Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра суперкомпьютеров и квантовой информатики

Арбузов Николай Романович

Исследование эффективности использования kd-деревьев при параллельной обработке облаков точек

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:
к.ф.-м.н., доцент
И. М. Никольский

Москва, 2023

Оглавление

1. Введение

Обработка облаков точек — это важная задача, которая имеет широкое применение в таких областях, как компьютерное зрение, графика, геодезия и многих других. Обработка этих данных требует высокой вычислительной мощности, поскольку облака точек могут содержать миллионы или даже миллиарды точек. Одним из наиболее эффективных способов индексации облаков точек является использование kd-деревьев, которые позволяют выполнять операции поиска, добавления и удаления точек с высокой скоростью и эффективностью. Однако, большинство существующих алгоритмов обработки облаков точек с использованием kd-деревьев являются последовательными и не могут полностью использовать мощность параллельных вычислений. Это ограничение означает, что время, необходимое для обработки больших объемов данных, может быть значительно увеличено, что делает такие задачи более сложными и затратными.

Целью данного исследования является создание и разработка параллельного алгоритма индексирования облаков точек с помощью kd-дерева, а также исследование его эффективности в условиях параллельных вычислений. Для достижения данной цели были поставлены следующие задачи:

- Изучение существующих работ по использованию kd-деревьев в обработке облаков точек и параллельных алгоритмов обработки облаков точек
- Разработка методики исследования, включающей описание используемых данных и методов обработки облаков точек, а также алгоритма параллельной обработки облаков точек с использованием kd-деревьев. Планируется создать алгоритм индексации данных, распределенных по узлам суперкомпьютера
- Проведение экспериментального исследования для сравнения эффективности параллельной обработки облаков точек с использованием kd-деревьев и последовательной
- Анализ результатов экспериментов и оценка эффективности использования kd-деревьев в параллельной обработке облаков точек.

Результаты данного исследования могут быть использованы для улучшения существующих алгоритмов обработки облаков точек с использованием kd-деревьев и разработки новых параллельных алгоритмов.

2. Обзор

2.1. Облака точек

Облако точек – набор трехмерных точек, представляющих поверхность некоторого объекта. Каждая точка облака имеет координаты (x, y, z), однако, в зависимости от задачи, помимо этих характеристик она так же может иметь параметры цвета, интенсивности или нормали в точке.

Существует множество способов получить облако точек, например получение облаков точек с помощью сенсоров. Сенсоры подразделяются на контактные и бесконтактные. Среди контактных можно выделить CMM (Coordinate Measuring Machine, Координатно-измерительная машина). Данный сенсор имеет от 3 до 5 осей для перемещения: 3 для перемещения сенсора по координатам X, Y и Z, а в случае 5 осей CMM так же имеет возможность поворачивать головку сенсора. На данный момент CMM является самым точным инструментом для сканирования предметов. [10] Среди бесконтактных сенсоров можно выделить, например LiDAR системы и сонары. LiDAR системы измеряют расстояние с использованием в работе разницу между временем излучением света и временем его получения после отражения от объекта. [11] Сонары схожи по принципу работы с LiDAR системами, но в отличии от них используют для определения расстояния звуковые волны, из-за чего они ограничены в средах применения и используются только под водой. [6] Помимо сенсоров, облака точек также можно получить с помощью специальных методов, например фотограмметрии. Фотограмметрия – это совокупность методов, позволяющая определить форму, размеры, положение и иные характеристики объекта путём соотнесения образов одних и тех же точек объекта на нескольких фотоснимках, сделанных с разных ракурсов. [13]

Облака точек используются во многих сферах деятельности человека, например можно выделить такие области, как:

- Компьютерное зрение. Облака точек применяются для распознавания образов при глубинном машинном обучении, например в сфере автопилотирования наземного транспорта. [3]
- Графика. В области графики облака точек используются для перевода реальных объектов (комнаты, механизмы, предметы, фасады зданий) в их виртуальную копию для дальнейшей обработки. [5]
- Геодезия. Облака точек часто используются для построения объемной карты местности, например, для отслеживания перемещения пород. [2]

2.2. Методы индексации облаков точек

Одним из ключевых аспектов обработки облаков точек является индексация, которая позволяет быстро находить нужные точки в больших объемах данных, а, соответственно, индекс – это структура данных, позволяющая ускорить поиск точек облака.

Существует множество различных способов индексации облаков точек, например: kd-деревья, октодеревья, r-деревья:

- Kd-деревья: это один из наиболее распространенных методов индексации облаков точек. kd-деревья являются иерархической структурой данных, которая разбивает пространство точек на более мелкие подпространства, так называемые ячейки. Каждый узел kd-дерева содержит точку и делит пространство на две части: левое и правое подпространства. Дерево строится путем рекурсивного разбиения ячеек до тех пор, пока не будет достигнуто определенное условие остановки. [4]
- Октодеревья: это метод индексации, который использует восьмиугольную структуру для разбиения трехмерного пространства на более мелкие ячейки. Каждый узел октодерева содержит информацию о ячейке, такую как ее размер и положение в пространстве. Дерево строится путем разбиения каждой ячейки на 8 более мелких ячеек, и процесс продолжается до тех пор, пока не будет достигнуто определенное условие остановки. [12]
- R-деревья: это метод индексации, который использует многомерное пространственное разбиение для организации точек в дерево. Каждый узел R-дерева может содержать несколько точек или других деревьев. Разбиение происходит путем поиска гиперплоскостей, которые разделяют точки на подмножества. Дерево строится путем рекурсивного разбиения до тех пор, пока не будет достигнуто определенное условие остановки. [9]

2.2.1. Модификации kd-дерева

С того момента, как в 1975 году Джон Бентли создал и описал принципы работы kd-дерева [4], многие ученые различными способами улучшали методы применения данной структуры, например, используя приближенные вычисления с помощью Akd-дерева (Approximate kd-дерева), в котором для решения задачи ICP (Iterative Closest Point, то есть алгоритм поиска ближайшего соседа) ищется не действительно ближайший сосед, а ближайшая точка из того блока, в котором располагается заданная точка [8]. Однако, у такого

подхода есть явно следующий из названия минус: в большинстве случаев данный алгоритм будет выдавать результат только с некоторой ошибкой, в то время как классическое kd-дерево дает точный результат. Другим примером улучшения может служить поиск в kd-дереве с кэшированием (Cashed kd-tree search) суть улучшения состоит в том, что узлы помимо указателей на свои дочерние узлы так же хранят и указатель на родительский узел, корневой узел хранит нулевой указатель, таким образом время поиска уменьшается вплоть до 50%, из-за перехода на уровень выше во время возможных тестов ball-within-bounds, тестов на поиск возможных ближайших точек в соседей узлах. [15]

Kd-деревья предназначены для работы с пространственными данными. Среди областей применения можно выделить такие, как трассировка лучей в реальном времени с использованием обычных процессоров [16] и с использованием графических ускорителей [18]. Помимо трассировки лучей, kd-деревья могут также использоваться и при обработке облаков точек, например, полученных с помощью датчиков LiDAR. [17]

2.2.2. Сравнение различных методов индексации

Как написано выше, помимо kd-деревьев для индексации пространственных данных часто используются такие структуры, как октодерево, r-дерево и так далее. В отличии от kd-деревьев октодеревья часто бывает несбалансированным ввиду своей идеи выделать сразу по 8 дочерних узлов при необходимости. Вследствие этого, как описано в работе Адамсона и Воркапика [1], октодеревья занимают в несколько раз больше места в оперативной памяти и затрачивают в несколько раз больше времени на выполнения поиска. Если брать r-деревья, то они представляют из себя древовидную структуру, узлы которой являются вложенными прямоугольниками (параллелепипедами), окруженными корневым узлом [9]. Однако, как описано в работе Нарасимхулу [14], r-дерево так же уступает kd-дереву по временным показателям, это происходит из-за того, что как алгоритм построения kd-дерева, так и алгоритмы поиска в нем проще и требует меньших вычислительных мощностей по нескольким параметрам: kd-дерево – это, в отличие от r-дерева, бинарное дерево, а значит каждый узел имеет максимум 2 дочерних узла, в то время, как в r-дереве их может быть больше.

2.3. Индексация с помощью kd-дерева при параллельной обработке облаков точек

Выше были описаны способы работы с облаками точек, а в частности подробно описано kd-дерево и приведены его улучшения и работы с его использованием, теперь обратимся к параллелизму при обработке облаков точек.

Один из методов параллелизма при обработке облаков точек предложили Госвами и пр. [7] Их метод так же использует kd-дерево, а точнее его модернизацию в виде мультинаправленного kd-дерева, в котором на каждом шагу облако разбивается на заданное количество областей с равным количеством точек по одной из осей. Параллелизм в данной работе используется только при рендеринге облака точек для его представления.

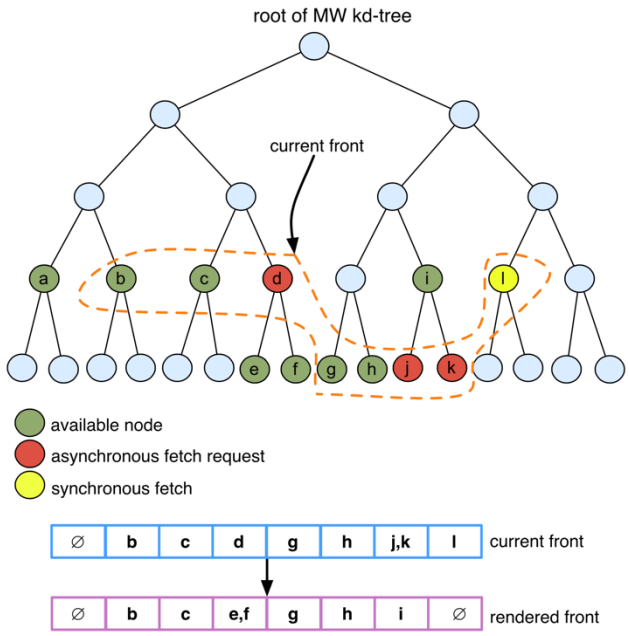

Fig. 6 LOD update and asynchronous fetching example for a MWKT with $N = 2$

Рисунок 1. Пример параллелизма в работе Госвами и пр.

Таким образом несмотря на сильный выигрыш по производительности, облако точек все еще необходимо хранить на одном узле и при поиске точек всем процессам придётся обращаться к данному узлу путём посылки сообщений, что будет замедлять работу программы.

3. Постановка задачи

Исследовать эффективность распределённого по узлам вычислительного кластера kd-дерева для обработки облака точек, точки которого также хранятся распределённым образом.

В качестве модельной задачи предполагается использовать задачу поиска точек облака, расположенных в заданном параллелепипеде.

Вычислительные эксперименты проводятся на следующих облаках точек: дракон, бюст (см. Рис. 2), а также на наборы из 100 тыс, 1 млн и 10 млн точек, случайным образом

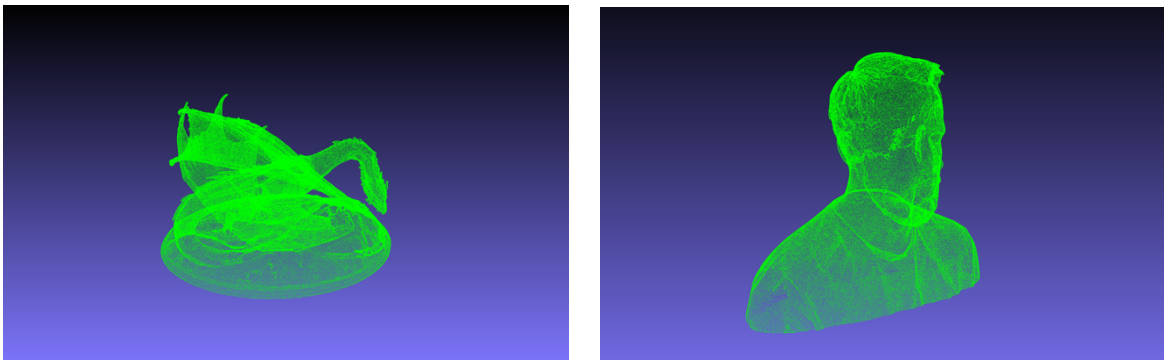

Рисунок 2. Слева: облако точек Fantasy Dragon (Дракон) (2 499 971 точек, https://www.artec3d.com/3d-models/fantasy-dragon). Справа: облако точек Bearded guy (Бюст) (1 024 356 точек, https://www.artec3d.com/3d-models/bearded-guy-hd)

распределённых внутри куба. При выполнении экспериментов с последовательной и параллельной версией программы отслеживаются время построения kd-дерева и время поиска точек в заданном диапазоне.

Те облака точек, которые не генерируются заново при каждом запуске программы (дракон и бюст), хранятся в формате XYZ. Данный формат файлов удобен своей простотой в использовании: координаты каждой точки хранятся на отдельной строке через пробел. Для работы с данным файлом можно использовать как стандартные функции для работы с текстовыми файлами, так и дополнительные библиотеки C++, например, библиотеку XYZ_IO.

Параллельная версия алгоритма поиска будет использовать библиотеку MPI (Message Passing Interface), которая предоставляет набор функций для передачи сообщений между процессами, а также для синхронизации выполнения процессов.

Для проведения опытов будут использованы локальный компьютер в части тестовых запусков и визуализации облаков, а также высокопроизводительная вычислительная система Polus для основных измерений. Локальный компьютер оснащен 4-ядерным процессором Intel Core i5 с частотой 2 ГГц и ОЗУ объемом 16 Гб с частотой 3733 МГц. Система Polus

состоит из 5 вычислительных узлов, на каждом узле установлены по 2 10-ядерных процессора IBM POWER8 и каждое ядро имеет 8 потоков, также Polus имеет общую оперативную память 256 Гбайт (в узле 5 оперативная память 1024 Гбайт) с ECC контролем.

4. Предлагаемая реализация распределенного kd-дерева

В качестве модели для изучения была создана модель распределенного kd-дерева, в котором процессоры выступают в роли узлов (см. Рис. 3). Предлагаемая реализация распределенного kd-дерева предполагает хранение на каждом процессе параллельной программы ровно одного узла дерева. Точки хранимого облака загружаются на корневой узел, затем через промежуточные узлы передаются в листья. Таким образом удаётся избежать дублирования данных, поскольку они хранятся только на листовых узлах.

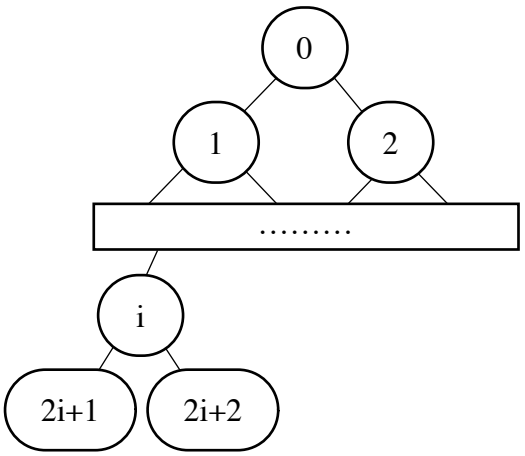

Рисунок 3. Схема распределения узлов дерева по процессам

Построение распределенного kd-дерева (см. Рис. 3) можно разделить на 3 этапа:

1. Построение дерева процессов. Определение отношения узлов родитель-потомок.
2. Пересылка частей облака точек к листовым узлам.
3. Построение локальных kd-деревьев на листовых узлах.

На первом этапе каждый процесс определяет в топологии дерева своего родителя и своих потомков, если таковые имеются, таким образом строятся связи в распределенном дереве. Ранги родителя и потомка можно получить из ранга процесса по следующим формулам:

$$r_{left\_c} = \begin{bmatrix} 2 * r + 1, \text{если } (2 * r + 1) \leq r_{max}; \\ -1, \text{ если } (2 * r + 1) > r_{max}; \end{bmatrix}$$

$$r_{right\_c} = \begin{bmatrix} 2 * r + 2, \text{если } (2 * r + 2) \leq r_{max}; \\ -1, \text{ если } (2 * r + 2) > r_{max}; \end{bmatrix}$$

$$r_{parent} = \begin{bmatrix} \left[\frac{(r-1)}{2}\right], \text{если } r \neq 0; \\ -1, \text{ если } r = 0; \end{bmatrix}, \text{где } [a] - \text{целая часть числа } a$$

После того, как все связи были построены, процесс с нулевым рангом считывает облако точек, определяет в нем медианную точку по оси и индексом 0, допустим это будет ось X. Далее этот процесс передает, если это возможно, своему левому потомку точки, у которых координата X меньше координаты найденной средней точки, а правому потомку остальные точки. В том случае, если у процесса только один потомок, он передает все полученные точки ему. На следующем уровне все происходит аналогично за исключением координаты, по которой производятся сравнения: для процессов с рангами 1 и 2 сравнения будут производиться по оси с индексом 1, допустим это ось Y, а для следующего уровня (процессы с рангами 3, 4, 5 и 6) по оси с индексом 2, допустим это ось Z. Таким же образом облако точек распределяется и для более глубоких уровней. Ось, по которой нужно разделять облако точек, можно определить с помощью ранга по формуле:

$$axis = lvl \bmod 3$$

$$lvl = [\log_2(r+1)], \text{где } [a] - \text{целая часть числа } a$$

Таким образом на листовых узлах появляются части облака точек.

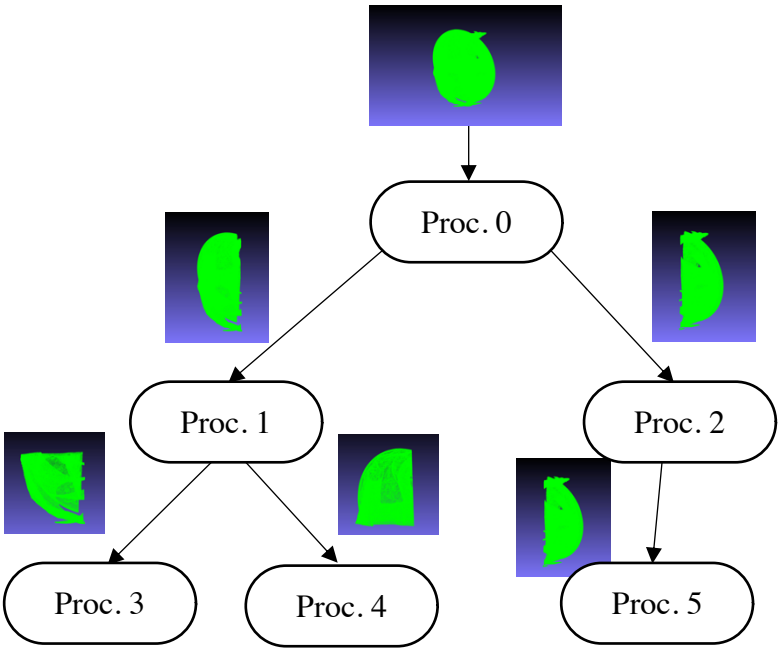

Рисунок 4. Организация распределенного дерева и передачи частей облаков точек в нем на примере системы с 5 процессами.

Третьим этапом построения распределенного дерева является построение локальных деревьев на листовых узлах. Данный процесс является рекурсивным и схож с распределением частей облака точек по процессам. На каждом шаге определяется медианная точка по оси, определенной из глубины рекурсии: на первом шаге это ось с индексом 0, на втором – с

индексом 1, на третьем – с индексом 2, а на четвертом – снова 0. После определения медианной точки алгоритм запускается для точек левее медианы и для точек правее. Рекурсия останавливается, когда она была вызвана для облака, состоящего из одной точки. Таким образом в конце этого алгоритма мы получаем дерево, с таким же количеством листьев, каким было количество точек в части облака.

5. Алгоритм поиска

В качестве алгоритма, на котором будет проверяться эффективность разработанной структуры, был выбран алгоритм поиска точек внутри заданной области пространства (параллелепипед) (см. Рис. 5, 6).

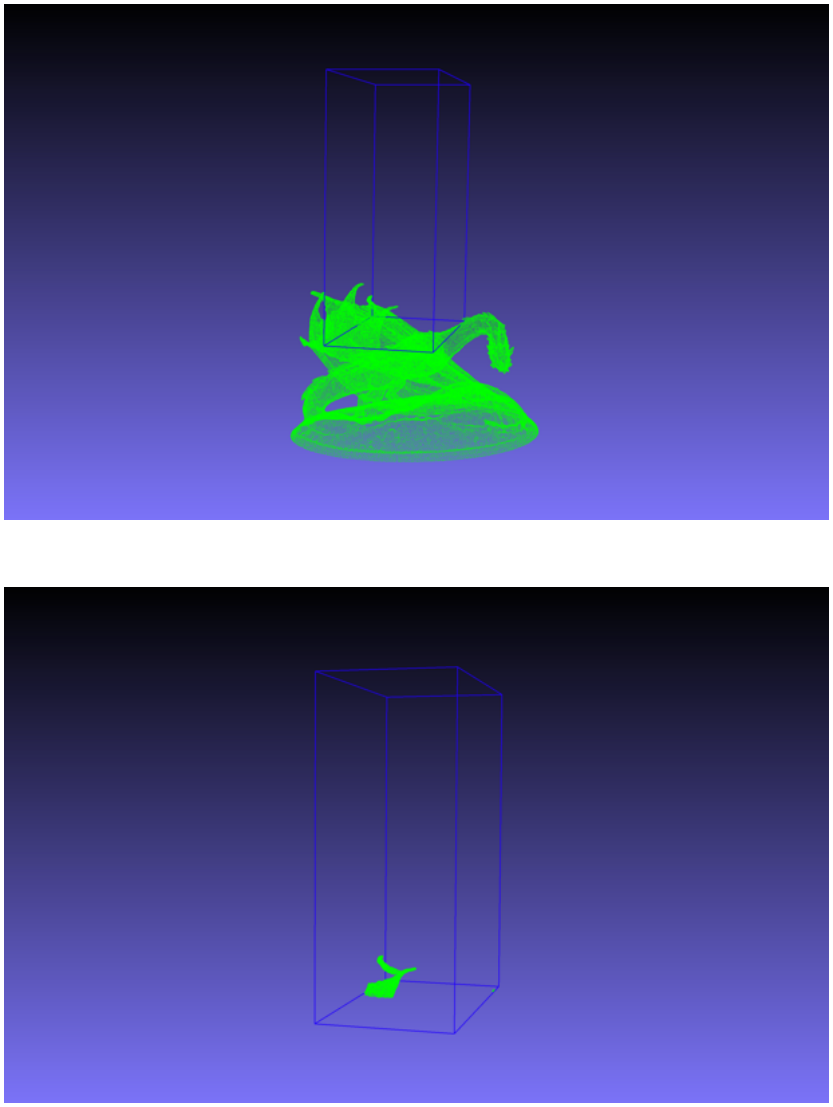

Рисунок 5. Пример работы алгоритма поиска точек в заданной области пространства для облака точек Fantasy Dragon.

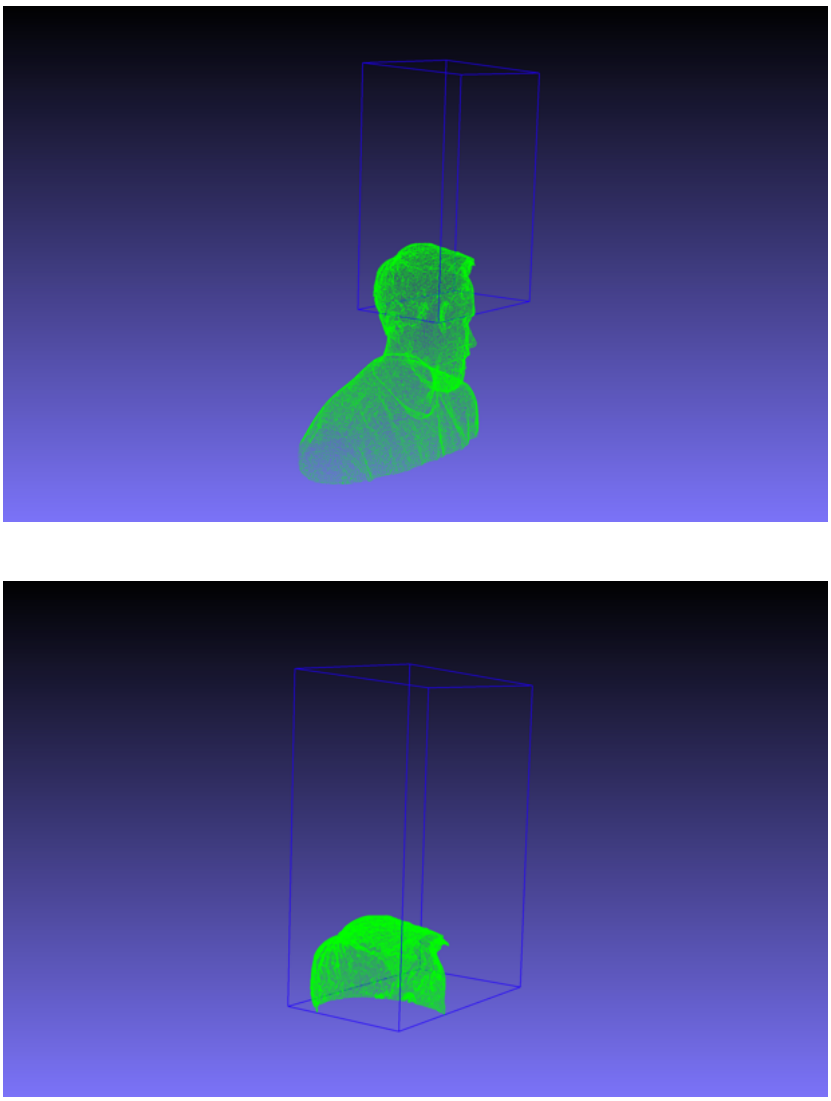

Рисунок 6. Пример работы алгоритма поиска точек в заданной области пространства для облака точек Bearded guy.

В процессе с рангом 0 определяются границы области: случайным образом, вводятся пользователем или определяются постоянными для одинаковых условий во время экспериментов. Далее эти границы передаются до листовых узлов, если хотя бы часть данных, содержащихся в этих листовых узлах, лежит внутри границ, иначе этим узлам передаются одинаковые границы пространства (пространства с нулевым объемом), это обозначает, что необходимых данных в них нет и от них не будет ожидаться передача найденных точек. После того как листовые узлы получили границы поиска, они производят поиск точек внутри этих границ точек своего локальном kd-дерева и, получив список таких точек, отправляют его родительскому узлу, те, в свою очередь, получают данные от всех своих потомков, объединяют их и отправляют своим родительским узлам. Так происходит до тех пор, пока все найденные

точки не будут получены корневым узлом (процессом с рангом 0). Таким образом процесс с рангом 0 получает полный набор точек из облака, содержащийся внутри заданной области пространства может применять его для дальнейшей работы.

Данный алгоритм может быть использован во многих областях, где используются и облака точек, при небольшом количестве доработок. Например, можно преобразовать алгоритм для поиска множества точек в произвольной области пространства (не только в параллелепипеде) и использовать найденное множество для локальных преобразований: фильтрация выбросов или триангуляция облака точек, данные преобразования будут выполняться гораздо быстрее, так как обработка будет происходить лишь на части облака точек. Из более прикладных задач можно выделить такие задачи, как выделение области облака точек для дальнейшего уточнения: увеличение плотности облака точек с помощью специальных алгоритмов. После такой операции уточненный фрагмент облака точек можно эффективнее использовать в автоматическом управлении транспортными средствами или в медицине.

6. Результаты исследования

На рис. 7, 8 и табл. 1 приведены результаты экспериментов, проведенных на системе Polus.

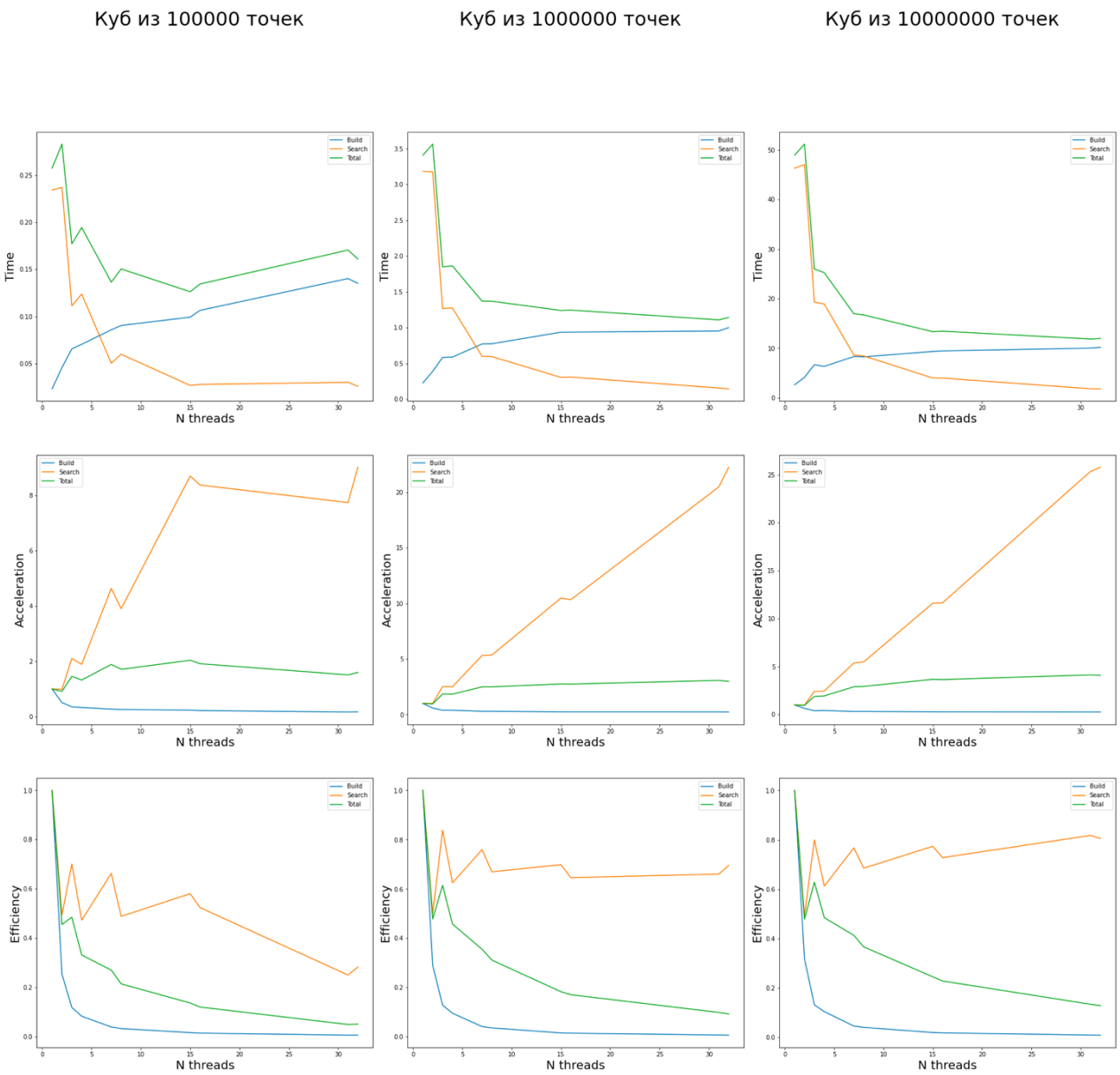

Рисунок 7. Графики времени выполнения (первая строка), ускорения (вторая строка) и параллельной эффективности (третья строка) для алгоритмов построения дерева (синим), поиска точек (оранжевым), а также программы целиком (зеленым) в зависимости от количества процессов для облаков точек: куб с 100 тыс точек, с 1 млн точек и с 10 млн точек.

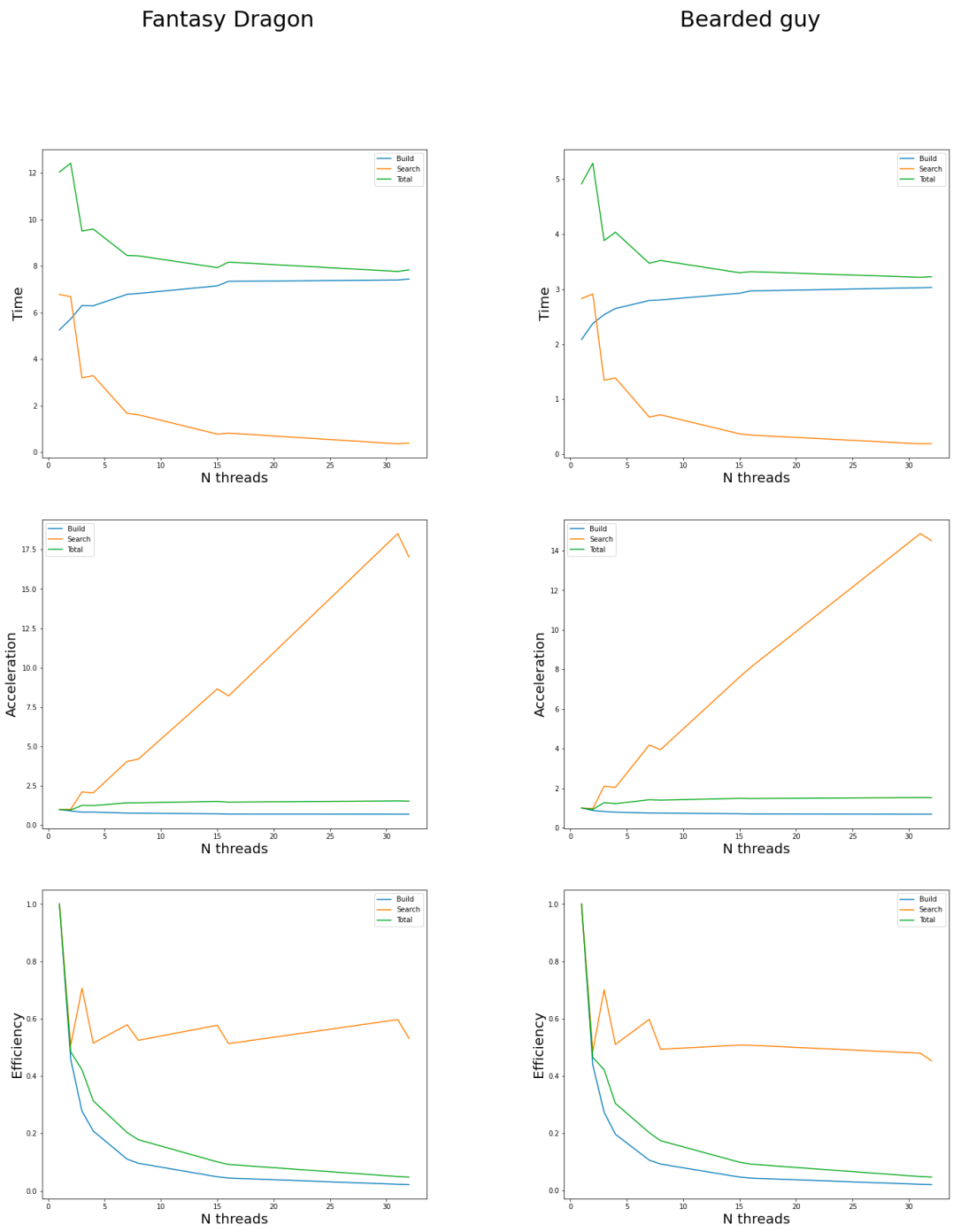

Рисунок 8. Графики времени выполнения (первая строка), ускорения (вторая строка) и параллельной эффективности (третья строка) для алгоритмов построения дерева (синим), поиска точек (оранжевым), а также программы целиком (зеленым) в зависимости от количества процессов для облаков точек Fantasy Dragon и Bearded guy.

Время выполнения операции поиска точек, сек.		Количество MPI процессов									
		1	**2**	**3**	**4**	**7**	**8**	**15**	**16**	**31**	**32**
Облако точек	**Куб 100000 точек**	0,23	0,24	0,11	0,12	0,05	0,06	0,03	0,03	0,03	0,03
	Куб 1000000 точек	3,19	3,18	1,27	1,27	0,60	0,59	0,30	0,31	0,16	0,14
	Куб 10000000 точек	46,34	47,07	19,32	18,93	8,63	8,46	4,00	3,98	1,83	1,80
	Fantasy Dragon	6,78	6,69	3,20	3,29	1,67	1,62	0,78	0,83	0,37	0,40
	Bearded guy	2,83	2,91	1,35	1,39	0,68	0,72	0,37	0,35	0,19	0,20

Таблица 1. Время выполнения операции поиска точек облака в заданной области пространства в зависимости от количества MPI процессов

Ускорение и параллельная эффективность считается по формулам:

Ускорение программы на N процессах:

$$A_N = \frac{T_N}{T_0}$$

где $T_i -$ время работы алгорима при i MPI процессах

Параллельная эффективность программы на N процессах:

$$E_N = \frac{A_N}{N}$$

По данным можно сделать несколько выводов:

- Как и следовало ожидать алгоритм построения распределенного kd-дерева замедляется при увеличении числа процессов. Это происходит из-за увеличения числа пересылок данных.
- С другой стороны, алгоритм поиска точек в заданной области пространства дает заметное ускорение. Это особенно заметно на облаках с большим количеством точек (куб из 10 млн точек против куба из 100 тыс точек), на них, в отличии от облаков с небольшим количеством, с увеличением числа процессов, эффективность остается близкой к постоянной (от 0,55 до 0,8 в зависимости от облака точек).
- Чем больше точек находится в облаке, тем более эффективно работает не только алгоритм поиска, но и вся программа в целом, это можно заметить, сравнивая результаты работы программы на облаках точек куб с 100 тыс точек и куб с 10 млн точек.

- Так же можно видеть, что в случае, когда на дереве процессов полностью заполнен самый глубокий слой (3, 7, 15 и т. д. процессов) эффективность алгоритма поиска точек резко повышается.

7. Основные результаты

Итак, в ходе работы были выполнены все поставленные задачи:

- Разработан алгоритм индексации распределенного облака точек с помощью kd-дерева.
- Алгоритм протестирован на модельной задаче поиска точек облака в заданной области пространства.
- Показано ускорение решения модельной задачи при использовании предложенного подхода.

8. Список литературы

1. **Adamsson M., Vorkapic A.** A comparison study of Kdtree, Vptree and Octree for storing neuronal morphology data with respect to performance [Отчет] : Degree Project in Computer Science / Computer Science and Communication ; KTH Royal Institute of Technology. - Stockholm : [б.н.], 2016.
2. **Aryal A. et al.** Displacement fields from point cloud data: Application of particle imaging velocimetry to landslide geodesy [Дневник]. - [б.м.] : Journal of Geophysical Research: Earth Surface, 2012 г.. - F1 : Т. 117.
3. **Bello S. A. et al.** Deep learning on 3D point clouds [Дневник]. - [б.м.] : Remote Sensing, 2020 г.. - 11 : Т. 12. - стр. 1729.
4. **Bentley J.** Multidimensional Binary Search Trees Used for Associative Searching [Дневник] // Communications of the ACM. - New YorkNYUnited States : Association for Computing Machinery, 1 сентябрь 1975 г.. - 9 : Т. 18. - стр. 509–517.
5. **Berger M. et al.** 35th Annual Conference of the European Association for Computer Graphics, Eurographics 2014-State of the Art Reports [Конференция] // State of the art in surface reconstruction from point clouds. - 2014 : The Eurographics Association.
6. **Blondel P.** The handbook of sidescan sonar [Книга]. - [б.м.] : Springer Science & Business Media, 2010.
7. **Goswami P. et al.** An efficient multi-resolution framework for high quality interactive rendering of massive point clouds using multi-way kd-trees [Дневник]. - [б.м.] : The Visual Computer, 2013 г.. - 69-83 : Т. 29.
8. **Greenspan M., Yurick M.** Approximate kd tree search for efficient ICP [Конференция] // Proceedings of the Fourth International Conference on 3-D Digital Imaging and Modeling (3DIM 2003). - Kingston, Ontario, Canada : IEEE, 2003. - стр. 442 - 448.
9. **Guttman A.** R-trees: a dynamic index structure for spatial searching [Конференция] // Proceedings of the 1984 ACM SIGMOD international conference on Management of data. - Berkeley : University of California , 1984. - стр. 47–57.
10. **Hocken R. J., Pereira P. H. (ed.).** Coordinate measuring machines and systems [Книга]. - [б.м.] : CRC press, 2016.
11. **Leberl F. et al.** Point clouds [Дневник]. - [б.м.] : Photogrammetric Engineering & Remote Sensing., 2010 г.. - 10 : Т. 76. - стр. 1123-1134.

12. **Meagher D.** Octree Encoding: A New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer [Книга]. - Troy : Rensselaer Polytechnic Institute, Image Processing Laboratory, 1980. - Т. 1.
13. **Mikhail E. M. Bethel J. S., McGlone J. C.** Introduction to modern photogrammetry [Книга]. - [б.м.] : John Wiley & Sons, 2001.
14. **Narasimhulu Y. et al.** CKD-Tree: An Improved KD-Tree Construction Algorithm [Конференция] // Proceedings of the ISIC 2021: International Semantic Intelligence Conference. - New Delhi, India : SCIS, University of Hyderabad, 2021. - стр. 211-218.
15. **Nuchter A., Lingemann K., Hertzberg J.** Cached kd tree search for ICP algorithms [Конференция] // Proceedings of the Sixth International Conference on 3-D Digital Imaging and Modeling (3DIM 2007). - Osnabruck, Germany : IEEE, 2007. - стр. 419-426.
16. **Shevtsov M., Soupikov A., Kapustin A.** Highly parallel fast KD-tree construction for interactive ray tracing of dynamic scenes [Дневник] // Computer Graphics Forum. - Oxford, UK : Blackwell Publishing Ltd, 2007 г.. - 3 : Т. 26. - стр. 395-404.
17. **Zhou H. et al.** Research on volume prediction of single tree canopy based on three-dimensional (3D) LiDAR and clustering segmentation [Дневник] // International Journal of Remote Sensing. - 2021 г.. - 2 : Т. 42. - стр. 738-755.
18. **Zhou K. et al.** Real-time kd-tree construction on graphics hardware [Дневник]. - [б.м.] : ACM Transactions on Graphics (TOG), 2008 г.. - 5 : Т. 27.