УДК 004.891

к.т.н., доцент **Огнєвий О.В**. (ХмНУ)
Присяжнюк В.В. (ХмНУ)
Коваль Б.А. (ХмНУ)

АНАЛІЗ АРХІТЕКТУРНИХ ОСОБЛИВОСТЕЙ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ З ПРОГРАМОВАНОЮ СТРУКТУРОЮ

к.т.н., доцент Огнєвий О.В., Присяжнюк В.В.

Стаття присвячена побудові мультиархітектурних кластерних та мультикластерних обчислювальних систем .

В роботі проведено аналіз обчислювальних систем з програмованною структурою. Дослідження показали, що сучасні розподілені ОС є мультіархітектурнимі. Залежно від рівня розгляду їх функціональних структур, вони можуть виглядати і як MISD, і як SIMD, і як MIMD системи.

Основна функціонально-структурна одиниця обчислювальних ресурсів в системах розглянутого класу - це елементарна машина (ЕМ), яка є композицією з обчислювального модуля і системного пристрою.

Встановлено, що ОС з програмованою структурою відносяться до масштабуємихся засобів обробки інформації і допускають формування конфігурацій з масовим паралелізмом (Scalable Massively Parallel Architecture Computing Systems).

Взаємодія між ЕОМ здійснюється за допомогою програмного налаштування мережі зв'язку. Структура ОС описується графом, множина вершин якого - кількість елементарних машини (або системних пристроїв, або локальних комутаторів), а множина ребер - лінії міжмашинного зв'язку.

До структур сучасних ОС пред'являється ряд вимог: простота вкладення паралельного алгоритму, зручність адресації елементарних машин, здійсненність принципу близькодії і мінімуму затримок при міжмашинних передачах інформації , масштабованість і великомасштабні структури, комутація, живучість, технологічність . Найбільш повно перерахованим вище вимогам задовольняють однорідні структури (тобто описувані однорідними графами). Такі структури є перспективними для формування масштабованих і великомасштабних обчислюваних систем (зокрема, ОС з програмованою структурою).

Для оцінки продуктивності каналів зв'язку між ЕМ системи використано показники пропускна здатність і латентність, щодозволяють здійснити досить повний аналіз структурних можливостей ОС.

В роботі наведено приклад побудови обчислювального кластера, укомплектованого N обчислювальними вузлами, які об'єднані мережею зв'язку стандарту Gigabit Ethernet.

Ключові слова: обчислювальні системи, елементарні машини, системи обробки данних, однорідні структури, комунікаційні середовища, кластер та мультикластер, алгоритми вкладення.

Вступ. Зростаюча потреба у вирішенні складних завдань науки і техніки привела до створення розподілених обчислювальних систем (ОС) [2]. В архітектурному плані розподілена ОС це група взаємодіючих елементарних машин (ЕМ), оснащених засобами комунікацій і зовнішніми пристроями. Всі основні ресурси розподілених ОС (арифметико логічні пристрої, пам'ять, засоби управління і комунікацій) являються логічно і технічно розосередженими. Число ЕМ в розподілених ОС допускає варіювання від декількох одиниць до сотень тисяч (на приклад, в МВС-15000ВМ число процесорів дорівнює 1148, в IBM Roadrunner - 122400, а в системах IBM BlueGene другого покоління до 884736).

Постановка задачі. Сучасні розподілені ОС є мультіархітектурнимі. Залежно від рівня розгляду їх функціональних структур, вони можуть виглядати і як MISD, і як SIMD, і як MIMD системи. Для таких систем характерні ієрархічна організація і різні пропускні спроможності каналів зв'язку між їхніми ресурсами (обчислювальними вузлами, ЕМ, процесорами і їх ядрами).

Час виконання паралельних програм на розподілених ОС істотно залежить від того

наскільки вони ефективно вкладені в систему. Під ефективним вкладенням розуміється такий розподіл гілок паралельної програми між ЕМ системи, при якому досягаються мінімуми накладних витрат на міжмашинний обмін інформацією і дисбаланс завантаження ЕМ.

При організації ефективного функціонування розподілених ЗС попереду стоїть завдання розробки моделей і алгоритмів вкладення па паралельно програм, які враховують архітектурні особливості современ них систем.

Виклад основного матеріалу дослідження Обчислювальні системи з програмованою структурою - це розподілені засоби обробки інформації з архітектурою MIMD (Multiple Instruction Multiple Data)- системи із множинним потоком команд і множинним потоком даних. До цього класу належать як векторні суперЕОМ, так і всі багатопроцесорні системи обробки данних (СОД). Загальна структурна схема таких систем показана на рис.1. Ця архітектура включає всі рівні паралелізму, від конвеєра операцій до незалежних операцій і команд. Вживаючи термін MIMD треба мати на увазі не тільки роботу багатьох процесорів, але і безліч обчислювальних процесів, що виконуються одночасно в ОС.

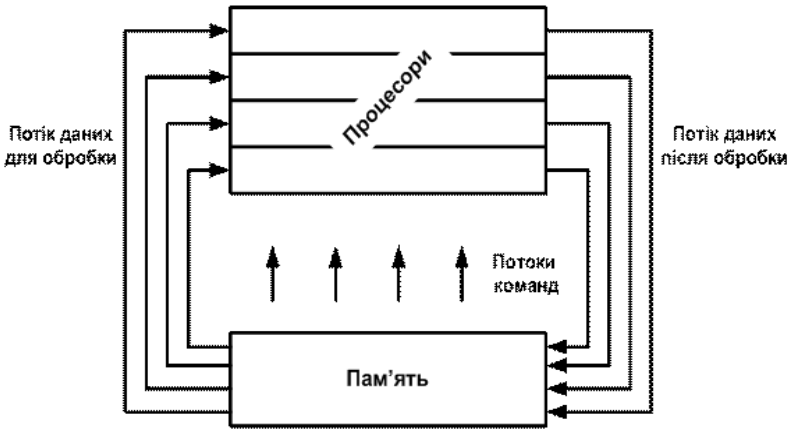


Рис.1. Загальна структура системи класу MIMD

Особливість систем в тому, що в них закладена можливість програмного переналаштування архітектури MIMD в архітектури MISD або SIMD. Основна функціонально-структурна одиниця обчислювальних ресурсів в системах розглянутого класу - це елементарна машина, яка є композицією з обчислювального модуля і системного пристрою.

Обчислювальний модуль (ОМ) служить як для обробки та зберігання інформації, так і для виконання функцій з управління системою в цілому.

Системний пристрій (СП) - це та апаратурна частина ЕМ, яка призначається тільки для забезпечення взаємодії даного ЕМ з найближчими сусідніми машинами (точніше, з системними пристроями, з якими є безпосередній зв'язок).

Допускається конфігурація ОС з довільним числом ЕМ. Отже, ОС з програмованою структурою відносяться до засобів обробки інформації що допускають формування конфігурацій з масовим паралелізмом (Scalable Massively Parallel Architecture Computing Systems).

Взаємодія між ЕМ здійснюється через програмно налаштовану мережу зв'язку. Структура ОС описується графом $G=(C,E)$, множина вершин C якого - кількість елементарних машини (або системних пристроїв, або локальних комутаторів), а множина ребер E - лінії міжмашинного зв'язку.

До структур сучасних ОС пред'являється ряд вимог [15].

1) Простота вкладення паралельного алгоритму розв'язання складної задачі в структуру ОС. Структура ОС повинна бути адекватна досить широкому класу вирішуваних

задач; настройка проблемно-орієнтованих віртуальних змін не повинна бути пов'язана зі значними накладними витратами.

2) Зручність адресації елементарних машин і "перенесення" підсистем в межах обчислювальної системи. Обчислювальна система повинна надавати можливість користувачам створювати паралельні програми з віртуальними адресами ЕМ. Отже, структура ОС повинна дозволяти реалізувати найпростіший "механізм" перетворення віртуальних адрес ЕМ в реальні (фізичні) адреси машин системи. Необхідність організації одночасного вирішення декількох задач на ОС (тобто необхідність поділу простору елементарних машин між задачами) обґрунтовує вимога простоти переміщення підсистем в межах системи (при збереженні їх топологічних властивостей).

3) Здійсненність принципу близькодії і мінімуму затримок при міжмашинних передачах інформації в ОС. Принцип близькодії перевизначає реалізацію обмінів інформацією між "віддаленими" один від одного ЕМ через проміжні машини системи. Отже, в умовах обмеженості числа зв'язків у кожній ЕМ структура повинна забезпечувати мінімум затримок при "транзитних" передачах інформації.

4) Масштабованість і великомасштабні структури ОС. Для формування конфігурацій ОС із заданою ефективністю потрібно, щоб структура була здатна до нарощування і скорочення числа вершин (машин). Зміна числа ЕМ в ОС не повинна призводити до корінних перекомутацій між машинами і (або) до необхідності зміни числа зв'язків для будь-яких ЕМ.

Для досягнення високої продуктивності ОС при існуючих можливостях мікропроцесорної техніки потрібно число ЕМ близько $10-10^6$. Для підтримки великомасштабних (масового паралелізму) необхідно, щоб структура ОС була здатна ефективно здійснювати міжмашинний обмін інформацією в умовах неможливості реалізації зв'язків повного графу (наприклад, через обмеженість числа виводів в корпусах ВІС).

5) Комутація структури ОС. Обчислювальна система повинна бути пристосована до реалізації групових між машинних обмінів інформацією. Отже, структура ОС повинна мати здатність здійснювати вказану кількість одночасних непересічних взаємодій між елементарними машинами.

6) Живучість структури ОС. Важливою вимогою до ОС в цілому являється забезпечення працездатності при відмові її компонентів або навіть підсистем. Основою функціональної цілісності ОС, як колективу елементарних машин, є живучість структури. Під останньою розуміють здатність структури ОС забезпечити зв'язність необхідного числа робочих ЕМ в системі при ненадійних лініях міжмашинних зв'язків.

7) Технологічність структур ОС. Структура мережі міжмашинних зв'язків ОС не повинна пред'являти особливих вимог до елементної бази, до технології виготовлення мікропроцесорних ВІС. Системи повинні бути сприйнятливі до масової технології, їх "обчислювальне ядро" має формуватися з масових мікропроцесорних ВІС. Останнє дозволить досягти прийнятних значень техніко-економічних показників ОС.

Структурні затримки при передачах інформації між машинами ОС визначаються відстанню (в сенсі теорії графів) між вершинами структури, зіставленими взаємо-дійючими машинами . Для оцінки структурних затримок в обчислювальних системах використовують діаметр d_i і середній діаметр $\bar{d}$ структури. $P_{ИС}=NC_{Ш}$

Діаметр визначається максимальною відстанню на множині коротших шляхів між парами вершин структури ОС:

$$d=\max_{i,j}\{d_{ij}\} \qquad (1)$$

а середній діаметр –

$$\bar{d} = (N-1)^{-1} \sum_{l} l \cdot n_l \qquad (2)$$

де d_{ij} - відстань, тобто мінімальне число ребер, що утворюють шлях з вершини i в вершину j; $i, j \in \{0,1,\ldots,N-1\}$; n_i - число вершин, що знаходяться на відстані l від будь-якої виділеної вершини (однорідного) графа G .

Показником що оцінює структуру комутованої ОС, є вектор-функція

$$\chi(G, s, s^i) = \chi_h(G, s, s^i)),\ h \in \{1,2,\ldots,N/2\}; \qquad (3)$$

в якій координата $\chi_h(G, s, s^i)),$ є ймовірність реалізації в системі при заданій структурі G і коефіцієнтах готовності s, s^i, відповідно, одній ЕМ і лінії зв'язку h .

Структурна живучість ОС оцінюється вектор-функцією

$$\chi(G, s, s^i) = \{\chi_i(G, s, s^i)\},\ r \in E_2^N\ \{1,2,\ldots,N\} \qquad (4)$$

$\chi_i(G, s, s^i),$ є ймовірністю існування підсистеми рангу r (підмножиною з r працездатних ЕМ, зв'язність яких встановлюється через працездатні лінії зв'язку) при заданій структурі G, коефіцієнтах готовності s і s' елементарної машини і лінії зв'язку, відповідно.

Для оцінки продуктивності каналів зв'язку між ЕМ системи використовують показники: пропускна здатність і латентність.

Пропускна здатність каналу зв'язку (bandwidth, throughput, channel capacity) - найбільша кількість інформації, що передається по каналу зв'язку в одиницю часу.

Латентність (latency) - це затримка при передачі інформації між ЕМ системи, викликана програмними і апаратними накладними витратами.

Введені показники дозволяють здійснити досить повний аналіз структурних можливостей ОС.

Найбільш повно перерахованим вище вимогам задовольняють однорідні структури (тобто описувані однорідними графами). Такі структури є перспективними для формування масштабованих і великомасштабних обчислюваних систем (зокрема, ОС з програмованою структурою).

У комп'ютерній індустрії набули поширення n-мірні структури обчислювальних систем, відомі зараз як циркулянтні (Circulant Sructures**).** Вперше вони були визначені і досліджені в відділі обчислювальних систем Інституту математики АН СРСР на початку 70-х років і спочатку називалися - D_n графами [15].

За визначенням D_n - граф або циркулянтна структура є граф G виду: $\{N; \omega_1, \omega_2, \ldots, \omega_n\}$, в якому:

- N число вершин або порядок графа;
- вершини позначені цілими числами i по модулю N, отже, $i \in \{0,1,\ldots,N-1\}$;
- вершина i з'єднана ребром (або є суміжною) з вершинами [illegible]
- $\{\omega_1, \omega_2, \ldots, \omega_n\}$ множина цілих чисел, яких називають утворюючими, такими, що, $0 < \omega_1 < \omega_2 < \ldots < \omega_n < (N+1)/2$, а для чисел $N; \omega_1, \omega_2, \ldots \omega_n$ найбільшим спільним дільником є 1;
- n - розмірність графа;
- $2n$ - ступінь вершини в графі.

Як приклад розглянемо D-граф (рис. 1) або двовимірний циркулянт виду: $\{12; 3, 4\}$

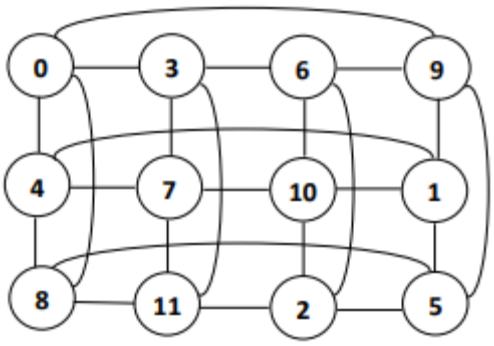


Рис. 2. D - граф: $(12;3,4)$

Графи G виду $(N;\omega_1,\omega_2,...,\omega_n)$ тобто D- графи або циркулянт з одиничною утворюючою (Loop Networks - петльові структури) інтенсивно вивчаються останнім часом. Циркулянтні структури $(N;1,\omega_2)$ широко внесені в практику обчислювальних систем.

Цілі числа $i \in (0,1...,N-1)$, що визначають вершини D- графа, називаються адресами. Адресація вершин в таких структурах називається діофантовою (в честь давньогрецького математика з Олександрії Діофанта, Diophantos, 3 століття).

У циркулянтних структурах при повному перенесенні будь-якої підструктури (всіх вершин підструктури на одну і ту ж відстань в одному з напрямків) зберігаються всі її властивості і адресація вершин.

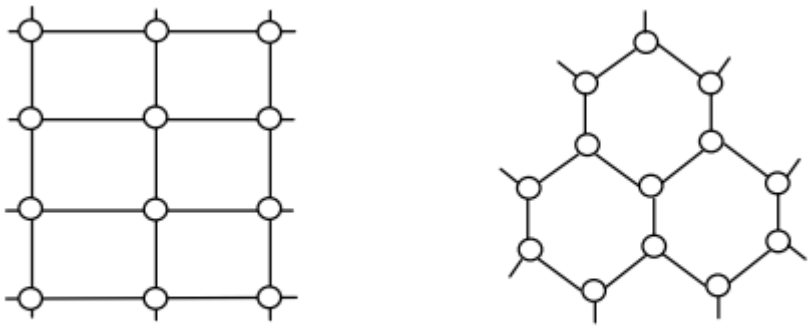
Отже, при діофантовій адресації елементарних машин ОС можна простими засобами реконфігурації здійснити віртуальну адресацію вершин-машин і, отже, створювати відмовостійкі паралельні програми, реалізовувати мультипрограмні режими обробки інформації, виключати вершини-машини, що відмовили, з підсистем, а значить забезпечити живучість ОС.

При цьому алгоритм роботи реконфігуратора структури ОС зводиться до зміни адреси всіх машин підсистеми за формулою:

$$\alpha := (\alpha + (j-i)) \pmod N, \alpha \in (0,1,....,N-1) \qquad (5)$$

де i - номер ЕМ, виключеного з підсистеми, а j - номер машини, підключеної в підсистему, $i, j \in (0,1,....,N-1)$;

В якості структур ОС, що допускають масштабування (зміна числа машин) без корінної перекомутації вже наявних міжмашинних зв'язків, використовуються $K(N,v,g)$ - графи [10]. У такі графи вкладаються D- графи; $K(N,v,g)$- граф - це не орієнтований однорідний граф з числом і степенями вершин, відповідно, N і v і значенням обхва та g (рис.3)



а *б*

Рис. 3. Фрагменти $K(N,v,g)$ - графів:

В $K(N,v,g)$ - графах кожна вершина при v ≥3 входить в не менше v найкоротших простих циклів довжиною g (довжина найкоротшого цикл в графі називається обхватом). При v =2 Л (N, v, g) -граф є простим циклом з N вершинами.

Паралельні алгоритми і програми. Поняття паралельного алгоритму (Parallel Algorithm) відноситься до фундаментальних в теорії обчислювальних систем [1]. Паралельний алгоритм - це опис процесу обробки інформації, орієнтований на реалізацію в колективаах обчислювачів Такий алгоритм, на відміну від послідовного, передбачає одночасне виконання множини операцій в межах одного кроку обчислень і як послідовний алгоритм зберігає залежність наступних етапів від результатів попередніх.

Паралельний алгоритм розв'язання задачі становить основу паралельної програми, яка, в свою чергу, впливає на алгоритм функціонування колективу обчислювачів. Запис паралельного алгоритму на мові програмування, доступному колективу обчислювачів, і називають паральною- програмою, а сама мова - паралельною. Паралельні алгоритми і програми слід розробляти для тих задач, які недоступні для вирішення на засобах, заснованих на моделі обчислювача. Ці задачі прийнято називати складними або трудомісткими.

Методи і алгоритми обробки інформації, вирішення задач, як правило, - послідовні. Процес "пристосування" методів до реалізації на колективі обчислювачів або процес "розщеплення" послідовних алгоритмів вирішення складних задач називається розпаралелюванням (Paralleling***).***

Теоретична і практична діяльність по створенню паралельних алгоритмів і програм обробки інформації називається паралельним програмуванням (Parallel Programming***).***

Якість паралельного алгоритму (або його ефективність) визначається методикою розпаралелювання складних задач. Виділяють два основних підходи до розпаралелювання задач: локальне і глобальне (великоблочне) розпаралелювання. Перший підхід орієнтований на розчеплення алгоритму розв'язання складної задачі на гранично прості блоки (операції або оператори) і вимагає виділення для кожного етапу обчислення максимально можливої кількості одночасно виконуваних блоків. Він не призводить до паралельних алгоритмів, ефективно реалізованим колективом обчислювачів. Справді, процес такого розпаралелювання досить трудомісткий, а одержування паралельних алгоритмів характеризуються не тільки структурною неоднорідністю, а й істотно різними обсягами операцій на різних етапах обчислень. Останнє є серйозною перешкодою на шляху (автоматизації) розпаралелювання і забезпечення ефективної експлуатації ресурсів колективу обчислювачів. Локальне розпаралелювання дозволяє оцінити граничні можливості колективу обчислювачів при вирішенні складних задач, отримати граничні оцінки по розпаралеленню складних задач.

Другий підхід орієнтований на розбиття складної задачі на великі блоки-підзадачі, між якими існує слабкий зв'язок. Тоді в алгоритмах, побудованих на основі велико-блочного розпаралелювання, операції обміну між підзадачами становитимуть незначну частину в порівнянні із загальним числом операцій в кожній підзадачі. Такі підзадачі називають гілками паралельного алгоритму, а відповідні їм програми - гілками паралельної програми.

Кластерні та мультикластерні ОС і GRID-системи. При побудові кластерних ОС, як правило, використовуються масові апаратно-програмні засоби. Останнє, по суті, є принципом конструювання кластерних ОС, що забезпечує їх високу технічно-економічну ефективність [15].

Для створення обчислювальних кластерів використовуються MISD-, SIMD-, MIMD-архітектури, різні функціональні структури і конструктивні рішення.

Основна функціонально-структурна одиниця обчислювальних ресурсів в кластерних ОС - це елементарна машина. Конфігурація ЕМ допускає варіювання в широких межах - від

процесорного ядра до ЕОМ, оснащеної засобами комунікацій і зовнішніми пристроями.

Сучасні кластерні ОС [9] переважно конфігуруються з багатопроцесорних вузлів і багатоядерних процесорів. Комунікаційні середовища кластерних ОС будуються на базі коммутаторів і мають ієрархічну організацію. На рис 3 наведено приклад обчислювального кластера, укомплектованого N обчислювальними вузлами, які об'єднані мережею зв'язку стандарту Gigabit Ethernet. Кожен вузол укомплектований парою двоядерних процесорів AMD Opteron 275, взаємодіють по шині HyperTransport. Кожен процесор має інтегрований контролер пам'яті, через який його ядра маю доступ до їх загальної пам'яті. Видно, що комунікаційне середовище кластера має ієрархічну організацію, в якій перший рівень - це мережа зв'язку стандарту Gigabit Ethernet; другий - шина HyperTransport; третій - спільна пам'ять ядер. Швидкість передачі інформації на різних рівнях комунікаційного середовища істотно різні.

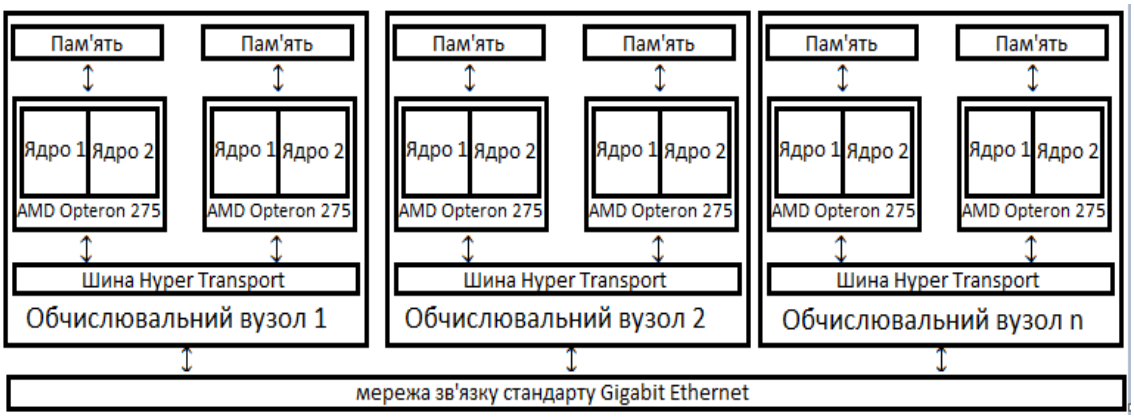


Рис. 4. Приклад ієрархічної організації комунікаційного середовища кластерної ОС

Комунікаційні середовища сучасних кластерних ОС будуються на базі технологій Gigabit Ethernet, InfiniBand, Myrinet. Для об'єднання обчислювальних вузлів в систему використовуються комутатори (switch) з фіксованою кількістю вхідних портів для підключення кінцевих пристроїв (обчислювальних вузлів, комутаторів, систем зберіганя даних і т.д.). У цих умовах для побудови великомасштабних кластерних ОС використовуються різні схеми формування комутаторів. Основна мета - комплекснування заданого числа обчислювальних вузлів і забезпечення високої продуктивності комунікаційного середовища.

З розвитком обчислювальної техніки і телекомунікаційних технологій з'явилася тенденція до побудови розподілених ОС шляхом об'єд нання за допомогою локальних і глобальних мереж зв'язку розосереджених обчислювальних ресурсів починаючи від простих ПК і закінчуючи ОС з масовим паралелізмом. Так в кінці XX століття в індустрії обробки інформації отримали розвиток GRID-технології (Global Resource Information Distribution) [15]. На основі цих технологій створюються быльш масштабні просторово-розподілені системи обробки інформації, здатні реалізувати паралельні алгоритми вирішення суперскладних задач на своїх розосереджених ресурсах. Така GRID-система являє собою композицію множини ЕОМ і ОС, просторово-розподіленої комунікаційної мережі і програмних компонентів для здійснення паралельних обчислень. В GRID-системі можуть використовуватися гетерогенні і не сумісні обчислювальні засоби, однак для їх сумісної роботи над паралельними алгоритмами повині бути застосовані спеціальні механізми (наприклад, стандартизовані протоколи) узгодженої взаємодії. GRID-система - це ні що інше як розподілена ОС в сенсі введеного вище визначення.

Що стосується мультикластерних ОС, то це не окремий клас ОС, а просторово-розподілені системи, що формуються шляхом об'єднання ресурсів розосереджених кластерів.

Обчислювальні кластери належать до класу систем з розподіленою пам'яттю.

Паралельні програми для таких ОС розробляються в моделі передачі повідомлень (message passing). Широке розповсюдження отримали стандарти, бібліотеки і середовища для створення паралельних програм в даній моделі, зокрема стандарт Message Passing Inter face (MPI), система Parallel Virtual Machine (PVM), IBM Message Passing Li brary (MPL), P4 і ін.

В рамках даної моделі паралельні гілки програми синхронізують свою роботу шляхом обміну інформацією по каналах між машиних зв'язків.

Залежно від складності задач і характеру їх надходження в теорії обчислювальних систем виділяють два основні режими функціонування ОС з програмованою структурою [15]: моно- і мультипрограмний режими.

У монопрограмному режимі для вирішення задачі використовуються всі ре сурси ОС. Задача представляється у вигляді паралельної програми, число гілок в якій або фіксоване, або допускає варіювання в заданому діапазоні. В якості одиниці ресурсу виступає елементарна машина ОС. Всі машини використовуються для вирішення задачі. Якщо максимальне число гілок в паралельній програмі менше загального числа ЕМ в системі, то "надлишкові" машини використовуються для підвищення надійності функціонування ОС.

До мультипрограмних відносять режими :

1) обробки наборів задач;

2) обслуговування потоків задач.

При роботі ОС в цих режимах одночасно вирішується кілька задач, отже, ресурси системи діляться між ним.

При організації функціонування ОС у разі обробки набору задач враховується не тільки кількість задач в наборі, але їх параметри: число гілок в програмі (точніше, число машин, на яких вона буде виконуватись), час рішення або імовірнісний закон розподілу часу рішення і ін. Алгоритми організації функціонування ОС задають розподіл задач по машинам і послідовність виконання задач на них. В результаті стає відомим, в якому проміжку часу і на яких машинах (або на який підсистемі) буде вирішуватися будь яка задача набору.

Обслуговування потоку задач на ОС - принципово відрізняється від вибору наборів: задачі надходять у випадкові моменти часу, їх параметри випадкові, отже, детермінований вибір підсистем для вирішення тих чи інших задач виключений. Для режиму потоку задач використовуються методи і алгоритми, що забезпечують оптимальне функціонування обчислювальних систем.

При роботі ОС в будь-якому з мультипрограмних режимів система представляється у вигляді композиції підсистем різних рангів. У міру рішення задачі ця композиція "автоматично" (за допомогою операційної системи) реконфігурується так, щоб забезпечити її адекватність поточної мультипрограмної ситуації. Будь-яка підсистема має всі архітектурні властивості системи, тому її організація при вирішенні виокремленної їй задачі може здійснюватися тими ж методами, що і організується робота всієї ОС в першому режимі.

Час виконання паралельних програм на розподілених ОС сутєво залежить від того наскільки вони ефективно вкладені в систему [2].

Під ефективним вкладенням (Task Map, Task Allocation, Task Assign ment) розуміється такий розподіл гілок паралельної програми між ЕМ системи, при якому досягаються мінімуми накладних витрат на міжмашинному обміні інформацією і дисбалансу завантаження ЕМ.

Залежно від того, чим характеризується паралельна програма, яка надійшла в систему, виникають різні постановки задачі вкладення.

Якщо для паралельної програми заданий інформаційний граф, то говорять про задачу вкладення інформаційного графа (структури) програми в ОС. У разі, якщо про паралельну програму відомо лише кількість ЕМ, які необхіно для її реалізації, то розглядається задача формування в межах ОС підсистеми ЕМ, що задовольняє заданим критеріям якості.

Задача оптимального вкладення паралельної програми виникає при організації функціонування ОС в будь-якому з основних режимів: рішення складної задачі, обробки наборів задач, обслуговування потоків задач.

В режимі обслуговування потоків задач в межах ОС формуються підсистеми ЕМ. Для них і надходять паралельні програми для виконання. Елементарні машини, що входять в підсистему, можуть належати різним обчислювальним системам, розміщуватися в різних комутаційних шафах і т. д. Наслідком цього є те, що швидкості передачі інформації між ЕМ можуть бути істотно різними. Тому потрібно не тільки формувати підсистеми з необхідним числом ЕМ, а й ефективно вкладати паралельні програми в них.

У режимі обробки наборів задач будується розклад виконання паралельних програм. В даному режимі задача вкладення виникає або на етапі запуску програми на виділеній підсистемі, або ж на етапі синтезу розкладу, в разі, якщо алгоритми вкладення інтегровані в підсистему підтримки мультипрограмних режимів функціонування ОС.

Існуючі методи і алгоритми вкладення паралельних програм в ОС розрізняються за типом систем, на які вони орієнтовані, по використовуваних моделях паралельних програм, показниками оптимальності вкладення, точності, централізованності або децентрализованности алгоритмів, по статичній або динамічнії схемі формування вкладень.

У роботах [11,12] запропоновано алгоритми вкладення паралельних програм, в яких ОС представлена у вигляді графа міжмашинних зв'язків. Передбачається, що інформація між ЕМ передається по найкоротшому шляху і всі канали зв'язку однорідні. Як показує продуктивність каналу зв'язку між ЕМ використовується довжина шляху (в сенсі теорії графів). Застосування подібних алгоритмів ускладнено тим, що сучасні ОС мультиархітектурні, їх комунікаційні середовища мають ієрархічну організацію і, як наслідок, не однорідні канали зв'язку між ЕМ. Крім того, більшість систем будується на базі комутаторів, в яких застосовуються алгоритми динамічної маршрутизації. Тому доцільно використання моделей обчислювальних систем, в яких продуктивність каналів зв'язку характеризується такими показниками як пропускна здатність і латентність.

Ефективне вкладення в ОС паралельної програми вимагає врахування структури інформаційних обмінів і обсягів даних, що передаються між її гілками. У роботах [11,12] запропоновано алгоритми, які основані лише на структурі інформаційного графа програми і ігнорують обсяги та інтенсивність міжмашиних обмінів. Однак облік таких параметрів практично необхідний. Аналіз популярних вільно розповсюджених паралельних програм, бібліотек і тестів продуктивності High-Performance Linpack, NAS Parallel Benchmarks і SPEC MPI2007 показує, що більшість з них характеризується неоднорідними інформаційними графами. Неоднорідність виражається в обсягах даних, що передаються між гілками, розмірах використовуваних повідомлень, а також в схемах реалізованих обмінів.

Як показники оптимальності вкладення використовують час виконання паралельної програми або функцію, мінімізація якої доставляє мінімум часу виконання програми.

Алгоритми вкладення спираються на припущення про те, що час виконання паралельної програми - це сума часу виконання усіх її гілок. Однак практика рішення промислових задач і аналізу їх продуктивності показує, що час виконання програми визначається максимальним часом виконання її гілок. Крім того, в ряді робіт [10,11,12] використовуються складові показники у вигляді лінійної комбінації двох функцій, наприклад, середня завантаженість елементарних машин в системі і час передачі даних по каналах зв'язку. Використання подібних показників ускладнюється вибором коефіцієнтів пропорційності, якими пов'язані функції і найчастіше їх не сумісними розмірностями. Наприклад, розмірність першої функції - кількість операцій, а другої - одиниці часу. Вкладення паралельних програм в просторово розподілені ОС вимагає обліку в показнику часу доставки паралельних програм до елементарних машин системи.

Вкладення в ОС паралельних програм, для яких не задані інформаційні графи, здійснюється шляхом формування підсистеми елементарних машин і розподілу по ній паралельних гілок програми. Зформована підсистема повинна забезпечувати ефективну реалізацію паралельних програм і забезпечувати мінімум часу для їх виконання.

Більшість алгоритмів формування в межах ОС підсистем орієнтовані на системи з певними структурами мереж міжмашинного зв'язку (наприклад, на 2D-решітки,

гіперкубічські і тороїдальні структури) і прагнуть зберігати топологічну схожість підсистеми структурі ОС. Можливості застосування таких алгоритмів для ОС з ієрархічною організацією, канали зв'язку в яких мають різну продуктивність, істотно обмежені. Тому актуальною є розробка алгоритмів формування підсистем, що допускають застосування як в системах зі статичною структурою мережі між машинного зв'язку, так і в системах з динамічною структурою.

Висновки. Сучасні ОС функціонують під управлінням систем пакетної обробки завдань (наприклад, TORQUE / OpenPBS, IBM LoadLeveler, Sun Grid Engine, Altair PBS Pro, SLURM, Cleo та ін.), які реалізують підтримку мультипрограмних режимів функціонування. Такі системи пакетної обробки завдань (СПОЗ) підтримують черги завдань і здійснюють планування обчислень. Для кожної паралельної програми за допомогою алгоритмів підтримки в межах системи формується підсистема ЕМ.

Поширення отримали елементарні алгоритми формування підсистем: PAL (Processor Allocation Linear) - здійснює вибір перших M вільних ЕМ, і PAR (Processor Allocation Random) - генерує випадкову підмножину з M вільних ЕМ.

Початкове вкладення паралельної програми і виділення підсистем також здійснюється СПОЗ. Дане завдання найчастіше вирішується циклічним(Task Map Round Robin, TMRR) або лінійним (Task Map Linear) алгоритмом формування вкладень. Алгоритм TMRR розподіляє паралельні гілки по ЕМ системи з N обчислювальних вузлів в наступному порядку: перша гілка на перший обчислювальний вузол, друга на другий, ..., гілка N на вузол N, гілка $N+1$ на вузол 1 і т. д. Алгоритм TML розподіляє гілки на перші M вільних ЕМ.

Засоби оптимізації вкладення паралельних програм присутні і в комунікаційних бібліотеках стандарту MPI (MPICH2, OpenMPI, MVAPICH2, Intel MPI, HP-MPI). Вкладення реалізується утилітою *mpiexec*, що підтримує в більшості випадків лише алгоритми TMRR і TML.

Аналіз архітектурних особливостей обчислювальних систем з програмованою структурою дозволяє зробити висновок:

1. Сучасні обчислювальні системи є мультиархітектурними, їх комунікаційні середовища мають ієрархічну організацію.

2. Ефективність експлуатації ресурсів розподілених ОС значно залежить від того, як організовано їх функціонування. У будь-якому з режимів функціонування виникає завдання ефективного вкладення паралельної програми в ОС. Під ефективним вкладенням розуміється такий розподіл гілок паралельної програми, при якому досягаються мінімуми накладних витрат на міжмашинний обмін і дисбаланс завантаження елементарних машин.

3. Існуючі алгоритми не дозволяють будувати ефективні вкладення паралельних програм в ОС з ієрархічною організацією комунікаційних середовищ. Актуальною є разрабка не трудомістких методів і алгоритмів вкладення паралельних програм в ОС з ієрархічною організацією комунікаційних середовищ.

4. Задача оптимального вкладення є важкою. Актуальною є розробка наближених методів і алгоритмів вкладення паралельних програм в розподіленій ОС.

ЛІТЕРАТУРА:

1. Воеводин, В. В. Параллельные вычисления / В. В. Воеводин, Вл. В. Воеводин. – СПб. : БХВ-Петербург, 2002. – 608 с.

2. Гергель, В. П. Основы параллельных вычислений для многопроцессорных вычислительных систем / В. П. Гергель, Р. Г. Стронгин. – Нижний Новго род : Изд-во ННГУ, 2003. – 184 с.

3. Каляев, А. В. Модульно-наращиваемые многопроцессорные системы со структурно-процедурной организацией вычислений /А. В. Каляев, И. И. Левин. – М. : Янус-К, 2003. – 380 с.

4. Каляев, И. А. Реконфигурируемые мультиконвейерные вычислительные структуры / И. А. Каляев ;под. ред. И. А. Каляева. – Ростов н/Д. : ЮНЦ РАН, 2008. – 320 с.

5. Кормен, Т. Х. Алгоритмы: построение и анализ, издание 2-е / Т. Х. Кормен [и др.] ;пер. с англ. под ред. И. В. Красикова. – М. : Изда тельский дом “Вильямс”, 2005. – 1296 с.

6. Корнеев, В. В. Вычислительные системы / В. В. Корнеев. – М. : Гелиос АРВ, 2004. – 512 с.

7. Курносов, М. Г. Опыт построения кластерных вычислительных систем с удаленной загрузкой узлов / М. Г. Курносов // Материалы пятого Между народного научно-практического семинара “Высокопроизводительные параллельные вычисления на кластерных системах”. – Нижний Новгород: Изд-во ННГУ, 2005. – С. 149-154.

8. Курносов, М. Г. Назначение ветвей параллельной программы на процес сорные ядра распределенной вычислительной системы / М. Г. Курносов // Материалы Международной научно-технической конференции “Много процессорные вычислительные и управляющие системы”. – Таганрог:Изд-во ТТИ ЮФУ, 2007. – Т. 1. – С. 227-231.

9. Левин, В. К. Современные суперкомпьютеры семейства МВС [Электрон ный ресурс] / В. К. Левин. – Электрон. текст. дан. – Б. м., 2002. – Режим доступа : http://www.parallel.ru/mvs/levin.html.

10. Монахов, О. Г. Параллельные системы с распределенной памятью: струк туры и организация взаимодействий / О. Г. Монахов, Э. А. Монахова. – Новосибирск : Изд-во СО РАН, 2000. – 242 с.

11. Tarkov, M. S. Mapping parallel programs onto distributed computer systems with faulty elements / M. S. Tarkov [et al.] // Lecture Notes in Computer Science: Computational Science. – 2001. – P. 148-157.

12. Тарков, М.С. Вложение структур параллельных программ в структуры живучих распределенных вычислительных систем / М. С. Тарков // Авто метрия. – 2003. – Том 39, № 3. – С. 84-96.

13. Хорошевский, В. Г. Алгоритмы распределения ветвей параллельных про грамм по процессорным ядрам вычислительных систем / В. Г. Хорошевский, М. Г. Курносов // Автометрия. – 2008. – Т. 44, № 2. – С. 56-67.

14. Хорошевский, В. Г. Моделирование алгоритмов вложения параллельных программ в структуры распределенных вычислительных систем / В. Г. Хорошевский, М. Г. Курносов // Труды Международной научной конференции “Моделирование-2008” (Simulation-2008). – Киев: Изд-во ИПМЭ им. Г.Е. Пухова, 2008. – Т. 2. – С. 435-440.

15. Хорошевский, В. Г. Архитектура вычислительных систем / В. Г. Хоро шевский. – М. : МГТУ им. Н. Э. Баумана, 2008. – 520 с.

REFERENCES:

1. Voevodyn, V. V. Parallelnыe vыchyslenyia / V. V. Voevodyn, Vl. V. Voevodyn. – SPb. : BKhV-Peterburh, 2002. – 608 s.

2. Herhel, V. P. Osnovы parallelnыkh vыchyslenyi dlia mnohoprotsessornыkh vыchyslytelnыkh system / V. P. Herhel, R. H. Stronhyn. – Nyzhnyi Novho rod : Yzd-vo NNHU, 2003. – 184 s.

3. Kaliaev, A. V. Modulno-narashchyvaemыe mnohoprotsessornыe systemы so strukturno-protsedurnoi orhanyzatsyei vыchyslenyi A. V. Kaliaev, / Y. Y. Levyn. – M. : Yanus-K, 2003. – 380 s.

4. Kaliaev, Y. A. Rekonfyhuryruemыe multykonveiernыe vыchyslytelnыe strukturы / Y. A. Kaliaev ;pod. red. Y. A. Kaliaeva. – Rostov n/D. : YuNTs RAN, 2008. – 320 s.

5. Kormen, T. Kh. Alhorytmы: postroenye y analyz, yzdanye 2-e / T. Kh. Kormen [y dr.] ;per. s anhl. pod red. Y. V. Krasykova. – M. : Yzda telskyi dom “Vyliams”, 2005. – 1296 s.

6. Korneev, V. V. Vыchyslytelnыe systemы / V. V. Korneev. – M. : Helyos ARV, 2004. – 512 s.

7. Kurnosov, M. H. Opыt postroenyia klasternыkh vыchyslytelnыkh system s udalennoi zahruzkoi uzlov / M. H. Kurnosov // Materyalы piatoho Mezhdu narodnoho nauchno-praktycheskoho semynara “Vыsokoproyzvodytelnыe parallelnыe vыchyslenyia na klasternыkh systemakh”. – Nyzhnyi Novhorod: Yzd-vo NNHU, 2005. – S. 149-154.

8. Kurnosov, M. H. Naznachenye vetvei parallelnoi prohrammы na protses sornыe yadra raspredelennoi vыchyslytelnoi systemы / M. H. Kurnosov // Materyalы Mezhdunarodnoi nauchno-tekhnycheskoi konferentsyy “Mnoho protsessornыe vыchyslytelnыe y upravliaiushchye systemы”. – Tahanroh:Yzd-vo TTY YuFU, 2007. – T. 1. – S. 227-231.

9. Levyn, V. K. Sovremennыe superkompiuterы semeistva MVS [Эlektron nыi resurs] / V. K. Levyn. – Эlektron. tekst. dan. – B. m., 2002. – Rezhym dostupa : http://www.parallel.ru/mvs/levin.html.

10. Monakhov, O. H. Parallelnыe systemы s raspredelennoi pamiatiu: struk turы y orhanyzatsyia vzaymodeistvyi / O. H. Monakhov, Э. A. Monakhova. – Novosybyrsk : Yzd-vo SO RAN, 2000. – 242 s.

11. Tarkov, M. S. Mapping parallel programs onto distributed computer systems with faulty elements / M. S. Tarkov [et al.] // Lecture Notes in Computer Science: Computational Science. – 2001. – P. 148-157.

12.Tarkov, M.S. Vlozhenye struktur parallelnыkh prohramm v strukturы zhyvuchykh raspredelennыkh vыchyslytelnыkh system / M. S. Tarkov // Avto metryia. – 2003. – Tom 39, № 3. – S. 84-96.

13.Khoroshevskyi, V. H. Alhorytmы raspredelenyia vetvei parallelnыkh pro hramm po protsessornым yadram vыchyslytelnыkh system / V. H. Khoroshevskyi, M. H. Kurnosov // Avtometryia. – 2008. – T. 44, № 2. – S. 56-67.

14.Khoroshevskyi, V. H. Modelyrovanye alhorytmov vlozhenyia parallelnыkh prohramm v strukturы raspredelennыkh vыchyslytelnыkh system / V. H. Khoroshevskyi, M. H. Kurnosov // Trudы Mezhdunarodnoi nauchnoi konferentsyy “Modelyrovanye-2008” (Simulation-2008). – Kyev: Yzd-vo YPMЭ ym. H.E. Pukhova, 2008. – T. 2. – S. 435-440.

15.Khoroshevskyi, V. H. Arkhytektura vыchyslytelnыkh system / V. H. Khoro shevskyi. – M. : MHTU ym. N. Э. Baumana, 2008. – 520 s.

Ph.D. in Technical SciencesOgnevy O.V.,
Prysyazhnyuk V., Koval B.

ANALYSIS OF ARCHITECTURAL PECULIARITIES OF COMPUTER SYSTEMS WITH PROGRAMMABLE STRUCTURE

The article is devoted to the construction of multi-architectural cluster and multi-cluster computing systems.

The paper analyzes computational systems with programmable structure. Studies have shown that modern distributed OSs are multiarchitectural. Depending on the level of consideration of their functional structures, they may look both as MISD, both as SIMD, and as MIMD systems.

The main functional and structural unit of computing resources in the systems of the class under consideration is an elementary machine (EM), which is a composition of a computing module and a system device.

It has been established that OS systems with a programmable structure are scalable information processing tools and allow the formation of scalable massively parallel architecture computing systems.

Interaction between the computer is carried out through the software configuration of the communication network. The structure of the OS is described by a graph, the set of vertices of which is the number of elementary machines (or system devices, or local switches), and the set of edges - the line of interconnection.

The structure of the modern OS presents a number of requirements: the simplicity of embedding a parallel algorithm, the convenience of addressing elementary machines, the feasibility of the principle of near-the-go and the minimum delays in inter-machine transmissions of information, scalability and large-scale structures, commutation, survivability, technology. The most complete requirements listed above satisfy the homogeneous structures (that is, they are described by homogeneous graphs). Such structures are promising for the formation of scalable and large-scale computing systems (in particular, OS with a programmable structure).

To assess the performance of the communication channels between the EM system used indicators of bandwidth and latency, to allow a fairly complete analysis of the structural capabilities of the OS.

In this paper, we present an example of constructing a computing cluster, complete with N computing nodes, which are connected by a Gigabit Ethernet connection.

Key words: computing systems, elementary machines, data processing systems, homogeneous structures, communication environments, cluster and multiclaser, embedding algorithms.

АНАЛІЗ АРХІТЕКТУРНИХ ОСОБЛИВОСТЕЙ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ З ПРОГРАМОВАНОЮ СТРУКТУРОЮ

Огнєвий Олександр Вікторович, кандидат технічних наук, доцент кафедри кібербезпеки та комп’ютерних систем і мереж Хмельницького національного університету (Хмельницький, Україна).

Коваль Богдан Анатолійович, магістр кафедри кібербезпеки та комп’ютерних систем і мереж Хмельницького національного університету (Хмельницький, Україна)

Присяжнюк Віталій Володимирович, магістр кафедри кібербезпеки та комп’ютерних систем і мереж Хмельницького національного університету (Хмельницький, Україна)

ANALYSIS OF ARCHITECTURAL PECULIARITIES OF COMPUTER SYSTEMS WITH PROGRAMMABLE STRUCTURE

Ognjevyj Oleksandr, Candidate of Technical Sciences, assistant professor of the Department of Cybersecurity and Computer Systems and Networks of the Khmelnitsky National University (Khmelnitsky, Ukraine).

Koval Bogdan, magister of cyber security and computer systems and networks Khmelnytsky National University (Khmelnitsky, Ukraine)

Prisyajnyuk Vitaliy, magister of cyber security and computer systems and networks Khmelnytsky National University (Khmelnitsky, Ukraine)