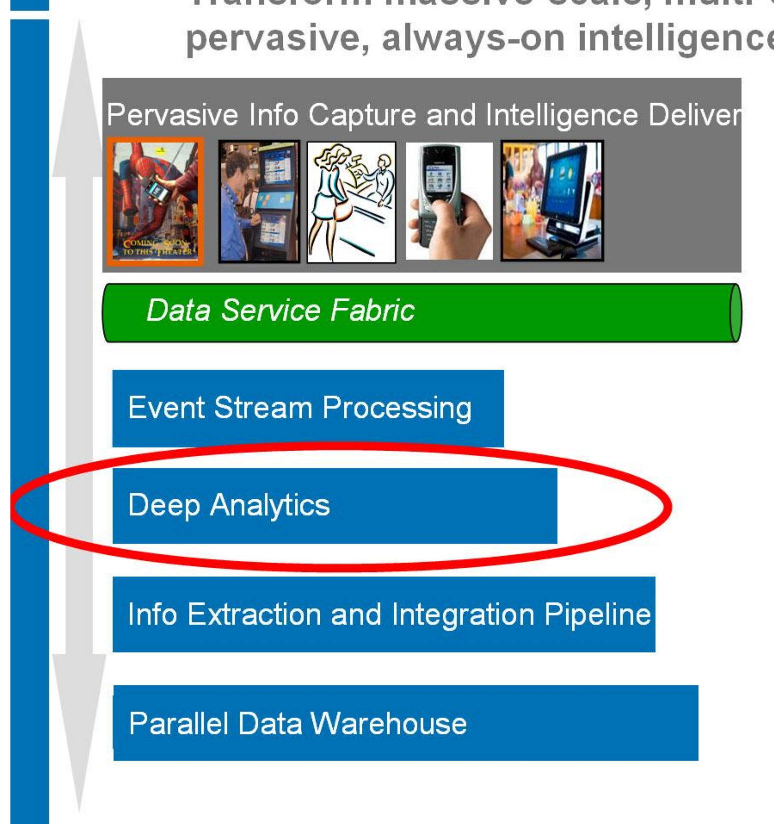
Very Large Scale Clustering on GPUs via Streaming

Ren Wu, Bin Zhang, Meichun Hsu
Intelligent Information Management Lab, HP Labs

LABS hp

Next-Gen Operational BI

Transform massive-scale, multi-channel, multi-modal data into pervasive, always-on intelligence



- Innovate in new paradigms for engaging customers, suppliers, partners
- With insights gained from aggregate analyses of many sources of information - OLTP, devices, structured, unstructured, historical, real-time…
- Delivered when and where it's needed,
- At scale and at an affordable cost.

An Example: K-means clustering

Find the positions of • that minimize

$$Perf_{KM}(X,M) = \sum_{i=1}^{N} MIN\{\| x_i - m_l \|^2 \mid l = 1,\ldots,K\}.$$

K-means algorithm: Starting from an initial position of the centers,

Partition the data set so that each data point goes with the closest center

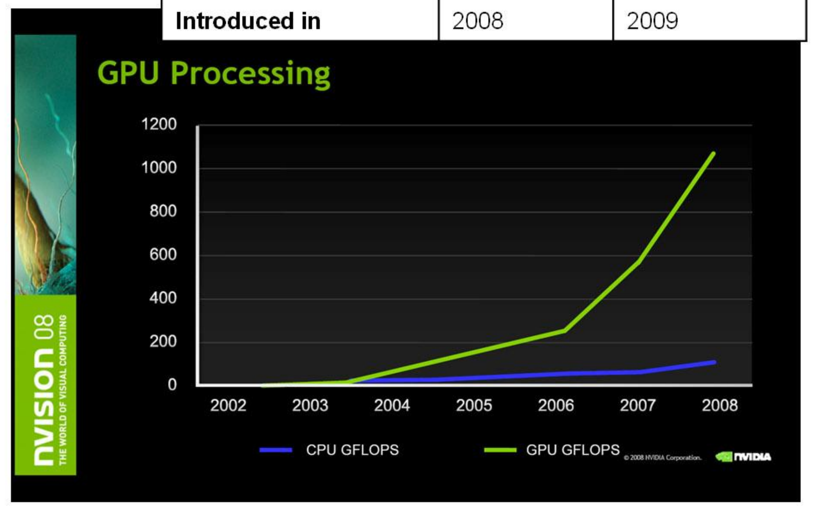
Recalculate the centers as the geometric center of each partition

Iterate the two steps above until no change happens.

Massively Parallel Accelerators

- General Purpose GPU
  - Multiple SIMD multi-processors
  - Much wider memory access bandwidth
  - Good for massively data-parallel compute-intensive tasks

Nvidia GPGPUs	GTX 280	GTX 295
# of Cores (SPs)	240	480
GFLOPS	900	1,788
Mem Bandwidth	142 GB/s	224 GB/s
Introduced in	2008	2009



Push for Performance in Clustering

- Memory Hierarchy
  - Data kept in both CPU and GPU memory
  - Re-arrange data to column based in GPU
  - Optimize memory allocation
    - E.g. Centroids kept in GPU's constant memory: Utilizing built-in cache of constant memory since no update during the assignCluster pass.
- Hybrid approach
  - Only centroid and membership array need to be communicated at each iteration
  - For the aggregation phase, multicore CPU can do as well as GPU, even with memory transfers.
    - Share nothing: Parallel aggregation with local aggregates

Handling Larger Data Set in Clustering

- Data set does not fit in GPU memory
  - Data needs to be streamed from CPU to GPU for each iteration
- Concurrent execution
  - Memory transfers (host <-> GPU board)
  - GPU kernels (data block transpose, Center assignments etc)
  - CPU kernels (aggregation per data block)
- Significant speedup can still be achieved with careful algorithm design and implementation

Large datasets
- Streaming algorithm (per iteration)

```
Memcpy(dgc, hgc);
while (1)
{
    while (ns = streamAvail() && !done)
    {
        hnb = nextBlock();
        MemcpyAsync(db, hnb, ns);
        DTranspose(db, dtb, ns);
        DAssignCluster(dtb, dc, ns);
        MemcpyAsync(hc, dc, ns);
    }
    while (ns = streamDone())
        aggregateCentroid(hc, hb, ns);

    if (done)
        break;
    else
        yield();
}
calcCentroid(hgc);
```

Results
- 1 billion data points case

dataset				time (s)		speedups
N	D	K	M	CPU (8c)	GPU	
1,000,000,000	2	200	50	4139	508	8.2
1,000,000,000	2	400	50	7470	744	10.0
1,000,000,000	2	600	50	10847	1012	10.7
1,000,000,000	2	800	50	14176	1284	11.0
1,000,000,000	2	1000	50	17515	1558	11.2
						10.2

- 10x faster than optimized CPU version on 8 cores
  - More than 300x faster than MineBench on single core
- Even with data transfer overhead!

Performance
Large number of clusters

dataset				time (s)		speedups
N	D	K	M	CPU (8c)	GPU	
100,000,000	2	2000	50	3415	299	11.4
100,000,000	4	2000	50	5969	446	13.4
100,000,000	6	2000	50	8343	2528	3.3
100,000,000	8	2000	50	10711	3354	3.2
						7.8

- If the centroids array is too big, program automatically switch to texture memory
- Performance took bigger hit
- But still offers more than 3x performance advantage vs. optimized CPU version on 8 cores.

R. Wu, B. Zhang and M. Hsu. *GPU-Accelerated Large Scale Analytics.* HP Labs Technical Report, HPL-2009-38. http://www.hpl.hp.com/techreports/2009/HPL-2009-38.html
R. Wu, B. Zhang and M. Hsu. *Clustering Billions of Data Points Using GPUs.* ACM UCHPC09: Second Workshop on UnConventional High Performance Computing. Italy, 2009.

(ren.wu, bin.zhang2, meichun.hsu)@hp.com