A Case of Study on Hadoop Benchmark Behavior Modeling Using ALOJA-ML

Josep Ll. Berral, Nicolas Poggi, David Carrera

Workshop on Big Data Benchmarks
Toronto, Canada 2015
Context

ALOJA: framework to interpret Hadoop benchmarks performance data and tuning to provide recommendations on cost-effectiveness of systems

Challenges:

1. Need to go from expert-guided to automated analysis

2. Need to deal with huge amounts of data and find the most relevant

Approach

– Predictive Analytics can be applied to deploy model-based methods helping such analysis
ALOJA and Predictive Analytics

**Predictive Analytics**
- Encompasses statistical and Machine Learning (ML) techniques
- To make predictions of unknown events from historical events
  - *Forecast and foresight*
- Formerly known also as “applied modeling and prediction”
- Predict behavior elements and apply them to **extract knowledge**
- Machine Learning
  - Science and methods part of Data Mining in charge of “learning” (modeling) a system from some of its observations

**ALOJA-ML**: the ALOJA predictive analytics component for modeling benchmarks

- Benchmark executions
- **Predictive Analytics**
- Benchmark behavior
- Knowledge about benchmark
- Oracles for benchmark
Welcome to the ALOJA project, ALOJA is an initiative of the BSC-MSR research centre in Barcelona to explore Hadoop's potential. You can find introductory Slides and Papers in the ALOJA Reference menu.

This site is under constant development and it is in the process of being documented. For any questions, feel free to browse the site, the code, and send inquiries, feature requests or bug reports to: hadoop@bsc.es.

If you’re curious about the name of the project, visit ALOJA.

Site’s content:

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video DEMO of ALOJA</td>
<td>Brief video showcasing ALOJA's main online features (a bit outdated)</td>
</tr>
<tr>
<td>Benchmark Executions</td>
<td>This section presents the benchmark execution repository. It features more than 30,000 executions and counting. This tool allows you to browse, filter, search, and select distinct executions to compare and analyse its execution details.</td>
</tr>
<tr>
<td>Hadoop Job Counters</td>
<td>The Hadoop Job Counters section allows to browse the counters output at each of the Hadoop executions, filter them, and to order by a specific counter the selected runs (or all). The section presents the summary of all the Job execution counters, Map and Reduce specific counters, and the I/O subsystem counters. It also features the details by task: to understand the running time of each Map or Reduce process.</td>
</tr>
</tbody>
</table>
User work-flow

1. ALOJA Web front-end
2. Data filtering
3. ML Processing
4. Processed data into ALOJA tools
5. Display / Visualization
Experiments and algorithms are codified in R (except some learning algorithms called from the WEKA package, in JAVA)

This allows us to run the methods locally or process them in an external Web Service
Modeling Hadoop jobs

Methodology

- 3-step learning:
  - Different split sizes tested
    - Grant enough samples for testing (≥ 25%)
    - Attempt to reduce the training set without degrading the learning (25% < training < 50%)
  - Different learning algorithms
    - Regression trees
    - Nearest-neighbors learning
    - Neural networks (FFANN)
    - Linear and polynomial regressions
Learning Benchmarks

Capabilities of learning Benchmarks

- Modeling of benchmark behaviors
  - Examine in which degree configurations affect the execution
- Prediction of benchmark
  - execution times, resource consumption, …

Benchmark representation

- Without parametrizing benchmarks
  - Learning algorithms must treat benchmark names as discrimination categories
  - A model must have seen a benchmark before to predict it
- With benchmark parametrization (numeric) [w.i.p.]
  - Discover which elements of a system tailor a benchmark
  - A new benchmark should have trial runs (we already do without parametrizing)
The ALOJA data-set

- Over 40,000 Hadoop benchmark executions, from the “HiBench” suite
- Selected benches:
  - kmeans, pagerank, sort, terasort, wordcount, dfsioe_r/w
- Inputs: benchmark, hardware features, cloud provider + type of deployment, software configurations
- Output: execution time, used resources, …

Features

- Hardware characteristics: network storage type, cluster description
- Software configurations: # of maps, sort factor, file buffer size, block size…

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>bayes, terasort, sort, wordcount, kmeans, pagerank, dfsioe_read, dfsioe_write</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware Configurations</td>
<td></td>
</tr>
<tr>
<td>Network</td>
<td>Ethernet, Infiniband</td>
</tr>
<tr>
<td>Storage</td>
<td>SSD, HDD, Remote Disks {1-3}</td>
</tr>
<tr>
<td>Cluster</td>
<td># Data nodes, VM description</td>
</tr>
</tbody>
</table>

| Software Configurations     |                                                                                  |
| Maps                        | 2 to 32                                                                         |
| I/O Sort Factor             | 1 to 100                                                                        |
| I/O File Buffer             | 1KB to 256KB                                                                   |
| Replicas                    | 1 to 3                                                                          |
| Block Size                  | 32MB to 256MB                                                                  |
| Compression Algs.           | None, BZIP2, ZLIB, Snappy                                                      |
| Hadoop Info                 | Version                                                                        |

Configuration parameters on data-set
Benchmark Modeling results

Modeling
- Use of 50% of executions to train a model, 25% to validate, 25% to test
- Use of regression trees and nearest neighbor algorithms, among others

General model for different classifiers

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>MAE Valid.</th>
<th>RAE Valid.</th>
<th>MAE Test</th>
<th>RAE Test</th>
<th>Best parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression Tree</td>
<td>135.19523</td>
<td>0.16615</td>
<td>323.78544</td>
<td>0.18718</td>
<td>M = 5</td>
</tr>
<tr>
<td>Nearest Neighbors</td>
<td>169.64048</td>
<td>0.18968</td>
<td>232.01521</td>
<td>0.18478</td>
<td>K = 3</td>
</tr>
<tr>
<td>FFA Neural Nets</td>
<td>189.60124</td>
<td>0.24541</td>
<td>333.93250</td>
<td>0.26099</td>
<td>5 neurons (1-hl), 1000 max-it, decay 5\cdot10^{-4}</td>
</tr>
<tr>
<td>Polynomial Regression</td>
<td>167.98270</td>
<td>0.2321720</td>
<td>354.93680</td>
<td>0.2541475</td>
<td>degrees = 3</td>
</tr>
</tbody>
</table>

Mean and Relative Absolute Error per method, on best split and parameters found
Generalization vs. Specialization

General model vs. Specialized models
– One model (one training/update) vs. individual models (more specialized)
– General model is expected to have higher error, but not too much

Comparative without parametrization
– General model behaves worse than specific ones (but not much)
  • G.M.: RAE 0.184
  • Average S.M.: RAE 0.132
– G.M. does not over-fit as specifics

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>RAE on Specific model</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFSIOE read</td>
<td>0.29965</td>
</tr>
<tr>
<td>DFSIOE write</td>
<td>0.10763</td>
</tr>
<tr>
<td>k-means</td>
<td>0.12842</td>
</tr>
<tr>
<td>pagerank</td>
<td>0.11948</td>
</tr>
<tr>
<td>sort</td>
<td>0.12823</td>
</tr>
<tr>
<td>terasort</td>
<td>0.12599</td>
</tr>
<tr>
<td>wordcount</td>
<td>0.09702</td>
</tr>
<tr>
<td>General Model</td>
<td>0.184</td>
</tr>
</tbody>
</table>

Encouraging next work
– Study the parametrization of benchmarks for automatic learning
– …so we expect to get a better general model
Issues and opportunities

- Detection of outliers
  - Benchmarks showing high learning errors: signal that executions are unstable or have high presence of outliers
  - Automating learning can put apart those benchmarks or executions

- E.g., pagerank (having outliers) returns a RAE of 1.18 → execs put to revision
  - After cleaning the pagerank anomalous executions → RAE of 0.12

- Prediction of output variables
- Ranking of relevant configuration features
Use case 1: Anomaly Detection

Anomaly Detection

- Model-based detection procedure

- The selected model becomes “the system”. Any execution not fitting into the model is supposed to be out of the system.
Use case 1: Anomaly Detection

Anomaly and Outlier Detection
- Use of statistic and model-based outlier detections
- Highlight executions with high probability of anomaly
- Mark down executions with high probability of being errors
Use case 2: Features and Discriminators

Discrimination of Features

- Use the models (general or specifics)
- Create a ranking of features, according to the estimated results
- Possible discrimination
  - By information gain
  - By ordered splits (best variable to split configurations by their outputs)

<table>
<thead>
<tr>
<th>Net</th>
<th>Disk</th>
<th>IO.FBuf</th>
<th>Blk.Size</th>
<th>Prediction (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETH</td>
<td>HDD</td>
<td>65536</td>
<td>128</td>
<td>2249.766</td>
</tr>
<tr>
<td>IB</td>
<td>HDD</td>
<td>65536</td>
<td>128</td>
<td>2737.112</td>
</tr>
<tr>
<td>ETH</td>
<td>SSD</td>
<td>65536</td>
<td>128</td>
<td>1036.366</td>
</tr>
<tr>
<td>IB</td>
<td>SSD</td>
<td>65536</td>
<td>128</td>
<td>1036.366</td>
</tr>
<tr>
<td>ETH</td>
<td>HDD</td>
<td>131072</td>
<td>128</td>
<td>2165.927</td>
</tr>
<tr>
<td>IB</td>
<td>HDD</td>
<td>131072</td>
<td>128</td>
<td>2653.273</td>
</tr>
<tr>
<td>ETH</td>
<td>SSD</td>
<td>131072</td>
<td>128</td>
<td>969.537</td>
</tr>
<tr>
<td>IB</td>
<td>SSD</td>
<td>131072</td>
<td>128</td>
<td>969.537</td>
</tr>
<tr>
<td>ETH</td>
<td>HDD</td>
<td>65536</td>
<td>256</td>
<td>2249.766</td>
</tr>
<tr>
<td>IB</td>
<td>HDD</td>
<td>65536</td>
<td>256</td>
<td>2737.112</td>
</tr>
<tr>
<td>ETH</td>
<td>SSD</td>
<td>65536</td>
<td>256</td>
<td>1036.366</td>
</tr>
<tr>
<td>IB</td>
<td>SSD</td>
<td>65536</td>
<td>256</td>
<td>1036.366</td>
</tr>
<tr>
<td>ETH</td>
<td>HDD</td>
<td>131072</td>
<td>256</td>
<td>2165.927</td>
</tr>
<tr>
<td>IB</td>
<td>HDD</td>
<td>131072</td>
<td>256</td>
<td>2653.273</td>
</tr>
<tr>
<td>ETH</td>
<td>SSD</td>
<td>131072</td>
<td>256</td>
<td>969.537</td>
</tr>
<tr>
<td>IB</td>
<td>SSD</td>
<td>131072</td>
<td>256</td>
<td>969.537</td>
</tr>
</tbody>
</table>

*Terasort, 4 maps, sort factor 10, no comp*

Tree Descriptor:

```
Disk=HDD
  └── Net=ETH
      └── IO.FBuf=128KB ⇒ 2935s
          └── IO.FBuf=64KB ⇒ 2942s
  └── Net=IB
      └── IO.FBuf=128KB ⇒ 3118s
          └── IO.FBuf=64KB ⇒ 3125s

Disk=SSD
  └── Net=ETH
      └── IO.FBuf=128KB ⇒ 1248s
          └── IO.FBuf=64KB ⇒ 1256s
  └── Net=IB
      └── IO.FBuf=128KB ⇒ 1233s
          └── IO.FBuf=64KB ⇒ 124s1
```
Case of use 3: Knowledge Discovery

- Make analyzing results easier
  - Multi-variable visualization
  - Trees separating relevant attributes
  - Other interesting tools
Conclusions

- Modeling: Specific models are a little bit more accurate than generals (but not so much)
  - High error at automatic modeling can indicate outliers
- Unfolding the search space of Hadoop configurations
  - Observe predictions
  - Rank features by possible relevance

Next steps:

- Characterization and parametrization of benchmarks
- Guided executions
Online repository and tools available at: http://hadoop.bsc.es

Publications

- ALOJA project: automatic characterization of cost-effectiveness on Hadoop deployments
  

- ALOJA-ML: Predictive analytics tools for benchmarking on Hadoop deployments

Thanks!

Q&A

Contact: hadoop@bsc.es