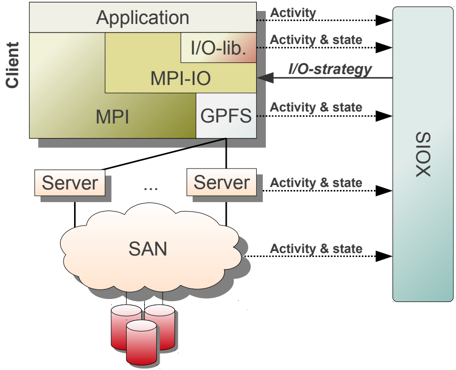
Towards Intelligent Self-Optimisation in HPC I/O

Julian Kunkel   Michaela Zimmer   Marc Wiedemann

University of Hamburg

June 20, ISC '13

Project Goals



SIOX will

- collect and analyse
  - activity patterns and
  - performance metrics

in order to

- assess system performance
- locate and diagnose problem
- learn optimizations

Partners and Funding

Bundesministerium
für Bildung
und Forschung

- Funded by the BMBF
  Grant No.: 01 IH 11008 B
- Start: Juli 1st, 2011
- Duration: 36 Months

DKRZ
DEUTSCHES KLIMARECHENZENTRUM

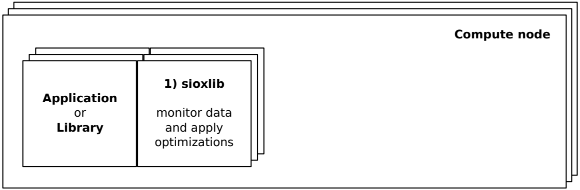
HLRS

IBM

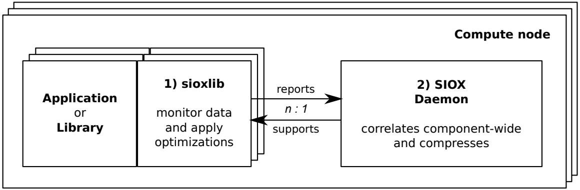
Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

ZIH
Zentrum für Informationsdienste und Hochleistungsrechnen

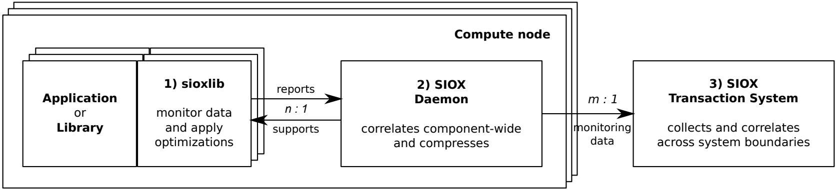
Architecture



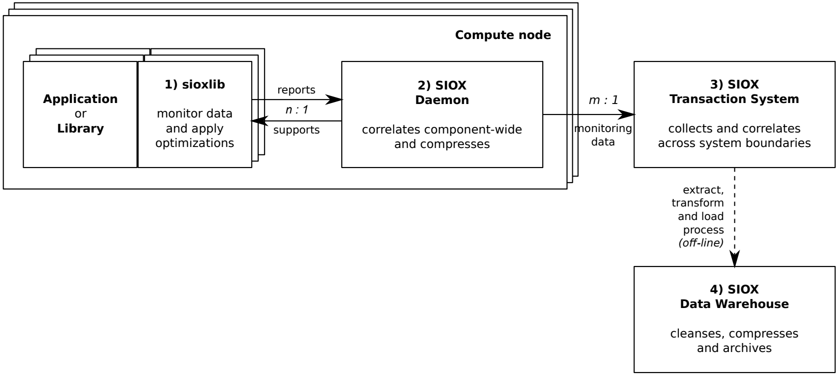
Architecture



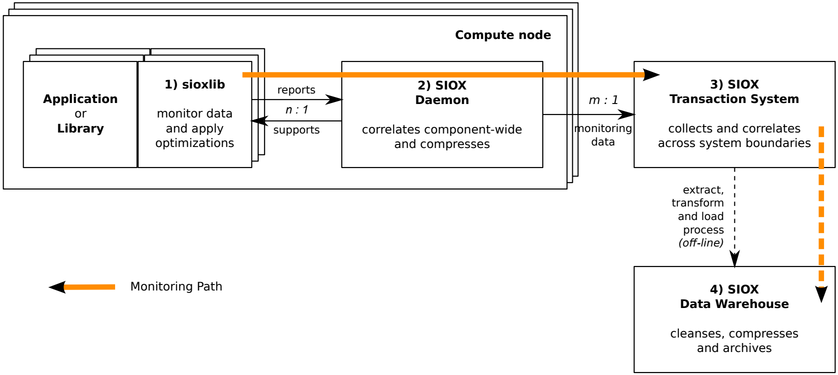
Architecture



Architecture

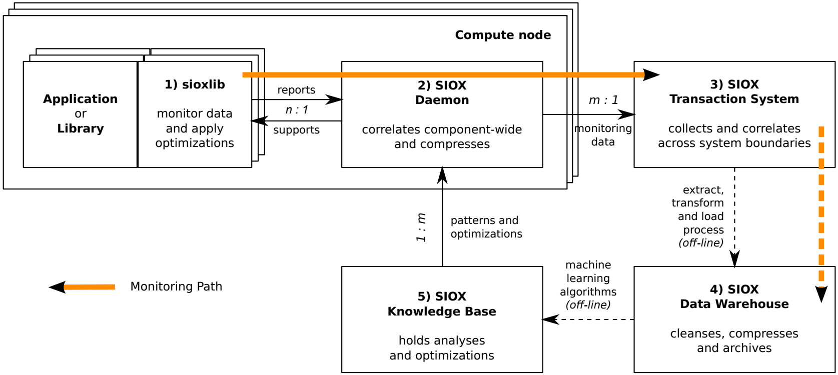


Architecture



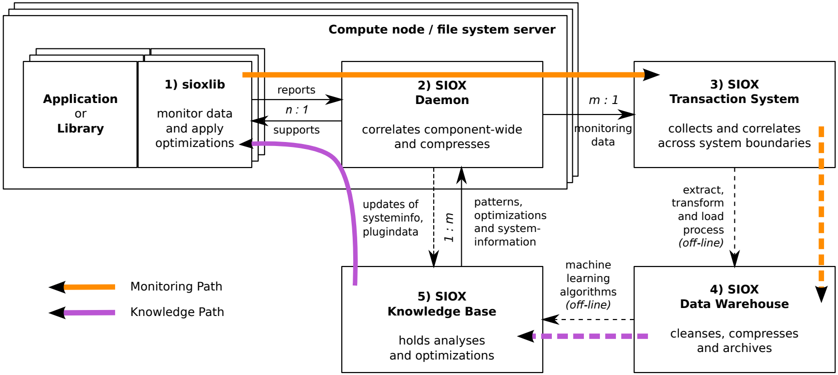
- Data gathered is stored via the *monitoring path*.

Architecture



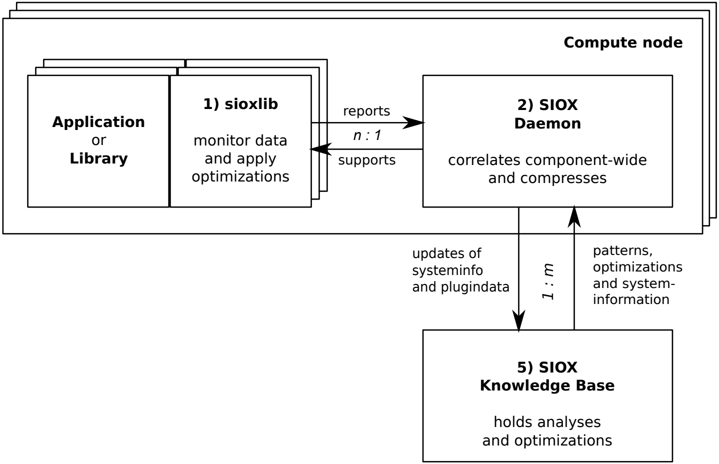
- Data gathered is stored via the *monitoring path*.

Architecture



- Data gathered is stored via the *monitoring path*.
- Components receive the knowledge gleaned via the *knowledge path*.

Alternative Architecture Configuration: Online-Mode



Configuration is loaded upon startup and initializes modules

Overview of Concepts and Mechanisms

- User-level monitoring API
  - "Wrapper" to ease instrumentation of software layers
- Relation of activities
  - Implicit linking of process-internal activities
  - Explicit linking between remote activities
  - Link is created while transerring data to the warehouse
- Observed activies and statistics are processed by multiple plugins
  - Synchronous and/or asynchronous
  - Activities can be handled statefull (within a process) or stateless
  - May use (static) system information/knowledge
  - Usage: Learning of optimizations, intelligent logging, own overhead
- System knowledge
  - One database entry per node, file system, storage device
  - Plugins may create their own node/fs/device specific entries
  - Detect hardware changes (upon startup)
- Local and global "reasoning" to assess system state

Semi-Automatic Instrumentation of Software-Layers

Workflow

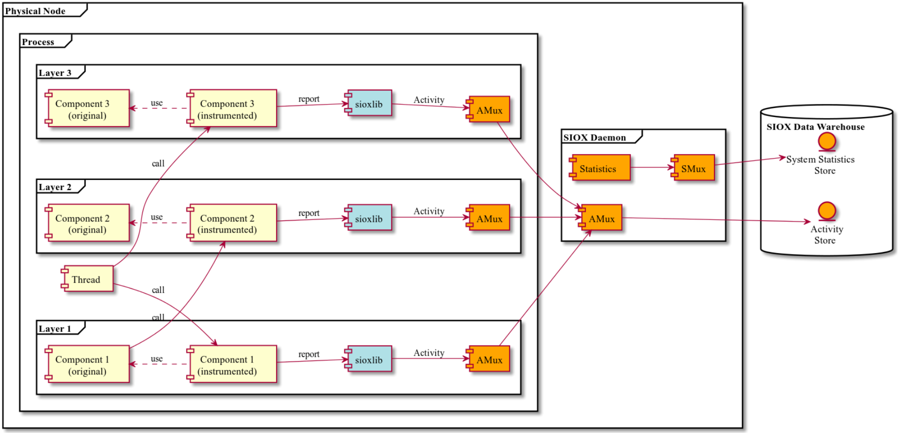
1. Saving relevant function prototypes in a header file
2. Annotate functions in the header
3. Tool parses header and creates either
   - a shared library for LD_PRELOAD
   - a library to use with `ld -wrap`

Instrumentation can be done incrementally

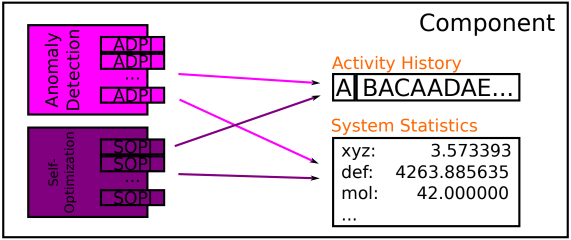
Example Header for POSIX

```
//@component "POSIX"

//@register_descriptor fileName "File Name" SIOX_STORAGE_STRING
/////// END GLOBAL SECTION ///////////////

//@activity
//@activity_attribute fileName pathname
//@horizontal_map_put_int ret
//@error ''ret < 0'' errno
int open(const char *pathname, int flags, ...);

//@activity
//@activity_attribute bytesToWrite count
//@activity_link_int fd
//@error ''ret < 0'' errno
ssize_t write(int fd, const void *buf, size_t count);
```

Logical View of the Monitoring Path



Intelligent Components



Each component/layer holds:

- Plug-ins to detect exceptional behaviour
- Plug-ins to suggest possible optimizations

Additionally, a daemon holds:

- Recent system statistics, updated regularly
- Statistics plug-ins
- A plugin to control SIOX behavior
- A rule-based reasoner classifies system-state and bottlenecks

Building SIOX's Brain

To harness the data gathered, SIOX uses *Knowledge Packages*.

A Knowledge Package. . .

contains of

- a Machine Learning Plug-In

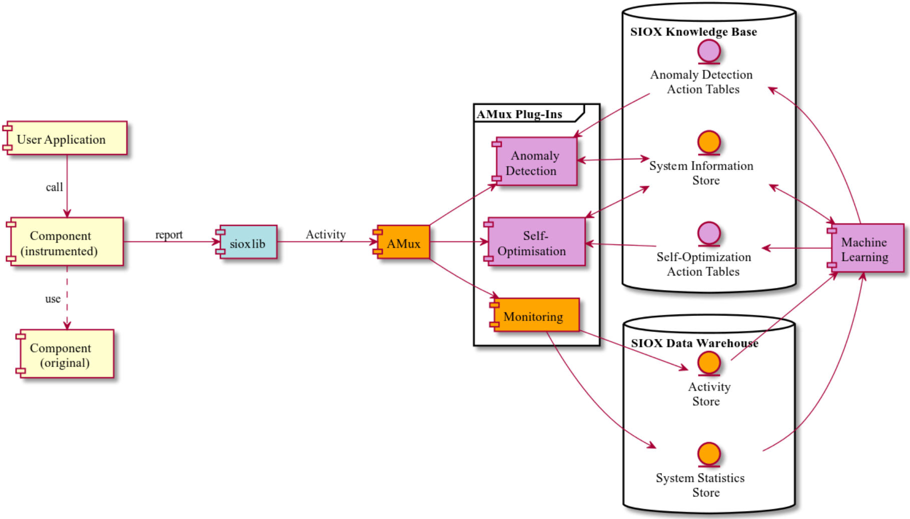
and corresponding plugins

- Anomaly Detection Plug-In
- Self-Optimization Plug-In

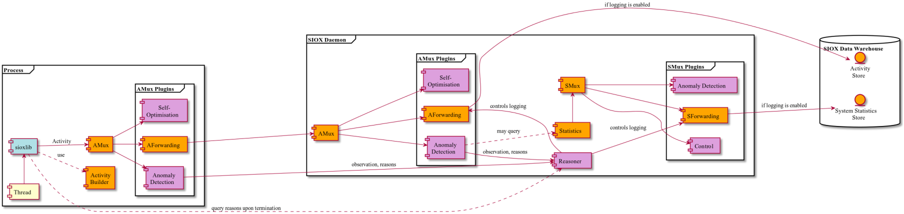
Knowledge Package may use private *Action Tables* in the Knowledge Base.

The MLPI will create (and possibly update) the action table, which may also be done manually.

Interplay Between Monitoring and Knowledge Path (1)



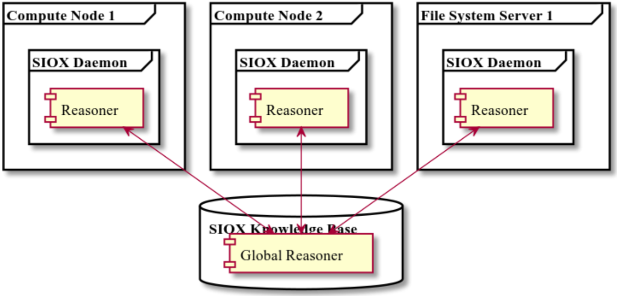
Interplay Between Monitoring and Knowledge Path (2)



Click to access the fullpage PDF of the design

Reasoning

- Node-local reasoner decides when and how long to log
- System-state, detected bottlenecks and reasons are communicated
  - E.g. “Server overloaded”, “Bad I/O pattern“
  - All knowledge to global reasoner
  - Overview is communicated to all daemons
- Global reasoner maintains statistics for later investigation



Anomaly-Detection Plugin Example 1

A simple rule-based and stateless plugin detecting exceptional performance

Mathematical model and Action Table

$$f_{\text{Utilization}}(\text{Component}, \text{Activity}) = \frac{\text{Time}(\text{Activity})}{t_{\text{expected}}(\text{Component}, \text{Activity})}$$

$$t_{\text{expected}} = \frac{\text{Size}(\text{Activity})}{\text{SequentialTransferRate}(\text{Component})} + \text{latency}(\text{Component})$$

Result	Action
$f_{\text{Utilization}} < 0.10$	Report("Exceptionally low")
$0.10 < f_{\text{Utilization}} < 0.95$	No Action
$0.95 < f_{\text{Utilization}}$	Report("Exceptionally high")

Component can be a subset of {current software layer, compute node, file system}

Self-Optimization-Plugin Example 1

A simple Action Table: Adjusting a system parameter

Action table for an SOPI write-behind plug-in

Pattern	Buffer Size
Open()	4 MiB
Write(size $<$ 2 KiB){5x}	1 MiB
Write(size $<$ 4 MiB) Write(size $<$ 4 MiB)	20 MiB
Write(size $\geq$ 100 MiB)	direct-write

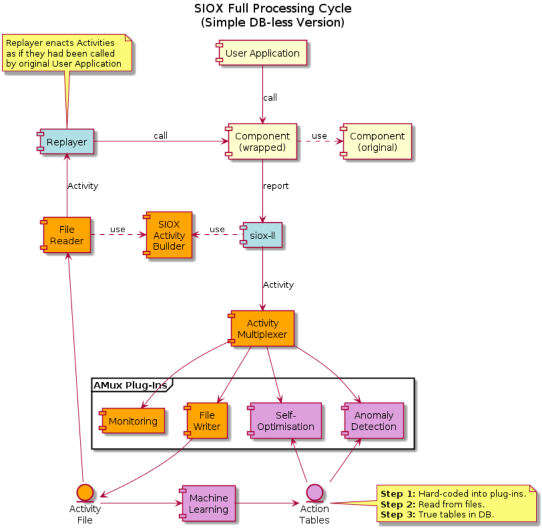
Self-Optimization-Plugin Example 2

A more complex Action Table: Injecting bespoke non-functional calls

Action table for an SOPI `fadvise()` plug-in

Pattern	Advice
SequentialRead() SequentialRead() SequentialRead()	seq & willneed(size)
Open(ext = "nc")	willneed(0, 20 KiB)
Open(ext = "dat")	noReuse & random
RandomWrite(size $<$ 4K){5x}	noReuse & random

Towards a First Prototype

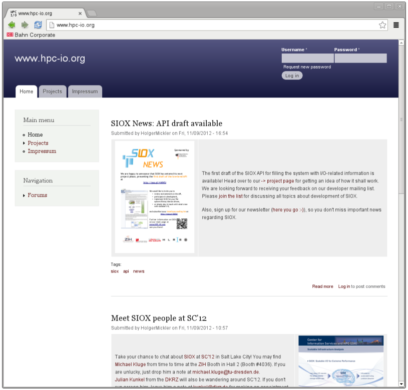


- Application behavior can be recorded in files
- Activities and their metrics read from files
- Replayer to mimic program behavior
- Machine learning restricted to parameters in heuristics

Summary

- SIOX aims to capture and optimize I/O
  - on all layers and filesystems
- Intelligent filtering reduces log size
- Integrated reasoning tries to localize causes and bottlenecks
- We are building a flexible and open system

Finally: SIOX and You



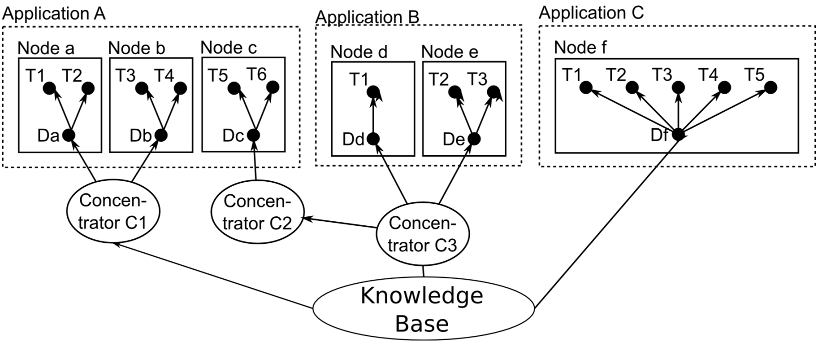
- Think we missed a problem?
- Think you could solve one?
- Like to see SIOX on your favourite file system?

We cordially invite you to become involved at

http://www.HPC-IO.org

Backupslides

Scalability through Hierarchical Data Transport



The Data Deluge – A Numerical Example (1)

A program writes a 1 GiB file to a parallel file system...

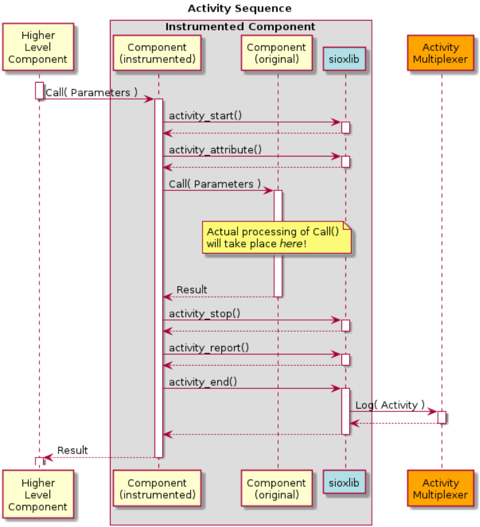
- ...of 100 I/O servers managing 5,000 storage devices
- ⇒ 200 KiB per device to write...
- ...writing 4 KiB per block on device
- ⇒ 250,000 blocks to write...
- ...logging 20 B per block written
- ⇒ 5 MiB logging data
- ⇒ *0.5 % logging overhead...*

The Data Deluge – A Numerical Example (2)

The HPC Cluster *Blizzard* at DKRZ reads and writes...

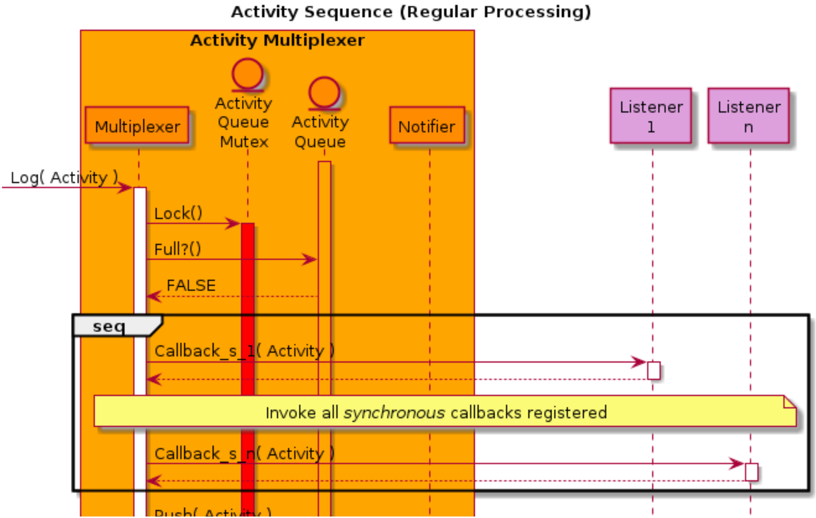
- 10 GiB/s, 24/7, 365 days a year
- ⇒ 50 MiB/s to log for SIOX
- ⇒ 1,576 PiB/a logging information

Instrumentation and the Activity Multiplexer



Activity Multiplexer Normal Behavior

Click to access the PNG of the design



Activity Multiplexer Throttling (Overflow) Behavior

Click to access the PNG of the design