Perun: Performance Under Control

Collaboration between Red Hat, FIT BUT, FI MUNI

T. Fiedor, J. Pavela, **T. Vojnar**, and many others

Motivation

- Energy savings are nowadays much welcome, especially:
  - in cloud applications run many times, expensive AI, supercomputing, …, or
  - battery-powered devices.

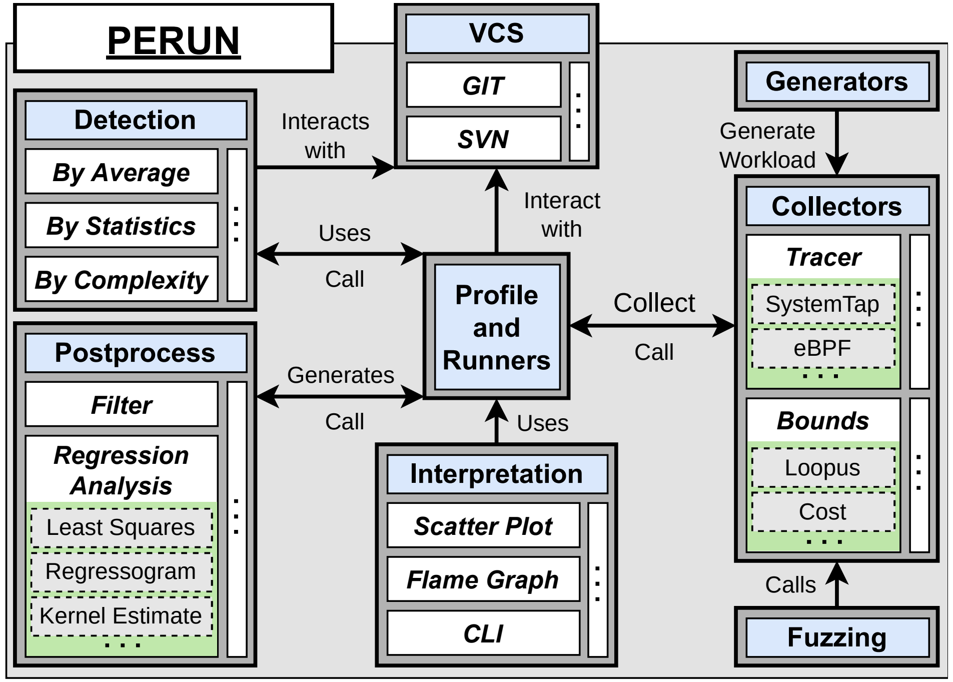
- Slow applications can disappoint customers.

- Performance bugs – e.g., “accidentally quadratic” – can also kill a system completely:
  - Apache Spark: an internal check for uniqueness
    → hanging effectively forever for a large job batch.
  - StackOverflow: A regular expression for stripping whitespaces
    → a 34 minutes long outage.
  - Chrome: one the parsers
    → a noticeable slowdown for long lines.

Overview of Perun

A complex solution for ***software performance analysis and testing***:

- Collects/imports performance data.
  - eBPF, SystemTap, static analysis, GNU perf, …
  - Various optimizations of the collection process.
- Integrates version control systems.
  - Maintains links of data to project versions.
- Creates performance models.
  - Constant c, linear a.n+b, ...
- Detects performance changes.
  - Degradations, optimizations.
- Supports performance fuzzing.
  - Generation of performance stressing inputs.
- Visualizes performance and its changes.



Performance Models in Perun

Mathematical functions of the input size

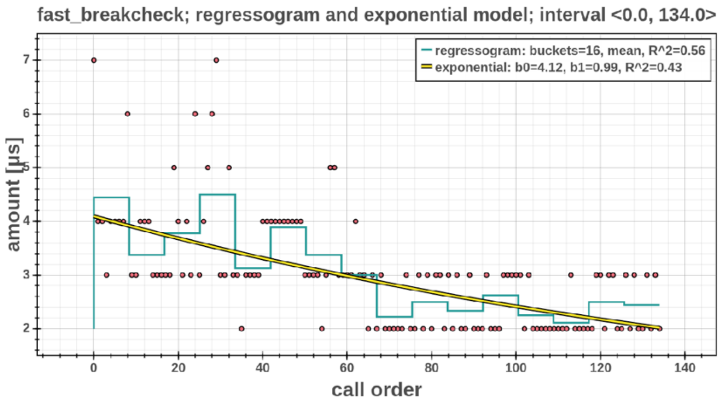
($a.n + b$, $a.n^2 + b.n + c$, …)

or

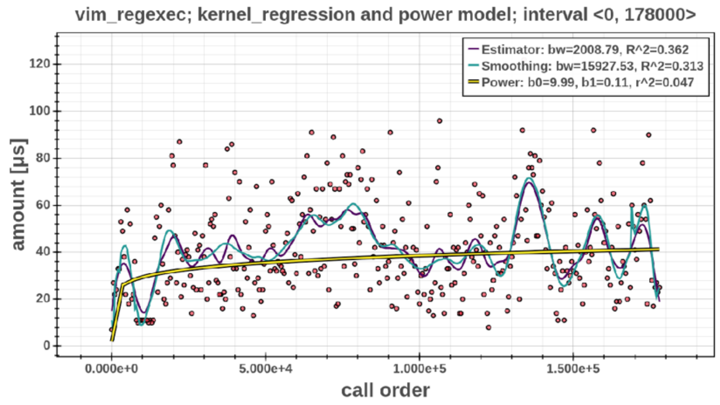
statistical summaries

(*average*, *median*, …)

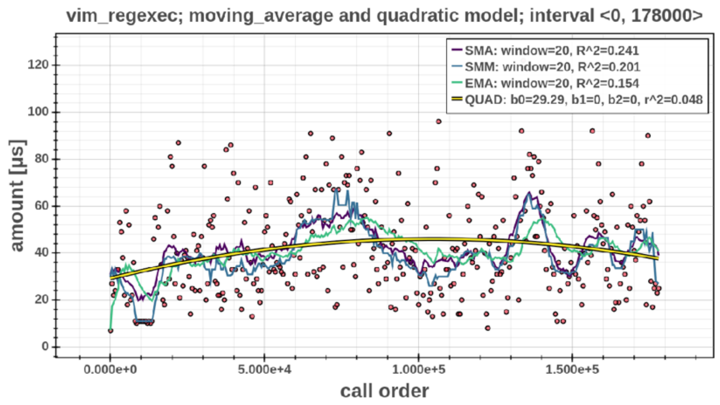
describing the main features of the profile.



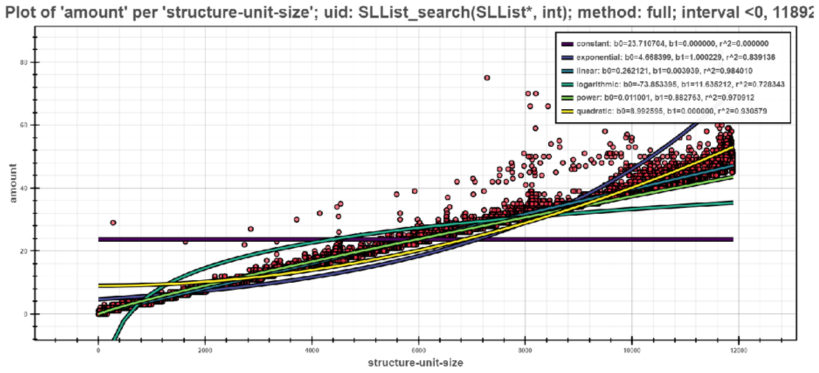
Regressogram



Kernel Regression



Moving Average



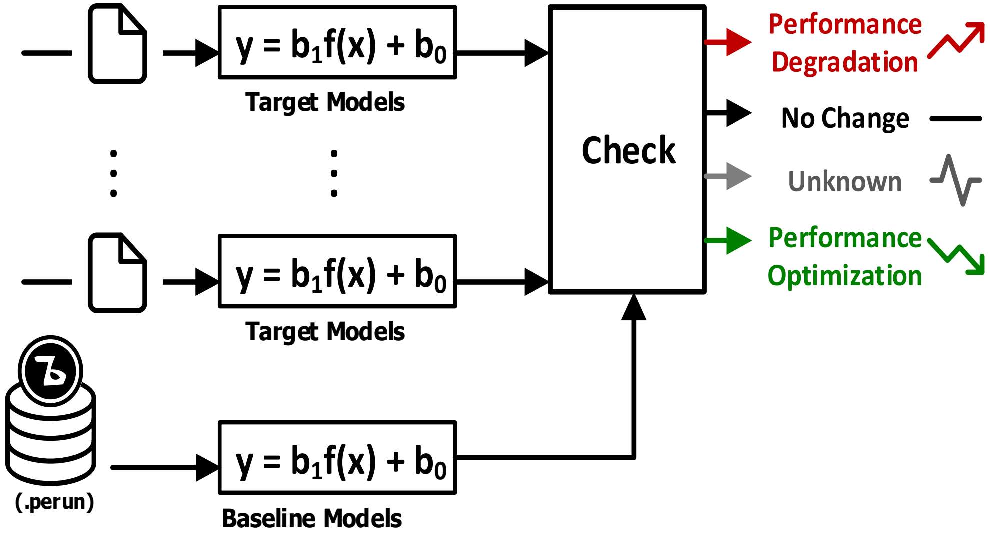
Regression Analysis

MUNI FI

Perun: Detection of Performance Changes

Multiple ***algorithms for detecting changes in the performance*** of program functions or entire programs are implemented in Perun:

- best model order equality,
- integral comparison,
- ...,
- exclusive-time outliers
  - several statistical methods for detecting changes of different severity.

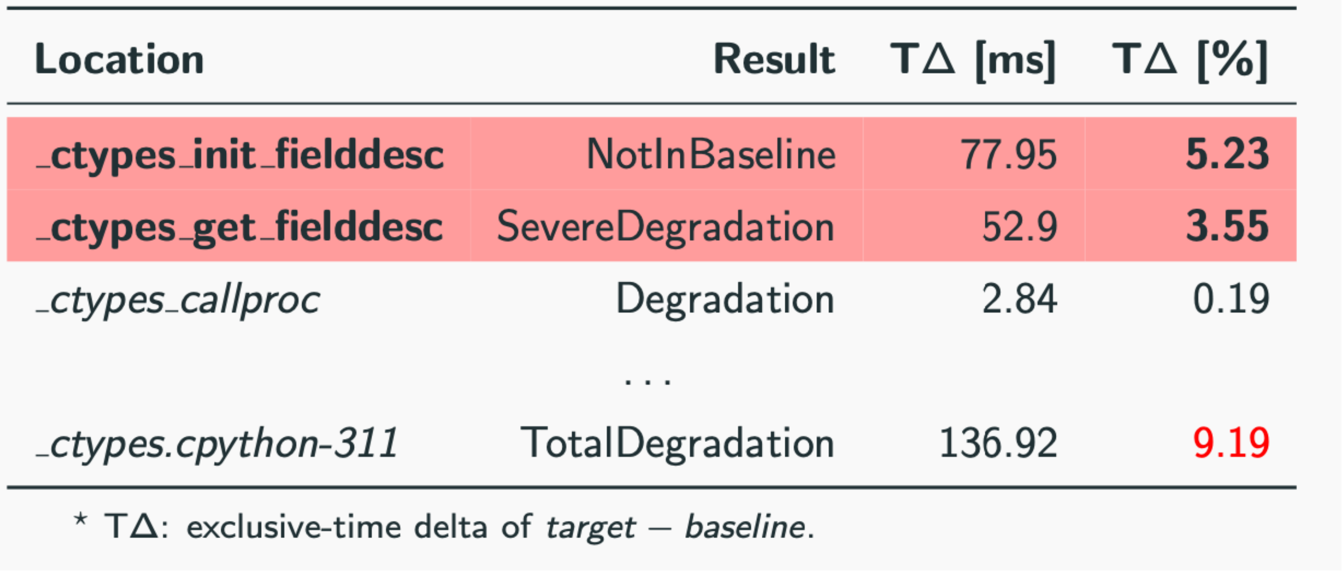


Can be done on models or also raw profiles.

Perun: Example of Degradation Detection

- CPython: Reference C implementation of a Python interpreter.
- Issue #923564: A performance regression in ctypes module:

  ≈ 8% higher function call overhead (py3.11.0a7 vs. py3.10.4).
- Detection in Perun:



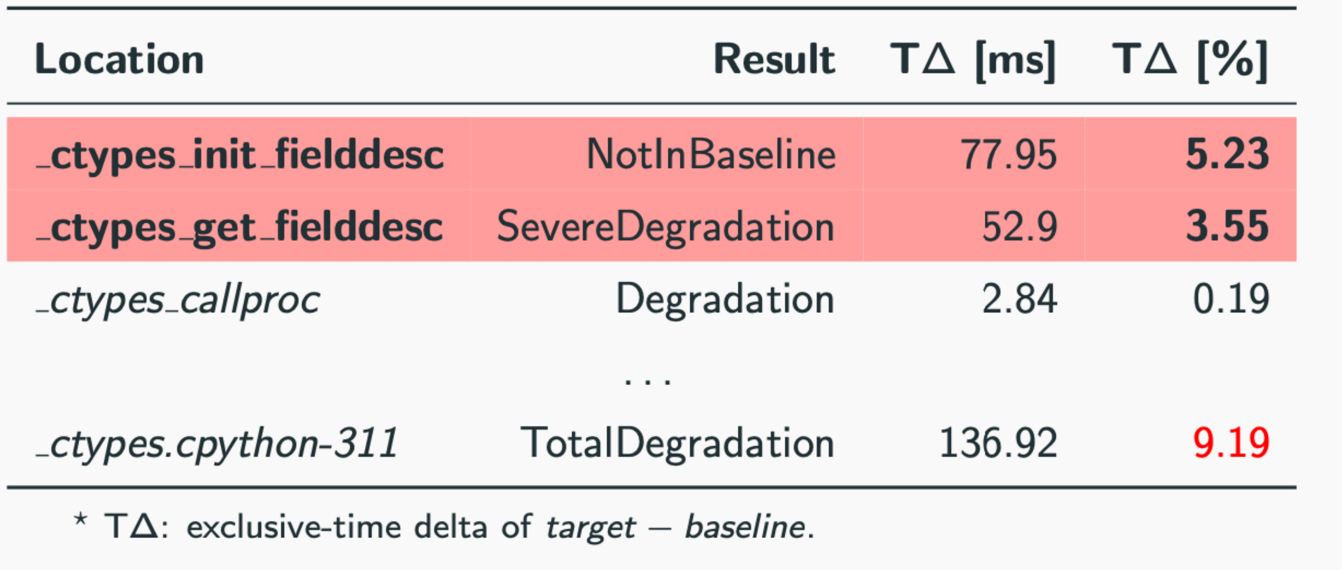
Location	Result	TΔ [ms]	TΔ [%]
_ctypes_init_fielddesc	NotInBaseline	77.95	**5.23**
_ctypes_get_fielddesc	SevereDegradation	52.9	**3.55**
_ctypes_callproc	Degradation	2.84	0.19
	...		
_ctypes.cpython-311	TotalDegradation	136.92	9.19

* TΔ: exclusive-time delta of *target* − *baseline*.

Perun: Example of Degradation Detection

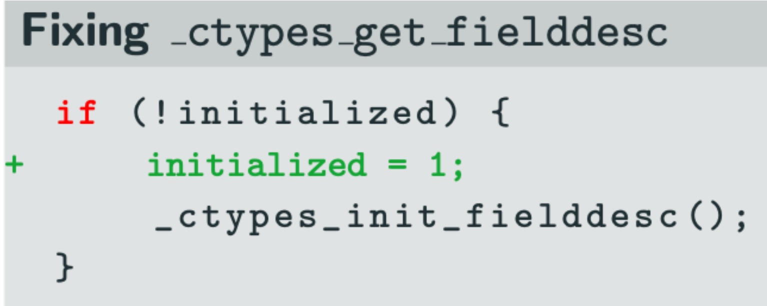
- CPython: Reference C implementation of a Python interpreter.
- Issue #923564: A performance regression in the ctypes module:

  ≈ 8% higher function call overhead (py3.11.0a7 vs. py3.10.4).
- Detection in Perun:



Location	Result	TΔ [ms]	TΔ [%]
_ctypes_init_fielddesc	NotInBaseline	77.95	**5.23**
_ctypes_get_fielddesc	SevereDegradation	52.9	**3.55**
_ctypes_callproc	Degradation	2.84	0.19
	. . .		
_ctypes.cpython-311	TotalDegradation	136.92	9.19

* TΔ: exclusive-time delta of *target* − *baseline*.

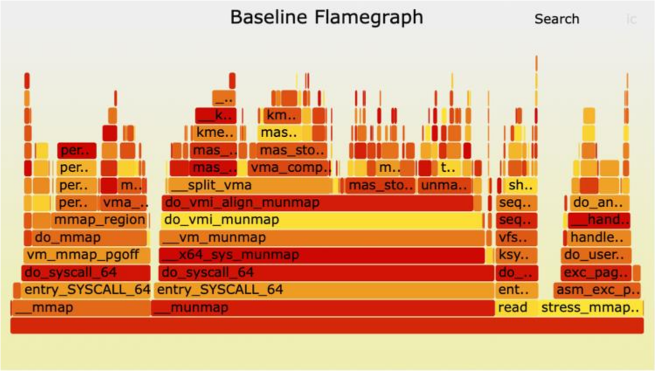


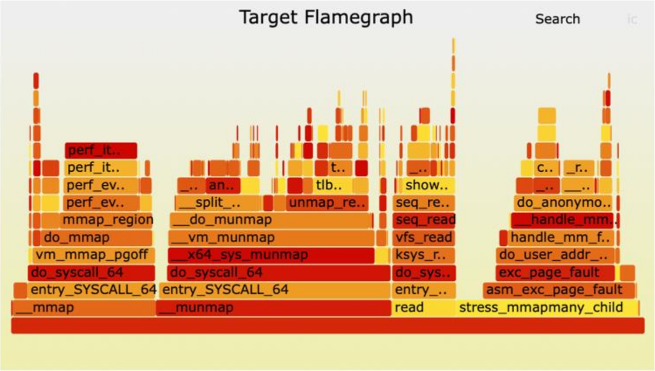
Fixing `_ctypes_get_fielddesc`

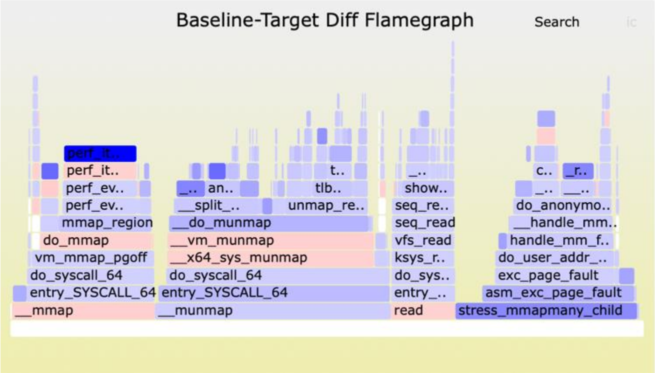
```
  if (!initialized) {
+      initialized = 1;
       _ctypes_init_fielddesc();
  }
```

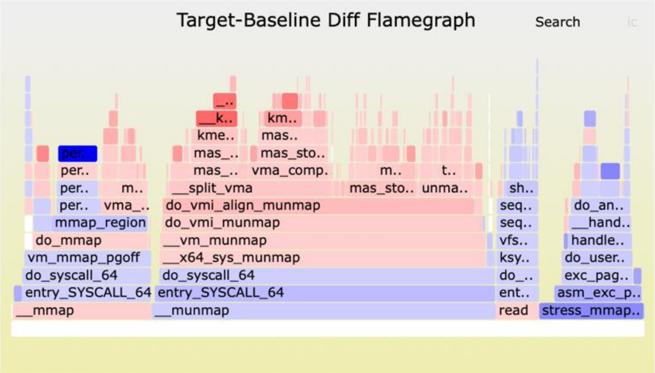
Perun: Visualisation

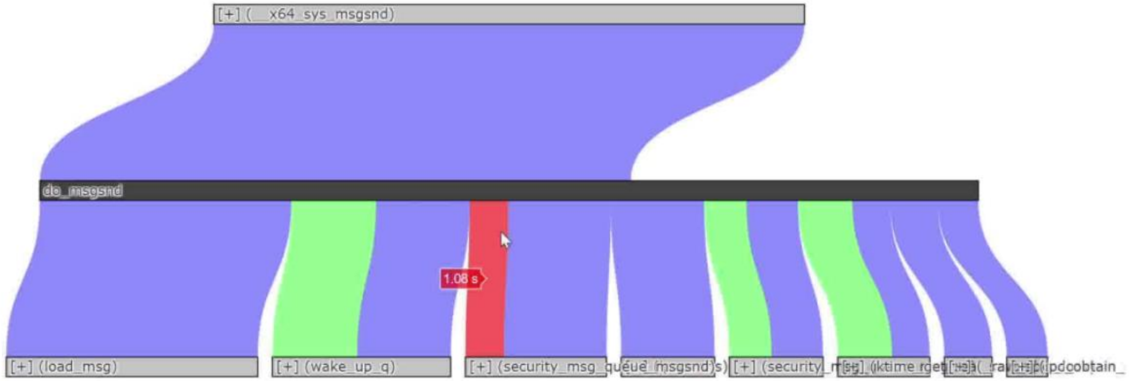
A rich set of various visualisations of performance data.

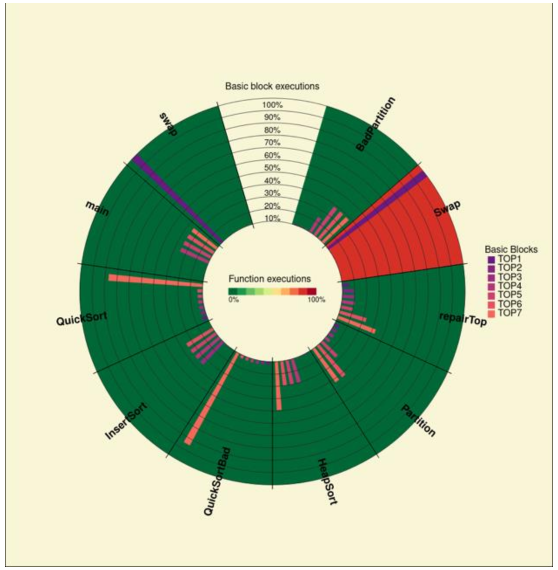












Perun: Summary of Results

Academia

- Cooperation: Red Hat, FIT BUT, FI MUNI, recently also interest from TU Graz.
  - ***From academia to industry and now back again***.
- Papers: 1 published tool paper (ICSME'22, CORE A), 1 accepted paper pending publication, 1 paper in preparation.
- Talks: DevConf'24, CHESS project'23, RH Research Days'20 and '24, RH PerfConf'23, etc.
- Students: ***15+ BSc and master theses*** extending Perun, ***2 supported PhD students***.
- Platform: further research, trying out new ideas, experiments.

Industry

- Perun integrated into the Red Hat Kernel Performance Engineering Team analysis toolchain and CI.
- Significant time savings, ranging from 1.5 hours up to 1 man-day, on performance drops.
- E.g.: excessive calls to XFS file system functions, needless calls to SELinux policy functions, …