K-means Algorithm Parallelization

Nome: Darwin Saire Pilco

Parallelization
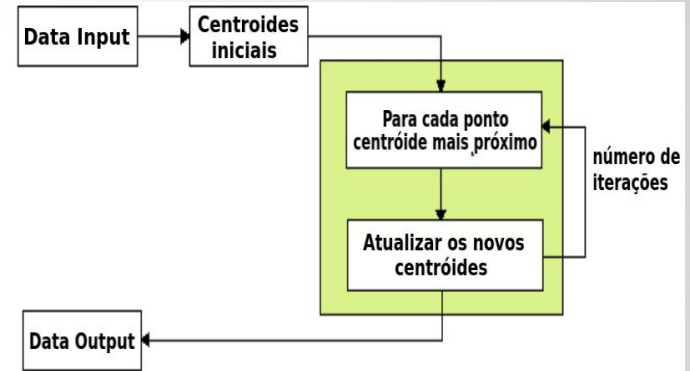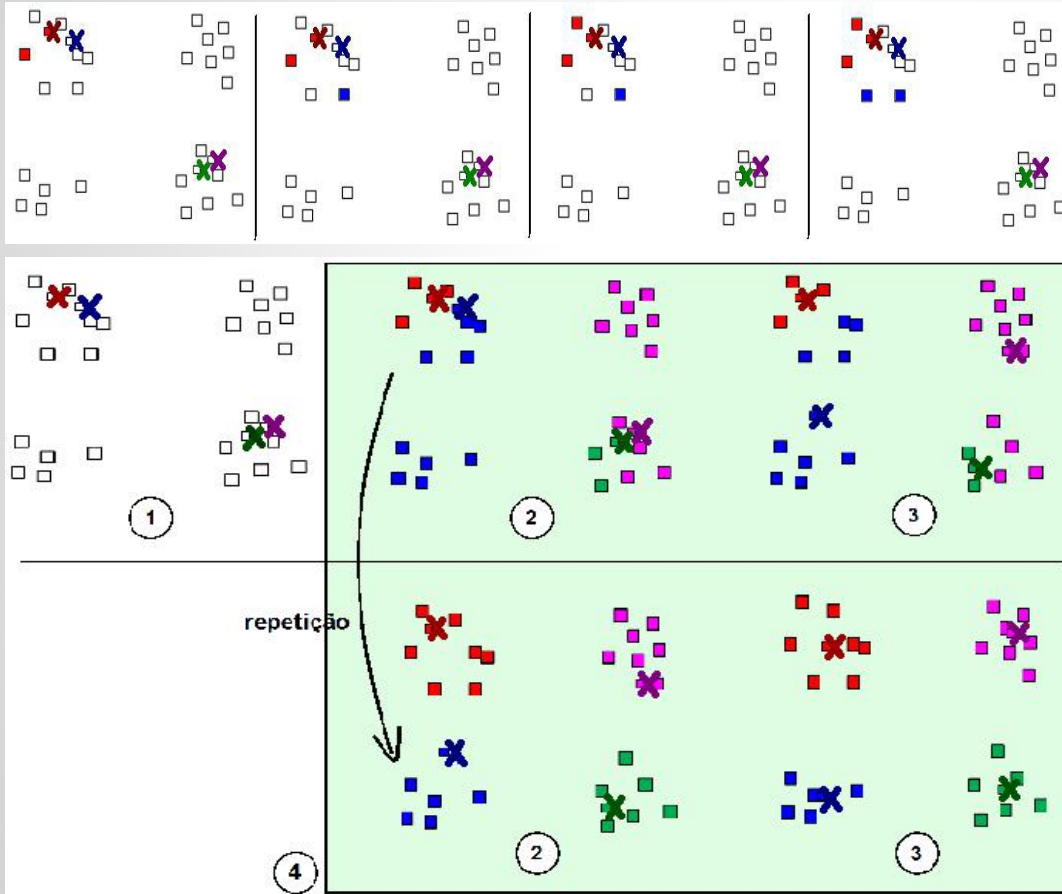
Profile

Source	CPU Time
/*selecionar o cluster mais próximo para cada ponto e sumar as dimensões em newCluster	
para obtem um novo cluster*/	
for(i = 0; i < num_point; i++){	
nearest_cluster = 0;	
dist = 0.0;	
min_dist = INF;	
for(j = 0; j < numClusters; j++){	0.012s
dist = 0.0;	
for(k = 0; k < numdim; k++)	0.291s
dist += (dim_point[i][k] - clusters[j][k]) * (dim_point[i][k]-	6.674s
if(dist < min_dist){	0.044s
min_dist = dist;	0.028s
nearest_cluster = j;	
}	
}	
belong_point[i] = nearest_cluster;	
newClusterSize[nearest_cluster]++;	
for(j = 0; j < numdim; j++)	
newClusters[nearest_cluster][j] += dim_point[i][j];	0.251s
}	

for(i = 0; i < num_point; i++){	0.008s
nearest_cluster = func_find_nearest_cluster(numClusters, numdim, dim_point[i], cluste	
belong_point[i] = nearest_cluster;	
newClusterSize[nearest_cluster]++;	
for(j = 0; j < numdim; j++)	0.056s
newClusters[nearest_cluster][j] += dim_point[i][j];	0.338s
}	

int func_find_nearest_cluster(int numClusters, int numdim, float *dim_point, float **clus.	
int i, nearest_cluster = 0;	
float dist;	
float min_dist = INF;	
for(i = 0; i < numClusters; i++){	0.008s
dist = func_distance(numdim, dim_point, clusters[i]);	0.068s
if(dist < min_dist){	0.092s
min_dist = dist;	
nearest_cluster = i;	
}	
}	
return nearest_cluster;	
}	

float func_distance(int numdim, float *dim_pointX, float *dim_pointY){	
int i;	
float ans = 0.0;	
for(i = 0; i < numdim; i++)	0.539s
ans += (dim_pointX[i] - dim_pointY[i]) * (dim_pointX[i]- dim_pointY[i]);	4.193s
return(ans);	0.020s
}	0.100s

Speedup

http://cs.joensuu.fi/sipu/datasets/
N = 1024 e k = 16

	dim128.txt			dim256.txt			dim512.txt	
	Time(s)	SpeeUp		Time(s)	SpeeUp		Time(s)	SpeeUp
sequencial	8,2399			16,1725			32,1575	
omp nt = 2	4,4694	1,84		8,4827	1,91		16,5957	1,9377
omp nt = 4	2,682	3,07		4,625	3,4968		8,9564	3,5905
omp nt = 8	1,923	4,28		2,8297	5,7151		5,1127	6,2897
omp nt = 16	1,8488	4,46		2,1247	7,6116		3,1544	10,1948
gpu	2,11	3,91		4,27	4,2743		7,33	4,487