3.8 СИМЕТРИЧНА БАГАТОПРОЦЕСОРНА ОБРОБКА

До недавнього часу усі персональні комп'ютери, розраховані на одного користувача, і робочі станції містили один процесор загального призначення. В результаті постійного підвищення вимог до продуктивності і пониження вартості процесорів виробники перейшли до випуску комп'ютерів з декількома процесорами. Для підвищення ефективності і надійності використовується технологія симетричної багатопроцесорності (Symmetric MultiProcessing – SMP). Цей термін належить до архітектури апаратного забезпечення комп'ютера, а також до образу дій ОС, що відповідає цій архітектурній особливості. Симетричну багатопроцесорність можна визначити, як автономну комп'ютерну систему з такими характеристиками:

1. У системі є декілька процесорів.
2. Ці процесори, сполучені між собою комунікаційною шиною, спільно використовують одну і ту ж основну пам'ять і одні і ті ж пристрої введення-виведення.
3. Усі процесори можуть виконувати одні і ті ж функції (звідси назва симетрична обробка).

ОС, працююча в системі з симетричною багатопроцесорністю, розподіляє процеси або потоки між усіма процесорами. У багатопроцесорних систем є декілька потенційних переваг в порівнянні з однопроцесорними.

Продуктивність. Якщо завдання можна організувати так, що його певні частини виконуватимуться паралельно, це призведе до підвищення продуктивності порівняно з однопроцесорними системами.

Надійність. При симетричній мультипроцесорній обробці відмова одного процесора не призведе до зупинки машини.

Нарощування. Додаючи в систему додаткові процесори, можна підвищити продуктивність системи.

Масштабованість. Виробники можуть пропонувати свої продукти в різних конфігураціях, що відрізняються ціною і продуктивністю.

3.8.1 Архітектура симетричної багатопроцесорності

Найранішою і найвідомішою є класифікація архітектури обчислювальних систем, запропонована в 1966 році М. Флінном. Класифікація базується на понятті потоку, під яким розуміється послідовність елементів, команд або даних, що обробляється процесором. На основі числа потоків команд і потоків даних Флінн виділяє чотири класи архітектури: SISD, MISD, SIMD, MIMD (ми розглянемо два). Ці чотири класи архітектури схематично представляються у вигляді квадрата, що називається квадратом Флінна (рис. 3.17).

		Data Stream	
		Single	Multiple
Instruction Stream	Single	**SISD**	**SIMD**
	Multiple	**MISD**	**MIMD**

Рисунок 3.17 – Класифікація Флінна

На рисунках, що ілюструють класифікацію М. Флінна, використані такі позначення:

- ПР – один або декілька процесорних елементів;
- ПУ – пристрій управління;
- ПД – пам'ять даних.

SISD (Single Instruction stream / Single Data stream) – одиночний потік команд і одиночний потік даних. До цього класу належать класичні послідовні машини, або інакше, машини фон-нейманівського типу, наприклад, PDP-11, IBM PC (рис. 3.18). У таких машинах один управляючий пристрій, він отправляє тільки один потік команд, усі команди обробляються послідовно одна за одною одним процесором, і кожна команда ініціює одну операцію з одним потоком даних. Не має значення той факт, що для збільшення швидкості обробки команд і швидкості виконання арифметичних

операцій може застосовуватися конвеєрна обробка – як машина CDC 6600 із скалярними функціональними пристроями, так і CDC 7600 з конвеєрними пристроями потрапляють в цей клас.

SIMD (Single Instruction stream / Multiple Data stream) – поодинокий потік команд і множинний потік даних. Ці системи мають один управляючий пристрій і велику кількість процесорів, які можуть виконувати одну і ту ж інструкцію щодо різних даних. Єдина інструкція паралельно виконується над багатьма елементами даних. Архітектура подібного роду включає, на відміну від попереднього класу, векторні команди. Це дозволяє виконувати одну арифметичну операцію відразу над багатьма даними – елементами вектору. Спосіб виконання векторних операцій не обмовляється, тому обробка елементів вектора може здійснюватися або процесорною матрицею, як в ILLIAC IV, або за допомогою конвеєра, як, наприклад, в машині CRAY-1 (див. рис. 3.18).

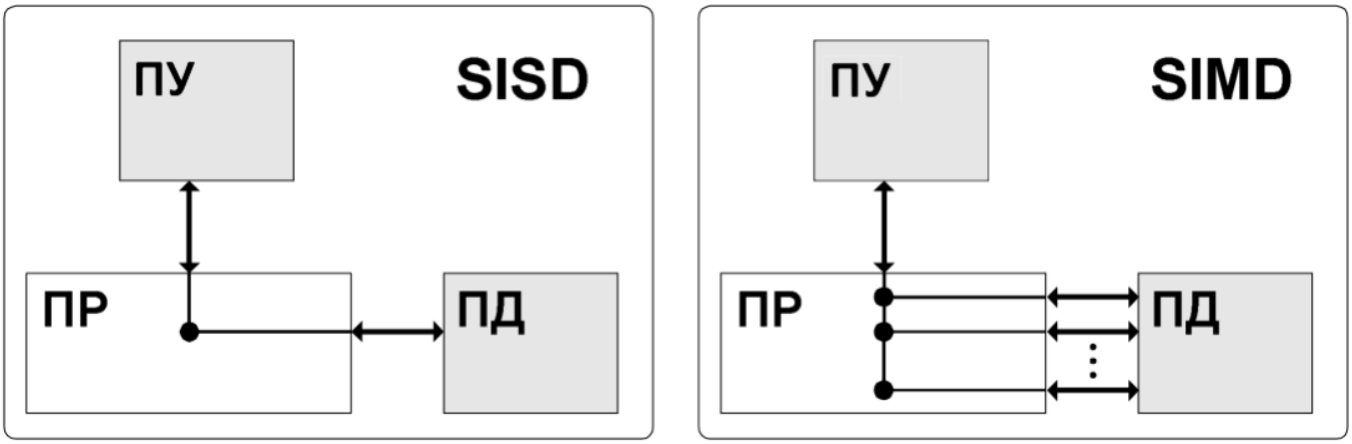

Рисунок 3.18 – Архітектура SISD і SIMD

Машини типу SIMD складаються з великого числа ідентичних процесорних елементів, що мають власну пам'ять. Усі процесорні елементи в такій машині виконують одну і ту ж програму. Очевидно, що така машина, складена з великого числа процесорів, може забезпечити дуже високу продуктивність тільки на тих задачах, при розв'язанні яких усі процесори можуть робити одну і ту ж роботу. Модель обчислень для машини SIMD дуже схожа на модель обчислень для векторного процесора: поодинока операція виконується над великим блоком даних. На відміну від обмеженого конвеєрного функціонування векторного процесора,

матричний процесор (синонім для більшості SIMD-машин) може бути значно гнучкішим. Оброблювальні елементи таких процесорів – це універсальні програмовані ЕОМ, так що задачі, що розв'язуються паралельно, можуть бути досить складними і містити галуження.

Моделі обчислень на векторних і матричних ЕОМ настільки схожі, що ці ЕОМ часто вважаються як еквівалентними.

Майже усі комп'ютери сьогодні реалізують певну форму набору команд SIMD. Процесори Intel реалізують набори команд MMX, Streaming SIMD Extensions (SSE), Streaming SIMD Extensions 2 (SSE2) і Streaming SIMD Extensions 3 (SSE3), які можуть обробляти декілька елементів даних за один такт. Безперечними представниками класу SIMD вважаються матриці процесорів: ILLIAC IV, ICL DAP, Goodyear Aerospace MPP тощо. У таких системах єдиний управляючий пристрій контролює певну кількість процесорних елементів. Кожен процесорний елемент отримує від пристроїв управління в кожен фіксований момент часу однакову команду і виконує її над своїми локальними даними. Для класичних процесорних матриць ніяких питань не виникає, проте в цей же клас можна включити і векторно-конвеєрні машини, наприклад, CRAY-1.

MISD (Multiple Iinstruction stream / Single Data stream) – множинний потік команд і поодинокий потік даних. Це архітектура багатьох процесорів, які оброблюють один і той же потік даних (рис. 3.19). Проте ні Флінн, ні інші фахівці в області архітектури комп'ютерів досі не змогли представити приклад реально існуючої обчислювальної системи, побудованої на цьому принципі. В більшості випадків декільком потокам команд потрібні декілька потоків даних, так що цей клас паралельних комп'ютерів застосовується дослідниками лише як теоретична модель, а не як реальний комп'ютер масового виробництва. Вважається, що дотепер цей клас порожній. Але не варто вважати це недоліком схеми. Така архітектура в майбутньому може стати надзвичайно корисною для розробки принципово нових обчислювальних систем.

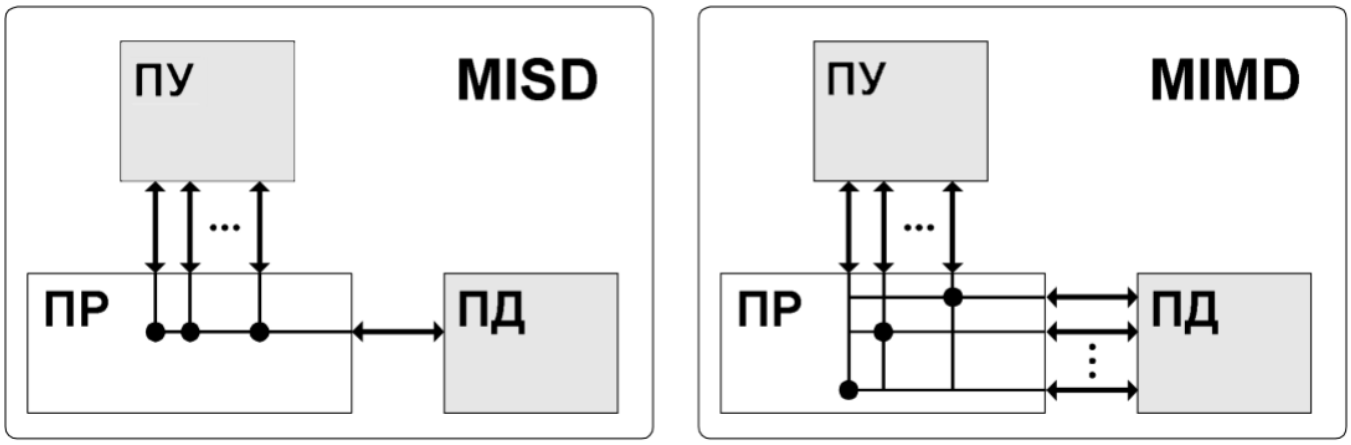

Рисунок 3.19 – Архітектура MISD і MIMD

MIMD (multiple instruction stream / multiple data stream) – множинний потік команд і множинний потік даних. У цій архітектурі кожен процесор має свій управляючий пристрій і незалежно виконує різні набори команд, що обробляють різні набори даних (див. рис. 3.19). Системи в архітектурі MIMD діляться на системи з розподіленою пам'яттю (слабкозв'язані системи), до яких належать кластери, і системи з спільно використовуваною пам'яттю. До останніх належать симетричні мультипроцесорні системи. В наші дні це – найпопулярніший тип паралельного комп'ютера. Нові багатоядерні платформи (такі, як процесор Intel Core Duo) потрапляють саме в цю категорію.

Термін «мультипроцесор» покриває більшість машин типу MIMD і (подібно до того, як термін «матричний процесор» застосовується до машин типу SIMD) часто використовується як синонім для машин типу MIMD. У мультипроцесорній системі кожен процесорний елемент виконує свою програму досить незалежно від інших процесорних елементів. Процесорні елементи, звичайно, повинні якось зв'язуватися один з одним, що робить необхідним детальнішу класифікацію машин типу MIMD.

У мультипроцесорах із загальною пам'яттю (сильнозв'язанних мультипроцесорах) є пам'ять даних і команд, доступна усім процесорним елементам. Із загальною пам'яттю процесорні елементи зв'язуються за допомогою загальної шини або мережі обміну. В протилежність цьому варіанту в слабкозв'язаних багатопроцесорних системах (машинах з локальною пам'яттю) уся пам'ять ділиться між процесорними елементами і кожен блок пам'яті доступний тільки пов'язаному з ним процесору. Мережа обміну зв'язує процесорні елементи один з одним.

Базовою моделлю обчислень на MIMD-мультипроцесорі є сукупність незалежних процесів, які епізодично звертаються до даних, що розділяються. Існує велика кількість варіантів цієї моделі. На одному кінці спектру – модель розподілених обчислень, в якій програма ділиться на досить велике число паралельних завдань, що складаються з декількох підпрограм. На іншому кінці спектру – модель потокових обчислень, в яких кожна операція в програмі може розглядатися як окремий процес. Така операція чекає своїх вхідних даних (операндів), які мають бути передані їй іншими процесами. Після їх отримання операція виконується, і отримане значення передається тим процесам, які його потребують.

Клас MIMD надзвичайно широкий, оскільки включає всілякі мультипроцесорні системи: CRAY Y-MP, Intel Paragon, CRAY T3D і багато інших. Цікаве те, що якщо конвеєрну обробку розглядати як виконання декількох команд не над поодиноким векторним потоком даних, а над множинним скалярним потоком, то усі розглянуті вище векторно-конвеєрні комп'ютери можна розташувати і в цьому класі. Подальша класифікація обчислювальних систем з архітектурою MIMD може виконуватися відповідно до того, як в них здійснюється обмін даними між процесорами.

У системі MIMD процесори є універсальними, тому що вони повинні мати можливість обробляти усі команди, необхідні для відповідного перетворення даних. Якщо кожному процесору виділяється окрема ділянка пам'яті, то кожен такий елемент є самостійним комп'ютером. Вони обмінюються між собою інформацією або через спеціальні канали, або через деякі мережеві пристрої. Такі системи відомі як кластери, або мультикомп'ютери. Якщо процесори спільно використовують загальну пам'ять, то кожен з них має доступ до програм і даних, які там зберігаються. Такі системи відомі під назвою багатопроцесорних систем із загальною пам'яттю.

Одна з класифікацій багатопроцесорних систем грунтується на тому, як процеси розподіляються між процесорами. Існують два головні підходи – виділення основних і підпорядкованих процесорів і симетрична багатопроцесорна обробка. В архітектурі з ведучим і веденими процесорами ядро ОС завжди виконується на спеціально виділеному процесорі. На інших процесорах можуть виконуватися тільки програми користувача і, можливо, утиліти ОС.

Провідний процесор відповідає за планування процесів або потоків. Якщо процесу, що виконується на веденому процесорі, знадобиться який-небудь системний сервіс, він повинен послати запит основному процесору і потім чекати, поки сервісна програма не закінчить свою роботу. Для реалізації такого підходу досить удосконалити ОС, призначену для однопроцесорних багатозадачних систем. Вирішення конфліктів спрощується, завдяки тому, що усією пам'яттю і усіма ресурсами введення-виведення управляє один процесор. Цей підхід має ряд недоліків:

1. Збій в роботі основного процесора призводить до відмови усієї системи.
2. Основний процесор може гальмувати роботу усієї системи, оскільки тільки на ньому повинні виконуватися усі дії з планування і управління процесами.

У симетричній багатопроцесорній системі ядро ОС може виконуватися на будь-якому процесорі. Як правило, кожен процесор сам планує свою роботу. Ядро може бути виконане у вигляді багатьох процесів або багатьох потоків, при цьому різні його частини здатні працювати паралельно. Симетричний підхід дещо ускладнює архітектуру ОС. Потрібно вжити запобіжні заходи, щоб два процесори не вибрали один і той же процес, або щоб процес яким-небудь чином не випав з черги. Необхідно застосувати спеціальні методи для дозволу запитів одного і того ж ресурсу різними процесами і синхронізації запитів.

3.8.2 Організація симетричної багатопроцесорної системи

Архітектура SMP-системи має декілька процесорів, кожен з яких містить свій власний управляючий модуль, арифметично-логічний пристрій і свої регістри. Кожен з процесорів має доступ до загальної основної пам'яті і до пристроїв уведення-виведення. Цей доступ здійснюється за допомогою деякого механізму взаємодії. Традиційно в такій ролі виступає загальна шина. Процесори можуть обмінюватися між собою інформацією через загальну пам'ять. Крім того, процесори можуть мати можливість безпосереднього обміну сигналами.

Як правило, в сучасних машинах процесори мають, принаймні, один рівень власного кеша. Це вносить деякі нюанси в архітектуру ОС. Оскільки в кожному локальному кеші зберігається образ якоїсь частини основної пам'яті, то в результаті зміни слова в одному кеші відповідне слово в іншому кеші може виявитися невірним. Для запобігання цьому усі процесори, кеш яких містить це слово, мають бути

сповіщені про необхідність змінити його. Ця проблема відома як когерентності кешів (від лат. cohaerens – що знаходиться в зв'язку) і вирішується на апаратному рівні.

3.8.3 Архітектура багатопроцесорних ОС

ОС, призначена для симетричної багатопроцесорної системи, управляє процесорами і іншими ресурсами комп'ютера так, щоб з точки зору користувача багатопроцесорна система виглядала так само, як і багатозадачна однопроцесорна. До числа особливостей архітектури багатопроцесорних ОС входять такі.

Одночасні паралельні процеси або потоки. Щоб декілька процесів могли одночасно виконувати один і той же код ядра, він має бути реєнтерабельним. При виконанні декількома процесорами одного і того ж коду ядра (чи різних його частин) потрібна організація управління таблицями і структурами ядра, щоб уникнути взаємоблокувань або неправильного виконання операції.

Планування. Планування може виконуватися на будь-якому з процесорів, тому необхідно передбачити механізм, що дозволяє уникнути конфліктів. При використанні багатопоточності на рівні ядра декілька потоків одного і того ж процесу можуть виконуватися на різних процесорах. Планування в багатопроцесорних системах розглядатиметься далі.

Синхронізація, синхронна і асинхронна взаємодія. За ситуації, коли декілька активних процесів мають можливість доступу до спільних адресних просторів або ресурсів введення-виведення, необхідного потурбуватись про їх ефективну синхронізацію.

Синхронізація – це засіб, який забезпечує реалізацію взаємовиключень і впорядкування подій. Загальноприйнятим механізмом синхронізації в багатопроцесорних ОС є блокування.

При описі взаємодії між елементами програмних систем ініціатор взаємодії, тобто компонент, що посилає запит на обробку, називається клієнтом, а компонент, що обробляє запит – сервером. У більшості випадків один і той же компонент може виступати в різних ролях – то клієнта, то сервера – в різних взаємодіях. Лише в невеликому класі систем ролі клієнта і сервера закріплюються за компонентами на увесь час їх існування.

Синхронною (блокуючою) називається така взаємодія між компонентами, при якій клієнт, відіславши запит, блокується і може продовжувати роботу тільки після отримання відповіді від сервера (рис. 3.20).

Синхронну взаємодію досить просто організувати, і вона набагато простіша для розуміння. Людині простіше розуміти процеси, які розгортаються послідовно, оскільки не треба постійно перемикати увагу на різні події, що відбуваються одночасно. Код програми клієнтського компонента, що описує синхронну взаємодію, влаштований простіше.

В той же час синхронна взаємодія веде до значних витрат часу на очікування відповіді. Цей час часто можна використати кориснішим чином – чекаючи відповіді на один запит, клієнт міг би зайнятися іншою роботою.

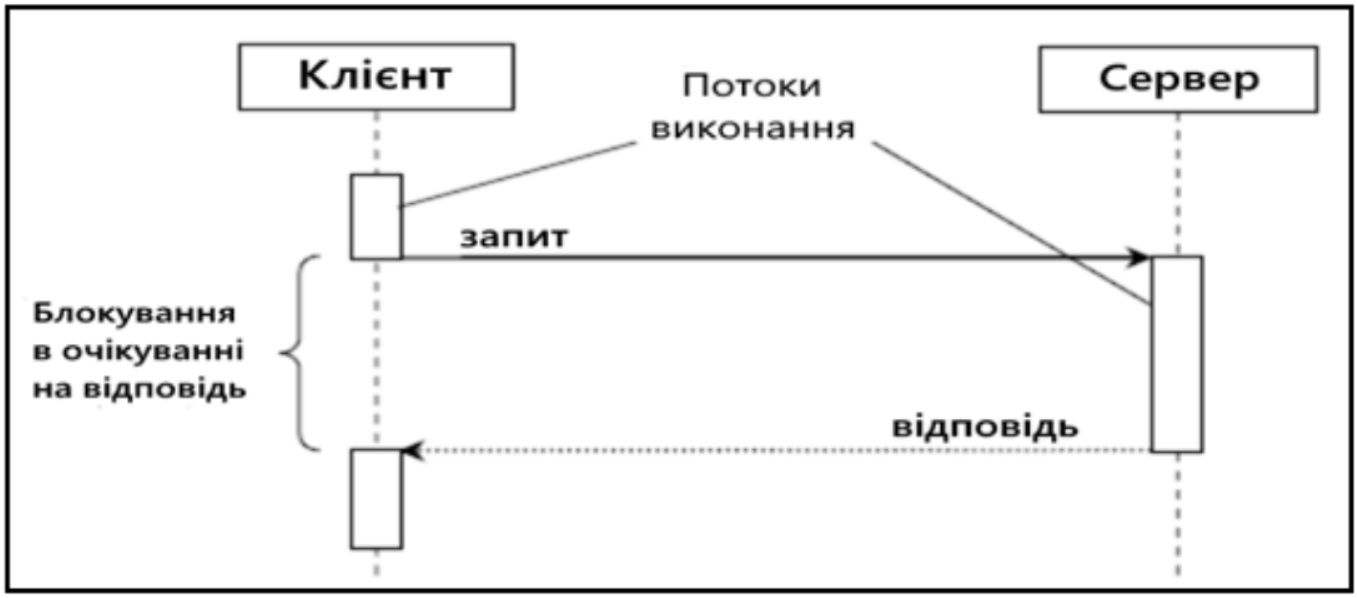

Рисунок 3.20 – Синхронна взаємодія

У рамках асинхронної або неблокуючої взаємодії клієнт після відправки запиту серверу може продовжувати роботу, навіть якщо відповідь на запит ще не прийшла. Асинхронна взаємодія дозволяє отримати вищу продуктивність системи за рахунок використання часу між відправкою запиту і отриманням відповіді на нього для виконання інших задач (рис. 3.21). Інша важлива перевага асинхронної взаємодії – менша залежність клієнта від сервера, можливість продовжувати роботу, навіть якщо машина, на якій знаходиться сервер, стала недоступною.

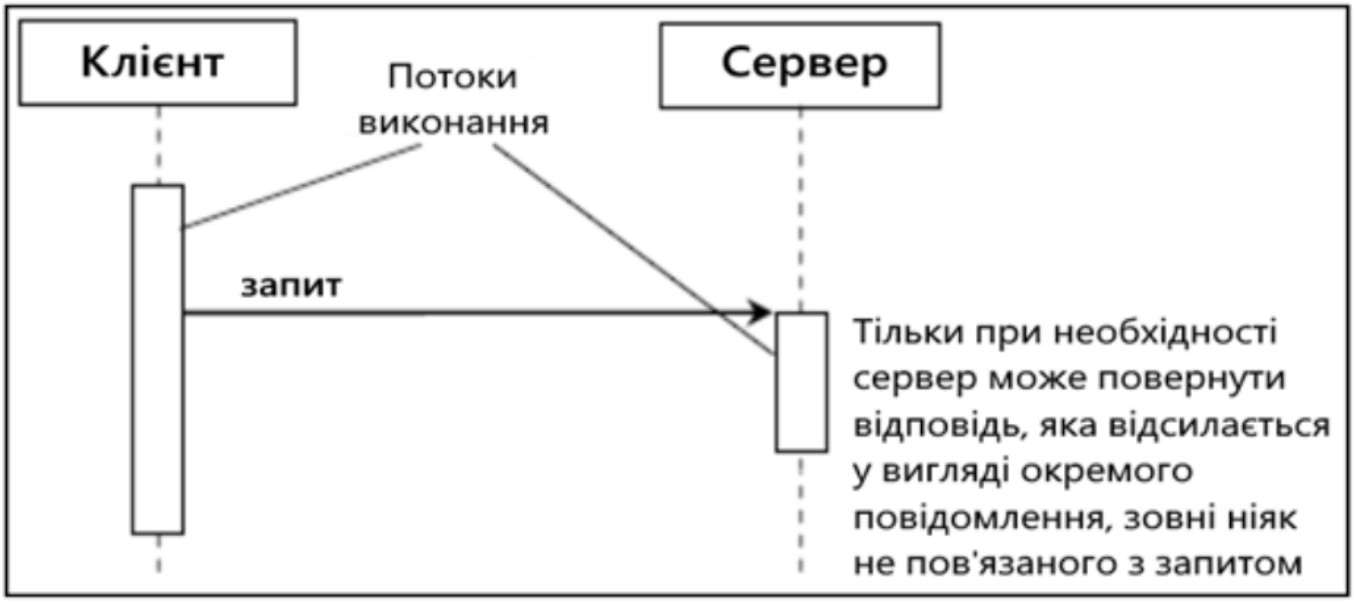

Рисунок 3.21 – Асинхронна взаємодія

В той же час асинхронну взаємодію набагато складніше організувати, розробляти і супроводжувати. Оскільки при такій взаємодії треба писати специфічний код для отримання і обробки результатів запитів.

Управління пам'яттю. Система управління пам'яттю в багатопроцесорній системі має бути здатна вирішувати усі проблеми, які виникають в однопроцесорних комп'ютерах. Крім того, ОС повинна уміти використати можливості, що надаються апаратним забезпеченням. Механізми сторінкової організації пам'яті різних процесорів мають бути скоординовани, щоб забезпечити узгодженість дій за ситуації, коли декілька процесорів використовують одну і ту ж сторінку або один і той же сегмент.

Надійність і відмовостійкість. При відмові одного з процесорів ОС повинна забезпечити продовження роботи системи. Планувальник ОС (як і інші його частини) повинен отримати інформацію про втрату одного з процесорів і відповідним чином перебудувати свої управляючі таблиці.