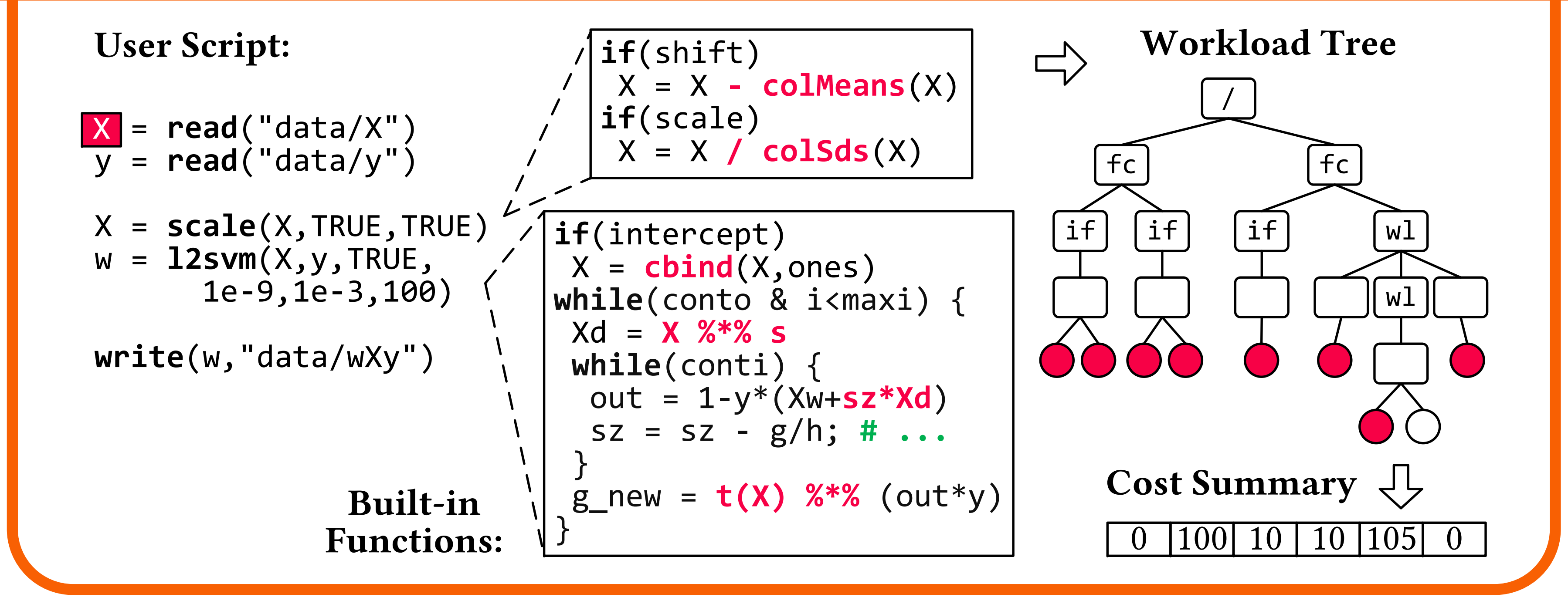
AWARE: Workload-aware, Redundancy-exploiting Linear Algebra

Sebastian Baunsgaard & Matthias Boehm
Technische Universität Berlin

MOTIVATION

- The next step from **Sparsity Exploitation** → **Redundancy Exploitation**
- We changed the compression goal from the norm of **Compression Ratio** to optimizing for **Execution Time** in a workload-aware manner
- Guaranteed **same results** as uncompressed data via **Lightweight Database Compression Techniques** and **Compressed Linear Algebra**
- Improved performance of individual operations by up to **10,000x**!
- Improved algorithmic performance including everything in end-to-end pipelines including **Online Compression** and algorithm!
- Grid search algorithms improving from **274.3 sec** to **92.6 sec** on same hardware due to **reduced memory bandwidth requirements** and faster **direct compressed operations**.

WORKLOAD EXTRACTION



User Script:

```
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:

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

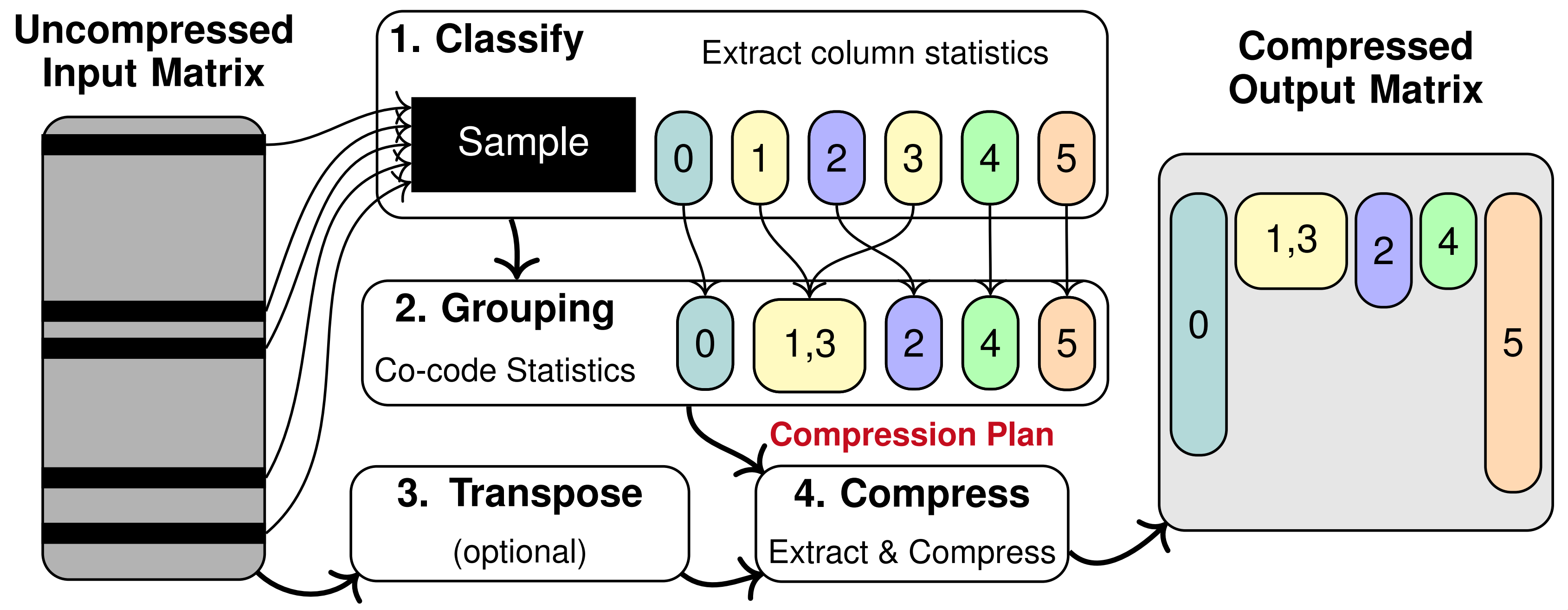
```
if(intercept)
 X = cbind(X,ones)
while(conto & i<maxi) {
 Xd = X %*% s
 while(conti) {
  out = 1-y*(Xw+sz*Xd)
  sz = sz - g/h; # ...
 }
 g_new = t(X) %*% (out*y)
}
```

Workload Tree

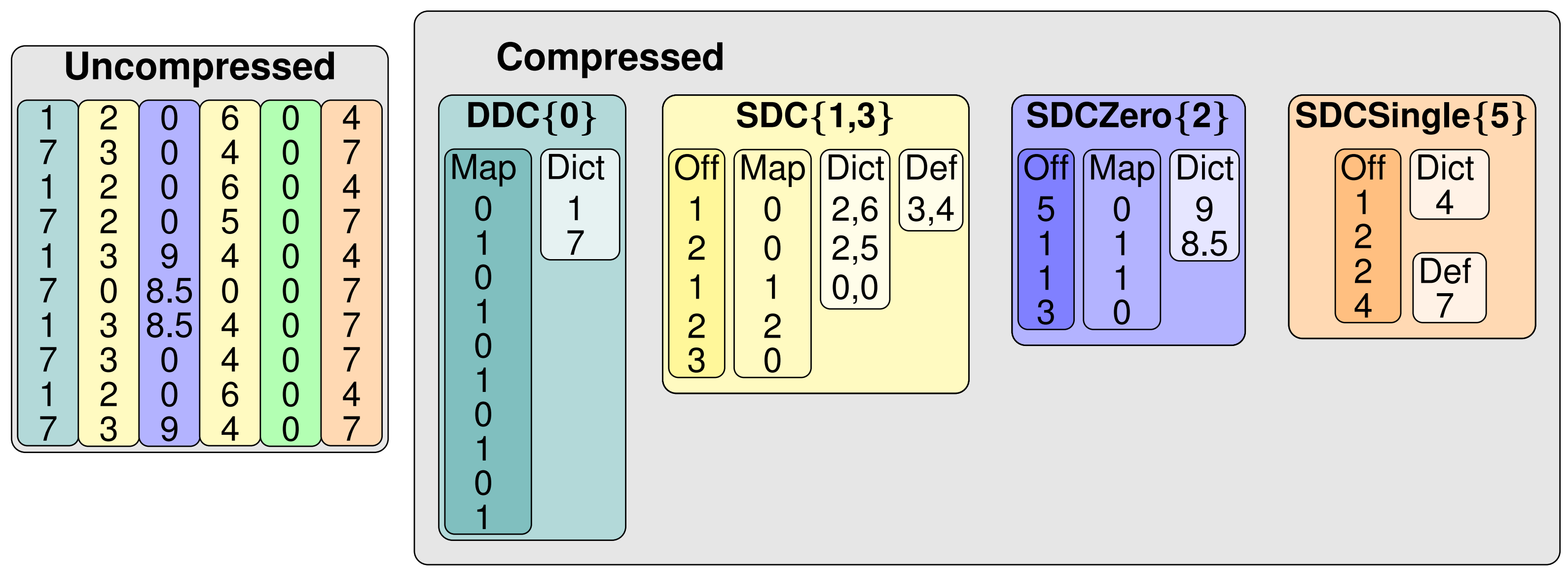
Cost Summary

0	100	10	10	105	0

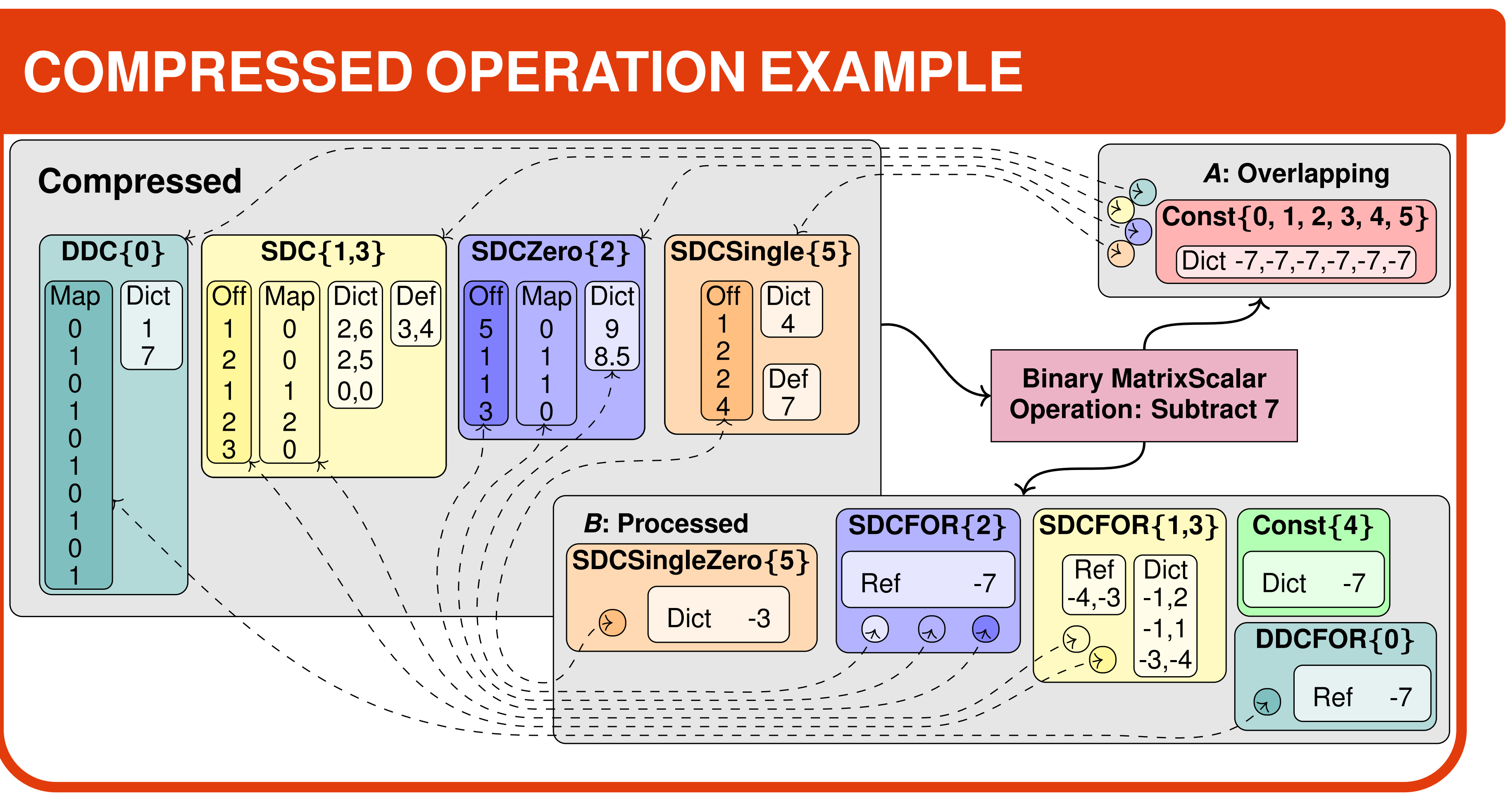
COMPRESSION WORKFLOW



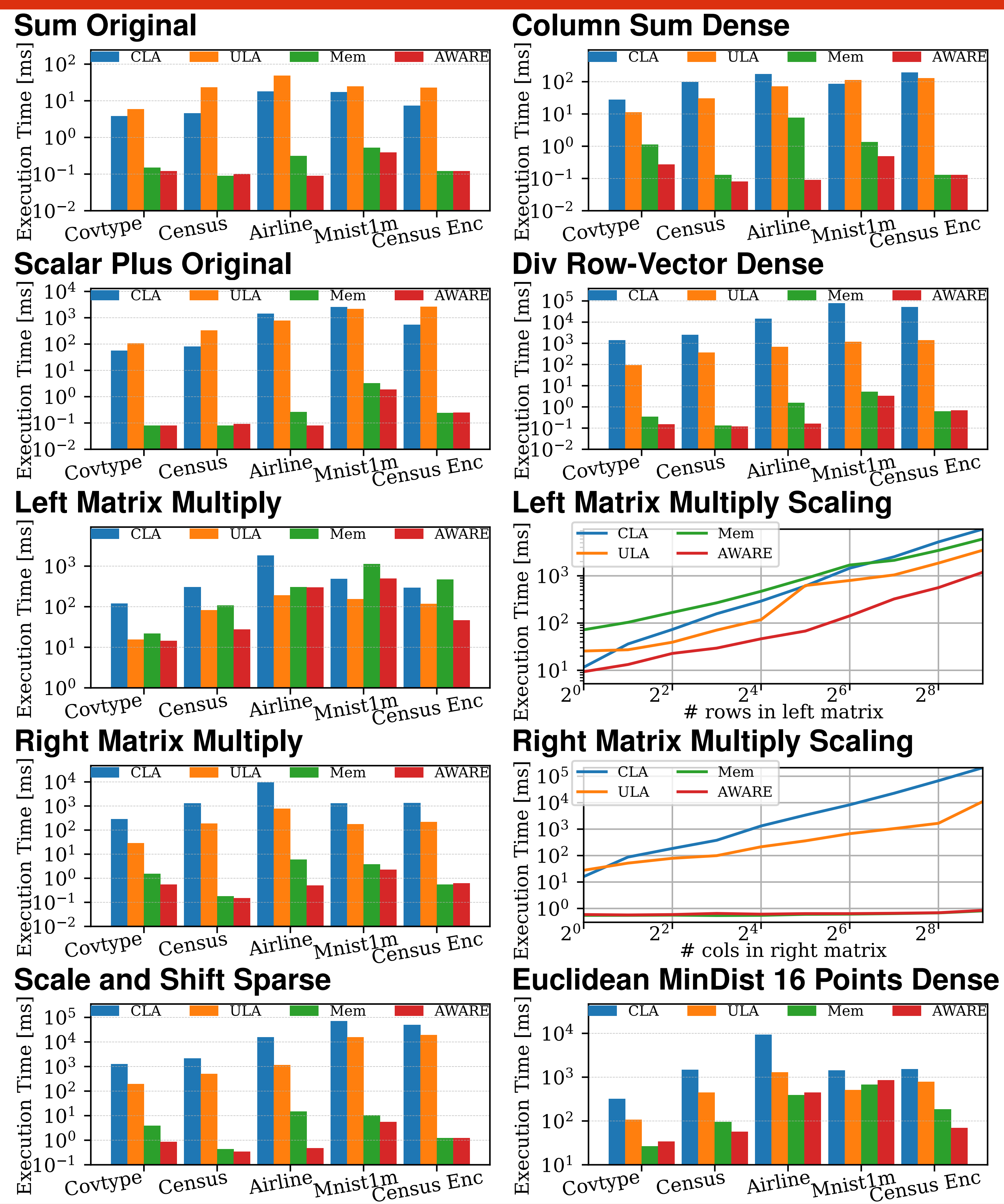
COMPRESSION EXAMPLE



COMPRESSED OPERATION EXAMPLE



OPERATIONS PERFORMANCE



Sum Original

Column Sum Dense

Scalar Plus Original

Div Row-Vector Dense

Left Matrix Multiply

Left Matrix Multiply Scaling

Right Matrix Multiply

Right Matrix Multiply Scaling

Scale and Shift Sparse

Euclidean MinDist 16 Points Dense

LOCAL END-TO-END EXPERIMENTS

Workload-awareness on Local End-to-End Algorithms (Data: US Census Enc)

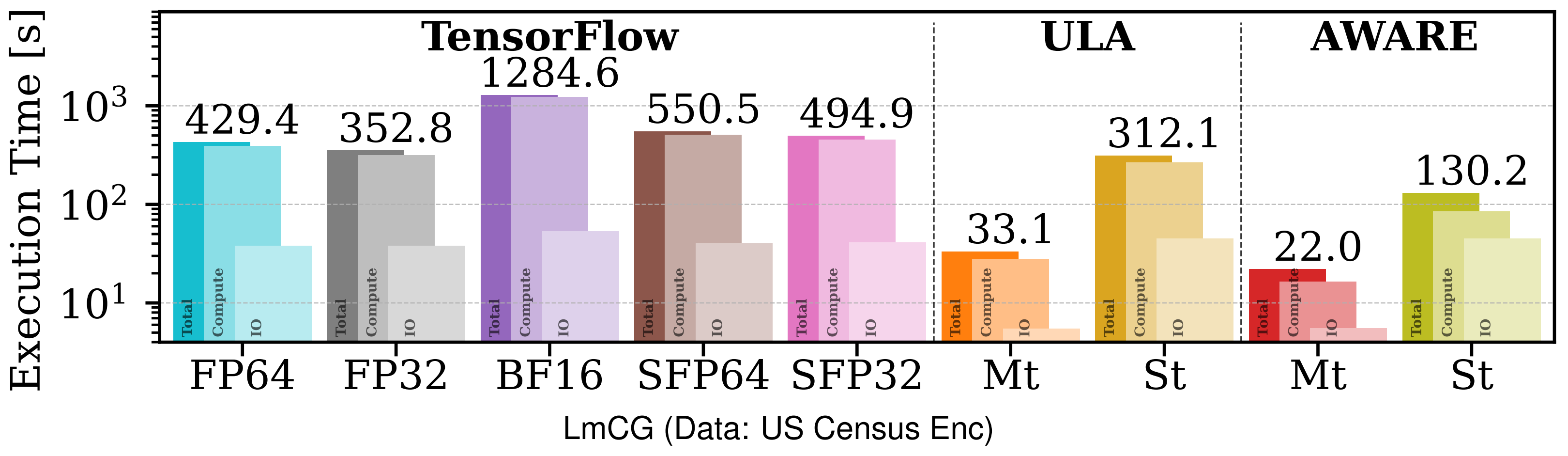
	ULA	*Aware*-Mem		*Aware*	
	Time	Comp	Time	Comp	Time
K-Means	51.6 sec	4.2 sec	46.2 sec	6.2 sec	27.1 sec
PCA	12.7 sec	4.0 sec	10.4 sec	6.0 sec	9.0 sec
MLogReg	32.0 sec	4.5 sec	32.5 sec	7.2 sec	26.0 sec
lmCG	19.8 sec	5.0 sec	20.7 sec	6.4 sec	18.6 sec
lmDS	15.6 sec	5.7 sec	15.5 sec	6.1 sec	14.3 sec
L2SVM	38.9 sec	6.5 sec	45.2 sec	6.2 sec	36.5 sec

HYBRID END-TO-END EXPERIMENTS

Hybrid End-to-End [Sec] (Data: US Census Enc, *D* .. Incl. Distributed Ops)

	K-Means		PCA		MLogReg		lmCG	
	ULA	*Aware*	ULA	*Aware*	ULA	*Aware*	ULA	*Aware*
1x	51.6	(6) 27.1	12.7	(6) 9.4	32.0	(7) 26.0	19.8	(6) 18.6
8x	471.0	(26) 117.8	330.3	(26) 42.6	393.3	(29) 88.2	366.2	(26) 60.6
16x	^{D}484.3	(48) 183.9	^{D}76.3	(47) 67.5	^{D}570.3	(58) 144.2	^{D}104.4	(44) 91.7
32x	^{D}1,491.6	^{D}1,496.3	^{D}70.3	^{D}61.2	^{D}671.5	^{D}629.9	^{D}264.6	^{D}105.3
128x	^{D}17,819.0	^{D}6,298.0	^{D}137.0	^{D}140.3	^{D}3,502.9	^{D}1,710.6	^{D}1,611.4	^{D}242.6
*128x	^{D}33,039.0	^{D}11,616.0	^{D}269.0	^{D}259.0	^{D}50,998.0	^{D}8,599.6	^{D}33,090.0	^{D}469.0

TENSORFLOW COMPARISON



LmCG (Data: US Census Enc)

SystemDS
github.com/apache/systemds

Paper
dl.acm.org/doi/10.1145/3588682

Reproducibility
github.com/damslab/reproducibility

Sebastian Baunsgaard and Matthias Boehm | baunsgaard@tu-berlin.de & matthias.boehm@tu-berlin.de | SIGMOD 2023