Research Group
German Climate Computing Center

Footprinting Parallel I/O

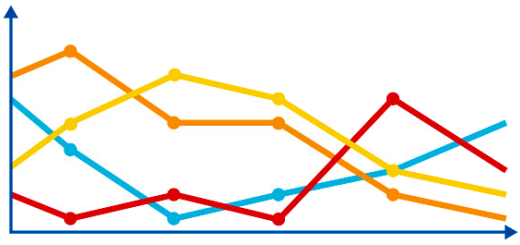
Machine Learning to Classify Application's I/O Behavior

ISC 2019 - IODC Workshop
20th June 2019

Eugen Betke, Julian Kunkel

Table Of Contents

DKRZ's Mistral Supercomputer



Goals

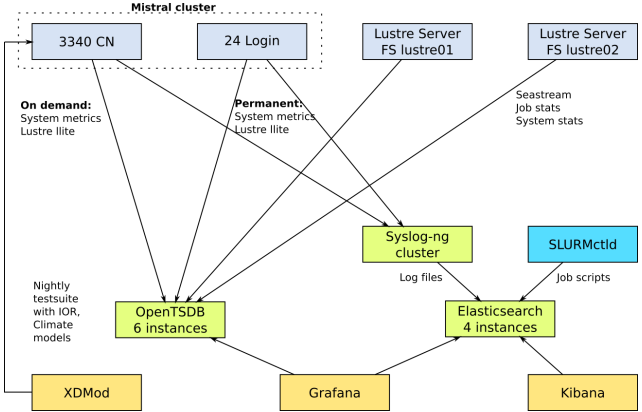
Motivation

- Understanding the workload of the Mistral Supercomputer.

Goals

- Monitoring system development
  - A flexible and extensible monitoring system
  - A portable solution for the next HPC generation
- Establishing analysis workflows
  - Identification of problematic applications and key workloads
  - Understanding of typical I/O patterns
- Tooling
  - **Automatic identification of inefficient applications** (long term goal)

DKRZ Supercomputer and Monitoring



- The Mistral Supercomputer
  - 3,340 client nodes
  - 24 login nodes
  - 2 Lustre file systems
  - Slurm workload manager
- Monitoring System is built of
  - open source components
  - a self-developed data collector
- Provides statistics about
  - login nodes
  - user jobs
  - workload manager queue

Captured Metrics

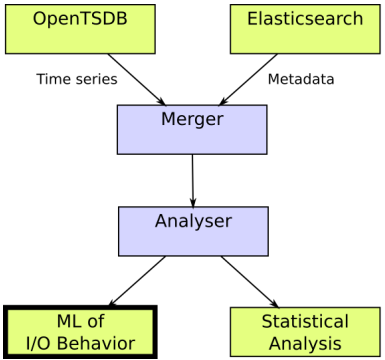
Metadata metrics from `/proc/fs/lustre/llite/lustre*-*/stats`

1	`md_read`	getattr + getxattr + readdir + statfs + listxattr + open + close
2	`md_mod`	setattr + setxattr + mkdir + link + rename + symlink + rmdir
3	`md_file_create`	create
4	`md_file_delete`	unlink
5	`md_other`	truncate + mmap + ioctl + fsync + mknod

I/O metrics from `/proc/fs/lustre/llite/lustre*-*/read_ahead_stats`

6	`read_bytes`	Application I/O requests to Lustre.
7	`read_calls`	
8	`write_bytes`	
9	`write_calls`	
10	`osc_read_bytes`	Lustre object storage client requests to object storage.
11	`osc_read_calls`	
12	`osc_write_bytes`	
13	`osc_write_calls`	

Analysis Workflow



- A daemon process iterates over all jobs
- Each job is fetched and analysed
  - Elasticsearch provides metadata
  - OpenTSDB provides I/O time series
- Output
  - I/O data is stored in separate JSON files
  - Job statistics
  - Sequence of I/O behaviour
- The tool is in an early development stage

Aggregated Data in JSON format: a Sample

Metadata


```
{
 "metadata": {
   "_source": {
     "@start": "2018-12-04T11:31:57",
     "@end": "2018-12-04T11:32:23",
     "elapsed": 26,
     "cpu_hours": 0.057778,
     "total_nodes": 1,
     "nodes": " m11515 ",
     "ntasks_per_node": 8,
     "ntasks": 8,
     "total_cpus": 8,
     "cpus_per_task": 1,
     "jobname": "mkmpost",
     "jobid": 14407,
     "state": "COMPLETED",
     "user_id": 237,
     "group_id": 210,
     "username": "m3",
     "groupname": "mpis",
     "account": "ba09",
   }}}
```


Time Series


```
"ts": {
 "read_bytes": [
   {
     "metric": "host.lustre.stats.read.bytes",
     "dps": {
       "1515756295": 5104980744214,
       "1515756305": 5104980753366,
       "1515756310": 5104980867566,
       "1515756290": 5104980741946,
       "1515756300": 5104980753366
     },
     "aggregateTags": [],
     "tags": {
       "name": "lustre01",
       "system": "mistral",
       "host": "m10753"
     }
   }
 ]
}
```

Footprint Representation

- Mapping of captured job data to a fixed length vector
- Each element represents weighted I/O behaviour

Goal

$$\text{footprint}(\text{jobid}) = \vec{v}_{\text{jobid}} \quad (1)$$

with $\vec{v}$ is a fixed length numeric vector

Footprint Representation

- Mapping of captured job data to a fixed length vector
- Each element represents weighted I/O behaviour

Example

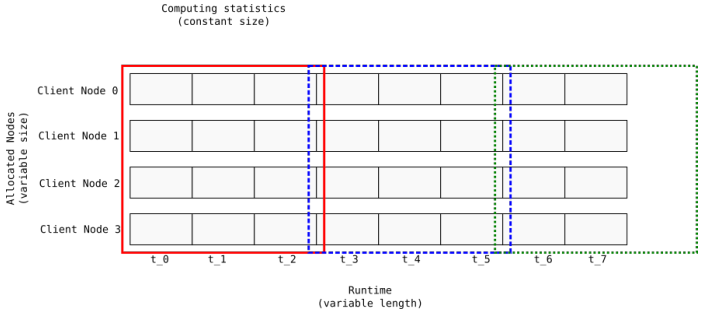
$$\text{footprint}(14400233) = \begin{pmatrix} X1:3 \\ X2:1 \\ X3:3 \\ X4:1 \end{pmatrix} \quad (1)$$

I/O Behavior

X1:	Metadata intensive
X2:	Using I/O node
X3:	Highly parallel I/O
X4:	No I/O

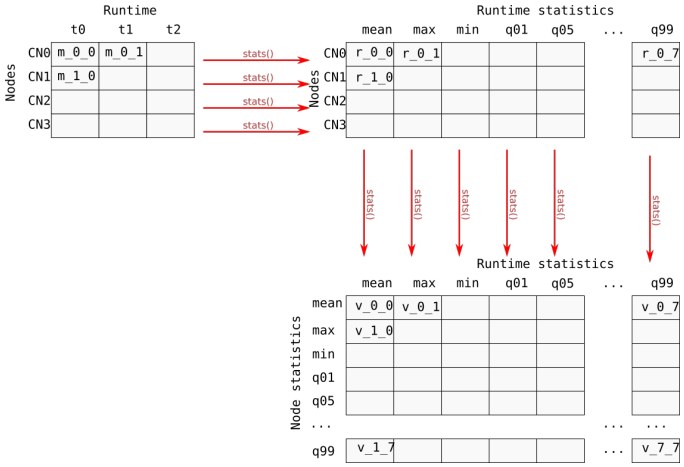
Segmentation

- Problem
  - Number of nodes is variable
  - Segment size to large / too many segments
- Solution
  1. Split data in 2D segments
  2. Convert to $n \times n$ matrix
     - for each segment and
     - for each metric



Fixed Length Statistic Matrix

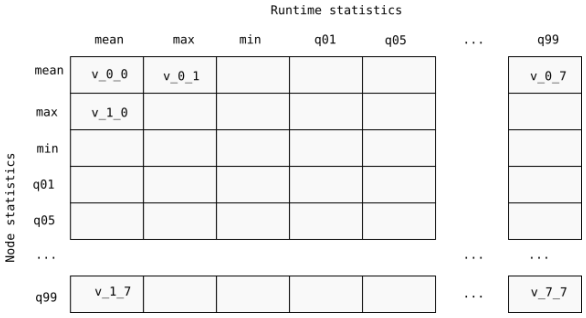
$$\text{stats}\left(\vec{v}\right) = \begin{pmatrix} \text{min} \\ \text{max} \\ \text{mean} \\ \text{q01} \\ \text{q10} \\ \text{q05} \\ \text{q25} \\ \text{q50} \\ \text{q75} \\ \text{q90} \\ \text{q95} \\ \text{q99} \end{pmatrix} \qquad (2)$$



Conversion of a 3x4 segment to a statistic matrix

Statistic Matrix

- Statistics are organized as a 2D-matrix
- stats $(\vec{v})$ is applied to both axis
  - x-axis combines runtime
  - y-axis combines nodes
- The computation is done
  - for each segment
  - for each of 13 metrics
- Similar matrices represent a kind of I/O behavior



Resulting segment size after conversion

12 stats on x-axis * 12 stats on y-axis * 13 metrics = 1872 floating-point values

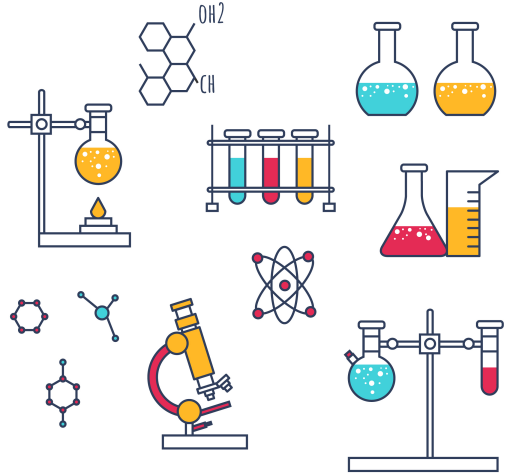
Footprint Example

Sequence of I/O behavior

```
sequence(14496682) =
[2 3 3 2 2 3 2 2 3 1 3 2 1 4 4 1 0 0 0 1 1 1 1 5 5 5 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 6 1 1 1 1 1 6 1 1 1 1 1 6 6 1 1 1 1 1 1 1 1]
```

Absolute and normalized footprints summarize I/O behavior.

```
footprint(14496682) =  [3, 44, 6, 5, 2, 5, 4, 0]
footprint_norm(14496682) = [0.04, 0.64, 0.09, 0.07, 0.03, 0.07, 0.06,
```



Test Dataset Statistics

- Data from 5 days
  - from 2018-12-07 to 2018-12-13
- 70846 jobs statistics downloaded in JSON format
  - uncompressed size is 360GB
- **33,000 (47%) jobs are evaluated**, that
  - contain non-empty time series and
  - and have exit status `COMPLETED`

Exit status statistics

JOBS	EXIT STATUS
1,026	CANCELLED
63,636	**COMPLETED**
5,753	FAILED
3	NODE_FAIL
426	TIMEOUT

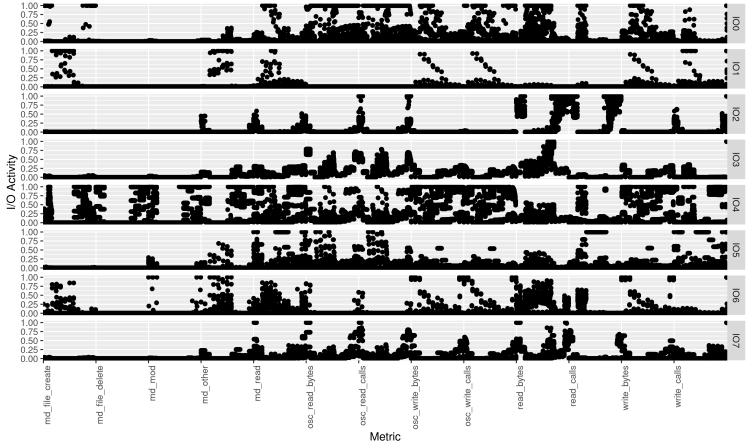
Slurm statistics

JOBS	SLURM PARTITION
37,989	**compute,compute2**
241	gpu
828	miklip
34	minerva
31,752	shared,prepost

Training Parameters

- Data from 33,000 jobs are converted to 128,000 segments
  - Each segment is 10 minutes long
  - Segments/jobs that are shorter are omitted
- Algorithm: kMeans
  - Input: segments (segment size is 1,872)
    - 50,000 samples are used for training
    - Other samples are labeled with the trained model
  - Output: 8 clusters

I/O Classes



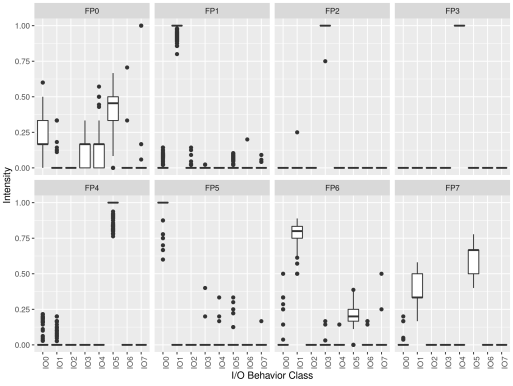
Visualization of 10 samples of each class

I/O Class	Percent	Count
IO0	1.12	1436
IO1	91.23	117176
IO2	0.01	13
IO3	0.36	459
IO4	0.43	548
IO5	6.83	8767
IO6	0.02	21
IO7	0.02	21

Footprint Clustering Parameters

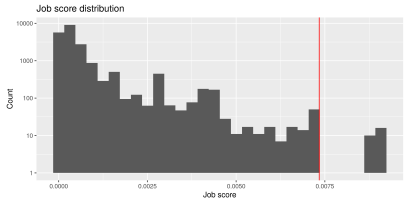
- Footprints for each of 33,000 jobs are created
- Footprints are normalized, to make them independent to job runtime
- Algorithm: kMeans
  - Input: Footprints
  - Output: 8 clusters

Footprint Categories and Distribution



Class	Percent	# Jobs
FP0	0.34	71
FP1	77.29	16003
FP2	1.96	406
FP3	0.78	162
FP4	12.59	2606
FP5	0.78	161
FP6	4.19	868
FP7	2.06	427

Score vs. Footprints



Red line represents the 0.998-quantile

- Metric scores
  - 0: Normal I/O activity
  - 1: High I/O activity
  - 2: Critical I/O activity
- Job score $= \frac{\text{Sum of all metric scores}}{\text{Job runtime}}$

- 53 jobs show critical average I/O activity
  - Determined by the 0.998-quantile
  - In histogram right to the red line
- Intersection of score with footprinting

Footprint		Match	
Class	Count	Count	in %
FP0	71	0	0
FP1	16003	12	22.64
FP2	406	0	0
FP3	162	0	0
FP4	2606	24	45.28
FP5	161	2	3.77
FP6	868	7	13.21
FP7	427	8	15.09

Summary

- DKRZ monitoring system
  - **Open source** components + self-developed collector
  - **Portable** to the next HPC and other machines
- Job-Footprinting
  - Fine-grained **Categorization** of jobs
  - **Insight look** in the job
- Good evaluation approach required