AWARE: Workload-aware, Redundancy-exploiting Linear Algebra

Sebastian Baunsgaard & Matthias Boehm
Lossy

- Main stream ML Systems
- Specialized data types
- Associated with trial and error
Matrix Compression

Lossy
- Main stream ML Systems
- Specialized data types
- Associated with trial and error

Lossless CLA
- Lightweight database compression
- Preaggregation Cocoding
- Correct results

Optimization Goal
- Normal: Compression Ratio
- We: Workload Time
Matrix Compression

**Lossy**
- Main stream ML Systems
- Specialized data types
- Associated with trial and error

**Lossless CLA**
- Lightweight database compression
- Preaggregation Cocoding
- Correct results
Matrix Compression

Lossy
- Main stream ML Systems
- Specialized data types
- Associated with trial and error

Lossless CLA
- Lightweight database compression
- Preaggregation Cocoding
- Correct results

Optimization Goal
- Normal: *Compression Ratio*
- We: *Workload Time*
User Script:

```r
X = read("data/X")
y = read("data/y")

X = scale(X,TRUE,TRUE)
w = l2svm(X,y,TRUE,
        1e-9,1e-3,100)

write(w,"data/wxy")
```

Workload Awareness

**Built-in Functions:**

- `read` for reading data
- `write` for writing data
- `scale` for scaling data
- `l2svm` for a linear SVM
- `cbind` for creating a column vector
- `colMeans` for calculating column means
- `colSds` for calculating column standard deviations
- `ones` for creating a column of ones
User Script:

\[
\begin{align*}
X &= \text{read}("data/X") \\
y &= \text{read}("data/y") \\
X &= \text{scale}(X, \text{TRUE}, \text{TRUE}) \\
w &= \text{l2svm}(X, y, \text{TRUE}, 1e^{-9}, 1e^{-3}, 100) \\
\text{write}(w, "data/wXy")
\end{align*}
\]

if(shift)
\[
X = X - \text{colMeans}(X)
\]
if(scale)
\[
X = X / \text{colSds}(X)
\]

Built-in Functions:

if(intercept)
\[
X = \text{cbind}(X, \text{ones})
\]
while(conto \& i<maxi) {
\[
Xd = X \text{%%% s}
\]
while(conti) {
\[
\begin{align*}
\text{out} &= 1 - y \times (Xw + sz \times Xd) \\
\text{sz} &= \text{sz} - g / h; \# ... \\
\text{g}\_\text{new} &= t(X) \text{%%% (out*y)}
\end{align*}
\]
}
User Script:

\[
\begin{align*}
X &= \text{read("data/X")} \\
y &= \text{read("data/y")} \\
X &= \text{scale}(X, \text{TRUE, TRUE}) \\
w &= \text{l2svm}(X, y, \text{TRUE, 1e-9, 1e-3, 100}) \\
\text{write}(w, "data/wXy")
\end{align*}
\]

if(shift)
\[X = X - \text{colMeans}(X)\]
if(scale)
\[X = X / \text{colSds}(X)\]

Built-in Functions:

\[
\begin{align*}
\text{if(intercept)} &: \text{X = cbind(X, ones)} \\
\text{while}(\text{conto & i<maxi}) &: \text{Xd = X \%\% s} \\
\text{while}(\text{conti}) &: \text{out = 1-\text{y}*(Xw+sz*Xd)} \\
&: \text{sz = sz - g/h}; \# ... \\
&: \text{g_new = t(X) \%\% (out*y)}
\end{align*}
\]
Workload Awareness

User Script:

```r
X = read("data/X")
y = read("data/y")

X = scale(X,TRUE,TRUE)
w = l2svm(X,y,TRUE,1e-9,1e-3,100)
write(w,"data/wXy")
```

Built-in Functions:

```r
if(shift)
  X = X - colMeans(X)
if(scale)
  X = X / colSds(X)
```

Workload Tree

Cost Summary

```
0 100 10 10 105 0
```
Compression Execution

Uncompressed Input Matrix

Sample

1. Classify

2. Grouping

3. Transpose (optional)

4. Compress Extract & Compress

Compression Plan

0

1,3

2

4

5

Compressed Output Matrix
Compression Execution

Uncompressed Input Matrix

1. Classify
   Extract column statistics

Sample

0 1 2 3 4 5

Compressed Output Matrix
Compression Execution

1. Classify
   Extract column statistics

2. Grouping
   Co-code Statistics

Uncompressed Input Matrix
Compression Execution

Uncompressed Input Matrix

1. Classify
   - Sample
   - Extract column statistics

2. Grouping
   - Co-code Statistics

Compression Plan

Uncompressed Input Matrix

1. Classify
   - Sample
   - Extract column statistics

2. Grouping
   - Co-code Statistics

Compression Plan
Compression Execution

Uncompressed Input Matrix

1. Classify
   Sample
   Extract column statistics
   0 1 2 3 4 5

2. Grouping
   Co-code Statistics
   0 1,3 2 4 5

3. Transpose
   (optional)

Compressed Output Matrix
## Compression Execution

1. **Classify**
   - Sample
   - Extract column statistics

2. **Grouping**
   - Co-code Statistics

3. **Transpose** (optional)

4. **Compress**
   - Extract & Compress

### Uncompressed Input Matrix

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
</table>

### Compression Plan

- 0 1,3 2 4 5
Compression Execution

1. Classify
   - Extract column statistics
   - Sample

2. Grouping
   - Co-code Statistics

3. Transpose (optional)

4. Compress
   - Extract & Compress

Uncompressed Input Matrix

Compressed Output Matrix
Compression Example

Uncompressed

<table>
<thead>
<tr>
<th>Uncompressed</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>9</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>8.5</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>8.5</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>9</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>7</td>
</tr>
</tbody>
</table>

Compressed
## Compression Example

<table>
<thead>
<tr>
<th>Uncompressed</th>
<th>Compressed</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Uncompressed Table" /></td>
<td><img src="image" alt="Compressed Table" /></td>
</tr>
</tbody>
</table>
Compression Example

Uncompressed

<table>
<thead>
<tr>
<th>Uncompressed</th>
<th>Compressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 0 6 0 4</td>
<td>DDC{0}</td>
</tr>
<tr>
<td>7 3 0 4 0 7</td>
<td></td>
</tr>
<tr>
<td>1 2 0 6 0 4</td>
<td></td>
</tr>
<tr>
<td>7 2 0 4 0 7</td>
<td></td>
</tr>
<tr>
<td>1 3 9 4 0 7</td>
<td></td>
</tr>
<tr>
<td>7 0 8.5 4 0 7</td>
<td></td>
</tr>
<tr>
<td>1 3 8.5 4 0 7</td>
<td></td>
</tr>
<tr>
<td>7 3 0 4 0 7</td>
<td></td>
</tr>
<tr>
<td>1 2 0 4 0 7</td>
<td></td>
</tr>
<tr>
<td>7 3 9 4 0 7</td>
<td></td>
</tr>
</tbody>
</table>

Map

<table>
<thead>
<tr>
<th>Map</th>
<th>Dict</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

Compressed

SDC

SDCSingle

SDCZero

Off

Off

Def

Def

Off

Off
### Compression Example

#### Uncompressed

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>9</td>
<td>8.5</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>8.5</td>
<td>8.5</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>9</td>
<td>9</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
</tbody>
</table>

#### Compressed

**DDC{0}**

- **Map**:
  - 0: 1
  - 1: 0
  - 2: 1
  - 3: 0
  - 4: 1

- **Dict**: 17

**SDC{1,3}**

- **Off**:
  - 1: 2.6
  - 2: 2.5
  - 3: 3.4

- **Map**:
  - 0: 0
  - 1: 1
  - 2: 0

- **Dict**:
  - 0: 0, 0
  - 1: 2, 5
  - 2: 0, 0

- **Def**:
  - 0: 1, 3
  - 1: 2, 4
Compression Example

Uncompressed

Compressed

**Uncompressed**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>0</th>
<th></th>
<th>6</th>
<th>0</th>
<th>4</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>0</td>
<td>8.5</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0</td>
<td>8.5</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Compressed**

**DDC{0}**

Map: 0 1 0 1 0 1 0 1
Dict: 1 7

**SDC{1,3}**

Off: 1 2 1 2
Map: 0 1 1 2
Dict: 2.6 2.5 0.0
Def: 3.4

**SDCZero{2}**

Off: 5 1 1 3
Map: 0 1 1 0
Dict: 9 8.5
# Compression Example

## Uncompressed

<table>
<thead>
<tr>
<th>uncompressed</th>
<th>1</th>
<th>7</th>
<th>7</th>
<th>1</th>
<th>2</th>
<th>0</th>
<th>6</th>
<th>0</th>
<th>4</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7</td>
<td>3</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>7</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>6</td>
<td>0</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>8.5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>4</td>
<td>0</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>8.5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>4</td>
<td>0</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3</td>
<td>9</td>
<td>0</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>7</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>7</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>3</td>
<td>9</td>
<td>0</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>7</td>
</tr>
</tbody>
</table>

## Compressed

### DDC\{0\}

<table>
<thead>
<tr>
<th>Map</th>
<th>Dict</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

### SDC\{1,3\}

<table>
<thead>
<tr>
<th>Off</th>
<th>Map</th>
<th>Dict</th>
<th>Def</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>2.6</td>
<td>3.4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2.5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0.0</td>
<td></td>
</tr>
</tbody>
</table>

### SDCZero\{2\}

<table>
<thead>
<tr>
<th>Off</th>
<th>Map</th>
<th>Dict</th>
<th>Def</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0</td>
<td>9</td>
<td>8.5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>8.5</td>
<td></td>
</tr>
</tbody>
</table>

### SDCSingle\{5\}

<table>
<thead>
<tr>
<th>Off</th>
<th>Map</th>
<th>Dict</th>
<th>Def</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Difference CLA

<table>
<thead>
<tr>
<th></th>
<th>CLA</th>
<th>Aware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Co-Coding</td>
<td>$O(m^2)$</td>
<td></td>
</tr>
<tr>
<td>Column Group Encodings</td>
<td>4 (5)</td>
<td></td>
</tr>
<tr>
<td>Materialization</td>
<td>Eager</td>
<td></td>
</tr>
<tr>
<td>Optimization Objective</td>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>Matrix Multiplication</td>
<td>MV, VM</td>
<td></td>
</tr>
</tbody>
</table>
## Difference CLA

<table>
<thead>
<tr>
<th></th>
<th>CLA</th>
<th>Aware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Co-Coding</td>
<td>$\mathcal{O}(m^2)$</td>
<td>$\mathcal{O}(m)$</td>
</tr>
<tr>
<td>Column Group Encodings</td>
<td>4 (5)</td>
<td></td>
</tr>
<tr>
<td>Materialization</td>
<td>Eager</td>
<td></td>
</tr>
<tr>
<td>Optimization Objective</td>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>Matrix Multiplication</td>
<td>MV, VM</td>
<td></td>
</tr>
</tbody>
</table>
## Difference CLA

<table>
<thead>
<tr>
<th></th>
<th>CLA</th>
<th>Aware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Co-Coding</td>
<td>$O(m^2)$</td>
<td>$O(m)$</td>
</tr>
<tr>
<td>Column Group Encodings</td>
<td>4 (5)</td>
<td>7 (327)</td>
</tr>
<tr>
<td>Materialization</td>
<td>Eager</td>
<td></td>
</tr>
<tr>
<td>Optimization Objective</td>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>Matrix Multiplication</td>
<td>MV, VM</td>
<td></td>
</tr>
</tbody>
</table>
## Difference CLA

<table>
<thead>
<tr>
<th></th>
<th>CLA</th>
<th>Aware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Co-Coding</td>
<td>$O(m^2)$</td>
<td>$O(m)$</td>
</tr>
<tr>
<td>Column Group Encodings</td>
<td>4 (5)</td>
<td>7 (327)</td>
</tr>
<tr>
<td>Materialization</td>
<td>Eager</td>
<td>Deferred</td>
</tr>
<tr>
<td>Optimization Objective</td>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>Matrix Multiplication</td>
<td>MV, VM</td>
<td></td>
</tr>
</tbody>
</table>
## Difference CLA

<table>
<thead>
<tr>
<th></th>
<th>CLA</th>
<th>Aware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Co-Coding</td>
<td>$O(m^2)$</td>
<td>$O(m)$</td>
</tr>
<tr>
<td>Column Group Encodings</td>
<td>4 (5)</td>
<td>7 (327)</td>
</tr>
<tr>
<td>Materialization</td>
<td>Eager</td>
<td>Deferred</td>
</tr>
<tr>
<td>Optimization Objective</td>
<td>Data</td>
<td>Data &amp; Ops</td>
</tr>
<tr>
<td>Matrix Multiplication</td>
<td>MV, VM</td>
<td></td>
</tr>
</tbody>
</table>
## Difference CLA

<table>
<thead>
<tr>
<th></th>
<th>CLA</th>
<th>Aware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Co-Coding</td>
<td>$O(m^2)$</td>
<td>$O(m)$</td>
</tr>
<tr>
<td>Column Group Encodings</td>
<td>4 (5)</td>
<td>7 (327)</td>
</tr>
<tr>
<td>Materialization</td>
<td>Eager</td>
<td>Deferred</td>
</tr>
<tr>
<td>Optimization Objective</td>
<td>Data</td>
<td>Data &amp; Ops</td>
</tr>
<tr>
<td>Matrix Multiplication</td>
<td>MV, VM</td>
<td>MV, VM, MM</td>
</tr>
</tbody>
</table>
### Difference CLA

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>CLA</th>
<th>Aware</th>
</tr>
</thead>
<tbody>
<tr>
<td>CON</td>
<td>Constant or Empty Columns</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>DDC</td>
<td>Dense Dictionary Coding</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>OLE</td>
<td>Offset-list Encoding</td>
<td>✓</td>
<td>(✓)</td>
</tr>
<tr>
<td>FOR</td>
<td>Frame of Reference</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>RLE</td>
<td>Run-length Encoding</td>
<td>✓</td>
<td>(✓)</td>
</tr>
<tr>
<td>SDC</td>
<td>Sparse Dictionary Coding</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>UC</td>
<td>Uncompressed (dense/sparse)</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
### Operation Example

#### Compressed

<table>
<thead>
<tr>
<th>DDC{0}</th>
<th>SDC{1,3}</th>
<th>SDCZero{2}</th>
<th>SDCSingle{5}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map 0 1 0 1 0 1 0 1</td>
<td>Off 1 2 1 2 3</td>
<td>Off 5 1 1 3</td>
<td>Off 1 2 2 4</td>
</tr>
<tr>
<td>Dict 1 7</td>
<td>Map 0 0 1 2 0</td>
<td>Map 0 1 1 0</td>
<td>Dict 4</td>
</tr>
<tr>
<td></td>
<td>Dict 2.6 2.5 0.0</td>
<td>Dict 9 8.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Def 3,4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Operation Example

Compressed

- **DDC{0}**
  - **Map**:
    - 0
    - 1
    - 1
    - 1
    - 1
  - **Dict**:
    - 1
    - 7

- **SDC{1,3}**
  - **Off**:
    - 1
    - 2
    - 1
    - 2
    - 3
  - **Map**:
    - 0
    - 0
    - 1
    - 2
    - 0
  - **Dict**:
    - 2.6
    - 2.5
    - 2.5
    - 0.0
    - 3.4
  - **Def**:
    - 3

- **SDCZero{2}**
  - **Off**:
    - 5
    - 1
    - 1
    - 3
  - **Map**:
    - 0
    - 1
    - 1
  - **Dict**:
    - 9
    - 8.5

- **SDCSingle{5}**
  - **Off**:
    - 1
    - 2
    - 2
    - 4
  - **Dict**:
    - 4
  - **Def**:
    - 7

**Binary MatrixScalar Operation**: Subtract 7
## Operation Example

### Compressed

<table>
<thead>
<tr>
<th>DDC{0}</th>
<th>SDC{1,3}</th>
<th>SDCZero{2}</th>
<th>SDCSingle{5}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map</td>
<td>Off</td>
<td>Map</td>
<td>Off</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Dict</td>
<td>Dict</td>
<td>Dict</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>2.6</td>
<td>3,4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0,0</td>
<td></td>
</tr>
</tbody>
</table>

### Overlapping

- **Const**: \(\{0, 1, 2, 3, 4, 5\}\)
- **Dict**: \((-7,-7,-7,-7,-7,-7)\)

### Processed

- **Binary Matrix Scalar Operation**: Subtract 7
Operation Example

Compressed

- **DDC{0}**
  - Map: 0 1 0 1 0 1 1 0 1 0 1 0
  - Dict: 1 7

- **SDC{1,3}**
  - Off: 1 2 0 1 1 2 3 0
  - Map: 0 1 2 0
  - Dict: 2.6 2.5 0.0
  - Def: 3,4

- **SDCZero{2}**
  - Off: 5 1 2 1 1 3 2 1 2 4
  - Map: 0 1 1 0
  - Dict: 9 8.5

- **SDCSingle{5}**
  - Off: 1 1 3 2 4
  - Dict: 4

**A: Overlapping**
- Const: {0, 1, 2, 3, 4, 5}
- Dict: -7,-7,-7,-7,-7,-7

**Binary MatrixScalar Operation:** Subtract 7
Operation Example

Compressed

- DDC\{0\}
  - Map: 0 1 0 1 0 1 1
  - Dict: 1 7

- SDC\{1,3\}
  - Off: 1 2 1 2 3
  - Map: 0 1 1 2 0
  - Dict: 2.6 2.5 0.0
  - Def: 3.4

- SDCZero\{2\}
  - Off: 5 1 1 3
  - Map: 0 1 1 0
  - Dict: 9 8.5

- SDCSingle\{5\}
  - Off: 1 2 2 4
  - Dict: 4

A: Overlapping
- Const\{0, 1, 2, 3, 4, 5\}
- Dict: -7, -7, -7, -7, -7, -7

Binary Matrix Scalar Operation: Subtract 7

B: Processed
- SDCSingleZero\{5\}
  - Dict: -3
- SDCFOR\{2\}
  - Ref: -7
  - Dict\{-4, -3\}
- SDCFOR\{1,3\}
  - Ref: -4, -3
  - Dict\{-1,2\}
  - Def\{-1.1\}
  - Const\{4\}
  - Dict: -7

- DDCFOR\{0\}
  - Ref: -7

- Const\{4\}
  - Dict: -7
### Overlapping Example

<table>
<thead>
<tr>
<th>Compressed</th>
<th>DDC{0}</th>
<th>SDCSingle{0}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map</td>
<td>Dict</td>
<td>Off</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

**Compressed**

- Additive: 5 + 14 + 5 + 14 + 8 + 14 + 5 + 14
- Uncompressed: A set of encodings that overlap with partial sum semantics.
Overlapping Example

A set of encodings that overlap with partial sum semantics.
Overlapping Example

<table>
<thead>
<tr>
<th>Compressed</th>
<th>Additive</th>
<th>Uncompressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDC{0}</td>
<td></td>
<td>5 14</td>
</tr>
<tr>
<td>SDCS{0}</td>
<td></td>
<td>5 14</td>
</tr>
<tr>
<td>Map</td>
<td>Off</td>
<td>1 + 4</td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>2 2 4</td>
<td>7 + 7</td>
</tr>
<tr>
<td>1 7</td>
<td>Dict</td>
<td>1 + 4</td>
</tr>
<tr>
<td>1</td>
<td>Dict</td>
<td>7 + 7</td>
</tr>
<tr>
<td>4</td>
<td>Off</td>
<td>1 + 4</td>
</tr>
<tr>
<td>Def</td>
<td>Off</td>
<td>7 + 7</td>
</tr>
<tr>
<td>7</td>
<td>Off</td>
<td>1 + 4</td>
</tr>
<tr>
<td>1 7</td>
<td>Off</td>
<td>7 + 7</td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>Off</td>
<td>1 + 4</td>
</tr>
<tr>
<td>1 7</td>
<td>Off</td>
<td>7 + 7</td>
</tr>
</tbody>
</table>

A set of encodings that overlap with partial sum semantics.
Overlapping Example

A set of encodings that overlap with partial sum semantics.
Left Matrix Multiplication

Index structures

Dictionaries

PreAggregate

$P_i$

$O_n$

$D_i$

$G_i$

$k$

$n$

$d_i$
Preaggregate

Pre-aggregates per distinct tuple

\[ P[I[j]] + A[j] \]

\[ P \]

\[ A \]

\[ DDC\{1,3\} \]

\[ D_i \]

\[ I_i \]

\[ g_i \]
Right Matrix Multiplication

Right Matrix $P_i\ D_n$

Slice

$k$

$G_i$

$Right_n$

$D_n$ $P_i$

Overlapping Output

Shallow Copy Indexes

Dictionaries

$k$

$G_i$

Shallow Copy Indexes
Datasets (\(n\) Rows, \(m\) Columns, \(sp\) Sparsity).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>(n) (\text{nrow}(X))</th>
<th>(m) (\text{ncol}(X))</th>
<th>(sp)</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airline78</td>
<td>14,462,943</td>
<td>29</td>
<td>0.54</td>
<td>3.4 GB</td>
</tr>
<tr>
<td>Amazon</td>
<td>8,026,324</td>
<td>2,330,066</td>
<td>1.2e-6</td>
<td>1.22 GB</td>
</tr>
<tr>
<td>Covtype</td>
<td>581,012</td>
<td>54</td>
<td>0.22</td>
<td>127 MB</td>
</tr>
<tr>
<td>Mnist1m</td>
<td>1,000,000</td>
<td>784</td>
<td>0.25</td>
<td>2.46 GB</td>
</tr>
<tr>
<td><strong>US Census</strong></td>
<td>2,458,285</td>
<td>68 (378)</td>
<td>0.43 (0.18)</td>
<td>1.34 GB</td>
</tr>
<tr>
<td>US Census 128x</td>
<td>314,660,480</td>
<td>68 (378)</td>
<td>0.43 (0.18)</td>
<td>289.5 GB</td>
</tr>
</tbody>
</table>

**Million column** dataset and **high cocoding potential** scale up dataset
## Compression Time

Local Compression Times [Seconds] and Ratios.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CLA</th>
<th>Aware-Mem</th>
<th>Aware</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td>ratio</td>
<td>time</td>
</tr>
<tr>
<td>Airline78</td>
<td>9.34 sec</td>
<td>10.22</td>
<td>1.74 sec</td>
</tr>
<tr>
<td>Amazon</td>
<td><strong>37.6 hours Crash</strong></td>
<td>8.54 sec</td>
<td>1.73</td>
</tr>
<tr>
<td>Covtype</td>
<td>1.10 sec</td>
<td>13.79</td>
<td>0.84 sec</td>
</tr>
<tr>
<td>Mnist1m</td>
<td>7.25 sec</td>
<td>7.14</td>
<td>4.57 sec</td>
</tr>
<tr>
<td>US Census</td>
<td>5.15 sec</td>
<td>35.38</td>
<td>1.16 sec</td>
</tr>
<tr>
<td>US Census Enc</td>
<td><strong>27.48 sec</strong></td>
<td><strong>41.03</strong></td>
<td><strong>5.78 sec</strong></td>
</tr>
</tbody>
</table>

**Intelligent compression abort** based on workload and Faster compression
Matrix Multiplication

(a) 16 row LMM

(b) LMM Census Enc Scaling

(c) 16 col RMM

(d) RMM Census Enc Scaling
Matrix Multiplication

(e) 16 row LMM

(f) LMM Census Enc Scaling
Matrix Multiplication

(i) 16 row LMM

(j) LMM Census Enc Scaling

(k) 16 col RMM
Matrix Multiplication

(m) 16 row LMM

(o) 16 col RMM

(n) LMM Census Enc Scaling

(p) RMM Census Enc Scaling
Operation Sequences

(q) Scale and Shift Sparse

Operation sequences maintain compressed formats and
Operation Sequences

Operation sequences maintain compressed formats and Decompressing sequences are competitive or faster than uncompressed.
### Workload-awareness on Local End-to-End Algorithms (Data: US Census Enc)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>ULA Time</th>
<th>Aware-Mem Comp</th>
<th>Aware Time</th>
<th>Aware Comp</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-Means</td>
<td>51.6 sec</td>
<td>4.2 sec</td>
<td>46.2 sec</td>
<td>6.2 sec</td>
</tr>
<tr>
<td>PCA</td>
<td>12.7 sec</td>
<td>4.0 sec</td>
<td>10.4 sec</td>
<td>6.0 sec</td>
</tr>
<tr>
<td>MLogReg</td>
<td>32.0 sec</td>
<td>4.5 sec</td>
<td>32.5 sec</td>
<td>7.2 sec</td>
</tr>
<tr>
<td>lmCG</td>
<td>19.8 sec</td>
<td>5.0 sec</td>
<td>20.7 sec</td>
<td>6.4 sec</td>
</tr>
<tr>
<td>lmDS</td>
<td>15.6 sec</td>
<td>5.7 sec</td>
<td>15.5 sec</td>
<td>6.1 sec</td>
</tr>
<tr>
<td>L2SVM</td>
<td>38.9 sec</td>
<td>6.5 sec</td>
<td>45.2 sec</td>
<td>6.2 sec</td>
</tr>
</tbody>
</table>

**AWARE** is able to amortize the cost of **online compression** in short jobs.
End-to-End Algorithms Hybrid Execution


<table>
<thead>
<tr>
<th></th>
<th>K-Means</th>
<th></th>
<th></th>
<th>MLogReg</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ULA</td>
<td>Aware</td>
<td>ULA</td>
<td>Aware</td>
</tr>
<tr>
<td>1x</td>
<td>51.6</td>
<td>(6) 27.1</td>
<td>12.7</td>
<td>(6) 9.4</td>
</tr>
<tr>
<td>8x</td>
<td>471.0</td>
<td>(26) 117.8</td>
<td>330.3</td>
<td>(26) 42.6</td>
</tr>
<tr>
<td>16x</td>
<td>484.3</td>
<td>(48) 183.9</td>
<td>76.3</td>
<td>(47) 67.5</td>
</tr>
<tr>
<td>32x</td>
<td>1,491.6</td>
<td>1,496.3</td>
<td>70.3</td>
<td>61.2</td>
</tr>
<tr>
<td>128x</td>
<td>17,819.0</td>
<td>6,298.0</td>
<td>137.0</td>
<td>140.3</td>
</tr>
<tr>
<td>*128x</td>
<td>33,039.0</td>
<td>11,616.0</td>
<td>269.0</td>
<td>259.0</td>
</tr>
</tbody>
</table>

**AWARE** keeps operations local, since they fit in memory.
TensorFlow Comparison

Our baseline is competitive, and AWARE is the fastest.

ImCG (Data: US Census Enc)
GridSearch MLogReg (Data: US Census Enc).

<table>
<thead>
<tr>
<th>ULA</th>
<th>Aware-Mem</th>
<th>Aware</th>
</tr>
</thead>
<tbody>
<tr>
<td>274.3 sec</td>
<td>238.1 sec</td>
<td><strong>92.6 sec</strong></td>
</tr>
</tbody>
</table>

Aware reduce **memory bandwidth** and reduce **compute** requirements.
Conclusions

- Optimize for **execution cost** not size of data
- Use Compressed Linear Algebra for **redundancy exploitation**, it is the natural next step from **sparsity exploitation**
- Workload awareness exploit **workload and data** characteristics together
Conclusions

- Optimize for **execution cost** not size of data
- Use Compressed Linear Algebra for **redundancy exploitation**, it is the natural next step from **sparsity exploitation**
- Workload awareness exploit **workload and data** characteristics together

**Future Work**

- Compressed **feature transformation** of heterogeneous data
- Orthogonal **lossy compression**