Excel@FIT 2018

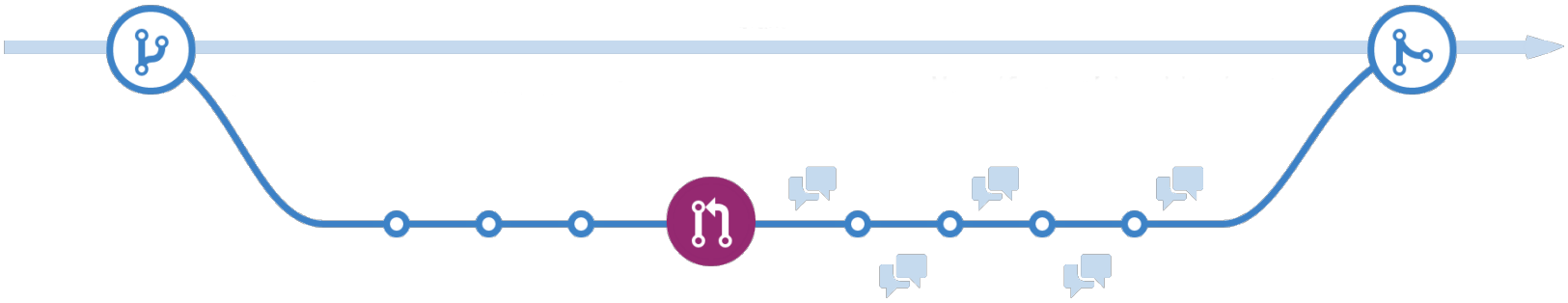
16 – Towards the detection of performance degradation

Tools for continuous performance monitoring

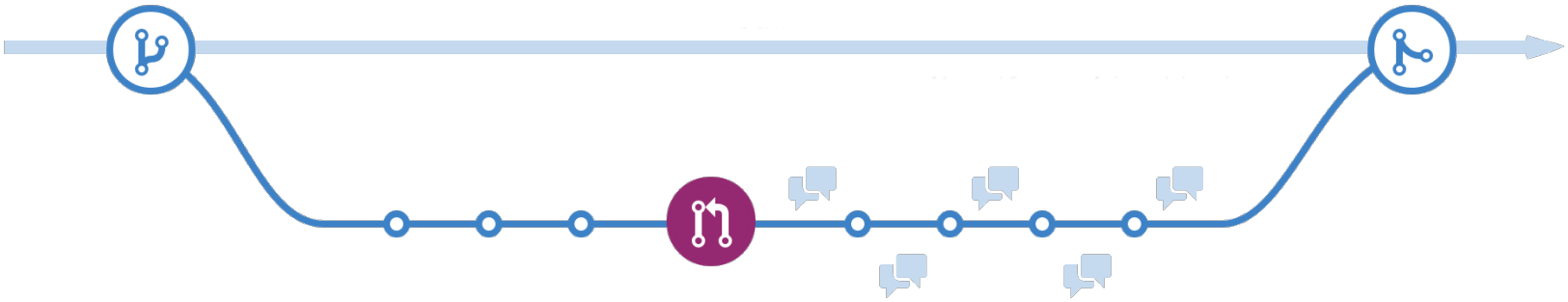
Jiří Pavela*, Šimon Stupinský**

*xpavel32@stud.fit.vutbr.cz
**xstupi00@stud.fit.vutbr.cz

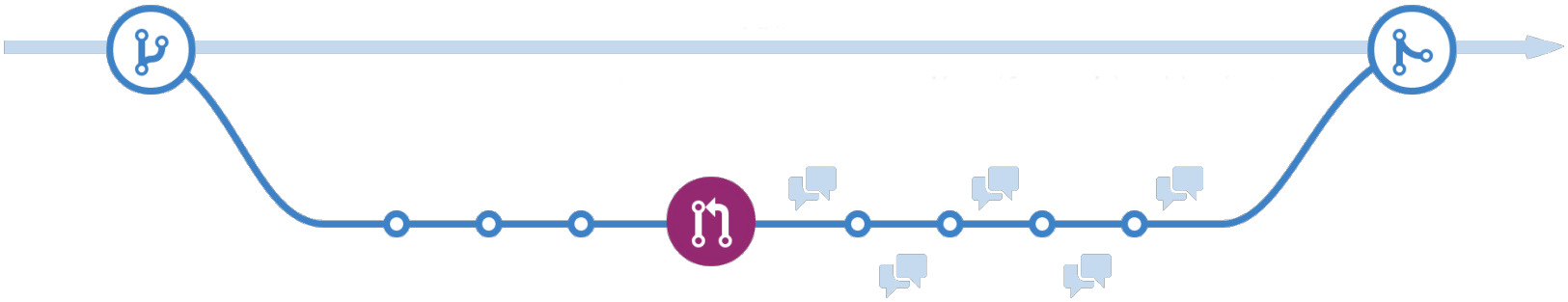
Brno University of Technology, Faculty of Information Technology



- Automatic regression testing with *Continuous Integration (CI)* tools is currently well-established practice in industry.

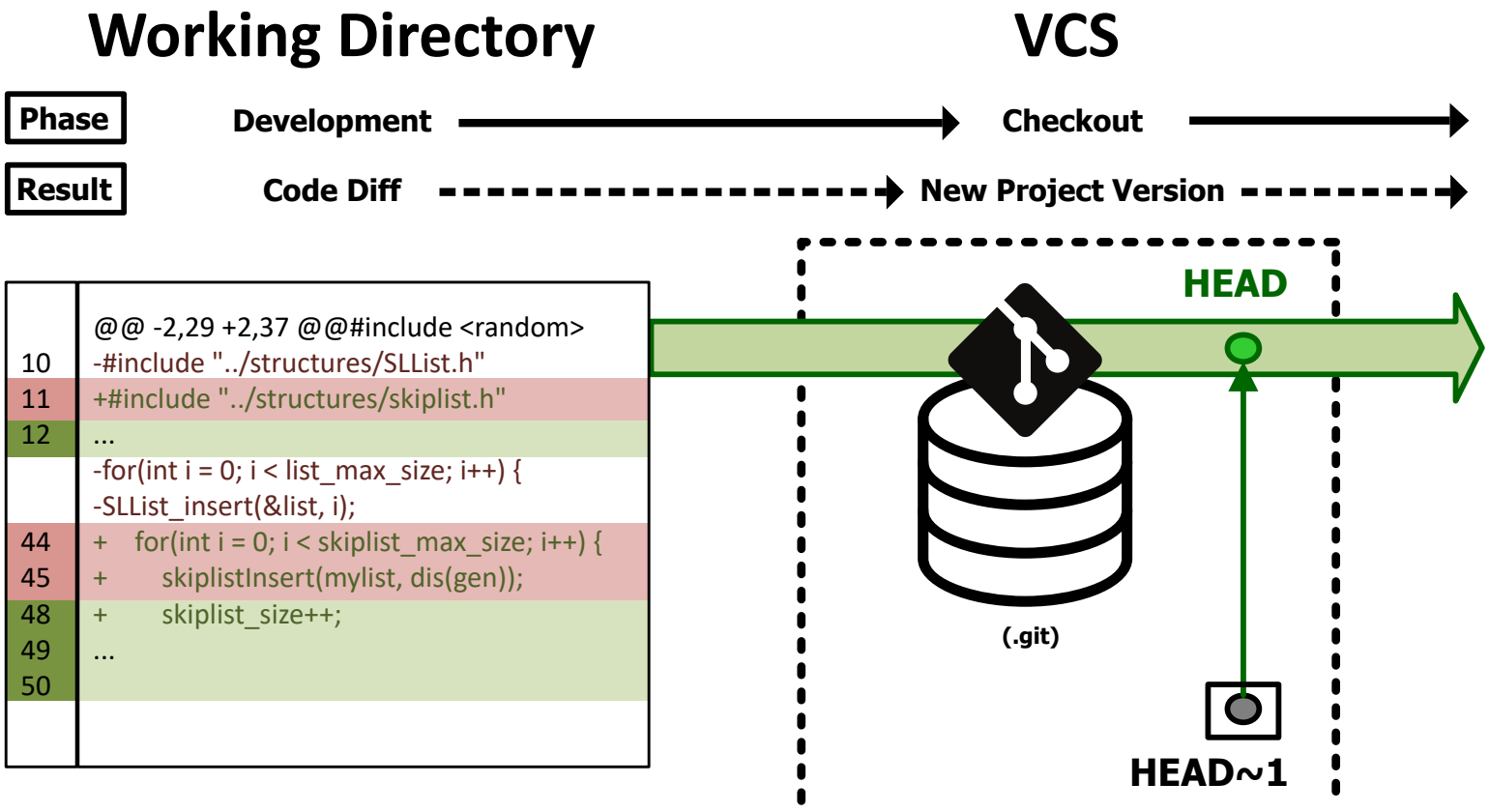


- Automatic regression testing with *Continuous Integration (CI)* tools is currently well-established practice in industry.
- Such tests reveal many bugs during development process.
  - **The bugfix ratio is quite high.**



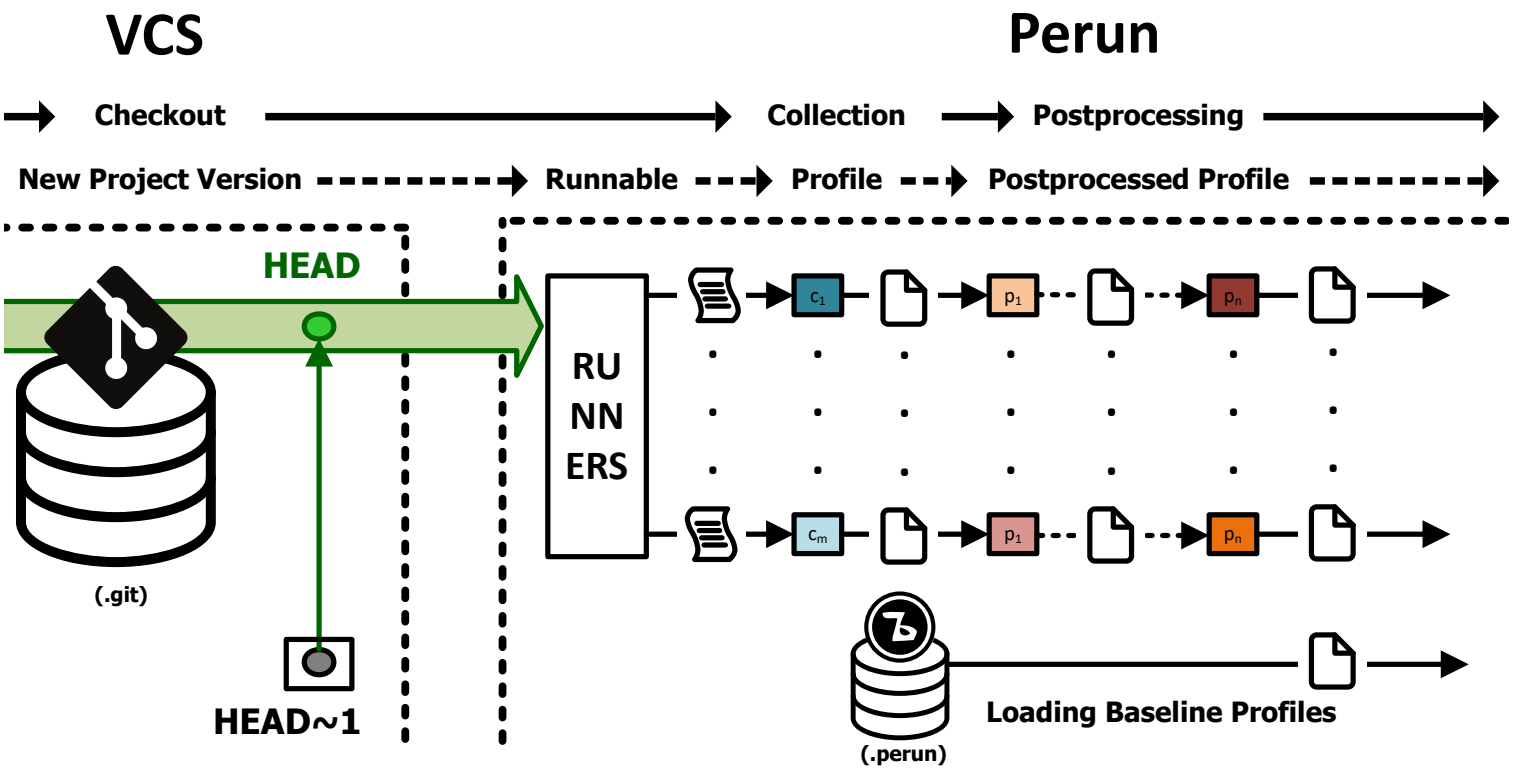
- Automatic regression testing with *Continuous Integration (CI)* tools is currently well-established practice in industry.
- Such tests reveal many bugs during development process.
  - **The bugfix ratio is quite high.**
- But what about performance bugs?
  - The CI does not really support thorough performance tests.
  - We aim at creating a framework specifically designed for **continuous performance monitoring**.

Project repository



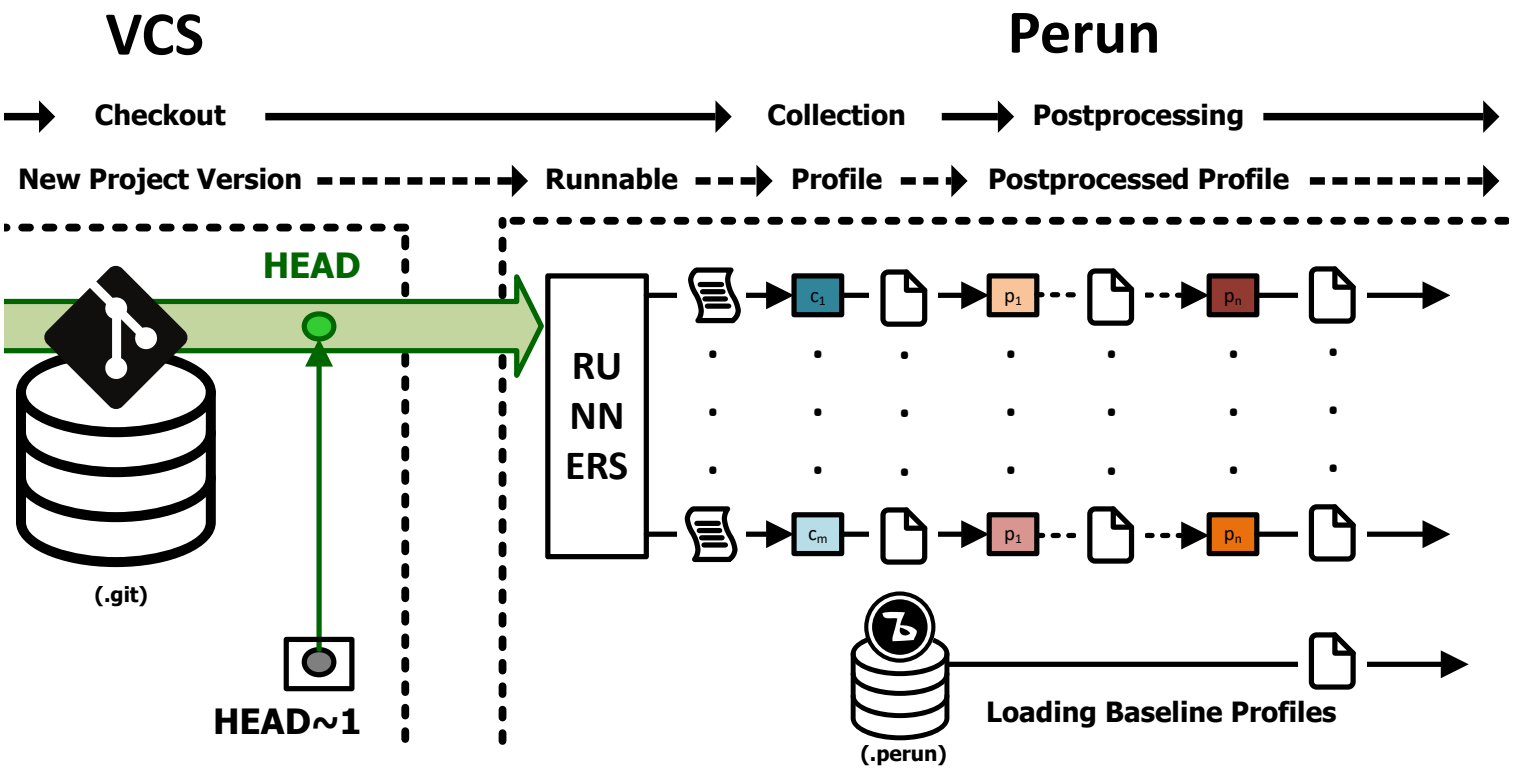
1. User makes changes to working directory and commits them.
   - This triggers performance monitoring action in Perun.

Data collection and postprocessing



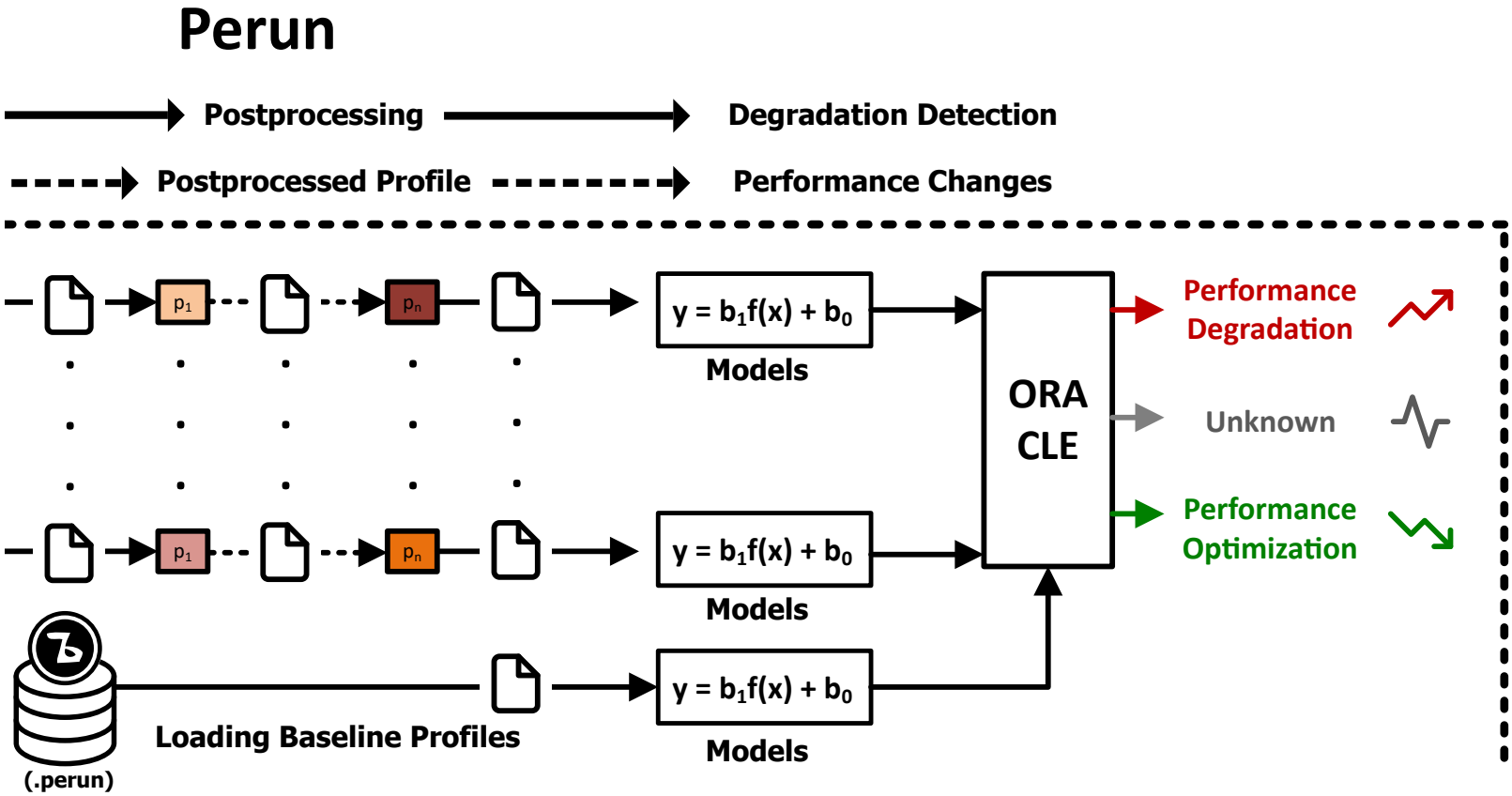
2. Performance data are collected using profilers (collectors)
   - **trace collector**, **memory collector**, etc..

Data collection and postprocessing



2. Performance data are collected using profilers (collectors)
   - **trace collector**, **memory collector**, etc..
3. Collected data are further postprocessed.
   - **clusterization**, **regression analysis**, etc..

The degradation detection



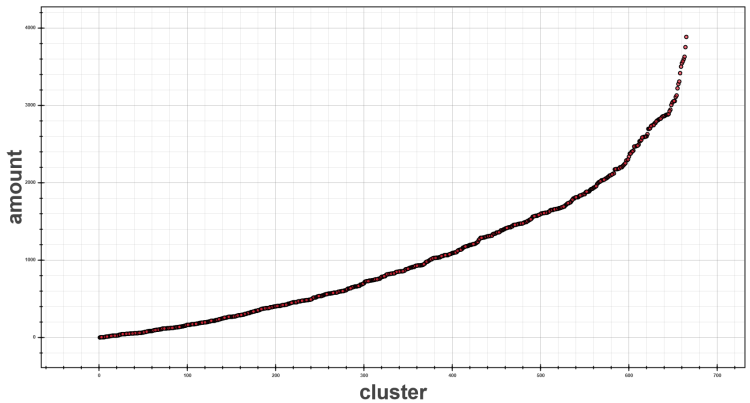
4. Older stored profiles (baseline profiles) are compared with newly generated profiles (target profiles).
   - **Detection** and **classification** of performance changes.

Data collection and postprocessing

Trace collector:

- Dynamic executable *tracing* and *profiling*.
- Based on the SystemTap tool.

Data collection and postprocessing



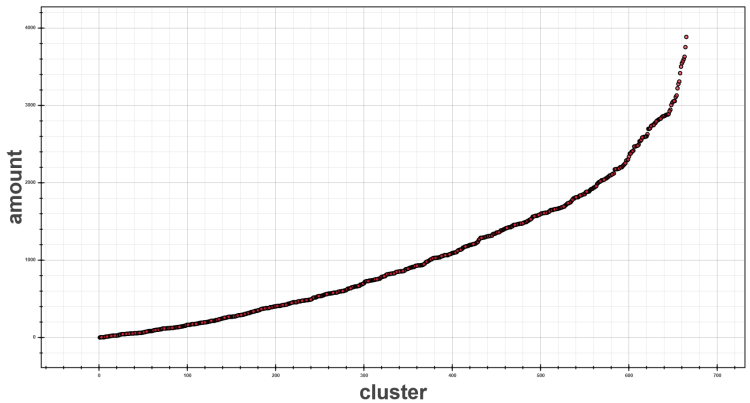
Trace collector:

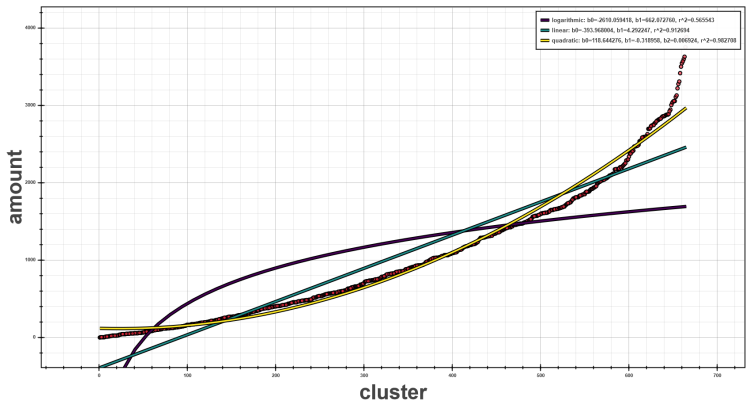
- Dynamic executable *tracing* and *profiling*.
- Based on the SystemTap tool.

Postprocessing:

- Clusterization
  - Estimates independent **variables**.
  - e.g. **input data size**

Data collection and postprocessing





Trace collector:

- Dynamic executable *tracing* and *profiling*.
- Based on the SystemTap tool.

Postprocessing:

- Clusterization
  - Estimates independent **variables**.
  - e.g. **input data size**
- Regression analysis
  - Models function **performance**.

Detection and classification

- Methods use baseline and target models of form:

$$m_{baseline} = b_{0\,baseline} + b_{1\,baseline} \cdot f(x)$$

$$m_{target} = b_{0\,target} + b_{1\,target} \cdot f(x)$$

- We subtract both models and then estimate the change model:

$$m_{change} = m_{baseline} - m_{target}$$

Detection and classification

- Methods use baseline and target models of form:

$$m_{baseline} = b_{0\,baseline} + b_{1\,baseline} \cdot f(x)$$

$$m_{target} = b_{0\,target} + b_{1\,target} \cdot f(x)$$

- We subtract both models and then estimate the change model:

$$m_{change} = m_{baseline} - m_{target}$$

1. Method based on linear models.
   - Classifies change based on linear model coefficients
   - Estimates change (m_{change}) using **intercept** (b_0) and **slope** (b_1) of **linear** models.

Detection and classification

- Methods use baseline and target models of form:

$$m_{baseline} = b_{0\,baseline} + b_{1\,baseline} \cdot f(x)$$

$$m_{target} = b_{0\,target} + b_{1\,target} \cdot f(x)$$

- We subtract both models and then estimate the change model:

$$m_{change} = m_{baseline} - m_{target}$$

1. Method based on linear models.
   - Classifies change based on linear model coefficients
   - Estimates change (m_{change}) using **intercept** (b_0) and **slope** (b_1) of **linear** models.
2. Method based on polynomial models
   - Classifies change based on best model coefficients
   - Estimates change (m_{change}) using **polynomial** regression analysis

Achieved results

		$\pm c$	$\pm n$	$\pm n^2$	overall	time [s]
Method 1	*detect*	80%	80%	100%	**86,67%**	5.26
	classify	25%	55%	65%	**48,33%**	
Method 2	*detect*	80%	100%	100%	**93,33%**	4.32
	classify	80%	80%	0%	**53,33%**	

- Set of artificial examples based on selected types of models.
  - 36 different pair of profiles
  - **constant**, **linear**, **logarithmic**, **quadratic**, **exponential** and **power** models
- Includes change
  - Detection
    - **constant change** (c), **linear change** (n), **quadratic change** (n^2)
  - Classification
    - **optimization** (-), **degradation** (+)

We would like to focus on the following:

- Fully integrate the detection and classification modules.
  - Currently being developed in the experimental branch.
- Improve the detection and classification accuracy.
- More precise localization of performance bugs.
- Do some experiments on real projects and bugs.

Perun repository: `https://github.com/tfiedor/perun`