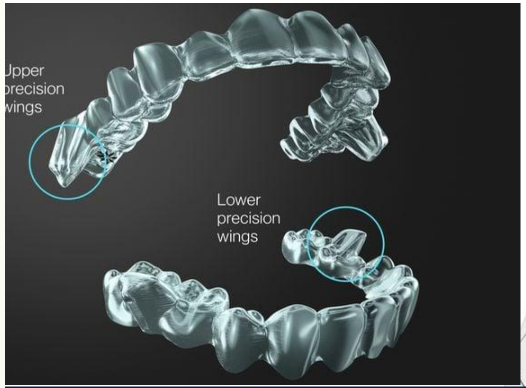
МГУ ВМК

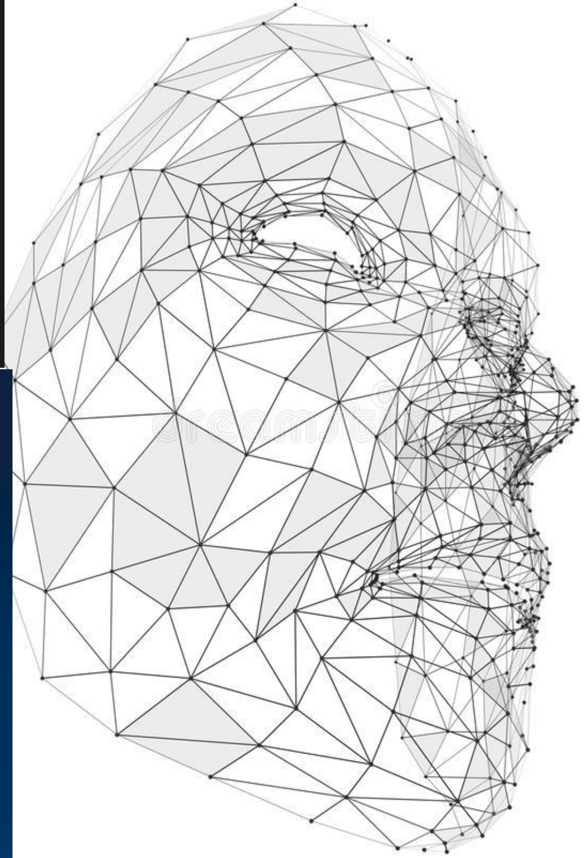
Кафедра суперкомпьютеров и квантовой информатики

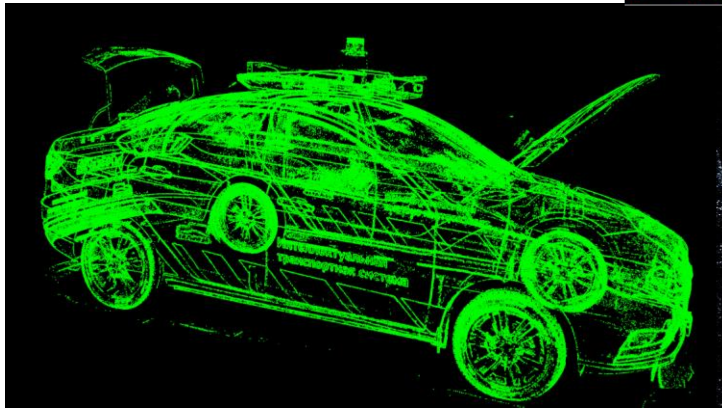
Диссертация на тему: «Исследование эффективности метода движущихся наименьших квадратов при реконструкции трёхмерной поверхности на суперкомпьютере»

Хабибулин Марат СКИ-638
Научный руководитель:
к.ф.-м.н., доцент Никольский Илья Михайлович

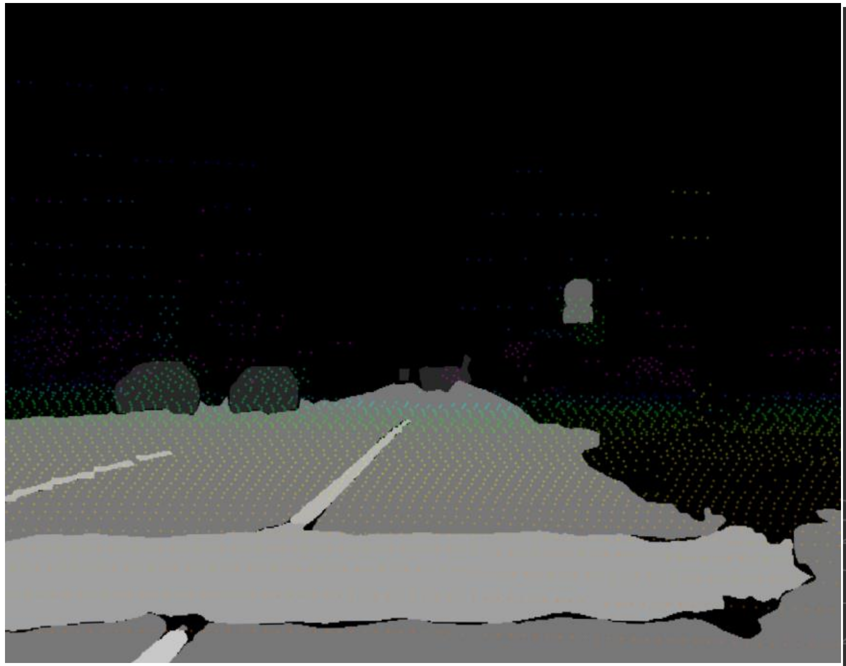
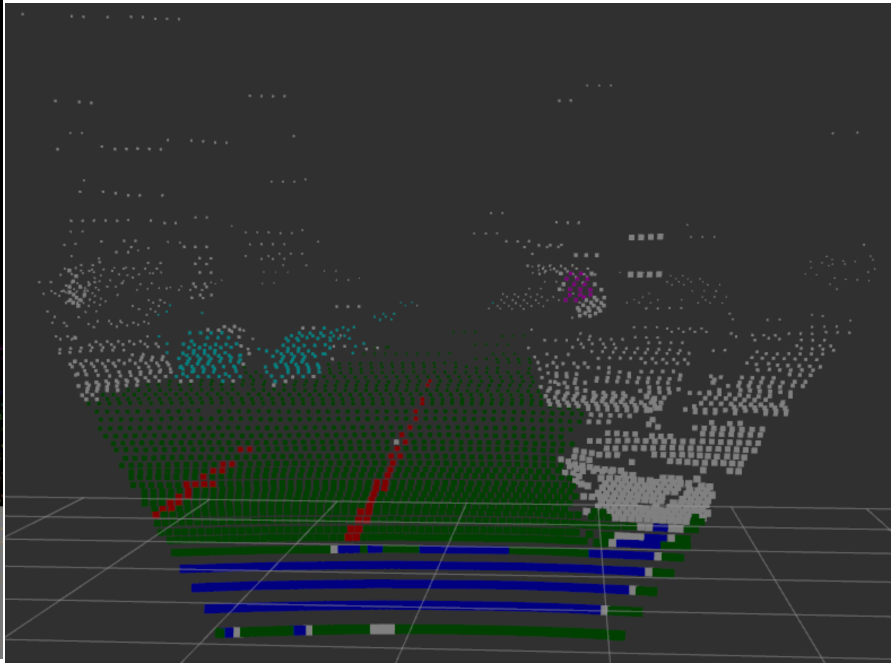
Актуальность







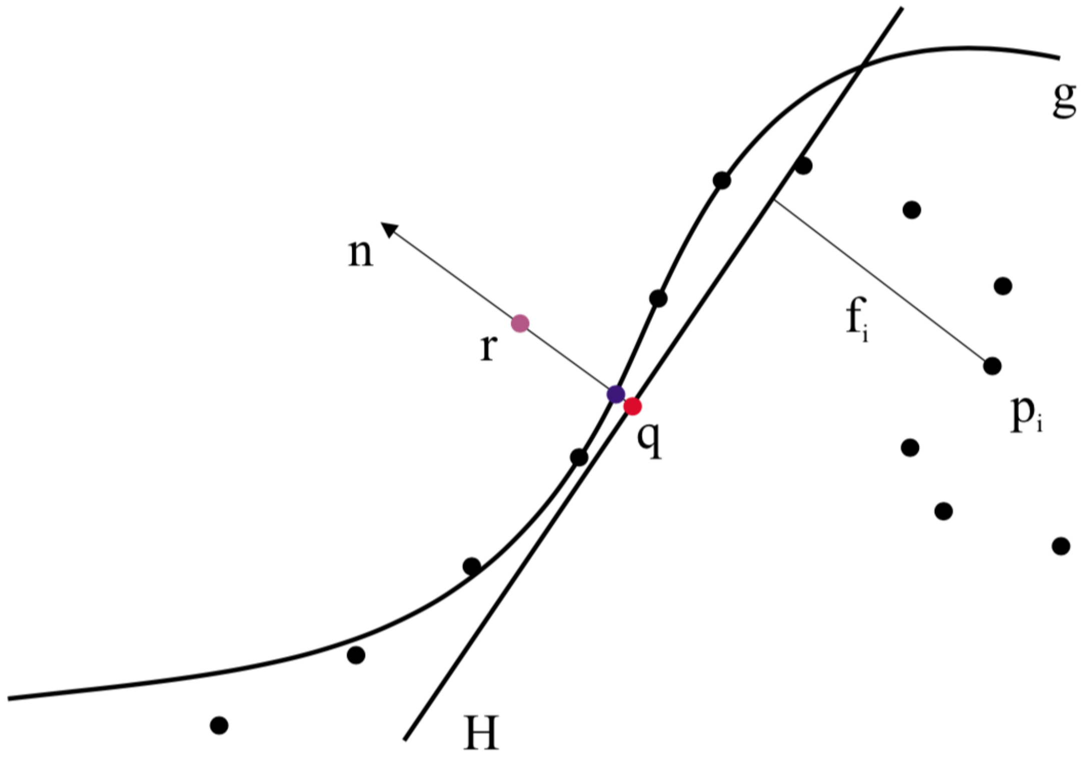
Актуальность

Цель работы и постановка задачи

- Целью настоящей работы является разработка параллельного метода реконструкции поверхности на высокопроизводительных кластерах, позволяющего добиться оптимальных результатов как в эффективности, так и в качестве восстановления поверхности.
- Постановка задачи
  1) Изучить существующие методы реконструкции поверхности
  2) Разработать алгоритм реконструкции поверхности, на распределенной памяти
  3) Протестировать разработанный алгоритм и оценить его эффективность, ускорение а также качество восстановленной поверхности

Алгоритм MLS(Moving Least Squares)

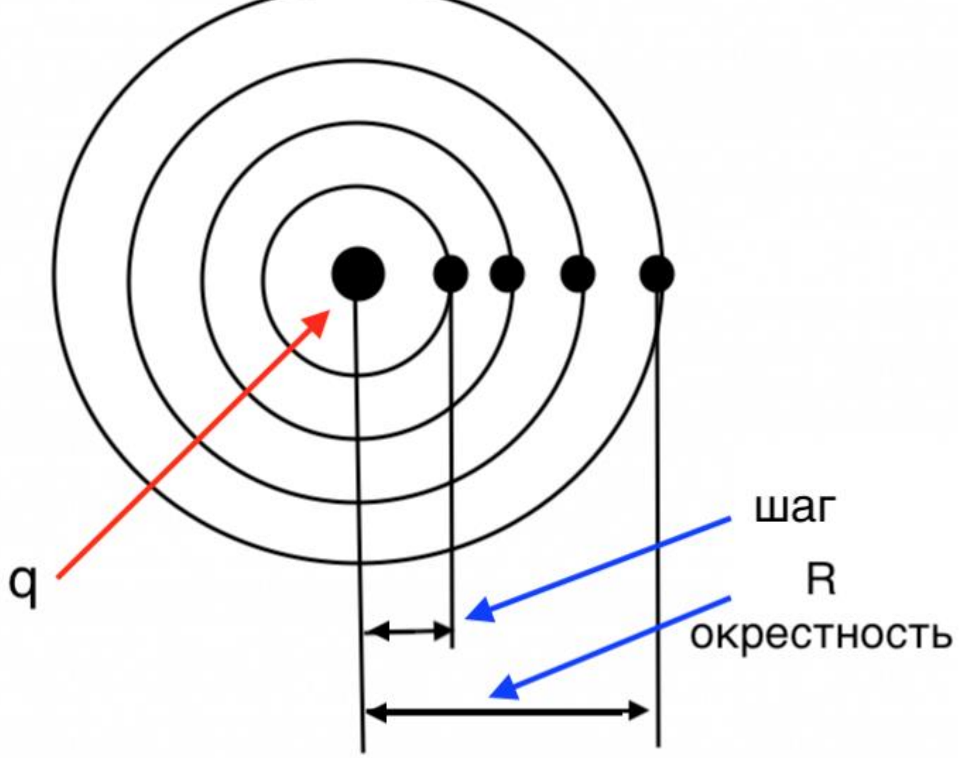


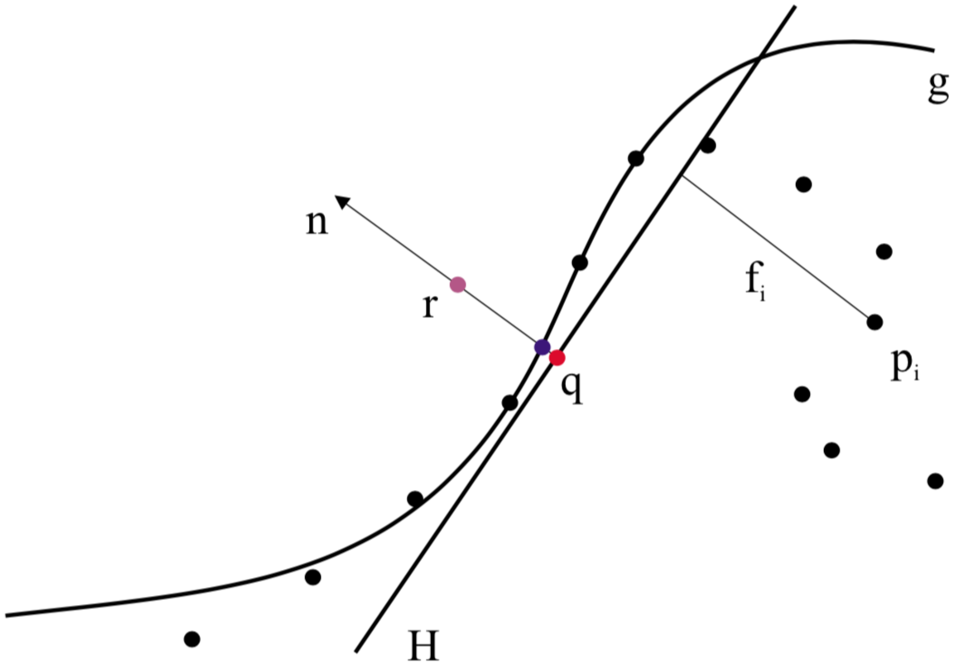
$$p_i \in R^3, i \in \{1, \dots, N\} \quad (1)$$

$$H = \{x \mid \langle n, x \rangle - D = 0, x \in R^3\}, n \in R^3, \| n \| = 1 \quad (2)$$

$$\sum_{i=1}^{N} (\langle n, p_i \rangle - D)^2 \theta(\| p_i - q \|) \quad (3)$$

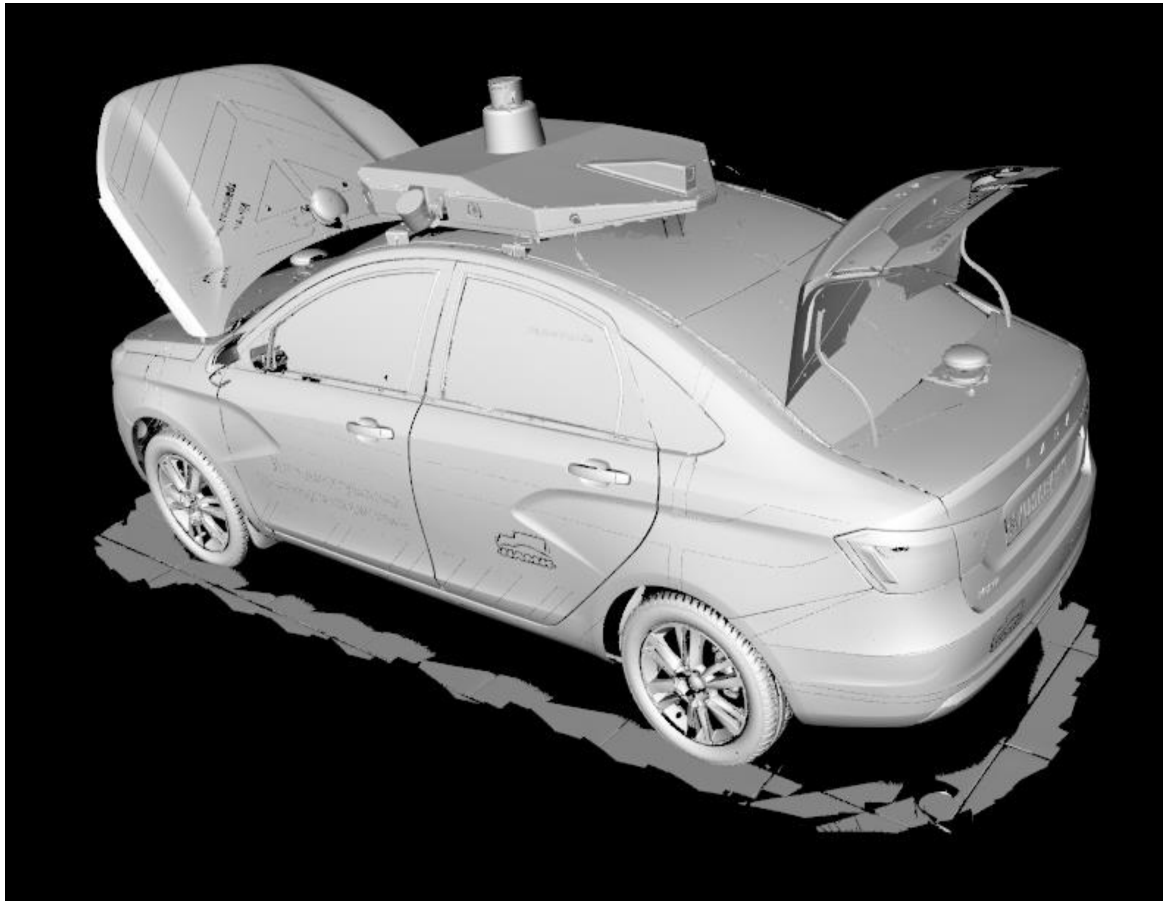
Повышение плотности точек



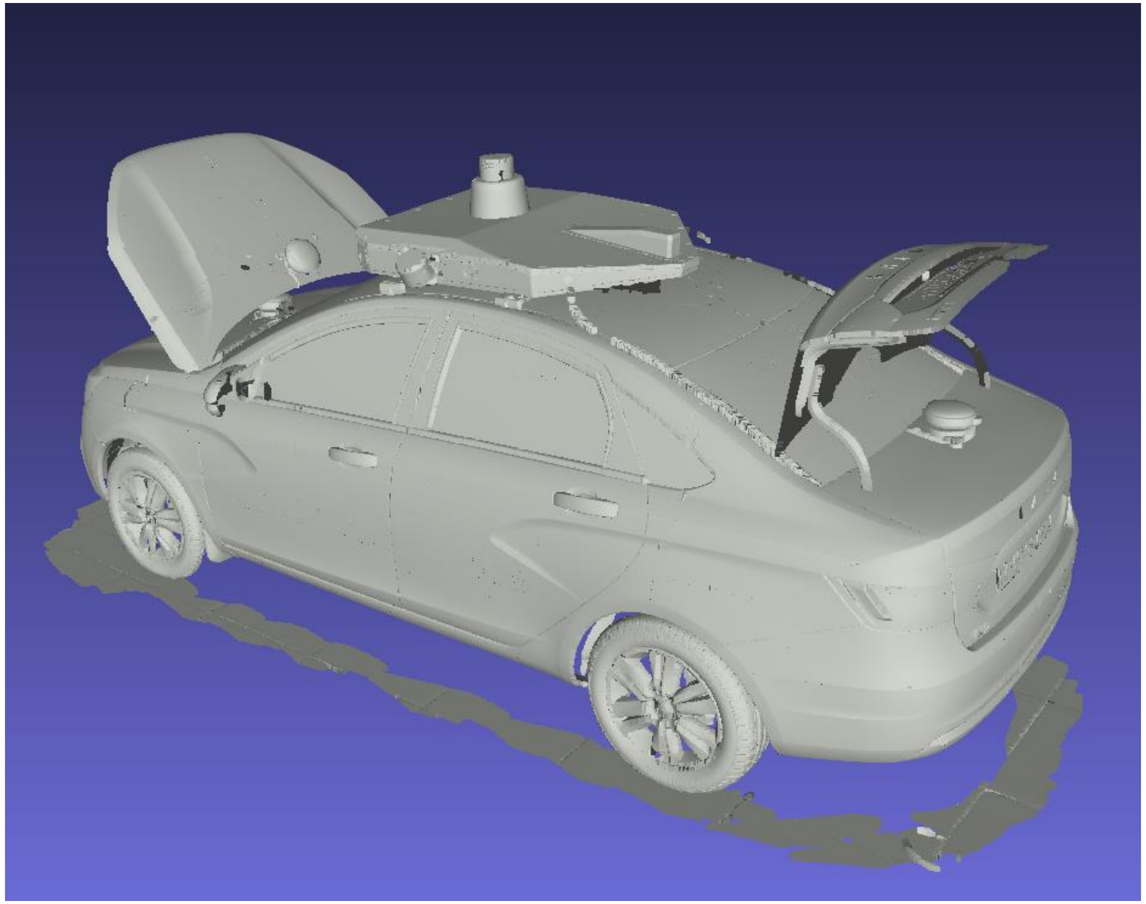


Рендеринг

25000000 полигонов



25000000 точек



http://graphics.stanford.edu/papers/qsplat/qsplat_paper.pdf

Предложенный параллельный метод

Алгоритм 1 Параллельный метод движущихся наименьших квадратов с MPI и OpenMP

Вход: набор точек $P = \{p_i\}\ i = 1..n$

Выход: поверхность представленная набором точек

for each process u **do**

  $P^{(u)} = read(P)$ // каждый процесс считывает свой сегмент облака точек $P^{(u)} = \{p_j\}\ j = 1..m$

  $P\_l^{(u)} = send\_recv(P\_r^{(u-1)})$ // получение левой границы

  $P\_r^{(u)} = send\_recv(P\_l^{(u+1)})$ // получение правой границы

  pragma omp parallel for

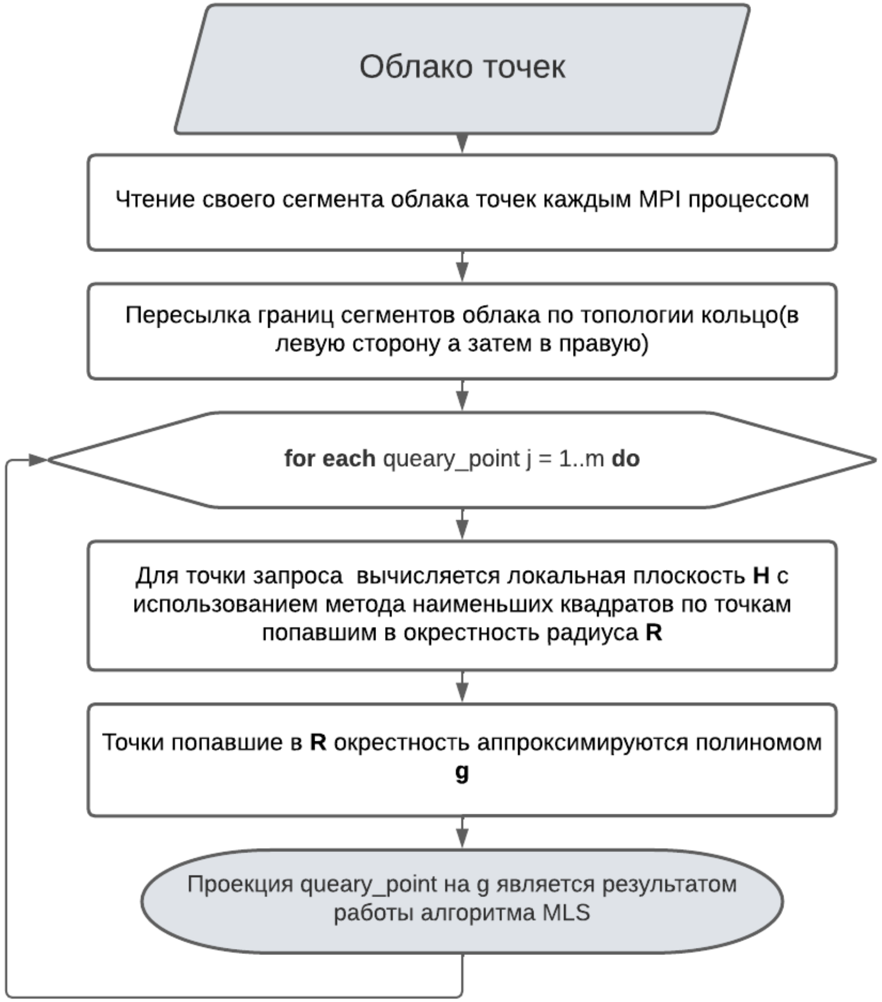
  for each point $j = 1..m$ **do**

    $H = generate\_plane(p_j)$

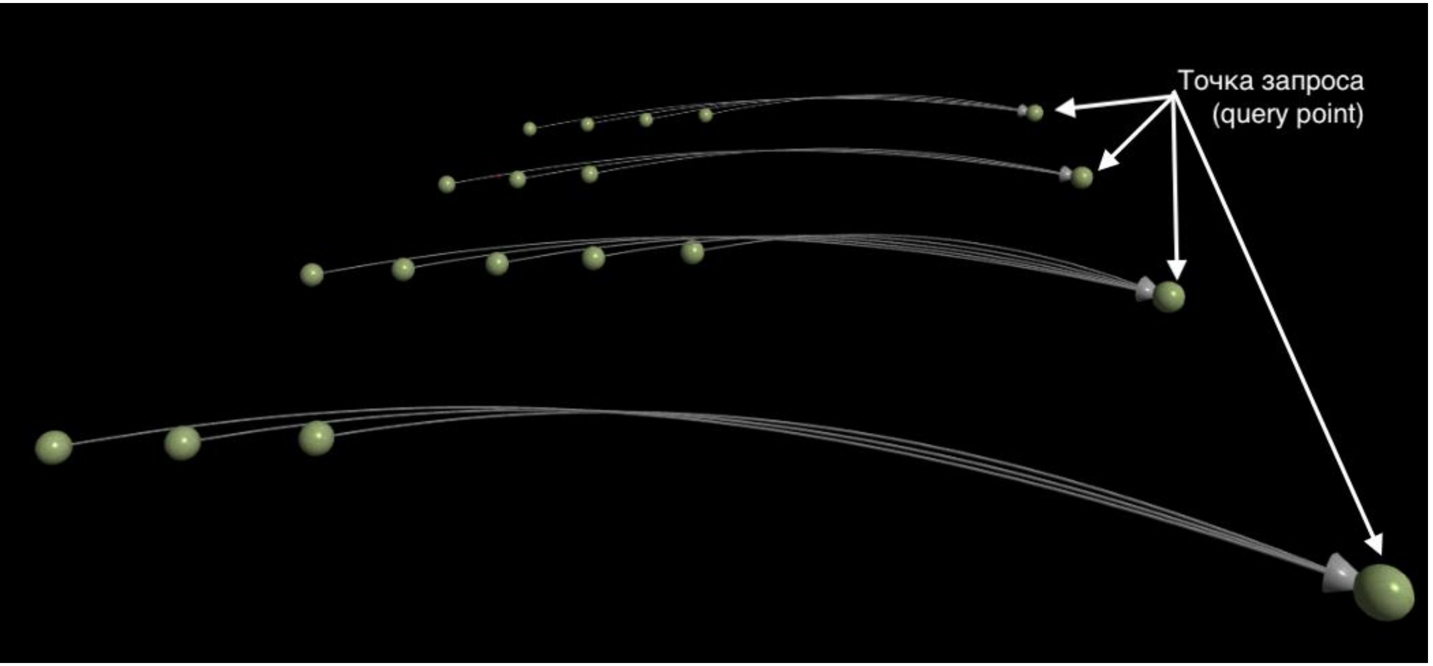
    $g = generate\_local\_polynomial\_approximation(H)$

    $result\_point = project\_on\_polynom(p_j, polynom)$

  end for



Граф информационной зависимости MLS



Алгоритм 1 Параллельный метод движущихся наименьших квадратов с MPI и OpenMP

Вход: набор точек $P = \{p_i\}$ $i = 1..n$

Выход: поверхность представленная набором точек

for each process u **do**

  $P^{(u)} = read(P)$ // каждый процесс считывает свой сегмент облака точек $P^{(u)} = \{p_j\}$ $j = 1..m$

  $P\_l^{(u)} = send\_recv(P\_r^{(u-1)})$ // получение левой границы

  $P\_r^{(u)} = send\_recv(P\_l^{(u+1)})$ // получение правой границы

  pragma omp parallel for

  for each point $j = 1..m$ **do**

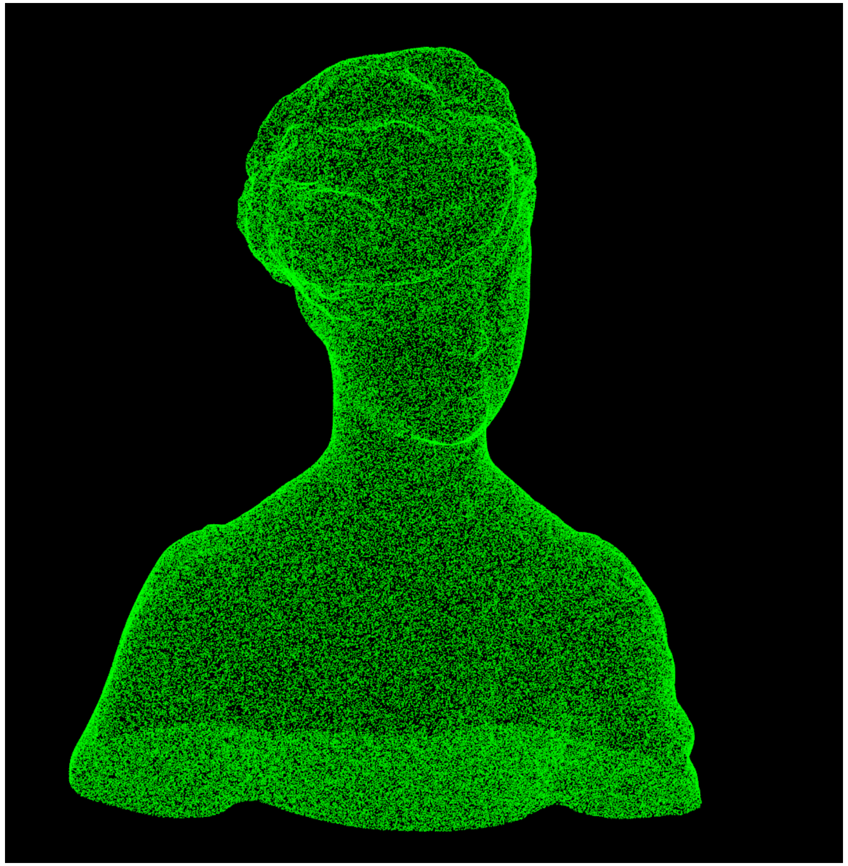
    $H = generate\_plane(p_j)$

    $g = generate\_local\_polynomial\_approximation(H)$

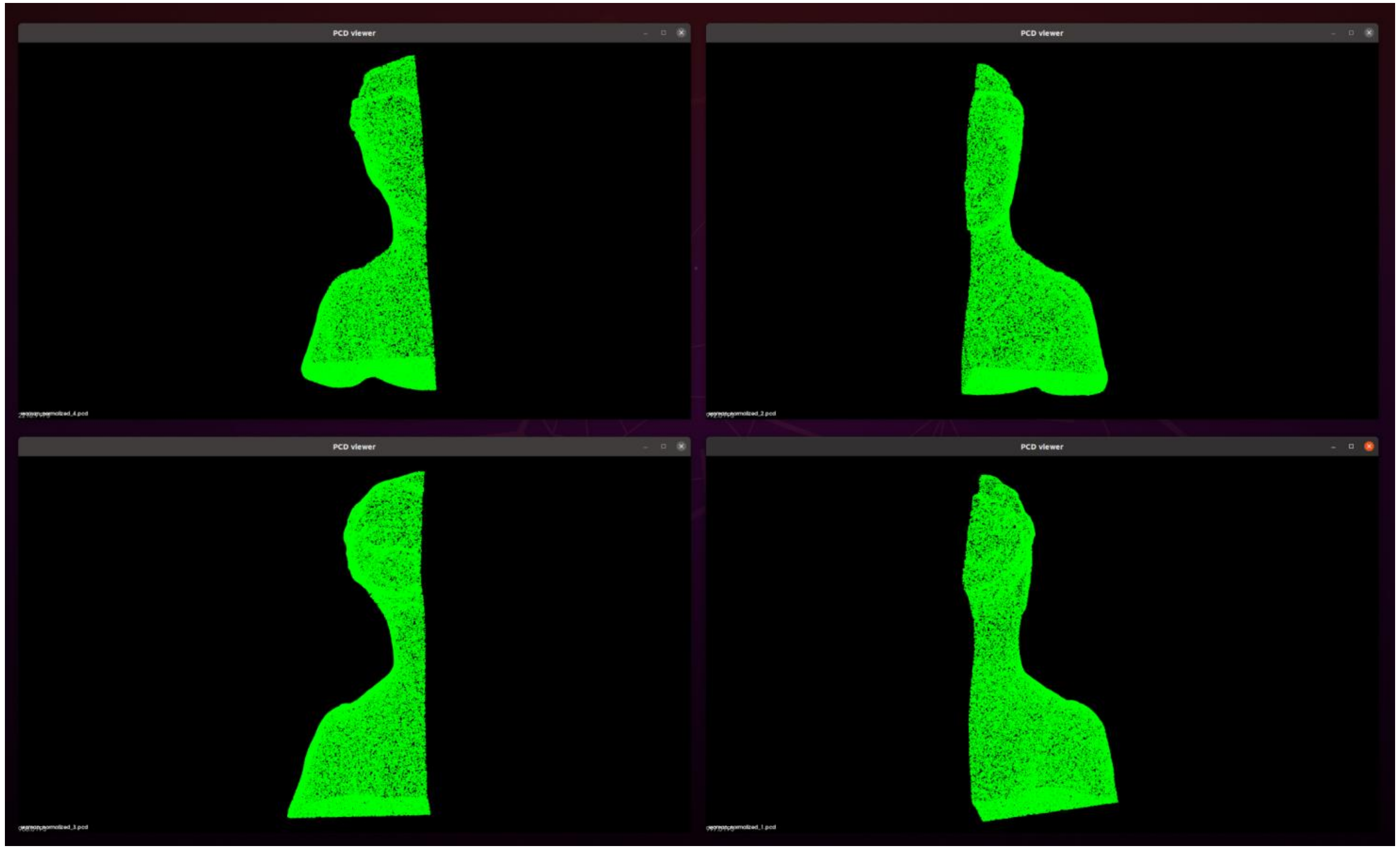
    $result\_point = project\_on\_polynom(p_j, polynom)$

  end for

Распределение сегментов облака точек по процессам



np = 1



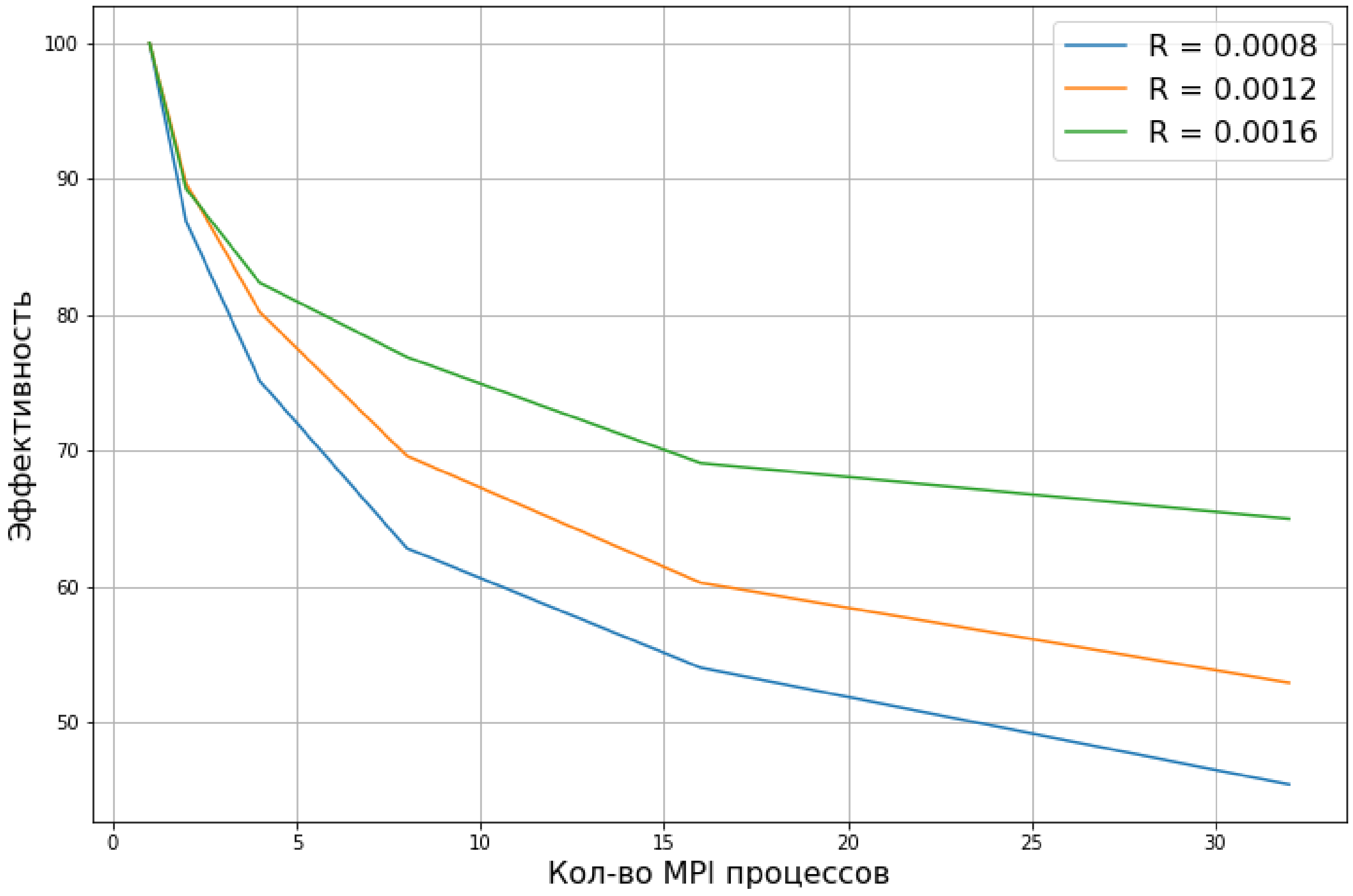
np = 4

Вычислительная сложность основных этапов алгоритма

Этап алгоритма	Вычислительная сложность
Построение k-d дерева	O(n*k*log(n)) (Несбалансированное O(n(k+log(n))))
Поиск точек в окрестности **R** по k-d дереву	O(log n)
Метод наименьших квадратов	$O(C^2*m)$ (C = 4)
Интерполяция полиномом	$O(m^2)$

https://jcgt.org/published/0004/01/03/
https://math.stackexchange.com/questions/84495/computational-complexity-of-least-square-regression-operation
https://cs.stackexchange.com/questions/93936/complexity-of-polynomial-interpolation

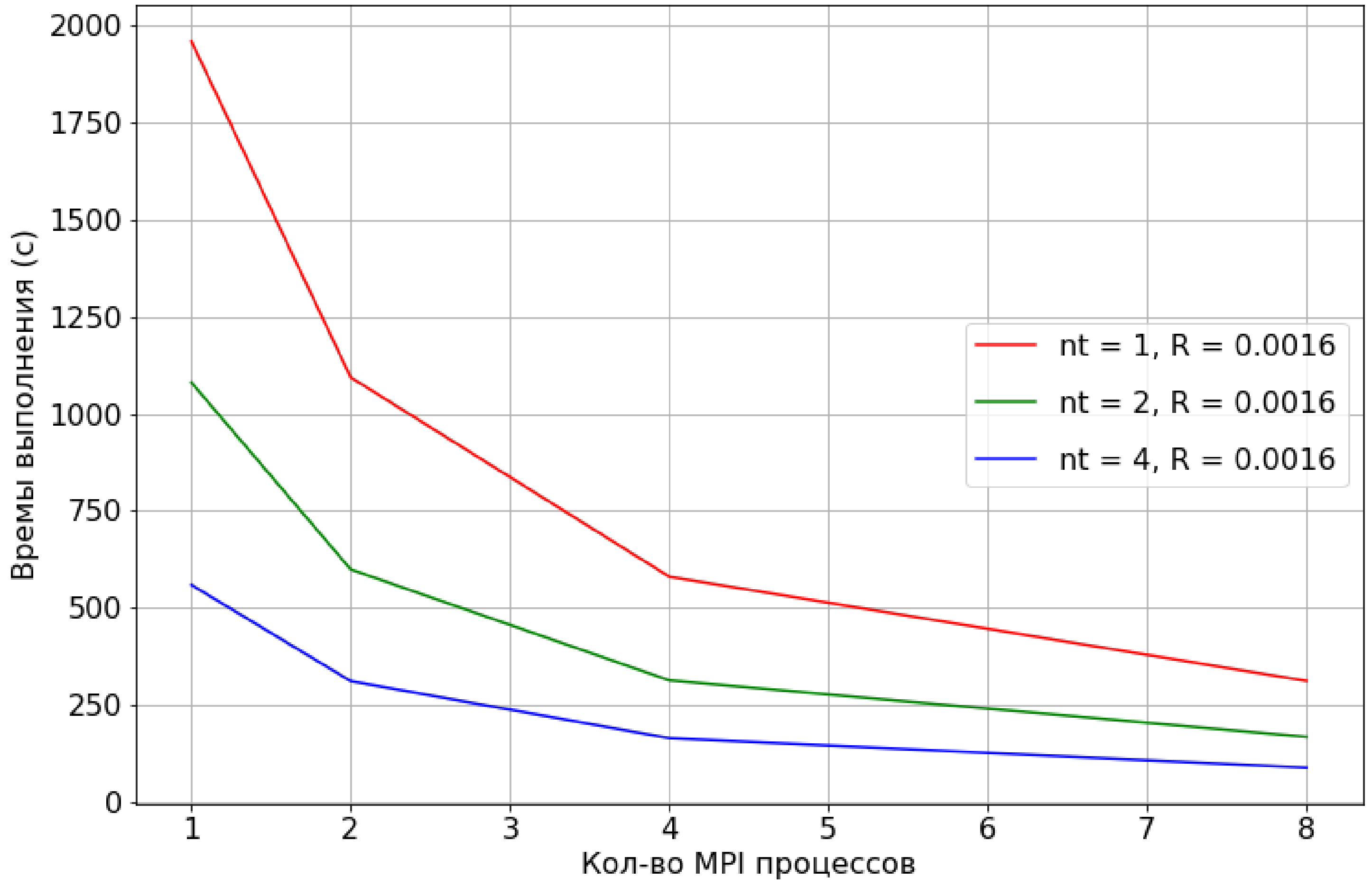
MPI программа



$$S_p = \frac{t_1}{t_p},$$

$$E_p = \frac{S_p}{p} * 100\%$$

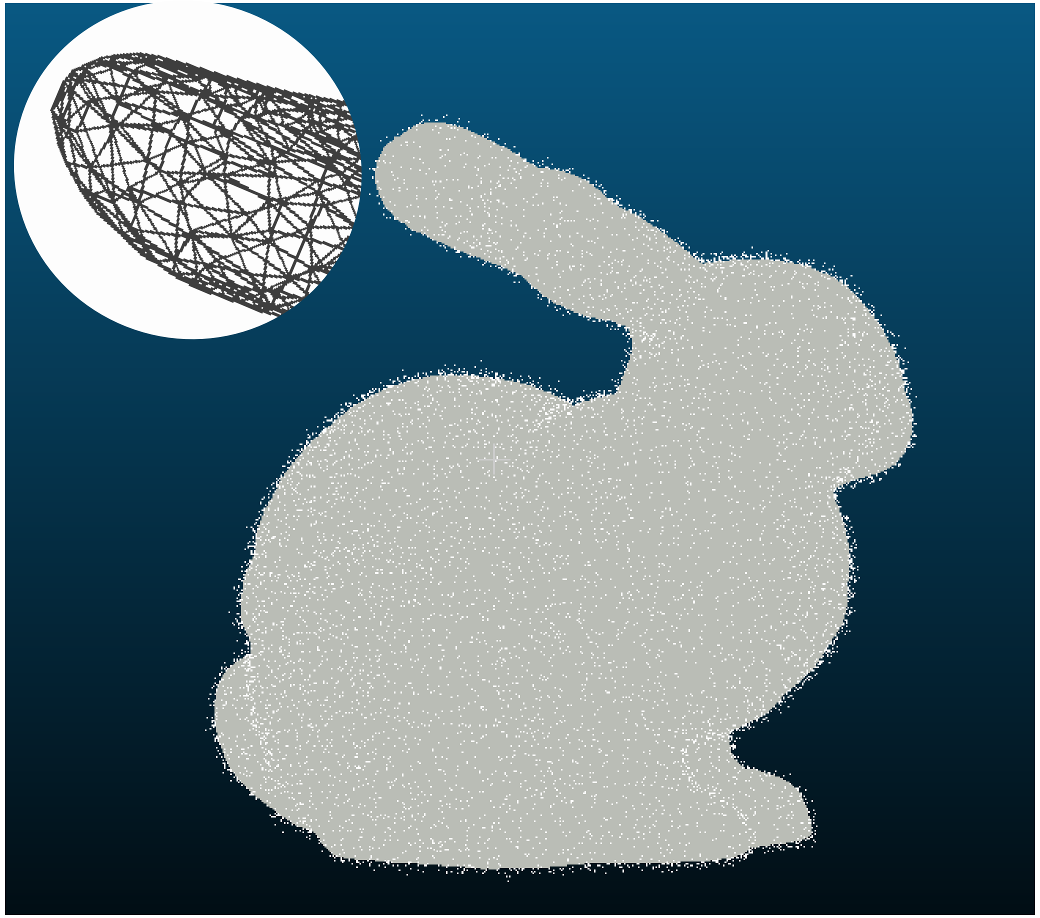
MPI + OpenMP



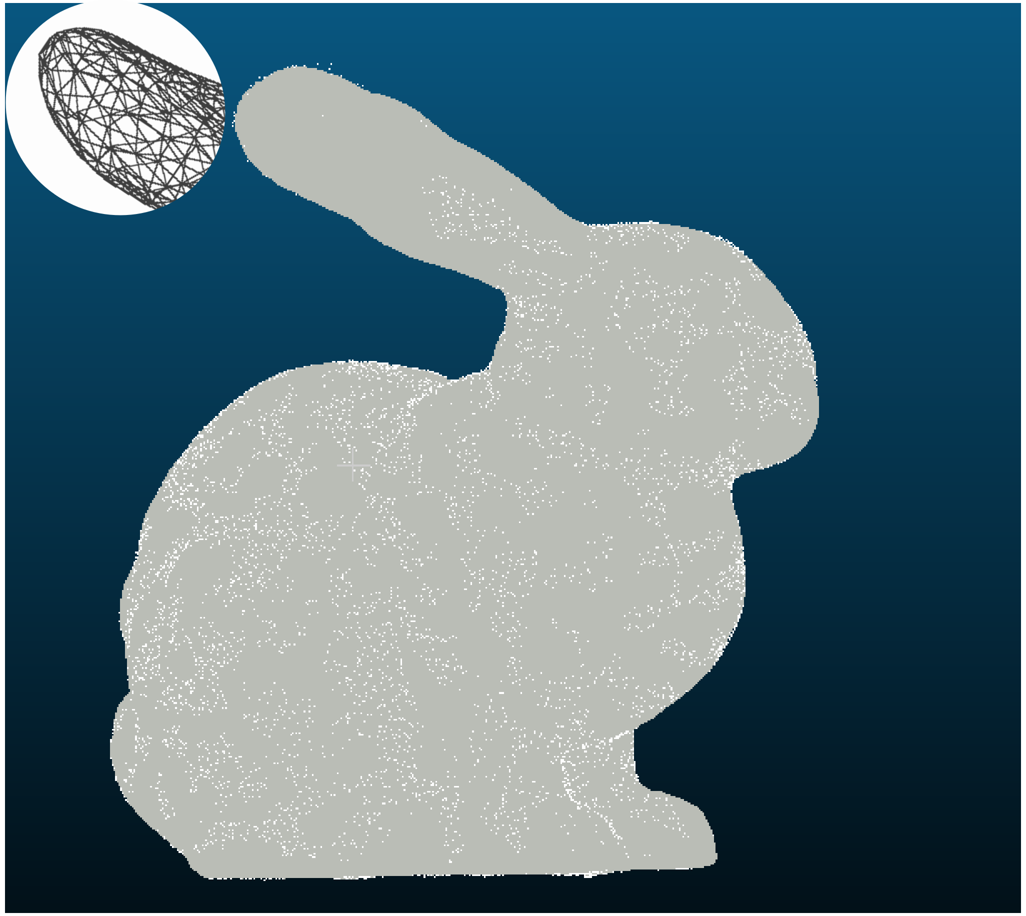
Сравнение ускорения для MPI и гибридной программы(R = 0.0016):

np*nt	MPI	MPI + OpenMP
4	3.38	3.51
8	6.3	6.7
16	11.61	12.9
32	21.65	23.7

Результат работы алгоритма



Зашумленное облако точек



После применения MLS

Результат работы алгоритма

Модель: Bunny

применен MLS	σ	**R**	ср. геом. откл.	ср. кв. откл.	min	max
×	0.005	–	0.00484	0.00269	3.8e-05	0.0245
✓	0.005	0.005	0.00372	0.00181	5.5e-05	0.01418
✓	0.005	0.01	0.00434	0.00245	6.7e-05	0.01715
✓	0.005	0.03	**0.00248**	**0.00117**	3.5e-05	0.01676
✓	0.005	0.05	0.00277	0.00185	1.3e-05	0.02362
×	0.01	–	0.00854	0.00563	0.0001	0.04387
✓	0.01	0.005	0.00578	0.00339	6.2e-05	0.02481
✓	0.01	0.01	0.00741	0.0045	8.7e-05	0.03055
✓	0.01	0.03	**0.00334**	**0.002**	9e-06	0.04387
✓	0.01	0.05	0.00344	0.00226	6.1e-05	0.03942
×	0.03	–	0.02333	0.01722	0.000167	0.13393
✓	0.03	0.005	0.01424	0.01001	0.000167	0.06133
✓	0.03	0.01	0.01672	0.01168	0.000158	0.07857
✓	0.03	0.03	0.02065	0.01572	0.00012	0.10811
✓	0.03	0.05	**0.01284**	**0.01116**	8.3e-05	0.11659

Основные результаты

- Реализованы следующие варианты параллельного алгоритма реконструкции поверхности:
  - для систем с общей памятью (с использованием OpenMP),
  - с распределённой памятью (с использованием MPI)
  - гибридный вариант (MPI + OpenMP)
- Исследована эффективность, разработанного алгоритма а также проведены тесты реконструкции реальных поверхностей.

Апробация

1. М.И. Хабибулин. Исследование эффективности метода движущихся наименьших квадратов при реконструкции трёхмерной поверхности на суперкомпьютере. Конференция "Ломоносов": труды международной научной конференции "Ломоносов". 10-21 апреля 2023 г., С. 40-43, Москва.

Спасибо за внимание

Список литературы

1. Levoy S. R. M. QSplat: A Multiresolution Point Rendering System for Large Meshes //
   Stanford University. — 2000. — C. 1.
2. Morton G. M. A computer oriented geodetic data base; and a new technique in file sequencing //
   IBM Ltd., Tech. Rep. — 1966.
3. Levin. D. The approximation power of moving least-squares // Mathematics of Computation. —
   1998. — C. 224.