**1. Introduction & Background:**

One challenge with the using the ppc64le architecture in the sciences is that most tools were built on x86 infrastructure. Alternative compute hardware usually has not been available to the academic researcher. A few years back, Rackspace acquired some ppc64le hardware in partnership with IBM.

Given the COVID-19 crisis, I decided to reconstitute the core of my protein engineering pipeline stack from graduate school on our ppc64le machines in the lab. Central to this tool stack is the Rosetta software suite. While much can be said for the protein folding problem, the Rosetta software suite is more than macromolecular folding – it is a full drug discovery pipeline: This advantage has resulted in numerous papers in Science and Nature. After twenty years of continued development, Rosetta continues to improve its bag of tools which depends on the collaboration of a global community of scientists. Rosetta is now over three million lines of code and available through www.rosettacommons.org.

As a researcher constrained by what resources are available at hand, you have to be able to adapt.

Rosetta still depends on a variety of other resources and I will provide a tutorial on building a deployment of Rosetta on ppc64le servers that is extensible to a multinode cluster with a shared file system such as BeeGFS or basic NFS.

One must install the following set of software and databases:

Rosetta version 3.10

Sparks-x (on an x86 machine) since the lab group distributes their code as a binary only.

PSIPRED

PFILT

BLAST (legacy version 2.2.22)

BLAST+ (version 2.10.0) \*optional

Uniref90 Database (~70GB)

Non Redundant Database (~130GB)

Durandal Clustering \*optional

Ambermini \*optional for initial coordinate generation

Ambertools \*some utilities needed for Ambermini

rdock

**2. Installing the software suites & required databases**

The OS chosen was Ubuntu 18.04 LTS

a. Install the dependencies beforehand (extended for illustration):

sudo apt install build-essential -y

sudo apt install libsqlite3-dev zlib1g-dev –y

sudo apt install libswitch-perl

sudo apt install libmpich-dev libmpich12 mpich

sudo apt install libcppdb-dev libxml2-dev libxml2

sudo apt install scons python python3

b. Download all the dependent software resources:

i) **Ambermini**:

git clone <https://github.com/choderalab/ambermini>

ii) **Ambertools**:

<https://ambermd.org/AmberTools.php>

Fill out the form

Ff14SB forcefield library <https://ambermd.org/AmberModels.php>

Download

ff99SB: https://ambermd.org/downloads/ff99SBildn.tar

from the ambertools copy over the “dat” folder into the ambermini

then untar ff99SBildn.tar file into the folder: ambermini/dat/leap/parm

under the ambermini folder configure and make the executables:

sudo apt install byacc flex

./configure --prefix <destination>

make

make install

iii) **psipred & pfilt**:

http://bioinf.cs.ucl.ac.uk/software\_downloads/

**Install psipred:**

wget <http://bioinfadmin.cs.ucl.ac.uk/downloads/psipred/psipred.4.02.tar.gz>

untar the file

cd to-wherever-you-untarred-PSIPRED

cd src

make

make install

**Install version 1.4 of pfilt since 1.5 causes a segmentation fault:**

wget <http://bioinfadmin.cs.ucl.ac.uk/downloads/pfilt/old/pfilt1.4.tar.gz>

compile simply by typing “make” in the folder after untaring.

iv) **Durandal optional**:

<http://www.riken.jp/zhangiru/software/durandal_released_qcp.tgz>

you will have to use g++-5 and gcc-5

sudo apt install g++-5 gcc-5

switch to g++-5 and gcc-5

untar and cd into the folder and type “make”

v) **sparks-x**:

On your x86 laptop or handy x86 server machine, download the sparks utility and untar it.

<https://servers.sparks-lab.org/downloads/SPARKS-X.tgz>

vi) **NCBI Tools and Databases**

blosum:

<https://www.ncbi.nlm.nih.gov/Class/FieldGuide/BLOSUM62.txt>

<https://ftp.ncbi.nlm.nih.gov/>

<https://ftp.ncbi.nlm.nih.gov/blast/matrices/>

**BLAST Utilites**:

https://ftp.ncbi.nlm.nih.gov/blast/

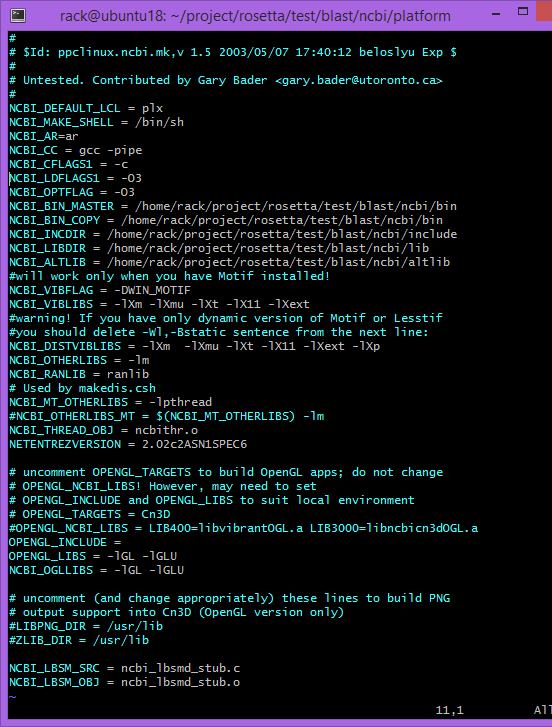
**BLAST (legacy version 2.2.22)**

<https://ftp.ncbi.nlm.nih.gov/blast/executables/legacy.NOTSUPPORTED/2.2.22/ncbi.tar.gz>

under the platform folder

ncbi/platform

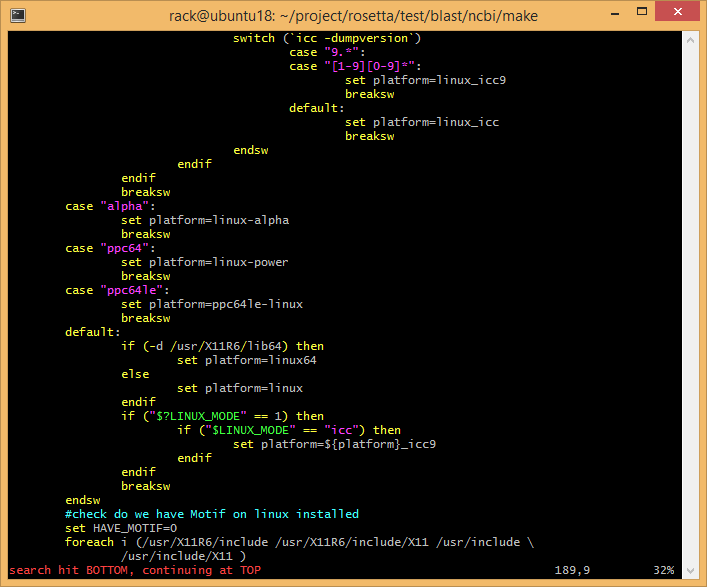
copy ppclinux.ncbi.mk as ppc64le-linux.ncbi.mk



Build instructions are in /ncbi/make

readme.unx

edit makedis.csh around line 186 to be the following:



Backout to the same level as where you untared the ncbi folder and execute:

./ncbi/make/makedis.sh

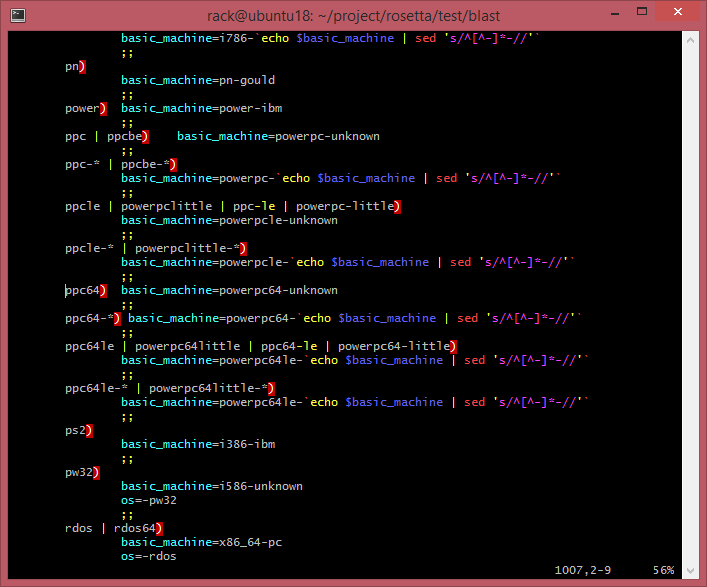
The binaries will be in the ./ncbi/bin folder

**BLAST+ (version 2.10.0)**

wget <https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/2.10.0/ncbi-blast-2.10.0+-src.zip>

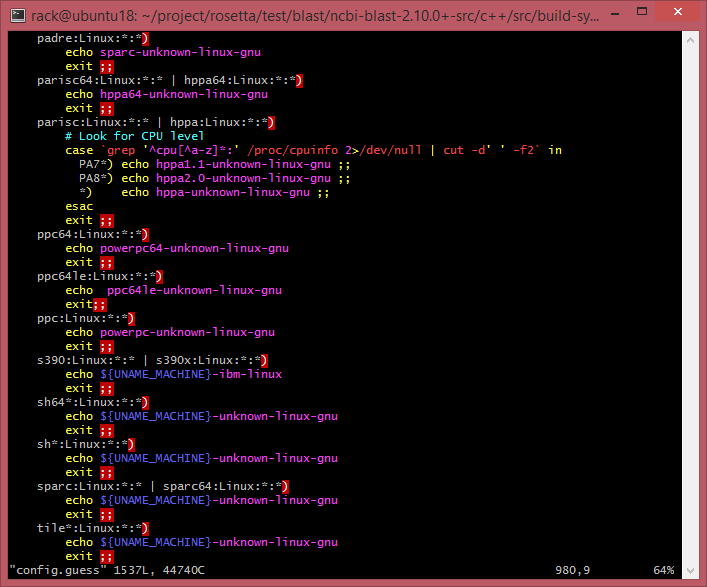
in the file config.sub found here ncbi-blast-2.10.0+-src/c++/src/build-system/config.sub

around line 1007:



and copy ppc64liux.make.in as ppc64le-linux.make.in

verify config.guess around line 980 is the following:



Go back to the c++ folder

And we assume that you switched back to g++-7

src/c++

Then type

cd c++

./configure

cd ReleaseMT/build

make all\_r

vii) **Protein Databases:**

These are very large files even compressed.

**uniref90**: from <https://www.uniprot.org/downloads>

wget <ftp://ftp.uniprot.org/pub/databases/uniprot/uniref/uniref90/uniref90.fasta.gz>

24GB compressed

**NCBI Non Redundant protein database**:

wget <ftp://ftp.ncbi.nlm.nih.gov/blast/db/FASTA/nr.gz>

70GB compressed

Go to each folder that these compressed files were uncompressed and execute

<where you put>/ncbi/bin/formatdb –i <dtabase> -p T –v 10000

You may want to hit control “z”

%> bg

%> disown (job #)

The legacy format database utility will shard and index the shards

We will need a “filtered” version of the nr database and will callit “nrfilt” in a separate folder.

<pfiltfolder>/pfilt nr > nrfilt

This will take a while so control “z”, bg and disown the process.

It is an old school c program but works. Version 1.5 has a buffer overflow bug.

After the creation of nrfilt, you will format it again

formatdb –i nrifilt –p T –v 10000

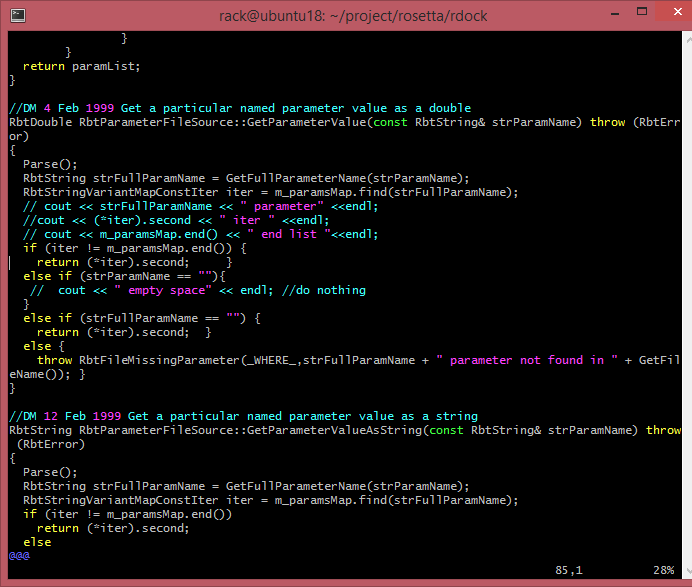
viii) **rdock (optional)**:

<http://rdock.sourceforge.net/download/>

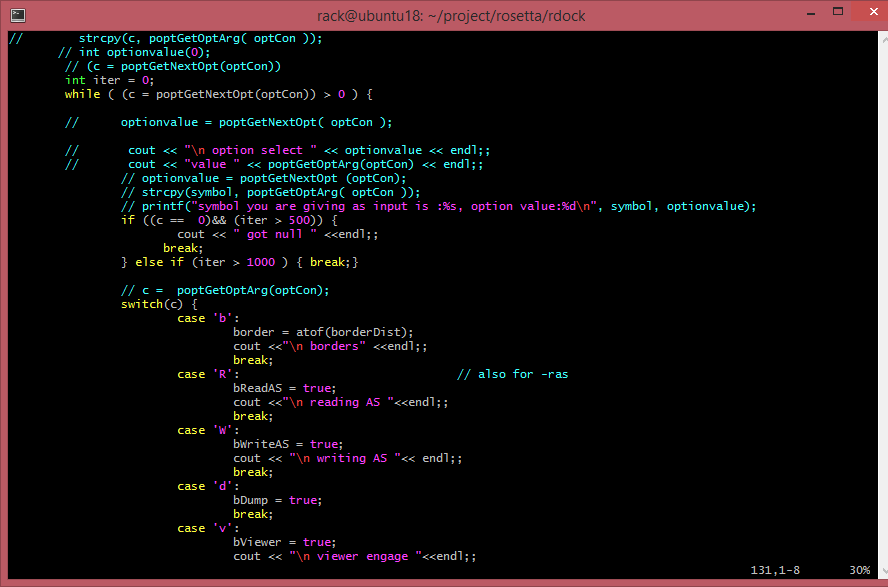
tar -xvzf rDock\_2013.1\_src.tar.gz

you need to comment out the “copy\_if” function form the c++ library of stl\_algo.h

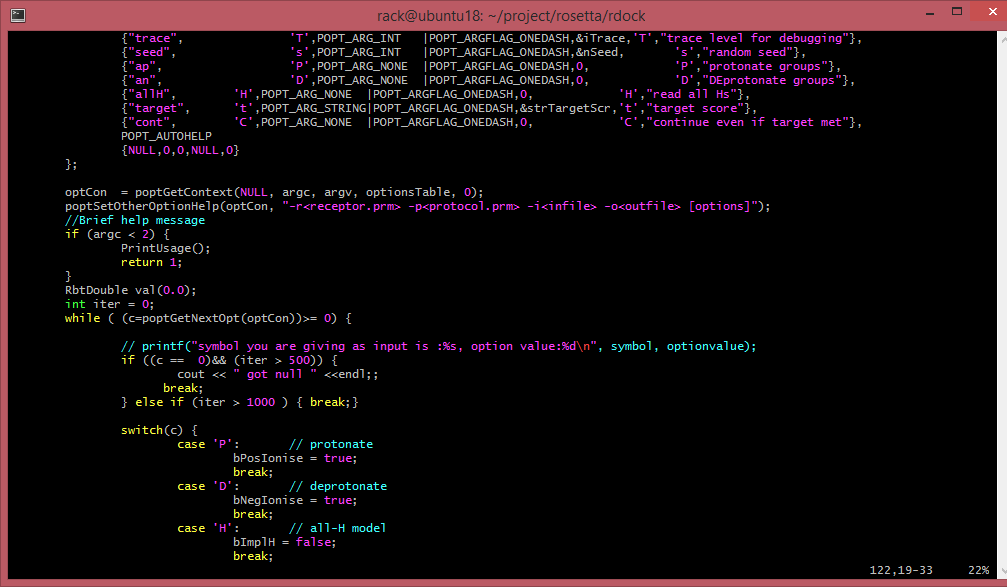
in the rDock\_2013.1\_src/src/lib.RbtParameterFile.source.cxx line 85, insert new code to ignore the null param calls:



Edit the rbcavity.cxx file to compensate for misbehavior of the libpopt library in ppc64le



In the file src/exe/rbdock



cd rDock\_2013.1\_src/build/

make linux-g++-64

make test

export RBT\_ROOT=/path/to/rDock/installation/

export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:$RBT\_ROOT/lib

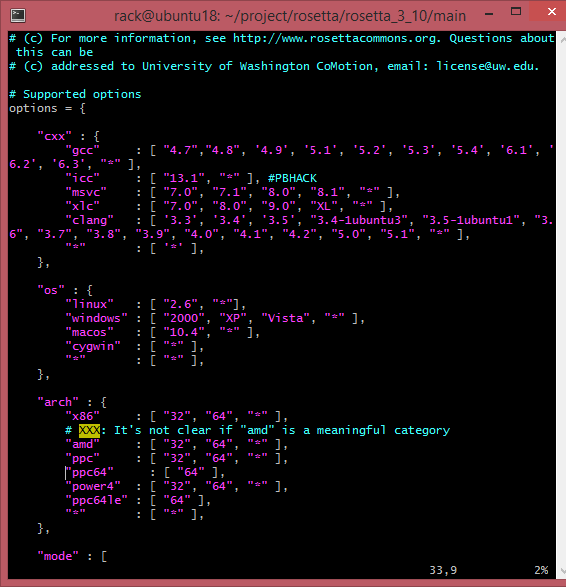
export PATH=$PATH:$RBT\_ROOT/bin

ix) **Install Rosetta v 3.10**

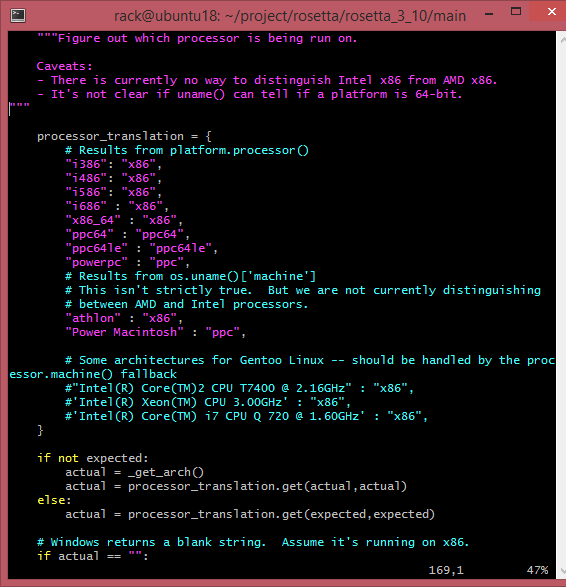
Once you download and untared rosetta, you must edit the following

Under main/source/tools/build

options.settings line 35: add “ppc64le : [ “64”]



setup\_platforms.py line 154: add “ppc64le” : “ppc64le”,



goto rosetta\_bundle/main/source

./scons.py bin mode=release extras=mpi

==============================

We have now completed the following:

Installing Rosetta

Installing NCBI DB

Uniref90.fasta

Non redundant database

Installing Legacy BLAST and BLAST+

Installing Psipred

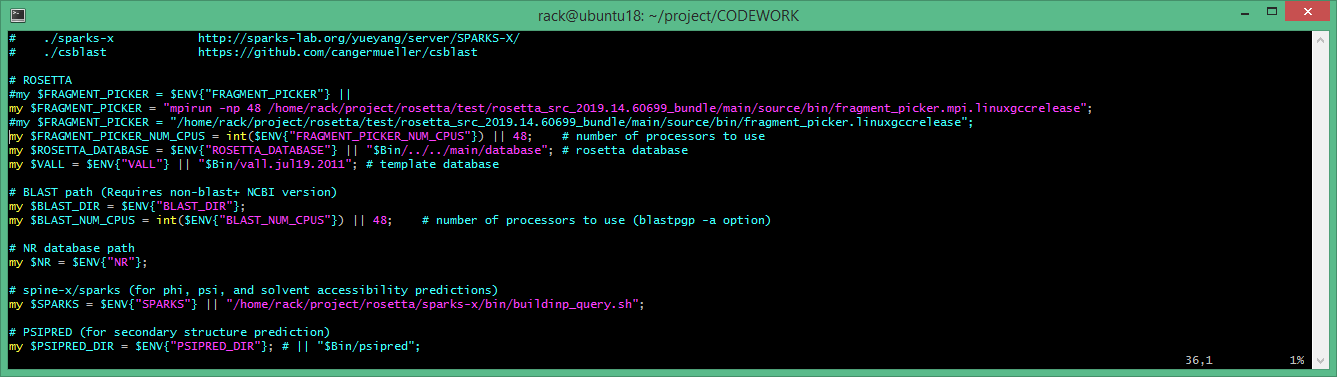
Installing Ambermini

Installing rdock 2013

**3. Set up environmental variables and adapt the make\_fragments.pl utility**

Edit the file tools/fragment\_tools/make\_fragments.pl

..and populate all the path variables.



N.B. the sparks path was before I realized that I couldn’t get the source code in order to compile for ppc64le

Copy the lips perl script from UIC into a file:

http://gila.bioe.uic.edu/lab/lips/lips.txt

and rename it as uic\_lips.pl

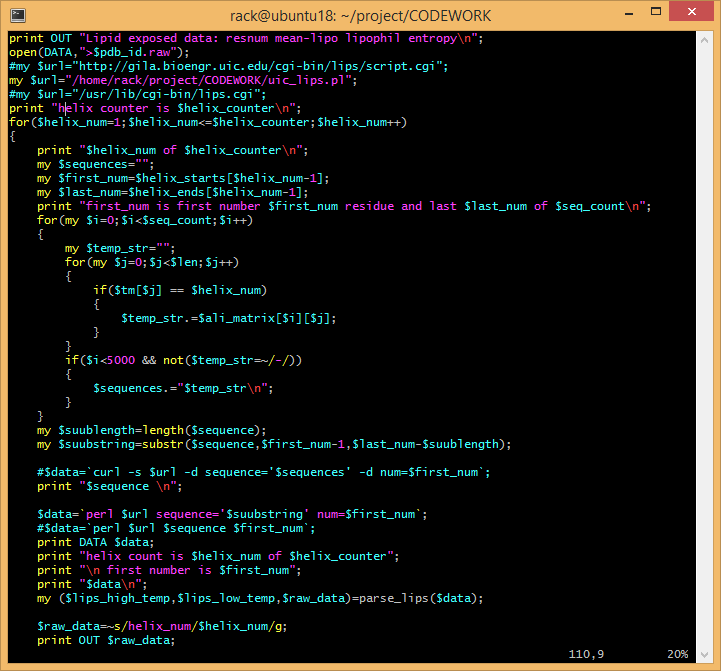
Locate the

rosetta\_3\_10/tools/membrane\_tools

run\_lips.pl

Edit the file around line 105 to the following:

1) Point the url variable to your local copy of the UIC file.



Add the changes

my $last\_num=$helix\_ends[$helix\_num-1];

…

my $suublength=length($sequence);

my $suubstring=substr($sequence,$first\_num-1,$last\_num-$suublength);

…

$data=`perl $url sequence='$suubstring' num=$first\_num`;

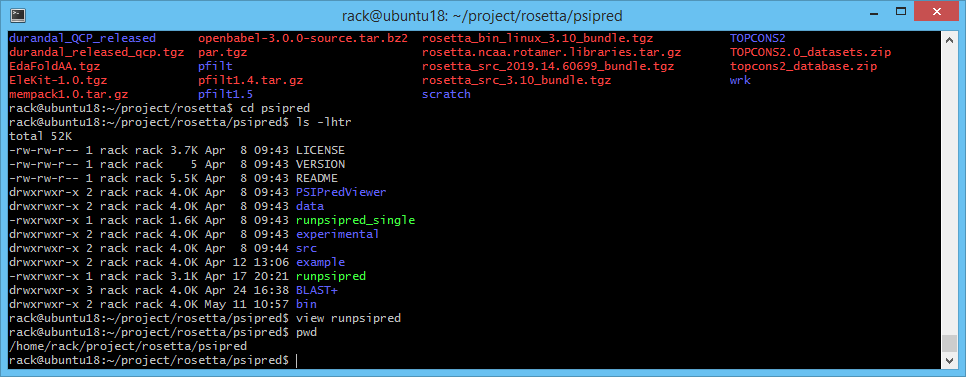
**Set environmental variables:**

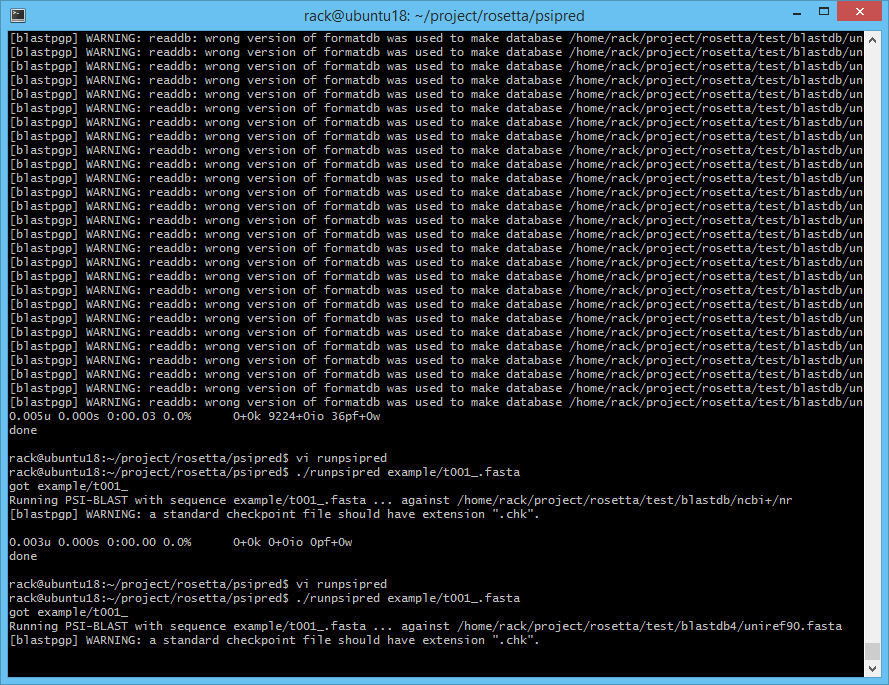
In the .bashrc profile, you must set up all the variables that are used by these programs to find each other.

**4. Making the input files for membrane proteins**

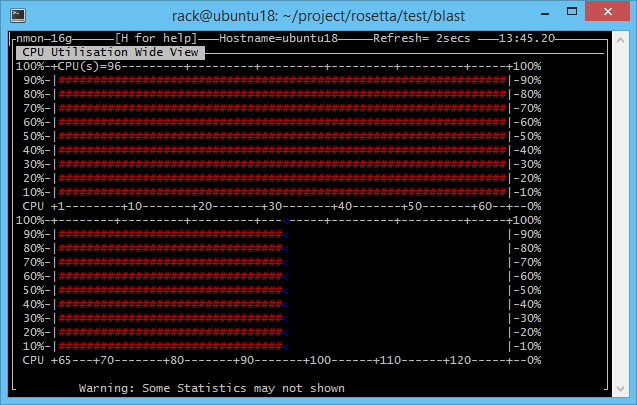
The abinitio structure prediction protocol requires at least the 3mer and 9mer fragment files.

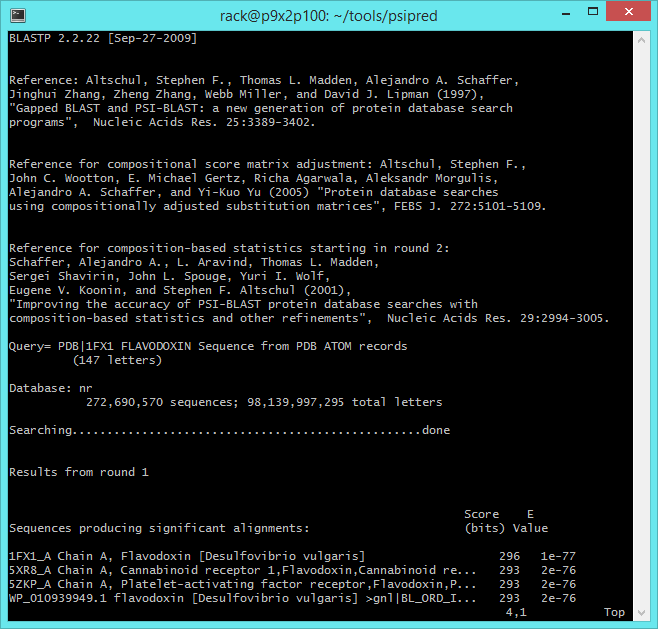
a. Run psipred separately to obtain the checkpoint file to be fed into sparks-x



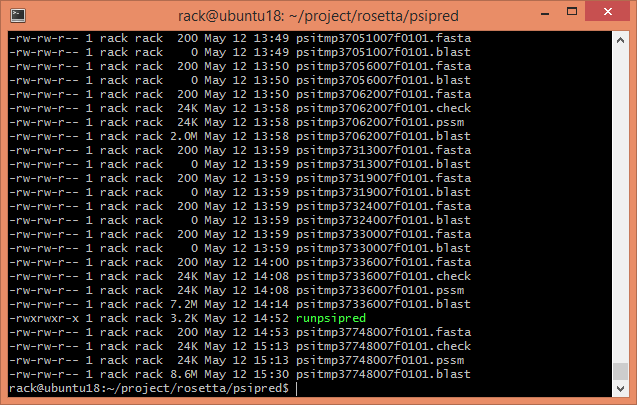


In this example, I am using all 96 threads





b. Run the sparks utility with the generated pssm file.



In practice, you should rename the pssm file to whatever your fasta file was called.

Copying the pssm file to your x86 machine where the Sparks utility is installed,

Set the environmental variable for the sparks directory assuming bash:

export SPARKSXDIR=/<where you put the directory>/sparks-x

then run the buildinp\_query.sh <your file>.pssm

sparks-x/bin/buildinp\_query.sh <your file>.pssm

and it will generate the phipsi file which you will copy back to your ppc64le environment to make fragments

<yourfile>.phipsi

<yourfile>.inp

copy the phipsi file over to where your project folder is and run the make\_fragments.pl utility

make\_fragments.pl -nojufo -nosam -noprof -verbose <five letter>.fasta

after some time, two fragment files will be generated

1. <fasta name>.200.3mers

2. <fasta name>.200.9mers

c.f. <https://www.rosettacommons.org/manuals/rosetta3_user_guide/file_fragments.html>

If this is a membrane protein, then you will need to run the lipophilicity profile with the run\_lips.pl protocol:

If octopus is not installed, then copy and paste your fasta sequence in the server:

<http://octopus.cbr.su.se/>

download the generated topology file and save it in your project folder

This file is used to generated a membrane map.

Without a script, you must run the these commands in succession:

octopus2span.pl octopus.topo <spanfile>.span

blastpgp –i <fasta file> –d <nonredundant database> -j 2 –h 0.001 –b5000 –v5000 –o <blastout file> -a <#procs>

run\_lips.pl <fasta> <spanfile>.span <nonredundant database> blastpgp alignblast.pl

this latter part generates the lipo file <name>.lips4

**5. Running a job**

**6. Analyzing results**

**References:**

Software Tools and Data Stack:

NCBI Tools & Data

BLAST (legacy version 2.2.22)

BLAST+ (version 2.10.0)

Uniref90 DB

NR DB: ftp://ftp.ncbi.nlm.nih.gov/blast/db/FASTA/nr.gz

Rosetta version 3.10: request a license (academic or commercial)

Ambermini:

Ff14SB forcefield library <https://ambermd.org/AmberModels.php>

ff99SB: https://ambermd.org/downloads/ff99SBildn.tar

Ambertools 18: <https://ambermd.org/downloads/>

Psipred:

Pfilt:

Optional for docking and clustering:

Durandal Clustering:

<https://www.ncbi.nlm.nih.gov/pubmed/22120171>

<http://www2.riken.jp/zhangiru/software.html>

<http://www.riken.jp/zhangiru/software/durandal_released_qcp.tgz>

<http://www.riken.jp/zhangiru/software/durandal_released.tgz>

rdock (version 2013):

Dependencies:

Rosetta: libsqlite3 libsqlite3-dev

Sparks-x: libswitch-perl

MPI dependencies: libmpich-dev libmpich12 mpich

scons

python python3

Notes:

If make\_fragements doesn’t work, be sure to check if the psipred file generated is in the vertical format and file paths are correct. Always add the verbose flag when debugging.

Additional Programs:

Spicker from Zhanglab:

https://zhanglab.ccmb.med.umich.edu/SPICKER/

Running Jobs/Experiments:

mpiexec –launcher fork –np (lscpu/4 -4) <rosetta protocol>.mpi.linuxgccrelease… @infile

mpirun –np <cores>