ACHRcu Documentation

Release 0.1

Marouen Ben Guebila

Apr 07, 2020

Contents

ACHR.cu is a CUDA implementation of the sampling algorithm Artificially Centered Hit-and-Run (ACHR) for the analysis of metabolic models. Metabolic models are mathematical representations of biological organisms forumlated as linear programs. Popular metabolic modeling tools like Flux Balance Analysis (FBA) assume an objective function that the organism optimizes for. When it is not obvious which objective function the system optimizes for, unbiased analysis like sampling is a tool of choice. Sampling is an MCMC method that explores the solution space or the set of possible phenotypes under the linear constraints.

But as metabolic models of biological systems become more complex, the sampling of the solution space of a metabolic model becomes unfeasible due to the large analysis time. In order to address the large analysis time for large metabolic models, I implemented a GP-GPU version of ACHR that reduces the sampling time by at least a factor of 10x for the sampling per se and a factor of 100x for the generation of warmup points which is the preprocessing step. Here you can find tutorials on the installation and analysis of ACHR.cu sampling software.

CHAPTER 1

General approach and parallel construct

Sampling metabolic models is a two-step process:

1. Generation of warmup points. The generation of p warmup points is basically solving the linear program with randomly generated coefficient vector *c* twice as a maximization problem and a minimization problem. The use of a randomly genrated coefficient vector makes the solution of linear program extremely slow and subject to numerical instability. Particularly in parallel setting, some cores might get the linear programs that require more time to solve while others get the easier ones, which can result in an overall slower analysis time. In a previous work, I addressed a similar question through dynamic load balancing. Briefly, if a worker gets a high computational load then the idle workers can take up some of that load. Using a dynamically load balanced generation of warmup points software, the speed up achieved is at least a 100x.

2. The actual sampling using the warmup points as a starting point. With the warmup points at hand, we can proceed to the actual sampling using a cuda implementation. The architecture uses the modern specs of Nvidia cards to perform dynamic parallelism . In fact, there will be p random starting points at the same time (first level of parallelism) that will each launch n random chains to sample the solution space. This procedure is repeated a number of times taking each time a new starting point and saving the sampled points. In particular each chain will sample the local space close to its starting point, which could improve the convergence of the algorithm and avoid the blocking of the sampling chain in the corners of the flux cone. Additionally, the provided computational power will allow the user to sample a greater number of points which can greatly help the assessement of the uniform representation of the solution space and address the sampling of large metabolic models.

CHAPTER 2

Contents

2.1 Installation guide

Sampling of metabolic models is a two-step process. The first step is the generation of warmup points that will be used as a startign point for the actual sampling.

2.1.1 Generation of warmup points

To generate warmup points for metabolic models, we will use the `VFWarmup` tool that uses a hybrid MPI/OpenMP distributed approach to ensure dynamic load balancing.

Requirements

- Linux-based system
- IBM CPLEX 12.6.3 and above Free academic version
- OpenMp comes by default in the latest gcc versions
- MPI through the OpenMPI 1.10.3 implementation.

Quick install

```
cd VFWarmup
source ./install.sh
make
```

Troubleshooting

Quick install downloads and installs 1) OpenMPI and 2) IBM CPLEX for 64-bit machines.

You can do each step separately if quick install did not work or if you have different machine specs.

- MPI: You can use the following code snippet to install MPI

```
VERSION=3.1.2
wget --no-check-certificate https://www.open-mpi.org/software/ompi/v3.1/downloads/
↪openmpi-$VERSION.tar.gz
tar -xzf openmpi-$VERSION.tar.gz
cd openmpi-$VERSION
mkdir build && cd build
../configure CFLAGS="-w" --prefix=$HOME/open-mpi \
             --without-verbs --without-fca --without-mxm --without-ucx \
             --without-portals4 --without-psm --without-psm2 \
             --without-libfabric --without-usnic \
             --without-udreg --without-ugni --without-xpmem \
             --without-alps --without-munge \
             --without-sge --without-loadleveler --without-tm \
             --without-lsf --without-slurm \
             --without-pvfs2 --without-plfs \
             --without-cuda --disable-oshmem \
             --disable-mpi-fortran --disable-oshmem-fortran \
             --disable-libompitrace \
             --disable-mpi-io  --disable-io-romio \
             --disable-static #--enable-mpi-thread-multiple
make -j2
make install
```

You might also need to add MPI path

```
export PATH=$HOME/open-mpi/bin:$PATH
```

- IBM CPLEX: The recommended approach is to download IBM CPLEX and register for the free academic version.

Make sure that the CPLEXDIR path in `VFWarmup/Makefile` corresponds to the installation folder of CPLEX.

- Once the required dependencies installed, `cd ACHR.cu/VFWarmup` then `make` at the root of `VFWarmup`.
- Alternatively, you can open an issue here.

2.1.2 Sampling

The actual sampling uses `ACHR.cu` and starts from the warmup points generated by `VFWarmup`. It is a CUDA-based GP-GPU software.

Requirements

- Linux-based system
- Nvidia GPU with `sm_35` architecture. Check the specs of your card here This architecture is needed as `ACHR.cu` uses nested parallelism to gain even higher speed-ups.
- CUDA v8.0 and above
- GSL linear algebra library needed for the sequential SVD and QR.

- IBM CPLEX 12.6.3 and above

Once the required dependencies installed, `make` at the root of `ACHRcu`

Quick install

```
cd ACHRcu
source ./install.sh
make
```

Troubleshooting

Quick install downloads and installs 1) CUDA 8.0 for 64 bit machines and 2) GSL.

You can do each step separately if quick install did not work or if you have different machine specs.

- CUDA v 8.0: You can download CUDA here, then follow the instructions for the installation.

You might also need to add CUDA path

```
export PATH=/usr/local/cuda-8.0/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda-8.0/lib64:$LD_LIBRARY_PATH
```

CUDA v 8.0 and above are required because ACHR.cu enables dynamic parallelism. For CUDA v 8.0, a GCC version < 5.0 is required.

You can install GCC 4.8 like the following:

```
sudo apt install gcc-4.8 g++-4.8
sudo ln -s /usr/bin/gcc-4.8 /usr/local/cuda/bin/gcc
```

You can also install newer CUDA versions that support recent versions of GCC.

- GSL: You can install GSL like the following

```
sudo apt-get install libgsl-dev
```

- IBM CPLEX: check the installation of `VFWarmup` for the download and install of IBM CPLEX.

Make sure that the CPLEXDIR path in `ACHRcu/Makefile` corresponds to the installation folder of CPLEX.

- Once the required dependencies installed, `cd ACHR.cu/ACHRcu` then `make` at the root of `ACHRcu`.
- Alternatively, you can open an issue here.

2.2 Usage guide

Sampling of metabolic models is a two-step process.

2.2.1 Generation of warmup points

The generation of warmup points is done through `VFWarmup` software. After installing the dependencies of `VFWarmup`, you can build the binaries at the root of `VFWarmup` using `make`.

Then call `VFWarmup` as follows:

```
mpirun -np nCores --bind-to none -x OMP_NUM_THREADS=nThreads createWarmupPts
model.mps SCAIND
```

Replace the following variables with your own parameters:

- nCores: the number of non-shared memory cores you wish to use for teh analysis
- nThreads: the number of shared memory threads within one core
- model.mps: the metabolic model in `.mps` format. To convert a model in `.mat` format to `.mps`, you can use the provided converter `convertProblem.m`
- SCAIND: (optional) corresponds to the scaling CPLEX parameter SCAIND and can take the values 0 (equilibration scaling: default), 1(aggressive scaling), -1 (no scaling). scaling is usually desactivated with tightly constrained metabolic model such as coupled models to avoid numerical instabilities and large solution times.

Example: `mpirun -np 2 --bind-to none -x OMP_NUM_THREADS=4 createWarmupPts ecoli_core.mps`

You will have to input the number of warmup points to be generated, this is usually a minimum of 2n, where n is the number of reactions in a metabolic model. `VFWarmup` will perform a minimization and a maximization in each dimension, which means that 2n is the minimum number of samples needed to delienate the solution space.

The ouput file is saved as `modelnPtswarmup.csv`, with model is the name of the metabolic model and nPts is the number of warmup points generated.

2.2.2 Sampling

The actual GPU sampling is done through `ACHR.cu` software. After installing the dependencies of `ACHR.cu`, you can build the binaries at the root of `ACHR.cu` using `make`.

Then call `ACHR.cu` as follows:

```
./ACHRCuda model.mps warmuppoints.csv nFiles nPoints nSteps
```

Replace the following varaibles with you own parameters:

- model.mps: the metabolic model in `.mps` format.
- warmuppoints.csv: the warmup points obtained using `VFWarmup`. You can also use warmup points generated using other software such as the MATLAB CobraToolbox and Cobrapy.
- nFiles: Number of files to stores the sampled solution points.
- nPoints: number of points per file.
- nSteps: number of steps per point.

Example: `./ACHRCuda ecoli_core.mps ecoli_core1000warmup.csv 1 1000 1000`

2.3 Tutorials

First, make sure that `VFWarmup` and `ACHR.cu` are correctly installed.

2.3.1 Sampling of Ecoli core

Sampling is a two-step process. First, let's generate the warmup points for sampling using `VFWarmup`

```
mpirun -np 1 --bind-to none -x OMP_NUM_THREADS=4 createWarmupPts ecoli_core.
mps
```

Make sur you pick a number of warmup points greater than 95*2=190, say 200 for instance. The output has been written to ecoli_core200warmup.csv

Then, let's sample the solution space of Ecoli core metabolic model starting from the warmup points generated previously:

```
./ACHRCuda model.mps ecoli_core200warmup.csv 2 1000 1000
```

We will generate 2 files contatining each 1000 points. Each points has converged after 1000 step. The total number of points generated is 2*1000=2000.

2.4 Changelog

- : Improve the docs
- : Changelog added to the doc
- : Added quick install script
- : Improve the changelog structure

2.5 License

The software is provided under MIT License.

```
MIT License

Copyright (c) 2018 Ben Guebila Marouen

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search