ЛЕКЦІЯ 4
ВЗАЄМОДІЯ В РОЗПОДІЛЕНИХ СИСТЕМАХ. МЕРЕЖЕВА КОМУНІКАЦІЯ

Складності у розробці розподілених систем.

Реалізація властивостей розподільних систем, про які ми говорили на попередній лекції на практиці є доволі складною. Основні складності пов'язані саме з тим, що компоненти розподільної системи знаходяться географічно далеко один від одного та алгоритм виконується на незалежних комп'ютерах. Кожен, хто вперше створює розподілену програму робить наступні припущення:

1. Мережа є надійною.
2. Затримка передачі повідомлень дорівнює нулю.
3. Мережа безпечна.
4. Мережева топологія незмінна.
5. Систему обслуговує 1 адміністратор.
6. Мережа однорідна.

Всі перераховані припущення є звісно невірними. В області розподільних та паралельних обчислень розглядаються методи та алгоритми, що дозволяють реалізувати розподілені системи, за умови, що хоча б одне з цих припущень невірне.

Взаємодія компонент розподіленої системи.

Концепції взаємодії компонент розподіленої системи

Нині розрізняють дві концепції взаємодії програмних компонент: обмін повідомленнями між компонентами й виклик процедур або методів об'єкта віддаленої компоненти за аналогією до локального виклику процедури. Оскільки будь-яка взаємодія між віддаленими компонентами

ґрунтується на сокетах TCP/IP, базовим з погляду проміжного середовища є низькорівневий обмін повідомленнями на основі мережних сокетів, сервіс яких ніяк не визна- чає формату переданого повідомлення. На основі протоколів TCP або HTTP потім можна побудувати прикладні протоколи обміну повідомленнями більш високого рівня абстрагування, щоб реалізувати складніший обмін пові- домленнями або віддалений виклик процедур.

Віддалений виклик є моделлю, що походить від мов програмування висо- кого рівня, а не від реалізації інтерфейсу транспортного рівня мережних прото- колів, тому протоколи віддаленого виклику мають обов'язково ґрунтуватися на будь-якій системі передачі повідомлень, включаючи як безпосереднє використан- тання сокетів TCP/IP, так інші проміжні середовища для обміну повідомлення- ми. Під час реалізації високорівневих служб обміну повідомленнями, у свою чергу, можуть використовуватися віддалений виклик процедур, в основі якого лежить більш низькорівнева передача повідомлень, що використовує, на- приклад, безпосередньо мережні сокети. Таким чином, одне проміжне середо- вище може використовувати для свого функціонування сервіси іншого, анало- гічно до того, як один протокол транспортного або мережного рівня може працювати поверх іншого у разі тунелювання протоколів.

Обмін повідомленнями

Розрізнюють два методи передачі повідомлень від однієї віддаленої системи до другої: безпосередній обмін повідомленнями й використання черг повідомлень. У разі безпосереднього обміну передача відбувається прямо, і можлива лише за умови, що сторона, яка приймає, готова одержати повідомлення в цей самий момент, інакше використовується посередник – менеджер черг повідомлень, тобто компонента надсилає повідомлення в

одну із черг менеджера, після чого вона може продовжити свою роботу. Надалі сторона, яка отримує повідомлення, вилучить його із черги менеджера й розпочне обробку.

Найпростішим видом реалізації безпосереднього обміну повідомленнями є використання транспортного рівня мережі через інтерфейс сокетів, минаючи будь-яке проміжне програмне забезпечення. Однак такий спосіб взаємодії зазвичай не застосовують у розподілених системах, оскільки в цьому разі реалізують усі функції проміжного середовища розробники прикладних про- грам. За такого підходу складно отримати розширювану й надійну розподілену систему, тому для розробки прикладних розподілених систем здебільшого використовують системи черг повідомлень.

Наявні кілька розробок у сфері проміжного програмного забезпечення, що реалізують високорівневі сервіси для обміну повідомленнями між програмними компонентами, зокрема Microsoft Message Queuing, IBM MQSeries і Sun Java System Message Queue, ці системи дають можливість прикладним програмам використовувати такі базові примітиви обробки черг:

- додати повідомлення в чергу;
- взяти перше повідомлення із черги, процес блокується до появи в черзі хоча б одного повідомлення;
- перевірити чергу на наявність повідомлень;
- установити обробник, який викликається з появою повідомлень у черзі.

Менеджер черги повідомлень у таких системах може перебувати на комп’ютері, розміщеному окремо від комп'ютерів з компонентами, що беруть участь в обміні. У цьому разі повідомлення спочатку поміщується у вихідну чергу на комп’ютері з компонентою, що надсилає повідомлення, а потім пересилається менеджерові необхідної компоненти. Для створення

великих систем обміну повідомленнями може використовуватися маршрутизація повідомлень, за якої повідомлення не передаються прямо менеджерові, що підтримує чергу, а проходять через низку проміжних менеджерів черг повідомлень (рис. 4.1).

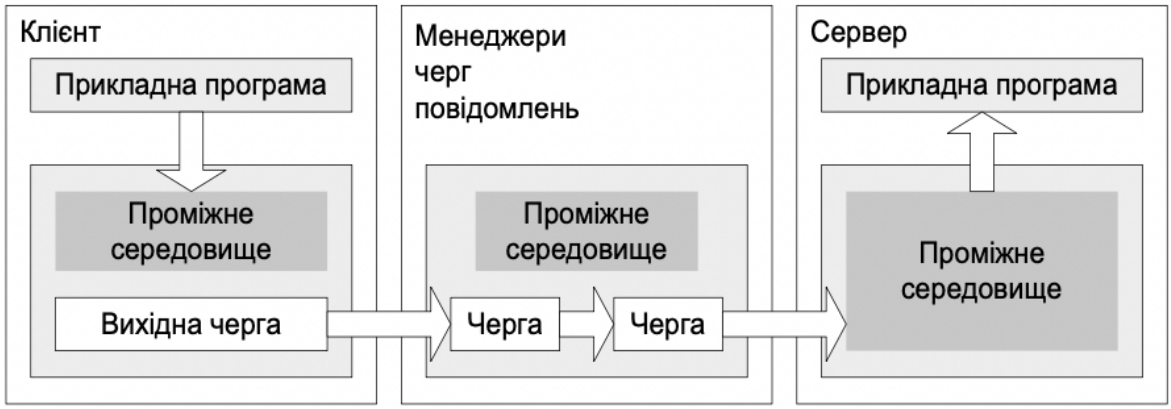

Рис. 4.1. Система черг повідомлень

Використання черг повідомлень орієнтовано на асинхронний обмін даними. Основні переваги таких систем:

- час функціонування сервера може не залежати від часу роботи клієнтів;
- незалежність проміжного середовища від засобу розробки компонент і мови програмування;
- зчитувати й обробляти заявки із черги можуть кілька незалежних компонент, що дає можливість досить просто створювати стійкі й масштабовані системи.

Недоліки систем черг повідомлень такі:

- необхідність явного використання черг розподіленою прикладною програмою;
- складність реалізації синхронного обміну;
- певні витрати на використання менеджерів черг;
- складність отримання відповіді, оскільки передача відповіді може ви-

магати окремої черги на кожний компонент, що надсилає заявки.

Віддалений виклик процедур

Ідея віддаленого виклику процедур (Remote Procedure Call, RPC) з'явилася в середині 80-х років і полягала в тому, що за допомогою проміжного програмного забезпечення функцію на віддаленому комп'ютері можна викликати так само, як і функцію на локальному комп'ютері.

Щоб віддалений виклик відбувався прозоро з погляду прикладної програми, яка виконує виклик, проміжне середовище має надати процедуру-заглушку (stub), що буде викликатися клієнтською прикладною програмою. Після виклику процедури - заглушки проміжне середовище надає переданим їй аргументам виду, придатного для передачі за транспортним протоколом, і спрямовує їх на віддалений комп'ютер з викликуваною функцією. На віддаленому комп'ютері параметри вилучаються проміжним середовищем з повідомлення транспортного рівня й передаються викликуваній функції (рис. 4.2).

Аналогічно на клієнтську машину надсилається результат виконання функції, викликаної з сервера.

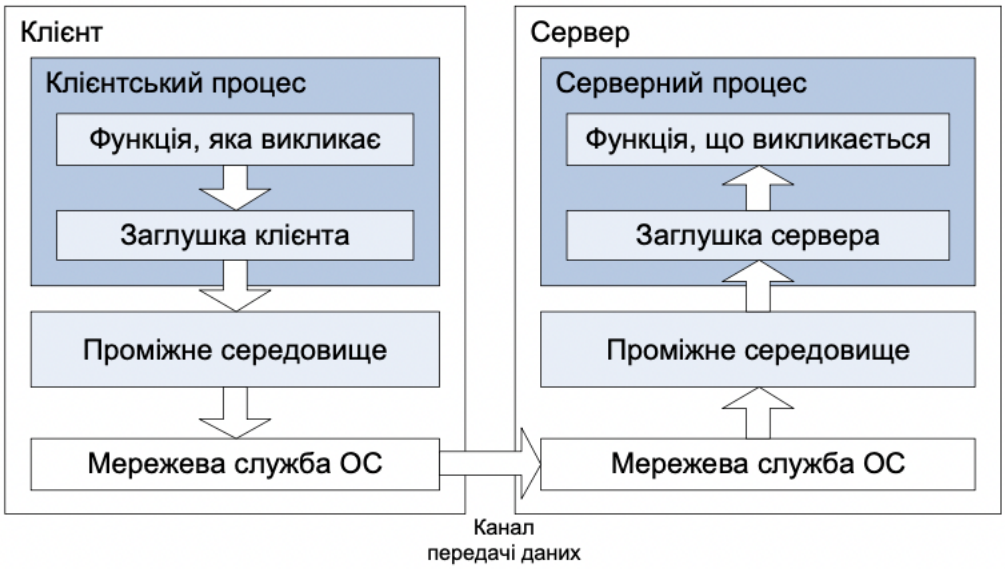

Рис. 4.2. Віддалений виклик процедур

Розрізняють три можливі варіанти віддаленого виклику процедур:

Синхронний виклик – клієнт очікує завершення процедури сервером і, у разі потреби, отримує від нього результат виконання віддаленої функції.

Однонаправлений асинхронний виклик – клієнт продовжує виконувати своє завдання, не отримуючи відповіді від сервера, відповідь або відсут- ня, або її реалізація якось інакше передбачена під час розробки (наприклад, через функцію клієнта, яку віддалено викликає сервер).

Асинхронний виклик – клієнт продовжує виконувати завдання, після завершення сервером процедури він отримує повідомлення й результат її вико- нання, наприклад через callback-функцію, яка викликається проміжним сере- довищем під час одержання результату від сервера.

Процес перетворення параметрів для їх передачі між процесами (або доменами прикладної програми у разі використання .NET) під час віддаленого виклику називають маршалізацією (marshalling). Перетворення екземпляра якого-небудь типу даних у придатний для передачі за межі викликаючого процесу набір байтів називаються серіалізацією.

Десеріалізація – процедура, обернена серіалізації, – полягає у створенні копії серіалізованого об’єкта на основі отриманого набору байтів. Такий під- хід до передачі об’єкта між процесами за допомогою створення його копій називають маршалізацією за значенням (marshal by value), на відміну від маршалізації за посиланням.

Маршалізація за посиланням під час передачі параметрів за посиланням використовує серіалізацію не самих вказівників, а об’єктів, на які вказують вказівники.

Процес серіалізації повинен бути визначений для всіх типів даних, переданих у процесі віддаленого виклику, зокрема для параметрів функції, яка викликається й результату, який повертає функція. У разі передачі параметрів за посиланням сериалізації підлягають об’єкти, на які посилаються, до самих вказівників сериалізація не може бути застосована, оскільки це ускладнює використання механізму віддаленого виклику в мовах, які підтримують вказівники на об’єкти невідомого типу.

Взаємодія у розподілених системах.

Фізичний час.

В розподілених системах обчислення реалізуються сукупністю незалежних процесів, що взаємодіють, надсилаючи повідомлення для обміну даними та координацією своїх дій. Час, витрачений кожним процесом на виконання окремих дій, різний та невідомий заздалегідь, а доставка окремих повідомлень також займає непередбачуваний час.

Тому точність, з якою розподілені процеси можуть координувати свої дії обмежена затримками часу та складністю представлення єдиного часу серед всіх компонентів системи.

Кожен з комп’ютерів у розподіленій системі має свій власний годинник, точніше системний таймер, що використовується його локальними процесами для отримання поточного часу.

Тому два процеси, що виконуються на різних комп’ютерах можуть асоціювати відмітку часу з кожною подією.

Однак навіть якщо всі процеси будуть зчитувати показники годинників в один і той самий час, повертаємі значення можуть різнитися.

Справа в тому, що ніякий годинник не є ідеальним: кварцеві генератори, що є основою для сучасних годинників у комп’ютерів, не можуть мати абсолютно однакову частоту, що призводить до поступового зниження синхронізації та повертання таймерами різних значень.

Ця різниця у показниках годинників називається розсинхронізацією годинників, а швидкість відхилення годиннику від показників універсального скоординованого часу UTC з плином часу – швидкість дрейфу (рис.4.3).

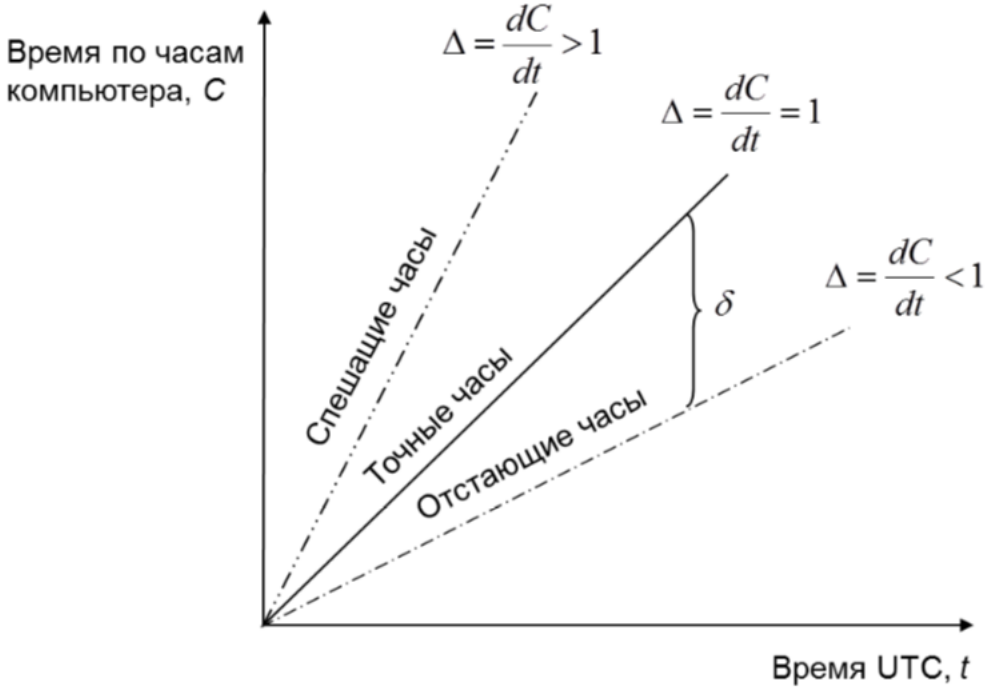

Рис. 4.3. Розсинхронізація годинників

∂ - розсинхронізація, Δ – швидкість дрейфу.

Таким чином, навіть якщо в один момент на годинниках всіх комп'ютерів, що входять до розподіленої системи встановлений однаковий час, з плином часу їх показники можуть значно відрізнятися.

Наведемо приклад. Банківська система.

Розберемо задачу підрахунку повної суми грошей, що знаходяться на рахунках у різній частині філіалу банка. Припустимо, що банківська система не дозволяє вносити додаткові кошти на рахунок філіалу та знімати їх, а лише здійснює переводи з одного філіалу в інший за допомогою надсилання повідомлень. Для ілюстрації цієї системи можна намалювати графік процесів та подій (Рис.4.4.):

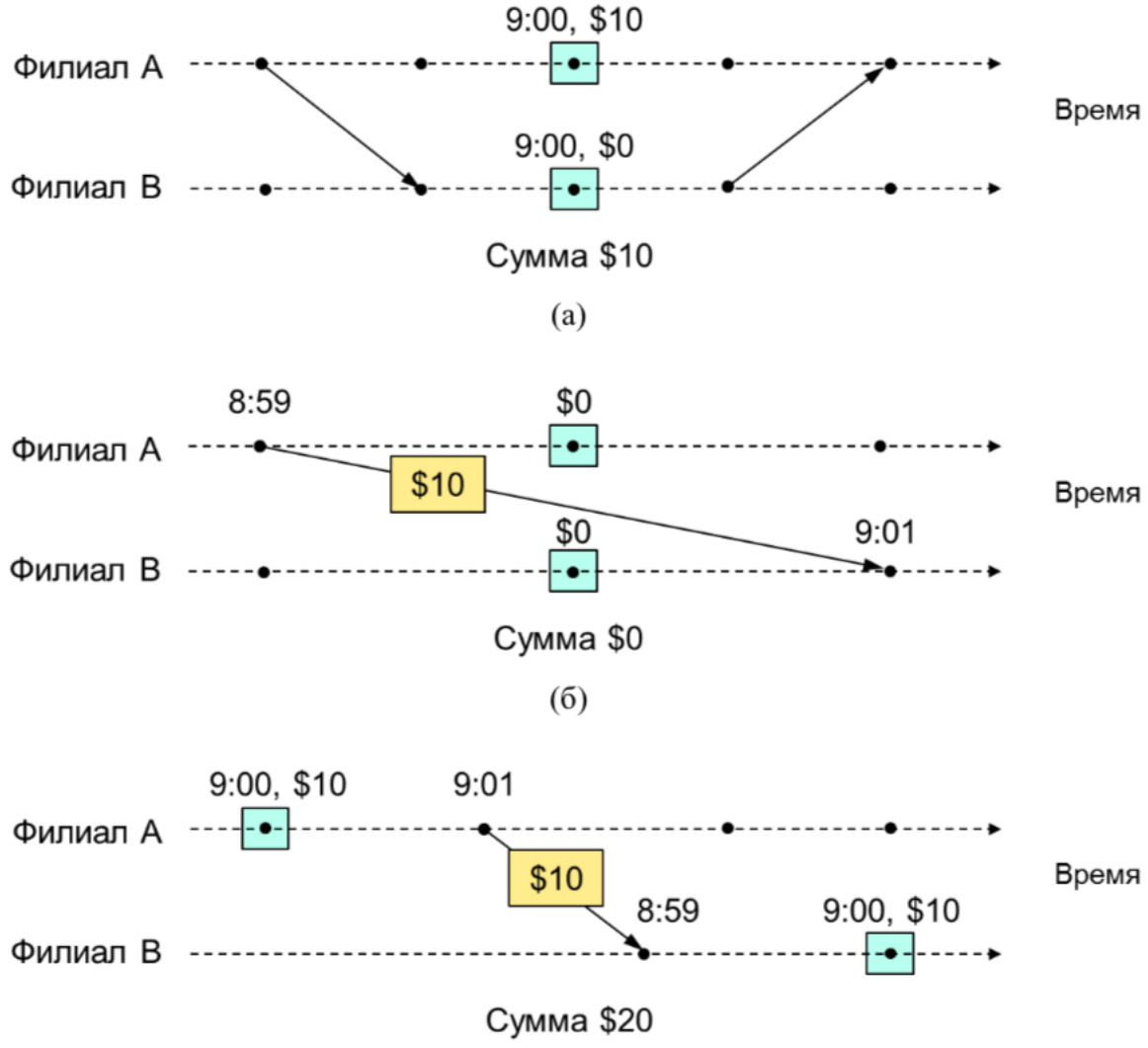

Рис. 4.4. Графік подій в банківскій системі (фргмент прикладу)

Горизонтальними лініями на графіку зображені часові осі, точки на яких відповідають певній події. Квадрат навколо точки означає стан локального процесу у фіксований момент часу, що відповідає даній точці. Стрілкою позначається передача повідомлень від одного процесу до іншого. Щоб визначити повну суму банк повинен мати інформацію про кількість грошей на кожному з філіалів.

Припустимо, що нам треба порахувати кошти о 9:00 – час, що відомий всім процесам. У випадку, що показаний на малюнку а сума буде $10, з ситуації на малюнку б (кошти знаходяться у стані передачі) сума буде $0. Цю проблему можна вирішити, вивчаючи всі події, що знаходяться у стані передачі у момент підрахунку коштів.

Наприклад, хай всі філіали зобов’язані зберігати дані про всі переводи та надходження коштів.

Перевіряючи ці дані система зможе детермінувати переводи, що вже були відправлені, але не були одержані та додавати їх до суми всіх коштів.

Але якщо годинники на комп’ютерах двох філіалів розсинхронізовані, може статися таке, що гроші будуть підраховані двічі.

С*инхронні та асинхронні розподілені системи*.

В розподілених мережах практично неможливо передбачити час виконання окремих дій різноманітних процесів, швидкість відхилення локального годинника від точного або затримку доставки повідомлення від одного процесу до іншого.

Так, виділяють дві діаметрально різні моделі обчислювальних систем, перша накладає строгі часові обмеження на відповідні характеристики, в іншій не робиться ніяких припущень відносно часу.

Синхронні розподільні системи.

Синхронні розподільні системи – це системи, де визначені наступні обмеження:

- Час виконання кожної окремої дії обмежено зверху та знизу деякими значеннями.
- Затримка доставки повідомлень не перевищує відомої границі.
- Кожен процес має свій локальний годинник зі швидкість відхилення від точного, що не перевищує відомого значення.

Важливо відмітити, що для більшості випадків, скоріш за все, можна дати деякі оцінки перерахованим часовим характеристикам. Однак, якщо на практиці все ж неможливо забезпечити їх виконання, наприклад, забезпечити гарантовану доставку усіх повідомлень в межах заданого часу, то будь-які системи та алгоритми, побудовані на припускаємих значеннях наведених вище границь, не будуть працювати у випадку їх порушення.

Через скоординовану роботу всіх процесів синхронної системи, розробка та відладка алгоритмів для них набагато простіша. Наприклад, у синхронних системах можна використовувати тайм аути для визначення відмови одного з процесів.