ScalaParBiBit

Reference Manual

Authors:

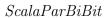
Basilio B. Fraguela Rodríguez Jorge González-Domínguez Diego Andrade

Institution:

Grupo de Arquitectura de Computadores Departamento de Ingeniería de Computadores Universidade da Coruña, Spain

Date:

July 23, 2019







Contents

1	Introduction	2
	Installation 2.1 Using a Makefile	
	2.2 Using CMake	ა 4

July / 2019





1 Introduction

ScalaParBiBit is a parallel tool to accelerate the search of biclusters on binary datasets, especially useful for gene expression data. This tool receives as input a file with ARFF extension that contais the binary values of m attributes and n samples and returns a file with the biclustering information. ScalaParBiBit is implemented by the Grupo de Arquitectura de Computadores at the Universidade da Coruña using C++11 and MPI in order to exploit the parallel capabilities of multicore clusters. It is distributed as free software and publicly available under the GPLv3 license at:

https://github.com/fraguela/ScalaParBiBit

The corresponding license file is shipped with the software but can also be accessed via:

http://www.gnu.org/licenses/gpl-3.0.en.html

If you want to reuse the code, please ensure compliance to the aforementioned license and a proper attribution/citation of the original work/authors.

July / 2019 2





2 Installation

To complete the installation of *ScalaParBiBit* you can follow either a procedure based on Makefiles or one based on the portable *CMake* tool. We explain both in turn

2.1 Using a Makefile

- 1. Untar the archive and move into the ScalaParBiBit directory.
- 2. Update the file *Makefile* of the root directory in order to indicate the correct path and libraries for the MPI compiler installed in your system.
- 3. Type make to build ScalaParBiBit.

2.2 Using CMake

- 1. Untar the archive
- 2. Make a directory to build the tool, for example with mkdir build.
- 3. Move in the directory created to build ScalaParBiBit.
- 4. Run ccmake providing as argument the directory where *ScalaParBiBit* sources are found.
- 5. Press c to perform the configuration. Here you'll mainly see, and have the opportunity to change, the degree of optimization and MPI compiler that will be used to build the tool.
- 6. Change any configuration item if wished.
- 7. Press c again to make the final configuration.
- 8. Press g to generate the building files that are native to your system and exit ccmake.
- 9. Build *ScalaParBiBit*. For example, in a UNIX system the building system will be based on make, and thus it will be only necessary to run make.

July / 2019 3





3 Execution

ScalaParBiBit can be executed with any MPI running command (e.g., mpirun, mpiexec). The arguments for the program are (some of them compulsory and some of them optional):

- -i. Compulsory. String with the path to the ARFF input file.
- -mv. Compulsory. Maximum level to create the discretized matrix when input is not binary. ParBiBit will evaluate iteratively the number of biclusters from this level to 1. Level l indicates that all values higher than l will be 1 in the discretized matrix. In case of a binary input indicate 1 for this parameter.
- -mr. Compulsory. Minimum number of rows in the biclusters. It corresponds to the number of genes in gene expression data.
- -mc. Compulsory. Minimum number of columns in the biclusters. It corresponds to the number of samples in gene expression data.
- -o. Compulsory. String with the path to the base of the output files. There will be one output file per level with the suffix $_{-}l.txt$.
- -r. Optional. String with the path to a file with the names that will be used in the output for the elements represented by the rows (genes in gene expression data). By default row i is called G.i.
- -c. Optional. String with the path to a file with the names that will be used in the
 output for the elements represented by the columns (samples in gene expression data).
 By default column j is called S_j.
- -C. Optional. Number of initialized biclusters in each chunk sent for completion to the slave processes from the master.
- -t. Optional. Integer with the number of threads per MPI process. The best configuration process/threads depends on the characteristics of the machine. Default is 1.

For instance, the following command finds the biclusters with a minimum of 3 rows and 2 columns of the values stored in example.arff for all levels l between 4 and 1, using two MPI processes that launch three threads each (six total threads). The biclusters are written into one file per level, with the format myOut_1.txt. example.arff is available with the distribution of the tool.

mpirun -np 2 ScalaParBibit -i example.arff -o myOut -t 3 -mr 3 -mc 2 -mv 4

July / 2019 4