

Инженерная школа информационных технологий и робототехники (ИШИТР)

Направление подготовки Искусственный интеллект и машинное обучение

Отделение школы (НОЦ) 09.04.01 Информатика и вычислительная техника

**Лабораторная работа**

По дисциплине «Параллельные и высокопроизводительные вычисления»

|  |
| --- |
| **Тема работы** |
| Определить наилучшую кластеризацию данных по индексу с помощью распределенных вычислений с использованием MPI |

Студент

|  |  |  |  |
| --- | --- | --- | --- |
| **Группа** | **ФИО** | **Подпись** | **Дата** |
| 8ВМ13 | Е.М. Якунин |  | 16.05.2022 |

Принял

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Должность** | **ФИО** | **Ученая степень, звание** | **Подпись** | **Дата** |
| К.Т.Н. Доцент (ОИТ, ИШИТР) | С.В. Аксёнов |  |  |  |

Томск – 2022 г.

# Задание

Обучающая выборка (<http://archive.ics.uci.edu/ml/datasets/Car+Evaluation>, объем выборки более 300 образцов, размерность более 3, числовые атрибуты) загружается одним из процессов. Значения атрибутов привести к диапазону от 0 до 1. Далее этот процесс отправляет загруженные данные всем остальным процессам. Все процессы начинают запускают несколько раз алгоритм кластеризации данных алгоритмом K-Means (значение k-любое). Каждый раз при этом вычисляется значение индекса качества. Среди решений выбирается лучшее (это решение имеет либо самый большой, либо самый маленький индекс – смысл выводится из уравнения индекса). Далее все лучшие индексы процессов сравниваются и процесс с лучшим результатом выводит свой результат.

Вариант 8. Индекс Дависа-Болдуина, в задании использовать MPI\_Bcast, MPI\_Reduce

# Отчет

Лабораторная работа была реализована в «PyCharm» с использованием языка программирования «Python». Для выполнения данного задания были применены библиотеки *numpy*, *mpi4py, pandas и sklearn*.

Сначала импортируем необходимые библиотеки и создаем коммуникатор:

from mpi4py import MPI  
import numpy as np  
import pandas as pd  
from sklearn.cluster import KMeans  
from sklearn.metrics import davies\_bouldin\_score  
  
comm = MPI.COMM\_WORLD  
numprocs = comm.Get\_size()  
rank = comm.Get\_rank()

Главным потоком (0-вой поток) загружаем и обрабатываем данные, и выделяем данные для кластеризации, а в остальных потоках подготавливаем область памяти для данных:

if rank == 0:  
 df = pd.read\_csv('car.data.csv', delimiter=',')  
  
 df.rename(columns={'vhigh': 'buying', 'vhigh.1': 'maint',  
 '2': 'doors', '2.1': 'persons',  
 'small': 'lug\_boot', 'low': 'safety'}, inplace=True)  
 df = df.drop('unacc', axis=1)  
  
 # Приводим все значения в таблице к численным значениям  
  
 df.loc[(df.buying == 'vhigh'), ('buying', 'maint')] = int(5)  
 df.loc[(df.buying == 'high'), ('buying', 'maint')] = int(4)  
 df.loc[(df.buying == 'med'), ('buying', 'maint')] = int(3)  
 df.loc[(df.buying == 'low'), ('buying', 'maint')] = int(2)  
  
 df.loc[(df.lug\_boot == 'big'), 'lug\_boot'] = int(4)  
 df.loc[(df.lug\_boot == 'med'), 'lug\_boot'] = int(3)  
 df.loc[(df.lug\_boot == 'small'), 'lug\_boot'] = int(2)  
  
 df.loc[(df.doors == '5more'), 'doors'] = int(5)  
 df.loc[(df.persons == 'more'), 'persons'] = int(5)  
  
 df.loc[(df.safety == 'high'), 'safety'] = int(4)  
 df.loc[(df.safety == 'med'), 'safety'] = int(3)  
 df.loc[(df.safety == 'low'), 'safety'] = int(2)  
 # print(df.head())  
  
 # Создаем выборку значений для кластеризации  
  
 x = df.values[:]  
 for i in range(len(x)):  
 for j in range(6):  
 x[i, j] = float(x[i, j]) / 5  
 y = np.array(x.flatten(), dtype=float)  
  
else:  
 y = np.zeros(shape=(10362,), dtype=float)

Засекаем время за которое потоки справятся с кластеризацией при троекратном выполнение метода K-means. Передаем всем остальным потокам выборку данных с помощью функции «MPI\_Bcast()». В каждом потоке создаем переменную, где будет храниться вычисленное значение индекса Дависа-Болдуина (ДБ), а также переменную, в которой будет в конце содержаться лучший результат индекс среди всех вычисленных потоками. Чем ближе индекс ДБ к 0, тем лучше кластеризация данных.

t\_start = MPI.Wtime()  
comm.Bcast([y, MPI.FLOAT], root=0)

index\_dbi = np.ndarray(1, dtype=float)  
index\_dbi[0] = 5  
min\_index = np.ndarray(1, dtype=float)  
min\_index[0] = 10  
  
y = np.reshape(y, (1727, 6))

Выполним кластеризацию данных и вычисление индекса ДБ. Количество кластеров будет равняться номеру потока + 2. Выведем на экран результаты наших потоков:

# Кластеризизируем данные и вычисляем индекс Дависа-Болдуина  
  
num\_clusters = rank + 2  
for i in range(3):  
 # Метод к-средних - KMeans  
 km = KMeans(init='k-means++', n\_clusters=num\_clusters, random\_state=42)  
 km.fit(y)  
 labels = km.labels\_  
 # Индекс Дависа-Болдуина  
 RS\_score = davies\_bouldin\_score(y, labels)  
 ss = float(RS\_score)  
 if ss < index\_dbi[0]:  
 index\_dbi[0] = ss  
  
print(f"I'm {rank} rank, my index\_dbi = {index\_dbi[0]} for {num\_clusters} clusters")

Синхронизируем наши потоки и применим функцию «MPI\_Reduce». Данная функцию принимаем на вход значения «index\_dbi» от каждого потока, среди всех значений отбирает меньшее «op = MPI.MIN» и присваиваем результат в область памяти «min\_index» нулевого потока. В конце выводим лучший результат и время выполнения (рис. 1):

comm.Barrier()  
comm.Reduce(index\_dbi, min\_index, op=MPI.MIN, root=0)  
  
t\_diff = MPI.Wtime() - t\_start  
  
if rank == 0:  
 print(f'Best dbi index equal: {min\_index[0]}')  
 print('Time worked programm equal: {0}'.format(t\_diff))

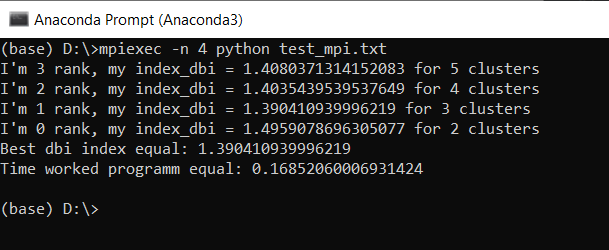


Рисунок 1 ‒ Новый красный цветовой канал для изображения 1024х768

# Результаты работы

С помощью «mpi4py» мы запустили выполнение кластеризации на нескольких потоках с различным количеством кластеров, получили index\_dbi = 1,39. Все это заняло 0,16 сек, что очень быстро, хоть у нас не очень большие данные. Думаю, при решение реальной задачи, это могло бы сократить время выполнения, если данные составляли тыс. и сотни тыс. значений.

# Выводы

В результате выполненной работы была написана небольшая программа для кластеризации данных на разных потоках с различным числом кластеров. Данная задача реализована c использованием MPI. Время выполнения кластеризации было зафиксировано. Были получены базовые знания работы с библиотекой «mpi4py», для распределения вычислений на языке Python. Познакомились с распределенными вычисления с использованием MPI.