1 Умножение матриц

1.1 Постановка задачи

Реализация алгоритма умножения двух матриц с использованием POSIX threads: необходимо вычислить произведение двух квадратных матриц A и B размера n: в результате чего получается матрица $C = A \times B$, $c_{i,j} = \sum_{k=1}^{n} a_{i,k} \cdot b_{k,j}$, где $c_{i,j}$ – элементы матрицы C.

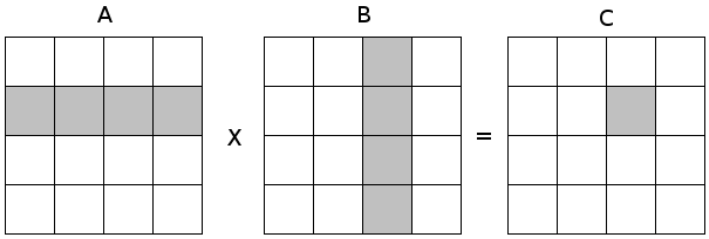
1.2 Подход к распараллеливанию

1.2.1 Простая реазиация

Статически распределяем задачу между потоками: при запуске каждый поток в качестве аргумента принимает интервал строк $[start_i; end_i)$ ($i = 0 \dots T$-1, $end_i = start_{i+1}$ для $i = 0 \dots T$-2, где T - количество потоков) результирующей матрицы, которые ему надо посчитать. Т. к. все потоки вычисляют различные элементы матрицы C, то синхронизация не требуется.

1.2.2 Блочное умножение

Матрицы A, B, C разбиваются на одинаковые квадратные (по количеству элементов) блоки размера, кратного размеру матриц. Пусть размер блока равен $bsize$, обозначим $bcount = \frac{n}{bsize}$, тогда каждя матрица разбита на $bcount \cdot bcount$ блоков. Блок (i, j) матрицы C вычисляется: $BC_{i,j} = \sum_{k=1}^{bcount} BA_{i,k} \cdot BB_{k,j}$, где умножение блоков $BA_{i,k} \times BB_{k,j} = BT$, элементы BT: $bt_{i,j} = \sum_{k=1}^{bsize} ba_{i,k} \cdot bb_{k,j}$.



Как и в предыдущем подходе, распределяее между потоками осуществлялось статически: занумеруем все блоки, при запуске каждый поток в качестве аргумента принимает интервал блоков $[start_i\ end_i)$ ($i = 0 \dots bcount \cdot bcount$-1, $end_i = start_{i+1}$ для $i = 0 \dots T$-2, где T - количество потоков) результирующей матрицы, которые ему надо посчитать. Как и в простой реализации, все потоки вычисляют различные элементы матрицы C и синхронизация не требуется.

1.2.3 Наибольший параллелизм

Из определения умножения следует, что для вычисления матрицы C необходимо произвести n^3 умножений. Распределим умножения между всеми потоками: занумеруем все умножения числами от 0 до n^3-1 и аналогично предыдущим реализациям будем в каждый поток передавать интервал $[start_i; end_i)$ – номера умножений, которые надо посчитать.

При данном подходе разные потоки могут считать одни элементы матрицы C, поэтому для каждого из них создаём отдельную матрицу C_i, т. о. синхронизация по-прежнему не требуется. Итоговая матрица $C = \sum_{i=0}^{T-1} C_i$.

1.3 Реузльтаты тестирования

Измерение времени осуществлялось с помощью системной команды time. Размер матрицы $n = 2000$.

Реализация	**Кол-во потоков**	**Время**		
		обычное	блочное (100)	необычное
Однопоточная		19.209	20.520	-
Многопоточная	1	19.148	20.520	19.182
Многопоточная	2	10.685	11.275	10.849
Многопоточная	3	7.887	8.241	8.203
Многопоточная	4	6.913	6.706	8.463
Многопоточная	5	7.858	7.644	7.941
Многопоточная	6	7.181	6.797	7.807

2 LU-разложение

2.1 Постановка задачи

Известно, что квадратную матрицу A размера n можно представить как произведение двух матриц $L \cdot U$ размера $n \times n$, где L – нижняя треугольная матрица, U – верхняя треугольная матрица. Элементы $l_{i,j}$ и $u_{i,j}$ соответственно матриц L и U могут быть вычислены по формулам:

$u_{0,j} = a_{0,j}$, $0 \leq j < n$

$u_{i,j} = a_{i,j} - \sum_{k=0}^{i-1} l_{i,k} \cdot u_{k,j}$, $i \leq j < n$

$l_{i,i} = 1$

$l_{i,0} = \frac{a_{i,0}}{u_{0,0}}$, $1 \leq j < n$

$l_{i,j} = \frac{1}{u_{j,j}}(a_{i,j} - \sum_{k=0}^{i-1} l_{i,k} \cdot u_{k,j})$, $i + 1 \leq i < n$

2.2 Подход к распараллеливанию

По побочным диагоналям.

2.3 Результаты тестирования

Измерение времени осуществлялось с помощью встроенной в bash утилиты time. Размер матрицы $n = 2000$.

Однопоточная		54.457
Многопоточная	1	63.709
Многопоточная	2	36.795
Многопоточная	3	28.918
Многопоточная	4	25.877
Многопоточная	5	36.348