Lazy Vals Initialization

Scala 3 implements Version 6 of the SIP-20 improved lazy vals initialization proposal.

Motivation

The newly proposed lazy val initialization mechanism aims to eliminate the acquisition of resources during the execution of the lazy val initializer block, thus reducing the possibility of a deadlock. The concrete deadlock scenarios that the new lazy val initialization scheme eliminates are summarized in the SIP-20 document.

Implementation

Given a lazy field of the form:

```
class Foo {
  lazy val bar = <RHS>
}
```

The Scala 3 compiler will generate code equivalent to:

```
class Foo {
  import scala.runtime.LazyVals
  var value_0: Int = _
  var bitmap: Long = 0L
  val bitmap_offset: Long = LazyVals.getOffset(classOf[LazyCell], "bitmap")

  def bar(): Int = {
    while (true) {
      val flag = LazyVals.get(this, bitmap_offset)
      val state = LazyVals.STATE(flag, <field-id>)

      if (state == <state-3>) {
        return value_0
      } else if (state == <state-0>) {
```

```
        if (LazyVals.CAS(this, bitmap_offset, flag, <state-1>, <field-id>)) {
          try {
            val result = <RHS>

            value_0 = result
            LazyVals.setFlag(this, bitmap_offset, <state-3>, <field-id>)
            return result
          }
          catch {
            case ex =>
              LazyVals.setFlag(this, bitmap_offset, <state-0>, <field-id>)
              throw ex
          }
        }
      } else /* if (state == <state-1> || state == <state-2>) */ {
        LazyVals.wait4Notification(this, bitmap_offset, flag, <field-id>)
      }
    }
  }
}
```

The state of the lazy val `<state-i>` is represented with 4 values: 0, 1, 2 and 3. The state 0 represents a non-initialized lazy val. The state 1 represents a lazy val that is currently being initialized by some thread. The state 2 denotes that there are concurrent readers of the lazy val. The state 3 represents a lazy val that has been initialized. `<field-id>` is the id of the lazy val. This id grows with the number of volatile lazy vals defined in the class.

Note on recursive lazy vals

Ideally recursive lazy vals should be flagged as an error. The current behavior for recursive lazy vals is undefined (initialization may result in a deadlock).

Reference

- SIP-20

‹ Chang... Main ... ›