IceProd 2: A Next Generation Data Analysis Framework for the IceCube Neutrino Observatory

D Schultz
Wisconsin IceCube Particle Astrophysics Center, University of Wisconsin-Madison, 222 W Washington Ave, Suite 500, Madison, WI 53703, USA

E-mail: david.schultz@icecube.wisc.edu

Abstract. We describe the overall structure and new features of the second generation of IceProd, a data processing and management framework. IceProd was developed by the IceCube Neutrino Observatory for processing of Monte Carlo simulations, detector data, and analysis levels. It runs as a separate layer on top of grid and batch systems. This is accomplished by a set of daemons which process job workflow, maintaining configuration and status information on the job before, during, and after processing. IceProd can also manage complex workflow DAGs across distributed computing grids in order to optimize usage of resources. IceProd is designed to be very light-weight; it runs as a python application fully in user space and can be set up easily. For the initial completion of this second version of IceProd, improvements have been made to increase security, reliability, scalability, and ease of use.

1. Introduction
IceProd has served the IceCube collaboration well over the last 8 years, processing Monte Carlo simulations, detector data, and analysis levels [1]. It has run thousands of CPU-core years and stored over 2PB of data. But it has experienced its share of growing pains and the initial software is running into fundamental design limitations. This article describes that history and the development of a second version of IceProd. The new version is a complete rewrite of the code base, fixing long-standing issues and preparing for the increased load of the next few years.

2. IceCube and Computing
As IceProd was made for and by IceCube, development of IceProd has been primarily driven by IceCube detector requirements. Experimental and analysis data undergoes additional filtering and reconstruction and the majority of simulated data is produced through IceProd. Computing resources impose several additional constraints that IceProd must fit into.

2.1. The IceCube Detector
The IceCube detector shown in figure 1 is located at the geographic South Pole and was completed at the end of 2010 [2]. It consists of 5160 optical sensors buried between 1450 and 2450 meters below the surface of the South Pole ice sheet and is designed to detect interactions of neutrinos of astrophysical origin. However, it is also sensitive to downward-going highly energetic muons and neutrinos produced in cosmic-ray-induced air showers. IceCube records $\sim10^{10}$ cosmic-ray events per
Figure 1 The IceCube detector: the dotted lines at the bottom represent the instrumented portion of the ice. The circles on the top surface represent IceTop, a surface air-shower sub-detector.

year. The cosmic-ray-induced muons outnumber neutrino-induced events by about 500,000:1. They represent a background for most IceCube analyses and are filtered prior to transfer to the data processing center in the northern hemisphere. Filtering at the data collection source is required because of bandwidth limitations on the satellite connection between the detector and the processing location [3]. About 100 GB of data from the IceCube detector is transferred to the main data storage facility at UW-Madison daily. Once the data has been transferred, additional, more computing-intensive event reconstructions are performed and the data is filtered to select events for various analyses. IceCube analyses also require production of Monte Carlo simulations with statistics that are comparable to the data collected by the detector. This requires thousands of years of CPU time.

2.2. IceCube Computing Resources

The IceCube collaboration is comprised of 44 institutions in 12 countries. There are two main data centers, one in North America (UW-Madison) and one in Europe (DESY Zeuthen). The other 42 institutions generally only have limited access to shared campus resources or smaller compute clusters. Additionally, there is significant availability of opportunistic resources via providers like the Open Science Grid [4]. Currently, over 50% of all CPU resources are opportunistic.

One of the highlights of the IceCube simulation chain is the direct propagation of photons through the ice using GPU accelerators [5]. The decision was made to switch from parameterization tables to direct propagation because of improved physics and the feasibility of the problem on GPUs. Over the last 3 years, IceCube has deployed more than 500 GPUs to help run this part of the simulation chain, most of which are located in the North American data center.

The IceCube simulation and data processing jobs have a large variability in resource requirements. Most simulation jobs are between 2-4 GB of memory, but several types of jobs require 6+ GB of memory. Certain simulation generation jobs require significant scratch disk, on the order of 100 GB, to manage temporary files. Finally, one of the problems with Monte Carlo simulations is that rare
events, such as extremely high energy events, can take extra resources. This is most noticeable with walltime, where a high energy event can cause order of magnitude increases in processing time. Handling these randomly occurring events is an ongoing issue.

2.3. IceProd Requirements
Based on the information in sections 2.1 and 2.2, a list of requirements appears:

- Anything that requires root permission at a site is unlikely to work. IceCube isn’t big enough to enforce policy, especially on opportunistic resources.
- There is a need for more than glideinWMS [6]. Many of the smaller sites aren’t set up to run glideins via a Grid Computing Element interface, and integrating them would be difficult. Glidein infrastructure is used where possible though.
- Multiple OS types and schedulers need to be supported. While Scientific Linux and other RedHat derivatives are common because of LHC experiment requirements, many clusters run other OS types like Ubuntu and SUSE. The range of schedulers is even broader, with clusters running HTCondor, PBS, LSF, SGE, SLURM, and more.
- Site-local disk mostly isn’t available. Most clusters only provide compute, not storage.
- A detailed record of job configuration and runtime statistics must be maintained. Analyzers will refer back to the configuration for details that may affect their analysis. Good reporting of resource usage, efficiency, or other quantities is valuable to various groups for decision making.

2.4. IceCube Job Workflow
The core software framework of IceCube, called IceTray, is stream-based with modules that act on events in the stream [7]. Simulation jobs run similarly, with a series of modular jobs forming a stream (or multiple streams) of data that is incrementally processed to further levels, as in figure 2. A series of these streams can run in parallel with different random seeds, and forms a dataset. It is typical to have up to a few hundred datasets running concurrently.

The technical term for the individual job stream is a directed acyclic graph (DAG). In figure 3 a more complex DAG can be seen, where two simulation generators run on CPUs, feeding four GPU jobs, which in turn feed their output to five parallel cpu jobs. These complex DAGs are used in real production, but they currently create significant scheduling overhead with extra database lookups to check dependency status; thus, they are only used when really needed. Simpler DAGs are preferred.

Figure 2 Each box is a job, with data movement between jobs represented by arrows. Many hundreds or thousands of parallel job streams forms a dataset.

Figure 3 Typical simulation job streams have a GPU job which often runs at a different site than the CPU jobs. This is a more complex example, showing many interdependent jobs making up one “stream.”
3. IceProd v1

The initial version of IceProd is a set of user-space site-local python daemons backed by a central
database [8]. All knowledge of state is stored in the database, with the python daemons interacting
with the queueing system or the jobs directly. One daemon submits new jobs and keeps track of the
queue. Another daemon listens for XML-RPC from running jobs so their status can be updated as the
job runs through its lifecycle. A third daemon is responsible for any cleanup tasks that are required,
and a final daemon responds to user commands such as dataset submission, suspension, or deletion. A
web interface was developed in PHP that interfaced with the database and displayed its contents in an
easy to read format, along with forwarding user commands to the IceProd daemon to control datasets.
Plugins have been designed for each batch system or other job submission interface. This turns a
multitude of heterogeneous systems into a homogeneous API that IceProd can interact with. It also
allows for new systems to be incorporated over time with minimal effort.

3.1. Additions

One of the major additions to IceProd came with the use of GPUs. It was necessary to break up jobs
into CPU and GPU parts instead of one large job. Initially, with GPUs in a testing phase, this was
accomplished locally at a few sites. DAGMan for HTCondor was the testbed platform [9]. This
worked fairly reliably, but did not extend to other job submission systems. It also did not allow jobs
from one site to utilize GPUs at another site. The DAG feature was integrated directly into IceProd
1.5 years ago and solved cross-site linking. It is now a prominent theme of IceCube computing.

Along with splitting jobs up to run on GPUs, there was a necessity to mark those jobs as special. A
special syntax was developed for GPU use, to describe both that a job needed a GPU and to include
site-local GPU configuration in the job. This has worked well for GPU resources, but is less
extensible to other resource types.

Reporting has always been a goal of IceProd, but has received a large increase in attention in the
last year. Effort was made to normalize the CPU and GPU speeds for each node, to present a more
consistent picture of site resources. Real-time and historical reporting has also improved quite a bit,
making monitoring of sites and spotting trends much easier. This has helped guide decision making
both locally and at higher levels.

3.2. Problems

When IceProd was first designed 8 years ago, the jobs it was expected to run were simple: a single
monolithic job running every part of the simulation chain, copying the final result to a data storage
repository. Clusters were generally small, only a few hundred nodes at most, and with jobs taking
hours to complete the queue rate was fairly slow. The first version of IceProd reflects these facts and
was designed more for job control and record-keeping than performance. With today’s realities of
DAGs of shorter jobs and clusters of many thousands of nodes, bottlenecks are appearing in multiple
places.

One of the most obvious bottlenecks is queueing jobs into the batch scheduler on a cluster. Jobs
are individually queued and typically take a second or more to queue. With short job walltime and
large clusters, we are starting to reach the situation where constantly queueing can’t saturate the
cluster. Carefully watching job walltime is helping, but won’t mitigate this problem for long. Eventually clusters will grow large enough that we just need to queue faster.

The central database is also seeing its share of issues. The move to DAGs of jobs has led to a
substantial rise in the number of queries on the database. At times the sheer number of queries per
second is enough to slow things down, and we have to be very careful of the new queries we add. We
are also starting to hit connection limits. While not a serious problem, it indicates a design issue when
over 500 clients are connected to the database and can potentially run a query at the same time. This
may be contributing to the database tables having locking issues, which has gotten worse in the last
year when scales of 10 thousand completed jobs per hour has been reached.
Essentially, several design decisions are hampering new ideas and current activity. There are better ways to do some things, but they are almost impossible to integrate into the current framework. On top of all of this is the code cruft accumulated over 8 years of patches without much cleanup.

4. IceProd v2

The second version of IceProd is a complete rewrite of the codebase, focusing on current and future usage and eliminating old usage patterns. Instead of being add-ons, features like global DAG are now standard, with a more efficient implementation that was not possible before. There is also a strong focus on unit tests, code coverage and documentation, which were lacking in IceProd v1. Python 3 is fully supported in order to make ready for the next few years, when it will become standard.

4.1. New Goals

IceProd v2 has a few additional goals that were not present for IceProd v1. The primary new goal is to serve as a global platform for analysis users as well as production. This is motivated by the scale that a number of recent analyses have reached, with individual users submitting millions of jobs by themselves and exceeding the limits of a single cluster. While users with good technical skills can make this work right now, IceProd v2 hopes to make this process significantly easier. In the future, analyzers can either submit jobs in the main production network under non-production accounts or start their own network of linked sites. A secondary goal for IceProd v2 is to increase the ability to debug production jobs by giving the same environment as production, but in an offline manual mode.

4.2. Major Improvements

There are several major improvements, but one of the overarching ones is to be able to run on stock python with no extra packages installed. While performance is degraded, it is sufficient for individual users or testing purposes. It also allows a single-site IceProd instance to be downloaded, set up, and started very quickly.

4.2.1. Database

The database is the single largest change to IceProd v2. It has moved from one central database to a local database at each site. This allows standalone single-site instances as well as continued running when the master/global queue is down. It also has a side effect of spreading out the load, since most things only need to talk to the local database instead of all hitting the master.

The primary database type is SQLite, which comes integrated with Python. Support is also available for MySQL for higher performance. MongoDB has been chosen to handle the large amount of statistics accumulated, and though it is optional it is recommended for production deployment at least at the master node and large sites.

4.2.2. Scalability

All external communications with IceProd v2 flow through the web server, so it must be fast and scalable. Not only does it handle user-facing websites, but also job-to-site and site-to-site communications. It is built on non-blocking principles, offloading longer tasks to worker threads or other processes. As seen in figures 4 and 5, it can easily handle 10 thousand connections per minute, even with over a thousand sustained connections.

4.2.3. Job Optimization

One long-desired feature for IceProd has been a pilot infrastructure. Especially in the last few years, shorter jobs under an hour have become normal. Clusters and queueing systems generally don’t like short jobs, so the obvious solution is to run multiple jobs one after the other inside the “single” job the queueing system is running. The pilot infrastructure takes this a step further by only asking for jobs that match the node’s resources where the pilot is started. This allows jobs to run where they are best matched and reduces failures.
4.2.4. Software Distribution. While not technically part of IceProd, software distribution is interrelated to a high degree. Jobs need software to run, and for IceProd v1 this meant tarballs shipped with each job and extracted on the worker node. While this will still be supported for legacy operations in IceProd v2, the primary mode of software distribution is moving to CernVM-FS [10]. For sites that have CernVM-FS installed, it is used natively. Otherwise, an I/O interposition agent called Parrot is used to make the filesystem appear to exist, dynamically fetching necessary files from the network and presenting them to the application in the correct location [11].

4.2.5. Users and Priority. While IceProd v1 only had a single account for production, IceProd v2 needs multi-user support to keep analysis users separate from production. A future version might also include support for “groups” of users to help organize things. Tied into the concept of multiple users is their job priority within the system. A fair-share accounting will be used, such that each submit cycle jobs will be weighted based on past usage and administrator-defined priority for that user. This should prevent one dataset or user from completely starving the rest.

4.2.6. Web Focus. IceProd v2 will feature a full JSON API as the primary interface to controlling IceProd. This allows great flexibility because of JSON’s ease of use and the ubiquity of the internet. This gives a single, shared interface that both the website and command line clients can use as a backend, while presenting users with their preferred frontend. If a more technical user wants to use it directly, they can also easily accomplish that to create a new, personalized frontend. But the primary frontend will be the website, which will work either on a traditional desktop browser, or on smaller devices such as tablets or phones.

5. Status and Future Work
IceProd v2 is currently in beta testing for production deployment, which is scheduled for late summer 2015. Multi-user features have been held back to complete this part and will be released later in v2.1, likely early 2016. After that comes a long list of “nice to have” features, such as better monitoring and job error handling. Fortunately, the improved code base will help to accelerate new development.

Outreach will also be needed in the future, first to get analyzers in IceCube to use IceProd for their batch computing needs, then to spread IceProd to other collaborations. Groups or individuals of any type, even outside high energy physics, are welcome to adopt this software.

6. Conclusions
IceProd is an easy to install user-level middleware designed to easily manage and monitor large sets of distributed jobs. The IceCube collaboration uses it for simulation, detector data, and analysis level processing. IceCube has heterogeneous, significantly opportunistic resources and diverse job.
requirements. GPUs have forced cross-site job dependencies, which has strained the existing IceProd v1 system. IceProd v2 is a complete rewrite designed to natively handle current production and more easily cope with future needs. IceProd can scale from a single user at one site up to sites around the world and many millions of jobs. Though it was originally developed for managing IceCube simulation production, IceProd is general enough for many types of grid applications and is generally available to the scientific community.

Acknowledgements
We acknowledge the support from the following agencies: U.S. National Science Foundation-Office of Polar Programs, U.S. National Science Foundation-Physics Division, University of Wisconsin Alumni Research Foundation, the Grid Laboratory Of Wisconsin (GLOW) grid infrastructure at the University of Wisconsin–Madison, the Open Science Grid (OSG) grid infrastructure; U.S. Department of Energy, and National Energy Research Scientific Computing Center, the Louisiana Optical Network Initiative (LONI) grid computing resources; Natural Sciences and Engineering Research Council of Canada, WestGrid and Compute/Calcul Canada; Swedish Research Council, Swedish Polar Research Secretariat, Swedish National Infrastructure for Computing(SNIC), and Knut and Alice Wallenberg Foundation, Sweden; German Ministry for Education and Research (BMBF), Deutsche Forschungsgemeinschaft (DFG), Helmholtz Alliance for Astroparticle Physics (HAP), Research Department of Plasmas with Complex Interactions (Bochum), Germany; Fund for Scientific Research (FNRS-FWO), FWO Odysseus programme, Flanders Institute to encourage scientific and technological research in industry (IWT), Belgian Federal Science Policy Office (Belspo); University of Oxford, United Kingdom; Marsden Fund, New Zealand; Australian Research Council; Japan Society for Promotion of Science (JSPS); the Swiss National Science Foundation (SNSF), Switzerland; National Research Foundation of Korea (NRF); Danish National Research Foundation, Denmark (DNRF).

References
[1] Aartsen M G and et al 2015 The IceProd framework: Distributed data processing for the IceCube neutrino observatory J. Parallel Distrib. Comput. 75 198–211
[2] Halzen F 2003 IceCube A Kilometer-Scale Neutrino Observatory at the South Pole IAU XXV General Assembly, Sydney, Australia, 13-26 July 2003, ASP Conf. Series vol 13 volume 13, pages 13-16, July 2003
[3] Halzen F and Klein S R 2010 Invited Review Article: Icecube: An Instrument for Neutrino Astronomy Rev. Sci. Instrum., AIP 81
[4] Pordes R and et al 2007 J. Phys.: Conf. Ser. 78 012057
[5] Chirkin D 2013 Photon tracking with GPUs in IceCube Nucl. Instrum. and Methods Phys. Res. Section A: Accelerators, Spectrometers, Detectors Associated Equipment 725 (0)141–143 VLVvT 11, Erlangen, Germany, 12 – 14 October, 2011. 5th International Workshop on Very Large Volume Neutrino Telescopes, The future of high-energy neutrino astronomy
[6] Sfiligoi I 2008 J. Phys.: Conference Ser. 119 062044
[7] DeYoung T 2005 IceTray: a software framework for IceCube Int. Conf. on Comp. in High-Energy Phys. and Nucl. Phys. (CHEP 2004) 463–466
[8] Díaz-Vélez J C 2014 J. Phys.: Conf. Ser. 513 032026
[9] Couvares P, Kosar T, Roy A, Weber J and Wenger K 2007 Workflow in Condor Workflows for e-Science
[10] Blomer J, Buncic P and Fuhrmann T 2011 Proc. of the 1st int. workshop on Network-aware data management (NDM’11) 49–56
[11] Thain D and Livny M 2005 Scalable Computing: Practice Experience 6 9