BoF: Autonomic I/O Optimization

Julian Kunkel[1] Michaela Zimmer[1]
Alvaro Aguilera[2] Holger Mickler[2]

1 University of Hamburg
2 ZIH Dresden

June 23th – 2014, ISC

Outline

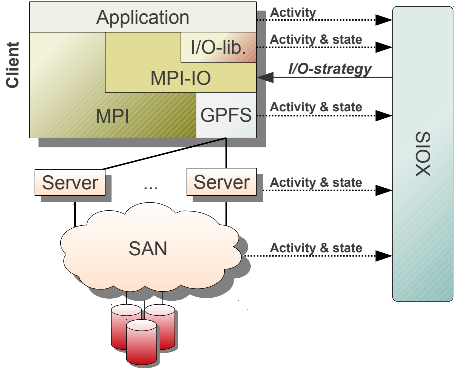
Autonomic I/O Optimization?

- There are many options to tune the I/O-stack, e.g.
  - MPI hints, HDF5 properties, open flags, cache size, `posix_fadvise()`
  - Command line tools: `lfs setstripe`
  - Setup/initialization of a storage system
  - Environment variables
- Many options are of technical nature
  - Performance gain/loss depend on hardware, software
  - Specific to file system, API (MPI, POSIX, HDF5)
  - Many types of hints/tweaks are not portable
- Performance loss forces us to use these optimization
- Performance-portability is unpredictable

Goals of this BoF

Discussion of
(Semi-)Automatic detection and healing of performance issues!

Towards Autonomic Optimization with SIOX



SIOX will

- collect and analyse
  - activity patterns and
  - performance metrics

in order to

- assess system performance
- locate and diagnose problem
- *learn & apply optimizations*
- *intelligently steer monitoring*

Partners and Funding

Bundesministerium
für Bildung
und Forschung

- Funded by the BMBF
  Grant No.: 01 IH 11008 B
- Start: Juli 1st, 2011
- End: September 30, 2013

DKRZ
DEUTSCHES KLIMARECHENZENTRUM

HLRS

IBM

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

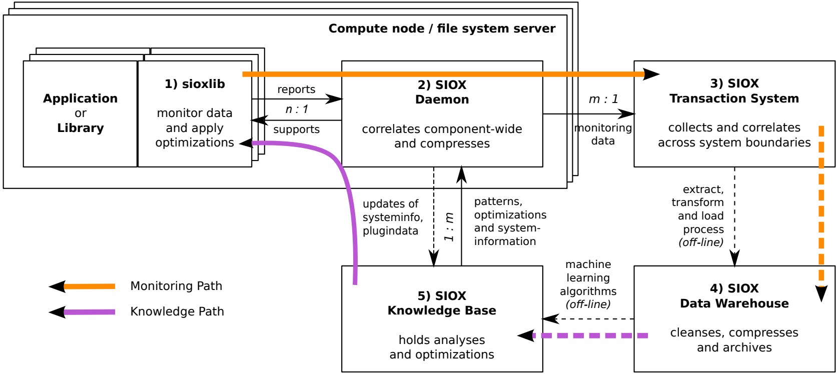
ZIH
Zentrum für Informationsdienste und Hochleistungsrechnen

Outline

Modularity

- The SIOX architecture is flexible and developed in C++ components
- License: LGPL, vendor friendly
- Upon start-up of (instrumented) applications, modules are loaded
- Configuration file defines modules and options
  - Choose advantageous plug-ins
  - Regulate overhead
- For debugging, modules may create **reports**
  - May gather statistics of (application) behavior / activity
  - Provide (internal) usage or overhead statistics
  - These reports can be output at "application" termination

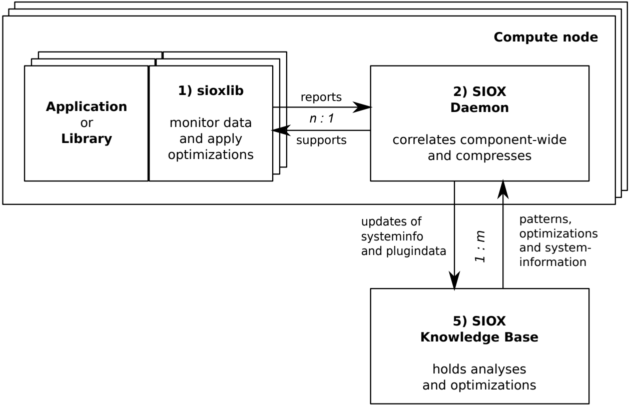
Faces of SIOX (1): General System Architecture



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

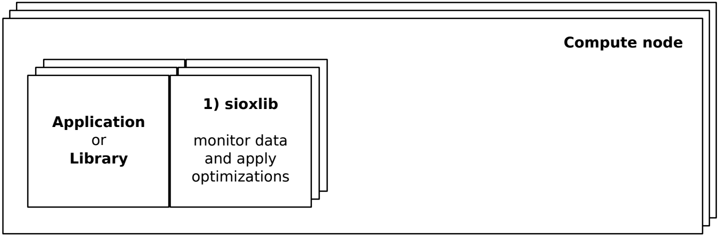
Faces of SIOX (2): Configuration for Online Mode

No pattern recording, optimization without machine learning



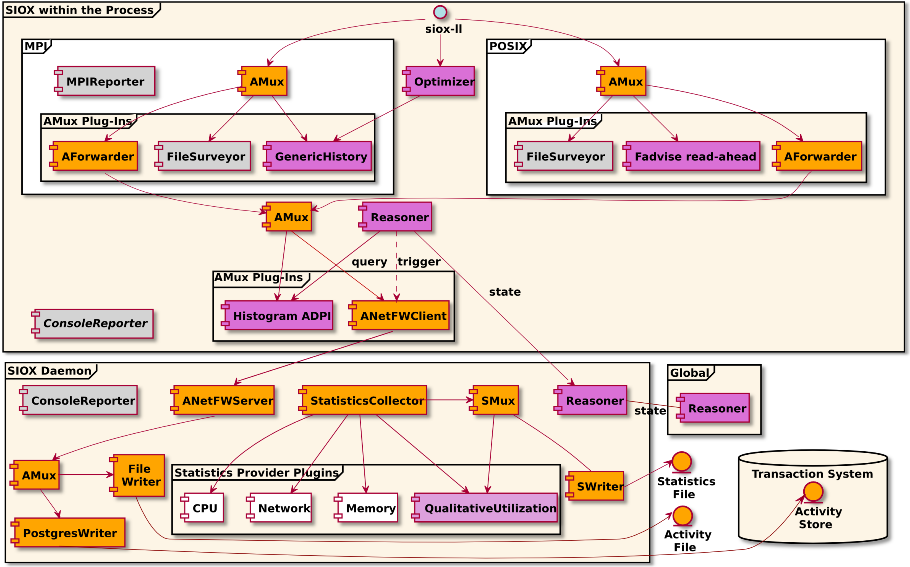
Faces of SIOX (3): Configuration for Static Knowledge

Apply static best-practices with low overhead.

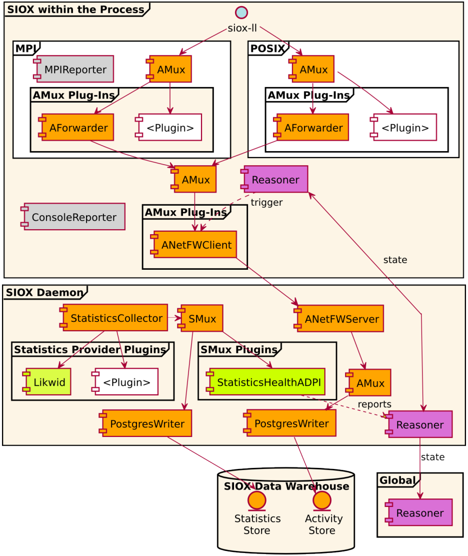


A configuration with a node-global daemon is also possible

Module Interactions of an Example Configuration



Intelligent Monitoring – Controlled by Energy Consumption



Features of the Working Prototype

- Monitoring
  - Application (activity) behavior
  - Ontology and system information
  - Data can be stored in files or Postgres database
  - Trace reader
- Daemon
  - Applications forward activities to the daemon
  - Node statistics are captured
  - Energy consumption (RAPL) can be captured
- Activity plug-ins
  - *GenericHistory* plug-in tracks performance, proposes MPI hints
  - Fadvise (ReadAhead) injector
  - *FileSurveyor* prototype – Darshan-like
- Reasoner component (with simple decision engine)
  - Intelligent monitoring: trigger monitoring on abnormal behavior
- Reporting of statistics on console or file (independent and MPI-aware)

System Configuration

Test system

- 10 compute nodes
- 10 I/O nodes with Lustre

Compute Nodes

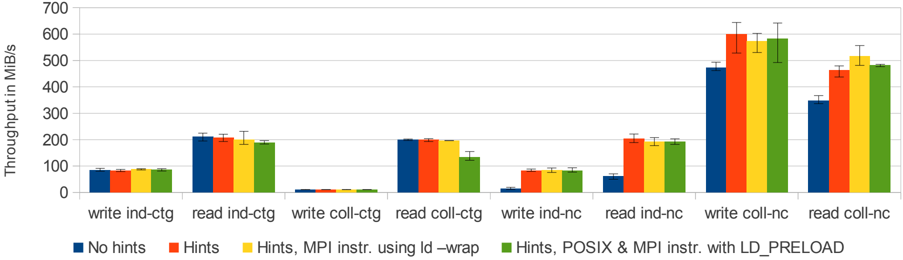
- Dual-socket Intel Xeon X5650@2.67 GHz
- Ubuntu 12.04
- Applications are compiled with: GCC 4.7.2, OpenMPI 1.6.5

I/O Nodes

- Intel Xeon E3-1275@3.4 GHz, 16 GByte RAM
- Seagate Barracuda 7200.12 (ca. 100 MiB/s)
- CentOS 6.5, Lustre 2.5

MPI 4-levels-of-Access

- Each process accesses 10240 blocks of 100 KiB
- Several hint sets are evaluated

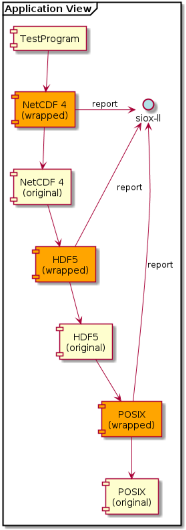


Performance comparison of the 4-levels-of-access on our Lustre file system. The hints increase the collective buffer size to 200 MB and disable data sieving.

Summary

- SIOX aims to capture and optimize I/O
  - on all layers and file systems
- **We are building a modular and open system**
- Intelligent monitoring: Reasoner triggers based on abnormality
- We can change behavior without modifying code!
  - Design the optimization once, apply on many applications
- **Remark: We are contributing components to Exascale10**

Low-Level API – Overview and Instrumentation



- C-Interface for monitoring / analysis
  - Monitor activities and system statistics
  - Query suitable optimization
- Relies on modules to
  - store activities
  - store and query (ontology and system) information
- Instrumentation uses low-level-API
  - A tool and workflow is provided; already instrumented:
    - POSIX (stdio and low-level)
    - MPIIO
    - NetCDF (initial)
    - HDF5 (initial)

Instrumentation

Workflow

- Annotation of header file
- Tool siox-wrapper-generator creates libraries
  - Run-time instrumentation with LD_PRELOAD
  - Compile-time instrumentation using ld -wrap
- siox-inst tool simplifies instrumentation

Header annotations for MPI_File_write_at()

```
//@activity
//@activity_link_size fh
//@activity_attribute filePosition offset
//@splice_before ''int intSize; MPI_Type_size(datatype, &intSize);
                   uint64_t size=(uint64_t)intSize*(uint64_t)count;''
//@activity_attribute bytesToWrite size
//@error ''ret!=MPI_SUCCESS'' ret
int MPI_File_write_at(MPI_File fh, MPI_Offset offset, void * buf, int count,
                      MPI_Datatype datatype, MPI_Status * status);
```

Optimization Plug-in: Read-Ahead with Fadvise

- Plug-in injects `posix_fadvise()` for strided access
- vs. no prefetching vs. in code embedded execution
- Compute "Benchmark" reads data, then sleeps
  - 100 μs and 10 ms for 20 KiB and 1000 KiB stride, respectively

Results

Experiment	20 KiB stride	1000 KiB stride
Regular execution	97.1 μs	7855.7 μs
Embedded fadvise	38.7 μs	45.1 μs
SIOX fadvise read-ahead	52.1 μs	95.4 μs

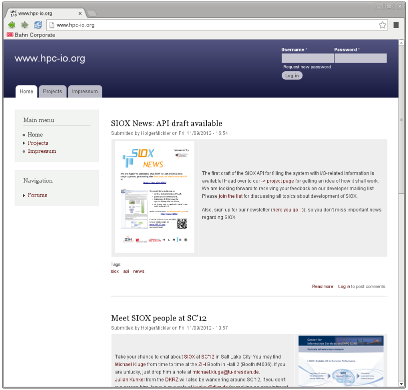
Time needed to read one 1 KiB data block in a strided access pattern.

Discussion

Other Activities at ISC

- **Research Poster 12**: *SIOX: An Infrastructure for Monitoring and Optimization of HPC-I/O*
- **Wednesday 11:00**: *BoF 17: Towards Exascale I/O with E10*
- **Thursday 10:30**: *Research Paper: The SIOX Architecture – Coupling Automatic Monitoring & Optimization of Parallel I/O*

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

Activity Patterns: Example Cause-and-Effect Chain