

K-Means Parallel Computing Documentation

Lecturer: Dr. Boris Morose

Student: Orel Damari

Student ID: 204739825



K-Means Algorithm:

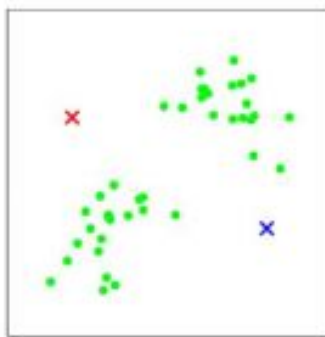
k-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells.

In this project I Implemented the K-Means in the following steps:

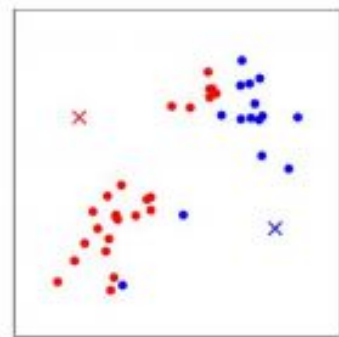
1. Calculating number of points
2. Calculate distances
3. Calculate clusters
4. Move points according to Velocity
5. If Quality Measure not enough and max time did not achieved : Back to step 2



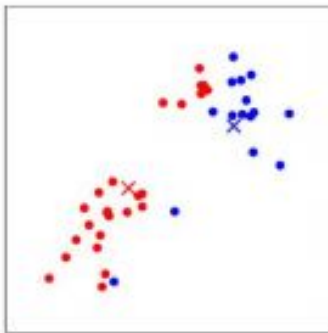
(a)



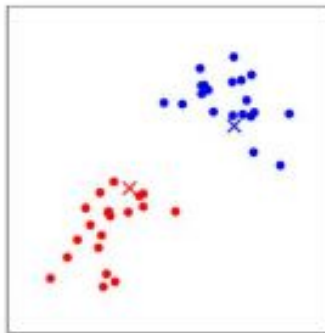
(b)



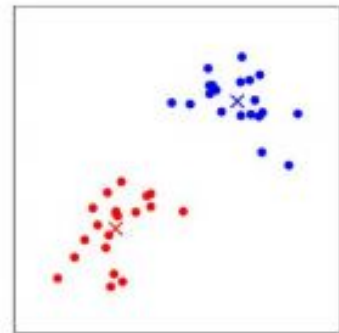
(c)



(d)



(e)



(f)

What Parallelized?

MPI (Message Passing Interface):

Sharing & dividing the points between the processes.



Cuda:

Moving the points

Calculating distances

Calculating clusters

Classifying point to cluster



OpenMP:

Quality Calculations



IMPLEMENTATION:

I Implemented the K-Means in the following way:

1. Read file, number of points, Clusters, Quality Measure, Time, dT.
2. Calculate number of points for every process. If there is a remainder it stays at the Master process.
3. Using MPI - send array of points to the processes.
4. Every process know which points from this array to read- according to the MPI process id.
5. Every process calculate the number of blocks needed in the cuda- According to the check of how many max threads can be in every block.
6. The Cuda move the points- One gpu thread to each point.
7. Calculate distances and new Cluster- using cuda.
8. Returning an array that classifies every point to the right Cluster.
9. Using OMP - Calculate the Quality Measure.
10. Check if Maximum time iteration achieved or Quality Measure achieved.
11. If Yes- Finish and write to file the clusters.
12. If No- go back to step 3.

vectorAdd130617_0...pture_000.nvreport x Activity1.nvact*

CUDA Source View

vectorAdd Grid Dim: {196, 1, 1} Block Dim: {256, 1, 1} Duration: 5.952 μ s Compute Capability: 3.0

File: vectoradd.cu View: Source and SASS High to Low: A Low to High: A

Source

```
/**
 * CUDA Kernel Device code
 *
 * Computes the vector addition of A and B into C. The 3 vectors have
 * number of elements numElements.
 */
__global__ void
vectorAdd(const float *A, const float *B, float *C, int numElements)
{
    int i = blockDim.x * blockIdx.x + threadIdx.x;

    if (i < numElements)
    {
        C[i] = A[i] + B[i];
    }
}
```

Line Address Source

1	33C08	MOV R1, c[0x0][0x44];
2	33C10	S2R R0, SR_CTaid.X;
3	33C18	S2R R3, SR_TID.X;
4	33C20	IMAD R0, R0, c[0x0][0x28], R3;
5	33C28	ISETP.GE.AND P0, PT, R0, c[0x0][0x14c], PT;
6	33C30	@!P0 BRA.U 0x78; # Target=0x00033c78
7	33C38	@!P0 ISCADD R3, R0, c[0x0][0x140], 0x2;
8	33C48	@!P0 ISCADD R2, R0, c[0x0][0x144], 0x2;
9	33C50	@!P0 LD.CG R3, [R3];
10	33C58	@!P0 ISCADD R0, R0, c[0x0][0x148], 0x2;
11	33C60	@!P0 LD.CG R2, [R2];
12	33C68	@!P0 FADD R3, R3, R2;
13	33C70	@!P0 ST [R0], R3;
14	33C78	EXIT;
15	33C80	BRA 0x80; # Target=0x00033c80
16	33C88	NOP;
17	33C90	NOP;

Formulas:

Quality Measure: (for k=3)

$$q = (d1/D12 + d1/D13 + d2/D21 + d2/D23 + d3/D31 + d3/D32) / 6$$

Distance between two points:

$$D(X1-X2) = \sqrt{(X2^2 - X1^2)}$$

Points Movement:

$$x_i(t) = x_i + t \cdot v_{xi}$$

$$y_i(t) = y_i + t \cdot v_{yi}$$

KMeans:

Given a set of observations (x_1, x_2, \dots, x_n) , where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into k ($\leq n$) sets $S = \{S_1, S_2, \dots, S_k\}$ so as to minimize the within-cluster sum of squares (WCSS) (i.e. variance). Formally, the objective is to find:

$$\arg \min_S \sum_{i=1}^k \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \arg \min_S \sum_{i=1}^k |S_i| \text{Var } S_i$$

where $\boldsymbol{\mu}_i$ is the mean of points in S_i . This is equivalent to minimizing the pairwise squared deviations of points in the same cluster:

$$\arg \min_S \sum_{i=1}^k \frac{1}{2|S_i|} \sum_{\mathbf{x}, \mathbf{y} \in S_i} \|\mathbf{x} - \mathbf{y}\|^2$$

The Equivalence can be deduced from identity

$$\sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \sum_{\mathbf{x} \neq \mathbf{y} \in S_i} (\mathbf{x} - \boldsymbol{\mu}_i)(\boldsymbol{\mu}_i - \mathbf{y})$$

Because the total variance is constant, this is also equivalent to maximizing the squared deviations between points in *different* clusters (between-cluster sum of squares, BCSS).