Short Paper: cvfHUB: A Gateway Supporting Microsoft Windows in Engineering Workflows

David Benham
HUBzero
Purdue University
West Lafayette, Indiana 47907
Email: dbenham@purdue.edu

Michael Zentner
HUBzero
Purdue University
West Lafayette, Indiana 47907
Email: mzentner@purdue.edu

***Abstract*—The HUBzero® platform enables scientists to build off one another's research by providing a robust collaborative platform to allow them to share the software they created and utilized for their work. Customized tools, code, and data were no longer forgotten after a research effort was complete. HUBzero enabled researchers to offer their tools for anyone to use and modify so they could continue to collaborate long after the initial project concluded.**

Early versions of HUBzero focused on providing researchers a Linux based environment to host their software. Our group realized there was growing interest in HUBzero hosting commercial, licensed software, which often requires Microsoft Windows, so we began to investigate how we could integrate Microsoft Windows into HUBzero. A key partner, the Institute for Advanced Composites Manufacturing Innovation (IACMI) joined the effort. In this paper, we discuss the approach of our solution and associated learnings.

I. Introduction

nanoHUB [1] pioneered scientific cloud computing beginning in 1996. Since its inception, nanoHUB has undergone several evolutions in the manner in which it delivers scientific simulation tools over the web. Ultimately this resulted in having developed a Linux solution from the ground up to host simulation tools and allow users to easily put graphical interfaces on those tools [2] [3]. The solution for Linux was a PHP driven system that supported remote rendering of a Linux desktop in a web browser using VNC. In its first release, this involved a Java applet running in the web browser. As technologies continued to evolve, Java encountered industry-wide security concerns and became less supported by browser creators, and more importantly, restricted by many of the organizations to which our users belong. A second iteration of the technology was released that utilizes HTML5 rather than the Java client, circumventing these limitations.

During this same time of evolution, the infrastructure that supports nanoHUB was also spun out as a separate entity called HUBzero with the intent of supporting many other science gateways with the same infrastructure that drove nanoHUB. As more scientific communities adopted HUBzero, the support team for HUBzero evolved the infrastructure to adapt to some of the unique needs of those communities. One community in particular that has become a significant HUBzero user is in the area of virtual composite materials design and manufacture. This community is further from the basic science and closer to the application of materials in the commercial marketplace than many other HUBzero gateways. Therefore, a significant requirement of this community was to host Microsoft Windows based tools, both academically and commercially developed.

To satisfy this need, the HUBzero team considered applying our own expertise to extend our existing platform into the Windows domain. However, hosting Windows applications and development workspaces in the cloud is significantly different than its Linux counterpart. The highly customized approach taken with our Linux based system was not a viable approach as we ventured into the world of Microsoft Windows because the operating system and many tools were not open source. We began to investigate commercial off the shelf software.

Next we will describe the nature of the engineering workflows in cvfHUB and motivation for the solution, several technologies tried in fulfilling this need, and the lessons learned in providing this capability.

II. Composite Virtual Factory HUB (cvfHUB)

The gateway sponsored by IACMI, cvfHUB, is focused on applying materials design principles to real world applications in the automotive industry. Specifically, the goal of cvfHUB is to allow design of materials virtually, before ever manufacturing them physically. This capability will allow thousands of possibilities to be tested virtually before investing significant funds into manufacturing and physically testing only the most promising options. Typically these tasks involve multi-objective optimization in the face of constraints such as optimizing the weight of a given part but maintaining durability and performance criteria within certain bounds expected to be encountered in deployment. The workflows involved in such part optimizations can include basic composite material design, analysis of how the material will perform during molding and forming processes, cutting processes, curing processes, finishing processes, and under expected operating conditions. Such an analysis involves the combination of academic and commercial codes, and a significant amount of translation code in between that interprets outputs of one tool and reformation of those outputs into inputs that a subsequent tool can accept. Once constructed, such a workflow may be executed hundreds or thousands of times during the design of

a given part to perform the optimization. cvfHUB is intended to host these workflows, along with collaboration tools for distributed engineering teams as they undergo design activities.

III. Solutions for Hosting Windows

Three different technologies were evaluated and used in providing Windows support: Amazon AppStream, Amazon AppStream v2, and Citrix. Below these three are contrasted with their respective capabilities and matched against the requirements of cvfHUB. The primary requirements cvfHUB are: 1) Intuitive integration of the remote environment with local resources, 2) Ability to deliver accelerated graphics to the client, 3) Flexible filesharing mechanisms available to end users, and 4) ability to integrate the solution into the HUBzero tool.

A. Amazon AppStream

The first technology offering Windows tools inside the HUBzero platform involved leveraging Amazon's AppStream service. The AppStream service was originally designed as a cloud service that offered video game developers a way to offer online game demos without requiring their prospective customers to download and install large pieces of software on local clients. Utilizing AppStream, software could be pre installed and run in on demand virtual machines running on the Amazon Web Services platform.

AWS AppStream integration was completed in a very short timeframe, and it initially appeared to offer an excellent user experience. The largest concern with HUBzero's Windows integration in the early stages was to develop as seamless an experience as possible for end users, and the need for a highly responsive UI and video streaming support were critical. Other benefits of AppStream included on-demand environments that greatly helped reduce costs over building infrastructure, as well as a robust web browser plug-in for client side display of application streams to end users.

However, as users began to work with this solution, it turned out AppStream's lack of support for shared file systems, and the level of integration with the local client did not allow sharing clipboard text or files from local filesystems. These limitations proved extremely difficult to overcome, and the AppStream virtual environments themselves did not allow users to customize the environment. Further, sessions running AppStream could not persist beyond the browser's session, making them impractical for long running tools. Finally, AppStream was more suited to running individual tools than it was running a full development workspace in Windows. Therefore, it remained an effective solution for delivering simple Windows tools, but clearly more powerful capabilities were needed.

B. Amazon AppStream version 2

During the integration of AppStream, the software engineers at AWS helped us address some of our early concerns. AWS engineers informed us AppStream v2 was in development. We were able to get early access to some of the design documentation. The new feature sets they were proposing for the next version of AppStream appeared to address some of our concerns, so we contemplated using the next version of AppStream. AppStream v2 added new features that allowed users more flexibility in contructing base images for the platform, options that gave users the ability to specify autoscaling criteria for their applications, and features to gave users the ability to offer different tiers of instance support.

However, the AWS engineering team could not provide a firm date on the release of AppStream version 2, and a tight deadline on our Windows integration effort required us to abandon investigating this possibility further. There was also no indication Appstream v2 addressed any of our client integration requirements for local file access. We do plan on revisiting AppStream version 2 and evaluating it for incorporation into HUBzero at a later time.

C. Citrix

Citrix is a market leader in providing remote access to Microsoft Windows software. Other groups at Purdue University were using Citrix, and early discussions with them indicated we had some in-house experience with Citrix technologies we could potentially leverage in our integration as we designed a way to offer Microsoft Windows applications to our end users. Citrix based solutions appeared to easily meet all of our design criteria, however, we hesitated pursuing this solution at first due to the extreme learning curve required to implement and manage a Citrix infrastructure, as well as the additional costs in doing so.

As we continued to evaluate Citrix's XenApp and XenDestkop, the level of client interoperability became a clear advantage. Citrix had spent the last several decades designing utilities to make remote application access client tools seamlessly integrate with all major operating systems of their end users. Citrix also offered a technology called HDX which provides a highly optimized and adaptive system that allows Citrix to closely emulate the level of performance users require in graphics intensive applications running on local hardware. When our end users began to see early versions of our Citrix based solution, it became clear this was the direction our team should pursue.

After meeting with Citrix engineers, we learned Citrix offered a cloud based solution in which they would host the bulk of the administrative infrastructure required to deploy their products. Their cloud solution could be configured to deploy and manage our Virtual Desktop Infrastructure (VDI) in AWS and had the flexibility to utilize a hybrid model that could deploy resources in a variety of other local and cloud based scenarios.

IV. Lessons Learned

A. Culture, Attitudes, and Skill Sets

Windows based software presented some unique challenges to our team. First, we had to embrace commercial off the shelf hardware due to the limited nature of open source software in the Windows environment. Many HUBzero components

heavily integrated with open source tools, and the lack of open source tools in many Windows environments forced our team to rethink how we approached our development. If a piece of software did not do what we needed, we could not fork the software and add our missing feature, we had to look at solving problems from different angles.

Our Linux tool sharing platform concentrated on development and collaboration in constructing applications. We found our Windows users were less concerned with developing the software themselves and they often preferred to use commercially available software. This realization helped us strip out some features that were largely unneeded in this particular environment, such as source code management, automatic application deployment, and tool version tracking.

We also had to overcome our staff's limited Microsoft Windows infrastructure experience. Over the lifetime of HUBzero, we built a staff based on Linux skillsets. Shifting to Windows proved difficult, not only were we missing skills, but we often dealt with staff that even had ideological differences with Microsoft Windows itself.

B. Client Side Considerations

Client interaction over the internet with remote environments can be challenging. This was nothing new to us, we struggled with the client interaction even in our Linux based system. However, the importance of this with our Windows users was even more pronounced. Missing features such as clipboard cut and paste meant a great deal of lost productivity to our users. Local drive access was a critical missing feature from our early efforts. We overcame these limitations in a variety of ways. As we expanded our user base, a growing number of users found these missing features more and more important.

C. Licensing

Embracing Windows based software also meant we needed to deal with the requirements of non open source software licensing models. Windows itself requires purchase of a license, and Windows based softare is often similarly licensed. Having built our system on Free and Open Source Software (FOSS), we would often need to ensure licensing allowed for redistribution and modification, which is generally allowed in most FOSS based licenses, but our concerns rarely went beyond this. With closed source software, proprietary software licensing methods involve a license purchase, and often the software employs a license enforcement mechanism that ranges from one time activation keys, to the use of specialized license servers to tighly control distribution and use.

Licensing presented some technical challenges, most of which were easily overcome with iptables based based port forwarding, though other options were considered, such as use of the socat tool. This approach gave us the flexibility to consolidate licensing into a single proxy that managed the license requests for a variety of different applications in one place. The primary challenges we faced in dealing with licenses often focused legal discussions with vendors. The specifics of these issues are beyond the scope of this paper, but efforts often required significant time to overcome. Some vendors we dealt with did not have licensing agreements that allowed for cloud based or shared desktop environments, and work here greatly affected our overall timelines.

V. Conclusions

Years of developing technology on Linux based platforms taught our entire team to look at problems from that perspective. Even many of our users were proficient Linux users, so the solutions we devise often assume a lot about our environment and our users. As we began to look at integrating Microsoft Windows into HUBzero, we realized we needed to revisit they way we approached designing and developing our system, both technologically and culturally.

While the core of our system will remain Linux based, we can no longer construct our software under the assumption that our platform will work in a homogenous environment. A key component of HUBzero has been collaboration, but we realized there is a new dimension of collaboration emerging. Early versions of HUBzero focused on collaboration in the development of applications. However, today we see an increased emphasis on collaboration in the application of the tools and the workflow that allows multiple tools to work together to solve complex problems. As we continue our efforts to support Microsoft Windows, we hope to unify applications from both the Linux and Windows environments to help researchers utilize and collaborate seamlessly with entire suites of interrelated tools tied together to automate scientific research process flow.

References

[1] Kapadia, Nirav H., Mark S. Lundstrom, and Jose AB Fortes. *A network-based simulation laboratory for collaborative research and technology transfer.* Semiconductor Research Corporation's TECHCON 96 (1996).

[2] Lundstrom, Mark, and Gerhard Klimeck. *The NCN: science, simulation, and cyber services.* Emerging Technologies-Nanoelectronics, 2006 IEEE Conference on. IEEE, 2006.

[3] M. McLennan, *The Rappture Toolkit*, http://rappture.org (2004)