Parallel Protein Structures Comparison PPSC

Tool for parallel execution of protein structure comparison and alignment programs

Hunan University
Ahmad Salah & Kenli Li

Table of Contents

1. About This Document

1.1 Intended audience

This document is designed for users with basic computer OS and network knowledge. The software in hand addresses the problem of running sequential programs that search for protein structure similarity in parallel environments. Users can use this tool on a standalone computer with multi-core CPU(s), or on LAN of computers. The software is available at http://aca.hnu.cn/ppsc/ .

1.2 System requirements

A single computer or a LAN of computers (which consists of a master node to and a set of server nodes) with the following installed on each:

- Linux/ Windows OS 32bit or 64-bit (openSUSE, Ubuntu, Fedora, CentOS and Windows 7 are tested).
- Java runtime.
- The other software requirements of the sequential program.

1.3 Brief Definition of the Addressed Problem

Protein structure comparison is a process of comparing a query protein structure, described in its 3D form, against a set of protein 3D structures database. The process is done on the element-wise comparison base. Given a database of N elements and a query, the database can be divided into P portions to fit the multi-core or multi server nodes. Each core and/or server node performs the element-wise comparison independently. Finally, the output of all the comparisons are merged and sorted by the score. The computation node manages the process is called the master node while any computational node does the actual comparison is called the server nodes. The system works on a LAN with one master node and at least one server node.

2. Software Components

2.1 The sequential program

Here, user should be able to successfully run the protein structure comparison program on all the computers the program should be run on. PPSC tool considers that the sequential program is running without any problems on the host computers. This is the sequential version which do the actual comparison work.

2.2 Input Divider (Load Balancer)

User uses this component to balance the input files to the heterogeneous and/or homogeneous server nodes. User should input the following:

- A description of all used computational nodes; this can be saved and loaded for the next usage.
- The database directory.
- The database file which contains path of each database file.
- The output path for the descriptor file of the balanced load.

2.3 Interactive input list Creator

This component is used when the sequential program requires interaction during the course of its execution. Since there are many instance of the sequential program are running simultaneously and may be remotely, user can interact with those instance. The purpose of this tool is to generate a unified list of interactive inputs, and then this list will be used to feed the input to the sequential program as required.

2.4 Multi-core version

This version can be used to run the tool on a standalone computer with a multi-core CPU(s).

2.5 LAN version

This version can be used to run the tool on LAN of a master node and a set of server nodes. It contains two components one to distribute and manage the server nodes, and another component to do the comparison work.

2.5.1 Master Node

This component is the key component; user will interact to this tool most of the time. User should input the following:

- The IP address of the master node (127.0.0.1 for multi-core use).
- A description of all used computational nodes; this can be saved and loaded for the next usage.

- The command to run sequential program. User can tune all of the available parameters of the sequential program.
- The output of the load balancer component. Input divider component can be launched from master node component or user can select the descriptor file of the balanced load.
- The query directory and the query file which contains path of each query file.
- The output of the merged comparison results.

User can use this component to distribute the balanced load to the server nodes, re-run queries or run a new query.

2.5.2 Server Node

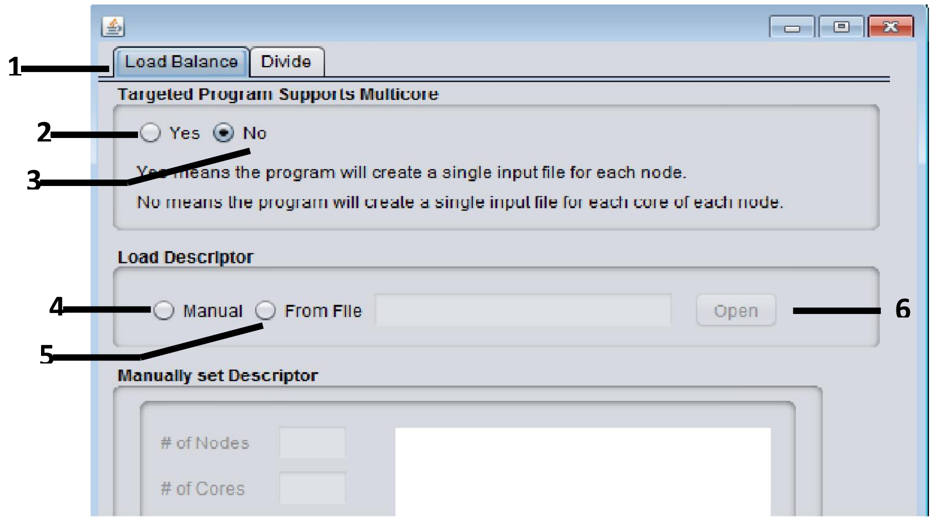
This component does the actual work of the protein structure comparison. It receives the portion of the database files and save them for future queries. Currently, it can manage to save only one database so user has no selection regarding the database. User must select the IP of that server node; this IP will be recognized by the master node.

3. Input Divider (Load Balancer)

3.1 Server nodes/standalone computational information

Input divider can be used as a standalone component or invoked from the Master Node component. The ultimate goal of this tool is to divide the database files in balance manner to fit the computational node powers. User can divide the database file into portions and then save the load balance descriptor file to be used by the master node. Otherwise user can instantly divide the input database by calling the input divider component from the master node component.

Figure 3-1 Nodes descriptor method shows the options to set the computational nodes information which are manual or automatically from a file.

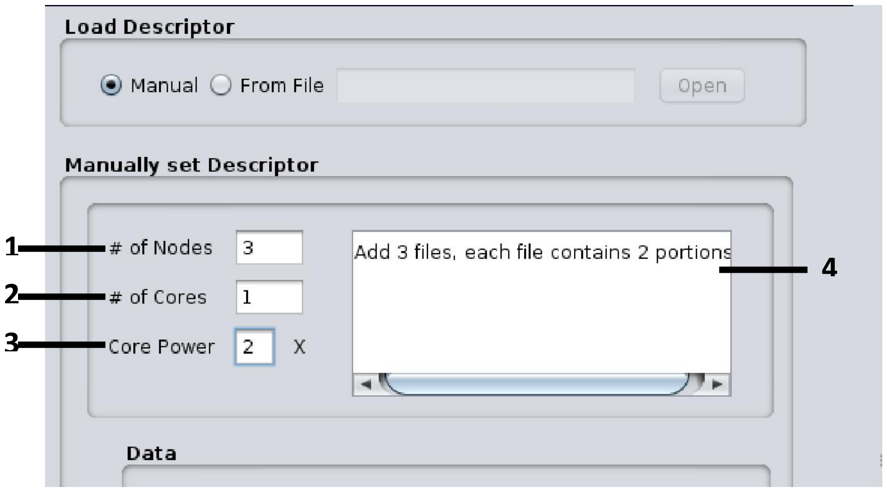


Yes means the program will create a single input file for each node.
No means the program will create a single input file for each core of each node.

Figure 3-1 Nodes descriptor method

1- Load balance tab.
2- Selecting this option means the target program supports the multicore option, which will be reflected into creating one batch file for each workstation.
3- Selecting this option means the target program does not support the multicore option, which will be reflected into creating one batch file for core of each computer.
4- Manual input for the computational nodes power.
5- Automatic load of the computational nodes power from a file.
6- Open nodes descriptor file button.

Figure 3-2 Node information input shows the input of computational nodes information. The text area to the right outlines the effect of this information on the division process.



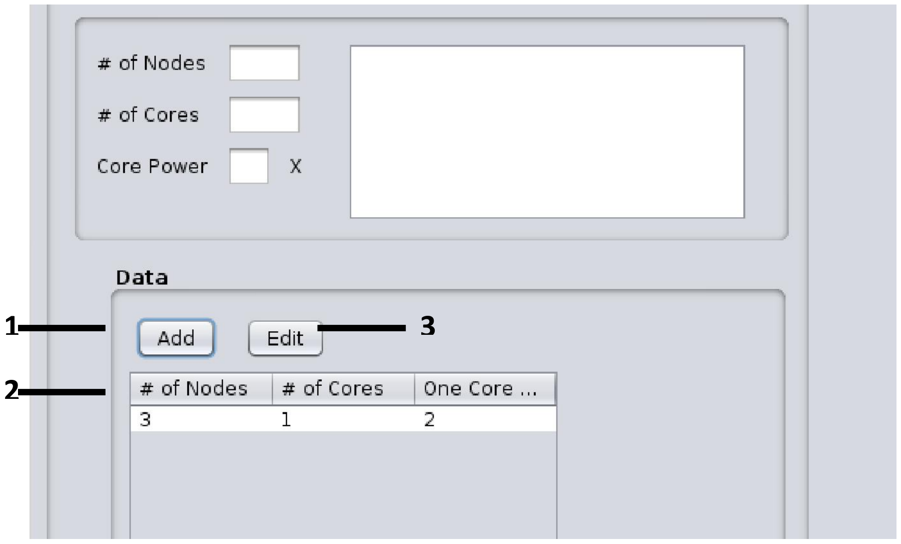
# of Nodes: 3
# of Cores: 1
Core Power: 2 X

Add 3 files, each file contains 2 portions

Figure 3-2 Node information input

1- Number of nodes of the same computational power.
2- Number of available cores per each.
3- Core power in comparison with other computational node (must be +ve integer).
4- The effect of this entry on the division process.

Figure 3-3 Nodes information displaying and editing shows the panel which displays and edits the nodes information. Add button forwards all the inputs to the table and reset the inputs text fields. Button edit delete the selected row from the table and forwards the selected row information to text fields to be edited.



# of Nodes	# of Cores	One Core ...
3	1	2

Figure 3-3 Nodes information displaying and editing

1- Add the current input to the information table.
2- The all nodes information table.
3- Edit the selected row nodes information.

3.2 Database information and division

Figure 3-4 Database and output tab shows the process of selecting the database directory and file. This figure also shows the operation of selecting the output directory of the division process. User can start the division and track the progress and errors during this process.

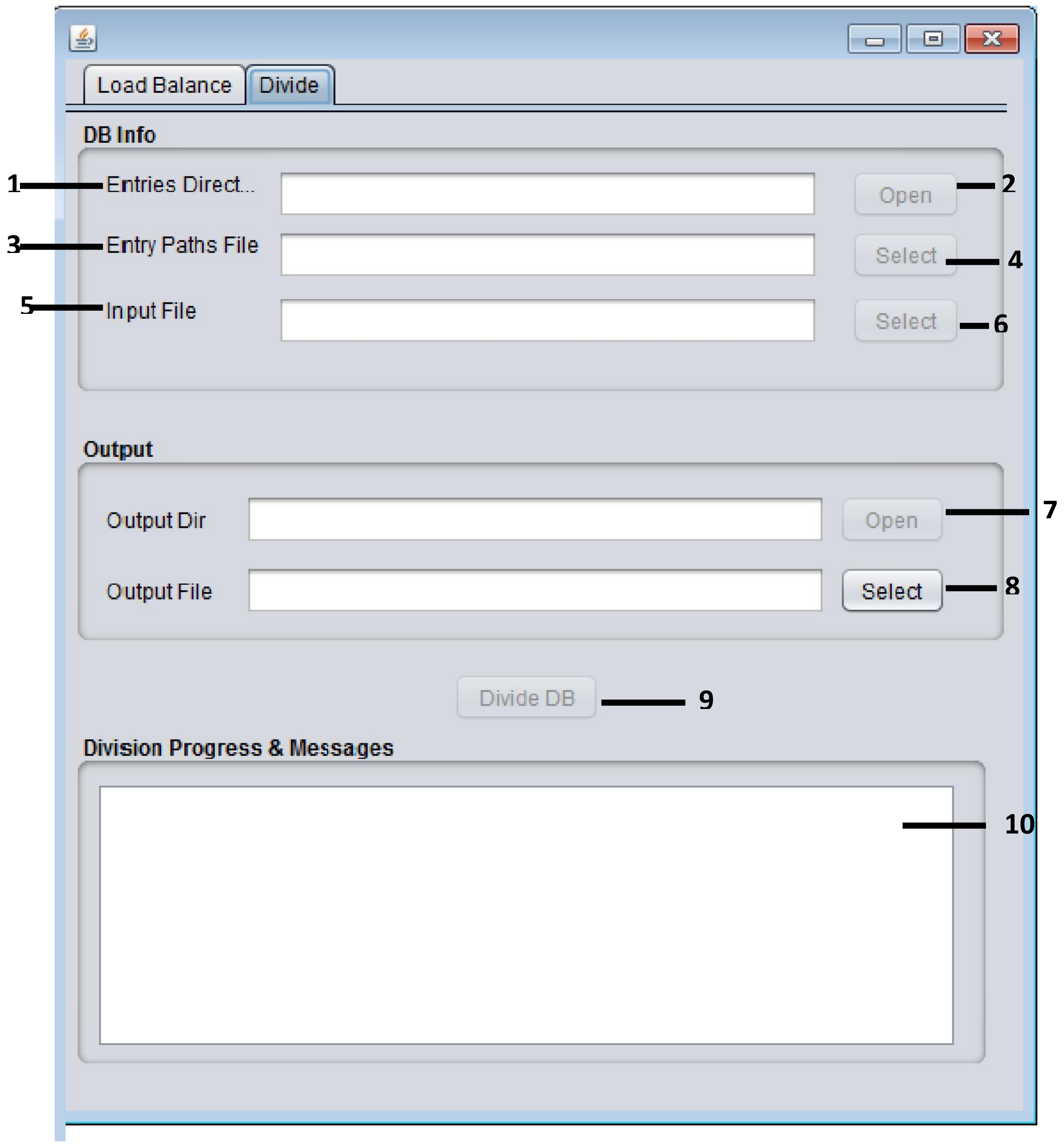


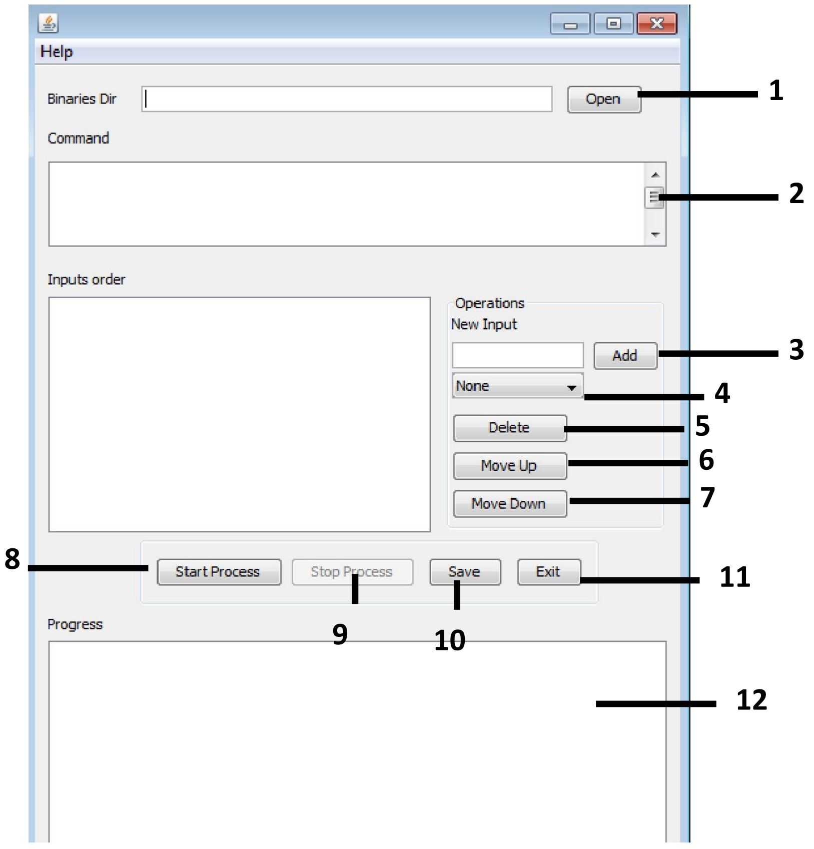
Figure 3-4 Database and output tab

1- Input database directory for the division process.
2- Open button to open the directory of the database.
3- Input file that includes one absolute path for one database entry at a line. Number of lines in this file must equal the number of the database entries.
4- Open button to select the file of the database entry paths.
5- File path for the database information; it is used in case the original program requires such input.
6- Open button to select the file of the database information.
7- Output directory of database portions. This directory is high recommended to be empty. It will contain files which describes the database portions.
8- Open button to choose the output of the portion descriptor files. **You must remember this input**. This input contains all the information required by the other component to distribute the data. The file should be the input to the next phases.
9- This button will start the division process.
10- This text area shows the progress of the division process. **You must be notified for the success of the division process by a message box.**

4. Interactive inputs list creator

This component is used when the sequential program requires interaction during the course of its execution. Since there are many instance of the sequential program are running simultaneously and may be remotely, user can interact with those instance. The purpose of this tool is to generate a unified list of interactive inputs, and then this list will be used to feed the input to the sequential program as required.

4-1 Interactive inputs list creator shows the component that can be used to create the list of the interactive inputs that will be used by the next component to run the interactive sequential program.



4-1 Interactive inputs list creator

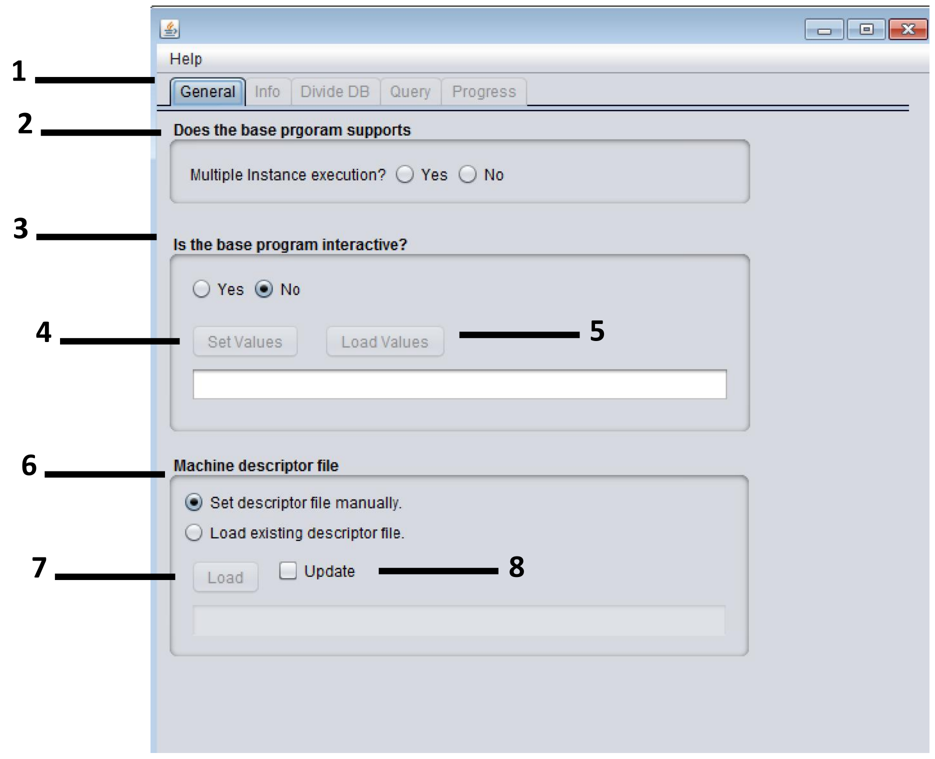
1- Open button, a button to select the directory of the sequential program binaries.
2- Text area to write the command in order to run the sequential program.
3- Textbox to write the input value and “Add” button to insert that input to the list “inputs order” on the left.
4- A dropdown list of the keywords.
5- A button deletes a selected input from the list to the left.
6- A button moves up a selected input from the list to the left.

7- A button moves down a selected input from the list to the left.
8- A button starts the process and to test the inputs.
9- A button stops the process for any reason, e.g., the inputs order is not correct. Usually the interactive programs hang, waiting for inputs, in case the inputs are in incomplete or disordered.
10- A button saves the interactive list to a file for next usage.
11- Exit button.
12- Text area shows the progress of running the interactive program.

5. Standalone multi-core version

5.1 Setting the general information

Figure 5-1 General information shows the general information user can provide before starting the tool. Optionally, user should load a file describing the computational nodes.



Does the base prgoram supports

Multiple Instance execution? Yes No

Is the base program interactive?

Yes [✓ No]

Machine descriptor file

[✓ Set descriptor file manually.]
Load existing descriptor file.

Figure 5-1 General information

1- Show an active General tab, this tab contains the basic information about the sequential program.
2- User should answer this question. If multiple script files can simultaneously use the executable files of the sequential program then user should select yes; otherwise user should select no.
3- User should answer this question. If the sequential program demands inputs during the running process, interactive, then user should select yes; otherwise user should select no.
4- If user selected yes in 3, then user should click "set values" button to create the interaction list; using the interactive inputs list tool.
5- If user selected no in 3 and the user already has the list of the interactive list, then user should click the "load values" button and select this list file.
6- User should answer this question; will the user set the standalone machine manually? User usually uses this option at the first execution, and the next times user can easily load

the automatically saved file or will the user load a file which contains all the information of the standalone machine?

7- Load button to select the descriptor file. It must be xml and it is saved from the last execution. A format check will be done to make sure it is the right descriptor file.
8- Update check box means after loading the file if any changes happened then it should be updated in that file.

5.2 Setting standalone machine information

Figure 5-2 sequential program info shows the required inputs required to manage adding the information of the server nodes.

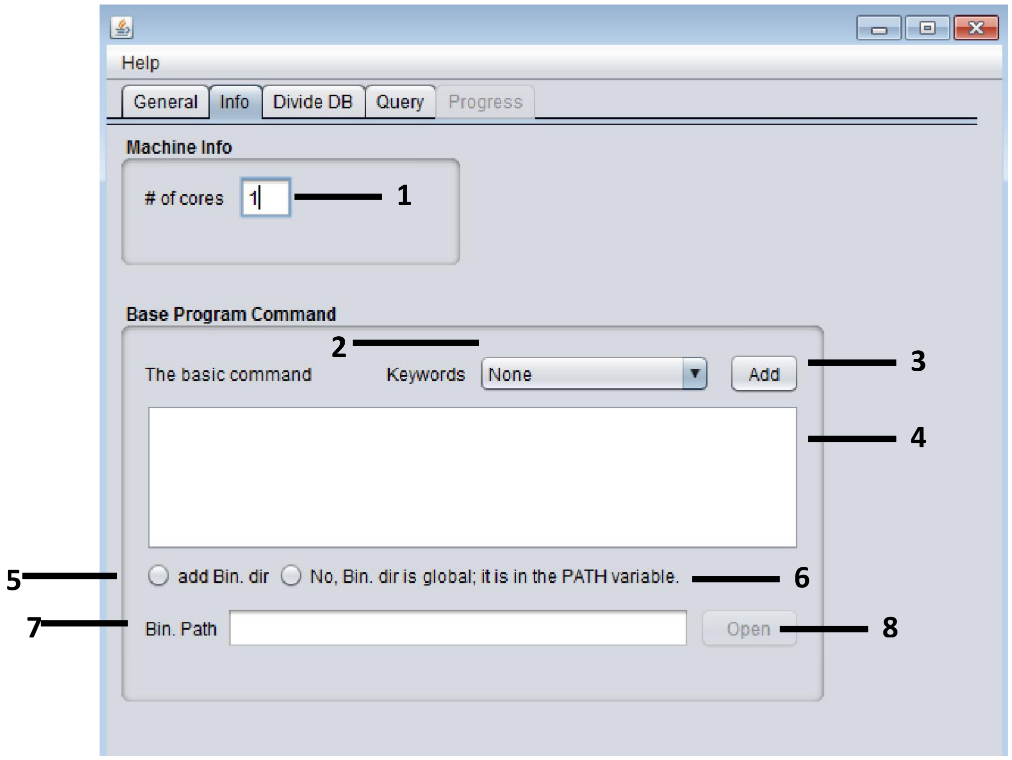


Figure 5-2 sequential program information

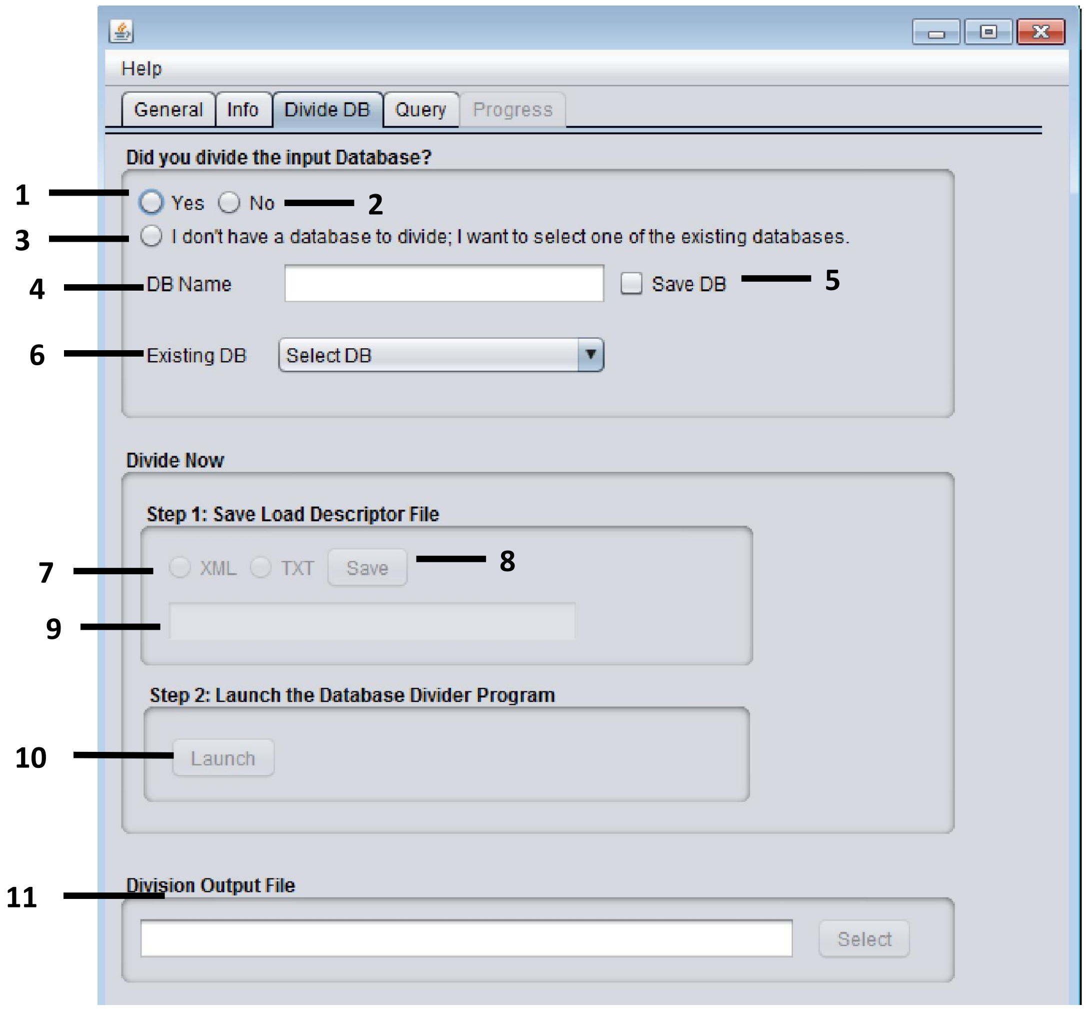
1- Number of cores to be used (number of instance of the sequential program).
2- Set of keywords to be used in the sequential program command.
3- Add button to add the selected keyword to the command. Table 5-1 keyword meanings a list of lists of the keywords and its meanings.
4- Sequential program command text. Each command should be ended by \newline.
5- Set the path of the sequential program binaries directory.
6- If the sequential program is installed globally and can be access from any directory.
7- If option 5 is selected then user must fill in this text field. If option 6 is selected then user should not fill in this text field.
8- Open button, a button to select the binaries path.

Table 5-1 keyword meanings

Keyword	Meaning
%<c>%	This keyword will be replaced by the number of cores.
%<serial>%	This keyword will be replaced by a number depends on the order of creation of the script files in the same directory. If the program has 5 script files in the same directory then each script will replace this keyword by a corresponding number.
%<dbdir>%	This keyword will be replaced by the database directory.
%<dbfile>%	This file contains a list of the database file; that keyword should appears after the –d option or –o option in make database.
%<querydir>%	This keyword will be replaced by the query directory.
%<queryfile>%	This key will be replaced by the query file.
%<eachquerypath>%	This keyword will be replaced by query entry path for each query belongs to each script file.
%<eachqueryname>%	This keyword will be replaced by the query file name for each query belongs to each script file.
%<queryfileline>%	This keyword will be replaced by a line from the %<queryfile>% belongs to each script file.
%<eachdbpath>%	This keyword will be replaced by database entry path for each database entry belongs to each script file.
%<eachdbname>%	This keyword will be replaced by the database file name for each database entry belongs to each script file.
%<outpath>%	This keyword will be replaced by the output directory.
\newline	Command separator

5.3 Dividing the database

Figure 5-3 Divide database asks the user about whether there is a divided database, if there is no database has been divide or user want to use the previously database used. If the database already divided then user need to locate the output file of the input divider component using 8. If user has no divide database then user must go through 4, 5, 7 and 8. In case user has no database then user can escape this tab and to the next tab (Query Info).

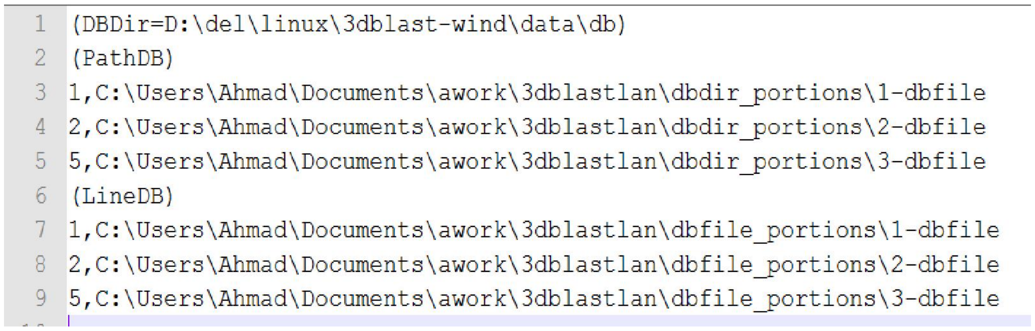


Help
General
Info
Divide DB
Query
Progress
Did you divide the input Database?
1
Yes
No
2
3
I don't have a database to divide; I want to select one of the existing databases.
4
DB Name
Save DB
5
6
Existing DB
Select DB
Divide Now
Step 1: Save Load Descriptor File
7
XML
TXT
Save
8
9
Step 2: Launch the Database Divider Program
10
Launch
11
Division Output File
Select

Figure 5-3 Divide database

1- This option means that the user already had divided the database. User should go to 11 to select the "database load division" descriptor file which is created by the input division component.
2- This option means user will launch the input divider component to divide the database. User should go to steps 7 to 11.
3- This option means the user want to use the database already in the server side. The user had no database. User should select a database from the database dropdown box in step 6 and then go the next tab (Query).
4- User should input the database name; the database name should be unique, different from all database names in the dropdown list of step 6.
5- If user wants to use this database for once then user should uncheck this check box; otherwise user should check this checkbox to be able to use this database for next times.
6- A dropdown box of all existing saved database names.
7- User saves the machine descriptor file as XML file or text file; this file will be the input to Input divider component which accepts text or xml files.

8- Save button, that button selects the output file to be saved to.
9- Text field to show the output path of the loads descriptor file.
10- Input divider component launcher button. User should divide the database using the input divider component and then go to 11 to set the input divider output file.
11- User use “Select” button to locate the output of the input divider button. A format check will be done to make sure that the descriptor file is in the right format. Figure 5-4 database division file descriptor format shows a sample of the accepted format. The format must starts with the database directory. User must follow the capital and small letters. Then at line 2, a PathDB tag is added to start list of file which includes the absolute paths of the database entry. At line 3 to 5, the file path is preceding by a number which presents the number of loads of the databases and a comma separator. The total number of loads is 8 distributed over 3 processes each of different computational power.



```
1 (DBDir=D:\del\linux\3dblast-wind\data\db)
2 (PathDB)
3 1,C:\Users\Ahmad\Documents\awork\3dblastlan\dbdir_portions\1-dbfile
4 2,C:\Users\Ahmad\Documents\awork\3dblastlan\dbdir_portions\2-dbfile
5 5,C:\Users\Ahmad\Documents\awork\3dblastlan\dbdir_portions\3-dbfile
6 (LineDB)
7 1,C:\Users\Ahmad\Documents\awork\3dblastlan\dbfile_portions\1-dbfile
8 2,C:\Users\Ahmad\Documents\awork\3dblastlan\dbfile_portions\2-dbfile
9 5,C:\Users\Ahmad\Documents\awork\3dblastlan\dbfile_portions\3-dbfile
```

Figure 5-4 database division file descriptor format

5.4 Setting query information

Figure 5-5 Query information shows a tab for selecting the query and the final output directories. After doing so, user should start the sequential program and wait for the results.

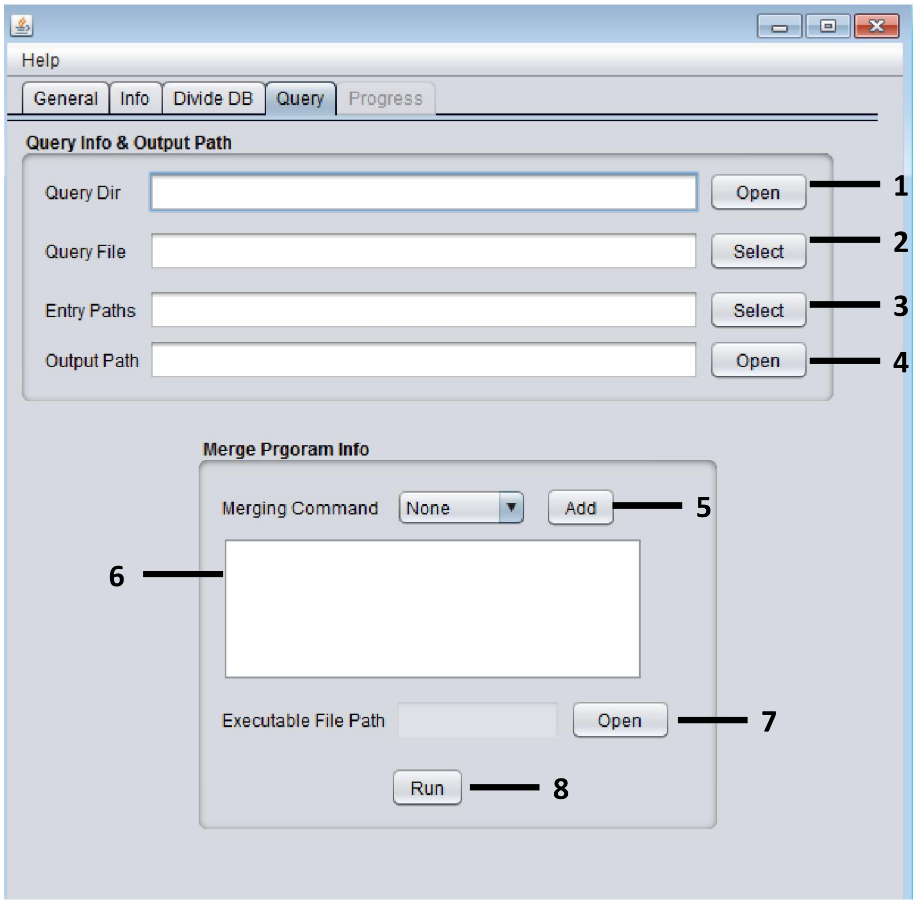
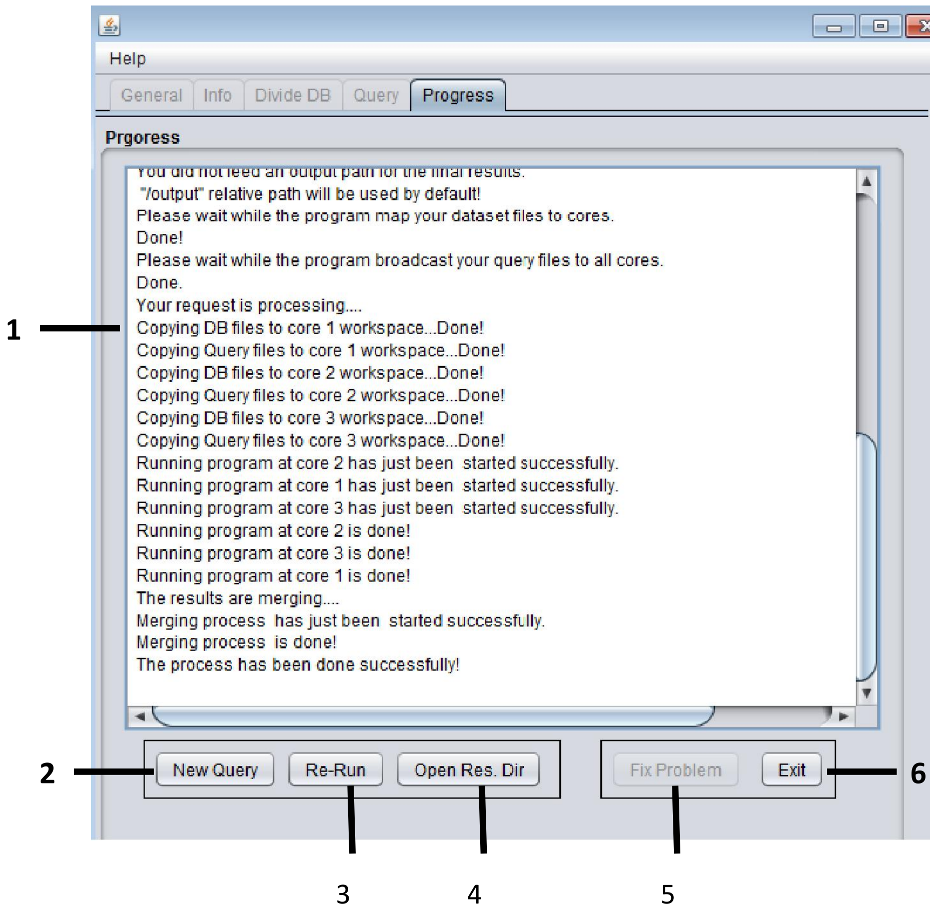


Figure 5-5 Query information

1- Open button, a button to select the query directory.
2- Select button, a button to select the query file; a file required by the original program.
3- Select button, a button to select the file that includes all the query entry paths; a file contains a query entry absolute path at a line. The number of lines of this file equals the number of queries.
4- Open button, a button to select the output of the final results.
5- Add button, to add a keyword of the merge command.
6- Text to write down the merge command.
7- Open button, a button to select the directory of the merge program binaries.
8- Start the sequential program.

5.5 Tracking execution, submitting new queries and re-run

User can track the execution of the sequential program from the progress tab as shown in Figure 5-6 Progress, re-run and new tab. This tab also enables user to submit a new query, re-run the query, fix any problem indicated in the progress text area, and explore the results by opening the output directory.



```
You did not feed an output path for the final results.
 "/output" relative path will be used by default!
Please wait while the program map your dataset files to cores.
Done!
Please wait while the program broadcast your query files to all cores.
Done.
Your request is processing....
Copying DB files to core 1 workspace...Done!
Copying Query files to core 1 workspace...Done!
Copying DB files to core 2 workspace...Done!
Copying Query files to core 2 workspace...Done!
Copying DB files to core 3 workspace...Done!
Copying Query files to core 3 workspace...Done!
Running program at core 2 has just been  started successfully.
Running program at core 1 has just been  started successfully.
Running program at core 3 has just been  started successfully.
Running program at core 2 is done!
Running program at core 3 is done!
Running program at core 1 is done!
The results are merging....
Merging process  has just been  started successfully.
Merging process  is done!
The process has been done successfully!
```

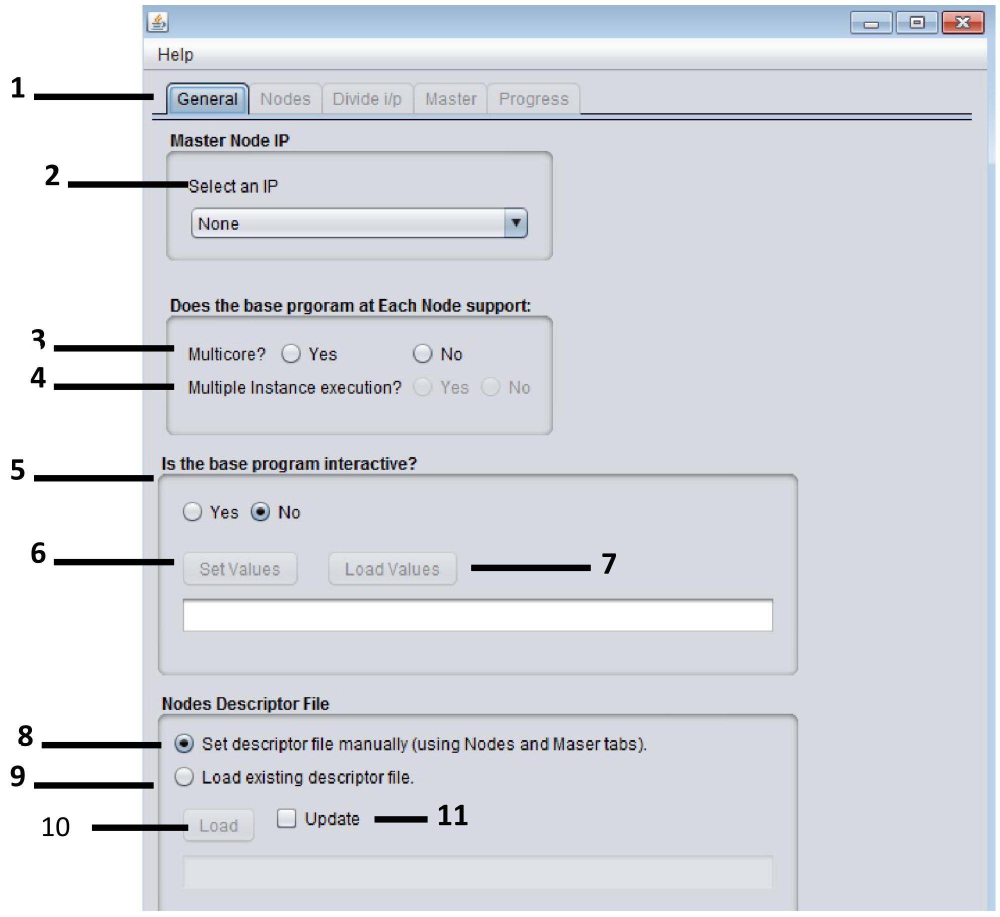
Figure 5-6 Progress, re-run and new tab

1- Text area for the messages of the sequential program executions.
2- New query button, a button submits a new query.
3- Re-run query button, a button re-runs the last run query.
4- Open Res dir, a button opens the results directory.
5- Fix Problem button, a button takes user to the master node tab to fix the indicated problem at the progress text area 1.
6- Exit button, a button exits the execution terminate the server and master nodes.

6. Launching the Master Node

6.1 Setting the general information

Figure 6-1 General information shows the general information user should provide before starting the tool. User must set the IP of the master node. Optionally, user should load a file describing the computational nodes.



Master Node IP

Select an IP: None

Does the base prgoram at Each Node support:

Multicore? Yes No

Multiple Instance execution? Yes No

Is the base program interactive?

Yes [✓ No]

Nodes Descriptor File

[✓ Set descriptor file manually (using Nodes and Maser tabs).]

Load existing descriptor file.

Update

6-1 General information

1- Show an active General tab; user should answer general question to proceed and activate the other tabs.
2- The first part is to select the IP of the master node. User can select 127.0.0.1 for standalone execution including the multi-core. If user work through a LAN then user must select an IP belongs to this LAN.
3- User should answer this question. If the sequential program supports multicore or not.
4- User should answer this question. If multiple script files can simultaneously use the executable files of the sequential program then user should select yes; otherwise user should select no.

5- User should answer this question. If the sequential program demands inputs during the running process, interactive, then user should select yes; otherwise user should select no.
6- If user selected yes in 3, then user should click “set values” button to create the interaction list; using the interactive inputs list tool.
7- If user selected no in 3 and the user already has the list of the interactive list, then user should click the “load values” button and select this list file.
8- User will set the server nodes information manually. User usually uses this option at the first execution, and the next times user can easily load the automatically saved file.
9- User will load a file which contains all the information of the computational nodes.
10- Load button to select the descriptor file. It must be xml and it is saved from the last execution. A format check will be done to make sure it is the right descriptor file.
11- Update check box means after loading the file if any changes happened then it should be updated in that file.

6.2 Setting server nodes information

Figure 6-2 Server nodes Information shows the required inputs required to manage adding the information of the server nodes.

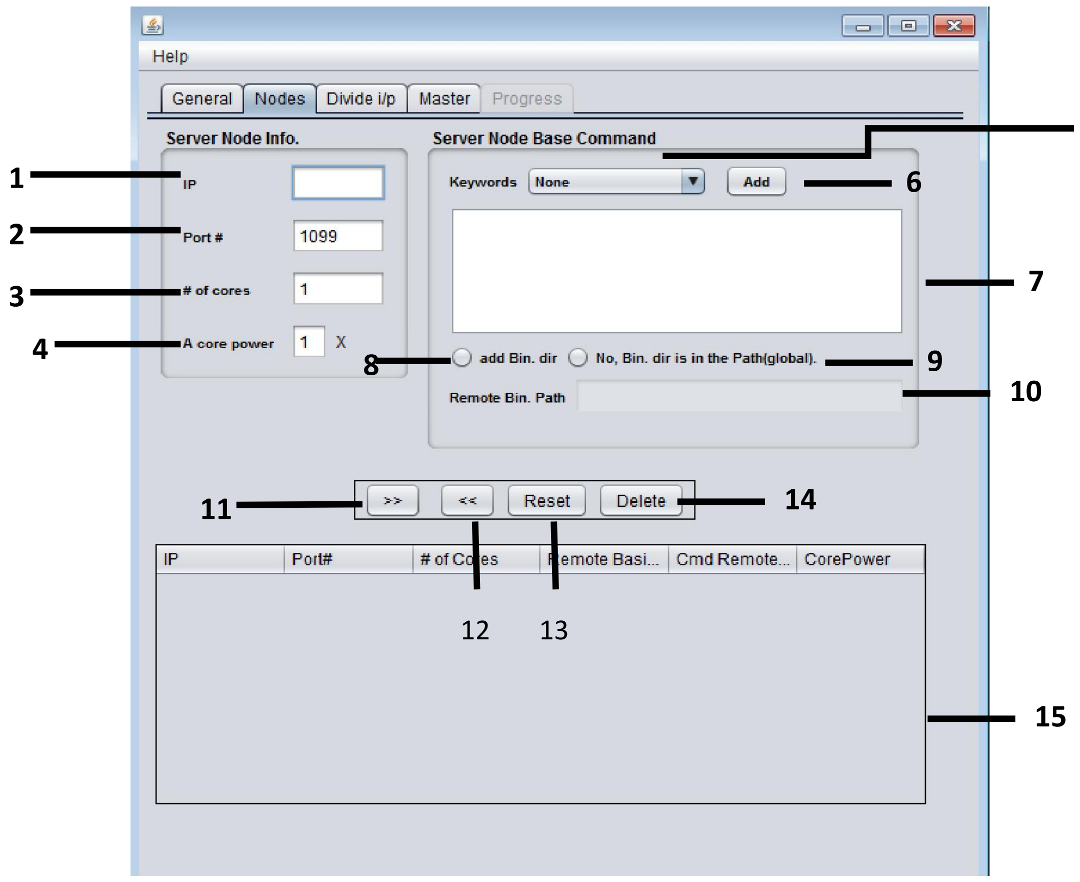


Figure 6-2 Server nodes Information

1- Server node IP.
2- Server node port #; it is recommended to use the default port# 1099.
3- Number of cores at the server node.
4- Core power in comparison with other computational node (must be +ve integer).
5- Set of keywords to be used in the remote command.
6- Add button to add the selected keyword to the command.
7- Sequential program command text. Each command should be ended by \newline.
8- Set the bath of the sequential program binaries directory at the remote path.
9- If the sequential program binaries in installed globally and can be access from any directory at the server node.
10- If option 8 is selected then user must fill in this text field. If option 9 is selected then user should not fill in this text field.
11- Add button, this button add the server node information to the table. It forwards the inputs to the last row of the table.
12- Edit button, this button remove the selected row from the table and forward information to the input text fields.
13- Reset button, this button clear all the text fields.
14- Delete button, this button delete the selected row from the table.
15- Information table contains all the user inputs for the server nodes.

6.3 Dividing the database

Figure 5-3 Divide database asks the user about whether there is a divided database, if there is no database has been divide or user want to use the previously database used. If the database already divided then user need to locate the output file of the input divider component using 8. If user has no divide database then user must go through 4, 5, 7 and 8. In case user has no database then user can escape this tab and to the next tab (Query Info).

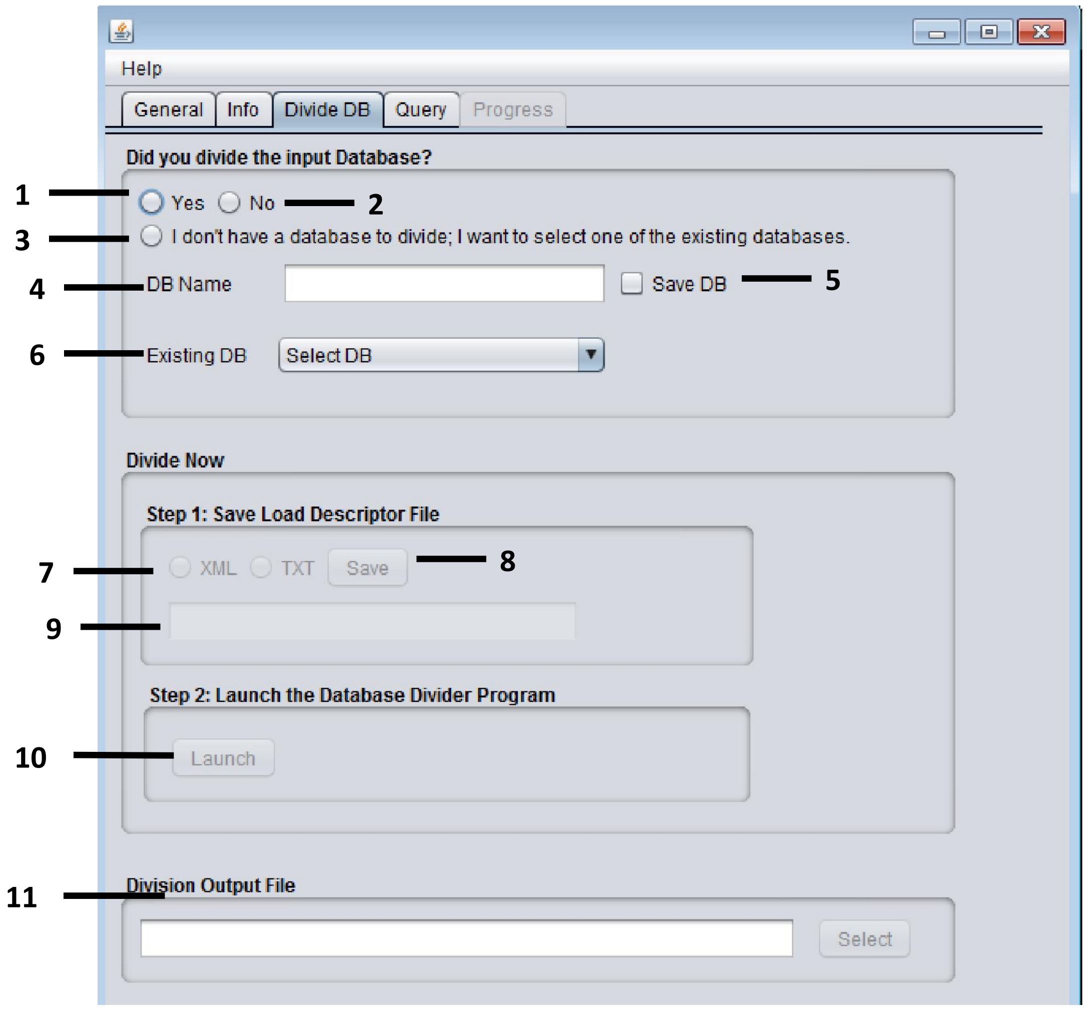
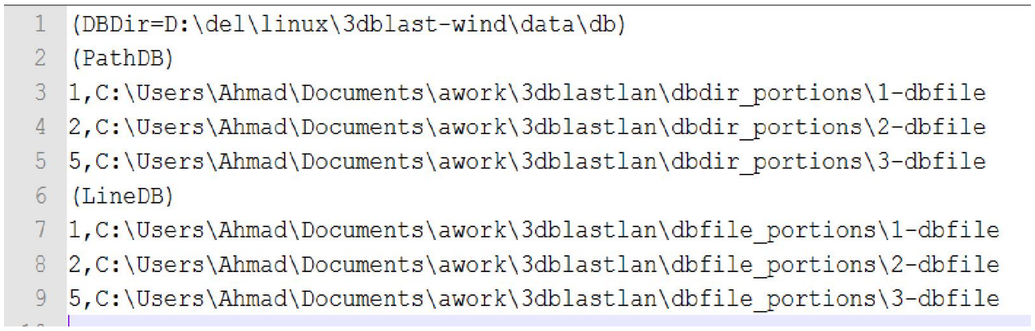

Help
General
Info
Divide DB
Query
Progress
Did you divide the input Database?
1
Yes
No
2
3
I don't have a database to divide; I want to select one of the existing databases.
4
DB Name
Save DB
5
6
Existing DB
Select DB
Divide Now
Step 1: Save Load Descriptor File
7
XML
TXT
Save
8
9
Step 2: Launch the Database Divider Program
10
Launch
11
Division Output File
Select

Figure 6-3 Divide database

1- This option means that the user already had divided the database. User should go to 11 to select the "database load division" descriptor file which is created by the input division component.
2- This option means user will launch the input divider component to divide the database. User should go to steps 7 to 11.
3- This option means the user want to use the database already in the server side. The user had no database. User should select a database from the database dropdown box in step 6 and then go the next tab (Query).
4- User should input the database name; the database name should be unique, different from all database names in the dropdown list of step 6.
5- If user wants to use this database for once then user should uncheck this check box; otherwise user should check this checkbox to be able to use this database for next times.
6- A dropdown box of all existing saved database names.
7- User saves the machine descriptor file as XML file or text file; this file will be the input to Input divider component which accepts text or xml files.

8- Save button, that button selects the output file to be saved to.
9- Text field to show the output path of the loads descriptor file.
10- Input divider component launcher button. User should divide the database using the input divider component and then go to 11 to set the input divider output file.
11- User use "Select" button to locate the output of the input divider button. A format check will be done to make sure that the descriptor file is in the right format. Figure 5-4 database division file descriptor format shows a sample of the accepted format. The format must starts with the database directory. User must follow the capital and small letters. Then at line 2, a PathDB tag is added to start list of file which includes the absolute paths of the database entry. At line 3 to 5, the file path is preceding by a number which presents the number of loads of the databases and a comma separator. The total number of loads is 8 distributed over 3 processes each of different computational power.



```
1 (DBDir=D:\del\linux\3dblast-wind\data\db)
2 (PathDB)
3 1,C:\Users\Ahmad\Documents\awork\3dblastlan\dbdir_portions\1-dbfile
4 2,C:\Users\Ahmad\Documents\awork\3dblastlan\dbdir_portions\2-dbfile
5 5,C:\Users\Ahmad\Documents\awork\3dblastlan\dbdir_portions\3-dbfile
6 (LineDB)
7 1,C:\Users\Ahmad\Documents\awork\3dblastlan\dbfile_portions\1-dbfile
8 2,C:\Users\Ahmad\Documents\awork\3dblastlan\dbfile_portions\2-dbfile
9 5,C:\Users\Ahmad\Documents\awork\3dblastlan\dbfile_portions\3-dbfile
```

Figure 6-4 database division file descriptor format

6.4 Setting master node and query information

Figure 6-5 Query information depicts the process of feeding the query information and executing the sequential program. User can distribute the database without having any query by selecting 8 the press button 10 "Start Maser Node". This action will result in distributing the database for all the involved server nodes. User can use option 9 in case there is no database and just want to run a query. In case the user has a database and query, user can select option 9 also.

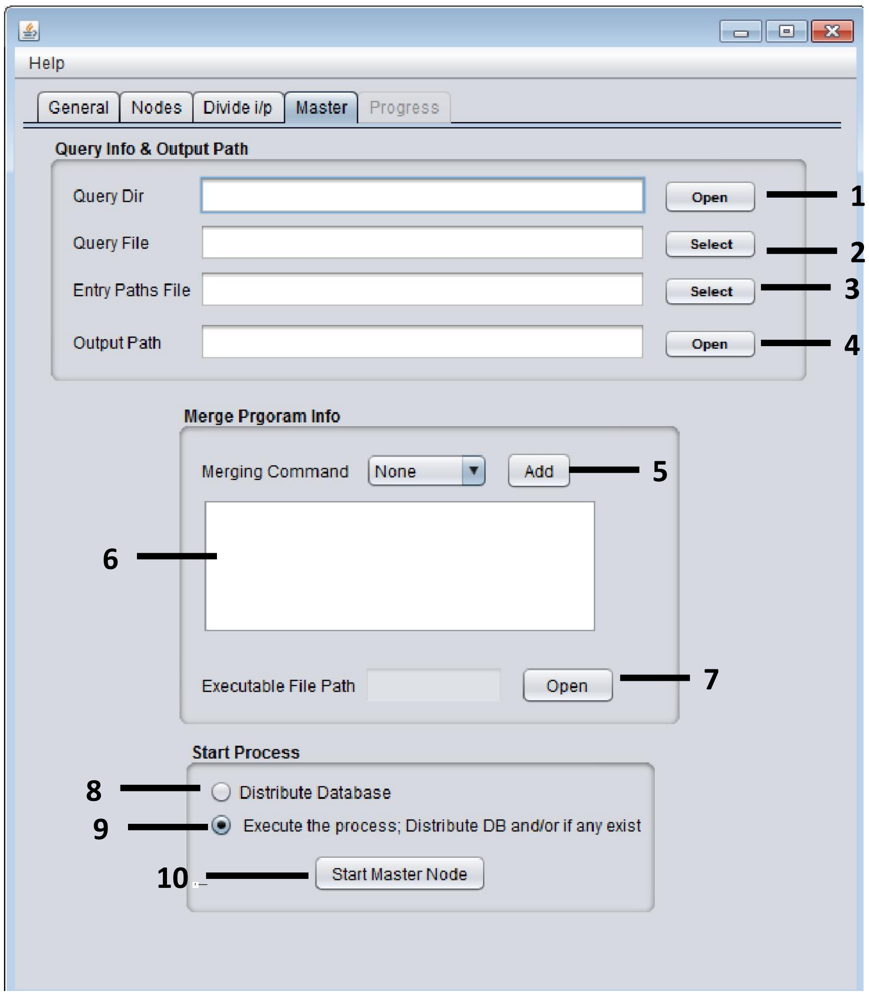
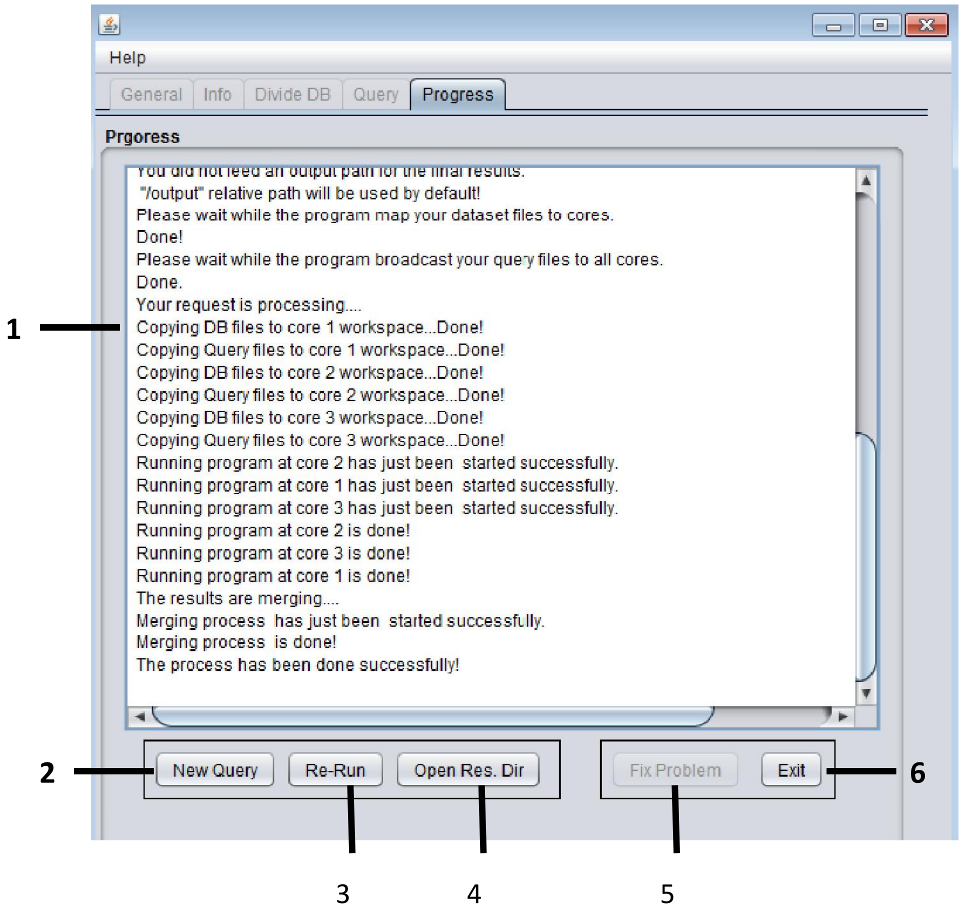


Figure 6-5 Query information

1- Open button, a button to select the query directory.
2- Select button, a button to select the query file; a file required by the original program.
3- Select button, a button to select the file that includes all the query entry paths; a file contains a query entry absolute path at a line. The number of lines of this file equals the number of queries.
4- Open button, a button to select the output of the final results.
5- Add button, to add a keyword of the merge command.
6- Text to write down the merge command.
7- Open button, a button to select the directory of the merge program binaries.
8- An option to distribute the database and save at the servers nodes. No queries or commands will be executed.
9- An option to distribute the database and to execute the supplied commands.
10- Start the master node to run the parallel version of the sequential program.

6.5 Tracking execution, submitting new queries and re-run

User can track the execution of the sequential program from the progress tab as shown in Figure 6-6 Progress, re-run and new tab. This tab also enables user to submit a new query, re-run the query, fix any problem indicated in the progress text area, and explore the results by opening the output directory.



```
You did not feed an output path for the final results.
"/output" relative path will be used by default!
Please wait while the program map your dataset files to cores.
Done!
Please wait while the program broadcast your query files to all cores.
Done.
Your request is processing....
Copying DB files to core 1 workspace...Done!
Copying Query files to core 1 workspace...Done!
Copying DB files to core 2 workspace...Done!
Copying Query files to core 2 workspace...Done!
Copying DB files to core 3 workspace...Done!
Copying Query files to core 3 workspace...Done!
Running program at core 2 has just been  started successfully.
Running program at core 1 has just been  started successfully.
Running program at core 3 has just been  started successfully.
Running program at core 2 is done!
Running program at core 3 is done!
Running program at core 1 is done!
The results are merging....
Merging process  has just been  started successfully.
Merging process  is done!
The process has been done successfully!
```

Figure 6-6 Progress, re-run and new tab

1- Text area for the messages of the sequential program executions.
2- New query button, a button submits a new query.
3- Re-run query button, a button re-runs the last run query.
4- Open Res dir, a button opens the results directory.
5- Fix Problem button, a button takes user to the master node tab to fix the indicated problem at the progress text area 1.
6- Exit button, a button exits the execution terminate the server and master nodes.