Experiences with Spark – Building K-means

* Purpose: Status report on my first semester working with Spark
* Outline: what I did, what I learned, future plans
* k-means
  + Quick intro
  + Distributed flowchart
  + Characteristics: iterative, typically CPU-bound rather than network-bound
    - Unless dealing with very high-dimensional points and centroids
    - Or using an data-intensive stopping condition
* Spark implementation
  + Implementation is simple, but parallelizing it is slightly less trivial than the current Spark examples because of the extra step of calculating partial sums
    - (side-by-side code)
* Collected benchmark data on…
  + Increasing dataset size
    - 10 million to 2.5 billion 2D points, or ~150 MB to ~38 GB
    - 2 m1.large slaves on EC2
    - Linear size vs. time, as expected
  + Scale-out
    - Slightly sublinear speedup
  + Scale-up
    - As expected, k-means is CPU-bound so instances with higher CPU performed better
  + Cache sizes and strategies
    - SoftReferenceCache and BoundedMemoryCache are the fastest
    - memoryFraction is important
* Experiences:
  + Spark is concise
    - K-means implementation:
      * On Windows Azure in C#: find . –iname \*.cs | xargs wc –l => 5690 lines
      * On Spark in Scala: find . –iname \*.scala | xargs wc –l => 239 lines
      * Not really a fair comparison because of different levels of abstraction and differences in the algorithm implemented – just for fun ☺
  + As we know, debugging a distributed system can be frustrating
    - Tasks get lost with no explanation up-front
    - Finding the reason requires fast reactions
    - Solution: Present framework logs *persistently* on the Mesos dashboard
    - Tasks can also appear to hang because of insufficient heap size, but when you run top there’s no CPU usage. Possible solution: use JVM flags to debug
  + Repeatability is important, so scripting a task and running it on the cloud is a must
    - Current method: scripted setup over SSH after mesos-ec2 has launched the cluster
    - Ideal (avoids setup time): create an AMI with code and pre-generated input data
  + Writing an application with Spark is not well-documented
    - Currently building JARs with a custom Makefile
    - Discovering what works takes trial and error
    - I would be happy to write up the correct way as a wiki page
  + Until recently, Mesos EC2 script needed a way to add instances to an existing cluster
    - Stopping n instances and then launching 2n costs O(3n) instead of O(2n)
    - Solution: Manually launch new instances, run resume
* Future
  + Improve the Spark core
  + Build more applications on Spark: frameworks like Pregel, other machine learning algorithms
  + Improve the Spark interfaces: data visualization, R integration­