### 分布式缓存

（1）Flink提供了一个分布式缓存，类似于hadoop，可以使用户在并行函数中很方便的读取本地文件，并把它放在taskmanager节点中，防止task重复拉取。

（2）此缓存的工作机制如下：程序注册一个文件或者目录(本地或者远程文件系统，例如hdfs或者s3)，通过ExecutionEnvironment注册缓存文件并为它起一个名称。当程序执行，Flink自动将文件或者目录复制到所有taskmanager节点的本地文件系统，仅会执行一次。用户可以通过这个指定的名称查找文件或者目录，然后从taskmanager节点的本地文件系统访问它

### 语法

(1)注册一个文件

env.registerCachedFile("hdfs:///path/to/your/file", "hdfsFile")

(2)访问数据

File myFile = getRuntimeContext().getDistributedCache().getFile("hdfsFile");

### 开发

f1.txt 中有一行数据 （执行时，会读取出来）

**package** com.ds;  
  
**import** org.apache.commons.io.FileUtils;  
**import** org.apache.flink.api.common.functions.RichMapFunction;  
**import** org.apache.flink.api.java.DataSet;  
**import** org.apache.flink.api.java.ExecutionEnvironment;  
**import** org.apache.flink.api.java.operators.DataSource;  
**import** org.apache.flink.configuration.Configuration;  
  
**import** java.io.File;  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**public class** ds6 {  
 **public static void** main(String[] args)**throws** Exception{  
 ExecutionEnvironment env = ExecutionEnvironment.*getExecutionEnvironment*();  
 env.registerCachedFile(**"D:\\fff\\f1.txt"**,**"f1"**);  
 DataSource<String> ds1 = env.fromElements(**"aa"**, **"bb"**, **"cc"**);  
 DataSet<String> reult1 =ds1.map(**new** RichMapFunction<String, String>() {  
 **private** ArrayList<String> **dataList** = **new** ArrayList<>();  
  
 @Override  
 **public void** open(Configuration parameters) **throws** Exception {  
 **super**.open(parameters);  
 *// read from file* File myFile = getRuntimeContext().getDistributedCache().getFile(**"f1"**);  
 List<String> lines = FileUtils.*readLines*(myFile);  
  
 **for**(String line:lines){  
 **dataList**.add(line);  
 System.***out***.println(**"cache: "** + line);  
 };  
 };  
  
 @Override  
 **public** String map(String value) **throws** Exception {  
 **return dataList** +**"："** + value;  
 }  
 });  
 reult1.print();  
 }  
}