Snowmass2021 - Letter of Interest

IceCube and IceCube-Gen2 User Analysis Computing

Thematic Areas: (check all that apply □/■)
□ (CompF1) Experimental Algorithm Parallelization
□ (CompF2) Theoretical Calculations and Simulation
□ (CompF3) Machine Learning
■ (CompF4) Storage and processing resource access (Facility and Infrastructure R&D)
■ (CompF5) End user analysis
□ (CompF6) Quantum computing
□ (CompF7) Reinterpretation and long-term preservation of data and code

Contact Information:
Kevin Meagher (University of Wisconsin–Madison) [kmeagher@icecube.wisc.edu],

Authors (alphabetical):
Kevin Meagher (University of Wisconsin–Madison) [kmeagher@icecube.wisc.edu],
Benedikt Riedel (University of Wisconsin–Madison) [briedel@icecube.wisc.edu],
David Schultz (University of Wisconsin–Madison) [dschultz@icecube.wisc.edu],
on behalf of the IceCube[1] and IceCube-Gen2[2] Collaboration [analysis@icecube.wisc.edu]

Abstract: (must fit on this page)
The IceCube Neutrino Observatory is a km^3 neutrino detector deployed at the South Pole. IceCube measures neutrinos by detecting the optical Cherenkov photons produced in neutrino-nucleon interactions. IceCube analyses span a wide variety of scientific topics, each of which places different needs on computing. This letter of interest covers IceCube's experiences with computing for end user analysis and the issues we have faced. Topics discussed are programming languages, file formats, computing hardware, and coding practices.

[1]Full author list available at https://icecube.wisc.edu/collaboration/authors/snowmass21_icecube
[2]Full author list available at https://icecube.wisc.edu/collaboration/authors/snowmass21_icecube-gen2

The IceCube Neutrino Observatory [1], located at the South Pole, instruments a cubic kilometer of Antarctic ice. IceCube uses 5160 digital optical modules (DOMs) arranged on 86 strings in a hexagonal array to detect Cherenkov radiation from relativistic charged particles emitted during neutrino interactions. This configuration can detect neutrinos as high as EeV energies while the center, more concentrated region of DOMs, called DeepCore, is optimized to extend the detection energy down to a GeV. The IceCube Upgrade plans to improve on the resolution of GeV neutrinos by adding an additional seven strings concentrated around DeepCore with varying DOM designs and additional calibration devices [2]. To improve resolution of TeV to EeV neutrino detection and increase the detection rate, Gen2 will add 120 strings to instrument a total volume of 7.9 km^3 [3]. These extensions will increase the data rate and complexity, increasing the scale of user analysis computing and challenges for individual end users.

Programming Languages

In IceCube, most analysis software is written in `Python`. Analysers find `Python` to be a very expressive language which makes it easy to develop detailed analyses. Our experience is that it is easy to read and understand python code written by others, which is important for avoiding the pitfall of relying on black box algorithms. Although `Python` is often criticized as slow for computational work, this only applies to algorithms written in pure `Python`. When numpy is used for computational work, speeds comparable to compiled languages can be achieved. In addition, analysers generally find numpy's vector arithmetic and splicing syntax to be a very intuitive and easy way to work with data.

`Python` comes with a number of scientific packages which are extremely useful for analysis software: scipy for mathematical and scientific routines, matplotlib for plotting, and pandas for data processing. `Python` is also the preferred language for many machine learning frameworks, including scikit-learn and tensorflow. Jupyter notebooks provide an interactive environment which many find appealing. For algorithms which cannot easily be vectorized using numpy, an extension module can be written in C/C++. One of IceCube's analysis libraries, csky, makes extensive use of extension modules for a significant speed up while allowing the analyser to call the algorithms from `Python`.

Even thought most analysers seem to prefer `Python`, `ROOT` is also used as an analysis environment by some users. Use of `ROOT` is generally unpopular because of the relatively cumbersome installation process, and because of difficulty accessing data in a columnar format. `ROOT` data structures such TTree have a niche syntax that students and other novice users find difficult to navigate.

File Formats

IceCube analysers use a variety of file formats for storing the final level sample. No one format has been found to satisfy all analysers. Icecube uses a custom file format for processing and storage of low level data called "i3". Because this format serializes entire events, random access of events is difficult and the i3 format is not a good choice for final level data samples. To convert data into more appropriate formats, IceCube's software has a converter to convert i3 data into various table-based data formats.

The most common formats used are HDF5, numpy, and `ROOT`. Data are often accessed in a columnar fashion, and HDF5 has a `Python` interface in which columnar access follows numpy conventions. This makes accessing data easy, but has some drawbacks. The format is not stored on disk in columnar format so the whole table must be read into memory, which can be a limitation for large datasets. The numpy format, in contrast, is a very simple format and is very fast to read, but it can only store a single table and has no metadata, which makes long term storage of analysis

events difficult. ROOT files are typically only used in analyses written in ROOT. ROOT does not allow simple columnar access to data and many find installing the entire ROOT library burdensome to access a file format.

Analysis Computing

The amount of computational resources for IceCube analyses varies widely from analysis to analysis. Some have relatively trivial hardware requirements while others take a significant amount of CPU and memory usage. Generally, analysers have found IceCube's main computing cluster at University of Wisconsin–Madison (roughly 6000 CPUs, 200 GPUs, and 3 PB of user storage) adequate to meet the demands of analyses which require high CPU usage. However, many of our analyses depend on algorithms which require memory allocations which exceed 64GB of memory. Although workarounds exist, high memory computing resources would be beneficial.

Current analysis computing is done in a fairly traditional manner by using ssh to access interactive and batch systems. In the future, we are looking towards JupyterLab web-based access. Many analysers already use private Jupyter Notebooks [4] for smaller work, and this would be the next step in providing a central, supported installation. There has been some community work in extending these notebooks to cluster computing via Dask, which we are considering to adopt.

Code Best Practices

IceCube has found that it is very important to understand the structure of analysis code. Many IceCube analyses are based on the method of maximum likelihood[5, 6] and use the same or similar algorithms to maximize the likelihood, perform pseudo-experiments to determine the background likelihood distribution, and calculate the sensitivity and discovery potential. However, many parts of these analysis differ such as the event sample, the definition and parameters of the likelihoods and the method of scrambling background data. We found that taking the time to understand the structure of the code and refactor it into a library where new analysers could quickly assemble provided components into a new analysis was highly beneficial. IceCube plans to continue to refine our analysis software and develop it into an modular and easy to use framework[7].

Reproduciblity of analysis code is also a major concern for IceCube. Significant effort has been invested to ensure that published results can be reproduced with newer versions of software. When new algorithms are developed, the older version is available as an option, so that new versions of software can reproduce published analyses exactly, and all changes between analyses are well understood.

Most graduate students, even ones with coding experience, are not well versed in the best practices of software development. It is important that every analysis performed by our collaboration be incorporated into analysis libraries. In the absence of clear guidance, most analysers are not interested in (or are too inexperienced or time-constrained) to do the work of incorporating any changes developed back into the main branch of the analysis library. We found that it is highly beneficial to create a development community to teach analysers how to commit their changes back into the main development branch.

References

[1] M. G. Aartsen et al. The IceCube Neutrino Observatory: Instrumentation and Online Systems. *JINST*, 12(03):P03012, 2017.

[2] Aya Ishihara. The IceCube Upgrade – Design and Science Goals. *PoS*, ICRC2019:1031, 2020.

[3] M.G. Aartsen et al. IceCube-Gen2: The Window to the Extreme Universe. 8 2020.

[4] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.

[5] Jim Braun, Jon Dumm, Francesco De Palma, Chad Finley, Albrecht Karle, and Teresa Montaruli. Methods for point source analysis in high energy neutrino telescopes. *Astroparticle Physics*, 29(4):299–305, May 2008.

[6] Jim Braun, Mike Baker, Jon Dumm, Chad Finley, Albrecht Karle, and Teresa Montaruli. Time-dependent point source search methods in high energy neutrino astronomy. *Astroparticle Physics*, 33(3):175–181, April 2010.

[7] M. Wolf. SkyLLH - A generalized Python-based tool for log-likelihood analyses in multi-messenger astronomy. In *36th International Cosmic Ray Conference (ICRC2019)*, volume 36 of *International Cosmic Ray Conference*, page 1035, July 2019.