Live Migration of Virtual Machines (review)

Ken Murphy
This paper talks about the challenges and problems associated with live migration of virtual machines. The authors chose to use the Xen hypervisor for their base VMM because it is free and easily modified. In the past migration was more focused on process migration instead of whole OS migration. This caused a number of problem however. For one, network state could not be preserved well, as that is part of the kernel space and not the user space. This was obviously not a good solution for interactive server programs as they may drop the users. By migrating the entire OS instead, all states are preserved and less downtime is observed. The authors say that in some instances they can see downtimes as low as 60ms.

The authors mention a variety of methods typically used in live migration. The first is ‘pure stop and copy’. This method involves halting the virtual machine, copying all memory, then resuming execution on the new host. The obvious disadvantage of this method is that downtime is high. It also means that the more memory a virtual machine has, the longer the downtime will be. The next method is ‘pure demand-migration’. This method stops the virtual machine for a very short amount of time and copies only kernel data to the new host. The virtual machine is then started on the new host and memory is copied to it as it is accessed. This method has many problem associated with it. For one, the new virtual machine will probably be very slow for some time after it is started due to it needing to pull all accessed memory across the network. It also requires the previous host to remain up for quite a while after the migration is initiated. The method the authors used it called ‘pre-copy migration’. This means that all memory is first copied to the new host. After this completes the virtual machine is stopped for a
very short amount of time. This allows the host to copy all memory that has been modified since
the initial copy. The virtual machine is then resumed on the new host. This method typically
incurs a very small amount of downtime, does not require the old host to remain running
unnecessarily, and does not slow down the virtual machine when it is resumed.

The second problem with live migration is how to deal with local resources that have
been allocated to the virtual machine in question. As the entire operating system is migrated as
well this problem is not as much of an issue as it was when process migration was more
popular. Network state is preserved as it resides in the kernel, all that must be done is to inform
the switch that the IP is on a new port. The authors mention a variety of methods to ensure that
this process completes smoothly. Another issue is that migrating a virtual machine at full speed
may degrade performance as you are tying up all the computers resources, such as network and
memory bandwidth. To work around this they employ algorithms to ensure that the host is not
negatively affected.

The authors used a variety of programs in order to benchmark their live migration
system. The first was Apache 1.3 being benchmarked with SPECweb99. They served a static
page and migrated during the middle of the benchmark. Their graph showed a 165ms downtime,
which is not bad at all. Their second test was a low latency server, Quake 3. Game servers
require low latency and response times to keep the experience enjoyable for the clients. During
this test they only showed a 60ms downtime. They say that the performance degradations
associated with the migration were not noticeable to the users playing on the server. The next
test involved a workload where memory is being changed faster than it could go out to the new
host. In this case they saw a downtime of 3.5 seconds as their algorithm noticed that it was
impossible to transfer the memory quickly enough while running live.

This paper was interesting. The authors raised a number of concerns associated with
live migration of virtual machines. They also developed a method and algorithm to minimize
downtime and maximize performance during such migrations.