2013년 8월 전자공학회 논문지 제 50 권 제 8 호
Journal of The Institute of Electronics Engineers of Korea Vol. 50, NO. 8, August 2013 http://dx.doi.org/10.5573/ieek.2013.50.8.203

논문 2013-50-8-25

모바일 TPC-C: 하둡 기반 안드로이드 모바일 TPC-C 어플리케이션 구현 및 성능 측정

(The Implementation and Performance Measurement for Hadoop-Based Android Mobile TPC-C Application)

장 한 을*, 노 재 춘*, 김 병 문*, 이 지 은*, 박 성 순**

(Han-uer Jang, Jaechun No©, Byung-moon Kim, Ji-eun Lee, and Sung-soon Park)

요 약

최근 스마트 기기와 모바일 어플리케이션에 대한 관심이 급증함에 따라 모바일 클라우드 컴퓨팅은 클라우드 서비스 발전에 중요한 이슈가 되고 있다. 그러나 모바일 클라우드 서비스가 다양한 분야로 확산되기 위해서는 많은 문제들이 해결되어야 하는데, 그중 하나로 모바일 기기들과 클라우드 서버간의 성능을 들 수 있다. 본 연구에서는 모바일 기기들과 클라우드 서버간의 성능 분석에 활용될 수 있는 모바일 TPC-C를 구현하였다. 모바일 TPC-C는 기존 TPC-C 벤치마크를 안드로이드 기반 모바일 기기 상에서 실행될 수 있도록 변환하였으며, 하둡 기반 클라우드 서버와 데이터 교환이 가능하도록 구현되었다. 본 논문에서는 모바일 TPC-C를 사용하여 모바일 환경에서 클라우드 서버로 요청될 수 있는 트랜잭션들의 성능을 측정하였고, 이를 데스크톱과 비교하여 성능 측정에 영향을 줄 수 있는 인자들을 분석하였다.

Abstract

Due to the rapid growth of mobile devices and applications, mobile cloud computing is becoming an important platform in the development of cloud services. However, the mobile cloud computing is facing many challenges in terms of the computing resources and communications. One of them is the performance issue between mobile devices and cloud server. In the paper, we implemented a hadoop-based android mobile application, called mobile TPC-C, and used it for evaluating the performance aspect between mobile devices and cloud server. The mobile TPC-C was implemented based on the existing TPC-C, to make it possible to execute on top of android mobile devices. The performance measurement using mobile TPC-C was executed on various transactions while changing the number of mobile clients. By comparing it to the evaluation on the personal PC, we tried to point out the important aspects affecting the performance improvement between mobile clients and cloud server.

Keywords : Mobile Cloud Computing, Mobile TPC-C, TPC-C Benchmark, Hadoop Cluster

* 정회원, 세종대학교 컴퓨터공학과
(Department of Computer Engineering, SejongUniversity)
** 정회원, 안양대학교 컴퓨터 공학과
(Department of Computer Engineering, Anyang University)
© Corresponding Author(E-mail: jano@sejong.ac..kr)
※ 본 논문은 2010년 정부(교육과학기술부)의 재원으로 한국 연구재단의 지원을 받아 수행된 기초 연구사업임 (2010-0022376). 또한, 2008년 안양대학교 안식년 연구를 지원받았음.
접수일자: 2013년4월16일, 수정완료일: 2013년7월23일

Ⅰ. 서 론

2008년 구글, 마이크로소프트, IBM 등 대형 IT기업들이 클라우드 컴퓨팅 시장에 뛰어든 이후로 '클라우드 컴퓨팅(Cloud Computing)'은 IT분야의 새로운 키워드로 빠르게 부상하고 있다.

기본적으로 클라우드 서비스[1-4]는 네트워크 접속을 통해 사용자 PC 뿐 아니라 노트북, 스마트폰과 같은 단말기를 통합할 수 있는 환경을 구축하며, 동시에 이에 필요한 프로세서, 스토리지, 소프트웨어 등을 제공하는 플랫폼을 의미한다. 그 결과 사용자는 개인용 PC의 성능을 향상시키기 위해 별도의 비용을 들이지 않아도 되며, 언제 어디서나 원하는 작업을 수행할 수 있는 것이 가능하다. 최근에는 3G나 무선 인터넷의 접근성이 개선된 다양한 스마트폰들이 등장하면서 개인 컴퓨팅 환경이 PC 중심에서 웹 기반의 모바일 단말 중심으로 이동하는 추세이고[5~6], 이에 따라 모바일 클라우드 컴퓨팅 서비스[7]가 새롭게 부각되고 있다.

최근 클라우드 서비스는 데이터의 다양화와 대용량화로[8~10] 클라우드 서버의 성능이 다른 때보다 중요하게 인식되고 있다. 특히 모바일 클라우드 컴퓨팅 서비스를 이용하는 사용자가 급증함에 따라 모바일 기기와 클라우드 서버간의 성능 또한 중요한 관심사로 대두되고 있다.

본 연구에서는 클라우드 서버와 안드로이드 모바일 기기간의 성능 분석에 활용할 수 있는 모바일 TPC-C를 구현하였다. 모바일 TPC-C는 OLTP (online transaction processing) 성능 측정에 활용되는 기존의 TPC-C 벤치마크를 모바일 기기 상에서 실행될 수 있는 어플리케이션으로 변환한 것으로, 모바일과 클라우드 서버간의 통신을 위해 본 연구에서 구현한 모바일 TPC-C WAS(Web Application Server)와 결합하여 하둡 기반 클라우드 서버와 데이터 교환이 가능하도록 되어 있다. 모바일 TPC-C 구성 요소는 크게 다음과 같다:

- 수행페이지 - 엑티비티(Activity)로 구성된 4개의 수행페이지로 TPC-C 트랜잭션을 실행하기 위해 필요한 정보를 입력받고 결과 값을 나타내 주는 사용자 인터페이스이다.
- 모바일 TPC-C WAS - 모바일 TPC-C와 클라우드 서버 간에 원활한 트랜잭션 수행을 위해 필요한 통신 세션을 유지하는 아파치 톰캣 기반의 웹 어플리케이션 서버이다.
- 하둡 기반 클라우드 서버 - 모바일 TPC-C와의 통신을 통해 데이터 저장 및 액세스를 실행하는 저장 매개체 역할을 담당한다.

본 논문은 다음과 같이 구성된다. Ⅱ장에서는 모바일 클라우드 컴퓨팅과 OLTP 성능 측정 벤치마크인 기존 TPC-C에 대해 서술한다. Ⅲ장에서는 모바일 TPC-C의 설계 및 구현을 기술하며, Ⅳ장에서는 하둡 기반 클라우드 서버를 사용하여 모바일 TPC-C에 대한 성능을 측정한다. 마지막으로 Ⅴ장에서는 결론을 맺고 향후 연구 방향을 제시한다.

Ⅱ. 관련 연구

2.1 모바일 클라우드 컴퓨팅

모바일 클라우드 컴퓨팅[5~6]은 기존의 클라우드 컴퓨팅의 개념에 최근 이슈화되고 있는 스마트폰의 휴대성을 접목시켜 사용자가 언제 어디서든지 클라우드 서비스를 받을 수 있는 환경을 제공하는 것을 말한다. 또한, 모바일 단말에서 처리해야할 작업 및 데이터 저장의 일부를 클라우드 컴퓨팅 환경으로 이동시켜 처리하고, 모바일 단말에서 처리결과를 보여주는 어플리케이션으로 볼 수 있다. 그림 1은 모바일 클라우드 컴퓨팅 구성도를 보여준다.

모바일 클라우드 컴퓨팅은 구성요소들이 모바일 단말기들에게 원하는 서비스를 제공하는 인프라스트럭처

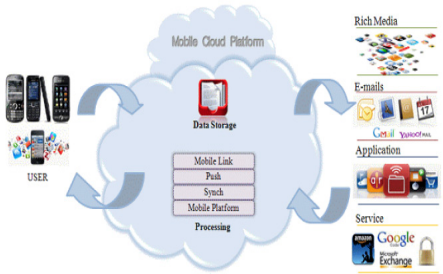


그림 1. 모바일 클라우드 컴퓨팅의 구성도[5]
Fig. 1. Mobile cloud computing architecture.

역할을 담당한다. 또한, 모바일 단말기 내에 내장되는 응용 소프트웨어들은 혼자만이 동작되는 단순 응용 프로그램이 아닌 모바일 클라우드 내에 있는 서버들의 지원을 받아 원하는 서비스를 실행하는 형태로 개발되고 있다.

대표적인 모바일 클라우드 서비스인 애플의 iCloud[20~21]는 응용 프로그램을 iPhone, iPod, iPad, Mac OS X, MS 윈도우 운영체제(vista이상 또는 윈도우7) 컴퓨터 등 다수의 장비 상에서 공유할 수 있으며, 그 외 이메일, 북마크 등을 포함한 다양한 형태의 데이터를 실시간으로 동기화할 수 있도록 구현되어 있다.

Google Drive[22]는 Gmail, 구글 캘린더 등의 서비스를 통합한 개인용 클라우드 서비스로서, 파일 처리 및 동기화에 필요한 저장 공간을 제공해준다. 구글 드라이브는 크롬 마켓 플레이스를 지원하며, 구글의 강력한 검색 엔진 기능을 포함하고 있다.

uCloud[23]는 KT에서 지원하는 모바일 클라우드 서비스로, 데이터를 실시간으로 저장할 수 있으며, 데이터 가용성을 위한 자동 백업을 지원한다. uCloud는 파일을 송수신 할 경우, 높은 수준의 암호화 기법을 사용하며 iPhone 및 안드로이드를 포함한 다양한 플랫폼에서의 호환성을 지원한다.

2.2 TPC-C 벤치마크

TPC-C[11-13]는 OLTP 벤치마크로 개발되었으며, 다양한 하드웨어와 소프트웨어들의 OLTP 성능을 비교하기 위한 척도로 사용되어왔다. TPC-C는 주문 입력(new order), 상품 배달(delivery), 비용 지불(payment), 주문 상태 확인(order status) 및 재고 수준(stock level) 등 5개의 트랜잭션으로 구성된다.

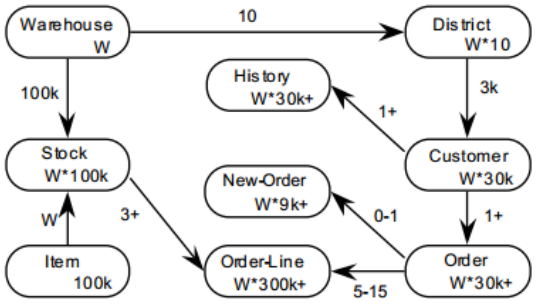


그림 2. TPC-C database table 스키마
Fig. 2. TPC-C database table schema.

표 1. 트랜잭션별 분포도[12]
Table 1. Transactional distribution.

Name of transaction	Share of all transactions
New order	≤ 45%
Payment	≤ 43%
Order status	≤ 4%
Delivery	≤ 4% (batch transaction)
Stock level	≤ 4%

그림 2에서 보여주듯이, TPC-C는 9개의 데이터베이스 테이블과 위에 설명한 5개의 트랜잭션으로 구성된다. 표 1은 TPC-C에서의 트랜잭션별 분포도를 보여준다.

본 논문에서는 자바 스윙으로 이루어진 OLTP 환경의 jTPC-C 벤치마크를 안드로이드 모바일 어플리케이션인 모바일 TPC-C로 변환하였으며, 이를 이용해 모바일 기기와 하둡 기반 클라우드 서버간의 데이터 액세스 중 발생하는 통신 오버헤드를 보다 객관적인 척도에 맞게 측정하고자 하였다.

Ⅲ. 모바일 TPC-C 설계 및 구현

3.1 수행페이지(Execution Activity) 구조

모바일 TPC-C 수행페이지는 데이터베이스 수행페이지, 터미널 수행페이지, 가중치 수행페이지, 컨트롤 수행페이지로 구분된다. 모바일 TPC-C는 수행페이지를 통해 실행에 필요한 데이터베이스 접속 정보와 트랜잭션 가중치 등을 입력받으며, 컨트롤 수행페이지에서 제공하는 실행 버튼을 통해 어플리케이션이 활성화 된다. 수행페이지 각각의 기능은 다음과 같다.

- 데이터베이스 수행페이지
  클라이언트가 접속할 데이터베이스를 설정해주는 역할을 한다. 위 수행페이지는 접속할 데이터베이스의 URL과 드라이버, 데이터베이스 접근에 필요한 사용자 이름과 비밀번호를 입력할 수 있는 창으로 구성되어 있으며, 입력된 정보를 사용하여 서버상의 데이터베이스와 접속이 이루어진다.

- 터미널 수행페이지
  데이터베이스에서 수행될 트랜잭션의 개수와 필요한 웨어하우스의 개수, 트랜잭션 실행 시간이나 TPT(Transaction Per Terminal)을 설정할 수 있는 창으로 이루어져있다.

● 가중치 수행페이지

TPC-C 트랜잭션별로 가중치를 부여할 수 있는 페이지이다. 일반적으로 표 1과 같이 주문입력 N/A, 비용 지불 43%, 주문 상태 확인 4%, 상품 배달 4%, 재고 현황 모니터링 4%로 가중치를 부여한다.

● 컨트롤 수행페이지

모바일 TPC-C 실행 버튼을 제공하며, 각각은 터미널 생성, 트랜잭션 시작, 트랜잭션 종료 등의 역할을 담당한다. 또한, 데이터베이스, 터미널, 가중치 수행 페이지들로부터 입력받은 정보를 토대로 서버의 데이터베이스에 접속하며, 트랜잭션을 실행시키고 결과 값인 tpmC(Transaction Per Minute type C)를 서버로부터 추출하는 기능을 수행한다. TpmC는 TPC-C 벤치마크 시나리오에 대한 1분당 최대 처리건수를 나타내는 수치이다.

그림 3은 모바일 TPC-C 수행페이지 구성도를 나타낸다. 데이터베이스, 터미널, 가중치 등의 수행 페이지에서 입력받은 정보는 어플리케이션 활성화를 위해 컨트롤 수행 페이지로 전달된다. 전달된 값에 오류가 없다면 create terminals 객체를 생성해 서버의 데이터베이스에 접속하게 된다.

터미널 생성작업이 오류 없이 완료되면 트랜잭션 시작 버튼이 활성화되며 클라이언트와 서버간의 트랜잭션 작업이 실행된다. 또한, 컨트롤 수행페이지는 트랜잭션 작업 완료시, 수행된 작업의 tpmC 수치를 표시해준다. TpmC 값은 기본적으로 트랜잭션을 실행하는 사용자 수에 반비례한다. 즉, 데이터베이스에 접속하는 사용자 수가 증가하면 같은 데이터베이스 테이블로의 동시 액세스가 증가하게 되므로 경쟁(contention)으로 인한 과부하가 발생하게 되어 tpmC 수치가 떨어지게 된다.

본 논문에서는 터미널 사용자 수를 변화시켜, 모바일 TPC-C와 클라우드 서버간의 성능 변화를 tpmC 값으로 표시하였다. 이는 모바일 어플리케이션 성능 향상을 위해 필요한 요소들을 향후 과제로 제시하는데 도움이 될 것이다.

3.2 모바일 TPC-C WAS

현재 안드로이드 모바일 기기는 보안상의 문제로 모바일에서 직접 서버 데이터베이스에 접근하는 것이 불가능하며, 이로 인해 데이터베이스가 존재하는 서버와 클라이언트 간 통신에 필요한 JDBC 프로토콜을 지원하지 않는다. 이에 본 연구에서는 모바일 기기와 클라우드 서버 사이에 웹 어플리케이션 서버인 WAS를 구현해 데이터 처리에 필요한 통신을 원활이 실행하도록 하였다.

WAS[17,18]는 클라이언트 요청을 받아 HTML이나 오브젝트를 HTTP 프로토콜을 이용해 전송해 주는 역할을 하며, 동시에 인터넷상에서 HTTP를 통해 사용자 컴퓨터나 장치에 어플리케이션을 수행해 줄 수 있는 소프트웨어 엔진이다. WAS는 동적 서버 콘텐츠를 수행하는 것으로 일반적인 웹 서버와 구별되며, 대부분이 자바 기반으로 자바 EE 표준을 수용하고 있다. WAS는 프로그램 실행 환경과 데이터베이스 접속을 담당하며, 어플리케이션들이 발생시키는 데이터베이스 관련 트랜잭션들을 관리한다.

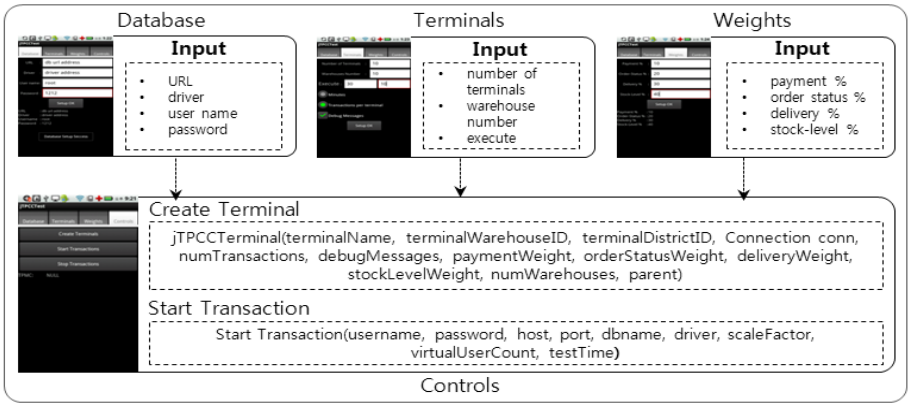


그림 3. TPC-C 어플리케이션 수행 페이지 구성도
Fig. 3. TPC-C application execution page.

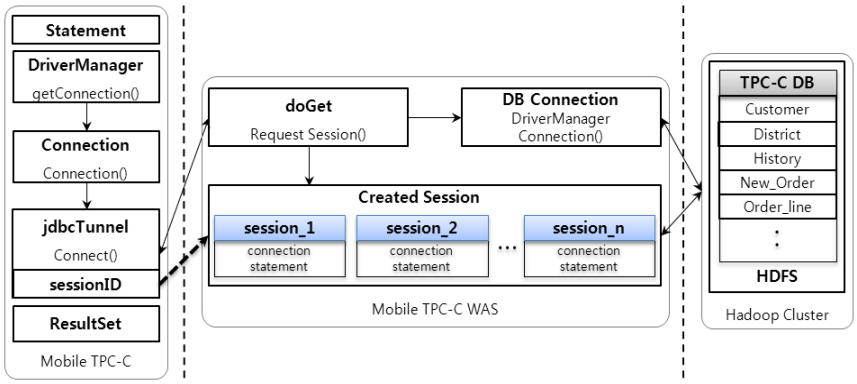


그림 4. 모바일 기기와 클라우드 서버간의 전체 통신 구조
Fig. 4. Communication structure between mobile devices and cloud server.

그림 4는 모바일 TPC-C WAS를 이용한 모바일 기기와 클라우드 서버간의 통신 구조를 보여준다. 기존의 JDBC는 데이터베이스 사용을 위해 DriverManager, Connection, Statement, Result Set을 포함한 네 개의 클래스를 제공한다.

본 연구에서는 기본적으로 제공하는 네 개의 클래스 뿐만 아니라 통신에 필요한 세션 유지를 위해 jdbcTunnel() 클래스를 추가하였다. 그 결과, 데이터베이스 접속을 위한 기존 실행 단계인 DriverManager → Connection은 연속적인 TPC-C 트랜잭션 수행에 필요한 세션 유지를 위해 DriverManager → Connection → jdbcTunnel로 변경된다. 또한, jdbcTunnel()의 Connect()가 호출되고, 이 객체에서 WAS 서블릿에 접속해 sessionID를 받으며, 이 정보는 jdbcTunnel 객체의 sessionID에 저장된다. WAS서블릿 세션에는 데이터베이스와 연결된 jdbcConnection 객체와 쿼리를 위한 statement 클래스가 저장되어 있다.

SessionID는 각각의 세션을 구분하기 위한 정수 값으로 사용되며, 각 세션들은 초기 접속 시 독자적인 sessionID를 할당 받는다. 최초 트랜잭션 쿼리가 수행될 때 할당받은 sessionID는 후속 쿼리 실행을 위한 통신 접속에 그대로 사용되고, 세션이 종료될 때, 종료 플래그를 전송하여 소멸되도록 하였다. 본 논문에서는 아파치 소프트웨어 재단에서 개발된 서블릿 컨테이너를 지원하는 WAS인 아파치 톰캣(Tomcat) 라이브러리를 사용했으며, 이를 모바일 TPC-C 트랜잭션 처리에 맞게 구현한 것이 특징이다[19].

표 2와 표 3은 앞서 설명한 모바일 기기와 클라우드 서버간의 접속과 트랜잭션 수행 과정을 의사코드

표 2. 데이터베이스 접속 알고리즘 의사코드
Table 2. Database connection algorithm.

```
Input:
  I(database connection info.), T(terminal#)
Output:
  sessionID
1  for n ← 0 to  T
2   if n∈T then
3     if connection == null then
4       input I
5       connect() ← jdbcTunnel
6       receive sessionID[n] from servlet
7     else
8       execute transaction using sessionID
9     end if
10   else
11     sessionID[n] = 0
12  end if
13 end for
14 return sessionID
```

(pseudocode)형태로 표현하였다. 표 2에서 I, T는 각각 데이터베이스 접속에 필요한 정보, 터미널 사용자 수를 나타내고, 표 3에서 W, M은 각각 웨어하우스 개수, 수행시간을 나타낸다. 모바일 TPC-C 어플리케이션에서 터미널을 생성하면 jdbcTunnel()을 통해 서블릿으로부터 세션을 할당받게 된다. 그 후 최초 접속에서 서블릿으로부터 sessionID를 터미널 사용자 수마다 할당받는다. 그리고 모바일 TPC-C 트랜잭션에서 발생하는 쿼리들은 할당받은 세션을 통해 클라우드 서버상의 데이터베이스와 접속하게 된다. 수행 중 데이터에 대한 변경사항을 취소하거나 정전 및 시스템 다운이 발생하는

표 3. 트랜잭션 수행 알고리즘 의사코드
Table 3. Transaction execution algorithm.

```
Input:
  W(warehouse#), M(exe. time)
Output:
  tpmC
1  while (M != 0)
2   if exception occurs then
3    rollback()
4   else
5     for each transaction
6      for each query
7      concatenate query statement using
                                  query.append()
8       execute query using stmt.executeQuery()
9        switch (query)
10        case select:
11            send result to mobile TPC-C
12        case default:
13              store result to database
14       end swich
15     commit()
16   end for
17  end if
18 end while
19 return tpmC
```

경우, rollback()을 호출하여 해당 트랜잭션에 의해 록(lock)이 걸린 모든 자원들에 대해 록을 해제하고 트랜잭션은 이전 상태로 돌아가게 된다. 또한, 모든 작업이 종료되었을 경우 commit() 메소드가 호출되며 측정된 tpmC 값이 반환된다.

Ⅳ. 성능 측정 및 분석

본 연구에서는 모바일 TPC-C와 하둡 기반 클라우드 서버간의 TPC-C 트랜잭션 처리 성능을 측정하였다. 또한, 데스크톱에 기존 jTPC-C의 자바 스윙를 설치하여 모바일 TPC-C와의 성능 비교에 사용하였다. 모바일 TPC-C의 경우 클라우드 서버와의 원활한 통신을 위해 3G가 아닌 M/Bit의 와이파이 환경에서 테스트를 수행하였다.

본 연구에서 성능 측정에 사용한 하둡 분산 파일 시스템 (HDFS)은 순수한 자바 파일시스템으로 여러 노드에 걸쳐 대용량의 파일을 분산, 저장할 수 있는 기능을 갖추고 있다. HDFS는 데이터 가용성을 위해 복제를 지원하며, 기본적으로 3개의 노드(2개는 같은 랙, 1개는 다른 랙)에 복사본을 저장한다. HDFS는 웹 브라우저나 다른 클라이언트를 통해서 분산된 데이터에 대한 접근을 허용할 수 있다[14-16].

표 4. 하둡 클러스터 마스터 노드 테스트 환경
Table 4. Hadoop cluster master node test environment.

OS	Ubuntu 10.10 (32Bit Linux)
CPU	Intel(R) Core(TM) i3 CPU 3.07GHz
RAM	Samsung DDR2 2Rx8 PC2-6400U 2GB
HDD	Seagate 500GB

표 5. 하둡 클러스터 슬레이브 노드 테스트 환경
Table 5. Hadoop cluster slave node test environment.

	Slave 1(Secondary Name Node)	Slave2
OS	Ubuntu 10.10 (32Bit Linux)	Ubuntu 10.10 (32Bit Linux)
CPU	Inter(R) Core(TM) i3 CPU	Inter(R) Core(TM)2 CPU 1.86GHz
RAM	Samsung DDR2 2Rx8 PC2-6400U 2B	Samsung DDR2 2Rx8 PC2-6400U 2B
HDD	ATA Hitachi HD 1.0TB	ATA ST3250410AS 250GB

표 6. 안드로이드 어플리케이션 테스트 환경
Table 6. Android application test environment.

기기명	Google NEXUS ONE
안드로이드 버전	Android 2.3.3(API 10)
CPU	1Ghz Qualcomm QSD8250(Single Core)
RAM/ROM	512MB/512MB

표 7. TPC-C 데이터베이스 테이블의 튜플 수
Table 7. The number of TPC-C database table tuples.

Customer	District	History	Item	New-order	Order_line	Orders	Stock	Warehouse
30000000	10000	30000000+	100000	9000000+	300000000+	30000000+	100000000	1000

표 4, 표 5은 클라우드 서버의 마스터와 슬레이브 노드 환경을 나타내며, 표 6는 모바일 TPC-C가 실행되는 안드로이드 기기 환경을 표시한다. 성능 측정을 위해 TPC-C 데이터베이스의 웨어하우스 SF(Scale Factor)를 1000으로 고정하고 트랜잭션의 수행 시간을 5분으로 제한했다. 표 7는 웨어하우스 SF를 1000으로 설정했을 경우, TPC-C에서 제공하는 9개의 테이블의 튜플(Tuple)수를 나타낸다. 각각의 테이블은 그림2에서 보는바와 같이 서로 연결되어 있으며, 웨어하우스의 개수

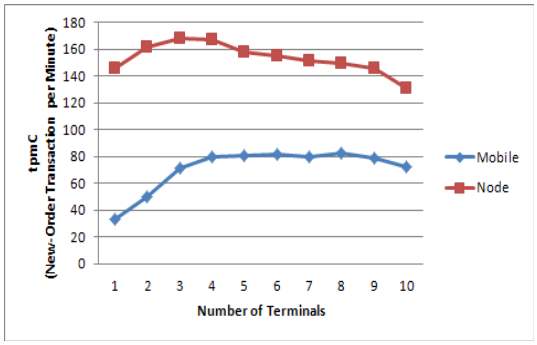


그림 5. 터미널 사용자수가 적은 수일 경우 모바일과 데스크톱 환경에서의 tpmC 비교
Fig. 5. TpmC Comparison between mobile and desktop with the small number of users,

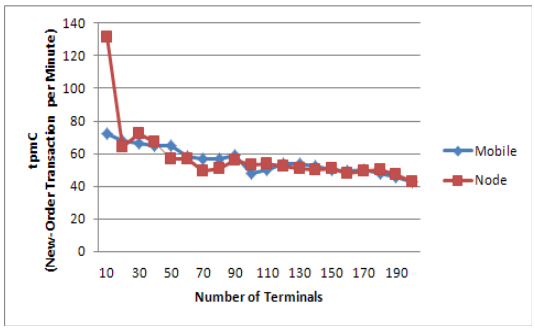


그림 6. 터미널 사용자수가 중간 수, 많은 수일 경우 모바일과 데스크톱 환경에서의 tpmC 비교
Fig. 6. TpmC Comparison between mobile and desktop with the medium number of users.

에 따라 테이블들의 튜플 수도 바뀌게 된다.

그림 5, 그림 6은 터미널 수를 각각 적은 수, 중간 수, 많은 수로 설정하여 tpmC 값을 기준으로 모바일 TPC-C와 데스크톱 환경에서의 jTPC-C의 성능 측정에 대한 비교 결과를 보여주고, 그림 7은 1부터 200까지의 터미널 사용자수를 설정하여 테스트한 tpmC 결과 값을 비교하여 보여준다.

그림 5에서 보는바와 같이 모바일 TPC-C 환경에서는 터미널 수가 8에서 최대 임계점을 형성하는 것을 보였으며, 최대 임계점을 기록하고 나서도 tpmC의 수치가 급격하게 떨어지지 않은 것을 보였다. 또한 데스크톱 환경에서의 성능측정 결과도 터미널 사용자 수와 tpmC 결과 값이 반비례 한다는 것을 보여준다. 데스크톱 환경에서는 터미널 사용자 수가 3에서 최대 임계점을 보이고 4부터 경쟁으로 인한 과부하로 성능 저하가 발생하는 것을 볼 수 있다.

그림 7에서 보는바와 같이 1단위의 터미널 사용자 수

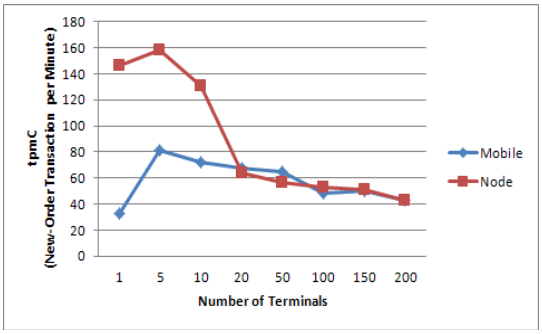


그림 7. 터미널 사용자수에 따른 모바일과 데스크톱 환경에서의 tpmC 비교
Fig. 7. TpmC Comparison as a function of terminal users on mobile and desktop

에서는 tpmC 수치가 2배 이상 차이가 나는 것을 볼 수 있다. 이는 모바일 기기의 한정된 하드웨어적인 측면과 네트워크 차이 때문이라고 추정할 수 있다. 그러나 터미널 사용자 수 20부터는 초기 네트워크 설정이 반복적으로 사용되므로 tpmC의 수치가 거의 동일하게 측정되었다. 모바일 기기는 하나의 트랜잭션이 클라우드 서버로 접속하기까지 5초정도의 수행시간을 소비하므로 전체 수행시간은 터미널 사용자 수에 비례하였다. 그 결과 모바일 기기에서 측정한 tpmC 수치는 평균적으로 데스크톱보다 낮은 것을 볼 수 있었으나, 네트워크를 포함한 하드웨어와 소프트웨어의 차이에도 불구하고 큰 성능 변화가 발생하지 않는 것을 볼 수 있었다. 이러한 결과는 최근 모바일 기기의 성능이 크게 발전하였음을 유추할 수 있는 부분이다.

또한, 모바일 TPC-C 테스트의 경우, 각각의 터미널 사용자가 모바일 기기부터 데이터베이스에 접속하는데 대략 5초가량의 시간이 소요되었다. 터미널 사용자 수를 데스크톱에서만큼 증가시키지 못했는데, 그 이유는 터미널 사용자 수를 100단위로 높일 경우 수행이 완료된 시점에서 데이터베이스에 남아있는 좀비 프로세스가 원활한 성능측정을 방해하였기 때문이다. 좀비 프로세스의 숫자는 터미널 사용자 수가 많아질수록 증가하였다.

V. 결 론

본 논문에서는 TPC-C 벤치마크를 안드로이드 모바일 TPC-C 어플리케이션으로 구현해 모바일 환경에서 클라우드 서버로 요청되는 다양한 종류의 트랜잭션들을

수행하고 성능을 측정함으로써, 최근 이슈화되고 있는 모바일 클라우드 컴퓨팅 환경에서 모바일 기기와 클라우드 서버 간에 보다 일반적인 성능 기준을 제시하려고 하였다. 모바일 기기는 데스크톱과 비교해 클라우드 서버로의 접속 시간이 현저히 떨어지지만 네트워크가 안정되는 시점에서는 데스크톱과 비슷한 성능을 보여주는 것을 알 수 있었다. 이를 통해 최근 스마트폰의 성능이 일반 PC와 비교해도 크게 뒤처지지 않을 만큼 큰 발전을 이루었다는 것을 유추할 수 있다. 이러한 결과들을 통하여 모바일 TPC-C를 활용하여 이종의 모바일 클라우드 컴퓨팅 상에서 데스크톱과 모바일 간의 성능 향상을 위한 연구에 활용될 수 있을 것이다.

향후과제로 우선 모바일 기기로부터의 접속 시 데이터베이스에서 발생하는 좀비 프로세스의 원인을 찾고 개선함으로써 보다 체계화된 성능측정을 보장할 것이다. 또한, 하둡 클러스터 상에서 제공하는 분산처리 프레임워크인 맵-리듀스를 이용해 어플리케이션 캐시 테이블을 구현함으로써 모바일 기기와 데스크톱간의 어플리케이션 성능 차이를 줄일 것이다.

REFERENCES

[1] B. Hayes, Cloud computing, “Communications of the ACM, Vol. 51, no. 7, pp. 9-11, July 2008.

[2] L. Wang, G. Laszewski, A. Younge, X. He, M. Kunze, J. Tao and C. Fu, “Cloud Computing: a Perspective Study,” New Generation Computing, Vol. 28, no. 2, pp. 137-146, April 2010.

[3] Z. Wei, G. Pierre and C.-H. Chi, “Cloud- TPS: Scalable Transactions for Web Applications in the Cloud”, IEEE Transac- tions on Services Computing, Vol. 5, no. 4, pp. 525-539, 2012.

[4] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg and I. Brandic, “Cloud computing and emerging IT platform: Vision, hype, and reality for delivering computing as the 5th utility,” Journal on Future Generation Computer Systems, Vol. 25, no. 6, pp. 599-616, June 2009.

[5] 윤용익, 김스베틀라나, “모바일 클라우드 컴퓨팅 기술 동향,”IT 기획 시리즈, 주간기술동향통권 1439호, pp. 32-34, 2010.

[6] 이강찬, “모바일 클라우드 개념과 기술 동향”, 한국정보통신기술협회, Special Report, 2012.

[7] A. Smailagic and M. Ettus, “System Design and Power Optimization for Mobile Computers,” in Proc. of IEEE Computer Society Annual Symposium on VLSI, pp. 10, August 2002.

[8] U. Varshney, “Pervasive healthcare and wireless health monitoring,” Journal on Mobile Networks and Applications, Vol. 12, no. 2-3, pp. 113-127, March 2007.

[9] C. Doukas, T. Pliakas and I. Maglogiannis, “Mobile Healthcare Information Management unitizing Cloud Computing and Android OS,” in Annual International Conference of the IEEE on Engineering in Medicine and Biology Society (EMBC), pp. 1037-1040, October 2010.

[10] M.T. Nkosi and F. Mekuria, “Cloud Computing for Enhanced Mobile Health Applications,” in Proc. of the 2nd IEEE International Conference on Cloud Computing Technology and Science, pp. 629, February 2011.

[11] Wikipedia, http://en.wikipedia.org/wiki/Transaction_Processing_Performance_Council, 2009.

[12] Transaction Processing Performance Council,“TPC benchmark C standard specification, revision 5,” 2006, http://www.tpc.org/tpcc/.

[13] Fujitsu Technology Solutions, “Benchmark Overview TPC-C”, Technical Report(2003), pp. 1-5.

[14] S. Ghemawat, H. Gobioff, S.-T. Leung, “The google file system,” ACM SIGOPS Operating Systems Review, Vol. 37, no. 5, pp. 29-43, December 2003.

[15] Hadoop, www.hadoop.apache.org/

[16] T. White, “Hadoop the Definitive Guide,” O’REILLY, 2009.

[17] Wikipedia, http://en.wikipedia.org/wiki/Web-_Application_server

[18] S. Li, “Tomcat for beginning web developers,” IBM developerWorks, 2005.

[19] Apache Tomcat, tomcat.apache.org/

[20] icloud, https://www.icloud.com/

[21] Wikipedia, wikipedia.org/wiki/iCloud

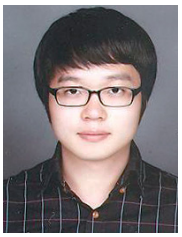
[22] google drive, drive.google.com/

[23] ucloud, home.ucloud.olleh.com/

저 자 소 개

장 한 을(정회원)
2010년 세종대학교 디지털콘텐츠 학사 졸업.
2012년 세종대학교 컴퓨터공학과 석사 졸업.
2013년~(주)데이터 스트림즈 근무
<주관심분야 : 분산파일시스템, 클라우드 컴퓨팅, ETL>

노 재 춘(정회원)-교신저자
세종대학교 컴퓨터공학과 교수
1985년 이화여자대학교 전자계산학과 학사 졸업.
1999년 Syracuse University 전산학과 박사 졸업.
<주관심분야 : 클라우드컴퓨팅, 파일시스템, 데스크탑 가상화>



김 병 문(정회원)
2012년 세종대학교 컴퓨터공학과 학사 졸업.
2013년 세종대학교 컴퓨터공학과 석사 재학 중.
<주관심분야 : 클라우드컴퓨팅, 데스크탑 가상화, 파일시스템>

이 지 은(정회원)
2012년 세종대학교 컴퓨터공학과 학사 졸업.
2013년 세종대학교 컴퓨터공학과 석사 재학 중.
<주관심분야 : 클라우드컴퓨팅, 데스크탑 가상화, 파일시스템>

박 성 순(정회원)
안양대학교 컴퓨터공학과 교수
㈜글루시스 대표이사
1984년 홍익대학교 전자계산학과 학사 졸업.
1994년 고려대학교 전산학과 박사 졸업.
<주관심분야 : 스토리지, 파일시스템, 클라우드컴퓨팅>