(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 103970520 B
(45)授权公告日 2017.06.16

(21)申请号 201310037826.0

(22)申请日 2013.01.31

(65)同一申请的已公布的文献号
申请公布号 CN 103970520 A

(43)申请公布日 2014.08.06

(73)专利权人 国际商业机器公司
地址 美国纽约

(72)发明人 史巨伟 李立 邹嘉 于琦

(74)专利代理机构 中国国际贸易促进委员会专利商标事务所 11038
代理人 李镇江

(51)Int.Cl.
G06F 9/44(2006.01)
G06F 9/50(2006.01)

(56)对比文件
US 2002/0078213 A1,2002.06.20,
CN 102467570 A,2012.05.23,
CN 102831102 A,2012.12.19,

审查员 朱来普

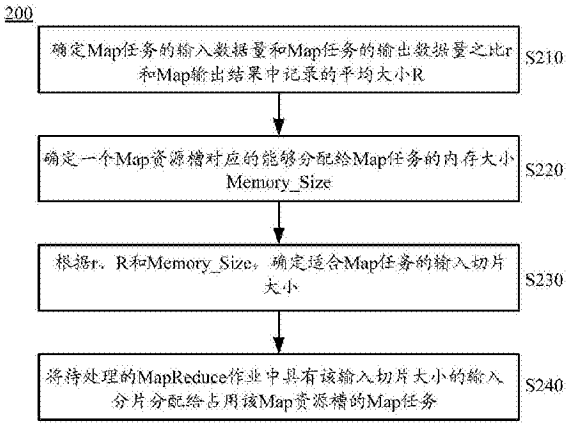
权利要求书3页 说明书14页 附图5页

(54)发明名称

MapReduce架构中的资源管理方法、装置和架构系统

(57)摘要

本发明实施例公开了一种用于MapReduce架构中的资源管理的方法、装置和MapReduce架构系统。该方法包括：确定Map任务的输入数据量和Map任务的输出数据量之比r以及Map输出结果中记录的平均大小R；确定一个Map资源槽对应的能够分配给Map任务的内存大小Memory_Size；根据所确定的r、R和Memory_Size，确定适合该Map任务的输入切片大小；以及将待处理的MapReduce作业中具有该输入切片大小的输入分片分配给占用该Map资源槽的Map任务。通过利用上述方法，可以尽量避免磁盘溢出，从而提高处理效率，避免资源浪费。



CN 103970520 B

1.一种用于MapReduce架构中的资源管理的方法,包括:

确定Map任务的输入数据量和Map任务的输出数据量之比r以及Map输出结果中记录的平均大小R;

确定一个Map资源槽对应的能够分配给Map任务的内存大小Memory_Size;

根据所确定的r、R和Memory_Size,确定适合该Map任务的输入切片大小;以及

将待处理的MapReduce作业中具有该输入切片大小的输入分片分配给占用该Map资源槽的Map任务。

2.根据权利要求1所述的方法,其中,所述确定Map任务的输入数据量和Map任务的输出数据量之比r以及Map输出结果中记录的平均大小R包括如下二者之一:

预先运行一个或多个Map任务,并根据运行后得到的结果确定r和R;

根据与历史上运行过的Map任务相关的日志信息,确定r和R。

3.根据权利要求1所述的方法,其中,所述根据所确定的r、R和Memory_Size、确定适合该Map任务的输入切片大小包括:

根据R、Map输出结果中一个记录所对应的管理开销和Memory_Size,确定Memory_Size中用于存储除所述管理开销之外的数据的内存大小Data_Buffer;以及

根据Data_Buffer和r,确定适合该Map任务的输入切片大小。

4.根据权利要求1所述的方法,其中,在MapReduce架构用于同构网络的情况下,所述确定一个Map资源槽对应的能够分配给Map任务的内存大小Memory_Size包括:

根据用于并行执行Map任务的资源槽总数和用于并行执行Map任务的总内存,确定Memory_Size。

5.根据权利要求4所述的方法,其中,还包括:

根据所述MapReduce作业的总输入数据量和所述输入切片大小,确定所述MapReduce作业的Map任务数量。

6.根据权利要求1所述的方法,其中,在MapReduce架构用于异构网络的情况下,所述确定一个Map资源槽对应的能够分配给Map任务的内存大小Memory_Size包括:

响应于一个Map资源槽空闲,确定该Map资源槽对应的内存大小,作为Memory_Size。

7.根据权利要求1所述的方法,还包括:

根据r和所述MapRuduce作业的总输入数据量,确定所述MapReduce作业的Map输出结果的总数据量;

确定一个Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小;以及

根据所述存储Reduce任务的输入数据的内存大小和所述Map输出结果的总数据量,确定所述MapReduce作业的Reduce任务数量。

8.根据权利要求7所述的方法,其中,在MapReduce架构用于同构网络的情况下,所述确定一个Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小包括:

根据用于并行执行Reduce任务的资源槽总数、用于并行执行Reduce任务的总内存以及小于1的预定系数,确定一个Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小。

9.根据权利要求7所述的方法,其中,在MapReduce架构用于异构网络的情况下,所述确

定一个Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小包括：

针对异构网络中执行所述MapReduce作业的各Reduce资源槽中的每一个，根据该Reduce资源槽对应的能够分配给Reduce任务的内存大小和小于1的预定系数，确定该Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小，

以及，所述根据所述存储Reduce任务的输入数据的内存大小和所述Map输出结果的总数据量确定所述MapReduce作业的Reduce任务数量包括：

根据所述各Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小的公约数和所述Map输出结果的总数据量，确定所述MapReduce作业的Reduce任务数量，

以及，在确定所述MapReduce作业的Reduce任务数量之后，所述方法还包括：

响应于一个Reduce资源槽空闲，根据该Reduce资源槽对应的能够分配来存储Reduce任务的内存大小和所述公约数，将具有如下个数的Reduce任务分配给占用该Reduce资源槽的Reduce任务，所述个数等于该Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小相对于所述公约数的倍数。

10.根据权利要求1所述的方法，还包括：

响应于占用Reduce资源槽的Reduce任务的读取Map输出结果的操作完成，开始还未占用Reduce资源槽的Reduce任务的读取Map输出结果的操作。

11.一种用于MapReduce架构中的资源管理的装置，包括：

第一确定部件，被配置为确定Map任务的输入数据量和Map任务的输出数据量之比r以及Map输出结果中记录的平均大小R；

第二确定部件，被配置为确定一个Map资源槽对应的能够分配给Map任务的内存大小Memory_Size；

第三确定部件，被配置为根据所确定的r、R和Memory_Size，确定适合该Map任务的输入切片大小；以及

第一分配部件，被配置为将待处理的MapReduce作业中具有该输入切片大小的输入分片分配给占用该Map资源槽的Map任务。

12.根据权利要求11所述的装置，其中，所述第一确定部件包括如下二者之一：

第一确定单元，被配置为预先运行一个或多个Map任务，并根据运行后得到的结果确定r和R；

第二确定单元，被配置为根据与历史上运行过的Map任务相关的日志信息，确定r和R。

13.根据权利要求11所述的装置，其中，所述第三确定部件包括：

第三确定单元，被配置为根据R、Map输出结果中一个记录所对应的管理开销和Memory_Size，确定Memory_Size中用于存储除所述管理开销之外的数据的内存大小Data_Buffer；以及

第四确定单元，被配置为根据Data_Buffer和r，确定适合该Map任务的输入切片大小。

14.根据权利要求11所述的装置，其中，在MapReduce架构用于同构网络的情况下，所述第二确定部件被配置为根据用于并行执行Map任务的资源槽总数和用于并行执行Map任务的总内存，确定Memory_Size。

15.根据权利要求14所述的装置，其中，还包括：

第四确定部件，被配置为根据所述MapReduce作业的总输入数据量和所述输入切片大

小，确定所述MapReduce作业的Map任务数量。

16.根据权利要求11所述的装置，其中，在MapReduce架构用于异构网络的情况下，所述第二确定部件被配置为响应于一个Map资源槽空闲，确定该Map资源槽对应的内存大小，作为Memory_Size。

17.根据权利要求11所述的装置，还包括：

第五确定部件，被配置为根据r和所述MapRuduce作业的总输入数据量，确定所述MapReduce作业的Map输出结果的总数据量；

第六确定部件，被配置为确定一个Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小；以及

第七确定部件，被配置为根据所述存储Reduce任务的输入数据的内存大小和所述Map输出结果的总数据量，确定所述MapReduce作业的Reduce任务数量。

18.根据权利要求17所述的装置，其中，在MapReduce架构用于同构网络的情况下，所述第六确定部件被配置为根据用于并行执行Reduce任务的资源槽总数、用于并行执行Reduce任务的总内存以及小于1的预定系数，确定一个Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小。

19.根据权利要求17所述的装置，其中，在MapReduce架构用于异构网络的情况下，所述第六确定部件被配置为针对异构网络中执行所述MapReduce作业的各Reduce资源槽中的每一个，根据该Reduce资源槽对应的能够分配给Reduce任务的内存大小和小于1的预定系数，确定该Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小，

以及，所述第七确定部件被配置为根据所述各Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小的公约数和所述Map输出结果的总数据量，确定所述MapReduce作业的Reduce任务数量，

以及，所述装置还包括：

第二分配部件，被配置为响应于一个Reduce资源槽空闲，根据该Reduce资源槽对应的能够分配来存储Reduce任务的内存大小和所述公约数，将具有如下个数的Reduce任务分配给占用该Reduce资源槽的Reduce任务，所述个数等于该Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小相对于所述公约数的倍数。

20.根据权利要求11所述的装置，还包括：

控制部件，被配置为响应于占用Reduce资源槽的Reduce任务的读取Map输出结果的操作完成，开始还未占用Reduce资源槽的Reduce任务的读取Map输出结果的操作。

21.一种MapReduce架构系统，包括：

根据权利要求11-20中任一项所述的装置。

MapReduce架构中的资源管理方法、装置和架构系统

技术领域

[0001] 本发明涉及MapReduce架构，并且更具体地，涉及MapReduce架构中用于资源管理的方法和装置以及具有该装置的MapReduce架构系统。

背景技术

[0002] MapReduce架构是一种编程模型架构，用于大规模数据集(例如大于1TB)的并行运算。MapReduce通过借助于主控节点的控制将数据集的大规模操作分发给网络上的计算节点来进行分布式处理，以提高对大规模数据的执行速度和效率。MapReduce将诸如对大量数据进行词频统计之类的MapReduce作业划分为多个Map(映射)任务和多个Reduce(化简)任务，Map任务的输出结果作为Reduce任务的输入数据。

[0003] 目前，MapReduce架构包括将近200个系统参数，用户通过设置这些系统参数中的一部分或全部来指定能够用于处理一个MapReduce作业的资源以及如何利用这些资源等。然而，这些系统参数的设置是由用户根据经验等人为确定的，而没有考虑到节点的处理能力和/或资源情况。这样设置出来的系统参数常常并不优化，例如，用户自己设置的系统参数可能导致节点处理效率低下等问题。

[0004] 举例来说，假设MapReduce中一个Map任务需要处理的输入切片大小为1000MB，对应的输出数据为300MB。如果该Map任务占用Map资源槽(slot)后被分配的内存为100MB，那么，由于输出数据大于内存，所以Map操作后得到的每条记录将首先作为中间结果溢出到磁盘，再由该Map任务分三次从磁盘读取中间结果，对其进行排序和合并，并将最终的Map输出结果再次溢出到磁盘，以供Reduce任务读取。

[0005] 在这种情况下，由于Map任务的输入切片大小过大，使得Map输出结果的数据量(300MB)大于能够处理该Map任务的内存大小(100MB)，造成了需要将对输入数据进行Map操作得到的数据首先溢出到磁盘中、再多次对磁盘进行反复读写才能得到最终Map输出结果的问题，严重影响处理效率。

发明内容

[0006] 本发明实施例提供了用于MapReduce架构中的资源管理的方法、装置和MapReduce架构系统，能够尽可能避免在Map任务的输入切片大小与内存不匹配的情况下所造成的反复对磁盘进行读写的问题，从而提高Map任务的处理效率。

[0007] 根据本发明的一个方面，提供了一种用于MapReduce架构中的资源管理的方法，包括：确定Map任务的输入数据量和Map任务的输出数据量之比r以及Map输出结果中记录的平均大小R；确定一个Map资源槽对应的能够分配给Map任务的内存大小Memory_Size；根据所确定的r、R和Memory_Size，确定适合该Map任务的输入切片大小；以及将待处理的MapReduce作业中具有该输入切片大小的输入分片分配给占用该Map资源槽的Map任务。

[0008] 根据本发明的另一方面，提供了一种用于MapReduce架构中的资源管理的装置，包括：第一确定部件，被配置为确定Map任务的输入数据量和Map任务的输出数据量之比r以及

Map输出结果中记录的平均大小R;第二确定部件,被配置为确定一个Map资源槽对应的能够分配给Map任务的内存大小Memoy_Size;第三确定部件,被配置为根据所确定的r、R和Memory_Size,确定适合该Map任务的输入切片大小;以及第一分配部件,被配置为将待处理的MapReduce作业中具有该输入切片大小的输入分片分配给占用该Map资源槽的Map任务。

[0009] 根据本发明的再一方面,提供了一种包含上述装置的MapReduce架构系统。

[0010] 根据本发明实施例提供的上述技术方案,通过预先估计出一个Map资源槽对应的能够分配给Map任务的内存大小,可以根据该内存大小向占用该Map资源槽的Map任务分配相匹配的输入切片大小,从而尽可能避免输入切片大小和内存大小不匹配而导致的多次磁盘溢出以及在该情况下造成的对磁盘的反复读写,因此能够提高Map任务的处理效率,避免资源浪费。

附图说明

[0011] 通过结合附图对本公开示例性实施方式进行更详细的描述,本公开的上述以及其它目的、特征和优势将变得更加明显,其中,在本公开示例性实施方式中,相同的参考标号通常代表相同部件。

[0012] 图1示出了适于用来实现本发明实施方式的示例性计算机系统/服务器12的框图。

[0013] 图2是根据本发明实施例的用于MapReduce架构中的资源管理的方法的流程图。

[0014] 图3是根据本发明实施例的确定Map任务的输入切片大小的方法的流程图。

[0015] 图4是根据本发明实施例的用于MapReduce架构中的资源管理的另一方法的流程图。

[0016] 图5是根据本发明实施例的针对Reduce任务的状态迁移的例子。

[0017] 图6是利用根据本发明实施例的方法来处理一个MapReduce作业的示例性方法的流程图。

[0018] 图7是根据本发明实施例的用于MapReduce架构中的资源管理的装置的结构框图。

[0019] 图8是根据本发明实施例的用于该MapReduce架构中的资源管理的另一装置的结构框图。

[0020] 图9是根据本发明实施例的MapReduce架构系统的结构框图。

具体实施方式

[0021] 下面将参照附图更详细地描述本公开的优选实施方式。虽然附图中显示了本公开的优选实施方式,然而应该理解,可以以各种形式实现本公开而不应被这里阐述的实施方式所限制。相反,提供这些实施方式是为了使本公开更加透彻和完整,并且能够将本公开的范围完整地传达给本领域的技术人员。

[0022] 所属技术领域的技术人员知道,本发明可以实现为系统、方法或计算机程序产品。因此,本公开可以具体实现为以下形式,即:可以是完全的硬件、也可以是完全的软件(包括固件、驻留软件、微代码等),还可以是硬件和软件结合的形式,本文一般称为“电路”、“模块”或“系统”。此外,在一些实施例中,本发明还可以实现为在一个或多个计算机可读介质中的计算机程序产品的形式,该计算机可读介质中包含计算机可读的程序代码。

[0023] 可以采用一个或多个计算机可读的介质的任意组合。计算机可读介质可以是计算

机可读信号介质或者计算机可读存储介质。计算机可读存储介质例如可以是——但不限于——电、磁、光、电磁、红外线、或半导体的系统、装置或器件，或者任意以上的组合。计算机可读存储介质的更具体的例子(非穷举的列表)包括：具有一个或多个导线的电连接、便携式计算机磁盘、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)、光纤、便携式紧凑磁盘只读存储器(CD-ROM)、光存储器件、磁存储器件、或者上述的任意合适的组合。在本文件中，计算机可读存储介质可以是任何包含或存储程序的有形介质，该程序可以被指令执行系统、装置或者器件使用或者与其结合使用。

[0024] 计算机可读的信号介质可以包括在基带中或者作为载波一部分传播的数据信号，其中承载了计算机可读的程序代码。这种传播的数据信号可以采用多种形式，包括——但不限于——电磁信号、光信号或上述的任意合适的组合。计算机可读的信号介质还可以是计算机可读存储介质以外的任何计算机可读介质，该计算机可读介质可以发送、传播或者传输用于由指令执行系统、装置或者器件使用或者与其结合使用的程序。

[0025] 计算机可读介质上包含的程序代码可以用任何适当的介质传输，包括——但不限于——无线、电线、光缆、RF等等，或者上述的任意合适的组合。

[0026] 可以以一种或多种程序设计语言或其组合来编写用于执行本发明操作的计算机程序代码，所述程序设计语言包括面向对象的程序设计语言—诸如Java、Smalltalk、C++，还包括常规的过程式程序设计语言—诸如"C"语言或类似的程序设计语言。程序代码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)。

[0027] 下面将参照本发明实施例的方法、装置(系统)和计算机程序产品的流程图和/或框图描述本发明。应当理解，流程图和/或框图的每个方框以及流程图和/或框图中各方框的组合，都可以由计算机程序指令实现。这些计算机程序指令可以提供给通用计算机、专用计算机或其它可编程数据处理装置的处理器，从而生产出一种机器，这些计算机程序指令通过计算机或其它可编程数据处理装置执行，产生了实现流程图和/或框图中的方框中规定的功能/操作的装置。

[0028] 也可以把这些计算机程序指令存储在能使得计算机或其它可编程数据处理装置以特定方式工作的计算机可读介质中，这样，存储在计算机可读介质中的指令就产生出一个包括实现流程图和/或框图中的方框中规定的功能/操作的指令装置(instruction means)的制造品(manufacture)。

[0029] 也可以把计算机程序指令加载到计算机、其它可编程数据处理装置、或其它设备上，使得在计算机、其它可编程数据处理装置或其它设备上执行一系列操作步骤，以产生计算机实现的过程，从而使得在计算机或其它可编程装置上执行的指令能够提供实现流程图和/或框图中的方框中规定的功能/操作的过程。

[0030] 图1示出了适于用来实现本发明实施方式的示例性计算机系统/服务器12的框图。图1显示的计算机系统/服务器12仅仅是一个示例，不应对本发明实施例的功能和使用范围带来任何限制。

[0031] 如图1所示，计算机系统/服务器12以通用计算设备的形式表现。计算机系统/服务器12的组件可以包括但不限于：一个或者多个处理器或者处理单元16，系统存储器28，连接不同系统组件(包括系统存储器28和处理单元16)的总线18。

[0032] 总线18表示几类总线结构中的一种或多种，包括存储器总线或者存储器控制器，外围总线，图形加速端口，处理器或者使用多种总线结构中的任意总线结构的局域总线。举例来说，这些体系结构包括但不限于工业标准体系结构(ISA)总线，微通道体系结构(MAC)总线，增强型ISA总线、视频电子标准协会(VESA)局域总线以及外围组件互连(PCI)总线。

[0033] 计算机系统/服务器12典型地包括多种计算机系统可读介质。这些介质可以是任何能够被计算机系统/服务器12访问的可用介质，包括易失性和非易失性介质，可移动的和不可移动的介质。

[0034] 系统存储器28可以包括易失性存储器形式的计算机系统可读介质，例如随机存取存储器(RAM)30和/或高速缓存存储器32。计算机系统/服务器12可以进一步包括其它可移动/不可移动的、易失性/非易失性计算机系统存储介质。仅作为举例，存储系统34可以用于读写不可移动的、非易失性磁介质(图1未显示，通常称为“硬盘驱动器”)。尽管图1中未示出，可以提供用于对可移动非易失性磁盘(例如“软盘”)读写的磁盘驱动器，以及对可移动非易失性光盘(例如CD-ROM，DVD-ROM或者其它光介质)读写的光盘驱动器。在这些情况下，每个驱动器可以通过一个或者多个数据介质接口与总线18相连。存储器28可以包括至少一个程序产品，该程序产品具有一组(例如至少一个)程序模块，这些程序模块被配置以执行本发明各实施例的功能。

[0035] 具有一组(至少一个)程序模块42的程序/实用工具40，可以存储在例如存储器28中，这样的程序模块42包括——但不限于——操作系统、一个或者多个应用程序、其它程序模块以及程序数据，这些示例中的每一个或某种组合中可能包括网络环境的实现。程序模块42通常执行本发明所描述的实施例中的功能和/或方法。

[0036] 计算机系统/服务器12也可以与一个或多个外部设备14(例如键盘、指向设备、显示器24等)通信，还可与一个或者多个使得用户能与该计算机系统/服务器12交互的设备通信，和/或与使得该计算机系统/服务器12能与一个或多个其它计算设备进行通信的任何设备(例如网卡，调制解调器等等)通信。这种通信可以通过输入/输出(I/O)接口22进行。并且，计算机系统/服务器12还可以通过网络适配器20与一个或者多个网络(例如局域网(LAN)，广域网(WAN)和/或公共网络，例如因特网)通信。如图所示，网络适配器20通过总线18与计算机系统/服务器12的其它模块通信。应当明白，尽管图中未示出，可以结合计算机系统/服务器12使用其它硬件和/或软件模块，包括但不限于：微代码、设备驱动器、冗余处理单元、外部磁盘驱动阵列、RAID系统、磁带驱动器以及数据备份存储系统等。

[0037] 首先参照图2，描述根据本发明实施例的用于MapReduce架构中的资源管理的方法200。

[0038] 如图2所示，方法200包括：在S210中，确定Map任务的输入数据量和Map任务的输出数据量之比r以及Map输出结果中记录的平均大小R；在S220中，确定一个Map资源槽对应的能够分配给Map任务的内存大小Memory_Size；在S230中，根据所确定的r、R和Memory_Size，确定适合该Map任务的输入切片大小；以及在S240中，将待处理的MapReduce作业中具有该输入切片大小的输入分片分配给占用该Map资源槽的Map任务。

[0039] 上述方法200例如可以由应用MapReduce架构的网络中的主控节点执行，也可以由将处理Map任务的计算节点执行，还可以由主控节点和计算节点配合执行。通过向占用Map资源槽的Map任务分配与该Map资源槽对应的内存相匹配的输入分片，可以避免数据多次被溢出到磁盘而产生的反复读写磁盘，由此提高处理效率。

[0040] 这里，资源槽对应的内存指的是占用该资源槽的任务能够被分配的内存。具体而言，Map资源槽对应的内存表示占用该Map资源槽的Map任务能够被分配的内存，以及，在下文中描述的Reduce资源槽对应的内存表示占用该Reduce资源槽的Reduce任务能够被分配的内存。

[0041] 根据本发明的实施例，作为Map任务的输入数据量和Map任务的输出数据量之比的r和作为Map输出结果中记录的平均大小的R可以通过探测的方式得到，也可以通过读取历史记录得到。

[0042] 具体而言，例如，可以预先运行一个或多个Map任务，并根据运行后得到的结果确定r和R。预先运行的Map任务可以是任意MapReduce作业对应的Map任务，例如，可以是待处理的MapRuduce作业对应的Map任务，也可以是提前准备好的由测试数据构成的Map任务。由于通常每个Map任务包含的记录数量非常多，因此对于一个或多个Map任务的统计特征可以基本上表征任意Map任务的个体特征。通常认为在MapReduce架构中每个Mapper(映射器，处理Map任务的处理器)的工作负载特性基本上相同，不同Map任务的输入数据量和输出数据量之间的比例关系近似，并且每个Map输出记录的长度也非常近似。因此，可以通过预先运行一个Map任务，根据该Map任务的输入数据量和输出数据量来确定r，并根据Map任务的输出数据量和输出记录数来确定R。其中，r等于Map任务的输入字节除以Map任务的输出字节，R等于Map任务的输出字节除以Map任务的输出记录数。也可以预先运行多个Map任务，将这些Map任务中的每一个的输入和输出数据量之比取平均值来作为r，将这些Map任务所有输出字节之和除以输出的总记录数求得的商作为R。

[0043] 再例如，由于MapReduce架构在执行一个MapReduce作业之后将生成日志信息，因此通过阅读历史日志信息，可以确定r和R。本领域技术人员都清楚地知道在MapReduce作业处理之后，通过任务级的内建计数器可以得到包含Map任务的输入字节、Map任务的输出字节和Map输出记录数等的日志信息。本发明实施例充分利用这些日志信息来提取所需的r和R。具体而言，通过将日志信息中的Map任务的输入字节除以Map任务的输出字节可以得到r，以及通过将Map输出字节除以Map输出记录数可以得到R。

[0044] 能够分配给Map任务的内存大小Memory_Size表示在Map任务占用Map资源槽的情况下，该Map资源槽对应的能够用于处理该Map任务的内存大小。例如，在Java编程中，能够分配给诸如Map任务、Reduce任务等任务的内存大小可以用JVM_Heap_Size表示。当然，本领域技术人员可以理解，能够分配给Map任务的内存大小也可以用其它参数形式来表示。要注意的是，在这里各参数的表现形式只是为了使本领域技术人员能够更好地理解本发明，而并不对本发明的保护范围构成任何限制。根据本发明的一个实施例，在MapReduce架构用于同构网络的情况下，可以根据用于并行执行Map任务的资源槽总数和用于并行执行Map任务的总内存，来确定Memory_Size。例如，首先，可以根据核(例如CPU)的数量来确定Map资源槽能力(即，一次可以同时执行的Map任务个数)Mc和Reduce资源槽能力(即，一次可以同时执行的Reduce任务个数)Rc，此时可以确定Mc和Rc都等于核的数量，即Mc＝Rc＝核的数量。由

于每个核可以同时兼处理Map任务和Reduce任务，因此一次可以同时执行的Map任务个数等于核数，一次可以同时执行的Reduce任务个数也等于核数。然后，可以确定Memory_Size=总内存/(Mc+Rc)。这里的总内存指的是上述的核能够利用的总内存。由于同构网络，因此可以认为每个资源槽对应的资源情况是相同的。从而，每个Map资源槽对应的能够分配给Map任务的内存大小都是Memory_Size。这样，只需要计算一个Memory_Size，就可以得到每个Map资源槽对应的能够分配给Map任务的内存大小。

[0045] 根据本发明的另一实施例，在MapReduce架构用于异构网络的情况下，可以响应于一个Map资源槽空闲，确定该Map资源槽对应的内存大小，并将所确定的该内存大小作为Memory_Size。具体而言，由于在异构网络中每个资源槽对应的能够分配给任务的内存可能不同，因此需要根据每个资源槽的具体情况来确定该资源槽对应的内存大小，并将其作为该资源槽对应的Memory_Size。在某一Map资源槽空闲的情况下，由于该Map资源槽对应的资源之前已经处理过Map任务，因此该Map资源槽对应多少内存是被网络中的主控节点和/或该Map资源槽所在的计算节点所知晓的，由此可以确定该Map资源槽对应的能够用于处理Map任务的内存大小Memory_Size。

[0046] 确定了r、R和Memory_Size之后，可以确定与该Memory_Size相匹配的Map任务的输入切片大小。例如可以采用图3所示的步骤S310和S320来确定输入切片大小Map_input_split_size。

[0047] 在S310中，根据R、Map输出结果中一个记录所对应的管理开销和Memory_Size，确定Memory_Size中用于存储除管理开销之外的数据的内存大小Data_Buffer。

[0048] 例如，在执行Map任务的情况下，数据在内存中和在磁盘中的存储方式都可以具有由管理开销和数据部分组成的数据结构。举例来说，在现有的一种MapReduce架构(例如，Apache Hadoop MapReduce实现方式)中，Map任务的数据在内存中和在磁盘中进行存储的数据结构可以针对记录进行定义，每个记录具有相同的数据结构。在该数据结构中，包括16字节的管理开销和R字节的数据部分。16字节的管理开销包括12字节的kvoffsets字段和4字节的kvindices字段。R字节的数据部分存储在kvbuffer字段中。当然，上述数据结构只是一个例子而并不对本发明的保护范围构成限制，随着技术的发展，可能增加新的字段并占用新的字节等。

[0049] 管理开销的字节数M可以由数据结构确定，也可以由所采用的数据存储标准确定。由于R在S210中已经通过探测或者读取历史信息的方式被确定，因此可以确定管理开销在记录中所占的比例io.sort.record.percent=M/(M+R)，或者数据在一个记录中所占的比例R/(M+R)。接着，可以确定在Map资源槽的内存Memory_Size中用于存储除管理开销之外的数据的内存大小Data_Buffer=Memory_Size*(1-io.sort.record.percent)=Memory_Size*R/(M+R)。

[0050] 在S320中，根据Data_Buffer和r，确定适合Map任务的输入切片大小。

[0051] 具体而言，由于Map任务的输入和输出数据量之比在S210中已经确定，因此可以适合确定Map任务的输入切片大小(即，适合该Map任务的输入数据量)Map_input_split_size=Data_Buffer*r。

[0052] 在MapReduce架构用于同构网络的情况下，由于每个Map资源槽可以认为具有相同的资源情况，所以每个Map资源槽能够处理的Map任务的输入切片大小可以认为相等。因此，

在确定Map_input_split_size之后，可以根据待处理的MapReduce作业的总输入数据量和Map任务的输入切片大小，确定该MapReduce作业的Map任务数量。即，该MapReduce作业需要的Map任务数量为该MapReduce作业的总输入数据量除以Map_input_split_size。

[0053] 可以确定按照上述方法得到的Map任务的输入切片大小与该Map任务占用的Map资源槽所对应的内存相匹配。通过将该切片大小的数据分配给该Map任务，不会由于切片大小过大且内存过小而产生将Map操作得到的每条记录多次溢出到磁盘的问题。也就是说，利用上述方法不会产生在Map任务的执行过程中反复读写磁盘的问题。此时，Map资源槽对应的内存可以一次性存储Map操作得到的所有记录，从而只需要对磁盘进行一次写操作(即进行到磁盘的一次溢出)就可以将Map输出结果保存在磁盘中以供Reduce任务处理。这样，处理效率得以提高，并且避免多次读写磁盘造成资源浪费和系统开销。

[0054] 虽然在方法200中S220在S210之后执行，但是S220也可以在S210之前执行，还可以与S210并发执行，本发明对此不作限制，只要S220和S210在S230之前执行即可。

[0055] 根据本发明实施例提供的上述方法，通过预先估计出一个Map资源槽对应的能够分配给Map任务的内存大小，可以根据该内存大小向占用该Map资源槽的Map任务分配相匹配的输入切片大小，从而尽可能避免输入切片大小和内存大小不匹配而导致的多次磁盘溢出以及在该情况下造成的对磁盘的反复读写，因此能够提高Map任务的处理效率，避免资源浪费。此外，由于利用上述方法可以自动得到包括内存大小、切片大小、任务数量等在内的参数，并且当利用这些参数时可以提高处理效率，因此，本发明实施例提供的方法可以对参数进行自动优化，避免经验设置或缺省设置参数导致的资源得不到有效利用的问题。

[0056] 上面描述了尽可能避免在Map操作时产生多次磁盘溢出的资源管理方法。然而，不必要的磁盘溢出的问题不一定只会发生在Map操作时，在Reduce操作时也可能发生不必要的磁盘溢出而降低处理效率。例如，由于Reduce任务数量的设置，可能导致Reduce任务的输入数据量大于该Reduce任务占用的Reduce资源槽中能够用于存储Reduce任务输入数据的内存。由于Reducer(化简器，处理Reduce任务的处理器)要等到所有输入给它的Map输出记录收集完全后才能开始处理，因此，如果输入给该Reducer的输入数据量大于它使用的内存，则需要溢出到磁盘，再通过从磁盘分多次读取能够处理的数据量并暂存中间结果，接着将所有中间结果进行归并排序才能得到Reduce函数可以使用的数据。

[0057] 为了进一步避免在Reduce操作时出现磁盘溢出而导致多次读写从而降低处理效率，可以使用根据本发明实施例的方法400。方法400中的S410至S440与方法200中的S210至S240基本相同，为了避免重复，在此不再赘述。

[0058] 在S450中，根据r和MapRuduce作业的总输入数据量，确定该MapReduce作业的Map输出结果的总数据量。

[0059] MapReduce作业的总输入数据量可以直接根据该MapReduce作业对应的文件大小确定。由于r表示Map任务的输入和输出数据量之比，因此，Map输出结果的总数据量(也就是，Reduce操作的总输入数据量)等于MapReduce作业的总输入数据量除以r。

[0060] 在S460中，确定一个Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小。

[0061] 一个Reduce资源槽对应的总内存除了用于存储Reduce任务的输入数据之外，还需要处理这些输入数据并输出处理结果等。因此，用于存储Reduce任务的输入数据的内存可

以是占用Reduce资源槽的Reduce任务被分配的总内存的一部分。为了折中存储Reduce输入数据的操作和包括处理Reduce函数及其后续操作的其他操作的处理性能，可以将存储Reduce输入数据的内存占用总内存的比例设置为0.5。当然，该比例也可以设置为小于1的其他系数。

[0062] 根据本发明的一个实施例，在MapReduce架构用于同构网络的情况下，可以根据用于并行执行Reduce任务的资源槽总数、用于并行执行Reduce任务的总内存以及小于1的预定系数，确定一个Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小。其中，所述预定系数是Reduce资源槽中存储Reduce输入数据的内存占该资源槽的总内存的比例。

[0063] 例如，在同构网络中可以认为每个Map资源槽和每个Reduce资源槽具有基本上同样的资源情况。如上所述，一个Map资源槽对应的能够分配给Map任务的内存大小是Memory_Size=总内存/(Mc+Rc)。在这种情况下，一个Reduce资源槽对应的能够分配给Reduce任务的内存大小也可以是Memory_Size。因此，Reduce资源槽对应的能够分配来存储Reduce输入数据的内存等于预定系数乘以Memory_Size。当预定系数是0.5以实现Reduce操作的处理性能折中时，Reduce资源槽对应的能够分配来存储Reduce输入数据的内存等于0.5*Memory_Size。

[0064] 根据本发明的另一实施例，在MapReduce架构用于异构网络的情况下，可以针对异构网络中执行MapReduce作业的各Reduce资源槽中的每一个，根据该Reduce资源槽对应的能够分配给Reduce任务的内存大小和小于1的预定系数，确定该Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小。例如，在异构网络中，主控节点和/或Reduce资源槽所在的计算节点知道该Reduce资源槽对应的内存，将该内存乘以预定系数(例如，0.5等其他小于1的系数)可以得到该Reduce资源槽对应的能够存储Reduce输入数据的内存大小。

[0065] 在S470中，根据S460中确定的存储Reduce任务的输入数据的内存大小和S450中确定的Map输出结果的总数据量，确定MapReduce作业的Reduce任务数量。

[0066] 具体而言，Reduce任务数量可以等于S450中的Map输出结果的总数据量除以S460中确定的内存大小。可以看到，当Reduce资源槽中用于存储Reduce输入数据的内存较小时，可以将MapReduce作业划分成较多的Reduce任务，这样，每个Reduce任务需要处理的输入数据可以减少以满足Reduce资源槽的较小的内存。在这种情况下，由于Reduce资源槽中存储Reduce输入数据的内存与Reduce输入数据可以尽量匹配，因此，可以尽量避免不必要的磁盘溢出而导致多次重复读写，由此可以提高处理效率。

[0067] 在MapReduce架构用于同构网络的情况下，每个Reduce资源槽可以具有相同的内存和内存分配，S470中计算出的Reduce任务数量是MapReduce作业实际上对应的Reduce任务数量。也就是说，在同构网络的情况下，S470中确定的一个Reduce任务对应一个Reducer。然而，在MapReduce架构用于异构网络的情况下，S470中确定的Reduce任务数量可以不等于实际的Reducer个数，下面进行具体说明。

[0068] 在MapReduce架构用于异构网络的情况下，由于Reduce资源槽之间可能具有不同的内存大小，在S460中确定的各Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小可能彼此不同。这些用于存储Reduce输入数据的内存大小可以具有一个公约

数。那么，根据本发明的实施例，可以根据S460中确定的各Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小的公约数和S450中确定的Map输出结果的总数据量，确定MapReduce作业的Reduce任务数量。这里的Reduce任务数量等于Map输出结果的总数据量除以上述公约数。这样计算出的Reduce任务数量大于实际的Reducer数量。

[0069] 在该情况下，可以根据异构网络中的各Reduce资源槽各自对应的内存大小和上述公约数的关系，将不止一个的根据上述公约数划分的Reduce任务分配给一个Reducer。具体来说，响应于一个Reduce资源槽空闲，根据该Reduce资源槽对应的能够分配来存储Reduce任务的内存大小和上述公约数，将具有如下个数的Reduce任务分配给占用该Reduce资源槽的Reduce任务，所述个数等于该Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小相对于该公约数的倍数。

[0070] 举例来说，假设在异构网络中有三个Reduce资源槽，每个Reduce资源槽对应的能够分配给Reduce任务的内存大小分别是1200MB、800MB和400MB。当预定系数是0.5时，这三个Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小分别是600MB、400MB和200MB。这里，取这三个内存大小的最大公约数200MB。通过200MB和Map输出结果的总数据量，可以确定Reduce任务数量。响应于内存为1200MB的Reduce资源槽空闲，将3个(600MB/200MB)Reduce任务分配给该Reduce资源槽。响应于内存为800MB的Reduce资源槽空闲，将2个(400MB/200MB)Reduce任务分配给该Reduce资源槽。响应于内存为400MB的Reduce资源槽空闲，将1个(200MB/200MB)Reduce任务分配给该Reduce资源槽。当然，公约数也可以是诸如100MB的其他公约数而不一定是最大公约数。

[0071] 使用最大公约数的优点之一是充分利用每个计算节点中的内存。在Reduce任务槽确定的情况下，能够并行运行的Reduce任务数量也是确定的。如果Reduce任务的输入数据量太小，那么计算节点上的内存将不能得到充分利用，而如果Reduce任务的输入数据量太大，那么计算节点就会将Reduce任务输入数据溢出到磁盘。通过以公约数来进行Reduce任务数的划分，可以为占用Reduce资源槽的Reduce任务分配相适宜的输入数据，从而适应该Reduce资源槽自身的资源情况，在充分利用其资源的情况下尽量避免不必要的磁盘溢出所导致的处理效率的降低。

[0072] 根据本发明的实施例，响应于占用Reduce资源槽的Reduce任务的读取Map输出结果的操作完成，开始还未占用Reduce资源槽的Reduce任务的读取Map输出结果的操作。

[0073] 在传统的MapReduce架构中，只有占用了Reduce资源槽的Reduce任务才能读取相应的Map输出结果，即，进行Shuffle(搬移)操作。等待占用了Reduce资源槽的Reduce任务执行完毕之后，该Reduce任务释放占用的Reduce资源槽。空闲的Reduce资源槽被另一Reduce任务占用后，才能开始向该另一Reduce任务传输对应的Map输出结果，即，进行与占用Reduce资源槽的该另一Reduce任务对应的Shuffle操作。这样，一个MapReduce作业的执行将有较大的延时。每当Reduce资源槽空闲出来之后，才能开始其他Reduce任务的数据读取和执行。

[0074] 根据本发明的实施例，可以为Reduce任务的状态机在已有的等待(Pending)状态和运行(Running)状态之间增加一个数据读取状态，如图5所示。处于数据读取状态下的Reduce任务，虽然还没有占用Reduce资源槽，但是其Shuffle操作已经可以开始执行了。这样，一旦处于数据读取状态的Reduce任务占用Reduce资源槽，该Reduce任务就可以跳过如

现有技术中那样需要先读取Map输出结果，而直接进入运行状态，由此节约处理时间，进一步提高处理效率。

[0075] 具体而言，处于等待状态的Reduce任务是还未占用Reduce资源槽的Reduce任务。在正在占用Reduce资源槽的Reduce任务中存在一个Reduce任务的数据读取操作完成的情况下，该Reduce任务的Reduce函数开始执行，而还未占用Reduce资源槽的Reduce任务则可以进入数据读取状态，开始读取它所对应的Map输出结果(即，Shuffle操作)。当有Reduce资源槽空闲出来时，通过主控节点的调度，所述还未占用Reduce资源槽的Reduce任务占用空闲的Reduce资源槽，并进入运行状态而开始Reduce函数的执行。

[0076] 此外，在有Reduce任务的Shuffle操作完成的情况下，可以进入数据读取状态的Reduce任务的数量可以不止一个。

[0077] 通过为Reduce任务的执行增加数据读取状态，相比于相关技术，可以提前向未占用Reduce资源槽的Reduce任务传输对应的Map输出结果，从而在实际的Reduce任务数(即，实际的reducer个数)大于Reduce资源槽的情况下，减少执行所有Reduce任务的延时，从而进一步提高处理效率。

[0078] 虽然在方法400中S450在S440之后执行，但是S450与S420至S440之间的执行关系没有特别的要求，只要S450在S410之后执行即可。此外，S450和S460之间的执行顺序本发明也不作限制，只要S450和S460在S470之前执行即可。

[0079] 图6具体示出了利用根据本发明的一个实施例的方法来处理一个MapReduce作业的示例性方法600的流程图。通过执行图6中的方法600，可以同时尽量避免在Map操作和Reduce操作中出现磁盘溢出。这里，以异构网络为例对方法600进行描述，但是该流程同样适用于同构网络，不同之处将在下文具体描述。当然，本领域技术人员可以理解，同构网络作为异构网络的一种特例，方法600的流程也可以完全适用于同构网络，另外，这里，以网络中的主控节点作为执行主体为例来对方法600进行描述，但是本领域技术人员可以容易地想到方法600也可以由除主控节点之外的其他节点来执行，或者由主控节点和其他节点配合执行。

[0080] 在S610中，方法600开始。

[0081] 在S615中，主控节点通过尝试一个或多个Map任务来得到Map任务的输入和输出数据量之比r和Map输出结果中一个记录的平均大小R。

[0082] 在S620中，主控节点确定是否有Map资源槽可用，即，是否有Map资源槽空闲。如果确定有Map资源槽可用，则方法600前进到S625，否则方法600继续S620。

[0083] 在S625中，主控节点基于r、R以及可用的Map资源槽对应的能够分配给Map任务的内存大小Memory_Size，确定占用该Map资源槽的Map任务的输入切片大小。

[0084] 在同构网络的情况下，S625可以只执行一次而被置于S615和S620之间，因为在同构网络的情况下，可以认为每个Map资源槽具有相同的Map任务的输入切片大小。

[0085] 在S630中，主控节点基于计算出的Map任务的输入切片大小，将MapReduce作业中具有该输入切片大小的数据动态分配给占用该Map资源槽的Map任务。

[0086] 在S635中，占用该Map资源槽的Map任务执行。

[0087] 在S640中，主控节点确定是否所有的Map任务都已经执行完成。如果确定所有的Map任务都已经执行完成，则方法600前进到S645，否则方法600返回S620，以等待有Map资源

槽空闲来处理新的Map任务。

[0088] 在S645中，主控节点根据各Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小，确定这些内存大小的公约数，从而将Map输出结果的数据量除以公约数的商作为“mini”Reducer的个数。所述公约数可以认为是一个“mini”Reducer的输入数据大小。这里，“mini”Reducer的个数是在假设公约数是Reducer的输入数据量的情况下所计算出的Reducer个数，该个数可能大于实际存在的Reducer个数。

[0089] 虽然，在方法600中S645在S640之后执行，但是S645执行顺序与S615至S640没有关系。S645只要在S650之前即可，而S650由于需要对Map输出结果进行Partition(分片)操作，因此需要在S640之后执行。

[0090] 在同构网络的情况下，因为可以认为每个Reduce资源槽对应的能够分配给Reduce任务的内存大小相同，因此不需要执行在S645中求取公约数和“mini”Reducer的个数。在同构网络中，在S645中，根据每个Reduce资源槽对应的能够分配来存储Reduce输入数据的内存大小和Map输出结果的数据量，来计算Reducer个数，此时计算出的Reducer个数等于实际的Reducer个数。

[0091] 在S650中，主控节点根据S645中的Reduce任务数量对Map输出结果进行Partition操作。

[0092] 在S655中，主控节点确定是否有Reduce资源槽可用，即，是否有Reduce资源槽空闲。如果确定有Reduce资源槽可用，则方法600前进到S660，否则方法600继续S655。

[0093] 在S660中，主控节点根据可用的Reduce资源槽对应的能够分配给Reduce任务的Memory_Size以及小于1的预定系数，确定出该Reduce资源槽对应的能够用于存储Reduce输入数据的内存大小，并将该内存大小作为占用该Reduce资源槽的Reduce任务的输入数据量。接着，主控节点根据该Reduce任务的输入数据量相对于公约数的倍数，将倍数个数的“mini”Reducer合并来分配给占用该Reduce资源槽的Reduce任务。

[0094] 在同构网络的情况下，由于在S645中计算得到的Reducer个数等于实际的Reduce个数，不存在“mini”Reducer个数的求取，因此在S645中不需要合并一定个数的“mini”Reducer。在同构网络的情况下，由于各Reduce资源槽可以认为具有相同的资源情况，因此在S660中，主控节点根据各Reduce资源槽对应的能够分配给Reduce任务的Memory_Size以及小于1的预定系数，确定各Reduce资源槽对应的能够用于存储Reduce输入数据的内存大小。由于该内存大小对于各Reduce资源槽相同，因此S660可以只需要执行一次而被置于S655之前。

[0095] 在S665中，占用该Reduce资源槽的Reduce任务执行。

[0096] 在同构网络中，如常规技术中那样，占用该Reduce资源槽的Reduce任务被分配有S650中的Partition操作之后对应该Reduce任务的Reduce输入数据。

[0097] 在S670中，主控节点确定是否所有的Reduce任务都已经执行完成。如果确定所有的Reduce任务都已经执行完成，那么方法600前进到S675，否则方法600返回S655。

[0098] 在S675中，方法600结束。

[0099] 以同构网络为例对本发明实施例的方法进行说明。假设，首先通过提前运行多个Map任务可以探测到r为3、R为每记录84字节。在同构网络中可用于处理MapReduce作业的计算节点有4个，每个计算节点具有2个核和1200MB的内存。此外，MapReduce作业的总输入数

据量为10GB，并且数据在内存中和在磁盘中的数据结构除了R之外还包括每记录16字节的管理开销。当利用根据本发明的一个实施例的方法时，可以计算得到Mc＝Rc＝核数＝2×4＝8，Memory_Size＝1200MB×4/[(8+8)＝300MB，Data_Buffer＝300MB×84/(16+84)=252MB，Map_input_split_size＝252MB×3＝756MB，Map任务数量＝10GB/756MB=13个，Reduce输入大小＝10GB/3＝3.33GB，Reduce任务数量＝3.33GB/(0.5×300MB)=23个。

[0100] 根据本发明实施例提供的上述方法，通过根据资源槽对应的内存大小，向占用该资源槽的任务分配与其内存相匹配的输入数据，可以尽可能避免不必要的磁盘溢出，同时能够对参数进行自动优化以充分利用各资源槽的核资源和内存资源，并可以提高处理效率。此外，通过将总内存除以(Mc+Rc)来求取Memory_Size并基于Memory_Size来确定切片大小，可以支持流水线的MapReduce操作。另外，即便在异构网络中，通过根据各资源槽自身的资源情况进行按需分配，也可以避免不必要的磁盘溢出，同时提高处理效率并避免资源浪费。

[0101] 上面描述了根据本发明实施例的用于MapReduce架构中的资源管理的方法，接下来描述根据本发明实施例的用于MapReduce架构中的资源管理的装置以及MapReduce架构系统的结构框图。

[0102] 如图7所示，根据本发明实施例的用于MapReduce架构中的资源管理的装置700包括第一确定部件710、第二确定部件720、第三确定部件730和第一分配部件740。第一确定部件710可被配置为确定Map任务的输入数据量和Map任务的输出数据量之比r以及Map输出结果中记录的平均大小R。第二确定部件720可被配置为确定一个Map资源槽对应的能够分配给Map任务的内存大小Memory_Size。第三确定部件730可被配置为根据所确定的r、R和Memory_Size，确定适合该Map任务的输入切片大小。第一分配部件740可被配置为将待处理的MapReduce作业中具有该输入切片大小的输入分片分配给占用该Map资源槽的Map任务。

[0103] 第一确定部件710、第二确定部件720、第三确定部件730和第一分配部件740的上述和/或其他操作和功能可以参考上述方法200的相关描述，为了避免重复，在此不再赘述。

[0104] 根据本发明实施例提供的上述装置，通过预先估计出一个Map资源槽对应的能够分配给Map任务的内存大小，可以根据该内存大小向占用该Map资源槽的Map任务分配相匹配的输入切片大小，从而尽可能避免输入切片大小和内存大小不匹配而导致的多次磁盘溢出以及在该情况下造成的对磁盘的反复读写，因此能够提高Map任务的处理效率，避免资源浪费。

[0105] 图8中示出了用于MapReduce架构中的资源管理的另一装置800的结构框图。装置800中的第一确定部件810、第二确定部件820、第三确定部件830和第一分配部件840分别与装置700中的第一确定部件710、第二确定部件720、第三确定部件730和第一分配部件740基本相同。

[0106] 根据本发明的实施例，第一确定部件810可以包括第一确定单元812和第二确定单元814中的至少一个。第一确定单元810可被配置为预先运行一个或多个Map任务，并根据运行后得到的结果确定r和R。第二确定单元820可被配置为根据与历史上运行过的Map任务相关的日志信息，确定r和R。

[0107] 根据本发明的一个实施例，第三确定部件830可以包括第三确定单元832和第四确定单元834。第三确定单元832可被配置为根据R、Map输出结果中一个记录所对应的管理开

销和Memory_Size，确定Memory_Size中用于存储除所述管理开销之外的数据的内存大小Data_Buffer。第四确定单元834可被配置为根据Data_Buffer和r，确定适合Map任务的输入切片大小。

[0108] 根据本发明的一个实施例，在MapReduce架构用于同构网络的情况下，第二确定部件820可被配置为根据用于并行执行Map任务的资源槽总数和用于并行执行Map任务的总内存，确定Memory_Size。

[0109] 根据本发明的一个实施例，装置800还可以包括第四确定部件850。第四确定部件850可被配置为根据MapReduce作业的总输入数据量和输入切片大小，确定MapReduce作业的Map任务数量。

[0110] 根据本发明的一个实施例，在MapReduce架构用于异构网络的情况下，第二确定部件820可被配置为响应于一个Map资源槽空闲，确定该Map资源槽对应的内存大小，作为Memory_Size。

[0111] 根据本发明的一个实施例，装置800还可以包括第五确定部件860、第六确定部件870和第七确定部件880。第五确定部件860可被配置为根据r和MapRuduce作业的总输入数据量，确定MapReduce作业的Map输出结果的总数据量。第六确定部件870可被配置为确定一个Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小。第七确定部件880可被配置为根据存储Reduce任务的输入数据的内存大小和Map输出结果的总数据量，确定MapReduce作业的Reduce任务数量。

[0112] 根据本发明的一个实施例，在MapReduce架构用于同构网络的情况下，第六确定部件870可被配置为根据用于并行执行Reduce任务的资源槽总数、用于并行执行Reduce任务的总内存以及小于1的预定系数，确定一个Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小。

[0113] 根据本发明的一个实施例，在MapReduce架构用于异构网络的情况下，第六确定部件870可被配置为针对异构网络中执行MapReduce作业的各Reduce资源槽中的每一个，根据该Reduce资源槽对应的能够分配给Reduce任务的内存大小和小于1的预定系数，确定该Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小。并且，第七确定部件880可被配置为根据各Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小的公约数和Map输出结果的总数据量，确定MapReduce作业的Reduce任务数量。在该情况下，装置800还可以包括第二分配部件890。第二分配部件890可被配置为响应于一个Reduce资源槽空闲，根据该Reduce资源槽对应的能够分配来存储Reduce任务的内存大小和所述公约数，将具有如下个数的Reduce任务分配给占用该Reduce资源槽的Reduce任务，所述个数等于该Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小相对于所述公约数的倍数。

[0114] 根据本发明的实施例，所述公约数可以是100MB或最大公约数。

[0115] 根据本发明的实施例，上述预定系数可以是0.5。

[0116] 根据本发明的一个实施例，装置800还可以包括控制部件895。控制部件895可被配置为响应于占用Reduce资源槽的Reduce任务的读取Map输出结果的操作完成，开始还未占用Reduce资源槽的Reduce任务的读取Map输出结果的操作。

[0117] 第一确定单元812、第二确定单元814、第二确定部件820、第三确定单元832、第四

确定单元834、第四确定部件850、第五确定部件860、第六确定部件870、第七确定部件880、第二分配部件890和控制部件895的上述和/或其它操作和功能可以参考上述方法200、300、400和600以及图5中的相关描述，为了避免重复，在此不再赘述。

[0118] 根据本发明实施例提供的上述装置，通过第四确定部件、第五确定部件和第六确定部件，可以进一步避免在Reduce操作中出现磁盘溢出，从而进一步提高处理效率。通过控制部件，可以减少执行所有Reduce任务的延时，从而还能进一步提高处理效率。此外，通过根据资源槽对应的内存大小，向占用该资源槽的任务分配与其内存相匹配的输入数据，可以尽可能避免不必要的磁盘溢出，同时能够对参数进行自动优化以充分利用各资源槽的核资源和内存资源，并可以提高处理效率。此外，利用本发明实施例提供的装置，还可以支持流水线的MapReduce操作。并且，即便在异构网络中，通过根据各资源槽自身的资源情况进行按需分配，也可以避免不必要的磁盘溢出，同时提高处理效率并避免资源浪费。

[0119] 上述的装置700和800可以被实现为单独的软件包或插件等，也可以被部分或全部集成到MapReduce架构。

[0120] 图9中示出的MapReduce架构系统900除了可以包括现有的部件之外，还可以包括用于资源管理的装置910。装置910可以是上述的装置700或800。

[0121] 附图中的流程图和框图显示了根据本发明的多个实施例的系统、方法和计算机程序产品的可能实现的体系架构、功能和操作。在这点上，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包含一个或多个用于实现规定的逻辑功能的可执行指令。也应当注意，在有些作为替换的实现中，方框中所标注的功能也可以以不同于附图中所标注的顺序发生。例如，两个连续的方框实际上可以基本并行地执行，它们有时也可以按相反的顺序执行，这依所涉及的功能而定。也要注意的是，框图和/或流程图中的每个方框、以及框图和/或流程图中的方框的组合，可以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，或者可以用专用硬件与计算机指令的组合来实现。

[0122] 以上已经描述了本发明的各实施例，上述说明是示例性的，并非穷尽性的，并且也不限于所披露的各实施例。在不偏离所说明的各实施例的范围和精神的情况下，对于本技术领域的普通技术人员来说许多修改和变更都是显而易见的。本文中所用术语的选择，旨在最好地解释各实施例的原理、实际应用或对市场中的技术的技术改进，或者使本技术领域的其它普通技术人员能理解本文披露的各实施例。

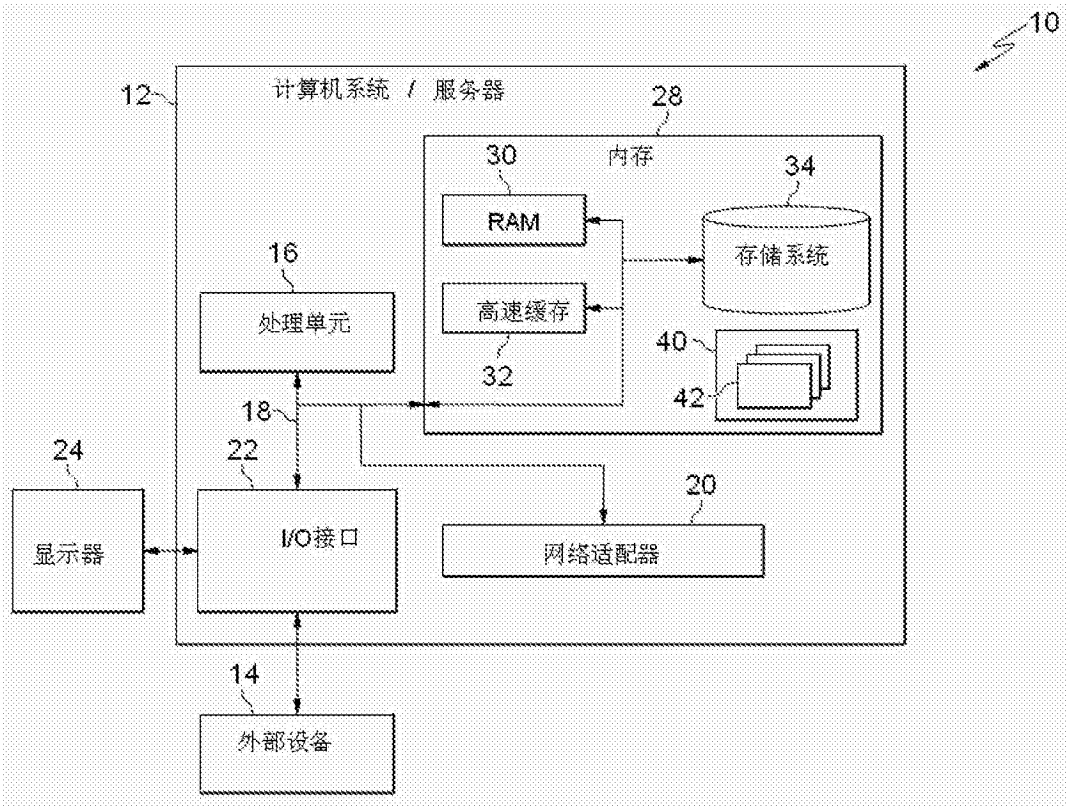

10
12
计算机系统 / 服务器
28
30
内存
34
RAM
存储系统
16
处理单元
高速缓存
32
40
42
18
24
22
20
显示器
I/O接口
网络适配器
14
外部设备

图1

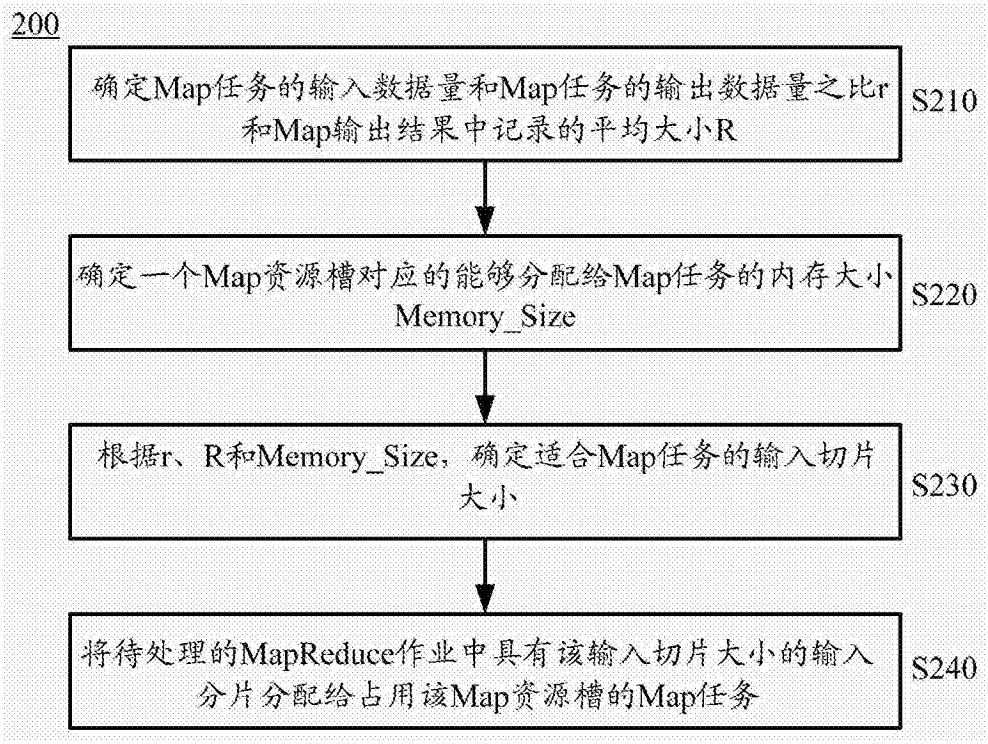

200
确定Map任务的输入数据量和Map任务的输出数据量之比r和Map输出结果中记录的平均大小R
S210
确定一个Map资源槽对应的能够分配给Map任务的内存大小Memory_Size
S220
根据r、R和Memory_Size，确定适合Map任务的输入切片大小
S230
将待处理的MapReduce作业中具有该输入切片大小的输入分片分配给占用该Map资源槽的Map任务
S240

图2

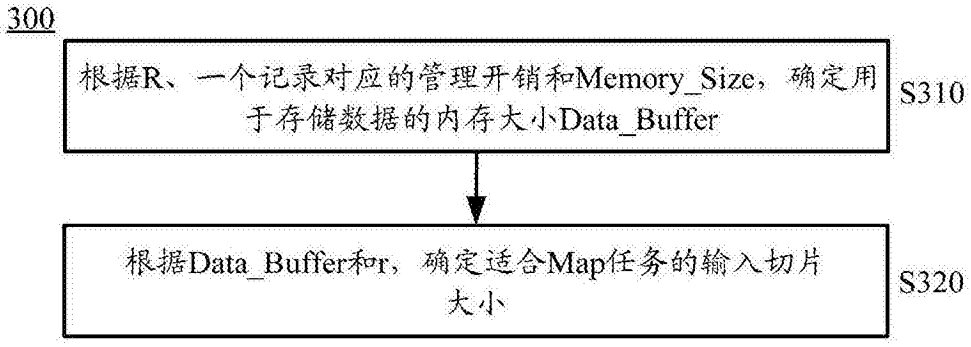

300
根据R、一个记录对应的管理开销和Memory_Size，确定用于存储数据的内存大小Data_Buffer
S310
根据Data_Buffer和r，确定适合Map任务的输入切片大小
S320

图3

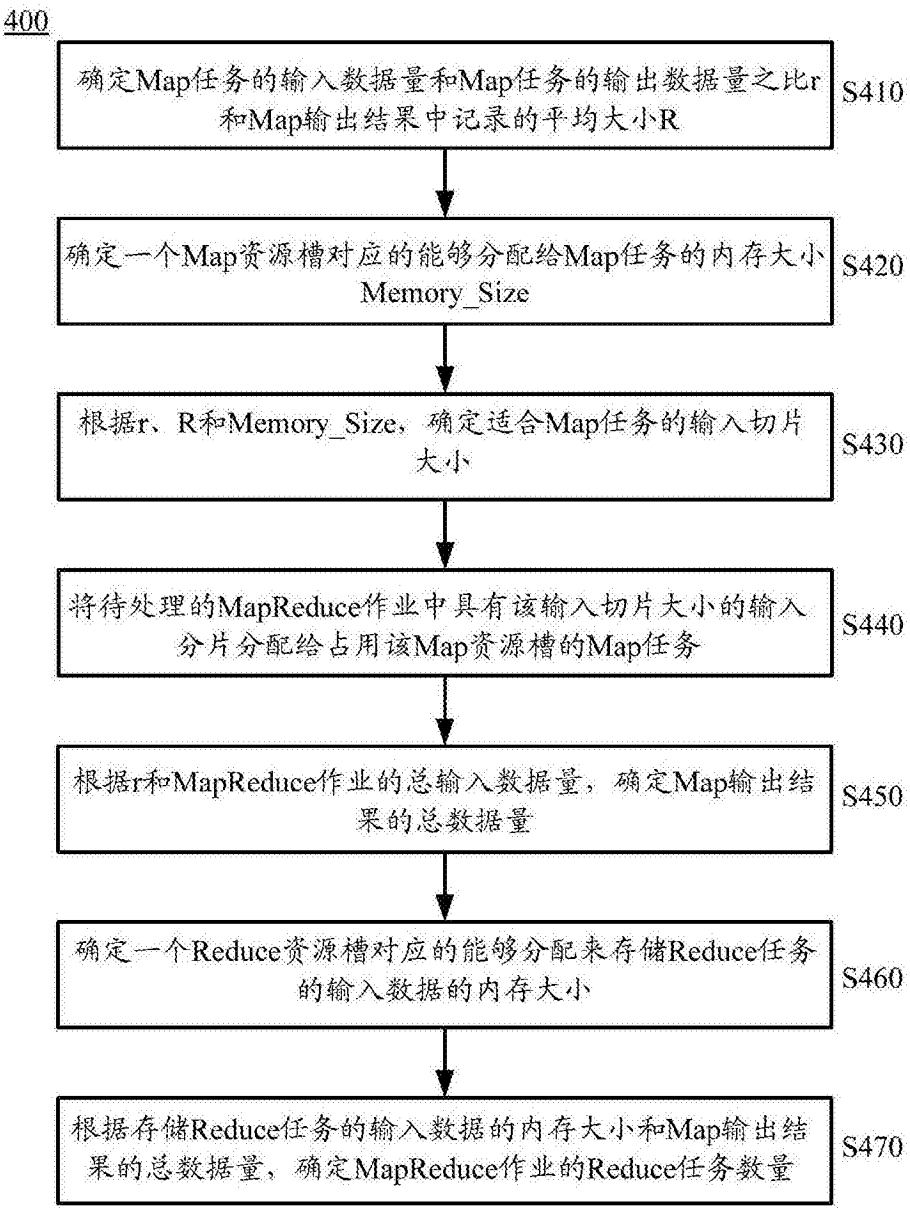

400
确定Map任务的输入数据量和Map任务的输出数据量之比r和Map输出结果中记录的平均大小R
S410
确定一个Map资源槽对应的能够分配给Map任务的内存大小Memory_Size
S420
根据r、R和Memory_Size，确定适合Map任务的输入切片大小
S430
将待处理的MapReduce作业中具有该输入切片大小的输入分片分配给占用该Map资源槽的Map任务
S440
根据r和MapReduce作业的总输入数据量，确定Map输出结果的总数据量
S450
确定一个Reduce资源槽对应的能够分配来存储Reduce任务的输入数据的内存大小
S460
根据存储Reduce任务的输入数据的内存大小和Map输出结果的总数据量，确定MapReduce作业的Reduce任务数量
S470

图4

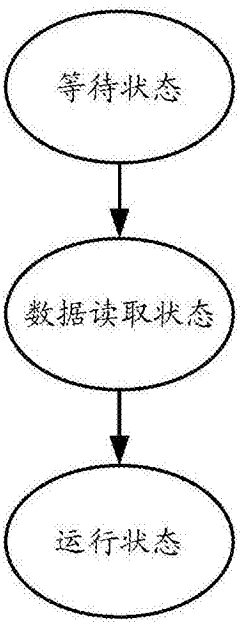

等待状态
数据读取状态
运行状态

图5

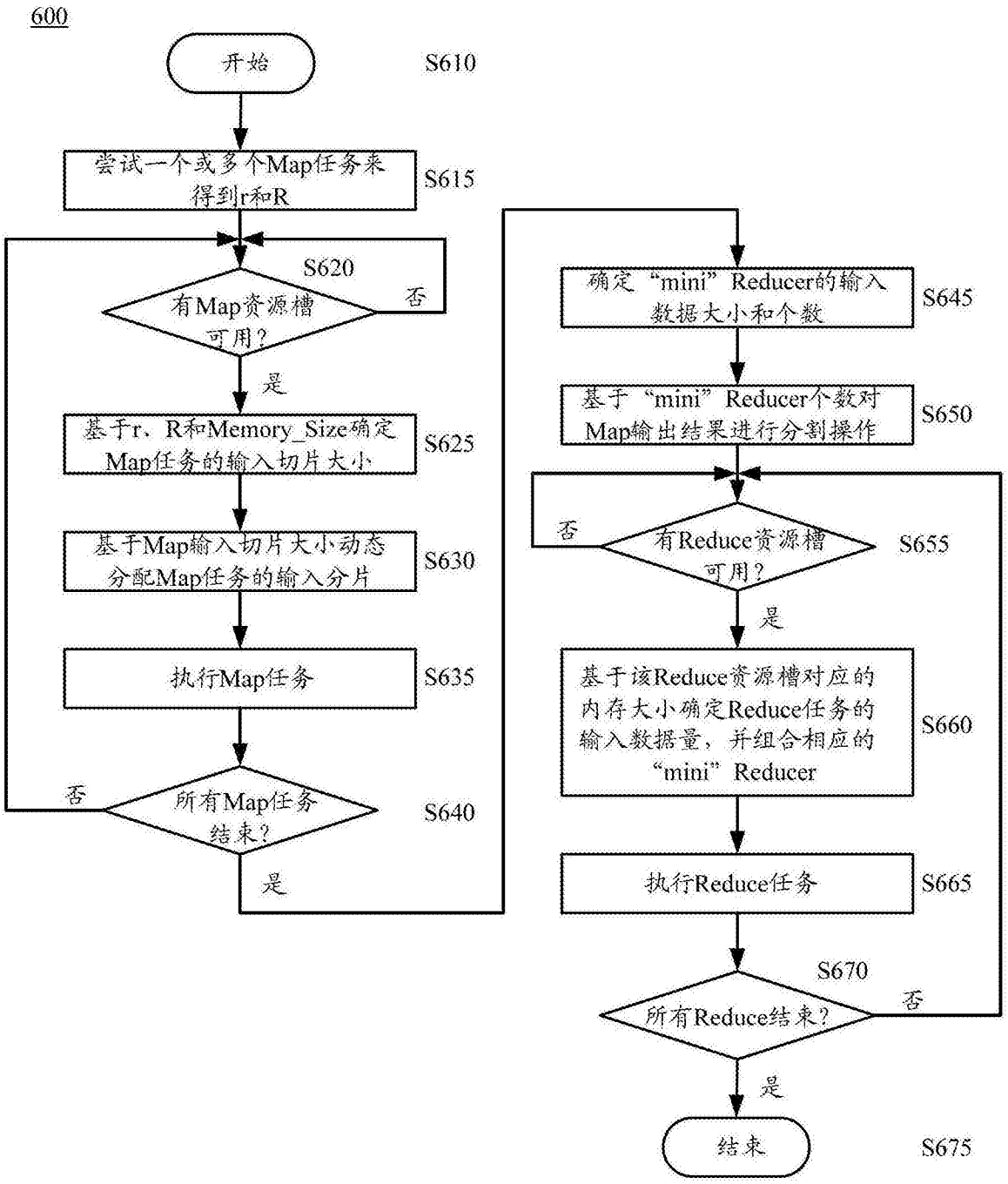

600
开始
S610
尝试一个或多个Map任务来得到r和R
S615
S620
有Map资源槽可用？
否
是
基于r、R和Memory_Size确定Map任务的输入切片大小
S625
基于Map输入切片大小动态分配Map任务的输入分片
S630
执行Map任务
S635
否
所有Map任务结束？
S640
是
确定“mini” Reducer的输入数据大小和个数
S645
基于“mini” Reducer个数对Map输出结果进行分割操作
S650
否
有Reduce资源槽可用？
S655
是
基于该Reduce资源槽对应的内存大小确定Reduce任务的输入数据量，并组合相应的“mini” Reducer
S660
执行Reduce任务
S665
S670
所有Reduce结束？
否
是
结束
S675

图6

图7

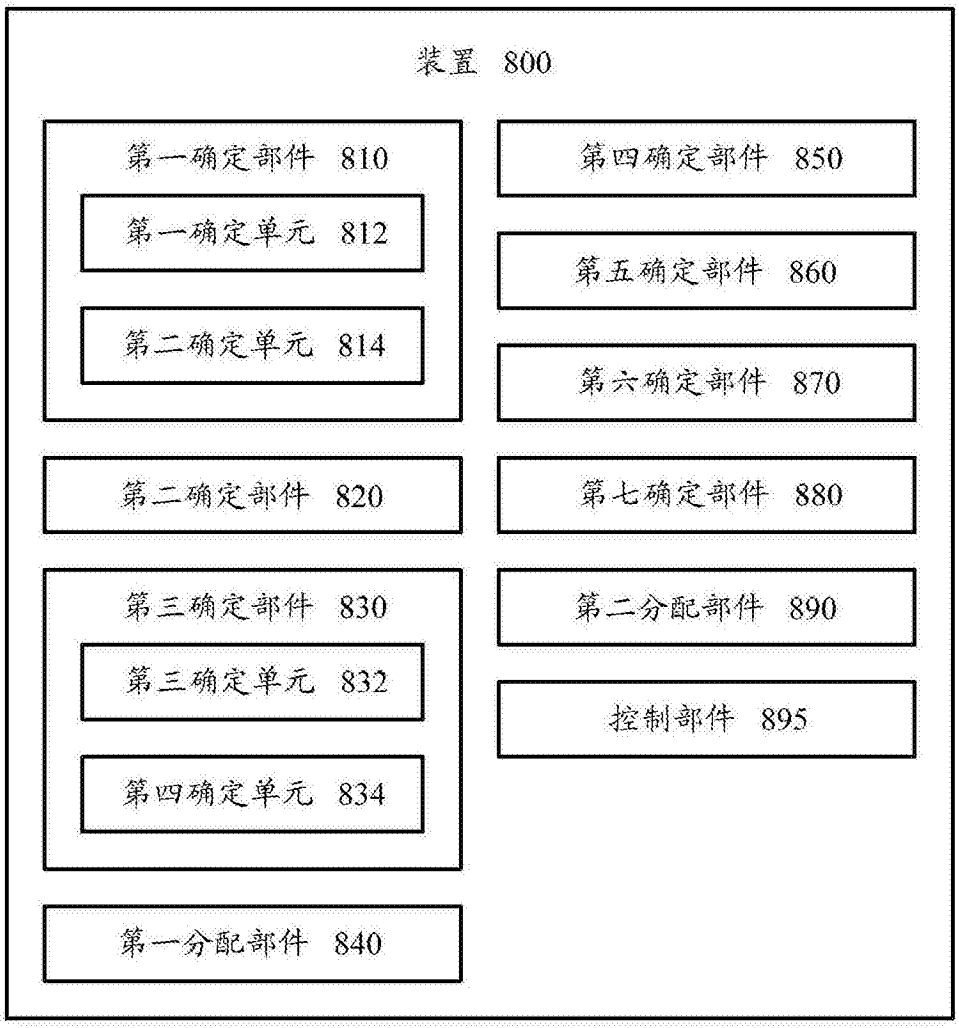

装置 800
第一确定部件 810
第一确定单元 812
第二确定单元 814
第二确定部件 820
第三确定部件 830
第三确定单元 832
第四确定单元 834
第一分配部件 840
第四确定部件 850
第五确定部件 860
第六确定部件 870
第七确定部件 880
第二分配部件 890
控制部件 895

图8

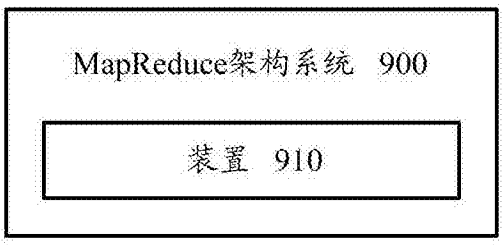

MapReduce架构系统 900
装置 910

图9