Use the Make Utility for the Maintenance of Complex Bioinformatics Pipelines

Justin Wilson (wilsonjr@umich.edu), Manhong Dai, Stanley Watson and Fan Meng
Psychiatry Department and Molecular and Behavioral Neuroscience Institute, U of Michigan, US

Complex bioinformtics pipelines usually integrate data from multiple sources using different data processing methods. For example, the SNP Function Portal developed in our group integrates SNP function annotation data from several dozen public data sources as well as data derived from multiple in-house data annotation procedures. Updating the SNP Function Portal is a complex and time-consuming process. Existing tools like cron, wget, shell scripts and custom parsers are adequate for projects requiring a small number of datasets with simple dependencies. In order to efficiently manage complex data download, processing and integration procedures involved in our SNP Function Portal data integration project, we decided to use the classic Make utility to augment existing maintenance techniques with dependency tracking. We also devised methods that allow the checking of table dependency and the management of jobs on computer cluster with the Make utility.

As described in the make utility documentation, `make' is not limited to programs. One can use it to describe any task where some files must be updated automatically from others whenever the others change. A Makefile is a dependency tree in which a target often depends on several prerequisite files. Only when a target file is older than any of its prerequisite files, the action that generates the target file would be triggered. This is very useful for maintaining a complicated bioinformatics pipeline involving many data sources and processing methods. Make can figure out which targets need to be updated based on which source files have changed, and remake them all and only, in a proper order automatically.
Make enable easier collaboration. Each collaborator only needs to implement his/her own dependency tree without knowing any details of others. The interface between each other is just a few targets. For example, in our SNP function annotation integration project, one member of our team is responsible for writing parsers to extract information from dbSNP and export results in predefined format, such as Oracle tables and a Fasta file containing SNP sequences from different organisms. Other members of the team or outside collaborators only need to put the results needed in his/her prerequisite list and there is no need to worry about the details of dbSNP extraction process.

Make has many helpful options. A good example is '-j', which specifies the number of jobs to run simultaneously. User can always adjust this number based on computer's work load, the target making order is always proper.

Since bioinformatics data integration often involves multiple tables with complex dependencies, it is essential to have the capability to check table dependency for a pipeline management solution. Although the Make utility cannot check table's dependency in database, we get around this shortcoming by creating a tag file for each database table to enable Make-based dependency tracking.

Another challenge is there is often a need to submit large-scale data processing jobs to a computer cluster. We also figured out a way to let Make communicate with cluster automatically. We also use command like "while qstat `cat $<`; do sleep 60; done" to make local processing that need utilize results from a prerequisite job running on a cluster will not start till the cluster job is done.

In summary, we believe the classic Make utility provides significant advantages for the management of complex bioinformatics work flows. Combined with methods for checking table dependency and computer cluster job management, Makefiles greatly enhances existing solutions in terms of flexibility, power, manageability, and automation. Sample codes for dbSNP data extraction, Oracle table dependency checking and cluster job management using the Make utility can be downloaded from the SNP Function Portal Data Integration Project website (http://brainarray.mbni.med.umich.edu/Brainarray/Database/SearchSNP/integrate.aspx). These codes use the GNU General Public License.