Accelerating the Configuration Tuning of Big Data Analytics with Similarity-aware Multitask Bayesian Optimization

Ayat Fekry, Lucian Carata, Thomas Pasquier, Andrew Rice

akmf3@cl.cam.ac.uk
lucian.carata@cl.cam.ac.uk
High-level problem overview

• We want to:
  – optimize configurations of data processing frameworks (Hadoop, Spark, Flink) in workload-specific ways.
  – allow amortization of tuning costs in realistic settings:
    • evolving input data (increase in size, change of characteristics)
    • an elastic cluster configuration
High-level problem overview

• We want to:
  – optimize execution of workloads in data processing frameworks (Hadoop, Spark, Flink)
  – allow amortization of tuning costs in realistic settings:
    ● evolving input data (increase in size, change of characteristics)
    ● an elastic cluster configuration

• When assuming repeated workload execution
  – daily/weekly/monthly reporting
  – incremental data analysis
  – frequent analytics queries/processing
High-level solution overview

- **How:**
  - By incrementally tuning the configuration of the framework
    - per workload
    - determining and tuning only significant parameters
    - aim is to quickly converge to configurations close to optimum
High-level solution overview

• How:
  - By incrementally tuning the configuration of the framework
    • per workload
    • determining and tuning only significant parameters
  - By leveraging existing tuning knowledge across similar workloads
High-level solution overview

• **How:**
  - By incrementally tuning the configuration of the framework
    - per workload
    - determining and tuning only significant parameters
  - By leveraging existing tuning knowledge across similar workloads
  - By carefully combining a number of established ML techniques and adapting them to the problem domain
**Required puzzle pieces**

- **Workload characterization**
  1) Workload monitoring
  2) Workload representations
  3) Similarity analysis
Required puzzle pieces

• **Workload characterization**
  1) Workload monitoring
  2) Workload representations
  3) Similarity analysis

• **Similarity-aware tuning**
  4) Multitask Bayesian Learning

[1] K. Swersky et. all, *Multi-task bayesian optimization*
Workload characterization

- Monitoring workload characteristics & resource consumption
  - Metric examples:
    - number of tasks per stage, input/output size, data spilled to disk, etc
    - CPU time, memory, GC time, serialization time, ...
  - Representing metrics in relative terms
    - GC time as proportion of total CPU time
    - Amount of shuffled/disk spilled data as proportion of total input data
Workload characterization

- **Workload representation**
  - Would like a **low-dimensionality** representation because it’s difficult to come up with informative distance metrics in high-dimensional space
  - We propose an autoencoder based solution, where the low-dimensionality representation is **learned**
    - offline phase based on historic execution metrics
    - resulting encoding/decoding model can be reused
• **Similarity analysis**
  
  - Given new workload, find a **source** (already tuned) workload
    
    • Closest in encoded representation space (using $L_1$ norm)
    
    • Distance computed on a fixed **fingerprinting configuration** for the new workload
Similarity-aware tuning

- Assume a source workload $s$ was found for workload $w$
  1) Tune the same significant parameters as for $s$
  2) Retrieve Bayesian tuning model of $s$, $T_s$
  3) Add $w$ as a new task to $T_s$
  4) Suggest the next (tuned) configuration sample, $cs_w$ for $w$
  5) Update tuning model with metrics from executing $w$ with configuration $cs_w$
Similarity-aware tuning

- Natural criteria for stopping the tuning
  - e.g: Acquisition function maximum (Expected Improvement) drops below 10%

- Method able to detect inaccurate similar workload matching
  - Large difference between cost predicted by model and actual execution, across multiple executions
## Experiments

The table below shows the input data sizes (DS) for different workloads and their corresponding units:

<table>
<thead>
<tr>
<th>Workload (Abbrev)</th>
<th>Input data sizes (DS)</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DS1</td>
<td>DS2</td>
</tr>
<tr>
<td>PageRank (PR)</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>Bayes Classifier (Bayes)</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>Wordcount (WC)</td>
<td>32</td>
<td>50</td>
</tr>
<tr>
<td>TPC-H Benchmark (TPCH)</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>Terasort (TS)</td>
<td>20</td>
<td>40</td>
</tr>
</tbody>
</table>

*pre-tuned (source) set*
Tuned execution times (at convergence)

Source dataset: *-DS1
Tuned execution times (at convergence)

Source dataset: *-DS1
Time until finding best configuration

Source dataset: *-DS1
Extended tuned (source) dataset for Bayes-DS3

Source dataset: *-DS1 + Bayes DS2
Tuning cost amortization (Bayes-DS3)

SimTune source dataset: *-DS1
SimTune-extended source dataset: *-DS1 + Bayes-DS2
Thank you! Ready for questions!

https://github.com/ayat-khairiy/simtune

Interested in discussing off-line or collaborating?

akmf3@cl.cam.ac.uk
lucian.carata@cl.cam.ac.uk