УДК 004.75

ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ ОБРАБОТКИ ИЗОБРАЖЕНИЙ В РЕАЛЬНОМ ВРЕМЕНИ

Темирбекова Ж.Е., Толеугазы Б., Черикбаева Л.Ш.

Казахстан, г. Алматы, Казахский Национальный университет им. аль-Фараби

zhanerke_3089@mail.ru

Изображения, получаемые с помощью космических средств дистанционного зондирования Земли, играют исключительно важную роль в научных исследованиях, промышленных, хозяйственных, военных и других приложениях. Разработка космических аппаратов дистанционного зондирования и соответствующих наземных комплексов обработки изображений активно ведется во всем мире. Для анализа гиперспектральных изображений дистанционного зондирования существует много алгоритмов. Один из наиболее популярных методов кластеризации является алгоритм k means, из-за его легкой реализации, простоте, эффективности и эмпирических успехов.

Ключевые слова: k means алгоритм, кластеризация, Mapper, Reduce.

PARALLEL PROCESSING ALGORITHM IMAGES IN REAL TIME

Temirbekova Zh.E., Toleugazy B., Cherikbaeva L.Sh.

Images obtained with the help of space means of remote sensing, play an extremely important role in scientific research, industrial, economic, military and other applications. Development of remote sensing spacecraft and related ground-based imaging systems are actively conducted worldwide. For the analysis of hyperspectral remote sensing images, there are many algorithms. One of the most popular methods is a clustering algorithm means, due to its easy implementation, simplicity, effectiveness and empirical success.

Key words: k means algorithm, clustering, Mapper, Reduce.

Алгоритм k means

Основная идея алгоритма k means заключается в минимизации расстояний между объектами в кластерах. Остановка происходит, когда минимизировать расстояния больше уже невозможно. Минимизируемая функция такова: $J = \sum_{k=1}^{M} \sum_{i=1}^{N} d^2(x_i, c_k), x_i \in X$ объект кластеризации $c_j \in C$ - центр кластера. $|X| = N, |C| = M$. На момент старта алгоритма должно быть известно число C (количество кластеров). Выбор числа C может базироваться на результатах предшествующих исследований, теоретических соображениях или интуиции.

Распараллеливание k means алгоритм

k means алгоритм часто работает на очень больших наборов данных, в порядке сотни миллионов точек и десятки гигабайт данных. Поскольку он работает на таких больших наборов данных, а также из-за некоторых характеристик алгоритма, он является хорошим кандидатом для распараллеливания. Последовательный и параллельный k means алгоритм кластеризации реализовано на языке Java с применением библиотеки MPI [1]. Реализация параллельного k means алгоритма, основанный на MPI, называется MK means.

Алгоритм MK means

На первом этапе, читает N объекты из входного файла, и разделяет данные N объекты равномерно между процессами, случайным образом выбирает точки K в качестве начального центроида кластеров, а затем итеративно присваивает каждому объекту в соответствующий кластер с минимальным расстоянием. Этот процесс будет повторяться до указанного пользователем порогового значения.

Данные объекты равномерно распределяются во всех процессах и кластерные центроиды реплицируются. Глобальные операции для всех кластерных центроидов выполняется в конце каждой итерации с целью создания новых центров тяжести кластеров. И, наконец, вывод результатов кластеризации: K центроиды, I/O времени и времени кластеризации.

Алгоритм MK means:

Input: число кластеров K, число объектов данных

Output: K центроиды

1: MPI_Init // начало процедуры;

2: Чтение N объектов из файла;

3: Раздел N объектов данных равномерно между всех процессах, и предположим, что каждый процесс имеет объектам N' данных;

4: Для каждого процесса, выполнить 5-11 шаги;

5: Случайно выбрать K точек в качестве начальных центроидов кластера, обозначаемая как $\mu_k (1 \le k \le K)$;

6: Рассчитать J в формуле $J = \sum_{n=1}^{N} \sum_{k=1}^{k} \left\| x_n - c_k \right\|^2$, обозначается как J';

7: Назначение каждого объекта x_n $(1 \le n \le N)$ до ближайшего кластера;

8: Вычислить новый центр тяжести для каждого μ_k кластеру в $\mu_k = \frac{1}{N_k} \sum_{n \in c_k} x_n$;

9: Пересчитать J в $J = \sum_{n=1}^{N} \sum_{k=1}^{k} \left\| x_n - c_k \right\|^2$;

10: Повторит шаги 6-9, пока J' - J <порог (J' - J <threshold);

11: Создание кластера идентификатор для каждого объекта данных;

12: Создание новых центров тяжести кластеров в зависимости от результатов кластеризации всех процессов в конце каждой итерации;

13: Создание окончательной *Centroid* набор тяжести по функциям слияния и выводить кластеризации. Результат: K центроидов;

14: MPI_Finalize / / завершения процедуры;

k means кластеризации в MapReduce

MapReduce является моделью программирования и связанного реализации для обработки и генерации больших наборов данных. MapReduce обычно разбивает входной набор данных на независимые части. Количество частей зависит от размера набора данных и количество доступных узлов. MapReduce лучше всего подходит для обработки больших наборов данных и поэтому идеально подходит для кластеризации k means алгоритмов [2,3].

Алгоритм работает итеративно в несколько этапов, следующем образом:

1. На первом этапе, Mapper читает доля входных данных и сжимает исходный набор данных в меньший набор данных, так называемой вспомогательный кластер.

2. Каждый Mapper создает k начальный кластер из этих вспомогательных кластеров, которые затем направляются в Reduce.

3.Reduce объединяет кластеры от каждого Mapper и пересчитывает центроиды всех k кластеров.

4. Эти центры тяжести возвращаются к первоначальному Mapper по трансляции операций.

5. Теперь каждый Mapper могут использовать новые центроиды и переназначить его вспомогательных кластеров этих центров тяжести. Mapper направляет свои локальные кластеры обратно в Reduce. Reduce снова объединяет кластеры и пересчитывает центроиды.

7. Эта процедура повторяется до тех пор пока Reduce решает остановить повторную данных Mapper . Это обычно происходит, когда алгоритм сходится.

Функции Map и Reduce для алгоритма k means выглядят так (рис. 1-2):

```
public static class MapClass extends Mapper<LongWritable, Text, IntWritable,
    public void map(LongWritable key, Text value,
            Context context) throws IOException, InterruptedException
    {

        String line = value.toString();
        StringTokenizer itr = new StringTokenizer(line);
        while (itr.hasMoreTokens())
        {
            int p_id = Integer.parseInt(itr.nextToken());
            double mindis = Double.MAX_VALUE;
            int l =  Integer.parseInt(itr.nextToken());
            int a =  Integer.parseInt(itr.nextToken());
            int b =  Integer.parseInt(itr.nextToken());
            Pixel pixel = new Pixel(p_id,l,a,b);

            Pixel tmp = new Pixel();
            for (Pixel p:c_centers)
            {
                if (pixel.find_distance(p) < mindis)
                    {
                        mindis = pixel.find_distance(p);
                        tmp = p;
                    }
            }
            context.write(new IntWritable(tmp.pixel_id),new Text(tmp.toStrin
        }
    }
}
```

Рис. 1. Функция Map алгоритма кластеризации

```
   public static class Reduce extends Reducer<IntWritable, Text, IntWritable
, Text>
{
   public void reduce(IntWritable key, Iterable<Text> values,
           Context context) throws IOException, InterruptedException
                {
                    String res = "";
                    ArrayList <Pixel> p = new ArrayList  <Pixel> ();
                    Pixel average = new Pixel(-1,0,0,0);
                    int count = 0;
                   //(OLD API)while (values.hasNext())
                   for (Text val: values)
                     {
                       Pixel pixel =  new Pixel();
                       String line = val.toString();
                       StringTokenizer t = new StringTokenizer(line);
                       pixel.pixel_id = Integer.parseInt(t.nextToken());
                       pixel.L = Integer.parseInt(t.nextToken());
                       pixel.A = Integer.parseInt(t.nextToken());
                       pixel.B = Integer.parseInt(t.nextToken());
                       average = average.find_average(average, pixel, count
                       count++;
                     }
                     //output.collect(key,new Text(res));
                     res += average.pixel_id + " " + average.L + " " + avera
                     context.write(new IntWritable(count) , new Text(res));
                }
}
```

Рис. 2. Функция Reduce алгоритма кластеризации

Из моих результатов, k means является очень параллелизуемым алгоритмом, который может быть эффективно реализован на MapReduce, чтобы дать значительное ускорение по сравнению с реализации непараллельного алгоритма.

Список литературы

1. Р. Миллер, Л. Боксер. Последовательные и параллельные алгоритмы. Издательство Бином. Лаборатория знаний 2006г., 408стр.

2. J. Dean, S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. Communications of The ACM, 2008. 51(1), 107-113.

3. 5. W. Zhao, H. Ma, Q. He, “Parallel K-Means Clustering Based on MapReduce,” Cloud Computing, vol. 5931, 2009. pp. 674-679,

4. Grace Nila Ramamoorthy. K-means Clustering Using Hadoop MapReduce// Final Project Report, University College Dublin, September 16. 2011.