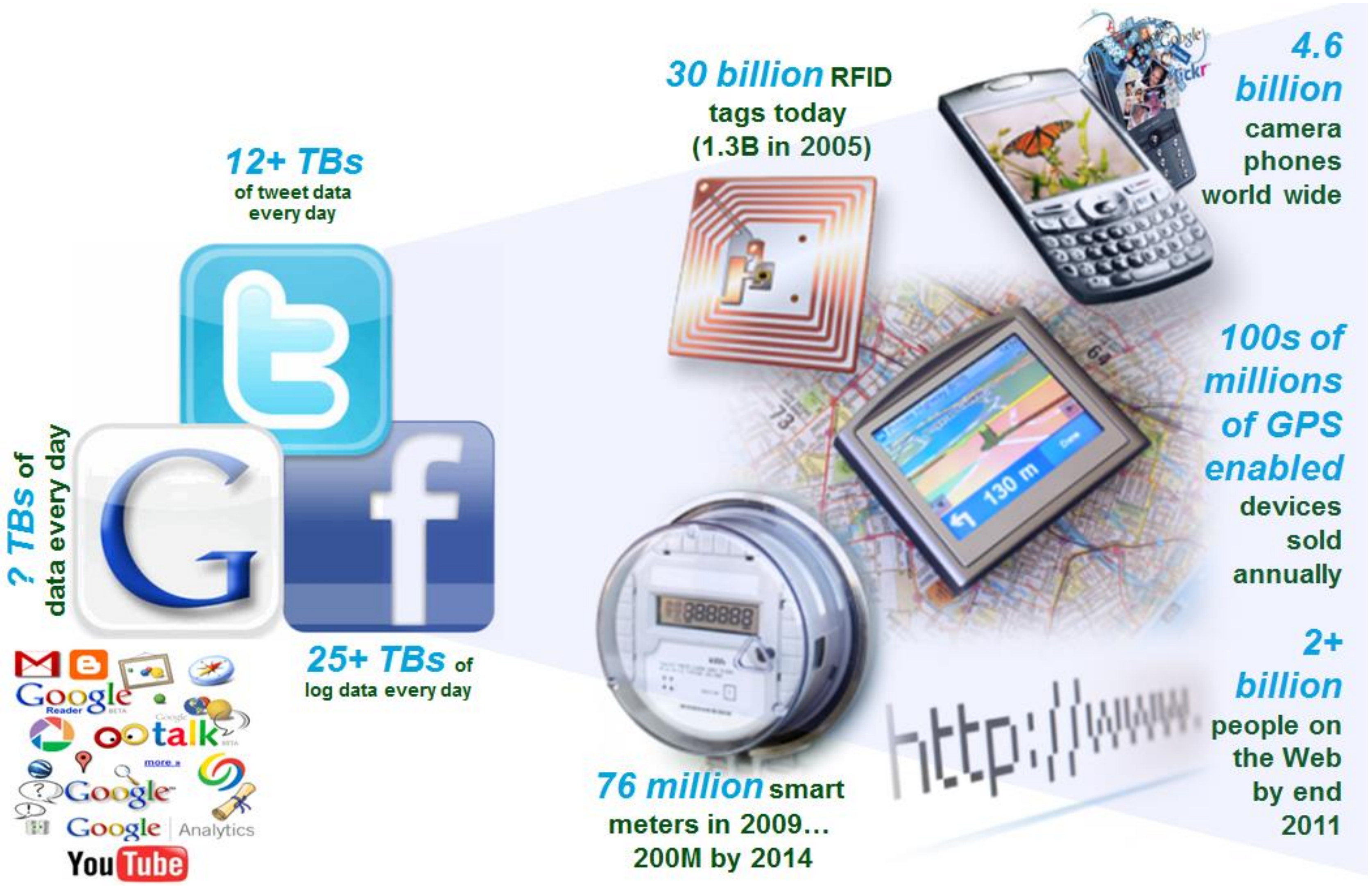
Progressive Clustering of Big Data with GPU Acceleration and Visualization

Jun Wang[1], Eric Papenhausen[1], Bing Wang[1], Sungsoo Ha[1], Alla Zelenyuk[2], and Klaus Mueller[1]

[1]Computer Science Department, Stony Brook University
[2]Pacific Northwest National Laboratory

The Internet of Things and People



Big Data for Scientific Research

Nature - Big Data

Sept. 3, 2008, Vol. 455, Issue 7209

Science - Dealing with Data

Feb. 11, 2011, Vol. 331, Issue 6018

Our Data – Aerosol Science

Understand the processes that control formation, physicochemical properties and transformations of particles

Acquired by a state-of-the-art single particle mass spectrometer (SPLAT II) often deployed in an aircraft

Our Data – Aerosol Science

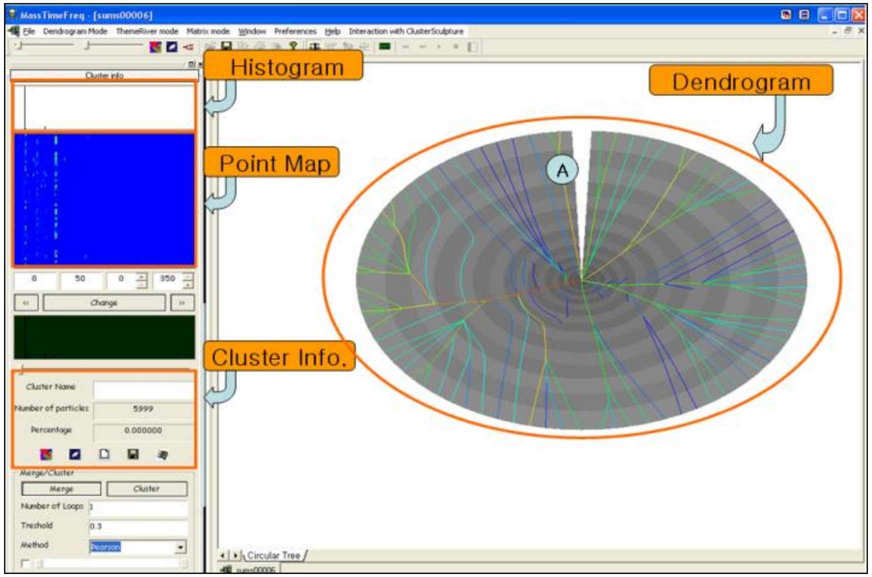
SPLAT II can acquire up to 100 particles per second at sizes between 50-3,000 nm at a precision of 1 nm

- Creates a 450-D mass spectrum for each particle

Overall Goal:

Build the hierarchical structure of particles that can be used in automated classification of new particle acquisitions

SpectraMiner



Our Data – Aerosol Science

Data Scale:

- 450 dimensions
- Typically, several millions of points

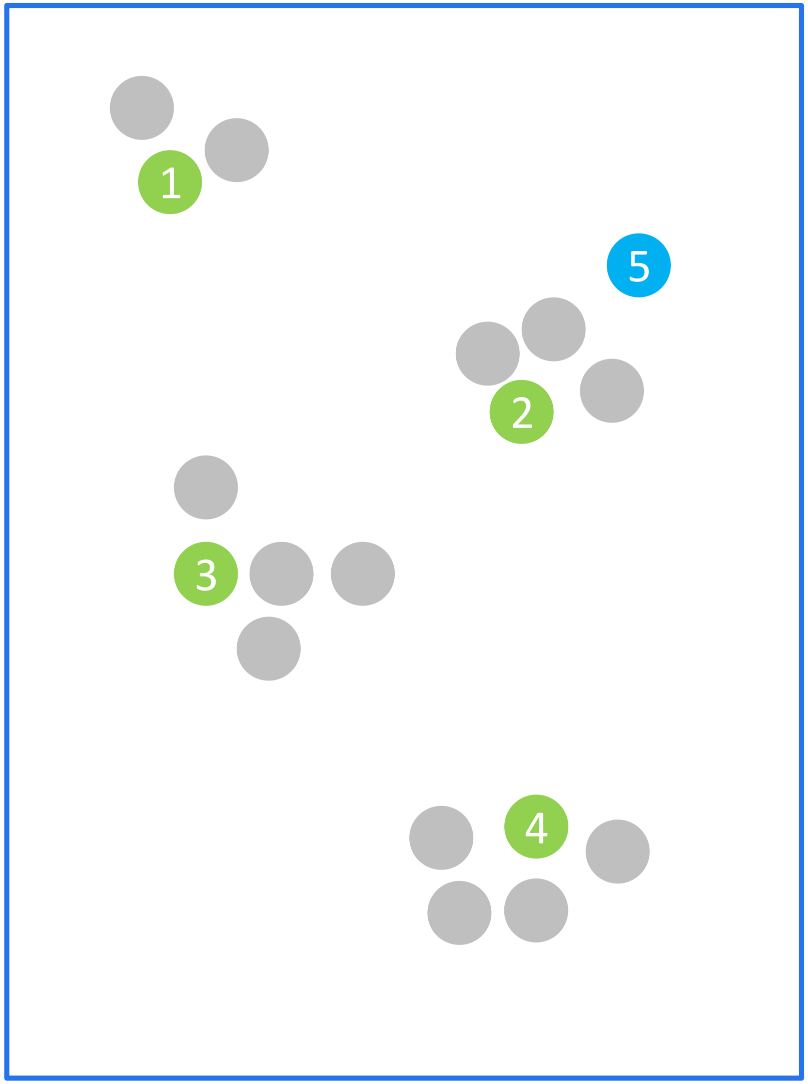
Goal:

- Overall: hierarchical (tree) structure
- In this talk: parallel clustering algorithms for
  - Redundancy elimination
  - Learning the leaf level of the tree

Incremental k-Means – Sequential

The old CPU-based solution

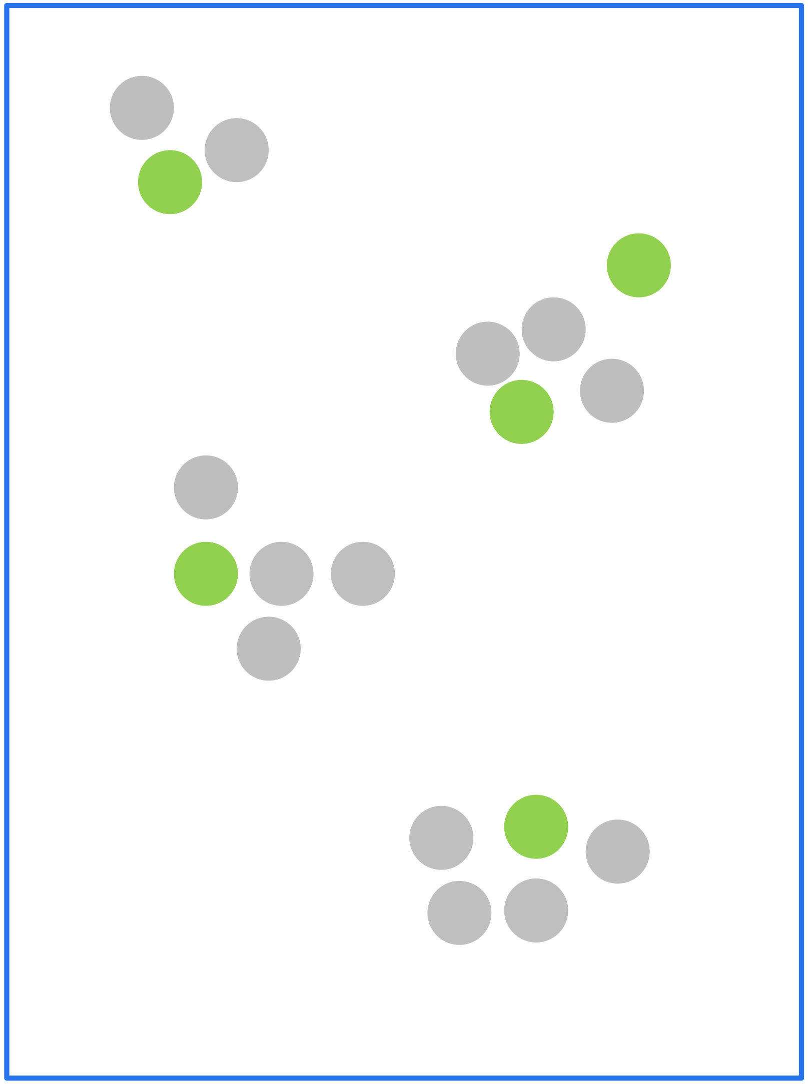
Input: data points ***P***, distance threshold *t*
Output: clusters ***C***
C *= empty set*
for each *unclustered point p in* ***P***
  if *C is empty* **then**
    Make p a new cluster center and add it into ***C***
  else
    p = next unclustered point
    Find the cluster center c in **C** *closest to p*
    let *d = distance(c, p)*
    if *d < t* **then** *Cluster p into c*
    else *Make p a new cluster center added to* ***C***
    end if
  end if
end for
return ***C***



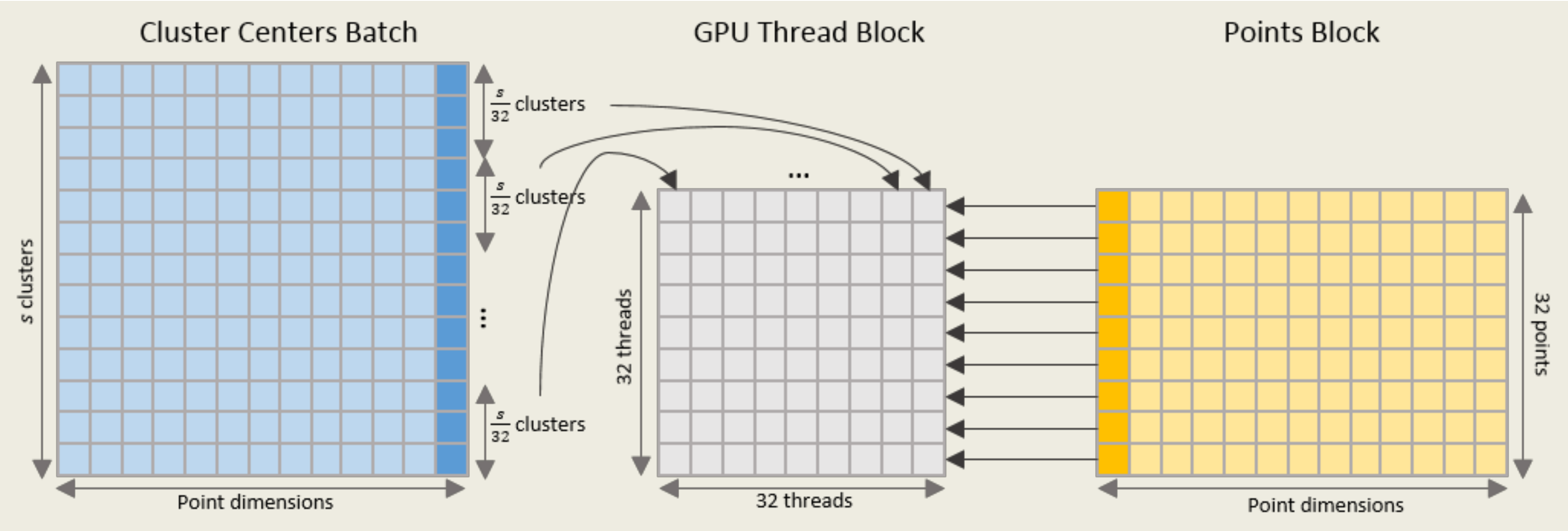
Incremental k-Means – Parallel

NEW GPU-based solution

Input: data points $\boldsymbol{P}$, distance threshold t, batch size b, max iteration M
Output: clusters $\boldsymbol{C}$
$\boldsymbol{C}$ = empty set
while *number of un-clustered points in $\boldsymbol{P}$ > 0*
    Run Alg. 1 until a number of b clusters $\boldsymbol{B}$ emerge
    Iteration $i = 0$
    while $i < M$ and $\boldsymbol{B}$ *is not stable*
        in parallel:
            for each *unclustered point* p_i
                Find the center b_i *in* $\boldsymbol{B}$ *closest to* p_i
                if $distance(b_i, p_i) < t$ **then** $c_i = b_i$
                else $c_i = null$
            end for
        on CPU: *Assign* p_i *to* b_i *if* c_i *is not null*
        in parallel: *update centers of* $\boldsymbol{B}$
    end while
    Add $\boldsymbol{B}$ *to* $\boldsymbol{C}$
end while
return $\boldsymbol{C}$



Incremental k-Means – Parallel



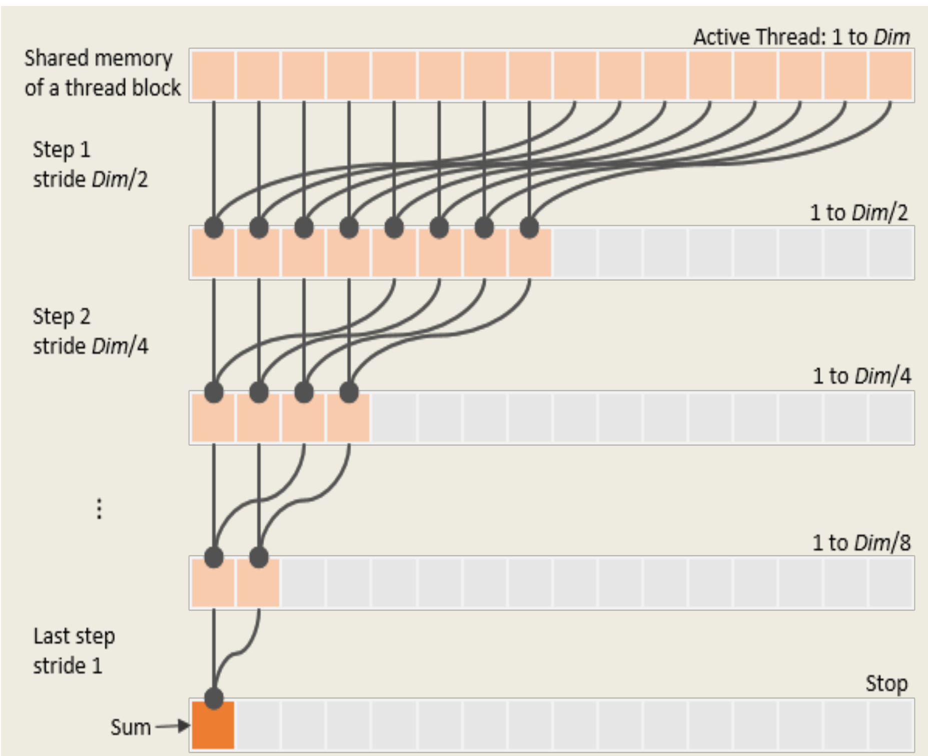
GPU Thread Block: 32 × 32 threads

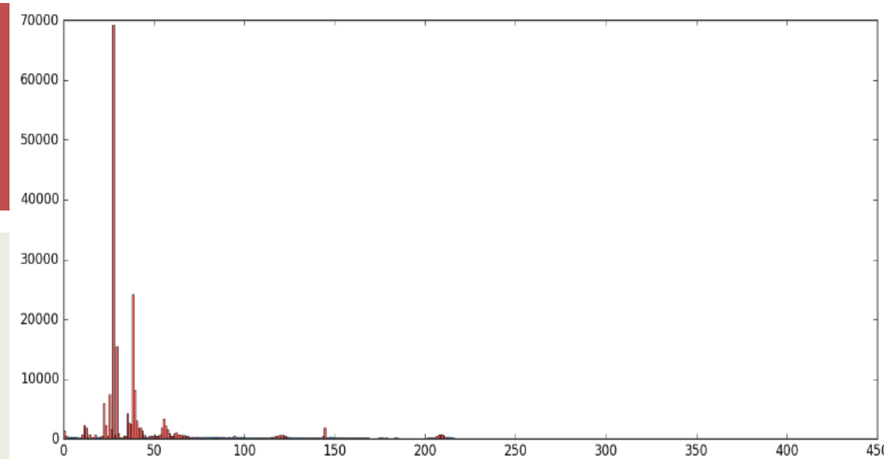
If we set the batch size *b* = 96:

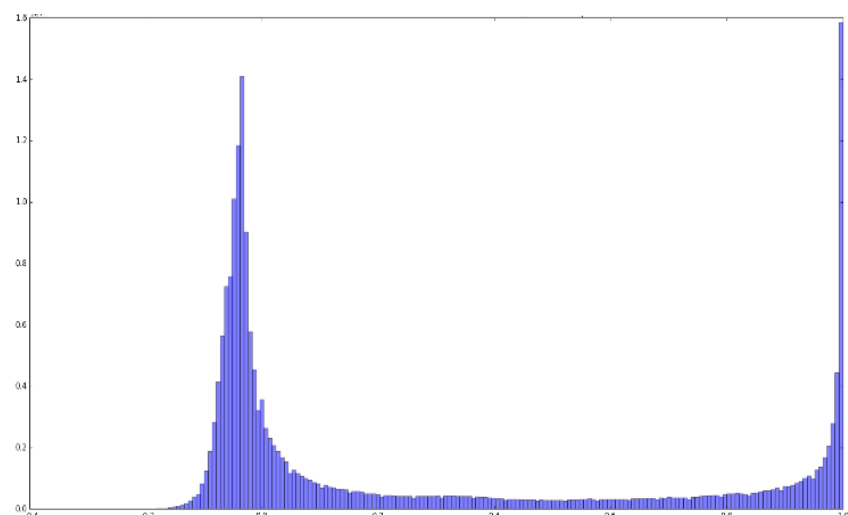
- A group of 96/32 = 3 cluster centers is mapped to a column of the block
- Number of thread block launched: ***N***/32

Dimension Reduction and the Threshold t

Dimension standard deviations by ***parallel reduction***







Comments and Observations

Algorithm merges the incremental k-means algorithm with a parallel implementation (k=C)

Design choices:

- C=96 good balance between CPU and GPU utilization
- With C>96 algorithm becomes CPU-bound
- With C<96 the GPU would be underutilized
- A multiple of 32 avoids divergent warps on the GPU
- Max iterations = 5 worked best

Advantages of the new scheme:

- Second pass of previous scheme no longer needed

GPU Implementation

Platform

- 1-4 Tesla K20 GPUs
- Installed in a remote 'cloud' server

Parallelism

- Launch N/32 thread blocks of size 32 x 32 each
- Each thread compares a point with 3 cluster centers
- Make use of shared memory to avoid non-coalesced memory accesses

Quality Measures

Cluster quality measure: *Davies-Bouldin (DB) index*

$$DB = \frac{1}{n}\sum_{i=1}^{n} \max\left(\frac{\sigma_i + \sigma_j}{M_{ij}}\right)$$

- σ_i - intra-cluster distance
- M_{ij} - inter-cluster distance
- The lower the DB, the better the quality

Acceleration by Sub-Thresholding

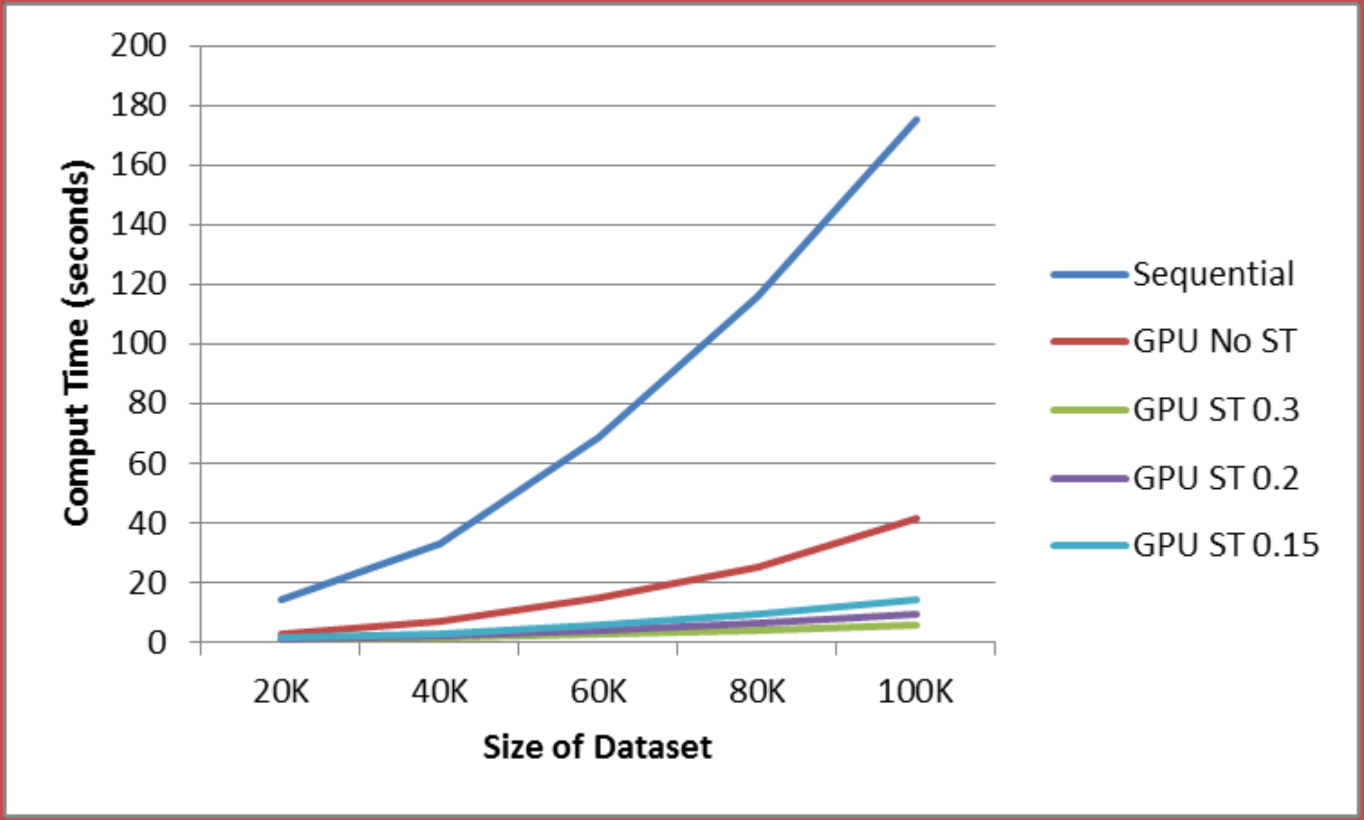
- Sub-thresholding:

  Points with ***small distance*** to the center are settled and will not be re-visited in inner iterations

- This also somehow improves the DB index

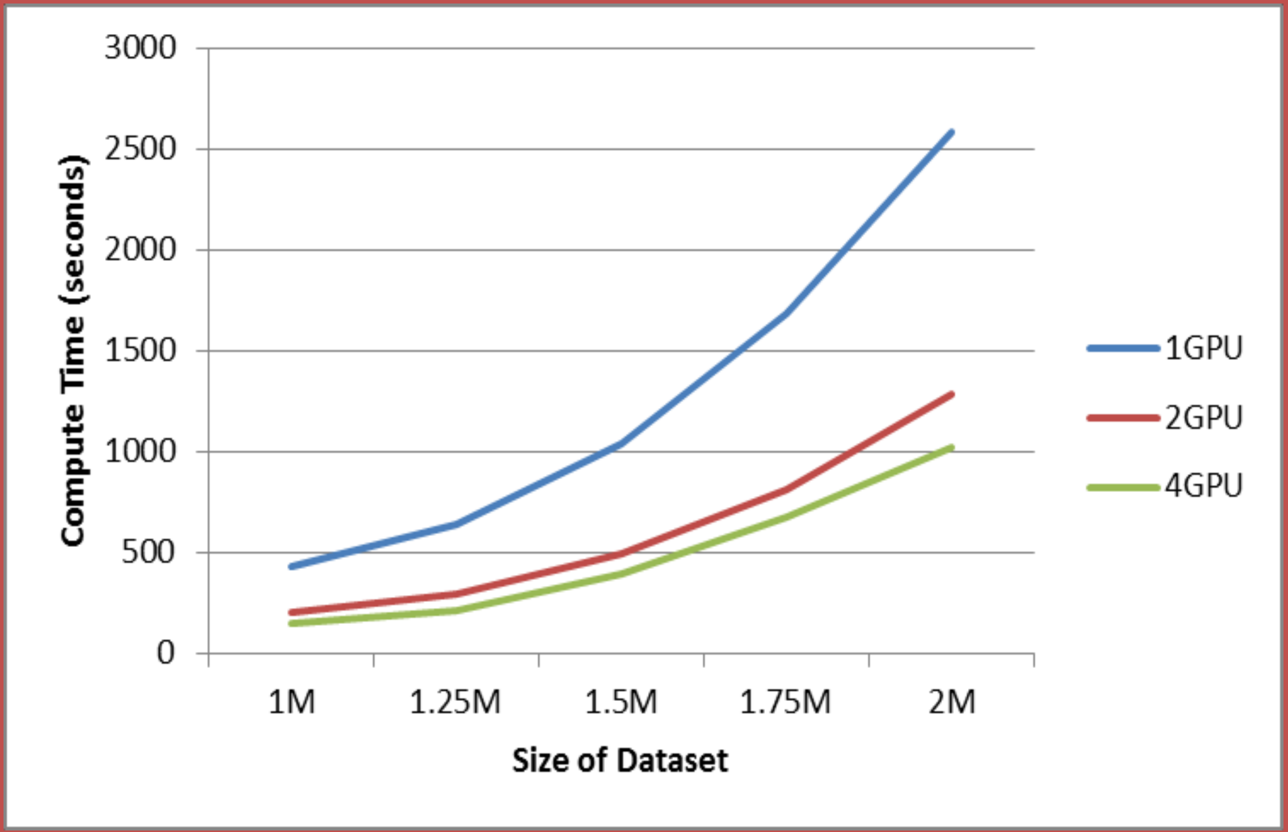
Data Size	Sequential	Parallel	ST: 0.3	ST: 0.2	ST: 0.15
10k	**0.527**	0.539	0.540	0.537	0.529
50k	0.546	0.590	0.548	0.554	**0.539**
100k	0.550	0.584	0.600	0.570	**0.544**
200k	0.564	0.587	0.640	0.593	**0.564**

Results – Different Settings



About 33x speedup

Results – Multi-GPU



4-GPU has about 100x speedup over sequential

In-Situ Visual Feedback (1)

Visualize cluster centers as summary snapshots

- Glimmer MDS algorithm
- Intuitive 2D layout

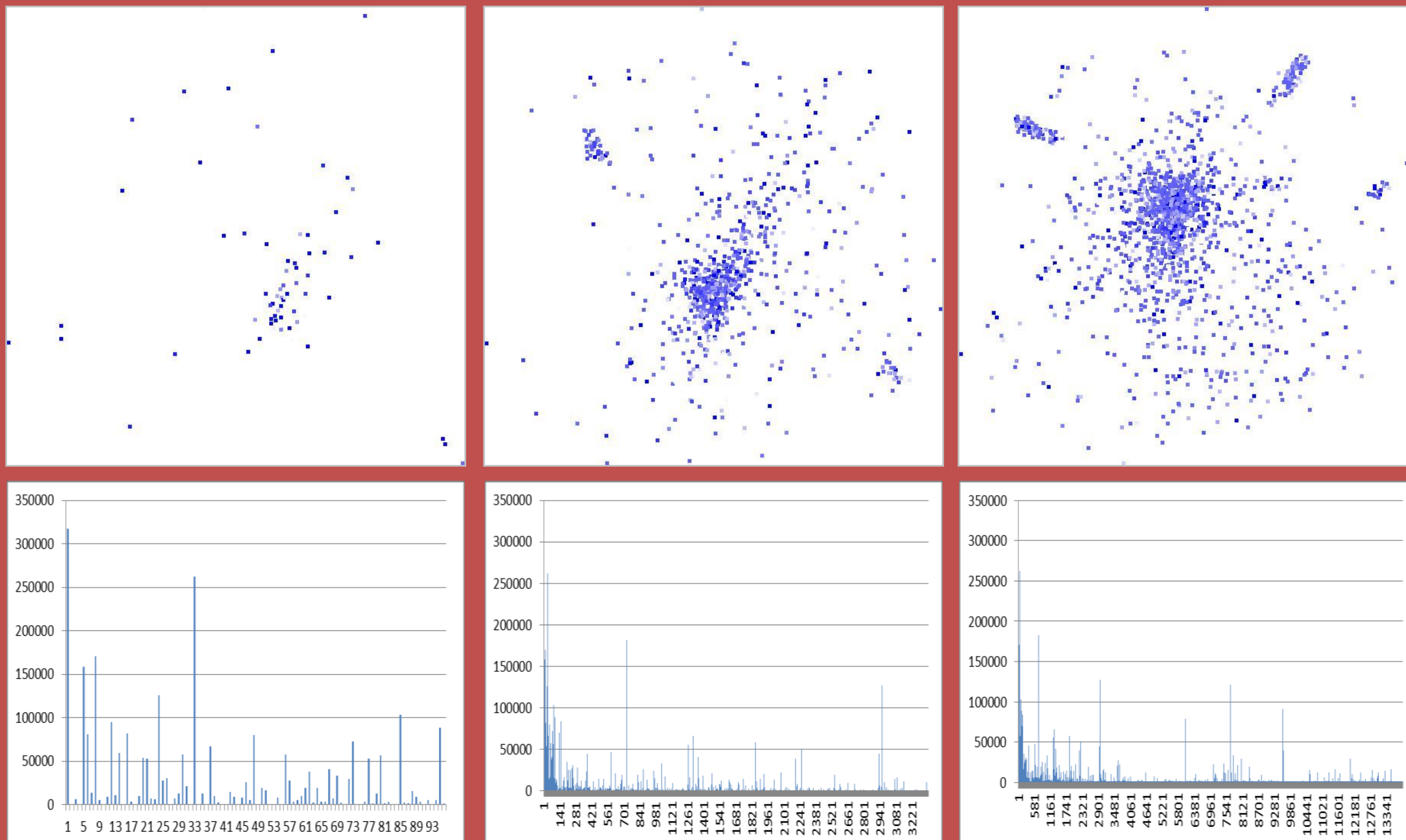
Color map:

- Small clusters map to mostly white
- Large clusters map to saturated blue

We find that early visualizations are already quite revealing

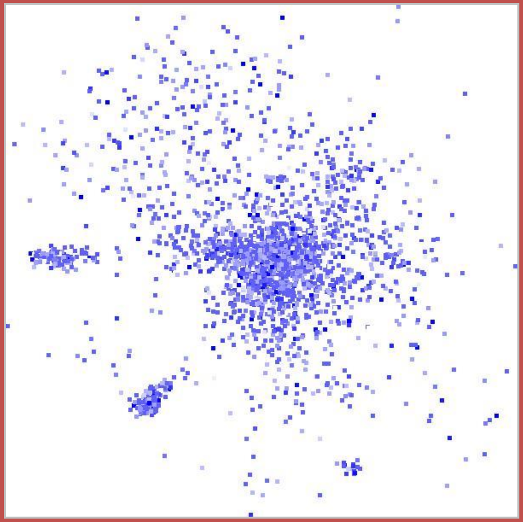
- This is shown by cluster size histogram
- Cluster size of M>10 is considered significant

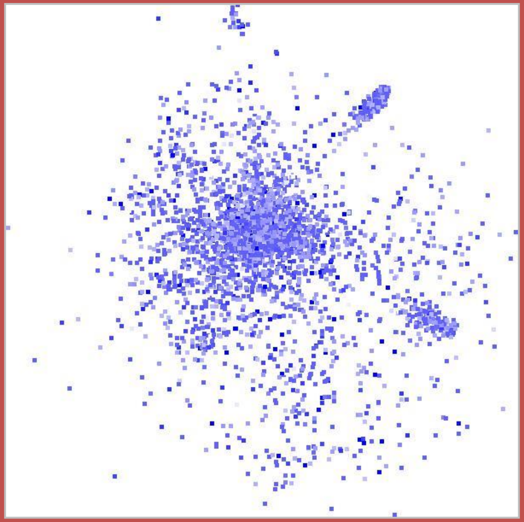
In-Situ Visual Feedback (2)

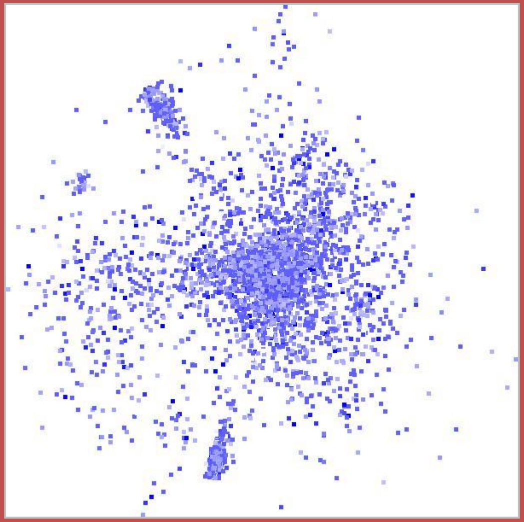


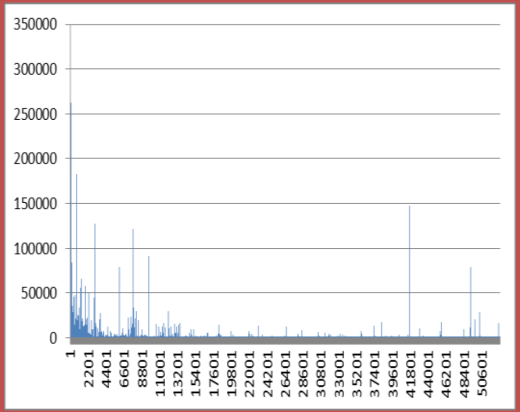
79/96 998/3360 2004/13920

In-Situ Visual Feedback (3)

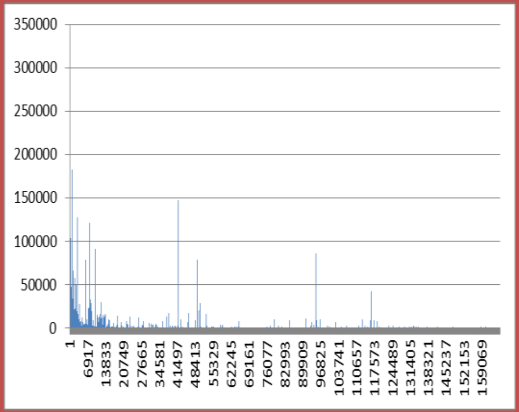




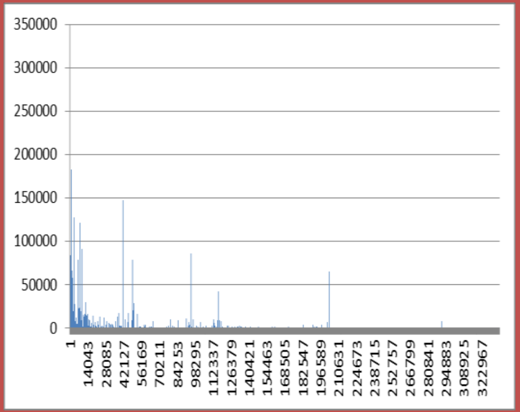




3001/52800



4002/165984



4207/336994

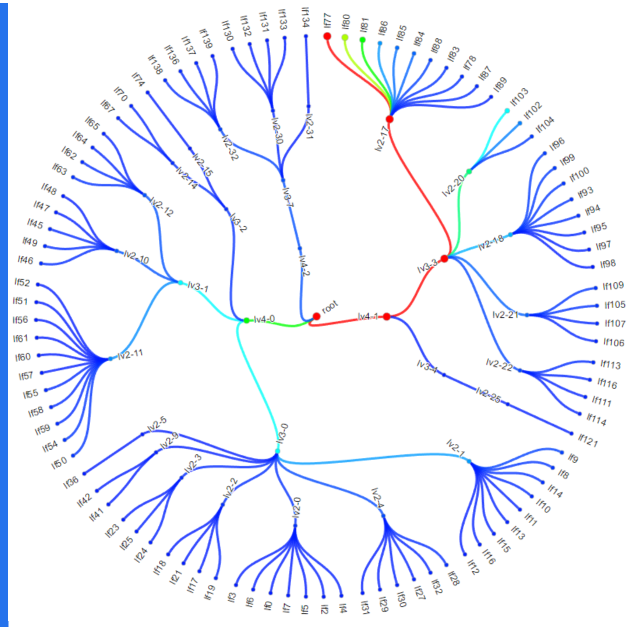
Conclusions and Future Work

Current approach quite promising

- Good speedup and results
- In-situ visualization of data reduction process with early valuable feedback

Future work

- More efficient data storage facilities
- Load-balancing point for multi-GPU
- Accelerate the hierarchy building
- A comprehensive VA system



Final Slide

Thanks for attending!

And thanks to NSF and DOE for funding

Any questions?