한국시뮬레이션학회 논문지
Vol. 16, No. 4, pp. 33-41 (2007. 12)

.NET Framework를 서비스 플랫폼으로 사용한 SOA모델 구현 및 성능분석

이성규[1] · 진찬욱[2] · 김태석[1†]

Implementation and Performance Analysis of SOA Model using Service Platform for .NET Framework

Seong-Kyu Lee · Chan-Uk Jin · Tai-Suk Kim

ABSTRACT

Service-Oriented Architecture(SOA) define the interaction method between two computing entities that one entity performs a unit task instead of another entity. This, unit task, is called "Service" and interaction of these services should have independency and loosely coupled task. The effect of SOA's main functions such as loosely coupled task and independent interoperability with influence the possibility of flexible message communication between different way and different users. In this article, we analyzed the performance about system stabilization between general web service and SOA based application that implemented through WCF based messaging framework using .NET Framework and integrated data presentation method. As the result of test, we confirmed that SOA environment using WCF have more advantages.

Key words : Service-oriented architecture, Web service, WCF, .NET framework, XML, SOAP

요 약

서비스 지향 아키텍처는 하나의 실체가 다른 하나의 실체를 대신해 단위 작업을 수행하도록 하는 방식으로 두 개의 컴퓨팅 실체가 상호작용 하는 방법을 정의한다. 이 단위 작업은 "서비스"라고 지칭되며, 이 서비스 상호작용들은 독립적이고 느슨한 결합을 가져야 한다. SOA의 이러한 특징인 느슨한 결합과, 높은 상호운영성의 효과는 서로 다른 이용자들이 서로 다른 방식으로 서비스와 의사소통을 하기 위해 얼마나 유연한 메시지 통신이 가능한가에 따라 결정된다. 본 논문에서는 .NET Framework를 이용한 WCF 기반의 메시징 프레임워크와 통합된 데이터 표현 방식을 이용하여 서비스 지향 아키텍처를 기반으로 한 웹 서비스 애플리케이션과 일반적인 웹 서비스에 대해 애플리케이션의 응답 성능 및 시스템의 안정성과 성능 지수를 비교 분석하였으며 그 결과를 통하여 WCF를 이용한 SOA환경의 웹 서비스가 가지는 장점을 확인 인하였다.

주요어 : 서비스 지향 아키텍처(SOA), 웹 서비스, WCF, .NET Framework, XML, SOAP

1. 서 론

오늘날 기업의 조직에서는 회사의 비즈니스가 민첩하게 움직이는 만큼 IT의 역할도 그에 맞게 움직이게 하기 위하여 SOA를 이용한 비즈니스 전략 수립에 관심을 가지고 있다. 그러나 SOA를 IT환경에 적용하는 방법은 너무나 다양하다. 예를 들어 업무의 핵심 비즈니스를 IT환경에 맞게 프로그램으로 처리하는 경우 비즈니스 로직과 프로세스의 복잡성에 따라, 주고받아야 할 데이터의 형태가 다양하게 되며 그에 따라 업무에 맞게 설계된 프로그램의 수정 또는 변경과 같은 변화중심의 환경에 맞는 프로그램 구조가 요구된다. 서비스 지향 아키텍처(service-oriented architecture : SOA)는 이러한 부분을 좀 더 쉽게 접근하고 해결하기 위해 Microsoft, IBM, Sun Micro-

2007년 10월 4일 접수, 2007년 12월 18일 채택
[1] 동의대학교 컴퓨터소프트웨어공학과
[2] 동의대학교 인터넷응용공학과
주 저 자 : 이성규
교신저자 : 김태석
E-mail; tskim@deu.ac.kr

system 및 Oracle과 같은 대표적인 IT업계에서 다양한 방법으로 접근하고 있다. 최근 XML 기반의 웹서비스 기술이 등장하면서 SOA는 더욱 새롭게 조명되고 있으며, SOAP (simple object access protocol), WSDL(web services description language), 그리고 UDDI(universal description, discovery and integration)등과 같은 공개표준이 정의되어 서로 다른 벤더간의 상호 운용이 용이하게 됨에 따라 SOA의 구현기술이 기업의 비즈니스를 서비로 수립하는 전략의 방법으로 자리를 잡고 있다[7].

본 연구에서는 서비스 지향 아키텍처를 구현하기 위한 방법으로 .Net Framework 3.0과 WCF(windows communication foundation)를 기반으로 XML을 이용한 메시지 처리 방법을 제시하고, 레거시 환경의 웹 프로그램이나 JAVA기반의 플랫폼과 비교하여 성능 부하테스트결과를 확인 하였다.

본 논문에서는 2장에서 SOA와 WCF의 기반기술을 설명하고, 3장에서 .NET Framework 3.0과 WCF를 이용한 SOA설계 방안을 제시한다. 4장에서는 J2EE기반으로 이미 구성되어 있는 주식거래 샘플 프로그램을 .NET기반의 SOA환경으로 재설계하였다. 5장에서 설계된 프로그램에 대해 성능 평가를 하고 6장에서 결론 및 향후 연구과제에 대해 기술하였다.

2. SOA의 정의와 WCF 관련 연구

SOA는 서비스들을 기반으로 하는 소프트웨어 아키텍처로 애플리케이션의 기능들을 사용자(consumer)에게 적합한 크기로 공개한 서비스의 집합을 제공하고 정책(policy) 또는 프레임워크(framework)를 통해 바인딩이 가능하도록 구현한다. 이 때 서비스는 단일한 표준 기반의 인터페이스 형태를 통하여 구현하며 독립적으로 추상화되고, 호출(invoke), 공개(publish), 발견(discovery)과 같은 오퍼레이션을 수행한다. 즉, SOA란 서비스로 정의되는 분할(decomposition)된 애플리케이션 조각들을 단위로 느슨하게 연결해 하나의 완성된 애플리케이션으로 만드는 아키텍처이다[4]. 일반적으로 SOA가 웹 서비스 기반의 애플리케이션에서 많이 적용되는 이유는 데이터 표현 및 전송을 XML이라는 표준 형식을 이용함으로써 서로 다른 시스템간의 상호 운영성을 높이기에 적합하기 때문이다. 이러한 관점에서 WCF(windows communication foundation)는 두 개의 프로세스 사이의 통신을 위해 SOAP (simple object access protocol)기반의 XML 메시지 사용을 제공하여 웹서비스 개발에 빠른 응용 개발 방법을

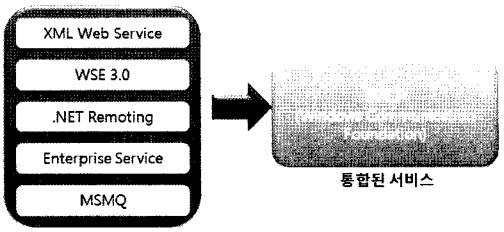


그림 1. WCF 통합 서비스 정의

제공한다. WCF는 여러 가지 분산서비스들을 통합하여 이루어져 있기 때문에 기존 분산 서비스들이 가지고 있는 특징과 기능들의 조합이 가능하다. 서비스를 구현할 때 보안을 어떻게 구현할지 설정하고, 안정성과 트랜잭션과 같은 기능을 자신이 원하는 서비스대로 조합하여 구현할 수 있다[1].

그림 1은 WCF가 통합하고 있는 분산기술의 항목을 정의한 것이다. 여기에서 정의된 WCF 통합 서비스의 각 분산기술은 아래에 나열된 특징을 통하여 서비스를 통합하고 원하는 서비스 기능을 쉽게 구현함으로써 높은 생산성을 가지게 된다. 그리고 웹 서비스의 확장 스펙(WS-*)을 모두 지원하기 때문에 좋은 상호 운영성을 보여준다. 아래는 WCF가 통합하고 있는 각 분산 기술에 대한 각 역할을 설명한다.

- XML Web Service - 타 플랫폼과의 상호 연동
- WSE 3.0 - WS-* 프로토콜 지원
- NET Remoting - 확장성 및 위치 투명성
- Enterprise Services - 분산 트랜잭션 지원
- MSMQ - 메시지 지향 프로그래밍

SOA를 구현하기 위해 WCF에서는 서비스와 클라이언트가 Endpoint를 통해서 통신하게 된다. 그림 2와 같이

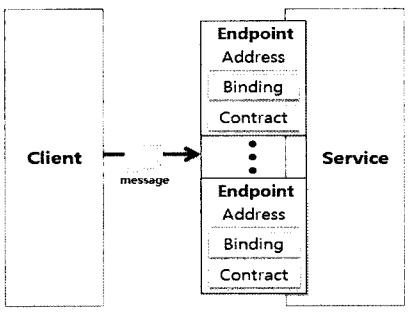


그림 2. Endpoint를 통한 WCF통신

각 Endpoint는 주소(address), 바인딩(binding), 그리고 계약(contract)로 구성된다. 주소는 메시지를 어디로 보낼 것인가를 결정하고, 메시지를 어떻게 전송할 것인지는 바인딩에서 설명하며 계약정보를 통해 메시지가 포함하고 있는 내용이 무엇인지를 알 수 있다[16].

본 논문에서는 이러한 SOA의 개념을 실제 애플리케이션 상에서 구현하기 위해 Microsoft의 .NET Framework와 WCF메시징 기법을 이용하여 웹서비스를 구현 할 때 어떤 기대 효과가 있는지 연구 하였다.

3. .NET Framework 기반의 SOA 모델 설계

3.1 XML 메시지와 WCF의 InofSet 역할

SOA기반의 서비스는 모두 메시지(message)를 통해 커뮤니케이션을 한다. 여기서 말하는 메시지란 하나의 서비스가 다른 서비스로 전송하는 정보의 단위로서 스키마(schema)를 사용하여 구조화되어 전송되어야 하며 메시지를 이해하는데 필요한 모든 정보를 포함하거나 참조함으로써 개념적으로 스스로 충족될 수 있어야 한다. 이렇게 서비스 간 커뮤니케이션에 사용되는 메시징 모델은 스키마를 통해 어렵지 않게 표현하기 위해 SOA에서는 XML을 이용한 메시지 교환을 기본으로 사용한다. XML이 처음으로 사용되던 시절 소프트웨어 업계는 XML 문서상의 데이터에 대한 표준 정의가 제공되지 않아 업계 간에 서로 상이한 형식을 사용하고 있었으나 W3.org에서 InfoSet(XML information set)을 정의하고 내부에 어떤 정보를 포함하고 있는지를 기준으로 요소와 특성을 결정할 수 있도록 하였다[10].

WCF에서의 메시지 처리는 클라이언트와 Endpoint의 데이터 교환 단위이다. 메시지는 기본적으로 SOAP 기반으로 Infoset의 메모리 내에 표현된다. 메시지는 텍스트형식이나 XML에 제한되지 않으며 사용된 인코딩 메커니즘에 따라 WCF 이진 형식, 텍스트, XML 또는 다른 사용자 지정 형식을 사용하여 메시지를 처리할 수 있다.

InfoSet은 데이터를 전혀 다르게 표현하는 다른 XML 사양 및 API일지라도 XML 문서의 데이터를 일관되게 보이도록 할 수 있는 중심 역할을 한다. 그림 3에 나와 있는 것처럼 XML 데이터를 사용하는 응용 프로그램과의 공통된 연결 방식을 제공하며 서로 다른 바이트 표현과 프로그래밍 모델 환경 사이의 변환을 위해 XML 프로세서로서의 역할을 한다.

3.2 .NET 플랫폼을 이용한 SOA기본 모델

IT가 발전하면서 애플리케이션 아키텍처의 변화는 터미널 단말을 사용하던 메인프레임에서 GUI 환경을 제공하는 클라이언트/서버 환경으로 그리고, 최근에 브라우저를 사용하는 웹 기반의 멀티 티어(Multi-Tier) 환경까지 왔다. 클라이언트/서버 환경에서의 가장 큰 문제점이던 배포(Deployment)문제를 멀티 티어 환경에서 제공하는 웹 서비스를 통해 보다 쉽게 해결할 수 있었으나 멀티 티어 환경에서는 조직의 변화에 신속하게 대응하기위한 비즈니스 로직의 변경이 어려워 졌다. 서비스 지향 아키텍처가 이러한 비즈니스 로직을 보다 민첩하게 적용할 수 있도록 비즈니스 로직을 최소의 단위로 분할하고, 서비스라는 접근 방법으로 비즈니스 로직과 사용자를 연계한다. 4에서 통해 비즈니스 로직과 서비스를 통한 상호 연동 흐름을 나타낸다. 위와 같은 기본적인 SOA 개념을 실제 .NET플랫폼 환경에서 서로 연관된 구성요소와 배치하기 위하여 서비스를 생성하고 각 서비스의 내용을 구성하는 부분, 기능에 따라 구분된 계층(Layer)부분 그리고, 데이터와 호스팅 서비스를 담당하는 부분으로 구분하였다. 특히 계층 부분은 기능의 역할에 따라 다음과 같이 5가지로 분리하여 정의 하였다.

- Delivery Layer: Presentation layer의 역할을 수행하는 ASP.NET기반의 application

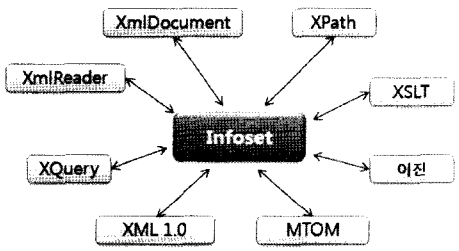


그림 3. XML InfoSet의 역할

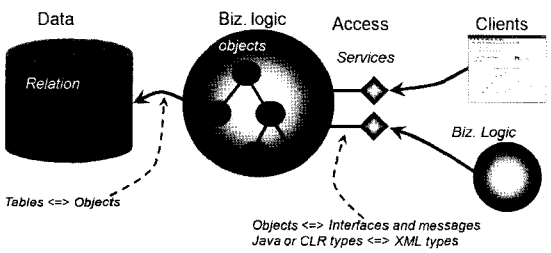


그림 4. SOA기반에서 애플리케이션 연동방법

- Orchestration Layer: Workflow나 Connected Service Framework와 같이 서비스와 서비스상호간에 중계역할을 담당하는 부분
- Service Connectivity Layer: 서비스와 플랫폼 상호간에 정보를 전달하여 플랫폼과 서비스를 연결하는 역할
- Platform Service Layer: SOA의 기반 플랫폼으로 동작하기 위한 Framework 계층
- Legacy Integration Layer: 레거시 시스템과 이기종 플랫폼을 통합하거나 연결하기 위한 계층

4. .NET Framework와 WCF를 이용한 SOA 서비스 구현

본 논문에서는 3장에서 설계한 .NET Framework 기반의 SOA서비스 모델 구성에서 Service Connectivity Layer를 기준으로 워크플로우 서비스 또는 BPM(Business Process Management) 레이어에 메시지를 주고받기 위하여 WCF를 중심으로 정보를 전달하도록 구성하였다.

4.1 WCF를 위한 메시징 설계

메시징 프레임워크를 설계하기 위해서는 임의의 헤더를 사용하여 메시지 페이로드를 확장해야 한다. 헤더는 메시지와 함께 이동하는 간단한 부가정보로서 보안, 안정적인 메시징, 트랜잭션 등의 추가 메시지처리 기능을 구현하는 데 사용된다. XML 메시지의 경우 XML 헤더를 사용하여 XML 페이로드를 확장한다는 의미이며, 둘 다 컨테이너 요소에 포함된 XML 요소로 표현하기 때문에 SOAP을 이용할 수 있다. SOAP 프레임워크를 사용하면 각 전송 방식에서 제공되는 고유한 기능에 의존하지 않고 모든 전송에서 사용할 수 있는 XML 기반 프로토콜을 정의할 수 있기 때문에 WS-Addressing은 SOAP을 확장하여 SOAP 메시지의 주소 지정과 라우팅을 위한 전송 중립적인 방식을 구성할 수 있다. SOAP과 WS-Addressing은 모두 InfoSet을 기반으로 하며 메시지를 전송할 때 XML 1.0 구문을 사용하도록 한다.

그림 6에서 Message 클래스를 사용하여 모든 메시지를 모델링 하였다. Message 클래스는 기본적으로 메시지 본문과 여러 메시지 헤더 및 속성을 모델링하며 각 메서드는 메시지를 만들고, 여러 헤더 및 속성을 조작하는 데 사용하도록 구성하였다. 그리고, WCF를 위한 메시지 클래스에서 XmlDictionaryReader/Writer 표현을 지원하도록 하여 이를 통해 응용 프로그램이 다양한 문서 표현과 성능 요구 사항을 충족할 수 있도록 읽거나 기록할 수 있다. 메시지 개체는 다양한 정적 CreateMessage 오버로드 중 하나를 호출하여 만들 수 있으며 IDisposable을 사용하거나 Close를 명시적으로 호출하여 메시지 개체를 제거할 있다. 또한 메시지를 보낼 때와 받을 때의 방식은 서로 다른 차이를 가지도록 하여 상황에 따라 유연하게 동작할 수 있도록 구성할 수 있는데, 메시지를 보낼 때에는 메시지의 처음부분 부터 작성하여 메시지를 보내고 받을 때에는 메시지 스트림으로부터 새 메시지를 만들 수 있도록 한다.

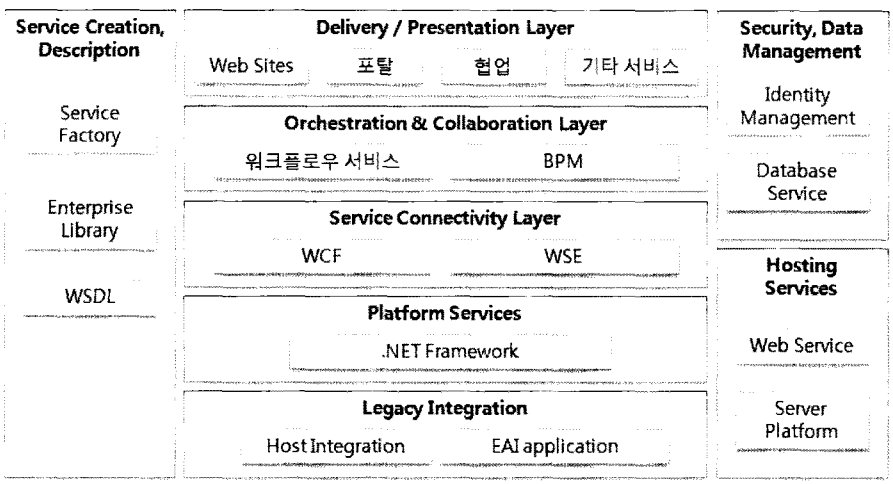


그림 5. .NET Framework기반의 SOA 서비스 모델

```
public abstract class Message : IDisposable
{
    // numerous overloads for creating messages
    public static Message CreateMessage(...);

    // creates a buffered copy of a message
    public MessageBuffer CreateBufferedCopy(int
maxBufferSize);

    // closes a message
    public void Close();

    // returns an XmlDictionaryReader for reading the body
    public XmlDictionaryReader GetReaderAtBodyContents();

    // deserializes the body into a .NET object
    public T GetBody<T>();
    public T GetBody<T>(XmlObjectSerializer serializer);

    // numerous methods/overloads for writing messages
    public void WriteMessage(XmlDictionaryWriter writer);
    public void WriteBody(XmlDictionaryWriter writer);
    public override string ToString();

    // properties
    public abstract MessageHeaders Headers { get; }
    public abstract MessageProperties Properties { get; }
    public MessageState State { get; }
    public abstract MessageVersion Version { get; }
    public virtual bool IsEmpty { get; }
    public virtual bool IsFault { get; }
}
```

그림 6. WCF를 위한 메시지 클래스

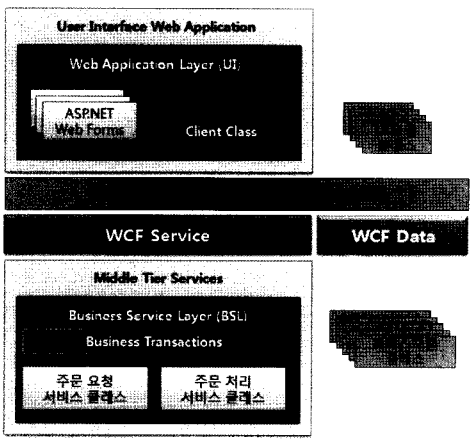


그림 7. 주문 요청/처리를 위한 서비스 모델

4.2 비즈니스 서비스 모델 구현

WCF 기반의 메시지가 SOA 서비스 모델의 각 레이어를 연결하고 정보를 주고 받기위한 프로세스를 구현하기 위해 주문 요청 및 처리 프로그램을 샘플로 작성하였다.

그림 7은 웹 서비스를 크게 사용자 인터페이스를 담당하는 웹 애플리케이션과 주문 요청/처리와 같은 비즈니스 서비스 부분을 담당하는 Middle-Tier 서비스로 구분하고

```
public OrderDataBean buy(string userID, string symbol,
double quantity, int orderProcessingMode)
    {
        System.Transactions.TransactionOptions
        TxOps = new TransactionOptions();
TxOps.IsolationLevel =
  System.Transactions.IsolationLevel.ReadCommitted;
        using (TransactionScope scope = new
TransactionScope(TransactionScopeOption.Required,TxOps))
        {
            try
            {
                IOrder dal =
Trade.DALFactory.Order.Create();
                OrderDataBean order = dal.buy(userID,
symbol, quantity);
                scope.Complete();
                return order;
            }
            catch (Exception e)
            {
                Util.LogError(e.Message, false);
                throw new Exception(e.Message);
            }
        }
    }
```

그림 8. 주문 처리를 위한 BSL

```
[ServiceBehavior(ReleaseServiceInstanceOnTransactionComplet
e = false, TransactionIsolationLevel =
System.Transactions.IsolationLevel.ReadCommitted,
ConcurrencyMode = ConcurrencyMode.Multiple,
InstanceContextMode = InstanceContextMode.PerCall)]
  [PoisonErrorBehavior]
  public class OrderProcessorService : IOrderProcessor
  {
    ....
  }

[OperationBehavior(TransactionScopeRequired = true,
TransactionAutoComplete = true)]
  public void SubmitOrder(Trade.TraderOrderHost.QueuedOrder
order)
  {
  ....
  }
```

그림 9. WCF를 통한 2단계 커밋 트랜잭션

WCF 서비스와 데이터를 통하여 각 클래스 상호간에 정보를 교환 하도록 하였다. 주문 요청/처리를 실행하는 비즈니스 서비스는 모든 트랜잭션에 대해 일관성을 가지고 데이터베이스의 종류에 상관없이 2-phase commit이 가능하도록 하기 위해 XA트랜잭션을 처리할 수 있도록 하고 System.Transactions 클래스를 사용하여 비즈니스 서비스 모델을 정의 하였다.

WCF를 통한 .NET의 System.Transcations 구조는 메시지 큐나 별도의 트랜잭션 처리를 위해 프로그래밍을 하지 않고서도 자동적으로 2단계 커밋(two-phase commit)로직으로 처리 된다. 그림 9의 코드 방식에서 나타내고 있는

바와 같이 Order Processing 부분과 Operation Behavior 부분이 WCF에서 2단계 커밋의 처리 방법을 보여 주고 있다.

5. 성능 평가 및 결론

5.1 성능 평가

4.2절에서 설계한 내용을 기준으로 .NET Framework 2.0 과 WCF를 이용하여 SOA기반으로 작성한 주문 요청/처리 프로그램과 .NET Framework 1.1 기반의 일반적인 웹 서비스 프로그램에 대해 서로 Request 처리 능력, 시스템 응답 속도 및 리소스 사용률을 비교 분석 하였다. 비교 분석을 확인하기 위해 ACT(application center test)라는 부하 벤치마크 툴과 Windows환경에서 제공하는 성능 카운터 분석 툴을 사용하여 테스트 시뮬레이션을 진행하였다. ACT는 테스트 수행 기간을 입력하고, 동시 접속자 수를 통한 부하의 정도를 입력하여 응용프로그램 초당 처리 할 수 있는 능력을 비교 할 수 있으며 Windows의 성능도구는 테스트 시나리오에 따라 시스템의 응답 및 리소스 현황을 파악할 수 있다. 테스트에 사용한 시스템 환경은, Microsoft Windows 2003 운영체제에서 IIS 6.0을 설치하였으며 동일한 사양의 서버에서 각각 프로그램을 구동시켜 테스트를 진행하였다.

가장 먼저 사용자 요청에 처리 능력을 확인 하였다. 동시 접속 브라우저 수를 샘플링 하기 위해 일반적으로 기업용 웹서버에서 사용하는 서버 스펙의 수준인 Intel P4, 2CPU 및 2GB의 서버에서 테스트 프로그램으로 작성한 애플리케이션에 동시 접속 브라우저 수를 조금씩 증가한 결과 100명을 초과하는 경우, 더 이상 시스템의 응답이 나타나지 않는 것을 확인 하였으며 본 연구에서 사용한 테스트 부하량은 100을 최대값으로 실험 하였다. 테스트를 진행하기 위해, 동시 접속 브라우저수를 10부터 20, 50 그리고 100으로 증가함에 따라 시스템의 반응을 살펴본 결과 사용자가 많을 때와 적을 때, 각 프로그램의 결과값에서 비교할 만한 차이가 나타났다. 다음은 시뮬레이션 과정에서 프로그램 환경에 따라 시스템에 적용한 부하 량과 RPS(Request per Second) 수치 결과를 나타낸다. 시뮬레이션 결과를 보면 사용자의 부하 량에 관계없이 SOA 기반의 프로그램은 일관성 있는 성능 수치를 나타내고 있음을 그림 12를 통해 알 수 있다. 그리고 시스템에 적용한 부하 량이 10~20 사이에서는 일반적인 웹 서비스가 평균 140% 정도 높은 수치를 나타냈으나 시스템에 부하가 50 이상 증가됨에 따라 SOA기반의 프로그램에서 155% 정도 향상된 처리능력을 보여 주었다.

시뮬레이션 과정에서 응용프로그램의 응답 성능과 함께 애플리케이션이 플랫폼 상에서 얼마나 효율적으로 동작하는가를 살펴보기 위하여 일반적인 웹 서비스 프로그램과 SOA기반의 웹 서비스가 시스템에서 사용하는 프로

표 1. 부하에 따른 RPS결과

동시 접속 브라우저 수	10	20	50	100
일반적인 웹 서비스	120	170	205	207
SOA기반의 웹 서비스	71	145	284	353

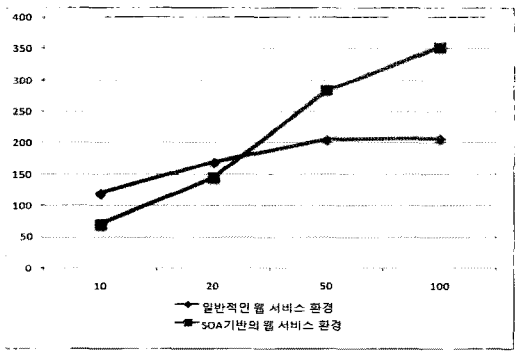


그림 **10.** 동시 처리량에 따른 변화 그래프

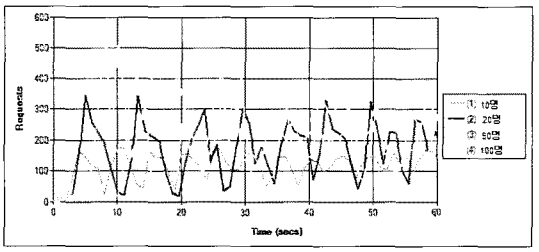


그림 **11.** 일반적인 웹 서비스에서 동시 처리량

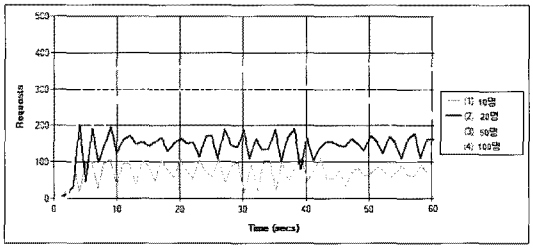


그림 **12.** SOA기반의 웹 서비스에서 동시 처리량

표 2. 시스템 리소스 사용률

	평균 CPU 점유율	평균 메모리 사용량
일반적인 웹 서비스	85%	44MB
SOA기반의 웹 서비스	52.5%	35MB

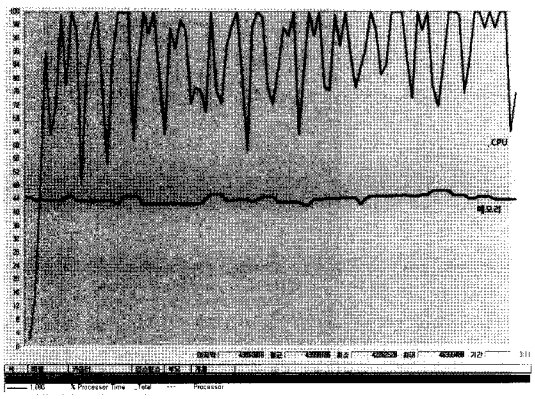


그림 13. 일반적인 웹 서비스에서 시스템 부하

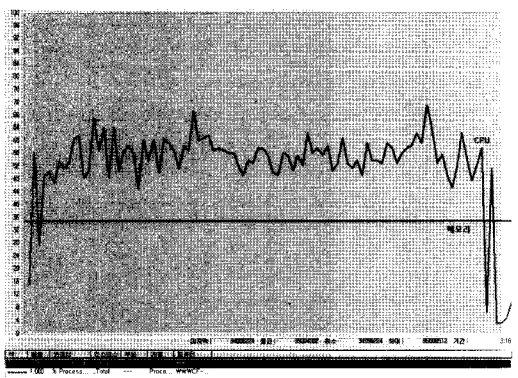


그림 14. SOA기반의 웹 서비스에서 시스템 부하

세서(CPU) 와 메모리의 사용량을 비교해 보았다. 시스템 사용률을 확인하기 위한 테스트과정에서 그림 10의 각 시나리오의 교차지점인 30 수준의 부하를 주고 테스트를 진행한 하였다. 동일한 부하 수준에서 상이한 시스템 성능 결과를 판단하기 위하여 사용자의 요청(Request)를 처리하는 과정에서 두 시나리오 모두 메모리 사용량은 안정적인 상태를 나타냈으나 SOA기반의 웹 서비스가 좀 더 작은 메모리 사용량을 보여 주었다. 시스템의 CPU점유율은 일반적인 웹 서비스 환경에서 평균 85%이상의 높은 리소스 사용률을 나타내어 SOA기반의 웹 서비스보다 30%높게 나타났다.

5.2 결론

최근 IT환경에서 웹 서비스가 중요한 자리를 차지하기 시작하고 많은 기업에서는 대부분의 업무 환경을 웹기반으로 개발하고 있다. 웹 서비스는 클라이언트/서버 환경과 달리 개발 프로그램을 클라이언트에 어렵게 배포하지 않아도 되는 강한 장점을 가지고 있으나, 기업이나 조직의 변화에 맞추어 애플리케이션을 재구성하기가 어렵다는 단점이 드러나기 시작했고 이러한 단점을 극복하기 위해 서비스 지향 아키텍처라는 방법이 연구되기 시작했다.

SOA에서는 서비스를 기반으로 하고 XML 메시지를 교환 할 수 있도록 WSE(web service enhancements)라는 웹 서비스 확장 표준 스펙을 따르고 있으나 WSE로 WS-*프로토콜의 기능을 모두 구현하는 것 역시 어느 정도 한계가 있으며, 이러한 한계를 극복하기 위해 Microsoft, IBM, BEA등과 같은 글로벌 IT기업에서는 표준 프로토콜을 기반으로 자체적인 개발 플랫폼으로 SOA를 구현하는 방법들을 연구 중이다. 특히 Microsoft에서는 .NET Framework 3.0 버전에서 WCF의 역할을 더욱 강화하고 있으며 SOA환경을 구현하기 위한 핵심이 될 것으로 보고되고 있다.

본 논문에서 제안한 .NET Framework 기반의 WCF 또한 이러한 WS-* 확장 사양을 간결하게 구현 할 수 있도록 제안된 새로운 통합 서비스로 SOA 기반의 웹 서비스들이 서로 유연한 통신을 하고 애플리케이션간의 상호 운영성을 높일 수 있는 메시징 아키텍처를 제안 하였다. 이를 통해 SOAP, WS-Addressing 또는 기타 WS-* 프로토콜을 사용하지 않는 기존의 평범한 XML 서비스를 SOA기반의 웹 서비스로 쉽게 구현할 수 있으며 보다 정교하게 바인딩을 구성한다면 다른 프로토콜과 버전이 필요한 특정 통합 시나리오에 맞게 메시징 환경을 구축할 수 있다. 연구 내용의 테스트를 위해 주문 요청/처리를 간단하게 처리하는 부분을 SOA형식의 비즈니스 서비스 모델로 정의하고 각 레이어 에서는 WCF기반의 XML메시지를 교환함에 따라 일반적인 웹 서비스와 성능 차이를 확인하였다. ACT를 이용한 부하테스트 시뮬레이션 결과 일반적인 웹 서비스에 비해 SOA를 기반으로 한 .NET Framework 환경의 웹 서비스가 보다 좋은 성능을 나타내는 것을 확인 하였고, 초기 부하 량이 적은 환경에서는 일반적인 웹 서비스 애플리케이션이 다소 좋은 결과를 나타내었으나 시스템에 부하가 심해질수록 응답 성능이 떨어지고 시스템의 CPU상태도 불안정하고 높아지는 것을 알 수 있었다.

일반적인 웹 서비스에 비해 초기 설계 및 SOA기반의

아키텍처 디자인에서 많은 시간과 노력이 필요하겠으나 재활용성과 상호 운영성을 고려한다면 SOA기반의 웹 서비스에 대한 기대 효과를 높게 가질 수 있을 것으로 전망한다.

참고문헌

1. Clemens Vasters, "Introduction to building Windows Communication Foundation services", http://msdn2.microsoft.com/en-us/library/aa480190.aspx, Sep. 2009.
2. Martin Gudgin, Marc Hadley, Noah Mendelsohn, "Simple Object Access Protocol (SOAP) 1.2", W3C, http://www.w3.org/TR/soap, Apr. 2007.
3. Thomas Erl, "Second generation (WS-*) specifications", http://www.soaspecs.com/page2.asp, 2007.
4. Munindar P. Singh, Michael N. Huhns, "Service-Oriented Computing: Semantics, Processes, Agents", John Wiley & Sons, 2005.
5. Don Box, Franciso Curbera, "Web Service Addressing (WS-Addressing)", http://www.w3.org/Submission/ws-addressing, W3C Member Submission, August 2004
6. Martin Gudgin, Marc Hadley, Tony Rogers, "Web Service Addressing Working Group", http://www.w3.org/2002/ws/addr, 2006.
7. David Booth. "Web Services Architecture", W3C Working Group Note, http://www.w3.org/TR/ws-arch/, Feb 2004.
8. Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana, "Web Services Description Language (WSDL) 1.1", W3C Note 15, http://www.w3.org/TR/wsdl, March 2001.
9. Microsoft Corporation, "Windows Communication Foundation MSDN Library", http://msdn2.microsoft.com/en-us/library/ms735119.aspx, 2006.
10. John Cowan, Richard Tobin, "XML Information Sec (Second Edition)", http://www.w3.org/TR/xml-infoset/, 2004.
11. David Sprott and Lawrence Wikes, "Understanding SOA", CBDI Journal, Sep. 2003.
12. D. Plummer, "Service-Oriented Developement Applications: SODA Pops, Gartners's Internet Strategies Commentary", COM-12-9640, Feb. 2001.
13. David Sprott, "Understanding the Component and Web Services Market", CBDI Journal, p.36-57, May 2001.
14. Park, D. S., Shin, H. J., and Kim, H. K, "A Study on the Development Web Services Component Based Service Oriented Architecture", Journal of Korea Multimedia Society. Vol. 7. No. 10, pp. 1496-1504, 2004.
15. Roberto Chinnici, "Web Services Description Language (WSDL) Version 2.0 Part 1:Core Langueage", http://www.w3.org/TR/2004/WD-wsdl20-20040803/, W3C Working Draft 3, August 2004.
16. Juval Lowy, "Programming WCF Services", O'Rellly, 2007.

이 성 규 (12284@deu.ac.kr)

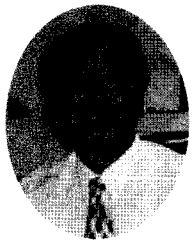
1993 경남대학교 컴퓨터공학과 학사
1998 광운대학교 정보통신학과 석사
2006~현재 동의대학교 컴퓨터소프트웨어공학과 박사과정
2006~현재 동의대학교 컴퓨터소프트웨어공학과 겸임 부교수

관심분야 : 서비스 지향 아키텍처, 웹서비스, 데이터베이스

진 찬 욱 (jinchanuk@hotmail.com)

1999 동의대학교 컴퓨터공학과 학사
2006~현재 동의대학교 인터넷응용공학과 석사과정

관심분야 : 서비스 지향 아키텍처, 웹서비스, XML



김 태 석 (tskim@deu.ac.kr)

1982 경북대학교 전자공학과
1992 일본 KEIO대학 이공학부 계산기과학전공 공학박사
1992 일본 KEIO대학 이공학부 객원연구원
2000~2003 동의대학교 전산정보원장
2003~2005 동의대학교 교무처장
1993~현재 동의대학교 컴퓨터소프트웨어공학과 교수

관심분야 : 웹서비스시뮬레이션, 원격강의, 자연어처리