jade: An End-To-End Data Transfer and Catalog Tool

To cite this article: P Meade 2017 *J. Phys.: Conf. Ser.* **898** 062050

jade: An End-To-End Data Transfer and Catalog Tool

P Meade

Systems Programmer, Wisconsin IceCube Particle Astrophysics Center, USA

E-mail: pmeade@wipac.wisc.edu

Abstract. The IceCube Neutrino Observatory is a cubic kilometer neutrino telescope located at the Geographic South Pole. IceCube collects 1 TB of data every day. An online filtering farm processes this data in real time and selects 10% to be sent via satellite to the main data center at the University of Wisconsin-Madison. IceCube has two year-round on-site operators. New operators are hired every year, due to the hard conditions of wintering at the South Pole. These operators are tasked with the daily operations of running a complex detector in serious isolation conditions. One of the systems they operate is the data archiving and transfer system. Due to these challenging operational conditions, the data archive and transfer system must above all be simple and robust. It must also share the limited resource of satellite bandwidth, and collect and preserve useful metadata. The original data archive and transfer software for IceCube was written in 2005. After running in production for several years, the decision was taken to fully rewrite it, in order to address a number of structural drawbacks. The new data archive and transfer software (JADE2) has been in production for several months providing improved performance and resiliency. One of the main goals for JADE2 is to provide a unified system that handles the IceCube data end-to-end: from collection at the South Pole, all the way to long-term archive and preservation in dedicated repositories at the North. In this contribution, we describe our experiences and lessons learned from developing and operating the data archive and transfer software for a particle physics experiment in extreme operational conditions like IceCube.

1. Project environment

The Geographic South Pole is one of the most isolated locations on the planet. It is accessible by plane only between late October and early February of each year. The 3000 meter elevation indicates all the challenges of working at a high-altitude site. The extreme southern latitude (90 degrees South) indicates extreme cold, no humidity, and six months of daylight followed by six months of darkness. Electricity is generated from jet fuel flown to the South Pole Station. Water is provided by melting ice, and is severely rationed except for drinking and washing hands. All food and other goods must be imported to the site. All waste must be removed from the site. Internet access is limited to periods of satellite visibility, bandwidth is limited, and traffic is heavily shaped.

Due to the isolation and harsh conditions of the Geographic South Pole, there is a rigorous physical and psychological qualification (PQ) process to select candidates to live and work at the Geographic South Pole during the austral winter. These Winter Over (WO) personnel are effectively trapped at the Geographic South Pole until it becomes accessible again in late October. For their own health, WO personnel are not allowed to winter over in two sequential seasons.

The IceCube Neutrino Observatory (IceCube) [1] instruments a cubic kilometer of ice at the Geographic South Pole. Approximately 5400 Digital Optical Modules (DOMs) detect and amplify faint light signals deep in the ice volume. Those signals are communicated to the IceCube Laboratory (ICL), a building which houses the South Pole Systems (SPS) data center. An online filtering farm (PnF) runs on the hosts in the data center, selecting ~10% (~100 GB/day) of the data for transmission to the data warehouse at the University of Wisconsin-Madison (UW-Madison).

Both the full data set (PFRAW) (~1 TB/day) and the filtered data set (PFFILT) (~100 GB/day) are provided to the Java Archive and Data Exchange (JADE) data transfer system for processing. JADE catalogs the provided data in a MySQL database, and then handles the data according to its configuration. PFRAW data are archived to multiple copies of archival media. These archival media are shipped back to UW-Madison during the austral summer, when the Geographic South Pole is accessible by cargo plane. PFFILT data are packaged into ~1 GB bundles and scheduled for transmission via the South Pole TDRSS Relay (SPTR) system. As of 2017, the IceCube project has an allocation of 105 GB/day for transmission via SPTR.

The IceCube project hires and deploys two WO operators to the Geographic South Pole every year. They are responsible for ensuring the continued operation of the detector and maintaining the computer systems in the SPS data center. One of their tasks is to operate the JADE data transfer system, ensuring that fresh archival media are always available for PFRAW data archival, and that PFFILT data is regularly sent to SPTR. Alternately, they may be required to withhold data from SPTR if/when satellite communications are not available.

2. Evolution of the data transfer system

The original data transfer system used by IceCube was the South Pole Archive and Data Exchange (SPADE) software [2]. SPADE was replaced with the first version of JADE (stylized JADE1) to allow for easier maintenance and expansion of the data transfer software. Those goals not realized with JADE1, it was heavily refactored to create the second version of JADE (stylized JADE2). The success of JADE2 prompted the replacement of the north-side data transfer software (Ingest) with another instance of JADE2 (stylized jadenorth).

The success of jadenorth prompted the creation of yet another instance of JADE2 (stylized "jade Long Term Archive"). As of this writing, efforts are underway to unify these JADE2 instances through a shared File Catalog metadata service. Unfortunately, the length of this paper does not allow us to cover these latter two projects in any detail.

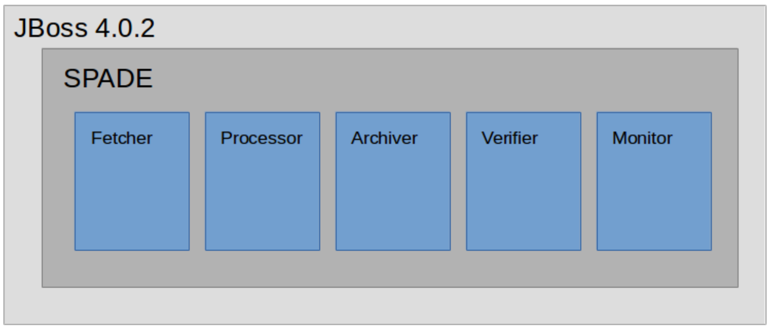

Figure 1. Architecture of SPADE

2.1. SPADE: Refactoring vs. Rewriting

Software projects require a significant investment of resources and effort. The decision to completely rewrite software is likewise a significant decision to take. Conventional wisdom is that software developers always prefer to rewrite, and that it is a bad decision to do so [3]. *Refactoring software* is often proposed as an alternative to rewriting software. Refactoring is an in-place restructuring; a way to improve the flexibility and maintainability of the software without discarding the software and all of the prior investment therein.

The decision to completely rewrite SPADE was taken after consideration of the effort required to refactor the software. The software itself was well structured, but it was beginning to show its age. SPADE ran as a daemon process inside JBoss Application Server (now called WildFly). The original design of IceCube called for all components to support Java Management Extensions (JMX) for intercommunication. When SPADE was assessed, it was the only JMX supporting software left in the entire project. Keeping the WildFly application container would have meant significant refactoring of the code to support the latest version for security reasons.

The code itself performed well, but Hashtable and Vector were used in the code instead of the more modern List and Map of the Java Collections API. Synchronization between threads was handled via hand-written synchronized blocks, as opposed to division into independent Runnable tasks that could be added to a thread pool. The database code was tightly coupled to a PostgreSQL database through SQL queries embedded in and constructed by the code. No dependency injection framework was used in the software and most systems and subsystems were constructed as singletons. All of these things made it more difficult to expand SPADE or integrate it with more modern technologies.

After identifying the code to be refactored, it became clear that the effort to refactor SPADE would equal or exceed the effort required to rewrite the software. When refactoring itself becomes a complete rewrite, the choice is no longer between refactoring and rewriting, but rather between rewriting piecewise or rewriting de novo. There are motivations for both choices which are outside the scope of this paper. For the data transfer software, the decision was taken to rewrite de novo.

2.2. JADE1: SPADE as a job engine

SPADE was a daemon process with subsystems dedicated to specific purposes: Fetcher, Processor, Archiver, Verifier, and Monitor. The idea in the JADE1 rewrite was to break these subsystems down into step-by-step jobs and arrange them as workflows. A host running JADE1 is a simple general purpose job engine to handle data transfer tasks as jobs. JADE1 used Hibernate ORM to store job information in and retrieve job information from a MySQL database. Any number of JADE1 hosts could access the database and contribute to job throughput. A Storage Area Network (SAN) would provide shared storage between JADE1 hosts and allow them to cooperate with one another.

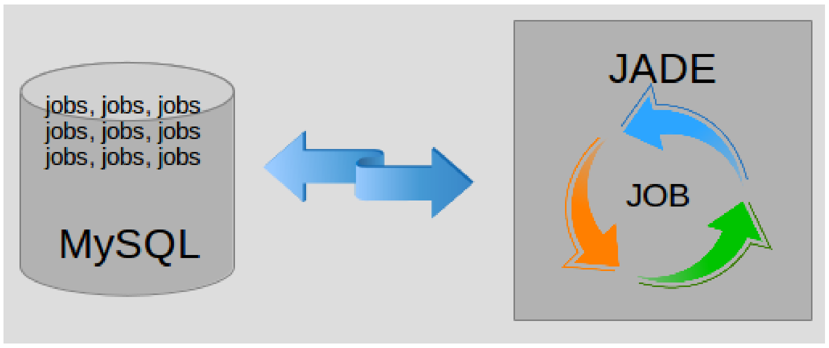

Figure 2. Architecture of JADE1

This design failed to achieve its goals for several reasons. First, although the SAN was deployed, it had insufficient performance in production so it was replaced with storage directly attached to the JADE hosts. This lack of shared storage meant that JADE hosts could not cooperate on data transfer tasks. Once a file had been copied to a particular JADE host, another JADE host could do nothing with it. Second, the job engine had very poor load balancing characteristics. JADE hosts often “bit off more than they could chew” in terms of data transfer tasks while other JADE hosts sat idle. Because of the lack of shared storage, the idle JADE hosts could not assist the overloaded JADE host.

Finally, the importance of workflow management was underestimated. Operators did not have the commands they needed in order to reschedule broken workflows or remove obsolete workflows from the JADE system. Often, a northern expert (the JADE developer) would need to wait for the satellite to rise, login to the SPS, and execute ad-hoc SQL commands in job database.

JADE1 ran concurrently with SPADE, but managed only to take over half the duties of SPADE. Specifically, the archival of PFRAW data to archival media. SPADE continued to handle scheduling the PFFILT data for satellite transmission to the north during the lifetime of JADE1.

2.3. *JADE2: Jobless JADE1*

JADE2 is a heavily refactored JADE1. The data transfer task code (copying files, calculating a checksum on a file, etc.) worked well. The scheduling of these tasks with a poorly written job engine was identified as the key problem in JADE1. The primary refactoring in JADE2 was to remove the job engine (creating a “Jobless JADE”) and reorganize the data transfer tasks back into SPADE-like subsystems.

The primary difference between SPADE and JADE2 was that SPADE gathered its subsystems into a single daemon process. JADE2 made each subsystem its own daemon process (stylized JADE component) so that each subsystem could be managed directly with existing commands. Moreover, each JADE component could be monitored independently for performance or lack thereof.

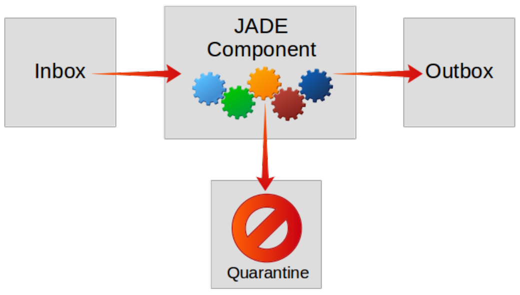

Figure 3. Architecture of a JADE2 component

The operational improvement was immediately evident. Where JADE1 lacked even fundamental commands like determining how many active jobs were in the database awaiting execution, JADE2 components could report the size of their inbox, or the operator could just run the ls command on the inbox directory of the component. A Nagios [5] monitoring system keeps track of every JADE component and informs the operators if anything is outside of defined control limits.

The performance improvement was also immediately evident. Errors anywhere in the data transfer system would often cause a backlog of work for JADE1. It would take several days to recover for each day of backlogged data. At times, a 7+ TB backlog (a week worth of data) would be present on the online filtering system. Since the deployment of JADE2, the data backlog has never reached the 1.5 TB threshold for an alert in the monitoring system. It is extremely rare for the disk occupancy of the online filtering system to exceed 50 GB (~1 hour worth of data).

The operational and performance benefits describe a single JADE2 host. Additional hosts can be added and cooperate well with the existing hosts. In practice, IceCube runs a single JADE2 host because it is more than sufficient to handle all the aspects of data archival and transfer. The other JADE hosts are powered down, but stand ready in case the primary JADE host requires maintenance, or if another large data backlog should appear again for any reason.

3. JADE in the SPS

In the SPS, there are eight JADE components that manage the data archival and transfer processes. This section enumerates those components and which mechanisms enable them to operate in an efficient and robust manner.

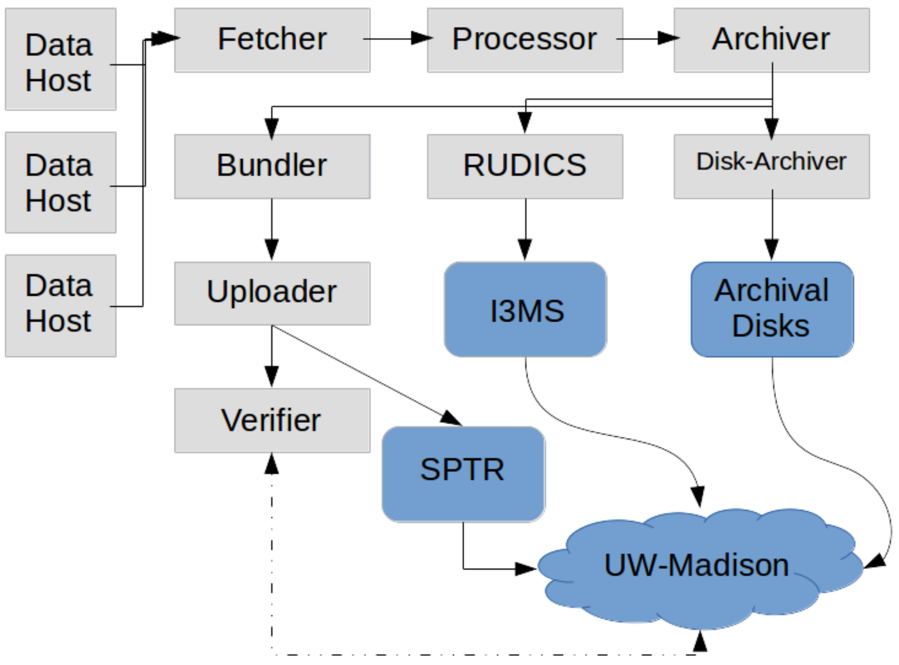

Figure 4. Architecture of JADE in the SPS

3.1. *Fetcher*

The Fetcher component is responsible for scanning SPS data hosts for files to process. A data stream is defined by a data host, a dropbox directory, a file name prefix, a binary file suffix, and a semaphore file suffix. When a data customer provides a pair of files (the binary file and the semaphore file) in the dropbox directory, it is considered ready for pickup by JADE.

Unlike most JADE components, the Fetcher has ~100 inbox directories and they reside on remote hosts in the SPS. The Fetcher employs a thread pool with parallel worker threads to ensure that a large backlog in one dropbox directory does not starve other dropboxes.

JADE hosts cooperate with one another by means of racing to record the file in the metadata database. The first JADE host to do so "wins" the right to process the file, which it immediately begins doing. The other hosts (if any) will receive an insert error from the database and attempt to move on to the next file. The winner also records a "fetching note" for itself in a local inbox directory. This way, if the component is restarted for any reason, it will be able to remember that it was in the middle of processing a file that it had previously "won".

For data integrity, the Fetcher will run a sha512sum checksum command on the remote data host, use scp to copy the data file from the remote host to itself, run another sha512sum checksum command on its local copy, ensure that both checksums match, and only then delete the original file from the dropbox directory on the remote host. By recording the checksums in the metadata database, we can verify that the file that we provide back to a data customer in the future is identical to the file that they provided to us in the SPS.

3.2. Processor

The Processor component is responsible for ensuring each binary file has appropriate metadata. It then compresses the binary and metadata files into a single archive file. Some data streams provide metadata, but most do not. Apart from this logic to generate metadata when it is not provided, the processor is basically a wrapper over a tar command.

3.3. Archiver

The Archiver component is an archive file multiplexer. Depending on configuration, data files may be sent on the Router-Based Unrestricted Digital Internetworking Connectivity Solutions (RUDICS) low latency link, bundled to be sent via satellite (SPTR), and/or written to archival disks for shipment back to the north. The Archiver looks at the configurations for these communication channels, and provides copies of the archive file to the appropriate downstream component inbox directories.

Originally, for performance purposes, all copies were made by creating hardlinks. An odd race condition occurred where a downstream tar command was operating on one hardlink, while another downstream component would delete a different hardlink to that same file. The tar command would error out, reporting that the file had changed during processing. The Archiver was modified to provide a true copy to the downstream component using tar.

3.4. Disk-Archiver

The Disk-Archiver component writes files to archival disks. A configurable set of paths is used to identify eligible archival media. An identity file is written to the disk and then files are copied to the archival disk. A configurable number of copies are made. The archival disks themselves are never one-to-one copies of one another, but each copy set of disks contains a complete copy of the data.

The Disk-Archiver component retains each of the files in an internal buffer on the JADE host until all the copies have been completely written. If an archival disk fails for any reason, a "regenerate" command can be issued to make the media as bad. The Disk-Archiver component will then select new media to replace the bad copy, and rewrite the files from its internal buffer.

When an archival disk is very nearly full, a metadata file is written to the root of the drive. This way the contents of the archival disk can be processed without reference to the metadata database, such as occurs when the disk arrives in the north and the metadata database is still in the SPS.

3.5. RUDICS

The RUDICS component writes files to the IceCube Message System (I3MS). This is a low-latency link with very limited bandwidth. It is intended for important status information, and there are configurable transfer size limits enforced by the component.

I3MS provides a ZeroMQ [6] interface for communication. I recommend that you do NOT use ZeroMQ in your projects if at all possible. ZeroMQ does its best to hide errors. This makes designing reliable message transport difficult without a significant investment of effort. Always prefer a WebSocket or a RESTful API instead of ZeroMQ for most use cases.

3.6. Bundler

The Bundler component creates ~1 GB bundles for upload to the SPTR system. The 1 GB size allows the National Science Foundation (NSF) contractors to manage satellite bandwidth effectively. The Bundler is also responsible for splitting very large files into smaller bundles.

The Bundler runs a sha512sum checksum on each bundle, and then records the checksum in the metadata database. The bundle is then moved to the inbox directory of the Uploader component.

3.7. Uploader

The Uploader checks the SPTR system to ensure there is space available. It then uploads bundles until available space is consumed or it runs out of bundles to upload. The satellite transfer bandwidth is shared between several projects at the South Pole. The Uploader allows us to be good satellite neighbors, because we retain our backlog on our hosts and never flood SPTR with excessive data.

3.8. Verifier

The Verifier checks with a web service (salamander) provided in the north. It asks the north for the checksums that the north computed for the bundles. If the checksum matches, the north has successfully received the bundle. The bundle is then deleted from the JADE host in the SPS.

The monitoring system checks to ensure that the bundles in the cache are no more than three (3) days old. This will activate the operators to do something about old unverified bundles. If a backlog is expected, this may be just to acknowledge the alert and wait. If transfer error occurred, a bundle may moved back to the Uploader inbox for another chance at transfer to the north.

4. JADE in the north

In the WIPAC data centers at UW-Madison, there are four jade components that manage data transfer and warehousing. This section enumerates those components and which mechanisms enable them to operate in an efficient and robust manner.

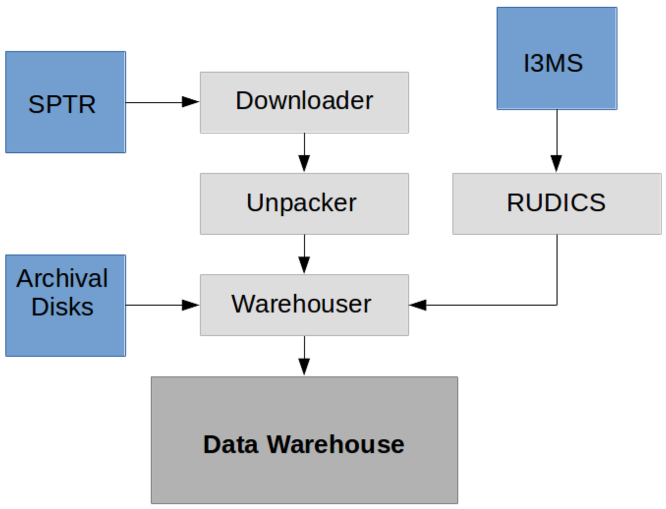

Figure 5. Architecture of JADE at UW-Madison (jadenorth)

4.1. Downloader

The Downloader checks the SPTR system for data that has arrived in the north. It downloads the bundles to the JADE host and then run a sha512sum checksum on the bundle. The checksum is recorded in the metadata database and the bundle file is passed to the Unpacker.

4.2. Unpacker

The Unpacker takes bundle files and processes them. Regular bundles have their archive files unpacked, sha512sum checksums run on them, and then they are passed to the Warehouser component. Split bundles are reassembled into a large bundle, and then the very large individual file is extracted and processed as described before for smaller archive files.

4.3. Warehouser

The Warehouser takes archive files and copies them to the appropriate location in the Data Warehouse. It then runs a sha512sum checksum on the file in the Data Warehouse to ensure the archive file was properly copied to its final destination. After verification of the checksum, the archive file is deleted from the JADE host.

4.4. RUDICS

The RUDICS component listens to the IceCube Message Service (I3MS) for files sent via the RUDICS low-latency link. When a file is received, it is handed to the Warehouser component for processing as per a normal archive file.

Note that despite having identical names, the RUDICS component in the SPS is different from the RUDICS component in the north. The roadmap for JADE2 is to someday refactor this naming, but it has not yet been a high priority to do so.

5. Acknowledgements
I would like to thank Gonzalo Merino for his encouragement to present at CHEP 2016. I would also like to thank Ralf Auer for *Ralf's "Basic JADE in less than 100 lines of bash..."* design document which inspired the design of JADE2. Finally, I would like to thank the Flying Spaghetti Monster for creating the universe [4], and touching all of our projects with His Noodly Appendage. Ramen.

References

[1] Aartsen MG, Ackermann M, Adams J, Aguilar JA, Ahlers M, Ahrens M, Altmann D, Andeen K, Anderson T, Ansseau I, Anton G. 2017 The IceCube Neutrino Observatory: Instrumentation and Online Systems. *Journal of Instrumentation*. **Volume 12**. 2017 Mar 14.
[2] Patton S, Samak T, Tull CE, Mackenzie C. Spade: Decentralized orchestration of data movement and warehousing for physics experiments. InIntegrated Network Management (IM), 2015 IFIP/IEEE International Symposium on 2015 May 11 (pp. 1014-1019). IEEE.
[3] Spolsky J. Things You Should Never Do, Part I [Internet]. Joel on Software. 2000 [cited 2017 Feb 7]. Available from: https://www.joelonsoftware.com/2000/04/06/things-you-should-never-do-part-i/
[4] Henderson B. Church of the Flying Spaghetti Monster [Internet]. Church of the Flying Spaghetti Monster. Church of the Flying Spaghetti Monster; [cited 2017Feb19]. Available from: http://www.venganza.org/
[5] The Industry Standard In IT Infrastructure Monitoring [Internet]. Nagios. Nagios Enterprises, LLC; [cited 2017Feb19]. Available from: https://www.nagios.org/
[6] zeromq [Internet]. Distributed Messaging - zeromq. iMatix Corporation; [cited 2017Feb19]. Available from: http://zeromq.org/