

Home

The GPar framework offers Java developers intuitive and safe ways to handle Java or Groovy tasks concurrently. Leveraging the enormous flexibility of the [Groovy programming language](#) and building on proven Java technologies, we aim to make concurrent programming for multi-core hardware intuitive, robust and enjoyable.

GPar - 'coz concurrency is Groovy

The traditional thread-based concurrency model built into Java doesn't match well with the natural human sense for parallelism. While this was not a problem at times, when the level of parallelism in software was low and concurrency offered only limited benefits compared to sequential code, nowadays, with the number of cores on a single main-stream chip doubling almost every year, sequential code quickly loses ground and fails to compete in performance and hardware utilization with concurrent code.

Inevitably, for concurrent programming to be effective, the mental models of concurrent systems interactions that people create in their heads have to respect the nature of human brains more than the wires on the chips. Luckily, such abstractions have been around for several decades, used at universities, in telephone switches, the super-computing industry and some other inherently concurrent domains. The current challenge for GPar is to bring these abstractions up to the mainstream software developers to help us solve our practical daily issues.

The framework provides straightforward Java or Groovy-based APIs to declare, which parts of the code should be performed in parallel. Collections can have their elements processed concurrently, closures can be turned into composable asynchronous functions and run in the background on your behalf, mutable data can be protected by agents or software transactional memory.

For the common scenario that one or multiple results are calculated concurrently but need to be processed as soon as they are available, GPar makes it a breeze to correctly model this with [Dataflow](#). Dataflow variables and channels gives you a handy abstraction of single-assignment multiple-read data elements, while dataflow operators let you build efficient concurrent data-processing networks.

The concept of [Actors](#) as an approach to organizing concurrent activities has recently gained new popularity (thanks to the Scala, Erlang, and other programming languages). GPar implements this concept for Java and Groovy developers. With actors support you can quickly create several independent Actors, which consume messages passed to them and communicate with other actors by sending them messages. You then build your solution by combining these actors into a communication network.

Please refer to the [User Guide](#) for a more extensive coverage of these topics or head over to the [Demos](#).

Let the fun begin!

Main Areas

- [Concurrent collection processing](#)
- [Composable asynchronous functions](#)
- [Fork/Join abstraction](#)
- [Actor](#) programming model
- [Dataflow concurrency constructs](#)
- [CSP](#)
- [Agent](#) - an thread-safe reference to mutable state
- [STM](#) (Software Transactional Memory)

Project's main priorities

- Clever and clean design
- Elegant Java and Groovy APIs

- Flexibility through meta-programming
- Application-level solutions that scale with number of cores

Fast Track

If you want to start experimenting with GParS right away, use our Fast Track to get up and running within minutes.

- [Groovy Fast Track](#)
- [Java Fast Track](#)
- [Grails and Griffon Fast Track](#)

What people say about GParS

Check out the [User Voices](#) to hear the opinions of people walking here before you.

Licensing

GParS is distributed under the open-source [Apache 2 License](#). By using GParS you accept fully the terms stated in the license. For full details, please see the [Apache 2 License](#) document.