Summary and Reaction:

_Disco: Running Commodity OS On Scalable Multiprocessors_ 
NCS490 – Virtualization 
Patrick Araya 
February 12, 2013
Disco is a prototype virtualized system designed to run on the Stanford FLASH, large-scale shared memory multiprocessor. It was written in about 13,000 lines of code providing less overhead while utilizing Silicon Graphics' IRIX operating system. Scalability and non-uniform memory access are easily handled due to the low overhead of the monitor. Disco emulates a virtual machine monitor because it runs between the operating system and the physical hardware on the machine. Simultaneous instances of “commodity” operating systems can share a pool of resources on a single scalable workstation. However, Disco operates more efficiently than previous machine monitors in that its overhead is lower and resource sharing is much more efficient. The shared resources create a global buffer cache that virtual machines cannot see.

The FLASH multiprocessor that Disco uses contains multiple processor nodes, memory, and I/O devices. The implementation of nodes provides for an increased performance and scalable environment. A directory provides non-uniform memory access to the shared memory. Each VM can run simultaneously on the same hardware because Disco successfully virtualizes the resources to each machine. Memory is allocated starting at address zero because that is how each VM will expect to use the memory. A dynamic page migration and replication emulates uniform memory access to each VM.

When a commodity operating system is ran, Disco sets the physical register on the CPU to that of the virtual and the PC of the virtual CPU jumps. By matching the CPU’s registers, performance is concurrent to that of using a physical system. Using this method, Disco preforms like preceding machine monitors.

Disco requires several changes to the system before successful implementation. The first step in setting up Disco is relocating the kernel code along with the data located in the KSEG0 segment. To correctly fix this the unmapped segment in the virtual machine is remapped to a supervisor segment in the MIPS. Another change for increased performance on Disco is to patch the HAL, or else an increased overhead occurs in the kernel code during synchronization. The final fix for configuring Disco is to create
a properly aligned unwritten page file, which allows for virtual networking devices to utilize remapping techniques.

SimOS machine simulator is used to test Disco because FLASH machines were unavailable at the time. Overheads were tested by running tests on a uniprocessor, IRIX directly on simulated hardware, and finally Disco running IRIX on a single VM. IRIX is configured and tested against 1, 2, 4, and 8 virtual machines with a proportional amount of memory, with the 8 VM configuration using memory as an NFS server for memory. Each configuration performed comparatively or better than bare-metal IRIX. Memory paging hotspots are eliminated due to the consecutive nodes. Scalability of Disco increase as a new VM is added, with a reduction of 60% in execution time. The NFS memory access also increase performance, even with a higher overhead.

I thought it was surprising how Disco was able to jump the register on the CPU to match that of the VM, which the VM then operated as fast as a physical machine. One quark I had was that they designed Disco to run on a FLASH machine, however for their testing purposes, a FLASH machine was unavailable and had to use SimOS. This caused issues during testing as well because some of the resource sharing policies weren’t possible without FLASH. But I did like how they took steps to ensure SimOS didn’t skew data by enabling fast mode to minimize cold start effects.