Korea@Home 에이전트를 위한 사용자패턴기반의 로컬 스케줄링기법

Local Scheduling method based on the User Pattern for Korea@Home Agent

최지현, 김미경*, 최장원**
한국과학기술정보연구원

JiHyun Choi, Mikyoung Kim*, JangWon Choi**
Korea Institute of Science and Technology Information

요약

우리나라의 컴퓨터 보유율과 인터넷사용률은 세계1위를 자랑한다. 2002년 이래로 운영되어 오고 있는 코리아앳홈(이하 Korea@Home)프로젝트는 우리나라가 보유한 풍부한 컴퓨팅 자원들을 모아서 대규모 응용에 활용하고자 하는 것이다. 이미 많은 회원들을 확보하고 있으며, 현재 신약개발과 기후예측 등의 응용을 수행하고 있다. 코리아앳홈 회원들이 자신의 컴퓨터 자원을 제공하는데 있어 사용자들에게 불편을 주지 않기 위해 자원제공자들의 기여 패턴을 분석하여 사용자가 컴퓨터를 사용하지 않는 시간에 따라 에이전트t의 작업을 수행하도록 하는 스케줄링 방법을 제안한다

Abstract

This paper proposes a local scheduling method based on user pattern for Volunteer computing project, Korea@Home. It enables Korea@Home participants to run the agent without disturbance. It is devised to prevent user's application from delay while running the agent and decreases the frequency of switching resource between the user and the agent. We analyze the user's patterns of donating computing resource with Korea@Home which is a representative volunteer computing project in Korea. It has contributed the computing power to several applications including climate prediction and virtual screening. It promotes the volunteers to participate continuously without disturbance and increases the potential computing power with non-disturbance scheduling based on user usage pattern for Volunteer Computing

I. 서 론

각 가정의 PC의 보급률과 인터넷 가입률이 점점 높아지고 있고, PC성능과 인터넷 연결 또한 사용자들의 요구에 따라 질적 성장이 계속되고 있다. 높은 성능의 슈퍼컴퓨터를 필요로 하는 연구들이 있고, 이러한 연구 분야에서 슈퍼컴퓨터를 요구하게 되었지만, 슈퍼컴퓨터는 아주 비싼 도입비용과 관리 비용 때문에 누구에게나 가용한 것이 아니며, 필요할 때 마다 마음대로 늘 이용할 수 있는 것이 아니다. 반면 우리의 가정마다 보유하고 있는 개인 컴퓨터들의 성능이 우수해지면서, 슈퍼컴퓨터를 대신해 대규모의 작업을 분할하여 인터넷에 연결된 수많은 PC를 이용해서 이러한 작업을 수행할 수 있다. 이것을 가능하게 하는 것을 자발적 컴퓨팅이라고 한다.

자발적 컴퓨팅은 많은 계산자원을 필요로 하는 응용을 위해서 사용되지 않는 계산자원을 활용하는 것이다. 자발적 컴퓨팅은 이전에는 SETI@home과 Great Internet Mersenne Prime Search(GIMPS)와 같이 실행 불가능한 것으로 인식되거나, 특히 한정된 시간 내에 수행되어야 하는 것이 아닌, 오랜 시간 동안 수행 프로젝트 등에서 수행되는 것으로 인식되었으나, 지난 10년 동안 자발적 컴퓨팅은 계산 집약적인 프로젝트를 수행하기 위한 새로운 컴퓨팅 패러다임으로 발전해 오고 있다[1][2]. 전 세계적으로 많은 PC들을 보유하고 있기 때문에 자발적 컴퓨팅은 많은 계산력을 과학 분야에 제공할 수 있다. 자발적 컴퓨팅 분야는 아직도 많은 연구가 되어져야 한다. 자원제공자들이 제공하는 인터넷에 연결된 PC들의 계산 자원을 더욱 더 효과적인 방법으로 활용하는 방법에 대해서 더 많은 연구가 필요하다[4][5].

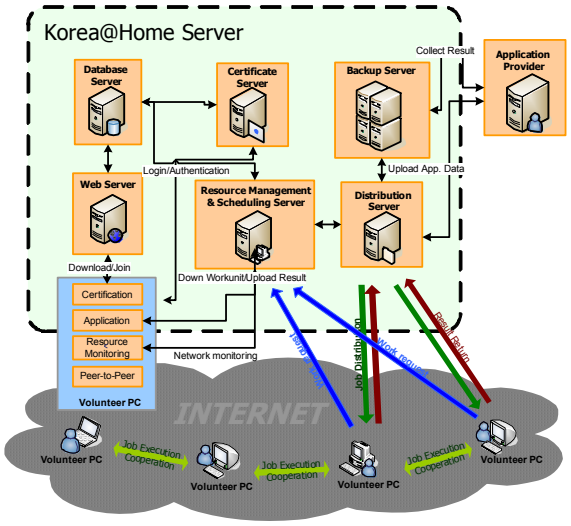
Korea@Home[3]프로젝트는 자발적 컴퓨팅의 한 사례로, 우리나라에서 2002년부터 시범적으로 개발되고 운영되어오고 있는 프로젝트로, 국내의 총 3만 8천대의 에이전트를 보유하고 있으나, 신약개발 및 한반도 기후예측, 고해상도 강수량 예측 등 여러 가지 응용들을 수행하고 있다. 자발적 컴퓨팅은 자원제공자들의 PC의 계산 자원을 제공받아서 컴퓨팅 파워를 모으는 것이기 때문에 사용자들에게 자원을 제공받기 위해 사람들에게 자발적인 참여의지를 불러일으킬 수 있는, 예를 들어

국익을 위한 연구를 수행하는 프로젝트와 같은 응용을 수행해야 하는 한계가 있다.

1. 목적

본 연구는 코리아앳홈 에이전트를 수행하는 사용자에게 불편함을 주지 않고, 에이전트를 수행할 수 있도록 하는 것이 궁극적인 목표이다. 자원제공자들이 에이전트를 수행하는데 있는 발생할 수 있는 불편사항들을 제거하여, 불편함 없이 자연스럽게 자원제공자들이 자신의 자원을 기꺼이 제공할 수 있도록 유도하여, 더 많은 제공자들을 코리아앳홈 프로젝트에 동참시키도록 하여, 최대한 많은 자원들을 참여시키고, 많은 계산자원을 확보하는 것 이다.

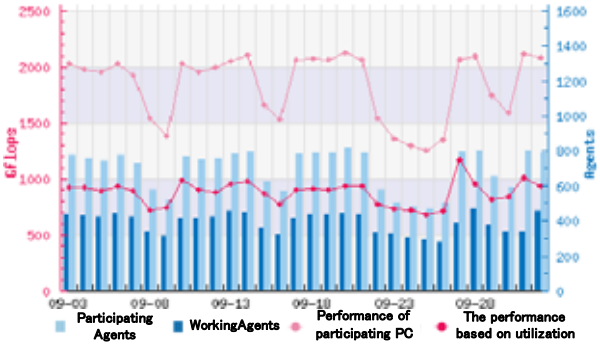
Ⅱ. 자발적컴퓨팅: Korea@Home



▶▶ 그림 1. 코리아앳홈 구조

그림1은 서버와 에이전트를 포함하는 플랫폼 구조를 묘사한다. 서버 파트는 주로 에이전트와 통신을 담당하는 메인서버와 대용량 응용의 데이터들을 분배하는 분배 서버, 그리고 그 외 인증서버, 데이터베이스서버, 웹서버 등으로 구성된다. 서버파트에서는 메인서버가 에이전트와 통신을 담당하고, 대용량 응용서버가 응용데이터를 분배하고 결과를 취합하는 역할을 한다.

그림2는 코리아앳홈의 일별 성능이다. 코리아앳홈 홈페이지에서는 해당일로부터 이전 30일동안의 성능그래프를 실시간으로 제공한다.



▶▶ 그림 2. 코리아앳홈 일별 성능

Ⅲ. 사용자 패턴 기반 로컬스케줄링기법

1. 목적

사용자 패턴을 분석하는 목적은 코리아앳홈 회원들이 에이전트를 수행하는데 있어 불편을 주지 않고 에이전트를 수행할 수 있게 하는데 있다. 대부분의 자발적 컴퓨팅 프로젝트들은 시스템이 유휴한 상태일때 혹은 컴퓨터가 사용 중이더라도, 사용자에게 방해되지 않도록 백그라운드로 에이전트 작업을 수행하도록 한다. 코리아앳홈은 38000대의 에이전트로 코리아앳홈을 안정적으로 운영하고 있으며, 에이전트의 설정사항에는 CPU활용개수, 프로세스 우선순위, 받아올 작업 개수, 동작 대기 시간 설정등을 설정할 수 있다. 이와 같이 사용자에게 여러 가지 설정사항을 허용한 것은 하여 자신의 PC 자원을 기여하는데 있어 불편함을 최소로 하고 편의에 맞게 에이전트를 수행할 수 있도록 하기 위함이다.

실제로 대부분의 에이전트 사용자들은 동작대기 없이 , 컴퓨터가 사용 중일 때 동시에 에이전트를 수행하는 것으로 나타났다. 그러나 컴퓨터가 사용 중일 때에 에이전트를 함께 수행하는 것은, 사용자에게 불편함을 초래할 가능성을 가진다. 사용자의 PC가 이미 부하가 높은 작업들로 시스템 자원이 여유롭지 않을 때는 에이전트를 수행하는 것은 시스템에 부담을 줄 수 있고, 사용자 응답시간이 늦어져 사용자들이 불편을 느끼는 경우가 생길 수 있다. 이러한 불편을 초래할 가능성을 최소화 하고 더 많은 에이전트의 참여를 유도하여 더 큰 컴퓨팅 파워를 모으는 것이 가능하다.

2. 사용자 패턴분석

2.1 패턴분석 방법

에이전트의 사용패턴을 파악하기 위한 두가지 방법이 있다. 첫째는 직접적으로 사용자의 컴퓨터 상태를 파악하는 방법으로, 에이전트에서 동작대기 시간을 탐지하는 유휴상태 감지 인터페이스를 이용해서 사용자로부터의 마우스나 키보드 입력

이 없는 상태를 파악하여 사용자가 컴퓨터를 사용하는 시간과 사용하지 않는 시간대를 구별할 수 있다. 두 번째로, 간적접인 방법으로, 에이전트가 수행한 작업들의 기록을 이용하는데, 이를 위해서 서버에 기록된 작업들의 로그를 에이전트별로 추출하여 에이전트가 수행한 작업들의 시간들로 에이전트가 주로 언제 작업을 수행하는가를 파악할 수 있다. 본 논문에서는 간접적인 방법을 이용하여 지난 1개월간의 로그데이터를 분석하여 에이전트들의 사용패턴을 분석하였다.

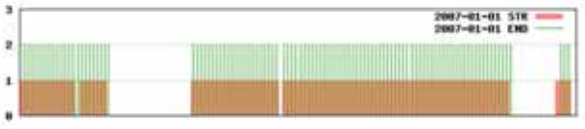
2.2 간접 패턴분석

본 논문에서는 사용자 패턴을 분석하기 위하여 서버와 에이전트간의 주고받는 메시지를 분석하여 에이전트의 작업시간 패턴을 추출한다.

```
04:09:24 Recv:<kath><request><cominfo><CPU num="1" cat="AMD
04:09:24 Send|<kath><response><flops_info><total_agent_coun
04:09:24 Recv:<kath><request><version></version></request><
04:09:24 Send|<kath><response><version>2.5.36</version></re
04:09:25 Recv:<kath><request><peerlist></peerlist></request
04:09:25 Send|<kath><response><peerlist><peer><id>imagon</i
04:09:26 Recv:<kath><request><applist test="0"></applist></
04:09:26 Send|<kath><response><applist><appitem appid="Virt
04:09:27 Recv:<kath><request><joblist appid="VirtualScreen"
04:09:28 Send|<kath><response><joblist><appid>VirtualScreen
04:09:30 Recv:<kath><request><cur_app>VirtualScreen</cur_ap
04:09:31 Send|<kath><response><cur_app>151</cur_app></respo
04:09:31 Recv:<kath><request><startwork cpu="0"><appid>Virt
04:09:31 Send|<kath><response><startwork result="ok" cpu="0
04:16:14 Recv:<kath><request><endwork><appid>VirtualScreen<
04:16:14 Send|<kath><response><unitinfo><runtime>18504</run
```

▶▶ 그림 3. 서버-에이전트간의 통신 프로토콜

그림3은 서버와 에이전트간의 주고받는 실제 메시지의 일부분이다. 이것은 서버가 에이전트로부터 받는 요청 메시지와 서버가 에이전트에게 보내는 응답 메시지로 구성된다. 에이전트의 작업시간을 파악하기 위해서는 에이전트가 보내는 *endwork* 메시지의 정보가 필요하다. 이 메시지 해당 작업에 대한 *starttime*과 *worktime*이 있으며, 본 메시지를 수한 시간의 정보를 가지고 있기 때문에 에이전트가 수행한 작업들의 수행시간을 알 수 있으며, 패턴 분석의 기본 단서가 되는 정보들이다. 한 에이전트의 하루의 메시지 로그에서 endwork메시지를 추출하여, 그림4의 패턴을 구하였다. 그림4는 0시부터 24시까지 에이전트가 작업들을 수행한 시간을 나타낸다. 붉은선은 작업이 시작한 시간을 나타내며, 초록선은 작업을 끝마친 *endwork*메시지를 수신한 시간이다.



▶▶ 그림 4. 간접분석방법을 이용한 에이전트의 일일 패턴

위와 같이 메시지로그로부터 일차적으로 사용자의 일일패턴을 구해보면, 여러 구간들 사이에 많은 틈이 생긴다. 이러한 구간의 단편들은 실제의 스케줄링에 필요한 사용자 패턴만을 활용하기 위해서 병합 알고리즘을 통해 패턴을 재가공한다.

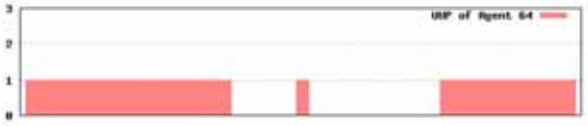
```
#st: starttime, et:endtime, AHS=Agent History Set
AHS_x  {(st_0,et_0), (st_1,et_1), ... (st_n,et_n)} of Agent X

prev_st, prev_et ← 0
i ← 0

Begin
    while AHS_x is not empty
    do
        work_i  AHS_x
        st,et  work_i
            if( st - prev_et ≤ Threshold )
            then
                work_i-1  (prev_st, et)
                prev_st, prev_et  work_i-1
            else
                i++
            end
            AHS_x  AHS_x - work_i
    done
End
```

▶▶ 그림 5. 패턴분석을 위해 worktime의 단편을 위한 병합 알고리즘

그림5는 병합 알고리즘의 의사코드이다. *worktime*은 각 수행한 작업의 시작시간과 완료시간으로 정해진다. 에이전트의 작업구간사이에서 지난 작업시간의 *endtime*와 다음 작업시간의 *stratime*의 시간차가 에이전트의 동작대기시간보다 적으면, 두 개의 구간은 하나로 병합되고, 에이전트가 연속적으로 일한 시간으로 구분된다.

2.3 공통 사용 패턴 생성

사용자 패턴을 분석하기 위하여 에이전트의 작업 구간을 파악하고, 병합 알고리즘을 통해서 불필요한 정보를 제거하고 사용자 패턴정보를 재가공하였다. 각 에이전트의 일일 패턴 정보을 이용하여, 한 달 동안 패턴을 통해서 각 에이전트의 일일 공통 패턴을 구하거나, 각 월요일의 패턴들을 통해서 에이전트의 월요일 사용패턴을 얻을 수 있다. 이와 같은 방법으로 그림6에서는 64번 에이전트의 한달 패턴을 구하였다.



▶▶ 그림 6. 64번 에이전트의 사용자 패턴

아래의 식은 공통사용패턴을 구하기 위한 과정으로써, AHS_x의 각 에이전트들의 작업들의 (starttime, endtime)들의 집합을 나타낸다. $MAHS_x$는 요일과 주간정보를 입력하여 그 기간에 해당하는 패턴들의 병합 패턴을 찾아 낸다. $CAUS_x$는 각 주어진 요일 또는 주간동안의 공통패턴을 찾아내기 위하여 주어진 기간의 패턴들의 공통분모를 찾아냄으로써, 사용자 패턴 기반의 스케줄링에 필요한 공통패턴을 생성한다.

Agent History Set (of Agent x)

$$AHS_x = \{(st_0, et_0), (st_1, et_1), ..., (st_n, et_n)\}, n = N_x$$
$$= \{(st_i, et_i) \mid 0 \le i \le N_x\}$$

Merged Agent History Set (of Agent x)

$$MAHS_x = \{(st'_0, et'_0), (st'_1, et'_1), ..., (st'_k, et'_k)\}, k = M_x \le N_x$$
$$= \{(st'_i, et'_i) \mid 0 \le i \le M_x \le N_x\}$$
$$= \{(dst_w^d, det_w^d) \mid 0 \le d \le 6, 0 \le w \le 3\}$$

Common Agent Usage Set (of Agent x)

FCD: Find Common Day

FCW: Find Common Week

$$CAUS_x = \{FCD(MAHS_x, day) \mid 0 \le day \le 31\}$$
$$= \{FCD(MAHS_x, date, week) \mid 0 \le date \le 6, 0 \le week \le 3\}$$

$$FCD(MAHS_x, day)$$
$$= \left\{\exists rday \mid rday \in \bigcap_{day=0}^{31} MAHS_x^{day}\right\}$$
$$= \left\{\exists w \mid w \in \bigcap_{day=0}^{31} term(st'_i, et'_i)_{day}, 0 \le i \le M_x\right\}$$

$$FCW(MAHS_x, day, week)$$
$$= \left\{\exists rday \mid rday \in \bigcap_{w=0}^{3} \bigcap_{d=0}^{6} (MAHS_x^{(w,d)})\right\}$$
$$= \left\{\exists w \mid w \in \bigcap_{w=0}^{3} term(dst_w^d, det_w^d), 0 \le d \le 6\right\}$$

Ⅳ. 성능 분석

코리아앳홈 에이전트의 성능을 분석하기 위해서, 여러 가지 수행모드에서의 성능을 비교·분석하고, 사용자패턴기반의 스케줄링 기법을 적용한 에이전트의 작업수행 성능을 비교분석한다.

1. 항상동작모드와 동작대기모드

사용자들은 두 가지 다른 모드로 에이전트를 수행할 수 있다. 항상동작 모드로 설정하여 컴퓨터가 켜져있는 동안 항상 작업을 수행하도록 할 수 있으며, 동작대기시간을 설정하여 사용자 입력이 없어진 이후의 설정 시간이후부터 에이전트의 작업을 수행하도록 설정할 수 있다. 여러 가지 모드에서의 성능을 비교하기 위해서 기본 단위작업의 성능(유닛성능)을 구하였다. 유닛 성능을 얻기 위해 DB에서 64번 에이전트의 각 응용의 평균시간을 구했다. {기후예측응용, 유전체분석응용}의 집합을 하나의 단위작업으로 단위성능은 22270(1016*1245) sec/ret이다.

Unit Performance (of Platform X)

$$Perf_{Unit}^X = \frac{\text{the time taking to get results}}{\text{the number of results}}$$
$$= \frac{ut_X\ time(\text{sec})}{1\ result} = ut_X(\text{sec}/\ ret)$$

Fully Shared (Volunteered) Mode

$$Perf_{FSM_{day}}^X = \frac{24_{hr}}{Perf_{Unit}^X} \cdot df_X = \frac{24_{\text{hr}} * 6o_{\text{min/hr}} * 60_{\text{sec/min}}}{ut_X} \cdot df_X$$

df_X = degrade factor of Platform X

Timeout (Partially Shared) Mode

$$Perf_{TM_{day}}^X = \frac{AgentTime_X}{Perf_{Unit}^X} \cdot sdf_X$$
$$= \frac{\sum_{i=0}^{n} worktime(AHS_i^X)}{ut_X} \cdot sdf_X$$

sdf_X = switching degrade factor of Platform X

FSM모드에서 두 응용의 작업을 수행시켰다면 24*60*60/2270 = 86400/2270=38(ret)개의 결과를 얻을 수 있다. 하지만 degrade factor(df)가 있기 때문에 100%자원을 사용할 수 있는 환경이 아니라, 사용자와 자원을 경우 하는 경우라면, 38개보다 더 적은 결과를 얻게 될 것이다.(df가 50%라면 사용자와 에이전트가 같은 CPU시간을 공유하는 것을 의미한다. 이때는 38의 반인, 19개의 결과를 얻을 수 있는 것으로 예상할 수 있다)

동작대기시간 모드로 에이전트가 작업을 수행하는 PSF모드에서의 성능을 구해보면, 64번 에이전트의 하루 동안의 총 61335초 동안 27개의 결과를 반환 했다.

User Usage Pattern Mode

$$Perf_{UUP_{day}}^X = \frac{\text{the agent time of common usage pattern}}{Perf_{Unit}^X}$$
$$= \frac{CUUP_X}{Perf_{Unit}^X} = \frac{\sum_{i=0}^{n} worktime(CAUS_i^X)}{ut_X}$$

UUP 모드에서 사용자의 공통 패턴을 적용하여 54600초 동안 24개의 결과를 얻을 수 있다. FSM이나 PSM보다는 적은

개수의 결과이지만, 사용자가 보여 왔던 사용패턴을 고려한 스케줄로 가능한 사용자를 방해하지 않는 범위에서 실행하는 것을 전제로 계산되었다.

Ⅴ. 결론 및 향후 과제

본 연구에서 사용자 사용패턴에 기반한 로컬스케줄링 기법을 제안하였다. 특히 코리아앳홈 에이전트 사용자들이 에이전트를 사용하는데 있어 불편함을 느끼지 않고 자신의 PC자원을 제공할 수 있도록 하는데 그 목적이 있다. 사용자 패턴을 분석하기 위해서 서버와 에이전트간의 주고받는 메시지를 이용하여 에이전트의 작업구간을 정보를 추출했다. 실험을 위해서 한달간의 메시지 로그로부터 패턴을 분석하였다. 세가지 다른 모드에서의 에이전트의 성능을 분석하여 비교 하였으며, 사용자 사용패턴기반의 로컬스케줄링을 적용한 수행모드에서의 성능이 FSM모드나 PSM에 비해서 조금 떨어진다. 그러나 이 모드에서는 사용자가 에이전트를 수행할 때 방해하지 않고 사용자 응용을 수행할 수 있도록 로컬스케줄링 기법을 적용하였으며, 이를 통해 더 많은 에이전트의 참여를 유도하여 장기적으로 더 많은 컴퓨팅 파워를 모으는 전략이 될 수 있다. 앞으로 이 스케줄링 기법을 적용한 에이전트를 배포하여 적용할 예정이며, 자원들간의 매치메이킹 기법을 이용하여 작업분배 방법을 개선하는등 여러 방면의 개선 방안들을 향후 과제에서 수행할 것이다.

▌참 고 문 헌▐

[1] BOINC: A System for Public-Resource Computing and Storage, David P. Anderson, Space Science Lab., Univ. of California at Berkeley

[2] D. Toth & D. Finkel, A Comparison of Techniques for Distributing File-based Tasks for Public-Resource Computing, 17th IASTED International Conference on Parallel and Distributed Computing and Systems, Phoenix, Arizona, USA, 2005, pp.398~403

[3] http://www.koreaathome.org

[4] Xiaojuan Ren, Seyong Lee, Rudolf Eigenmann, Saurabh Bagchi, Resource Availability Prediction in Fine-Grained Cycle Sharing Systems, 15th IEEE International Symposium on High Performance Distributed Computing, Paris, 2006, pp.93~104

[5] Derrick Kondo, Gilles Fedak, Franck Cappello, Andrew A. Chien and Henri Casanova, Characterizing Resource Availability in Enterprise Desktop Grids. Future Generation Comp. Syst. 2007 Vol.23, No.7 pp.888~903