HDF Virtualization Review

Scott Wegner

Beginning in July 2008, The HDF Group embarked on a new project to transition Windows support to a virtualized environment using VMWare Workstation. We utilized virtual machines in order to improve the overall experience for Windows HDF users and developers, as well as to increase the stability of our Windows products through automated testing.

1 Introduction

Windows has historically been a difficult target for cross-platform software projects because of the vast differences that exist between it and other popular systems. For The HDF Group, Windows represents a particularly important platform, as 60% of all HDF5 users are working with the Windows version of the product. We found that by moving Windows development resources to virtual machines, Windows maintainers could work more effectively and provide a robust user experience for each of the Windows product configurations our users need.

1.1 HDF Windows User Experience

The needs and expectations of Windows users are different than those on other platforms. For example, Windows does not provide a native `make' tool and users prefer to develop in an IDE such as Visual Studio. It is important to have a simple build system that supports the users' familiar development environment. We allow users to build HDF4 and HDF5 in Visual Studio by providing special *project files* for each of the many Visual Studio versions we support. The build system is completely separate from that used on other platforms, but still contains all of the same library features and utilities. We also have a native test suite script run exclusively on Windows that users can invoke to ensure the HDF library is working properly. All of this is independent from the rest of our platform support and exists to ensure a quality experience for our Windows users.

HDF supports a growing number of platform and compiler configurations on Windows. Each requires separate project files and scripts, and thus separate maintenance. Unfortunately, the Windows development environment is fragile; complications often arise when multiple compiler versions are installed side-by-side. Application and library DLLs need to be installed in a global system path and can conflict with different versions. Installed applications will often augment the system environment in ways that can conflict with other development tools.

Virtualization helps alleviate the growing burden of testing and maintaining the various versions of Windows, Visual Studio, and Intel Fortran for HDF4 and HDF5. We were able to avoid many of the traps of Windows maintenance by installing only the tools necessary for HDF, with each configuration separated in a new virtual machine. We used virtualization to extend our Windows support and also

to introduce a new platform for automated testing of HDF, while simultaneously decreasing the overall maintenance burden of Windows support.

2 Virtualization Overview

Virtualization is a technology that allows a complete operating system to be emulated inside of another computer. A single *host* computer can have multiple virtual machine *guests* running inside of it, each with a different operating system and environment. Each guest is a complete machine with a set of virtualized hardware, i.e. memory, processor, hard disk. However, the virtual guest machine operates under the assumption that it is a real machine, and is thus useful for staging common scenarios.

2.1 Virtualization Software

Virtual machines require special software to run. The software is responsible for managing the creation and maintenance of virtual machines, as well as mapping the virtual hardware to the underlying physical host. Many choices for virtualization software exist, and we've found they fall into three general tiers: hobbyist, desktop, and server -grade.

In the first category are a variety of free solutions for virtualization, such as Sun VirtualBox and Microsoft VirtualPC. These solutions provide all the necessary features you would expect getting started with virtualization. VirtualPC supports 32-bit Windows XP and Vista, and can run an unlimited number of virtual machines concurrently, limited only by the host's physical hardware. It also supports technology called paravirtualization, which allows virtual machines to communicate direction with the host machine's physical processor, if the hardware supports it, to increase performance. VirtualPC has a very simple and easy-to-use interface with support for basic virtual machine snapshots.

At the next level of support are a series of more robust desktop-grade virtualization products, most notably VMWare Workstation. Workstation includes all the features of the free products listed and expands upon them as well. It can run any 32- or 64-bit virtual operating system including flavors of Windows and Linux. It also expands on the snapshot feature and allows users to create full trees of snapshot configurations. Users can clone a virtual machine at any configuration to make a new copy of the virtual machine. VMWare Workstation also has many performance optimizations that make it faster and more efficient than the free products.

Even more powerful are the server-grade virtualization products on the market, dominated by VMWare ESX Server. These products are targeted at users seeking to virtualize server infrastructure. It is installed as a full hypervisor, meaning no underlying operating system needs to run on the host. It has many performance features allowing it to load-balance virtual hardware resources across multiple physical host machines. Virtual machine maintenance is also simplified.

For our project, we chose VMWare Workstation. We require some features unavailable in VirtualPC, such as 64-bit support. But we didn't feel that a server-grade solution was necessary. VMWare is the industry leader for virtualization products, and their Workstation product is well supported and continually being improved, now at version 6.5. Our virtualization setup runs on a new server machine purchased in order to host multiple virtual machines simultaneously without encumbering any existing server tasks. When customizing the server, we chose a high end processor along with

very fast disk and RAM. The server's physical hardware dictates the virtual hardware and overall effectiveness of every virtual machine. Our experience with and analysis of virtualization technologies are based on this setup: VMWare Workstation 6.5 running on a dedicated, high-end Windows server.

3 Virtualization Features

Virtualization works well for our project because it offers a new experience for creating and utilizing customized development environments. The specific features of VMWare Workstation were individually helpful for us to complete common tasks, and together the virtual machines' features were beneficial in easing the overall experience of working between many virtual machines.

3.1 Snapshot Trees and Cloning

VMWare Workstation has the ability to take *snapshots* of a virtual machine at any point in time to save its current state. Then, a user can always go back to that snapshot to revert any subsequent changes and return to the exact previous state. Moreover, a user can create a "trail' of snapshots progressing through time and creating a configuration history for the virtual machine. By reverting to a previous snapshot and creating new snapshots from there, the virtual machine effectively takes on a new branch of configuration. This can be extended to create entire snapshot trees. Each snapshot can be replayed in the same machine, or the snapshot can be cloned into a new virtual machine.

Snapshots and cloning were particularly useful for us in the initial creation of virtual machines. We were able to create a separate virtual machine for each version of Windows we support, and then used snapshots to incrementally install the development tools needed to build in different configurations. For example, one snapshot captures the install of Visual Studio 2005, and multiple branches extend to the various versions of Intel Fortran supported. In the end we had a robust series of snapshot configurations representing every supported build environment, and each could be cloned into new virtual machines for any number of support tasks.

3.2 Virtual Machine Integration

In each new virtual machine, a set of virtual hardware is set up to allow seamless integration with the host operating system. Users interact with virtual machines with the same keyboard and mouse in a windowed or full-screen representation of the virtual machine--very similar to operating on a Windows machine through a remote connection. Low-level devices such as memory, processor and disk are abstracted by VMWare so that the user doesn't need to be concerned with them at all.

The smooth and familiar transition to virtual machines has allowed us to rapidly improve our Windows support without being burdened with the learning curve generally associated with new technologies. Developers benefit by having a clean working environment any time they need to work on a new Windows task. Each virtual machine is configured with the same set of standard development tools, making it easy for developers to switch between virtual machines as necessary. It used to be a burden for developers to track down configuration-specific bugs, and even impossible when we couldn't replicate the user's environment. Now we have every supported Windows configuration available to us and can easily tweak our maintenance environment to match a user's. This is all possible without sacrificing the developer's production environment, because it is very easy to revert a virtual machine back to its clean production state. Our virtual machine setup has been so successful that our current Windows maintainer does all Windows work inside of virtual machines.

3.3 VMWare Scripting Interface

VMWare also distributes libraries and APIs for its virtual machine scripting interface. Nearly all virtual machine tasks can be accomplished through this interface. Management tasks can be automated such as powering on or off, taking a new snapshot, or checking the virtual hardware of a virtual machine. Integration tasks can be scripted as well, including checking environment variables in a virtual machine or running a program or script inside a virtual machine. The scripting capabilities are available in many popular programming languages, including C++, C#, Python, Bash and Shell.

Windows HDF daily tests use the scripting interface to run HDF tests on virtual machines automatically every night. Each configured virtual machine--representing our full line of Windows support configurations--is reverted to a clean state, powered on, tested, and then powered off. Test results are emailed out to a Windows developer mailing list to be checked in the morning. If a test fails on the virtual machine, the email results will pinpoint the location of the failure, and developers can boot up the virtual machine to the exact failure point to investigate.

4 Virtualization Analysis

Though the virtualization project has been decidedly successful, there are aspects of virtualization technologies which could use improvement or some alternative process.

4.1 VM Maintenance

Maintenance tasks on the virtual machines are equally as important as maintenance on physical machines, and cannot be neglected. They become cumbersome because there are far more virtual machines than we would have physical machines. The setup process for new platforms is rigid and tedious; it's important to ensure a clean configuration early on because many machines will generally be cloned from a single snapshot. Most virtual hardware decisions are final, and these decisions will also propagate to many virtual machines. Also, system patches and maintenance that is performed late in configuration won't be available to previous snapshots, so effort is duplicated if the machine ever gets cloned again from the earlier snapshot. Overall, it was important to be very cautious in our initial set up to minimize the maintenance burden of the virtual machines themselves.

4.2 Performance

We learned a good deal about the performance of virtual machines through our experimentation. In most situations, the performance of virtual machines was comparable to the performance that could be expected of the host machine. However, running multiple virtual machines simultaneously allowed us to more efficiently take advantage of resources that were idle with only one machine. Our performance analysis was taken while the HDF5 automatic test script was being run.

In general, disk access was the primary bottleneck of performance. This is expected, given the data-centric nature of compilation and testing the HDF5 library. Performance tuning was thus focused on optimizing disk access on the virtual machines. We found that running too many virtual machines at once would cause thrashing for disk and memory resources, and thus degrade performance. We found for our tests that the optimal number of virtual machines to test at once was three.

Disk fragmentation was also a large problem in our early tests, because it happens at three different levels: fragmentation of the virtual machine’s virtual disk, fragmentation of the virtual disk as it is

stored on the host machine, and the host machine's physical disk. These three levels of fragmentation can significantly and unnecessarily degrade disk performance. Thus an important aspect of virtual machine maintenance was to make sure the host and virtual machines were properly defragmented at each snapshot.

VMWare also touts a feature called *page trimming* in which identical page files between virtual machines can be shared to decrease storage and increase performance. The expected gains from page trimming were cited as 5% - 30%, although we only experienced a 5% improvement in our tests.

5 Conclusion

We have had great success with virtualization applied to the HDF Windows support effort, and would recommend the technology to others also considering virtual machines. Through the project, we have compiled this log of experiences and tips for working with virtual machines and virtualization software, for going forward with virtualization in HDF, and for those with similar projects.

VMWare Workstation 6.5 worked very well for us, and contains all the features we expected as we got started. We would also consider looking into VMWare ESX server, which is even more robust and increases performance.

Virtualization has been very successful for Windows maintenance and we feel that it could be easily expanded to benefit others as well. Once virtual machines are configured, any developer could make a clone and easily have their own preconfigured Windows development environment. Virtual machines are also portable to new hardware, so they can be stored on a laptop or workstation at the developer's convenience. Virtualization also represents a viable test-bed for new platforms or compilers. They could be installed easily in a virtual machine without any new hardware cost, and would find the same benefits that we have experienced in our Windows machines. Automatic testing could also be expanded to new platforms on the virtual machines. This could alleviate the maintenance complications of having machines online for both testing and development. Virtual machine clones would exist strictly for testing purposes, and can be integrated directly into the existing Windows testing framework.