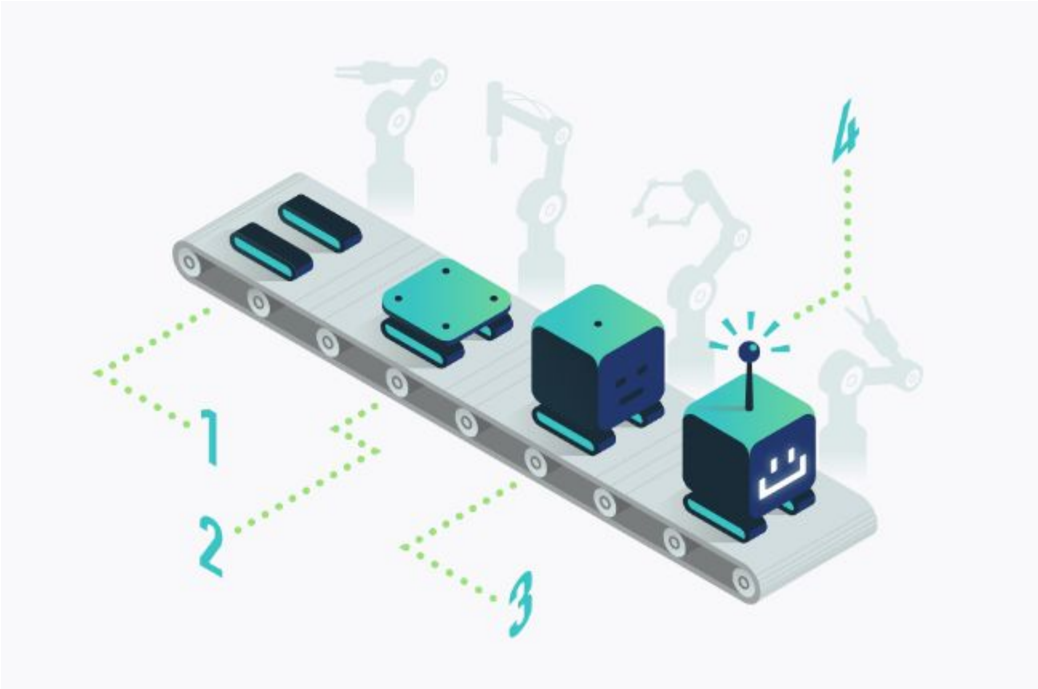
SystemML: Declarative Machine Learning on Spark

05/03/19

Presented by: Juan Carrillo
Candidate for MAScc. in Computer Software
Department of Electrical & Computer Engineering
University of Waterloo



UNIVERSITY OF WATERLOO

Agenda

1. Introduction
2. SystemML core features
3. Experiments
4. Conclusions
5. Discussion

Introduction

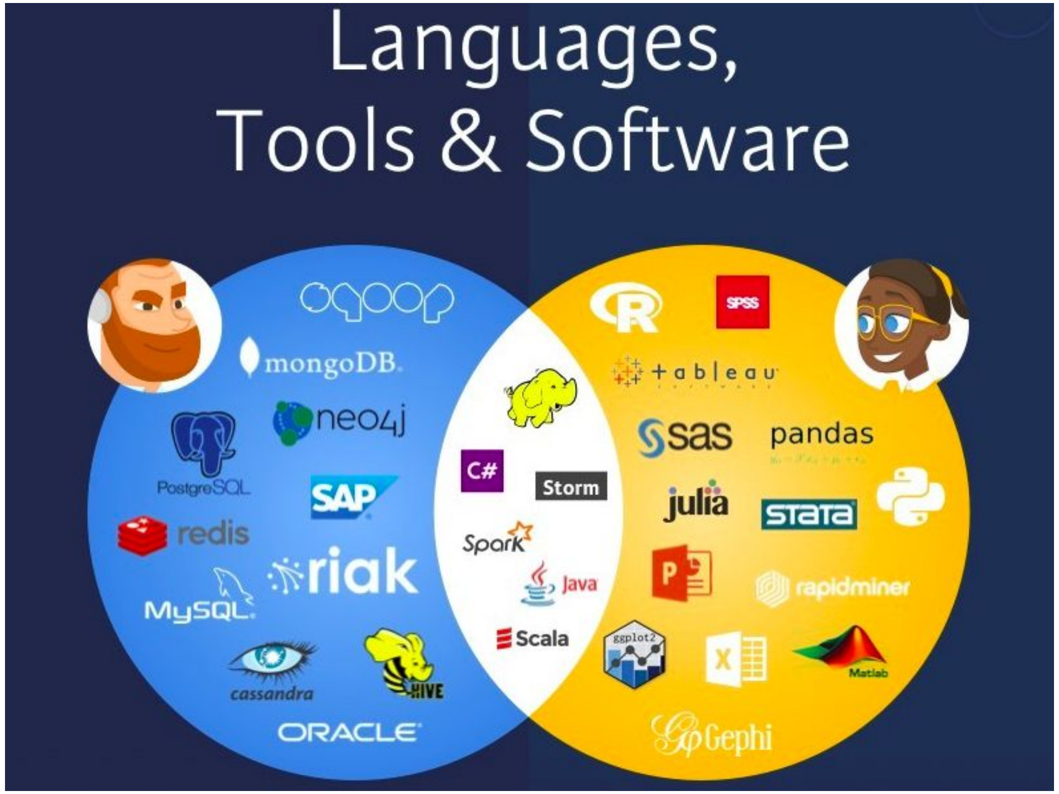
1. Introduction

Machine Learning for Big Data Analytics

DATA
Engineer
DataCamp
Learn Data Science By Doing
DATA
Scientist

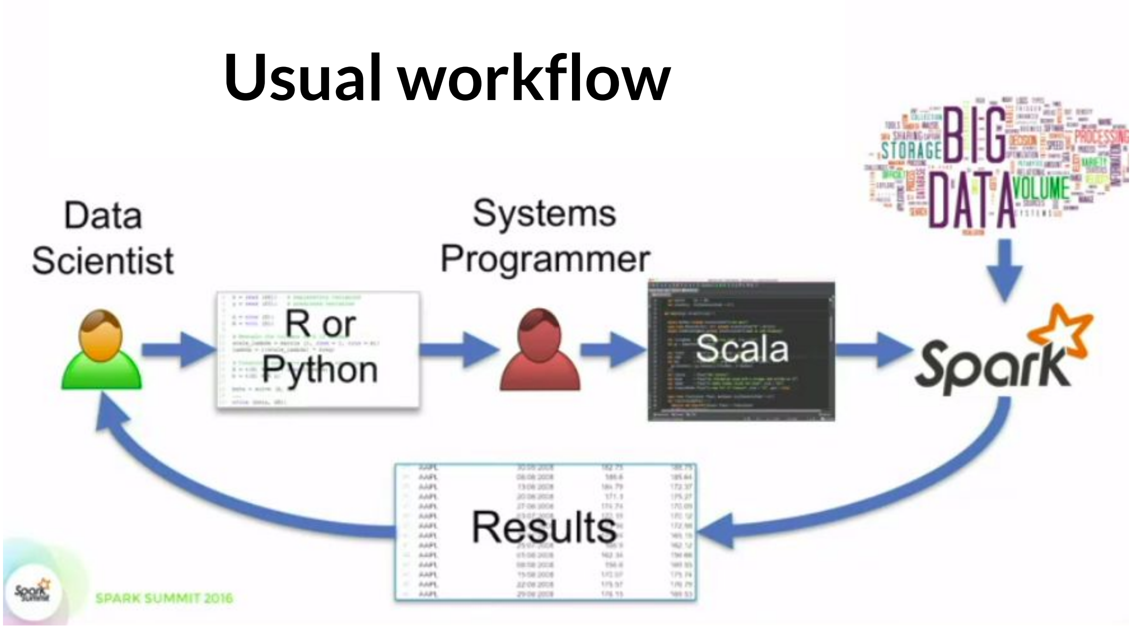
Develops, constructs, tests, and maintains architectures. Such as databases and large-scale processing systems.

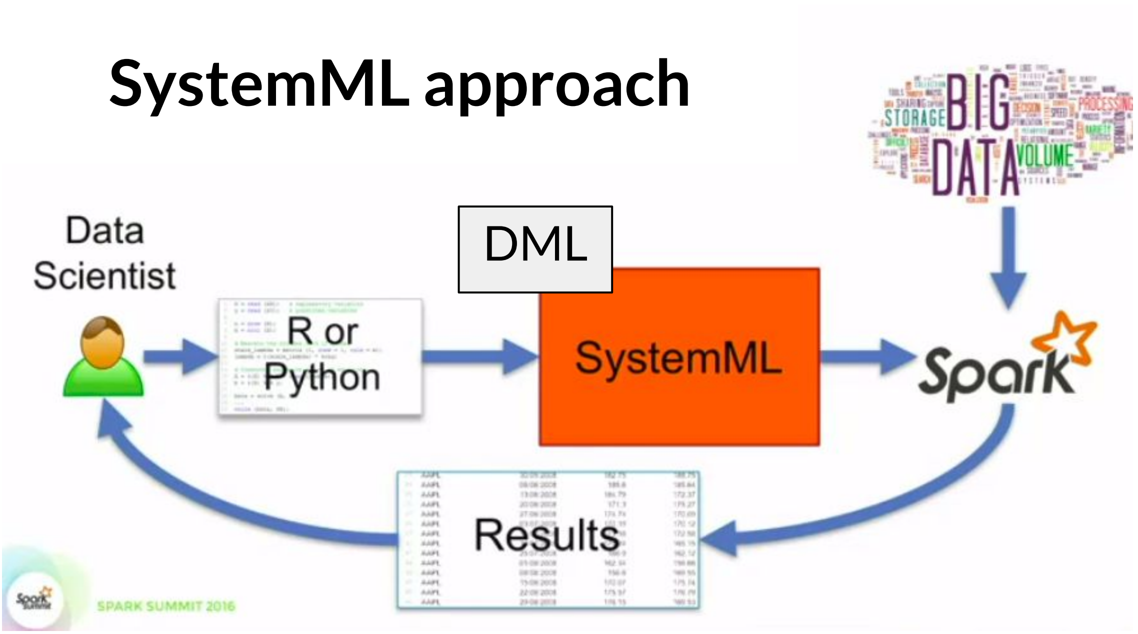
Cleans, massages and organizes (big) data. Performs descriptive statistics and analysis to develop insights, build models and solve a business need.



1. Introduction

The problem, and the SystemML approach



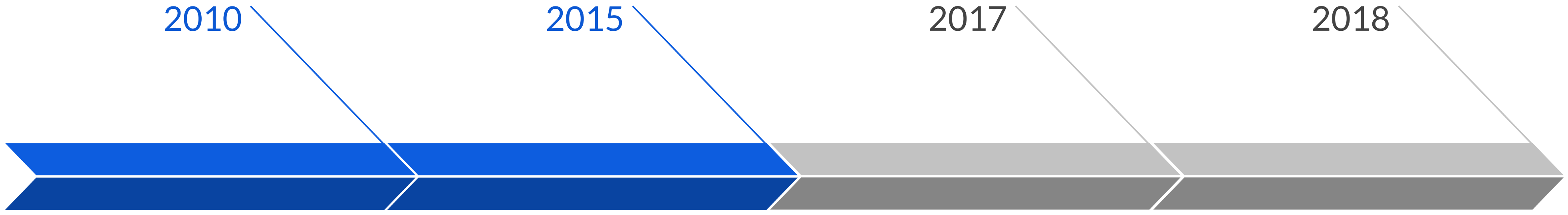


- Time consuming
- Error prone

- Accelerates model development
- Simplifies deployment

Source: Spark Summit. Inside Apache SystemML

SystemML background



2010

Creation

By researchers at the IBM Almaden Research Center

2015

Open-source

Spark Summit in San Francisco

2017

Top Level Project

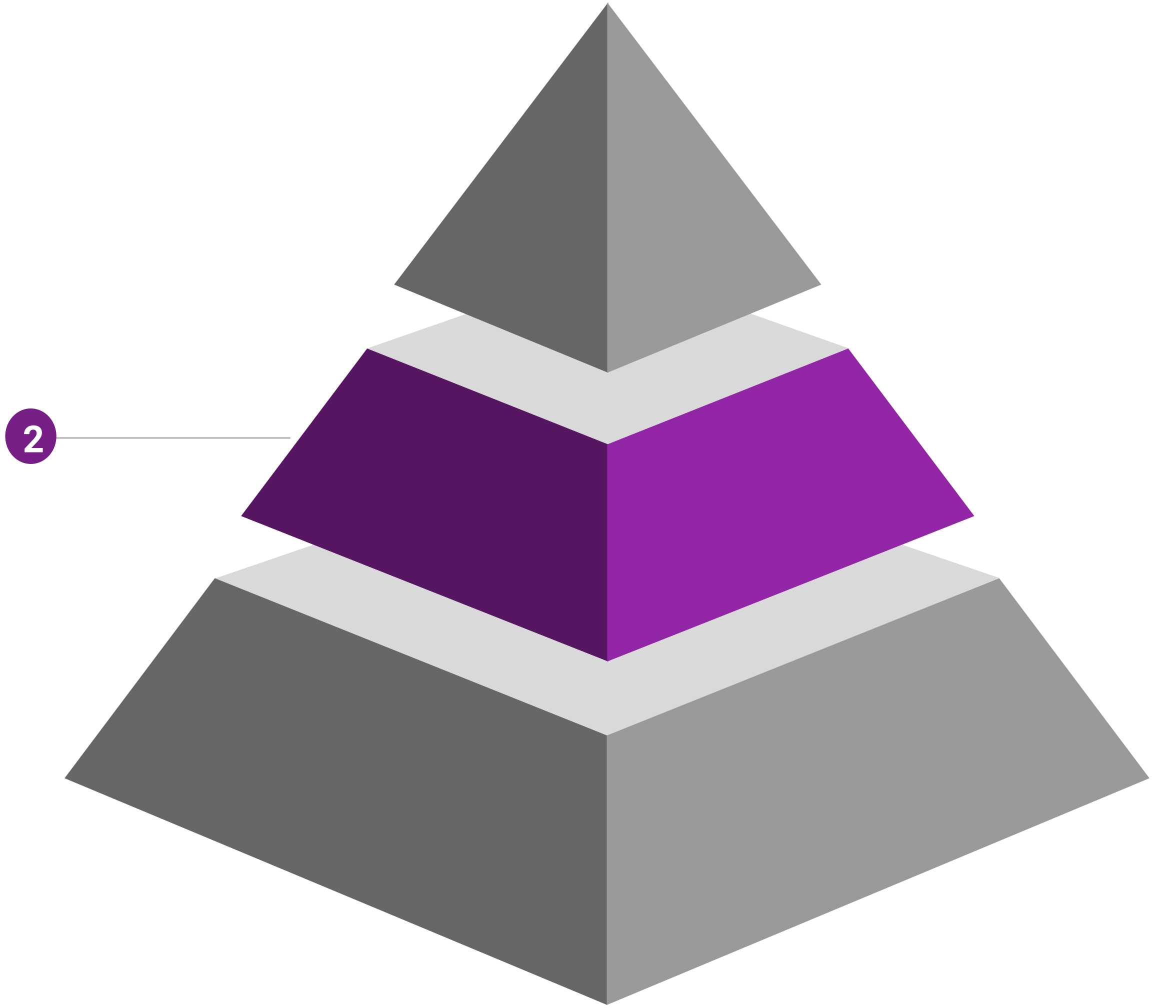
Apache Software Foundation Board

2018

Current release 1.2

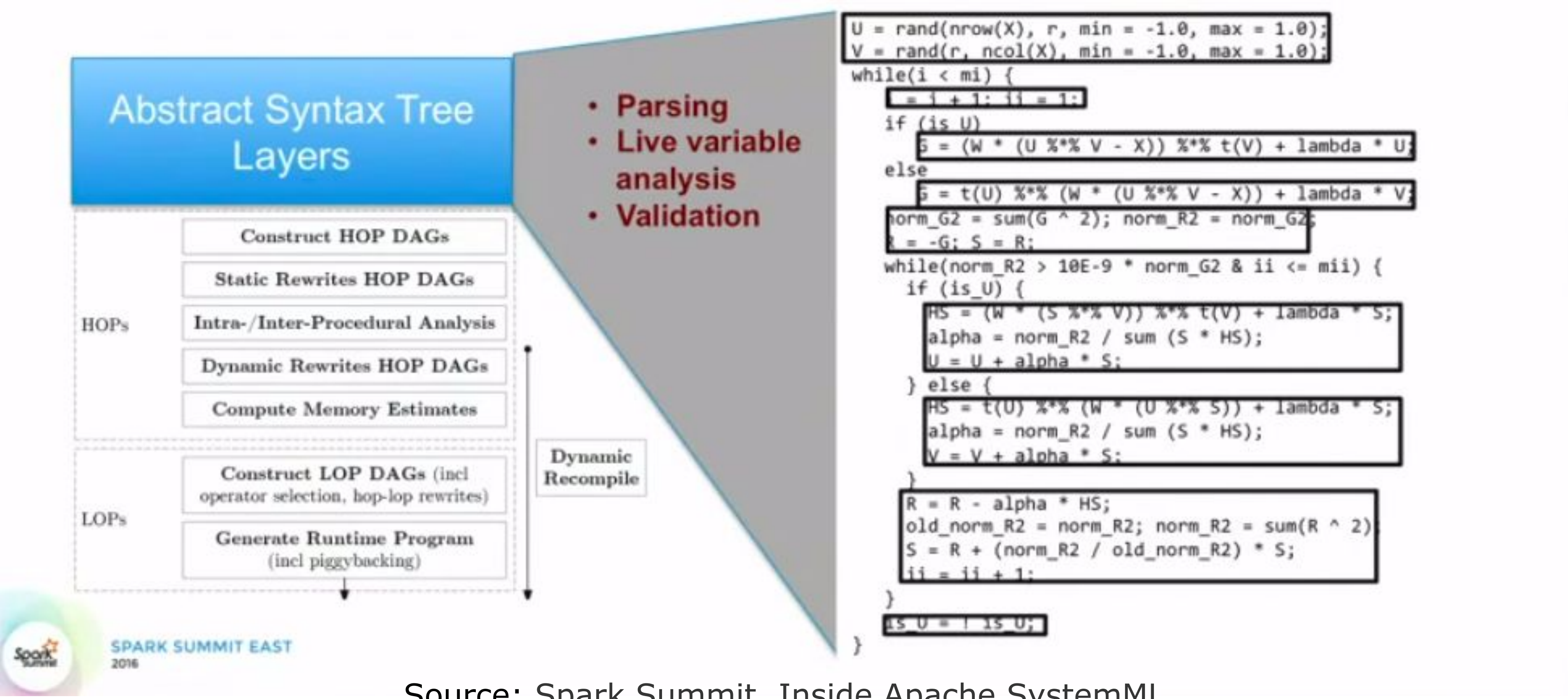
Deep learning functions
Ultra-sparse data

SystemML core features



2. SystemML core features

Optimizer integration



Abstract Syntax Tree Layers

- Parsing
- Live variable analysis
- Validation

HOPs

- Construct HOP DAGs
- Static Rewrites HOP DAGs
- Intra-/Inter-Procedural Analysis
- Dynamic Rewrites HOP DAGs
- Compute Memory Estimates

LOPs

- Construct LOP DAGs (incl operator selection, hop-lop rewrites)
- Generate Runtime Program (incl piggybacking)

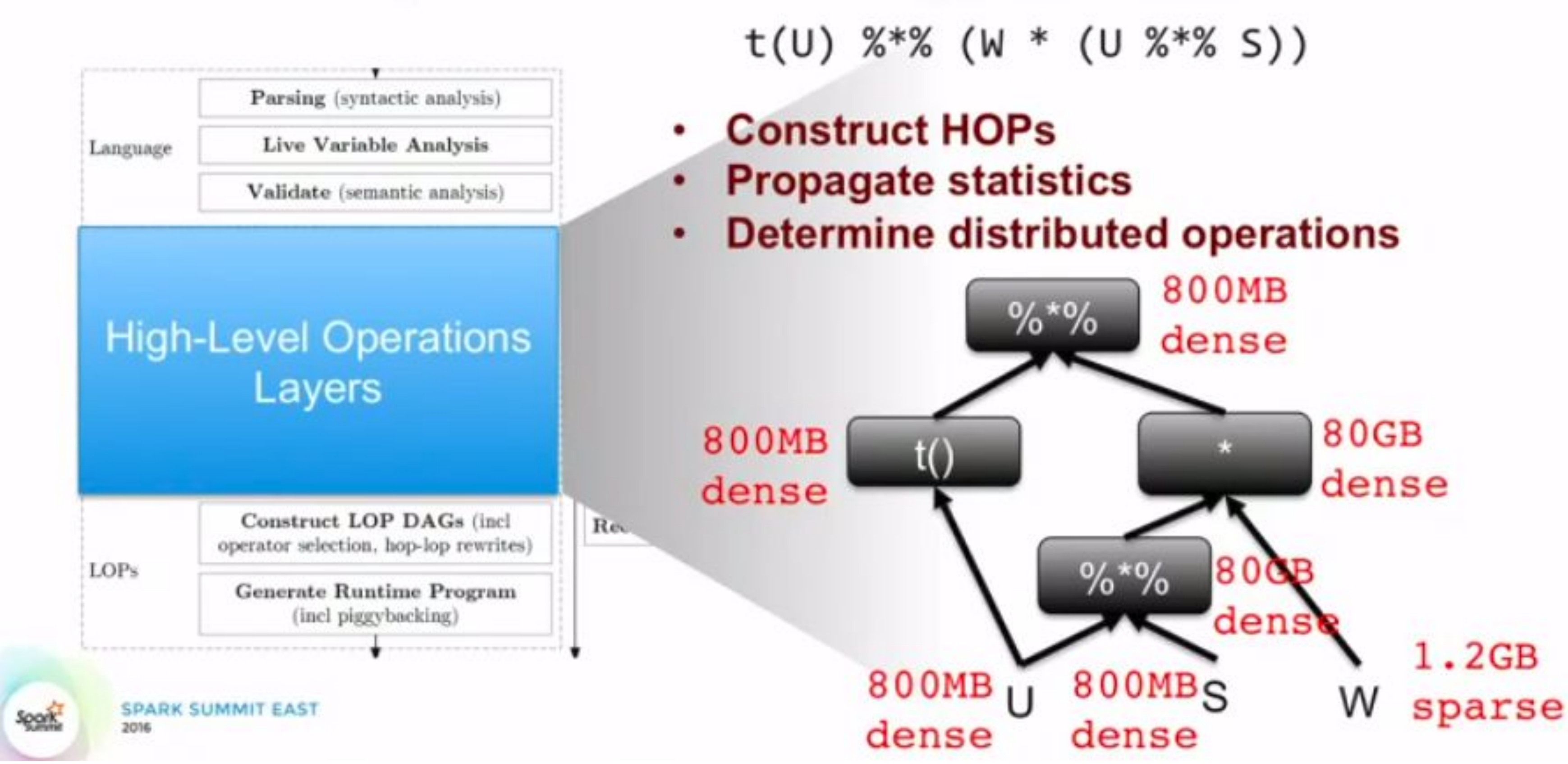
Dynamic Recompile

```
U = rand(nrow(X), r, min = -1.0, max = 1.0);
V = rand(r, ncol(X), min = -1.0, max = 1.0);
while(i < mi) {
    i = i + 1; ii = 1;
    if (is_U)
       G = (W * (U %*% V - X)) %*% t(V) + lambda * U;
    else
       G = t(U) %*% (W * (U %*% V - X)) + lambda * V;
    norm_G2 = sum(G ^ 2); norm_R2 = norm_G2;
    R = -G; S = R;
    while(norm_R2 > 10E-9 * norm_G2 & ii <= mii) {
      if (is_U) {
        HS = (W * (S %*% V)) %*% t(V) + lambda * S;
        alpha = norm_R2 / sum (S * HS);
        U = U + alpha * S;
      } else {
        HS = t(U) %*% (W * (U %*% S)) + lambda * S;
        alpha = norm_R2 / sum (S * HS);
        V = V + alpha * S;
      }
     R = R - alpha * HS;
     old_norm_R2 = norm_R2; norm_R2 = sum(R ^ 2);
     S = R + (norm_R2 / old_norm_R2) * S;
     ii = ii + 1;
    }
    is_U = ! is_U;
}
```

Source: Spark Summit. Inside Apache SystemML

2. SystemML core features

Optimizer integration



```
t(U) %*% (W * (U %*% S))
```

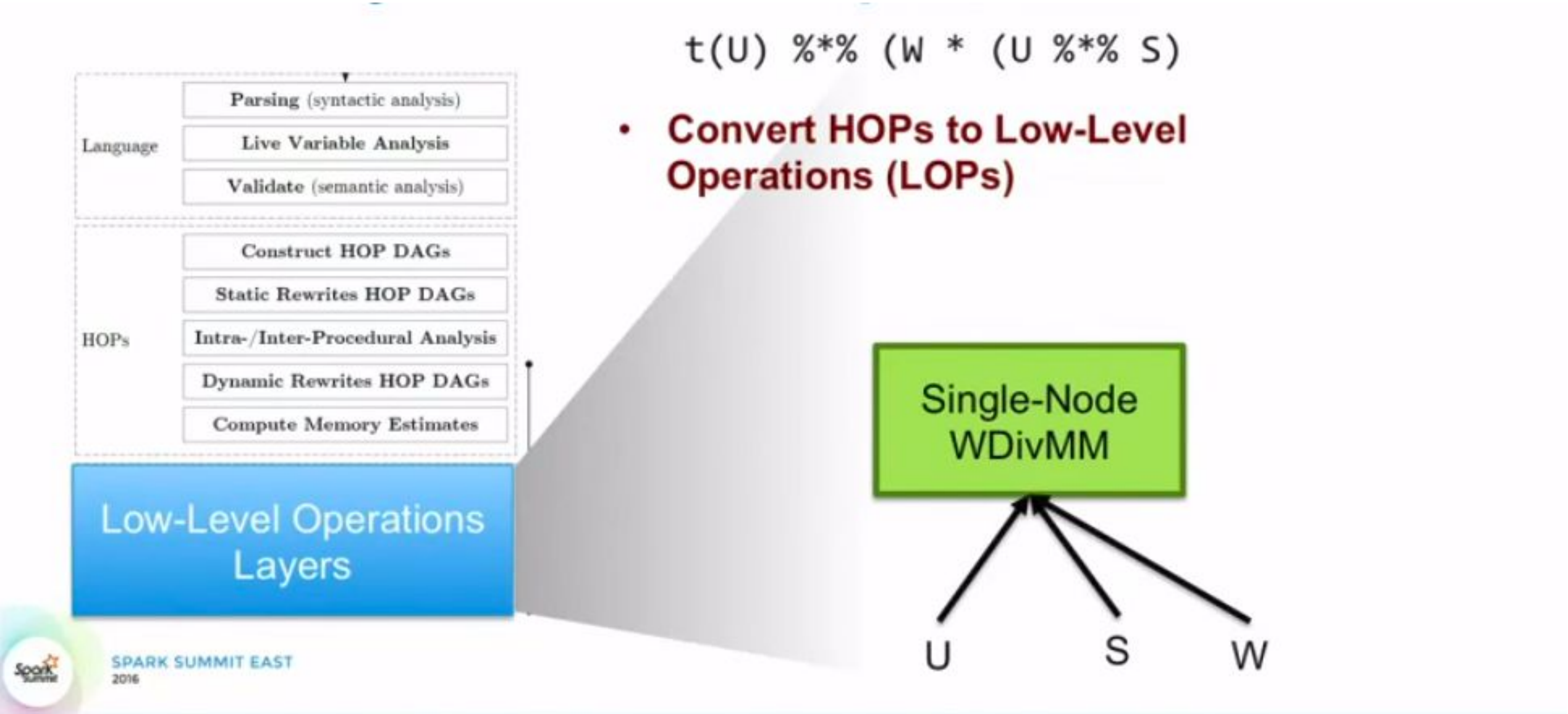
- Construct HOPs
- Propagate statistics
- Determine distributed operations

Language
Parsing (syntactic analysis)
Live Variable Analysis
Validate (semantic analysis)
High-Level Operations Layers
LOPs
Construct LOP DAGs (incl operator selection, hop-lop rewrites)
Generate Runtime Program (incl piggybacking)
Re
%*%
800MB dense
t()
800MB dense
*
80GB dense
%*%
80GB dense
800MB dense
U
800MB dense
S
W
1.2GB sparse
SPARK SUMMIT EAST 2016

Source: Spark Summit. Inside Apache SystemML

2. SystemML core features

Optimizer integration



`t(U) %*% (W * (U %*% S))`

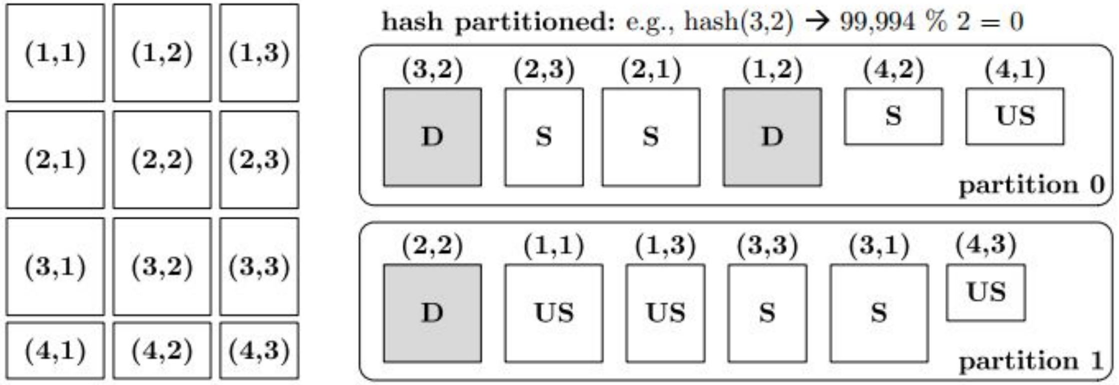
- Convert HOPs to Low-Level Operations (LOPs)

Source: Spark Summit. Inside Apache SystemML

2. SystemML core features

Runtime integration

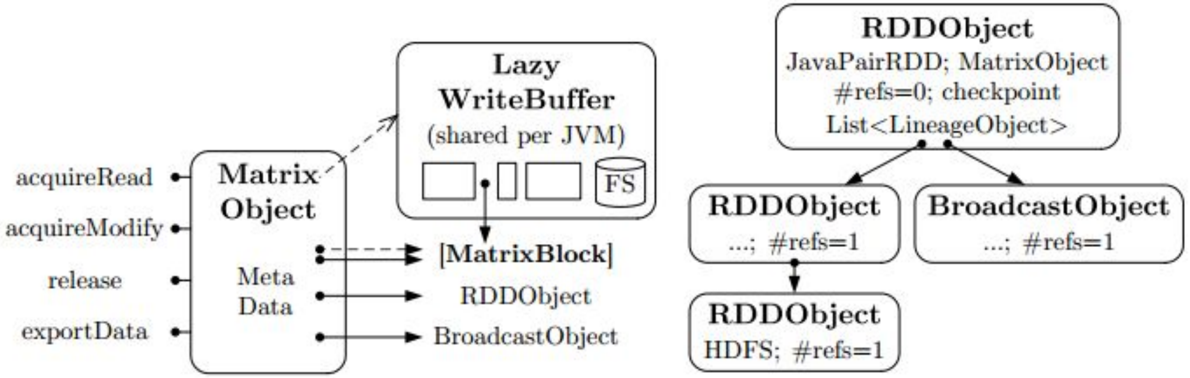
Distributed Matrix Representation



(a) Logical (b) Physical Partitioning

Figure 2: Distributed Matrix Representation.

Buffer Pool Integration

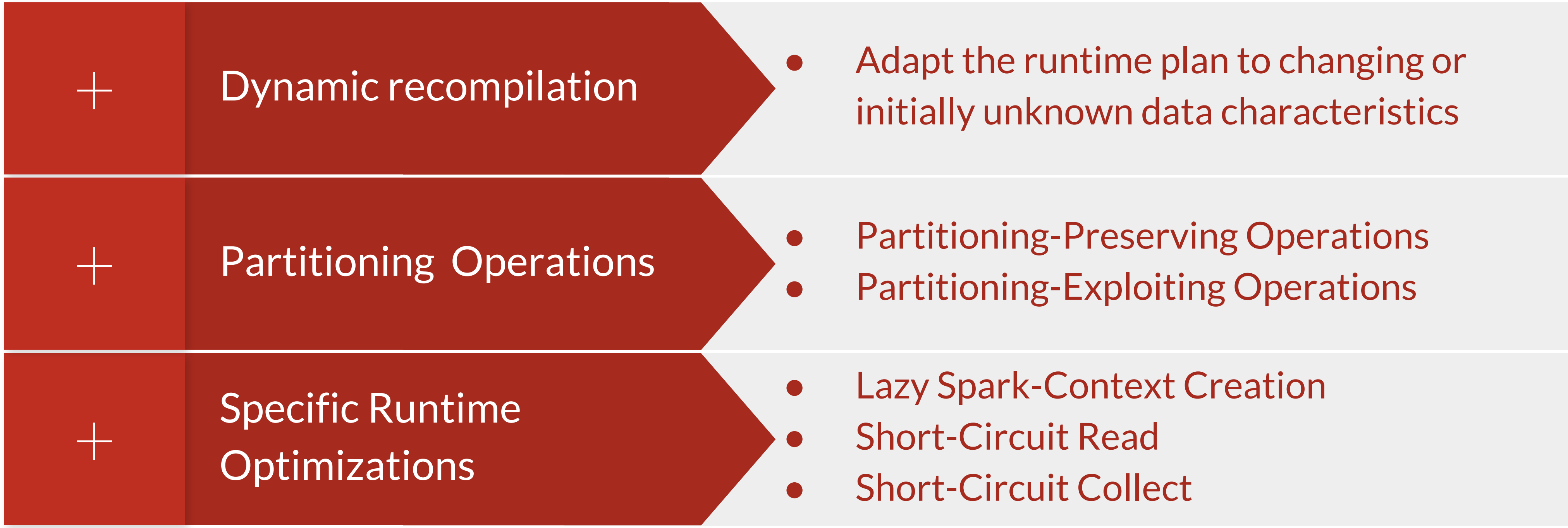


(a) MatrixObject / WriteBuffer (b) Lineage Tracking

Figure 3: Buffer Pool Integration.

2. SystemML core features

Runtime integration

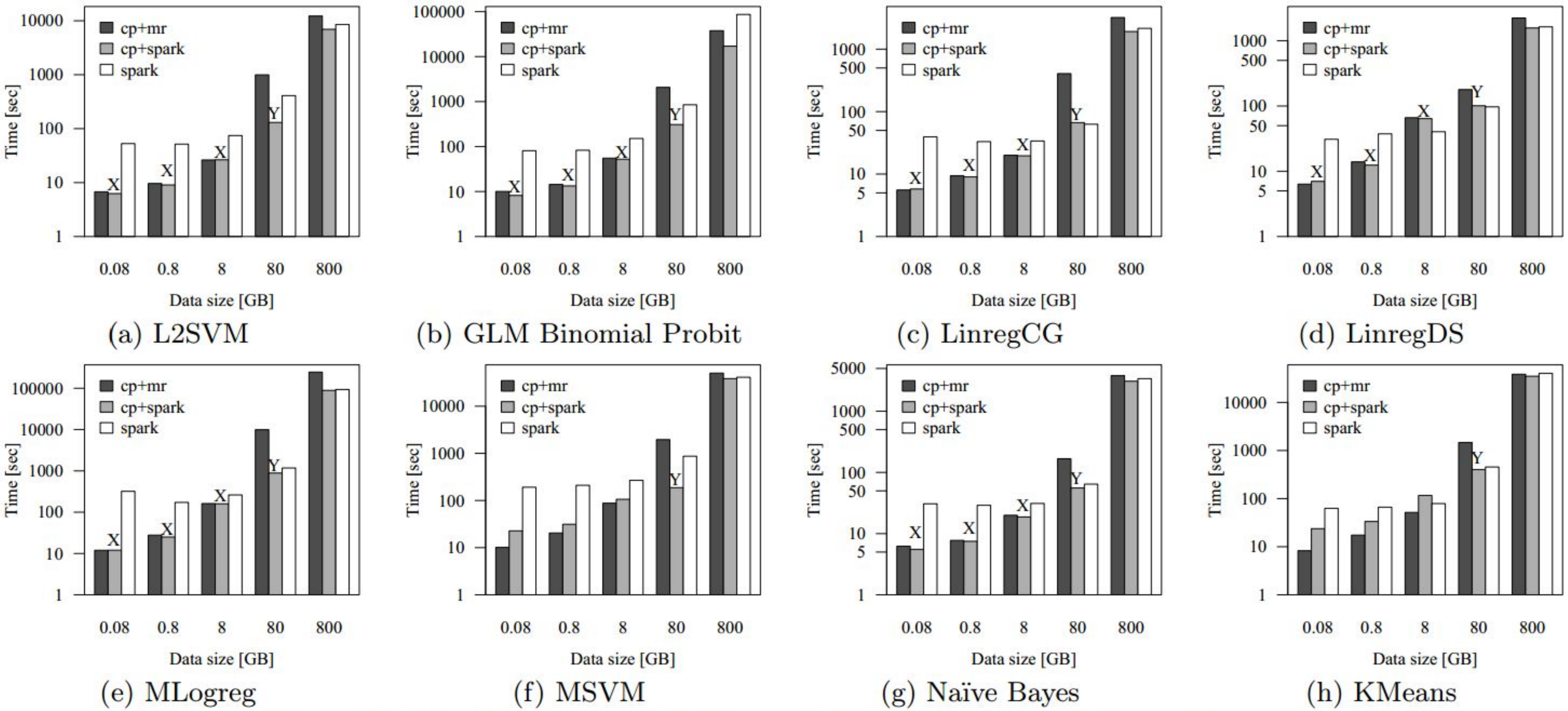


- Dynamic recompilation
  - Adapt the runtime plan to changing or initially unknown data characteristics
- Partitioning Operations
  - Partitioning-Preserving Operations
  - Partitioning-Exploiting Operations
- Specific Runtime Optimizations
  - Lazy Spark-Context Creation
  - Short-Circuit Read
  - Short-Circuit Collect

3 Experiments

3. Experiments

End-to-End Performance

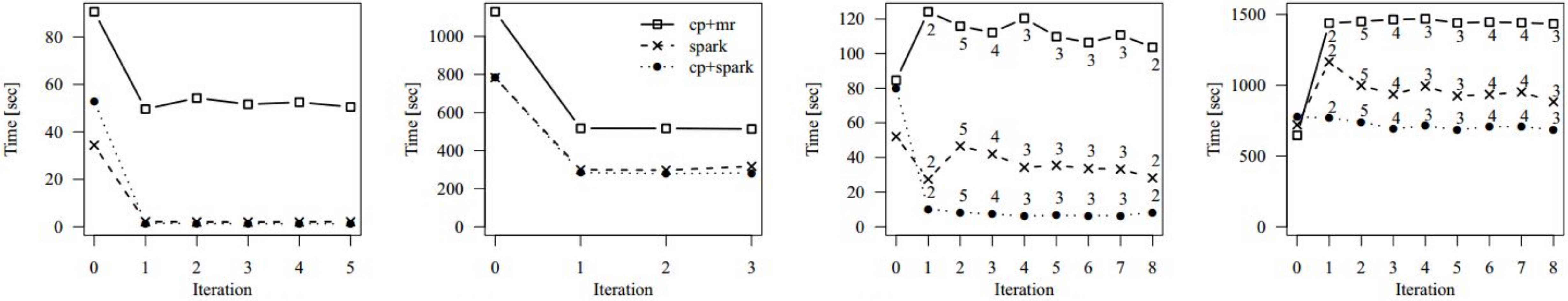


(a) L2SVM (b) GLM Binomial Probit (c) LinregCG (d) LinregDS

(e) MLogreg (f) MSVM (g) Naïve Bayes (h) KMeans

Figure 4: End-to-End Performance of Different Algorithms with Different Execution Modes.

Runtime per Iteration



(a) LinregCG 80GB (b) LinregCG 800GB (c) L2SVM 80GB (d) L2SVM 800GB

Figure 5: Runtime per Iteration of LinregCG and L2SVM with Different Execution Modes.

Conclusions 4

4. Conclusions

Takeaways and paper contributions

- ✓ Importance of DML as a high-level language to improve interoperability and scalability of Machine Learning models on Spark
- ✓ Multiple layers of abstraction and optimizations make SystemML a powerful tool for accelerating the development of Machine Learning models over Big Data
- ✓ Experimental evaluation on multiple ML models and datasets

Thanks for your attention

Discussion 5

5. Discussion

Research

1. Optimizer. How to optimize ML models over data streams?
2. Runtime. In dynamic recompilation, what could be unknown data characteristics?
3. Experiments. How SystemML might perform for the KNN algorithm?

Industry

5. Current capabilities compared to other tools such as Numpy, Scikit Learn, or TensorFlow?
6. Adoption in the current ML and Big Data user base?
7. SystemML in Cloud computing infrastructure. Beyond IBM?