<u>Тема №9:</u> Мови паралельного програмування

Питання:
1. Загальні зауваження
2. Класифікація мов і систем паралельного програмування
3. Особливості організації паралельної програми
4. Технології паралельного програмування Message Passing Interface (MPI)
5. Операції обміну повідомленнями

1. Загальні зауваження

Є різні підходи до організації паралельного програмування. Одні широко використовуються на практиці, інші хороші своєю ідеєю, треті локанічні, тощо. Необхідно звернути увагу на то, що процес розробки паралельної програми відрізняються від процесу розробки послідовної програми. Тут необхідно попередньо проаналізувати і виявити ті фрагменти програми, які займають найбільше процесорного часу. Саме розпаралелення цих фрагментів дозволяє підвищити швидкість виконання програми. Розпаралелення зв'язане з виявленням підзадач, розв'язання яких можна доручити різним процесорам. Вимагається організація взаємодії між такими підзадачами, що можна зробити шляхом обміну повідомленнями-посилками (містять вхідні дані, проміжні і кінцеві результати роботи). Спеціальні прийоми вимагаються і при відлагодженні паралельної програми.

На ефективність розробки програм впливає і наявність відповідного програмного інструментарію. Розробнику паралельних програм необхідні *засоби аналізу і виявлення паралелізму, транслятори, операційні системи (ОС),* які забезпечують надійну роботу багатопроцесорних конфігурацій. Необхідні також *засоби відлагодження і профілювання* (побудова каркасу програми, прив'язаного до часових параметрів, що дозволяє оцінити продуктивність програми та окремих її частин). Середовищем виконання паралельних програм є ОС, які підтримують *мультипроцесування, багатозадачність і багатопотоковість*, наприклад, *UNIX* чи *Microsoft Windows NT*.

При розпаралеленні, яке ведеться на рівні алгоритмів необхідно: *виміряти часові параметри, визначити необхідну кількість пересилань* та *необхідні об'єми пам'яті.* Засоби відлагодження повинні забезпечити можливість безтупикової перевірки програм.

2. Класифікація мов і систем паралельного програмування

Вибір технології паралельного програмування є складною задачею, якщо прийняти до уваги, що тільки мов і систем розробки паралельних програм є понад 100 і їх кількість невпинно зростає. Відповідно до цього є різні підходи до їх узагальнення і класифікації. В табл.8.1 наведена класифікація, яка частково орієнтована на архітектури паралельних систем.

Таблиця 8.1. – Паралельні мови і системи програмування

Для систем з спільною пам'яттю	Для систем з розподіленою пам'яттю	Паралельні об'єктно-орієнтовані	Паралельні декларативні
a	b	C	d
OpenMP	PVM	HPC++	Parlog
Linda	MPI	MPL	Multilisp
Orca	HPF	CA	Sisal
Java	Cilk	Distributed Java	Concurrent Prolog
Pthreads	C	Charm++	GHC
Opus	ZPL	Concurrent Aggregates	Strand
SDL	Occam	Argus	Tempo
Ease	Concurrent C	Presto	

SHMEM	Ada	Nexus	
	FORTRAN M	uC++	
	CSP	sC++	
	NESL	pC++	

Деякі мови параллельного програмування машинно-залежні і створені для конкретних систем. Використовуються і універсальні мови програмування високого рівня, такі як сучасні версії мови FORTRAN і мова С.

FORTRAN D – розширення мови FORTRAN 77/90, за допомогою якої можна визначити машинно-незалежним чином як треба розподілити дані між процесорами. Мова дозволяє використовувати спільний простір імен. Транслятори можуть генерувати код як для SIMD, так і для MIMD-машин. FORTRAN D є попередником мови High Perfomance FORTRAN (HPF).

В мовах *попереджуючих обчислень* паралелізм реалізований за допомогою паралельного виконання обчислень ще до того, як їх результат потрібний для продовження виконання програми. Прикладами таких мов є: *MultiLisp* – паралельна версія мови Lisp, *QLisp, Mul-T.*

Мови програмування в *моделі паралелізму даних: C, APL, UC, HPL*. *PARLOG* є паралельною версією мови *Prolog*. *Erland* – паралельна мова для застосувань в режимі реального часу. *Maisie* - мова, яка базується на С. *Scheme* – один з діалектів *Lisp*. *Cilk* – алгоритмічна багатопотокова мова.

Середовища розробки паралельних програм: aCe – середовище розробки і виконання програм в рамках моделі паралельних даних. *ADAPTOR* – система основана на мові *HPF*. *Arjuna* – об'єктно-орієнтована система програмування розподілених додатків. *CODE* – візуальна система паралельного програмування.

При розробці паралельних програм *в рамках моделі паралелізму задач* найчастіше використовуються спеціалізовані бібліотеки і системи паралельного програмування *PVM* і *MPI*. *PVM* існує для різних платформ, є реалізації для мов *Java* і *Python*, і включає різні засоби створення паралельних програм.

MPI – специфікація, перший варіант якої був розроблений комітетом *MPIF* (Message Passing Interface Forum) в 1994 р., яка описує основні особливості і синтаксис прикладного програмного інтерфейсу паралельних програм. *Хоча MPI не властиві певні хороші властивості PVM, але ця специфікація базується на погоджених стандартах і найчастіше використовується для створення паралельних програм.* Вона використовується з мовами *C, C++, FORTRAN*. Є кілька , вільно розповсюджуваних, реалізацій MPI для різних платформ: *MPICH, LAM, CHIMP, NT-MPICH, WMPI, MacMPI* [8].

3. Особливості організації паралельної програми

Розглянемо основні нюанси організації паралельної роботи при використанні різних мов паралельного програмування.

Технологія програмування OpenMP

Стандарт розроблений для мов FORTRAN (77, 90,95), C, C++і підтримується практично всіма виробниками великих обчислювальних систем. Реалізація стандарту доступна як на *UNIX*-платформах, так і в середовищі *Windows NT*.

В стандарті за основу береться послідовна програма, а для створення її паралельної версії користувачу надається набір директив, процедур і навколишніх змінних.

Текст програми розбивається на послідовні і паралельні області (див. рис.8.1). В початковий момент часу породжується нитка-майстер (основна нитка), яка починає виконувати програму з стартової точки. *Основна нитка* і тільки вона *виконує всі послідовні області* програми. Для підтримки паралелізму використовується схема *FORK/JOIN*. При вході в паралельну нитку-майстер створюються додаткові нитки (виконується операція FORK). Після створення кожна нитка отримує свій унікальний номер, причому нитка-майстер завжди має номер 0. Всі створені нитки використовують однаковий код, який відповідає паралельній області. При виході з паралельної області основна нитка дочікується завершення решту ниток, і подальше виконання програми продовжує тільки вона.

В паралельній області всі змінні програми діляться на два класи: загальні *(SHARED)* і локальні *(PRIVATE)*. Загальні змінні завжди існують тільки в одному екземплярі для всієї програми і доступні всім ниткам під однаковим іменем. Оголошення локальних змінних викликає створення свого екземпляра кожної змінної для кожної нитки.

Система програмування DVM

Система об'єднує моделі паралелізму за даними та керуванню, і складається з 5-ти основних компонентів: *компілятори з мов FORTRAN – DVM і C – DVM, системи підтримки виконання паралельних програм, відлагоджувача паралельних програм, аналізатора продуктивності, передбачувача продуктивності*. Загальна схема відображення програми і взаємозв'язок основних понять системи наведена на рис.8.2.

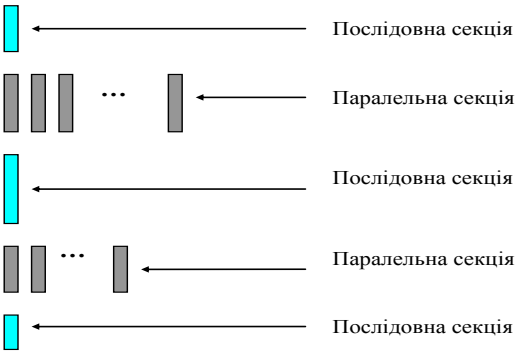

Рис.8.1. OpenMP: процес виконання програми

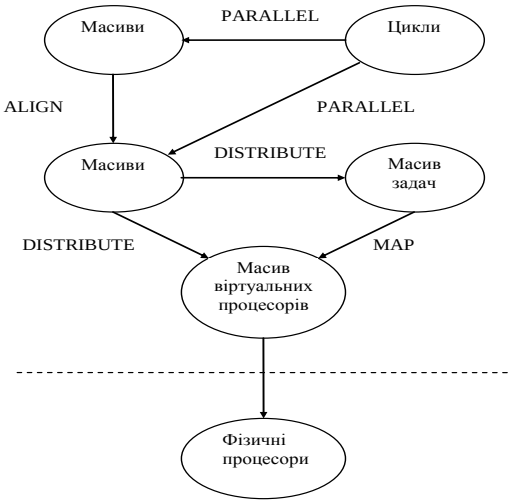

Рис.8.2. Відображення послідовної програми в системі DVM

Відображення віртуальних процесорів на фізичні здійснюється засобами операційної системи.

Для *розподілення масивів* використовується директива DISTRIBUTE (має описовий статус).

Для організації погодженого відображення деяких масивів використовується *механізм вирівнювання* одного масиву відносно іншого за допомогою директиви ALING.

Директива *паралельного виконання циклів* PARALLEL вбудована в мову Fortran у виді спецкоментара.

Директива MAP вбудована в мову Fortran і специфікує то, що *n*-а задача деякого вектора *T* буде виконуватися секцією віртуальних процесорів $P(i_1:i_2, \ldots, j_1:j_2)$. Всі відображені на цю задачу масиви (директива DISTRIBUTE) і обчислення (директива PARALLEL) будуть автоматично відображені на цю ж секцію віртуальних процесорів.

В *DVM* передбачені дві форми відображення блоків програми на задачі. Статична форма є безпосереднім аналогом паралельних секцій в інших мовах, зокрема, директиви SECTIONS в мові *OpenMP*. В динамічній формі ітерація циклу відображається на задачу.

Спільними даними в *DVM* є дані, що обчислюються на одних процесорах, а використовуються іншими.

DVM-програма може виконуватися на довільній кількості процесорів, починаючи з одного. Це є наслідком того, що директиви паралелізму в *DVM* не залежать ні від кількості процесорів, ні від конкретних номерів процесорів. Єдиним винятком є рівень задач. За вимогою директиви MAP користувач повинен явно описати масив віртуальних процесорів. При цьому кількість процесорів, що описані в програмі не повинен перевищити кількості процесорів, що задаються при запуску програми.

4. Технології паралельного програмування Message Passing Interface (MPI)

MPI - бібліотека функцій, яка забезпечує взаємодію паралельних процесів за допомогою механізму передачі повідомлень і не має ніяких засобів для розподілення процесів по обчислювальних вузлах і для запуску їх на виконання. MPI не містить механізмів динамічного створення і знищення процесів під час виконання програми.

Для *ідентифікації* наборів процесів вводиться поняття *групи* і *комунікатора.*

Процеси об'єднуються в *групи*, можуть бути вкладені групи. Усередині групи всі процеси понумеровані. З кожною групою асоційований свій комунікатор. Тому при здійсненні пересилок необхідно вказати ідентифікатор групи, усередині якої проводиться це пересилка.

Процедури MPI:

- ініціалізації та закриття MPI –процесів;
- реалізації комутаційних операцій типу "точка-точка";
- реалізації колективних операцій;
- для роботи з групами процесів і комунікаторами;
- для роботи з структурами даних;
- формування топології процесів.

До базових функцій MPI відносяться:

- ініціалізація MPI;
- завершення MPI;
- визначення кількості процесів в області зв'язку;
- визначення номеру процесу, який виконується;
- передача повідомлень;
- приймання повідомлень;
- функції відліку часу.

Кожна MPI – функція характеризується *способом виконання*.

1. Локальна функція – виконується всередині процесу, що її викликав. Її завершення не вимагає комунікацій.

2. Нелокальна функція – для її завершення необхідно виконати MPI – процедуру іншим процесом.

3. Глобальна функція – процедуру повинні виконати всі процеси групи. Невиконання цієї умови може привести до "зависання" задачі.

4. Блокуюча функція – повернення керування з процедури гарантує можливість повторного використання параметрів, які приймали участь у виклику. Ніякої змін в стан процесу, що викликав блокуючий запит до виходу з процедури не може відбуватися.

5. Неблокуюча функція – повернення з процедури відбувається негайно, без очікування завершення операції. Завершення неблокуючих операцій здійснюється спеціальними функціями.

5. Операції обміну повідомленнями

Розглянемо: *режими обміну, обмін типу "точка-точка", колективний обмін, способи реалізації моделі передачі повідомлень*.

Режими обміну:

В загальному випадку є чотири режими обміну: *асинхронний (стандартний), синхронний, з буферизацією, по "готовності"*.

Обмін типу "точка-точка" – найпростіша форма обміну повідомленнями, в якій приймають участь тільки два процеси: джерело і адресат. Є кілька різновидностей двохточкового обміну:

- *синхронний* обмін – супроводжується повідомленням про завершення прийому повідомлення;
- *асинхронний* обмін – таким повідомленням не супроводжується;
- *блокуючі прийом/передача* – призупиняють виконання процесу на час приймання повідомлення. Організація блокуючого обміну повідомленнями наведена на рис.8.3;
- *неблокуючі прийом/передача* - виконання процесу продовжується в фоновому режимі, а програма в потрібний момент може запитати підтвердження завершення приймання повідомлення. Організація неблокуючого обміну повідомленнями наведена на рис.8.4.

Неблокуючий обмін вимагає акуратності при виконанні функцій прийому. Оскільки неблокуючий прийом завершується негайно, для системи неважливо, чи прибуло повідомлення до місця призначення чи ні. Переконатися про це можна за допомогою функції перевірки отримання повідомлення. Звичайно виклик таких функцій розміщується в циклі, який повторюється до тих пір, доки функція перевірки не поверне значення "істина" (перевірка отримання пройшла успішно). Після цього можна викликати функцію прийому повідомлення з буферу повідомлень.

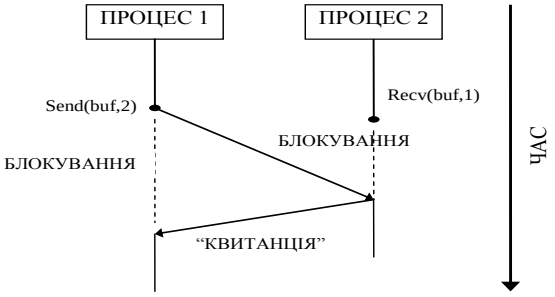

Рис.8.3. Блокуючий обмін повідомленнями

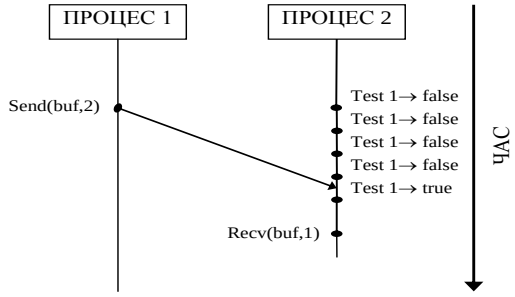

Рис.8.4. Неблокуючий обмін повідомленнями

Колективний обмін . В операціях використовуються не два а більше процесів. Різновидностями обміну є:

- *широкосмугова передача* – передача виконується від одного процесу до всіх;
- *обмін з бар'єром* – форма синхронізації роботи процесів, коли обмін повідомленнями проходить тільки після того, як до певної процедури звернулась певна кількість процесів;
- *операції приведення* – вхідними є дані кількох процесів, а результат – одне значення, яке стає доступним всі процесам, які приймали участь в обміні.

Важливою властивістю системи передачі повідомлень є гарантія збереження порядку прийому повідомлень (при відправленні одним процесом іншому кількох повідомлень вони повинні бути прийняті в тій самій послідовності в якій були відправлені). Більшість реалізацій моделі передачі повідомлень забезпечують цю властивість, але не у всіх режимах обміну.

Способи реалізації моделі передачі повідомлень:

- створення спеціалізованої мови паралельного програмування. Приклад – мова Occam;
- розширення звичайної послідовної мови шляхом включення в неї засобів обміну повідомленнями. Приклад – мова CC++, FORTRAN M;
- використання спеціалізованих бібліотек в програмах, що написані на звичайних мовах послідовного програмування.