News

Russell Winder

Version 1.0, 2015-10-01

Table of Contents

TOPICS :

Remoting for GPars

This is a quick overview of **Remoting for GPars** realized during **Google Summer of Code,2014** by *Rafal Slawik*.

The implementation has already become part of the **GPars** 1.3-SNAPSHOT and is available for immediate use.

Behind the scenes, the **Netty** library and the standard serialization mechanism were used as the transportation layer.

Basically, you can use *Dataflows* with any data type, send custom messages to *Actors* to store custom states in *Agents* as long as these objects are seralizable.

The *Dataflow* structures that support remoting: *DataflowVariable*, *DataflowBroadcast*, *DataflowQueue*.

Remote Implementation Requirements

General use of our remoting implementation requires:

- **at host A:** creating a context and publishing a structure (variable, queue, actor, etc.) under some name
- **at host B:** creating a context and retrieval of a structure with that name

The **context** concept is useful for testing. One can have an original instance and a remote proxy within that same VM or other purposes eg. each thread has its own remote proxy. What's important is that a remote proxy has the same interface and therefore can be used as though it was a regular intstance. Let's see an example on how to use remoting for *DataflowVariables*:

- **At host A**:

Create context, start server, create instance & register it

```
def remoteDataflows = RemoteDataflows.create() // creates context
remoteDataflows.startServer HOST PORT // starts server that waits for requests at
HOST:PORT
def variable = new DataflowVariable() // creates variable instance
remoteDataflows.publish variable "my-first-variable" // registers it within the context
under given name
```

- **At host B:**

Retrieves a promise of variable with given name

```
def remoteDataflows = RemoteDataflows.create() // creates context
def remoteVariablePromise = remoteDataflows.getVariable HOST, PORT, "my-first-variable"
// retrieves promise of variable with given name
def remoteVariable = remoteVariablePromise.get() // extracts remote proxy from promise
```

You can find more examples in our samples package:

```
groovyx.gpars.samples.remote.dataflow.*
```

A Ping-Pong Example

Now, let's take a look at remoting for *Actors* and consider a **Ping-Pong** example like *groovyx.gpars.samples.remote.actor.pingpong*. Let's start with creating an *actor* that responds to every message with **PONG**. Such actor can look like this :

Actor Setup

```
def pongActor = Actors.actor { loop { react { println it reply "PONG" } } }
```

It waits in a loop for messages and when one arrives, it prints it and replies with **PONG**. To be able to access this *actor* from a remote host, it has to be published:

Creates context,starts server for requests at HOST:PORT,registers pongActor within context under name **pong**

```
def remoteActors = RemoteActors.create() // creates context
remoteActors.startServer HOST, PORT // starts server that waits for requests at HOST:PORT
remoteActors.publish pongActor, "pong" // registers pongActor within context under name
"pong"
```

What's left is to retrieve the proxy object to that *actor* at the remote host. It can be done as follows:

Actor Setup

```
def remoteActors = RemoteActors.create() // creates context
def pingActor = Actors.actor {
        def remotePongActor = remoteActors.get HOST, PORT, "pong" get() // gets remote
proxy to actor name "pong" at HOST:PORT
        remotePongActor << "PING" // sends message to it
        react {
            println it // prints reply from remote actor
        }
}
```

An extended example can be found in *groovyx.gpars.samples.remote.actor.pingpong*.

More examples of remoting for *Actors* are available in *groovyx.gpars.samples.remote.actor.**. An example of remotes for *Agents* is available in *groovyx.gpars.samples.remote.agent.*

In the future, we can introduce the multiplexing of connections between hosts (currently each **get** will open a new connection) and some form of discovery mechanism (to avoid using explicit HOST:PORT).

Posts

The GA release of **GPars 1.1.0** has just been published and is ready for you to grab. It brings gradual improvements in *dataflow* logic as well as a few other domains. Some highlights:

- *LazyDataflowVariable* added to allow for lazy asynchronous values
- *Timeout for Selects*
- Added a *Promise*-based **API** for value selection through the *Select* class
- Enabled listening for bind errors on *DataflowVariables*
- Minor **API** improvement affecting *Promise* and *DataflowReadChannel*
- Protecting an *agent*'s blocking methods from being called from within commands
- Updated to the latest 0.7.0 GA version of **Multiverse**
- Migrated to **Groovy 2.0**
- Used **@CompileStatic** where appropriate
- A few bug fixes

You can download **GPars 1.1.0** directly or grab it from the maven repo.

Have a lot of fun trying out **GPars 1.1.0** !

A first release candidate for **GPars 1.1.0** has been made available. The final `1.1.0 GA` should be expected in a few days. The 1.1.0 release is a gradual improvement of 1.0.0 with additions mostly in the *Dataflow* domain. Starting with 1.1, **GPars** requires **Groovy 2.0** or higher. Check out the most noteworthy new capabilities:

Dataflow

- *LazyDataflowVariable* added to allow for lazy asynchronous values
- *Timeout for Selects*
- Added a *Promise*-based **API** for value selection through the *Select* class
- Enabled listening for bind errors on *DataflowVariables*
- Minor **API** improvement affecting *Promise* and *DataflowReadChannel*

Agent

- Protecting an *agent* blocking methods from being called from within commands

Software Transactional Memory

- Updated to the latest `0.7.0 GA` version of **Multiverse**

Other

- Migrated to **Groovy 2.0**
- Used **@CompileStatic** where appropriate

Get **GPars** 1.1.0-rc1, take it for a spin and please report all issues so we can fix them before GA.

GPars 1.0 Arrived, *Vaclav Pech* posted on Dec 19, 2012

I'm happy to announce that after four years of development **GPars**, the *Groovy Concurrency Library*, has just reached its 1.0 mark. A fresh and crispy **GPars 1.0.0** is now ready for you to grab or download and use on your projects. Also, the up-coming **Groovy** releases will bundle **GPars 1.0**.

Compared to the previous release, 1.0 brings several performance enhancements, considerable **API** updates, polished documentation and numerous functionality improvements, mostly in the *dataflow* area. Please, check out the `What's new` section of the user guide for the details.

I would like to use this opportunity to thank all the **Groovy** people, who have over time contributed in one way or another to the success of **GPars**. It is my honour to be part of such a helpful and encouraging community. In particular, I would like to thank my colleague **GPars** commiters, namely *Paul King*, *Dierk Koenig*, *Alex Tkatchman* and *Russel Winder*, who we've been consistently pushing the project forward and without whom it would hardly ever get this far. I also greatly appreciate the support we received from *Guillaume Laforge*, the **Groovy** supreme commander. Thank you all gentlemen!

Groovy concurrency times ahead!

- *Vaclav*

The First Release Candidate of 1.0 Is Available, *Vaclav Pech* posted on Dec 11, 2012

We are almost there. The 1.0 release is just round the corner. To ensure that 1.0 meets your quality expectations we first prepared a release candidate to test the waters.

To take **GPars** for a test ride, please download or grab it at the usual places, check out the release notes and let us know if something is missing.

- *Vaclav*

Beta 3 is out, *Vaclav Pech* posted on Sep 10, 2012

GPars-1.0-beta-3 has been made available for you to try out.

Apart from the usual doze of features and fixes, including speed-up for some operations on parallel collections or lifecycle events for *dataflow* operators, there is one major change compared to beta-2 worth pointing out explicitly:

> **GPars** no longer depends on the **extra166y** artifact

The parallel array library by *Doug Lea* has been integrated into **GPars**. The **jsr166y** (*Fork/Join*) jar still remains in the dependency list until we migrate **GPars** to jdk7 BUT **GPars** no longer depends on the **extra166y** artifact.

Grab **GPars-1.0-beta-3** and have a lot of fun with the new release.

GPars 1.0 beta-1 ready for a test ride, *Vaclav Pech* posted on Dec 30, 2011

Our first step towards the 1.0 release has been achieved. The *beta-1* release is now available for you to grab or download. Have fun and if you feel something needs our attention, please let us know.

- *The **GPars** team*

Parallel Game of Life, *Vaclav Pech* posted on Sep 01, 2011

I'd like to direct you to my recent blog post detailing the use of *Dataflow* operators. It uses the popular Game of Life coding excercise to illustrate the principles of the *dataflow* concept. Check it out at my personal blog.

GPars turns 0.12 today, *Vaclav Pech* posted on Jun 02, 2011

We have some great news to all the parallel souls out there - **GPars 0.12** has just hit the shelves. The new version comes with lots of big and small improvements, out of which these are the most notable ones:

- Composable asynchronous functions
- The newest version of *Doug Lea's* super cool *Fork/Join* framework (aka **jsr-166y**)
- *Active Objects*
- Initial stub at *Software Transactional Memory* support using **Multiverse**

Check out the full release notes for more details.

To quickly get up-to-speed with **GPars**, check out our updated User Guide, which is now also available in pdf format.

- *Your **GPars** team*

JVM Concurrency and *Actors* with GPars, *Vaclav Pech* posted on Apr 26, 2011.

Dr.Dobb's has just published my overview article on *actors* in **GPars**. You may check it out at Drdobbs.com/High-performance Computing

- *Vaclav*