(1) Overview

Title

PFHub: The Phase-Field Community Hub

`https://pages.nist.gov/pfhub`

Paper Authors

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

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. Argonne National Laboratory,
Lemont, IL 60439 USA

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

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

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

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

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

Abstract

An online portal provides a valuable space for scientific communities to summarize a shared challenge, collect attempts at a solution, and present a quantitative comparison of past attempts in a compelling way. An exemplar of such a portal is μMAG [1]. The reusable PFHub framework leverages existing online services to build a static portal website that is considerably easier to deploy and maintain without sacrificing content or scope. 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

The phase-field method (PFM) describes material interfaces at the mesoscopic scale between atomic scale models and macroscale models. The PFM is well established and there are an assortment of code frameworks (*e.g.*, MOOSE [21], PRISMS-PF [4], FiPy [6], MMSP [5]) available for solving the wide variety of mesoscale phenomena (*e.g.*, dendritic growth, spinodal decomposition, grain growth). However, phase-field research groups often develop codes in isolation and do not support or distribute the code bases to the wider community. PFHub is a community effort spearheaded by the Center for Hierarchical Materials Design at Northwestern University and the National Institute of Standards and Technology to support the development of phase-field codes. In particular, the PFHub effort focuses on improving cross-collaboration between phase-field code developers and practitioners by providing a standardized set of benchmark problems [10, 11] along with a web framework for uploading and comparing benchmark results from different codes.

Implementation and architecture

The PFHub website provides a facility for uploading, displaying and comparing results from the benchmark problems. The website uses the Jekyll static website

generator [9] along with automated frontend processing to eliminate the need for content management systems [3], which are generally costly to maintain especially for small scientific communities with limited funding and staffing.

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

- The users are first required to upload simulation outputs to an archival resource (*e.g.*, Figshare [1]) configured with permissive cross-origin resource sharing (CORS).
- 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.
- Upon submission, the Staticman app [19] submits the entered metadata as a GitHub pull request to the PFHub GitHub repository. The metadata is stored in a YAML file with a unique path in the repository.
- Travis CI [23] performs linting on the submission and then launches a temporary version of the proposed website using Surge [20]. The PFHub admins can then examine the new submission and further changes can be made if necessary.
- 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 [8] as it provides a programmable interface and requires minimal configuration.

The benchmark specifications consist 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 [12].

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

[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.

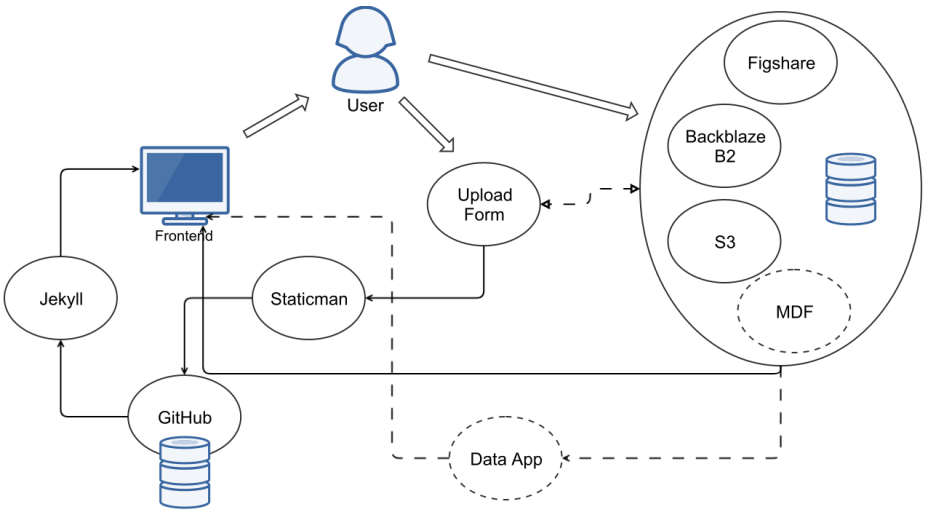

Figure 1: Schematic overview of the PFHub framework for building scientific research portals, simply.

back-end application. The PFHub infrastructure provides a template for other small scientific communities to host custom content and integrate data from members of their community.

Quality control

The framework has a fully automated test recipe deployed on Travis CI with an environment built using the Nix Package Manager [16]. The environment is pinned to a specific version of the Nixpkgs repository 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 `.travis.yml` file stored in the repository [22] and consists of the following steps.

- Build the Nix environment from a cached storage reducing the build time.
- Run automated tests on Jupyter Notebooks using NBval [14] and Py.test [18].
- Run validation tests on HTML files using HTMLProofer [7].
- Lint and test front-end Coffeescript using appropriate tools.
- Display a temporary version of the website using Surge [20] 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.

Programming language

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

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

Additional system requirements

There are no additional system requirements.

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 [17].

List of contributors

1. Wheeler, Daniel; A, `@wd15`
2. Keller, Trevor; A, `@tkphd`
3. DeWitt, Stephen J.; E, `@stvdwtt`
4. Jokisaari, Andrea M.; D, `@amjokisaari`

5. Schwen, Daniel; D, `@dschwen`
6. Guyer, Jonathan E.; A, `@guyer`

See the contributors list on GitHub [13].

Software location:

Archive

Name: Zenodo
Persistent identifier: 10.5281/zenodo.2592705
Licence: NIST Software License [15]
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 [15]
Date published: 13/03/19

Languages

English

(3) Reuse potential

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

- Upload new benchmark definitions 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 [2].

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.

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

[3] 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).

[4] 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`.

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

[6] 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.

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

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

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

[10] 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.

[11] 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.

[12] 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.

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

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

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

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

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

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

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

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

[21] 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.

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

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