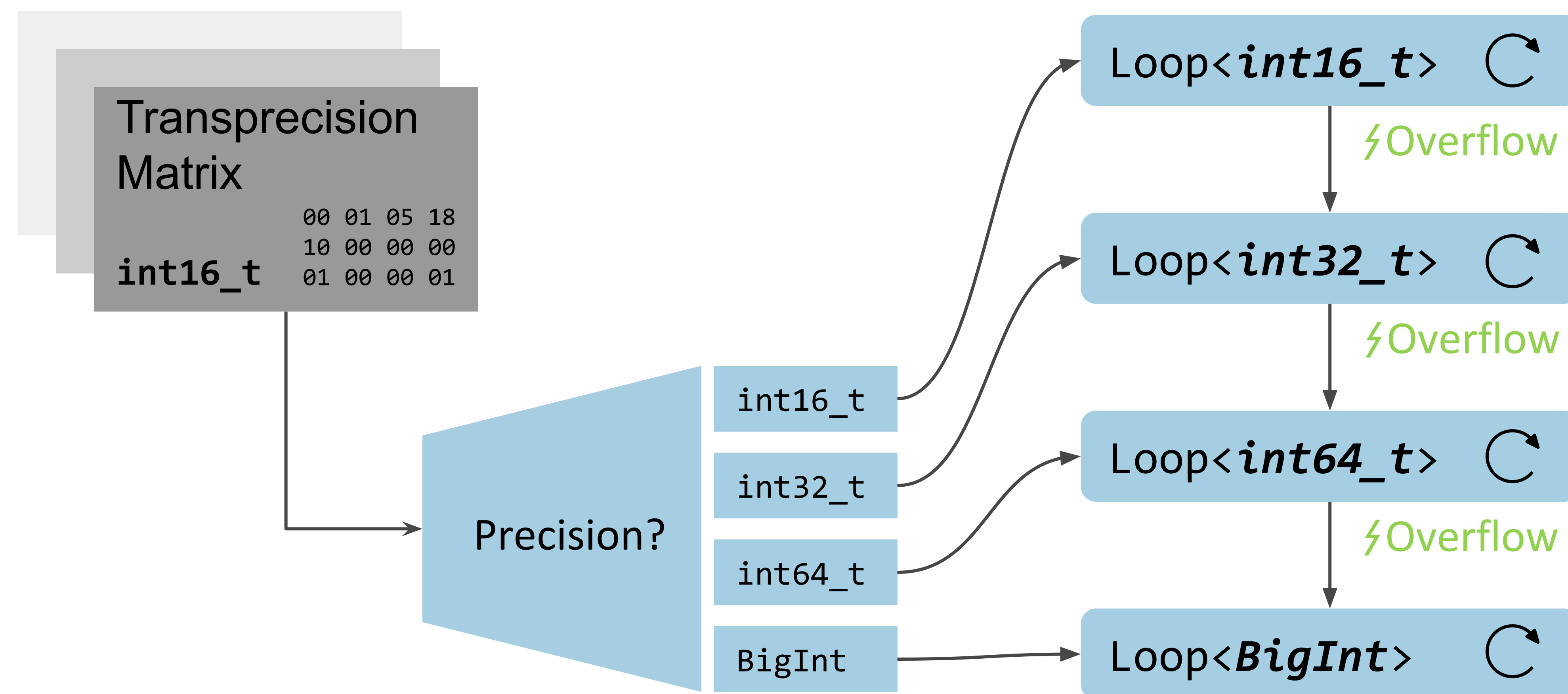


Fast Linear Programming through Transprecision Computing on Small and Sparse Data

Tobias Grosser, Theodoros Theodoridis, Arjun Pitchanathan, Maximilian Falkenstein, Michael Kruse, Manuel Riger, Zhendong Su, Torsten Hoefer

University of Edinburgh, ETH Zurich, Argonne National Labs, IIIT Hyderabad



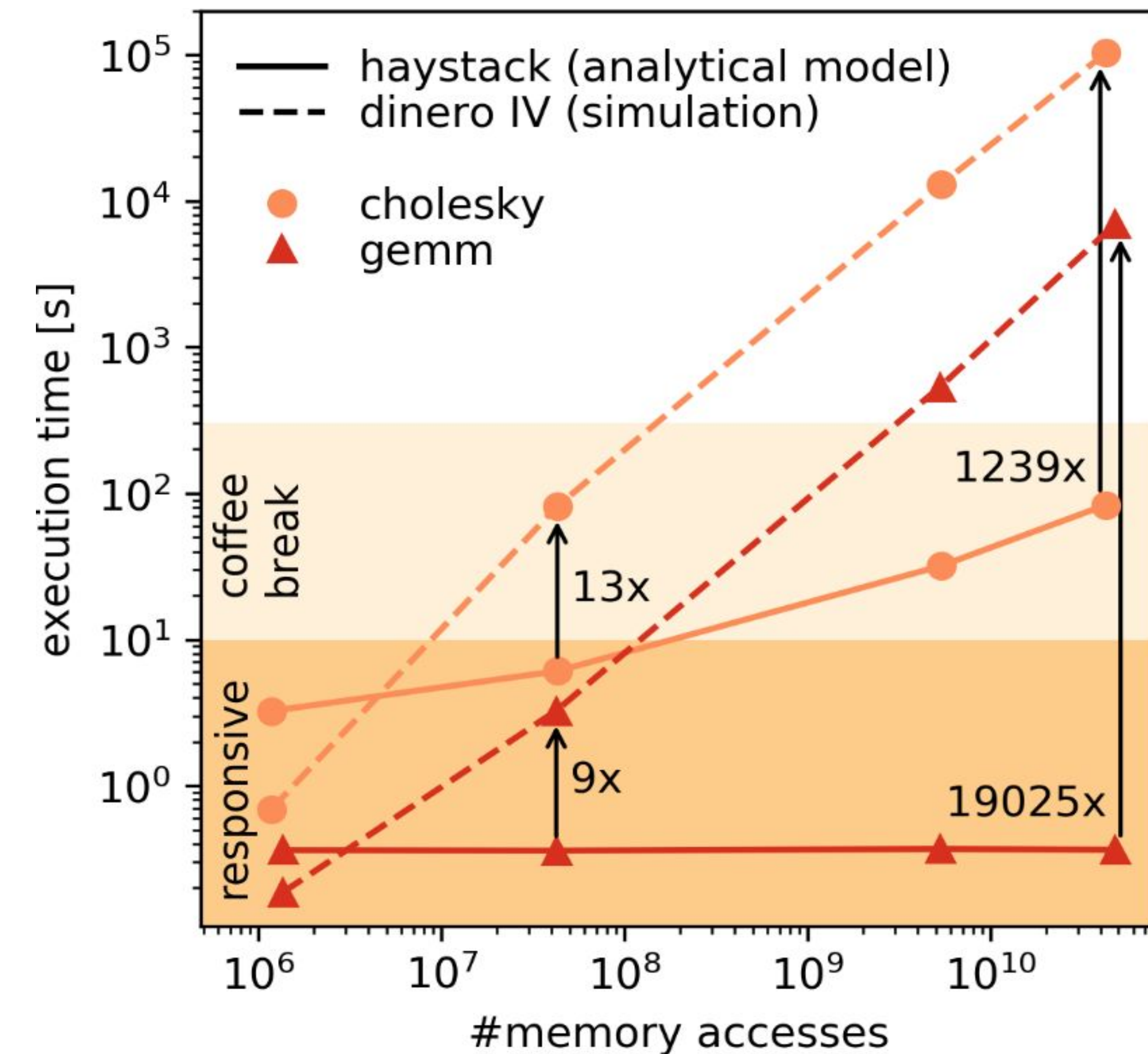
A Fast Cache Model

```

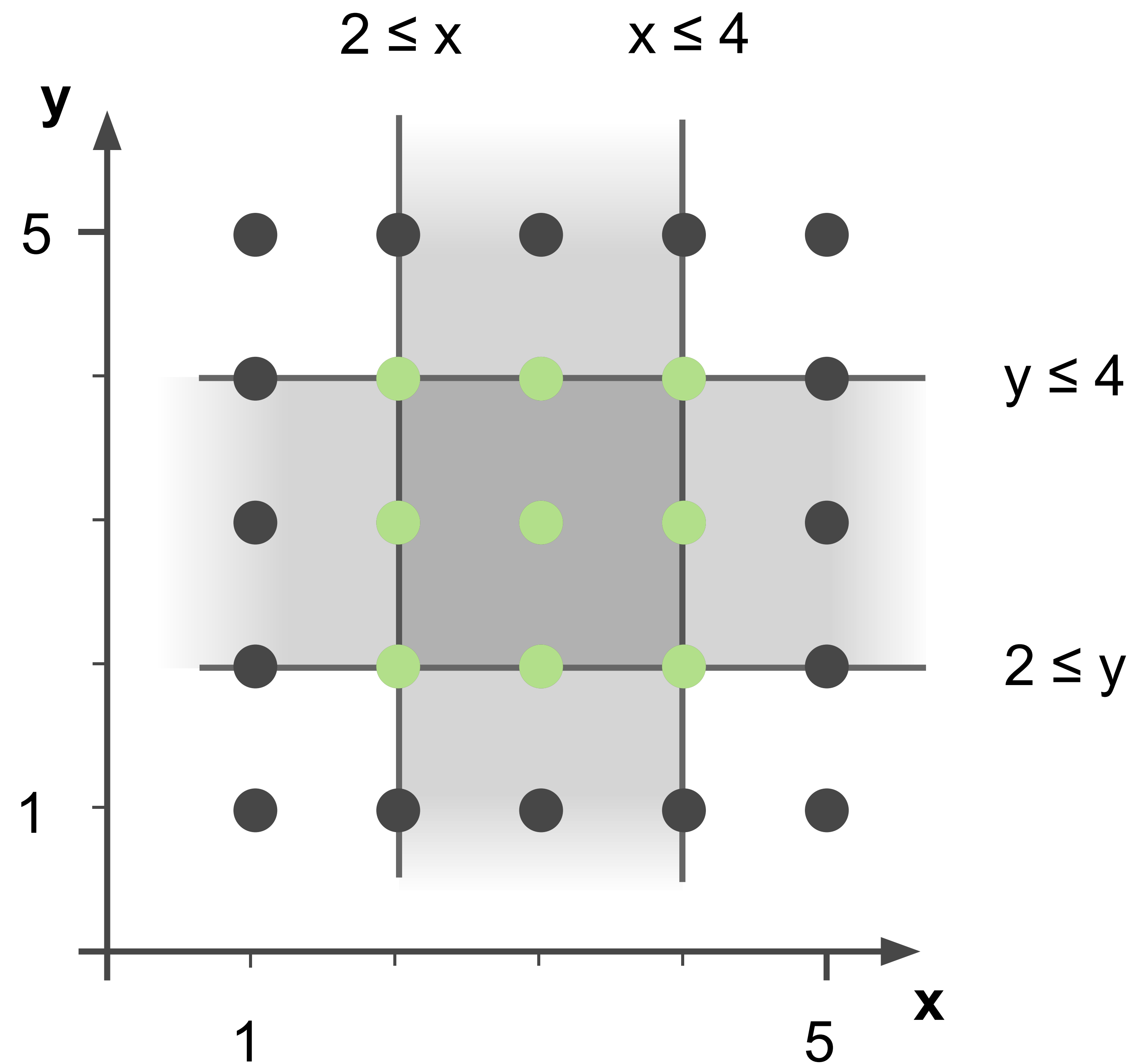
for (i = 0; i < _PB_NI; i++) {
  for (j = 0; j < _PB_NJ; j++)
    C[i][j] *= beta;
  for (k = 0; k < _PB_NK; k++)
    for (j = 0; j < _PB_NJ; j++)
      C[i][j] += alpha * A[i][k] * B[k][j];
}

```

memory access cost?
benefits of tiling?



Integer Sets



$$\mathbf{S} = \{ (x, y) \mid 2 \leq x \leq 4 \wedge 2 \leq y \leq 4 \}$$

Integer Set Operations

$$\begin{array}{cc} \bullet & \bullet \\ \bullet & \bullet \end{array} \cap \begin{array}{cc} \bullet & \bullet \\ \bullet & \bullet \end{array} = \begin{array}{cc} \bullet & \bullet \\ \bullet & \bullet \end{array}$$

$$\text{empty} \begin{array}{cc} \bullet & \bullet \\ \bullet & \bullet \end{array} = F$$

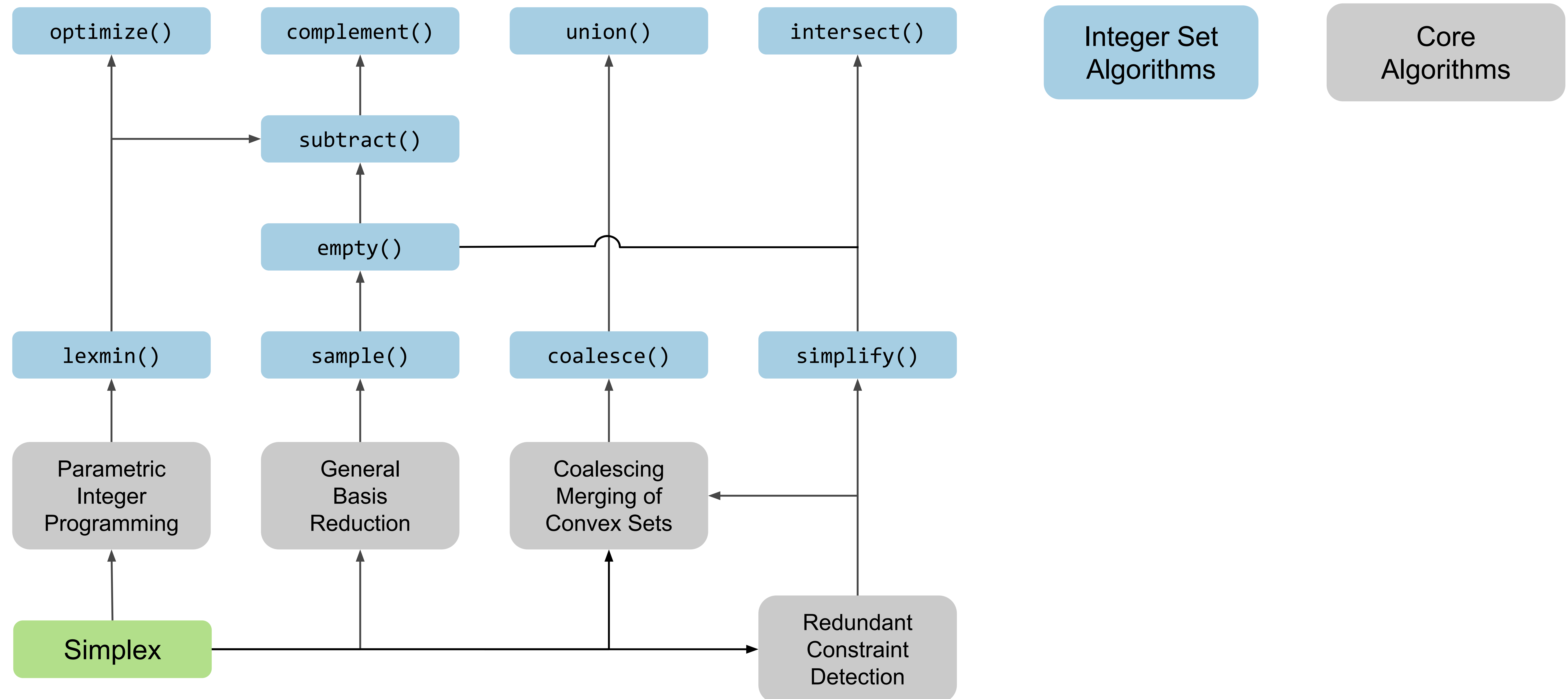
$$\begin{array}{cc} \bullet & \bullet \\ \bullet & \bullet \end{array} \cup \begin{array}{cc} \bullet & \bullet \\ \bullet & \bullet \end{array} = \begin{array}{cc} \bullet & \bullet \\ \bullet & \bullet \end{array}$$

$$\text{sample} \begin{array}{cc} \bullet & \bullet \\ \bullet & \bullet \end{array} = \begin{array}{cc} \bullet & \bullet \\ \bullet & \bullet \end{array}$$

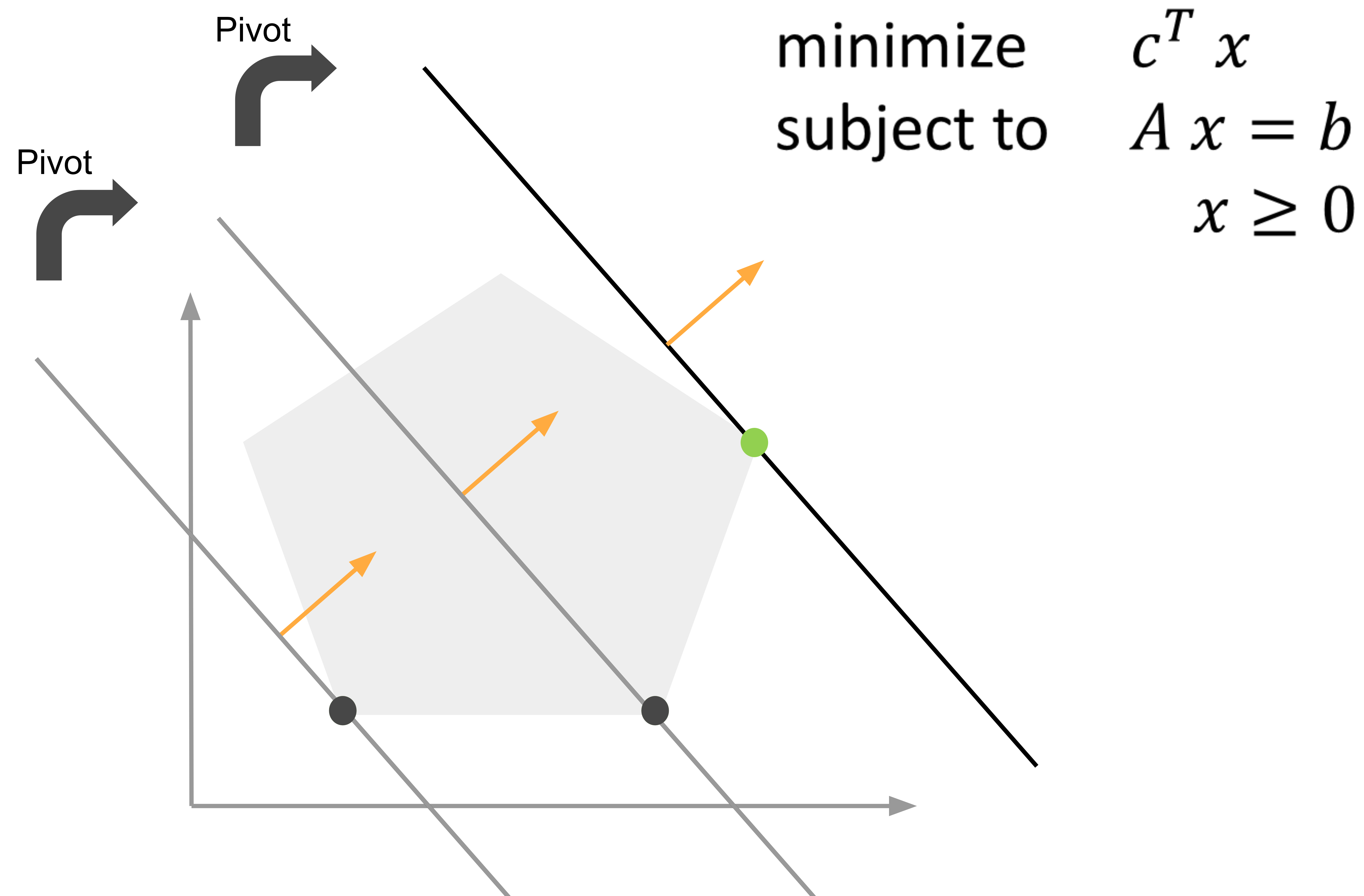
$$\begin{array}{cc} \bullet & \bullet \\ \bullet & \bullet \end{array} \setminus \begin{array}{cc} \bullet & \bullet \\ \bullet & \bullet \end{array} = \begin{array}{cc} \bullet & \bullet \\ \bullet & \bullet \end{array}$$

$$\neg \begin{array}{cc} \bullet & \bullet \\ \bullet & \bullet \end{array} = \begin{array}{cc} \bullet & \bullet \\ \bullet & \bullet \end{array}$$

Simplex at the Heart of Integer Set Arithmetic



Simplex: A Sequence of Pivot Steps



Pivot – Row Operations over a (Dense) Matrix

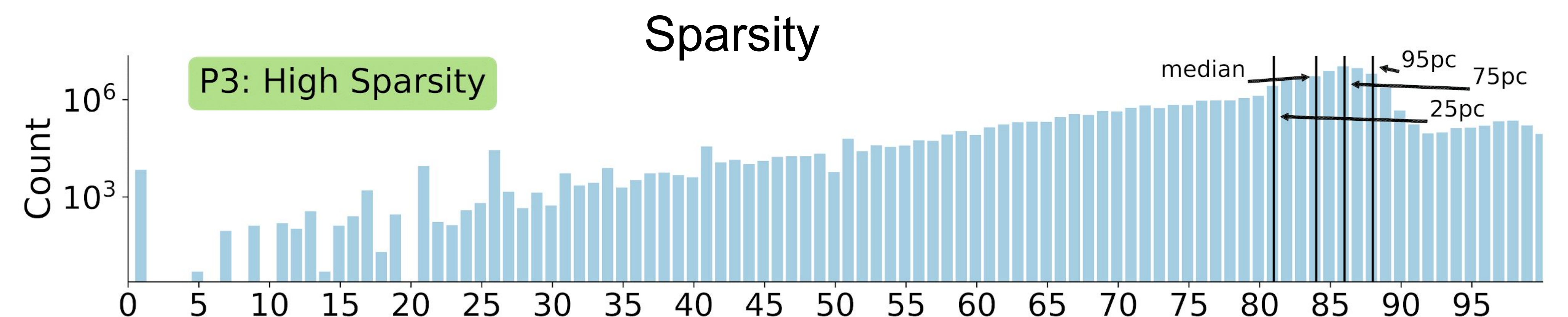
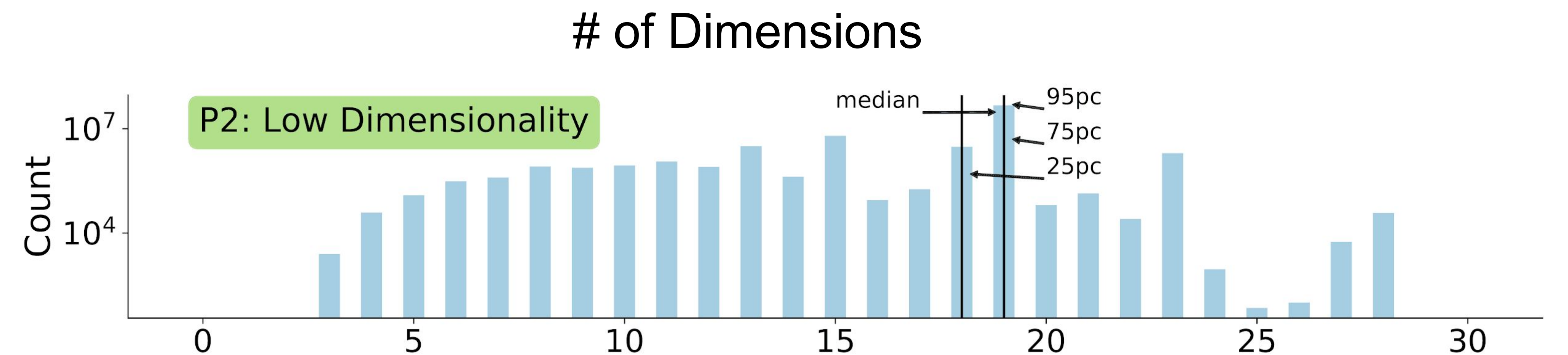
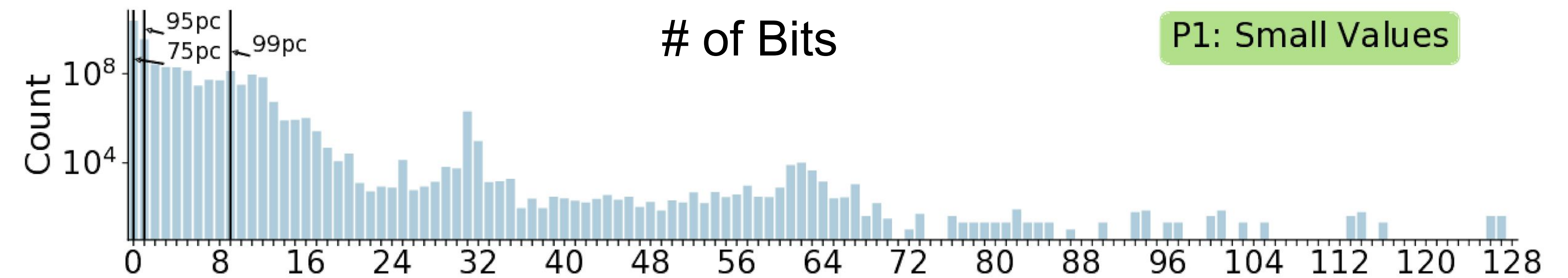
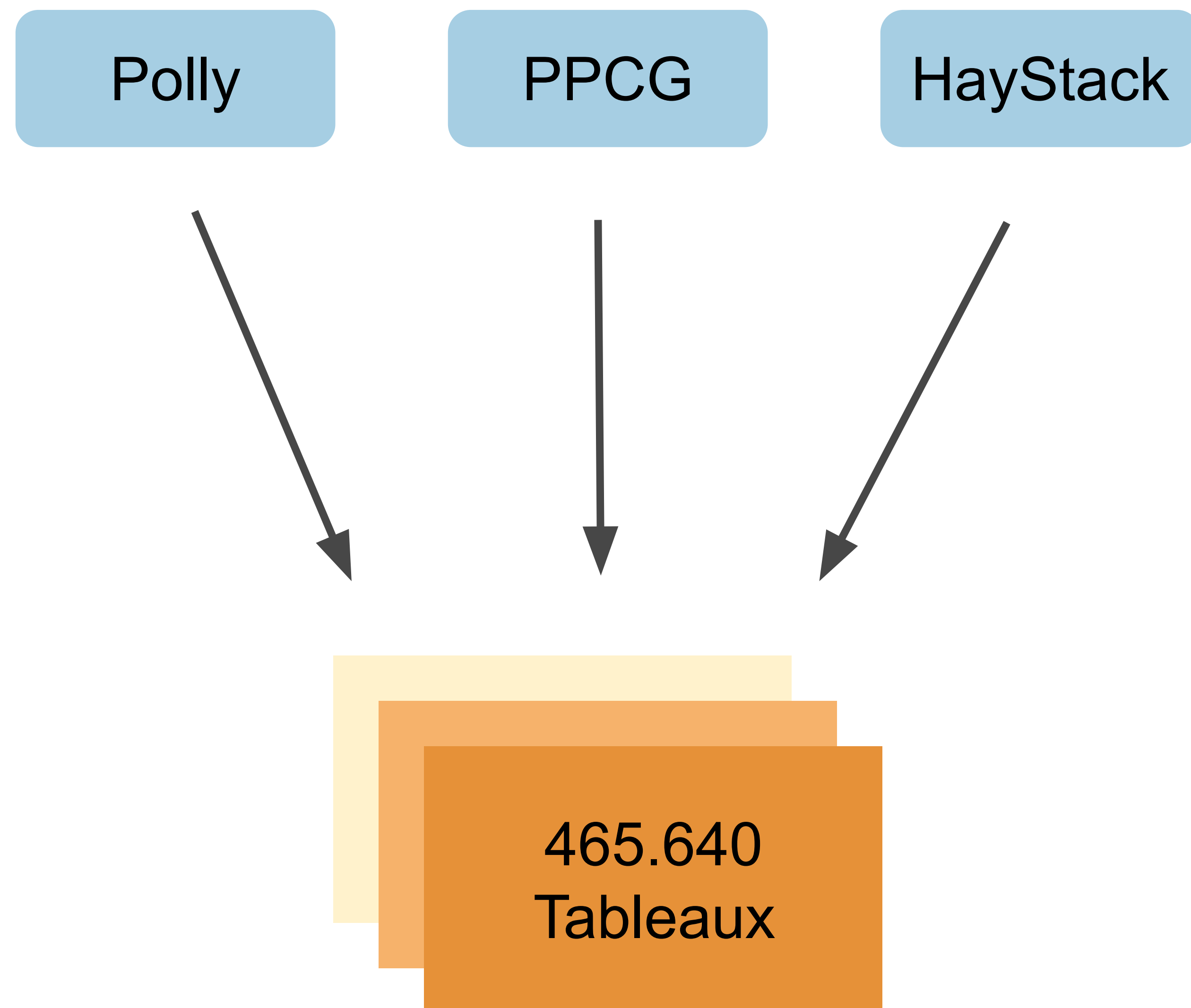
	Pivot column	Other columns
Pivot row	a	b
Other rows	y	d



	Pivot column	Other columns
Pivot row	$1/a$	$-b/a$
Other rows	y/a	$d-(by)/a$

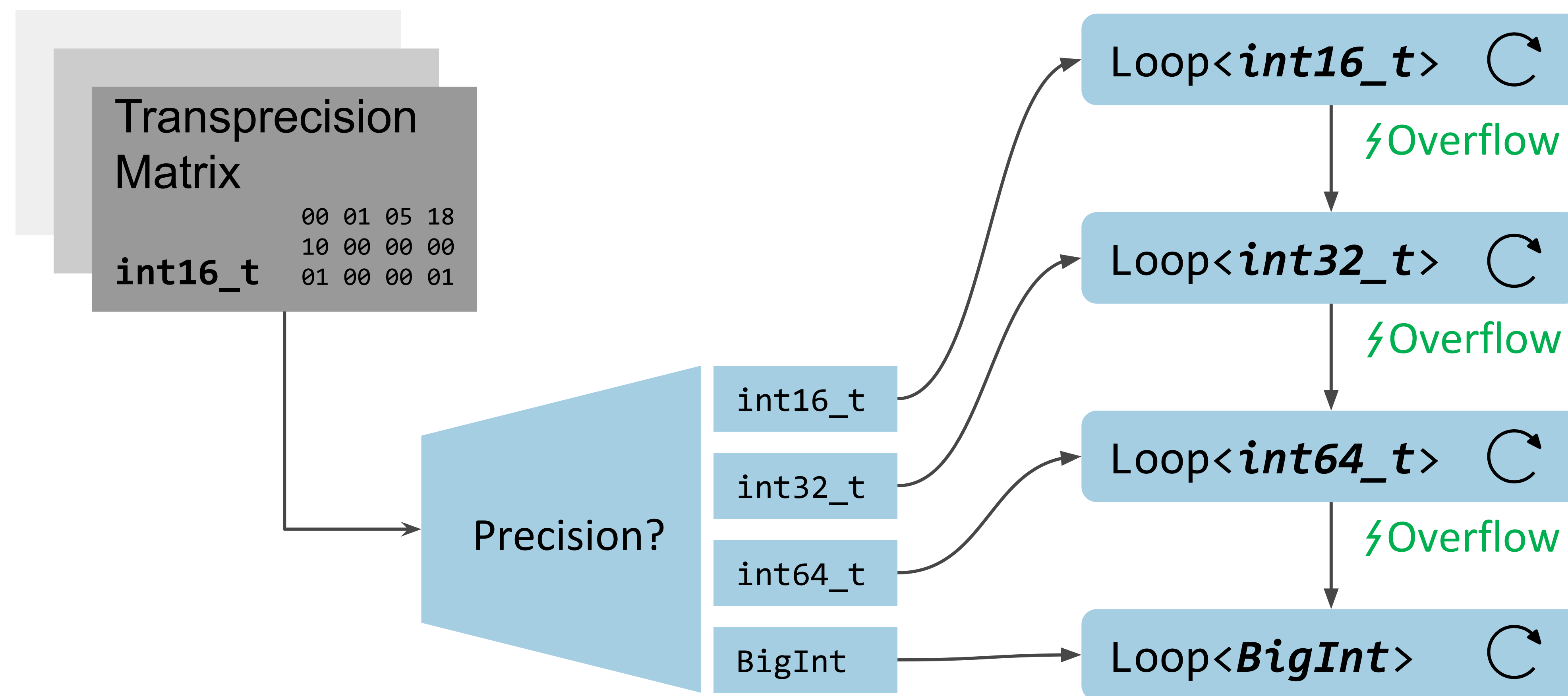
Fast Linear Programming

Characteristics Of Simplex Tableaux in Compilers



Transprecision Computing

Small Values



Implemented with C++14
Meta-Programming

Heterogeneity of SIMD Instruction Sets (AVX-512)

Low Dimensionality

Small Values

Inst. Type	8-bit	16-bit	32-bit	64-bit
Mul	×	✓	✓	✓
Mul High	×	✓	✓	×
Add	✓	✓	✓	✓
Add (Sat.)	✓	✓	×	×



Fast overflow checks

Keeping Coefficients Small: Normalization

Low Dimensionality

Small Values

Den	d0	d1	d2	d3	c
6	12	18	9	24	36

GCD: 3

Keeping Coefficients Small: Normalization

Low Dimensionality

Small Values

Den	d0	d1	d2	d3	c
$6 / 3 = 2$	$12 / 3 = 4$	$18 / 3 = 6$	$9 / 3 = 3$	$24 / 3 = 8$	$36 / 3 = 12$

GCD: 3

Keeping Coefficients Small: Normalization

Low Dimensionality

Small Values

Den	d0	d1	d2	d3	c
$6 / 3 = 2$	$12 / 3 = 4$	$18 / 3 = 6$	$9 / 3 = 3$	$24 / 3 = 8$	$36 / 3 = 12$

GCD: 3

GCD requires div/mod -> slow

Prime GCD for SIMD & Small Numbers

High-Sparsity → Paper

Low Dimensionality

Small Values

Column	Base 10	Prime-Base
Den	6	2 2 2 3 3
d0	12	2 2 2 3 3
d1	18	2 2 2 3 3
d2	9	2 2 2 3 3
d3	24	2 2 2 3 3
c	36	2 2 2 3 3
		AND
GCD	3	2 2 2 3 3

Pre-Computed
Prime-Base Tables

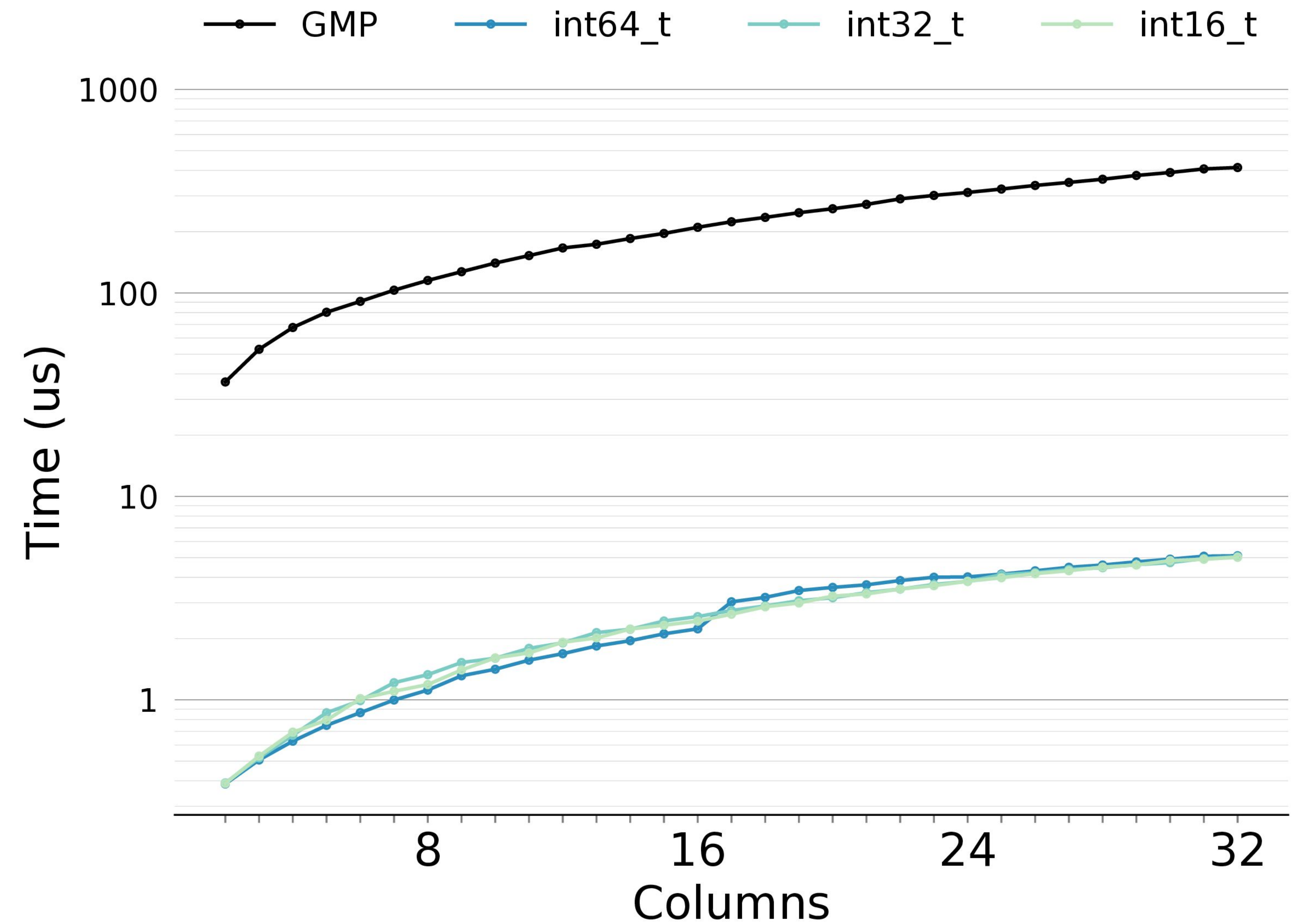
SIMDization with
Horizontal “AND”

Divisions Is
SIMD XOR

Evaluation

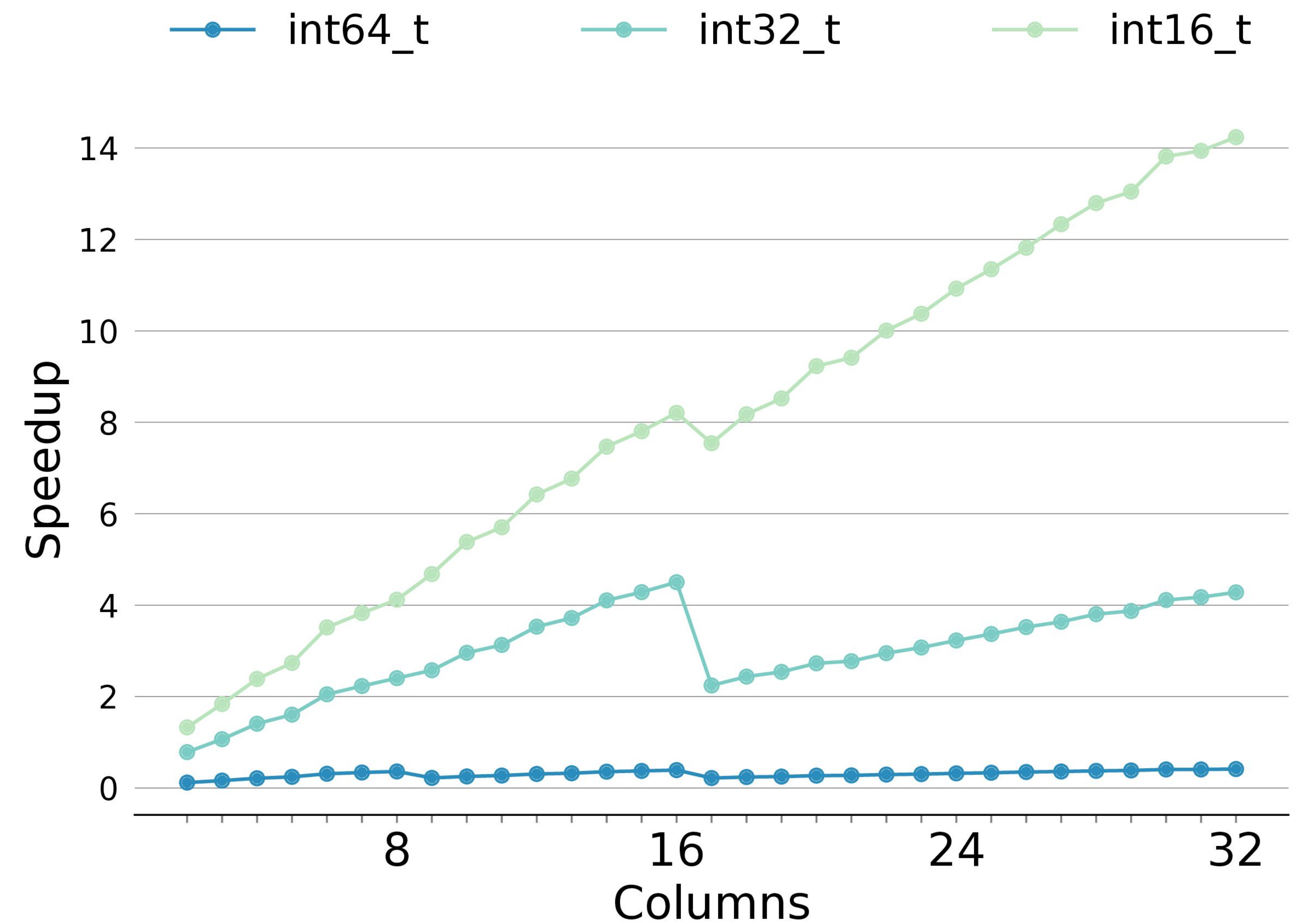
Accelerating the Simplex Pivot Loop

- Arbitrary Precision is Costly



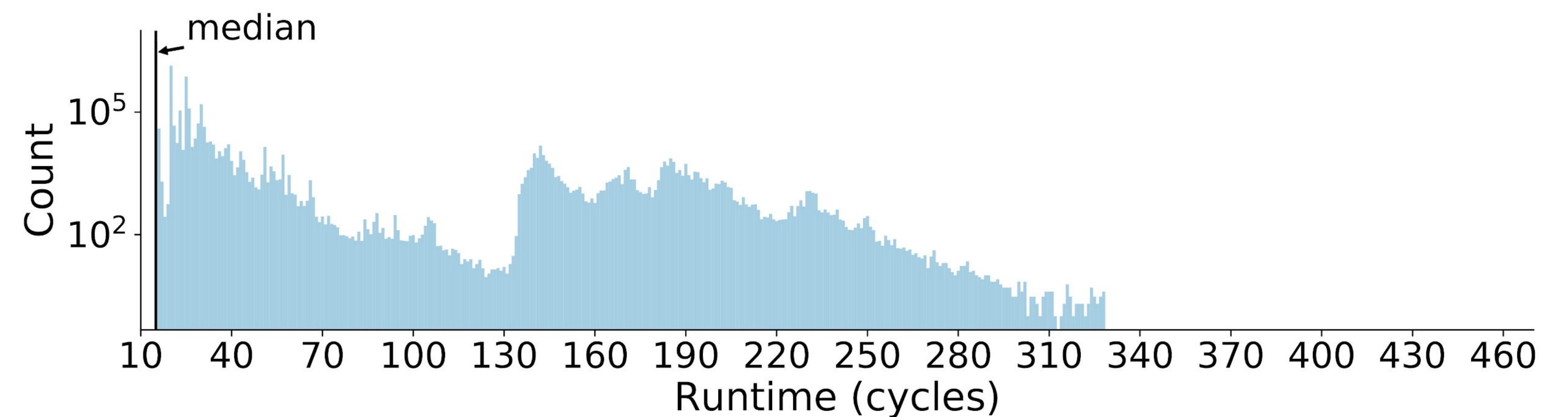
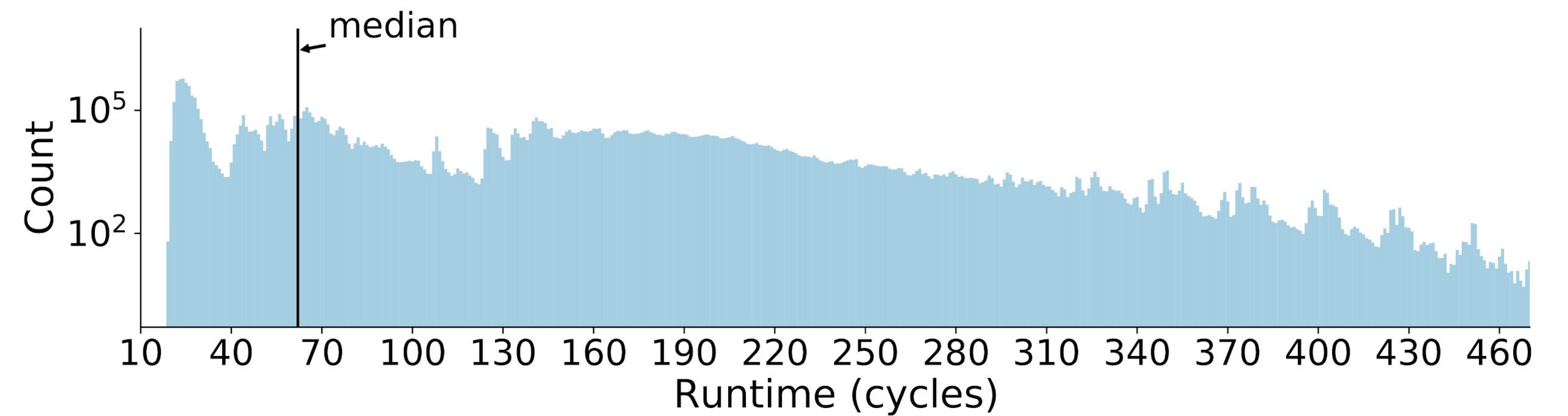
Accelerating the Simplex Pivot Loop

- Arbitrary Precision is Costly
- SIMD Scales for 16 and 32 bit



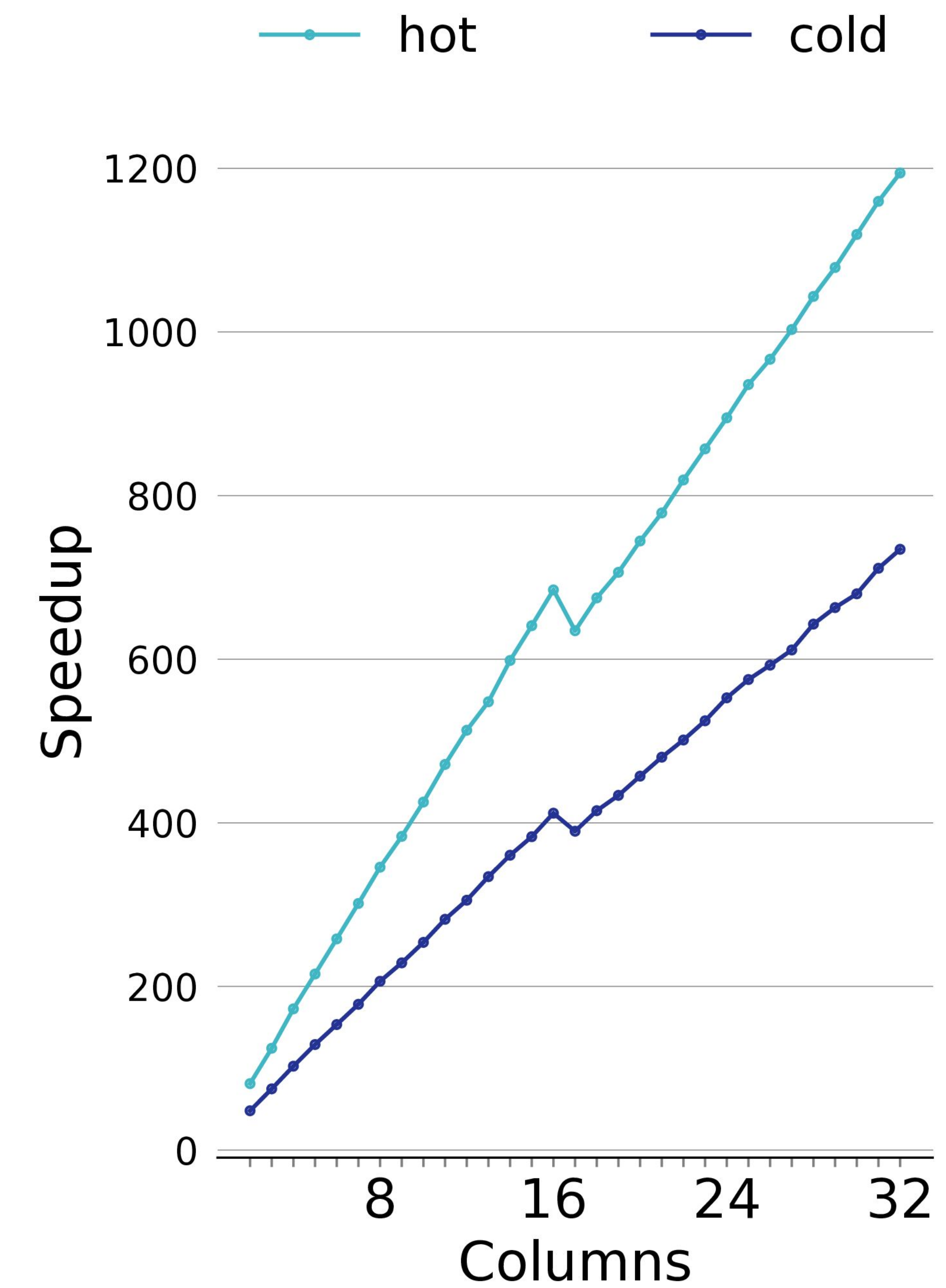
Accelerating the Simplex Pivot Loop

- Arbitrary Precision is Costly
- SIMD Scales for 16 and 32 bit
- Prime-Factor GCD is Fast

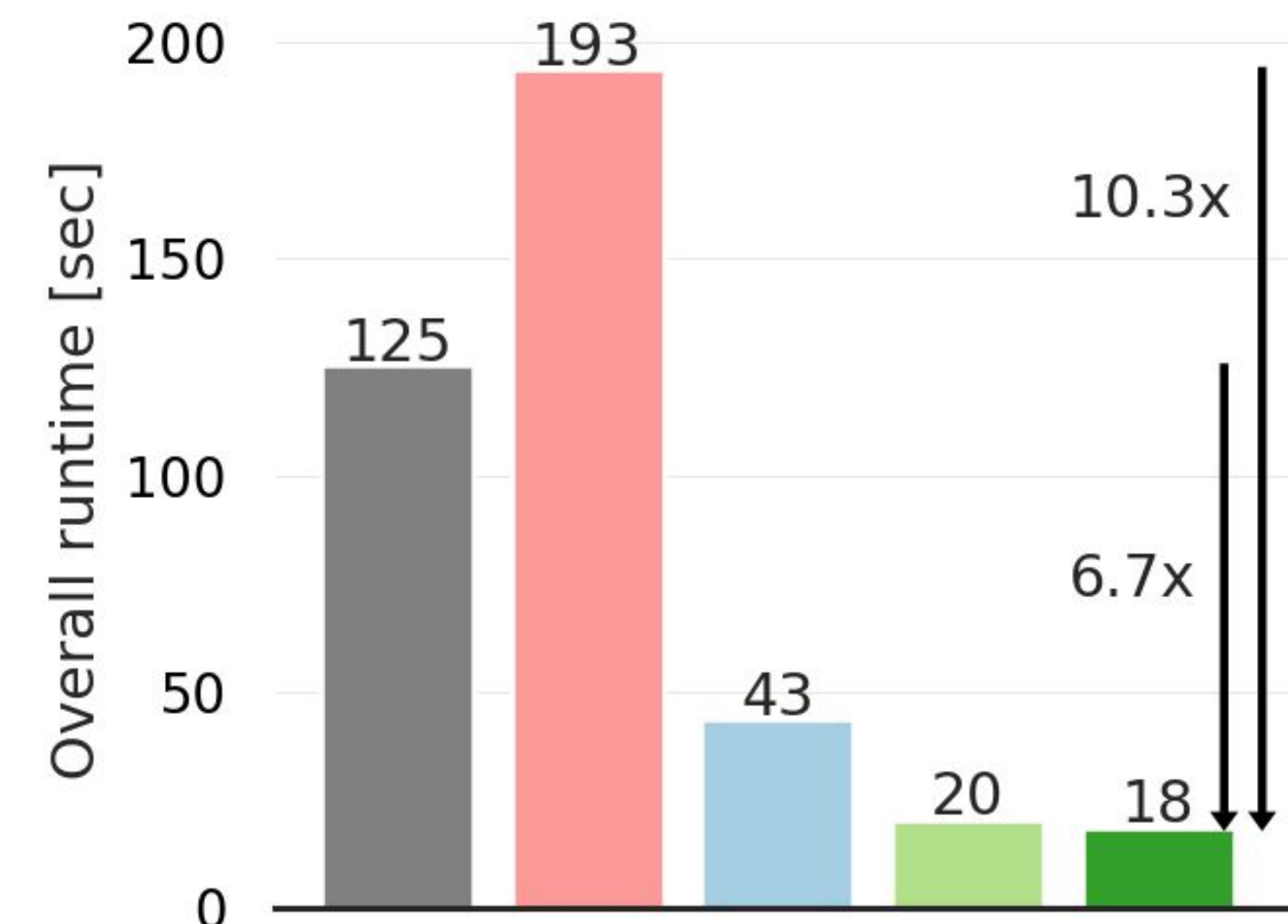
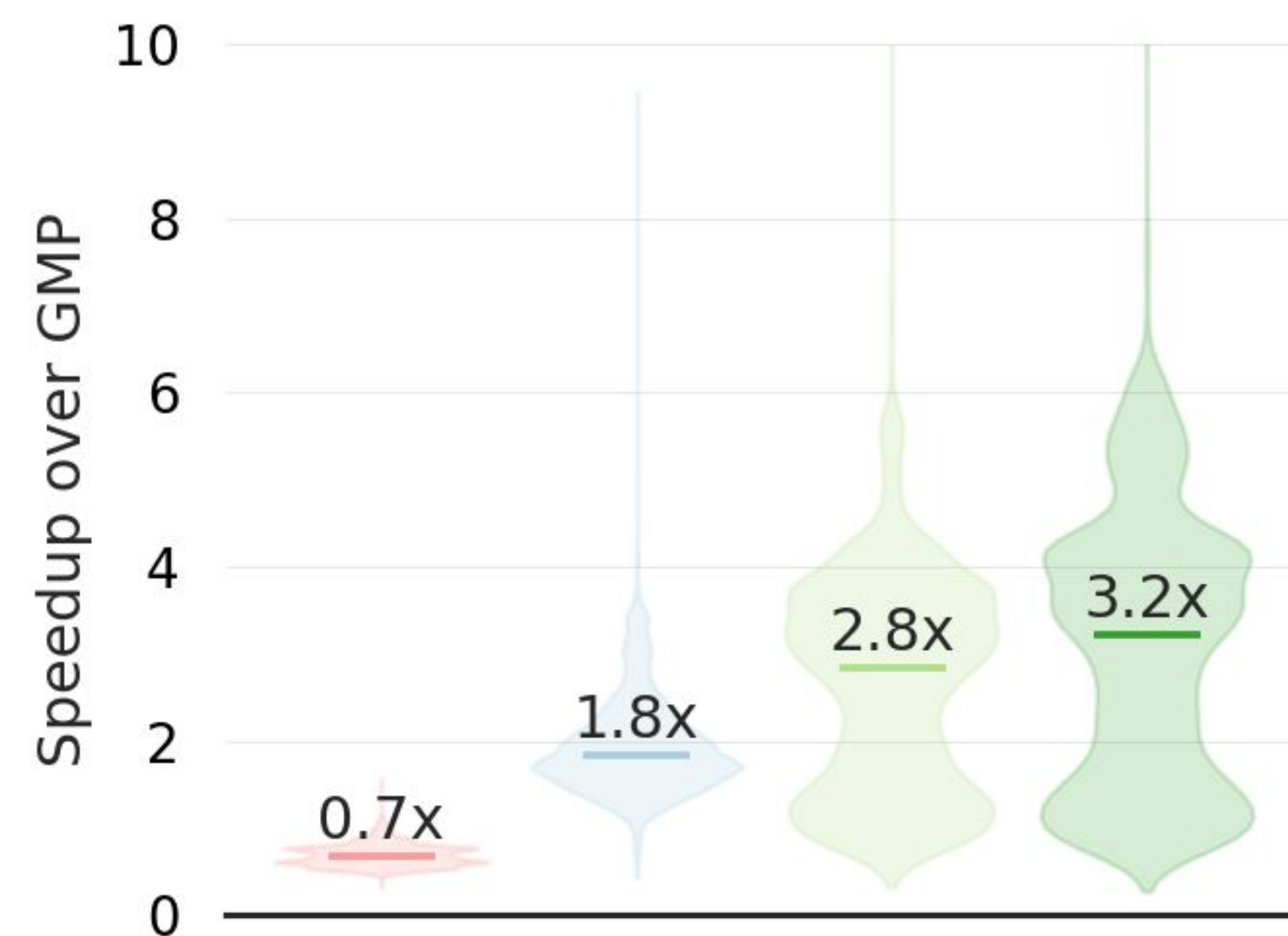
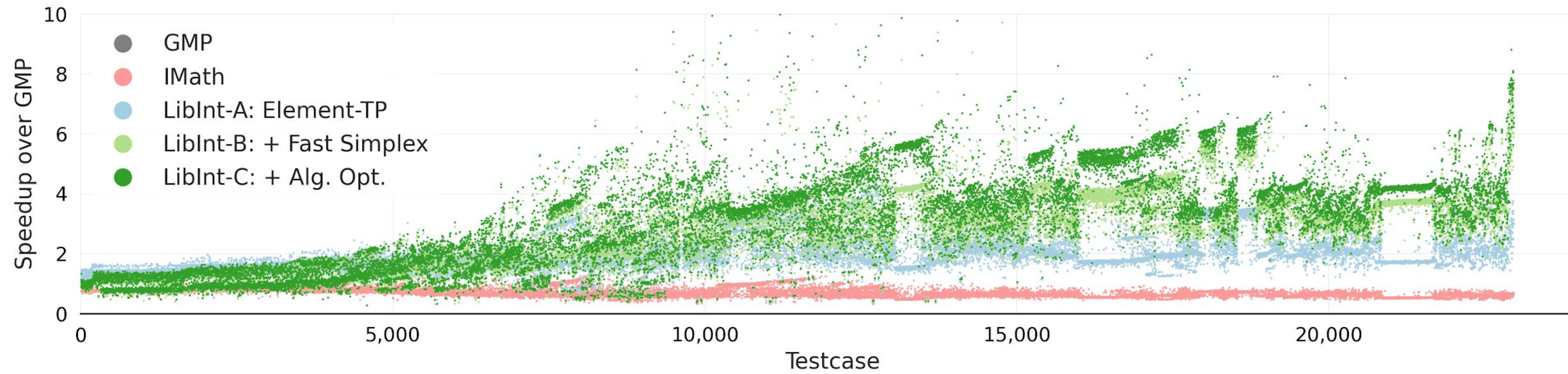


Accelerating the Simplex Pivot Loop

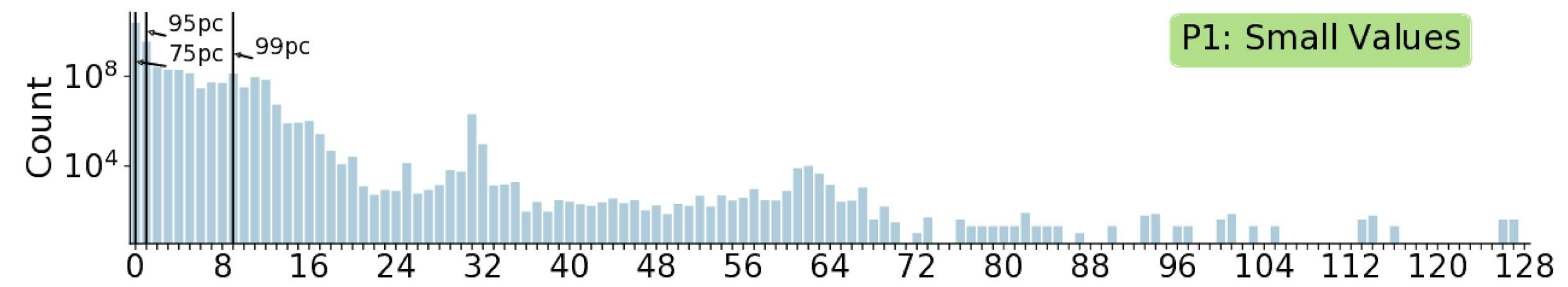
- Arbitrary Precision is Costly
- SIMD Scales for 16 and 32 bit
- Prime-Factor GCD is Fast
- High Speedup in Combination



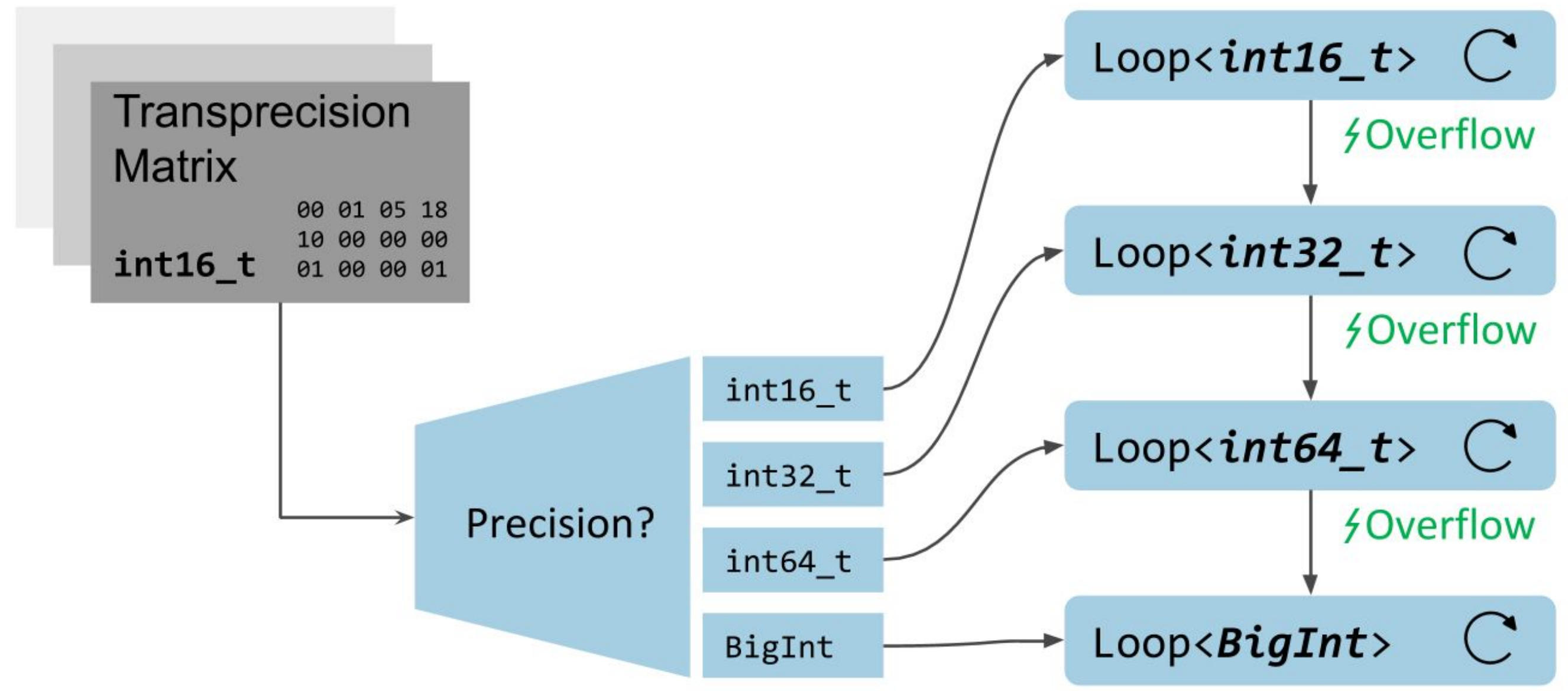
Fast “Coalesce” with Transprecision Simplex



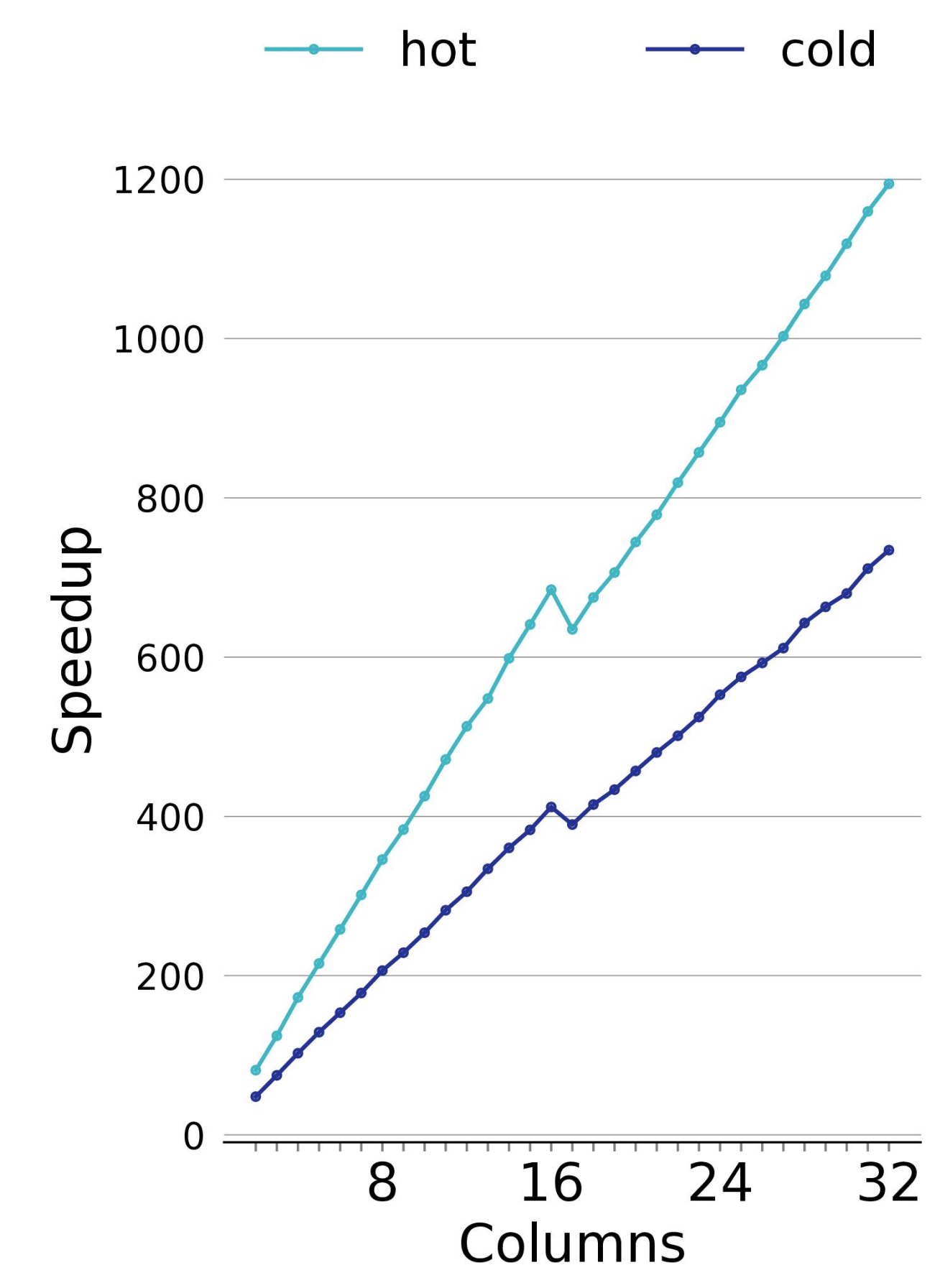
Conclusion



P1: Small Values



Inst. Type	8-bit	16-bit	32-bit	64-bit
Mul	×	✓	✓	✓
Mul High	×	✓	✓	×
Add	✓	✓	✓	✓
Add (Sat.)	✓	✓	×	×



Fast Coalesce

Open Questions:

- Can we accelerate other operations?
- Can data-structure level transprecision arithmetic scale to full polyhedral compilers?

Fast Simplex

