데스크탑 그리드에서 자원 사용 경향성을 고려한 효율적인 스케줄링 기법

(An Efficient Scheduling Method Taking into Account Resource Usage Patterns on Desktop Grids)

현 주 호 [†] 이 승 구 [††] 김 상 철 [†††] 이 민 구 [††††]
(Ju-Ho Hyun) (Sunggu Lee) (Sang Cheol Kim) (Min-Gu Lee)

요 약 데스크탑 그리드는 컴퓨팅 집약적인 분산 어플리케이션을 수행하는데 있어서 유망한 플랫폼으로 부각되고 있다. 그러나 비 신뢰적이고 예측할 수 없는 자원의 특성 때문에 데스크탑 그리드에서 병렬 어플리케이션의 효율적인 스케줄링은 어려운 문제로 알려져 있다. 이에 따라서 빈약한 스케줄링 능력과 함께 현재 데스크탑 그리드는 고 처리 어플리케이션(high throughput application)의 실행에는 적합하지만 빠른 반환 시간을 요구하는 어플리케이션의 실행을 지원하는데 있어서 어려움을 갖는다. 빠른 반환 시간을 요구하는 어플리케이션의 효율적인 실행은 어플리케이션의 전체 실행 시간(makespan)을 축소함으로써 해결할 수 있는 문제로써 데스크탑 그리드가 이를 지원할 수 있게 하는 것은 매력적인 제안이 될 것이다. 본 논문에서는 데스크탑 그리드에서 효율적인 어플리케이션의 실행을 지원하기 위한 새로운 스케줄링 방법을 제안한다. 7주간의 시간동안 40대의 데스크탑에서 추출된 추적(trace) 데이타의 분석을 통해서 데스크탑 사용 경향성과 비 신뢰적인 데스크탑의 영향이 스케줄링의 성능을 개선하는데 있어서 활용 될 수 있음을 확인하였고 이 요소들을 고려함으로써 데스크탑 그리드의 비 신뢰적이고 예측할 수 없는 자원의 특성을 스케줄링에 적절하게 반영 할 수 있는 스케줄링 기법이 제안되었다. 제안된 스케줄링 기법은 실제 데스크탑들의 행동 패턴을 반영한 추적 기반 시뮬레이션(trace-driven simulation)을 통해서 기존의 스케줄링 방법들과 스케줄링 성능이 비교되었고 시뮬레이션 결과를 통해서 제안된 스케줄링 기법이 기존의 데스크탑 스케줄링 기법들에 비해서 병렬 어플리케이션의 전체 실행 시간을 축소하고 중지(suspension)와 장애(failure)의 발생 빈도를 줄이는 것을 보여준다.

키워드 : 데스크탑 그리드, 병렬 어플리케이션, 전체 실행시간, 추적 기반 시뮬레이션

Abstract A desktop grid, which is a computing grid composed of idle computing resources in a large network of desktop computers, is a promising platform for compute-intensive distributed computing applications. However, due to reliability and unpredictability of computing resources, effective scheduling of parallel computing applications on such a platform is a difficult problem. This paper proposes a new scheduling method aimed at reducing the total execution time of a parallel application on a desktop grid. The proposed method is based on utilizing the histories of execution behavior of individual computing nodes in the scheduling algorithm. In order to test out the feasibility of this idea, execution trace data were collected from a set of 40 desktop workstations over a period of seven weeks. Then, based on this data, the execution of several representative parallel applications were simulated using trace-driven simulation. The simulation results showed that the proposed method improves the execution time of the target applications significantly when compared to previous desktop grid scheduling methods. In addition, there were fewer instances of application suspension and failure.

Key words : desktop grid, application, total execution time, trace-driven simulation

† 정 회 원 : 삼성전자 DM연구소 연구원
juho.hyun@samsung.com
†† 종신회원 : 포항공과대학교 전자전기공학과 교수
slee@postech.ac.kr
††† 정 회 원 : 한국전자통신연구원 임베디드 연구단 연구원
sheart@etri.re.kr
†††† 학생회원 : 포항공과대학교 전자전기공학과
bluehope@postech.ac.kr
논문접수 : 2006년 1월 15일
심사완료 : 2006년 4월 7일

1. 서 론

지난 수십 년 동안, 개인용 컴퓨터의 성능은 급격하게 증가되었고 여러 연구들을 통해서 대부분의 워크스테이션과 데스크탑 컴퓨터들이 많은 시간 동안 사용되지 않거나 낮은 수준의 사용률을 가지는 것으로 밝혀졌다

[1-3]. 이런 사실은 네트워크로 연결된 데스크탑과 워크스테이션들의 유휴 컴퓨팅 자원들을 추출하여 외부 컴퓨팅 작업(guest job)을 수행하는데 이용 하는 사이클 도용 시스템(cycle stealing system)의 출현 배경이 되었다. 사이클 도용 시스템은 초기에는 근거리 통신망을 기반으로 구축 되었지만 네트워크의 성능이 개선되면서 광역 통신망을 기반으로 하는 인터넷 기반 사이클 도용 시스템으로 발전하였고 이것이 현재 데스크탑 그리드로 불리는 분산 컴퓨팅 환경이라 할 수 있다. 데스크탑 그리드는 네트워크로 연결 된 수많은 데스크탑 또는 개인용 컴퓨터 등으로 구성된 시스템으로써 SETI@home[4]과 Folding@home[5]등과 같은 프로젝트가 성공하면서 컴퓨팅 집약적인 분산 어플리케이션을 실행하는데 있어서 유망한 플랫폼으로 부각 되었다.

그러나 데스크탑 그리드를 구성하는 컴퓨팅 자원들은 비신뢰적이고(unreliable) 예측할 수 없는(unpredictable) 특성을 가지고 있기 때문에 병렬 어플리케이션을 효율적으로 스케줄링 하는데 있어서 어려운 문제를 가지고 있다. 전용된 컴퓨팅 자원을 가지고 있는 클러스터 환경이나 일반적인 그리드 환경과는 달리 데스크탑 그리드의 컴퓨팅 자원인 데스크탑은 서로 다른 주인(owner)에 의해서 소유되고 데스크탑 그리드에 자원으로 제공되는 것은 주인의 사용 성향 및 데스크탑 사용 목적에 따라 가변적이다. 결국 각 데스크탑은 주인으로부터 사용되지 않을 경우에만 데스크탑 그리드의 컴퓨팅 자원으로 활용할 수 있고 데스크탑이 언제 주인에 의해 사용 될지 언제 전원이 꺼지고 다시 재시작 될지 예측하기 힘든 측면이 있다. 이런 점이 바로 각 데스크탑을 신뢰성 없고 상태를 예측할 수 없는 특성을 가진 컴퓨팅 자원으로 만들고 데스크탑 그리드에서 빠른 반환 시간을 요구하는 병렬 어플리케이션의 효율적인 실행과 신뢰성 있는 실행은 이런 데스크탑 그리드의 특성에 기인하는 빈약한 태스크 스케줄링 능력으로 인하여 해결하기 어려운 문제점으로 남아있다.

본 논문에서는 이런 문제점을 해결하기 위해 데스크탑의 사용 경향성을 정의하고 데스크탑의 특성을 고려하는 새로운 스케줄링 기법이 제안한다. 데스크탑의 사용 경향성은 데스크탑이 일반적으로 사용되는 성향을 나타내는 것으로 본 논문에서는 각 데스크탑의 사용 경향성을 확률적인 값으로써 표현하고 이를 스케줄링에 이용함으로써 데스크탑들의 각각의 독특한 특성을 반영하였다. 이것은 특정 시간 구간의 데스크탑의 가용성 여부를 예측 할 수 있게 하는 방법으로써 신뢰성이 없고 자원 상태의 예측이 어려운 데스크탑 그리드에서 자원 관리 및 스케줄링을 효율적으로 할 수 있는 새로운 접근이 될 것이다. 본 논문에서는 제안된 스케줄링 기법과 함께 독립적이고 동일한 크기를 가진 태스크들로 구성된 병렬 어플리케이션의 전체 실행 시간(makespan)을 최소화하는 것과 신뢰성 있는 실행을 위해 병렬 어플리케이션의 장애 및 중지의 발생을 축소하는 것을 목적으로 하며 추적 기반 시뮬레이션(trace-driven simulation)을 통해서 기존의 방법과의 비교를 통해서 유용성을 나타내었다.

2. 관련연구

병렬 어플리케이션의 효율적인 수행을 위한 스케줄링 기법에 대해서 멀티프로세서 환경에서 오래 전부터 많은 연구가 진행 되어 왔고 최적의 해에 근접한 기법들도 제시되었다. 그러나 이 연구들은 병렬 어플리케이션의 수행에 전용된(dedicated) 멀티프로세서 환경에서의 결과로써 어플리케이션 실행에 있어서 데스크탑의 비신뢰적인 특성으로 인해 장애 및 중지가 빈번하게 발생하는 데스크탑 그리드에서의 스케줄링과는 차이가 있다.

또한, 사이클 도용 시스템과 데스크탑 그리드 환경에서 각기 다른 주인을 가진 컴퓨팅 자원의 유휴한 상태를 예측하기 위한 방법을 제시한 워크스테이션을 대상으로 하는 연구들[3,9-12], 그리고 데스크탑을 대상으로 하는 연구들[13-15]이 있었지만 이들 모두 데스크탑의 유휴 상태를 예측하는데만 초점을 맞추고 연구가 진행되어 비 신뢰적인 데스크탑에 대한 고려가 없었고 이를 실제로 데스크탑 그리드에서의 스케줄링에 적용한 방법은 제시되지 않았다. 현재 데스크탑 그리드에서 스케줄링 문제는 연구가 많이 진행 되지 않은 상태로써 특히 병렬 어플리케이션의 전체 실행 시간을 축소하는 것을 목적으로 하는 스케줄링 기법에 대한 연구는 데스크탑 그리드의 비 신뢰적이고 예측하기 힘든 컴퓨팅 자원의 영향으로 미미한 상태이다.

현재의 대부분의 데스크탑 그리드들은 XtreamWeb [6]에서 사용되는 것처럼 선도착 선처리(FCFS: First-Come First-Served) 방식이 사용된다. 이 기법은 스케줄러의 준비 큐에 먼저 가용한 상태로 등록된 호스트가 먼저 작업을 할당 받는 방식으로 간단하지만 어플리케이션의 완료 시간을 축소하기 위한 고려가 없기 때문에 데스크탑 그리드에서 병렬 어플리케이션의 빠른 반환을 지원하지는 못한다. [7]에서는 이전까지의 기존의 데스크탑 그리드 스케줄링과는 달리 병렬 어플리케이션의 전체 실행 시간 또는 반환 시간을 축소하는 것을 목적으로 스케줄링 기법들을 제시하였다. 높은 CPU 성능을 가진 호스트에 높은 우선 순위를 주는 클럭 스피드에 의한 우선순위 채택 방식(PRI_CR: Resource Prioritization by Clock Rate)과 자원 배제 기법을 이용한 스케줄링 기법 등이 제시되었고 시뮬레이션을 성능 비교

를 통해서 선도착 선처리에 비해서 뛰어난 성능을 보이는 것을 나타내었다.

3. 데스크탑 그리드 모델

데스크탑 그리드는 스케줄러의 역할을 수행하는 서버와 데스크탑을 가진 N개의 호스트들로 구성된다. 데스크탑 그리드의 컴퓨팅 자원인 데스크탑은 대부분 서로 다른 주인에 의해서 관리되며 주인에 의해서 사용되지 않고 데스크탑 그리드에 응답할 수 있는 상태 일 때 그리드 작업을 수행하는데 이용될 수 있다.

데스크탑 그리드에 참여한 모든 데스크탑에는 그리드 클라이언트 프로그램이 설치되어 있다. 데스크탑 그리드 클라이언트는 다음과 같은 역할을 수행한다.

- 데스크탑의 가용성을 결정하기 위해서 실시간으로 CPU 사용률, 키보드/마우스 입력, 메모리 사용량 등의 시스템 정보를 측정한다.
- 측정된 시스템 정보를 토대로 데스크탑이 유휴한 상태 인지 감지한다.
- 데스크탑이 유휴한 상태에 놓여지게 되면 서버로 데스크탑이 유휴하다는 것을 알려주기 위해 유휴 상태 통지 메시지(notification message)를 보낸다.
- 스케줄러로부터 그리드 작업을 할당 받고 데스크탑의 운영체제에 그리드 작업을 전달하여 실행한다.
- 데스크탑에서 실행되고 있던 그리드 작업 또는 게스트 작업은 데스크탑에 지역 작업(local job)이 제출되거나 주인이 데스크탑을 사용하기 시작 할 때 중지(suspend)되고 이후에 다시 데스크탑이 유휴한 상태에 놓이게 될 때 다시 중지 되었던 작업을 이어서 재시작(resume) 하게 된다.

서버는 스케줄러로써 데스크탑 그리드에 제출된 작업들을 유휴한 상태에 있는 호스트에 할당을 하는 역할을 한다. 서버의 세부적인 역할은 다음과 같다.

- 병렬 어플리케이션이 데스크탑 그리드에 제출되면 이를 부 작업들로 나누어서 작업 큐에 스케줄링이 될 그리드 작업들로 유지한다.
- 유휴한 상태에 놓인 데스크탑 정보를 수집하고 이들을 호스트 큐에서 관리한다.
- 채택하고 있는 스케줄링 전략에 따라서 그리드 작업을 적절한 호스트에 할당한다.
- 그리드 작업이 할당되어 실행되고 있는 호스트의 그리드 클라이언트에 주기적으로 킵 얼라이브 메시지(keep-alive message)를 보내고 응답을 받는다.
- 만일 킵 얼라이브 메시지의 응답이 클라이언트로부터 정해진 시간 안에 도착하지 않으며 해당 호스트에 할당된 그리드 작업은 실패(fail)했다고 간주하게 되고 재 스케줄링을 통해서 다른 호스트에서 해당 작업을 처음부터 수행 되도록 한다.

그림 1은 데스크탑 그리드의 기본 구조를 나타내고 있다.

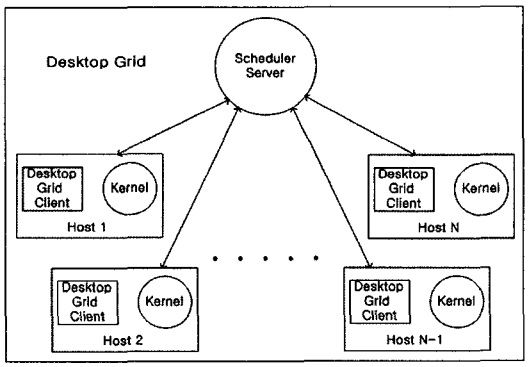

그림 1 데스크탑 그리드의 구성

4. 스케줄링 성능 개선 요소

4.1 데스크탑 사용 경향성

대부분의 데스크탑들은 각각 서로 다른 고유한 패턴에 따라서 사용된다. 예를 들면, 어떤 데스크탑은 오전 9시부터 오후 6시까지의 비즈니스 시간에 주로 사용되는 반면 다른 데스크탑은 늦은 오후 또는 새벽 시간에 주로 이용될 수 있다. 또한 어떤 데스크탑은 하루의 대부분 전원이 켜져 있는 반면 다른 데스크탑은 하루의 많은 시간에 걸쳐 전원이 꺼져 있을 수 있다. 본 논문에서는 이런 데스크탑의 특징을 데스크탑 사용 경향성(desktop usage pattern)으로 정의하였고 스케줄링에 활용 될 수 있음을 보였다.

데스크탑 사용 경향성은 호스트 가용성과 CPU 가용성으로 결정된다. 호스트 가용성은 하루의 특정 시간 구간에서 해당 호스트의 데스크탑이 응답이 있는지 없는지를 나타낸다. 이것은 단순히 호스트의 데스크탑의 전원이 꺼진 것만을 의미하는 것이 아니라 호스트 또는 네트워크의 장애를 포함한다. 반면에 CPU 가용성은 하루의 특정 시간 구간 안에서 데스크탑이 유휴한지 그렇지 않은지를 나타낸다. 이것은 데스크탑이 사용되는 경향을 나타내는 척도로써 CPU 사용률, 키보드/마우스 입력, 그리고 가용 상태 결정 임계치(recruitment threshold) 세 가지 요소에 의해서 결정된다[1,8]. 가용 상태 결정 임계치는 데스크탑이 가용 상태에 있는 것을 보장하기 위해 기다리는 시간으로 데스크탑이 사용되지 않기 시작한 시간으로부터 이 임계치를 초과하게 되면 데스크탑은 유휴 상태로 인식된다.

데스크탑들이 서로 구별되는 고유의 사용 경향성을 가진다는 점과 그 활용 가능성을 확인하기 위해서 7주

간의 시간 동안 매 초마다 40대의 window/XP 환경의 데스크탑으로부터 CPU 사용률, 메모리 사용량, 그리고 키보드/마우스 입력이 추적되어 데이타로 산출 되었다. 호스트 가용성과 CPU 가용성을 기준으로 추적된 데이타들이 분석되었으며 각각의 데스크탑의 사용 경향성이 구해졌다. 하루는 5분 크기의 시간 구간(time slot)들로 나누어 졌고 각 시간 구간에는 추적된 데이타를 바탕으로 계산된 호스트 가용성과 CPU 가용성을 반영하는 확률 값이 대응되었다. 이 확률 값은 특정 시간 구간에서 데스크탑이 사용된 날의 빈도수를 데스크탑의 정보가 추적된 날의 빈도수로 나눈 값으로 특정 시간 구간에서의 데스크탑의 사용경향을 나타낸다. 예를 들어 한 데스크탑이 10일 동안 성능 정보가 추적 되었고 특정 시간 구간에서 데스크탑이 사용되거나 네트워크 장애 및 전원의 꺼짐으로 인하여 응답 가능하지 않은 날의 수가 1일 이었다면 이때 확률 값은 0.1이 되고 이 데스크탑이 해당 시간 구간에서 주인에 의해서 사용 되거나 응답 가능 하지 않을 확률은 10%이고 반면에 데스크탑이 유휴 상태(idle state)에 있을 확률이 90%라는 것을 의미하게 된다. 이런 방식으로 데스크탑 사용 경향성은 각 시간 구간에 대응되는 확률적 값들의 연속적인 전개를 통해서 표현된다.

그림 2,3,4,5는 추적된 40대의 데스크탑들 중에서 4개의 표본 데스크탑들의 사용 경향성을 나타낸다. 각각의 표본 데스크탑들은 각각 확연히 구별되는 사용 경향성을 나타낸다. 그림 2는 비즈니스 시간인 오전 9시부터 오후 6시까지 주로 많이 사용되는 데스크탑의 사용 경향성으로써 가장 보편적인 데스크탑의 사용 경향을 나타낸다. 이와 대조적으로 그림 3은 늦은 오후부터 새벽까지 주로 많이 사용 되는 데스크탑의 사용 경향성을 나타내고 전체적으로 그림 2의 데스크탑에 비해서 데스크탑 사용률이 높은 것을 보여준다. 이 두 대의 표본 데스크탑들은 7주 동안 응답 가능한 상태를 유지 하였고 결과에서 나타난 데스크탑 사용 경향성의 차이는 결국 각 시간 구간마다의 CPU 가용성의 차이에 의해서 나타난 것으로 보여 진다.

그림 4는 하루의 대부분이 거의 사용 되지 않은 데스크탑의 사용 경향성이고 그림 5 역시 하루 중 많은 시간이 사용되지는 않는 데스크탑을 나타낸다. 그럼에도 불구하고 그림 4와 5는 서로 대조적인 결과를 보여 주고 있다. 그 이유는 두 데스크탑의 호스트 가용성의 차이로써 그림 4의 데스크탑은 계속 응답 가능한 상태를 유지하는 반면 그림 5의 데스크탑은 자주 전원 꺼져서 응답 가능하지 못한 경우가 많았기 때문이다. 데스크탑 그리드의 관점에서는 데스크탑이 주인에 의해 사용되고 있는 상태이거나 전원이 꺼져 있거나 네트워크 장애로

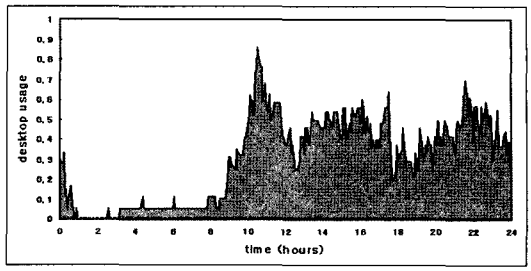

그림 2 비즈니스 시간(9AM~6PM)에 주로 사용되는 데스크탑(2.6GHz)의 사용 경향

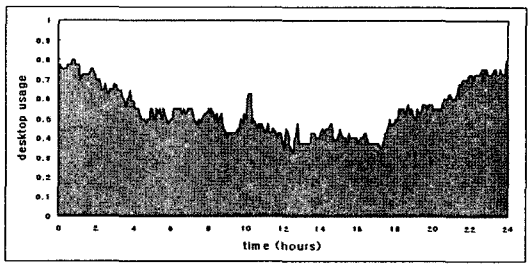

그림 3 늦은 오후부터 새벽시간까지 주로 사용되는 데스크탑(2.6GHz)의 사용 경향

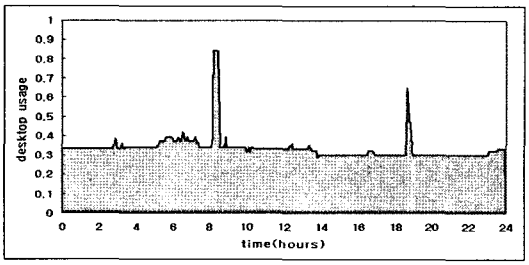

그림 4 하루의 대부분의 시간 동안 응답 가능 하고 거의 사용되지 않는 데스크탑(3.2GHz)의 사용 경향

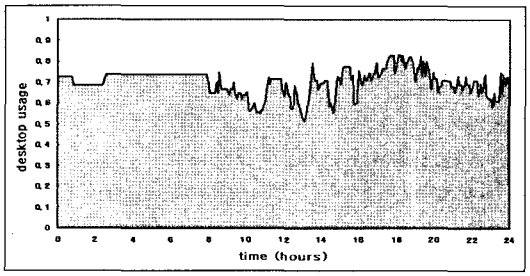

그림 5 자주 사용 되지 않지만 하루의 대부분 전원이 꺼져 있는(응답불능) 데스크탑(3.2GHz)의 사용 경향

응답할 수 없는 경우에 모두 데스크탑을 사용 할 수 없다. 따라서 데스크탑의 사용 경향성은 데스크탑이 응답할 수 없는 경우 역시 포함하고 결국 데스크탑의 사용

패턴뿐만 아니라 신뢰성도 표현하게 된다. 여기에서 가장 주목해야 할 사실은 데스크탑의 비 신뢰적이고 예측할 수 없는 특성들이 데스크탑들을 서로 구별되는 사용 경향을 갖도록 만든다는 점이고 그림 2,3,4,5를 통해서 알 수 있듯 사용 경향성은 각 데스크탑마다 확연히 다르게 나타 날 수 있다는 것이다. 이에 따라, 데스크탑 그리드에서 스케줄링에 이런 데스크탑의 사용 경향성을 고려 할 수 있다면 데스크탑의 특성을 보다 적절히 반영함으로써 이전까지의 빈약한 스케줄링의 능력이 개선될 수 있을 것으로 추측되었고 데스크탑 사용 경향성을 기반으로 하는 새로운 스케줄링 기법이 본 논문에서 제시되었다.

4.2 어플리케이션의 완료 시간을 지연 시키는 비 신뢰적인 호스트의 영향

병렬 어플리케이션은 여러 개의 부 작업(sub-task)들로 나누어져 실행되고 완료시간(completion time)은 모든 부 작업들이 끝나는 시간으로 결정된다. 그런데 만약 병렬 어플리케이션 중 하나의 부 작업의 완료시간이 다른 부 작업의 완료 시간에 비해서 현저하게 늦어지게 되면 결국 다른 모든 부 작업 들이 끝났을 때에도 단지 하나의 부 작업이 끝나지 않아서 어플리케이션의 완료 시간이 지연되게 된다. [7]에서는 이런 현상이 클럭 스피드가 낮은 데스크탑을 가진 호스트, 즉 성능이 낮은 호스트에 의해 발생한다고 주장 하였다.

그러나 본 논문은 실제로 데스크탑 그리드에서 전체 어플리케이션의 완료시간에 치명적인 영향을 미치는 요인은 비 신뢰적인 호스트에 의해서 임을 주장한다. 비 신뢰적이라는 것은 컴퓨팅 작업이 자주 중지되고 장애가 발생한다는 것을 의미한다. 특히 중지가 자주 오랫동안 발생하는 호스트는 전체 어플리케이션의 완료시간에 큰 영향을 주게 된다. 예를 들면 특정 데스크탑에서 그리드 작업이 수행되고 있을 때 데스크탑의 주인이 돌아와서 데스크탑을 사용하게 된다면 이때 그리드 작업은 중지(suspension)된다. 그러나 이때 이 중지 시간은 상황에 따라서 매우 길어질 수 있고 이는 전체 어플리케이션의 완료 시간을 극단적으로 지연 시킬 수 있게 된다. 본 논문에서는 비 신뢰적인 호스트가 실제로 얼마나 어플리케이션의 완료 시간에 영향을 미치는 지 시뮬레이션을 통해서 나타내었다. 기존의 FCFS를 스케줄링 방법으로 채택하는 100개의 호스트를 가진 데스크탑 그리드를 구성하여 실행 시간이 1.5GHz의 데스크탑에서 30분인 부 작업들로 구성되는 어플리케이션의 실행을 시뮬레이션 하였고 이 결과는 그림 6과 7에 나타난다. 그림 6은 데스크탑 그리드 환경에서 50개의 부 작업을 가진 어플리케이션을 실행한 결과이다. 작업의 80% 이상이 전체 실행 시간의 절반 정도인 2500초 정도에서

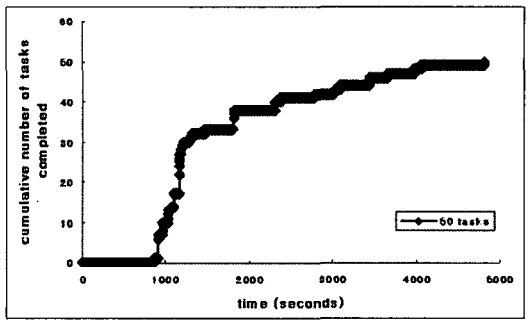

그림 6 비 신뢰적인 호스트가 전체 실행시간에 미치는 영향

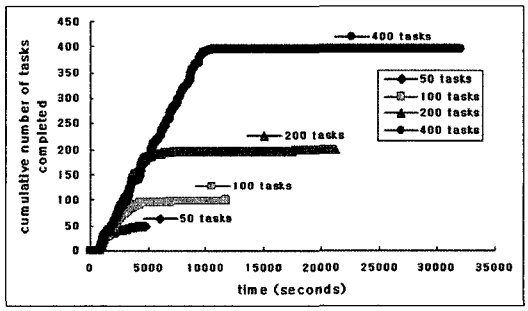

그림 7 작업 수가 50개~400일 때 신뢰성 없는 호스트의 영향에 의한 전체 실행 시간의 비교

끝난 것을 볼 수 있고 나머지 20%정도 남은 작업을 실행하는데 절반 이상의 시간이 소요 된 것이 확인 된다. 이는 소수의 신뢰성이 부족한 호스트에서 오랜 기간 동안 작업들이 중지되기 때문에 발생하는 것이 확인 되었다. 이 현상은 어플리케이션을 구성하는 작업의 수가 많아질수록 보다 확연히 드러나고 그림 7은 어플리케이션의 부 작업이 400개 일 때 전체 실행 시간의 거의 60% 이상의 시간이 전체 작업 중에서 단지 10% 이하의 작업을 완료하는데 소요되는 것을 보여준다.

5. 제안된 방법

5.1 데스크탑 사용 경향표

데스크탑 사용 경향성은 본 논문의 4.1장에서 설명된 방법에 의해서 확률 값들로 표현된다. 하루는 동일한 크기를 가진 여러 개의 시간 구간으로 나누어지고 각 시간 구간마다 데스크탑의 사용 성향과 신뢰성을 반영하는 확률 값이 대응된다. 모든 시간 구간에 대한 확률 값들은 데스크탑 사용 경향표(desktop usage propensity table)에 저장되고 관리된다. 이 표는 주기적으로 데스크탑 그리드 클라이언트에 의해서 갱신되며 데스크탑의 사용 경향성을 나타내게 된다. 데스크탑 그리드 클라이언트는 데스크탑이 유휴 하다는 것을 인지하게 되면 이

데스크탑 사용 경향표를 유휴 상태 통지 메시지와 함께 서버로 보내서 스케줄링에 이용하게 한다.

그림 8은 데스크탑 사용 경향표의 예시로써 어떻게 데스크탑 사용 경향성이 표현되는지를 나타낸다. 각 시간 구간의 크기는 t분(minute)으로 정해지고 데스크탑 사용 경향표의 전체 항목(entry)의 수는 (24 ×60)/t가 된다. 즉, 하루를 (24 ×60)/t개의 t분의 크기를 가진 시간 구간으로 나누어 표현한 것이다. 각 항목에는 4.1에서 정의한 것처럼 해당하는 시간 구간에 데스크탑이 주인에 의해서 사용 되거나 응답 가능 하지 않을 확률 값을 저장된다. 이 값은 결국 데스크탑 그리드 관점에서는 데스크탑의 불가용성(unavailability)을 표현한 것이고 결과적으로 데스크탑 사용 경향표는 하루의 각 시간 구간마다의 데스크탑의 사용 패턴과 신뢰성을 표현하게 된다.

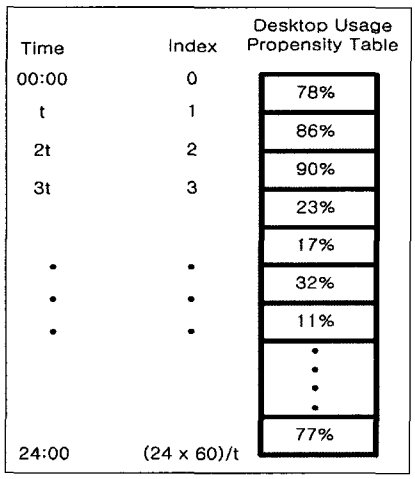

그림 8 데스크탑 사용 경향표

5.2 데스크탑 사용 경향성 기반 스케줄링 기법

데스크탑 사용 경향성 기반 스케줄링 기법(DUPBS: the desktop usage pattern based scheduling method)은 데스크탑 그리드에서 각 데스크탑의 사용 성향을 고려한 스케줄링 전략이다. 스케줄러는 데스크탑 클라이언트로부터 보내진 데스크탑 사용 경향표를 이용하여 각 데스크탑의 특성을 스케줄링에 고려한다. 데스크탑 사용 경향표를 통하여 스케줄러는 각 데스크탑이 유휴한 상태를 유지할 가능성을 추측 할 수 있다. 스케줄러는 유휴 상태가 되어 유휴 상태 통지 메시지와 함께 데스크탑 경향표를 보낸 데스크탑 정보들을 호스트 큐에 관리하게 되고 그리드 작업이 작업 큐에 존재할 때 각 데스크탑 경향표를 참조하여 그리드 작업이 끝나는 시간까지 유휴 상태를 유지할 확률, 즉 그리드 작업을 실행할 때 장애(failure)나 중지(suspension)가 발생하지 않을 확률이 가장 높은 데스크탑에게 우선적으로 그리드 작업을 할당하게 된다. 데스크탑 사용 경향성 기반 스케줄링 기법의 자세한 알고리즘은 그림 9에서 표기되었고 알고리즘의 각 단계에 대한 설명은 아래에서 명시된다.

- (1): 스케줄링이 시작되는 시간을 데스크탑 경향표의 대응되는 항목을 지시하는 시작 색인(S)으로 변환한다. 시작 색인은 특정 그리드 작업을 할당 받았을 때 호스트들이 작업을 시작하는 시간에 대응하는 시간 구간을 지시한다. (실제로 모든 호스트들이 특정 그리드 작업을 할당 받고 시작 할 수 있는 시간은 네트워크 상태 등에 따라서 다를 수 있지만 컴퓨팅 집약적인 어플리케이션의 실행을 목적으로 하고 있기 때문에 작업의 실행 시간에 비해 극히 미미한 데이타 전송 시간은 무시 되었고 호스트 큐에 존재하는 모든 호스트들은 특정 그리드 작업을 할당 받았을 때 같은 시간에 시작한다고 가정되었다.)
- (2): 작업 큐에 스케줄링이 되지 않은 그리드 작업이 있고 호스트 큐에 유휴 호스트가 존재 할 때 스케줄링 알고리즘은 반복하여 스케줄링을 실행한다.
- (3): 작업 큐로부터 큐의 첫 부분(head)에 존재하는 작업을 선택한다. (제출된 병렬 어플리케이션은 여러 개의 부 작업들로 나누어지고 작업 큐에서 관리 된다.)
- (4) (5) (6) (7): (4)에서 (7)까지는 호스트 큐에 존재하는 모든 호스트들에 대해서 (3)에서 선택된 작업(T_i)을 끝날 때까지 장애나 중지가 발생하지 않을 확률을 구하는 일을 수행한다. (5)에서는 각 호스트 H_j의 상대적인 성능(R_j)과 작업 T_i의 길이 $Length(T_i)$를 통해서 각 호스트의 데스크탑 경향표에서 작업 T_i가 끝나는 시간 구간의 항목을 가리키는 완료 색인(E)을 구한다(각 호스트의 상대적인 성능은 각 데스크탑의 CPU 클럭 속도를 척도로 만들어지고 결국, 부 작업 T_i의 길이 $Length(T_i)$는 어떤 성능을 가진 데스크탑에 할당되는지에 따라서 유동적으로 변하게 된다. 예를 들면 데스크탑 그리드의 기준 데스크탑 성능이 1.0GHz이고 부 작업의 길이가 10분이라면 2.0GHz의 데스크탑에서는 부 작업의 길이는 5분이 될 것이고 반면에 0.5GHz의 데스크탑에서는 부 작업의 길이는 20분이 될 것이다.) (6)에서 pk는 데스크탑 경향표의 한 항목에 저장되어 있는 확률 값으로써 특정 시간 구간에 데스크탑이 유휴 상태가 아닐 확률을 나타내고 있다. 따라서 이 시간 동안 데스크탑이 유휴 상태일 확률은 1-pk가 된다. PV_j는 S에 해당하는 시간부터 E에 해당하는 시간까지 데스크탑이 유휴 상태에 있을 확률을 나타내는 값으로 작업 T_i가 호스트 H_i

에 할당 되었을 때 장애나 중지가 발생하지 않고 끝나게 될 확률을 나타낸다. 연속적인 사건이 발생할 확률은 각 사건의 확률의 곱이므로 PV는 S부터 E까지의 항목들에 해당하는 확률 값들의 곱으로 표기 된다. 예를 들어서 만약 한 호스트의 데스크탑 사용 경향표가 그림 8과 같고 작업의 시작과 끝을 나타내는 색인 S와 E는 각각 0과 3이라면 이 호스트의 PV값은 (1-0.78) ×(1-0.86) ×(1-0.9) ×(1-0.23)의 결과인 0.00237을 갖게 된다.

- (8) : (7)에서 구한 호스트 큐의 모든 호스트들의 PV 값들은 서로 비교되어지고 작업 T_i는 이 중 가장 큰 PV 값을 가진 호스트에 H_l에 할당 된다.
- (9) (10) : 작업 큐에서 T_i가 삭제되고 호스트 큐에서 H_l가 삭제 된 후 (2)의 조건에 따라서 반복한다.

S : The start index in the desktop propensity table
E : The end index in the desktop propensity table
R : The relative speed between hosts
H : The set of idle hosts
T : The set of non-scheduled tasks
PV : The predicted value that a task will be finished on a host without suspension or failure
p : The probabilistic value corresponding to an entry in the desktop propensity table
$Length(t)$: The size of a task t

(1) Transform start time into S in the desktop propensity table.
(2) while (there are unscheduled tasks in task queue) and (there are idle hosts in host queue) do
(3) Select the first task T_i, from the task queue.
(4) for each host H_j in the host queue do
(5) Using R_j and $Length(T_i)$, calculate E (end index).
(6) Compute $PV_j = \prod_{k=S}^{E}(1-p_k)$
(7) end for
(8) Assign task T_i to the host H_l that have the highest PV
(9) Delete T_i from the task queue, delete H_l from the host queue
(10) end while

그림 9 데스크탑 사용 경향성 기반 스케줄링

5.3 자원 배제 기법을 적용한 스케줄링 기법

병렬 어플리케이션의 전체 완료 시간은 모든 부 작업의 실행이 완료 되는 시점이고 하나의 부작업의 실행이 완료되는 것이 지연될 때 전체 어플리케이션의 완료 시간이 지연 되는 문제가 있다. [7]에서는 데스크탑 그리드에서 이런 문제점은 성능이 낮은 데스크탑에 의해서 발생한다고 주장하였고 이를 해결하기 위해서 자원 배제(resource exclusion) 기법을 제시하여 스케줄링의 성능 개선을 보여 주었다. 데스크탑의 클럭 스피드에 따라서 기준 이하의 데스크탑들이 스케줄링에 제외 되었고 평균 CPU 클럭 스피드에 비교해서 50% 미만의 클럭 스피드를 가진 데스크탑을 배제시키는 방법(EXCL-S.5)이 뛰어난 성능을 보이는 것을 나타내었다.

그러나 4.2장에서 언급한 것처럼 비 신뢰적인 호스트는 어플리케이션의 완료시간을 지연 시키는데 치명적인 영향을 미치는 요소로써 본 논문에서는 데스크탑 그리드의 사용성향을 이용하여 그리드 작업이 할당 되었을 때 중지 될 확률이 높은 데스크탑 자원을 배제 시키는 기법이 제안되었다. 이는 자원 배제를 적용한 데스크탑 사용 경향성 기반 스케줄링 기법(DUPBS-EX)로 명명되었고 데스크탑 경향표에서의 시간 구간의 크기를 T라고 하고 병렬 어플리케이션을 구성하는 부 작업의 길이를 L이라고 할 때 부 작업을 장애나 중지 없이 실행을 시킬 확률(PV)이 T/L 보다 작은 호스트들이 배제 되었다.

6. 시뮬레이션

6.1 시뮬레이션 환경

5장에서 제안된 스케줄링 기법의 유효성을 검증하기 위해서 추적 기반 시뮬레이션이 수행 되었다. 7주 동안 40대의 데스크탑으로부터 매 초마다 CPU 사용률, 메모리 사용량, 그리고 키보드/마우스 입력이 추적되어 데이타로 산출 되었다. 이 데이타를 통해서 각 데스크탑의 상태 및 행동 패턴이 결정되었고 시간에 따르는 데스크탑의 상태 변화는 호스트 가용성과 CPU 가용성을 이용하여 결정 되었다. CPU의 유휴 상태는 CPU 사용률이 5%이하이고 키보드/마우스 입력이 없고 1분의 유휴 상태 결정 임계치가 만료 되었을 때로 정의 되었다. 또한 7주 동안 축적된 데이타를 통해서 각 데스크탑의 데스크탑 경향표가 만들어 졌고 하루를 5분 크기를 가진 시간 구간으로 구성하여 데스크탑의 경향을 나타내었다. 그리고 시뮬레이션에서 구축된 데스크탑 그리드는 추적 데이타를 사용하여 100개의 호스트로 구성 되었다. 각 호스트의 데스크탑 성능은 733MHz에서 3.2GHz까지의 범위에서 분산되었고 기준 호스트의 성능은 1.5GHz로 정의 되어 각 호스트의 상대적인 속도가 이에 비례하여 결정 되었다.

위와 같이 구축된 데스크탑 그리드 플랫폼 모델에서 여러 개의 부 작업으로 구성된 병렬 어플리케이션의 실행의 시뮬레이션을 수행하였다. 병렬 어플리케이션을 구성하는 부 작업들은 모두 독립적이고 동일한 크기를 갖는 형태로써 부 작업의 크기와 개수를 변화 시키면서 시뮬레이션이 진행 되었다.

6.2 시뮬레이션 결과

제안된 스케줄링 기법의 성능은 FCFS와 [7]에서 제시된 PRI_CR 및 EXCL-S.5와 성능이 비교 되었다. 각 시뮬레이션은 어플리케이션의 부 작업의 개수와 길이를 변화시키면서 100회씩 수행 되었고 각 스케줄링 기법의 성능은 병렬 어플리케이션의 평균 전체 실행시간(make-

span)과 병렬 어플리케이션을 실행 발생하는 장애와 중지의 평균 빈도수를 척도로 하여 비교가 이루어 졌다.

6.2.1 어플리케이션의 부 작업의 수가 50개일 때

그림 11은 50개의 부 작업으로 구성된 병렬 어플리케이션을 각 스케줄링 기법에 따라 데스크탑 그리드에서 실행 했을 때의 평균 완료시간을 나타낸다. 이 시뮬레이션 결과에서 FCFS와 PRI_CR, 그리고 본 논문에서 제시된 DUPBS는 비슷한 성능을 나타낸다. 실제로 이 스케줄링 기법들은 상당한 차이가 있지만 비슷한 성능이 나오게 되는 현상은 특정 시간에 스케줄링을 해야 할 작업 수에 비해 유휴 호스트의 수가 적기 때문이다. 예를 들면 특정 시간에 10개의 유휴 호스트가 존재하고 이때 5개의 작업을 스케줄링 한다면 위의 스케줄링 기법들은 모두 구별되는 다른 결과를 만들 것이다. 그러나

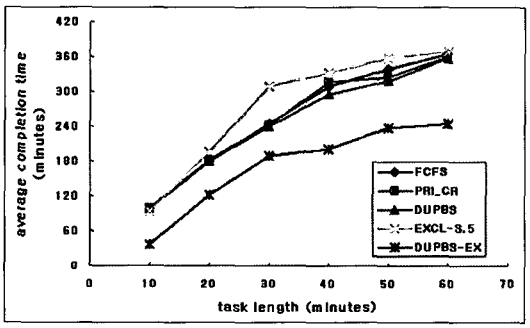

그림 10 부 작업 수가 50개인 어플리케이션의 평균 완료시간

표 1 부 작업 수가 50개일 때 어플리케이션이 완료 될 때까지 발생하는 평균 장애(failure)의 빈도 수

Length (minutes)	FCFS	PRI_CR	DUPBS	EXCL-S.5	DUPBS-EX
10	0.34	0.35	0.34	0.40	0.06
20	0.64	0.63	0.62	0.78	0.19
30	0.71	0.72	0.71	1.02	0.31
40	1.20	1.22	1.18	1.46	0.50
50	0.96	0.94	0.97	1.42	0.30
60	1.99	1.97	1.92	2.80	0.41

표 2 부 작업 수가 50개일 때 어플리케이션이 완료 될 때까지 발생하는 평균 중지(suspend)의 빈도 수

Length (minutes)	FCFS	PRI_CR	DUPBS	EXCL-S.5	DUPBS-EX
10	32.18	31.76	31.70	32.51	13.27
20	67.92	67.06	66.83	73.98	33.17
30	105.16	103.86	102.58	128.07	52.16
40	147.56	147.74	145.50	175.94	64.22
50	151.10	147.44	146.49	184.92	78.42
60	196.75	195.61	195.81	179.82	95.50

5개의 유휴 호스트가 존재하고 10개의 작업이 존재한다면 이때는 각 스케줄링 기법은 각 작업을 실행 시킬 호스트를 모두 다른 순서로 선택 하겠지만 결국 모든 호스트를 작업을 할당 할 때 사용 하게 되고 이때의 스케줄링 결과는 비슷해지게 된다. 반면에 DUPBS-EX는 다른 스케줄링 기법에 비해 현격하게 개선된 성능을 나타낸다. 이 기법은 장애와 중지가 빈번하게 일어나는 호스트를 스케줄링에서 배제시킴으로써 장애와 중지의 발생을 줄여 성능 개선을 이루었다. DUPBS-EX는 데스크탑 사용 성향을 이용하여 장애와 중지가 일어날 확률을 예측하였기 때문에 다른 스케줄링 기법에 비해서 장애가 발생하는 빈도수와 중지가 발생할 빈도수가 표 1과 표 2에서 보여주듯이 각각 약 4배와 2배 이상 줄어들었다.

6.2.2 어플리케이션의 부 작업의 수가 100개일 때

그림 11은 부 작업의 수가 100개인 병렬 어플리케이션의 평균 완료 시간을 나타낸다. DUPBS-EX는 작업의 수가 100개로 증가한 경우에도 다른 스케줄링 방법들에 비해 좋은 성능을 나타내었다. 표 3을 통해서 DUPBS-EX는 작업의 크기가 커짐에 따라서 장애를 다른 방법에 비해 크게 축소시킴을 볼 수 있고 표 4에서 DUPBS-EX는 다른 스케줄링 기법과 비교하여 약 2배 정도 적은 중지의 발생을 보인다. 이점들이 바로 DUPBS-EX가 호스트 배제를 통해서 보다 적은 호스트를 가지고 100개로 증가된 작업들을 실행 시키게 됨에도 불구하고 다른 스케줄링 기법의 성능에 비해 개선된 결과를 나타낼 수 있는 점이다.

6.2.3 어플리케이션의 부 작업의 수가 200개일 때

부 작업의 수가 200개 일 때의 시뮬레이션 결과는 작업 수가 100개 일 때와 비슷하게 나타난다. 하지만 그림 12를 통해서 DUPBS-EX가 다른 스케줄링 기법의 성능을 개선하는 비율이 작업의 길이가 증가함에 따라서 작아지는 것이 목격된다. 이것은 결국 작업의 수와 길이가

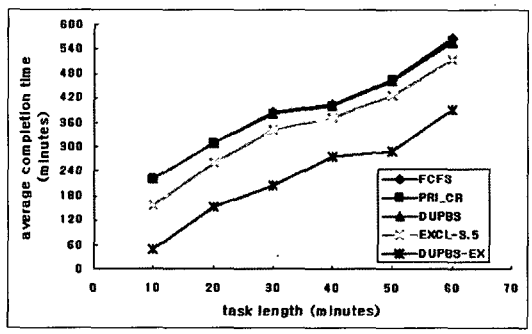

그림 11 부 작업 수가 100개인 어플리케이션의 평균 완료시간

표 3 부 작업 수가 100개일 때 어플리케이션이 완료 될 때까지 발생하는 평균 장애(failure)의 빈도 수

Length (minutes)	FCFS	PRI_CR	DUPBS	EXCL-S.5	DUPBS-EX
10	1.17	1.17	1.17	1.01	0.11
20	2.12	2.16	2.12	1.93	0.37
30	2.52	2.52	2.52	2.28	0.80
40	3.67	3.70	3.68	3.25	0.72
50	4.92	4.87	4.84	4.44	0.74
60	5.81	5.76	5.73	5.32	0.69

표 4 부 작업 수가 100개일 때 어플리케이션이 완료 될 때까지 발생하는 평균 중지(suspend)의 빈도 수

Length (minutes)	FCFS	PRI_CR	DUPBS	EXCL-S.5	DUPBS-EX
10	74.73	74.67	74.61	57.18	25.13
20	137.69	137.79	137.36	111.43	59.39
30	236.48	236.08	236.08	189.10	95.68
40	252.16	252.46	251.94	202.63	112.69
50	348.07	349.11	348.13	286.16	146.39
60	415.48	415.76	414.65	343.09	183.72

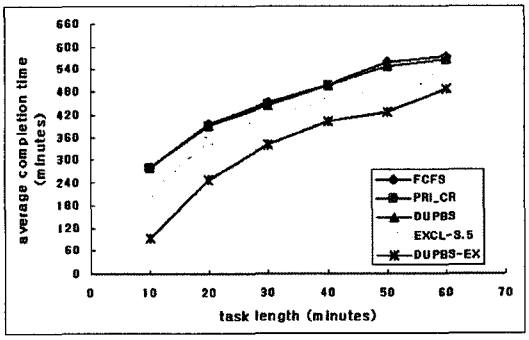

그림 12 부 작업 수가 200인 어플리케이션의 평균 완료 시간

표 5 부 작업 수가 200개일 때 어플리케이션이 완료 될 때까지 발생하는 평균 장애(failure)의 빈도 수

Length (minutes)	FCFS	PRI_CR	DUPBS	EXCL-S.5	DUPBS-EX
10	1.77	1.77	1.77	1.52	0.34
20	3.64	3.67	3.67	3.25	0.95
30	4.55	4.55	4.53	4.02	1.15
40	7.87	7.94	7.85	6.90	1.47
50	8.42	8.26	8.21	7.71	1.26
60	9.48	9.42	9.39	8.59	1.15

증가함에 따라서 신뢰성이 없는 호스트를 배제 시키는 것이 얻는 이점이 줄어드는 것을 의미한다. DUPBS-EX에서 중지가 자주 되는 호스트는 배제되지만 작업의

표 6 부 작업 수가 100개일 때 어플리케이션이 완료 될 때까지 발생하는 평균 중지(suspend)의 빈도 수

Length (minutes)	FCFS	PRI_CR	DUPBS	EXCL-S.5	DUPBS-EX
10	129.81	129.91	129.73	105.26	53.24
20	245.44	245.38	245.41	199.15	126.69
30	318.67	319.14	318.69	263.78	175.03
40	478.20	479.91	478.83	404.66	243.10
50	569.96	570.10	568.61	477.35	276.03
60	664.69	665.88	665.22	562.38	362.36

수와 길이가 증가함에 따라서 스케줄링에 이용되는 유휴 호스트의 수가 작업의 수와 길이에 비해 크게 작아지게 되고 결국 적은 수의 호스트에 많은 부하가 걸리면서 성능 개선 비율이 줄어들게 된다. 하지만 여전히 DUPBS-EX는 어플리케이션을 실행 할 때 중지와 장애의 발생 빈도가 적고 다른 스케줄링 방법에 비교해서 어플리케이션의 완료시간을 축소함을 확인 할 수 있다.

6.2.4 어플리케이션의 부 작업의 수가 400개일 때

그림 13에서 DUPBS-EX의 스케줄링 성능이 작업의 수가 400개이고 길이가 40분 이상일 경우 가장 나빠지는 것이 목격된다. 이 현상의 발생은 작업의 수가 200개일 때 다른 스케줄링에 비교한 성능 개선 비율이 작업의 길이가 증가함에 따라서 작아지는 현상과 같은 원인에 의해서 이다. 스케줄링에 이용되는 유휴 호스트에 비해서 실행해야 할 작업의 수와 양이 너무 많아지게 됨으로써 자원의 배제가 오히려 역효과를 일으켜서 성능 역전 현상을 보이게 된다. 자원 배제를 통하여 자주 그리고 오랜 시간 중지 되는 호스트는 작업을 실행 하는데 있어서 배제 되었지만 적은 수의 유휴 호스트를 통해서 실행 시간이 긴 많은 수의 작업을 실행하게 되어 작업의 처리는 병렬적으로 처리되기 보다는 특정 몇몇의 호스트에서 순차적으로 실행되어 부하를 가중시키는 결과가 나타나게 되고 결국 성능 저하를 초래하게 된다.

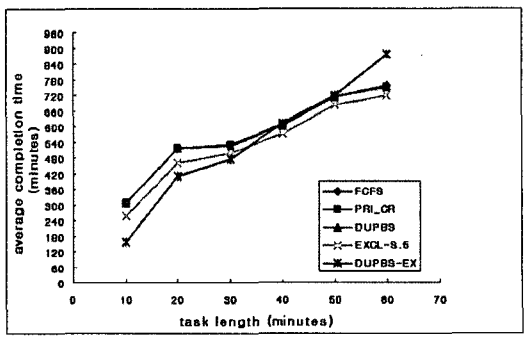

그림 13 부 작업 수가 400개인 어플리케이션의 평균 완료시간

표 7 부 작업 수가 400개일 때 어플리케이션이 완료 될 때까지 발생하는 평균 장애(failure)의 빈도 수

Length (minutes)	FCFS	PRI_CR	DUPBS	EXCL-S.5	DUPBS-EX
10	4.03	4.03	4.03	3.53	0.53
20	6.94	6.91	6.92	6.25	1.76
30	13.63	13.59	13.57	12.32	2.20
40	15.32	15.33	15.26	13.40	1.93
50	15.89	15.90	15.85	14.48	2.13
60	25.53	25.58	25.50	23.14	2.07

표 8 부 작업 수가 400개일 때 어플리케이션이 완료 될 때까지 발생하는 평균 중지(suspend)의 빈도 수

Length (minutes)	FCFS	PRI_CR	DUPBS	EXCL-S.5	DUPBS-EX
10	206.46	206.56	206.53	173.25	101.69
20	404.77	404.58	404.58	337.51	251.00
30	637.67	637.85	637.34	542.56	363.52
40	903.91	903.65	902.94	774.45	538.22
50	1088.51	1089.75	1088.84	941.93	615.75
60	1323.43	1325.53	1324.61	1130.25	754.91

하지만 장애 및 중지가 발생한 빈도가 여전히 DUPBS-EX의 경우 다른 스케줄링 방법에 비해 훨씬 작기 때문에 여전히 어플리케이션의 보다 안정적인 실행을 가능하게 하는 장점을 가지고 있다.

7. 결 론

본 논문에서는 데스크탑 그리드에서 자원의 사용경향을 고려하여 병렬 어플리케이션을 효율적으로 실행 할 수 있도록 지원하는 스케줄링 기법을 제시하였다. 데스크탑 그리드의 컴퓨팅 자원인 데스크탑의 비 신뢰적이고 예측 할 수 없는 특성은 확률적인 값으로 표현된 데스크탑 사용 경향성으로 반영되었고 이를 통해서 제시된 스케줄링 기법은 보다 효율적이고 신뢰성 있는 스케줄링이 가능하도록 지원한다. 제안된 스케줄링 기법인 DUPBS-EX는 시뮬레이션을 통해서 기존의 스케줄링의 성능을 개선하는 결과를 나타내었다. 이는 데스크탑의 사용 경향성을 통하여 어플리케이션 실행에 있어서 긴 지연시간을 야기 시키는 비 신뢰적인 호스트를 배제 킴으로써 다른 스케줄링에 비해서 작업의 중지와 장애의 발생을 축소한 결과이다.

하지만, DUPBS-EX는 데스크탑 그리드를 구성하는 호스트 수에 비해서 작업의 수가 비약적으로 커지게 되고 작업의 길이 또한 길어지게 되는 경우에는 자원 배제에 의해서 작업 수에 비해서 스케줄링에 사용될 수 호스트가 너무 많이 줄게 되어 성능 역전 현상이 스케줄링 성능이 저하되는 결과를 보여준다. 이점은 제안된 스케줄링 기법의 단점이 될 수 있지만 데스크탑 수에 비해서 어플리케이션의 작업 수가 극단적으로 큰 경우는 일반적이지 않고 이 경우에도 장애가 발생하는 빈도수는 줄어들게 되므로 신뢰성 있게 작업의 실행 할 수 있는 장점을 가지고 있다.

결론적으로 본 논문에서 제안된 스케줄링 기법은 데스크탑 자원의 사용 경향성을 고려함으로써 데스크탑 그리드가 효율적인 스케줄링을 통해 병렬 어플리케이션 실행 성능의 개선뿐만 아니라 신뢰성 있는 수행을 지원할 수 있는 가능성을 보여준다.

참 고 문 헌

[1] R.H Arpaci, A.D. Dusseau, A.M. Vahdat, L.T. Liu, T.E. Anderson, and D.A. Patterson, "The interaction of parallel and sequential workloads on a network of workstations," Proceedings of the 1995 ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems, pp. 267-278, May 1995.

[2] A. Acharya, G. Edjlali, and J. Saltz, "The Utility of Exploiting Idle Workstations for Parallel Computation," Proceedings of SIGMETRICS '97, pp. 225-236, May 1997.

[3] M. W. Mutka and M. Livny, "The Available Capacity of a Privately Owned Workstation Environment," Performance Evaluation, 12(4) pp. 269-284, July, 1991.

[4] W. T. Sullivan, D. Werthimer, S. Bowyer, J. Cobb, G. Gedye, and D. Anderson, "A new major SETI project based on Project Serendip data and 100,000 personal computers," Proceedings of the Fifth International Conference on Bioastronomy, 1997.

[5] M.R. Shirts and V.S. Pande, "Screen Savers of the World, Unite!," Science, 290:1903-1904, 2000.

[6] G. Fedak, C. Germain, V. Neri, and F. Cappello, "XtremWeb:A Generic Global Computing System," CCGRID'01, Brisbane, 2001.

[7] Derrick Kondo, Andrew A. Chien, and Henri Casanova, "Resource Management for Rapid Application Turnaround on Enterprise Desktop Grids," Proceedings of Supercomputing, SC2004, Pittsburgh, Pennsylvania, November 2004.

[8] Ryu, K.D. and J.K. Hollingsworth, "Exploiting Fine Grained Idle Periods in Networks of Workstations," IEEE Transactions on Parallel and Distributed Computing, 2000.

[9] P. Wyckoff, T. Johnson, and K. Jeong, "Finding Idle Periods on Networks of Workstations," Technical Report CS761, Dept. of Computer Science, New York University, March 1998.

[10] R. Wolski, "Forecasting network performance to support dynamic scheduling using the network

weather service," Proceedings of the High Performance Distributed Computing Conference, 1997.
[11] R. Wolski, N. Spring, and J. Hayes, "Predicting the CPU availability of time-shared unix systems on the computational grid," Proceedings of 8th IEEE Symposium on High Performance Distributed Computing, 1999.
[12] R. Wolski. N. Spring, and J. Hayes, "The network weather service: A distributed resource performance forecasting service for metacomputing," Future Generation Computer systems, 15(5-6):757-768, October 1999.
[13] D. Kondo, M. Taufer, C. Brooks, H. Casanova, and A. Chien, "Characterizing and Evaluating Desktop Grids: An Empirical Study," Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'04), April 2004.
[14] Patricio Domingues, Paulo Marques, Luis Silva, "Resource Usage of Windows Computer Laboratories," International Conference on Parallel Processing Workshops (ICPPW'05), 2005.
[15] John Brevik, Daniel Nurmi, and Rich Wolski, "Automatic methods for predicting machine availability in desktop Grid and peer-to-peer systems," CCGRID 2004: 190-199.

현 주 호

2004년 충남대학교 컴퓨터공학과 졸업(학사). 2006년 포항공과대학교 정보통신대학원 졸업(석사). 2006년~현재 (주) 삼성전자 DM연구소 연구원. 관심분야는 그리드 컴퓨팅, Peer-to-Peer 컴퓨팅

이 승 구

1985년 미국 Kansas 대학교 전기공학 졸업(학사). 1987년 미국 Michigan 대학교 전자전기 및 전산학과 졸업(석사) 1990년 미국 Michigan 대학교 전자전기 및 전산학과 졸업(박사). 1990년~1991년 미국 Delaware 대학교 전기공학과 조교수. 1991년~현재 포항공과대학교 전자전기공학과 조교수, 부교수, 교수. 관심분야는 이동 애드 혹 네트워크, 병렬 및 분산 처리(그리드/클러스터 포함), 실시간 컴퓨팅, 결함포용 컴퓨팅

김 상 철

1999년 경북대학교 전자전기공학부 졸업(학사). 2001년 포항공과대학교 전자전기공학과 졸업(석사). 2006년 포항공과대학교 전자전기공학과 졸업(박사). 2006년~현재 한국전자통신연구원 임베디드 S/W 연구단 선임 연구원. 관심분야는 그리드/클러스터 컴퓨팅, 센서 노드 운영체제

이 민 구

2000년 한양대학교 전자,전자통신,전파공학과군 졸업(학사). 2002년 포항공과대학교 전자전기공학과 졸업(석사). 2002년-현재 포항공과대학교 전자전기공학과 박사과정. 관심분야는 이동 애드 혹 네트워크, 실시간 시스템, 분산 처리