논문 22-47-08-08 The Journal of Korean Institute of Communications and Information Sciences '22-08 Vol.47 No.08
https://doi.org/10.7840/kics.2022.47.8.1081

LSTM기반 CPU 사용예측을 통한 사설 클라우드 시스템의 다중클러스터 안정성 향상 방안 연구

박 선 철◆, 김 영 한°

An Improvement of Multi-Cluster Stability of Private Cloud Systems through LSTM-Based CPU Usage Prediction

Seon-cheol Park◆, Young-han Kim°

요 약

본 논문은 다수의 기업에서 도입·운영 중인 Cloud Infrastructure as a Service(IaaS) 환경에서 단일 cluster에 소속된 전용 고가용성 자원을 다수의 cluster에서 같이 사용할 수 있는 공유 고가용성 자원으로 활용할 수 있도록 하는 시스템 아키텍처 모델을 설계하고 성능 증명을 통해 유효성을 입증하였다.

본 연구에서는 스토리지 전용 파일시스템 문제를 해결하기 위해 국제 인터넷 표준화 기구(ITEF) 표준규약인 SSH기술 기반 파일시스템을 채용하여 운용중인 환경의 변화없이 적용토록 설계하였으며, 역할기반 Agent로 이기종 cluster 혼용 환경에서도 적용할 수 있는 호환성을 확보하여 특정 솔루션으로 인한 종속성 문제를 해결하였다.

또한 성능 검증을 통해 일반적인 상황에서 약 1시간 이내 서비스 복구 가능성을 증명하여 H/W 파트교체 상황과 대비하여 복구시간을 약 75% 단축할 수 있는 실용성도 입증하였다. 특히 서비스가 migration된 후 대상 cluster에서 동작중인 다른 서비스에 대한 영향 최소화와 migration 이후의 서비스 원복까지의 서비스 연속성을 보장할 수 있도록 Deep Learning 중 Recurrent Neural Network(RNN) 알고리즘으로 cluster의 향후 자원사용 상황을 예측할 수 있도록 하여 최적의 클러스터를 선정할 수 있도록 하였다.

Key Words : Cloud, IaaS, High-Availability, Resource redundancy, Deep Learning, RNN, LSTM

ABSTRACT

In this paper, when building a cluster in a Private-cloud introduced by several companies, and architecture using dedicated available resources configured for HA as a shared virtual cluster that is used jointly by multiple clusters was proposed and designed and verified through performance demonstration. In this study, to solve the problem of the dedicated file system for each vendor, the dependency problem was solved by applying the file system based on the IETF standard technology without changing the current operating environment. In addition, performance tests have demonstrated the practicality of reducing disaster recovery time by approximately 75% compared to demonstrating service recovery within two hours in an environment. In particular, after measuring the variable resource utilization of each cluster to ensure the performance of the service, the optimal cluster for continuity can be selected through the LSTM based on the RNN algorithm.

◆ First Author : Soongsil University Department of IT Convergence, sunpark@soongsil.ac.kr, 학생회원
° Corresponding Author : Soongsil University School of Electronic Engineering, younghak@ssu.ac.kr, 종신회원
논문번호 : 202203-029-C-RE, Received May 2, 2022; Revised May 25, 2022; Accepted May 25, 2022

Ⅰ. 서 론

1.1 연구 배경

기업의 Digital 전환이 가속화 함에 따라 IT서비스 기업뿐 아니라 다양한 업계의 기업들에서도 Cloud 환경을 도입하여 비즈니스 적시성을 달성하는 데 적극적으로 활용하려 하고 있다. 2021년 시장조사기관인 한국 IDC에서 발표한 향후 2025년까지의 한국의 Cloud IT 인프라 시장에 대한 전망 예측에서 매년 평균 15%씩 성장하여 2025년에는 2조 2,189억원의 규모로 매년 지속해서 성장할 것으로 전망하였다.[1] D. K. Yoon은 2019년도 보고서에서 전체 Public-Cloud 시장은 Private-Cloud 시장보다 두 배 이상 발전이 앞서지만, 전체적인 성장세는 Private-Cloud 시장이 훨씬 높다는 것에 주목해야 한다고 기술하였다.[2]

Cloud의 배포(운용)모델은 글로벌 IT 서비스 기업들이 제공하는 Amazon AWS, Microsoft Azure, Google Cloud와 국내 IT 서비스 기업인 Never Business Platform의 NCP, KT의 G-Cloud 등으로 대표되는 Cloud Service Provider(CSP)에서 인프라 소유회사가 제공하는 IaaS 서비스를 이용하는 방식인 Public-Cloud 모델과 기업내 전산센터에 독자적인 Cloud 환경을 구축하는 On-Premise 형태의 Private-Cloud 모델, 공공 클라우드·금융 클라우드처럼 공통의 유사한 요구사항이 있는 조직이 함께 구축하여 사용하는 Community-Cloud 모델, 특정 기업에 맞춤형 네트워크 및 보안 설정을 제공하여 Private-Cloud 환경처럼 사용할 수 있도록 Public-Cloud에서 독립성을 제공하는 Virtual Private-Cloud 모델, 마지막으로 2개 이상의 클라우드 배포 모델을 조합한 Hybrid-Cloud 모델 총 5가지 유형으로 분류할 수 있다.[3]

다수의 기업은 외부 고객을 대상으로 하는 커스터머 서비스는 Public-Cloud 모델을 채용하고 내부 직원용 및 경영관리를 위한 서비스들은 서버 가상화(Server Virtualization)를 지원하는 기술인 Hypervisor에 기반한 Private-Cloud 모델을 적용하는 이원화 전략을 구사하고 있다. 기업에서 Cloud 배포모델을 선택할 때 중요한 요소는 투자대비 효용을 최우선으로 고려하고 있다.

K. H. Choi는 Cloud 배포모델의 장·단점을 비교하며 보안, 다중 소유 문제, 신뢰성과 성능, 서비스 벤더 종속적 기술환경, 기존 투자 장비의 비용 문제 및 거버넌스와 감사 대응 등 6대 항목으로 인해 Public-Cloud로 모두 전환할 수 없는 저해요인을 제시하고 특히, 보안성과 거버넌스 측면에서 Private-Cloud를 구축 운영할 수밖에 없는 환경적 요인에 대해 분석하였다.[4]

기업의 규모에 따라 차이는 있으나 대다수 기업은 Table 1과 유사하게 Tenant 별 특징에 따른 응용 서비스 분류를 적용하여 중요도에 따라 관리하고 있으며, 날로 엄격해지는 기업 내 산업기밀 보호와 정부의 개인정보보호 및 데이터 위치투명성 요구 등의 거버넌스 요구에 맞추기 위해 다수의 Virtualization Clusters(VCs)를 운용할 수밖에 없는 상황에 놓여있다.[5]

다른 관점에서는 투자 비용 절감을 위한 선택이다. 중요도가 낮은 서비스에는 오픈소스(Open Source Software, OSS) Hypervisor를 채용하고 중요도가 높은 응용 서비스에는 상용솔루션으로 구성하는 다원화하는 방식이 비용 절감을 위해 선택되고 있다.

앞서 기술한 내용처럼 기업에서 다수의 VCs를 운영할 수밖에 없는 요인으로 정부의 거버넌스 요구, 기업 내 비즈니스의 Tenant 별 보안 특성, 특정 벤더 종속탈피를 위해 OSS와 혼용하는 기술전략 외에도 CPU의 세대교체나 Hypervisor의 major 버전 업그레이드와 같은 기술의 진보성도 하나의 중요한 원인이 되고 있다.

Private-Cloud 운용모델을 기업 내에서 구축·운용하게 될 때 직면하는 문제는 단일장비의 내결함성(Fault Tolerance, FT) 수준을 넘어서는 상황에 대비한 업무 연속성 계획(Business Continuity Planning, BCP)을 수립하고 이를 만족하는 고가용성(High Availability, HA) 수준을 구현하기 위한 구축비용(Capital Expenditures, CapEx)과 라이선스 유지비 등

Table 1. Class level by application classification (source: Wikibon)

Level	Applications
Class 1	Development, File & Print, Small Data Marts, Small-scale applications & Data Base servers
Class 2	Medium-scale applications & DB servers, Data Marts, Small CRM, Small Data Warehouses, Messaging (Exchange, etc)
Class 3	Mission Critical Applications, Enterprise CRM, ERP, Large scale OLTP, Large scale DB Servers, Large Scale messaging
Class 4	Large-scale applications requiring highest levels of availability, security & recoverability

연간 운영비용(Operating Expenditure, OpEx)이 추가로 발생하는 문제이다.

HA 비용은 고가용성 수준을 높일수록 기업의 투자 규모가 기하급수적으로 증가하게 되는 특성이 있다. HA 수준 대비 투자를 높은 순서로 나열하면 센터 redundancy, 자원 redundancy, 서비스 redundancy 순으로 높은 투자 비용이 요구된다.

본 논문에서는 자원 redundancy를 이용한 HA 수준 구현에 초점을 맞추고 있다. 자원 측면에서 HA를 구현하기 위해서는 전체 가용자산을 하나로 묶는 clustering 기법을 적용하여 VC를 구성한다. 설계단계에서 대상 VC의 소요 자원에 여유율을 추가한 후 H/W 장애에 대비한 host node 단위의 추가 인프라 비용을 반영하게 된다. TTA는 인프라 설계시 시스템 여유율로 1.3을 적용토록 권고하고 있다.[6]

수식(1)은 VC 인프라 설계 시 필요한 전체 host node의 수량($T_{servers}$)을 나타낸다. 필요로 하는 자원을 전체 요구자원(R_{req}), 향후 확장을 위한 여유 자원(R_{ext}), 고가용성 지원을 위한 자원(R_{ha})을 각각 곱해서 산정한다. 이렇게 산정된 자원을 단일서버 기준 제공 가능한 자원(R_{server})으로 나눈 값을 반올림하면 필요한 전체 host node의 수량을 산정할 수 있다.

$$T_{servers} = \partial \frac{(R_{req} \times R_{ext} \times R_{ha})}{R_{server}} \quad (1)$$

이렇게 구축된 HA 자원의 단점은 비용대비 효율성의 문제이다. 고비용의 HA 자원을 도입하였으나 소속된 VC에서만 사용할 수 있는 독점 고가용성(Private High Availability, PHA) 자원으로만 사용할 수 있으며, 장애 등 긴급상황에 대비한 투자로 평소에는 예비자원으로 운영해야 하므로 활용성도 매우 낮을 수밖에 없다. 이러한 특성에도 불구하고 cluster 내에서 발생하는 host node의 예기치 못한 장애대비 또는 계획된 H/W 유지보수 상황에서 해당 cluster 내 서비스의 연속성을 보장하기 위해서는 필수적으로 구축해야 하는 요소이다. 이런 PHA 자원의 특성으로 인해 일부 경영자층에게서는 고비용, 저효율의 낭비 요소로 인식되어 IT 투자 비용 조달 시 걸림돌이 되고 있다.

1.2 연구 목적

본 논문에서는 기업 내 Private-Cloud를 구축하며 도입된 여러 PHA 자원들의 활용성 향상을 목적으로 한다. 각 VC에 소속된 PHA 자원을 VCs들 간에 공동으로 사용할 수 있도록 공유 HA(Shared High Availability, SHA) 자원으로 활용하는 시스템 모델을 제안한다.

연구에서는 동일 벤더사의 Hypervisor 뿐 아니라, 서로 다른 솔루션이 혼용된 이기종 환경의 VCs 환경에서도 적용할 수 있도록 국제표준 및 OSS를 중심으로 한 시스템 아키텍처를 채택하여 전역 VM migration 환경을 구현한다.

특히 Cloud 시스템은 사용량 등에 따라 VM을 자동 확장(Scale-Out) 및 축소(Scale-In)를 수행하는 특성이 있다. 이러한 Cloud 서비스에 대한 이해 없이 VM 전역 migration을 수행하면 오히려 relocation 된 VM으로 인해 타 VC에서 운영 중이던 서비스들의 자원 사용 schedule에 영향을 미칠 수 있다. 따라서, VC 전체의 자원 사용 schedule에 대한 이해를 바탕으로 신중하게 migration VC를 선정하는 방법이 매우 중요하다.

또한, 장애가 발생한 host node의 VM이 relocation 된 후 문제가 해결되어 운영상태로 전환되고 서비스가 다시 relocation되어 정상화 될 때까지 최소 약 2~4시간 이상의 서비스 생명주기(life-time)도 보장해야 한다.

본 연구에서는 이러한 문제의 해결책으로 Deep Learning 알고리즘을 채택하였다. 각 VC의 현시점 이후의 자원 사용계획 예측을 통해 relocation 된 서비스의 life-time을 보장할 수 있는 VC를 선택할 수 있도록 고려하였다. 연구에서 선택한 RNN 알고리즘인 LSTM과 GRU는 다양한 미래 수치를 예측하는 연구가 여러분야에서 활발하게 진행되고 있다. 하지만, 대부분의 연구가 주가, 기온, 부품 수명 예측 등 계단식으로 변화하는 예측에 집중되어 있다.

본 연구는 On-Premise 형 Private-Cloud를 도입하려 할 때 대다수 기업이 직면하게 되는 투자 대비 효율성을 위한 걸림돌 제거에 초점을 맞추고 있다. 특히, 평소에는 유휴 상태로 사용하지 못하던 개별 VC에 속한 HA 자원을 최대한 활용할 수 있도록 하는 PHA 자원의 clustering 구현을 통해 자원 활용률을 증대시켜 고효율성을 달성하는 것이 목표이다.

이를 위해 현재 다수의 기업 운영환경에 신속히 적용할 수 있고 시스템 아키텍처 변화로 인한 적용 문제가 발생하지 않도록 OSS 중심의 아키텍처를 설계하였고, 예측 알고리즘을 적용하여 서비스의 life-time을 보장하는 방안까지 고려한 더욱 진일보한 응용예측에 중점을 두었다.

본 연구를 결과를 활용하면 Private-Cloud 환경의

고가용성 목표 수준을 기존에 도입한 자원을 중점으로 활용한 저비용 구조로 손쉽게 확보할 수 있어, 기업의 Digital 전환을 목표로 하는 IT 전략 가속화를 지원하는 데 유용할 것이다.

1.3 논문의 구성

본 논문 총 3장으로 구성되어 있으며 장별 주요 내용은 다음과 같다. 제1장은 연구의 배경과 목적 및 논문의 구성을 설명한다. 제2장은 본론으로서 관련 연구, 제안 기법과 성능실험에 관해 설명한다.

관련 연구에서는 가상화 고가용성과 관련된 Hypervisor 기술, 고가용성 클러스터 및 가용성 모델, 가상화 migration 그리고 순환신경망에 관해 설명하고, 제안 기법에서는 본 논문에서 제시하는 시스템 아키텍처 모델과 구현 방법을 성능실험에서는 migration 적용 시의 성능 검증과 시뮬레이션 데이터를 이용한 Deep Learning 알고리즘 적용과 그에 따른 평가를 하였다. 제3장은 결론으로 연구 결과에 대한 요약과 한계점, 향후 연구 계획에 관해 서술하였다.

Ⅱ. 본 론

2.1 관련 연구

2.1.1 Hypervisor 기술

Cloud를 구현하기 위해 가장 중요 기술은 서버 가상화를 지원하는 Hypervisor 기술이다. Hypervisor 기술에 대한 분류를 과거에는 전가상화(Full Virtualization), 하드웨어 지원 가상화(Hardware-assist Virtualization), 반가상화(Paravirtualization) 3가지로 구분하여 식별하였으나, 최신 IaaS (Infrastructure as a Service)를 지원하는 기술 분류로는 Table 2처럼 Host 가상화, Hypervisor 가상화와 함께 PaaS(Platform as a Service)를 위한 Container 가상화로 구분한다.

Table 2. Type of Hypervisors

Type	Working Type	Solutions
Host Virtualization	OS(Operation System) 설치 환경위에서 구동하는 가상화	KVM, OpenStack, etc.
Hypervisor Virtualization	OS없이 H/W에 직접 설치하여 구동하는 가상화	VMware ESXi, etc.
Container Virtualization	Container 환경을 사용하기 위한 가상화	Docker, Mesos, etc.

최근 Hypervisor OSS로는 안정적인 OpenStack 외에도 Eucalyptus, CloudStack, OpenNebula 등이 있으며, O. S. Holovnia1, et al, 은 각 OSS Hypervisor의 특징에 대해 비교·분석하였다.[7]

2.1.2 Virtual-Machine migration

Cloud 환경에서 하나의 host node에서 서비스 중인 VMs가 부하분산, H/W 결함, 그리고 계획된 유지보수 등에 의해 대상 VMs를 다른 host node로 옮겨서 서비스를 재개하도록 하는 기술을 VM migration이라 한다. 대다수의 VM migration은 비실시간 VM migration 기법을 사용한다.

비실시간 VM migration 방법은 기존에 실행되는 VM을 정상적으로 시스템을 종료시킨 후 다른 host node에서 해당 VM을 재기동하는 방식으로 다수의 Hypervisor에서 채택되어 널리 활용되고 있는 기술이다. 이 기술을 적용하려면 VM image 파일이 상호 공유되고 있어야 하므로 고비용의 SAN 네트워크로 연결된 host node 사이에서만 활용 가능한 한계점이 있어 VM migration의 source와 target host node가 동일 VC에 위치해야만 사용할 수 있는 단점이 있다.

Fig. 1은 VM migration 상황을 설명하고 있다. 오른쪽 host node에서 서비스 중이던 VM을 왼쪽 host node로 이동시키려고 할 때 공유 스토리지에 저장된 VM image 파일을 연결된 SAN 스토리지 경로를 통해 이동시킨 후 왼쪽 host node에서 VM 서비스를 재개하는 VM migration 방식을 이용한 relocation을 설명하고 있다.

최근에는 기존 서비스의 clone 및 memory copy 기술을 이용한 실시간 migration(Live migration) 기법도 등장하였다. Live migration 방식은 동작 중인 VM의 clone인 target VM을 이동하고자 하는 host node에 임시 loading 후 현재 실행 중인 source VM의 메모리 정보를 복사하여 전달하는 방식이다. Target VM

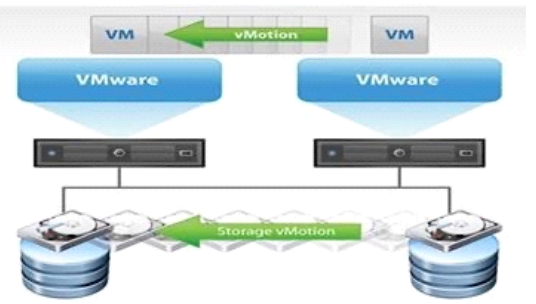

Fig. 1. VM migration in case of failure (Source: VMware)

이 source VM 상태와 완전한 복제 상태가 만들어진 후 source VM을 종료하면 사용자는 마치 서비스가 실시간으로 서비스 중에 이동된 것 같은 현상처럼 느껴지게 된다.[8]

선행연구에서 Live migration을 OSS Hypervisor에 적용하기 위한 연구도 진행되었다. F. Xu, et al, 은 Xen Hypervisor에서 기술적용 방안을 연구하였으며,[9] H. M. Lee는 OSS 계열 Hypervisor에 Live migration을 지원하기 위해 Memory Copy 방식을 활용하여 1차 동기화 후 짧은 시간 동작 중이던 VM을 일시정지 시키고 변화된 부분(dirty page)을 재복사하여 Live migration을 구현하는 방식을 시뮬레이션하였다.[10]

하지만, 앞서 기술한 방식들은 아직 운영환경에 적용할 만큼의 수준에 이르지 못하고 있어, 현재까지 Live migration은 특정 상용솔루션 벤더사의 솔루션에서만 사용할 수 있다.

2.1.3 High-Availability

고가용성은 내결함성(Fault Tolerance, FT)와 함께 서비스의 연속성을 유지하는데 중요한 요소이다. FT는 host node를 이루는 단일장비 수준에서의 결함 허용에 대해, 고가용성은 서버나 서비스 단위의 결함 허용을 다루는 차이가 있다.

AWS는 FT를 시스템 일부 구성요소가 작동하지 않더라도 계속 작동할 수 있는 기능으로 정의하고 있다.[11] 즉, H/W 측면에서 어떤 부품의 결함이 발생할 것 허용하는 기법을 적용하는 것으로 일반적으로는 단일장애점(Single Point Of Failure, SPOF)을 제거하기 위해 여러 부품을 이중화하는 방식으로 구현할 수 있다.

HA는 그 범위를 전체 시스템 관점에서 SPOF를 제거하는 방식으로 장비 이중화, 경로 이중화, 센터 이중화 등으로 정의되는 고비용이 투자되는 방식이다. 이렇게 HA를 구성하면 Table 3처럼 가용성 수준에 따른 시스템 중단 허용 시간 내에 서비스의 연속성을 보장할 수 있으며, 기업들은 상황에 따라 주요 핵심 서비스들을 대상으로 장비 이중화 등을 채택해 99% 이상의 가용성 수준을 유지하고 있다. 하지만, 가용성 수준을 높이는 것에는 많은 비용투자가 요구된다.

Fig. 2는 가용성 수준 증가에 따른 비용증가를 표시한 그래프이다. 가용성 수준을 높임에 따라 투자비용 또한 급격한 우상향 포물선 그래프를 나타내게 되므로 IT전략과 기업의 상황과 필요성에 따른 적정한 수준에서 협상이 필요하다.

Table 3. Failure tolerance time from HA level (source: Wikipedia)

Availability	Downtime	
	Year	Day
90%	36.53 days	2.40 hours
99%	3.65 days	14.40 min
99.9%	8.77 hours	1.44 min
99.99%	52.60 min	8.64 sec
99.999%	5.26 min	864 ms
99.9999%	31.56 sec	86.40 ms
99.99999%	3.16 sec	8.64 ms
99.999999%	315.58 ms	864 usec
99.9999999%	31.56 ms	86.40 usec

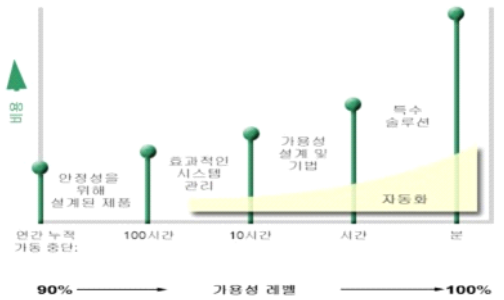

Fig. 2. HA Level and cost (Source: IBM)

HA를 위한 다른 접근에는 네트워크 스위치 차원에서 경로를 전환하는 방식도 연구되었다. 네트워크 경로에서 장애를 감지하고 다른 host node로 경로를 전환하여 서비스를 찾아갈 수 있도록 해준다.[12] 하지만, 연구에서 cluster가 분리된 host node의 이동에 대한 고려 없이, 모든 VC가 단일 cluster로 묶여 있는 경우에만 적용할 수 있어 다중 VC를 운영 중인 기업의 IT 운영 현황에 대한 고려가 부족하여 서로 독립된 cluster 간에는 적용할 수 없는 한계가 있다.

2.1.4 High-Availability cluster

장애 대응을 위한 clustering 기술은 용도에 따라 Table 4와 같이 구분된다.[13] 그중 Cloud 환경을 설계할 때 중요한 기술 요소는 HA cluster의 구성이다.

기업에서 Private-Cloud를 구축한다면 VC에 대한 HA 설계도 포함하여야 한다. 서버 가상화에서의 clustering도 HA cluster의 기술방식과 유사하게 구성하여야 하나 동작방식은 Active-Active로 동작되어야 하므로 더 많은 비용이 소요된다.

설계에 필요한 자원은 VC 내에서 서비스를 동작시

Table 4. A Clustering for Design for Failure (source: NEC)

Type	Comments	App.
Load-Balanced Clustering	하나의 서비스를 위해 다수의 서버가 동작하고 있을 때 L4 등의 부하 분산장치를 통해 부하를 분산하는 방식	WEB/ WAS
High-Availability Clustering	하나의 서버는 Active 상태로 다른서버는 Standby 상태로 대기하며 Active에서 문제 발생시 Standby로 Failover 하는 방식	DBMS

키려는 VMs의 총자원 사용량에 맞추어 서비스 및 서버 redundancy를 위한 자원을 추가로 반영해야 한다. VC를 구성할 host node 수량이 2대 이하라면 추가로 1대의 host node를 추가한 총 3대의 host node로 cluster를 구성하여 예기치 못한 H/W 장애까지 대비한 HA cluster를 구성하는 것이 인프라 설계의 기본 원칙이다.

Fig. 3은 3대로 구성된 기본 VC의 개념도이다. 중앙에 있는 host node에서 장애가 발생하면 해당 host node에서 동작 중인 VMs는 각각 좌우 host node로 relocation 시킨 후 서비스를 재개하도록 하여 최단 시간 중단을 지원하게 된다.

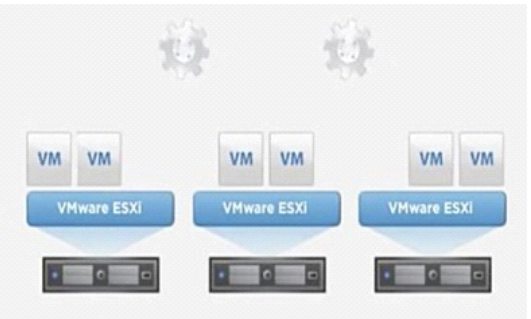

Fig. 3. Virtual Clustering of 3 host nodes (Source: VMware)

2.1.5 RNN

Deep Learning 기술은 다양한 분야에서 예측연구에 활용되고 있다. 본 연구에서 선정한 순환신경망인 RNN 모델은 입력과 출력을 sequence 단위로 처리하는 시퀀스(Sequence) 모델로서 Google translate 같은 언어 번역기에서 그 성능이 검증되어 여러분야에서 널리 연구되고 있다.[14] Vanilla RNN으로 불리는 기본 RNN 모델은 비교적 짧은 sequence에 대해 효과가

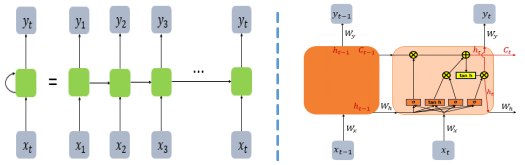

Fig. 4. Compare logic gates between Vanilla RNN(left) and LSTM(right) (source: WikiDocs)

있지만, 장문에는 성능이 떨어지는 장기 의존성 문제(The problem of Long-Term Dependencies)를 가지고 있다.

Fig. 4는 Vanilla RNN와 LSTM을 비교하고 있다. 장기 의존성 문제를 해결하기 위해 LSTM 알고리즘에는 은닉상태(hidden state)를 계산하는 논리게이트를 추가하였으며, 이로 인해 Vanilla RNN보다 긴 sequence 입력을 처리하는데 탁월한 성능을 보여주는 알고리즘이다. LSTM 알고리즘의 연산성능을 개량한 GRU 알고리즘도 널리 활용되고 있다.

장기 의존성 문제 해결을 위해 LSTM과 GRU에 연산 게이트를 추가함으로써 번역기의 성능이 급격히 향상되는 계기가 되었다. 현재는 이러한 RNN의 특징을 시계열을 갖는 수치 데이터 예측에 적용하여 주가, 기상, 온도 등 미래를 예측하기 위한 다양한 분야에서 활용 연구도 활발하게 진행되고 있다. IT 분야에서도 미래에 발생할 장비의 수명 예측을 통해 장애 발생을 사전 예측하기 위한 다각도의 적용이 연구되고 있다.[15-17]

2.2 제안 기법

2.2.1 VR-SAS(Virtualization Resource Shared Availability System)

본 연구는 단일 센터 내의 다중 VC 환경에서 서로 다른 VC의 자원을 SHA로 사용하는 가상화 리소스를 공유하는 가용성 시스템 VR-SAS 모델을 고안하고 실증을 통해 성능을 검증한다. VR-SAS는 특정 VC에서 장애가 발생하면 센터내의 전체 VCs를 탐색하여 최적의 migration 대상 VC를 선정하여 VM migration을 지원하는 시스템이다. 장애로 인해 발생하는 VM의 중단시간 최소화를 달성하고 서비스의 안정성을 보장하기 위한 모델이다.

Fig. 5는 VR-SAS로 SHA를 활용하는 수행 프로세스를 나타낸다. Agent VM은 시스템 상태를 수집하는 monitoring 단계, Manager VM은 각 VCs의 상태를 수집하는 ETL 단계, 수집된 데이터로 Deep Learning을 수행하는 analysis 단계, 마지막으로 장애가 발생한

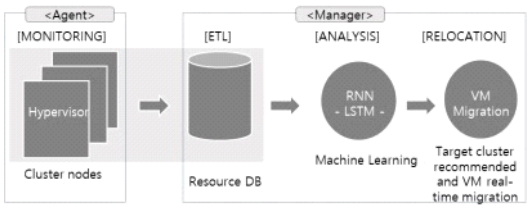

Fig. 5. Process stage of WM migration using VR-SAS

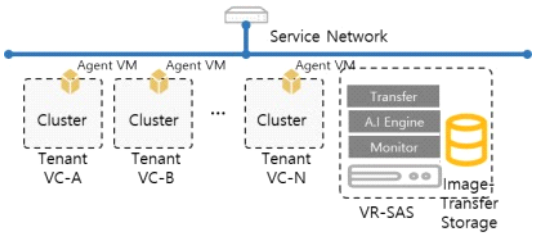

Fig. 6. VR-SAS system architecture

VMs을 재배치하는 relocation 단계로 동작도록 고안하였다.

Fig. 6은 Deploy model을 나타내고 있다. VR-SAS의 Agent VM은 각 VC에 탑재되어 자원 모니터링과 VM image 파일이 위치한 스토리지 자원 접근을 지원한다. Manager VM은 전체 VC의 자원 사용현황을 ETL(Extract, Transform, Load) 하는 Resource Monitor 부, VM 자원을 이동시키는 Image Transfer 부, 최적의 VC를 탐색하는 A.I Engine 부로 구성되어 있다.

Fig. 7에서는 VR-SAS를 구성하는 Manager VM 및 Agent VM의 S/W(Software) 아키텍처이다. 적용용이성을 위해 Virtualization 환경에서 동작할 수 있도록 설계하였으며, OSS를 중심으로 Python 2.7.x와 3.x에서 동작할 수 있도록 module을 구현하였다.

Table 5는 자원 사용현황을 모니터링하기 위해 Agent VM으로부터 수집하는 데이터 구조에 대한 설계이다. VC에서 동작 중인 host node는 각각의 자원 사용에 대하여 주기적으로 사용현황을 기록하고, 이렇

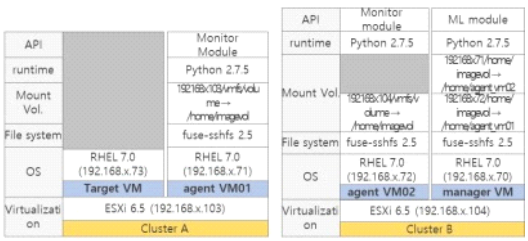

Fig. 7. VR-SAS software architecture

Table 5. Data architecture

Name	Type	Bytes	Remarks
Date	TEXT	10	YYYY-MM-DD
Time	TEXT	8	HH:MM:SS
Hostname	TEXT	64	TEXT
IP addr	TEXT	15	IPv4
Max vCPUs	INT	4	Total cores
Left vCPUs	INT	4	left cores
Max Ram	INT	4	Total Ram size
Left Ram	INT	4	Left Ram
Avg. CPU rate	Float	8	CPU using rate

게 기록된 데이터는 ETL을 통해 Manager VM에서 수집하여 주기적 Deep Learning으로 학습한 후 시간대별 CPU 사용 예측에 활용한다.

2.2.2 Agent VM 구성

Hypervisor에서 Linux OS가 탑재된 Agent VM을 생성한다. Agent VM은 각 Hypervisor 솔루션의 VM 저장소로 활용하는 공유 스토리지 독자적인 Filesystem 문제를 해결하기 위해 SSHfs로 이미지 저장소를 mount 하여 준비한다.

Fig. 8은 Agent VMs을 구성하는 설치 순서와 환경설정값을 설명하고 있다.

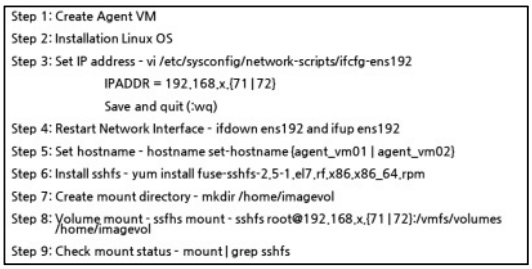

Fig. 8. Configuration guide for Agent VMs

2.2.3 Manager VM 구성

Hypervisor에서 Linux OS가 탑재된 Manager VM을 생성한다. Manager VM은 내장 스토리지의 저장공간과 더불어 각 VC에서 mount하고 있는 SSHfs를 통해 mount된 공유 스토리지 volume을 원격 mount 하여 재배치를 위한 준비를 한다.

Fig. 9는 Manager VM을 구성하는 설치 순서와 환경 설정값을 설명하고 있다. Manager VM은 모든 Agent VM에서 mount 한 VM image 스토리지의 volume을 mount하고 있는 점이 Agent VM과 다른

```
Step 1: Create Agent VM
Step 2: Installation Linux OS
Step 3: Set IP address - vi /etc/sysconfig/network-scripts/ifcfg-ens192
                IPADDR = 192.168.x.70
                Save and quit (:wq)
Step 4: Restart Network Interface - ifdown ens192 and ifup ens192
Step 5: Set hostname - hostname set-hostname manager_vm
Step 6: Install sshfs - yum install fuse-sshfs-2.5-1.el7.rf.x86.x86_64.rpm
Step 7: Create mount directory - mkdir /home/agent_vm01 && mkdir
        /home/agent_vm02
Step 8: Volume mount - ssfhs mount - sshfs root@192.168.x.{71|72}:/vmfs/volumes
        /home/{agent_vm01|agent_vm02}
Step 9: Check mount status - mount | grep sshfs
```

Fig. 9. Configuration guide for Manager VM

특징이다.

2.2.4 System resource monitoring

VC에서 VM relocation으로 인한 영향도를 검토하기 위해서는 각 host node의 CPU 사용 이력에 대한 데이터가 필요하다. 본 연구에서는 개별 host node에 자원 사용현황을 모니터링 할 수 있도록 python 기반의 모듈을 등록하여 사용한다.

시스템 자원 사용률을 수집하기 위해서 python의 psutil library를 활용하여 각 VC의 자원정보를 수집한다. 정보수집은 1분 간격으로 자원 사용현황을 수집하는 방식으로 동작한다. Fig. 10은 psutil library를 이용하여 CPU와 memory의 현황과 해당 시점의 사용량을 수집하는 소스코드의 일부이다.

VC에 등록된 host node에 python과 해당 모듈을 설치한 후 OS의 crontab을 이용해 분 단위로 동작하도록 등록하여 서버에서 ETL하기 위해 적합한 형태의 JSON 형식으로 데이터로 추출한다. 데이터에는 host node를 식별하기 위한 IP주소와 데이터를 추출한 timestamp, 그리고 CPU와 Memory의 총량 및 사용량을 저장한다.

```
# CPU Infomations
cpu_Ghz = round(psutil.cpu_freq().max/1024, 1)
cpu_t = psutil.cpu_count(logical=False)
cpu_u = psutil.cpu_percent(interval=1)
core_t = psutil.cpu_count()

# Memory Information
mem = psutil.virtual_memory()
ram_t = int(mem.total/1024**3)
ram_f = int(mem.available/1024**3)
```

Fig. 10. Source-code for resource monitoring

2.2.5 Analysis algorithm using LSTM

장애로 인한 긴급한 상황에서 VM을 relocation 하려 할 때 쉽게 일으킬 수 있는 운영상의 문제는 현시점에서 자원 사용 여유율이 가장 높은 VC를 단순하게 선택하게 되는 오류이다. 이러한 선택은 자원 사용 가변성이 높은 Cloud의 특징에서 고려해 보면 현시점에서 자원이 여유가 있는 VC보다 장애가 복구되어 서비스를 복원할 수 있는 시간 동안 자원의 가변상황이 안정적인 VC를 선택하는 것이 더욱 중요한 고려사항이다.

VC의 선택은 migration 하여 이동하려는 target VC에서는 장애 시점 이후부터 조치 완료 후 복원할 때까지 최소 2~4시간 동안 안정적으로 운영될 수 있어야 하며, target VC에서 이미 동작 중인 서비스들이 Scale-Out 하려 할 때 migration으로 인한 자원 부족으로 인해 파급 장애가 발생하지 않도록 기존자원의 미래 여유 공간도 확보해야 하는 다방면의 문제까지 고려해야 한다.

VR-SAS는 이런 측면에서 VM을 relocation 하기 전 대상 VC의 현시점 이후부터 복구할 때까지의 CPU 자원 소요에 대한 예측을 제공하며, 미래 자원 사용률에 대한 예측은 VCs로부터 수집된 데이터로 학습한 LSTM 알고리즘을 통해 판단할 수 있도록 제공한다. Deep Learning 분석을 위해서는 Tensorflow, Numpy, Pandas, Keras library를 이용한 python 코드를 사용한다.

첫 번째 과정에서는 데이터를 로딩 후 데이터 전처리(Preprocessing)를 수행한다. Fig. 11처럼 로딩된 데이터 중에서 Deep Learning에 사용할 데이터는 날짜와 시간인 timestamp와 CPU 사용현황 정보이다. 로딩된 데이터에서 연산에 필요한 항목만을 선택하여 Dataframe으로 변환하는 과정을 수행한다.

또한 수치적인 연산을 위해 날짜와 시간 데이터는

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14400 entries, 0 to 14399
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Date      14400 non-null  object
 1   Time      14400 non-null  object
 2   DateTime  14400 non-null  object
 3   CPU_rate  14400 non-null  int64
dtypes: int64(1), object(3)
memory usage: 450.1+ KB
==========================================================================================
         Date      Time            DateTime  CPU_rate
0  2022-01-01  00:00:00  2022-01-01 00:00:00        28
1  2022-01-01  00:01:00  2022-01-01 00:01:00        29
2  2022-01-01  00:02:00  2022-01-01 00:02:00        17
3  2022-01-01  00:03:00  2022-01-01 00:03:00        27
4  2022-01-01  00:04:00  2022-01-01 00:04:00        11
==========================================================================================
[Change data types]
                     CPU_rate
DateTime
2022-01-01 00:00:00      28.0
2022-01-01 00:01:00      29.0
2022-01-01 00:02:00      17.0
2022-01-01 00:03:00      27.0
2022-01-01 00:04:00      11.0
==========================================================================================
```

Fig. 11. Loading resource data and Preprocessing

하나의 field로 묶어 날짜형 변수 타입으로 치환한다. CPU 사용률은 실수형 변수 타입으로 변환하는 처리도 수행한다. 이후 CPU 사용률은 scaler를 통해 데이터를 연산에 최적화되도록 변형하여 사용할 것이다.

두 번째 단계로는 전처리된 데이터가 의도한 데이터인지를 확인하는 단계이다. Test-set으로 사용할 데이터의 이전 자원 사용률 데이터를 그래프로 출력하여 적합한 패턴인지를 확인한다. 본 연구에서는 5일간의 샘플 데이터 중 4일간의 데이터를 Matplotlib library를 이용하여 그래프로 출력하는 과정을 수행하였다.

세 번째는 전체 데이터를 Train-set과 Test-set으로 나누는 단계이다. 이번 연구에서는 Fig. 12처럼 2022년 1월 10일 11시 10분에 장애가 발생한 것으로 정의하고 해당 시점을 기준으로 앞 데이터를 Train-set, 이후 데이터를 Test-set으로 구분하였다. Validation-set은 전체 Train-set 중에서 약 25~30%를 분리하여 모델의 학습 결과검증에 활용해야 한다는 이론에 따라 총 25%의 데이터를 지정하였다.

네 번째로는 데이터 정제를 수행한다. 데이터 연산처리 속도 향상을 위해 0~100까지의 CPU 사용률을 0.0~1.0까지의 실수형 변수로 변형하는 Data-scaling을 수행한다. Data-scaling을 수행하면 수치의 범위가 줄어 연산을 더 빠르게 할 수 있는 장점이 있다.

Fig. 13에서는 이렇게 scaling 된 데이터를 예측을 위한 shift를 정의한다. window size는 shift를 정의하는 데 인자로 사용되며, 정의 후 비어있는 값(NaN)을 삭제하여 오류 발생을 제거하고 예측을 windows size 만큼 앞당기는 데 사용하였다. Dataframe에서 window size를 정의하고 NaN을 삭제하면 불완전한 앞쪽과 뒤쪽의 데이터가 삭제되게 되므로 이를 고려한 충분한 데이터의 확보가 중요하다.

마지막은 처리된 데이터로 Deep Learning을 수행하는 단계이다. RNN 분석을 위한 3D vector 형태로 변환된 데이터를 Fig. 14처럼 LSTM 모델을 로딩하여 optimizer을 적용한 후 fitting 하는 순서로 수행한다.

Epoch는 최적을 탐색하기 위해 1,000부터 시작 후

```
split_date_train = pd.Timestamp('01-07-2022 11:10:00')
split_date_val = pd.Timestamp('01-09-2022 11:10:00')
split_date_test = pd.Timestamp('01-10-2022 11:10:00')

train_set = raw_dataframe.loc[:split_date_train, ['CPU_rate']] # Train set
val_set = raw_dataframe.loc[split_date_train:split_date_test, ['CPU_rate']] # Validation set은 0.25
test_set = raw_dataframe.loc[split_date_test:, ['CPU_rate']] # Test set
```

Fig. 12. Split data to Train-set, Validation-set, and Test-set

```
# 과거 값에서 예측을 위한 shift 정의
# Window는 10 Min
window_size = 10
for s in range(1, window_size+1):
  train_sc_df['shift_{}'.format(s)] = train_sc_df['CPU_rate'].shift(s)
  test_sc_df['shift_{}'.format(s)] = test_sc_df['CPU_rate'].shift(s)

train_sc_df.head()

# Remove invalid value(NaN )
#
X_train = train_sc_df.dropna().drop('CPU_rate', axis=1)
y_train = train_sc_df.dropna()[['CPU_rate']]

X_test = test_sc_df.dropna().drop('CPU_rate', axis=1)
y_test = test_sc_df.dropna()[['CPU_rate']]
```

Fig. 13. Shift define and remove invalid values

```
model = Sequential() # Sequeatial Model
model.add(LSTM(20, input_shape=(window_size, 1))) # (timestep, feature)
model.add(Dense(1)) # output = 1
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['mse', 'acc'])
model.summary()
epochs_count = 1,000
batch_size = 1

# Model fitting
result = model.fit(X_train_t, y_train, validation_split=0.3, epochs=epochs_count,
batch_size=batch_size, verbose=1)
```

Fig. 14. LSTM model fitting

loss function의 fitting 결과를 확인하며 수를 조정하여 최적의 Hyper-parameter를 결정하는 과정이 매우 중요하다. 이와 함께 windows size와 batch size를 조정하며 추가적인 결과를 확인하는 과정을 수행한다.

2.3 성능시험

2.3.1 시험 개요

본 연구의 실증을 위한 성능 검증을 2가지 분야로 나누어 수행하였다. 첫 번째 시험은 H/W 장애시 VM migration을 수행하는 relocation에 소요되는 시간과 H/W 장애 복구를 위한 부품 조달 후 조치 시간인 약 2~4시간과 비교하여 더 효율적인지를 증명하는 장애 복구 성능시험이다.

두 번째 시험은 LSTM을 이용한 CPU 사용률 예측에 대한 효과성을 검증하고 다양한 Hyper-parameter를 적용해 가며 최적의 모델을 탐색하기 위한 Analysis Optimizing 시험으로 실시하였다.

2.3.2 장애 복구 시험환경 구성

장애시험의 시험환경은 Table 6과 같다. Hypervisor 환경은 현재 시험환경에서 운영 중인 VMware 회사의 솔루션 환경에서 실시하였으며 자원에 대한 배정은 일반 WEB 서버 수준인 2 core에

Table 6. Test configurations

Type		Comments	Remarks
Hypervisor		VMware ESXi	ver. 6.5
Guest VM	OS	RHEL	ver. 7.4
	vCore	2 Core	-
	RAM	4 GB	-
	Internal Disk	100 GB	-

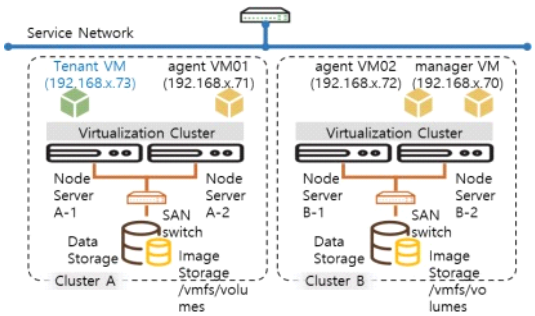

Fig. 15. Network configurations for Testing

4GB ram을 적용한 소형VM 크기로 생성하였다.

내장 디스크는 가상화에서 널리 사용되는 100GB의 OS 영역만 할당하였다. Fig. 15는 테스트를 위한 VC의 시스템 구성 및 각 VM에게 할당된 테스트용 IP주소를 설명하고 있다.

Fig. 16은 VR-SAS를 활용한 H/W 장애 발생에 따른 서비스 복구 시나리오를 나타내고 있다. 복구 시나리오는 Tenant A 클러스터에서 동작 중인 Server #A에서 H/W 장애가 발생하면 해당 클러스터의 Image 스토리지에 저장된 VM image disk 파일을 Server #B에서 동작 중인 Agent VM을 통해 Tenant N 클러스터로 VM migration을 실시하여 서비스를 개시하는 relocation 상황을 나타내고 있다.

Table 7은 장애 복구 시험에서 대상으로 선정한 이

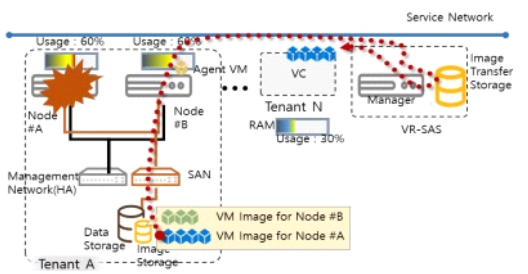

Fig. 16. Service recovery scenario using VR-SAS

Table 7. VM image file size for Testing

VM Size	Comments	Remarks
100GB	OS 100GB	Standard
300GB	OS 100GB + Data 200GB	-
500GB	OS 100GB + Data 400GB	-
1,000GB	OS 100GB + Data 900GB	-

미지 파일의 크기이다. 일반적으로 현장에서 많이 할당하여 사용하는 용량인 100GB의 OS 크기를 기준하고, 추가적인 데이터 스토어를 할당한 300GB, 500GB, 1TB로 총 4가지 데이터 크기를 시험 대상으로 준비하였다.

2.3.3 장애복구 시험 결과

Fig. 17은 데이터 전송에 대한 시험을 Hypervisor 솔루션에서 제공하는 Network traffic monitoring 도구를 통해 측정한 화면이다. 솔루션에서 제공하는 기능은 전송 전 상호 정보를 교환하는 과정이 포함되어 있어 전송속도에만 포커스 하기 위해 OS script를 통해 timestamp를 출력하는 방식으로도 확인하였다.

데이터 전송 테스트는 사용자 입력 대기로 인한 불필요한 시간지연을 없애기 위해 Linux Shell script를 이용한 batch 파일을 작성하여 Fig. 18처럼 실시하였다. 먼저 표준크기를 통해 지표 데이터 측정을 위한 테스트를 시행하였고 이를 통해 Table 8과 같은 기준 측정값 결과를 취득하였다.

지표 데이터 측정 시험에서 100GB 크기의 VM 전송에 사용된 실측 데이터를 기준으로 크기별로 소요

Fig. 17. Test to measure value for VM transfer

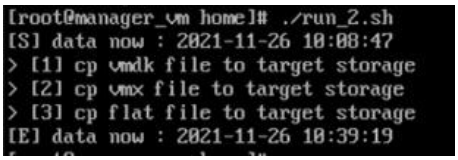

Fig. 18. Testing with shell script

Table 8. Measured value by transfer test

Std. size	Elapsed time	Byte per sec.
100GB	30 min.	55.5

시간을 예측 후 실측해본 결과와 비교하는 추가 시험을 실시하였다.

그 결과 범용적인 VM 크기인 100GB에서는 복구시간이 약 30분 내외로 H/W 최소 복구시간인 약 2시간에 비해 약 75% 신속함을 증명할 수 있었다. Table 9와 Fig. 19의 결과에서 보면 H/W 복구 최대 시간인 4시간과 비교하여 약 800GB 크기를 갖는 VM을 relocation 할 때 대체로 유사한 시간이 소요되며, 이보다 적은 크기의 VM에서는 보다 좋은 성능을 내는 것으로 측정되었다.

장애시험에서 취득한 결과를 바탕으로 VM Image 크기가 300GB 이하에서는 H/W 복구를 진행하는 중에도 서비스를 제공할 수 있어 복구 후 재기동보다 매우 효과적으로 서비스 연속성을 확보할 수 있는 것으로 확인되었다.

Table 9. Test result for prediction and measures

VM size (GB)	Prediction (Hour)	Measures (Hour)	Gap (Min.)
100	0.5	0.5	0
300	1.5	1.5	0
500	2.5	2.4	-6
1,000	5.0	5.4	+20

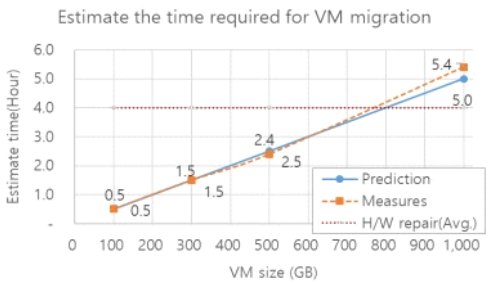

Fig. 19. Graph to prediction and measures

2.3.4 Simulation data generator

기업에서 운영 중인 서비스의 자원 사용 특성은 몇 가지 형태로 군집화할 수 있다. Table 10에서 보는 것처럼 3가지 workload 타입으로 분류할 수 있으며, 각

Table 10. Business service classification and workload

Type	Workload Type	Remarks
Workload-1	Batch processor job	Mail server, etc.
Workload-2	Used for special time	Groupware, etc.
Workload-3	Frequently used	Interfacing job, etc.

workload 형태에 맞는 자원 사용현황 데이터가 필요하다. 하지만, 실제 운영 서버에서 취득하는 것은 의도치 않은 시스템 장애를 유발할 수 있어 쉽게 취득하기 어려운 현실적인 제약이 있다.

본 연구에서는 이러한 문제의 우회적인 방법으로 simulation 방법을 적용하였다. 별도의 simulation 데이터 생성기를 이용하여 각 workload의 자원 사용 유형에 부합하는 데이터를 생성하는 방식으로의 접근하였다.

Fig. 20은 simulation 데이터 생성기로 생성된 workload 데이터를 그래픽으로 출력한 화면이다. 앞서 기술한 총 3가지 workload에 맞춘 시뮬레이션 데이터를 생성하였으며, 설정한 기간만큼의 데이터를 생성하여 Deep Learning에 적용할 수 있는 CSV 파일 포맷의 데이터를 생성토록 python 소스를 개발하였다.

Workload-1은 전자메일처럼 지속해서 사용하지만, CPU 사용 부하가 높지 않은 시스템의 특성을 반영하여 최대 20% 이내에서 부하를 발생시키고 있으며, Workload-2는 그룹웨어처럼 특정 시간대에 사용자 이용이 급증하여 CPU 부하가 몰리는 특성을 반영하고 있다. 일반적인 기업 내 그룹웨어처럼 출근 시간 30분 전부터 출근 후 1시간까지, 점심시간 전후와 퇴근 전후로 최대 부하량이 발생하도록 설계되었다. Workload-3는 연계시스템처럼 지속해서 중간규모의 CPU 부하가 발생하는 특징을 반영한 데이터를 설계하였다.

생성된 simulation 데이터 중에서 Workload-2 형태

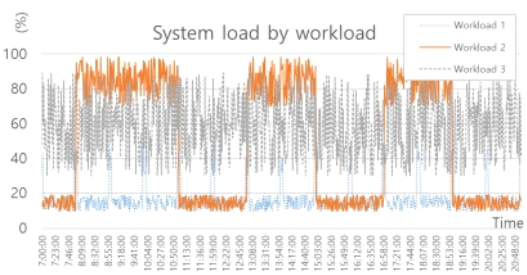

Fig. 20. CPU load graph for simulation using generation data

의 데이터를 사용해 장애가 발생하는 시간을 오전 11:00로 예상하고 향후 2시간 동안의 CPU 발생 추이를 예측하는 데 사용한다.

2.3.5 LSTM을 이용한 미래 CPU 자원 사용 예측

준비된 simulation 데이터와 Deep Learning 알고리즘을 적용하여 사용현황 미래 예측을 수행하였다. 테스트에 사용된 simulation 데이터는 Table 11처럼 분당 1개씩 생성하여 1일 1,440건의 데이터로 총 5일간의 데이터인 7,200건을 사전에 준비하였다.

전체 데이터 중에서 4일간의 생성 데이터를 추출하여 사전에 정의된 패턴대로 데이터가 생성되었는지를 확인하였고, Fig. 21에 표시되는 것처럼 모든 날짜의 데이터가 Workload-2 형태의 데이터 특성을 반영하여 생성된 것을 확인할 수 있다.

Simulation 전체 데이터를 각각 용도에 맞춘 데이터로 준비한다. Test-set은 최종 1일치의 오전 11시 이후의 데이터 770건으로 준비하였고, 나머지 6,430건의 데이터는 학습을 위한 Train-set으로 사용하였다. Train-set 중 약 25%의 데이터를 Validation-set으로 할당하여 학습모델을 검증하는 데 사용한다.

Fig. 22은 data set을 각각의 용도에 맞추어 색상으로 표시한 그래프이다. 각각의 용도에 맞추어 좌측부터 Train-set, Validation-set, Test-set으로 정확히 데이터가 나누어진 것을 확인할 수 있다.

Table 12는 최적의 모델을 탐색하기 위한 Hyper-parameter 설정 테스트 값이다. 이번 실험에서는 먼저 Window size, Optimizer 와 Loss-function을 고정한 상태에서 Epoch와 Batch-size를 조정해 가며

Table 11. Simulation data summery

Period	Data count per day	Total data count
1 min	1,440 (24 hours * 60 min)	7,200 (1 day * 5 days)

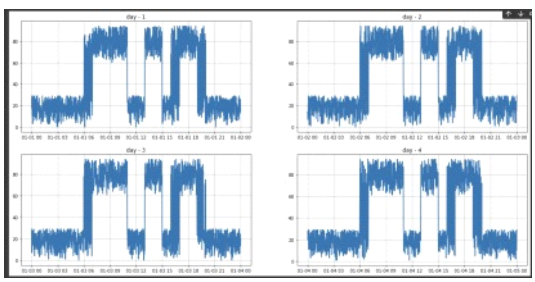

Fig. 21. Data sampling graph

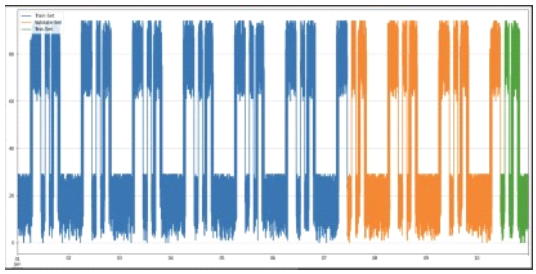

Fig. 22. Data graph divided for work

Table 12. Hyper-parameters for testing

Type	Test case 1	Test case 2	Test case 3
Epoch	1,000	300	100
Batch size	1	10	30
Window size	10		
Optimizer	MSE(Mean-squared-error)		
Loss func.	Adam		

over-fitting과 under-fitting 상태가 되지 않는 최적의 상태를 확인하였다.

Fig. 23은 LSTM 모델로 예측한 결과 그래프이다. Test-set 데이터를 모델에게 학습시킨 결과 A와 B 구간은 예상대로 사전 예측을 적합하게 수행하는 구간으로 나타났으며, C 구간에서는 예측이 실제값의 범위를 벗어나는 상태로 표시되지만 대체로 적합하게 CPU 자원의 미래 사용률을 사전에 예측하는 것으로 나타났다.

추가로 모델의 정확도를 향상시키기 위해 Fig. 24에서는 Hyper-parameter 중에서 데이터의 Scaler를 바꿔가며 모델에 미치는 영향을 확인해 보았다. 총 4가지의 Scale-model을 적용하여 비교해본 결과 Scaler를 적용하지 않은 상태와 근접하게 나타난 것은 MinMaxScaler와 MaxAbsScaler였다.

Fig. 25는 window size를 조정하며 확인한 시험 결과이다. window size는 설정하는 값에 따라 얼마나 사전에 이후 데이터를 예측하고자 하는지 선택할 수

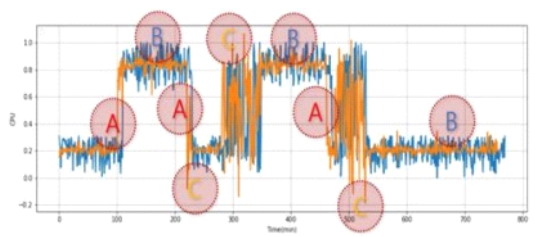

Fig. 23. Fitting result graph using LSTM model

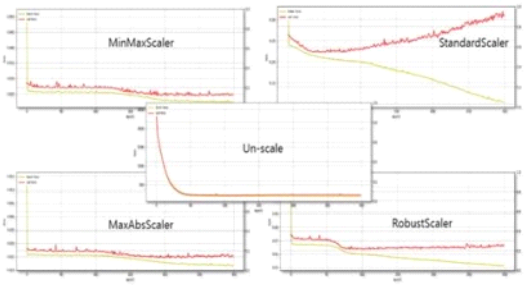

Fig. 24. Testing for Scaler parameter

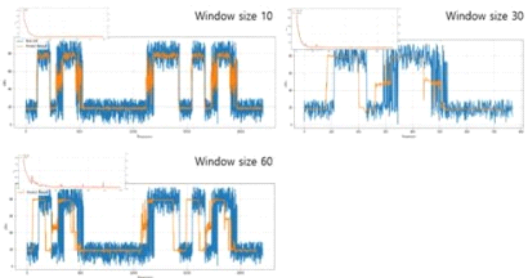

Fig. 25. Testing for window-size parameter

있다. 분당 1회의 데이터이므로 10을 선택하면 10분 전부터 예측할 수 있다. 테스트에서는 10분, 30분, 60분에 해당하는 windows size를 변형하여 각각 사전에 정확히 패턴이 예측되는지를 검증하였다.

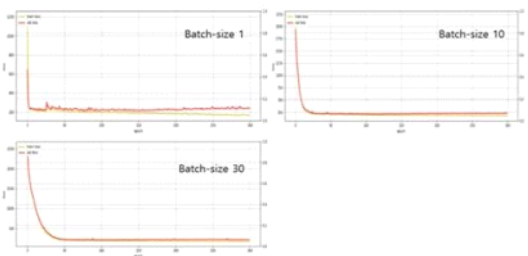

Fig. 26. Testing for Batch-size parameter

Table 13. Selected parameters

Type	Parameters
Model	LSTM
Scaler	MinMaxScaler
Epoch	100
Batch size	30
Window size	30
Optimizer	MSE
Loss func.	adam

Fig. 26은 Batch-size를 조절한 결과이다. Batch-size는 연산의 성능에 영향을 미치므로 30 정도에서 설정하는 것이 적합한 것으로 나타났다.

이러한 테스트를 통해 선택한 최적의 Hyper-Parameter는 Table 13과 같다.

Ⅲ. 결 론

본 연구에서는 기업에서 Private-Cloud를 구성할 수밖에 없는 상황을 분석하였고 다양한 이기종 Hypervisor를 이용한 다수의 cluster의 운영환경을 고찰하였다. 이런 다수의 VC 환경에서 반드시 필요하지만, 평시에는 활용하지 못하는 고가용성 자원을 전역 시스템으로 공동 활용할 수 있는 SHA 개념을 정의하고, 이를 통해 장애 시 PHA와 동일한 기능을 수행할 수 있는 VR-SAS 시스템 아키텍처 모델을 설계하였다.

또, 제안하는 VR-SAS 아키텍처 모델을 통해 장애 발생 시 VM relocation의 실현 가능성을 증명하여 H/W 장애 상황에서 파트 수급 및 교체 대비 최소 75% 신속함도 보장할 수 있는 성능도 검증하였다. 특히 Deep Learning 중 RNN 알고리즘을 적용한 예측 모델과 통합하여 CPU의 미래 사용 추이를 예측할 수 있는 최적의 Hyper-parameter를 검증하고, 이를 통해 relocation 된 VM의 생명주기까지 보장하되 다른 서비스에 미치는 영향을 최소화하는 방안도 제시하였다.

본 연구는 내결함성 측면의 단일 시스템 차원보다 Data Center 내의 전역 시스템 차원에서, 요소 기능보다 요소 기능을 아우르는 시스템 아키텍처 모델 차원에서 청사진을 제시하여 기업의 IT 전략의 저해 요인을 제거하는 것을 목표로 한 연구였다는 점에서 시사하는 바가 크다.

본 연구에서 사용된 데이터는 운영환경에서 실측된 데이터를 활용하지 못해 시뮬레이션 방식으로 접근하여 CPU 자원 사용률이 획일적인 측면과 Cloud 환경에서 확산하고 있는 PaaS(Platform as a Service) 등을 고려하지 못한 한계점이 있다.

따라서 다음에는 실측 데이터를 활용할 수 있는 방안과 PaaS 나아가 Serverless 컴퓨팅 환경도 지원할 수 있도록 그 대상과 범위를 확장하는 연구를 진행할 계획이다.

References

[1] M. C. Kim, *Korea Cloud IT Infrastructure Market Forecast, 2021-2025: Accelerating*

Enterprise Digitization (Aug. 12, 2021), Retrieved Dec. 10, 2021, https://www.idc.com/getdoc.jsp?containerId=prAP48159621.

[2] D. K. Yoon, *Cloud Issue Report 2019*, CEART, vol. 03, pp. 3-15, Mar. 2019.

[3] Z. Pantić and M. A. Babar, *Guidelines for Building a Private Cloud Infrastructure - Technical Report*, IT University of Copenhagen, ISBN 978-87-7949-254-7, 2012.

[4] K. H. Choi, *Weelky Techincal Report 2011*, NIPA, Jun. 10, 2011.

[5] F. David, *Private Cloud is more Cost Effective than Public Cloud for Organizations over $1B* (2010), Retrieved Dec. 10, 2021, http://wikibon.org/wiki/v/Private_Cloud_is_more_Cost_Effective_than_Public_Cloud_for_Organizations_over_$1B.

[6] Telecommunications Technology Association (TTA), *A Guideline for Hardware Sizing of Information Systems*, TTAK.KO-10.0292/R2, TTA Standard, 2018.

[7] O. S. Holovnia1 and V. P. Oleksiuk, "Selecting cloud computing software for a virtual online laboratory supporting the Operating Systems course," *9th Wkshp. CTE 2021*, pp. 216-227, Kryvyi Rih, Ukraine, Dec. 2021.

[8] P. D. Patel, M. Karamta, M. D. Bhavsar, and M. B. Potdar, "Live virtual machine migration techniques in cloud computing: A survey," *Int. J. Comput. Appl.*, vol. 86, no. 16, 2014.

[9] F. Xu, F. Liu, L. Liu, H. Jin, B. Li, and B. Li, "iAware: Making live migration of virtual machines interference-aware in the cloud," *IEEE Trans. Comput.*, vol. 63, no. 12, pp. 3012-3025, Dec. 2014. (https://doi.org/10.1109/TC.2013.185)

[10] H. M. Lee, "A development of adaptive VM migration techniques in cloud computing," *J. KIPS Trans. Comp. and Commun. Syst. 2015*, vol. 4, no. 9, pp. 315-320, Sep. 2015. (https://doi.org/10.3745/KTCCS.2015.4.9.315)'

[11] AWS Certification Study, *Fault tolerance and High Availability* (2022), Retrieved Mar. 18, 2022, https://ausg.github.io/aws-certification-study/docs/module4/script3.

[12] K. J. Naik, "An alternate switch selection for fault tolerant load administration and VM migration in fog enabled cloud datacenter," *Innovation and Intell. for Informatics, Comput., and Technol. (3ICT)*, 2021. (https://doi.org/10.1109/3ICT53449.2021.9581964)

[13] NEC Co. LTD., *Server Redundancy Solution (HA Cluster) Basic Concepts* (2021), Retrieved Dec. 12, 2021. (https://kr.nec.com/ko_KR/file/NEC_HA_Cluster_Study.pdf)

[14] WikiDocs, *Recurrent Neural Network* (2022), Retrieved Mar. 18, 2022, https://wikidocs.net/22886.

[15] H. J. Seo, J. C. No, and S. S. Park, "Machine learning based failure prediction method for supporting optimal system reliability," *J. IEIE*, vol. 58, pp. 41-56, Jan. 2021. (https://doi.org/10.5573/ieie.2021.58.1.41)

[16] J. J. Jong and J. Y. KIM, "A performance analysis by adjusting learning methods in stock price prediction model using LSTM," *J. Digital Convergence 2020*, vol. 18, no. 11, pp. 259-266, Nov. 2020. (https://doi.org/10.14400/JDC.2020.18.11.259)

[17] M. H. Lee, Y. R. Yoon, and H. J. Moon, "Performance evaluation of an indoor temperature forecasting model based on GRU for floor heating system operation," *J. Korean Soc. Living Environ. Syst.*, vol. 27, no. 3, pp. 272-282, Jun. 2020. (https://doi.org/10.21086/ksles.2020.06.27.3.272)

박 선 철 (Seon-cheol Park)

2007년 2월 : 방송대 컴퓨터과학과 졸업

2020년 9월~현재 : 숭실대학교 일반대학원 IT융합학과 석사과정

<관심분야> 인공지능, 클라우드, 데이터센터, 통신/네트워크

[ORCID:0000-0003-1838-974X]

김 영 한 (Young-han Kim)

1984년 : 서울대학교 졸업

1986년 : 한국과학기술원 공학석사

1990년 : 한국과학기술원 공학박사

1994년~현재 : 숭실대학교 전자정보공학부 교수

<관심분야> 차세대 인터넷 프로토콜, 이동/무선 네트워크, 인터넷 텔레포니, 센서/모바일 애드혹 네트워크