Final Project: Protein Docking

Dani Jiménez

1415-Q2

Index

1

Final Project

1.1 Objective of the project

The objective of the project is to evaluate if the student is able to apply different optimization techniques (the possible ones), using a correct optimization methodology, to a given source code.

1.2 Description

We propose you to optimize a known and used protein docking application, called `ftdock`. In fact, we propose you to optimize part of the `ftdock`. We have already modified the code so that the execution is stopped at some point with a message (*"PCA STOPS HERE"*). At this point, 20 solutions to the docking algorithm are written down. You can redirect that output to a file to be able to check the correctness of your optimizations later on.

For instance, test1.output looks like:

```
        3D-Dock Suite (March 2001)
        Copyright (C) 1997-2000 Gidon Moont
        This program comes with ABSOLUTELY NO WARRANTY
        for details see license. This program is free software,
        and you may redistribute it under certain conditions.

        Biomolecular Modelling Laboratory
        Imperial Cancer Research Fund
        44 Lincoln's Inn Fields
        London WC2A 3PX
        +44 (0)20 7269 3348
        http://www.bmm.icnet.uk/

Starting FTDock (v2.0) global search program
PCA TIMING SHOULD start here
  reading parsed pdb file: ../2pka.parsed
  reading parsed pdb file: ../5pti.parsed
Total number of rotations is 9240
Assigning charges
Using automatic calculation for grid size
Span = 93.436 angstroms
Grid size = 134
Each Grid cube = 0.69728 angstroms
Electrostatics are on
Creating plans
Setting up Static Structure
```

```
  surfacing grid
  electric field calculations ( one dot / grid sheet ) ..........................................
.............................................
  one time forward FFT calculations
  done
Starting main loop through the rotations
.G_DATA      1        0        147         -1  -12  -42         0   0   0
G_DATA      1        0        145         11  -27  -28         0   0   0
G_DATA      1        0        139         28   21   10         0   0   0
.G_DATA      2        0        204          10   37   -1        12   0   0
G_DATA      2        0        157        -33   27   -6        12   0   0
G_DATA      2        0        148        -34  -12   -2        12   0   0

...

PCA TIMING SHOULD stop here
PCA STOPS HERE
```

That output will show the best three scores for 20 rotations angles; from the rotation angle (0, 0, 0) up to (228,0,0) (last three columns show those angles) for docking 5pti.parsed ligand to 2pka.parsed receptor. First column is fixed (G_DATA), second column is the rotation number, third column is fixed to 0, fourth is the score given to the docking (147 in the first row) in the coordinates after that number. Finally next six numbers are the coordinates (3 of them) and the angles (last three, all of them fixed to 0 because is the first rotation tested). **NOTE THAT the ONLY thing you should check for the correctness of your program are the coordinates. It DOESN'T mind that the three G_DATA appear in different order or with slightly different score (2 points more or less is normal). Also, it may happens that few coordenates are changed by others but the scoring should be similar**.

You should apply the techniques and methodology used in the course to re-code the parts of the code that you consider necessary to reduce the execution time of the program. Also, upgrading to up-to-date software that can help to improve the overall performance of the ftdock is suitable for this project.

The final project should be done in pairs.

1.3 Project Description

`ftdock` performs the rigid-body docking of two proteins: one small and the another bigger than the another. Docking means the way the two proteins can be joined in the space. In this final project, the computation of the electrostatic matrix at the begining of the program is important to be able to accept or not a possible relative position of the two proteins in the space.

There are two parameters that you have to use when running `ftdock`: `-static` and `-mobile`. `-static` parameter is used in order to indicate the parsed protein that is big. `mobile` is used in order to indicate the parsed protein that is small. You can find the parsed input proteins in the `proteins.tar.gz` file, in the inputs directory.

1.3.1 Source code and input files

In order to be able to run and test your final project you need the following files:

1. `fftw-2.1.3.tar.gz` : Old version of the FFTWs library.
2. `gnu_licensed_3D_Dock.tar.gz`: Source code to be optimized.
3. `proteins.tar.gz`: Parsed proteins.

The application that you should optimize is ftdock, which is under the progs directory when decompressing the gnu_licensed_3D_Dock.tar.gz file.

1.3.2 Installation Steps

We ask you to use the Intel Corei5 of the lab in order to be able to compare the optimizations done among all the students. However, if you want do some experiments in another processor (or you are not able to use the lab computer for some reason) you should specify the processor and environment setup you have used in the final document that you have to submit to us, so that we can understand the improvents achieved by your optimizations. The optimization flags should be the ones given in these installation steps. However, if you want to use others you can do it, specifying which is the improvement achieved in this case by the compiler flags you use.

In the case of the library installation you have to follow the steps below. Note that PATH_LLIB is the absolute path that you want to use for your FFTW library installation. We enable floats in order to work with real number of 4 bytes rather than 8 bytes.

```
> cp fftw-2.1.3.tar.gz PATH_LLIB
> cd PATH_LLIB
> tar -xvzf fftw-2.1.3.tar.gz
> cd fftw-2.1.3
> mkdir installation
> env CFLAGS="-O3 -march=native" ./configure --prefix=$PWD/installation --enable-float
> make
> make install
```

IMPORTANT NOTE: In the case you have any problem installing the fftw in your FIB account, you will need to change the Makefile files that you will find in fftw, rfftw and doc directories. Edit those makefiles and look for "/usr/bin/install -c" and change this command by "cp" command. Then, you will also need to remove "-m 644" options some lines below in the Makefiles.

In the case of the application installation you have to follow the steps below. Note that PATH_APLI is the absolute path that you want to use for your application (ftdock) installation.

```
> cp gnu_licensed_3D_Dock.tar.gz PATH_APLI
> cd PATH_APLI
> tar -xvzf gnu_licensed_3D_Dock.tar.gz
> cd PATH_APLI/3D_Dock/progs/
> vim Makefile
1.- change the FFTW_DIR variable value with that:
FFTW_DIR = PATH_LLIB/fftw-2.1.3/installation
2.- change the -march option to use native
-march=native
> make
```

1.3.3 How to run it?

The `ftdock` program is located in `PATH_APLI/3D_Dock/progs`. We suggest you to use the following parameters in order to analyze and optimize the `ftdock` program.

```
-static 2pka.parsed -mobile 5pti.parsed
-static 1hba.parsed -mobile 5pti.parsed
-static 4hhb.parsed -mobile 5pti.parsed
```

Redirect the standard output to a file, in order to be able to compare/check the program results.

```
./ftdock -static 2pka.parsed -mobile 5pti.parsed > absolute_path/output
```

Note that the parsed protein should be in your PATH variable or in the current directory you are running `ftdock`.

1.4 Evaluation

The evaluation of this work will be based on the modifications and optimizations of the code you make, the source code of your optimizations, the shell scripts you have used in order to run, check and test your optimizations, and the pdf report you submit. That document should explain all the optimizatons you have done, the methodology you have used, results, speedups, etc.

The methodology we expect you to apply is that we have explained at the beginnning of the course. Remember that the analysis that you do about the results you obtain is as important as the final speedup you get. So, for instance, we expect:

- For each optimization step, you should show the weight of the most consuming functions.
- Which optimization options you have for each part of the code.
  - It would be good to analyze them one by one (and all together if they are compatible).
- Possible optimizations that you thought but they didn't work.
  - In this case, it would be good to analyze why those optimizations didn't work.
- Figures with speedup's, time, etc.
- Assembly code analysis if it is needed.
- Etc.

1.4.1 Which documents you have to submit us?

You have to submit a .pdf file with all the analysis of the results, and the methodology of the optimizations you have done. **This .pdf smust not have more than 15 pages**. Also, you have to submit the source code, makefiles, and any scritps you have used in order to do your work.

Enumeration of the documents, etc. you have to submit:

1. .pdf document
2. scripts used
3. execution output results
4. patches or source code of your optimizations
5. makefiles
6. README to run your scripts
7. INSTALL guide to install whatever you have done

How to submit it?

You will have to create a .zip file with all the documents explained above and submit it via the RACO of PCA.