ToySpark

Documentation

2019/11/29

郝天翔　　张洋

整体结构

项目结构

- 节点分为 master 和 worker 两类
  master 管理 worker，计算结果在 master 上得到
- 运行时每个节点有 manager 和 executor 两类线程
  manager 负责通信和协调，executor 负责计算
- 每个 executor 负责计算一个 stage 中的一个 partition 对应的部分
  何时开启：stage 开始时
  何时结束：自己的数据已经被下游 stage 的所有 executor 取走

代码结构

```
main/scala/toyspark
├── Action.scala ............... Action 处理逻辑、manger 线程运行逻辑等
├── Communication.scala ........ 节点通信模块
├── Context.scala .............. 每个节点共享的 Context
├── Dataset.scala .............. 各种 Dataset 的定义
├── Executor.scala ............. Transformation 处理逻辑、executor 线程运行逻辑等
├── StageSplit.scala ........... Stage 划分模块
├── TypeAliases.scala
└── utilities
    ├── Config.scala ........... 配置读入与解析
    ├── HDFSUtil.scala ......... HDFS 相关
    └── SocketWrapper.scala
```

API 设计

- 用户需要手动配置各个节点
  通过配置文件确定节点间通信地址、HDFS 相关配置
  通过命令行参数确定身份
- 大体上模仿 Spark 的 API，用户只需提供运算逻辑
- 目前已实现：
  - **数据输入**：`generate`、`read`
  - **Transformation**：`map`、`filter`、`coalesce`
  - **Action**：`reduce`、`collect`、`count`、`take`
  - **数据输出**：`saveAsSequenceFile`

配置文件示例

```
{
  "master": {
    "ip": "172.21.0.20",
    "port": "23333"
  },
  "workers": [
    { "ip": "172.21.0.4" },
    { "ip": "172.21.0.33" }
  ],
  "hdfs": {
    "coreSitePath": "/home/ubuntu/hadoop-2.7.7/etc/hadoop/core-site.xml",
    "url": "hdfs://master:9000"
  }
}
```

命令行参数示例

```
$ java -jar Toy-Spark-assembly-0.2.jar 0
```

```
$ java -jar Toy-Spark-assembly-0.2.jar 1
```

main 函数示例

```
def main(args: Array[String]): Unit = {
  Communication.initialize(args)
  val pi = Dataset
    .generate(
      List(4, 4, 4),
      (_, _) => (0 until 1000000).map(
        _ => (Random.nextDouble(), Random.nextDouble())).toList)
    .map({ case (x, y) => (x * 2 - 1, y * 2 - 1) })
    .coalesced(List(8, 8, 8))
    .filter({ case (x, y) => x * x + y * y < 1 })
    .map(_ => 1)
    .reduce((x, y) => x + y, 0)
  println(s"The result of Pi is ${pi / (1000000.0 * 12) * 4}")
}
```

```
The result of Pi is 3.141732333333333
```

目前进度

已完成 | 进行中 | 废弃

- 节点之间可进行通信和协作
- 主节点自动划分任务
- 惰性计算
- 简单的 transformations 和 actions 操作
- 实现更多的 transformations 和 actions 操作
- 简单的需要进行 shuffle 的操作
- 支持 RDD 之间的依赖关系为简单的线性关系
- 支持 RDD 之间的依赖关系为 DAG
- 支持从 HDFS 读入数据和写入数据到 HDFS
- 内存不够时使用磁盘进行交换
- 在特定目标上运行效率接近 Spark
- 性能优化（使用线程池和连接池等）

Demo 演示

Thanks

Q & A ?