УДК 681.3

Голощук Р.О., Кісь Я.П., Чура І.І.
НУ “Львівська політехніка”, кафедра ІСМ

ВИКОРИСТАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПРОМІЖНОГО ПРОШАРКУ (MIDDLEWARE) ДЛЯ РОЗРОБКИ МЕРЕЖЕВОЦЕНТРИЧНИХ ЗАСТОСУВАНЬ

Проведено огляд сучасних інформаційних технологій розподілених обчислень для розробки мережевоцентричних застосувань, зокрема ПЗ проміжного прошарку та архітектури мережевих обчислень. Розглянуто та проаналізовано існуючі стандарти для розподілених об'єктів, опрацювання транзакцій і служб каталогів.

Обчислювальне середовище розподілених мережевоцентричних застосувань може містити множину різноманітних операційних систем, апаратних платформ, комунікаційних протоколів, баз даних і різноманітних засобів розробки. Отже, розробка розподілених корпоративних застосувань усе більш ускладнюється, оскільки необхідно інтегрувати технології різних виробників, успадковані програмні засоби й прикладні програми та сучасні програмні пакети. Інформаційна система усе частіше містить модулі, розроблені різними виробниками ПЗ [3, 4].

Щоб створити потужне інтегроване рішення з можливістю масштабування, необхідна інфраструктура, що об'єднає різні модулі і дозволить їм взаємодіяти один з одним, не заглиблюючись у тонкощі реалізації комунікацій. Основа такої інфраструктури – **проміжне ПЗ (middleware, MW)**, бурхливий розвиток якого пов'язаний саме з новими задачами розробки інтегрованих мережевоцентричних систем. Різні типи MW обслуговують програми з різними вимогами до міжкомпонентних комунікацій. Крім того, об'єктна орієнтованість прикладних розробок і побудова застосувань із готових компонентів стимулює розвиток нових об'єктних рішень проміжного прошарку.

Сучасна індустрія MW розвивається навколо середньої ланки (middle-tier) моделі клієнт-сервер, а еволюція систем MW йде в напрямку їх ускладнення. Спочатку призначення MW обмежувалося побудовою високого рівня абстракції для взаємодії програми з ресурсами даних або різноманітних компонентів програми між собою. Розробники мали можливість використовувати загальні API, що приховували розходження специфічних інтерфейсів комунікаційних протоколів більш низького рівня, наприклад, TCP/IP Sockets, SNA LU6. 2 або DECNet. Нині цього вже явно недостатньо для побудови складних мережевоцентричних застосувань. Сучасні рішення MW не тільки забезпечують міжпрограмну взаємодію, але й реалізують платформу для сервера застосувань (середня ланка), забезпечуючи великий набір необхідних служб: керування транзакціями, найменування, захисту тощо.

MW відповідає за можливість обміну різнорідною інформацією. Формат подання даних на мейнфреймах відрізняється від подання в Unix- або Windows-системах, тому прозоре для користувача перетворення даних також входить до завдання MW. Отже, у розподіленому неоднорідному середовищі MW відіграє роль “інформаційної шини”, надбудованої над мережевим рівнем, яка забезпечує доступ програми до різнорідних ресурсів, а також незалежний від платформ взаємозв'язок різноманітних прикладних компонентів.

Весь існуючий на сьогодні спектр продуктів MW можна розділити на два основних типи – ПЗ доступу до баз даних і ПЗ міжпрограмної взаємодії. До останнього належать:

- ✓ системи, що реалізують виклик віддалених процедур (RPC);
- ✓ монітори опрацювання транзакцій (transaction processing monitors – TP-монітори);
- ✓ засоби інтеграції розподілених об'єктів;
- ✓ MW, орієнтоване на опрацювання повідомлень (MOM).

Розглянемо ПЗ міжпрограмної взаємодії докладніше.

Розподілені об’єкти

З усіх рівнів, що знаходяться вище базового мережевого рівня, найбільш фундаментальну роль у розподілених обчисленнях виконує той, що забезпечує обмін викликами між програмами. Вже протягом декількох років існують стандарти для дистанційного виклику процедур (remote procedure call – RPC), серед яких слід зазначити специфікацію DCE RPC (Distributed Computing Environment – розподілене обчислювальне середовище) організації Open Group, що забезпечує виконання універсальних крос-платформних дистанційних викликів.

Об'єктна орієнтація, покликана спростити задачу завдяки укладанню складних обчислювальних елементів в оболонку функціональних об'єктів, являє собою кращий підхід до побудови розподілених застосувань, ніж використання викликів RPC.

RPC підтримує синхронний режим комунікацій між двома прикладними модулями (клієнтом і сервером). Для встановлення зв'язку, передавання виклику і повернення результату клієнтський і серверний процеси звертаються до спеціальних компонентів – клієнтського і серверного сурогатів (client stub і server stub). Ці stub-процедури не реалізують прикладної логіки і призначені тільки для організації взаємодії віддалених (у загальному випадку) прикладних модулів. Кожна функція на сервері, що може бути викликана віддаленим клієнтом, повинна мати такий сурогатний процес. Якщо клієнт викликає віддалену процедуру, виклик разом із параметрами передається клієнтському сурогату. Він упаковує ці дані в мережеве повідомлення (цей процес називається marshaling) і передає його серверному сурогату. Той, у свою чергу, розпаковує отримані дані (unmarshaling), передає їх реальні функції серверу і потім пробробляє обернену процедуру з результатами. Отже, процедури-сурогати ізолюють прикладні модулі клієнта і сервера від рівня мережевих комунікацій.

За своїм змістом, RPC реалізує в розподіленому середовищі принципи традиційного структурного програмування. Клієнт звертається до процесу-сурогату так, начебто він і є реальний серверний процес, і цей виклик нічим не відрізняється від виклику локальної функції. Як і у випадку нерозподіленої програми, виклик процедури на віддаленому комп'ютері спричиняє передачу керування цій процедурі, тобто блокує виконання клієнтської програми на час опрацювання виклику. Доцільність використання об'єктної технології зростає зі складністю розподілених Web-систем. У даній галузі вже існує декілька важливих стандартів, до певної міри узгоджених між собою. Їх архітектура має багато спільного, тому у перспективі можна очікувати появи потужних засобів взаємодії.

Фірми IBM, Microsoft і Netscape значною мірою зв'язали своє майбутнє в Internet із моделями розподілених об'єктів. Розподілені об'єкти не належать до тих, із якими програмісти регулярно мають справу. Об'єкти розподіленої моделі зі спільними об'єктами DCOM (Distributed Common Object Model) або архітектури керування об'єктами OMA (Object Management Architecture) існують на системному рівні. Локальні об'єкти в інших частинах системи або мережі можуть взаємодіяти з ними як розподілені. Вони можуть бути

наділені здатністю отримувати інформацію про можливості об'єктів і успадковувати їх для наступного власного використання. Системні об'єкти моделей OMA і DCOM можуть бути реалізовані багатьма мовами програмування. Компанія Oracle вирішила об'єднати всі ці й інші стандарти в єдину архітектуру мережевих обчислень Network Computing Architecture (NCA). Розглянемо це докладніше.

Розподілена модель зі спільними об’єктами

DCOM є розподіленим варіантом COM-моделі, що лежить в основі технологій OLE і Active X. Ця об'єктна модель реалізована в середовищі Windows і на платформах Macintosh і UNIX [4].

Хоча для моделі COM існують потужні інструментальні засоби розробки, створення ефективної програми COM або DCOM може виявитися важким завданням. У високопродуктивний серверний об'єкт, наприклад, повинна бути закладена інформація про складні потокові моделі. У такому випадку можна використати сервер транзакцій Transaction Server фірми Microsoft.

Програмна модель DCOM подібна до моделі COM, проте для перенесення програм COM на платформу DCOM доведеться здійснити додаткове опрацювання. Оскільки відомо, що будь-який об'єкт COM працюватиме на будь-якій реалізації моделі COM, вже існує відкритий ринок COM-об'єктів. Модель DCOM працює також через протоколи без установлення логічного з'єднання (такі, як UDP і IPX). У моделі OMA ці протоколи не використовуються.

Архітектура керування об’єктами

Специфікацію OMA організації Object Management Group часто називають CORBA (Common Object Request Broker Architecture – узагальнена архітектура з посередниками запитів до об'єктів), хоча насправді CORBA – лише частина архітектури OMA. До складу OMA входить протокол взаємодії в Internet (Internet Interoperability Protocol – IIOP) та інші функціональні служби [4, 5].

Структура CORBA відповідає трирівневій архітектурі. На першому рівні знаходиться фронтальний Web-клієнт, призначений для реалізації взаємодії з користувачем. На другому рівні знаходиться Web-сервер, який містить документи HTML, аплети і посередники об’єктних запитів (ORB – object request broker). Третій рівень складається з кінцевих систем, які або надають інформацію Web-серверу на основі запитів, або опрацьовують різноструктуровану інформацію в інтерактивному режимі.

Постачальники реалізують архітектуру CORBA на багатьох платформах, у тому числі SOMobjects фірми IBM (одна з реалізації належить компанії Netscape). Архітектура CORBA схожа на структуру DCOM, проте в CORBA доданий новий рівень: Interface Definition Language (IDL – мова визначення інтерфейсу). IDL служить для опису інтерфейсу об'єкта без зв'язку з його реалізацією. Програмісти компілюють вихідний текст IDL у інтерфейсний код, що забезпечує з'єднання з посередником об'єкта і код реалізації об'єкта на обох системах. Даний вихідний текст може бути реалізований багатьма мовами, і механізми перетворення типів даних між вихідним текстом реалізації і IDL є частиною стандарту OMA.

Архітектура мережевих обчислень

Корпорація Oracle розробила архітектуру мережевих обчислень Network Computing Architecture (NCA), спираючись на такі постулати: жодна окремо узята технологія або стан-

дарт не буде і не зможе займати чільне місце в майбутньому. Потреба у будь-якому інформаційному забезпеченні буде задовольнятися тільки за допомогою комбінування нових і вже існуючих технологій.

Архітектура мережевих обчислень забезпечує уніфіковану, засновану на стандартах архітектуру для технологій "клієнт-сервер", Web і розподілених об'єктів. Підтримка основних відкритих стандартів гарантує розробникам можливість вибору клієнтської частини, мови програмування (яка підходить для даної розробки) і відповідної моделі програмування [1-5].

NCA забезпечує середовище для об'єднання будь-якої мови програмування або архітектури компонентів. Можна навіть створювати шлюзи мейнфреймів для взаємодії з CICS. Крім того, можна створювати інтерфейси практично для будь-якої моделі програмування (картриджі Java і Perl). Основа архітектури NCA спирається на відкриті стандарти. Комутатор картриджів Inter-Cartridge Exchange (ICX), що підтримує розподілене опрацювання, незабаром стане цілком сумісним із стандартом CORBA. Архітектура NCA підтримує також незалежну від мов програмування мову визначення інтерфейсу Interface Definition Language (IDL), що дозволяє використовувати для розробки практично будь-які мови й засоби програмування.

Основними компонентами архітектури мережевих обчислень є:

- під'єднувані об'єкти (картриджі);
- шина міжкартриджного обміну;
- сімейство клієнтів;
- універсальний сервер застосувань;
- універсальний сервер БД

Основним стандартним блоком сервера застосувань Web Application Server компанії Oracle є програмний модуль, що називається картриджем. Картриджі приймають вхідні дані, опрацьовують їх і повертають результат. Компанія Oracle визначила модель їх роботи і зробила її загальнодоступною.

Картриджі забезпечують конкретні функціональні можливості, а також обслуговування таких ресурсів, як пам'ять. Хоча сервер застосувань Web Application Server постачається разом із декількома попередньо розробленими картриджами, розробники інформаційних систем також можуть використовувати дану архітектуру для створення власних картриджів. Переваги NCA реалізуються в розподіленому середовищі. Велика частина типових мережевоцентричних застосувань може бути опрацьована на сервері, зводячи тим самим до мінімуму необхідність передачі великих обсягів даних від клієнта до сервера.

У картриджах використовуються всі три рівні Web-застосування. Вони можуть бути розроблені для виконання на клієнтському комп'ютері, на сервері або в самій базі даних. Подібна гнучкість дає розробнику можливість застосовувати багато ефективних методик.

Опрацювання транзакцій

Для організації розподілених обчислень потрібно, щоби дії різних комп'ютерів були взаємопов'язані. Опрацювання транзакцій починається з технології баз даних для виконання послідовності взаємозалежних операцій. В ефективній системі опрацювання транзакцій повинен існувати механізм для спрощення розробки складних програм.

У ситуації, коли користувачам надане право доступу до певних ресурсів через мережу, забезпечення безпеки стає непростим завданням. Адміністратори повинні мати можливість керувати та контролювати доступ до цих ресурсів, і служби каталогів були розроблені для того, щоб призначати категорії захисту пристроїв мережі (таких, як клієнти і сервери).

Транзакція являє собою послідовність операцій, що визначається як єдина логічна група [5, 6]. Якщо якась з операцій виявляється невдалою, то немає потреби виконувати й подальші. Прикладом транзакції є купівля за кредитною карткою. Якщо ваша кредитна спроможність не буде підтверджена, то банк не перерахує гроші на рахунок підприємства роздрібної торгівлі.

Серверні й хост-процеси задовольняють вимоги стандартів, які потрібні для коректної роботи таких систем, у тому числі архітектури X/Open XA і частини специфікації LU6.2 фірми IBM. Програміст ініціює транзакцію, визначає операції і (якщо не виникає перешкод із боку серверів) виконує їх. При виникненні перешкод у ході будь-якої операції в програмі передбачений *"відкат"* (rollback). Така схема має назву *двофазного виконання з (обопільним) контролем* (two-phase commit) і керує нею система опрацювання транзакцій (transaction processing – TP). У нашому прикладі з кредитною карткою, якщо на рахунку допущена перевитрата коштів або не підтверджений кредит магазину, то система TP повинна скасувати всі інші операції.

Розподілені системи є структурами, схильними до відмов. Хоча добре спроектовані розподілені операції рідко проходять за участю більш ніж трьох різноманітних систем, кількість потенційних точок відмови набагато більша ніж в одній системі. Належить врахувати також, що розподілені об'єкти являють собою клієнтські і серверні процеси. Перераховані чинники утруднюють проектування високопродуктивних серверних об'єктів.

Протягом багатьох років у корпоративних системах, побудованих на базі великих ЕОМ і розрахованих на виконання відповідальних завдань, для керування такими операціями використовуються дорогі фірмові програми – TP-монітори. Завдяки високопродуктивним серверним операційним системам, новим мережним комунікаційним стандартам, більш досконалим інструментам розробки і можливості розподіляти об'єктні моделі, що лежать у фундаменті TP-систем, серійні ПК спроможні справитися із завданнями, що раніше були під силу лише універсальним машинам (мейнфреймам).

Сервери застосувань

Поява серверів застосувань як окремих готових рішень пов'язана і з бурхливим вторгненням Web у сферу корпоративних висококритичних систем. У середині 90-х років Web стає платформою для розгортання застосувань електронної комерції. Здавалося б, Internet може стати універсальним механізмом комунікацій між прикладними компонентами, ресурсами даних і клієнтами, а Web-сервер можна використати як проміжну ланку розподіленого Internet-застосування. Проте можливості протоколу HTTP обмежені зв'язком без засобів зберігання інформації про стан, тому він не дозволяє реалізувати динамічні Web-сторінки й організувати інтерактивний режим взаємодії з клієнтом. CGI-механізми надто повільні для підтримки потужних корпоративних систем і погано підходять для роботи з об'єктами. Тому з'являються перші інтегровані системи, що виступають у ролі посередника між різноманітними клієнтами, у тому числі Web-браузерами і різнорідними джерелами даних. Часто в обхід традиційних Web-серверів вони реалізують різноманітні сервіси для підтримки прикладної логіки і спираються на різноманітні механізми комунікацій традиційних систем проміжного прошарку.

Індустрія серверів застосувань має на меті створення об'єктно-орієнтованих розподілених систем і в найближчій перспективі – побудову прикладних програм із готових компонентів. Тому існуючі на сьогодні розробки базуються на двох компонентних архітектурах – MTS/DCOM (у перспективі COM+) і CORBA/EJB.

1. Philip J. Gill. Enabling the Age of Network Computing // Oracle magazine. November/December, 1997. 2. Philip J. Gill. Network Computing Architecture // Oracle magazine. November/December, 1997. 3. Philip J. Gill. The Network ComputingRule Book: Playing to Win // Oracle magazine. November/December, 1997. 4. Архитектура распределенных приложений. По материалам Gupta Corp // Компьютеры+Программы. 1995. № 5. 5. Голощук Р.О. Огляд інформаційних технологій розподілених обчислень для розробки мережевоцентричних застосувань // Вісн. ДУ "Львівська політехніка". 1998. № 383. С.62-68. 6. Пелещишин А. М. Розробка інформаційних систем у Web-середовищах // Вісн. ДУ "Львівська політехніка". 1997. № 315. С.193-207

УДК 681.32:681.5

Дубровін В., Субботін С.
Запорізький державний технічний університет

ВИБІР ФУНКЦІЙ АКТИВАЦІЇ ФОРМАЛЬНОГО НЕЙРОНА ТА ДОСЛІДЖЕННЯ ЇХ ВПЛИВУ НА ЯКІСТЬ НАВЧАННЯ НЕЙРОННИХ МЕРЕЖ

Розглянуто модель формального нейрона. Досліджено загальні та особисті вимоги до функцій активації і вироблено критерії їх порівняння. Наведено дані про результати експериментів із вибору найкращих функцій активації.

Вступ

Необхідність усунення не самих невідповідностей, а їх причин – і насамперед потенційних – завжди підкреслювалася засновниками Total Quality Management. Значно краще – знайти потенційні проблеми якомога раніше та знищити на самому початку, до того, як вони зможуть створити реальні труднощі. Тому при управлінні якістю акцент зміщується з перевірки продукції на діагностику виробничих процесів [1].

При будівництві адаптивних систем діагностики перспективним є використання нейронних мереж (НМ). Однією з важливих проблем у галузі штучних НМ є розробка ефективних методів та алгоритмів навчання НМ. На жаль, сьогодні ця проблема залишається невирішеною через те, що більшість алгоритмів навчання (особливо, для багатошарових НМ) повільно збігаються. Тому особливу значимість для практичної реалізації систем на основі НМ можуть мати дослідження оптимізації окремих параметрів НМ. Одним з таких параметрів, що впливає як на швидкість, так і на точність навчання, є тип функції активації формального нейрона НМ, що досліджується в цій роботі.

Модель формального нейрона

Формальный нейрон є основним елементом при будівництві НМ. Він виконує параметричне нелінійне перетворення вхідного вектора x в скалярну величину y. Це перетворення складається з двох етапів: спочатку обчислюється дискримінантна функція, що є відрізком багатовимірного ряду Тейлора, яка далі перетворюється у вихідну величину y. Коефіцієнти розкладання відрізка багатовимірного ряду Тейлора утворюють вектор вагових коефіцієнтів w, або пам'ять нейрона.