Привязать program к cpu 1,2,3: taskset -c 1,2,3 program
Привязать program к ноде 0: numactl -N 0 program
Привязать program к cpu 4,5,6: numactl -C 4,5,6 program

taskset: https://man.archlinux.org/man/taskset.1
numactl: https://man.archlinux.org/man/numactl.8

Для всех заданий:
1) собирать при помощи make или cmake;
2) прикреплять ссылку на github с исходным кодом программы; 3) все вычисления проводить на сервере.

Анализ эффективности OpenMP-версии

- **Введем обозначения:**
  - $T(n)$ – время выполнения последовательной программы (serial program) при заданном размере n входных данных
  - $T_p(n,p)$ – время выполнения параллельной программы (parallel program) на p процессорах при заданном размере n входных данных
- **Коэффициент $S_p(n)$ ускорения** параллельной программ (Speedup):

$$S_p(n) = \frac{T(n)}{T_p(n)}$$

Во сколько раз параллельная программа выполняется на p процессорах быстрее последовательной программы при обработке одних и тех же данных размера n

- Как правило

$$S_p(n) \leq p$$

- **Цель распараллеливания –**
достичь линейного ускорения на наибольшем числе процессоров: $S_p(n) \geq c \cdot p$, при $p \to \infty$ и $c > 0$

Задание 1

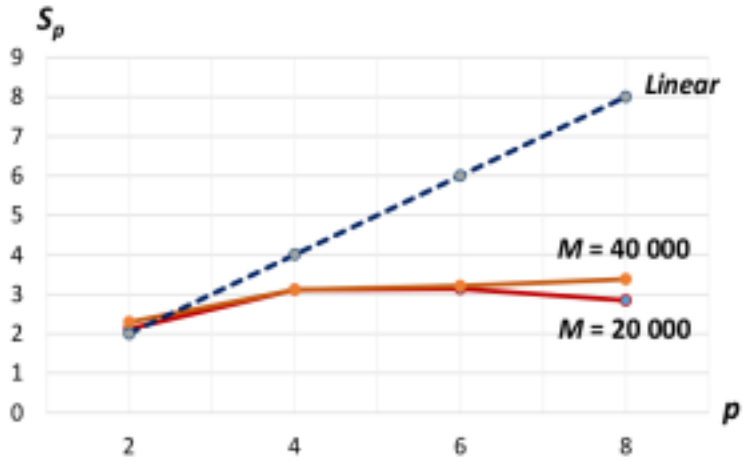
Описать вычислительный узел: наименование и краткая характеристика CPU (lscpu), наименование сервера (cat /sys/devices/virtual/dmi/id/product_name), сколько NUMA node (numactl --hardware), сколько памяти у каждой ноды, операционная система (cat /etc/os-release).

Используя семинар 1 Михаила Курносова, реализовать многопоточную версию программы умножения матрицы на вектор с параллельной инициализацией массивов (Код можно писать на C\C++). Провести анализ маштабируемости, (заполнить таблицу, по шаблону

ниже):

M = N	Количество потоков								
	2			4		6		8	
	T_1	T_2	S_2	T_4	S_4	T_6	S_6	T_8	S_8
20 000 (~ 3 GiB)									
40 000 (~ 12 GiB)									

Заполнить таблицу для 1,2,4,7,8,16,20,40 потоков. Для элементов массива используйте тип double. Размеры матриц: 20000x20000, 40000x40000. Построить график ускорения в зависимости от количества потоков:



Написать вывод о маштабируемости. Результат сохранить в формате pdf, и прикрепить в гугл класс. *Дополнительно* (по желанию): Привязать потоки к ноде или конкретным ядрам, и сравнить результат с предыдущим (без привязки).

Задание 2

Описать вычислительный узел: наименование и краткая характеристика CPU (lscpu), наименование сервера (cat /sys/devices/virtual/dmi/id/product_name), сколько NUMA node (numactl --hardware), сколько памяти у каждой ноды, операционная система (cat /etc/os-release).

Используя семинар 2 Михаила Курносова, реализовать параллельную версию программы численного интегрирования, написать код функции «integrate_omp» с использованием «#pragma omp atomic» и локальной переменной (Код можно писать на C\C++). Оценить ускорение

программы на 1,2,4,7,8,16,20,40 потоках при числе точек интегрирования nsteps = 40 000 000. Построить график ускорения в зависимости от количества потоков.

Написать вывод о маштабируемости. Результат сохранить в формате pdf, и прикрепить в гугл класс. *Дополнительно* (по желанию): Привязать потоки к ноде или конкретным ядрам, и сравнить результат с предыдущим (без привязки).

Задание 3

Выполнить «Лабораторную работу №2», использовать метод простой итерации. Написать код на C++.

Исходные данные

Элементы главной диагонали матрицы A равны 2.0, остальные равны 1.0.Все элементы вектора b равны N+1. В этом случае решением системы будет вектор, элементы которого равны 1.0. Начальные значения элементов вектора x можно взять равными 0.