A Minimalist Kernel for Linear and Relational Algebra

Dylan Hutchison, Bill Howe, Dan Suciu

BeyondMR @SIGMOD, 19 May 2017

Contact: dhutchis@cs.washington.edu

UNIVERSITY of WASHINGTON

Related Work

2017

Scalable Linear Algebra on a Relational Database System

Shangyu Luo*, Zekai J. Gao*, Michael Gubanov†, Luis L. Perez‡, Christopher Jermaine*

*Rice University, {sl45, jacobgao, cmj4}@rice.edu, ‡Rice University, lperezp@gmail.com

The BUDS Language for Distributed Bayesian Machine Learning

Zekai J. Gao, Shangyu Luo, Luis L. Perez, Chris Jermaine

Rice University

Towards Linear Algebra over Normalized Data

Lingjiao Chen[1] Arun Kumar[2] Jeffrey Naughton[3] Jignesh M. Patel[1]

[1]University of Wisconsin-Madison [2]University of California, San Diego [3]Google

2017

LARADB

LaraDB: A Minimalist Kernel for Linear and Relational Algebra Computation

Dylan Hutchison, Bill Howe, Dan Suciu

- Sensor Query on *Array of Things* data
- Logical + Physical Algebra
- Properties / Optimizations
- LaraDB Implementation
- Experiments

- Extend SQL with LA types, UDFs on *SimSQL*
- Compile MATLAB-like LA to SQL in *BUDS*
- *Morpheus*: Push LA into matrix multiply for ML

LARA: A Key-Value Algebra underlying Arrays and Relations

Dylan Hutchison, Bill Howe, Dan Suciu

University of Washington

Draft: April 1, 2016

Bridging the Gap: Towards Optimization Across Linear and Relational Algebra

Andreas Kunft Alexander Alexandrov Asterios Katsifodimos Volker Markl

TU Berlin

first.last@tu-berlin.de

2016

- Logical Algebra
- Properties / Optimizations
- Translations

- Vision paper
- Hypothetical Sensor Query
- Properties / Optimizations

Motivating Use Case: Mean & Covariance of sensor differences

Array of Things Sensor Data

t	c	v
466	temp	55.2
466	hum	40.1
492	temp	56.3
492	hum	35.0
528	temp	56.5

Collected in CSV files

Filter, bin onto common time buckets

Filter, bin onto common time buckets

Subtract

Compute Mean

Compute Covariance

Preprocessing (easier to express in RA)

Impedance Mismatch?

Analysis (easier to express in LA)

https://arrayofthings.github.io/

Bin query: easy in RA, awkward in LA

t	c	v
466	temp	55.2
466	hum	40.1
492	temp	**56.3**
492	hum	35.0
528	temp	**56.5**

RA

SELECT bin(**t**) AS **t'**, **c**, avg(**v**) AS **v**
GROUP BY **t'**, **c**

t'	c	v
460	temp	55.2
460	hum	40.1
520	temp	**56.4**
520	hum	35.0

LA

Multiply:

$$\begin{array}{c} \\ 460 \\ 520 \end{array}\begin{array}{c} \begin{array}{ccc} 466 & 492 & 528 \end{array} \\ \begin{bmatrix} 1 & & \\ & 1 & 1 \end{bmatrix} \end{array} * \begin{array}{c} \begin{array}{cc} temp & hum \end{array} \\ \begin{array}{c} 466 \\ 492 \\ 528 \end{array}\begin{bmatrix} 55.2 & 40.1 \\ \mathbf{56.3} & 35.0 \\ \mathbf{56.5} & \end{bmatrix} \end{array} = \begin{array}{c} \begin{array}{cc} temp & hum \end{array} \\ \begin{array}{c} 460 \\ 520 \end{array}\begin{bmatrix} 55.2 & 40.1 \\ \mathbf{56.4} & 35.0 \end{bmatrix} \end{array}$$

using *avg* on added elements

$$\text{bin}(t) = t - (t \,\%\, 60) + 60\left\lfloor \frac{t \,\%\, 60}{60} + .5 \right\rfloor$$

Covariance query: easy in LA, awkward in RA

$$X = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix}$$

LA

N = size(**X**, 1);

M = mean(**X**, 1);

C = **X**'***X** / **N** – **M**'***M**;

RA

(Generated SQL statements for each entry)

T = SELECT FROM **X** sum(1.0) AS N,

sum(X_1) AS M_1, sum(X_2) AS M_2, …, sum(X_d) AS M_d,

sum(X_1*X_1) AS Q_{11}, sum(X_1*X_2) AS Q_{12}, …,

sum(X_{d-1}*X_d) AS $Q_{(d-1)d}$, sum(X_d*X_d) AS Q_{dd}

C = SELECT FROM **T**

(1 AS i, 1 AS j, $Q_{11}/N - M_1*M_1$ AS v) UNION

(1 AS i, 2 AS j, $Q_{12}/N - M_1*M_2$ AS v) UNION

…

Carlos Ordonez.
Building Statistical Models and Scoring with UDFs. SIGMOD 2007.

Properties, easier in LA or RA

- $(A^TA)^T = A^TA$ [inner product is symmetric]
  - RA version is verbose (renames, joins, aggregates)
  - Impact: eliminate half of computation

- $A \bowtie_i B = \sigma_{i=i}(A \times B)$ [cross product to join]
  - LA version is verbose (tensor products and contractions)
  - Impact: eliminate large intermediates

Is a common algebra possible?

Evidence:

- Overlapping expressiveness
  - Matrix Multiply = GroupBy ○ Apply ○ Join

- Emergence of hybrid systems
  - Spark, Myria, MADlib, SystemML, Weld, …
  - Systems with RA and LA interfaces or data models

Roadmap

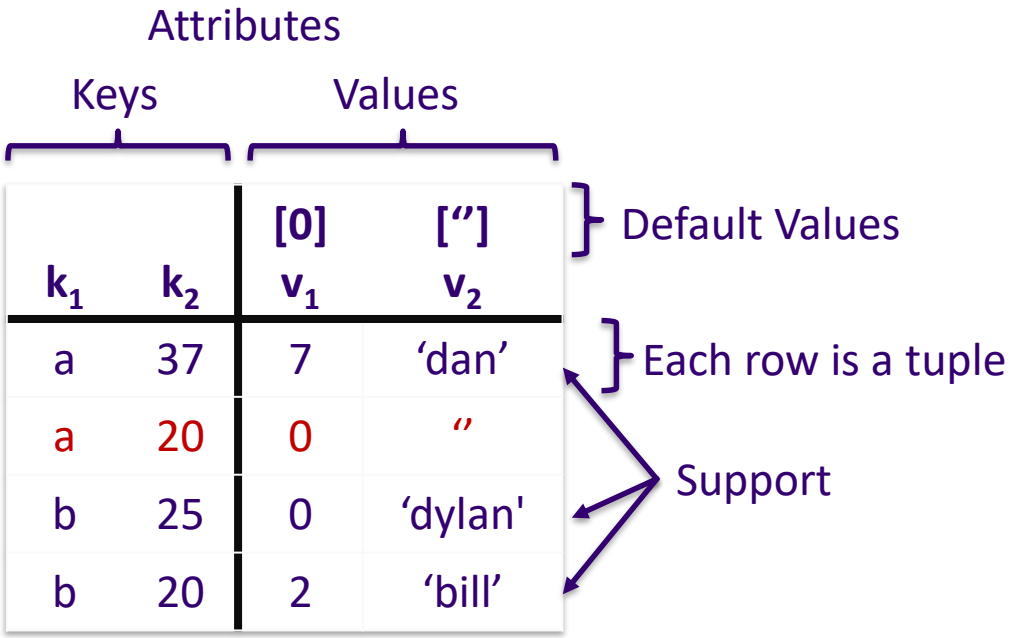
- Motivation: Sensor query
  - Sometimes RA is better, sometimes LA
- **LARA Algebra**
  - **3 Operators & Properties**
- LARADB on *Sorted Distributed Maps*
  - Physical operators & optimizations
- Experiments
  - Optimizations: LARA affords impactful optimizations
  - Performance: LARA affords competitive performance

LARA

Objects:

Associative Tables

Total functions from keys to values with finite support



Operators:

Parameterized by UDFs: $\otimes$, $\oplus$, f

Join

$\bowtie_{\otimes}$

"join keys, multiply values"

Union

$\Sigma_{\oplus}$

"group by keys, add values"

Extension

ext_f

"flatmap"

Join and Union adapted from:
M. Spight and V. Tropashko.
First steps in relational lattice. 2006.

Ext is similar to monadic bind, without deleting keys

Join with ⊗: Join keys, multiply values

a	c	[0] x	[0] z
a_1	c_1	11	1
a_1	c_2	12	2
a_2	c_1	13	3
a_3	c_3	14	4

$\bowtie_{\otimes}$

c	b	[0] z	[0] y
c_1	b_1	5	15
c_2	b_1	6	16
c_2	b_2	7	17
c_4	b_1	8	18

=

a	c	b	[0 ⊗ 0 = 0] z
a_1	c_1	b_1	1 ⊗ 5
a_1	c_2	b_1	3 ⊗ 5
a_1	c_2	b_2	2 ⊗ 6
(a_3	c_3	b_1	4 ⊗ 0 = 0)

$$A \bowtie_{\otimes} B : a, c, b \rightarrow z : 0^z \otimes 0^z$$

$$(A \bowtie_{\otimes} B)(a, c, b) := [z : \pi_z A(a, c) \otimes \pi_z B(c, b)]$$

Requires (see paper for general case):

$v_A \otimes 0 = 0 \otimes v_B = 0 \otimes 0 = 0$

Union with $\oplus$: Group by keys, sum values

a	c	[0] x	[0] z
a_1	c_1	11	1
a_1	c_2	12	2
a_2	c_1	13	3
a_3	c_3	14	4

$\bowtie_{\oplus}$

c	b	[0] z	[0] y
c_1	b_1	5	15
c_2	b_1	6	16
c_2	b_2	7	17
c_4	b_1	8	18

=

c	[0] x	[0 ⊕ 0 = 0] z	[0] y
c_1	11 ⊕ 13	1 ⊕ 3 ⊕ 5	15
c_2	12	2 ⊕ 6 ⊕ 7	16 ⊕ 17
c_3	14	4	0
c_4	0	8	18

$$A \bowtie_{\oplus} B : c \to x, z, y : 0^x, 0^z, 0^y$$

$$(A \bowtie_{\oplus} B)(c) := [x: \bigoplus_a \pi_x A(a,c),$$

$$z: \bigoplus_a \pi_z A(a,c) \oplus \bigoplus_b \pi_z B(c,y),\ y: \bigoplus_b \pi_x B(c,y)]$$

Requires:

$v \oplus 0 = 0 \oplus v = v$

Ext with f: Flatmap

docid	["] text
d_1	she sells sea shells by the sea shore
d_2	the shells she sells are surely seashells

ext_f

```
f(docid, text) =
  text.split(' ')
  .groupBy(word)
```

Produces new key 'word'

docid	word	[0] value
d_1	she	1
d_1	sells	1
d_1	sea	1
d_1	shells	1
d_1	by	1
d_1	the	1
d_1	shore	1
d_2	the	1
d_2	shells	1
d_2	she	1
d_2	sells	1
d_2	are	1
…	…	…

$$\text{ext}_f\ A : a, c, k' \rightarrow v' : 0'$$

$$(\text{ext}_f\ A)(a, c, k') := f(a, c, A(a, c))(k')$$

Requires that *f* behave same on default value input

Summary: Union, Join, Ext

	Key Types	Value Types	Support
Union ($A \between_{\oplus} B$)	$= K_A \cap K_B$	$= V_A \cup V_B$	$\subseteq S_A \cup S_B$
Join ($A \bowtie_{\otimes} B$)	$= K_A \cup K_B$	$= V_A \cap V_B$	$\subseteq S_A \cap S_B$
Ext ($\text{ext}_f A$)	extended by f	set by f	$\subseteq S_A \times S_f$

Duality

For Support, '$\subseteq$' becomes '=' if $\oplus$ is zero-sum-free or $\otimes$ has zero-product-property (similar for f)

LARA Properties

- If $\oplus$ or $\otimes$ is *associative*, *commutative*, or *idempotent*,
  - then so is Union or Join
- If $\otimes$ *distributes* over $\oplus$,
  - Push Union into Join
  - Distribute Join over Union

- See paper for more properties:

$$
\begin{aligned}
& \mathrm{tr}(ABC) \\
= \; & \Join_{+}\mathrm{ext}_{i=l}(ABC) && \text{tr defn.} \\
= \; & \Join_{+}\mathrm{ext}_{i=l}\,\Join_{+}^{\mathrm{i,l}}(\Join_{+}^{\mathrm{i,k}}(A_{ij}\bowtie_{\otimes}B_{jk})\bowtie_{\otimes}C_{kl}) && ABC \text{ defn.} \\
= \; & \Join_{+}\Join_{+}^{\mathrm{i,l}}\mathrm{ext}_{i=l}(\Join_{+}^{\mathrm{i,k}}(A_{ij}\bowtie_{\otimes}B_{jk})\bowtie_{\otimes}C_{kl}) && \text{push ext into } \Join \\
= \; & \Join_{+}\mathrm{ext}_{i=l}(\Join_{+}^{\mathrm{i,k}}(A_{ij}\bowtie_{\otimes}B_{jk})\bowtie_{\otimes}C_{kl}) && \text{combine } \Join \\
= \; & \Join_{+}(\Join_{+}^{\mathrm{i,k}}(A_{ij}\bowtie_{\otimes}B_{jk})\bowtie_{\otimes}C_{ki}) && \text{apply ext} \\
= \; & \Join_{+}\Join_{+}^{\mathrm{i,k}}(A_{ij}\bowtie_{\otimes}B_{jk}\bowtie_{\otimes}C_{ki}) && \text{distr. } \bowtie \text{ into } \Join \\
= \; & \Join_{+}(A_{ij}\bowtie_{\otimes}B_{jk}\bowtie_{\otimes}C_{ki}) && \text{combine } \Join \\
= \; & \Join_{+}(B_{jk}\bowtie_{\otimes}C_{ki}\bowtie_{\otimes}A_{ij}) && \text{commute } \bowtie_{\otimes} \\
= \; & \ldots \;\; \textit{// reversing the above steps} \\
= \; & \mathrm{tr}(BCA)
\end{aligned}
$$

Sensor Query: SQL/MATLAB to Lara

Filter, bin onto common time buckets

Filter, bin onto common time buckets

Subtract

Compute Mean

Compute Covariance

```
A, B = LOAD('s1.csv'), LOAD('s2.csv')
A' = SELECT bin(t) AS t', c, avg(v) AS v
     FROM A WHERE 460 <= t <= 860
     GROUP BY t', c
B' = ...      // repeat above
// change perspective from RA to LA
X = A - B;
N = size(X, 1);
M = mean(X, 1);
disp(M);
U = X - repmat(M, N, 1);
C = U.' * U ./ (N - 1);
disp(C);
```

Ext

Union

Ext

Join

Ext, Union

Join

Rename (Ext), Join, Union

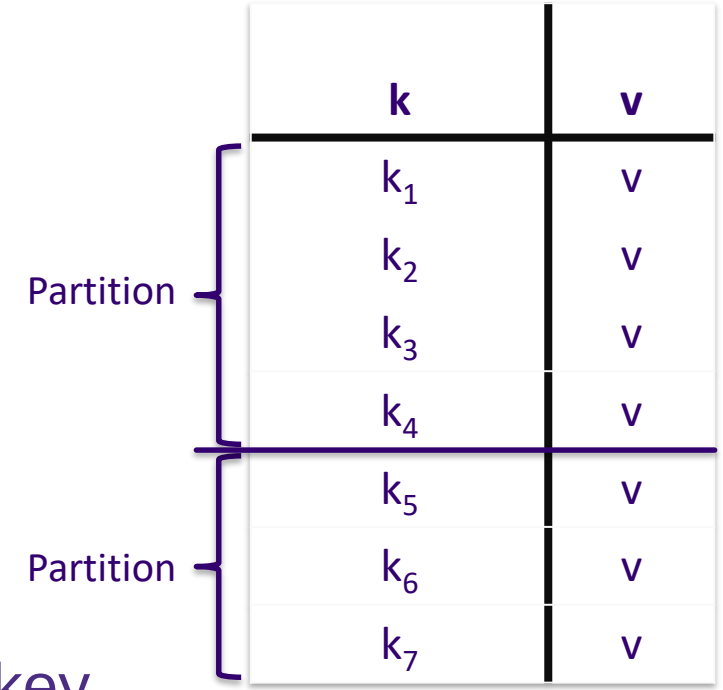
Roadmap

- Motivation: Sensor query
  - Sometimes RA is better, sometimes LA
- LARA Algebra
  - 3 Operators & Properties
- **LARADB on *Sorted Distributed Maps***
  - **Physical operators & optimizations**
- Experiments
  - Optimizations: LARA affords impactful optimizations
  - Performance: LARA affords competitive performance

Implementing LARA on relevant systems with **Sorted Partitioned Maps**

An abstraction for:

> Relational systems
  – with primary key
> NoSQL key-value systems
> Array systems



Map: O(1) access by key

Sorted: O(1) access to "next" element

Partitioned: Scale-out

Physical LARA on Sorted Partitioned Maps

Sorted Associative Table:
Ordered key attributes **[t, c]**

Physical Operators:

- MergeJoin
- MergeUnion
- Ext
- **Sort**
- (Load)
- (Store)

} Physical Optimizations

	t	c	[⊥] v
Partition	440	temp	54.0
	440	hum	40.8
	466	temp	55.2
	466	hum	40.1
Partition	492	temp	56.3
	492	hum	35.0
	528	temp	56.5

Accumulo Row | Column Qualifier | Value

LARADB prototype in Accumulo on Graphulo iterators

Roadmap

- Motivation: Sensor query
  - Sometimes RA is better, sometimes LA
- LARA Algebra
  - 3 Operators & Properties
- LARADB on *Sorted Distributed Maps*
  - Physical operators & optimizations
- **Experiments**
  - **Optimizations: LARA affords impactful optimizations**
  - **Performance: LARA affords competitive performance**

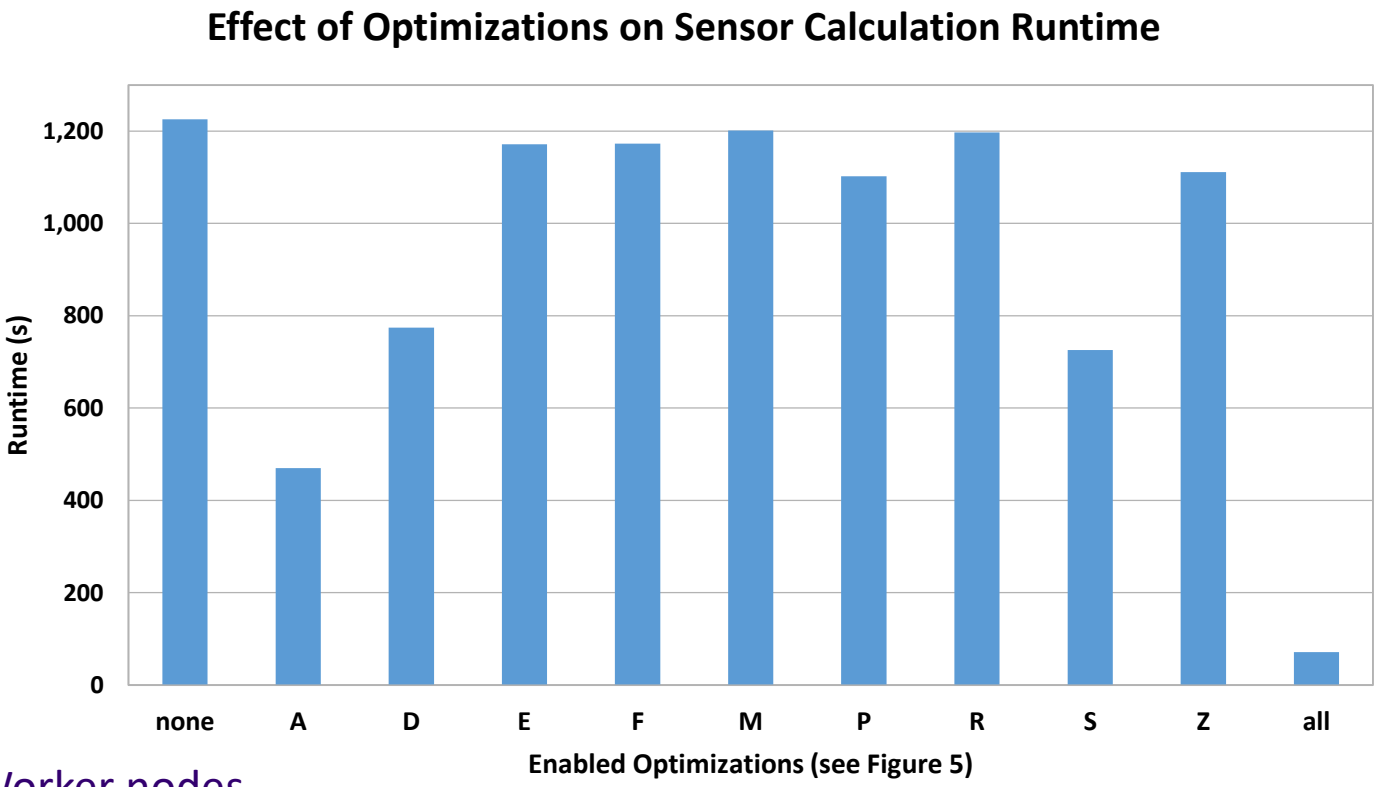
Optimization Experiment

Does LARA afford optimizations that impact performance?

Approach: Benchmark Sensor query on LARADB. Implement optimizations. Plot runtime reduction.

Cluster: Amazon EC2 m3.large
(2 vCores, 7.5 GB mem, SSD per node)

Optimization Experiment



- 4 Worker nodes
- 1.5 months' sensor data
  - Partitioned in 3 day segments
  - 1.2 GB raw; 60 MB after filter to 1 month, project, compress in HDFS

Performance Experiment

Does LARA afford competitive performance when implemented?

Approach: Benchmark Matrix Multiply on data in Accumulo

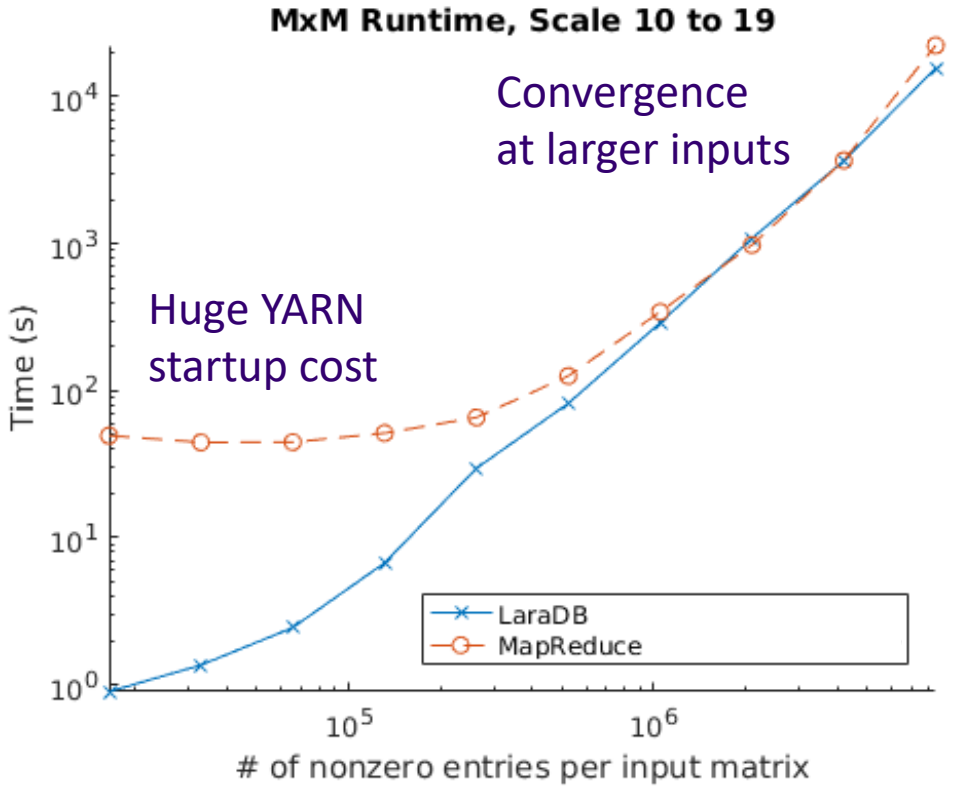
Engines: LARADB and Hadoop MapReduce

Task: Matrix-Matrix Multiply (Graph500 power law matrices)

- MxM is a building block of both LA and RA (join, aggregate)

Cluster: Amazon EC2 m3.large
(2 vCores, 7.5 GB mem, SSD per node)

Performance Experiment



- 8 Worker nodes
- 16×2^{SCALE} entries, generated from Graph500, partitioned evenly on input rows
- Scale 19 MxM forms 8×10^9 partial products, skewed due to power law

Recap

LARA

$\oplus \bowtie \otimes$

- Motivation: Sensor query
  - Sometimes RA is better, sometimes LA
- LARA Algebra
  - 3 Operators & Properties
- LARADB on *Sorted Distributed Maps*
  - Physical operators & optimizations
- Experiments
  - Optimizations: LARA affords impactful optimizations
  - Performance: LARA affords competitive performance

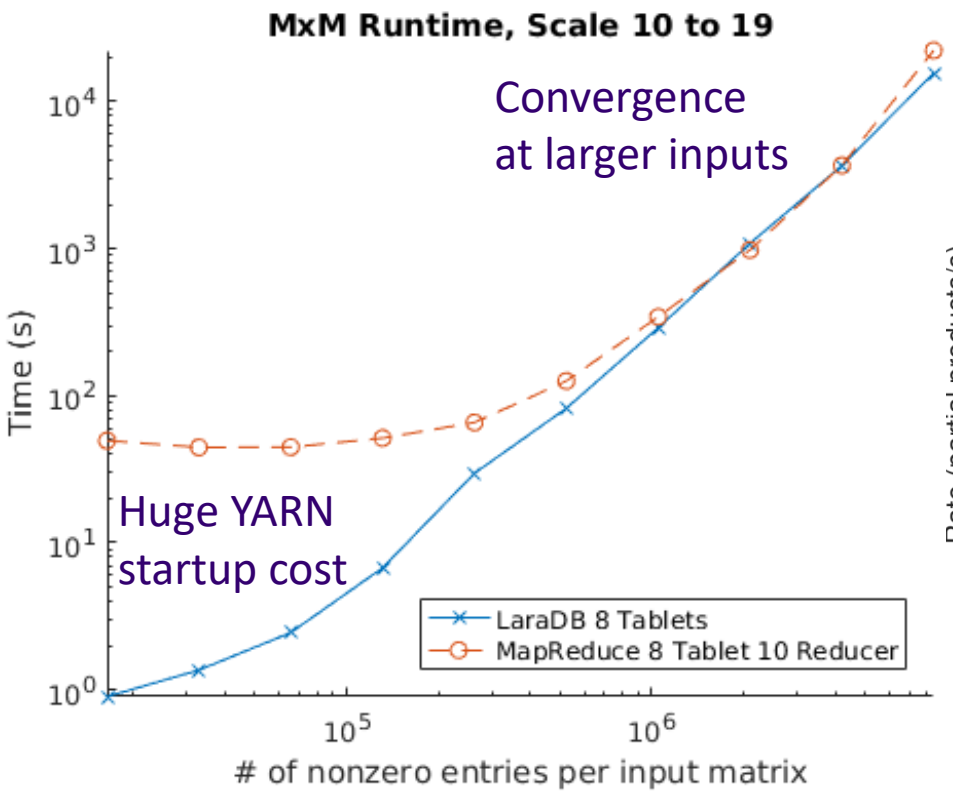
Join $\bowtie_{\otimes}$ ***Union*** $\Sigma_{\oplus}$

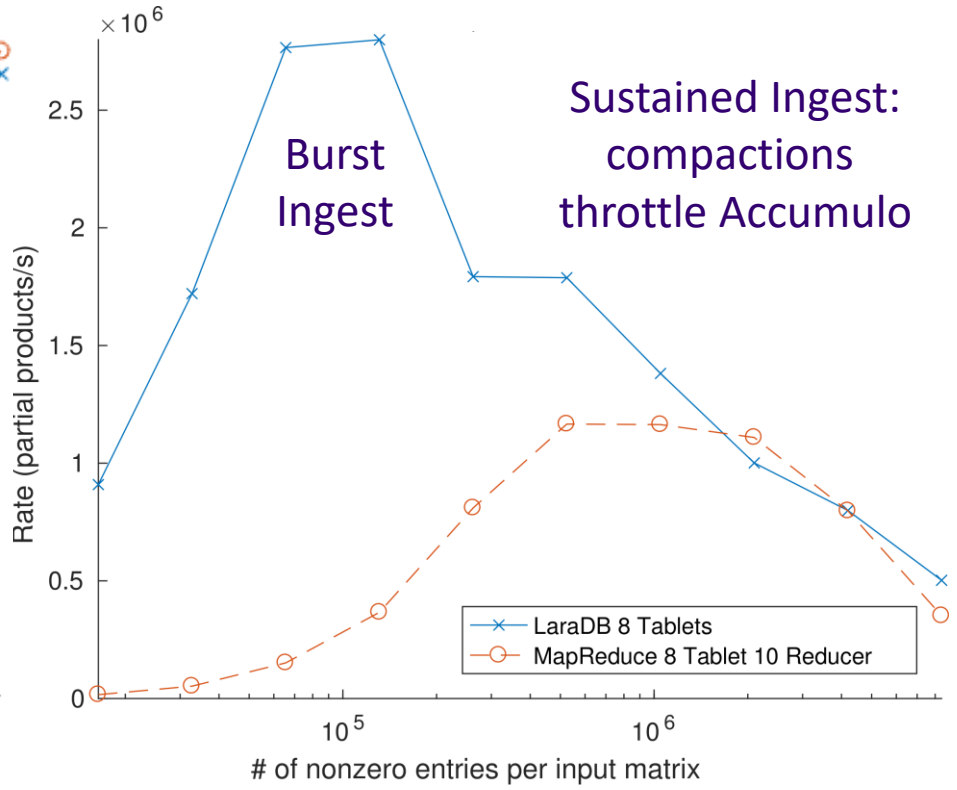
Extension ext_f

Additional thanks to
David Maier, Tim Mattson,
Jeremy Kepner, Vijay Gadepally

Backup Slides

Competitiveness Experiment





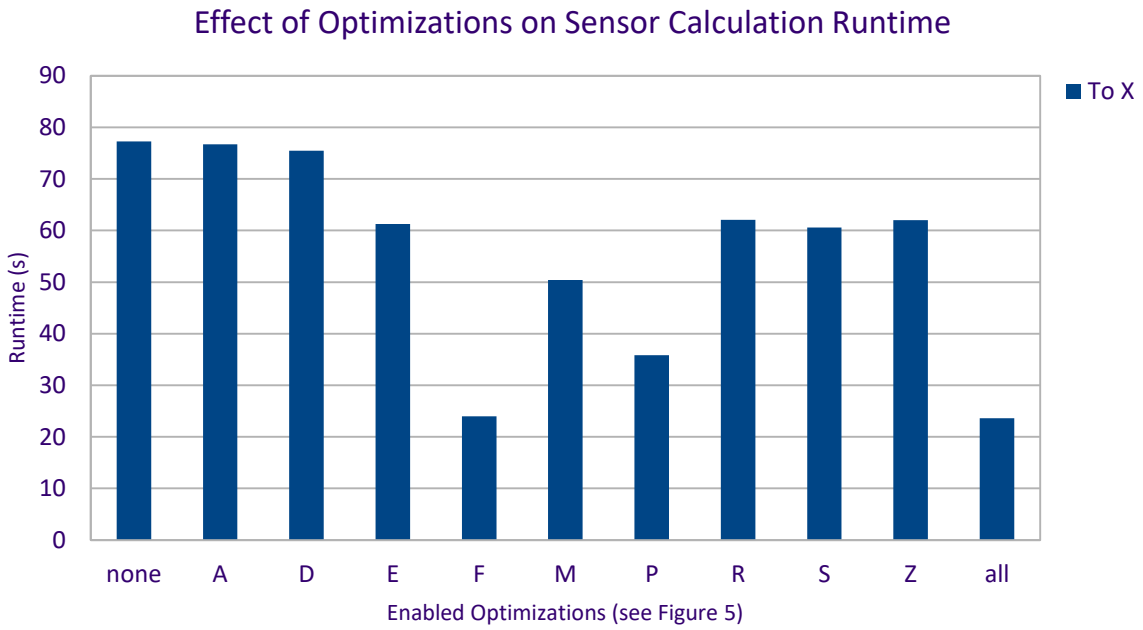
- 8 Worker nodes
- 16×2^{SCALE} entries, generated from Graph500, partitioned evenly on input rows
- Scale 19 MxM forms 8×10^9 partial products, skewed due to power law

Data for Competitiveness Experiment

Scale	Entries in Table **C**		LaraDB		MapReduce 10 Reducers	
	PartialProducts	AfterSum	Time (s)	Rate (pp/s)	Time (s)	Rate (pp/s)
10	8.16×10^5	2.69×10^5	8.98×10^{-1}	9.09×10^5	4.95×10^1	1.65×10^4
11	2.34×10^6	7.90×10^5	1.36	1.72×10^6	4.45×10^1	5.26×10^4
12	6.77×10^6	2.41×10^6	2.45	2.77×10^6	4.47×10^1	1.52×10^5
13	1.89×10^7	7.04×10^6	6.74	2.80×10^6	5.15×10^1	3.66×10^5
14	5.32×10^7	2.34×10^7	2.97×10^1	1.79×10^6	6.57×10^1	8.10×10^5
15	1.47×10^8	6.38×10^7	8.21×10^1	1.79×10^6	1.26×10^2	1.17×10^6
16	4.00×10^8	2.30×10^8	2.89×10^2	1.38×10^6	3.44×10^2	1.16×10^6
17	1.09×10^9	9.19×10^8	1.09×10^3	1.00×10^6	9.80×10^2	1.11×10^6
18	2.94×10^9	2.55×10^9	3.67×10^3	7.99×10^5	3.68×10^3	7.97×10^5
19	7.84×10^9	6.58×10^9	1.56×10^4	5.02×10^5	2.23×10^4	3.52×10^5

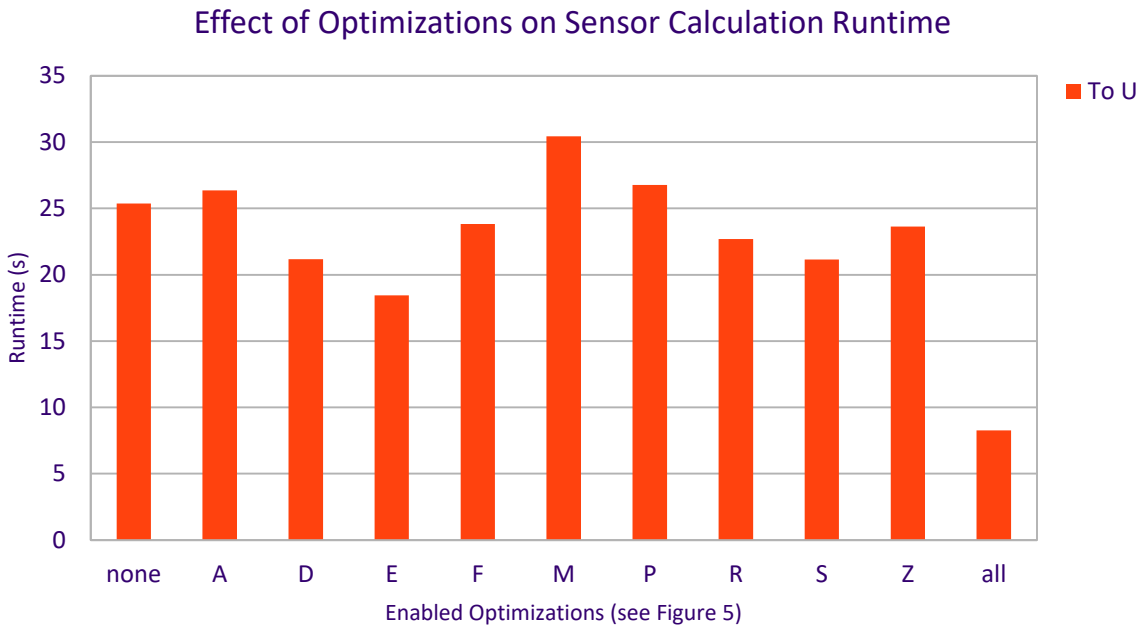
Table 1: 8-worker $A^\intercal B$ experiment Runtimes, Rates, and Statistics
AfterSum counts the number of entries in C after summing colliding partial products together.

Expressiveness Experiment



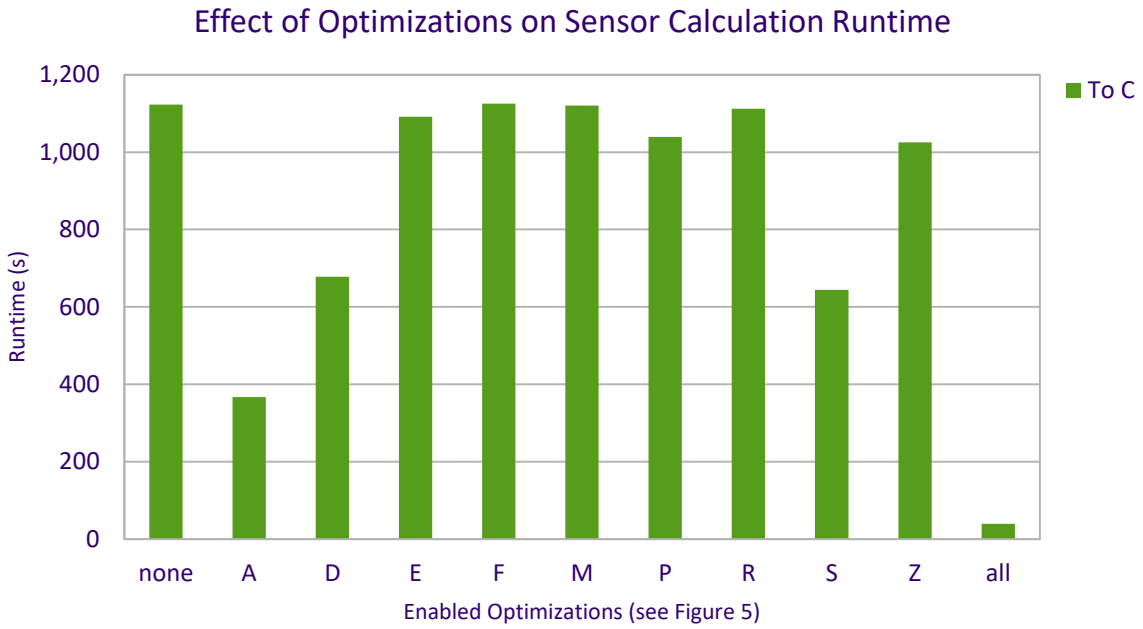
- 4 Worker nodes
- 1.5 months' sensor data
  - Partitioned in 3 day segments
  - 1.2 GB raw; 60 MB after filter to 1 month, project, compress in HDFS

Expressiveness Experiment



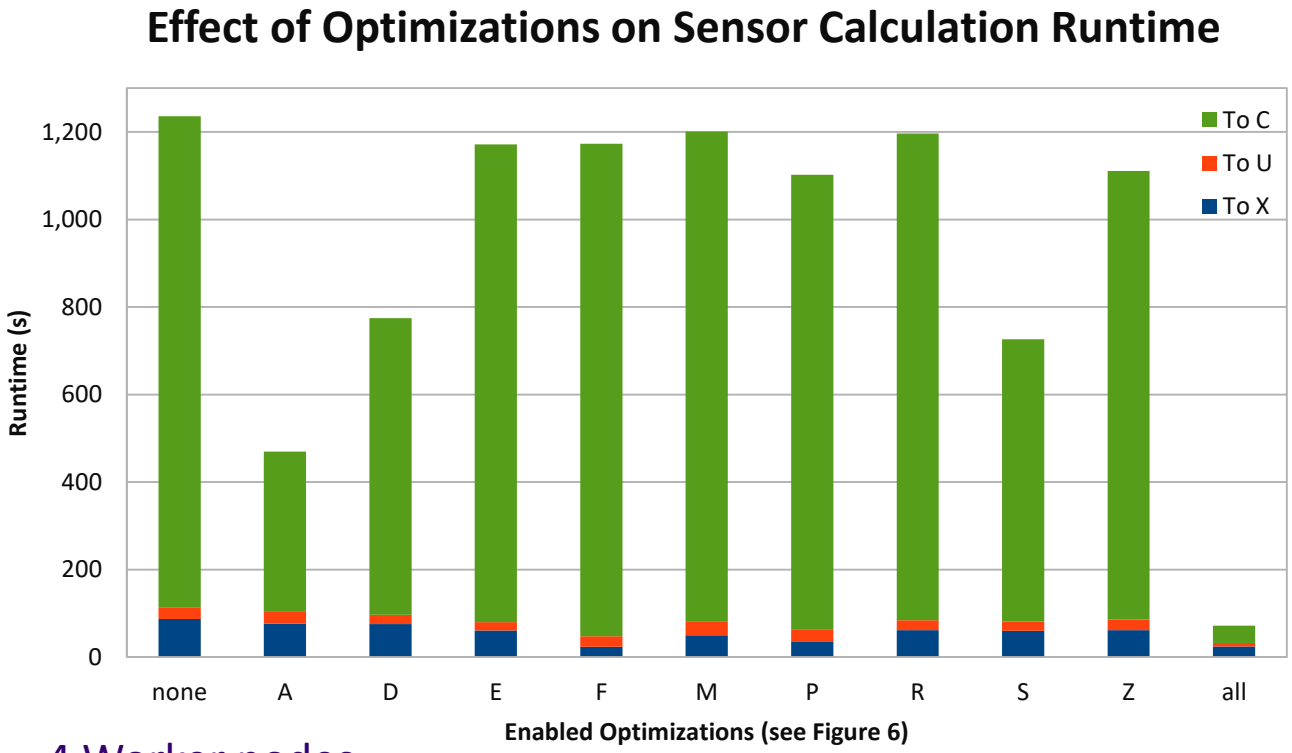
- 4 Worker nodes
- 1.5 months' sensor data
  - Partitioned in 3 day segments
  - 1.2 GB raw; 60 MB after filter to 1 month, project, compress in HDFS

Expressiveness Experiment



- 4 Worker nodes
- 1.5 months' sensor data
  - Partitioned in 3 day segments
  - 1.2 GB raw; 60 MB after filter to 1 month, project, compress in HDFS

Expressiveness Experiment



3 Parts of Sensor Calc
Matrix multiply & divide
Subtract mean
Filter, bin, join sensors

- 4 Worker nodes
- 1.5 months' sensor data
  - Partitioned in 3 day segments
  - 1.2 GB raw; 60 MB after filter to 1 month, project, compress in HDFS

Pseudocode	LARA Logical Plan
1 $A, B =$ LOAD 's1', 's2'	$A =$ LOAD 's1'
2 *// RA-style (SQL) bin, filter*	$A_1 =$ MAP A BY [v: if($460 \le t \le 860$) v else $\perp$]
3 $A' =$ SELECT bin(t) AS t', c, avg(v) FROM A WHERE $460 \le t \le 860$	$A_2 =$ EXT A_1 BY $\begin{array}{c\|cc} t' & v & cnt \\ \hline \text{bin}(t) & v & v \neq \perp \end{array}$
4 GROUP BY t', c	$A_3 =$ AGG A_2 ON t', c BY [v: +, cnt: +]
5	$A' =$ MAP A_3 BY [v: v/cnt]
6 $B' =$ SELECT bin(t) AS t', c, avg(v) FROM B WHERE $460 \le t \le 860$ GROUP BY t', c	$B' = \ldots$ *// repeat above for second sensor*
// LA-style (MATLAB) mean, covariance of residuals $A' - B'$, viewed as dense matrices:	
7 $X = A' - B'$ *// residuals; $\|t'\| \times \|c\|$ matrix*	$X =$ JOIN A', B' BY [v: $-$]
8 $N =$ size(X, 1) *// # unique t's; scalar*	$X_1 =$ MAP X BY [v: $v \neq \perp$]
9	$X_2 =$ AGG X_1 ON t' BY [v: any]
10	$N =$ AGG X_2 BY [v: +]
11 $M =$ mean(X, 1) *//c means; $1 \times \|c\|$ vector*	$X_3 =$ MAP X BY [v: v, cnt: $v \neq \perp$]
12 STORE M	$X_4 =$ AGG X_3 ON c BY [v: +, cnt: +]
13	$M =$ MAP X_4 BY [v: v/cnt]
14 $U = X -$ repmat($M, N, 1$) *//subtract mean*	$U =$ JOIN X, M BY [v: $-$]
15 $C = U^\intercal U/(N-1)$ *// c covariances;*	$U_1 =$ RENAME U FROM c TO c'
16 STORE C *// $\|c\| \times \|c\|$ matrix*	$U_2 =$ JOIN U_1, U BY [v: $\times$]
17	$U_3 =$ AGG U_2 ON c, c' BY [v: +]
18	$C =$ JOIN U_3, N BY [v: $v/(v'-1)$]

t'	c	[⊥] v
460	temp	-3.1
460	hum	1.6
520	temp	-4.0
520	hum	-0.8

(c) Table X

()	v
	2

(d) N

c	[⊥] v
temp	0.4
hum	-3.5

(e) Table M

t'	c	[⊥] v
460	temp	0.4
460	hum	1.2
520	temp	-0.4
520	hum	-1.2

(f) Table U

c_1	c_2	[⊥] v
temp	temp	0.32
temp	hum	0.96
hum	temp	0.96
hum	hum	2.88

(g) Table C

Sensor Query, complete translation to LARA

LARA Physical Plan	Access Path	Optimizations
1 $A =$ LOAD 's1'	$[t, c]$	(E) Encode numeric attributes in packed byte form
2 $A_1 =$ MAP A BY $[v$: if$(460 \leq t \leq 860)$ v else $\perp]$	$[t, c]$	(F) Push filter into LOAD 's1' FROM 460 TO 860
3 $A_2 =$ EXT A_1 BY $[t'$: bin$(t) \to v$: v, cnt: $v \neq \perp]$	$[t, c, t']$	(M) Eliminate **SORT** by t': bin(t) monotone in t
3.5 $A_{20} =$ **SORT** A_2 TO $[t', c, t]$	$[t', c, t]$	
4 $A_3 =$ MERGEAGG A_{20} ON t', c BY $[v$: $+$, cnt: $+]$	$[t', c]$	
5 $A' =$ MAP A_3 BY $[v$: $v/cnt]$	$[t', c]$	
6 $B' = \ldots$ // *repeat above for second sensor*	$[t', c]$	
7 $X =$ MERGEJOIN A', B' BY $[v$: $-]$	$[t', c]$	(P) Propagate A, B's partition splits throughout
8 $X_1 =$ MAP X BY $[v$: $v \neq \perp]$	$[t', c]$	
9 $X_2 =$ MERGEAGG X_1 ON t' BY $[v$: any$]$	$[t']$	
10 $N =$ AGG X_2 BY $[v$: $+]$	$[]$	(C) Store scalar N at client instead of a table
10.5 $X_0 =$ **SORT** X TO $[c, t']$	$[c, t']$	(Z) If M, C relaxed to *sparse* matrix interpretation,
11 $X_3 =$ MAP X_0 BY $[v$: v, cnt: $v \neq \perp]$	$[c, t']$	identify $\perp$ with 0, discarding 0-valued entries
12 $X_4 =$ MERGEAGG X_3 ON c BY $[v$: $+$, cnt: $+]$	$[c]$	in X_3 and all following tables
13 $M =$ MAP X_4 BY $[v$: $v/cnt)]$	$[c]$	(D) Defer X_3, X_4, M to future scans on X_0,
13.5 STORE M	$[c]$	eliminating write-out of M
14 $U =$ MERGEJOIN X_0, M BY $[v$: $-]$	$[c, t']$	(R) Reuse X_0 data source (common sub-expression)
14.5 $U_0 =$ **SORT** U TO $[t', c]$	$[t', c]$	(U_2 has a similar sub-expression below)
15 $U_1 =$ RENAME U_0 FROM c TO c'	$[t', c']$	(S) $U^\intercal U$ is symmetric; only compute upper triangle
16 $U_2 =$ MERGEJOIN U_0, U_1 BY $[v$: $\times]$	$[t', c, c']$	via MAP filter $c \leq c'$
16.5 $U_{20} =$ **SORT** U_2 TO $[c, c', t']$	$[c, c', t']$	(A) Push sum of partial products into U_{20} compaction and
17 $U_3 =$ MERGEAGG U_{20} ON c, c' BY $[v$: $+]$	$[c, c']$	flush; assume no repeated writes due to server failure
18 $C =$ MERGEJOIN U_3, N BY $[v$: $v/(v'-1)]$	$[c, c']$	(D) Defer U_3, C to future scans on U_{20},
18.5 STORE C	$[c, c']$	eliminating final pass

Example optimization:

- ✓ Eliminate **SORT** after monotonic EXT

$$\text{(M)}\ \frac{A : k \to v \qquad f(k, v) : k' \to v' \qquad f \text{ monotone in } k \qquad \textbf{SORT}\ (\text{EXT}\ A\ \text{BY}\ f)\ \text{TO}\ [k', k]}{\text{EXT}\ A\ \text{BY}\ f\ \text{OVER}\ [k', k]}$$

LARADB

t	c	t'	[⊥] v	[0] cnt
466	temp	460	55.2	1
466	hum	460	40.1	1
492	temp	520	56.3	1
492	hum	520	35.0	1
528	temp	520	56.5	1

Join $\bowtie_{\otimes}$

Union $\overline{\bowtie}_{\oplus}$

Extension ext_{f}

Contact: dhutchis@cs.washington.edu

slide from Andreas Kunft, Alexander Alexandrov, Asterios Katsifodimos, Volker Markl's 2016 BeyondMR talk: *Towards optimization across linear and relational algebra*

Introduction

Relational Algebra (RA): Preprocessing
Linear Algebra (LA): Analytics

Dataflow programs evolved from simple jobs to **complex pipelines**

> These pipelines are composed of **different domains**, e.g. collection processing, matrix processing

> Two implementation choices

Some *tasks* and *properties* easier to express in RA, others in LA

Single DSL > **Impedance Mismatch** (SQL OR Matlab)

Multiple DSLs > **Lost optimization potential** (SQL AND Matlab)

Using both RA and LA is difficult without a means of translation, such as a *common algebra*

Last year's Sensor example: Combine measurements, compute means & covariances

```
1  // Read measurements into the DataBags A and B
2  val A = readCSV(...) //
3  val B = readCSV(...)
4  // SELECT a_1, ...,a_N, b_1, ..., b_M
5  // FROM   A, B
6  // WHERE  A.id = B.id
7  val X = for {
8          a <- A
9          b <- B
10         if a.id == b.id
11        } yield (a_1, ...,a_N, b_1, ..., b_M)
12
13 // Convert DataBag X into Matrix M
14 val M = X.toMatrix()
15 // Calculate the mean of each column c of the matrix
16 val means = for ( c <- M.cols() ) yield (mean(c))
17 // Compute the deviation of each cell of M
18 // to the cell's column mean.
19 val U = M - Matrix.fill(M.numRows, M.numCols)
20                          ((i,j) => means(j))
21 // Compute the covariance matrix
22 val C = 1 / (U.numRows - 1) * U.t %*% U
```

Preprocessing (easier to express in RA)

Analysis (easier to express in LA)

Use Lara underneath Weld?

- Weld replaced the implementation of Pandas (RAish) and NumPy (Laish)
- Weld chose a MapReduce algebra
- Would Weld benefit from using LARA algebra instead?

S. Palkar, J. J. Thomas, A. Shanbhag, D. Narayanan, H. Pirk, M. Schwarzkopf, S. Amarasinghe, M. Zaharia, and S. InfoLab. Weld: A common runtime for high performance data analytics. CIDR, Jan. 2017.

Sensor Query, Alternate Plan

```
A, B = LOAD('s1.csv'), LOAD('s2.csv')
A' = SELECT bin(t) AS t', c, avg(v) AS v
     FROM A WHERE 460 <= t <= 860
     GROUP BY t', c
B' = …      // repeat above

X = A - B;
N = size(X, 1);
M = mean(X, 1);
disp(M);
U = X - repmat(M, N, 1);
C = U.' * U ./ (N - 1);
disp(C);
```

```
A, B = LOAD('s1.csv'), LOAD('s2.csv')
A' = SELECT bin(t) AS t', c, avg(v) AS v
     FROM A WHERE 460 <= t <= 860
     GROUP BY t', c
B' = …      // repeat above

X = A - B;
N = size(X, 1);
M = mean(X, 1);
disp(M);

C = N ./ (N – 1) .* ((X.' * X ./ N) – M.' * M);
disp(C);
```

Filter as LARA Map (Ext)

t	c	[⊥] v
440	temp	54.0
440	hum	40.8
466	temp	55.2
466	hum	40.1
492	temp	56.3
492	hum	35.0
528	temp	56.5

A' = SELECT **t**, **c**, **v** FROM **A**
WHERE 460 <= **t** <= 860

$\mathbf{A}_1$ = MAP **A** BY
[**v**: if (460 ≤ **t** ≤ 860) **v** else ⊥]

()	[⊥] v
()	if (460 ≤ **t** ≤ 860) **v** else ⊥

t	c	[⊥] v
440	temp	⊥
440	hum	⊥
466	temp	55.2
466	hum	40.1
492	temp	56.3
492	hum	35.0
528	temp	56.5

UDF *f* as a table-valued function, ready for ext (flatmap)

LARA on One Slide

f(a, c, x, z) =

k'	[0']
	v'

UDFs: $\oplus : i \times i \to i \quad \text{for } i \in \{x, y, z\}$

$\otimes : z \times z \to z'$

$f : a, c \times x, z \to (k' \to v' : 0')$

UDF Requirements:

$\forall i,\ i \oplus 0^i = 0^i \oplus i = 0^i \text{ for } i \in \{x, y, z\}$

$\forall z,\ 0^z \otimes z = z \otimes 0^z = 0^z \otimes 0^z$

$\forall a, c, k',\ f(a, c, 0^x, 0^z)(k') = 0'$

$\forall a, c, x, z,\ f(a, c, x, z) \text{ has finite support}$

A

a	c	[0^x] x	[0^z] z

B

c	b	[0^z'] z	[0^y] y

JOIN **A, B** BY $\otimes$

a	c	b	[0^z ⊗ 0^z'] z

$A \bowtie_{\otimes} B : a, c, b \to z : 0^z \otimes 0^z$

$(A \bowtie_{\otimes} B)(a, c, b) := [z\colon \pi_z A(a, c) \otimes \pi_z B(c, b)]$

UNION **A, B** BY $\oplus$

c	[0^x] x	[0^z (= 0^z')] z	[0^y] y

$A \between_{\oplus} B : c \to x, z, y : 0^x, 0^z, 0^y$

$(A \between_{\oplus} B)(c) := \Big[x\colon \bigoplus_a \pi_x A(a, c),$

$z\colon \bigoplus_a \pi_z A(a, c) \oplus \bigoplus_b \pi_z B(c, y),\ y\colon \bigoplus_b \pi_x B(c, y)\Big]$

EXT **A** BY f

a	c	k'	[0'] v'

$\text{ext}_f\, A : a, c, k' \to v' : 0'$

$(\text{ext}_f\, A)(a, c, k') := f(a, c, A(a, c))(k')$

Translating LA and RA to LARA

Given $A : i, j \to v,\ B : j, k \to v,\ C : i, j \to v$

GraphBLAS

RA	LARA
σ_p	map_p
π	map or $\bigoplus$
$\times, \bowtie$	$\bowtie$
$\gamma, \cup$	$\bigoplus$

(a) RA to LARA

LA	LARA
$A \oplus . \otimes B$	$\bigoplus_{\oplus}^{i,k}(A \bowtie_{\otimes} B)$
$A \otimes C$	$A \bowtie_{\otimes} C$
$A \oplus C$	$A \bigoplus_{\oplus} C$
$\text{reduce}(A, \oplus)$	$\bigoplus_{\oplus}^{i} A$
$A(I, J)$	$A \bowtie I \bowtie J$
$f(A)$	map_f
$A^\intercal$	rename

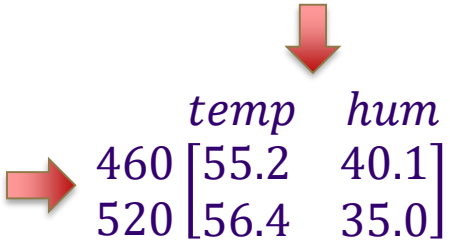
(b) LA to LARA

Roughly: LA, RA ⊆ LARA ⊆ MapReduce

Convert RA Relation, LA Matrix to LARA Associative Table

1. Identify keys; separate attributes into keys and values
2. Introduce sparsity; make the table a total function: keys → values with finite support

t'	c	v
460	temp	55.2
460	hum	40.1
520	temp	56.4
520	hum	35.0



t'	c	v
460	temp	55.2
460	hum	40.1
520	temp	56.4
520	hum	35.0

Convert RA Relation, LA Matrix to LARA Associative Table

1. Identify keys; separate attributes into keys and values
2. Introduce sparsity; make the table a total function: keys → values with finite support

Set of keys that map to non-default values

Support

t'	c	v
460	temp	55.2
460	hum	40.1
520	temp	56.4
520	hum	35.0
461	temp	⊥
461	wind	⊥
902	hum	⊥
3	wind	⊥

	temp	*hum*	*wind*	
3	⊥	⊥	⊥	…
29	⊥	⊥	⊥	…
460	55.2	40.1	⊥	…
461	⊥	⊥	⊥	…
520	56.4	35.0	⊥	…
557	⊥	⊥	⊥	…
902	⊥	⊥	⊥	…
911	⊥	⊥	⊥	…
	…	…	…	…

Default value

t'	c	[⊥] v
460	temp	55.2
460	hum	40.1
520	temp	56.4
520	hum	35.0
(3	wind	⊥)
(461	temp	⊥)
(…	…	…)

Filter as LARA Map (Ext)

t	c	[⊥] v
440	temp	54.0
440	hum	40.8
466	temp	55.2
466	hum	40.1
492	temp	56.3
492	hum	35.0
528	temp	56.5

A' = SELECT **t**, **c**, **v** FROM **A**
WHERE 460 <= **t** <= 860

$\mathbf{A}_1$ = MAP **A** BY
[**v**: if (460 ≤ **t** ≤ 860) **v** else ⊥]

t	c	[⊥] v
440	temp	⊥
440	hum	⊥
466	temp	55.2
466	hum	40.1
492	temp	56.3
492	hum	35.0
528	temp	56.5

Bin as LARA Ext

t	c	[⊥] v
466	temp	55.2
466	hum	40.1
492	temp	56.3
492	hum	35.0
528	temp	56.5

$\mathbf{A_2}$ = SELECT **t**, bin(**t**) AS **t'**, **c**, **v**, **v** ≠ ⊥ AS **cnt** FROM $\mathbf{A_1}$

$\mathbf{A_2}$ = EXT $\mathbf{A_1}$ BY

f(t, c, v) =

t'	[⊥] v	[0] cnt
bin(**t**)	**v**	$\begin{cases} 1, & \mathbf{v} \neq \perp \\ 0, & \mathbf{v} = \perp \end{cases}$

t	c	t'	[⊥] v	[0] cnt
466	temp	460	55.2	1
466	hum	460	40.1	1
492	temp	520	56.3	1
492	hum	520	35.0	1
528	temp	520	56.5	1

- Appends new key **t'**, replaces values
- Ext function f maintains finite support because

$$\forall t, c, t', \qquad f(t, c, \perp)(t') = (\perp, 0)$$

Avg as LARA Agg (Union)

t	c	t'	[⊥] v	[0] cnt
466	temp	460	55.2	1
466	hum	460	40.1	1
492	temp	520	**56.3**	**1**
492	hum	520	35.0	1
528	temp	520	**56.5**	**1**

$\mathbf{A_3}$ = SELECT **t'**, **c**, sum(**v**) AS **v**, sum(**cnt**) AS **cnt** FROM $\mathbf{A_2}$

$\mathbf{A_3}$ = AGG $\mathbf{A_2}$ ON **t'**, **c** BY [**v**: +, **cnt**: +]

t'	c	[⊥] v	[0] cnt
460	temp	55.2	1
460	hum	40.1	1
520	temp	**112.8**	**2**
520	hum	35.0	1

- Agg is Union with an empty table

t'	c	()

- Another Map finishes the calculation of avg(v): $f(t', c, v, cnt) = [v\colon v \,/\, cnt]$

Subtract as LARA Join

t'	c	[⊥] v
460	temp	55.2
460	hum	40.1
520	temp	56.4
520	hum	35.0

$\bowtie_{-}$

t'	c	[⊥] v
460	temp	58.3
460	hum	38.5
520	temp	60.4

=

t'	c	[⊥] v
460	temp	-3.1
460	hum	1.6
520	temp	-4.0

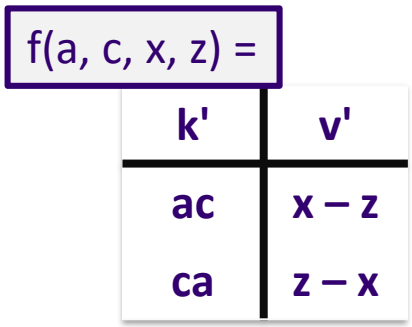
X = A – B

X = JOIN **A**, **B** BY [**v**: –]

Let ∀a, ⊥ – a = a – ⊥ = ⊥

- Last year Andreas observed that computational pipelines increasingly cross data models, in particular RA and LA. He illustrated why this is a problem with a sensor example.
- We found data that made the sensor example real, from a project called the Array of Things started right here in the city of Chicago.
  - Added a few more preprocessing tasks (filtering, binning) to fit the real data.
- Why is a common algebra possible?
- Lara; discuss associative table on the spot
- Join, Union, (Join-Union duality), Ext, properties
- Sensor example pseudocode, logical Lara
- Sorted Distributed Maps – relational, key-value, array systems
- Lower to Physical algebra, Accumulo
- Admits lots of optimizations; these significantly reduced runtime
- Does performance take a nosedive if we use Lara as a guiding abstraction for implementations? No; competitiveness experiment with MapReduce, analytics engine natively integrated with Accumulo on MxM, a fundamental building block for LA and RA

Ext: Flatmap



ext_f

a	c	[0] x	[0] z
a_1	c_1	11	1
a_1	c_2	12	2
a_2	c_1	13	3

=

a	c	k'	[0 – 0 = 0] v'
a_1	c_1	a_1c_1	11 – 5
a_1	c_1	c_1a_1	5 – 11
a_1	c_2	a_1c_2	12 – 2
a_1	c_2	c_2a_1	2 – 12
a_2	c_1	a_2c_1	13 – 3
a_2	c_1	c_1a_2	3 – 13

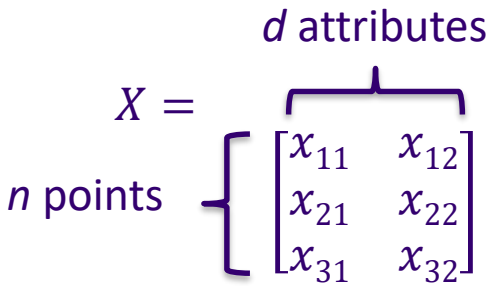
$$\text{ext}_f\ A : a, c, k' \to v' : 0'$$

$$(\text{ext}_f\ A)(a, c, k') := f(a, c, A(a, c))(k')$$

Requires:

$$\forall a, c, k',\ f(a, c, 0^x, 0^z)(k') = 0'$$

Covariance query: easy in LA, harder in RA



$$X = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix}$$

$M = \frac{1}{n} 1^T X$ is a $1 \times d$ matrix

$C = \frac{1}{n} X^T X - M^T M$ is a $d \times d$ matrix

LA

N = size(**X**, 1);

M = mean(**X**, 1);

C = **X**'*****X** / **N** – **M**'*****M**;

RA

(Generated SQL statements for each entry)

T = SELECT FROM **X** sum(1.0) AS N,

sum(X_1) AS M_1, sum(X_2) AS M_2, …, sum(X_d) AS M_d,

sum(X_1*X_1) AS Q_{11}, sum(X_1*X_2) AS Q_{12}, …,

sum(X_{d-1}*X_d) AS $Q_{(d-1)d}$, sum(X_d*X_d) AS Q_{dd}

C = SELECT FROM **T**

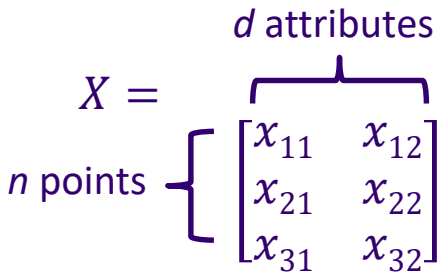
(1 AS i, 1 AS j, Q_{11}/N – M_1*M_1 AS v) UNION

(1 AS i, 2 AS j, Q_{12}/N – M_1*M_2 AS v) UNION

…

Carlos Ordonez.
Building Statistical Models and Scoring with UDFs. SIGMOD 2007.

Covariance query: easy in LA, harder in RA



$$\begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix}$$

Mean $M = \frac{1}{n} 1^T X$

Covariance $C = \frac{1}{n} X^T X - M^T M$

LA

N = size(**X**, 1);

M = mean(**X**, 1);

C = **X**'*****X** / **N** – **M**'*****M**;

RA

(Generated SQL statements for each entry)

T = SELECT FROM **X** sum(1.0) AS N,

sum(X_1) AS M_1, sum(X_2) AS M_2, …, sum(X_d) AS M_d,

sum(X_1*X_1) AS Q_{11}, sum(X_1*X_2) AS Q_{12}, …,

sum(X_{d-1}*X_d) AS $Q_{(d-1)d}$, sum(X_d*X_d) AS Q_{dd}

C = SELECT FROM **T**

(1 AS i, 1 AS j, Q_{11}/N – M_1*M_1 AS v) UNION

(1 AS i, 2 AS j, Q_{12}/N – M_1*M_2 AS v) UNION

…

Carlos Ordonez.
Building Statistical Models and Scoring with UDFs. SIGMOD 2007.

LARA Properties

- If $\oplus$ or $\otimes$ is *associative*, *commutative*, or *idempotent*, then so is Union or Join
- (Push Aggregation into Join) If $\otimes$ distributes over $\oplus$, then

$$(A \bowtie_{\otimes} B) \Join_{\oplus} C = ((\Join_{\oplus}^{\mathrm{k_B} \cup \mathrm{k_C}} A) \bowtie_{\otimes} (\Join_{\oplus}^{\mathrm{k_A} \cup \mathrm{k_C}} B)) \Join_{\oplus} (\Join_{\oplus}^{\mathrm{k_A} \cup \mathrm{k_B}} C)$$

- (Distribute Join over Union) If $(k_B \,\Delta\, k_C) \cap k_A = \emptyset$, then

$$A \bowtie_{\otimes} (B \Join_{\oplus} C) = (A \bowtie_{\otimes} B) \Join_{\oplus} (A \bowtie_{\otimes} C)$$

$$\begin{aligned}
& \mathrm{tr}(ABC) && \\
= \; & \Join_{+} \mathrm{ext}_{i=l}(ABC) && \text{tr defn.} \\
= \; & \Join_{+} \mathrm{ext}_{i=l} \Join_{+}^{\mathrm{i,l}} (\Join_{+}^{\mathrm{i,k}} (A_{ij} \bowtie_{\otimes} B_{jk}) \bowtie_{\otimes} C_{kl}) && ABC \text{ defn.} \\
= \; & \Join_{+} \Join_{+}^{\mathrm{i,l}} \mathrm{ext}_{i=l} (\Join_{+}^{\mathrm{i,k}} (A_{ij} \bowtie_{\otimes} B_{jk}) \bowtie_{\otimes} C_{kl}) && \text{push ext into } \Join \\
= \; & \Join_{+} \mathrm{ext}_{i=l} (\Join_{+}^{\mathrm{i,k}} (A_{ij} \bowtie_{\otimes} B_{jk}) \bowtie_{\otimes} C_{kl}) && \text{combine } \Join \\
= \; & \Join_{+} (\Join_{+}^{\mathrm{i,k}} (A_{ij} \bowtie_{\otimes} B_{jk}) \bowtie_{\otimes} C_{ki}) && \text{apply ext} \\
= \; & \Join_{+} \Join_{+}^{\mathrm{i,k}} (A_{ij} \bowtie_{\otimes} B_{jk} \bowtie_{\otimes} C_{ki}) && \text{distr. } \bowtie \text{ into } \Join \\
= \; & \Join_{+} (A_{ij} \bowtie_{\otimes} B_{jk} \bowtie_{\otimes} C_{ki}) && \text{combine } \Join \\
= \; & \Join_{+} (B_{jk} \bowtie_{\otimes} C_{ki} \bowtie_{\otimes} A_{ij}) && \text{commute } \bowtie_{\otimes} \\
= \; & \ldots \;\; // \textit{ reversing the above steps} && \\
= \; & \mathrm{tr}(BCA) &&
\end{aligned}$$

$= \mathrm{sum}(AB \otimes C^{\mathsf{T}})$

Δ is symmetric difference

Is a common algebra possible?

Evidence:

> Overlapping expressiveness
  - Matrix Multiply = GroupBy ∘ Apply ∘ Join

$$A_{i,j} \oplus.\otimes B_{j,k} = \gamma^{i,k}_{\oplus(v)}\ \pi_{i,j,k,v=v_A \otimes v_B}(A_{i,j,v_A} \bowtie B_{j,k,v_B})$$

> Emergence of hybrid systems
  - Spark, Myria, MADlib, SystemML, Weld, …
  - Systems with RA and LA interfaces and/or data models

Roadmap

- Motivation: Sensor query
  - Sometimes RA is better, sometimes LA
- LARA Algebra
  - 3 Operators & Properties
- LARADB on *Sorted Distributed Maps*
  - Physical operators & optimizations
- Experiments
  - Expressiveness: LARA affords impactful optimizations
  - Competitiveness: LARA affords competitive performance