Towards an Adaptive, Fully Automated Performance Modeling Methodology for Cloud Applications

Ioannis Giannakopoulos\textsuperscript{1}, \textbf{Dimitrios Tsoumakos}\textsuperscript{2} and Nectarios Koziris\textsuperscript{1}

\textsuperscript{1}: Computing Systems Laboratory, School of ECE, National Technical University of Athens, Greece \{ggian, nkoziris\}@cslab.ece.ntua.gr
\textsuperscript{2}: Department of Informatics, Ionian University, Greece dtsouma@ionio.gr
Introduction
Performance Modeling (or profiling)

“The ability to predict an application’s behavior, given the parameters that affect it as input”

Valuable for:

- Optimal Resource Configuration
- Bottleneck Identification
- Application Scaling
- Multi-Engine Analytics
Towards an Adaptive, Fully Automated Performance Modeling Methodology for Cloud Applications
D. Tsoumakos, IEEE IC2E 2018 April 20, Orlando, Florida, USA

Approaches

Existing approaches:

- White-box
  - Analytical Modeling
  - Interpretable Models
  - Good Accuracy for simple cases
  - Require Experience
  - Poor Results for complex cases
Towards an Adaptive, Fully Automated Performance Modeling Methodology for Cloud Applications
D. Tsoumakos, IEEE IC2E 2018 April 20, Orlando, Florida, USA

Approaches

Existing approaches:
- **White-box**
  - Analytical Modeling
  - Interpretable Models
  - Good Accuracy for simple cases
  - Require Experience
  - Poor Results for complex cases
- **Black-box**
  - Benchmarking & ML
  - Require no experience
  - Accuracy
  - Require time/computation
“Black-box” Profiling Challenges

Traditional Challenges:

1. Which dimensions?
   a. data (e.g., dataset size, # of columns, distribution, etc.)
   b. resources (e.g., # nodes, # cores, RAM, etc.)
   c. workload (e.g., k parameter of k-means, clustering implementation, etc.)

2. Which model?
   a. Linear Models
   b. Neural Networks
   c. Support Vector Machines

New Challenge: **Massive Configuration Space**
The challenges - Distributed Design

WordCount

Nodes = [2]

Nodes = [2,4]

Nodes = [2,4,6]
The challenges - Cloud Computing

The configuration space grows exponentially with the #dimensions and the #values per dimension!
Problem statement

Given:
1. Application A
2. Configuration space $D = d_1 \times d_2 \times \ldots \times d_n$
3. A positive number $B$

“Estimate an application profile $F_A : D \rightarrow \mathbb{R}$ using $D_s$ of size $B$, with the highest possible accuracy!”

or:

“Find the top-$B$ representative configurations to train a ML classifier”
Optimality & Observations
Optimality & Observations

Optimal solution is **NP-Hard:**
- For each possible set of size \( B \):
  - Deploy configuration
  - Train ML classifier
  - Evaluate it (test set)

Even deploying the entire \( D \) is not feasible:
- Prohibitive time + cost

Two observations regarding performance functions:
- Locality
- (piecewise) Linearity
Locality

“Neighboring” configurations (tend to) give similar performance

Hint:

*Divide-and-Conquer* to solve more (but simpler) problems
(piecewise) Linearity

“Neighborhoods” (tend to) present linear behavior (especially for resource-related dimensions)

Hint:

Start with linear models
Methodology
Overview

Algorithm 1 DT-based Adaptive Profiling Algorithm

1: procedure DTAdaptive(D, B, b)
2:   tree ← TREEINIT(∅), samples ← ∅
3:   while |samples| ≤ B do
4:     tree ← PARTITION(tree, samples)
5:     s ← SAMPLE(D, tree, samples, b)
6:     d ← DEPLOY(s)
7:     samples ← samples ∪ d
8:   model ← CREATEMODEL(samples)
9: return model
Space Partitioning with Decision Trees

Decision Trees:

- Tree Structures
- Space Partitioning
- Node types:
  - Test nodes: Space Boundaries
  - Leaves: Linear Models
Space Partitioning with Decision Trees

Space Partitioning → Decision Tree construction

- Substitute Leaves with Test nodes
- Which line to use?
  - Maximize leaf homogeneity
  - Well-known criteria: GINI Impurity, Entropy, Variance minimization, etc.
- Optimization Problem
  - Objective Function: \( \text{Score}(I) = -\frac{|L_1|R_1^2 + |L_2|R_2^2}{|L_1| + |L_2|} \)

Intuitively: the split line will generate two regions which best fit to linear models
Space Sampling

Objective: distribute a deployment budget $b$ ($b<B$) to each leaf
Space Sampling

Objective: distribute a deployment budget $b$ ($b<B$) to each leaf
Space Sampling

- How can we decide on the budget of each leaf?
  - Leaf error (exploitation)
  - Leaf size (exploration)

- Leaf’s score: 
  \[ \text{Score}(\text{leaf}) = w_{\text{error}} \cdot \frac{\text{error}(\text{leaf})}{\text{maxError}} + w_{\text{size}} \cdot \frac{\text{size}(\text{leaf})}{\text{maxSize}} \]

- # of samples assigned to each leaf \( l \): \( \text{Score}(l) \times b \)
- Uniform sampling inside the leaf
Space Sampling

Exploration vs Exploitation

\[ W_{\text{error}} = 1.0 \]
\[ W_{\text{size}} = 0.0 \]
Towards an Adaptive, Fully Automated Performance Modeling Methodology for Cloud Applications
D. Tsoumakos, IEEE IC2E 2018, April 20, Orlando, Florida, USA

Space Sampling

Exploration vs Exploitation

$W_{\text{error}} = 1.0$
$W_{\text{size}} = 0.5$
Space Sampling

Exploration vs Exploitation

$W_{\text{error}} = 0.0$
$W_{\text{size}} = 1.0$
Deployment

- **AURA**: A Python-based deployment tool with error-recovery enhancements
- Application description: DAG of scripts for each module
- Transient failures (network glitches, poor coordination, etc.)
- AURA re-executes only scripts that failed
- Achieve idempotence via lightweight filesystem snapshotting
Deployment DAG example

Hadoop Application Description:
- 1 Master node
- 2 Slave nodes
Modeling

- Training a new Decision Tree from scratch
- When $B$ comparable to $\#\text{dimensions}$, DT degenerates into linear regression
  - Use multiple Regressors (e.g., ANNs, SVMs, etc.)
  - Keep the one with the lowest Cross Validation score
  - In practice, such budgets are too low to extract meaningful models
- Piecewise linearity
  - Approximates complex functions given higher deployment budgets
Evaluation
Methodology & Performance Functions

● Testbed: private Openstack Cluster (8 nodes, 200 cores, 600G RAM, 8TB disk)

● Competitors:
  ○ PANIC
    ■ More samples between points samples with highest variance
  ○ Active Learning (Uncertainty Sampling)
    ■ More samples to areas of highest uncertainty
  ○ Uniform Sampling
    ■ Uniform distribution of samples

● Modeling with WEKA regressors
## Methodology & Performance Functions

<table>
<thead>
<tr>
<th>Application (perf. metric)</th>
<th>Dimensions</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spark Bayes (execution time)</td>
<td>YARN nodes # cores per node memory per node # of documents # of classes</td>
<td>4–20 2–8 2–8 GB 0.5–2.5 ($\times 10^6$) 50–200 classes</td>
</tr>
<tr>
<td>Hadoop Wordcount (execution time)</td>
<td>YARN nodes # cores per node memory per node dataset size</td>
<td>2–20 2–8 2–8 GB 5–50 GB</td>
</tr>
<tr>
<td>Media Streaming (throughput)</td>
<td># of servers video quality request rate</td>
<td>1–10 144p–1440p 50–500 req/s</td>
</tr>
<tr>
<td>MongoDB (throughput)</td>
<td># of MongoDB # of Mongoose request rate</td>
<td>2–10 2–10 5–75 ($\times 10^3$) req/s</td>
</tr>
</tbody>
</table>

SR: $|D_s| / |D| \times 100\%$
Modeling Error

- **Bayes**
  - Media Streaming
  - MongoDB

- **Wordcount**
  - Media Streaming
  - MongoDB

- **UNI**
  - Media Streaming
  - MongoDB

- **ACTL**
  - Media Streaming
  - MongoDB

- **PANIC**
  - Media Streaming
  - MongoDB

- **DTA**
  - Media Streaming
  - MongoDB
Cost-aware profiling

$$\text{Score(leaf)} = w_{\text{error}} \frac{\text{error(leaf)}}{\text{max Error}} + w_{\text{size}} \frac{\text{size(leaf)}}{\text{max Size}} - w_{\text{cost}} \frac{\text{cost(leaf)}}{\text{max Cost}}$$

<table>
<thead>
<tr>
<th>App/tons</th>
<th>SR</th>
<th>MSE</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0.2</td>
<td>0.5</td>
</tr>
<tr>
<td>Bayes</td>
<td>3%</td>
<td>+1%</td>
<td>-1%</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>+4%</td>
<td>+10%</td>
</tr>
<tr>
<td>Wordcount</td>
<td>3%</td>
<td>-1%</td>
<td>-5%</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>0%</td>
<td>+13%</td>
</tr>
<tr>
<td>Media Str.</td>
<td>3%</td>
<td>-1%</td>
<td>-3%</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>-6%</td>
<td>-2%</td>
</tr>
<tr>
<td>MongoDB</td>
<td>3%</td>
<td>+6%</td>
<td>+12%</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>-1%</td>
<td>-7%</td>
</tr>
</tbody>
</table>
Conclusions

● Divide-and-Conquer strategy
● “zoom-in” to *interesting* areas of the configuration space
● exploration vs exploitation
● Space Partitioning: important dimensions are partitioned more frequently
● Dimension Importance: linearity
Towards an Adaptive, Fully Automated Performance Modeling Methodology for Cloud Applications
I. Giannakopoulos, D. Tsoumakos and N. Koziris
IEEE International Conference on Cloud Engineering, April 17-20, Orlando, Florida, USA