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 $C = A \times B$, $c_{i,j} = \sum_{k=1}^{n} a_{i,k} \cdot b_{k,j}$.

2 Подход к распараллеливанию

2.1 Простая реазиация

Статически распределяем задачу между потоками: при запуске каждый поток в качестве аргумента принимает интерал 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.

2.2 Блочное умножение

Аналогично предыдущему, за исключением того, что каждый поток принимает номера блоков, которые ему надо посчитать.

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 $C = \sum_{i}^{T-1} C_i$.

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