Implementing Linux-Enabled Condor in Windows Computer Labs

Dr. Henry Neeman, Director of Oscer
Dr. Horst Severini, Associate Director for Remote & Heterogeneous Computing

Chris Franklin, Computer Science undergrad, IT employee
Joshua Alexander, Computer Engineering undergrad, IT employee

What is Condor®?

Condor® is a program developed by the University of Wisconsin to allow desktop computers to harness idle time to perform computationally intensive operations. See "http://www.cs.wisc.edu/condor/" for more information about Condor®.

Why do you need it?

Condor® provides free computing cycles for scientific and research use, which extends current super-computing resources by adding additional computing time.

If this is so simple, why can't I just install it?

Most scientific and research programs are designed for Linux, but most desktops are running Windows®. This means that Windows® computer labs would not be useful for researchers.

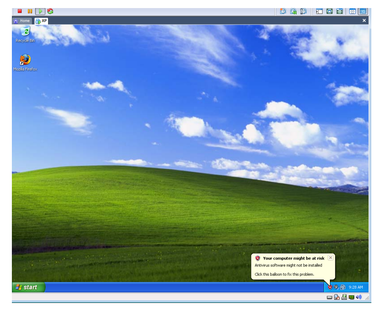
What's the solution?

Install Linux inside Windows®.

The first attempt

The initial solution we devised was to install VMWare® within a native Linux install, and then to install Windows inside VMWare®. The steps were:

1. Install Linux as the native host operating system
2. Install Condor inside Linux
3. Install VMWare® inside Linux
4. Install Windows® inside VMWare



We installed this solution on approximately 200 lab computers across OU's campus during the summer of 2005.

During testing, we noticed a significant performance decrease using Windows inside VMWare®. To alleviate this problem, we changed VMWare® to use raw disk mode. This mode significantly increased disk performance inside VMWare®.

Once we deployed VMWare® in the labs, several more issues appeared:

- CD/DVD Burning from inside VMWare® did not work for some hardware
- Some USB thumb drives had trouble working with VMWare®
- Our performance optimizations resulted in increased likelihood of disk corruption if the computer was improperly shut down, either by power failure or people power-cycling the machines
- Some people accidentally happened upon the key combination to change VMWare® out of full-screen mode
- VMware® offered no 3d hardware support, so some programs would not work

The Linux environment and Condor both ran fine, but the end-user running Windows on the lab computers had numerous problems.

Our Next Attempt

At this point, we realized that many of these problems could not be fixed cost-effectively, and that new issues caused by VMWare® were appearing. Searching for a new solution, we found out about Cooperative Linux (coLinux), which is available for free at "http://colinux.sourceforge.net/".

So what is coLinux?

coLinux is an open-source program that allows the user to run Linux inside Windows®. We tested using coLinux in the following manner:

1. Install Windows as the native host operating system
2. Install coLinux inside Windows
3. Install Linux inside coLinux
4. Install Condor inside Linux

Why did coLinux work better for us than VMWare®?

- CoLinux is completely invisible to the students using lab computers.
- CoLinux provides improved performance for end-users compared to VMWare®.
- CoLinux is not affected by the physical machines hardware, making it easy to install on all of the lab computers.
- Separate images would no longer be needed for machines running VMWare® and machines not running VMware®. This would result in easier, faster imaging and easier administration.
- CoLinux is free, open-source software.
- CoLinux is more easily customizable to take maximum advantage of new hardware

But, what problems were encountered running coLinux inside Windows®?

1) coLinux can use either bridged or NATed networking.
NAT networking requires setting up a generic connection broker (GCB) to allow Condor to communicate
Bridged networking requires an additional IP address for coLinux, but doesn't require setting up a GCB.
Bridged networking offered a better solution on our campus, because it didn't require setting up a GCB, which would require additional hardware (at additional cost) and additional time. Additionally, not using a GCB reduces complexity and maintenance.

2) Condor is natively able to monitor CPU and keyboard usage. This allows Condor to suspend its operation if someone is using the computer and resume operating once the user has left the computer. While running inside coLinux, Condor was not able to monitor the computer's CPU and keyboard usage. This meant that we needed a method to pass usage information into coLinux.

The solution we created was to write a VB script that monitors CPU usage and whether or not someone is logged in. Periodically, the VB script writes the CPU usage and how long it has been since someone was logged in to a directory that coLinux is able to read. Then, Condor reads that information in and suspends its operations if the Windows is using the CPU or someone logs in.

3) Some of our hardware had problems running coLinux. Out of the approximately 230 machines we initially deployed it on, a single lab of approximately 30 machines would not run coLinux. We solved this problem by installing the pre-release version of coLinux in that lab.

4) The Data Execution Prevention feature inside Windows®, when running on some newer processors, can conflict with coLinux and cause system failure. The solution to this problem is to add the /NOEXECUTE switch to the Windows® boot.ini.

Conclusion

Currently, we have approximately 200 computers currently running Condor inside coLinux. These computers have been running without problem for approximately a month. We expect to triple this number over the next month.

Still a work in progress, we're currently working on the following areas:
- Currently working on a "drop-in" CD for easy installation.
- Allow for additional monitoring of keyboard and mouse usage
- Vista compatibility

Of course there will be updates in the future to address any unexpected issues.