클라우드 컴퓨팅 연구실 (ds.korea.ac.kr)
- 2018년-현재 ㈜이노그리드 전문연구요원(www.innogrid.com)
- 관심분야 : Distributed System, Cloud Computing, Virtualization, Single System Image