World Conference on Information Technology

Parallel Wavelet-based Clustering Algorithm on GPUs using CUDA

Ahmet Artu YILDIRIM
Cem ÖZDOĞAN

October 2010

Outline

- Cluster Analysis
- WaveCluster Approach
- Parallel Computing on GPUs and NVIDIA CUDA™ Technology
- Implementation of Low-Frequency Component Extraction
- Implementation of Connected Component Labeling
- Results
- Conclusion

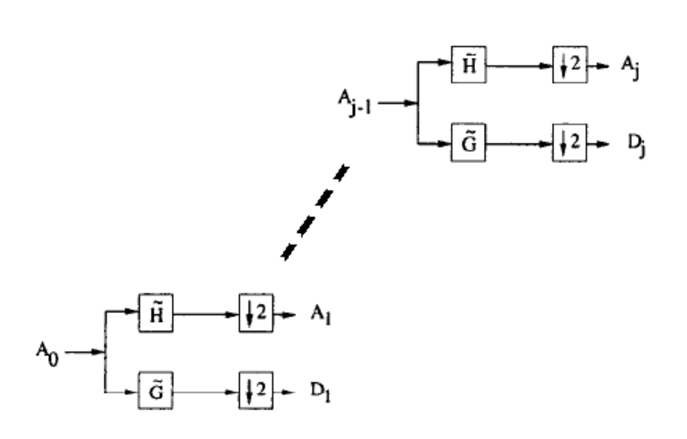
Cluster Analysis

- Common technique for grouping a set of objects into classes of “similar” objects or clusters
- Highly used in many fields such as:
    - To discriminate between star and galaxy in Astronomy
    - To partition customers into market segments in Marketting
    - To cluster protein sequences in Bioinformatics

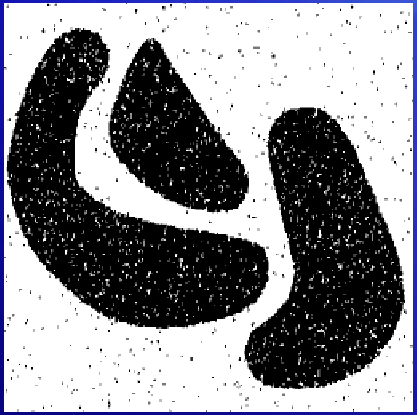
WaveCluster Approach

- Is novel clustering approach with multi-resolution feature based on wavelet transform
- Defines the notion of cluster as a dense region consisting of connected components in the transformed feature space
- Has the ability to discover of clusters with arbitrary shapes
- Can deal with outliers (data points that don't belong to any cluster) effectively

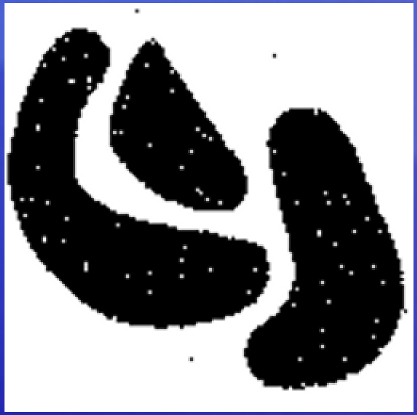
WaveCluster Approach



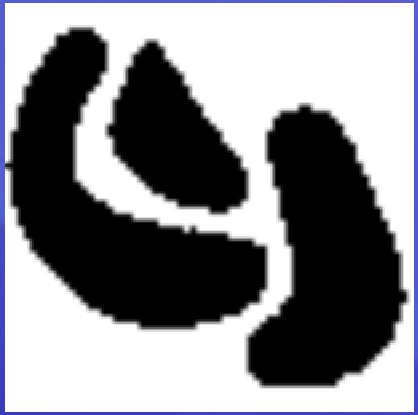
- Wavelet transform is used to decompose signal into high-frequency and low-frequency components



a) Input Dataset



b) ρ = 2, # of cluster = 6



c) ρ = 3, # of cluster = 3

ρ (scale level) represents how many times wavelet transform is applied on the feature space

Parallel Computing on GPUs and NVIDIA CUDA Technology

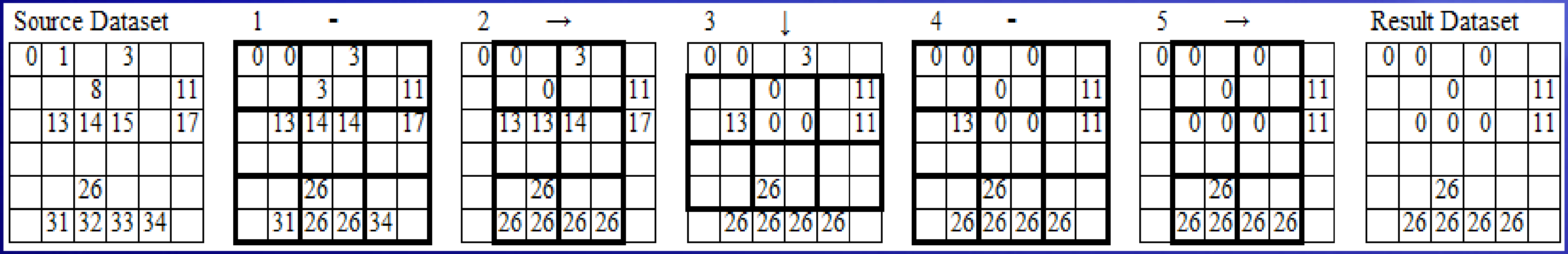
- GPU runs the computationally intensive data-parallel parts of the application as a co-processor
- NVIDIA introduced CUDA (Compute Unified Device Architecture) in November 2006 to enable data-parallel general purpose computations on NVIDIA GPUs
- GPU is regarded as a device that runs hundreds of concurrent lightweight threads
- Each thread is distinguished by the multi-dimensional built-in index variables which are automatically assigned by CUDA runtime

Implementation of Low-Frequency Component Extraction

- Divide and conquer approach is followed in the CUDA implementation of this process
- Each thread calculates one approximation value from disjoint 2x2 square-shaped region of feature space
- Kernel is invoked as much as wavelet transformation level value
- At the last level, thresholding operation is performed to remove outlier points
- Maximum float value is assigned to background points and unique thread index value to foreground points

Implementation of Connected Component Labeling

- Multi-pass algorithm based on sliding window approach
- Each thread is responsible of 2x2 square-shaped field forming a window where calculates the minimum value of these points and assigns the minimum value to its foreground points
- The window slide directions are right, down and steady
- When no point value are changed in all three sub-operations, the algorithm halts



Source Dataset

0	1		3		
		8			11
	13	14	15		17
		26			
	31	32	33	34	

1 -

0	0		3		
		3			11
	13	14	14		17
		26			
	31	26	26	34	

2 →

0	0		3		
		0			11
	13	13	14		17
		26			
	26	26	26	26	

3 ↓

0	0		3		
		0			11
	13	0	0		11
		26			
	26	26	26	26	

4 -

0	0		0		
		0			11
	13	0	0		11
		26			
	26	26	26	26	

5 →

0	0		0		
		0			11
	0	0	0		11
		26			
	26	26	26	26	

Result Dataset

0	0		0		
		0			11
	0	0	0		11
		26			
	26	26	26	26	

Results

Processor Specifications:

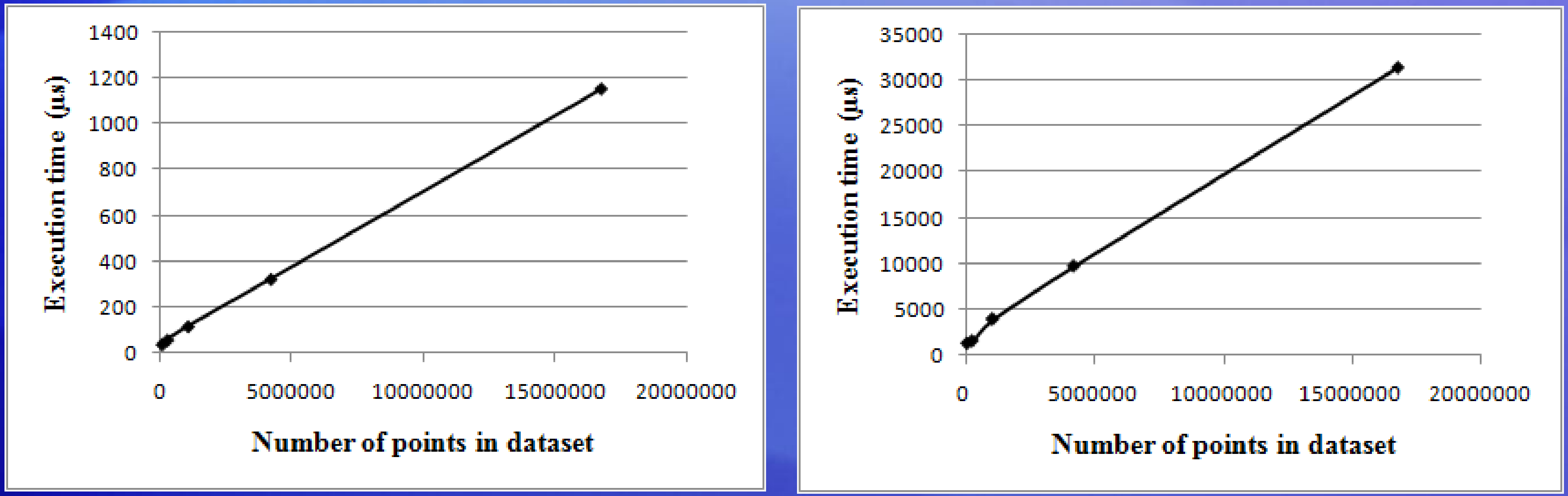
CPU:
Intel Core2Duo
(2 Cores, 2.4 GHz, 4MB L2 Cache)

GPU:
NVIDIA GTX 465 (1 GB memory, 352 computing cores, each core runs at 1.215 GHz)

Results

Dataset Size	Number of Points	Extraction of LF	CCL	Aggregate Speedup
		Kernel Speedup	Kernel Speedup	
256	65536	14.17	0.87	0.02
512	262144	37.49	2.74	0.1
1024	1048576	69.62	3.99	0.37
2048	4194304	97.31	5.94	1.14
4096	16777216	107.1	5.56	2.18

Results



(a)

(b)

Execution times of the kernel (in microseconds) for varying number of points in the dataset (a) Extraction of Low Frequency Component; (b) Connected Component Labeling

Conclusion

- CUDA algorithms expose good speedup values as dataset size increase

- The execution times of both CUDA algorithms scales nearly linear with the number of points in the used dataset

- The data transfer between CPU and GPU may introduce a considerable latency delay which is known as the main bottleneck on GPU computation

Thank You…