Vol.6, No.12 pp.479~486 pISSN: 2287-5891 https://doi.org/10.3745/KTCCS.2017.6.12.479

The Optimization Mechanism of CPU/GPU Computing Resource for Minimization of Performance Interference and Calculation Efficiency in Volunteer Computing Environment

Bong Woo Bak† · Chung Geon Song†† · Heon Chang Yu†††

ABSTRACT

Volunteer computing is a new computing paradigm that performs operations on idle resources of many nodes. The operation method of the client application for the execution of the volunteer computing is determined by the setting information of the user. Ideal operation requires optimized settings for system features and operating methods of other applications. In this paper, we analyze the usage ratio of CPU and GPU periodically, and develop a manager that dynamically applies optimized options. Through our proposed mechanism, the performance of the task computing is higher than that of the existing Volunteer Computing, and the performance interference is minimized. It is expected that volunteers will be able to provide higher computing resources for Volunteer Computing Project.

Keywords : Volunteer Computing, Resource Management, Task Scaling

볼런티어 컴퓨팅 환경에서 성능간섭 최소화와 연산 효율성 증대를 위한 CPU/GPU 컴퓨팅 자원 최적화 기법

박 봉 우† · 송 충 건†† · 유 헌 창†††

요 약

볼런티어 컴퓨팅(Volunteer Computing)은 많은 노드들의 유휴자원을 이용하여 연산을 수행하는 새로운 컴퓨팅 패러다임이다. 볼런티어 컴퓨팅 수행을 위해 운영하는 클라이언트 어플리케이션은 사용자의 설정 정보에 의해 동작 방식이 결정된다. 이상적인 동작을 위해서는 시스템 특징과 다른 어플리케이션의 동작 방식에 최적화된 설정이 요구된다. 본 연구에서는 유휴 자원 정보를 주기적으로 CPU와 GPU의 사용 비율을 분석하고 최적화된 옵션을 정해 동적으로 적용하는 관리자를 개발하였다. 또한 CPU 자원의 높은 활용도를 위해 태스크 스케일링을 진행하고 CPU코어를 주기적으로 재 할당 하여 CPU자원이 균등하게 사용되게 하였다. 제시하는 기법을 통해 기존의 볼런티어 컴퓨팅보다 높은 태스크 연산 능력을 보였으며 성능간섭 또한 최소화 시켰다. 볼런티어 컴퓨팅을 진행하는데 있어 볼런티어들이 더 높은 컴퓨팅 자원을 제공할 수 있게 될 것으로 예상한다.

키워드 : Volunteer Computing, Resource Management, Task Scaling

1. 서 론

분산 컴퓨팅 환경의 발전과 활용으로 대규모 연산을 이기종 디바이스에서 병렬적으로 처리하는 기술이 성행하고 있다. 하지만 일반적인 분산 컴퓨팅 환경은 많은 비용을 요구하기 때문에 환경을 구성하는데 어려움이 있다. 볼런티어 컴퓨팅(Volunteer Computing)[1]은 볼런티어들의 자발적인 참여로 유휴자원을 제공받기 때문에 자원 확보에 대한 추가적인 비용 없이 대규모 컴퓨팅 연산을 수행할 수 있다. 또한 CPU의 연산능력이 증대되고 GPU를 활용한 GPGPU기술이 연구됨에 따라 그래픽 연산을 넘어 일반적인 컴퓨팅 연산을 효율적으로 수행 할 수 있게 되어 일반 사용자들이 제공할 수 있는 자원이 증가하였고 방대한 작업을 할 수 있게 되었다.

하지만 유휴자원이 발생하지 않을 수도 있고 네트워크의

※ 2017년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임(No.2017-0-00587(기반SW-창조씨앗1단계) 분산 클라우드 기반 유무선 컴퓨팅 시스템의 유휴자원 공유 플랫폼 개발).
† 준 회 원 : 고려대학교 컴퓨터학과 석사과정
†† 준 회 원 : 고려대학교 컴퓨터학과 박사과정
††† 종신회원 : 고려대학교 컴퓨터학과 교수
Manuscript Received : October 31, 2017
Accepted : November 14, 2017
* Corresponding Author : Heon Chang Yu(yuhc@korea.ac.kr)

상황에 따라 태스크 또는 연산의 결과가 유실될 수 있으며 위탁한 태스크의 연산이 언제 완료될지 불분명하기 때문에 볼런티어 컴퓨팅의 자원은 안정적이지 않아 대규모 자원을 확보하는데 어려움이 있다.

이와 같은 문제점을 해결하기 위해 본 연구는 볼런티어 유저 어플리케이션에 성능간섭을 최소화하고 프로젝트 별 특성을 고려한 최적의 CPU/GPU의 사용률을 도출하여 볼런티어 컴퓨팅의 연산효율을 향상시키는 기법을 제안한다.

제안하는 기법에서는 프로젝트 특화 된 CPU와 GPU 사용률을 제시하여 최적의 설정을 도출하여 자원을 제공함으로 볼런티어 컴퓨팅의 효율을 증대 시킨다. 또한 태스크 스케일링과 CPU코어 재 할당 기법을 제안하여 유저 어플리케이션의 성능간섭을 최소화 시킨다. 본 연구의 목표는 프로젝트 특화된 최적의 설정 값을 찾음으로써 볼런티어 환경에서 적은 수의 볼런티어만으로도 많은 자원을 확보하는 것과 유휴자원 뿐만 아니라 더 많은 자원을 기부하면서 성능간섭을 최소화 시켜, 안정적인 볼런티어 컴퓨팅 환경을 구축하는 것이다.

본 논문의 구성으로는 2장에서는 관련연구에 대해 설명하고, 3장에서는 제안하는 기법에 대해 설명하고, 4장에서는 실험의 방식과 실험결과에 대해 분석하고 평가 하고 5장에서는 결론과 향후 연구에 대해 설명한다.

2. 관련 연구

볼런티어 컴퓨팅은 볼런티어들의 유휴자원을 이용해 대규모 연산을 하는 분산 컴퓨팅 환경이다. 볼런티어들은 유휴자원을 제공함으로써 볼런티어 컴퓨팅 프로젝트에게 대규모 연산을 수행하게 해준다. 대규모 컴퓨팅 자원을 필요로 하는 볼런티어 컴퓨팅 프로젝트는 볼런티어들에게 수행할 태스크들을 분배하여 그 결과 값을 반환 받아 연산을 수행한다. 볼런티어들은 해당 각각의 설정에 맞게 유휴자원을 제공하며 그 대가로 연산을 수행한 만큼 크래딧을 부여받는다. 볼런티어의 구성은 Fig. 1과 같다. 프로젝트는 하나의 볼런티어 컴퓨팅 서버를 가지고 있고 수행해야 하는 연산을 태스크 단위로 만들어 작업을 수행할 수 있는 볼런티어들에게 분배한다. 볼런티어는 유휴자원을 사용하여 태스크를 수행하고 그에 따른 결과 값을 서버에게 반환하고 프로젝트는 이러한 반환 값들을 조합하여 원하는 연구의 결과를 도출한다.

볼런티어 컴퓨팅의 장점은 비용을 들이지 않고 대규모 컴퓨팅 자원을 사용할 수 있다는 것이다. 해당 프로젝트에 참여하는 볼런티어들이 많을수록 태스크 수행에 활용 가능한 자원이 늘어나는 것이다. 하지만 볼런티어의 유휴자원을 제공 받기에 자원에 대한 불확실성이 존재한다. 보장 하지 못한다. 이는 안정적으로 제공되는 자원이 아고 또한 볼런티어의 익명성으로 인해 신뢰하지 못하는 태스크의 결과 값이 반환될 위험도 있다. 그렇기 때문에 많은 컴퓨팅 자원을 확보하기 위해서는 많은 볼런티어들이 참여해야하고 유휴자원이 많이 발생해야 한다.

이러한 볼런티어 컴퓨팅 프로젝트 중 본 논문에서는 볼런티어 컴퓨팅을 활용한 대표적인 두 개의 프로젝트의 태스크를 이용하여 실험을 한다. 그 프로젝트 중 첫 번째는 SETI@home이다. SETI@home(Search for Extra-Terrestrial I-ntelligence)[2]은 분산 컴퓨팅 기술을 활용하여 전파 망원경으로 수집한 좁은 대역폭의 시그널 데이터를 분석해 외계 지적 생명체를 탐구하는 프로젝트이다. BOINC 플랫폼에 속해있으며 가장 많은 볼런티어들을 가지고 있는 프로젝트이다. SETI@home은 푸리에 변환을 통해 주파수 해상도를 높이고 그로부터 아주 좁은 주파수 대역에서의 특이 신호를 찾는 연산을 볼런티어들에게 위탁한다.

두 번째 프로젝트는 Einstein@home이다. Einstein@home [3]은 SETI@home과 같이 BOINC 기반의 분산 컴퓨팅 프로젝트 중 하나이다. Einstein@home의 목적은 전천을 대상으로 중력파의 존재를 실증하는 것이다. 아인슈타인의 일반상대성이론에 따르면 모든 질량을 가진 물체는 중력파를 발생하게 되어있고 특히 펄서나 블랙홀 등과 같이 비정상적으로 큰 중력을 가진 물체들의 경우에는 더 큰 중력파를 발생하게 되어있지만 아직까지 중력파가 존재한다는 명확한 증거는 찾지 못하였다. Einstein@home은 LIGO나 GEO로부터 넘겨받은 데이터를 볼런티어들에게 분배하여 고속 푸리에 변환을 이용하여 연산하고 결과 값을 분석한다.

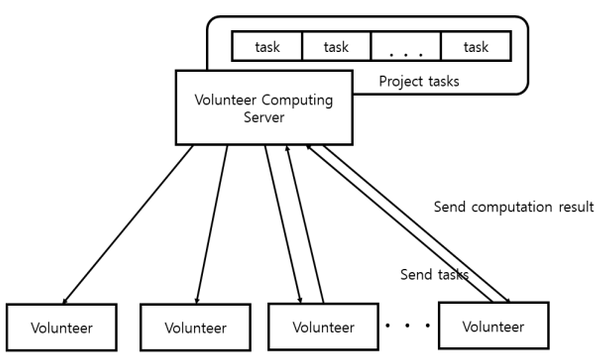

Fig. 1. Structure of Volunteer Computing

이러한 프로젝트의 수가 증가되면서 볼런티어 컴퓨팅 환경을 더욱 효율적으로 수행하기 위한 연구들이 성행되었다. [4]는 프로젝트로부터 받아온 태스크들에 대한 빠른 응답과 높은 컴퓨팅 자원 기부를 위해 스케줄링 방식을 차별화 한 연구이다. 프로젝트로부터 받아 올 수 있는 태스크들과 이미 받고 실행 가능한 태스크들을 기존의 방식과는 다르게 스케줄링 하여 데드라인을 지키고 연산 효율을 증대 시킨 연구이다. 하지만 본 연구에서는 프로젝트로부터 받은 태스크들의 특성을 고려하지 않았다.

[5]에서는 다음 4개의 정책을 달리하였을 때 볼런티어 컴퓨팅 성능의 변화를 연구하였다.

1) CPU scheduling : 수행 가능한 태스크들 중에서 어떤 것을 수행할 것인지에 대한 정책.

2) Work fetch : 어떤 상황에서 프로젝트에게 태스크를 더 요청할 것인지, 어느 프로젝트에게 태스크를 요청할 것인지, 얼마나 많은 태스크를 요청할 것인지에 대한 정책.

3) Work Send : 프로젝트가 태스크 요청을 볼런티어로부터 받았을 때 어떤 태스크를 보내야 하는지에 대한 정책.

4) Job Completion Estimation : 태스크의 남은 CPU 연산시간을 어떻게 계산할지에 대한 정책.

해당 연구는 4개의 정책들을 각각 다르게 설정하여 볼런티어 컴퓨팅에 주는 영향을 분석하였다.

3. 제안하는 CPU/GPU 컴퓨팅 자원 최적화 기법

3.1 System model

제안하는 자원 최적화 기법 기법의 시스템 구조는 유저 어플리케이션의 성능간섭을 최소화 하면서 볼런티어 컴퓨팅 작업의 성능을 최대로 하는 것을 목표로 한다. Fig. 2는 제안하는 자원 최적화 기법이 수행되는 전체적인 시스템 구조를 나타내고 있다.

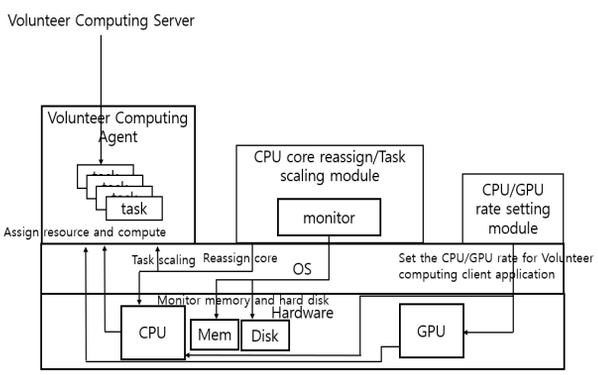

Fig. 2. Proposed System Model

기존의 볼런티어 컴퓨팅 모델에서 CPU core reassign/Task scaling module과 CPU/GPU rate setting module을 추가하였다. CPU core reassign/Task scaling module은 특정 CPU core에 작업이 집중되어 유저 어플리케이션의 성능에 간섭을 주지 않도록 사용되는 CPU core를 바꾸는 작업을 한다. 또한 볼런티어 컴퓨팅 에이전트가 차지하는 메모리와 디스크의 크기가 과도하게 증가하면 받아오는 태스크의 개수를 조절하는 태스크 스케일링을 수행한다. CPU/GPU rate setting module은 프로젝트 특화된 최적의 CPU, GPU 사용률을 찾기 위한 모듈이다.

3.2 CPU core reassign/Task scaling module

해당 모듈은 볼런티어 컴퓨팅의 효율보다는 성능 간섭과 부하를 방지하기 위한 모듈이다. 이를 위해 CPU core reassign/Task scaling module은 Algorithm 1과 같이 작동한다. Table 1은 알고리즘에서 사용하는 변수들을 나타낸 것이다.

Table 1. Used Variable

Symbol	Representation
$idle^{memory}$	idle meomory
$idle^{disk}$	idle disk
$idle_n^{CPU}$	idle CPU resource of core n
$threshold^{memory}$	threshold of memory
$threshold^{disk}$	threshold of disk
$threshold^{CPUtime}$	threshold of CPU time
$threshold^{CPUusage}$	threshold of CPU resource
N^{core}	number if CPU core
CPU_n^{usage}	usage of CPU core n
$count$	number of CPU core reassignment

Algorithm 1.
CPU core reassignment / Task scaling

```
1:  int resource_opt()
2:  /* CPU core reassignment mechanism*/
3:  for i=1 to N^core do
4:   if(CPU_i^usage > threshold^CPUusage) then
5:      for j=1 to N^core do
6:       if(idle_j^CPU > threshold^CPUusage ∧ i ≠ j) then
7:            release(core_i)
8:            alloc.core(j)
9:            count++
10:           break
11:       end if
12:      end for
13:   end if
14:  end for
15:
16: /* Task Scaling mechanism */
17: while (idle_t^memory < threshold^memory
18:    || idle_t^disk < threshold^disk )
19:    pause.current.task
20: end while
21: return count
```

Algorithm 1에서 3~14는 모든 CPU코어들을 검사하면서 특정시간 이상으로 임계값 이상의 CPU 사용률이 유지되었을 때 해당 CPU 코어에 있는 태스크들을 다른 CPU 코어로 재 할당 하여 작업을 수행하게 한다. 이렇게 CPU 코어를 재 할당했을 때 성능간섭이 발생한다. 즉 재 할당 작업이 최대한 발생하지 않는 것이 이상적이다. 17~20은 태스크 스케일링 기법이다. 프로젝트에서 위탁한 태스크들이 사용

하는 메모리와 디스크의 사용량이 사용자가 지정한 임계값보다 클 경우 태스크를 일시정지 시킨다. 이러한 작업은 태스크들이 사용하는 자원의 크기가 임계값보다 낮아질 때까지 반복한다. 21에서는 CPU 코어의 재 할당 횟수를 나타내는 변수 $count$를 반환한다.

3.3 CPU/GPU rate setting module

제시하는 모듈은 프로젝트에서 위탁한 태스크들을 효율적으로 수행하기 위한 최적의 설정 값을 찾기 위한 모듈이다. CPU 사용률과 GPU사용률을 다르게 하여 SETI@home과 Einstein@home에 특화된 설정 값을 찾기 위해 Algorithm 2를 제시한다. Table 2는 Algorithm 2에서 사용하는 변수들이다.

Table 2. Used Variables

Symbol	Representation
$CPUrate^n$	i th CPU usage
$GPUrate^n$	i th GPU usage
$optCPUrate$	optimal CPU usage
$optGPUrate$	optimal GPU usage
N_{rate}^{CPU}	Num of CPU usage
N_{rate}^{GPU}	Num of GPU usage
$flag$	if it is 1 the algorithm ends
$task.exe_{time}$	execution time of task
$threshold_{reassign}$	threshold of reassignment number of times

Algorithm 2.
CPU/GPU setting module

```
1:  /* CPU/GPU rate setting mechanism*/
2:  flag = 0
3:  for i=1 to N_rate^CPU do
4:      for j=1 to N_rate^GPU do
5:          set.cpu.gpu.rate(CPUrate^i, GPUrate^j)
6:          pause(task.exe_time)
7:      if(resource_opt() > threshold_reassign)
8:          flag = 1
9:          optCPUrate = CPUrate^i
10:         optGPUrate = GPUrate^i
11:         break
12:     end if
13:     end for
14:     if(flag)
15:         break
16:     end if
17: end for
```

$CPUrate^n$와 $GPUrate^n$에 포함된 CPU와 GPU의 사용률은 오름차순으로 정렬되어 있다. 3~4에서는 CPU의 사용률이 $CPUrate^i$일 때 GPU의 사용률을 증가시킨다.

5에서 CPU와 GPU의 사용률을 설정하여 태스크를 수행한다. 6에서는 설정한 사용률로 태스크를 수행하기 위해 지정한 시간만큼 태스크를 수행한다. 7~12에서 Algorithm 1에서 반환한 CPU 코어의 재 할당 횟수를 지정한 임계 값와 비교하여 반환한 값이 더 클 경우 더 이상 GPU의 사용률을 증가시키지 않고 해당 설정 값을 최적의 설정 값으로 정한다. Algorithm 1에서 반환한 값이 임계값보다 크다는 것은 더 이상 GPU의 사용률을 증가시키면 유저 어플리케이션에 대한 성능 간섭이 심하다는 것이다. 이와 같이 최적의 CPU GPU사용률을 찾으면 알고리즘을 종료한다.

4. 성능 분석

4장에서는 제안하는 기법의 성능을 확인하기 위한 실험을 설명한다. 먼저 실험환경과 실험 방식에 대한 설명을 진행하며, 다음으로 실험 결과에 대한 분석을 설명한다.

4.1 실험환경

실험은 Table 3과 같은 환경에서 진행된다. 유저 어플리케이션으로는 sysbench[6]를 사용하여 성능간섭을 분석하였다. SETI@home과 Einstein@home 가각 따로 실험하였으며 두 프로젝트에서 진행한 실험의 방식은 동일하다. 볼런티어에서 수행되는 태스크는 10개로 하였고 사용하는 CPU 코어의 개수는 3개로 하였다. 다음은 실험 방식과 측정한 값이다.

Table 3. System Environment

CPU	intel core i7 7700
GPU	NVIDIA GeForce 1050 Ti
Memory	16GB
OS	ubuntu 14.04 LTS
Project	SETI@home, Einstein@home

다음은 실험에서 측정하는 데이터들이다.

1) 볼런티어 컴퓨팅 태스크를 수행하지 않을 때 sysbench를 Prime number 10000으로 두어 수행하고 실행 시간을 측정한다.

2) Algorithm 2와 같이 수행하는 중 $pause(task.exe_{time})$ 동안에 유저 어플리케이션을 실행하여 실행시간은 측정한다.

3) Algorithm 2에서 CPU와 GPU의 사용률마다 Algorithm 1의 $count$ 값을 측정한다.

4) Algorithm 2에서 측정했던 CPU와 GPU의 사용률만으로 태스크를 수행하여 각 사용률에 따른 credit per second 값을 측정한다.

5) Algorithm 2를 통해 도출한 최적의 설정 값으로 10개의 태스크를 수행하고 기존의 BOINC설정으로 10개의 태스크를 수행 후 credit per second를 측정한다.

6) 최적의 설정 값으로 태스크를 수행하는 중 유저어플리케이션을 측정하여 실행시간을 측정한다.

SETI@home과 Einstein@home 각각 1)~6) 항목의 값들을 측정한다. Table 4는 Algorithm 1, Algorithm 2와 같이 수행하기 위해 필요한 상수들의 값들이다.

4.2 실험결과 및 분석

이번 섹션에서는 제안하는 기법을 적용 했을 때 태스크를 연산하는 효율의 변화와 성능간섭의 정도를 분석하고 평가한다. SETI@home과 Einstein@home에서 위탁한 태스크들을 각각의 결과를 분석한다.

Table 4. Constant Value

$threshold^{memory}$	20%
$threshold^{disk}$	10%
$threshold^{CPUtime}$	10sec
$CPUrate$	{10%, 20% ,30%, 40%, 50%, 60%, 70%, 80%, 90%}
$GPUrate$	{10%, 20% ,30%, 40%, 50%, 60%, 70%, 80%, 90%}
$threshold_{reassign}$	20

1) SETI@home

Fig. 3은 SETI@home의 태스크에 제안하는 기법을 적용하였을 때 CPU와 GPU의 사용률 당 시간대비 크래딧의 양을 측정한 것이다.

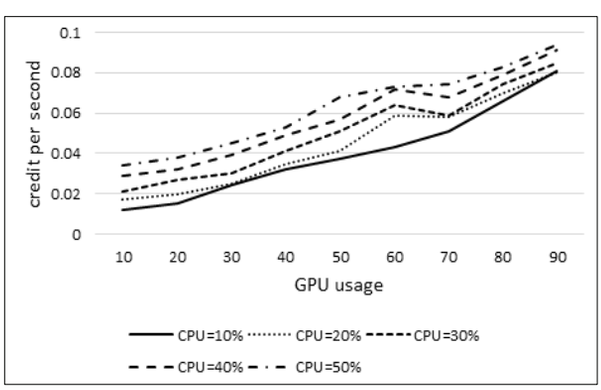

Fig. 3. Credit per Second When Run the Task at Each Usage of CPU/GPU in SETI@home

CPU의 사용률이 일정할 때 GPU의 사용률이 높아지면 시간대비 크래딧 또한 증가하는 것을 볼 수 있다. CPU보다는 GPU의 사용률에 따라 시간대비 크래딧이 더 크게 증가한다. 본 실험에서는 Algorithm 2에서의 종료 조건에 따라 count가 일정 임계치를 넘으면 측정을 중단하고 그 때 CPU/GPU의 사용률을 최적의 설정 값으로 결정하였다. Fig. 3에서 CPU의 사용률을 50%까지만 측정한 이유도 동일하다. Fig. 4는 각 CPU와 GPU의 사용률 당 유저어플리케이션의 실행시간을 나타낸 것이다. 유저 어플리케이션의 실행시간은 CPU의 사용률에 따라 증가하는 것을 볼 수 있다.

Fig. 5에서는 Algorithm 1의 count값의 변화를 나타낸 것이다. count는 성능 간섭의 정도를 나타내며 count를 기준으로 퇴적의 설정 값을 도출 했다. SETI@home에서는 CPU와 GPU의 사용률이 각각 50% 60% 일 때 최적인 것으로 나타났다. Fig. 6에서는 도출한 최적의 설정 값으로 SETI@home의 태스크를 수행하였을 때와 기본 BOINC로 태스크를 수행 하였을 때 시간대비 크래딧을 나타낸다. Fig. 6에서 볼 수 있듯이 태스크마다 성능의 차이가 있는 것은 태스크 마다 연산이 일정하지 않기 때문이다. Fig. 7은 태스크를 수행하지 않을 때와 제시하는 기법으로 태스크를 수행할 때 유저 어플리케이션의 수행시간의 차이를 나타낸 것이다.

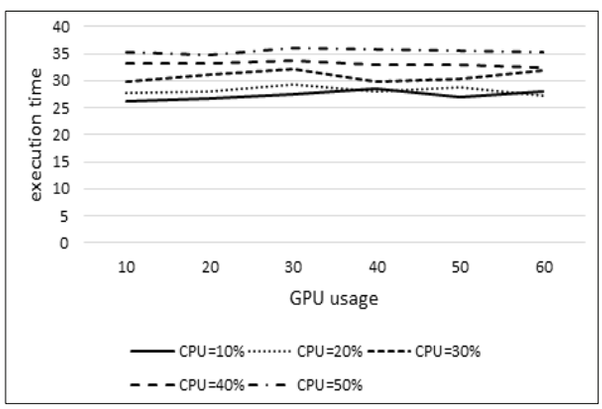

Fig. 4. User's Application Execution Time When Run the Task at Each Usage of CPU/GPU in SETI@home

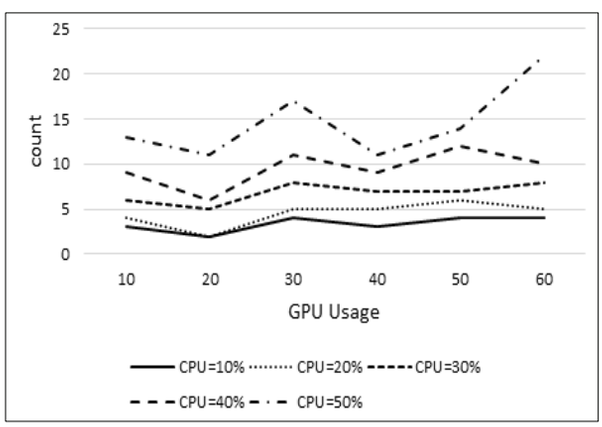

Fig. 5. Count Value of Each CPU/GPU Usage in SETI@home

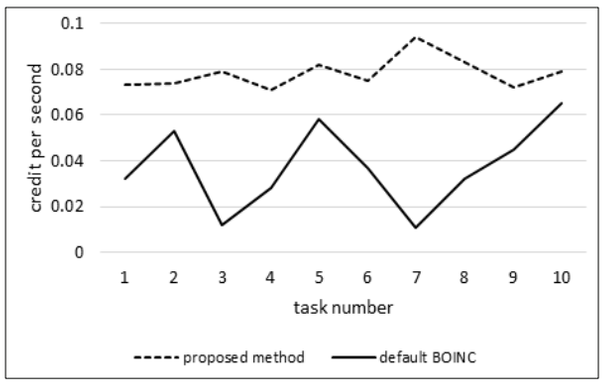

Fig. 6. Credit per Second of optimal CPU/GPU Usages in SETI@home

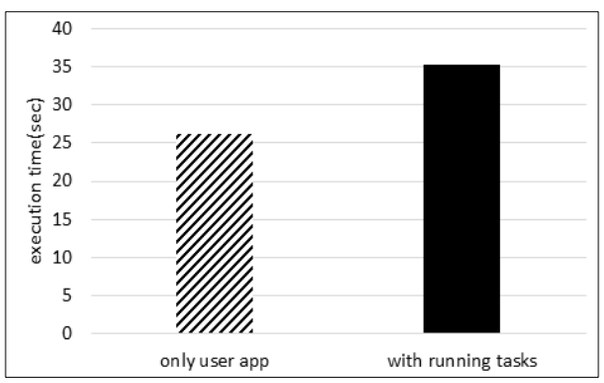

Fig. 7. User's Application Execution Time of Default Setting and with Running Tasks in Optimal Option in SETI@home

제시하는 기법으로 태스크를 수행했을 때 태스크를 수해하는 연산은 크게 증가하였지만 그에 따라 성능간섭도 증가함을 볼 수 있다. 하지만 성능간섭의 증가 정도가 태스크 연산 효율의 증가량 보다는 적다는 데에 의미가 있다.

2) Einstein@home

1)과 같은 실험을 Einstein@home에서도 진행하였다. Fig. 8은 Einstein@home에서 위탁한 태스크에 제안하는 기법을

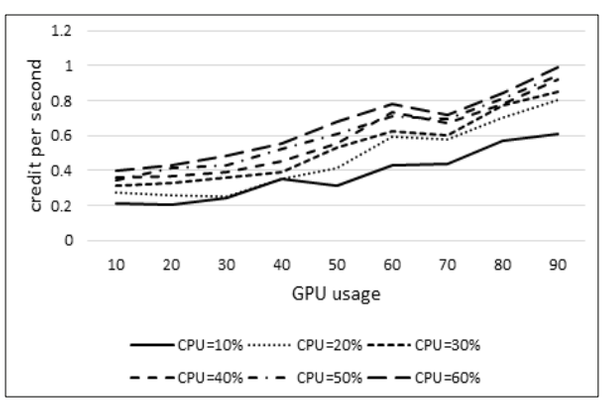

Fig. 8. Credit per Second When Run the Task at Each Usage of CPU/GPU in Einstein@home

적용하였을 때 CPU와 GPU의 사용률 당 시간대비 크래딧의 양을 측정한 것이다. SETI@home보다는 증가하는 정도가 낮다는 것을 관찰할 수 있다. 이는 Einstein@home에서 위탁한 태스크들의 GPU 연산 의존도가 비교적 낮기 때문이다. Fig. 9에서는 각 CPU와 GPU의 사용률 당 유저 어플리케이션의 실행시간을 나타낸 것이다.

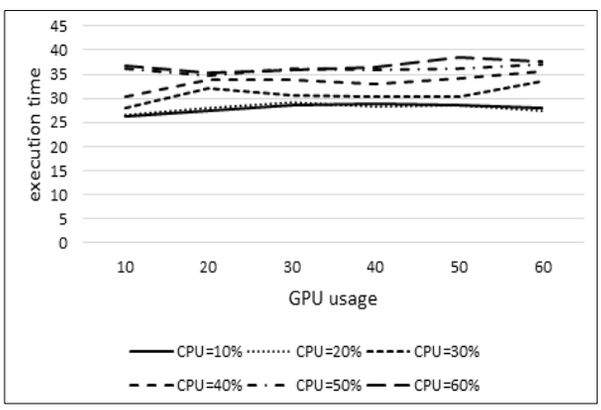

Fig. 9. User's Application Execution Time When Run the Task at Each Usage of CPU/GPU in Einstein@home

CPU의 사용률이 50%와 60%일 때 유저 어플리케이션의 실행시간이 크게 차이가 나지 않는 것을 볼 수 있다. 하지만 이는 성능간섭의 차이가 없다는 것을 나타내지는 않는다. Fig. 10에서 확인할 수 있듯이 CPU의 사용률이 50%일 때와 60%일 때 count의 값이 크게 차이나기 때문에 성능간섭은 증가한 것이다. Algorithm 2에 따라 Einstein@home에서는 CPU와 GPU의 사용률이 각각 60% 60%일 때 최적인 것으로 나타났다. 이는 Einstein@home에서 위탁한 태스크들이 SETI@home에서 위탁한 태스크들 보다는 CPU의 의존도가 높다는 것을 보여준다.

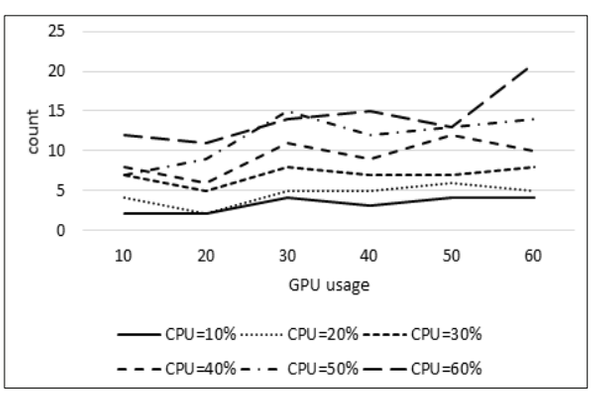

Fig. 10. Count Value of Each CPU/GPU Usage in Einstein@home

Fig. 11은 도출한 최적의 설정 값으로 Einstein@home의 태스크를 수행하였을 때와 기본 BOINC로 태스크를 수행하

였을 때 시간대비 크래딧을 나타낸다. 태스크 간 크래딧의 차이가 크게 나지 않는 것은 Einstein@home의 태스크가 더 규칙적이고 일관성 있기 때문이다. Fig. 12는 태스크를 수행하지 않을 때와 제시하는 기법으로 Einstein@home에서 위탁한 태스크를 수행할 때 유저 어플리케이션의 수행시간의 차이를 나타낸 것이다.

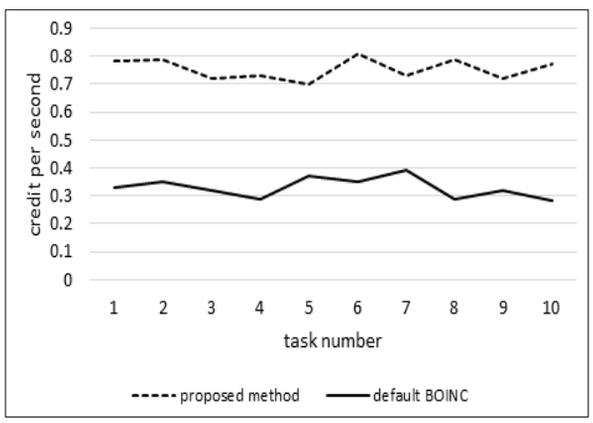

Fig. 11. Credit per Second of Optimal CPU/GPU Usages in Einstein@home

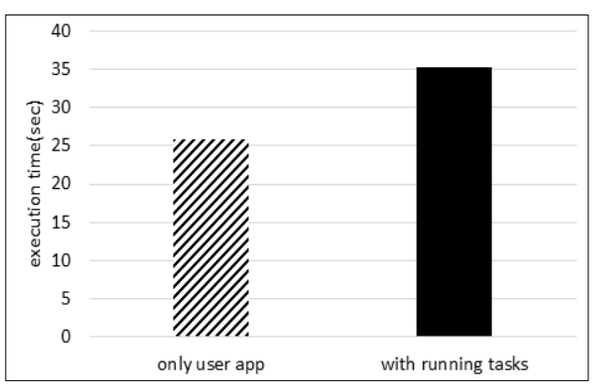

Fig. 12. User's Application Execution Time of Default Setting and with Running Tasks in Optimal Option in Einstein@home

5. 결 론

본 논문에서는 볼런티어 컴퓨팅의 효율을 증대시키기 위해 프로젝트 특화된 최적의 설정 값을 도출한다. CPU와 GPU의 사용률을 증가시켰을 때 발생하는 성능 간섭을 최소화하기 위해 태스크 스케일링을 하고 특정 CPU 코어에 작업이 몰리지 않도록 CPU 코어를 동적으로 재 할당 해주었다. 해당 연구는 볼런티어의 컴퓨팅 효율을 증대시켜 볼런티어 컴퓨팅 환경의 전체 컴퓨팅 자원을 증가시킨다. 실험결과 SETI@home에서는 109%의 성능향상을 보였고 성능간섭은 35%증가하였다. Einstein@home에서는 129%의 성능향상을 보였고 성능간섭은 36% 증가하였다. 향후 연구로는 Task 마다의 특징을 분석하여 그에 특화된 연산 설정을 도출하여 연산을 진행하는 연구이다.

References

[1] L. F. G. Sarmenta, "Volunteer Computing," Ph.D. dissertation, Massachusetts Institute of Technology, 2001.

[2] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, D. Werthimer, "SETI@ home: an experiment in public-resource computing," *Communications of the ACM* Vol.45, No.11, pp.56-61, 2002.

[3] Abbott, B. P. et al., "Einstein@ Home search for periodic gravitational waves in early S5 LIGO data," *Physical Review. D.*, Vol.80, Issue 4, pp.042003, 2009.

[4] D. P. Anderson, "Local scheduling for volunteer computing," *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, IEEE, 2007.

[5] D. Kondo, D. P. Anderson, and J. M. Vii, "Performance evaluation of scheduling policies for volunteer computing," *e-Science and Grid Computing, IEEE International Conference on.*, IEEE, 2007.

[6] A. Kopytov, "SysBench: a system performance benchmark," [Internet], http://sysbench.sourceforge.net (2004).

박 봉 우

http://orcid.org/0000-0002-4562-4656
e-mail : bongwoo@korea.ac.kr
2016년 고려대학교 컴퓨터학과(공학사)
2016년~현 재 고려대학교 컴퓨터학과 석사과정
관심분야 : Cloud Computing, Volunteer Computing, Resource Virtualization

송 충 건

http://orcid.org/0000-0001-7059-1123
e-mail : security0730@korea.ac.kr
2014년 백석대학교 정보통신학부(공학사)
2015~현 재 고려대학과 컴퓨터학과 박사과정
관심분야 : Distributed System, Cloud Computing, Virtualization, Volunteer Computing

유 헌 창

http://orcid.org/0000-0003-2216-595X

e-mail : yuhc@korea.ac.kr

1989년 고려대학교 전산과학과(이학사)

1991년 고려대학교 컴퓨터학과(이학석사)

1994년 고려대학교 컴퓨터학과(이학박사)

1995년～1998년 서경대학교 컴퓨터공학과 조교수

1998년～현 재 고려대학교 컴퓨터학과 교수

관심분야 : Distributed System, Cloud Computing