Communicating Sequential Processes

The Whole GPars Team <gpars-developers@googlegroups.com>

Table of Contents

To read this topic in the PDF format, please click here.

CSP

[Communicating Sequential Processes](#) (**CSP**) provides a formal concurrency model consisting of synchronously communicating independent processes.

The model offers deterministic behavior plus it allows developers to combine the processes into composable and reusable components.

Processes, in **GPars** called *Tasks*, are concurrently run independent activities, which communicate by sending data through (typically synchronous) channels.

Here's a concurrent implementation of the Sieve of Eratosthenes

CSP Sample - Sieve of Eratosthenes

```groovy
inal int requestedPrimeNumberCount = 1000
final DataflowQueue initialChannel = new DataflowQueue()
/**
 * Generating candidate numbers
 */
group.task {
    (2..10000).each {
        initialChannel << it
    }
    initialChannel << -1  //poisson
}

/**
 * Chain a new filter for a particular prime number to the end of the Sieve
 * @param inChannel The current end channel to consume
 * @param prime The prime number to divide future prime candidates with
 * @return A new channel ending the whole chain
 */
def filter(inChannel, int prime) {
    def outChannel = new DataflowQueue()

    group.task {
        while (true) {
            def number = inChannel.val
            if (number % prime != 0) {
                outChannel << number
            }
            if (number == -1) break  //handle poisson and stop
        }
    }
    return outChannel
}

/**
 * Consume Sieve output and add additional filters for all found primes
 */
def currentOutput = initialChannel
requestedPrimeNumberCount.times {
    int prime = currentOutput.val
    println "Found: $prime"
    currentOutput = filter(currentOutput, prime)
}
```

GPars tasks represent active computations. Indirect addressing through channels gives you enormous flexibility in how and when you wire tasks together. The concept of *Promises* allows tasks to easily signal events or values to other parts of your program in a thread-safe manner. CSP programms are highly deterministic, which is a very useful quality of concurrent programs.

Tasks can be easily combined with other **GPars** concepts - with *Agents* to ease shared-state management or with *Dataflow Operators* to process streamed data.

For further details, please refer to the Groovy CSP section of this User Guide.