SCIENCE CHINA
Information Sciences

• LETTER •

June 2022, Vol. 65 169101:1–169101:3
https://doi.org/10.1007/s11432-019-2769-4

SWVM: a light-weighted virtualization platform based on Sunway CPU architecture

Jianguo YAO[1*], Qiumin LU[1], Xingyan WANG[2], Chao WANG[2], Hanyang MA[1] & Haibing GUAN[1]

[1]*School of Software, Shanghai Jiao Tong University, Shanghai* 200240*, China;*
[2]*Wuxi Jiangnan Institute of Computing Technology, Wuxi* 214083*, China*

Received 21 July 2019/Revised 17 October 2019/Accepted 27 December 2019/Published online 13 May 2021

Citation Yao J G, Lu Q M, Wang X Y, et al. SWVM: a light-weighted virtualization platform based on Sunway CPU architecture. Sci China Inf Sci, 2022, 65(6): 169101, https://doi.org/10.1007/s11432-019-2769-4

Dear editor,

The national-dominated Sunway CPU architecture [1–3] is now becoming mature in supporting professional application areas from large-scale numeric computing in supercomputer to maintaining online services in the network. However, as a newly-developed platform, its supporting environment still requires perfection in developing more system functionalities to improve its applicability, where the virtualization support is a critical one for cloud computing and other advanced techniques for distributed and large-scale computing. We propose the SWVM, a light-weighted virtualization solution based on Sunway architecture with high flexibility, and introduce the design of its structure with the included functional modules. The description about this solution covers the mechanism of how the virtual machines are initialized and maintained, and how the virtualization of CPU, memory and IO is realized on the basis of the specific Sunway architecture. According to the evaluations, this SWVM solution has good performance with acceptable loss in the virtualized environment, which supports good isolation in computing resources and efficient workload performance when the platform contains multiple running virtual machines sharing the computing resources concurrently.

Sunway architecture. (1) **Platform.** The Sunway series microprocessors are inspired by the design based on the Alpha 21164 [4]. It uses a RISC architecture instead of the CISC architecture which is applied in Intel processors, and as a result requires different virtualization implementation [5] compared with supports in X86 [6]. It has the individual instruction sets and designs its customized assembly codes. Each SW1600 series processor has 16 cores, and each core has 8 KB instruction cache and 8 KB data cache in L1 cache and 96 KB L2 cache. And it has quad-channel 128-bit memory support. In SW1600 series architecture processors, it uses a 43-bit virtual address and 40-bit physical address in the system so it can support up to 8 TB virtual memory and 1 TB of physical memory space.

(2) **Privilege design.** To support the virtualization mechanism, an important feature of Sunway instruction set architecture is the privilege level system limiting and defining the behaviors of the executed instructions, and the related protocol of storing and switching such critical states. First, the storing pattern of privilege level is based on the reserved space of the program counter register in the Sunway CPU chip. The two digits PC[1:0] are adopted to distinguish each privilege state including hardware, virtual, kernel and user modes. Some privileged instructions such as pri_wcsr, pri_rcsr, pri_ld and pri_st are only available in hardware mode, and all these privileged instructions can access the physical address. Two specific privileged instructions, sys_call and pri_ret, are used for privilege level switching. The sys_call instruction can be invoked from any other privilege modes and then the execution process will be trapped into the hardware mode, where specific privileged code will be executed. When the process is finished, the instruction pri_ret will be invoked, which can only be called in hardware mode, and will result in the return to other modes.

(3) **HMCODE content.** The HMCODE is a group of shared procedures for bottom-layer supporting with privileged functionalities. These privileged codes are aggregated into a bottom-level library to simplify the system design, which acts as an indispensable component of the SWVM virtualization solution. In the Sunway architecture design, these codes are loaded at first during the boot initialization. After that, all the bottom-level hardware operations can only be finished through the execution of these privileged codes. In the Sunway architecture, all the bottom-level operations will first switch to the hardware privilege level and then jump into the HMCODE scope to execute the related codes. As a result, most IO behaviors or hypervisor functionalities can only be accomplished by executing HMCODE.

SWVM design. (1) **Top-level structure.** The SWVM platform is constructed according to the hosted hypervisor structure as shown in Figure 1 [7], where the virtualization components are embedded into the host operating sys-

* Corresponding author (email: jianguo.yao@sjtu.edu.cn)

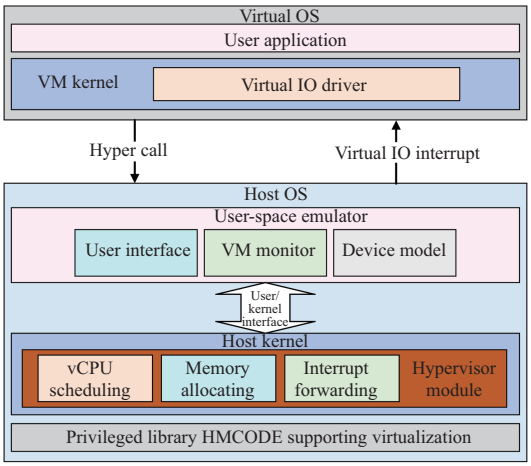

Figure 1 (Color online) The overall design of the SWVM system.

tem at different layers, and the activated virtual machines act as normally executing programs in this operating system [8]. When the user intends to start a virtual machine, the user-space emulator accepts the user commands, and then invokes the kernel/user-space interfaces to activate the functions in the hypervisor kernel module, where the virtual machine instances are created, initialized and configured. Then during the lifecycle of the virtual machine, the hypervisor manages the virtual CPU scheduling and memory address translating, along with the interrupt forwarding to support the IO operations.

First, the HMCODE here serves as the interface in the switching process among different hosts and virtual scopes, and also provides the functions of hardware operations for the virtualization hypervisor, such as changing hardware states, operating memory allocation and processing interrupts. Then the hypervisor kernel module is the critical component for the virtualization solution embedded in the operating system kernel of the host, which provides and realizes most of the fundamental virtualization functionalities, such as scope switching and scheduling for the CPU virtualization, address translation, and allocated region management for the memory virtualization, along with the necessary kernel/user-space interfaces for communication and monitoring. The user-space emulator includes all the virtualization functionalities with no necessity to be embedded into the kernel. Generally, it provides user command interfaces, emulates the virtual devices to support the virtual machine IO and monitors the execution of the active virtual machines.

(2) **CPU virtualization.** The first part of the CPU virtualization implementation is the whole process of instruction simulations. In SWVM, each thread is created and allocated for a vCPU when the virtual machine is initialized. Then common instructions are handled in those threads. Privileged instructions will cause a trap into hardware mode, which switches the context to the host OS and invokes corresponding functions in the SWVM hypervisor. The second part of the CPU virtualization implementation is handled by the hypercalls which trap into the hardware mode and invoke the HMCODE functionalities. The host can get the data from the guest through the register which is set before the context switches. And the guest can also get the data from the host after interrupts. The third part is the CPU resource scheduling for multiple vCPUs running on a pCPU. In SWVM, we adopt the round-robin mechanism to schedule a pCPU between multiple vCPUs along with tasks in host scope. Each vCPU will be viewed as a single thread running on a certain pCPU. The pCPU schedules those threads of the host and the guest together. When the time slice is scheduled to a virtual machine, the control of the pCPU will be transferred to the active vCPU thread and the execution context will be switched to the guest by HMCODE.

(3) **Memory virtualization.** As there is no hardware page table in the Sunway CPU architecture, the page table of Sunway architecture is created and managed by software, which refers to the chance to construct an efficient solution of memory virtualization in SWVM. For efficiency, we pre-allocate the memory space required for a virtual machine when initializing the guest in the SWVM hypervisor. The address translation from the virtual memory address to the physical memory address is based on the start address and size of the allocated memory blocks stored in the data structure of the hypervisor. When an instruction of reading from or writing to the virtual memory is executed in the guest OS, the address base register configured with the offset decided by the memory region pre-allocation in virtual machine (VM) initialization process is adopted to get the correct memory mapping between the virtual memory address and the corresponding physical memory address. As the memory is pre-allocated to the guest while initializing from the perspective of the host, page faults will never occur in the host memory address space. If a page fault occurs in the guest OS, it can be detected and solved directly by the SWVM guest kernel. The design is easy to be implemented with considerable efficiency. The safety and isolation of the virtual memory are guaranteed by the implementation of software page table in the Sunway architecture.

(4) **IO virtualization.** The key of IO virtualization is the mechanism by which the host OS send interrupts to the virtual machine. All interrupts should be handled in the host OS by the device driver. So, the problem comes down to the interrupt forwarding. This implementation is also based on the hypercall mechanism mentioned above. When the data transfer happens, the guest invokes a hypercall to forward the request parameters including the data location. Then, the context is switched to the host and parameters are received by the user-space emulator. The hypervisor then handles the request by manipulating the block devices. After the request is done, the corresponding data pointers and the result will be stored in registers and the running context is switched to the guest by hypervisor. We use a virtual queue for virtual IO devices handling their messages in order in our implementation. Moreover, for some PCI devices, we apply a para-virtualization solution for the guest. By modifying the guest kernel, the guest is able to interact with devices without invoking a hypercall. It greatly improves the efficiency of SWVM.

Evaluation conclusion. We execute our evaluations on a Sunway server with 16-cores SW6A CPU, 32 GB memory and HDD disks. The operating system is Ubuntu Kylin 16.04 with linux kernel v4.4.15 specially modified and compiled for Sunway CPU architecture. According to the benchmark evaluation result, compared with the host, a certain vCPU can achieve 99.1% efficiency of the host. The memory performance of the guest OS is about 98% of the host OS. And the performance of the guest is about 48.4% of the host OS in disk IO.

Also, in the situation with multiple VMs, the perfor-

mance result is as follows. The performance of a single VM running on the server is taken as our guideline. The CPU and memory performance achieves almost 98% on average when there are less than 4 VMs running. When the number of virtual machines is more than 4, the performance of CPU decreases a lot. The performance of threads is almost 94% with multiple VMs and also decreases after 5 or more VMs being executed. However, there is only 50.49% efficiency when there are 2 virtual machines, and it decreases to 32.3%, 25.1% and less along with more VMs executed. The maximum number of available virtual machines reaches the level of 11 virtual machines concurrently because of the bottleneck caused by IO performance degradation. Obviously, this maximum number cannot be further improved through modifications on CPU or memory resource configurations.

Acknowledgements This work was supported in part by National Key Research & Development Program of China (Grant No. 2018YFB1003603), National Natural Science Foundation of China (Grant Nos. 61772339, 61572322), and the National Defense (Grant No. 61400010107).

References

1 Fu H H, Liao J F, Yang J Z, et al. The Sunway TaihuLight supercomputer: system and applications. Sci China Inf Sci, 2016, 59: 072001
2 Dongarra J. Report on the Sunway Taihulight System. Tech Report UT-EECS-16-742, 2016
3 Dongarra J. Sunway TaihuLight supercomputer makes its appearance. Natl Sci Rev, 2016, 3: 265–266
4 Kessler R E, McLellan E J, Webb D A. The alpha 21264 microprocessor architecture. In: Proceedings of International Conference on Computer Design: VLSI in Computers and Processors, 1998. 90–95
5 Dall C, Nieh J. KVM/ARM: the design and implementation of the linux arm hypervisor. ACM SIGARCH Comput Archit News, 2014, 42: 333–348
6 Neiger G, Santoni A, Leung F, et al. Intel virtualization technology: hardware support for efficient processor virtualization. Intel Technol J, 2006, 10: 167-177
7 Popek G J, Goldberg R P. Formal requirements for virtualizable third generation architectures. Commun ACM, 1974, 17: 412–421
8 Kivity A, Kamay Y, Laor D, et al. KVM: the linux virtual machine monitor. In: Proceedings of the Linux Symposium, Dttawa, 2007. 225–230