Змістовний модуль:

СУЧАСНІ ПРОЦЕСОРИ ТА ПАРАЛЕЛІЗАЦІЯ ОБЧИСЛЕНЬ

Розділ: КРИТЕРІЇ ТА ПОКАЗНИКИ ПРОГРАМ

ЛЕКЦІЯ 3. РІВНІ ПАРАЛЕЛІЗМУ. КРИТЕРІЇ ТА ПОКАЗНИКИ ПРОГРАМ. АНАЛІТИЧНІ МЕТОДИ ОЦІНКИ

РІВНІ ПАРАЛЕЛІЗМУ. ПРОДОВЖЕННЯ

- Паралелізм на рівні бітів
- Паралелізм на рівні команд
- Паралелізм на рівні даних
- Паралелізм функцій
- Паралелізм програм
- Пам’ять та паралельне виконання

ТИПИ ПАРАЛЕЛІЗМУ. ПАРАЛЕЛІЗМ НА РІВНІ ДАНИХ. SIMD КОМАНДИ

Класи команд:

- **MMX** (**M**ulti**m**edia E**x**tensions)(64 біт, 2, 4, 8 елементів, цілі, регістри з плаваючою точкою).
- **3DNow!** (64 біт, 2 float, регістры з плаваючою точкою, горізонтальні операції)
- **SSE** (*Streaming SIMD Extensions*)(128 біт, int, double, float, 2-16 елементів, розширений набір операцій), *XMM0..XMM7/XMM16)32/64*
- **AVX** (**Advanced Vector Extensions**)(256 (512, 1024)), YMM0-YMM15 (ZMM0-ZMM15)

ТИПИ ПАРАЛЕЛІЗМУ. ПАРАЛЕЛІЗМ НА РІВНІ ЗАДАЧ

Паралелізм на рівні задач

Потоки **(**визначення**)**

Способи завдання паралелізму на рівні задач

- **Псевдо** паралелізм **(**операційна система, витіснення**)**
- **HYPER Threading (**операційна система, витіснення, якщо кількість потоків більше ніж 2**), додаткові регістри, PIC**
- **multi-core (**операційна система, витіснення, якщо кількість потоків більше кількості ядер**)**
- **many-core** (операційна система, спеціальні бібліотеки, наприклад, CUDA технологія)
- **Системи з розподіленою пам'яттю** (Системи з масовим паралелізмом або кластери) – розподілені операційні системи

СТРУКТУРА БАГАТОЯДЕРНОГО ПРОЦЕСОРУ

Процесорне ядро 1 Кеш L1	Процесорне ядро 2 Кеш L1	Процесорне ядро 3 Кеш L1	Процесорне ядро 4 Кеш L1
Кеш L2 (512 КБайт)	Кеш L2 (512 КБайт)	Кеш L2 (512 КБайт)	Кеш L2 (512 КБайт)
Кеш L3 (2 / 4 Мбайт) **Рекомендації по використанню пам'яті**			

ПИТАННЯ ДЛЯ ВИВЧЕННЯ

- Поняття алгоритму.
- Обчислювальна складність.
- Показники для оцінки паралельних алгоритмів.
- Закон Амдаля.
- Закон Густафсона.
- Порівняння законів Амдаля та Густафсона.
- Приклад.
- Недоліки аналітичних методів

ПОНЯТТЯ АЛГОРИТМУ

- **Алгоритм** – **Алгоритмом** називається точний та зрозумілий опис **послідовних** дій, які необхідні для вирішення поставленої задачі.

- **Визначення Т. Кормену (Алгоритми, побудова та аналіз)**. Алгоритм – це формально визначена обчислювальна процедура, яка отримує вихідні дані (input), які називаються також входом алгоритму або його аргументом, та видає результат обчислення на вихід (output).

Ми будемо розглядати **алгоритм**, як «чорна скриня», яка виконує перетворення вхідних даних в вихідні, ці перетворення можуть бути послідовними, паралельними або змішаними.

ОБЧИСЛЮВАЛЬНА СКЛАДНІСТЬ

Розмірність задачі (n) - сукупність параметрів (їх кількість), які характеризують обсяг вихідних даних та визначають необхідні для виконання розрахунків ресурси (час, пам'ять).

Обчислювальна складність: часова, просторова.

Часова складність: - час для виконання алгоритму в залежності від його розмірності T (n).

Просторова складність : - необхідний обсяг пам'яті

M (n)

Далі по замовчанню:

Складність  **обчислювальна складність**

ОБЧИСЛЮВАЛЬНА СКЛАДНІСТЬ. СИМВОЛИ O,Ω,Θ

Визначити обчислювальну складність алгоритму пошуку максимального числа.

```
int max = x [0];
for(int i=1; i < n; i++) {
        if (x [i] > max)
                max = x [i]
}
```

	cmp	Mov	Σ
Найгірший випадок (O (n))	(n-1)	1 + n-1 = n	2n-1
Найкращий випадок (**Ω(**n**)**)	n-1	1	n
Середній варіант(**Θ**(n))	n-1	n/2	n+n/2-1

ПОКАЗНИКИ ДЛЯ ОЦІНКИ АЛГОРИТМІВ

- ***Прискорення Sp(n)*** для паралельного алгоритму розмірністю *n* визначається відношенням часової складності послідовного (*T1 (n)*) та паралельного алгоритмів для *p* процесорів (*Tp(n)*): *Sp(n) = T1 (n)/ Tp(n).*

  Ідеал! *Sp(n) = n.*

  А може бути *Sp(n) > n?* **Так!!!**
- ***Ефективність Ep(n)*** для паралельного алгоритму розмірністю *n* визначається прискоренням цього алгоритму відносно одного процесора:

  Ep(n) = Sp(n) /p = T1 (n)/ (p * Tp(n))

  Ідеал! *Ep(n)* = 1.

  Теоретично можливе значення ефективності, більше одиниці (Чому???).
- ***Вартість обчислень*** *(Cp)*. Чим більша частина алгоритму виконується паралельно, тим менше його часова складність (*Tp(n)*). Чим більше процесорів (*p*) використовується, тим дорожче обчислювальна система. Вартість обчислень оцінюється їх добутком, тобто: ***Cp = p * Tp(n).***

ЗАКОН AMDAHL

Закон визначає теоретичне значення прискорення без урахування накладних витрат, пов'язаних з паралельними обчисленнями.

Позначимо час виконання програми в послідовному режимі 1.

Нехай β – частина програми, що винна виконуватися послідовно, тоді 1 - β – частина програми, що може виконуватися паралельно.

Нехай кількість процесорів дорівнює *p,* всі процесори рівномірно завантажені при виконанні паралельної частини програми й накладними витратами можна зневажити.

Тоді прискорення визначається формулою (*перше формулювання закону Амдаля)*.

$$Sp = \frac{1}{\beta + \frac{1-\beta}{p}} = \frac{p}{\beta * p + 1 - \beta}$$

ЗАКОН AMDAHL

Другий закон Амдаля визначає прискорення та ефективність, якщо кількість процесорів збільшується до нескінченності.

Як видно з наведених формул максимальне прискорення 1/β

$$Sp = \frac{1}{\beta + \frac{1-\beta}{p}}$$

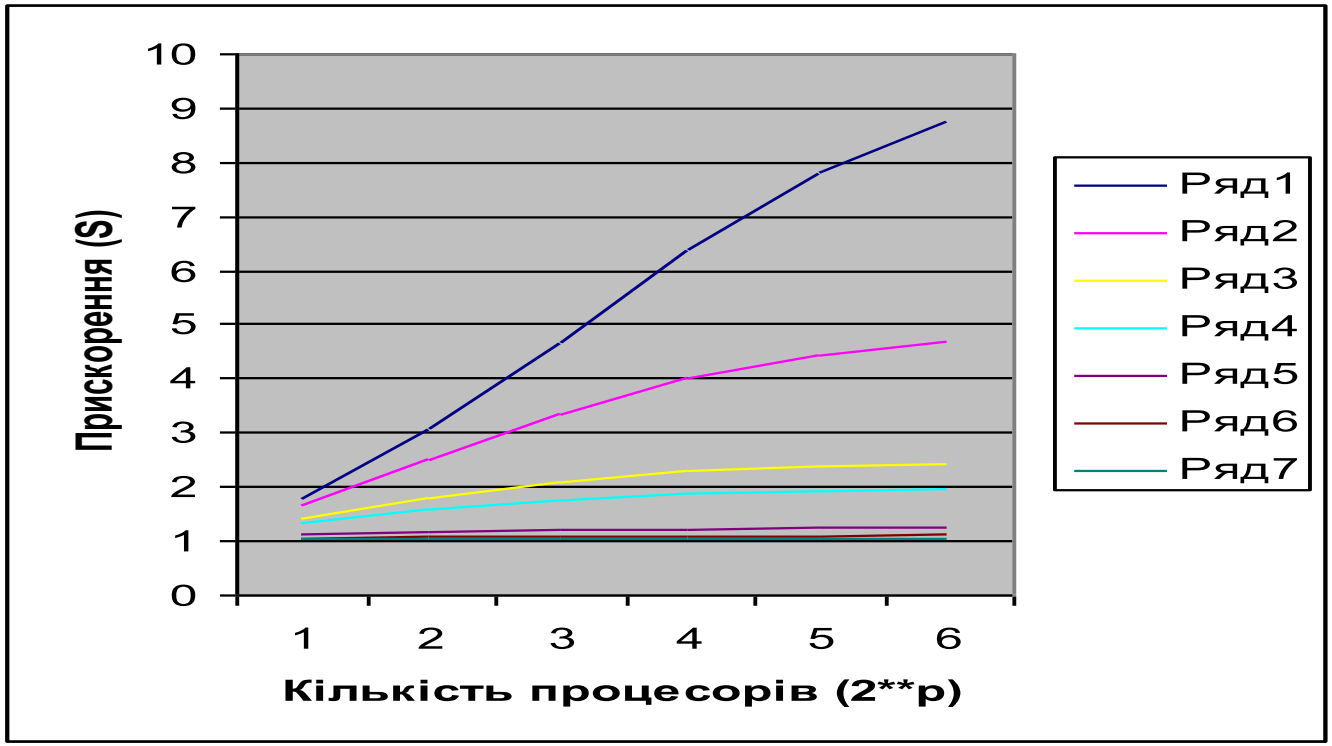
$$\lim_{p\to\infty} \frac{1-\beta}{p} \to 0;$$

$$\lim_{p\to\infty} S_p \to \frac{1}{\beta};$$

$$E_p = 0;$$

ЗАКОН AMDAHL

Ряд 1. β= 0.1.
Ряд 2. β= 0.2.
Ряд 3. β= 0.4.
Ряд 4. β= 0.5.
Ряд 5. β= 0.8.
Ряд 6. β= 0.9.
Ряд 7. β= 0.95



ЗАКОН AMDAHL

Приклади.

$$Sp = \frac{1}{\beta + \frac{1-\beta}{p}}$$

Приклад 1. Хай послідовна частина коду займає 90% коду. Визначити прискорення, якщо паралельна частина прискорена в 10 разів.

Приклад 2. Визначити кількість процесорів, яка необхідна для одержання прискорення в 4 рази, якщо половина коду виконується паралельно (β = 0.5).

Приклад 3. Визначити кількість процесорів, що необхідна для одержання прискорення в 4 рази, якщо 90% коду виконується паралельно (β = 0.1).

Недолік закону: не завжди відповідає практичним результатам!

ЗАКОН AMDAHL

Приклади.

Приклад 1. 100/91 ≈1.1, тільки на 10 %!!!

Приклад 2. Визначити кількість процесорів, яка необхідна для одержання прискорення в 4 рази, якщо половина коду виконується паралельно (β = 0.5).

Спочатку визначимо максимальне значення прискорення (2 закон Амдаля): 1/0.5 = 2 < 4. Необхідне значення прискорення перевищує максимально можливе, тому дане значення досягти неможливо.

Приклад 3. Визначити кількість процесорів, що необхідна для одержання прискорення в 4 рази, якщо 90% коду виконується паралельно (β = 0.1).

Визначимо максимальне значення прискорення 1/ (0.1) = 10>4.

Необхідне значення прискорення не перевищує максимально можливе, тому визначимо кількість процесорів *p* згідно з 1 законом Амдаля:

4 = p/(0.1p + 0.9); 0.4p+3.6 = p; 0.6p = 3.6;

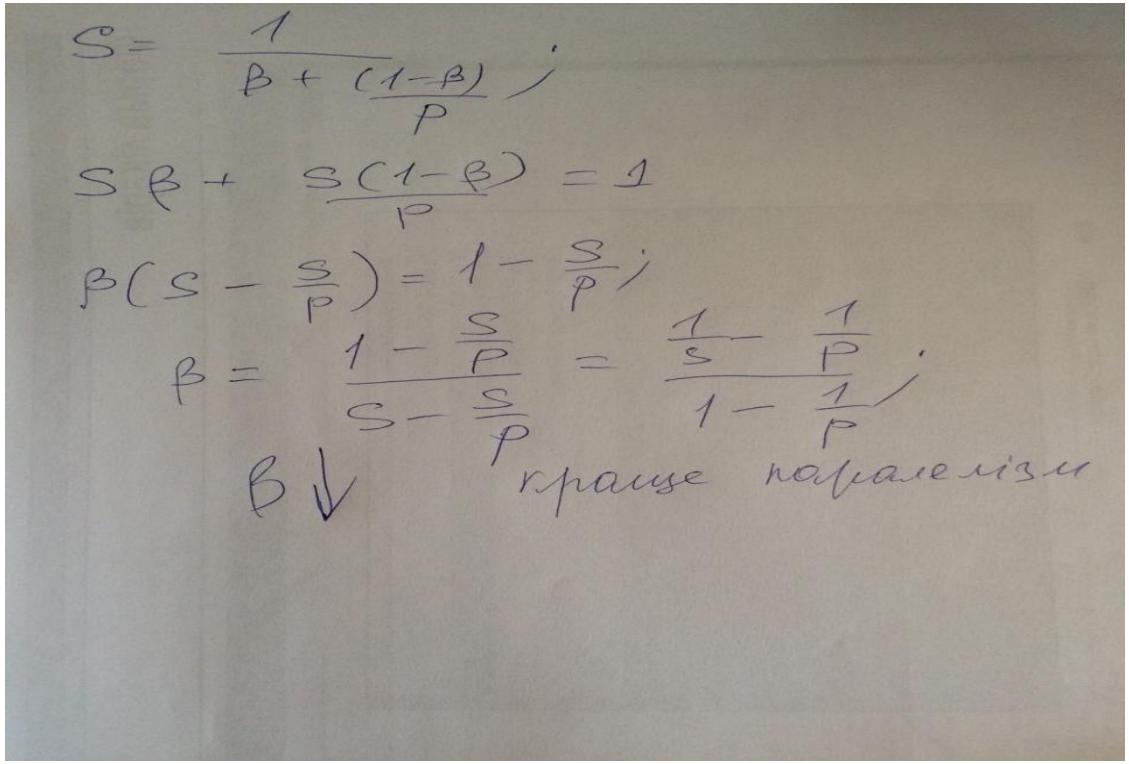
P = 6; - достатньо 6 процесорів!

Недолік закону: не завжди відповідає практичним результатам!

ЗАКОН АМДАЛЯ І KARP–FLATT MIPA

Фактично визначити значення β важко.

Визначимо його математично



ВИЗНАЧЕННЯ НАКЛАДНИХ ВИТРАТ

Хай відомо T_1 – час для послідовних обчислень, T_p- час для паралельних обчислень

Накладні витрати: $T(0) = T(p) * p – T(1)$

Тоді:

$T(p) = (T(0) + T(1) / p$

$S = T(1) / (T(0) + T(1) / p) = T(1)\, p/(T(0) + T(1)$

$E = S/p = T(1) /(T(0) + T(1)) = 1/ (1 + T(0) / T(1))$

З іншого боку:

$T(0) / T(1) = 1/E-1 = (1-E)/E$

Для отримання заданої ефективності T_0 не може перевищувати заданого значення.

Приклад.

Хай необхідно обчислити x[0]+ x [1] + … + x[n-1]

$T(1) = n$; $T(p) = n/p + p$; $T(0) = (n/p + p)*p – T(1) = p^2$; $S = pn/(n+ p^2)$;

$E = n / p^2 + 1$

ЗАКОН ГУСТАФСОНА (GUSTAFSON)

Визначимо обсяг робіт (кількість команд), які можна виконати у випадку послідовної й паралельної обробки.

Значення відносини цих обсягів робіт визначає *масштабоване* (з урахуванням кількості процесорів) *прискорення*.

Нехай β - частина коду, яку потрібно виконувати послідовно. Тоді паралельна частина становить 1 - β.

Нехай у послідовному режимі виконується *Vs* команд. Тоді в паралельному режимі за цей же час буде виконане *Vp* = β **Vs* + (1 - β)**Vs* * p.

Прискорення в цьому випадку дорівнює:

$V_p/V_s = \beta + (1 - \beta)*p$.

Це і є оцінка **Густафсона – Барсиса.**

ЗАКОН ГУСТАФСОНА (GUSTAFSON)

Приклади.

Визначити прискорення згідно закону Густафсона. Порівняти отримані результати з результатами по закону Амдаля.

S = β + (1 - β)*p.

$$Sp = \frac{1}{\beta + \frac{1-\beta}{p}}$$

Приклад 1. Визначити кількість процесорів, необхідну для одержання прискорення в 4 рази, якщо половина коду виконується паралельно (β = 0.5):

Приклад 2. Визначити кількість процесорів, необхідну для одержання прискорення в 4 рази, якщо 90% коду виконується паралельно (β = 0.1):

ЗАКОН ГУСТАФСОНА (GUSTAFSON)

Приклад 1. Визначити кількість процесорів, необхідну для одержання прискорення в 4 рази, якщо половина коду виконується паралельно (β = 0.5):

$4 = 0.5 + 0.5 * p$; $p = 7$.

Відповідь: 7 процесорів (по оцінці Амдаля таке прискорення неможливо).

Приклад 2. Визначити кількість процесорів, необхідну для одержання прискорення в 4 рази, якщо 90% коду виконується паралельно (β = 0.1): $4 = 0.1 + 0.9 * p$;

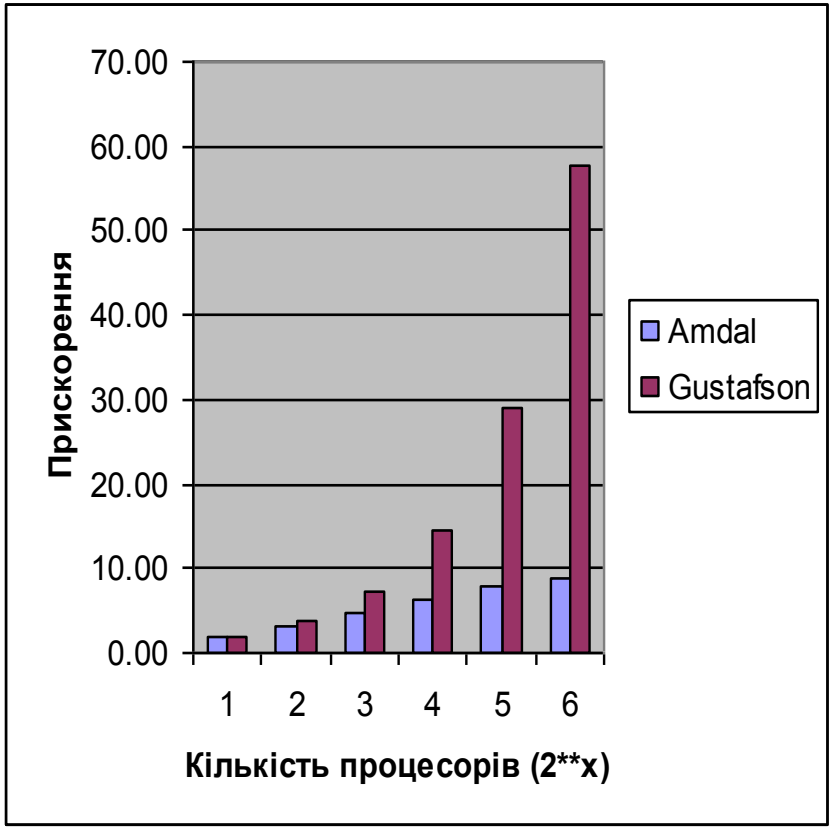
$p = 5$.

Відповідь: 5 процесорів (по оцінці Амдаля 6 процесорів). **Які оцінки вірні?**

ЗАКОНИ АМДАЛЯ, ТА ГУСТАФСОНА

- На діаграмі задані результати обчислення прискорення для β=0.1 для закону Амдаля та Густафсона для 2,4, 8, 16, 32, 64 процесорів.
- Різниця тим більше, чим більше кількість процесорів
- **Чому така різниця, який закон вірний?**

Далі покажемо, що обидва законі вірні!!!



ЗАКОНИ АМДАЛЯ, ТА ГУСТАФСОНА

Приклад.

Нехай у програмі в послідовному режимі виконується 100 команд, з яких 90 команд можна виконувати паралельно й 10 команд тільки послідовно. У цьому випадку β = 0.1.

Нехай використовується p = 3 процесори.

Закон Амдаля:

Sp(A) = 1/(0.1 + 0.9/3) = 2.5.

Закон Густафсона:

Sp(G) = 0.1 + 0.9 * 3 = 2.8.

2.5 != 2.8. Де помилка при отриманні результатів? – **Дивись далі!**

ЗАКОНИ АМДАЛЯ, ТА ГУСТАФСОНА

Закон **Густафсона:** прискорення - відношенням кількості команд.
У послідовному режимі буде виконано 100 команд, у паралельному режимі необхідно виконати 10 + 90/3 = 40 команд, прискорення Густафсона дорівнює Sp’(G) = 100/40 = 2.5 = Sp(A), тобто Sp’(G) = Sp(A).
Результат співпадає з результатом, отриманим за допомогою закону Амдаля.
Зворотний доказ.
Визначимо кількість команд, що можуть бути виконані в паралельному режимі за той же час, за який виконується 100 команд у послідовному режимі. Ця кількість команд дорівнює 10 + 90 * 3 = 280.
Таким чином, для 10 команд послідовної частини одержуємо 270 команд у паралельній частині, тобто β =10/280 або 1/28.
Для β = 1/28 за законом Амдаля одержуємо прискорення:
Sp’(A) = 3 (3/28 + 1 -1/28)= (3*28)/(2 + 28)= 2.8 = Sp(G), тобто:
Sp’(A) = Sp(G).
Висновок!
Закон Амдаля використовувати для постійного навантаження!!!
Закон Густафсона використовувати, якщо навантаження може збільшуватись при збільшенні кількості процесорів!!!

ПРИКЛАД

Визначити прискорення, ефективність та вартість для обчислення значення багаточлена:

$Pn(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_i x^i + \ldots + a_1 x^1 + a_0$.

Необмежений паралелізм. n процесорів.

Послідовний алгоритм.

Ефективний послідовний алгоритм (схема Горнера) вимагає для обчислення 2*n операцій. Припустимо, що час обчислень пропорційний кількості операцій. Тоді значення показників :

$Sp(n) = 2*n /(2*n) = 1$

$Ep(n) = 1$

$Cp(n) = (T(p) *p) = 2 * n.$

ПРИКЛАД

Паралельний алгоритм 1.

Кількість процесорів дорівнює *n*.

a_nx	$a_{n-1}x$	…	a_1x	*(n процесорів)*
a_nx^2	$a_{n-1}x^2$	…	$a_1x + a_0$	*(n процесорів)*
a_nx^3	$a_{n-1}x^3$	…	$a_2x^2 + a_1x + a_0$	*(n - 1 процесорів)*
…				
a_nx^n	$a_{n-1}x^{n-1} + \ldots + a_0$			*(2процесора)*
Y				
				(1 процесор)

Треба *n + 1* кроків і *n* процесорів для обчислення значення багаточлена:

$Sp(n) = 2*n /(n + 1)$

$Ep(n) = 2/(n + 1)$

$Cp(n) = n*(n + 1)$

Недолік: Нерівномірний розподіл процесорів між окремими кроками.

ПРИКЛАД

Паралельне обчислення. Алгоритм 2.

x^2		$r = a_1x$	*2 процесора*
x^3	a_2x^2	$r \mathrel{+}= a_0$	*3 процесора*
x^4	a_3x^3	$r \mathrel{+}= a_2x^2$	
x^5	a_4x^4	$r \mathrel{+}= a_3x^3$	
…			
x^n	$a_{n-1}x^{n-1}$	$r \mathrel{+}= a_{n-2}x^{n-2}$	
	a_nx^n	$r \mathrel{+}= a_{n-1}x^{n-2}$	
		$r \mathrel{+}= a_nx^n$	

Треба *n* + *1* кроків і 3 процесори для обчислення значення багаточлена:

*Sp(n) = 2*n /(n + 1)*

*Ep(n) = 2/3*n /(n + 1)*

*Cp(n) = 3**(n + 1)

ПРИКЛАД

Показник (n =1000)	Схема Горнера	Алгоритм 1	Алгоритм 2
Sp(n)	1	2n/(n+1) 1.998	2n/(n+1) 1.998
Ep(n)	1	2/(n+1) 0. 001996	2n/(3(n+1)) 0.666
Cp(n)	2n 2000	n(n+1) 1001000	3(n+1) 3003

ПРИКЛАД

Порівняння алгоритмів для обчислення значення поліному

- Прискорення за рахунок паралельного виконання однакові для обох алгоритмів, але при цьому Алгоритм 2 більш ефективний і його вартість набагато менше, ніж для Алгоритму 1.
- при збільшенні кількості ядер (n > 3) прискорення для алгоритму 2 не змінюється, а показники *Ep(n)* і *Cp(n)* погіршуються, але при цьому залишаються не гірше, ніж для Алгоритму 1;
- алгоритм 2 потребує накопичення суми, що може погіршите його характеристики.

А чи можна побудувати алгоритм, ефективність якого визначається кількістю процесорів?

Спробуємо!!!

ПРИКЛАД

Алгоритм 3 обчислення багаточлену.

Нехай у нас є p ядерний процесор ($p < n$). Нехай для простоти n кратне $p-1$. Якщо це не так, то старші коефіцієнти можна доповнити нулями.

Розділимо багаточлен на $m = p-1$ порцій однакового розміру. У кожну порцію входять суміжні елементи багаточлена, розмір кожної порції дорівнює

k = n / (p - 1). p = 5

$Pn(x) = a_{99}x^{99} + a_{98}x^{98} + \ldots + a_1x^1 + a_0$

Представимо наш багаточлен у вигляді:

$Pn(x) = A_{m-1}x^{(m-1)k} + A_{m-2}x^{(m-2)k} + \ldots + A_1x^k + A_0$, $Pn(x) = A_3x^{75} + A_2x^{50} + A_1x^{25} + A_0$

де:

$m = p - 1$ (кількість порцій) *4,*

$k = n / (p - 1)$ – Кількість елементів в порції *25*

$A_0 = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \ldots + a_1x^1 + a_0$, $A_0 = a_{24}x^{24} + a_{23}x^{23} + \ldots + a_1x^1 + a_0$,

$A_1 = a_{2k-1}x^{k-1} + a_{2k-2}x^{k-2} + \ldots + a_{k+1}x^1 + a_k$, $A_1 = a_{49}x^{24} + a_{48}x^{23} + \ldots + a_{26}x^1 + a_{25}$,

... ...

$A_{m-1} = a_{mk-1}x^{k-1} + a_{mk-2}x^{k-2} + \ldots + a_{(m-1)k+1}x^1 + a_{(m-1)k}$ $A_3 = a_{99}x^{99} + a_{98}x^{98} + \ldots + a_{76}x^1 + a_{75}$,

$A_0, A_1, \ldots A_{m-1}$ - «цифри» в системі числення, рівної x^k.

ПРИКЛАД

Схема паралельних обчислень для алгоритму 3:

Крок 1	Крок 2	Крок 3
П1: $x^k, x^{2k}, \ldots,$	$A_1 * x^k$	A_0+A_1
П2: A_0 ,	$A_2 * x^{2k}$	A_2+A_3
П3: A_1. …		…
…		
П p-1: A_{m-1},	$A_{m-1} * x^{(m-1)k}$	

2k 50 1 log 2 m (4)

200/54 - прискорення

ПРИКЛАД

Крок 1. Визначення кількості операцій для алгоритму 3.

Кількість операцій для обчислення x^k. Навіть в раз звичайного множення. K-1 операцій.

Кількість операцій для обчислення $A_0, A_1, \ldots, A_{m-1}$ – $2 * k$ **операцій (схема Горнера)**.

Загальна кількість операцій для кроку 1:

R(кроку 1) =2k

ПРИКЛАД

Крок 2. Обчислення $A_{m-1}x^{(m-1)k}$ (m =1, k) – 1 операція

Крок 3. Обчислення суми.

$S = s_1+s_2+\ldots+sn = ((s_1+s_2) + (s_3+s_4))+\ldots$

$((s_{n-3}+s_{n-2}) + (s_{n-1}+s_n))$

Додаткові дужки показують порядок обчислень в паралельному режимі.

Кількість кроків для обчислення S дорівнює $\log_2 p$ (перевірте це!!!).

Загальна кількість операцій для Алгоритму 3

$T = 2k+1+\log_2 p \approx 2k=2n/(p-1)$

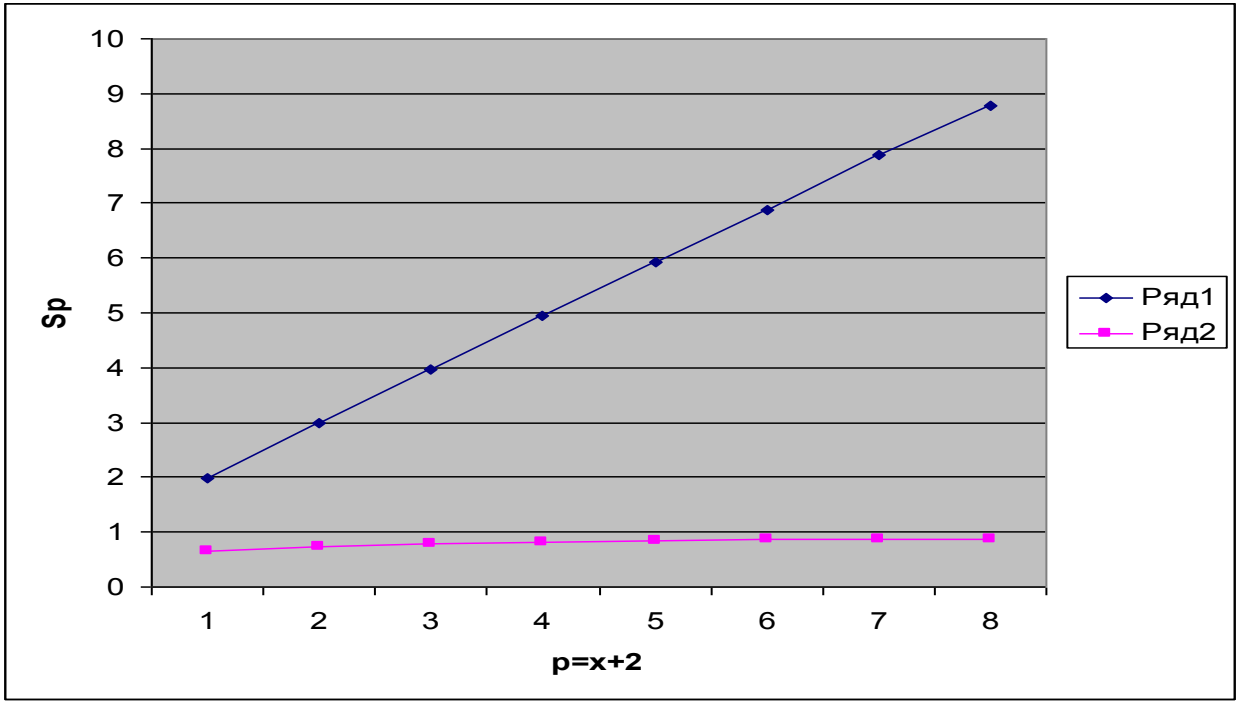
Показники:

$S(p)=2n/(2k+1+\log_2 p) = p-1$

$E(p)=S(p)/p =(p-1)/p;$ $C(p) = (p-1)*p$

ПРИКЛАД

Залежність основних характеристик алгоритму 3 від кількості процесорів при n = 1024, p>=3 (прискорення (ряд 1), ефективність (ряд 2))



НЕДОЛІКИ АНАЛІТИЧНИХ МЕТОДІВ

1. Виведення формули обчислення кількості операцій може бути дуже складним, особливо з урахуванням різних варіантів виконання програми (різні гілки програми мають різну обчислювальну складність).
2. Не враховує суперскалярну архітектуру процесора.
3. Не враховує складність операцій, так операція додавання й множення має різну складність, а враховуються у формулі як однакові.
4. Не враховує накладних витрат, пов'язаних з паралельним виконанням, а ці витрати можуть бути істотними, якщо паралельні гілки мають невелику обчислювальну складність.

ВИСНОВКИ

- Аналітичні методи оцінки дозволяють виконати оцінку алгоритму без його реалізації.
- Аналітичні методи не враховують накладні витрати, пов'язані з забезпеченням паралельного виконання.
- Аналітичні методи не враховують Суперскалярність процесорів та його конвеєр. Тому точність цих методів не висока.
- Для порівняння різних алгоритмів використовуються показники: прискорення, ефективність, вартість. Останній показник – інтегрований.
- Закони Амдаля та Густафсона дозволяють визначити прискорення для програми, яка складається з послідовної та паралельної частини.
- Закон Амдала використовується для визначення прискорення для випадку вирішення задачі фіксованого обсягу.
- Закон Густафсона використовується для випадку, коли навантаження збільшується зі збільшенням кількості процесорів, тому прискорення в цьому випадку називається масштабованим.
- Приклади алгоритмів показують, що самий ефективний алгоритм для послідовного методу є самим неефективним для паралельного.
- Далі будуть розглянуті експериментальні методи визначення складності програми

ПИТАННЯ ДЛЯ САМОСТІЙНОГО ВИВЧЕННЯ

1. Обчислювальні проблеми.
2. Машина Тьюрінга.
3. Класи складності

(http://www.machinelearning.ru/wiki/index.php?title=%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C)

МАТЕРІАЛИ ДЛЯ ЕКСПРЕС-КОНТРОЛЮ

1. Що таке часова складність алгоритму.
2. Які показники використовуються для оцінки паралельних алгоритмів.
3. Дайте визначення прискорення, ефективності й вартості паралельного алгоритму.
4. У якому випадку прискорення менше 1, більше 1? Чи можливі такі значення прискорення?
5. У якому випадку ефективність менше 1, більше 1, чи можливі ці значення?
6. Порівняйте два алгоритми, вартості яких у послідовному й паралельному режимі однакові. Чи має сенс використовувати паралельний алгоритм у цьому випадку?
7. Чому оцінки Амдаля й Густафсона можуть не збігатися для одних і тих же алгоритмів?
8. Докажіть, що в рівних умовах оцінки Амдаля й Густафсона завжди збігаються