

(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 103649921 A

(43) 申请公布日 2014. 03. 19

(21) 申请号 201280026745. 4

(51) Int. Cl.

(22) 申请日 2012. 05. 10

G06F 9/54 (2006. 01)

(30) 优先权数据

13/172, 978 2011. 06. 30 US

(85) PCT国际申请进入国家阶段日

2013. 11. 29

(86) PCT国际申请的申请数据

PCT/EP2012/058687 2012. 05. 10

(87) PCT国际申请的公布数据

W02013/000616 EN 2013. 01. 03

(71) 申请人 国际商业机器公司

地址 美国纽约

(72) 发明人 A · 马基亚诺 R · 陶尔曹

A · 温特尔 G · 西特曼三世

J · 斯蒂文斯

(74) 专利代理机构 北京市中咨律师事务所

11247

代理人 于静 张亚非

权利要求书3页 说明书18页 附图14页

(54) 发明名称

促进通信环境的隔离存储空间之间的通信

(57) 摘要

将同步数据传输自动转换为异步数据传输。
使用同步数据传输协议启动将数据从发送器传输
到接收器。响应于确定将异步地发送数据，将数据
传输从同步数据传输自动转换为异步数据传输。

1. 一种用于促进通信环境中的通信的计算机程序产品,所述计算机程序产品包括 :
计算机可读存储介质,其可由处理电路读取并存储指令以便由所述处理电路执行以执行一种方法,所述方法包括 :
由所述通信环境的处理器从所述通信环境的发送器获得从同步数据传输转换为异步数据传输的预授权 ;
由所述处理器从所述发送器获得将数据发送到所述通信环境的接收器的请求 ;
由所述处理器启动将所述数据发送到所述接收器,其中所述启动采用所述同步数据传输 ;
由所述处理器确定所述同步数据传输将被转换为异步数据传输 ;以及
响应于所述确定,由所述处理器将所述同步数据传输自动转换为所述异步数据传输以便完成将所述数据发送到所述接收器,其中所述自动转换与所述发送器执行的除所述预授权之外的操作无关。
2. 根据权利要求 1 的计算机程序产品,其中所述确定包括确定在能够在所述数据被发送时接收所述数据的方面,所述接收器被延迟,并且响应于此,将所述同步数据传输自动转换为所述异步数据传输以便在其中所述接收器能够接收所述数据时完成所述数据的发送。
3. 根据权利要求 2 的计算机程序产品,其中所述预授权包括获得存储块的跟踪所述异步数据传输的指示。
4. 根据权利要求 3 的计算机程序产品,其中所述自动转换包括 :
将所述请求存储在所述存储块中 ;
将所述存储块排队到所述接收器的队列 ;
确定所述接收器能够接收所述数据 ;以及
使用所述存储块将所述数据发送到所述接收器。
5. 根据权利要求 4 的计算机程序产品,其中确定所述接收器能够接收所述数据包括确定所述接收器具有用于接收所述数据的空缓冲区。
6. 根据权利要求 4 的计算机程序产品,其中所述方法还包括指示完成到所述发送器的发送。
7. 根据权利要求 6 的计算机程序产品,其中所述指示完成包括在所述发送器可访问的存储器中的完成队列上发布所述存储块的地址。
8. 根据权利要求 7 的计算机程序产品,其中所述方法还包括响应于在所述完成队列上发布所述地址而生成到所述发送器的中断。
9. 根据权利要求 2 的计算机程序产品,其中确定所述接收器被延迟包括确定接收器缓冲区不可用于接收所述数据。
10. 根据权利要求 1 的计算机程序产品,其中所述方法还包括从所述发送器获得另一个请求,所述另一个请求将在完成将所述请求异步地发送到所述接收器之前被同步地发送到另一个接收器。
11. 根据权利要求 1 的计算机程序产品,其中所述方法还包括在完成将所述数据异步地发送到所述接收器之前,从所述发送器接收一个或多个其他请求,其中可以将所述一个或多个其他请求的数据同步地或异步地发送到一个或多个接收器。
12. 根据权利要求 1 的计算机程序产品,其中所述方法还包括响应于从所述同步数据

传输转换为所述异步数据传输，在所述接收器处维护数据到达顺序。

13. 一种用于促进通信环境中的通信的计算机系统，所述计算机系统包括：

存储器；以及

与所述存储器通信的处理器，其中所述计算机系统被配置为执行一种方法，所述方法包括：

由所述通信环境的处理器从所述通信环境的发送器获得从同步数据传输转换为异步数据传输的预授权；

由所述处理器从所述发送器获得将数据发送到所述通信环境的接收器的请求；

由所述处理器启动将所述数据发送到所述接收器，其中所述启动采用所述同步数据传输；

由所述处理器确定所述同步数据传输将被转换为异步数据传输；以及

响应于所述确定，由所述处理器将所述同步数据传输自动转换为所述异步数据传输以便完成将所述数据发送到所述接收器，其中所述自动转换与所述发送器执行的除所述预授权之外的操作无关。

14. 根据权利要求 13 的计算机系统，其中所述确定包括确定在能够在所述数据被发送时接收所述数据的方面，所述接收器被延迟，并且响应于此，将所述同步数据传输自动转换为所述异步数据传输以便在其中所述接收器能够接收所述数据时完成所述数据的发送。

15. 根据权利要求 14 的计算机系统，其中所述预授权包括获得存储块的跟踪所述异步数据传输的指示。

16. 根据权利要求 15 的计算机系统，其中所述自动转换包括：

将所述请求存储在所述存储块中；

将所述存储块排队到所述接收器的队列；

确定所述接收器能够接收所述数据；以及

使用所述存储块将所述数据发送到所述接收器。

17. 根据权利要求 16 的计算机系统，其中确定所述接收器能够接收所述数据包括确定所述接收器具有用于接收所述数据的空缓冲区。

18. 根据权利要求 16 的计算机系统，其中所述方法还包括指示完成到所述发送器的发送，其中所述指示完成包括在所述发送器可访问的存储器中的完成队列上发布所述存储块的地址，并且其中所述方法还包括响应于在所述完成队列上发布所述地址而生成到所述发送器的中断，其中确定所述接收器被延迟包括确定接收器缓冲区不可用于接收所述数据。

19. 一种促进通信环境中的通信的方法，所述方法包括：

由所述通信环境的处理器从所述通信环境的发送器获得从同步数据传输转换为异步数据传输的预授权；

由所述处理器从所述发送器获得将数据发送到所述通信环境的接收器的请求；

由所述处理器启动将所述数据发送到所述接收器，其中所述启动采用所述同步数据传输；

由所述处理器确定所述同步数据传输将被转换为异步数据传输；以及

响应于所述确定，由所述处理器将所述同步数据传输自动转换为所述异步数据传输以便完成将所述数据发送到所述接收器，其中所述自动转换与所述发送器执行的除所述预授

权之外的操作无关。

20. 根据权利要求 19 的方法,其中所述确定包括确定在能够在所述数据被发送时接收所述数据的方面,所述接收器被延迟,并且响应于此,将所述同步数据传输自动转换为所述异步数据传输以便在其中所述接收器能够接收所述数据时完成所述数据的发送。

促进通信环境的隔离存储空间之间的通信

背景技术

[0001] 本发明的一个方面一般地涉及通信环境中的通信，具体地说，涉及促进通信环境的隔离存储空间之间的数据传输。

[0002] 为了在隔离存储空间之间传输数据，通常使用联网技术和协议。例如，可以使用TCP/IP协议通过以太网链路将数据从一个隔离存储空间发送到另一个隔离存储空间。当前联网技术使能同步或异步发送数据。这就发送器而言是排他选择。

[0003] 当同步发送数据时，发送器被暂停直到数据传输完成。另一方面，如果异步发送数据，则发送器可以继续操作。

发明内容

[0004] 通过提供一种用于促进通信环境中的通信的计算机程序产品，克服现有技术的缺点并提供其他优点。所述计算机程序产品包括存储介质，所述存储介质可由处理电路读取并存储指令以便由所述处理电路执行以执行一种方法。所述方法包括：例如由所述通信环境的处理器从所述通信环境的发送器获得从同步数据传输转换为异步数据传输的预授权；由所述处理器从所述发送器获得将数据发送到所述通信环境的接收器的请求；由所述处理器启动将所述数据发送到所述接收器，其中所述启动采用所述同步数据传输；由所述处理器确定所述同步数据传输将被转换为异步数据传输；以及响应于所述确定，由所述处理器将所述同步数据传输自动转换为所述异步数据传输以便完成将所述数据发送到所述接收器，其中所述自动转换与所述发送器执行的除所述预授权之外的操作无关。

[0005] 在此还描述并要求保护与本发明的一个或多个方面相关的方法和系统。此外，在此还描述并可以要求保护与本发明的一个或多个方面相关的服务。

[0006] 通过本发明的一个或多个方面的技术实现其他特性和优点。在此详细描述了本发明的其他实施例和方面并将它们视为要求保护的本发明的一部分。

附图说明

[0007] 在说明书结尾处的权利要求中作为实例具体指出并明确要求保护了本发明的一个或多个方面。从下面结合附图的详细描述，本发明的一个或多个方面的上述和其他目标、特性和优点将变得显而易见，这些附图是：

[0008] 图1示出结合和/或使用本发明的一个或多个方面的通信环境的一个实例；

[0009] 图2示出根据本发明的一个方面的与图1的逻辑分区关联的存储空间的实例；

[0010] 图3示出根据本发明的一个方面使用的同步出站数据传输的一个实例；

[0011] 图4示出根据本发明的一个方面的用于从同步数据传输自动转换为异步数据传输的控制结构的实例；

[0012] 图5A示出根据本发明的一个方面的用于将同步数据传输自动转换为异步数据传输的逻辑的一个实施例；

[0013] 图5B以图形方式示出根据本发明的一个方面的成功完成异步数据传输的一个实

例；

- [0014] 图 6 示出结合本发明的一个或多个方面的计算机程序产品的一个实施例；
- [0015] 图 7 示出结合和使用本发明的一个或多个方面的主机计算机系统的一个实施例；
- [0016] 图 8 示出用于结合和使用本发明的一个或多个方面的计算机系统的进一步实例；
- [0017] 图 9 示出包括用于结合和使用本发明的一个或多个方面的计算机网络的计算机系统的另一个实例；
- [0018] 图 10 示出结合和使用本发明的一个或多个方面的计算机系统的各种元件的一个实施例；
- [0019] 图 11A 示出用于结合和使用本发明的一个或多个方面的图 10 的计算机系统的执行单元的一个实施例；
- [0020] 图 11B 示出用于结合和使用本发明的一个或多个方面的图 10 的计算机系统的分支单元的一个实施例；
- [0021] 图 11C 示出用于结合和使用本发明的一个或多个方面的图 10 的计算机系统的加载 / 存储单元的一个实施例；以及
- [0022] 图 12 示出结合和使用本发明的一个或多个方面的被仿真主计算机系统的一个实施例。

具体实施方式

[0023] 根据本发明的一个方面，提供一种能力以便将同步数据传输自动转换为异步数据传输。例如，响应于确定在完成数据传输时具有延迟(例如，在传输时数据接收器无法接收数据)，将同步数据传输自动转换为异步数据传输。自动执行从同步数据传输到异步数据传输的转换，因为它未根据发送器(或接收器)的请求，并且在启动转换时发送器(或接收器)并不知道转换。进一步，在转换时，发送器(或接收器)无需采取任何操作或干预转换。

[0024] 参考图 1 描述结合和 / 或使用本发明的一个或多个方面的通信环境的一个实施例。在一个实例中，通信环境 100 包括中央处理器复合体(CPC) 102，其基于国际商业机器公司(**IBM**[®])提供的**z/Architecture**[®]。在标题为“*z/Architecture Principles of Operation (z/Architecture 操作原理)*”(编号为 SA22-7832-08 的 IBM 出版物，2010 年 8 月，在此全部引入作为参考)的**IBM**[®]出版物中描述了**z/Architecture**[®]的各方面。可以包括中央处理器复合体 102 的一个系统是位于纽约阿蒙克的国际商业机器公司提供的 zEnterprise 196(z196)系统。**IBM**[®]和**z/Architecture**[®]是位于美国纽约阿蒙克的国际商业机器公司的注册商标，zEnterprise 196 和 z196 是其商标。在此使用的其他名称可能是国际商业机器公司或其他公司的注册商标、商标或产品名称。

[0025] 中央处理器复合体 102 例如包括一个或多个分区 104、系统管理程序 106、一个或多个中央处理器 108，以及输入 / 输出子系统 110 的一个或多个组件。在该实例中，一个或多个分区 104 是逻辑分区(又称为 LPAR)，它们包括被虚拟化为单独系统的一组系统硬件资源。

[0026] 每个逻辑分区 104 能够用作单独系统。即，每个逻辑分区可以独立重置，初始加载有操作系统 120(如果需要)并与不同的程序一起运行。在逻辑分区中运行的操作系统

或应用程序看似有权访问整个系统,但实际上,仅系统的一部分可用。硬件和许可内部代码(LIC)的组合(称为固件)防止一个逻辑分区中的程序干预不同逻辑分区中的程序。这允许多个不同的逻辑分区以时间片方式在单个或多个物理处理器上运行。在该实例中,多个逻辑分区具有常驻操作系统 120,对于一个或多个逻辑分区,常驻操作系统 120 可能有所不同。在一个实施例中,操作系统 120 是位于纽约阿蒙克的国际商业机器公司提供的**Z/OS®** 操作系统。

[0027] 如在此使用的,固件例如包括处理器的微代码、毫代码和 / 或宏代码。它例如包括用于实现更高级机器代码的硬件级别指令和 / 或数据结构。在一个实施例中,它例如包括典型地作为微代码提供的专用代码,专用代码包括特定于底层硬件的可信软件或微代码,并且控制操作系统对系统硬件的访问。

[0028] 逻辑分区 104 通过系统管理程序 106 管理,系统管理程序 106 通过在中央处理器 108 上运行的固件实现。系统管理程序 106 的一个实例是位于纽约阿蒙克的国际商业机器公司提供的 Processor Resource/Systems Manager (PR/SM™)。

[0029] 中央处理器 108 是分配给逻辑分区的物理处理器资源。例如,逻辑分区 104 包括一个或多个逻辑处理器,每个逻辑处理器表示分配给该分区的物理处理器资源 108 的全部或一部分。特定分区 104 的逻辑处理器可以专用于分区,以便针对该分区保留底层处理器资源;或者与另一个分区共享,以便底层处理器资源可能用于另一个分区。

[0030] 逻辑分区 104 和系统管理程序 106 均包括一个或多个程序,这些程序驻留在与中央处理器关联的主存储器 150 的相应部分中。在一个实例中,为每个逻辑分区分配主存储器的一部分,称为存储空间,如参考图 2 进一步详细描述的那样。

[0031] 参考图 2,在一个实施例中,主存储器 150 包括多个存储空间,每个存储空间包括主存储器中的一系列地址。可以将存储空间分配给实体,例如逻辑分区或其他实体。在图 2 中所示的实例中,存在分别分配给两个逻辑分区的两个存储空间。一个存储空间在此称为发送器的存储空间 202,另一个存储空间在此称为接收器的存储空间 204,因为在下面进一步描述发送器和接收器之间的通信。发送器的存储空间 202 例如包括一个或多个输入队列 210、一个或多个输出队列 212,以及一个或多个缓冲区 214。同样,接收器的存储空间 204 包括一个或多个输入队列 220、一个或多个输出队列 222,以及一个或多个缓冲区 224。在下面进一步描述队列和缓冲区的使用。

[0032] 个体存储空间彼此隔离,因为在没有固件控制的情况下,不能将数据从一个存储空间直接写入到另一个存储空间。在一个实例中,使用例如利用 TCP/IP 通过以太网链路的联网传输将数据从一个存储空间传输到另一个存储空间。在一个特定的实例中,使用国际商业机器公司提供的一种称为 HiperSockets™ 的技术执行传输。

[0033] HiperSockets™ 在中央处理器复合体中提供高速 TCP/IP 连接性。它在不同逻辑分区中运行的服务器之间不需要任何物理布线或外部联网连接。相反,通信通过处理器的系统存储器。HiperSockets™ 实现基于 OSA-Express 排队直接 I/O (QDIO) 协议。固件仿真 OSA-Express QDIO 接口的链路控制层。

[0034] 使用联网技术的从一个存储空间到另一个存储空间的数据传输例如是同步的,其中在启动数据传输之后,发送器被暂停直到传输完成。参考图 3 描述从一个存储空间到另一个存储空间的同步数据传输的一个实例。

[0035] 参考图 3,发送器 300 (例如在发送器的存储空间中运行的 TCP/IP 堆栈或程序)启动请求以便将数据发送到接收器 310(例如接收器的存储空间中的另一个 TCP/IP 堆栈或程序)。因为接收器的存储空间与发送器的存储空间隔离,所以在该实例中,使用网络通信协议针对从发送器发送到接收器的数据执行同步数据传输。使用同步数据传输,通过在固件的控制下执行从一个位置到另一个位置的存储器传输,在发送器和接收器之间提供非常快速、低延迟的直接通信路径。在一个实例中,通过 HiperSockets™ 执行存储器到存储器数据传输。只要接收器接收数据的速度等于或快于发送器发送数据的速度,此传输机制便会非常高效。

[0036] 为了传输数据,发送器获得包含在选定数据缓冲区 320 中的数据,并且将其放置在发送器的输出队列 330 上。例如,将选定数据缓冲区的指针 332 放置在输出队列上。然后,发送器 300 用信号通知(340)处理器以便执行到接收器 310 的数据传输。在一个实例中,用信号通知处理器的固件 350,并且固件 350 将执行传输;但是,在其他实例中,不是固件而是处理器的其他代码和 / 或硬件。

[0037] 响应于接收到请求从发送器到接收器的数据传输的信号,固件复制发送器的输出队列中的数据,并且将其放置在接收器的输入队列 360 上。例如,将数据复制到空缓冲区 370,并且将指向该数据的指针 372 放置在输入队列 360 中。在完成数据传输之后,固件向发送器返回传输完成的信号。在发送器接收该完成信号之前,发送器被暂停并且不能执行任何其他操作。

[0038] 作为一个特定的实例,为了用信号通知处理器,发送器发出 Signal Adapter (SIGA) 指令,其指定 write 函数(SIGA-w),该函数用信号通知处理器一个或多个输出队列具有要传输到接收器的数据。该 write 函数被指定为在指令使用的第一通用寄存器中提供的函数代码,并且在指令使用的第二通用寄存器中指示该 write 函数的网络连接地址(例如,子系统标识字)。进一步,还在指令使用的第三通用寄存器中指定输出队列。

[0039] 在该特定的实例中,队列实现为排队直接 I/O (QDIO) 队列,并且每个队列具有与其关联的多个缓冲区以及各种控制信息。在一个实施例中,QDIO 队列包括描述队列的数据结构,以及用于数据传输的缓冲区存储块。作为一个实例,共同描述队列的特性并且提供控制以便允许数据交换的多个存储数据结构(称为队列组件)例如包括:

[0040] 队列信息块(QIB),其包括有关 QDIO 输入和输出队列集合的信息。

[0041] QIB 包括输入队列的存储列表信息块(SLIB)地址和输出队列的 SLIB 地址。

[0042] 每个队列存在一个 SLIB,并且每个 SLIB 提供有关队列和队列的每个队列缓冲区的信息。每个 SLIB 具有头和一个或多个表项,称为存储列表信息块表项(SLIBE),这些表项包含有关每个队列的每个缓冲区的信息。在一个实例中,每个存储列表信息块包括下一个存储列表信息块的地址、存储列表(SL)的地址和存储列表状况块(SLSB)的地址。

[0043] 针对每个队列定义一个存储列表,并且 SL 例如包括 128 个表项,队列的每个缓冲区一个表项。存储列表提供有关主存储器中的 I/O 缓冲区位置的信息。每个表项包括存储块地址列表(SBAL)的绝对地址。每个存储块地址列表包括共同组成与每个队列关联的数据缓冲区之一的存储块的绝对地址列表。

[0044] 存储块列表表项(SBALE)作为每个 SBAL 的一部分提供。每个 SBALE 包括存储块的绝对存储地址。总体上,通过单个 SBAL 的所有表项寻址的存储块组成 QDIO 队列的许多可

能 QDIO 缓冲区之一。在一个实例中, QDIO 队列可以具有与其关联的 128 个 QDIO 缓冲区。

[0045] SLSB 包括状态指示符,它们提供有关组成队列的缓冲区的状态信息。

[0046] 有关 SIGA、QDIO 队列和关联的控制结构的进一步详细信息,在以下各项中描述 : Baskey 等人的标题为“Self-Contained Queues With Associated Control Information For Receipt And Transfer Of Incoming And Outgoing Data Using A Queued Direct Input-Output Device”的美国专利 6,332,171B1,2001 年 12 月 18 日公告 ;Brice 等人的标题为“Method And Apparatus For Simulation Of Data In a Virtual Environment Using A Queued Direct Input-Output Device”的美国专利 6,345,241B1,2002 年 2 月 5 日公告 ;Markos 等人的标题为“Method And Apparatus For Providing Configuration Information Using A Queued Direct Input-Output Device 的美国专利 6,519,645B2, 2003 年 2 月 11 日公告 ;以及 Easton 等人的标题为“Interpreting I/O Operation Requests From Pageable Guests Without Host Intervention”的美国专利 7,941,799B2,2011 年 5 月 10 日公告。

[0047] 在上面的处理中,当接收器无法以发送器发送数据的相同速度提供空缓冲区时,在发送器侧引入增加的延迟和 CPU 开销,这是由于协议的同步性质所致。当接收器上未提供空缓冲区时,发送器具有两个选择。它可以承受对失败操作进行排队并且随后将数据重新传输到同一接收器的开销,或者丢弃数据从而允许较高级别通信协议(如 TCP/IP)重新驱动操作。对数据进行排队和重新传输的缺陷是它不仅需要额外的 CPU 周期以便恢复,而且潜在地可阻止或延迟发送器到其他目的地(可能能够接受传入数据)的其他传输。

[0048] 允许多个服务器(例如,发送器、接收器)共享 CPU 资源的虚拟化环境更可能产生以下情况 :其中接收器可能无法跟上向其发送数据的各种发送器。这通常是在以下情况 :当系统管理程序(如 PR/SM)控制在各种可用的共享处理器上分派服务器时。当在发送器和共享的 CPU 资源之间具有多个级别的系统管理程序时,问题变得复杂起来。这例如是在以下情况 :当发送器在 **z/VM®** 下的虚拟机中运行时, **z/VM®** 也在逻辑分区中运行。在此,两个系统管理程序要分派服务器,以便允许服务器及时补充空缓冲区。

[0049] 当提供足够的 CPU 资源时,及时分派接收器并不是问题。仅当 CPU 资源在短期或长期内变得受限时,同步数据传输才会中断。因此,根据本发明的一个方面,发送器能够在无约束 CPU 环境中利用同步低延迟数据传输,而当 CPU 资源变得受限时消除关联的缺陷和开销。这通过以下操作实现 :当接收器无法跟上发送器时,将通信协议从同步转换为异步。在一个实施例中,自动执行同步到异步协议转换,而无需到发送器的任何推回(即,在转换时发送器不需要执行任何操作)。当 CPU 资源针对特定接收器变得受限时,这使得发送器无需执行任何类型的恢复处理,或者阻止或延缓到其他接收器的数据传输。此外,它允许同步通信自动继续到先前受限的接收器。

[0050] 根据本发明的一个方面,提供一种能力以便发送器在其存储器中对数据进行排队,直到接收器提供可以接收数据的空缓冲区。为了促进该处理,使用一个或多个控制结构,如参考图 4 描述的那样。例如,发送器针对每个待处理异步传输,在发送器存储器中分配存储器 400 的一个空块。在一个实例中,在任何时刻可能存在 X 个这样的块,其中 X 与模型相关并且可配置。X 表示发送器允许的并发异步请求数量。使用该存储块(称为 QDIO(排队直接输入 / 输出)异步操作块(QAOB))跟踪异步数据传输,直到固件完成操作。在该实例

中, QAOB 仅包含控制信息而不包含数据本身。当针对发送器将允许可选地异步执行的请求启动数据传输时, 发送器提供该块; 否则, 它不需要提供控制块。这使发送器能够控制未完成异步请求的最大数量。当 QAOB 确定要异步执行数据传输时, 仅使用 QAOB 本身并且通过固件对其进行初始化。在要异步执行数据传输的情况下, 除了为 QAOB 提供存储器之外, 发送器不必针对数据传输执行任何操作。在一个特定的实例中, QAOB 包括在提供请求的 SIGA 指令中。发送器发出 SIGAwrite with QAOB (SIGA-wq) 指令, 其指定第一通用寄存器中指示 writewith QAOB 函数的选定函数代码。在指令使用的第四通用寄存器中指定 QAOB 地址。该通用寄存器具有 0 (当不指定 QAOB 时) 或者 QAOB 的绝对地址(例如, 256 字节 QAOB)。响应于设置 write with QAOB 函数代码, 固件判定是否在第四通用寄存器中指定可以用于异步数据传输的 QAOB。

[0051] 当固件将数据传输更改为异步协议时, 它使用 QAOB 跟踪驻留在发送器的存储器中与出站数据传输关联的数据。在其中使用 HiperSockets™ 进行传输的实例中, QAOB 跟踪发送器在存储块地址列表(SBAL)中指定的与数据传输关联的地址和控制。从 SBAL 中提取并且放置在 QAOB 中的字段实例例如包括:

[0052] ● 来自 SBAL 的所有有意义的 SBALE (例如, 第一个 SBALE 到设置了最后一个表项位的 SBALE)。这包括数据的绝对缓冲区地址和字节计数;

[0053] ● SBAL 的输出队列号;

[0054] ● 启动请求的 SBAL 的缓冲区编号(例如, 1-27);

[0055] ● 有意义的 SBALE 表项数量; 以及

[0056] ● 用于访问由每个有意义的 SBALE 指定的存储块的存储键。

[0057] 除了 QAOB 之外, 使用另一个控制结构, 称为完成队列(CQ)410。即, 在一个实例中, 当建立通信队列时, 除了提供 QAOB 之外, 发送器还在其存储器中分配新的队列类型, 即, 完成队列。在 HiperSockets™ 的情况下, 这是新类型的 QDIO 输入队列, 其具有 SBAL 但没有与 SBALE 关联的缓冲区。不使用这种新的输入队列传输数据; 相反, 固件使用它发布完成事件以便用信号通知发送器异步数据传输已完成。当队列表项变成“入站就绪”时, 有关完成事件的信息位于 SBALE 本身中, 这些 SBALE 包括在完成队列上。固件在 CQ 中发布与完成的异步数据传输关联的 QAOB 的地址, 并且生成中断(如有必要)以便用信号通知发送器完成的数据传输。此时, 发送器可以重用与完成的操作关联的存储器以便实现其他用途。(在一个实施例中, 使用 SBALE 发布单个完成事件。因为 SBAL 例如包括 16 个 SBALE, 所以固件在单个 SBAL 中最多可以发布 16 个完成事件(QAOB)。)

[0058] 进一步, 在一个实施例中, 固件针对每个预定接收者具有另一个队列 TPQ420, 其用于记住它具有未完成的数据传输请求。

[0059] 参考图 5A-5B 描述有关从同步数据传输转换为异步数据传输的进一步细节。图 5A 示出固件用于执行转换的逻辑的一个实施例, 图 5B 以图形方式示出转换的一个实例。在下面的讨论中参考这两个图。

[0060] 参考图 5A-5B, 初始地, 固件 350 接收 QAOB 的指示(例如, 可以用于异步数据传输的存储块的地址)(步骤 500)。因此, 已知如果需要(或要求), 它可以异步执行被请求数据传输。在一个实施例中, 固件可以接收多个 QAOB, 这些 QAOB 指示它最多可以异步执行该数量的数据传输。发送器提供 QAOB 是对固件的预授权—固件可以异步执行数据传输(如果它如

此选择)。

[0061] 在一个特定的实例中, QAOB 被包括为固件从发送器 300 接收的数据传输请求的一部分(步骤 502)。响应于接收到数据传输请求, 固件尝试将数据发送到接收器(步骤 504)。如果接收器能够接收数据(例如, 接收器处具有空缓冲区)(查询 506), 则异步地传输数据(步骤 508), 并且完成数据传输。之后, 发送器可以执行另一个数据传输, 并且如果发送器允许异步处理, 则发送器可以将 QAOB 包括在新的请求中。

[0062] 但是, 如果接收器当前无法接收数据(例如, 缓冲区状态确定接收器处没有空缓冲区, 因此接收器在接收数据方面被延迟), 则固件自动将数据传输从同步请求转换为异步请求(假设在请求中提供了 QAOB)。将请求保存在 QAOB 中(步骤 510), 并且例如通过将指针放置到 TPQ 上的 QAOB, 在 TPQ420 中的预定目的地上对 QAOB 进行排队(步骤 512)。QAOB 现在包括 SBAL 的内容, 因此 SBAL 可以用于其他处理。

[0063] 在一个实施例中, 如果未指定 QAOB, 则请求失败或等待, 直到可以同步发送该请求。

[0064] 在一个实例中, 响应于对 QAOB 进行排队, 为发送服务器提供控制, 并且具有使用指定的 QAOB 异步执行该数据传输的指示。发送服务器然后可以立即针对其下一个数据传输进行设置, 并且在后续数据传输需要的情况下, 可选地分配另一个 QAOB。

[0065] 接下来, 判定接收器是否能够接收数据(查询 514)。例如, 判定固件是否从接收器接收到它现在能够接受数据的信号(例如, 现在提供空缓冲区)或者固件是否通过检查缓冲区状态而确定缓冲区可用。在一个特定的实例中, 接收器使用 SIGA 指令用信号通知固件它在其输入队列上放置了空缓冲区。在第一通用寄存器中指定 SIGA read (SIGA-r) 函数代码。SIGA read 函数导致固件将任何待处理分组从接收器的 TPQ 传输到目标的输入缓冲区。

[0066] 如果接收器无法接受数据, 则固件等待。否则, 如果接收器能够接收数据(例如, 具有至少一个空缓冲区), 则固件判定它是否具有指向接收器 TPQ 上的 QAOB 的指针(查询 518)。如果否, 则处理完成。否则, 固件使用 QAOB 执行数据传输。具体地说, 它将 QAOB 指向的数据转送到接收器, 从而将该数据放置在空缓冲区中, 并且将指针放置到接收器的输入队列上的现在填满的缓冲区。

[0067] 之后, 固件向发送器指示数据传输完成(步骤 520)。在一个实例中, 该指示包括在发送器的完成队列 410 (图 5B) 中发布与完成的异步数据传输关联的 QAOB 地址, 并且生成中断 550 (如有必要)以便用信号通知发送器完成的数据传输。此时, 发送器可以重用与完成的操作关联的存储器以便实现其他用途。在 CQ 上发布的 QAOB 包括有关异步处理的信息, 例如包括状态信息、完成代码、错误代码等。

[0068] 在一个特定的实例中, 当在完成队列表项中发布 QAOB 地址(即, QAOB 地址包括在位于完成队列表项中的 SBALE 中)时, 固件例如向 QAOB 中的程序返回以下信息:

[0069] ●反映异步 I/O 操作的结果的原因代码;以及

[0070] ●异步数据传输的缓冲区状态:“队列状态—缓冲区 N (SQBN)”。这是针对同步数据传输放置在 SLSB 中的相同值。SQBN 包含指示 QAOB 的当前状态的值。状态值例如包括两个部分:指示程序固件是否拥有缓冲区的第一部分;以及指示 QAOB 的当前过程状态的第二部分。

[0071] 尽管异步执行特定的请求, 但可以同步执行从发送器到其他接收器的其他请求,

除非特定接收器例如在接收数据方面被延迟。此外，自动将从发送器到接收器的其他请求恢复到同步，除非再次确定接收器在接收数据方面被延迟。例如，固件尝试将数据发送到接收器(如上所述)，并且如果接收器能够接收数据，则同步发送数据。在一个实施例中，即使在同步和异步之间转换传输，但数据仍然采用与传输相同的 FIFO 顺序。在从该发送器或任何其他发送器接收任何未来同步请求之前，固件将传输接收器 TPQ420 上的所有排队的 QAOB。如果向接收器进行已经在其 TPQ420 上具有 QAOB 的另一个同步数据传输，则固件自动将该同步数据传输转换为异步请求(如果发送器授权)，或者该同步数据传输由于没有可用的缓冲区响应而失败。保持该顺序可避免成本高昂的重新排序处理，从而提高接收器的 TCP/IP 堆栈的性能和整体 CPU 利用率。

[0072] 上面详细描述了一种能力，其将同步数据传输自动转换为异步数据传输以响应处理器(例如，固件)确定发生这种转换。例如，响应于接收器在能够接收数据方面被延迟(例如，没有可用的缓冲区、响应缓慢等)而执行转换。此外，所述能力允许自动同步执行到同一接收器的其他传输。

[0073] 此支持的一个或多个方面提供这种能力，以便调整到瞬态或更长期的数据传输，而不必对目标目的地进行成本高昂的重新传输，并且对网络中的其他运行的目的地具有最小影响或者没有影响。在一个方面，提供一种能力以便在同步和异步数据传输之间来回转换，而不必推回到发送器程序。在使用 HiperSockets 完成队列(CQ)的一个特定实例中，设备驱动器不暂停 I/O 操作(SIGA 不阻止此过程)。使用 CQ，暂不考虑写入过程并且允许发送器继续向同一目标或其他目标发出其他写入(SIGA(多个))，其中某些写入可能同步完成而其他写入异步完成。此非阻止方面使 CQ 异步。

[0074] 根据本发明的一个方面，仅将数据从发送器复制到接收器一次(无内部缓冲区)，与传输是同步还是异步无关。

[0075] 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，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。

[0076] 可以采用一个或多个非临时性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存储介质。计算机可读存储介质例如可以是一但不限于一电、磁、光、电磁、红外线、或半导体的系统、装置或器件，或者上述的任意合适的组合。计算机可读存储介质的更具体的例子(非穷举的列表)包括：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。

[0077] 现在参考图 6，在一个例子中，计算机程序产品 600 包括，例如，一个或多个非易失性计算机可读存储介质 602，在其上存储有计算机可读的程序代码装置或逻辑 604，以提供并方便本发明的一个或多个方面。

[0078] 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。

[0079] 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的各个方面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、汇编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)。

[0080] 在此参照根据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 / 或框图描述本发明的各个方面。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、专用计算机或其他可编程数据处理装置的处理器,从而生产出一种机器,使得这些指令在通过计算机或其他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。

[0081] 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、其他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)。

[0082] 也可以把计算机程序指令加载到计算机、其他可编程数据处理装置、或其他设备上,使得在计算机、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,从而使得在计算机或其他可编程装置上执行的指令提供实现流程图和 / 或框图中的一个或多个方框中规定的功能 / 动作的过程。

[0083] 附图中的流程图和框图显示了根据本发明的不同实施例的系统、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。

[0084] 除了上述,本发明的一个或多个方面可由服务提供商提供、许诺(offer)、部署、管理、服务等,该服务提供商提供用户环境的管理。例如,服务提供商可创建、维持、支持等计算机代码和 / 或计算机基础设施,其为一个或多个用户执行本发明的一个或多个方面。反过来,服务提供商可例如根据预订和 / 或费用协议从用户接受付费。额外地或可替换地,服务提供商可从向一个或多个第三方销售广告内容接受付费。

[0085] 在本发明的一个方面,可部署用于执行本发明的一个或多个方面的应用。作为一

个例子，部署应用包括提供计算机基础设施，其可操作以执行本发明的一个或多个方面。

[0086] 作为本发明的另一个方面，可部署计算基础设施，其包括将计算机可读代码集成到计算机系统，其中与计算系统结合的代码能够执行本发明的一个或多个方面。

[0087] 作为本发明的再一个方面，可提供用于集成计算基础设施包括将计算机可读码集成到计算机系统的过程。计算机系统包括计算机可读介质，其中计算机介质包括本发明的一个或多个方面。与计算机系统结合的代码能够执行本发明的一个或多个方面。

[0088] 尽管上面描述了各种实施例，但这些只是实例。例如，其他架构的计算环境可以结合和使用本发明的一个或多个方面。例如，zEnterprise 服务器之外的服务器可以包括、使用和 / 或受益于本发明的一个或多个方面。进一步，从同步到异步的转换可以响应于缓冲区是否可用之外的考虑事项。更进一步，本发明的一个或多个方面可以用于隔离存储空间之间的任何存储器到存储器传输。还可能具有许多其他变型。

[0089] 而且，其他类型的计算环境可受益于本发明的一个或多个方面。作为例子，可使用适于存储和 / 或执行程序代码的数据处理系统，其包括至少两个通过系统总线直接或间接耦合到存储元件的处理器。存储器元件包括，例如，在程序代码的实际执行期间使用的本地存储器、大容量存储器以及高速缓冲存储器，其提供至少一些程序代码的临时存储，以便减少在执行期间必须从大容量存储器取回代码的次数。

[0090] 输入 / 输出或 I/O 设备(包括但不限于键盘、显示器、指点设备、DASD、磁带、CD、DVD、拇指驱动器(thumb drive)以及其他的数据处理系统等)可直接或通过介于其间的 I/O 控制器被耦合到系统。网络适配器也可被耦合到系统以使得数据处理系统能够通过介于其间的私有或公共网络而耦合到其他的数据处理系统或远程打印机或存储设备。调制解调器、电缆调制解调器和以太网卡仅是一些可获得的网络适配器类型。

[0091] 参考图 7，其描述了实施本发明的一个或多个方面的主机计算机系统 5000 的代表性组件。代表性主机计算机 5000 包括与计算机存储器(即，中央存储器)5002 通信的一个或多个 CPU5001，以及到存储介质设备 5011 和网络 5010 的 I/O 接口，以用于与其他计算机或 SAN 等通信。CPU5001 符合具有架构指令集和架构功能的架构。CPU5001 可具有动态地址转换(DAT)5003，其用于将程序地址(虚拟地址)转变为存储器的真实地址。DAT 典型地包括用于高速缓存转换的转换后备缓冲器(TLB)5007，这样稍后对计算机存储器 5002 块的访问不需要地址转换的延迟。典型地，高速缓存 5009 被使用在计算机存储器 5002 和处理器 5001 之间。高速缓存 5009 可以是分层的，具有可被多于一个的 CPU 获得的大高速缓存，以及大高速缓存和每个 CPU 之间的较小、较快(较低级)的高速缓存。在一些实施方式中，较低级的高速缓存被拆分以为指令获取和数据访问提供单独的低级高速缓存。在一个实施例中，由指令获取单元 5004 经由高速缓存 5009 从存储器 5002 获取指令。指令在指令解码单元 5006 中被解码，且(在一些实施例中与其他指令一起)被发送到一个或多个指令执行单元 5008。典型地，使用若干执行单元 5008，例如算术执行单元、浮点执行单元和分支指令执行单元。指令被执行单元执行，如需要，从指令指定的寄存器或存储器访问操作数。如果将从存储器 5002 访问(加载或存储)操作数，加载 / 存储单元 5005 典型地在被执行的指令的控制下处理该访问。指令可在硬件电路或内部微代码(固件)中或其组合中被执行。

[0092] 需注意的是，计算机系统包括本地(或主)存储器中的信息，以及寻址、保护以及引用和改变记录。寻址的一些方面包括地址格式、地址空间的概念、地址的各种类型和其中一

种类型的地址被转换为另一种类型地址的方式。一些主存储器包括永久分配的存储位置。主存储器向系统提供直接可被寻址的快速访问的数据存储。数据和程序在可被处理之前都将(从输入设备)被加载到主存储器。

[0093] 主存储器可包括一个或多个更小、更快速访问的缓冲存储器,有时候被称为高速缓存。高速缓存典型地与 CPU 或 I/O 处理器物理地关联。物理结构以及不同存储介质的使用的效果,除了在性能上,通常不会被程序观察到。

[0094] 可维护用于指令和数据操作数的单独的高速缓存。高速缓存中的信息可被维护为相邻的字节,所述字节位于被称为高速缓存块或高速缓存线(或简称为线)的整数界限上。模型可提供 EXTRACT CACHE ATTRIBUTE(提取高速缓存属性)指令,其返回高速缓存线的字节大小。模型也可提供 PREFETCH DATA (预取数据)和 PREFETCH DATA RELATIVE LONG (预取较长数据) 指令,其实现存储到数据或指令高速缓存中的预取,或数据从高速缓存的释放。

[0095] 存储器被视为位的长水平串。对于大部分操作来说,以从左到右的顺序进行对存储器的访问。位串被细分为八个位的单位。八位单位被称为字节,其是所有信息格式的基本构件。存储器中的每个字节位置由唯一的非负整数标识,该非负整数是该字节位置的地址,或简称为字节地址。相邻的字节位置具有连续的地址,在左边从 0 开始且以从左到右的顺序进行。地址是无符号二进制整数,且是 24、31 或 64 位。

[0096] 信息一次一个字节或一组字节地在存储器和 CPU 或通道子系统之间传递。除非另有指定,例如在 **z/Architecture** 中,存储器中的一组字节由该组的最左边的字节寻址。组中的字节的数量可由将被执行的操作暗示或显式地指定。当在 CPU 操作中使用时,一组字节被称为字段。在字节的每个组内,例如在 **z/Architecture** 中,位以从左到右的顺序被编号。在 **z/Architecture** 中,最左边的位有时候被称为“高阶”位且最右边的位被称为“低阶”位。但是,位数不是存储器地址。仅字节可被寻址。为了操作存储器中的字节的单个位,访问整个字节。字节上的位从左到右被编号为 0 到 7(例如在 **z/Architecture** 中)。对于 24 位地址,地址中的位被编号为 8-31 或 40-63,对于 31 位地址,编号为 1-31 或 33-63,对于 64 位地址,编号为 0-63。在多个字节的任何其他的固定长度的格式中,构成格式的位从 0 开始被连续编号。为了错误检测,且优选地为了校正,一个或多个校验位可与每一个字节或一组字节一起被传递。这样的校验位由机器自动生成且不能被程序直接控制。存储容量以字节的数量来表示。当存储器操作数字段的长度由指令的操作码暗示时,字段被称为具有固定长度,其可以是一个、两个、四个、八个或十六个字节。可为某些指令暗示更大的字段。当存储器操作数字段的长度没有被暗示,而是被显式地表示时,该字段被称为具有可变长度。可变长度的操作数可以一个字节的增量(或者对于一些指令,以两个字节倍数或其他倍数)在长度上可变。当信息被放在存储器中时,仅替换被包括在指定的字段中的那些字节位置的内容,即使到存储器的物理路径的宽度可能大于正被存储的字段的长度。

[0097] 某些信息单元位于存储器中的整数界限上。对于信息单元,当其存储器地址是以字节表示的单元长度的倍数时,界限被称为是整数的。特殊的名称被给予整数界限上的 2、4、8 和 16 字节的字段。半字是两字节边界上的一组两个连续的字节,且是指令的基本构件。字是四字节边界上的一组四个连续的字节。双字是八字节边界上一组八个连续的字节。四倍长字(quadword)是 16 字节边界上的一组 16 个连续的字节。当存储器地址指定半字、字、

双字和四倍长字时,地址的二进制表示分别包括一个、两个、三个或四个最右边的零位。指令将位于二字节整数边界上。大多数指令的存储器操作数不具有界限对准要求。

[0098] 在为指令和数据操作数实现单独的高速缓存的设备上,如果程序在高速缓存线中存储且指令被随后从该高速缓存线获取,可经历显著的延迟,不管该存储是否改变随后被获取的指令。

[0099] 在一个实施例中,本发明可被软件(有时候被称为许可的内部代码、固件、微代码、毫代码、微微代码(pico-code)等,其任何一个都将符合本发明)实施。参考图7,体现本发明的软件程序代码可典型地由主系统5000的处理器5001从长期存储介质设备5011(诸如CD-ROM驱动器、磁带驱动或硬盘驱动器)访问。软件程序代码可体现在与数据处理系统一起使用的各种已知介质(诸如软磁盘、硬盘驱动或CD-ROM)中的任何一个上。代码可在这样的介质上被分发,或可从一个计算机系统的计算机存储器5002或存储设备通过网络5010被分发给其他计算机系统的用户,以由这样的其他系统的用户使用。

[0100] 软件程序代码包括操作系统,其控制各种计算机组件和一个或多个应用程序的功能和交互。程序代码通常可从存储介质设备5011调页到相对更高速的计算机存储器5002,在此它对于处理器5001是可用的。用于在存储器中、物理介质上体现软件程序代码和/或经由网络分发软件代码的技术和方法是熟知的,且不会在此被进一步讨论。当程序代码被创建并存储在有形介质(包括但不限于电子存储模块(RAM)、闪存、光盘(CD)、DVD、磁带等)上时,其经常被称为“计算机程序产品”。计算机程序产品介质典型地可由优选地位于计算机系统中的处理电路读取以由处理电路执行。

[0101] 图8示出了可在其中实施本发明的代表性工作站或服务器硬件系统。图8的系统5020包括代表性基本计算机系统(base computer system)5021,诸如个人计算机、工作站或服务器,包括可选的外围设备。根据已知技术,基本计算机系统5021包括一个或多个处理器5026以及被用于连接并使能处理器5026和系统5021的其他组件之间的通信的总线。总线将处理器5026连接到存储器5025以及可包括例如硬盘驱动器(例如,包括磁介质、CD、DVD和闪存中的任何一个)或磁带驱动器的长期存储器5027。系统5021也可包括用户接口适配器,其经由总线将微处理器5026连接到一个或多个接口设备,诸如键盘5024、鼠标5023、打印机/扫描仪5030和/或其他接口设备,其可以是任何用户接口设备,诸如触摸敏感屏、数字化输入垫(digitized entry pad)等。总线也可经由显示适配器将诸如LCD屏幕或监视器的显示设备5022连接到微处理器5026。

[0102] 系统5021可通过能与网络5029通信5028的网络适配器与其他计算机或计算机网络通信。示例性网络适配器是通信通道、令牌环网、以太网或调制解调器。或者,系统5021可使用诸如CDPD(蜂窝数字分组数据)卡的无线接口来通信。系统5021可与局域网(LAN)或广域网(WAN)中的这样的其他计算机关联,或系统5021可以是与另一个计算机的客户机/服务器安排中的客户机等。所有这些配置以及合适的通信硬件和软件在本领域中是已知的。

[0103] 图9示出了其中可实施本发明的一个或多个方面的数据处理网络5040。数据处理网络5040可包括多个单独的网络,诸如无线网和有线网,其每个可包括多个单独的工作站5041、5042、5043、5044。此外,本领域技术人员将理解,可包括一个或多个LAN,其中LAN可包括多个耦合到主处理机的智能工作站。

[0104] 仍然参考图 9, 网络也可包括大型计算机或服务器, 诸如网关计算机(客户机服务器 5046)或应用服务器(远程服务器 5048, 其可访问数据储存库, 且也可直接从工作站 5045 被访问)。网关计算机 5046 用作到每个单独网络的进入点。当将一个联网协议连接到另一个时, 需要网关。网关 5046 可通过通信链路优选地耦合到另一个网络(例如因特网 5047)。也可使用通信链路将网关 5046 直接耦合到一个或多个工作站 5041、5042、5043、5044。可以利用可从国际商业机器公司获得的 IBM eServer™System z® 服务器来实现网关计算机。

[0105] 同时参考图 8 和 9, 可体现本发明的一个或多个方面的软件编程代码可被系统 5020 的处理器 5026 从诸如 CD-ROM 驱动器或硬盘驱动器的长期存储介质 5027 访问。软件编程代码可被体现在与数据处理系统一起使用的各种已知介质(诸如软盘、硬盘驱动器或 CD-ROM)中的任一个上。代码可在这样的介质上被分发, 或从一个计算机系统的存储器或存储设备通过网络被分发到其他计算机系统的用户 5050、5051, 以供这样的其他系统的用户使用。

[0106] 或者, 编程代码可体现在存储器 5025 中, 且由处理器 5026 使用处理器总线访问。这样的编程代码包括操作系统, 其控制各种计算机组件和一个或多个应用程序 5032 的功能和交互。程序代码通常从存储介质 5027 调页到高速存储器 5025, 在此它可用于由处理器 5026 进行处理。用于在存储器中、在物理介质上体现软件编程代码和 / 或经由网络分发软件代码的技术和方法是公知的, 不会在此进一步讨论。程序代码, 当其被创建且在有形介质(包括但不限于电子存储模块(RAM)、闪存、光盘(CD)、DVD、磁带等)上存储时, 通常被称为“计算机程序产品”。计算机程序产品介质典型地可以被优先地位于计算机系统中的处理电路读取以由处理电路执行。

[0107] 最容易被处理器使用的高速缓存(通常比处理器的其他高速缓存更快更小)是最底层(L1 或级别 1)高速缓存, 且主存储(主存储器)是最高级高速缓存(如果有三个级别的話是 L3)。最低级高速缓存经常被分为保持将被执行的机器指令的指令缓存(I- 高速缓存), 和保持数据操作数的数据高速缓存(D- 高速缓存)。

[0108] 参考图 10, 为处理器 5026 示出了示例性处理器实施例。典型地, 使用一个或多个级别的高速缓存 5053 来缓冲存储器块, 以便改善处理器性能。高速缓存 5053 是高速缓冲器, 其保持很可能被使用的存储器数据的高速缓存线。典型的高速缓存线是 64、128 或 256 字节的存储器数据。通常使用单独的高速缓存以用于缓存指令而不是缓存数据。高速缓存一致性(存储器和高速缓存中的线的副本的同步)通常由本领域中熟知的各种“窥探”算法提供。处理器系统的主存储器 5025 通常被称为高速缓存。在具有 4 个级别的高速缓存 5053 的处理器系统中, 主存储器 5025 有时候被称为级别 5 (L5) 高速缓存, 因为它典型地更快, 且仅保持可被计算机系统使用的非易失性存储器(DASD、磁带等)的一部分。主存储器 5025 可“高速缓存”由操作系统向主存储器 5025 调页入或从其调页出的数据页。

[0109] 程序计数器(指令计数器)5061 保持跟踪将被执行的当前指令的地址。**z/Architecture®** 处理器中的程序计数器是 64 位的, 且可被截短为 31 或 24 位以支持先前的寻址界限。程序计数器典型地体现在计算机的 PSW(程序状态字)中, 这样它可在上下文转换中持续。因此, 具有程序计数值的进行中的程序可被例如操作系统中断(从程序环境到操作系统环境的上下文转换)。当程序不活动时, 程序的 PSW 维持程序计数器值, 且在操作系统执行时, 操作系统的(PSW 中的)程序计数器被使用。典型地, 程序计数器以等于当

前指令的字节数的量增量。RISC (精简指令集计算) 指令典型地是固定长度, 而 CISC (复杂指令集计算) 指令典型地是可变长度。**IBMz/Architecture®** 的指令是具有长度为 2、4 或 6 字节的 CISC 指令。程序计数器 5061 被例如上下文转换操作或分支指令的分支采取操作修改。在上下文转换操作中, 当前的程序计数器值与关于正被执行的程序的其他状态信息(诸如条件码)一起被保存在程序状态字中, 且新程序计数器值被载入并指向将被执行的新程序模块的指令。执行分支采取操作, 以通过将分支指令的结果加载到程序计数器 5061 中而允许程序进行决定或在程序内循环。

[0110] 典型地, 使用指令获取单元 5055 代表处理器 5026 获取指令。获取单元可获取“下一序列指令”、分支采取指令的目标指令或上下文转换后的程序的第一指令。现在的指令获取单元通常使用预取技术基于被预取的指令将被使用的可能性来推测性地预取指令。例如, 获取单元可获取 16 字节的指令, 其包括下一顺序指令以及进一步的顺序指令的额外字节。

[0111] 获取的指令随后被处理器 5026 执行。在一实施例中, 获取的指令被传递给获取单元的分派单元 5056。分派单元解码指令并将关于解码的指令的信息转送给合适的单元 5057、5058、5060。执行单元 5057 将典型地从指令获取单元 5055 接收关于解码的算术指令的信息, 并将根据指令的操作码对操作数执行算术操作。优选地从存储器 5025、架构寄存器 5059 或从正被执行的指令的立即字段(immediate field)向执行单元 5057 提供操作数。执行的结果, 当被存储时, 被存储在存储器 5025、寄存器 5059 或其他机器硬件(诸如控制寄存器、PSW 寄存器等)中。

[0112] 处理器 5026 典型地具有一个或多个用于执行指令的功能的单元 5057、5058、5060。参考图 11A, 执行单元 5057 可通过接口逻辑 5071 与架构通用寄存器 5059、解码 / 分派单元 5056、加载存储单元 5060 和其他 5065 处理器单元通信。执行单元 5057 可使用几个寄存器电路 5067、5068、5069 来保持算术逻辑单元(ALU)5066 将操作的信息。ALU 执行诸如加减乘除的算术操作, 以及诸如和、或以及异或(XOR)、旋转和移位的逻辑运算。优选地, ALU 支持依赖于设计的专门操作。其他电路可提供其他架构工具 5072, 例如包括条件码和恢复支持逻辑。典型地, ALU 操作的结果被保持在输出寄存器 5070 中, 该输出寄存器电路可将结果转送到多种其他处理功能。有许多处理器单元安排, 本说明书仅旨在提供对一个实施例的代表性理解。

[0113] 例如, ADD 指令将在具有算术和逻辑功能的执行单元 5057 中被执行, 而例如浮点指令将在具有专用浮点能力的浮点执行中被执行。优选地, 执行单元通过在操作数上执行操作码定义的功能在由指令标识的操作数上操作。例如, ADD 指令可被执行单元 5057 在由指令的寄存器字段标识的两个寄存器 5059 中发现的操作数上执行。

[0114] 执行单元 5057 对两个操作数执行算术加法, 并在第三操作数中存储结果, 其中第三操作数可以是第三寄存器或两个源寄存器中的一个。执行单元优选地利用算术逻辑单元(ALU)5066, 其能执行多种逻辑功能, 诸如移位、旋转、和、或、异或, 以及多种代数函数, 包括加减乘除中的任何一个。有些 ALU5066 被设计为用于标量运算, 有些用于浮点。根据架构, 数据可以是大端(big endien)(其中最低有效字节位于最高字节地址)或小端(little endien)(其中最低有效字节位于最低字节地址)。IBM z/Architecture® 是大端。根据架构, 带符号字段可以是符号和幅度、1 的补码或 2 的补码。2 的补码数是有利的, 其在于

ALU不需要设计减法能力,因为不管是2的补码中的负值还是正值,都仅要求ALU中的加法。数字通常以速记描述,其中12位的字段定义了4096字节块的地址,且通常被描述为例如4Kbyte(千字节)块。

[0115] 参考图11B,用于执行分支指令的分支指令信息典型地被发送到分支单元5058,该分支单元经常使用诸如分支历史表5082的分支预测算法,在其他条件运算完成前预测分支结果。在条件运算完成前,当前分支指令的目标将被获取并推测性地执行。当条件运算完成时,基于条件运算的条件和推测的结果,推测性执行的分支指令或被完成或被丢弃。典型的分支指令可测试条件码,以及如果条件码满足分支指令的分支要求,分支到目标地址,分支地址可基于若干数被计算,所述数包括例如在寄存器字段或是指令的立即字段中找到的数。分支单元5058可利用具有多个输入寄存器电路5075、5076、5077和一个输出寄存器电路5080的ALU5074。分支单元5058可与例如通用寄存器5059、解码分派单元5056或其他电路5073通信。

[0116] 一组指令的执行可由于多个原因中断,所述原因包括例如由操作系统发起的上下文转换、引起上下文转换的程序异常或错误、引起上下文转换的I/O中断信号或多个程序(在多线程环境中)的多线程活动。优选地,上下文转换动作保存关于当前执行的程序的状态信息,且随后加载关于正被调用的另一个程序的状态信息。状态信息可被存储在例如硬件寄存器或存储器中。状态信息优选地包括指向将被执行的下一个指令的程序计数器值、条件码、存储器转换信息和架构寄存器内容。上下文转换活动可被硬件电路、应用程序、操作系统程序或固件代码(微代码、微微代码或许可内部码(LIC))单独地或其组合实现。

[0117] 处理器根据指令定义的方法而访问操作数。指令可使用指令的一部分的值提供立即操作数,可提供一个或多个寄存器字段,其显式地指向通用寄存器或专用寄存器(例如浮点寄存器)。指令可利用由操作码字段确定的暗示的寄存器作为操作数。指令可利用用于操作数的存储器位置。可由寄存器、立即字段或寄存器和立即字段的组合提供操作数的存储器位置,如由**z/Architecture®**长位移工具(facility)所示例的,其中该指令定义了基寄存器、索引寄存器和立即字段(位移字段),它们加到一起,以提供例如存储器中的操作数的地址。除非另外指明,此处的位置典型地意味着主存储器(主存储设备)中的位置。

[0118] 参考图11C,处理器使用加载/存储单元5060访问存储器。加载/存储单元5060可以通过获取存储器5053中的目标操作数的地址并将操作数加载到寄存器5059或其他存储器5053位置中,来执行加载操作,或可以通过获取存储器5053中的目标操作数的地址并将从寄存器5059或另一个存储器5053位置获得的数据存储在存储器5053中的目标操作数位置,来执行存储操作。加载/存储单元5060可以是推测性的,且可以相对于指令顺序来说无序的顺序访问存储器,但是加载/存储单元5060将向程序维持指令按顺序执行的外观。加载/存储单元5060可与通用寄存器5059、解密/分派单元5056、高速缓存/存储器接口5053或其他元件5083通信,且包括各种寄存器电路、ALU5085和控制逻辑5090以计算存储器地址并提供流水线顺序以使操作保持次序。一些操作可不按顺序,但加载/存储单元提供功能以使不按顺序执行的操作对程序看起来如已按顺序执行一样,如本领域所熟知的。

[0119] 优选地,应用程序“看到的”地址通常被称为虚拟地址。虚拟地址有时候被称为“逻辑地址”和“有效地址”。这些虚拟地址之所以虚拟,在于它们由多种动态地址转换

(DAT) 技术中的一种重定向到物理存储器位置,所述动态地址转换技术包括但不限于简单地给用偏移值给虚拟地址加前缀、经由一个或多个转换表转换虚拟地址,所述转换表优选地包括至少一个段表和一个页表(单独地或组合地),优选地,段表具有指向页表的项。在 **z/Architecture** 中,提供转换分级结构,包括区域第一表、区域第二表、区域第三表、段表和可选的页表。地址转换的性能通常通过利用转换后备缓冲器(TLB)被改善,该转换后备缓冲器包括将虚拟地址映射到相关的物理存储位置的项。当 DAT 使用转换表转换虚拟地址时,创建项。于是,虚拟地址的随后使用可利用快的 TLB 的项,而不是慢的顺序转换表访问。TLB 内容可由包括 LRU(最少最近使用)的多个替换算法来管理。

[0120] 在处理器是多处理器系统的处理器的情况下,每个处理器具有保持共享资源的责任,所述共享资源诸如 I/O、高速缓存、TLB 和存储器,它们互锁以实现一致性。典型地,“窥探”技术将被用于维持高速缓存一致性。在窥探环境中,每个高速缓存线可被标记为正处于共享状态、独占状态、改变状态、无效状态等中的一个,以便有助于共享。

[0121] I/O 单元 5054(图 10)向处理器提供用于附加到例如包括磁带、盘、打印机、显示器和网络的外围设备的装置。I/O 单元通常由软件驱动器向计算机程序呈现。在诸如来自 **IBM** 的 System z 的大型计算机中,通道适配器和开放系统适配器是提供操作系统和外围设备之间的通信的大型计算机的 I/O 单元。

[0122] 而且,其他类型的计算环境可受益于本发明的一个或多个方面。作为例子,环境可包括仿真器(例如,软件或其他仿真机制),其中特定架构(包括例如指令执行、诸如地址转换的架构功能、以及架构寄存器)或其子集被仿真(例如,在具有处理器和存储器的本机计算机系统中)。在这样的环境中,仿真器的一个或多个仿真功能可实施本发明的一个或多个方面,即使执行仿真器的计算机可具有与正被仿真的能力不同的架构。作为一个例子,在仿真模式中,解码正被仿真的特定指令或操作,且建立合适的仿真功能以实施单个指令或操作。

[0123] 在仿真环境中,主计算机包括例如存储器以存储指令和数据;指令获取单元以从存储器获取指令,且可选地,提供用于获取的指令的本地缓冲;指令解码单元以接收获取的指令并确定已被获取的指令的类型;以及指令执行单元以执行该指令。执行可包括将数据从存储器加载到寄存器;从寄存器将数据存储回存储器;或执行如由解码单元确定的某些类型的算术或逻辑运算。在一个例子中,每个单元在软件中实现。例如,被所述单元执行的操作被实现为仿真器软件中的一个或多个子例程。

[0124] 更具体地,在大型计算机中,程序员(通常是如今的“C”程序员)一般通过编译器应用使用架构机器指令。存储在存储介质中的这些指令可以在 **z/Architecture** IBM 服务器中本机地执行,或在执行其他架构的机器中执行。它们可在现有的和未来的 **IBM** 大型计算机服务器以及 **IBM** 的其他机器(例如,Power Systems 服务器和 System x 服务器)中被仿真。它们可在使用由 **IBM**、**Intel**、AMD™ 等制造的硬件的各种机器上运行 Linux 的机器中被执行。除了在 **z/Architecture** 下的该硬件上执行,Linux 也可被用于这样的机器,其使用由 TurboHercules (www.trubohercules.com/)、Hercules (www.hercules-390.org/)、UMX 或 FSI (Fundamental Software, Inc) (其中一

般地执行是处于仿真模式中)提供的仿真。在仿真模式中,仿真软件由本机处理器执行以仿真被仿真处理器的架构。

[0125] 本机处理器典型地执行仿真软件,其包括固件或本机操作系统,以执行被仿真处理器的仿真程序。仿真软件负责获取并执行被仿真处理器架构的指令。仿真软件维护仿真的程序计数器以保持跟踪指令界限。仿真软件可一次获取一个或多个仿真的机器指令,并将所述一个或多个仿真的机器指令转换为对应的本机机器指令组,以由本机处理器执行。这些转换的指令可被高速缓存,这样可完成更快的转换。仿真软件将维持被仿真的处理器架构的架构规则以保证为被仿真处理器编写的操作系统和应用正确操作。而且,仿真软件将提供由被仿真的处理器架构确定的资源,包括但不限于控制寄存器、通用寄存器、浮点寄存器、例如包括段表和页表的动态地址转换功能、中断机制、上下文转换机制、日中时间(TOD)时钟和到I/O子系统的架构接口,这样被设计为在被仿真处理器上运行的操作系统或应用程序可在具有仿真软件的本机处理器上运行。

[0126] 解码正被仿真的特定指令,且调用子例程以执行该单个指令的功能。仿真被仿真处理器的功能的仿真软件功能例如在“C”子例程或驱动器中实现,或由提供用于特定硬件的驱动器的其他方法实现,如本领域技术人员在理解优选实施例的描述后将理解的。包括但不限于Beausoleil等人的标题为“Multiprocessor for Hardware Emulation”的美国专利证书号5,551,013;以及Scalzi等人的标题为“Preprocessing of Stored Target Routines for Emulating Incompatible Instructions on a Target Processor”的美国专利证书号6,009,261;以及Davidian等人的标题为“Decoding Guest Instruction to Directly Access Emulation Routines that Emulate the Guest Instructions”的美国专利证书号,5,574,873;以及Gorishek等人的标题为“Symmetrical Multiprocessing Bus and Chipset Used for Coprocessor Support Allowing Non-Native Code to Run in a System”的美国专利证书号6,308,255;以及Lethin等人的标题为“Dynamic Optimizing Object Code Translator for Architecture Emulation and Dynamic Optimizing Object Code Translation Method”的美国专利证书号6,463,582,;以及Eric Traut的标题为“Method for Emulating Guest Instructions on a Host Computer Through Dynamic Recompilation of Host Instructions”的美国专利证书号5,790,825;以及许多其他专利的各种软件和硬件仿真专利示出各种已知的方式来实现针对可为本领域技术人员获得的目标机器对为不同机器进行架构设计的指令格式的仿真。

[0127] 在图12中,提供了仿真主计算机系统5092的例子,其仿真主架构的主计算机系统5000'。在仿真主计算机系统5092中,主处理器(CPU)5091是仿真主处理器(或虚拟主处理器),并包括具有与主计算机5000'的处理器5091不同的本机指令集架构的仿真处理器5093。仿真主计算机系统5092具有可被仿真处理器5093访问的存储器5094。在示例性实施例中,存储器5094被分区为主计算机存储器5096部分和仿真例程5097部分。根据主计算机架构,主计算机存储器5096对于仿真主计算机5092的程序来说是可用的。仿真处理器5093执行与被仿真处理器5091不同架构的架构指令集的本机指令(即来自仿真程序处理器5097的本机指令),且可通过使用从顺序和访问/解码例程获得的一个或多个指令从主计算机存储器5096中的程序访问用于执行的主机指令,所述顺序和访问/解码例程可解码访问的主机指令,以确定用于仿真被访问的主机指令的功能的本机指令执行例程。被定

义用于主计算机系统 5000' 架构的其他工具可被架构工具例程仿真，所述架构工具例程包括诸如通用寄存器、控制寄存器、动态地址转换和 I/O 子系统支持和处理器高速缓存等工具。仿真例程也可利用在仿真处理器 5093 中可获得的功能(诸如通用寄存器和虚拟地址的动态转换)以改善仿真例程的性能。也可提供专用硬件和卸载引擎以辅助处理器 5093 来仿真主计算机 5000' 的功能。

[0128] 在此使用的术语仅是为了描述特定实施例，且不旨在限制本发明。如在此使用的，单数形式“一”、“一个”和“该”也旨在包括复数形式，除非上下文另外清楚地指明。还将理解，当在说明书中使用时，术语“包括”和 / 或“包含”指明存在所述的特征、整体、步骤、操作、元件和 / 或组件，但不排除存在或附加一个或多个其他特征、整体、步骤、操作、元件和 / 或它们的组合。

[0129] 所附权利要求书中的所有装置或步骤加功能元件的相应结构、材料、操作以及等价物，如有的话，旨在包括用于结合如特别要求保护的其他所要求保护的元件来执行所述功能的任何结构、材料或操作。呈现本发明的说明是为了示出和描述的作用，但不是穷尽性的或将本发明限制于所公开的形式。许多修改和变化对本领域普通技术人员来说是明显的，且不脱离本发明的范围。选择和描述实施例是为了最佳地解释本发明的原理和实际应用，并使得本领域普通技术人员能针对适于考虑的特定用途的具有各种修改的各种实施例理解本发明。

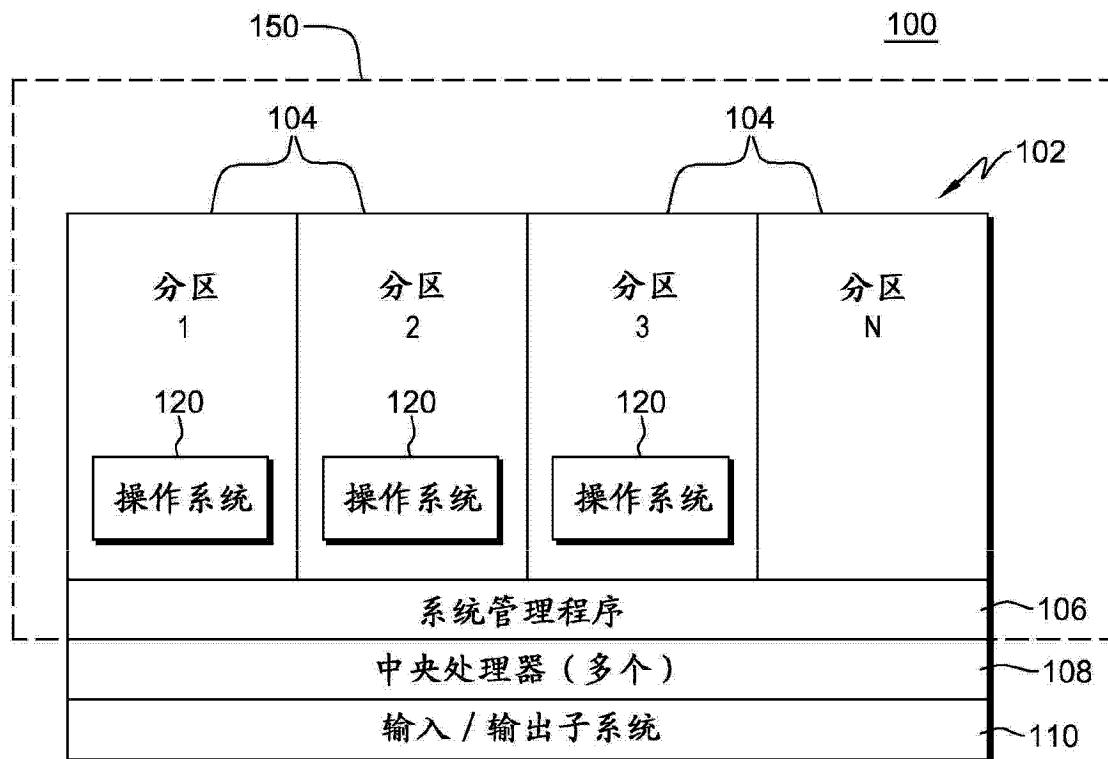


图 1

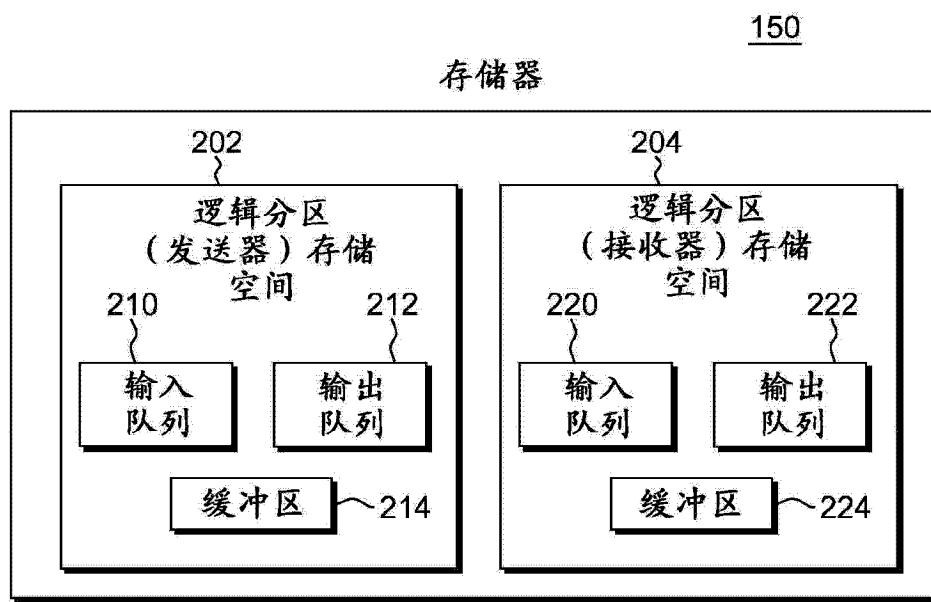


图 2

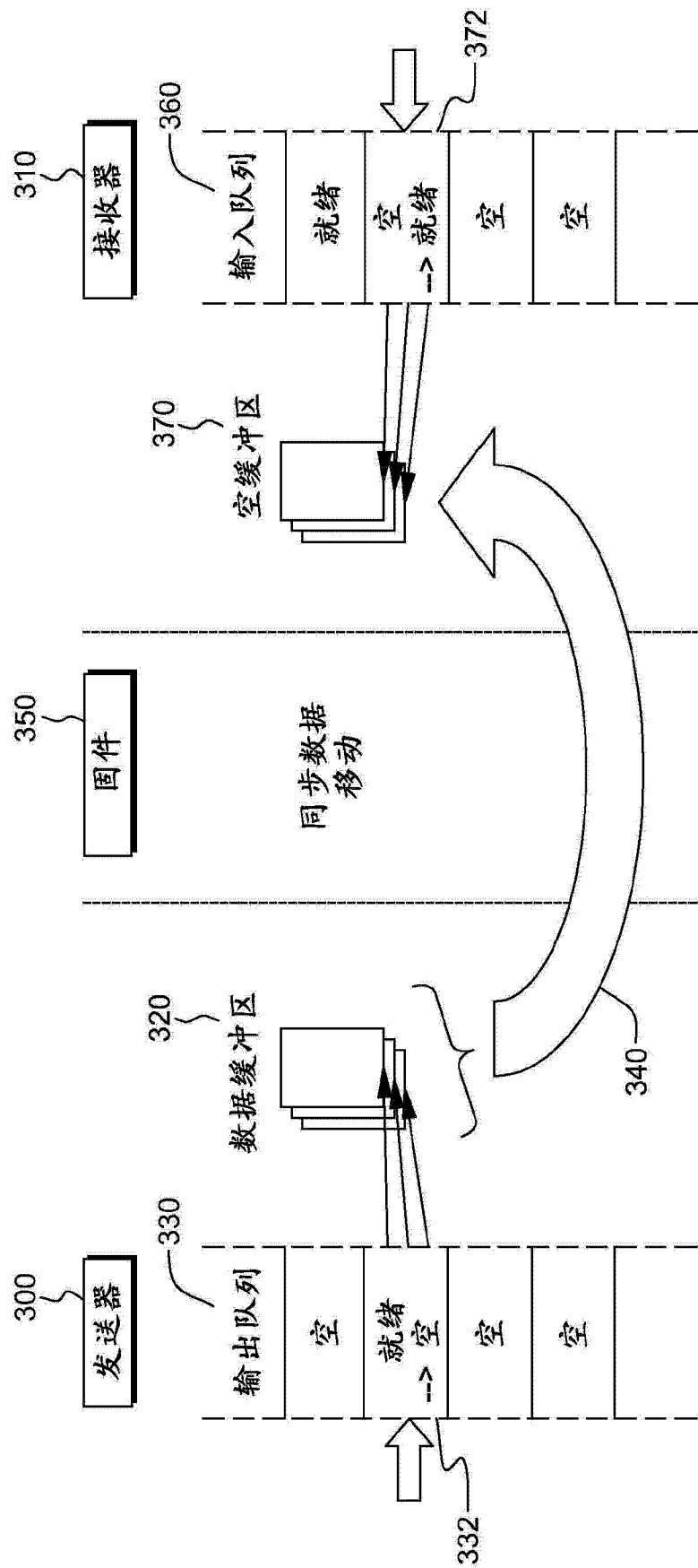


图 3

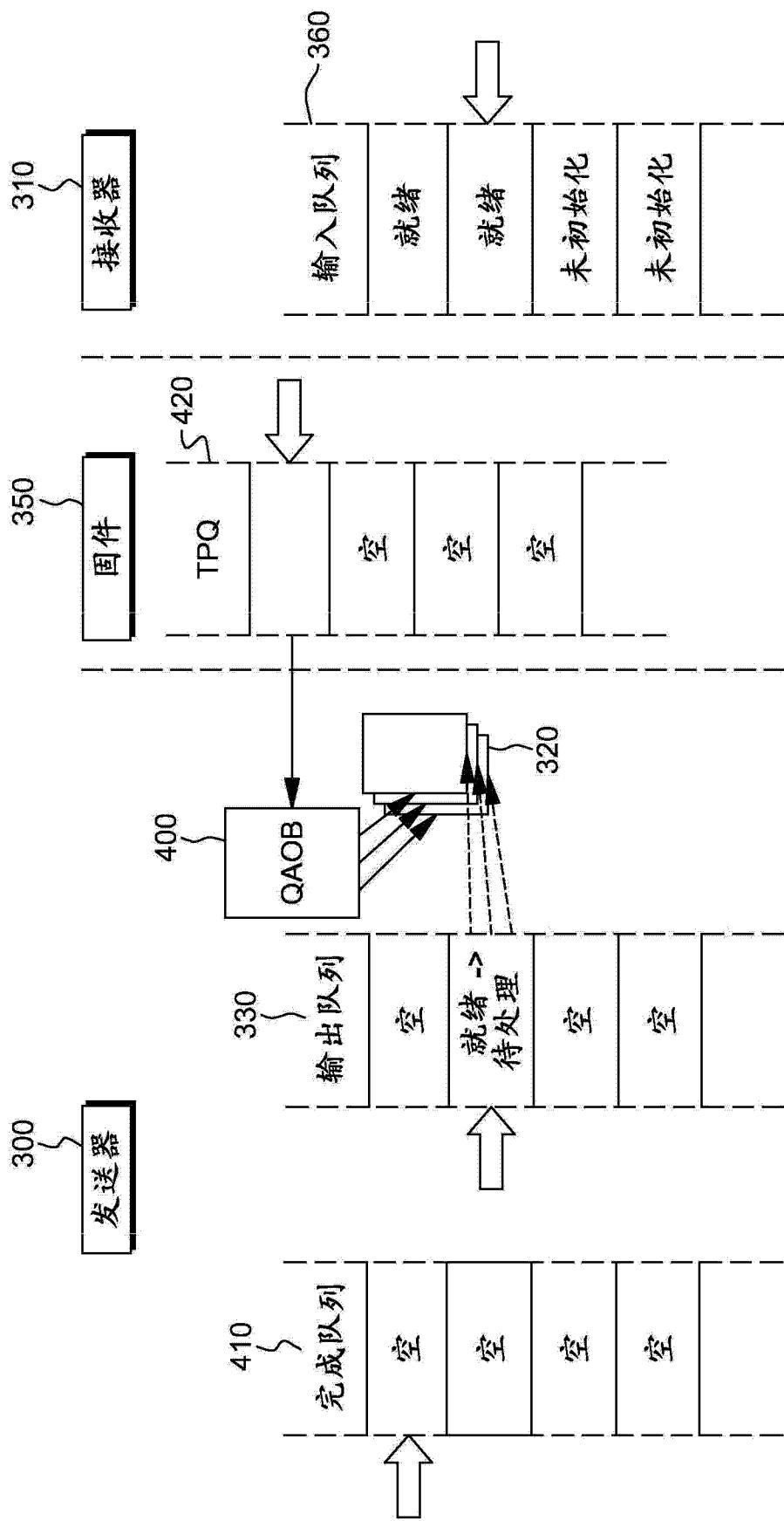


图 4

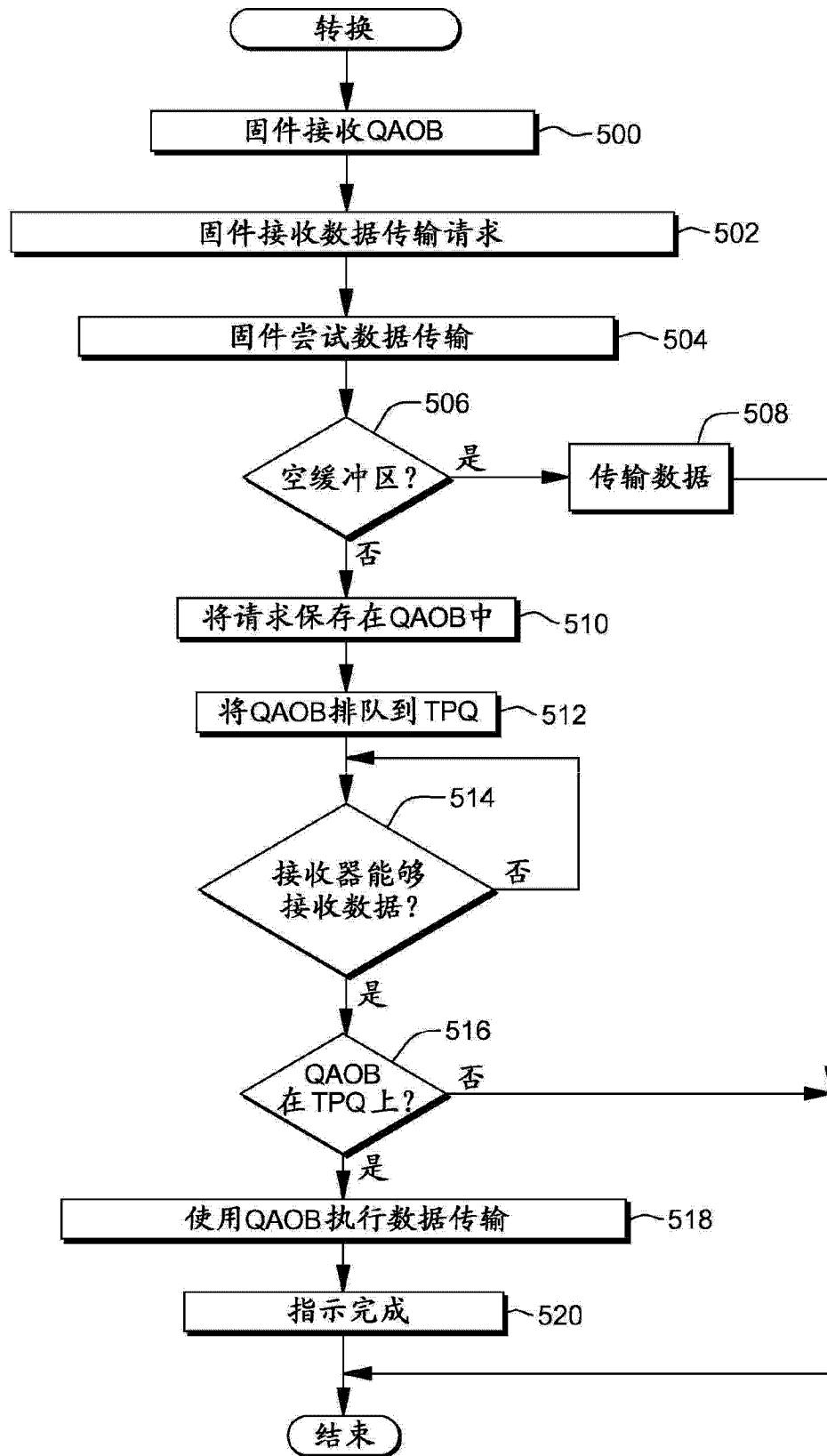


图 5A

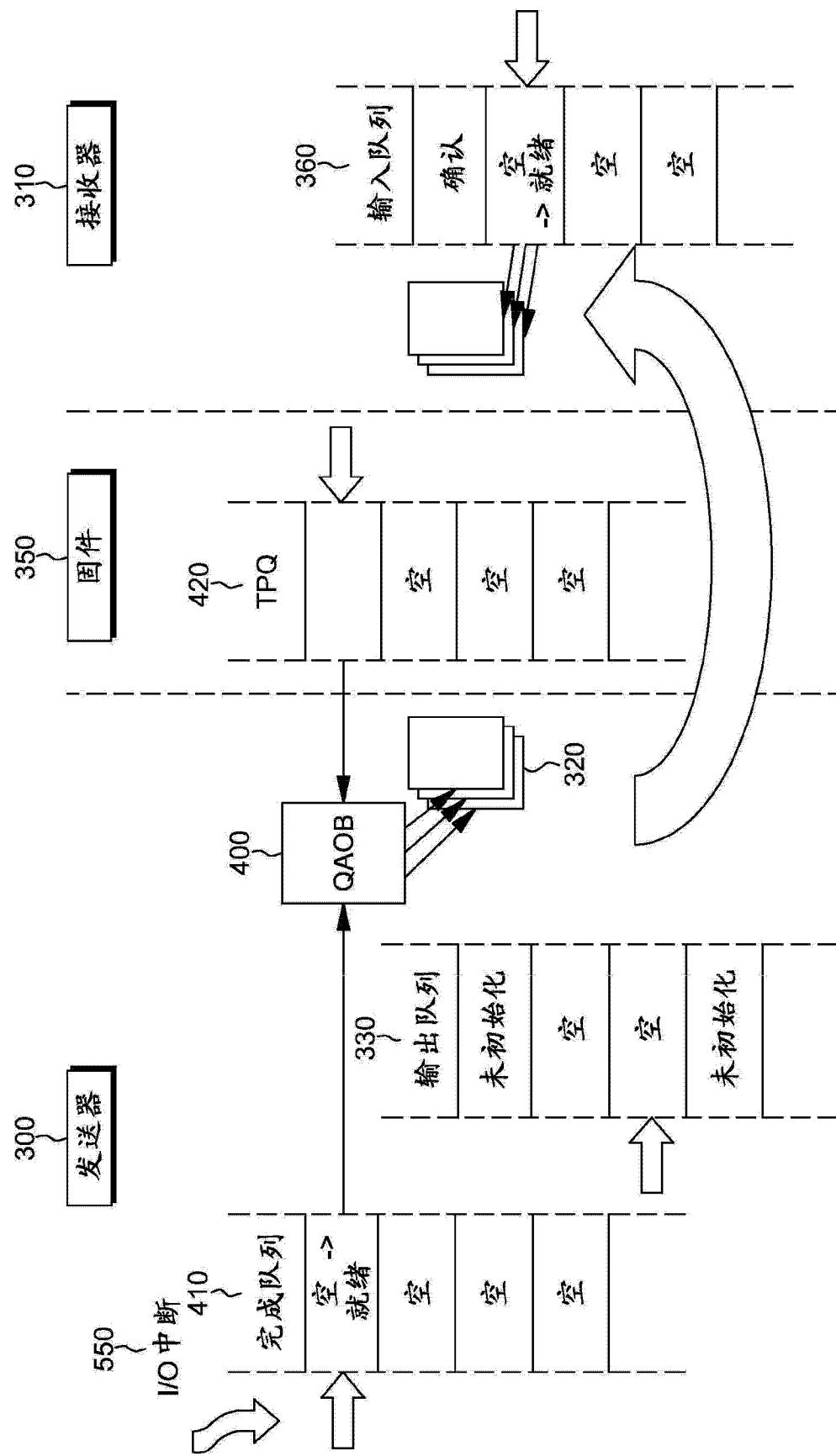


图 5B

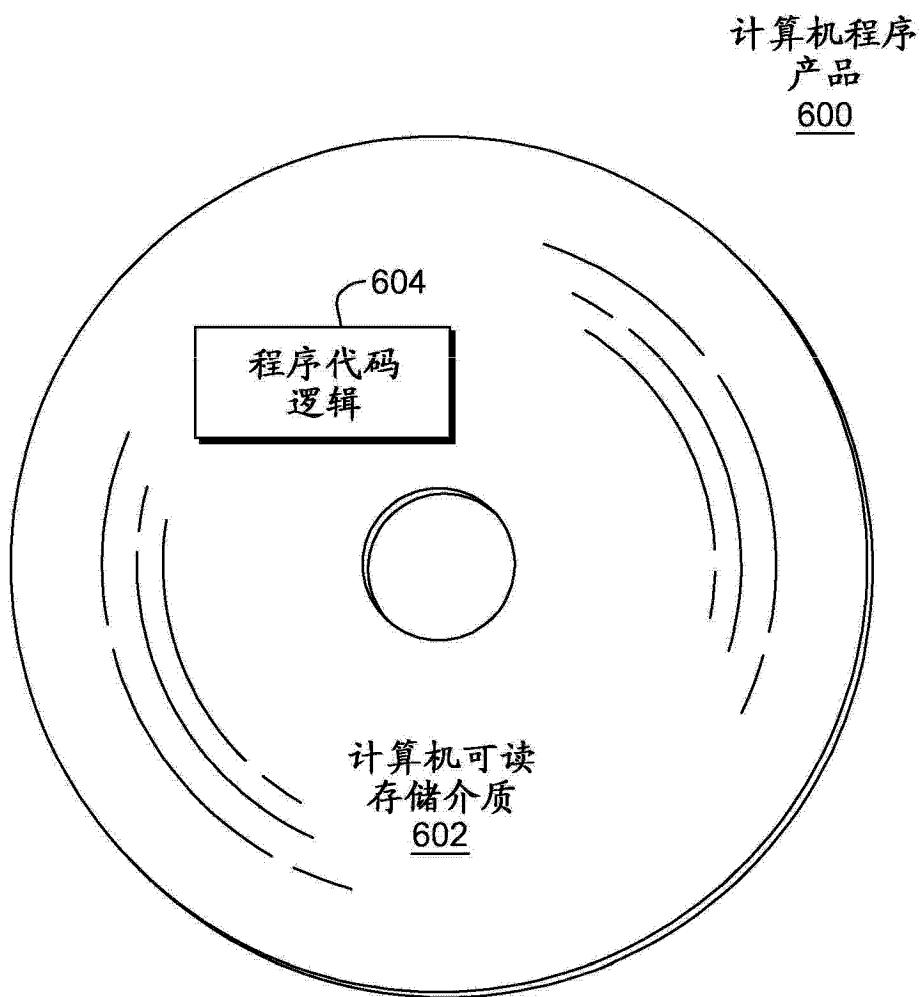


图 6

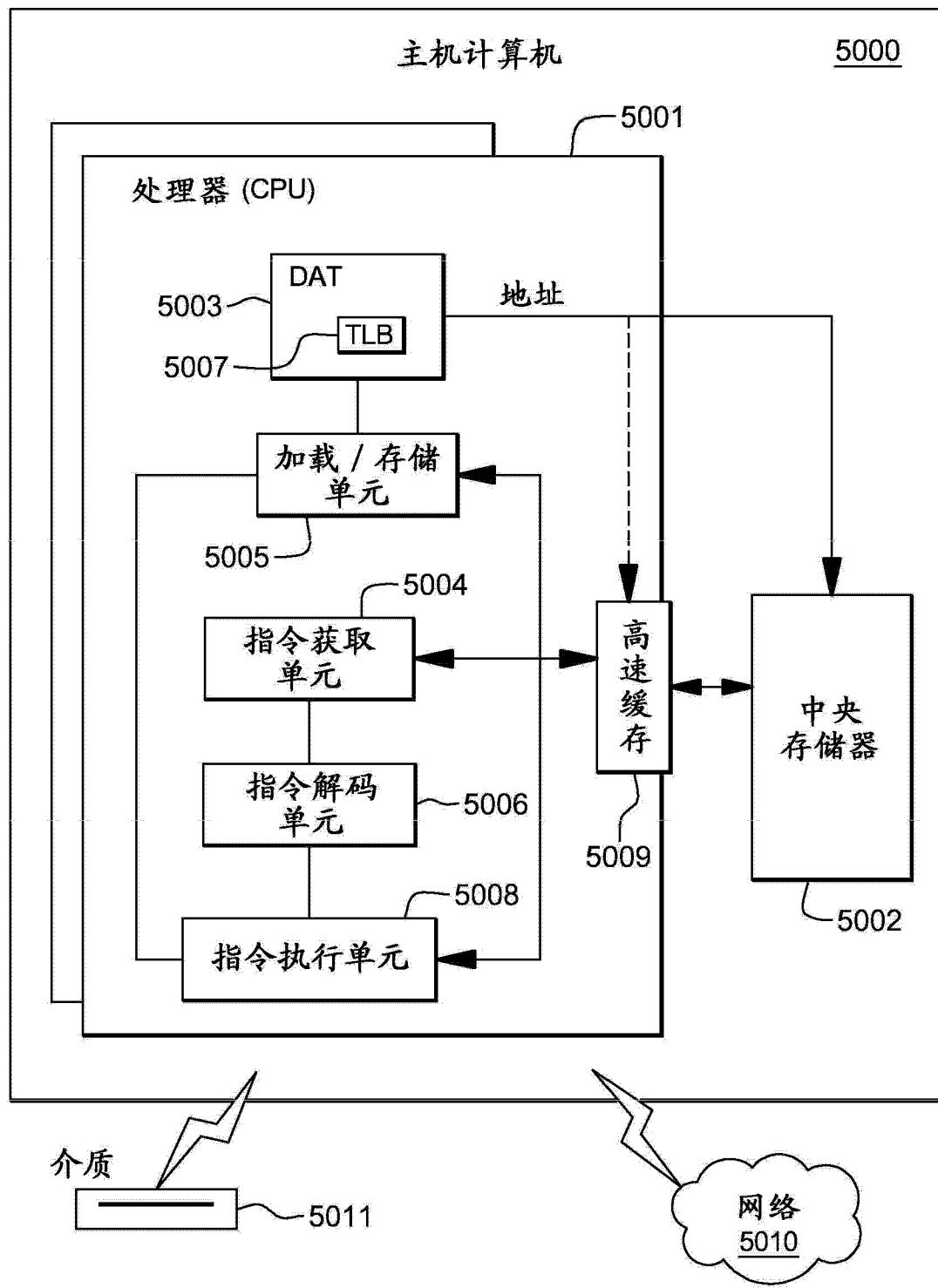


图 7

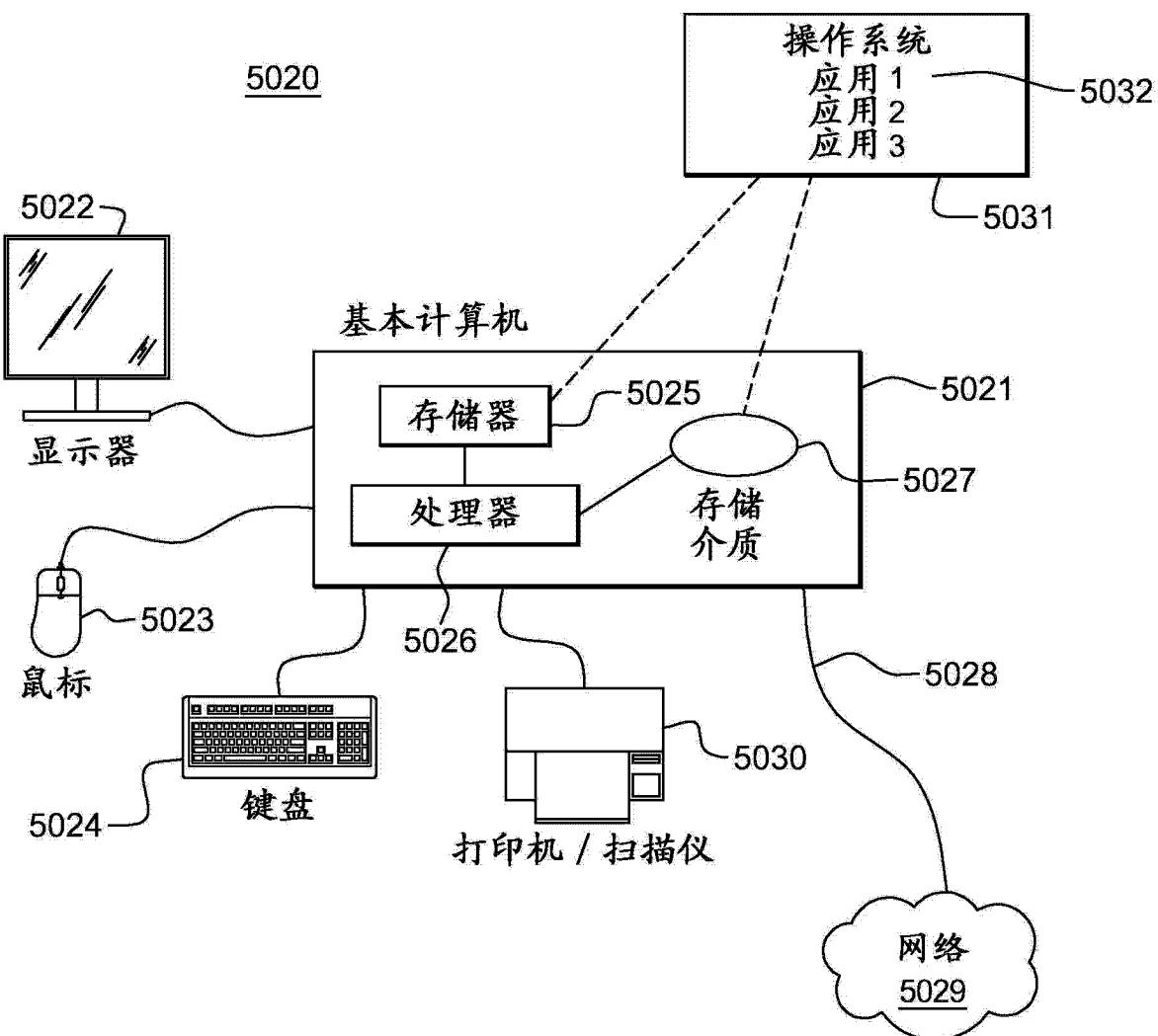


图 8

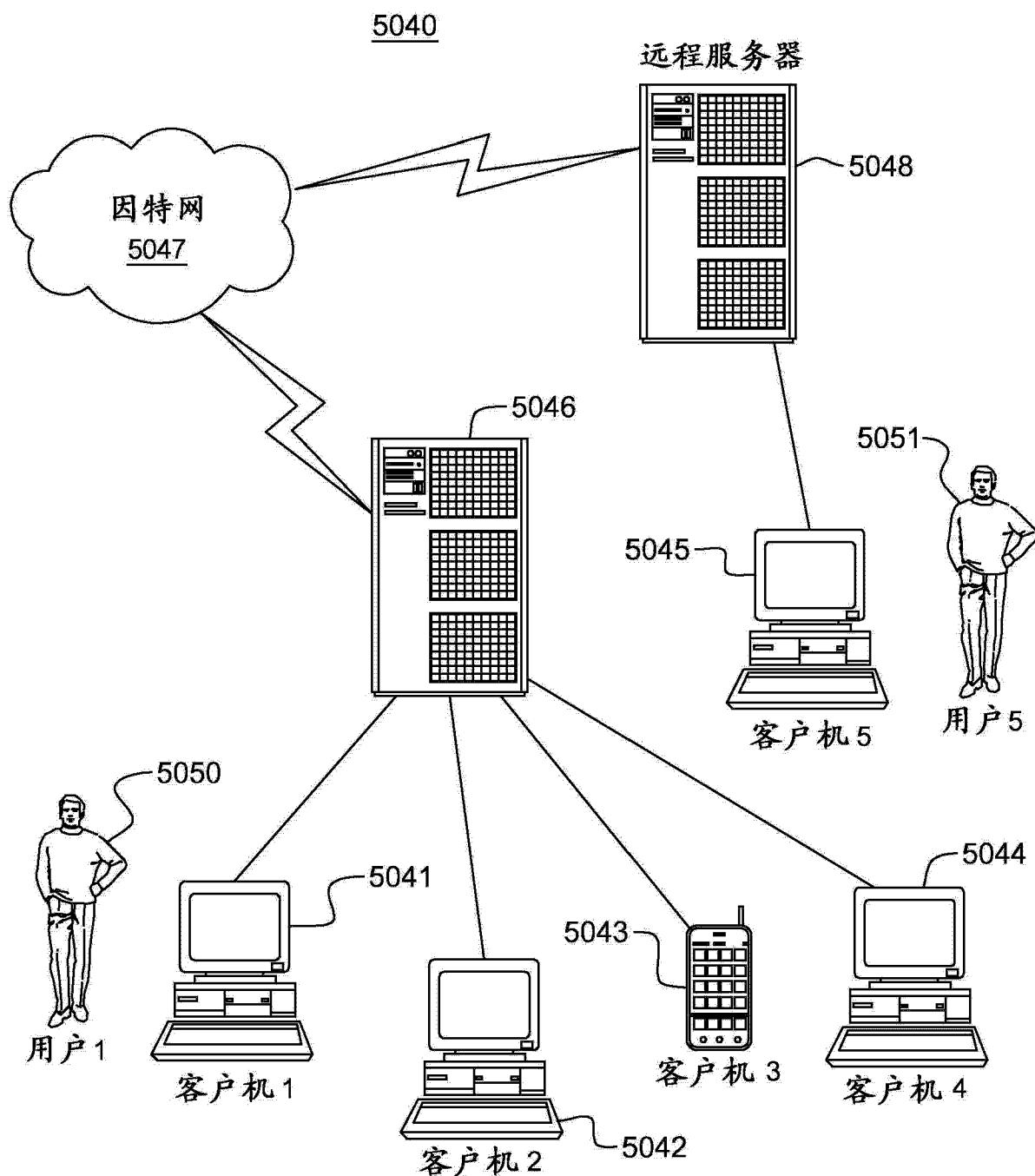


图 9

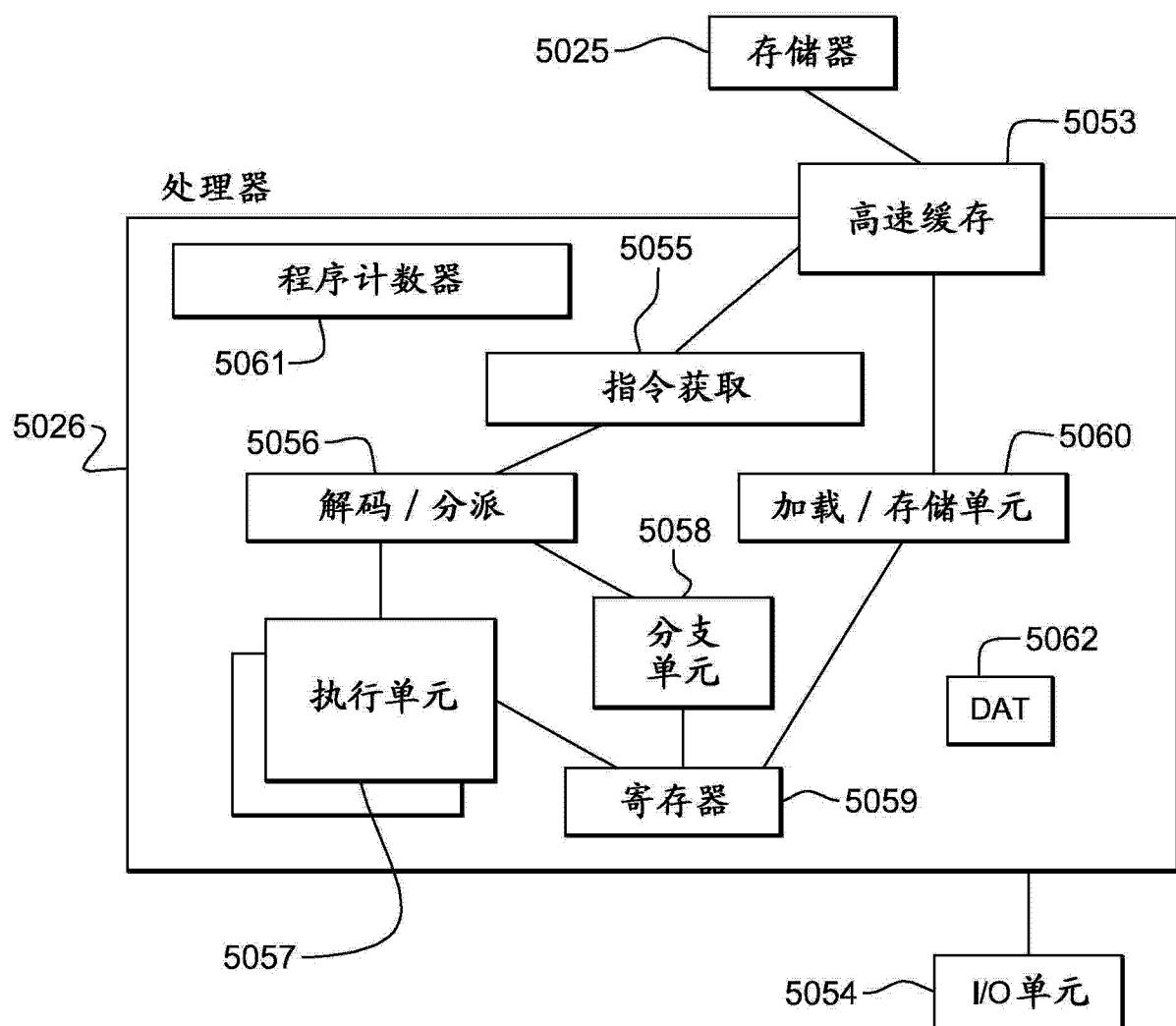


图 10

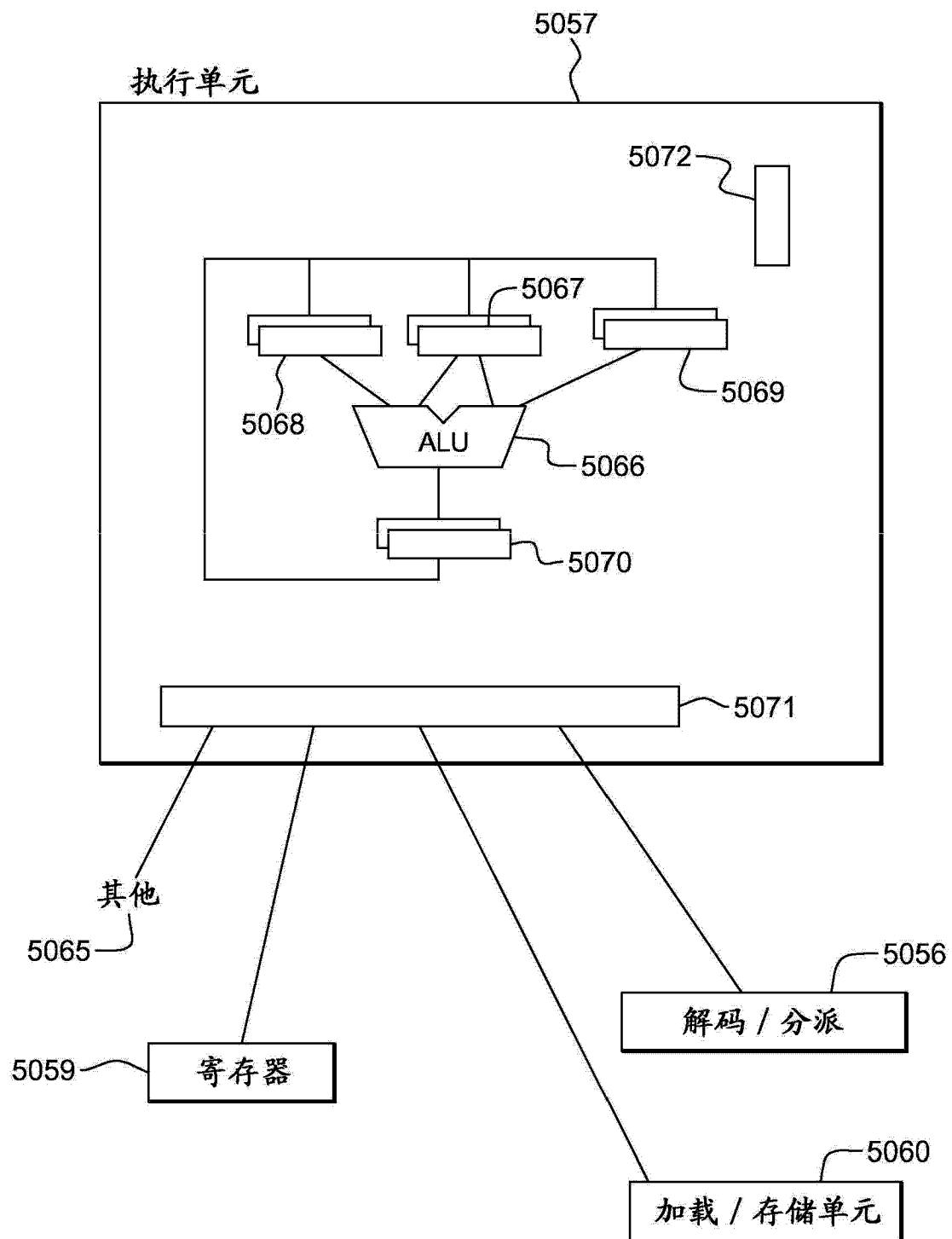


图 11A

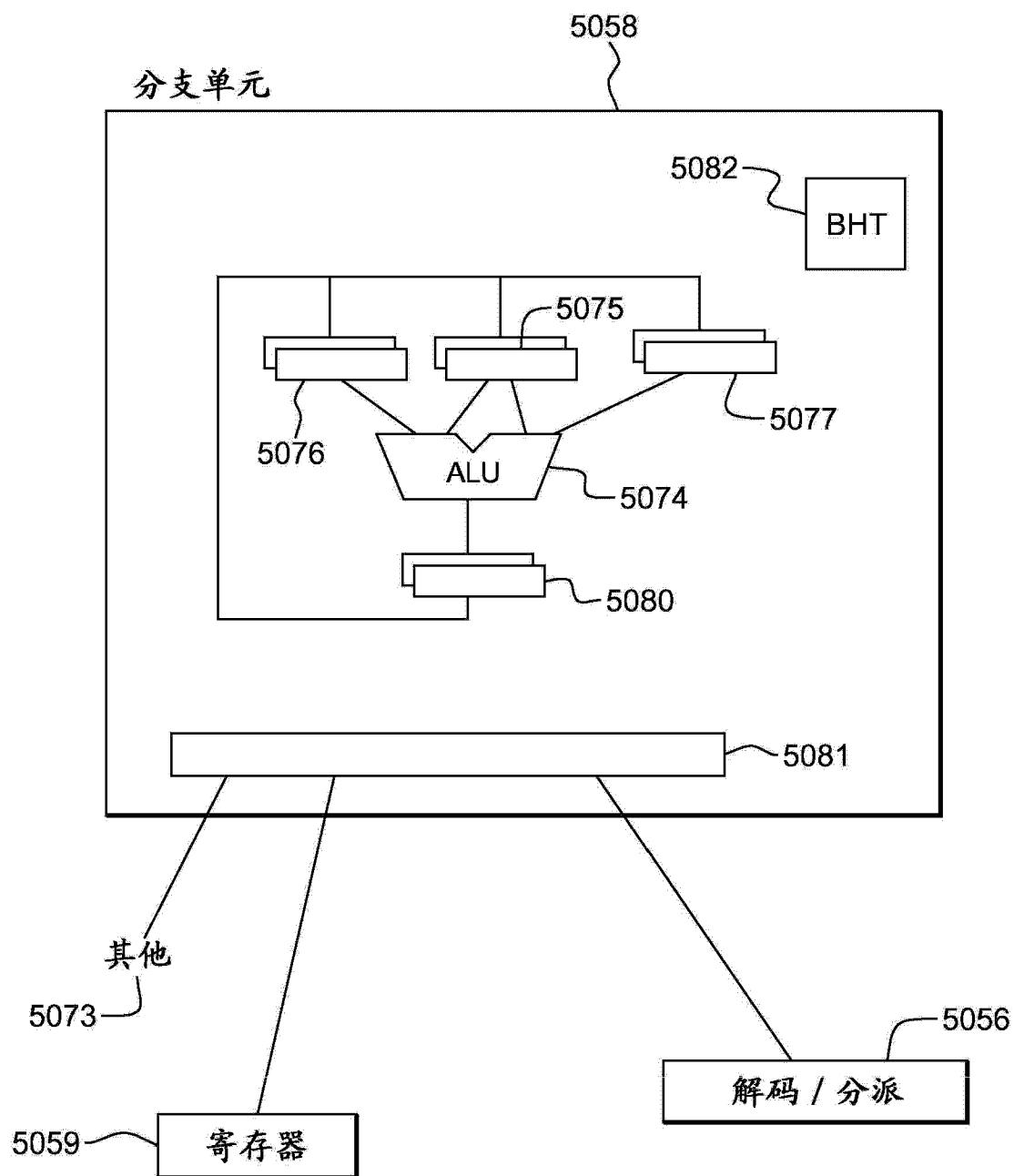


图 11B

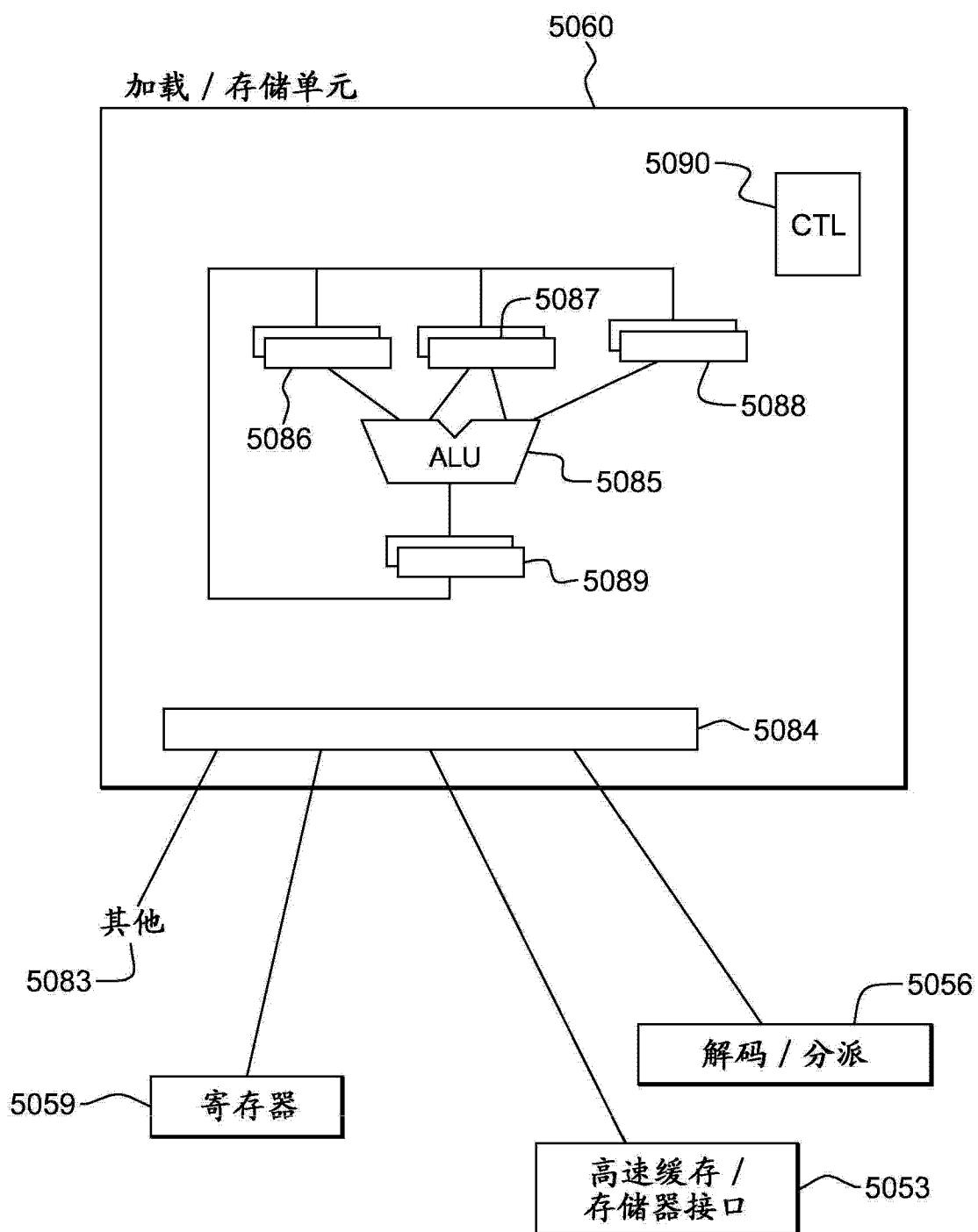


图 11C

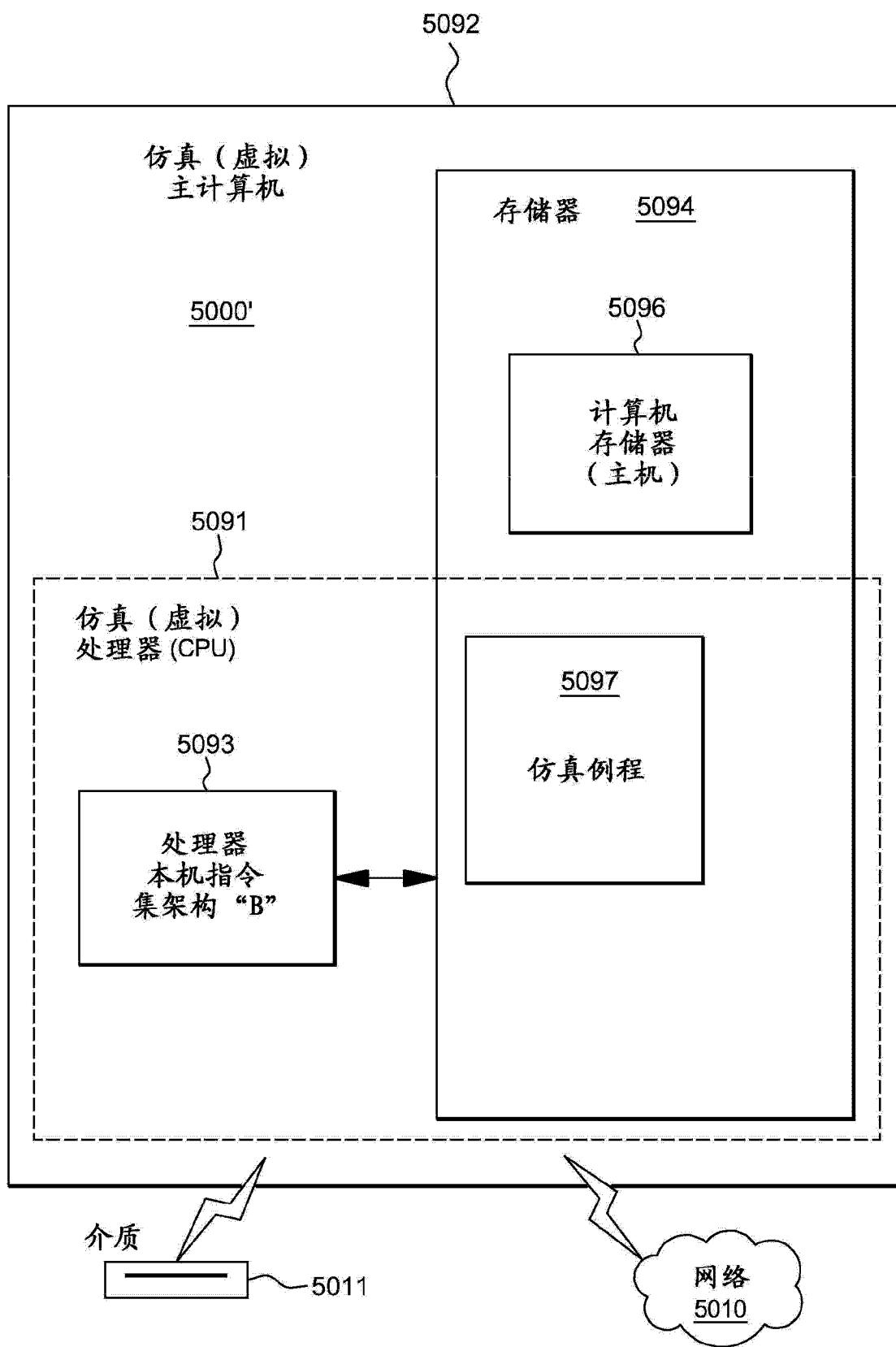


图 12