A Performance Analysis of Large Scale Scientific Computing Applications from Logs

Liqiang Cao, Xu Liu, Xiaowen Xu and Zhanjun Liu

HPCC, IAPCM

Schedule

- Motivation
- Log archive
- Characterization of performance
- Summary

1. Motivation

- Performance is a key issue for applications as well as systems
  - Tuning Peak Flops and efficiency of HPL/HPCG for system
  - Many scientists develop physical and mathematical methods for efficiency of applications
  - Users tuning performance of an application by parameters in an HPC
- Performance of an application may varies drastically
  - Run time of an application varies from hours to tens of hours
  - Reason is unclear to user: Application itself? Number of nodes? Environments?
- Few users know of the performance of a job in an HPC
  - Job system in an HPC wraps the run time of applications
    - Time of a job for user = queue time + stage time + run time
  - A user may submit tens of jobs to the job system in a day
    - Have few time to care about run time of each job

Performance audit for HPC jobs

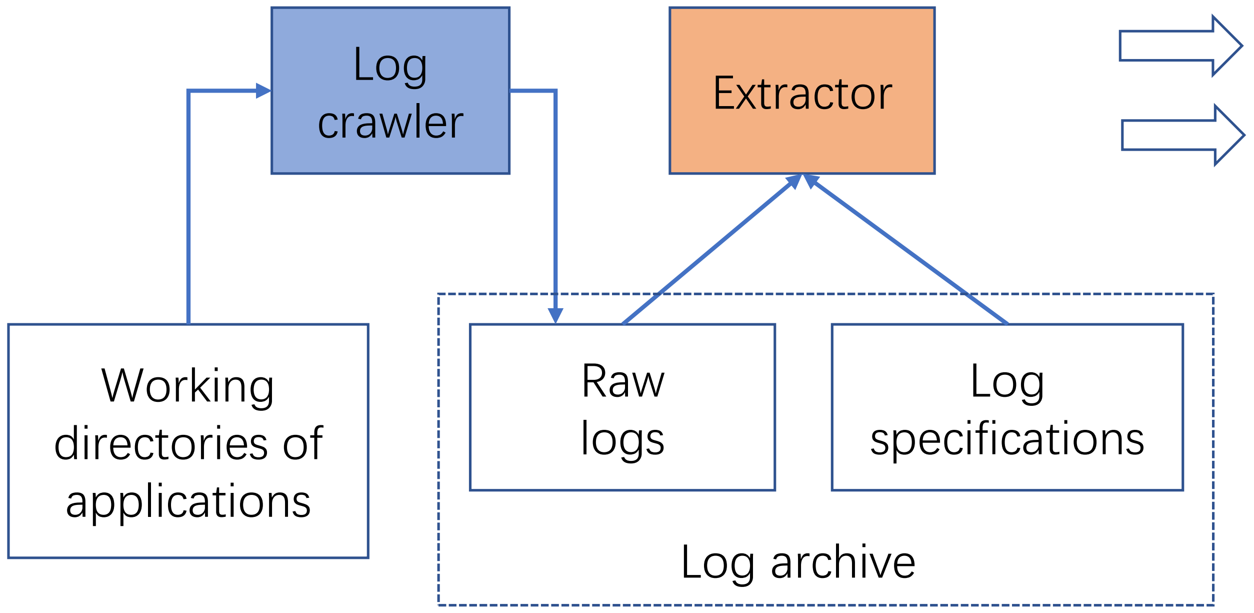
- Performance audit is a way to make performance of HPC jobs sensitive for end users and developers
  - Analyze run time of productive applications by logs
- Possible ways for performance audit of HPC jobs
  - Profiling tools, eg. HPCToolkit, Scalasca
    - Resource consuming, few of them run with productive jobs
    - Profile job by job, seldom for comparative analysis
  - Performance counters
    - System level performance audit(not application level)
  - Code instrumentation and log based methods
    - Negotiate with developers

2. Logs for performance audit

- A log archive includes a log specification and running logs for an application
  - Log specification: log data type and log data format of an application.
  - Running logs: files with running environments, input parameters, run time of key components, etc.
- We have build log archives in last year
  - 364 logs in a HPC system test
  - 1500 logs for 2 applications
    - Time is from 2018.5 to 2018.12
    - More than 100 GB in size

Save logs and extract information

We store logs from crawler and extract information to tables



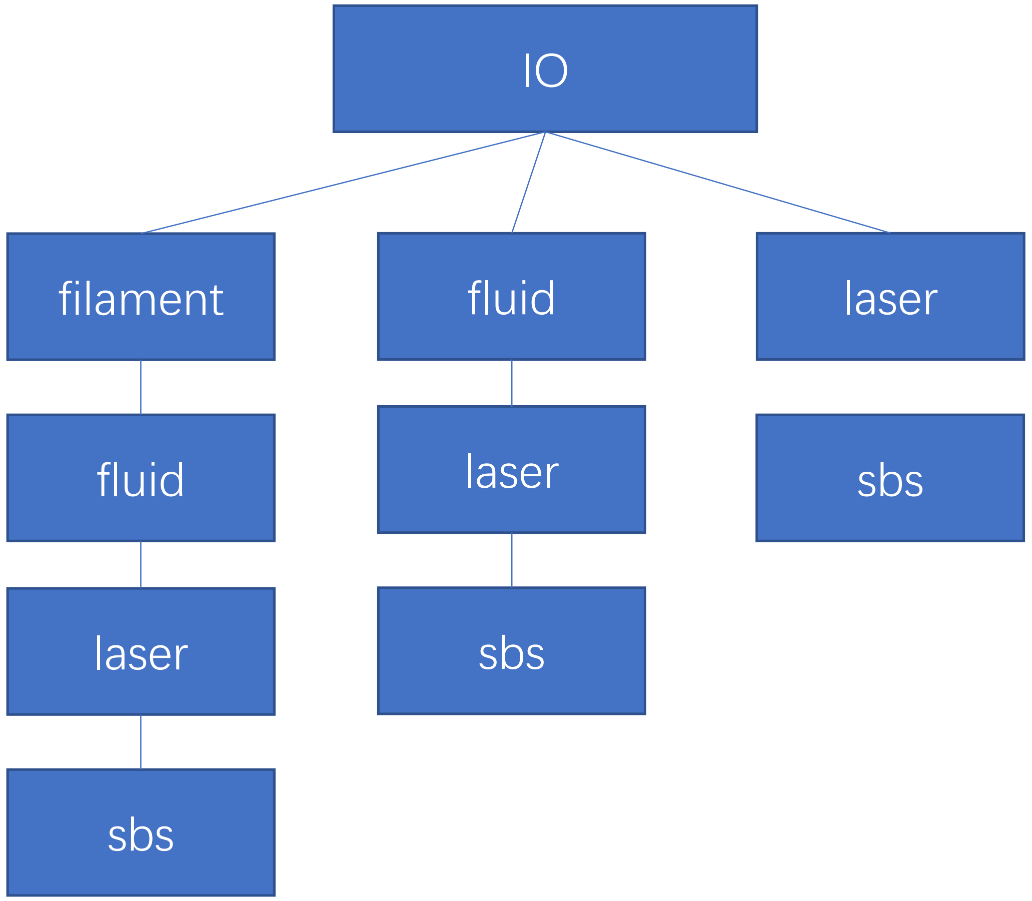
	Seconds	AppType	LogName	... ...
43	0. 7715	APP1	_20180522_051917_565	...
78	2. 6689	APP1	_20180525_064356_964	...
79	2. 6737	APP1	_20180525_064925_594	...
80	2. 6756	APP1	_20180525_065108_554	...
81	2. 677	APP1	_20180525_065236_615	...
82	2. 6835	APP1	_20180525_065532_220	...
83	2. 6822	APP1	_20180525_065555_824	...
84	2. 6845	APP1	_20180525_065805_25	...
85	2. 6768	APP1	_20180525_073355_230	...
135	2. 9442	APP1	_20180530_112742_590	...
136	2. 9852	APP1	_20180530_112742_707	...
137	2. 8771	APP1	_20180530_112748_570	...
138	2. 922	APP1	_20180530_112753_469	...
139	3. 5992	APP2	_20180530_112757_493	...
140	2. 9357	APP2	_20180530_112800_504	...
141	2. 9295	APP2	_20180530_112802_903	...

Structured log data

Log crawler: find the job log and save to log archive
Extractor: convert unstructured logs to structured data with the help of specifications

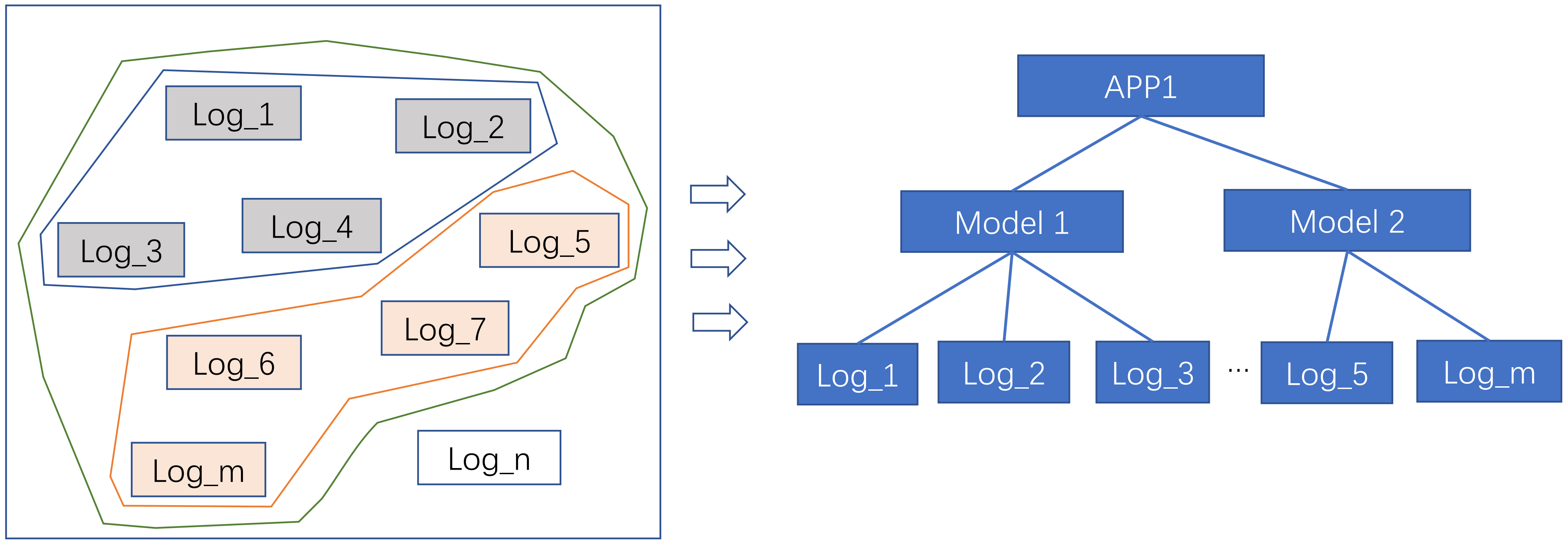
Cluster logs to models

- APP1: group jobs by flow chart
  - Jobs with “filament fluid laser sbs”, “fluid laser sbs” and “laser sbs”are different
  - More than 270 jobs grouped to 5 models (#jobs>10)
- APP2: group jobs by DBSCAN
  - More than 1200 jobs grouped to 14 models (#jobs > 10)
- Groups are divided to subgroups by set of parameters’ name



Clustering of logs by DBSCAN

Vectorized input parameters => DBSCAN => groups of logs

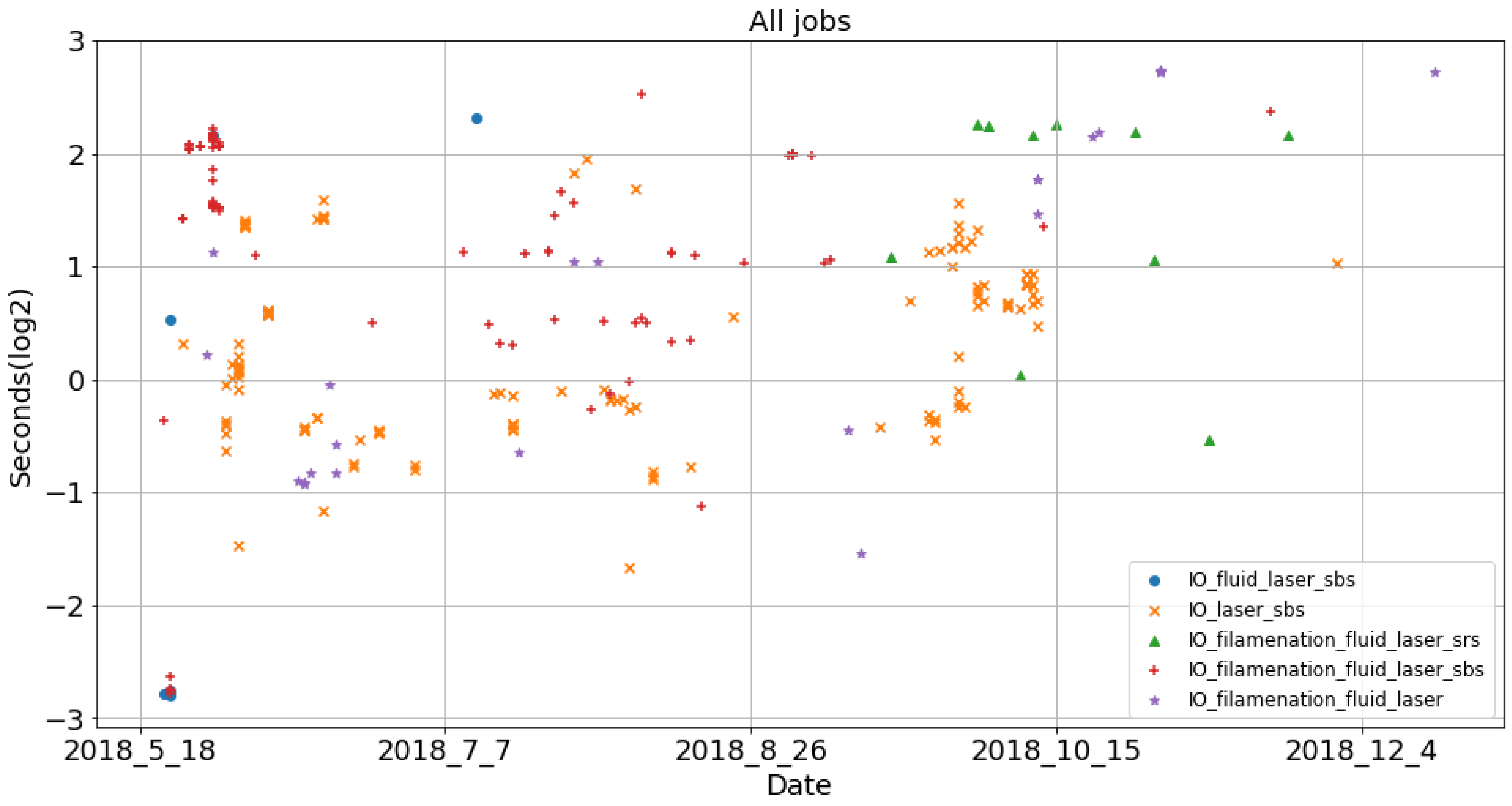


In a test of 368 job logs for 5 applications, the DBSCAN algorithm clustered job logs into 19 models with a contour factor of 0.84.

3. Characterization of performance

- APP1 has about 50 parameter
  - Environment parameters (start time, node name, username, version, etc)
  - Geometry parameters (dimensions, domains, segments, etc)
  - Physical parameters (material, pressure, intensive, etc, about 30 parameters)
- Find performance law by expert’s knowledge
  - We know some of parameters have impact on step time of jobs, for example number of process, geometry size of model
- Find the performance law by machine learning
  - Find out significant parameters on performance by correlation analysis and lasso regression

Overview on step time of APP1



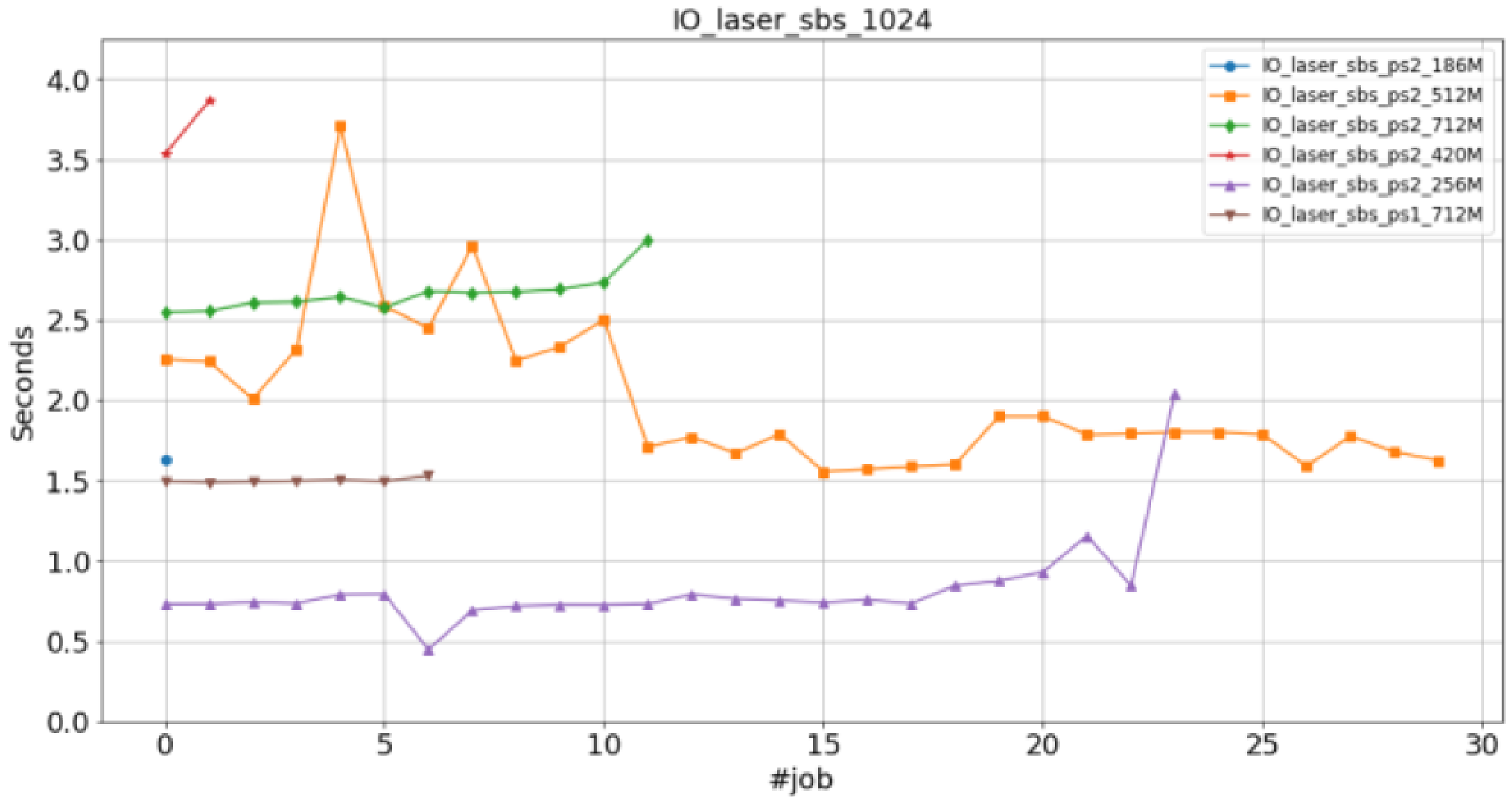
For more than 270 jobs of APP1, the interval of step time is from 0.2 to 8 seconds.

Analyze logs by expert’s knowledge

Experts said number of process, geometry and PS parameter of APP1 have big impact on step time

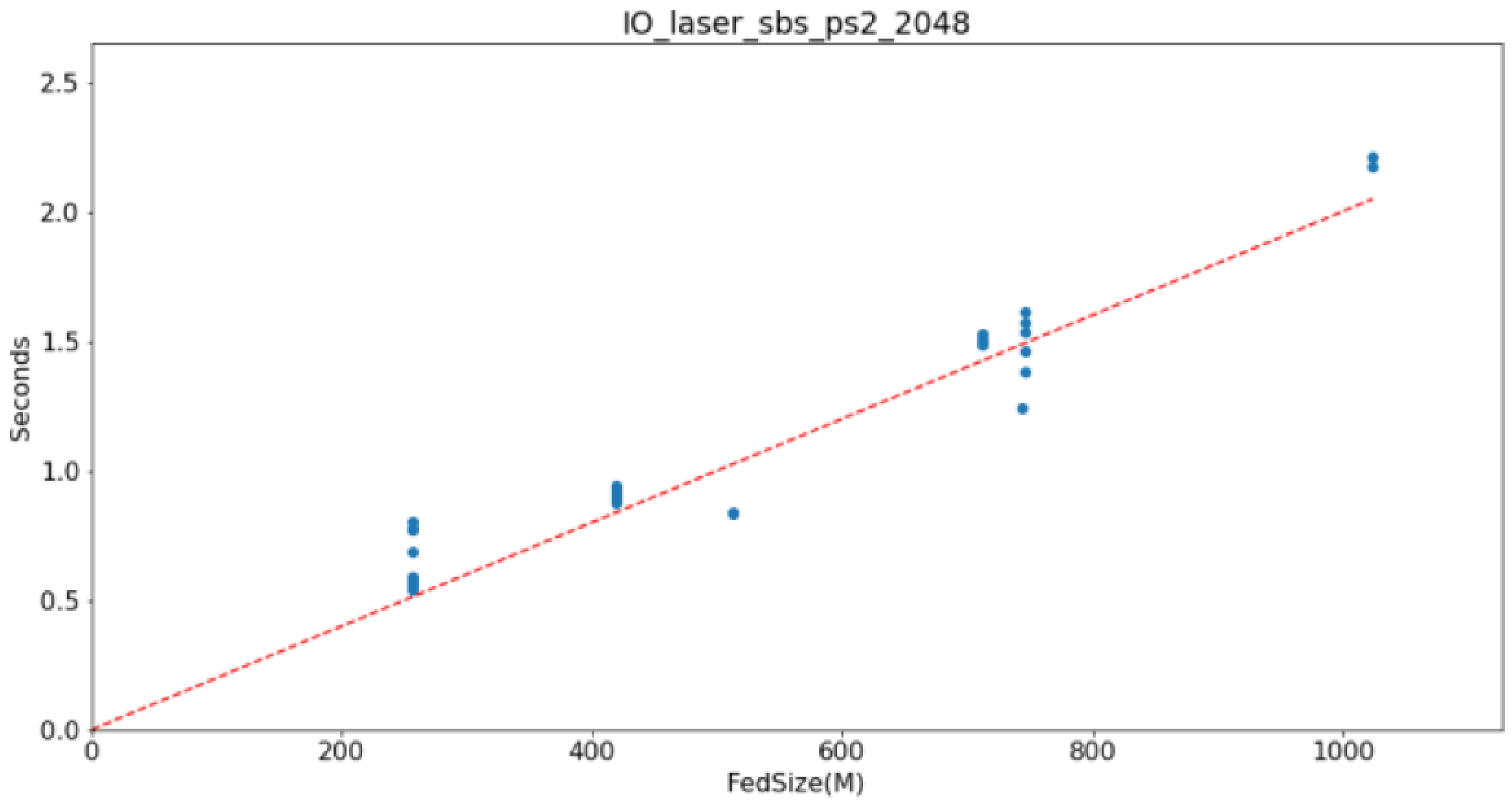
Step time of IO_laser_sbs model

Nprocess = 1024



Step time of jobs with fixed Nprocess, geometry and PS parameter is in an time interval

Regression on IO_laser_sbs_ps2 with 2048 process

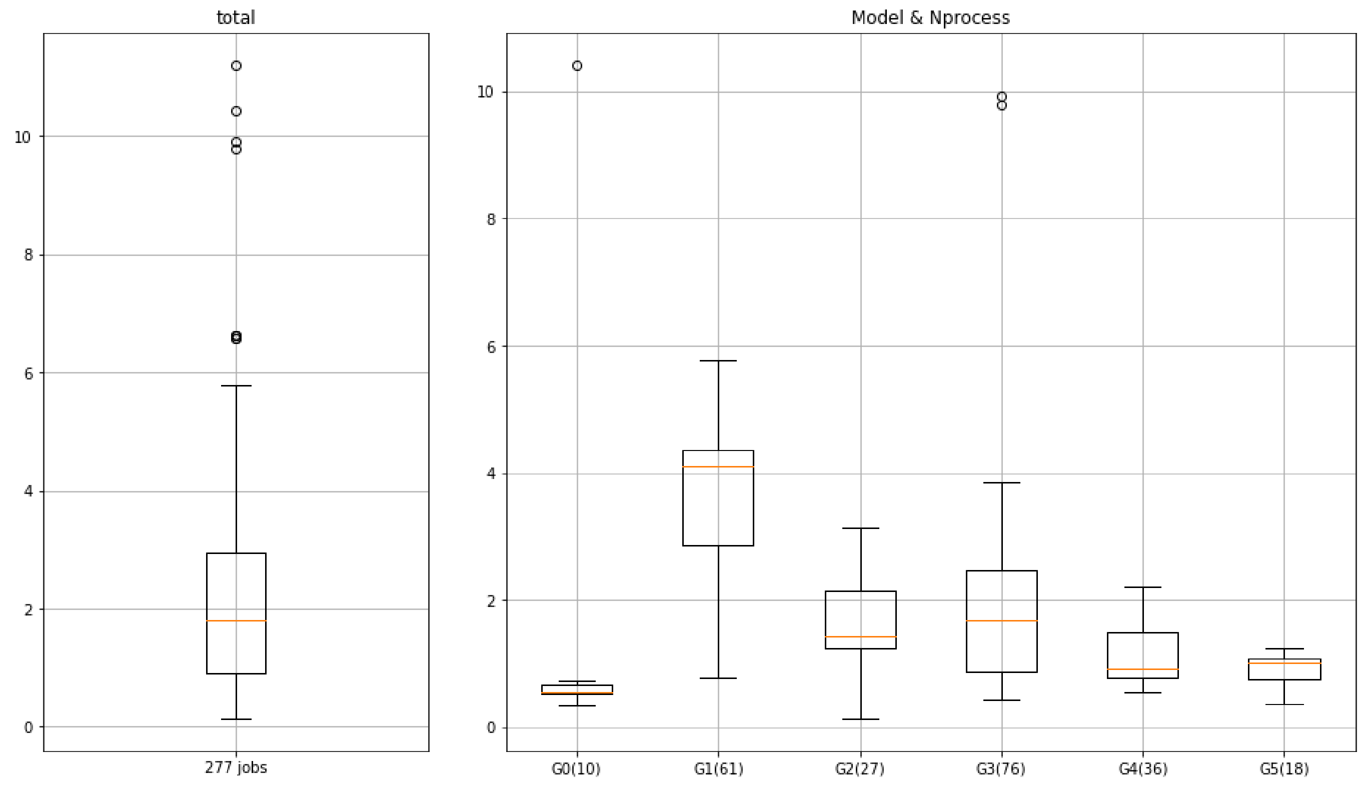


the average step time for these jobs is better fitted to a straight line with a coefficient of 0.0020.

Analyze logs by machine learning

Can we build a performance model from parameters?

Group jobs by Model and Nprocess



For jobs of APP1, the high quartile and low quartile interval of step time is about 2 seconds, if we group jobs by model and Nprocess, the intervals are changed from 0.2 to 1.7 seconds

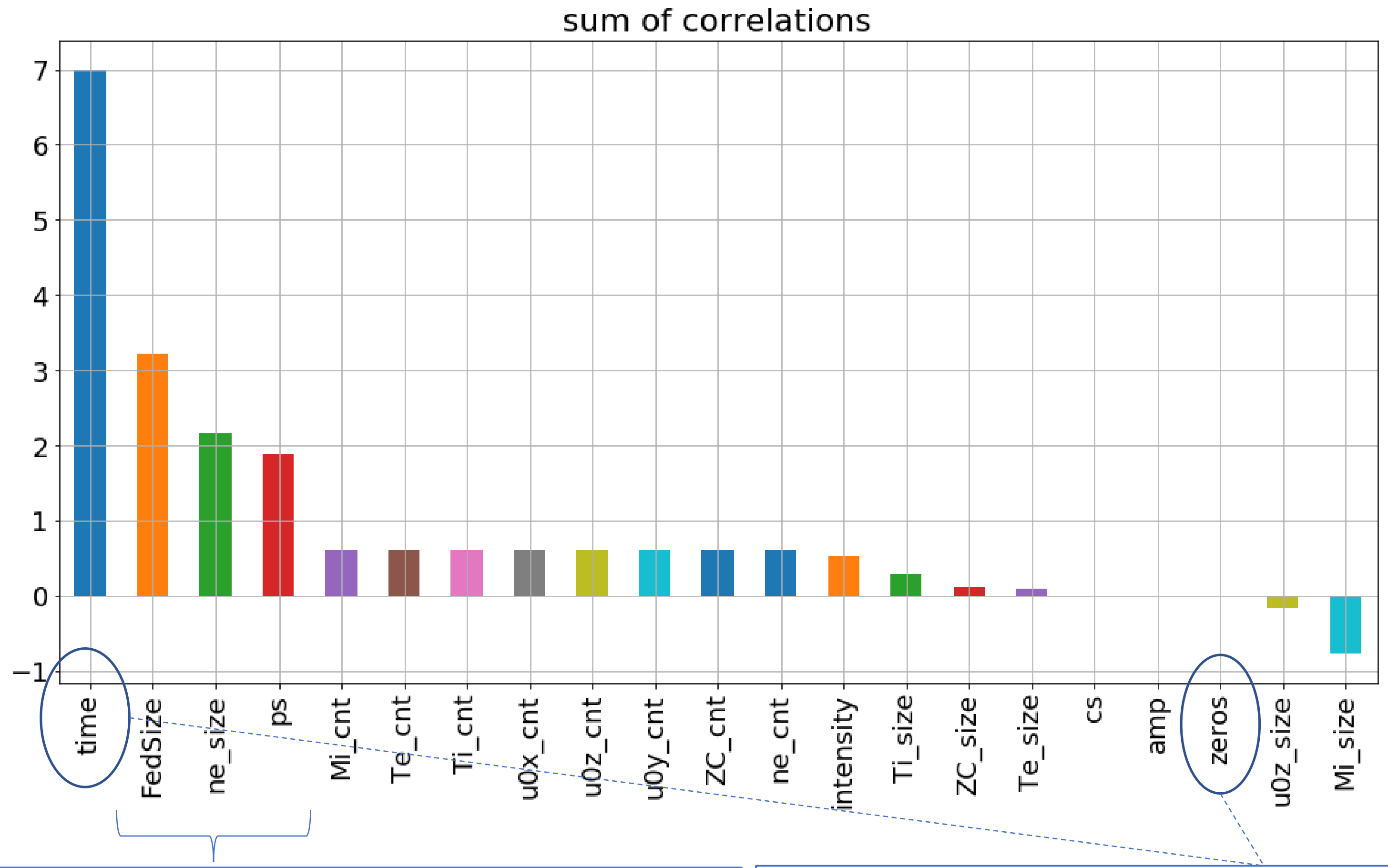
Find the top K performance parameters

- Correlation analysis between parameters and step time

$$R = \frac{\sum_1^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_1^n (x_i - \bar{x})^2 \sum_1^n (y_i - \bar{y})^2}}$$

  - If ‘R’ is equal to 1, then there is perfect positive correlation between two values;
  - If ‘R’ is equal to -1, then there is perfect negative correlation between two values;
  - If ‘R’ is equal to zero, then there is no correlation between the two values.
- Lasso regression on parameters
  - A method in sklearn
  - It prefers solutions with fewer non-zero coefficients. It can effectively reducing the number of features upon which the given solution is dependent.

Correlation coefficients of parameters with step time

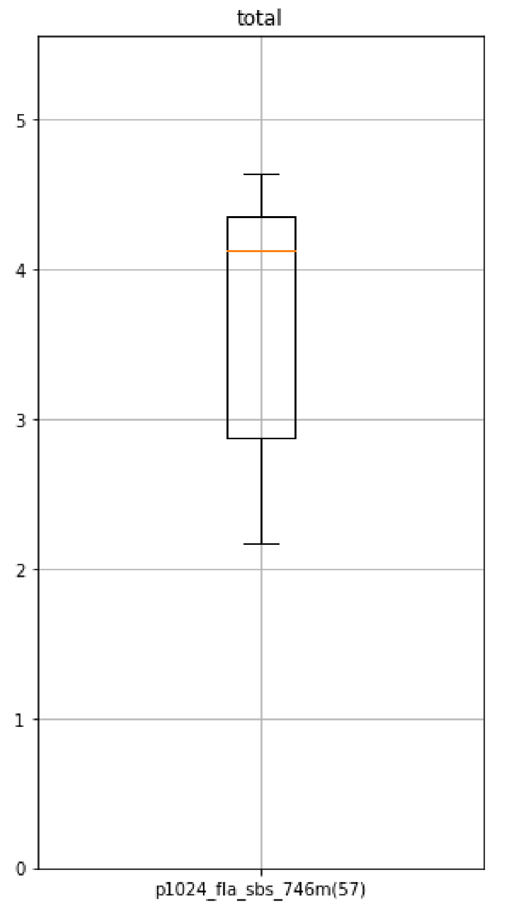


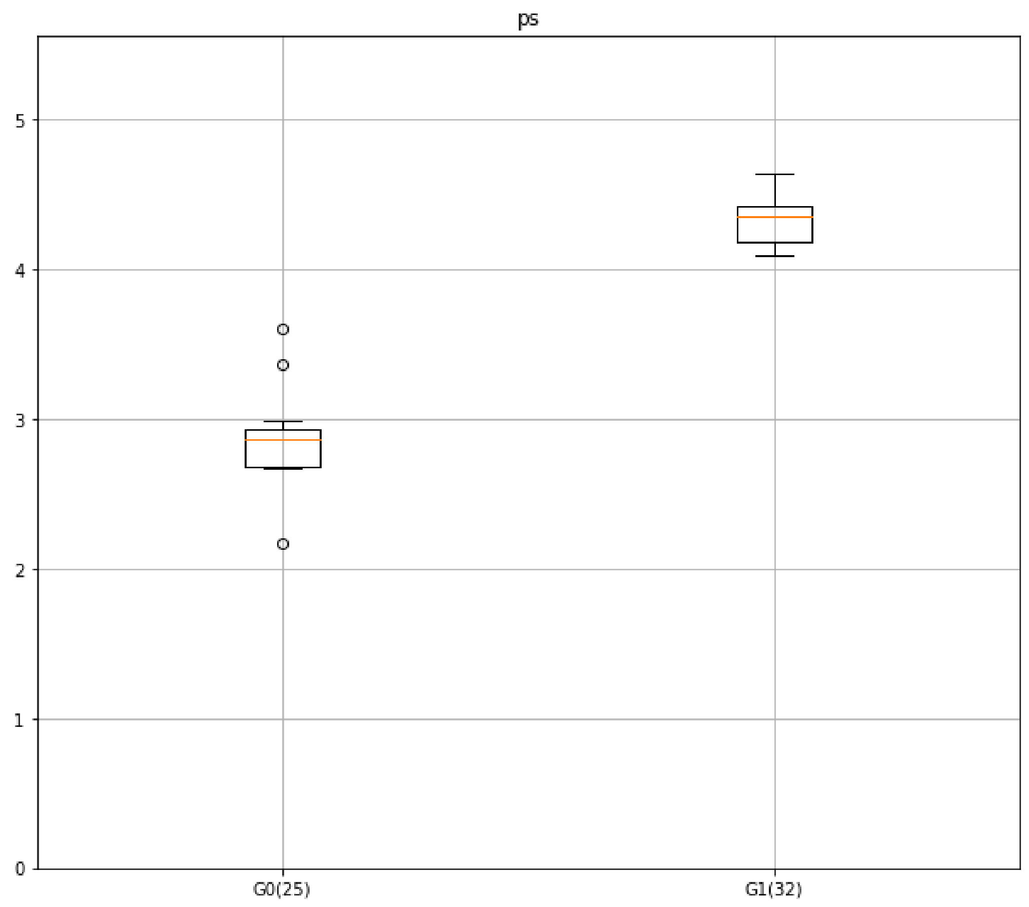
FedSize, ne_size and PS are top 3 parameters for the run time of APP1

Referenced values for correlation of parameters

Boxplot for IO_filamentation_fluid_laser_sbs

Nprocess = 1024 and FedSize = 746M

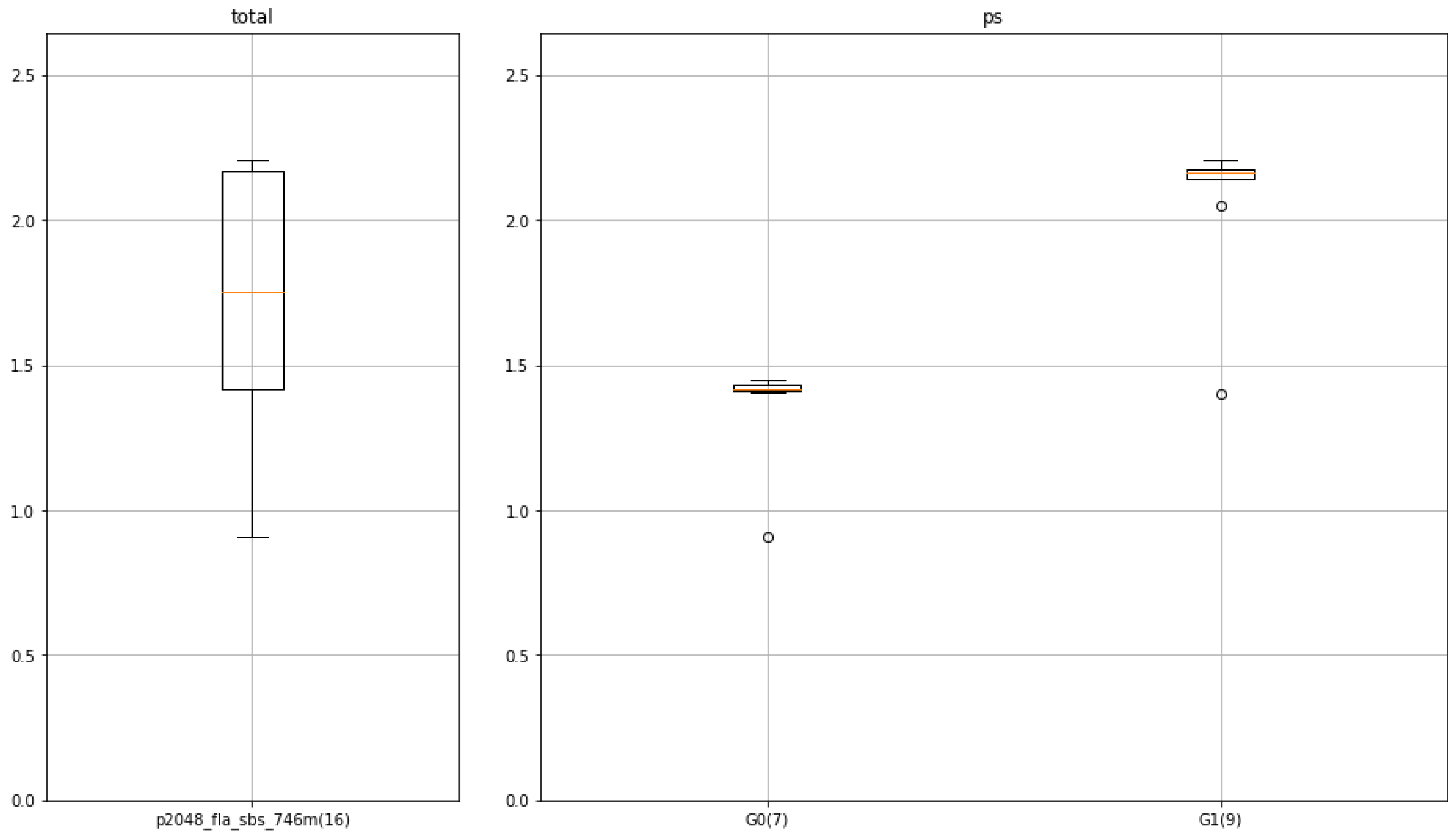




Group jobs with ps parameter high quartile and low quartile interval is changed from 1.5 seconds to 0.3 seconds

Boxplot for IO_filamentation_fluid_laser_sbs

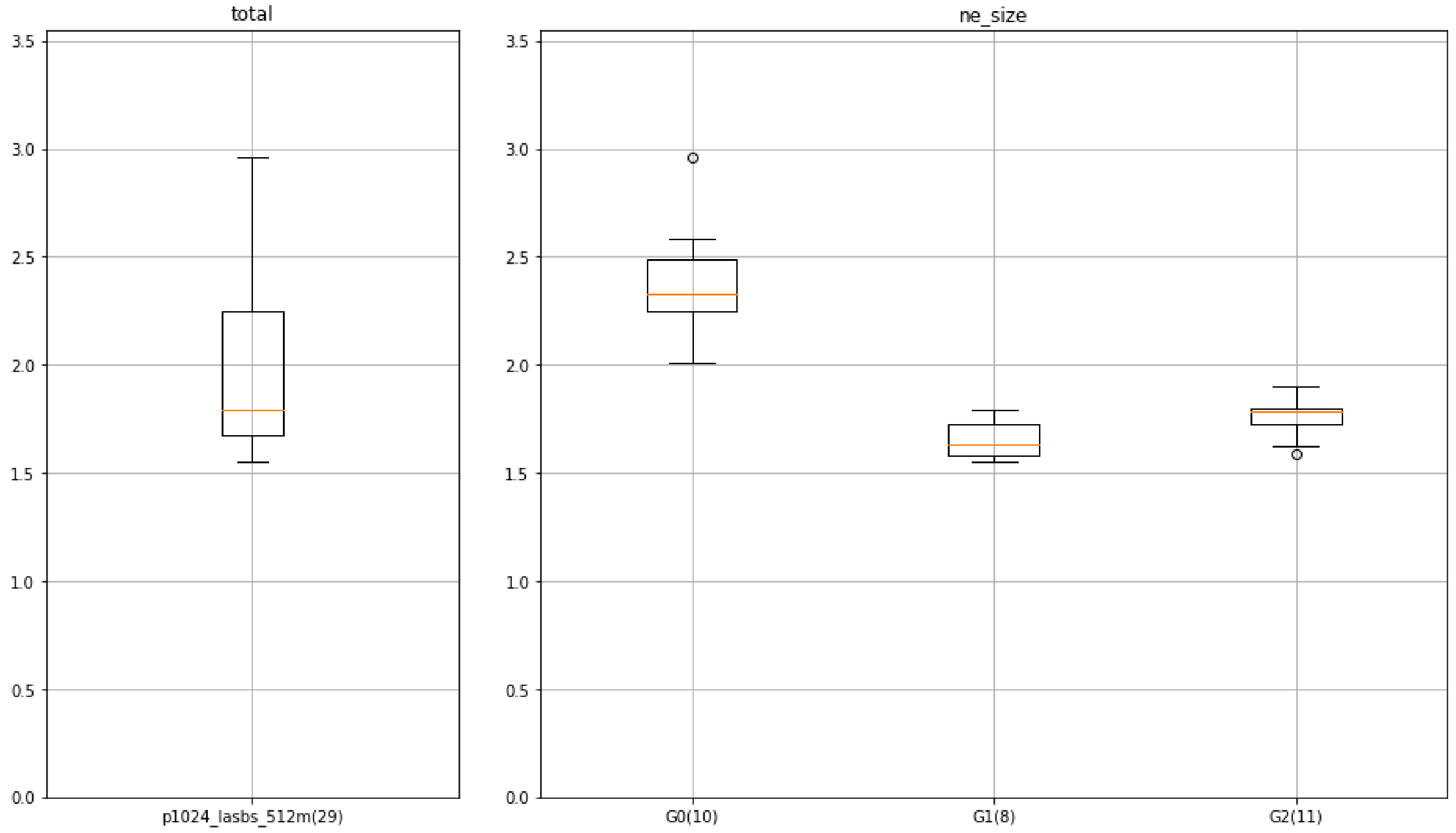
Nprocess = 2048 and FedSize = 746M



Group jobs with ps, high quartile and low quartile interval is changed from 0.8 seconds to less than 0.1 seconds

Boxplot for IO_filamentation_fluid_laser_sbs

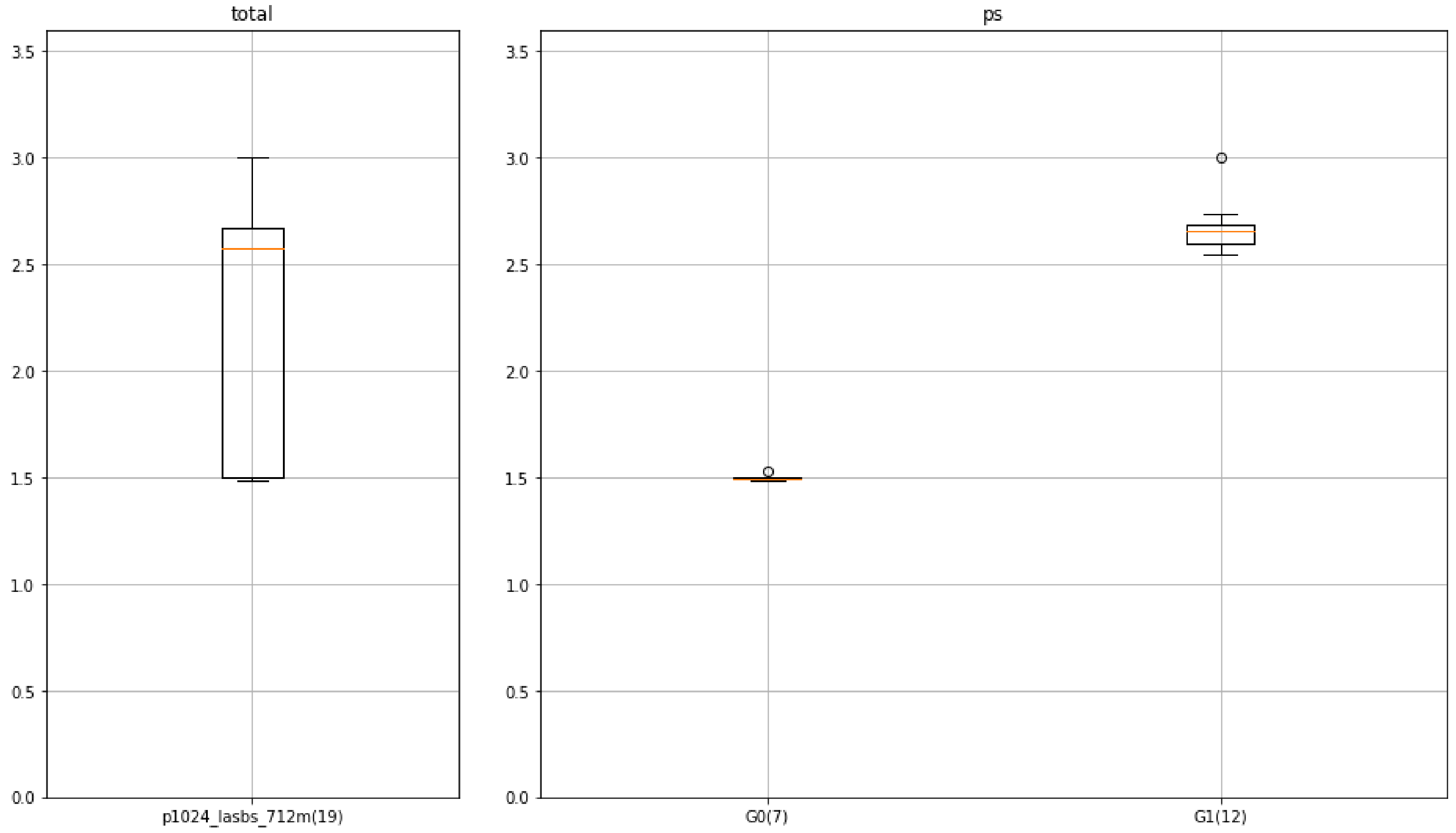
Nprocess = 2048 and FedSize = 746M



Grouped by ne_size, high quartile and low quartile interval is changed from 0.7 seconds to 0.3 seconds

Boxplot for IO_laser_sbs model

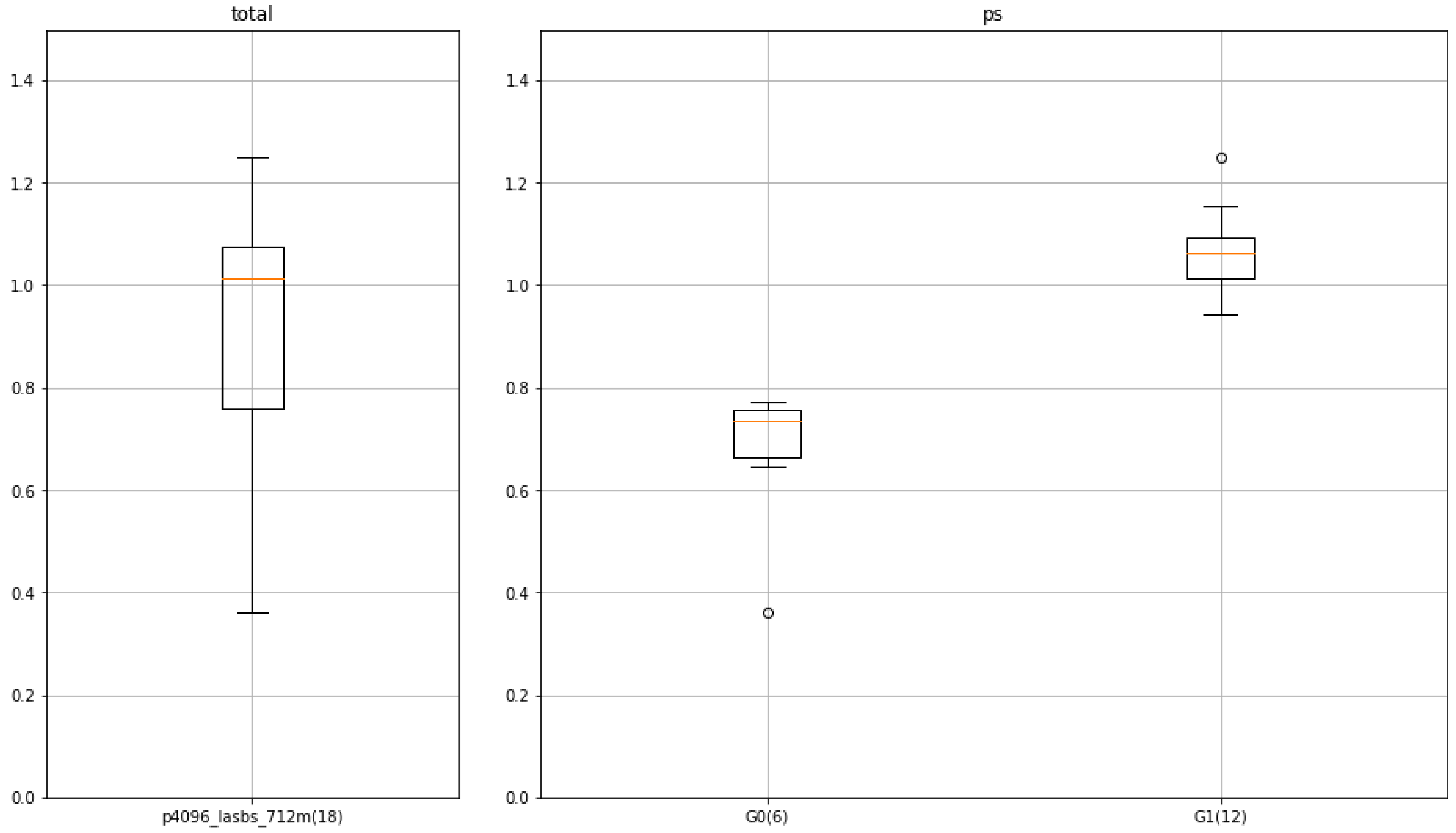
Nprocess = 1024 and FedSize = 712M



Grouped by ps, high quartile and low quartile interval is changed from 1.2 seconds to 0.1 seconds

Boxplot for IO_laser_sbs model

Nprocess = 4096 and FedSize = 712M



Grouped by ps, high quartile and low quartile is changed from 0.3 seconds to 0.1 seconds

Lasso regression on step time of APP1

Train the regressor with logs of model:

Table 1. Model overview

Model(process)	njobs
IO_filamentation_fluid_laser(2048)	10
IO_filamentation_fluid_laser_sbs(1024)	61
IO_filamentation_fluid_laser_sbs(2048)	27
IO_laser_sbs(1024)	78
IO_laser_sbs(2048)	36
IO_laser_sbs(4096)	19
Row_sum	231

Table2. weights of parameters

W_FedSize	W_ps	W_ne_size
12.352	0.557	0.000
3.855	2.593	0.000
2.054	0.430	0.000
3.924	1.826	0.710
0.277	0.000	0.659
0.000	0.710	0.000
22.462	6.118	1.369

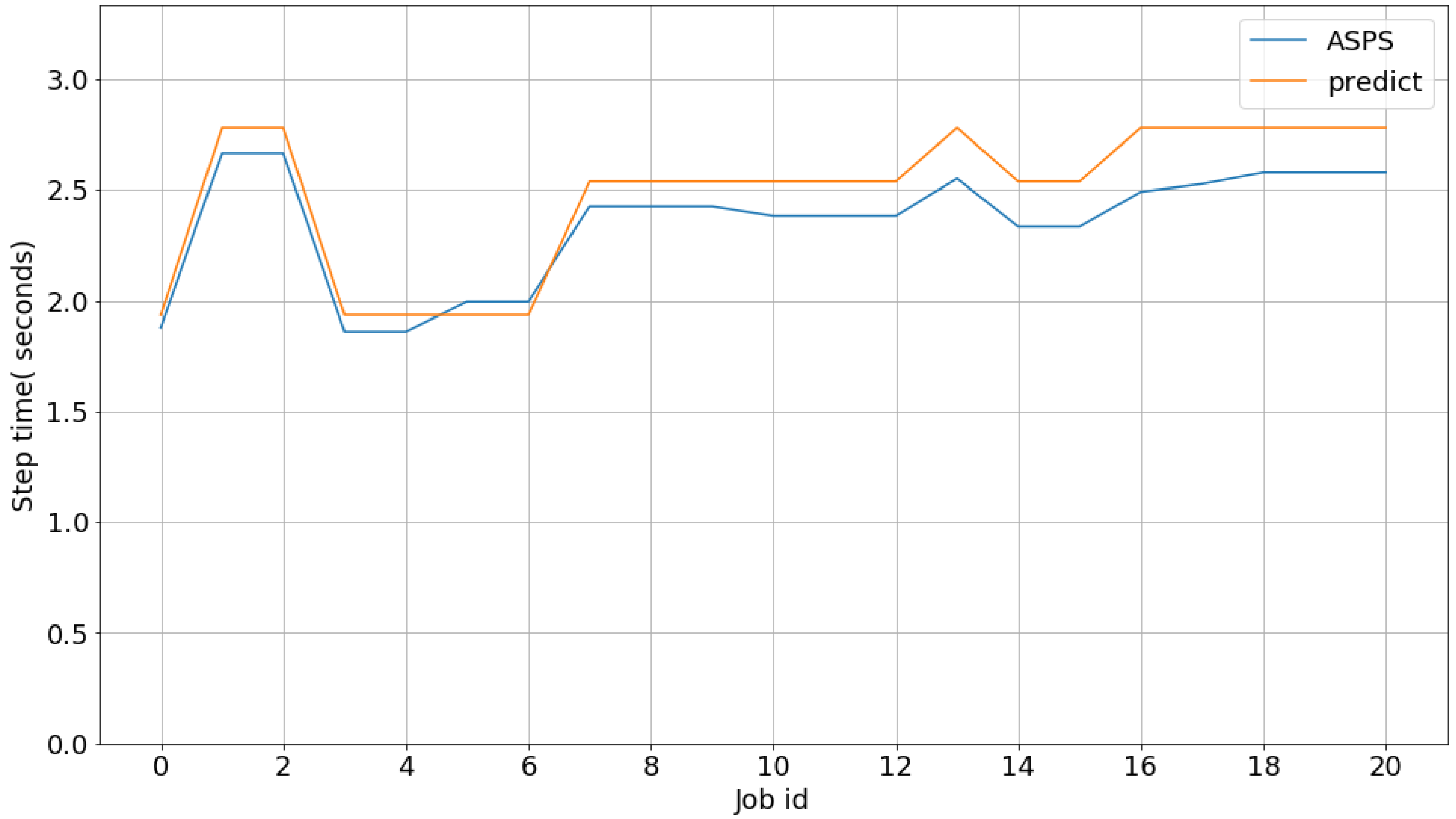
Table3.

intercept
-3.452
-0.345
0.171
-4.401
0.666
0.417

Predict time = w_Fedsize * Fedsize + W_ps * ps + W_ne_size * ne_size + intercept

Time prediction of APP1 jobs in 2019

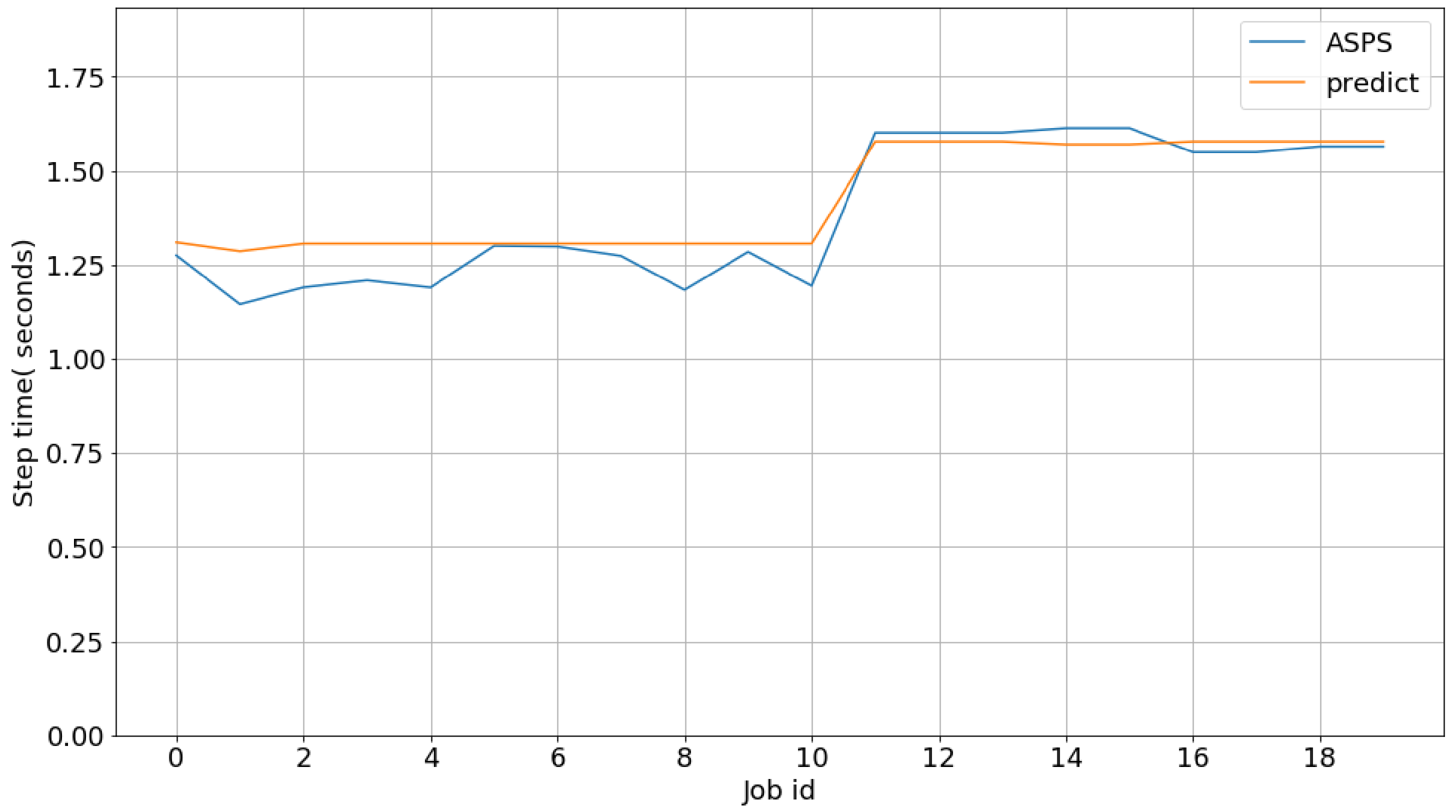
IO_filamentation_fluid_laser_sbs model, 2048 processes



Mean value of step time is 2.35. Mean value of predict is 2.49. Derivation is -0.14 seconds, which is about 6% of step time.

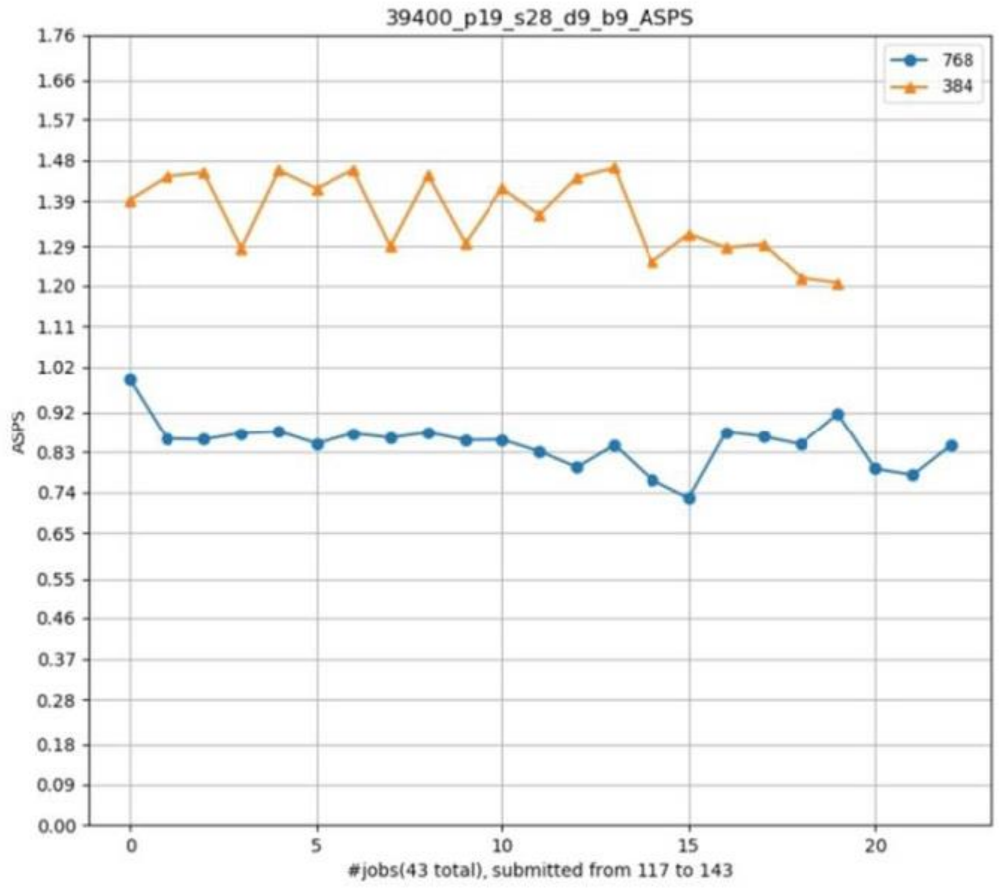
Time prediction of APP1 jobs in 2019

IO_laser_sbs model, 2048 processes



Mean value of step time is 1.39. Mean value of predict is 1.43. Derivation is -0.04 seconds, which is about 2.8% of step time.

Performance of APP2 in 384 and 768 procedures



The average step time of the 384 process is 1.32 seconds. The average step time of 768 processes is 0.83 seconds. The model has a parallel acceleration ratio of 1.59 and a parallel efficiency of 79.5% from 384 to 768 processes.

4. summary

- This paper introduces a performance analysis method directly from a large amount of logs for productive applications
  - Logs in archive are organized into a model-based tree.
- With machine learning, this paper builds a performance model for APP1.
  - Correlation analysis and lasso regression find the top significant parameters on job performance.
  - With significant parameters, it predicts step time of 2 models in APP1, derivations are 2.8% and 6%.

Thank You!