将Flink DataSet中的数据写入到ElasticSearch(高级篇)

我在《将Flink DataSet中的数据写入到ElasticSearch(低级篇)》文章中介绍了如何使用Flink将DataSet中的数据写入到ElasticSearch中。正如文章标题写的，那只是低级篇，我们不会在写入大量数据的时候使用那种方法，所以我们得使用另外一种。我们肯定会想，能不能一次批量写入大量数据呢？翻翻ElasticSearch就知道，其提供了bulk API，可以帮助我们同时完成执行多个请求，比如：create，index, update以及delete。当你在处理海量数据的时候，你就可以一下处理成百上千的请求，这个操作将会极大提高效率。

如果我们去阅读一下Flink是如何将DataStream中的数据写入ElasticSearch，你会发现其就是在使用bulk API。我顺便将这个类改写成支持将DataSet写入到ElasticSearch中了，也就是实现了ElasticSearchOutputFormat，目前代码我开源到GitHub：https://github.com/397090770/flink-elasticsearch2-connector。编译的包已经上传到Maven中央仓库（可以参见《如何发布Jar包到Maven中央仓库》），这也就意味着你可以直接在pom.xml文件中使用我那个ElasticSearchOutputFormat：

```
<dependency>
    <groupId>com.iteblog</groupId>
    <artifactId>flink-elasticsearch2-connector</artifactId>
    <version>1.0.2</version>
</dependency>
```

在Scala中使用

```
import scala.collection.JavaConversions._
val config = Map("bulk.flush.max.actions" -> "1000", "cluster.name" -> "elasticsearch")
val hosts = "www.iteblog.com"

val transports = hosts.split(",").map(host => new InetSocketAddress(InetAddress.getByName(host), 9300)).toList

val data : DataSet[String] = ....
data.output(new ElasticSearchOutputFormat(config, transports, new ElasticsearchSinkFunction[String] {
    def createIndexRequest(element: String): IndexRequest = {
      Requests.indexRequest.index("iteblog").`type`("info").source(element)
    }

    override def process(element: String, ctx: RuntimeContext, indexer: RequestIndexer) {
```

```
        indexer.add(createIndexRequest(element))
    }
}))
```

在Java中使用

```
Map<String, String> config = new HashMap<>();
config.put("bulk.flush.max.actions", "1000");
config.put("cluster.name", "elasticsearch");

String hosts = "www.iteblog.com";

List<InetSocketAddress> list = Lists.newArrayList();
for (String host : hosts.split(",")) {
   list.add(new InetSocketAddress(InetAddress.getByName(host), 9300));
}

DataSet<String> data  = ....;

data.output(new ElasticSearchOutputFormat<>(config, list, new ElasticsearchSinkFunction<Stri
ng>() {
   @Override
   public void process(String element, RuntimeContext ctx, RequestIndexer indexer) {
      indexer.add(createIndexRequest(element));
   }

   private IndexRequest createIndexRequest(String element) {
      return Requests.indexRequest().index("iteblog").type("info").source(element);
   }
}));
```

我在同样的环境下测试了写入1,172,235条数据到ElasticSearch中，这次我只使用了20s不到！可见效率提高不少啊。

这个ElasticSearchOutputFormat目前只支持写入到Elasticsearch:
2.x.x，后面我会再写个支持ElasticSearch 1.x.x的版本。