

# 클라우드 컴퓨팅 - HW2

2015410012 김현섭

1. C파일 상단에 gcc 명령어 주석 넣기

```
1 // gcc -o dav dav.c
2 #include <stdio.h>
3 #include <string.h>
4 #include <time.h>
5 #include <stdlib.h>
```

2. 모듈 구동 전과 후의 sysbench 결과 비교

모두 vCPU 2개 할당하고 시작 했음.

```
root@xen-VirtualBox: /etc/xen
Doing CPU performance benchmark
Threads started!
Done.
Maximum prime number checked in CPU test: 60000

Test execution summary:
total time: 101.1667s
total number of events: 10000
total time taken by event execution: 101.0201
per-request statistics:
  min: 8.23ms
  avg: 10.10ms
  max: 31.69ms
  approx. 95 percentile: 10.72ms

Threads fairness:
  events (avg/stddev): 10000.0000/0.00
  execution time (avg/stddev): 101.0201/0.00
user@ubuntu:~$
```

```
root@xen-VirtualBox: /etc/xen
Doing CPU performance benchmark
Threads started!
Done.
Maximum prime number checked in CPU test: 60000

Test execution summary:
total time: 51.3289s
total number of events: 10000
total time taken by event execution: 205.0402
per-request statistics:
  min: 9.04ms
  avg: 20.50ms
  max: 84.74ms
  approx. 95 percentile: 27.67ms

Threads fairness:
  events (avg/stddev): 2500.0000/4.53
  execution time (avg/stddev): 51.2601/0.02
user@ubuntu:~$
```

모듈 구동 전 - Thread 1개짜리 VM: 101초, Thread 4개짜리 VM: 51초

```
root@xen-VirtualBox: /etc/xen
Doing CPU performance benchmark
Threads started!
Done.
Maximum prime number checked in CPU test: 60000

Test execution summary:
total time: 109.1714s
total number of events: 10000
total time taken by event execution: 109.0327
per-request statistics:
  min: 8.37ms
  avg: 10.90ms
  max: 95.76ms
  approx. 95 percentile: 12.67ms

Threads fairness:
  events (avg/stddev): 10000.0000/0.00
  execution time (avg/stddev): 109.0327/0.00
user@ubuntu:~$
```

```
root@xen-VirtualBox: /etc/xen
Doing CPU performance benchmark
Threads started!
Done.
Maximum prime number checked in CPU test: 60000

Test execution summary:
total time: 36.8690s
total number of events: 10000
total time taken by event execution: 147.3285
per-request statistics:
  min: 7.96ms
  avg: 14.73ms
  max: 184.22ms
  approx. 95 percentile: 38.66ms

Threads fairness:
  events (avg/stddev): 2500.0000/2.74
  execution time (avg/stddev): 36.8321/0.00
user@ubuntu:~$
```

모듈 구동 후 - Thread 1개 VM: 109초, Thread 4개 VM: 36초

결론: Thread 1개 VM은 실행시간이 조금 늘어났지만, Thread 4개 VM은 실행시간이 대폭 줄어듦

### 3. var 구조체 변경

```
13 //this code needs to running vm more than one
14 struct var{
15     char name[10]; // name
16     int id;
17     char state[8]; // state
18     int cpu_sec; // execution time
19     double cpu_per; // utilization
20     int vcpu;
21 }; //vm info structure
22 struct var topvar[VM_NUM];
```

vcpu라는 vCPU의 개수를 담는 변수를 하나 만들었음

이 구조체를 VM 개수만큼 배열로 만들어서 전역 변수에 선언해 둬

### 4. main 함수 설명

```
int main() {
    FILE *openxentop = popen("xentop -b d 1", "r");

    struct tm *t;
    time_t timer;

    while(1) {
        timer = time(NULL);
        t = localtime(&timer);
        printf("\n[Time] %04d-%02d-%02d %02d:%02d:%02d\n",
            t->tm_year+1900, t->tm_mon+1, t->tm_mday, t->tm_hour, t->tm_min, t->tm_sec);

        get_xllist();
        get_xentop(openxentop);
        print_topvar();
        dvam();
        sleep(3);
    }

    pclose(openxentop);

    return 0;
}
```

- 1) time과 localtime이라는 함수를 이용하여 현재 시간을 출력
- 2) get\_xllist, get\_xentop 함수에서 VM에 관한 정보를 가져와서 topvar라는 전역변수에 저장
- 3) print\_topvar라는 함수로 VM에 관한 정보를 출력하고 dvam 함수로 vCPU 재할당

## 5. vCPU 할당 알고리즘 설명

```
98 void dvam() {
99     char command[64];
100     int i = 0;
101     for(i = 0; i < VM_NUM; i++) {
102         if(strcmp(topvar[i].state, "-----r") != 0) // If VM is not running, then no re-allocation
103             continue;
104
105         int now_vcpu = topvar[i].vcpu;
106         int new_vcpu = topvar[i].vcpu;
107         double max_cpu_per = 100.0 * now_vcpu;
108
109         if(max_cpu_per - topvar[i].cpu_per < 10.0 && now_vcpu < MAX_VCPU)
110             new_vcpu = now_vcpu + 1;
111
112         if(max_cpu_per - topvar[i].cpu_per > 100.0 && now_vcpu > MIN_VCPU)
113             new_vcpu = now_vcpu - 1;
114
115         if(now_vcpu != new_vcpu) {
116             sprintf(command, "xl vcpu-set %s %d", topvar[i].name, new_vcpu);
117             pclose(popen(command, "r"));
118             printf("[DVAM] %s vcpu changed:%d -> %d\n", topvar[i].name, now_vcpu, new_vcpu);
119         } else {
120             printf("[DVAM] %s vcpu not changed:%d\n", topvar[i].name, now_vcpu);
121         }
122     }
123 }
```

현재의 vCPU 개수에 100을 곱해 현재 vCPU 개수에서의 가능한 최대 CPU 사용률을 계산

1. 현재 CPU 사용률과 최대 CPU 사용률의 차이가 10퍼센트보다 작으면 거의 CPU를 최대로 활용하고 있다는 뜻이므로 vCPU 하나를 더 할당
2. 현재 CPU 사용률과 최대 CPU 사용률의 차이가 100퍼센트보다 크면 vCPU 하나가 높고 있다는 뜻이므로 vCPU 하나를 뺀