클라우드 컴퓨팅 환경에서 사용자 서비스의 시나리오 기반 테스트 케이스 생성

김종필*, 홍장의*
*충북대학교 컴퓨터과학과
e-mail: kimjp@selab.cbnu.ac.kr

Development of Scenario-based Test Cases for User Service in Cloud Computing Environment

Jong-Phil Kim*, Jang-Eui Hong*
*Dept of Computer Science, Chungbuk National University

요 약

클라우드 서비스는 네트워크 환경을 통해 사용자가 원하는 서비스를 장소에 구애받지 않고 수행될 수 있도록 지원하는 소프트웨어 응용의 일종이다. 이와 같은 클라우드 서비스의 개발과정에 서비스가 정확하게 실행된다는 것을 확인하는 것은 매우 중요하다. 그러나 신규 서비스 또는 새로운 패러다임의 클라우드 서비스를 개발하는 과정에서는 사용자 단말에서 서버까지 사용자가 원하는 서비스가 정확히 실행되는 가를 확인하는 것은 어려운 일이다. 왜냐하면 서비스 실행을 위해 다양한 경로에 존재할 수 있는 소프트웨어 컴포넌트가 올바르게 동작할 수 있는 가를 확인할 수 있어야 하기 때문이다. 본 연구에서는 이와 같이 클라우드 서비스의 개발과정에서 서비스 실행을 위한 경로 상에 존재하는 컴포넌트를 고려하는 시나리오 기반의 테스트 케이스 생성 기법을 제안한다.

1. 서론

클라우드 컴퓨팅이란 모든 소프트웨어 및 데이터가 클라우드라는 다수의 또는 대형의 컴퓨터 군집에 저장되고, 네트워크 접속이 가능한 PC나 스마트폰 등의 다양한 단말기를 통해 장소에 구애받지 않고 원하는 작업을 수행할 수 있는 웹 기반의 컴퓨팅 기술을 말한다.

이러한 클라우드 환경이 다양한 기술의 발전과 함께 진화함에 따라 사용자들은 다양하고 새로운 서비스의 출현을 기대하고 있다. 이러한 새로운 패러다임의 사용자 클라우드 서비스를 개발하는 과정에서 개발하고 있는 서비스가 올바르게 동작하는지에 대한 테스트를 수행하는 일은 중요하다[1,2,3]. 특히 사용자의 서비스가 네트워크로 연결된 다양한 기기나 시스템 소프트웨어를 경유하여 실행되기 때문에 이들이 올바로 동작하고 있는지, 또는 서비스 실행과정에 새롭게 개발되어야 하는 소프트웨어 컴포넌트는 없는지를 확인하는 과정이 요구된다[4,5,6]. 이를 지원하기 위해서 우리 정부에서는 클라우드 서비스의 테스트베드 센터를 운영하고 있다[7,8].

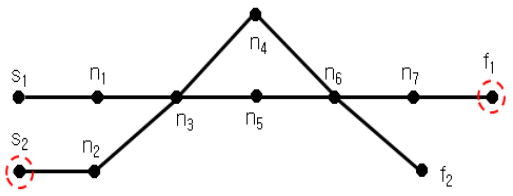
본 연구에서는 클라우드 서비스가 정확히 동작하는지를 테스팅하기 위한 테스팅 방안을 제시한다. 특히 기존의 독립적인 소프트웨어 테스팅 기법[9]으로는 클라우드 서비스의 테스팅이 어렵기 때문에 테스팅을 수행하는 통신망 환경에서의 시나리오를 도출하고, 시나리오 기반으로 테스트 케이스를 개발하는 방법에 주안점을 두었다.

2. 클라우드 서비스 테스트 시나리오

사용자가 요구하는 서비스를 테스트하기 위하여 시나리오의 유형을 정의하였다. 시나리오는 다음과 같이 3가지 유형의 시나리오로 분류한다.

(1) 글로벌 시나리오: 사용자 서비스가 End-to-End로 올바르게 실행되는지를 점검하기위한 시나리오.
(2) 로컬 시나리오: 서비스 실행 경로 상에서 특정 컴포넌트의 동작을 점검하기 위한 시나리오.
(3) 인터페이스 시나리오: 특정 서비스의 실행 과정에서 상호작용하는 메시지나 인자의 이벤트 추적을 위한 시나리오.

그림 1에서 보여주는 것과 같은 클라우드 서비스의 실행에 대한 경로를 그래프로 표현할 수 있다.



(그림 1) 클라우드 서비스 실행 환경 그래프

그림 1에서 S1과 S2는 서비스 시작점, f1과 f2는 서비스가 실행되는 노드를, 그리고 ni (i=1..7)는 서비스 실행경로 상에 존재하는 소프트웨어 컴포넌트 노드를 의미한다. 이들은 물리

적으로 이격된 것이 아닌 논리적인 개념이다.

3. 테스트 시나리오 생성

사용자 서비스를 테스트하기 위한 시나리오들은 다음과 같이 생성할 수 있다.

3.1 글로벌 시나리오(Global Scenario, GS)

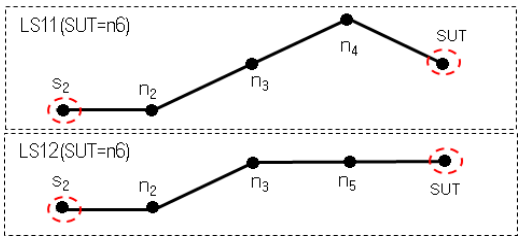
그림 1에 나타난 모든 글로벌 시나리오는 S1과 S2를 각각 시작점으로 f1과 f2에 다다를 수 있는 모든 경로들의 집합으로 총 8개의 시나리오를 얻을 수 있다. 그중에서 s2에서 f1까지 도달하는 GS는 다음과 같다.

GS1 = <s2, n2, n3, n4, n6, n7, f1>
GS2 = <s2, n2, n3, n5, n6, n7, f1>

3.2 로컬 시나리오(Local Scenario, LS)

LS는 각각의 GS에 대하여 정의한다. 앞의 GS1 글로벌 시나리오에 대하여 (그림 2)와 같은 LS를 생성할 수 있다. 이때 LS에 의해 테스트 되는 노드(System under Test)는 n2에서 n7까지의 임의의 노드가 될 수 있으며, SUT를 n6라고 가정할 때, 생성된 LS는 다음과 같다.

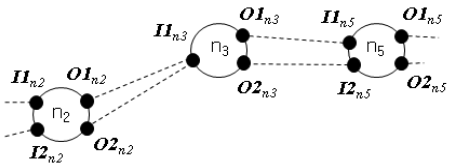
LS11(SUT=n6) = <s2, n2, n3, n4, SUT>
LS12(SUT=n6) = <s2, n2, n3, n5, SUT>



(그림 2) 로컬 시나리오 생성 예시

3.3 인터페이스 시나리오(Interface Scenario, IS)

IS는 LS의 인스턴스(instance)에 해당된다. 즉 LS의 실행 경로 상에서 어떤 기능이 선택되어지는 가를 명세하기 위함이다. 사용자 서비스가 어떤 메시지를 보내는 가에 따라 호출되는 대상 서비스의 모듈이 달라질 수 있기 때문에 IS가 필요하다. IS는 (그림 3)과 같은 인터페이스 구성도로부터 생성할 수 있다.



(그림 3) 노드별 인터페이스 구성도

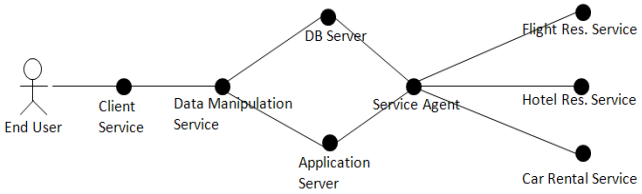
그림 3의 인터페이스 구성 개념으로부터 LS12 로컬 시나리오에 대한 IS는 다음과 같이 16개를 정의할 수 있다.

$IS1_{LS12}$ =<RI_{s2}, $I1_{n2}$, $O1_{n2}$, $I1_{n3}$, $O1_{n3}$, $I1_{n5}$, $O1_{n5}$, TI_{SUT}>
$IS2_{LS12}$ =<RI_{s2}, $I1_{n2}$, $O2_{n2}$, $I1_{n3}$, $O1_{n3}$, $I1_{n5}$, $O1_{n5}$, TI_{SUT}>
$IS3_{LS12}$ =<RI_{s2}, $I2_{n2}$, $O1_{n2}$, $I1_{n3}$, $O1_{n3}$, $I1_{n5}$, $O1_{n5}$, TI_{SUT}>
$IS4_{LS12}$ =<RI_{s2}, $I2_{n2}$, $O2_{n2}$, $I1_{n3}$, $O1_{n3}$, $I1_{n5}$, $O1_{n5}$, TI_{SUT}>
$IS5_{LS12}$ =<RI_{s2}, $I1_{n2}$, $O1_{n2}$, $I1_{n3}$, $O2_{n3}$, $I2_{n5}$, $O1_{n5}$, TI_{SUT}>
$IS6_{LS12}$ =<RI_{s2}, $I1_{n2}$, $O2_{n2}$, $I1_{n3}$, $O2_{n3}$, $I2_{n5}$, $O1_{n5}$, TI_{SUT}>
$IS7_{LS12}$ =<RI_{s2}, $I2_{n2}$, $O1_{n2}$, $I1_{n3}$, $O2_{n3}$, $I2_{n5}$, $O1_{n5}$, TI_{SUT}>
$IS8_{LS12}$ =<RI_{s2}, $I2_{n2}$, $O2_{n2}$, $I1_{n3}$, $O2_{n3}$, $I2_{n5}$, $O1_{n5}$, TI_{SUT}>
$IS9_{LS12}$ =<RI_{s2}, $I1_{n2}$, $O1_{n2}$, $I1_{n3}$, $O1_{n3}$, $I1_{n5}$, $O2_{n5}$, TI_{SUT}>
$IS10_{LS12}$ =<RI_{s2}, $I1_{n2}$, $O2_{n2}$, $I1_{n3}$, $O1_{n3}$, $I1_{n5}$, $O2_{n5}$, TI_{SUT}>
$IS11_{LS12}$ =<RI_{s2}, $I2_{n2}$, $O1_{n2}$, $I1_{n3}$, $O1_{n3}$, $I1_{n5}$, $O2_{n5}$, TI_{SUT}>
$IS12_{LS12}$ =<RI_{s2}, $I2_{n2}$, $O2_{n2}$, $I1_{n3}$, $O1_{n3}$, $I1_{n5}$, $O2_{n5}$, TI_{SUT}>
$IS13_{LS12}$ =<RI_{s2}, $I1_{n2}$, $O1_{n2}$, $I1_{n3}$, $O2_{n3}$, $I2_{n5}$, $O2_{n5}$, TI_{SUT}>
$IS14_{LS12}$ =<RI_{s2}, $I1_{n2}$, $O2_{n2}$, $I1_{n3}$, $O2_{n3}$, $I2_{n5}$, $O2_{n5}$, TI_{SUT}>
$IS15_{LS12}$ =<RI_{s2}, $I2_{n2}$, $O1_{n2}$, $I1_{n3}$, $O2_{n3}$, $I2_{n5}$, $O2_{n5}$, TI_{SUT}>
$IS16_{LS12}$ =<RI_{s2}, $I2_{n2}$, $O2_{n2}$, $I1_{n3}$, $O2_{n3}$, $I2_{n5}$, $O2_{n5}$, TI_{SUT}>

앞에서 생성된 16개의 시나리오를 기준으로 각각 테스트 케이스가 개발되어 서비스가 테스트 된다.

4. 사례 적용

일반적으로 널리 사용되는 클라우드 서비스의 대표적인 사례로써, TRSaaS (Travel Reservation Software as a Service)에 대하여 본 연구에서 제시하는 테스트 방법을 적용하였다. (그림 4)는 TRSaaS에 대한 실행환경 형상을 나타낸다.



(그림 4) TRSaaS의 실행환경 구성도

4.1 생성된 인터페이스 시나리오

TRSaaS에 대하여 생성된 시나리오의 일부를 제시하면 다음과 같다. 이때 SUT는 Service Agent로 정의하였다.

- 글로벌 시나리오
  GS1 = <End User, Client Service, Data Manipulation Service, DB Server, Service Agent, Flight Reservation Service>
  GS2 = <End User, Client Service, Data Manipulation Service, Application Server, Service Agent, Flight Reservation Service>
  :
- 로컬 시나리오
  LS11 = <End User, Client Service, Data Manipulation Service, DB Server, Service Agent>
  LS12 = <End User, Client Service, Data Manipulation Service, Application Server, Service Agent>
- 인터페이스 시나리오
  $IS1_{LS12}$ = <LoginClient, SendClientData, InputFlightResInfo, OutputFlightResData, ReceiveResReq, SendResData,

CreateRes, InvokeFlightService>

$IS2_{LS12}$ = <LoginClient, SendClientData, InputHotelResInfo, OutputHotelResData, ReceiveResReq, SendResData, CreateRes, InvokeHotelService>

$IS3_{LS12}$ = <LoginClient, SendClientData, InputCarResInfo, OutputCarResData, ReceiveResReq, SendResData, CreateRes, InvokeCarService>

:

4.2 테스트 케이스의 생성

테스트 케이스의 생성은 일반적인 인터페이스 기반의 블랙박스(black-box) 기법에 의하여 생성할 수 있다. 앞의 인터페이스 시나리오 $IS2_{LS12}$의 테스트 케이스는 <표 1>과 같다.

<표 1> $IS2_{LS12}$로부터 생성된 테스트케이스

No.	ID	Pass word	City	Hotel Name	Check-in Date	Check-out Date
T1	Kim	912345	LA	Hilton	2012/9/30	2012/10/01
T2	Lee	345481	NewYork	Grand	2012/10/02	2012/10/05
T3	Choi	145839	Seoul	Ramada	2012/10/07	2012/10/10
T4	Hong	456832	Cheongju	Ramada	2012/11/11	2012/11/14
:	:	:	:	:	:	:
T7	Kim	989468	Jeju	TheHotel	2012/11/30	2012/12/02

표 1에 주어진 테스트를 이용하여 개발하고자 하는 사용자 서비스를 테스트할 수 있다. 테스트 런을 통하여 얻는 결과는 테스트 형상에 존재하는 각각의 노드로부터 실행의 결과가 테스트 시작 노드로 전달되어 로그에 기록되게 된다. 테스트 로그는 다음과 같은 형식으로 나타날 수 있다.

```
Test Log for IS2LS12
Test Case T1
  I1n2: call activated !
  O1n2: <ER> Login OK <AR> Login OK
  I1n3: call activated !
  O2n3: <ER> ResDataReceived <AR> ResDataReceived
    :
```

5. 결론

본 연구에서는 클라우드 환경에서 다양하게 출현하는 사용자 서비스를 테스트하기 위한 방안에 대하여 제시하였다. 특히 본 연구에서는 기존 연구와 달리 계층구조의 테스트 시나리오를 정의하였다. 또한 본 연구에서는 계층별 시나리오 도출방법과 시나리오로부터의 테스트 케이스 생성 방법을 예제를 통해 제시하였다. 제시한 방법은 클라우드 서비스의 개발과정에서 점진적인 서비스의 정확성을 테스트하기 위한 방법으로 적용가능하며, 이를 통해 개발 서비스의 신뢰성을 향상시킬 수 있다. 향후의 연구는 서비스의 성능에 대한 테스팅 방법과 테스트 경로상에 Vacant Module이 존재하는 경우의 테스팅 기법 등에 대하여 연구할 것이다.

Acknowledgement

논문은 정부(교육과학기술부)의 재원으로 한국연구재단-차세대정보컴퓨팅기술개발사업(2011-0020523)과 기초연구사업지원(2011-0010396)을 받아 수행된 것임.

참고문헌

[1] L. Ribarov, I. Manova, S. Ilieva, Testing in a Service-Oriented World, Proceedings of the Int'l Conf. on Information Technologies, Sep. 2007, pp1-10

[2] L. Copeland, A Practitioner's Guide to Software Test Design, Artech House Publishers, 2004

[3] Ed Morris, et al., Testing in Service-Oriented Environments, CMU SEI, CMU/SEI-2010-TR-011, 2010

[4] Jerry Gao, Xiaoying Bai, and Wei-Tek Tsai, Cloud Testing- Issues, Challenges, Needs and Practice, Software Engineering : An International Journal, Vol. 1, No. 1, Sep. 2011, pp. 9-23

[5] S.P. Ahuja and S. Mani1, Availability of Services in the Era of Cloud Computing, Network and Communication Technologies, Vol. 1, No. 1; June 2012

[6] ISO/IEC 9126, Information Technology - Software Quality Characteristics and Metrics, ISO Standard, 2004.

[7] K. Priyadarsini, V. Balasubramanian and S. Karthik, Cloud Testing as a Service, Int'l Journal of Advanced Engineering Sciences and Technologies, Vol No.6, No.2, 2011, pp.173 - 177.

[8] 김진택, 클라우드 컴퓨팅 기술 및 표준화 동향, IT standards and Certification, Sep. 2009, pp.42-47

[9] J. Ryser and M. Glinz, "Using Dependency Charts to Improve Scenario-Based Testing", Proceedings of the 17th Int'l Conf. on Testing Computer Software, pp.1-10, 2000.