Automatic Detection of Performance Degradation

Šimon Stupinský

Brno University of Technology
Faculty of Information Techology
Department of Intelligent Systems

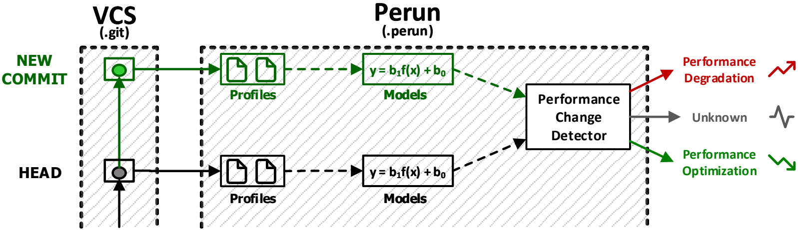
June 12, 2018

Motivation

- Manual control of performance is hard.
  - Precise preservation of the history is required.
  - One needs to maintain the context of the executed performance test.
- Version Control Systems (GIT) $\Rightarrow$ Performance Versioning Systems (Perun)
- Profiles stored in Perun could be used to automatically detect possible degradation between two points in project history.

Regression between commits

- Everything revolves around profiles
- After each version "release" (i.e. commit):
  1. Generate Performance Profiles
  2. Postprocess Profiles using Regression Analysis
  3. Store it in Perun Directory
  4. **Detect Potential Performance Degradation**



Profiles

- Format based on **JSON** structure (easy to process, well-supported, etc.)
- **Unification** for various metrics (time, memory, etc.)
- Profiles are composed of **few regions**
  - header, global, snapshots table of chunks

```
Profile = {
 'header': {
  'type': 'mixed',
  ...
 }

 'global': {
  "resources": [
   {
    "amount": 61,
    "structure-unit-size": 0,
    "subtype": "time delta",
    "type": "mixed",
    "uid": "skiplistInsert(skiplist*, int)"
   },
   {
    "amount": 13,
    "structure-unit-size": 1,
    "subtype": "time delta",
    "type": "mixed",
    "uid": "skiplistSearch(skiplist*, int)"
   },
   ...
```

Regression models

- Computed by **Regression analysis** module
- We use the method of least squares for finding the model and coefficient of determination for evaluating its fitness.
- Models are represented by two coefficients $\mathbf{b_o}$ and $\mathbf{b_1}$, e.g. $y = b_0 + b_1 * f(x))$
- Type of models, which supports Perun are:
  - constant, linear, logarithmic, quadratic, power

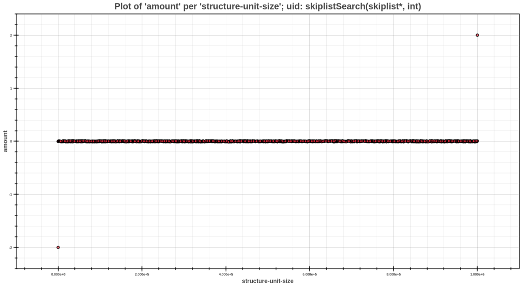
Methods of performance degradation

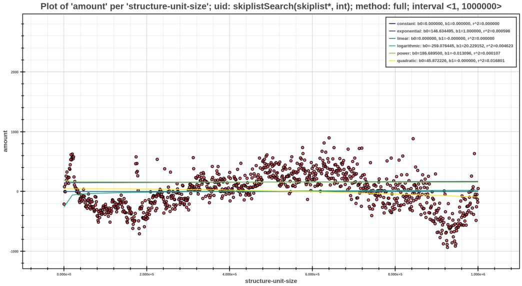
- Methods of comparing two profiles:
  1. between dependent variables themselves
  2. between the best fitting models

- Used statistical methods for performance change detection:
  1. Absolute error: $\Delta x_i = f(x_i) - f_b(x_i)$
  2. Relative error: $\delta x_i = \frac{f(x_i) - f_b(x_i)}{f_b(x_i)} = \frac{\Delta x_i}{f_b(x_i)} = \frac{f(x_i)}{f_b(x_i)} - 1$
  3. Mean squared error: $MSE = \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - f_b(x_i))^2$
  4. Studentized residual: $t_i = \frac{\Delta x_i}{\sqrt{MSE_{(i)}(1 - h_i)}}$

Experiment Evaluation: No error

- Problem with finding an appropriate threshold for detection.
- Sum of the absolute errors is close enough to a zero.





Experiment Evaluation: Constant error

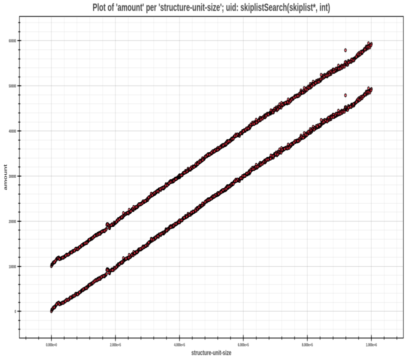


Figure: Resources of the two profiles, where the best models, are linear. The points above are resources from the target profile, which contains injected constant error.

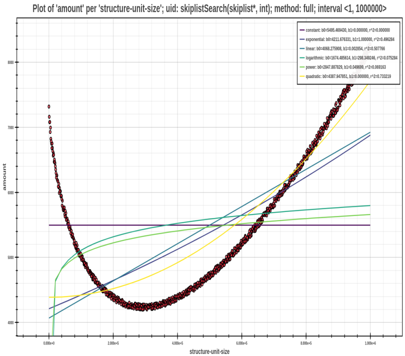
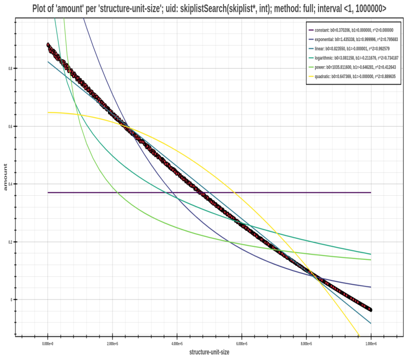


Figure: The results of the experimental detection of the constant error. The analysis was performed using the absolute error method. We can notice the values of the coefficients b_0 on all the models.

Experiment Evaluation: Constant error



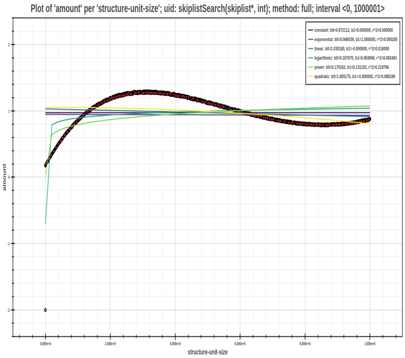


Figure: The results of the experimental detection of the constant error. The analysis was performed using the relative error method. We can see decreasing rate of degradation.

Figure: The analysis was perfomed using the studentized residuals method. We can see, that the values of coefficient b_0 in the all models and values of results resources, have approximately value of one.

Experiment Evaluation: Constant error

Type of model	**Linear**	**Quadratic**	**Logarithmic**	**Power**	**Exponential**
$\sqrt{\textbf{MSE}}$	9999.98	10003.37	10009.78	10026.27	10055.68
	4986.72	4990.37	4997.93	5495.47	4699.01

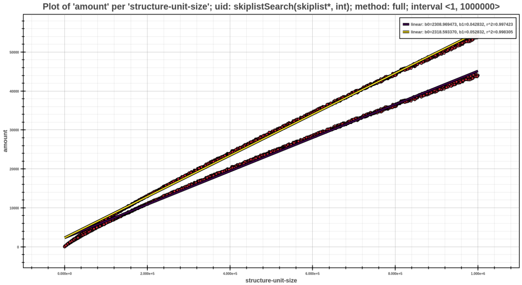
Table: The value of the MSE computed from **absolute error**. In the first attempt of the detection, it was the constant `z = 10000` and in the second attempt, `z = 5000`.

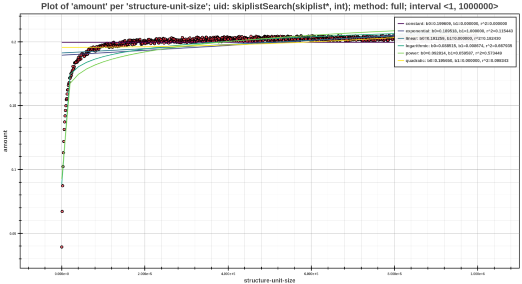
Type of model	**Linear**	**Quadratic**	**Logarithmic**	**Power**	**Exponential**
MSE	1.00145	1.00193	1.00248	1.00231	1.00181
$\mathbf{b_0}$	0.99296	0.97196	1.18174	1.20521	0.99296

Table: The value of the MSE computed from **studentized residuals**. In the third row, is shown the coefficients b_0 from the selected profile of this analysis.

Experiment Evaluation: Linear error

- Can be identified in graph by increasing value of relative error
- Experiments show change in coefficients $\mathbf{b_0}$ and $\mathbf{b_1}$
- Problems with detection for some types of models



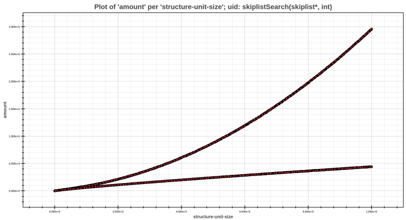


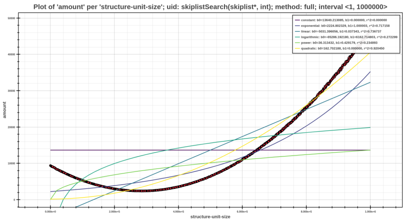
Experiment Evaluation: Linear error

		Linear	Quadratic	Logarithmic	Power	Exponential
$\mathbf{b_0}$	baseline	238.5889	-7092.9499	7942.4242	2308.9694	1838.5053
	target	248.0437	-7081.8701	7953.0452	2318.5934	1848.1188
$\mathbf{b_1}$	baseline	0.0491	0.0656	0.0336	0.0428	0.0459
	target	0.0590	0.0756	0.0436	0.0529	0.0558

Table: The value of both coefficients of linear model from the baseline and target profile. We can see, that in all types of models, values are changed by the same constant.

Experiment Evaluation: Quadratic error





	Kind of error	**Linear**	**Quadratic**	**Logarithmic**	**Power**	**Exponential**
σ	**Constant**	2892.66	2799.13	2395.15	3803.29	3983.17
	Linear	160.472	160.520	160.506	1169.87	2011.64
	Quadratic	12572.9	12088.4	13445.9	12578.8	12354.3

Table: Comparing the values of the standard deviation in the different kinds of errors. The assumption, that value will be the highest for the quadratic error, was demonstrated by experiments.

Conclusion

- Summary of Our Research
  - ✓ This work has shown, that some of the errors can be detected using explored methods.
  - × Further studies will explore more errors and those not detected by current methods
- Next Step of Our Research
  - Formalization of Proper Algorithm
  - Integration in the **Perun** Tool Workflow
  - Testing on Real World Code