Distributed Computing using the multiR Package

Daniel J Grose[1]

[1]Centre for e-Science , Lancaster University , United Kingdom

March 31, 2008

Abstract

There exist a large number of computationally intensive statistical procedures that can be implemented in a manner that is suitable for evaluation using a parallel computing environment. Within this number there exists a class of procedures, often described as "course grained parallel" or "embarrassingly parallel". The defining characteristic of these procedures is that they can be reduced to a number of sub-procedures that are independent of each other and require little or no inter-procedure communication i.e. they can be executed concurrently. Initially, it might be thought that this class is too small to warrant significant attention, however this is far from being the case. For example, methodologies such as bootstrapping, cross-validation, many types of Markov Processes (including MCMC), and certain optimisation and search algorithms are of this type. Importantly, the increase in availability of High Throughput Computing (HTC) environments, consisting of large numbers of interconnected computers, has made employing such procedures particularly attractive, leading to a significant increase in the amount of research being undertaken using HTC, notably in the areas of biochemistry, genetics, pharmaceuticals, economics, financial modelling and the social sciences.

A High Throughput Computing environment provides a means for processing a large number of independent (non-interacting) tasks simultaneously. In the simplest case, the HTC environment may employ only a single multi-processor system. At the other extreme, the HTC environment might comprise a large number of systems with different operating systems and hardware located across a number of different institutions and administrative domains. When this is the case the environment may be said to provide High Throughput Distributed Computing (HTDC).

HTC on a single multiprocessor system is relatively straightforward. Typically the user has an account on the system (can be identified to the system by a user name and password) and can submit the tasks for processing by using the software tools available on that system. Higher level means of submitting tasks exist, such as the **snow** package for R [1]. This package allows functions defined in R or installed R packages to be invoked multiple times with varying argument signatures and executed on a number of processors simultaneously. In [1] it is noted that the functionality offered by **snow** could be extended to use the GRID, which by its nature provides a HTDC environment. Some of these extensions have been addressed within the GridR system [3], which is similar in principle to **snow** but provides some of the technical requirements necessary

for using GRID based resources which it achieves by employing the COG toolkit [2].

However, there are a number of important considerations which arise when using generalised HTDC (GRID based or otherwise) not all of which have been encapsulated in either **snow** or GridR. These considerations are

1. A client session may terminate before all tasks have been processed. For instance, the results of the completed tasks may need to be collected in a future client session, possibly from a different system.
2. The systems employed to process the tasks may be multi-fold and reside in different administrative domains, thus it is not practical for a client to have to obtain and manage accounts on all (potentially hundreds or even thousands) of these systems. Consequently, a single means of identifying the client is required.
3. The client system must employ a secure channel for communication.
4. Host systems are typically shared by many clients and have scheduling systems to allocate resources, thus the execution time and the order in which tasks are processed may vary.
5. Individual tasks may fail to complete (this is quite common on certain systems, such as Condor pools).
6. The client interface should be independent of the nature of the distributed systems used for undertaking the computation.

All of these considerations have been well studied in many varied contexts and the design pattern most associated with realising the above design criteria is a three-tier client server employing the public key infrastructure for authentication and security. The technologies required for implementing such an architecture to host a HTDC service for R are readily available and have been used to develop servers which expose an interface for use within a client R session. The **multiR** package contains an implementation of a client interface for use in R which is similar in many respects to that of **snow** and GridR in that it extends the **apply** family of functions (available in the base package) for submitting multiple function invocations in a distributed environment. **multiR** also provides the functionality required to generate certificate based proxy credentials, manage active jobs and harvest results when they become available. Importantly, the interface provided by **multiR** is independent of the many different types of hardware and software systems employed within a HTDC environment and requires no additional software components (Globus, CoG and so on) to be installed before it can be used.

The full presentation of this work demonstrates how **multiR** is installed and used using several example applications which include bootstrapping, calculating multivariate expectation values and function optimisation. For each of the examples the benefits of using **multiR** are examined, with particular reference to the reduced time required to compute them.

References

[1] A. J. Rossini, Luke Tierney, and Na Li. Simple parallel statistical computing in R. *Journal of computational and Graphical Statistics*, 16(2):399–420, June 2007.

[2] Gregor von Laszewski and Mike Hategan. Workflow concepts of the Java CoG kit. *Journal of Grid Computing*, 3(3–4):239–258, 2005.

[3] Denis Wegener, Thierry Senstag, Stelios Sfakianakis, Stefan Ruping, and Anthony Assi. GridR: an R-based grid-enabled tool for data analysis in ACGT clinico-genomic trials. In *Third IEEE International Conference on e-Science and Grid Computing (e-Science 2007), Bangalore, India.*, pages 205–212. IEEE, 2007.