[spark-issue1] 根据任务并发数调整上传HDFS文件副本数

需求描述

这个需求是从MR任务而来，在MR任务中，不管map数多少，上传至HDFS的用户应用jar包副本数均为10，这对一些小任务（map数很少）来说，明显多余了。但是对于一些大任务来说，比如（数千map并发）来说，如果用户jar包size达到百兆以上，数千并发的从10台机器下载应用包，又会对DataNode产生很大的压力，可能会将单点DataNode的网络带宽打满，影响其他任务。

需求实现

MR任务在上传用户jar包至HDFS之前无法准确计算并发数，所以并发数取了map数和reduce数的最大值。副本数的解决方案是增加多分支判断确定具体数值，取消一刀切10副本策略：

- 20并发以下，取默认值3；
- 20-50并发，副本数取5；
- 1000并发以上，取副本数30；
- 其他情况，取副本数10。

Spark这边稍微复杂一点。因为针对spark任务来说，在任务执行开始时，申请的executor数目是已知的（只针对生产yarn-cluster模式，且用户不是通过代码设置executor数目的情况）。所以从HDFS拉文件的并发数是相对确定的，近似是executor的数目。Spark任务向HDFS上传的文件包括：spark-submit命令中指定的 `--jars` ， `--files` 和用户应用包 `userJars` ；配置文件 `spark-defaults.conf` 指定的 `spark.yarn.dist.files` ， `spark.yarn.dist.jars` 等。Spark确定副本数的算法如下：

首先单DataNode节点的最大支持带宽可设置：默认100，单位MB，参数为 `spark.hdfs.datanode.maxnio` 。

用户自定义副本数可设置，spark本身提供该参数： `spark.yarn.submit.file.replication`

- 如果用户设置了该参数，不进行副本数计算，副本数取用户设置的参数值。
- 如果用户未设置该参数，进行计算，公式：

$$\text{实际所需副本数（}calculateReplication\text{）} = \frac{executor\text{数目} * \text{上传文件总大小}}{DataNode\text{最大支持带宽}}$$

- 根据 `calculateReplication` 值进行判断，如果小于3，取默认值3；如果大于最大副本限制30，取30；其他情况取计算出来的实际值。

patch

```
val replication = if (sparkConf.contains("spark.yarn.submit.file.replicat
ion")) {
  sparkConf.get(STAGING_FILE_REPLICATION).map(_.toShort)
    .getOrElse(fs.getDefaultReplication(destDir))
} else {
  List(
    JARS_TO_DISTRIBUTE, FILES_TO_DISTRIBUTE, ARCHIVES_TO_DISTRIBUTE
  ).foreach { key =>
    sparkConf.get(key).foreach {
      distributeFilesString =>
        distributeFilesString.trim.split(",").foreach {
          filePath =>
            localFilesSize +=
              Utils.getFileLength(new File(filePath.split(":")(1)), spark
Conf)
        }
    }
  }

  if (args.userJar != null) {
    localFilesSize += Utils.getFileLength(new
File(args.userJar.split(":")(1)), sparkConf)
  }

  val dataNodeMaxNIO = sparkConf.get("spark.hdfs.datanode.maxnio",
"100").toInt
  val calculateReplication = localFilesSize / (1024 * 1024) *
    sparkConf.get(EXECUTOR_INSTANCES).get / dataNodeMaxNIO

  val rep = if (calculateReplication < defaultReplication) {
    defaultReplication
  } else if (calculateReplication > MAX_HDFS_REPLICATION) {
    MAX_HDFS_REPLICATION
  } else {
    calculateReplication
  }
  rep
}
```