condor_dbq.pl

Provides a relational database management system interface to Condor. Condor submit file contents are inserted into a table, *condor_dbq.pl* submits the job to Condor, and updates a table with the status of the work to allow the monitoring of its progress.

Synopsis

condor_dbq.pl [**--dbinfofile** *dbinfofile*] [**--dbinfodir** *dbinfodir*] [**--logdir** *logfiledir*] [**--sleepamount** *seconds*] [**--maxwork** *numofwork*] [**--maxjobs** *numofjobs*] [**--grabamount** *numjobs*]

condor_dbq.pl [**--initdb**] [**--createlang**] [**--revokepublic**] [**--createsubmituser** *[dbinfofile]*] [**--createworkuser** *[dbinfofile]*] [**--submit** *submitfile*] [**--noexecute**]

condor_dbq.pl [**--help**]

condor_dbq.pl [**--version**]

Description

condor_dbq.pl provides a relational database interface to submit Condor jobs, and monitor their progress as they are executed. The user submits work to the system by inserting a row into a database table. The user can query the *condor_dbq.pl* tables to monitor the progress of their work.

condor_dbq.pl is careful to submit each piece of work into Condor exactly once. *condor_dbq.pl* can be stopped at any time. Upon restart it will recover its state and resume operation.

condor_dbq.pl has two main modes of operation. The first, its normal mode, performs the role of taking jobs from the database, submitting them to Condor and updating the results of their execution. The second mode performs administrative tasks on the database including creating the necessary tables and accounts to use *condor_dbq.pl*.

In the normal mode of operation, *condor_dbq.pl* alternates between polling the database for new work to submit to Condor, and monitoring the submitted work's user logs for new events. The database is updated to reflect the current state of the system. There are several options described in the Options Section that can be used to throttle *condor_dbq.pl*

The submission data for each work is in the same format allowed by *condor_submit*. The only limitation is that the user can not specify a user log using the *log* attribute as *condor_dbq.pl* will override this attribute with its own value. *condor_dbq.pl* uses *condor_submit* to perform the submission into Condor. The only changes made to the submission are the user log is overridden, and attributes with a prefix of *CDBQ_* are added. For correct operation, work submitted to this *condor_schedd* outside of *condor_dbq.pl* must not add attributes or modify atttibutes with a prefix of *CDBQ_*.

condor_dbq.pl operates by using two tables. The *work* table represents each piece of work. The *jobs* table represents the individual jobs once the work is submitted to Condor. Other than inserting the work initially, users should only query the tables, and should not otherwise update values unless specified below. A description of the fields and types of each table are described next.

work table Each row represents a piece of work to be executed by Condor. The table's attribute names, their SQL types and descriptions are as follows:

work_data TEXT Stores the Condor submit file data. This is the only attribute that needs to be set when inserting a new piece of work.

create_ts TIMESTAMP Database timestamp when record was created.

id BIGINT Unique id for this record.

insert_user TEXT Database user that inserted this record.

state TEXT The state of the piece of work in the system. The possible values are as follows:

initial Job has not yet been submitted to Condor.

chosen Job is about to be submitted to Condor.

in_batch Job has successfully been submitted to Condor, but has not yet completed.

complete All the jobs of this work have completed successfully or have beeen removed.

failed The work has failed due to a permanent error.

cdbq_user TEXT Database user that inserted this record into Condor. NULL if *condor_submit* not executed.

cmd_stdout TEXT Standard out of *condor_submit* after submitting this job. NULL if *condor_submit* not executed.

cmd_stderr TEXT Standard error of *condor_submit* after submitting this job. NULL if *condor_submit* not executed.

cmd_exit_code INTEGER Exit code of *condor_submit*. 0 is success, other values indicate failure. NULL if *condor_submit* not executed.

cmd_exit_signal INTEGER Signal causing this *condor_submit* to fail. 0 is non-signal exit, other values represent the signal number. NULL if *condor_submit* not executed.

log_file TEXT Path to Condor's user log file for this work.

next_pos INTEGER Offset between last and next event record to read.

total_jobs INTEGER Total number of jobs that are represented by this work.

complete_jobs INTEGER Total number of jobs that have completed or been removed.

log_err_msg TEXT Last error message caused by accessing the user log file.

log_err_num INTEGER Last error number caused by accessing the user log file.

update_ts TIMESTAMP Database timestamp when record was last updated.

user_id INTEGER For use by the user to store an integer value.

user_text TEXT For use by the user to store a text value.

jobs table: Each row repreents a job of a piece of work. These rows are inserted once the work has been submitted to the Condor system and Condor has written a submitted event to the user log file. The table's attribute names, their SQL types and descriptions are as follows:

work_id BIGINT Id from the *work* table.

cluster INTEGER Condor cluster id of this job.

proc INTEGER Condor process id of this job.

subproc INTEGER Condor subprocess id of this job.

state INTEGER Condor user log record event type of the last event seen for this job.

info TEXT Additional information from the last log record seen for this job.

record_ts TIMESTAMP User log timestamp from the last log record event processed.

create_ts TIMESTAMP Database timestamp when job was first seen.

update_ts TIMESTAMP Database timestamp when record was last updated.

exit_code INTEGER Exit code of of this job. 0 is success, other values indicate failure. NULL if job not yet complete.

exit_signal INTEGER Signal causing this job to fail. 0 is non-signal exit, other values represent the signal number. NULL if job not yet complete.

The other functions of *condor_dbq.pl* perform administrative tasks on the database. These include inserting a new record into the work table given a Condor submit file, and creating the tables and accounts required by *condor_dbq.pl*. If any of this functionality is requested, only the administrative tasks are performed; the normal processing of work is not started.

If **--noexecute** is used with any of the administrative functions, the SQL statements are printed to standard out instead of being executing. This is useful to review the statements for security purposes, or to modify them before creating the database objects.

The **--submit** *filename* option is used to insert a work record into the system using the contents of *filename* for the Condor submit file data. If *filename* is '-', the program reads the Condor submit file from standard in.

The **--initdb** option causes the tables and other database objects to be created.

The **--createlang** option creates the plpgsql language to allow the trigger functions created by **--initdb** to be used. This is not necessary if the database administrator created the language for use when the database was created.

The **--revokepublic** option prevents the database users from creating object new tables in the database in the 'public' schema that PostgreSQL grants by default to all users. If the user accounts do not need to create tables this option should be used to revoke this unnecessary privilege.

The **--createsubmituser** *[dbinfofile]* option creates a database user that can be used to submit jobs. The username and password for this account is obtained from the db information file specified, or using the default location for submit db information file if none is specified. This account has privileges to insert work into the work table, and read values from both tables.

The **--createworkuser** *[dbinfofile]* option creates a database user that can be used by *condor_dbq.pl* to perform its normal operation. The username and password for this account is obtained from the db information file specified, or using the default location for worker db information file if none is specified. This account has privileges to update existing records in the work table, and to insert and update records in the jobs table.

There are three types of database accounts required by *condor_dbq.pl*. The first is an administrative database account that can create database accounts, tables and other objects. The administrative account requires database privileges to create tables and users in the database. This account is used by the administrative commands except the submit function.

The second account type is used to insert work records into the database by the **--submit** option. This account requires privileges to insert records into the work table. The account can be created using the **--createsubmituser**, which also grants access to select records from the two tables allowing this account to used to both submit work and to monitor its progress. Multiple acconts of this type may be used.

The final account type is used to perform the actual work of the system. This account requires privileges to update records in the work table and to both insert and update records in the jobs table. The account can be created using the **--createworkuser**.

The account information describing an account is stored in a database information file that contains three lines: the perl DBI string describing the database, the username, and the password. As these files contain sensitive information, only the operating system accounts used to run the *condor_dbq.pl* should be able to read the database account information files. By default, a separate db information file is used to store each of the three accounts with the program selecting the credential information based on the selected operation. These files are stored in a directory specified by **--dbinfodir**, and are named db.admin.conf, db.submit.conf, and db.work.conf for the administrative, submit and work accounts respectively. If this scheme is used the correct account will be used without having to specify the account information file. If an account information file is missing *condor_dbq.pl* will fail if an operation is selected that requires the missing account information file.

Options

--dbinfofile *dbinfofile* Db information file containing the DBI connection string, username and password one to a line. If not specified this file defaults to files in the **--dbinfodir** directory with the names db.work.conf, db.submit.conf, or db.admin.conf, based on the functionality requested being process the queue, submit a job, or perform administrative database functions respectively. The default value is the current directory.

--dbinfodir *dbinfodir* Directory to look for default database information files when **--dbinfofile** is not specified.

--logdir *logfiledir* Directory name to place the Condor user log files generated by the Condor system. The default value is the current directory.

--sleepamount *seconds* Amount of time to sleep between each iteration of polling the database system for new work and processing new events in the Condor user logs. The default value is 10 seconds.

--maxwork *numofwork* Maximum number of inprogress work in the Condor system. If there is less work in the Condor system, new work will be submitted if available.

--maxjobs *numofjobs* Maximum number of incomplete jobs in the Condor system. If there are fewer jobs, new work will be submitted if available.

--grabamount *numofwork* Maximum amount of work to pull from the database system to insert into the Condor system per iteration. The default value is 10 work records.

--initdb Create the tables and other associated database objects required by the condor_dbq system.

--createlang Causes the plpgsql language to created in the database.

--revokepublic Revokes the default ability of database users to create new tables in the 'public' schema in the database.

--createsubmituser *[dbinfofile]* Create a database account that has proper permissions to insert a work submission to the database. The username and paosword for the account is taken from the db information file specified or the default submit db information file is used if none is specified.

--createworkuser *[dbinfofile]* Create a database account that has proper permissions to allow condor_dbq to process the queue. The username and paosword for the account is taken from the db information file specified or the default worker db information file is used if none is specified.

--submit *submitfile* Insert a row in the database containing the contents of *submitfile*. A *submitfile* value of '-' will read the submit file from standard in.

--noexecute Do not execute any of the administrative database commands. Just print them out to standard out.

--help Print a help message and exit.

--version Print the version and exit.

Examples

This section will present a complete introduction to using *condor_dbq.pl* including setup of the environment, creation of the database accounts and tables, inserting work, and querying results. This example assumes the database 'condor_dbq_db' is already created, and accessible on the same host via TCP/IP on the standard port. It also assumes that the administrative database account, 'condor_dbq_admin_user,' exists with the password 'condor_dbq_admin_password,' and that this account has the ability to create new database users and tables in the 'condor_dbq_db' database. The example code is written in the bash shell.

Create a secure directory to store the db information files.

```
mkdir condor_dbq_conf
chmod 0700 condor_dbq_conf
```

Create the database account information files for a PostgreSQL database named condor_dbq_db:

```
cat > condor_dbq_conf/db.admin.conf <<EOF
DBI:Pg:database=condor_dbq_db;host=localhost
condor_dbq_admin_user
condor_dbq_admin_password
EOF

cat > condor_dbq_conf/db.submit.conf <<EOF
DBI:Pg:database=condor_dbq_db;host=localhost
condor_dbq_submit_user
condor_dbq_submit_password
EOF

cat > condor_dbq_conf/db.worker.conf <<EOF
DBI:Pg:database=condor_dbq_db;host=localhost
condor_dbq_worker_user
condor_dbq_worker_password
EOF
```

Create the plpgsql language used by objects created by **--initdb**:

```
condor_dbq.pl --dbinfodir=condor_dbq_conf --createlang
```

Create the database tables and objects:

```
condor_dbq.pl --dbinfodir=condor_dbq_conf --initdb
```

Revoke the default ability of database users to create new tables:

```
condor_dbq.pl --dbinfodir=condor_dbq_conf --revokepublic
```

Create the worker account that *condor_dbq.pl* will use for its normal operation using the default db worker information file for account information:

```
condor_dbq.pl --dbinfodir=condor_dbq_conf --createworkuser
```

Create the submit account that users can use to insert work into the system using the default submit db information file for account information:

```
condor_dbq.pl --dbinfodir=condor_dbq_conf --createsubmituser
```

Create the directory to store the user logs:

```
mkdir condor_dbq_logs
chmod 0722 condor_dbq_logs
```

Start the system:

```
condor_dbq.pl --dbinfodir=condor_dbq_conf --logdir=condor_dbq_logs
```

Create a Condor submit file containing a simple job:

```
cat > my.submit <<EOF
Universe = vanilla
transfer_executable = false
notification = never
output = my.out
error = my.err
log = my.log
Executable = /bin/sh
Arguments = "-c 'exit 0'"
Queue
EOF
```

Insert the Condor submit file into the work table:

```
condor_dbq.pl --dbinfodir=condor_dbq_conf --submit=my.submit
```

Exit Status

condor_dbq.pl will exit with a status value of 0 (zero) upon success, and it will exit with a non-zero value upon failure.

Author

Condor Team, University of Wisconsin–Madison

Copyright