Resource Elasticity for Large-Scale Machine Learning

Botong Huang1, Matthias Boehm1, Yuanyuan Tian1, Berthold Reinwald1, Shirish Tatikonda1, Frederick R. Reiss1

1 IBM Research – Almaden; San Jose, CA, USA
2 Duke University; Durham, NC, USA

Motivation

Problem Description

- **Declarative Machine Learning**
  - Goal: Write ML algorithms independent of input data / cluster characteristics
- **Full flexibility** → ML DSL
- **Data independence** → Coarse-grained ops
- **Efficiency and scalability** → Hybrid plans
- **Specified algorithms** → Opt performance

- **Problem of Memory-Sensitive Plans**
  - State-of-the-art compilers sensitive to memory constraints of static cluster configuration
  - Issue #1: Users need to reason about plans
  - Issue #2: Finding a good cluster configuration is hard (algorithm variety/multi-tenancy)

Experiments (as of 10/2014)

- **Optimization**
  - Resource of Memory-Sensitive Plans
  - Problem of Memory-Sensitive Plans
  - State-of-the-art compilers sensitive to memory constraints of static cluster configuration
- **Resource Adaptation**
  - Integrated w/ Dynamic Recompilation
  - (1) Determine re-optimization scope
  - (2) Resource re-optimization
  - (3) Adaptation decision
  - (4) Runtime migration

Resource Optimization

System Architecture

- **SystemML's YARN Integration**
  - Control program (CP/MR) runs as an AM

Resource Optimizer Overview

- **Basic Ideas**
  - Optimize ML program resources via online what-if analysis
  - Program-aware grid point generation and pruning

End-to-End Runtime

- **Linreg DS (XS-L), dense1000**
  - 3.5x
  - 0.2
  - 0.8

- **Linreg CG (Conjugate Gradient)**
  - 6.8x
  - 0.35s

Optimization Framework

- **ML Program Resource Allocation Problem**
  - Goal: Minimize costs w/o unnecessary over-provisioning

- **Efficiency and scalability**
  - Data independence
  - Full flexibility

- **ML DSL**
  - Hybrid plans
  - Coarse-grained ops

- **Hybrid plans**
  - Efficient and scalable
  - Data independence
  - Full flexibility

- **System Architecture**
  - SystemML's YARN Integration
  - Control program (CP/MR) runs as an AM

- **Resource Optimizer Overview**
  - Basic Ideas
    - Optimize ML program resources via online what-if analysis
  - Grid Point Generators
    - Systematic: Equi-/Exp-spaced grid
    - Directed: Memory-based grid
    - Composite grids
    - Default: (Exp/MinMax) × (Exp/MinMax)

- **End-to-End Throughput**
  - Linreg DS (Scenario S, dense1000)
    - B-LL: 63 GB/44 GB → new parallelism: 6
    - Opt. 8GB/2GB → 6x

Experiment Setting

- **Hadoop Cluster**
  - 1+6 nodes
  - 1 node: 2x Intel X5550, HT, 64 GB RAM,
  - Hadoop Cluster
    - IBM Hadoop 2.2.0, IBM JDK 1.6.0 64bit SR12
    - YARN max allocation per node: 90 GB
      - 1.9x (1.9x-M), HDF5 block size 128 MB

- **ML Programs, Data and Baselines**
  - b full-fledged ML scripts
  - Dense/sparse (1.0/0.01), 1000/100 features
  - X16: (10^10) cells, in dense: 80MB – 800 GB
  - B-SS (6.6MB), B-LS (6.6MB), B-LL (16MB/4.4GB)

- **Resource Adaptation**
  - Integrated w/ Dynamic Recompilation
  - (1) Determine re-optimization scope
  - (2) Resource re-optimization
  - (3) Adaptation decision
  - (4) Runtime migration

- **Grid Enumeration Algorithm**
  - Overall Algorithm
    - Grid enumeration of CP/MR resources
    - Program-aware grid point generation and pruning
  - Grid Point Generators
    - Systematic: Equi-/Exp-spaced grid
    - Directed: Memory-based grid
    - Composite grids
    - Default: (Exp/MinMax) × (Exp/MinMax)

- **Pruning Techniques**
  - Pruning blocks of (1) small ops / (2) unknowns
  - High impact of pruning strategies

Contact: Matthias Boehm
mboehm@us.ibm.com

03/2015: IBM BigInsights 4.0 GA, BigR/SystemML (opt level)