```scala

import java.util.concurrent._
import scala.util.DynamicVariable

package object common {

  val forkJoinPool = new ForkJoinPool

  abstract class TaskScheduler {
    def schedule[T](body: => T): ForkJoinTask[T]
    def parallel[A, B](taskA: => A, taskB: => B): (A, B) = {
      val right = task {
        taskB
      }
      val left = taskA
      (left, right.join())
    }
  }

  class DefaultTaskScheduler extends TaskScheduler {
    def schedule[T](body: => T): ForkJoinTask[T] = {
      val t = new RecursiveTask[T] {
        def compute = body
      }
      Thread.currentThread match {
        case wt: ForkJoinWorkerThread =>
          t.fork()
        case _ =>
          forkJoinPool.execute(t)
      }
      t
    }
  }

  val scheduler =
    new DynamicVariable[TaskScheduler](new DefaultTaskScheduler)

  def task[T](body: => T): ForkJoinTask[T] = {
    scheduler.value.schedule(body)
  }

  def parallel[A, B](taskA: => A, taskB: => B): (A, B) = {
    scheduler.value.parallel(taskA, taskB)
  }

  def parallel[A, B, C, D](taskA: => A, taskB: => B, taskC: => C, taskD: => D): (A, B, C, D) = {
    val ta = task { taskA }
    val tb = task { taskB }
    val tc = task { taskC }
    val td = taskD
    (ta.join(), tb.join(), tc.join(), td)
  }

}
```