http://dx.doi.org/10.3745/KIPSTA.2012.19A.3.161

클라우드 환경의 Thin-Client 모바일을 위한 동적 자원 분배 기술

이 준 형[†] · 허 의 남[††]

요 약

클라우드 컴퓨팅의 폭발적 성장으로 IT를 활용하는 다양한 분야에 걸쳐 클라우드 기반의 시스템을 이용한 연구가 활발하게 진행되어 왔다. 특히, 클라우드 시스템을 이용하여 가상 머신 기반의 데스크탑 환경(DaaS, Desktop as a Service) 및 가상 모바일 환경을 구축하여 씬 클라이언트 형태로 어플리케이션 서비스를 이용하는 연구가 활발하게 진행 중에 있다. 본 논문은 씬 클라이언트 모바일 단말에서 모바일 어플리케이션 수행에 필요한 클라우드 시스템과, 자원 분배 시스템인 DRAMMA(Dynamic Resource Allocation Manager for Mobile Application)를 제안하고 있다. 또한 성능평가를 통해 DRAMMA가 기존의 클라우드 환경의 자원 분배 알고리즘과 비교하여 클라우드 시스템의 활용성을 높였고, 더 적은 가상머신의 이동, 자원 할당에 나타나는 오류를 줄여 보다 효율적인 서비스가 가능함을 확인 할 수 있었다.

키워드 : 클라우드 컴퓨팅, 씬 클라이언트, 가상 머신, 자원 할당, 자원 분배

An Efficient Dynamic Resource Allocation Scheme for Thin-Client Mobile in Cloud Environment

Jun-Hyung Lee[†] · Eui-Nam Huh[††]

ABSTRACT

The study of Cloud based system is emerging to become the core technology in IT field due to the tremendous growth of Cloud Computing. Researches to deliver applications to Thin-Client based mobile virtual machine and Desktop as a Service(DaaS) using Cloud Computing are conducted actively. In this paper, we propose a Cloud system to run the mobile application in the mobile Thin-Client device and resource allocation mechanism Dynamic Resource Allocation Manager for Mobile Application(DRAMMA). Thus, through performance check, we show DRAMMA has improved the utilization of Cloud system, less migration of virtual machines and decreased the error rate of resource allocation. Also our proposed system delivers service more efficiently than the previous resource allocation algorithm.

Keywords : Cloud Computing, Thin-Client, Virtual Machine, Resource Allocation, Resource Distribution

1. 서 론

오늘날 IT의 화두인 클라우드 컴퓨팅은 최근 몇 년간 폭발적 성장과 함께 IT를 활용하는 다양한 분야에 걸쳐 많은 부분에 있어 큰 변화를 가져왔고, 다양한 분야에 걸쳐 활발한 연구가 진행 중에 있다.

클라우드 컴퓨팅은 가상화(Virtualization)를 기반으로 일정 수준의 비용을 지불하고 사용하는 형태를 일컫는 의미로 본래 1960년대 미국의 저명한 컴퓨터 학자 존 매카시가 제시한 "컴퓨팅 환경은 공공 시설을 쓰는 것과도 같을 것"[1] 에서 유래하였다. 클라우드 컴퓨팅은 무형의 형태로서 컴퓨터 하드웨어를 하이퍼바이져 (Hypervisor)라는 하드웨어와 운영체제 사이의 얇은 소프트웨어 계층을 통해 가상화시키고 이에 논리적인 가상 하드웨어(Virtual Machine)를 하나의 물리적 시스템으로부터 다수의 논리적 시스템을 만드는 개념이다[2].

클라우드 컴퓨팅을 통해 사용자는 필요한 만큼의 자원을 요청하고 이를 가상 하드웨어로 만들어 인터넷을 통해 이를 사용할 수 있는데, 각 사용자마다 요구하는 성능이 각기 다르므로 전체 시스템으로부터 일정양의 자원을 할당하고 남는 유휴자원에 대하여 재사용이 가능해 진다.

※ 이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단-차세대 정보컴퓨팅기술개발사업의 지원을 받아 수행된 연구임(No.2011-0020515).
† 정 회 원 : 한국과학기술정보연구원 슈퍼컴퓨팅센터 차세대연구 환경개발실 연구원
†† 종신회원 : 경희대학교 컴퓨터공학과 교수(교신저자)
논문접수 : 2012년 5월 3일
수 정 일 : 1차 2012년 5월 16일
심사완료 : 2012년 5월 17일

본 논문은 클라우드 컴퓨팅을 바탕으로 사용자에게 모바일 어플리케이션과 그에 필요한 연산 자원에 대하여 동적으로 할당하는 것을 다루고 있다. 이를 통해 전체 컴퓨터 시스템들의 활용성을 높이고, 잉여 자원을 낮추며, 자원의 동적 할당을 통해 효율성을 높이는데 그 목적을 두고 있다. 특히, 본 논문은 최근 급속도로 보급되고 있는 스마트폰의 모바일 어플리케이션 서비스 제공과 동적 자원 할당에 필요한 알고리즘 제안한다. 단순한 설치형 로컬 어플리케이션이 아닌 인터넷을 통해 클라우드 기반의 가상 모바일 하드웨어 장치에서 어플리케이션을 수행하고, 이를 Thin-Client 형태로 제공 하는 것을 고려하였다[3][4][5].

본 논문의 구성은 다음과 같다. 우선 2장에서는 클라우드 컴퓨팅과 Thin-Client 등 관련기술에 대하여 살펴본다. 또한 제안하는 시스템에서 사용되는 자원 할당 알고리즘과 관련된 연구들에 대하여 살펴본다. 3장에서는 본 논문에서 제안하는 클라우드 기반의 Thin-Client 모바일과 DRAMMA (Dynamic Resource Allcation Manager for Mobile Application)의 설계 및 구현에 관하여 알아보고, 4장에서는 제안하는 DRAMMA에서 사용하는 자원 할당 알고리즘에 대하여 살펴본다. 이를 바탕으로 5장에서는 기존 알고리즘과의 성능평가를 통해 본 논문에서 제안하는 알고리즘의 효율성을 검증할 것이며, 6장에서는 결론 및 향후 연구 방향을 제시하고자 한다.

2. 관련 연구

2.1 클라우드 컴퓨팅과 Thin-Client 기술

클라우드 컴퓨팅은 앞서 서론에서 살펴본바와 같이 인터넷을 기반으로 다양한 소프트웨어와 하드웨어를 제공하는 기술이다. 특히, 하나의 특정 어플리케이션 같은 서비스뿐 아니라 개인 PC 환경과 같은 데스크탑을 클라우드 상에 구축하고 이를 네트워크를 통해 접속하여 사용하는 데스크톱 가상화(VDI, Virtual Desktop Infrastructure) 서비스가 유행하고 있는데 이는 Thin-Client 형태의 발전 형태라 할 수 있다.

Thin-Client는 과거 메인프레임이 주를 이루던 시절 터미널을 통해 메인프레임에 접속하여 다양한 업무를 처리하던 것을 시작으로 Telnet과 SSH 같은 콘솔 기반의 환경 뿐 아니라 RDP(Remote Desktop Protocol)[8], VNC(Remote Frame Buffer Protocol)[9], PCoIP[10] 등과 같은 그래픽 환경으로 발전해 오고 있으며, 최근 VMware 사의 vDesktop과 Citrix 사의 Xen Desktop 에 이르기까지 다양한 제품이 나오고 있으며, 이는 모두 클라우드 상의 자원에 접속하여 이용하는 형태라 할 수 있다.

2.2 Sandpiper 시스템

Sandpiper[6] 시스템은 Timothy Wood 이 제안한 시스템으로, 클라우드 상의 가상머신들에 대한 자원 활용 정보 모니터링 값을 바탕으로 자원의 결핍(Hotspot)을 판별하여 물리적 호스트머신과 가상머신간의 배치를 다루고 있다.

Sandpiper 시스템은 자원의 결핍이 발생하였을 때 모니터링된 값을 기반으로 자원의 배치를 결정하게 되는데 실시간 가상머신 이전(Live Migration) 혹은 교체(Swap)를 이용하고 있다.

$$Volume = \frac{1}{1-cpu} \times \frac{1}{1-mem} \times \frac{1}{1-net}$$

특히 Sandpiper 시스템은 실시간 모니터링 되는 값을 기반으로 “Volume” 이라는 다차원 값을 정의 하고 있는데 이를 바탕으로 물리적 하드웨어와 가상머신간의 배치하고 있다.

생성된 “Volume” 을 바탕으로 가상머신 중 부하가 가장 심한 “Volume”과 가장적은 “Volume”간의 교체를 통하여 자원을 분산하는 방안으로 다 차원의 자원 활용 정보를 바탕으로 효율적으로 분산을 제안하고 있다.

2.3 V-Man 알고리즘

Moreno Marzolla가 제안하고 있는 V-Man[7]은 서버간의 가십(Gossiping) 메시지를 통하여 서로의 시스템 사용형태를 분석하고 분산되어있는 가상머신을 가장 큰 부하의 서버로 통합하는 알고리즘이다.

클라우드 상에는 다양한 가상머신들이 다수의 물리적 하드웨어에 걸쳐 수행 되는데 우선 V-Man은 각각의 물리적 하드웨어에서 수행되고 있는 가상머신의 정보를 수집하여 이후 가장 많은 양의 가상머신을 수행하는 물리적 하드웨어의 가상머신을 이전하여 유휴자원을 높이는 것을 핵심으로 하고 있다.

3. 클라우드 환경의 Thin-Client 모바일

본 논문에서 제안하는 클라우드 환경의 Thin-Client 모바일 전체 구성은 (그림 1)과 같다.

클라우드 환경의 Thin-Client 모바일 시스템 구조는 크게 두 가지로 구성 되어있다.

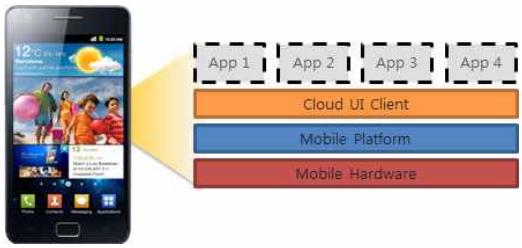
- Thin-Client 모바일 단말
- 가상 모바일 단말 클라우드 시스템

Thin-Client 모바일 단말은 가상 모바일 단말 클라우드 시스템 간에는 VNC(RFB Protocol)을 이용하여 접속하게 되며, 어플리케이션을 수행하는 동안의 모든 사용자의 입력을 클라우드 시스템의 가상 모바일 단말에서 처리하며, 모바일 단말은 화면 정보만을 받아 출력한다.

(그림 1) 클라우드 환경의 Thin-Client 모바일 시스템 구조

3.1 Thin-Client 모바일 단말

씬 클라이언트 모바일 단말은 스마트폰같이 흔히 사용하는 네트워킹이 가능한 모바일 단말을 의미한다. 여기에 기본적으로 사용가능한 최소한의 씬 클라이언트 인터페이스를 탑재하고 있어 네트워크를 통해 클라우드 시스템에 접속하게 된다.



(그림 2) Thin-Client 모바일 단말 구조

물리적 모바일 하드웨어 상에 최소한의 모바일 플랫폼(Platform) 이 존재하며, Cloud UI Client 라는 씬 클라이언트 프로토콜을 수행하기 위한 어플리케이션을 갖고 있다.

테스트 환경을 위하여 씬 클라이언트 단말은 Android 플랫폼을 갖고 있는 스마트폰을 사용하였고, 실제 클라우드 환경으로의 접속을 담당하는 어플리케이션인 Cloud UI Client는 VNC 프로토콜을 기반으로 하는 오픈소스 클라이언트 프로그램인 Android VNC Viewer[11]를 기반으로 서비스 제공을 위한 인터페이스를 개발하여 사용하였다.

3.2 가상 모바일 단말 클라우드 시스템

가상 모바일 단말 클라우드 시스템은 Thin-Client 모바일에 어플리케이션을 제공하기 위한 시스템 환경이다. x86 서버 환경을 기반으로 Hypervisor를 통해 컴퓨터 리소스를 가상화 시키고, 가상 모바일 단말 환경을 구성하여 Thin-Client로부터 들어오는 모바일 어플리케이션 요청을 DRAMMA를 통해 제공하는 역활을 담당한다.

DRAMMA는 요청된 모바일 어플리케이션에 따라 자원의 요구량을 체크하고, Hypervisor를 통해 가상 머신의 자원을 동적으로 할당하게 되는데 DRAMMA 알고리즘의 자세한 설명은 4장에서 다루도록 한다.

(그림 3) 가상 모바일 단말 클라우드 시스템 구조

4. DRAMMA 알고리즘

본 논문에서 제안하는 DRAMMA (Dynamic Resource Allocation Manager for Mobile Application) 는 아래와 같은 목적을 갖고 있다.

- 모바일 어플리케이션의 가상 머신 할당과 수행 성공 횟수의 최대화
- 클라우드 환경의 가상 머신 이동(Migration) 횟수를 최소화
- 가상 머신 자원 할당 시 발생될 수 있는 장애의 최소화

이를 바탕으로 DRAMMA는 모바일 어플리케이션 수행 성공 횟수의 최대화, 클라우드 환경의 가상 머신 이동으로 인한 서비스 지연 등의 문제를 최소화, 가상 머신 자원의 할당 시 물리적 하드웨어 자원 부족, 가상 머신 이동 횟수 증가로 인한 장애 등을 최소화를 위한 알고리즘을 제안하고 있다.

〈표 1〉 용어 정리

용어	정의
PM_k	물리적 하드웨어 k
VM_x^k	물리적 하드웨어 k에서 수행중인 가상머신 x
App_n	타입n의 모바일 어플리케이션
R	모바일 어플리케이션이 수행을 위한 자원 요구량
W	현재 사용 중인 자원의 양
T	총 개수의 합
C	총 CPU 코어의 합
$\alpha \rightarrow \beta$	α에서 β로의 가상 머신 이동

<표 1>에서 본 논문에서 다루고 있는 알고리즘을 위한 용어를 정리하였다.

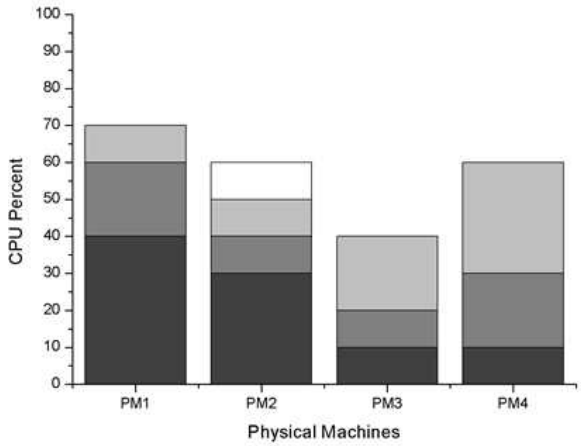
PM_k는 클라우드 시스템 내에 존재하는 물리적 하드웨어 k를 의미하며, PM_k에서 수행되는 가상머신 x를 VM_x^k로 나타낸다. 또, 모바일 어플리케이션을 그 타입(n)에 따라 App_n으로 나타낸다. App_n이 수행되기 위한 자원의 요구량은 R로 표현할 수 있으며 $R(App_n)$와 같이 사용된다. $W(PM_k)$는 PM_k에서 현재 사용되는 자원의 양을 의미하고, $W(VM_x^k)$는 VM_x^k가 현재 사용하는 자원의 양을 의미한다. T는 총 개수의 합을 의미하며 $T(PM)$은 전체 물리적 하드웨어 개수의 합을 나타낸다. C는 각 PM의 CPU 코어의 합을 의미한다. 또한 $\rightarrow$는 가상머신의 이동을 나타내며 $\alpha \rightarrow \beta$와 같이 사용한다.

이때, <표 1>과 같은 클라우드 시스템의 모바일 어플리케이션 수행상태를 고려해 보자.

〈표 2〉 클라우드 시스템의 모바일 어플리케이션 수행 현황 예시

	VM_1	VM_2	VM_3	VM_4
PM_1	40%	20%	10%	0%
PM_2	30%	10%	10%	10%
PM_3	10%	10%	20%	0%
PM_4	10%	20%	30%	0%

이를 그래프 형태로 나타내면 (그림 4)와 같으며 시스템의 한계치(Threshold)를 80으로 가정하여 생각해보자.

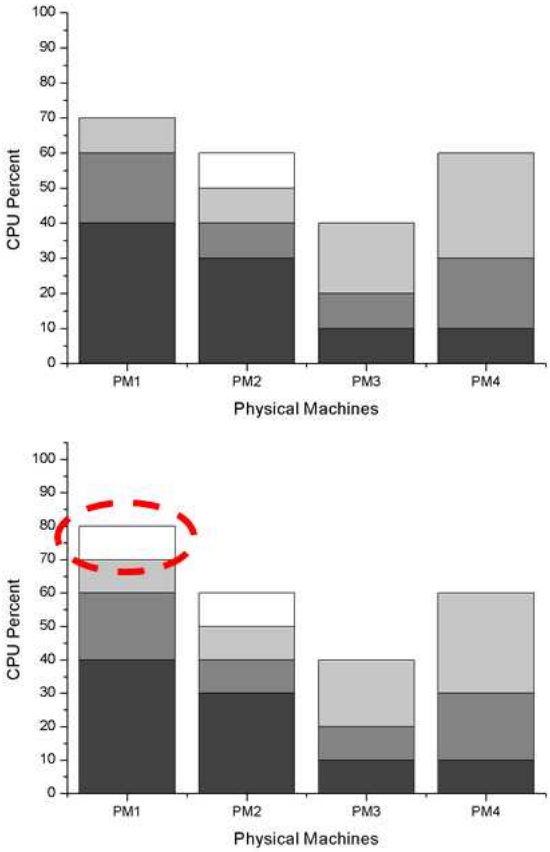


(그림 4) 모바일 어플리케이션 수행 현황 그래프

이때, $R(App_1)=10\%$, $R(App_2)=20\%$, $R(App_3)=80\%$인 App_1, App_2, App_3의 할당을 고려해보자.

(그림 5)는 $R(App_1)=10\%$으로 PM_1에 별다른 VM의 이동 없이 할당이 가능하다.

하지만 (그림 6)에서 보듯 $R(App_2)=20\%$인 어플리케이션을 PM_1에 할당할 때에는 $VM_3^1 \rightarrow VM_4^3$ 을 통해 할당이 가능하며, 좀 더 빠른 할당을 위해서는 R의 여유가 있는 PM_3, PM_4에 할당이 가능하다.

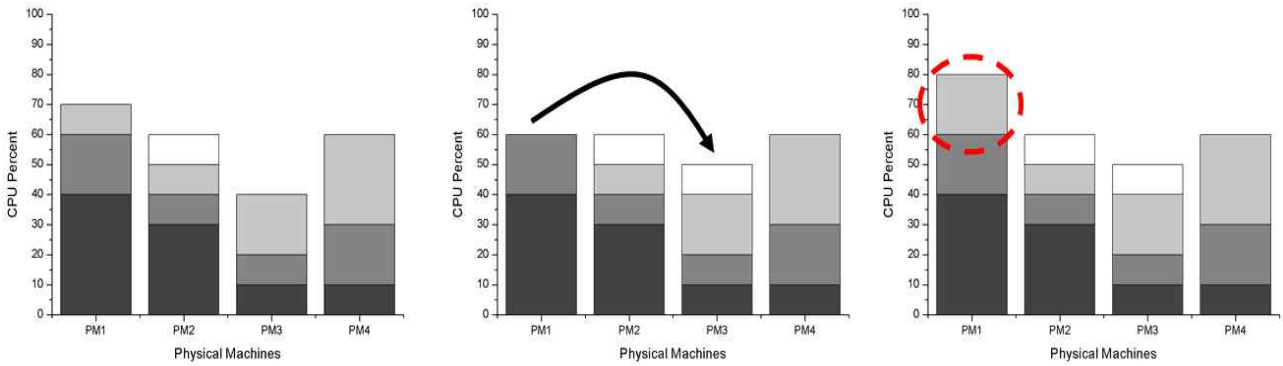


(그림 5) $R(App_1)=10\%$인 App_1할당

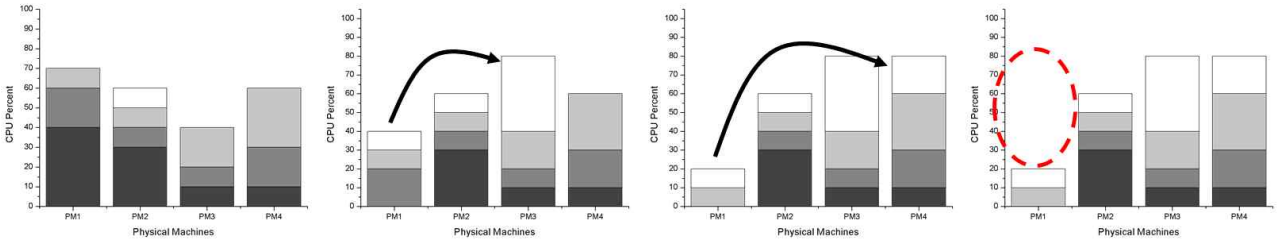
반면, (그림 7)에서 보듯 $R(App_3)=80\%$인 어플리케이션을 PM_1혹은 그 외의 PM에 할당 시에는 다수의 VM에 이동 후에도 할당이 불가능한 상태가 되어 모바일 어플리케이션 수행 실패로 돌아가게 된다.

이러한 상황을 케이스를 고려하여 모바일 어플리케이션의 할당 상태를 3가지로 정리 할 수 있다.

- VM의 이동 없이 App 할당이 가능한 상태
- VM을 이동하여 App에 필요한 R을 확보하고 할당이



(그림 6) $R(App_2)=20\%$인 App_2할당



(그림 7) $R(App_3)=80\%$인 App_3할당

가능한 상태

- VM을 이동하였지만 App에 필요한 R을 확보하지 못해 할당이 불가능한 상태

따라서 이러한 상태를 고려하여 효율적인 모바일 어플리케이션 할당을 위하여 DRAMMA는 아래와 같은 순서에 수행된다.

① PM_1 ~ PM_k까지 각 PM별 사용 중인 각 VM의 W를 모니터링
② App_n이 새롭게 할당될 때 $R(App_n)$을 확인
③ 현재 시스템 중 App_n을 할당할 수 있는 VM여부를 확인하여 할당
④ App_n할당할 여분의 VM이 없을 경우 각 $W(VM)$에 따라 VM을 이동하며 App_n의 할당 가능 여부를 확인하여 할당

이를 슈도코드로 나타내면 다음과 같다.

```
1  function madram (all_of_process)
2      // initialize variable
3      is_success = true;
4      final_pms = {};
5
6      pms = {};
7
8      for idx from 0 to count of all of process:
9          data = all_of_process[idx];
10         pm = pms[idx mod NUM_OF_PM];
11
12         sum_of_pm = sum of usage cpu in pm;
13         if  (sum_of_pm + data) <= MAX_CPU_USAGE and core count in pm < NUM_OF_CORE_PER_PM then
14             add data to pm
15         else
16             final_pms = find_minimum_swap_or_move_then_push_data($data);
17
18             # Replace to success finally swapping pms.
19             if is_success_swap == true then
20                 pms = final_pms
21             end if
22
23             if is_success_madram_swap == false then
24                 is_success = false;
25                 exit loop;
26             else
27                 is_success = true;
28             end if
29         end if
30
31         if is_success == true then
32             final_pms = pms;
33         end if
34
35     is_success_madram_algorithm = is_success;
```

(그림 8) DRAMMA 알고리즘

5. 성능평가

성능평가는 기존의 클라우드 자원 분배 알고리즘인 Sandpiper와 V-Man을 비교하여 수행하였다.

〈표 3〉 Thin-Client 모바일 단말 환경

타입	비고
모델명	Samsung Galaxy S2
CPU	Exynos 4210 1.2GHz Dual-Core
RAM	LPDDR2 1Gbyte
HDD	16Gbtye

〈표 4〉 클라우드 시스템 환경

타입	비고
모델명	HP DL120 G6
CPU	Intel X3430 Xeon 2.4GHz Quad-Core
RAM	DDR2 ECC 8Gbyte
HDD	256Gbtye
STORAGE	Qnap 4Bay 8Tbyte NAS
Hypervisor	VMware vSphere 4 ESX

〈표 5〉 가상 모바일 단말 환경

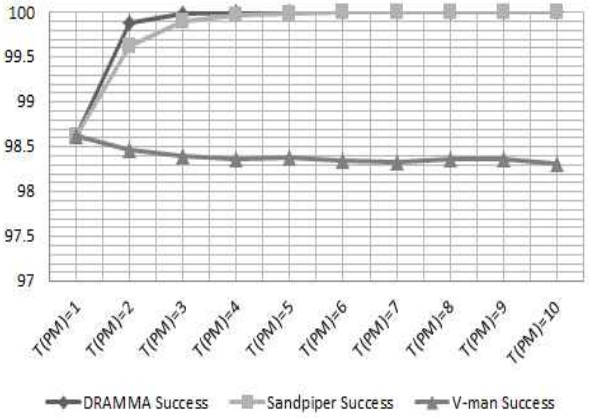
타입	비고
Virtual CPU	(Virtual) Intel 2.4GHz Dual-Core
Virtual RAM	(Virtual) DDR2 1Gbyte
Virtual HDD	(Virtual) 16Gbtye
OS	Android x86 2.2

<표 3>, <표 4>, <표 5>는 각각 Thin-Client 모바일 단말 환경과 클라우드 시스템 환경, 그리고 클라우드 시스템에서 생성된 가상 모바일 단말 환경을 나타내고 있다.

본 논문에서의 Thin-Client 모바일 단말은 현재 상용중인 고성능의 스마트폰 단말기를 사용하여 실험 수행하였지만, 모바일 단말에서 수행되는 모든 어플리케이션을 가상 머신에서 수행하고 그 결과의 화면만을 VNC를 통해 가져오는 환경으로 Thin-Client라 할 수 있다.

실험 수행에 사용된 시나리오는 통계적 방법으로 추출된 모바일 사용자를 대상으로 하는 설문조사 자료[12]로부터 모바일 어플리케이션 수행 데이터를 수집하여 다양한 조합의 경우를 만들어 실험에 사용하였다.

이를 통해 실제 실험평가를 수행한 결과는 다음과 같다.

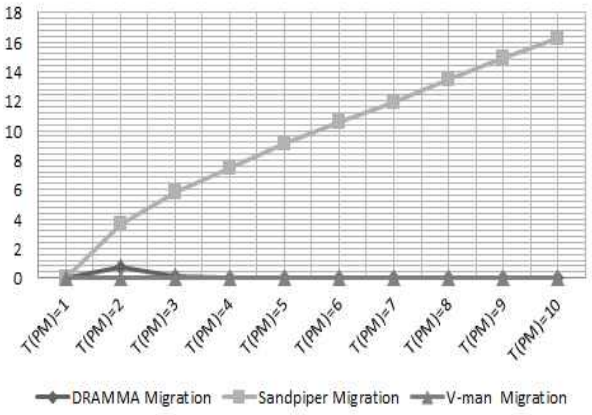


(그림 9) 모바일 어플리케이션 할당 성공 횟수

(그림 9)는 모바일 어플리케이션 할당 결과를 나타내며, Sandpiper 와 V-Man 보다 할당 성공 횟수가 높은 것을 볼 수 있다.

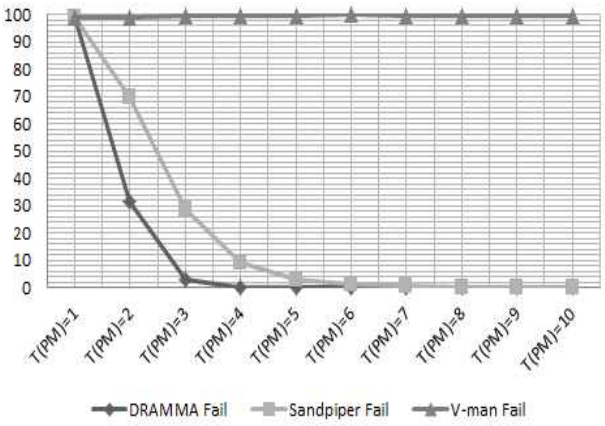
〈표 6〉 스마트폰 이용 목적 비교 자료 및 대표 어플리케이션 현황

타입	목적	시간	대표 어플리케이션
정보 접근	정보검색	0:47	인터넷
	둘러보기	0:11	인터넷
	뉴스 보기 및 듣기	0:15	인터넷, 유튜브
	업무	0:16	Crazy Remote
	학습	0:05	비디오플레이어
	쇼핑 및 금융	0:05	티켓몬스터, 하나은행
	위치 및 지도정보	0:04	서울버스, 네이버지도
	생활정보	0:04	날씨
커뮤니 케이션	메일송수신	0:08	이메일
	문자송수신	1:06	문자, 카카오톡
	전화송수신	0:35	전화
	블로그 미니홈피	0:15	Facebook
	카페 커뮤니티	0:07	인터넷
엔터테 인먼트	음악듣기	0:32	음악
	사진 찍기 및 보기	0:02	갤러리
	책 읽기	0:02	성경
	게임	0:14	Angry Birds
	영화	0:01	비디오플레이어
	드라마	0:05	DMB TV
	오락 및 스포츠	0:09	클럽맞고
	기타	0:11	기타
총계		5:14	(시:분)



(그림 10) 모바일 어플리케이션 할당 중 이동횟수

(그림 10)은 모바일 어플리케이션 할당 중 발생하는 이동 횟수를 나타낸 것이며, Sandpiper 보다 낮고, V-Man보다 높은 것을 확인 할 수 있었다.



(그림 11) 모바일 어플리케이션 할당 중 실패 횟수

(그림 11)은 모바일 어플리케이션 할당 중 발생하는 실패 횟수를 나타낸 것이며 Sandpiper 및 V-Man 보다 낮음을 확인할 수 있었다.

〈표 7〉 실험평가 결과 종합

	수행 성공	이동 횟수	수행 실패
Sandpiper	1.0004배 증가	8.25배 감소	1.58배 감소
V-Man	1.0147배 증가	12.38배 증가	7.45배 감소

실험 평가 결과를 종합하면 <표 7>과 같이 나타나며 이를 통해 DRAMMA가 다른 클라우드 환경의 자원 분배 알고리즘과 비교하여 모바일 어플리케이션 서비스에서 수행 성공 횟수의 증가와 VM이동의 감소, 수행 실패 횟수의 감소로 효율적인 자원할당 성능을 보여주는 것을 확인할 수 있다.

6. 결 론

본 논문을 통해 클라우드 환경의 Thin-Client 모바일을 위한 동적 자원 분배 기술인 DRAMMA를 살펴보았고, 실험 평가를 통해 DRAMMA가 기존의 자원 분배 알고리즘 대비 효율적으로 모바일 어플리케이션을 할당하는 것을 확인할 수 있었다.

하지만 다른 Sandpiper 등의 알고리즘과 비교할 때 DRAMMA은 오직 CPU만을 고려하여 가상 머신을 분배하기 때문에 향후 CPU외에 Memory, Network 등의 가상 머신 성능을 고려한 자원 할당 기술의 개선이 필요할 것으로 생각된다. 또한 같은 모바일 어플리케이션을 이용 시 Memory 공유 기법 등의 성능 개선하여 모바일 어플리케이션 동적 자원 할당 기술로 자리 매김할 것이라 예상된다.

참 고 문 헌

[1] John McCarthy, “Centennial Keynote Address”, 1961.
[2] “Cloud Computing”, http://en.wikipedia.org/wiki/Cloud_computing, 2011.
[3] Eric Y. Chen, Mistutaka Itoh, “Virtual smartphone over IP”, IEEE International Symposium on A World of Wireless, Mobile and Multimedia Networks, pp.1-6, June, 2010.
[4] 이준형, 탕웨이, 허의남, “Cloud 환경에서의 Thin-Client 모바일 시스템”, 2011년도 한국통신학회 동계종합학술발표회 논문초록집, Vol.44, pp.300, 2011년 2월.
[5] Wei Tang, Jun-hyung Lee, Diao Song, Md. Motaharul Islam, Sangho Na, Eui-Nam Huh, “Multi-Platform Mobile Thin Client Architecture in cloud environment”, International Conference on Innovative Computing and Communication and Information Technology and Ocean Engineering, March, 2011.
[6] Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif, “Sandpiper: Black-box and gray-box resource management for virtual machines”, Computer Networks Journal (ComNet), Vol.53(17), pp.2923-2938, 2009.
[7] Moreno Marzolla, Ozalp Babaoglu, Fabio Panzieri, “Server Consolidation in Clouds through gossiping”, IEEE International Symposium on A World of Wireless, Mobile and Multimedia Networks, June, 2011, pp.1-6.
[8] “Remote Desktop Protocol”, http://en.wikipedia.org/wiki/Remote_Desktop_Protocol, 2011.
[9] “Application of VNC”, http://www.realvnc.com/vnc/how.html, 2011.
[10] “Desktop virtualization”, http://en.wikipedia.org/wiki/PCoIP, 2011.
[11] “Android-VNC-Viewer”, http://code.google.com/p/android-vnc-viewer, 2011.
[12] 황주성, 이재현, 이나경, “모바일 인터넷으로 인한 미디어 이용 패턴의 변화 : 스마트폰 이용자를 중심으로”, 정보통신정책연구원, 2010년 12월.

이 준 형

e-mail : junhyung@kisti.re.kr

2010년 경희대학교 컴퓨터공학과(학사)

2012년 경희대학교 컴퓨터공학과 (공학석사)

2012년~현 재 한국과학기술정보연구원 슈퍼컴퓨팅센터 차세대연구 환경 개발실 연구원

관심분야 : 클라우드 컴퓨팅, 씬 클라이언트, 슈퍼컴퓨터 등

허 의 남

e-mail : johnhuh@khu.ac.kr

1990년 부산대학교 전산통계학과(학사)

1995년 University of Texas 전산학과 (공학석사)

2002년 The Ohio University 전산학과 (공학박사)

2003년~2005년 서울여자대학교 컴퓨터공학과 조교수

2005년~현 재 경희대학교 컴퓨터공학과 교수

관심분야 : 클라우드 컴퓨팅, 씬 클라이언트, 센서네트워크, 네트워크 보안 등