

# Compiling Machine Learning Algorithms with SystemML

## Extended Abstract

M. Boehm D. Burdick A. Evfimievski B. Reinwald P. Sen S. Tatikonda Y. Tian

IBM Research - Almaden, San Jose, CA, USA

Analytics on big data may range from passenger volume prediction in transportation to customer satisfaction in automotive diagnostic systems, and from correlation analysis in social media data to log analysis in manufacturing. Expressing and running these analytics for varying data characteristics and at scale is challenging. To address these challenges, SystemML implements a declarative, high-level language using an R-like syntax extended with machine learning specific constructs, that is compiled to a MapReduce runtime [1]. The language is rich enough to express a wide class of statistical, predictive modeling and machine learning algorithms (Figure 1). We chose algorithms that are robust and scale to large, and potentially sparse data with large numbers of features.

The declarative approach of SystemML provides flexibility in terms of implementing new algorithms or customizing existing algorithms, and enables automatic optimization. The flexibility provided by the high-level language leads to significant productivity gains for developers of machine learning algorithms in terms of lines of code and development time compared to Java MapReduce. Automatic optimization frees the algorithm developer from hand-tuning specific execution plans for different data characteristics.

SystemML applies optimizations at different levels of abstraction during algorithm compilation, and generates low-level execution plans for the runtime (Figure 1). The compiler includes a cost-based optimizer for operator ordering, selecting the best execution strategy for individual operators, as well as piggybacking/packing planned SystemML runtime operators into a workflow of MapReduce jobs. Our descriptive statistics are numerically stable [2]. SystemML generates hybrid runtime execution plans that range from in-memory, single node execution to large-scale cluster execution of operators, hence enabling scaling up and down of computation. Key technical innovations are the careful combination of the integrated runtime for single node and distributed execution, caching, cost-based execution planning using a cost-model based on accurate time and worst-case memory estimates, as well as progressive re-optimization. SystemML also supports task parallelism

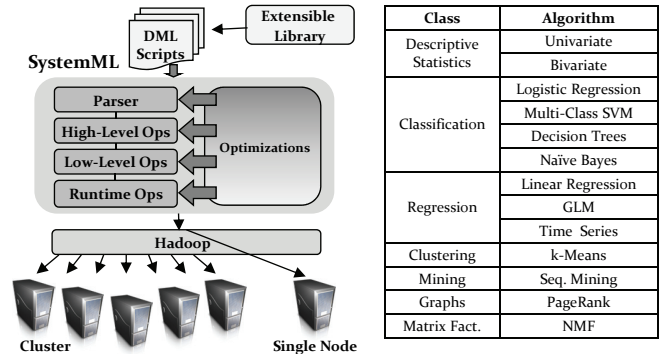


Figure 1: Algorithms and Architecture

through which independent iterations in loops can be executed in parallel [3].

In summary, the expressiveness of the SystemML language allows for a diverse class of algorithms which can be extended by leveraging external libraries. Implementing algorithms using the declarative language may also have a standardization effect in an enterprise which simplifies reuse and maintenance of analytic algorithms. SystemML on top of MapReduce scales to large clusters and thus enables analyzing PBytes of data and/or solving compute-intensive problems. In contrast to custom distributed runtime frameworks, MapReduce provides global scheduling which allows to share cluster resources with other systems such as Jaql, SystemT, Pig, or Hive.

## 1. REFERENCES

- [1] A. Ghoting, R. Krishnamurthy, E. Pednault, B. Reinwald, V. Sindhwani, S. Tatikonda, Y. Tian, and S. Vaithyanathan. Systemml: Declarative machine learning on mapreduce. In *ICDE*, pages 231–242, 2011.
- [2] Y. Tian, S. Tatikonda, and B. Reinwald. Scalable and numerically stable descriptive statistics in systemml. In *ICDE*, pages 1351–1359, 2012.
- [3] M. Boehm, S. Tatikonda, B. Reinwald, P. Sen, Y. Tian, D. Burdick, and S. Vaithyanathan. *Hybrid Parallelization Strategies for Large-Scale Machine Learning in SystemML*. in submission, 2013.