DCU
DAEGU CATHOLIC UNIVERSITY

SW캡스톤디자인 과제 수행 결과보고서

SW
대구가톨릭대학교 소프트웨어중심대학사업단

1. 과제 수행 결과보고

과 제 명	컨테이너 환경에서 CPU 가상자원 성능 측정 방법에 관한 연구

□ 과제 개요 및 필요성

▶ 과제 개요

컨테이너 환경에서 가상자원의 성능의 차이가 있는지를 미리 확인해보고 예상치 못한 오류를 방지하고자 한다. 그리고 AWS의 EC2 가상 환경과 Docker를 활용한 컨테이너 환경의 성능의 차이를 비교해보고자 한다.

▶ 필요성

- 사전 성능측정이 이루어지지 않을 시 예상치 못한 성능 저하 문제나 심각한 시스템 오류 현상에 직면할 수 있다.
- 비용 절감과 신뢰도 향상, 불필요한 자원 사용 방지를 위해 서비스 개시 전 성능측정이 필요하다.
- 특히, 컨테이너 환경에서 가상자원의 성능의 차이가 있는지를 미리 확인해보고 예상치 못한 오류를 방지한다.
- 문제점을 사전에 해결하고자 컨테이너 환경에서의 성능이 가상환경에서의 성능과 차이가 있는지 확인하고자 한다.
- 실제 프로그램을 실행하였을 상황을 가정하여 성능을 측정했을 때 컨테이너 환경과 가상환경의 차이를 비교한다.

□ 과제의 개발 방법 및 과제 수행 과정

<실험1>

AWS EC2를 활용한 가상환경과 Docker를 활용한 컨테이너 환경을 이용하여 각 환경에서의 CPU 개수에 따른 성능 차이를 분석.

<실험2>

실제 환경에서 프로그램을 돌렸을 때의 성능을 확인하기 위해 stress 부하프로그램을 활용해 각 환경에 1코어 수만큼 스트레스를 부하 후 각 환경의 성능 평균을 비교 분석.

1. Amazon Web Service의 EC2를 CPU 개수를 다르게 설정하여 생성함.
  - Instance 유형은 가장 많이 사용하는 싱글 코어, 멀티코어, 쿼드코어 선택
  - 각각의 CPU 코어 수 (싱글 코어 : 1개 / 멀티코어 : 2개 / 쿼드코어 : 4개)

  < 실험에 필요한 환경 설정 >

  1. 벤치마킹 프로그램 sysbench 설치

  [컨테이너 환경을 위한 도커 설치 및 이미지 빌드]

  2 Docker 설치

  2-1 Docker 실행

ubuntu@ip-172-31-19-232: ~

```
ubuntu@ip-172-31-19-232:~$ sudo apt-get docker-io
ubuntu@ip-172-31-19-232:~$ sudo systemctl start docker
```

2-2 Docker 이미지 생성 - vim Dockerfile

* 인스턴스별로 스레드 수 변경 필요함 *

```
FROM ubuntu:20.04
RUN apt update
RUN apt-get install -y sysbench
CMD sysbench cpu --events=10000 --cpu-max-prime=20000 --time=0 --threads=2 run
```

2-3 Docker 이미지 빌드

```
ubuntu@ip-172-31-16-184:~$ sudo docker build -t ubuntu/test:1.0 .
```

3. STRESS 부하프로그램 Stress 설치

▶ CPU 성능측정 데이터

* CPU Speed : CPU의 연산 처리 속도
* Total Time : 연산 소요 시간
* Latency : 지연시간

= min(최솟값), **avg(평균)**, max(최댓값)

<1차 실험>

: 가상환경과 컨테이너 환경의 인스턴스별 CPU 성능측정

측정하기에 앞서 본 인스턴스의 CPU 개수를 확인한다….

ex)

```
ubuntu@ip-172-31-16-215: ~
ubuntu@ip-172-31-16-215:~$ cat /proc/cpuinfo | grep CPU |  wc -l
1
```

[sysbench, CPU 성능측정 명령어]

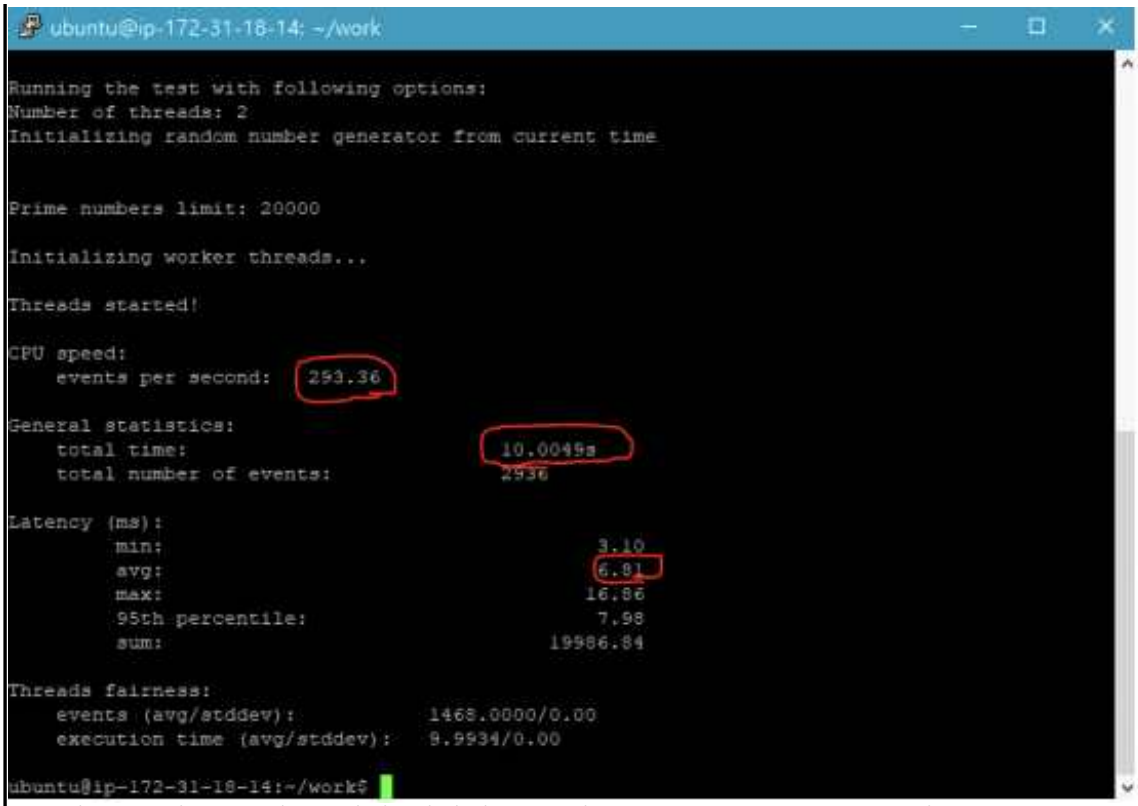
• **< micro 인스턴스 cpu 성능측정 명령어>**
• sysbench cpu --events=10000 --cpu-max-prime=20000 --time=0 --threads=2 run
• **< medium 인스턴스 cpu 성능 측정 명령어>**
• sysbench cpu --events=10000 --cpu-max-prime=20000 --time=0 --threads=4 run
• **<xlarge 인스턴스 cpu 성능 측정 명령어>**
• sysbench cpu --events=10000 --cpu-max-prime=20000 --time=0 --threads=8 run

(CPU 성능측정 명령어 중 --threads=2 run -> threads의 개수는 CPU 개수의 2배로 설정한다.)

< 실험 1-1 : 가상환경에서 인스턴스별 CPU 성능차이 테스트 >

CPU 성능측정

* CPU Speed, Total Time, Latency_avg



(반복 측정을 통해 30개의 데이터를 수집 - medium, xlarge 동일)

< 실험 1-2 : 컨테이너 환경에서 인스턴스별 CPU 성능 차이 테스트 >

▶ Docker 이미지 실행

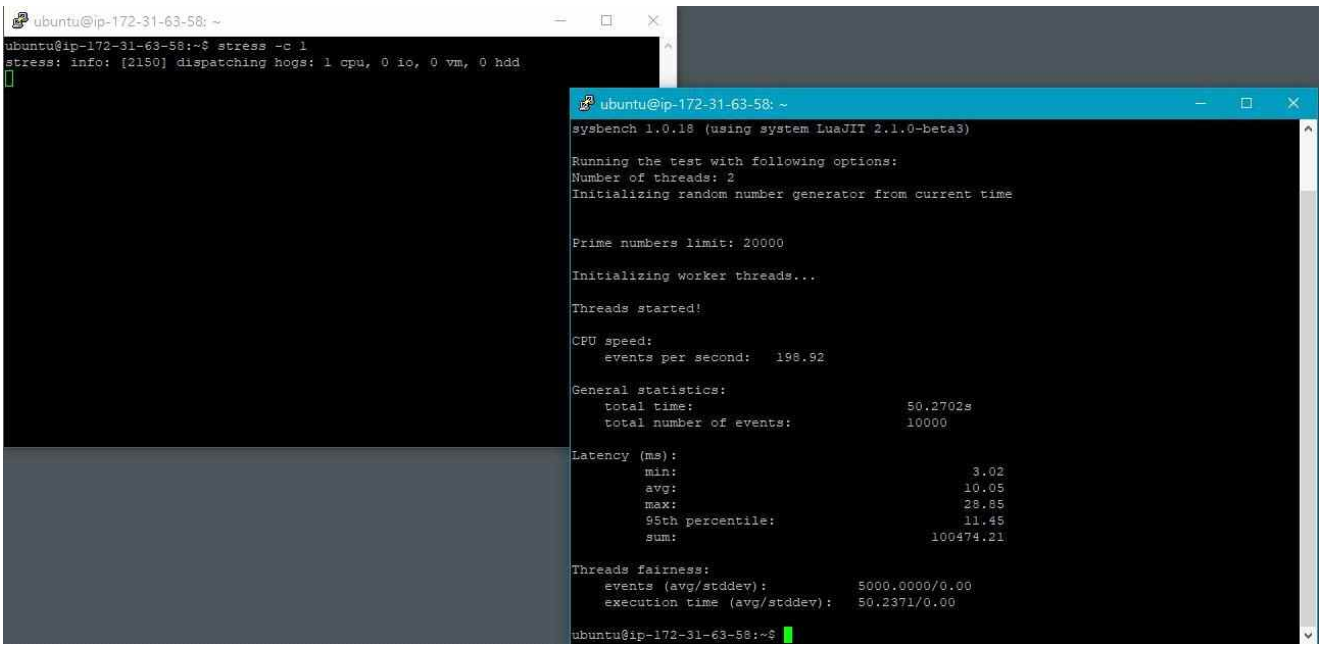
```
ubuntu@ip-172-31-16-184:~$ sudo docker run ubuntu/test:1.0
```

▶[실험1-1] 방법과 같은 방식으로 인스턴스별 데이터 30개 수집.

```
ubuntu@ip-172-31-16-143:~$ sudo docker run ubuntu/test:1.0
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 2
Initializing random number generator from current time


Prime numbers limit: 20000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:   147.29

General statistics:
    total time:                          67.8923s
    total number of events:              10000

Latency (ms):
         min:                                    3.36
         avg:                                   13.57
         max:                                   30.79
         95th percentile:                       15.55
         sum:                                135730.27

Threads fairness:
    events (avg/stddev):           5000.0000/0.00
    execution time (avg/stddev):   67.8651/0.01
```

<실험 2> Stress 부하 한 후 CPU 성능측정

: 실제 프로그램을 돌렸을 때의 상황을 알아보고자 CPU 코어 수 1개만큼의 스트레스를 부하 한 후 각 환경 성능을 비교



▶ stress 스트레스 부하프로그램으로 cpu 코어 1개 수 만큼 스트레스를 부하 한 후 인스턴스별 성능을 측정하고 똑같은 방법으로 컨테이너 환경에서도 성능을 측정하였음.

< 데이터 정리 >

Micro 가상환경			Micro 컨테이너 환경			Micro Stress 가상환경			Micro Stress 컨테이너		
CPU_Speed	Total_time	Latency_avg	CPU_Speed	Total_time	Latency_avg	CPU_Speed	Total_time	Latency_avg	CPU_Speed	Total_time	Latency_avg
294.78	33.9216	6.78	293.47	34.0727	6.81	195.77	51.0798	10.21	147.29	67.8923	13.57
291.56	34.2968	6.85	293.46	34.0742	6.81	195.91	51.0426	10.2	147.29	67.8906	13.57
295.79	33.8055	6.76	294.58	33.9451	6.78	196.16	50.9787	10.19	146.78	68.1253	13.61
295.72	33.814	6.76	294.74	33.9259	6.78	193.21	51.7546	10.34	147.31	67.8834	13.56
295.64	33.8233	6.76	294.24	33.9871	6.79	195.82	51.0646	10.21	147.21	67.9304	13.57
294.68	33.9329	6.78	294.72	33.928	6.78	196.27	50.948	10.18	147.31	67.8806	13.56
295.14	33.8796	6.77	294.3	33.9766	6.79	196.01	51.0162	10.2	147.11	67.9743	13.58
295.01	33.8955	6.78	294.63	33.9388	6.78	196.09	50.9958	10.19	147.18	67.942	13.58
294.99	33.8978	6.77	294.26	33.9821	6.79	195.74	51.087	10.21	147.47	67.8047	13.55
295.56	33.832	6.76	294.54	33.9495	6.78	195.86	51.0548	10.2	147.45	67.8126	13.55
295.18	33.8759	6.77	294.55	33.9481	6.78	195.92	51.0392	10.2	147.41	67.8312	13.56
294.99	33.8978	6.78	293.95	34.0178	6.8	192.79	51.8682	10.37	147.51	67.7841	13.55
294.8	33.9189	6.78	294.12	33.9975	6.8	195.81	51.0679	10.21	147.57	67.7597	13.54
294.96	33.9014	6.78	294.47	33.9574	6.79	195.01	51.2788	10.25	147.18	67.9442	13.57
294.55	33.9485	6.79	295.05	33.8901	6.77	195.83	51.0637	10.21	147.37	67.8538	13.56
295.08	33.8873	6.77	294.57	33.9464	6.79	196.03	51.0107	10.2	147.54	67.7723	13.54
295.72	33.8134	6.76	297.33	33.9736	6.79	195.89	51.0462	10.2	145.56	68.7001	13.73
288.24	34.6911	6.93	293.98	34.0137	6.8	196.09	50.9946	10.19	147.37	67.8487	13.56
294.78	33.9211	6.78	295.07	33.8878	6.77	196.07	51.0012	10.19	147.51	67.7849	13.55
295.02	33.8939	6.78	293.92	34.0212	6.8	195.71	51.0949	10.21	147.27	67.8985	13.57
295.48	33.841	6.76	294.45	33.9592	6.79	195.77	51.0781	10.21	147.37	67.8533	13.56
294.76	33.9235	6.78	294.48	33.9561	6.79	194.06	51.5281	10.3	147.43	67.8286	13.55
294.3	33.9766	6.79	293.53	34.0657	6.81	196.18	50.9706	10.19	146.74	68.147	13.62
295.06	33.8887	6.77	293.91	34.0219	6.8	195.81	51.068	10.21	146.81	68.1098	13.61
294.72	33.9287	6.78	293.78	34.0372	6.8	195.96	51.0282	10.2	145.89	68.5437	13.69
294.9	33.9079	6.78	293.93	34.0198	6.8	195.83	51.0632	10.21	146.45	68.281	13.65
294.6	33.9422	6.79	294.08	34.003	6.8	195.76	51.0797	10.21	146.72	68.1488	13.61
294.73	33.9272	6.78	294.13	33.9971	6.79	195.9	51.0427	10.2	146.83	68.0999	13.62
294.98	33.8983	6.78	294.42	33.9635	6.79	196.15	50.9788	10.19	146.82	68.1105	13.61
294.96	33.9005	6.78	293.6	34.0579	6.81	193.2	51.7547	10.34	146.98	68.0361	13.59

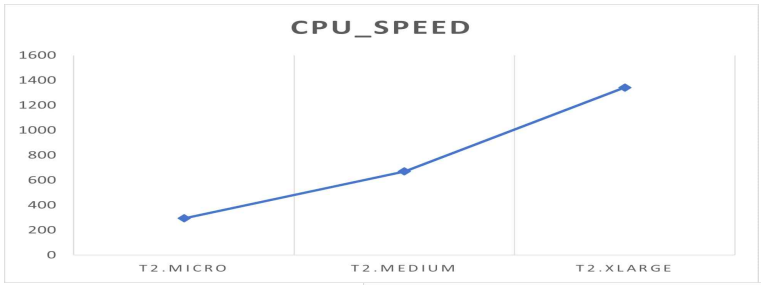
- 다음과 같이 모든 데이터를 R 프로그래밍으로 분석하기 쉽게 csv 파일로 저장.
- medium, xlarge 도 동일한 방법으로 저장

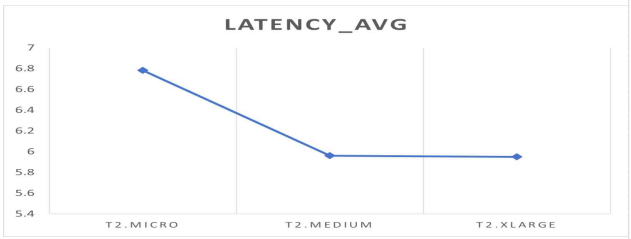
□ 결과

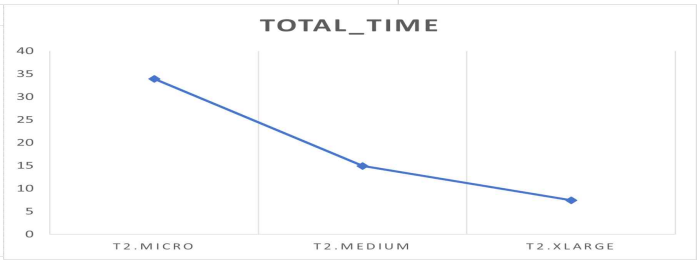
< 데이터 시각화 >

▶ [실험 1]

1. 가상환경에서 인스턴스 유형별 성능 데이터 시각화

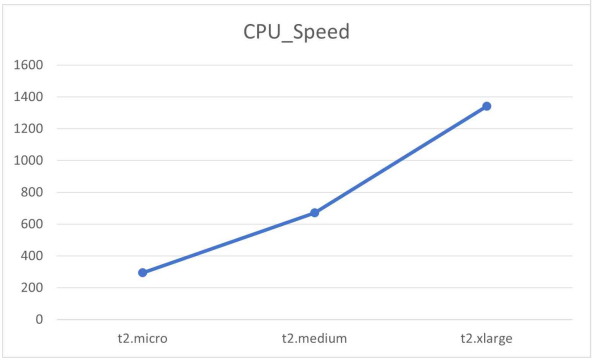


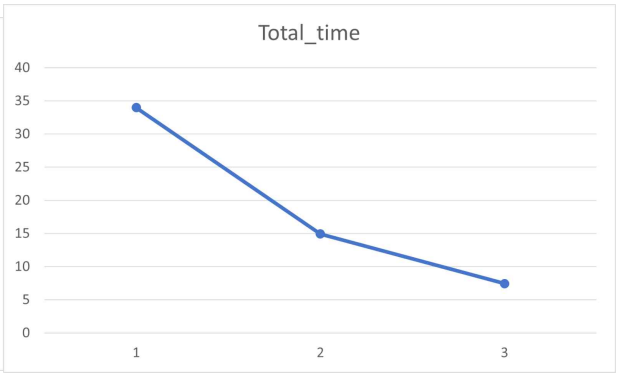


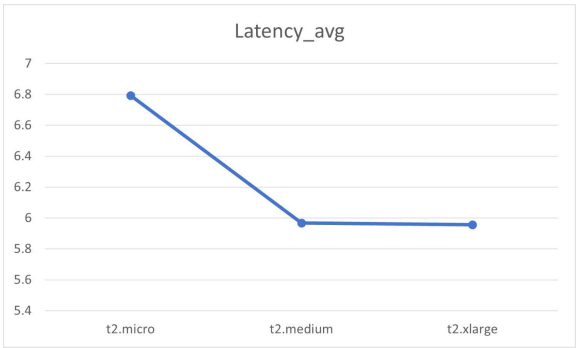


- 다음의 그래프는 가상환경에서의 각 인스턴스 유형 별로 CPU_SPEED, LATENCY_AVG, TOTAL TIME의 평균 비교를 그래프로 나타낸 것이다.

1. 컨테이너 환경에서 인스턴스 유형별 성능 데이터 시각화







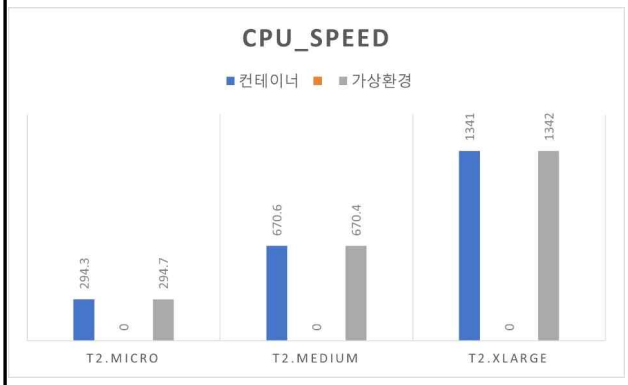
- 다음의 그래프는 컨테이너에서의 각 인스턴스 유형 별로 CPU_SPEED, LATENCY_AVG, TOTAL TIME의 평균 비교를 그래프로 나타낸 것이다.

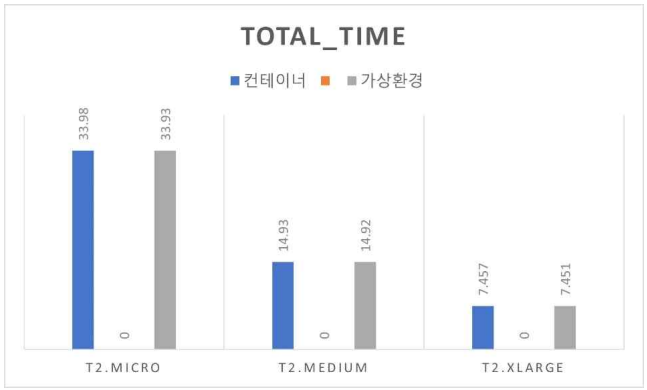
✓ 각 환경에서 인스턴스별 성능 차이를 시각화하였을 때 나타난 결과
- CPU_SPEED는 CPU 코어 수가 높을수록 더 빠른 수치를 가진다. (비례)
- TOTAL_TIME은 CPU 코어 수가 높을수록 더 낮은 시간을 가진다. (반비례)

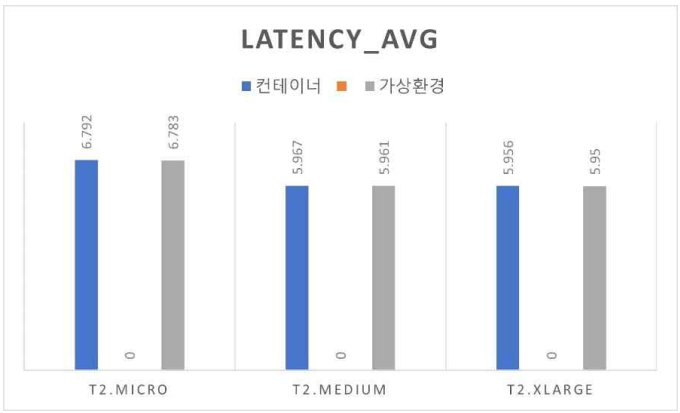
- LATENCY_AVG는 CPU 코어 수가 높을수록 더 낮은 지연시간을 가진다. (반비례)

▶ **[실험 2]**

1. Stress 부하 전 가상환경과 컨테이너 환경의 성능 평균 비교

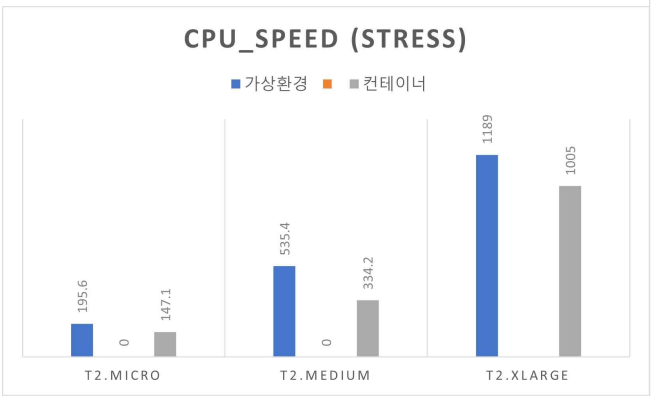


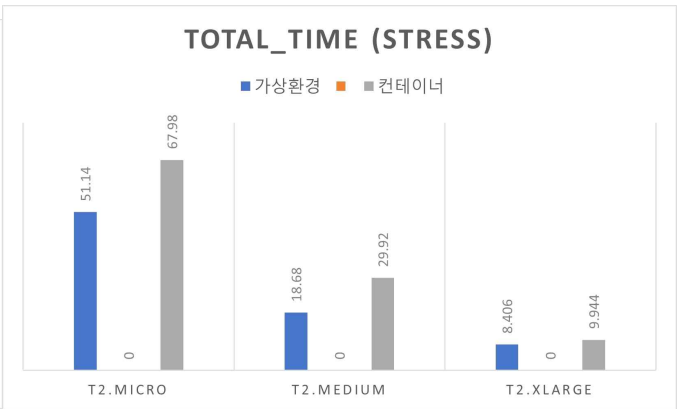


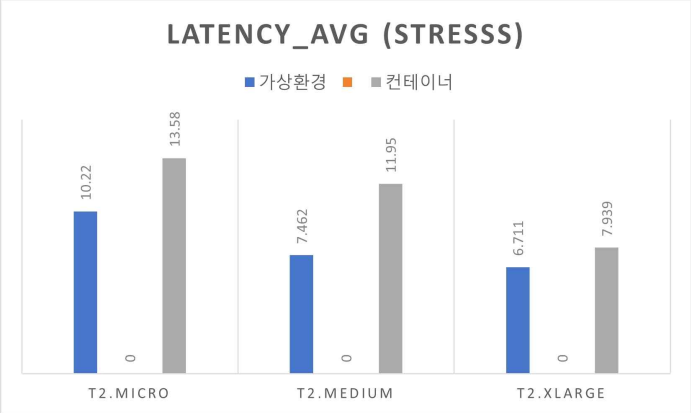


- 성능의 차이가 거의 없음을 확인 할 수 있다.

2. Stress 부하 후 가상환경과 컨테이너 환경의 성능 평균 비교







- 스트레스를 부하하였을 때 컨테이너의 성능이 가상환경에서의 성능보다 떨어짐을 확인한다.

▶ Stress 부하 후 가상환경과 컨테이너 환경의 성능 평균 비교 결과 :

✓ CPU_speed, Total time, Latency_avg 모두 가상환경에서 더 좋은 결과를 나타낸다.

✓ 컨테이너 환경의 한계를 파악해서 가상환경의 성능보다 적은 성능을 보이므로 미리 사전 성능측정을 통해 예상치 못한 오류를 방지한다.

< R 프로그래밍을 사용하여 정밀 분석 >

[실험 1]

1) **가상환경의 인스턴스 유형 별 평균값 차이 검정.** (분산분석)

> CPU speed

```
> cpu_speed_output<-aov(test$CPU_Speed ~ test$Type)
> summary(cpu_speed_output)
             Df   Sum Sq Mean Sq  F value Pr(>F)
test$Type     2 16885786 8442893 10335271 <2e-16 ***
Residuals    87       71       1
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

> Total Time

```
> total_time_output<-aov(test$Total_time ~ test$Type)
> summary(total_time_output)
             Df Sum Sq Mean Sq F value Pr(>F)
test$Type     2  11187    5593  603530 <2e-16 ***
Residuals    87      1       0
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

> Latency_avg

```
> latency_avg_output<-aov(test$Latency_avg ~ test$Type)
> summary(latency_avg_output)
             Df Sum Sq Mean Sq F value Pr(>F)
test$Type     2 13.686   6.843   19285 <2e-16 ***
Residuals    87  0.031   0.000
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

결과)

- **세 가지 성능 모두 귀무가설이 기각되어 유형별로 차이가 있음을 확인.**

1) **컨테이너 환경의 인스턴스 유형 별 평균값 차이 검정.** (분산분석)

> CPU speed

```
> cpu_speed_con_output<-aov(test1$CPU_Speed ~ test1$Type)
> summary(cpu_speed_con_output)
             Df   Sum Sq  Mean Sq  F value Pr(>F)
test1$Type    2 21895334 10947667 23055202 <2e-16 ***
Residuals    87       41        0
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

> Total Time

```
> total_time_con_output<-aov(test1$Total_time ~ test1$Type)
> summary(total_time_con_output)
             Df Sum Sq Mean Sq F value Pr(>F)
test1$Type    2  14073    7037 4208896 <2e-16 ***
Residuals    87      0       0
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

> Latency_avg

```
> latency_avg_con_output<-aov(test1$Latency_avg ~ test1$Type)
> summary(latency_avg_con_output)
             Df Sum Sq Mean Sq F value Pr(>F)
test1$Type    2 13.967   6.983   73584 <2e-16 ***
Residuals    87  0.008   0.000
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

결과)

- <u>세 가지 성능 모두 귀무가설이 기각되어 유형별로 차이가 있음을 확인.</u>

[실험 2]

1) Stres 부하 후 두 환경의 **CPU 성능 차이 분석.** (T-Test 분석)

CPU_Speed)

```
> out<-t.test(test2$t2.micro~test2$Type, var.equal=T)
> out

        Two Sample t-test

data:  test2$t2.micro by test2$Type
t = -252.94, df = 58, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -48.84618 -48.07915
sample estimates:
mean in group con mean in group ec2
         147.0910          195.5537
```

세 가지 인스턴스 유형 모두

P-value : 2.2e-16 이 0에 가까운 결과를 나타냄으로

귀무가설 기각.

결론 : 두 환경의 CPU_Speed평균 차이가 있다.

Total_time:

```
        Two Sample t-test

data:  test2$t2.medium by test2$Type
t = 287.88, df = 58, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 11.16892 11.32533
sample estimates:
mean in group con mean in group ec2
         29.92342          18.67629
```

세 가지 인스턴스 유형 모두

P-value : 2.2e-16 이 0에 가까운 결과를 나타냄으로

귀무가설 기각.

결론 : 두 환경의 Total_time평균 차이가 있다.

Latency_avg:

```
> out2<-t.test(test2$t2.medium~test2$Type, var.equal=T)
> out2

        Two Sample t-test

data:  test2$t2.medium by test2$Type
t = 283.15, df = 58, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 4.458259 4.521741
sample estimates:
mean in group con mean in group ec2
           11.952             7.462
```

세 가지 인스턴스 유형 모두

P-value : 2.2e-16 이 0에 가까운 결과를 나타냄으로

귀무가설 기각.

결론 : 두 환경의 Latency_avg평균 차이가 있다.

결론 : 스트레스 부하 후 두 환경의 성능 차이가 있음을 확인함.

2. 활용방안 및 기대효과

□ 활용방안

- 컨테이너 환경에서 CPU의 성능을 측정하는 연구를 통해 인공지능 및 빅데이터 분야에 도움을 줄 수 있다.
- 컨테이너 환경과 가상 환경에서의 CPU의 성능의 차이의 분석한 통계 연구 결과를 제시할 수 있다.
- 사용자가 필요로 하는 작업량과 요구속도를 알게 되면 적합한 CPU를 고를 수 있어 효율적으로 사용할 수 있다.
- CPU 사용자가 필요로 하는 작업량과 요구속도를 알게 되면 적합한 CPU를 고를 수 있어 효율적으로 사용할 수 있다.
- 사용자가 사용하는 작업량과 요구속도를 계산하는 프로그램을 제작하여 배포하게 되면 본인이 사용하는 작업량과 요구속도를 어렵지 않게 구함으로써 시너지 효과를 내어 더욱 쉽게 선택하고 효율적으로 사용할 수 있을 것이다.
- 컨테이너의 한계를 미리 예측하고 예상치 못한 오류를 방지할 수 있다.

□ 기대효과

- 사전의 성능측정을 통해서 측정 결과를 기반으로 하여 최소한의 개발을 할 수 있다.
- 초기 투자 및 운영에 필요한 비용을 절감할 수 있다.
- 불필요한 자원 사용을 방지할 수 있다.
- 서비스의 신뢰도를 향상할 수 있다.
- 불필요한 자원 사용을 방지할 수 있지만, 가상환경보다 컨테이너 환경이 더 적은 성능을 보이기 때문에 오히려 성능 부족이라는 부작용을 낼 수 있다. 이를 위해 성능 차이를 계산하고 예측함으로써 성능 부족이라는 불가피한 상황을 방지할 수 있을 것이다.

□ 기타

<벤치마킹 프로그램 sysbench>

- sysbench는 인텐시브한 로드에서 데이터베이스를 운영해야하는 시스템에 필요한 운영체제를 평가하는 모듈화된 크로스 플랫폼 멀티 쓰레드 벤치마크 도구이다.

<Docker>

- 도커는 리눅스의 응용 프로그램들을 프로세스 격리 기술들을 사용해 컨테이너로 실행하고 관리하는 오픈 소스 프로젝트이다. 도커 컨테이너는 일종의 소프트웨어를 소프트웨어의 실행에 필요한 모든 것을 포함하는 완전한 파일 시스템 안에 감싼다.