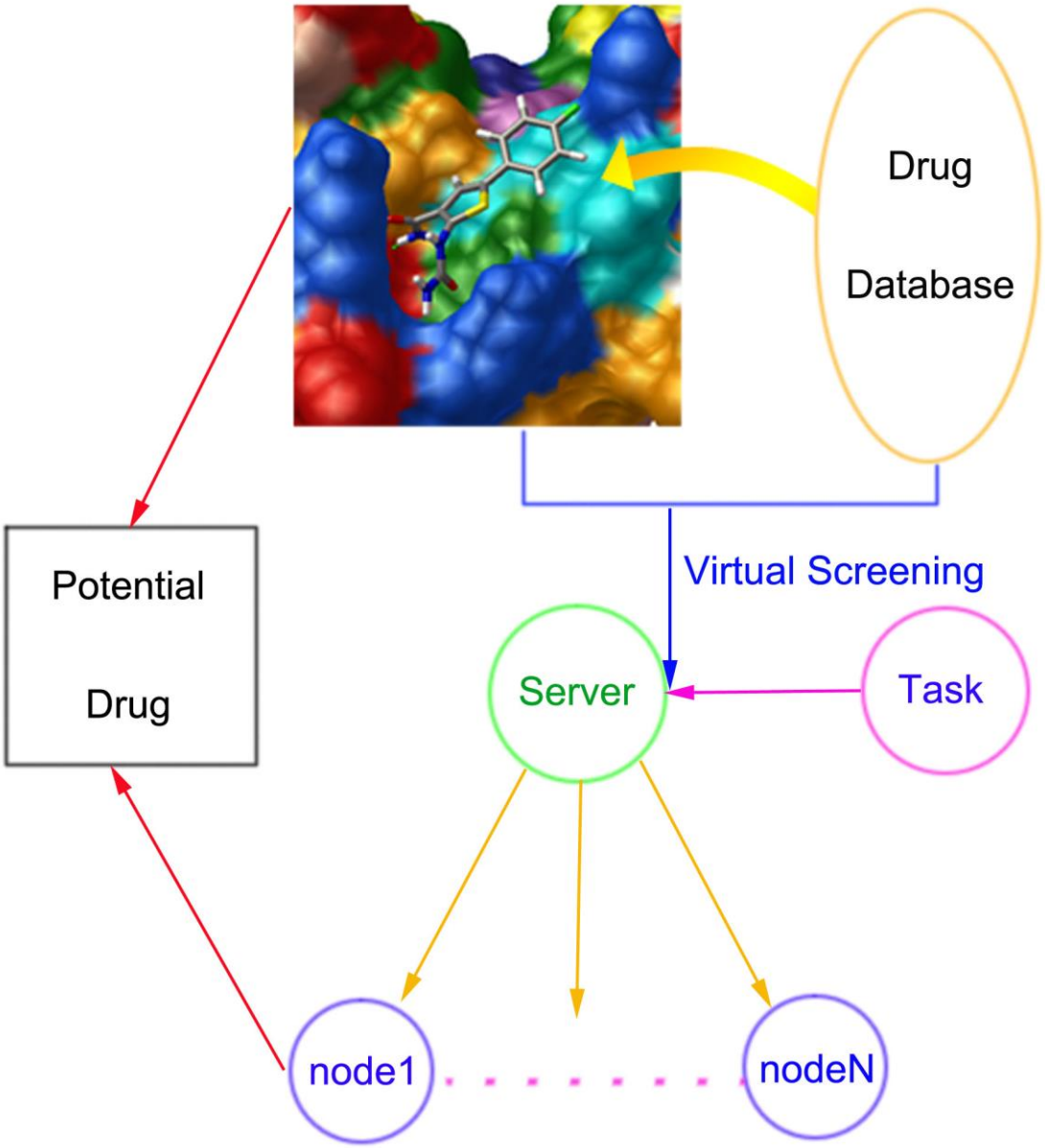
Tutorial of Virtual Screening of MolGridCal



Update by Qifeng Bai

2017-11

Contents

Introduction

Grid computing could utilize the idle computer resource to realize the common goals. It can be applied to the different many different fields, such as Hadron Collider[1], nuclear magnetic resonance (NMR)[2], image analysis[3] and so on. Especially, if the grid computing is used to discovery drugs, it will save money and time. About 3.5 billion target proteins, one of which possibly contained more than one active site, resulted in a large and complicated virtual screening work. The screensaver project of grid computing could supply enough computing resource to solve these problems[4]. JPPF (www.jppf.org) supplies a powerful framework for grid computing. A useful and detail example will supply a good demonstration to further apply the grid computing in drug discovery. Here, I wrote the grid computing program called MolGridCal for virtual screening of ligand database, such as ZINC database which contained over 35 million purchasable compounds[5]. The related useful links as below:

JPPF: http://www.jppf.org

Globus: http://www.globus.org

Hadoop: http://hadoop.apache.org

Openstack: http://www.openstack.org/

Now we started our tutorial!

Required Software

1. MolGridCal program (*https://molgridcal.codeplex.com/*).
2. Java SE 6 or later:

(*http://www.oracle.com/technetwork/java/javase/downloads/index.html*), we recommend Java SE 7. JPPF and MolGridCal were wrote by pure JAVA language, so the Java compiled environment was needed.

3. Apache Ant program *(http://ant.apache.org)*. The ANT 1.8.x or later were needed. In this study, the ANT (version 1.9.2) was chosen for compiling the source code.
4. The JPPF-4.2.1-driver and JPPF-4.2.1-node were needed (*http://www.jppf.org*) or (*http://sourceforge.net/projects/jppf-project*). **Caution:** Because some libraries exists the difference between different versions of JPPF, the version 4.2.1 is chosen for this tutorial. If you want to use the higher version of JPPF, just copy the library files of new version JPPF into the library path of MolGridCal.
5. The FTP server is required. You can pick up Serv-U FTP, FileZilla, Apache FtpServer, Wing FTP Server, *etc.* Here, we used the Apache FtpServer for file transferring *(http://mina.apache.org/ftpserver-project/)*
6. Autodock VINA (*http://vina.scripps.edu*) and MGLTools (*http://mgltools.scripps.edu/*) are required.

Installation

JDK and ANT environment

1. Decompressing files

In windows:

JDK: just **double click** and follow install instruction step by step.

ANT: just decompressing files.

In Linux:

%> tar -xzvf jdk-7u17-linux-x64.tar.gz

%> tar -xzvf apache-ant-1.9.2-bin.tar.gz

2. set the environment path:

In windows:

1. From the Desktop, right-click **My Computer and click Properties.**

In the System Properties window, click on the Advanced tab (Figure 1).

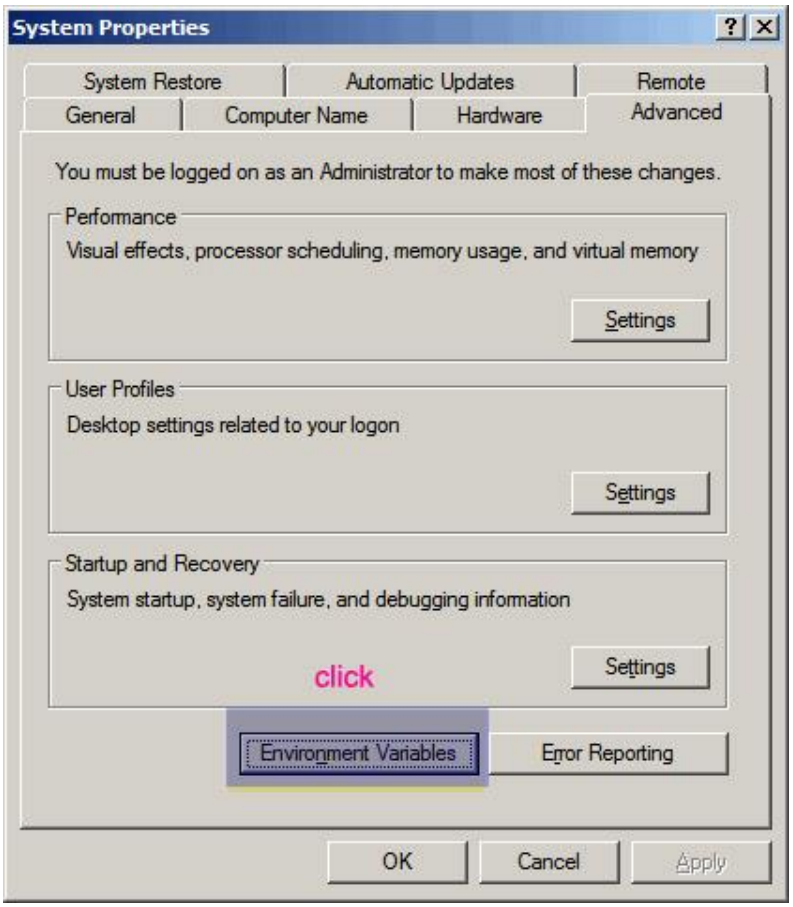


Figure 1: The Properties of system.

Then click the Environment Variables button (Figure 2):

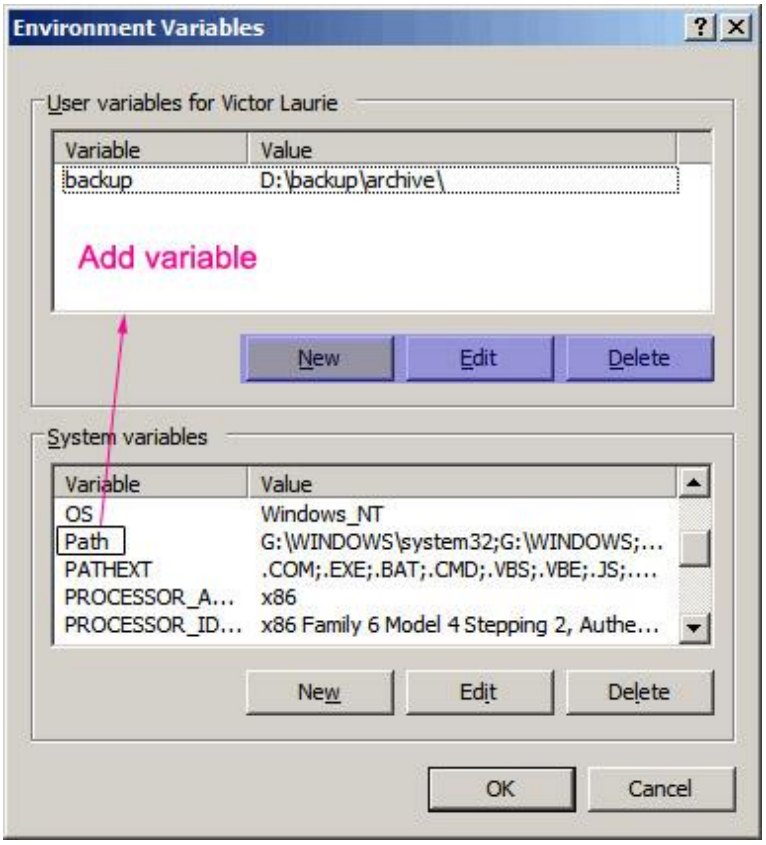


Figure 2: path of variables.

Finally, the Environment Variables in every entry is like as below:

```
C:\Program Files;C:\Winnt;C:\Winnt\System32
```

The different variables are separated by semicolon "**;**". I supply an instance for setting environment of JDK and ANT:

Double click the "Path" of Environment Variables and add

```
;C:\Program Files\Java\jdk1.7.0_02\bin
:C:\Program Files\Java\jre7\bin:D:\Program Files\ant\bin
```

Create or double click **CLASSPATH**

```
;C:\Program Files\Java\jdk1.7.0_02\lib;C:\Program Files\Java\jre7\lib;D:\Program Files\ant\lib
```

Create the variable JAVA_HOME

```
;C:\Program Files\Java\jdk1.7.0_02
```

Caution: this is just an example, please modify it according to your real environment.

JAVA_HOME is needed because the running of ant is required.

In Linux:

Open the root directory of current user. Edit the .bashrc and set the pathway of JDK and ANT. I just give an example, please modify it according to your real path of program.

```
export JAVA_HOME=/home/tim/bai/soft/jdk7    ###(Your own directory of JDK)
export ANT=/home/tim/bai/soft/ant192        ###(Your own directory of ANT)
export PATH=$ANT/bin:$JAVA_HOME/bin:$JAVA_HOME/jre/bin:$PATH
export
LD_LIBRARY_PATH=$ANT/lib:$JAVA_HOME/lib:$JAVA_HOME/jre/lib:$JAVA_HOME/jr
e/lib/amd64:$LD_LIBRARY_PATH
```

MGLTools and VINA install

MGLTools: In windows operation system, just double click and install it followed by the instruction.

In Linux operation system, using chmod command like below:

%> chmod +x mgltools_Linux-x86_64_1.5.6_Install

Then double click and install it followed by instruction.

The VINA has a similar install way with MGLTools.

MolGridCal setup

It is very easy to install MolGridCal. Just decompress the MolGridCal and use it directly.

Getting started

Protein preparation

Here, the agonist binding sites of β_2AR is chosen as the target of virtual screening. The dealing protocol of β_2AR is omitted. The PDBQT is provided directly. More about dealt method of receptor, please refer: *http://vina.scripps.edu/tutorial.html*

Ligand preparation

Open VStutorial/01-vsligands, you would found the file "**ligands.mol2**" which contain 45 mol2 from the ZINC database. we will used it for our quick start tutorial.

All the ligands were prepared on the Linux operation system using batch shell.

Split database into the single files

split.csh---spilt the database into the single files.

```
#!/bin/csh
mkdir ligtmp
cd ligtmp
cat ../ligands.mol2 | csplit -ftmp -n4 -ks - '%^@.TRIPOS.MOLECULE%' '/^@.TRIPOS.
MOLECULE/'  '{*}'
foreach f (tmp*)
echo $f
set zid = `grep ZINC $f`
if !(-e "$zid".mol2) then
set filename = "$zid".mol2
else foreach n (`seq -w 1 99`)
if !(-e "$zid"_"$n".mol2) then
set filename = "$zid"_"$n".mol2
break
endif
end
endif
mv -v $f $filename
end
```

%> csh split.csh

It will split the database into the single files.

Covert MOL2 to the PDBQT

convertPDBQT.csh---convert Mol2 to PQBQT

```
#!/bin/csh
cd ligtmp
foreach f (`ls *`)
        echo $f
        pythonsh ../prepare_ligand4.py -l $f    -d ../ligand_dict.py
end
```

set nonomatch=1 //if foreach no mathch

%> csh convertPDBQT.csh

It will convert the Mol2 to PDBQT format.

Caution: the **pythonsh** is part of MGLTools, so the full path should be gave if the variable environment is not set.

extract.csh---extract the PDBQT file

```
#!/bin/csh
mkdir ligands
cd ligtmp
mv *.pdbqt ../ligands
```

%> csh extract.csh

It will extract the PDBQT file into the pointed folder.

Please refer the “Using AutoDock 4 for Virtual Screening” if you want to get more information:

(*http://autodock.scripps.edu/faqs-help/tutorial/using-autodock4-for-virtual-screening*)

Build the FTP server

Here, we build the FTP server on Linux operation system.

1. %> tar -xzvf ftpserver-1.0.6.tar.gz

2. Because there will be a lot of connects of nodes when the virtual grid computing

start, so the ftp server should be set the enough number of max login at the same time. To make sure the transferring security of FPT server, the new user name and password should be set firstly. Open "**ftpserver/res/conf**" folder and edit the file of "**users.properties**" for user name and password (see Figure 3). In this tutorial, the MD 5 was used for encrypting. You can convert your password to MD 5 (http://www.miraclesalad.com/webtools/md5.php).

Figure 3: setting the username and password

"ftpd-typical.xml" was set as below**:**

If you want to start quickly, copy VStutorial/02-soft/ftpserver to overwrite the FTPserver. In this situation, the username and password of FTP server was gridfvs and molgridcal, respectively.

```
<server xmlns="http://mina.apache.org/ftpserver/spring/v1"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="
            http://mina.apache.org/ftpserver/spring/v1
http://mina.apache.org/ftpserver/ftpserver-1.0.xsd
            "
        id="myServer"
        max-logins="50000"
        max-threads="999999"
        anon-enabled="false"
        max-anon-logins="56000"
        max-login-failures="124"
        login-failure-delay="125">
        <listeners>
                <nio-listener name="default" port="2121">
                    <ssl>
                <keystore file="./res/ftpserver.jks" password="password" />
              </ssl>
                </nio-listener>
        </listeners>
        <file-user-manager file="./res/conf/users.properties" />
</server>
```

3. When the configure was setting, the ftp could be started using command in Linux as below easily:

%> bin/ftpd.sh res/conf/ftpd-typical.xml

In Windows operation system:

%> bin/ftpd.bat res/conf/ftpd-typical.xml

If you want to run the FTP server in background, run command:

bin/ftpd.sh res/conf/ftpd-typical.xml >& ftp.log &

4. This is just optional according to your tasks required. If you want to kill your background work, just use:

%> ps -ux

You find the ID of ftp server, and use command: kill “ID”, for example: kill 25691

The port of this ftp server was set to 2121 above.

More information about the configure of apache java FTP server could be found at *http://mina.apache.org/ftpserver-project/documentation.html*

Generate the certificate SSL

Because the nodes can search the server and connect to the server automatically, the Secure Socket Layer (SSL) was needed to authorized the certificate for connecting to server. If you want to start the MolGridCal quickly, you can skip this step using the default setting. The keytool was used to generate the certificate for servers, nodes and clients.

Please refer information of keytool:

JDK6 reference:

http://docs.oracle.com/javase/6/docs/technotes/guides/security/jsse/JSSERefGuide.html

JDK7 reference:

http://docs.oracle.com/javase/7/docs/technotes/tools/windows/keytool.html

http://docs.oracle.com/javase/7/docs/technotes/guides/security/jsse/JSSERefGuide.html

%> keytool -genkeypair -alias QFB -keyalg RSA -validity 7 -keystore keystore.ks

```
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  QFB
What is the name of your organizational unit?
  [Unknown]:  lzu
What is the name of your organization?
  [Unknown]:  lzu
What is the name of your City or Locality?
  [Unknown]:  lanzhou
What is the name of your State or Province?
  [Unknown]:  gansu
What is the two-letter country code for this unit?
  [Unknown]:  cn
Is CN=QFB, OU=lzu, O=lzu, L=lanzhou, ST=gansu, C=cn correct?
  [no]:  yes

Enter key password for <QFB>
        (RETURN if same as keystore password):
Re-enter new password:
```

%> keytool -list -v -keystore keystore.ks

```
Enter keystore password:

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: duke
Creation date: Aug 24, 2013
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=QFB, OU=lzu, O=lzu, L=lanzhou, ST=gansu, C=cn
Issuer: CN=QFB, OU=lzu, O=lzu, L=lanzhou, ST=gansu, C=cn
Serial number: 5de6f703
Valid from: Sat Aug 24 00:12:01 CST 2013 until: Sat Aug 31 00:12:01 CST 2013
Certificate fingerprints:
         MD5:  D2:42:A5:E2:F0:42:C0:87:77:5C:E6:BB:03:72:2D:D1
         SHA1: B8:CE:D0:1F:BF:03:46:B2:0B:E5:0C:46:8B:5D:35:5D:92:39:07:C9
         SHA256:
0C:86:FA:CB:31:3C:4C:C7:28:37:27:F0:14:3B:A4:90:B2:BD:87:ED:C2:74:AB:A7:E9:1B:11
:BD:97:DE:4E:F6
         Signature algorithm name: SHA256withRSA
         Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 33 65 DC 54 63 E5 D2 59    53 96 0B 19 AB DD 66 77   3e.Tc..YS.....fw
0010: 0B F2 5D 27                                          ..]'
]
]



*******************************************
*******************************************
```

%> keytool -export -alias QFB -keystore keystore -rfc -file QFB.cer

%> keytool -import -alias dukecert -file QFB.cer -keystore truststore.ks

```
Enter keystore password:
Re-enter new password:
Owner: CN=QFB, OU=lzu, O=lzu, L=lanzhou, ST=gansu, C=cn
Issuer: CN=QFB, OU=lzu, O=lzu, L=lanzhou, ST=gansu, C=cn
Serial number: 4ed4f509
Valid from: Sat Aug 24 00:19:16 CST 2013 until: Sat Aug 31 00:19:16 CST 2013
Certificate fingerprints:
        MD5:    7A:E6:F9:1F:EE:86:B5:EC:06:08:FF:93:D8:CF:32:F5
        SHA1: 16:8B:0E:58:EC:C1:75:F9:70:23:66:46:34:B2:FC:23:34:D6:15:0B
        SHA256:
08:96:81:8E:6D:FD:24:95:33:2D:D7:C4:DF:08:C6:B1:D4:50:73:6E:77:3D:D1:DE:B0:30:36:
4B:77:6F:00:AF
        Signature algorithm name: SHA256withRSA
        Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 6A 3C C8 82 B8 16 44 B8      2E 4C 8B 90 2E F4 4F D3    j<....D..L....O.
0010: EB C3 C2 EC                                             ....
]
]

Trust this certificate? [no]:    yes
Certificate was added to keystore
```

%> keytool -list -v -keystore truststore.ks

You will generated the “keystore.ks” and “truststore.ks”, and add them into the ssl folder of servers, nodes and clients. For example, the ssl folder is in “node31/config/ssl”.

Open ”config/ssl” folder of servers, nodes and clients. Edit their ssl.properties and modify all passwords corresponding to your passwords of “keystore.ks” and “truststore.ks”. The below is just an example for reference.

```
jppf.ssl.keystore.password = Your password when generate keystore.ks
jppf.ssl.truststore.file = config/ssl/truststore.ks
jppf.ssl.truststore.password = Your password when generate truststore.ks
jppf.ssl.truststore.password.source = org.jppf.ssl.PlainTextPassword Your password
```

Now the SSL setting is finished. Anyone who wants to connect to your server will need your password and certificate.

Distinct driver trust stores for nodes and clients

The above part gives the example for the generation of “keystore.ks” and “truststore.ks”. However, these certificates cannot distinguish connection between server-clients and server-nodes. In other words, if you obtain the node key, you can put it into the client key-folder, in this case, you can submit your job arbitrarily. It is not safe for the grid computing networking management. Different kinds of “keystore.ks” and “truststore.ks” can be generated to distinguish the node and client. The detail examples are introduced in the JPPF demo part:

http://svn.code.sf.net/p/jppf-project/code/branches/b_3_3/demo/config/ssl3/ and

http://www.jppf.org/doc/v3/index.php?title=Configuring_SSL/TLS_communications#Trust_store_and_associated_password

Now I give an instance to generate different kinds of certificates using command as below in the Linux environments:

```
export DNAME="CN=MolGridCal $1, OU=MolGridCal, O=MolGridCal, L=LZU, S=LZU, C=Gansu"
keytool -genkey -dname "$DNAME" -alias node -keypass password -keyalg RSA -validity 3650 -keystore node_keystore.ks -storepass password
keytool -genkey -dname "$DNAME" -alias client -keypass password -keyalg RSA -validity 3650 -keystore client_keystore.ks -storepass password
keytool -genkey -dname "$DNAME" -alias driver -keypass password -keyalg RSA -validity 3650 -keystore driver_keystore.ks -storepass password
keytool -genkey -dname "$DNAME" -alias client -keypass password -keyalg RSA -validity 3650 -keystore client_keystore.ks -storepass password
keytool -export -alias driver -keypass password -keystore driver_keystore.ks -storepass password -rfc -file driver.cer
keytool -export -alias node -keypass password -keystore node_keystore.ks -storepass password -rfc -file node.cer
keytool -export -alias client -keypass password -keystore client_keystore.ks -storepass password -rfc -file client.cer
keytool -import -alias driver -keypass password -file driver.cer -keystore node_truststore.ks -storepass password -noprompt
```

```
keytool -import -alias driver -keypass password -file driver.cer -keystore client_truststore.ks
-storepass password -noprompt
keytool -import -alias node -keypass password -file node.cer -keystore driver_truststore.ks
-storepass password -noprompt
keytool -import -alias client -keypass password -file client.cer -keystore
driver_client_truststore.ks -storepass password -noprompt
```

Configure JPPF-5.2.8-driver

1. Customize your own IP address, you should point out your server IP address as below:

```
jppf.server.host = Your ip adress
```

2. Make sure the SSL was uncommented in the config/jppf-driver.properties:

```
jppf.peer.ssl.enabled = true
jppf.ssl.configuration.file = config/ssl/ssl-server.properties
```

To avoid the unauthorized nodes to connect to server, close the ordinary port

```
#jppf.server.port = 11111
jppf.server.port = -1
```

3. You must write the right port for SSL connection, for example, the default node listens to the connection at port 11143, you should set the SSL port in server part as below:

```
jppf.ssl.server.port = 11143
```

More information is available at:

http://www.jppf.org/doc/v4/index.php?title=Configuring_SSL/TLS_communications

4. The algorithm of distributed tasks is as below:

```
jppf.load.balancing.algorithm = manual
strategy.manual.size = 1
```

If you want to start quickly, copy VStutorial/02-soft/driver31 to overwrite the JPPF-3.1-driver.

Configure JPPF-5.2.8-node

Make sure the SSL in “config/jppf-node.properties” is open:

```
jppf.ssl.enabled = true
```

Idle system

Option: if you want to use the screensaver function using the idle resource of computer, you can change the parameters of “config/jppf-node.properties” as below.

```
jppf.idle.mode.enabled = true
jppf.idle.detector.factory = org.jppf.example.idlesystem.IdleTimeDetectorFactoryImpl
jppf.idle.timeout = 6000
jppf.idle.poll.interval = 1000
```

If you want to start quickly, copy VStutorial/02-soft/node31 to overwrite the JPPF-3.1-node. And copy the vina program into the 02-soft/node31.

Start MolGridCal

Until now, all configures are finished. Let us start the work of MolGridCal

1. If you generate your certificate, please replace “keystore.ks” and “truststore.ks” into “config/ssl”. The detailed method has been introduced above.

2. Copy the file of β_2AR, “executed program VINA”, “conf.txt” into the directory of **nodes** directly. In this version of MolGridCal, the name of “conf.txt” is absolute. Please do not modify it to other name, or it will be wrong.

3. Create the file “parameter.mgc” in the directory of MolGridCal:

parameter.mgc

```
Ipaddress              2001:da8:c000:201:f66d:4ff:fe21:10e7
IpPort                 2121
User                   gridfvs
Password               molgridcal
Downloaddir            /VS1/ligands
UploadDir              /VS1/result
MaxNodes               10000

# The line show program run as below
Token                  ZINC
Program                Autodock_Vina
Command                vina
```

Ipaddress: Your IP address of FTP server. Now it supports IPv6 address (see above).

IpPort: The port of FTP server

User: the login name of FTP server

Password: the login password of FTP server

Downloaddir: The directory of ligands for virtual screening

UploadDir: The gathered directory of docking results

MaxNodes: The maximum number of loading files in memory at one time.

Token: To delete the molecules easily

Program: choose program for virtual screening

Command: choose command for executing the virtual screening

The parameters are introduced above, now run the ant command orderly:

1) Start the server, change into the directory of server: “cd driver421”

%> ant

2) Start the nodes, change into the directory of nodes: “cd node421”

%> ant

3) Run MolGridCal, change into the directory of MolGridCal: “cd MolGridCal”

%> chmod +x vina autodock4

%> ant

All the ant command can run in the background of linux if you do not want to display them in the console:

ant >& ant.log &

But you must remember the run ID (see the id:18118 of Figure 4), if you do not want to use it, just kill it. Or use command: “ps -ux”, and find the running ID. Then input: “kill ID”. In this example, input command: kill 18118

Now the work of nodes started as shown in the below figure:

```
[java] node process id: 18118
[java] Attempting connection to the class server at 202.201.5.90:11443
[java] Reconnected to the class server
[java] JPPF Node management initialized
[java] Attempting connection to the node server at 202.201.5.90:11443
[java] Reconnected to the node server
[java] Node successfully initialized
[java] **************Working started***************
[java] ########################################################################
[java] # If you used MolGridCal in your work, please cite:                    #
[java] #                                                                      #
[java] # Qifeng Bai, MolGridCal: A Molecular Grid Calculation program on      #
[java] # basis of JPPF, Version x.xx, http://molgridcal.codeplex.com          #
[java] #                                                                      #
[java] # I sincerely hope users can send me their accepted paper to the       #
[java] # E-mail (molgridcal@yeah.net), we will list them in our official      #
[java] # blog website (http://molgridcal.blog.163.com).                       #
[java] #                                                                      #
[java] # In order to discuss the MolGridCal, please join the group:          #
[java] # http://groups.google.com/group/molgridcal                           #
[java] ########################################################################
[java] Connect to the server for working!
[java] Now the Autodock_Vina running*********
```

Figure 4. The screening output of MolGridCal in the nodes.

Now all the works are finished. You could have a coffee for a rest.

Rank the docking results

When all the results were collected, it need get the good score of ligands. It was easy to rank the docking results using scripts or program language. Here, we refer the rank program of PYTHON which refers in the *http://vina.scripps.edu/manual.html*

vina_screen_get_top.py

```
#! /usr/bin/env python

import sys
import glob

def doit(n):
    file_names = glob.glob('*/*.pdbqt')
    everything = []
    failures = []
    print 'Found', len(file_names), 'pdbqt files'
    for file_name in file_names:
        file = open(file_name)
        lines = file.readlines()
        file.close()
        try:
            line = lines[1]
            result = float(line.split(':')[1].split()[0])
            everything.append([result, file_name])
        except:
            failures.append(file_name)
    everything.sort(lambda x,y: cmp(x[0], y[0]))
    part = everything[:n]
    for p in part:
        print p[1]
    print
    if len(failures) > 0:
        print 'WARNING:', len(failures), 'pdbqt files could not be processed'

if __name__ == '__main__':
    doit(int(sys.argv[1]))
```

Usage: ./vina_screen_get_top.py “num”

num is the number waht you want to rank

For example, get the top 600 of docking results:

%> chmod +x vina_screen_get_top.py

%> ./vina_screen_get_top.py 600 > rank600.log

Reference:

1. Geddes, N., The Large Hadron Collider and Grid computing. *Philos Trans A Math Phys Eng Sci* **2012**, 370, 965-77.

2. Mareuil, F.; Blanchet, C.; Malliavin, T. E.; Nilges, M., Grid computing for improving conformational sampling in NMR structure calculation. *Bioinformatics* **2011**, 27, 1713-4.

3. Kayser, K.; Gortler, J.; Borkenfeld, S.; Kayser, G., Grid computing in image analysis. *Diagn Pathol* **2011**, 6 Suppl 1, S12.

4. Yim, W. W.; Chien, S.; Kusumoto, Y.; Date, S.; Haga, J., Grid heterogeneity in in-silico experiments: an exploration of drug screening using DOCK on cloud environments. *Stud Health Technol Inform* **2010**, 159, 181-90.

5. Irwin, J. J.; Sterling, T.; Mysinger, M. M.; Bolstad, E. S.; Coleman, R. G., ZINC: A Free Tool to Discover Chemistry for Biology. *J Chem Inf Model* **2012**.