@AndrzejWasowski
Andrzej Wąsowski

Advanced Programming

Laziness and Streams

IT UNIVERSITY OF COPENHAGEN
PROCESS AND SYSTEM MODELS GROUP

Applications of Call-By-Name and Laziness

- Implementing **non-strict-functions**
  - If function **accesses parameters at most once** (simple control-flow like if, or, and, etc.) it can be built with call-by-name only, without memoization; for instance error handling code can be passed as one of parameters
- Handling **large amounts of data**, only accessing necessary parts; especially when it is hard to see which parts need to be accessed/loaded/precomputed
- Improving **memory locality** for processing large data structures in functional style
- Implementing **generators of object/value sequences** elegantly (stream of naturals, stream of prime numbers, stream of messages, stream of random numbers, etc)
- Implementing **internal DSLs** aka **fluent interfaces**
  - Call-by-name allows to hide the lambda expressions, when building **control-flow** constructs in an internal DSL [more next semester]

Example of Call-by-name [1/4]

In Apache Spark's implementation

```scala
/**
 * Return a new RDD by applying a function to all elements of this RDD.
 */
def map[U: ClassTag](f: T => U): RDD[U] = withScope {
  val cleanF = sc.clean(f)
  new MapPartitionsRDD[U, T](this, (context, pid, iter) => iter.map(cleanF))
}
```

How do we implement a function as withScope?

With **call-by-value** the body of the block would always be executed.

But we can use **call-by-name**.

https://github.com/apache/spark/blob/master/core/src/main/scala/org/apache/spark/rdd/RDD.scala, captured 2017-02-25

Example of Call-by-name [2/4]

In Apache Spark's implementation

```
/**
 * Execute a block of code in a scope such that all new RDDs created in this body will
 * be part of the same scope. For more detail, see {{org.apache.spark.rdd.RDDOperationScope}}.
 *
 * Note: Return statements are NOT allowed in the given body.
 */
private[spark] def withScope[U](body: => U): U = RDDOperationScope.withScope[U](sc)(body)
```

The body (U) is passed by-name and then

Forwarded to a similar method in another class, also by-name

No forcing happens

https://github.com/apache/spark/blob/master/core/src/main/scala/org/apache/spark/rdd/RDD.scala, captured 2017-02-25

Example of Call-by-name [3/4]

In RDDOperationScope ...

```
private[spark] def withScope[T](
    sc: SparkContext,
    name: String,
    allowNesting: Boolean,
    ignoreParent: Boolean)(body: => T): T = {
  // Save the old scope to restore it later
  val scopeKey = SparkContext.RDD_SCOPE_KEY
  val noOverrideKey = SparkContext.RDD_SCOPE_NO_OVERRIDE_KEY
  val oldScopeJson = sc.getLocalProperty(scopeKey)
  val oldScope = Option(oldScopeJson).map(RDDOperationScope.fromJson)
  val oldNoOverride = sc.getLocalProperty(noOverrideKey)
  try {
    if (ignoreParent) {
      // Ignore all parent settings and scopes and start afresh with our own root scope
      sc.setLocalProperty(scopeKey, new RDDOperationScope(name).toJson)
    } else if (sc.getLocalProperty(noOverrideKey) == null) {
```

Example of Call-by-name [4/4]

```
} else if (sc.getLocalProperty(noOverrideKey) == null) {
  // Otherwise, set the scope only if the higher level caller allows us to do so
  sc.setLocalProperty(scopeKey, new RDDOperationScope(name, oldScope).toJson)
}
// Optionally disallow the child body to override our scope
if (!allowNesting) {
  sc.setLocalProperty(noOverrideKey, "true")
}
body
```

The body is executed if the control flow reaches the last line above
(in here: no exceptions thrown)
Otherwise the body will never be executed

https://github.com/apache/spark/blob/master/core/src/main/scala/org/apache/spark/rdd/RDDOperationScope.scala, 2017-02-25