(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 108540407 A
(43)申请公布日 2018.09.14

(21)申请号 201810172352.3

(22)申请日 2018.03.01

(71)申请人 山东大学
地址 250061 山东省济南市历下区经十路17923号

(72)发明人 史玉良 王新军 陈志勇 胡静 臧淑娟

(74)专利代理机构 济南圣达知识产权代理有限公司 37221
代理人 杨哲

(51)Int.Cl.
H04L 12/927(2013.01)
H04L 12/823(2013.01)
G06F 9/50(2006.01)
H04L 12/803(2013.01)

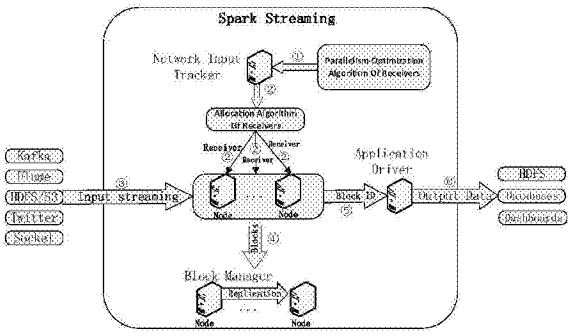
权利要求书2页 说明书12页 附图1页

(54)发明名称

一种大数据平台中Spark Streaming接收器动态配置方法及装置

(57)摘要

本发明公开了一种大数据平台中Spark Streaming接收器动态配置方法及装置，该方法包括：根据系统吞吐量和数据处理时延，确定基于系统吞吐量和数据处理时延平衡的非线性优化目标函数；求解非线性优化目标函数得到接收器个数最优的近似解作为接收器个数，并将接收器个数发送至网络接收器；网络接收器根据接收的接收器个数和集群数据对接收器进行分配，完成接收器并行度动态配置。



CN 108540407 A

1.一种大数据平台中Spark Streaming接收器动态配置方法，其特征在于，该方法包括：

根据系统吞吐量和数据处理时延，确定基于系统吞吐量和数据处理时延平衡的非线性优化目标函数；

求解非线性优化目标函数得到接收器个数最优的近似解作为接收器个数，并将接收器个数发送至网络接收器；

网络接收器根据接收的接收器个数和集群数据对接收器进行分配，完成接收器并行度动态配置。

2.如权利要求1所述的方法，其特征在于，在本方法中，采用NP难问题确定基于系统吞吐量和数据处理时延平衡的非线性优化目标函数，且非线性优化目标函数满足系统吞吐量大的同时数据处理时延小。

3.如权利要求1所述的方法，其特征在于，所述系统数据处理时延为接收器端处理时延、CPU处理时延和传输时延之和；

所述系统吞吐量根据接收器从系统外部抽取的数据量与系统数据处理时延确定；所述接收器从系统外部抽取的数据量根据接收器个数对所有接收器的单位时间内每个接收器的抽取数量与批次间时间间隔乘积求和。

4.如权利要求3所述的方法，其特征在于，所述接收器端处理时延为接收器的总处理速度与外部数据到达接收器速度的函数；所述接收器的总处理速度服从每个接收器的处理速度的指数分布，所述外部数据到达接收器速度服从一定参数的泊松分布。

5.如权利要求3所述的方法，其特征在于，所述CPU处理时延根据计算任务的批次处理数量与批次间时间间隔呈线性关系或非线性关系；

若计算任务为单批次处理，则CPU处理时延与批次间时间间隔呈线性关系；否则，CPU处理时延与批次间时间间隔呈非线性关系。

6.如权利要求1所述的方法，其特征在于，所述求解非线性优化目标函数得到接收器个数最优的近似解的具体步骤包括：

设置第一初始解和第二初始解，设定第一初始解为初始化的接收器个数最优解，根据此执行一个时间间隔的流处理任务，并记录此时耗时；

根据第一初始解和第二初始解设置初始温度，初始化外迭代次数，判断设定第一初始解或第二初始解的系统数据处理时延是否大于批次间时间间隔，若满足，则退出算法；否则继续执行；

进行外迭代计算，当前系统吞吐量最优值为计算系统吞吐量与目前最优系统吞吐量的最大值，当前系统数据处理时延最优值为计算系统数据处理时延与目前最优系统数据处理时延的最小值；若接收器数量改变导致系统能量改变的增益不小于零，记录此时参数为最优解的接收器个数、系统吞吐量和系统数据处理时延，否则根据计算接收概率，并记录接收概率大于(0,1)随机函数时的参数为最优解的接收器个数、系统吞吐量和系统数据处理时延；

对温度进行内迭代计算，若不小于设定的最低温度，则采用快速退火算法执行迭代计算，否则退出算法；若内迭代执行次数不大于设定内迭代次数，执行内迭代计算，否则退出内迭代，产生新的接收器个数，执行外迭代，直至当前接收器个数的系统数据处理时延大于

批次间时间间隔，则退出算法。

7.如权利要求6所述的方法，其特征在于，根据接收器抽取外部数据源的不同机制设置第一初始解和第二初始解；

若接收器抽取外部数据源无缓存机制，则第一初始解设置为物理节点个数的一半，第二初始解设置为物理节点个数；

若接收器抽取外部数据源有缓存机制，则第一初始解设置为最大连接数的一半，第二初始解设置为最大连接数。

8.如权利要求1所述的方法，其特征在于，该方法还包括：

所述网络接收器根据接收的接收器个数和集群数据对接收器进行分配，将接收器分配到各个节点；所述集群数据包括CPU核数和内存大小；

各个节点的接收器从外部数据源接收传输到系统的数据，将数据根据批次间隔进行分块得到块id，并将块id传给块管理器记录；

应用驱动获取块id，将数据转换成rdd提交spark进行处理，得到处理结果；

将处理结果保存在外部数据源中。

9.一种计算机可读存储介质，其中存储有多条指令，其特征在于，所述指令适于由终端设备设备的处理器加载并执行根据权利要求1-8中任一项所述的方法。

10.一种终端设备，包括处理器和计算机可读存储介质，处理器用于实现各指令；计算机可读存储介质用于存储多条指令，其特征在于，所述指令用于执行根据权利要求1-8中任一项所述的方法。

一种大数据平台中Spark Streaming接收器动态配置方法及装置

技术领域

[0001] 本发明属于大数据处理的技术领域，尤其是涉及一种大数据平台中Spark Streaming接收器动态配置方法及装置。

背景技术

[0002] 近些年来，“大数据”实时处理技术日益渗透到经济发展、社会进步和人类生活的各个领域，已经成为生产力中重要的活跃因素。目前传统批处理方式在计算过程中会发生大量的读写I/O，影响了流式数据的处理性能，传统的基于批次处理的分布式计算方式已经不能适应于实时处理的场景，因此流处理技术应运而生。分布式流处理系统在实际的生产和运用中会涉及到几十甚至几百个节点，由于各个节点的性能不同，节点故障和慢节点情况将变得很常见。在流处理背景下，由于需要在较短的时间内对数据进行处理，如果故障恢复和慢任务处理时间过长会导致系统的处理性能的降低甚至会影响计算的正确性。目前流处理系统如Storm、S4、Online和流式数据库等都是基于连续操作的模型，在这种模型中，长期运行带有状态的操作会接受每条记录，更新内部状态，并且发送新纪录，但是难以应付系统故障和慢任务问题。

[0003] 为了解决流处理系统的故障恢复和慢任务处理问题，有研究提出了一种叫DStream新的流处理模型。该处理模型将流式数据按照一定的粒度进行切分，并且将对流数据的运算转化为对切分后的数据进行批次运算。DStream模型提供了并行恢复模式，大大提高了系统恢复的效率，与此同时，Dstream通过推测执行机制对慢任务进行恢复处理，达到了对慢任务进行容错的目的。另外，DStream模型处理粒度较连续处理模型较大，延迟较连续处理模型较高。但是在定时数据统计、日志分析等应用场景下，系统的处理延迟在秒级还是可以接受的。目前，DStream模型在Spark软件栈的Spark Streaming中得到了实现，该模型将流式处理按照配置的接收器从数据源中抽取流式数据，并按照一定时间间隔进行划分，最后利用Spark计算引擎进行计算。Spark Streaming从数据源获取外界数据，外界数据经过接收器(Receiver)进入系统中，接收器并行度设置和分配机制会影响流处理的时延和负载的均衡。目前在基于DStream的Spark Streaming流处理系统中，用户需要在流处理任务开始执行前手动配置接收器个数即并行度，由网络接收器(Network Input Tracker)根据配置数，将每个接收器作为一个长期驻留的任务按照随机分配的方式分发到执行节点上执行。这种方式虽然简化了系统的复杂度，但是通过人工经验配置receiver个数既容易导致接收器并行度设置无法达到最优，也无法根据系统环境和负载进行动态调整。

[0004] 综上所述，现有技术中如何对基于DStream模型的流处理系统接收器进行并行度动态配置问题，尚缺乏行之有效的解决方案。

发明内容

[0005] 针对现有技术中存在的不足，解决现有技术中基于DStream模型的流处理系统接

收器并行度动态配置问题，本发明提出了一种大数据平台中Spark Streaming接收器动态配置方法及装置，基于时延和吞吐率的模拟退火算法对接收器并行度进行自动化确定，并根据系统环境和负载进行动态调整，有效均衡系统的吞吐量和系统处理能力，提高系统资源利用率。

[0006] 本发明的第一目的是提供一种大数据平台中Spark Streaming接收器动态配置方法。

[0007] 为了实现上述目的，本发明采用如下一种技术方案：

[0008] 一种大数据平台中Spark Streaming接收器动态配置方法，该方法包括：

[0009] 根据系统吞吐量和数据处理时延，确定基于系统吞吐量和数据处理时延平衡的非线性优化目标函数；

[0010] 求解非线性优化目标函数得到接收器个数最优的近似解作为接收器个数，并将接收器个数发送至网络接收器；

[0011] 网络接收器根据接收的接收器个数和集群数据对接收器进行分配，完成接收器并行度动态配置。

[0012] 作为进一步的优选方案，在本方法中，采用NP难问题确定基于系统吞吐量和数据处理时延平衡的非线性优化目标函数，且非线性优化目标函数满足系统吞吐量大的同时数据处理时延小。

[0013] 作为进一步的优选方案，所述系统数据处理时延为接收器端处理时延、CPU处理时延和传输时延之和；

[0014] 所述系统吞吐量根据接收器从系统外部抽取的数据量与系统数据处理时延确定。

[0015] 作为进一步的优选方案，所述接收器端处理时延为接收器的总处理速度与外部数据到达接收器速度的函数；所述接收器的总处理速度服从每个接收器的处理速度的指数分布，所述外部数据到达接收器速度服从一定参数的泊松分布。

[0016] 作为进一步的优选方案，所述CPU处理时延根据计算任务的批次处理数量与批次间时间间隔呈线性关系或非线性关系；

[0017] 若计算任务为单批次处理，则CPU处理时延与批次间时间间隔呈线性关系；否则，CPU处理时延与批次间时间间隔呈非线性关系。

[0018] 作为进一步的优选方案，所述接收器从系统外部抽取的数据量根据接收器个数对所有接收器的单位时间内每个接收器的抽取数量与批次间时间间隔乘积求和。

[0019] 作为进一步的优选方案，所述求解非线性优化目标函数得到接收器个数最优的近似解的具体步骤包括：

[0020] 设置第一初始解和第二初始解，设定第一初始解为初始化的接收器个数最优解，根据此执行一个时间间隔的流处理任务，并记录此时耗时；

[0021] 根据第一初始解和第二初始解设置初始温度，初始化外迭代次数，判断设定第一初始解或第二初始解的系统数据处理时延是否大于批次间时间间隔，若满足，则退出算法；否则继续执行；

[0022] 进行外迭代计算，当前系统吞吐量最优值为计算系统吞吐量与目前最优系统吞吐量的最大值，当前系统数据处理时延最优值为计算系统数据处理时延与目前最优系统数据处理时延的最小值；若接收器数量改变导致系统能量改变的增益不小于零，记录此时参数

为最优解的接收器个数、系统吞吐量和系统数据处理时延，否则根据计算接收概率，并记录接收概率大于(0,1)随机函数时的参数为最优解的接收器个数、系统吞吐量和系统数据处理时延；

[0023] 对温度进行内迭代计算，若不小于设定的最低温度，则采用快速退火算法执行迭代计算，否则退出算法；若内迭代执行次数不大于设定内迭代次数，执行内迭代计算，否则退出内迭代，产生新的接收器个数，执行外迭代，直至当前接收器个数的系统数据处理时延大于批次间时间间隔，则退出算法。

[0024] 作为进一步的优选方案，根据接收器抽取外部数据源的不同机制设置第一初始解和第二初始解；

[0025] 若接收器抽取外部数据源无缓存机制，则第一初始解设置为物理节点个数的一半，第二初始解设置为物理节点个数；

[0026] 若接收器抽取外部数据源有缓存机制，则第一初始解设置为最大连接数的一半，第二初始解设置为最大连接数。

[0027] 作为进一步的优选方案，该方法还包括：

[0028] 所述网络接收器根据接收的接收器个数和集群数据对接收器进行分配，将接收器分配到各个节点；所述集群数据包括CPU核数和内存大小；

[0029] 各个节点的接收器从外部数据源接收传输到系统的数据，将数据根据批次间隔进行分块得到块id，并将块id传给块管理器记录；

[0030] 应用驱动获取块id，将数据转换成rdd提交spark进行处理，得到处理结果；

[0031] 将处理结果保存在外部数据源中。

[0032] 本发明的第二目的是提供一种计算机可读存储介质。

[0033] 为了实现上述目的，本发明采用如下一种技术方案：

[0034] 一种计算机可读存储介质，其中存储有多条指令，所述指令适于由终端设备设备的处理器加载并执行以下处理：

[0035] 根据系统吞吐量和数据处理时延，确定基于系统吞吐量和数据处理时延平衡的非线性优化目标函数；

[0036] 求解非线性优化目标函数得到接收器个数最优的近似解作为接收器个数，并将接收器个数发送至网络接收器；

[0037] 网络接收器根据接收的接收器个数和集群数据对接收器进行分配，完成接收器并行度动态配置。

[0038] 本发明的第三目的是提供一种终端设备。

[0039] 为了实现上述目的，本发明采用如下一种技术方案：

[0040] 一种终端设备，包括处理器和计算机可读存储介质，处理器用于实现各指令；计算机可读存储介质用于存储多条指令，所述指令适于由处理器加载并执行以下处理：

[0041] 根据系统吞吐量和数据处理时延，确定基于系统吞吐量和数据处理时延平衡的非线性优化目标函数；

[0042] 求解非线性优化目标函数得到接收器个数最优的近似解作为接收器个数，并将接收器个数发送至网络接收器；

[0043] 网络接收器根据接收的接收器个数和集群数据对接收器进行分配，完成接收器并

行度动态配置。

[0044] 本发明的有益效果：

[0045] 1、本发明所述的一种大数据平台中Spark Streaming接收器动态配置方法及装置，对spark streaming的执行过程做出改进，解决了当前基于DStream的Spark Streaming框架中Receiver接收器并行度是依据人工经验进行配置，从而导致数量无法达到最优，而且不能根据执行器(Executor)现有计算能力和数据吞吐量情况进行动态调整的缺陷。

[0046] 2、本发明所述的一种大数据平台中Spark Streaming接收器动态配置方法及装置，通过分析处理时延、吞吐量等因素对Receiver接收器并行度的影响，提出receiver接收器动态配置策略的优化目标函数。

[0047] 3、本发明所述的一种大数据平台中Spark Streaming接收器动态配置方法及装置，针对贪心等算法求解非线性目标优化问题时容易引起局部最优和解的停滞，故采取基于时延和吞吐量的模拟退火算法(DTSA)，使得均衡系统吞吐量和系统时延之间的关系，求得最优的接收器个数，达到提高资源利用率的目的。

附图说明

[0048] 构成本申请的一部分的说明书附图用来提供对本申请的进一步理解，本申请的示意性实施例及其说明用于解释本申请，并不构成对本申请的不当限定。

[0049] 图1为本发明的改进的Spark Streaming执行框架；

[0050] 图2是本发明的接收器并行度动态配置方法流程图。

具体实施方式：

[0051] 下面将结合本发明实施例中的附图，对本发明实施例中的技术方案进行清楚、完整地描述，显然，所描述的实施例仅仅是本发明一部分实施例，而不是全部的实施例。基于本发明中的实施例，本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其他实施例，都属于本发明保护的范围。

[0052] 应该指出，以下详细说明都是例示性的，旨在对本申请提供进一步的说明。除非另有指明，本实施例使用的所有技术和科学术语具有与本申请所属技术领域的普通技术人员通常理解的相同含义。

[0053] 需要注意的是，这里所使用的术语仅是为了描述具体实施方式，而非意图限制根据本申请的示例性实施方式。如在这里所使用的，除非上下文另外明确指出，否则单数形式也意图包括复数形式，此外，还应当理解的是，当在本说明书中使用术语“包含”和/或“包括”时，其指明存在特征、步骤、操作、器件、组件和/或它们的组合。

[0054] 需要注意的是，附图中的流程图和框图示出了根据本公开的各种实施例的方法和系统的可能实现的体系架构、功能和操作。应当注意，流程图或框图中的每个方框可以代表一个模块、程序段、或代码的一部分，所述模块、程序段、或代码的一部分可以包括一个或多个用于实现各个实施例中所规定的逻辑功能的可执行指令。也应当注意，在有些作为备选的实现中，方框中所标注的功能也可以按照不同于附图中所标注的顺序发生。例如，两个接连地表示的方框实际上可以基本并行地执行，或者它们有时也可以按照相反的顺序执行，这取决于所涉及的功能。同样应当注意的是，流程图和/或框图中的每个方框、以及流程图

和/或框图中的方框的组合，可以使用执行规定的功能或操作的专用的基于硬件的系统来实现，或者可以使用专用硬件与计算机指令的组合来实现。

[0055] 在不冲突的情况下，本申请中的实施例及实施例中的特征可以相互组合下面结合附图与实施例对本发明作进一步说明。

[0056] 实施例1：

[0057] 本实施例1的目的是提供一种大数据平台中Spark Streaming接收器动态配置方法。

[0058] 为了实现上述目的，本发明采用如下一种技术方案：

[0059] 如图1-2所示，一种大数据平台中Spark Streaming接收器动态配置方法，具体步骤包括：

[0060] A.确定Spark应用程序，运行时间和输入数据集；

[0061] B.改进spark streaming的执行框架，进而提出receiver接收器动态配置策略；Receiver接收器并行度是依据人工经验进行配置，这样配置既容易导致数量无法达到最优，针对spark steaming存在的问题，对spark streaming的执行框架进行改进，提出receiver接收器动态配置策略。

[0062] 所述步骤B中改进spark streaming的执行框架包括4个步骤：

[0063] B1.将原来人工经验值设置receiver接收器个数改为通过Parallelism Optimization Algorithm Of Receiver接收器s生成最优receiver接收器个数，将receiver接收器个数通知Network Input Tracker；

[0064] B2.然后将随机分配方式改为由Allocation Algorithm Of Receivers根据receiver接收器个数及集群情况对receiver接收器进行分配，将receiver接收器分配到各个节点上；

[0065] B3.各个节点上的Receiver接收器从Kafka、Socket等外部数据源接收传输到系统的数据，负责将数据根据批次间隔分块，并将块id传给block manager记录；

[0066] B4.Application Driver获取block id将数据转换成rdd提交spark进行处理；

[0067] B5.最后将处理结果保存到HDFS、DataBase等外部数据源中；

[0068] C.定义Receiver接收器并行度问题；

[0069] 一般来讲，分配给系统receiver接收器资源越多，可以从外部数据源同时采集的数据就越多，那么，随着系统receiver接收器并行配置的增加，系统的吞吐量增加。然而分配较多的receiver接收器，导致执行器用于计算的资源减少，并且所分析数据增加，执行器(Executor)得不到有效处理，却导致批次处理延时加大。当数据积累过大时会使得系统不稳定。因此，在增加receiver接收器个数时，要使数据处理时延和系统的吞吐量达到平衡。

[0070] 在本实施例中，所述步骤C中设置receiver接收器并行度的步骤如下：

[0071] C1.假设数据在receiver接收器端的处理延时$Cost_{rec}(n)$，CPU处理时延$Cost_{cpu}(n)$和传输时延$Cost_{tra}(n)$。得到该情况下任务的总时延即端对端时延(end-to-end latency)的表达式：

[0072] $Cost(n)=Cost_{rec}(n)+Cost_{tra}(n)+Cost_{cpu}(n)$

[0073] 其中，$Cost_{cpu}(n)$通过负载函数$f(x)$计算得到。

[0074] C2.Spark Streaming的批次处理时延和时间间隔T的关系分成两种，第一种当计

算任务只涉及到单批次处理时，计算时间和时间间隔T是线性关系，第二种当计算任务设计到两个批次任务时，计算时间和时间间隔T是非线性关系。CPU处理时延$Cost_{cpu}(n)$：

[0075] $Cost_{cpu}(n)=f(T)$

[0076] C3.假设共有n个Receiver接收器并行配置，数据接收阶段可以看成一个M/M/C排队系统，假设每个接收器(Receiver接收器)的处理性能一样，每个receiver接收器的处理速度为s_r，则总的处理速度服从s_r的指数分布，外部数据到达Receiver接收器速度服从参数λ_{out}的泊松分布，数据在Receiver接收器端的处理延时$Cost_{rec}(n)$为：

[0077] $$Cost_{rec}(n)=\frac{s_r\left(\frac{\lambda_{out}}{s_r}\right)^n P_0}{(n-1)!\left(ns_r-\lambda_{out}\right)^2}+\frac{1}{s_r}$$

[0078] 其中，$Cost_{rec}(n)$为接收器端处理时延，s_r为每个接收器的处理速度，总的处理速度服从s_r的指数分布，外部数据到达接收器速度服从参数λ_{out}的泊松分布，n为接收器个数，P_0为泊松分布的概率。观察外部数据到达接收器平均发生ρ次的条件下，实际发生k次。

[0079] $$P_0=\left[\sum_{k=0}^{n-1}\frac{\rho^k}{\kappa!}+\frac{\rho^n}{n!\left(1-\frac{\rho}{n}\right)}\right]^{-1}$$

[0080] $$\rho=\frac{\lambda_{out}}{s_r}$$

[0081] C4.吞吐量tps_n是由接收器从系统外部抽取的数据量D_T和系统时延$cost_n$决定的，因此计算公式为：

[0082] $$tps_n=\frac{D_T}{cost_n}=\frac{\sum_{j=1}^{n}c_j*T}{cost_n}$$

[0083] 其中，tps_n为系统吞吐量，D_T接收器从系统外部抽取的数据量，$cost_n$为系统数据处理时延，c^j为单位时间内每个接收器的抽取数量，j为接收器变量。

[0084] C5.假设系统批次间隔是定值T，分配给系统的接收器个数为n，单位时间内每个receiver接收器的抽取数量为c^j。当receiver接收器接收速率一定时，D_T随着n的变化而变化，当n很小时，由于大部分资源用于计算任务，导致$Cost_{cpu}(n)$较小。当n增加到一定量时，计算能力达到瓶颈。

[0085] 当系统的吞吐量和数据处理延时达到一个平衡点时，此时Receiver接收器个数为最优。定义一个优化目标即满足系统吞吐量较大，同时系统时延较少的函数。针对上述目标和条件，用表达式表述如下：

[0086] $$\min\left(p\left|\frac{tps_n-tps^*}{tps^*}\right|+(1-p)\left|\frac{const_n-cost^*}{cost^*}\right|\right)\ st.\begin{cases}cost_n\le T\\ n\le\begin{cases}\max-connections\\ \frac{N}{2}\end{cases}\\ cost_{cpu}<\max\left(cost_{cpu}\right)\end{cases}$$

[0087] 其中，p和(1-p)为吞吐量和数据处理时间的比例分配，tps_n为系统吞吐量，tps^*为当前最优系统吞吐量，$cost_n$为系统数据处理，$cost^*$为当前最优数据处理时间，T为批次间时间间隔，n为接收器个数，max为一个执行器最多可有的接收器数量，connections为已分配的接收器数量，N为集群中所有内核数量。

[0088] 在本实施例中，目标函数是一个非线性目标优化Receiver接收器问题，属于NP难问题。

[0089] D.根据系统时延和吞吐量的关系，设计了基于时延和吞吐量的模拟退火算法(DTSA)求得receiver接收器个数最优的近似解。

[0090] 所述步骤D中基于时延和吞吐量的模拟退火算法(DTSA)近似求解receiver接收器最优个数，步骤如下：

[0091] D1.首先产生初始解n_0、n_1，系统分配n_0个receiver接收器，并按此参数分别执行一个时间间隔T的流处理任务，并且记录此时耗时cost，初始化receiver接收器最优解n*并将将n_0的值赋给n*，其中按照Receiver接收器抽取外部资源的不同设置n_0和n_1值，当外部数据源无缓存机制时，n_0设置为物理节点个数的一半，n_1设置为物理节点个数，当外部数据源有缓存机制时，n_0设置为最大连接数的一半，n_1设置为最大连接数；

[0092] D2.设置初始温度T(0)，设置T(0)为$T(0)=\frac{(n_0-n_1)}{c}, c\in(0,1)$，并记录此时$n_0$的三元组<个数，吞吐量，耗时>，并初始化迭代次数k＝0。用三元组OPT(n*)＝<n*,tps*,cost*>代表目前最优的Receiver接收器个数，吞吐量和耗时组合。用OPT(n_k)＝<n_k,tps_{nk},$cost_{nk}$>代表要比较的receiver接收器个数。用tps_δ和$cost_\delta$表示当前的最优值：

[0093] $tps_\delta = \max(tps^*, tps_{nk})$

[0094] $cost_\delta = \min(cost^*, cost_{nk})$

[0095] D3.判断此时批次处理时间Cost(n_0)>T或者Cost(n_1)>T，如果满足则退出算法，否则继续执行；

[0096] D4.计算T(k)，若T(k)>＝T(m)，T(m)是设定的最低温度，m则为到达最低温度的时间，则执行下面迭代，否则退出算法。迭代分为两层，初始内迭代执行次数i，令i＝0；其中降温公式采用快速退火算法：

[0097] $T(k)=\frac{T_0}{1+k}$

[0098] D5.若i<＝N_{in}执行内迭代，Nin为设定的内循环迭代次数，否则退出内迭代，根据产生式规则生成新的receiver接收器个数n_k，执行并记录此时的吞吐量、耗时，若此时Cost(n_k)>T，退出算法。其中，当前receiver接收器个数与下一次迭代的receiver接收器个数的关系如下：

[0099] $$\frac{n_{k+1}}{n_k}=\left(\frac{1+\frac{1}{idle^{\alpha}_{n_{k+1}}}}{1+\frac{1}{idle^{\alpha}_{n_k}}}\right)^{\beta} \Rightarrow n_{k+1}=\left(\frac{1+\frac{1}{idle^{\alpha}_{n_{k+1}}}}{1+\frac{1}{idle^{\alpha}_{n_k}}}\right)^{\beta} * n_k$$

[0100] 式中，α和β为一个调整量。k是当前迭代次数。$idle_n$代表的是当Receiver接收器个数是n时，在时间间隔T内，每个物理节点中的Executor除去接受任务其余CPU平均空载时间。

[0101] $$idle_n=\frac{\sum_{j=1}^{N-n} idle\_cpu_j}{N-n}$$

[0102] 其中$idle_cpu_j$表示第j个CPU内核在T时间间隔内的空闲时间；

[0103] D6.否则，计算增益Gain(n_k)，用来表示Receiver接收器数量改变时导致系统能量的增加或者减少，若此时增益Gain(n_k)>=0，将此时参数值记录为最优值，否则计算p(n_k)，若p(n_k)>random(0,1)，那么将此时参数值记录为最优值，即<n*,tps*,cost*>←<n_k,tps_{nk},$cost_{nk}$>；

[0104] 增益Gain(n)计算公式为：

[0105] Gain(n)=-(E(n,tps_δ,$cost_\delta$)-E(n^*,tps_δ,$cost_\delta$))

[0106] 其中，系统能量公式为：

[0107] $$E(t,tps^*,cost^*)=p\left|\frac{tps_t-tps^*}{tps^*}\right|+(1-p)\left|\frac{\cos t_t-\cos t^*}{\cos t^*}\right|$$

[0108] 接收概率P(n)采用经典的Metropolis准则为：

[0109] $$p=e^{[Gain(t)/t_k]}$$

[0110] D7.执行i=i+1,k=k+1并根据降温公式计算下一次温度，直到Cost(n_k)>T，则返回n*，退出算法。

[0111] 实施例2：

[0112] 本实施例3的目的是基于实施例1中的方法，进行实验验证。

[0113] 实验环境采用Spark1.6+hadoop2.2，编写程序为wordCount，通过Maven进行编译后部署到实验集群上。本实施例在一个真正的Spark集群上部署了11个虚拟机(VM)，每个虚拟机有8个2GHz内核，8GB RAM和500GB硬盘，使用一个虚拟机作为ResourceManager和NameNode，其余10个虚拟机作为工作者，每个工作人员配置16个虚拟内，7GB内存(后台进程需要1GB)和500GB硬盘，本实施例已经实现了独立的资源管理和调度。为了保证数据的可靠性，本实施例采用HDFS(Hadoop分布式文件系统)在Spark底部获得永久结果。HDFS块大小设置为64MB，复制级别设置为3。采用RedHat6.3服务器版本作为实验操作系统，并使用不同的数据源来评估本方法。使用WordCount来评估本文方法，使用维基百科数据作为输入数据集。本文实验中批次间隔为400ms。

[0114] 步骤包括如下：

[0115] A.使用WordCount应用程序来评估本文方法，使用维基百科数据作为输入数据集；

[0116] B.计算同一个数据源在不同工作节点个数情况下最优的receiver接收器个数以及比较本文算法的优越性。例如当工作节点个数为6时，分别测试receiver接收器个数n从1到N/2的系统吞吐量和时延，求解每个个数下的系统能量E，将E最小的n的取值为此节点数量下的最优receiver接收器个数。实验中，本实施例采用p=0.5，tps*是吞吐量最大值，cost*是延迟最小值，

[0117] 首先第一组使用Kafka作为数据源，得到的最优receiver接收器个数如表1所示：

[0118] 表1工作节点数量变化时算法准确度的比较(Kafka作为数据源)

[0119]

节点数量	实际最优值	本实施例算法 DTSA	经验值	爬山算法
2	1	1	1	1
3	2	2	2	2
4	4	4	2	4
5	5	5	5	4
6	6	6	5	6
7	7	6	7	6
8	7	7	6	6
9	8	7	10	7
10	10	10	12	10

[0120] 由表1可以看出，本实施例的准确率为88.9％，Manual的准确率为33.3％，Hill Climbing的准确率为55.6％，可以看出本实施例算法较于其他两种配置方法有较高的准确率。而且在数据源有缓存时，最佳receiver接收器个数与节点个数相差不大。

[0121] 第二组采用的是socket与数据源连接，没有缓存，直接将抓取的数据输入系统，计算该情况下实际最优receiver接收器个数，如表2所示：

[0122] 表2工作节点数量变化时算法准确度的比较(Socket作为数据源)

[0123]

节点数量	实际最优值	本实施例算法 DTSA	经验值	爬山算法
2	1	1	1	1
3	3	3	3	3
4	4	4	3	3
5	7	6	6	4
6	9	9	9	9
7	10	10	11	10
8	13	14	13	12
9	15	15	15	14
10	16	16	17	16

[0124] 由表2可以看出，本实施例算法的准确率为77.8％，经验值算法的准确率为55.6％，爬山算法的准确率为55.6％，可以看出本实施例算法较于有缓存的数据源准确性下降，但其他两种配置方法准确率还是较高。在没有缓存的情况下，最佳receiver接收器个数与节点个数没有相关性。

[0125] C.比较数据量大小对receiver接收器最优个数的影响以及本实施例算法的优越性。实验数据采用的是record size分别为100byte、500byte和1000byte，数据源为Kafka，集群节点个数为10。对于不同数据量对本实施例算法和爬山算法进行测试，实验结果表3所

示：

[0126] 表3数据量大小对receiver接收器最优个数的影响

[0127]

算法	100byte	500byte	1000byte
本实施例算法DTSA	10	10	11
爬山算法	10	13	17

[0128] 通过以上两个实验，验证了本实施例提出算法有较高的准确性，而且不受数据量变化的影响。

[0129] 实施例3：

[0130] 本实施例3的目的是提供一种计算机可读存储介质。

[0131] 为了实现上述目的，本发明采用如下一种技术方案：

[0132] 一种计算机可读存储介质，其中存储有多条指令，所述指令适于由终端设备设备的处理器加载并执行以下处理：

[0133] 根据系统吞吐量和数据处理时延，确定基于系统吞吐量和数据处理时延平衡的非线性优化目标函数；

[0134] 求解非线性优化目标函数得到接收器个数最优的近似解作为接收器个数，并将接收器个数发送至网络接收器；

[0135] 网络接收器根据接收的接收器个数和集群情况对接收器进行分配，完成接收器并行度动态配置。

[0136] 实施例4：

[0137] 本实施例4的目的是提供一种终端设备。

[0138] 为了实现上述目的，本发明采用如下一种技术方案：

[0139] 一种终端设备，包括处理器和计算机可读存储介质，处理器用于实现各指令；计算机可读存储介质用于存储多条指令，所述指令适于由处理器加载并执行以下处理：

[0140] 根据系统吞吐量和数据处理时延，确定基于系统吞吐量和数据处理时延平衡的非线性优化目标函数；

[0141] 求解非线性优化目标函数得到接收器个数最优的近似解作为接收器个数，并将接收器个数发送至网络接收器；

[0142] 网络接收器根据接收的接收器个数和集群情况对接收器进行分配，完成接收器并行度动态配置。

[0143] 这些计算机可执行指令在设备中运行时使得该设备执行根据本公开中的各个实施例所描述的方法或过程。

[0144] 在本实施例中，计算机程序产品可以包括计算机可读存储介质，其上载有用于执行本公开的各个方面的计算机可读程序指令。计算机可读存储介质可以是可以保持和存储由指令执行设备使用的指令的有形设备。计算机可读存储介质例如可以是——但不限于——电存储设备、磁存储设备、光存储设备、电磁存储设备、半导体存储设备或者上述的任意合适的组合。计算机可读存储介质的更具体的例子(非穷举的列表)包括：便携式计算机盘、硬盘、随机存取存储器(RAM)、只读存储器(ROM)、可擦式可编程只读存储器(EPROM或

闪存)、静态随机存取存储器(SRAM)、便携式压缩盘只读存储器(CD-ROM)、数字多功能盘(DVD)、记忆棒、软盘、机械编码设备、例如其上存储有指令的打孔卡或凹槽内凸起结构、以及上述的任意合适的组合。这里所使用的计算机可读存储介质不被解释为瞬时信号本身，诸如无线电波或者其他自由传播的电磁波、通过波导或其他传输媒介传播的电磁波(例如，通过光纤电缆的光脉冲)、或者通过电线传输的电信号。

[0145] 本文所描述的计算机可读程序指令可以从计算机可读存储介质下载到各个计算/处理设备，或者通过网络、例如因特网、局域网、广域网和/或无线网下载到外部计算机或外部存储设备。网络可以包括铜传输电缆、光纤传输、无线传输、路由器、防火墙、交换机、网关计算机和/或边缘服务器。每个计算/处理设备中的网络适配卡或者网络接口从网络接收计算机可读程序指令，并转发该计算机可读程序指令，以供存储在各个计算/处理设备中的计算机可读存储介质中。

[0146] 用于执行本公开内容操作的计算机程序指令可以是汇编指令、指令集架构(ISA)指令、机器指令、机器相关指令、微代码、固件指令、状态设置数据、或者以一种或多种编程语言的任意组合编写的源代码或目标代码，所述编程语言包括面向对象的编程语言—诸如C++等，以及常规的过程式编程语言—诸如“C”语言或类似的编程语言。计算机可读程序指令可以完全地在用户计算机上执行、部分地在用户计算机上执行、作为一个独立的软件包执行、部分在用户计算机上部分在远程计算机上执行、或者完全在远程计算机或服务器上执行。在涉及远程计算机的情形中，远程计算机可以通过任意种类的网络—包括局域网(LAN)或广域网(WAN)—连接到用户计算机，或者，可以连接到外部计算机(例如利用因特网服务提供商来通过因特网连接)。在一些实施例中，通过利用计算机可读程序指令的状态信息来个性化定制电子电路，例如可编程逻辑电路、现场可编程门阵列(FPGA)或可编程逻辑阵列(PLA)，该电子电路可以执行计算机可读程序指令，从而实现本公开内容的各个方面。

[0147] 应当注意，尽管在上文的详细描述中提及了设备的若干模块或子模块，但是这种划分仅仅是示例性而非强制性的。实际上，根据本公开的实施例，上文描述的两个或更多模块的特征和功能可以在一个模块中具体化。反之，上文描述的一个模块的特征和功能可以进一步划分为由多个模块来具体化。

[0148] 本发明的有益效果：

[0149] 1、本发明所述的一种大数据平台中Spark Streaming接收器动态配置方法及装置，对spark streaming的执行过程做出改进，解决了当前基于DStream的Spark Streaming框架中Receiver接收器并行度是依据人工经验进行配置，从而导致数量无法达到最优，而且不能根据执行器(Executor)现有计算能力和数据吞吐量情况进行动态调整的缺陷。

[0150] 2、本发明所述的一种大数据平台中Spark Streaming接收器动态配置方法及装置，通过分析处理时延、吞吐量等因素对Receiver接收器并行度的影响，提出receiver接收器动态配置策略的优化目标函数。

[0151] 3、本发明所述的一种大数据平台中Spark Streaming接收器动态配置方法及装置，针对贪心等算法求解非线性目标优化问题时容易引起局部最优和解的停滞，故采取基于时延和吞吐量的模拟退火算法(DTSA)，使得均衡系统吞吐量和系统时延之间的关系，求得最优的接收器个数，达到提高资源利用率的目的。

[0152] 以上所述仅为本申请的优选实施例而已，并不用于限制本申请，对于本领域的技

术人员来说，本申请可以有各种更改和变化。凡在本申请的精神和原则之内，所作的任何修改、等同替换、改进等，均应包含在本申请的保护范围之内。因此，本发明将不会被限制于本文所示的这些实施例，而是要符合与本文所公开的原理和新颖特点相一致的最宽的范围。

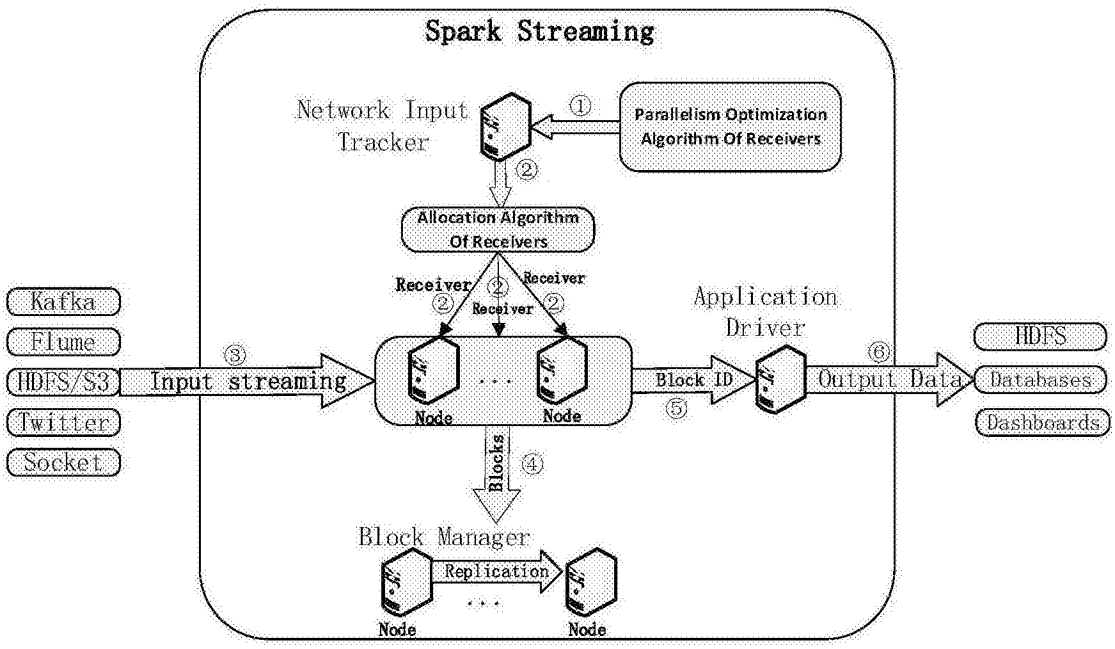

Spark Streaming
Network Input Tracker
①
Parallelism Optimization Algorithm Of Receivers
②
Allocation Algorithm Of Receivers
Receiver
Receiver
Receiver
Kafka
Flume
HDFS/S3
Twitter
Socket
③
Input streaming
Node
Node
Application Driver
Block ID
⑤
⑥
Output Data
HDFS
Databases
Dashboards
Blocks
④
Block Manager
Replication
Node
Node

图1

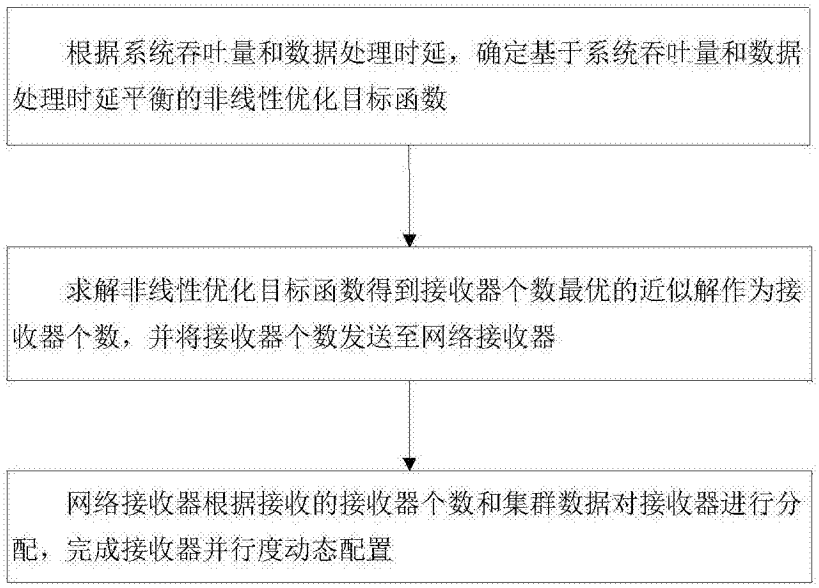

根据系统吞吐量和数据处理时延，确定基于系统吞吐量和数据处理时延平衡的非线性优化目标函数
求解非线性优化目标函数得到接收器个数最优的近似解作为接收器个数，并将接收器个数发送至网络接收器
网络接收器根据接收的接收器个数和集群数据对接收器进行分配，完成接收器并行度动态配置

图2