23rd September 2019

This whitepaper has been produced by Ellexus, the I/O profiling company.

For more information about Ellexus' tools, including case studies, videos and blogs, visit

www.ellexus.com

I/O profiling to improve the storage performance at Diamond Light Source on a Univa Grid Engine cluster

By Liam McClean, Senior Software Engineer at Ellexus Ltd

No two high-performance computing (HPC) teams and architectures will ever be the same, but the team at Diamond Light Source handles a wider variety of workloads than many. Performance of both in-house and third-party applications is therefore vital.

The Diamond Light Source team used Ellexus Mistral to identify straightforward performance improvements that could be made in order to improve performance and cut down runtime.

> "By using Mistral, our team has already made a marked improvement to various applications that we maintain in-house. We have been able to reduce the impact of noisy neighbours, reduce runtime and identify applications with bad I/O. We intend to keep using Mistral to profile more applications and improve the overall architecture of our systems for in-house and third-party tools."
>
> - **Frederik Ferner, Senior Computer Systems Administrator at Diamond Light Source**

Diamond Light Source uses Univa Grid Engine as job scheduler.

Read on for instructions on how to set up Ellexus Mistral with Univa Grid Engine.

Diamond Light Source

Diamond Light Source is the UK's national synchrotron or particle accelerator. Unlike particle accelerators such as those used in the labs at CERN to look at the result of colliding particles, the Diamond Light Source uses the radiation from speeding electrons to beam bright light into a range of experiments arranged around the edges of the accelerator. It works like a giant microscope, harnessing the power of electrons to produce bright light that scientists can use to study anything from fossils to jet engines to viruses and vaccines.

The machine accelerates electrons to near light speeds so that they give off light 10 billion times brighter than the sun. These bright beams are then directed into laboratories known as 'beamlines'. Here, scientists use the light to study a vast range of subject matter, from new medicines and treatments for disease to innovative engineering and cutting-edge technology.

Each beamline is used in an experiment that generates vast amounts of data, which must be stored in real time and processed quickly. Experiments cover a range of fields from industrial engineering to microbiology and cystalography, which means the HPC team needs to be able to handle a wide range of data rates and compute applications.

I/O profiling tools from Ellexus

Ellexus has a range of I/O profiling tools that provide visibility into how applications are accessing data. While Mistral offers live system telemetry and scalable I/O profiling, Breeze provides detailed dependency analysis and data hygiene capabilities. The tools provide always-on, system- and storage-agnostic APN monitoring and profiling that allows the user to scale rapidly or migrate to new platforms seamlessly.

Diamond Light Source has been using the tools from Ellexus to profile the wide range of applications run by the team and to check them for data access efficiency. In particular, they have been using Mistral.

Not all applications are written or maintained in house at Diamond Light Source. The team wanted to see if there were any obvious changes that could be made to the maintained tools to improve performance, or changes to the way that third-party applications are used.

Data collected by Mistral includes:

- Reads, writes and meta data
- Random vs streaming I/O
- Throughput and IOPS
- I/O sizes and performance
- Burst vs steaming profile

Read on to see how to integrate Mistral with Elastic Search and a Grafana dashboard for live per-job telemetry.

The results

Unnecessary metadata

Following a profile of various workloads using Mistral, the Diamond Light Source team noticed one particular application doing a lot more stat() calls than expected.

This eventually turned out to be a number of programs where static files were tested for in a loop even though they either existed at the start of the program or were created by it at the end. The same file was stat'ed many times over, even though there was no change to it throughout the execution. By amending this program, run time was reduced and the meta data load on the file system was improved.

Python can take a long time to load

The profile using Mistral also revealed a number of inefficient program startups. Unfortunate setups with long pythonpath chains resulted in lots of attempts to load python modules from different locations until they were found in the last configured path. The programs searched for the python modules in every location in the path, trawling the file system each time when the module was in the last location in the path.

By cutting down the pythonpath settings, the team managed to decrease startup times and increase the performance of the application significantly.

Catching noisy neighbours with live I/O telemetry

Several other findings revealed inefficiencies that slowed the system down, impeding the work of different users. For example, the team discovered an instance where one particular program in a specific invocation completely saturated the network of a number of compute nodes. The program used only a single core for execution and shared node with other jobs on the node, so just by looking at the node it wasn't obvious which program it was.

The problem was quickly resolved by looking at the live telemetry from Mistral because Mistral logs not only the hostname, but also the job ID of any noisy neighbours. It's easy to see how the application is overloading the network and preventing other compute nodes on that network from operating.

Conclusion

The Diamond Light Source team has implemented many of the improvements uncovered by Mistral. By reducing runtime and improving the overall architecture, productivity has been increased.

info@ellexus.com
+44 1223 421646
www.ellexus.com

Integrating Mistral with Univa Grid Engine

To run Mistral in production or on a large number of jobs it's easiest to use the job scheduler such as Univa Grid Engine or launch configuration to automatically set up the Mistral environment. For each queue that is required to automatically wrap jobs with Mistral, add a `starter_method` setting that points to the mistral launcher script.

```
$ qconf -mq mistral.q
```

The following snippet of queue configuration shows the appropriate setting to use the file described above:

```
epilog                NONE
shell_start_mode      unix_behavior
                            starter_method      /home/ellexus/ugedemo/launch.sh
suspend_method        NONE
resume_method         NONE
```

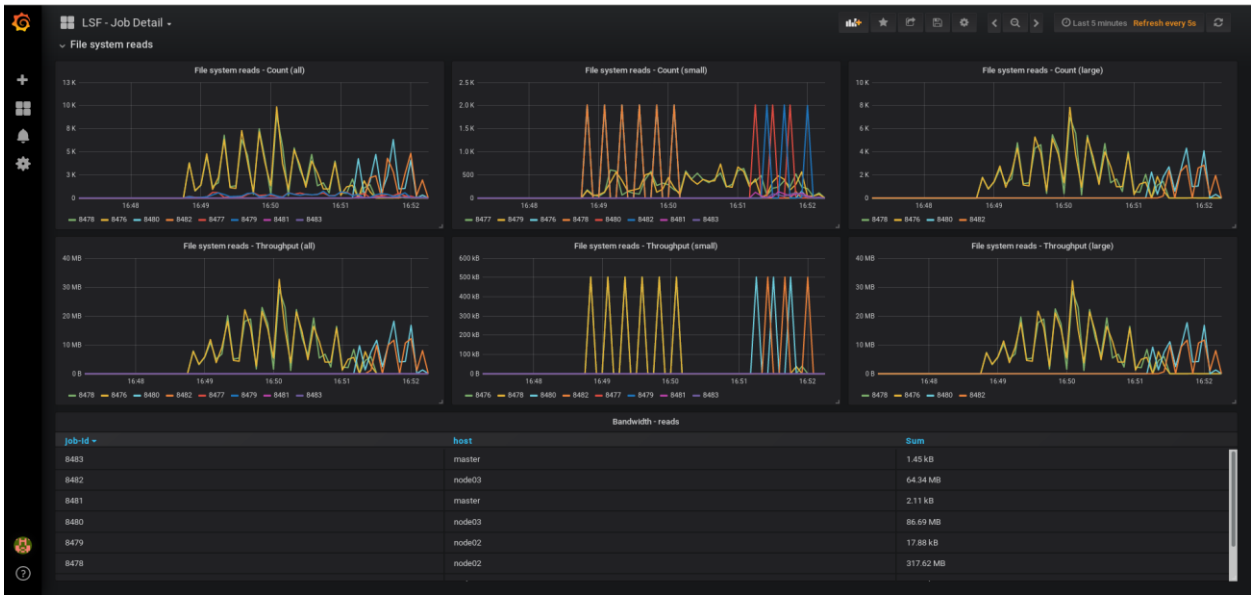
The launcher script will invoke Mistral on every job on that queue. Mistral is invoked by setting environment variables in the launch script. These variables enable Mistral, configure the licence and set the monitoring configuration and settings that control where to send the data.

How to send the data to a database

Mistral logs data in a scalable database which can be sent to any graphing dashboard to view the information such as Grafana or Splunk. Mistral is shipped with plugins for Elasticsearch, InfluxDB, MySQL and Splunk. These give live updates so you can find rogue jobs quickly or look back over historical data. Both Splunk and Grafana let you filter on job, host, project or user, which makes it easy to do chargeback or to find system bottlenecks.

The Diamond Light Source team pushes the Mistral data to Elasticsearch and uses Grafana as a live dashboard. Both are free tools and Mistral comes with templated dashboards for Grafana. The following shows the database settings for an Elasticsearch plugin that pushed data every 5s:

```
PLUGIN,OUTPUT
PLUGIN_PATH,${MISTRAL_INSTALL}/ mistral_elasticsearch
INTERVAL,5
PLUGIN_OPTION, index=mistral
PLUGIN_OPTION, host=10.0.0.100
PLUGIN_OPTION, port=9200
PLUGIN_OPTION, username= mistralelastic
PLUGIN_OPTION, password= mistral
PLUGIN_OPTION, error=${HOME}/mistral_elastic.log
END
```



Mistral data in Grafana

Data collected by Mistral

It's easy to add in custom metrics to measure machine health for a full system overview.

The Mistral I/O contract is a flexible way to control what Mistral measures. To profile an application in detail you can set all available rules to trigger as soon as one I/O operation is performed. To monitor a whole cluster, it is better to set fewer rules with higher thresholds so that only the data you need is collected. For the full list of what can be measured please refer to the Mistral user manual.

```
#data is collected every 5s
2,monitortimeframe,5s

#log the number of open calls that exceed 500 to /homehome-count-
open,/home,open,all,count,500

#log the mean latency of read calls from /usrthat are less than 4KB
#if the mean latency exceeds 50us
usr-lat-read_small,/usr,read,-4kB,mean-latency,50us
```