# IBM POWER6 partition mobility: Moving virtual servers seamlessly between physical systems

W. J. Armstrong
R. L. Arndt
T. R. Marchini
N. Nayar
W. M. Sauer

*This paper presents the IBM Power Architecture™ extension for enhanced virtualization that is first implemented in the POWER6™ processor. Virtual partition memory enables all of the memory of a virtual server running in a logical partition to be made virtual by the POWER Hypervisor™ firmware. The Processor Compatibility register allows backward compatibility by providing a mode in which a POWER6 processor behaves like a POWER5™ processor. Enhancements to the timebase facility enable updates to the virtual timebase of a logical partition while maintaining consistency with other partitions in the system. These fundamental enhancements are explained and their role in implementing the new partition mobility function is described. Partition mobility allows the seamless migration of virtual servers from one physical POWER6 microprocessor-based system to another.*

## Introduction

Partition mobility makes it possible to move running partitions from one physical server to another. This provides considerable systems management flexibility and improved availability. Applications no longer have to be shut down to move them from one server to another. This is useful in several ways:

- Planned outages for hardware and firmware maintenance and upgrades can be avoided by moving the running partitions to an adequately configured alternate server during the maintenance or upgrade.
- Workloads running on servers that are indicating a predictive failure can be moved to other servers so repairs can be made. This avoids a scheduled outage and a potential unscheduled one.
- Workloads on several small, underutilized servers can be consolidated onto a single large server without an application outage.
- Workloads can be moved from server to server in order to optimize server utilization and workload performance within a data center—all while the applications are running.

These are just some examples of capabilities enabled by partition mobility. IBM POWER6* processor partition mobility distinguishes itself from competitive products by supporting mobility between heterogeneous POWER6 processor-based systems. In part, this is accomplished by allowing the operating system (OS) and firmware to cooperate with partition mobility. Additionally, the ability to make partitions more virtualized, and therefore more mobile, is enabled by several architectural enhancements in the POWER6 processor. Among these enhancements are virtual partition memory (VPM), the Processor Compatibility register (PCR), and new POWER6 processor timebase adjustment facilities. VPM allows the IBM POWER Hypervisor* firmware to relocate the real memory of a partition underneath an active OS by optionally adding a level of indirection in the translation process and intercepting page fault exceptions. Running under VPM, the partition memory state can be copied from the source to the destination platform, and the hypervisor monitors any changed pages on the source platform. The partition can then be momentarily stopped while the processor state is transferred and processing restarts on the destination platform. After processing starts on the destination platform, any changed pages are copied over, with priority given to changed pages that are required to



**0018-8646/07/$5.00 © 2007 IBM**

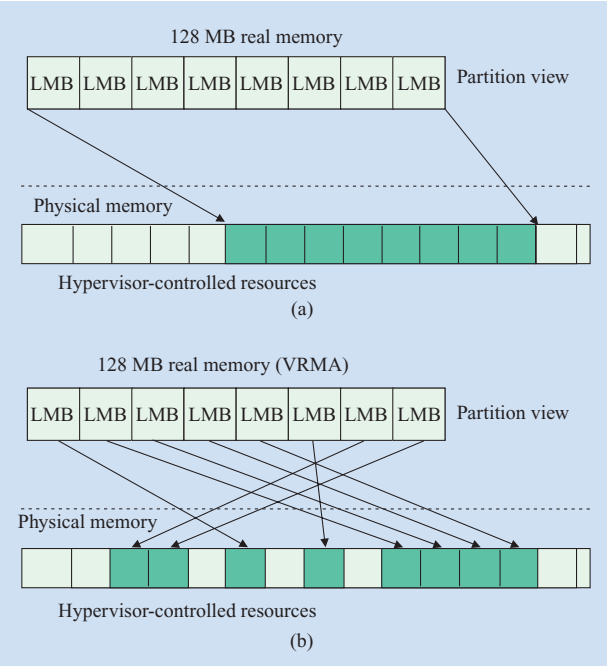


**Figure 1**

RMA memory allocation (a) without VPM and (b) with VPM (VRMA).

satisfy processing accesses. This forgoing sequence is standard in the industry; however, unlike other solutions, POWER6 firmware notifies the OS just prior to halting processing and after restarting in order to allow it to gracefully prepare and notify location-sensitive programs of the change in physical location [1]. The PCR facility allows IBM POWER* processors to operate in a backward-compatible mode, increasing the potential targets for partition mobility. The POWER6 processor timebase adjustment facilities ensure that the OS view of the processor time never goes backward during a move from one platform to another.

## Virtual partition memory

### *Logical real memory*

Power virtualization provides for complete isolation of the logical partitions running in the system. This includes the partition memory, because it is essential that no partition can read or write the memory of another partition, so the POWER Hypervisor firmware needs complete control over all of the partition-accessible memory. Thus, the POWER4* and POWER5* processors provide the facilities to enable the hypervisor to control the real system partition-accessible memory [2, 3]. These facilities have been designed to not only maintain hypervisor control, but also minimize the performance impact on the program running in the partition.

A logical partition sees its memory as logically contiguous address space of 0 through $N$. The low portion of the logical address space, called the *real mode area* (RMA), is addressable by the partition when address translation is disabled in the machine state register. The remaining memory must be addressed through the virtual address translation mechanism provided for in the IBM Power Architecture* technology.

The hypervisor allocates memory to the partitions in units called *logical memory blocks* (LMBs). Each LMB is a physically contiguous area of memory, and all LMBs in a system have the same size (e.g., the LMB size in a POWER6 processor-based system is 16 MB). LMB size can be changed with a system reboot. Prior to the POWER6 processor, the LMBs assigned to the RMA had to be physically contiguous. The RMA for the partition is defined by a real start address (defined in the real-mode offset register [RMOR]) and a length (defined in the RMLS field of the Logical Partition Control register [LPCR]). The hypervisor has to supply a sufficiently large contiguous block of real storage to satisfy the OS real memory requirements (e.g., 128 MB), as shown in **Figure 1(a)**. The non-RMA of the partition logical real address space can be populated by any set of LMBs in the system (they do not have to be contiguous, as do the RMA LMBs).

Although the existing mechanisms provided the necessary control and optimized performance, they lacked the capability for the hypervisor to make the memory of a partition virtual at a fine granularity. One of the reasons that fine-grain control is required is to track changed pages for partition mobility. For example, there is no mechanism for the hypervisor to track stores to RMA pages when address translation is disabled. VPM provides the capability without compromising the partitioning integrity of the system and without an additional layer of translation in the processor.

### *VPM architecture*

VPM allows the hypervisor to track changes made to memory pages by a partition that is being moved from one physical server to another. Because the partition continues to run while it is being moved, it may make changes to memory pages after the hypervisor has sent the earlier copy of memory to the destination server. By tracking changed pages during partition movement, the hypervisor knows which partition pages must be re-sent to the destination. The idea is to treat partition memory like virtual memory at the hypervisor level. This could have been done by requiring a second level of address translation for all partition memory accesses, an

approach that has been applied successfully on IBM zSeries* servers for decades. IBM pSeries* virtualization previously kept a single level of address translation by maintaining the partition page table in the hypervisor, which reduces complexity and improves performance. VPM was required to keep address translation to a single level but still provide the control to the hypervisor for virtualization functions, such as partition mobility.

The VPM architecture adds support for translating RMA addresses through the page table of the partition. The following architecture facilities had to be defined in order to make this approach work:

- Three bits in the LPCR control the VPM mode.
  - One bit controls VPM mode for memory accesses with partition address translation disabled ($VPM_0$).
  - A second bit controls VPM mode for memory accesses with partition address translation enabled ($VPM_1$).
  - The ISL mode bit forces all virtual accesses to be treated by hardware as if they were accesses to 4-KB pages. (*ISL* refers to the ignore SLB large-page specification, and *SLB* refers to segment look-aside buffer.)
- Define and reserve a special virtual segment ID (VSID) to be used for partition real address translation. The VSID is normally obtained by searching the SLB, when address translation is enabled. There are no SLB entries for partition real address translation.
- The VRMASD field in the LPCR defines the page size to be used for the translation of partition real mode addresses. This information is also provided by the SLB entry for virtual memory accesses.
- New interrupts are defined to enter the hypervisor page fault handler if a partition real memory access misses in the page table.
- Along with the interrupts, new special-purpose registers were defined to assist the hypervisor in handling the interrupts.

The above mechanisms allow the hypervisor to allocate noncontiguous LMBs for the partition RMA. With VPM support, memory is allocated for the RMA in the same manner as it is for the non-RMA. The term *VRMA* (virtual real mode area) is used to describe the RMA where the address translation for storage accesses occurs through the page table. **Figure 1(b)** shows the allocation of LMBs for a partition with a VRMA.

Since both the VRMA accesses and the non-VRMA accesses are translated through the partition page table, the VPM architecture also allows the hypervisor to take full control of the mapping of logical memory to physical memory for a partition. On previous systems, the page table for a partition was a hypervisor resource, but interrupts alerting the control program to a missing entry in the page table (page fault) were always directed to the OS running in the partition. The OS would then call a hypervisor service to install an entry in the page table. Thus, the hypervisor decided which physical memory page was mapped to the virtual page and guaranteed partition isolation. In a VPM environment, the page table is still owned by the hypervisor. However, if $VPM_1$ is set, all page faults trigger new hypervisor interrupts instead of interrupts for the OS. The hypervisor can react in different ways depending on the status of the page causing the exception:

- If the page is not installed in the page table by the OS running in the partition, it will forward the interrupt to the OS. This action is transparent to the OS, which sees a normal page fault and engages its virtual memory manager to resolve the situation.
- If the page is marked as read only by the hypervisor for the purposes of tracking stores to partition memory, the hypervisor will record that a store is about to occur, change the protection on the page to allow a store, and return control to the interrupted instruction. Since address translation for both RMA and non-RMA accesses occurs through the page table, stores to all of the partition memory can be tracked by the hypervisor.

## Processor Compatibility register

The PCR controls which level of architecture is visible and can be exploited by an application program (running in problem state). Because this is a new facility, there are currently only two levels of architecture from which to choose: POWER6 and POWER5. With the POWER6 architecture, all new architecture facilities and instructions are available. With POWER5 (architecture version 2.04 [4]), the following new instructions and facilities are not available: the vector facility and all of the instructions associated with it, the decimal floating-point facility and all instructions associated with it, `cmpb`, `fcpsgn`, `lfdp`, `lfdpx`, `stfdp`, `stfdpx`, `lfiwax`, `prtyd`, `prtyw`, the W field in the `mtfsfi` instruction, and the L and W fields in the `mtfsf` instruction.

POWER processor-based servers are generally backward compatible, which means programs running on a previous-generation server will still run on the next generation. Performance characteristics may change between designs to some degree, but the semantics of an existing program do not. The next generation of servers does not take away existing instructions or facilities or

modify the semantics of instructions and facilities in an incompatible way. A new generation of servers typically adds to the instruction set and architecture facilities to improve performance or enable new function; these new instructions and facilities can then be exploited by recompiling programs or by upgrading to newer versions of programs that support the new functions.

The PCR provides a means of forward compatibility: The POWER5 processor-based server is compatible with the POWER6 processor-based server running in POWER5 processor mode. When performing software testing, it is important to have an environment in a POWER5 processor partition on a POWER6 processor-based server that behaves exactly like a POWER5 processor-based server for application programs (more precisely, in a problem state).

The PCR mechanism enables backward migration. For example, a future processor could implement a POWER6 processor mode using the PCR mechanism. In the POWER6 processor mode, a future processor would disable all of the newer facilities. Thus, a partition could migrate back to a POWER6 processor-based server without incurring errors that might result because the new facilities are not available on a POWER6 processor-based server.

## Timebase

The timebase is the main source of time information on a POWER processor. At the time of platform initialization, the timebase of each platform processor is set to the same value, which then increments synchronously for all processors so that an OS detects time advancing identically on each processor. The OS then adds a fixed offset, which is computed when it does a time-set operation, to the timebase value to compute the wall clock time. To prevent requiring a time-set operation when migrating a partition from one system to another, the timebase for that partition must be set by the hypervisor on the destination machine to match the value on the source machine, even though the timebase value on the destination system will not have any correlation to the timebase value on the source system. The partition must not observe time to go backward in this case but may see it move forward only by the amount of time that elapsed during the migration. The rate of change of the timebase must be the same on the source and destination systems. If either of these two conditions is not satisfied, the timing facilities of the partition will encounter errors on the destination system. It is a requirement of the Power Architecture technology that fine-grained synchronization with other processors in the system must also be maintained. This means low-order bits of the timebase on all of the processors within a system must be close and not set arbitrarily. The following are architectural enhancements that allow the partition to observe a consistent view of its timing facilities after migration:

- Bits 0:59 of the timebase register are incremented at a rate of once per every 31.25 ns. The processor architecture requires that the rate of timebase incrementing on all server processors be 512 MHz.
- TBU40 (timebase update facility) is a new special-purpose register that allows access to the upper 40 bits of the timebase. The hypervisor can set up the upper 40 bits of the timebase for a partition without disturbing fine-grained synchronization based on the low-order 24 bits.

## Partition mobility

This section describes the use of the VPM architecture and the TBU40 facility to implement partition mobility. VPM is used to migrate the main store of a partition. The TBU40 facility is used to present consistent timing facilities to the partition when it resumes execution on the destination system.

### *Page sizes*

Besides the 4-KB page size, the POWER6 processor supports large page sizes of 64 KB and 16 MB and a huge page size of 16 GB. The large and huge page sizes present a challenge for tracking "dirty" pages for migration. ("Dirty" pages are those that must be re-sent to the destination system, as explained later.) Partitions that are assigned 16-GB pages are ineligible for migration. The mechanism used to track stores to large pages is explained in the following section.

### *Main store migration*

**Figure 2** illustrates main store migration. Prior to moving a partition, the source system hypervisor places the virtual processors of the partition in VPM mode. It does this invisibly to the OS and applications. As part of the switch to VPM mode, the hypervisor starts using two page tables. The first is the "normal" hashed page table that is visible to the partition. This table is called the *virtual page table* (VPT). The hypervisor makes the second hashed page table, the *physical page table* (PPT), visible to the processor. The hypervisor uses the PPT to track stores to the partition memory based on a 4-KB page size.

The partition may use large pages. The hypervisor keeps track of stores to large pages on a 4-KB page size basis by breaking the large page down into constituent 4-KB pages and installing page table entries for only 4-KB page sizes in the PPT. Since the processor searches the PPT based on the page size in the SLB entry (SLBE),

a mechanism is required to ignore the large-page indicator in the SLBE. This mechanism is the ISL bit in the LPCR, described in the list of architecture facilities in the section "VPM architecture," earlier in this paper. As part of the switch to VPM mode, the hypervisor also switches the processor into a mode in which it ignores the large-page indicator in an SLBE.

The hypervisor keeps track of the pages that need to be migrated in a dirty page table. All pages of the partition are marked as dirty at the start of the migration. Pages that have been sent are set to an effective state of read only in the PPT and marked clean. Whenever the partition attempts to write to one of the clean pages, it is intercepted by the hypervisor by means of a VPM interrupt. The hypervisor reverts that page to the dirty state. The hypervisor then makes the page writable again and returns control to the partition at the point of interruption.

The process of sending or resending pages to the destination hypervisor continues until there is sufficient partition memory state on the destination hypervisor so that the processing of the partition can be transferred to processors on the destination server and resume its operation there. The source hypervisor suspends the partition and transfers its internal processor and other necessary state to the destination hypervisor. The source hypervisor also sends the dirty page table to the destination hypervisor. The destination hypervisor receives the dirty page table and uses it to set the state of all dirty pages to an "invalid" access state. The partition is then resumed on the destination hypervisor. The source hypervisor continues sending the remaining partition page frames to the destination hypervisor, which marks them as clean upon their successful arrival.

The destination hypervisor resumes the partition with the virtual processors of the partition in VPM mode. After the partition is resumed, any attempt by the partition to access a page whose state is invalid causes a VPM interrupt, which is handled by the hypervisor. The destination hypervisor blocks the virtual processor and then makes a high-priority "demand paging" request to the source hypervisor for that page. The requested page is sent ahead of other pages that are waiting to be transferred to the destination hypervisor. When the requested page arrives, the hypervisor marks the page as "valid" and resumes the virtual processor at the point of interruption. This process continues transparently to the partition until all remaining partition pages have been transferred from the source to the destination. Once all pages are resident on the destination server, the destination hypervisor takes the virtual processors of the partition out of VPM mode.

During the period of time that the partition is in VPM mode for movement, other storage access interrupts may occur. The source or the destination hypervisor uses the VPT to analyze an interrupt and passes control to the OS interrupt handler if the interrupt is not associated with partition movement.

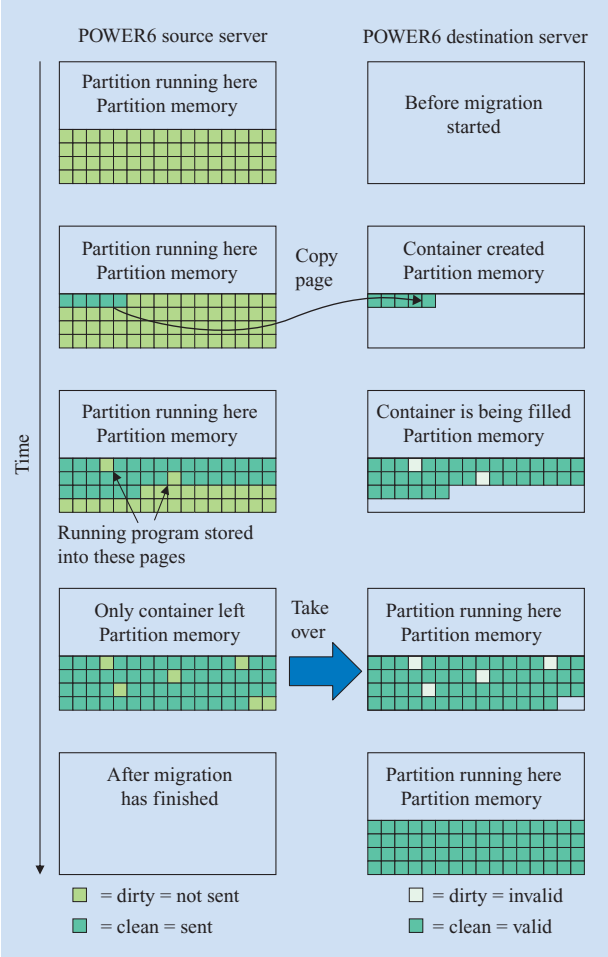


Figure 2

Partition mobility: main store migration.

### *Consistent timing facilities for a migrating partition*

It is advisable to have the time of day synchronized between the source and destination hypervisors through the use of time reference partitions (TRPs). A TRP is tagged as such by the system administrator. The hypervisor synchronizes its time with the Universal Time Coordinated (UTC) of the TRP. A TRP is expected to use the Network Time Protocol (NTP) to maintain its UTC.

When the partition stops executing on the source system, the hypervisor takes a snapshot of the time of day of the hypervisor, the virtual time of day of the migrating partition, and the value of the timebase register on the source system. The values are sent to the destination hypervisor. The destination hypervisor takes a snapshot

of its time of day when the partition is about to resume on the destination system. Because the two hypervisors have synchronized time, the difference between the snapshots of the hypervisor time on the source system and that on the destination system is the elapsed time when the partition was not executing on the source or destination system. The elapsed time is used to compute the virtual time-of-day and timebase value offset for the migrating partition. The timebase offset value is placed in the partition TBU40 register whenever a virtual processor of the partition is dispatched to provide a consistent view of its virtual timebase value.

## Concluding remarks

POWER processor virtualization is enhanced on POWER6 processor-based systems by a set of substantial improvements. Virtualization of memory and timing facilities provides complete partition virtualization when used with virtual I/O and frees the partition from any specific server. The PCR provides additional application compatibility between processor families. This enhanced virtualization, mobility, and portability provide numerous virtual server management advantages and improvements to physical resource utilization for users of POWER6 processor-based systems.

## Acknowledgments

Many people were involved in the conception, definition, and implementation of the POWER6 virtualization improvements. The authors specifically thank the Power Architecture team, Cathy May, Ed Silha, Giles Frazier, and Brad Frey, for their roles in defining the basic mechanisms and transforming them into concise architecture documents.



## References

1. Power.org, "Power Architecture™ Platform Requirements"; see *http://www.power.org/members/developers/specs/PAPR* (registration required).
2. W. J. Armstrong, R. L. Arndt, D. C. Boutcher, R. G. Kovacs, D. Larson, K. A. Lucke, N. Nayar, and R. C. Swanberg, "Advanced Virtualization Capabilities of POWER5 Systems," *IBM J. Res. & Dev.* **49**, No. 4/5, 523–531 (2005).
3. IBM Corporation, "Logical Partition Security in the IBM eServer pSeries 690," white paper; see *http://www.ibm.com/servers/eserver/pseries/hardware/whitepapers/lpar_security.html*.
4. Power.org, "Power ISA™ Version 2.04"; see *http://www.power.org/resources/downloads/PowerISA_203.Public.pdf*.

*Received January 9, 2007; accepted for publication February 6, 2007; Internet publication October 12, 2007*

**William J. Armstrong** *IBM Systems and Technology Group, 3605 Highway 52 N., Rochester, Minnesota 55901 (billarm@us.ibm.com)*. Mr. Armstrong is an IBM Distinguished Engineer. He received a B.S. degree in computer engineering from the University of Notre Dame and an M.S. degree in computer engineering from Iowa State University. He is the Lead Architect for the POWER Hypervisor firmware. He has worked for IBM since 1988 on a variety of projects, focusing on kernel development, systems performance, and logical partitioning and virtualization of POWER platforms for the IBM pSeries and iSeries* systems. Mr. Armstrong holds numerous patents in the areas of partitioning, virtualization, and kernel design.

**Richard L. Arndt** *IBM Systems and Technology Group, 11501 Burnet Road, Austin, Texas 78758 (rlarndt@us.ibm.com)*. Mr. Arndt is a Senior Technical Staff Member and a RISC Platform Architect. He received B.S. and M.S. degrees in electrical engineering from the University of Wisconsin at Madison. Mr. Arndt has worked with several industry groups to develop standards for I/O and system firmware and architecture. He is responsible for defining the architecture for IBM pSeries logical partitioning (LPAR). Mr. Arndt holds numerous patents in the area of LPAR and platform resource virtualization.

**Timothy R. Marchini** *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, NY 12601 (marchini@us.ibm.com)*. Mr. Marchini is a Senior Technical Staff Member in the System Design organization. He received a B.S. degree in electrical engineering from Gannon University in 1978. He subsequently joined IBM at the development lab in Poughkeepsie, New York, and worked on virtualization and enhanced memory functions for the IBM S/370* (later the S/390* and zSeries) processor design, simulation, testing, and product engineering. He joined the System Design organization in 2001. Mr. Marchini has held a number of management and technical positions in server development.

**Naresh Nayar** *IBM Systems and Technology Group, 3605 Highway 52 N., Rochester, Minnesota 55901 (nayar@us.ibm.com)*. Dr. Nayar is a Senior Technical Staff Member. He holds a B.Tech. degree in electrical engineering from the Indian Institute of Technology, New Delhi, India, and M.S. and Ph.D. degrees in computer science from Iowa State University. He joined IBM in 1992 and has worked on many i5/OS* kernel projects with focus on synchronization primitives and task dispatching. His most recent work has been in the area of LPAR for iSeries and pSeries systems. Dr. Nayar holds numerous patents in the area of partitioning and kernel design.

**Wolfram M. Sauer** *IBM Systems and Technology Group, 11400 Burnet Road, Austin Texas 78758 (wsauer@us.ibm.com)*. Mr. Sauer is a Senior Technical Staff Member in the processor development area. He received a diploma degree (Diplom-Informatiker) in computer science from the University of Dortmund, Germany, in 1984. He subsequently joined IBM at the development lab in Boeblingen, Germany, and worked on the S/370 (later S/390 and zSeries) processor design, microcode, and tools. He joined IBM Austin in 2002 to work on the POWER6 processor project.