IB的体系结构及其特征

IB体系结构及其特征

Chapter 1 provided a high-level overview of the key concepts underpinning InfniBand. Some of these concepts include channel I/O (described as a connection between two virtual address spaces), a channel endpoint called a queue pair (QP), and direct application level access to a QP via virtual to physical memory address mapping. A detailed description of the architecture and its accompanying features is the province of an entire multi-volume book and is well outside the scope of this minibook. This chapter is designed to deepen your understanding of the architecture with an eye toward understanding how
InfniBand's key concepts are achieved and how they are applied.

第一章从high-level的角度介绍了IB架构的关键概念。这些概念包括channel I/O（Channel I/O是一个connection，它连接两个应用的虚拟地址空间）、其中channel I/O的端点被叫做Queue Pair（它包含Send Queue和Receive Queue）。本文不会对IBA进行详细的描述，这超出了该mini文档的范围。这个章节的主要目的是加深读者对IBA一些关键概念的理解，不主要包含两个方面：（1）IBA的关键概念怎样被实现？（2）IBA的关键概念怎样被应用？

- Address Translation

也许和其它的网络技术不同，IB有两个最主要的概念：

- 应用程序直接访问IB Messaging Service服务
- Messaging Service服务被使用直接在不同应用程序的虚拟地址空间中传输数据

上述两个特征之所以能够生效主要是因为IB的地址解析机制。因此，地址解析是支撑IB特征的关键技术。注意：在这里地址解析是指虚拟地址到物理地址的解析，它发生在服务器的虚拟地址空间内；它和传统的NAT（Network Address Translation）技术不同。

IBA作为Virtual Interface Architecture（VIA）的一部分，而VIA的关键是“Virtual interface”在应用程序和底层Messaging Service服务支架被创建。QP作为Channel端点的逻辑表示。通过使用地址映射，QP被直接映射到应用程序的虚拟地址空间中，这使得应用程序能够直接访问channel I/O。

物理的I/O channel实际上被底层的HCA（Host Channel Adapter）创建。QP充当应用程序和底层IB HCA的物理连接，因此，QP可以被看作物理IB HCA的接口。QP被映射到应用程序的虚拟地址空间并提供和底层HCA的接口。HCA创建物理communication channel。因此，当应用程序提交一个Work Request到Send Queue时，它实际上被物理HCA控制。HCA携带request请求到合适的远端目的地（我的理解：这实际上就建立了不同应用程序之间的物理连接）。

因为HCA占据了一定的物理地址范围，因此，需要虚拟地址到物理地址的转换。应用程序通过内存请求必要的地址转换注册过程，之后HCA使用地址转换表进行所需的虚拟到物理地址转换。由于操作系统拥有地址转换表的所有权，这保证了每个应用程序虚拟地址空间的隔离性和受保护性。甚至地址转换表不直接参与应用程序和HCA直接的交互。由于HCA支持许多的QPs，并且每个QP可以被注册给一个不同的应用，很显然，单个的HCA能够给大量的应用同时提供服务。从应用程序的角度来看，HCA似乎已经虚拟化了应用程序。将QP映射进程序的虚拟地址空间中，这使得应用程序能够访问IB的硬件。

The InfiniBand transport protocol enables this sort of “cross-server” direct communication by providing for the exchange of a set of virtual addresses and keys between a local application and a remote application. This combination of a virtual address and a key, which is provided by the remote (responding) application, is used by the local (initiating) application to perform an RDMA Read or RDMA Write operation directly into or out of the remote application’s virtual buffer. The remote application, at the time it passes the virtual addresses and keys to the local application, is effectively passing control of that buffer to the local application, which returns control of the buffer to the remote application after the data transfer is completed. This transfer of control of the remote application’s virtual buffer via the passing of a virtual address and key combination ensures that no third party, such as another application, can intrude on the channel connecting the local and remote applications. Channel I/O delivers private, protected channels for the exclusive use of the applications connected by that channel.

IB能够实现跨服务器连接，虚拟地址+keys的结合（由响应应用程序提供）被使用。远程应用程序提供虚拟地址+keys的结合给局部应用程序，这使得远程应用程序将

buffer的控制权交给局部应用程序，当完成数据传输之后，buffer的控制权将返回给远程应用程序。虚拟地址和key的结合，使得没有第三方应用能够在本地和远端应用程序之间强行加一个channel连接。Channel I/O在一对应用程序之间提交一个私有的、受保护的channels。

A SEND/RECEIVE operation works as follows. A remote (receiving) application uses a Post Receive Request verb to build one or more WRs (work requests) which are placed on a receive queue. Each RECEIVE WR represents a receive buffer in the application's virtual address space. A local (sending) application uses a Post Send Request verb to place one or more WRs on a send queue. Each WR on the send queue represents a message that the applications wishes to be sent to the application at the other end of the channel; a SEND operation targets a buffer represented by a WR on the receive queue at the opposite end of the channel. The sending application does not know the virtual address of the buffer to which the SEND is targeted; instead the SEND is targeted to the receiver QP's receive queue, which contains a pointer to the application's buffer. This rather complex sounding operation is sometimes referred to as a "Channel Semantic." SEND/RECEIVE operations are frequently used, for example to transfer short control messages. [来自 Introduction to InfniBand™ for End Users 文档第五章，也就是文档的Page 41]

接下来将给出SEND/RECEIVE操作的工作流程。接收端的应用程序使用POST RECEIVE VERB去构建一个或者多个Work Request（WR），这些新构建的Work Request被放在Receive Queue中。每一个RECEIVE WR表示一个应用程序虚拟地址空间的接收buffer。发送方应用程序使用POST SEND VERB去放一个或者更多个Work Request到Send Queue中。每个在Send Queue中的WR都表示一个应用程序希望发送给另外一个应用程序的"消息"。发送方应用程序根本不知道接收方程序的虚拟地址空间。发送方的发送目标是QP的receive queue，它包含了一个指向接收方应用程序的buffer。SEND/RECEIVE被频繁的使用，尤其是在传递短的控制信息时被频繁用到。

Beyond the channel semantic operations (SEND/RECEIVE), InfniBand provides a unique service called RDMA. There is no equivalent transport service available on a TCP/IP network. An RDMA operation is sometimes also referred to as a "Memory semantic." This is because the local application is able to directly read or write a virtual memory location (e.g. a buffer) provided by the remote application. Before the local application can execute an RDMA READ or WRITE, it must be in possession of a key and the virtual address of the target buffer provided by the remote application. The actual manipulation of the remote application’s virtual buffer is performed by its HCA, using the virtual address and key provided as part of the RDMA operation launched by the local application. RDMA operations are often used for bulk data transfers.

IB提供了成为RDMA的服务。它是和TCP/IP不相同的传输服务。RDMA操作有时也被称作内存原语，这主要是因为local应用程序能够直接读取remote端的虚拟内存空间的buffer。在执行RDMA READ和RDMA WRITE之前，remote端必须要提供buffer+keys给local端。远程应用程序的虚拟缓冲区的实际操作由其HCA执行，使用作为本地应用程序启动的RDMA操作的一部分提供的虚拟地址和密钥。RDMA操作经常被使用去传输大量的数据。

Consider the example of a storage access. Assume that a file system (which we will call the local application), wishes to write a block of data to a block storage device (which we will call the remote application). Assume that an appropriate “channel” (i.e. association between the QPs at the local and remote application ends) has already been created. To initiate the block write, the file system places the data to be written into a data buffer in its local virtual address space. The data buffer had been previously registered with the HCA, which in turn provided a “key” to the file system. Next, the file system constructs a SEND operation. The SEND operation contains a protocol specifc command (e.g. SCSI block write), an extent and a virtual address and key. This virtual address and key together represent the buffer at the file system end containing the data to be written. In effect, the file system has passed control of that buffer to the storage device. At the same time, the file system also places a WR on its receive queue, because it knows that the storage device will soon be returning ending status.

考虑一个存储访问实例。假设当前存在一个文件系统（在这里我们将其看作local应用程序），它希望写一块数据到一个块存储设备上（我们将这个程序看作remote应用程序），并且假设当前一个channel已经被创建好。文件系统（local应用程序）将需要被写的数据放在自己的buffer中（该buffer想要使用毕业要registration），并且还要提供一个key。接下来文件系统发送SEND操作。这就代表着local应用程序已经将buffer的控制权拿给remote应用程序，remote应用程序通过Receive Queue拿到数据。

The storage device, in turn, uses the virtual address and key to construct an RDMA Read operation in order to read the data from the fle system’s virtual data buffer. Once the RDMA Read is complete and the data has been written to disk, the storage device uses a SEND operation to send ending status back to the fle system. The receipt of the ending status message by the file system completes the operation.

remote应用程序拿到virtual address+keys后，发起RDMA READ操作从local应用陈旭中读取数据。数据传输完成之后，Remote应用程序将发起Send操作，将结束状态信息告知local应用程序，local应用程序接收到结束状态之后，也会在自己的completion queue中发送完成信息。

说明

本文的内容大部分翻译自：Introduction to InfniBand™ for End Users 文档第五章。

参考文献

1. https://www.mellanox.com/pdf/whitepapers/Intro_to_IB_for_End_Users.pdf