

IU RGRbench Workload/Benchmark User Guide to Database Creation and Population

The IU RGRbench is designed to evaluate different possible platforms as the basis for a Grid Information Server (GIS). The benchmark uses synthetic data to ensure specific levels of selectivity and randomness, (as in the Wisconsin benchmark) and uses scenarios designed to represent a workload that a GIS may be required to handle. Although deployed on three different platforms, the same benchmark data is used on all three platforms and the test cases/scenarios are designed to return the same result sets on each of the platforms to the extent that each platform can handle a specific query.

The purpose of this documentation is to provide a guide to deploying the RGRbench as a base case for comparison to other possible GIS platforms. For each of the three platforms, we will summarize what the platform requires and where the necessary software can be downloaded. For installation of the underlying software platforms we direct you to the documentation available from the developers/vendors of that platform. To the extent that there are specific settings needed for the RGRbench, this guide will provide details on those settings as well as discussing issues we came across in deploying the RGRbench on each platform. Since each of the platforms can be deployed separately, you can implement all of the benchmark or only selected platforms.

To provide consistent data across all three platforms, we developed a Perl script (in the db_schema folder of the RGRbench) that was used to populate the relational database MySQL and then that database was used to populate Xindice and MDS2. Since data population begins with the relational database, this user guide provides details on implementing the RGRbench on MySQL and then discusses the process for loading the data from MySQL into Xindice and LDAP. For installing each of these platforms, this user guide directs you to the web sites where the software and documentation can be downloaded from each of the platform's respective organizations.

The benchmark is written for Unix or Linux platforms. It was tested on a Dell dual processor Xeon server running Red Hat Linux 7.3 .

The benchmark can be obtained from <http://www/cs.indiana.edu/~plale/projects/RGR>. After unzipping and un-tarring the tarball, set the environment variable \$RGR_HOME to the top-level directory of the IURGRbench release.

1.0 Relational Platform

The relational platform uses MySQL v 4.0 for the database, with the queries written in C and compiled using the gcc compiler. According to the MySQL Reference Manual, the gcc compiler should be version 2.95.2 or later. The database and benchmark can either be compiled from the source code or run from binaries.

Installing mySQL:

The source code tarball as well as the binary installations can be downloaded from the mySQL web site at <http://www.mysql.com/>. Although the mySQL web site recommends using the binary downloads, after installing the RPM files for the server, client, libraries and headers, and changing the paths as needed in the makefile, we encountered issues with linking the benchmark (although it was able to locate the mhsq.h header file and compile). When we downloaded the tarball and compiled mySQL, we did not encounter problems compiling the benchmark. The mySQL Reference Manual can be downloaded from the mySQL web site and provides an wealth of information on installing the mySQL database.

1.1 Creating And Populating The mySQL Benchmark Database

Starting and Stopping the Database

After mySQL is downloaded and compiled, the mySQL database needs to be started, the database for the RGRbench must be created, and then the database must be populated with the synthetic test data using the scripts included in \$RGR_HOME/db_schema/relational. Before creating the database, first ensure that mySQL is running. The mySQL Reference Manual details the settings needed to automatically start up mySQL, but you can start the mySQL daemon manually by typing the following in the mysql directory:

```
bin/mysqld_safe &
```

(Note: prior to version 4 of mySQL, the database was started by typing safe_mysqld.)

To stop the database, type the following in the mysql directory:

```
bin/mysqladmin -u root -p shutdown
```

This assumes that a password has already been set for the root user in mySQL (which is not the same as the root user for Unix/Linux). After typing this command, you will be prompted for the mySQL root user's password and then the database will shut down.

Creating the Database

The database used in RGRbench is named "grid" and the database itself can be created through the mySQL client. Once the database is started, start the mySQL client by typing the following in the mysql directory:

```
bin/mysql -u <username> -p
```

where <username> is replaced by the name of a user you have created (or the root user if you have not created any users). You will be prompted for the password, and then the client prompt (mysql>) will be displayed. When the mySQL client starts, it will display a message to remind you that commands end with a semicolon in the client. If you forget the semicolon when typing a command and hit the enter key, a line continuation symbol will be displayed – at that point just type a semicolon and press the Enter key again.

To create the "grid" database, type the following at the mysql prompt and press the Enter key:

```
CREATE DATABASE grid;
```

After you have created the database, you can exit the MySQL client by typing `exit` and pressing the Enter key (no semicolon is needed for exiting). The creation of the tables and populating of the tables will be done by the Perl scripts. Although “grid” is the default name used in the scripts for populating the database and running the queries, any valid MySQL database name can be used as long as it is specified when running the relevant scripts.

Setting up MySQL to Connect From Perl

Creation of the benchmark tables and population of the tables with the synthetic benchmark data is done through Perl scripts located in `$RGR_HOME/db_schema/relational`. However, before you can run those scripts, you will need to set up DBI for MySQL access from Perl. DBI is a generic database interface for Perl, and you will need to have both DBI and a DataBase Driver (DBD) for MySQL installed (there are DBDs available for many data sources, but you only need the DBD for MySQL installed to run the grid benchmark). Starting with version 3.22.8 of MySQL, the Perl support is separate from the MySQL distribution. The MySQL reference manual has detailed instructions on loading DBI and the DBD for MySQL. The MySQL manual states that three separate tarballs need to be downloaded and installed in the following order: Data-Dumper, DBI, and DBD-mysql. However, according to the README file for Data-Dumper, as of version 5.004_71 of Perl, the Data-Dumper comes standard in Perl so you do not need to install it separately. Since the MySQL manual states that you need to have Perl version 5.004 or later installed for the DBD/DBI client code, and the Makefile.PL script for DBI states that Perl 5.6.1 will be the lowest version supported, there is no need to separately install Data-Dumper.

You can download the DBI and DBD-mysql tarballs from the CPAN web site; the easiest way to determine which versions of DBI and DBD-mysql to download is to follow the links from the MySQL “Download Perl/DBI modules” web page at: <http://www.mysql.com/downloads/api-dbi.html>

After you have downloaded the tar archives, unpack the DBI archive:

```
gunzip DBI-<version>.tar.gz
tar xvf DBI-<version>.tar
```

Change directories to the DBI-<version> directory, and then execute each of the following steps in the order specified. “make test” should complete successfully before running make install:

```
perl Makefile.PL
make
make test
make install
```

Once the DBI installation is successful, unpack the DBD-mysql-<version>.tar.gz archive and perform the same four steps in the directory created when the DBD archive was unpacked. When installing the DBD on Linux Red Hat, we specified the path to the MySQL libraries (the

same path as in the benchmark's Makefile). This is done by using the following to run the Makefile Perl script:

```
Perl Makefile.PL "--libs=-L/usr/local/mysql/lib/mysql -lmysqlclient"
```

This may be due to having installed MySQL in a non-default location. In using the above line for the lib, mysql is the pseudonym set up for the MySQL main directory under /usr/local/. If you did not create a pseudonym for the MySQL directory, you would need to specify the entire directory name including the MySQL version number.

As noted in the MySQL Reference Manual, MySQL must be running when the make test step for the DBD-mysql installation is executed. In addition, since during installation the ownership of the MySQL directories is changed, you will need to be logged in as the mysql user when running make test, but switch back to the root user when running make install so that you have rights to install in the Perl directory.

Creating the Benchmark Tables

Once DBI/DBD is set up, run the script that creates the benchmark tables. The script is named create_db.pl and it is located in the RGRbench/db_schema directory.

To execute the script, you will need to provide two arguments; the user name and password needed to connect to MySQL. If you are using the root user and your password is "password" (admittedly not too secure) then type the following from within the db_schema directory:

```
perl create_db.pl --username root --dbpassword password
```

As a default, the scripts to create and populated the table assume that the database is named "grid". However, if the database was given a different name when created, the optional parameter -dbname followed by the name of the database can be used. Also, the script uses "localhost" as the default host. If a different host is needed, it can be specified by including the optional parameter -host <hostpath>. The "create" script takes only a couple of seconds to execute.

If the script was not able to create a connection to the database, then the message "Bad Connection" will be displayed and the script will terminate. If this occurs, be sure that MySQL is running, and that the database named "grid" exists, (or the -dbname parameter is being used with the correct database name). Also, be sure that the correct user name and password were entered (remember these are the MySQL user name and password – not your UNIX name or password). If the script was able to create a connection, then it will display a message that says "Good Connection", followed by a series of messages that the tables in the benchmark were first dropped and then recreated. If the create script is run on an existing database, all of current data will be deleted since the tables are dropped.

Populating the Benchmark Tables

After the database tables are created, the Perl script named insert_data.pl, (which is also located in the db_schema directory) is run to populate the tables with the synthetic data. However,

before running the “insert” script, the following three text files must exist in the same directory as the Perl script:

- iplist.txt
- users
- cluster_names

These three files are included in the RGRbench in the db_schema directory. Since the same three files are used to populate MDS2 in addition to the mySQL database, the files are stored in \$RGR_HOME/db_schema/ldap. Prior to running the insert script for the mySQL database, copy these three files to \$RGR_HOME/db_schema/relational. Before making any changes to these three files, please read the Section 1.3 discussion of the contents and use of these files. The insert script is executed as follows:

```
perl insert_data.pl -username root -dbpassword password
```

As with the script to create the tables, the -dbname and -host parameters are supported. This script completes within 10-20 minutes. After the insert script completes, the database is ready. Section 1.3 contains a discussion of the methodology used to populate the grid database.

1.2 Running The RGRbench Relational Benchmark

After the mySQL database has been populated, compile the benchmark queries. The queries are written as separate C programs. The source code for the queries and the “make” file are located in \$RGR_HOME/benchmark_scripts/relational/mySQL. Please consult the separate documentation on building the query executables that exists in that directory.

After the 14 queries have been compiled, each can be run separately from the system prompt by typing the name of the executable and using the parameters listed below. In addition, the queries can be run as a set using thescript \$RGR_HOME/run_bench/REL_run.sh.

When running the benchmark queries, the following parameters must be specified:

- username <user name>
- dbpassword <password>
- dbname <grid>

For these parameters, the user name and password refer to the mySQL username and password and not the system login username or password (which may be different). The -dbname parameter specifies the mySQL database to be used. The default name is “grid”. In addition to the required parameters, there are the following optional parameters:

- iter <L or S>
- host <host name>

Each of the queries programs runs a query for either 1,000 or 10,000 iterations, which are regarded as the Small and Large options respectively. If no -iter parameter is specified when the query is executed, the default is to use the “small” option. To run the query for 10,000 iterations, include the -iter option with the value L. When running all of the queries as a set using the REL_run.sh script, the -iter parameter is required. In addition, the query programs default to using “localhost” when connecting to mySQL. If another path needs to be specified, then use the -host option and specify the host path to be used.

Example

To run the “manyRelations” query with a mySQL user name of “bplale”, a mySQL password of “profplale”, for 10,000 iterations, where the mySQL host is “bench.indiana.edu”, and the database name is the default of “grid”, the following would be entered at the system prompt in the benchmark_scripts/Relational/mySQL directory under RGRbench after the benchmark queries were compiled and the mySQL database was populated as discussed earlier in this user guide:

```
manyRelations -iter L -dbname grid -username bplale -dbpassword profplale -host  
bench.indiana.edu
```

When running the benchmark queries, typing `-help` as the only attribute will also display the parameters and their possible values:

```
manyRelations -help
```

For details regarding the actual mySQL queries in each of the benchmark executables as well as the number of tuples returned, please refer to the separate query documentation in the `run_bench` directory.

1.3 Benchmark Data

This section summarizes our approach and some design decisions that were made regarding the structure of the database and the Grid Laboratory Uniform Environment (GLUE) schema.

Required Data Files

When the insert script was run, three input text files were needed. These three files provide a list of eight organizations and a corresponding IP address (`iplist.txt`), a list of 60 user names and their corresponding user IDs (`users`), and a list of twenty server names for the clusters (`cluster_names`). Although you can change the names of users, please see the discussion below regarding the format and keep in mind that the file should contain *exactly* 60 unique users in order to match the published query results. There should be no need to change the data in either the `iplist.txt` or `cluster_names` files, but if changes are made to either file, care should be taken since these files are closely related and changes to either file are likely to cause problems with the “insert” script. In designing the IU RGRbench, one goal was to provide a deterministic model in that would generate the same results each time it was run, while also simulating the randomness that would occur in a real environment. If changes are made to any of these three files, the result set may not match the published results of the RGRbench. The details regarding each of these files are as follows:

users

The `users` file contains 60 rows with an alphanumeric user ID followed by a user name. If any changes are made to this file, each line must start with an alphanumeric character, (a-z, 0-9, or an underscore). Any line starting with whitespace or any other character that’s not alphanumeric will be skipped. Any line starting with the `#` symbol will be treated as a comment line and skipped. All alphanumeric characters at the start of a line up to the first whitespace or non-alphanumeric character will be considered the user ID. The remainder of the line will be treated as the user name, but any whitespace at the start or end of that name will be trimmed off. The

user named “amahabal” is referenced in one of the benchmark queries and should not be altered in the users file.

iplist.txt

This file contains eight domains and corresponding IP addresses in the format of an alphanumeric domain name, (a-z, 0-9, or an underscore) followed by a tab, and then the remainder of the line is treated as the IP address. If any changes are made to this file, extreme care should be taken since there is a strong possibility that it could cause problems with both the “insert” script and the benchmark queries. If any changes are made, be sure not to add any whitespace characters other than the tab between the domain and IP address. In addition, the domain names need to match the domain names embedded in the records in the cluster_names file. The domain name embedded in each of the cluster_name records (e.g., “indiana” in “tamarack.cs.indiana.edu”) must be included as a domain name in the iplist.txt file. Because some of the benchmark queries specifically look for cluster records including the domains “sdsc”, “uchicago”, and “utexas”, these should always be included in the iplist.txt file. The actual digits included in the IP addresses in the iplist.txt file can be altered without any consequences.

As is the case with both the users file, any line starting with whitespace will be skipped and any line starting with the # symbol will be treated as a comment.

cluster_names

This file contains 20 rows with alphanumeric cluster names. These cluster names contain the organization names from the iplist.txt. Making any changes to this file could cause problems with both the insert script and the benchmark queries. In particular, the four clusters named “mds.sdsc.edu”, “perigee.sdsc.edu”, “golden.tacc.utexas.edu”, and “sharkestra.uchicago.edu” are all referenced as criteria in the benchmark queries. As is the case with both the users and iplist.txt files, any line starting with whitespace will be skipped and any line starting with the # symbol will be treated as a comment.

Methodology Used to Generate the Benchmark Data

In populating the database, we strike a balance between creating realistic data that can be used to simulate potential real world situations and having consistent data that will allow for deterministic modeling and consistent results across platforms. As an example, each cluster contains a number of subclusters, and each subcluster will contain a number of hosts. However, for the data to be realistic, each cluster should not contain the same number of subclusters, and each subcluster should not contain the same number of hosts. To achieve this goal, a range was determined for the number of subclusters each cluster contains. While some clusters may have a single subcluster - creating a one-to-one relationship, other clusters may contain a number of subclusters. For the cluster-to-subcluster relationship from 1 – 30 subclusters were created for each cluster. Since this relationship could not be linear, (i.e., the first cluster would not have one subcluster and the second cluster have two clusters, etc.) the insert script starts in the middle of the range (14 subclusters) and then increments by 7, taking the modulus based on 30. In creating cluster/subcluster relationships, the first cluster would have 14 subclusters, the second cluster would have 21 subclusters, the third cluster would have 28 subclusters, and the fourth cluster would have 5 subclusters. This approach results in an even distribution within the desired range while also ensuring repeatable results for different users of the benchmark data. In all situations

where the data varies over a range, this approach is used. Appendix A contains details as to the range, increment, and starting values for each data element where a distribution over some range was necessary to simulate realistic data.

2.0 XML and LDAP Platforms

After the mySQL database has been populated, that data is used as the source for populating both the XML-based Xindice database and the LDAP database under MDS2.

2.1 Xindice

The Xindice database software can be downloaded from the Apache Xindice site at <http://xml.apache.org/xindice/>. In addition, you will need to have Apache Tomcat version 4.1.12 or higher installed. The Tomcat software can be downloaded from the Apache Jakarta Project web site at: <http://jakarta.apache.org/tomcat/>. While the Xindice and Tomcat web sites are your best source for general information on installing their respective software platforms, following are the additional settings and procedures for populating the Xindice database. Directions for populating the Xindice database are also contained in the README file in the db_schema/xml/ directory.

- Set the environment variable "XINDICE_HOME" to the directory where Xindice was installed. For example, if Xindice was install in the directory /Xindice, the environment variable should be set using:

```
setenv XINDICE_HOME /Xindice/xml-xindice
```

- The data for the Xindice database is imported from the mySQL database to provide a consistent set of data for comparison purposes. The data is extracted from the mySQL database using the mysqldump utility. This utility can be run from any directory and will create an extract file. The extract file created must be renamed outgrid.txt and moved to the \$RGR_HOME/db_schema/xml directory before running the import for Xindice. Run the mysqldump utility using the following, but substituting the mySQL username for "username", the host server name for "hostname", and the actual mySQL database name for "db_name" ("grid" is the default name when the mySQL database is created). The output is directed to the file name used for "backup_file".

```
mysqldump --user "username" --host "hostname" --password "db_name" > backup_file
```

- The backup_file created using mysqldump must be renamed as outgrid.txt and moved to the \$RGR_HOME/db_schema/xml directory.
- A Perl script named "dataGrid.pl" in the \$RGR_HOME/db_schema/xml directory is used to process the mySQL dump file and create the import files needed for loading data into Xindice. This script will create a large volume of files used to import both data and indices for the Xindice database. A separate script, clean.sh can be used to remove these files after all of the import process is completed. The dataGrid.pl script is run by executing the following within the \$RGR_HOME/db_schema/xml directory:

```
./dataGrid.pl
```

- The files created by the Perl script are used to import the data into Xindice using a Java script. This script is run by entering the following in the \$RGR_HOME/db_schema/xml directory:

```
java PopulateGrid
```

- The Java script loads the documents into the Xindice database, and a separate script creates the indexes needed. The java script is rerunnable and will clear all of the data before repopulating the database, but a separate script is run (deleteIndex.sh) to delete the indexes before recreating them. The script to create the indexes is run by executing the following in the \$RGR_HOME/db_schema/xml directory:

```
./createIndex.sh
```

- After running both the PopulateGrid and createIndex.sh scripts, the Xindice database has been populated with the same data as the mySQL database. The files created by the dataGrid.pl Perl script can be removed by executing the following script in the \$RGR_HOME/db_schema/xml directory:

```
./clean.sh
```

The approach outlined above allows for a consistent set of test data between the relational platform and the Xindice XML and MDS2 platforms.

2.2 MDS2

The main Globus Alliance web site is at: <http://www.globus.org/>. The IU RGRbench uses MDS2 Globus Toolkit 2.2. Installation instructions as well as the binary and source code downloads are available from the archive site. Be sure to set the environment variable for Globus. Although your environment variable settings may be different depending on where you install the software, it should be similar to:

```
setenv GLOBUS_LOCATION /usr/local/packages/globus/2.2
```

As with the Xindice database, the IU RGRbench contains the scripts necessary to import the data for the MDS2 database from the mySQL database to ensure a consistent set of test data between the platforms. After the Globus location environment variable has been set, the procedures outlined below can be used to populate the MDS2 database. The scripts used to populate the database extract data directly from the mySQL database. However, in testing the MDS2 platform, we encountered very long response times using the full mySQL database. For this reason, the Perl script used to populate the mySQL database (insert_data.pl) has been modified to accept an optional parameter, “-dbsize” which can be used with the values 5, 10, or 25 to create a mySQL database with only 5%, 10%, or 25% of the volume of data. The default is 100%. A separate database can be created in mySQL to use with the import scripts for MDS2.

Following are the steps needed to populate and run the MDS2 database:

- After you have installed the Globus version 2.2 toolkit, MDS2 must be started with the GLUE schema and RGR schema extensions. These schemas, four in all, are located in \$RGR_HOME/db_schema/ldap. They must be copied to \$GLOBUS_LOCATION/etc and added to the startup configuration file \$GLOBUS_LOCATION/etc/grid-info-slapd.conf.

- The data file for MDS2 is a “.ldif” file. There is a PHP script in the \$RGR_HOME/db_schema/ldap directory that is executed to create an ldif from the MySQL data. This script requires four parameters:
 1. The name of the host server containing the MySQL database.
 2. The name of the MySQL database. The default name is “grid” when creating the MySQL database, but a different name can be used (particularly if the –dbsize option is used to create a smaller data set for MDS2).
 3. The MySQL database user name.
 4. The MySQL database password.

The PHP script in the \$RGR_HOME/db_schema/ldap directory is run by executing the following in that directory:

```
./insert_data.php "host_name" "database_name" "user_name" "password"
```

- The PHP script will create the dump.ldif file in the directory where the script is run. That file must be copied to the \$GLOBUS_LOCATION/etc/ directory and a softlink named rgr.ldif must be created for that file. The softlink can be created by executing the following in that directory:

```
ln -s rgr.ldif dump.ldif
```

When changing the target file (data source file) for the softlink, the MDS2 database must be stopped. The database can be stopped and started with the following commands:

Starting MDS2:

```
$GLOBUS_LOCATION/sbin/globus-mds start
```

Stopping MDS2:

```
$GLOBUS_LOCATION/sbin/globus-mds stop .
```

- When running the MDS2 database with 100% of the RGRbench data from the MySQL database, we needed to adjust the time-to-live parameters in MDS2 to get the test data to stay in the cache. These values are mainly in grid_info-resource-ldif.conf. We would suggest that you build an ldif file that has 5% to 10% of the total database size.

Directions for importing data into the MDS2 database are also contained in the README file in the db_schema/ldap/ directory.