유비쿼터스 환경에서 그리드 컴퓨팅을 이용한 적응형 모바일 시스템

오제환[1],이은석[2]
성균관 대학교[1 2]
{hide7674[1],eslee[2]}@selab.skku.ac.kr

An Adaptive Mobile System Through Grid Computing in Ubiquitous Environment

Jehwan Oh[1], Eunseok Lee[2]
Sungkyunkwan University[1 2]

요 약

본 논문에서는 모바일 그리드 컴퓨팅 환경에서, 작업분배에 대한 보다 효율적인 계획을 스스로 결정하고, 이를 지속적으로 개선하는 'Self-Growing Engine' 기반 적응시스템을 제안한다. 최근, 모바일 컴퓨팅 환경에서의 다양한 제약사항을 극복하기 위해, 주변의 여러 컴퓨팅 단말기들의 유휴자원을 공유하여 하나의 작업을 처리하는 그리드 컴퓨팅 개념을 무선환경에 적용하려는 연구가 이슈로 등장하고 있다. 이때, 대부분의 기존 연구들은 그리드 컴퓨팅에 참여하는 단말기들의 리소스 상태만 고려하여 작업을 할당하는 방식을 취하고 있다. 따라서 상대적으로 작업효율이 낮은 단말기에서 작업이 할당되는 경우도 생기게 된다. 제안 시스템에서는 보다 효율적인 작업분배 결정을 위해, 다양한 사항을 고려하여 적절한 단말기를 선택하며, 각 작업수행 결과를 history 로 저장하며, 이후에 같은 요청이 있을 때 이를 분석하여 보다 적절한 단말기를 선택하도록 스스로 진화하는 특성을 갖는다. 우리는 제안 시스템의 평가를 위해, 데스크 탑에서 프로토타입을 구현하여 시뮬레이션을 수행하였으며, 그 결과를 통해 제안시스템의 효율성을 증명하였다.

Keyword : Adaptive System, Grid Computing, Mobile Computing

1. 서 론

최근 무선 네트워크 기술의 급격한 발전을 통해, handheld 단말기의 보급이 확산되고 있고, 그 이용도 폭발적으로 증가하고 있다. 이러한 handheld 단말기들은 지속적인 기술발전에 의해 컴퓨팅 파워가 계속 증가하고 있긴 하지만, 휴대성이 강조되면서 아직까지 제한된 성능을 가질 수밖에 없다. 따라서 handheld 단말기에서 실행되는 어플리케이션은 매우 가벼워야 하며, 복잡한 계산이 수행되는 대규모 어플리케이션은 동작이 어렵다는 제약사항을 가지고 있다. 때문에, 데스크탑 컴퓨터와 같은 품질의 서비스를 제공받기 원하는 사용자들의 욕구는 증가하고 있지만, 개발자들은 제한된 서비스만을 제공하는 어플리케이션을 개발할 수밖에 없다. 우리는 이러한 handheld 단말기의 제약사항을 극복하고, 다수의 작업을 가진 대규모 어플리케이션을 무선 환경에서도 운용하기 위해, 낮은 성능을 가진 여러 대의 컴퓨터를 이용하여 작업을 분산 처리하고, 이를 통해 하나의 컴퓨팅 단말기가 해결하기 힘든 큰 작업을 수행하는 그리드 컴퓨팅(grid computing)[1] 연구를 적용하였다.

"Grid computing 에서 분할된 작업을 누구에게 할당할 것인가?" 라는 문제는 매우 중요한 부분이다. 그러나 대부분의 기존 연구에서는 참여 단말기들의 리소스 상태만을 고려하여 작업을 할당하는 방식을 취하고 있다. 따라서 상대적으로 작업

효율이 낮은 단말기에게 작업이 할당되는 경우도 생기게 된다. 예를 들면, 리소스가 풍부하더라도 작업에 대한 코드를 가지고 있지 않은 경우, 리소스가 부족하긴 하지만 작업코드를 가지고 있는 단말기 의 작업속도가 훨씬 빠를 수 있다.

제안시스템은 이러한 문제를 해결하기 위해, 보다 다양한 현재 상황들(사용패턴, 여유리소스, 네트워크 속도, 작업파일 유무)을 고려하여 적절한 단말기를 선택하며, 이를 통해 보다 효율적인 작업분배가 가능하게 된다. 또한 제안시스템은 작업 처리 결과를 중앙에서 모니터 하여, 이를 히스토리에 저장하고, 이후에는 매번 계산을 수행하는 것이 아니라, 가장 결과가 좋았던 호스트의 상황과 현재 유사한 상황을 가진 호스트를 찾아 작업을 할당한다. 제안시스템은 이와 같이 시간이 지나며, 보다 적절한 호스트를 스스로 찾아 선택할 수 있는 자가 성장(Self-Growing) 특성을 갖고 있다.

본 논문에서는 제안시스템의 평가를 위해, 프로토타입을 구현하여 동작을 테스트하였으며, 실제 작업처리결과가 가장 빠른 호스트를 스스로 찾아 작업을 할당하는 결과를 통해, 시스템의 유효성을 증명하였다.

본 논문의 구성은 다음과 같다. 2 장에서는 관련연구로 기존의 그리드 컴퓨팅과 무선 환경의 제약을 극복하기 위한 적응시스템들을 소개하고, 3 장에서는 제안시스템의 전체적인 구조와 동작과정을 기술하였다. 또한 적절한단말기를 선택하기 위한, 보다 자세한 계산과정과, SGE 의 학습과정을 기술하였다. 4 장에서는 프로토타입을 구현하여 시스템의 평가를 수행하였으며, 결론과 향후 과제를 5 장에 기술하였다.

2. 관련연구

최근 모바일 컴퓨팅 환경의 여러 제약사항을 극복하려는 연구가 다양하게 진행되고 있다. 먼저, 제공되는 서비스의 품질이나 애플리케이션의 파라매터, 또는 자신의 구성요소를 변경하는 적응(Adaptation)관련 연구[7][8][9]가 활발히 진행되고 있다. 그러나 이는 제공되는 서비스의 질(QoS)에 초점을 맞추고 있다.

최근 유선 상에서 네트워크에 접속해 있는 수많은 컴퓨터들의 유휴 자원들을 이용하여 한대의 컴퓨터에서 해결할 수 없는 단일 문제를 해결하고자 하는 Grid computing[1]이라는 개념이 이슈로 등장하고 있다. 대표적으로 Globus[5]와 Condor[6] 을 들고 있다. 이러한 연구는 시스템을 실행하기 위해서는 해당 어플리케이션이 단말기에 설치되어 있어야 한다. 하지만, 이러한 어플리케이션을 실행하는데 많은 리소스 자원들을 요구하기 때문에 리소스가 풍부하지 못한 모바일 단말기는 그러한 어플리케이션을 실행하기가 어렵다.

Xiaohui Gu, et al.의 Adaptive Offloading System[4]은 모바일 단말기의 부족한 메모리의 제약을 해결하기 위해 요청 작업을 풍부한 메모리를 가지고 있는 Surrogate 에 분산시켜 처리하고 결과만을 모바일 단말기에 받을 수 있는 시스템을 제안하였다. 또 다른 연구로 프락시 기반의 모바일 그리드 컴퓨팅을 구현함으로써 모바일 환경에 있는 PDA 나 laptop 과 같은 모바일 단말기가 그리드 환경에 접근할 수 있게 하였다[11]. 그러나 이러한 대부분의 연구들은 작업을 능동적으로 할당할 하는 것은 매우 어려운 일이며, 단지 각 단말기의 여유 리소스만을 고려하고 있다.

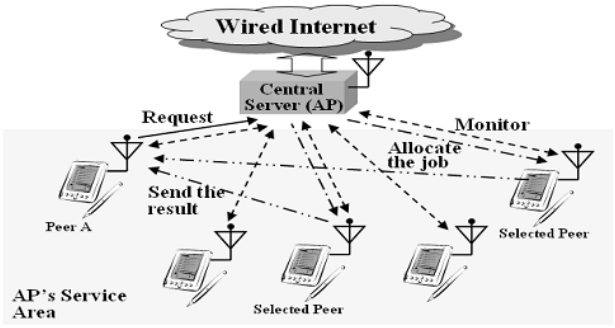
우리는 보다 효율적인 작업 분배를 위해, 여유 리소스뿐만 아니라 리소스를 사용하는 패턴, 소스의 유무 등을 반영하여 최적의 단말기를 선택한다. 이후, 자가 성장 엔진(Self-growing Engine)을 기반으로 스스로 진화하는 지능형 시스템을 제안한다. 다음 장에서는 이러한 개념들을 적용한 제안시스템에 대해 보다 자세히 설명한다.

바랍니다.

3. 제안시스템

3-1 시스템 프레임워크

본 논문에서는 모바일 그리드 컴퓨팅환경에서, 요청된 작업을 다른 단말기들에게 효율적으로 작업할당하기 위해서 작업 할당하는 계획을 지속적으로 개선하는 자가 성장 엔진(Self-growing engine)을 탑재한 지능형 적응 시스템을 제안하였다.

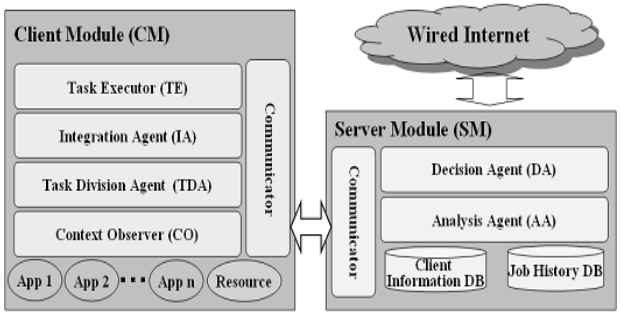


(그림 1) 제안 시스템의 전체 구조

제안 시스템은 효율적인 작업할당 계획을 결정하기 위해 주변의 단말기들이 제공할 수 있는 단말기의 리소스와 여러 반영 요소들을 점수화하여 작업을 할당하는데 반영하였다. 이렇게 선택된 단말기가 특정 서비스를 수행하고 그 결과를 job history DB 에 저장하면서, 추후 다른 단말기가 같은 서비스를 요청했을 때, 보다 적절한 단말기를 선택하도록 스스로 진화하는 특성을 갖는다.

3-2 시스템 구조

제안 시스템의 전체 시스템 구조는 (그림 2)와 같이 구성하였다. 제안 시스템은 크게 두 부분으로 구성되어 있다. 각 단말기에 설치되어 있는 Client Module(CM)과 서버에 설치되어있는 Server Module(SM)이 있다. SM 은 Access Point 가까이에 존재하거나 AP 내에 존재할 수 있다. 각 모듈을 구성하고 있는 세부적인 모듈에 대한 설명은 다음과 같다.

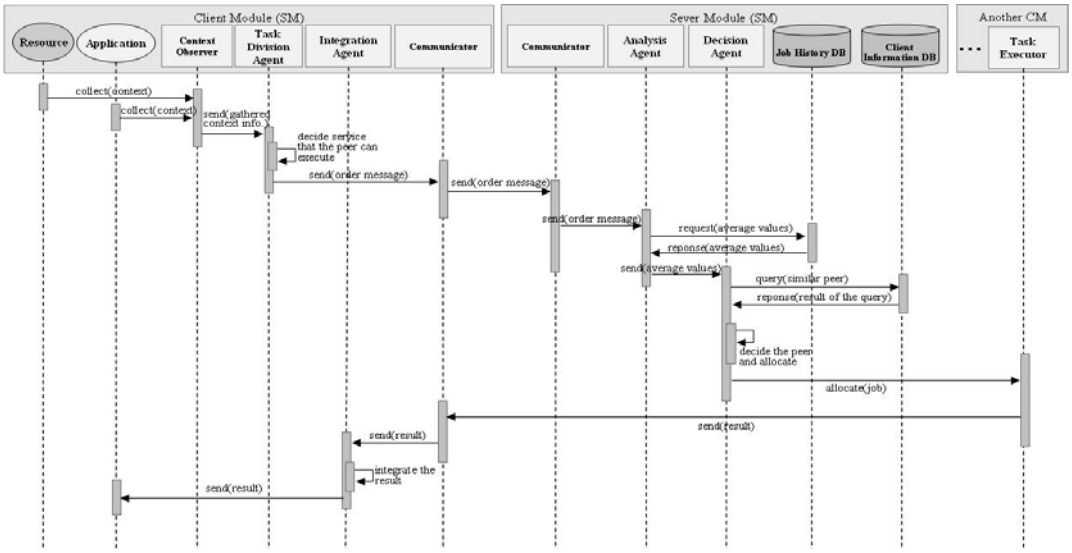


(그림 2) 제안 시스템을 구성하는 컴포넌트

- Context Observer(CO): 단말기의 고유한 성능과 특징에 대한 정보와 어플리케이션의 정보를 수집하여 System Profile 로 생성하고 이를 관리한다. 또한 동적으로 변화하는 주변 상황 정보(ex. CPU 사용량, 네트워크 최대속도, etc.)를 관찰하고 변화를 감시하는 역할을 수행한다.
- Task Division Agent(TDA): 사용자가 요청한 작업을 현재 리소스 상태에서 해줄 수 있는 작업과 없는 작업을 분할한다.
- Integration Agent(IA): 여러 단말기들로부터 얻은 부분적인 결과를 통합한다.
- Task Executor(TE): 외부에서 요청된 component 를 수행한다.
- Communicator(client): server 모듈과의 통신과단말기간의 상호작용을 위한 통신을 담당한다.
- Communicator(sever): sever 에 연결되어 있는 단말기들과의 상호작용을 하기 위한 통신을 담당한다.
- Analysis Agent(AA): Context Observer 로부터 수집된 정보를 기반으로 현재의 상황을 분석하고 추론하는 역할을 수행한다.
- Decision Agent(DA): Analysis Agent 가 분석한 현재의 상황과 Rule 을 기반으로 적절한 partition 계획을 결정한다. 이후, 수행결과를 피드백으로 받아 Rule 을 조절하는 학습기능을 갖고 있다.
- Client Information DB: 해당 Server 에 연결된 단말기들의 System Profile 을 저장한다.
- Job history DB: 각각의 서비스를 수행한 결과가 저장된다.

3-3 시스템 동작

제안 시스템의 전체 흐름은 (그림 3)과 같다.
먼저 CM 에 있는 CO 는 현재 자신의 리소스의 상태와 실행중인 어플리케이션의 정보를 지속적으로 수집하고, 수집된 정보들을 통해서 현재 단말기에서 유휴 자원을 얼마만큼 보유 하고 있는지에 대한 정보를 SM 에 있는 Client Information DB 에 저장된다.
사용자가 특정 어플리케이션을 실행하였을 때, TDA 는 현재 리소스의 상태에서 수행해 줄 수 있는 서비스를 결정하고, 수행할 수 없는 서비스에 대해서는 Communicator 를 통해서 서비스의 메타정보를 AA 에 보낸다. AA 에서는 Job history DB 에서 요청 서비스를 수행하기 하는데 최적의 상태가



(그림 3) 전체 시스템 동작의 Sequence Diagram

무엇인지 검색하고, 검색된 결과를 가지고 Client Information DB 에 최적의 상태를 만족하는 단말기가 있는지를 찾는다. 일치하는 단말기가 있다면, 할당된 정보를 communicator(client)에게 바로 전송하지만, 그렇지 않다면, DA 는 현재 유휴 자원을 가지고 있는 단말기들 중에서 해당 작업을 처리하는데 가장 효율적인 단말기를 찾는다(3.5 장). 이렇게 결정된 allocation list 는 communicator(client)게 전송된다. communicator(client)는 allocation list 에 따라 해당 단말기에게 작업을 요청하고, 결과를 받는다. 이때, 하나의 서비스를 수행하데 걸리는 시간과 작업을 수행한 단말기에 대한 정보는 SM 에 보내고, SM 은 이러한 정보를 Job history DB 에 저장하게 된다. IA 는 각각의 단말기에서 수행된 partial 결과를 통합하여, 통합된 결과를 어플리케이션에게 보내다.

3-4 작업할당

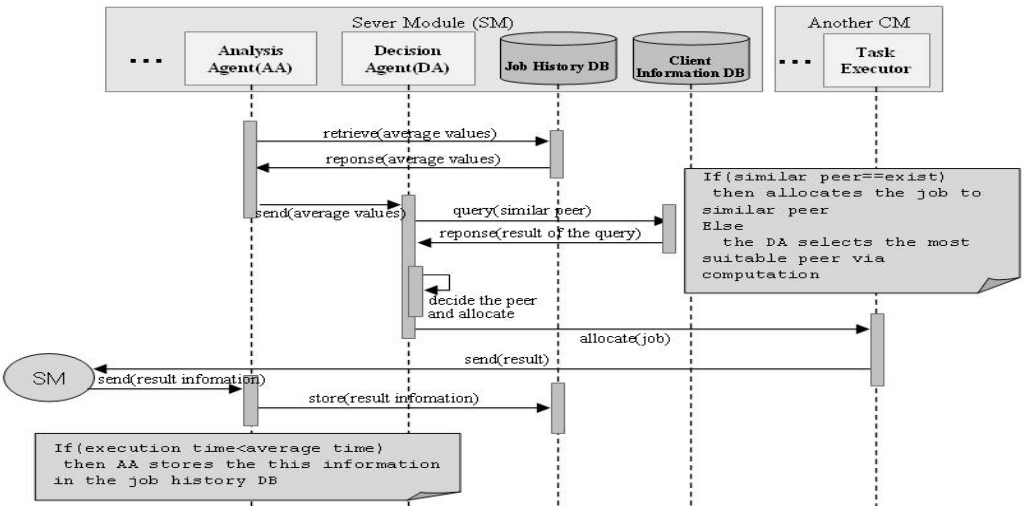
모바일 그리드 컴퓨팅 환경에서 client 가 요청한 서비스를 주변의 단말기에게 할당할 때, Server 가 리소스의 상태만을 고려해 할당경우, 상대적으로 작업효율이 낮은 단말기에게 작업이 할당될 수도 있다. 예를 들어, 리소스가 풍부하더라도 실행코드가 없다면, 리소스가 부족하더라도 실행코드를 가지고 있는 단말기가 작업을 수행하고 결과를 얻는데 까지 걸리는 시간이 더 빠를 수 도 있다. 제안 시스템은 이러한 문제를 해결하기 위해, 다음의 내용을 고려하여 적절한 단말기를 선택하고 효율적인 작업분배를 결정한다.

- 각 참여 단말기들의 현재 유휴 리소스(CPU, RAM, battery, …)
- 각 참여 단말기들의 네트워크속도
- 각 참여 단말기들의 사용패턴을 분석하여, 현재 시간에서 리소스가 변동할 수 있는 확률
- 각 참여 단말기들의 작업에 대한 코드가 설치되어 있는지 여부

각각의 정보들은 CM 에서 수집되어 SM 에 있는 Client Information DB 주기적으로 저장한다. 각각의 요소들은 (표 1)의 식들을 이용하여 상대적인 점수로 계산된다. SM 은 가장 높은 점수를 얻은 단말기를 선택하고, 해당 작업을 할당 한다.

컨텍스트 타입	계 산 식	예제	점수
유휴 자원	$1-\frac{CPU_{available\ Rate}\times CPU_{clock}}{10000}=CPU_{level}$	$1-\frac{20\times 400}{10000}=0.2$	2
	$\frac{RAM_{freeSpace}}{RAM_{size}}=RAM_{level}$	$\frac{20}{64}=0.312$	4
	:	:	:
리소스가 변화할 확률(w)	$1-\frac{\sqrt{(x_1-m)^2+(x_2-m)^2+\cdots+(x_n-m)^2}}{10}$	$1-\frac{2.49}{10}=0.75$	8
:	:	:	:

(표 1) 각 요소들을 계산하는 식



(그림 4) 학습 단계를 보여주는 Sequence Diagram

$$CPU_{level} \cdot w_1 + RAM_{level} \cdot w_2 + \cdots = Final\,Score \quad (1)$$

```
(REQUEST
  :sender(agent-identifier :name CM10HIDE:1099/)
  :receiver (set(agent-identifier :name SM@selab:1099/))
  :context "(action translator)and
           (action image_compression)
      …
)
```

```
(INFORM
  :sender(agent-identifier :name CM10HIDE:1099/)
  :receiver (set(agent-identifier :name SM@selab:1099/))
  :context "(service_name(translator)),(entrusted_device(tester76)),
           (execution_time(3.5)),
      …
)
```

(그림 5) ACL 메시지의 예제

3-5 자가 성장 엔진을 통한 학습단계

이전 장에서 CM 이 SM 에게 서비스를 요청했을 때, 특정 단말기를 할당 받는 과정을 설명하였다. CM 이 특정 단말기를 할당 받아 서비스를 완료하게 되면, CM 은 서비스를 요청한 시간부터 결과를 받는데 까지 걸린 소요시간과 서비스를 수행한 단말기에 대한 정보를 AA 통보한다. AA 는 해당 서비스를 수행하는데 소요된 시간과 job history DB 에 저장된 평균 소요시간을 비교한다. 만약, 평균 소요시간보다 더 짧은 시간 내에 수행 되어졌다면, AA 는 서비스를 수행한 단말기의 정보와 소요시간을 job history DB 에 저장한다. 이러한 동작이 반복적으로 수행 할수록, 특정 서비스를 수행하는데 필요한 최적의 단말기 상태에 가까워진다. Job history DB 를 통해서 얻은 특정 서비스를 수행하는데 필요한 최적의 단말기 상태에 대한 정보를 기반으로 Client Information DB 에서 가장 유사한 단말기를 선택하도록 스스로 진화한다.

Job history DB 의 데이터 수는 지속적으로 증가하는 것을 막기 위해서 데이터의 수는 임의로 제한하였다. Job history DB 에서 각 서비스에 대한 history table 의 데이터 개수를 몇 개로 잡는 것이 가장 효율적인가 하는 문제는 향후 과제로 삼겠다.

4. 시스템구현 및 평가

4.1 시스템 구현

본 시스템의 평가를 위해 개발된 프로토타입의 각 모듈은 JAVA 를 이용하여 개발되었으며, Client 단말기는 WindowsCE 를 운영체제로 사용하였다. 각 Client 단말기에서 JAVA 를 실행하기 위해 WebSphere Studio Device Developer 에 포함되어 있는 J9[10] JVM 을 사용하였다. Client 단말기로 사용된 PDA 는 Intel(R) PXA255 프로세서와 128Mb 의 메모리를 가지고 있는 hp iPAQ h5550 을 사용하였다.

단말 이름	상 태	실행 시간	날 짜
tester 34	7, 5, 4, 1, 4, 8	3.23	20050704
tester 21	4, 4, 5, 2, 3, 4	3.74	20050621
tester 28	3, 9, 4, 1, 4, 7	3.59	20050520
:	:	:	:
평균 값	4.7, 7.3, 6.1, 1.4, 7.2, 3.4	3.82	

(표 2) Job history DB 에 저장되는 데이터의 예제

시스템의 효율성을 보이기 위해서 우리는 기존에 구현된 “Remote Video Medical Treatment” 어플리케이션을 제안 시스템에 적용시켰다. 이 어플리케이션이 동작하는 과정에서 적응(adaptation)이 필요로 하는 시점에 각 adaptation service 를 할당 하게 된다. 할당되는 적응 서비스(adaptation service)로는 ‘Translator’, ‘Image Converter’, ‘Video Converter’이 있다.

4.2 시스템 평가

우리는 두 가지의 실험을 통해서 제안 시스템의 효율성을 확인하였다.

첫 번째 실험은 특정 서비스를 수행해주는 단말기 결정하는데 리소스의 상태만 반영하여 결정한 단말기가 수행하는데 걸리는 시간과 우리가 제안하는 6 가지의 요소들을 반영하여 결정한 단말기가 수행하는데 걸리는 시간을 비교하였다. 두 번째 실험은 제안 시스템을 적용하여 특정 서비스를 반복하여 수행하였다. 이 실험을 통하여 서비스를 수행함에 따라 얼마만큼 효과적인 단말기를 선택하는지를 알아보았다.

첫 째, 우리는 Client Information DB 에 임의로 10 개의 단말기 정보를 저장하였다. 그리고 특정 단말기가 translator 서비스를 요청하고, SM 이 Client Information DB 에서 translator 서비스를 실행해 줄 단말기를 선택하는 과정에서 각 단말기들의 리소스만을 반영하여 결정한 단말기 가 서비스를 수행하는데 걸리는 소요시간과 우리가 제안하는 6 가지의 요소들을 반영하여 결정한 단말기가 서비스를 수행하는데 걸리는 소요시간을 비교하였다.

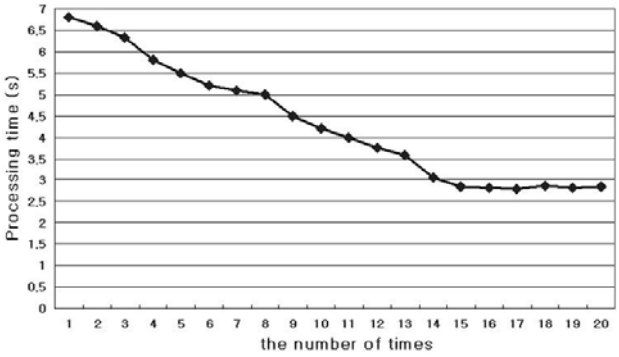
	선택된 단말의 상태	처리 시간 (s)
기존 방법	6, 8, 9	5.6
제안 방법	4, 7, 8, 1, 8, 9	4.3

(표 3) 처리 시간 비교

(표 3)은 실험에 대한 결과를 보여준다. 실험은 동일한 조건에서 수행되었다. 결정하는데 반영되는 요소만 변경하였다. 제안 시스템이 선택한 단말기는 리소스만 반영하여 선택한 단말기보다 리소스의 상태는 낮았다. 하지만, 리소스의 유무와 네트워크 속도 등에서 뛰어난 단말기를 선택하였기 때문에 수행시간은 더 짧았다.

둘 째, 우리는 translator 서비스를 시뮬레이션 PDA 를 통해서 상태가 각기 다른 임의의 조건으로 하여 100 번을 수행하여, 이를 통해 평균 시간과 각 반영 요소들의 임의의 평균값을 얻을 얻었다. 이 평균 시간과 평균값을 Job History DB 의 초기 값으로 입력하였다. 다음으로는 특정 단말기가 SM 에게 translator 서비스를 반복하여 요청하였다.

(그림 6)은 translator 서비스를 반복하여 실행함에 따라 실행하는데 걸리는 소요시간을 보여 준다. 처음 실행했을 때에는 임의로 100 번을 수행했을 때 걸리는 평균 소요시간과 비슷한 소요시간이 걸렸으나 계속 반복적으로 수행함에 따라 지속적으로 소요시간이 감소하였다. 하지만, 15 번째 수행했을 때부터는 소요시간에 변화가 거의 일어나지 않았다. 이것은 SGE 가 translator 서비스를 실행하는데 최적의 실행 조건을 찾은 것을 의미하며, 이 조건에 가장 유사한 단말기를 검색하여 선택하기 때문이다.



(그림 6) 'translator service'를 반복 실행을 통한 처리 시간

5. 결론 및 향후과제

본 연구에서는 모바일 그리드 컴퓨팅 개념을 적용시켜 주변의 유휴 자원을 이용한 가상의 컴퓨터를 구성하여 단말기의 제약을 해결할 수 있는 프레임워크를 제안하였다. 제안 시스템은 자가 성장 엔진(Self Growing Engine)을 기반으로 효율적으로 작업 분해하도록 하였다. 제안 시스템은 두 가지

실험을 통하여 효율성을 확인하였다. 제안시스템의 다양한 특성은 제한된 무선컴퓨팅 환경의 여러 문제점을 해소하고, 보다 편리한 무선 컴퓨팅 환경을 제공해줄 것으로 기대된다.

[참고문헌]

[1] Ian Foster, Carl Kesselman, Jeffrey M. Nick, Steven Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration", OPEN Grid Service Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure WG, Global Grid Forum, Jun 2002.

[2] Alvin T.S. Chan, Siu-Nam Chuang, "MobiPADS: A Reflective Middleware for Context-Aware Mobile Computing", IEEE Transaction on Software Engineering. Vol. 29, No.12 pp.1072-1085, Dec.2003

[3] httpPaolo Bellavista, Antonio Corradi, Rebecca Montanari, Cesare Stefanelli, "Context-aware middleware for resource management in the wireless Internet", Software Engineering, IEEE Transactions on Vol. 29, Issue 12, pp. 1086-1099, Dec. 2003

[4] Xiaohui Gu, Klara Nahrstedt, Alan Messer, Ira Greenberg, Dejan Milojicic, "Adaptive offloading for pervasive computing", Pervasive Computing, IEEE Vol 3, Issue 3, pp.66-73, Jul-Sep. 2004

[5] http://www.globus.org/

[6] http://www.cs.wisc.edu/condor/

[7] A. Friday, N. Davies, G.S. Blair and K.W.J. Cheverst, "Developing Adaptive Applications: The MOST Experience", Journal of Integrated Computer-Aided Engineering, 6(2), pp.143-157, 1999

[8] Wai Yip Lum, Francis C. M. Lau, "User-Centric Content Negotiation for Effective Adaptation Service in Mobile Computing", IEEE Transaction on Software Engineering. Vol. 29, No.12 pp.1100-1111, Dec.2003

[9] Vahe Poladian, João Pedro Sousa, David Garlan, Mary Shaw, "Dynamic Configuration of Resource-Aware Services", 26th International Conference on Software Engineering (ICSE'04), pp.604-613, May.2004

[10] http://www-306.ibm.com/software/websphere/

[11] Junseok Hwang; Aravamudham, P., "Middleware services for P2P computing in wireless grid networks", Internet Computing, IEEE, Vol 8, Issue 4, pp.40-46, July-Aug. 2004