XML을 이용한 응용 프로그램의 분산 처리

박 긴[0] 조동섭
이 화 여 자 대 학 교 컴 퓨 터 학 과
{ kiny[0], dscho }@ewha.ac.kr

Distributed Processing of APEX(Application Program Embedded XML)

Kin Park[0] Dong-Sub Cho
Dept. of Computer Science and Engineering, Ewha Womans University

요 약

웹을 이용한 많은 양의 데이터의 분산 처리 방식이 XML 등장 이후 보다 가속화 되고 있다. 하지만 지금까지의 어플리케이션간 데이터 전달 방식은 단순히 데이터의 직렬화와 데이터를 처리하기 위한 외부 함수 호출 등의 전달을 위주로 제안된 XML 프로토콜을 중심으로 개선되어 왔다. 이에 본 논문에서 제안하는 APEX(Application Program Embedded XML) 시스템은 방대한 양의 데이터를 분산된 각 MiniServer에서 기존의 어플리케이션 혹은 사용자가 직접 제공한 어플리케이션을 통해 처리하여 그 결과값을 원 데이터와 어플리케이션과 함께 데이터베이스에 HTML 문서와 XML 문서의 형태로 저장한 후, 사용자에게 되돌려주는 시스템으로 대용량 데이터 처리에 드는 비용을 줄여주는 효과를 보인다.

1. 서론

로컬 네트워크를 이용한 방대한 데이터의 분산 처리 방법에는 여러 가지가 있다. 기본적인 모델로서 서로 다른 머신 오브젝트간에 메시지를 주고받는(Message Passing) 방법이 있는데, 이 방법은 각각의 분산 어플리케이션을 만들어야 하며 이것은 서로 통신을 할 수 있는 프로토콜의 표준이 없어 로우-레벨의 통신 처리가 반드시 필요하여 그 오버헤드가 크다[1].

컴포넌트 기술은 메모리, 소스 코드, 데이터 타입 정보, 네트워크에 따른 계층별 상호운용성(interoperation)을 가지고 있다. 널리 사용되고 있는 컴포넌트 기술인 Microsoft의 COM/DCOM, OMG의 CORBA, JAVA의 BEAN등은 그 기술들 간에 서로 호환되지 않으며 웹과의 연동 시 직접 호환이 되지 않아 별도의 프로토콜이 반드시 필요하다.

이러한 어플리케이션 간 데이터 처리 요구는 로컬 영역에서만 그 의미가 있고 수행할 수 있는 것이 아니라, 웹을 이용하여 클라이언트 어플리케이션과 클라이언트 어플리케이션 간, 클라이언트 어플리케이션과 서버 어플리케이션 간에서도 데이터 처리 요구가 있을 수 있다.

아직 표준화되지는 않았지만 XML 프로토콜은 어플리케이션간 데이터 교환을 지원한다. WDDX, XML-RPC와 SOAP등은 XML 프로토콜 표준안의 후보로서 컴포넌트 기반의 모델이다[2].

이 후보안들은 대체로 데이터를 교환할 때 해당 데이터만을 태그로 캡슐화 하여 전달하게 된다. 이 때 태그를 이용하여 쌍방간의 데이터를 직렬화하여 전달하게 되고 데이터를 받은 쪽의 시스템은 태그의 시맨틱을 바탕으로 데이터를 재구성하게 된다. 재구성된 데이터는 시스템에 이미 존재하는 어플리케이션을 통해 처리가 된다.

그러나 데이터만 전달하는 것이 아니라 그에 적합한 처리를 할 수 있는 어플리케이션까지 전달 할 수 있다면 데이터 처리 비용 측면에서 볼 때 큰 가치가 있을 것이다. 이를 위해 본 논문에서는 XML을 이용한 어플리케이션 프로그램의 분산처리 시스템을 구현하고자 한다.

2. 관련연구

2.1 컴포넌트 개념

소프트웨어를 개발함에 있어 미리 구현된 블록을 사용하여 소프트웨어 개발비용과 시간을 단축할 수 있는데, 이 블록을 컴포넌트라고 한다. 이 컴포넌트는 실행단위로 개발자에게 인터페이스만을 제공하여 내부의 상세한 부분을 숨기게 되어 보다 쉽고 빠르게 소프트웨어를 개발할 수 있게 해준다[3].

널리 사용되고 있는 컴포넌트로는 Microsoft의 COM/DCOM, OMG의 CORBA, JAVA의 BEAN 등이 있는데 아직은 각각의 컴포넌트 간에 통합된 표준 사양이 존재하지 않아 서로 상호 운용성을 얻을 수 없다[4].

2.2 XML 프로토콜

XML은 HTML의 확장으로 설계되었는데, 서로 다른 이종의 컴포넌트 기반의 시스템을 통합하는 기술로 사용된다. 이는 XML이 플랫폼과 프로그래밍 언어에 독립적이라 최소한의 표준만 갖추어지면 상호운용성을 획득할 수 있기 때문에 가능한 것이다.

XML 프로토콜(XP)은 어플리케이션 간에 데이터를 교환하는 컴포넌트 모델로 데이터 교환 시 데이터를 태그를 이용하여 캡슐화하고 외부함수를 호출할 수 있게 하며 비구조적 데이터에 대한 직렬화를 지원하며 HTTP 프로토콜을 그대로 사용하여 목적을 달성한다[2].

지금까지 제안되고 있는 대표적인 XML 프로토콜에는 XML-RPC(XML-Remote Procedure Calling Protocol), WDDX(Web Distributed Data exchange), SOAP(Simple Object Access Protocol) 등이 있는데 본 논문에서는 이들 중에서 XML-RPC와 WDDX에 관하여 살펴보도록 한다.

2.2.1 XML-RPC

XML 프로토콜의 후보안인 XML-RPC(XML-Remote Procedure Call)는 리모트 머신에 있는 메소드를 호출하는 방식이다. XML을 이용하여 리모트 머신에 대한 함수 호출을 캡슐화한 후 리모트 머신에 XML 문서를 전달하고 그 결과값을 XML 문서로 되돌려 받는다. XML-RPC는 기존의 HTTP 프로토콜의 POST 방식을 사용하며 XML 표준을 그대로 따르므로 프로그래밍 언어에 독립적일 수 있다 [5].

이진 데이터의 전송은 Base64 인코딩 방식을 이용하며, 데이터 타입 또한 XML 태그를 이용하여 표현한다. 기존의 HTTP 프로토콜을 XML 바인딩으로 이용할 수 있음으로써, XML-RPC는 방화벽에서의 특정 프로토콜에 대한 거부를 피하면서 분산 환경을 지원할 수 있는 특징을 가진다[6].

그러나 XML-RPC는 자체 데이터 타입을 사용함으로써 데이터 타입 레벨의 상호운용성을 보장할 수 없다.

2.2.2 WDDX

WDDX(Web Distributed Data exchange)는 프로그래밍 언어 레벨에서 복잡한 데이터 구조를 교환하기 위한 방법으로 제안된 XML 프로토콜이다. WDDX는 XML을 이용하여 데이터를 데이터 타입, 변수 이름, 변수 값을 포함하여 직렬화 한다.

하지만 WDDX는 데이터의 직렬화에 대한 방법만을 정의하여 리모트 머신의 메소드를 호출하는 기능이 없어서 XML 프로토콜에서 요구하는 조건을 완전히 만족시키지 못한다. 그리고 XML 프로토콜의 바인딩이 HTTP로만 한정되어 있고 지원하는 데이터 타입 역시 한정 되어 있는 단점을 지니고 있다.

표 1은 WDDX의 예이다.

표 1 WDDX의 예

```
<?xml version='1.0'?>
<!DOCTYPE wddxPacket SYSTEM 'wddx_0090.dtd'>
<wddxPacket version='0.9'>
<header/>
  <data>
    <struct>
      <var name='s'>
          <string>a string </string>
      </var>
      <var name='a'>
          <array length='2'>
             <number>10</number>
             <string>second element</string>
          </array>
      </var>
    </struct>
  </data>
</wddxPacket>
```

3. Application Program Embedded XML System

3.1 APEX의 개념

최근의 전자상거래 시스템을 포함한 웹 어플리케이션들은 사용자에게 많은 정보를 제공한다. 많은 양의 상품 데이터를 여러 가지 기준으로 분류하여 분산된 데이터베이스에 저장을 하고자 할 때, 상품 데이터를 모든 데이터베이스 서버에 전송해야 하고 그 분류를 위한 어플리케이션 또한 모든 서버에 필요하다. 하지만 이를 위해 일일이 서버에 어플리케이션을 설치하고 따로 구동한다면 그 오버헤드가 크게 되므로, 어플리케이션의 바이너리 레벨의 코드와 함께 데이터를 서버에 전송함으로써 그 목적을 달성하고자 하는 것이다. 이러한 데이터 및 어플리케이션의 전달은 따로 프로토콜을 마련하지 않고 기존의 HTTP 프로토콜을 이용하여 웹 수준에서 이루어질 수 있도록 한다. 이 시스템을 간단히 APEX라 하고 다음과 같이 구성되며 작동된다.

3.2 APEX System의 구성 및 작동 원리

APEX System의 구성은 그림 1과 같다. 이 시스템은 크게 3부분으로 나눌 수 있는데, 사용자의 데이터 처리 요구를 받아들여 어플리케이션과 데이터를 모아 APEX 메시지를 작성하여 각 프로세싱 서버인 Mini Server에 전달하고 사용자에게 결과를 돌려 주는 역할을 하는 서버인 SURF 서버와 전달된 APEX 메시지를 분석하여 어플리케이션과 데이터를 분리하고 직접 데이터를 처리하게 되는 n개의 데이터베이스를 가진 n개의 MiniServer, 어플리케이션, 원본 데이터와 함께

처리된 결과를 XML 문서와 HTML 문서로 만들어 저장하는 데이터베이스로 구성되어 있다.

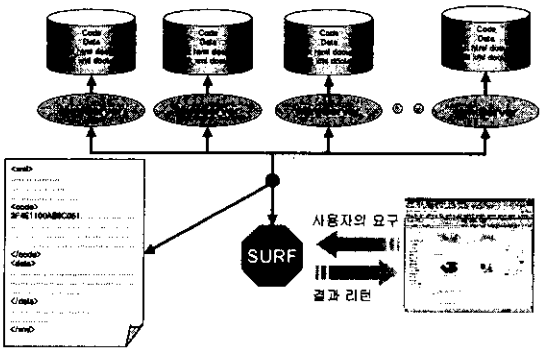

그림 1 APEX의 시스템 흐름도

3.3 APEX 메시지 구조

APEX 시스템에서 기본적으로 채택한 메시지 구조는 그림 2와 같이 APEX 헤더와 APEX 바디로 구성된 APEX 인벨롭 구조이다. APEX 헤더에는 따로 어플리케이션이 필요한지 여부와 필요한 경우 해당 어플리케이션을 명시하고, 전체 메시지의 길이 등을 기록한다. APEX 바디는 어플리케이션의 바이너리 코드와 데이터를 포함한다.

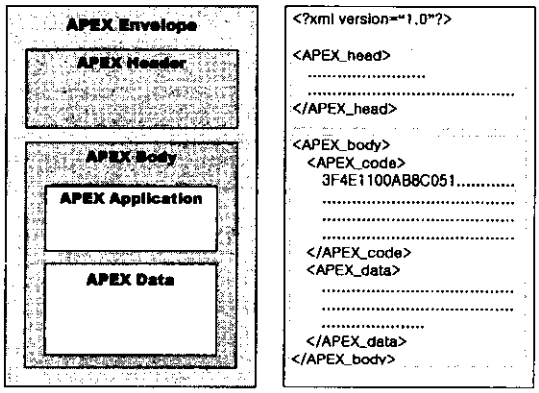

그림 2 APEX의 메시지 구조

3.4 APEX 시스템의 데이터베이스

APEX 시스템에서 데이터베이스에 저장되는 내용은 다음과 같다.

- ▶ 어플리케이션의 바이너리 코드
- ▶ SURF로부터 전달 받은 데이터
- ▶ 프로세싱 결과 : HTML 형태
- ▶ 프로세싱 결과 : XML 형태

4. 결론

APEX 시스템은 방대한 양의 데이터를 웹을 이용하여 분산되어 있는 각 MiniServer에서 사용자에 의해 제공된 어플리케이션 혹은 MiniServer에 있던 기존의 어플리케이션을 통해 처리하여 그 결과값을 원 데이터와 어플리케이션과 함께 데이터베이스에 저장하고 또한 사용자에게 HTML 문서 혹은 XML 문서로 되돌려 주는 시스템이다.

사용자 제공 어플리케이션을 이용하여 데이터를 처리하는 경우, SURF 서버에서 어플리케이션 바이너리 코드와 데이터는 각각 <APEX_code>, </APEX_code>와 <APEX_data>, </APEX_data> 태그로 캡슐화 된다. 전달받은 어플리케이션의 바이너리 코드는 MiniServer에서 다시 디코드하여 실행코드를 만들어 데이터에 적용한 후 그 결과를 사용자에게 알리고, 데이터베이스에 저장한다.

5. 참고 문헌

[1] Dan Grigoras, Stefan Mihaila, " A Framework for Component-Based Distributed Applications Design The CODE: Component Oriented Distributed Environment," Proceeding of the International Conference on Parallel Computing in Electrical Engineering (PARELEC' 00), 2000.

[2] W3 Consortium, " XML Protocol Working Group Charter," http://www.w3.org/2000/09/XML-Protocol-Charter, 2000.

[3] Eduardo Pelegri-Lopart, Laurence P.G. Cable, " How to be a Good Bean," Sun Microsystems, Inc. Sep. 1997

[4] Don Box, DevelopMentor Inc, " Lessons from the Component Wars: An XML Manifesto," http://msdn.microsoft.com/xml/articles/xmlmanifesto.asp, Sep. 1999.

[5] Dave Winer, Userland Software Inc, " XML-RPC Specification," http://www.xml-rpc.com/spec

[6] 이경하, 이규철, " XML 프로토콜," 대한정보과학회지 제19권 제1호 통권 140호, 2000년 1월.

[7] Eric Prud' hommeaux, " XML Protocol Comparisons," http://www.w3.org/2000/03/29-XML-protocol-matrix, 3, 2000.