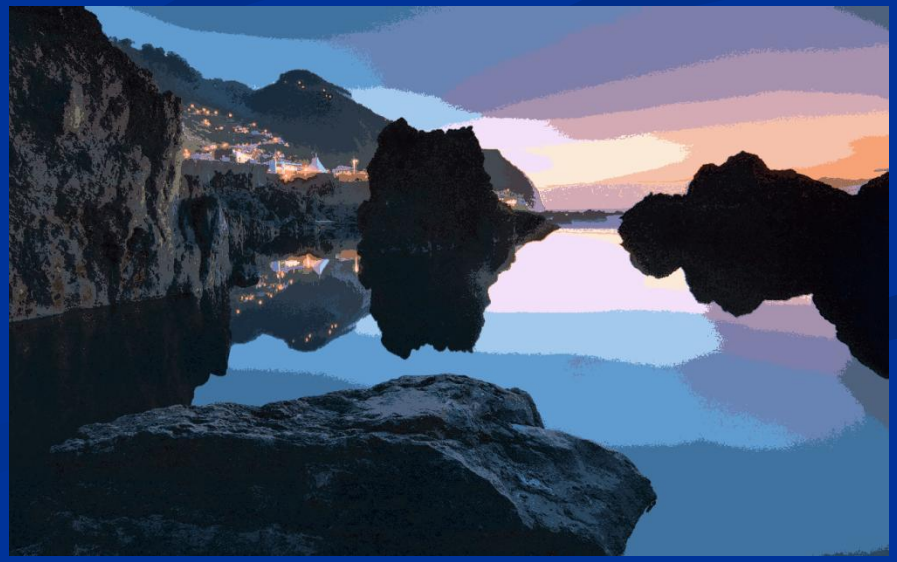
Parallelizing K-means for Image Segmentation

Chris Jordan, Victor Janmey, and Raymond Ko

Problem and Applications

- Group points into k clusters
- Machine Learning
  - Unsupervised learning methods
  - Dividing data for more specific classifiers
  - Selection of examples for k-nearest neighbors
- Computer Vision
  - Image segmentation
  - Estimating object boundaries
  - Cue for object recognition



K-means Algorithm

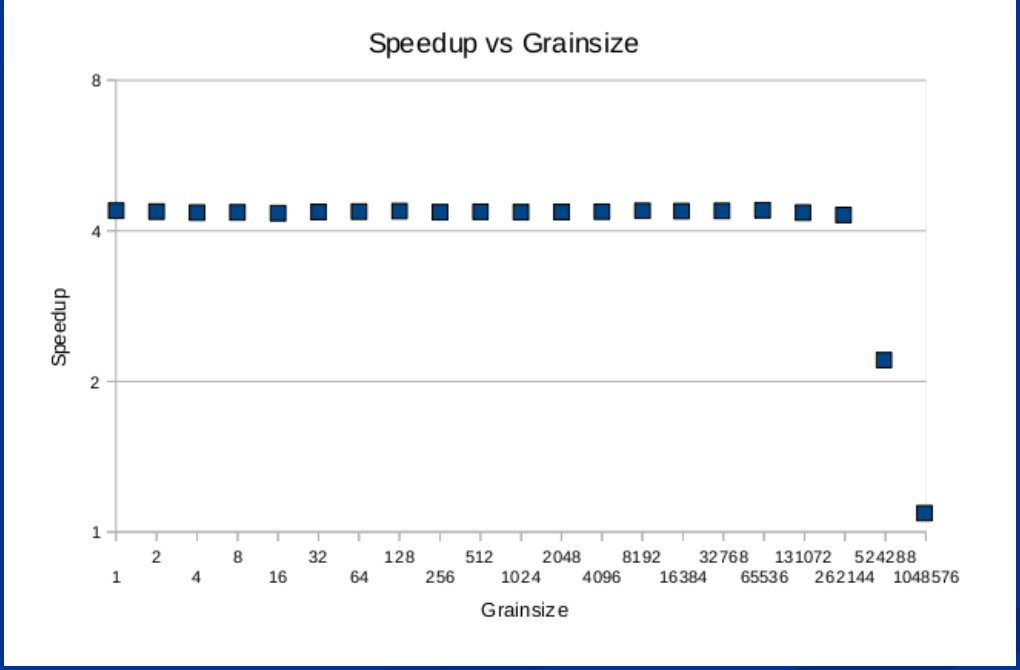
- Initialization:
  - Select initial centers
- Iteration:
  - Step 1:
    Assign each particle to nearest cluster
  - Step 2:
    Recompute cluster centers
  - Repeat until convergence
- Step 1 dominates since is it is (k*n) vs. O(n) for step 2

Implementation

- C++ and OpenCV image library
- Parallel implementation with TBB
- Cluster assignment: parallel_for
- Cluster center calculation: parallel_reduce

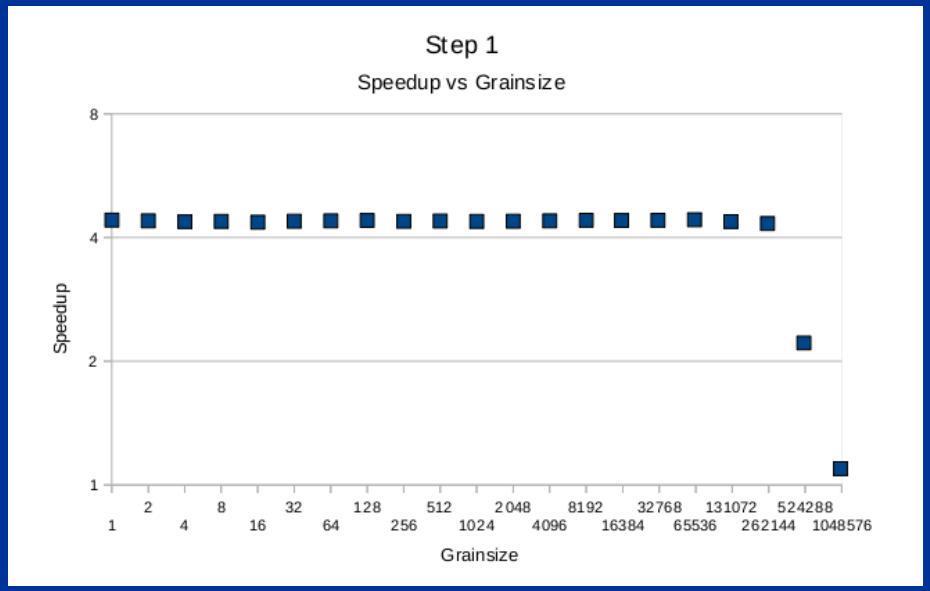
Preliminary Results

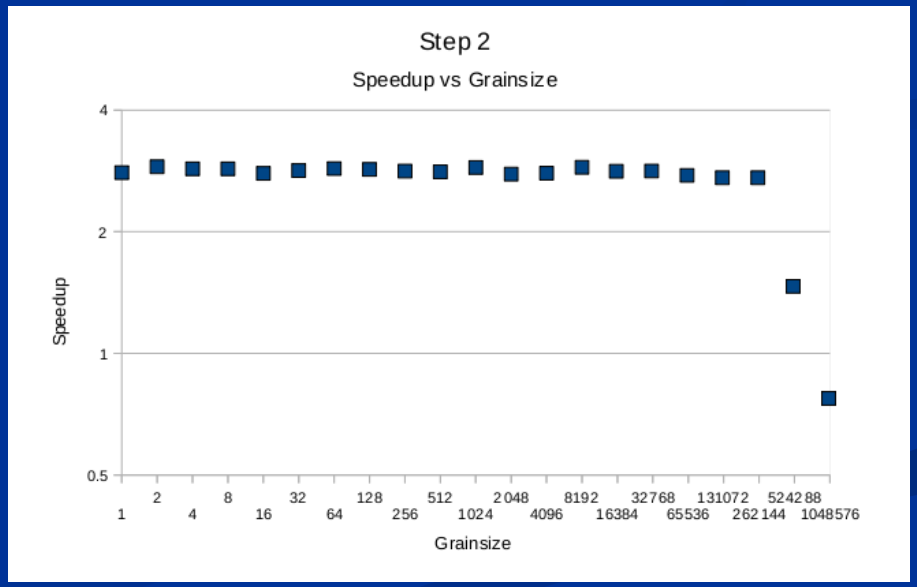
- Sample run with 50 clusters on 4 cores
- 1,024,000 pixels over 3 dimensions (HSV)
- 4.4 speedup at Grain Size: 64,000
- Superlinear because of loop unrolling in TBB?



Results by Step

- Step 1 speedup: 4.42 at grain size of 64,000
- Step 2 speedup: 2.89 at grain size of 8,192





Next Steps

- OpenMP
- Experiments
  - Speedup vs. Number of Particles
  - Speedup vs. Number of Cores
  - Speedup vs. Number of Clusters
- Algorithm improvements
  - K-means++
  - Triangle inequality