(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 104115124 A
(43)申请公布日 2014.10.22

(21)申请号 201380008762.X
(22)申请日 2013.02.06
(30)优先权数据
13/369,879 2012.02.09 US
(85)PCT国际申请进入国家阶段日
2014.08.08
(86)PCT国际申请的申请数据
PCT/IB2013/050983 2013.02.06
(87)PCT国际申请的公布数据
WO2013/118062 EN 2013.08.15
(71)申请人 国际商业机器公司
地址 美国纽约阿芒克
(72)发明人 M·韦弗 J·凯里 C·阿彻
(74)专利代理机构 北京市金杜律师事务所 11256
代理人 酆迅
(51)Int.Cl.
G06F 11/00(2006.01)

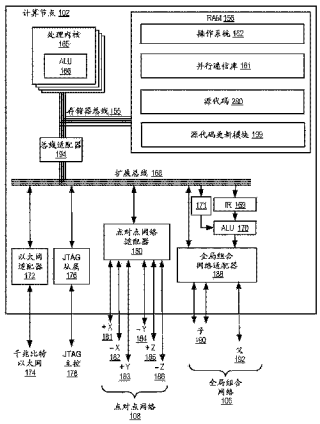
权利要求书3页 说明书15页 附图8页

(54)发明名称

对在多个计算节点上执行的源代码实施更新

(57)摘要

提供了用于对在多个计算节点上执行的源代码实施更新的方法、装置和计算机程序产品。实施方式包括由计算节点接收广播更新通知消息，其指示存在针对在多个计算节点上执行的源代码的更新；响应于接收到该更新通知消息，实施分布式屏障；基于该分布式屏障，在源代码内的特定位置停止该源代码的执行；基于该分布式屏障，在计算节点的存储器中适当更新包括保留工件数据的源代码，该工件数据对应于源代码的执行；并且基于源代码的更新完成，利用所保留工件数据在源代码内停止执行的特定位置继续源代码的执行。



CN 104115124 A

1. 一种对在多个计算节点上执行的源代码实施更新的方法，所述方法包括：

由计算节点从源代码更新器接收广播更新通知消息，其指示存在针对在所述多个计算节点上执行的所述源代码的更新；

响应于接收到所述更新通知消息，由所述计算机节点在所述计算节点上实施分布式屏障，所述分布式屏障控制针对所述源代码的更新的实施；

基于所述分布式屏障，由所述计算节点在所述源代码内的特定位置停止所述源代码的执行；

基于所述分布式屏障，由所述计算节点在所述计算节点的存储器中适当更新包括保留工件数据的所述源代码，所述工件数据对应于所述源代码的执行；以及

基于所述源代码的所述更新完成，由所述计算节点利用所保留工件数据在所述源代码内停止执行的所述特定位置继续所述源代码的执行。

2. 根据权利要求 1 所述的方法，其中所实施的分布式屏障防止了所述计算节点继续执行所述源代码直至所述多个计算节点中的每个计算节点都已经完成更新所述源代码。

3. 根据权利要求 2 所述的方法，进一步包括：

响应于完成所述源代码的所述更新，由所述计算节点更新源代码在用版本的指示以记录所述计算节点完成了所述源代码的更新；

由所述计算节点从所述源代码更新器接收更新完成消息，其指示所述多个计算节点中的每个计算节点的所述源代码在用版本与所更新源代码的版本相对应；以及

其中继续所述源代码的执行响应于接收到所述更新完成消息。

4. 根据权利要求 1 所述的方法，其中所实施的分布式屏障防止了所述计算节点更新所述源代码直至所述多个计算节点中的每个计算节点都已经指示准备好更新所述源代码。

5. 根据权利要求 4 所述的方法，进一步包括：

响应于接收到所述更新通知消息，由所述计算节点确定所述计算节点是否准备更新所述源代码；

如果所述计算节点准备更新所述源代码，则由所述计算节点传送准备更新消息；以及

由所述计算节点从所述源代码更新器接收立即更新消息，其指示所述多个计算节点中的每个计算节点都准备更新所述源代码；

其中适当更新所述源代码响应于接收到所述立即更新消息；以及

其中继续所述源代码的执行响应于所述计算节点完成所述计算节点处的所述源代码的所述更新。

6. 根据权利要求 5 所述的方法，其中所述更新通知消息是活动消息，其包括用于从所述多个计算节点中的每个计算节点收集准备更新消息的全归约操作。

7. 一种用于对在多个计算节点上执行的源代码实施更新的装置，所述装置包括计算机处理器，操作耦合至所述计算机处理器的计算机存储器，所述计算机存储器具有部署于其中的计算机程序指令，当被所述计算机处理器执行时，所述计算机程序指令使得所述装置执行以下步骤：

由计算节点从源代码更新器接收广播更新通知消息，其指示存在针对在所述多个计算节点上执行的所述源代码的更新；

响应于接收到所述更新通知消息，由所述计算机节点在所述计算节点上实施分布式屏

障，所述分布式屏障控制针对所述源代码的更新的实施；

基于所述分布式屏障，由所述计算节点在所述源代码内的特定位置停止所述源代码的执行；

基于所述分布式屏障，由所述计算节点在所述计算节点的存储器中适当更新包括保留工件数据的所述源代码，所述工件数据对应于所述源代码的执行；以及

基于所述源代码的所述更新完成，由所述计算节点利用所保留工件数据在所述源代码内停止执行的所述特定位置继续所述源代码的执行。

8. 根据权利要求 7 所述的装置，其中所实施的分布式屏障防止了所述计算节点继续执行所述源代码直至所述多个计算节点中的每个计算节点都已经完成更新所述源代码。

9. 根据权利要求 8 所述的装置，进一步包括计算机程序指令，当被所述计算机处理器执行时，所述计算机程序指令使得所述装置执行以下步骤：

响应于完成所述源代码的所述更新，由所述计算节点更新源代码在用版本的指示以记录所述计算节点完成了所述源代码的更新；

由所述计算节点从所述源代码更新器接收更新完成消息，其指示所述多个计算节点中的每个计算节点的所述源代码在用版本与所更新源代码的版本相对应；以及

其中继续所述源代码的执行响应于接收到所述更新完成消息。

10. 根据权利要求 7 所述的装置，其中所实施的分布式屏障防止了所述计算节点更新所述源代码直至所述多个计算节点中的每个计算节点都已经指示准备好更新所述源代码。

11. 根据权利要求 10 所述的装置，进一步包括计算机程序指令，当被所述计算机处理器执行时，所述计算机程序指令使得所述装置执行以下步骤：

响应于接收到所述更新通知消息，由所述计算节点确定所述计算节点是否准备更新所述源代码；

如果所述计算节点准备更新所述源代码，则由所述计算节点传送准备更新消息；以及

由所述计算节点从所述源代码更新器接收立即更新消息，其指示所述多个计算节点中的每个计算节点都准备更新所述源代码；

其中适当更新所述源代码响应于接收到所述立即更新消息；以及

其中继续所述源代码的执行响应于所述计算节点完成所述计算节点处的所述源代码的所述更新。

12. 根据权利要求 11 所述的装置，其中所述更新通知消息是活动消息，其包括用于从所述多个计算节点中的每个计算节点收集准备更新消息的全归约操作。

13. 一种用于对在多个计算节点上执行的源代码实施更新的计算机程序产品，所述计算机程序产品部署在计算机可读介质上，所述计算机程序产品包括计算机程序指令，当被执行时，所述计算机程序指令能够使得计算机执行以下步骤：

由计算节点从源代码更新器接收广播更新通知消息，其指示存在针对在所述多个计算节点上执行的所述源代码的更新；

响应于接收到所述更新通知消息，由所述计算机节点在所述计算节点上实施分布式屏障，所述分布式屏障控制针对所述源代码的更新的实施；

基于所述分布式屏障，由所述计算节点在所述源代码内的特定位置停止所述源代码的执行；

基于所述分布式屏障，由所述计算节点在所述计算节点的存储器中适当更新包括保留工件数据的所述源代码，所述工件数据对应于所述源代码的执行；以及

基于所述源代码的所述更新完成，由所述计算节点利用所保留工件数据在所述源代码内停止执行的所述特定位置继续所述源代码的执行。

14. 根据权利要求 13 所述的计算机程序产品，其中所实施的分布式屏障防止了所述计算节点继续执行所述源代码直至所述多个计算节点中的每个计算节点都已经完成更新所述源代码。

15. 根据权利要求 14 所述的计算机程序产品，其进一步包括计算机程序指令，当被执行时，所述计算机程序指令能够使得计算机执行以下步骤：

响应于完成所述源代码的所述更新，由所述计算节点更新源代码在用版本的指示以记录所述计算节点完成了所述源代码的更新；

由所述计算节点从所述源代码更新器接收更新完成消息，其指示所述多个计算节点中的每个计算节点的所述源代码在用版本与所更新源代码的版本相对应；以及

其中继续所述源代码的执行响应于接收到所述更新完成消息。

16. 根据权利要求 13 所述的计算机程序产品，其中所实施的分布式屏障防止了所述计算节点更新所述源代码直至所述多个计算节点中的每个计算节点都已经指示准备好更新所述源代码。

17. 根据权利要求 16 所述的计算机程序产品，其进一步包括计算机程序指令，当被执行时，所述计算机程序指令能够使得计算机执行以下步骤：

响应于接收到所述更新通知消息，由所述计算节点确定所述计算节点是否准备更新所述源代码；

如果所述计算节点准备更新所述源代码，则由所述计算节点传送准备更新消息；以及

由所述计算节点从所述源代码更新器接收立即更新消息，其指示所述多个计算节点中的每个计算节点都准备更新所述源代码；

其中适当更新所述源代码响应于接收到所述立即更新消息；以及

其中继续所述源代码的执行响应于所述计算节点完成所述计算节点处的所述源代码的所述更新。

18. 根据权利要求 13 所述的计算机程序产品，其中所述更新通知消息是活动消息，其包括用于从所述多个计算节点中的每个计算节点收集准备更新消息的全归约操作。

19. 根据权利要求 13 所述的计算机程序产品，其中所述计算机可读介质是信号承载介质。

20. 根据权利要求 13 所述的计算机程序产品，其中所述计算机可读介质是存储介质。

对在多个计算节点上执行的源代码实施更新

技术领域

[0001] 本发明的领域是数据处理，或者更具体地，是用于对在多个计算机节点上执行的源代码实施更新的方法、装置和计算机程序产品。

背景技术

[0002] 1948 年的 EDVAC 计算机系统的发展经常被称作计算机时代的开始。自那时起，计算机系统已经进化为极其复杂的设备。今天的计算机明显比诸如 EDVAC 的早期系统更为繁杂。计算机系统通常包括硬件组件和软件组件、应用程序、操作系统、处理器、总线、存储器、输入 / 输出设备等的组合。半导体处理和计算机架构的发展将计算机的性能推向越来越高的水平，更为繁杂的计算机软件已经进化为对硬件的更高性能加以利用，这使得今天的计算机系统与仅仅几年前相比也明显更为强大。

[0003] 现代计算系统可以是大量并行的并且在计算系统内包括许多计算节点。相同源代码经常在多个计算节点上执行。为了保持结果的连续性，可能需要同时对所有计算节点上的源代码进行更新。通常的解决方案是利用新代码重新启动所有计算节点并重新加载数据。对于许多系统而言，尽管同时源代码更新是关键的，将源代码重新加载到计算节点中所导致的延迟也是不可接受的。

发明内容

[0004] 提供了用于对在多个计算节点上执行的源代码实施更新的方法、装置和计算机程序产品。实施方式包括由计算节点从源代码更新器接收广播更新通知消息，其指示存在针对在多个计算节点上执行的源代码的更新；响应于接收到该更新通知消息，由计算机节点在计算节点上实施分布式屏障，该分布式屏障控制针对源代码的更新的实施；基于该分布式屏障，由计算节点在源代码内的特定位置停止该源代码的执行；基于该分布式屏障，由计算节点在计算节点的存储器中适当更新包括保留工件数据的源代码，该工件数据对应于源代码的执行；并且基于源代码的更新完成，由计算节点利用所保留工件数据在源代码内停止执行的特定位置继续源代码的执行。

[0005] 本发明的上述和其它目标、特征和优势将由于以下对附图所示的本发明的示例性实施方式更为具体的描述而容易理解，附图中同样的附图标记总体上标识本发明示例性实施方式中同样的部分。

[0006] 从第一方面看到，本发明提供了一种对在多个计算节点上执行的源代码实施更新的方法，该方法包括：由计算节点从源代码更新器接收广播更新通知消息，其指示存在针对在多个计算节点上执行的源代码的更新；响应于接收到该更新通知消息，由计算机节点在计算节点上实施分布式屏障，该分布式屏障控制针对源代码的更新的实施；基于该分布式屏障，由计算节点在源代码内的特定位置停止该源代码的执行；基于该分布式屏障，由计算节点在计算节点的存储器中适当更新包括保留工件数据的源代码，该工件数据对应于源代码的执行；并且基于源代码的更新完成，由计算节点利用所保留工件数据在源代码内停止

执行的特定位置继续源代码的执行。

[0007] 优选地，本发明提供了一种方法，其中所实施的分布式屏障防止了计算节点继续执行源代码直至多个计算节点中的每个计算节点都已经完成更新源代码。

[0008] 优选地，本发明提供了一种方法，其进一步包括：响应于源代码的更新完成，由计算节点更新源代码在用版本的指示以记录计算节点完成了源代码的更新；由计算节点从源代码更新器接收更新完成消息，其指示多个计算节点中的每个计算节点的源代码在用版本与所更新源代码的版本相对应；并且其中继续源代码的执行响应于接收到该更新完成消息。

[0009] 优选地，本发明提供了一种方法，其中所实施的分布式屏障防止了计算节点更新源代码直至多个计算节点中的每个计算节点都已经指示准备好更新源代码。

[0010] 优选地，本发明提供了一种方法，其进一步包括：响应于接收到该更新通知消息，由计算节点确定该计算节点是否准备更新源代码；如果该计算节点准备更新源代码，则由该计算节点传送准备更新消息；并且由计算节点从源代码更新器接收立即更新消息，其指示多个计算节点中的每个计算节点都准备更新源代码；其中响应于接收到该立即更新消息而适当更新源代码；并且其中继续源代码的执行响应于计算节点完成该计算节点处的源代码更新。

[0011] 优选地，本发明提供了一种方法，其中该更新通知消息是活动消息，其包括用于从多个计算节点中的每个计算节点收集准备更新消息的全归约操作(all reduce operation)。

[0012] 从第二方面看到，本发明提供了一种用于对在多个计算节点上执行的源代码实施更新的装置，该装置包括计算机处理器，操作耦合至该计算机处理器的计算机存储器，该计算机存储器具有部署于其中的计算机程序指令，当被计算机处理器执行时，该计算机程序指令使得该装置执行以下步骤：由计算节点从源代码更新器接收广播更新通知消息，其指示存在针对在多个计算节点上执行的源代码的更新；响应于接收到该更新通知消息，由计算机节点在计算节点上实施分布式屏障，该分布式屏障控制针对源代码的更新的实施；基于该分布式屏障，由计算节点在源代码内的特定位置停止该源代码的执行；基于该分布式屏障，由计算节点在计算节点的存储器中适当更新包括保留工件数据的源代码，该工件数据对应于源代码的执行；并且基于源代码的更新完成，由计算节点利用所保留工件数据在源代码内停止执行的特定位置继续源代码的执行。

[0013] 优选地，本发明提供了一种装置，其中所实施的分布式屏障防止了计算节点继续执行源代码直至多个计算节点中的每个计算节点都已经完成更新源代码。

[0014] 优选地，本发明提供了一种装置，其进一步包括计算机程序指令，当被计算机处理器执行时，该计算机程序指令使得该装置执行以下步骤：响应于源代码的更新完成，由计算节点更新源代码在用版本的指示以记录计算节点完成了源代码的更新；由计算节点从源代码更新器接收更新完成消息，其指示多个计算节点中的每个计算节点的源代码在用版本与所更新源代码的版本相对应；并且其中继续源代码的执行响应于接收到该更新完成消息。

[0015] 优选地，本发明提供了一种装置，其中所实施的分布式屏障防止了计算节点更新源代码直至多个计算节点中的每个计算节点都已经指示准备好更新源代码。

[0016] 优选地，本发明提供了一种装置，其进一步包括计算机程序指令，当被计算机处理

器执行时，该计算机程序指令使得该装置执行以下步骤：响应于接收到该更新通知消息，由计算节点确定该计算节点是否准备更新源代码；如果该计算节点准备更新源代码，则由该计算节点传送准备更新消息；并且由计算节点从源代码更新器接收立即更新消息，其指示多个计算节点中的每个计算节点都准备更新源代码；其中适当更新源代码响应于接收到该立即更新消息；并且其中继续源代码的执行响应于计算节点完成该计算节点处的源代码更新。

[0017] 优选地，本发明提供了一种装置，其中该更新通知消息是活动消息，其包括用于从多个计算节点中的每一个收集准备更新消息的全归约操作。

[0018] 从另一个方面看到，本发明提供了一种用于对在多个计算节点上执行的源代码实施更新的计算机程序产品，该计算机程序产品部署在计算机可读介质上，该计算机程序产品包括计算机程序指令，当被执行时，该计算机程序指令能够使得计算机执行以下步骤：由计算节点从源代码更新器接收广播更新通知消息，其指示存在针对在多个计算节点上执行的源代码的更新；响应于接收到该更新通知消息，由计算机节点在计算节点上实施分布式屏障，该分布式屏障控制针对源代码的更新的实施；基于该分布式屏障，由计算节点在源代码内的特定位置停止该源代码的执行；基于该分布式屏障，由计算节点在计算节点的存储器中适当更新包括保留工件数据的源代码，该工件数据对应于源代码的执行；并且基于源代码的更新完成，由计算节点利用所保留工件数据在源代码内停止执行的特定位置继续源代码的执行。

[0019] 优选地，本发明提供了一种计算机程序产品，其中所实施的分布式屏障防止了计算节点继续执行源代码直至多个计算节点中的每个计算节点都已经完成更新源代码。

[0020] 优选地，本发明提供了一种计算机程序产品，其进一步包括计算机程序指令，当被执行时，该计算机程序指令能够使得计算机执行以下步骤：响应于源代码的更新完成，由计算节点更新源代码在用版本的指示以记录计算节点完成了源代码的更新；由计算节点从源代码更新器接收更新完成消息，其指示多个计算节点中的每个计算节点的源代码在用版本与所更新源代码的版本相对应；并且其中继续源代码的执行响应于接收到该更新完成消息。

[0021] 优选地，本发明提供了一种计算机程序产品，其中所实施的分布式屏障防止了计算节点更新源代码直至多个计算节点中的每个计算节点都已经指示准备好更新源代码。

[0022] 优选地，本发明提供了一种计算机程序产品，其进一步包括计算机程序指令，当被执行时，该计算机程序指令能够使得计算机执行以下步骤：响应于接收到该更新通知消息，由计算节点确定该计算节点是否准备更新源代码；如果该计算节点准备更新源代码，则由该计算节点传送准备更新消息；并且由计算节点从源代码更新器接收立即更新消息，其指示多个计算节点中的每个计算节点都准备更新源代码；其中适当更新源代码响应于接收到该立即更新消息；并且其中继续源代码的执行响应于计算节点完成该计算节点处的源代码更新。

[0023] 优选地，本发明提供了一种计算机程序产品，其中该更新通知消息是活动消息，其包括用于从多个计算节点中的每一个收集准备更新消息的全归约操作。

[0024] 优选地，本发明提供了一种计算机程序产品，其中该计算机可读介质是信号承载介质。

[0025] 优选地，本发明提供了一种计算机程序产品，其中该计算机可读介质是存储介质。

附图说明

[0026] 现在将参考附图仅通过示例对本发明的优选实施方式进行描述，其中：

[0027] 图 1 图示了根据本发明实施方式的用于对在多个计算节点上执行的源代码实施更新的示例系统。

[0028] 图 2 给出了根据本发明实施方式的可在能够对多个计算节点上执行的源代码实施更新的并行计算机中使用的示例计算节点的框图。

[0029] 图 3A 给出了根据本发明实施方式的可在用于对多个计算节点上执行的源代码实施更新的系统中使用的示例点对点适配器的框图。

[0030] 图 3B 给出了根据本发明实施方式的可在用于对多个计算节点上执行的源代码实施更新的系统中使用的示例全局组合网络适配器的框图。

[0031] 图 4 给出了图示根据本发明实施方式的示例数据通信网络的线路图，该数据通信网络针对可在能够对多个计算节点上执行的源代码实施更新的系统中使用的点对点操作进行了优化。

[0032] 图 5 给出了图示根据本发明实施方式的可在能够对多个计算节点上执行的源代码实施更新的系统中使用的示例全局组合网络的线路图。

[0033] 图 6 给出了图示根据本发明实施方式的用于对多个计算节点上执行的源代码实施更新的示例性方法的流程图。

[0034] 图 7 给出了图示根据本发明实施方式的用于对多个计算节点上执行的源代码实施更新的另外的示例性方法的流程图。

[0035] 图 8 给出了图示根据本发明实施方式的用于对多个计算节点上执行的源代码实施更新的另外的示例性方法的流程图。

具体实施方式

[0036] 以图 1 开始，参考附图对依据本发明的用于对多个计算节点上执行的源代码实施更新的示例性方法、装置和产品进行描述。

[0037] 图 1 图示了根据本发明实施方式的用于对在多个计算节点上执行的源代码实施更新的示例系统。图 1 的系统包括并行计算机 (100)，用于计算机的数据存储设备 (118) 形式的非易失性存储器，用于计算机的打印机 (120) 形式的输出设备，以及用于计算机的计算机终端 (122) 形式的输入 / 输出设备。

[0038] 图 1 的示例中的并行计算机 (100) 包括多个计算机节点 (102)。计算机节点 (102) 通过若干独立数据通信网络耦合以便进行数据通信，该网络包括高速以太网 (174)、联合测试行动组 (JTAG) 网络 (104)、针对使用二叉树网络拓扑的集体操作进行优化的全局组合网络 (106)，以及针对使用环形网络拓扑的点对点操作进行优化的点对点网络 (108)。全局组合网络 (106) 是包括连接至计算机节点 (102) 的数据新链路以便将计算机节点 (102) 组织为二叉树的数据通信网络。每个数据通信网络利用计算机节点 (102) 之间的数据通信链路而实施。数据通信链路为并行计算机 (100) 的计算机节点 (102) 之间的并行操作提供数据通信。

[0039] 并行计算机(100)的计算机节点(102)被组织为计算节点的至少一个操作群组(132)以便在并行计算机(100)上进行集体操作。计算节点的每个操作群组(132)是在其上执行集体并行操作的计算节点的集合。操作群组(132)中的每个计算节点被分配以识别操作群组(132)中的特定计算节点的唯一等级。集体操作利用操作群组的计算节点之间的数据通信来实施。集体操作是涉及到操作群组(132)的所有计算节点的那些功能。集体操作是一种同时即基本上在同一时间由计算节点的操作群组(132)中的所有计算节点所执行的消息输送计算机程序指令的操作。这样的操作群组(132)可以包括并行计算机(100)中的所有计算节点(102)或者所有计算节点(102)的子集。集体操作经常围绕点对点操作来构建。集体操作要求操作群组(132)内的所有计算节点上的所有处理都调用具有匹配变元(argument)的相同集体操作。“广播”是用于在操作群组的计算节点之间移动数据的集体操作的示例。“缩减”操作是对操作群组(132)的计算节点之间所分布的数据执行算术或逻辑函数的集体操作的示例。操作群组(132)例如可以被实施为MPI“通信方(communicator)”。

[0040] “MPI”是指“消息输送接口”,这是现有技术的并行通信库,用于在并行计算机上进行数据通信的计算机程序指令的模块。可以针对根据本发明实施方式进行配置的系统中的使用而有所改进的现有技术的并行通信库的示例包括MPI和“并行虚拟机”(PVM)库。PVM由田纳西大学、Oak Ridge国家实验室和Emory大学所研发。MPI由MPI论坛所发布,这是具有来自定义和维护MPI标准的许多组织的代表的开放性组织。在本文撰写时,MPI是实际上用于在分布式存储器并行计算机上运行并行程序的计算节点之间进行通信的标准。本说明书有时为了便于解释而使用MPI术语,虽然这样使用MPI并非本发明的要求或限制。

[0041] 一些集体操作具有在操作群组(132)中的特定计算节点上运行的起始或接收处理。例如,在“广播”集体操作中,向所有其它计算节点发布数据的计算节点上的处理就是起始处理。例如,在“收集”操作中,从其它计算节点接收所有数据的计算节点上的处理就是接收处理。在其上运行这样的起始或接收处理的计算节点被称作逻辑根。

[0042] 大多数集体操作是四种基本操作的变化或组合:广播、收集、分散和缩减。用于这些集体操作的接口在MPI论坛所发布的MPI标准中有所定义。然而,MPI标准中并未定义用于执行集体操作的算法。在广播操作中,所有处理指定相同的跟处理,其缓冲内容将被发送。根以外的其它处理指定接收缓冲器。在该操作之后,所有缓冲器都包含来自根处理的消息。

[0043] 与广播操作一样,分散操作也是一对多的集体操作。在分散操作中,逻辑根将根上的数据划分为分段并且向操作群组(132)中的每个计算节点分配以不同的分段。在分散操作中,所有处理通常规定相同的接收计数。发送变元仅对于根处理是重要的,其缓冲器实际上包含给定数据类型的sendcount*N个元素,其中N是给定计算节点群组中的处理的数量。发送缓冲器被划分并分散至所有处理(包括逻辑根上的处理)。每个计算节点被分配以称作“等级”的顺序标识符。在该操作之后,根已经以等级增序向每个处理发送了sendcount数据元素。等级0从发送缓冲器接收第一sendcount数据元素。等级1从发送缓冲器接收第二sendcount数据元素,等等。

[0044] 收集操作是多对一的集体操作,其与分散操作的描述完全相反。即,收集是其中数据类型元素从分级计算节点被收集到根节点中的接收缓冲器的多对一集体操作。

[0045] 缩减操作也是多对一的集体操作,其包括对两个数据元素所执行的算术或逻辑函

数。所有处理指定相同的“计数”以及相同的算术或逻辑函数。在缩减之后，所有处理都已经从计算节点发送缓冲器向根处理发送了计数数据元素。在缩减操作中，来自相对应的发送缓冲器位置的数据元素通过算术或逻辑运算而按对进行组合以在根处理的接收缓冲器中产生单个相对应的元素。在运行时能够定义特定于应用的缩减操作。并行通信库可以支持预定义操作。例如，MPI 提供了以下的预定义缩减操作：

[0046]

MPI_MAX	最大值
MPI_MIN	最小值
MPI_SUM	和值
MPI_PROD	乘积
MPI_LAND	逻辑与

[0047]

MPI_BAND	按位与
MPI_LOR	逻辑或
MPI_BOR	按位或
MPI_LXOR	逻辑异或
MPI_BXOR	按位异或

[0048] 除了计算节点之外，并行计算机 (100) 包括输入 / 输出 (I/O) 节点 (110,114)，其通过全局组合网络 (106) 耦合至计算节点 (102)。并行计算机 (100) 中的计算节点 (102) 可以被划分为处理集合而使得处理集合中的每个计算节点为了进行数据通信而连接至相同的 I/O 节点。因此，每个处理集合由一个 I/O 节点和计算节点 (102) 的子集所组成。整个系统中计算节点的数量与 I/O 节点的数量之间的比率通常取决于并行计算机 (100) 的硬件配置。例如，在一些配置中，每个处理集合可以由 8 个计算节点和 1 个 I/O 节点所组成。在一些其它配置中，每个处理集合可以由 64 个计算节点和 1 个 I/O 节点所组成。然而，这样的示例仅是为了解释而并非用于限制。每个 I/O 节点在前处理集合的计算节点 (102) 与 I/O 设备集合之间提供 I/O 服务。在图 1 的示例中，I/O 节点 (110,114) 通过使用高速以太网而实现的局域网 (LAN)(130) 而连接用于数据通信 I/O 设备 (118,120,122)。

[0049] 图 1 的并行计算机 (100) 还包括通过网络 (104) 之后耦合至计算节点的服务节点 (116)。服务节点 (116) 提供多个计算节点共用的服务，管理计算节点的配置，向计算节点中加载程序，开始计算节点上的程序执行，获取计算节点上的程序操作的结果，等等。服务节点 (116) 运行服务应用 (124) 并且通过在计算机终端 (122) 上运行的服务应用接口 (126) 与用户 (128) 进行通信。服务节点 (116) 还包括源代码更新器 (198)，其被配置为根据本发明的实施方式向多个计算节点中的每个计算节点广播更新通知消息。源代码更新器 (198) 还可以包括所更新源代码以便分布至每个计算节点，或者备选地，所更新源代码可以存储在另一位置以便由计算节点所获取。

[0050] 在特定实施方式中，源代码更新器 (198) 被配置为保持追踪每个计算节点的正在

使用的哪个版本的源代码，并且被配置为确定每个计算节点何时完成更新。响应于确定每个计算节点已经更新了其源代码，源代码更新器 (198) 可以传送更新完成消息。如以下更为详细解释的，响应于接收到该更新完成消息，计算节点可以丢弃分布式屏障并且继续原代码的执行。

[0051] 图 1 的并行计算机 (100) 总体上进行操作以便依据本发明的实施方式对多个计算节点上执行的源代码实施更新。为了对源代码实施更新，多个计算节点 (102) 中的每个计算节点可以包括源代码更新模块 (199)。图 1 的源代码更新模块 (199) 包括计算机程序指令，当被计算节点的处理器执行时，该计算机程序指令使得该计算节点执行步骤：由计算节点从源代码更新器接收广播更新消息，其指示存在针对多个计算节点上执行的源代码的更新。广播更新通知消息是从源代码更新器 (198) 向每个计算节点 (102) 进行广播的消息。该更新通知消息指示可获得针对计算节点上执行的源代码的更新。

[0052] 源代码更新模块 (199) 还包括计算机程序指令，当被计算节点的处理器执行时，该计算机程序指令使得该计算节点执行步骤：响应于接收到该更新通知消息，由计算机节点在计算节点上实施分布式屏障；基于该分布式屏障，由计算节点在源代码内的特定位置停止该源代码的执行；并且基于该分布式屏障，由计算节点对源代码进行适当更新。分布式屏障是一种对针对源代码的更新的实施进行控制的操作。即，分布式屏障用作所有计算节点的停止或集合点以在更新处理中继续特定步骤之前进行收集。停止电的位置可以基于更新的类型、所更新源代码的类型、计算节点的配置以及计算节点所执行的处理或应用的类型进行设置。例如，在一种配置中，分布式屏障可以在更新之后但是继续执行源代码之前的点停止所有计算节点的更新处理。在该示例中，每个计算节点都将在任意节点被分布式屏障允许继续执行源代码之前对源代码进行了更新。作为另一个示例，在另一种配置中，分布式屏障可以在所有计算节点都已经开始更新源代码之前的点停止更新处理。在该示例中，计算节点将指示它们何时准备开始更新源代码并且所有计算节点都将进行等待直至每个计算节点都准备进行更新。

[0053] 用于由计算节点基于分布式屏障对源代码进行适当更新的计算机程序代码包括在计算节点的存储器中保留工件 (workpiece) 数据。工件数据的示例可以包括在源代码的执行期间所生成的数据；变量状态、指针、索引以及本领域技术人员将会意识到的前提执行数据。

[0054] 源代码更新模块 (199) 还包括计算机程序指令，当被计算节点的处理器执行时，该计算机程序指令使得该计算节点执行步骤：基于源代码的更新完成，由计算节点利用所保留工件数据在源代码内停止执行的特定位置继续源代码的执行。即，源代码更新模块被配置为允许计算节点直接在计算节点停止更新源代码的地方继续执行。

[0055] 根据本发明实施方式的对多个计算节点上所执行的源代码实施更新通常在并行计算机上实施，后者包括通过至少一个数据通信网络而组织用于集体操作的多个计算节点。实际上，这样的计算机可能包括数千个这样的计算节点。每个计算节点自身进而是一种由一个或多个计算机处理内核、其自己的存储及其自己的输入 / 输出适配器所组成的计算机。

[0056] 为了进一步进行揭示，图 2 给出了根据本发明实施方式的可在能够对多个计算节点上执行的源代码实施更新的并行计算机中使用的示例计算节点 (102) 的框图。图 2 的

计算节点 (102) 包括多个处理内核 (165) 以及 RAM(156)。图 2 的处理内核 (165) 可以在一个或多个集成电路组模上进行配置。处理内核 (165) 通过高速存储器总线 (155) 连接至 RAM(156)，并且通过总线适配器 (194) 和扩展总线 (168) 连接至计算节点的其它组件。RAM(156) 中存储有应用程序 (159)，这是使用并行算法执行并行的用户级数据处理的计算机程序指令的模块。

[0057] RAM(156) 中还存储并行通信库 (161)，这是在计算节点之间执行包括点对点操作以及集合操作在内的并行通信的计算机程序指令的库。使用诸如 C 编程语言的传统编程语言，并且使用传统编程方法编写在两个独立数据通信网络上的节点之间发送和接收数据的并行通信例程，可以重新研发并行通信例程的库以便在根据本发明实施方式的系统中使用。备选地，可以对现有技术的库进行改进以根据本发明的实施方式进行操作。现有技术的并行通信库的示例包括“消息输送接口”(MPI) 库和“并行虚拟机”(PVM) 库。

[0058] RAM(156) 中还存储应用 (226)。图 2 的示例中的应用 (226) 可以被配置为并行应用，其具有在被组织为操作群组的多个计算节点之间执行的其它实例。图 2 的示例中的应用 (226) 被配置为用于依据本发明的实施方式对多个计算节点上执行的源代码实施更新。图 2 的示例中的应用 (226) 能够通过执行以下步骤而依据本发明的实施方式对多个计算节点上执行的源代码实施更新：由计算节点从源代码更新器接收广播更新通知消息，其指示存在针对在多个计算节点上执行的源代码的更新；响应于接收到该更新通知消息，由计算节点在计算节点上实施分布式屏障，该分布式屏障控制针对源代码的更新的实施；基于该分布式屏障，由计算节点在源代码内的特定位置停止该源代码的执行；基于该分布式屏障，由计算节点在计算节点的存储器中适当更新包括保留工件数据的源代码，该工件数据对应于源代码的执行；并且基于源代码的更新完成，由计算节点利用所保留工件数据在源代码内停止执行的特定位置继续源代码的执行。

[0059] RAM(156) 中还存储操作系统 (162)，这是用于应用程序对计算节点的其它资源进行访问的计算机程序指令和例程的模块。对于并行计算机的计算节点中的应用程序和并行通信库而言，通常运行没有用户登录且没有安全问题的单个执行线程，这是因为该线程有权完成对节点的所有资源的访问。因此与具有同时运行的许多现成的串行计算机上的操作系统相比，并行计算机中的计算节点上的操作系统所要执行的任务的数量更小且复杂度更低。此外，在图 2 的计算节点 (102) 上没有视频 I/O，这是降低了对操作系统的需求的另一个因素。因此，与通用计算机的操作系统或者为特定并行计算机上的操作专门研发的操作系统相比较，就如同其精简版本一样，操作系统 (162) 可能是非常轻量级的。为了在计算节点中使用而在可用性方面进行改进、简化的操作系统包括 UNIX™、Linux™、Windows XP™、AIX™、IBM 的 i5/OS™，以及本领域技术人员将会意识到的其它操作系统。

[0060] 图 2 的示例计算节点 (102) 包括多个通信适配器 (172,176,180,188)，它们用于与并行计算机的前提节点实施数据通信。这样的通信可以通过 RS-232 连接、通过诸如 USB 的外部总线、通过诸如 IP 网络的数据通信网络以及本领域技术人员将会意识到的其它方式串行执行。通信适配器实现了数据通信的硬件层面，一个计算机通过其直接或通过网络向另一个计算机发送数据通信。可在能够对多个计算节点上执行的源代码实施更新的装置中使用的通信适配器的示例包括用于有线连接的调制解调器、用于有线网络通信的以太网 (IEEE802.3) 适配器以及用于无线网络通信的 802.11b 适配器。

[0061] 图 2 的示例中的数据通信适配器包括千兆比特以太网适配器 (172)，其耦合示例数据节点 (102) 以便针对千兆比特以太网 (174) 进行数据通信。千兆比特以太网是 IEEE802.3 标准中所定义的网络传输标准，其提供了每秒钟十亿比特 (1 千兆比特) 的数据速率。千兆比特以太网是在多模式光纤线缆、单模式光纤线缆或无屏蔽双绞线上进行操作的以太网的变化形式。

[0062] 图 2 的示例中的数据通信适配器包括 JTAG 从属电路 (176)，其耦合示例数据节点 (102) 以便针对 JTAG 主控电路 (178) 进行数据通信。JTAG 是用于 IEEE1149.1 标准所命名的标准测试访问端口和边界扫描架构的常用名称，其用于使用边界扫描来测试印刷电路板。JTAG 的使用非常广泛而使得目前边界扫描或多或少地与 JTAG 同义。JTAG 不仅被用于印刷电路板，而且被用于进行集成电路的边界扫描，并且还可被用作调试嵌入式系统的机制，其向系统中提供了便利的可替换访问点。图 2 的示例计算节点可以这些中的全部三种：且通常包括安装在印刷电路板上的一个或多个集成电路，并且可以被实施为具有其自己的处理内核、其自己的存储器及其自己的 I/O 能力的嵌入式系统。通过 JTAG 从属 (176) 进行的 JTAG 边界扫描可以有效地配置计算节点 (102) 中的处理内核寄存器和存储器以便指向可在根据本发明实施方式的能够对多个计算节点上执行的源代码实施更新的系统中使用的计算节点的模块动态地重新分配所连接节点时使用。

[0063] 图 2 的示例中的数据通信适配器包括点对点网络适配器 (180)，其耦合示例计算节点 (102) 以便针对对于点对点消息输送操作而言最优的网络 (108) 进行数据通信，该网络例如是被配置为三维环形或网格的网络。通过六条双线链路：+x(181)、-x(182)，+y(183)，-y(184)、+z(185) 和 -z(186)，点对点适配器 (180) 在三个通信轴 x、y 和 z 上以六个方向提供数据通信。

[0064] 图 2 的示例中的数据通信适配器包括全局组合网络适配器 (188)，其耦合示例计算节点 (102) 以便针对对于集体消息输送操作而言最优的全局组合网络 (106) 进行数据通信，该网络例如是被配置为二叉树的网络。全局组合网络适配器 (188) 通过全局组合网络适配器 (188) 所支持的每个全局组合网络 (106) 的三条双向链路提供数据通信。在图 2 的示例中，全局组合网络适配器 (188) 通过全局组合网络 (106) 的三条双向链路提供数据通信：两条指向子节点 (190) 而一条指向父节点 (192)。

[0065] 示例计算节点 (102) 包括多个算术逻辑单元 (ALU)。每个处理内核 (165) 包括 ALU(166)，并且单独的 ALU(170) 专用于全局组合网络适配器 (188) 的独有用途，以便在执行缩减操作的包括全归约操作在内的算术和逻辑功能时使用。并行通信库 (161) 总的缩减例程的计算机程序指令可以将算术或逻辑功能的指令锁存到指令寄存器 (169) 中。例如，当缩减操作的算术或逻辑功能是“和”或“逻辑或”时，集体操作适配器 (188) 可以通过使用处理内核 (165) 中的 ALU(166) 来执行该算术或逻辑运算，或者通常更快地，通过使用专用 ALU(170) 而使用全局组合网络 (106) 上的节点 (190,192) 所提供的数据以及计算节点 (102) 上的处理内核 (165) 所提供的数据来执行。

[0066] 然而，当在全局组合网络适配器 (188) 中执行算术运算时，全局组合网络适配器 (188) 经常仅用于对从子节点 (190) 所接收的数据进行组合并且将结果经网络 (106) 向上送给父节点 (192)。类似地，全局组合网络适配器 (188) 可以仅用来传送从父节点 (192) 所接收的数据并且将该数据经网络 (106) 向下送给子节点 (190)。即，计算节点 (102) 上的处

理内核 (165) 都不贡献数据和改变 ALU(170) 的输出，后者随后经全局组合网络 (106) 向上或向下进行输送。由于 ALU(170) 在其接收到来自处理内核 (165) 之一的输入之前通常并不向网络 (106) 上输出任何数据，所以处理内核 (165) 可以为了在 ALU(170) 中执行特定算术预案算以便防止 ALU(170) 的输出改变而将标识元素注入到专用 ALU(170) 中。然而，将标识元素注入到 ALU 中经常耗用许多处理周期。为了在这种情况下进一步提升性能，示例计算节点 (102) 包括用于向 ALU(170) 注入标识元素以减少防止改变 ALU 输出所需的处理内核资源数量的专用硬件 (171)。专用硬件 (171) 注入与 ALU 所执行的特定算术运算相对应的标识元素。例如，当全局组合网络适配器 (188) 对从子节点 (190) 所接收的数据执行按位 OR 时，专用硬件 (171) 可以向 ALU(170) 中注入零以提高整个全局组合网络 (106) 的性能。

[0067] 为了进一步进行解释，图 3A 给出了根据本发明实施方式的可在能够对多个计算节点上执行的源代码实施更新的系统中使用的示例点对点适配器 (180) 的框图。点对点适配器 (180) 被设计为用于在对于点对点操作而言最优的数据通信网络中使用，这是将计算节点以三维环形或网格进行组织的网络。往来于 -x 方向中的下一个节点 (182) 以及往来于 +x 方向中的下一个节点 (181)，图 3A 的示例中的点对点适配器 (180) 通过四条双向数据通信链路而沿 x 轴提供数据通信。往来于 -y 方向中的下一个节点 (184) 以及往来于 +y 方向中的下一个节点 (183)，图 3A 的示例中的点对点适配器 (180) 还通过四条双向数据通信链路而沿 y 轴提供数据通信。往来于 -z 方向中的下一个节点 (186) 以及往来于 +z 方向中的下一个节点 (185)，图 3A 的示例中的点对点适配器 (180) 还通过四条双向数据通信链路而沿 z 轴提供数据通信。

[0068] 为了进一步进行解释，图 3B 给出了根据本发明实施方式的可在能够对多个计算节点上执行的源代码实施更新的系统中使用的全局组合网络适配器 (188) 的框图。全局组合网络适配器 (188) 被设计为在针对集体操作而言最后的网络中使用，这是以二叉树对并行计算机的计算节点进行组织的网络。图 3B 的示例中的全局组合网络适配器 (188) 通过四条双向数据通信链路 (190) 而往来于全局组合网络的子节点提供数据通信，并且还通过两条双向数据通信链路 (192) 而往来于全局组合网络的父节点提供数据通信。

[0069] 为了进一步进行解释，图 4 给出了图示根据本发明实施方式的示例数据通信网络 (108) 的线路图，该数据通信网络 (108) 针对可在能够对多个计算节点上执行的源代码实施更新的系统中使用的点对点操作进行了优化。在图 4 的示例中，点表示并行计算机的计算节点 (102)，而点之间的虚线则表示计算节点之间的数据通信链路 (103)。该数据通信链路利用类似于针对图 3A 中的示例所图示的点对点数据通信适配器所实施，其具有三条轴线 x、y 和 z 上且往来于六个方向 +x(181)、-x(182)，+y(183)，-y(184)、+z(185) 和 -z(186) 的数据通信链路。该链路和计算节点通过针对点对点操作进行优化的该数据通信网络而被组织为三位网格 (105)。网格 (105) 在每条轴线上具有环绕链路，其连接网格 (105) 相反侧上的网格 (105) 中的最外侧的计算节点。这些环绕链路形成环形 (107)。该环形中的每个计算节点在该环形中具有由 x、y、z 坐标集合所唯一指定的位置。读者将会注意到，为了清楚已经省略了 y 和 z 方向中的环绕链路，但是它们与 x 方向所示的环绕链路以类似方式进行那个配置。为了图示的清楚，图 4 的数据通信网络仅利用 27 个计算节点进行图示，但是读者将会认识到，针对点对点操作进行优化以便在依据本发明实施方式而对多个计算节点

上执行的源代码实施更新时使用的数据通信网络可以仅包含几个通信节点或者可以包含数千个通信节点。为了便于图示，图 4 的数据通信网络仅利用三个维度进行图示，但是读者将会认识到，针对点对点操作进行优化以便在依据本发明实施方式而对多个计算节点上执行的源代码实施更新时使用的数据通信网络实际上可以以两个维度、四个维度、五个维度等来实施。若干超级计算机现在使用五维网格或环形网络，例如可 IBM 的 Blue Gene Q™。

[0070] 为了进一步进行解释，图 5 给出了图示根据本发明实施方式的可在能够对多个计算节点上执行的源代码实施更新的系统中使用的示例全局组合网络 (106) 的线路图。图 5 的示例数据通信网络包括连接至计算节点以便将计算节点组织为树形的数据通信链路 (103)。在图 5 的示例中，点表示并行计算机的计算节点 (102)，而点之间的虚线则表示计算节点之间的数据通信链路 (103)。该数据通信链路利用类似于图 3B 中的示例所图示的全局组合网络适配器所实施，其中每个节点通常提供往来于两个子节点的数据通信以及往来于一个父节点的数据通信，也具有一些例外情况。注意到，在全局组合网络 (106) 中，其特征可以是物理根节点 (202)、分支节点 (204) 和叶节点 (206)。物理根 (202) 具有两个子但是没有父，并且被这样称呼是因为物理根节点 (202) 是在物理上配置于二叉树的顶端的节点。叶节点 (206) 均具有父，但是叶节点没有子。分支节点 (204) 均具有父和两个子。链路和计算节点因此被针对集体操作所优化的该数据通信网络组织为二叉树 (106)。为了图示的清楚，图 5 的数据通信网络仅利用 31 个计算节点进行图示，但是读者将会认识到，针对集体操作进行优化以便在依据本发明实施方式而对多个计算节点上执行的源代码实施更新时使用的全局组合网络 (106) 可以仅包含几个通信节点或者可以包含数千个通信节点。

[0071] 在图 5 的示例中，树中的每个节点被分配以称之为“等级”的单元标识符 (250)。该等级实际上标示了执行根据本发明实施方式的并行操作的任务或处理。假设仅有一个这样的任务在每个节点上执行而使用等级来识别节点。针对多于一个的参与任务在单个节点上执行的情况，等级由此识别任务而不是节点。等级唯一地识别任务在树形网络中的位置以便在该树形网络中的点对点和集体操作中使用。该示例中的等级被分配为从分配给根任务或根节点 (202) 的 0，分配给该树中第二层中的第一节点的 1，分配给该树中第二层中的第二节点的 2，分配给该树中第三层中的第一节点的 3，分配给该书中第三层中的第二节点的 4 等开始的整数。为了便于图示，这里仅示出了该树的前三层中的等级，但是树形网络中的所有计算节点都被分配以唯一等级。

[0072] 为了进一步进行解释，图 6 给出了图示根据本发明实施方式的用于对多个计算节点上执行的源代码实施更新的示例性方法的流程图。图 6 的方法包括由分布式处理系统的计算节点 (102a) 从源代码更新器 (198) 接收 (602) 广播更新通知消息 (650)，其指示存在针对在多个计算节点 (102) 上执行的源代码 (280) 的更新。在图 6 的示例中，该更新通知消息是活动消息，其包括用于从多个计算节点中的每一个收集准备更新消息的全归约操作。由分布式处理系统的计算节点 (102a) 从源代码更新器 (198) 接收 (602) 示存在针对在多个计算节点 (102) 上执行的源代码 (280) 的更新的广播更新通知消息 (650) 可以通过经由耦合服务节点 (116) 和计算节点 (102) 的联合测试行动组 (JTAG) 网络接收消息来实现。

[0073] 图 6 的方法还包括响应于接收到该更新通知消息 (650)，由计算节点 (102a) 在计算节点 (102a) 上实施 (604) 分布式屏障 (652)。该分布式屏障 (652) 控制针对源代码 (280)

的更新的实施。由计算节点 (102a) 在计算节点 (102a) 上实施 (604) 分布式屏障 (652) 可以通过执行屏障操作并且执行系统中断来实现。

[0074] 图 6 的方法包括基于该分布式屏障 (652)，由计算节点 (102a) 在源代码 (280) 内的特定位置 (654) 停止 (606) 该源代码 (280) 的执行。由计算节点 (102a) 基于该分布式屏障 (652) 在源代码 (280) 内的特定位置 (654) 停止 (606) 该源代码 (280) 的执行可以通过停止源代码的执行来实现。

[0075] 图 6 的方法还包括基于该分布式屏障 (652)，由计算节点 (102a) 在计算节点 (102a) 的存储器 (156) 中适当更新 (608) 包括保留工件数据 (656) 的源代码 (280)。在图 6 的示例中，工件数据 (656) 对应于源代码 (280) 的执行。由计算节点 (102a) 基于该分布式屏障 (652) 在计算节点 (102a) 的存储器 (156) 中适当更新 (608) 包括保留工件数据 (656) 的源代码 (280) 可以通过存储与源代码执行相关联的变量、指针、寄存器内容；并且存储最后源代码执行的指针至；从源代码更新器 (198) 或另一位置获取所更新源代码；并且将所更新源代码存储在计算节点内来实现。

[0076] 图 6 的方法还包括基于源代码 (280) 的更新完成，由计算节点 (102a) 利用所保留工件数据 (656) 在源代码 (280) 内停止执行的特定位置 (654) 继续 (610) 源代码 (280) 的执行。由计算节点 (102a) 基于源代码 (280) 的更新完成利用所保留工件数据 (656) 在源代码 (280) 内停止执行的特定位置 (654) 继续 (610) 源代码 (280) 的执行可以通过从存储获取指针、寄存器内容、变量和其它工件数据；获取最后的源代码执行的位置；并且在所存储位置继续执行来实现。

[0077] 为了进一步进行解释，图 7 给出了图示根据本发明实施方式的用于对多个计算节点上执行的源代码实施更新的另外的示例性方法的流程图。图 7 的方法与图 6 的方法的相似之处在于，图 7 的方法包括接收 (602) 广播更新通知消息 (650)；响应于接收到该更新通知消息 (650) 在计算节点 (102a) 上实施 (604) 分布式屏障 (652)；基于该分布式屏障 (652) 在源代码 (280) 内的特定位置 (654) 停止 (606) 该源代码 (280) 的执行；基于该分布式屏障 (652) 对源代码 (280) 进行适当更新 (608)；并且基于源代码 (280) 的更新完成在源代码 (280) 内停止执行的特定位置 (654) 继续 (610) 源代码 (280) 的执行。

[0078] 在图 7 的示例中，所实施的分布式屏障防止了计算节点继续执行源代码直至多个计算节点中的每个计算节点都已经完成了更新源代码。图 7 的方法还包括响应于完成源代码 (280) 的更新 (608)，由计算节点 (102a) 更新 (702) 源代码的在用版本的指示 (780) 以记录计算节点 (102a) 完成了源代码 (280) 的更新 (608)。源代码的在用版本的指示是指计算节点正在执行哪个版本的源代码。由计算节点 (102a) 响应于完成源代码 (208) 的更新 (608) 更新 (702) 源代码的在用版本的指示 (780) 以记录计算节点 (102a) 完成了源代码 (280) 的更新 (608) 可以通过向源代码更新器 (198) 发送消息以改变对应于计算节点的源代码版本来实现。即，计算节点告知源代码更新器完成了原代码的更新。

[0079] 图 7 的方法包括由计算节点 (102a) 从源代码更新器 (198) 接收 (704) 更新完成消息 (752)，其指示多个计算节点 (102) 中的每个计算节点的源代码在用版本与所更新源代码的版本相对应。由计算节点 (102a) 从源代码更新器 (198) 接收 (704) 指示多个计算节点 (102) 中的每个计算节点的源代码在用版本与所更新源代码的版本相对应的更新完成消息 (752) 可以通过从源代码更新器 (198) 接收每个计算节点具有最新源代码并且因此

可以开始继续执行的指示而实现。

[0080] 在图 8 的方法中，基于源代码 (280) 的更新完成在源代码 (280) 内停止执行的特定位置 (654) 继续 (610) 源代码 (280) 的执行包括响应于接收到更新完成消息 (752) 而继续 (706) 源代码的执行。响应于接收到更新完成消息 (752) 而继续 (706) 源代码的执行可以通过从所存储的起始位置开始执行所更新源代码来实现。

[0081] 为了进一步进行解释，图 8 给出了图示根据本发明实施方式的用于对多个计算节点上执行的源代码实施更新的另外的示例性方法的流程图。图 8 的方法与图 6 的方法的相似之处在于，图 7 的方法包括接收 (602) 广播更新通知消息 (650) ；响应于接收到该更新通知消息 (650) 在计算节点 (102a) 上实施 (604) 分布式屏障 (652) ；基于该分布式屏障 (652) 在源代码 (280) 内的特定位置 (654) 停止 (606) 该源代码 (280) 的执行 ；基于该分布式屏障 (652) 对源代码 (280) 进行适当更新 (608) ；并且基于源代码 (280) 的更新完成利用所保留工件数据 (656) 在源代码 (280) 内停止执行的特定位置 (654) 继续 (610) 源代码 (280) 的执行。

[0082] 在图 8 的示例中，所实施的分布式屏障防止了计算节点 (102a) 更新 (608) 源代码 (280) 直至多个计算节点 (102) 中的每个计算节点都已经指示准备好更新源代码。图 8 的方法包括响应于接收到该更新通知消息 (852)，由计算节点 (102a) 确定 (802) 该计算节点是否准备更新源代码 (280)。由计算节点 (102a) 响应于接收到该更新通知消息 (852) 而确定 (802) 该计算节点是否准备更新源代码 (280) 可以通过由计算节点确定源代码的执行是否可中断 ；并且当计算节点准备更新时，由该计算节点向源代码更新器 (198) 传送消息来实现。

[0083] 在图 8 的方法中，如果该计算节点准备更新源代码，则该方法以由该计算节点 (102a) 传送 (804) 准备更新消息 (850) 而继续进行。由计算节点 (102a) 传送 (804) 准备更新消息 (850) 可以通过在 JTAG 网络 (104) 上向源代码更新器 (198) 传送消息来实现。

[0084] 图 8 的方法还包括由计算节点 (102a) 确定 (805) 是否接收到立即更新消息 (852)。由计算节点 (102a) 确定 (805) 是否接收到立即更新消息 (852) 可以通过查询存储立即更新消息的队列以确定是否接收到立即更新消息来实现。

[0085] 图 8 的方法包括由计算节点 (102a) 从源代码更新器 (198) 接收立即更新消息 (852)，其指示多个计算节点 (102) 中的每个计算节点都准备更新源代码。由计算节点 (102a) 从源代码更新器 (198) 接收指示多个计算节点 (102) 中的每个计算节点都准备更新源代码的立即更新消息 (852) 可以通过从源代码更新器接收指示所有计算节点都准备进行更新的消息来实现。

[0086] 在图 8 的方法中，对源代码进行适当更新 (608) 包括响应于接收到该立即更新消息 (852) 而适当更新 (808) 源代码。即，在图 8 的示例中，在计算节点 (102a) 响应于该计算节点 (102a) 接收到立即更新消息 (852) 而进行源代码的更新。响应于接收到立即更新消息 (852) 适当更新 (808) 源代码可以通过存储与源代码的执行相关联的变量、指针、寄存器内容 ；并且存储源代码内最后知性的指针位置 ；从源代码更新器 (198) 或另一个位置获取所更新源代码 ；并且在计算节点内存储所更新源代码来实现。

[0087] 在图 8 的方法中，基于源代码 (280) 的更新完成在源代码 (280) 内停止执行的特定位置 (654) 继续 (610) 源代码 (280) 的执行包括由计算节点响应于计算节点处源代码

的更新而继续 (810) 源代码的执行。由计算节点响应于计算节点处源代码的更新而继续 (810) 源代码的执行可以通过从所存储的起始位置开始所更新源代码的执行来实现。

[0088] 本发明的示例性实施方式主要以用于对多个计算节点上执行的源代码实施更新的全功能计算机系统为背景进行了描述。然而，本领域技术人员将会认识到，本发明还可以以部署在计算机可读存储媒体上以便随人意适当处理系统使用的计算机程序产品而得以体现。如本领域技术人员将会意识到的，这样的计算机可读存储媒体可以是用于机器可读信息的任意存储媒体，包括磁性媒体、光学媒体或其它适当媒体。这样的媒体的示例包括硬盘或碟片中的磁盘、用于光驱的紧致盘、磁带等。本领域技术人员将会立刻认识到的是，具有适当编程工具的任意计算机系统都将能够执行以计算机程序产品所体现的本发明的方法的步骤。本领域技术人员还将会认识到，虽然一些示例性实施方式在该说明书中被描述为面向于安装在计算机硬件上并在其上执行的软件，但是被实施为固件或硬件的可替换实施方式同样处于本发明的范围之内。

[0089] 所属技术领域的技术人员知道，本发明的各个方面可以实现为系统、方法或计算机程序产品。因此，本发明的各个方面可以具体实现为以下形式，即：完全的硬件实施方式、完全的软件实施方式（包括固件、驻留软件、微代码等），或硬件和软件方面结合的实施方式，这里可以统称为“电路”、“模块”或“系统”。此外，在一些实施方式中，本发明的各个方面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。

[0090] 可以采用一个或多个计算机可读介质的任意组合。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。

[0091] 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。

[0092] 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、有线、光缆、RF 等等，或者上述的任意合适的组合。

[0093] 可以以一种或多种程序设计语言的任意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++ 等，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）。

[0094] 以上将参照根据本发明实施方式的方法、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，使得这些计算机程序指令在通过计算机或其它可编程数据处理装置的处理器执行时，产生了实现流程图和/或框图中的一个或多个方框中规定的功能/动作的装置。

[0095] 也可以把这些计算机程序指令存储在计算机可读介质中，这些指令使得计算机、其它可编程数据处理装置、或其他设备以特定方式工作，从而，存储在计算机可读介质中的指令就产生出包括实现流程图和/或框图中的一个或多个方框中规定的功能/动作的指令的制造品 (article of manufacture)。

[0096] 计算机程序指令还可以被加载到计算机、其它可编程数据处理装置或者其它设备上而使得一系列操作步骤得以在该计算机、其它可编程数据处理装置或者其它设备上执行，以使得在计算机或其它可编程装置上执行的指令提供用于实施流程图和/或框图的一个或多个框中所指定的功能/动作的处理。

[0097] 附图中的流程图和框图显示了根据本发明的多个实施方式的系统、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动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。

[0098] 从以上描述将要理解的是，可以在本发明的各个实施方式中进行修改和变化而并不背离其实际精神。该说明书中的描述仅是处于说明的目的而并非以限制的含义进行理解。本发明的范围仅由以下权利要求的语言进行限定。

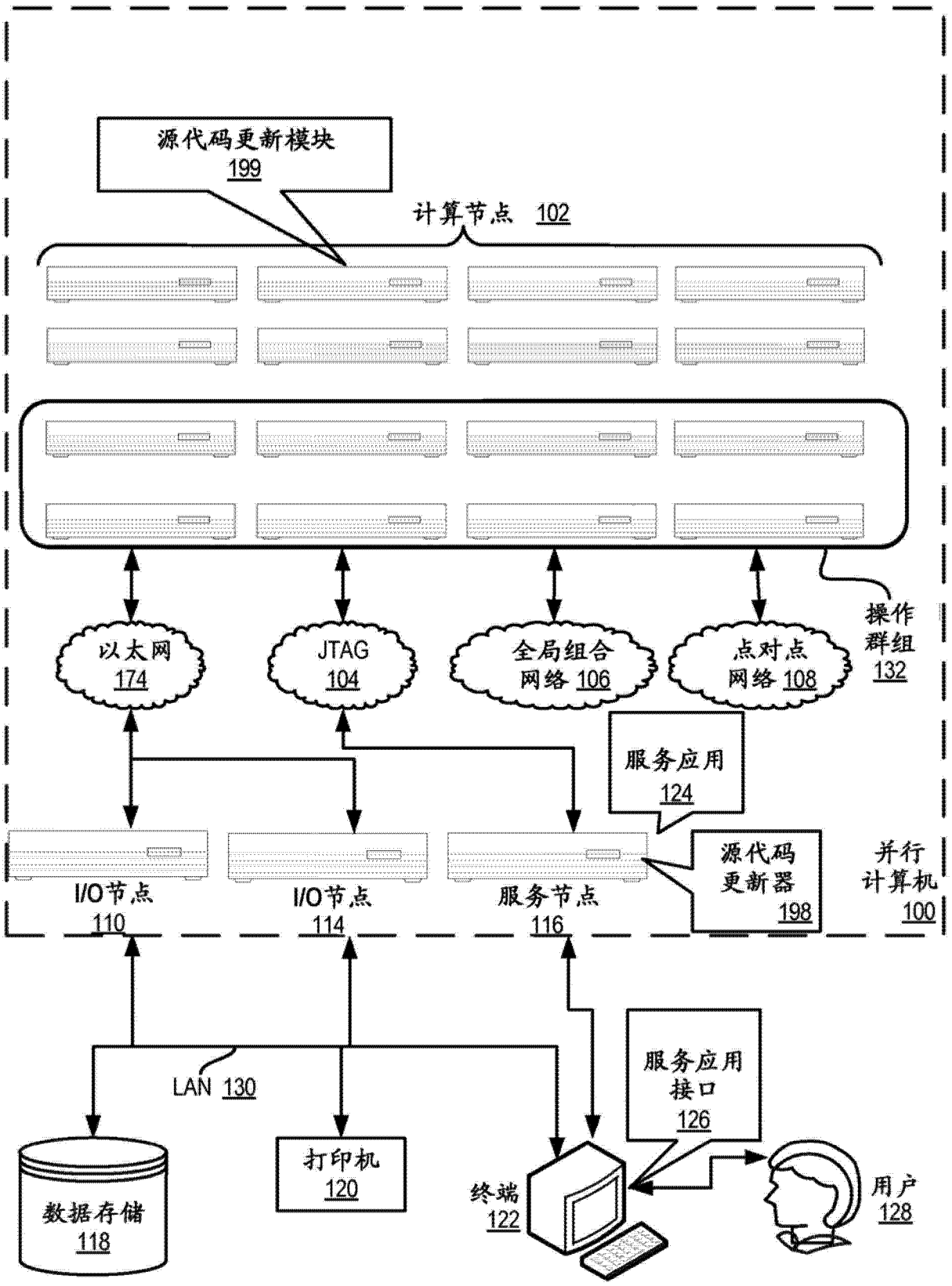

源代码更新模块
199
计算节点 102
操作
群组
132
以太网
174
JTAG
104
全局组合
网络 106
点对点
网络 108
服务应用
124
源代码
更新器
198
并行
计算机
100
I/O节点
110
I/O节点
114
服务节点
116
LAN 130
服务应用
接口
126
数据存储
118
打印机
120
终端
122
用户
128

图 1

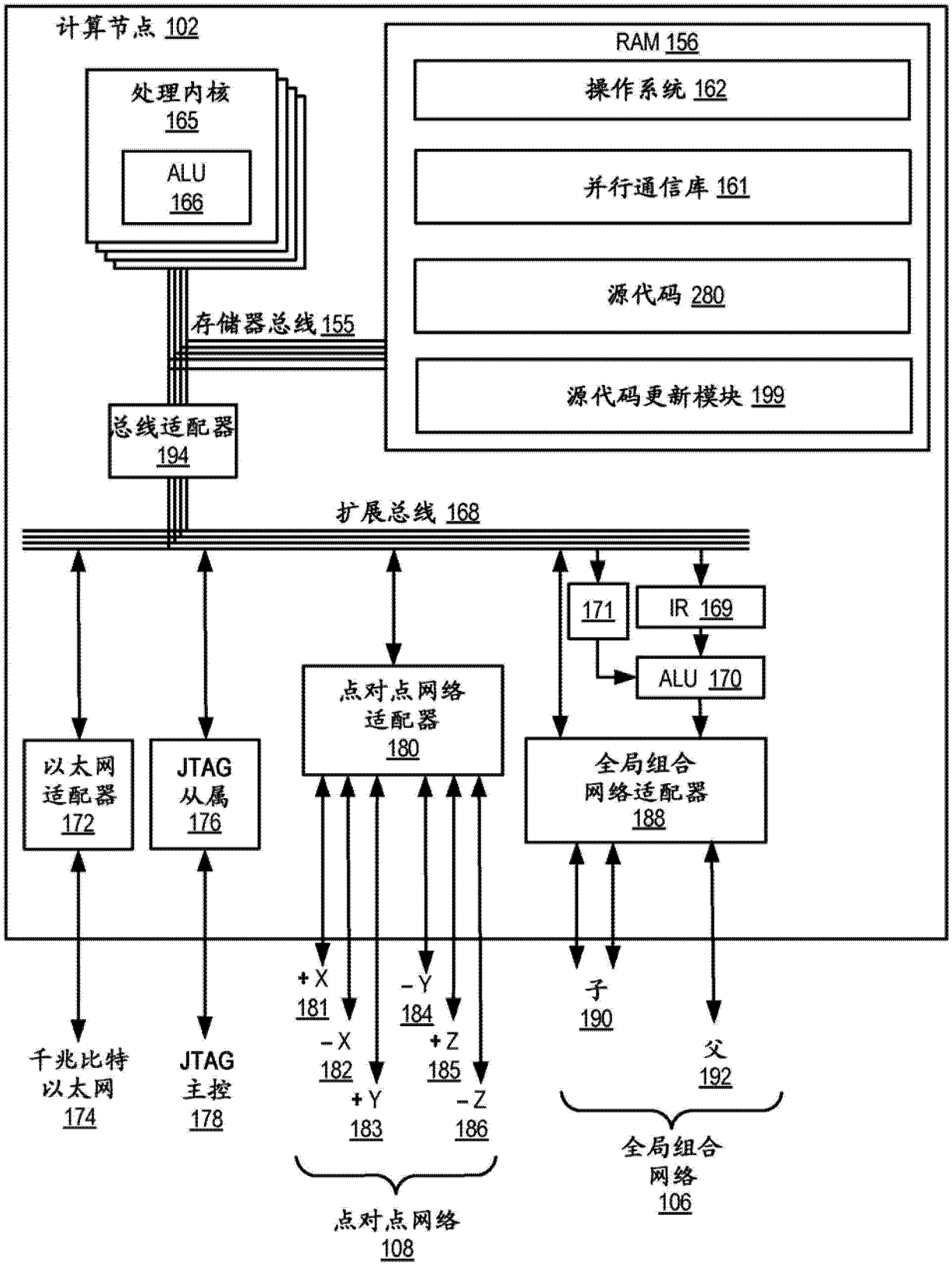

计算节点 102
RAM 156
处理内核 165
ALU 166
操作系统 162
并行通信库 161
源代码 280
存储器总线 155
源代码更新模块 199
总线适配器 194
扩展总线 168
171
IR 169
ALU 170
点对点网络适配器 180
以太网适配器 172
JTAG 从属 176
全局组合网络适配器 188
+X 181
-X 182
+Y 183
-Y 184
+Z 185
-Z 186
子 190
父 192
千兆比特以太网 174
JTAG 主控 178
点对点网络 108
全局组合网络 106

图 2

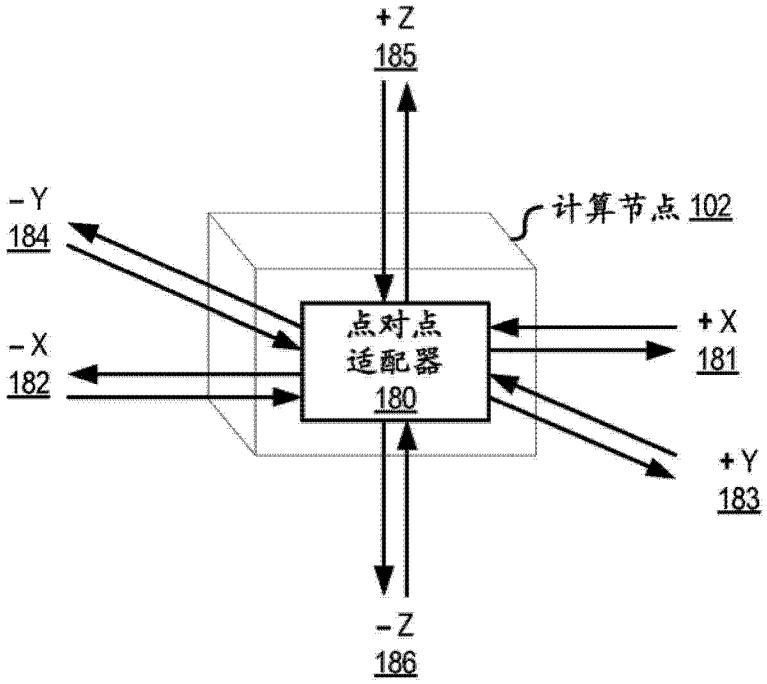

+Z
185
-Y
184
计算节点 102
点对点
适配器
180
+X
181
-X
182
+Y
183
-Z
186

图 3A

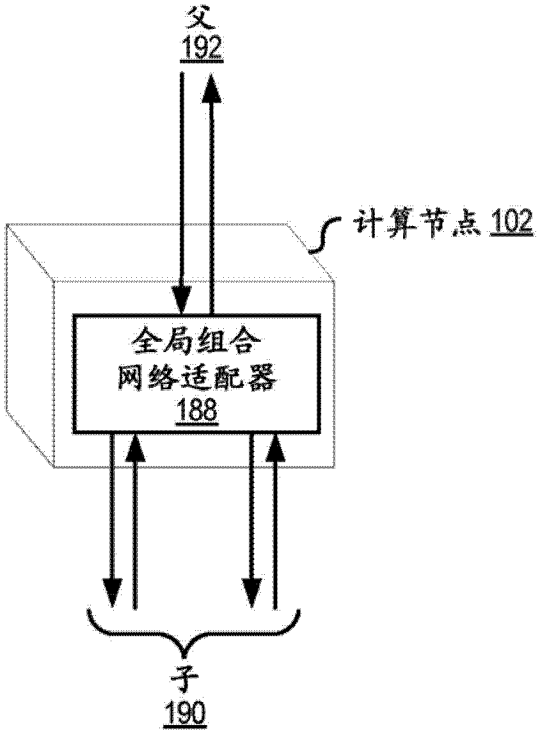

父
192
计算节点 102
全局组合
网络适配器
188
子
190

图 3B

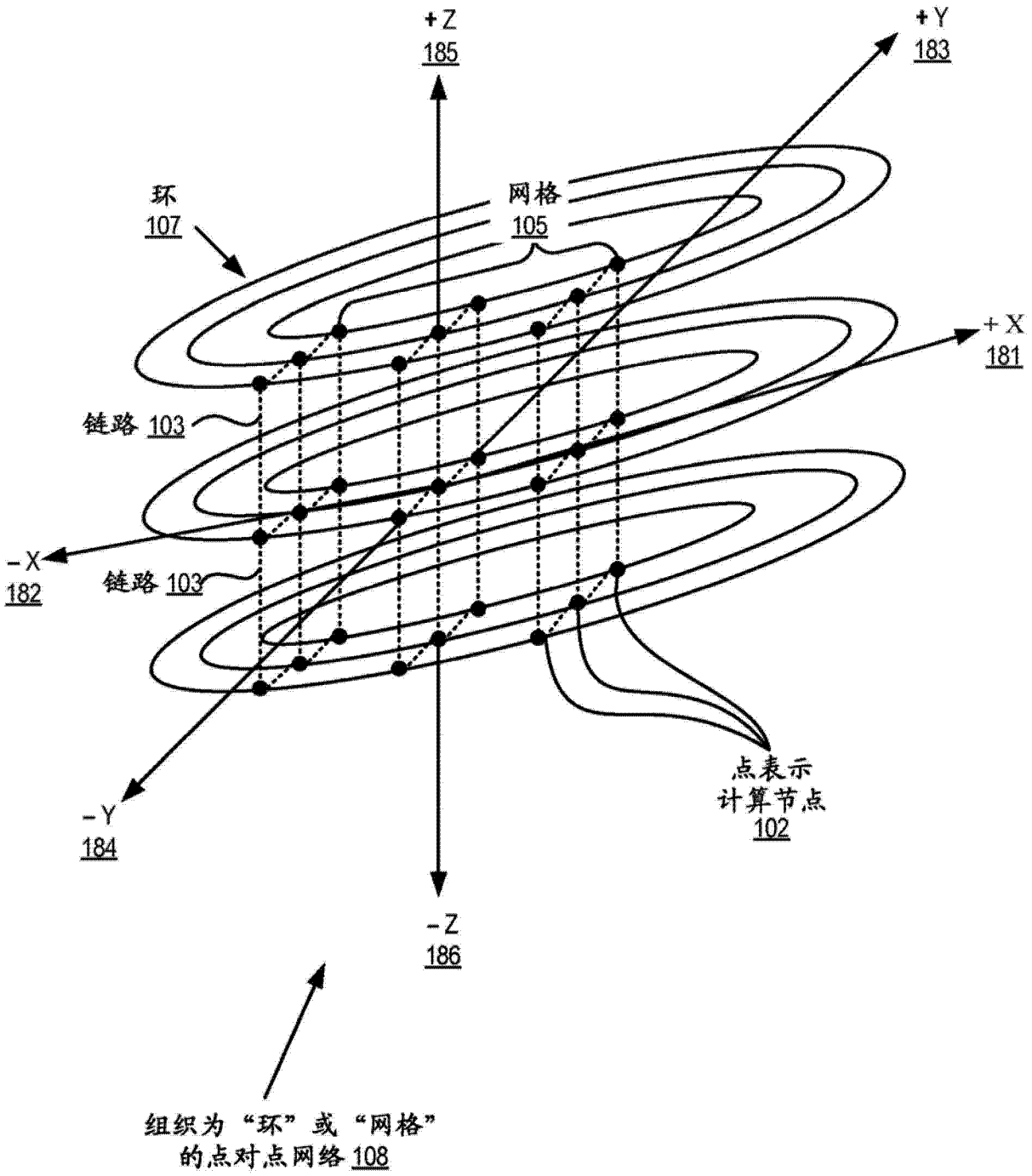

+Z
185
+Y
183
环
107
网格
105
+X
181
链路 103
-X
182
链路 103
-Y
184
点表示
计算节点
102
-Z
186
组织为“环”或“网格”
的点对点网络 108

图 4

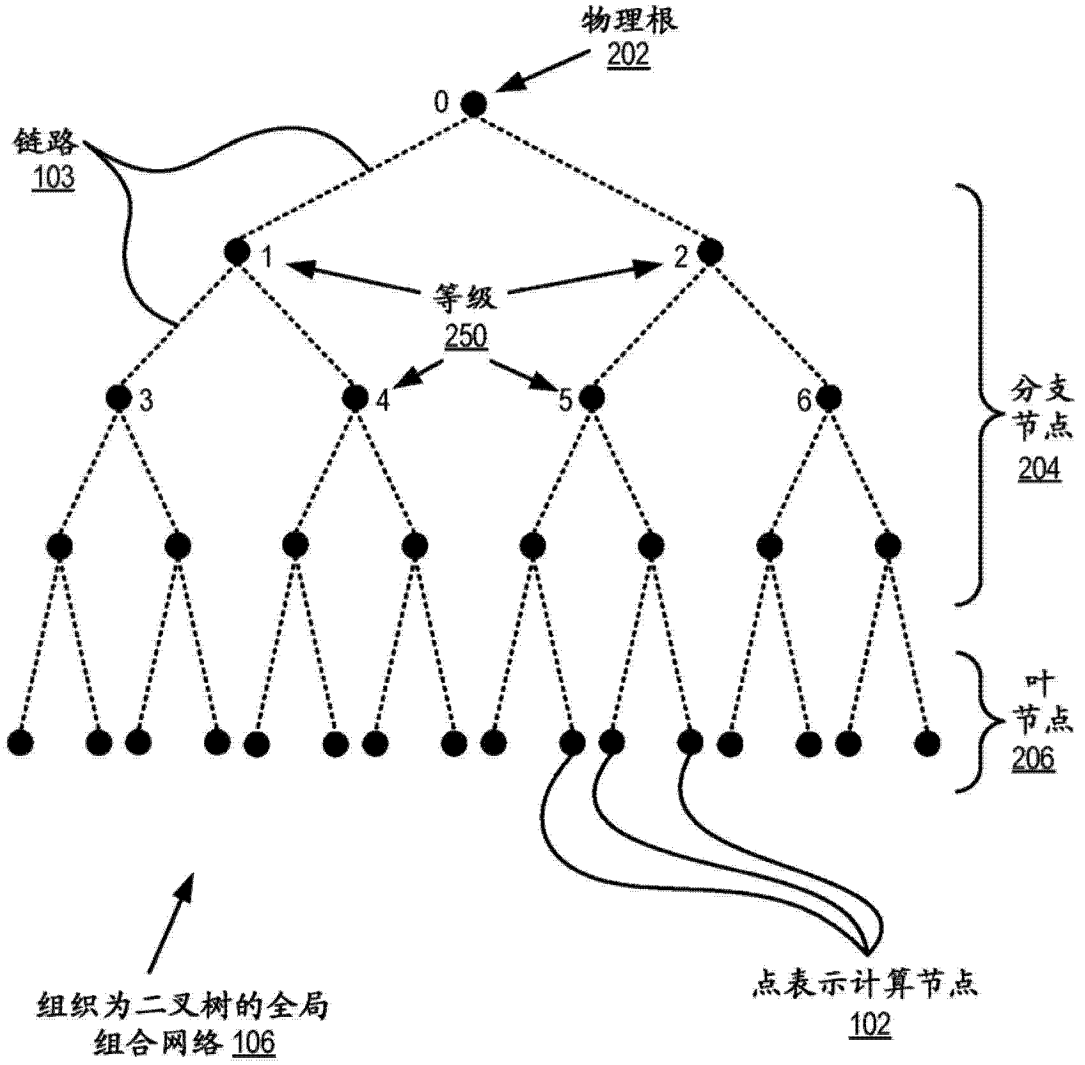

物理根
202
0
链路
103
1
2
等级
250
3
4
5
6
分支
节点
204
叶
节点
206
组织为二叉树的全局
组合网络 106
点表示计算节点
102

图 5

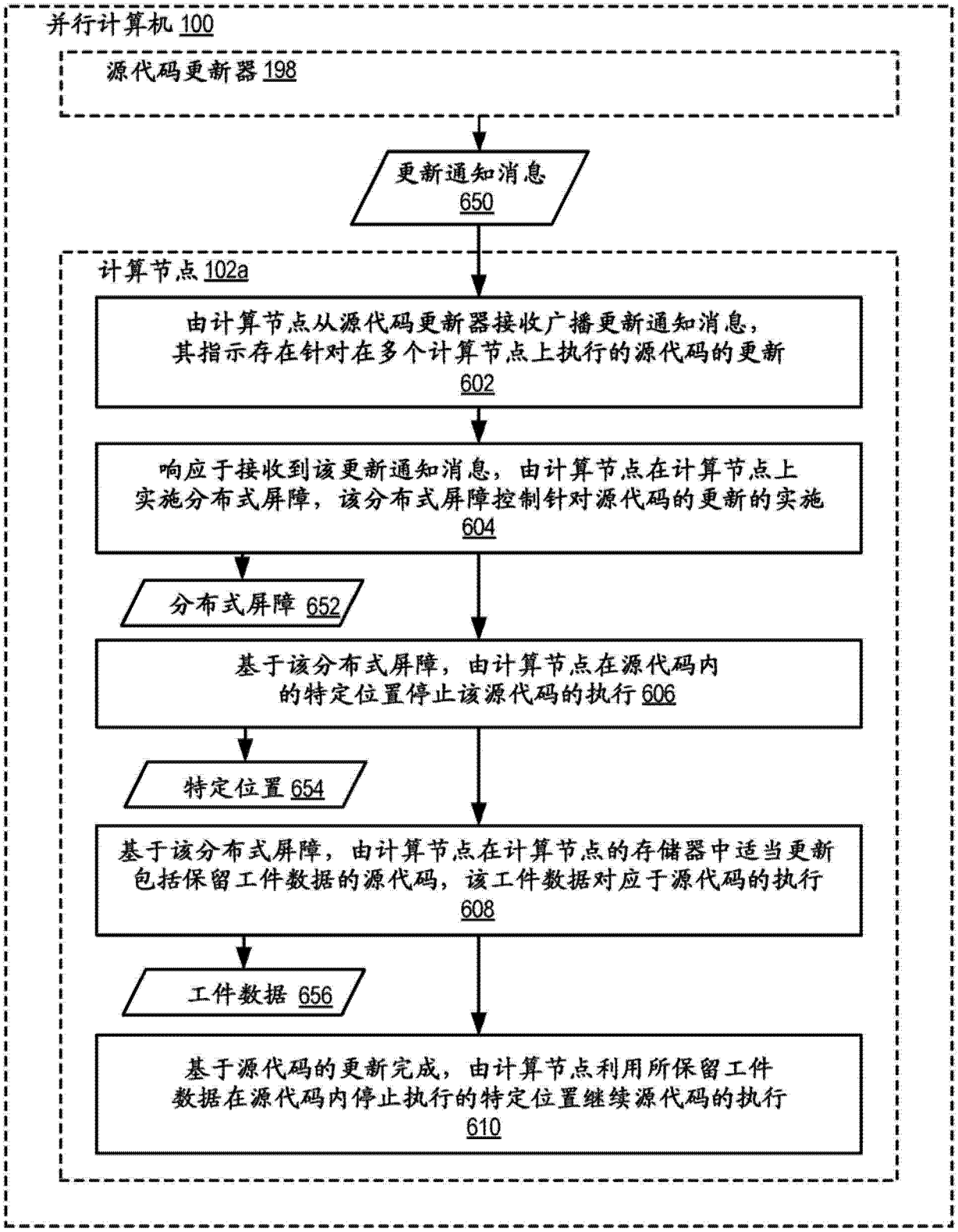

并行计算机 100
源代码更新器 198
更新通知消息 650
计算节点 102a
由计算节点从源代码更新器接收广播更新通知消息，其指示存在针对在多个计算节点上执行的源代码的更新 602
响应于接收到该更新通知消息，由计算节点在计算节点上实施分布式屏障，该分布式屏障控制针对源代码的更新的实施 604
分布式屏障 652
基于该分布式屏障，由计算节点在源代码内的特定位置停止该源代码的执行 606
特定位置 654
基于该分布式屏障，由计算节点在计算节点的存储器中适当更新包括保留工件数据的源代码，该工件数据对应于源代码的执行 608
工件数据 656
基于源代码的更新完成，由计算节点利用所保留工件数据在源代码内停止执行的特定位置继续源代码的执行 610

图 6

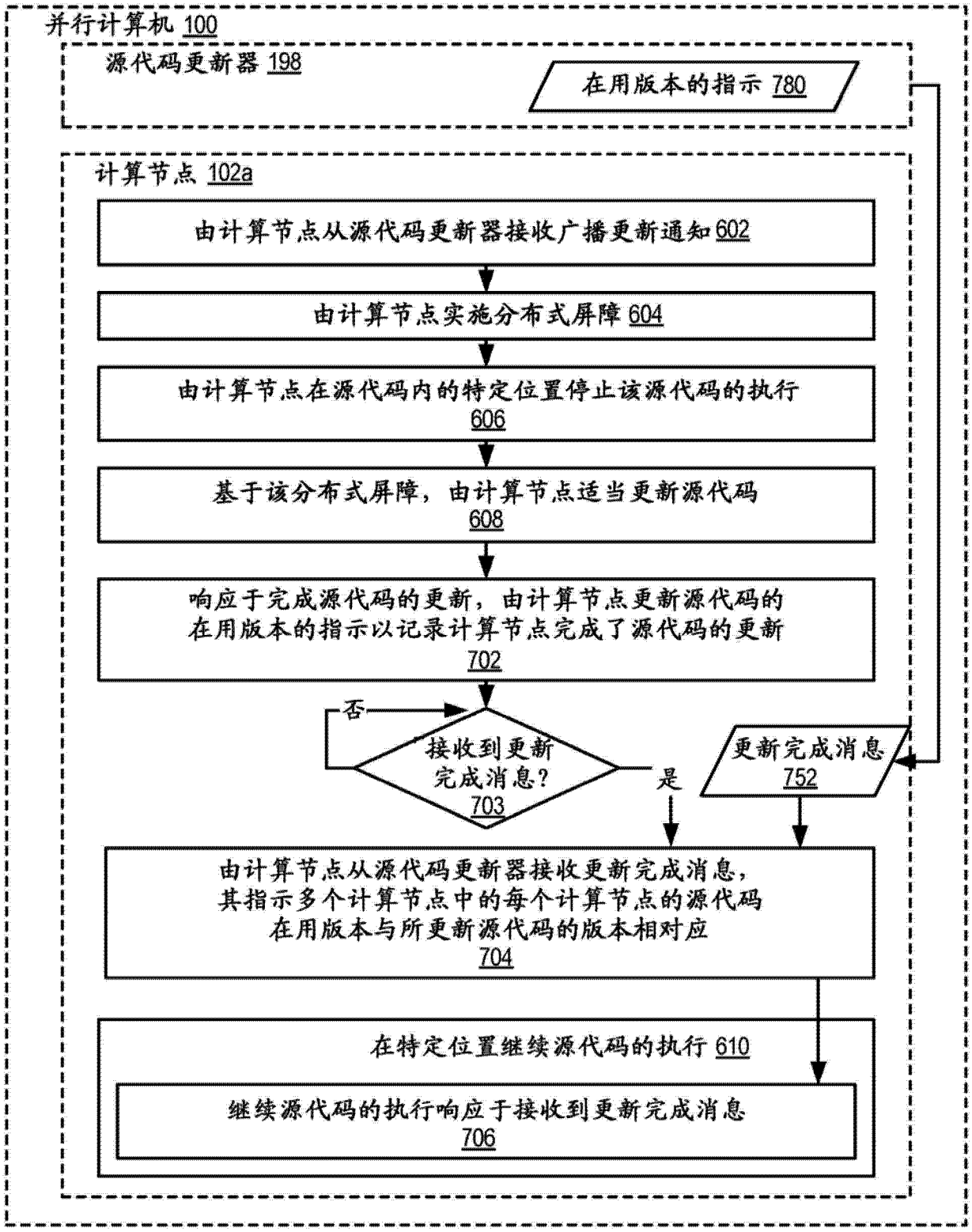

并行计算机 100
源代码更新器 198
在用版本的指示 780
计算节点 102a
由计算节点从源代码更新器接收广播更新通知 602
由计算节点实施分布式屏障 604
由计算节点在源代码内的特定位置停止该源代码的执行 606
基于该分布式屏障，由计算节点适当更新源代码 608
响应于完成源代码的更新，由计算节点更新源代码的在用版本的指示以记录计算节点完成了源代码的更新 702
否
接收到更新完成消息？ 703
是
更新完成消息 752
由计算节点从源代码更新器接收更新完成消息，其指示多个计算节点中的每个计算节点的源代码在用版本与所更新源代码的版本相对应 704
在特定位置继续源代码的执行 610
继续源代码的执行响应于接收到更新完成消息 706

图 7

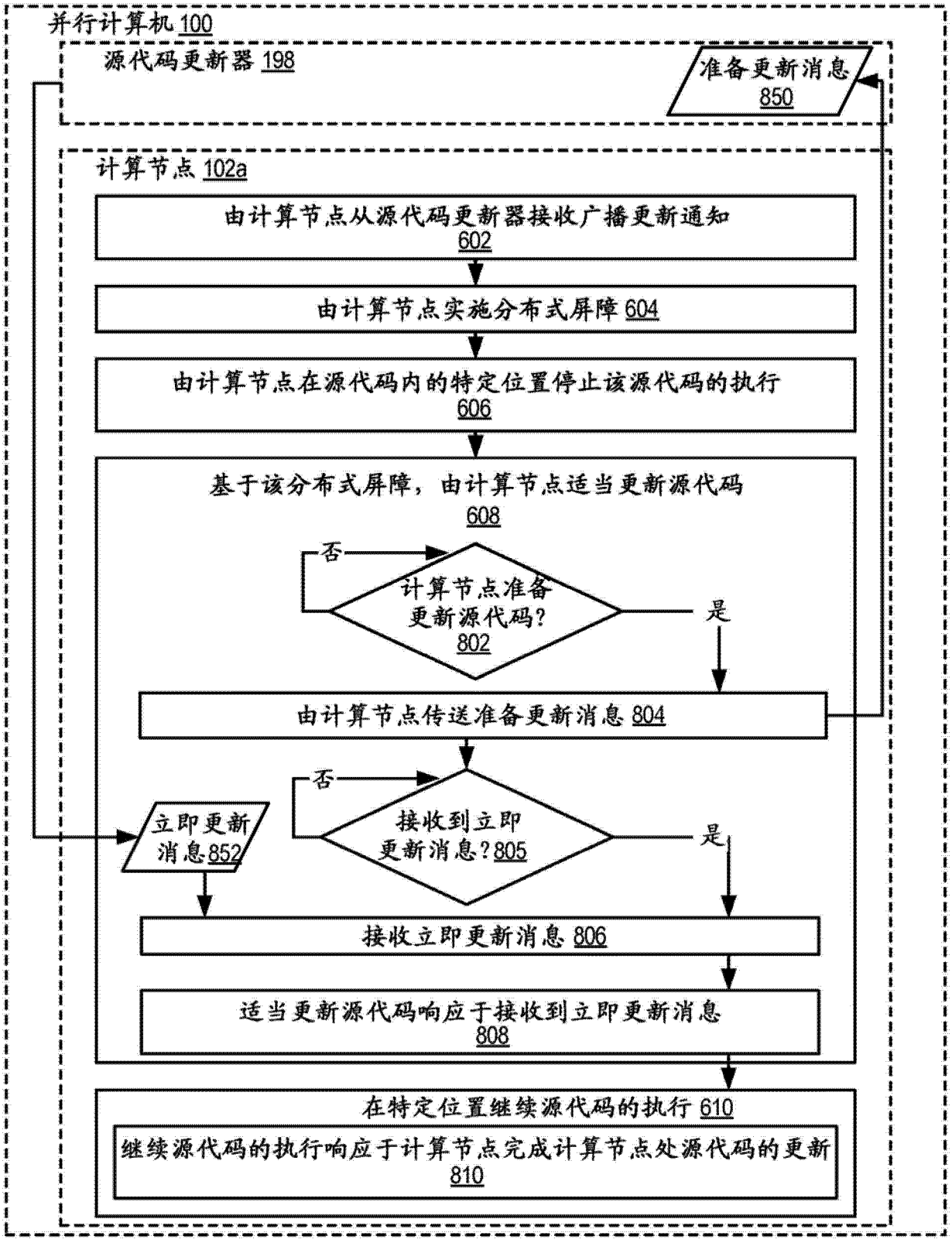

并行计算机 100
源代码更新器 198
准备更新消息 850
计算节点 102a
由计算节点从源代码更新器接收广播更新通知 602
由计算节点实施分布式屏障 604
由计算节点在源代码内的特定位置停止该源代码的执行 606
基于该分布式屏障，由计算节点适当更新源代码 608
否
计算节点准备更新源代码? 802
是
由计算节点传送准备更新消息 804
否
立即更新消息 852
接收到立即更新消息? 805
是
接收立即更新消息 806
适当更新源代码响应于接收到立即更新消息 808
在特定位置继续源代码的执行 610
继续源代码的执行响应于计算节点完成计算节点处源代码的更新 810

图 8