Automated Performance Testing and Tuning

December 2\textsuperscript{nd}, 2021

\textbf{Presented by}

Jerry Watkins

Contributors: Max Carlson, Carolyn Kao, Kyle Shan, Irina Tezaur

Trilinos User-Developer Group Meeting

SAND2021-14919 PE
Outline

1) Automated Performance Testing
2) Automated Performance Tuning
3) Conclusions/Discussion
Automated Performance Testing
Motivation – Automated performance testing

1) **Maintaining performance and portability** in the presence of active development
   - Code changes can cause performance regressions (compiler/algorithmic optimizations)
   - Compiler/TPL changes can cause performance regressions (updates)
   - Architecture changes can cause performance regressions (CPU->GPU)

2) **Improving performance and portability** in the presence of active development
   - Performance can vary greatly with code changes (robustness)
   - Performance can vary greatly between compiler, architecture (CPU/GPU)
   - Performance can vary greatly between executions (noise)

3) **Manual testing/analysis is increasingly infeasible**
   - Directly tied to developer productivity
   - Progress has been made towards automating this task
   - Creating a performance test can be difficult
   - Are we doing better?

Time series data for code executed daily
Maintaining performance and portability through only time series data plots still requires an expert to determine significant changes.

- **Changepoint detection**: process of finding abrupt variations in time series data.
Changepoint detection for performance testing

Single Changepoint:

Given time series data: \( X = \{x_1, x_2, \ldots, x_n\} \), and subset: \( X^j_i = \{x_i, x_{i+1}, \ldots, x_j\} \),

Hypothesis tests:

\[ \begin{align*}
H_0 &: f_1^{\nu-1} = f_\nu^n, \\
H_A &: f_1^{\nu-1} \neq f_\nu^n,
\end{align*} \quad \forall \nu \in \mathcal{K},
\]

- **Null hypothesis** – \( X \) belongs to a single distribution
- **Alternative hypothesis** – there exists a changepoint \( \nu \) s.t. \( X_1^{\nu-1} \) and \( X_\nu^n \) belong to two separate distributions

- Two-sample Student’s t-test (equal variance), other options not tested
- Bonferroni correction used for multiple hypothesis testing: \( \alpha/k \)
- Only \( k = 10 \) number of tests determined from largest changes in time series
- Outliers removed above \( \sim 3 \) STD, up to 10%
Changepoint detection for performance testing

Multiple Changepoint:

- Sequential algorithm
  - Store changepoints as they appear, new subset is created after change
  - $m = 3$ consecutive detections required before confirming changepoint
  - Max sample size or “lookback window” set to, $w = 30$, to avoid hypersensitivity

Implementation:

- Performance metrics are store in json files
- Automated post-processing in python
- Results uploaded in html and email reports
  - [https://sandialabs.github.io/ikalash.github.io/](https://sandialabs.github.io/ikalash.github.io/)

Example of improvements: Kokkos memory 45k->8k MiB
[Greenland Ice Sheet, 1-7km, First-order Stokes]
Performance comparisons

**Performance Analysis:**

Given two time series: $X$ and $Y$

- Compute difference
- Find changepoints
- Compute mean between changepoints and 99% confidence interval

**Example:**

- Red/Squares: MueLu
- Blue/Circles: ML
- Latest results:
  - Starting Nov. 6th (8 samples)
  - Relative difference mean: **2.49%**
  - 99% CI: (1.16%, 3.81%)
Performance comparisons

More Examples:

Ifpack2/FROSch: 20.40% (99% CI: (19.48%, 21.30%))

MueLu/Ifpack2 without/with block decoupling: -4.12% (99% CI: (-5.13%, -3.11%))
Automated Performance Tuning
Motivation – Automated performance tuning

Problem Description:
- Find a **robust** set of parameters for optimal **performance** and **accuracy**.
- Often many runtime parameters to choose from (e.g. discretization, solver)
- Abundance of research/development on this topic

Motivations are similar to performance testing:

1) **Maintaining performance and portability** in the presence of active development
   - Code changes can cause optimal parameters to shift (algorithmic optimizations)
   - Compiler/TPL changes can cause optimal parameters to shift (new parameters)
   - Architecture changes can cause optimal parameters to shift (CPU->GPU)

2) **Improving performance and portability** in the presence of active development
   - Optimal parameters can vary greatly between compiler, architecture (CPU/GPU)
   - Performance can vary greatly with code changes (robustness)

3) **Manual tuning is increasingly infeasible**
   - Directly tied to developer/user productivity
   - Parameters become outdated
Blackbox optimization for performance tuning

Grid/Random Search:
• Simple, can be used for parameter exploration

Implementation:
• Utilize performance test to check robustness, performance and accuracy
• Parameters are in input files with yaml format
• Files are modified with python and scikit-learn is used for parameter selection

Partially Explored Extensions:
• Sequential optimization algorithms (model-based, Bayesian optimization)
• Sandia or open-source libraries (GPTune, Dakota)
• Integrate into performance testing framework (automated tuning)
Blackbox optimization for performance tuning

**Example:** Albany Land Ice multigrid preconditioner smoothers

**Smoothen parameters:**
- Limited to three levels, two smoothers
- Good parameter ranges provided by Christian/Ichi

**Results:**
- Applied to four cases (Greenland, 3-20km)
  - Different equations
  - Different architectures (CPU/GPU)
- 100 iterations, random search
- Timer: NOX Preconditioner + Linear Solve

<table>
<thead>
<tr>
<th>Cases</th>
<th>Manual Tuning (sec.)</th>
<th>Autotuning (sec.)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>blake_vel</td>
<td>3.533972</td>
<td>2.658731</td>
<td>1.33x</td>
</tr>
<tr>
<td>blake_ent</td>
<td>3.07725</td>
<td>2.036044</td>
<td>1.51x</td>
</tr>
<tr>
<td>weaver_vel</td>
<td>19.13084</td>
<td>16.30672</td>
<td>1.17x</td>
</tr>
<tr>
<td>weaver_ent</td>
<td>19.76345</td>
<td>15.00014</td>
<td>1.32x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cases</th>
<th>#Passed Runs</th>
<th>#Failed Runs</th>
<th>%Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>blake_vel</td>
<td>70</td>
<td>30</td>
<td>30%</td>
</tr>
<tr>
<td>blake_ent</td>
<td>37</td>
<td>63</td>
<td>63%</td>
</tr>
<tr>
<td>weaver_vel</td>
<td>71</td>
<td>29</td>
<td>29%</td>
</tr>
<tr>
<td>weaver_ent</td>
<td>26</td>
<td>74</td>
<td>74%</td>
</tr>
</tbody>
</table>
Conclusions/Discussion
Conclusions/Discussion

Automated Performance Testing

• Changepoint detection adds some level of confidence to changes in performance (regressions/improvements)
  • Doesn’t always work – sometimes too sensitive – trade-offs when tuning
  • Still requires good performance tests/metrics
  • Still requires human-in-the-loop to address regressions
  • Large number of tests/metrics could be overwhelming

Automated Performance Tuning

• Blackbox optimization coupled with nightly testing adds some level of confidence in optimal parameters
  • Preliminary results are promising but more research needed
  • Large number of failures, raises questions about robustness/applicability
  • Optimization is expensive!