ЕФЕКТИВНІСТЬ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ

1 Показники ефективності обчислювальних машин

Обчислювальну машину можна визначити множиною показників, що характеризують окремі її властивості. Виникає завдання введення міри для оцінки ступеня пристосованості ОМ до виконання покладених на неї функцій – міри ефективності.

Ефективність визначає ступінь відповідності ОМ своєму призначенню. Вона вимірюється або кількістю витрат, необхідних для отримання певного результату, або результатом, отриманим при певних витратах. Провести порівняльний аналіз ефективності декількох ОМ, ухвалити рішення на використання конкретної машини дозволяє критерій ефективності.

Критерій ефективності – це правило, що служить для порівняльної оцінки якості варіантів ОМ. Критерій ефективності можна назвати правилом переваги порівнюваних варіантів.

Будуються критерії ефективності на основі окремих показників ефективності (показників якості). Спосіб зв'язку між окремими показниками визначає вид критерію ефективності.

Як основні показники ОМ зазвичай розглядають: ємність пам'яті, швидкодію та продуктивність, вартість і надійність [25]. Зупинимося тільки на показниках швидкодії та продуктивності, що зазвичай представляють основний інтерес для користувачів.

Швидкодія. Доцільно розглядати два види швидкодії: номінальну і середню.

Номінальна швидкодія характеризує можливості ОМ під час виконання стандартної операції. Як стандартну зазвичай вибирають коротку операцію додавання. Якщо позначити через $\tau_{дод}$ час додавання, то номінальна швидкодія визначиться з виразу

$$V_{ном} = \frac{1}{\tau_{дод}} \left[\frac{оп}{с} \right]. \qquad (1)$$

Середню швидкодію характеризує швидкість обчислень при виконанні еталонного алгоритму або деякого класу алгоритмів. Величина середньої швидкодії залежить як від параметрів ОМ, так і від параметрів алгоритму і визначається співвідношенням

$$V_{сер} = \frac{N}{T_e}, \qquad (2)$$

де T_e – час виконання еталонного алгоритму;

N – кількість операцій, що містяться в еталонному алгоритмі.

Позначимо через n_i число операцій *i*-го типу; l – кількість типів операцій в (*i = 1, 2..., l*); τ_i – час виконання операції *i*-го типу.

Час виконання еталонного алгоритму розраховується за формулою

$$T_e = \sum_{i=1}^{l} \tau_i n_i \,. \tag{3}$$

Підставивши (3) у вираз для $V_{сер}$, отримаємо

$$V_{сер} = \frac{N}{\sum_{i=1}^{l} \tau_i n_i} \,. \tag{4}$$

Розділимо чисельник і знаменник в (4) на *N*:

$$V_{сер} = \frac{1}{\sum_{i=1}^{l} \frac{n_i}{N} \tau_i} \,. \tag{5}$$

Позначивши частоту появи операції *i*-го типу в (5) через $q_i = \frac{n_i}{N}$, запишемо остаточну формулу для розрахунку середньої швидкодії

$$V_{сер} = \frac{1}{\sum_{i=1}^{l} q_i \tau_i} \left[\frac{on}{c} \right]. \tag{6}$$

У виразі (6) вектор $\{\tau_1, \tau_2, ..., \tau_l\}$ характеризує систему команд ОМ, а вектор $\{q_1, q_2, ..., q_l\}$, що називається частотним вектором операцій, характеризує алгоритм.

Очевидно, що для ефективної реалізації алгоритму необхідно прагнути до збільшення $V_{сер}$. Якщо $V_{ном}$ головним чином відштовхується від швидкодії елементної бази, то $V_{сер}$ дуже сильно залежить від оптимальності вибору команд ОМ.

Формула (6) дозволяє визначити середню швидкодію машини під час реалізації одного алгоритму. Розглянемо більш загальний випадок, коли повний алгоритм складається з декількох окремих, періодично повторюваних алгоритмів. Середня швидкодія під час рішення повної задачі розраховується за формулою

$$V_{сер}^{n} = \frac{1}{\sum_{j=1}^{m} \sum_{i=1}^{l} \beta_j q_{ji} \tau_i} \left[\frac{on}{c} \right], \tag{7}$$

де m – кількість окремих алгоритмів;

β_j – частота появи операцій *j*-го окремого алгоритму в повному алгоритмі;

q_{ij} – частота операцій *i*-го типу в *j*-му окремому алгоритмі.

Позначимо через Nj і Tj – кількість операцій і період повторення j-го окремого алгоритму; $T_{\max} = \max_j (T_1, ..., T_j, ..., T_m)$ – період повторення повного алгоритму; $a_j = T_{max}/T_j$ – циклічність включення j-го окремого алгоритму в повному алгоритмі.

Тоді за час T_{max} в ОМ буде виконано $N_{\max} = \sum_{j=1}^{m} \alpha_j N_j$ операцій, а частоту появи операцій j-го окремого алгоритму в повному алгоритмі можна визначити з виразу

$$\beta_j = \frac{\alpha_j N_j}{N_{\max}}. \qquad (8)$$

Для розрахунку за формулами (6.7, 6.8) необхідно знати параметри ОМ, представлені вектором $\{\tau_1, \tau_2, ..., \tau_l\}$, параметри кожного j-го окремого алгоритму – вектор $\{q_{j1}, q_{j2}, ..., q_{jl}\}$ і параметри повного алгоритму – вектор $\{\beta_1, \beta_2, ..., \beta_m\}$.

Продуктивність ОМ оцінюється кількістю еталонних алгоритмів, що виконуються в одиницю часу:

$$P = \frac{1}{T_e} \left[\frac{задач}{c} \right]. \qquad (9)$$

Продуктивність під час виконання повного алгоритму оцінюється за формулою

$$P_n = \frac{1}{\sum_{j=1}^{m} \sum_{i=1}^{l} \alpha_j N_j q_{ij} \tau_i} \left[\frac{задач}{c} \right]. \qquad (10)$$

Продуктивність є більш універсальним показником, ніж середня швидкодія, оскільки в явному вигляді залежить від порядку проходження завдань через ОМ.

На практиці зазвичай використовують простіші вирази [16]. Для оцінки часу виконання програми з динамічною кількістю команд N використовують вираз

$$T = \frac{N \cdot K}{F}, \qquad (11)$$

де K – середня кількість тактів, що витрачаються на вибірку і виконання однієї команди;

F – тактова частота процесора.

Для оцінки продуктивності використовують *пропускну здатність* процесора – кількість команд, що виконуються за одну секунду. У разі послідовного виконання команд пропускна здатність P_s визначається за формулою

$$P_s = \frac{F}{K} . \qquad (12)$$

Для *конвеєрного* процесора пропускна здатність сама по собі ще не є свідоцтвом хорошої продуктивності.

Реальною мірою продуктивності комп'ютера є загальний час виконання програми. В загальному випадку *n*-ступінчастий конвеєр потенційно підвищує продуктивність в *n* разів. Виходить, що чим більше значення *n,* тим вище продуктивність процесора. Проте реальна продуктивність конвеєрного процесора нижча. Кожного разу, коли відбувається зупинка конвеєра, потік команд, що обробляються процесором, різко скорочується. Таким чином, продуктивність конвеєра багато в чому залежить від таких чинників, як накладні витрати переходів і промахів у разі звернення до кеш-пам'яті. Скорочення накладних витрат можна добитися за рахунок введення вторинного кеша, який розміщується між первинним кешем, інтегрованим у мікросхему процесора, і основною пам'яттю. Крім того, в комп'ютерах використовується оптимізуючий компілятор, який по можливості віддаляє залежні один від одного команди, поміщаючи між ними інші. А також, якщо в процесорі існує черга команд, промах під час вибірки команди може мати значно менші негативні наслідки, оскільки процесор продовжує виконання команд, що знаходяться в черзі.

Для підвищення продуктивності конвеєрного процесора здавалося б доціль-ним розділяти процес виконання команд на якомога більшу кількість ступенів. Проте чим більше ступенів, тим вище вірогідність зупинок конвеєра. Це пов'яза-но з тим, що більшість команд виконуються паралельно, в зв'язку з чим навіть значно віддалені одна від одної команди, між якими є залежності, можуть викликати зупинки конвеєра, що, у свою чергу, приводить до зростання наклад-них витрат переходів. Через перераховані причини підвищення продуктивності за рахунок збільшення кількості ступенів виявляється не таким уже істотним.

Ще один важливий чинник, що впливає на продуктивність, – внутрішні затримки під час виконання процесором базових операцій. Особливої уваги заслуговує затримка в АЛП. У багатьох процесорах тактова частота підбирається так, щоб операція складання в АЛП здійснювалася за один такт. Решта операцій розділяється на кроки, що займають той же час, що і операція складання. При цьому АЛП може мати конвеєрну організацію.

У багатьох конвеєрних процесорах використовується від чотирьох до шести ступенів. У ряді процесорів процедура виконання команди ділиться на менші кроки, використовується більша кількість ступенів конвеєра і вища тактова частота.

.

Наприклад, у процесорі UltraSPARC II застосовується 9-ступінчастий конвеєр, а в процесорі Intel Pentium Pro – 12-ступінчастий, Intel Pentium 4 містить 20-ступінчастий конвеєр і функціонує на тактовій частоті від 1,3 до 1,5 ГГц. Для прискорення роботи на кожному такті виконуються два ступені конвеєра

2 Продуктивність мультипроцесорних систем

Через особливості паралельних обчислень для оцінки їх ефективності використовують специфічну систему показників.

Число процесорів багатопроцесорної системи, що паралельно беруть участь у виконанні програми в кожен момент часу *t*, визначають поняттям *ступінь паралелізму D(t)* (DOP, Degree Of Parallelism). Графічне зображення параметра *D(t)* як функції часу називають *профілем паралелізму програми*.

Типовий профіль паралелізму для алгоритму декомпозиції (divide-and-conquer algorithm) показаний на рис. 1.

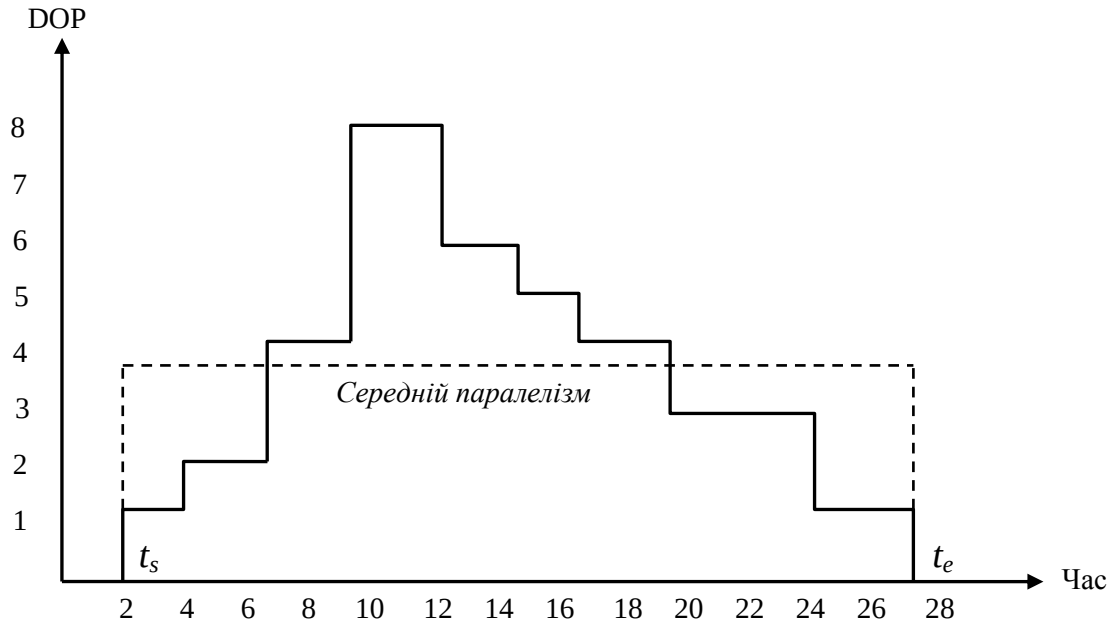

Рисунок 1 - Профіль паралелізму

Зміни в рівні завантаження процесорів за час спостереження залежать від багатьох чинників (алгоритму, доступних ресурсів, ступеня оптимізації, що забезпечується компілятором, і т.д.).

Надалі виходитимемо з наступних припущень: система складається з *n* гомогенних процесорів; максимальний паралелізм у профілі рівний *m* і, в ідеальному випадку, *n* >> *m*. Продуктивність E одиничного процесора системи виражається в одиницях швидкості обчислень (кількість операцій в одиницю часу) і не враховує витрат, пов'язаних із зверненням до пам'яті і пересилкою даних. Якщо за спостережуваний період завантажені *i* процесорів, то *D = i*.

Загальний об'єм обчислювальної роботи *W* (команд або обчислень), виконаної починаючи із стартового моменту t_s до моменту завершення t_e, пропорційний площі під кривою профілю паралелізму:

Середній паралелізм A визначається як

$$A = \frac{1}{t_e - t_s} \int_{t_s}^{t_e} D(t)\, dt \ . \tag{13}$$

У дискретній формі це можна записати так:

$$A = \frac{\sum_{i=1}^{m} i \cdot t_i}{\sum_{i=1}^{m} t_i} \ . \tag{14}$$

Прискорення (speedup) або, точніше, середнє прискорення за рахунок паралельного виконання програми – це відношення часу, потрібного для виконання якнайкращого з послідовних алгоритмів на одному процесорі, і часу паралельного обчисленню на *n* процесорах.

Без урахування комунікаційних витрат прискорення *S(n)* визначається як

$$S(n) = \frac{T(1)}{T(n)} \ . \tag{15}$$

Як правило, прискорення задовольняє умову $S(n) \le n$.

Ефективність (efficiency) *n*-процесорної системи – це прискорення на один процесор, що визначається виразом

$$E(n) = \frac{S(n)}{n} = \frac{T(1)}{n \cdot T(n)} \ . \tag{16}$$

Ефективність зазвичай відповідає умові $1/n \le E(n) \le n$.

Залежно від кількості процесорів у системі розрізняють три можливі варіанти прискорень.

Власне прискорення визначається шляхом реалізації паралельного алгоритму на одному процесорі.

Якщо прискорення, досягнуте на *n* процесорах, дорівнює *n*, то говорять, що алгоритм показує *лінійне прискорення*.

У виняткових ситуаціях прискорення *S(n)* може бути більше, ніж *n*. *У* цих випадках іноді застосовують термін *суперлінійне прискорення*.

До чинників, що обмежують прискорення, слід віднести:

- Програмні витрати. Навіть якщо послідовні і паралельні алгоритми виконують одні і ті ж обчислення, паралельним алгоритмам властиві додаткові програмні витрати – додаткові індексні обчислення, що неминуче виникають через декомпозицію даних і розподіл їх по процесорах; різні види облікових операцій, потрібні в паралельних алгоритмах, але відсутні в алгоритмах послідовних.

• *Витрати через дисбаланс завантаження процесорів.* Між точками синхронізації кожен з процесорів повинен бути завантажений однаковим об'ємом роботи, інакше частина процесорів чекатиме, поки останні завершать свої операції. Ця ситуація відома як дисбаланс завантаження. Таким чином, прискорення обмежується найбільш повільним з процесорів.

• *Комунікаційні витрати.* Якщо прийняти, що обмін інформацією та обчислення можуть перекриватися, то будь-які комунікації між процесорами знижують прискорення. В плані комунікаційних витрат важливий рівень гранулярності, що визначає об'єм обчислювальної роботи, яка виконується між комунікаційними фазами алгоритму. Для зменшення комунікаційних витрат вигідніше, щоб обчислювальні гранули були достатньо великими і частка комунікацій була менша.

3 Закон Амдала

В ідеальному випадку можна вважати, що обчислювальна ситема з n процесорів могла б прискорити обчислення в n разів. Реально досягти такого показника з ряду причин не вдається. Головна з цих причин полягає в неможливості повного розпаралелювання жодного із завдань. Як правило, в кожній програмі є фрагмент коду, який принципово повинен виконуватися послідовно і лише одним з процесорів. Це може бути частина програми, яка відповідає за запуск задачі і розподіл розпаралеленого по процесорах коду, або фрагмент програми, що забезпечує операції вводу/виводу. Можна привести і інші приклади, але головне полягає в тому, що про повне розпаралелювання задачі говорити не доводиться. Відомі проблеми виникають і з тією частиною задачі, яка піддається розпаралелюванню. Тут ідеальним був би варіант, коли паралельні гілки програми постійно завантажували б усі процесори системи, причому так, щоб навантаження на кожен процесор було однакове. На жаль, обидві ці умови на практиці важко реалізовуються. Таким чином, орієнтуючись на паралельну ОС, необхідно чітко усвідомлювати, що добитися прямо пропорційного числу процесорів збільшення продуктивності не вдасться, і, природно, встає питання про те, на яке реальне прискорення можна розраховувати. Відповідь на це питання в якійсь мірі дає закон Амдала.

Джин Амдал (Gene Amdahl) – один з розробників всесвітньо відомої системи IBM 360. У своїй роботі, опублікованій у 1967 році, він запропонував формулу, яка відображає залежність прискорення обчислень, що досягається на багатопроцесорній ОС, від числа процесорів і співвідношення між послідовною і паралельною частинами програми. Показником скорочення часу обчислень служить така метрика, як «*прискорення*». Нагадаємо, що прискорення S – це відношення часу T_s, що витрачається на проведення обчислень на однопроцесорній ОС (у варіанті якнайкращого послідовного алгоритму), до часу T_p, розв'язання тієї ж задачі на паралельній системі (у разі використання якнайкращого паралельного алгоритму):

$$S = \frac{T_s}{T_p} . \quad (17)$$

Обмеження щодо алгоритмів розв'язання задачі зроблені, щоб підкреслити той факт, що для послідовного і паралельного вирішення кращими можуть бути різні реалізації, а оцінюючи прискорення, необхідно виходити саме з якнайкращих алгоритмів.

Проблема розглядалася Амдалом у наступній постановці. Перш за все, об'єм вирішуваної задачі із зміною числа процесорів, що беруть участь в її розв'язанні, залишається незмінним. Програмний код вирішуваної задачі складається з двох частин: послідовної і такої, що розпаралелює. Позначимо частку операцій, які повинні виконуватися послідовно одним з процесорів, через f, де $0 \le f \le 1$ (тут частку розуміють не по числу рядків коду, а по числу реально виконуваних операцій). Звідси частка, що приходиться на розпаралелювану частину програми, складе $1 - f$. Крайні випадки в значеннях f відповідають повністю паралельним ($f = 0$) і повністю послідовним ($f = 1$) програмам. Розпаралелювана частина програми рівномірно розподіляється по всіх процесорах.

З урахуванням приведеного формулювання маємо:

$$T_p = f \cdot T_s + \frac{(1-f)\cdot T_s}{n} . \quad (18)$$

У результаті отримуємо формулу Амдала, що виражає прискорення, яке може бути досягнуте на системі з n процесорів:

$$S = \frac{T_s}{T_p} = \frac{n}{1+(n-1)\cdot f} . \quad (19)$$

Формула виражає просту і таку, що володіє великою спільністю залежність.

Якщо спрямувати число процесорів до нескінченності, то в границі отримуємо:

$$\lim_{n\to\infty} S = \frac{1}{f} . \quad (20)$$

Це означає, що якщо в програмі 10% послідовних операцій (тобто $f = 0{,}1$), то, скільки б процесорів не використовувалося, прискорення роботи програми більш ніж вдесятеро ніяк не отримати, але і 10 – це теоретична верхня оцінка найкращого випадку, коли ніяких інших негативних чинників немає. Слід зазначити, що розпаралелювання веде до певних витрат, яких немає у разі послідовного виконання програми. Як приклад таких витрат можна згадати додаткові операції, пов'язані з розподілом програм по процесорах, обмін інформацією між процесорами і так далі.

4 Закон Густафсона

Відому частку оптимізму в оцінку, що дається законом Амдала, вносять дослідження, проведені Джоном Густафсоном з NASA Ames Research [25]. Вирішуючи на обчислювальній системі з 1024 процесорів три великі завдання, для яких частка послідовного коду f лежала в межах від 0,4 до 0,8%, він набув значень прискорення в порівнянні з однопроцесорним варіантом, рівні відповідно 1021, 1020 і 1016. Згідно з законом Амдала для даного числа процесорів і діапазону f, прискорення не повинне було перевищити величини порядку 201. Намагаючись пояснити це явище, Густафсон прийшов до висновку, що причина криється в початковій передумові, що лежить в основі закону Амдала: збільшення числа процесорів не супроводжується збільшенням об'єму вирішуваного завдання. Реальна ж поведінка користувачів істотно відрізняється від такого уявлення. Зазвичай, отримуючи в своє розпорядження могутнішу систему, користувач не прагне скоротити час обчислень, а, зберігаючи його практично незмінним, старається пропорційно потужності ОС збільшити об'єм вирішуваного завдання. І тут виявляється, що нарощування загального об'єму програми стосується головним чином частини програми, що розпаралелює. Це веде до скорочення значення f. Прикладом може служити вирішення диференціального рівняння в окремих похідних. Якщо частка послідовного коду складає 10% для 1000 вузлових точок, то для 100 000 точок частка послідовного коду знизиться до 0,1%. Тобто, залишаючись практично незмінною, послідовна частина в загальному об'ємі збільшеної програми має вже меншу питому вагу.

Було відмічено, що в першому наближенні об'єм роботи, яка може бути проведена паралельно, зростає лінійно із зростанням числа процесорів у системі. Для того щоб оцінити можливість прискорення обчислень, коли об'єм останніх збільшується із зростанням кількості процесорів у системі (коли постійний загальний час обчислень), Густафсон рекомендує використовувати вираз, запропонований Е. Барсисом (E. Barsis):

$$S = \frac{T_s}{T_p} = \frac{f \cdot T_s + n \cdot (1 - f) \cdot T_s}{f \cdot T_s + (1 - f) \cdot T_s} = n + (1 - n) \cdot f \ . \qquad (21)$$

Даний вираз відомий як *закон масштабованого прискорення* або *закон Густафсона* (іноді його називають також законом Густафсона-Барсиса). На закінчення відзначимо, що закон Густафсона не протирічить закону Амдала. Відмінність полягає лише у формі утилізації додаткової потужності ОС, що виникає у разі збільшення числа процесорів.