On Optimizing Operator Fusion Plans for Large-Scale Machine Learning in SystemML

Matthias Boehm[1], Berthold Reinwald[1], Dylan Hutchison[2], Prithviraj Sen[1], Alexandre Evfimievski[1], Niketan Pansare[1]

VLDB 2018 Session
"DB Techniques for ML"

[1] IBM Research – Almaden; San Jose, CA, USA
[2] University of Washington; Seattle, WA, USA

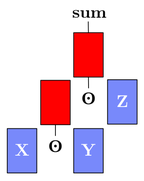
Contact: Matthias Boehm
mboehm7@gmail.com

Motivation/Problem

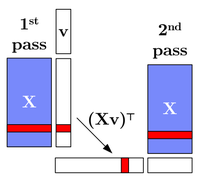
Fusion Opportunities

- **State-of-the art ML Systems**
  - DAGs of linear algebra (LA) operations and statistical functions
  - Materialized intermediates ➔ **ubiquitous fusion opportunities**

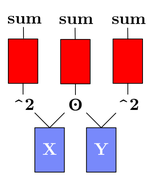
Intermediates



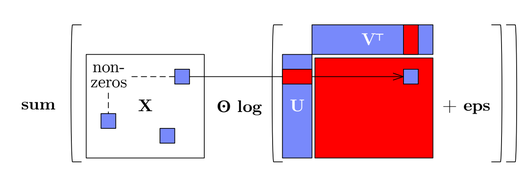
Single-Pass



Multi-Aggregates



Sparsity Exploitation

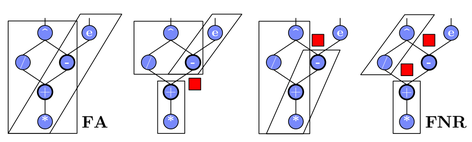


Optimizing Fusion Plans

FNR .. Fuse-No-Redundancy
FA .. Fuse-All

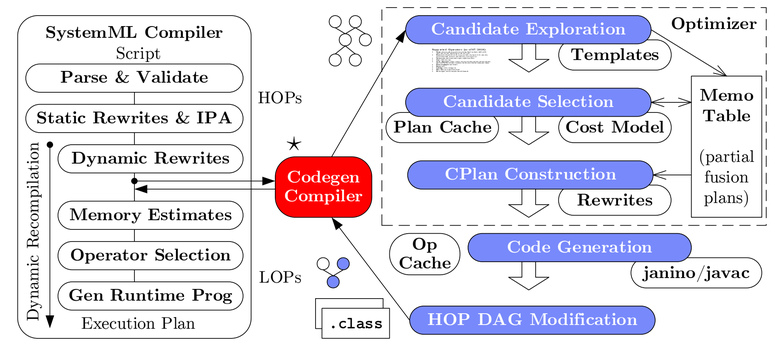
- **Problem:** Fusion heuristics ➔ **poor plans** for complex DAGs (cost/structure), sparsity, and local/distributed ops
- **Materialization Points**
- **Sparsity Exploitation**
- **Fusion Patterns**
- **Constraints** (e.g., memory budget and block sizes)

C = A + s * B
D = (C/2)^(C-1)
E = exp(C-1)



System Architecture

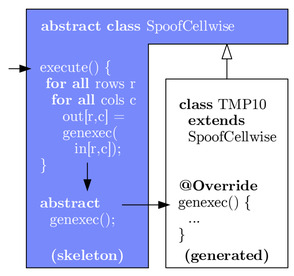
Compiler Overview



Runtime Integration

- **Templates**
  - Cell, MAgg,
  - Row,
  - Outer
- **Template Skeleton**
  - Data access, blocking
  - Multi-threading
  - Final aggregation

Example L2SVM inner loop (2 main roots w/ large CSE)




```
public final class TMP23 extends SpoofMAgg {
  public TMP23() {
    super(false, AggOp.SUM, AggOp.SUM);
  }
  protected void genexec(double a, SideInput[] b,
   double[] scalars, double[] c, ...) {
    double TMP11 = getValue(b[0], rowIndex);
    double TMP12 = getValue(b[1], rowIndex);
    double TMP13 = a * scalars[0];
    double TMP14 = TMP12 + TMP13;
    double TMP15 = TMP11 * TMP14;
    double TMP16 = 1 - TMP15;
    double TMP17 = (TMP16 > 0) ? 1 : 0;
    double TMP18 = a * TMP17;
    double TMP19 = TMP18 * a;
    double TMP20 = TMP16 * TMP17;
    double TMP21 = TMP20 * TMP11;
    double TMP22 = TMP21 * a;
    c[0] += TMP19;
    c[1] += TMP22;
  }
}
```

Candidate Exploration

Memo Table

- **Partial Fusion Plans** (candidates)
- **Memo Table Entry** (type, {i_1, ..., i_k}, closed)

ba .. binary aggregate (matrix multiply)
b .. binary
rix .. right indexing
r(t) .. transpose
ua .. unary aggregate

Open-Fuse-Merge-Close

- **Template Fusion API**
  - **Open** new template
  - **Fuse**/**Merge** open template
  - **Close** open template
- **OFMC Exploration Algorithm**
  - Bottom-up exploration (single-pass, template-agnostic)
  - Linear space and time (O($2^{|g_i|}$ * |T|) per node, but ternary ops / 4 templates)

Candidate Selection

Plan Partitions and Interesting Points

- **Determine Plan Partitions**
  - Materialization points M
  - Connected components of fusion plans
  - Root and input nodes
  - ➔ **Optimize independently**
- **Determine Interesting Points**
  - Materialization point consumers
  - Template / sparse switches
- **Cost Model**
  - Cost partitions with **analytical cost model**
  - Efficient and correct costing via **DAG traversal** and **cost vectors**
  - **Handling of constraints** via prefiltering and infinite/scaled costs

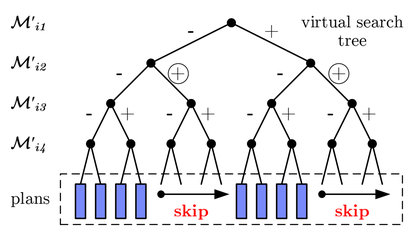
$$C(\mathcal{P}_i|\mathbf{q}) = \sum_{p \in \mathcal{P}_i|\mathbf{q}} \left(\hat{T}^w_p + \max\left(\hat{T}^r_p, \hat{T}^c_p\right)\right).$$

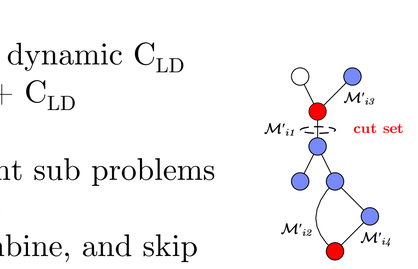
Enumeration Algorithm MPSkipEnum

- **Basic Enumeration**
  - Linearize search space: from - to *
  - Evaluate and cost plans

```
for( j in 1:pow(2,|M'_i|) ) {
  q = createAssignment(j)
  C = getPlanCost(P_i, q)
  maintainBest(q, C)
}
```

- **Cost-Based Pruning**
  - Opening: evaluate FA and FNR heuristics first
  - **Upper bound:** cost C_U of best plan q*
  - **Lower bound:** C_{LS} (min read/write/compute) + dynamic C_{LD} (intermediates q) ➔ **skip subspace** if $C_U \leq C_{LS} + C_{LD}$
- **Structural Pruning**
  - **Observation:** Assignments can create independent sub problems
  - Build **reachability graph** to determine **cut sets**
  - During enum: probe cut sets, recursive enum, combine, and skip





Experiments

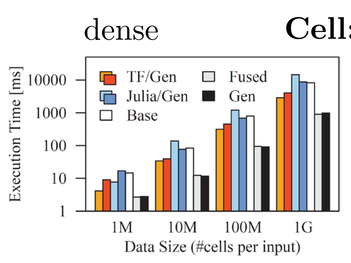
Experimental Setting

Data	Dimensions	Sparsity
Airline78	14,462,943 x 29	0.73
Mnist8m	8,100,000 x 784	0.25
Netflix	480,189 x 17,770	0.012
Amazon	8,026,324 x 2,330,066	0.0000012

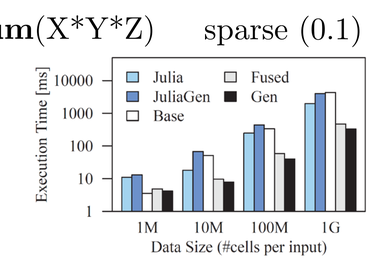
- **Cluster setup**
  - 1+6 node cluster (head 2x4 Intel Xeon E5530, 64GB RAM; 6 workers 2x6 Intel Xeon E5-2440, 96GB RAM, peak 2x32GB/s 2x115GFLOP/s, 10Gb Ethn)
  - Spark 2.2, 6 executors (24 cores, 65GB), 35GB driver
- **Baselines**
  - **SystemML 1.0++** (Feb'18): **Base**, **Fused***, **Gen** (opt), heuristics: **FA, FNR**
  - **Julia 0.6.2** (Dec'17): **Julia** (w/o fusion), **JuliaGen** (fusion via dot syntax)
  - **TensorFlow 1.5** (Jan'18): **TF** (w/o fusion), and **TFGen** (fusion via XLA)

Operations Performance

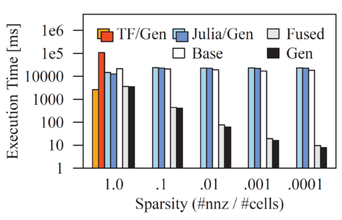
Cell: sum(X*Y*Z) dense



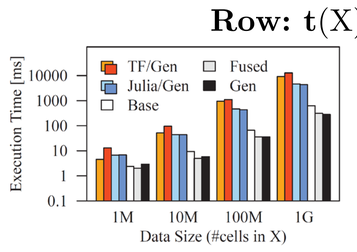
Cell: sum(X*Y*Z) sparse (0.1)

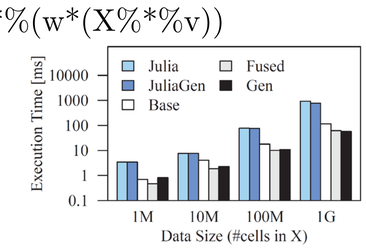


Outer: sum(X* **log**(U%*%**t**(V)+1e-15))

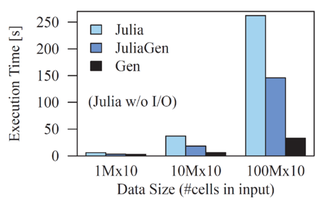


Row: t(X)%*%(w*(X%*%v))





L2SVM End-to-End



ALS-CG End-to-End

	Data	Base	Fused	Gen	FA	FNR
local	10^8 x 10, D	446	276	**37**	44	92
	Airline78, D	151	105	**24**	26	45
	Mnist8m, S	203	156	**113**	115	116
	$2*10^8$ x 100, D	1218	895	**347**	1433	539
	$2*10^8$ x 10^3, S	1481	1066	**373**	2205	575
distr.	Mnist80m, S	1593	1114	**552**	1312	896

	Data	Base	Fused	Gen	FA	FNR
local	10^4 x 10^4, S (0.01)	426	**20**	25	215	226
	10^5 x 10^5, S (0.01)	23,585	96	**80**	13,511	12,353
	10^6 x 10^6, S (0.01)	N/A	860	**722**	N/A	N/A
	Netflix	N/A	1,026	**789**	N/A	N/A
	Amazon	N/A	17,335	**7,420**	N/A	N/A