EPJ Web of Conferences **214**, 03032 (2019) https://doi.org/10.1051/epjconf/201921403032
CHEP 2018

GRACC: GRid ACcounting Collector

Derek Weitzel[1,*], *Brian* Bockelman[1,**], *Marian* Zvada[1], *Kevin* Retzke[2], and *Shreyas* Bhat[2]

[1]University of Nebraska - Lincoln
[2]Fermilab, Batavia, IL, US

Abstract. The OSG has long maintained a central accounting system called Gratia. It uses small probes on each computing and storage resource in order to collect resource usage. The probes report to a central collector which stores the usage in a database. The database is then queried to generate reports. As the OSG aged, the size of the database grew very large. It became too large for the database technology to efficiently query to generate detailed reports.

The design of a replacement requires data storage that could be queried efficiently to generate multi-year reports. Additionally, it requires flexibility to add new attributes to the collected data.

In this paper we will describe updates to the GRACC architecture in the last 18 months. GRACC uses modern web technologies that were designed for large data storage, query, and visualization. That includes the open source database Elasticsearch, message broker software RabbitMQ, and Grafana and Kibana as data visualization platforms. It uses multiple agents that perform operations on the data to transform it for easier querying and summarization.

1 Introduction

In a previous paper [1], we described the GRACC modular architecture. In this paper we will describe the updates to GRACC that have occurred over the last 18 months.

The Grid Accounting Collector (GRACC), collects, processes, and visualizes usage information for the Open Science Grid. The usage information is primarily usage of compute resources and data transfer information. It collects information such as the Virtual Organization (VO) and the particular user running on the OSG, as well as the wall hours, how many cores, and the memory usage of the user.

Due to the modularity of GRACC, we have been able to use GRACC to collect other data such as data transfer information. The StashCache [2] data federation usage data and the HTCondor TCP transfer statistics are sent to and stored by GRACC. Each of these collections use a common architecture utilizing 5 components: probes, data collection, message broker, data sinks, and visualization and querying tools.

Probes query the resources for usage information and convert it into a common format that can be more easily read by the data collection component. The data collection component receives data from multiple probes, performs some manipulation of the data, and sends the data to a message bus for distribution. The message bus routes the data according to rules

---

[*] e-mail: dweitzel@cse.unl.edu
[**] e-mail: bbockelm@cse.unl.edu

that have been set, including sending copies of messages to multiple data sinks. Data sinks listen for messages on the message bus and store the data.

Each of the examples shown below follow this common architecture, probes, data collection, message broker, data sinks, and visualization and querying tools.

2 Background

The GRACC ecosystem consists of 5 main components: probes, data collection, message broker, data sinks and visualization. Figure 1 shows the overview of the GRACC system.

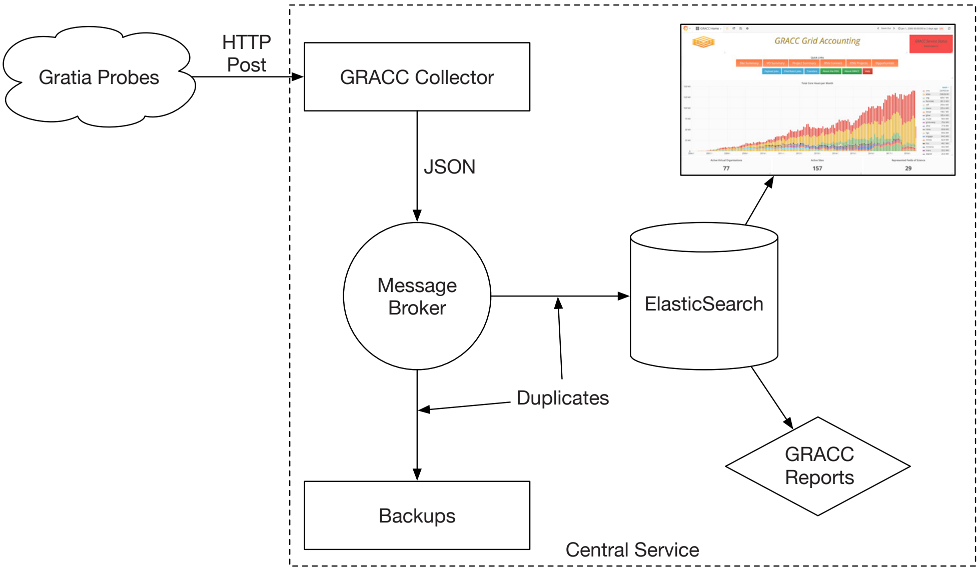

Figure 1. GRACC Overview

The probes [3] are from the original Gratia ecosystem. Each supported batch system (HTCondor, PBS, SLURM, LSF) has a corresponding probe. The probe knows how to query the batch system for data and convert it to a common format. For example, with HTCondor, the probe parses the history files of completed jobs. For the SLURM batch system, the probe queries the SLURM database for completed jobs. The resulting usage records are uploaded to the GRACC collector via an HTTP POST. All data is buffered locally until successfully sent to the GRACC data collection service.

The data collection service is a daemon written in the GO programming language [4]. The service listens for incoming records, parses, and transforms the records into an internal format. It sends the transformed record to a message broker.

The message broker distributes the records to multiple destinations. The message broker based on the RabbitMQ [5] software. It handles the durability of messaging between components in the system. It also buffers messages in case of a transient outage.

A GRACC data sink is the LogStash instances that store the data into an ElasticSearch [6] database. The database stores the records for querying and visualization.

Another data sink is the backup infrastructure. Each usage record is automatically backed up to tape in case of a failure of the database. The usage records can be replayed and the data base restored from the tape backups.

Visualization of GRACC usage records is done using either Grafana [7] or Kibana [8] visualization tools. The GRACC team maintains numerous preformatted dashboards displaying usage of the Open Science Grid.

3 Data Transfer Collection

Each cache in the OSG Federation StashCache [2] sends transfer information to a central collector for each file downloaded. Figure 2 shows an example of the StashCache data. Each colored bar is a different organization downloading data from the federation.

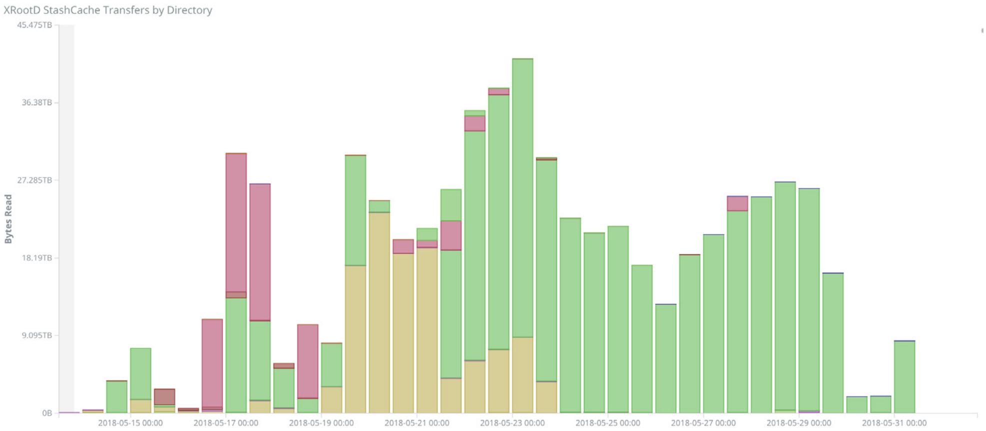

Figure 2. StashCache Data

The probes are the XRootD servers that send records to a central collector in a binary format. The data collection is a central collector that parses the binary packets, converts the message into JSON and enriches the information with GeoIP data about the sources and destinations of the transfer. The central collector sends this data to the message broker.

The data sink is a LogStash instance that listens on the message broker for incoming messages and ingests them into GRACC's ElasticSearch. Visualization of the data is performed in Kibana.

4 HTCondor TCP Collection

GRACC also collects network transfer statistics from submit hosts across the Open Science Grid. HTCondor can create a log of TCP transfer statistics for each data transfer between the submit host and the worker node for each job. The statistics include TCP attributes such as number of packets reordered, packets retransmitted. An example line is shown in Figure 3. HTCondor writes these messages to a file on the submit host.

Figure 4 shows the data flow from the HTCondor submit host to the GRACC.

We use Filebeats [9] as the probe to parse and send the TCP log lines to GRACC. Filebeats has many features that we utilize including log rotation detection and backoff pressure detection. The records are sent to the data collection component, a LogStash instance hosted on the GRACC infrastructure. The LogStash instance forwards the raw log lines to the RabbitMQ message bus.

A record enricher reads messages from the message bus, parses the log lines and adds additional data. The record enricher performs the actions:

```
11/27/18 15:10:16 (D_STATS) (124102867.13155) (1058548): File Transfer
    Download: JobId: 124102867.13155 files: 2 bytes: 612 seconds: 0.01 dest:
     128.104.101.131 rto: 205000 ato: 40000 snd_mss: 1448 rcv_mss: 682
    unacked: 1 sacked: 0 lost: 0 retrans: 0 fackets: 0 pmtu: 1500
    rcv_ssthresh: 19644 rtt: 5500 snd_ssthresh: 2147483647 snd_cwnd: 10
    advmss: 1448 reordering: 3 rcv_rtt: 1000 rcv_space: 14480 total_retrans:
     0
```

Figure 3. Example HTCondor TCP Log Message

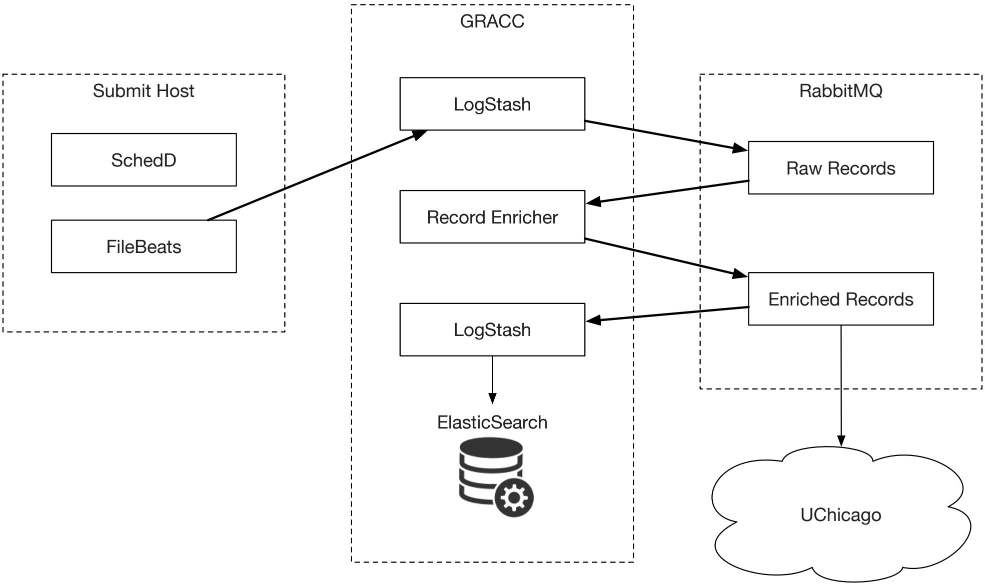

Figure 4. HTCondor TCP Dataflow

- Parse the log line, creating a key-value record.
- Annotate the record with tags for either upload or download.
- Add Geographical information to the source and destination of the transfer.

The LogStash instance combined with the record enricher act as the data collection component.

Finally, the enricher sends the key-value record back to RabbitMQ for distribution to the data sinks that have subscribed to the data. The data is ingested by LogStash, and stored in Elasticsearch. Visualization of the data is performed in Kibana.

5 Conclusion

Before GRACC, each of these different data sources would have used different infrastructure for storage and analysis. Using the same infrastructure as the GRACC accounting system, we have been able to collect, analyze, and visualize data from multiple sources. The GRACC infrastructure's flexibility allows us add agents and translators for different data sources, while using the same message broker and data store as GRACC.

Component	GRACC Usage	StashCache	HTCondor TCP
Probe	Gratia Probes	XRootD	FileBeats
Data Collection	Gratia Collector	Python Collector	LogStash & Enricher
Message Broker	RabbitMQ	RabbitMQ	RabbitMQ
Data Sinks	ElasticSearch & Backups	ElasticSearch	ElasticSearch
Visualization	Kibana & Grafana	Kibana	Kibana

Table 1. Components of the example GRACC data collections

As can be see from Table 1, the different data collections share many of the same components. Every data collection uses same RabbitMQ message broker, and stores the data in ElasticSearch. They additionally use the same Kibana for visualization. The differences in the data collections are the probes since the data comes from different sources. The data collection mechanisms are also different in order to format the data correctly for sending to the message broker and eventually injestion into ElasticSearch.

References

[1] K. Retzke, D. Weitzel, S. Bhat, T. Levshina, B. Bockelman, B. Jayatilaka, C. Sehgal, R. Quick, F. Wuerthwein, *GRACC: New generation of the OSG accounting*, in *Journal of Physics: Conference Series* (IOP Publishing, 2017), Vol. 898, p. 092044

[2] D. Weitzel, B. Lin, T. Aburaad, C. Largent, R. Illingworth, S. Thapa, B. Bockelman, R. Gardner, M. Zvada, L. Bryant et al., *Stashcache* (2017), `https://doi.org/10.5281/zenodo.557111`

[3] C. Green, P. Canal, M. Mambelli, T. Levshina, C. Edquist, B. Bockelman, S. Thapa, E. Fajardo, D. Weitzel, B. Lin et al., *gratia-probe* (2018), `https://doi.org/10.5281/zenodo.1473146`

[4] A.A. Donovan, B.W. Kernighan, *The Go programming language* (Addison-Wesley Professional, 2015)

[5] A. Videla, J.J. Williams, *RabbitMQ in action: distributed messaging for everyone* (Manning, 2012)

[6] C. Gormley, Z. Tong, *Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine* (" O'Reilly Media, Inc.", 2015)

[7] G. Labs, *Grafana* (2018), `https://grafana.com/`

[8] elastic.co, *Kibana* (2018), `https://www.elastic.co/products/kibana`

[9] elastic.co, *Filebeats* (2018), `https://www.elastic.co/products/beats/filebeat`