(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 107005596 B
(45)授权公告日 2020.12.08

(21)申请号 201580067419.1
(22)申请日 2015.12.15
(65)同一申请的已公布的文献号
申请公布号 CN 107005596 A
(43)申请公布日 2017.08.01
(30)优先权数据
14/577,629 2014.12.19 US
(85)PCT国际申请进入国家阶段日
2017.06.09
(86)PCT国际申请的申请数据
PCT/CN2015/097350 2015.12.15
(87)PCT国际申请的公布数据
WO2016/095791 EN 2016.06.23

(73)专利权人 华为技术有限公司
地址 518129 广东省深圳市龙岗区坂田华为总部办公楼
(72)发明人 蔡乐
(74)专利代理机构 广州三环专利商标代理有限公司 44202
代理人 郝传鑫 熊永强

(51)Int.Cl.
H04L 29/08(2006.01)
审查员 解淑瑄

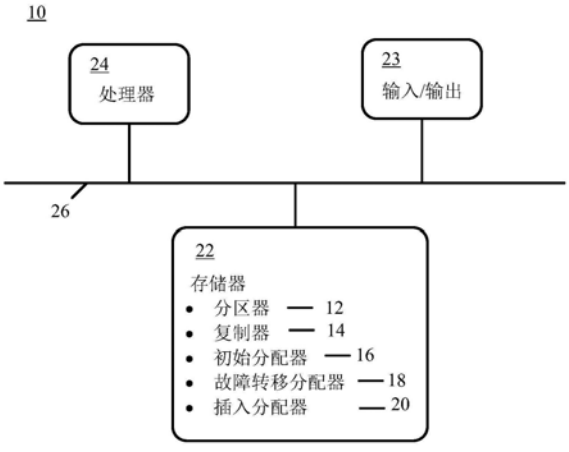
权利要求书3页 说明书12页 附图8页

(54)发明名称

用于在集群重新配置后的工作负载平衡的复制型数据库分配

(57)摘要

一种用于将多个数据分区分配在一群计算机节点之间的计算机实现方法,包括:将数据库划分成多个主分区;将所述主分区存储在与计算机网络相关联的多个节点上;复制所述主分区来创建多个从分区并将所述从分区存储在所述节点上,其中,所述主从分区大致均匀地分配在所述节点之间,各从分区而不是其对应的主分区存储在不同节点上,关于各从分区和存储其的节点存在分集,所述主分区可用来服务于系统工作负载,并且故障节点的一个或多个主分区对应的一个或多个从分区可用于在所述计算机节点中的一个故障之后就升级到主用状态来保持工作负载平衡和数据的可用性。



CN 107005596 B

1.一种用于在一群计算机节点之间分配多个数据分区的方法,其特征在于,所述方法包括:

将数据库划分成多个主分区;

将所述主分区存储在多个与计算机网络相关联的节点集群上,其中,所述主分区使用循环布局算法均匀地分配在所述节点集群之间,并且所述主分区可用来服务系统工作负载;

将所述主分区复制来创建多个从分区;

将所述从分区存储在所述节点集群上,其中,所述从分区使用改进的循环布局算法均匀地分配在所述节点集群之间,其中,存储在不同节点上的各从分区不包括与所述各从分区分别对应的主分区,关于各从分区和存储其的节点存在分集,并且故障节点的一个或多个主分区对应的一个或多个从分区可用于在所述计算机节点中的一个节点故障之后就升级到主用状态来保持工作负载平衡和数据的可用性;所述分集指其中对应于一个特定节点上的从分区的主分区均匀分配在其他节点之间的程度;

在所述数据分区的至少一个的重新分配期间来验证数据分区转移到和从第一节点转移的数量;还至少部分地基于维持在或低于预先设定的极限的转移的数量来将第一从分区存储在所述第一节点上,其中,在所述计算机节点中的至少一个故障或从所述集群分离之后,所述存储就成为重新分配的一部分;

还包括升级第二节点上的第二从分区,其中,所述第二从分区变成所述第二节点上的主分区,并且在所述计算机节点的至少一个节点故障或变成从所述集群分离之后,所述第二节点就成为仍然可用的幸存节点。

2.根据权利要求1所述的方法,还包括考虑存储所述第一从分区来确定关于第一从分区和第一节点是否存在分集。

3.根据权利要求2所述的方法,其中,确定关于所述第一从分区和所述第一节点是否存在分集,还包括来确定所述第一从分区对应的第一主分区以及已存储在所述第一节点上的第二从分区对应的第二主分区是否都已存储在第二节点上。

4.根据权利要求3所述的方法,其中,所述第一从分区是所述第一主分区的复制物。

5.根据权利要求1至4中任一项所述的方法,还包括确定关于第一从分区和当前存储所述第一从分区的第二节点是否存在分集,其中在所述第一节点上存储所述第一从分区是还基于关于所述第一从分区和所述第二节点不存在分集,并且在附加的计算机节点与所述集群相关联之后所述存储就成为重新分配的一部分。

6.一种非瞬时性的计算机可读存储介质,其特征在于,所述存储介质编码有适配来由处理器执行的指令以实现:

将数据库划分成多个主分区;

将所述主分区存储在多个与计算机网络相关联的节点集群上,其中,所述主分区使用循环布局算法均匀地分配在所述节点集群之间,并且所述主分区可用来服务系统工作负载;

将所述主分区复制来创建多个从分区;

将所述从分区存储在所述节点集群上,其中,所述从分区使用改进的循环布局算法均匀地分配在所述节点集群之间,其中,存储在不同节点上的各从分区不包括与所述各从分

区分别对应的主分区，关于各从分区和存储其的节点存在分集，并且故障节点的一个或多个主分区对应的一个或多个从分区可用于在所述计算机节点中的一个节点故障之后就升级到主用状态来保持工作负载平衡和数据的可用性；所述分集指其中对应于一个特定节点上的从分区的主分区均匀分配在其他节点之间的程度；

在所述数据分区的至少一个的重新分配期间来验证数据分区转移到和从第一节点转移的数量；

还至少部分地基于维持在或低于预先设定的极限的转移的数量来将第一从分区存储在所述第一节点上，其中，在所述计算机节点中的至少一个故障或从所述集群分离之后，所述存储就成为重新分配的一部分；

还包括升级第二节点上的第二从分区，其中，所述第二从分区变成所述第二节点上的主分区，并且在所述计算机节点的至少一个节点故障或变成从所述集群分离之后，所述第二节点就成为仍然可用的幸存节点。

7.根据权利要求6所述的计算机可读存储介质，其中考虑存储所述第一从分区，所述计算机可读指令还适配来执行实现确定关于第一从分区和第一节点是否存在分集。

8.根据权利要求7所述的计算机可读存储介质，其中确定关于所述第一从分区和所述第一节点是否存在分集还包括来确定所述第一从分区对应的第一主分区以及已存储在所述第一节点上的第二从分区对应的第二主分区是否都已存储在第二节点上。

9.根据权利要求6至8中任一项所述的计算机可读存储介质，其中，所述第一从分区是第一主分区的复制物。

10.根据权利要求9所述的计算机可读存储介质，其中，所述计算机可读指令还适配来执行实现确定关于第一从分区和当前存储所述第一从分区的第二节点是否存在分集，其中在所述第一节点上存储所述第一从分区是还基于关于所述第一从分区和所述第二节点不存在分集，并且在附加的计算机节点变成与所述集群相关联之后所述存储就成为重新分配的一部分。

11.一种用于在一群计算机节点之间分配多个数据分区的设备，其特征在于，所述设备包括：

分区器，用于将数据库划分成多个主分区；

复制器，用于将所述主分区复制来创建多个从分区；

初始分配器，用于将所述主分区和所述从分区存储在与计算机网络相关联的多个节点上，其中，所述主分区和所述从分区均匀地分配在所述节点之间，其中使用循环算法将所述主分区存储在所述节点上；使用改进的循环算法来将所述从分区存储在所述节点上，所述存储是所述数据分区的初始分配的一部分；所述主分区可用来服务系统工作负载，其中，存储在不同节点上的各从分区不包括与所述各从分区分别对应的主分区，关于各从分区和存储其的节点存在分集，并且故障节点的一个或多个主分区对应的一个或多个从分区可用于在所述计算机节点中的一个节点故障之后就升级到主用状态来保持工作负载平衡和数据的可用性；所述分集指其中对应于一个特定节点上的从分区的主分区均匀分配在其他节点之间的程度；

还包括故障转移分配器配置为在所述数据分区的至少一个分区的重新分配期间来验证数据分区转移到和从第一节点转移的数量，以及至少部分地基于维持在或低于预先设定

的极限的所述转移的数量来将第三从分区存储在所述第一节点上,其中在所述计算机节点的至少一个节点故障或变成从集群分离之后,所述存储就成为所述重新分配的一部分;

还包括插入分配器来配置为确定关于第二从分区和当前存储所述第二从分区的第二节点是否存在分集,至少部分地基于关于所述第二从分区和所述第二节点不存在分集的判断将所述第二从分区存储在第三节点上,并且在所述第三节点与所述集群相关联之后所述存储就成为重新分配的一部分。

12.根据权利要求11所述的设备,还包括至少部分地基于判断第一主分区以及已存储在所述第一节点上的第二从分区对应的第二主分区是否都已存储在第二节点上,来确定关于第一从分区和第一节点是否存在分集。

用于在集群重新配置后的工作负载平衡的复制型数据库分配

[0001] 相关申请案交叉申请

[0002] 本发明要求于2014年12月19日递交发明名称为“用于在集群重新配置后的工作负载平衡的复制型数据库分配”的美14/577,629号美国非临时专利申请案的在先申请优先权,该在先申请的内容以引入的方式并入本文。

技术领域

[0003] 本发明大体涉及分配式数据库,更具体地,涉及分配式关系数据库的复制型分区在网络计算机节点上的布局。

背景技术

[0004] 数据库用来存储信息和信息片之间的关系。一些数据库由单个网络或者多个互联的网络存储在多台计算机上的分配式文件系统中。通常,数据库划分成大小相等、相似或不相等的分区,并且信息数据的一个或多个分区存储在经由网络的系统与多台服务器相关联的存储器设备中。

[0005] 在一些现有的分配式文件系统中,如典型的Hadoop分配式文件系统(Hadoop distributed file system,简称HDFS)或典型的对等网络(peer-to-peer,简称P2P)分配式文件系统,各数据库文件划分成数据块,所述数据块分配到多个计算机节点上以便保持数据冗余并实现工作负载平衡。一些现有的分配式文件系统利用相当大量的节点,包括成千上万个节点,当附加的节点动态添加到和从所述集群去除时,所述节点随时间变化。这些系统中的计算机节点通常由商用硬件,如个人电脑(personal computer,简称PC)组成,并且可包括各用户系统。现有的方法能够实现显著的数据冗余和相当高的数据可用性来用于面向吞吐量的大规模松散耦合的分配式系统,如HDFS和P2P分配式文件系统。

[0006] 其他现有的分配式文件系统包括大规模并行处理(massively parallel processing,简称MPP)数据库管理系统,通过分配数据到具有相当高性能的一群或多群服务器如数据仓储设备中,以及通过在多台服务器上并行地运行各计算事务向外扩展。MPP数据库系统通常比HDFS或P2P分配式文件系统包括更少的节点,并且MPP数据库系统中的节点的数量相比于HDFS或P2P大体上相对更稳定。因此,有时MPP数据库的性能由并行处理计算机的集群中的更慢的各节点来控制。结果,工作负载平衡有时就变成同时实现高吞吐量和低延时的关键因素。在高性能的基于服务器的分配式文件系统中,假设各节点上的数据接收到计算资源的相似访问,那么数据在所述分配式服务器节点之间的均匀分配大体上就能够平衡工作负载。

[0007] 典型的MPP数据库集群使用“非共享”存储架构,其中存储硬件如磁盘驱动器,在所述集群的各节点之间不共享。在MPP数据库系统中,数据位置信息在运行时是不可用的。另外,MPP数据库集群越来越多地利用商用硬件资源,如PC。因此,如果集群中的单个服务器故障的话,存储在故障服务器上的数据就变得对该集群不可用。高可用性MPP数据库系统的设计要求所述集群能够容忍各服务器故障而提供持续的不丢失数据的服务。数据在多个服务

器中的复制已与故障转移过程相结合，使其在单个服务器出现故障的情况下仍然能够进行，以便达到容忍服务器故障的一般水平。

[0008] 此外，特别是关于节点故障以及冗余恢复期间，工作负载平衡没有高设计优先。例如，一致性哈希算法经常使用来在对等网络分配式文件系统中定位分配式和复制型数据。在一致性哈希算法中，哈希函数的输出范围会处理当作固定的圆形空间或环(例如，最大的哈希值绕接到最小的哈希值)。每个系统节点在这个空间内指定有一个值来代表环上的节点的位置。每个数据项在环内指定到其中一个节点，通过识别标识或与数据项相关联的密钥来在环上产生唯一位置，然后顺时针方向沿环“行走”或移动来找出位置比其他项更大的第一服务器。

[0009] 也就是说，每个节点变成对所述节点和其在环上的处理器节点之间的环的区域负责。然后，当节点故障时，对应的工作负载就转发到环上的下一个服务器。类似地，当一个新的节点附加到环上时，该节点仅共同承担环中相邻服务器的工作负载，结果导致工作负载不平衡。虽然在包括成千上万个节点的集群中可以合理地容忍这种不平衡的工作负载，但是相同水平的工作负载不平衡可在MPP数据库中对性能具有很大的负面影响。由于吞吐量和时延都具有相当的重要性，结果导致当一些现有的文件分配方法用于MPP数据库系统中时会有缺点。

发明内容

[0010] 根据一个一般方面，一种方法包括：将数据库划分成多个主分区；将所述主分区存储在与计算机网络相关联的多个节点上，其中，所述主分区大致均匀地分配在所述节点之间，并且所述主分区可用来服务系统工作负载；复制所述主分区来创建多个从分区；将所述从分区存储在所述节点上，其中，所述从分区大致均匀地分配在所述节点之间，各从分区而不是其对应的主分区存储在不同节点上，关于各从分区和存储其的节点存在分集，并且故障节点的一个或多个主分区对应的一个或多个从分区可用于在所述计算机节点中的一个故障之后就升级到主用状态来保持工作负载平衡和数据的可用性。

[0011] 根据另一个一般方面，一种用于将多个数据分区分配在一群计算机节点之间的计算机程序产品包括：非瞬时性的有形计算机可读存储介质，所述介质具有呈现在其中的计算机可读指令，所述计算机可读指令适配来执行实现将数据库划分成多个主分区，将所述主分区存储在与计算机网络相关联的多个节点上，其中，所述主分区大致均匀地分配在所述节点之间，并且所述主分区可用来服务于系统工作负载，复制所述主分区来创建多个从分区，以及将所述从分区存储在所述节点上，其中，所述从分区大致均匀地分配在所述节点之间，各从分区而不是其对应的主分区存储在不同节点上，关于各从分区和存储其的节点存在分集，并且故障节点的一个或多个主分区对应的一个或多个从分区可用于在所述计算机节点中的一个故障之后就升级到主用状态来保持工作负载平衡和数据的可用性。

[0012] 根据另一个一般方面，一种用于将多个数据分区分配在一群计算机节点之间的设备包括：分区器，用于将数据库划分成多个主分区；复制器，用于将所述主分区复制来创建多个从分区；初始分配器，用于将所述主分区和所述从分区存储在与计算机网络相关联的多个节点上，其中，所述主分区和所述从分区大致均匀地分配在所述节点之间，所述主分区可用来服务系统工作负载，各从分区而不是其对应的主分区存储在不同节点上，关于各从

分区和存储其的节点存在分集，并且故障节点的一个或多个主分区对应的一个或多个从分区可用于在所述计算机节点中的一个故障之后就升级到主用状态来保持工作负载平衡和数据的可用性。

[0013] 本发明的一个或多个实施例的细节在附图和下面的详细描述中阐明。本发明的其他特征、目的和优点从详细描述和附图并从权利要求书中将是明显的。

附图说明

[0014] 图1是描述一种用于在一群计算机节点之间分配数据分区的设备的示意图。

[0015] 图2是描述数据库数据分区在一群计算机节点之间的分区和分配的示意图。

[0016] 图3是表示主分区和从分区在一群计算机节点之间的分配的表格。

[0017] 图4是表示主分区和从分区在出现节点故障之后重新分配的表格。

[0018] 图5是表示一个集群中的各计算节点的等级的数组。

[0019] 图6是表示主分区和从分区在附加一个节点到所述集群中之后重新分配的表格。

[0020] 图7是示出一种在一群计算机节点之间初始化分配数据分区的方法的流程图。

[0021] 图8是示出一种在出现节点故障之后重新分配数据的方法的流程图。

[0022] 图9是示出一种在附加一个节点到所述集群中之后重新分配数据分区的方法的流程图。

具体实施方式

[0023] 图1中示出了根据本发明的一个实施例，来详细说明用于将与数据库相关联的信息数据分区、复制和分配在多台互联的服务器之间的装置10。装置10可在例如一个或多个关系数据库管理系统协调器节点、数据节点或任何其组合中应用。所述装置10包括由数据中继器26互联的输入/输出设备23、存储器22和处理器24。存储器22包括分区器12、复制器14、初始分配器16、故障转移分配器18和插入分配器20。

[0024] 使用装置10的结果可导致保持数据冗余的同时提高工作负载平衡，并提供关于现有文件分配系统高的数据可用性。特别地，使用装置10的结果可导致在出现单个计算机节点故障或多个节点故障之后，或在系统容量出现其他改变，如插入一个附加的节点之后，提高关于现有布局方法的工作负载平衡。结果，装置10可在大规模并行处理(massively parallel processing，简称MPP)数据库系统中提供显著优势。

[0025] 分区器12可将与数据库相关联的信息数据划分或分区成各种数据分组或大小相等、相似或不相等的分区。例如，分区器12可利用哈希函数来操作与数据库中的各数据项或信息段相关联的关键值来创建唯一标识符，该唯一标识符进一步将所述数据项或信息段与单个分区相关联。在一个优选实施例中，各分区的大小相当相似或几乎相等。

[0026] 复制器14可复制任何或所有的分区来创建所述一个或多个分区的备份副本或复制物。复制器14还可复制任何或所有的分区来创建所述一个或多个分区的附加备份复制物。初始分配器16在将要初始化分配式数据库时可影响所述数据分区在多个计算机节点上的布局。如果一个计算机节点故障，那么故障转移分配器18就可在幸存节点之间重新分配所述数据分区来实现工作负载平衡。类似地，如果一个节点附加到计算机的集群中，那么插入分配器20就可将数据分区从现有节点重新分配到新添加的节点来保持工作负载平衡。

[0027] 输入/输出设备23、存储器22和处理器24可为一般计算机设备，如个人电脑(personal computer，简称PC)、服务器或大型计算机的一部分。外接组件可包括存储在计算机可读介质上的编程代码，如源代码、目标代码或可执行代码，这些代码可加载到存储器22中并由处理器24来执行以便实现装置10的功能。因此，在各种实施例中，装置10的功能可在任何合适的处理器，如服务器、大型计算机、PC、PDA、网络服务器的集合或PC的集合等上来执行。此外，由于已研制出装置10的改进或改良的版本以便例如，来修订或添加一个模板或特定国家的信息，因此就可升级与处理器相关联的软件。

[0028] 在各种实施例中，装置10可连接到可包括能够链接基于计算机的系统的任意设备和系统的可行组合的通信网络，如互联网；企业内部网或外部网；局域网(local area network，简称LAN)；广域网(wide area network，简称WAN)；直接电缆连接；专用网络；公用网络；基于以太网的系统；令牌环网；增值网络；基于电话的系统，例如，包括T1或E1设备；异步传输模式(Asynchronous Transfer Mode，ATM)网络；有线系统；无线系统；光纤系统；任意数量的分配式处理网络或系统等的组合。

[0029] 地址数据收集系统10可经由本地数据链26耦合到通信网络，该系统在多种实施例中，可包含设备的任意组合，以及包含用于耦合基于处理器的系统的相关软件或固件，如调制解调器、网络接口卡、串行总线、并行总线、LAN或WAN接口、无线或光纤接口等，连同包含如同设计可能需要或要求的任意相关的传输协议。

[0030] 本发明的一个实施例可将信息通信给用户并请求用户经由，例如，交互的、菜单驱动的、基于视觉显示的用户界面或图形用户界面(graphical user interface，简称GUI)输出信息。所述用户界面可在，例如，个人电脑(personal computer，简称PC)上用鼠标和键盘来执行，通过键盘和鼠标用户可利用GUI的直接操纵来交互地输入信息。直接操纵可包括使用如鼠标或手写笔的定点设备，来从各种可选的字段，包括可选菜单、下拉菜单、选项卡、按钮、项目符号、复选框、文本框等中进行选择。尽管如此，本发明的各种实施例可包含任意数量的使用或不使用鼠标或按钮或键的附加功能的用户接口方案，包括，例如轨迹球、触摸屏或声控系统来代替这个接口方案。

[0031] 参照图2，数据库表28可由图1中的分区器12划分成包括来自于数据库表28的字段的多个行30的各种分区28。在本例中，分割器12可使用一个哈希函数在与每个行30相关联的识别号(identification number，简称ID)34上操作，以创建一个唯一标识符，唯一标识符进一步将数据库表28的行30的每个与分区28的其中一个在哈希桶中相关联。数据库表28的大小大致相等的记录可大致均匀地分配在各分区28之间来创建大小大致相等的分区28。所述分区28可随后布局，或分配在与数据库相关联的计算机集群的多个节点28之间。

[0032] 大体而言，当前正在任何给定时间服务于系统工作负载，或当前可用来服务于系统工作负载的存储分区，可为指定的“主”分区。当前不正在服务于工作负载，或当前不可用来服务于工作负载的存储备份复制物，可为指定的“从”分区。在任意给定的时间，任何单个主分区可对应多个从分区或复制物。

[0033] 大体而言，各分区在以下三种情况之一可分配到各计算机节点：1)一经数据库初始化时；2)一个或多个节点出现故障之后；或3)一经一个附加节点或各节点插入到所述集群中时。分区布局或分配算法，以在各节点之间平衡工作负载并提供高水平的数据可用性的方式，在各节点之间分配各分区。为此，大体而言，所述主分区和从分区都大致均匀地分

配在各计算机节点之间。主分区的均匀分配大体促进了工作负载平衡，而所述从分区的均匀分配在一个或多个节点故障之后，促进了故障转移情况期间的工作负载平衡。

[0034] 为了推进工作负载平衡并提供k-安全数据可用性(即基于对应于各主分区的K从分区，容忍k同步节点故障的能力)，所述布局算法大体上遵从下述四项准则：

[0035] ●任意单个分区的主副本和各从复制物位于不同节点上；

[0036] ●各主分区基本均匀地分配在各节点之间；

[0037] ●各从分区基本均匀地分配在各节点之间；

[0038] ●各从分区分配来最大化分集。

[0039] 第一准则大体是在一个或多个节点出现故障时，避免或最小化数据丢失。第二和第三准则大体上促进工作负载平衡。第四准则在一个或多个节点出现故障之后，推进故障转移情况期间的工作负载平衡。这里，分集指的是其中对应于一个特定节点上的从分区的主分区均匀分配在其他节点之间的程度。虽然布局算法试图尽可能遵从所有四个准则，但是严格遵守第二、第三和第四准则不是绝对的要求。

[0040] 因此，数据库初始化期间，一旦来自数据库中的信息已分组成各分区，初始分配器16就可根据本发明中所描述的分区布局算法将每个分区储存或布局在与数据库相关联的多个计算机节点之一中。需要访问数据库记录的后续查询可转发到存储包含该记录的分区的节点。

[0041] 参照图3，如数据库初始化期间分区布局的一个具体的例子，表40示出了主分区42和从分区44在与数据库相关联的一群服务器的节点46之间的布局。来自数据库表中的得分分为编号为1到32的三十二个分区。每个分区的主副本和从复制物分配在指定为N1到N8的一群八计算节点之间。为了实现工作负载平衡，每个节点46上分配有四个主分区42和四个从分区44。

[0042] 因此，例如，主分区42使用循环布局算法来均匀地分配在各节点之间，节点N1上分配编号为1、9、17和25的四个主分区；节点N2上分配编号为2、10、18和26的四个主分区；节点N3上分配编号为3、11、19和27的四个主分区；节点N4上分配编号为4、12、20和28的四个主分区；节点N5上分配编号为5、13、21和29的四个主分区；节点N6上分配编号为6、14、22和30的四个主分区；节点N7上分配编号为7、15、23和31的四个主分区；节点N8上分配编号为8、16、24和32的四个主分区。

[0043] 然后，从分区44使用改进的循环布局算法均匀地分配在各节点46之间。每个从分区44分配到除了其中存储对应的主分区的节点之外的一个节点。例如，对应于存储在节点N1上的主分区1的从分区存储在节点N5上。此外，每个对应于存储在单个节点上的主分区42的从分区44优选存储在独立的节点40上。例如，对应于存储在节点N1上的主分区1、9、17和25的从分区存储在四个独立的节点上：N5、N6、N7和N8。类似地，从分区44优选布局使得每个对应于存储在单个节点上的从分区44的主分区42存储在独立的节点46上。例如，对应于存储在节点N1上的从分区5、12、19和26的主分区42存储在四个独立的节点：N5、N4、N3和N2上。

[0044] 这种布局算法大体上确保每对对应的主分区和从分区42，44中的至少一对在一个节点出现故障之后仍然保持在幸存节点上。如果附加的分区复制物按照关于对应主分区和各对应从分区的所述相同的准则来存储，那么系统就可容忍尽可能多的从分区的存储复制物的同步节点故障的数量。因此，如果存储两个从分区复制物，那么系统就能够容忍两个同

步节点故障而不丢失数据。也就是说，k从分区复制物的存储提供了k-安全数据可用性。

[0045] 大体而言，当一个节点离开或加入分配式文件系统集群时，需要在各节点之间重新分配或转移数据分区，以便保持或恢复适当的工作负载平衡。分区分配在完成重新分配后来继续满足上述四个准则是非常重要的。

[0046] 由于单个分区可包含例如从多个千兆字节到多个万亿字节的范围的大量数据，所述重新分配过程可潜在地消耗非常重要的能够暂时影响查询性能的网络和存储带宽。因此，最大限度地在重新分配中减少分区的数量同工作负载平衡一样重要。

[0047] 当一个服务器节点故障或从网络断开连接或解离，将重新分配以便恢复数据冗余的分区的最小数量大体上等于存储在故障或断开连接的节点上的分区的总数。存储在故障或断开连接的节点上的主分区不能再服务于工作负载的请求。由于这个原因，与存储在故障的或断开连接的节点上的主分区相对应的存储在其他当前运行的节点上的从分区就升级到主用状态来应对节点故障的检测。这种情况就称为“故障转移”。

[0048] 对应于故障的节点的主分区的后续请求被转发到存储对应的从分区的节点。创建已被提升到主用状态的从分区的新复制物，以便在故障转移后恢复数据冗余。工作负载在故障转移情况下仍然保持适当的平衡，因为受影响的从分区在重新分配期间均匀地分配于剩余的节点上。

[0049] 本发明中所描述的布局算法在一个节点出现故障后，当从分区升级到主用状态并布局在幸存节点中时，可大体上平衡故障转移情况下的工作负载。重新分配的工作负载应当由所有的节点均匀地分担。本发明中所描述的布局算法最小化重新分配过程中的分区数，并且在重新分配期间以及之后实现工作负载平衡的同时保持数据冗余的水平。

[0050] 参照图4，如分区布局在一个节点故障或离开集群之后的重新分配过程中的一个具体例子，表50示出了主分区52和从分区54在与数据库相关联的一群服务器的节点56之间的布局。图3的配置中的所述三十二个分区的主副本和从复制物在图3的节点N8故障或离开集群之后，已重新分配在剩余七个指定为N1到N7的计算机节点56之间。为了实现工作负载平衡，每个节点56分配有四个或五个主分区52和四个或五个从分区54。

[0051] 由于存储在图3的节点N8上的编号为8、16、24和32的主分区在出现故障或从节点N8上的集群网络断开后不再可用，存储在节点N4、N5、N6和N7上的对应从分区就升级到主用状态。其结果是，如图3的配置中的每个节点N1、N2和N3在节点N8故障或断开连接之前继续存储相同的四个主分区，而在重新分配后每个节点N4、N5、N6和N7则存储五个主分区：其中，节点N1继续存储编号为1、9、17和25的四个主分区；节点N2继续存储编号为2、10、18和26的四个主分区；节点N3继续存储编号为3、11、19和27的四个主分区；节点N4上分配编号为4、8、12、20和28的五个主分区；节点N5上分配编号为5、13、16、21和29的五个主分区；节点N6上分配编号为6、14、22、24和30的五个主分区；节点N7上分配编号为7、15、23、31和32的五个主分区。

[0052] 节点N8不再包含在所述集群中并且当前不存储任何分区，而直接定向编号为8、16、24和32的分区查询则被重定向或从节点N8分别故障转移到节点N4、N5、N6和N7。只要编号为8、16、24和32的四个分区已升级到主用状态，那么所有的数据库中的数据对于数据库用户仍然是可用的。

[0053] 在四个故障转移分区自从从用状态升级到主用状态之后的时间点，没有分区重新

分配的介入。然而，新指定的编号为8、16、24和32的主分区不会一开始就具有对应的从分区。此外，先前存储在节点N8上的编号为4、11、18和25的从分区不再可用。其结果是，八个从分区(编号为4、8、11、16、18、24、25和32)需复制并重新分配到其他节点以便保持数据冗余。这是分区重新分配的最小数量，因为它是节点故障之前先前存储在节点N8上的分区的数量。

[0054] 再次，从分区使用改进的循环布局算法均匀地分配在幸存节点之间。每个从分区优先分配到除了其中存储对应的主分区的节点之外的一个节点。例如，对应于现有储在节点N4上的主分区8的从分区存储在节点N6上。编号为8的从分区不会优选布局在节点N4上，因为编号为8的对应主分区位于节点N4上。

[0055] 此外，每个对应于存储在单个节点上的主分区的从分区优选存储在独立的节点上。例如，对应于存储在节点N4上的主分区4、8、12、20和28的从分区分别存储在五个独立的节点:N5、N6、N1、N2和N3上。类似地，从分区优选地布局使得每个对应于存储在单个节点上的从分区的主分区存储在独立的节点上。例如，对应于从分区3、10、16和17的各主分区分别存储在四个独立的节点:N3、N2、N5和N1上。

[0056] 所述重新分配主要是由有关布局算法的四个准则的强制执行来驱动的，即，最大限度地提高分集或一个特定节点上的从分区对应的主分区在其他节点之间均匀分配的程度。例如，在改进的循环方法中，如果第一节点遇到关于该方法存在分集，则可选择来用于正被考虑的从分区的布局。

[0057] 因此，例如，如图4中所示，在节点N8故障之后的重新分配期间，由于关于节点N6上的从分区存在分集，因此新创建的编号为8的从分区布局在节点N6上。该从分区优选地不布局在节点N1上，因为关于节点N1上的该从分区不存在分集，由于位于节点N1上的编号为12的从分区对应节点N4上的主分区，其中编号为8的主分区布局在N4上。

[0058] 类似地，编号为8的从分区优选地不布局在节点N2、N3或N5上，因为关于该从分区不存在分集，由于分别位于节点N2、N3和N5上的编号为20、28和4的从分区对应节点N4上的各主分区。这种方法是一种贪婪算法，因为单个从分区的布局完全基于关于该从分区的分集。因此，该结果可能不是全局最优的。

[0059] 在重新分配期间，关于分区布局算法可基于在重新分配期间转移到或从单个节点转移的分区的数量施加一个附加的准则，该数量称为节点的等级。因此，为了在重新分配期间平衡工作负载，一个数组可以创建来表示与数据库集群中的每个节点相关联的当前等级值。该数组可以改良来反映各分区到和从集群中的每个节点的每个移动或转移。因此，等级数组记录了分区从每个相应的节点转移的数量。

[0060] 如一个具体的例子，在图3的配置和图4的配置中的数据分区的重新分配期间，八个从分区被复制并分配在七个幸存的节点之间。每个分区布局会影响两个节点:即一个发送该分区的节点和另一接收该分区的节点。因此，在本例中共发送了八个分区并接收了八个分区。从单个节点的角度来看，共有十六个发送和接收的操作，其中每个操作导致与受影响的节点相关联的等级值的增量。

[0061] 为了在重新分配期间平衡工作负载，一个有资格接收从分区的节点的最大允许等级优选地等于对分区转移操作的总数量除以幸存节点的数量的值加1的和取整(四舍五入，取最近的整数值)：

[0062] $$\mathrm{degree}_{\max} = \mathrm{trunc}\left(\frac{\textit{total no.of partition transfer transactions}}{\textit{no. of surviving nodes}} + 1\right)$$

[0063] 在本例中，该最大等级是3，如下所示：

[0064] $$\mathrm{degree}_{\max} = \mathrm{trunc}\left(\frac{(8+8)}{7} + 1\right) = 3$$

[0065] 当选择一个节点用于布局从分区时，优选地仅考虑那些当前还未达到最大等级值的节点。

[0066] 现参照图5，在本例中示出了随着重新分配布局过程中的时间62推移(纵向前进)的一个示例性等级数组60。一开始，所有节点的等级为零。然后，随着重新分配的发生，每个受影响的节点对应的等级数组元素在每次节点被选择来布局从分区时就递增，因为该元素对应于复制的分区所位于的节点。

[0067] 因此，例如，在图5中所示的在第一个重新分配布局的步骤64中，当节点N4中编号为8的主分区被复制并且编号为8的从分区被布局在节点N6中时，对应于N4和N6的元素都递增为一的值。随后，在布局步骤6中，当确定从分区11的布局时，尽管分区将有资格在节点N4中布局，对应的等级值却已经达到用于重新分配的最大值三。其结果是，N5替代来选择用于从分区11的布局。

[0068] 当一个附加的服务器节点连接到网络上新与集群相关联或者集群的容量另有变化时，数据分区一般需要重新分配，以便在计算机节点的当前集群之间恢复工作负载平衡。分区重新分配以便恢复数据冗余的最小数量可基于主从分区的总数量以及集群中的节点数来计算。

[0069] 例如，主分区将转移到新添加的节点上的最小数量可计算为对每个节点上的主分区的平均数量取整，并且相同数量的从分区也可转移到所述新添加的节点上。可替代地，分区将转移到新添加的节点上的最小总数量可计算为对每个节点上的分区的总数量取整，并且这些可基本均匀地划分在主从分区之间。

[0070] 将要移动到新插入的节点上的特定主分区和从分区可优选地使用每个分区的分集来在附加节点插入之前的原始分配中确定。关于主分区的分集被认为与对应的从分区的分集相同，或等价地，对应的从分区的分集也用于对应主分区。因此，所公开的方法，或布局算法，建立一个表示各分区的分集的数组。缺乏分集或分集等于零的分区被给予优先权来用于转移到新添加的节点，以便增加这些分区的分集。

[0071] 参照图6，如分区布局在容量改变后的重新分配期间的一个具体例子导致，当一个附加节点插入到集群中时，表70示出主分区72和从分区74在与数据库相关联的一群服务器的节点76之间的布局。图3中所示的配置中的三十二个分区的主副本和从复制物在节点N9加至集群之后已重新分配在九个当前指定为N1到N9的计算机节点76之间。为了实现工作负载平衡，每个节点分配有三个或四个主分区以及三个或四个从分区。分区的最小数量为六个，包括三个主分区和三个从分区，布局在节点N9上。

[0072] 最小数量的三个主分区转移到新添加的节点N9，并且所述主分区使用循环布局算法基本均匀地分配在所有当前的集群节点之间。编号最后或最高的主分区来自图3中示出的配置中的前三个节点中的每个，如来自节点N1编号为25的主分区，来自点N2编号为26的主分区，以及来自节点N3编号为27的主分区，转移到新添加的节点N9来在新添加的服务器

上构成主分区的最小数量。另外，编号最后或最高的从分区来自图3中示出的配置中的后续三个节点中的每个，如来自节点N4编号为29的从分区，来自节点N5编号为30的从分区，来自节点N6编号为31的从分区，转移到新添加的节点N9来在新添加的服务器上构成从分区的最小数量。

[0073] 其结果是，在节点N9加至集群之前，每个如图3的配置中所示的节点N4、N5、N6和N7继续存储四个相同主分区，而在重新分配后，各节点N1、N2、N3和N8则存储三个主分区：节点N1上分配编号为1、9和17的三个主分区；节点N2上分配编号为2、10和18的三个主分区；节点N3上分配编号为3、11和19的三个主分区；节点N4继续存储编号为4、12、20和28的四个主分区；节点N5继续存储编号为5、13、21和29的四个主分区；节点N6继续存储编号为6、14、22和30的四个主分区；节点N7继续存储编号为7、15、23和31的四个主分区；节点N8继续存储编号为8、16、24和32的四个主分区。新添加的节点N9分配编号为25、26和27的三个主分区。

[0074] 每种布局遵从关于本发明中所描述的布局算法的适用准则。因此，每个分配到新添加的节点N9上的编号为25、26和27的主分区与每个分配到新添加的节点N9上的编号为39、30和31的从分区不同。此外，每个节点N1至N9存储三个或四个主分区，连同三个或四个从分区。此外，与存储在新添加的节点N9上的编号为29、30和31的从分区相对应的每个主分区分别存储在单独的节点N5、N6和N7上，其中各主分区保持关于各从分区的分集。

[0075] 现参照图7，示出了一个可由图1的装置10在初始化数据库时使用来实现在本发明中所描述的分区布局算法的流程图。该过程在框70处开始“启动分区布局”。在框72中，节点号变量(例如，“N”)设定为等于与数据库相关联的一群服务器中的节点的编号，分区号变量(例如，“M”)设定为等于自数据库中的记录创建的数据分区的编号，以及主/从变量(例如，“IsPrimary＝true”)设定为指示当前所考虑的分区是一个主分区。在框74中，两种计数变量，第一种指示当前所考虑的节点编号(例如，“i”)并且第二种指示当前所考虑的分区编号(例如，“j”)，被设定为等于值“1”。

[0076] 在框76中，判定关于当前所考虑的分区是否是一个主分区。如果是主分区，则在框78中倾斜变量设定为等于值“0”。否则，如果当前所考虑的分区不是主分区而是一个从分区，那么在框80中的倾斜变量设定为等于值“3”。在这两种情况下，程序从框78或80进行到框82，在框82中来进行判定关于当前所考虑的分区数量是否小于或等于预先在框72中设定的分区的数量。如果不是主分区(即，当前所考虑的分区数量大于预先在框72中设定的分区的数量)，那么在框84中的主/从变量则设定为指示当前所考虑的分区不是一个主分区而是一个从分区，并且程序如上所述继续在框74处进行。

[0077] 在另一方面，如果在框82中判定目前所考虑的分区数量小于或等于预先在框72中设定的分区的数量，那么在块86中则进一步判定关于当前考虑的分区是否为一个主分区。如果不是主分区，而是相反目前所考虑的分区是一个从分区，则在框88中进行分析，以判定包含相同数据的对应主分区是否存储在当前所考虑的节点上。如果不是这种情况，而是所述对应主分区存储在另一个节点上，则在框90中进行关于存储在当前所考虑的节点上的其他从分区的有关分集的分析。

[0078] 分集指的是其中对应于一个特定节点上的从分区的主分区均匀分配在其他节点之间的程度。完全或最大分集是指与一个特定节点上的从分区相对应的每个主分区存储在集群中的独立的单个节点上。这就是说，对应于存储在特定节点上的从分区的两个主分区

存储在集群中的任何单个节点上。

[0079] 在框90处执行的分析，可例如，在没有与存储在所考虑的节点上的任意从节点相对应的主节点，同被考虑用于布局的从分区的主分区一样也存储在所述同一节点上的情况下，来判定分集的存在(“分集＝1”)。否则，例如，分集可判定为不存在(“分集＝0”)。这种方法是一种贪婪算法，因为它仅基于特定的从分区的分集来判定特定从分区的布局。因此，该结果可能不是全局最优的。在一个可选实施方案中，在框90处执行的分析可，例如，进一步判定当没有对应于在所考虑的节点上的从分区的两个主分区存储在集群中的任意单个节点上时，存在分集。

[0080] 如果分集判定为不存在框90中，或者，在另一方面，如果要被存储在同一节点上的对应主分区确定在框88中，则在框92中的节点号变量被增量值增加一，并且程序继续到框88。否则，如果分集判定为存在于框90中，或者如果在框82中的判定为当前所考虑的分区数大于预先前在框72中设定的分区的数量，那么所考虑的主分区或从分区则在框94中布局在节点上，该节点具有来源于对存储在节点数变量中的值加上存储在倾斜变量中的值的和相比于存储在节点数变量中的值执行模运算的数字：

[0081] 布局＝(所考虑的节点数+倾斜变量值)mod(集群中的节点数)

[0082] ＝(i+倾斜)mod N

[0083] 现参照图8，示出了一个可由图1的装置10在数据库中的节点故障或者从网络断开连接之后使用的以便实现本发明中所描述的分区布局算法的流程图。在节点出现故障或断开连接之后来执行分区布局算法之前，与先前存储在故障或断开连接的节点上的主分区对应的从分区应当在存储那些从分区的同一节点上升级到主用状态。下面的过程然后相对于每个新升级的主分区以及关于每个先前存储在故障或断开连接的节点上的从分区在复制这些分区时来执行。

[0084] 从分区布局程序开始于框100，“启动故障转移重新分配。”在框102中，两种计数变量，第一种指示当前所考虑的节点编号(例如，“i”)以及第二种指示当前所考虑的分区编号(例如：“j”)，被设定为等于值“1”。

[0085] 在框104中执行分析，以判定同当前所考虑的从节点包含相同数据的对应主分区是否存储在当前所考虑的节点上。如果不是这种情况，而是所述对应主分区存储在另一节点上，则在框106中进行关于存储在当前所考虑的节点上的其他从分区的有关分集的分析，所述分集如上所述与框90相联系。上述分集分析可例如，在没有与存储在所考虑的节点上的任意从节点相对应的主节点，同被考虑用于布局的从分区的主分区一样也存储在所述同一节点上的情况下，来判定分集的存在(“分集＝1”)。

[0086] 如果在框106中的分析判定关于当前所考虑的从分区存在分集，则在框108中执行对有关与当前所考虑的节点相关联的当前等级的分析，如上所述。所述等级分析可，例如，判定该节点有资格用于分区布局，如果其等级的当前值小于最大可接受值的话。如果在框108中的分析判定所考虑的节点的等级还未达到可接受的最大等级，则所考虑的从分区在框110中布局在所考虑的节点上。在块112中，节点数和分区数计数变量由增量值1在框114中增加，并且程序在框104中继续下一个复制分区。

[0087] 否则，如果在框104中的分析判定对应的主分区存储在同一节点上，或者如果判定分集不存在于框106中，或者如果判定所考虑的节点的等级在框108中已达到最大可接受

值，那么节点数计数变量(i)由增量值1在框112中增加，并且程序通过考虑从分区在下一个节点上的布局在框104中继续。

[0088] 现参照图9，示出了一个可由图1的装置10在一个附加节点已插入到数据库集群中之后使用以便实现本发明中所描述的分区布局算法的流程图。下面的程序是关于要转移到新添加的节点的最小数量的主分区以及关于要转移到新添加的节点的最小数量的从分区来执行的。该过程开始于框120，“启动插入重新分配”。在框122中，从分区“j”基于关于从分区在初始位置或当前位置缺乏分集而选择来将要移动到新添加的节点“i”。

[0089] 例如，将要移动到新插入的节点上的特定主分区和从分区可优选地使用每个节点的分集来在附加节点插入之前确定。关于主分区的分集被认为与对应的从分区的分集相同，或等价地，对应的从分区的分集也用于对应主分区。因此，例如，一个阵列可创建来表示每个节点的分集，并且可给予节点缺乏分集或分集等于零的那些分区优先级，用于转移到新添加的节点，以便增加那些节点的分集。可替代地，循环算法(图9中未示出)可用来判定主从分区在新添加的节点上的布局。

[0090] 在框124中，如果一个主分区正考虑转移到新添加的节点，则有关同当前所考虑的从节点包含相同数据的对应从分区是否存储在新添加的节点上的判定就进行。相应地，如果一个从分区正考虑转移到新添加的节点，则有关同当前所考虑的从节点包含相同数据的对应主分区是否存储在新添加的节点上的判定就进行。

[0091] 如果不是这种情况，而是所述对应主分区或从分区存储在另一节点上，则在框126中进行关于所选分区和存储在新添加的节点上的其他分区是否存在分集的分析。如果在框126中的分析判定关于当前所考虑的分区存在分集，则在框128中执行对有关与新添加的节点相关联的当前等级的分析，如上所述。所述等级分析可，例如，判定所述新添加的节点有资格用于分区布局，如果其等级的当前值小于最大可接受值的话。

[0092] 如果在框128中的分析判定新添加的节点的等级还未达到可接受的最大等级，则在框130中所考虑的分区布局在新添加的节点上，并且该程序通过选择另一被考虑来转移到框122中新添加的节点上的主分区或从分区而继续。否则，如果在框124中的分析判定对应的主分区或从分区存储在新添加的节点上，或者如果判定在框126中不存在分集，或者如果判定新添加的节点的等级在框128中还未达到最大可接受值，那么程序就通过选择另一被考虑来转移到框122中新添加的节点上的主分区或从分区而继续。

[0093] 本发明的方面是参照流程图或框图来描述的，其中，每个框或任意框的组合可通过计算机程序指令来实施。所述指令可提供给通用计算机，专用计算机或其他可编程数据处理设备的处理器，以实现机器或制品的生产，并且当由处理器执行时，所述指令创建用于实现在每个框或多个框的组合中指定的所述功能、动作或事件的方法。

[0094] 在这方面，流程图或框图中的每个框可对应于包括用于实现指定的逻辑功能(多个)的一个或多个可执行指令的模块、段或代码的一部分。还应当指出的是，在一些替代实施方式中，与任意框相关联的功能可能会脱离图中指出的顺序发生。例如，连续示出的两个框实际上可大致同时地执行，或者框有时可以相反的顺序来执行。

[0095] 本领域的普通技术人员的人员将会理解，本发明的各方面可以体现为设备、系统、方法或计算机程序产品。因此，本发明的各方面，此处通常指电路、模块、部件或系统，可以体现在硬件、软件(包括固件、驻留软件、微代码等)，或在软件和硬件的任意组合，包括体现

在具有计算机可读程序代码的计算机可读介质中的计算机程序产品。

[0096] 在这方面，可以利用一个或多个计算机可读介质的任何组合，包括但不限于：电子，磁，光，电磁，红外，或半导体系统，装置或设备，或上述的任意合适组合。计算机可读存储介质的更具体的例子包括以下的非详尽列表：便携式计算机磁盘、硬盘、随机存取存储器(random access memory，简称RAM)、只读存储器(read-only memory，简称ROM)、可擦除可编程只读存储器(erasable programmable read-only memory，简称EPROM)、闪存、便携式光盘只读只读存储器(portable compact disc read-only memory，简称CD-ROM)、光存储设备，网络连接存储(network-attached storage，简称NAS)、存储区域网络(storage area network，简称SAN)、磁带，或上述的任意合适组合。在本发明的上下文中，计算机可读存储介质可包括能够包含或存储被数据处理系统、装置或设备使用或与其相关的程序指令的任何有形介质。

[0097] 可以用一个或多个编程语言的任意组合来写用于实现与本发明的各方面的操作的计算机程序代码，包括如Java、Smalltalk或C++等面向对象的程序设计语言，以及如“C”、FORTRAN、COBOL或Pascal等传统过程化编程语言。程序代码可完全地在个人计算机上作为一个独立的软件包执行，部分地在客户端计算机上部分地在远程服务器计算机上执行，以及完全地在远程服务器或计算机或分配式计算机节点的集群上执行。大体而言，远程计算机、服务器或分配式计算机节点的群集可以通过任何类型的网络连接到个人(用户)计算机，包括一个局域网(local area network，简称LAN)、广域网(wide area network，简称WAN)、互联网接入点，或上述的任意组合。

[0098] 在一个示例性实施例中，使用一种设备来在一群计算机节点之间分配多个数据分区。所述设备包括：划分模块，用于将数据库划分成多个主分区；主存储模块，用于将所述主分区存储在与计算机网络相关联的多个节点上，其中，所述主分区大致均匀地分配在所述节点之间，并且所述主分区可用来服务系统工作负载；复制模块，用于复制所述主分区来创建多个从分区；从存储模块，用于将所述从分区存储在所述节点上，其中，所述从分区大致均匀地分配在所述节点之间，各从分区而不是其对应的主分区存储在不同节点上，关于各从分区和存储其的节点存在分集，并且故障节点的一个或多个主分区对应的一个或多个从分区可用于在所述计算机节点中的一个节点故障之后就升级到主用状态来保持工作负载平衡和数据的可用性。在一些实施例中，所述设备可包括其他或附加的模块用于完成所述实施例中所描述的步骤的任何一个步骤或步骤的组合。

[0099] 可以理解的是可以进行各种修改。例如，如果所公开的技术的步骤以不同的顺序进行，和/或如果所公开的系统中的组件以不同的方式合并，和/或由其他组件代替或补充，有用的结果仍可能会实现。因此，其他实现方式在以下权利要求的保护范围之内。

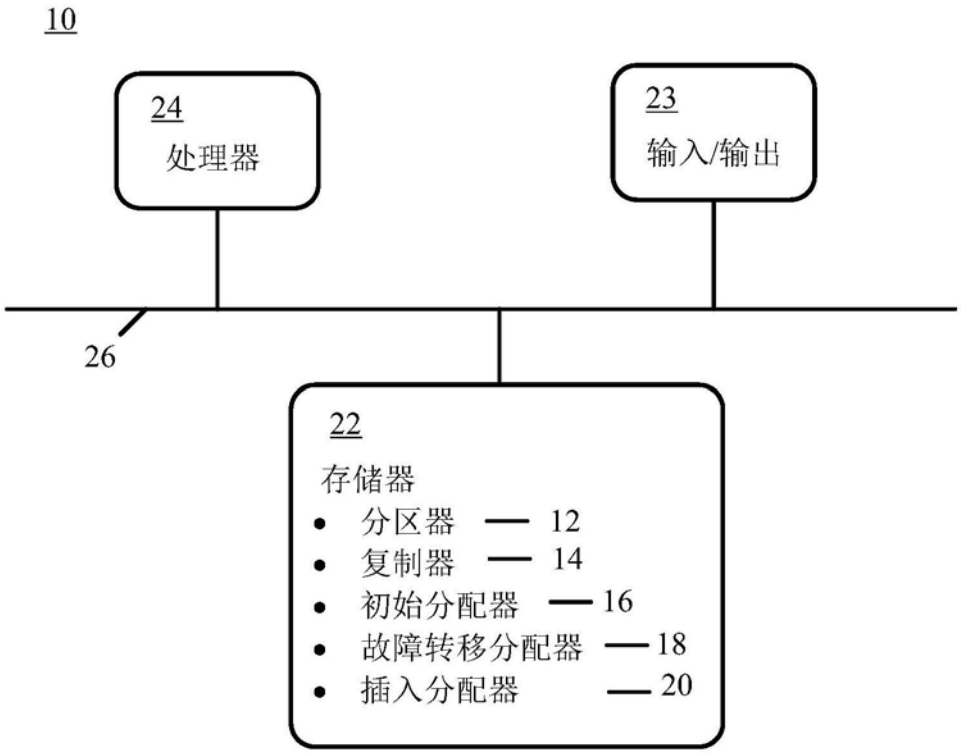

10
24
处理器
23
输入/输出
26
22
存储器
分区器 12
复制器 14
初始分配器 16
故障转移分配器 18
插入分配器 20

图1

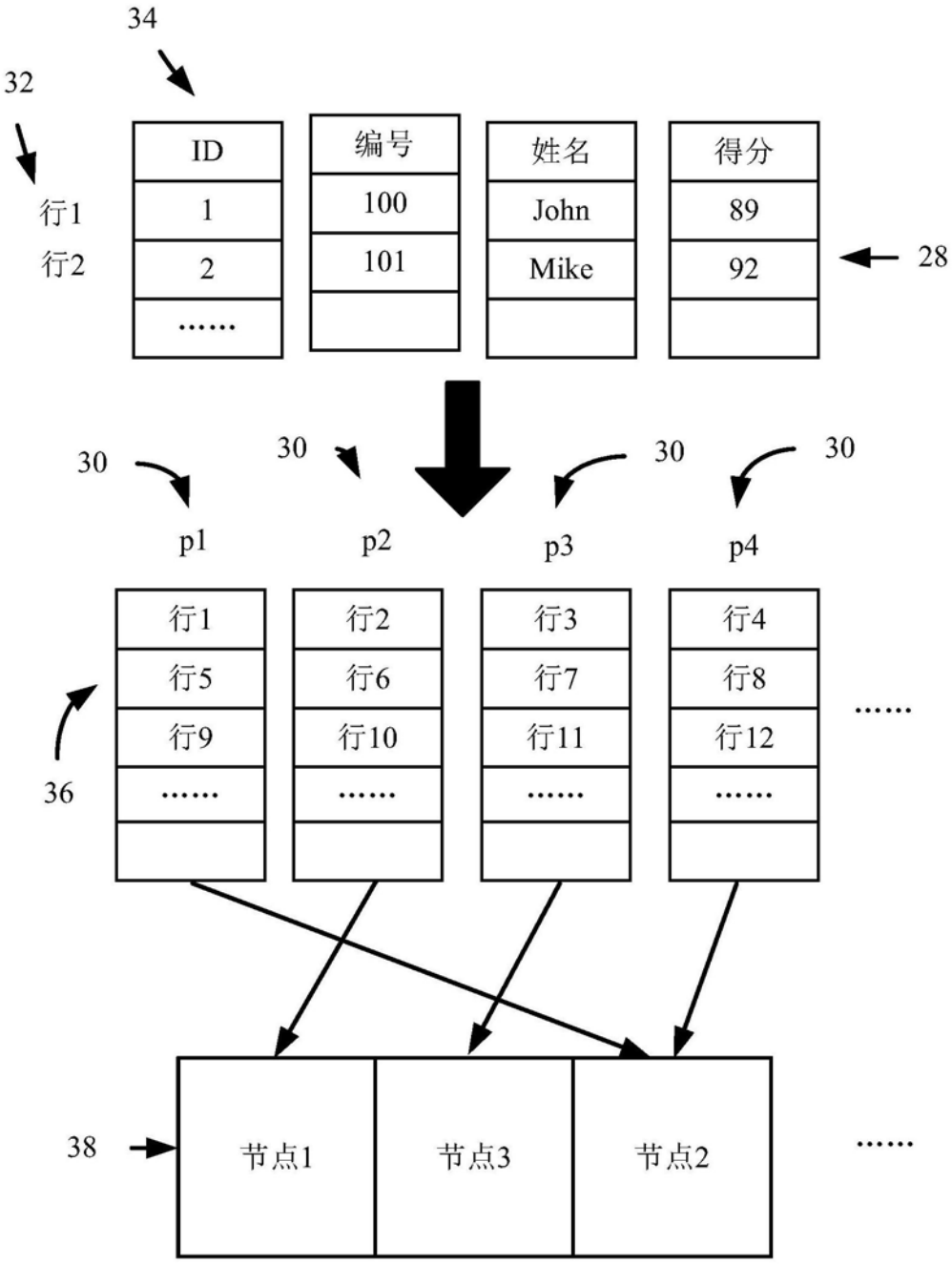

34
32
ID
编号
姓名
得分
行1
1
100
John
89
行2
2
101
Mike
92
28
30
30
30
30
p1
p2
p3
p4
行1
行2
行3
行4
行5
行6
行7
行8
行9
行10
行11
行12
36
38
节点1
节点3
节点2

图2

46 42 44

40

节点	主				从		
N1	1	9	17	25	5	12	26
N2	2	10	18	26	6	13	27
N3	3	11	19	27	7	14	28
N4	4	12	20	28	8	15	29
N5	5	13	21	29	1	16	30
N6	6	14	22	30	2	9	31
N7	7	15	23	31	3	10	32
N8	8	16	24	32	4	11	25

图3

56 52 54

50

节点	主					从				
N1	1	9	17	25		5	12	19	24	26
N2	2	10	18	26		6	13	20	25	27
N3	3	11	19	27		7	14	18	21	28
N4	4	8	12	20	28	15	22	29	32	
N5	5	13	16	21	29	1	4	11	23	30
N6	6	14	22	30	24	2	8	9	31	
N7	7	15	23	32	31	3	10	16	17	

图4

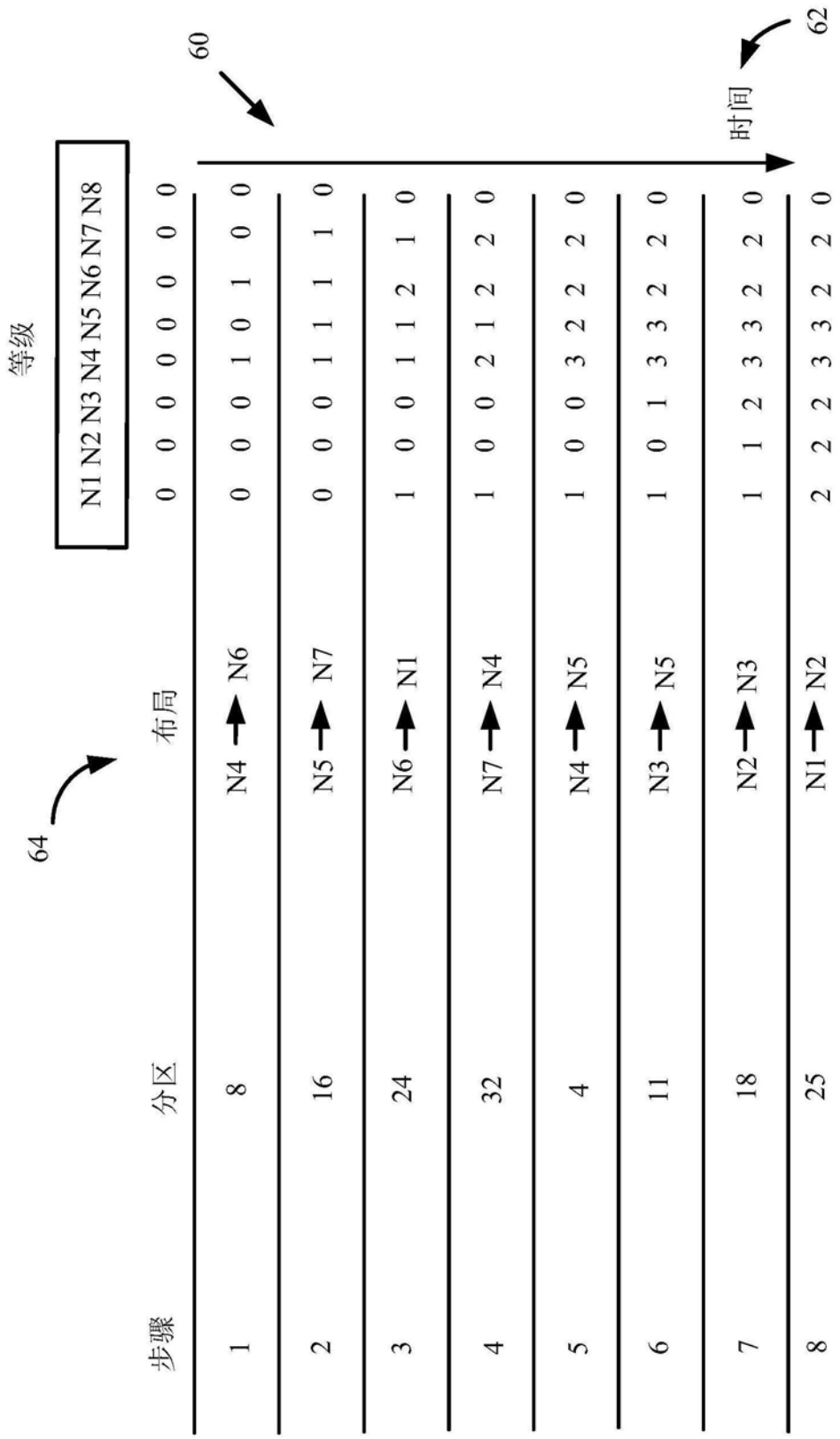

60
62
64
时间
等级
N1 N2 N3 N4 N5 N6 N7 N8
步骤
分区
布局
1 8 N4 → N6 0 0 0 0 0 0 0 0
0 0 0 1 0 1 0 0
2 16 N5 → N7 0 0 0 1 1 1 1 0
3 24 N6 → N1 1 0 0 1 1 2 1 0
4 32 N7 → N4 1 0 0 2 1 2 2 0
5 4 N4 → N5 1 0 0 3 2 2 2 0
6 11 N3 → N5 1 0 1 3 3 2 2 0
7 18 N2 → N3 1 1 2 3 3 2 2 0
8 25 N1 → N2 2 2 2 3 3 2 2 0

图5

76 72 74

70

节点	主				从			
N1	1	9	17		5	12	19	26
N2	2	10	18		6	13	20	27
N3	3	11	19		7	14	21	28
N4	4	12	20	28	8	15	22	
N5	5	13	21	29	1	16	23	
N6	6	14	22	30	2	9	24	
N7	7	15	23	31	3	10	17	32
N8	8	16	24	32	4	11	18	25
N9	25	26	27		29	30	31	

图6

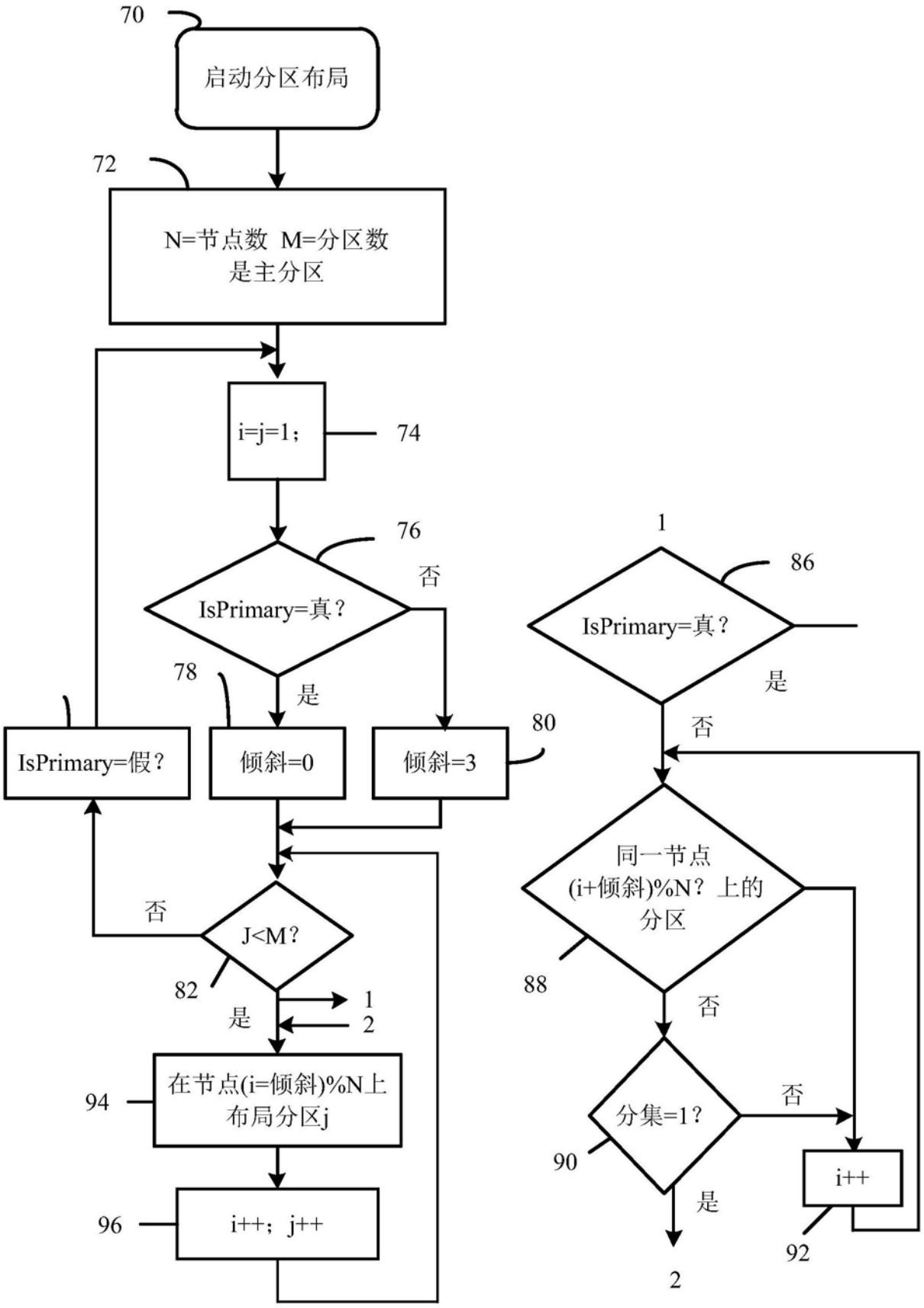

70
启动分区布局
72
N=节点数 M=分区数
是主分区
i=j=1;
74
76
IsPrimary=真?
否
是
78
IsPrimary=假?
倾斜=0
倾斜=3
80
否
J<M?
82
是
1
2
94
在节点(i=倾斜)%N上
布局分区j
96
i++; j++
1
86
IsPrimary=真?
是
否
同一节点
(i+倾斜)%N? 上的
分区
88
否
分集=1?
90
否
是
i++
92
2

图7

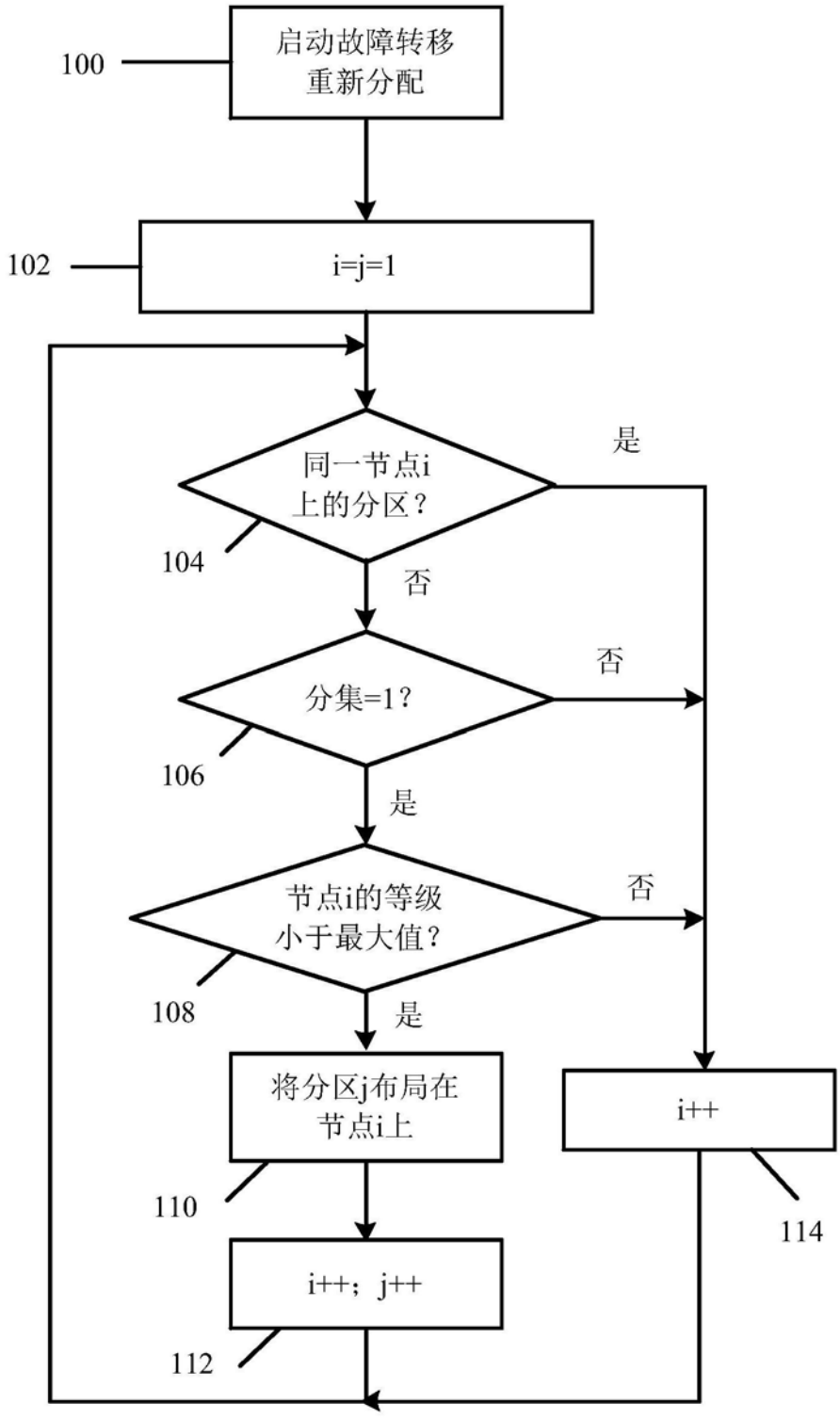

100
启动故障转移
重新分配
102
i=j=1
同一节点i
上的分区？
是
104
否
分集=1？
否
106
是
节点i的等级
小于最大值？
否
108
是
将分区j布局在
节点i上
i++
110
114
i++；j++
112

图8

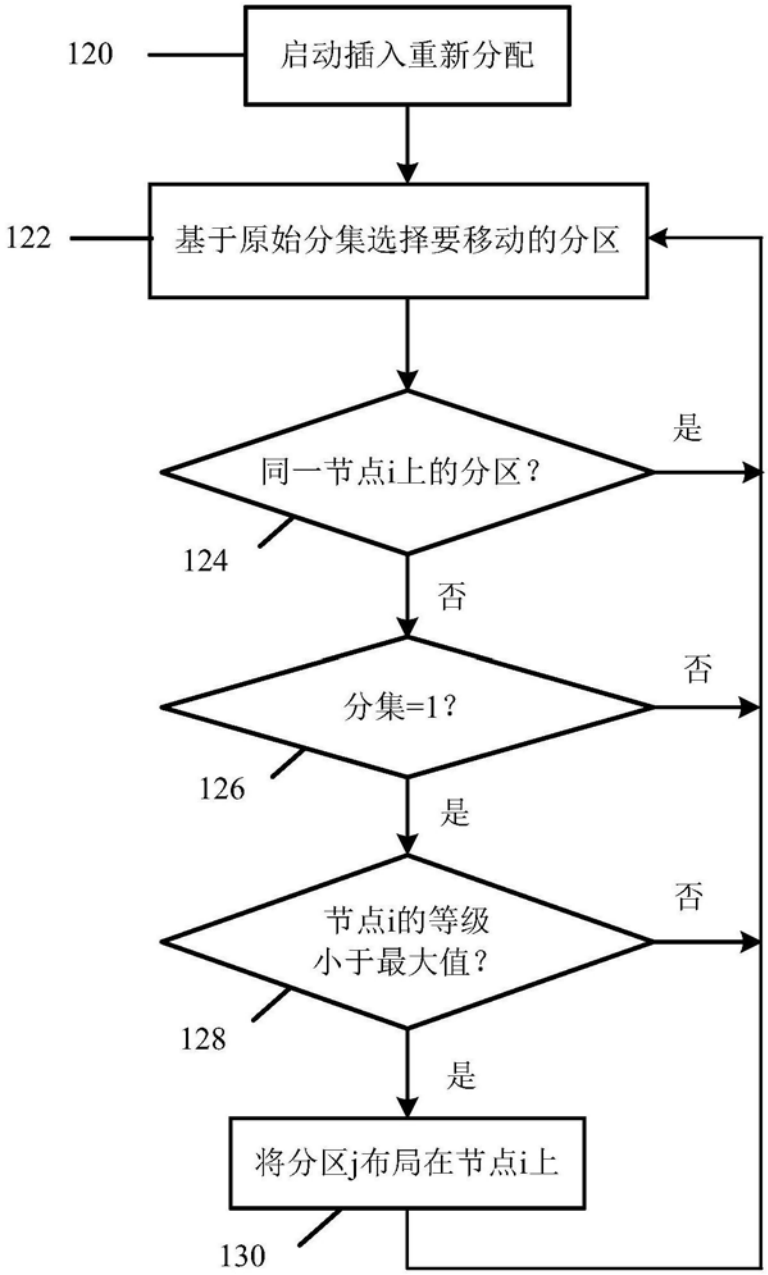

120
启动插入重新分配
122
基于原始分集选择要移动的分区
同一节点i上的分区？
是
124
否
分集=1？
否
126
是
节点i的等级
小于最大值？
否
128
是
将分区j布局在节点i上
130

图9