浅析Spark推测执行机制

caolei

Exported on 01/10/2020

Table of Contents

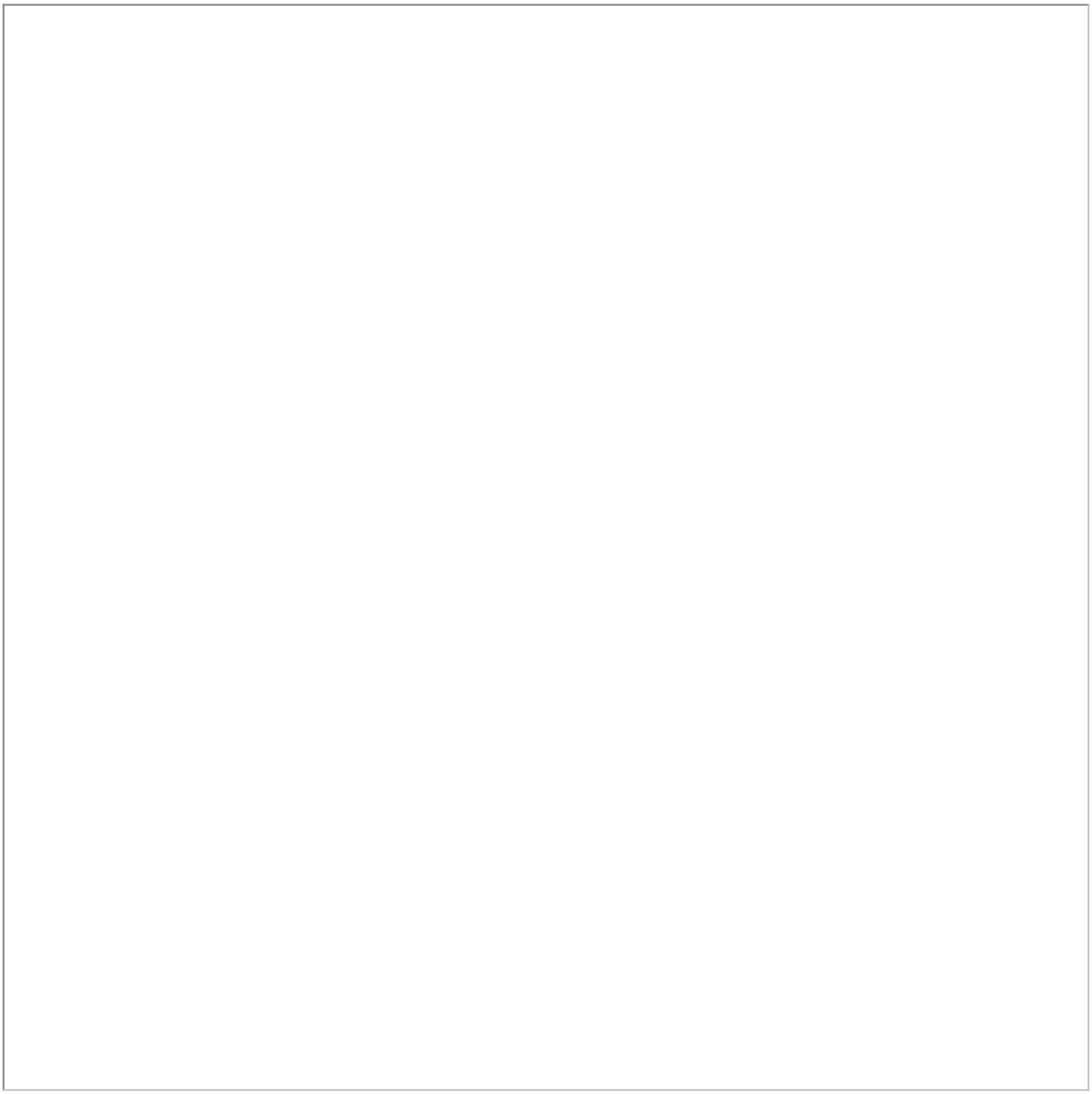
1 背景

我们都知道，Hadoop生态系统中的一个核心思想是分而治之: 将计算任务进行分解，分发到多个节点上去执行，然后再把多个节点上的计算结果进行汇总聚合，进而得到最终结果。分而治之使任务可以被并行处理，进而加快了任务的处理速度。

因为存在多个处理节点，每个节点分配到的任务可能或者处理任务的速度不是完全相同的，这种现象被称为负载不均衡或者资源分布不均。这种现象的出现会导致各个任务运行时间的不一致，甚至会出现一个Task明显慢于同一Job的其它Task。

为了解决这个问题，Hadoop的MapReduce组件实现了一个被称为推测执行(Speculation Execution)的功能机制，当推测执行机制被开启后，MapReduce会检测每个任务的执行情况，当发现某个任务执行较慢时，MapReduce将会启动一个冗余(输入、算子、输出全是一样的)的任务来并行执行，两个任务中只要有一个任务完成，就说明此任务执行完成。下图(图片来自https://data-flair.training/)大致展示了MapReduce的推测执行过程。

关于MapReduce的推测执行机制，之前已经介绍过，我们今天要介绍的是在Spark中的推测执行机制。



2 Spark推测执行

推测执行机制在Spark和MapReduce的实现思路虽然大致一样，但实现上还是有些差异的，下面我们看看Spark中的具体实现。

2.1 推测执行参数

下表列出了Spark(2.4.2版本)中关于推测执行的所有参数、默认状态及参数说明。

参数	默认状态	说明	中文解释
spark.speculation	false	If set to "true", performs speculative execution of tasks. This means if one or more tasks are running slowly in a stage, they will be re-launched.	是否打开推测执行模式
spark.speculation.interval	100ms	How often Spark will check for tasks to speculate.	打开会推测执行模式后，调度器检查是否有任务需要被推测执行的时间间隔
spark.speculation.multiplier	1.5	How many times slower a task is than the median to be considered for speculation.	定义任务执行多慢才会被推执行；默认情况下是某个任务的执行速度比其他任务的执行速度中值慢1.5倍就会被推测执行。
spark.speculation.quantile	0.75	Fraction of tasks which must be complete before speculation is enabled for a particular stage.	推测执行的作用域是Stage, 一个State的任务开始执行后，并不是马上就启用推测执行，而是当Stage中完成的任务数占比超过某个阈值。这个阈值的默认值是0.75

2.2 推测执行实现

上表解释了Spark中关于推测执行参数的含义，接下来我们看看Spark是如何使用这些参数呢，搞清楚Spark是如何使用这些参数的，也就搞清楚了Spark的推测执行机制。

推测执行是伴随着任务调度器(TaskSchedulerImpl)启动而启动的，其中 **start()** 方法中用到了参数 **spark.speculation** 和 **spark.speculation.interval**

```
// 类名：org.apache.spark.scheduler.TaskSchedulerImpl


override def start() {
  backend.start()
  // 非本地调度后端且推测执行参数设置为true
  if (!isLocal && conf.getBoolean("spark.speculation", false)) {
    logInfo("Starting speculative execution thread")
    // 启动推测执行线程
    speculationScheduler.scheduleWithFixedDelay(new Runnable {
      override def run(): Unit = Utils.tryOrStopSparkContext(sc) {
        // 检查所有的处于活跃状态的任务
        checkSpeculatableTasks()
      } // SPECULATION_INTERVAL_MS 是 spark.speculation.interval的值
    }, SPECULATION_INTERVAL_MS, SPECULATION_INTERVAL_MS, TimeUnit.MILLISECONDS)
  }
}
// How often to check for speculative tasks
val SPECULATION_INTERVAL_MS = conf.getTimeAsMs("spark.speculation.interval", "100ms")
```

start()中只做了一些简单的判断和线程的启动，核心逻辑应该都在 **checkSpeculatableTasks()**中

在org.apache.spark.scheduler.TaskSchedulerImpl#checkSpeculatableTasks 中，并没有实际的业务逻辑，而是放到了org.apache.spark.scheduler.TaskSetManager中

```
// 类名：org.apache.spark.scheduler.TaskSchedulerImpl


// 检查所有的处于活跃状态的任务
def checkSpeculatableTasks() {
  var shouldRevive = false
  synchronized {
    // 判断是否有需要推测执行的任务
    shouldRevive = rootPool.checkSpeculatableTasks(MIN_TIME_TO_SPECULATION)
  }
  if (shouldRevive) {
    backend.reviveOffers()
  }
}


// 类名：org.apache.spark.scheduler.TaskSetManager
override def checkSpeculatableTasks(minTimeToSpeculation: Int): Boolean = {
  // Can't speculate if we only have one task, and no need to speculate if the task set is a
  // zombie.
  if (isZombie || numTasks == 1) {
    return false
  }
  var foundTasks = false
  // 计算启动推测执行需要完成任务数的最小值
  val minFinishedForSpeculation = (SPECULATION_QUANTILE * numTasks).floor.toInt
  logDebug("Checking for speculative tasks: minFinished = " + minFinishedForSpeculation)

  // 如果成功的任务数大于上面计算的阈值并且成功的任务数大于0，进入推测执行检查
  if (tasksSuccessful >= minFinishedForSpeculation && tasksSuccessful > 0) {
    val time = clock.getTimeMillis()
    // 成功执行Task的执行成功时间的中位数
    val medianDuration = successfulTaskDurations.median
    // 取中位数的SPECULATION_MULTIPLIER倍和minTimeToSpeculation的最大值作为阈值threshold
    val threshold = max(SPECULATION_MULTIPLIER * medianDuration, minTimeToSpeculation)
    // TODO: Threshold should also look at standard deviation of task durations and have a lower
    // bound based on that.
    logDebug("Task length threshold for speculation: " + threshold)
    // 遍历所有需要判断推测执行的task
    for (tid <- runningTasksSet) {
      val info = taskInfos(tid)
      val index = info.index
          // 放入推测执行任务列表的条件：任务为成功、任务正在执行、任务执行时间超过threshold且未在推测执行任务列表
      if (!successful(index) && copiesRunning(index) == 1 && info.timeRunning(time) > threshold &&
        !speculatableTasks.contains(index)) {
        logInfo(
          "Marking task %d in stage %s (on %s) as speculatable because it ran more than %.0f ms"
            .format(index, taskSet.id, info.host, threshold))
        speculatableTasks += index
        sched.dagScheduler.speculativeTaskSubmitted(tasks(index))
        foundTasks = true
      }
    }
  }
```

```
  foundTasks
}


// Quantile of tasks at which to start speculation
val SPECULATION_QUANTILE = conf.getDouble("spark.speculation.quantile", 0.75)
val SPECULATION_MULTIPLIER = conf.getDouble("spark.speculation.multiplier", 1.5)
```

在这个类(org.apache.spark.scheduler.TaskSetManager)中，我们找到了另外两个参数的使用逻辑

检测出来的被推测执行的任务会通过org.apache.spark.scheduler作为普通任务进行调度。两个任务谁先完成，另外一个任务都将会被终止。