Online Monitoring of I/O

Eugen Betke and Julian Kunkel

Research Group
German Climate Computing Center

23-03-2017

Table Of Content

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 can not be manipulated

Virtual Memory

mmap()

mmap() is a system call to map the contents of a file into memory.

```
1 void *mmap(void *addr, size_t len, int prot, int flags, int fildes, off_t off);
```

`fildes`	file descriptor
`off`	offset within the file to be mapped
`len`	length of data from the offset to be mapped

Problematic

Virtual memory run in kernel space. Non-priviliged applications have no access.

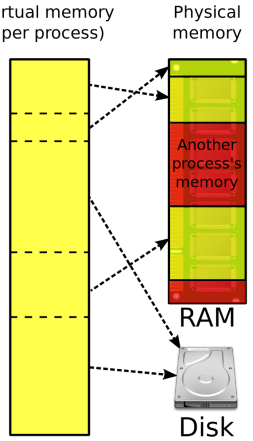


Figure: Virtual Memory[4]

Table Of Content

Table Of Content

FUSE File System - Overview

- File System in User Space
- Software interface for Unix-like computer operating systems
- Non-privileged file systems run without editing kernel code
- File system code runs in user space
- FUSE module provides only a "bridge" to the actual kernel interfaces

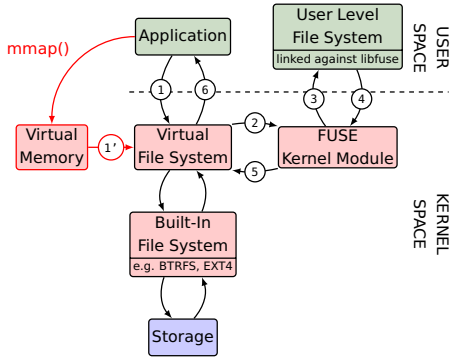


Figure: FUSE architecture

FUSE File System - IOFS

- Auxiliary tool
  - Mounts a directory to a mount point
- Features
  - No cache
  - No mmap() operations
  - No root priviledges required

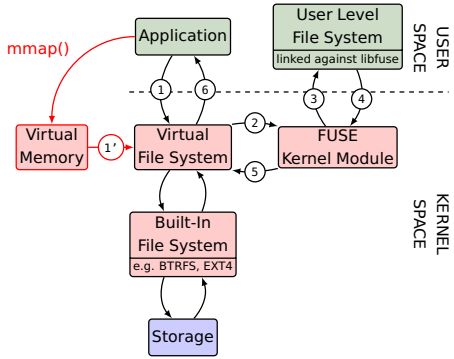


Figure: FUSE architecture

SIOX - Scalable I/O for Extreme Performance [3]

- Performance Analysis Framework
- Open-Source-Framework published under LGPL [1]
- Supports POSIX-, MPI-, HDF5- and NETCDF4-Layers
- Modular design
- Online Analysis
  - Analyse activities during program execution
- Offline Analysis
  - Analyse activities after program termination

[1] https://github.com/JulianKunkel/siox.git

SIOX - On-line Monitoring Plug-in

Plug-in

- Aggregates I/O traffic to statistics
- Uses I/O categories
  - write: pwrite(), write(), . . .
  - read: get(), read(), . . .
- Sends I/O statistics in specified time intervals

I/O Statistics

- Typed for visualization
- Types
  - metrics - y-axis
  - timestamp - x-axis
  - tags - filtering

I/O Statistics

Name	Type	Value
write_duration	metric (basic)	time spent for writing
write_bytes	metric (basic)	bytes written
write_calls	metric (basic)	number of I/O operations
write_bytes_per_call	metric (derived)	write_bytes, write_calls
read_duration	metric (basic)	time spent for reading
read_bytes	metric (basic)	bytes read
read_calls	metric (basic)	number of I/O operations
read_bytes_per_call	metric (derived)	read_bytes, read_calls
filename	tag	I/O operations
access	tag	I/O operations
username	tag	SLURM_USER
hostname	tag	HOSTNAME
procid	tag	SLURM_PROCID
jobid	tag	SLURM_JOBID
layer	tag	user defined
timestamp	date	system clock

Elasticsearch

- Scalable, real-time search and analytics engine
- Apache 2 license
- Indexing of all field allow fast look-ups
- Highly scalable
  - Runs on laptops as well as on large-scaled super computers

Grafana

- Rich graphing
  - Interactive, editable graphs
  - Multiple Y-axes, Logarithmic scales and options
- Mixed Styling
  - Mix lines, points and bars
  - Mix stacked w/ isolated series
- Template Variables
  - Variables are automatically filled with values from DB
- Repeating Panels
  - Automatically repeat rows or panels for each selected variable value
- Annotations
  - Show events from datasources in the graphs

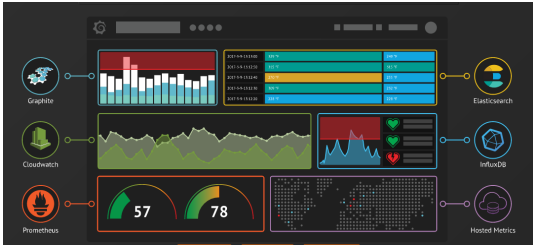


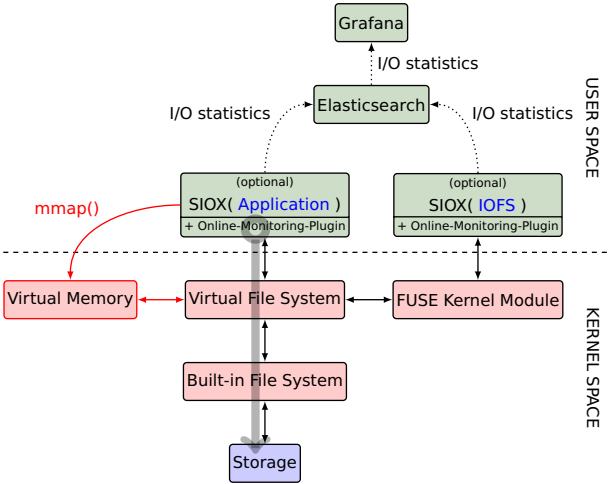
Figure: Grafana [1]

Table Of Content

On-line Monitoring Framework

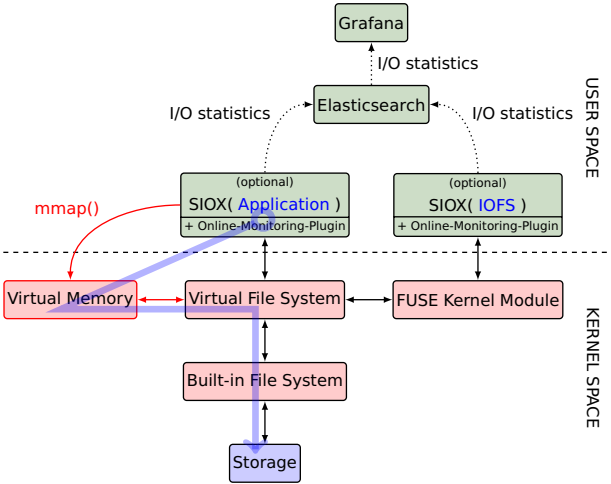
- High scalability
- Almost real-time on-line monitoring
- Non-intrusive framework
  - No changes need to be done in applications
- Components
  - Interception of mmap I/O: FUSE
  - I/O statistics: SIOX + OnlineMonitoringPlugin
  - DB back-end: Elasticsearch
  - Visualization: Grafana

On-line Monitoring Architecture 1/4



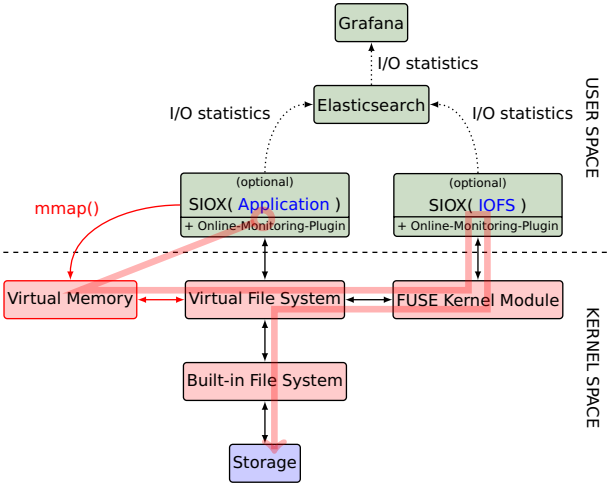
- file I/O

On-line Monitoring Architecture 2/4



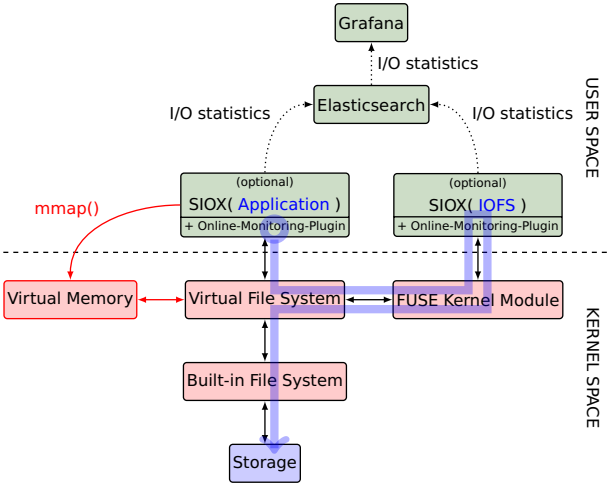
- mmap I/O
- here still hidden

On-line Monitoring Architecture 3/4



- mmap I/O

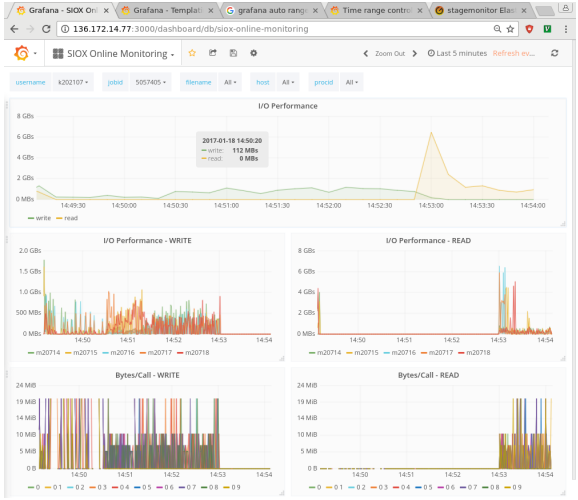
On-line Monitoring Architecture 4/4



- file I/O

Grafana Web-Interface (User Perspective)

- Interactive web interface
  - Zoom, time shift, filtering, . . .
- Elaborated filtering
  - Based on templates
  - Auto update of templates
- Flaws
  - No Auto range finder
  - Template update functionality not user friedly



On-line monitoring live demo

Table Of Content

Table Of Content

Elasticsearch performance

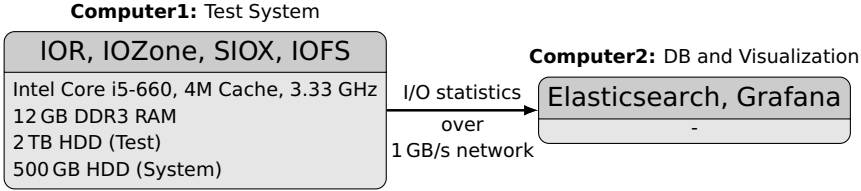
- Elasticsearch was deployed on an office PC
- Test setup
  - Nodes: 10
  - Processes per Node: 20
  - Metrics were
    - generated on our HPC “Mistral” [2] with a python script
    - sent in 100 metrics packages
- Result
  - 100 x 7500 metrics per second

Package

```
{
  'metric1': '1',
  'metric2': '2',
  'metric3': '3',
  ...
  'metric100': '100'
}
```

Table Of Content

Overhead - Test Setup

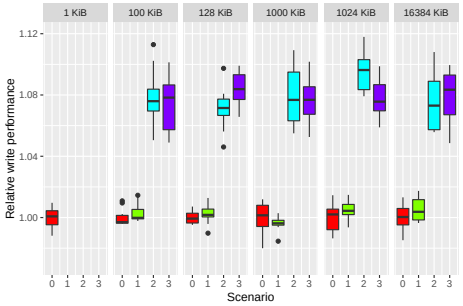


Experiment configuration

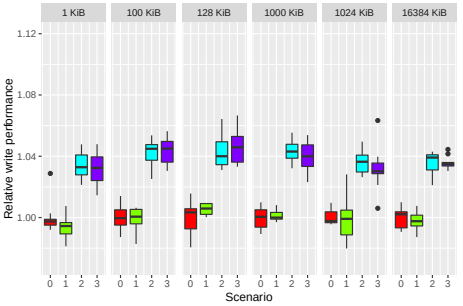
- Block sizes 1 KiB, 100 KiB, 128 KiB, 1000 KiB, 1024 KiB, 16384 KiB
- 1 nodes and 1 processes per node (in SLURM)
- 4 GiB test file
- 10 test runs for each block size
  - IOR for file I/O
  - IOZone for mmap I/O
- Scenarios without monitoring and with monitoring (application, mount point, both)

Overhead [1/4] - Write

FILE I/O



MMAP I/O



$$P_{rel} = \frac{mean(P_{no\_monitoring})}{P_{<scenario>}}$$

Scenarios

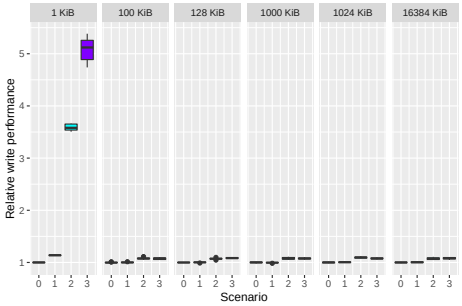
0	no monitoring
1	monitoring of application
2	monitoring of mount point
3	both, (1) and (2)

Exp. configuration

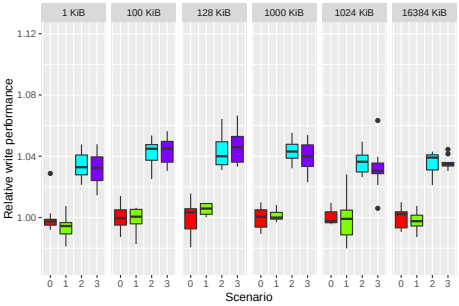
nodes/processes per node	1/1
test file	4 GiB
test runs	10

Overhead [2/4] - Write (zoomed)

FILE I/O



MMAP I/O



$$P_{rel} = \frac{mean(P_{no\_monitoring})}{P_{<scenario>}}$$

Scenarios

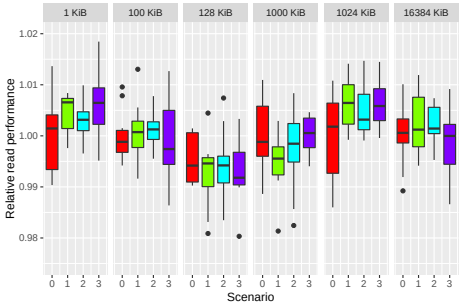
0	no monitoring
1	monitoring of application
2	monitoring of mount point
3	both, (1) and (2)

Exp. configuration

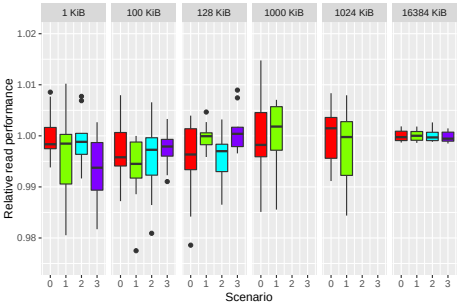
nodes/processes per node	1/1
test file	4 GiB
test runs	10

Overhead [3/4] - Read

FILE I/O



MMAP I/O



$$P_{rel} = \frac{mean(P_{no\_monitoring})}{P_{<scenario>}}$$

Scenarios

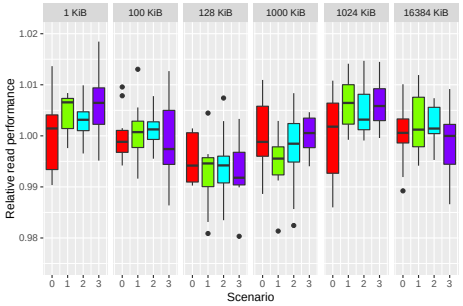
0	no monitoring
1	monitoring of application
2	monitoring of mount point
3	both, (1) and (2)

Exp. configuration

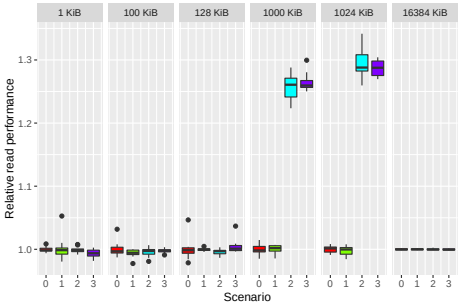
nodes/processes per node	1/1
test file	4 GiB
test runs	10

Overhead [4/4] - Read (zoomed)

FILE I/O



MMAP I/O



$$P_{rel} = \frac{mean(P_{no\_monitoring})}{P_{<scenario>}}$$

Scenarios

0	no monitoring
1	monitoring of application
2	monitoring of mount point
3	both, (1) and (2)

Exp. configuration

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

Table Of Content

Summary

- Non-intrusive On-line Monitoring Framework
  - Built on top of open source software: FUSE, SIOX, Elasticsearch, Grafana
  - Provides near real-time on-online monitoring
  - Collects I/O statistics from applications and mount points
  - Provides support for file I/O and mmap I/O
    - file I/O: Detailed information about file accesses
    - mmap I/O: Non-intrusive way for instrumenting virtual memory (novelity)
- Scalability (office PC)
  - 100 x 7500 metrics/second
- Overhead (office PC)
  - Write: file I/O < 1%/12% (+outlier) and Read mmap I/O < 1%/6%
  - Read: file I/O < 1%/1% and Read mmap I/O < 1%/30% (+outlier)
- Results for our HPC “Mistral” [2] are coming soon

References

- **Grafana**. https://grafana.com/. Accessed: 2017-03-22.
- **HLRE-3 "Mistral"**. https://www.dkrz.de/Klimarechner/hpc. Accessed: 2017-03-22.
- **SIOX**. https://wr.informatik.uni-hamburg.de/research/projects/siox. Accessed: 2017-03-22.
- **Virtual Memory**. https://en.wikipedia.org/wiki/Virtual_memory. Accessed: 2017-03-22.