

Samsara: Declarative Machine Learning on Distributed Dataflow Systems

Sebastian Schelter
Technische Universität Berlin
sebastian.schelter@tu-berlin.de

Andrew Palumbo
apalumbo@apache.org

Shannon Quinn
University of Georgia
spq@uga.edu

Suneel Marthi
smarthi@apache.org

Andrew Musselman
akm@apache.org

Overview

Motivation:

- apply ML to **large datasets stored in distributed filesystems** using **dataflow engines** like Spark

Problem:

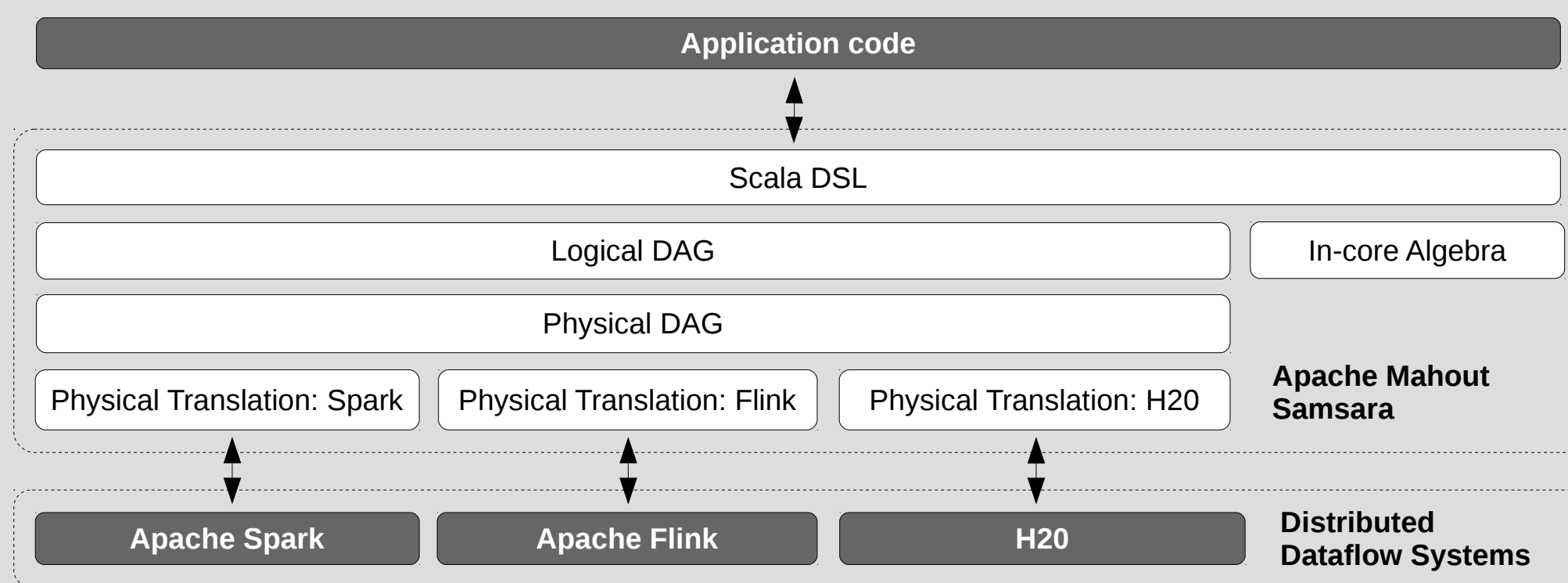
- **dataflow engines are hard to program:**
 - programs consist of **sequence of parallelizable second-order functions** on partitioned data
 - **mismatch for ML applications** which mostly operate on tensor type

Samsara

- **domain-specific language and execution layer for declarative machine learning**
 - allows advanced users to rapidly create new algorithms and adapt existing ones
 - reduces need for knowledge of programming and execution model of underlying systems

Architectural Overview

- programs written in declarative Scala DSL
- compilation to DAG of logical operators
- translation to backend-specific physical operators
- execution on dataflow backend



Compilation of Programs to DAGs of Logical Operators

Example: Distributed Ridge Regression

- assumption: 'tall-and-skinny' input matrix X
- distributed computation of $X^T X$ and $X^T Y$
- solve normal equation $(X^T X + \lambda I)^{-1} X^T Y$ locally afterwards

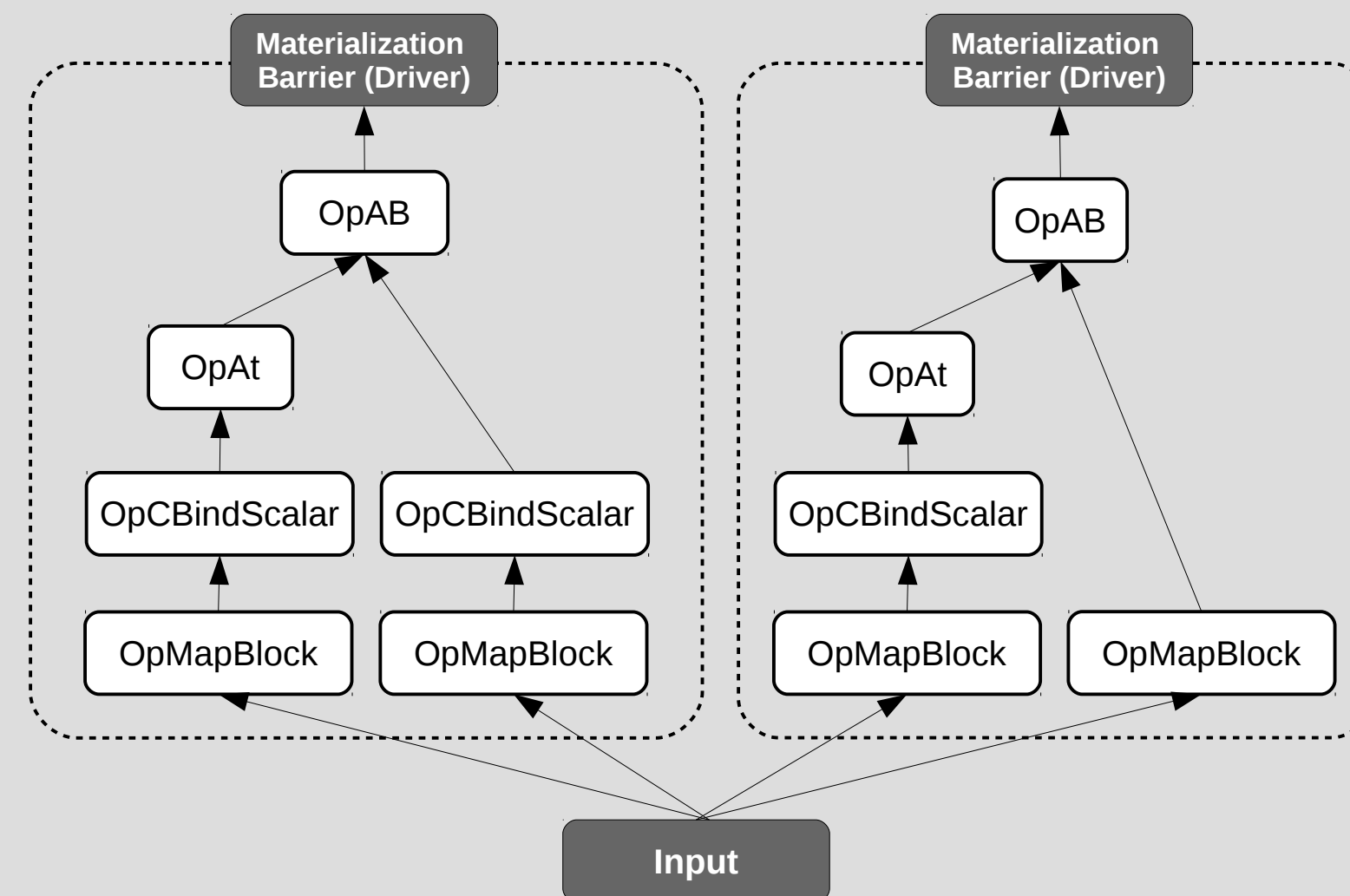
```
def dridge(data: DrmLike[Int], lambda: Double): Matrix {
  // slice out features, add column for bias term
  val drmX = data(:, 0 until data.ncol) cbind 1
  val drmY = data(:, data.ncol)

  val drmXtX = drmX.t %*% drmX // distributed matrix
  val drmXtY = drmX.t %*% drmY // multiplications

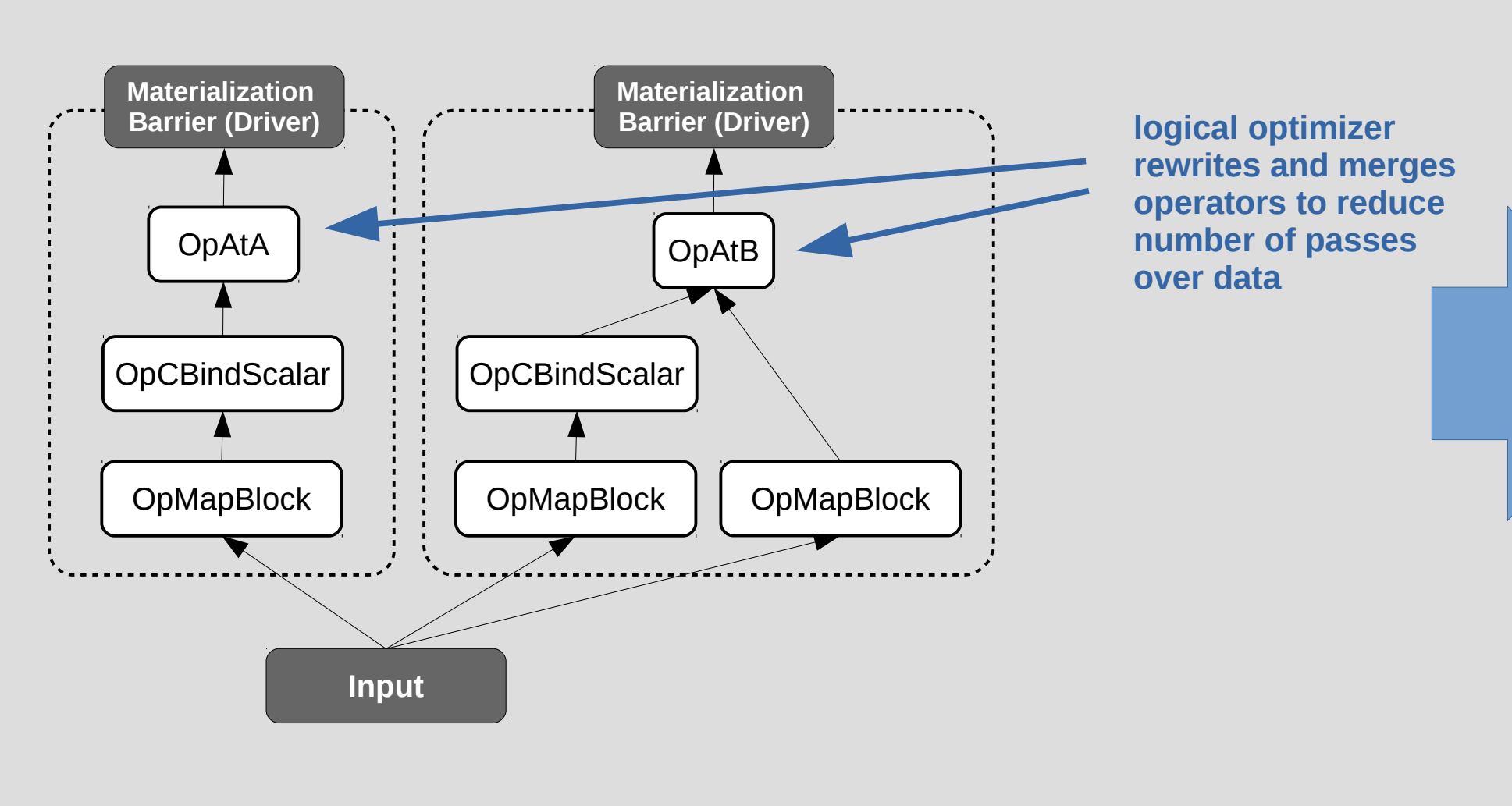
  val XtX = drmXtX.collect // fetch results
  val XtY = drmXtY.collect // into driver memory

  XtX.diag += lambda // add regularization
  solve(XtX, XtY) // compute parameters in-core on driver
}
```

Resulting DAG of Logical Operators



Rewritten DAG after Logical Optimization

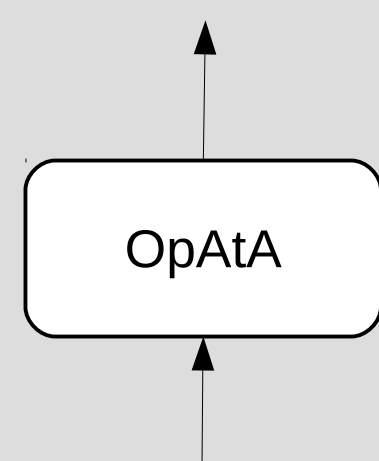


Choice of Physical Operators

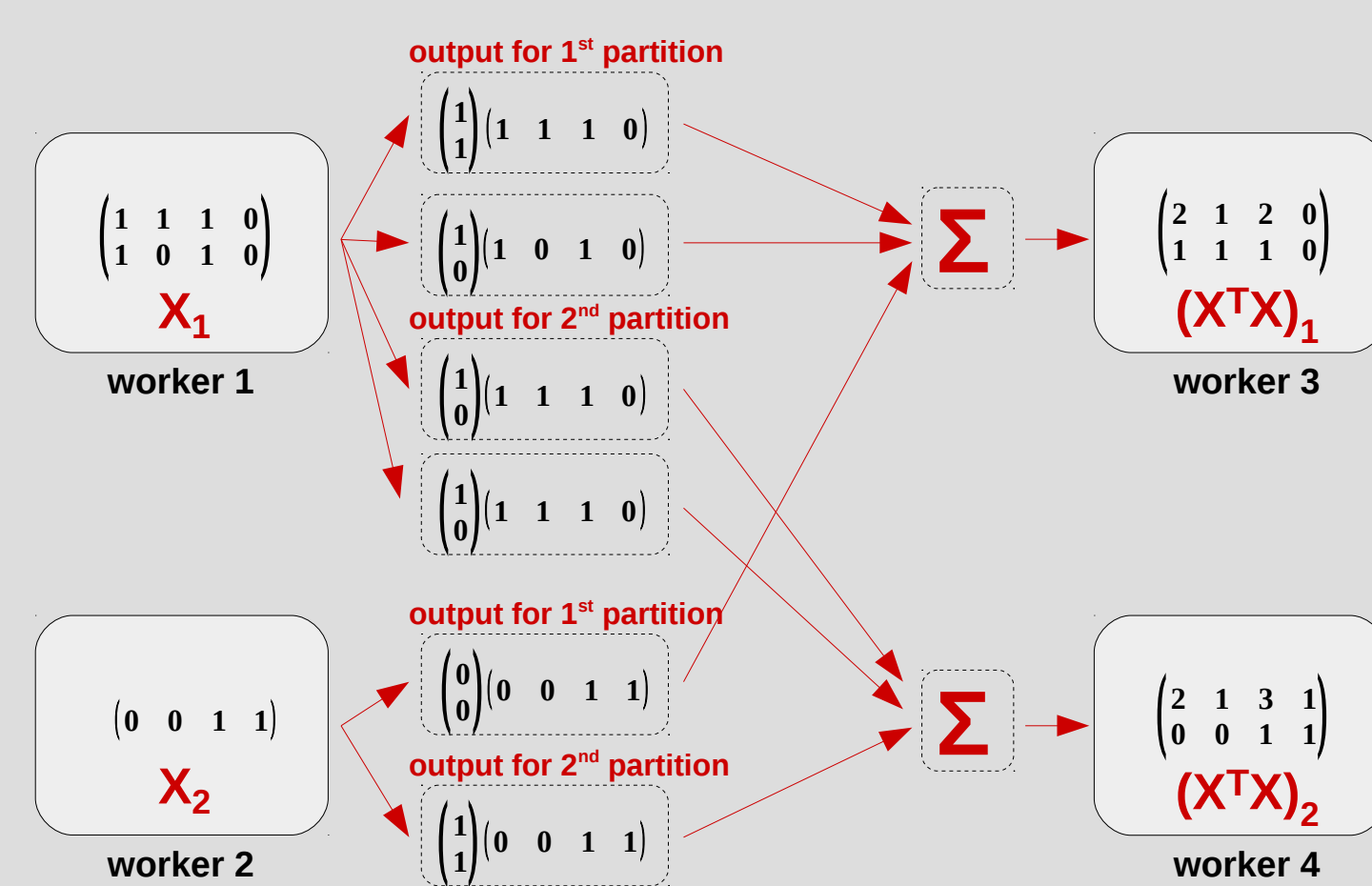
Example: Transpose-Times-Self on Spark

- system chooses physical execution strategy for operators on a given backend
- based on three characteristics of the operands
 - (1) special structure (e.g., diagonal matrices)
 - (2) dimensions (e.g., 'tall-and-skinny')
 - (3) partitioning (e.g., use local instead of distributed joins for co-partitioned inputs)

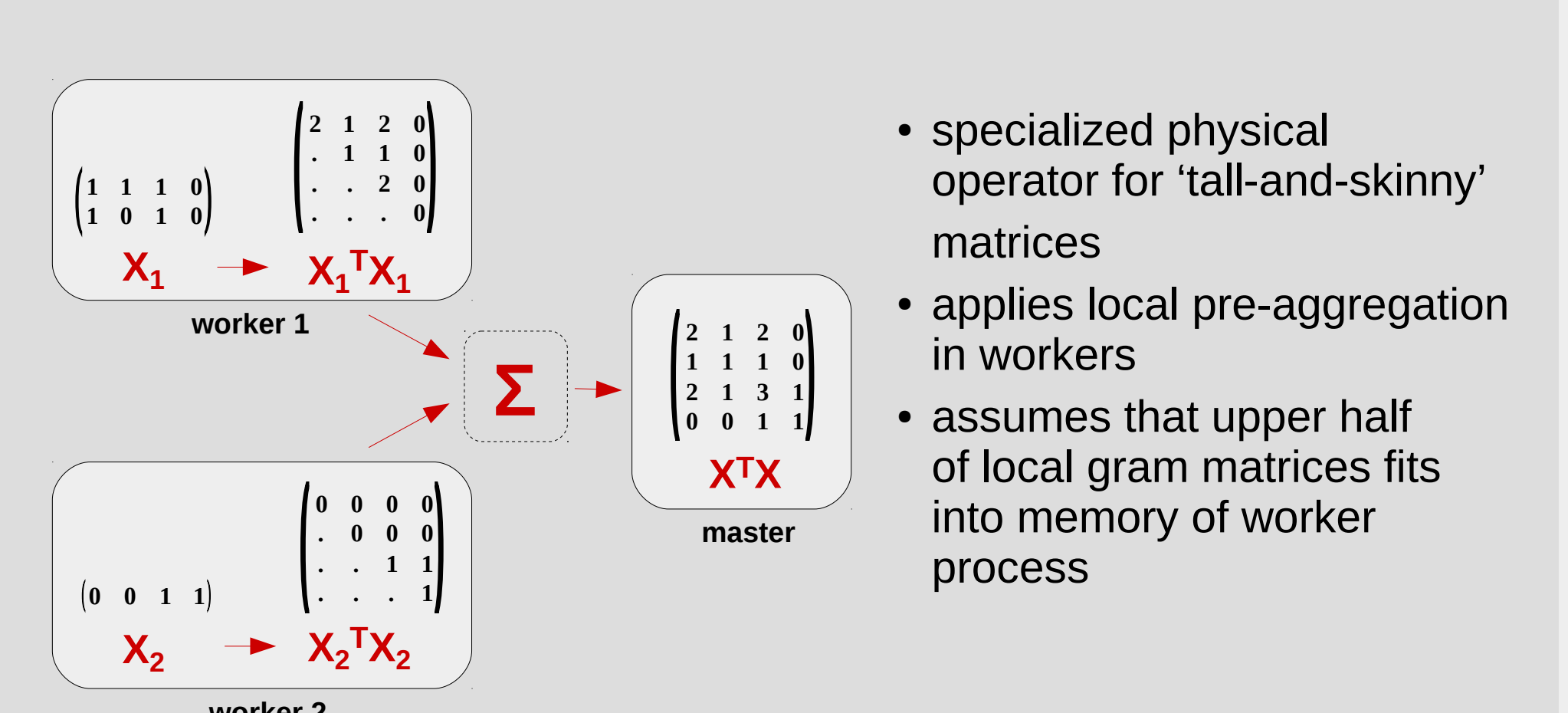
Example: Computation of $X^T X$ in our regression code on Apache Spark (represented by $OpAtA$)



Variant A: Distributed Computation of $X^T X$ via Summation of Partial Outer Products



Variant B: Distributed Computation of $X^T X$ via Summation of Local Gram Matrices

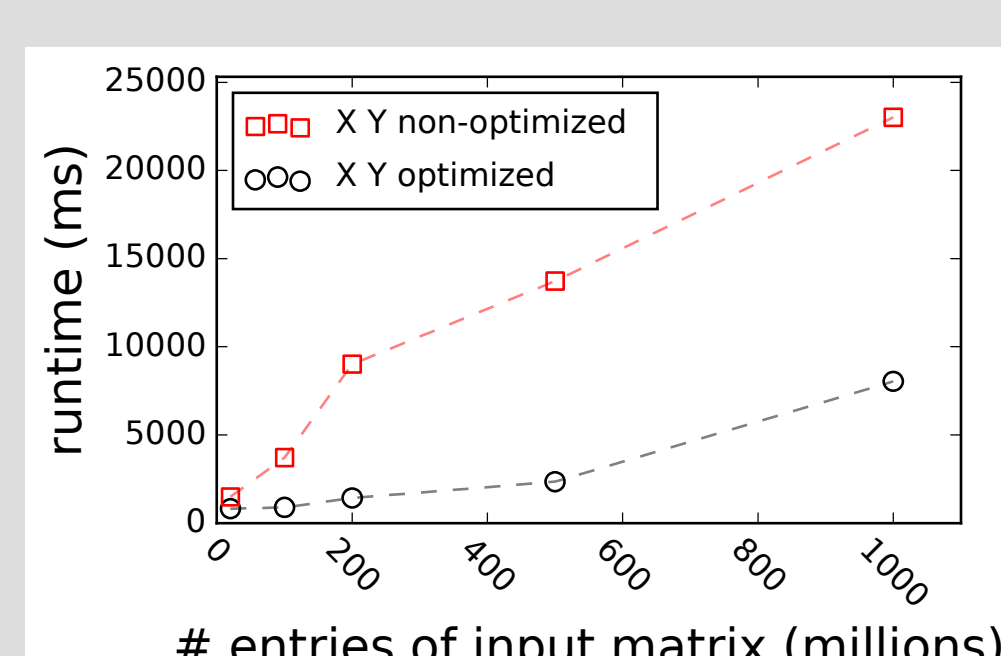
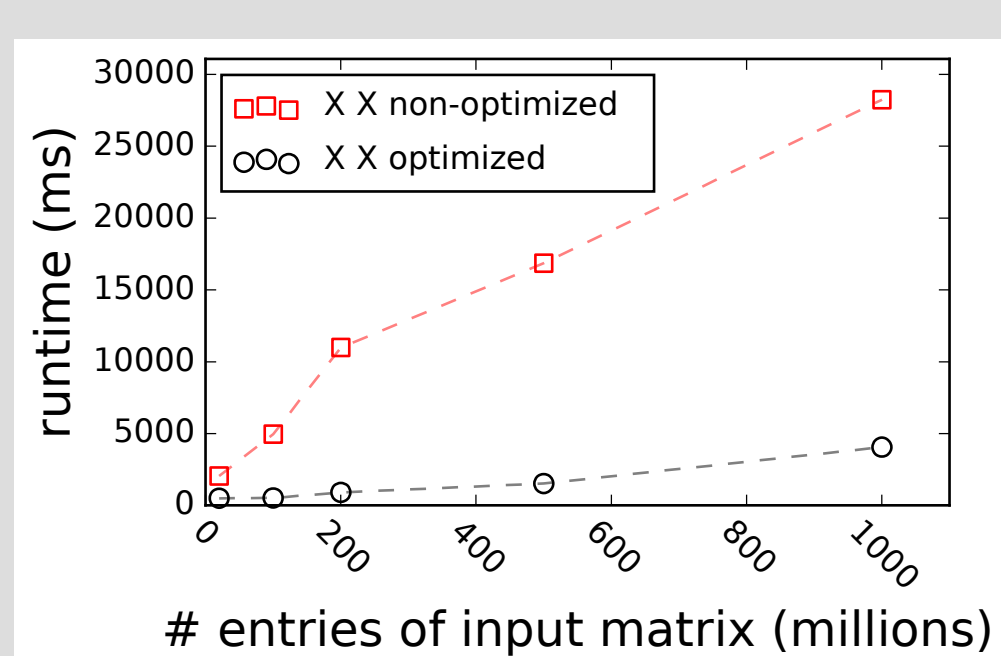


- specialized physical operator for 'tall-and-skinny' matrices
- applies local pre-aggregation in workers
- assumes that upper half of local gram matrices fits into memory of worker process

Evaluation, Limitations, Future Work

Benefit of Proposed Optimizations

- compared standard execution mode of Samsara with a variant that has optimizations disabled
- cluster of 24 machines, synthetically generated matrices with 20 columns and a growing number of rows
- 5x performance improvements due to choice of specialized operator for $X^T X$ and 3x due to choice of local vs distributed joins in $X^T Y$

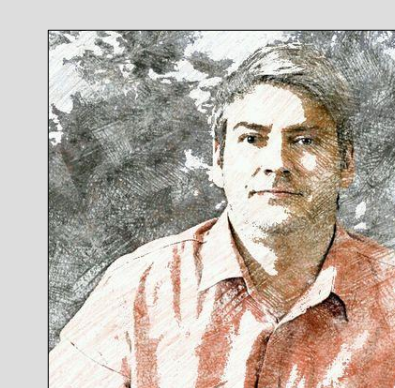


Future Work & References

- **Limitations & Future Work**
 - lack of speed for in-core operations due to JVM-based matrix libraries
 - currently exploring integration of ViennaCL for selected operators to provide a bridge to native performance on many-core architectures
 - high variance in performance between different backends
 - e.g., due to lack of efficient caching of intermediate results in Apache Flink

Acknowledgements

Samsara is the result of a community effort from the Apache Mahout project, with fundamental contributions to design and codebase by Dmitriy Lyubimov



References

- Lyubimov and Palumbo. "Apache Mahout: Beyond MapReduce." (2016)
- "Mahout Scala and Spark Bindings" <http://mahout.apache.org/users/sparkbindings/ScalaSparkBindings.pdf>