Docker CPU, Memory 리소스 제한 및 부하 테스트

이 문서는 Docker CPU, Memory 리소스 제한 및 부하 테스트 내용을 공유하기 위해 작성되었다.

도구명	docker-compose
버전	2.16.0
OS	centOS 8
비고	

---

docker-compose CPU 제한 및 부하 테스트

1. 목표 설정

Docker Compose에서 실행되는 컨테이너의 CPU 사용률과 메모리 사용률 테스트

2. 테스트 구성

docker-compose upgrade

- docker-compose deploy의 resource를 사용하기 위해 docker-compose upgrade

> docker-compose 버전이 1.X 일 경우 해당 부분 지원하지 않음
>
> #docker-compose 버전 확인
> docker-compose -v

```
sudo rm /usr/local/bin/docker-compose
# curl + grep
VERSION=$(curl --silent https://api.github.com/repos/docker/compose/releases/latest | grep -Po '"tag_name": "\K.*\d')

# curl + jq
VERSION=$(curl --silent https://api.github.com/repos/docker/compose/releases/latest | jq .name -r)
DESTINATION=/usr/local/bin/docker-compose
sudo curl -L https://github.com/docker/compose/releases/download/${VERSION}/docker-compose-$(uname -s)-$(uname -m) -o $DESTINATION
sudo chmod 755 $DESTINATION
```

docker-compose.yml 예시

```
services:
  service_name:
    ...
    deploy:
      resources:
        limits:
          cpus: '0.5' # half a CPU
          memory: 256M # 256 MB of RAM
        reservations:
          cpus: '0.25' # a quarter of a CPU
          memory: 128M # 128 MB of RAM
```

docker-compose 실행 명령어

```
docker-compose -f do-limits.yml up -d
```

docker 상태 확인

```
docker stats [OPTIONS] [CONTAINER...]
```

```
CONTAINER ID   NAME         CPU %     MEM USAGE / LIMIT   MEM %     NET I/O       BLOCK I/O         PIDS
f11f3382718c   postgres12   0.04%     26.79MiB / 1GiB     2.62%     6.77kB / 0B   0B / 100MB        7
7d6ac39ad2b5   test_3       0.00%     3.988MiB / 1GiB     0.39%     6.73kB / 0B   16.4kB / 12.3kB   5
e0e3efee15d0   test_1       0.00%     3.977MiB / 256MiB   1.55%     6.73kB / 0B   8.19kB / 43kB     5
```

3. 테스트

- **컨테이너 내부 접속 및 필요 command 설치**

```
#
docker exec -it [Name  Container ID] /bin/bash

#
exit

#OS
cat /etc/*-release
apt-get update &&
apt-get -y install vim &&
apt-get -y install stress
```

- **Container 내부 bash 접속 후 shell script 부하 테스트**
  docker container 내부에 접속, 해당 스크립트 작성 후 실행

cpu 부하주는 bash 스크립트

```
#vi cpu_stress.sh
#chmod 777 cpu_stress.sh

#!/bin/bash

# Function to check if a number is prime
is_prime() {
  local number=$1
  for ((i=2; i<=$((number/2)); i++)); do
    if [ $((number%i)) -eq 0 ]; then
      return 1
    fi
  done
  return 0
}

# Infinite loop to calculate prime numbers
while true; do
  for ((i=2; i<=100000; i++)); do
    if is_prime $i; then
      echo $i
    fi
  done
done
```

memory 부하주는 bash 스크립트

```
#vi mem_stress.sh
#chmod 777 mem_stress.sh

#!/bin/bash

# Run the memory stress test loop
while true
do
        #head -c ${memory}k </dev/zero> /dev/null
        </dev/zero head -c 512m | tail
        #</dev/zero head | tail
        #(</dev/zero head -c 512m) <(sleep 300) | tail
done
```

- **Container 내부 bash 접속 후 stress 명령어 부하 테스트**
  docker container 내부 접속 후 stress 명령어 실행

```
#CPU
stress --cpu 2 --timeout 20s

#Memory
stress --verbose --vm 1 --vm-bytes 512M --timeout 20s
```

- **docker-compose.yml entrypoint, command 부하 테스트**
  entrupoint, command: container 실행 시 실행되는 명령어(entrypoint > command)

```
version: "3"
services:
  web1:
    container_name: test_1
    image: nginx
    #command: sh -c "apt update&&apt-get install vim -y&&apt-get install stress -y&&stress --verbose --vm 1 --
vm-bytes 512M"
    entrypoint: sh -c "apt update&&apt-get install vim -y&&apt-get install stress -y&&stress --verbose --vm 1 --
vm-bytes 512M"
    ports:
      - "8083:8080"
    deploy:
      resources:
        limits:
          cpus: '1' # half a CPU
          memory: 256M # 256 MB of RAM
        reservations:
          cpus: '0.5' # a quarter of a CPU
          memory: 256M # 128 MB of RAM
  db:
    image: postgres:12
    container_name: postgres12
    environment:
      POSTGRES_USER: sonar
      POSTGRES_PASSWORD: sonar
    deploy:
      resources:
        limits:
          cpus: '1.5' # half a CPU
          memory: 1024M # 256 MB of RAM
        reservations:
          cpus: '0.6' # a quarter of a CPU
          memory: 256M # 128 MB of RAM
  web3:
    container_name: test_3
    image: nginx
    ports:
      - "8082:8080"
    deploy:
      resources:
        limits:
          cpus: '1.5' # half a CPU
          memory: 1024M # 256 MB of RAM
        reservations:
          cpus: '0.6' # a quarter of a CPU
          memory: 256M # 128 MB of RAM
```

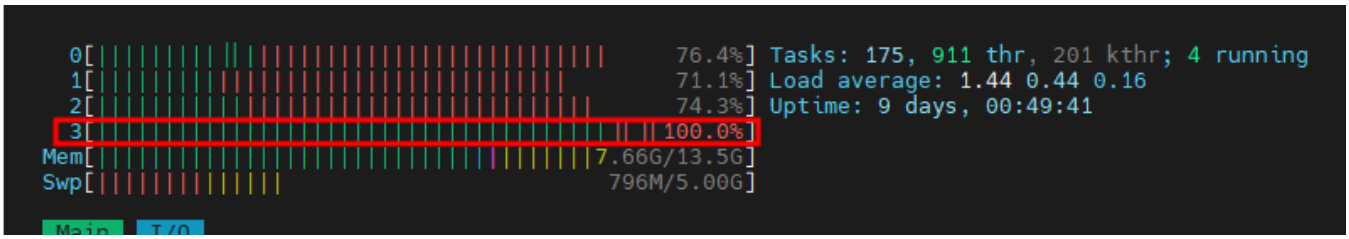
4. 부하 테스트 수행 결과

4-1. Container 내부 bash 접속 후 shell script 부하 테스트

- CPU 부하 스크립트 결과-1(docker stats)

```
CONTAINER ID   NAME         CPU %     MEM USAGE / LIMIT   MEM %     NET I/O          BLOCK I/O         PIDS
f11f3382718c   postgres12   0.00%     28.07MiB / 1GiB     2.74%     8.22kB / 0B      0B / 100MB        7
7d6ac39ad2b5   test_3       0.00%     3.988MiB / 1GiB     0.39%     8.18kB / 0B      16.4kB / 12.3kB   5
e0e3efee15d0   test_1       99.43%    29.19MiB / 256MiB   11.40%    16.9MB / 77.2kB  6.46MB / 4.55MB   7
```

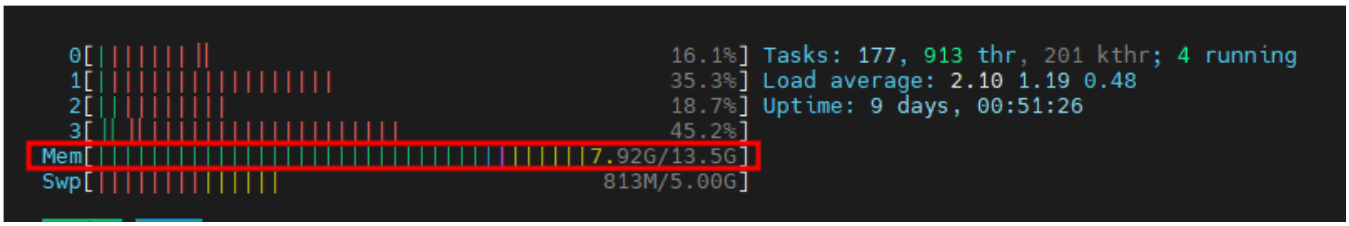
- CPU 부하 스크립트 결과-2(htop - 호스트)



- Memory 부하 스크립트 결과-1(docker stats)

```
CONTAINER ID   NAME         CPU %     MEM USAGE / LIMIT   MEM %     NET I/O          BLOCK I/O         PIDS
f11f3382718c   postgres12   0.00%     28.2MiB / 1GiB      2.75%     8.58kB / 0B      0B / 100MB        7
7d6ac39ad2b5   test_3       0.00%     3.988MiB / 1GiB     0.39%     8.54kB / 0B      16.4kB / 12.3kB   5
e0e3efee15d0   test_1       16.11%    255.7MiB / 256MiB   99.89%    16.9MB / 77.2kB  6.71MB / 4.61MB   9
```

- Memory 부하 스크립트 결과-2(htop-호스트)



- 부하에 따른 script 부하 kill

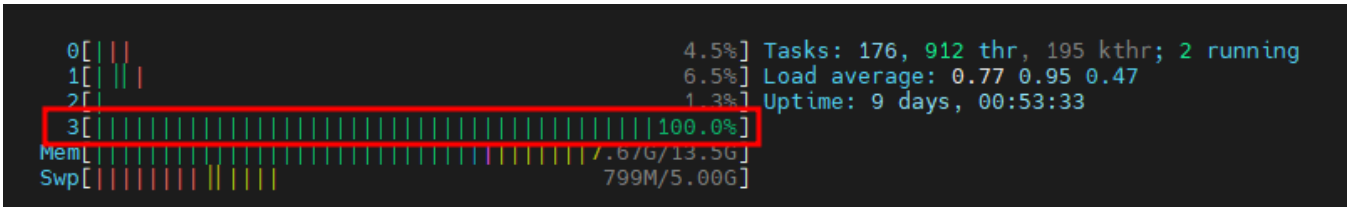
```
./mem.sh: line 10:   466 Broken pipe             head -c 512m < /dev/zero
      467 Killed                  | tail
./mem.sh: line 10:   468 Broken pipe             head -c 512m < /dev/zero
      469 Killed                  | tail
./mem.sh: line 10:   470 Broken pipe             head -c 512m < /dev/zero
      471 Killed                  | tail
```

4-2. Container 내부 bash 접속 후 stress 명령어 부하 테스트

- CPU 부하 command: `stress --cpu 2 --timeout 20s`

```
CONTAINER ID   NAME         CPU %     MEM USAGE / LIMIT   MEM %     NET I/O          BLOCK I/O         PIDS
f11f3382718c   postgres12   0.00%     28.32MiB / 1GiB     2.77%     8.58kB / 0B      0B / 100MB        7
7d6ac39ad2b5   test_3       0.00%     3.988MiB / 1GiB     0.39%     8.54kB / 0B      16.4kB / 12.3kB   5
e0e3efee15d0   test_1       99.25%    2.004MiB / 256MiB   0.78%     16.9MB / 77.2kB  23.7MB / 4.61MB   9
```

- Memory 부하 command: `stress --verbose --vm 1 --vm-bytes 512M`



- 부하에 따른 stress 명령 kill #stress무한루프 : `stress --verbose --vm 1 --vm-bytes 512M`

```
root@da1c5ca873bb:/# stress --verbose --vm 1 --vm-bytes 512M --timeout 20s
stress: info: [319] dispatching hogs: 0 cpu, 0 io, 1 vm, 0 hdd
stress: dbug: [319] using backoff sleep of 3000us
stress: dbug: [319] setting timeout to 20s
stress: dbug: [319] --> hogvm worker 1 [320] forked
stress: dbug: [320] allocating 536870912 bytes ...
stress: dbug: [320] touching bytes in strides of 4096 bytes ...
stress: FAIL: [319] (415) <-- worker 320 got signal 9
stress: WARN: [319] (417) now reaping child worker processes
stress: FAIL: [319] (451) failed run completed in 2s
```

4-3. docker-compose.yml entrypoint, command 부하 테스트

- docker-compose.yml entrypoint, command 부하 테스트

```
CONTAINER ID   NAME         CPU %     MEM USAGE / LIMIT   MEM %     NET I/O          BLOCK I/O         PIDS
da1c5ca873bb   test_1       0.00%     3.984MiB / 256MiB   1.56%     7.86kB / 0B      8.19kB / 25.6kB   5
00b4f8e340c7   postgres12   24.49%    1024MiB / 1GiB      99.99%    9.09MB / 48.7kB  337MB / 101MB     10
6f24d0c81eca   test_3       0.00%     4.02MiB / 1GiB      0.39%     7.69kB / 0B      8.19kB / 43kB     5
```

- while true;do timeout 2s bash -c "docker stats" >> test.txt;done #test.txt 파일

```
c070873177d4   test_3       0.00%     3.961MiB / 1GiB     0.39%     7.35kB / 0B      8.19kB / 34.8kB   5
2c8f258eaf73   test_1       0.00%     3.969MiB / 256MiB   1.55%     7.35kB / 0B      8.19kB / 49.2kB   5
993aa1247e50   postgres12   0.00%     1024MiB / 1GiB      99.98%    17.3MB / 98.9kB  73.4MB / 4.75MB   3
^[[2J^[[HCONTAINER ID   NAME         CPU %     MEM USAGE / LIMIT   MEM %     NET I/O          BLOCK I/O         P
IDS
c070873177d4   test_3       0.00%     3.961MiB / 1GiB     0.39%     7.35kB / 0B      8.19kB / 34.8kB   5
2c8f258eaf73   test_1       0.00%     3.969MiB / 256MiB   1.55%     7.35kB / 0B      8.19kB / 49.2kB   5
993aa1247e50   postgres12   0.00%     1024MiB / 1GiB      99.98%    17.3MB / 98.9kB  73.4MB / 4.75MB   3
^[[2J^[[HCONTAINER ID   NAME         CPU %     MEM USAGE / LIMIT   MEM %     NET I/O          BLOCK I/O         PIDS
c070873177d4   test_3       0.00%     3.961MiB / 1GiB     0.39%     7.35kB / 0B      8.19kB / 34.8kB   5
2c8f258eaf73   test_1       0.00%     3.969MiB / 256MiB   1.55%     7.35kB / 0B      8.19kB / 49.2kB   5
```

5. 결과 분석

- **cpu의 경우**

1. cpu의 경우 컨테이너 종료 없이 풀 차지

- **memory의 경우**

1. memory의 경우 단순 stress 명령어나 스크립트로 부하를 줄 경우 limit 설정한 임계 값에 도달 할 시 stress 명령이나 스크립트가 종료처리 됨(kill)
    - 컨테이너는 종료되지 않음
2. docker-compose.yml에 entrypoint로 stress 명령을 줄 경우 임계 값에 도달 시 stress 명령이 종료 처리되면서 컨테이너가 종료됨
3. stress명령 및 스크립트로 postgresql에 계속적인 부하를 준 경우 postgresql 성능은 매우 느려지나 컨테이너는 안꺼짐

참조

docker-compose 버전 upgrade

- Docker-compose upgrade

memory script

- https://unix.stackexchange.com/questions/99334/how-to-fill-90-of-the-free-memory

stress 명령어

- https://klero.tistory.com/entry/%EB%A6%AC%EB%88%85%EC%8A%A4-stress-%ED%88%B4%EC%9D%84-%ED%86%B5%ED%95%B4-CPU-Memory-%EC%8A%A4%ED%8A%B8%EB%A0%88%EC%8A%A4-%EB%B6%80%ED%95%98-%EC%A3%BC%EB%8A%94-%EB%B0%A9%EB%B2%95