Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 4 (2011) 648–657

Procedia Computer Science

International Conference on Computational Science, ICCS 2011

A Provenance-Based Infrastructure to Support the Life Cycle of Executable Papers

David Koop[a,*], Emanuele Santos[a], Phillip Mates[a], Huy T. Vo[a], Philippe Bonnet[b], Bela Bauer[c], Brigitte Surer[c], Matthias Troyer[c], Dean N. Williams[d], Joel E. Tohline[e], Juliana Freire[a], Cláudio T. Silva[a]

[a] *University of Utah*
[b] *IT University of Copenhagen*
[c] *ETH Zurich*
[d] *Lawrence Livermore National Laboratory*
[e] *Louisiana State University*

Abstract

As publishers establish a greater online presence as well as infrastructure to support the distribution of more varied information, the idea of an executable paper that enables greater interaction has developed. An executable paper provides more information for computational experiments and results than the text, tables, and figures of standard papers. Executable papers can bundle computational content that allow readers and reviewers to interact, validate, and explore experiments. By including such content, authors facilitate future discoveries by lowering the barrier to reproducing and extending results. We present an infrastructure for creating, disseminating, and maintaining executable papers. Our approach is rooted in *provenance*, the documentation of exactly how data, experiments, and results were generated. We seek to improve the experience for everyone involved in the life cycle of an executable paper. The automated capture of provenance information allows authors to easily integrate and update results into papers as they write, and also helps reviewers better evaluate approaches by enabling them to explore experimental results by varying parameters or data. With a provenance-based system, readers are able to examine exactly how a result was developed to better understand and extend published findings.

Keywords:
executable paper, reproducibility, provenance, scientific workflows

1. Introduction

While computational experiments have become an integral part of the scientific method, it is still a challenge to repeat such experiments, because often, computational experiments require specific hardware, non-trivial software installation, and complex manipulations to obtain results. Generating and sharing repeatable results takes a lot of work with current tools. Thus, a crucial technical challenge is to make this easier for (i) authors, (ii) reviewers, (iii) publishers, and (iv) readers.

*Correspondence: dakoop@cs.utah.edu

Selection and/or peer-review under responsibility of Prof. Mitsuhisa Sato and Prof. Satoshi Matsuoka
doi:10.1016/j.procs.2011.04.068

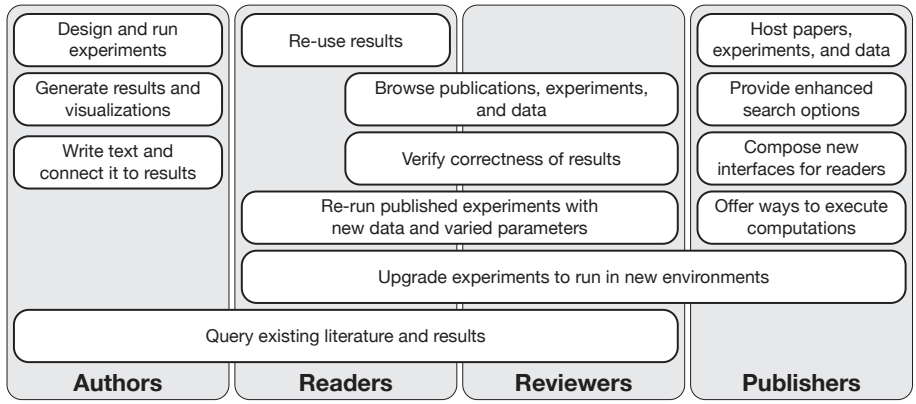

Figure 1: Each group involved in creating, reviewing, reading, and maintaining executable papers requires a different set of features.

While a number of tools have been developed that attack sub-problems related to the creation of reproducible papers, no end-to-end solution is available. Besides giving authors the ability to link results to their provenance, such a solution should enable reviewers to assess the correctness and the relevance of the experimental results described in a submitted paper. Furthermore, upon publication, readers should be able to repeat and utilize the computations embedded in the papers. But even when the provenance associated with a result is represented as a workflow, which consists of a precise and executable specification of computational processes, shipping the workflow to be run in an environment different from the one in which it has been designed at raises many challenges. From hard-coded locations for input data, to dependencies on specific versions of software libraries and hardware, adapting workflows to run in a new environment can be challenging and sometimes impossible. One possible solution is to encapsulate workflows within a virtual machine or create a package that includes all the dependencies (e.g., system libraries) [1], but this may not be possible in some scenarios. For example, an execution that requires special hardware or the use of very large data sets cannot be packaged or moved. In addition, repeating an experiment does not guarantee its correctness or relevance, and interpreting results requires deep insight into the experimental framework.

We posit that integrating data acquisition, derivation, analysis, and visualization as executable components throughout the publication process will make it easier to generate and share repeatable results. In this paper, we describe an infrastructure we have built to support the life cycle of *executable publications*—their creation, review and re-use. Our focus is on publications whose *computational experiments* can be reproduced and validated. Videos of our infrastructure in action are available at `http://www.vistrails.org/index.php/ExecutablePapers`.

In our design we have considered the following desiderata: *Lower Barrier for Adoption*—it should help authors in the process of assembling their submissions; *Flexibility*—it should support multiple mechanisms that give authors different choices of how to package their work; *Support for the Reviewing Process*—reviewers should be able to reproduce the experiments, as well as validate them. A key component of the architecture is a provenance management system that captures useful meta-data, including workflow provenance, source code, and library versions. We have also developed a set of solutions to address different aspects of the executable paper problem. These include methods to link results to their provenance, reproduce results, explore parameter spaces, interact with results through a Web-based interface, and upgrade the specification of computational experiments to work in different environments and with newer versions of software. To cater to the wide range of requirements for different types of scientific results, these components were developed to be mixed, matched, and replaced.

In Section 2, we outline the desired features for those involved in the life cycle of an executable paper. Next, Section 3, we outline both the challenges we have encountered in supporting executable papers and the solutions and components we have developed to address these challenges. We give short descriptions of real-world uses of our infrastructure in Section 4 before concluding with a discussion of our progress in Section 5.

2. The Life Cycle of Executable Papers

An infrastructure for executable papers must address each step in the life cycle of a paper. As authors develop ideas and experiments, we can aid their future writing by tracking the computations performed, the data used, and the parameters selected. The resulting data, plots, or visualizations can be linked directly from the paper to the

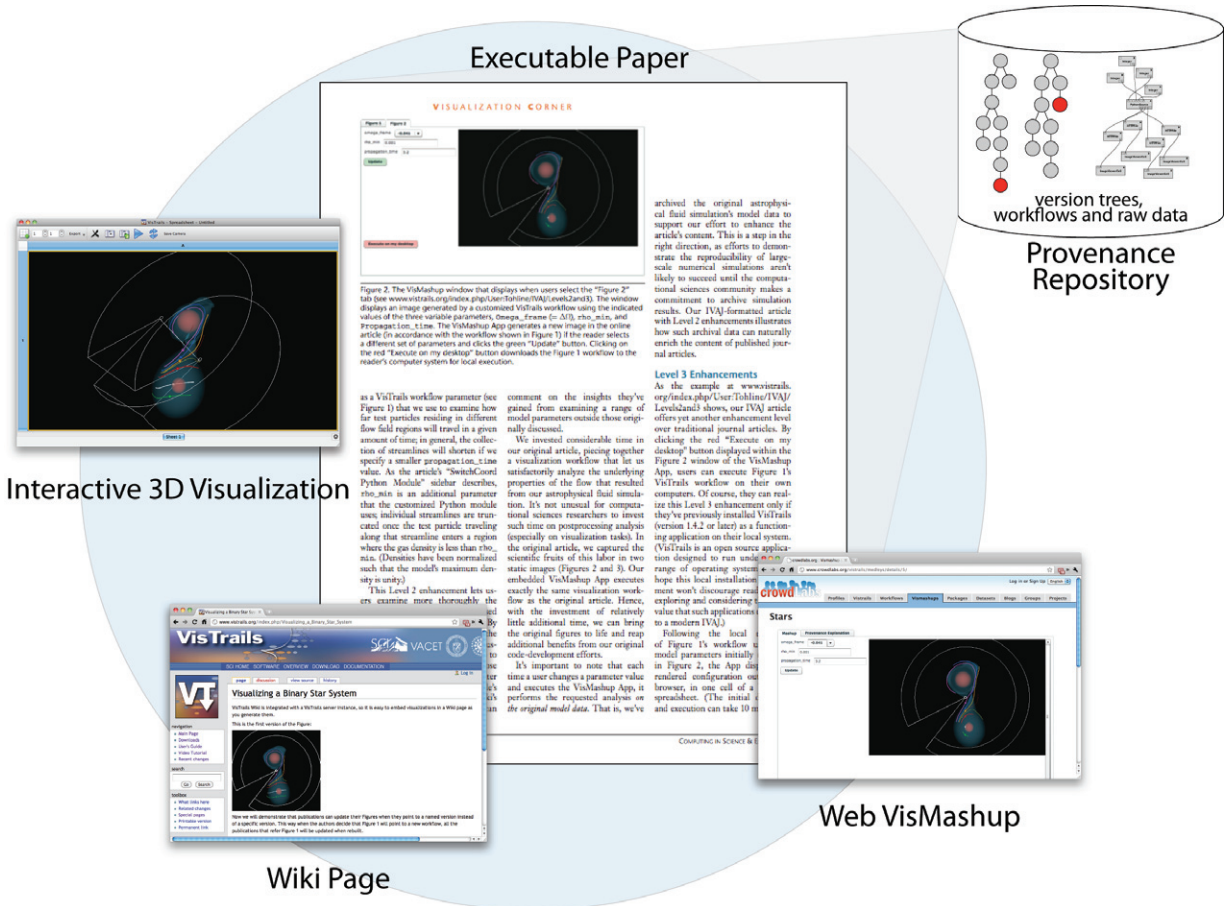

Figure 2: Our infrastructure integrates data acquisition, derivation, analysis, and visualization as executable components throughout the publication process. Provenance-rich results, both static and interactive, can be published using different media, including PDF, Word, PowerPoint, Wiki and HTML pages

experiments and inputs. Changes that influence results can then be automatically incorporated into papers. Reviewers need to evaluate submitted work, and having access to executable papers allows them to better examine the results. At the same time, reproducing or modifying an execution requires access to data and a suitable environment. Publishers are concerned with disseminating papers, and executable papers add value as they can provide a more interactive experience. At the same time, they require more infrastructure to allow distribution of added resources and features to allow enhanced querying. When readers access published executable papers, they can better understand results and explore new directions using the embedded computations as a starting point. However, like reviewers, they must have the means to access data and compatible execution environments. Figure 1 gives an overview of these requirements.

Authors. An executable paper begins with its author. But often, ideas and results have been generated before the writing begins. Thus, an author benefits from doing work in an environment that simplifies the writing of an executable paper. Provenance is a critical ingredient for such work [2, 3, 4]. If we know how a figure or table was generated, we can incorporate that computation information in the paper so it can be reproduced. Similarly, if we know exactly what data was used to derive conclusions, we can more easily determine what data to include with the executable paper. In addition to tracking provenance, we must consider how computations should be specified so that not only provenance can be captured but also for future review and use.

We believe that support for a wide variety of modes of development, writing, and packaging is crucial for the wide adoption of executable papers. Workflow systems present a suitable infrastructure to both integrate a large number of tools and abstract computation to achieve understandable representations. They can also automatically capture the provenance of both computations and data as authors generate results.

Reviewers. An executable paper has the potential to improve the quality of reviews because reviewers have the ability to explore and validate conclusions. At the same time, there are a number of potential issues when a reviewer receives an executable paper. The reviewer needs to be able to reproduce and test the existing computations, but without access to the correct hardware or software packages, this can be difficult. In addition, a reviewer needs to be able to test different parameter configurations and access and interact with the computations.

Publishers. After an executable paper is accepted, it is important that the executable nature of the publication is maintained. The format of data used as well as the format of the computation will be important to ensure the longevity and future use of the paper. While authors may distribute data or code on their own, publishers can ensure that such resources are maintained and archived. To do so, they need infrastructure to support storage and distribution of the different components of an executable paper. In addition, they can add value to by incorporating Web-based methods to search and interact with papers.

Readers. Executable papers enable a richer experience for readers by allowing them to interact with results, review computations, and link to input data. However, they face similar challenges as reviewers. In many cases, the easier the access to resources is, the better. The ability to query more than text can help readers locate similar data or experiments. Web-based interaction can serve to interest readers who can start exploring data or experiments; later, they may wish to extend or modify computations on their own machine.

3. Provenance-Based Infrastructure: Challenges and Approaches

3.1. Provenance

If a figure or table is generated from some computation, how do we know exactly what code, parameters, and input were used in that computation? One common problem in reproducing a solution is a step that is omitted from the pseudo-code or a parameter that was changed for a result but not updated in the text or caption. Such situations should be more transparent so that readers can reference the exact computation run. We can reduce the authors' effort in providing such transparency by automatically tracking and updating the provenance of each result (see Figure 2).

In our infrastructure, we are using VisTrails, a provenance-enabled, workflow-based data exploration tool, to capture provenance information. While VisTrails captures data and workflow evolution provenance, we have developed infrastructure for versioning and storing data so that we can trace provenance back to the exact version of a file used as input [5]. The *persistence package* contains modules that identify data by its contents (via hashing), its use (via the workflow that generated it), and its history (via a version control system). This ensures that an author can always retrieve data used in previous work, even if the original file has been changed or even removed. In addition, it permits efficient re-use of data that has already been generated. To support provenance-rich results in papers and the ability to link them back to the workflows that derived them, we have *developed code and plug-ins for LaTeX, Wiki, Microsoft Word, and PowerPoint* (see Figure 2). This allows authors to easily embed and reproduce results, and readers to follow links to and explore the actual computations. We have used crowdLabs (`http://www.crowdlabs.org`) to allow papers to point to results that can be executed on a remote server and interactively explored from a Web browser. Figure 3 shows a high-level overview of our infrastructure. Below, we describe its components in more details.

We should note that while we have used VisTrails, our infrastructure is compatible with other provenance-enabled tools [6]. The basic requirement is that the tool not only captures provenance but can also replay the provenance to reproduce the results. We are currently working on provenance enabling tools in different domains, including bioinformatics and visualization, and we plan combine them with our infrastructure.

3.2. Execution Infrastructure

Specifying Computations. Some analyses are conducted using domain-specific tools, others using command-line scripts, and some with workflow systems. A publisher could limit the environments for development, but this will likely limit participation. Workflows present the possibility of abstracting computation using wrappers, and different levels of granularity can be used to facilitate the understanding of the processes and interpretation of the results. VisTrails is a workflow system built using Python and allows a wide variety of existing tools and libraries to be incorporated using a simple wrapping scheme. Even if existing tools cannot be integrated with Python, it is possible to run them via command-line wrappers. In addition, the system automatically and unobtrusively captures workflow evolution provenance as well as data provenance [7, 8].

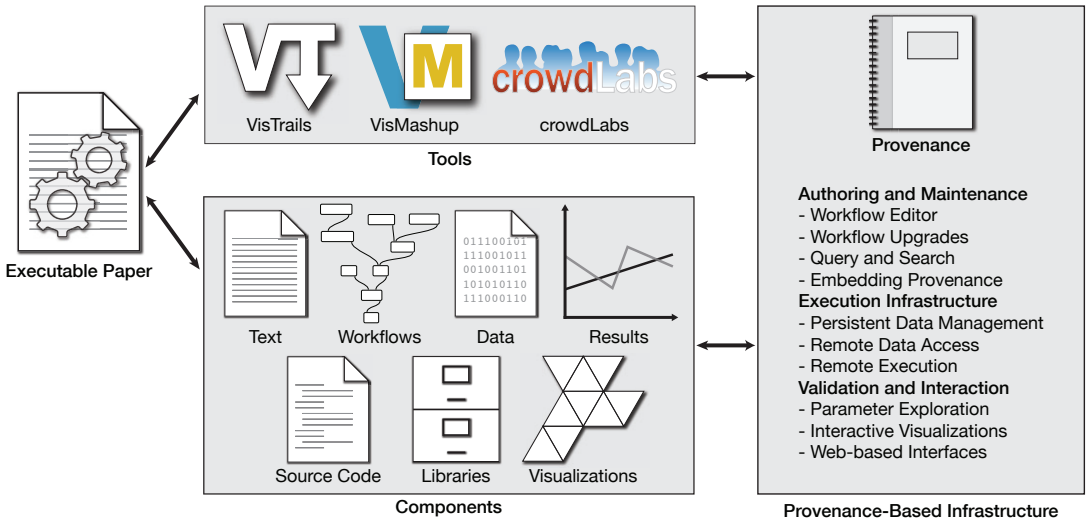

Figure 3: Our infrastructure uses existing systems and includes a set of components required to support the life cycle of executable papers.

Execution Environment. One of the first issues for an executable paper is the infrastructure for execution. If a solution runs on Linux, a Windows user will not be able to easily run the solution. Some executables like Java archives or Python scripts can be run on other systems, but require some infrastructure to be installed. Another solution is to leverage server-based execution; if all computations occur on a server farm, a publisher or author might provide global access. However, this requires that all computations run in that specific environment and limits the ability to tweak and explore the computations.

While VisTrails is cross-platform, the code, libraries, and other dependencies underlying a workflow also need to be cross-platform. We can employ virtual machines to mimic a specific environment, but we also encourage authors to include special modules that check pre- and post-conditions as part of the workflow. If specific requirements are not met, the workflow can alert the reader of any incompatibility. We are currently working on functionality to package a workflow with both the underlying implementations of referenced modules and the data used. There are also tools (see e.g., [1]) that capture dependencies at the system-level that could be used to build such self-contained workflows. Finally, most workflow systems, including VisTrails, support annotation of workflows and data. This allows authors to include details about workflow use or valid parameter settings that need not be featured in the paper.

Supercomputing and Special Systems. When published work utilizes special architectures that cannot be easily accessed, how can results be reproduced or interacted with? If the system is open, workflows or scripts can be executed remotely. However, for closed systems, this process is more difficult. If hardware is not public, it may be possible to allow reviewers access via special accounts. It will thus be important to provide a flexible mechanism that allows different kinds of execution.

Some problems can be scaled to allow demonstration on consumer hardware, but for work that depends on the hardware, this might not be feasible. Some computations might have taken weeks or months and consumed substantial resources. Results from such long-running computations could instead be treated like raw experimental data, and archived with extensive provenance information—allowing an expert reader to redo the simulations in the future on their own supercomputer systems if desired. The executable paper will then contain the full post-processing of this data but not attempt to redo any expensive calculation. This restriction in scope is in practice not a limitation, neither for the reviewer, nor the reader. Neither of them will typically want to invest substantial supercomputing resources and potentially wait for days and weeks to redo the simulation, but their main interest will be on the analysis of the supercomputer output. Only specialists might want to redo the full simulations, but then typically using their own code in scientific follow-up projects. For that purpose, having access to the full provenance information, inputs and outputs of the simulations reported on in the paper is often sufficient. For examples of this approach see the discussion of the ALPS project [9, 10, 11, 12] in Section 4. The ALPS 2.0 paper [12] is an existing executable paper that contains examples of results that can be reproduced in a local machine as well as others that access archived results of supercomputer simulations.

Local, Remote, and Mixed Execution. As discussed above, while some workflows can run on a local machine, others must run remotely, e.g., on a cloud or cluster infrastructure. We have been exploring different strategies to support workflow execution on clusters both using native code and via Kitware's ParaView [13] for parallel execution. One of the concerns is controlling these executions from a local machine, and we have developed workflow components to automate the process monitoring. Another concern is a result that uses a very large data set or data that is proprietary; we may be able to run the workflow locally but need to remotely query or aggregate the data. We have developed modules that work with relational databases to support such remote queries in local workflows. It is also possible for authors to create modules that are specific to their data as well as that provide remote access to their own computational infrastructure (hardware and software). For an example of these, see [14].

3.3. Validation and Interaction

Interacting, Testing and Validating Workflows and their Results. We have developed interfaces to simplify workflow interaction and exploration. If a reader is unfamiliar with the environment where the computation is performed, how can she be expected to interact with it? Furthermore, if it is not obvious what parameters can be changed or how they might be changed, a reader cannot explore different cases. Even for reviewers familiar with programming, we cannot assume knowledge of any programming language. That said, it is also unreasonable to enforce a specific run-time or language upon the author. Abstract, higher-level access must be available, but it must still allow full configuration.

VisMashup is a tool that allows authors to quickly build high-level views of workflows [15]. It enables authors to highlight a subset of all parameters and suggest starting values as reviewers explore computations. It also can generate turnkey and Web-based applications with an intuitive, simplified interface to lower the barrier for reviewers to validate results. Through crowdLabs (see above), authors can publish these mashups on the Web. We use Flash to support interactive visualizations, and we are exploring newer technology, such as HTML5 and WebGL.

Currently, it is difficult for reviewers to evaluate work not only because they cannot easily verify results but also because they cannot test cases that are not discussed. At the same time, it is tedious to manually test each case. Parameter sweeps are an important tool in automating much of this evaluation. VisTrails provides a parameter exploration interface for quickly selecting and setting ranges. In addition, it has an intuitive spreadsheet interface for visually comparing results [7]. This allows reviewers to assess how general a solution is, how much tuning it requires, and cases where it fails.

Data Access and Formats. While some papers present results which use proprietary data, others may use data that is too large to duplicate. As with hardware, if the data is stored in an accessible repository, we can envision running tests or extensions by pushing the computation to local systems and only retrieving results. Other data may not be available due to copyright or ownership concerns. Another frustration when evaluating work is that the format of input data is often specific to the paper. A set of papers that address similar data might adopt different formats for their data. There has been some standardization, especially in certain fields, but morphing data from one format to another is still a necessary part of adopting computations. In VisTrails, we provide data conversion modules to help with such changes but also allow users to write source code to tweak data for their own use.

3.4. Maintenance and Sharing

Longevity. As systems evolve and hardware changes, simply archiving an executable will not suffice. Even archiving code can be problematic as language specifications change. Archiving environments as virtual machines usually allows reproduction but presents issues with scale. Furthermore, while exact reproduction is important to verify results, utilizing published work to extend solutions is more efficient when modern tools and algorithms can be substituted. Such upgrades can accelerate progress by allowing readers to take advantage of new hardware and infrastructure.

VisTrails provides a mechanism to store provenance for workflows with the exact version of each module used [16]. If a reader downloads an old executable paper, VisTrails can automatically upgrade the computations to match the current environment, allowing him to immediately start exploring. This is accomplished by requiring package developers to specify upgrade paths when specifications change.

Querying and Re-Using Published Results. Because executable papers contain more than text and meta-data, it is important that the executable elements can also be queried. The ability to query this information opens up new opportunities for knowledge re-use: readers may more easily find papers that use a specific data set or employ a set

The ALPS project release 2.0:
Open source software for strongly correlated systems

B. Bauer[1] L. D. Carr[2] H.G. Evertz[3] A. Feiguin[4] J. Freire[5] S. Fuchs[6] L. Gamper[1] J. Gukelberger[1] E. Gull[7] S. Guertler[8] A. Hehn[1] R. Igarashi[9,10] S.V. Isakov[1] D. Koop[5] P.N. Ma[1] P. Mates[1,2] H. Matsuo[11] O. Parcollet[12] G. Pawlowski[13] J.D. Picon[14] L. Pollet[1,15] E. Santos[5] V.W. Scarola[16] U. Schollwöck[17] C. Silva[5] B. Surer[1] S. Todo[10,11] S. Trebst[18] M. Troyer[1,‡] M. L. Wall[2] P. Werner[1] S. Wessel[19,20]

[1]Theoretische Physik, ETH Zurich, 8093 Zurich, Switzerland
[2]Department of Physics, Colorado School of Mines, Golden, CO 80401, USA
[3]Institut für Theoretische Physik, Technische Universität Graz, A-8010 Graz, Austria
[4]Department of Physics and Astronomy, University of Wyoming, Laramie, Wyoming 82071, USA
[5]Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, Utah 84112, USA
[6]Institut für Theoretische Physik, Georg-August-Universität Göttingen, Göttingen, Germany
[7]Columbia University, New York, NY 10027, USA
[8]Bethe Center for Theoretical Physics, Universität Bonn, Nussallee 12, 53115 Bonn, Germany
[9]Center for Computational Science & e-Systems, Japan Atomic Energy Agency, 110-0015 Tokyo, Japan
[10]Core Research for Evolutional Science and Technology, Japan Science and Technology Agency, 332-0012 Kawaguchi, Japan
[11]Department of Applied Physics, University of Tokyo, 113-8656 Tokyo, Japan
[12]Institut de Physique Théorique, CEA/DSM/IPhT-CNRS/URA 2306, CEA-Saclay, F-91191 Gif-sur-Yvette, France
[13]Faculty of Physics, A. Mickiewicz University, Umultowska 85, 61-614 Poznań, Poland
[14]Institute of Theoretical Physics, EPF Lausanne, CH-1015 Lausanne, Switzerland
[15]Physics Department, Harvard University, Cambridge 02138, Massachusetts, USA
[16]Department of Physics, Virginia Tech, Blacksburg, Virginia 24061, USA
[17]Department for Physics, Arnold Sommerfeld Center for Theoretical Physics and Center for NanoScience, University of Munich, 80333 Munich, Germany
[18]Microsoft Research, Station Q, University of California, Santa Barbara, CA 93106, USA
[19]Institute for Solid State Theory, RWTH Aachen University, 52056 Aachen, Germany

‡ Corresponding author: troyer@comp-phys.org

http://arxiv.org/abs/1101.2646

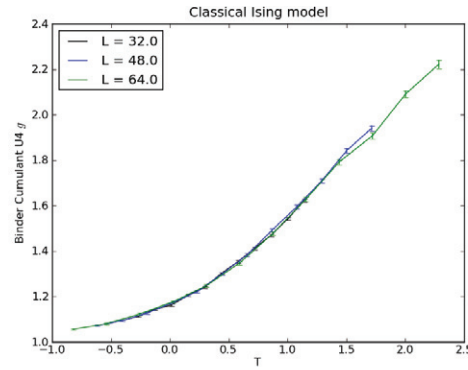

Figure 3. *In this example we show a data collapse of the Binder Cumulant in the classical Ising model...*

LaTeX command used to embed plot:

```
\vistrail[wfid=935,
  buildalways=false]{width=1.0\linewidth}
```

Figure 4: Supporting the publication of reproducible results in the paper describing the ALPS 2.0 release. On the bottom right, the command used to include the plot in the LaTeX source is displayed. Clicking on the plot will open the original workflow in crowdLabs.org (available at http://www.crowdlabs.org/vistrails/workflows/details/935/).

of specific computational techniques in a given sequence. Since specifying a textual query about the structure of a workflow is difficult and unintuitive, we developed a technique for building queries by example [17]. In addition, we have worked on developing search engines that display snippets that allow users to filter results based on computational structure [18]. Finally, we have built a workflow-specific language for incorporating textual query with queries on provenance and computational structure [19].

As described in the previous section, authors can choose to provide higher-level interfaces as VisMashups. This tool also allows readers to combine these views as mashups in a similar manner to Web mashups. A reader might wish to run a simulation using one published result and analyze the results using a second analysis technique. VisMashups provide an interface for quickly combining such tools.

4. Case Studies

We now present three projects where our infrastructure is being used to create executable papers. We also discuss examples of papers that use different approaches to packaging their experiments.

The ALPS Project. The ALPS project (Algorithms and Libraries for Physics Simulations) is an open-source initiative for the simulation of large quantum many body systems [9, 10, 11, 12] which has been used in about two hundred research projects over the past six years. One of its core goals has been to simplify archival longevity and repeatability of simulations by standardizing input and result file formats.

Motivated both by demands for repeatable simulations and the ETH Zurich research ethics guidelines [20], which require that all steps from input data to final figures need to be archived and made available upon request, the recent 2.0 release takes an important further step. ALPS 2.0 [12] makes use of the infrastructure presented in this paper to support data analysis and exploration, as well as the publication of reproducible results.

For small simulations, VisTrails combined with ALPS allows for complete reproducibility of any result in a paper, as we have demonstrated in the ALPS 2.0 paper [12], which is an existing executable paper employing our infrastructure. After installing VisTrails and ALPS on either a Linux, MacOS, or Windows system, clicking on a figure activates a “deep caption” which retrieves the workflow and executes the calculation leading to the figure on the user’s machine. For an example see Figure 4, which is the same as Figure 3 in the ALPS 2.0 paper [12]. For large scale simulations, ALPS 2.0 projects are typically split into two parts—a time-consuming set of simulations and a set

of analysis workflows. The simulations are run on high performance machines, and their results are stored on archival servers together with execution logs and provenance information. The workflows included in executable papers start from these result files, retrieved from an archival data server, and perform analyses to generate a final figure. For an example, see Figure 3 in the ALPS 2.0 paper [12].

The ALPS libraries and applications support full backward compatibility to older versions of the input and output files, thus enabling executable ALPS papers to be run on future versions. Several of the ALPS applications have been completely rewritten since the original version, keeping backward compatibility to all old input and result files, and future version wills continue to support old data files. The compatibility of past and current workflows with future versions is achieved by using the upgrade framework from VisTrails. The workflow used for Figure 1 of the ALPS 2.0 paper [12] has actually already gone through such automated upgrades!

The ACM SIGMOD Repeatability Effort. As a step towards the goal of computational repeatability, the ACM SIGMOD conference has offered, since 2008 [21, 22], to verify the experiments published in the papers accepted at the conference. A committee is in charge of repeating the experiments provided by authors (repeatability), and exploring the impact of modifying experiment parameters (workability). In 2010, 20% of the accepted papers received the repeatability label.

Proper verification requires that reviewers have access to the necessary software and data, and that they can determine the relevance and accuracy of the submitted experiments. Without proper infrastructure or guidelines, the task can be frustrating and take considerable time. At the same time, authors cannot be expected to spend significant time and effort porting their experiments to a specific infrastructure.

To aid reviewers, we are encouraging SIGMOD authors to adhere to the following guidelines:[1]

(a) Rely on our provenance-based workflow infrastructure to automate experimental setup and execution tasks.
(b) Use a virtual machine (VM) as the environment for experiments.
(c) Explicitly represent pre- and post-conditions for setup and execution tasks.

A common infrastructure guarantees the uniformity of representation across experiments so reviewers need not re-learn the experimental setup for each submission. By making data derivation or acquisition explicit, the structure of workflows helps reviewers understand the design of the experiments as well as determine which portions of the code are accessible. Once a VM is configured to run an experiment, it should be straightforward to install it on a different platform running the appropriate virtualization software, e.g., VirtualBox (it is the author's responsibility to test it). This should dramatically improve the communication between authors and reviewers as they actually can exchange VM instance snapshots and thus potentially leverage the provenance features of VisTrails. While virtual machines ensure the portability of the experiments so reviewers need not worry about system inconsistencies, explicit pre- and post-conditions makes it possible for reviewers to check the correctness of the experiment under the given conditions.

To help authors, we have extended VisTrails to better support *remote* executables in a workflow via XML RPC [23]. This allows authors with proprietary software or data, or even specific hardware, to make portions of their experimental framework accessible remotely to reviewers. In addition, if authors use our infrastructure throughout the life cycle of their papers, satisfying the repeatability criteria should be automatic. Note that our infrastructure can also be used to archive results internally, allowing them to be reproduced later for a journal version of a given paper or after the person who designed and ran the experiment has left the group.

Visualizing Mass-Transfer in Binary Star Systems. As computational science evolves, the static printed journal article cannot serve readers in the same way that a digital, executable version can. Not only is it more difficult for readers to rebuild algorithms and regather data, but readers are also not able explore the results by simply viewing static images. Many journals have started to break out specific elements of papers in online formats to allow readers to view high-resolution versions of figures or examine source code in a proper format (*e.g.,* Python code with proper indentation). However, such efforts do little to allow readers or reviewers to validate or interact with data and results; the ability to change parameter values and see results can be very instructive. In addition, higher-resolution figures cannot capture the depth inherent in three-dimensional or time-varying visualizations.

The astrophysics group at Louisiana State University (LSU) has used our provenance-based infrastructure to explore and publish results on the time-varying behavior of isodensity surfaces that arise in computational fluid dynamic

[1] See the Repeatability section of the ACM SIGMOD 2011 home page: `http://www.sigmod2011.org/calls_papers_sigmod_research_repeatability.shtml`

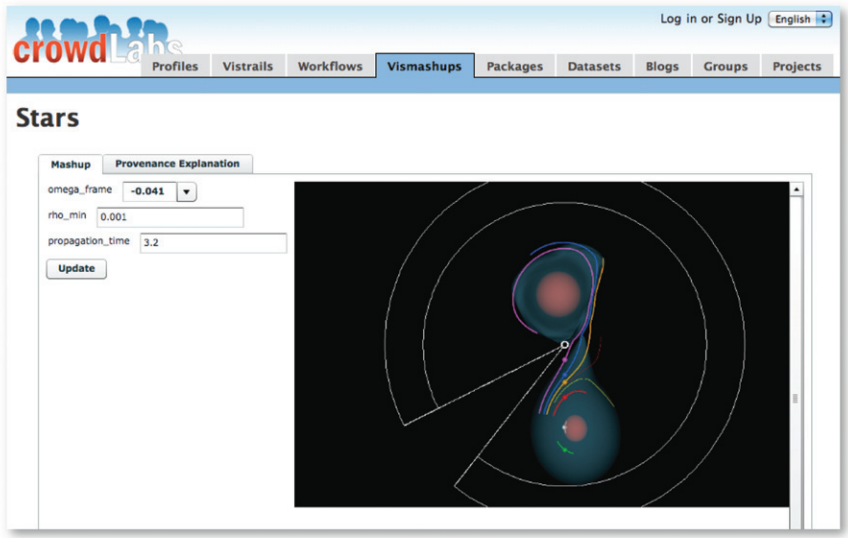

Figure 5: Interacting with the visualization of a binary star system simulation using VisMashup. Users change parameters on the left and see the resulting visualization on the right. This figure is *active* and linked to http://www.crowdlabs.org/vistrails/medleys/details/5/, where its provenance is also available.

(CFD) simulations of mass-transferring and merging binary star systems [24]. Visualization has helped researchers there discover new types of resonances in simulations. In exploring the binary mass-transfer simulations, scientists used parameter exploration to explore frame rotation frequencies which helped to identify the best measure of the star system's orbital period. They were also able to interactively compare the three-dimensional visualizations resulting from each parameter setting using a visual spreadsheet that allowed all cells to have synchronized views.

The visualizations and the ability to interact with them were essential to the evaluation of the simulations and to obtain insights that led to the discoveries. Giving readers the opportunity to access this information can help increase the impact of published articles and also serves an important educational purpose. To provide such access for readers, we have added not only Web-hosted enhancements, including VisMashup applications, but also downloadable workflows that allow users to fully explore and reconfigure results [25]. An expanded, executable, and interactive version of [24] is available at `http://www.vistrails.org/index.php/User:Tohline/IVAJ/Levels2and3`. Mashups created by VisMashup can be embedded in Web pages, including Wiki pages, allowing readers to comment on insight they have gained from examining new views or changing parameters. Figure 5 shows a example of a mashup that allows readers to interact with a visualization of a binary star system. Because such visualizations can be hosted on the Web, readers need not install additional software to casually browse results. However, with the ability to download full workflows, readers can achieve greater understanding or build new results by reusing the published workflows.

Packaging Experiments. As part of our effort to help authors create executable papers, we have developed a set of examples that serve as tutorials and cover different scenarios authors may be faced with [23, 14]. The Tuning example [23] shows how to wrap experiments that were originally designed without the use of a workflow system, and how the final workflows were encapsulated within a virtual machine. The WikiQuery example [14], besides illustrating how the authors' code can be wrapped for both local and remote execution also makes use of the persistence package (Section 3.1) to cache intermediate results and speed up the execution of workflows that analyze these.

5. Discussion

To the best of our knowledge, our infrastructure is the first approach that provides an end-to-end solution to the problem of computational repeatability. It relies on a provenance-based workflow system, VisTrails, that we have extended to meet the requirements of an executable paper life cycle. This infrastructure represents a first step towards simplifying the creation, review, and re-use of executable papers. Our long-term goal is to use it as the basis for a comprehensive system supporting executable publications, allowing uesrs to integrate different components to satisfy a wide range of requirements. There are several open problems we are currently investigating, including methods for improved testing and debugging, easier management of complex workflows and better support for remote resources.

Acknowledgments. We would like to thank all the developers that have contributed to the VisTrails system: Erik Anderson, Louis Bavoil, Clifton Brooks, Jason Callahan, Steve Callahan, Lorena Carlo, Lauro Lins, Tommy Ellkvist, Daniel Rees, Carlos Scheidegger, and Nathan Smith. The research and development of the VisTrails system has been funded by the National Science Foundation under grants IIS-1050422, IIS-0905385, IIS-0844572, ATM-0835821, IIS-0844546, IIS-0746500, CNS-0751152, IIS-0713637, OCE-0424602, IIS-0534628, CNS-0514485, IIS-0513692, CNS-0524096, CCF-0401498, OISE-0405402, CCF-0528201, CNS-0551724, the Department of Energy (SciDAC VACET and SDM centers, and SBIR 85821S08-II), National Institutes of Health (NCRR ARRA), and IBM Faculty Awards (2005, 2006, 2007, and 2008). E. Santos was partially supported by a CAPES/Fullbright fellowship. We also acknowledge support through a grant from the Army Research Office with funding from the DARPA OLE program.

References

[1] CDEpack, http://www.stanford.edu/~pgbovine/cdepack.html.
[2] C. Silva, J. Freire, S. P. Callahan, Provenance for visualizations: Reproducibility and beyond, IEEE Computing in Science & Engineering 9 (5) (2007) 82–89.
[3] J. Freire, E. Santos, C. Silva, Provenance-enabled data exploration and visualization with vistrails, in: SciDAC, Vol. 125, 2009.
[4] C. Silva, J. Freire, E. Santos, E. Anderson, D. Koop, Provenance-enabled data exploration and visualization, in: IEEE Visualization Conference, 2009, refereed tutorial.
[5] D. Koop, E. Santos, J. F. Bela Bauer, Matthias Troyer, C. T. Silva, Bridging workflow and data provenance using strong links, in: SSDBM, 2010, pp. 397–415.
[6] S. P. Callahan, J. Freire, C. E. Scheidegger, C. T. Silva, H. T. Vo, Towards provenance-enabling paraview, in: IPAW, 2008, pp. 120–127.
[7] J. Freire, C. Silva, S. Callahan, E. Santos, C. Scheidegger, H. Vo, Managing rapidly-evolving scientific workflows, in: International Provenance and Annotation Workshop (IPAW), LNCS 4145, Springer Verlag, 2006, pp. 10–18.
[8] J. Freire, D. Koop, E. Santos, C. T. Silva, Provenance for computational tasks: A survey, Computing in Science and Engineering 10 (3) (2008) 11–21.
[9] The ALPS project, http://alps.comp-phys.org.
[10] F. Alet, P. Dayal, A. Grzesik, A. Honecker, M. Körner, A. Läuchli, S. R. Manmana, I. P. McCulloch, F. Michel, R. M. Noack, G. Schmid, U. Schollwöck, F. Stöckli, S. Todo, S. Trebst, M. Troyer, P. Werner, S. Wessel, The ALPS Project: Open Source Software for Strongly Correlated Systems, Journal of the Physical Society of Japan 74S (Supplement) (2005) 30–35. doi:10.1143/JPSJS.74S.30.
[11] A. Albuquerque, F. Alet, P. Dayal, A. Feiguin, S. Fuchs, L. Gamper, , E. Gull, S. Gürtler, A. Honecker, R.Igarashi, M. Körner, A. Kozhevnikov, A. Läuchli, S. R. Manmana, M. Matsumoto, I. P. McCulloch, F. Michel, R. M. Noack, G. Pawłowski, L. Pollet, T. Pruschke, U. Schollwöck, F. Stöckli, S. Todo, S. Trebst, M. Troyer, P. Werner, S. Wessel, The ALPS project release 1.3: Open-source software for strongly correlated systems, Journal of Magnetism and Magnetic Materials 310 (2, Part 2) (2007) 1187–1193.
[12] B. Bauer, L. D. Carr, H. G. Evertz, A. Feiguin, J. Freire, S. Fuchs, L. Gamper, J. Gukelberger, E. Gull, S. Guertler, A. Hehn, R. Igarashi, S. V. Isakov, D. Koop, P. N. Ma, P. Mates, H. Matsuo, O. Parcollet, G. Pawlowski, J. D. Picon, L. Pollet, E. Santos, V. W. Scarola, U. Schollwöck, C. Silva, B. Surer, S. Todo, S. Trebst, M. Troyer, M. L. Wall, P. Werner, S. Wessel, The ALPS project release 2.0: Open source software for strongly correlated systems, accepted for publication in Journal of Statistical Mechanics: Theory and Experiment. Paper available at http://arxiv.org/pdf/1101.2646 and underlying workflows at http://arxiv.org/abs/1101.2646. (Jan. 2011). arXiv:1101.2646.
[13] Kitware, Paraview, http://www.paraview.org.
[14] H. Nguyen, Computational repeatability: The wikiquery case study, http://www.vistrails.org/index.php/WikiQuery, accessed on Feb 12, 2011.
[15] E. Santos, L. Lins, J. Ahrens, J. Freire, C. T. Silva, Vismashup: Streamlining the creation of custom visualization applications, IEEE Transactions on Visualization and Computer Graphics 15 (6) (2009) 1539–1546.
[16] D. Koop, C. Scheidegger, J. Freire, C. T. Silva, The provenance of workflow upgrades, in: IPAW, 2010, pp. 2–16.
[17] C. E. Scheidegger, H. T. Vo, D. Koop, J. Freire, C. T. Silva, Querying and creating visualizations by analogy, IEEE Transactions on Visualization and Computer Graphics 13 (6) (2007) 1560–1567.
[18] T. Ellkvist, L. Strömbäck, L. D. Lins, J. Freire, A first study on strategies for generating workflow snippets, in: Proceedings of the ACM SIGMOD Intenational Workshop on Keyword Search on Structured Data (KEYS), 2009, pp. 15–20.
[19] C. E. Scheidegger, D. Koop, E. Santos, H. T. Vo, S. P. Callahan, J. Freire, C. T. Silva, Tackling the provenance challenge one layer at a time, Concurrency and Computation: Practice and Experience 20 (5) (2008) 473–483.
[20] Guidelines for Research Integrity and Good Scientific Practice at ETH Zurich, http://www.vpf.ethz.ch/services/researchethics/Broschure, accessed January, 2011.
[21] I. Manolescu, L. Afanasiev, A. Arion, J. Dittrich, S. Manegold, N. Polyzotis, K. Schnaitter, P. Senellart, S. Zoupanos, D. Shasha, The repeatability experiment of sigmod 2008, SIGMOD Record 37 (1) (2008) 39–45.
[22] S. Manegold, I. Manolescu, L. Afanasiev, J. Feng, G. Gou, M. Hadjieleftheriou, S. Harizopoulos, P. Kalnis, K. Karanasos, D. Laurent, M. Lupu, N. Onose, C. Ré, V. Sans, P. Senellart, T. Wu, D. Shasha, Repeatability & workability evaluation of sigmod 2009, SIGMOD Record 38 (3) (2009) 40–43.
[23] Computational repeatability: Tuning case study, http://effdas.itu.dk/repeatability/tuning.html, accessed on Feb 12, 2011.
[24] J. Tohline, J. Ge, W. Even, E. Anderson, A customized python module for cfd flow analysis within VisTrails, Computing in Science Engineering 11 (3) (2009) 68–73. doi:10.1109/MCSE.2009.44.
[25] J. E. Tohline, E. Santos, Visualizing a journal that serves the computational sciences community, Computing in Science and Engineering 12 (2010) 78–81. doi:http://doi.ieeecomputersociety.org/10.1109/MCSE.2010.72.