Bobby Crimi, Alex Ackerman, Richard Green

Scala Array Problems

//Richard, sorry this took so long: <http://docs.scala-lang.org/sips/completed/scala-2-8-arrays.html>

//on the above page, it cites a few people, if you click on the number next to their name, it will take you //to some really good material. Like I texted you, I have just been having a real hard time picking out //what is important and what isn’t. I would be down to meet with you today to work on this, I think I //just need to talk it out, I usually do better that way

//Paragraph 1: outlines what paper will be about…uses bullet point 1.

In 2008, there was a great deal of discussion regarding the use of arrays in Scala 2.7. To the non-expert eye, it would appear that they were simply the same as Java arrays. However, this is simply not the case, and if used incorrectly, they can negatively impact performance. The key issues with Scala 2.7 stemmed from the fact that ideally, Scala arrays would have the same representation as in Java so that the data could be used interchangeably. Unfortunately, there are a number of reasons why the 2.7 arrays were unable to do this in light of Java’s particularly low-level representation. First off, Java has multiple array representations which is redundant and can cause a certain amount of ambiguity. Also, while 2.7 technically did have constructors for arrays of a generic type, these constructors caused some issues, such as the fact that various data types would not function as well as using uninitialized arrays. Lastly, there were very few effective methods for array indexing and manipulation. This paper will continue to examine these issues, potential fixes, and what was actually implemented in subsequent releases of Scala in order to avoid some of these problems.

1. Scala Array Problems
   1. We want arrays to have same representation as in Java so that we can use this data interchangeably, however Java’s representation is very low level
      1. Java has multiple array representations which is confusing and redundant.
      2. There are no type arrays for arrays of a generic type
         1. In Scala 2.7 there were constructors for arrays of a generic type however, these constructors allowed for many problematic occurrences.
            1. Some data types would not work as well as working with uninitialized arrays
      3. There are very few methods for array indexing and manipulation.

//Paragraph 2/3…use bullet 2. “Ways this can be solved”.

1. Ways this can be solved
   1. Create two different implementations of array
      1. Java representation for interoperation
         1. Would have all the same traits as Java arrays
         2. Would be very fast
      2. Scala representation for use in collection hierarchy
         1. Would have all the nice methods for indexing and manipulation
      3. These two implementations should be interchangeable
      4. Since one would need to make a decision on which type to use, large pieces of code could get very complicated and problematic when both types are used
   2. Wrap Java-like arrays up in Scala arrays
      1. Native arrays would have same implementation as new Scala arrays, however they would still have the same capabilities as the Java arrays.
         1. Similar to String/RichString
      2. String/RichString has been problematic
   3. Ways this was solved in 2.8
      1. In the updated 2.8 Scala Collections, an update to Seq made the integration of arrays possible.
         1. Seq will now inherit its operations from its parent (SeqLike)
      2. First arrays are converted to ArrayOps
         1. Subtype of VectorLike(T, Array[T])
         2. Will have same methods as Array[T]
      3. Then are converted to WrappedArray
         1. Mutable Vectors (sequence)
         2. Has different type of methods
      4. Different methods would invoke different calls for conversion to WrappedArray and ArrayOps