JDCS
디지털콘텐츠학회논문지
Journal of Digital Contents Society
Vol. 18, No. 1, pp. 167-173, Feb. 2017

클라우드 교육 시스템의 SLA 보장을 위한 오픈소스기반 요소 성능 분석

윤 준 원[1] · 송 의 성[2*]
[1]한국과학기술정보연구원 슈퍼컴퓨팅본부
[2]부산교육대학교 컴퓨터교육과

Analysis of Component Performance using Open Source for Guarantee SLA of Cloud Education System

JunWeon Yoon[1] · Ui-Sung Song[2*]
[1]Department of Supercomputing Center, KISTI, Daejeon 34141, Korea
[2*]Department of Computer Education, Busan National University of Education, Busan 47503, Korea

[요 약]

클라우드의 사용이 보급화 됨에 따라 가상화 기술에 다양한 요구사항이 접목, 적용되고 있다. 클라우드 컴퓨팅의 대표적인 특징은 사용자가 원하는 자원 요구사항에 따라 최적화 된 환경을 구축할 수 있으며, 나아가 확장성에도 유연하게 대처할 수 있다. 이런 장점으로 인해 다양한 분산컴퓨팅 분야에 클라우드 컴퓨팅이 적용, 활용되고 있는 실정이다. 이를 위해 클라우드 환경의 성능 안정성을 보장하는 것이 무엇보다 중요하다. 본 연구에서는 구축된 클라우드 교육 시스템 테스트베드 환경에서 시스템의 성능을 보장하기 위한 다양한 요소성능(metric) 측정을 오픈소스 기반의 툴들을 이용하여 분석하였다. 이를 위해 프로세서, 메모리, 캐시, 네트워크 등 가상화 환경에 영향을 주는 요소 성능을 구분하고, 그 성능을 호스트머신(Host Machine) 및 가상머신(Virtual Machine)에서 각각 측정하였다. 이로서 시스템의 상태를 명확하게 파악할 수 있으며, 문제점을 빠르게 진단하여 가용성을 증대시키고 나아가 클라우드 컴퓨팅의 SLA(Service Level Agreement) 수준을 보장할 수 있다.

[Abstract]

As the increasing use of the cloud computing, virtualization technology have been combined and applied a variety of requirements. Cloud computing has the advantage that the support computing resource by a flexible and scalable to users as they want and it utilized in a variety of distributed computing. To do this, it is especially important to ensure the stability of the cloud computing. In this paper, we analyzed a variety of component measurement using open-source tools for ensuring the performance of the system on the education system to build cloud testbed environment. And we extract the performance that may affect the virtualization environment from processor, memory, cache, network, etc on each of the host machine(Host Machine) and a virtual machine (Virtual Machine). Using this result, we can clearly grasp the state of the system and also it is possible to quickly diagnose the problem. Furthermore, the cloud computing can be guaranteed the SLA(Service Level Agreement).

Key word : Cloud Computing, Virtualization, Component Performance, Benchmark, Practical Education
색인어 : 클라우드컴퓨팅, 가상화, 요소성능, 벤치마크, 교육실습

http://dx.doi.org/10.9728/dcs.2017.18.1.167

Received 27 December 2016; **Revised** 28 January 2017
Accepted 25 February 2017

***Corresponding Author; Ui-Sung Song**
Tel: +82-051-500-7326
E-mail: ussong@bnue.ac.kr

Ⅰ. 서 론

클라우드는 컴퓨팅 자원뿐만 아니라 어플리케이션 서비스까지도 사용자의 요구사항에 맞게 구성하여 제공할 수 있는 특징을 가지고 있다. 클라우드 컴퓨팅은 가상화 기술을 기반으로 실제 구현 및 서비스가 가능하다[1]. 최근 x86 기반의 멀티코어가 일반화 되고, 나아가 가속 프로세서(NVIDIA GPU, INTEL PHI 등)의 장착으로 단일 노드의 성능이 급격히 증가하고 있다. 이런 고성능의 자원을 효율적으로 분배, 캡슐화하기 위한 방법으로도 가상화 기술이 사용되고 있다[2]. 이렇게 클라우드 환경은 다양한 하드웨어 구성, 가상화(오픈소스 또는 상용 프로그램) 솔루션, 사용자 응용 소프트웨어 등이 서비스에 맞게 스택(Stack)화 되어 제공된다.

가상화를 통한 클라우드 서비스가 보급화, 일반화 되고 다양한 용도로 정착되면서 성능과 서비스 수준을 안정적으로 제공해야 하는 요구사항에 맞닥뜨리게 되었다[3]. 국내 방송통신위원회에서는 사용자가 사용하는 자원의 요구사항과 성능을 보장하기 위한 사용자 계약(SLA: Service Level Agreement)과 관련 “클라우드 SLA 가이드 도입방안(2011.10)”을 통해 서비스 제공자와 이용자가 고려해야 할 서비스 수준을 정량화하여 제시하고 있다. 여기에 클라우드 서비스 가용성은 아래와 같이 나타내고 기준을 99.5% 이상 되도록 권고하고 있다[4].

$$\text{가용성}(\%) = \frac{\text{클라우드서비스접속가능시간(실제가동시간)}}{\text{정해진서비스운영시간(예정가동시간)}} \quad (1)$$

본 논문에서는 사용자의 요구사항에 따른 다양한 클라우드 환경에서 서비스를 안정성 있게 제공할 수 있도록 SLA 보장을 위한 시스템 성능을 요소별로 측정하고자 한다. 이를 위해 기존에 구축된 가상화 기반 실습환경 테스트베드를 활용하여 시스템 안정성 측정에 필요한 요소들을 구분하였다[5][6]. 이후 선택된 요소별 성능을 측정할 수 있는 오픈소스 기반의 벤치마크 도구들을 선정하였다. 이 도구들을 통해 얻은 성능 결과를 이용하여 안정화된 가상화 실습환경을 보장할 수 있으며, 장애 발생 시 이슈를 보다 빠르게 파악할 수 있어 서비스 전체의 가용성을 향상시키게 된다.

Ⅱ. 관련 연구

클라우드 서비스의 핵심인 가상자원에 대한 성능 측정 항목을 도출하고 지표를 선정하는 다양한 방법들이 연구되고 있다. 국내에서 또한 협회나 컨소시엄을 통해 측정 항목에 대한 지침과 도구를 가지고 프레임워크를 세우는 노력이 진행 중이다.[7] 그러나 가상화 환경에 대한 시스템 요소 성능을 측정하고 해당 시스템에 적합한 구체적인 성능치를 제시하고 있지 않다.

본 논문에서는 가상화를 통해 구축된 클라우드 컴퓨팅 실습환경 테스트베드를 활용하여 실험을 수행하였다. 시스템 성능을 측정하기 위한 다양한 벤치마크 도구들이 존재하며 CPU, 메모리, 네트워크 등의 요소성능부터 시스템의 연산(부동소수점), 파일시스템의 I/O 성능까지 다양하게 시스템의 성능을 추출할 수 있다. 본 논문에서는 가상화 환경에서 성능 분석시 필요한 요소들을 파악한 후 <표 1>과 같은 오픈소스기반의 벤치마크 도구를 이용하여 성능 분석을 수행하였다.

표 1. 요소 성능 벤치마크 리스트
Table. 1. Component Performance Benchmark List

이름	평가기준	특징
HPL	FLOPS	분산 메모리 환경에서 시스템의 성능 측정
STREAM	GB/s	프로세서와 메모리간의 데이터 전송 속도 측정
STRIDE	GF/s	메모리/캐시 성능 및 스트레스 테스트
MDTEST	IOPS	파일시스템의 메타데이터 처리 성능 측정
IOR	GB/s	파일시스템의 POSIX-IO 및 MPI-IO 성능
b_eff	MB/s	인터커넥션의 커뮤니케이션 대역폭 측정

- HPL

분산-메모리 컴퓨터 환경 즉, 독립적인 클러스터 시스템에서의 성능을 측정하기 위해 사용된다. HPL은 MPI 라이브러를 이용하여 Linpack(여러 개의 프로세서가 하나의 커다란 메모리 공간에 접근하여 계산을 수행)을 병렬로 처리하게 된다[8]. Linpack은 선형시스템의 해를 구하는데 일반적 사용하는 패키지로 BLAS 같은 라이브러리를 이용하여 Ax=B 형태의 연립 방정식의 해를 구하게 된다. 연산의 대부분이 부동 소수점(Floating-point)로 구성되어 있어 이 계산을 통해 컴퓨터의 성능을 측정할 수 있다. HPL은 슈퍼컴퓨터 Top500 선정에 사용되는 벤치마크 프로그램으로 병렬처리 라이브러리인 MPI와 수학 라이브러리인 BLAS, CBLAS, ATLAS, 그리고 C, Fortran 컴파일러 등의 설치가 우선되어야 한다.

- STREAM

STREAM은 단일노드에서 프로세서와 메모리 간의 대역폭을 측정하는 도구로 네가지 타입의 벡터 커널 연산(Copy, Scale, Add, Triad)을 OpenMP 프로그래밍 모델을 사용하여 수행하게 된다[9]. 최초 STREAM 소스 코드를 다운로드 받은 후(V.5.10) C 컴파일러 빌드, 설치 후 OMP_NUM_THREADS 값을 설정하여 성능 결과를 얻을 수 있게 된다.

- STRIDE

STRIDE는 단일노드에서 메모리와 캐시의 성능을 측정하는 도구이다. 이 벤치마크는 스칼라 루프와 벡터 연산을 수행하면서 메모리에 과도한 부하를 주어 성능을 얻게 된다[10]. STRID3, VECOP, CACHE, STRIDOT, CACHEDOTS의 테스트를 수행하며 MFLOPS로 측정된다.

▪ MDTEST

MDTEST는 다양한 메타데이터 동작에 대한 성능을 측정하는 도구로, MPI를 활용하여 파일시스템의 open, stat, close 동작을 측정하게 된다[11]. 본 연구의 클라우드 실습환경은 공용의 파일시스템을 마운트 하여 사용하게 된다. 여러 사용자들 실습환경에 접속해서 공용의 파일시스템에 파일들을 확인하거나 생성, 삭제하는 동작을 동시에 수행하게 된다. 이 벤치마크 도구를 이용하여 관련된 성능을 사전에 측정해 볼 수 있다.

▪ IOR

IOR은 공용 병렬파일시스템의 I/O 성능을 측정하는 도구이다. POSIX, MPIIO와 같은 인터페이스와 액세스 패턴을 사용하여 병렬파일시스템의 성능을 측정한다[12]. MPI를 통해 파일시스템이 마운트 된 클러스터 환경에서 읽기(read), 쓰기(write) 성능을 측정한다. 이 테스트에서는 수행되는 프로세스가 하나의 파일을 생성(Single Shared File)하는 방식과 각 프로세스마다 각자 파일을 생성(File Per Process)방식의 두 가지 형태로 수행하게 된다.

▪ b_eff

Effective Bandwidth(b_eff)는 분산 컴퓨팅 환경에서 인터커넥션을 통한 커뮤니케이션 성능을 측정하는 도구로 모든 프로세스가 동시에 병렬로 이웃 노드에게 메시지를 전송한다[13]. 실제 현업에 어플리케이션들이 MPI통신을 통해 작업을 수행할 때 크고 작은 메시지들이 서로 다른 환경에서 다양한 대역폭으로 전송하게 되는데 이를 고려한 평균값을 측정하게 된다.

Ⅲ. 오픈소스기반 성능 분석

본 논문에서는 가상화를 통해 구축된 클라우드 컴퓨팅 교육용 실습 환경의 시스템 성능을 보장하고 가용성을 확보하기 위해 오픈 소스 기반의 벤치마크 도구를 이용하여 요소성능을 분석하였다. 여기에는 분산-메모리 환경에서의 시스템 성능, 네트워크 응답/지연 시간, 공용의 파일시스템을 사용하는 경우에는 I/O 대역폭(Bandwidth) 등에 대한 요소 성능을 측정하게 된다. 성능 실험은 크게 클라우드가 구성된 호스트 머신(Host Machine)과 그 위에서 동작하는 가상화 환경(Virtual Machine)의 두 형태로 구분하여 실험하게 된다. 두 환경에서 각각 시스템을 구성하고 있는 개별 구성요소(메모리, 스토리지, 네트워크 등)와 전체 실측 성능을 테스트하였다. 측정된 실측 성능은 클라우드 실습 환경의 성능을 측정하는 기준이 된다. 또한 개별 요소들의 성능을 부분별로 신속하게 파악하고 대처할 수 있어 전체 시스템의 가용성을 증가시켜 클라우드 서비스의 SLA 수준을 높일 수 있다.

▪ HPL(High Performance LINPACK)

사용자의 실습 환경 성능 측정을 위해 호스트머신(Host Machine)과 가상머신(Virtual Machine)에 <표 2>과 같이 소프트웨어를 설치하였다.

표 2. 소프트웨어 스택
Table. 2. Software Stack

운영체제(OS)	CentOS 7.2(Kernel 3.10.0-327)
컴파일러	gcc(v.4.8.5), gfortran(v.4.8.5)
MPI라이브러리	mpich3.2
수학라이브러리	ATLAS
벤치마크 툴	HPL(v.2.2)

HPL 벤치마크는 MPI를 기반으로 동작하며 정밀한 수학 계산을 위한 라이브러리 패키지를 필요로 한다. 수학 라이브러리 패키지로는 위의 표와 같이 BLAS library를 생성하는 프로그램인 ATLAS(Automatically Tuned Linear Algebra Software)를 사용하였다[14][15]. 실험을 위해 설치된 소프트웨어 패키지는 n차 선형방정식 Ax=B의 n by n+1 계수행렬(Coefficient Matrix)을 LU 인수분해(LU factorization)를 통해 계산하게 된다.

호스트머신과 가상머신의 각 이론 성능은 아래의 방식으로 구할 수 있다.

- Theoretical Performance(Rpeak) = Number of cores x Floating point instructions per clock cycle x Clock frequency
- Quad-Core AMD Opteron Processor 8356 @ 2.3GHz : 16 x 4 x 2.3 GHz = 147.2GFlops (1 core=9.2GFlops)

표 3. HPL 성능결과
Table. 3. HPL Performance result

구 분	호스트머신			가상머신	
노드수(core)	1(16)	4(64)	16(256)	1(1)	2(2)
이론성능 (GFlops)	147.2	588.8	2,355.2	9.6	19.0
실측성능 (GFlops)	85.4	329.8	1,286.0	5.4	10.13
이론대비 실측성능(%)	58.0	56.0	54.6	56.8	52.8

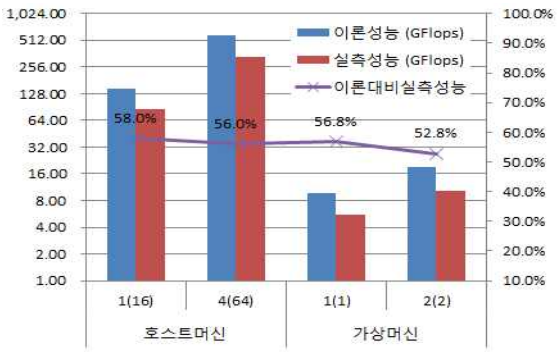

그림 1. HPL 성능 비율
Fig. 1. HPL Performance Ratio

HPL 벤치마크에서는 다양한 옵션을 설정할 수 있는데 성능 측정 결과에 큰 영향을 주는 것은 문제 크기 Ns와 프로세스 격자 Ps, Qs 그리고 블록사이즈 NBs 값 등이 있다. 실험의 성능은 시스템의 메모리 용량에 따른 문제 크기 Ns의 설정에 따라 성능치가 결정되며 일반적으로 커널에서 사용하는 기본 메모리 사용량이 있기 때문에 전체 메모리 중 대략 80%정도로 설정해서 테스트하게 된다. 메모리 사이즈는 아래와 같이 구할 수 있다.

$$Ns \simeq 0.8 \sqrt{\frac{Total\ Memory\ Size(bytes)}{8\,(double\ precision)}} \quad (2)$$

본 실험에서는 성능테스트에 소요되는 시간을 줄이기 위해 문제의 사이즈를 줄여서 수행하였고, 이때 호스트머신과 가상머신에서 이론대비 실측성능이 50% 이상인 경우 시스템의 성능이 정상범위라 볼 수 있다.

▪ STREAM

STREAM은 단일노드에서 프로세서와 메모리 간의 대역폭을 측정하는 도구로 STREAM 소스 코드를 다운로드 받아 아래와 같이 OpenMP 옵션을 사용하여 컴파일 하면 된다..

```
# gcc -o stream_omp –fopenmp stream.c
```

성능 테스트는 Quad-Core CPU를 장착한 노드에서 OMP_NUM_THREADS 환경변수를 1부터 4까지 변경하면서 테스트 하였다.

표 4. STREAM 성능결과
Table. 4. STREAM Performance result

연산(MB/s)	Thread 개수			
	1	2	3	4
Copy	2898.8	3700.2	3595.3	3652.8
Scale	2972.4	3703.7	3580.8	3624.9
Add	3999.2	4310.3	4225.8	4182.6
Triad	3625.8	4334.6	4237.2	4206.8

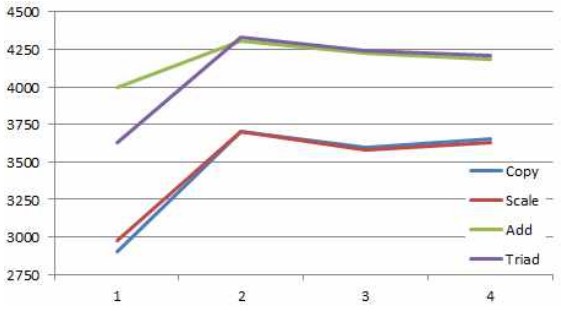

그림 2. STREAM 성능 테스트
Fig. 2. STREAM Performance Test

▪ STRIDE

STRIDE는 단일노드에서 메모리와 캐시의 성능을 측정하는 도구이다. 본 실험에선 STRID3 테스트를 수행하였는데 이 테스트에서는 일정한 크기의 배열에 접근 길이(stride)를 최대 1,024까지 늘려가면서 테스트를 수행하였고 배열의 길이는 256×stride로 설정하였다. 이 코드는 C와 Fortran 버전으로 되어 있다.

표 5. STRIDE 성능결과
Table. 5. STRIDE Performance result

```
Running benchmarks on machine: Linux 3.10.0-327.22.2.el7.x86_64 GNU/Linux
Starting STRIDE benchmark
0m0.003s 0m0.003s
0m0.000s 0m0.002s
STRID3 Optimized FORTRAN MEMORY Benchmark
*********************************
/usr/bin/time -p ./strid3.Opt
 /usr/bin/time -p ./strid3.Opt
STRID  TIME    MFLOPS    RATIO
 1 2.1038998E-02 2.4919817E+03 1.0000000E+00
 1 3.9804004E-02  1.3171740E+03 1.8919154E+00
 2 4.0238999E-02  1.3029349E+03 1.9125910E+00
 3 4.0233999E-02  1.3030969E+03 1.9123534E+00
 4 4.0232986E-02  1.3031296E+03 1.9123052E+00
 5 4.0231004E-02  1.3031938E+03 1.9122111E+00
 ..............
```

▪ MDTEST

MDTEST는 MPI를 이용하여 메타데이터 동작과 관련된 시스템 성능을 측정하는 도구로 클라우드 실습환경의 공용 파일 시스템 성능을 측정하게 된다.

이 벤치마크 도구를 수행하기 위해서는 C 컴파일러와 MPI 라이브러리가 필요하다. 성능 측정을 위해서 공용 또는 개별 디렉터리 선택, 프로세스에서 실행될 파일의 개수, 테스트 수행 옵션(creat, stat, remove) 등이 설정되어야 한다.

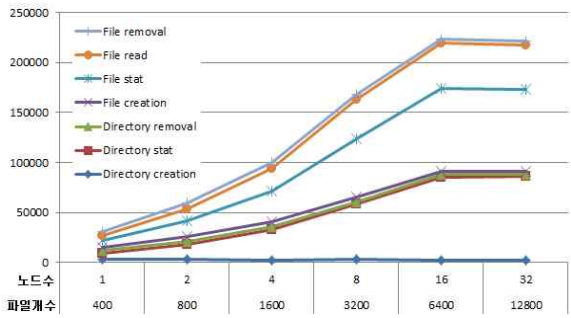

그림 3. MDTEST 수행 결과
Fig. 3. MDTEST Test Result

이 실험은 공용 파일시스템의 단일 디렉터리에서 수행하였고, 실험에서 보는 것과 같이 creation, removal 동작의 경우 파일에 대한 락킹(locking) 메커니즘의 영향으로 인해 성능이 증가하지 않고 일정함을 볼 수 있다[16]. 디렉터리 파일의 초당 I/O오퍼레이션이 3000~4000인 반면에 stat의 초당 I/O오퍼레이션 16노드까지는 비례적으로 증가하였다.

▪ IOR

IOR 테스트는 가상화 실습환경이 구성되는 호스트머신에서 수행되었으며, 클라이언트는 블록사이즈는 2GB, 4GB, 6GB. 8GB로 변경하면서 수행하였다[17].

- Command: #./IOR.mpiio –a MPIIO -b 2g –o /gfs/ior/iorData- -t 2m -v -w –r –k

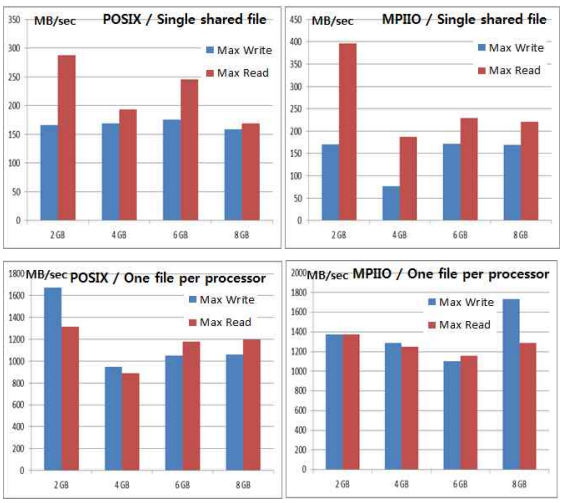

그림 4. 블록 사이즈 차이에 따른 I/O 성능(POSIX/MPIIO와 single-shared-file /file-per-process)

Fig. 4. I/O Performance under different block size (POSIX/MPIIO and single-shared- file/ file-per-process)

▪ b_eff

Effective Bandwidth(b_eff)는 분산 컴퓨팅 환경에서 인터커넥션을 통한 커뮤니케이션 성능을 측정하는 도구로 간단하게 얼마만큼 크기에 얼마만큼 메시지를 보냈을 때 그 평균값이 얼마인지를 체크하게 된다. 여기서는 두가지 패턴인 ring patterns, random patterns 형태로 MPI collective communication을 수행하게 된다.

아래와 같이 MPI를 이용해 수행되며 노드의 개수와 프로세스의 개수를 설정하여 수행하게 된다.

- Command: # mpirun -p multi -N #nodes -n #processes ./b_eff rsult_file

수행결과는 <표 6>과 같이 ring patterns, random patterns의 대역폭, 지연시간, ping-pong 대역폭 등의 결과를 얻게 된다. 본 실험에서는 호스트 머신의 프로세서 수를 4개에서 16개까지 늘려가면서 테스트를 수행하였다.

표 6. b_eff 수행 결과

table. 6. b_eff test result

Processors 수	b_eff	Lmax(MB)
4	2241	80
8	4452	80
16	8608	80
b_eff at Lmax rings & random (MByte/s)	b_eff at Lmax rings only (MByte/s)	Latency rings & random micro-sec
5576	5504	1.836
10935	10926	2.805
22293	22341	3.648
Latency rings only micro-sec	Latency ping-pong micro-sec	ping-pong bandwith (MByte/s)
1.892	2.153	2771
2.701	2.209	2769
3.229	2.244	2766

b_eff는 아래의 식과 같이 ring patterns, random patterns 평균값을 나타내며, (그림 5)와 같이 프로세서의 개수가 늘어남에 따라 일정한 수준으로 성능이 증가하게 된다.

$$-b_{eff} = (\log avg(\log avg_{ringpatterns}, \log avgr_{randompatterns}) \quad (3)$$

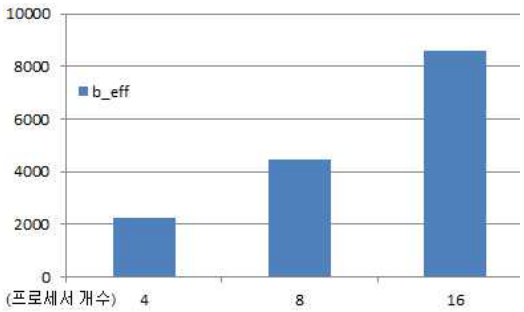

그림 5. b_eff 성능 결과

Fig. 5. b_eff performance result

Ⅴ. 결 론

클라우드 컴퓨팅의 장점은 사용자가 수행하고자 하는 환경에 따라 자원 요구사항을 최적화하여 지원할 수 있을 뿐만 아니라 확장성에도 유연하게 대처할 수 있다.

본 연구에서는 구축된 클라우드 환경의 실습 시스템에서 호스트 머신과 가상 머신으로 구분하여 다각도로 성능분석을 수행하였다. 이를 위해 오픈소스 기반의 벤치마크 도구들을 사용

하였고 시스템 사양과 환경 등을 반영하여 성능측정을 수행하였다. 최대성능 보다는 시스템의 안정적인 환경을 제공할 수 있는지를 판단하기 위해 각 성능측정마다 적정한 설정 값 등을 지정하여 수행하였다. 수행측정은 예방정비기간 또는 시스템의 주기적인 수행도구를 사용해서 사전에 클라우드 환경의 안정적인 성능을 보장할 수 있다. 또한 시스템의 성능 이슈가 발생할 경우 어느 요소에서 문제가 있는지를 수월하고 빠르게 파악할 수 있다.

향후, 다양한 시스템 성능요소에 따라 필요한 벤치마크 도구를 추가적으로 적용 및 사용자의 응용코드를 사전에 테스트 해봄으로써 클라우드 컴퓨팅 성능의 SLA를 좀 더 견고하게 보장할 수 있다. 나아가 crontab(cron table)과 같은 시스템의 스케줄러 도구를 통해 주기적으로 성능 점검하고 로그들을 분석함으로써 클라우드 서비스 가용성을 더더욱 높일 수 있다.

감사의 글

이 논문은 2016년도 부산교육대학교 교내 연구과제로 지원을 받아 수행된 연구임

참고문헌

[1] J. Diaz, G. v. Laszewski, F. Wang, and G. Fox, " Abstract Image Management and Universal Image Registration for Cloud and HPC Infrastructures", IEEE Cloud. 2012.

[2] Rad, P., et al. "Benchmarking Bare Metal Cloud Servers for HPC Applications." 2015 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM). IEEE, pp. 153-159, 2015.

[3] HWANG, Kai, et al., “Cloud performance modeling with benchmark evaluation of elastic scaling strategies”, IEEE Transactions on Parallel and Distributed Systems,27.1, pp. 130-143, 2016.

[4] “Introduction of cloud SLA guide”, Korea Communications Commission, Oct, 2011.

[5] Y JunWeon, S Ui-Sung, "Build the Teaching Practice System based on Cloud Computing for Stabilization through Performance Evaluation”, Journal of Digital Contents Society, 15.5, pp. 595-602, 2014.

[6] Y JunWeon, P ChanYeol, S Ui-Sung, "Building the Educational Practice System based on Open Source Cloud Computing." Journal of Digital Contents Society 14.4, pp. 505-511, 2013.

[7] J. HanGu, "Standard technology trend of cloud service performance measurement", TTA Journal,v.164,pp.45-49, Mar, 2016.

[8] PETITET, Antoine., “HPL-a portable implementation of the high-performance Linpack benchmark for distributed-memory computers”, http://www.netlib.org/benchmark/hpl/, 2004.

[9] J. D. McCalpin, “STREAM: Sustainable memory bandwidth in high performance computers”, University of Virginia, Charlottesville, Virginia, Tech. Rep., 1991-2007, a continually updated technical report. http://www.cs.virginia.edu/stream/.

[10] STRIDE benchmark. “STRIDE_summary”, https://asc.llnl.gov/sequoia/benchmarks/STRIDE_summary_v1.0.pdf, 1994.

[11] Mdtest benchmark. sourceforge.net/projects/mdtest.

[12] L, William, Mclarty, T. Morrone, C. “IOR benchmark”, 2012, https://sourceforge.net/projects/ior-sio.

[13] Effective Bandwidth (beff) Benchmark, https://fs.hlrs.de/projects/par/mpi/b_eff/b_eff_3.1/

[14] ATLAS, http://math-atlas.sourceforge.net/

[15] Whaley, R. C., & Dongarra, J. J., (1998, November). “Automatically tuned linear algebra software”, In Proceedings of the 1998 ACM/IEEE conference on Supercomputing, pp. 1-27, 1998.

[16] Alexander Oltu UniBCCS, “Performance Analysis-Synthetic benchmarks: IOR, bonnie++, mdtest”, 2010.

[17] S. Hongzhang, S. John, “Using IOR to Analyze the I/O performance for HPC Platforms”, Lawrence Berkeley National Laboratory, 2007.

윤 준 원 (JunWeon Yoon)

2002년~2004년 : 고려대학교 대학원 컴퓨터학과(이학석사)
2010년~2011년 : 고려대학교 대학원 컴퓨터학과 박사 수료
2005년~현 재 : KISTI 국가슈퍼컴퓨팅연구소 선임연구원

관심분야: 분산 컴퓨팅, 결함포용시스템, 슈퍼컴퓨팅, 병렬파일시스템, 배치스케줄링

송 의 성 (Ui-Sung Song)

1991년~1997년 : 고려대학교 컴퓨터학과(학사)
1998년~1999년: 고려대학교 대학원 컴퓨터학과(이학석사)
2000년~2005년: 고려대학교 대학원 컴퓨터학과(이학박사)
2006년~현 재: 부산교육대학교 컴퓨터교육과 교수

관심분야: 컴퓨터교육, 교육용로봇교육, 컴퓨터네트워크, 스마트러닝