Міністерство освіти і науки України
Криворізький національний університет
Кафедра комп’ютерних систем та мереж

Методичні вказівки

до виконання лабораторних робіт

з дисципліни:

"Комп'ютерні системи"

для студентів спеціальності 123 - “Комп'ютерна інженерія” всіх форм навчання

Кривий Ріг
2019

Укладачі: зав. кафедрою КСМ, д-р техн. наук, проф. Купін А.І.,
доц., к.т.н., Кузнєцов Д.І.,
.асистенти каф. КСМ Сенько А.О., Рябчина Л.С,

Відповідальний
за випуск: зав. кафедрою КСМ, д-р техн. наук, доц. Купін А.І.

Рецензент: канд. техн. наук, доц. А.А. Жосан

Методичні вказівки містять лабораторні роботи з дисципліни “Комп'ютерні системи”. Для кожної роботи представлено необхідні теоретичні відомості та приклади, які допоможуть студентам краще засвоїти лекційний матеріал. Завдання лабораторних робіт дозволять отримати навички вирішення практичних задач, які виникають при побудові комп’ютерних систем.

Розглянуто
на засіданні кафедри
комп’ютерні системи
та мережі

Протокол № 1
від 27.08.2019 р.

Схвалено
на вченій раді
факультету інформаційних
технологій

Протокол № 1
від 29.08.2019 р.

Зміст

Лабораторна робота №1

Тема: ***Визначення критичного шляху на графі задачі в багатопроцесорних обчислювальних системах з різною організацією***

Мета: розглянути різні алгоритми пошуку критичного шляху на графі задачі, отримати практичні навички визначення критичного шляху при виконанні задачі в багатопроцесорних обчислювальних системах з різною організацією.

Короткі теоретичні відомості

Суть алгоритму пошуку критичного шляху (КШ) виконання задачі міститься у визначенні мінімально можливого та максимально можливого часу початку виконання вузлів графу із зваженими вузлами та дугами, тобто із врахуванням часу обробки вузлів графу задачі та передачі даних між вузлами графу. Даний алгоритм аналогічно відповідному алгоритму для графу без врахування часу передачі по магістралі оснований на послідовному проході по дереву графу від початкової вершини (вершин) до кінцевої та поверненню до початкової вершини. При початковому проході (рисунок 1.1) визначається мінімально можливий час початку виконання кожного вузла за формулою:

$$\mathrm{T}_{\min \mathrm{i}} = \max_{j=1}^{s}\left(T_{\min j} + t_j + T_{ji}\right) \qquad (1.1)$$

де s – число вузлів-попередників i-го вузла;

$T_{min\ i\ (j)}$ – мінімально можливий час початку виконання i-го (j-го) вузла;

t_j – час виконання j-го вузла;

T_{ji} – час передачі даних між вузлами j та i, якому присвоюється одне із значень множини $\{0,\ \tau_{ji},\ 2\tau_{ji}\}$ в залежності від способу організації пам'яті, де τ_{ji} – час передачі даних між вузлами j та i, що задається на вихідному графі задачі.

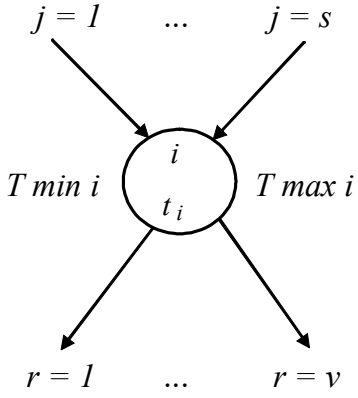


Рисунок 1.1 – Пояснення до визначення критичних вузлів

При повторному аналізі графу обчислювального процесу при проходженні від кінцевої вершини до початкової визначається максимально можливий час початку виконання вузла за формулою (1.2):

$$\mathrm{T}_{\max\,\mathrm{i}} = \min_{r=1}^{v}\left(T_{\max\,r} - t_i - T_{ir}\right) \qquad (1.2)$$

де v – число вузлів-наступників і-го вузла;

$T_{max\ i\ (r)}$ – максимально можливий час початку виконання i-го (r-го) вузла;

$t_{\ i}$ – час виконання i-го вузла;

$T_{i\ \ r}$ – час передачі даних i-го вузла вузлу r, значення якому присвоюються аналогічно $T_{j\,i}$.

При цьому для початкової вершини (вершин) $T_{min\ i} = 0$, а для кінцевої вершини мінімально можливий час початку її виконання співпадає із максимально можливим часом початку виконання – $T_{max\ i} = T_{min\ i}$.

i -й вузол є критичним , якщо виконується рівність $T_{min\ i} = T_{max\ i}$.

Критичним шляхом графу є множина послідовних вузлів, що починаються вхідною вершиною та закінчуються вихідною вершиною, у яких $T_{min\ i} = T_{max\ i}$, і в якому для кожної пари вузлів графу співвідношення (1.1) і (1.2) визначаються сусідньою вершиною у парі.

При цьому необхідно відмітити наступне. Довжиною критичного шляху визначається мінімальний час виконання задачі. Граф задачі при виконанні її на багатопроцесорній обчислювальній системі (БОС) із різною організацією може мати різні критичні шляхи внаслідок змін часу передач між вузлами. Останнє визначається способом організації пам'яті. Наприклад, у БОС тільки з загальною пам'яттю (без локальної пам'яті процесорів) при визначенні критичного шляху час передачі даних від j-го вузла і-му та від і-го вузла r-му необхідно подвоювати. Подвійне врахування цього часу відбувається тому, що при передачі даних, по-перше, необхідний час $\tau_{\mathrm{j\ i}}$, щоб передати дані j-го вузла у загальну пам'ять, і по-друге, час $\tau_{\mathrm{j\ i}}$, щоб забрати дані із загальної пам'яті та передати і-му вузлу, навіть у випадку виконання і-го та j-го вузлів на одному процесорі. У БОС із розподіленою пам'яттю кожний процесор має локальну пам'ять, у якій можуть зберігатися проміжні результати. При цьому, якщо вузли j та і обробляються одним процесором, то час обміну між ними рахується рівним 0, тобто $T_{ji}=0$, якщо різними, то час обміну між вузлами дорівнює τ_{ji}.

Розглянемо приклад пошуку критичного шляху для графу задачі без врахування часу передач (рисунок 1.2). Цифра всередині кожного кола (вузла графу) – номер вузла, цифра над колом – час виконання вузла $t_{\,i.}$ Дуги графу не зважені, а отже, $T_{ji} = T_{ir} = 0$.

Основним припущенням при пошуку КШ на графі є те, що граф задачі реалізується в системі з необмеженими ресурсами, у даному випадку на БОС із необмеженим числом процесорів.

При русі по графу від початкової вершини до кінцевої зліва від вузла записується мінімально можливий час початку виконання цього вузла $T_{min\ i}$. Наприклад, вузол 6 може почати виконуватися, коли виконаються 2 та 3 вузли. Тоді $T_{min\ 6} = 7$, так як мінімально можливий час початку виконання

шостого вузла є максимальним із часів закінчення другого $(T_{min\ 2} + t_2)$ та третього $(T_{min\ 3} + t_3)$ вузлів.

При русі по графу назад від кінцевої вершини до початкової, справа від вузла записується максимально можливий час початку виконання цього вузла $T_{max\ i}$. Наприклад, для вузла 2 за формулою (1.2) знаходимо:

$$T_{max\ 5} - t_2 = 11 - 3 = 8;\ T_{max\ 6} - t_2 = 7 - 3 = 4.$$

Таким чином, $T_{max\ 2} = min(T_{max\ 5}, T_{max\ 6}) = 4$.

Отже, для графу, зображеного на рисунку 2 , є два критичні шляхи: 1; 2; 6; 9; 11; 12 та 1; 4; 10; 11; 12. При цьому шлях 1; 2; 9; 11; 12 не є критичним, оскільки для пари вузлів 2 та 9 співвідношення (1.1) та (1.2) не визначаються сусіднім у парі вузлом. Наприклад, $T_{min\ 9}$ визначається не вузлом 2, а вузлом 6, аналогічно і $T_{max\ 2}$ не визначається вузлом 9, тобто:

$$T_{min\ 9} \neq T_{min\ 2} + t_2 = 4 + 3 = 7;$$
$$T_{max\ 2} \neq T_{max\ 9} - t_2 = 12 - 3 = 9.$$

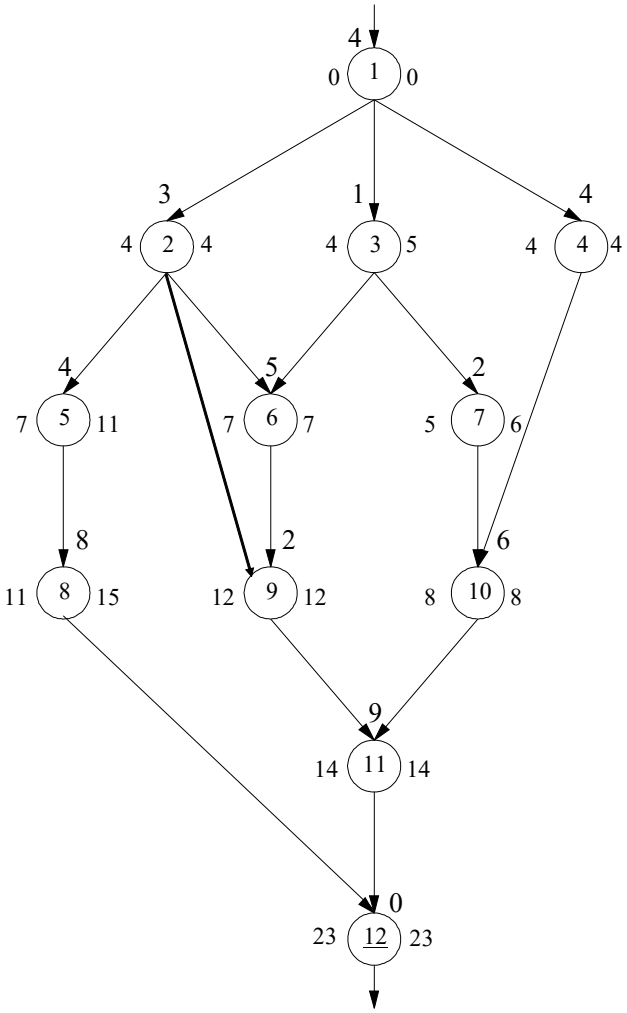


Рисунок 1.2 – Визначення критичного шляху для графу задачі без врахування часу передач

Звідси, для даної прикладної задачі $T_{min} = 23$ МТ, $T_{max} = 48$ МТ.

Розглянемо тепер приклад визначення КШ для графу середньо-зв'язаної задачі із врахуванням передач (рисунок 1.3). Дуги графу задачі зважені числами, що відповідають часу зайнятості шини.

Пошук критичного шляху на графі, а відповідно, і мінімального часу його виконання із врахуванням часу передачі даних від вузла до вузла, у загальному випадку зводиться до задачі повного перебору. Проте, при введенні ряду обмежень на умову передачі даних, а вони зв'язані із організацією обмінів між процесорами та пам'яттю, можна запропонувати алгоритми, які скорочують повний перебір і видають необхідний результат.

Зробимо наступні припущення. Граф задачі реалізується на системі з необмеженими ресурсами. Кожний вузол графу може передавати (приймати) дані по усім вихідним (вхідним) гілкам паралельно.

При цих умовах T_{min} визначається, як було вказано вище, в залежності від способу організації пам'яті, оскільки цим визначається час передачі між вузлами. Для випадку БОС із загальною пам'яттю час подвоюється. Розглянувши рисунок 1.3, можна зробити висновок, що критичний шлях графу із врахуванням передачі змінився – 1; 3; 7; 10; 11; 12, а $T_{min} = 46$ МТ.

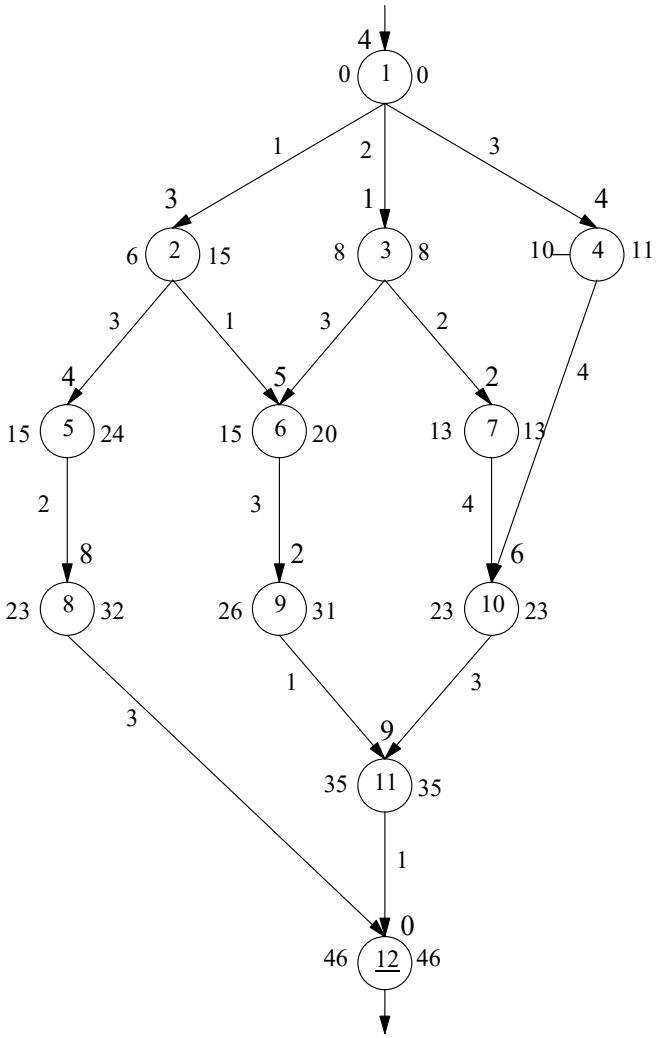


Рисунок 1.3. Визначення критичного шляху для графу задачі із врахуванням передач в БОС із загальною пам'яттю

Розглянемо тепер алгоритм пошуку КШ для графу задачі із врахуванням передач в БОС із розподіленою пам'яттю (рисунок 1.4). Відмітимо, що ця задача аналогічна задачі, граф якої наведений на рисунку 3. Потрібно звернути увагу на те, що метою пошуку КШ при реалізації задачі на БОС із розподіленою пам'яттю є визначення серед вузлів-наступників ($r=1, ..., v$) (дивитись рисунок 1.1) вузла, котрий призначається на процесор, що виконує вузол-попередник. У цьому випадку дані від вузла-попередника не передаються вузлу-наступнику ($T_{ir} = 0$).

Наступні припущення є основними при реалізації графу задачі на БОС із розподіленою пам'яттю:

1. Граф задачі реалізується на системі із необмеженими ресурсами.

2. Від кожного вузла графу дані можуть передаватися паралельно по вихідним гілкам.

3. Кожний вузол графу може приймати дані паралельно по вхідним гілкам.

4. Кожний вузол графу може мати не більше однієї вхідної та вихідної гілок, час передачі якими в результаті пошуку критичного шляху дорівнює нулю.

Алгоритм визначення КШ для графу задачі, що виконується в БОС із розподіленою пам'яттю, визначається тим, що при розрахунку за формулою (1.1) мінімально можливого часу початку виконання і-го вузла T_{mini} визначається одна із гілок, що входять у нього, часу передачі по якій присвоюється значення, рівне нулю. При цьому ця гілка повинна вносити максимальний ефект у зменшення $T_{min\ i}$. Вочевидь, що не завжди з першої ж спроби можна визначити $T_{min\ i}$. Наприклад, визначивши гілку d_i, що входить в і-й вузол, і яка обумовлює $T_{min\ i}$, необхідно провести аналіз вузла d, із якого виходить ця гілка. Якщо у цього вузла вже є одна вихідна гілка, котрій присвоєне значення $T_{dk} = 0$ (тобто визначений k-й вузол-наступник d-го вузла, і, відповідно, процесор залишив необхідні дані для виконання k-го вузла у своїй локальній пам'яті), то гілка d_i із розгляду виключається. Знаходиться наступна за значенням ефективності гілка, що входить у і-й вузол тощо.

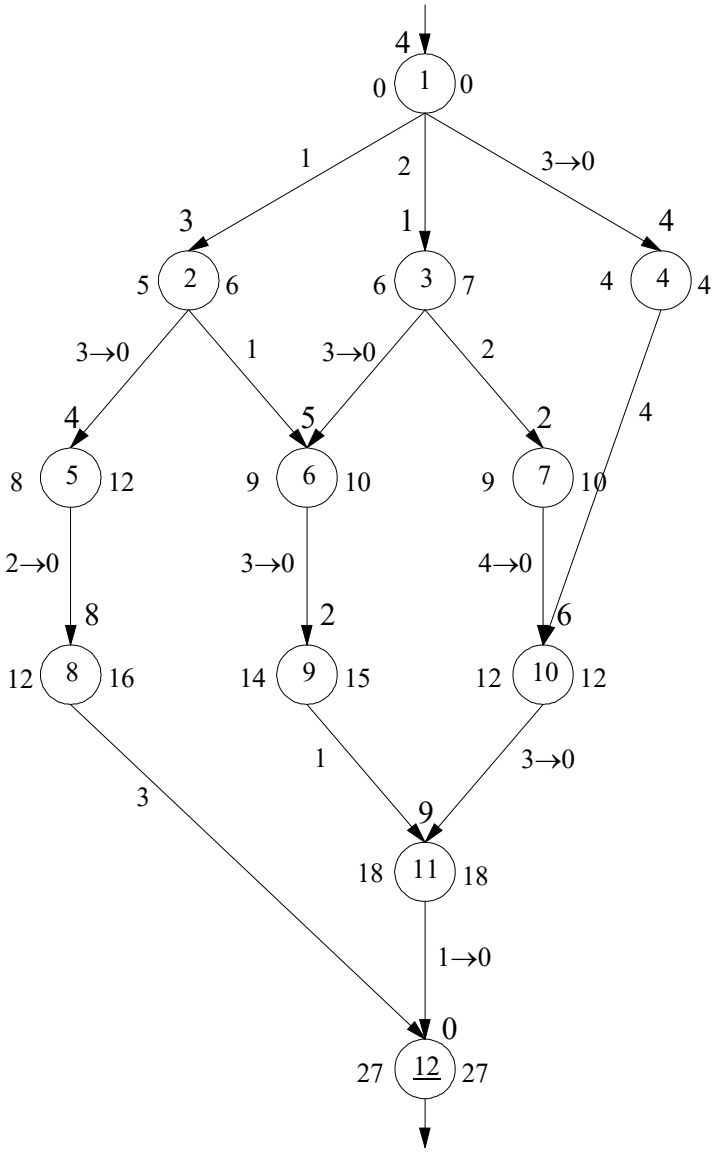


Рисунок 1.4 – Визначення критичного шляху для графу задачі із врахуванням передач в БОС із розподіленою пам’яттю

Якщо у d-го вузла немає вихідною гілки, котрій присвоюється значення, рівне нулю, то для цього d-го вузла визначається наявність такої гілки, у якої $\tau_{d\,k} > \tau_{d\,i}$. Якщо така гілка є, то і-й вузол виключається із розгляду до тих пір, поки не буде прийняте рішення про присвоєння або неприсвоєння гілці d_k значення $T_{dk} = 0$. Якщо такої гілки немає, то T_{di} присвоюється значення, рівне нулю, і уже із врахуванням цього визначається $T_{min\,i}$.

Потрібно звернути увагу на наступне. Якщо k-й вузол розташований на більш низькому ярусі графу, чим і-й вузол, то при знаходженні $T_{min\,k}$ аналізуються усі шляхи від вершини d до вершини k, а значення 0 може бути присвоєне (або не присвоєне) у відповідності з правилами, приведеними вище для вузла і, одній із гілок, що виходять із вузла d (необов’язково для гілки d_k).

Максимально можливий час початку виконання вузла T_{max} і розраховується за формулою (1.2) при зворотному русі по графу уже із врахуванням визначених вище нульових гілок. Дані гілки помічені на рисунок 1.4 стрілкою (наприклад, 3 $\rightarrow$ 0). У наведеному прикладі граф задачі має єдиний КШ – 1,4,10,11,12, довжина якого дорівнює T_{min} *=27* МТ.

Із розглянутих прикладів видно, що довжина критичного шляху (T_{min}) являється основним параметром, що характеризує інформаційно-логічну структуру обчислювального процесу, який реалізує розв'язання заданої прикладної задачі. У випадку врахування часу передачі даних від одного вузла до іншого при рішенні задачі у БОС із обмеженими ресурсами, значення мінімального часу її виконання збільшується і визначається характеристиками реальної структури БОС. Очевидно, що це збільшення буде тим менше, чим вдаліше розташовані вузли графу по процесорам БОС.

Хід роботи

1. Вивчити наведений вище теоретичний матеріал.
2. За номером у журналі групи обрати графи обчислювального процесу (графи видаються викладачем в електронному вигляді): граф задачі без врахування часу передач, граф задачі із врахуванням передач в БОС із загальною пам'яттю та граф задачі із врахуванням передач в БОС із розподіленою пам'яттю.
3. Провести розрахунок мінімально можливого і максимально можливого часу початку виконання кожного вузла обчислювального процесу для кожного графу.
4. Визначити критичні шляхи і обґрунтувати вибір.
5. Зробити висновок по роботі.

Звіт має містити

1. Тему, мету лабораторної роботи.
2. Завдання для виконання.
3. Варіанти обраних графів.
4. Розрахунки необхідних часових параметрів та критичних шляхів.
5. Висновки.

Лабораторна робота №2

Тема: ***Побудова кластерної системи на базі linux-дистрибутиву BCCD.***

Мета: ознайомитись з ОС BCCD; розглянути основні типи кластерних систем; на базі декількох комп'ютерів створити кластер для високопродуктивних обчислень; перевірити швидкодію кластера на прикладі програми обчислення числа Пі.

Короткі теоретичні відомості

1 Винекнення кластерної технології

До середини 1990рр. минулого століття, основний напрям розвитку суперкомп'ютерних технологій був пов'язаний з побудовою спеціалізованих багатопроцесорних систем з масових мікросхем. Один із підходів – SMP (Symmetric Multi Processing), мав на увазі об'єднання багатьох процесорів з використанням загальної пам'яті, що значно облегшувало програмування, але вимагало великих потреб від самої пам'яті. Зберегти швидкодію таких систем при збільшенні кількості вузлів до десятків було практично неможливо. Окрім цього даний підхід став одним із найдорожчих підходів у апаратній реалізації. На порядок дешевшим та майже нескінченно масштабованим став підхід MPP (Massively Parallel Processing), при якому незалежні спеціалізовані обчислювальні модулі об'єднувались спеціалізованими каналами зв'язку, причому і ті та інші створювалися під конкретний суперкомп'ютер і в ніяких інших цілях не використовувались.

Ідея створення «Кластеру» робочих станцій фактично є розвитком методу MPP, адже логічно система MPP – не сильно відрізнялась від звичайної локальної мережі. Ця ідея стала реальністю у середині 90-х рр. коли з'явилася шина PCI у багатьох персональних ЕОМ, і появі відносно дешевої локальної мережі (ETHERNET).

У перше поняття кластер ввела компанія DEC, майже 18 років тому назад. І за її визначенням кластер – це група обчислювальних машин, які пов'язані між собою та функціонують як один вузол обробки інформації, і для кінцевого користувача виглядає як один комп'ютер. Тобто, кластер являє собою два або більше комп'ютерів (які часто називають вузлами кластеру), об'єднаних за допомогою мережевих технологій (наприклад FastEthernet) на базі шинної топології. В якості вузлів кластеру можуть бути обрані сервери, робочі станції і навіть звичайні ЕОМ.

Кластеризація може бути створена на різних рівнях комп'ютерної системи, включаючи апаратне забезпечення, операційні системи, програми утиліти, системи управління та прикладні програми. І чим більше рівнів системи об'єднані кластерною технологією, тим більша надійність, масштабованість та керованість кластера.

2 Основні типи кластерів

Всі існуючі на даний момент кластери можна розділити на три типи:

1) системи високої надійності (HA – High Available);
2) системи високопродуктивного обчислення (HPC – High Performance Computing)
3) багатопоточні системи.

2.1 Системи високої надійності

Такі типи кластерів є найкращою схемою для збільшення надійності інформаційно – обчислювальної системи. Завдяки єдиному представленні, окремі вузли чи компоненти кластеру можуть непомітно для користувача змінити непридатні до роботи компоненти, на час їхнього налагодження, забезпечуючи таким чином безперервну та надійну роботу серверів, баз даних.

Також за допомогою кластерів даного типу, можливе створення катастрофонадійних рішень, за рахунок рознесення вузлів кластеру на декілька кілометрів та забезпечуючи механізми синхронізації даних між такими вузлами.

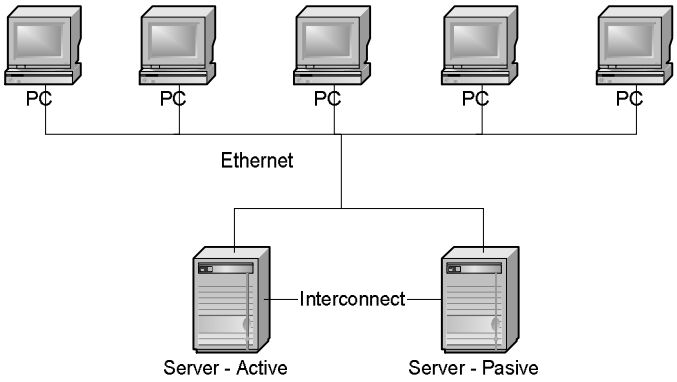


Рисунок 2.1 – Типова схема кластеру HA

2.2 Системи високопродуктивного обчислення

Такі кластери призначені для паралельних обчислень. Кластери для високопродуктивних обчислень як правило зібрані з великої кількості комп'ютерів. Розробка таких кластерів являє собою складний процес, який потребує на кожному кроці правильних узгоджень, таких питань як інсталяція, експлуатація та одночасне управління великою кількістю комп'ютерів, технічних вимог паралельного та високопродуктивного доступу до одного файлу, та між процесорного зв'язку між вузлами і координації роботи у паралельному режимі.

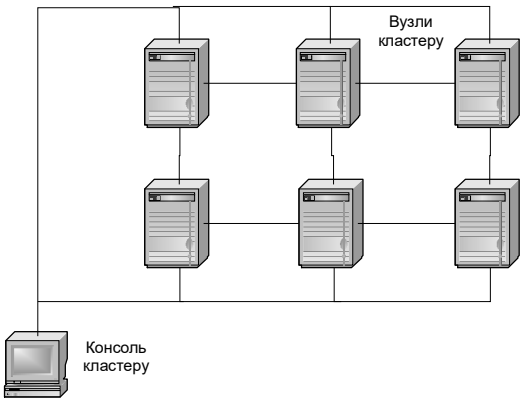


Рисунок 2.2 – Типова схема кластера HPC

2.3 Багато поточні системи

Такі системи використовуються для забезпечення єдиного інтерфейсу до ряду ресурсів, котрі з часом можуть випадково збільшуватись у розмірі. Найбільш загальний приклад цього являє собою група веб серверів.

3 Створення кластеру високої готовності

Для побудови обчислювальних кластерів (HA) можна використовувати різне мережене обладнання. І так як, характеристики звичайного мереженого обладнання значно поступаються спеціалізованим комунікаціям у «нормальних» MPP комп'ютерах, пропускна здатність мережі, яка об'єднує вузли кластеру, в багатьох випадках є найголовнішою характеристикою, яка впливає на швидкодію кластеру.
Також дуже важливою характеристикою мережі є латентність.

Латентність – це середній час передачі між викликом функції передачі і самою передачею. Фактично пропускна здатність і латентність не тільки характеризують кластер, але й визначають клас задач, які зможуть ефективно розв'язуватися на ньому. І тому для мало бюджетних кластерів використання Myrinet, SCI, cLAN є нереальним, але можливе використання технології FastEthernet чи GigabitEthernet.

Отже для створення кластеру використаємо звичайний комутатор з підтримкою швидкості передачі 10/100/1000 Мб/с.

У якості мереженої карти можна використати будь-яку мережеву карту, яка підтримує 100BaseTx чи GigabitEthernet.

У якості вузлів кластеру можна використати звичайні комп'ютери.

У якості програмного забезпечення можна використовувати наступні ОС: Red Hut Linux, FreeBSD, BCCD, MacOs.

Хід роботи

1. Для налагодження кластеру, оберіть серед вузлів (комп'ютерів об'єднаних в мережу) вузол, який буде головним – «консоль» кластера, через який будуть завантажуватися та запускатися на виконання програми.
2. Вставте у лоток CD чи DVD-приводу диск BCCD, та перезавантажте комп'ютер.
3. Після завантаження ви побачите наступний екран:

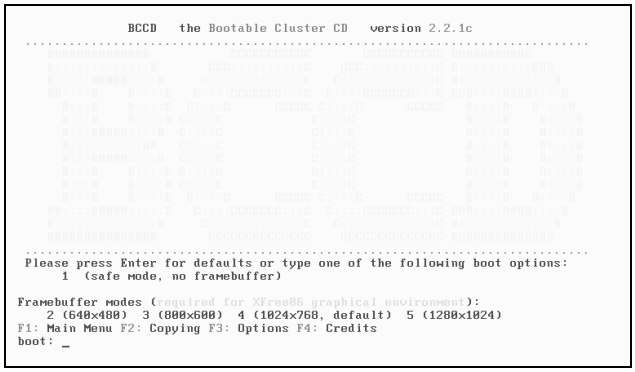


Рисунок 2.3 – Головне вікно BCCD

4. Натиснути Enter; після чого почнеться завантаження операційної системи. Через деякий час система запитає пароль для користувача (рисунок 2.4). Введіть, наприклад, «11111».

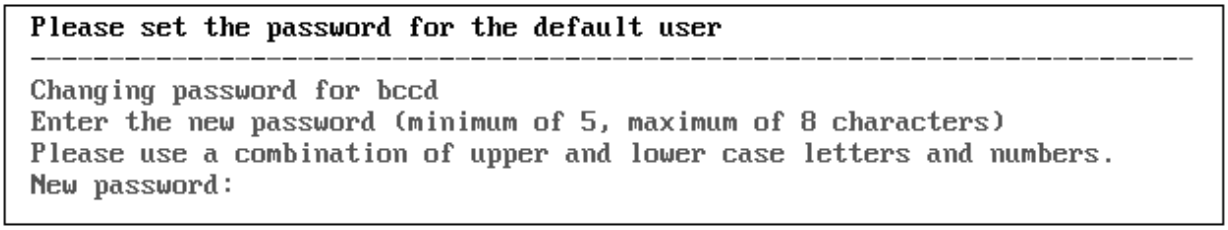


Рисунок 2.4 – Запит паролю користувача

5. Вкінці завантаження ОС з'явиться вікно запиту DHCP (рисунок 2.5):

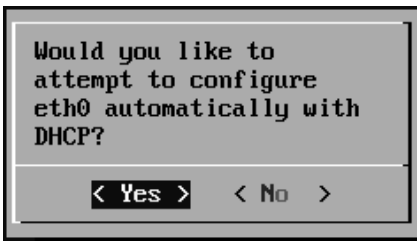


Рисунок 2.5 – Вікно запиту DHCP

У відповідь потрібно обрати опцію «NO», бо налаштування мережевої карти потрібно провести у ручному режимі.

6. На запит «Налагодити параметри вручну?» (рисунок 2.6) потрібно обрати опцію «YES».

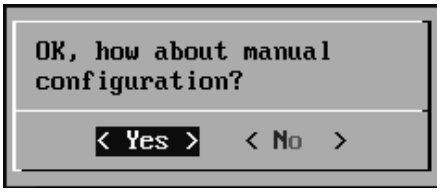


Рисунок 2.6 – Вікно запиту типу налагодження

7. Для «консолі» кластеру введіть наступні параметри:

IP адреса 192.168.1.1
Маска – 255.255.255.0
IP адреса шлюзу за умовчанням – 192.168.1.254
IP адреса DNS серверу – 192.168.1.1

8. Після налаштування ви отримаєте вікно повідомлення (рисунок 2.7). Воно сигналізує про те, що ви налаштували перший вузол кластеру.

Рисунок 2.7 – Вікно повідомлення

Увага!!! Після отримання надпису про введення імені та паролю користувача, не вводьте їх до тих пір поки не будуть налаштовані інші вузли кластера.

9. Налаштуйте інші вузли кластеру використовуючи наступні данні:

IP адреса 192.168.1.x
Маска – 255.255.255.0
IP адреса шлюзу за умовчанням – 192.168.1.254
IP адреса DNS серверу – 192.168.1.1

10. Після налагодження останнього вузла ввійдіть під користувачем, починаючи з останнього вузла, і натисніть «Так», коли запропонують розіслати ключі RSA, вони потрібні для ідентифікації та аутентифікації вузлів між собою.
11. На головному вузлі, «консолі», після розсилання RSA ключа, увійдіть під root. Для цього виконайте наступну команду: exit. А потім увійдіть під логіном root (пароль – letmein).
12. Зкопіюйте файл flop.f до каталогу /home/bccd/mpich/examples та надайте всім користувачам права на зміну та модифікацію файла. Для цього виконайте наступні команди:

```
#cd /mnt/discs/disc1/part2
#cp flop.f  /home/bccd/mpich/examples
#cd  /home/bccd/mpich/examples
#chmod ugo+rwx flop.f
#exit
```

13. Зайдіть під логіном bccd.
14. Зробіть синхронізацію вузлів кластеру: команда #bccd-snarfhosts
15. Скомпілюйте файл flop.f та розішліть його на всі вузли кластеру. Для цього виконайте команди:

```
#cd /home/bccd/mpich/examples
#mpif77 flop.f -o exe
#bccd-syncdir  /home/bccd/mpich/examples /home/bccd/machines          (2.1)
```

16. Поставте на виконання програму exe:

```
#mpirun –np 2 –machinefile /home/bccd/machines  /tmp/XXX/exe
```

де

-np 2 – кількість вузлів котрі будуть приймати участь в обчисленнях;

XXX – назва каталогу, яку ви отримаєте в результаті виконання команди (2.1).

17. Зробіть висновки по результатам роботи програми.

Звіт має містити

1. Тему, мету лабораторної роботи.
2. Порядок виконання роботи.
3. Опис виконання команд.
4. Висновки.

Лабораторна робота №3

Тема: ***Дослідження структурно-топологічних характеристик обчислювальних систем та комунікаційної мережі комп'ютерних систем.***

Мета: дослідити структурно-топологічні характеристики обчислювальних систем, навчитися їх розраховувати; промоделювати комп'ютерну систему з різною топологією мережі передачі даних; провести експерименти з метою визначення основних топологічних характеристик.

Короткі теоретичні відомості

1. Зв'язність структури

Дана характеристика *S* дозволяє виявляти наявність обривів в структурі, висячі вершини і інші її властивості. Найбільш повно структура *S* (при представленні системи орграфом) визначається матрицею зв'язності $C=\|C_{ij}\|$. Елементи матриці *C* можна обчислити на основі матриці $A_s=\sum_{k=1}^{n}A^k$, де A^k – матриця суміжності *k*-го порядку. Елемент $C_{ij}=1$, якщо $a_{ij}^k=1$; $C_{ij}=0$, якщо $a_{ij}^k=0$. для неорієнтованих графів зв'язність усіх елементів в структурі відповідає виконанню наступної умови:

$$S=\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}a_{ij}^{k}\geq n-1, i\neq j \qquad (3.1)$$

Перша частина нерівності (3.1) визначає необхідне мінімальне число зв'язків у структурі неорієнтованого графа, що має *n* вершин.

2. Структурна надлишковість

Структурний параметр, що відображає перебільшення загального числа зв'язків над мінімально необхідним, називається структурною надлишковістю *R*:

$$R = \frac{1}{2}\left[\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^{k}\right]\frac{1}{n-1} - 1 \qquad (3.2)$$

Ця характеристика використовується для оцінки економічності і надійності систем, що досліджуються. Для систем з максимальною надлишковістю, що мають структуру типу «повний граф» $R \geq 0$, для систем з максимальною надлишковістю $R=0$, для незв'язних систем $R \leq 0$. Таким чином, система з більшим R потенційно більш надійна. Її можна доповнити іншим параметром, який враховує нерівномірність зв'язків $\sum^2$.

Рівномірний розподіл зв'язків в структурі неорієнтованого графу, що має m ребер та n вершин, характеризується середнім степенем вершини $\bar{\rho} = \frac{2m}{n}$. Тоді, ввівши поняття відхилення $\rho_i - \bar{\rho}$, де ρ_i - дійсний степінь *i-ї* вершини заданого графу, можна обчислити відхилення заданого розподілу степенів вершини від рівномірного:

$$\sum^2 = \sum_{i=1}^{n}\left(\rho_i - \bar{\rho}\right)^2 = \sum_{i=1}^{n}\rho_i^2 - 2\bar{\rho}\sum_{i=1}^{n}\rho_i + 4\frac{m^2}{n^2} = \sum_{i=1}^{n}\rho_i^2 - 2\cdot 2\frac{m}{n}\cdot\sum_{i=1}^{n}\rho_i + 4\frac{m^2}{n^2} =$$
$$= \sum_{i=1}^{n}\rho_i^2 - 4\frac{m}{n}\sum_{i=1}^{n}\rho_i + 4\frac{m^2}{n^2} \qquad (3.3)$$

Показник $\sum^2$ характеризує недовикористані можливості заданої структури, що має m ребер та n вершин, в досягненні максимальної зв'язності.

3. Структурна компактність

Для кількісної оцінки структурної компактності вводиться параметр, що відображає близькість елементів між собою. Близькість двох елементів *i* та *j* між собою визначається через мінімальну довжину шляху для орграфу d_{ij}. Тоді величина:

$$Q = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}\ (i \neq j) \qquad (3.4)$$

і є структурною компактністю (близкістю).

Окрім Q часто використовують відносну близькість

$$Q_{otn} = \frac{Q}{Q_{min}} - 1 \qquad (3.5)$$

де $Q_{min}=n(n-1)$ – мінімальне значення компактності для структури системи «повний граф».

Структурну компактність можна характеризувати іншою характеристикою – діаметром структури:

$$d = max\ d_{ij} \qquad (3.6)$$

4. Степінь централізації

Для кількісної оцінки степені централізації в структурі використовується поняття індексу центральності:

$$\delta = (n-1)\cdot(2z_{max} - n)\cdot\frac{1}{z_{max} - (n-2)} \qquad (3.7)$$

де z_{max} – максимальне значення величини

$$z_i = \frac{Q}{2}\left(\sum_{i=1}^{n} d_{ij}\right)^{-1} \quad i = 1\ldots n,\ i \neq j \qquad (3.8)$$

Для структур систем, що мають максимальну степінь централізації $\delta=1$, і для структур з рівномірним розподілом зв'язків $\delta=0$.

5. Ранг елемента

Використовується при представленні структури у вигляді орграфу. Дана характеристика дозволяє розподілити елементи системи в порядку їх значущості. Значущість елемента визначається лише числом зв'язків даного елемента з іншими.

Чим вище ранг елементу, тим сильніше він зв'язаний з іншими елементами системи і тим тяжкими будуть наслідки при зміненні якості його функціонування.

$$z_i = \frac{\sum_{j=1}^{n} a_{ij}^{(k)}}{\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^{(k)}} \qquad (3.9)$$

де a_{ij}^{k} – елемент матриці A^k, $k=3\div4$.

6. Комунікаційна мережа комп'ютерних систем

Найважливішим елементом архітектури комп'ютерної системи є комунікаційна мережа, що зв'язує процесори з оперативною пам'яттю, процесори між собою, процесори з іншими пристроями. Вона має вагомий вплив на продуктивність системи. Трафік в такій мережі складається із даних і команд, що пересилаються. Основною характеристикою є пропускна спроможність, що вимірюється в бітах за секунду.

Структура ліній комутації (топологія мережі передачі даних) визначається з урахуванням можливостей ефективної технічної реалізації. До числа типових статичних топологій зазвичай відносять наступні схеми комунікації процесорів:

- *повний граф (мережа з повністю зв'язаною топологією)* – система, в якій між будь-якими парами процесорів існує пряма лінія зв'язку, внаслідок чого дана топологія забезпечує мінімальні витрати при передачі даних, але складно реалізується при великій кількості процесорів;

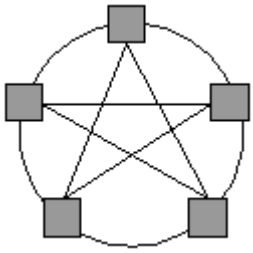


Рисунок 3.1 – Топологія «повний граф»

- *лінійка (одномірна решітка)* – система, в якій кожен процесор має лінії зв'язку лише з двома сусідніми (з попереднім і наступним) процесорами; така схема реалізується просто і відповідає структурі передачі даних при розв'язанні багатьох задач;

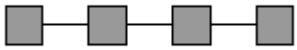


Рисунок 3.2 – Топологія «лінійка»

- *кільце* – дана топологія отримується із лінійки процесорів з'єднанням першого і останнього процесорів лінійки;

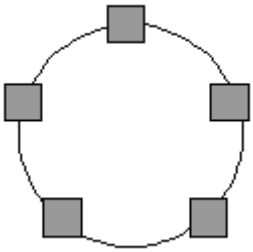


Рисунок 3.3 – Топологія «кільце»

- *зірка* - система, в якій всі процесори мають лінії зв'язку з центральним вузлом, є ефективною при організації централізованих схем обчислень;

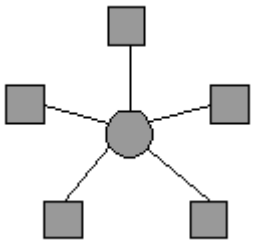


Рисунок 3.4 – Топологія «зірка»

- *решітка* – система, в якій граф ліній зв'язку утворює прямокутну сітку (зазвичай, дво- чи трьохвимірну); просто реалізується, ефективно використовується для розв'язання задач науково-технічного характеру;

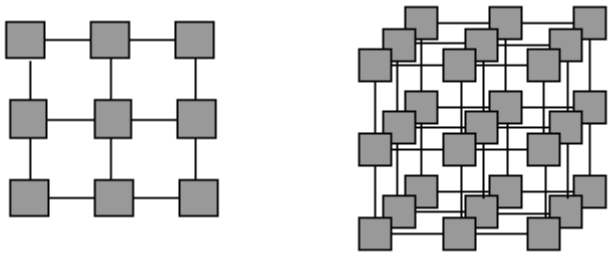


Рисунок 3.5 – Топологія «решітка»

- *гіперкуб* – дана топологія представляє окремий випадок структури решітки, коли по кожній розмірності сітки є лише два процесори (тобто гіперкуб має 2^N процесорів при розмірності N); цей варіант організації достатньо розповсюджений і має ряд ознак: два процесори з'єднуються, якщо двійкове представлення їх номерів має лише одну позицію, яка розрізняється; в *N*- вимірному гіперкубі кожен процесор зв'язаний з *N* сусідами; *N*-вимірний гіперкуб може бути розділений на два *(N-1)*-вимірних гіперкуба; найкоротший шлях між двома будь-якими процесорами має довжину, що співпадає з кількістю бітових значень в номерах процесорів, що розрізняються (відстань Хеммінга).

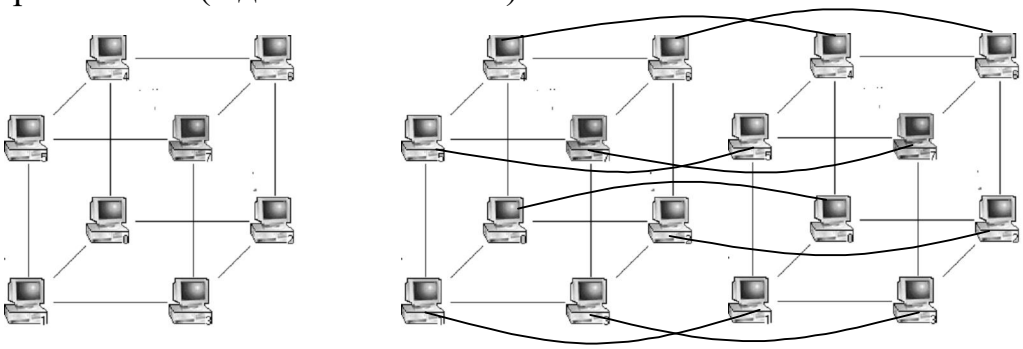


Рисунок 3.6 – Топологія «гіперкуб»

7. Характеристики топології мережі передачі даних

В якості основних характеристик найчастіше використовуються:

- *діаметр* – показник, що визначається як максимальна відстань між дома процесорами мережі; характеризує максимально-необхідний час для передачі даних між процесорами;
- *зв'язність* – показник, що характеризує наявність різних маршрутів передачі даних між процесорами мережі (наприклад, мінімальна кількість дуг, які потрібно видалити для розподілу мережі передачі даних на дві незв'язані області);
- *ширина бінарного ділення* – показник, що визначається мінімальною кількістю дуг, які потрібно видалити для розподілу мережі передачі даних на дві незв'язані області однакового розміру;

- *вартість* – показник, який визначається, як загальна кількість ліній передачі даних в багатопроцесорній системі.

Для порівняння в таблиці наводяться значення перелічених показників для різних топологій мережі передачі даних (p – кількість процесорів):

Таблиця 3.1 – Характеристики топології мережі передачі даних

Топологія	Діаметр	Ширина бісекції	Зв'язність	Вартість
Повний граф	1	$p^2 / 4$	$p - 1$	$p\,(p-1) / 2$
Зірка	2	1	1	$p - 1$
Повне двійкове дерево	$2 \log ((p+1)/2)$	1	1	$p - 1$
Лінійка	$p - 1$	1	1	$p - 1$
Кільце	$\lfloor p / 2 \rfloor$	2	2	p
Решітка ($N = 2$)	$2(\sqrt{p} - 1)$	$\sqrt{p}$	2	$2\,(p - \sqrt{p})$
Решітка-тор ($N = 2$)	$2\lfloor \sqrt{p} / 2 \rfloor$	$2\sqrt{p}$	4	$2p$
Гіперкуб	$\log p$	$p / 2$	$\log p$	$(p \log p) / 2$

Час передачі даних між процесорами визначає комунікаційну складову тривалості виконання процесу в комп'ютерній системі. Основний набір параметрів, що описують час передачі даних:

- *час початкової підготовки* $t_п$ характеризує тривалість підготовки повідомлення для передачі, пошуку маршруту в мережі тощо;
- *час передачі службових даних* $t_с$ між двома сусідніми процесорами (заголовок повідомлення, блок даних для виявлення помилок передачі тощо);
- *час передачі одного слова даних одним каналом передачі* $t_к$ – тривалість визначається смугою пропускання комунікаційних каналів в мережі.

До числа найрозповсюдженіших методів передачі даних відносяться два основні способи комунікації. Перший з них орієнтований *на передачу повідомлень* як нероздільних блоків інформації. При такому підході процесор, що має повідомлення для передачі, готує весь обсяг даних, визначає процесор-адресат і запускає операцію пересилання даних. Адресат спершу здійснює прийом всього повідомлення, а вже потім пересилає його далі за маршрутом. Час пересилання даних для методу передачі повідомлення розміром m маршрутом довжиною l визначається як:

$$t_{пд} = t_п + (mt_к + t_с)l. \qquad (3.10)$$

Другий спосіб оснований на представленні повідомлень у вигляді блоків інформації меншого розміру (пакетів), в результаті чого передача

даних зводиться до *передачі пакетов*. При такому методі комунікації приймаючий процесор може здійснювати пересилання даних далі за маршрутом безпосередньо після прийому чергового пакета, не чекаючи завершення прийому даних всього повідомлення. Тоді час пересилання даних визначається, як:

$$t_{пд} = t_n + mt_к + t_c l. \quad (3.11)$$

В більшості випадків метод передачі пакетів призводить до більш швидкого пересилання даних; крім того, зменшується необхідність в пам'яті для зберігання даних, що пересилаються, а для передачі пакетів можуть використовуватись одночасно різні комунікаційні канали. Але реалізація цього методу вимагає більш складного забезпечення мережі, при передачі пакетів можуть виникати конфліктні ситуації (дедлоки).

Хід роботи

Частина 1

1. Вивчити опис лабораторної роботи.
2. За останньою цифрою порядкового номеру в журналі обрати дані для дослідження із таблиці 3.2.
3. У відповідності з заданим варіантом обчислити значення S, R, $\sum^2$, Q, $Q_{отн}$, d, δ, z_i .

Частина 2

1. Промоделювати за допомогою програми ParaLab розв'язання задачі в багатопроцесорній комп'ютерній системі.
   1.1. Визначити топологію системи, задати число процесорів, встановити їх продуктивність, обрати характеристики комунікаційної мережі і спосіб комунікації.
   1.2. В меню *Задача* обрати *Сортування* та в меню *Задача>Параметры задачи* задать размер массива равным 4000.
   1.3. Встановити параметри візуалізації для обрання бажаного темпу демонстрації, способу відображення даних, що пересилаються, степені детальності візуалізації обчислень.
   1.4. Виконати експеримент, змінюючи:
   - метод передачі даних (повідомлення, пакети);
   - пропускну здатність мережі (від 100 до 600 Мбіт/с);
   - латентність (від 10 до 100 мкс);
   - продуктивність процесорів (від 0.5 до 1 ГФлопс).

   1.5. За результатами експериментів побудувати графіки (для обох методів передачі дани):
   - часу вирішення задачі від пропускної здатності мережі;
   - часу вирішення задачі від латентності;
   - часу вирішення задачі від продуктивності.
2. Звести результати експериментів пункту 4.1 до табличного вигляду.

3. Проаналізувати отримані числові дані і графіки та зробити висновки.

Таблиця 3.2 – Варіанти завдань до лабораторної роботи

Варіант	Кількість процесорів	Топологія
1	8	Лінійка
2	8	Кільце
3	8	Гіперкуб
4	8	Повний граф
5	9	Решітка
6	9	Лінійка
7	9	Кільце
8	9	Повний граф
9	10	Лінійка

Звіт має містити

1. Тему, мету лабораторної роботи.
2. Завдання для виконання.
3. Результати розрахунків частини 1.
4. Таблиці, графіки і висновки по частині 2.
5. Загальні висновки по роботі.

Лабораторна робота №4

Тема: ***Дослідження задачі складання розкладу в багатопроцесорній обчислювальній системі***

Мета: вивчити принципи розподілення вузлів обчислювального процесу (ОП) в однозадачному режимі з використанням різних стратегій призначення готових до виконання вузлів ОП.

Короткі теоретичні відомості

Під час виконання лабораторної роботи №1 було виявлено, що довжина критичного шляху T_{min} є основним параметром, який характеризує інформаційно-логічну структуру обчислювального процесу, що реалізує розв'язання заданої прикладної задачі. У випадку врахування часу передачі даних від одного вузла до іншого при розв'язанні задачі в багатопроцесорній обчислювальній системі з обмеженими ресурсами, значення мінімального часу її виконання збільшується і визначається характеристиками реальної структури системи. Очевидно, що це збільшення буде тим менше, чим вдаліше розподілені вузли графа по процесорам. Таким чином, виникає задача побудови оптимального розкладу (призначення готових до виконання вузлів обчислювального процесу на вільні процесори).

Задача призначення може розв'язуватись як у статичному, так і в динамічному режимах. В першому випадку вона виконується до початку реалізації обчислювального процесу в багатопроцесорній обчислювальній системі (БОС), в другому – безпосередньо в процесі реалізації обчислювального процесу.

Під розв'язанням задачі призначення розуміють процес розподілення вузлів графа задачі (набору задач), що виконується в БОС, між її процесорами, при якому визначається час початку виконання вузла, його тривалість і призначення процесора, який забезпечить це виконання. Модель процесу розподілення містить засоби, що описують ресурси, систему вузлів і дуг графа задачі (набору задач) і критерій оптимальності розподілення. Під ресурсами розуміють: модулі обробки (процесори), модулі пам'яті (розподілена, загальна чи змішана), внутрішньо системний інтерфейс (загальна чи мультиплексна шина). При побудові алгоритмів призначення у відмовостійких БОС в модель мають бути введені засоби, що описують систему забезпечення відмовостійкості БОС. Наприклад, засоби, що забезпечують введення додаткових копій вузлів графа задачі і додаткових процесорів.

Нехай в якості ресурсів в моделі використовується набір однотипних процесорів з однаковою швидкодією. На ньому виконується обчислювальний процес, який має мережну структуру і являє собою сукупність окремих алгоритмів (сегментів задачі) $Z=\{z_i\}$.

Формально модель виконання задачі можна уявити сукупністю $\{Z, \prec, T, W, \Theta\}$, де $Z=\{z_1, \ldots, z_L\}$ – множина сегментів задачі, що виконуються в системі (вузли графа); $\prec$ - означає задання в множині Z відношення, яке визначає послідовність виконання сегментів і інформаційні зв'язки між ними (зв'язність вузлів); $T=\{t_1, \ldots, t_L\}$ – вектор часу виконання сегментів на процесорі з заданою швидкодією; $W=\{w_1, \ldots, w_L\}$ – вектор коефіцієнтів важливості сегментів, відповідних коефіцієнтам відносних втрат ефективності із-за невиконання сегментів задачі внаслідок відмови процесора, на якому виконується даний сегмент; $\|\Theta\| = \|\tau_{iq}\|$, i = 1…L, q = 1…L-1 – матриця часу зайнятості шини з заданою пропускною спроможністю при передачі даних між вузлами i та q.

В якості моделі задачі використовується направлений граф, вузли якого відображають сегменти задачі. При побудові алгоритмів, що реалізують задачу призначення, направлений граф перетворюється в таблицю зв'язності, елементи якої

$$A_{i,j} = \begin{cases} 1, \text{ якщо вихідна інформація вузла } z_i \text{ є вхідною для вузла } z_j \\ 0 \text{ - в іншому випадку} \end{cases}$$

В якості критеріїв оптимальності розподілення вузлів в БОС використовуються зазвичай або мінімум часу виконання задачі (набору задач) при обмеженні на число процесорів, або мінімум числа необхідних процесорів при обмеженні на час розв'язання задачі (набору задач).

Похідні критерії:

- максимум завантаження процесора (коефіцієнт завантаження) $К_{зав\ i} = Т_i / Т_{вик}$, де $Т_i$ – час, протягом якого і-й процесор зайнятий обробкою задачі, $Т_{вик}$ – час виконання задачі;
- мінімум простоїв кожного процесора $К_{пр\ i} = 1-К_{зав\ i}$;
- коефіцієнт прискорення $К_{приск} = Т_{max} / Т(n)$, де $Т_{max}$ – час розв'язання задачі на одному процесорі, Т(n) – час розв'язання задачі на n процесорах;
- максимальний час розв'язання задачі $Т_{max\ розв}=\sum_{i=1}^{k} t_i$, де k – число вузлів задачі.

Під час виконання лабораторних робіт використовуються методи наближеної оптимізації (евристичні методи). В таких розкладах вузлам задачі надається пріоритет за тими чи іншими правилами (стратегії призначення), після чого вузли впорядковуються у вигляді лінійного списку за зменшенням пріоритетів. Будуть використовуватись наступні стратегії: вибір вузла з мінімальним часом виконання; вибір вузла з максимальним часом виконання; вибір вузла, що належить критичному шляху; вибір вузла, що має найбільшу кількість зв'язків з наступними вузлами.

Таким чином, постановка задачі організації паралельних обчислень в БОС зводиться до реалізації наступної цільової функції: визначити мінімальну кількість процесорів, шин зв'язку комутаційної мережі, модулів пам'яті, спосіб організації пам'яті і тип стратегії призначення, що забезпечують виконання задачі (набору задач) за заданий час.

В даній лабораторній роботі потрібно вивчити принципи розподілення вузлів обчислювального процесу в однозадачному режимі, провести аналіз часових діаграм з метою визначення часу простою процесорів із-за неготовності вузлів задач. В інтервалах простою («порожнинах») можуть бути виконані копії сегментів, готових до виконання. Тим самим збільшується ймовірність знаходження помилок в БОС, яка розраховується за формулою $P_{пом} = \sum T_{дубл} / T_{max}$, де $\sum Т_{дубл}$ – сумарний час виконання копій вузлів задач, $Т_{max}$ – сумарний час виконання всіх вузлів.

В процесі виконання необхідно побудувати часові діаграми виконання обчислювального процесу, проаналізувати «порожнини» і визначити оптимальну множину копій вузлів для досягнення максимальної ймовірності знаходження помилки при заданому часі виконання обчислювального процесу.

Хід роботи

Частина 1

(!!!) Дана частина виконується без моделювання на комп'ютері (див. рекомендації щодо виконання даної роботи)

1. Вивчити опис лабораторних робіт.
2. Для графу задачі без врахування часу передач (часові характеристики якого розраховувались в л.р. №1) з використанням різних стратегій

розподілити вузли ОП по процесорам для виконання ОП за мінімальний час, визначивши необхідну для цього кількість процесорів, коефіцієнти їх завантаження.
3. Побудувати часову діаграму.

Частина 2

4. Промоделювати за допомогою програми LaboratoryWork1 граф, що досліджується, з використанням різноманітних стратегій вибору готових вузлів ОП для досягнення заданого часу виконання графу ОП.
5. Результати моделювання для різних стратегій вибору готових вузлів звести в таблицю (кількість процесорів, час розв'язання задачі, коефіцієнти завантаження).
6. За результатами моделювання побудувати залежності t(n) для різних стратегій вибору готових вузлів ОП, де t – час виконання ОП, n – кількість процесорів, на яких виконується ОП.
7. Порівняти отримані графіки, обрати кращу стратегію і пояснити результати.

Частина 3

(!!!) Дана частина виконується без моделювання на комп'ютері (див. рекомендації щодо виконання даної роботи)

8. Розподілити вузли ОП в режимі максимального заповнення "пустот" часової діаграми розв'язанням копій вузлів ОП.
9. Визначити ймовірність знаходження помилки $P_{пом}$.

Звіт має містити

6. Тему, мету лабораторної роботи.
7. Завдання для виконання.
8. Результати виконання по кожній частині завдання.
9. Висновки.

Контрольні питання

1. Що розуміють під розв'язанням задачі призначення?
2. Які критерії оптимальності розподілення вузлів ОП застосовують при розв'язанні задачі призначення?
3. Як визначити критичний шлях для задачі без урахування передач? Чому вузол є критичним?
4. Що впливає на вибір стратегії призначення готових до виконання вузлів ОП? Які стратегії найчастіше використовуються і чому?
5. Якими перевагами володіють різні способи підвищення ймовірності знаходження помилок в ОП? Як можна досягти $P_{пом}$?
6. Чому дорівнює максимальна ймовірність знаходження помилок при виконанні задачі за заданий час на певній кількості процесорів? Чи завжди можна досягти цього значення?

Рекомендації до виконання лабораторної роботи

Для виконання п.2 перемалювати часові діаграми виконання ОП, отримані в п.2.1. для випадку виконання ОП за заданий час на мінімальній кількості процесорів.

Вибір копій вузлів для заповнення "пустот" слід виконувати, виходячи із умови $P_{пом} = \Sigma T_{дубл} / T_{max} \rightarrow max$.

Для програмного моделювання ОП з урахуванням копій вузлів слід виконати зміну вхідної матриці суміжності, яка слугує формальною моделлю графа в програмних моделях, що використовуються. Для цього слід додати в матрицю суміжності число додаткових рядків і стовпчиків, що дорівнює числу копій вузлів ОП, правильно розташувавши зв'язки між вузлами.

Для досягнення $P_{пом}=1$ слід продублювати всі вузли ОП, тобто виконати паралельно дві однакові задачі в багатозадачному режимі.

Приклад виконання деяких пунктів роботи

Розглядається граф задачі, наведений на рис.2 лабораторної роботи №1.

Вище було отримано, що для цієї задачі T_{max} = 48 МТ, T_{min} = 23 МТ, тому перше наближення числа процесорів, за допомогою яких можна виконати задачу за мінімальний час – n=3 (найближче більше ціле від значення T_{max} / T_{min}). Стратегія призначення – вибір готового до виконання вузла ОП з максимальним часом виконання. Часова діаграма виконання задачі наведена на рисунку 4.1.

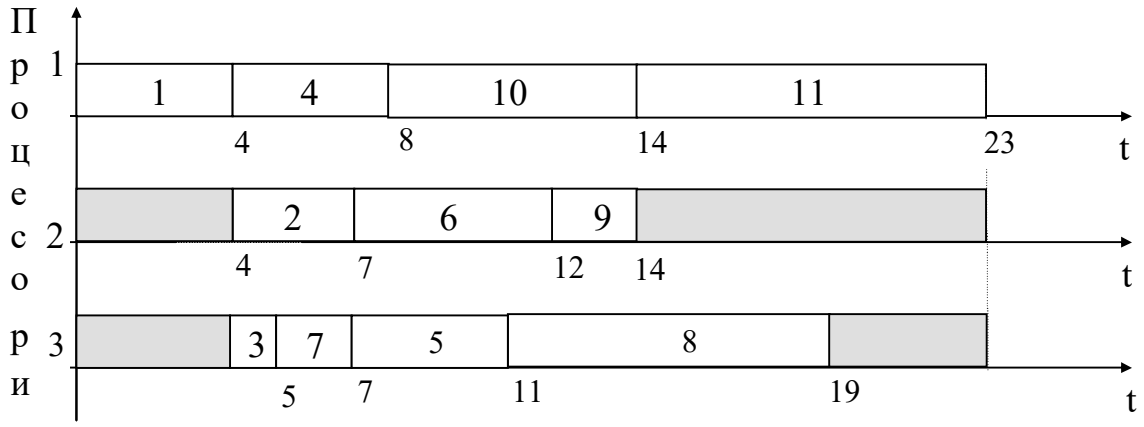


Рисунок 4.1 – Діаграма виконання вузлів задачі

У випадку декількох вільних процесорів призначення готового до виконання вузла здійснюється на процесор з меншим номером). По вертикальній вісі – номера процесорів, по горизонтальній – час, заданий в умовних одиницях (МТ), в інтервалах зайнятості процесорів проставлені номери вузлів задачі. Аналізуючи часову діаграму можна зробити наступні висновки:

1. Мінімальне число процесорів, за допомогою яких задача виконується за T_{min} – 3 процесори, так як перший працює без простоїв, два інших зайняті безперервно.

2. Коефіцієнти завантаження процесорів відповідно дорівнюють:
$К_{зав\ 1} = 1$, $К_{зав\ 2} = 10/23$, $К_{зав\ 3} = 15/23$.
3. Коефіцієнт прискорення (максимально можливий для даної задачі) $К_{приск} = 48/23$.
4. Заштриховані інтервали часу («порожнини») для другого і третього процесорів можна використати для виконання копій вузлів ОП, готових до виконання на початок інтервалу простою процесора. Бажано, щоб копії однакових вузлів виконувались на різних процесорах. Відповідна діаграма представлена на рисунку 4.2.

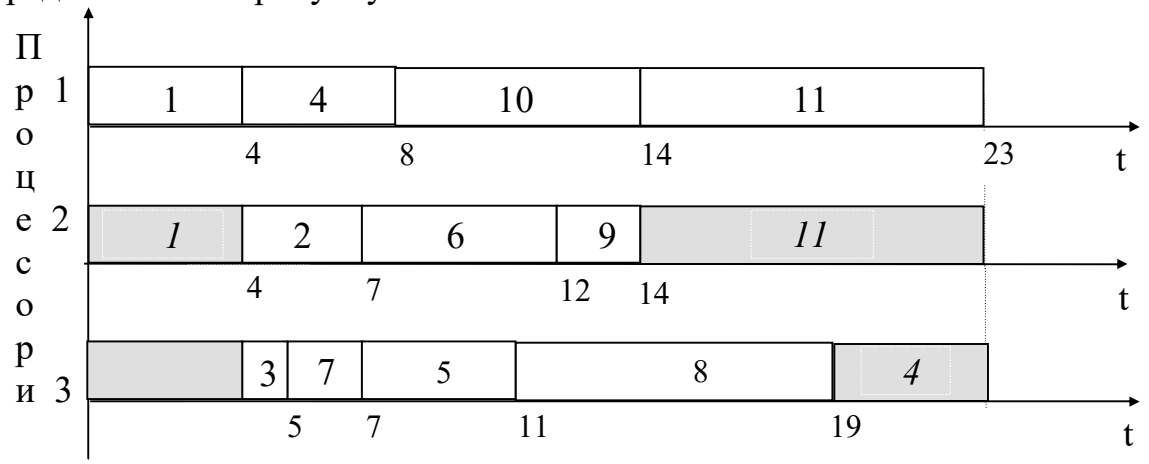


Рисунок 4.2 – Приклад виконання копій вузлів в "порожнинах"

На основі результатів аналізу діаграми визначимо ймовірність знаходження помилки $Р_{пом} = (4+9+4)/48 = 17/48$.

Лабораторна робота №5

Тема: ***Організація багатозадачного режиму виконання обчислювального процесу в багатопроцесорній обчислювальній системі***

Мета: дослідити задачу складання розкладу виконання ОП в багатозадачному режимі в БОС з використанням різних стратегій призначення готових вузлів, з урахуванням пріоритету задачі, що виконується і без врахування пріоритету; вивчити методику вибору характеристик БОС (кількість процесорів, стратегія призначення, пріоритетність задачі) для досягнення заданого часу виконання набору задач.

Хід роботи

Частина 1

(!!!) Дана частина виконується без моделювання на комп'ютері (див. рекомендації щодо виконання даної роботи)

1. За номером по журналу взяти дані для дослідження з таблиці 5.1
2. Для заданого набору задач розрахувати максимальний і мінімальний час його виконання.

Частина 2

3. В однозадачному режимі за допомогою програми LaboratoryWork2 промоделювати окремо виконання кожної задачі з використанням заданої стратегії вибору готових вузлів на заданій кількості процесорів (відповідно до таблиці 5.1).
4. Результати моделювання для кожної задачі звести в таблицю (кількість процесорів, час розв'язання задачі, коефіцієнти завантаження). Визначити сумарний час виконання двох задач в однозадачному режимі.
5. Виконати набір із двох задач з рівними пріоритетами в багатозадачному режимі на тій же кількості процесорів і з тією ж стратегією призначення, що і в п.3. Визначити час виконання набору задач і коефіцієнти завантаження процесорів.
6. Порівняти результати, отримані в п.3 і в п.5, обчислити коефіцієнти прискорення виконання задач і зміну коефіцієнтів завантаження процесорів в різних режимах. Проаналізувати і пояснити отримані результати.
7. Визначити характеристики БОС, що забезпечують досягнення заданого часу виконання набору задач при мінімальних апаратних затратах (min числу процесорів) $T_{зад} = T_{min} + 4$. Параметрами, що варіюються є кількість процесорів, стратегії призначення готових до виконання вузлів. Для виконання даного пункту необхідно
    - проаналізувати різні варіанти зміни характеристик БОС при досягненні $T_{зад}$ також в залежності від пріоритету задач: у задач можуть бути рівні пріоритет або пріоритет одної може бути вище пріоритету другої.
    - дослідити залежність коефіцієнта прискорення від кількості процесорів при різних стратегіях призначення готових вузлів.
    - проаналізувати отримані результати, обрати найкращий спосіб організації ОП.

Звіт має містити

10. Тему, мету лабораторної роботи.
11. Завдання для виконання.
12. Варіанти заданих графів.
13. Розрахунки необхідних часових параметрів, таблиці, діаграми згідно завданню до лабораторної роботи.
14. Висновки.

Контрольні питання

1. Чому дорівнює максимальний і мінімальний час виконання набору задач?
2. Пояснити залежність коефіцієнта прискорення від кількості процесорів в однозадачному режимі і при виконанні набору задач.
3. Як впливає врахування пріоритету задач у наборі на вибір стратегії призначення готових до виконання вузлів ОП?

4. Пояснити зміну коефіцієнтів завантаження процесорів при виконанні набору задач і кожної задачі окремо.
5. Як визначити мінімальні апаратні витрати, що забезпечують досягнення заданого часу виконання набору задач?

Рекомендації до виконання

Так же, як і в однозадачному режимі, готовий до виконання вузол ОП, обраний з врахуванням пріоритету задачі і стратегії призначення, починає виконуватись на вільному процесорі без часових затримок незалежно від того, якій задачі він належить.

В багатозадачному режимі максимальний час виконання набору задач дорівнює сумі часу виконання всіх вузлів всіх задач набору. Мінімальний час виконання набору задач дорівнює максимальному значенню із мінімальних часів виконання кожної задачі.

Слід звернути увагу на те, що в багатозадачному режимі з урахуванням пріоритетів серед готових до виконання вузлів ОП формуються окремі черги на виконання у відповідності з пріоритетами. Першими призначаються на виконання вузли із черги з найвищим пріоритетом, у відповідності з обраною стратегією. Якщо ця черга порожня, то аналізується черга з меншим пріоритетом тощо.

Таблиця 5.1 – Варіанти завдань до лабораторної роботи №5

Варіант	**Графи задач**	**Стратегія**	**Кількість процесорів**
1	f2, f7	З максимальним часом виконання	3
2	f4, f8	З мінімальним часом виконання	4
3	f3, f10	З максимальною кількістю послідовників	5
4	f5, f9	З мінімальним часом виконання	2
5	f2, f7	З максимальним часом виконання	2
6	f4, f8	З максимальною кількістю послідовників	5
7	f3, f10	З максимальною кількістю послідовників	4
8	f5, f9	З максимальним часом виконання	3
9	f2, f7	З мінімальним часом виконання	4
10	f4, f8	З мінімальним часом виконання	5
11	f3, f10	З максимальним часом виконання	2
12	f5, f9	З максимальною кількістю послідовників	3
13	f2, f7	З максимальною кількістю послідовників	5
14	f4, f8	З максимальним часом виконання	3

15	f3, f10	З мінімальним часом виконання	4
16	f5, f9	З максимальним часом виконання	2
17	f2, f7	З мінімальним часом виконання	3
18	f4, f8	З максимальною кількістю послідовників	4
19	f3, f10	З мінімальним часом виконання	5
20	f5, f9	З максимальним часом виконання	4
21	f2, f7	З максимальною кількістю послідовників	3
22	f4, f8	З мінімальним часом виконання	2
23	f3, f10	З максимальним часом виконання	5
24	f5, f9	З максимальною кількістю послідовників	4
25	f2, f7	З максимальним часом виконання	3

Лабораторна робота №6

Тема: ***Дослідження принципів організації обчислювального процесу в багатопроцесорній обчислювальній системі із загальною пам'яттю***

Мета: дослідити способи організації ОП при виконанні набору задач різних типів на БОС з загальною пам'яттю і шинною організацією комутації при визначенні параметрів ОС (кількість процесорів, шин і модулів пам'яті), що дозволяють виконати набір задач конкретного типу за заданий час

Хід роботи

1. За номером по журналу обрати із таблиці 6.1 набори із трьох задач наступних типів:
   - слабкозв'язані задачі, в яких час виконання вузлів задачі набагато більше часу передач між вузлами $t_р >> t_п$;
   - середньозв'язані, в яких $t_р \approx t_п$;
   - сильнозв'язані, в котрих $t_р << t_п$.
2. Побудувати залежності часу розв'язання задач від числа процесорів та числа шин для набору задач кожного типу; знайти найкращий варіант. Виявити параметри, які мають найсуттєвіший виграш.
3. Визначити структуру ОП, що дозволяє виконати набір задач кожного типу за заданий час $T_{зад} = 1{,}33T_{min}$. Встановити, яка стратегія призначення вузлів є найкращою Проаналізувати отримані результати і пояснити їх.

Звіт має містити

1. Тему, мету лабораторної роботи.
2. Завдання для виконання.

3. Варіанти заданих графів.
4. Розрахунки необхідних часових параметрів, таблиці, діаграми згідно завданню до лабораторної роботи.
5. Висновки.

Контрольні питання

1. Задачі яких типів досліджуються в даній роботі?
2. Як визначити критичний шлях і мінімальний час виконання задачі з урахуванням передач на БОС з загальною пам'яттю, яка структура БОС цьому відповідає? Чи завжди цей критичний шлях співпадає з критичним шляхом задачі без врахування передач?
3. Зміна якого параметру (число процесорів, шин чи модулів пам'яті) є найбільш суттєвою для зменшення часу виконання задач різного типу? Як це можна пояснити на основі залежностей, отриманих в п.3 частини 2?
4. Як можна пояснити збільшення в деяких випадках часу виконання набору задач при збільшенні кількості процесорів? На яких типах задач це найбільш наочно? Які значення інших параметрів БОС цьому відповідають?
5. Як визначити значення параметрів БОС (число процесорів, шин і модулів пам'яті), що дозволяють виконати набір задач за заданий час при найменшій кількості кроків моделювання?

Рекомендації до виконання

Слід звернути увагу на наступне:

1. БОС складається із n процесорів, m шин, зв'язуючих кожен процесор із кожним модулем пам'яті (рисунок 6.1).
2. Процесор, виконавши черговий вузол ОП, може «захопити» будь-яку вільну шину і передати всі результати обробки в будь-який один вільний з точки зору процесу запису/зчитування модуль пам'яті (ємність модуля пам'яті вважається необмеженою).
3. Процесор, на котрий призначений для виконання вузол ОП, може «захопити» будь-яку вільну шину, і якщо модуль пам'яті, в якому знаходяться вхідні дані для даного вузла не зайнятий записом/зчитуванням даних іншим процесором, то відбувається їх зчитування.

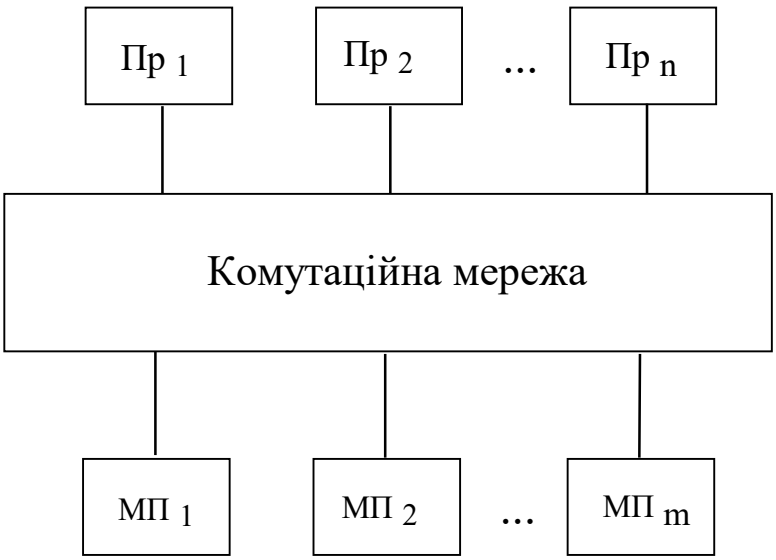


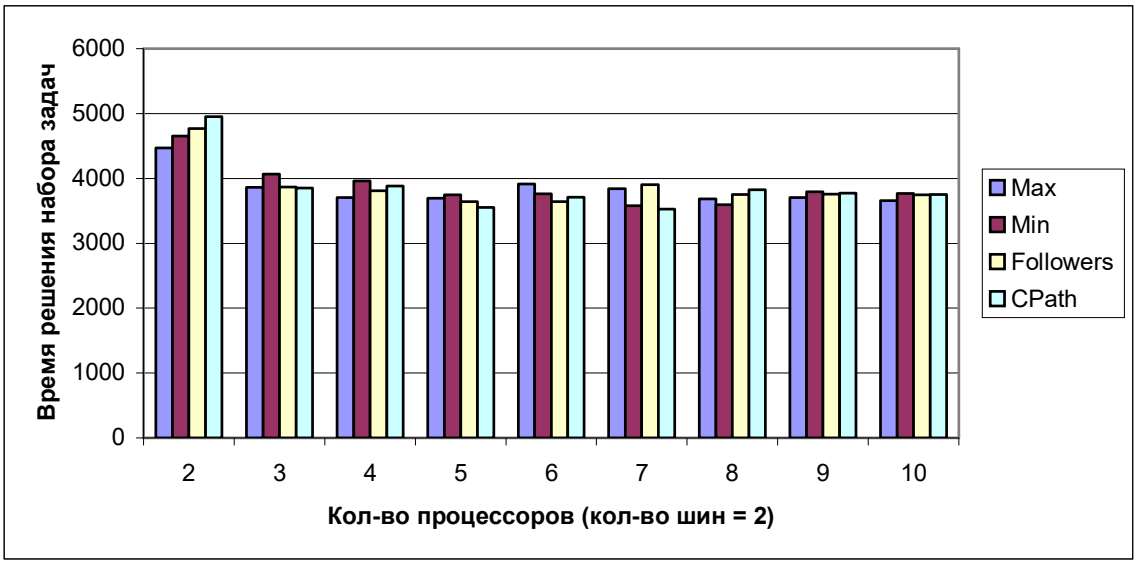
Рисунок 6.1 - БОС із загальною пам’яттю

4. При реалізації стратегії призначення враховуються лише часи виконання вузлів задачі.

Приклади виконання п. 2-3

Сильнозв’язані задачі (T_{min} = 638 МТ):

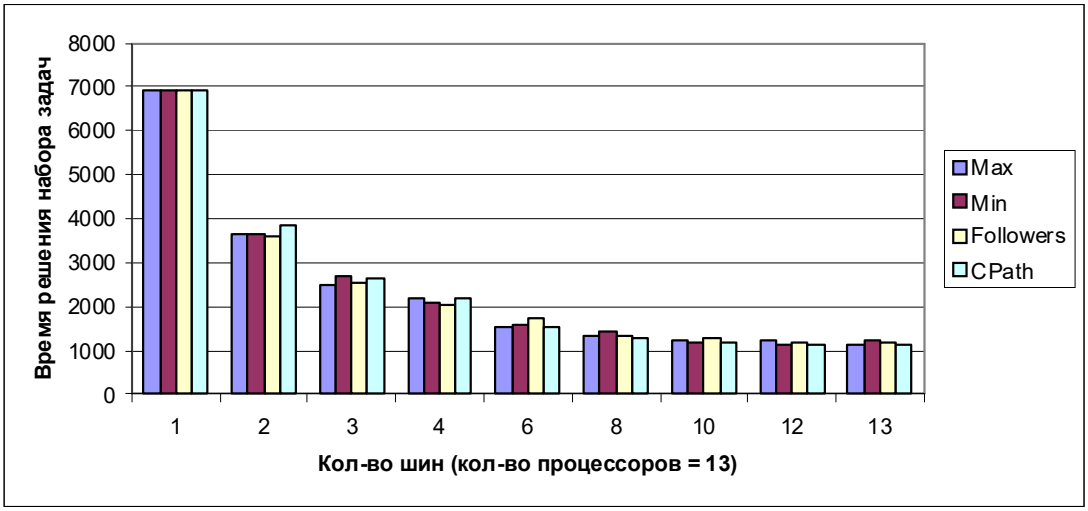
Кількість процесорів (кількість шин = 2), шт.	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
З максимальним часом виконання, МТ	4471	3864	3703	3693	3917	3840	3683	3704	3659
З мінімальним часом виконання, МТ	4655	4065	3961	3746	3764	3580	3593	3796	3768
З максимальною кількістю послідовників, МТ	4769	3865	3808	3641	3642	3905	3751	3755	3749
За приналежністю до КШ, МТ	4952	3851	3885	3551	3710	3528	3826	3775	3752
Оптимальна конфігурація для мін. часу розв’язання	P=16, B=12, T=1115 МТ, стратегія – з max часом.								



Оптимальна конфігурація БОС для розв'язання набору задач за $T_{зад}$=1150:

P=13, B=12, T=1114 МТ, стратегія – з min часом.

Кількість шин (кількість процесорів = 13), шт.	**1**	**2**	**3**	**4**	**6**	**8**	**10**	**12**	**13**
З максимальним часом виконання, МТ	6927	3639	2495	2176	1557	1349	1227	1223	1157
З мінімальним часом виконання, МТ	6946	3637	2705	2083	1563	1444	1163	1144	1255
З максимальною кількістю послідовників, МТ	6916	3615	2554	2061	1721	1312	1292	1165	1163
За приналежністю до КШ, МТ	6928	3830	2660	2195	1519	1256	1175	1158	1137



Таблиця 6.1 – Варіанти завдань до лабораторної роботи №6

Варіант	**Графи задач**
1	f11, f12, f13
2	f21, f22, f23
3	f31, f32, f33

4	f41, f42, f43
5	f51, f52, f53
6	f61, f62, f63
7	f71, f72, f73
8	f81, f82, f83
9	f11, f12, f13
10	f21, f22, f23
11	f31, f32, f33
12	f41, f42, f43
13	f51, f52, f53
14	f61, f62, f63
15	f71, f72, f73
16	f81, f82, f83
17	f11, f12, f13
18	f21, f22, f23
19	f31, f32, f33
20	f41, f42, f43
21	f51, f52, f53
22	f61, f62, f63
23	f71, f72, f73
24	f81, f82, f83
25	f11, f12, f13

Лабораторна робота №7

Тема: ***Дослідження принципів організації обчислювального процесу в багатопроцесорній обчислювальній системі із розподіленою пам'яттю***

Мета: дослідити способи організації ОП в БОС з розподіленою пам'яттю, що дозволяє виконати сильнозв'язану задачу за заданий час; порівняти отримані характеристики з аналогічними для БОС з загальною пам'яттю.

Хід роботи

1. За номером по журналу обрати із таблиці 6.1 набори із трьох задач наступних типів:
   - слабкозв'язані задачі, в яких час виконання вузлів задачі набагато більше часу передач між вузлами $t_р >> t_п$;
   - середньозв'язані, в яких $t_р \approx t_п$;
   - сильнозв'язані, в котрих $t_р << t_п$.
2. За допомогою програми LaboratoryWork4 виконати пункти 2 і 3 з лабораторної роботи №6.

3. Порівняти отримані результати з аналогічними результатами лабораторної роботи №6. Зробити відповідні висновки.

Звіт має містити

6. Тему, мету лабораторної роботи.
7. Завдання для виконання.
8. Варіанти заданих графів.
9. Розрахунки необхідних часових параметрів, таблиці, діаграми згідно завданню до лабораторної роботи.
10. Висновки.

Контрольні питання

1. Задачі яких типів досліджуються в даній роботі?
2. Як визначити критичний шлях і мінімальний час виконання задачі з урахуванням передач на БОС з загальною пам’яттю, яка структура БОС цьому відповідає? Чи завжди цей критичний шлях співпадає з критичним шляхом задачі без врахування передач?
3. Зміна якого параметру (число процесорів, шин чи модулів пам’яті) є найбільш суттєвою для зменшення часу виконання задач різного типу? Як це можна пояснити на основі залежностей, отриманих в п.3 частини 2?
4. Як можна пояснити збільшення в деяких випадках часу виконання набору задач при збільшенні кількості процесорів? На яких типах задач це найбільш наочно? Які значення інших параметрів БОС цьому відповідають?
5. Як визначити значення параметрів БОС (число процесорів, шин і модулів пам’яті), що дозволяють виконати набір задач за заданий час при найменшій кількості кроків моделювання?

Рекомендації до виконання

В БОС із розподіленою пам’яттю (рисунок 6.1) кожен із n процесорів має свою локальну пам’ять, ємністю достатньою для зберігання проміжних результатів обробки будь-якого вузла задачі. Кожен процесор має доступ до пам’яті будь-якого іншого процесора по зчитуванню із неї даних в свою пам’ять по одній із m шин, при цьому $m<n/2$. Одночасно запис в пам’ять чи зчитування із неї відбувається лише одним каналом.

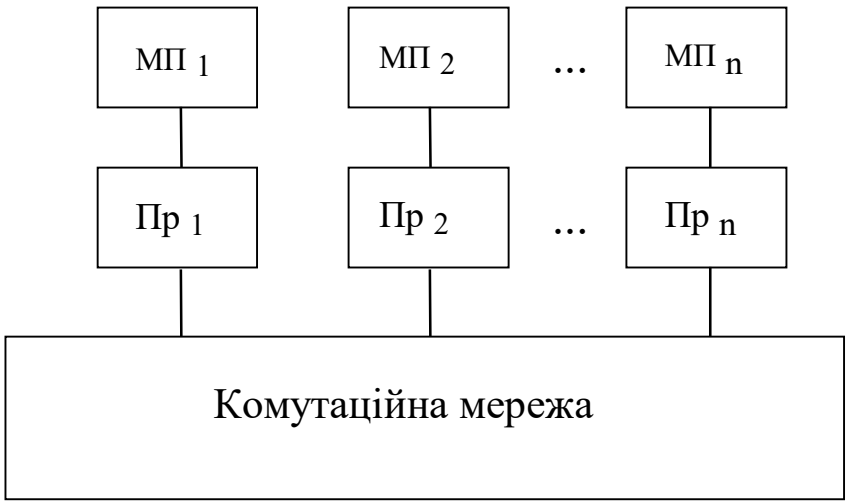


Рисунок 7.1 - БОС із розподіленою пам’яттю

Таблиця 7.1 – Варіанти завдань до лабораторної роботи №7

Варіант	**Графи задач**
1	f11, f12, f13
2	f21, f22, f23
3	f31, f32, f33
4	f41, f42, f43
5	f51, f52, f53
6	f61, f62, f63
7	f71, f72, f73
8	f81, f82, f83
9	f11, f12, f13
10	f21, f22, f23
11	f31, f32, f33
12	f41, f42, f43
13	f51, f52, f53
14	f61, f62, f63
15	f71, f72, f73
16	f81, f82, f83
17	f11, f12, f13
18	f21, f22, f23
19	f31, f32, f33
20	f41, f42, f43
21	f51, f52, f53
22	f61, f62, f63
23	f71, f72, f73
24	f81, f82, f83
25	f11, f12, f13

Лабораторна робота №8

Тема: ***Розрахунок ефективності багатомагістральних систем.***

Мета: розглянути багатопроцесорну обчислювальну систему з декількома модулями пам'яті і декількома магістралями, вивчити метод розрахунку ефективності системи на основі декомпозиції її на одномагістральні системи та застосувати його для розрахунку ефективності дво- та багатомагістральних систем.

Короткі теоретичні відомості

Нині широке розповсюдження отримали обчислювальні системи з мікропроцесорами невеликої продуктивності. При реалізації великої системи із малих модулів виникає проблема їх ефективного зв'язку. Найпростіше рішення – в єдиній магістралі, що розділяється в часі. Але із-за конфліктів при одночасному зверненні до магістралі декількох процесорів істотно зменшується продуктивність кожного процесора і системи в цілому. Із-за обмеженості пропускної спроможності магістралі обмежується число процесорів, і, відповідно, максимальна продуктивність одномагістральних систем.

Представимо обчислювальну систему у вигляді системи масового обслуговування, що складається із *m* магістралей та *p* модулів пам'яті. До *i*-ї магістралі (*i=1,...,m*) підключено N_i процесорів, що мають зв'язок лише з *i-ю* магістраллю. Кожен із процесорів *i-ї* магістралі працює незалежно від інших процесорів системи деякий випадковий час ξi, розподілений за законом

$$F_i(t) = 1 - e^{\lambda_i t}, \qquad (8.1)$$

після чого з деякою ймовірністю q_{ij} звертається до *j-го* модуля пам'яті, використовуючи для обміну *i-му* магістраль. Причому:

$$\sum_{j=1}^{p} q_{ij} = 1.$$

Процесори обслуговуються на магістралі згідно дисципліни «першим прийшов – першим пішов». При зверненні чергового процесора в модуль пам'яті, зайнятий обслуговуванням другої магістралі, він має очікувати завершення обслуговування, займаючи при цьому магістраль на весь час обслуговування. Під час очікування відповіді із пам'яті процесор простоює. В *j-му* модулі пам'яті запит з *i-ї* магістралі обслуговується за випадковий час τ_{ij}, що розподілений за законом

$$H_{ij} = 1 - e^{-\mu_{ij} t}. \qquad (8.2)$$

Отримавши відповідь із пам'яті, процесор знову починає працювати за законом (8.1). Закінчивши обслуговування одного запиту, модуль пам'яті обирає на обслуговування наступний запит згідно циклічної дисципліни опитування магістралей.

Під ефективністю роботи системи розуміється середнє число процесорів, що працюють

$$M = \sum_{i=1}^{m} M_i , \qquad (8.3)$$

де M_i – ефективність *i-ї* магістралі (середнє число працюючих процесорів на *i-й* магістралі).

Задача полягає у визначенні ефективності системи. Причому, в описаній проблемі виникають конфлікти двох видів: конфлікти при одночасному зверненні процесорів однієї магістралі і конфлікти при зверненні процесорів з різних магістралей в один модуль пам'яті.

Для визначеності і наочності розглянемо двомагістральну однорідну систему з двома модулями пам'яті: $m = 2; \mu_{ij} = \mu; \lambda_i = \lambda$; $q_{11}=q_{12}=q_1$; $q_{21}=q_{22}=q_2$. Стан такої системи описується чотиривимірним вектором (n_1, l_1, n_2, l_2), де n_i – число процесорів *i-ї* магістралі, що очікують обслуговування в пам'яті; $l_i=0$ – якщо черговий запит *i-ї* магістралі очікує обслуговування в модулі пам'яті, що зайнятий обслуговування протилежної магістралі; $l_i=j$, якщо черговий запит *i-ї* магістралі обслуговується в *j-му* модулі пам'яті.

У всіх можливих станах, за виключенням $(0,0,0,0)$, $l_1 \neq l_2$ і якщо $l_i \neq 0$, то $n_i \neq 0$.

Для локально зблокованих стохастичних систем можна розглядати кожен вузол обслуговування окремо від інших. Доведено, що для цих систем існує еквівалентна мережа з таким же розподілом кількості вимог у вузлі, що розглядається, як і у вихідній системі. Еквівалентна мережа конструюється із основного вузла обслуговування, що розглядається, і додаткового вузла з інтенсивністю обслуговування $\mu(n)$, яка дорівнює інтенсивності вимог із основного вузла в вихідній мережі при умові, що в ній є n вимог і час обслуговування в основному вузлі дорівнює нулю.

При дослідженні ефективності нас цікавить кількість процесорів, що працюють на кожній магістралі в будь-який момент часу. Коли $N_i \to \infty$, завантаженість *i-ї* магістралі досягає насищення, тобто ймовірність того, що на магістралі є запит, прагне до одиниці. У випадку повної завантаженості обох магістралей стан системи визначається вектором (l_1, l_2), а ефективність кожної магістралі пропорційна інтенсивності обслуговування запитів магістралі в модулях пам'яті.

Тоді інтенсивності обслуговування γ_{i0} запитів *i-ї* магістралі в обох модулях пам'яті:

$$\gamma_{10} = \gamma_{20} = \mu(1 + 2q_1q_2)^{-1} \qquad (8.4)$$

Ефективність одномагістральної системи з N_i процесорами і з одним обслуговуючим приладом, час обслуговування якого розподілений по закону з параметром γ_{i0}, визначається:

$$\mathrm{MN_i} = \frac{1}{\rho_i}\left[1 - \left(\frac{1}{\sum_{k=0}^{N_i}\frac{N_i!}{(N_i-k)!}\rho_i^k}\right)\right], \qquad (8.5)$$

де $\rho_i = \dfrac{\lambda}{\gamma_{i0}}$ - коефіцієнт завантаження *i-ї* магістралі;

λ – інтенсивність завантаження магістралі.

При довільному законі розподілення час обслуговування в пам'яті з кінцевою інтенсивністю γ_{i0} і $\mathrm{N_i} \to \infty$, ефективність системи досягає насищення, що дорівнює $\dfrac{\gamma_{i0}}{\lambda}$. Так як інтенсивність обслуговування γ_{i0} визначалась при великій завантаженості магістралі, тому вираз (8.4) з меншою похибкою визначає ефективність магістралі при великих N_i. Основна похибка в тому, що час обслуговування в пам'яті розраховується, припускаючи постійну наявність запиту на протилежній магістралі, тобто ймовірність відсутності запиту на *i-й* магістралі p^i_0 вважається рівною нулю. При кінцевій N_i ймовірність $p^i_0 \neq 0$ і визначається виразом:

$$\mathrm{p_0^i}(\gamma_{i0}) = \frac{1}{\sum_{k=0}^{N_i}\frac{N_i!}{(N_i-k)!}\cdot\left(\frac{\lambda}{\gamma_{i0}}\right)^k} \qquad (8.6)$$

При відсутності запитів на одній магістралі, запити на іншій магістралі обслуговуються без затримки (з інтенсивністю μ), тобто ефективність магістралі вище, ніж визначається за формулою (8.5).

Середній час обслуговування магістралі з урахуванням ймовірності відсутності запитів на протилежній магістралі:

$$\left.\begin{aligned}\frac{1}{\gamma_{11}} &= \frac{1}{\mu}\left[p_0^2(\gamma_{20}) + \left(1 - p_0^2(\gamma_{20})\right)\cdot(1 + 2q_1q_2)\right]\\ \frac{1}{\gamma_{21}} &= \frac{1}{\mu}\left[p_0^1(\gamma_{10}) + \left(1 - p_0^1(\gamma_{10})\right)\cdot(1 + 2q_1q_2)\right]\end{aligned}\right\} \qquad (8.7)$$

Очевидно, що $\gamma_{i0} < \gamma_{i1} < \mu$. Із виразу (8.6) слідує, що $p^i_0(\gamma)$ зростає зі збільшенням γ, а із (8.7) слідує, що γ_{i1} зростає при збільшенні p^i_0. Тому за допомогою покрокової ітерації отримується зростаюча послідовність $\{\gamma_{ik}\}$, яка має границю $\gamma_{\mathrm{i}} = \underset{k\to\infty}{Lim}\,\gamma_{ik}$. Для визначення граничних значень γ_i з задовільною точністю достатньо п'яти-шести ітерацій.

Цей метод можна перенести на загальний випадок. При не модифікованому методі розрахунок ведеться в два етапи. На першому визначається середній час обслуговування запиту з кожної магістралі $\dfrac{1}{\gamma_{i0}}(\mathrm{i} = 1,\ldots,\mathrm{m})$, а на другому етапі з використанням γ_{i0} визначається ефективність *i-ї* магістралі за формулою (8.5).

Для визначення середнього часу обслуговування запиту з кожної магістралі розглянемо стохастичну схему при умові великого завантаження всіх магістралей, тобто при умові, що на кожній магістралі є безліч запитів на обслуговування в пам'яті. Стан такої системи визначається чергами в кожному модулі пам'яті $\{A_1,...,A_p\}$, де A_k – впорядкована множина індексів магістралей, запити котрих є в черзі до *k-го* модуля пам'яті: $\mathrm{A_k} = (k_1,...,k_c)$; $\mathrm{k_j} \in \{1,...,m\}$, запит k_1-ї магістралі обслуговується $\mathrm{A_i} \cap A_j = \varnothing$; при $i \neq j$. На основі розв'язання системи рівнянь стаціонарного режиму можна визначити ймовірності станів $P\{A_1,...,A_p\}$.

Нехай E – множина натуральних чисел від 1 до m: $E=\{1,2,...,m\}$, D – множина усіх підмножин множини E, B_{ij} – множина станів системи, в яких запит *i-ї* магістралі обслуговується в *j-му* модулі пам'яті; C_{ijk} – множина станів, в яких запит *i-ї* магістралі очікує обслуговування в *j-му* модулі пам'яті, а запит *k-ї* магістралі обслуговується в цьому модулі; Q_{ij} – ймовірність множини B_{ij}; R_{ijk} – ймовірність множини C_{ijk}; S_j – множина індексів j, таких, що $q_{ij} \neq 0$.

Ймовірності множини B_{ij} і C_{ijk} обчислюються:

$$\mathrm{Q_{ij}} = \sum_{B_{ij}} p(A_1,\ldots,A_p)$$

$$R_{ijk} = \sum_{C_{ijk}} p(A_1,\ldots,A_p)$$

А середній час обслуговування запиту *i-ї* магістралі:

$$\frac{1}{\gamma_{i0}} = \sum_{j \in S_i} q_{ij}\,\mu_{ij}^{-1}\left(1 + \frac{\sum_{k=1}^{p} R_{ijk}}{Q_{ij}}\right) \qquad (8.8)$$

Модифікований метод вимагає врахування ймовірності відсутності запитів на деяких магістралях. Для цього необхідно для кожної магістралі розрахувати за формулою (8.8) середній час обслуговування $\omega_{\mathrm{i}}^{1}(a)$ в системі з неповним набором магістралей $\mathrm{a} \in \mathrm{D}$. Нехай $\omega_i(E) = \gamma_{i0}$ і $\omega_i^{-1}(a) = 0$ при $i \notin a$. Тоді на кожному кроці ітерації середній час обслуговування запиту *i-ї* магістралі визначається виразом:

$$\gamma_{\mathrm{il+1}}^{-1} = \sum_{a \in D} \prod_{j \neq a} p_0^j(\gamma_{jl}) \prod_{j \in a} \left(1 - P_0^j(\gamma_{jl})\varpi_i^{-1}(a)\right)$$

де $p_0^j(\gamma_{jk})$ - ймовірність відсутності запитів на *j-й* магістралі (розраховується за формулою (8.6) з параметром γ_{jl}).

Послідовності γ_{il} мають кінцеві границі γ_i, які в модифікованому методі використовуються в якості інтенсивності обслуговування при розрахунку ефективності одномагістральних систем з одним обслуговуючим приладом (вираз (8.5) $\rho_{\mathrm{i}} = \lambda_i \gamma_i^{-1}$).

Потужність множини D дорівнює 2^m*-1*, тому при великому числі магістралей розрахунок усіх $\omega_i(a)$ може виявитися складною задачею.

Враховуючи, що на практиці до кожного модуля пам'яті під'єднується невелика кількість магістралей і на час обслуговування запиту *i-ї* магістралі великий вплив мають лише сусідні відносно загальних модулів пам'яті магістралі, розрахунок $\omega_i(a)$ можна проводити за наближеними формулами:

$$\widetilde{\omega}_i^{-1}(a) = \sum_{j \in S_i} q_{ij} \mu_{ij}^{-1} \left(1 + \frac{\sum_{k \in a} R_{ijk}}{Q_{ij}} \right) \qquad (8.9)$$

В якості приклада багатомагістральних систем розглянемо систему з двома модулями пам'яті і *m* процесорами, кожен із яких зв'язаний з пам'яттю, власною магістраллю і з рівною ймовірністю може звертатися в будь-який модуль пам'яті ($q_i=0.5$). Припустимо, що $\mu_{ij} = \mu$ і $\lambda_i = \lambda$. На основі однорідності процесорів і модулів пам'яті легко показати, що в системі з *m* магістралями при усіх значеннях *i, j, k*:

$$Q_{ij} = \frac{1}{(m+1)}; \; Ri_{jk} = \frac{0.5}{m+1},$$

а $\omega_i(a)$ визначається лише кількістю *C* магістралей у наборі a_c: $\omega_i^{-1}(a_C) = \omega^{-1}(c) = \frac{c+1}{2} \cdot \mu^{-1}$.

Таким чином, вираз (8.9) в даному прикладі дає точний результат:

$$\gamma_{i0}^{-1} = \gamma_0^{-1} = 0{,}5 \cdot (m+1) \cdot \mu^{-1} \qquad (8.10)$$

$$\gamma_{il+1}^{-1} = \gamma_{l+1}^{-1} = \mu^{-1} \sum_{k=0}^{m-1} C_{m-1}^{k} \left[P_0(\gamma_l) \right]^{m-k-1} \cdot \left(1 - \left[P_0(\gamma_l) \right]\right)^k \cdot (0.5k+1) \qquad (8.11)$$

де $p_0(\gamma_1) = \gamma_l (\lambda + \gamma_l)^{-1}$ - ймовірність відсутності запиту на одній із магістралей.

Дослідження продуктивності багатомагістральних систем показали, що збільшення кількості магістралей збільшує продуктивність систем. Тому для підвищення загальної продуктивності системи потрібно будувати багато магістральні системи з конвеєрною структурою і з використанням багатовходових модулів пам'яті.

Хід роботи

1. Провести наближений розрахунок ефективності двомагістральної системи, розрахувавши значення $\frac{1}{\gamma_{11}}$ і $\frac{1}{\gamma_{12}}$ та побудувавши графіки залежностей $MN_i=f_1(N_i)$ і $P_0^i = f_2(N_i)$. Варіант завдання брати за списком по журналу з таблиці 8.1. При розрахунках та побудові залежностей слід врахувати наступне:
   - інтенсивності обслуговування γ_{10}, γ_{20} необхідно розрахувати за формулою (8.4);
   - ефективність одногомагістральної системи MN_i необхідно розраховувати за формулою (8.5);
   - ймовірність P_0^i необхідно розраховувати за формулою (8.6);

- середній час обслуговування магістралі $\frac{1}{\gamma_{11}}$ і $\frac{1}{\gamma_{12}}$ необхідно розрахувати за формулою (8.7).

2. Провести наближений розрахунок ефективності багатомагістральної системи, розрахувавши значення $\frac{1}{\gamma_{i0}}$ та побудувавши графік залежність $M N_i = f_3(N_i)$. Варіант завдання брати за списком по журналу з таблиці 8.1. При розрахунках та побудові залежності слід врахувати наступне:
   - середній час обслуговування магістралі $\frac{1}{\gamma_{i0}}$ необхідно розрахувати за формулою (8.10);
   - ефективність одногомагістральної системи $M N_i$ необхідно розраховувати за формулою (8.5).

Таблиця 8.1 – Варіанти завдань до лабораторної роботи №8

№ вар.	q1	q2	μ	λ1	λ2	m
1	0,1	0,9		0,751	0,459	4
2	0,2	0,8		0,385	0,735	5
3	0,3	0,7		0,458	0,655	6
4	0,4	0,6		0,455	0,743	7
5	0,5	0,5		0,245	0,642	8
6	0,6	0,4		0,355	0,548	9
7	0,7	0,3		0,852	0,321	10
8	0,8	0,2		0,345	0,625	11
9	0,9	0,1		0,925	0,125	12
10	0,1	0,9		0,547	0,225	13
11	0,2	0,8		0,571	0,915	14
12	0,3	0,7		0,674	0,115	15
13	0,4	0,6	0,55	0,274	0,751	16
14	0,5	0,5		0,459	0,751	17
15	0,6	0,4		0,735	0,385	18
16	0,7	0,3		0,655	0,458	19
17	0,8	0,2		0,743	0,455	20
18	0,9	0,1		0,642	0,245	21
19	0,1	0,9		0,548	0,355	22
20	0,2	0,8		0,321	0,852	23
21	0,3	0,7		0,625	0,345	24
22	0,4	0,6		0,125	0,925	25
23	0,5	0,5		0,225	0,547	26
24	0,6	0,4		0,915	0,571	27
25	0,7	0,3		0,115	0,674	28

Звіт має містити

1. Тему, мету лабораторної роботи.
2. Завдання для виконання.
3. Розрахунки та графіки залежностей за власним варіантом
4. Висновки.

Література

1. Головкин Б.А. Параллельные вычислительные системы. — М.: Наука, 1980. — 518 с.
2. Корнеев В.В. Параллельные вычислительные системы. - М.: “Нолидж”, 1999. - 320 с.
3. Ларионов А.М. и др. Вычислительные комплексы, системы и сети / А.М. Ларионов, С.А. Майоров, Г.И. Новиков: Учебник для вузов. - Л.: Энергоатомиздат. Ленингр. отд-ние, 1987. - 288 с., ил.
4. Литвинский И.Е. Обеспечение безотказности персональных ЭВМ. – М.: Радио и связь, 1993. – 208 с.
5. Немнюгин С.А., Стесик О.Л. Параллельное программирование для многопроцессорных вычислительных систем. – СПб.: БХВ-Петербург, 2002. – 400 с.: ил.
6. Основы теории вычислительных систем: Учебное пособие / Под ред. С.А. Майорова. - М.: Высшая школа, 1981. – 324 с.
7. Погребинский С.Б., Стрельников В.П. Проектирование и надежность многопроцессорных ЭВМ. – М.: Радио и связь, 1988. – 168 с.
8. Пятибратов А.П. и др. Вычислительные системы, сети и телекоммуникации: Учебник. - 2-е изд., перераб. и доп. / А.П. Пятибратов, Л.П. Гудыко, А.А. Кириченко: Под ред. А.П. Пятибратова. - М.: Финансы и статистика, 2001. - 512 с., ил.
9. Смирнов А.Д. Архитектура вычислительных систем: Учеб. пособие для вузов. — М.: Наука. Физматлит, 1990. — 320 с.
10. Столлингс В. Структурная организация и архитектура компьютерных систем, 5-ое изд.: Пер. с англ. – М.: Издательский дом "Вильямс", 2002.– 896 с.: ил.
11. Шпаковский Г.И. Организация параллельных ЭВМ и суперскалярных процессоров: Учебн. пособие. – Мн.: Белгосуниверситет, 1996. – 296 с.
12. Каган Б.М. Электронные вычислительные машины и системы : Учебн. пособие для вузов. - М.: Энергоатомиздат, 1985. - 552 с.
13. Перспективы развития вычислительной техники: В 11 кн.: Справ. пособие / Под ред. Ю.М.Смирнова. Кн.4. Многопроцессорные ЭВМ и методы их проектирования / Б.А.Бабаян, А.В.Бочаров, В.С.Волин и др. - М.: Высшая школа, 1990. - 143 с.
14. Шпаковский Г.И. Архитектура параллельных ЭВМ: Учеб. пособие для вузов. — Мн.: Университетское, 1989. — 192 с.
15. Ульянов М.В. Архитектуры процессоров: Учеб. пособие. – М.: МГАПИ, 2002. – 68 с.
16. http://www.csa.ru/analitik/distant/q_start.html “Высокопроизводительные алгоритмы”, Институт высокопроизводительных вычислений и баз данных

17. http://support.vologda.ru/Book/ARCHITECTURE/Svk/contents.htm Электронное пособие «Современные высокопроизводительные компьютеры» В. Шнитман
18. http://www.hpc.nw.ru/COURSES/HPC/index.html Электронное пособие «Программирование для высокопроизводительных ЭВМ» А.В.Комолкин, С.А.Немнюгин

Методичні вказівки

до виконання лабораторних робіт

з дисципліни:

"Комп'ютерні системи "

для студентів спеціальності 6.050102 “Комп'ютерні системи та мережі” всіх форм навчання

УКЛАДАЧІ: Купін Андрій Іванович

Кузнєцов Денис Іванович

Сенько Антон Олександрович

Рябчина Любов Сергіївна

Реєстраційний №

Підписано до друку _____ 2019 р.
Формат A5
Обсяг 46 стор.
Тираж _____ прим.

Видавничий центр КНУ,

вул. В.Матусевича, 11,

м. Кривий Ріг