Research Group
German Climate Computing Center

Classifying Temporal Characteristics of Job I/O Patterns

Using Machine Learning Techniques

Sig-IO-UK
April 23, 2020

Eugen Betke, Julian Kunkel

Table Of Contents

Motivation

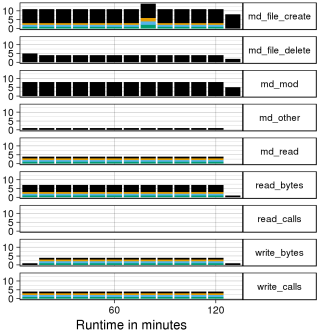
1. Understanding I/O
   - Procurement of new systems
   - Development of better job scheduling strategy
   - Finding applications with high I/O optimization potential
2. Automatization
   - Ability to analyze tens of thousands of parallel jobs
   - Detection of specific I/O behavior

Monitoring data

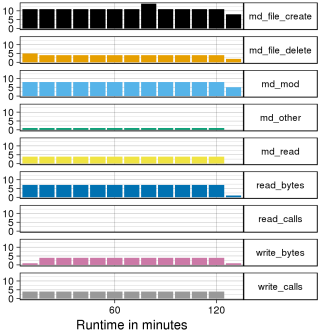
- Raw monitoring data
  - are captured by the DKRZ's monitoring system with a sample rate of five seconds
  - contains nine I/O metrics
    - five metadata and four data metrics
  - are four-dimensional
    - Nodes $\times$ Metrics $\times$ Filesystem $\times$ Runtime
- Categorized monitoring data
  - are aggregated to ten minutes time intervals
  - categorizes mean I/O performance into three categories
    - LowIO, HighIO, CriticalIO
    - is based on our previous work [1] (not yet published)

Categorized monitoring data

Node usage



Metric usage



File system usage

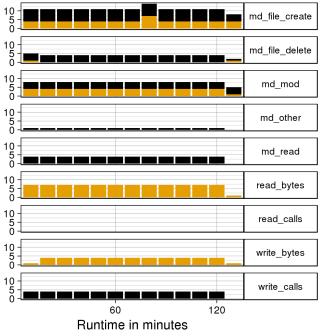


Table Of Contents

Overview

- Datasets
  - are derived from categorized monitoring data
  - contain more than 1.000.000 samples
  - 580.000 samples are doing I/O
- The following datasets are available
  - Job-Metadata
  - Job-IO-Metrics
  - Job-IO-Duration
  - Job-Codings
- Github repository
  - `https://github.com/joobog/job_io_datasets.git`

Job-Metadata

- The dataset contains
  - anonymised user data
  - job configuration
  - job statistics
- Additional information for other datasets
- Can be joined by jobid

Dataset

jobid	user_id	group_id	partition	total_cpus	total_nodes	ntasks	ntasks_per_node	cpus_per_task	@start	@end	exit_code	state	elapsed
8043454	79855	20005	compute2:compute	72	1	2	0	1	2020-02-07T08:11:37	2020-02-07T08:12:49	0:0	COMPLETED	72
7153657	66417	2100	compute:compute2	48	1	1	0	36	2020-03-17T09:32:34	2020-03-17T09:38:37	0:0	COMPLETED	363
7121050	66363	1480	compute	48	1	40	0	1	2020-03-22T03:36:55	2020-03-22T03:38:20	1:0	FAILED	85
7846511	77699	32054	prepost:compute:compute2	48	1	0	0	1	2020-02-19T23:40:32	2020-02-19T23:48:05	0:0	COMPLETED	453

Job-IO-Metrics

- Based on our previous work [1] (not yet published)
- The dataset contains the three job metrics
  - Job-IO-Balance
  - Job-IO-Utilization
  - Job-IO-Problem-Time

Dataset

jobid	utilization	problem_time	balance
8043454	0.0	0.0	
8035741	0.0	0.0	
4618758	0.0	0.0	
8092715	4.0	1.0	1.0
8062925	1.0	0.25	0.21428571428571427
7121050	0.0	0.0	
5134134	1.0	0.08695652173913042	1.0
7082551	0.0	0.0	
7354974	1.0	1.0	1.0

Job-IO-Duration

- Relative I/O duration
  - describes the fraction of time a metric spend doing I/O
- Dataset contains 27 features
  - 9 metrics x 3 categories

Metric
md_file_create
md_file_delete
md_mod
md_other
md_read
read_bytes
read_calls
write_bytes
write_calls

Category	Score
LowIO	0
HighIO	1
CriticalIO	4

Dataset (7/28 columns)

jobid	0_md_file_create	0_md_file_delete	1_md_file_create	1_md_file_delete	4_md_file_create	4_md_file_delete
4385791	1.0	0.5	0.0	0.5	0.0	0.0
4636938	1.0	1.0	0.0	0.0	0.0	0.0
7846511	0.5	0.5	0.5	0.0	0.0	0.5
7153657	1.0	1.0	0.0	0.0	0.0	0.0
7518140	1.0	0.5	0.0	0.5	0.0	0.0

Job-Codings

- Absolute mode coding
  - Whole job is represented as sequence of states
  - Each state describes usage of IO-Metrics
- Hexadecimal coding
  - Each metric is represented as sequence of hexadecimal numbers
  - Describes average usage of a metric

Definition

Job coding is a compact representation of categorized monitoring data.

Absolute mode coding

- Compute a number for each segment, that describes unambiguously the file system usage
  - A metric segment is considered to be active, iff. it does HighIO or CriticalIO
  - In a 9-bit number, each bit position represents the state of metric segment (active or non-active)
  - e.g. iff. md_read (16) and read_bytes (32), then the segment is coded by the value 48

Metric	Code
md_file_create	1
md_file_delete	2
md_mod	4
md_other	8
md_read	16
read_bytes	32
read_calls	64
write_bytes	128
write_calls	256

Absolute coding example

```
{'jobid': 'job1',
'coding': [1, 5, 0, 96, 96, 96, 96, 96, 96, 96],
'length': 15}
```

Hexadecimal coding

- Average (over Nodes and Filesystems) and quantized metric scores
- Quantization function $q(x) \rightarrow y = hex(round(4x))$
  - $x \in [0, 4]$
  - y is a hexadecimal number

Hexadecimal coding example

```
{
'jobid': job1,
'coding': metric
  md_file_create [4,4,0,0,0,0,0,0,0,0,0,0,0,0,0]
  md_file_delete [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
  md_mod         [0,4,0,0,0,0,0,0,0,0,0,0,0,0,0]
  md_other       [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
  md_read        [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
  read_bytes     [0,0,0,0,0,0,0,0,2,2,2,2,2,2,2]
  read_calls     [0,0,0,0,0,0,0,0,2,2,2,2,2,2,2]
  write_bytes    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
  write_calls    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
'length': 15}
```

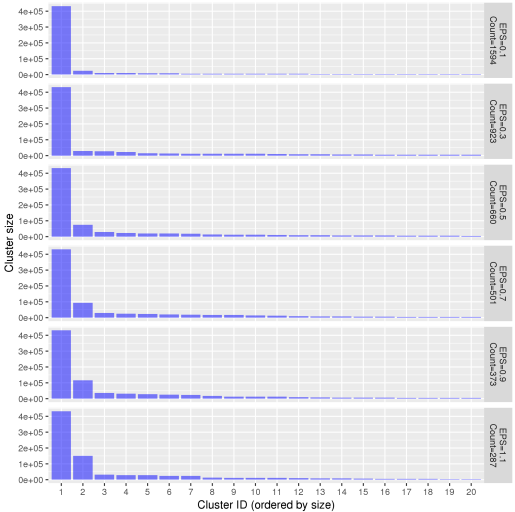
Table Of Contents

Approach

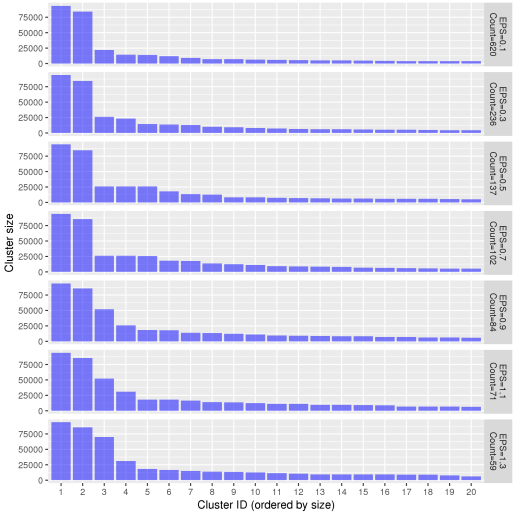
- Job-IO-Duration and Job-Metric datasets
  1. Clustering of 10.000 samples with different configuration
  2. Training of a classification model by using the clustered data
     - Algorithm of choice is “Decision Trees”
  3. Labeling of 1.000.000 samples by using the classification model
- Job-Codings dataset (Absolute mode only)
  1. Compare job similarity by means of Levenshtein distance
  2. Group jobs with high similarity

Clustering by means of general purpose ML algorithms

Job-IO-Duration



Job-Metrics



Conclusions

- Naive approaches don't work well
  - Dilemma
    - Low EPS $->$ too many clusters
    - High EPS $->$ poor clustering results
  - Cluster labeling is difficult

Clustering by means of Levenshtein distance

$$\text{similarity}(\text{job1}, \text{job2}) = 1 - \frac{\text{levenshtein}(\text{coding}_{\text{job1}}, \text{coding}_{\text{job2}})}{\max(\text{length}_{\text{job1}}, \text{length}_{\text{job2}})} \quad (1)$$

- Grouping of 1.000.000 jobs with similarity larger than 0.7 ends up in 19000 Clusters

Definition

Levenshtein distance is the number of changes (insertion, substitutions, deletions) required to convert one word into another, e.g., S**at**u**r**day to Sunday requires 3 changes.

Table Of Contents

Algorithm overview

1. Create hexadecimal representation of monitoring data
2. Detect I/O phases
3. Compute similarity for all jobs (quadratic runtime)
   - Find the best phases matches
   - Find the best phase position match
4. Cluster the most similar jobs
   - e.g. with similarity larger than 0.7

Phase similarity

- Hexadecimal encoding
- Using sliding window approach to find best match

Example: Finding best match

```
phase_coding_1 = [2,2,2,2,8,2,2]
phase_coding_2 = [2,2,2,9,3]
```

```
2222822
22293-- -> 51%
```

$$\frac{\left(\frac{2}{2}+\frac{2}{2}+\frac{2}{2}+\frac{2}{9}+\frac{3}{8}+0+0\right)}{7} = 0.51$$

```
2222822
-22293- -> 65%
```

$$\frac{\left(0+\frac{2}{2}+\frac{2}{2}+\frac{2}{2}+\frac{8}{9}+\frac{2}{3}+0\right)}{7} = 0.65$$

```
2222822
--22293 -> 45%
```

$$\frac{\left(0+0+\frac{2}{2}+\frac{2}{2}+\frac{2}{8}+\frac{2}{9}+\frac{2}{3}\right)}{7} = 0.45$$

Job Similarity

- Find best phase position match
- Compute weighted similarity

Example: Finding best match with highest similarity

```
job_coding_1: [[2,2,2,9,3], [9,1,1]]
job_coding_2: [[2,2,2,2,8,2,2], [1], [8,1,1]]
```

```
-22293-  911 ---
2222822  -1- 811
```

$$\frac{\left(\frac{2}{2}+\frac{2}{2}+\frac{2}{2}+\frac{8}{9}+\frac{2}{3}+\frac{1}{1}\right)}{13} = 0.43$$

```
------- 22293  911
2222822 1----  811
```

$$\frac{\left(\frac{1}{1}+\frac{8}{9}+\frac{1}{1}+\frac{1}{1}\right)}{13} = 0.3$$

```
-22293- - 911
2222822 1 811
```

$$\frac{\left(\frac{2}{2}+\frac{2}{2}+\frac{2}{2}+\frac{8}{9}+\frac{2}{3}+\frac{8}{9}+\frac{1}{1}+\frac{1}{1}\right)}{13} = 0.57$$

Robustness

- Suppose an application with two I/O phases runs on
  - a fast (job_coding_1) and
  - a slow (job_coding_2) file system

Example: Simulation of file system slow down

```
job_coding_1: [[2,2,2,9,3], [9,1,1]]
job_coding_2: [[2,2,2,8,3,2], [8,2,1,1]]
```

```
22293- 911-
222822 8111
```

$$\frac{\left(\frac{2}{2}+\frac{2}{2}+\frac{2}{2}+\frac{8}{9}+\frac{3}{3}+\frac{8}{9}+\frac{1}{2}+\frac{1}{1}\right)}{10} = 0.73$$

Table Of Contents

Summary

- Datasets
  - Job-Metadata, Job-IO-Metrics, Job-IO-Duration, Job-Codings
- Naive machine learning approach
  - Combination of clustering and classification
    - No optimal clustering parameters found
    - Labeling of clusters is difficult
- Phases-Matching-Algorithm (Work in progress)
  - Matches phases and phase positions
  - Sensitive to file system performance

Further Reading I

Eugen Betke, Julian Kunkel.
Semi-automatic Assessment of I/O Behavior by Inspecting the Individual Client-Node Timelines — An Explorative Study on 10^6 *Jobs*
Not yet published