Monitoring Parsl Workflows

Poster Submission Summary

Connor Pigg
University of Illinois
cpigg2@illinois.edu

Daniel S. Katz
University of Illinois
dskatz@illinois.edu

ABSTRACT

As workflow software, Parsl provides users the ability to define complex workflows in simple Python to be executed in parallel on any computer system [1]. One useful feature that Parsl lacked was convenient workflow monitoring. This project enhanced Parsl with the addition of workflow monitoring. Simple and comprehensive monitoring of a workflows state and resource usage provides users value by allowing auditing, debugging, and profiling of workflow execution. The work discusses the components of workflows that are monitored by Parsl, the strategy for capturing these components, and the tools used to accumulate and display these components. Primarily, this project used Python libraries to collect task status and resource usage and log these tasks to an Elasticsearch database - a flexible and searchable indexing database [2]. A Kibana dashboard - a tool for visualizing data stored in an Elasticsearch database - was then created to visualize the collected logs in a realtime and interactive user interface (UI). The functionality implemented in this work has been included in the latest release of Parsl, allowing users the option to monitor the status and resource usage of their workflows via an Elasticsearch database and Kibana dashboard.

ACM Reference Format:
Connor Pigg and Daniel S. Katz. 2018. Monitoring Parsl Workflows: Poster Submission Summary. In *Proceedings of SC conference (SC'18).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 METHODS

The work described in this poster follows a three-part strategy. First, from within the Parsl source code, capture important information. Second, store these logs to a central location that a user can access. Finally, present the information in these logs using a dashboard accessible to a user. The particular source logging implementation uses the Python logging module CMRESHandler. An Elasticsearch instance is used for central log storage. Finally, a Kibana dashboard provides the UI. The benefit of this strategy is that it allows simple solution development and provides the needed core functionality out-of-the-box. The downside of this strategy is that it is not trivial to distribute to users and customization can be limited.

This project was performed iteratively by adding logging statements to the code and then crafting visuals for the new information. This was an efficient way to increase presented information while keeping each addition simple.

Specifically, information about the status and resource usage is generated from the Parsl source code during execution. This information is then collected on a central Elasticsearch instance. This Elasticsearch instance is then exposed to a Kibana instance. Kibana hosts the dashboards that present the information to users; they are live and interactive. Because these instances only require a simple connection, the services may be run from arbitrary systems and only require a user to have access to the Kibana instance in order to monitor the Parsl workflow.

Kibana and Elasticsearch are full tools in their own right and leave plenty of opportunity for this solution to be expanded by the Parsl developers or customized by the users running these services. In order to provide a default that satisfies the majority of users, a generic Kibana dashboard template is provided.

There are many ways information can be stored and visualized using online resources, such as other database and visualization tools. However, the information generated within Parsl must be efficiently captured for all of them. Information is primarily collected in two distinct places, by the task scheduler and by the task executors (workers). The task scheduler is aware of the overall workflow status and all scheduling events. During each of the scheduling events, useful information such as task execution state, execution location, dependencies, etc. is packaged into a log that is sent to Elasticsearch. This log packaging follows a prescribed pattern in order to smoothly interact with Kibana visualization tools. On the task executor, a new process is spawned that monitors the resource usage of the specific running task.

2 CONCLUSIONS

A monitoring solution has been added to the Parsl workflow software. This solution uses Elasticsearch and Kibana to store and present recorded and real-time workflow information. The significance of this work is that it clearly outlines a simple and effective way to provide such information to users independent of the location(s) where the workflow is run and submitted. Clear use cases for such techniques are in live monitoring of the execution of workflows that are being executed on arbitrary and potentially distributed locations, either by an individual or by a larger research team working under a single allocation of resources. This work also demonstrates how monitoring may be simplified by separating the components into specialized tools.

Future work includes additional and large-scale user testing. Feedback and experience reports will be collected and incorporated into the UI to increase the usability of monitoring. Future iterations of monitoring in Parsl will incorporate such feedback. We will also explore more customizable solutions for monitoring using the same strategies but with different visualization tools.

In summary, this work has demonstrated that remote monitoring of arbitrary software can be effectively achieved using straightforward tools such as Elasticsearch and Kibana.

SC'18, November 2018, Dallas, Texas USA
2018. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

REFERENCES

[1] Yadu Babuji, Alison Brizius, Kyle Chard, Ian Foster, Daniel S. Katz, Michael Wilde, and Justin Wozniak. 2017. Introducing Parsl: A Python Parallel Scripting Library. https://doi.org/10.5281/zenodo.853492 DOI: 10.5281/zenodo.853492.

[2] Elasticsearch BV. 2018. Getting Started with Elasticsearch/Kibana. https://elastic.co elastic.co.