Joseph Paris
CS443

Disco: Running Commodity Operating Systems on Scalable Multiprocessors

Bugnion et al.

The purpose of Disco is to examine the problem of extending modern operating systems to run efficiently on large-scale shared memory multi-processors without a large implementation effort. The idea is to run a number of commodity operating systems on virtual machines governed by a virtual machine monitor. The reasoning behind doing this is that system software for these machines has often trailed hardware in achieving the functionality and reliability required by its users. So, instead of modifying existing operating systems to run on scalable shared-memory multiprocessors, an additional layer (vm monitor) is inserted between the hardware and the operating system. The monitor virtualizes all the resources of the machine, exporting a more conventional hardware interface to the operating system. The monitor manages all the resources so that multiple virtual machines can coexist on the same multiprocessor. For example, the monitor can move memory between vm's to keep applications from paging to disk when free memory is available in the machine. In addition, the monitor dynamically schedules virtual processors on top of the physical ones to balance the load across the machine. Basically, the monitor creates a virtualization of every resource on the machine.

As such, a lot of overhead is introduced into the system. For example, operations such as the execution of privileged functions cannot be safely exported directly to the operating system and must be emulated in software by the monitor. In addition, access to I/O devices and physical memory is virtualized, so requests must be intercepted and remapped by the monitor. While these might not be huge hits to performance, they do require some overhead which will slow down the application. However, the direct mapping to physical processors is a significant gain, however the virtual monitor still needs to intercept and emulate operations that cannot be safely exported to the virtual machine.

I didn't care too much for their performance analysis of the Disco system. They seemed to want to offload much of the slowdowns seen in Disco onto faulty underlying OS issues or the architecture. Also, many of the graphs seemed incoherent and required in-depth reading to understand the point they are trying to make. While that is not a huge issue it tended to lend itself towards issues/slowdowns seen in the Disco system due to emulation, etc. I think the point here is that you can't map OS's designed in a certain fashion to a virtual machine monitor and expect everything to speed up. Since those issues exist at the OS level the problems still exist no matter what scenario you shove them in. At most this is a test-case scenario that might provide some useful information on inherent scalability issues seen at a particular level of the OS/system.

A possibly better idea would be to extend the micro/exo-kernel methodologies to a scalable infrastructure. This way you have a more tightly coupled interface and user-level libraries that let you take advantage of the underlying architecture in any way that the application sees fit. In that you can have huge gains in performance while allowing the overall complexity to remain at a minimal.