

Real-Time I/O-Monitoring of HPC Applications

Eugen Betke, Julian Kunkel

Research Group
German Climate Computing Center
2017-11-15



Introduction

Why monitoring?

Monitoring is important to **find inefficient applications**

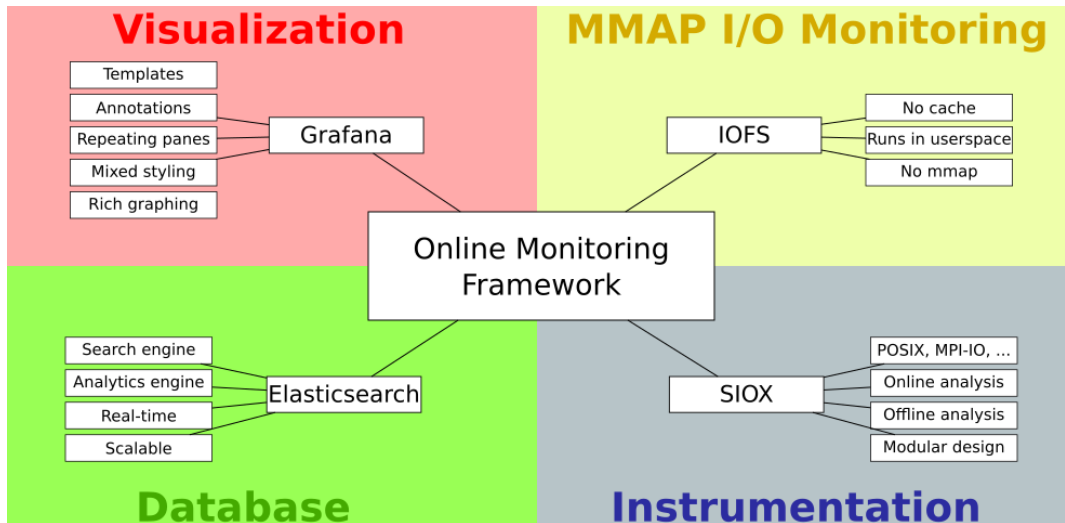
What I/O levels to monitor?

- **node I/O**
 - Overview of total I/O traffic on a node
 - Available in user space
- **file I/O**
 - Filtered I/O traffic for a specific file
 - Available in user space
- **mmap I/O**
 - I/O traffic done by virtual memory in the background
 - Hidden in the kernel space

How do monitoring tools get data?

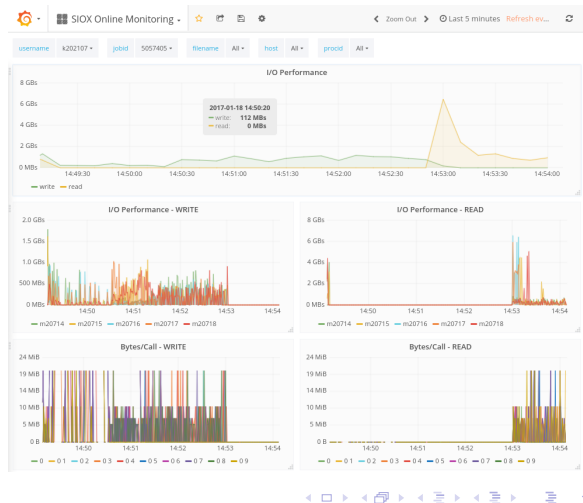
- Capturing of proc-files statistics
- Instrumentation code injection
 - Static approach
 - Injection of new compiled C code into a binary executable or dynamic library file
 - Re-compilation necessary
- **Interception with LD_PRELOAD**
 - Dynamic approach
 - Works with dynamic libraries only
 - Static linked functions not trackable

On-line Monitoring Framework [1] Components

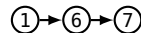
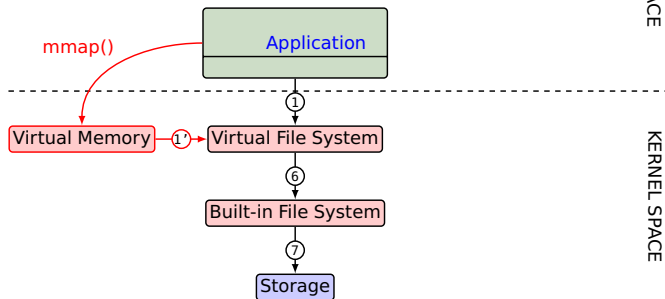


Web-Interface

- Online monitoring
 - Visualization while application runs
 - Delay in order of 1sec
- Interactive web interface
 - Zoom, time shift, filtering, ...
- Elaborated filtering
 - Based on templates
 - Auto update of templates



Existing I/O paths

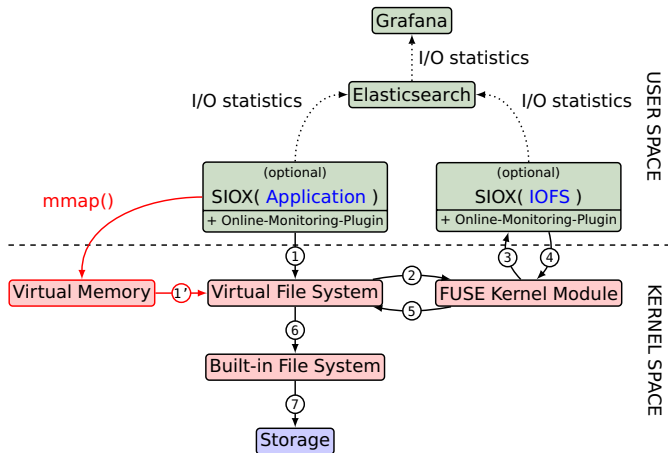


Existing file I/O path. Typically, supported by most instrumentation tools.



Existing mmap I/O path. Tools cannot trace this.

On-line Monitoring Architecture



Key features

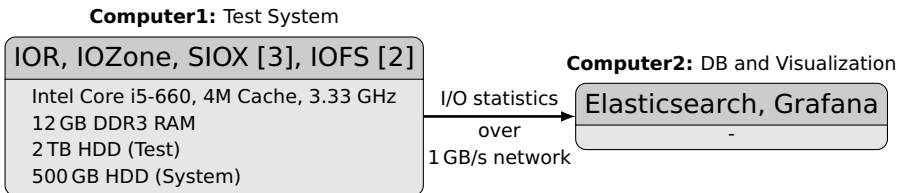
- 1 Traditional monitoring of file I/O
① → ⑥ → ⑦

File I/O calls are intercepted directly in application by SIOX

- 2 Monitoring of mmap I/O (Novelty)
①' → ② → ③ → ④ → ⑤ → ⑥ → ⑦

Redirected mmap I/O path from kernel to user space allows SIOX to intercept the I/O calls within elevated privileges.

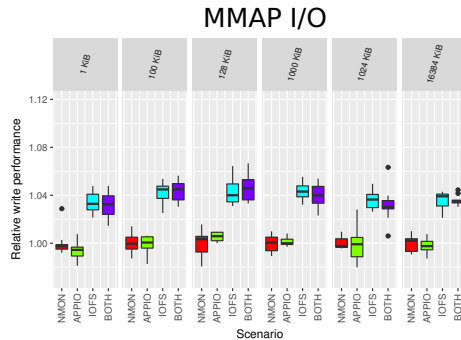
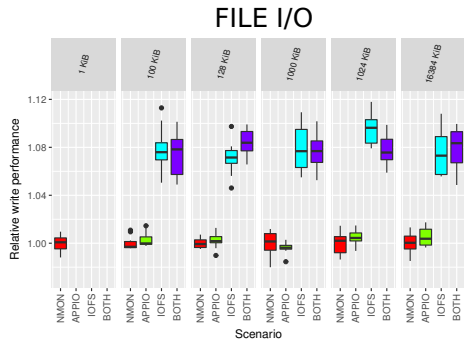
Test Setup



Experiment configuration

- 4 GiB test file
- 1 nodes and 1 processes per node
- Block sizes 1 KiB, 100 KiB, 128 KiB, 1000 KiB, 1024 KiB, 16384 KiB
- 10 test runs for each block size
 - IOR for file I/O
 - IOZone for mmap I/O
- Scenarios without monitoring and with monitoring (application, mount point, both)

Overhead [1/4] - Write



$$P_{rel} = \frac{\text{mean}(P_{no_monitoring})}{P_{\langle scenario \rangle}}$$

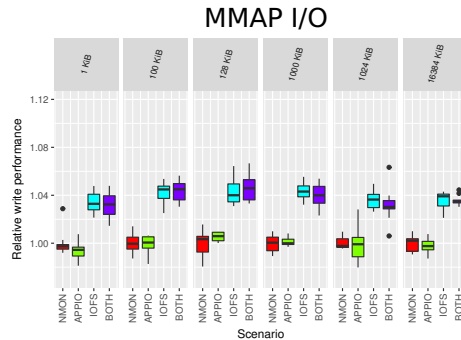
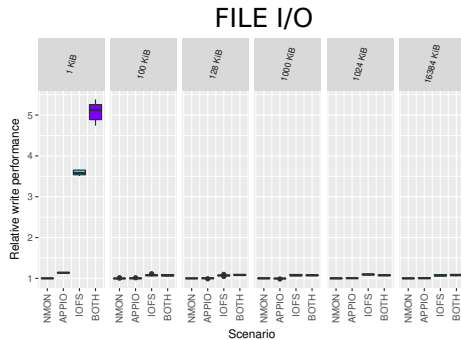
Scenarios

NMON	no monitoring
APPIO	monitoring of application
IOFS	monitoring of mount point
BOTH	APPIO and IOFS

Exp. configuration

nodes/processes per node	1/1
test file	4 GiB
test runs	10

Overhead [2/4] - Write (zoomed)



$$P_{rel} = \frac{\text{mean}(P_{no_monitoring})}{P_{\langle scenario \rangle}}$$

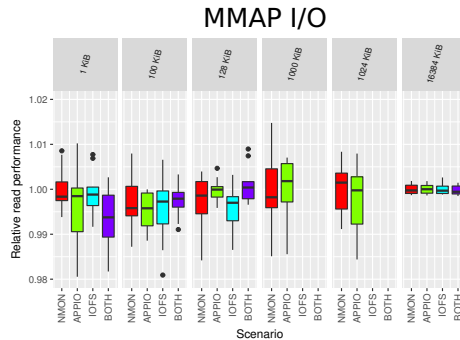
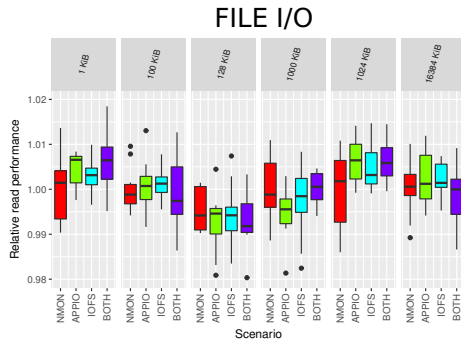
Scenarios

NMON	no monitoring
APPIO	monitoring of application
IOFS	monitoring of mount point
BOTH	APPIO and IOFS

Exp. configuration

nodes/processes per node	1/1
test file	4 GiB
test runs	10

Overhead [3/4] - Read



$$P_{rel} = \frac{\text{mean}(P_{no_monitoring})}{P_{\langle scenario \rangle}}$$

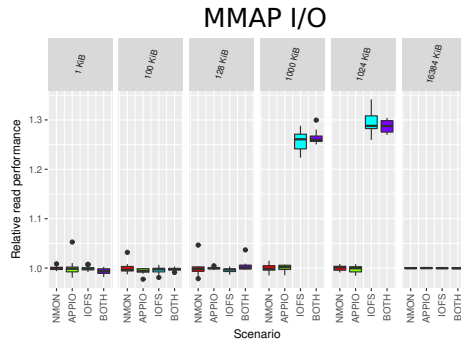
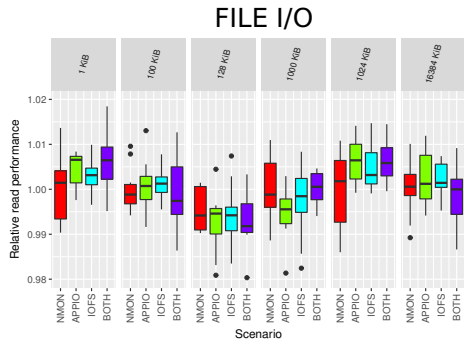
Scenarios

NMON	no monitoring
APPIO	monitoring of application
IOFS	monitoring of mount point
BOTH	APPIO and IOFS

Exp. configuration

nodes/processes per node	1/1
test file	4 GiB
test runs	10

Overhead [4/4] - Read (zoomed)



$$P_{rel} = \frac{\text{mean}(P_{no_monitoring})}{P_{\langle scenario \rangle}}$$

Scenarios

NMON	no monitoring
APPIO	monitoring of application
IOFS	monitoring of mount point
BOTH	APPIO and IOFS




Exp. configuration

nodes/processes per node	1/1
test file	4 GiB
test runs	10

Summary

- **Non-intrusive** On-line Monitoring Framework
 - is built on top of **open source** software: FUSE, SIOX, Elasticsearch, Grafana
 - provides near **real-time** monitoring
 - supports **file I/O** and **mmap I/O**
- Overhead
 - is **mostly small**
 - **can be significant with FUSE** when
 - writing small block
 - reading suboptimal block sizes

References

-  Eugen Betke and Julian Kunkel. “Real-Time I/O-Monitoring of HPC Applications with SIOX, Elasticsearch, Grafana and FUSE”. In: Springer International Publishing, 2017.
-  **IOFS**. <https://github.com/joobog/iofs>. Accessed: 2017-11-12.
-  **SIOX**. <https://github.com/JulianKunkel/siox>. Accessed: 2017-03-22.