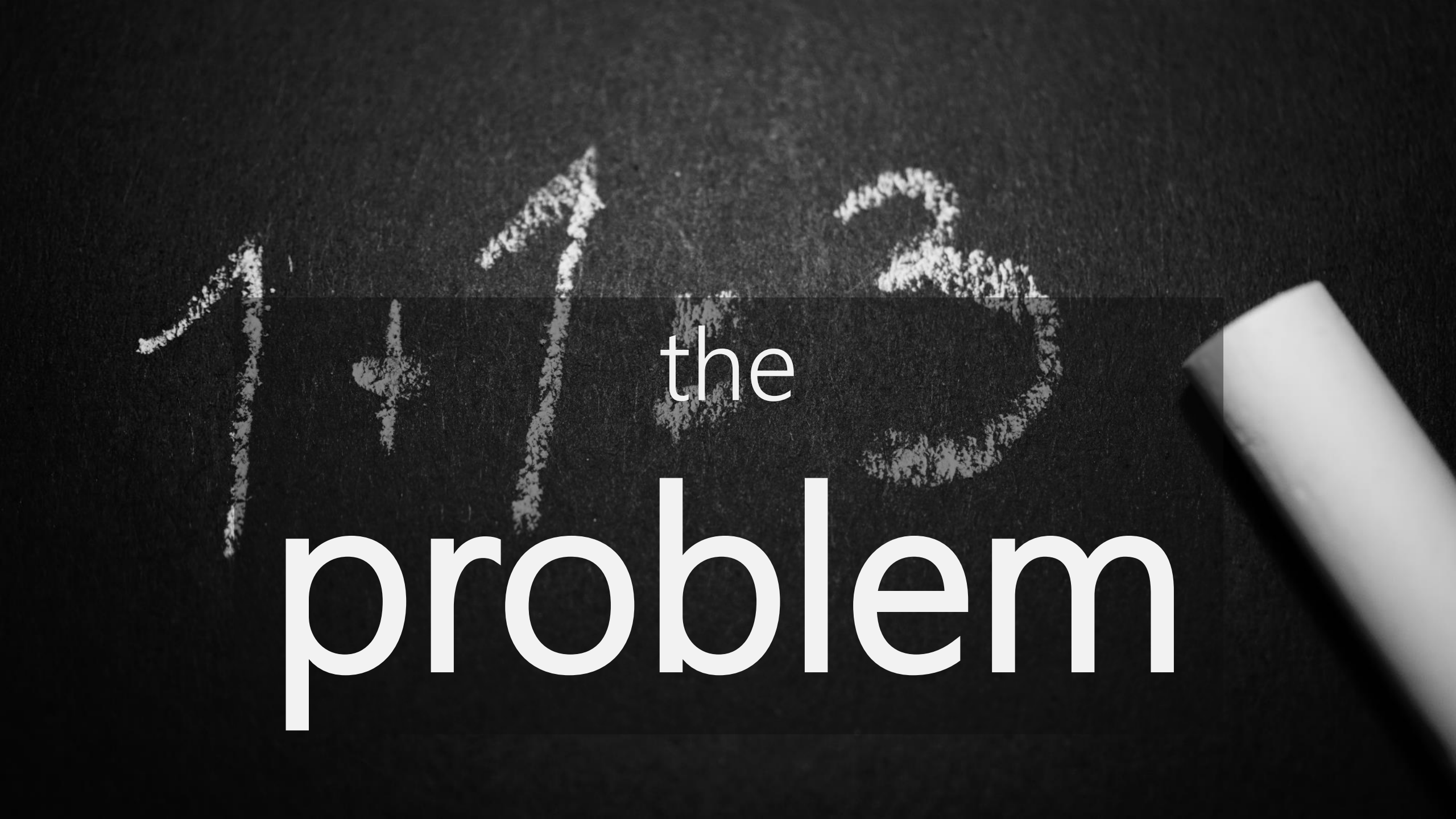
Minimizing Risk While Maximizing Gain

Full Feature Space Representation While Upgrading Minimal Subset of PCs

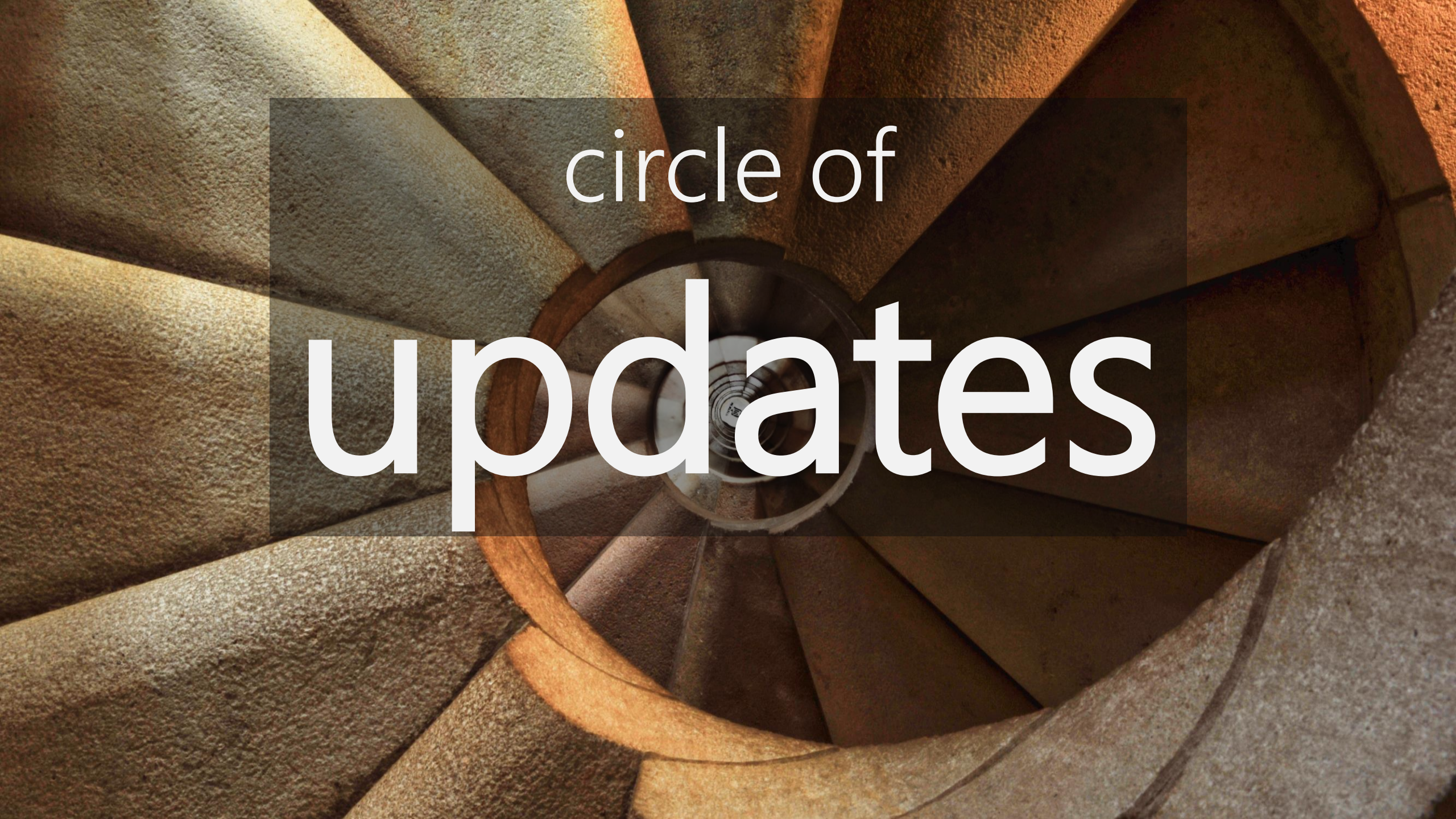
Tom Drabas

Senior Data Scientist

Microsoft


the
problem


highly diverse
ecosystem


circle of
updates


data is
biased

selection

bias

confirmation

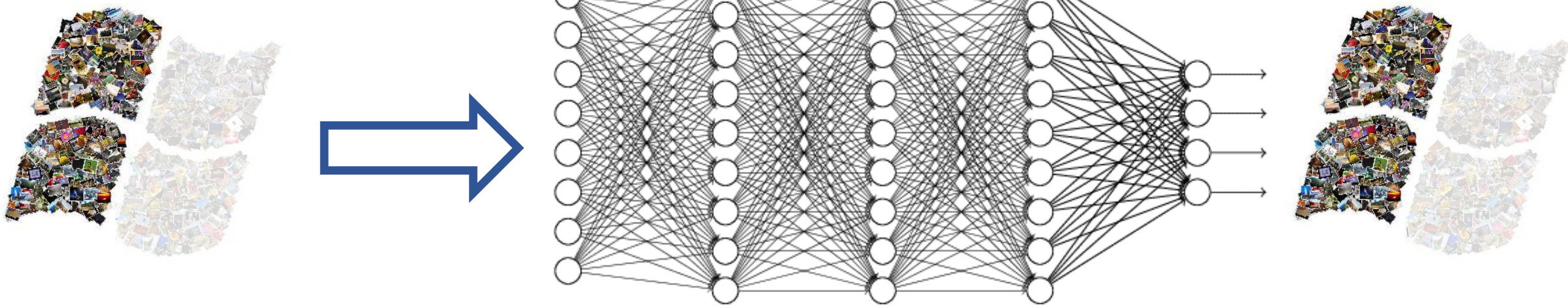
bias

gender

bias

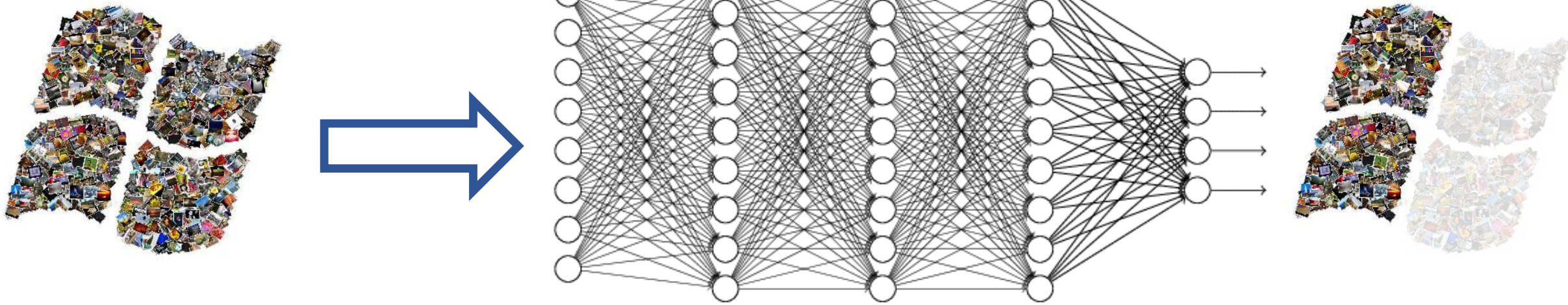
• • •


asking for
trouble



a machine learning model

learns from the data



we don’t know what

“we don’t know”


the
solution


full
view


minimize
risk


be
selective


intel
Kingston
SEAGATE
NVIDIA

intel
Kingston
Western Digital
NVIDIA

intel
Kingston
Western Digital
AMD

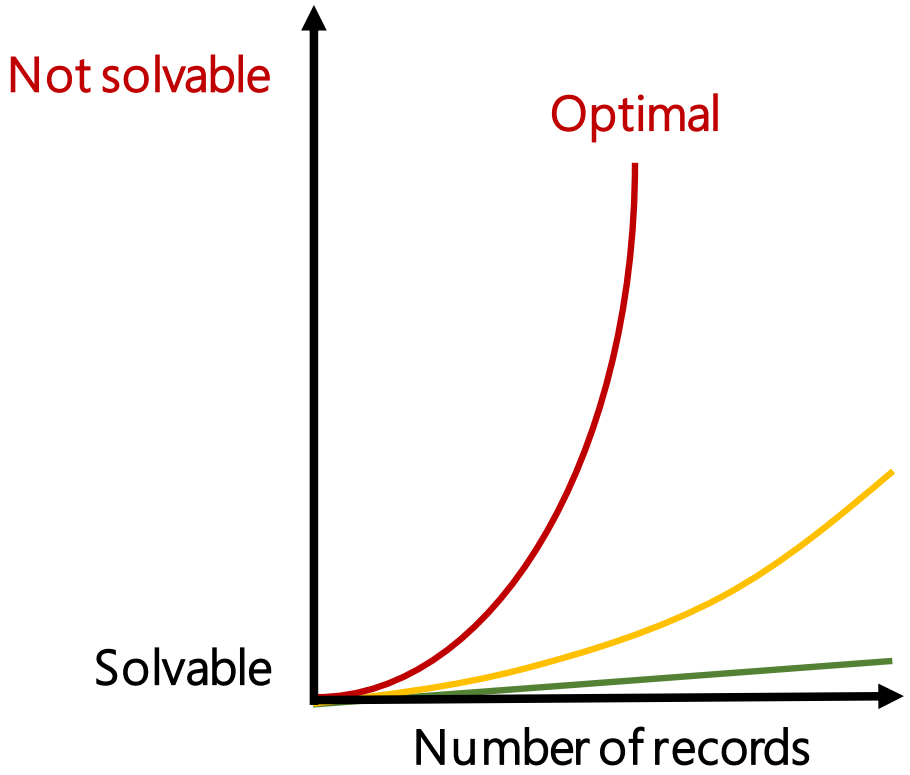

intel
CORSAIR
SanDisk
AMD
Western Digital

intel
CORSAIR
Western Digital
NVIDIA

intel
Kingston
Western Digital
NVIDIA

intel
Kingston
Western Digital
AMD


intel
CORSAIR
SanDisk
SEAGATE
NVIDIA



this problem is

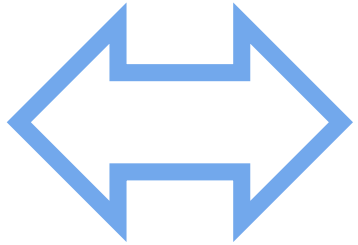
hard

naïve
~O(n^3)

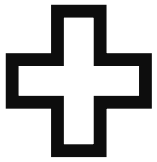
work efficient
~O(n^2)

https://aka.ms/pi_movie

find a minimal subset of transactions
that covers the universe of all values



minimize the cost
of covering the universe of all values



set parallel
~O(*n* log*n*)

RAPIDS

1. Calculate *cost*
2. Sort in ascending order


intel
Kingston
TECHNOLOGY
Western
Digital
NVIDIA


intel
Kingston
TECHNOLOGY
Western
Digital
AMD


intel
SanDisk
Western
Digital
CORSAIR
AMD


intel
Kingston
TECHNOLOGY
Western
Digital
NVIDIA


intel
SanDisk
SEAGATE
CORSAIR
NVIDIA


intel
Kingston
TECHNOLOGY
Western
Digital
AMD


intel
Kingston
TECHNOLOGY
SEAGATE
NVIDIA


intel
CORSAIR
Western
Digital
NVIDIA

RAPIDS

8

5

3

5

3

6

2

2

cost

=

average of log of frequencies of individual components

$$c_i = \frac{1}{n}\sum_j \ln(f_j)$$

RAPIDS

1.77

1.64

1.35

1.77

1.23

1.64

1.50

1.64

RAPIDS

Increasing cost

final

order

RAPIDS

```
import cudf
import pandas as pd
import numpy as np

def calc_log(count_id):
    return np.log(float(count_id))

gdf = cudf.read_csv(
      '../data/exploded.csv'
    , delimiter=‘,’
    , names=['id', 'feature’]
    , skiprows=1
)

freq_items = gdf.groupby('feature').agg('count')
freq_items['ln_freq'] = gdf['count_id'].applymap(calc_log)

gdf = gdf.set_index('feature’)
freq_items = freq_items.set_index('feature’)
gdf = gdf.join(freq_items, how='left’)

gdf = gdf.groupby('id').agg(['mean'])

gdf = gdf.sort_values(by='mean_ln_freq')
```

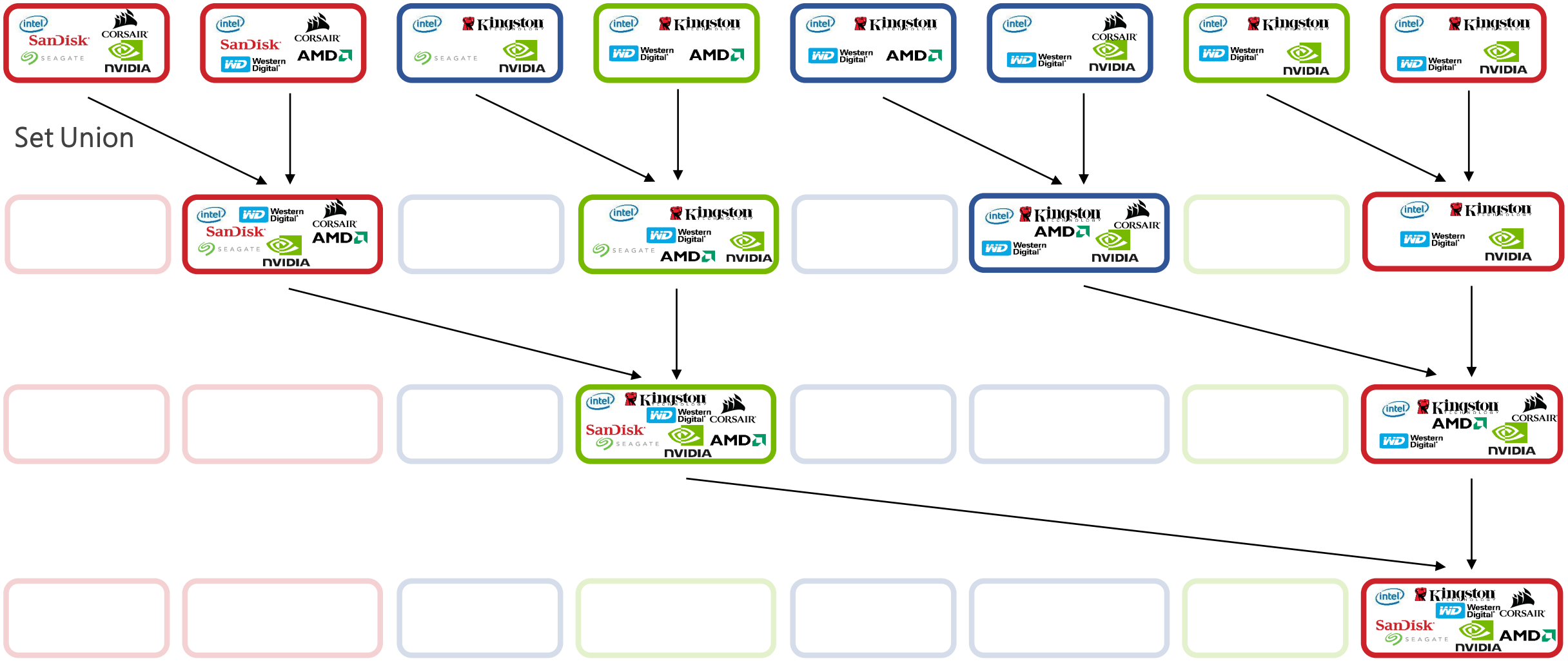
RAPIDS

data framework

RAPIDS

3. Run *Set Prefix Scan* on GPU

Based on https://aka.ms/mharris_pps



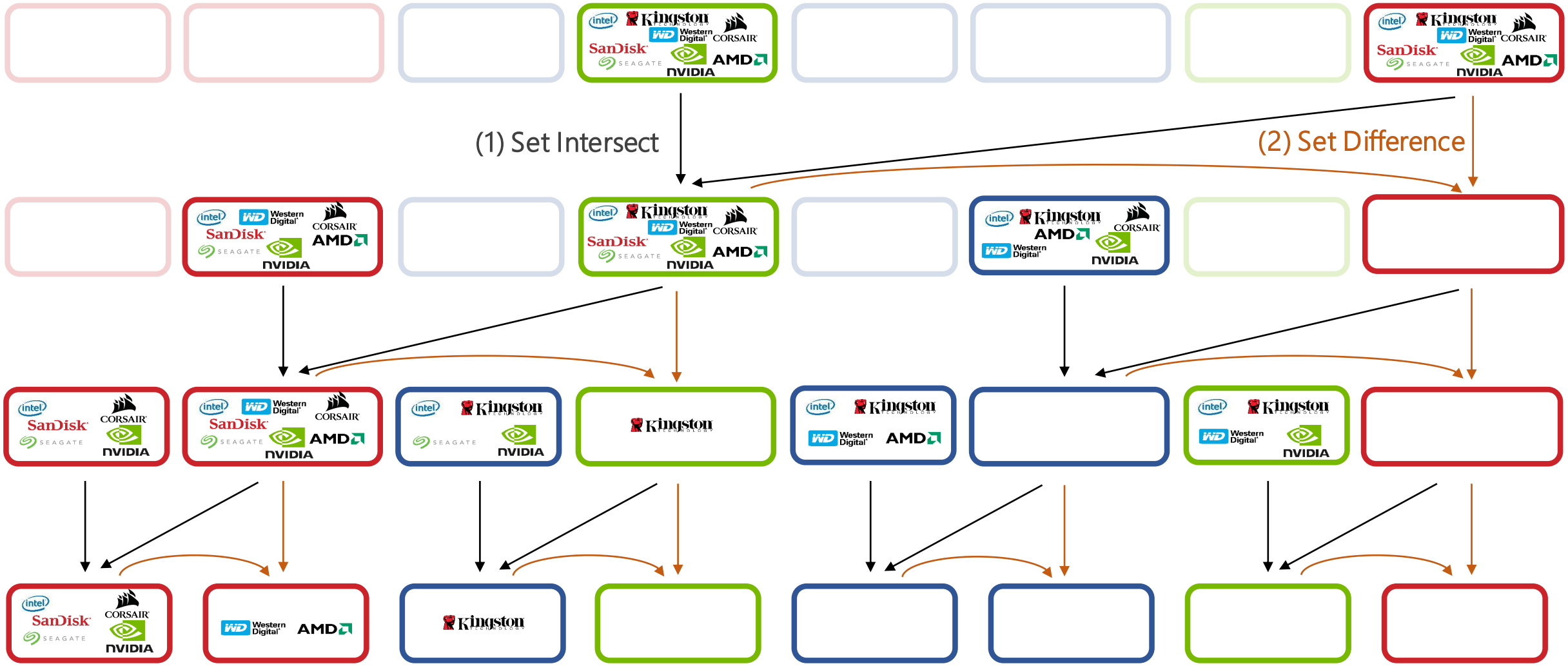
Prefix Set Scan

up the tree

```
__global__
void gpu_prefix_set_scan_full_kernel(
          const uint32_t* input
        ,       uint32_t* output
        ,       uint32_t  curr_val_size
        ,       uint32_t  rec_cnt
    )
{
    extern __shared__ uint32_t temp[];

    int thid = blockIdx.x * blockDim.x + threadIdx.x;
    int offset = 1;

    // STORE IN TEMP
    ...

    // SCAN UP THE TREE
    int n = rec_cnt;

    for(int d = n >> 1; d > 0; d >>= 1)
    {
        __syncthreads();

        if(thid < d)
        {
            int ai = offset * (2 * thid + 1) - 1;
            int bi = offset * (2 * thid + 2) - 1;

            set_union_device(ai, bi, temp, curr_val_size, rec_cnt);
        }

        offset *= 2;
    }
```

Prefix Set Scan

up the tree



Prefix Set Scan

down the tree

DESIGNED FOR NVIDIA CUDA

```
    ...

    for(int d = 1; d < n; d <<= 1)
    {
        offset >>= 1;
        __syncthreads();

        if(thid < d)
        {
            int ai = offset * (2 * thid + 1) - 1;
            int bi = offset * (2 * thid + 2) - 1;

            set_intersect_device(bi, ai, temp, curr_val_size, rec_cnt);
            set_difference_device(ai, bi, temp, curr_val_size, rec_cnt);
        }
    }
}
```

Prefix Set Scan

down the tree


the
benefits

	naïve	work efficient	set parallel
			RAPIDS DESIGNED FOR NVIDIA CUDA
time (minutes)	54.1	18.1	0.43 (~26s)
speedup (naïve)		2.98x	125.8x
speedup (work efficient)			42.1x

1M records

100k feature values

NVIDIA RTX 2080Ti, i5 2.4GHz, 64GB RAM, NVMe


keeping
track


account for
everything


Summary


QUESTIONS