МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет информационных технологий

Кафедра параллельных вычислений

ОТЧЕТ

О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

“Параллельная реализация решения системы линейных алгебраических уравнений с помощью OpenMP”

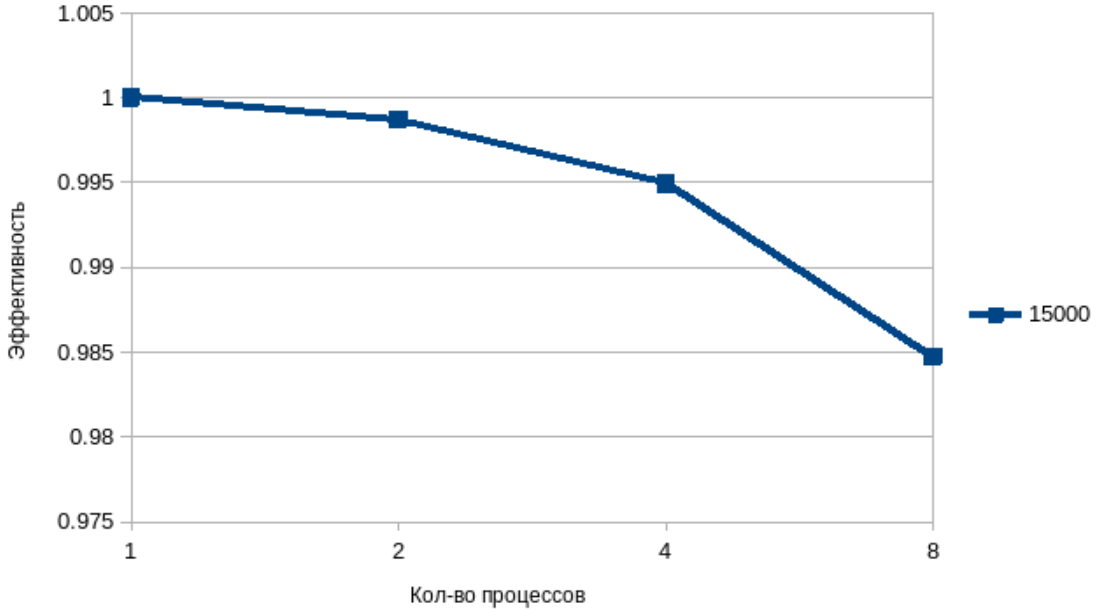
студента 2 курса, 20212 группы

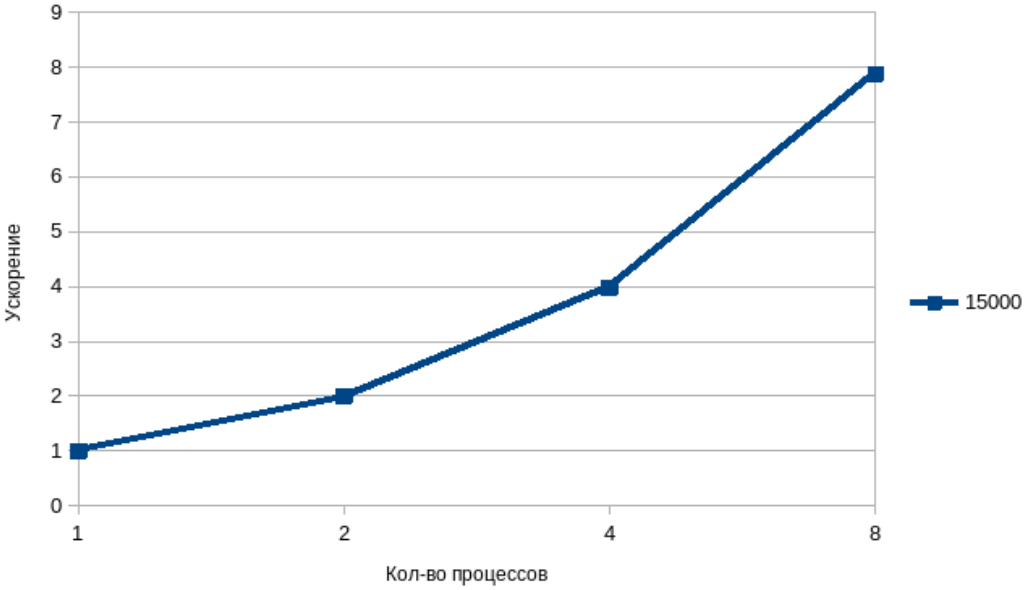
Курбатова Максима Андреевича

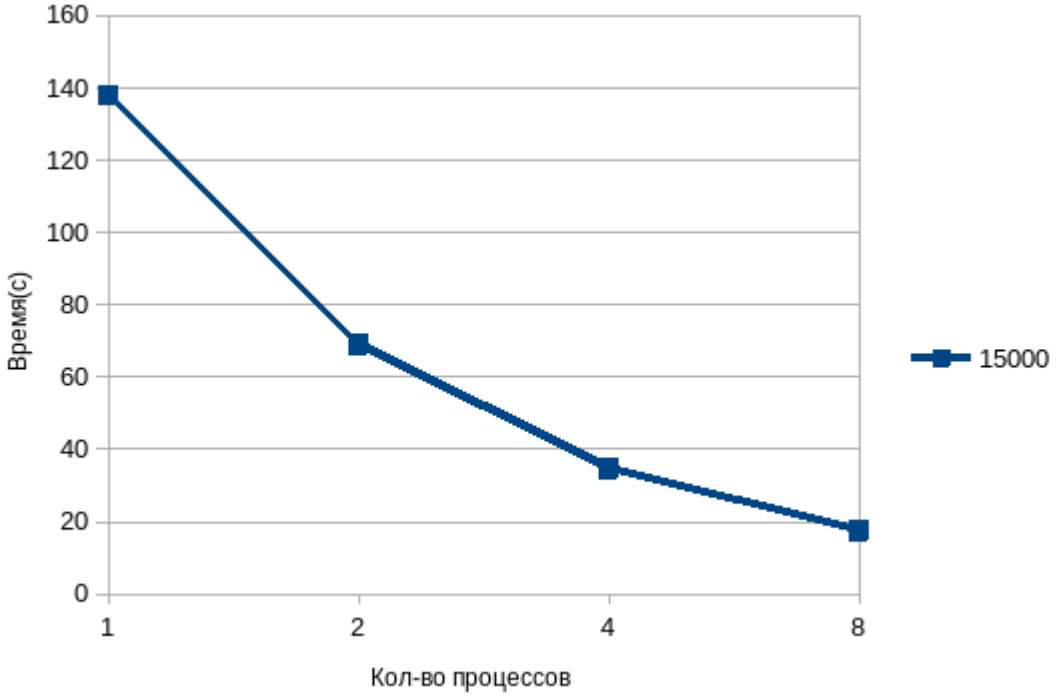
Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель: Кудинов А.Ю.

Новосибирск 2022

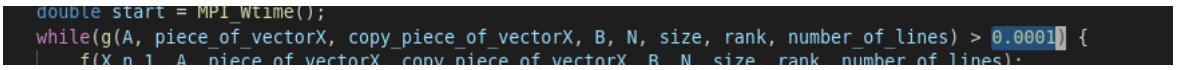

1.005
1
0.995
0.99
0.985
0.98
0.975
Эффективность
1
2
4
8
Кол-во процессов
15000


9
8
7
6
5
4
3
2
1
0
Ускорение
1
2
4
8
Кол-во процессов
15000

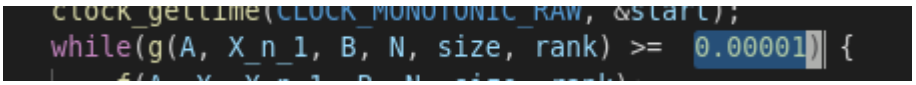

160
140
120
100
80
60
40
20
0
Время(с)
1
2
4
8
Кол-во процессов
15000

Вопрос с семинара: Почему OpenMP на одном потоке работает дольше, чем MPI на одном процессе?
Ответ:

MPI:



OpenMP:



Просто я задал параметр t другой и не заметил этого. Из-за этого решения сходились за разное количество итераций. После замены на t = 0.00001 обе программы стали считать на одно и тоже время на одном процессе/потоке(с точностью до десятой секунды).

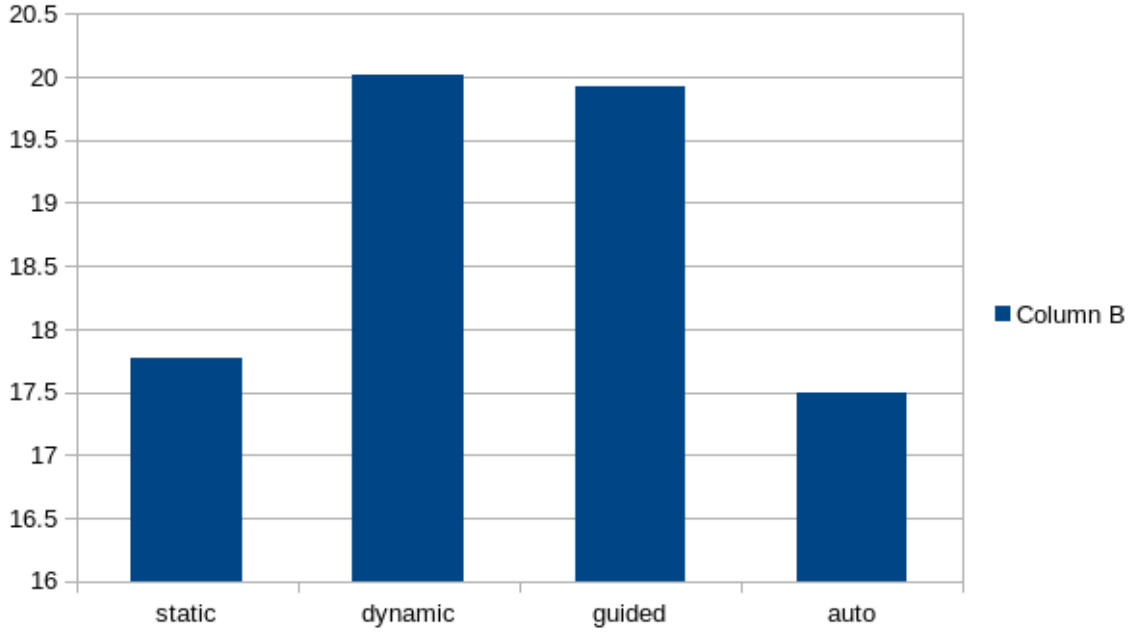
Сравнение с MPI на 8 процессах/потоках.

	Размер	1	2	4	8
OpenMP	15000	138.1	69.14	34.7	17.53
MPI	15000	138.21	69,8	35.1	18.2

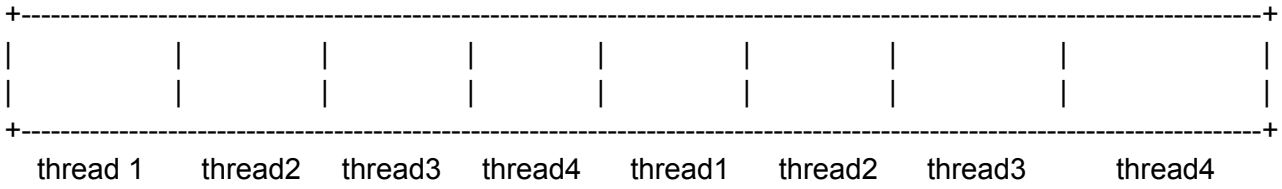
На семинаре, я сказал, что OpenMP проигрывает в скорости MPI. Я ошибался. Я получил неверные результаты, когда указал разные значения параметра t. Все-таки доступ к разделяемой памяти не такой уж и долгий, по сравнению с общением процессов.

Стратегии распараллеливания

Матрица 15000 x 15000 и 8 потоков

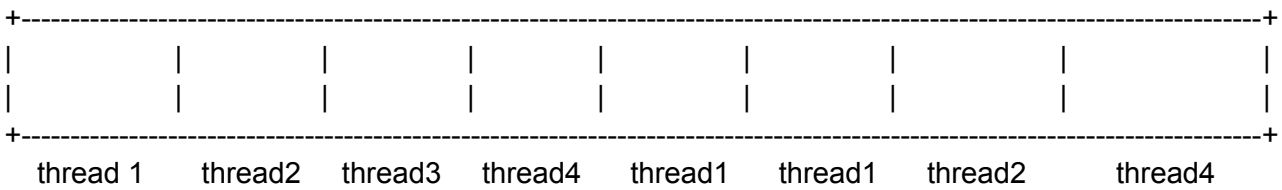


- **schedule(static, chunk_size)** - пространство итераций делится на кусочки, размером chunk_size и раздаются потокам циклически(см. иллюстрацию для 4 потоков)



Если chunk_size не указан, то он равен кол-во итераций / кол-во потоков

- **schedule(dynamic)** - пространство итераций делится на кусочки, размером 1 и раздаются потокам. Как только один из потоков посчитает свои итерации, то будет послан запрос, чтобы получить новые итерации. Очень полезно, если разные итерации цикла работают разное время.(Можно явно указать размер chunk_size.)



- **schedule(guided)** - тоже самое, что и dynamic, но пространство итераций делится на кусочки, размером кол-во итераций / кол-во потоков

- **schedule(auto)** - стратегия распараллеливания определяется компилятором.

Почему dynamic/guided самый долгий?

Это связано с тем, что во время компиляции не определено, какие итерации будет исполнять конкретный поток. Следовательно, в run-time посылаются запросы на получение итераций, что занимает время.