BioLib: Sharing high performance code between BioPerl, BioPython, BioRuby, R/Bioconductor and BioJAVA

by Pjotr Prins

Dept. of Nematology, Wageningen University, The Netherlands (pjotr.prins@wur.nl)

website: `http://biolib.open-bio.org/`
git repository: `http://github.com/pjotrp/biolib/`
LICENSE: BioLib defaults to the BSD license, embedded libraries may override

BioLib provides the infrastructure for mapping existing and novel C/C++ libraries against the major high level languages using a combination of SWIG and cmake. This allows writing code once and sharing it with all bioinformaticians - who, as it happens, are a diverse lot with diverse computing platforms and programming language preferences.

Bioinformatics is facing some major challenges handling IO data from microarrays, sequencers etc., where every innovation comes with new data formats and data size increases rapidly. Writing solutions in every language domain usually entails duplication of effort. At the same time developers are a scarce resource in every Bio* project. In practice this often means missing or incomplete functionality. For example microarray support is only satisfactory in R/Bioconductor, but lacking in all other domains. By mapping libraries using SWIG we can support all languages in one strike, thereby concentrating the effort of supporting IO in one place.

BioLib also provides an alternative to webservices. Webservices are often used to cross language barriers using a 'slow' network interface. BioLib provides an alternative when fast low overhead communication is desired, like for high throughput, high performance computing.

BioLib has mapped libraries for Affymetrix microarray IO (the Affyio implementation by Ben Bolstad) and the Staden IO lib (James Bonfield) for 454-sequencer trace files, amongst others, and made these available for the major languages on the major computing platforms (thanks to CMake). Current efforts include mapping of R/QTL analysis libraries and libsequence and/or BioC++ libraries. Also a solution is worked on for documenting API of mapped libraries automatically for all supported languages - something not provided by SWIG. Finally BioLib is attracting new code development with a focus on high performance computing including parallelization and GPU optimizations.

In this talk I will discuss the ins and outs of library mapping with SWIG, what it means, and how BioLib takes the pain away of deployment on different platforms, even with external dependencies.