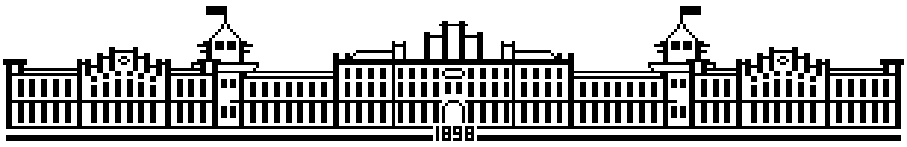


МІНІСТЕРСТВО НАУКИ І ОСВІТИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
Кафедра обчислювальної техніки

Лабораторна робота №2
з Комп’ютерних систем

Виконав студенти групи ІО-11
Ротенберг О.В
Бабак С.В
Номер варіанту - 4

Київ — 2014

Мета роботи:

Аналіз функціональності і ефективності мультипроцесорних систем (SMP) з основною пам'яттю.

Завдання:

Варіант №4: LU-розкладання матриці. α = 3, β = 5.

Короткі теоретичні відомості:

LU розклад матриці — представлення матриці у вигляді добутку нижньої трикутної матриці та верхньої трикутної матриці. Квадратна матриця A розміру n може бути представлена у вигляді - A = LU, де L та U — нижня та верхня трикутна матриця того ж розміру.

LDU розклад матриці — це представлення у вигляді A = LDU, де D — діагональна матриця, а L та U є одиничними трикутними матрицями, тобто, всі їх діагональні елементи рівні одиниці.

LUP розклад матриці — це представлення в формі A = LUP, де L та U — нижня та верхня трикутна матриця того ж розміру, а P — матриця перестановки.

Алгоритм є модифікованим методом Гауса і потребує $2n^3/3$ арифметичних операцій. Позначимо як lij, uij, aij елементи матриць L,U та A відповідно. З означення LU-розбиття $lij=0(j>i), uij=0(j<i), uii=1$. Очевидно, що

$$a_{ij}=\sum_{k=0}^{n-1} l_{ik}u_{kj}=\sum_{k=0}^{i-1} l_{ik}u_{kj}+l_{ii}u_{ij}+\sum_{k=i+1}^{n-1} l_{ik}u_{kj}=\sum_{k=0}^{i-1} l_{ik}u_{kj}+l_{ii}u_{ij}$$

$$a_{ij}=\sum_{k=0}^{n-1} l_{ik}u_{kj}=\sum_{k=0}^{j-1} l_{ik}u_{kj}+l_{ij}u_{jj}+\sum_{k=j+1}^{n-1} l_{ik}u_{kj}=\sum_{k=0}^{j-1} l_{ik}u_{kj}+l_{ij}u_{jj}$$,тут n — розмір мат. A.

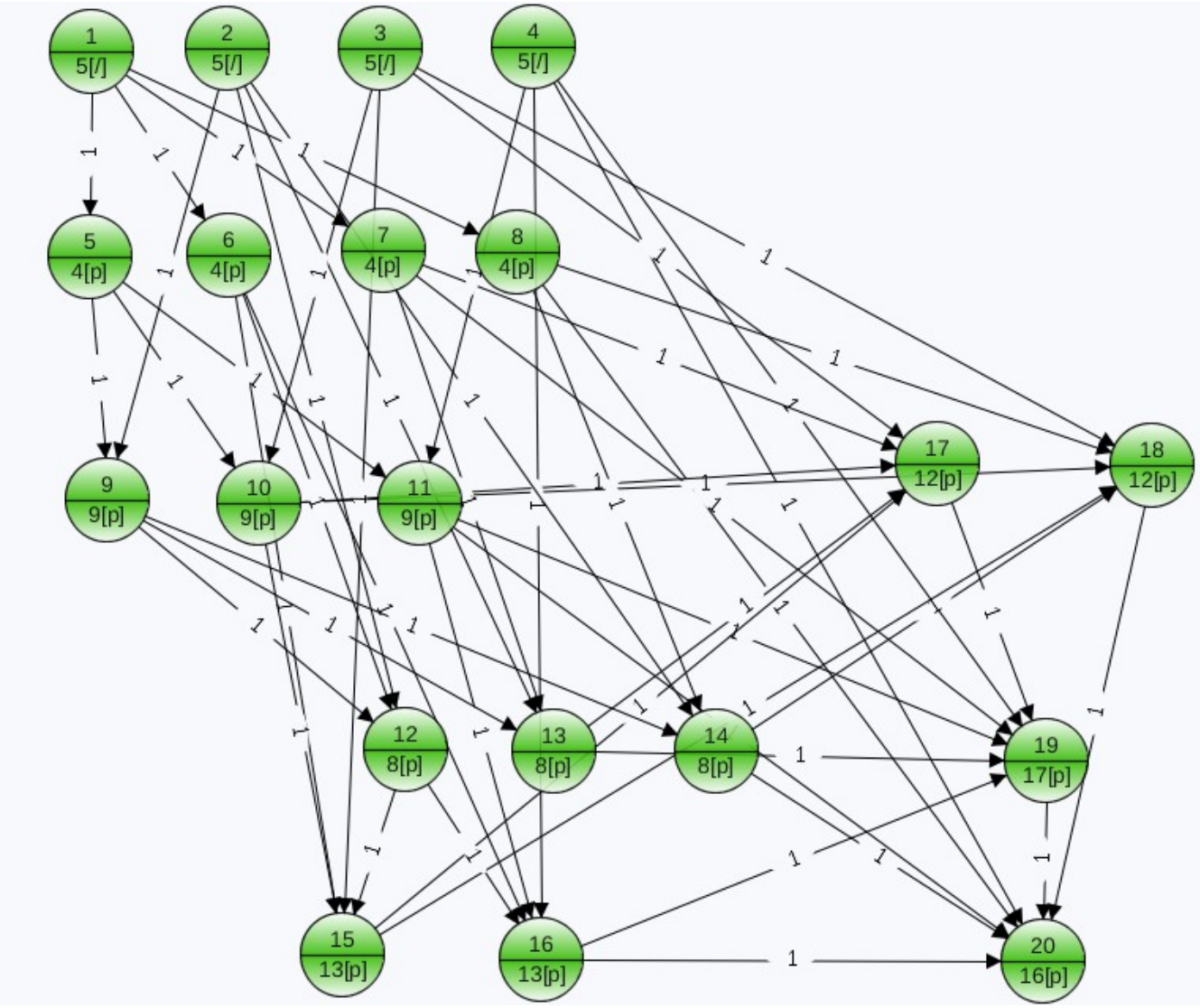
Звідки легко в явній формі отримати вирази для елементів матриць L та U:

$$l_{ij}=a_{ij}-\sum_{k=0}^{j-1} l_{ik}u_{kj}(i\le j) \quad \text{та} \quad u_{ij}=\frac{1}{l_{ii}}\left[a_{ij}-\sum_{k=0}^{i-1} l_{ik}u_{kj}\right](i<j) \quad .$$

Схема алгоритму:

№	Дія	Кількість тактів
1	l21 = a21/u11	5
2	l31 = a31/u11	5
3	l41 = a41/u11	5
4	l51 = a51/u11	5
5	u22=a22 – l21 * u12	4
6	u23=a23 – l21 * u13	4
7	u24=a24 – l21 * u14	4
8	u25=a25 – l21 * u15	4
9	l32=(a32 – l31 * u21) / u22	9
10	l42=(a42 – l41 * u21) / u22	9
11	l52=(a52 – l51 * u21) / u22	9
12	u33=a33 – (l31 * u13 + l32 * u23)	8
13	u34=a34 – (l31 * u14 + l32 * u24)	8
14	u35=a35 – (l31 * u15 + l32 * u25)	8
15	l43 = (a43-(l41*u13+l42*u23))/u33	13
16	l53 = (a53-(l51*u13+l52*u23))/u33	13
17	u44=a44-(l41*u14+l42*u24+l43*u34)	12
18	u45=a45-(l41*u15+l42*u25+l43*u35)	12
19	l54 = (a54-(l51*u14+l52*u24+l53*u34))/u44	17
20	u55=a55-(l51*u15+l52*u25+l53*u35+l54*u45)	16

Ярусно-паралельна форма алгоритму:



Результати:

	2 банк	3 банка	4 банка
2 процесора			
t	143	126	126
$k_{прискорення}$	1.36	1.54	1.54
$k_{ефективності}$	0.68	0.77	0.77
3 процесора			
t	121	117	110
$k_{прискорення}$	1.6	1.66	1.75
$k_{ефективності}$	0.53	0.55	0.58
4 процесора			
t	106	106	105
$k_{прискорення}$	1.83	1.83	1.85
$k_{ефективності}$	0.46	0.46	0.46

Висновок: у даній лабораторній роботі ми побудували ярусно-паралельну форму алгоритму LU-розкладу матриці і виконали задачу потактового розподілу обчислювальних гілок алгоритму по процесорам. Результати перевірки на програмній моделі показали, що найбільший коефіцієнт прискорення, який дорівнює 1.85 ми отримуємо на SMP-системі, яка має 4 процесора і 4 банки пам'яті. В результаті виконання даної лабораторної роботи ми можемо зробити висновок, що час виконання завдання, частини якого можна розпаралелити, можна зменшити. Цього можна досягти, збільшуючи кількість процесорів. Збільшення їх кількості має сенс до максимального ступеня ярусу в алгоритмі. При цьому буде збільшуватися коефіцієнт прискорення. Подальше збільшення кількості процесорів не має сенсу, так як коефіцієнт прискорення перестане рости, а частина процесорів буде простоювати. Однак зростання коефіцієнта прискорення не означає зростання коефіцієнта ефективності, так як деякі яруси алгоритму можуть мати ступінь менше кількості процесорів, і частина процесора не будуть задіяні. Кількість банків пам'яті варто наблизити до кількості процесорів, щоб уникнути затримок через конфлікти доступу до пам'яті. Наприклад, при 2-х банках пам’яті та ширині ярусу 4, ми зможемо зчитати дані тільки з 2-х процессорів, а з інших 2-х тільки на наступному такті. Конфліктна ситуація виникає при доступі до загального ресурсу, наприклад, процесам No5-8 необхідні дані з процесу No1, а так як система з загальною пам’яттю, то виникає конфліктна ситуація, і зчитування кожного наступного процесу проходить на такт пізніше.