The GPars Guide to Core Features

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

Version 1.2.1, 2016-01-04

Table of Contents

Overview

The Actor Mechanism

Actors are independent isolated active objects, which mutually share no data and communicate solely by messages passing. The code body of each actor is executed by a random thread from a thread pool and so actors can proceed concurrently and independently.

The Agent Mechanism

In several programing languages, we find a concept of **Agents**, who behave like actors, taking coded functions as messages. After reception, the received function is run against the internal state of the **Agent** and the return value of the function is considered to be the new internal state of the **Agent**.

The Asynchronous_Functions Mechanism

GPars provides several ways to run tasks in the background asynchronously.

The CSP Mechanism

The **CSP** (`Communicating Sequential Processes`) concurrency concept is a message-passing model with synchronous rendezvous-type communication. It's valued for its high level of determinism and the ability to compose parallel processes.

The Dataflow Mechanism

Dataflow Concurrency offers an alternative concurrency model, which is inherently safe and robust. It puts an emphasis on the data and their flow though your processes

---

The ForkJoin Mechanism

Our code frequently needs to manipulate collections. Lists, arrays, sets, maps, iterators, strings and lot of other data types can be viewed as collections of items. The common pattern to process collections is to take elements sequentially, one-by-one, and make an action for each of the items in row.

Thanks to **Groovy**, a number of methods are currently supported for `Parallel Collection`

`Processing`.

The GParsPool Mechanism

On multi-core systems, we can benefit from having some tasks run asynchronously in the background, and so off-load our main thread of execution.