Improved Pilot Logging in DIRAC

Simon Fayer[1,*], *Janusz* Martyniak[1], and *Federico* Stagni[2,**]

[1]Blackett Laboratory, Department of Physics, Imperial College London, London SW7 2AZ, United Kingdom

[2]CERN, EP Department, Geneva, Switzerland

Abstract.
DIRAC is a widely used framework for distributed computing. It works by building a layer between the users and the resources offering a common interface to a number of heterogeneous providers. DIRAC, like many other workload management systems, uses pilot jobs to check and configure the worker-node environment before fetching a user payload. The log messages generated by these pilot jobs are crucial for diagnosing problems in the infrastructure. In these proceedings we present a configurable, resource independent remote pilot logging system.

1 Introduction

The DIRAC[1] workload management creates an interface layer between end user and compute resources. User jobs are submitted to a common interface and then brokered out to a number of heterogeneous resource providers. DIRAC supports all of the common grid compute elements (CEs) such as HTCondor-CE and ARC-CE, as well as a number of different cloud interfaces and other providers. The brokering works by submitting pilot jobs [2] to the target resources which perform a number of verification tasks on the worker node before pulling one or more user jobs from the central queue. A single DIRAC instance can support multiple Virtual Organisations (VOs) with overlapping resources, although a pilot is always bound to a particular VO at submission time.

The pilot jobs generate log files which are primarily accessed for debugging if there are issues with a particular resource; these log files are stored in a resource dependent manner. On a grid CE, the pilot writes logs to stdout/stderr which are captured by the batch system and can later be retrieved using a CE specific tool. For a cloud resource the logs are typically written to a file on a given virtual machine instance where there is no standard or simple way for them to be retrieved. There is also no fixed retention policy for any of these log files; they are generally stored for a site-specific amount of time that can range anywhere from hours to weeks. It is not uncommon to find that the log files have been purged by the time a problem is identified and needs investigating.

The DIRAC frontends, the web interface and command line interface, offer the DIRAC administrators a facility to fetch the pilot logs for a given pilot job. This is a wrapper around

*e-mail: lcg-site-admin@imperial.ac.uk
**e-mail: federico.stagni@cern.ch

the system-specific tools to fetch the pilot log; in the cases where no interface exists a “not implemented” error is returned.

The aim of this project is to implement a new resource agnostic logging system to ensure that the pilot logs are captured and made readily accessible for all resources as an extra debugging facility in parallel with the existing CE-based logging system. It will also offer the ability to preview logs while the pilot is running and, in the future, send them to a standard log analysis platform using an appropriate back-end plugin.

2 Design and Implementation

The design of the new pilot logging system for DIRAC is based around having the pilot jobs periodically send their logs back to a central storage service. This is done in two stages; the pilot jobs must send the log entries back to a collector and then the collector must forward them on to a back-end storage archive. For some back-ends the log messages must be collated while the pilot is running so that only the complete log file is sent to the archive. The archiving is done through a plugin based system as different DIRAC instances may have different requirements. An overview of the new logging system can be seen in Figure 1.

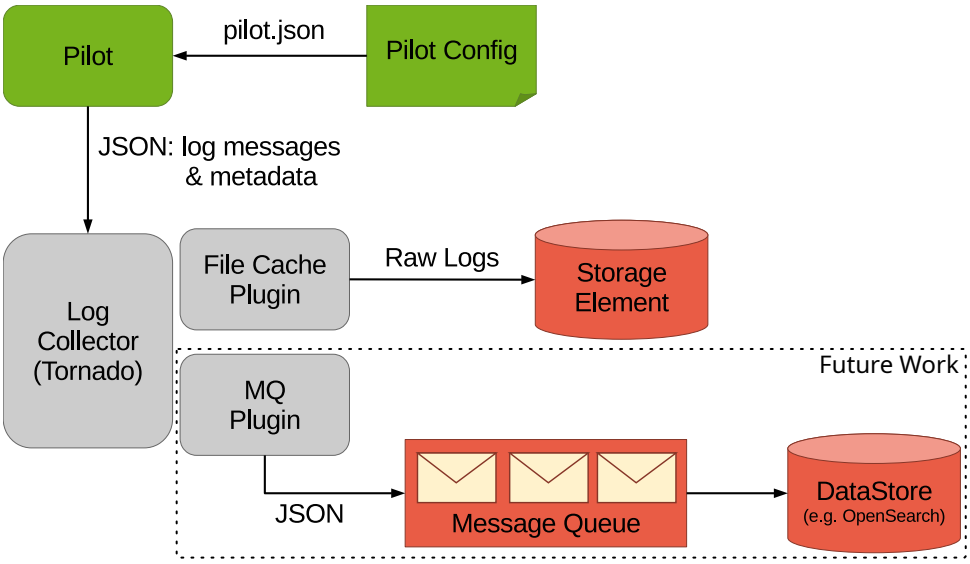

Figure 1. The remote pilot logging system. The config file sets whether the remote logging should be enabled and contains the collector service URL. The collector configuration selects one of the plugins to archive the log records. A message queue plugin is planned but not implemented yet.

The existing DIRAC pilot code uses a logging class which was extended to write the logs to a remote logging buffer as well as the pre-existing logging locations. This allows for backwards compatibility as the other log locations are also still populated with messages. As the pilot job starts, it fetches a configuration file (pilot.json) from the central server which contains its basic configuration parameters. If the remote log collector URL is set in the parameters, the remote logging module is activated and messages are sent. If remote logging is not required the remote logging buffer is disabled and its contents are discarded. The remote logging can be enabled on a VO-by-VO basis by the DIRAC administrator; it is envisaged this will be enabled as needed for debugging rather than being left enabled permanently to

minimise the load on the system. When enabled, the remote logging module waits until a fixed number of messages have been collected; it then posts these in JSON format to the collector. The records are batched to minimise the number of calls to the collector, thereby keeping the load manageable. At the end of the job, the last message posted also includes a flag to signal that the pilot is exiting.

The collector service is written in python and is based on the Tornado web server. The pilot job starts with a credential that is created at submission time and forwarded through the submission system to the execution node. The pilot logging system authenticates with the pilot credential and sends batches of messages using HTTP POST requests. Further processing of the log entries is done by a back-end plugin; the plugin to use is selected by the collector service configuration.

A storage element (SE) storage plugin for the collector service has been created. The inbound log files are initially written to a cache area local to the collector service. A log file is designated as complete once the last message from the pilot with the finishing flag set is received; this triggers an upload of the completed file to a designated grid SE. This plugin also includes a periodic task to clean the local cache area.

A second plugin for sending to a message queue has been designed but has not yet been implemented. This plugin will immediately forward the batch of messages to a message queue system such as Apache ActiveMQ. This will allow the logs to be stored and processed using an industry standard full-text search system such as OpenSearch and the data to be visualised through Kibana of Grafana.

The administrator interface for retrieving pilot log files has also been connected to the collector. When the admin requests a pilot log from the DIRAC PilotManager service, the default resource-based method for fetching the log file is tried first; if for any reason this fails (e.g. log not available or the resource is offline) then the remote log collector is queried instead. The collector uses the configured plugin to try to retrieve the log file from the store. The order of the log sources is configurable by the DIRAC administrator, allowing the collector to be queried before the resource-based system. This fallback mechanism is completely transparent to the administrator, the log is simply fetched from whichever source has it available.

3 Status and Future Plans

The remote logging system is pending final review for inclusion in DIRAC version 8.1. The next steps will be implementing the message queue collector plugin and then building an OpenSearch dashboard to process and display the forwarded logs. The remote logging can also currently only be enabled for all pilots for a specific VO; it is planned to allow this to be enabled for just individual CEs by extending the data stored in the pilot configuration file.

The current design only allows the DIRAC administrators to view the pilot log files for all sites, which is a limitation inherited from the original logging system. A proposed extension for this project is to add a new interface for site administrators to be able to retrieve the job files for their sites. This would allow the site admins to do more debugging themselves without relying on the DIRAC admins to relay log files to them.

References

[1] A. Tsaregorodtsev, fstagni, P. Charpentier, A. Sailer, ubeda, K.D. Ciba, chaen, C. Burr, A. Lytovchenko, R. Graciani et al., *Diracgrid/dirac: v8.1.0a19* (2023), https://doi.org/10.5281/zenodo.8298362

[2] F. Stagni et al., *DIRAC universal pilots* (2017), J. Phys.: Conf. Ser. **898** 092024