Intermediate presentation
Gilbert: A sparse linear algebra environment

Till Rohrmann
till.rohrmann@campus.tu-berlin.de

Technische Universität Berlin

March 19, 2014

Table of Contents

Demand for big data analytics

- More and more data gathered
- Companies want to exploit the gathered data to obtain new insights
  - Crime site prediction
  - Recommender systems
- Data grows exponentially
- Analytic methods have to scale up as well

Ways to scale up

1. Develop new algorithm
2. Increase computer performance
3. Run in parallel

Data analytic methods

- Many data analytic and machine learning algorithms based on linear algebra
- Developed with linear algebra systems, such as Matlab, R, Octacve, etc.
    - + Easy to code and test -> Quick development
    - + Huge existing code base
    - - Explicit parallelization

Distributed computing systems

- Explicit parallelization, such as OpenMP and MPI, tedious and error-prone
- New parallel programming paradigms emerged, MapReduce, Spark, Stratosphere, Pregel, etc.
- Frees from low-level parallelization tasks
- Requires to adhere to a certain programming model

Distributed computing systems and data analytics

- Intersection of people familiar with both domains is really small
- Laborious to become acquainted with new domain
- Tedious to transform existing algorithms to new programming model
- Can't we bring both worlds together?
- Solution: Gilbert

Gilbert

- Sparse linear algebra system
- Matlab frontend for distributed computing frameworks
- Allows to almost seamlessly move from local execution to distributed execution in a heterogenous environment
- Enable machine learning and data analytics algorithm to be run on web-scale data

Approach

1. Compiling Matlab code into intermediate representation
2. Intermediate representation: Abstraction to apply transformations independently of distributed computing system
3. Execution on specific distributed computing system

Language

- Matlab clone
- Support of basic linear algebra operations
- Some built-in functions, repmat, linspace, pdist2
- Loop support, fixpoint-Iteration
- Language expressive enough to support variety of algorithms, Pagerank, k-means, NNMF

Compilation

- Scala's combinator parsing utility -> Creates backtracking parser
- Powerful enough for our language
- Matlab is dynamically typed
- Execution on Stratosphere requires type knowledge at compilation time
- Hindley-Milner type inference algorithm to infer types and dimensions

Runtime

- Translation into runtime specific execution format
- Support of Stratosphere, (maybe Spark)

Current state

- Language layer working (except for the typing algorithm which is still a little bit buggy)
- Execution on Stratosphere with iteration and convergence support
- PageRank, NNMF and K-means executable

Live demonstration

Outlook

- Implement typing system with constraints, e.g. Haskell typing system with type class support
- Evaluate implementation: Runtime, scalability
- Compare to specialized algorithms
- Implement optimizations for the intermediate representation

Bibliography I