K-means is the most widely used clustering algorithm. The increasing availability of large databases and streaming and distributed data has driven the development of novel k-means implementations. Two important classes are single pass and parallel algorithms. Single pass algorithms (ideally) process each data item only once and are useful for streaming data and very large data sets. Parallel algorithms take advantage of multiple processors, local or remote. A SQL implementation is also included to highlight the use of declarative languages and industry-centric applications of k-means.

### Single Pass

**Scalable K-means**
- Complex, single pass framework for clustering large data with limited RAM
- 2 levels of compression track points that may be needed and stats of those not needed
- Multiple mini-clustering passes using k-means and hierarchical clustering impact performance

**Simple Single Pass K-means**
- Removes complex machinery from ScaleKM
- Clusters RAM cache, saves suff. stats
- No decline in cluster quality
- Can be faster than standard k-means

**Incremental K-means**
- Single pass k-means for binary data
- Sparse distance computations
- Reduced sufficient stats
- Fast with high quality results

**Fast and Exact K-means**
- Few (1-3) iterations for most data sets
- Equivalent to standard k-means
- k-means on sample followed by one full scan

### Parallel

**Parallel K-means**
- Natural parallelism on nearest neighbor calculation
- Linear speedup

**Distributed Framework**
- NN parallelism in distributed environments
- Only communicate sufficient stats
- Argues for in-place clustering on distributed resources for data > 50k
- K-means, k-harmonic means, EM

**GPU K-means**
- NN parallelism using GPU pipelines
- SIMD on distance vectors
- Depth buffer contains nearest centroid

### Database

**SQL K-means**
- K-means entirely in SQL
- Use DB index structures to improve performance