Shashi Shivaraju
XID:C88650674

Paper Review 2 - A Comparison of Software and Hardware Techniques for x86 Virtualization

This paper describes the software only and hardware-assisted virtualization techniques used for x86 architecture. Until recently x86 architecture had not permitted classical trap-and-emulate virtualization and the need to virtualize unmodified x86 operating systems has given rise to software techniques that go beyond the classical trap-and-emulate Virtual Machine Monitor (VMM).VMM for x86 have instead used binary translation of the guest kernel code for virtualization. Recently, the major x86 CPU manufacturers have announced architectural extensions to directly support virtualization in hardware. This paper tries to examine and compare the strengths and weaknesses of existing software VMM with a new VMM designed for the emerging hardware support.

Initially the paper reviews Classical virtualization as per. Popek and Goldberg’s 1974 paper which establishes three essential characteristics for system software to be considered a VMM: Fidelity,Performance,Safety. It reviews the most important ideas from classical VMM implementations virtualized purely with trap-and-emulate: deprivileging, shadow structures and traces.

x86 without hard-ware support is not classically virtualizable (i.e. virtualized purely with trap-and-emulate) due to: Visibility of privileged state and Lack of traps when privileged instructions run at user-level. x86 without hard-ware support is virtualizable using a software technique known as binary translation (BT), wherein the guest executes on an interpreter instead of directly on a physical CPU. Thus a VMM built around a suitable binary translator can virtualize the x86 architecture and it is a VMM according to Popek and Goldberg. Modern CPUs have expensive traps, so a BT VMM can outperform a classical VMM by avoiding privileged instruction traps. Additionally use of adaptive BT to essentially eliminates traps and thus improves overall efficiency.

Recent hardware architectural changes known as x86 architecture extensions permits classical virtualization of the x86.The x86 architecture extensions exports a number of new primitives to support a classical VMM for the x86. An in-memory data structure called virtual machine control block (VMCB) which combines control state with a subset of the state of a guest virtual CPU. A new, less privileged execution mode, guest mode, which supports direct execution of guest code, including privileged code. A new instruction, vmrun, transfers from x86 execution host environment to guest mode. These hardware extensions provide a complete virtualization solution, essentially prescribing the structure of hardware VMM.

An ideal VMM runs the guest at native speed. A guest that never exits runs at native speed, incurring near zero overhead. Reducing the frequency of exits is the most important optimization for classical VMMs. The software and hardware VMMs suffer from different overheads in their attempts to approach this ideal. The authors of the paper compare an existing software VMM with a new VMM designed for the emerging hardware support (VMware’s experimental VMM).They examined a number of 64-bit workloads under VMware Player 1.0.1’s software and hardware-assisted VMMs. To better understand the performance differences between the two VMMs, the authors of the paper wrote a series of “nanobenchmarks” that each exercise a single virtualization-sensitive operation. The experiments shows that both software and hardware VMMs both perform well on compute-bound workloads. For workloads that perform I/O, create processes, or switch contexts rapidly, software outperforms hardware. In two workloads rich in system calls, the hardware VMM prevails.

Shashi Shivaraju
XID:C88650674

The experimental results show that the hardware support fails to provide an unambiguous performance advantage for two primary reasons: first, it offers no support for MMU virtualization; second, it fails to co-exist with existing software techniques for MMU virtualization. Thus in general software virtualization requires careful engineering to ensure efficient execution of guest kernel code whereas the hardware virtualization delivers native speed for anything that avoids an exit but levies a higher cost for the remaining exits (on current hardware).The software VMM has a richer set of options available, including adaptation, whereas current hardware mechanisms aim more narrowly at trap-and-emulate style virtualization, leaving less flexibility to use other software/hardware combinations.

The paper discusses few future software and hardware approaches in both hardware and software to close the gap between hardware VMM and software VMM performance, and to ultimately approach native performance. Impact of Micro-architecture,Hardware VMM algorithmic changes (to completely remove exits), hybrid VMM,Hardware MMU support on performance are discussed as possible future implementations. The authors of the paper believe that future hardware support should be be guided by an understanding of the trade-offs between today's software and hardware virtualization techniques. They hope that coming hardware support for virtualization will be designed to blend easily with and complement existing software techniques. Thus it may contribute to faster adoption of the hardware and advance the general state of virtualization.