The Journal of Open Source Software

ipc: An R Package for Inter-process Communication

Ian E. Fellows[1]

1 Fellows Statistics, http://www.fellstat.com

DOI: 10.21105/joss.00988

Software

- Review
- Repository
- Archive

Submitted: 06 September 2018
Published: 06 November 2018

Summary

Asynchronous processing is critical for performing a wide array of tasks, from high performance computing to web services. Communication between these disparate asynchronous processes is often required. The `parallel` package, which is part of the R computing environment (R Core Team, 2018) allows the user to send computations to be executed by idle worker processes, and the `drake` package (Landau, 2018a) is a general purpose workflow manager, where tasks can be executed in parallel. Several packages have been written to handle the passing of text or binary data between processes (e.g. (Landau, 2018b), (Csárdi, 2017), and (Armstrong & Ooms, 2017)). Currently the statistical computing language R provides no built in features to handle interprocess communication between R processes while they are performing computations. `ipc` allows you to easily pass R objects between processes along with an associated signal, and have handler functions automatically execute them in the receiving process. In `ipc`, communication is backed either through the file system or a database connection.

For example, one might signal for the execution of an expression in one thread to set a variable `a`.

```
q <- queue()
q$producer$fireEval(a <- 1)
```

Then in another thread, this signal can be processed, resulting in the value `a` being set to 1 in the receiving thread after calling the `consume` method.

```
q$consumer$consume()
```

This package can be applied to high performance computing environments, easily allowing parallel worker processes to communicate partial results or progress to the main thread. This functionality can be used to supporting interactive diagnostics and progress for parallelized functions. Particular focus is paid to the use case of supporting asynchronous web based user interfaces ((Chang, Cheng, Allaire, Xie, & McPherson, 2018)).

Use in Shiny

When building a Shiny interface that involves long running processes, there are two major considerations that need to be addressed:

1. The server should not be blocked by the computation.
2. The user should be able to communicate to the computation (e.g. stop it), and the computation should be able to communicate with the user (e.g. display progress).

The first item is critical for scalable application development, because if one user blocks the server, then the UI is unresponsive for all other users of the application. Shiny's new async infrastructure addresses this by allowing long running operations to be preformed in worker processes.

The second item is critical for developing responsive applications. If a user interface is locked while a computation is proceeding, the user is in the frustrating position of not being able to cancel it and can't know how long the computation will take before they'll be able to get their interface back. While async solves the first consideration, inter-process communication is needed to handle the second.

The ipc package provides an easy way to signal between the main thread of your Shiny application and any workers that you create. Intermediary results of the computation can be assigned to reactive values, allowing the user to monitor the internal progress of the operation. In fact, you can send any arbitrary computation from one thread to another using the `fireEval` and `fireCall` methods. Full details are outlined in the package vignette.

Acknowledgements

We acknowledge the Center for Disease Control, and in particular Ray Shiraishi for their support in the development of the `ipc` package.

References

Armstrong, W., & Ooms, J. (2017). *Rzmq: R bindings for 'zeromq'*. Retrieved from https://CRAN.R-project.org/package=rzmq

Chang, W., Cheng, J., Allaire, J., Xie, Y., & McPherson, J. (2018). *Shiny: Web application framework for r*. Retrieved from https://CRAN.R-project.org/package=shiny

Csárdi, G. (2017). *Liteq: Lightweight portable message queue using 'sqlite'*. Retrieved from https://CRAN.R-project.org/package=liteq

Landau, W. M. (2018a). *Drake: A pipeline toolkit for reproducible computation at scale*. Retrieved from https://CRAN.R-project.org/package=drake

Landau, W. M. (2018b). *Txtq: A small message queue for parallel processes*. Retrieved from https://CRAN.R-project.org/package=txtq

R Core Team. (2018). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from https://www.R-project.org/