Compiling Machine Learning Algorithms with SystemML

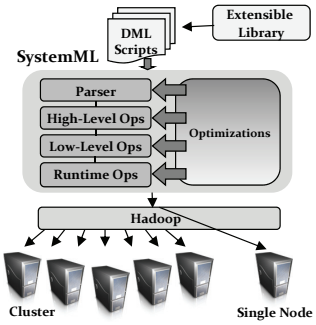
M. Boehm, D. Burdick, A. Evfimievski, B. Reinwald, P. Sen, S. Tatikonda, and Y. Tian
IBM Research - Almaden, San Jose, CA 95120, USA

Analytics on big data 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 [2]. The language is rich enough to express a wide class of statistical, predictive modeling and machine learning algorithms (Fig. 1). We chose robust algorithms that scale to large, and potentially sparse data with many 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 and cluster characteristics.

SystemML applies optimizations at different levels of abstraction during algorithm compilation, and generates low-level execution plans for the runtime (Fig. 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 [3]. 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 integrated runtime for

SoCC'13, 1–3 Oct. 2013, Santa Clara, California, USA.
ACM 978-1-4503-2428-1. http://dx.doi.org/**10.1145/2523616.2525965**



Class	Algorithm
Descriptive Statistics	Univariate
	Bivariate
Classification	Logistic Regression
	Multi-Class SVM
	Decision Trees
	Naïve Bayes
Regression	Linear Regression
	GLM
	Time Series
Clustering	k-Means
Mining	Sequence Mining
Graphs	PageRank
Matrix Factorization	NMF

Figure 1: Architecture and Algorithms

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 through which independent loop iterations can be executed in parallel [1].

In summary, the expressiveness of the SystemML language allows for a diverse class of algorithms which can be extended by 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, Pig, or Hive.

References

[1] 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.

[2] 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.

[3] Y. Tian, S. Tatikonda, and B. Reinwald. Scalable and Numerically Stable Descriptive Statistics in SystemML. In *ICDE*, pages 1351–1359, 2012.