서비스 기반 모바일 어플리케이션의 MVC 아키텍처 및 적용 사례연구

(MVC Architecture and a Case Study for Service-based Mobile Applications)

이 호 중 † 라 현 정 †
(Ho Joong Lee) (Hyun Jung La)
김 수 동 ††
(Soo Dong Kim)

요 약 최근 들어 모바일 디바이스는 편리한 통신기능과 휴대성을 제공할 뿐 아니라, 다양한 어플리케이션을 실행할 수 있는 장비로 활용되고 있다. 그러나, 모바일 디바이스의 자원 제약성으로 인해 복잡도가 높은 어플리케이션에는 한계를 가지고 있다. Model-View-Control(MVC)아키텍처는 다양한 어플리케이션 설계에 널리 사용되고 있지만, 서비스 기반의 모바일 어플리케이션의 특징을 모두 반영하지 못한다. 본 논문에서는 고품질 고성능의 서비스 기반 모바일 어플리케이션 설계를 위해, 기존의 MVC 아키텍처를 확장한 효과적인 모바일 앱 아키텍처 모형과 적용 기법을 제시한다. 이를 적용하면, 자원제약성 문제를 크게 해소하고, 복잡도가 높은 어플리케이션 개발이 가능해 진다.

키워드 : 모바일 디바이스 모바일 앱, MVC 아키텍처

Abstract Mobile devices are utilized not only for communications but also for running applications. However, applications with high complexity could not be deployed on mobile devices which have inherent resource limitation. While Model-View-Control(MVC) is commonly used in designing applications, it does not address mobile device specific characteristics. In this paper, we propose effective mobile application architectures which extend the conventional MVC architecture, to develop mobile application with high performance while remedying their resource constraints.

Key words : Mobile Device, Mobile Application, MVC Architecture

· 이 논문은 정보통신산업진흥원의 SW공학 요소기술 연구개발사업에 의해 지원 되었음을 밝힙니다. 이 논문은 MKE/KEIT R&D 연구 사업의 (KI002076, 네트워크 기반 수요자 지향 융합 서비스 공통플랫폼 기술개발) 지원을 받아 수행 되었음을 밝힙니다.
· 이 논문은 2010 한국컴퓨터종합학술대회에서 '서비스 기반 모바일 어플리케이션의 MVC 아키텍처'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 숭실대학교 컴퓨터학과
leehojoong@gmail.com
hjla80@gmail.com
†† 종신회원 : 숭실대학교 컴퓨터학과 교수
sdkim777@gmail.com

논문접수 : 2010년 8월 5일
심사완료 : 2010년 10월 5일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제16권 제11호(2010.11)

1. 서 론

스마트 폰으로 불리는 최근의 모바일 다바이스는 통신 기능뿐 아니라, 어플리케이션 실행도 가능하게 한다. 그러나, 다양한 자원 제약성으로 복잡성이 높은 어플리케이션 실행은 어렵다. 서비스 기반 모바일 컴퓨팅은 이를 효과적으로 해결하기 위한 효과적인 방법이다.

Model-View-Control(MVC) 아키텍처는 다양한 어플리케이션 설계에 널리 사용되고 있다[1]. 기본적으로 세 부분으로 나누어져 있으며 입출력을 담당하는 뷰는 클라이언트에만 존재하고 나머지 컨트롤과 모델만 기능에 따라 나누면 되기 때문에 서비스 기반 모바일 어플리케이션을 설계함에 있어서 적합하다. 그러나, 기존의 MVC 아키텍처는 모바일 어플리케이션의 주요 특성들을 고려한 설계를 지원하지 못하고 있다.

본 논문은 MVC를 확장하여 서비스 기반 모바일 어플리케이션 개발에 활용할 수 있는 아키텍처를 제안하고, 이를 적용한 사례연구를 보여준다. 제시된 아키텍처를 적용하면, 자원 제약성을 해결하면서 높은 성능을 제공하는 모바일 어플리케이션을 효과적으로 개발할 수 있다.

2. 관련 연구

Peter의 연구에서는 모바일 디바이스가 자바를 기반으로 된 어플리케이션의 실행이 가능하고 내장된 웹 브라우저가 있는 특징을 이용해 자신들이 개발하는 소프트웨어 아키텍처 설계하였다[2]. 설계한 아키텍처는 GUI 비즈니스 로직으로 구성된 클라이언트와 비즈니스 로직과 데이터베이스로 구성된 서버로 구성되어 있다. 이러한 구조를 모델 기반으로 변경해 전체를 포괄하는 어플리케이션 모델이 존재하고 그 안에 GUI 모델로 화면을 담당하는 부분과 각각의 특징으로 특징 모델을 가지고 있는 아키텍처를 제안한다.

Chen의 연구에서는 두 가지의 MVC 패턴의 문제점을 제시하고 다른 패턴들을 이용하여 해결하고 있다[3]. 본 논문은 기존 MVC 아키텍처의 제약사항을 규명하고, 이를 보완하기 위한 방법을 제안한다.

Natchetoi의 연구에서는 모바일 어플리케이션의 문제점을 해결하기 위한 방안으로 SOA를 사용, 네트워크 연결에 대한 투명성, 서비스간의 낮은 결합도, 저 비용의 ownership을 제시한다[4].

이러한 일련의 연구들은 MVC 기법의 보완과 SOA의 네트워크 비용을 최소화하는 기법에 주안점을 두고 있고, 서비스 기반 모바일 컴퓨팅의 주요 기술적인 관점은 다루고 있지 않다.

3. 서비스기반 모바일 어플리케이션의 개요

3.1 모바일 어플리케이션의 특성

모바일 어플리케이션은 모바일 디바이스의 특징이 잘 반영되도록 개발된다[5]. 일반적인 모바일 디바이스의 특성은 다음과 같다.

- 컴퓨팅 자원 한계성: 편리한 이동성 때문에 메모리가 작고 CPU의 성능이 낮아 성능 면에서 컴퓨터보다는 많이 떨어진다.
- 물리적인 위험: 휴대성과 이동성에 따라, 분실이나 파손의 가능성이 높고, 이로 인해 개인적인 정보의 손실이 발생할 수 있다.
- 다양한 네트워킹과 불안정한 연결성: Wi-Fi, 3G등 다양한 무선 네트워킹 지원으로 서비스 기반의 어플리케이션 개발을 가능하게 한다. 반면, 좁은 대역폭과 불안정한 연결성이 존재하여 아키텍처 관점의 해결이 필요하다.

3.2 서비스 기반 모바일 어플리케이션

모바일 디바이스의 제약사항을 극복하기 위해 풍부한 네트워크 연결성이라는 특성을 이용한 서비스 기반 모바일 어플리케이션을 사용한다.

서비스 기반 모바일 어플리케이션은 디바이스에서 실행해야 할 기능을 일부 또는 대부분을 서비스로 제공받는 어플리케이션을 말한다[4]. 서비스 기반 모바일 어플리케이션은 그림 1과 같은 구조를 가진다.

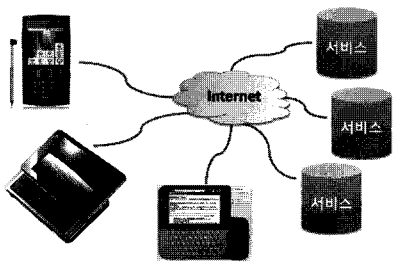

그림 1 서비스 기반 모바일 어플리케이션

표 1 서비스를 이용한 모바일 디바이스 특성 해결

제약사항	서비스 기능	해결
부족한 컴퓨팅 자원	풍부한 자원	복잡도 높은 연산 풍부한 저장공간
물리적인 위험	데이터 저장 및 보관	데이터 손실 위험 최소화

서비스 기반의 모바일 어플리케이션은 표 1과 같이 모바일 디바이스 특성을 극복할 수 있다.

4. 서비스 기반 모바일 어플리케이션에 적용한 MVC 아키텍처

4.1 Thin-Client MVC 아키텍처

모바일 어플리케이션도 UI를 가지는 어플리케이션 이기 때문에 기존 MVC 패턴이 적용이 가능하지만 모바일 어플리케이션의 특성 때문에 문제점이 발생한다. 이것을 해결하기 위해 그림 2와 같이 Thin-Client[6]을 이용한 MVC 아키텍처를 제안한다.

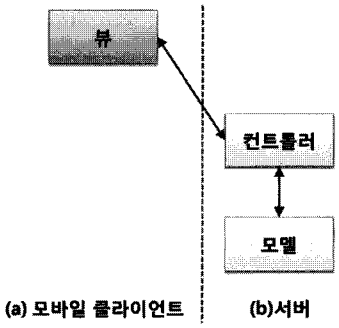

그림 2 Thin-Client MVC 아키텍처

Thin-Client MVC 아키텍처에서 사용자에게 정보를 입 출력하는 뷰는 모바일 클라이언트에 존재하고 컨트롤러와 모델은 서버에서 서비스로 제공된다.

Thin-Client MVC에서 컨트롤러가 모바일 클라이언트에서 분리됨으로써 복잡한 연산 시에 클라이언트의 낮은 연산 능력 때문에 발생하는 문제점을 극복할 수 있다. 그리고 모델의 분리는 부족한 데이터 저장공간을 확보하고 모바일 디바이스의 분실 및 파손으로 인한 데이터 손실을 막을 수 있다.

4.2 Balanced MVC 아키텍처

Thin-Client MVC 아키텍처의 문제점은 네트워크 의존성이 높다는 것으로 보완할 수 있는 그림 3과 같은 Balanced MVC 아키텍처를 제안한다.

컨트롤러와 모델이 서버에만 국한되어 존재하게 된다면 모든 정보와 결과들을 네트워크를 통해서만 가져오기 때문에 데이터를 빈번하게 이용하는 어플리케이션에

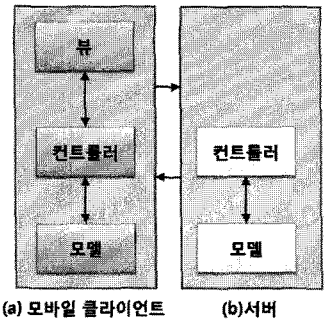

그림 3 Balanced MVC 아키텍처

서는 비효율적이다.

그러므로, 컨트롤러와 모델의 필수적인 부분들을 모바일 클라이언트에서 자주 사용되는 데이터의 사본을 가지고 있음으로써 네트워크 비용을 줄일 수 있다. 또한 개인정보를 디바이스에 저장함으로써 노출을 방지할 수 있다. 하지만 모델간의 데이터 일관성 문제가 발생해 서로 통신하며 업데이트하는 기능이 필수적이다.

5. 모바일 클라이언트와 서버간의 상호작용

Balanced MVC 아키텍처는 그림 4와 같이 컨트롤러와 모델이 모바일 클라이언트와 서버에 각각 존재한다. 따라서 구성요소들 간의 상호작용이 존재한다.

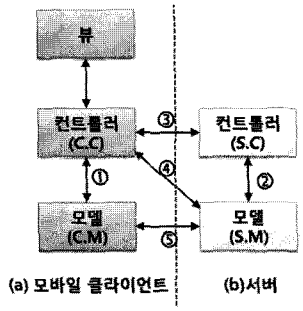

그림 4 레이어 사이의 상호작용

• 클라이언트 컨트롤러(C.C)-클라이언트 모델(C.M)

C.C가 뷰에서 요청을 받으면 C.M의 정보를 가져와 그것을 연산 또는 변환하여 뷰에 반환한다. 하지만 C.C는 간단한 연산만을 하게 되어있고 C.M도 모든 데이터를 가진 것이 아니라 일부 데이터(캐시, 개인 정보, 임시 저장 정보)를 가지고 C.C에 제공한다.

• 서버 컨트롤러(S.C) - 서버 모델(S.M)

S.C와 S.M은 클라이언트가 처리하기 힘들거나 부족한 메모리를 극복하기 위해 분리되어 제공되는 기능이기 때문에 S.C은 뷰의 요청을 받는 것이 아니라 S.C는 C.C의 요청을 받고 S.M에 처리할 데이터를 요청한다.

• 클라이언트 컨트롤러(C.C) - 서버 컨트롤러(S.C)

C.C와 S.C는 기존 MVC 아키텍처의 컨트롤러의 기능을 분리하여 가지고 있다. C.C는 자신이 처리할 수 없는 복잡성 높은 연산을 S.C에 요청하고 S.C는 요청에 대한 답을 C.C로 반환한다.

• 클라이언트 컨트롤러(C.C) - 서버 모델(S.M)

실제 C.C가 사용하고 싶은 정보를 아무런 처리 없이 가져오고 싶을 때 발생되는 부분이다. 하지만 C.C와 S.M은 서로 네트워크로 연결되어 있기 때문에 많은 양의 데이터를 가져오는 것은 S.C로 처리한다.

• 클라이언트 모델(C.M) - 서버 모델(S.M)

C.M과 S.M가 가지고 있는 데이터는 일부분 동일하다. C.M은 사용자의 개인정보를 S.M로 보내지 않고 보관하고 있으며 S.M의 일부 데이터를 캐시형태로 보관하고 있다. S.M은 모바일 어플리케이션이 사용하는 공용 데이터를 가지고 있다. C.M과 S.M가 동시에 가지고 있는 데이터의 일관성이 문제가 된다.

6. 사례 연구

본 장에서는 제안한 MVC 아키텍처의 적용성을 보여주기 위하여, Mobile Mate Service(MMS) 개발에 적용한 사례를 제시한다. MMS는 모바일 디바이스 기반의 사용자의 위치정보를 이용하여 소셜 네트워킹을 지원하는 어플리케이션이다.

6.1 모바일 메이트 서비스의 설계 및 구현

MMS는 Balanced MVC 아키텍처를 이용해 설계하고 구현되었다. 그림 5는 기능 뷰(Information View) 관점에서의 MMS 아키텍처를 보여준다. 그림의 왼쪽 부분은 모바일 디바이스에 배포되는 클라이언트 측 어플리케이션을 나타내며, 오른쪽 부분은 서버 측에 배포된 서비스를 나타낸다. Balanced MVC 아키텍처를

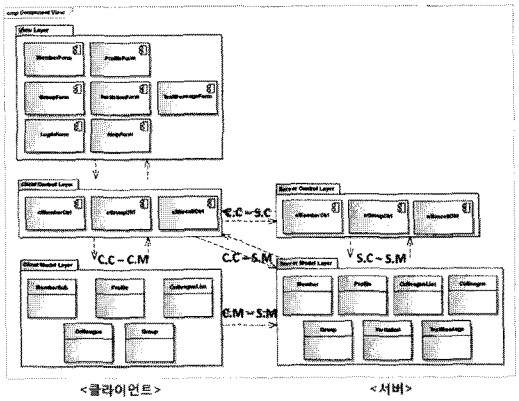

그림 5 MMS의 아키텍처 - 기능 뷰

적용하여, 클라이언트 측에는 모델, 컨트롤러, 뷰로, 서버 측에는 모델과 컨트롤러로 구성되어 있다. 그리고, 5장에서 언급한 것과 같이 각각의 레이어들은 상호작용을 가지고 있다.

5장에서 상호작용에 따라 MMS의 아키텍처가 나타난 특징들은 다음과 같다.

• 클라이언트 컨트롤러(C.C)-클라이언트 모델(C.M)

사용자의 정보를 이용, 사용자의 그룹에 대한 디바이스 안의 정보를 관리할 때 발생한다. 자신이 가지고 있는 정보만을 가져오는 것이기 때문에 네트워크가 사용되지 않는다.

• 서버 컨트롤러(S.C) - 서버 모델(S.M)

사용자가 서버로 요청한 사용자 정보관리, 그룹 관리, 초대 또는 텍스트 메시지에 대한 정보 관리할 때 발생한다. 서버가 들어온 요청을 처리하는데 정보의 관리가 필요할 때 사용된다.

• 클라이언트 컨트롤러(C.C) - 서버 컨트롤러(S.C)

다른 사용자에게 메시지를 보내거나 정보를 검색할 때 발생한다. 클라이언트는 사용자와 연결된 정보만을 가지고 있다. 따라서 다른 사용자를 초대하거나 복잡한 연산이 필요한 경우 클라이언트는 서버에게 요청해 그 정보를 받아온다. 따라서 많은 정보를 모바일 디바이스에 저장하고 있을 필요가 없으며 복잡한 연산으로 인한 리소스를 소비하지도 않는다.

• 클라이언트 컨트롤러(C.C) - 서버 모델(S.M)

다른 사용자의 사용자 정보를 가져올 때 발생한다. 별도의 가공이 필요 없기 때문에 C.C가 직접 S.M에 접근하는 것이 S.C를 통하는 것보다 더 효율적이다.

• 클라이언트 모델(C.M) - 서버 모델(S.M)

사용자 또는 그룹의 정보의 변경이 발생하였을 때 클라이언트와 서버의 모델의 정보를 일관성 있게 자동으로 동기화 할 때 필요하다.

(a) 초대 메시지 입력 (b) 현재 그룹멤버 위치 표시

그림 6 구현 화면

각각의 상호작용을 반영하여 설계된 아키텍처를 따라 구현된 MMS의 실행화면은 그림 6과 같다. (a)는 사용자를 초대하기 위해 초대 메시지를 입력하는 화면이다. 초대 메시지를 보내게 되면 그 메시지를 받은 사용자는 것을 수락하거나 거부할 수 있다. 이 결정에 따라 메시지가 달리 전송되며 수락 시 초대한 사용자의 멤버리스트에 초대한 사용자가 추가된다. 이렇게 초대한 멤버들은 그룹으로 나누어 관리할 수 있으며 그룹별로 (b)와 같이 멤버들의 위치를 지도로 볼 수 있다.

6.2 사례 연구의 기술적 관점의 레슨

사례연구에서는 모바일 어플리케이션 제약사항을 해결하기 위해 Balanced MVC 아키텍처를 사용하였다. 이를 적용한 아키텍처의 구현을 통해 도출된 이슈는 다음과 같다.

• Balanced MVC의 사용으로 디바이스 자원 사용 최소화

MMS는 클라이언트의 컨트롤과 모델을 서버로 기능 분할시켜 클라이언트의 연산을 최소화고 서버의 넓은 저장 공간을 활용한다. 따라서 클라이언트의 자원 사용을 최소화하고 충분한 저장 공간을 활용한다.

• 외부서비스인 Google Map 활용

MMS는 지도를 사용하기 위해 Google Map 서비스를 활용하고 있다. 그림 7과 같이 설계에 그것을 반영하였다. 외부서비스를 사용함해 개발 시간을 단축시켰다.

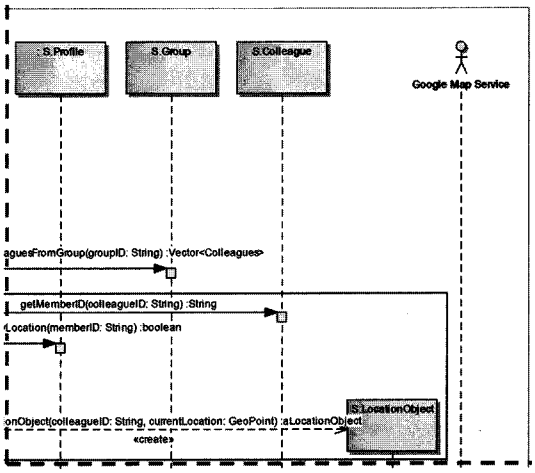

그림 7 외부서비스 사용 - 동적 모델(일부)

• 클라이언트와 서버의 통신에 사용된 기술

MMS는 모바일 클라이언트에서 수행되는 어플리케이션으로 서버와 클라이언트 사이의 메시지 전달 문제가 발생하였다. 클라이언트가 서버로 정보를 요청하고 응답을 받는 것은 모두가 가능하지만 서버에서 클라이언트로 정보를 보내거나 요청하는 것은 불가능하다. 그 이유

는 클라이언트에 서버의 역할이 없기 때문에 연결을 받아 들일 수 없다. 또한 여러 정보를 한번에 전송하기 위해 객체 전달이 필요하다.

표 2는 이러한 특징들을 비교한 것이다. 따라서 MMS에서 필요한 양방향 메시지 전달과 객체 전달에 부분을 모두 만족하는 Socket을 구현 시 사용하였다.

표 2 데이터 전송 기술 별 특징 비교

	메시지 전달		객체 전달
	Client->Server	Server->Client	
Socket	●	●	●
SOAP	●	×	×
RESTful	●	●	×

• 전송 받지 못한 메시지의 처리

MMS는 소셜 네트워킹을 지원하는 어플리케이션으로 메시지와 초대 정보를 교환하고 있다. 하지만 이러한 교환에 있어서 로그인과 로그아웃 상태를 구분해야 한다. 로그인 상태이면 받는 사람이 즉시 확인할 수 있지만 로그아웃 상태이면 즉시 확인하는 것이 아니라 로그인 했을 때 그 메시지를 확인할 수 있도록 설계해야 한다. 따라서 MMS는 이러한 기능을 서버에 구현해 사용자의 초대, 메시지 전송에 대한 부분을 보완하였다.

• 데이터 동기화

MMS는 Balanced MVC 아키텍처 사용하여 설계되었기 때문에 모델이 클라이언트와 서버로 나누어져 있다 또한 모든 서버의 데이터를 클라이언트들이 공유하기 때문에 서버와 클라이언트의 정보 동기화는 필수 정이다. 따라서 이렇게 자주 발생하는 데이터 교환은 컨트롤을 거치는 것이 아니라 모델 사이에서 자동적으로 수행하는 것이 더 효율적인 성능을 보장할 수 있다.

7. 결 론

모바일 디바이스는 제한된 리소스, 물리적인 위험, 다양한 네트워킹과 불안정한 연결성이라는 특성을 가지고 있다. 따라서 모바일 디바이스에서 실행될 수 있는 어플리케이션의 복잡도에 제약이 존재한다. 이것을 보완하기 위해 다른 특성인 풍부한 네트워크 연결성을 이용한다. 이를 통해 복잡도가 높은 어플리케이션이 실행 가능해지며 풍부한 메모리 공간을 확보할 수 있다.

모바일 어플리케이션의 단점을 보완할 수 있는 방법으로 본 논문에서는 기존의 어플리케이션 설계시 일반적으로 사용하는 MVC 아키텍처를 적용하여 서비스 기반의 어플리케이션을 설계에 이용할 수 있는 두 MVC 아키텍처를 제안하고 아키텍처의 장단점을 제시하였다. 또한 Balanced MVC 아키텍처의 각 구성요소 사이의 5가지 상호작용을 제시하고, 장단점을 비교하였다.

표 3은 자립형(Stand-alone) 앱과 Balanced MVC 아키텍처 사용 앱을 비교한 것이다. 네트워크 의존도가 높다는 단점을 제외하면 Balanced MVC 아키텍처 사용 앱은 자립형 앱에서 발생되는 모바일 디바이스의 한계를 극복할 수 있다는 것을 알 수 있다.

이렇게 도출된 기법을 숭실대학교 모바일 서비스 소프트웨어 공학센터 내 연구 프로젝트인 MMS 도메인에 적용한 사례연구를 통해 제시된 Balanced MVC 아키텍처의 적용성을 증명하였다.

표 3 자립형 앱과 Balanced-MVC 사용 앱 비교

구분	자립형	Balanced-MVC
자원사용	많음	적음
대용량 데이터 이용	불가능	가능
네트워크 의존도	낮음	높음
장기간 지속 메시지 전송	불가능	가능
데이터 백업	불가능	가능

본 논문에서 제안하는 확장된 MVC 아키텍처는 자원 제약성을 해결하면서 높은 성능을 제공하는 서비스 기반 모바일 어플리케이션을 효과적으로 개발하는데 적용할 수 있다.

참 고 문 헌

[1] E. Gamma, R. Helm, R. Johnson, and J. M. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.

[2] P. Braun, R. Eckhaus, "Experiences on model-driven software development for mobile applications," *In proceedings of the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, 2008 (ECBS 2008)*, pp.490-493, 2008.

[3] Chen Liyan, "Application research of using design pattern to improve layered architecture," *In proceedings of IITA International Conference on Control, Automation and Systems Engineering 2009 (CASE 2009)*, pp.303-306, 2009.

[4] Y. Natchetoi, V. Kaufman, and A. Shapiro, "Service-oriented architecture for mobile applications," *In Proceedings of the 1st international workshop on Software architectures and mobility (SAM '08)*, pp.27-32, 2008.

[5] I. Salmre, *Writing Mobile Code: Essential Software Engineering for Building Mobile Applications*, Addison-Wesley Professional, 2005 (chapter 2).

[6] Lai, A.M and Nieh, J., "On the performance of wide-area thin-client computing," *ACM Transaction on Computer System*, vol.24, no.2, pp.175-209, 2006.