A History of Virtualization

Bob Rogers

Abstract

Virtualization in computer systems has a deep and interesting history, starting with early systems using symbolic references to access real resources. In this webcast, Bob Rogers will overview the evolution of virtualization to its current level of sophistication.

Before the advent of the S/360 architecture, older processor models were virtualized through emulation of older processor models on newer, faster models that could not otherwise run the software written for the older machine. Virtualization as we know it began with the experimental CP-40 hypervisor which provided full virtualization to run over a dozen virtual machine on a modified S/360 model 40. The CP-67 hypervisor, which followed up on the CP-40 research, was made available to customers to run on the S/360 model 67. It allowed a single real machine to appear to be a multitude of virtual machines. VM/XA took advantage of new architecture on 370-XA processors to greatly increase the efficiency of processor virtualization. Concurrent with the introduction of ESA/370 architecture, PR/SM brought much of this virtualization technology into the processor itself. Most recently, KVM is providing another choice for virtualization of the IBM mainframe.

Why a Talk on Virtualization?

- Why should z/OS people be interested in Virtualization
  - It's quite the buzz now and it's good to understand the underlying concepts.
  - Everyone seems to want to take credit for something IBM invented and brought to maturity about 50 years ago.
  - I think it's very interesting, even if there's nothing you can do about it.
- This is not intended to be a history of zVM.

Two Phases of Virtualization History

- Virtual Machine technology was brought to maturity in a “big bang” with the CP-67 hypervisor on the S/360 Model 67 processor.
- In the following 50 years, more and more of the simulation provided by CP-67 has been replaced with emulation by hardware and microcode/millicode.

Virtual Computers

- In the beginning was emulation
  - Early in the computer era, each new processor model – even from the same manufacturer – implemented a different instruction set architecture (ISA).
  - Rewriting all programs for the new machine would have just about halted the forward march of automation with computers.
  - Simulation of older architectures with software was very slow.
  - Emulation with hardware and microcode provided a much better solution.

The Power of Emulation

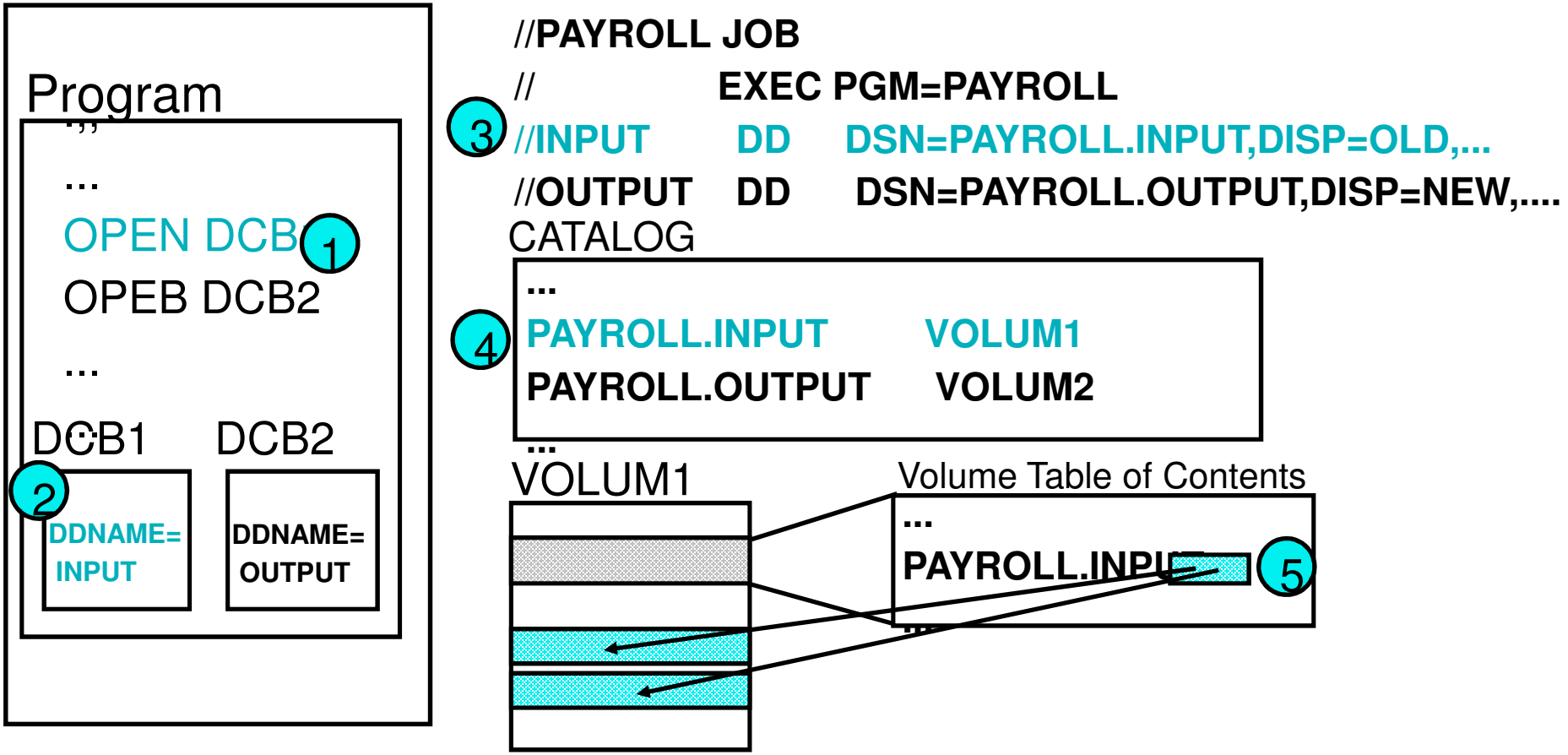
- Many S/360 processors actually emulated the S/360 architecture.
- The machine is implemented as a processor for a microcode language with microcode that then implements an external architecture.
- For S/360 models, this could be the S/360 ISA or, for example, the ISA of the 1401.
- For more on microcode/millicode, see my article *The What and Why of zEnterprise Millicode.*

Accessing Data Abstractly

- Another early from of “virtualization” was the ability to access data symbolically
- Data sets were given names.
- A directory on the disk (aka VTOC) mapped the data set name to actual physical data location on disk.
- Further, job control language could map the data set name to a data definition name so that a program only need use the data definition name within the program.
- These mapping allowed a single program to process multiple sets of data, perhaps even on differing device types.

Abstract Reference to Data

Program access to data:
Several levels of indirection or abstraction



Credit: me from an old presentation

Necessity: Mother of Invention

- In the early days, memory (core memory) was in short supply - insufficient to support a number of timesharing users on a system.
- Virtualizing memory provided a possible solution.
- An early attempt was the Ferranti Atlas system which provided an automatic extension of main memory using drum storage. But this was not the right approach.

The Beginning of the Rest of Time

- IBM needed to get into the timesharing game so they started research on a virtual memory system.
- The hardware was a modified S/360 Model 40, augmented with an address translation box called “CAT BOX”.
- An operating system called CP-40 was developed to run on this hardware.

System/360 Model 44

The CP-40 Experiment

- The modified Model 40 had an array of 16 associative memory registers for address translation.
- The 16 registers mapped one-to-one with the 16 4K frames of real memory on a 256K Model 40.
- If a translation was not available, an exception would be raised to CP-40
- CP-40 would then read the page of data from auxiliary storage and update the array.
- Of course, it first needed to write out the data that currently occupied the frame to be read into.
- A page replacement algorithm is used to determine which page of virtual to expel from real memory out to auxiliary.

Virtues of Virtualization

- It gives the appearance of more memory than is actually installed.
- Eliminates the memory fragmentation problem suffered by all-real systems (like OS/360).
- Allows a single version of a program to run on systems of varying sizes with the same amount of virtual memory regardless of the amount of real memory installed.

Two Really Great Ideas #1

- First, the researchers at IBM decided to separate the management of resources from the interfacing with interactive users.
- CP-40 was developed to manage the system resources like memory, I/O devices and CPU.
- CP-40 supported a number of virtual machines in which an operating system could run independent of other occurrences running in other virtual machines.
- The CMS operating system was developed as a single user operating system to support the interaction with a (just one) user.
- A number of CMS users could be run concurrently, timesharing, on a single machine managed by CP-40.

Two Really Great Ideas #2

- The second idea was that the interface between CMS and CP-40 would be the S/360 architecture.
  - This made the interface clear and well-defined.
  - It also enable concurrent development of CP-40 and CMS
  - As a side-effect, which became very valuable in the future, other operating systems like OS/360 and DOS/360 could also run under the CP-40 “hypervisor”.
  - The virtual memory defined by CP-40 acted as the “real” memory of a guest

Virtual Supervisor State

- The S/360 architecture define two execution modes: problem state and supervisor state.
- The difference is that instructions that can change the state of the system require supervisor state. The instructions typically used by application could execute in problem state. No problem state instructions behaved differently in supervisor state.
- CP-40 took advantage of this. It ran guests in problem state and therefore would get control on a program interruption whenever the guest issues an instruction requiring supervisor state.
- When such an interruption occurred, CP-40 would simulate the instruction within the context of the guest “virtual machine”.
- Most guest instructions executed at full speed, while an occasional supervisory instruction on the guest operating system (e.g. CMS) experienced the overhead of simulation by the hypervisor.
- Because of this mode of operation, the guest’s environment is called a “virtual machine” and CP-40 and its descendants are called hypervisors rather than operating systems.

CP-67 and the Model 67

- CP-40 was updated to become CP-67 which was made available to customers – basically as freeware.
- CP-67 took advantage of some enhanced features of the Model 67 which supported both 24-bit and 32-bit addressing.
- The Model 67 supported address translation tables that resided in memory – almost identical to the scheme used by current processors.
- It also had hardware to reflect reference and/or change of real memory.

System/360 Model 67

CP-67 and the Model 67

- To reduce the overhead of accessing the translation tables for every translation:
  - The Model 67 had an array of 8 associative registers to remember the most recent translations for data access
  - It also had an additional register to remember the translation for the current instruction address.
  - There were called the Dynamic Look-aside Table (DLAT) but now we call it a Translation Lookaside Buffer (TLB).
  - Today, TLBs – and there are several in modern mainframes – can have as many as 1024 registers in a 4-way associative array of 256 rows.

Efficient Page Replacement

- The S/360 architecture defines a 5-bit storage key for each 4K of real memory.
- The Model 67 also implemented two additional bits indicating whether the referenced page had been referenced or changed since the last time the bits were reset.
- This enabled CP-67 to implement a least-recently-use (LRU) page replacement algorithm and avoid rewriting pages that have not been changed.
- Modern operating systems still use variants of LRU page replacement algorithms today.

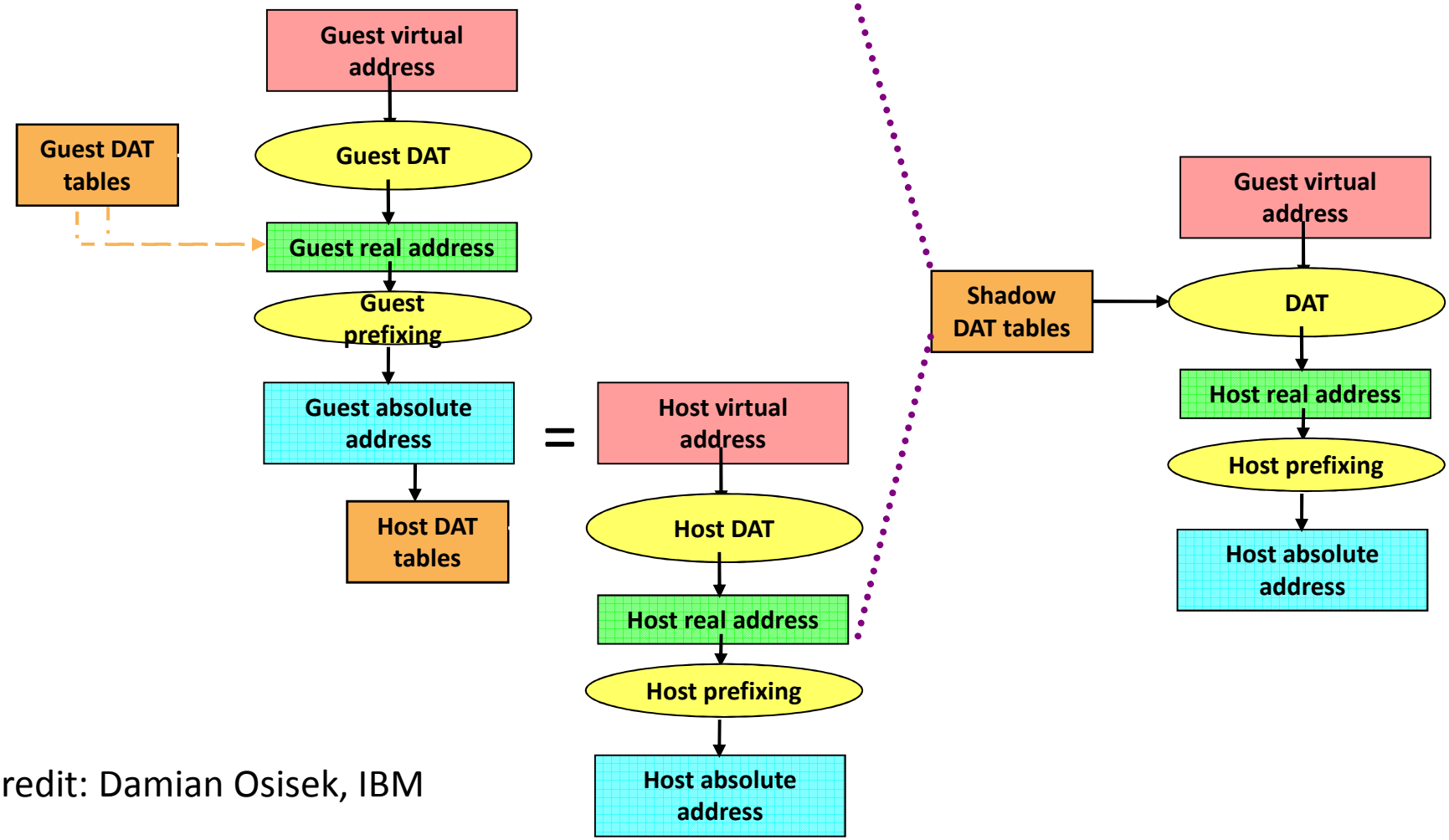
Multiple Address Spaces

- The Model 67 also had a control register that was used to locate a segment table – and its associated page tables - to be used for translating virtual addresses.
- CP-67 loaded this control register with the address of the segment table for a guest whenever that guest was dispatched on the CPU.
- This enabled CP-67 to support a large number of guests as opposed to the mere 14 guest supported by the CP-40 prototype.
- As more real memory became available, the number of guest virtual machine was not limited by an architectural constraint.

Virtualizing Virtual Memory

- Although CP-40 provided a S/360 virtual machine to its guests, it did not support guests which themselves defined virtual memory.
- CP-67 did support guest use of virtual storage. In fact, CP-67 could itself run on top of CP-67.
- The hardware supported only one level of translation so that CP-67 had to use the trick of “shadow translation tables” to get the job done.
- The tables were built by CP-67 to provide the composite translation of guest virtual to guest real (which is host virtual) and then to host real addresses.
- CP-67 built shadow translation tables “on the fly” as the guest accessed and later invalidated guest virtual memory.

Guest address translation using shadow tables



Credit: Damian Osisek, IBM

VM/370

- Although the first IBM S/370 processors shipped in 1971.
- But virtual memory support was not shipped until 1973 with the S/370 models 158 and 168.
- The CP-67 prototype was rewritten as VM/370 to support the S/370 architecture and was available when the hardware became available.
- VM/370 include the ability to run itself at second level as a standard feature.

System/370 Model 168

On to Optimization

- The virtualization provided by z/VM and PR/SM is, in essence, just optimizations on the operation of CP-67.
- This not to belittle optimization because it is optimization that turns a good, but financially unfeasible, idea into something people can build their business on.
- The advancements after CP-67 have been to drive more and more of the processing deeper into the iron to reduce any performance penalty presented by virtualization.

VM Assist (VMA)

VMA was introduced to reduce the overhead of instruction simulation by the hypervisor. A control register was used to expose some of the virtual machine state (e.g. virtual machine supervisor state).

1. Certain supervisor instruction were emulated by microcode when execute in guest supervisor state.
2. Supervisor Call (SVC) instruction emulated for the guest to avoid interruption to the hypervisor.
3. Shadow table validation: when an invalid entry in the shadow translation tables is encountered, microcode looks at the guest and host translation tables and validates the shadow table entry if the page is determined to be in real memory.

Extended Virtual Machine Assist (EVMA)

- EVMA emulated additional privileged instructions.
- It also provide a fast path into the hypervisor when a Diagnose instruction was issued by a guest. The Diagnose is means to request services from VM. It is used extensively by CMS.
- A interval timer assist emulated the Interval Timer for guest machines and presented the interval timer in interrupt directly to the guest if possible.

Preferred Guest Support

- One of the major uses of VM/370 was to be able to run multiple OS occurrances (e.g. test and production) on a single machine.
- One way to improve the performance of a production guest is to avoid the overhead of host address translation
- By giving the preferred guest the real memory starting with location 0, one level of translation is eliminated (called virtual equal real, V=R).
- Also, channel programs no longer needed to be translated.

Preferred Machine Assist (PMA)

- Preferred Guest support provided a good boost for production systems, but VM still needed to be involved in all I/O operations.
- PMA provided I/O Passthrough for the preferred guest.
  - I/O Passthrough was selected at a channel granularity.
  - I/O operations over authorized channels were executed directly and interruptions could be presented directly to the preferred guest.
- Guest Survival: the preferred guest could be given direct control of the processor if VM failed.

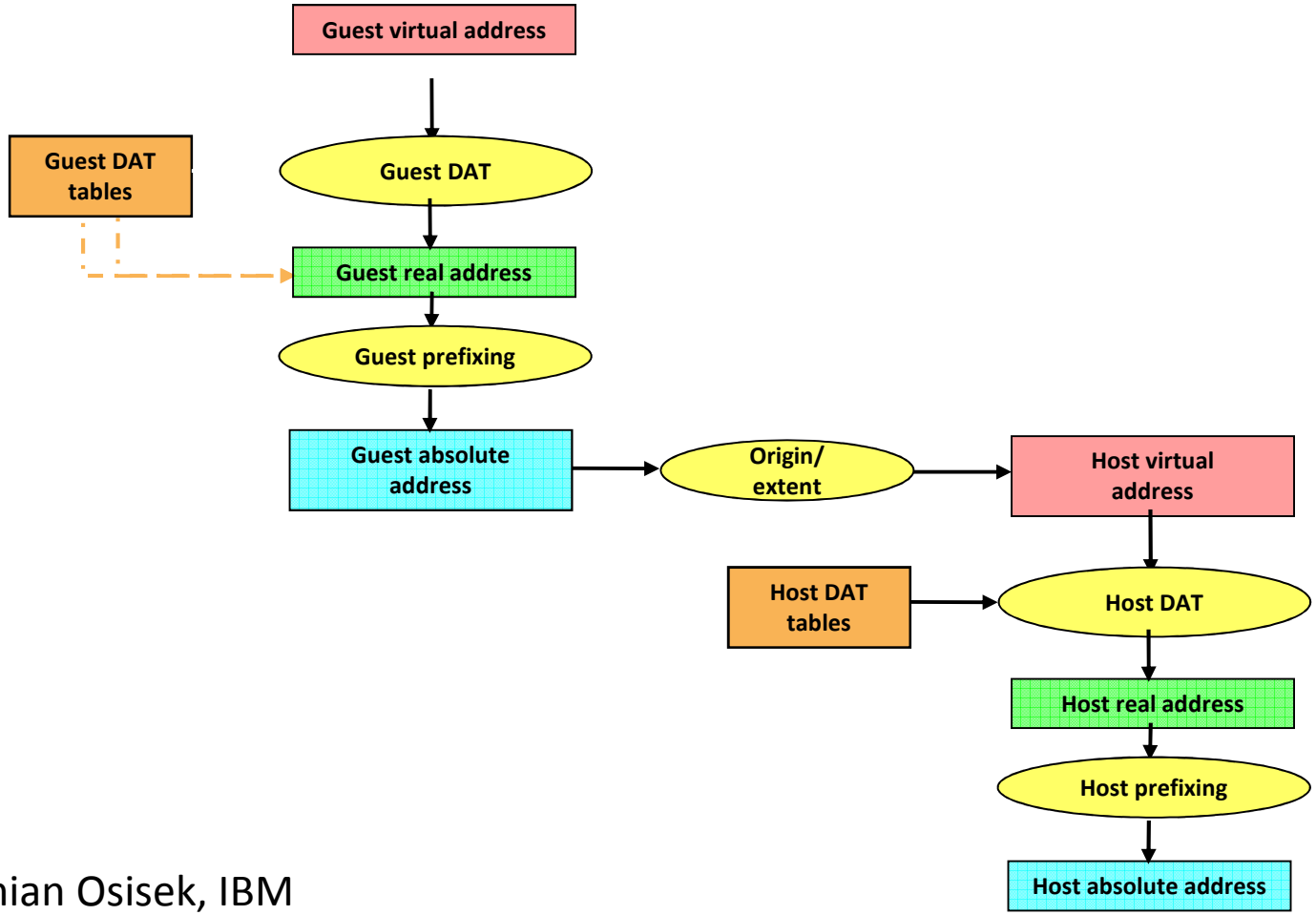
Interpretive Execution

- The 370-XA architecture introduced not only 31-bit addressing and dynamic channel architecture but also interpretive execution.
- A state descriptor was introduced which provided a near-complete representation of the virtual machine state.
- The Start Interpretive Execution (SIE) instruction took this descriptor as input and updated it when interpretive execution was exited.

SIE capabilities

- SIE provided the capability to emulate nearly all privileged instructions.
- It supported two levels of address translation to eliminated the need for shadow tables.
- The TLB recorded guest virtual to host real translation for added efficiency.
- Guest translation faults were routed directly to the guest.
- It provided separate host and guest timing facilities.
- Enabled maintenance of both host and guest reference and change bits to support page replacement.

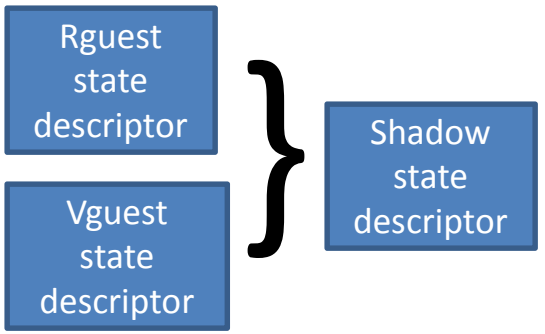
Pageable guest address translation under SIE



Credit: Damian Osisek, IBM

Virtual SIE

- The XA architecture supported only one level of interpretive execution so it did not support running VM under VM in interpretive mode.
- VM implemented VSIE to simulate SIE issued by an immediate guest (RGuest) to run a higher-level guest (VGuest).
- A shadow state descriptor was built as a composite of the RGuest and VGuest state descriptors and shadow translation tables were reintroduced.



PR/SM (aka LPAR)

- PR/SM extended the preferred guest concept to multiple preferred guests.
- Each guest (partition) is provided with a contiguous block of real memory (called virtual=fixed, V=F)
- Each “zone” is bounded by origin and limit which are used in lieu of host translation tables for COU access.
- Zone relocation was provided in the I/O subsystem to obviate channel program translation.
- Two levels of SIE enable VM to run with high performance in a PR/SM partition.
- Channel paths could be shared across partitions.
- Starting with the IBM System z990, PR/SM is mandatory – native mode is no longer supported.

A Word About KVM

- KVM is a hypervisor that run within Linux.
- It is available on System z
- The differences from z/VM are superficial other than that KVM runs within an Operating System and z/VM runs on bare hardware.
- Both use the System z SIE architecture
- For a lot more comparison, see Christian Ehrhardt's presentation *KVM for z/VM Lovers,* at SHARE Summer 2014.

Some References

- R. J. Creasy, " The origin of the VM/370 time-sharing system ", *IBM Journal of Research & Development*, Vol. 25, No. 5 (September 1981) http://pages.cs.wisc.edu/~stjones/proj/vm_reading/ibmrd2505M.pdf
- Gum, P.H. "System/370 Extended Architecture: Facilities for Virtual machines," *IBM J Res Dev* 27:6, 1983. http://www.research.ibm.com/journal/rd/276/ibmrd2706L.pdf
- Osisek, D.L. *et al.* "ESA/390 interpretive-execution architecture, foundation for VM/ESA," IBM Systems Journal 30:1, 1991.http://www.research.ibm.com/journal/sj/301/ibmsj3001E.pdf
- Rogers, R.R, "The What and Why of zEnterprise Millicode", http://www.ibmsystemsmag.com/mainframe/administrator/performance/millicode_rogers/