Expanding user communities with HPC Carpentry

Alan Ó Cais∗
Jülich Supercomputing Centre
Jülich, Germany
a.o.cais@fz-juelich.de

Peter Steinbach
Scionics Computer Innovation GmbH
Dresden, Germany
Max Planck Institute of Molecular Cell Biology and Genetics
Dresden, Germany
steinbach@scionics.de

ABSTRACT
Adoption of HPC as a research tool and industrial resource is a priority in many countries. The use of data analytics and machine learning approaches in many areas also attracts non-traditional HPC user communities to the hardware capabilities provided by supercomputing facilities. As a result, HPC at all scales is experiencing rapid growth of the demand for training, with much of this at the introductory level.

To address the growth in demand, we need both a scalable and sustainable training model as well as a method to ensure the consistency of the training being offered. Adopting the successful training model of The Carpentries (https://carpentries.org/) for the HPC space provides a pathway to collaboratively created training content which can be delivered in a scalable way (serving everything from university or industrial HPC systems to national facilities).

We describe the ongoing efforts of HPC Carpentry to create training material to address this need and form the collaborative network required to sustain it. We outline the history of the effort and the practices adopted from The Carpentries that enable it. The lessons being created as a result are under active development and being evaluated in practice at sites in Europe, the US and Canada.

KEYWORDS
Carpentries, Software Carpentry, Education, Training, HPC

1 INTRODUCTION
Many countries are now spending substantial budgets on high-performance computing (HPC) related research initiatives. These initiatives are creating computational laboratories of unprecedented scale. At the same time, big data analytics, cloud computing and deep neural networks are influencing the development of HPC to the extent that the possible future convergence of HPC and big data applications is being discussed [10]. Cross-domain interest in Deep Learning alone is driving new sets of users to seek out access to the latest hardware, and in many cases this hardware is to be found in the national and regional supercomputing facilities. The explosion of interest in all of these fields brings with it many challenges, not least of these is providing researchers with adequate training so they can effectively and efficiently leverage these computational laboratories for their research.

To highlight a specific example of this growth, we consider EuroHPC (https://eurohpc-ju.europa.eu) which contains 25 participating EU states. In EuroHPC, each of the participant states is expected to have an HPC Competence Centre which will provide HPC services to industry, academia and public administrations. These HPC Competence Centres will be a gateway into the European HPC landscape and, in many cases, the first landing point for HPC interest in those countries. Since it is an European initiative one must strive for consistent levels of service across all states, despite greatly varying experience of the HPC domain. From a training perspective, how does one achieve this? Addressing this challenge is of critical importance to the long-tail impact of the investment in EuroHPC.

In this paper we will address the initial training requirements of a new user who is freshly exposed to HPC resources. Within academic research there is always a constant flow of early career researchers entering the field and accessing resources at all levels of the hardware pyramid. For this reason, we see the ‘HPC novice’ profile as something that is relevant across the spectrum of HPC facilities, from institutional resources all the way up to national and international facilities.

A scalable, collaborative training model is an effective and sustainable way to tackle large increases in demand for “HPC novice” training. We propose to adopt and adapt the model developed by the Software Carpentry initiative [11] and apply it to the novice HPC learner. In Section 2, we introduce aspects of that model and why it has the potential to map well to the HPC space. In Section 3, we outline some of the training material design principles and how they influence the creation process. In Section 4, we look at two distinct evaluation processes, a review process that happens during material creation and learner evaluations that occur during/after training events. Finally, in Section 5, we consider future development efforts in light of progress and outcomes that have occurred to date.
2 SOFTWARE CARPENTRY AND HPC CARPENTRY

In the lessons-learned review of Software Carpentry initiative [12], the author draws a distinction between the "minority who do high-performance computing" and other scientists who use computing as a research tool. However, many who have been involved with training of novice HPC users will recognise that the philosophy of Software Carpentry to teach "researchers the computing skills they need to get more done in less time and with less pain" is just as relevant to the HPC novice, but that the scope of necessary skills is wider. The goal of HPC Carpentry is to target the skills gap as well as the technical and conceptual barriers faced by the HPC novice as they transition from desktop/server computation to HPC resources.

At its core, Software Carpentry is a volunteer project dedicated to teaching basic computing skills to researchers. This is done by treating lessons the same way you would an open source software project: collaboratively created and free to access or use. This approach is inherently scalable as the lesson itself is a shared resource. Most importantly, Software Carpentry provides training on modern research in education, and associated evidence-based teaching practices [3]. In short, the instructor is taught how to deliver training material effectively and the community provides them with the high quality training content that they will teach. Just as importantly though, a shared lesson and a shared approach to teaching opens up the possibility to create a community around that material. Such a community is a resource in and of itself, and provides a platform for further collaboration on more complex training topics.

HPC Carpentry seeks to crystallise such a community in the HPC domain. This approach resembles the same process that has already been undertaken with the Data Carpentry and Library Carpentry initiatives2.

2.1 The Importance of Collaboration

For some time, there has been significant interest among the HPC training community in the Software Carpentry approach to generating and delivering training content. There have already been two distinct efforts to develop the type of HPC novice material that is being considered here:

- HPC in a day (https://psteinb.github.io/hpc-in-a-day/),

Further interest has included a Birds of a Feather session at SC17 [9].

HPC Carpentry is not, currently, a formal part of the Carpentries but the current development drive is being carried with the knowledge and participation of the Carpentries. At CarpentryCon 2018, HPC Carpentry had 2 sessions [4, 7] (each with ~40 participants), where much of the discussion centred around how to merge existing efforts and form a single group of collaborators to drive the lesson development forward. The creators and maintainers of the previously listed novice material were central to this discussion and have agreed to engage in a unification effort under the HPC Carpentry umbrella. This has resulted in the creation of a

3 LESSON DESIGN

The "rules" that govern lesson integration and development are reproduced from [6] in Fig. 2.

One of the first things the interested parties did was to clarify our audience by outlining a set of Learner Profiles that are representative of the intended audience (Rule 1)4. Furthermore, we restricted the training window to be a single day since this is what many sites already do in practice for the targeted level of this material. In addition, there was lengthy discussion about what should, and should not, be included in the lesson. In particular, there is the issue of what should be considered as required prerequisite knowledge.

The most significant potential prerequisite is the UNIX shell, whether it should be incorporated or not was intensely discussed.

[318x528 to 563x708]

Figure 1: Collaborative open lesson development (reproduced from [6]).
Teaching UNIX shell fundamentals is already core material within Software Carpentry [2] but there are some additional topics (such as ssh sessions, for example) that are specific to remote computing. Having introduced the restriction that there be only one full day to address all the material, the decision was made to split the lesson in two equal parts: the first part dealing with the shell (in the context of remote computing) and the second part with the introduction to working with an HPC resource. Our decision aligns with Rule 2 about modularity of lessons, it provides the instructor with an opportunity to omit the first part if the prerequisite knowledge exists among the participants. In addition, every lesson is structured into chapters. For example:

1. Why use HPC?
2. Logging in to a cluster with ssh
3. Submitting jobs
4. Environment modules
5. Transferring data

By virtue of creating chapters with high independence and low to no overlap, an instructor is free to skip or remove portions of the content without loss of material coherence.

To respect Rule 3 as regards best practices, we take the Carpentries Instructor Training [1] as our standard. We begin from the end: what do we want our learners to be able to do by the end of the lesson? We work backwards from this to create a concept map and give the lesson structure. The map also helps us decide where to introduce formative assessments to help learners commit each concept to memory. Formative assessment ties into Rule 7 of Fig. 2 to infer the lesson audience knowledge intake for a given topic. Furthermore, the HPC Carpentry community strives to implement continuous feedback loops during a workshop, as has become a standard with the Carpentries. This can be considered as evidence for Rule 10, but also emphasises the challenge within HPC Carpentry to implement Rule 1 to 9 in such an environment. It is an ongoing effort to dissect clearly the central topics that HPC carpentry wants to focus on and what motivates them.

3.1 Lesson Portability

A key technical issue for lesson portability is the fact that the learner environment can vary significantly between HPC sites. For example, a truly portable lesson should be possible to be configured for different resource managers, scheduler queue configurations, MPI launchers, …

Since the Carpentries lessons are delivered via GitHub pages using Jekyll, we leverage liquid templates to enable portability. For example, in the YAML configuration file for the Jekyll website, we can set a variable with:

```
scheduler: "slurm"
```

which can then be referenced throughout the lesson material with:

```
{{ site.scheduler }}
```

Lesson portability then means adding a YAML configuration file appropriate for the target site. This approach is based on that originally taken in [8].

Figure 2: 10 rules for collaborative lesson development (reproduced from [6]).
It should be noted that this also influences lesson design since we must make an additional effort to avoid referencing specific features of tools that may not be available in all of the expected configurations. While this increases the burden for contribution and makes the lesson source code harder to read, we choose to embrace the heterogeneity of HPC as it exists and thus enlarge the number of possible lesson users.

4 EVALUATION

There are two different types of evaluation of the lesson material: the evaluation of contributions to the lessons themselves and the evaluation from learners in workshops where the lessons are delivered.

In the case of evaluation of contributions, the contributions themselves are made via Pull Requests\(^\text{13}\) to the lesson repositories. These contributions are reviewed by lesson maintainers and by any interested parties who wish to engage in discussing the contribution. The intention is that this process is not only open (anyone can make a Pull Request) but also provides a platform for exposing new contributors to the design principles used in the lessons. It also gives us a mechanism by which we can follow Rule 4 of Fig. 2: we can actively encourage new contributors and incorporate them into the community.

From the learners, there are three methods of gathering feedback\(^\text{13}\):

1. pre- and post-workshop surveys;
2. minute cards - anonymous notes gathered at lunch time and the end of the day which have one positive and one negative comment;
3. the "one up, one down" technique - the instructor asks the learners to alternately give one positive and one negative point about the day, without repeating anything that has already been said.

Summaries of these evaluations are communicated during the meetings of the lesson collaborators. The surveys help us to assess whether the learning goals have been achieved, while the minute cards and the "one up, one down" technique are usually an evaluation of the instructor as much as the lesson content. Where appropriate, matters arising from the evaluations are raised as issues in the relevant repository.

5 CONCLUSIONS

The “HPC novice” lessons https://hpc-carpentry.github.io/hpc-shell and https://hpc-carpentry.github.io/hpc-intro are both undergoing significant development currently. These lessons have been delivered by a number of people within the HPC Carpentry community and initial feedback has been largely positive. There has not, as yet, been a more objective evaluation of the lessons in the HPC Carpentry repository. However, the experience has been sufficient to encourage new contributors to the lesson repositories. The lesson repositories on GitHub are the best place to keep track of recent developments and contributions\(^\text{14}\).

Much in the same way that Software Carpentry developed, there is a desire from collaborators to ultimately move beyond novice lessons to more advanced topics. Priority has been given to novices content because it is common ground before we approach the branching of learner profiles expected when we consider more advanced topics (software users, software developers, hardware-specific training, domain-specific training…).

ACKNOWLEDGMENTS

Alan Ó Cais acknowledges support from the European Union’s Horizon 2020 research and innovation program, under grant agreement No. 676531 (project E-CAM) and grant agreement No. 823964 (project FocusCoE).

REFERENCES


\(^{13}\)https://help.github.com/en/articles/about-pull-requests


