Технічні науки

УДК 004.054

Чанцова Катерина Вікторівна

студент

Національного технічного університету України

«Київський політехнічний інститут імені Ігоря Сікорського»

Чанцова Катерина Викторовна

студент

Национального технического университета Украины

«Киевский политехнический институт имени Игоря Сикорского»

Chantsova Kateryna

Student of the

National technical university of Ukraine

«Igor Sikorsky Kyiv Polytechnic Institute»

РОЗПОДІЛЕНІ СИСТЕМИ, ПОНЯТТЯ ТА АРХІТЕКТУРА
РАСПРЕДЕЛЕННЫЕ СИСТЕМЫ, ПОНЯТИЕ И АРХИТЕКТУРА
DISTRIBUTED SYSTEMS, DEFINITION AND ARCHITECTURE

Анотація: Розглянуто поняття розподіленої системи, основні види її архітектури.

Ключові слова: Розподілена система, архітектура розподілених систем, клієнт-серверна архітектура, сервіс-орієнтована архітектура, децентралізована архітектура.

Аннотация. Рассмотрено понятие распределенной системы, основные виды её архитектуры.

Ключевые слова: Распределенная система, архитектура распределенных систем, клиент-серверная архитектура, сервис-ориентированная архитектура, децентрализованная архитектура.

Summary. Definition and architecture types of distributed systems were considered.

Key words: Distributed system, architecture of distributed systems, client-server architecture, service-oriented architecture, peer-to-peer architecture.

За твердженням Е. Таненбаума розподілену систему можна визначити, як набір з'єднаних каналами зв'язку незалежних комп'ютерів, які, з точки зору користувача деякого програмного забезпечення, виглядають єдиним цілим.

У розподілених системах функції одного рівня додатку можуть бути рознесені між декількома комп'ютерами. З іншого боку, програмне забезпечення, встановлене на одному комп'ютері, може відповідати за виконання функцій, що відносяться до різних рівнів. Тому підхід до визначення розподіленої системи, що визначає її сукупністю комп'ютерів, умовний. Для опису та реалізації розподілених систем було введено поняття програмної компоненти.

Програмна компонента - це одиниця програмного забезпечення, що виконується на одному комп'ютері в межах одного процесу, і яка надає певний набір сервісів, які використовуються через її зовнішній інтерфейс іншими компонентами, що може виконуватися як на цьому ж комп'ютері, так і на віддалених комп'ютерах. Ряд компонент користувацького інтерфейсу надають свій сервіс кінцевому користувачеві.

Ґрунтуючись на визначенні програмної компоненти, можна дати більш точне визначення розподіленої системи. Згідно з ним, розподілена система – це набір взаємодіючих програмних компонент, що виконуються на одному або декількох пов'язаних комп'ютерах і виглядають, з точки зору користувача, системи як єдине ціле.

Прозорість є атрибутом розподіленої системи. При справному функціонуванні системи від кінцевого користувача має бути приховано, де і як виконуються його запити.

Програмна компонента є мінімальною одиницею розгортання розподіленої системи. В ході модернізації системи одні компоненти можуть бути оновлені незалежно від інших компонент [1].

Архітектура – базова організація системи, втілена у її компонентах, їх відносинах між собою та оточенням, а також принципи, що визначають проектування та розвиток системи. Для кожної розподіленої системи архітектура є дуже важливою, оскільки охоплює не тільки її структурні аспекти, але і правила її використання та інтеграції з іншими системами, функціональність, продуктивність, гнучкість та надійність. Архітектура також впливає на те, яким буде інтерфейс користувача.

Архітектура клієнт-сервер

Архітектура клієнт-сервер - це базова модель організації розподілених систем. У цій моделі всі процеси діляться на дві групи. Процеси, що реалізують певну роботу, наприклад, службу файлової системи або бази даних, називаються серверами. Процеси, що запитують служби у серверів шляхом посилки запиту і подальшого очікування відповіді від сервера, називаються клієнтами. Нерідко клієнти і сервери взаємодіють через комп'ютерну мережу і можуть бути як різними фізичними пристроями, так і програмним забезпеченням [2].

Найчастіше додатки типу клієнт-сервер поділяють на три логічних (рис. 1): призначений для користувача інтерфейс (ІК), логіка додатку (ЛД) і доступ до даних (ДД), що працює з базою даних (БД). Користувач системи взаємодіє з нею через інтерфейс користувача, база даних зберігає дані, що описують предметну область додатка, а рівень логіки додатка реалізує всі алгоритми, які стосуються предметної області.

Рис. 1. Логічні рівні додатку [1]

Дволанкова архітектура клієнт-сервер

Одним з варіантів клієнт-серверної архітектури є класична дволанкова архітектура (Two-tier architecture). Під клієнт-серверним додатком в цьому випадку розуміється інформаційна система, заснована на використанні серверів баз даних.

Схематично таку архітектуру можна представити, як показано на рис. 2.

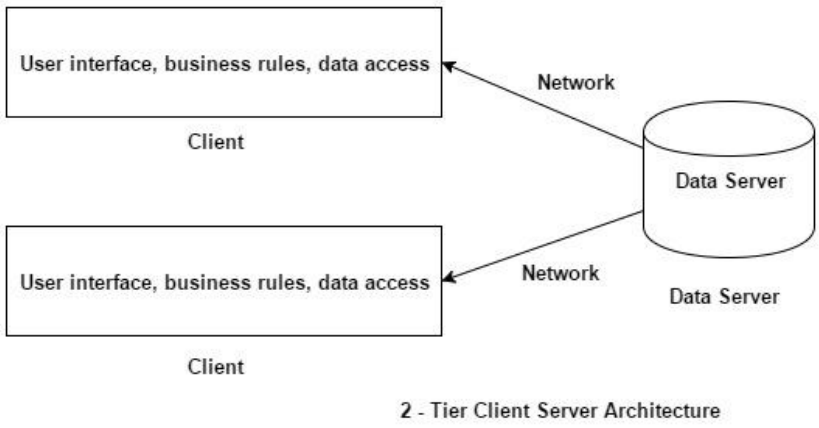

Рис. 2. Класичне уявлення архітектури клієнт-сервер

На стороні клієнта виконується код додатка, в який обов'язково входять компоненти, що підтримують інтерфейс з кінцевим користувачем, створюють звіти, виконують інші специфічні для додатку функції. Також

клієнтська частина програми взаємодіє з клієнтською частиною програмного забезпечення управління базами даних.

На практиці системи, побудовані на дволанковій архітектурі, часто не відносять до класу розподілених, але формально вони можуть вважатися найпростішими представниками розподілених систем [1].

Перевагами даної архітектури є:

- можливість, в більшості випадків, розподілити функції обчислювальної системи між декількома незалежними комп'ютерами в мережі;
- всі дані зберігаються на сервері, який, як правило, захищений набагато краще за більшість клієнтів, а також на сервері простіше забезпечити контроль повноважень, щоб дозволити доступ до даних тільки клієнтам з відповідними правами доступу;
- підтримка роботи з багатьма користувачами;
- гарантія цілісності даних.

Недоліки:

- непрацездатність сервера може зробити непрацездатною всю обчислювальну мережу;
- висока вартість обладнання;
- бізнес логіка додатків залишається в клієнтському програмному забезпеченні.

Триланкова архітектура клієнт-сервер

Найбільш поширеною є триланкова архітектура (three-tier, у якої інтерфейс користувача, логіка програми та доступ до даних виділені в самостійні складові системи, які можуть працювати на незалежних комп'ютерах (рис. 3).

Рис. 3. Триланкова архітектура [1]

Запит користувача в подібних системах послідовно оброблюється клієнтською частиною системи, сервером логіки додатка і сервером баз даних (рис. 4).

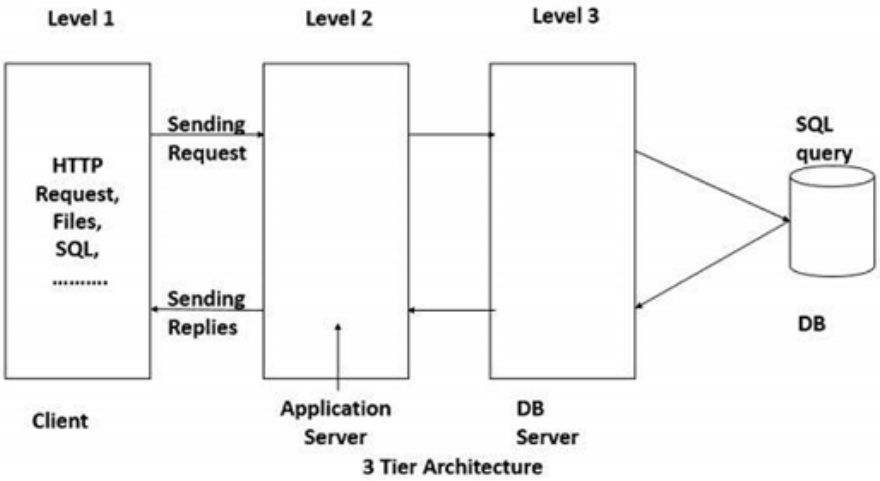

Рис. 4. Обробка запиту користувача в триланковій архітектурі
Складено автором на основі [2]

Дана архітектура має такі переваги:

- клієнтське програмне забезпечення не потребує адміністрування;
- масштабованість;
- конфігурованість - ізольованість рівнів один від одного дозволяє швидко і простими засобами переконфігурувати систему при виникненні збоїв або при плановому обслуговуванні на одному з рівнів;
- висока безпека та надійність;

- низькі вимоги до швидкості каналу (мережі) між терміналами і сервером додатків;
- низькі вимоги до продуктивності і технічних характеристик терміналів, як наслідок, зниження їх вартості.

Недоліки:

- зростає складність серверної частини і, як наслідок, витрати на адміністрування і обслуговування;
- більш висока складність створення додатків;
- більша складність в розгортанні і адмініструванні;
- високі вимоги до продуктивності серверів додатків і сервера бази даних, і, як наслідок, висока вартість серверного обладнання;
- високі вимоги до швидкості каналу (мережі) між сервером бази даних і серверами додатків [2].

Багатоланкова архітектура клієнт-сервер

Багатоланкова архітектура клієнт-сервер - це пряме продовження поділу додатків на рівні інтерфейсу користувача, логіки додатку та доступу до даних.

Можливі різні варіанти реалізації багатоланкової архітектури. Один з них: різні ланки взаємодіють відповідно до логічної організації додатку. Такий тип розподілу називається вертикальним. Головна його особливість - це розміщення логічно різних компонентів на різних машинах.

Інший вид розподілу - горизонтальний розподіл. При такому типі розподілу клієнт або сервер може містити фізично розділені частини логічно однорідного модуля, причому робота з кожною з частин може відбуватися незалежно. Це робиться для вирівнювання навантаження.

Є й інші варіанти організації архітектури, наприклад, розподіленої як вертикально, так і горизонтально [3].

Сервіс-орієнтована архітектура

Сервіс-орієнтована архітектура (SOA, service-oriented architecture) - модульний підхід до розробки програмного забезпечення, заснований на використанні сервісів (служб) із стандартизованими інтерфейсами (рис. 5).

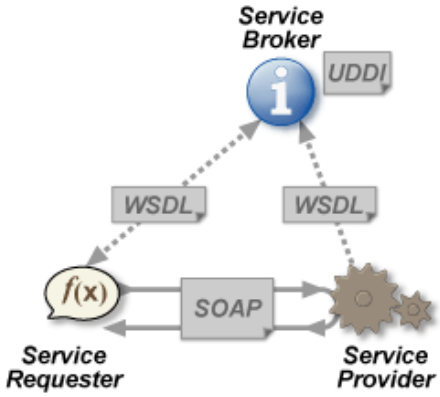

Рис. 5. Сервіс-орієнтована архітектура [4]

Сервіс (служба) - це компонента програми, безпосередньо доступна користувачу.

OASIS (Організація з розповсюдження відкритих стандартів структурованої інформації) визначає SOA наступним чином (OASIS Reference Model for Service Oriented Architecture V 1.0): сервіс-орієнтована архітектура - це парадигма організації та використання розподілених інформаційних ресурсів, таких як додатки і дані, що знаходяться в сфері відповідальності різних власників, для досягнення бажаних результатів споживачем, яким може бути кінцевий користувач або інша програма.

В основі SOA лежать принципи багатократного використання функціональних елементів, ліквідації дублювання функціональності в програмному забезпеченні, уніфікації типових операційних процесів, забезпечення переводу операційної моделі компанії на централізовані процеси і функціональну організацію на основі промислової платформи інтеграції.

Компоненти програми можуть бути розподілені по різних вузлах мережі, і пропонуються як незалежні, слабо пов'язані, замінні сервіси-

додатки. Програмні комплекси, розроблені відповідно до SOA, часто реалізуються як набір веб-сервісів, інтегрованих за допомогою відомих стандартних протоколів (SOAP, WSDL, і т. п.)

Інтерфейс компонентів SOA-програми представляє собою інкапсуляцію деталей реалізації конкретного компонента (операційної системи, платформи, мови програмування, вендора) від інших компонентів. Таким чином, SOA надає гнучкий і елегантний спосіб комбінування і багаторазового використання компонентів для побудови складних розподілених програмних комплексів.

SOA добре зарекомендувала себе для побудови великих корпоративних програмних додатків. Цілий ряд розробників та інтеграторів пропонують інструменти і рішення на основі SOA (наприклад, платформи IBM WebSphere, Oracle / BEA Aqualogic, Microsoft Windows Communication Foundation, SAP NetWeaver, ІВК Юпітер, TIBCO, Diasoft).

Основними цілями застосування SOA для великих інформаційних систем, рівня підприємства, і вище є:

- скорочення витрат при розробці додатків, за рахунок упорядкування процесу розробки;
- розширення повторного використання коду;
- незалежність від використовуваних платформ, інструментів, мов розробки;
- підвищення масштабованості створюваних систем;
- поліпшення керованості створюваних систем.

Принципи SOA:

- архітектура не прив'язана до якоїсь певної технології;
- незалежність організації системи від використовуваної обчислювальної платформи (платформ);
- незалежність організації системи від вживаних мов програмування;
- використання сервісів, незалежних від конкретних додатків, з однаковими інтерфейсами доступу до них;

- організація сервісів як слабкозв'язаних компонентів для побудови систем.

Те, що архітектура не прив'язана до якоїсь певної технології, означає можливість її реалізації з використанням широкого спектру технологій, включаючи такі технології як REST, RPC, DCOM, CORBA або веб-сервіси. SOA може бути реалізована, використовуючи один з цих протоколів, і, наприклад, може використовувати додатково механізм файлової системи для обміну даними.

Web-сервіси базуються на широко поширених і відкритих протоколах: HTTP, XML, UDDI, WSDL і SOAP. Саме ці стандарти реалізують основні вимоги SOA - по-перше, сервіс повинен піддаватися динамічному виявленню і виклику (UDDI, WSDL і SOAP), по-друге, повинен використовуватися незалежний від платформи інтерфейс (XML). Нарешті, HTTP забезпечує функціональну HTTP сумісність. Web-сервіси розглядаються як ефективний інструмент для інтеграції, в тому числі для взаємодії процесів, які виконуються в різних компаніях. Особливе місце серед різних специфікацій, призначених для опису систем і додатків на рівні бізнес-процесів, займає мову BPEL4WS.

Головне, що відрізняє SOA, це використання незалежних сервісів, з чітко визначеними інтерфейсами, які, для виконання своїх завдань, можуть бути викликані якимось стандартним способом. Це здійснюється за умови, що сервіси заздалегідь нічого не знають про програму, яка їх викличе, а додаток не знає , яким чином сервіси виконують своє завдання.

SOA також може розглядатися як стиль архітектури інформаційних систем, який дозволяє створювати додатки, побудовані шляхом комбінації слабкозв'язаних і взаємодіючих сервісів. Ці сервіси взаємодіють на основі будь-якого чітко визначеного платформо-незалежного та мовно-незалежного інтерфейсу (наприклад, WSDL). Визначення інтерфейсу приховує мовно-залежну реалізацію сервісу.

Таким чином, системи, засновані на SOA, можуть бути незалежні від технологій розробки і платформ (таких як Java, .NET і т. д.). Наприклад, сервіси, написані на C #, що працюють на платформах .Net і сервіси на Java, що працюють на платформах Java EE, можуть бути з однаковим успіхом викликані загальним складеним додатком. Програми, що працюють на одних платформах, можуть викликати сервіси, які працюють на інших платформах, що полегшує повторне використання компонентів.

SOA може підтримувати інтеграцію і консолідацію операцій в складі складних систем, однак SOA не визначає і не надає методологій або фреймворків для документування сервісів [2].

Децентралізована архітектура

Однорангова, децентралізована або пірингова (від англ. Peer-to-peer, P2P - рівний до рівного) мережа - це комп'ютерна мережа, заснована на рівноправності учасників. У такій мережі відсутні виділені сервери, а кожен вузол (peer) є як клієнтом, так і сервером. На відміну від архітектури клієнт-сервер, така організація дозволяє зберігати працездатність мережі при будь-якій кількості і будь-якому поєднанні доступних вузлів.

Кожен комп'ютер при децентралізованій архітектурі є незалежним від інших, містить тільки ту інформацію, з якою повинен працювати, а актуальність даних у всій системі забезпечується завдяки безперервному обміну повідомленнями з іншими комп'ютерами. Обмін повідомленнями між ними може бути реалізований різними способами, від відправки даних по електронній пошті до передачі даних по мережах.

Однією з переваг такої схеми експлуатації та архітектури системи, є забезпечення можливості персональної відповідальності за збереження даних. Так як дані, доступні на конкретному робочому місці, знаходяться тільки на цьому комп'ютері, при використанні засобів шифрування і особистих апаратних ключів виключається доступ до даних сторонніх осіб [3].

Література

1. Савельева, Е. А. Методические материалы к курсу «Технологии распределенных систем» / Е. А. Савельева. – Алматы: Научно-издательский центр КазНТУ, 2010. – 90 с.
2. Архитектурные особенности проектирования и разработки Веб-приложений [Электронный ресурс] - Режим доступа: http://www.intuit.ru/studies/courses/611/467/lecture/28784?page=1. Дата доступу: 15.05.17.
3. Поддержка разработки распределенных приложений в Microsoft .NET Framework [Электронный ресурс] – Режим доступа: http://www.intuit.ru/studies/courses/1115/177/info. Дата доступу: 17.04.17.
4. Веб-служба [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/Веб-служба. Дата доступу: 29.05.17.