Joining data in MapReduce is an expensive operation. Depending on the size of the datasets, you can choose to perform a **map-side** join or a **reduce-side** join. In a map-side join, two or more datasets are joined on a key in the map phase of a MapReduce job. In a reduce-side join, the mapper emits the join key, and the reduce phase is responsible for joining the two datasets. In this recipe we will demonstrate how to perform a map-side replicated join using Pig. We will join a weblog dataset, and a dataset containing a list of distinct IPs and their associated country. As the datasets will be joined in the map-phase, this will be a map-only job.

How it works...

In step 1, we called the following static method:

```
DistributedCache.addCacheFile(new URI("/user/hadoop/nobots_ip_country_
tsv.txt"), conf)
```

This method will set the `mapred.cache.files` property in the job configuration. The `mapred.cache.files` property tells the MapReduce framework to distribute the `nobots_ip_country_tsv.txt` file to every node in the cluster that will launch a mapper (and reducer if your job is configured to run reducers).

In step 2, we overrode the `setup()` method of the mapper. The `setup()` method is called by the MapReduce framework only once, prior to any calls to the `map()` method. The `setup()` method is an excellent place to perform any one-time initialization to the mapper class.

To read from the distributed cache, we used the static method `DistributedCache.getLocalCacheFiles(context.getConfiguration())` to get all of the files that have been placed, into the distributed cache. Next, we iterated over every file in the distributed cache, which was only one, and loaded the `nobots_ip_country_tsv.txt` dataset into a HashSet.

Finally, in the `map()` method, we used the HashSet loaded in the `setup()` method to join the `nobots_ip_country_tsv.txt` and the `apache_nobots_tsv.txt` files by emitting the country associated with every IP in the `apache_nobots_tsv.txt` file.

There's more...

The MapReduce framework also supports distributing archive files using the distributed cache. An archive file can be a ZIP file, GZIP file, or even a JAR file. Once the archives have been distributed to the task nodes, they will be decompressed automatically.

To add an archive to the distributed cache, simply use the `addCacheArchive()` static method of the `DistributedCache` class when configuring the MapReduce job:

```
        DistributedCache.addCacheArchive(new URI("/user/hadoop/nobots_
ip_country_tsv.zip"), conf);
```