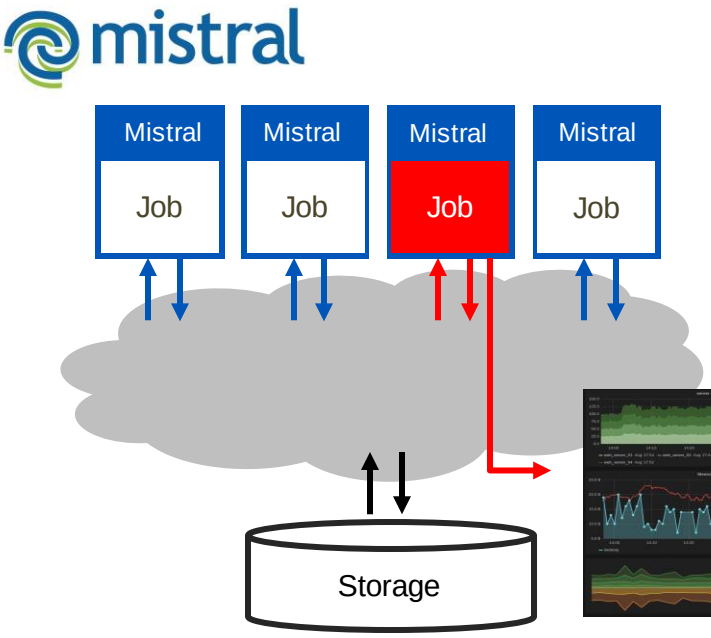
Mistral
Protection against bad I/O patterns

As distributed systems and compute clusters become more complex, the need for monitoring becomes more important. It is no longer enough to monitor compute and memory performance; for good system performance you need to care about I/O performance as well.

Always-on monitoring in production

Mistral is the leading I/O profiling tool in the new family of always-on monitors of Linux system. By allowing you to control the amount of information gathered, you can ensure that you know about the important events without being overwhelmed with too much data. It is storage agnostic and monitors I/O bandwidth, latency and meta data as well as system stats such as CPU and memory utilisation.



Mistral can be used with Elasticsearch and Grafana, or other frameworks such as Splunk, to give a system overview with per user, per host or per project overviews updated in real time.

Once a problem has been identified, the specific user, job, host, program and file can be extracted from the Mistral log data and the problem quickly solved.

Zoom in on application-specific data:

```
Job 42
user janesmith
/bin/perl start.pl
exceeded 500MB/s to
/share/scratch
```

Identify rogue jobs in production and test
Mistral is lightweight enough to run in production or on large test suites to find bad I/O before it becomes a problem.

Protect the storage by throttling job I/O
Mistral can actively protect shared storage by throttling the I/O of individual jobs to maintain a good quality of service.

Eliminate bad I/O patterns
Mistral can detect bad meta data accesses, random I/O and small reads and writes as well as simpler problems such as writing too much data. It gives users the data they need to fix the problem.

Solving the noisy neighbour problem with Arm

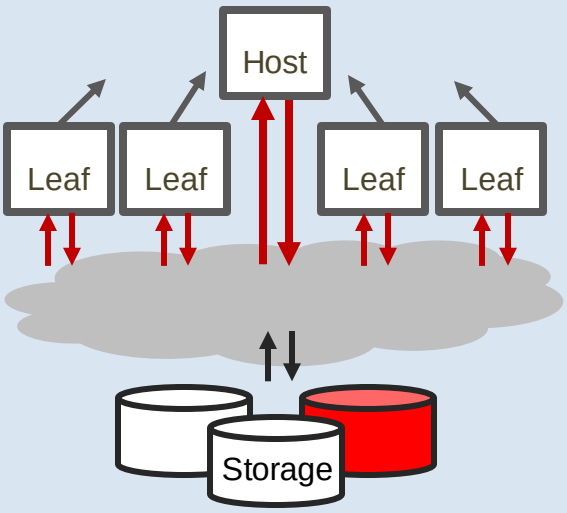
We developed Mistral with Arm when they were experiencing problems with users overloading shared storage with bad I/O patterns. Mistral will monitor application I/O and cluster performance so that I/O hungry jobs can be identified quickly.

Olly Stephens, Engineering Systems Architect, describes the project with Ellexus:
"One of our key long-term objectives is to develop a better understanding of the storage requirements of each compute job."

Mistral case study

Catching rogue jobs before they overload the storage

One Ellexus customer is a software vendor in the high-performance computing industry and has its own cluster for continuous integration testing. The team has deployed Mistral across their site. They wanted to protect their storage in-house, but wanted to make sure that bad I/O patterns are not seen by their customers either.



Customer set up

The customer has a host leaf architecture with about 1,000 compute nodes in total and 8 shared filers.

Problem 1: Data going to the wrong place

Users are supposed to aggregate data in the host node and write to the storage from there, but this doesn't always happen. Distributed writes from leaf nodes can overload the storage.

Problem 2: Too much data being written

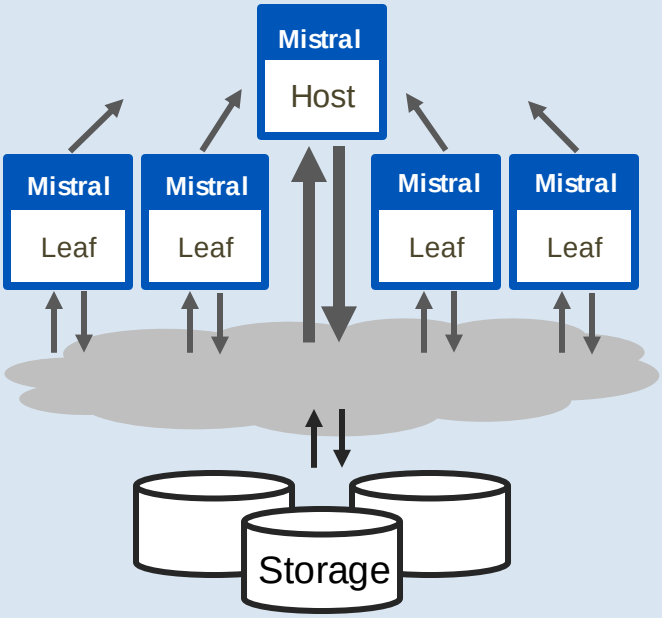
Users sometimes leave the debug flag on when they hand over a job to be run at scale. This generates a lot of extra data from host and leaf nodes.

Mistral deployment goals

- To provide IT administrators with a way to profile applications prior to running them at scale.

- To integrate with the existing fault management framework and provide at-a-glance monitoring exceptions delivered by Mistral.

- To provide users with the storage profile of their applications and enough information to find a solution to the performance problems.



Solution: Always-on deployment for production jobs with an extra healthcheck queue for R&D

Mistral is deployed to all production jobs using IBM LSF and the I/O profiling data is displayed in the RTM dashboard. This allows IT managers to identify problem jobs live as well as giving a historic view for when problems have passed. The data is then passed back to users so they can ensure that the application workflow is fixed.

Applications that run as part of the R&D workflow are periodically checked with a more detailed I/O healthcheck queue that has more alerts and can give a more detailed breakdown of good and bad I/O.

The customer has seen an overall reduction in the time taken to resolve a problem as well as the number of rogue jobs submitted. Now that users have concrete information about what causes a problem they are more proactive about preventing bad I/O.

Mistral installation

Technical overview

Mistral logs data in a scalable database so a graphing dashboard such as Grafana or Splunk can be used to view the information. These give you 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. This makes it easy to do chargeback or to find system bottlenecks.

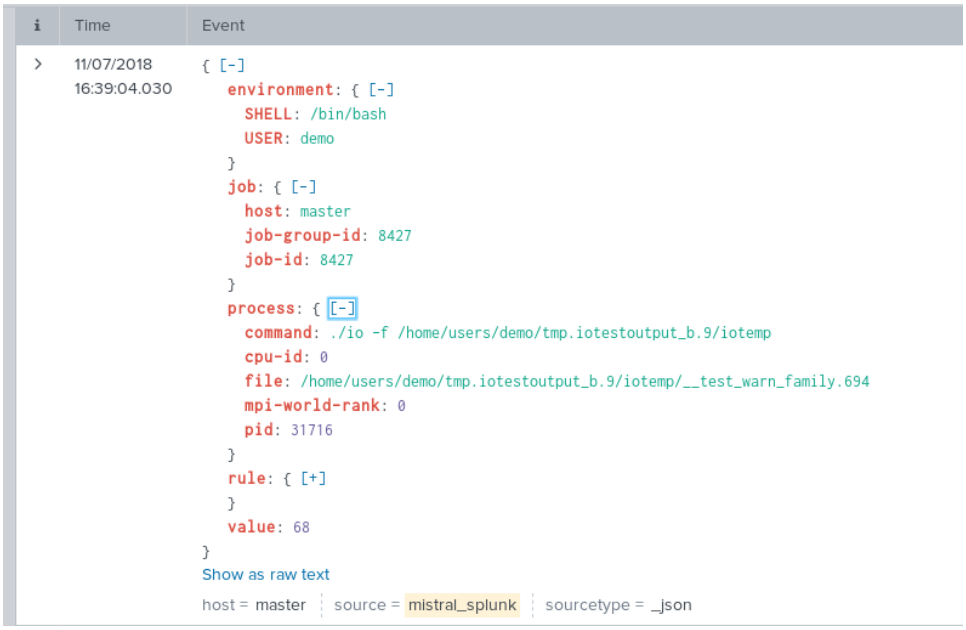
You can also set custom alerts that tell you when a measure has deviated too far from the mean so you can quickly spot a problem.

Mistral shows:

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

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

Although the graphical overview gives you a good summary of the system, to triage a bottleneck once it is found it is better to look at the additional information available in the database. Mistral logs the process that caused the I/O to spike and the last file it accessed giving you *what happened* as well as the *why it happened*.



Mistral is easy to configure using the template setting that comes with the tool.

Intregration with the job scheduler
Mistral can be launched on a single application when you start it, but to run Mistral in production or on a large number of jobs it's easiest to use the job scheduler or launch configuration to automatically set up the Mistral environment.

```
JOB_STARTER= /share/tools/ellexus/mistral_2.12.0/launch.sh; %USRCMD
```

What goes in the launch script?
The launch script sets up the licence, the database configuration and the Mistral I/O contract.

```
# The contract controls what Mistral will measure
export MISTRAL_CONTRACT_MONITOR_GLOBAL=${MISTRAL_INSTALL}/global.contract

# Tell Mistral to send output to elastic search via an output plug-in.
export MISTRAL_PLUGIN_CONFIG=${MISTRAL_INSTALL}/elastic_plugin.conf

source ${MISTRAL_INSTALL}/mistral
```

Data can be sent to any database using a configurable plugin. Mistral is shipped with plugins for Elasticsearch, InfluxDB, MySQL and Splunk. 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
```

How does the Mistral contract control what is measured?
The Mistral I/O contract is a flexible way of controlling 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 user manual.

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

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

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