

# GParS

## Parallelism the Right Way

**Dr Russel Winder**

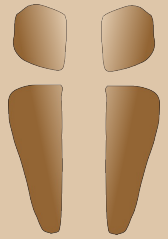
It'z Interactive Ltd

[russel@itzinteractive.com](mailto:russel@itzinteractive.com)

[russel@russel.org.uk](mailto:russel@russel.org.uk)

[@russel\\_winder](https://twitter.com/russel_winder)

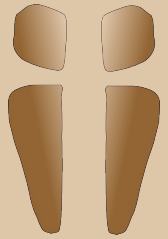




## *Moore's Law*

- More transistors.
- More cores.
- More parallelism.
- Ubiquitous parallelism.

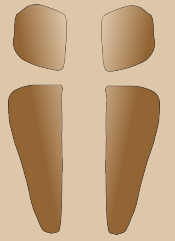




## *Caveat*

- Not all applications require parallelism.
- I/O bound best handled with single threaded event loop.

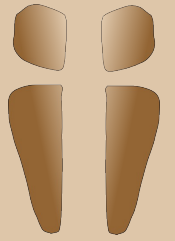




## *Where Parallelism Is Needed*

- Operating systems offer threads as access to the cores.
- Languages offer threads as infrastructure.

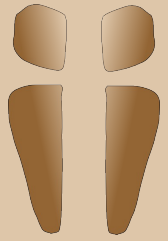




*Hummm...*

- Shared memory multithreading is hard.
- Locks, semaphores, monitors, ...

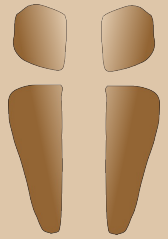




## *Models of Management*

- Actors
- Dataflow
- Communicating Sequential Processes (CSP)
- Data parallelism

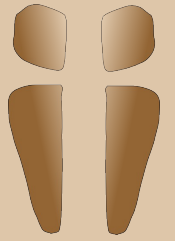




## *GPars*

- A Groovy/Java implemented framework for easy expression of parallelism.
- Offers:
  - Actors
  - Dataflow
  - CSP
  - Data parallelism

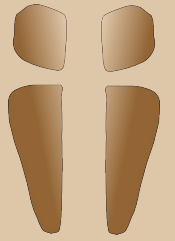




## *Builds on...*

- GParS builds on `java.util.concurrent` and all the JSR166 goodness.



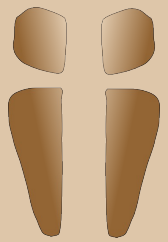


## Usage

- GParS works supremely well in Groovy scripts that manage Groovy and Java classes.
- GParS works very well as a Java API.



@Grab ( 'org.codehaus.gpars:gpars:1.0-SNAPSHOT' )

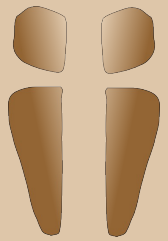


```
import groovyx.gpars.GParsPool
```

```
void execute ( final int numberOfTasks ) {
    GParsPool.withPool {
        final int n = 100000000i
        final double delta = 1.0d / n
        final startTimeNanos = System.nanoTime ( )
        final int sliceSize = n / numberOfTasks
        final items = [ ]
        for ( int i in 0i ..< numberOfTasks ) { items << i }
        final pi = 4.0d * delta * items.collectParallel { taskId ->
            final int start = 1i + taskId * sliceSize
            final int end = ( taskId + 1i ) * sliceSize
            double sum = 0.0d ;
            for ( int i in start .. end ) {
                final double x = ( i - 0.5d ) * delta
                sum += 1.0d / ( 1.0d + x * x )
            }
            sum
        }.sumParallel ( )
        final elapsedTime = ( System.nanoTime ( ) - startTimeNanos ) / 1e9
        println ( '==== Groovy GPars GParsPool pi = ' + pi )
        println ( '==== Groovy GPars GParsPool iteration count = ' + n )
        println ( '==== Groovy GPars GParsPool elapse = ' + elapsedTime )
        println ( '==== Groovy GPars GParsPool processor count = ' + Runtime.getRuntime ( ).availableProcessors ( ) )
        println ( '==== Groovy GPars GParsPool task count = ' + numberOfTasks )
    }
}
```



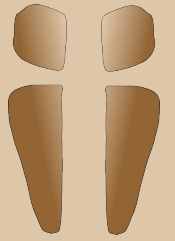
```
@Grab ( 'org.codehaus.gpars:gpars:1.0-SNAPSHOT' )
```



```
import groovyx.gpars.ParallelEnhancer
```

```
void execute ( final int numberOfTasks ) {  
    final int n = 100000000i  
    final double delta = 1.0d / n  
    final startTimeNanos = System.nanoTime ( )  
    final int sliceSize = n / numberOfTasks  
    final items = []  
    for ( int i in 0i ..< numberOfTasks ) { items << i }  
    ParallelEnhancer.enhanceInstance ( items )  
    final pi = 4.0d * delta * items.collectParallel { taskId ->  
        final int start = 1i + taskId * sliceSize  
        final int end = ( taskId + 1i ) * sliceSize  
        double sum = 0.0d ;  
        for ( int i in start .. end ) {  
            final double x = ( i - 0.5d ) * delta  
            sum += 1.0d / ( 1.0d + x * x )  
        }  
        sum  
    }.sumParallel ( )  
    final elapsedTime = ( System.nanoTime ( ) - startTimeNanos ) / 1e9  
    println ( '==== Groovy GPars ParallelEnhancer pi = ' + pi )  
    println ( '==== Groovy GPars ParallelEnhancer iteration count = ' + n )  
    println ( '==== Groovy GPars ParallelEnhancer elapse = ' + elapsedTime )  
    println ( '==== Groovy GPars ParallelEnhancer processor count = ' + Runtime.getRuntime  
        ( ).availableProcessors ( ) )  
    println ( '==== Groovy GPars ParallelEnhancer task count = ' + numberOfTasks )  
}
```



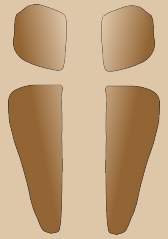


# GParS

Check it out, you know you  
want to.

<http://gpars.codehaus.org/>



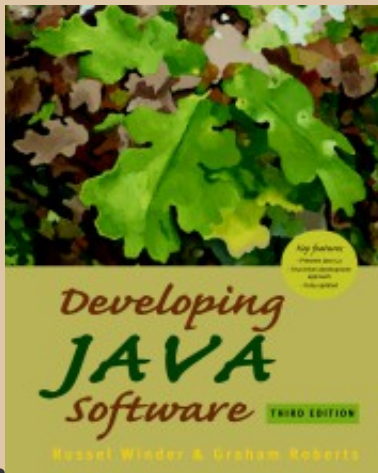
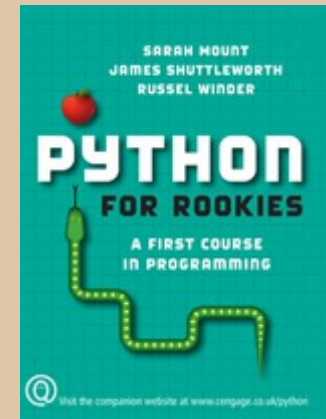


## Mandatory Book Advert

### ***Python for Rookies***

Sarah Mount, James Shuttleworth and  
Russel Winder

*Thomson Learning* Now called *Cengage Learning*.



### ***Developing Java Software Third Edition***

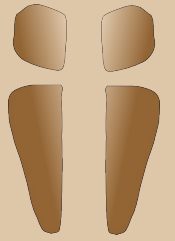
Russel Winder and Graham Roberts

*Wiley*



***Buy these books!***





# GParS

## Parallelism the Right Way

**Dr Russel Winder**

It'z Interactive Ltd

[russel@itzinteractive.com](mailto:russel@itzinteractive.com)

[russel@russel.org.uk](mailto:russel@russel.org.uk)

[@russel\\_winder](https://twitter.com/russel_winder)

