The IceProd (IceCube Production) Framework

J. C. Díaz-Vélez
Wisconsin Particle and Astrophysics Center, University of Wisconsin-Madison
E-mail: juancarlos.diazvelez@icecube.wisc.edu

Abstract.
IceProd is a data processing and management framework developed by the IceCube Neutrino Observatory for processing of Monte Carlo simulations and data. IceProd runs as a separate layer on top of middleware or cluster job schedulers and can take advantage of a variety of computing resources including grids such as EGI, OSG, and NorduGrid as well as local clusters running batch systems like HT Condor, PBS, and SGE. This is accomplished by a set of dedicated daemons which process job submission in a coordinated fashion through the use of middleware plug-ins that serve to abstract the details of job submission and job management. IceProd can also manage complex workflow DAGs across distributed computing grids in order to optimize usage of resources. We describe several aspects of IceProd’s design and its applications in collaborative computing environments. We also briefly discuss design aspects of a second generation IceProd, currently being tested in IceCube.

1. Introduction
Large experimental collaborations often need to produce large volumes of computationally intensive Monte Carlo simulations as well as data processing. For such large datasets, it is important to be able to document software versions and parameters including pseudo-random generator seeds used for each dataset produced. Individual members of such collaborations might have access to modest computational resources that need to be coordinated for production and could be pooled in order to provide a single, more powerful, and more productive system that can be used by the entire collaboration. For this purpose, we have designed a software package consisting of queuing daemons that communicate via a central database in order to coordinate production of large datasets by integrating small clusters and grids.

2. IceCube
The IceCube detector shown in Figure 1 consists of 5160 optical sensors buried between 1450 and 2450 meters below the surface of the South Polar ice sheet and is designed to detect interactions of neutrinos of astrophysical origin [1]. However, it is also sensitive to downward-going highly energetic muons and neutrinos produced in cosmic-ray-induced air showers. IceCube records $\sim 10^{10}$ cosmic-ray events per 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 [2]. Once the data has been transferred, additional, more computing-intensive event reconstructions are performed and the data is filtered...
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.

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 hundreds of years of CPU time.

3. IceProd
The IceProd framework is a software package developed for IceCube with the goal of managing productions across distributed systems and pooling together isolated computing resources that are scattered throughout the Collaboration [3]. It consists of a central database and a set of daemons that are responsible for management of grid jobs and data handling through the use of existing grid technology and network protocols.

The software package includes a set of libraries, executables and daemons that communicate with the central database and coordinate to share responsibility for the completion of tasks. The details of job submission and management in different grid environments are abstracted through the use of plug-in modules that launch the corresponding commands or library calls as the case may be. IceProd contains a growing library of plug-ins, including classes for interfacing with systems such as HT Condor, PBS, SGE, Globus, gLite, Edg, CREAM, SweGrid and other batch systems.

IceProd is not meant to be a replacement for other cluster and grid management tools such as Globus, gLite or any other middleware. Instead, it runs on top of these as a separate layer providing additional functionality. IceProd fills a gap between the user or production manager and the powerful middleware and batch system tools currently available on computing clusters and grids.

Many of the existing middleware tools including Condor-C, Globus and CREAM already make it possible to interface any number of computing clusters into a larger pool. In contrast to most of these applications, IceProd runs at the user level and does not require administrator privileges. This makes it possible for individual users to build large production systems by pooling small computational resources together.
Figure 2. Network diagram of IceProd system: The IceProd clients and JEPs communicate with `iceprod-server` modules via XML-RPC. Database calls are restricted to `iceprod-server` modules. Queueing daemons called `soapqueue` are installed at each site and periodically query the database for pending job requests. The `soapmon` server receives monitoring update from the jobs. An instance of `soapdh` handles garbage collection and any post processing tasks after job completion.

3.1. IceProd Server
The IceProd server is comprised of four daemons depicted in Figure 2 and their respective libraries. Upon submission of a dataset, all parameters are stored in the central database which also tracks the progress of each job. The `soapqueue` daemon running at each of the sites periodically queries the database to check if any tasks have been assigned to it and submits jobs to its local cluster or grid. The number of jobs that IceProd maintains in the queue at each site can be configured individually.

3.2. The JEP
IceProd submits a Job Execution Pilot (JEP) as part of the management of each job to the cluster/grid queue. This script determines what platform a job is running on and what software it needs to stage. During runtime the JEP will perform status updates through the monitoring server via remote procedure calls using XML-RPC [4]. This information is updated on the database and is displayed on the monitoring web page. Upon completion, the JEP will remove temporary files and directories created for the job.

Jobs can fail under many circumstances. These can include submission failures due to transient system problems and execution failures due to problems with the execution host. In order to account for possible transient errors, the design of IceProd includes a set of states through which a job will transition in order to guarantee successful completion of a well-configured job. As illustrated on the state diagram in figure 3, each non-error state includes a time limit before the job is reset and rescheduled. If a job enters a n error state, it will automatically get rescheduled up to some maximum number of failures. If this limit is reached, manual intervention is required since it can be assumed that the error is most likely due to a configuration or software problem. User identity is handled by standard GSI authentication and communication between client and server is encrypted through HTTPS. In addition each...
job is assigned a temporary random-generated string at the time of submission and it is used for authenticating communication between the job and the monitoring server and is only valid during the duration of the job. Data integrity is insured by generating and storing MD5sum for each file prior to transfer. All data transfers are handled through GridFTP. More details about IceProd’s security and data integrity features are given in [6].

3.3. Web Interface
The current web interface for IceProd was designed to work independently from the IceProd framework but utilizes the same database. Each of the simulation and data processing web monitoring tools provide different views with various levels of detail. This web interface also provides the functionality to control jobs and datasets by authenticated users.

4. The IceProd DAG
The IceProdDAG is native management system similar to HT Condor’s DAGMan [7]. The IceProdDAG relies on a consumer-based approach to scheduling where local resources consume tasks on demand. This approach naturally lends itself to efficient utilization of computing resources. This workflow management system consists of a pair of plug-ins that take the roles of a master queue and a slave queue, respectively.

The master queue IceProdDAG class interacts solely with the database rather than acting as an interface to a batch system while the slave task queue manages task submission via existing plug-in interfaces. Both of these classes are implemented as plug-in modules.

The TaskQ or slave class handles task errors, resets, and time outs in a similar way that other plug-ins do for jobs but also takes additional task states into account. The state diagram for a task is shown in Figure 3. Each instance of a TaskQ needs to handle its own garbage collection and set the appropriate states independent of the job being handled by the IceProdDAG instance. Details of the implementation of IceProdDAG are given in Ref. [6].

A major advantage of this approach is that it allows a single DAG to span multiple sites and thereby make optimal use of resources. An example application would allow to combine one site with vast amounts of CPU power but no GPUs available and another site better equipped with GPUs.

![Figure 3. JEP state diagram for task. Each of the non-error states through which a task passes includes a configurable timeout.](image)

4.1. DAGs Based on System Requirements
Individual parts of a job may have different system hardware and software requirements. Breaking these up into tasks that run on separate nodes allows for better utilization of resources.
The IceCube detector simulation chain is a good example of this scenario in which tasks are distributed across computing nodes with different hardware resources.

4.2. GPU Tasks
Light propagation in the instrumented volume of ice at the South Pole is difficult to model, but recent developments in IceCube’s simulation include a much faster approach for direct propagation of photons in the optically complex Antarctic ice [8, 9] by using general-purpose GPUs. This new simulation module is much faster than a CPU-based implementation and more accurate than using parametrization tables [10], but the rest of the simulation requires standard CPUs. When executing an IceProd job monolithically, only one set of cluster requirements can be applied when it is submitted to the cluster. Accordingly, if any part of the job requires use of a GPU, the entire monolithic job must be scheduled on a cluster machine with the appropriate hardware.

The modular design of the IceCube simulation design is used to divide the CPU and GPU dependent portions of jobs into separate tasks in a DAG. Since each task in a DAG is submitted separately to the cluster, their requirements can be specified independently and CPU tasks can be executed on general-purpose grid nodes while photon propagation tasks can be executed on GPU-enabled machines, as depicted in Figure 4.

Figure 4. A simple DAG in IceProd. This DAG corresponds to a typical IceCube simulation. The two root nodes require standard computing hardware and produce different types of signal. Their output is then combined and processed on GPUs.

5. Future Work
A second generation of IceProd designed to be more robust and flexible is currently under development. The database in this new implementation is partially distributed to remove the single point of failure and to better handle higher loads. Caching of files is more prevalent and easier to implement to optimize bandwidth usage. Tasks are completely native and fully supported at all levels. These changes should allow for interoperability between grids and more productive use of resources.

6. Conclusions
IceProd has been a success for managing simulation production and data processing for IceCube across a heterogeneous collection of individual grid sites and batch computing clusters through
the use of existing trusted grid technology and network protocols. IceProd has been used to process multiple years of IceCube data and generate Petabytes of Monte Carlo simulations by integrating multiple computing sites around the world. It has also allowed for seam-less integration of clustered GPU resources into a distributed grid environment. IceProd server plug-ins allow users to easily extend the functionality of the code by providing an interface that abstracts the details of job submission and management in different grid environments. IceProd was originally developed for managing IceCube productions but it is general enough for many types of grid applications and there are plans to make it generally available for the scientific community in the near future.

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, Compute Canada and Compute West High Performance Computing; 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.

References
[1] F. Halzen. IceCube A Kilometer-Scale Neutrino Observatory at the South Pole. In IAU XXV General Assembly, Sydney, Australia, 13-26 July 2003, ASP Conference Series, Vol. 13, 2003, volume 13, pages 13–16, July 2003.
[2] Francis Halzen and Spencer R. Klein. Invited Review Article: IceCube: An Instrument for Neutrino Astronomy. Review of Scientific Instruments, AIP, 81, August 2010.
[3] J. C. Díaz-Vélez. Management and Monitoring of Large Datasets on Distributed Computing Systems for the IceCube Neutrino Observatory. In ISUM 2011 Conference Proceedings., San Luis Potosí, Mar. 2011.
[4] Dave Winer. XML/RPC Specification. Technical report, Userland Software, 1999.
[5] M. G. Aartsen and et al. The iceprod framework: Distributed data processing for the icecube neutrino observatory. Submitted to the Journal of Parallel and Distributed Computing, arXiv:1311.5904 [cs.DC], 2013.
[6] J. C. Díaz-Vélez. A Framework for management of distributed data processing and event selection for the IceCube Neutrino Observatory. Master’s thesis, Boise State University, Boise, Idaho, USA, 2013.
[7] Peter Couvares, Tevik Kosar, Alain Roy, Jeff Weber, and Kent Wenger. Workflow in Condor. In Workflows for e-Science, 2007.
[8] M.G. Aartsen et al. Measurement of South Pole ice transparency with the IceCube LED calibration system. Nucl.Instrum.Meth., A711:73–89, 2013.
[9] The IceCube Collaboration. Study of South Pole Ice Transparency with Icecube Flashers. In Proceedings of the 32nd International Cosmic Ray Conference, Beijing, China 2011, International Cosmic Ray Conference, Beijing, China, 2011.
[10] Dmitry Chirkin. Photon tracking with GPUs in IceCube. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 725(0):141–143, 2013. VLV+T 11, Erlangen, Germany, 12 - 14 October, 2011. 5th International Workshop on Very Large Volume Neutrino Telescopes, The future of high-energy neutrino astronomy.