# sPAC (Web Service Performance Analysis Center): 성능 중심의 웹 서비스 조합 도구

장희정*, 송형기**, 이강선***

# sPAC(Web Services Performance Analysis Center): A performance-aware web service composition tool

Heejung Chang, Hyungki Song, Kangsun Lee

**Abstract**

Web services and their composition (web processes) are promising technologies to efficiently integrate disparate software components over various types of systems. As many web services are nowadays available on Internet, quality of services (QoS) and performance/cost become increasingly important to differentiating between similar service providers. In this work, we introduce sPAC (Web Services Performance Analysis Centre) and show how customers can benefit from sPAC to consider performance in composing and commercializing web services. sPAC 1) helps users to graphically describe the workflow of web services, 2) invokes web services to test out performance for light load conditions, 3) automatically converts the web services and the flow between them into a simulation model, 4) conducts extensive simulations for heavy load conditions and various usage patterns, and 5) reports analysis results and estimation data for the web services.

**Key Words:** Web Services and information management

* 명지대학교 컴퓨터공학과 대학원
** (주)네트빌 연구소
*** 명지대학교 컴퓨터공학과

## 1. 서론

웹 서비스는 SOAP(Simple Object Access Protocol), WSDL(Web Services Description Language), UDDI(Universal Description Discovery and Integration)라는 XML 기반의 공개 표준을 이용하여, XML 메시지를 웹을 통해 전송함으로써 이기종 시스템간의 상호 작용을 돕는 소프트웨어 시스템으로, 이종의 플랫폼, 시스템, 기관간의 업무 통합 환경을 제공하기 위한 필수 요소로 자리 잡고 있다[1]. 기업간 프로세스의 공유 및 통합의 확대로 웹 서비스의 이용이 증가하게 되었으며, 기업은 새로운 비즈니스를 빠르고 효율적으로 창출하기 위해 기존 웹 서비스를 조합(web service composition)하여 웹 프로세스(web process)를 구성하고 있다[2]. 이러한 웹 서비스의 활용이 증대함에 따라 웹 서비스의 QoS(Quality of Service)는 기업간 차별화를 위한 요건으로 그 중요성이 증가하고 있다. 웹 서비스의 QoS는 가용성(availability), 보안성(security), 신뢰성(reliability), 성능(performance)등과 같은 서비스의 비 기능적(non-functional) 요구사항을 나타낸다[3]. 이 중 성능은 네트워크의 상태나 동시 사용자 접속 패턴의 갑작스런 변화, 예외상황의 발생 등으로 인해 평가에 어려움이 따르는 품질 요소이다. 이로 인해, 웹 프로세스의 성능을 효과적으로 분석하기 위한 연구들이 진행되어 왔으며, 수학적 방법과 테스트 기반의 방법 등이 대표적이다. 수학적 성능 분석 방법은 웹 서비스의 QoS를 도출하기 위한 수학적 모델을 작성하고 계산에 의해 프로세스의 성능을 측정하는 방법으로 상대적으로 빠른 시간에 결과를 도출할 수 있으나 현실성이 떨어질 수 있다[4]. 테스트 기반 분석법은 실제 환경에서 웹 서비스를 호출하고 결과를 분석하는 방법으로 분석 결과에 대한 높은 신뢰성을 제공하나 분석에 소요되는 시간과 비용이 높다는 단점이 있다[5]. 이에 대한 대안으로 시뮬레이션 기반 분석법이 있다. 시뮬레이션 기반 분석법은 웹 프로세스에 대한 시뮬레이션 모델을 작성하고 실제 실행 없이 시뮬레이션을 통해 성능을 분석하는 방법이다. 시뮬레이션 기법은 분석에 소요되는 시간과 비용을 절감하고 다양한 환경에서의 성능분석이 가능하다. 그러나 시뮬레이션 모델과 환경 변수에 대한 타당성이 검증되지 않으면 분석 결과의 정확성을 보증하기 어렵다는 단점이 있다[6]. 따라서 본 연구에서는 시뮬레이션 기반의 분석법에 테스트 기법을 적용하여 분석에 소요되는 비용 및 시간을 절감하고, 분석 결과의 정확성을 높이고자 한다.

본 논문에서는 성능 중심의 웹 서비스 조합 환경인 sPAC(Web Services Performance Analysis Center)를 소개한다. sPAC은 분석의 효율성과 정확성을 높이기 위해 시뮬레이션 기반 성능 분석법과 테스트 기반 분석법을 결합하였다. 또한 시각적인 웹 프로세스 디자인 환경을 제공하고 프로세스의 성능을 분석하여 사용자가 분석결과를 토대로 웹 프로세스의 성능 개선을 위한 전략을 세울 수 있도록 한다.

본 논문의 구성은 다음과 같다. 2장에서 관련 연구를 통해 기존의 웹 프로세스 조합 및 QoS 보장에 관한 연구의 문제점에 대해 알아보고 3장에서 본 논문이 제안하는 성능 평가 기법과 성능 측정 기준을 살펴보고 이를 지원하는 도구인 sPAC의 구성에 대해 알아본다. 4장에서는 sPAC을 사용하여 프로세스를 조합하고 성능을 평가하고 5장에서 결론을 맺는다.

## 2. 관련 연구

웹 서비스란 네트워크를 통해서 이기종 시스템간의 상호 작용을 가능하게 하기 위한 소프트웨어 시스템을 말한다. 이러한 시스템들은 인터넷 프로토콜을 통해 전송된 XML 기반의 메시지를 이용하며, 다른 웹 서비스와 그들이 정의한 규정된 방식에 따라 상호 작용한다[7]. 기능적 제한이 있는 하나의 웹 서비스는 다른 기능의 여러 웹 서비스들과 다양한 형태로 조

합되어 웹 프로세스를 만들게 되며, 이를 통해 기업에서 요구하는 다양한 기능들을 인터넷을 통해 적절한 웹 서비스 제공자로부터 찾을 수 있고 실시간으로 조합될 수 있다. 따라서 개발에 소요되는 비용과 시간을 줄이고 사용자의 요구사항을 만족시키는 유연한 조합이 가능하다. 예를 들어, '도서검색', '장바구니', '결제', '배송요청' 등의 웹 서비스를 조합하여 '도서구매' 웹 프로세스를 구성할 수 있다. 웹 프로세스의 구축을 위한 연구 및 개발은 현재도 활발히 이루어지고 있으며 웹 서비스 조합 명세를 위한 언어(flow language)로 WSFL[8], XLANG[9], BPEL4WS[10] 등이 발표되었다. 그러나 이러한 언어들은 웹 서비스 조합 시 QoS를 고려하기 위한 지원이 없거나 미비하여 사용자가 요구하는 품질 수준을 보장할 수 없다. 따라서 이를 보완하기 위한 연구가 요구되고 있다.

웹 서비스의 QoS는 다양한 품질 요소들의 조합으로 나타낼 수 있으며 주요 품질 요소는 다음과 같다[3].

- 가용성(Availability): 웹 서비스가 존재하는지 또는 즉시 사용가능한 상태인지의 측면에서의 품질을 의미하며 웹 서비스 전체 가동 시간 중 실제로 사용자가 사용할 수 있는 시간의 비율로 나타낸다.
- 성능(Performance): 응답시간, 처리량 등으로 나타낼 수 있는 수행 성능. 응답시간은 요청을 보내고 응답을 받는데 걸린 시간으로, 처리량은 지정된 시간동안 처리할 수 있는 웹 서비스 요청 수로 나타낼 수 있다.
- 신뢰성(Reliability): 사용자가 요청한 전체 요청 중 올바른 응답의 비율로 나타내며 서비스의 품질을 유지할 수 있는 정도를 의미한다.
- 보안성(Security): 웹 서비스에 관련된 당사자들(Parties)의 인증, 메시지 암호화, 그리고 접근 제어를 제공함으로써 기밀성과 부인 봉쇄(Non-Repudiation)를 제공하는 웹 서비스의 품질을 의미한다.

웹 서비스의 QoS 관리는 크게 서비스 제공자의 관점과 서비스 요청자의 관점으로 나누어 생각할 수 있다[11]. 서비스 제공자 관점에서의 QoS에 대한 연구는 Shuping Ran[12]등에 의해 이루어졌다. Shuping Ran과 Peter Farkas[13]는 웹 서비스의 품질에 대한 검색이 가능하도록 UDDI를 개선하여 웹 서비스 조합 시에 QoS를 고려할 수 있는 웹 서비스 발견 모델(Web services discovery model)을 연구하였고 Tao Yu[14]는 웹 서비스 서버를 위한 QoS 브로커를 통해 웹 서비스의 QoS 정보를 제공하기 위한 알고리즘과 프레임워크를 제안하였다. 그러나 이러한 연구들은 현재 확립된 웹 서비스 표준 기술만으로는 실제 적용이 어려운 단점이 있다. John A. Miller [15]는 서비스 요청자의 관점에서 웹 서비스의 성능을 평가하기 위한 방법을 연구하고 이를 지원하는 도구인 SCET(Service Composition and Execution Tool)를 개발하였다. SCET는 웹 서비스 표준 기술을 그대로 사용하고 서비스 사용자가 손쉽게 조합된 웹 서비스의 성능을 평가하기 위한 환경을 제공한다. SCET는 시뮬레이션 모델의 구축과 실행을 통해 웹 서비스의 성능을 평가한다. 이 경우 시뮬레이션 모델과 사용되는 파라미터에 대한 검증이 이루어지지 않으면 평가 결과의 정확성이 떨어질 수 있다. 웹 서비스의 실제 실행을 통한 테스트 기반 성능 분석법[5]은 분석 결과에 높은 신뢰성을 제공하나 소요되는 시간과 비용이 높고 요구되는 시스템 자원이나 테스트 가능한 동시사용자의 제약 등이 문제점으로 발생할 수 있다.

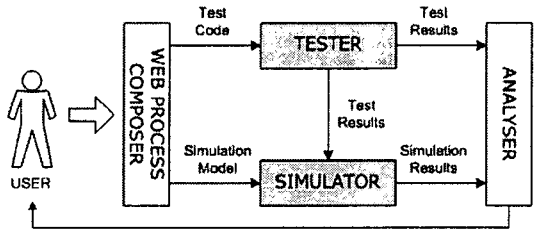
## 3. sPAC (Web Services Performance Analysis Center)

sPAC은 서비스 사용자를 위한 웹 서비스 성능 평가·예측 도구로 다음의 특징을 갖는다.

- 시뮬레이션 모델의 자동생성: 사용자는 UML의 Activity Diagram을 이용하여 웹 프로세스를 구성한다. Activity Diagram은 변환 규칙[16]에 의해 대등한 시뮬레이션 모델로 자동 변환되어 시뮬레이션 모델의 타당성을 높이고 성능 평가의 정확성을 증가시킬 수 있다.
- 테스트 결과를 적용한 시뮬레이션 수행: 제한된 동시 사용자 조건에서 실행되었을 경우의 테스트 결과를 시뮬레이션 모델의 파라미터로 자동 설정하여, 과부하(heavy load) 조건에서 시뮬레이션을 통한 성능 예측 시 결과의 정확도를 높이도록 한다.

### 3.1 성능 평가 기법

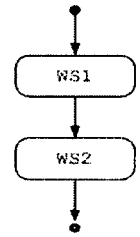
<그림 1>은 sPAC의 성능 분석 과정을 도식화 한 것이다.




<그림 1> sPAC 성능 분석 기법

사용자는 UML의 Activity Diagram을 사용하여 웹 프로세스의 조합을 정의한다. Activity Diagram은 각각의 서비스를 노드(node)로, 실행의 흐름을 링크(link)로 표현하며 분할(fork), 합병(join) 등의 표현을 통해 다양한 실행 조건을 표현한다. 웹 프로세스가 구성되면 동적으로 웹 서비스들을 호출, 실행시키고 저부하(low load) 조건에서 DRT(Dissected Response Time)와 TRT(Traced Response Time)를 측정하여 이를 데이터베이스에 저장한다. DRT와 TRT는 본 논문에서 제안하는 성능 측정 기준으로 3.2절에서 설명한다.

테스트가 완료되면 웹 프로세스를 위한 시뮬레이션 모델을 자동으로 생성한다. 시뮬레이션 모델은 자바 기반의 시뮬레이션 패키지인 Simjava[17]로 작성되었다. Simjava 시뮬레이션은 쓰레드(thread)를 통해 개별적으로 실행되는 엔티티(entity)와 각 엔티티들을 연결하는 포트(port)로 구성된다. Activity diagram의 노드(node)와 링크(link)는 Simjava에서 엔티티와 포트로 변환되어 표현된다. 모델이 작성되면 앞서 수행된 테스트 결과를 분석하여 단일 요청에 대한 평균 응답시간과 분산을 구하고 이를 적용하여 시뮬레이션을 수행한다. <그림 2>는 sPAC의 Simjava 모델로 변환된 웹 프로세스를 보여준다.




(a) Activity Diagram

```
// Modeling WS1
class WS1 extends Sim_entity {
   // private variables
   ..
   public WS1(String name, int index, int state) {
      super(name);
      this.index = index;
      this.state = state;
      out = new Sim_port("out");
      add_port(out);
   }
   public void body(){
      Sim_event ev = new Sim_event();
      int i;
      for(i = 0; i < simulation_time; i++){
         sim_schedule(out, normal(2.0, 0.05), 0);
         sim_wait(ev);
         sim_hold(normal(10.0, 0.03));
      }
   }
}
// Modeling WS2
class WS2 extends Sim_entity {
   // Similar to WS1
   ..
}
class Example1 {
   public static void main(String args[]){
      Sim_system.initialise();
      Sim_system.add(new WS("Sender", 1,
                     WS1.SRC_OK));
      Sim_system.add(new WS2("Receiver", 2,
                     WS2.WS2_OK));
      Sim_system.link_ports("Senter", "out",
                     "Receiver", "in");
      Sim_system.run();
   }
}
```

Behavior History Database (from Test-based Analysis)

(b) Simjava Simulation Model

<그림 2 > Simjava 시뮬레이션 모델

예를 들어, <그림 2>의 (a)와 같이 activity diagram으로 표현된 웹 프로세스는 (b)와 같은 Simjava 시뮬레이션 모델로 변환되며 이때, 앞서 수행된 테스트 결과가 적용된다. 마지막으로, DRT, TRT와 TPM (Transaction per Minute) 결과에 대한 보고서를 생성한다. 생성된 보고서는 그래프와 함께 제공되며 구성된 웹 프로세스의 성능 만족 여부를 판단할 수 있도록 한다.

### 3.2 성능 측정 기준

DRT(Dissected Response Time)와 TRT (Traced Response Time)는 본 논문이 제안하는 성능 측정 기준이다. DRT는 *Network Time(N), Messaging Time(M), Service Time(S)*의 세 가지 요소로 구성되며 단일 웹 서비스 s에 대하여 식(1)과 같이 정의된다.

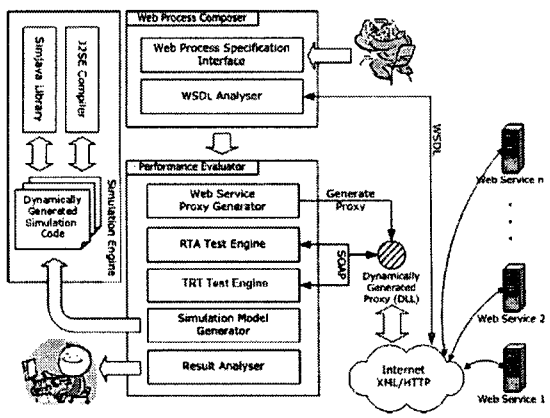
$$T(s) = N(s) + M(s) + S(s) \qquad (1)$$

*Network Time*은 웹 서비스 제공자와 사용자 사이의 네트워크의 대역폭, 트래픽, 장비의 성능에 의해 결정되는 지연 시간을, *Messaging Time*은 SOAP 메시지를 처리하기 위해 서비스 제공자에 의해 소요되는 시간을 의미한다. SOAP은 XML 기반의 프로토콜로 메시지 크기가 SNMP[18] 등의 이진(binary) 프로토콜에 비해 크기 때문에 이에 대한 처리 시간을 고려하여야 한다. *Service Time*은 웹 서비스에서 요청을 처리하기 위해 소요되는 시간으로 웹 서비스의 운영 시스템이나 프레임워크, 하드웨어의 성능, 비즈니스 로직의 효율성 등에 의해 좌우된다.

웹 프로세스가 웹 애플리케이션의 형태로 사용자에게 서비스를 제공하게 되면 각각의 서비스에는 동시 사용자 증가에 따른 성능 저하가 발생할 수 있다. TRT는 Java 쓰레드(Thread)로 가상의 동시사용자를 발생시켜 동시 사용자의 증가에 따른 웹 프로세스의 성능을 평가한다. TRT는 시간과 시스템 자원이 높게 요구되며, 메모리나 네트워크의 조건, 웹 서비스의 호스트 컴퓨터의 프레임워크 등에 의해 테스트 가능한 최대 동시 사용자의 수가 한정되어 있다. 이러한 제약을 극복하기 위하여 3.1절에서 설명한 시뮬레이션 기반 성능 평가 방법을 적용하였다. 제안된 방법은 실제 시스템의 자원이나 웹 서비스의 호출 없이 시뮬레이션을 통해 TRT가 평가된다. 또한 결과의 정확성을 높이기 위해 실제 DRT 테스트 결과를 적용하여 시뮬레이션을 수행하고 과부하 조건에서의 프로세스의 성능을 예측한다.

### 3.3 sPAC 아키텍처

<그림 3>은 sPAC의 아키텍처를 보인다. 주요 구성 요소는 다음과 같다.




<그림 3> sPAC 시스템 아키텍처

#### 3.3.1 Web Process Composer

*Web Process Composer*는 UML의 activity diagram을 통한 웹 프로세스의 명세와 테스트와 시뮬레이션을 위한 환경을 설정한다. *Web Process Specification Interface*는 웹 프로세스를 손쉽게 조합하기 위한 그래픽 기반의 환경을 제공한다. 또한 조합된 웹 프로세스의 성능에 대한 테스트 및 시뮬레이션을 위한 최소·최대 동시 사용자, 테스트와 시뮬레이션

횟수 등의 환경변수를 설정할 수 있도록 한다. 사용자는 *UDDI Search Module*을 통해 기존의 웹 서비스를 검색하여 프로세스를 구성할 수 있고 구성된 프로세스의 동적 실행을 위해 *WSDL Analyser*가 웹 서비스의 WSDL 파일을 분석한다.

3.3.2 Performance Evaluator

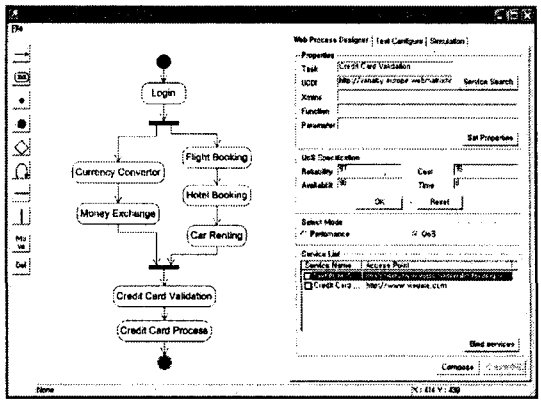
*Performance Evaluator*는 DRT와 TRT 테스트를 수행하고 분석 및 예측 결과를 보고서로 작성한다.

*Web Service Proxy Generator*는 Microsoft. NET 프레임워크의 Common Language Runtime (CLR) 및 C# 언어를 통해 웹 서비스의 실행 시 동적으로 프록시를 생성한다. *Test Engine*은 저부하 환경에서의 성능 분석을 위한 DRT 및 TRT 테스트를 수행한다. *Simulation Model Generator*에 의해 주어진 웹 프로세스에 대한 시뮬레이션 모델이 생성되면 *Simulation Engine*을 통해 시뮬레이션이 수행되고 과부하 환경에서의 프로세스 성능을 분석·예측한다. 마지막으로 *Result Analyser*를 통해 성능 분석 및 예측 결과에 대한 보고서가 생성된다. 사용자는 생성된 보고서를 통해 DRT 결과와 테스트 및 시뮬레이션 기반의 TRT 결과를 그래프로 확인할 수 있으며 TPM(Transaction Per Minute)등의 세부적인 성능 정보를 확인할 수 있다.

## 4. 실험 및 결과

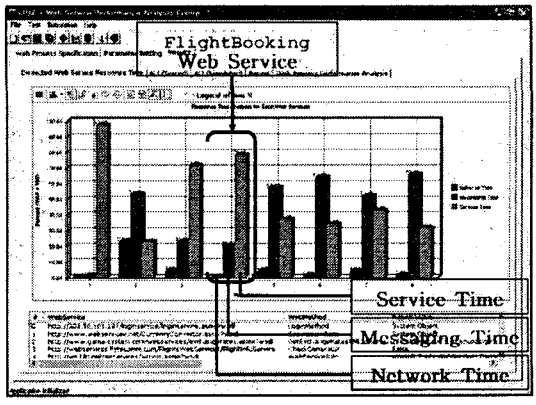
본장에서는 "My Travel Planner" 예제를 통해 웹 서비스 사용 고객이 sPAC을 이용하여 어떻게 성능을 고려한 웹 프로세스 조합을 수행할 수 있는지를 보인다.

My Travel Planner는 여행을 위한 비행기 예약, 숙소 예약, 자동차 렌트, 환전 등의 서비스와 카드 결제 등의 서비스를 제공하게 되며 현재 웹상에서 운영되고 있는 서비스들로 구성될 것이다. <그림 4>와 같이 사용자는 UDDI를 검색하여 서비스를 찾고 시각적으로 웹 프로세스를 구성할 수 있다.

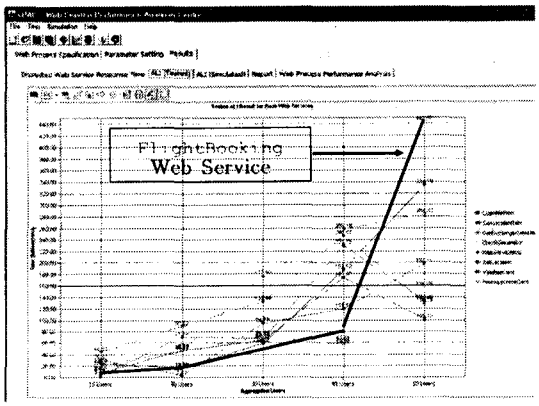



<그림 4> 웹 프로세스의 명세

<그림 6>의 TRT 테스트 결과를 통해 동시 사용자가 증가하게 되면 FlightBooking 서비스에서 병목현상이 유발될 수 있음을 알 수 있다. 또한 <그림 5>의 DRT 테스트 결과를 통해 FlightBooking 서비스의 응답 시간을 결정짓는 요소는 service time(78.89%) 임을 알 수 있다. 따라서 사용자는 FlightBooking 서비스의 비즈니스 로직(business logic)이나 하드웨어를 교체함으로써 전체 프로세스의 성능을 향상 시킬 수 있음을 발견하게 된다.

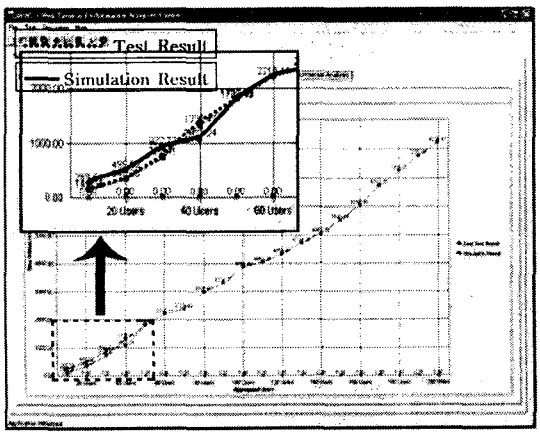



<그림 5> DRT 테스트 결과



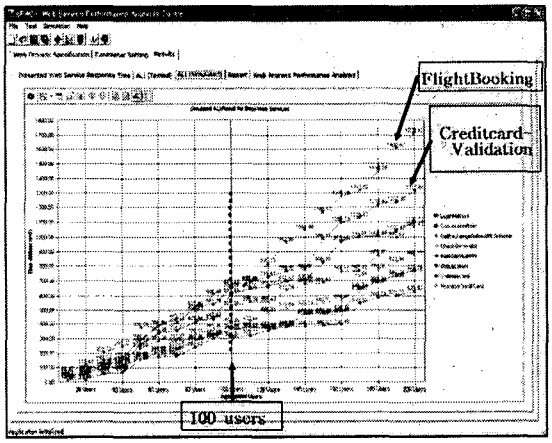

<그림 6> TRT 테스트 결과

<그림 7>은 시뮬레이션을 통한 10~200명의 동시 사용자 증가에 따른 성능 예측 결과를 보이고 있다. 동시 사용자가 10~50명으로 변화할 때의 테스트 결과를 시뮬레이션에 반영함으로써 50~200명의 동시 사용자 증가에 따른 성능 예측 결과의 정확성 높일 수 있었다.




<그림 7> 시뮬레이션의 정확도와 예측 결과

<그림 8>은 시뮬레이션을 통해 동시 사용자의 증가에 따른 각 서비스의 TRT 테스트를 수행한 결과이다. 200명으로 동시 사용자가 증가하면 FlightBooking과 CreditcardValidation 서비스에서 병목현상이 발생할 수 있음을 확인할 수 있다. 또한 "My Travel Planner" 프로세스의 경우 100명 이상의 동시 사용자가 발생할 경우 전체 응답시간이 급격히 증가함을 볼 수 있다. 사용자는 이러한 성능 평가 자료를 통해 프로세스의 성능을 미리 예측할 수 있고 이를 통해 적절한 하드웨어의 구성이 가능하다.




<그림 8> 시뮬레이션 기반 TRT 테스트 결과

## 5. 결론

본 논문에서는 성능 중심의 웹 서비스 조합 도구인 sPAC를 소개하였다. 제안된 도구를 통해 기존의 웹 서비스를 검색하여 프로세스를 구성하고 이에 대한 성능을 미리 평가하여 사용자에게 알림으로써 사용자의 QoS 요구에 만족하는 서비스를 효율적으로 조합할 수 있었다. 향후 가용성, 신뢰성, 보안 등의 QoS 요소에 대한 분석 기법뿐 아니라 이러한 요소들이 복합적으로 평가될 수 있는 분석법에 대한 연구를 수행할 계획이다.

## 참고문헌


[1] Eric Newcomer, 「Understanding Web Services: XML, WSDL, SOAP and, UDDI」, Addison-Wesley, 2002

[2] IBM Korea, "웹 서비스 요소기술", http://www-903.ibm.com/kr/software/kr/element/element.html, 2003

[3] Anbazhagan Mani, Arun Nagarajan, "Understanding Quality of Service for Web Services", http://www-106.ibm.com/developerworks/library/ws-quality.html, 2002

[4] Daniel A. Menasce and Virgilio A.F. Almeida, 「Capacity Planning for Web Services: Metrics, Models, and Methods」, Prentice-Hall, 2002

[5] J.D. Meier, Srinath Vasireddy, Ashish Babbar, Alex Mackman, 「How To: Use ACT to Test Web Services Performance」, Microsoft Developer Network, Microsoft Corporation, 2004

[6] Gregory Silver, John A. Miller, Jorge Gardoso, Amit P. Sheth, "Web service technologies and their synergy with simulation", In Proc. of the 2002 Winter Simulation Conference, pp.606 - 615

[7] W3C, "Web services Architecture Requirements", W3C Working Draft, http://www.w3.org/TR/2002/WD-wsa-reqs-20020429

[8] Frank Leymann, "Web Services Flow Language (WSFL 1.0)", http://www-3.ibm.com/software/solutions/Webservices/pdf/WSFL.pdf, 2001

[9] Satish Thatte, "XLANG: Web services for business process design", http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm, 2001

[10] Tony Andrews, Francisco Curbera, and et. al, "Specification: Business Process Execution Language for Web Services Version 1.1", http://www- 128.ibm.com/developerworks/library/ws-bpel, 2003

[11] Daniel A. Menasce, "QoS Issues in Web services", IEEE Internet Computing, Nov. 2002, pp. 72-75

[12] Shuping Ran, "A model for Web services discovery with QoS", ACM SIGecom Exchanges, Volume 4, Issue 1, Spring, 2003, pp. 1-10

[13] Peter Farkas, Hassan Charaf, "Web Services Planning Concepts", Journal of WSCG, Vol.11, No.1, 2003, ISSN 1213-6972

[14] Tao Yu, Kwei-Jay Lin, "The Design of QoS Broker Algorithms for QoS-Capable Web Services", International Journal of Web Services, Vol. 1, No. 4, 2004, pp. 17-24

[15] Senthilanand Chadrasekaran, John A. Miller, Gregory Silver, I. Budak Arpinar, and Amit P. Sheth, "Composition, Performance analysis and Simulation of Web services", Technical Report UGA-CS-TR-02-005, Dept. of Computer Science, University of Georgia, 2002

[16] Hyunggi Song, "sPAC: Web Services Performance Analysis Center", Master Thesis, Dept.. of Computer Engineering, MyongJi University, Korea, 2004

[17] Fred Howell, Ross McNab, "Simjava Library", http://www.dcs.ed.ac.uk/home/hase/simjava, 1996

[18] "SNMP", http://www2.rad.com/networks/1995/snmp/snmp.htm, 1995



주 작 성 자 : 장 희 정

논문투고일 : 2005. 06. 30

논문심사일 : 2005. 07. 11(1차), 2005. 07. 18(2차), 2005. 07. 21(3차)

심사판정일 : 2005. 07. 21

## 저자소개

장희정
2001 명지대학교 공과대학 컴퓨터공학과 학사
2003 명지대학교 공과대학 컴퓨터공학과 석사
현재 명지대학교 공과대학 컴퓨터공학과 박사과정
관심분야: Web Service, Medical Informatics, 컴퓨터시뮬레이션

송형기
2002 명지대학교 공과대학 컴퓨터공학과 학사
2005 명지대학교 공과대학 컴퓨터공학과 석사
현재 네트빌연구소(주) 연구원
관심분야: Web Service, SOA, 컴퓨터시뮬레이션

이강선
1992 이화여자대학교 공과대학 전자계산학과 학사
1994 이화여자대학교 공과대학 전자계산학과 석사
1998 미)University of Florida, Gainesville, Ph. D
현재 명지대학교 공과대학 컴퓨터공학과 교수
관심분야: Web Service, 컴퓨터시뮬레이션