1

Reproducibility, Virtual Appliances, and Cloud Computing

Bill Howe

University of Washington

CONTENTS

In many contexts, virtualization and cloud computing can mitigate the challenges of computational reproducibility without significant overhead to the researcher.

1.1 Introduction

Science in every discipline increasingly relies on computational and data-driven methods. Perhaps paradoxically, these *in silico* experiments are often *more* difficult to reproduce than traditional laboratory techniques. Software pipelines designed to acquire and process data have complex version-sensitive inter-dependencies, their interfaces are often complex and under-documented, and the datasets on which they operate are frequently too large to efficiently transport.

At the University of Washington eScience Institute, we are exploring the role of cloud computing in mitigating these challenges. A virtual machine can snapshot a researcher's entire working environment, including data, software, dependencies, notes, logs, scripts, and more. Snapshots of these images can be saved, hosted publicly, and cited in publications. This approach not only facilitates reproducibility, but incurs very little overhead for the researcher. Coupled with cloud computing, either commercial or taxpayer-funded [16], experimenters can avoid allocating local resources to host the virtual machine, large datasets and long-running computations can be managed efficiently, and costs can be partially shared between producer and consumer.

In many cases, the application of virtualization to support reproducible research does not require any significant change to the researchers' workflow: The same experiments can be conducted in the virtual environment as in the physical environment, using the same code, data, and environment. When the experiments are complete, the experimenter will save a snapshot of the virtual machine, make it publicly available, and cite it in all appropriate papers. Readers of the paper who wish to reproduce the results can launch their own instance of the author's virtual appliance, incurring no additional cost to the author (and, we will argue, only minimal cost to the reproducer), and re-execute the experiments. Further, new experiments, modifications, and extensions implemented by the reproducer can be saved in a new virtual machine image and re-shared as desired. The consequences and benefits of this basic model, as well as discussion of adaptations to support more complicated reproducibility scenarios, is the subject of this chapter. We provide examples from the literature of how this approach can significantly improve reproducibility with minimal additonal effort, and that this approach is largely complementary to other technologies and best practices designed to improve reproducibility. We will conclude with the remaining challenges to be overcome to fully realize this model.

1.2 Background on Cloud Computing and Virtualization

A virtual machine provides a complete interface to a physical computer in software. This interface allows a complete operating system and any other software to run within a managed "virtual" software environment without requiring any direct access to the underlying "host" computer. A virtual machine along with an operating system and other software can be stored, transported, and managed as a single file called an "image." The use of virtualization helps solve problems in a variety of areas technique can be applied to a huge number of problems: Software can be tested on N different platforms without having to purchase and maintain N different computers. In a data center, if a physical machine fails, the virtual machine can be migrated elsewhere, in some cases without interruption. Software developed for Windows can be used in a linux environment, without complicated dual-boot

scenarios. In the enterprise, IT departments can upgrade thousands of desktop environments by distributing a new virtual machine image nching new virtual machines rather than physically installing software on each machine individually. In this paper, we focus on using virtual machines to distribute software. This mechanism allows users to skip the installation step entirely — an attractive option for software with many complex dependencies.

Computing resources offered on-demand, elastically, over the Internet — cloud computing — affords even more use cases for virtualization. Amazon Web Services, for example, allows virtual machines to be shared among users and launched on Amazon's hardware — users rent not only the software but the hardware on which to run it. This model has been wildly successful, allowing customers to get out of the business of administering computing resources and focus entirely on their business or their research.

1.3 Related Approaches and Examples

Lincoln Stein cogently argued why cloud computing and virtualization will be transformative for genome informatics [32]:

> Cloud computing ... creates a new niche in the ecosystem for genome software developers to package their work in the form of virtual machines. For example, many genome annotation groups have developed pipelines for identifying and classifying genes and other functional elements. Although many of these pipelines are open source, packaging and distributing them for use by other groups has been challenging given their many software dependencies and site-specific configuration options. In a cloud computing environment these pipelines can be packaged into virtual machine images and stored in a way that lets anyone copy them, run them and customize them for their own needs, thus avoiding the software installation and configuration complexities.

Stein's argument focuses on the cloud's role for distributing *software* rather than distributing and reproducing specific experimental results, but the mechanisms are closely related. Dudley and Butte articulate the connection between cloud computing and reproducible research [11], and argue that the need for specialized, non-standad software motivates the need to share complete operating environments ("Whole System Snapshot Exchange") as opposed to packaged tools. Dudley and Butte consider many of the issues we discuss in this chapter: reproducibility in the context of large datasets and specialized computational environments and the economics of long-term preservation.

As Stein and others predicted, the use of virtual machines for the purposes of software dissemination is becoming commonplace in the life sciences [1, 17, 4, 9]. For example, the CloVR project provides a set of metagenomic analysis tools and a browser-based graphical dashboard for interacting with them [4]. Both the tools and the dashboard are distributed as a virtual machine, reducing installation effort for the client, eliminating the need for installation documentation by the providers, and trivially providing cross- platform support. Examples in other fields are emerging as well: The CernVM [31] simplifies the process to setup and run high-energy physics codes and now supports all experiments associated with the Large Hadron Collider [18], as well as many other experiments.

These applications of virtualization and cloud computing for tool-delivery can help improve reproducibility by encouraging standardization on particular tools. However, new experiments tend to require new software that has not yet been packaged into clean reusable components. In this chapter, we consider the role of VMs and cloud computing even in these

"early stage" software situations: *ad hoc* software to demonstrate experimental results as opposed to *engineered* software intended for long-term reuse.

In these situations, we can put current approaches to reproducibility into four categories: Researchers can simply post their *raw code and data* on the web and rely on the published paper for documentation. This approach requires very little investment from the researcher, although they still need to find a place to host their materials. The effort required by the reproducer is significant, however, and this approach is generally not seen as sufficient (although it is still arguably an improvement over common current practices, where the code is simply not made available at all.)

Beyond just posting the code and data, experiments and software can be equipped with *extensive documentation* for recreating the original experimental environment. This approach requires significant up-front effort by the experimenter, but can improve long-term reusability and will typical improve near-term reproducibility as well. Brown et al. emphasize the use of source control repositories and explicit documentation for reproducibility, and include instructions for reproducing results with every paper [7]. Their argument is that complete instructions for reproducing the environment can be more robust than providing a single instance of a working environment. However, reconstruction of the software environment typically requires a different set of skills than simply reproducing the experiments: installation of software using linux package managers, the use of version control software, the configuration of environment variables, and sometimes familiarity with relatively advanced linux tools such as *screen* [8]. As a result, this approach restricts reproducibility to those who have similar technical expertise to the authors. We conjecture that of those who wish to reproduce the results published in this way, some non-zero fraction will be unable or unwilling to do so due to the additional overhead of configuring the environment. But perhaps more significantly, Brown et al. are exemplars in providing thorough, clear, and accurate instructions for reproducing their results. For authors who are unable to justify the cost of this effort given that long-term reusability may or may not be one of their goals, virtualization offers a low-overhead "minimum bar" for reproducibility. Further, virtualization is entirely complementary to this documentation-oriented approach: a working example environment disseminated as a virtual machine provides a means of checking that you have followed the authors instructions properly, and provides redundancy in case the instructions are incomplete.

A third approach is to adopt some kind of *controlled environment* in which to conduct your experiment that simplifies the metadata capture, provenance, logging, and dissemination process. These environments may be scientific workflow systems [15, 36, 35, 38] or an augmented programming environments [10, 28]. But in each case, the environment restricts the language, programming style, or libraries to which the researcher has access. Projects with specialized needs (large datasets, combinations of languages and libraries) may not be able to tolerate these restrictions, or the cost of re-engineering the experiment to conform to the provided environment may make the value proposition unclear. We will discuss these solutions in more detail later in the chapter.

The fourth approach, described in this chapter, is to use virtual machines to capture and publish the code, data, and experimental environment.

Figure 1.1 illustrates how these four approaches compare in relative effort for the experimenter, those who wish to reproduce the experiments (left-hand plot), and those who wish to reuse and extend the experimental software (right-hand plot). Posting raw code and data requires little effort from the experimenter (lower half of each plot), but requires significant effort from both reproducers and extenders. Augmenting the code with documentation requires more up-front effort from the experimenter, and reproducers are required to re-establish the original environment from scratch, a task they may or may not possess the skills to do. However, this approach is critical to support those who wish to reuse and

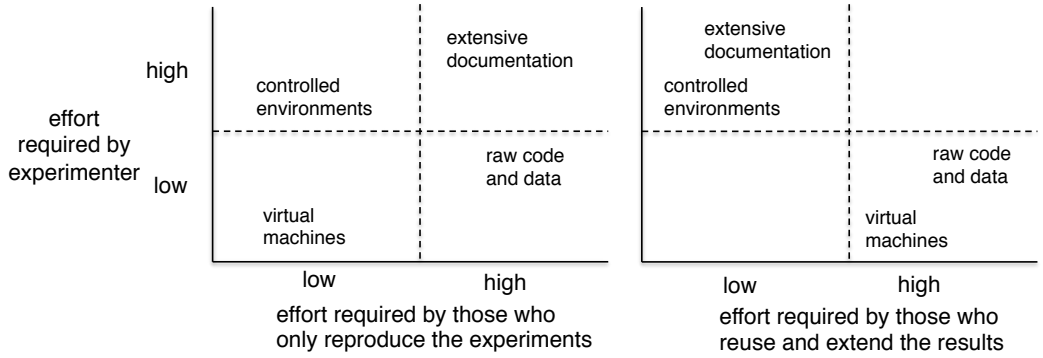

FIGURE 1.1
These four approaches to disseminating science software vary in the effort required by the original experimenter, those who wish to directly reproduce the results, and those who wish to reuse and extend the software for other purposes. Virtual machines (VMs) incure very little overhead for the original experimenter and support direct reproducibility, but are not sufficient for long-term extensibility. For extensibility, complete documentation is generally required, though some scientific workflow systems and other controlled environments offer a possible solution.

extend the software and adapt it for their own projects — there is no "shortcut" for software reuse. For the extenders, this documentation significantly reduces the effort required. Controlled environments also require some up-front effort, but can significantly reduce the effort required by both reproducers and extenders. Finally, virtual machines impose very little overhead on the experimenter, and direct reproducibility of results is straightforward, but an undocumented VM with all software pre-installed does very little to support long-term reusability and extensibility, perhaps offering only a small improvement over providing the raw code and data. These approaches are not mututally exclusive; releasing a VM demomstrating particular results along with complete documentation for the requisite software is an appropriate strategy [8].

1.3.1 Other Uses of Virtual Machines

Beside reproducibility, the creation and exchange of virtual machines has other benefits for scientific knowledge sharing. First, VMs are also increasingly used to distribute software for educational purposes. Sorin Mitran at the University of Washington uses virtual machines in classes ranging from non-technical first-year seminars to graduate classes to package the software environment for teaching scientific computing.[1] He finds that "using VMs allows a class to concentrate on the math and programming as opposed to installing all the utilities that come together to solve a problem." Second, VMs can be used to deliver custom prototypes and proofs of concept. Paradigm 4, the company that develops and distributes the SciDB database engine [33], routinely uses Amazon Machine Images for customer projects. They develop a prototype on behalf of a customer and deliver it as an AMI, allowing the customer to reproduce results by running the scripts developed by P4. This approach provides a "try before you buy" mechanism that allows customers to experiment with the system without investing IT resources to install the software locally.

[1] http://mitran.web.unc.edu/teaching/

Another facet of reproducibility exerised by the SciDB system is to adopt a "no overwrite" philosophy for operaton of the system — all results are derived from previous results, affording complete reproducibility and provenenance.

1.3.2 Towards Services Instead of Artifacts

A virtual machine avoids the need to install unfamiliar software on a potentially new platform, but still presumes that the reproducer can navigate your experimental environment and operate your code. Increasingly, we see bioinformatics tools exposed as a service, where users can interact with a web interface instead of the "raw" scripts and files. In the future, we can imagine publishing scripts as web-based interfaces directly, without goingthrough an engineering project to do so. This kind of capability is among the goals of the HubZero project [21].[2] Instances of the HubZero framework for specific domains, for example NanoHub in the area of nanotechnology, allows analysis routines to be uploaded to a server, attached to simple graphical interfaces for passing parameters and viewing results, and executed remotely. We are evolving towards a "standard pipeline" of computational science that can expose an experimental result as a reusable and reproducible tool in a matter of hours.

1.4 How Cloud Computing can Improve Reproducibility

1.4.1 Capturing More Variables

Virtual machines allow researchers to share the entire context of their environment — data, code, logs, usage history, intermediate results, figures, notes, failed experiments, operating system configuration details, and more. Sharing at this level of abstraction mitigates most portability issues encountered when trying to install, configure, and run someone else's software.The experimenter need not re-package their code for multiple platforms, and the reproducer need not install additional software or debug portability problems.

The virtual machine provides an exact replica of the original "laboratory" complete with all variables — controlled and uncontrolled — intact. The success of reproducibility is not contingent on the experimenter's awareness and explicit control of every variable that might have influenced the outcome (library versions, operating system versions, subtle bugs). The analogy is a "crime scene:" Keep everything pristine so investigators can reconstruct the events that led to your paper.

1.4.2 Fewer constraints on research methods

In many cases, no changes are required to your research methodology to use a virtual machine (except for a one-time cost of establishing a virtual environment in which to work). You are free to use whatever operating system, languages, libraries, tools, conventions, and practices you see fit. Except for experiments requiring specialized hardware or large external datasets (cases we will consider below), any experiments that can be run on a local machine can also be run on a virtual machine in the cloud. Other proposals to enhance reproducibility rely on the experimenter adopting some form of managed environment: language extensions and packages [27], technology-assisted metadata and documentation conventions [22, 29,

[2] http://hubzero.org/

13], or scientific workflow systems with visual programming environments and advanced provenance features [19, 3]. These systems offer enormous benefits in certain contexts, but they put significant constraints on the experimenter's research methodology: a particular language must be used, a particular programming style must be adopted, existing code must be ported to a workflow environment, or a particular documentation convention must be adopted.

In those contexts where these approaches are feasible, we advocate their use — they are generally compatible with (and complementary to) virtualization. Also, virtualization alone provides no support for managing provenance, typechecking workflows, generating documentation, or most other features provided by, say, scientific workflow systems. However, we contend that there will always be experiments performed (and data stored) outside the jurisdiction of any managed environment. Rather than ignore this data or rely on fiat, our approach is to cast a wide net to capture all data, all dependencies, and all possible variables.

The freedom to mix and match a variety of tools, to throw out one solution and rebuild another from scratch, and to re-execute one's experiment over and over at essentially zero cost are strengths of computational science that are not shared by most laboratory techniques. We should be conservative about sacrificing these properties by artificially constraining which tools can be used and how they may be combined.

1.4.3 On-Demand Backups

Snapshots of virtual machines, saved at regular intervals or on demand, offer a comprehensive (though noisy) laboratory notebook with minimal overhead. The entire state of the experiment, including controlled and uncontrolled variables, are saved and are immediately accessible. Returning to a previous state in the project involves identifying the appropriate snapshot and launching the virtual machine. The overhead of saving many copies of nearly identical VMs is increasingly mitigated by deduplication and delta techniques [34]. Search and management services for a large set of related VMs are also beginning to emerge [2].

1.4.4 Virtual Machines as Citable Publications

Virtual machines hosted on Amazon Web Services, unlike those created and managed locally, are given a unique and permanent identifier that can be referenced in papers with no additional work by the experimenter. Concerns about longevity and preservation of public cloud resources can be addressed over time and need not be considered a limitation in the near term (we will discuss preservation and longevity issues in more detail in the next section).

1.4.5 Code, Data, Environment, plus Resources

So far, all the reasons we have described apply to virtualization alone, whether or not the cloud is involved. Virtualization certainly predated cloud computing and has been used regularly in a variety of computing contexts for many years. However, only a public cloud computing environment (whether commercial or taxpayer-funded) provides not only the virtual machine, but a host in which to run it that is identical to the original author's. An experiment with any significant resource requirements cannot be fully reproduced by simply downloading the VM to run on one's laptop. For example, *de novo* assembly tasks for analyzing short read *omics sequences typically use graph-based methods that require significant memory resources [39]. Such techniques are difficult to replicate in local envi-

ronments, especially by small labs or individual researchers who cannot absorb significant investments in hardware. Cloud computing provides a common platform for anyone to use to reproduce experiments, at a cost that scales elastically with the resources required to run it. Sharing virtual machines via the cloud provides access to not only the code, data, and environment used by the experimenter, but a carbon copy of the computing resources to run them.

1.4.6 Automatic Upgrades

Data and code hosted in the cloud automatically benefits from technology advancements with no additional effort from the experimenter or those who wish to reproduce their results. For example, virtual machine images can be launched using new instance types — with, say, more memory, more cores, or special hardware[3] — as they become available. Additionally, cloud providers such as Amazon routinely release new capabilities. Consider Elastic MapReduce,[4] a new parallel processing framework natively deployed in Amazon Web Services that can be applied to any data stored in their Simple Storage Service (S3). Ths software need not be installed or configured prior to use.

1.4.7 Competitive, Elastic Pricing

The price of one EC2 compute unit, one gigabyte of RAM, and one terabyte of storage have all dropped over 50% in most usage scenarios since AWS was first released [37]. These price drops are automatically applied for all users and in some cases, retroactively. The price drops generally reflect the falling cost of hardware and new economies of scale realized in the design of AWS.

1.4.8 Reproducibility for Complex Architectures.

Computational experiments increasingly involve complex architectures, consisting of multiple servers interacting in application-specific ways: database servers [6], many-core architectures [12], and specialized resources such as GPGPUs [20]. In these cases, the code and the data are not enough. Reproducers need access to the specific hardware platforms or application architectures used in the experiment. The only choices are to document the platform carefully and hope those who wish to reproduce your results can build an appropriate environment from scratch, or to provide outside access to one's own environment. Amazon Web Services and other public cloud providers offer native facilities for precisely this purpose. Amazon's CloudFormation service allows configurations of related VMs to be saved and deployed as a single unit, making such complex architectures significantly easier to reproduce.[5] Moreover, specialized hardware includign GPGPUs and clusters with fast interconnects are now available on AWS.

1.4.9 Unfettered Collaborative Experiments

Two researchers at different institutions cannot typically work in the same development environment without one institution provisioning accounts, a process that undermines security and often takes weeks. A shared instance launched in the cloud provides "neutral territory" for developers to work in a common environment while maintaining local security.

[3] `http://aws.amazon.com/ec2/`
[4] `http://aws.amazon.com/elasticmapreduce/`
[5] `http://aws.amazon.com/cloudformation/`

Reproducible experiments can therefore be shared among multiple researchers, potentially reducing the number of independent verifications of the same result.

1.4.10 Data-intensive Computing

As science becomes increasingly data-intensive, reproducibility requires shared access to large datasets. Downloading large datasets to one's local environment to reproduce experiments simply cannot scale as datasets grow beyond a few terabytes. The only viable solution is to bring the computation to the data rather than bring the data to the computation. Unless each experimenter is equipped to open one's own environment to outside usage, the public cloud becomes the only platform that can both host the data and host the computation.

1.4.11 Cost Sharing

Reproducibility necessarily involves consideration of costs. By hosting data and code in the public cloud, the costs are shared by both the experimenter and those who wish to reproduce their results. In the simplest case, the experimenter incurs only the minimal costs of a single virtual machine image: A 30GB image will cost less than US$3.00 a month under most usage scenarios. Those wishing to reproduce the results launch their own instances from this image, incurring all relevant costs themselves. In more complex circumstances, the experimenter may need to pay storage costs to host large datasets. In some circumstances, Amazon and other cloud providers offer free hosting of public datasets in the interest of attracting traffic to their services, and online data markets are emerging that can more effectively share data storage costs between producers and consumers [5].

1.4.12 A Foundation for Single-Payer Funding

Federal funding agencies require a data management plan to accompany all proposals [24]. Hosting one's research output in the public cloud lays a foundation for a single payer system where NSF, NIH, and other agencies work directly with cloud providers to pay the costs of hosting scientific data. Individual investigators can put their research grants fully into science rather than having to plan, design, and implement a sustainable data management strategy. This approach also neatly solves a current obstacle to the uptake of cloud computing: Universities inadvertantly subsidize local deployments of hardware by charging large indirect cost rates on every dollar spent on "services," including cloud computing. Capital expenditures, including computing infrastructure, is ironically *not* subject to this overhead, despite incurring significant ongoing costs to the university in the form of power, cooling, space, and maintenance. By passing these funds directly from the funding agencies to the cloud providers, no overhead is charged.

1.4.13 Compatibility with Other Approaches

The approach we advocate — performing one's computational experiments inside virtual machines hosted in the public cloud — is compatible with and complementary to other approaches. If researchers begin to converge on a particular language, framework, workflow engine, file format, or coding convention, then these shared virtul machines will become increasingly easier to (re)use. But it is unlikely that the virtual machines will become obsolete. We argue that there will always be exceptional circumstances that require an unconstrained

programming environment, and virtual machines provide a "catch all" solution for such exceptions.

1.5 Remaining Challenges

Cloud computing offers a compelling approach to improving reproducibility in computational research, but there are risks and obstacles.

1.5.1 Cost

The economics favor a shift toward cloud computing in many situations, but not all. Storage in particular is still too expensive for science use cases, having been designed for a nearly opposite set of requirements: high availability, low latency, high frequency access by a large number of concurrent users. In contrast, most science use cases need to store large amounts of data for use by relatively few individuals who can typically tolerate delays in accessing data or even occasional outages. In some cases, even lost data can be tolerated (simulations can be re-executed, for example). In return for this tolerance, they expect prices to approximate the cost of the raw disks, which is dropping precipitously.

There are three mitigating factors to this problem: First, there are programs to host public data at no charge.[6] Second, centralization in the cloud lays a foundation for a single payer system to pay directly for publicly funded research, as we argued in the previous section. Third, the requirements for scientifc data storage are not dissimilar from those of archival applications, suggesting that the cloud providers will soon offer lower-cost services with the performance characteristics described [14].

1.5.2 Culture

A more difficult problem to solve is one of culture. Researchers are accustomed to the "ownership" of laboratory equipment, and computing infrastructure is considered just a new form of such equipment. There is skepticism (and sometimes outright misunderstanding) of the cost, reliability, security, and longevity of cloud systems. This difficulty will diminish over time as computing is increasingly considered a utility (akin to power, telephone, or Internet) and less of a specialized solution designed expressly for one's unique requirements.

The implicit subsidies for purchasing and maintaining local hardware in the University context, as we considered above, must also be eliminated before cloud computing will be fully competitive.

1.5.3 Provenance

There is no de facto method of storing and querying the history of activity within a VM, apart from primitive methods provided by the guest operating system. Reasoning about the sequence of steps that led to a particular result is the real goal of reproducing another's work, and tools to manipulate the provenance of results are key enabler [23].

[6] `http://aws.amazon.com/publicdatasets/`

1.5.4 Reuse

A virtual machine alone offers no assistance in reusing or extending software for new purposes. A pre-installed, pre-configured environment simplifies the direct reproducibility of specific commands, but is a relatively opaque representation of the underlying technique and implementation. As illustrated in Figure 1.1, other techniques make different tradeoffs in attempting to minimize effort for the experimenter and the consumers of their work, but we find that reproducibility can and should be separable from the far more general software engineering problem of reuse. In fact, demanding that all experimenal results also deliver effective reusability may be harmful: researchers will tend to over-rely on existing, standard tools and services if they know that new tools will be difficult to properly publish.

1.6 Non-challenges

Not all concerns about the application of cloud computing for scientific research are warranted in practice.

1.6.1 Security

Perceived security limitations of cloud-based systems are largely untenable. At the physical layer, it is not controversial to claim that the security of the data centers owned and managed by Microsoft, Amazon, and Google are more secure than the server room in a university lab. At the virtual layer, the system in the cloud is no less and no more vulnerable to attacks from the external Internet than local systems: firewalls are still enabled, etc. Other potential vulnerabilities, such as those arising from the hypervisor itself, have had no significant impact on the uptake of cloud computing in the enterprise, and it is difficult to argue that science data — mandated to be made public by funding agencies — is substantially more sensitive.

For sensitive data (HIPAA, ITAR, etc.), the cloud may or may not be appropriate for a given institution or application. Exemplar applications demonstrating the feasibility of cloud computing for applications involving sensitive data do exist, however[7], and the federal government is a significant customer.[8]

1.6.2 Licensing

Licensing issues may seem to complicate the use of public cloud infrastructures, but the problems appear to be transient. In some cases, licensing issues may actually be simpler in the cloud due to the rigorous administrative control of the underlying infrastructure, aggregate buying power of large cloud vendors, and the payment infrastructure already integrated. For example, one can rent fully licensed virtual machines equipped with Mathematica,[9] and MathWorks offers a variety of demonstrations of using MATLAB with AWS.[10]

[7] `http://aws.amazon.com/about-aws/whats-new/2009/04/06/whitepaper-hipaa/`
[8] `http://fedcloud.gov`
[9] `http://www.wolfram.com/news/cloudcomputing.html`
[10] `http://aws.typepad.com/aws/2008/11/parallel-comput.html`

1.6.3 Vendor Lock-In and Long-Term Preservation

Dependencies on commercial providers of goods and services are ubiquitous in all areas of science, but dependencies on vendors of computing infrastructure receive significantly more scrutiny. We have encountered enough instances of inaccessible data locked in university-owned resources that the service level agreements (SLAs) and relative mature infrastructure offered by cloud providers appears significantly more reliable for long-term access. In addition to commercial providers, open source and taxpayer-funded efforts, sometimes modeled after Amazon Web Services but designed expressly for research, are also emerging [16, 30, 25]. Moreover, high-performance computing facilities are increasingly interested in supplying cloud-like facilities, especially virtualization and on-demand provisioning [26].

The long-term role of cloud computing in the sciences is still evolving, but there appear to be immediate benefits for reproducibility in using and sharing virtual environments for computational experiments. Informed by these benefits, the University of Washington eScience Institute emphasizes cloud computing as a key strategy in enabling the next generation of rigorous and reproducible computational science.

Bibliography

[1] Enis Afgan, Dannon Baker, Nate Coraor, Brad Chapman, Anton Nekrutenko, and James Taylor. Galaxy cloudman: delivering cloud compute clusters. *BMC Bioinformatics*, 11 Suppl 12:S4, 2010.

[2] Glenn Ammons, Vasanth Bala, Todd Mummert, Darrell Reimer, and Xiaolan Zhang. Virtual machine images as structured data: the mirage image library. In *Proceedings of the 3rd USENIX conference on Hot topics in cloud computing*, HotCloud'11, pages 22–22, Berkeley, CA, USA, 2011. USENIX Association.

[3] Erik Andersen, Steven P. Callahan, David A. Koop, Emanuele Santos, Carlos E. Scheidegger, Huy T. Vo, Juliana Freire, and Claudio T. Silva. Vistrails: Using provenance to streamline data exploration. In *Poster Proceedings of the International Workshop on Data Integration in the Life Sciences (DILS)*, page 8, 2007. Invited for oral presentation.

[4] Samuel Angiuoli, Malcolm Matalka, Aaron Gussman, Kevin Galens, Mahesh Vangala, David Riley, Cesar Arze, James White, Owen White, and W Florian Fricke. Clovr: A virtual machine for automated and portable sequence analysis from the desktop using cloud computing. *BMC Bioinformatics*, 12(1):356, 2011.

[5] Magdalena Balazinska, Bill Howe, and Dan Suciu. Data markets in the cloud: An opportunity for the database community. *PVLDB*, 4(12):1482–1485, 2011.

[6] BioSQL. `http://biosql.org`.

[7] C. Titus Brown. Our approach to replication in computational science. `http://ivory.idyll.org/blog/replication-i.html`.

[8] C. Titus Brown, Adina Howe, Qingpeng Zhang, Alexis B. Pyrkosz, and Timothy H. Brom. Running the diginorm paper script pipeline. `http://ged.msu.edu/angus/diginorm-2012/pipeline-notes.html`.

[9] J. Gregory Caporaso, Justin Kuczynski, Jesse Stombaugh, Kyle Bittinger, Frederic D. Bushman, Elizabeth K. Costello, Noah Fierer, Antonio Gonzalez G. Peña, Julia K. Goodrich, Jeffrey I. Gordon, Gavin A. Huttley, Scott T. Kelley, Dan Knights, Jeremy E. Koenig, Ruth E. Ley, Catherine A. Lozupone, Daniel McDonald, Brian D. Muegge, Meg Pirrung, Jens Reeder, Joel R. Sevinsky, Peter J. Turnbaugh, William A. Walters, Jeremy Widmann, Tanya Yatsunenko, Jesse Zaneveld, and Rob Knight. QIIME allows analysis of high-throughput community sequencing data. *Nature methods*, 7(5):335–336, May 2010.

[10] Andrew Davison. Automated capture of experiment context for easier reproducibility in computational research. *Computing in Science and Engg.*, 14(4):48–56, July 2012.

[11] Joel T. Dudley and Atul J. Butte. In silico research in the era of cloud computing. *Nature biotechnology*, 28(11):1181–1185, November 2010.

[12] Francisco Jos Esteban, David Daz, Pilar Hernndez, Juan Antonio Caballero, Gabriel Dorado, and Sergio Glvez. Direct approaches to exploit many-core architecture in bioinformatics. *Future Generation Comp. Syst.*, 29(1):15–26, 2013.

[13] Matan Gavish and David Donoho. Three dream applications of verifiable computational results. *Computing in Science and Engineering*, 14(4):26–31, 2012.

[14] James Hamilton. Internet scale storage. Keynote presentation, SIGMOD 2011, Athens Greece.

[15] The Kepler Project. `http://kepler-project.org`.

[16] Jonathan Klinginsmith, Malika Mahoui, and Yuqing Melanie Wu. Towards reproducible escience in the cloud. In *CloudCom2011*, Athens, Greece, 12/2011 2011. IEEE, IEEE.

[17] K. Krampis, T. Booth, B. Chapman, B. Tiwari, M. Bicak, D. Field, and K.E. Nelson. Cloud biolinux: pre-configured and on-demand bioinformatics computing for the genomics community. *BMC Bioinformatics*, 13, 2012.

[18] Large Hadron Collider (LHC). `http://lhc.web.cern.ch`.

[19] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger-Frank, M. Jones, E. Lee, J. Tao, and Y. Zhao. Scientific Workflow Management and the Kepler System. *Concurrency and Computation: Practice & Experience*, 2005.

[20] Syed Faraz Mahmood and Huzefa Rangwala. Gpu-euler: Sequence assembly using gpgpu. In *HPCC*, pages 153–160, 2011.

[21] Michael McLennan and Rick Kennell. Hubzero: A platform for dissemination and collaboration in computational science and engineering. *Computing in Science and Engineering*, 12(2):48–53, 2010.

[22] Luc Moreau, Ben Clifford, Juliana Freire, Joe Futrelle, Yolanda Gil, Paul Groth, Natalia Kwasnikowska, Simon Miles, Paolo Missier, Jim Myers, Beth Plale, Yogesh Simmhan, Eric Stephan, and Jan Van den Bussche. The open provenance model core specification (v1.1). *Future Gener. Comput. Syst.*, 27(6):743–756, June 2011.

[23] Kiran-Kumar Muniswamy-Reddy, Peter Macko, and Margo Seltzer. Provenance for the cloud. In *Proceedings of the 8th USENIX conference on File and storage technologies*, FAST'10, pages 15–14, Berkeley, CA, USA, 2010. USENIX Association.

[24] NSF data management plan requirements. `http://www.nsf.gov/eng/general/dmp.jsp`.

[25] Daniel Nurmi, Richard Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, and Dmitrii Zagorodnov. The eucalyptus open-source cloud-computing system. In *Cloud Computing and Its Applications (CCA '08)*, 2008.

[26] US Department of Energy. Magellan final report. `http://science.energy.gov/~/media/ascr/pdf/program-documents/docs/Magellan_Final_Report.pdf`.

[27] Roger Peng. Caching and distributing statistical analyses in r. *Journal of Statistical Software*, 26(7):1–24, 7 2008.

[28] Roger D. Peng. Reproducible research and Biostatistics. *Biostatistics*, 10(3):405–408, July 2009.

[29] Eric Schulte, Dan Davison, Thomas Dye, and Carsten Dominik. A multi-language computing environment for literate programming and reproducible research. *Journal of Statistical Software*, 46(3):1–24, 1 2012.

[30] San diego supercomputing center cloud storage services. `https://cloud.sdsc.edu`.

[31] B. Segal, P. Buncic, C. Aguado Sanchez, J. Blomer, D. Garcia Quintas, A. Harutyunyan, P. Mato, J. Rantala, D. Weir, and Y. Yao. LHC Cloud Computing with CernVM. In *Proceedings of the 13th International Workshop on Advanced Computing and Analysis Techniques in Physics Research. February 22-27, 2010, Jaipur, India. http://acat2010.cern.ch/. Published online at ¡A href="http://pos.sissa.it/cgi-bin/reader/conf.cgi?confid=93"¿http://pos.sissa.it/cgi-bin/reader/conf.cgi?confid=93¡/A¿, p.4*, 2010.

[32] Lincoln Stein. The case for cloud computing in genome informatics. *Genome Biology*, 11(5):207, 2010.

[33] Michael Stonebraker. Scidb: An open-source dbms for scientific data. *ERCIM News*, 2012(89), 2012.

[34] Petter Svärd, Benoit Hudzia, Johan Tordsson, and Erik Elmroth. Evaluation of delta compression techniques for efficient live migration of large virtual machines. *SIGPLAN Not.*, 46:111–120, March 2011.

[35] The Taverna Project. `http://taverna.sourceforge.net`.

[36] The Triana Project. `http://www.trianacode.org`.

[37] Cloud economics: Visualizing aws prices over time. `http://escience.washington.edu/blog/cloud-economics-visualizing-aws-prices-over-time`.

[38] The VisTrails Project. `http://www.vistrails.org`.

[39] Daniel R Zerbino and Ewan Birney. Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome Research*, 18(5):821–9, 2008.