УДК 004.7(045)

Кременецький Г.М., канд. техн. наук

АНАЛІЗ РЕАЛІЗАЦІЙ WEB-СЕРВІСІВ В КОНТЕКСТІ ВИКОРИСТАННЯ У НЕЙРОННИХ МЕРЕЖАХ, ЩО ДИНАМІЧНО КЛАСТЕРИЗУЮТЬСЯ

Інститут комп'ютерних технологій
Національного авіаційного університету

Проведено порівняльний аналіз відомих реалізацій побудови WEB-сервісів стосовно побудови штучних нейронних мереж, що динамічно кластеризуються. Запропоновано використання AXIS, як найбільш гнучкого рішення

Вступ

Сучасні розподілені додатки будуються на основі *WEB*-сервісів. Це дозволяє використовувати більшість існуючих можливостей Internet мережі та багато вже створених додатків на різних платформах та різних мовах програмування. Крім того використання спеціальних *UDDI (Universal Description Discovery and Integration)* реєстрів *WEB*-сервісів дозволяє використовувати однакові сервіси але від різних постачальників з різних серверів додатків. Тобто, існує безліч можливостей оптимізації розподілених обчислень, як з точки зору фінансових витрат, так і швидкості обчислень.

Технологію *WEB*-сервісів вже використовують найбільші гравці ринку Інтернету провідні пошукові гіганти *Google* та *Yahoo*. Тому цей стандарт має велике майбутнє. Для взаємодії із *WEB*-сервісом необхідно відправити *XML*- повідомлення через *HTTP* протокол. Оскільки кожен сумісний з Інтернет пристрій підтримує протокол *HTTP*, і практично кожна мова програмування має доступ до *XML*- аналізатора, тому обмежень, що стосуються розробки додатків на базі *WEB*-сервісів майже не існує. *WEB*-сервіс – це фрагмент бізнес логіки, до якого можна отримати доступ через Інтернет. Наприклад, сайти електронної комерції можуть використовувати *WEB*-сервіс компанії, що займається доставкою і пакуванням для обчислення вартості доставки того чи іншого товару. Сайти новин можуть збирати заголовки новин і статті, що складаються зовнішніми службами новин і відображати їх на своїх власних сторінках у реальному часі. Компанія може навіть надавати в реальному часі значення своїх фондових опціонів, зчитуючи з якогось фінансового чи інвестиційного сайту. *WEB*-сервіси – це нова технологія, що створена для вирішення великої кількості проблем і недоліків, які виникали в попередніх технологіях розподіленої обробки даних. *WEB*-сервіс – фрагмент логіки програми (компонент), який може викликатись віддалено через Інтернет. Задачі у технології *WEB*-сервісів ті й що у технології компонентних об'єктів, наприклад, таких як *DCOM (Distributed Component Object Model)*, головними із яких є спрощення процесу зборки додатку із існуючих модулів, забезпечення спільного використання функціональних можливостей організаціям і партнерам та дозволити створювати програми, що складаються із великої кількості модулів, які крім того можуть знаходитись у різних куточках світу. В технології *Web*-сервісів робиться акцент на функціональній сумісності. Усі платформи для розробки програмного забезпечення, які дозволяють створювати *Web*-сервіси, використовують однакові принципи відкритих стандартів *XML*, що гарантує можливість створення *Web*-сервісу використовуючи мову. *NET* і викликати його з *Java* додатка, або навпаки.

Використання *WEB*-сервісів дозволяє розділити логіку додатку на блоки, які будуть виконуватись на двох чи більшій кількості комп'ютерних систем. Причин для розподілення логіки програм достатньо багато. Висока масштабованість. Роз-

ділення логіки додатку для розподілу навантаження між різними обчислювальними вузлами. Ці рішення не призводять до покращення продуктивності додатку для окремого користувача, але підвищує масштабованість додатку, тим самим дозволяючи йому обслуговувати одночасно набагато більшу кількість користувачів.

WEB-сервіси мають наступні позитивні якості: просте розгортання, можливість часткового оновлення програми, додаток може бути легко територіально розподілено, додаток збирається розробником як конструктор.

Ця технологія дуже схожа на *RMI* та *CORBA*, але ці технології потребують виділених портів та протоколів передачі даних. *WEB*-сервіси насамперед розраховані на використання існуючого протоколу *HTTP* та надбудови *SOAP* рис. 1. Це дає можливість використовувати існуючу систему безпеки мережі без змін.

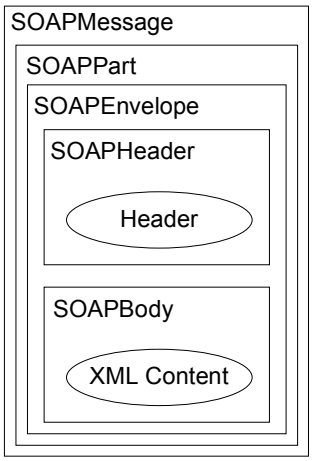

Рис. 1. Ієрархічна структура *SOAP*-повідомлення

Обмін даними між клієнтом-споживачем і провайдером-постачальником послуг – це передача *SOAP*-повідомлень. Протокол *SOAP* пропонує спосіб формування повідомлень в *XML*-форматі, заснований на об'єктній моделі *XML*-документа *DOM (Document Object Model)*. *SOAP*-повідомлення – це ієрархічна структура вкладених XML-елементів (або вузлів): *SOAPMessage*, *SOAPPart*, *SOAPEnvelope*, *SOAPHeader*, *SOAPBody* і вміст передаваного повідомлення, також у форматі *XML* (*XML content*). На рис. 3 представлена ієрархічна структура *SOAP*-повідомлення.

Порівняння AXIS JAX-WS

Сьогодні реалізації *AXIS* від *Apache Group* та *JAX-WS Sun Microsystems* є найбіль поширеними серед розробників програмного забезпечення на мові *Java*. Обидві реалізації дозволяють відтворювати однакову парадигму використання *WEB*-сервісів рис 2.

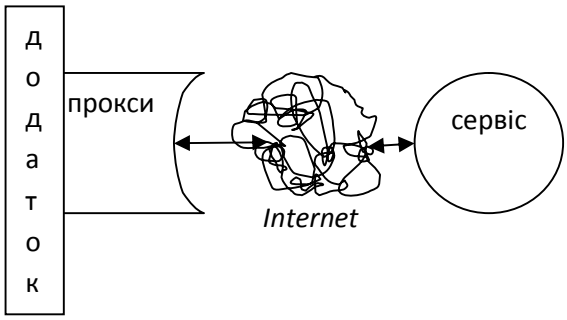

Рис. 2. Парадигма використання *WEB*-сервісів

Тобто, програма додаток, яка виконується на віддаленому клієнті використовує інші модулі, які розташовані на серверах додатків. При цьому сервера додатків можуть знаходитися у будь якому місці. Однією умовою є підключення до мережі *Internet.*

Наведемо приклад створення *WEB*-сервісу у *JAX-WS*:

```
import javax.jws.WebService;
@WebService
public class Hello {
  private String message = new String("Hello, ");
  public void Hello() {}
  @WebMethod
  public String sayHello(String name) {
    return message + name + ".";
  }
}
```

Для побудови сервісу необхідно використовувати спеціальні аннотації *@WebService, @WebMethod*.

Після компіляції класа за допомогою *ant* або *NetBeans*, буде створено додаткові файли необхідні для імпорту *WEB*-сервісу до серверу додатків.

```
package simpleclient;
```

```
import javax.xml.ws.WebServiceRef;
import HelloService;
import Hello;

public class HelloClient {
@WebServiceRef(wsdlLocation="http://localhost:8080/
        helloservice/hello?wsdl")
   static HelloService service;
   public static void main(String[] args) {
      try {
         HelloClient client = new HelloClient();
         client.doTest(args);
      } catch(Exception e) {
         e.printStackTrace();
      }
   }

   public void doTest(String[] args) {
      try {
         System.out.println("Retrieving the port from
                the following service: " + service);
         Hello port = service.getHelloPort();
         System.out.println("Invoking the sayHello operation
                on the port.");
         String name;
         if (args.length > 0) {
            name = args[0];
         } else {
            name = "No Name";
         }
         String response = port.sayHello(name);
         System.out.println(response);
      } catch(Exception e) {
         e.printStackTrace();
      }
   }
}
```

У *AXIS* існує декілька способів створення *WEB*-сервісів.

Розглянемо клас *Calculator*

```
public class Calculator {
  public int add(int i1, int i2) {
    return i1 + i2;
  }
  public int subtract(int i1, int i2) {
    return i1 - i2;
  }
}
```

Перший спосіб це скопіювати файл класу *Calculator.java* у *<your-webapp-root>/axis/Calculator.jws*. У цьому випадку сервер додатків автоматично сформує потрібні файли опису та зробить компіляцію файлу.

Другий спосіб це створення спеціальних дескрипторів. Наприклад:

```
<deployment
xmlns="http://xml.apache.org/axis/wsdd/"
xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
  <service name="Calculator" provider="java:RPC">
    <parameter name="className" value="Calculator"/>
    <parameter name="allowedMethods" value="*"/>
  </service>
</deployment>
```

Завдяки дескриптору можна підключати як сервіс будь-який класс, що вже скомпільовано.

Дескриптори мають також, додаткові можливості, наприклад, ви маєте можливість додати обробники вхідних даних – *Handler.*

```
<handler name="track"
type="java:samples.userguide.example4.LogHandler">
    <parameter name="filename" value="MyService.log"/>
  </handler>
```

Існує можливість описати вхідні параметри сервісу більш детальніше (вказати тип та рівень доступу).

У реалізації *AXIS* існують різні стилі *WEB*-сервісів: *RPC*, *Document*, *Wrapped*, and *Message*.

Розглянемо більш детальніше ці стилі.

RPC – цей стиль сервісу є стилем за замовченням. Він реалізує стандартне представлення *WEB*-сервісу, тобто віддалений виклик методів класу.

Document та *Wrapped* стилі можна розглянути на прикладі єдиного *SOAP*

```
<soap:Envelope
xmlns="http://xml.apache.org/axis/wsdd/"

xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
  <soap:Body>
    <myNS:PurchaseOrder
xmlns:myNS="http://commerce.com/PO">
      <item>SK001</item>
      <quantity>1</quantity>
      <description>Sushi Knife</description>
    </myNS:PurchaseOrder>
  </soap:Body>
</soap:Envelope>
```

Відповідна схема буде виглядати наступним чином:

```
<schema targetNamespace="http://commerce.com/PO">
  <complexType name="POType">
    <sequence>
      <element name="item"
type="xsd:string"/>
      <element name="quantity"
type="xsd:int"/>
      <element name="description"
type="xsd:string"/>
    </sequence>
  </complexType>
  <element name="PurchaseOrder"
type="POType"/>
</schema>
```

Використання стилю *Document* побудує наступний виклик на боці серверу

public void method(PurchaseOrder po)

Виклик з використанням стилю *Wrapped* буде виглядати наступним чином

public void purchaseOrder(String item, int quantity, String description)

Якщо стиль сервісу встановлено *Message*, то існує можливість використовувати наступні методи:

public Element [] method(Element [] bodies);

public SOAPBodyElement [] method (SOAPBodyElement [] bodies);

Ці методи створюють масиви *DOM* елементів типу *Elements* або *SOAP-BodyElements*.

public Document method(Document body);

Метод будує об'єкт класу *Document*

public void method(SOAPEnvelope req, SOAPEnvelope resp);

Метод повертає об'єкти запиту та відповіді класу *SOAPEnvelope.*

Використання цього стилю дозволить легко будувати нейромережеву архітектуру на основі *SOAP* запиту.

Крім того *AXIS* може бути використано на різних серверах додатків: *Tomcat*, *jBoss*, *WEBSphera* та ін.

Оба продукти розповсюджуються безкоштовно та з відкритим кодом.

Висновки

У порівняльному аналізу було виділено основні властивості основних конкурентів.

Майбутня архітектура програмного забезпечення штучних нейронних мереж, що динамічно кластеризуються повинна бути максимально відкрита для подальшого розвитку, мати можливість створення нових классів, що реалізують різні архітектури та математичні алгоритми штучних нейроннизх мереж, нові алгоритми навчання.

Використання *AXIS* має наступні переваги у побудові штучних нейронних мережах, що динамічно кластеризуються: стилі сервісів, різні типи інтеграції нових классів, можливість використання різних серверів додатків. Їх використання дозволить спростити процесс підключення нових сервісів та легко розвивати мережу серверів.

Подано до редакції 26.05.10