Citrina User Guide

Introduction

Citrina is a program that is used for mirroring and processing biological databases. It uses the Ant build system (http://ant.apache.org/) to handle most of the work. Ant lends itself very well to this type of a system because of the need for fault tolerance, file management, and extensibility. The way Citrina works is by specifying the database parameters in a properties file. This file defines information like ftp server, remote directory location, and the required files to download. When you run Citrina it reads the property file and then checks the remote site for the files you specify. It then checks the timestamp of the local files against what is on the remote ftp server. If a newer file is found it is added to the download list. Once all files are checked then Citrina downloads all the necessary files using wget to the offline/temporary staging area.

After downloading has completed the files are extracted and then moved to the online/production staging area. There are also pre and post processing targets for each database and global pre/post targets. Global pre/post processing targets are executed before any database is checked and after every database has been extracted respectively. Pre/post targets for each database are executed before the database is checked and after it is extracted. The pre/post targets are the mechanisms through which complex analysis and/or reformatting pipelines can be tied into your data mirroring jobs.

Requirements

- Java 1.4.x
- Ant 1.6.1 or higher
- wget 1.6 or higher
- gunzip 1.3.3 or higher
- gnu tar 1.13 or higher
- bunzip 0.9.0 or higher
- · RedHat Linux 9, Solaris 2.8, SuSe 9.x

Gunzip, gtar, and bunzip are only required if you are downloading files that require that tool for extraction. Citrina has only been tested on these platforms but should work on other versions and other Unix operating systems like AIX, Fedora, IRIX, or Mac OS X.

Installation

- 1. Install the required applications listed in the requirements section.
- 2. Set the JAVA_HOME and ANT_HOME environment variables to the locations where you installed them. If ANT HOME is not set then ant must be in your PATH.
- 3. Copy the global.ftp.properties and global.system.properties files from citrina/conf/db properties/default into citrina/conf/db properties.
- 4. Edit both these files according to the instructions given in the configuration section of this document.
- 5. That is it!!!

Configuration

Directory Layout

citrina/-Root directory that contains the main build file.

lib/ -Contains 3rd party and custom Citrina jar files.

log/ -Holds log files of all Citrina runs.

doc/ -Program and API documentation for the custom ant tasks. projectfiles/ -Holds the bulk of the ant build files for citrina. conf/db_properties/ -Holds the database and global property files.

bin/ -Holds shell scripts used to run citrina.

Global Properties

There are two global properties files that control some critical aspects of Citrina. The first file is named global.system.properties. This file sets the number of threads to use for updating and the paths to external binaries used by Citrina. Values in this file should match what exists on your system and what your system can handle. Setting a thread value of 2 will start 2 simultaneous downloads of separate databases. If only one database is specified for download then only one thread is started.

The second file is called global.ftp.properties and configures options related to the data retrieval. The only parameter in this file that needs to be modified is the ftp.data.dir property. This property sets the root directory where all the database directories will be stored. This can be overridden in the individual database property files if you do not have a single root location for data storage. Take care not to modify the wget parameters unless you really know what you are doing. Incorrect wget options can cause data to be downloaded incorrectly.

Note: Before running Citrina for the first time you will need to copy global.system.properties and global.ftp.properties from citrina/conf/db_properties/default/ into citrina/conf/db_properties/. Then modify them according to the previous instructions.

Database Properties

Each database target in Citrina has to be defined in a property file. Citrina comes with a default set of many databases under citrina/conf/db_properties/default. To start downloading a database simply copy the dbname.properties file from citrina/conf/db_properties/default to citrina/conf/db_properties. Then create the online and offline directories under the ftp.data.dir that you set in the global.ftp.properties file. In the following table is a description of all the properties defined in a property file if you need to modify any of the default settings or want to create your own.

Property Name	Description
ftp.db.fullname	The full name of the database. Can contain spaces if placed within quotes.
ftp.db.name	The short name of the database without spaces.
ftp.server	The URL for the ftp server. eg. ftp.ncbi.nih.gov
ftp.port	Override the default port setting defined in global.ftp.properties for this database only.

Property Name	Description
ftp.username	Set if the server requires a login other than anonymous.
ftp.password	Set if the server requires a login other than anonymous.
ftp.remote.dir	Remote ftp directory where the data resides.
ftp.remote.files	A Java regex pattern that tells Citrina which files to mirror.*
ftp.data.dir	Overrides the default data directory set in global.ftp.properties.
ftp.online.dir.name	Directory name for the online database directory.
ftp.offline.dir.name	Directory name for the offline database directory.
ftp.local.files	A Java regex pattern that matches the local database filenames after they have been extracted.*
ftp.link1.dir	If the online directory is a link to another directory then this should be set to the name of the link target.
ftp.link2.dir	Same as above only for the offline directory. See the section on switch links for more information.
wget.options	Overrides the default wget options set in global.ftp.properties.
ftp.no.extract	Set to true if you don't want to extract the database files.
do.link.copy	When set to true this property tells Citrina to create symlinks from the online to offline directory instead of copying. The default is true for most databases but can be set to false to disable it.
db.pre.process	The filename of the preprocess ant build file to be executed before the database is downloaded. This file needs to exist under the citrina/projectfiles directory.
db.post.process	The filename of the postprocess ant build file to be executed after the database is downloaded. This file should also be placed in the projectfiles directory.

^{*}Note: See next section on pattern matching for details on the regular expression format.

Pattern Matching

There are two database properties that are responsible for file pattern matching. The first pattern property is called ftp.remote.files and is used to tell Citrina what files to download from the ftp server. This pattern is of the form defined by Java's java.util.regex package

(http://java.sun.com/j2se/1.4.2/docs/api/java/util/regex/package-summary.html) and it accepts Perl compliant patterns. For instance, if we wanted to exclude all files relating to the EST division of genbank we could set the regex to gb[a-df-z].*\\.seq\\.gz. That would match any file except the EST files since their

name is gbestxxx.seq.gz where xxx is a number sequence. If your files are located in sub directories of the remote ftp directory then you can tell citrina to descend into the directory by giving a pattern like this: .*/myfiles\\.tar\\.gz

That pattern would descend into any sub directory of the remote ftp directory and grab a tarball called myfiles.tar.gz. For more examples of using this remote pattern look at the genbank, pdb, uniref, or any of the other default property files in the conf/db properties/default directory.

The second property is called ftp.local.files. This pattern is also a java regex compliant pattern and is used to tell Citrina what the local file names will look like once they are downloaded and extracted. In most cases it essentially indicates that the zipped extension is gone like pdb1ab.ent.Z to pdb1ab.ent. However, in some cases it is used to indicate a flattening or conservation of remote ftp directory structures.

If more than one pattern is required to match files in either case of pattern matching, then multiple patterns can be placed on the same line delimited by a space. For example, the Pfam database property file has this line for remote file matching:

 $\label{lem:condition} $$ftp.remote.files=Pfam-A\.(full|seed)\.gz\ Pfam-[BC]\.gz\ Pfam_l[fs]\.gz\ swisspfam\.gz$

All on one line.

Please note that all regular expression patterns that use the backslash character ('\') to escape certain characters or use character classes need to use two backslashes instead of one because of how Java parses properties files. For more information about Java regular expressions in general read: http://java.sun.com/j2se/1.4.2/docs/api/java/util/regex/Pattern.html.

Staging

Citrina uses staging to limit the interruption to end users as much as possible. The terminology we use is online and offline to refer to production and processing staging areas. Citrina will download files into the offline area and will not move it until all processing has finished. If multiple databases are being updated then all data will remain offline until the last database has finished processing. This also helps in cases of data corruption. If Citrina runs into a problem during a database update then the data is left in place so you can correct the problem before the old and otherwise good data is replaced with data that could contain errors.

Switch Links

The term switch links is used to describe the use of symbolic links to manage data directories. They are used in cases of large databases that are updated in bulk at once and that would require a large amount of hard drive IO to move files from the offline staging area into the online or production area. For example, with Genbank we use switch links in this manner:

```
genbank -> genbank_0
genbank tmp -> genbank 1
```

In this case genbank and genbank_tmp are the online and offline data directories, respectively. They are in fact not real directories but symbolic

links that point to two directories that hold the actual data. During an update all new files for genbank would be downloaded and processed in the genbank_tmp directory. After this process is completed they need to be moved to the production area. Since genbank is a fairly large data set this can require a lot of disk IO. With switch links, Citrina eliminates the disk IO problem by swapping the symbolic links. To use switch links for your database all you need to do is setup the correct directory and symbolic link structure and then place the name of the link targets in the database properties named ftp.link1.dir and ftp.link2.dir.

Here is an example of how to setup switch links:

- \$ cd /data
- \$ mkdir mydb online
- \$ mkdir mydb offline
- \$ ln -s mydb online mydb
- \$ ln -s mydb offline mydb tmp

Then you need to edit the mydb property file and set ftp.online.dir.name to mydb and ftp.offline.dir.name to mydb_tmp. Then set ftp.link1.dir to mydb online and ftp.link2.dir to mydb offline. Your done!

Note: Some databases are setup to use switch links by default. If you do not setup the directory structure for switch links then you will have to comment out the ftp.link1.dir and ftp.link2.dir lines from the database property file.

Copy Phase Links

During the copy phase of the mirroring process, database files that have not changed since the last update are copied from the online directory to the offline. In order to speed up updates, Citrina 0.5.1 and higher creates links for the unchanged files pointing from the offline to the online file location. Then, when the database is moved online, only the new files have to be moved. However, if your database processing scripts edit or modify the database files in any way before the database is moved online this can disrupt the symbolic links and cause problems. In these cases it is best to disable the symbolic link copy by setting the do.link.copy property to false in the database property file. If your post processing scripts only use the files for input and do not edit them everything should be fine.

The copy phase links are not used for databases that use switchlinks or for databases like Swissprot Update that never add new files to their ftp release. Databases like Genbank Update do take full advantage of copy phase links.

Running

Starting Citrina

To start Citrina you need to use the citrina.sh script located in the bin directory. This program takes a few arguments. The first argument is given by using the -d flag followed by a comma separated list of databases to mirror. The database list is the name of the property file in conf/db_properties/ without the .properties suffix. The second argument(s) is a list of tasks to perform. If no task is specified the default mirror tasks is used. Multiple tasks can be chained together by providing a space separated list.

i.e. To only check GenBankEST and Uniprot run

> bin/citrina.sh -d genbankest,uniprot check

To mirror PIR run > bin/citrina.sh -d pir

To run the post-process task and move files for the Pathway database run > bin/citrina.sh -d pathway post-process move

Tasks

Citrina has several tasks that can be performed. The following is a list of these tasks and a brief description of what they accomplish. The full list can be viewed by executing citrina/bin/citrina.sh -p.

Task Name	Description
Mirror	Performs all tasks including checking, downloading, extracting, moving, and pre and post process tasks. This is the default behavior.
Check	Checks the specified database for new files.
Download	Downloads any new database files into the offline staging area.
Extract	Extracts the downloaded database files.
Сору	Copies any files that haven't changed from the online to the offline staging area.
Move	Moves all database files from the offline to the online staging area.
Db-pre-process	Runs the pre processing ant file that is defined in the database property file.
Pre-process	Runs the global pre processing tasks before any database has been mirrored.
Db-post-process	Runs the post processing ant file that is defined in the database property file.
Post-process	Runs the global post processing ant file after all the databases have been mirrored and processed.
Cleanlogs	Removes all files from the log directory.
Cleandblogs	Removes only the log files for the specified database list.

Task Workflow

In most cases the task you will run is the mirror task. This task is responsible for managing the entire mirror process including checking, downloading, extracting, moving, and processing. Below is a workflow that illustrates each step that is taken when the mirror task is executed.

Pre-process --> cleandblogs --> db-pre-process --> check --> download --> copy --> extract --> db-post-process --> post-process --> move

Logging

Each database and each task has a separate log file in the log directory. The filename is usually in the form of <dbname>.<taskname>.log. This information is also output to the main terminal but can get too crowded and confusing especially when running multiple threads. If a problem occurs during the mirroring process you should scan these files in the order they are executed.

Pre and Post Processing

There are four types of processing scripts used by Citrina during a mirror process. The first two types are the global pre and post processing scripts. These are Ant build files that define a set of tasks. They are executed before anything has been done and after all databases have been processed respectively. By default these two files do not do anything and can be found in citrina/projectfiles/default. To modify them to perform certain actions copy the files from citrina/projectfiles/default into citrina/default and edit accordingly.

The second set of files are the database pre and post processing scripts. These scripts are defined in the database property file and should be located in the projectfiles directory. With the exception of the supplied Medlinenew database property file, the supplied database property files do not define pre and post processing. To add a pre or post processing file to a database workflow edit the database property file in conf/db_properties, adding the build file name to the appropriate line. Then place the build file in the citrina/projectfiles directory.

Since all these files are Ant build files that are simply executed by Citrina they need to be in the correct form. For information about writing Ant scripts please see http://ant.apache.org. I also recommend the Java Development with Ant book by Erik Hatcher and Steve Loughran. If you already have some scripts that you use to process database files you can incorporate those into your build file by utilizing the <exec> task. This task is used to execute external programs from within Ant.

For examples of processing build files that utilize the exec task you can have a look at the srsbuild.xml and medlinenew_post.xml build files in citrina/projectfiles. The srsbuild.xml file is used as as global post processing file and automatically launches SRS indexing jobs on the new data. The medlinenew_post.xml file executes a Perl script that removes duplicate records in the Medline update files we download from the NIH.

If any of your processing scripts edit database files before they are moved online then it is best to set do.link.copy property in the database file to false. For more information about this see the section Configuration->Copy Phase Links.

Citrina BLAST Tasks

Included with Citrina are some Ant tasks that can be used to generate fasta files, run formatdb, and run blastall. To use these tasks include this line in your Ant build file:

Fasta task

The fasta task can be used to automate the generation of fasta files. This task runs any user supplied script to do its work. If more than one file is passed to the script this task will call the script once for each file appending the output to the output file. Here is a summary of all the available options and an example.

Parameters:

dir - Directory where you want to run the fasta script

binary - Path to script that you want to use for the fasta conversion.

The full path is only required if the program is not in your PATH.

output - Output file where fasta formatted files are stored. Parameters as nested elements.

Parameters as nested elements:

args - Arguments to pass to the script. Must be enclosed in Ant's arg

task. http://ant.apache.org/manual/using.html#arg.

files - Files passed to the fasta script. Must be an Ant fileset. http://ant.apache.org/manual/CoreTypes/fileset.html

e.g. This converts all the pdb files in /home/johnsmith/dbs/pdb/ into one fasta file called pdb in the /home/johnsmith/fasta directory.

Formatdb task

The formatdb task can be used to automate the generation of blast databases from fasta files. If multiple fasta files are passed to this task, formatdb will be executed once over each file.

Parameters:

dir - Directory where you want to run formatdb.

binary – Path to the formatdb binary. The full path is only required if the program is not in your PATH.

Parameters as nested elements:

args – Arguments to pass to formadb. Must be enclosed in Ant's arg task. http://ant.apache.org/manual/using.html#arg. Do not pass the -i flag because this is handled automatically.

files – Files passed to formatdb. Must be an Ant fileset. http://ant.apache.org/manual/CoreTypes/fileset.html

Note: Be sure not to mix protein and nucleotide files within the same fileset or

formatdb will generate incorrect results or fail to run.

Blastall task

The blastall task can be used to automate execution of the blastall binary. If multiple query files are supplied to this task blastall is executed once for each file using the same options each time. Output from each run it appended to the output file.

Parameters:

```
dir – Directory where you want to run blastall.
binary – Path to the blastall binary. The full path is only required if
blastall is not in your PATH.
program – Blast program to execute (blastn, blastp, etc...).
output – Output file to store the blast results in.
```

Parameters as nested elements:

args – Arguments to pass to the script. Must be enclosed in Ant's arg task. http://ant.apache.org/manual/using.html#arg. Do not pass the -i and -p flags as these are handled automatically. queryfile – The query file to pass to the blastall program.