Volunteer Computing 환경에서 클라이언트의 고유작업에 대한 성능간섭 분석†

이재학, 송충건, 유헌창
고려대학교 컴퓨터학과
e-mail:{smreodmlvl, security0730, yuhc}@korea.ac.kr

Performance Interference Analysis for Performing Client's Specific Tasks in Volunteer Computing Environment

JaeHak Lee, ChungGeon Song, HeonChang Yu
Dept of Computer Science and Engineering, Korea University

요 약

Volunteer Computing 환경에서 프로젝트 서버가 클라이언트들의 유휴 컴퓨팅 자원을 효율적으로 활용하기 위한 연구는 활발하게 진행되었으나, 프로젝트 서버로부터 받아온 외부작업과 클라이언트의 고유작업 사이에서 발생하는 성능간섭을 최소화시키기 위한 연구는 미비하다. 본 논문은 대표적인 Volunteer Computing 플랫폼인 BOINC 환경에서 외부작업이 클라이언트의 고유작업에 미치는 성능간섭을 확인하기 위한 다양한 실험을 수행하였으며, 그 원인에 대한 분석을 제시하였다. 실험 결과 고유작업의 쓰레드 수가 증가하거나 전체 연산에서 I/O 작업의 비중이 높을수록 성능간섭이 늘어나는 사실을 확인하였다. 이러한 연구의 결과는 사용자가 Volunteer Computing 어플리케이션을 운영할 시 고유작업의 특징을 고려하여 성능간섭을 최소화 하는 옵션을 설정하는 목적에 활용될 것으로 기대된다.

1. 서론

Volunteer Computing(VC)은 네트워크에 연결된 클라이언트의 유휴 컴퓨팅 자원들을 이용해 서버의 처리량을 늘려 자원의 한계를 극복할 수 있게 해주는 분산 컴퓨팅 패러다임이다. 이러한 VC는 연산 작업이 많이 필요한 과학적 연구 프로젝트에 많이 사용되어진다. 그 이유는 연결된 클라이언트의 유휴 컴퓨팅 자원을 이용해 결과를 반환받음으로서 프로젝트의 비용을 줄이고 자원의 한계를 극복할 수 있기 때문이다. 대표적인 VC 플랫폼인 BOINC를 이용하고 있는 과학적 프로젝트는 현재 BOINC 홈페이지에 공개적으로 약 34개가 등록되어 있으며, 현재 174,836명 참여자와 587,042대의 컴퓨터가 BOINC 프로젝트 서버에 연결되어 자신의 유휴 컴퓨팅 자원을 제공하고 있다[1]. 기부하는 사용자들이 많아질수록, 많은 유휴 컴퓨팅 자원을 사용할 수 있는 VC 플랫폼에서 서버의 작업 할당 및 관리 개선에 관련된 많은 연구들이 진행되어 서버의 효율적인 요청 처리와 최적화가 이루어졌다.

하지만 유휴 컴퓨팅 자원을 제공하는 클라이언트가 프로젝트 서버로 부터 받아온 외부작업 수행에 따른 클라이언트의 성능간섭은 아직까지 문제가 되고 있다. 클라이언트에서 외부작업 처리로 인한 CPU사용률 및 네트워크 트래픽 제한 등은 아직까지 성능간섭을 일으키고 있다. 향후, 사물 인터넷제품과 같은 경량화 된 하드웨어의 유휴 컴퓨팅 자원 사용에서도, 클라이언트에서 일어나는 외부작업으로 인한 성능간섭은 클라이언트의 고유작업 처리에 지연을 유발시킬 가능성이 있다[2].

본 논문에서는 VC 플랫폼인 BOINC를 이용하여 프로젝트 서버로부터 받아온 외부작업 수행에 대한 클라이언트의 유휴 컴퓨팅 자원제공 정도에 따른 클라이언트의 성능간섭을 분석 하였다. 그 결과, 외부작업에 클라이언트의 자원을 할당할수록 클라이언트에서 나타나는 고유작업에 대한 성능간섭은 증가하는 것을 확인하였다.

논문의 구성은 다음과 같다. 2장은 VC 플랫폼에서 클라이언트의 외부작업으로 인한 성능간섭 개선에 관련된 연구를 살펴본다. 3장은 VC 플랫폼인 BOINC의 동작방식을 살펴보고, 4장은 클라이언트가 받아온 외부작업에 의한 성능간섭 실험 및 분석을 진행한다. 마지막으로 5장에서는 결론과 향후 연구내용을 서술한다.

2. 관련 연구

[3]은 "Available volunteering computing"플랫폼을 제시하여 클라이언트의 고유작업 특성을 식별하고, 외부작업을

† 이 논문은 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)

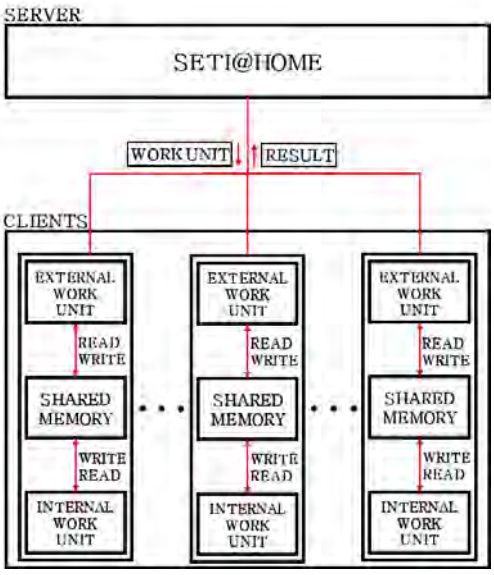
가상머신에서 동적으로 스케줄링을 하여 고유작업에 대한 성능간섭을 줄이는 방법을 제시했다.

[4]는 Peer-To-Peer 접근을 사용하여 클라이언트에서 임계치를 넘는 자원이 필요한 고유작업 요청이 들어오게 되면 네트워크에 연결되어진 다른 클라이언트에게 그들의 유휴 컴퓨팅 자원 사용을 요청하여 자신의 고유작업을 처리하는 프레임 워크를 소개했다.

마지막으로, [5]는 로컬 CPU 스케줄링을 최적화한 “The adaptive priority tuning middleware system”을 제시하였다. 클라이언트에 설치된 BOINC의 자원 사용 설정과 고유 운영체제 정보를 수집한다. CPU 사용량이 낮게 설정된(사용률 90%이하) 외부작업에서 수집한 정보를 기준으로 우선순위를 두어 제시한 BOINC 미들웨어 시스템의 알고리즘에 반영한다. 순정의 BOINC 시스템에서 클라이언트의 고유작업 처리 시간보다 낮은 처리 시간으로 외부작업을 처리하는데 성공했다.

하지만 VC 플랫폼의 클라이언트에서 외부작업으로 인한 성능간섭 개선 연구는 VC 플랫폼의 서버 연구만큼 이루어지지 않고 있으며, 기존 연구에도 불구하고 클라이언트의 고유작업에 대한 성능간섭 문제는 아직까지 해결되지 않고 있다.

3. BOINC 시스템



(그림 1) BOINC 구조

BOINC는 기본적으로 프로젝트 서버에는 BOINC-서버 컴포넌트, 클라이언트에는 BOINC-클라이언트 컴포넌트가 설치된 서버-클라이언트 구조의 미들웨어이다. (그림 1)에서와 같이 클라이언트가 수행할 외부작업을 프로젝트 서버로 요청하면, 서버는 클라이언트에게 작업을 전송한다. 클라이언트는 작업을 다운로드 받은 후, 유휴 컴퓨팅 자원을 사용해 외부작업을 처리 후 결과를 프로젝트 서버에게 반환한다. BOINC-클라이언트 컴포넌트의 구조를 살펴보자면, 크게 세 가지 구조로 분류되어 클라이언트에 설치되어 동작한다. 각 동작은 아래를 참고한다[6].

● BOINC CORE CLIENT - 스케줄러, 파일 업로드와 다운로드, 외부작업 수행 및 배치를 담당한다. 외부작업 처리에 대한 기본 스케줄링은 선점적인 라운드 로빈 방식이다.

● BOINC MANAGER - 프로젝트 참여자에게 그래픽 인터페이스를 제공하고, 외부작업 계산 정도를 조절한다. 외부작업 진행정도와 완료까지의 시간을 측정하고, BOINC CORE CLIENT와 TCP로 RPC(Remote procedure call)를 이용해 통신한다.

● BOINC SCREENSAVER - 클라이언트의 동작이 없을 시, 코어 클라이언트와 통신하여 풀-스크린 그래픽으로 동작중인 외부작업의 상태를 클라이언트의 모니터 화면에 노출한다.

클라이언트는 프로젝트 서버로부터 받아온 외부작업에 대해 CPU스케줄링을 하는데, 외부작업은 클라이언트의 고유작업과 함께 운영체제의 상단에서 제일 낮은 우선순위로 스케줄링 된다[2]. 외부작업과 고유작업은 공유된 메모리를 사용하는데, 이 부분에서 고유작업의 페이지 폴트율이 증가한다. 우선순위가 더 높은 고유작업이 실행될 때, 공유 메모리에 적재되어 외부작업 처리에 사용되고 있던 페이지들에 의한 페이지 폴트와, 클라이언트의 고유작업이 가상 메모리로부터 가져오는 페이지 폴트에 의한 오버헤드가 일어날 가능성이 매우 높다.

4. 성능 평가

프로젝트 서버로 부터 받아오는 외부작업으로 인한 클라이언트의 성능간섭 분석을 수행한다. 클라이언트 환경은 외계에서 오는 신호를 분석하는 과학적 프로젝트인 SETI@home[7] 서버로부터 받아온 외부작업이 수행 되는 클라이언트 환경과 수행 하지 않는 클라이언트 환경으로 구분한다. 실험 환경 구성은 <표 1>과 같다. 클라이언트의 외부작업 정도는 cc_config.xml 파일에서 소프트웨어 레벨의 쓰레드를 생성하는 npuc의 값을 각 1, 50, 100으로 수정하여 외부작업을 수행하는 쓰레드를 각 1개, 50개, 100개씩 수행한다.

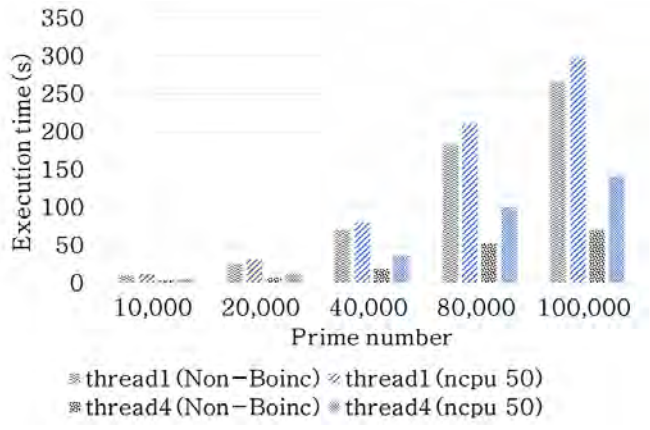
<표 1> 실험 환경 구성

CPU	i5-4590T CPU@2.00GHz
RAM	8GB
MEMORY	500G
OS	Ubuntu 14.04LTS
BOINC VERSION	7.2.47
BOINC PROJECT	SETI@home

성능 분석 방법은 시스템 벤치마크 툴인 Sysbench를 사용하여 만들어낸 고유작업의 처리 시간을 기준으로 외부작업으로부터의 성능간섭 정도를 3가지 방법으로 분류하였다. 첫 번째로, 프로젝트 서버로부터 클라이언트가 받아온 외부작업 수행에 따른 클라이언트의 고유작업 처리 시간과 두 번째로, 외부작업이 수행되는 환경에서 고유작업의 쓰레드 개수에 따른 처리 시간을 비교 및 분석 한다. 마지막으로, 외부작업 수행에 따른 클라이언트의 대용량 데이터 처리 성능을 비교 및 분석 한다.

4.1 외부작업 수행에 따른 성능 분석

클라이언트는 ncpu가 50으로 설정된 BOINC-클라이언트로부터 할당된 외부작업의 수행으로 고유작업 처리에 쓰레드가 1개일 때와 4개일 때의 CPU 처리 시간을 측정한다. (그림 2)는 소수를 찾는 클라이언트의 고유작업에 대해 외부작업이 수행되는 클라이언트와 수행되지 않은 클라이언트에서의 성능 분석 결과 그래프이다. 외부작업이 수행될 때가 더 긴 처리 시간을 가지는 것을 (그림 2)를 통해 확인 했다. 소수의 숫자가 커질수록, 외부작업이 수행되는 것과 수행되지 않는 작업간의 차이가 선형적으로 증가하고 있다. 클라이언트의 고유작업 처리에 대해 쓰레드의 개수가 증가할수록 외부작업에 의한 클라이언트의 고유작업 처리 시간이 외부작업이 없는 환경에 비해 증가했다.

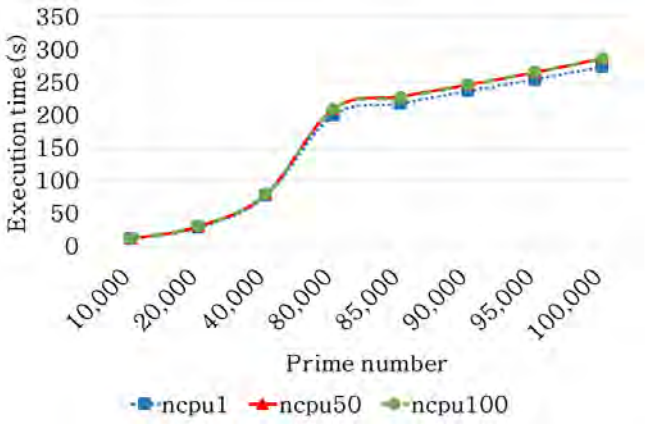


(그림 2) 외부작업 수행에 따른 고유작업 처리 시간

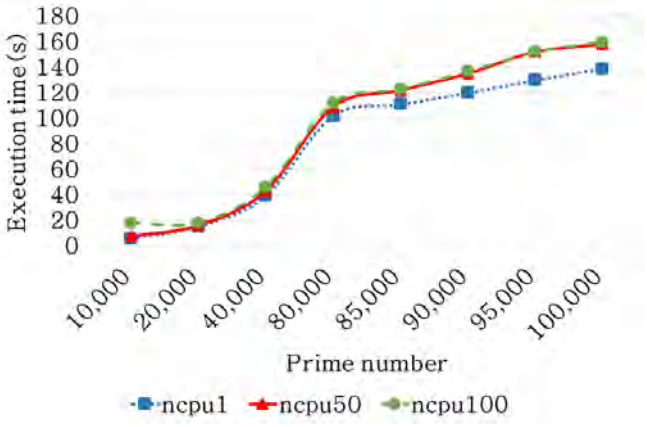
4.2 고유작업 쓰레드 개수에 따른 성능 분석

프로젝트 서버로 부터 받아온 외부작업에 할당되는 유휴 컴퓨팅 자원에 대해 소프트웨어 레벨 쓰레드 ncpu에 대해 순서대로 1, 50, 100을 대입하고, 이에 따른 클라이언트의 고유작업을 처리하는 쓰레드의 개수에 따라 (a), (b), (c)로 각 1개, 2개, 4개씩 할당한다. (그림 3)은 소수와 쓰레드의 개수에 따른 클라이언트의 고유작업 처리 시간을 외부작업 정도에 따라 나타낸 그래프이다.

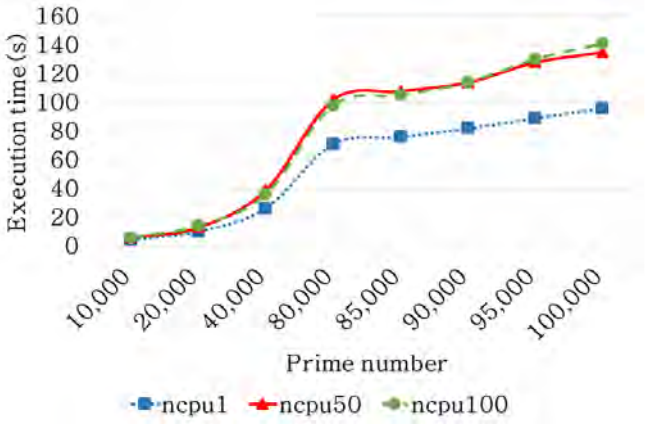
(a), (c)를 비교해보면, ncpu1과 ncpu100의 차이가 (a)보다 (c)가 더 폭이 넓다. (a)에서는 각 ncpu에 따른 고유작업 처리 시간 차이가 크지 않지만, 클라이언트의 고유작업에 쓰레드를 2개 이상 할당하면 ncpu1과 ncpu100의 처리 시간 차이가 커지고 있다. (a)에서 ncpu50, ncpu100 처리 시간은 ncpu1에 비해 둘 다 평균적으로 약 3.67%씩 증가하였다. (b)와 (c)의 각 ncpu1 처리 시간을 기준으로



(a) 쓰레드가 1개 일 때 고유작업 처리 시간



(b) 쓰레드가 2개 일 때 고유작업 처리 시간



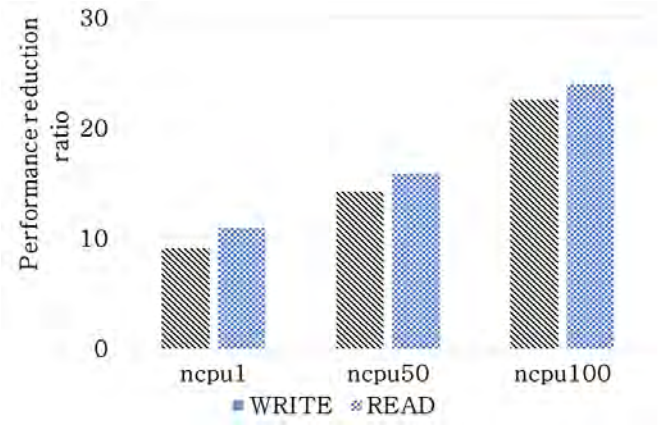
(c) 쓰레드가 4개 일 때 고유작업 처리 시간

(그림 3) 쓰레드 개수에 따른 외부작업 처리 시간

처리시간이 (b)에서는 ncpu50일 때 10.22%, ncpu100일 때 13.12% 증가 했고. (c)에서는 각 19.25%. 21.33%로 (b)에서 보다 처리 시간이 더 증가했다. 클라이언트의 고유작업을 처리하는 쓰레드가 많아질수록, 외부작업이 요구하는 유휴 컴퓨팅 자원이 많을수록 클라이언트에서 나타나는 성능간섭이 증가했다.

4.3 외부작업 수행에 따른 OLTP 성능 분석

이번 장에서는 대용량 데이터 입출력을 수행하는 OLTP 작업의 성능을 분석하는 실험을 수행하였다. OLTP 작업은 MySQL 5.5 환경에서 1분 동안의 입출력 작업을 통해 진행하였으며, SETI@home의 외부작업과 고유작업이 동시에 수행되는 환경과 고유작업만 독립적으로 수행되는 환경을 비교하였다. 클라이언트의 고유작업에 사용되는 쓰레드의 개수는 8개로 동일하게 설정하였다.



(그림 4) OLTP 성능

(그림 4)는 외부작업이 수행되지 않는 환경에서의 OLTP 성능 결과와 외부작업이 수행되는 환경에서 ncpu 값을 1, 50, 100으로 할당하여 측정한 OLTP작업의 성능 결과를 비교한 그래프이다. 외부작업이 수행되지 않는 환경의 READ 요청 개수와 WRITE 요청 개수의 합을 100%(총 561,690번)로 정한 기준에 대해, ncpu1은 8.11%, ncpu50은 14.22%, ncpu100은 23.59% 만큼 READ 요청 개수와 WRITE 요청 개수 합이 줄어들었다. 결과적으로 I/O-Intensive한 고유작업 수행 중 VC 작업을 증가시킬 경우 성능 저하 비율이 늘어났다. 이는 I/O 작업이 요구하는 메모리 영역을 VC 작업이 선점하면서 스와핑 현상이 발생하고 결과적으로 성능 저하로 이어졌다.

5. 결론

본 연구에서는 VC 환경의 외부작업으로 인해 발생하는 클라이언트 고유작업에 대한 성능간섭을 확인하는 실험을 수행하였으며, 그 원인에 대한 분석을 제시하였다. 실험의 결과로 더욱 많은 자원을 요구하는 외부작업일수록, 쓰레드가 많이 쓰이는 클라이언트의 고유작업 일수록 공유메모리를 사용하는 VC 플랫폼에서 성능간섭이 증가한다는 사실을 확인하였다. 클라이언트에서 자원을 많이 사용해야 되는 특정 고유작업을 식별하고 자원 사용 정도에 대한 임계치를 두어, 그 임계치가 넘어갔을 때 성능간섭으로 간주하여 외부작업에 제공되는 유휴 컴퓨팅 자원을 조절 할 수 있어야 된다. 또 한, 외부작업과 고유작업의 처리에 사용되는 공유메모리에 클라이언트의 고유작업을 기준으로 외부작업 페이지들의 적재 비율을 동적으로 조절하는 소프트웨어적 관리기법의 연구가 필요하다. 이에 따른 향후 연구에서는 클라이언트에 대한 고유작업의 연산 패턴을 고려한 메모리 블록 최적화 기법에 대하여 연구할 계획이다.

참고문헌

[1] https://boinc.berkeley.edu

[2] Jiangtian Li, Amey Deshpande, Jagan Srinivasan, Xiaosong Ma, “Energy and performance impact of aggressive volunteer computing with multi-core computers”. Modeling, Analysis & Simulation of Computer and Telecommunication Systems, 21-23, 2009.

[3] Nunziato Cassavia, Sergio Flesca, Michele Ianni, Elio Masciari, Giuseppe Papuzzo, Chiara Pulice, “A Peer to Peer Approach to Efficient High Performance Computing”. Parallel, Distributed and Network-based Processing (PDP), 2017 25th Euromicro International Conference on, 6-8, 2017.

[4] Adel B Mnaouer, Colin Ragoonath, “An Adaptive Priority Tuning System for Optimized Local CPU Scheduling using Boinc Clients”. Journal of Physics: Conference Series, 2010.

[5] David P. Anderson, Carl Christensen, Bruce Allen, “Designing a Runtime System for Volunteer Computing”. SC ‘06: Proceedings of the 2006 AMC/IEEE Conference on Supercomputing, 11-17, 2006.

[6] P. Anderson, “Volunteer computing: the ultimate cloud”. Space Sciences Laboratory University of California, Berkeley.

[7] http://setiathome.ssl.berkeley.edu/