МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ М. В. ЛОМОНОСОВА
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И КИБЕРНЕТИКИ
КАФЕДРА СУПЕРКОМПЬЮТЕРОВ И КВАНТОВОЙ ИНФОРМАТИКИ

Аксой Тевфик Огузхан

Параллельное вычисление расстояния Хаусдорфа между сеточными моделями

КУРСОВАЯ РАБОТА

Научный руководитель:
Доцент кафедры СКИ , канд. физ.-мат. наук
Никольский Илья Михайлович

С текстом работы ознакомлен

_____
И. М. Никольский

Москва, 2022

Оглавление

Аннотация

В данной работе рассматриваются существующие подходы, библиотеки и программы по решению задачи вычисления расстояния между сеточными моделями и предлагается новый метод распараллелизации алгоритма NaiveHDD для ускорения вычисления расстояния Хаусдорфа между сеточными моделями, а так же эксперименты и результаты экспериментов этого параллельного алгоритма. В ходе вычислительных экспериментов, были использованы сеточные модели из работ "A Benchmark for 3D Mesh Segmentation"[1] и "A FACS Valid 3D Dynamic Action Unit database with Applications to 3D Dynamic Morphable Facial Modelling"[2].

1. Введение

Вычисление меры расстояния — это важный и фундаментальный шаг в различных областях в сферах наук и инженерств. Различные варианты мер расстояния были широко исследованы и эффективные алгоритмы были предложены в течение последних двух десятилетий[3]. В частности, благодаря их практической важности, многими исследователями уже были предложены быстрые и надежные алгоритмы для вычисления евклидова расстояния (также известного как расстояние разделения), которые считаются решенной проблемой для полигональных моделей в $\mathbb{R}^3$[4]. Напротив, меры расстояния, такие, как расстояние Хаусдорфа, подходящие для оценивания сходство между двумя полигональными моделями, были относительно плохо изучены, хотя, существует очень много приложений график и компьютерного зрения, такие, как соответствие формы, симплификацию сеток, геометрическое моделирование, рендеринг моделей, распознавание изображений и лиц, которым были бы полезны эти меры. Однако из-за высокой вычислительной сложности и сложной реализации предложенных подходов, существует очень мало алгоритмов для вычисления расстояния Хаусдорфа для полигональных моделей в $\mathbb{R}^3$. Таким образом, многие приложения обходят эту проблему: например, используя консервативные приближения[5] или с помощью других мер[6].

В большинстве приложений, количество множеств точек, полученное из трёхмерных моделей, не идентично и из-за этого возникают сложности установить взаимного соответствия между ними. По этой причине, расстояние Хаусдорфа хорошо подходит для измерение расстояния и сходства между трёхмерными моделями в практике.

Повышение эффективности алгоритма, обеспечив при этом точность вычисления расстояния Хаусдорфа является непростой задачей. Вообще говоря, мера подобия и измерения расстояния для 3D-моделей, основанная на расстоянии Хаусдорфа, сталкивается как минимум с трёмя проблемами[7].

Целью этой работы является исследование параллельного вычисления точного расстояния Хаусдорфа между 3D-моделями и обеспечить его эффективность.

1.1. Постановка задачи

Расстояние Хаусдорфа[8] — это максимальное отклонение, измеряющее насколько далеко друг от друга находятся два множества точек[9]. Даны два непустого множества точек $A = \{x_1, x_2, ..., x_n\}$ и $B = \{y_1, y_2, ..., y_m\}$, расстояние Хаусдорфа между A и B определяется как $H(A, B)$.

$$H(A, B) = \max(h(A, B), h(B, A))$$

где,

$$h(A, B) = \max_{x \in A} \left(\min_{y \in B} \|x - y\| \right)$$

$$h(B, A) = \max_{y \in B} \left(\min_{x \in A} \|y - x\| \right)$$

$H(A, B)$ обозначает расстояние Хаусдорфа в $\mathbb{R}^3$. $h(A, B)$ и $h(B, A)$ обозначают одностороннее значение расстояния Хаусдорфа с A до B и с B до A соответственно. Если $H(A, B)$ маленько число, то значит, что модели частично совпадают, а если равен нулю, то модели полностью совпадают.

1.2. Теоретическая часть

ОПРЕДЕЛЕНИЕ 1

Пусть даны два компактного множества A и B в $\mathbb{R}^3$, односторонним растоянием Хаусдорфа из A до B определяется по формуле:

$$h(A, B) = \max_{a \in A} \left(\min_{b \in B} d(\mathbf{a}, \mathbf{b}) \right),$$

где $d(\cdot, \cdot)$ обозначает оператор Эквлидового расстояния в $\mathbb{R}^3$. Тогда, двустороннее расстояние Хаусдорфа между A и B определяется по формуле:

$$H(A, B) = \max(h(A, B), h(B, A)).$$

Из верхнего определения, можно легко установить следующую теорему для полигональных моделей:

ТЕОРЕМА 1

Если A и B — полигональные модели, и $\triangle^A$ обозначает треугольник в A, тогда

$$h(A,B) = \max_{\triangle^A \in A} \Big(h(\triangle^A, B) \Big).$$

Из теоремы 1 следует, что вычисление $h(A,B)$ сводится к вычислению $h(\triangle^A, B)$.

ЛЕММА 1

Пусть даны компактные множество A, A', B и B' и $A \subseteq A'$ и $B \subseteq B'$, тогда следующие неравенства верны:

$$h(A,B') \leqslant h(A,B)$$

$$h(A,B) \leqslant h(A',B)$$

ЛЕММА 2

Пусть $v_i^a, i = (1,2,3)$ обозначает одну из трёх вершин треугольника $\triangle^A \in A$. Тогда, верхняя и нижняя грани $h(\triangle^A, B)$ можно получить следующим образом[8]:

$$\overline{h}(\triangle^A, B) = \min_{\triangle_b \in B} \Big(h\big(\triangle^A, \triangle^B\big) \Big)$$

$$\underline{h}(\triangle^A, B) = \max_{i=1,2,3} \Big(d\big(v_i^a, B\big) \Big)$$

Заметим, что нижнюю грань $\underline{h}(\triangle^A, B)$ (или $d(v_i^A, B)$) можно легко получить с помощью библиотек, как PQP(Proximity Query Package), с небольшими изменениями, поскольку это вычисление требует лишь Эвклидово расстояние из точек в объекты. Более того, поскольку $h(\triangle^A, \triangle^B) = \max_i \big(d(v_i^A, \triangle^B)\big)$, где v_i^A это вершина в $\triangle^A$[10], верхнюю грань $\overline{h}(\triangle^A, B)$ так же легко можно получить.

ТЕОРЕМА 2

Мы получаем верхнюю грань $\overline{h}$ и нижнюю грань $\underline{h}$ для $h(A,B)$ следующим образом:

$$\overline{h}(A,B) = \max_{\triangle^A \in A} \Big(\overline{h}(\triangle^A, B) \Big)$$

$$\underline{h}(A,B) = \max_{\triangle^A \in A} \Big(\underline{h}(\triangle^A, B) \Big)$$

Доказательство следует из Теоремы 1 и Леммы 2.

1.3. Существующие алгоритмы, программы и библиотеки

Naive Hausdorff Distance (Brute-force Algorithm)

Этот алгоритм является самым простым способом вычисления расстояния Хаусдорфа с точки зрения программирования. Он состоится из двух вложенных циклов, внешний цикл проходит по всем вершинам модели A, в то время как внутренний цикл проходит по всем точкам модели B. Временная сложность алгоритма — $O(m \times n)$.

Algorithm 1.1: NaiveHDD Algorithm

Input: Два конечных множества вершин A, B
Output: Одностороннее расстояние Хаусдорфа

```
1   cmax ← 0
2   for x ∈ A do
3       cmax ← ∞
4       for y ∈ B do
5           d ← ||x, y||
6           cmin ← min{cmin, d}
7       end
8       cmax ← max{cmax, cmin}
9   end
10  return cmax
```

MeshLab

В ПО MeshLab существует инструмент, с помощью которого можно вычилсить расстояние Хаусдорфа между двумя моделями. Этот инструемент доступный под меню Filters(Filters → Sampling → Hausdorff Distance). По выбору пользователя семплирует один из двух моделей и для каждой выборки находит ближайшую ему точку из другой модели. То есть, фиксируя одну точку в семплированном модели, итерируясь по всем точкам второй модели, находит ближайшую к фиксированной точке первой модел и итоговый результат выводит в консоли:

```
Hausdorff Distance computed
Sampled 334 pts (rng: 0) on ModelA.stl searched closest on
   ModelB.stl
min : 21.792477 max 162.134644 mean : 78.636330
```

EARLYBREAK (One-side Break Algorithm)

Как в [11] указывали на проблему очень большой временной сложности алгоритма по методу "грубой силы"даже для небольших входных моделей, оказывается, что проходить по всем точкам внутренней модели B не нужно, поэтому был предложен алгоритм EARLYBREAK с механизмом выхода из цикла ранее чем конца цикла когда расстояние меньше чем текущий $cmax$ находится.

Algorithm 1.2: EARLYBREAK Algorithm

Input: Два конечных множества вершин A, B
Output: Одностороннее расстояние Хаусдорфа

```
1   cmax ← 0
2   A_r ← randomize(A)
3   B_r ← randomize(B)
4   for x ∈ A_r do
5       cmin ← ∞
6       for y ∈ B_r do
7           d ← ||x, y||
8           if d < cmax then
9               cmin ← 0
10              break
11          end
12          cmin ← min{cmin, d}
13      end
14      cmax ← max{cmax, cmin}
15  end
16  return cmax
```

Худшая временная сложность, что может достичь этот алгоритм — это $O(m * n)$, где m — количество точек, которые составляют первую модель и соответственно n — количество точек во второй модели. Однако, авторы этого алгоритма в [11] формально показывали, что в практике этот случай почти никогда не встречается и среднее время выполнения алгоритма ближе к $O(m)$. Алгоритм имеет временную сложность $O(m)$ в лучшем случае, когда условие на ранний выход сразу встречается с первой итерации в внутреннем цикле.

Scipy

SciPy — это библиотека Python с открытым исходным кодом, предназначенная для решения научных и математических проблем. Она построена на базе NumPy и позволяет управлять данными, а также визуализировать их с помощью разных высокоуровневых команд.

Scipy вычисляет расстояние Хаусдорфа, используя алгоритм EARLYBREAK с подходом произвольного самплирования(random sampling).

```
from scipy.spatial.distance import directed_hausdorff
#Assume m1_v and m2_v are vertices of models
scipy_res = directed_hausdorff(m1_v, m2_v)
#scipy_res[0] => Result, [1] and [2] => Indexes
print(f"Scipy: {scipy_res[0]:.6f}\nIndexes: ({scipy_res[1]},
    {scipy_res[2]})")
```

Функция находится в библиотеке Scipy под названием directed_hausdorff. Она принимает точки(вершины) обоих моделей и возвращает Tuple(кортеж) трёх элементов, первый — результат вычисления одностороннего расстояния Хаусдорфа, второй и третьий — индексы точек в моделях соответственно, между которыми происходит вычисления. Для нахождения двустороннего расстояния Хаусдорфа достаточно применить функцию 2 раза меняя местами параметров и взять максимум среди полученных результатов.

В Scipy так же доступна функция cdist, с помощью которой можно вычислить расстояние между каждой парой двух коллекций входных данных и который позволяет использовать любую метрику расстояния в качестве параметра. Если не указать эту метрику, то ею по умольчанию является эвклидова расстояние.

2. Параллелизм

При параллелизации, результаты алгоритма Naive Hausdorff Distance больше предсказуемы, чем результаты алгоритма EARLYBREAK, потому что в алгоритме EARLYBREAK, выход из цикла, то есть встреча подходящей вершины с минимальным расстоянием, происходит разным образом для разных пар моделей и до полного окончания работы циклов при параллельной работе, мы не можем получить окончательный правильный результат расстояния Хаусдорфа. Поэтому, параллелизация алгоритма EARLYBREAK путём разбиения и распределения вершин по MPI процессам может дать значительное ускорение насчёт превосходства алгоритма, а не параллелизации.

2.1. Параллелизация алгоритма и разбиение моделей

Algorithm 2.1: Parallel Algorithm

Input: Два конечных множества вершин A, B
Output: Одностороннее расстояние Хаусдорфа

1. Загружать/Импортировать модели(.STL or .OFF)
2. Фиксировать одну из моделей, по которой будет вычислено расстояние Хаусдорфа
3. Разбить все модели по частям
4. Разослать фиксированную модель по всем процессам
5. Разослать все части всех моделей по разным процессам
6. Каждый процесс вычисляет расстояние Хаусдорфа от своей части до фиксированной модели
7. Мастер процесс получает/собирает все результаты

3. Вычислительные эксперименты

В ходе работы сделаны 3 вычислительного эксперимента. Эксперименты включали в себя измерение пространственного расстояния между сеточными моделями, оценивание подобности моделей для маленьких и больших коллекций данных. Для вычислительных экспериментов я написал программу на языке **Python3**. В программе:

- В качестве моделей были использованы сеточные модели из [1].
- Для загрузки моделей в программу была использована библиотека *trimesh*.
- Для удобной работы с точками(вершинами) моделей и их разбиение на количество процессов была использована библиотека *numpy*.
- Был использован пакет *mpi4py* для работы с **MPI** на языке Python.
- Для проверки на корректность работы параллельной программы, результаты были сравнены с результатами работы функции *directed_hausdorff* из библиотеки *scipy.spatial.distance*.

3.1. Вычислительная инфраструктура

Для запуска программы и экспериментов был использован вычислительный комплекс **IBM Polus**, состоящийся из 5 вычислительных узлов, на один из которых возложены frontend узла.

Основные характеристики каждого узла:

- 2 десятиядерных процессора IBM POWER8 (каждое ядро имеет 8 потоков) всего 160 потоков
- Общая оперативная память 256 Гбайт (в узле 5 оперативная память 1024 Гбайт) с ECC контролем
- 2 x 1 ТБ 2.5” 7K RPM SATA HDD
- 2 x NVIDIA Tesla P100 GPU, 16Gb, NVLink
- Производительность кластера (Tflop/s): 55,84 (пиковая), 40,39 (Linpack)

Для запуска программы на **IBM Polus** нужно:

1) Загрузить модули **OpenMPI** и **Anaconda**:
   - module load OpenMPI/2.1.3
   - module load Anaconda3/2019.07
2) Активировать окружение Anaconda:
   - source /polusfs/modules/anaconda/anaconda-3/etc/profile.d/conda.sh
   - conda activate

3.2. Модели экспериментов

Эксперимент 1: Пространственное расстояние

В эксперименте пространственного расстояния была создана сцена из произвольно поставленных на пространство следующих моделей:

Models	Vertices	Size
GoldenRetriever	9478	865K
Wolf	1687	135K
Horse	5538	541K
Lion	301	29K
Eagle	2135	209K
Dolphin	7573	740K
Bull	2087	204K
Dog	1480	145K
Camel	9757	953K
Giraffe	9239	903K
Human	10050	982K

Таблица 1: Количество веришн и размеры моделей

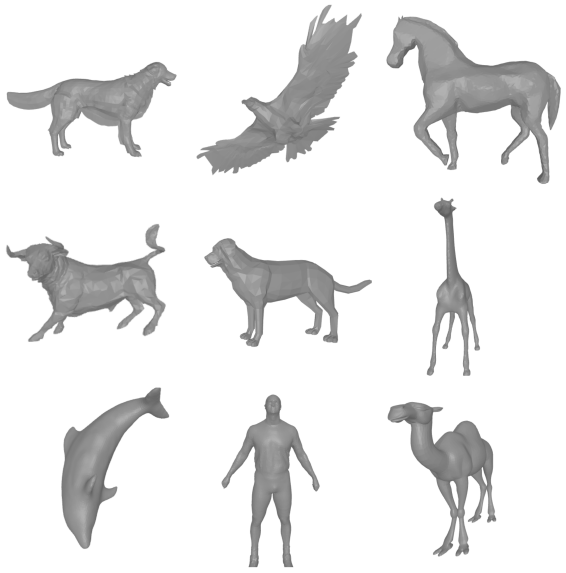

Рис. 1: Модели эксперимента пространственного расстояния

Эксперимент 2: Подобность моделей

В этом эксперименте были использованы перекрывающиеся модели голов разных людей из [1] и кружек из [2].

Models	Vertices	Size
Cup1	15002	1466K
Cup2	15198	1485K
Darren	5023	488K
Eva	5023	488K
Helen	5023	488K
James	5023	488K
Joe	5023	488K
Kornelia	5023	488K
Nat	5023	488K

Таблица 2: Количество веришны и размеры моделей

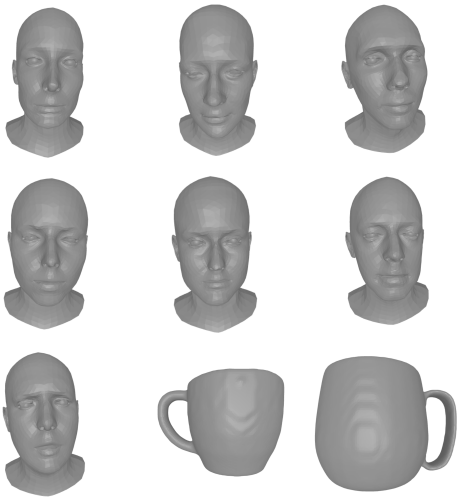

Рис. 2: Модели эксперимента подобности

Эксперимент 3: Подобность моделей в большом наборе данных

В этом эксперименте был использован весь набор модедей из [1], который составляется из 380 моделей и всего 3885011 вершина у моделей(среднее количество вершин составляет 10223 вершин).

3.3. Результаты экспериментов

Результаты эксперимента пространственного расстояния между сеточными моделями

Ниже представлена таблица[1] с временами исполнения, ускорения и эффективности программы в зависимости от количества MPI процессов.

PROCESSES	TIME (s)	SPEEDUP	EFFICIENCY
1	38990,1	1,00	1,00
2	19594,1	1,99	0,99
4	9795,5	3,98	1,00
8	4950,0	7,88	0,98
16	2575,4	15,14	0,95
32	1394,3	27,96	0,87

Таблица 3: Таблица времени исполнения, ускорения и эффективности в зависимости от количества MPI процессов

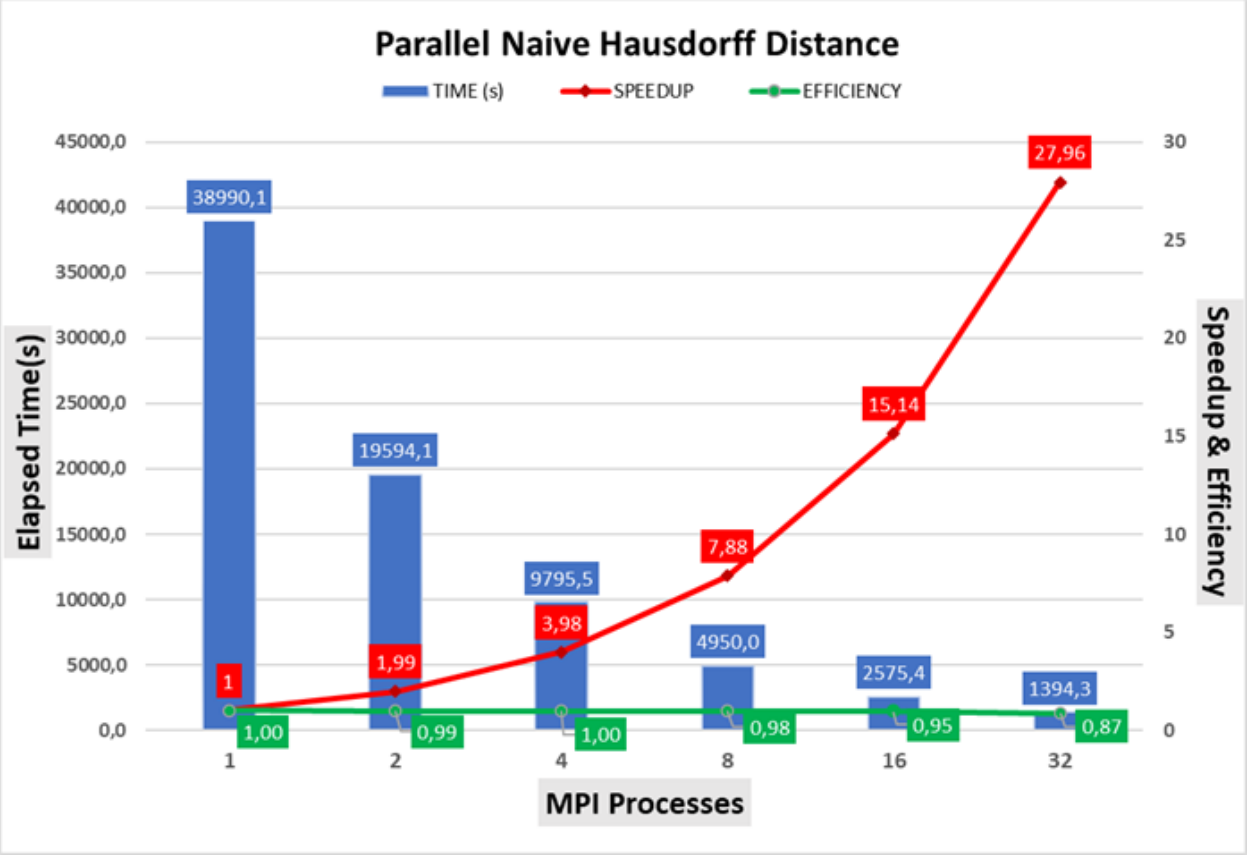

Рис. 3: График времени испольнения, ускорения и эффективности

[1]Поскольку временная сложность алгоритма зависит от количества вершин выбранных моделей и множество моделей было использовано в эксперименте, в таблице приведены итоговые времена выполнения программы для всех моделей эксперимента.

Результаты эксперимента подобности перекрывающих друг друга сеточных моделей

Ниже представлены таблицы с временами исполнения программы при применении алгоритмов NHDD[2] и EB[3], ускорения и эффективности программы в зависимости от количества MPI процессов и применённых алгоритмов.

PROCESSES	NHDD TIME (s)	EB TIME (s)
1	46682,93	303,94
2	23376,13	191,77
4	11698,77	135,84
8	5917,92	101,84
16	3098,13	89,12
32	1765,62	79,74

Таблица 4: Таблица времени исполнения разных алгоритмов зависимости от количества процессов

PROCESSES	SPEEDUP	EFFICIENCY
1	1,00	1,00
2	2,00	1,00
4	3,99	1,00
8	7,89	0,99
16	15,07	0,94
32	26,44	0,83

Таблица 5: Таблица ускорения и эффективности **NHDD** в зависимости от количества MPI процессов

PROCESSES	SPEEDUP	EFFICIENCY
1	1,00	1,00
2	1,58	0,79
4	2,24	0,56
8	2,98	0,37
16	3,41	0,21
32	3,81	0,12

Таблица 6: Таблица ускорения и эффективности **EB** в зависимости от количества MPI процессов

Можем заметить, что EARLYBREAK заметно сокращает время работы программы, но параллелизация при этом не настолько эффективно, как у работы алгоритма NaiveHDD, потому что при разбиении вершин модели в алгоритме EARLYBREAK, нет гарантии того, что во всех процессах величины, который собираются в списке, в котором возможные окончательные расстояния Хаусдорфа, раньше определяется, чем достижения конца цикла.

[2]NHDD = Naive Hausdorff Distance

[3]EB = EARLYBREAK

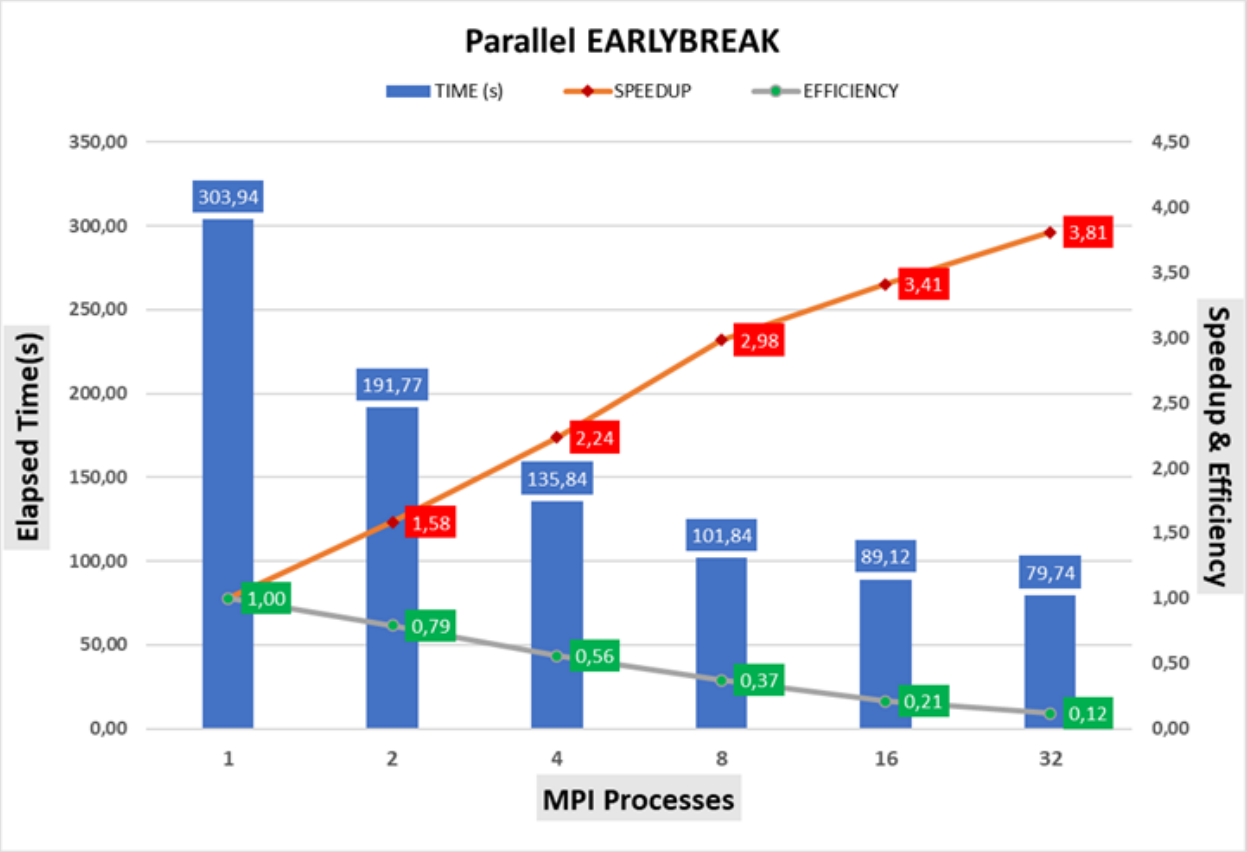

Рис. 4: График времени испольнения, ускорения и эффективности

3.4. Анализ полученных результатов

В первом эксперименте вычисление расстояния Хаусдорфа между каждыми парами моделей без параллелизации заняло **38990,1 секунд**, увеличивая количество процессов и параллелизуя программу, этот процесс ускоряется **27,96** раз при наличии **32** MPI процессов и при этом эффективность составляет **0,87**.

Во втором эксперименте сравниваем работы двух распараллелизованных алгоритмов. Сразу можно замечать, что EARLYBREAK, насчёт смены алгоритма, очень сильно сокращает время работы программы по сравнению с NaiveHDD, но параллелизация при этом не настолько эффективно, как у работы алгоритма NaiveHDD. NaiveHDD выдаёт результаты почти одинаковые во всех экспериментах. Причиной заметного сокращения эффективности выше 20 процессов может служить межузельные коммуникации.

Список Литературы

1. *Chen X.*, *Golovinskiy A.*, *Funkhouser T.* A Benchmark for 3D Mesh Segmentation // ACM Transactions on Graphics (Proc. SIGGRAPH). — 2009. — Авг. — Т. 28, № 3.

2. *Darren Cosker E. K.*, *Hilton A.* A FACS Valid 3D Dynamic Action Unit database with Applications to 3D Dynamic Morphable Facial Modelling // 13th International Conference on Computer Vision (ICCV), Barcelona, Spain. — 2011. — Т. 36, № 6. — С. 6—13.

3. *Lin M.*, *Manocha D.*, *Kim Y.* Collision and proximity queries // Handbook of Discrete and Computational Geometry. — 2003. — Т. 3, № 39.

4. *Ehmann S.*, *Lin M.* Accurate and fast proximity queries between polyhedra using convex surface decomposition // Computer Graphics Forum, Eurographics'2001). — 2001. — Т. 3, № 20. — С. 500—510.

5. Simplification envelopes / J. Cohen [и др.] // ACM Siggraph'96. — 1996. — С. 119—128.

6. *Garland M.*, *Heckbert P. S.* Surface simplification using quadric error metrics // Proc. of ACM SIGGRAPH. — 1997. — С. 209—216.

7. An efficient approach to directly compute the exact Hausdorff distance for 3D point sets / F. He [и др.] // Integrated Computer-Aided Engineering. — 2017. — № 24. — С. 261—277.

8. *Tang M.*, *Lee M.*, *Kim Y.* Interactive Hausdorff distance computation for general polygonal models // ACM T Graphic. — 2009. — Т. 3, № 28. — С. 1—9.

9. *Guthe M.*, *Borodin P.*, *Klein R.* Fast and accurate hausdorff distance calculation between meshes // International Conferences in Central Europe on Computer Graphics and Visualization. — 2005. — С. 41—48.

10. *Atallah M.* A linear time algorithm for the Hausdorff distance between convex polygons // Inf. Process. Lett. — 1983. — Т. 17. — С. 207—209.

11. *Taha A.*, *Hanbury A.* An efficient algorithm for calculating the exact Hausdorff distance // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 2015. — Т. 37, № 11. — С. 2153—2163.