ROBOTune: High-Dimensional Configuration Tuning for Cluster-Based Data Analytics

Md Muhib Khan, Weikuan Yu
Florida State University

Outline

- **Introduction**
- Background and Motivation
- Design
- Implementation
- Evaluation
- Conclusion

What is ROBOTune?

- ROBOTune is an automatic configuration tuning framework.
- What does it aim to tune?
  - Cluster-based data analytics applications.
- ROBOTune is geared towards tuning configuration parameters of Apache Spark.

Outline

- Introduction
- **Background and Motivation**
- Design
- Implementation
- Evaluation
- Conclusion

The Need for Tuning Spark

- Most data analytics workloads are recurring jobs in a cluster.
  - Optimization can save a tremendous amount of time and resources.
- Why not use the default configuration?
  - Underperforms significantly compared to near-optimal configurations.
- Why not tune manually?
  - Random trial-and-error by inexperienced users would give vastly different results.
  - Complex relationship between configuration and performance.

Challenges for Parameter Configuration

- High-dimensional configuration space.
  - The number of Spark configuration parameters has increased by 10x!
  - Exponential increase in the complexity of the configuration space.
- Costly sample collection.
  - State-of-the-art learning-based tuners require a significant number of samples to work (at least 2000). Impractical!
  - Auto-tuners should be cost efficient and flexible.

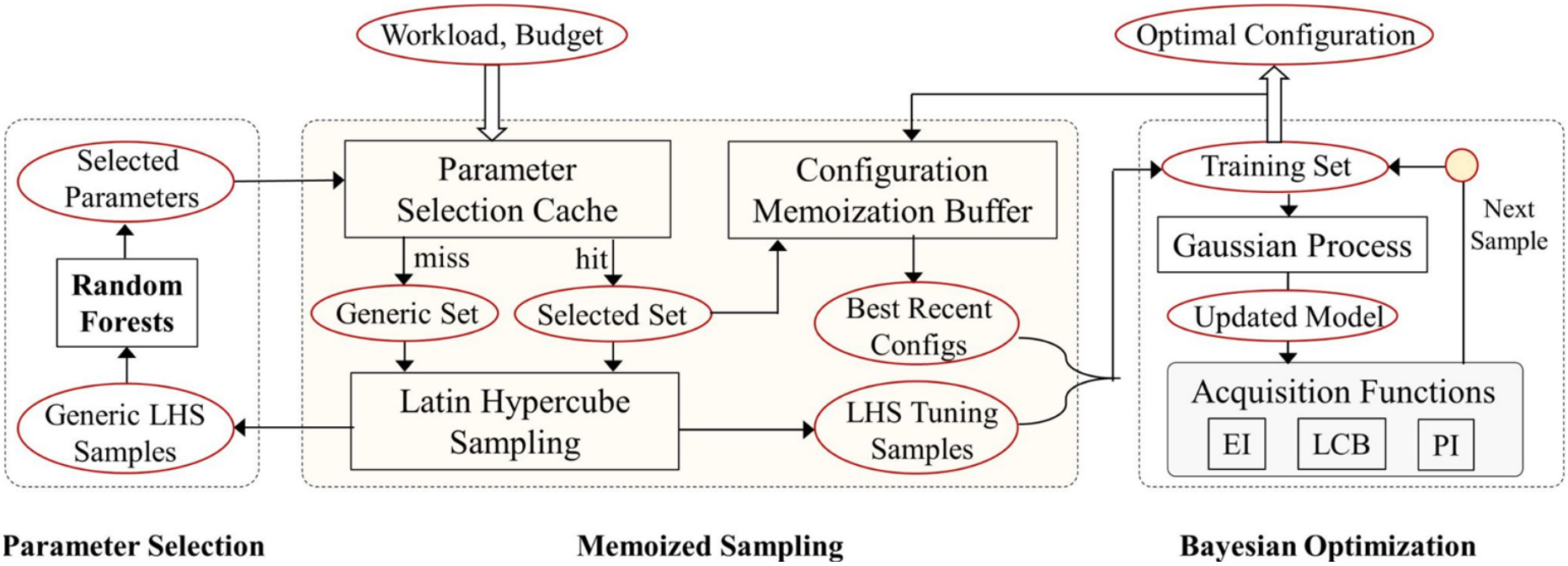
Bayesian Optimization

- Bayesian Optimization (BO) is a powerful method of optimizing expensive black-box functions.
  - Works for non-convex and noisy functions.
  - Effectively avoids local optima.
  - A minimal number of samples is required. Thus very cost-efficient!
  - Caveat! Works best when the number of dimensions is low.

Outline

- Introduction
- Background and Motivation
- **Design**
- Implementation
- Evaluation
- Conclusion

Design Overview



Design Overview Cont.

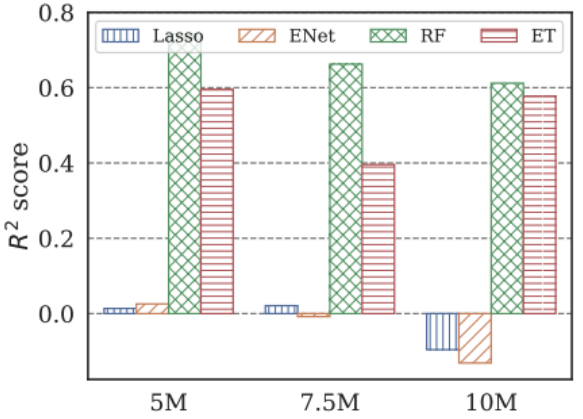
1. Memoized Sampling.
    - Latin Hypercube Sampling (LHS) with a parameter selection cache and a configuration memoization buffer.
2. Parameter Selection.
    - Selects impactful parameters using a Random Forests Model.
3. Bayesian Optimization Engine.
    - Balances exploration and exploitation.
    - Iteratively searches for the best configuration using a Gaussian Process (GP) model and a portfolio of acquisition functions.

Memoized Sampling

- LHS requires fewer samples than random sampling to reach a similar conclusion.
- Parameter Selection Cache.
  - The high-impact parameters of a workload remain the same for a range of datasets.
  - Reuses the selected parameters for new datasets of the same workload.
- Configuration Memoization Buffer.
  - Includes well-performing configurations from previous tuning sessions in the BO training set.

Parameter Selection - Choice of Random Forests

- Using ML models, we can measure the strength of the relationship between the objective and the parameters.
- Tree-based estimators (e.g., RF) perform better with lower number of samples than Linear models.
- We do not need a perfect predictor!



(a) PageRank.

Parameter Selection Cont.

- Mean Decrease in Accuracy (MDA) is used for feature (parameter) ranking with RF.
  - Slower but more robust than the default method.
- Handling collinearity.
  - Group dependent parameters with the independent one, during the parameter importance calculation.

Bayesian Optimization Engine

- Bayesian Optimization is composed of two main components.
  - A Bayesian statistical model for modeling the objective function. Also known as the surrogate.
  - An acquisition function that guides the search by selecting candidate points for sampling.

Choice of Model - Gaussian Process

- The surrogate can be Random Forests, Gradient Boosted Trees, Gaussian Process etc.
- A Gaussian Process is a distribution over multivariate functions.
  - Provides a theoretically justified way to trade-off exploration and exploitation.
  - Has been applied successfully to real-word systems.

Choice of Acquisition Function - GP-Hedge

- Three main choices.
  - Probability of Improvement (PI).
  - Expected Improvement (EI).
  - Lower Confidence Bound (LCB).
- No acquisition function is guaranteed to perform the best on an unknown objective.
- Why not use all of them?
  - GP-Hedge is a portfolio of acquisition functions.
  - The probability of choosing an acquisition function is updated based on the cumulative rewards at each step.

Outline

- Introduction
- Background and Motivation
- Design
- **Implementation**
- Evaluation
- Conclusion

Implementation Details

- Parameter selection is implemented using the Scikit-learn library.
- Memoized Sampling is implemented using the DOEPY library.
- Bayesian Optimization Engine is built on top of the Scikit-Optimize library.
- More details
  - Acquisition functions are optimized using the L-BFGS-B minimizer.
  - The covariance kernel is the summation of Matern 5/2 and White noise kernel.

Outline

- Introduction
- Background and Motivation
- Design
- Implementation
- **Evaluation**
- Conclusion

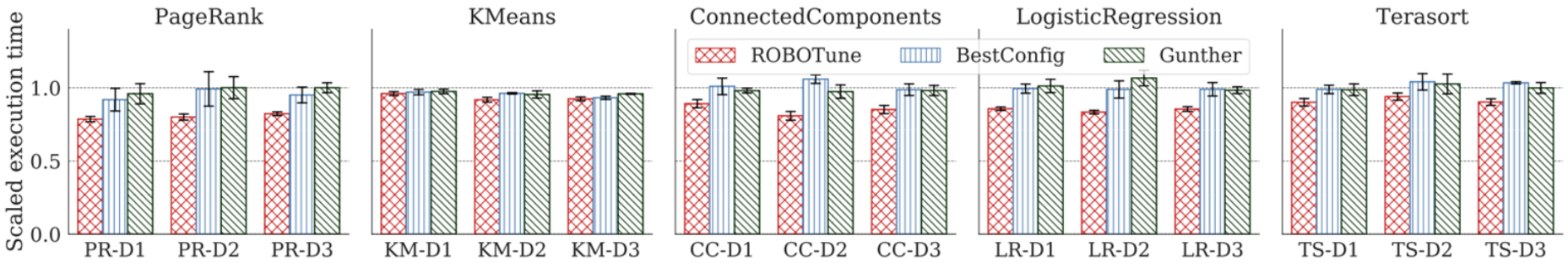
Experimental Setup

- Cluster configuration.
  - 6 nodes with a total of 192 cores and 1152 GB of memory.
  - Spark 2.4.1.
  - HDFS 2.7.3 for data storage.
- Configuration Parameters
  - 44 performance-related parameters that cover shuffle, networking, memory management, execution behavior etc.
  - Exhaustive search is infeasible due to the enormity and high-dimensionality of the configuration space.

Comparative Solutions And Workloads

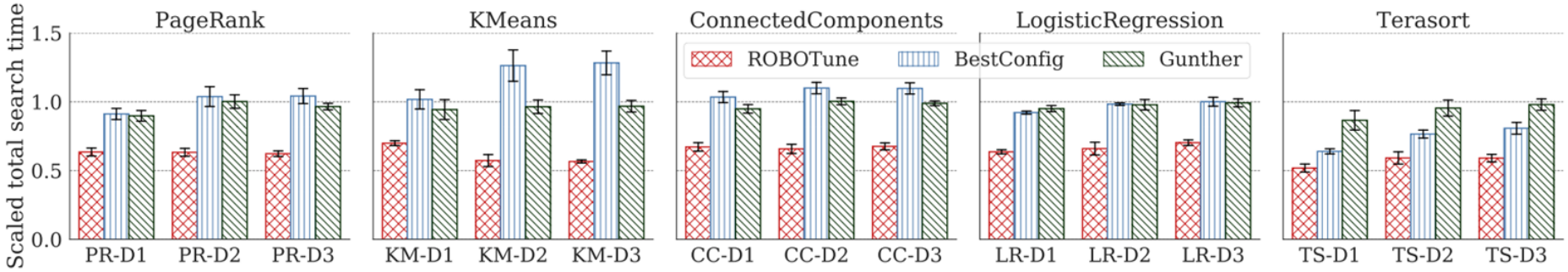
- BestConfig: A search-based tuning approach that uses a divide-and-diverge sampling and recursive bound-and-search algorithm.
- Gunther: A configuration tuner for Hadoop, which utilizes Genetic Algorithm for searching near-optimal configurations.
- Random Search: A simple random search-based tuning approach.
- Five representative workloads, each with three datasets from SparkBench.
  - PageRank (PR).
  - KMeans (KM).
  - ConnectedComponents (CC).
  - LogisticRegression (LR).
  - Terasort (TS).

Performance Of Optimal Configurations



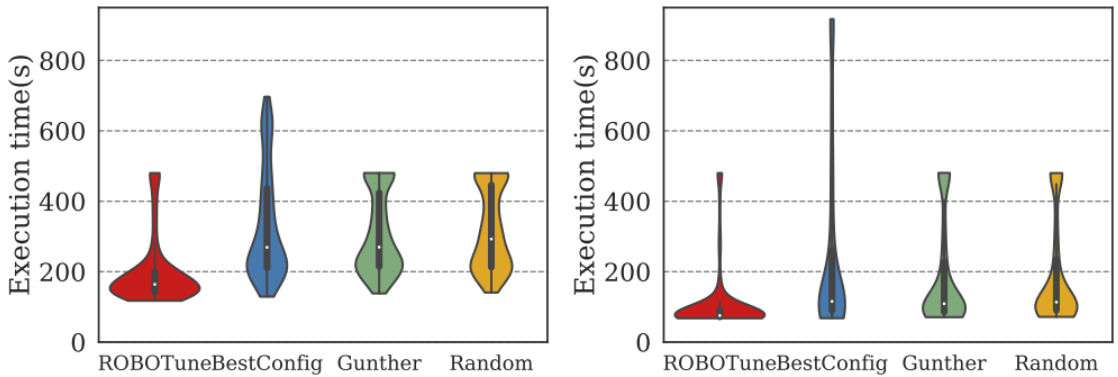
- ROBOTune outperforms BestConfig by 1.14× on average and up to 1.3×.
- Similarly, outperforms Gunther by 1.15× on average and up to 1.28×.
- For Random Search, the speedup is 1.15× on average and up to 1.27×.
- Finds much better configurations for PageRank and CC and and moderately better for Terasort.

Search Cost



- The overall improvement of ROBOTune over BestConfig is 1.59× on average and up to 2.27×.
- The improvement over Gunther is 1.53× on average and up to 1.71×.
- Random Search is outperformed by 1.6× on average and up to 1.93×.

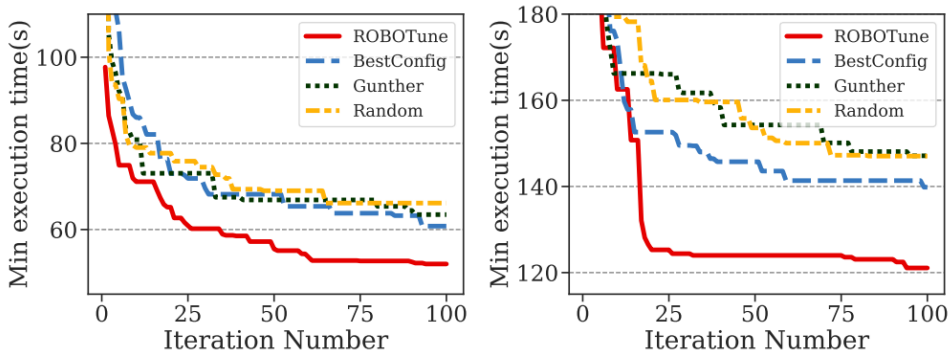
Why is the Search Cost Lower for ROBOTune?



(a) Distribution for PR-D3 (b) Distribution for KM-D3

- Distribution of workload execution time shows that, ROBOTune in general samples better performing configurations.
- ROBOTune avoids underperforming and imbalanced configuration regions, and in turn, lowers search costs.

Search Speed



- ROBOTune quickly finds configurations within 5% of the observed best.
- Using well-performing samples in the initial set, ROBOTune immediately gets within 10% of the best observed time and then further improves the performance.

Outline

- Introduction
- Background and Motivation
- Design
- Implementation
- Evaluation
- **Conclusion**

Conclusion

- We have designed and developed ROBOTune for tuning configurations of data analytics frameworks while tackling the issues of
  - High-dimensionality of the configuration space.
  - Complex configuration-performance relationship.
- Our evaluation shows that ROBOTune achieves a significant search cost and search speed improvement while finding better or similar configurations.

Sponsor of this Research

Questions?