УДК 658.012.011.56

Н.М. Сидорчук

АРХІТЕКТУРНІ ТА СИСТЕМОТЕХНІЧНІ ПІДХОДИ ДО КОНСТРУЮВАННЯ УКРАЇНСЬКОГО НАЦІОНАЛЬНОГО ЛІНГВІСТИЧНОГО КОРПУСУ

1. Архітектура програмної системи УНЛК

В інформаційних проектах великого масштабу вибір генеральної (технологічної чи технічної) лінії є одним із основних питань, що визначає успіх впровадження та ефективність використання системи. Як детальний план будь-якого проекту, найпершим аспектом, який слід проаналізувати на предмет адекватності забезпечення якісних властивостей інформаційної системи виступає її архітектура. Під архітектурою програмної системи, розгляд якої поданий нижче, ми розуміємо опис всіх структур системи (а саме структури декомпозиції на модулі, структури процесів, локалізацій, рівнів тощо).

Вибір програмної архітектури під час проектування системи УНЛК здійснювався з огляду на основні вимоги до системи, а саме:

— ефективне опрацювання великих текстових масивів;

— забезпечення паралельного обслуговування великої кількості клієнтів;

— розподілення функцій системи за різними групами користувачів;

— забезпечення масштабованості системи;

— вимоги до високого рівня відмовостійкості системи;

— забезпечення надійності збереження та обміну даними;

— проведення ресурсоємних обчислень та обробки даних;

— ефективного функціонування розподілених застосувань в глобальній мережі в онлайновому режимі.

Під час аналізу поставлених вимог до системи вибір розподіленої архітектури став очевидним. Така технологія забезпечує централізоване збереження та обробку даних, надає можливість розподіленого введення даних, вирішує проблему обмеження доступу до ресурсів, надає можливість використання потужних обчислювальних можливостей серверу.

При альтернативі між дворівневою і багаторівневою системою ми зупинили свій вибір на багаторівневій архітектурі. Причиною відмови від дворівневої архітектури була проблема масштабованості для дворівневої архітектури «клієнт-сервер», а це суперечить вимогам до системи, яка розробляється. Безперечно, реалізація дворівневої архітектури вимагає менше затрат на розробку і легша в реалізації, адже «клієнт–серверна» архітектура взагалі передбачає наступний механізм взаємодії. Клієнтський процес робить запити на обслуговування до серверного, в результаті якого обмінюється з ним певними наборами даних. У клієнтському процесі виконується перевірка введених користувачем даних, виконання запиту до серверу, та, можливо, виконання певних бізнес-правил. Клієнт відповідає за рівень подання цих даних. «Сервер» діє як механізм реалізації клієнтських запитів шляхом виконання прикладних програм та взаємодії з такими ресурсами, як бази даних та файлові системи. На відміну від дворівневих систем, які складаються лише з рівня подання та рівня збереження даних, у трирівневій системі, на якій ми зупинили свій вибір, вводиться проміжний рівень, на який виноситься рівень бізнес-логіки (рис. 1).

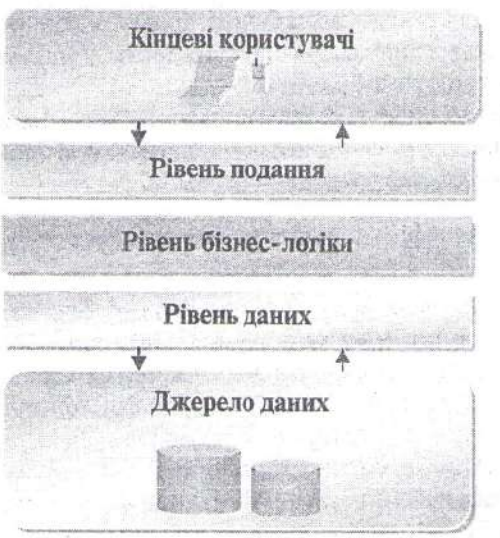

Рис. 1. Архітектура програмного комплексу

У цьому випадку проміжний рівень перевіряє правильність даних, що передаються від клієнта та обробляє їх у відповідності до бізнес-правил. Ця обробка може взаємодіяти з рівнем даних або виконувати локальні обчислення чи перетворення, в результаті яких результати передаються на рівень даних для збереження, або ж на рівень подання (клієнтський). Крім цього використання такої архітектури дає можливість логічного розподілення функцій системи, що в свою чергу забезпечує можливість розподілення роботи між різними розробниками, можливість розробляти окремо кожен рівень, переносити на інші сервери в залежності від вимог масштабованості. Зосередження бізнес-логіки на проміжному рівні дозволяє модифікувати її,

не змінюючи клієнтські системи та інформаційні масиви. І навпаки, з'являється можливість розробки різних клієнтських програм, що використовують один і той самий рівень бізнес-логіки. Отже, вибір такої архітектури забезпечує досягнення таких переваг в порівнянні з дворівневою:

— відмовостійкість. Природа розподілених застосувань дозволяє будувати відмовостійкі програмні системи застосовуючи такий спосіб як збитковість. Розподілення копій функціональних модулів, або копій об'єктів по різних вузлах збільшує ймовірність того, що збій на одному з вузлів не викличе збою в роботі об'єктів на іншому вузлі і система вцілому зможе продовжити роботу;

— масштабованість. Здатність системи витримувати збільшення навантаження з мінімальним зменшенням продуктивності забезпечується шляхом розподілення різних функціональних частин застосування по різних вузлах. Це скорочує обсяг обробки, що виконується одним вузлом і дозволяє виконувати великі обсяги роботи паралельно;

— адміністрування. Ефективне керування апаратною і програмною конфігурацією великої мережі ПК питання досить актуальне. При багаторівневій архітектурі зміна бізнес-правил здійснюється на сервері, що дозволяє клієнтські застосування практично не змінювати.

Отже, вибір такої архітектури є оптимальним як з точки зору масштабування, так і гнучкості підтримки та розповсюдження програмної системи.

2. Специфіка функціонування онлайнової версії УНЛК

Онлайнова версія УНЛК має розроблюватися з урахуванням обмежень, зумовлених середовищем функціонуванням, загальноприйнятих стандартів роботи розподілених систем в глобальній мережі, різнорідного програмного забезпечення кінцевих користувачів, специфіки обміну інформацією, вимог забезпечення захисту і надійності роботи системи. На продуктивність розподіленого застосування впливає велика кількість факторів. Наприклад, це мережна швидкість, завантаженість мережі, апаратні проблеми, пов'язані з окремими комп'ютерами в мережі (проблеми процесорів, розмір і швидкість оперативної пам'яті).

Незважаючи на те, що загальнодоступний Інтернет існує уже досить давно, до недавніх пір він застосовувався в основному для передачі файлів, електронної пошти та перегляду HTML-сторінок, які надавалися Web-серверами. Хоча розвиток технологій розподілених застосувань має значний прогрес, однак специфіка їх застосування, в основному, зорієнтована на роботу в закритих мережах. Більшість розробників не використовували Інтернет як мережу для роботи розподілених застосувань. З розвитком технологій компанії почали захищати свої внутрішні мережі від трафіка крім HTTP, зазвичай лише через порт 80. Можливо, на даний момент справедливо стверджувати, що більшість клієнтів працюють за протоколом HTTP. Це не означає, що HTTP настільки ефективний. Це просто угода, якої досягнуто в зв'язку зі зростанням популярності Інтернету. Як правило, старі мережні формати даних і протоколи вимагають відкритого доступу через брандмауери до незахищених портів. Крім того, подібні формати і протоколи інколи використовують не один порт, а декілька портів або навіть діапазон номерів портів. Спеціалісти з забезпечення захисту систем вважають, що відкриття доступу через брандмауери на пропуск подібного трафіка фактично суперечить їх прямому призначенню. Тому для забезпечення функціонування розподілених застосувань в глобальній мережі без обмежень на настройки брандмауерів стало зв'язування закритих мереж з використанням HTTP. Таке завдання вирішувалося і вирішується до сьогодні за допомогою тунелювання нестандартних мережних форматів через HTTP, для чого створюються спеціалізовані клієнти та сервери або використовуються Web-сервери. Але такий варіант не дуже привабливий, адже насправді, така технологія являє собою трудомісткі «вставки» до розроблених програмних засобів з метою подолання обмежень мережевих форматів даних. Стало зрозуміло, що використання спеціалізованих форматів в Інтернеті не задовольняє вимоги ефективного функціонування онлайнових систем. Загальноприйнятим стандартом для проникнення через брандмауер стала розробка розподілених застосувань, які функціонують з використанням HTTP.

3. Програмні засоби розробки системи

Розробка системи УНЛК проводиться на базі платформи .NET. Це зумовлено наявністю :

— ієрархічної множини уніфікованих бібліотек класів;

— потужної та надійної технології доступу до різних сховищ даних ADO.NET. За даною технологією, використовуючи OLE DB-провайдерів, .Net-провайдерів або XML, отримується доступ до джерел даних. Так з реляційними даними можна виконувати всі необхідні операції (вибору, додавання, обновлення, видалення і т.д.). З'являється можливість створення проміжних колекцій із запитів як до гетерогенних так і розподілених баз даних;

— засобів створення багаторівневих застосувань .NET Remoting. З використанням .NET Remoting виникає можливість взаємодії об'єктів, що знаходяться в різних процесах чи доменах застосувань з використанням різних транспортних протоколів, різних форматів серіалізації, схем життєвого циклу об'єктів та способів створення об'єктів;

— можливості використання будь-яких мов програмування, які відповідають специфікації CLS (Common Language Specification);

— можливості розробки розподілених застосувань для глобальної мережі (он-лайн систем).

Механізм функціонування в глобальній мережі розглянемо в наступному пункті.

4. Особливості програмної платформи для розробки он-лайн систем

Під час розробки онлайнових систем серед вимог до функціонування на перший план постає відкритість системи. Відкритість .NET Remoting досягається за рахунок підтримки відкритих стандартів, таких як HTTP, SOAP, WSDL та XML. .NET Remoting пересилає серіалізовані об'єкти-повідомлення через межі .NET Remoting за допомогою каналів. Об'єкти-канали по обидві сторони межі надають транспортний механізм, що володіє потужними можливостями підтримки найрізноманітніших протоколів та мережних форматів. Інфраструктура .NET Remoting пропонує два стандартних типи каналів, які можна використовувати в розподілених застосуваннях: TCP і HTTP. Є можливість створення власного транспортного механізму та підключення його до інфраструктури .NET Remoting. Розглянемо детальніше стандартні типи каналів.

TCP. У випадку, коли вимагається максимальна ефективність інфраструктури .NET Remoting пропонує транспорт на основі сокетів, що використовує для передачі серіалізованого потоку повідомлень через межі .NET Remoting протокол TCP. У просторі імен System.Runtime.Remoting.Channeis визначено тип каналу TcpChannel. Tcp реалізує інтерфейси IChannel, IChannelReceiver і IChannelSender, а це означає, що тип TcpChannel підтримує як відправлення так і оримання даних через межі .NET Remoting.

HTTP. У тому випадку, коли вимагається максимальна відкритість (так як у випадку он-лайн систем), NET Remoting пропонує транспорт, що використовує протокол HTTP для передачі серіалізованого потоку повідомлень через Інтернет і через брандмауер. Функціональність транспорту HTTP реалізовано в типі HttpChannel, який визначений в просторі імен System.Runtime.Remoting.Channels.Http. Як і TcpChannel, канал HttpChannel приймає і передає дані через межі .NET Remoting. Для серіалізації об'єктів-повідомлень канал Http Channel використовує мережний формат SOAP (Simple Object Access Protocol). Формат SOAP являє собою XML-"конверт" із запитом (відповідь на запит), у якому зазначені засоби подання XML-структури та засоби зв'язку за протоколом HTTP.

Отже, в залежності від поставлених цілей можна застосовувати два найбільш розповсюджені сценарії використання .NET Remoting:

— канал HTTP/SOAP, що забезпечує максимальну відкритість;

— канал TCP/бінарні дані – забезпечує максимальну продуктивність.

Забезпечення обох конфігурацій виконується за допомогою простого підключення необхідних модулів.

5. Взаємодія між розподіленими об'єктами

.NET Remoting дозволяє об'єктам, що функціонують всередині різних доменів застосувань або контекстів, взаємодіяти між собою через межі .NET Remoting. Межа.NET Remoting в деяких випадках дозволяє екземпляру пройти через неї без змін, в інших – примушує екземпляр об'єкта за межами домену чи контексту взаємодіяти з внутрішніми об'єктами за точно визначеним протоколом. З точки зору інфраструктури .NET Remoting-об'єкти поділяються на дві категорії: ті, що дистанціюються та ті, які не дистанціюються. До першої категорії належать об'єкти, для яких виконуються хоча б одна з таких умов як:

— екземпляри типу здатні проходити межі NET Remoting;

— інші об'єкти можуть звертатися до екземплярів даного типу через межі .NET Remoting.

І навпаки, якщо тип не володіє жодним з перерахованих властивостей, то його відносять до другої категорії. У свою чергу, об'єкти, що дистанціюються, поділяються на три категорії за типом зв'язку: за значенням, за посиланням, контекстно-пов'язані.

Передача за значенням. Обмін об'єктами, що передаються за значенням, відбувається за допомогою процесу серіалізації (подання поточного стану об'єкта у вигляді послідовності біт). Після серіалізації об'єкта інфраструктура .NET Remoting передає отриману послідовність біт через межі .NET Remoting в інший домен застосування або контекст, де інфраструктура виконує їх десеріалізацію в екземпляр типу, що є точною копією початкового стану.

Передача за посиланням. Інколи необхідно, щоб усі виклики об'єкта, що створений в деякому домені застосування, зверталися саме до екземпляра в даному домені, а не до його копії в іншому домені. Наприклад, об'єкту можуть бути необхідні ресурси, доступні тільки об'єктам, що виконуються на даному комп'ютері. У такому випадку використовуються типи, що передаються за посиланням, для яких інфраструктура .NET Remoting передає посилання на екземпляр об'єкта, а не його серіалізовану копію.

Контекстно-пов'язані об'єкти. Екземпляри даного типу мають залишатися в межах певного контексту. А зовнішні до даного контексту об'єкти не мають безпосереднього доступу до типу контестно-пов'язаного, навіть якщо вони знаходяться в одному і тому ж домені застосувань.

6. Активація об'єктів

Перш ніж працювати з екземпляром типу, що дистанціюється, його необхідно створити і ініціалізувати, виконавши так звану процедуру активації. В .NET Remoting типи, що передаються за посиланням, підтримують два види активації: клієнтську та серверну. .NET Remoting підтримує два режими серверної активації *Sigleton* та *SingleCall*.

Режим Singleton. В окремий момент часу може бути активним не більш ніж один екземпляр об'єкта, конфігурація якого Singleton. Єдиний екземпляр активізується при першому клієнтському зверненні до нього. До тих пір поки цей екземпляр активний, він оброблює всі наступні запити, як від того ж клієнта, так і від інших. Екземпляр типу Singleton може зберігати стан в проміжках між викликами його методів.

Режим SingleCall. Інший режим серверної активізації — SingleCall — забезпечує підтримку моделі програмування без збереження проміжного стану. Для типу, сконфігурованого як SingleCall, інфраструктура .NET Remoting активізуватиме новий екземпляр при кожному зверненні клієнтом до методів цього типу. Після того як звернення до методу оброблено, інфраструктура .NET Remoting робить екземпляр типу доступним для збиральника сміття. У випадку, коли необхідно, щоб кожен клієнт працював з окремим екземпляром віддаленого об'єкта в інфраструктурі .NET Remoting передбачено клієнтську активацію. Екземпляри типу з клієнтською активацією можуть залишатися активними в проміжках між зверненнями методів і використовувати ту ж схему управління часом життя, що і в Singleton з єдиною відмінністю, що кожен клієнт посилається на окремий екземпляр дистанційованого типу.

7. Час існування віддаленого об'єкта

Мережі за своєю природою ненадійні. Клієнтські застосування «падають», користувачі залишають свої робочі місця, а мережі, інколи втрачають свою працездатність.

Для управління часом життя віддалених об'єктів .NET Remoting використовує варіант розподіленого збирання сміття на основі ліцензій (lease). Ліцензія – це об'єкт, що інкапсулює значення проміжку часу, який використовується інфраструктурою .NET Remoting для визначення часу існування віддаленого об'єкта. Вибір даної схеми управління часом життя об'єкта продиктовано специфікою роботи розподіленої системи. Розглянемо ситуацію, коли достатньо велика кількість клієнтів взаємодіє з сервером у режимі Singleton. Застосовуючи схему управління часом життя, що не використовує ліцензії, для визначення необхідності ліквідації об'єкта проводиться опитування клієнтів та підраховування посилань на об'єкт. У подібній схемі мережний трафік, пов'язаний з опитуванням клієнтів інколи негативно впливає на роботу розподіленого застосування вцілому. На противагу такій схемі система управління часом існування на основі ліцензій використовує ліцензії, спонсори та диспетчер ліцензій. Відсутність у такій схемі опитування дозволяє збільшити загальну продуктивність застосування.

Розглянемо загальну схему управління часом існування віддаленого об'єкта на основі ліцензій. У кожному домені застосувань існує диспетчер ліцензій. Він містить посилання на об'єкт-ліцензію для кожного об'єкта Singleton з серверною активацією або для кожного об'єкта з клієнтською активацією, що існують в домені застосування цього диспетчера. З кожною ліцензією пов'язано один або більше спонсорів, які здатні продовжувати ліцензію, коли диспетчер ліцензій визначає, що термін її існування закінчився (диспетчер періодично перевіряє список ліцензій, порівнюючи час закінчення дії ліцензії з поточним часом).

Висновки

Зазначимо, що вибір поданої багаторівневої програмної архітектури та використання інструментальних засобів платформи .Net Remoting стали базою для розробки та запуск у промислову експлуатацію в Українському мовно-інформаційному фонді НАН України лексикографічну систему УНЛК. На базі розробленої системи виконуються різноманітні лінгвістичні дослідження, програмний комплекс використовується як додатковий інструмент під час розробки нових лексикографічних систем.

Список літератури: 1. *Широков В.А* Інформаційна теорія лексикографічних систем. – К.: Довіра, 1998. – 331 с. 2. *Широков В.А.* Феноменологія лексикографічних систем. – К.: Наук. думка, 2004. – 327 с. 3. *Широков В.А.* та ін. Корпусна лінгвістика : Монографія / Широков В.А., Бугаков О.В., Грязнухіна Т.О., Любченко Т.П., Рабулець О.Г., Сидоренко О.О., Сидорчук Н.М., Шевченко І.В., Шипнівська О.О., Якименко К.М.; Український мовно-інформаційний фонд НАН України. — К. : Довіра, 2005. – 472 с. 4. *Сидорчук Н.М.* Онлайнові лексикографічні системи (на прикладі системи „Словники України") // 2-й Международный радиоэлектронный форум «Прикладная радиоэлектроника. Состояние и перспективы развития» МРФ-2005. Сб. Научных трудов. Т.3. Международная конференция «Информационные системы и технологии». – Харьков: АНПРЭ, ХНУРЭ. 2005. – С. 39–42. 5. *Басс Л.*, Клементс П., Кайман Р. Архитектура программного обеспечения на практике. 2-е издание. – СПб.: Питер, 2006. – 575 с. 6. *Маклин С.* и др. Microsoft. NET Remoting: Пер. с англ.; Нафтел Дж.; Уильмс К. — М. : Русская Редакция, 2003. – 384 с.

Поступила в редколлегию 07.10.2005