User Guide

Russell Winder

Version 1.0, 2015-10-01

Table of Contents

An Introduction 1
Enter **GPars** 2
Credits 3

An Introduction

The world of mainstream computing is changing rapidly these days. If you open the case and look under the covers of your computer, you'll most likely see a dual-core processor there, or a quad-core one, if you have a high-end computer. We all now run our software on multi-processor systems.

Why do people still create single-threaded code ?

The code we write today and tomorrow will probably never run on a single processor system: parallel hardware has become standard. Not so with the software though, at least not yet. People still create single-threaded code, even though it will not be able to leverage the full power of current and future hardware.

The code we write today will probably never run on a single processor system !

Some developers experiment with low-level concurrency primitives, like threads, and locks or synchronized blocks. However, it has become obvious that the shared-memory multi-threading approach used at the application level causes more trouble than it solves. Low-level concurrency handling is usually hard to get right, and it's not much fun either.

With such a radical change in hardware, software inevitably has to change dramatically too. Higher-levelS OF concurrency and parallelism concepts like *map/reduce*, *fork/join*, *actors* and *dataflow* provide natural abstractions for different types of problem domains while leveraging the multi-core hardware.

Enter GPars

Meet **GPars**, an open-source concurrency and parallelism library for **Java** and **Groovy** that gives you a number of high-level abstractions for writing concurrent and parallel code in **Groovy** (*map/reduce, fork/join, asynchronous closures, actors, agents, dataflow concurrency* and other concepts), which can make your **Java** and **Groovy** code concurrent and/or parallel with little effort.

With **GPars** your **Java** and/or **Groovy** code can easily utilize all the available processors on the target system. You can run multiple calculations at the same time, request network resources in parallel, safely solve hierarchical divide-and-conquer problems, perform functional style map/reduce or data parallel collection processing or build your applications around the actor or dataflow model.

Apache

The **GPars** project is open sourced under the Apache 2 License.

If you're working on a commercial, open-source, educational or any other type of software project in **Groovy**, download the binaries or integrate them from the **Maven** repository and get going. The door to writing highly concurrent and/or parallel **Java** and **Groovy** code is wide open. [blue]Enjoy!

Credits

This project could not have reached the point where it stands currently without all the great help and contributionS from many individuals, who have devoted their time, energy and expertise to make **GPars** a solid product. First, it's the people in the core team who should be mentioned:

- Václav Pech
- Dierk Koenig
- Alex Tkachman
- Russel Winder
- Paul King
- Jon Kerridge
- Rafał Sławik

Over time, many other people have contributed their ideas, provided useful feedback or helped **GPars** in one way or another. There are many people in this group, too many to name them all, but let's list at least the most active:

- Hamlet d'Arcy
- Hans Dockter
- Guillaume Laforge
- Robert Fischer
- Johannes Link
- Graeme Rocher
- Alex Miller
- Jeff Gortatowsky
- Jiří Kropáček

Many thanks to everyone!