Conc-Tree Combiners

Parallel Programming in Scala

Aleksandar Prokopec

Conc Buffers

The `ConcBuffer` appends elements into an array of size k.

When the array gets full, it is stored into a `Chunk` node and added into the Conc-tree.

```
class ConcBuffer[T: ClassTag](val k: Int, private var conc: Conc[T]) {
  private var chunk: Array[T] = new Array(k)
  private var chunkSize: Int = 0
```

Conc Buffers

The += operation in most cases just adds an element to the chunk array:

```
final def +=(elem: T): Unit = {
  if (chunkSize >= k) expand()
  chunk(chunkSize) = elem
  chunkSize += 1
}
```

Occasionally, the chunk array becomes full, and needs to be expanded.

Chunk Nodes

`Chunk` nodes are similar to `Single` nodes, but instead of a single element, they hold an array of elements.

```
class Chunk[T](val array: Array[T], val size: Int) extends Conc[T] {
  def level = 0
}
```

Expanding the Conc Buffer

The expand method inserts the chunk into the Conc-tree, and allocates a new chunk:

```
private def expand() {
  conc = appendLeaf(conc, new Chunk(chunk, chunkSize))
  chunk = new Array(k)
  chunkSize = 0
}
```

Combine Method

The `combine` method is straightforward:

```
final def combine(that: ConcBuffer[T]): ConcBuffer[T] = {
  val combinedConc = this.result <> that.result
  new ConcBuffer(k, combinedConc)
}
```

Above, the `combine` method relies on the `result` method to obtain the Conc-trees from both buffers.

Result Method

The `result` method packs `chunk` array into the tree and returns the resulting tree:

```
def result: Conc[T] = {
  conc = appendLeaf(conc, new Chunk(chunk, chunkSize))
  conc
}
```

Result Method

The `result` method packs `chunk` array into the tree and returns the resulting tree:

```
def result: Conc[T] = {
  conc = appendLeaf(conc, new Chunk(chunk, chunkSize))
  conc
}
```

Summary:

- $O(\log n)$ `combine` concatenation
- fast $O(1)$ `+=` operation
- $O(1)$ `result` operation

Conc Buffer Demo

Demo – run the same benchmark as we did for the `ArrayCombiner`:

```
xs.par.aggregate(new ConcBuffer[String])(_ += _, _ combine _).result
```