IBM Virtual Management Channel Kernel Driver (IBMVMC)

Authors: Dave Engebretsen <engebret@us.ibm.com>
Adam Reznechek <adreznec@linux.vnet.ibm.com>
Steven Royer <seroyer@linux.vnet.ibm.com>
Bryant G. Ly <bryantly@linux.vnet.ibm.com>

Introduction

Note: Knowledge of virtualization technology is required to understand this document.

A good reference document would be:

https://openpowerfoundation.org/wp-content/uploads/2016/05/LoPAPR_DRAFT_v11_24March2016_cmt1.pdf

The Virtual Management Channel (VMC) is a logical device which provides an interface between the hypervisor and a management partition. This interface is like a message passing interface. This management partition is intended to provide an alternative to systems that use a Hardware Management Console (HMC) - based system management.

The primary hardware management solution that is developed by IBM relies on an appliance server named the Hardware Management Console (HMC), packaged as an external tower or rack-mounted personal computer. In a Power Systems environment, a single HMC can manage multiple POWER processor-based systems.

Management Application

In the management partition, a management application exists which enables a system administrator to configure the systemâ€™s partitioning characteristics via a command line interface (CLI) or Representational State Transfer Application (REST API's).

The management application runs on a Linux logical partition on a POWER8 or newer processor-based server that is virtualized by PowerVM. System configuration, maintenance, and control functions which traditionally require an HMC can be implemented in the management application using a combination of HMC to hypervisor interfaces and existing operating system methods. This tool provides a subset of the functions implemented by the HMC and enables basic partition configuration. The set of HMC to hypervisor messages supported by the management application component are passed to the hypervisor over a VMC interface, which is defined below.

The VMC enables the management partition to provide basic partitioning functions:

- Logical Partitioning Configuration
- Start, and stop actions for individual partitions
- Display of partition status
- Management of virtual Ethernet
- Management of virtual Storage
- Basic system management

Virtual Management Channel (VMC)

A logical device, called the Virtual Management Channel (VMC), is defined for communicating between the management application and the hypervisor. It basically creates the pipes that enable virtualization management software. This device is presented to a designated management partition as a virtual device.

This communication device uses Command/Response Queue (CRQ) and the Remote Direct Memory Access (RDMA) interfaces. A three-way handshake is defined that must take place to establish that both the hypervisor and management partition sides of the channel are running prior to sending/receiving any of the protocol messages.

This driver also utilizes Transport Event CRQs. CRQ messages are sent when the hypervisor detects one of the peer partitions has abnormally terminated, or one side has called H_FREE_CRQ to close their CRQ. Two new classes of CRQ messages are introduced for the VMC device. VMC Administrative messages are used for each partition using the VMC to communicate capabilities to their partner. HMC Interface messages are used for the actual flow of HMC messages between the management partition and the hypervisor. As most HMC messages far exceed the size of a CRQ buffer, a virtual DMA (RMDA) of the HMC message data is done prior to each HMC Interface CRQ message. Only the management partition drives RDMA operations; hypervisors never directly cause the movement of message data.

Terminology

RDMA
: Remote Direct Memory Access is DMA transfer from the server to its client or from the server to its partner partition. DMA refers to both physical I/O to and from memory operations and to memory to memory move operations.

CRQ

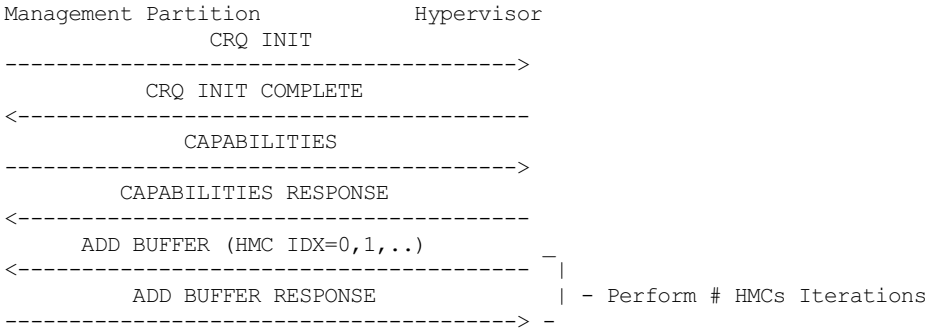
Command/Response Queue a facility which is used to communicate between partner partitions. Transport events which are signaled from the hypervisor to partition are also reported in this queue.

Example Management Partition VMC Driver Interface

This section provides an example for the management application implementation where a device driver is used to interface to the VMC device. This driver consists of a new device, for example /dev/ibmvmc, which provides interfaces to open, close, read, write, and perform ioctlâ€™s against the VMC device.

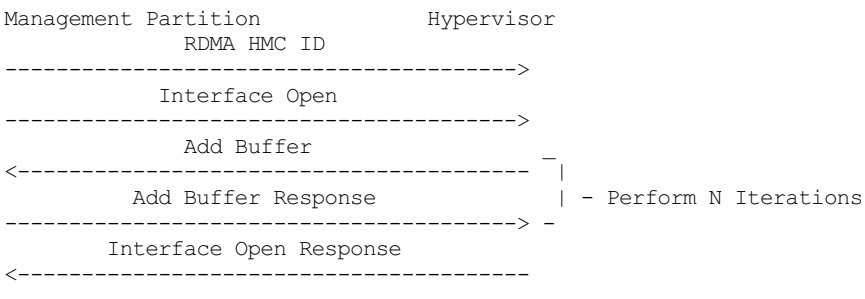
VMC Interface Initialization

The device driver is responsible for initializing the VMC when the driver is loaded. It first creates and initializes the CRQ. Next, an exchange of VMC capabilities is performed to indicate the code version and number of resources available in both the management partition and the hypervisor. Finally, the hypervisor requests that the management partition create an initial pool of VMC buffers, one buffer for each possible HMC connection, which will be used for management application session initialization. Prior to completion of this initialization sequence, the device returns EBUSY to open() calls. EIO is returned for all open() failures.



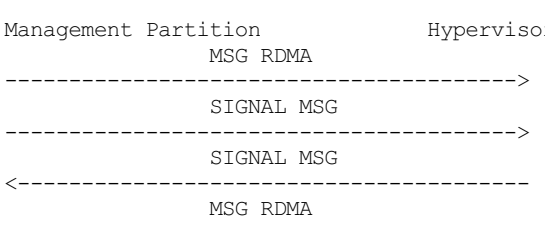
VMC Interface Open

After the basic VMC channel has been initialized, an HMC session level connection can be established. The application layer performs an open() to the VMC device and executes an ioctl() against it, indicating the HMC ID (32 bytes of data) for this session. If the VMC device is in an invalid state, EIO will be returned for the ioctl(). The device driver creates a new HMC session value (ranging from 1 to 255) and HMC index value (starting at index 0 and ranging to 254) for this HMC ID. The driver then does an RDMA of the HMC ID to the hypervisor, and then sends an Interface Open message to the hypervisor to establish the session over the VMC. After the hypervisor receives this information, it sends Add Buffer messages to the management partition to seed an initial pool of buffers for the new HMC connection. Finally, the hypervisor sends an Interface Open Response message, to indicate that it is ready for normal runtime messaging. The following illustrates this VMC flow:



VMC Interface Runtime

During normal runtime, the management application and the hypervisor exchange HMC messages via the Signal VMC message and RDMA operations. When sending data to the hypervisor, the management application performs a write() to the VMC device, and the driver RDMAâ€™s the data to the hypervisor and then sends a Signal Message. If a write() is attempted before VMC device buffers have been made available by the hypervisor, or no buffers are currently available, EBUSY is returned in response to the write(). A write() will return EIO for all other errors, such as an invalid device state. When the hypervisor sends a message to the management, the data is put into a VMC buffer and an Signal Message is sent to the VMC driver in the management partition. The driver RDMAâ€™s the buffer into the partition and passes the data up to the appropriate management application via a read() to the VMC device. The read() request blocks if there is no buffer available to read. The management application may use select() to wait for the VMC device to become ready with data to read.



```
<----------------------------------------
```

VMC Interface Close

HMC session level connections are closed by the management partition when the application layer performs a close() against the device. This action results in an Interface Close message flowing to the hypervisor, which causes the session to be terminated. The device driver must free any storage allocated for buffers for this HMC connection.

```
Management Partition            Hypervisor
             INTERFACE CLOSE
---------------------------------------->
        INTERFACE CLOSE RESPONSE
<----------------------------------------
```

Additional Information

For more information on the documentation for CRQ Messages, VMC Messages, HMC interface Buffers, and signal messages please refer to the Linux on Power Architecture Platform Reference. Section F.