Домашна задача – Hadoop и Map-Reduce

Група 37

Опис на проблемот

Податоците од log датотеките можат да послужат за стекнување на многу информации кои се од големо значење. Во склоп на оваа задача ние сакаме да извлечеме корисни информации обработувајќи log датотека превземена од материјалите на udacity.org од курсот за Hadoop. Датотеката содржи 44477844 линии во следниот формат:

Ip_address	- -	timestamp	“request type	File “	bytes	cs

На пример:

10.223.157.186	- -	[15/Jul/2009:15:50:36]	"GET	RagingPhoenix_2DSleeve.jpeg “	200	168

Цел: да да ги излистаме сите IP адреси од кои бил отворен секој фајл во текот на еден месец.

Ова значи дека сакаме да добиеме излез во следниот формат:

month	file	[ip_address_1	ip_address2	ip_address_3	…]

Секој клик на веб страна креира околу 100 бајти податоци во типичен лог. Како последица на ова може да се генерираат стотици гигабајти податоци за еден ден. Издвојувањето на информациите кои ни требаат од оваа огромна маса податоци побарува повеќе од обичен секвенцијален алгоритам.

Изворен код

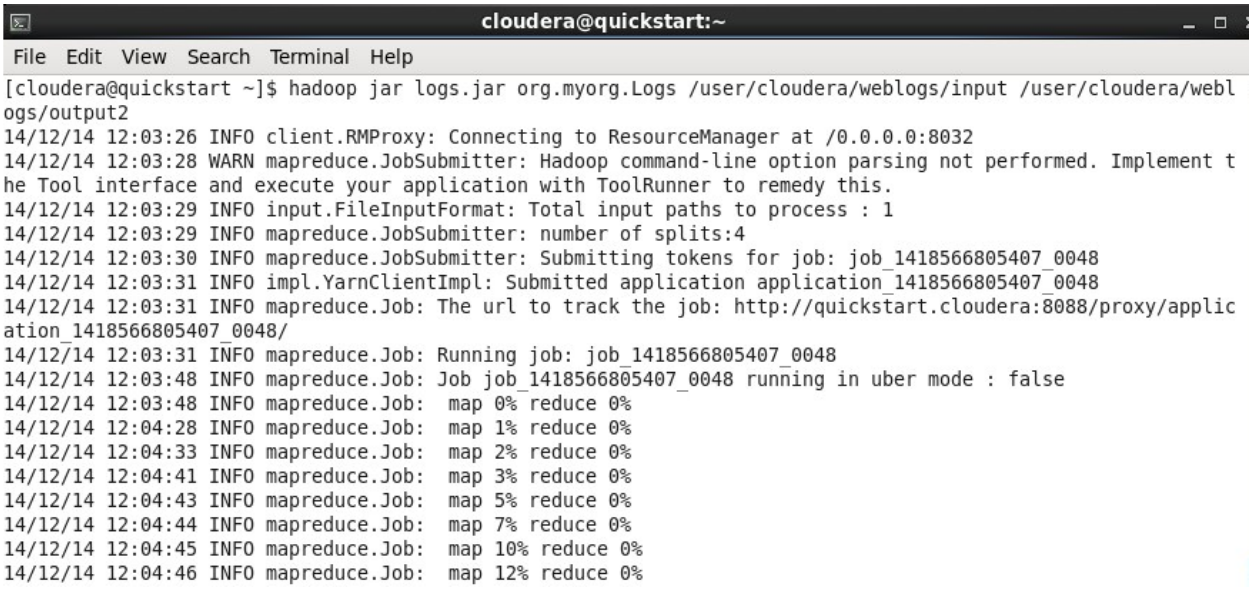
Бидејќи целта ни е клуч во мапата да се месецот и пристапениот фајл, креиравме класа FileAccess која ги содржи истите. За секој ред во влезната датотека, секој mapper креира објект од FileAccess како клуч и му доделува вредност еднаква на IP адресата од која бил направен пристапот. Потоа reducer-от ги зема сите IP адреси кои имаат ист клуч и ги става во еден TreeSet.

```
public class Logs {
	public static class FileAccess implements Comparable<FileAccess> {
		private Text file, month;
		public FileAccess() {
			this.file = new Text();
			this.month = new Text();
		}
		public void set(Text file, Text month) {
			this.file = file;
			this.month = month;
		}
		public int compareTo(FileAccess o) {
			if (file.compareTo(o.file) == 0) return (month.compareTo(o.month));
			else return (file.compareTo(o.file));
		}
		public String toString() {
			return month.toString() + "\t" + file.toString();
		}
	}
	public static class Map extends  Mapper<LongWritable, Text, FileAccess, Text> {
		private final FileAccess fileAccess = new FileAccess();
		private Text file = new Text();
		private Text ipAddress = new Text();
		private Text month = new Text();
		public void map(LongWritable key, Text value, Context context)
				throws IOException, InterruptedException {
			final String[] data = value.toString().trim().split(" ");
			if (data.length > 6) {
				ipAddress.set(data[0]);
				String[] datetime = data[3].trim().split(":");
				if (datetime.length > 0 && datetime[0].length()>1) {
					String date = datetime[0].substring(1);
					file.set(data[6]);
					String[] nums = date.split("/");
```

```
                                if (nums.length > 1) {
                                        month.set(date.split("/")[1]);
                                        fileAccess.set(file, month);
                                        context.write(fileAccess, ipAddress);
                                }
                        }
                }
        }
        public static class Reduce extendsReducer <FileAccess, Text, FileAccess, Text> {
                public void reduce(FileAccess key, Iterable<Text> values,
                                Context context) throws IOException, InterruptedException {
                        TreeSet<String> files = new TreeSet<String>();
                        for (Text val : values) files.add(val.toString());
                        context.write(key, new Text(files.toString()));
                }
        }
        public static void main(String[] args) throws Exception {
                Job job = new Job();
                job.setJobName("logs");
                job.setJarByClass(Logs.class);
                job.setMapperClass(Map.class);
                job.setReducerClass(Reduce.class);
                job.setOutputKeyClass(FileAccess.class);
                job.setOutputValueClass(Text.class);
                job.setMapOutputKeyClass(FileAccess.class);
                job.setMapOutputValueClass(Text.class);
                FileInputFormat.addInputPath(job, new Path(args[0]));
                FileOutputFormat.setOutputPath(job, new Path(args[1]));
                System.exit(job.waitForCompletion(true) ? 0 : 1);
        }
}
```

Резултати

Програмата е извршена на виртуелната машина на cloudera. На слика 1 е прикажан дел од излезот од командата за извршување. Целиот излез , кодот и податочните множества се достапни тука.



Излезна датотека

Sep /release-schedule/ sch.php [10.118.250.30, 10.220.112.1]
Oct /release-schedule/ sch.php [10.118.250.30, 10.14.237.135, 10.143.126.177, 10.95.232.88]
Oct /release-schedule/ sdh.php [10.124.148.99, 10.146.140.55, 10.175.21.83, 10.232.98.236, 10.7.189.2, 10.95.232.88]