Vol. 36 No. 5 Transactions of Information Processing Society of Japan May 1995

Regular Paper

Methods for Consistency of Channel-Path-Reconnection with Direct I/O-Execution

HIDENORI UMENO,† TARO INOUE †† and SHUNJI TANAKA ††

Methods are presented for the consistency of the Channel-Path-Reconnection (CPR) with the Direct I/O-Execution (DIO). The CPR is the native function of the channel subsystem, and dynamically reconnects devices to free channel-paths at ends of I/O operations. The CPR is effective in a channel-path-group, which is a set of channel-paths, and defined by an OS to use it. The CPR is the key function to enhance I/O response. On the other hand, the DIO means that real host computers directly execute the I/O instructions and I/O interrupts of virtual machines (VMs) without intervention of a virtual machine monitor (VMM), which is a control program for controlling the VMs. The DIO is the key feature to improve VM CPU performance. Conventionally, in shared channel-paths by the VMs, guest OSs (i.e., OSs in VMs) cannot use the CPR and the DIO at the same time. This restriction comes from uniqueness requirement of a channel-path: A channel-path with the CPR has to belong to only one channel-path-group. This requirement is reasonable in real machine environment. In VM environment, multiple guest OSs are going to define their own channel-path-groups that contain shared channel-paths. The definitions do not satisfy the above-stated uniqueness requirement. Therefore, they are rejected and the guest OSs cannot use the CPR to the shared channel-paths, though they can use the DIO for other I/O instructions. The proposed methods consist of handshakings, which mean modifying OSs for VMs, and several hardware extensions for channel-path-groups and channel-path modes. Channel-path-groups are recognized by their identifiers. In the handshakings, the VMM gives the OSs channel-path-group identifiers that satisfy the uniqueness requirement of channel-paths based on the forms of channel-path allocation. One of the hardware extensions eases the uniqueness requirement of channel-paths, that is, the uniqueness is required only in guest OSs, and not in a total system. The extension enables the guest OSs to compose their own channel-path-groups that contain shared channel-paths under the DIO. Any one of the proposed methods can provide consistency of the CPR with the DIO to the VMs whether channel-paths are shared by the VMs or not.

1. Introduction

Virtual Machine Systems (VMSs) run multiple Operating Systems (OSs) concurrently in a single real computer[12]. VMSs define multiple logical computers called virtual machines (VMs), which can run concurrently in a single real computer[12]. A control program of a VMS is called a virtual machine monitor (VMM). A real computer that runs the VMS is called a real host computer. An OS can run in a VM in the same way as in a real machine except for its lower performance.

The practical use of VMs is dependent on their performance. Real host computers support VM architectures as their features[3)-7)]. The host computers can directly execute most control instructions of VMs by hardware, and the hardware support of VM architectures has greatly improved VM CPU performance. One of the most important techniques to improve VM performance is the Direct I/O-Execution (DIO)[3),5)] which means that VM I/O instructions and its I/O interrupts are executed by hardware/microprogram without intervention of software, that is, the VMM. The purpose of the direct I/O-execution is to boost VM CPU performance up to near-native performance.

A computer system contains a channel subsystem, which controls data transfer between main storage of CPU and I/O devices. The channel subsystem has the function of the Channel-Path-Reconnection (CPR), which dynamically reconnects disk devices to free channel-paths at the ends of device operations, such as seek, search, and read/write operations[9),10)], and enhances I/O response of the devices[11)]. The CPR is not always consistent with the DIO. The reason is as follows:

The channel subsystem has multiple channel-paths through which it sends/receives control signals and data to/from devices. An OS has to

† Software Development Department, General Purpose Computer Division, Hitachi Ltd.
†† Department 3, Systems Development Laboratory, Hitachi Ltd.

define a channel-path-group, which is a group of channel-paths, to use the CPR. An OS issues I/O instructions to define a channel-path-group in a device. The channel subsystem and I/O control unit (IOC) dynamically reconnect a device to channel-paths within the channel-path-group in the device. That is, the CPR is effective within the channel-path-group in the device. An OS uses the CPR to enhance I/O response. The more channel-paths are available in the channel-path-group, the better is the response.

A channel-path with the CPR has to belong to only one channel-path-group in a system[10]. This is called uniqueness requirement of channel-paths, and is reasonable in real machine environment because of the following: An OS defines a unique maximal channel-path-group in the system, and defines its subsets in devices. Multiple real systems can share IOCs and devices through their own dedicated channel-paths. The channel-path-group of one system is exclusive with that of another system. That is, they do not share any channel-path. I/O control units can easily determine their channel-path-groups by using this uniqueness requirement.

On the other hand, in virtual machine environment run multiple guest OSs in a real system. In general, they are going to define their own channel-path-groups that contain shared channel-paths in the real system. These definitions are rejected because they do not satisfy the uniqueness requirement of the shared channel-paths. If one guest OS first issues I/O instructions to make its own channel-path-group, they are directly and successfully executed, and the first guest OS can use both the CPR and the DIO. When another OS secondly issues the I/O instructions to make its own channel-path-group, which contains a channel-path shared with the first OS, they are rejected because they do not satisfy the above-stated uniqueness requirement. That is, the shared channel-path belongs to the channel-path-group the first guest OS has already defined. Therefore, all the guest OSs different from the first OS cannot compose their own channel-path-groups that contain shared channel-paths, and cannot use the CPR[3]. They can use the DIO for other I/O instructions than those to define channel-path-groups. That is, in this case the CPR is not consistent with the DIO.

To this problem, conventionally, a set of channel-paths is dedicated to VMs. OSs in the VMs can directly, (i.e., with the DIO) compose their own channel-path-groups by using the dedicated set of channel-paths because this dedication satisfies the uniqueness requirement of channel-paths. Therefore, the OSs can use the CPR and the DIO simultaneously in the dedicated set of channel-paths[2]. This is not preferable, however, because the channel-path dedication exceedingly decreases the flexibility of system configurations.

We provide new methods for providing consistency of the CPR with the DIO, whether channel-paths are shared by VMs or not. Their outlines are as follows. First is handshaking: Channel-path-groups are recognized by their identifiers OSs define. In the handshaking, the VMM determines the identifiers that satisfy the uniqueness requirement based on the sharing forms of the channel-paths. Second is a full extension of hardware functions[13],[14]. It eases the uniqueness requirement. That is, the uniqueness of channel-paths has to be satisfied in a VM, and may not in a total system. The extension enhances I/O control units to manage different channel-path-groups and different channel-path modes to all combinations of devices, channel-paths, and VMs. The first method modifies OSs, and does not modify hardware. The full hardware extension method modifies hardware, and does not modify OSs. We also present other two intermediate methods.

The above-mentioned problems are stated in detail in section 2. The new methods to solve them are presented in section 3. The methods proposed are evaluated in section 4. Conclusion is presented in Section 5.

2. Problems of Channel-Path-Reconnection in VMs

Figure 1 shows the relation of computer system components, such as instruction processors (IPs), which execute instructions (e.g., add, subtract, start I/O) main storage, I/O processors (IOPs), which control data transfer through channel-paths between the main storage and I/O devices (e.g., disk units and Magnetic Tape (MT) units). It also shows I/O Control units (IOCs), such as disk control unit and MT control unit, and disk units, and MT units connected to their IOCs.

In this section, we explain conventional functions of channel-path-reconnection and direct I/O-execution, and describe the problems we are to solve.

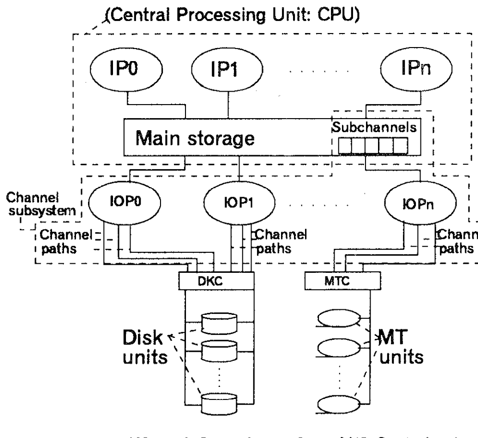

(Central Processing Unit: CPU)
IP0
IP1
IPn
Main storage
Subchannels
Channel subsystem
IOP0
IOP1
IOPn
Channel paths
Channel paths
Channel paths
DKC
MTC
Disk units
MT units
DKC: Disk Control unit: One of I/O Control unit
MTC: MT Control unit: One of I/O Control unit
IP: Instruction Processor
IOP: I/O Processor

Fig. 1 Computer system components.

2.1 Channel-Path-Reconnection

We explain channel-path-reconnection, related subsystems and concepts :

(1) Channel subsystem

As **Fig. 1** shows, a computer system contains a channel subsystem, which consists of IOPs, channel-paths, subchannels, and their control logic, controlling data transfer through the channel-paths between main storage of CPU and I/O devices. OSs issue I/O instructions to I/O devices. Some of them specify a sequence of Channel Command Words (CCWs), which specify I/O operations to be executed by the devices, as one of their operands, and send start signals to the channel subsystem.

(2) Subchannels

The channel subsystem manages subchannels, which are in main storage system area as shown in Fig. 1, and contain channel-path identifiers and device numbers. To a combination of a channel-path and an I/O device connected to it, a subchannel can be defined. When an I/O device is connected to several channel-paths, several subchannels can be defined to the same I/O device. Each subchannel contains status information of the associated I/O device viewed from the associated channel-paths. We have two types of OSs. To the type 1 OS, a subchannel is an object that contains status from a channel-path to a device connected to it[8]. To a type 2 OS, a subchannel is an object that one to one corresponds to an I/O device[9].

(3) Channel-path-reconnection

The channel subsystem has multiple channel-paths through which it sends/receives control signals and data to/from devices. The channel subsystem and I/O control unit (IOC) have the function of the Channel-Path-Reconnection (CPR), which dynamically reconnects disk devices to free channel-paths at the ends of device operations, such as seek, search, and read/write operations[9],[10], and enhances I/O response of the devices[11].

An OS has to define a channel-path-group, which is a group of channel-paths, to use the CPR. An OS issues I/O instructions to define a channel-path-group in a device. The channel subsystem and I/O control unit (IOC) dynamically reconnect a device to free channel-paths within the channel-path-group in the device. That is, the CPR is effective within the channel-path-group in a device. An I/O control unit dynamically selects a free channel-path within the designated channel-path-group in a device and it reconnects the device to the free channel-path. Therefore, an I/O interrupt of a device may occur through a channel-path different from a channel-path through which the device is initially started.

An OS uses the CPR to enhance I/O response. The more channel-paths are available in the channel-path-group, the better is the response. An OS defines a unique maximal channel-path-group in the system, and defines its subsets in devices. Two real systems can share IOCs and devices through their own dedicated channel-paths. In multiple real systems any channel-path-group of one system is exclusive with that of another system. That is, they do not share any channel-path.

(4) Channel-path modes

The channel subsystem and IOC make the CPR effective to a channel-path-group only when all its channel-paths have multiple channel-path mode. When a channel-path does not belong to any channel-path-group or when it has a single path mode, the channel subsystem and the IOC do not apply the CPR to the channel-path. As above-stated we have two types of OSs. A type 2 OS supports the CPR, and a type 1 OS does not support it.

To enhance reliability with reservation of devices, A type 1 OS may compose a channel-path-group, which contains only single path mode channel-paths, to use the function of extended reserve/release, where the OS can reserve a device via one channel-path and later release it via another channel-path in the channel-path-group. A type 2 OS can also use the function of extended reserve/release in its channel-path-group to enhance the reliability.

A type 2 OS always uses the CPR. Therefore, all the channel-paths in its channel-path-group have multiple path mode. A type 1 OS never uses the CPR. Therefore, all the channel-paths in its channel-path-group have single path mode. Therefore, in a real machine environment any channel-path in a channel-path-group has only one channel-path mode.

2.2 Direct I/O-Execution for VMs

The Direct I/O-Execution (DIO) is supported as follows. Subchannels are dedicated to a VM beforehand[6),7)]. Most I/O instructions issued by a running OS in a VM is directly executed by hardware/microprogram without intervention of the VMM, when the subchannel designated by the I/O instruction is dedicated to the running VM. Moreover, an I/O interrupt subclass is dedicated to a VM beforehand[6),7)], and is set in a subchannel. An I/O interrupt request of a subchannel is directly executed if the subclass of the subchannel is dedicated to the running VM and the VM is enabled for it.

2.3 Problems

The native function of the channel-path-reconnection of the channel subsystem should be available for OSs in the VMs as in real machines. It is not always available, however. When they use the direct I/O-execution in shared channel-paths, VMs cannot conventionally use the full function of the native channel subsystems. The reasons are explained in detail :

(1) Uniqueness requirement of channel-Paths

A channel-path with the CPR has to belong to only one channel-path-group[10)] in a system. This is called uniqueness requirement of a channel-path. This requirement is reasonable because of the following : Multiple real systems will define their channel-path-groups by using their own channel-paths. The channel-path-group of one system is exclusive with that of another system. That is, they do not share any channel-path. In addition I/O control units can easily determine their channel-path-groups by using this requirement.

On the other hand, in VM environment run multiple guest OSs in a real system. In general, they are going to define their own channel-path-groups that contain channel-paths shared by the guest OSs in the real system. The definitions are rejected because of the uniqueness requirement of the shared channel-paths. If one guest OS first issues I/O instructions to define its own channel-path-group, they are directly and successfully executed, and the first guest OS can use both the CPR and the DIO. When another OS secondly issues the I/O instructions to define its own channel-path-group that contain a shared channel-path, they are rejected because the second definition does not satisfy the uniqueness requirement. That is, the shared channel-path belongs to the channel-path-group the first guest OS has already defined. Therefore, all the guest OSs different from the first OS cannot compose their own channel-path-groups containing shared channel-paths. In this case, the guest OS may fail, because the guest OS may find two contradictory events : One is that it cannot compose its channel-path-group, the other is that it has recognized that I/O control units have the function of the channel-path-reconnection. Otherwise, in the direct I/O-execution, the guest OS may run without channel-path-groups to the dedicated subchannels, that is, the guest OS cannot use the CPR[3)]. Therefore, though the VM CPU performance may be improved by the DIO, the VM I/O response will deteriorate.

Furthermore, we discuss this problem in view of the identifiers of channel-path-groups. A channel-path-group is recognized by its identifier an OS defines for it. For example, one OS in a VM is going to specify its own channel-path-group, whose identifier is (id1), for a channel-path (path-1). Another OS in a VM is also going to specify its own channel-path-group, whose identifier is (id2) different from (id1) to the same channel-path (the path-1). This does not satisfy the uniqueness requirement of a channel-path : No channel-path can belong to the two different channel-path-groups in the system. An OS can compose channel-path-groups in devices. When they have a channel-path in common, two channel-path-groups are merged into one large channel-path-group. Therefore, they have to have the same identifier[10)].

To support CPR to all guest OSs, the VMM composes a unique channel-path-group in the real host system. A guest OS issues I/O instructions to compose its own channel-path-group. It also issues other I/O instructions to request general I/O operations, such as read/write operations. The VMM cannot know which I/O instructions the guest OS has issued. Therefore, the VMM intercepts every I/O instruction of the guest and simulates it. The VMM composes a channel-path-group in place of the guest OS as a subset of the unique channel-path-group, which the VMM has already composed, according to the request of the guest OS. Then, the guest OS can use the CPR through the simulation of the VMM. However, this simulation increases I/O simulation overhead and is contrary to the concept of the direct I/O-execution.

(2) Uniqueness requirement of channel-path modes

A channel-path with the CPR has to have only one channel-path mode. This requirement is reasonable in a real machine environment, because an OS specifies only one channel-path mode to all channel-paths in its channel-path-group.

On the contrary, the requirement has a large problem in a VM environment. This is because the type 1 OS will compose its channel-path-group with the single path mode to enhance reliability. On the other hand, the type 2 OS will compose its channel-path-group with the multiple path mode to enhance I/O response. Therefore, the type 1 OS and the type 2 OS will specify different path modes to their shared channel-paths. This does not satisfy the uniqueness requirement of channel-path modes.

(3) Conventional methods for consistency

Conventionally, a set of channel-paths is dedicated to a VM to get consistency of the CPR with the DIO in the VM[2]. By using the dedicated set of channel-paths, an OS in the VM can directly, that is, with the direct I/O-execution, compose its own channel-path-groups with its own identifiers and path modes[2]. This is because the dedication satisfies the uniqueness requirement of channel-paths and of path modes. Therefore, the VM can use both the CPR and the DIO simultaneously in the dedicated set of channel-paths. This dedication is not preferable, however, because this channel dedication exceedingly decreases the flexibility of system configurations.

2.4 Channel-Path Allocation Forms

Here, we will discuss the allocation forms of channel-paths and devices, investigating that to what form of the allocation the DIO and the CPR should be supported simultaneously.

(1) Dedicated channel-paths

Figure 2 shows a set of channel-paths (0, 1) dedicated to VM 1 and a device (A), which is dedicated to VM1 and connected to the set of channel-paths. It also shows a set of channel-paths (2, 3) dedicated to VM 2 and a device

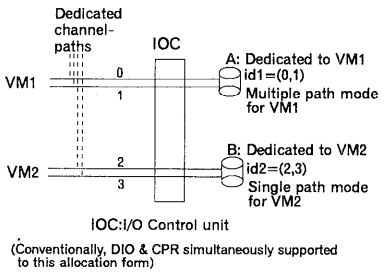


Fig. 2 Dedicated channel-paths & dedicated devices.

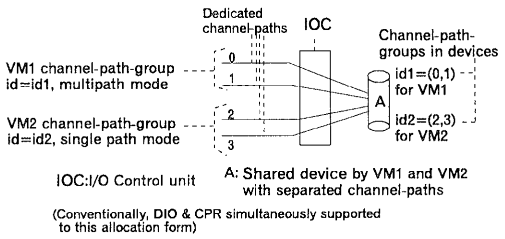

Dedicated channel-paths
IOC
Channel-path-groups in devices
VM1 channel-path-group id=id1, multipath mode
VM2 channel-path-group id=id2, single path mode
0
1
2
3
A
id1=(0,1) for VM1
id2=(2,3) for VM2
IOC:I/O Control unit
A: Shared device by VM1 and VM2 with separated channel-paths
(Conventionally, DIO & CPR simultaneously supported to this allocation form)

Fig. 3 Dedicated channel-paths and shared device.

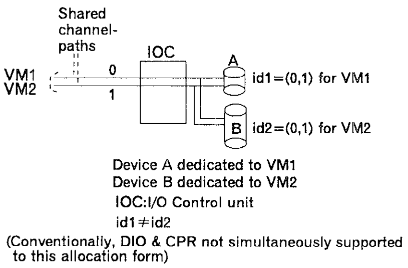

Shared channel-paths
IOC
VM1
VM2
0
1
A
B
id1=(0,1) for VM1
id2=(0,1) for VM2
Device A dedicated to VM1
Device B dedicated to VM2
IOC:I/O Control unit
id1≠id2
(Conventionally, DIO & CPR not simultaneously supported to this allocation form)

Fig. 4 Shared channel-paths & dedicated devices.

(B), which is dedicated to VM 2 and connected to the set of channel-paths. In this case the DIO and the CPR of the VM are supported simultaneously to the device because OS 1 in the VM 1 and OS 2 in the VM 2 can compose their own channel-path-groups with their own identifiers and channel-path modes. This is the conventional technique.

Figure 3 shows a device (A) shared by VM1 and VM2, and two sets of channel-paths (0, 1) and (2, 3), dedicated to VM1 and VM2, respectively. In this case the DIO and the CPR of the two VMs are supported to the device simultaneously because OSs in the VMs can compose their own channel-path-groups with their own identifiers and channel-path modes. This is also the conventional technique.

(2) Shared channel-paths

Figure 4 shows two devices A and B dedicated to VM1 and VM2, respectively, and channel-paths (0, 1) shared by VM1 and VM2. In this

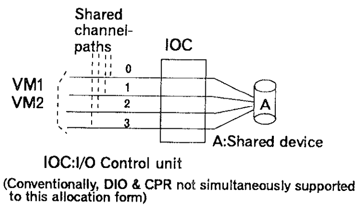

Shared channel-paths
IOC
0
VM1
1
VM2
2
3
A
A:Shared device
IOC:I/O Control unit
(Conventionally, DIO & CPR not simultaneously supported to this allocation form)

Fig. 5 Shared channel-paths & shared device.

case, OSs in the VMs will compose the same channel-path-group (0, 1) in their devices, and they will specify different identifiers to the channel-path-group (0, 1). Channel-path-groups are recognized by their identifiers. The different identifiers represent different channel-path-groups even when they consist of the same set of channel-paths. Therefore, the different identifiers cannot be defined because of the uniqueness requirement of channel-paths. Consequently, the OSs cannot compose their own channel-path-group, and cannot use the CPR to their devices.

Figure 5 shows channel-paths (0, 1, 2, 3) shared by VM1 and VM2, and a device (A), which is connected to the channel-paths also shared by them. OSs in the VMs cannot compose their own channel-path-groups containing the shared channel-paths as stated above. Therefore, they cannot simultaneously use the DIO and the CPR to the device.

The above investigation states whether or not the DIO and the CPR are simultaneously supported to a VM depends on the forms of channel-path allocations to the VM. Conven tionally, the allocation forms shown in Fig. 2 and 3 are used for the simultaneous support for the DIO and the CPR. In the following section present we new methods for concurrently supporting the DIO and the CPR even to the allocation forms shown in the Fig. 4 and Fig. 5.

3. New Methods for Consistency of CPR with DIO in VMs

We propose new methods here to make the Channel-Path-Reconnection (CPR) be consistent with the Direct I/O-Execution (DIO) whether channel-paths are shared by VMs or not. The methods consist of handshakings and hardware extensions.

3.1 Handshaking

It depends on the forms of channel-path allocations whether the DIO and the CPR can be simultaneously supported to a VM. An OS in the VM cannot know the total channel-path allocation forms, and a channel-path-group is recognized by its identifier. An OS specifies an identifier when it composes a channel-path-group. All the channel-path-groups that have the same identifier are merged into one larger channel-path-group. In the following handshaking, the hypervisor (i.e. the VMM) informs a guest OS of channel-path-group identifiers, which the guest OS should use to compose its channel-path-groups, based on the allocation forms of channel-paths. Two handshaking methods are proposed as follows.

(1) Informing guest OS of channel-path status

Channel-path-groups are recognizes by their identifiers by OSs and hardware (i.e., channel subsystems and IOCs). When guest OSs use different identifiers for their channel-path-groups that contain a shared channel-path, the shared channel-path is recognized to belong to different channel-path-groups. This does not satisfy the uniqueness requirement of channel-paths. Therefore, guest OSs cannot use their own identifiers of channel-path-groups that contain a shared channel-path. In the following handshaking, the VMM determines identifiers of channel-path-groups for guest OSs.

This handshaking is as follows: A guest OS should issue hypervisor call (i.e. the VMM call) to get channel-path status after it has been IPLed. The VMM, responsive to the call, determines whether or not channel-paths (0, 1, 2, ⋯, 255) are dedicated to the calling VM, and to them determines appropriate channel-path-group identifiers, which satisfy the uniqueness requirement of the channel-paths, and gives the identifiers to the calling guest OS. The guest OS has to use them to compose their channel-path-groups because the guest OS has to satisfy the uniqueness requirement of the channel-paths. The channel-path-group identifiers are determined by the VMM as follows.

(1) When the channel-path (i) is dedicated to the calling VM:

This case applies to Fig. 2 and Fig. 3, that is, channel-path (i) is one of 0, 1, 2, 3, 4 in the figures. In VM directories, users can specify dedicated channel-paths, and they can also specify whether guest OSs compose channel-path-groups or not. Figures 2 and 3 show two different identifiers (id1, id2) of two channel-path-groups (group 1 and group 2), which are composed of dedicated channel-paths ((0, 1) and (2, 3)), respectively.

I/O control units (IOCs) recognize channel-path-groups with their identifiers in devices, and reconnect the devices with channel-paths at various timing, e.g. at I/O interrupts, within the channel-path-groups recognized. In Fig. 2, IOCs recognize channel-path-group 1 in device (A) and group 2 in device (B). Therefore, their identifiers may be the same. In Fig. 3, the IOCs must recognize two different channel-path-groups in the same device (A). Therefore, their identifiers must be different.

Accordingly, the identifier of the channel-path-group containing the channel-path (i) has to be different from identifiers of another channel-path-group defined by another VM. Therefore, the VMM determines the identifier of the channel-path-group containing the channel-path (i) as follows.

Identifier of a channel-path-group that contains the channel-path (i) = *VMid.Sysid*

Here, Sysid: Identifier of system where VMS runs.

VMid: the identifier of the calling VM to which the path (i) is dedicated.

This *Sysid* is a system identifier, which can be gotten by Store CPU Identifier instruction. It is necessary because different systems, each of which may be run by a VMM or by only one OS, may share a device connected to the channel-path (i). The channel-path-group identifiers of one system have to be different from those of another system. Though they are different each other in one system, VM identifiers may be the same in different systems. Therefore, the Sysid is appended to the VMid. The channel-path mode of the channel-path (i) is determined by the guest OS to which the path (i) is dedicated as shown in Figs. 2 and 3.

(2) When the channel-path (i) is shared by the calling VM and another VM that composes channel-path-groups containing the channel-path (i):

This case applies to Figs. 4 and 5, that is, the channel-path (i) is 0 or 1 in the two figures or 2 or 3 in Fig. 5. In the figures VM1 and VM2 share the channel-paths. Therefore, they must use the same identifier and the same path mode to the shared channel-paths when they compose channel-path-groups containing the shared channel-paths. When the channel-path (i) is shared by VMs (VM1, VM2) that compose a channel-path-group containing the path (i), they must use the same identifier and the same path mode to the channel-path-group. This is due to the uniqueness requirement of the shared channel-path (i).

We define VM-groups as follows. That is, one VM-group is made of any VMs that share a channel-path and compose a channel-path-group containing the shared channel-path. Moreover, we merge two VM-groups that contain the same VM into one larger VM-group. We also merge any two VM-groups that share a channel-path into one larger VM-group. Therefore, any two VM-groups share neither VMs nor channel-paths. We define an identifier of channel-path-groups for one VM-group. A VM-group has only one identifier of a channel-path-group for a device connected to a channel-path of the channel-path-group. That is, any two guest OSs in a VM-group must compose the same channel-path-group to one device, and cannot define two different channel-path-groups to one device.

Figure 6 shows an example of this prohibition. Two channel-path-groups id1 = (0, 1) and id2 = (1, 2, 3) (id1 ≠ id2) cannot be defined for VM-group (VM1, VM2). This is because the VM-group can have only one identifier for its channel-path-groups, therefore, two groups of channel-paths cannot be defined in one device. The VM-group can define different channel-path-groups to different devices as shown in **Fig. 7** because they are subsets of one larger channel-path-group recognized by one identifier. The identifiers of different VM-groups have to be different each other because two VMs of

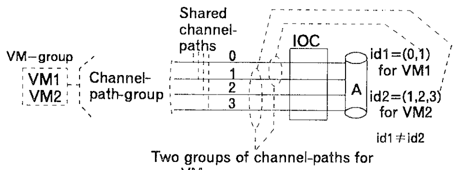


Fig. 6 Two channel-path-groups for one VM-group cannot be defined. (Conventionally and even in handshaking)

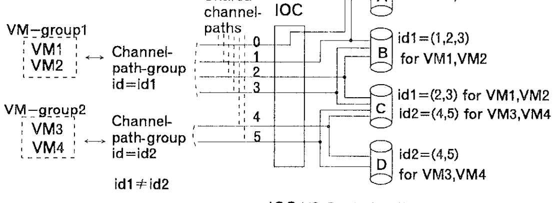


Fig. 7 VM-groups and shared channel-paths. (Proposed handshaking)

different VM-groups can share devices with separated channel-paths as shown in Fig. 7.

Figure 7 shows two VM-groups (VM-group1, VM-group2). One has identifier (id1), four shared channel-paths (0, 1, 2, 3). Moreover, it has three channel-path-groups, which have the same identifier (id1), that is, (0, 1) in device (A), (1, 2, 3) in device (B), and (2, 3) in device (C). The VM-group2 has shared channel-paths (4, 5), and channel-path-group (4, 5), which has identifier (id2), in device (C) and (D).

Accordingly, the VMM determines a channel-path-group identifier to a VM-group as follows.

Identifier = VMgroupid.Sysid

Here, VMgroupid : A group identifier of a VM-group

In handshaking the VMM informs a guest OS of this identifier defined for a VM-group when the channel-path (i) is shared by VMs belonging to the VM-group. Sysid is system identifier. This is necessary to discriminate VMgroupids of one system from those of another system. The VMgroupid is necessary for discriminating the channel-path-group identifier and the path mode of one VM-group from those of another VM-group. The channel-path-group of type 2 VMs, whose OSs are type 2 only, has to be exclusive with that of type 1 VMs, whose OSs are type 1 only, because the two channel-path-groups have different path modes. Therefore, type 1 VM-group and type 2 VM-group have to be exclusive with each other. System generation has to define the VM-groups based on I/O system configurations.

(2) Informing guest OS of channel-path-group identifier to subchannel

The second handshaking is that the VMM informs an OS in a VM of channel-path-group identifiers to subchannels dedicated to the VM. In general, an OS issues an instruction to read all its subchannel status information when being IPLed. The instruction is simulated by the VMM. This handshaking requires the VMM, responsive to the instruction, to give each dedicated subchannel an identifier of a channel-path-group to be composed of the channel-paths of the subchannel. The VMM determines the channel-path-group identifier in the same manner as in the first handshaking.

(3) Summary of handshaking

Guest OSs have to use the channel-path-group identifiers given by the VMM. The identifiers given satisfy the requirements of the uniqueness of the channel-path-groups and channel-path modes. Therefore, using the identifiers given by the VMM, the OSs can directly, that is, with the direct I/O-execution, compose their own channel-path-groups, and use the channel-path-reconnection. That is, one of the above-stated two handshakings makes the CPR be consistent with the DIO.

3.2 Hardware Extensions

We propose three means to extend hardware functions[13),14)]:

(1) Extension for channel-path modes

This is to allow each channel-path of channel-path-groups to have different path modes in different devices. This extension eases the uniqueness requirement of channel-path modes. That is, the uniqueness of channel-path modes is required only in a device, and not in a total system. **Figure** 8 shows an example, that is, a device (A) dedicated to VM1 composes the channel-path-group (0, 1) with the single path mode, on the other hand, a device (B) dedicated to VM2 composes the same path group with the multiple path mode. This composition is allowed by this extension because it satisfies the uniqueness requirement in devices. This extension requires additional memory and control logic in each I/O control unit. For example, conventionally, an I/O control unit controls maximum 32 devices and has maximum 8 channel switches. Then, each channel-path needs 1 bit to each device for indicating the multiple path mode or the single path mode. Therefore, this extension needs the following additional memory per I/O control unit.

$$1 \cdot (32\ \textit{devices}) \cdot (8\ \textit{channel-paths}) = 256\ \textit{bits}$$

Moreover, OSs in VMs have to use the above-stated handshakings for channel-path-group identifiers, because the OSs have to satisfy the uniqueness requirement of channel-paths. The handshakings become easier with this extension, because each channel-path is allowed to have different path modes in different devices

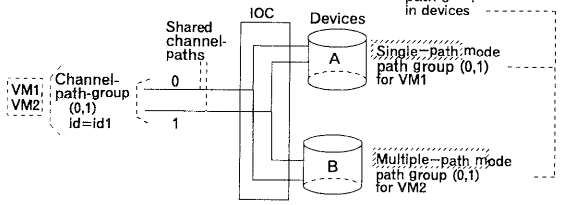

Channel-path-groups in devices
IOC
Devices
Shared channel-paths
VM1
VM2
Channel-path-group (0,1) id=id1
0
1
A
B
Single-path mode path group (0,1) for VM1
Multiple-path mode path group (0,1) for VM2
IOC:I/O Control unit

Fig. 8 Different path modes to the same channel-paths in different devices. (Hardware extension required)

as shown in Fig. 8. This extension allows that type 1 VM and type 2 VM share a channel-path, therefore, they may belong to the same VM-group. If they share a device, their I/O requests to the device are simulated by the VMM, because even in this extension, a device cannot have two different path modes at the same time.

(2) Suppression of channel-path-reconnection to type 1 OS

This is to suppress the function of the channel-path-reconnection to devices of the type 1 VMs. That is, the VMM makes the type 1 OSs unable to recognize the function to any devices, even if the devices have the function. This suppression is considered reasonable as follows. First, the type 1 OSs do not support the CPR. Second, there are few type 1 OSs that compose channel-path-groups. Third, the type 2 architecture will replace the type 1 architecture in the near future.

The concrete methods are as follows. An OS examines whether devices have the function by issuing an I/O instruction to them. The I/O instruction specifies a CCW such as,

Sense Identification,

which reads the device features. When this examination shows that the devices do not have the function, the OS does not compose any channel-path-groups to the devices. The VMM suppresses the function to the devices used by the type 1 VMs before IPLing the OSs in the VMs. This suppression makes the type 1 OSs not compose any channel-path-groups. This method does not require the type 1 OSs to be modified.

We have to provide some new commands to activate/inactivate the function to the devices. The commands have to be effective to each combination of a device and its channel-path, because type 1 VM and type 2 VM may share the device in the direct I/O-execution by using separately allocated channel-paths as shown in Fig. 3. This suppression allows only the type 2 OSs to compose the channel-path-groups. Therefore, the path mode is always the multiple path mode.

In this method, type 2 guest OSs have to use the above-stated handshakings to satisfy the uniqueness requirement of channel-paths. The handshaking with this extension is easier, because only the multiple path mode is used in the system. This extension also allows that type 1 VM and type 2 VM share a channel-path, therefore, they may belong to the same VM-group. This is because the type 1 OS does not compose any channel-path-group.

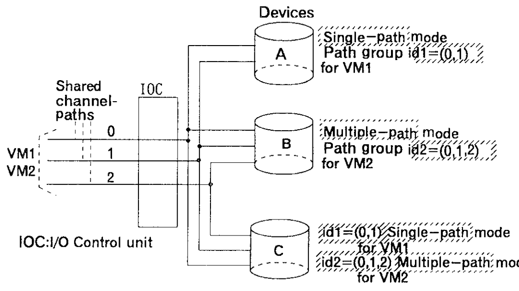


Fig. 9 Different channel-path-group identifiers and different path modes to the same channel-paths and to the same device. (Hardware extension required)

(3) Extension for channel-path-groups and channel-path modes

This extension is to allow channel-paths to have the different path group identifiers and the different path modes in devices. This extension eases the uniqueness requirement of channel-paths and that of channel-path modes. That is, they are required only in a VM, and not in a total system. **Figure 9** shows an example, that is, a device (A) dedicated to VM1 has the channel-path-group (0, 1) with the identifier (id1) and the single path mode. Moreover, a device (B) dedicated to VM2 has the same channel-path-group with the identifier (id2) different from (id1) and the multiple path mode. Moreover, device (C) has two channel-path-groups (id1=(0, 1) for VM1 with single path mode, and id2=(0, 1, 2) for VM 2 with multiple path mode).

To support this extension an IOC has to have a channel-path-group management table extended for VMs[13),14)]. **Figure 10** shows the extended table containing the information for channel-path-group management shown in Fig. 9. The information added by the extension is VMID field and the lines added at second or later occurrences of the same channel-paths, which are specified by different VMs. We suppose that a real system has different channel-path-group identifiers from those of another real system, though the two real systems may have the same VM identifier. Moreover, we suppose that in one real system VM identifiers are different, though channel-path-group identifiers may be same. These suppositions are reasonable because channel-path-group

Channel path	Channel-path-group identifier	VM identifier	Channel-path mode	Device & its state
0	id1	VM1	S	(A,0),(C,0)
1	id1	VM1	S	(A,0),(C,0)
0	id1 or id2	VM2	M	(B.0),(C,0)
1	id1 or id2	VM2	M	(B,0),(C,0)
2	id1 or id2	VM2	M	(B,0),(C,0)
4	id3	VM2 or VM3	M	(D,0)
5	id3	VM2 or VM3	M	(D,0)

▨ : Added field for VMs

S: Single path mode
M: Multiple path mode

id1,id2,and id3 are different channel path group identifiers
(A,0): defined in device A and its status is free
(A,1): defined in device A and its status is busy

Fig. 10 Channel-path-group management table extended for VMs.

identifiers will contain system identifiers and two different VMs in one real system have to have different VM identifiers.

An IOC has to have modified logic, which makes and updates the extended table shown in Fig. 10. The IOC logic has to accept the request of VM (e.g.VM2) for making a channel-path-group, whose identifier is id2, that contains a channel-path shared by VMs (e.g. VM1 and VM2), even when the shared channel-path belongs to another channel-path-group, whose identifier is id1, which another VM (e.g.VM1) has already defined. That is, the IOC logic has to accept the request even when it specifies different channel-path-group identifiers and different channel-path modes to the shared channel-path. In addition, the IOC logic has to determine a channel-path-group in a device by recognizing both channel-path-group identifiers and VM identifiers when making the CPR effective. Figure 9 shows an example, that is, the device (C) has a channel-path-group (0, 1) for VM1 and another channel-path-group (0, 1, 2) for VM2.

This extension completely solves the problems of channel-path-groups, because it allows guest OSs to make their own channel-path-groups with their own identifiers and path modes in any devices shared or dedicated. Therefore, this is the best among the two handshakings and the extension of hardware specifications for the benefit of VMs.

This extension requires a large control table in the I/O control units. Conventionally, I/O control units have 12-byte channel-path-group identifiers containing path mode bits[10], and controls 32 devices and 8 channel switches, and VM identifiers are 8 characters long. Then, this method requires the following additional memory for the extended management table per I/O control unit.

$$(12+8)\cdot(32\,devices)\cdot(8\,channels)\cdot N$$
$$=5{,}120\cdot N\ bytes$$

N: the number of VMs

In addition, the IOC has to have new logic for controlling the channel-path-group identifiers and the path modes to channel-paths, devices, and VMs. This will require relatively large hardware/microprogram modification. In this meaning, this extension is hard to implement.

(4) Summary of hardware extension

Three hardware extensions are presented. The first two of the three ease the restrictions in channel-path modes, and require the guest OSs to use one of the two handshakings. The other, by itself, provides complete means for solving the problems about the consistency of the channel-path-reconnection with the direct I/O-execution.

4. Evaluation

The above-mentioned methods are summarized in **Table 1** and they are evaluated functionally in **Table 2** under the following load model supposed.

(1) At least two type 2 OSs, which compose channel-path-groups, are run. For example, one is for running application programs and the other is for developing systems.

(2) At least two type 1 OSs, which compose channel-path-groups, are run.

Table 2 shows the following evaluation concerning the methods.

(1) If no hardware modifications are allowed, the full handshaking (S) has to be selected, where a loader and an I/O supervisor of a guest OS have to be modified. The loader or its extension has to issue hypervisor-call to get an identifier for a channel-path-group and the I/O supervisor has to use the identifier designated by the VMM to define the channel-path-group. The loader and the I/O supervisor account for less than 4% of the total steps of the guest OS.

(2) If hardware modifications are allowed, the full hardware extension (H) is the best, because it requires no modifications of OSs. If hardware modifications are allowed, and have to be small, the hardware extension (SH1) or (SH2) is desirable.

It is practically impossible to modify all the

Table 1 Methods for consistency of channel-path-reconnection with direct-I/O execution.

Abbreviation	Methods
(S)	Full handshaking for channel-path-groups, and channel-path modes
(H)	Full hardware extension for channel-path-groups and channel-path modes
(SH1)	Hardware extension for mixing channel-path modes, and handshaking for channel-path-groups
(SH2)	Hardware extension for suppressing channel-path-reconnection to devices of type 1 OS, and handshaking for channel-path-groups

Table 2 Functional evaluation of methods for consistency.

Evaluation / Methods*1	OS modification	Hardware modification	Share of channel-paths by type 1OS and type 2 OS	Composing channel-path groups	
				type 1 OS	type 2 OS
(S)	Less than 4% of the total steps of OS (Type 1, 2 OSs)	No	No	OK	OK
(H)	No	5120 * N*2 Bytes/IOC +Control Logic	OK	OK	OK
(SH1)	Less than 4% of the total steps of OS (Type 1, 2OSs)	32 Bytes/IOC +Control Logic	OK	OK	OK
(SH2)	Less than 4% of the total steps of OS (No mod. for type 1 OSs)	A few Commands to Suppress CPR*3	OK	No*4	OK

*1 : Stated in Table 1
*2 : N : Number of VMs
*3 : CPR : Channel-Path-Reconnection
*4 : Because the dynamic channel-path-reconnection is suppressed to devices of the type 1 OS

existing I/O control units. Therefore, OSs have to check all the devices by using the command such as "Sense Identification" which has to inform the OSs of hardware extensions for VMs.

Effects of the channel-path-reconnection are evaluated in the literature[11], which reveals that I/O response is improved by 30% when channel utilization is 40%. On the other hand, effects of the direct I/O-execution are reported in the literatures[3),5)], which show that the direct I/O-execution gives near-native performance to the VMs. The proposed methods provide the consistency of the channel-path-reconnection with the direct I/O-execution whether channel-paths are shared by VMs or not. Therefore, with them, users can obtain the above-mentioned two kinds of effects simultaneously.

5. Conclusion

Conventionally, guest OSs cannot share channel-paths when they use both the channel-path-reconnection (CPR) of the channel subsystem and the direct I/O-execution (DIO) simultaneously. This restriction comes from uniqueness requirement of a channel-path : A channel-path with the CPR has to belong to only one channel-path-group. This requirement is reasonable in a real machine environment. Therefore, a set of channel-paths has to be dedicated to a VM when it uses both the CPR and the DIO at the same time. The CPR is the key to improve I/O response[11]. On the other hand, the DIO is the key to VM CPU performance[3),5)]. This channel dedication is not preferable, however, because it considerably decreases the flexibility of system configurations.

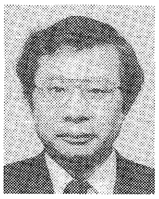
We have presented new methods to give the consistency of the CPR with the DIO whether channel-paths are shared by VMs or not. The methods consist of the following. First, handshaking : The VMM informs a VM, responsive to the hypervisor calls of an OS in the VM, of the identifiers of channel-path-groups, based on the allocation forms of the channel-paths in the total system. The VMM determines the identifiers that satisfy the uniqueness requirement of the channel-paths. Second, hardware extensions: One of them eases the uniqueness requirement of channel-paths, that is, it is required only in VMs, and not in the total system. The extension allows OSs in VMs to define their own channel-path-groups, which may contain shared channel-paths, with their own identifiers and their own channel-path modes, which may be different each other, to all combinations of devices, their channel-paths, and VMs. We also have presented other intermediate methods.

With these methods, OSs in VMs, being allowed to share channel-paths, can obtain two kinds of benefits simultaneously, that is, improvement of I/O response with the CPR and that of CPU performance with the DIO.

References

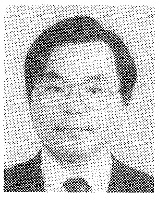
1) Goldberg, R. P.: Architectural Principles for Virtual Computer System, Ph. D. dissertation, Div. Eng. Appl. Phys., Harvard Univ., Cambridge, MA. (1972).
2) Borden, T. L. et al.: Multiple Operating Systems on One Processor Complex, *IBM Syst. J.*, Vol. 28, No. 1, pp. 104-123 (1989).
3) IBM: Virtual Machine/Extended Architecture System Product (VM/XA SP) Release 1E, 2, Programming Announcement (June 1987).
4) Gum, P. H.: System/370 Extended Architecture: Facilities for Virtual Machines, *IBM J. Res. Develop.*, Vol. 27, pp. 530-544 (Nov. 1983).
5) Umeno, H. et al.: New Methods for Realizing Plural Near-Native Performance Virtual Machines, *IEEE Trans. Computers*, Vol. C-36, No. 9, pp. 1076-1087 (Sep. 1987).
6) Umeno, H. et al.: I/O-Execution Method for Virtual Machine System, USP4885681, Ap. Jan. 18, 1984.
7) Bean, G. H. and Borden, T. L.: Logical Resource Partitioning of a Data Processing System, USP4843541, Ap. July 29, 1987.
8) IBM: System/370 Principles of Operation, GA22-7000.
9) IBM: System/370-XA Principles of Operation, SA22-7085.
10) IBM: 3880 Storage Control Unit model 13 description 3rd edition, GA32-0067 (June 1982).
11) Brandwajn, A.: A Study of Dynamic Reconnection, *SIGMETRICS*, Special Issue, pp. 1-11 (1983).
12) HITACHI: VMS/AS General Information Guide.
13) Umeno, H. et al.: Channel-path-group Management, Patent Application, File: Jan. 20, 1987.
14) Inoue, T., Umeno, H. et al.: I/O Control Unit, Patent Application, File: Nov. 29, 1988.

(Received February 16, 1994)
(Accepted January 12, 1995)



Hidenori Umeno was born in Ohita, in 1947. He received the B.S. in mathematics from Kyushu University in 1970. From 1970 to 1976, he was with Central Research Laboratory, Hitachi Ltd., where he made researches in productivity improvement of compilers. From 1976 to 1993, he was with Systems Development Laboratory, Hitachi Ltd., where he made researches in performance and reliability improvement of virtual machines, file systems, and operating systems. Since 1993, he has been with General Purpose Computer Division, Hitachi Ltd., where he has been engaged in the development of logical partition systems of mainframes. His main research fields are performance and function improvement of virtual macchines, logical partition systems, operating systems, and computer architectures. He received a best paper award of Information Processing Society of Japan (IPSJ) in 1982. Since 1991, he has been a part-time instructor of Musasi Institute of Technology. He is a member of IPSJ and ACM, and an affiliate of IEEE Computer Society.

Taro Inoue was born in 1961. He received B.E. in 1984 and M.E. in 1986 on the Production Machinery Engineering from Kyoto Institute of Technology. Since 1986, he has been with Systems Development Laboratory, Hitachi, Ltd. He has engaged in research on virtual machine system, management of storage system, and management of distributed system. He is currently a Researcher of the Systems Development Laboratory, Hitachi, Ltd. He is a member of the Institute of Systems, Control and Information Engineering and the Information Processing Society of Japan.



Shunji Tanaka received the B.S. and M.S. degrees in mathematics from Kyushu University in 1979 and 1982, respectively. He is a researcher in Systems Development Laboratory, Hitachi, Ltd. Since 1983, he has been engaged in research on performance improvements for virtual machine systems and reliability improvements for operating systems. His current research includes parallel computer architectures.