비동기 웹 서비스 기반 계산 시스템의 구현

박재훈○ 박여삼 유재규 정영식 한성국
원광대학교 컴퓨터공학과
{pjh98○, maronie, jkyoo82, ysjeong, skhan}@wku.ac.kr

Implementation of Calculation System based on Asynchronously Web Service

Jaehun Park○ Yeosam Park Jaekyu Yoo Youngsik Jeong Sungkook Han
Wonkwang University Dept of Computer Engineer

요 약

현재의 웹 서비스 표준 기술은 XML, SOAP, WSDL, 그리고 UDDI를 중심으로 매우 기본적인 상호작용 모델을 가지고 있다. 위의 기술들을 이용해 단일 웹 서비스에 대한 정의와 비즈니스 레지스트리에 등록된 웹 서비스를 이용할 수 있다. 현재 일반적으로 개발되고 사용되고 있는 웹 서비스는 사용자의 요청이 발생하면 작업처리의 결과를 반환할 때까지 기다리는 동기적인 방식을 사용하고 있다. 즉, 사용자가 처리요청을 하면 그 결과가 반환될 때까지 기다리는 동안 손실이 발생하게 된다. 본 논문에서는 동기식 웹 서비스로 인해 발생할 수 있는 손실을 최소화 할 수 있는 비동기 웹 서비스를 시스템에 접목한다. 간단한 사칙연산을 동시에 비동기 방식으로 호출하는 웹 서비스의 구현을 통해 다른 영역에도 적용할 수 있는 기반 시스템을 구축한다.

1. 서 론

웹은 단순한 마크업 언어인 HTML을 사용하여 정보 자원을 다양한 형태로 표현하고 간편하게 전달할 수 있는 방법을 제공한다. 이는 인터넷 기반의 정보를 획기적으로 관리할 수 있는 계기가 되었다. 또한 풍부한 인프라와 용이한 확장성으로 인해 기존의 서비스들을 웹 환경으로 이전하려는 시도가 계속 있었지만 기존의 서비스들의 유기적인 수행을 위한 문제에 봉착한다. 무엇보다도 현재의 컴퓨팅 환경은 이기종이 복합적으로 존재하고 다양한 종류의 어플리케이션, 프로토콜, 포맷들이 혼재하는 매우 복잡한 상태이기 때문에 이들을 하나로 통합하기 위한 솔루션은 정형화된 규약을 정하는 것이다. 예를 들면 규약에는 원격지에 있는 서비스 객체나 API를 사용하는 방법이 정의되어 있다. 이에 웹의 개념이 확장되면서 단순한 정보 인프라로서의 역할을 넘어 기존 정보 인프라 환경에서 제공할 수 없었던 다양한 서비스를 제공이 가능한 웹 서비스의 개념이 등장하였다. [1]

현재의 웹 서비스 표준 기술은 XML, SOAP, WSDL, 그리고 UDDI를 중심으로 매우 기본적인 상호작용 모델을 가지고 있다. 위의 기술들을 이용해 단일 웹 서비스에 대한 정의와 비즈니스 레지스트리에 등록된 웹 서비스를 이용할 수 있다. 웹 서비스는 일반적인 프로그래밍 언어에서의 함수로 볼 수 있다. 단, 표준 기술을 이용해 플랫폼에 독립적이고 분산되어 있는 서비스들을 사용할 수 있다. 현재 일반적으로 개발되고 사용되고 있는 웹 서비스는 사용자의 요청이 발생하면 작업처리의 결과를 반환할 때까지 기다리는 동기적인 방식을 사용하고 있다. 즉, 사용자가 처리요청을 하면 그 결과가 반환될 때까지 기다리는 동안 손실이 발생하게 된다. [2]

본 논문에서는 동기식 웹 서비스로 인해 발생할 수 있는 손실을 최소화 할 수 있는 비동기 웹 서비스 기반의 시스템을 구현한다.

2. 관련 연구

2.1 웹 서비스

현재의 컴퓨팅 환경은 이기종이 복합적으로 존재하고 다양한 종류의 어플리케이션, 프로토콜, 포맷들이 혼재하는 매우 복잡한 것이기 때문에 이들을 하나로 통합하기 위해서는 정형화된 규약이 필요하다. 예를 들면 규약에는 원격지에 있는 서비스 객체나 API를 접근해 사용하는 방법이 정의되어 있다.

웹 서비스는 표준 인터넷 프로토콜을 사용하여 접속할 수 있는 프로그램화가 가능한 어플리케이션 로직이다. 이는 컴포넌트 기반 개발기법과 웹의 장점만을 활용하며, 컴포넌트처럼 웹 서비스는 서비스의 구현 결과에 대한 걱정 없이 재사용이 가능한 블랙박스 기능을 가지고 있다. 웹 서비스는 특정 객체를 위한 프로토콜이 아닌, HTTP나 FTP와 같이 언제 어디서나 사용할 수 있는 웹 프로토콜이나 데이터 포맷을 사용하여 접속한다.

웹 서비스의 이용 절차는 다음과 같다. 먼저 서비스 제공자는 공개할 서비스들을 UDDI 레지스트리에 등록한다. UDDI 레지스트리는 웹 서비스에 대한 정보와 이

서비스에 대한 추가 정보의 저장소를 의미한다. 클라이언트가 자신이 이용하고자 하는 웹 서비스를 UDDI에서 검색하면, UDDI는 검색된 웹 서비스의 위치 정보를 클라이언트에 반환한다. 이후 클라이언트는 반환된 서비스의 위치정보를 참조하여 WSDL 파일을 요청한다. 이 요청에 대해 서비스 제공자는 서비스의 상세 정보가 기술되어 있는 WSDL 파일을 서비스 클라이언트에게 반환한다. 이러한 일련의 과정을 거친 후에 클라이언트가 자신이 이용하고자 하는 웹 서비스를 SOAP 메시징을 통해 요청하면, 서비스 제공자는 요청된 서비스를 처리한 후 결과를 SOAP 메시징을 통해 서비스 클라이언트로 반환한다. (그림 1)는 웹 서비스 이용 과정을 보여주고 있다.

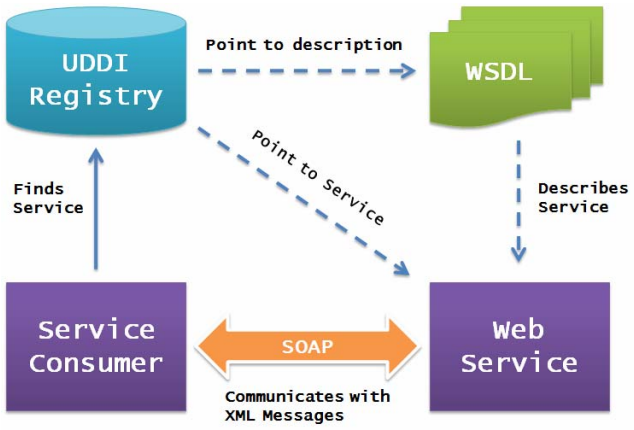

그림 1. 웹 서비스 이용 절차

웹 서비스는 컴포넌트와 같이 서비스 구현 방법을 고려하지 않고도 다시 사용할 수 있는 블랙박스 기능을 제공하며, 제공된 서비스를 설명하는 잘 정의된 인터페이스나 계약서도 제공한다. [3]

2.2 웹 서비스의 데이터 전송 표준, SOAP

SOAP(Simple Object Access Protocol)은 요청(request)/응답(response) 메시지의 내용을 XML로 표현하고, 전송 프로토콜로서 HTTP를 채택하였다. SOAP 메시지는 크게 HTTP상에서 XML 메시지를 전달하기 위해 부가적으로 필요한 정보를 포함하고 있는 HTTP Header 파트와 파라미터, 데이터타입, 반환값 등 실제 전달되어야 할 값들을 포함하고 있는 SOAP Envelope 파트로 구분할 수 있다. SOAP Envelope 파트는 (그림 2)와 같이 다시 클라이언트가 메시지를 처리하는데 필요한 부가적인 처리 정보를 지니고 있는 SOAP Header 부분과 실질적인 메시지를 지니고 있는 SOAP Body 부분으로 구분된다. [4]

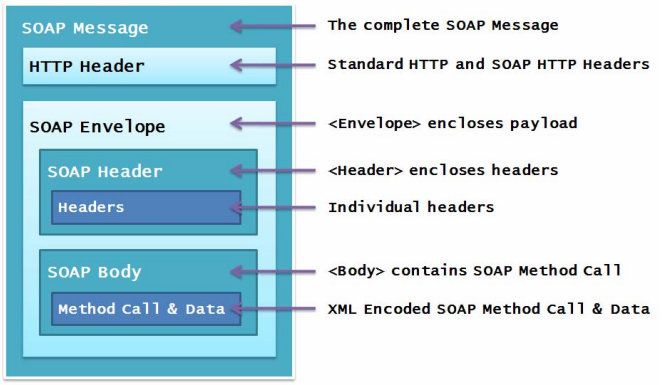

그림 2. SOAP 메시지 구조

(그림 2)는 SOAP 메시지의 전체적인 구조를 보여주고 있다. 분산 객체 프로토콜은 원격 어플리케이션을 활용하는데 효과적인 수단이지만 오늘날의 인터넷 환경에서 그대로 사용하기 위해서는 인프라의 변경이 필요하다. SOAP은 이러한 문제를 해결해 보안상의 문제없이 기존의 인프라를 그대로 활용 가능하다.

2.3 웹 서비스 기술 언어, WSDL

WSDL(Web Service Description Language)은 웹 서비스를 기술하는 일종의 스크립트 언어이다. WSDL은 XML 포맷으로 구성되고 HTTP를 통해서 전달될 수 있으며 다르게 말한다면 인터페이스를 정의하는 IDL(Interface Definition Language)에 해당한다. 즉, 이 서비스가 어떤 메소드, 어떤 속성을 가지며, 어떤 인수로 호출해야 하고 어떤 방식의 반환값을 제공하는지를 알려주는 것이다. 이 내용을 알게 되면 클라이언트에서는 알아낸 인터페이스 규약에 맞추어 호출하고 서비스를 사용할 수 있게 된다.

2.4 웹 서비스의 위치 정보 서비스, UDDI

UDDI (Universal Description, Discovery & Integration) 는 표준 인터넷을 통하여 글로벌 레지스트리에 저장된 정보를 공유하도록 구성되어 있으며, 이러한 방식을 이용하여 전 세계의 원격지 구성 요소를 간단하면서도 동적으로 사용이 가능하다. 또한 UDDI는 웹 서비스와 통합되어 있기 때문에 웹 서비스의 설명과 발견하는 기능이 연동되어 있다. 이것은 마치 전 세계 모든 기업의 상호와 전화번호를 기록한 거대한 전화번호부와 같은 역할을 한다. 그래서 UDDI를 통하여 개발자는 자신이 원하는 기능을 제공하는 서비스 제공자를 발견하고 사용할 수 있다. [5]

3. 시스템 설계

3.1 동기 방식의 웹 서비스

동기 웹 서비스의 경우 클라이언트는 서비스를 호출한 다음 요청에 대한 응답을 기다린다. 클라이언트는 웹 서비스 요청 후에 자체 프로세싱을 보류하기 때문에 짧은 시간에 요청을 처리할 수 있는 동기 웹 서비스를 비동기 웹 서비스보다 선호하고 있다. 또한 동기 웹 서비스는 어플리케이션이 요청에 대해 더 빠르게 응답해야 하는 상황에서도 자주 사용된다. 동기 통신에 의존하는 웹 서비스는 대개 RPC 지향 방식을 사용하고 있다.

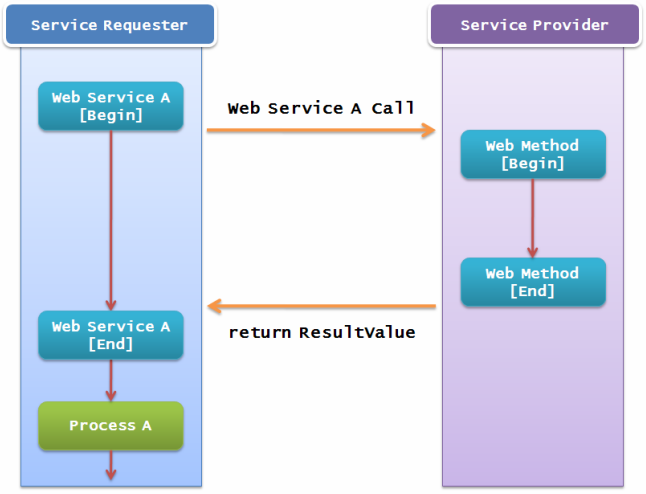

그림 3. 동기 웹 서비스의 동작 절차

전자상거래 어플리케이션 등에 사용되는 신용 카드 서비스는 동기 웹 서비스의 좋은 예라 할 수 있다. 일반적으로 클라이언트인 전자상거래 어플리케이션은 체크아웃 시에 자세한 신용 카드 정보를 사용하여 신용 카드 서비스를 호출한 다음 신용 카드 트랜잭션의 승인이나 거부를 기다린다. 클라이언트는 트랜잭션이 완료될 때까지 프로세싱을 계속할 수 없으며, 신용 승인 획득은 트랜잭션을 완료하기 위한 전제 조건이다. [6]

(그림 3)은 동기 웹 서비스의 동작 절차이다. Service Requester가 Web Service A를 요청하면 결과가 반환될 때까지 기다려야 하는 대기시간이 발생하게 된다.

3.2 비동기 방식의 웹 서비스

비동기 웹 서비스의 경우 클라이언트가 서비스를 호출하지만 응답을 기다리지 않거나 기다릴 수 없다. 이러한 웹 서비스의 경우 대개 서비스가 요청을 처리하는데 일정한 시간이 걸리기 때문에 클라이언트가 응답을 기다리지 않는다. 클라이언트는 응답을 기다리는 대신 다른 프로세싱을 계속 진행할 수 있으며, 나중에 클라이언트가 응답을 수신하면 서비스 요청을 시작했던 프로세싱을 재개할 수 있다.

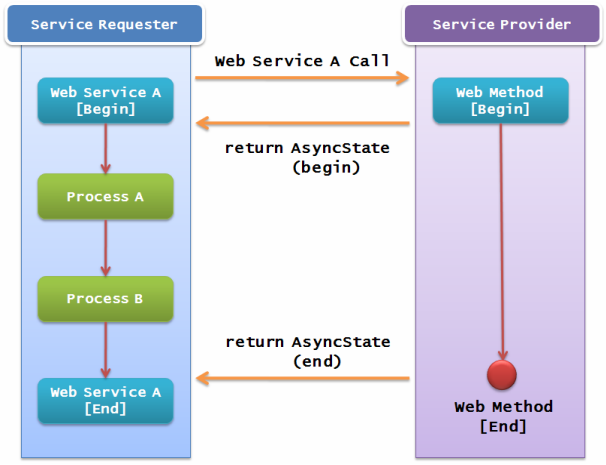

그림 4. 비동기 웹 서비스의 동작 절차

(그림 4)는 비동기 웹 서비스의 동작 절차이다. 비동기 방식에서는 Web Service A를 요청한 후 다른 프로세스를 처리할 수 있다. 즉, 동기 방식과는 다르게 Web Service A의 처리가 완료될 때까지 기다리는 대기 시간 없기 때문에 비교적 효율적이다.[7]

3.3 일괄처리 계산 시스템 설계

비동기 방식의 웹 서비스는 처리 시간이 길거나, 여러 서비스가 동시에 수행되어야 하는 경우에 적용할 수 있다. 처리 시간이 긴 경우 웹 서비스 요청을 한 후 사용자는 다른 프로세스를 수행할 수 있기 때문에 결과가 반환될 때까지 기다려야 하는 대기시간을 활용할 수 있다. 또한, 하나의 작업이 여러 프로세스로 구성될 경우 동시에 호출해 일괄처리를 할 수 있으므로 효율적이다.

본 논문에서 구현한 사칙연산 일괄처리 시스템은 사칙연산을 비동기 방식의 웹 서비스로 구현해 일괄적으로 호출해 결과를 받아오는 방식이다. 동기 방식으로 구현할 경우 각각의 연산을 별도로 요청해야 하는 번거로움과 결과가 반환될 때까지 기다려야 하므로 비효율적이다. 비동기 방식으로 호출할 경우 한 번의 요청 행위로 4가지의 웹 서비스를 동시에 처리할 수 있다.

4. 시스템 구현

4.1 비동기 방식의 웹 서비스 호출

비동기 방식으로 웹 서비스를 호출하기 위해 서비스 제공자는 사용자가 요청한 웹 서비스가 완료되었는지의 여부를 파악하고 있어야 한다.

(그림 5)의 calPlusCompletedEventHandler 이벤트는 웹 서비스가 수행되는 동안 계속해서 완료 여부를 감시하기 위해 실행되고, calPlusAsync 메소드를 호출하면 각각의 처리 절차에 따라 상태값을 관리하게 된다.[8]

```
try
{
    statusBarPanel1.Text = "Async Web Service Start";
    statusBarPanel2.Text = "연결서버 : " + CalService2.Url;

    a = Convert.ToDouble(this.maskedTextBox1.Text);
    b = Convert.ToDouble(this.maskedTextBox2.Text);

    CalService2.calPlusCompleted += new ExternalService.calPlusCompletedEventHandler(service_calPlusCompleted);
    listBox1.Items.Insert(0, "calPlus 웹 서비스 처리 중...");
    MessageBox.Show("aa");
    string address = "http://61.245.232.200:413/Service.asmx?op=calPlus";
    webBrowser1.Navigate(new Uri(address));
    CalService2.calPlusAsync(a, b);

    CalService3.calMinusCompleted += new ExternalService.calMinusCompletedEventHandler(service_calMinusCompleted);
    listBox1.Items.Insert(listBox1.Items.Count , "calMinus 웹 서비스 처리 중...");
    MessageBox.Show("bb");
    address = "http://61.245.232.200:413/Service.asmx?op=calMinus";
    webBrowser2.Navigate(new Uri(address));
    CalService3.calMinusAsync(a, b);

    CalService4.calMultiplyCompleted += new ExternalService.calMultiplyCompletedEventHandler(service_calMultiplyCompleted);
    listBox1.Items.Insert(listBox1.Items.Count , "calMultiply 웹 서비스 처리 중...");
    MessageBox.Show("cc");
    address = "http://61.245.232.200:413/Service.asmx?op=calMultiply";
    webBrowser3.Navigate(new Uri(address));
    CalService4.calMultiplyAsync(a, b);

    CalService5.calDivisionCompleted += new ExternalService.calDivisionCompletedEventHandler(service_calDivisionCompleted);
    listBox1.Items.Insert(listBox1.Items.Count, "calDivision 웹 서비스 처리 중...");
    MessageBox.Show("dd");
    address = "http://61.245.232.200:413/Service.asmx?op=calDivision";
    webBrowser4.Navigate(new Uri(address));
    CalService5.calDivisionAsync(a, b);

    statusBarPanel1.Text = "Async Web Service End";
}
catch (Exception err)
{
    MessageBox.Show(err.ToString());
}
```

그림 5. 비동기 웹 서비스 호출

요청한 웹 서비스의 처리 상태값이 완료되었다고 반환되면 service_calPlusCompleted 메소드를 호출해 사용자에게 웹 서비스가 완료되었음을 알린다.

4.2 실행 화면

사칙연산을 위해 사용자는 두 항의 값을 입력하고 “Async Run”을 클릭하면 비동기 방식으로 웹 서비스를 호출한다. (그림 6)의 실행 결과는 사칙연산의 결과와 각각의 웹 서비스에 해당하는 SOAP 메시지를 출력해준다. 또한 각각의 서비스가 수행되는 순서를 확인하기 위한 “Work History”를 출력한다. “Work History”의 내용을 보면 여러 웹 서비스가 동시에 호출되고 결과가 반환되는 것을 볼 수 있다.

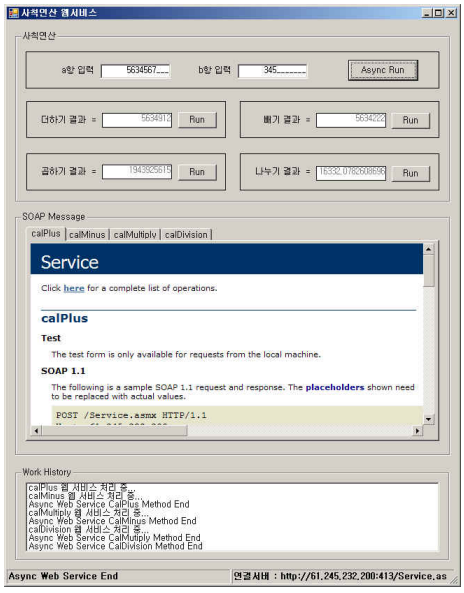

그림 6. 실행 화면

5. 결론

모든 서비스를 비동기 방식으로 구현할 수 있는 것은 아니다. 동기 방식으로 서비스를 호출해서 해당 결과를 받아와 다음 프로세스로 넘기는 흐름일 경우 비동기 방식은 적용할 수 없다. 구현하고자 하는 비즈니스를 분석해 비동기 방식으로 적용할 수 있는 부분들을 추출하면 동기 방식과 비동기 방식을 혼용할 수 있기 때문에 효율성을 극대화 할 수 있다. 하지만, 대부분의 비즈니스가 데이터베이스와 연동되기 때문에 추후 데이터베이스의 처리까지 비동기 방식으로 처리해야만 프로세스 흐름에 모순이 발생하지 않는다. 효율성을 극대화 시킬 수 있는 비동기 방식의 웹 서비스 및 데이터베이스 시스템은 선택이 아닌 필수가 될 것이다.

Acknowledgement

“이 논문은 2007년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(지방연구중심대학육성사업/헬스케어기술개발사업단).”

참고문헌

[1] 김은주, "시맨틱 웹", 한국정보통신기술협회 TTA 저널 제87호, 2003.6

[2] 최중민, "시맨틱 웹의개요와 연구동향", 정보과학회지, 제21권 3호, pp. 4-10, 2003.3

[3] W3C, Web Service Specification, http://www.w3.org/2002/ws

[4] W3C, Simple Object Access Protocol Specification, http://www.w3.org/TR/SOAP

[5] UDDI.org, Universal Description, Discovery and Integration 2.0 Data Structure Specification, http://www.uddi.org/pubs/DataStructure-V2.00-Open-20010608.pdf

[6] Microsoft MSDN, XML Web Service, http://msdn.microsoft.com/library/default.asp?url=/nhp/Default.asp?contentid=28000442

[7] 이경욱, Sun Korea Marketing Development, http://kr.sun.com/developers/tech_docs/web1112/java01.html, 2003년 11월

[8] Matt Powell, Microsoft MSDN, 서버 측 비동기 웹 메소드, http://www.microsoft.com/korea/msdn/library/ko-kr/dnservice/html/service10012002.aspx, 2002년 10월