Dynamic ELK Stack Monitoring for Lobster

Anna Yannakopoulos
Department of Physics, University of Notre Dame, IN 46556

Lobster

Lobster is a program that enables large-scale distributed computing on non-dedicated clusters for tasks that have complicated and intricate requirements. Managing the diverse and dynamic resources available in a non-dedicated cluster and ensuring that each task has access to the files and dependencies that it requires is a complex problem that invites a wide range of failure modes. In order for the user to detect and act upon unpredictable failure modes, Lobster must include a robust monitoring system.

Currently, Lobster's built-in monitoring produces a selection of static plot images on an auto-generated web page. These plots are hard-coded and cannot be modified without editing the Lobster source code, meaning that it is difficult for the user to add or modify plots. In the event that a previously-undiscovered failure mode is encountered, the existing plots may not provide useful insight into the cause of the problem, leaving the user to puzzle out a solution by searching log files manually.

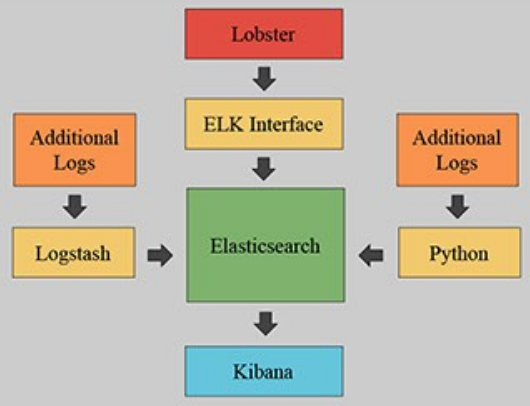
ELK Stack

The ELK software stack is an open-source monitoring system consisting of

- **Elasticsearch**, a distributed search and analytics engine
- **Logstash**, a data collection pipeline
- **Kibana**, an interactive visualization web application

Logstash and other data collection pipelines feed log files into Elasticsearch, which acts as a database and a full-text search engine queryable via a REST API. Kibana is a graphical interface for creating, executing, and visualizing Elasticsearch queries that allows the user to create dynamic graphs, charts, and tables organized into dashboards.

Below: The flow of log data between Lobster and the ELK stack.

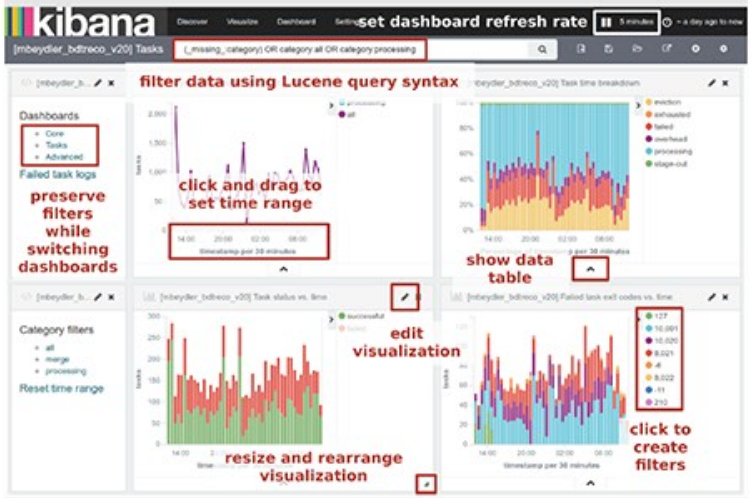


Implementation

Elasticsearch and Kibana were installed on a virtual machine and configured to run as services using ports opened to the campus secure network. Using the official Elasticsearch Python client module, an optional ELK monitoring interface was implemented within Lobster.

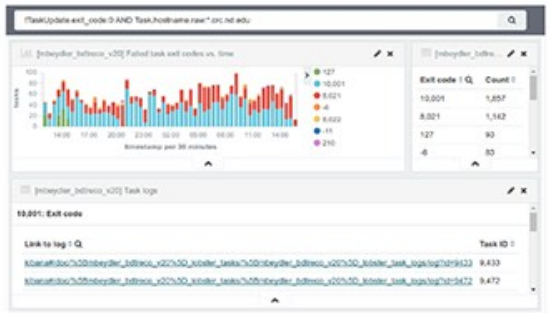
The ELK interface object populates Kibana with a set of dashboards, each containing a set of default visualizations, from locally-stored templates and provides a link to the Kibana instance containing them. New templates can be created from a user's current Kibana dashboards using the ELK interface, and the user's current Kibana dashboards can be updated to include additional templates at any time.

All Lobster log data is sent to the ELK interface object, which parses the log according to its type, performs any additional calculations, and sends the complete log file to a run-specific index in Elasticsearch. Additional data from sources other than Lobster, such as information about the campus network bandwidth usage, is collected and sent to Elasticsearch via a Logstash process or a Python daemon.



Features

- New **plots can be created** and **existing plots modified** through Kibana's graphical interface as new failure modes are encountered and plots are automatically updated as Elasticsearch receives new log data.
- **Modular dashboards** allow users to customize the monitoring system to suit their needs; modules can be created from scratch, added from a selection of templates, or removed at any time.
- User-defined **time and content filters** (written using the Apache Lucene query syntax) can be applied to individual visualizations, entire dashboards, or all log data at once, allowing specialized monitoring and isolation of interesting events.
- **New sources of log data** can be added to the ELK monitoring system using Logstash **during an ongoing Lobster run**; this will in most cases seamlessly augment previously-monitored log data.



Above: An example of using Kibana's filtering to troubleshoot problems. The visualizations show tasks that failed on Notre Dame CRC hosts and links to their full log files.

Comparison

The dynamic nature of ELK monitoring, in comparison to the static nature of current Lobster-controlled monitoring, offers new and powerful tools for quickly discovering and solving problems as they occur. ELK monitoring offers some clear benefits over the current Lobster-provided monitoring:

- **Lobster's built-in monitoring is hard-coded** and cannot be modified without editing the Lobster source code and restarting Lobster; **ELK monitoring can be modified on-the-fly** using a graphical interface.
- **Lobster's built-in monitoring does not provide any method of searching multiple log files** other than standard GNU/Linux command-line tools; **ELK monitoring allows the user to execute queries and aggregations on any part of the complete log file** and any data associated with it through a graphical interface.
- **Lobster's built-in monitoring can only display data that originated from Lobster**, meaning that important external data such as network bandwidth logs must be monitored using a different system; **ELK monitoring allows Lobster data and external data to be monitored in a central location** using the same dashboards and visualizations.
- **Lobster's built-in monitoring scales in time cost per plot with the amount of data** that Lobster has generated and thereby with the time elapsed since the run was started; **ELK monitoring does not have this problem** and performs equally well on runs of any length within reasonable limits.

ELK monitoring can improve the efficiency and speed of Lobster troubleshooting and thereby leave more time and resources for useful scientific computations.

References

1. Scaling Data Intensive Physics Applications to 10k Cores on Non-Dedicated Clusters with Lobster, *IEEE Cluster 2015*
2. Elasticsearch, Logstash, & Kibana, *www.elastic.co*
3. Elasticsearch-py, *elasticsearch-py.readthedocs.io*