Parallel recipes: towards a common coordination language for scientific workflow management systems

Yves Vandriessche

Software Languages Lab, Vrije Universiteit Brussel, Brussels, Belgium. Email: yvdriess@vub.ac.be
Exascience Life Lab, Imec, Leuven, Belgium

Project Website: http://www.exascience.com/
Source Code: http://github.com/yvdriess/precipes/
License: BSD three-clause license

It is evident from examining best practices pipeline implementations that there is significant inherent complexity: coordinating the execution of a large number of heterogeneous applications, informally specified data formats, heavy dependence on execution context (e.g. environment variables, specific Java VM version), etc. Glue languages such as Perl or bash are great tools for handling this type of complexity. However, the traditional glue languages were not made to deal with challenges faced in today's distributed computing environments. Using a simple software lines of code metric on standard sequencing pipelines has shown us an increase from roughly 250sLoC to 2500sLoC, an order of magnitude increase. Scientific workflow management systems have stepped in and have greatly simplifying the task of gluing together analysis jobs into a larger reusable whole. However, as a recent report on parallelisation in such systems shows[1]: *"In closing, while substantial progress has been made in parallel scientific workflow enactment, the field of solutions is still heterogeneous and leaves room for improvement."*

We report on the an initial experiment where we tackle the coordination challenges inherent in the large scale distribution and parallelisation of bioinformatics workflows. We work towards a common coordination language with which workflow management systems can coordinate with the various actors involved in the execution: analysis tools, operating system, other workflow management systems, schedulers, compute infrastructures, etc. The rationale behind such common coordination language follows the basic claims put forward by Gelernter and Carriero[2] of Linda[3] fame: that it is possible to treat coordination as *orthogonal* to computation and that it is possible to define coordination in a *general* way such that it applies to every asynchronous part of the system. Concretely, we implemented a simplified workflow description language called *parallel recipes* or *precipes.* As a case study we implemented the standard exome sequencing pipeline in *precipes*. Important is how this workflow's parallel execution is implemented using the Concurrent Collections (CnC) coordination language model[4]. We use the Intel CnC++ implementation (https://icnc.github.io/) as an execution platform and execute transparently on top of a workstation, cluster or Amazon EC2 nodes. We demonstrate automatic exploitation of in-node as well as across-node parallelisation with predictable linear scaling.

---

[1] M. Bux and U. Leser, "Parallelization in Scientific Workflow Management Systems," CORD Conference Proceedings, 2013.

[2] D. Gelernter and N. Carriero, "Coordination languages and their significance," Commun. ACM, vol. 35, no. 2, p. 96, 1992.

[3] D. Gelernter, "Generative communication in Linda," ACM Transactions on Programming Languages and Systems (TOPLAS), vol. 7, no. 1, pp. 80–112, 1985.

[4] Z. Budimlić, M. Burke, V. Cavé, K. Knobe, G. Lowney, R. Newton, J. Palsberg, D. Peixotto, V. Sarkar, and F. Schlimbach, "Concurrent collections," Scientific Computing, vol. 18, no. 3, pp. 203–217, 2010.