

“Compiler and Runtime Support for Enabling Generalized Reduction Computations on Heterogeneous Parallel Computations”

By Ravi, Ma, Chiu, & Agrawal
Presented by Julian Bui

Outline

- Goals
- The focus problems
- Key ideas
- Results



Goals

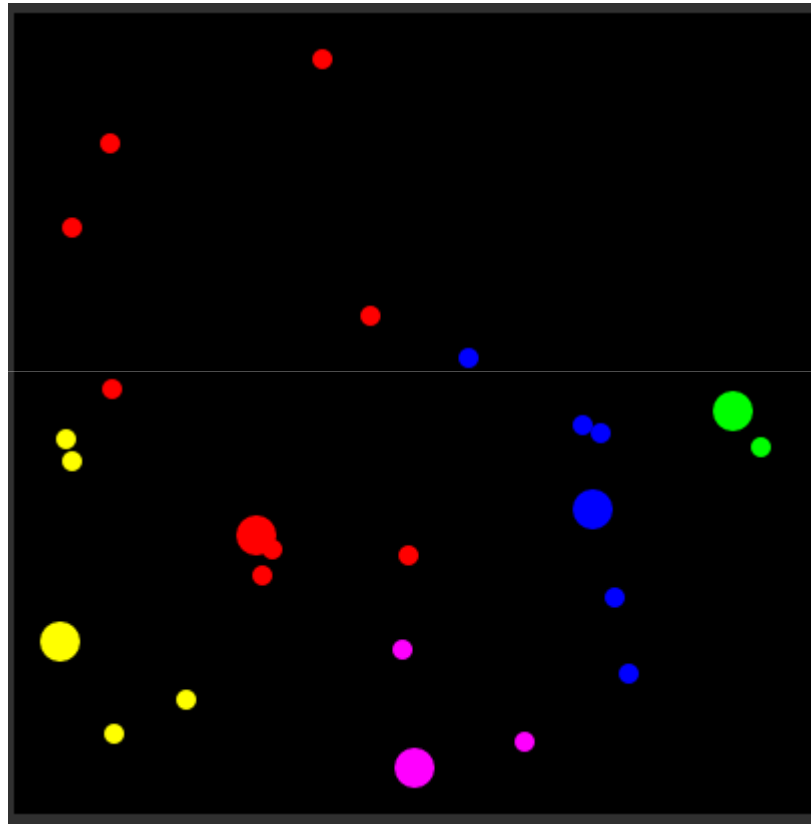
- Programmability
- Performance
- Effective work distribution



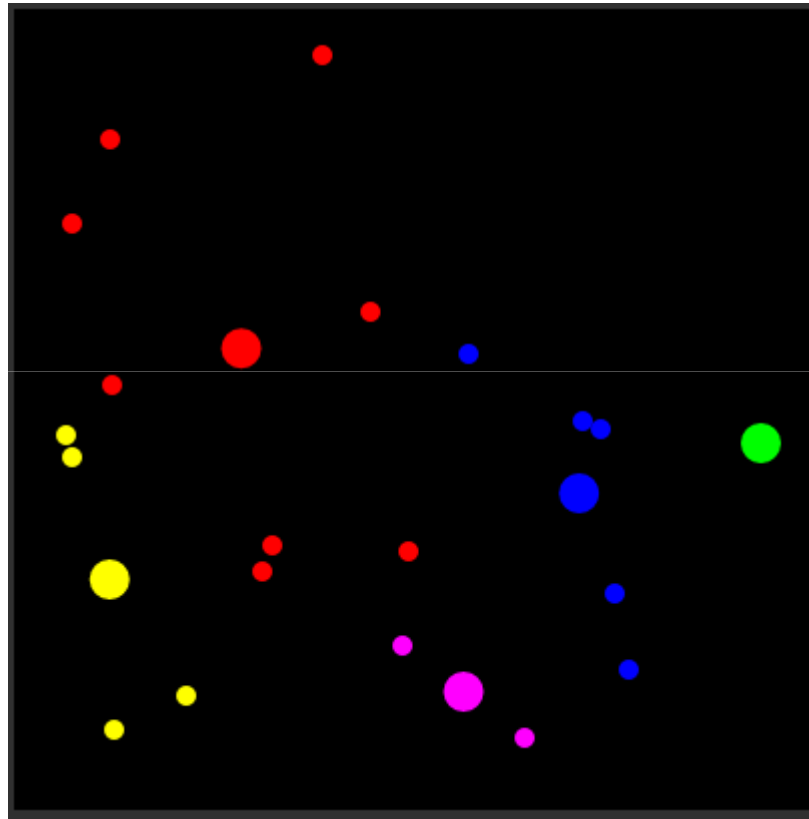
The Focus Problems

- K-Means Clustering
- Principal Components Analysis

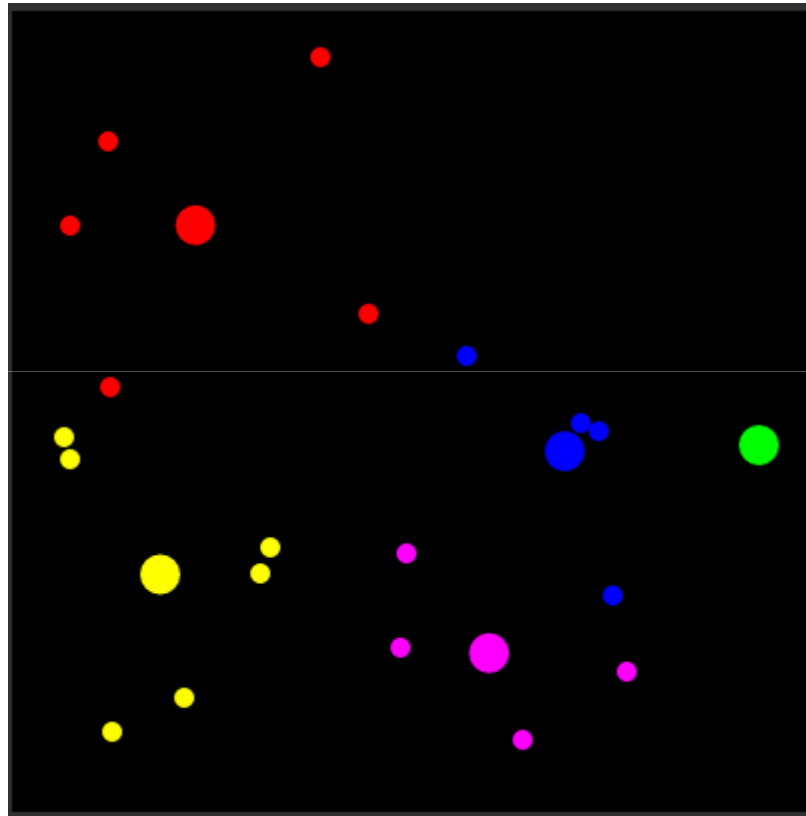
K-means clustering - 1



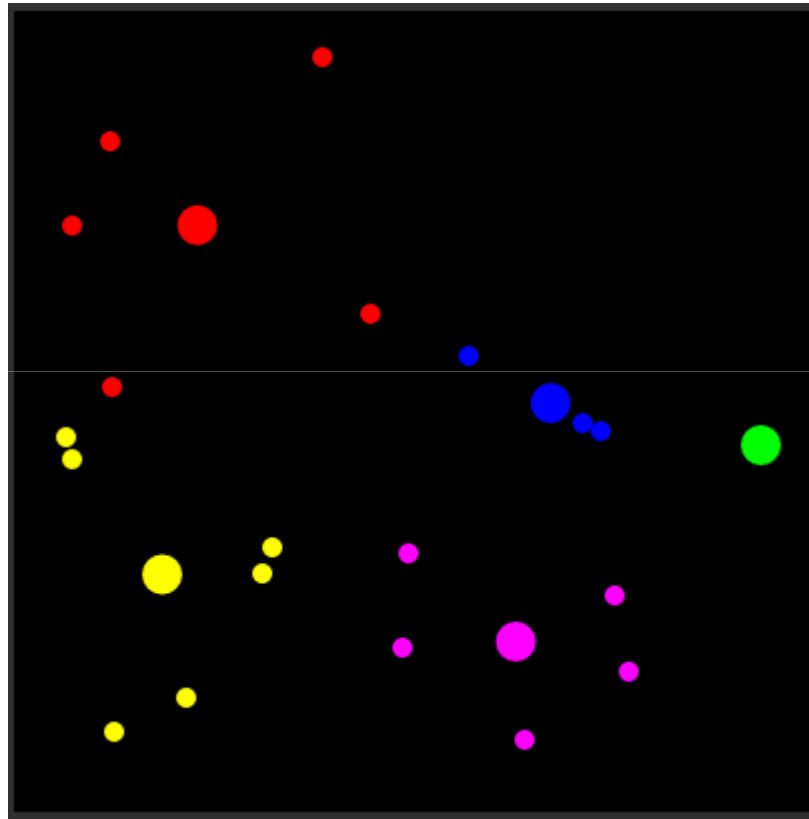
K-means clustering - 2



K-means clustering - 3



K-means clustering - 4



K-Means Clustering

- K-Means Clustering
 1. Randomly guess k cluster center locations
 2. Each datapoint figures out which cluster center it's closest to
 3. Center now "owns" a set of points
 4. Cluster calculates the actual center of the points it owns
 5. Cluster's center point now becomes the actual center point
 6. Repeat steps 2-5 until the cluster's dataset doesn't change between iterations



K-Means Clustering

$$J = \sum_{j=1}^K \sum_{n \in S_j} |x_n - \mu_j|^2$$

Centroid of the cluster

For all clusters

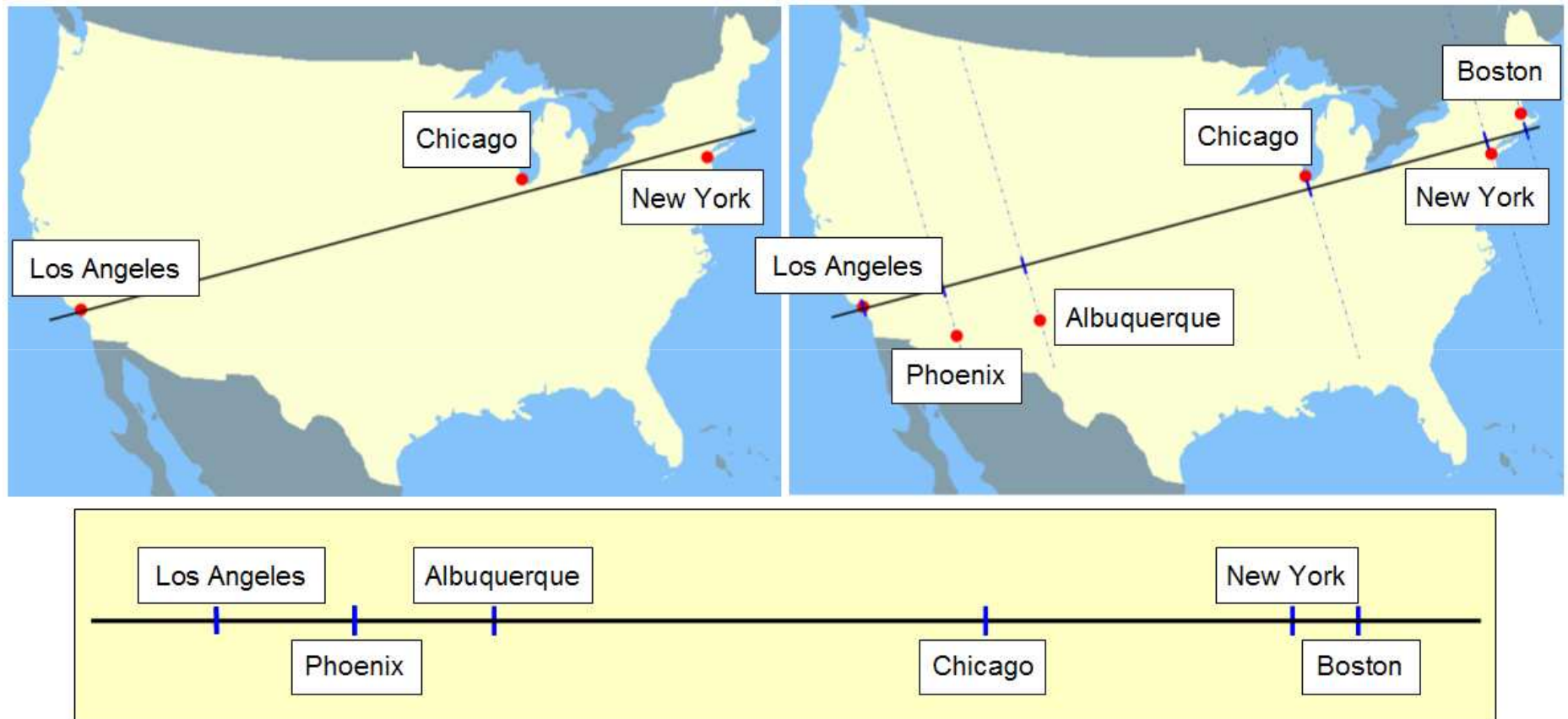
Principal Component Analysis (PCA)

- Goals
 - Dimensionality Reduction
 - INPUT: set of M-dimensional pts
 - OUTPUT: set of D-dimensional pts where $D \ll M$
 - Extract patterns in the data, machine learning
 - Transforms possibly correlated data into a smaller number of uncorrelated data
 - Principal components account for the greatest possible statistical variability

PCA

- Principal components are found by extracting eigenvectors from the covariance matrix of the data

PCA - Simple Example



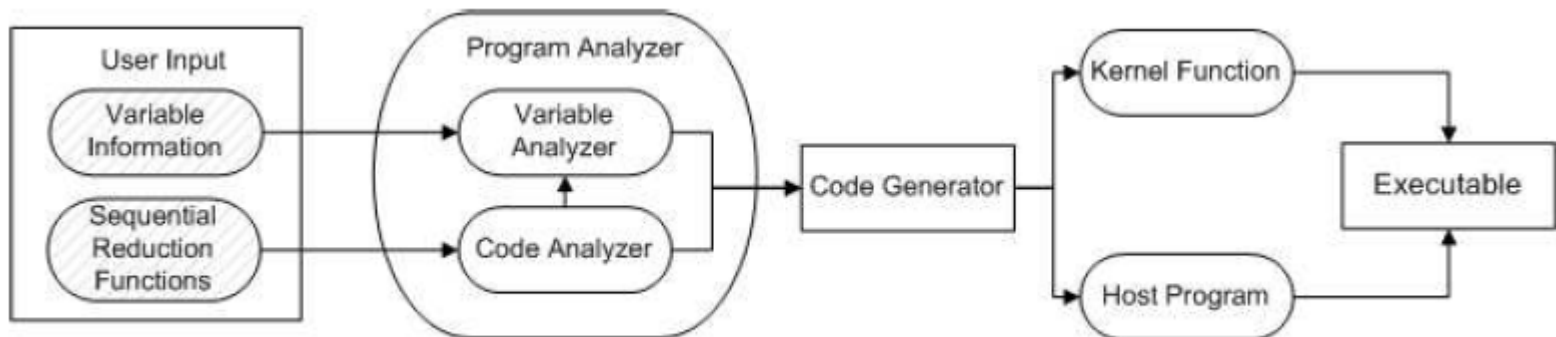


PCA – Facial Recognition



So how does this work apply to those problems?

- Code generator input:
 - Reduction function
 - Variable list (data to be computed)
- Code Generator output
 - Host functions
 - Kernel code



System Architecture

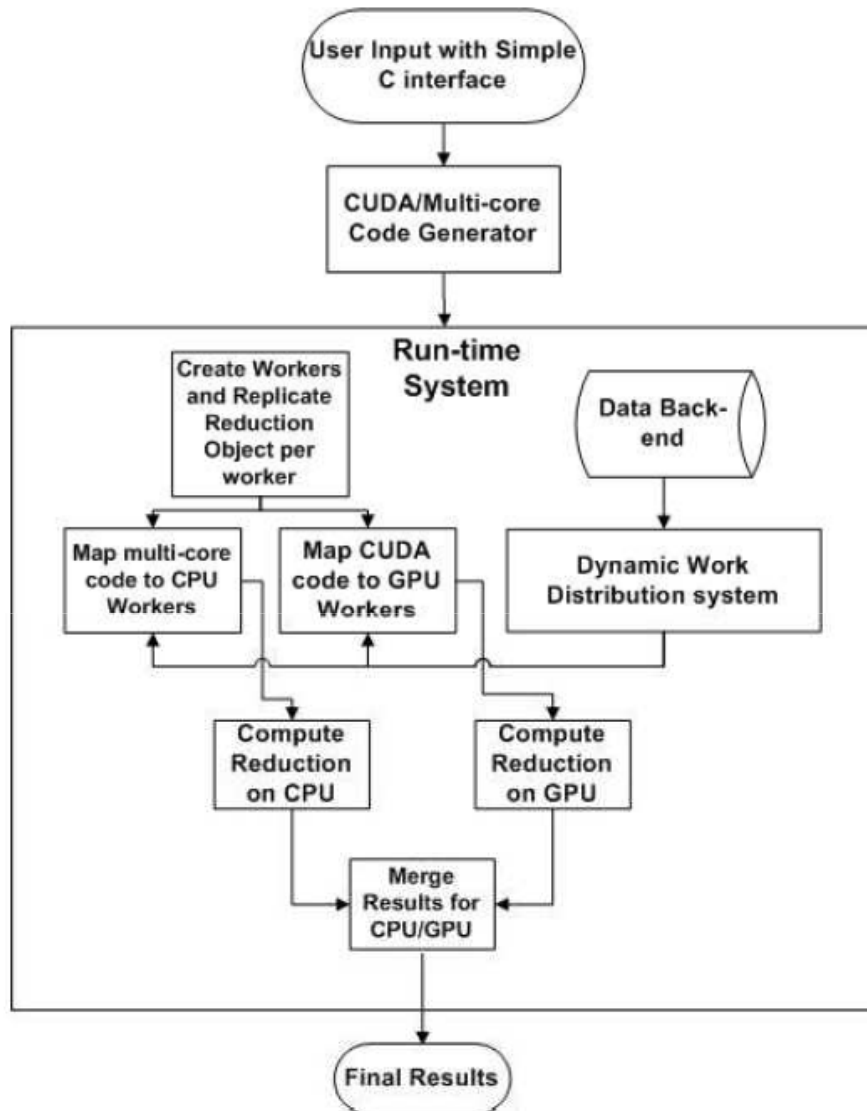
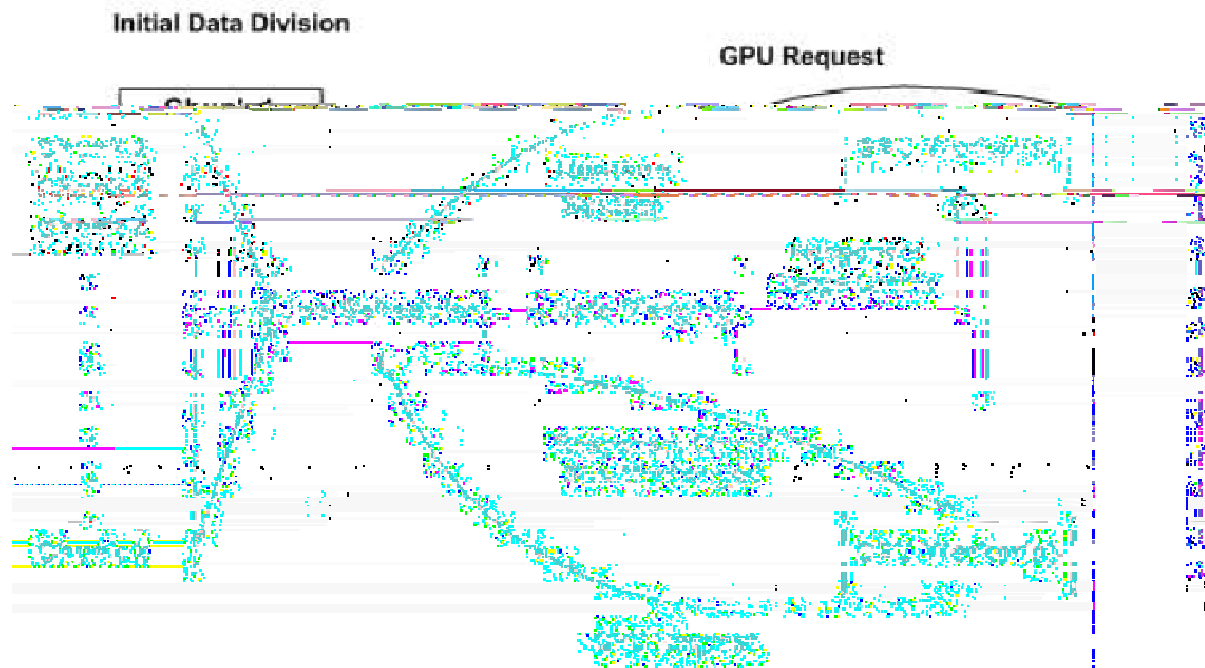


Figure 3: High-level Architecture of the System

Work Distribution

- Work Sharing vs. Work Stealing
- Uniform vs. Non-uniform chunk sizes



Experiments

Machine: AMD Opteron 8350 w/ 8 cores and 16 GB of main memory, GeForce9800 GTX w/ 512MB memory

K-Means: $K = 125$, 6.4GB file, 100M 3D points

PCA: 8.5GB data set, covariance matrix width of 64

GPU Chunk Size & Num. Threads vs. Performance

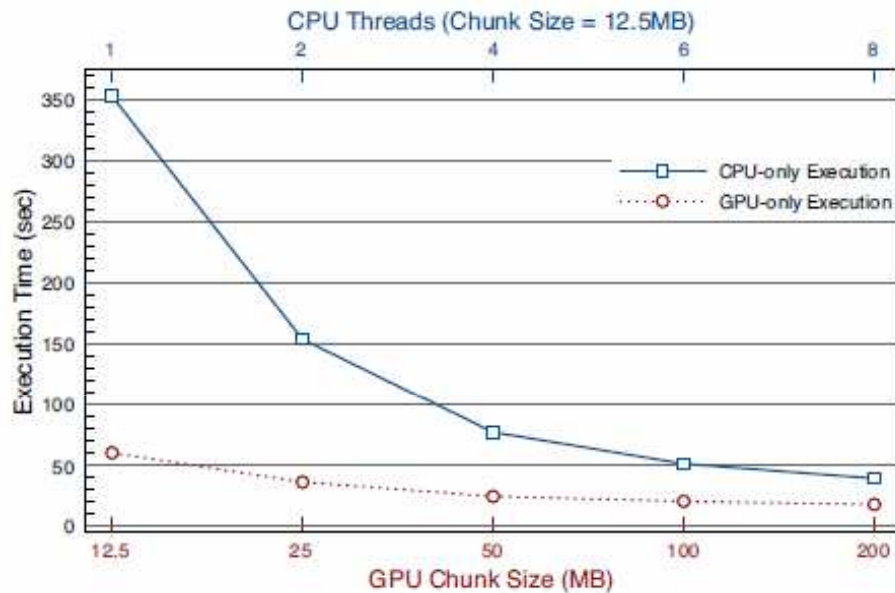


Figure 7: Scalability of K-Means with CPU-only and GPU-only

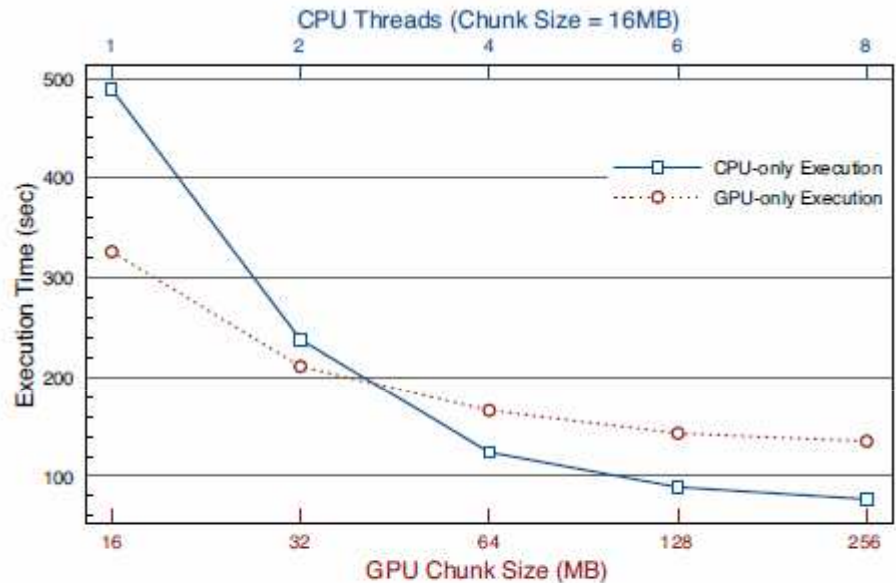


Figure 8: Scalability of PCA with CPU-only and GPU-only

K-Means, Hetero., Uni vs. Non-uniform chunk sizes

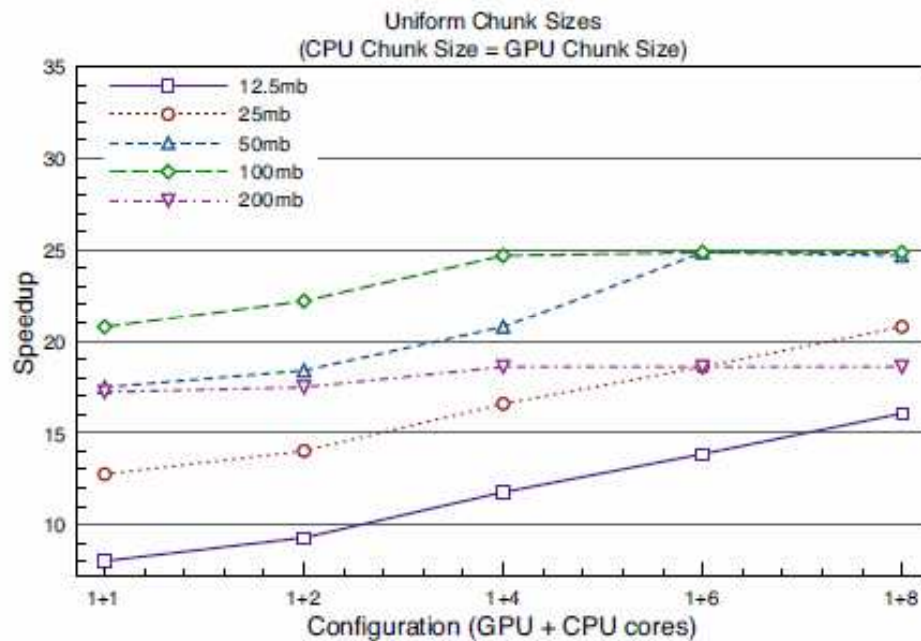


Figure 9: K-Means Using Heterogeneous Version with Uniform Chunk Size

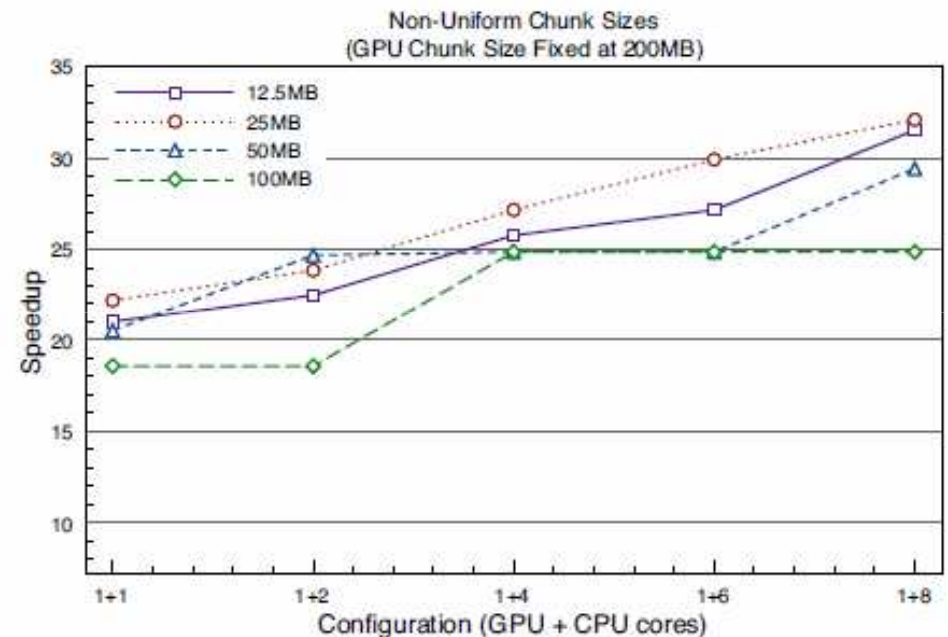


Figure 10: K-Means Using Heterogeneous Version with Non-Uniform Chunk Size

PCA, Hetero., Uni vs. Non-uniform chunk sizes

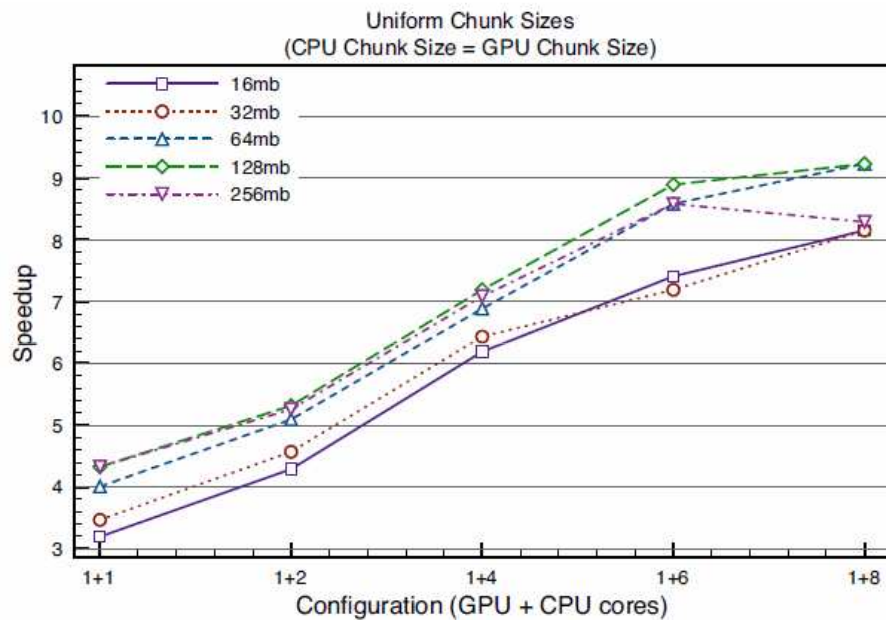


Figure 11: PCA Using Heterogeneous Version with Uniform Chunk Size

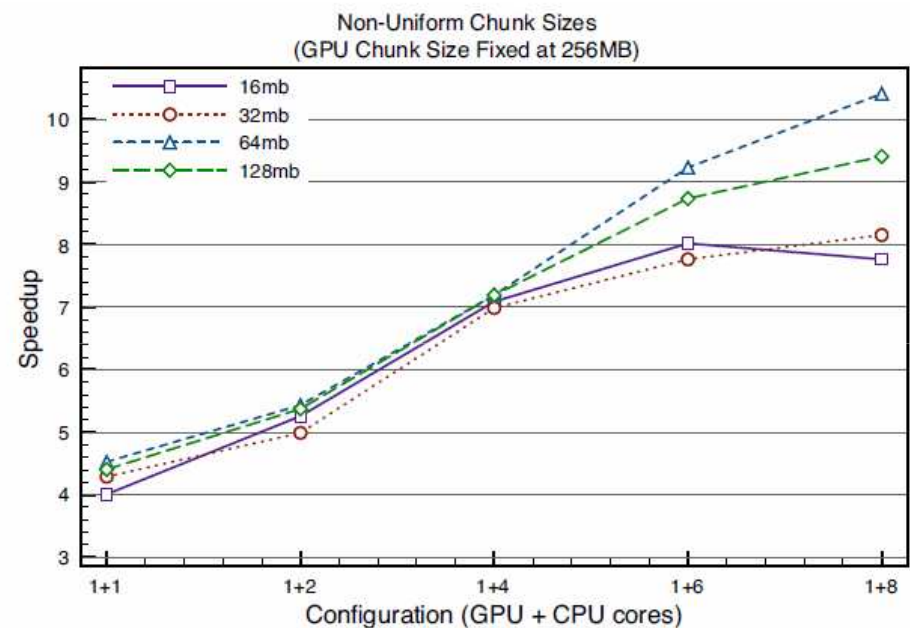


Figure 12: PCA Using Heterogeneous Version with Non-Uniform Chunk Size

Idle Time

K-Means		PCA	
Chunk Size (MB)	Idle %	Chunk Size (MB)	Idle %
100	35.2	128	11.3
200	45	256	16.9

Table 1: % Idle with Uniform Chunk Size

Work Distribution – K-Means

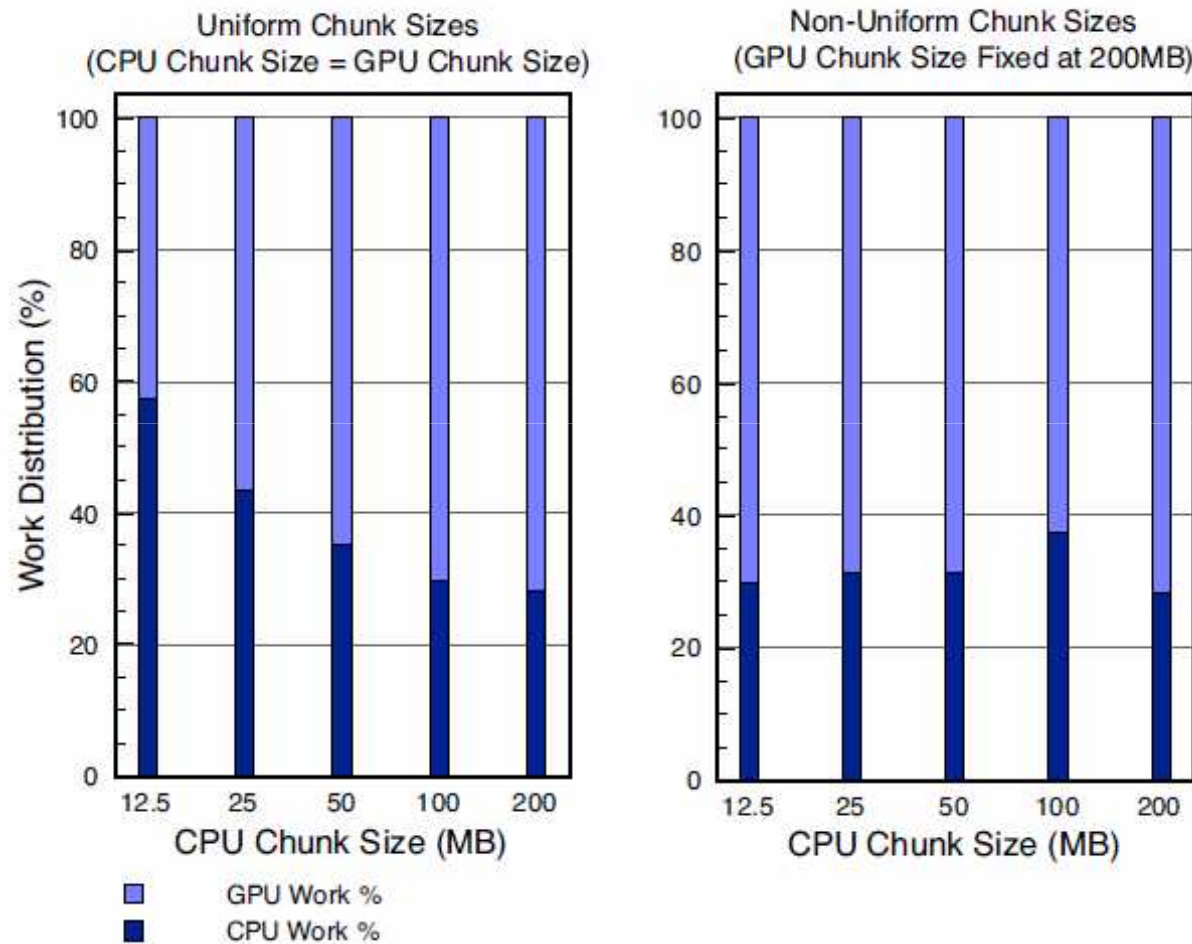


Figure 13: Work Distribution (K-Means)



Work Distribution – PCA

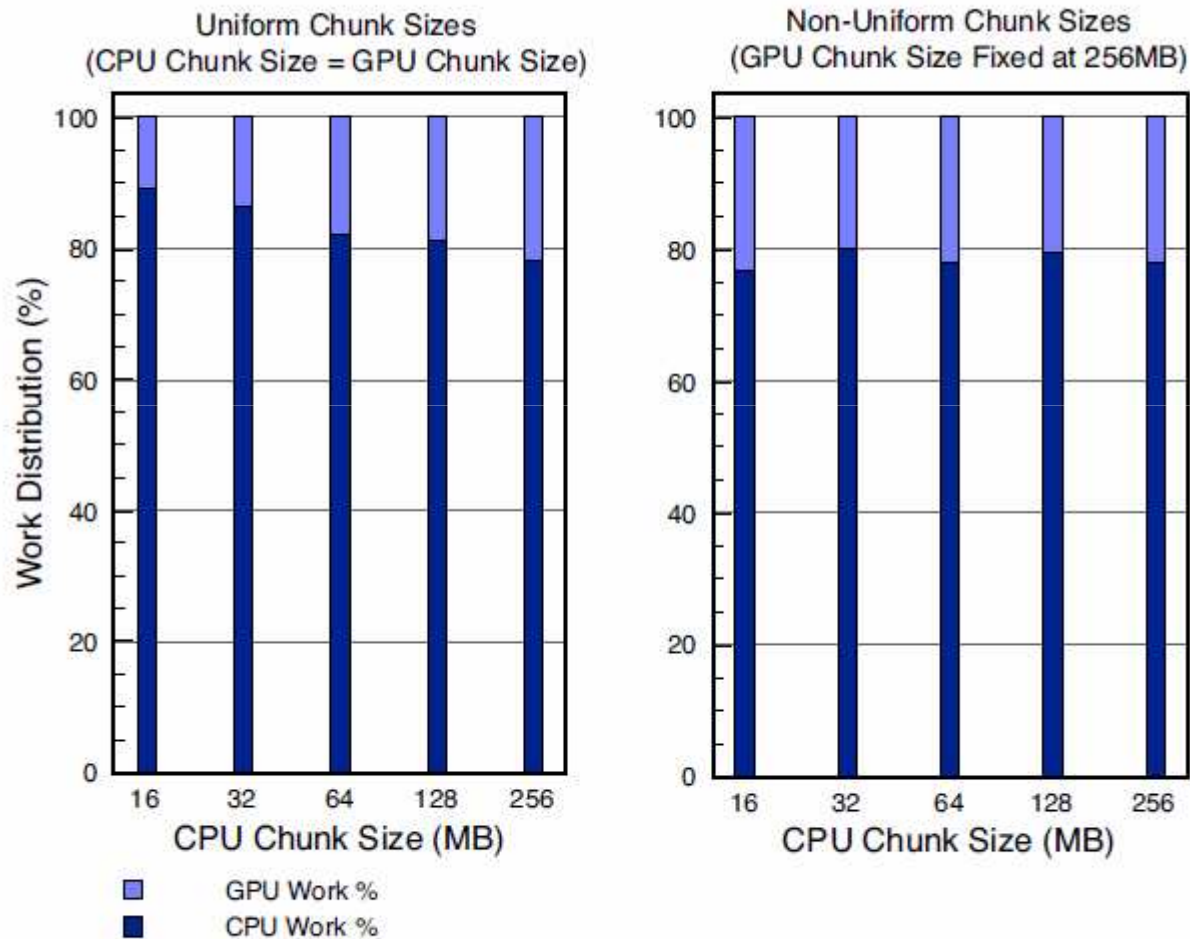


Figure 14: Work Distribution (PCA)