



过往记忆

文章总数：825

浏览总数：9,885,423

评论：4919

分类目录：94 个

注册用户数：2607

最后更新：2017年6月2日



欢迎关注微信公共帐号：

iteblog_hadoop

Spark Streaming性能调优详解



Spark



2015-04-28



18030



0评论

下载为PDF

为什么不允许复制

Spark Streaming提供了高效便捷的流式处理模式，但是在有些场景下，使用默认的配置达不到最优，甚至无法实时处理来自外部的数据，这时候我们就需要对默认的配置进行相关的修改。由于现实中场景和数据量不一样，所以我们无法设置一些通用的配置（要不然Spark Streaming开发者就不会弄那么多参数，直接写死不得了），我们需要根据数据量，场景的不同设置不一样的配置，这里只是给出建议，这些调优不一定适用于你的程序，一个好的配置是需要慢慢地尝试。

1、设置合理的批处理时间(batchDuration)。

在构建StreamingContext的时候，需要我们传进一个参数，用于设置Spark Streaming批处理的时间间隔。Spark会每隔batchDuration时间去提交一次Job，如果你的Job处理的时间超过了batchDuration的设置，那么会导致后面的作业无法按时提交，随着时间的推移，越来越多的作业被拖延，最后导致整个Streaming作业被阻塞，这就间接地导致无法实时处理数据，这肯定不是我们想要的。

另外，虽然batchDuration的单位可以达到毫秒级别的，但是经验告诉我们，如果这个值过小将会导致因频繁提交作业从而给整个Streaming带来负担，所以请尽量不要将这个值设置为小于500ms。在很多情况下，设置为500ms性能就很不错了。

那么，如何设置一个好的值呢？我们可以先将这个值位置为比较大的值（比如10S），如果我们发现作业很快被提交完成，我们可以进一步减小这个值，知道Streaming作业刚好能够及时处理完上一个批处理的数据，那么这个值就是我们要的最优值。





微信扫一扫，加关注

即可及时了解Spark、Hadoop或者Hbase等相关的文章

欢迎关注微信公共帐号: iteblog_hadoop

过往记忆博客(<http://www.iteblog.com>)
专注于Hadoop、Spark、Flume、Hbase等技术的博客，欢迎关注。

Hadoop、Hive、Hbase、Flume等交流群: 138615359和149892483

如果想及时了解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号: iteblog_hadoop

2、增加Job并行度

我们需要充分地利用集群的资源，尽可能的将Task分配到不同的节点，一方面可以充分利用集群资源；另一方面还可以及时的处理数据。比如我们使用Streaming接收来自Kafka的数据，我们可以对每个Kafka分区设置一个接收器，这样可以达到负载均衡，及时处理数据（关于如何使用Streaming读取Kafka中的数据，可以参见《[Spark Streaming和Kafka整合开发指南\(一\)](#)》和《[Spark Streaming和Kafka整合开发指南\(二\)](#)》）。

再如类似reduceByKey()和Join函数都可以设置并行度参数。

3、使用Kryo序列化。

Spark默认的是使用Java内置的序列化类，虽然可以处理所有自继承java.io.Serializable的类序列化的类，但是其性能不佳，如果这个成为性能瓶颈，可以使用Kryo序列化类，关于如何在Spark中使用Kryo，请参见《[在Spark中自定义Kryo序列化输入输出API](#)》。使用序列化数据可以很好地改善GC行为。

4、缓存需要经常使用的数据

对一些经常使用到的数据，我们可以显式地调用 `rdd.cache()` 来缓存数据，这样也可以加快数据的处理，但是我们需要更多的内存资源。

5、清除不需要的数据

随着时间的推移，有一些数据是不需要的，但是这些数据是缓存在内存中，会消耗我们宝贵的内存资源，我们可以通过配置 `spark.cleaner.ttl` 为一个合理的值；但是这个值不能过小，因为如果后面计算需要用的数据被清除会带来不必要的麻烦。而且，我们还可以配置选项 `spark.streaming.unpersist` 为true(默认就是true)来更智能地去持久化(unpersist) RDD。这个配置使系统找出那些不需要经常保有的RDD，然后去持久化它们。这可以减少Spark RDD的内存使用，也可能改善垃圾回收的行为。



6、设置合理的GC

GC是程序中最难调的一块，不合理的GC行为会给程序带来很大的影响。在集群环境下，我们可以使用并行Mark-Sweep垃圾回收机制，虽然这个消耗更多的资源，但是我们还是建议开启。可以如下配置：

```
spark.executor.extraJavaOptions=-XX:+UseConcMarkSweepGC
```

更多的关于GC行为的配置，请参考Java垃圾回收相关文章。这里就不详细介绍了。

7、设置合理的CPU资源数

很多情况下Streaming程序需要的内存不是很多，但是需要的CPU要很多。在Streaming程序中，CPU资源的使用可以分为两大类：（1）、用于接收数据；（2）、用于处理数据。我们需要设置足够的CPU资源，使得有足够的CPU资源用于接收和处理数据，这样才能及时高效地处理数据。

Hadoop大数据学习的秘密



本博客文章除特别声明，全部都是原创！

禁止个人和公司转载本文、谢谢理解：过往记忆 (<https://www.iteblog.com/>)

本文链接：【Spark Streaming性能调优详解】 (<https://www.iteblog.com/archives/1333.html>)

♡ 喜欢 (15)

赏

✎ 分享 (0)



《 深入浅出Hive视频教程百度网盘免费下载

OpenCloud 2015大会PPT资料免费下载[Spark篇]



[电子书]Machine Learning with Spark Second Edition PDF下载	Spark sql解析异常 java.lang.StackOverflowError 处理	解决Spark shell模式下初始化Job出现的异常	[电子书]Mastering Spark for Data Science PDF下载
---	---	-----------------------------	---

[电子书]Machine Learning with Spark

Spark sql解析异常 java.lang.StackOverflowError

解决Spark shell模式下初始化Job出现的异常

[电子书]Mastering Spark for Data Science PDF




Apache Spark常见的三大误解	Apache Hivemall:可运行在Hive, Spark 和 Pig 上的可扩展机器学习库	Spark Structured Streaming入门编程指南	Spark 2.1.0与CarbonData 1.0.0集群模式部署及使用入门指南
---------------------	--	----------------------------------	---



Apache Spark常见的三大误解	Apache Hivemall:可运行在Hive, Spark 和 Pig 上	Spark Structured Streaming入门编程指南	Spark 2.1.0与CarbonData 1.0.0集群模
---------------------	---	----------------------------------	---------------------------------

下面文章您可能感兴趣

Spark配置属性详解(1)	Ubuntu在命令行里面更新系统
Guava学习之TreeMultimap	Apache Beam成为Apache顶级项目
Hive几种参数配置方法	Spark Task序列化代码分析
Web数据挖掘	Linux库memcpy函数实现
OpenCloud 2015大会PPT资料免费下载[Spark篇]	Kafka集群Leader均衡(Balancing leadership)
Java 8的lambda使得编写Spark应用更简单	利用VPN申请Akismet KEY
通过分区(Partitioning)提高Spark的运行性能	ElasticSearch系列文章：安装
Kafka管理工具介绍	AdminLTE：基于Bootstrap3的免费高级管理控制面板主题
Ubuntu安装依赖文件	Apache YARN各组件功能概述
Apache Hadoop 2.3.0三大重要的提升	Flink Forward 201704所有PPT资料下载

 发表我的评论

写点什么吧...





表情

本博客评论系统带有自动识别垃圾评论功能，请写一些有意义的评论，谢谢！



提交评论

版权所有，保留一切权利 · 基于WordPress构建 © 2013-2015 · 广告合作 · 网站地图 · 所有文章 本主题基于欲思博客主题修改 京ICP备14057018号



