EPJ Web of Conferences **214**, 03058 (2019) https://doi.org/10.1051/epjconf/201921403058
CHEP 2018

IceProd - a dataset management system for IceCube: update

David Schultz[1,*], and *Juan Carlos* Díaz Vélez[1]

[1]Wisconsin IceCube Particle Astrophysics Center, University of Wisconsin-Madison, 222 W Washington Ave, Suite 500, Madison, WI 53703, USA

Abstract. IceCube is a cubic kilometer neutrino detector located at the south pole. IceProd is IceCube's internal dataset management system, keeping track of where, when, and how jobs run. It schedules jobs from submitted datasets to HTCondor, keeping track of them at every stage of the lifecycle. Many updates have happened in the last years to improve stability and scalability, as well as increase user access. Along the way, the IceProd codebase switched from Python 2 to Python 3.

1 Introduction

The IceCube detector [1] is located at the geographic South Pole and was completed at the end of 2010. 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. The IceProd [2] software framework was developed by IceCube in 2006 to manage distributed simulation workloads, primarily for data provenance and dataset submission to HTCondor [3]. It has been used in production for 12 years to handle the data processing and simulation needs for the IceCube collaboration. It has run thousands of CPU years and stored over four petabytes of data. This article describes improvements in scalability and multi-user support, formally called the IceProd 2.4 release.

2 IceProd 2.4 upgrade

The primary motivation for the IceProd 2.4 update is to increase scalability. After a year of use of the 2.x series, bottlenecks with large datasets were apparent. While manual intervention every day could "solve" operational issues, this was not sustainable. Two main problems were identified: the complex task of queueing jobs, and the split database between the central master and the client causing multiple related issues (as seen in figure 1).

The split database problem was a design choice from several years ago, before the advent of a unified worldwide grid using HTCondor and glidein software [4]. Besides causing a somewhat split-brain problem, and lots of syncing between the databases, the Client database was SQLite – not the most performant database at large scale.

* Corresponding author : david.schultz@icecube.wisc.edu

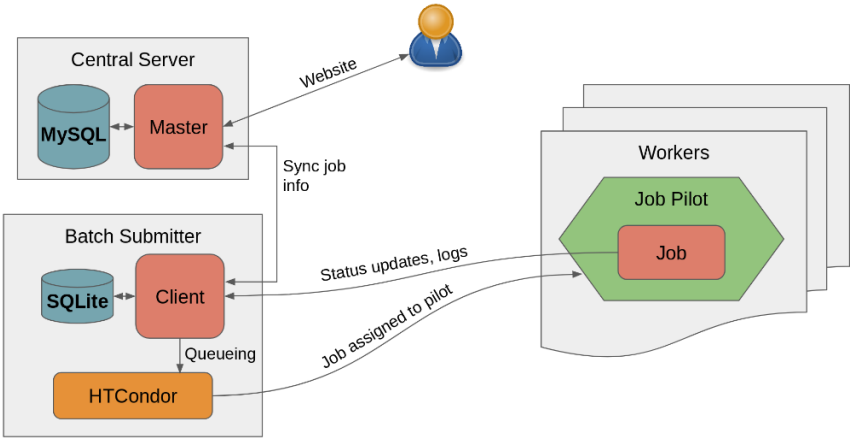

Fig. 1. A diagram of communications between different parts of IceProd, before the 2.4 release.

2.1 Move to Python 3

It was decided to use the opportunity of a substantial rewrite in the server code to move to Python [5] 3.6, the latest version of Python at the time (subsequent versions have been released, and are being used). Python 2.7 is nearing end-of-life at the beginning of 2020, and Python packages are dropping support for it. Several features in Python 3.6 would also simplify the code design, asynchronous programming being the largest one as parts of the server were already using the Tornado [6] asynchronous framework.

2.2 Creation of a REST API

Removing the split between master and client required the master to pick up a lot of functionality. Previously, communication was handled by a special JSON-RPC [7] handler in the website. This handler was becoming overloaded, since every function was using it. Traceability was also a problem, with the actual function being called buried in the request body.

Additionally, users have been requesting a better way to get data in a machine-readable format than the website pages. The trend in industry is to provide a REST API, which would nicely solve both problems. It would split requests across many urls, and potentially across different processes or servers.

The new REST API in IceProd 2.4 can be seen in figure 2. This also enabled a simpler create, read, update, and delete abstraction, speeding development and hopefully lowering the code complexity. One of the criticisms of IceProd 2 was the maze of code and difficulty of debugging, which this API should substantially reduce.

2.2.1 Database update

Although not strictly necessary, this was a golden opportunity to switch away from MySQL [8] and to MongoDB [9] for databases. IceProd 2 has always had a very simple table structure with almost no joins, and with the REST API this was even more so. This maps better to MongoDB's document structure, especially for searching over nested data that may or may not be present – relevant for queueing jobs. One other nice feature of the database redesign was forcing separation between all tables, allowing them to be located in separate databases

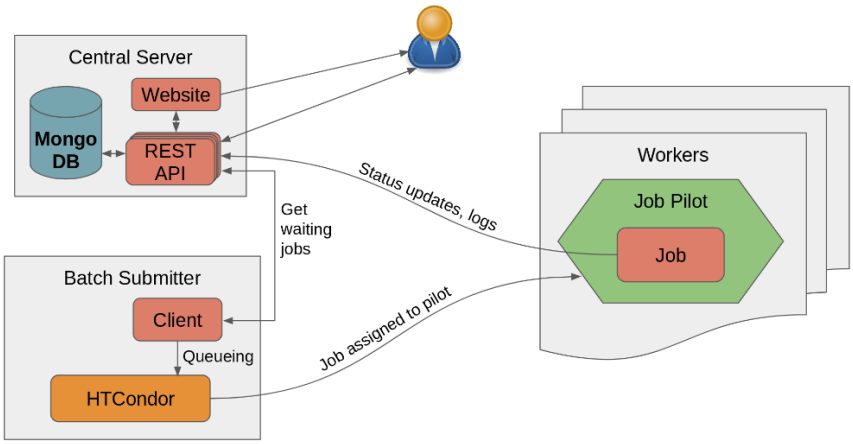

Fig. 2. A diagram of communications between different parts of IceProd, after the 2.4 release. The Client has reduced functionality, with the central REST API handling most queries.

for increased scalability. Finally, MongoDB tables can be easily sharded across machines if necessary, leaving that scaling problem separate from IceProd.

2.2.2 Scaling possibilities of REST API

One of the advantages of the new REST API is the ability to create more instances if the scaling bottleneck is within the API itself. Each call to MongoDB is completely separate, with no transactions or locking. Multiple calls in parallel are not an issue, so as long as the database can keep up the API can scale with the load.

2.3 Authorization

Earlier versions of IceProd 2 continued the model from IceProd 1, with very little authorization needed to access the whole system. While this was acceptable for production uses, physics analyzers and working groups have more and more wanted to use the same system as production. Because every external method needs authorization controls, this work kept being delayed. The new REST API was a good time to implement this, since every method would need to be rewritten anyway.

IceProd 2.4 institutes two types of authorization – role based access control (RBAC) and attribute based access control (ABAC). Roles are mostly to designate admin, user, or system-level credentials. The bulk of user accesses will use attributes, specifically whether a user is a member of the group that owns a dataset.

Every access to the REST API is authenticated using JSON Web Tokens (JWT) [10]. A token is obtained either by a user logging in using their IceCube LDAP credentials, or a sub-token of that original token. Admins can create system tokens, necessary for starting the Website and Client. Jobs are given individual short-term limited-access tokens (that have a restricted role) to communicate with the REST API.

JWT was chosen specifically for its ability to store information inside the token in a secure manner. This allows every REST API component to have immediate access to the token's role and attribute information without doing a database lookup, eliminating one bottleneck. Since all system requests use RBAC, no further authorization information is needed for the majority of API queries. For ABAC requests, an additional lookup is necessary to determine if the attributes of the resource match the token.

3 Conclusions

IceProd 2.4 was released to production a short while ago, so no strong statements about usage can be made so far. But initial tests and usage have shown several advantages in terms of scheduling, the place where most bottlenecks were occurring. It is hoped that the changes with the new REST API have created a more favourable environment for growth.

Future work will be to find and fix the next scaling bottlenecks. As such, monitoring is a priority. More long-term, it is known that the central scratch server, which stores intermediate files between dependent jobs, will hit a bandwidth bottleneck. The scratch space will need to be distributed amongst multiple servers, preferably at different sites. This is an area of active interest, which we hope to follow up on in the future.

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 Center for High Throughput Computing (CHTC) 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; 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. M. G. Aartsen et al, JINST **12**, P03012 (2017)
2. M. G. Aartsen et al, J. Parallel Distrib. Comput. **75**, 198–211 (2015)
3. D. Thain, T. Tannenbaum, M. Livny, Concurrency and Computation: Practice and Experience **17**, 323–356 (2005)
4. D Schultz, B Riedel, G Merino, J. Phys.: Conf. Ser. **898**, 092018 (2017)
5. Python project, “Python” [software], version 3.7.0, 2018. Available from https://www.python.org/downloads/release/python-370/ [accessed 2018-10-03]
6. Tornado web server, “Tornado” [software], version 5.1.1, 2018. Available from https://github.com/tornadoweb/tornado/releases/tag/v5.1.1 [accessed 2018-10-03]
7. JSON-RPC 2.0 specification, http://www.jsonrpc.org/specification
8. MySQL server, “MySQL” [software], version 5.5.61, 2018. Available from https://dev.mysql.com/downloads/mysql/5.5.html [accessed 2018-10-03]
9. MongoDB server, “MongoDB” [software], version 3.6.8, 2018. Available from https://github.com/mongodb/mongo/releases/tag/r3.6.8 [accessed 2018-10-03]
10. M. Jones, J. Bradley, N. Sakimura, RFC 7519 (2015)