Storm任务提交流程：

1.Client端提交Topology到nimbus

调用命令：

storm jar WordCount.jar com.stone.WordCountMain wordcount

实际上是调用：

java -client WordCount.jar com.stone.WordCountMain wordcount

2.通过TopologyBuilder将Spout、Bolt按照一定的逻辑顺序构建Topology程序。

```java
TopologyBuilder builder = new TopologyBuilder();
//RandomSentenceSpout类，在已知的英文句子中，随机发送一条句子出去。
builder.setSpout("spout1", new RandomSentenceSpout(), 3);
// SplitSentenceBolt类，主要是将一行一行的文本内容切割成单词
builder.setBolt("split1", new SplitSentenceBolt(), 9).shuffleGrouping("spout1");
// WordCountBolt类，对单词出现的次数进行统计
builder.setBolt("count2", new WordCountBolt(),3).shuffleGrouping("split1");
```

3.调用TopologyBuilder的createTopology()方法，获取StormTopology实例对象。源码如下：

```java
// STONE_NOTE 调用TopologyBuilder的此方法，创建StormTopology的实例对象
public StormTopology createTopology() {
    Map<String, Bolt> boltSpecs = new HashMap<String, Bolt>();
    Map<String, SpoutSpec> spoutSpecs = new HashMap<String, SpoutSpec>();
    for (String boltId : _bolts.keySet()) {
        IRichBolt bolt = _bolts.get(boltId);
        ComponentCommon common = getComponentCommon(boltId, bolt);
        boltSpecs.put(boltId, new Bolt(ComponentObject.serialized_java(Utils.javaSerialize(bolt)), common));
    }
    for (String spoutId : _spouts.keySet()) {
        IRichSpout spout = _spouts.get(spoutId);
        ComponentCommon common = getComponentCommon(spoutId, spout);
        spoutSpecs.put(spoutId, new SpoutSpec(ComponentObject.serialized_java(Utils.javaSerialize(spout)), common));

    }
    // STONE_NOTE 将Spout和Bolt的相关信息都封装在对应的map中，然后获取StormTopology实例对象
    return new StormTopology(spoutSpecs, boltSpecs, new HashMap<String, StateSpoutSpec>());
}
```

4.开始提交任务，具体过程如下：

（1）调用StormSubmitter.*submitTopologyWithProgressBar*("WordCount", conf, builder.createTopology())提交任务。

*submitTopologyWithProgressBar*的源码如下：

```
// STONE_NOTE 调用此方法提交任务
public static void submitTopologyWithProgressBar(String name, Map stormConf, StormTopology topology, SubmitOptions opts) throws AlreadyAliveException,
        InvalidTopologyException {

    /**
     * remove progress bar in jstorm
     */
    // STONE_NOTE 调用submitTopology方法，传入Topology的名称、配置参数、实例对象
    submitTopology(name, stormConf, topology, opts);
}
```

（2）在*submitTopologyWithProgressBar*方法中，调用了`StormSubmitter`的`submitTopology(name, stormConf, topology, opts)`方法。

submitTopology方法的源码如下：

```
public static void submitTopology(String name, Map stormConf, StormTopology topology, SubmitOptions opts) throws AlreadyAliveException,
            InvalidTopologyException {
    // STONE_NOTE 检验Stormconf，必须是json-serializable Json的序列化对象
    if (!Utils.isValidConf(stormConf)) {
        throw new IllegalArgumentException("Storm conf is not valid. Must be json-serializable");
    }
    // STONE_NOTE 利用stormConf创建一个hashmap的实例，并传给stormConf
    stormConf = new HashMap(stormConf);
    // STONE_NOTE 获得命令行参数，并放入stormConf中
    stormConf.putAll(Utils.readCommandLineOpts());
    Map conf = Utils.readStormConfig();
    conf.putAll(stormConf);
    putUserInfo(conf, stormConf);
    try {
        String serConf = Utils.to_json(stormConf);
        if (localNimbus != null) {
            LOG.info("Submitting topology " + name + " in local mode");
            // STONE_NOTE 如果localNimbus不为空的话，调用本地模式运行
            localNimbus.submitTopology(name, null, serConf, topology);
        } else {
            // STONE_NOTE 通过Topology的配置信息，获取到NimbusClient
            NimbusClient client = NimbusClient.getConfiguredClient(conf);
            try {
                // STONE_NOTE 检测Topology的名称在集群上是否存在
                if (topologyNameExists(client, conf, name)) {
                    // STONE_NOTE 如果已经存在，抛出异常；提示Topology的名称已存在。
                    throw new RuntimeException("Topology with name `" + name + "` already exists on cluster");
                }
                // STONE_NOTE 调用submitJar方法，提交jar文件
                submitJar(client, conf);
                LOG.info("Submitting topology " + name + " in distributed mode wit
h conf " + serConf);
                // STONE_NOTE 否则的话，调用分布式集群模式
                if (opts != null) {
                    // STONE_NOTE 新的提交方式，携带opts参数 提交Topology任务
                    client.getClient().submitTopologyWithOpts(name, path, serConf, topology, opts);
                } else {
                    // this is for backwards compatibility
                    // STONE_NOTE 这个是为了兼容之前的版本 默认将opts设置为ACTIVE
                    client.getClient().submitTopology(name, path, serConf, topology);
                }
            } finally {
                client.close();
            }
        }
        LOG.info("Finished submitting topology: " + name);
    } catch (InvalidTopologyException e) {
        LOG.warn("Topology submission exception", e);
        throw e;
    } catch (AlreadyAliveException e) {
        LOG.warn("Topology already alive exception", e);
        throw e;
    } catch (TopologyAssignException e) {
        LOG.warn("Failed to assign " + e.get_msg(), e);
        throw new RuntimeException(e);
    } catch (TException e) {
        LOG.warn("Failed to assign ", e);
        throw new RuntimeException(e);
    }
}
```

在submitTopology()方法中,做了一下工作：

1）检验Stormconf，必须是json-serializable Json的序列化对象

```
    Utils.isValidConf(stormConf)
```

2）判断Topology的运行模式

```
     // STONE_NOTE 如果localNimbus不为空的话，调用本地模式运行

    localNimbus.submitTopology(name, null, serConf, topology);
```

3）如果为分布式集群模式运行

```
// STONE_NOTE 检测Topology的名称在集群上是否存在

topologyNameExists(client, conf, name)

// STONE_NOTE 调用submitJar方法，提交jar文件

submitJar(client, conf);
```

```
// STONE_NOTE 新的提交方式，携带opts参数 提交Topology任务
client.getClient().submitTopologyWithOpts(name, path, serConf, topology, opts);
```

最终任务提交完成！