모바일 그리드에서 체크포인트 기반 작업 이주 기법

정대용[†] · 서태원[††] · 정광식[†††] · 유헌창[††††]

요 약

모바일 그리드에서 모바일 장치를 작업 처리에 이용하고자 하는 연구들이 많이 이루어지고는 있지만 모바일 장치는 무선 연결 및 배터리 용량에 관한 제약을 가지고 있으므로 모바일 장치를 이용한 작업 처리는 기존 그리드 환경에서의 작업 처리에 비해 신뢰성 및 효율성이 낮다. 따라서 모바일 장치가 가지는 제약 사항들을 고려한 작업 처리 방법이 필요하다. 이 논문에서는 모바일 그리드 환경에서 작업 이주 기법을 통해 모바일 장치를 이용하여 작업을 수행하는 방법을 제안한다. 즉, 모바일 장치에서 작업 수행 시 문제가 되는 상황들을 미리 예측하고 실행중인 작업을 체크포인팅하여, 모바일 장치에 문제가 발생했을 경우 체크포인팅 정보를 이용하여 다른 모바일 장치에게 작업을 이주할 수 있도록 한다. 이를 위해 프록시 서버에는 모바일 장치 관리자를 두고 모바일 장치에는 상태 관리자를 두며, 두 관리자를 통해 모바일 장치의 접속, 무선 신호 세기, 배터리 용량을 확인한다. 시뮬레이션 결과는 제안한 작업 이주 방법이 작업 수행시 효율성 및 신뢰성을 높일 수 있음을 보여준다.

주제어 : 모바일 그리드, 작업이주, 체크포인트, 무선세기, 배터리용량

Checkpoint-based Job Migration Technique in Mobile Grids

Dae-Yong Jung[†] · Tae-Weon Suh[††] · Kwang-Sik Chung[†††] · Heon-Chang Yu[††††]

ABSTRACT

There are many researches considering mobile devices as resources in mobile grids. However, the mobile device has some limitations: wireless connection and battery capacity. So, the grid operations using mobile devices have lower reliability and efficiency than those in fixed grid environments. In this paper, we propose a job migration scheme using mobile devices to overcome these limitations. The proposed job migration scheme predicts failure condition during execution and takes checkpoints. Then, if the failure occurs on mobile device during execution, the executing job can be migrated to other mobile device by checkpoint information. To perform the proposed migration scheme, we establish a mobile device manager on a proxy server and a status manager on a mobile device. Connection, wireless signal strength and battery capacity of mobile devices are identified through two managers. The simulation results show improvement of efficiency and reliability during execution.

Keywords : Mobile Grid, Job Migration, Checkpoint, Signaling Strength, Battery Capacity

† 정 회 원: 고려대학교 컴퓨터교육과 석박사통합과정
†† 종신회원: 고려대학교 컴퓨터교육과 교수
††† 정 회 원: 한국방송통신대학교 컴퓨터과학과 교수
†††† 종신회원: 고려대학교 컴퓨터교육과 교수(교신저자)
논문접수: 2009년 4월 06일, 심사완료: 2009년 6월 21일
* 본 논문은 2006년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2006-311-D00173).

1. 서 론

그리드 컴퓨팅은 지리적으로 분산된 컴퓨터 자원을 가지고 대량의 데이터 처리나 복잡한 문제를 해결하는 분산 시스템 환경이다. 계산 자원이 많이 필요하거나 대량의 데이터를 필요로 하는 기상예측, 생명과학, 우주과학 등 여러 분야에서 적용되고 있다[1]. 최근 노트북, PDA 등과 같은 모바일 장치의 보급으로 이 장치들을 그리드 자원으로 활용하려는 모바일 그리드에 대한 연구가 많이 이루어지고 있다. 그러나 모바일 그리드는 기존 그리드에 비해 모바일 장치의 낮은 성능, 배터리 및 무선 신호의 제약으로 작업을 완료시키지 못하는 문제점을 가진다[2]. 모바일 장치에서 작업 수행 중에 무선 접속이 끊어지거나 배터리가 부족하여 작업을 완료할 수 없는 현상 즉, 결함이 발생할 수 있다. 이런 문제를 해결하기 위해 이용되는 방법들 중 하나가 작업 이주(job migration) 기법이다. 작업 이주 기법은 하나의 모바일 장치에 결함이 발생하더라도 다른 모바일 장치에서 결함이 발생한 시점부터 작업을 이어서 수행할 수 있다. 이 방법은 작업을 처음부터 다시 수행하는 방법보다 효율적이다[3,4].

본 논문에서는 기존 그리드에 비해 많은 제약을 가지는 모바일 그리드에서의 작업 수행시 발생하는 문제점을 개선하기 위한 방법을 제안한다. 이를 위해 모바일 그리드 환경에서 모바일 장치의 배터리, 무선 신호세기, 작업 중 접속이 끊어지는 경우들에 대해 미리 예측하고 작업을 이주할 수 있도록 한다. 작업을 이주하기 위해 현재 작업 상태를 저장해야 하며 이를 위해 체크포인트 기법을 사용한다. 그 중에서도 사용자 시스템에 독립적인 파일형식으로 작업을 저장하는 사용자 정의 체크포인팅 기법을 사용한다[5]. 이는 일정한 간격으로 체크포인팅을 하는 시스템 레벨 체크포인팅 기법에 비해 사용자가 정한 상태에서 체크포인팅을 하기 때문에 모바일 장치에 더 적합한 방법이다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를, 3장에서는 본 논문에 적용되는 시스템 환경과 구성요소들에 대해 알아본다. 그리고 4장에서는 구성된 시스템 환경에서 작업을 체크포인팅하고 작업 이주를 하기 위한 조건에 대해 알아본다. 5장과 6장에서는 작업이주의 흐름과 알고리즘을 제안하고 실험 결과를 보여 준다. 7장에서는 결론과 향후 연구 과제에 대하여 언급한다.

2. 관련 연구

모바일 그리드 컴퓨팅은 무선 인터넷 기술과 모바일 장치의 성능 향상으로 발전되었다. 그리고 모바일 그리드 컴퓨팅은 노트북, PDA, 스마트폰, 등과 같은 다양한 자원을 이용할 수 있다. 그러나 모바일 장치를 자원으로 사용할 때 많은 문제점이 발생한다. 불안정한 배터리 용량, 무선 신호의 불안정, 모바일 장치의 낮은 성능과 같은 문제가 있다.

이 같은 문제를 해결하기 위해 작업 이주 기법은 매우 중요하다. 또한 작업 이주 기법은 모바일 자원에 대한 결함 포용을 위하여 많은 도움이 된다. 작업 이주 기법 연산은 수행중인 모바일 장치에 결함이 발생하면 다른 모바일 장치에서 재시작하지 않고 수행중인 작업을 이어서 수행한다. 그러나 수행중인 안정한 상태를 스토리지에 체크포인팅 시점과 작업 이주 시점에 통신비용을 증가시키는 원인이 된다. 따라서 작업 이주 기법은 재시작하는 방법보다 더 나은 성능을 보장하지는 않는다[3,4,6].

모바일 장치에서는 결함에 대비해 스토리지(storage)에 작업을 저장할 수 있으며 이때 주로 체크포인팅 기법을 사용한다. 체크포인팅 기법은 시스템 레벨 체크포인팅(system-level checkpointing)과 사용자 정의 체크포인팅(user-defined checkpointing)으로 구분된다. 시스템 레벨 체크포인팅은 이질적인 장치들 사이에서 체크포인트를 사용하기 어렵다[5]. 체크포인트 종류로는 관리자 프로세스인 코디네이터 프로세스를 두고 모든 프로세스를 관리하는 방법인 조정된 체크포인트(coordinated checkpoint)와 프로세서가 자율적으로 체크포인팅하는 독립적인 체크포인트(independent checkpoint)가 있다[7]. 모바일 환경에서는 체크포인팅 시점을 다른 모바일 장치에서 관

리하는 것보다 자신이 관리하는 것이 결함 포용에 적당하다.

[8]에서는 모바일 에이전트 환경에서 모바일 장치에 메일박스를 구성해서 체크포인팅하는 방법을 제안하였다. 이 논문에서 제안한 메일박스(mailbox)는 이전까지 작업한 모바일 에이전트의 작업을 체크포인팅하기 위해 사용된다. 이 메일박스를 통해 모바일 에이전트에 결함이 발생하면 메일박스에 저장되어 있던 체크포인팅 정보 즉, 이전 작업 정보를 가져와서 작업을 계속 수행한다. 그러나 이 방법은 체크포인팅 정보를 가진 모바일 장치에 결함이 발생했을 경우, 체크포인팅 정보가 메일박스와 함께 없어지는 문제점이 있다.

[9]에서는 모바일 그리드 환경에서 시스템 관리 브로커(SMB: System Management Broker)를 제안하였다. SMB에서 제공하는 주요 기능은 3가지로서, 서비스를 찾기 위한 Self Discovery, 서비스의 배치와 작업이주를 위한 Self Configuration, 결함을 발견과 상태를 관리하는 Self Healing이 있다. 이러한 프록시는 서비스 프록시(service proxy)와 클라이언트 프록시(client proxy)로 구성되어 서비스를 수행한다. 서비스 프록시는 service management component와 service instantiation component로 구성해서 SMB를 수행한다. 그리고 클라이언트 프록시는 클라이언트에 존재하며 결과를 전송하거나 실패된 서비스를 재요구하는 역할을 한다. 그러나 이 방법은 무선 신호 세기와 배터리 용량에 대해 고려하지 않는다.

본 논문에서는 작업 이주를 위해 각 프로세스에서 체크포인팅 시점을 결정하고, 체크포인팅한 작업을 외부 스토리지에 저장하는 방식을 사용한다.

모바일 장치는 AP 간을 이동할 때 핸드오버가 발생하는데, 핸드오버 구간에서 접속이 끊어지는 문제가 발생할 수 있다. 이 문제를 해결하기 위해서 본 논문에서는 핸드오버 기법 중 하나인 패스트 핸드오버(fast handover)[4] 기법을 이용하며 여기에 임계치를 추가로 설정한다. 이 임계치를 통해서 체크포인팅 시점을 결정할 수 있다.

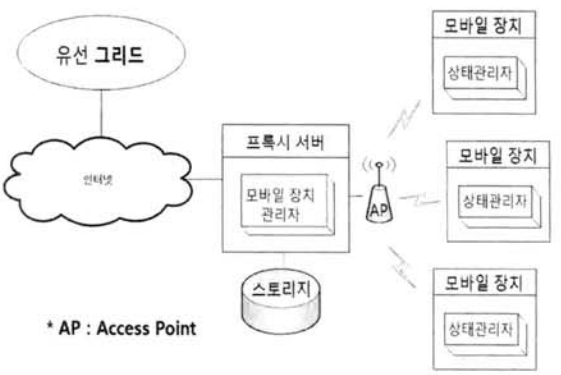
3. 시스템 구성

본 논문에서 제안하는 시스템 구성은 <그림 1>과 같다. 모바일 장치와 AP(Access Point) 사이에서 무선 신호세기와 모바일 장치의 배터리 용량을 관리하는 상태 관리자를 모바일 장치에 둔다. 그리고 모바일 장치들의 접속 상태와 모바일 장치 등록을 관리하는 모바일 장치 관리자를 프록시 서버에 둔다. 두 관리자를 통해서 모바일 장치의 현재 접속 상태, 모바일 장치의 무선 신호세기와 배터리 상태를 확인하여 결함이 발생할 수 있는 경우를 감지한다.

시스템의 전체적인 동작은 다음과 같다.

1) 모바일 장치에 결함이 발생할 수 있는 상황을 예측해서 결함이 발생하기 전에, 현재 모바일 장치의 작업을 프록시 서버의 스토리지에 체크포인팅한다.

2) 결함이 발생했을 경우 프록시 서버의 스토리지에 저장된 체크포인팅 정보를 가져와서 남은 작업 종료 시간 안에 작업을 처리할 수 있는 다른 모바일 장치를 선택하여 작업을 이주시킨다.



<그림 1> 시스템 구성

<그림 1>에서 두 관리자의 기능은 다음과 같다. 상태 관리자는 AP와 모바일 장치에서의 무선 신호세기, 모바일 장치의 배터리 용량을 확인하여 체크포인팅하는 시점을 결정한다.

모바일 장치 관리자는 모바일 장치의 접속 상태를 확인한다. 모바일 장치에 결함이 발생하면

스토리지에 저장된 체크포인팅 정보를 이용하여 남은 작업 종료 시간 안에 작업할 수 있는 다른 모바일 장치를 선택하여 작업을 이주시킨다.

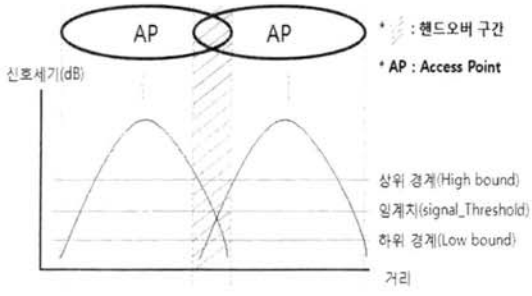
4. 체크포인팅 및 작업 이주 조건

이 장에서는 현재 수행 중이던 작업의 정보를 저장하기 위한 체크포인팅의 조건과 작업 이주를 위한 조건에 대해 살펴본다.

체크포인팅 및 작업이주를 위해 두 가지 조건 즉, 모바일 장치의 무선 신호세기와 배터리 용량을 정하고 이들에 대한 임계치를 설정하였다. 이 임계치는 체크포인팅하는 시점을 결정하게 되고, 신호세기나 배터리 용량이 임계치 이하로 내려가면 체크포인팅하게 된다. 또 신호세기나 배터리 용량이 임계치 이하의 설정 값보다 낮아지면 모바일 장치에 결함이 발생한 것으로 보고 체크포인팅한 작업을 다른 모바일 장치로 이주해서 계속 작업을 수행한다.

4.1 무선 신호세기

무선 신호세기는 모바일 장치에서 수신하는 AP로 부터의 신호세기를 말한다. 본 논문에서는 모바일 장치가 AP 간을 이동할 때 <그림 2>와 같이 패스트 핸드오버가 발생하는 상위 경계(high bound)와 하위 경계(low bound)에 체크포인팅을 하기 위한 임계치(signal_threshold)를 추가하였다. 즉, 무선 신호의 세기가 임계치보다 낮아지면 체크포인팅이 이루어져야 한다.



<그림 2> 무선 신호세기 조건

<그림 2>는 두 AP간의 무선 신호세기의 변화를 보여준다. 무선 신호세기가 상위경계보다 크면 모바일 장치와 프록시 서버 간에 통신이 원활함을 뜻하므로 해당 모바일 장치가 계속 작업을 수행할 수 있다. 무선 신호세기가 하위 경계보다 낮은 경우에는 AP를 벗어남을 뜻하므로 다른 모바일 장치로 작업을 이주시켜 작업을 수행한다. 임계치와 하위 경계는 모바일 장치가 체크포인팅할 시점과 작업 이주할 시점이 된다.

```
    // signal : AP로부터 받은 signal

1   signal_condition(signal) {
2     if(signal_threshold == signal) {
3       checkpoint();
4     }
5     else if(low bound > signal) {
6       request the job migration to
7       other mobile device;
8     }
9   }
```

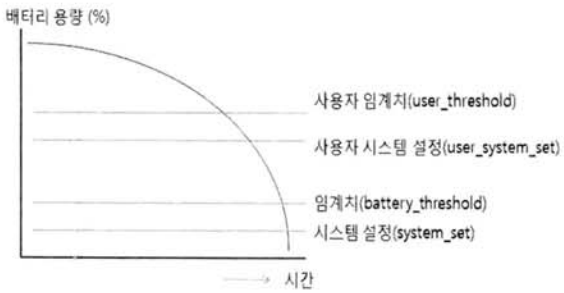
<그림 3> 무선 신호세기 조건을 이용한 알고리즘

<그림 3>은 무선 신호세기에 따라 체크포인팅하는 시점과 체크포인팅한 작업을 다른 모바일 장치에게 작업을 이주하는 시점을 보여준다. <그림 3>의 알고리즘은 AP로부터 모바일 장치에서 받는 신호세기에 대하여 표현하고 있다. 1줄은 무선 신호세기를 받아서 signal_condition함수에 입력한다. 2~4줄은 signal_condition함수에 받은 신호세기 값으로부터 임계치 값과 같을 경우에 체크포인팅을 수행한다. 5~8줄은 신호세기 값이 하위 경계보다 낮아질 경우 다른 모바일 장치에게 작업 이주를 요청하도록 한다.

4.2 배터리 용량

모바일 장치들은 배터리 용량에 제한을 가지고 있다. 이 문제를 해결하기 위해 본 논문에서는 <그림 4>와 같이 두 종류의 임계치를 이용한다. 즉, 사용자가 설정하는 사용자 시스템 설정과 운영체제에서 미리 설정된 시스템 설정이 그것이다. 사용자가 사용자 시스템 설정을 하지 않을 경우에는 모바일 장치의 운영체제에 미리 설정되어 있는 시스템 설정을 이용한다. 모바일 장치의

배터리 용량이 임계치(battery_threshold) 혹은 사용자 임계치(user_threshold)보다 낮을 경우 수행 중이던 작업을 체크포인팅하여 스토리지에 저장하고, 배터리 용량이 시스템 설정(system_set) 혹은 사용자 시스템 설정(user_system_set)보다 낮을 경우 다른 모바일 장치에게 작업을 이주한다.



<그림 4> 배터리 용량 조건

<그림 5>는 배터리 용량에 따라 체크포인팅하는 시점과 체크포인팅한 작업을 다른 모바일 장치에게 작업을 이주하는 시점을 보여준다.

```
// battery : battery capacity
// 사용자가 사용자 시스템 설정을 한 경우,
// system_set은 user_system_set으로,
// battery_threshold는 user_threshold로
// 변경

1  battery_condition(battery) {
2    if(battery_threshold == battery) {
3      checkpoint();
4    }
5    else if(system_set > battery) {
6      request the job migration to other
7      mobile device;
8    }
9  }
```

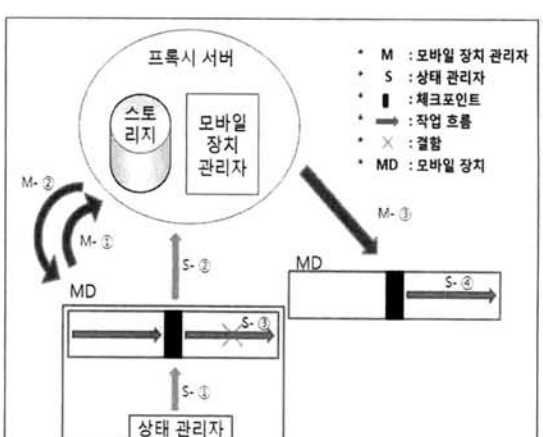
<그림 5> 배터리 용량 조건을 이용한 알고리즘

<그림 5>의 알고리즘은 모바일 장치에 대한 배터리 용량에 대하여 표현하고 있다. 1줄은 현재 모바일 상태의 배터리 용량 값을 가지고 battery_condition함수에 값을 입력한다. 입력된 배터리 용량 값으로부터 다음 두 가지 조건문에 대하여 수행한다. 2~4줄은 배터리 용량과 배터리 임계치가 같을 경우에 체크포인팅을 수행하도록 한다. 5~8줄은 배터리 용량이 시스템 설정보다 작아질 경우 다른 모바일 장치에게 작업 이주를 요청하도록 한다.

5. 작업 이주(job migration)

모바일 장치에서 다른 모바일 장치로 작업을 이주할 때 새로운 모바일 장치의 선택 기준은 남은 작업 종료 시간이 된다. 스토리지에서 체크포인팅 정보를 가져오고, 모바일 장치들에게 예상되는 남은 작업의 종료 시간을 계산하여 모바일 장치를 선택해서 작업을 이주시킨다. 총 작업 수행 시간은 다음과 같다.

총 작업 수행 시간 = 전송 시간
+ 작업 시작한 시간 + 체크포인팅 시간
+ 남은 수행 시간



모바일 장치 관리자(M)
① 모바일 장치를 등록시킴
② 등록된 모바일 장치의 접속 상태를 확인함
③ 남은 작업 종료 시간을 계산하여 작업을 이주할 다른 모바일 장치를 선택함

상태 관리자(S)
① 신호세기와 배터리 용량을 확인한 후 체크포인팅을 결정함
② 모바일 장치에서 체크포인팅하여 프록시 서버를 통하여 스토리지에 저장함
③ 결함 발생
④ 작업을 이주 받은 모바일 장치는 체크포인트 시점부터 작업을 수행함

<그림 6> 작업 이주 흐름

<그림 6>은 상태 관리자와 모바일 장치 관리

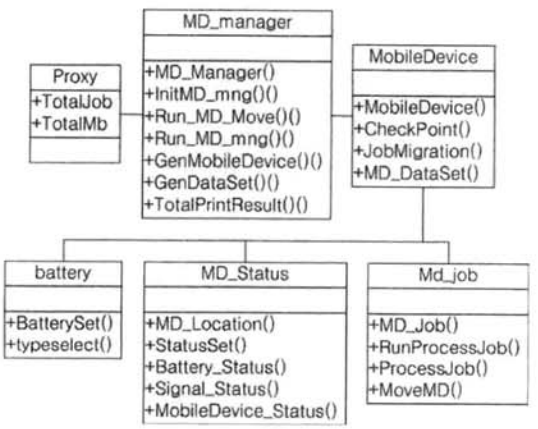
자로 구성된 시스템과 그들 간의 작업의 흐름을 보여 주고 있다. 모바일 장치 관리자는 등록된 모바일 장치의 접속 상태를 확인하고 상태 관리자는 모바일 장치의 무선 신호세기와 배터리 용량을 확인한다. 두 관리자를 통해 결함이 발생했을 경우, 이를 감지하고 작업을 이주시킬 수 있다.

```
1   EVENT status manager() {
2    while(true) {
3     signal_condition(signal);
4     battery_condition(battery);
5    }
6  }
7   EVENT mobile devices manager() {
8    while(true) {
9     connection_check_of_mobile_devices;
10    time_check_of_job_completion;
11      if(fault before checkpoint) {
12        restart the job on other mobile
13        device;
14     }
15     if(fault after checkpoint) {
16       job_migration();
17     }
18   }
19 }
20  job_migration() {
21    select other mobile device;
22    send the checkpoint from storage to
23    the selected mobile device;
24 }
25  checkpoint () {
26    take a checkpoint on the mobile
27    device;
28    send the checkpoint to the storage;
29 }
```

<그림 7> 작업 이주 알고리즘

<그림 7>은 작업 이주를 위한 알고리즘을 보여준다. 이 알고리즘에서는 크게 모바일 장치 관리자와 상태 관리자에 대한 이벤트에 관하여 서술하고 있다. 1~6줄은 <그림 3>과 <그림 5>의 무선 신호세기 조건과 배터리 용량 조건을 이용한 알고리즘을 수행하는 상태 관리자에 대한 이벤트를 표현한다. 7~19줄은 장치를 관리하는 모바일 장치 관리자에 대한 이벤트를 표현한다. 9줄은 모바일 장치의 접속 상태를 확인하고 10줄은 작업 수행 시간을 확인한다. 11~14줄은 체크포인트 이전에 결함이 발생했을 경우 다른 모바일 장치에서 작업을 재시작하여 작업을 계속 수행하게 하고, 15~17줄은 체크포인트 이후에 결함이 발생했을 경우에 20줄에 있는 작업 이주 함수를 호출하도록 한다. 20~24줄의 작업 이주 함수는 다른 모바일 장치를 선택하고 선택한 모바일 장치에게 스토리지로부터 저장된 체크포인팅 시점을 보내서 재수행하도록 한다. 25~29줄은 <그림 3>과 <그림 5>에서 체크포인팅 시점에 수행하는 작업에 대하여 표현하고 있다. 모바일 장치에서 체크포인팅을 수행하고 체크포인팅한 시점을 스토리지에 저장할 수 있도록 보내는 함수이다.

6. 성능 평가

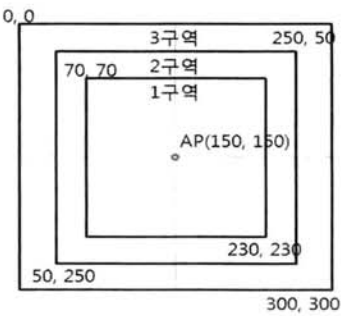


<그림 8> 클래스 구조

성능 평가를 위해 JAVA를 이용하여 작업 이주 기법을 시뮬레이션 하였다. <그림 8>은 이를 위해 구현한 클래스 구조를 보여준다. 이것은 <그림 1>의 시스템 구성을 기반으로 한다. 프록시에 모바일 장치 관리자(MD_manager)를 구성하고 이 모바일 장치 관리자를 통해서 모바일 장치(Mobile Device)를 호출한다. 모바일 장치는 상태 관리자(MD_Status)를 포함하며, 상태 관리자를 통해서 모바일 장치의 배터리 용량과 무선 신호세기를 검사한다.

<그림 9>는 시뮬레이션에서 이용한 신호세기의 구역을 보여준다. 각 구역은 작업을 수행할 수 있는 상태(1구역), 신호세기가 불안정한 상태

(2구역), 결함 발생한 상태(3구역)에 해당된다. 즉, 모바일 장치는 각 구역 안에 위치할 수 있다. 그리고 구역 경계를 이용하여 체크포인팅을 하는 시점(1-2구역 경계), 결함 발생 후 작업 이주하는 시점(2-3구역 경계)을 정하였다.



<그림 9> 신호세기 구역

실험 평가를 하기 위한 가정은 다음과 같다. 첫째, 작업 하나당 배터리 소모를 1mAh로 가정한다. 둘째, 모바일 장치의 초기 위치와 이동하는 위치는 임의로 구성한다.

작업량의 변화에 따라 다음의 각 경우에 대해 반복 수행하여 결과를 비교하였다.

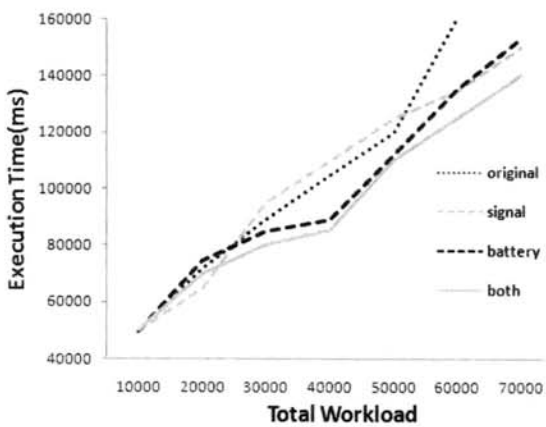
① 작업 이주 없이 다른 모바일 장치에서 작업을 재수행하는 경우(original)
② 신호세기 조건을 사용해서 다른 모바일 장치에 작업을 이주하는 경우(signal)
③ 배터리 용량 조건을 사용해서 다른 모바일 장치에 작업을 이주하는 경우(battery)
④ 신호세기, 배터리 용량 조건을 모두 사용하여 다른 모바일 장치에 작업을 이주하는 경우(both)

시뮬레이션을 하기 위하여 5종류의 장치에 대하여 배터리 용량과 연산 속도에 대하여 <표 1>로 정의하고 시뮬레이션을 하였다.

<표 1> 장치 사양

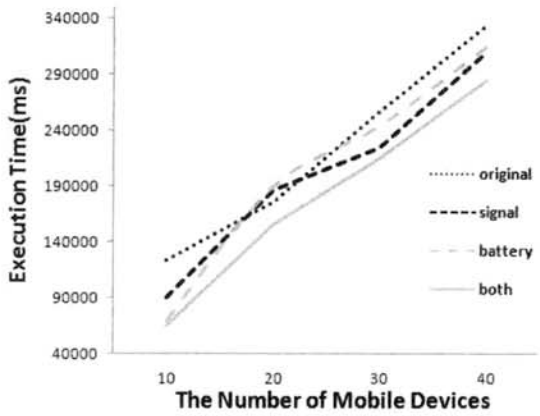
	배터리 용량 (mAh)	연산 속도 (Sleep)
Laptop	4,800	50, 300
Laptop(대용량)	6,400	50, 200
UMPC	2,400	100, 350
UMPC(대용량)	4,000	100, 450
PDA	2200	600, 700

모바일 장치에서 배터리 용량과 연산 속도에 관한 두 가지 조건에 대하여 시뮬레이션을 하기 위하여 설정하였다. 배터리 1cell당 800mAh로 정의되어 있는 것을 계산하여 mAh로 계산하여 나타냈다. 그리고 연산 속도는 CPU 속도에 비례하여 Sleep함수의 값을 입력하였다. 계산 속도는 낮을수록 빠른 연산 속도를 나타내고 있다. 그리고 현재 사용할 수 있는 배터리 용량은 70%정도로 정의해서 사용하였고 장치의 개수에 따라서 임의로 장치를 선택하도록 시뮬레이션을 구성하였다. <그림 10>은 장치의 개수를 동일하게 설정하고 동일한 작업의 작업 크기를 다르게 하여 실험한 결과이고, <그림 11>은 작업량을 동일하게 설정하고 모바일 장치의 개수를 다르게 하여 실험한 결과이다.



<그림 10> 전체 작업량에 따른 수행 시간

<그림 10>은 전체 작업흐름 수행 시간 결과를 보여준다. 이는 본 논문에서 제안한 모바일 장치의 신호세기와 배터리 용량을 동시에 고려한 작업 이주 기법이 나머지 경우들보다 전체 작업 수행시간이 줄어들었음을 보여준다. 실험결과, 작업량이 적은 경우에는 4가지 조건에서 비슷한 성능을 보이지만 작업량이 많이 증가하게 되면 ①의 경우에는 더 이상 작업을 수행할 수 없지만 ②, ③, ④의 경우에는 작업 이주를 통해서 계속 연산이 가능함을 알 수 있었다. 또한 ④의 경우는 ①의 경우에 비해 작업량이 많아질수록 더 효과적인 성능을 보였다.



<그림 11> 모바일 장치의 개수에 따른 수행 시간

<그림 11>은 모바일 장치의 개수에 따른 수행 시간 결과를 보여준다. 이 실험에서 ②, ③, ④의 경우에는 작업 이주기법을 사용하지 않은 ①의 경우보다 효율적인 성능을 보여준다. ④의 경우는 신호 세기와 배터리 용량을 모두 고려한 작업 이주 기법을 사용하므로 하나의 작업 이주 조건을 고려했을 경우보다 더 좋은 성능을 나타내고 있다. 다시 말해서 작업 이주의 결함인 무선 신호세기와 배터리 용량에 대한 결함에 대해 해결할 수 있게 되었다.

7. 결 론

이 논문에서는 모바일 그리드 환경에서의 제약사항을 해결하기 위한 체크포인트 기반 작업 이주 기법을 제시하였다. 무선 신호세기와 배터리 용량을 이용하여 모바일 장치의 결함을 예측하고, 현재 작업한 내용을 체크포인팅을 한 후, 결함이 발생했을 경우 체크포인팅 정보를 이용하여 다른 모바일 장치에서 작업을 이어서 수행하도록 한다. 이를 통해 모바일 장치의 제약 사항인 배터리 용량, 무선 신호세기에 대한 문제를 해결할 수 있다. 성능 평가를 통해서 본 논문에서 제시한 작업 이주 기법이 작업 수행에 있어 효율성 및 신뢰성을 향상시킬 수 있다는 것을 증명하였다. 그러나 여전히 불안정한 신호 연결, 작업 분배, 장치의 이질성, 보안 등 연구과제가 남아 있다. 이 같은 문제를 해결하기 위하여 작업 분배를 위한 효율적인 알고리즘의 개발이 필요할 것이다.

참 고 문 헌

[1] Foster and C. Kesselman, and S. Tueke. (2001). *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*, International Journal of Supercomputing Applications.

[2] G. Forman and J. Zahorjan. (1994). The Challenges of Mobile Computing, *IEEE Computer, 27*(4).

[3] Allen G, Angulo D, Foster I, Lanfermann G, Liu C, Radke T, Seidel E. (2001). The Cactus Worm: Experiments with dynamic resource discovery and allocation in a grid environment, *International Journal of High Performance Computing Applications, 15*, (4), 345-358.

[4] P.Roe and C.Szyperski. (1999). Transplanting in Gardens : Efficient Heterogeneous Task Migration for Fully Inverted Software Architectures, *Proceedings of the Fourth Autralasian Computer Architecture Conference*, January.

[5] Luis Moura Silva, Joao Gabriel Silva. (1998). System-Level versus User-Defined Checkpointing, *Symposium on Reliable Distributed Systems*, 68-74.

[6] Sriram Krishnan, Dennis Gannon. (2004). Checkpoint and Restart for Distributed Components in XCAT3, *In Proceedings of the fifth IEEE/ACM International Workshop on Grid Computing*, 281-288, Pittsburgh, Pennsylvania, November.

[7] E. Elnozahy and D. Johnson and Y. Wang. (1996). *A Survey of Rollback-Recovery Protocols in Message-Passing Systems*, Technical Report CMU-CS-96-181, Carnegie Mellon University, October.

[8] Jin Yang, Jiannong Cao, Weigang Wu. (2006). CIC: An Integrated Approach to Checkpointing in Mobile Agent Systems,

Proceedings of the Second International Conference on Semantics, Knowledge, and Grid (SKG'06).

[9] Michael Messig, Andrzej Goscinski. (2007). Autonomic system management in mobile grid environments, *Australian Computer Society*, 49-58.

[10] Hyon G. Kang and Chae Y. Lee. (2007). Fast Handover Based on Mobile IPv6 for Wireless LAN, *Korea Advanced Institute of Science and Technology*, February.

정 대 용

2007 한밭대학교 전자공학과 (공학사)
2007~현재 고려대학교 컴퓨터교육과 석·박사통합과정
관심분야: 그리드 컴퓨팅, 분산시스템, 결함포용 시스템
E-Mail: karat@comedu.korea.ac.kr

서 태 원

1993 고려대학교 공과대학 전기공학과(공학사)
1995 서울대학교 공과대학 전자공학과(공학석사)
2006 Georgia Institute of Technology (컴퓨터공학박사)
1995~1998 LG종합기술원 주임연구원
1998~2001 하이닉스반도체 선임연구원
2004 Intel Corp. Research Intern
2005~2006 Intel Corp. Research Intern
2007~2008 Intel Corp. Systems Engineer
2008~현재 고려대학교 컴퓨터교육과 조교수
관심분야: 임베디드 시스템, 컴퓨터 아키텍처
E-Mail: suhtw@korea.ac.kr

정 광 식

1995 고려대학교 컴퓨터학과 (이학석사)
2000 고려대학교 컴퓨터학과 (이학박사)
2005~현재 한국방송통신대학교 컴퓨터과학과 교수
관심분야: e-러닝, m-러닝, u-러닝, 분산처리시스템, 모바일 그리드컴퓨팅
E-Mail: kchung0825@knou.ac.kr

유 헌 창

1989 고려대학교 이과대학 전산과학과(이학사)
1991 고려대학교 대학원 전산과학과(이학석사)
1994 고려대학교 대학원 전산과학과(이학박사)
1998~현재 고려대학교 컴퓨터교육과 교수
2006~현재 한국컴퓨터교육학회 부회장
2004 조지아텍 방문교수
관심분야: 그리드 컴퓨팅, 분산시스템, 결함포용 시스템, u-learning
E-Mail: yuhc@comedu.korea.ac.kr