About me

['Passionate programmer',
'Concurrency enthusiast',
'GPars lead',
'Developer/technology evangelist @ JetBrains']

.eachParallel {say it}

http://www.jroller.com/vaclav
http://twitter.com/vaclav_pech
We're all parallel now

Use them or leave them!
# of cores

![Graph showing the increase in the number of cores from today to tomorrow.](image-url)
Parallel Collections

images.eachParallel {it.process()}

documents.sumParallel()

candidates.maxParallel {it.salary}.marry()
Parallel Collections

progLanguages.parallel.filter {it.concurrent}
  .max {it.javaInteroperability}
  .map {it.logo} == Groovy
Languages are either concurrent or obsolete.
Java 5

Asynchronous calculations
Java 7

Asynchronous calculations

Fork/Join
Java 8

Asynchronous calculations
Fork/Join
Parallel collections
Scala

Asynchronous calculations
Fork/Join
Parallel collections
Actors
Clojure

Asynchronous calculations
Fork/Join
Parallel collections
Actors
Agents, Stm
Oz

Asynchronous calculations
Fork/Join
Parallel collections
Actors
Agents, Stm
Dataflow
Google's Go

Asynchronous calculations
Fork/Join
Parallel collections
Actors
Agents, Stm
Dataflow
CSP
Asynchronous calculations
Fork/Join
Parallel collections
Actors
Agents, Stm
Dataflow
CSP
Agenda

✓ Asynchronous calculations
✓ Fork/Join
✓ Parallel collections
✓ Actors
✓ Agents, Stm
✓ Dataflow
✓ CSP
Actors

Processes with a mail-box
Share no data
Communicate by sending messages
Use a thread-pool
Active Objects

@ActiveObject
class MyCounter {
    private int counter = 0

    @ActiveMethod
def incrementBy(int value) {
        println "Received an integer: $value"
        this.counter += value
    }
}
Composing async functions

```java
int hash1 = hash(download('http://www.gpars.org'))
int hash2 = hash(loadFile('/gpars/website/index.html'))
boolean result = compare(hash1, hash2)
println result
```
Composing async functions

@AsyncFun hash = oldHash
@AsyncFun compare = oldCompare
@AsyncFun download = oldDownload
@AsyncFun loadFile = oldLoadFile

def hash1 = hash(download('http://www.gpars.org'))
def hash2 = hash(loadFile('/gpars/website/index.html'))
def result = compare(hash1, hash2)
println result.get()
Composing async functions

```python
@AsyncFun hash = oldHash

@AsyncFun(blocking = true) compare = oldCompare

@AsyncFun download = oldDownload

@AsyncFun loadFile = oldLoadFile

def hash1 = hash(download('http://www.gpars.org'))
def hash2 = hash(loadFile('/gpars/website/index.html'))

boolean result = compare(hash1, hash2)
println result
```
int hash(String text) {...}

Promise<int> hash(Promise<String> | String text)
int hash(String text) {...}

Promise<int> hash(Promise<String> | String text)

compare(
    hash(download()),
    hash(loadFile())
)
int hash(String text) {

Promise<int> hash(Promise<String> | String text) {

1. Return a Promise for the result
2. Wait (non-blocking) for the text param
3. Call the original hash()
4. Bind the result

}
Composing async functions

Combine functions as usual

Parallelism is detected automatically
Dataflow Concurrency

- No race-conditions
- No live-locks
- Deterministic deadlocks
  Completely deterministic programs

BEAUTIFUL code

(Jonas Bonér)
Dataflow Variables / Promises
Dataflows

def df = new Dataflows()

task { df.z = df.x + df.y }
task { df.x = 10 }
task {
    println ”I am task 3”
    df.y = 5
}
assert 15 == df.z
Dataflow Operators

operator(inputs: [headers, bodies, footers],
       outputs: [articles, summaries])

{header, body, footer ->
  def article = buildArticle(header, body, footer)
  bindOutput(0, article)
  bindOutput(1, buildSummary(article))
}

+\rightarrow\ast\rightarrow\langle\rangle
Dataflow Operators

- Speculator
- Downloader (Cache updates)
- Evaluator
- Splitter
- Groovy scanner
- Scala scanner
- Reporter
- Confirm

Dataflow Operators Diagram:

- Url resolver
- Approvals

Diagram showing the flow of data through various operators and processes.
GPars

'coz concurrency is Groovy

Find more at:

http://gpars.codehaus.org
http://www.jroller.com/vaclav
http://twitter.com/vaclav_pech