(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 102591899 B
(45)授权公告日 2016.07.06

(21)申请号 201110360076.1

(22)申请日 2011.11.14

(30)优先权数据
12/956,424 2010.11.30 US

(73)专利权人 国际商业机器公司
地址 美国纽约

(72)发明人 E·L·巴斯尼斯 R·K·克拉迪克 M·D·普法伊费尔 J·M·桑多索索

(74)专利代理机构 北京市中咨律师事务所 11247
代理人 于静 杨晓光

(51)Int.Cl.
G06F 17/30(2006.01)

(56)对比文件
US 2005188358 A1,2005.08.25,
US 2006048098 A1,2006.03.02,
US 2009158257 A1,2009.06.18,
US 2003041315 A1,2003.02.27,
US 2011088016 A1,2011.04.14,
US 2011161730 A1,2011.06.30,

Bugra Gedik 等.《tools and strategies for debugging distributed stream processing applications》.《online02.10.2009 in wiley interscience(www.interscience.weley.com)》.2009,1347-1376.

Bugra Gedik et al..《tools and strategies for debugging distributed stream processing applications》.《online02.10.2009 in wiley interscience(www.interscience.weley.com)》.2009,1347-1376.

审查员 刘宇儒

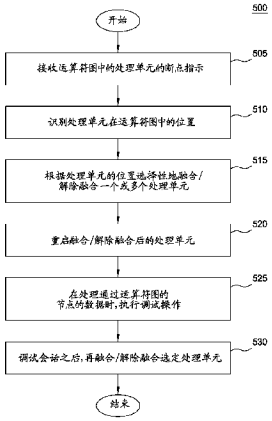
权利要求书2页 说明书8页 附图8页

(54)发明名称

用于基于流的调试的方法和系统

(57)摘要

本发明涉及一种用于基于流的调试的方法和系统。描述了通过选择性地融合(解除融合)在一组计算节点上运行的处理单元,以降低调试一个或多个处理单元对总体运行环境的影响的方式,调试基于流的数据库应用中的一个或多个处理单元的技术。除了融合和解除融合处理单元或以其他方式修改流应用的状态以外,调试应用和流管理器还可以以各种方式修改应用流中的数据流以尽量减少由调试会话导致的任何破坏。



CN 102591899 B

1.一种调试从一个或多个计算节点上执行的多个处理单元形成的流应用的方法，所述方法包括：

接收所述多个处理单元中的第一处理单元内的调试断点的位置指示；

评估运算符图以识别所述第一处理单元相对于所述多个处理单元中的一个或多个其他处理单元的位置；

通过克隆，融合以及解除融合中的至少一种修改所述流应用的状态以允许调试所述第一处理单元；以及

启动所述第一处理单元的调试会话；

其中所述克隆包括：生成所述第一处理单元的调试克隆单元；将所述断点插入所述调试克隆单元；以及将所述调试克隆单元插入所述流应用；

所述融合包括：暂停至少第二处理单元的执行；从所述第一和第二处理单元生成融合后的处理单元；将所述断点插入所述融合后的处理单元；以及将所述融合后的处理单元插入所述流应用；

其中所述解除融合包括：暂停所述第一处理单元的执行，所述第一处理单元是包括所述第一处理单元和第二处理单元的融合后的处理单元；生成包括所述第一处理单元的第一解除融合后的处理单元；将所述断点插入所述第一解除融合后的处理单元；生成包括至少所述第二处理单元的第二解除融合后的处理单元；将所述第一和第二解除融合后的处理单元插入所述流应用。

2.如权利要求1中所述的方法，其中每个处理单元在所述计算节点中的一个给定计算节点上作为分配有进程ID的计算进程而执行。

3.如权利要求1中所述的方法，其中所述第一处理单元被配置为接收第一N元组、处理所述第一N元组，以及生成发送到第二处理单元的第二N元组。

4.如权利要求1中所述的方法，还包括：

在所述调试会话之后，恢复所述流应用的状态。

5.一种调试从一个或多个计算节点上执行的多个处理单元形成的流应用的系统，所述系统包括：

配置为接收所述多个处理单元中的第一处理单元内的调试断点的位置指示的模块；

配置为评估运算符图以识别所述第一处理单元相对于所述多个处理单元中的一个或多个其他处理单元的位置的模块；

配置为通过克隆模块，融合模块以及解除融合模块中的至少一种修改所述流应用的状态以允许调试所述第一处理单元的模块；以及

配置为启动所述第一处理单元的调试会话的模块；

其中所述克隆模块包括：配置为生成所述第一处理单元的调试克隆单元的模块；配置为将所述断点插入所述调试克隆单元的模块；以及配置为将所述调试克隆单元插入所述流应用的模块；

其中所述融合模块包括：配置为暂停至少第二处理单元的执行的模块；配置为从所述第一和第二处理单元生成融合后的处理单元的模块；配置为将所述断点插入所述融合后的处理单元的模块；以及配置为将所述融合后的处理单元插入所述流应用的模块；

其中所述解除融合模块包括：配置为暂停所述第一处理单元的执行的模块，所述第一

处理单元是包括所述第一处理单元和第二处理单元的融合后的处理单元;配置为生成包括所述第一处理单元的第一解除融合后的处理单元的模块;配置为将所述断点插入所述第一解除融合后的处理单元的模块;配置为生成包括至少所述第二处理单元的第二解除融合后的处理单元的模块;配置为将所述第一和第二解除融合后的处理单元插入所述流应用的模块。

6.如权利要求5中所述的系统,其中每个处理单元在所述计算节点中的一个给定计算节点上作为分配有进程ID的计算进程而执行。

7.如权利要求5中所述的系统,其中所述第一处理单元被配置为接收第一N元组、处理所述第一N元组,以及生成发送到第二处理单元的第二N元组。

8.如权利要求5所述的系统,还包括:

配置为在所述调试会话之后,恢复所述流应用的状态的模块。

用于基于流的调试的方法和系统

技术领域

[0001] 本发明涉及一种用于基于流的调试的方法和系统。

背景技术

[0002] 当计算机数据库变得极其复杂时，对数据库系统的计算要求也快速增加。数据库系统通常被配置为将存储数据的过程与访问、操纵或使用数据库中存储的数据进行分离。更具体地说，数据库使用这样的模型：其中数据首先被存储，然后被索引，然后被查询。但是，此模型无法满足某些实时应用的性能要求。例如，数据库系统接收和存储入站数据的速率会限制可以处理或以其他方式计算多少数据，这又会限制数据库应用实时处理大量数据的能力。

[0003] 为了解决此问题，出现了基于流的计算和基于流的数据库计算作为数据库系统开发技术。目前存在这样的产品：所述产品允许用户创建在流数据到达数据库文件之前对流数据进行处理和查询的应用。借助这种新兴的技术，用户可以在入站数据记录仍在流入之时指定应用于它们的处理逻辑，且结果可以在几毫秒之内得出。使用此类处理构建应用开启了新的编程范例，它将允许开发各种新颖的应用、系统和过程并且为应用编程人员和数据库开发人员带来了新的挑战。

发明内容

[0004] 本发明的实施例提供了用于调试基于流的数据库应用中的一个或多个处理单元的技术。例如，本发明的一个实施例包括一种调试从一个或多个计算节点上执行的多个处理单元形成的流应用的方法。所述方法通常可包括接收所述多个处理单元中的第一处理单元内的调试断点的位置指示，以及评估运算符图以识别所述第一处理单元相对于所述多个处理单元中的一个或多个其他处理单元的位置。此方法还可包括修改所述流应用的状态以允许调试所述第一处理单元以及启动所述第一处理单元的调试会话。

[0005] 本发明的另一实施例包括包含程序的计算机可读存储介质，当执行所述程序时，所述程序将执行用于调试从一个或多个计算节点上执行的多个处理单元形成的流应用的操作。所述操作本身通常可包括接收所述多个处理单元中的第一处理单元内的调试断点的位置指示，以及评估运算符图以识别所述第一处理单元相对于所述多个处理单元中的一个或多个其他处理单元的位置。所述操作还可包括修改所述流应用的状态以允许调试所述第一处理单元以及启动所述第一处理单元的调试会话。

[0006] 本发明的又一实施例包括具有多个计算节点的系统，每个计算节点包括处理器和存储器。所述计算节点被配置为执行流应用的处理单元。所述系统还可包括同样包含处理器和存储器的管理系统。所述存储器存储流调试应用，所述流调试应用当在所述管理系统上执行时，被配置为执行用于调试在所述多个计算节点上执行的所述流应用的操作。

[0007] 所述操作本身通常可包括接收所述多个处理单元中的第一处理单元内的调试断点的位置指示，以及评估运算符图以识别所述第一处理单元相对于所述多个处理单元中的

一个或多个其他处理单元的位置。所述操作还可包括修改所述流应用的状态以允许调试所述第一处理单元以及启动所述第一处理单元的调试会话。

附图说明

[0008] 通过参考附图，可以获得详细地实现和理解上述各方面的方式，以及上面简要介绍的本发明实施例的更具体的说明。但是需要指出，附图仅示出本发明的典型实施例，因此不能被认为是对本发明的范围的限制，因为本发明可允许其他等同的有效实施例。

[0009] 图1A-1B示出根据本发明的一个实施例的被配置为执行流数据库应用的计算基础设施；

[0010] 图2是根据本发明的一个实施例的图1的分布式计算节点的更详细的视图；

[0011] 图3是根据本发明的一个实施例的流数据库应用中的管理计算系统的更详细的视图；

[0012] 图4示出根据本发明的一个实施例的流数据库应用中的计算节点的一个实例；

[0013] 图5示出根据本发明的一个实施例的调试流数据库应用中的处理单元的方法；以及

[0014] 图6-8提供根据本发明的各实施例的首先在图4中示出的被修改以例示调试流应用的情况的计算节点和流应用的实例。

具体实施方式

[0015] 在流应用中，运算符相互连接以便数据从一个处理单元流向下一个处理单元(例如通过TCP/IP套接字)。通过跨节点分布应用(通过创建许多小的可执行代码段(运算符))以及在多个节点上复制处理单元并在处理单元之间进行负载平衡来获得可伸缩性。流应用中的处理单元(和运算符)可以融合在一起以形成较大的处理单元。这样做可允许处理单元共享公共进程空间，从而导致运算符之间的通信远快于使用进程间通信技术(例如，使用TCP/IP套接字)获得的通信。此外，处理单元可以被动态地插入表示通过流应用的数据流的运算符图或被从运算符图动态地删除，以及在运行时与流应用融合或取消融合。

[0016] 尽管可以通过查看运行代码所生成的日志文件来调试分布式流环境中的处理单元，但是通常还需要完整的调试会话。然而，无论是否命中断点，运行调试器都会减缓被调试进程的速度，并且触发断点可导致流过正在调试的运算符的数据大幅减缓(如果不是完全暂停)。因此，运行正在调试的处理单元可导致“流阻塞”，因为流向该处理单元中的运算符的数据流不会停止流动。更概括地说，流计算基于流不断流过运算符的前提。

[0017] 当多个处理单元融合在一起以形成单个运行进程时，触发一个处理单元中的断点会导致融合后的处理单元中的所有处理单元都停止处理，从而可能使大部分运算符图停止。另一方面，一组解除融合后的处理单元可能包括被配置为加入来自第二和第三处理单元的数据流的第一处理单元。如果触发第二或第三处理单元中的断点，则可能破坏流处理结果。

[0018] 本发明的各实施例提供了调试基于流的应用中的一个或多个处理单元的技术。具体地说，本发明的各实施例提供了以降低调试一个或多个处理单元对整体运行环境的影响的方式在调试处理单元期间修改流应用的状态的技术。例如，假设需要调试具有二十个运

算符的融合后的处理单元中的一个运算符。在这种情况下，流调试器可以被配置为从运行的流中移除融合后的处理单元、解除融合运算符，然后将它们重新插入运行的流。具有被调试的单个运算符的处理单元然后可作为独立运行进程(具有其自己的进程ID(PID)和存储器空间)执行，这允许独立于其他十九个运算符而调试该运算符。一旦调试会话完成，调试器可重新融合已解除融合的运算符，将具有二十个融合后的运算符的处理单元恢复到调试会话之前存在的同一运行状态。

[0019] 在另一实施例中，调试器可以作为调试会话的一部分而将多个独立处理单元融合在一起。这可用于在调试一个运算符时暂停一部分运算符图的运行。例如，此操作可允许满足处理单元之间的任何相关性。类似地，当正在调试给定处理单元时，调试器可以在适当情况下阻止数据进入流。这可在所调试运算符图中的处理单元的上游(下游)的任何点处发生。更具体地说，调试器可阻止数据流向源运算符(即，是流过运算符图的数据的源点的处理单元)或阻止数据流向运算符图中的特定部分(或特定处理单元)。相反地，调试器可被配置为将入站数据的某些元组识别为免于断点。

[0020] 作为阻止数据流动的备选方式，可以“丢弃(load shed)”正在调试的处理单元处的数据(或选择性地允许数据在某些条件下流动)。这可有助于避免在调试会话完成时由于数据而使系统过载；尤其是在数据因未及时处理而可能失去价值时。在另一实施例中，调试器可以被配置为复制处理单元，以便在复制的单元中进行调试。流出被调试处理单元的数据可以被丢弃，而不是被发送到链接的下游运算符。此外，可以将正在调试的运算符输出的结果与实际运算符相比较以作为对调试进程准确性的检查。

[0021] 以下参考了本发明的各实施例。但是应该理解，本发明并不限于描述的特定实施例。相反，将构想以下特性和元素(无论是否与不同实施例相关)的任意组合以实施和实现本发明。此外，虽然本发明的各实施例可以相对于其他可能解决方案和/或相对于现有技术实现优点，但特定优点是否由给定实施例实现并不会限制本发明。因此，以下方面、特性、实施例和优点仅是示例性的，并不被视为所附权利要求的元素或限制，除非在权利要求(多个)中明确指出。同样，对“本发明”的引用不应被理解为在此披露的任何发明主题的概括，并且不应被视为所附权利要求的元素或限制，除非在权利要求(多个)中明确指出。

[0022] 如本领域的技术人员将理解的，本发明的各方面可以体现为系统、方法或计算机程序产品。因此，本发明的各方面可以采取完全硬件实施例、完全软件实施例(包括固件、驻留软件、微代码等)或组合了在此通常被称为“电路”、“模块”或“系统”的软件和硬件方面的实施例的形式。此外，本发明的各方面可以采取体现在一个或多个计算机可读介质(在介质中包含计算机可读程序代码)中的计算机程序产品的形式。

[0023] 可以使用一个或多个计算机可读介质的任意组合。所述计算机可读介质可以是计算机可读信号介质或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、红外线或半导体系统、装置或设备或它们的任意适当组合。计算机可读存储介质的更具体的实例(非穷举列表)可以包括以下项：具有一条或多条线的电连接、便携式计算机软盘、硬盘、随机存取存储器(RAM)、只读存储器(ROM)、可擦写可编程只读存储器(EPROM或闪存)、光纤、便携式光盘只读存储器(CD-ROM)、光存储设备、磁存储设备或它们的任意适当组合。在本文档的上下文中，计算机可读存储介质可以是任何能够包含或存储由指令执行系统、装置或设备使用或与所述指令执行系统、装置或设备结合的程序的有形介

质。

[0024] 计算机可读信号介质可以包括例如在基带中或作为载波的一部分传播的带有计算机可读程序代码的数据信号。此类传播信号可以采取任何不同的形式，包括但不限于电磁、光或它们的任意适合组合。计算机可读信号介质可以是任何并非计算机可读存储介质并且可以传送、传播或传输由指令执行系统、装置或设备使用或与所述指令执行系统、装置或设备结合的程序的计算机可读介质。

[0025] 可以使用任何适当的介质(包括但不限于无线、有线、光缆、RF等或它们的任意适当组合)来传输计算机可读介质中包含的程序代码。

[0026] 用于执行本发明的各方面的操作的计算机程序代码可以使用一种或多种编程语言的任意组合来编写，所述编程语言包括诸如Java、Smalltalk、C++之类的面向对象的编程语言或者诸如“C”编程语言或类似的编程语言之类的常规过程编程语言。所述程序代码可以完全地在用户的计算上执行、部分地在用户的计算机上执行、作为一个独立的软件包执行、部分在用户的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。在后一种情形中，远程计算机可以通过任何种类的网络一包括局域网(LAN)或广域网(WAN)一连接到用户的计算机，或者，可以(例如利用因特网服务提供商来通过因特网)连接到外部计算机。

[0027] 下面参考根据本发明的各实施例的方法、装置(系统)和计算机程序产品的流程图和/或方块图对本发明的各方面进行描述。将理解，所述流程图和/或方块图的每个方块以及所述流程图和/或方块图中的方块的组合可以由计算机程序指令来实现。这些计算机程序指令可以被提供给通用计算机、专用计算机或其他可编程数据处理装置的处理器以产生机器，以便通过所述计算机或其他可编程数据处理装置的处理器执行的所述指令产生用于实现在一个或多个流程图和/或方块图方块中指定的功能/操作的装置。

[0028] 这些计算机程序指令也可以被存储在可引导计算机、其他可编程数据处理装置或其他设备以特定方式执行功能的计算机可读介质中，以便存储在所述计算机可读介质中的所述指令产生一件包括实现在所述一个或多个流程图和/或方块图方块中指定的功能/操作的指令的制品。

[0029] 所述计算机程序指令还可被加载到计算机、其他可编程数据处理装置或其他设备，以导致在所述计算机、其他可编程装置或其他设备上执行一系列操作步骤以产生计算机实现的过程，从而在所述计算机或其他可编程装置上执行的指令提供用于实现在一个或多个流程图和/或方块图方块中指定的功能/操作的过程。

[0030] 本发明的各实施例可以通过云计算基础设施被提供给最终用户。云计算通常指通过网络作为服务来供应可扩展的计算资源。更正式地说，云计算可以被定义为在计算资源及其基础技术架构(例如，服务器、存储设备、网络)之间提供抽象的计算能力，从而能够对可配置的计算资源共享池进行方便、按需的网络访问，可以以最少的管理工作或服务提供商交互快速地供应和释放所述可配置的计算资源。因此，云计算允许用户访问“云”中的虚拟计算资源(例如，存储设备、数据、应用，甚至完整的虚拟化计算系统)，而不考虑用于提供所述计算资源的基础物理系统(或这些系统的位置)。

[0031] 通常，云计算资源通过按使用付费的方式被提供给用户，其中用户仅针对实际使用的计算资源(例如，用户使用的存储空间量或用户实例化的虚拟化系统的数量)付费。用

户可以通过因特网在任何时间、从任何位置访问驻留在云中的任何资源。在本发明的上下文中，用户可以访问云中提供的应用或相关数据。例如，用于创建流数据库应用的节点可以是由云服务提供商托管的虚拟机。

[0032] 图1A-1B示出了根据本发明的一个实施例的被配置为执行流应用的计算基础设施100。如所示出的，计算基础设施100包括管理系统105和多个计算节点130_{1-4}，每个计算节点连接到通信网络120。此外，管理系统105包括运算符图132和流管理器134。如下面更详细描述的，运算符图132表示从一个或多个源处理单元(PE)开始一直到一个或多个汇(sink)处理单元的流应用。数据元素流入流应用的源PE并由该PE处理。通常，处理单元从流接收具有数据元素的N元组以及将具有数据元素的N元组发出到流(除了流在该处终止的汇PE之外)。当然，由处理单元接收的N元组不必是向下游发送的同一N元组。此外，处理单元可以被配置为以不同于N元组的格式接收或发出数据(例如，处理单元可交换标记为XML文档的数据)。此外，每个处理单元可以被配置为对所接收的元组执行任何形式的数据处理功能，例如包括写入数据库表或执行其他数据库操作(例如数据联接、拆分、读取等)以及执行其他数据分析功能或操作。

[0033] 流管理器134可以被配置为监视在计算节点130_{1-4}上运行的流应用，以及更改运算符图132的结构。例如，流管理器134可以将处理单元(PE)从一个计算节点130移动到另一个节点，以便例如管理计算基础设施100中的计算节点130的处理负载。进而，流管理器134可以通过插入、删除、融合、解除融合或以其他方式修改在计算节点130_{1-4}上运行的处理单元(或何种数据元组流入处理单元)来控制流应用。

[0034] 图1B示出了包括十个在计算节点130_{1-4}上运行的处理单元(标记为PE1-PE10)的实例运算符图。虽然处理单元可以作为独立运行的进程(具有它自己的进程ID(PID)和存储器空间)执行，但是也可以融合多个处理单元以作为单个进程(具有PID和存储器空间)运行。在两个(或更多)处理单元独立运行的情况下，可以使用网络套接字(例如，TCP/IP套接字)进行进程间通信。但是，当进程融合在一起时，融合后的处理单元可以使用更快速的通信技术在处理单元(以及每个处理单元中的运算符)之间传递N元组(或其他数据)。

[0035] 如图所示，运算符图从源PE 135(标示为PE1)开始并在PE 140_{1-2}(标示为PE6和PE10)结束。计算节点130_1包括源PE1以及PE2和PE3。源PE1发出PE2和PE3所接收的元组。例如，PE1可以拆分在元组中接收的数据单元并将某些数据单元传递到PE2，同时将其他数据元素传递到PE3。流向PE2的数据产生被发出到计算节点130_2上的PE4的元组。由PE4发出的数据元组流向汇PE6 140_1。类似地，从PE3流向PE5的数据元组也到达汇PE6 140_1。因此，除了对于此实例运算符图是汇之外，PE6可以被配置为执行联接运算，从而组合从PE4和PE5接收的元组。此实例运算符图还示出了从PE3流向计算节点130_3上的PE7的数据元组，计算节点130_3本身示出了流向PE8并循环返回PE7的数据元组。从PE8发出的数据元组流向计算节点130_4上的PE9，PE9又发出要由汇PE10 140_2处理的元组。

[0036] 图2是根据本发明的一个实施例的图1A-1B的计算节点130的更详细的视图。如所示出的，计算节点130包括但不限于中央处理单元(CPU)205、网络接口215、互连220、存储器225和存储单元230。计算节点130还可以包括用于将I/O设备212(例如，键盘、显示器和鼠标设备)连接到计算节点130的I/O设备接口210。

[0037] CPU 205检索和执行存储在存储器225中的编程指令。类似地，CPU205存储和检索

驻留在存储器225中的应用数据。互连220用于在CPU 205、I/O设备接口210、存储单元230、网络接口215和存储器225之间传输编程指令和应用数据。所包括的CPU 205表示单个CPU、多个CPU、具有多个处理核心的单个CPU等。所包括的存储器225通常表示随机存取存储器。诸如硬盘驱动器、固态设备(SSD)或闪存存储驱动器之类的存储单元230可以存储非易失性数据。

[0038] 在此实例中，存储器225包括融合后的处理单元(PE)235、解除融合后的PE 245、调试器应用250以及流连接数据255。融合后的PE 235包括一系列运算符240。如上所述，每个运算符240可以提供被配置为处理流入处理单元(例如，PE 235)的数据以及将数据发出到该PE中的其他运算符240和流应用中的其他PE的一小段可执行代码。此类PE可以位于同一计算节点130(例如，解除融合后的PE 245)上，也可以位于通过数据通信网络120访问的其他计算节点上。流连接数据255表示计算节点130上的PE之间的连接(例如，融合后的PE 240和解除融合后的PE 245之间的TCP/IP套接字连接)，以及到具有流应用中的上游或下游PE的其他计算节点130的连接(也通过TCP/IP套接字(或其他进程间数据通信机制))。

[0039] 缓冲的流数据260表示从上游处理单元(或从流应用的数据源)流入计算节点105的数据的存储空间。例如，缓冲的流数据可以包括等待由PE240或245之一处理的数据元组。缓冲的流数据260还可以存储由PE 240或245执行的数据处理的结果，所述结果将被发送到下游处理单元(或被丢弃)。

[0040] 调试器250提供被配置为允许开发人员调试计算节点130上运行的处理单元245的软件应用。例如，调试器250可用于设置断点、执行指令中的指令步进函数调用、检查变量以及提供其他各种用于调试处理单元245的功能和/或特性。在本发明的上下文中，调试器250可以被配置为选择性地融合和解除融合PE(或以其他方式修改流应用的状态)以促进调试过程。例如，在一个实施例中，调试器可以如开发人员指定的那样融合和解除融合PE。备选地，调试器250可被配置为提供用于调试一个或多个给定处理单元的流状态。为此，调试器250可以与管理系统130上的流调试器交互以分析运算符图以及处理单元跨一系列计算节点105的分布，以便确定如何修改与流应用关联的运算符图。除了融合和解除融合处理单元之外，调试器250还可以例如通过以下方式修改流应用的运行状态：暂停(或限制)跨被调试PE的数据流、指定不参与调试过程的数据、复制PE以创建PE的调试副本，或执行其他操作以允许调试一个或多个处理单元。

[0041] 图3是根据本发明的一个实施例的图1的服务器计算系统105的更详细的视图。如所示出的，服务器计算系统105包括但不限于中央处理单元(CPU)305、网络接口315、互连320、存储器325和存储单元330。客户机系统130还可以包括将I/O设备312(例如，键盘、显示器和鼠标设备)连接到服务器计算系统105的I/O设备接口310。

[0042] 与图2的CPU 205一样，CPU 305被配置为检索和执行存储在存储器325和存储单元330中的编程指令。类似地，CPU 305被配置为存储和检索驻留在存储器325和存储单元330中的应用数据。互连320被配置为在CPU 305、I/O设备接口310、存储单元330、网络接口315和存储器325之间移动诸如编程指令和应用数据之类的数据。与CPU 205一样，所包括的CPU 305表示单个CPU、多个CPU、具有多个处理核心的单个CPU等。所包括的存储器325通常表示随机存取存储器。网络接口315被配置为通过通信网络120传输数据。虽然被示为单个单元，但存储单元330可以是固定和/或可移动的存储设备(例如固定盘驱动器、可移动存储器卡、

光存储设备、SSD或闪存设备、网络连接存储(NAS)或到存储区域网络(SAN)设备的连接)的组合。

[0043] 如图所示,存储器325存储流调试器335、流编辑器300以及流管理器132。存储单元330包括运算符图134。如上所述,流调试器可用于管理一个(或多个)计算节点105上的处理单元(PE)的调试。例如,流调试器335可确定一个处理节点应与其他PE融合(或解除融合)。在这种情况下,流调试器可确定(通过运算符图)一大组PE已在一个计算节点上融合,但是调试断点仅包括在几个处理单元中。在这种情况下,流调试器335可决定将具有断点的PE与该较大的组解除融合。在一个实施例中,流调试器335通过调用流编辑器340以根据需要重新编辑PE/运算符源代码的单元来解除融合PE。在当前实例中,带有断点的PE可被重新编辑成第一PE,而剩余的PE(来自融合后的PE)被重新编辑成第二PE。准备就绪之后,流管理器132可以从运行的流中移除融合后的PE,并代之以流编辑器340所生成的第一和第二PE。之后,命中第一PE中的断点,第一PE停止执行而不破坏第二PE所执行的任何处理。

[0044] 图4示出了根据本发明的一个实施例的流应用中的计算节点的实例。如所示出的,计算节点130_2上的融合后的处理单元405包括三个处理单元(标记为PE1-PE3),处理单元405接收N元组数据流并将N元组发出到计算节点130_3上的处理单元410(标记为PE4)。在此实例中,融合后的处理单元405包括源PE 135(标记为PE1),源PE 135接收包括<姓名,部门,薪金,性别>的元组。PE1获得此N元组,并根据由PE1接收的元组中的性别值生成一组被发送到PE2的元组和另一组被发送到PE3的元组。依次地,PE2和PE3针对从PE1接收的每个元组执行数据库写入,并生成被发送到PE4的包括<姓名,部门,薪金>的元组。一旦接收到该元组,PE4就访问第三方Web服务并生成在流应用中进一步向下游发送的元组。

[0045] 图5示出根据本发明的一个实施例的调试流数据库应用中的处理单元的方法500。如图所示,方法500从步骤505开始,其中调试器应用接收流应用的运算符图中包括的处理单元的断点指示。在步骤510,调试器应用可识别该处理单元在运算符图中相对于其他处理单元的位置。根据此位置,调试器应用可以确定融合(或解除融合)运算符图中的处理单元(步骤515)。此外,调试器应用可以例如通过复制带有断点的处理单元或使特定数据元组(或数据流)免于调试来对流应用的操作做出其他更改。类似地,调试器应用可以指定应阻止某些处理单元发送(或接收)数据元组,或应丢弃调试会话期间生成的数据处理结果。

[0046] 在步骤520,任何已经融合或解除融合的PE都可以被重启并部署到应用流(包括带有断点的PE)。在步骤525,一旦PE在应用流中运行(并且调试器选择对流应用做出任何其他更改),便可评估被调试PE的操作。例如,当触发断点时,被调试PE可暂停执行,从而允许开发人员接着通过步进函数调用以逐步的方式执行该PE,以便检查变量和执行其他任何调试功能以评估处理单元。同时,数据元组继续流向正在被调试的PE并可以被存储在缓冲区中。也就是说,一旦触发断点,就可以缓冲流向正在被调试的PE的元组,直到该PE恢复执行。

[0047] 在步骤530,一旦调试会话完成,调试应用便可恢复流应用的状态,撤销为了支持调试会话所做的任何更改。相应地,可以解除融合(或再融合)作为调试会话一部分融合(或解除融合)的PE、可以恢复元组流,或者可以丢弃缓冲的数据结果,可以删除重复的处理单元等。

[0048] 图6-8提供根据本发明的各实施例的首先在图4中示出的被修改以例示调试流应用的情况的计算节点和流应用的实例。首先,图6示出图4的融合后的处理单元405被部分地

解除融合以允许调试处理单元之一的实例。假设开发人员在PE2中添加了一个或多个断点。在这种情况下，调试器应用可确定能够独立于PE1和PE3执行的处理而调试PE2，并且从融合后的处理单元405解除融合PE2。因此，如图所示，融合后的PE 410已经被修改为删除PE2，从而形成部分解除融合的PE 405'。此外，PE2现在作为解除融合的PE 605执行。将PE2作为独立进程运行允许在数据元组流过部分解除融合的PE 405'时调试PE2而不延缓应用流。同时，运算符图的基本结构保持不变；元组仍然首先流向PE1，PE1生成发送到PE2和PE3的元组。

[0049] 类似地，图7示出图4的处理单元410与其他PE融合之后的实例。在该实例中，假设用户在PE4中放置断点。在这种情况下，调试器应用可确定下游处理单元(PE5)具有相关性，这样当PE4停止执行时，PE5也应停止执行。例如，PE5可以被配置为按照指定间隔定期从PE4接收数据。给定此相关性，当用户在PE4中插入断点时，调试器应用可将PE4和PE5融合成融合后的处理单元410'。这允许在不破坏PE5的操作的情况下调试PE4，因为当命中PE4中的断点时，将停止执行PE4和PE5两者的操作。

[0050] 图8示出修改图4的处理单元410的另一实例。在该实例中，用户再次在PE 410(标示为PE4)中放置断点。但是在此情况下，假设调试器应用确定调试PE 410不应破坏通过应用流流向PE 805(标示为PE5)的数据。在这种情况下，调试器应用可以创建调试克隆单元(clone)410'并将该克隆单元插入应用流。此结果在图8中示出，其中从PE 405流出的数据被复制并被发送到PE 410和调试克隆单元410'两者。因此，PE 410继续处理流并将元组发出到PE 805所接收的流。同时，从PE 405流出的数据还被发送到调试克隆单元410'并将元组输出到调试日志。这允许调试器在不破坏应用流的情况下针对PE 410'运行调试会话。

[0051] 有利地，本发明的上述实施例提供了调试基于流的应用中的一个或多个处理单元的技术。具体地说，本发明的各实施例提供了通过选择性地融合(解除融合)在一组计算节点上运行的处理单元，以降低调试一个或多个处理单元对总体运行环境的影响的方式来调试处理单元的技术。除了融合和解除融合处理单元以外，调试应用和流管理器还可以以各种方式修改应用流中的数据流以尽量减少由调试会话导致的任何破坏。

[0052] 附图中的流程图和方块图示出了根据本发明的各种实施例的系统、方法和计算机程序产品的可能实施方式的架构、功能和操作。在此方面，所述流程图或方块图中的每个方块都可以表示代码的模块、段或部分，所述代码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。还应指出，在某些备选实施方式中，在方块中说明的功能可以不按图中说明的顺序发生。例如，示为连续的两个方块可以实际上被基本同时地执行，或者某些时候，取决于所涉及的功能，可以以相反的顺序执行所述方块。还将指出，所述方块图和/或流程图的每个方块以及所述方块图和/或流程图中的方块的组合可以由执行指定功能或操作的基于专用硬件的系统或专用硬件和计算机指令的组合来实现。

[0053] 虽然上述内容涉及本发明的各实施例，但可以在不偏离本发明的基本范围的情况下设计本发明的其他和进一步的实施例，并且本发明的范围由以下权利要求确定。

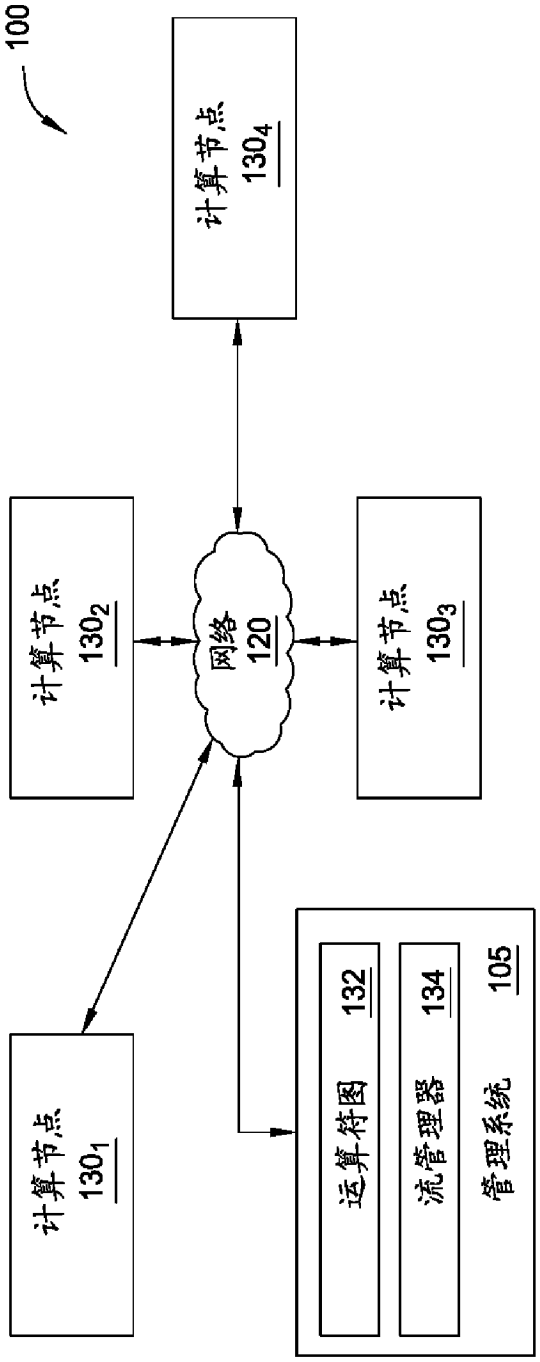

100
计算节点 130_4
计算节点 130_2
网络 120
计算节点 130_3
计算节点 130_1
运算符图 132
流管理器 134
管理系统 105

图1A

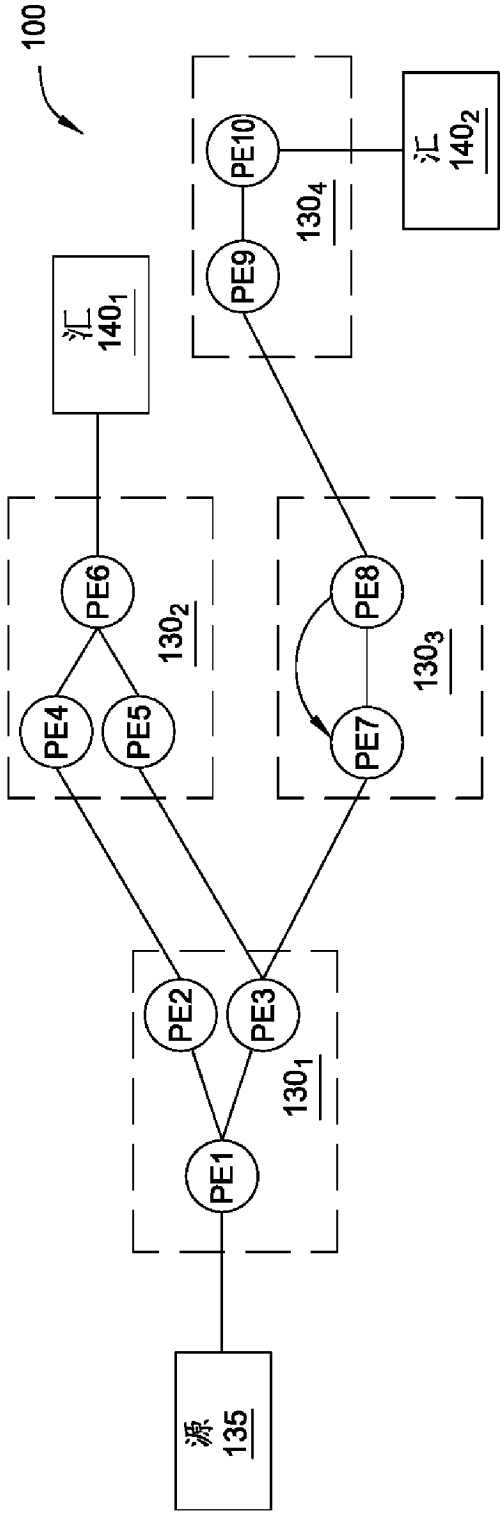

100
PE10
汇 140_2
130_4
PE9
汇 140_1
PE6
PE8
130_2
130_3
PE4
PE5
PE7
PE2
PE3
130_1
PE1
源 135

图1B

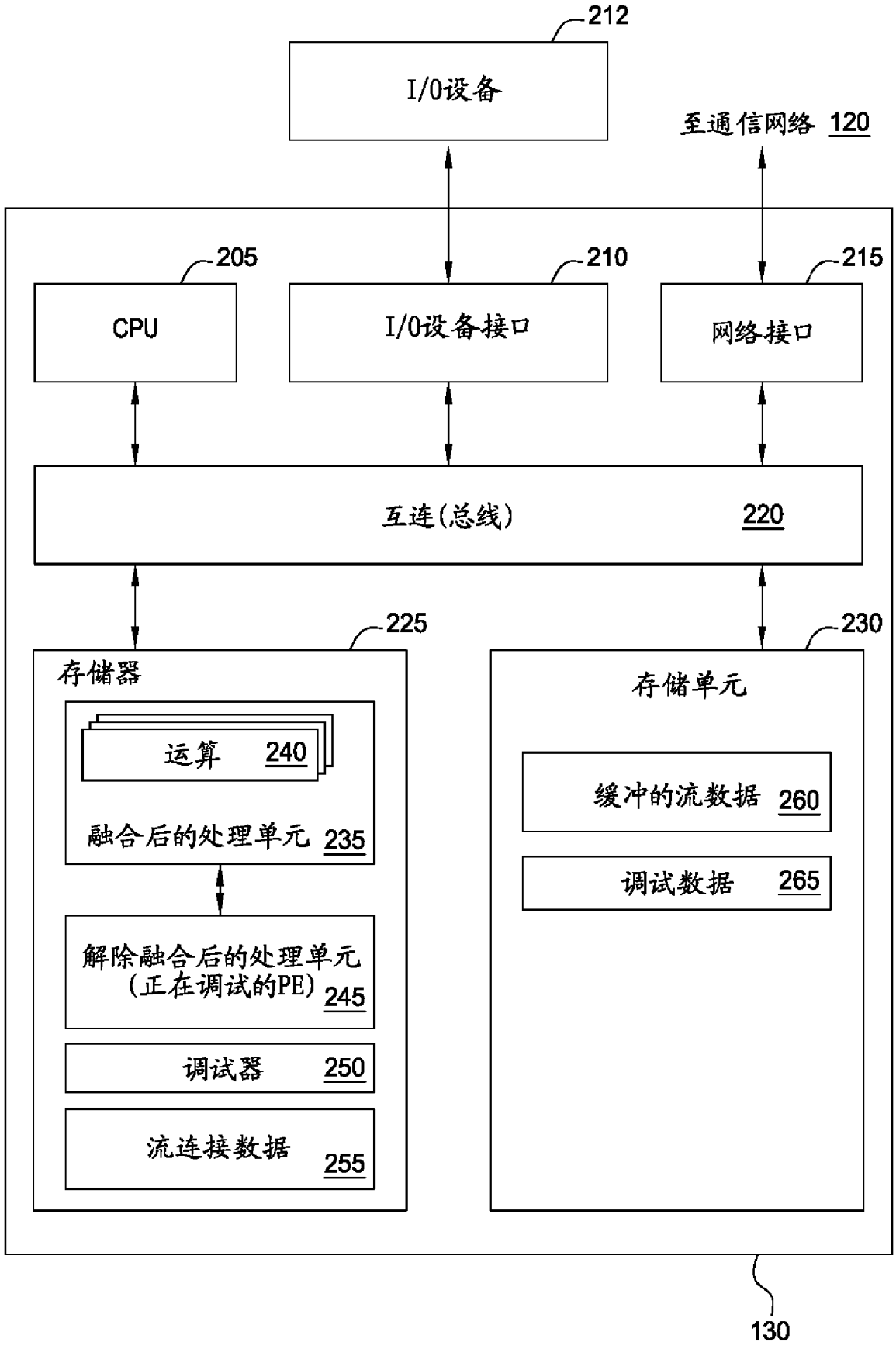

212
I/O设备
至通信网络 120
205
210
215
CPU
I/O设备接口
网络接口
互连(总线) 220
225
230
存储器
运算 240
融合后的处理单元 235
解除融合后的处理单元
(正在调试的PE) 245
调试器 250
流连接数据 255
存储单元
缓冲的流数据 260
调试数据 265
130

图2

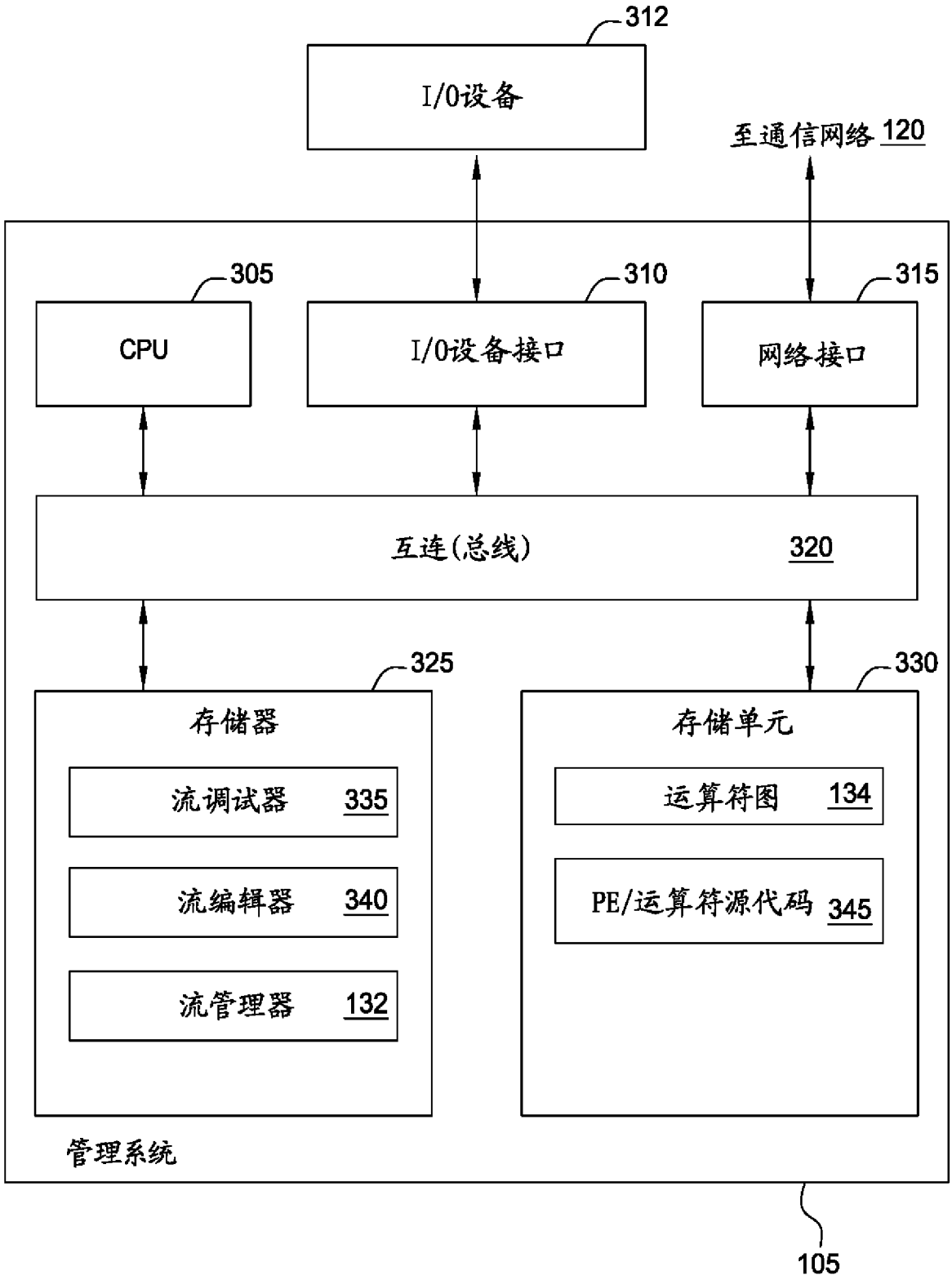

312
I/O设备
至通信网络 120
305
CPU
310
I/O设备接口
315
网络接口
互连(总线) 320
325
存储器
流调试器 335
流编辑器 340
流管理器 132
330
存储单元
运算符图 134
PE/运算符源代码 345
管理系统
105

图3

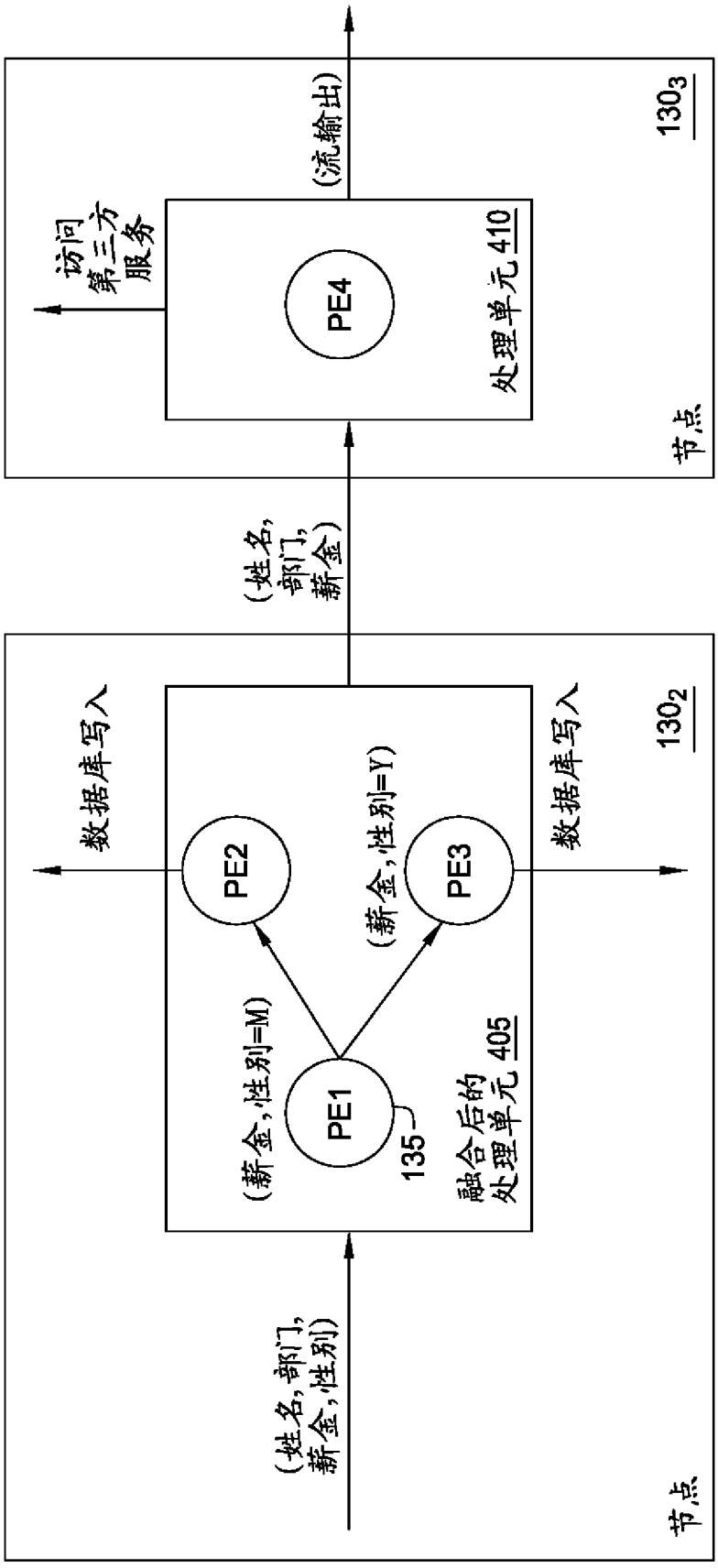

节点
(姓名,部门,
薪金,性别)
融合后的
处理单元 405
PE1
135
(薪金,性别=M)
(薪金,性别=Y)
PE2
PE3
数据库写入
数据库写入
1302
(姓名,
部门,
薪金)
节点
处理单元 410
PE4
访问
第三方
服务
(流输出)
1303

图4

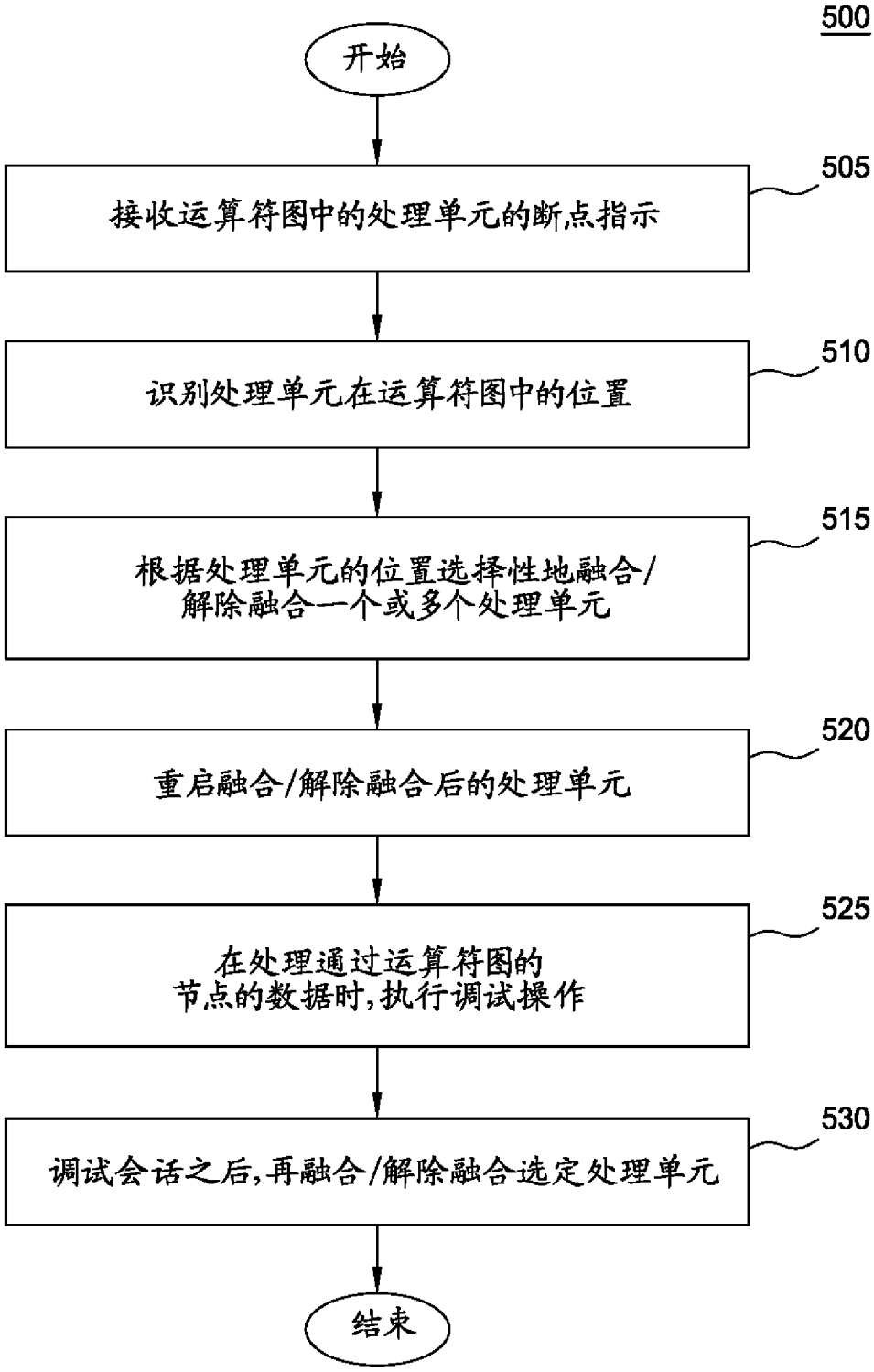

500
开始
505
接收运算符图中的处理单元的断点指示
510
识别处理单元在运算符图中的位置
515
根据处理单元的位置选择性地融合/
解除融合一个或多个处理单元
520
重启融合/解除融合后的处理单元
525
在处理通过运算符图的
节点的数据时,执行调试操作
530
调试会话之后,再融合/解除融合选定处理单元
结束

图5

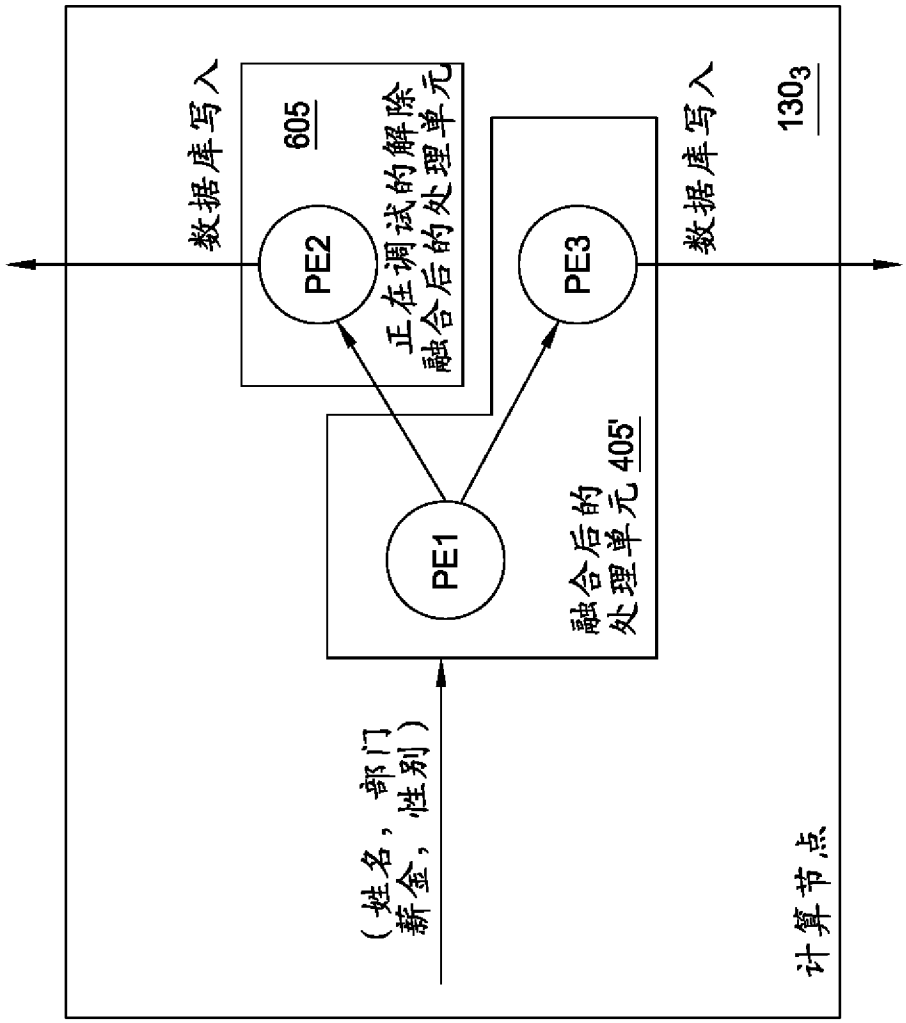

数据库写入
605
PE2
正在调试的解除融合后的处理单元
PE3
数据库写入
130_3
融合后的处理单元 405'
PE1
（姓名，部门
薪金，性别）
计算节点

图6

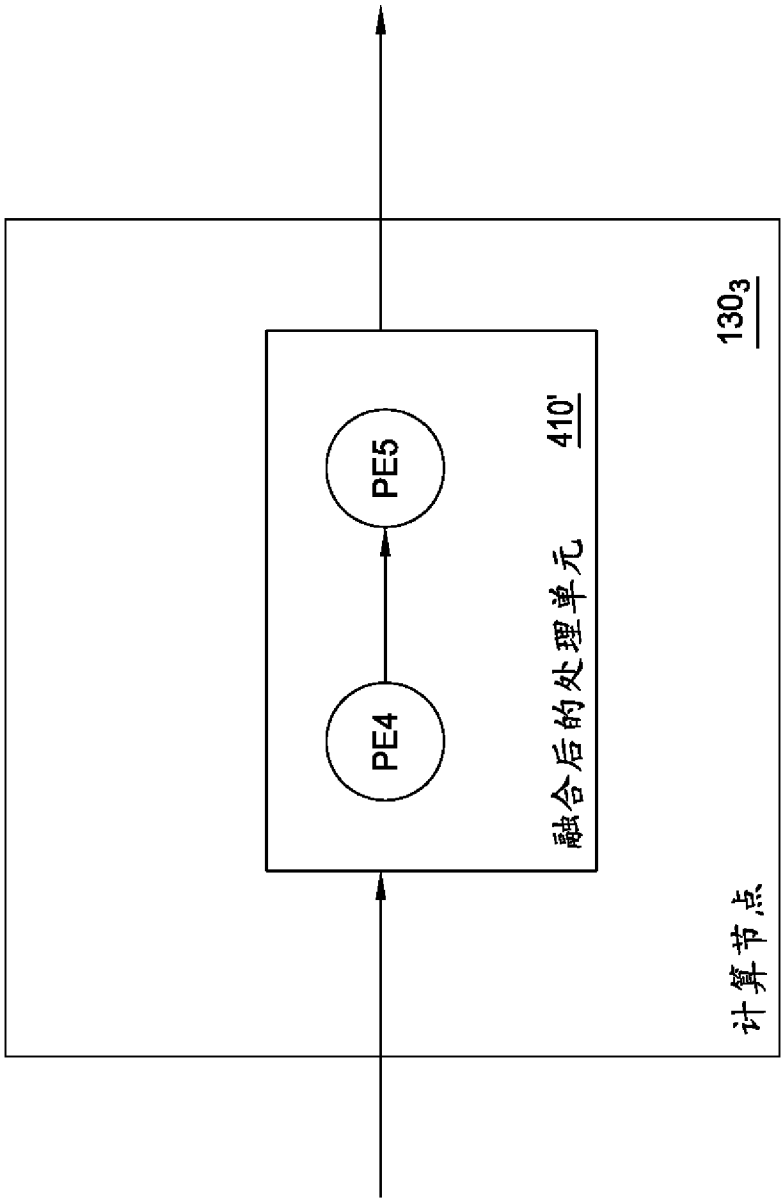


图7

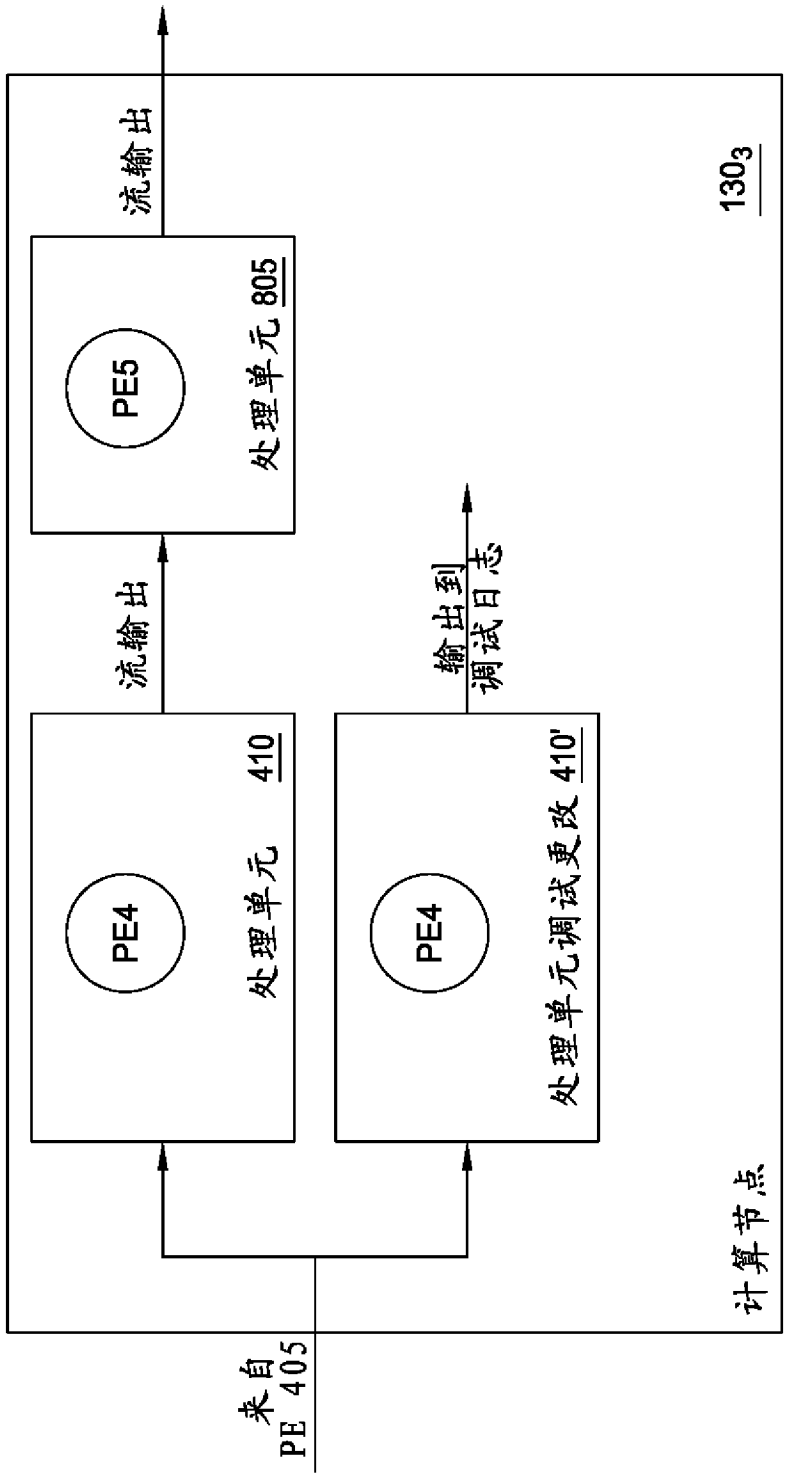

来自
PE 405
计算节点
1303
PE4
处理单元 410
PE4
处理单元调试更改 410'
输出到
调试日志
流输出
PE5
处理单元 805
流输出

图8