(1) Overview

Title

PFHub: The Phase-Field Community Hub

Paper Authors

1. Wheeler, Daniel; (corresponding author) A
2. Keller, Trevor; A
3. DeWitt, Stephen J.; B
4. Jokisaari, Andrea M.; C
5. Schwen, Daniel; C
6. Guyer, Jonathan E.; A
7. Aagesen, Larry; C
8. Heinonen, Olle G.; D
9. Tonks, Michael R.; E
10. Voorhees, Peter W.; F
11. Warren, James A.; G

Paper Author Roles and Affiliations

A. Materials Science and Engineering Division,
Material Measurement Laboratory,
National Institute of Standards and Technology,
Gaithersburg, MD 20899 USA

B. Materials Science and Engineering Department,
University of Michigan,
Ann Arbor, MI 48109 USA

C. Fuel Modeling and Simulation Department,
Idaho National Laboratory,
Idaho Falls, ID 83415 USA

D. Argonne National Laboratory,
Lemont, IL 60439 USA

E. Department of Materials Science and Engineering,
University of Florida,
Gainesville, FL 32611

F. Department of Materials Science and Engineering,
Northwestern University,
Evanston, IL 60208 USA

G. Material Measurement Laboratory Office,
Material Measurement Laboratory,
National Institute of Standards and Technology,
Gaithersburg, MD 20899 USA

Abstract

Scientific communities struggle with the challenge of effectively and efficiently sharing content and data. An online portal provides a valuable space for scientific communities to discuss challenges and collate scientific results. Examples of such portals include the Micromagnetic Modeling Group (µMAG [1]), the Interatomic Potentials Repository (IPR [2, 3]) and on a larger scale the NIH Genetic Sequence Database (GenBank [4]). In this work, we present a description of a generic web portal that leverages existing online services to provide a framework that may be adopted by other small scientific communities. The first deployment of the PFHub framework supports phase-field practitioners and code developers participating in an effort to improve quality assurance for phase-field codes.

Keywords

phase-field; materials-science; jekyll-website; reproducible-science

Introduction

Generally, small scientific communities do not have the resources to build and host dedicated web infrastructure to support their varied content and data requirements. In particular, hosting and supporting a complex content management system (CMS) including web servers, web frameworks and databases requires a great deal of configuration and long term support and funding. Furthermore, a turnkey CMS solution may not meet requirements for most scientific communities that often use arcane data formats and require custom data displays along with client-side automation. The PFHub effort, instead of focusing on the CMS tool, focuses on customizing and delivering the client-side requirements whilst delegating back-end functionality to external services that provide dependable APIs [5].

PFHub is a community effort spearheaded by the Center for Hierarchical Materials Design at Northwestern University and the National Institute of Standards and Technology in support of phase-field code development. The current PFHub deployment [6] focuses on improving cross-collaboration between phase-field code developers and practitioners by providing a standardized set of benchmark problems [7,

8] and a workflow for uploading and comparing benchmark results from different phase-field codes (*e.g.*, FiPy [9], MMSP [10], MOOSE [11], PRISMS-PF [12]).

This paper presents the first deployment of the PFHub framework including its client-side focused design, how it employs external services and meta-data about the code base. The paper describes the relative ease with which other scientific groups might adapt the framework for their own purposes and deploy using the fully reproducible Nix environment [13].

Implementation and architecture

The PFHub framework provides a template for other small scientific communities to host custom content and integrate data from members of their community. The current deployment provides a facility for uploading, displaying and comparing results from benchmark problems supporting phase-field code developers and practitioners. However, the framework and overall philosophy are broadly transferable to other communities with some custom configuration and content generation. The framework uses the Jekyll static website generator [14] along with automated front-end processing to eliminate the need for a CMS [5], which is generally costly to maintain especially for small scientific communities with limited funding and staffing. The framework relies on the API, WebSocket and webhook infrastructure that underpins the modern web and allows external services to have full-duplex communication between servers and browsers. In particular, PFHub relies on GitHub's well maintained API and webhook functionality for external services (such as Travis CI [15] and Staticman [16]).

The workflow for uploading benchmark results relies on third party tools using the following steps, illustrated in Figure 1.

1. The users are first required to upload simulation outputs to an archival resource (*e.g.*, Figshare[1] [17]) configured with permissive cross-origin resource sharing (CORS).

2. The metadata summarizing each simulation is entered into a form on the website, including relevant details such as memory usage, run time and links to the data archived in the first step.

3. Upon submission, the Staticman app [16] submits the entered metadata as a pull request against the PFHub repository hosted by GitHub. The metadata is stored in a YAML file with a unique path in the repository.

[1]Certain commercial equipment, instruments, or materials (or suppliers, or software, ...) are identified in this paper to foster understanding. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

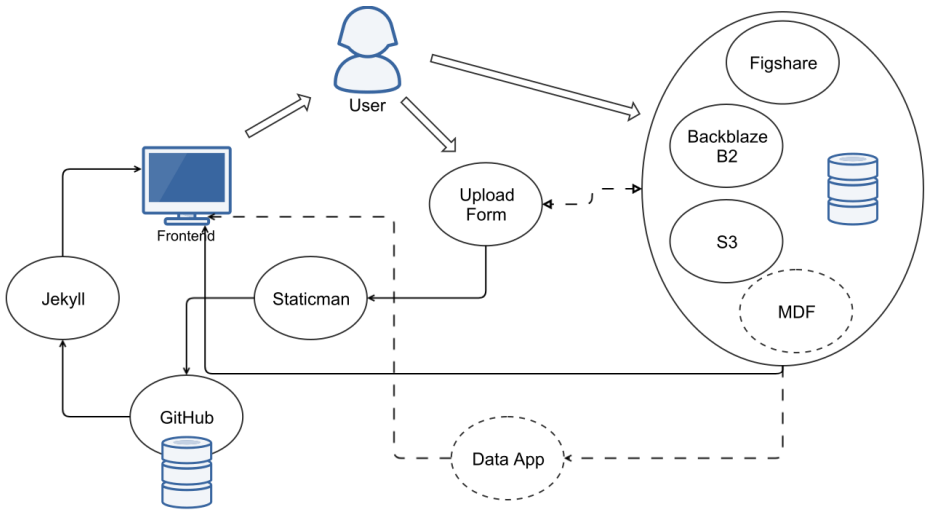

Figure 1: Schematic overview of the PFHub framework for building scientific research portals, simply.

4. Travis CI [15] performs linting on the submission and then launches a temporary version of the proposed website using Surge [18]. The PFHub admins can then examine the new submission and further changes can be made if necessary.

5. Once review has been completed to the satisfaction of both the uploading scientist and the website maintainers, the pull request is merged and served to the World Wide Web using a hosting service compatible with GitHub Pages.

A combination of Jekyll templates and CoffeeScript are used to access and download the data links in the submitted YAML files and then display the data in interactive plots on the website. The interactive plots are displayed using the Plotly JavaScript Graphing Library [19] as it provides a programmable interface and requires minimal configuration.

The current deployment of PFHub has benchmark specifications consisting of equations, narrative, plots and code samples, and are composed in Jupyter Notebooks. The Jupyter Notebooks are included as static objects in the website after translation into HTML using the nbconvert tool [20]. There are currently 7 benchmark problems each with a number of variations. At the time of writing there are 108 separate benchmark result uploads [6] submitted as pull requests and approved following code review to ensure compatibility with the website.

The combination of a central repository on GitHub for website source code and metadata with distributed data records on third-party archives avoids the complexity and administrative overhead of maintaining a live database and associated back-end application.

Quality control

The framework has a fully automated test recipe deployed on Travis CI with an environment built using the Nix Package Manager [13]. A fully automated test environment using continuous integration allows all developers and users to have common feedback on code updates and determine the compatibility of result uploads with the deployed website. The environment is pinned to a specific version of the Nix Packages Collection (Nixpkgs [21]), ensuring fully reproducible build and test phases as well as ensuring that the development and automated testing environments are identical. The full test recipe is outlined in a YAML file, `.travis.yml`, file stored in the repository [22] and consists of the following steps.

1. Build the Nix environment from a persistent cache on Travis CI, reducing the build time.
2. Run automated tests on Jupyter Notebooks using NBval [23] and Py.test [24].
3. Run validation tests on HTML files using HTMLProofer [25].
4. Lint and test front-end CoffeeScript using appropriate tools.
5. Display a temporary version of the website using Surge [18] for visual review.

(2) Availability

Operating system

The PFHub framework can be deployed on any platform supporting Nix, which includes all contemporary Linux and Mac OS X platforms. Since the framework is built with Jekyll and automated front-end processing, it can be deployed on GitHub's Pages infrastructure, which enables streamlined deployment without the need for any back-end infrastructure and, thus, is largely platform independent. For development purposes, a local installation of either Nix (on Linux or Mac) or Docker (on Linux, Mac or Windows) is required.

Programming language

PFHub is currently built and tested using the programming languages and versions outlined in Table 1.

Additional system requirements

There are no additional system requirements.

Table 1: PFHub programming languages and corresponding supported versions.

Language	Version
HTML	5
Jupyter Notebook	5.4.0
JavaScript	5
Nix	2.1.3
CoffeeScript	1.12.7
CSS	4

Dependencies

The entire environment can be built using the Nix Package Manager so the only required dependency is a functional Nix installation. The PFHub framework has over 2000 separate package dependencies using data from the Nix package manager. The full dependency graph for PFHub can be seen online [26].

List of contributors

This list is for contributors to the code base, but not those that have only uploaded output results to the website.

1. Wheeler, Daniel; A, `@wd15`
2. Keller, Trevor; A, `@tkphd`
3. DeWitt, Stephen J.; B, `@stvdwtt`
4. Jokisaari, Andrea M.; C, `@amjokisaari`
5. Schwen, Daniel; C, `@dschwen`
6. Guyer, Jonathan E.; A, `@guyer`

Also, see the contributors list on GitHub [27].

Software location:

Archive

Name: Zenodo
Persistent identifier: 10.5281/zenodo.2592705
Licence: NIST Software License [28]
Publisher: Daniel Wheeler
Version published: v0.1
Date published: 13/03/19

Code repository

Name: GitHub
Persistent identifier: `https://github.com/usnistgov/pfhub/tree/v0.1`
Licence: NIST Software License [28]
Date published: 13/03/19

Languages

English

(3) Reuse potential

The PFHub framework can be readily adopted by other communities that want to follow a CMS-free philosophy and use well supported external services. The website infrastructure can be cloned as a Git repository or downloaded as a ZIP archive and deployed with minimum effort. The mechanism for uploading data using Staticman can be easily configured for a new repository location. However, customizing the content of the website for a particular scientific community would require considerable effort. The current effort is closely integrated with GitHub, but future deployments could be modified to use other repository services such as GitLab or BitBucket.

The following steps are the more challenging aspects of deploying the framework for a new community.

- Upload new data upload specifications (*e.g.* the phase-field benchmarks in the PFHub website [6]) in a format that Jekyll can parse, *e.g.*, Jupyter Notebook, Markdown or HTML.
- Edit the `benchmarks.yaml` file to reflect the new upload requirements and describe the figures that need to be generated on the upload comparison pages.
- Edit the `_config.yml` file to update links and text related to the configuration for all aspects of the website.
- Update Markdown files to reflect the new content and mission of the scientific community.
- Remove data and files that are not required by the new community.

Currently, a deployment for a new community has not been attempted and, thus, the above steps need to be refined and documented.

Acknowledgments

We gratefully acknowledge input and guidance from all participants in the series of Phase-Field workshops held between 2015 and 2018 at the Center for Hierarchical Material Design [29].

Funding statement

D.W. wishes to acknowledge the Materials Genome Initiative funding allocated to the National Institute of Standards and Technology. S.J.D wishes to acknowledge funding from the U.S. Department of Energy, Office of Basic Energy Sciences, Division of Materials Sciences and Engineering under Award #DE-SC0008637 as part of the Center for PRedictive Integrated Structural Materials Science (PRISMS Center) at University of Michigan. P.W.V. is grateful for the financial assistance under the award 70NANB14H012 from the National Institute of Standards and Technology as part of the Center for Hierarchical Materials Design (CHiMaD).

Competing interests

The authors declare that they have no competing interests.

References

[1] *μMAG: The Micromagnetic Modeling Activity Group*. 2019. URL: https://www.ctcms.nist.gov/~rdm/mumag.org.html (visited on 03/11/2019).

[2] Lucas M. Hale, Zachary T. Trautt, and Chandler A. Becker. "Evaluating variability with atomistic simulations: the effect of potential and calculation methodology on the modeling of lattice and elastic constants". en. In: *Modelling and Simulation in Materials Science and Engineering* 26.5 (July 2018), p. 055003. ISSN: 0965-0393, 1361-651X. DOI: 10.1088/1361-651X/aabc05. URL: http://stacks.iop.org/0965-0393/26/i=5/a=055003?key=crossref.1d3c4d0412b252a7afd7d58da45f306d (visited on 03/15/2019).

[3] Chandler A. Becker et al. "Considerations for choosing and using force fields and interatomic potentials in materials science and engineering". en. In: *Current Opinion in Solid State and Materials Science* 17.6 (Dec. 2013), pp. 277–283. ISSN: 13590286. DOI: 10.1016/j.cossms.2013.10.001. URL: https://linkinghub.elsevier.com/retrieve/pii/S1359028613000788 (visited on 03/21/2019).

[4] *GenBank: NIH genetic sequence database*. URL: https://www.ncbi.nlm.nih.gov/genbank/ (visited on 03/26/2019).

[5] Dave Cole. *How We Build CMS-Free Websites*. July 2018. URL: https://medium.com/devseed/how-we-build-cms-free-websites-d7e19d94a0ff (visited on 03/11/2019).

[6] *PFHub: The Phase Field Community Hub*. URL: https://pages.nist.gov/pfhub (visited on 03/27/2019).

[7] Andrea M. Jokisaari et al. "Benchmark problems for numerical implementations of phase field models". In: *Computational Materials Science* 126 (2017), pp. 139–151. ISSN: 0927-0256. DOI: 10.1016/j.commatsci.2016.09.022. URL: http://www.sciencedirect.com/science/article/pii/S0927025616304712.

[8] Andrea M. Jokisaari et al. "Phase field benchmark problems for dendritic growth and linear elasticity". In: *Computational Materials Science* 149 (2018), pp. 336–347. ISSN: 0927-0256. DOI: 10.1016/j.commatsci.2018.03.015. URL: http://www.sciencedirect.com/science/article/pii/S092702561830168X.

[9] Jonathan E. Guyer, Daniel Wheeler, and James A. Warren. "FiPy: Partial Differential Equations with Python". In: *Computing in Science & Engineering* 11.3 (2009), pp. 6–15. ISSN: 1521-9615. DOI: 10.1109/MCSE.2009.52. URL: http://www.ctcms.nist.gov/fipy.

[10] Jason Gruber et al. *mesoscale/mmsp: Zenodo integration*. Mar. 2019. DOI: 10.5281/zenodo.2583258. URL: https://doi.org/10.5281/zenodo.2583258.

[11] Michael R. Tonks et al. "An object-oriented finite element framework for multiphysics phase field simulations". In: *Computational Materials Science* 51.1 (2012), pp. 20–29. ISSN: 0927-0256. DOI: 10.1016/j.commatsci.2011.07.028. URL: http://www.sciencedirect.com/science/article/pii/S0927025611004204.

[12] Stephen DeWitt et al. *prisms-center/phaseField: PRISMS-PF (Version 2.1.1)*. Mar. 2019. DOI: 10.5281/zenodo.2583308. URL: https://doi.org/10.5281/zenodo.2583308.

[13] *Nix Package Manager*. URL: https://nixos.org/nix/ (visited on 03/13/2019).

[14] *Jekyll*. URL: https://jekyllrb.com/ (visited on 03/14/2019).

[15] *Travis CI: Test and Deploy with Confidence*. URL: https://travis-ci.org/ (visited on 03/11/2019).

[16] *Staticman: Static sites with superpowers*. URL: https://staticman.net (visited on 03/11/2019).

[17] *Figshare*. URL: https://figshare.com/ (visited on 03/27/2019).

[18] *Surge: Static web publishing for Front-End Developers*. URL: https://surge.sh/ (visited on 03/11/2019).

[19] Plotly Technologies Inc. *Collaborative data science*. 2015. URL: https://plot.ly (visited on 03/20/2019).

[20] Thomas Kluyver et al. "Jupyter Notebooks – a publishing format for reproducible computational workflows". In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. Ed. by F. Loizides and B. Schmidt. IOS Press. 2016, pp. 87–90.

[21] *Nix Packages Collection*. URL: https://github.com/NixOS/nixpkgs (visited on 04/05/2019).

[22] *Travis CI Recipe for PFHub*. URL: https://github.com/usnistgov/pfhub/blob/master/.travis.yml (visited on 03/13/2019).

[23] *NBVal: Py.test plugin for validating Jupyter notebooks*. URL: https://github.com/computationalmodelling/nbval (visited on 03/14/2019).

[24] *Py.test*. URL: https://docs.pytest.org/en/latest/ (visited on 03/14/2019).

[25] *HTML Proofer*. URL: https://github.com/gjtorikian/html-proofer (visited on 03/14/2019).

[26] *PFHub Dependency Graph*. URL: https://github.com/usnistgov/pfhub/blob/master/_publications/jors/dependencies.svg (visited on 03/19/2019).

[27] *List of PFHub Contributors*. URL: https://github.com/usnistgov/pfhub/graphs/contributors (visited on 03/13/2019).

[28] *NIST Software License*. URL: https://www.nist.gov/director/copyright-fair-use-and-licensing-statements-srd-data-and-software (visited on 03/13/2019).

[29] *CHiMaD Phase-Field Workshops*. URL: https://pages.nist.gov/pfhub/wiki/workshops/ (visited on 03/14/2019).