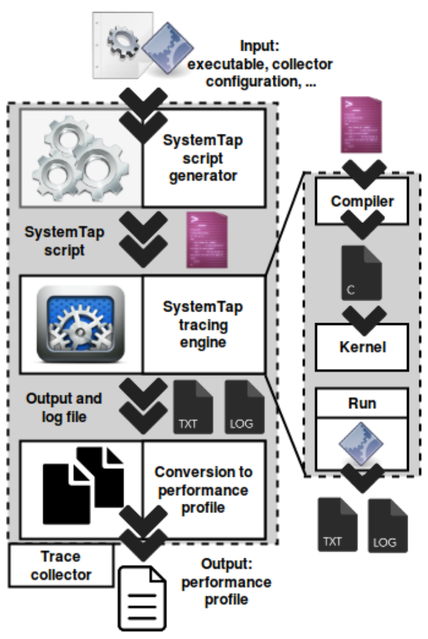
16 Towards the detection of performance degradation

Jiří Pavela, Šimon Stupinský
xpavel32@stud.fit.vutbr.cz, xstupi00@stud.fit.vutbr.cz

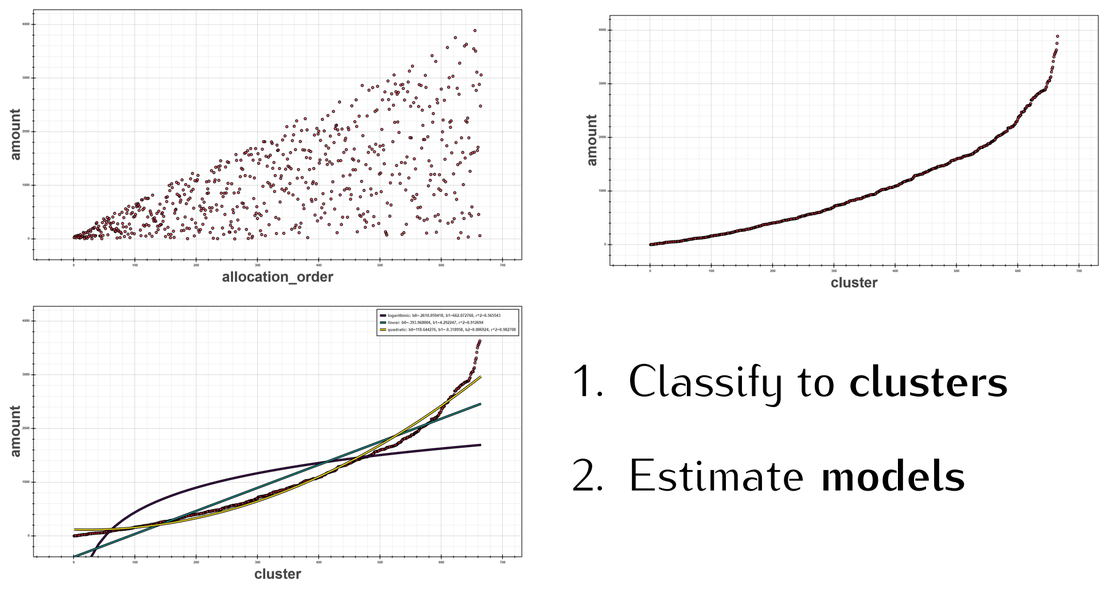
Methodology of Detecting Performance changes between Project Versions

Step 1 Data collection



- Based on **SystemTap**
- **Dynamic tracing**
- **No annotation** needed
- Collects function **times**

Step 2 Postprocessing



1. Classify to **clusters**
2. Estimate **models**

Step 3 Difference Analysis

Method check ($\mathcal{M}_b$, $\mathcal{M}_t$)
  foreach (m_b, m_t), (m_b^n, m_t^n) $\in \mathcal{M}_b$, $\mathcal{M}_t$ **do**
    $\mathcal{M} = \{m_b, m_t, m_b^n, m_t^n\}$
    $dp_{t-b} = m_t - m_b$
    $dp_{t-b}^n = m_b^n - m_t^n$
    $\varepsilon_{rel} = \sum dp_{t-b} \;/\; |dp_{t-b}|$
    $\mathcal{M}$.add(FindPoly (dp_{t-b}), 1)
    for $deg = 0$ *to* MAX_DEG **do**
      ρ = FindPoly (dp_{t-b}^n, deg)
      if $\rho[residual] \leq \xi$ **then**
        break
      $\mathcal{M}$.add(ρ)
    Classify ($\mathcal{M}$, ε_{rel})

Based on:

1. **coefficients**
2. **residuals**

Uses:

- **models**
- **data points**

Step 4 Evaluation

#2 \ #1	+c	-c	+n	-n	$+n^2$	$-n^2$
cst	OK / OK	n / OK	n^2 / c	OK / c	OK / n	n / c
lin	OK / OK	n / OK	c / OK	OK / OK	OK / n	n / n
log	NO / NO	NO / NO	OK / OK	OK / OK	OK / n	OK / n
quad	OK / OK	n / OK	n^2 / OK	OK / OK	OK / n	n / n
exp	n / OK	n / OK	OK / OK	OK / OK	OK / n	n / n
pow	n / OK	n / OK	NO / OK	NO / OK	OK / n	OK / n

Includes error:

- **Detection**
- **Classification**

Detection rate:

- **90%**

Classification rate:

- **50%**

Acknowledgements

The work is supported by various projects, groups and companies including:

- The H2020 ECSEL project **AQUAS**: Aggregated Quality Assurance for Systems
- **RedHat** and the **VeriFIT** group (BUT FIT)