**Dune: Safe User-level Access to Privileged CPU Features**

*Adam Belay, Andrea Bittau, Ali Mashtizadeh, David Terei, David Mazieres, Christos Kozyrakis; Stanford University*

**Link:**<https://www.usenix.org/conference/osdi12/technical-sessions/presentation/belay>

**Summary:**

The main motivation behind developing dune was to allow users to safely access privileged hardware features at a user level rather than at the kernel level thereby enhancing the performance of applications. Dune is a system that enables applications to safely access privileged CPU features such as ring protection, page tables and hardware TLB’s without altering the main page tables or hardware features of the OS kernel. If an application wants to customize the page table according to its requirements then by using Dune that can be done without altering the original page tables that stay intact and can be used by other applications without any side effects. The Dune system uses virtualization hardware to provide process rather than machine abstraction. A dune module present inside the kernel of an operating system initializes the virtualization hardware and is responsible for mediating communication with the kernel. The system also consists of a small dune library called libDune is provided to the applications for easy access to privileged features and allows the user to customize those features according to their own requirements to significantly improve the performance of the application.

Providing applications the ability to use kernel-only features can provide substantial benefit to the application. To develop such systems we are often required to change those features in the kernel which is highly intrusive and may affect the whole systems stability if done incorrectly and changes made to the kernel to improve the performance of one application may not be compatible with how a second application might use those features. A way to solve this problem would be to use a virtual machine on the host operating system but it does not integrate very well with the host operating system and would require a lot of work in order to get it to work.

Dune solves that by using virtualization hardware that is ubiquitously available in most of the computer systems available today to provide process abstraction to the application. A loadable Dune kernel is loaded in the kernel module and with the help of virtualization, the application that runs in guest mode, gets safe and fast access to privileged kernel functions including privilege modes, virtual memory registers, page tables, and interrupt, exception, segmentation and system call vectors that can be invoked using the dune library. Since process virtualization is used, the original kernel functions are unmodified and the application gets access to its own modified copy while at the same time guaranteeing that the other applications running on the system are not affected at all and also being able to fully access other features like memory sharing and utilizing the operating systems file system just as a regular process would. Since machine abstraction is not done and a POSIX module is invoked directly, the dune module is both simpler and faster. The key idea is to leverage the existing virtualization hardware. Dune uses the VT-x to provide applications with full access to x86 protection hardware that places the kernel in VMX root mode (the host mode). In a traditional OS when an exception occurs, reporting them to a user program requires privileged kernel mode system calls and use of signals. Dune reduces the exception overhead by using VT-x that delivers exceptions directly in the hardware. Page table updates can be performed in batches as directed by the application since the ability to manually control TLB invalidations is also given to the user programs in Dune. A single program may even switch between multiple page tables. Dune also exposes access to privilege modes such as ring 0(supervisor mode), ring 1 , ring 2 and ring 3(user mode) since VMX non-root mode (the guest mode) maintains its own set of privileged modes and therefore, dune offers hardware enforced protection within user programs in the same way a kernel restrains user program’s access to privileged features. Signals are delivered as injected interrupts which switches the hardware to ring 0.

VMCALL (a hypercall) is used by a Dune process to perform normal kernel system calls. The Dune module intercepts these VM exits and forwards the request to the kernel system call table. Dune can be enabled for a process by initiating a transition to ioctl on the /dev/dune device. Once a process enters Dune mode, it cannot disable Dune. If the process chooses the fork, then the newly created child, unless done explicitly, will not have the dune mode enabled whereas the parent will continue to have it enabled. libDune is an untrusted user library which contains a collection of useful utilities for manipulating page allocator tables, intercepting exceptions and system calls, managing page tables, etc. All the applications built in Dune are built on top of this library. Dune can also be used for sandboxing since it runs untrusted code in ring 3(the least privileged mode) that invokes the system call handler of the process rather than invoking the kernels system call handler.

Dune adds extra overhead when making transitions from kernel to guest mode and vice-versa. Also overhead is added as a result of the EPT transitions. However, there is a large scope for optimization in Dune since it directly uses the kernel level features in user programs. It has fast system call interpostition and traps and provides user level virtual memory Allocation. Dune results in a 7x increase over Linux in Appel and Li user level virtual memory benchmarks. In my opinion, this was a great research that clearly demonstrated how performance can be significantly improved by customizing some of the privileged features without effecting the system as a whole. It also lists out various scenarios where Dune can be used and the performance boost caused because of that. An interesting area to explore further would be to divide the kernel into data plane and control plane as in the Arrakis kernel, using the I/O device’s ability for virtualization thus speeding up I/O and use the Dune module in the control pane and the libDune deployed on the I/O to access privileged CPU features and check if that further enhances the performance.