ICEBERG: Imagery Cyber-infrastructure and Extensible Building blocks to Enhance Research in the Geosciences. (A Research Programmer's Perspective)

BRADLEY D. SPITZBART and HEATHER J. LYNCH, Stony Brook University
MATTEO TURILLI and SHANTENU JHA, Rutgers University

The ICEBERG (Imagery Cyber-infrastructure and Extensible Building blocks to Enhance Research in the Geosciences) project (NSF # 1740595) aims to (1) develop open source image classification tools tailored to high-resolution satellite imagery of the Arctic and Antarctic to be used on HPDC resources, (2) create easy-to-use interfaces to facilitate the development and testing of algorithms for application to specific geoscience requirements, (3) apply these tools through use cases that span the biological, hydrological, and geoscience needs of the polar community, and (4) transfer these tools to the larger non-polar community. Here we report on the project status and lessons learned.

CCS Concepts: • **Software and its engineering → Programming teams**; • **Applied computing → Environmental sciences**.

Additional Key Words and Phrases: cyberinfrastructure, geosciences, high resolution satellite imagery

ACM Reference Format:
Bradley D. Spitzbart, Heather J. Lynch, Matteo Turilli, and Shantenu Jha. 2020. ICEBERG: Imagery Cyber-infrastructure and Extensible Building blocks to Enhance Research in the Geosciences. (A Research Programmer's Perspective) . In *Practice and Experience in Advanced Research Computing (PEARC '20), July 26–30, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3311790.3396640

1 INTRODUCTION

Satellite imagery is rapidly transforming the way we see the planet, including our ability to study the most remote parts of the Arctic and Antarctic. Satellite imagery can help us map networks of rivers, study changes in the flow and thickness of glaciers, identify rock and soil types, and even find animals like penguins and seals. Because the availability of imagery in polar areas has increased rapidly over the last decade, we are now faced with a challenge: How do we scale-up the scientific discoveries that have been enabled by satellite imagery to larger spatial scales?

2 DESIGN

This project was inspired by several events sponsored by the Research Coordination Network (RCN; NSF Award 1542110) on high-performance computing in the polar sciences, including a workshop held at the University of Minnesota's Polar Geospatial Center (PGC) in June 2016 and two hackathons for the polar science community in July 2016 and August 2017. The goal of the RCN's 2016 PGC workshop, which included 20 experts from polar geosciences and high performance computing, was to identify current barriers to the integration of high performance computing and satellite imagery-enabled science. There was broad consensus that individual sciencePIs did not want the responsibility or hassle of moving and storing high-resolution imagery. Local campus infrastructure often makes the transfer, storage and computing of imagery from the PGC impractical. While custom analyses on small numbers of images are feasible, scaling up an existing workflow to large portions of the imagery archive represents a significant barrier to the community. Under the current system, imagery needs to be transferred from the PI's local system to a high-performance machine, any analysis pipeline involving licensed software needs to be re-written using open source tools, and tasks need to be scheduled and distributed efficiently among compute nodes. That workshop identified several core needs within the polar science community that could be simultaneously addressed by the development of software tools for high-throughput processing of high resolution commercial satellite imagery. Despite varied programming expertise within the polar science community, the details of executing large-scale distributed computing processes creates a real barrier to the kinds of iterated problem solving typical for scientific discovery. In terms of priority imagery-enabled science questions, there was remarkable convergence around the need for open-source image classification algorithms to automatically identify target features. The ICEBERG (SPELL OUT HERE) project, which was designed around a series of diverse use cases all requiring similar computational infrastructure, was initiated in late 2017 under the auspices of funding from the NSF EarthCube program (<Award Number>). The collaborations, roles, and workflows were deliberately designed as a novel and efficient means to facilitate domain researchers' understanding and access to cyberinfrastructure and high performance computing to expand their science.

ICEBERG then began in fall 2017 and was fully formed by spring 2018. The collaborations, roles, and workflows were deliberately designed as a novel and efficient means to facilitate domain researchers' understanding and access to cyberinfrastructure and high performance computing to expand their science.

2.1 Collaborations

Our proposed cyberinfrastructure (CI) was a direct response to the needs of the polar and geoscience communities and our solution developed through an extended period of requirements scoping. The selection of the ICEBERG team was driven by a desire to have use cases that spanned the types of problems faced by the geosciences community, even if the specific questions to be addressed were focused on polar science specifically.

Several core applications were identified at the outset of the ICEBERG projectas both scientifically important and with similar overlapping computational needs. All of these initial use cases involved high-resolution commercial imagery, though at its outset, only one of these projects involved the explicit application of computer vision tools such as Convolutional Neural Networks (CNNs). In the early development of ICEBERG, however, all but one of our use cases ended up involving CNNs for feature detection and classification, which generated even greater synergy between the different applications and the opportunity to focus, at least initially, on building libraries and cyberinfrastructure tailored for computer vision and other GPU-intensive tasks.

It must, therefore, be emphasized that our proposed cyberinfrastructure (CI) was a direct response to the needs of the polar and geoscience communities and our solution developed through an extended period of requirements scoping. The selection of the ICEBERG team was driven by a desire to have use cases that spanned the types of problems faced by the geosciences community, even if the specific questions to be addressed were focused on polar science specifically.

ICEBERG is now a collaboration of 5 U.S. institutions including Stony Brook University, Rutgers University, Northern Arizona University, the University of California - Santa Barbara, and the University of Colorado Boulder. Additional use cases have been supported in the latter phases of ICEBERG; these new use cases have engaged teams from the University of Connecticut, University of Alaska - Fairbanks, University of South Florida, and Boise State University. There are 12 - 15 individuals participating in the project at any given time. Other affiliates include the PGC, which provides imagery access and processing support, as well as NSF's XSEDE supercomputing resources.

2.2 Roles

Roles within the organization are split among Principle Investigators (PIs), Cyberinfrastructure (CI) Architect, CI team, Research Programmer, Co-PIs, and Developers, with many overlapping facets and responsibilities.

The PIs are Lynch, heading the science teams and Jha, leading the cyberinfrastructure side. They are responsible for the overall direction of the project, communication and reporting with funding sources, and recruiting new projects.

The CI architect (Turilli) oversees the graduate student CI team. They advise on best practices and lead research involving the bridge between distributed and high performance computing, designing middleware for scientific cyberinfrastructures, and supporting domain-specific scientific workflows. This role brings extensive experience and knowledge for solving technical, implementation, and logistical issues.

The CI team are the developers of RADICAL-Cybertools (RCT) [13] [4] at Rutgers University which allow to transition desktop image processing to large-scale processing on high performance computing (HPC) platforms. Specifically, the CI team are experts at converting linear codes that process one image at a time into parallelized tasks that process many images at once on multi-core systems. RADICAL-Cybertools comprise three main software systems: EnsembleToolkit (EnTK) [1], RADICAL-Pilot (RP) [9] and RADCIAL-SAGA (RS) [10]. When used together, these systems enable the specification and execution of workflows (EnTK) via a pilot-based workload management system (RP) on diverse HPC platforms (RS). RADICAL-Cybertools offers three main benefits: (1) explicit specification of adaptive ensemble applications; (2) high-throughput concurrent execution of heterogeneous tasks (1 minute to 24 hours long, single/multi core/GPU, OpenMP, MPI); and (3) concurrent and sequential task executions on the same pilot. The CI team is also responsible for performance testing, profiling, and optimization [11].

The Research Programmer (Spitzbart) is the link between the science and CI teams. This role oversees the day-to-day operations of the ICEBERG team, including keeping the general timeline and schedules. It requires a basic understanding of the overall science goals and detailed experience with the code development and algorithm implementations. In the case of ICEBERG, knowledge of image analysis, GIS and Remote Sensing concepts and a good network within the HPC community have been very helpful. They first work with the Co-PIs to learn the science drivers, previous work, current status, and end products of their projects. Then, works closely with the developers on software engineering methodology and any technical issues that arise. Next, the research programmer communicates with the CI team to be sure the algorithms are properly translated for large-scale processing in order to achieve the science goals and to facilitate the dialogue between the CI and science teams. Finally, at all times they keep the PIs apprised of progress and difficulties. At various stages this role can be described as trainer, motivator, and process checker.

The Co-PIs oversee their individual research programs at their own institutions. They each lead one of the use cases described in section 3.

The developers generate the core algorithms and code for their own projects, with one or two members devoted to each use case. They consist of postdocs, graduate students, and undergraduates.

2.3 Workflows

In keeping with NSF policy and good open source doctrine all code is developed in public GitHub repositories (https://github.com/iceberg-project). ICEBERG uses git-flow [5] as the development and branching model, with some simplifications. All development (i.e. commits to git) should be related to an issue or pull request. Any active development is done to branches based on a specified procedure. When they are stable and tested, a pull request is created towards the devel branch. At least one reviewer should check the pull request, before it is merged. GitHub allows protecting branches from direct commits. The master branch should be protected and commits should happen only through approved pull requests with at least one reviewer accepting. Along the way, Travis CI [3] is deployed to catch bugs and maintain good style. When a significant development phase is finished, and tested, a pull request will be submitted from devel to master. When that is merged we will consider everything released and ready for executing production runs.

3 USE CASES

There are three major use cases that are being developed initially for ICEBERG, which collectively cover many possible applications in geosciences and biology. We anticipate processing more than 300 TB of imagery data, using 100,000 core hours of CPU time and 15,000 GPU hours.

3.1 Biological - Seals and Penguins

Antarctic pack-ice seals, a group of four species of true seals (Phocidae), play a pivotal role in the Southern Ocean foodweb as wide-ranging Antarctic krill (Euphausia superba) predators. This project aims to automate pack-ice seal surveys using a combination of remote sensing, computer vision, seal ecology and HPC. Apart from being cheaper and safer when compared to aerial imagery, the scale at which we can survey pack-ice seals with remote sensing imagery will fill an important information gap in Antarctic ecology. This study also showcases the ever-growing benefits of incorporating AI into ecological sampling designs. We provide a detection algorithm to extract the location of seals from high-resolution imagery. This algorithm was developed by convolutional neural network training to detect and count seal haul-outs. This is beneficial, as a comprehensive pack-ice seal census and monitoring will provide key information on the health and evolution of the Southern Ocean ecosystem.

We also provide a detection algorithm (ICEBERG-Penguins) to extract the location of penguin colonies from high-resolution imagery. This algorithm was developed by convolutional neural network training to detect the extent of guano patches indicating penguin colonies. This is beneficial, as a penguin colony census and monitoring will provide key information on the health and evolution of the Antarctic ecosystem.

3.2 Geological - LandCover

ICEBERG-LandCover is a pipeline for automated processing of satellite imagery, automated detection and removal of snow, ice, water, and shadows from the scene, automated atmospheric characterization and removal, and automated "stretching" of the scenes to provide spatial coverage of surveyed area, reasonable estimates on atmospheric contributions, and comparisons to a spectral library of known geologic materials.

3.3 Hydrological - Rivers

We provide a classification algorithm for ice surface features from high-resolution imagery. This algorithm was developed by convolutional neural network training to detect regions of large and small rivers and to distinguish them from crevasses and non-channelized slush. We also provide a detection algorithm to extract polyline water features from the classified high-probability river areas.

4 IMPLEMENTATION

This section describes key aspects of a successful program. Many of these follow from standard best practices while others came about through trial and error.

4.1 Developer Training

Most of the researchers involved with the ICEBERG project are domain scientists with no formal programming training and little-to-no experience with software development. While this presents some challenges, it also provides the opportunity to enforce a level of code simplicity and uniformity across the various use cases.The Research Programmer and CI team provide initial training and ongoing support in topics such as Linux, Python, command line, Github, HPC using Software Carpentry modules [2] and other resources. While the geographically dispersed nature of the team made it difficult to use pair-programming or other techniques for co-production and training, regular screen sharing sessions facilitated learning. As is perhaps not uncommon, most of the code development was undertaken by Ph.D. students and postdocs, who worked closely with the research programmers and CI middleware developers. PIs leading the application research groups were less involved in the details of the software development and slower to learn the nuances of collaborative software development.

4.2 Communication

Given the wide geographical dispersion of the ICEBERG team members, regular consistent communication has been key. We accomplish this through bi-weekly "all-hands" Zoom meetings involving the entire ICEBERG team and alternating bi-weekly meetings of the CI team alone. A typical "all-hands" meeting agenda consists of (1) a review of the activities of each group over the last two week period, (2) a 30 minute presentation of project status from one use-case, an affiliate such as PGC, or a guest interested in using ICEBERG in their work, and (3) action items for the next two weeks. In person meetings of the CI team occur approximately every three months.

The Research Programmer is available at all times for Zoom calls with Co-PI's, developers, and/or the CI team. We use Slack rather than email for written discussions to maintain a common record in one place and the ability to easily add or subtract team members from the channels involving each use case.

4.3 Documentation

Each use case is required to complete a Software Requirements Specification (SRS) that lays out the functional and non-functional requirements including descriptions of user interfaces, inputs, outputs, and default parameters. This is used by the CI team to pre-plan their approach before seeing any code.

Also required from each use case is a schematic of the stages and tasks implemented in the code. This is formulated in conjunction with the developers and the Research Programmer. Figure 1 shows an example architecture split into stages and tasks for efficient ensemble execution.

During development we make heavy use of GitHub's "issue" tabs to track bugs, enhancements, or other requests. In fact, it is our policy that all commits should be in response to an open issue (of course, new issues can be opened directly before a commit then closed immediately). This has been valuable to provide a long history of all changes.

User documentation will consist of command line help pages and web resources linked to the ICEBERG web portal [7].

4.4 Challenges

One initial challenge was onboarding each domain science teams to the use of Linux and GitHub. Most project members were only familiar with Windows or Mac systems and had not been exposed to version control. The transition took some time though thoughtful and patient training and trust building and some concessions were made on, for example, the stricter Travis CI rules. All now agree that the structure, while cumbersome to learn at first, makes the work easier

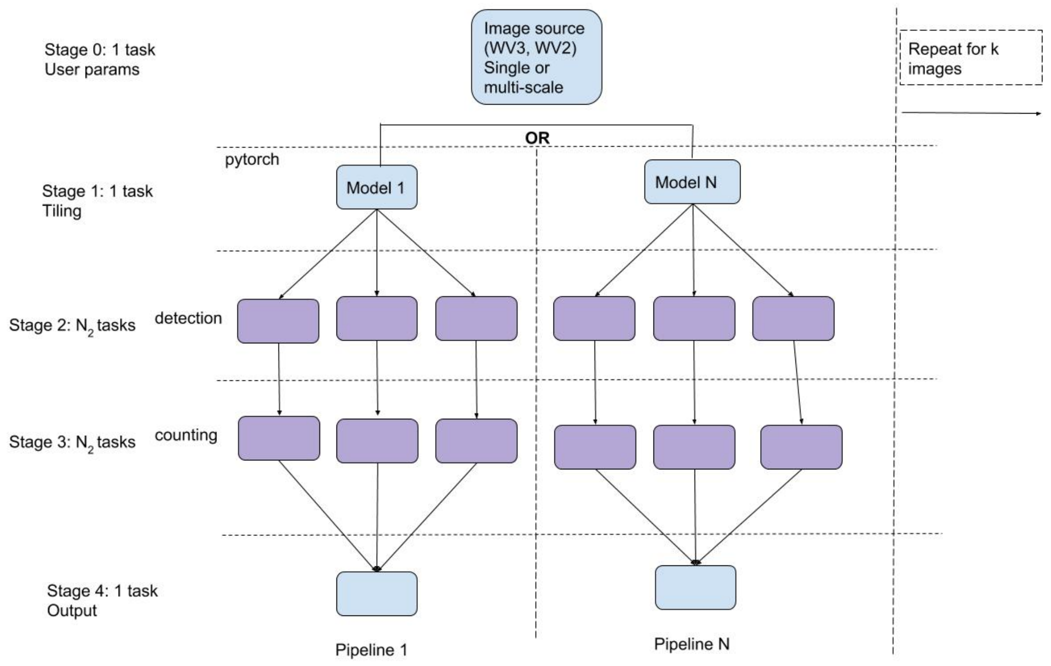

Fig. 1. An example schematic of the Seals execution architecture.

and this experience will have a lasting impact on the project management for all the domain science participants involved.

Another challenge faced by the ICEBERG team was the reluctance for domain scientists to work within the Agile-based framework we had intended at the outset of the project. It was particularly difficult to get domain scientists to release code at its intermediate stages, even to share the code with other members of the software development team. These challenges speak to the culture within domain science research groups, whereby typically a single researcher is developing code and code is only shared or published once it is "perfect". These challenges persisted even once we had an established GitHub repository and were not a function of not knowing how, at a technical level, to share code with other members of the team. As a result, domain science teams labored too long on problems that could have been more easily addressed by the research programmers involved with the project. It also made it difficult to ensure consistency across different use cases in how larger workflows were divided into smaller modules.

An additional challenge is that domain scientists are often uncomfortable working in a public github environment where other researchers could "scoop" their code before it has been fully developed and published. We have overcome this dichotomy by assuring the teams that the HPC pipeline can move forward without the "perfect" code needed for publishable results. The CI team can work with placeholder functions, but does need all the steps to create, verify, and profile the pipeline. An example of this is within the LandCover use case where atmospheric correction and shadow detection are crucial. There is no concensus among researchers on the correct way to handle these issues, but we added "dummy" stages to the pipeline to permit progress on the rest of the workflow and its optimization that can be easily replaced when an acceptable solution is found.

5 SCIENCE RESULTS

The first science-domain results from ICEBERG have been published in *Remote Sensing of Environment* on the Seals use case [6] This paper represents the first step towards a regular, cost-effective, pan-Antarctic seal monitoring program. It outlines the first automated system for surveying seals using satellite imagery, finding 30% of seals while only generating less than 2 false-positives per correct detection, and running over 10× faster than an experienced human observer using a single GPU.

Other preliminary results have been presented for the Rivers, LandCover, and Penguins use cases at EarthCube and American Geophysical Union (AGU) conferences over the past two years.

6 FURTHER WORK

6.1 Public distribution and support

The ICEBERG packages will be released via PyPI [12] and will also be installable from the GitHub repository. The Seals prototype was released in August 2019 and Penguins in February 2020. The LandCover and Rivers use cases will follow in summer and fall 2020.

Each use case is released as a separate, stand-alone package and the middleware is contained in its own package. The stand-alone product can be used for small batches of imagery or coupled with the middleware for large-scale production.

We encourage any researchers working on high resolution imagery intensive projects to contact any ICEBERG team member via the website for support in implementing ICEBERG.

6.2 Non-polar applications

Throughout the ICEBERG project we have been struck by the commonalities that appeared among the wide-ranging use cases. The same principles, techniques, and even code can be reused by multiple projects. We have recently partnered with the Institute for Marine Remote Sensing (IMaRS) [8] at the University of South Florida to help with processing imagery of coastal regions around the Gulf of Mexico. We look forward to assisting other groups in similar ways.

ACKNOWLEDGMENTS

ICEBERG is supported by NSF collaborative grant #1740595.

REFERENCES

[1] Vivek Balasubramanian, Matteo Turilli, Weiming Hu, Matthieu Lefebvre, Wenjie Lei, Ryan Modrak, Guido Cervone, Jeroen Tromp, and Shantenu Jha. 2018. Harnessing the power of many: Extensible toolkit for scalable ensemble applications. In *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 536–545.

[2] Software Carpentry. 2020. *Software Carpentry:Teaching basic lab skills for research computing.* Retrieved February 16, 2020 from https://software-carpentry.org/

[3] Travis CI. 2020. *Test and Deploy with Confidence.* Retrieved February 16, 2020 from https://travis-ci.org/

[4] Radical CyberTools. 2020. *RADICAL Cybertools (RCT) - Scalable, Interoperable and Sustainable Tools for Science.* Retrieved February 16, 2020 from http://radical.rutgers.edu/projects/rct

[5] Vincent Driessen. 2010. *A successful Git branching model.* Retrieved February 16, 2020 from https://nvie.com/posts/a-successful-git-branching-model/

[6] Bento Collares Gonçalves, Bradley Spitzbart, and Heather J. Lynch. 2020. SealNet: A fully-automated pack-ice seal detection pipeline for sub-meter satellite imagery. *Remote Sensing of Environment* 239 (2020). https://doi.org/10.1016/j.rse.2019.111617

[7] ICEBERG. 2020. *ICEBERG: Imagery Cyber-infrastructure and Extensible Building blocks to Enhance Research in the Geosciences.* Retrieved February 16, 2020 from https://iceberg-project.github.io/about.html

[8] IMaRSi. 2020. *Institute for Marine Remote Sensing.* Retrieved February 16, 2020 from http://imars.usf.edu/

[9] André Merzky, Mark Santcroos, Matteo Turilli, and Shantenu Jha. 2015. RADICAL-Pilot: Scalable Execution of Heterogeneous and Dynamic Workloads on Supercomputers. *CoRR* abs/1512.08194 (2015). arXiv:1512.08194 http://arxiv.org/abs/1512.08194

[10] Andre Merzky, Ole Weidner, and Shantenu Jha. 2015. SAGA: A standardized access layer to heterogeneous distributed computing infrastructure. *SoftwareX* 1 (2015), 3–8.

[11] Ioannis Paraskevakos, Matteo Turrili, Bento Collares Gonçalves, Heather J. Lynch, and Shantenu Jha. 2019. Workflow Design Analysis for High Resolution Satellite Image Analysis. *CoRR* abs/1905.09766 (2019). arXiv:1905.09766 http://arxiv.org/abs/1905.09766

[12] Pypi. 2020. *PyPI · The Python Package Index.* Retrieved February 16, 2020 from https://pypi.org/

[13] Matteo Turilli, Vivek Balasubramanian, Andre Merzky, Ioannis Paraskevakos, and Shantenu Jha. 2019. Middleware building blocks for workflow systems. *Computing in Science & Engineering* 21, 4 (2019), 62–75.