数据偏斜研究文献综述

摘要：
随着大数据时代的到来，使用 MapReduce、Spark 等分布式编程模型处理数据取得巨大成功。但是在分布式系统中，对于大数据处理会存在数据偏斜，这将会带来低资源利用率、高整体执行时间、内存溢出等问题，成为影响性能提升的一大关键因素，在本文中，我们首先对产生数据偏斜的原因和影响因素进行分析，接着研究用于缓解数据倾斜和分区倾斜的各种方法和技术，最后针对方法的优缺点进行总结分析。

关键词：
数据偏斜，MapReduce,分区

一：研究背景

近年来，随着互联网的快速发展，数据量呈指数级增长，数据并行编程模型已被广泛用于处理兆兆字节和千兆字节密集型分布式数据，例如 MapReduce，MPI 和 OpenMP 等。但是，它仍然有改进的空间，例如中间数据容错，数据偏斜和局部数据。在 MapReduce 中，当出现数据偏斜时，部分小量任务耗时远高于其他任务，从而使整体耗时过大，甚至导致任务失败。这种由于不同的执行时间导致低资源利用率和高整体执行时间，极大影响了系统的性能（新的 mapreduce 任务只有当前面所有的 reducer 完成后才能开始，即 MapReduce 中的作业完成时间取决于作业中运行最慢的任务）。接下来本文将对数据偏斜产生的原因和一些缓解数据倾斜和分区倾斜的方法与技术进行阐述分析。

二：数据偏斜产生的原因

数据偏斜，指的是并行处理的过程中，某些分区或节点处理的数据，显著高于其他分区或节点，导致这部分数据处理任务时间比其他任务大很多，从而成为整个作业的运行瓶颈，一般发生在 map 和 reduce 阶段。

产生数据倾斜的原因大致可分为以下三种：

读入数据源时便是倾斜的：读入数据是计算的开始，对于一些复杂数据分布不均衡，在读入阶段就可能出现个别 partition 执行时间过长。例如读取 id 分布跨度较大的 mysql 数据、partition 分布不均的 kafka 数据，或者不可分割的压缩文件等等。

过滤导致倾斜：有些场景下，数据原本是均衡的，但是由于进行了一系列的数据剔除操作，可能在过滤掉大量数据后，造成数据倾斜。例如大部分节点被过滤掉很多数据，余少量数据，但个别节点被过滤少量数据，保留着大部分数据，随着计算逐渐积累，将会引发倾斜问题。

shuffle 产生倾斜：在进行 shuffle 时，必须将各个节点相同的 key 拉取对应节点的 task 来进行处理，比如按照 key 进行聚合或者 join 操作。此时如果某个 key 对应的数据量特别大的话，就会发生数据偏斜。

三：缓解数据倾斜和分区倾斜的方法：

3.1 CORP：a communication-oriented reduce placement method

Zhuo Tang 等人设计了 CORP 的精简放置算法。其基本思想是将相关的 map 和 reduce 任务放置在群集或机架的临近节点上以实现数据本地性。由于在 map 任务处理输入数据之前无法计算 key 的数量，可通过对 SplitSampler 方法进行重载，然后采用 reservoir sampling 算法对输入数据进行采样，从而使 key/value 的分布更接近原始数据的整体情况。基于每个分区中中间结果的分布矩阵，通过计算跨节点通信之间的距离和成本矩阵，可以将相关的 map 和 reduce 任务调度到相对较近的物理节点进行数据本地化处理。

分布矩阵 C 用来定义中间结果在每个分区中的分布，$C = C_{i,j}^{\sigma,l}$表示分配给第 j 个 reduce 任务的来自第 l 个节点上的 i 个 map 任务的键值对的数量；D 是一个 n×n 矩阵用来定义物理节点间距离；成本矩阵用 T 表示，影响 T 的关键因素是数据的传输量和 map 节点到 reduce 节

点之间的距离，T 中元素 $t_{1,j}$可以表示为：

$$t_{l,j} = DV_l \times RV_j^T, \forall l \in [0, n-1], \forall j \in [0, p-1]$$

$$= \sum_{d=0}^{n-1} dis_{l,d} \times r_{d,j}$$ ，其中 $$\sum_{l} r_{l,j} = 1, \forall l \in [0, n-1]$$ 是一个布尔变量，确保每个 reduce 任务能放在一个节点上，向量 DV_l来计算节点 N_l与其他节点之间的距离，向量 RV_j表示 reduce j 在矩阵 R 中某个节点上的位置。

接下来整体的最小成本 MC 可以通过将分布矩阵 C 乘以一个可区分的中心节点的成本矩阵 T

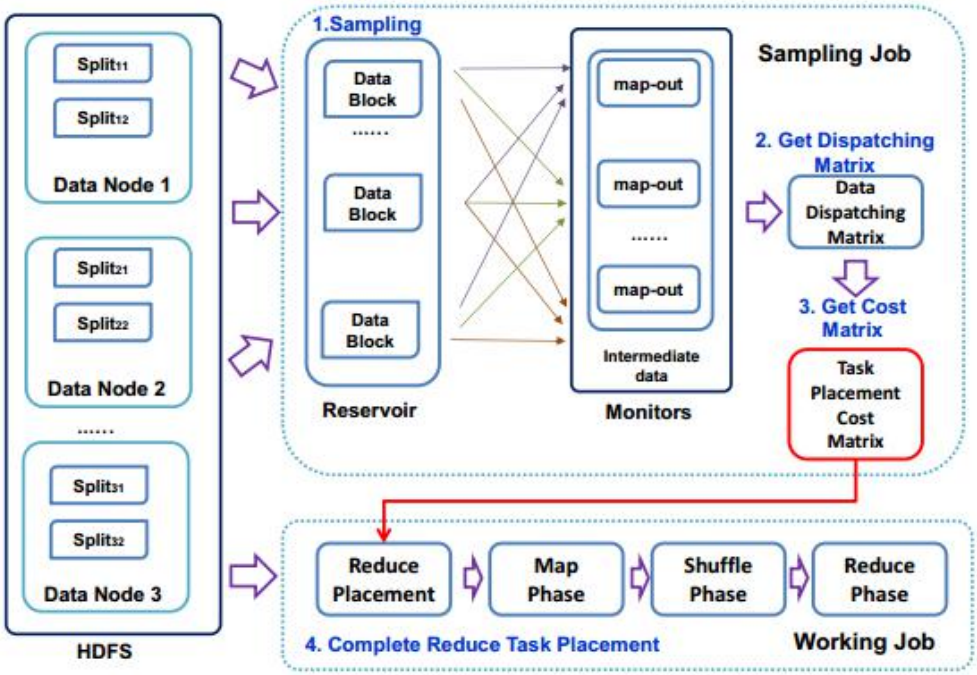
$$MC(C^{\sigma,l}, T) = \min_{K} \left(\sum_{i=0}^{m-1} \left[\sum_{j=0}^{p-1} c_{i,j}^{\sigma,l} \right] \times t_{l,j}^{k} \right)$$

来计算，表示为 ，其中$t_{i,j}^{k}$为节点 l 上 reduce 任务 j 的第 k 个位置选择。

为了实现最优的 reduce 任务布置方案，使不同物理节点之间的网络通信开销最小化，则令：

$$Minimize \sum_{m,n} MC(C^{\sigma,l}, T)$$ ，因此通过上述表达式，reduce 任务的放置可以被特化为一个问题来获得矩阵 T 的分配。

系统整体过程如下：



它由两个独立的作业组成。首先，对原始输入数据进行采样，以按节点估计每个 reducer 的键/值元组的来源，此阶段的输出是一个矩阵，用于记录当前输入数据的大小和键/值分布，可以将其传输到成本矩阵以进行 reduce 任务放置。在运行工作之前，应根据此矩阵完成 reduce 任务的放置。这样，由 reduce 任务处理的大多数数据都将尽可能地本地化，从而节

省了流量成本并提高了 reduce 任务的性能。

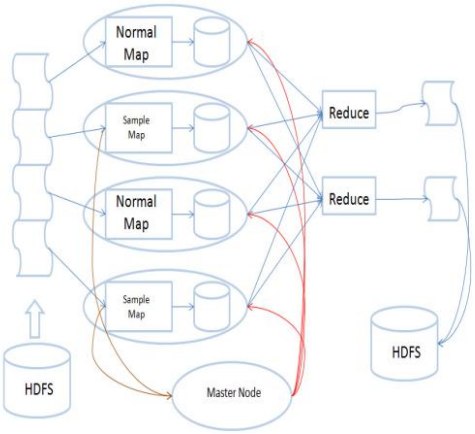
3.2 MAP：Maximum Cost Performance

Qi Chen 等人提出了一种新的推测执行策略——最大成本性能来提高 Mapreduce 的性能。对于执行时间过长的节点（称为掉队节点），一种常见的方法就是推测性执行，通过在备选节点上备份该节点执行的任务。各种推测性执行策略虽然可以提高异构和同构环境中的性能，但是有一些缺点会降低性能。当现有的策略不能很好地工作时，可引入了新的策略 MCP。在这种策略中，成本是指任务占用的计算资源，性能是指作业执行时间的减少和集群吞吐量的提高。 MCP 提供了三种方法来识别掉队节点：

1. 利用一个阶段内的进度率和流程带宽来决定慢速任务。
2. 使用 EWMA(指数加权移动平均)来预测进程速度并计算任务的剩余时间。
3. 考虑到集群上的负载，它使用成本效益模型来决定哪个任务在另一个节点上进行备份。要为备份任务选择适当的工作节点，需要同时考虑数据倾斜和数据位置。

3.3 LIBRA:平衡 reduce 任务负载的采样和划分算法

Qi Chen 等人提出了一种 LIBRA:平衡 reduce 任务负载的采样和划分算法。从 map 任务中选择一小部分作为样本任务，当系统中有空闲槽时，首先发出这些样本任务。只有当所有样本任务都完成时，才会发出普通的 map 任务。这些样本任务在普通 map 任务处理中间数据的期间收集统计信息，并在它们完成后将收集到的关于数据的分布发送给 master 节点。master 节点收集数据分布的所有样本信息，根据数据分布的预估，然后做出分区决策，然后通知从节点。当从节点接收到分区决策时，需要根据决策对数据进行分区，并且已经发出的普通的 map 任务也对中间数据进行分区。无需 reduce 任务来等待所有已发布的 map 任务的完成；一旦有关分区的决策准备就绪，便可以发布它们。



上图为 LIBRA 的系统结构。LIBRA 的抽样方法对整个原始数据集的分布有很好的估计。LIBRA 通过在 reduce 任务中更均匀地划分中间数据，显著降低了工作执行时间的不一致性或不可预测性。它允许 reduce 任务在选择的样本 map 任务(只有一小部分首先发出的 map 任务)完成后立即开始复制。当应用程序语义允许时，它支持大键的分割和输出数据的总顺序。在适当地平衡 reduce 任务之间的负载时，它还考虑了计算资源的异构性

3.4 AEGEUS： an on-line streaming based skew mitigation approach

Kumaresan V 等人研究了 reduce 阶段的分区偏斜问题，提出了 Aegeus，一种基于在线流的倾斜缓解方法，它可以动态地预测基于 reducer 负载或分区的容器规格。 Aegeus 可预测每个 map 任务的分区大小，甚至在 map 阶段完成之前就根据其需求创建资源规范。从而，该系统就可以根据工作负载创建容器，从而提高总体作业完成时间和系统性能。

Aegeus，这是一种用于分区大小预测的主动学习算法,Aegeus 包括 artitionTracker ，Partition Size Predictor 和 Resource Broker 。执行流程如下：

1. PartitionTracker 接收所有 map 任务的分区（中间输出）的大小。PartitionTracker 位于工作节点，在该组件中，抽样是基于拒绝抽样技术进行的。
2.修改后的心跳消息将传达给 Partition Size Predictor 。
3.对于大小预测，使用回归。预测的分区大小将发送到 Resource Broker 。Resource Broker 创建资源需求。
4.将资源请求从 Resource Broker 发送到 ResourceManager。
5.之后，ApplicationsManager (AsM)将资源配置任务发送到调度程序。
6.调度器将提供的资源列表作为响应从 RM 发送到 ResourceBroker。
7.Application Master 中 Resource Broker 的可以向其 NodeManager. 发送相应的容器创建请求。

四、结论：

在 MapReduce 系统中，如何克服数据倾斜，提高系统的整体性能，不同的作者有不同的解决方案，我将分析如下：

CORP 算法不仅可以有效地提高 reduce 任务的平衡性，而且可以减少内部数据通信的作业执行时间。与其他一些 reduce 调度算法相比，核心交换机上整个系统的平均数据传输量已大大减少。

MCP 显示了良好的结果，与 Hadoop 版本 0.21 相比，它可以以高达 39%的速度运行作业，并将集群吞吐量提高了 44%。它成功地通过检测掉队节点来最小化偏斜，并引入了集群吞吐量的增加。但是 MCP 只考虑在阶段结束时的投机性执行。

LIBRA 与以前的工作不同，不需要对输入数据进行任何预运行采样，也不需要防止 map 和 reduce 阶段之间的重叠。它采用了一种创新的采样方法，通过在普通处理过程中只对中间数据的小部分进行采样，就可以实现对中间数据分布的高精度近似。LIBRA 适用于广泛的应用程序，支持大键分裂，对用户是透明的，LIBRA 的开销可以忽略不计。

Aegeus 是处理 MapReduce 作业中的一种基于在线流的倾斜缓解方法，可用于动态环境的分区大小预测，它不需要很长的等待时间和额外的操作来解决倾斜问题。通过朴素方法，实验结果表明通过最大化应用程序和系统的整体性能，Aegeus 的性能比 naive Hadoop 高出 42%。但是它不支持分裂集群和异构环境，因此如果集群的大小大于节点的资源，那么该算法的性能就会下降。并且当数据量较低或数据偏度较小时，该算法可能无法很好地工作。

五：参考文献

1. Chen Q , Yao J , Xiao Z . LIBRA: Lightweight Data Skew Mitigation in MapReduce[J]. IEEE Transactions on Parallel and Distributed Systems, 2015, 26(9):2520-2533.
2. Tang Z , Ma W , Li K , et al. A Data Skew Oriented Reduce Placement Algorithm Based on Sampling[J]. IEEE Transactions on Cloud Computing, 2016, PP(99):1-1.
3. Chen Q , Liu C , Xiao Z . Improving MapReduce Performance Using Smart Speculative Execution Strategy[J]. IEEE Transactions on Computers, 2014, 63(4):954-967.
4. Le Y , Liu J , Ergun F , et al. Online load balancing for MapReduce with skewed data input[C]// Infocom, IEEE. IEEE, 2014.
5. Liu Z , Zhang Q , Ahmed R , et al. Dynamic Resource Allocation for MapReduce with Partitioning Skew[J]. IEEE Transactions on Computers, 2016, 65(11):1-1.
6. Kumaresan V , Baskaran R . AEGEUS: An online partition skew mitigation algorithm for mapreduce[C]// the International Conference. ACM, 2016.
7. Irandoost M A , Rahmani A M , Setayeshi S . MapReduce Data Skewness Handling: A Systematic Literature Review[J]. International Journal of Parallel Programming, 2019.