This paper is under review by the May 2014 open challenge of the National Academy of Sciences Board on Research Data and Information

Olive: Sustaining Executable Content Over Decades

Mahadev Satyanarayanan[†], Gloriana St. Clair[‡],
Benjamin Gilbert[†], Jan Harkes[†], Dan Ryan[‡], Erika Linke[‡], Keith Webster[‡]

[†]School of Computer Science and [‡]University Libraries
Carnegie Mellon University

ABSTRACT

We describe a system called *Olive* that freezes and precisely reproduces the environment necessary to execute software long after its creation. It uses virtual machine (VM) technology to encapsulate legacy software, complete with all its software dependencies. This legacy world can be completely closed-source: there is no requirement for availability of source code, nor a requirement for recompilation or relinking. The entire VM is streamed over the Internet from a web server, much as video is streamed today.

1. Software in Science

Reproducibility is at the heart of the scientific method. Confidence in a result grows as researchers all over the world are able to reproduce it independently. Today, an increasing fraction of the world's intellectual output is in the form of *executable content* — i.e., software. This is true in virtually all areas of scholarship, from physics, chemistry, biology, and engineering to economics, political science and the humanities. Examples of such executable content include data analysis tools to slice and dice raw data, zoomable visualization tools that enable results to be viewed at many levels of abstraction, and simulation models written in a variety of programming languages and using a wide range of supporting libraries and reference data sets. Such software is central, not peripheral, to the discovery of new results today. Raw scientific data is often of limited value unless it is accompanied by the uniquely customized software that was created to decode, interpret, analyze and display that data.

The role of software in the scientific method is illustrated by a recent controversy [6]. In early 2010, Reinhart and Rogoff published an analysis of economic data spanning many countries [8, 9]. Herndon et al [4] refuted their findings in 2013 by discovering an error in their calculations. The significance of the error was described as follows [7]:

> "The Reinhart-Rogoff research is best known for its result that, across a broad range of countries and historical periods, economic growth declines dramatically when a country's level of public debt exceeds 90 per cent of gross domestic product.
> ...
> When we performed accurate recalculations using their dataset, we found that, when countries' debt-to-GDP ratio exceeds 90 per cent, average growth is 2.2 per cent, not -0.1 per cent."

The controversy continues, but regardless of how it is eventually resolved, there is no denying the central role of software (in this case, a Microsoft Excel spreadsheet) in the original analysis, its refutation and its eventual resolution.

2. The Ravages of Time

In the Reinhart-Rogoff example, there was no difficulty in obtaining the software necessary to perform the recalculations. Only three years had elapsed since the original publication of results, and the same version of Microsoft Excel continued to be in widespread use. Imagine, however, that the recalculations were attempted by a researcher 30 years later. Would Microsoft Excel still be in use? If so, would the version then in use accept the data format used by the original researchers? Would the calculations performed by that version be identical in every respect (including, for example, handling of rounding errors) to the version used by the original researchers? What if Microsoft goes out of business ten years after the original publication of results, and the Windows environment (which is needed to run Excel) ceases to be in use? As these questions suggest, our growing dependence on software in scientific research introduces new challenges to the premise of reproducibility that is the bedrock of science. Unless these challenges are addressed, our ability to re-validate published results will evaporate over time.

In this paper, we describe a system called *Olive* that seeks to freeze and precisely reproduce the environment necessary to execute software long after its creation (possibly many decades later). It uses virtual machine (VM) technology to encapsulate legacy software, complete with all its software dependencies. This includes the operating system, dynamically linked libraries, tool chains, configuration files, data files and other supporting items. This legacy world can be completely closed-source: there is no requirement for availability of source code, nor a requirement for recompilation or relinking. The entire VM is streamed over the Internet from a web server, much as video is streamed today. One-click execution of pre-packaged legacy software from a web site thus becomes possible. The rest of this paper examines the challenges addressed by Olive, and then describes its design, implementation and current status.

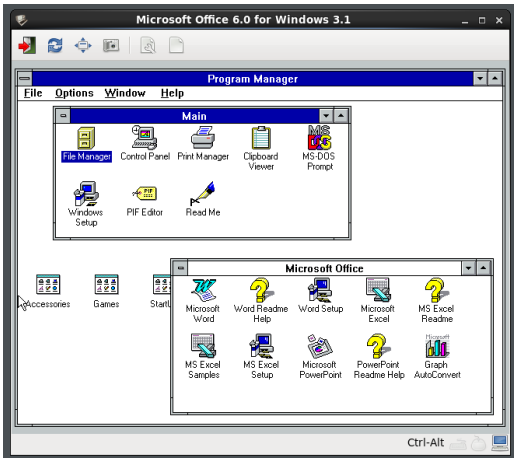

Figure 1: Microsoft Office 6.0 on Windows 3.1

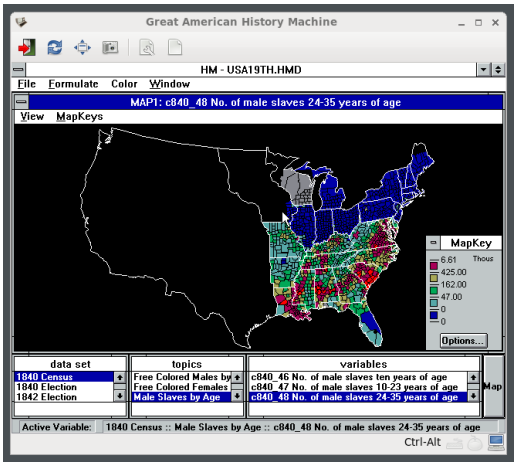

Figure 2: Great American History Machine on Windows 3.1

3. Execution Fidelity

Precise reproduction of software execution, which we call *execution fidelity,* is a complex problem in which many moving parts must all be perfectly aligned for a solution. Preserving this alignment over space and time is difficult. Many things can change: the hardware, the operating system, dynamically linked libraries, configuration and user preference specifications, geographic location, execution timing, and so on. Even a single change may hurt fidelity or completely break execution.

Unfortunately, the available mechanisms for enforcing execution fidelity are weak. Most software distribution today takes the form of *install packages,* typically in binary form but sometimes in source form. The act of installing a package involves checking for a wide range of dependencies, discovering missing components, and ensuring that the transitive closure of dependencies involving these components is addressed. Tools have been developed to simplify and partially automate these steps. However, the process still involves considerable skill and knowledge, remains failure-prone, and typically involves substantial time and effort.

These difficulties loom large to any researcher who attempts to re-validate old scientific results. Software install packages themselves are static content, and can be archived in a digital library using the same mechanisms that are used to archive scientific data. However, the chances of successfully installing and executing this software in the distant future are low. In addition to all of the software installation challenges mentioned above, there is the additional difficulty that the passage of time makes hardware and software environments obsolete. The chances of finding compatible hardware and operating system on which to even attempt an install become vanishingly small over time scales of decades. These challenges have long stymied efforts to archive executable content [2, 3, 5].

4. Virtual Machines

Olive leverages VM technology to encapsulate and deliver a bit-exact, pre-packaged execution environment. The VM abstraction is implemented by a *virtual machine monitor (VMM).* This is a computer architecture and instruction set emulator of such high accuracy and transparency that neither an application nor the operating system is able to detect its presence. In other words, emulated execution is indistinguishable from execution on genuine hardware. VMs have a venerable history, dating back to the late 1960s. In the past decade, the emergence of cloud computing has spawned tremendous activity and investment in advancing the VM abstraction. Olive benefits indirectly from the many efforts in academia and industry that are aimed at improving the performance and functionality of VM-based systems. In Olive, VMs are transparently streamed from servers to execution sites over the Internet. The archived software within the VM executes without any awareness of the streaming process.

5. Olive Status

Olive contains over 15 VMs today, including operating systems and applications dating back to the late 1980s. The collection continues to grow. For brevity, we describe only four of these VMs below.

Microsoft Office 6.0

Figure 1 shows a screenshot of this VM, containing Word, Excel and PowerPoint for Windows 3.1. If Reinhart and Rogoff had published their controversial paper [8] in the 1993-94 timeframe, this is the VM that you would need to re-validate their results today.

Great American History Machine

This application was originally created in the late 1980s by Professor David Miller of Carnegie Mellon University. It was used by him and by many professors at other universities nationwide to teach 19th century and early 20th century

Figure 3: TurboTax 1997 on Windows 3.1

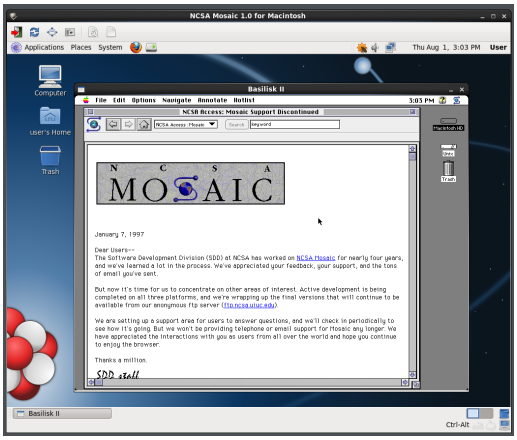

Figure 4: Mosaic Browser on MacOS 7.5

American history. As the screenshot in Figure 2 shows, this educational software used census and election data to teach students important historical concepts such as the origins of the Civil War. The Windows 3.1 version of this software was created in collaboration with the University of Maryland. Because of lack of financial resources to port the software to newer Windows platforms, it fell into disuse over time. No modern equivalent of this software exists today.

TurboTax 1997

This application for Windows 3.1 and Windows 95 was used by millions of Americans to prepare their 1997 tax returns. Figure 3 shows a screenshot of this application. Since TurboTax is updated each year to reflect the current tax laws, a suite of TurboTax VMs from consecutive years can offer unique historical value. Imagine a class in political science, public policy or economics assigning students a project based on TurboTax versions that are ten years apart. By calculating the tax returns for hypothetical families with different sources and amounts of income, students can see for themselves the impact of tax code changes over time. Such active learning can transform the abstract topic of tax law into a source of valuable real-world insights.

NCSA Mosaic

As the world's first widely-used web browser dating back to 1992-93, Mosaic has a unique historical status. This VM, whose screenshot is shown in Figure 4, is also interesting for a second reason. The version of Mosaic that it encapsulates was written for the Apple MacOS 7.5 operating system on Motorola 68040 hardware. The VM also encapsulates Basilisk II, an open source hardware emulator for Motorola 68040 on modern Intel x86 hardware running Linux. The bootable disk image of MacOS 7.5 with Mosaic is stored as a file in the virtual file system of the outer Linux guest. In spite of two levels of virtualization, performance is acceptable because modern hardware is so much faster than the original Apple hardware. Pointing the Mosaic browser at modern web sites is instructive. Since Mosaic predates web technologies such as JavaScript, HTTP 1.1, Cascading Style Sheets, and HTML5 it is unable to render content from modern web sites. It is, however, capable of rendering web pages from some older sites that are still on the Internet.

6. Olive Implementation

Figure 5 illustrates the conceptual structure of an Olive client. At the bottom (1 and 2) is standard Intel x86 desktop or laptop hardware running Linux (generically called the "host operating system"). Layered above this (3) is an Olive component called *VMNetX* that implements caching and prefetching of VM images over the Internet. VMNetX presents the illusion of a fully assembled VM image to the VMM layer above, which virtualizes the x86 host hardware. We use KVM/QEMU as our VMM. Layers 5 through 8 in Figure 5 are encapsulated within the archival VM image that is streamed from Olive servers. The lowest of these layers (5) is a hardware emulator that presents the illusion of now-obsolete hardware (such as Motorola 68040). This layer can be omitted if the archived environment targets x86 hardware. Layer 6 is the archived operating system (generically called the "guest" operating system). The virtual disk of the VM is managed by the guest operating system, and appears as a local file system to higher layers.

Layer 7, which represents the archived application (such as the Great American History Machine) is the focal point of interest in archiving. It is to support execution of this application with high fidelity that the entire edifice shown in Figure 5 is necessary. Layer 8 represents input that is provided to the archived application. In the Reinhart-Rogoff example, Layer 8 would be the original Excel spreadsheet that was used in their analysis. Layer 7 would be the version of Excel that they used. In a different situation, such as examining an old archived engineering drawing, Layer 7 might be the AutoCAD application and Layer 8 would be the input files to AutoCAD that represent the drawing. Alternatively, Layer 8 may be placed on an external data source such as a distributed file system and exposed to the guest OS as a virtual floppy disk or virtual CD-ROM.

Figure 6 shows how the abstract layers shown in Figure 5

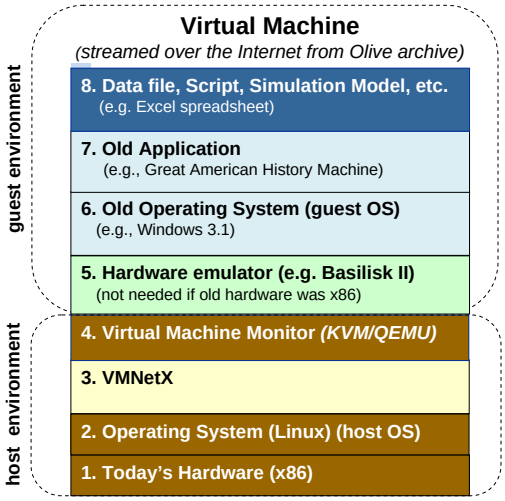

Figure 5: Abstract Olive Client Structure

are mapped to the Olive architecture. Layers 8 through 5 are encapsulated within the VM instance shown on the left. Layer 4 (KVM/QEMU) is explicitly shown in Figure 6. Layer 3 (VMNetX) maps to the user-level process and file caches ("pristine" and "modified"). As the VM instance executes, it may access parts of its VM image that have not been cached yet. VMNetX services these cache misses using HTTP range requests to a standard Web server such as Apache. The "web page" in this case is a large file on the server that contains all components of the VM image, including its disk image, its memory image, and its hardware configuration.

To support non-Linux clients, the entire client structure shown in Figure 6 can be executed on a nearby cloudlet [10] or private cloud. The SPICE remote desktop protocol [1] is used for thin client user interactions with the VM instance. Low-latency, high-bandwidth network connectivity between the user and the VM instance is necessary for thin clients to provide a good user experience.

7. Conclusion

Executable content ranging from simulation models to visualization tools plays an increasingly important role in scientific research. The ability to archive these artifacts for posterity would be an important transformative step. Imagine being able to reach back across time to execute the simulation model of a long-dead scientist on new data that you have just acquired. What do the results suggest? Would they have changed the conclusions of that scientist? Although you aren't quite bringing the scientist back to life, you are collaborating with that person in a way that was not possible until now. Olive is the first system to provide this new capability, which we predict will become a *sine qua non* for the scientific method in the 21st century and beyond.

Availability

The Olive web site is at `http://olivearchive.org`. Due to software licensing restrictions outside our control, the VMs on the web site are currently accessible only to our research collaborators. We hope to lift this restriction in the future. VMNetX is open-source software available under the GPLv2 license.

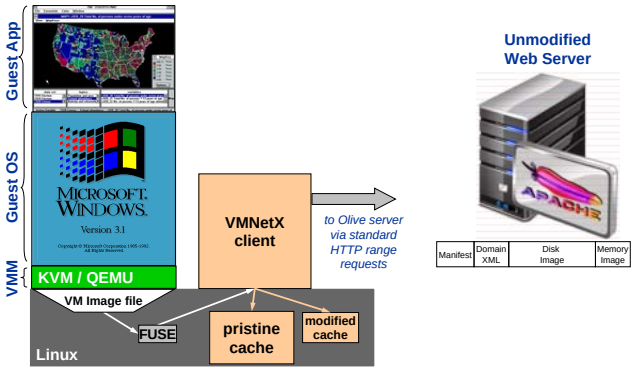

Figure 6: Olive Architecture

Acknowledgements

Vas Bala and his colleagues at IBM collaborated with us on the early research that led to the Olive vision. We wish to thank IBM for its support and early advocacy of Olive. This work was supported by the Alfred P. Sloan Foundation and the Institute of Museum and Library Sciences. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and should not be attributed to Carnegie Mellon University or the funding sources.

8. REFERENCES

[1] Spice, 2014. `http://www.spice-space.org/`.
[2] P. Conway. Preservation in the Digital World. `http://www.clir.org/pubs/reports/conway2/`, March 1996.
[3] P. Conway. Preservation in the Age of Google: Digitization, Digital Preservation, and Dilemmas. *Library Quarterly*, 80(1), 2010.
[4] T. Herndon, M. Ash, and R. Pollin. Does High Public Debt Stifle Economic Growth? A Critique of Reinhart and Rogoff. Working Paper 322, Political Economy Research Institute, University of Massachussets Amherst, April 2013. `http://www.peri.umass.edu/236/hash/31e2ff374b6377b2ddec04deaa6388b1/publication/566/`.
[5] B. Matthews, A. Shaon, J. Bicarreguil, and C. Jones. A Framework for Software Preservation. *The International Journal of Digital Curation*, 5(1), June 2010.
[6] P. Monaghan. 'They Said at First That They Hadn't Made a Spreadsheet Error, When They Had'. *The Chronicle of Higher Education*, April 2013. `https://chronicle.com/article/UMass-Graduate-Student-Talks/138763/`.
[7] R. Pollin and M. Ash. Austerity after Reinhart and Rogoff. *Financial Times*, April 2013. `http://www.ft.com/cms/s/0/9e5107f8-a75c-11e2-9fbe-00144feabdc0.html#axzz2zLV7HuwS`.
[8] C. M. Reinhart and K. S. Rogoff. Growth in a Time of Debt. *American Economic Review*, 100(2):573–78, May 2010.
[9] C. M. Reinhart and K. S. Rogoff. Growth in a Time of Debt. Working Paper 15639, National Bureau of Economic Research, January 2010. `http://www.nber.org/papers/w15639`.
[10] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Computing*, 8(4), October-December 2009.