Generate users – manual

Plugin	hu_skawa_genusers (for Elgg 1.7.2+)
Version	1.0.0
Author	András Szepesházi
Licence	GNU GPL v2.0

Introduction

This plugin generates test users for your Elgg installation. Depending on options provided by the administrator, generated users will have custom profile icons, friend relationships, messages sent to each others, blogs, uploaded contents, and comments on blog entries and uploaded content.

This is a plugin for site administrators, ordinary community members will not benefit from it in any way – it will only be visible in the Administration panel. If you are a community site owner, and you can estimate the future number of potential members of your community, you can use this plugin to “generate them” before going live with your site - and can make performance measurements, checking the future load of your server. If you are a plugin developer, this plugin will create a fake but lively community for you, with active members, messages, blogs, etc. – so you can test your new plugin’s user interaction related functions.

This plugin should never, under any circumstances, be used on a live site already having real users and an active community. If anything goes wrong during user generation, there can – and probably will – be inconsistencies in the Elgg database (not to mention that you probably don’t want fake users mixed with your real ones in a single installation). Worst case scenario, you might have to drop the whole database and install a new, empty Elgg database from scratch. You’ve been warned. You can read more about this in the “Warnings” and “Direct database manipulation” section of this manual.

Think of this plugin rather as a pre-deployment utility, that will help you test site performance and community functions in a sandbox, making your deployment process more thorough and safe.

As of version 0.91 beta, the plugin comes with JMeter integration. JMeter, in a nutshell, is a great open source application, that will allow you to test your site under pressure. This means you can unleash any number of your generated users onto your website simultaneously and have them perform different kind of activities. They will log in, read their messages, upload new files, post comments on blog entries, and do a lot more, just like real users would do. While your fake users do their activities, you can check your server’s load, average response times, deviations and other indicators. All that will help you to scale your server(s) properly for a real-life environment.

Prerequisites

- An empty Elgg installation with a single admin user (required version: 1.7.2)
- Following mods enabled: messages, friends, blogs, files, hu_skawa_genusers
- max_execution_time set to 0 in your php.ini file, at least for the time while users are being generated or deleted
- PHP memory limit set to 128M in your .htaccess file (Elgg installation directory)
- JMeter 2.3.2 or higher installed, if you’d like to create and use performance tests for your site

Warnings

⚠ The plugin uses direct database insert statements instead of the Elgg API for creating users and their content. This practice is clearly against Elgg's recommendations, and generally should be avoided. The only reason I did it this way was performance: the Elgg API was slow, and made it impossible to generate, say 50,000 users with an average number of 100 friend relationships for each user. As a result, the plugin's current version works only with the database structure of Elgg 1.7.2. If there will be any changes to the database in future Elgg releases, they will automatically break this plugin's functionality. If you try to run this plugin with a previous or future Elgg version with a different database structure, I can't even predict the mess that you'll end up in.

⚠ Although I'm using batched insert statement, the generation is a lengthy process. There is a very strong chance that even with relatively low number of users, friendships, messages, etc. the runtime will exceed the default PHP script execution limit, i.e. 30 seconds. Hence the requirement of setting PHP max_execution_time to zero, enabling the script to run as long as it takes. If you do not have rights or access to modify this PHP init value, you should either avoid using this plugin, or stick to very small set of users and other entities. You can read about estimated runtime later, in the "Performance" section of this manual.

⚠ If the script gets interrupted while generating users - either by the "max_execution_time" limit or by you, not being able to wait until it finishes its job -, the database integrity will be broken. Really, make no mistake here. There might be orphan records or records that point to non-existent entries in several tables. In that case, your best bet is to drop the whole database and recreate it – something I had to do frequently while writing and testing this plugin.

⚠ This is a very greedy script on server resources. If you are planning to generate large number of users with a lot of activity and user related content, the runtime might be several hours – or even days! - with a very heavy and constant load on your server. If your site is installed in a shared environment, you should never increase the default parameters (50 users, 5 friend relationship, 5 messages, etc.). With default parameters the generation process should not take more than 20-25 seconds on a reasonable server. Read more about this in the "Performance" section of this manual.

Installation

Installing the plugin is pretty straightforward, as with all Elgg plugins.

1. Download the plugin (hu_skawa_genusers_1.0.0.zip) into your Elgg installation's "mod/" directory, and unzip it.

2. On UNIX/Linux systems, make sure that file ownership and access rights match the following requirements. The whole hu_skawa_genusers directory and its content should be readable and executable for your web server user, and the "log/" subdirectory should be writeable as well.

So, making it simple, you should do something like this in a standard LAMP environment:

```
cd /var/www/your-elgg-site-directory/mod
chown –R www-data:www-data hu_skawa_genusers
chmod –R 755 hu_skawa_genusers
```

3. Log in to your Elgg's front end as administrator, and then go to the Administration / Tools Administration menu.

4. Enable the hu_skawa_genusers plugin. Also enable all plugins listed in the "Prerequisites" section of this manual.

5. In the Administration menu, you will see a new item called "User generation". This is the place where you can generate your test users, or delete previously generated ones.

6. If you'd like to use performance tests that come with this plugin, you'll also have to install JMeter v2.3.2 or higher. You can find all JMeter related information at http://jakarta.apache.org/jmeter/.

Upgrading from a previous version

If you already installed a previous version of this plugin, the update process is extremely easy. Simply replace the content of the whole "mod/hu_skawa_genusers/" directory with the contents of the latest zip package, check again if the file ownership and access rights comply with what's listed in the Installation chapter, section 2, and you are good to go.

The really good news is, if you previously generated a bunch of users with an earlier version of this plugin, and now you'd like to create a JMeter test to check your site's load, you can do this without deleting your old users and recreating them with the new version.

Generating users

As of version 0.91 beta, the plugin comes with a tabbed user interface. When generating your users, you can set up user related parameters on the first tab, and JMeter test related parameters on the second tab. Once you've generated your users, you'll have the option of creating any number of different JMeter test scenarios afterwards. So no need to re-generate users, just because you'd like to try out a new test plan.

User parameters

On the "User parameters" tab, you will find several options to control the size of the site you will eventually generate. These are:

Path to you mysqldump command (like /usr/bin/ or C:/Program Files/MySQL/bin/). Enter the absolute path pointing to your mysqldump executable. Don't forget the trailing slash. This was added in version 1.0-RC1 to avoid cross-platform issues when making automatic database backup. Leaving this field empty will force the program to try to execute a simple "mysqldump" command with the Elgg database parameters, which works anyway on most platforms if your path system variable includes the MySQL bin directory.

Number of users to be generated (1 - 1,000,000 users). This is pretty straightforward; simply enter how many users you'd like to have on your site. Default value is 50.

⚠ The largest "community" I generated was 50,000 users with a lot of additional content (icons for users, and an average of 100 friends, 100 internal messages, 2 blogs, 2 comments and 2 file uploads for any given user). This took several hours on my development server. So be very conservative with providing large numbers for this and the following options. You might end up with a week runtime, seriously.

Username prefix. Users will be generated with a user id starting with this prefix, and having trailing numbers, starting from one and ending with the previously provided number of users. In other words, if you leave this field to the default 'user', and generate 100 users, then you will have 'user1', "user2', ... , 'user100' identifiers for your new users. Prefix should be 4-8 alphanumeric characters, starting with a letter.

Default password. All test users will be generated with the given password. Password should be 6-10 characters.

Create icons for users from image template files. If checked, images from the icon resource directory will be randomly fetched for each user, and all required thumbnails for icons will be written to the user's Filestore.

⚠ If you decide to generate profile icons for your users, remember that this means 6 different image files for each user (number of differently sized thumbnails required by Elgg). So if you have 100,000 users, that means 600,000 file creations on your disk, with 50 to 100Kbytes required storage space for each of those users. You get the picture. Make sure you have enough space on your disk, and be patient, as icon creation for a large number of users really takes a long time.

Average number of friends. This is the number of friend relationships that your generated users will have on average. To make the site more life-like, I'm using a lot of randomizations during generation. The actual number of friends for any given user will be a random number between zero and two times of this value (rand(0,num_of_friends*2)).

Average number of internal mails. This is the number of messages that your generated users will "send" on average. Average here has the same meaning as for number of friends. Recipients will be randomly selected from generated users.

Average number of blog entries. This is the number of blogs that your generated users will "create" on average. Average here has the same meaning as for number of friends.

Average number of comments on blog entries. This is the number of comments that your generated users will "create" on average, for a randomly selected blog. Average here has the same meaning as for number of friends.

Average number of files uploaded. This is the average number of uploaded files created by your generated users. Currently only image files having an extension of ".jpg" are supported. Images will be randomly fetched from the "uploads" resource directory, and all required thumbnails for images will be written to the user's Filestore.

⚠ If you decide to generate uploaded content for your users, remember that this means 4 different image files for each upload (3 differently sized thumbnails plus the original image). Same warning applies as for user icons: depending on the number of users and the number of average file uploads, a large amount of storage space might be required, and the runtime might be very long.

Average number of comments on uploaded files. This is the number of comments that your generated users will "create" on average, for a randomly selected uploaded file. Average here has the same meaning as for number of friends.

Now, if you checked and filled all user related parameters, it's time to go to the next tab where you can set up JMeter test parameters.

JMeter test parameters

Create Apache JMeter test for generated users. Check this box if you'd like to use JMeter tests for your site, uncheck if you don't. If you don't, you can ignore the rest of this section and continue with user generation by clicking on the "Generate…" button at the bottom of the screen. JMeter test creation, unlike user generation, is a very quick process, almost regardless of the number of users you generated, or the parameters you enter here on the JMeter tab. It should take less than a second.

Number of generated users to keep online during load test. This is the number of simultaneous users you'd like to unleash on your site during the load test. Naturally, this number has to be less than or equal to the total number of generated users. Test users will be randomly selected from the pool of generated users.

⚠ This parameter - along with the next few parameters - will decide how much pressure will be put on your server during the test cycle. Again, as I mentioned at the user parameters section before, be conservative with these values, and preferably start with lower numbers. If you try to unleash 10,000 simultaneous, very active users on your site, it might simply collapse under the number of requests.

Ramp up period for all test users in seconds. For each test user, a standalone thread will be started in JMeter – the ramp up period is telling JMeter how much time it should take to start all these threads. If you have 50 test users and a ramp up period of 10 seconds, that means that users will become active at 0,2 second intervals, one by one, and within 10 seconds all your test users will be active, creating requests on your site.

Total duration of a user session in seconds. This tells JMeter how much time a user should spend online and perform activities on the site. Initial login and logout at the end do not count into this period, so the actual test cycle for each user will be slightly longer than this value.

Minimum idle time between two activities for any given user in seconds. Any test user will wait at least that long between two consecutive http requests.

Maximum idle time between two activities for any given user in seconds. Any test user will wait not longer than this value between two consecutive http requests.

⚠ These last two parameters – minimum and maximum idle time - define how active your users will be during the load test. Providing large number of test users and low numbers for minimum and maximum idle time is a good recipe for boiling your server.

Selecting desired user activities

There are a number of predefined user activities that you can choose from. Each user, after logging in, will select a random activity from those that have been enabled, perform it, wait for a random period of time (between minimum and maximum idle time), then perform another random activity, and so on. This goes on until the user thread reaches the predefined total duration value, upon which the user will log out and the corresponding thread will quit.

Some of the activities, like commenting on a random blog entry, consist more than one individual HTTP requests. In multiple-request activities, a random period of idle time will be inserted between each consecutive HTTP requests.

Here is a summary table of all the available activities, and their request details:

Activity	Request details
View dash board page	Single request to pg/dashboard
View friends page	1. Request friends page of logged-in user 2. If has friends, select a random friend and load his/her profile page
View “friends of” page	1. Request “friends of” page of logged-in user 2. If has “friends of” connections, select a random user from the list and load his/her profile page
Check messages in inbox	Single request to “Inbox” page
Send a random message	1. Load “Write a message” page 2. Select random message title and text from the resource file, select a random generated user as recipient, and post message.
View own profile	Single request to user’s own profile page
Update “About me”	1. Load own profile page 2. Load modify profile page 3. Fill “About me” section with random text, and save profile
Read a random blog entry	1. Load all site blogs page 2. Go to a random page on the paged list of blogs 3. Select a random blog entry on the loaded page, and read it
Comment on a blog entry	1. Load all site blogs page 2. Go to a random page on the paged list of blogs 3. Select a random blog entry on the loaded page, and read it 4. Post a random text comment on the selected blog entry
View a random uploaded file	1. Load all site files page 2. Go to a random page on the paged list of files 3. Select a random file on current page, and view it
Comment on an uploaded file	1. Load all site files page 2. Go to a random page on the paged list of files 3. Select a random file on current page, and view it 4. Post a random text comment on the selected file
Upload a new image file	1. Load “Upload a new file” page 2. Select random image file, title and description from the resource files and upload the file.

After filling in all parameters and checking/unchecking activities to be performed in your test, you can hit the “Generate users now!” button. Both users and the JMeter test files will be generated in one go.

Progress tracking

As of version 0.95 beta, the plugin comes with a visual progress tracking feature. As you hit the “Generate users now!” button, a third tab will be added to the GUI, and on this tab you will see a list of all the activities to be done during the generation process.

For each of those activities, you’ll see a progress bar, an “elapsed time” and an “estimated time left” counter. These bars and values should provide server-independent and fairly accurate values during the whole process. There is a total progress indicator at the bottom of the screen, that is far less accurate. I tried to do my best to use dynamic weights for estimating the total progress, however don’t be surprised if the actual generation time will be wildly out of range of the predicted time value.

Also, there is a log file, that you can check for progress, residing in the “log/” directory. This logs all database inserts (in a condensed form) that are executed along the way. On a UNIX/Linux system, you might want to “tail –f” this log file, and see the insert statements that got executed. All lines in this file should end with an “OK.” label – if you see any “ERROR!” label, then one of the insert statements failed, which is pretty bad news. In that case please send the whole log file to my e-mail address (aszepeshazi_at_skawa_dot_hu) and I will do my best to fix the bug as soon as possible.

Working with your generated users

After successful termination of the script, you will see a paged list of all your generated users (with 10 users at a time).

Once you’ve some generated users in the database, you can’t generate more users on top of previous ones. Your only option is to delete the previously generated users, and start a new generation with different parameters.

To check out your new users and their “activities”, simply open a new web browser instance and log in with one of the newly created user ids and the password you provided on the generation page. You will see, that your user has some friends, blog entries, comments, etc. depending on the parameters you provided before. All in all, you’ll have a site with a lot of activity.

Executing the JMeter test

Depending on what computer you are planning to run the JMeter test from, you should either follow the instructions in the “Local testing” or in the “Remote testing” section. “Local testing” is for a server contained environment (where you use a single computer acting both as client and server), “Remote testing” is for a single client-server architecture, where you run the test on one computer and the requests are served by another one.

Local testing

If you have JMeter installed on the same computer where your Elgg site resides, and you’d like to run the test locally, i.e. on your Elgg server, then it’s really easy to do so.

1. Start JMeter (version 2.3.2 or higher) on your server, select “File -> Open” from the menu, and navigate to the “your-elgg-site/mod/hu_skawa_genusers/JMeter” directory. Open the file called “elggloadtest.jmx”.

2. You’ll see an expanded tree of the test plan in the left frame of JMeter window. Navigate to the bottom of the tree, and click on the item called “Summary Report”.

3. In the top menu bar, select “Run -> Start”. Your test users will start their activities.

While the test is running, at the top-right-hand corner of the window you will see two numbers (like 22/50) followed by a green square. This indicates the number of active users / number of total users at the current moment. When it falls back to 0 / 0, and the green square becomes gray, your test has ended.

In the summary report you’ll see important indicators, like minimum, maximum and average response times broken down by request types. Also, you can run “top” in Linux/UNIX systems, or “Task Manager / Performance” in Windows systems to check how your server copes with the load. Additionally, what I personally found very useful, you can log on to your Elgg site in a browser while running the test, and try to perform different activities. This is how your users will see your site and its response times under the predefined load.

If you see anywhere a greater than “0,00%” value for the “Error %” column, then some of the requests failed during test execution. Unfortunately, you will not be able to see what went wrong at this point (maybe you could dig into your http server log, but there is a better way). Also, it’s a good indication of an error if you see suspiciously fast response times (like a couple of milliseconds). So if you encounter any errors, you can do the following:

In the left side frame of JMeter window, enable the “View Results Tree” and “Debug Sampler” nodes (right click on the node, then select “Enable”). Press CTRL + E to clear all previous test results. Start the test again, and this time watch the “View Results Tree” listener in the main window. For each request, you’ll see the request itself, the response from the server, and the resulting document. Whenever you see a red node in the results tree, that indicates a failed request. From this additional information you might figure out what went wrong during the test – for example I got a couple of “502 Bad Gateway” responses when I heavily overloaded my server.

Remote testing

I use the expression "remote testing" for the scenario where you'd like to use a single separate client computer to run the test, and all requests will be sent to your server. This has nothing to do with what JMeter documentation calls remote testing – they're talking about distributed testing there, a very powerful JMeter feature involving several test clients for the same test.

So if you have a single client / single server setup for your test, here is what you should do:

1. After generating the JMeter test plan on your *server*, download the whole "your-elgg-site/mod/hu_skawa_genusers/JMeter" directory and it's recursive contents *to your client computer*, saving everything to any location you like.

2. Unfortunately, you'll have to edit the "files.csv" file manually (if you are lazy to do so, file uploads will not work, however the rest of the test will still run). This file lists a number of absolute paths to image files, that will be used for image uploads during the JMeter test cycle. As the content of this (and the other files) were generated on your *server*, naturally *these paths will not be valid on your client*. (I'm sorry, I couldn't figure out how to use relative paths for the upload file parameter, thus saving you from this trouble). So either edit this file, and correct all the file paths so they point to valid locations, or simply compile a new file with any number of image file paths you like, that can be found on your client computer. The file format is one absolute path per line, enclosed in double quotes. Save the edited file for future use.

3. From here own, you are good to go. Start JMeter (version 2.3.2 or higher) on your *client computer*, select "File -> Open" from the menu, and navigate to the directory where you saved the downloaded "hu_skawa_genusers/JMeter" directory content. Open the file called "elggloadtest.jmx".

4. Follow from point 2. of the previous chapter (Local testing).

Advanced JMeter

JMeter is a very powerful, yet easy to learn-and-use software for load testing your server. If you're interested in more detailed result analysis or more accurate (distributed) testing over the network, you should read the JMeter documentation, and start modifying the test that this plugin has created for you.

I especially recommend two sections from JMeter documentation to read and start experimenting with. Section 18.3 of the User's Manual talks about listeners. Listeners are the way to visualize and interpret resulting data from your tests. The other one is Section 15., talking about how to set up a distributed test environment. Distributed tests will give you more accurate results than tests run in a single client environment, and also with the help of several client computers you can really see the cracking point of your server. (If that's your intention, but better to know it beforehand, than to figure it out during live operation inadvertently, right?)

Creating new JMeter test scenarios

Once you have a bunch of generated users, this is a piece of cake. Simply go to the user generation menu, and on the second tab of the screen set up your new JMeter test parameters, hit the “Create new JMeter test scenario now!” button, and that’s it. Previous test will be overwritten, hopefully that’s not a problem as you can repeat this any number of times you want, and the whole procedure takes less than a second.

There is only one additional parameter you’ll have to supply, when generating new test scenarios, and that is the default user password. As passwords are stored in the Elgg database in an encrypted form, the plugin won’t be able to extract passwords from the DB, and you’ll have to enter the previously used default password for your test users (so they will be able to log in).

Deleting your previously generated users

Once you’ve some generated users in the database, you can’t generate more users on top of the previous ones. If you’d like to have more/less/differently named/etc. users, your only option is to delete all previously generated users, and start a new user generation with different parameters.

As of version 0.95 beta, there are two options for deleting your users. The old method (“Slow deletion”) is basically a series of “delete“ SQL statements. As some of the joins used in the where clauses are complicated, this can rather take a long time. Your other option is to use the new method (“Fast deletion”) that relies on restoring the database from a previous backup.

You’ll need to make sure that your system complies with the following requirements for fast deletion to work:

- the exec() command should not be disabled by php.ini (check the “disable_functions” variable in your php.ini file)
- the last user generation should have done with 0.95b+ version of the plugin (previous versions did not create database backups when starting the user generation process, so there would be no backup file to restore the database from)
- mysql and mysqldump command line tools should be included in your system path, or you should enter the absolute path pointing to your mysql/mysqldump executable where prompted.

Simply click on the “Delete all previously generated users” on the second tab of your generation page to make them disappear. If you check the box called “Also remove all icon files and uploaded content from disk”, it will do as it says and will physically remove all previously created files from the file store.

There is a progress tracking feature for the delete procedure as well, but it’s quite lousy. As I use roughly one delete statement per one Elgg database table, it is not that easy to estimate the expected runtime of these statements. However, at least you still get some information on the current activity.

Customizing content

User names, icons, text messages and uploaded content

There is a resource directory (hu_skawa_genusers/resources) containing files for user names, profile icons, uploaded content and the text for messages, blog entries and comments. Modifying these files, or replacing them, gives you the opportunity to customize your generated users and their content, tailoring them to your needs. The resource files and directories are:

firstnames.txt – a plain text list of first names that will be used when generating names for users. The file format is one name per line separated by CRLF, without any further delimiters. I supplied a list of Hungarian first names with this plugin, feel free to replace it with any other nation's list of first names.

lastnames.txt – same as firstnames.txt, just with last names. I supplied a partial list of German last names with this plugin, feel free to replace it with any other nation's list of last names. A couple of names in the text file got broken due to a faulty charset conversion, sorry about that. It won't affect the usability of the plugin.

mailtext.txt – The content of this file will be used to create all text entries on the generated site. This includes messages, blog entries, comments and tags. I simply used a part of the standard "lorem ipsum" text in the file, you can change that to whatever you like. For each text entry, a random length part of the text will be fetched from a random position in the text. I do a short pre-procession of this text file, I strip the text from commas, colons, semicolons and dots, and I use whole words when creating the text entries.

profileicons directory – In this directory you will find a number of jpeg images, that – in case you checked the "create profile icons" option during generation – will be randomly associated with your users. Replace them, if you like, and please don't forget, that only files having a ".jpg" extension will be used for profile icon generation. Again, to make the generated site more lifelike, a number of users will *not* have profile icons, even if you opted for icon generation. The more image files you put in this directory, the fewer users will not have a profile icon. During generation all images are fetched into memory, so be conservative with the number of image files you place here, not to run out of memory allowed by PHP for a single script.

uploads directory – In this directory you will find a number of jpeg images, that – in case you provided a number greater than 0 for the "number of uploaded files" option during generation – will be randomly associated with your users' uploaded contents. Replace them, if you like, and please don't forget, that only files with a ".jpg" extension will be used for uploaded content.

Messages, comments and uploaded files during JMeter test

Content created by users while executing a JMeter test can also be fully customized. The editable resource files and directories are the following (all residing in "hu_skawa_genusers/JMeter" directory.

quotes.csv – This is a list of famous quotes, that will be used for various textual content creation during test. The file consists of one quote and it's author's name per line, separated by the “#” character. The following reference table shows you how different textual contents are built from these quotes:

	Quote text	Author
Send a random message	Message text	Message title
Update “About me” information	About me text	Not used
Comment on blog entry	Comment text	Not used
Comment on uploaded file	Comment text	Not used
Upload a new file	Description	Title

uploads directory – contains a number of image files, that (by default) will be used by your test users when uploading a new file to the site.

files.csv - List of file names that will be used for content upload. At JMeter test generation, this file is automatically created, and will contain absolute paths pointing to all the image files residing in the above mentioned uploads directory. You can replace this file containing file paths to other image files, if you’d like. The file format is one absolute path per line, enclosed in double quotes.

Direct database manipulation

The plugin uses direct database insert statements instead of the Elgg API for creating users and their content. This practice is clearly against Elgg’s recommendations, and generally should be avoided. The only reason I did it this way was performance: it’s a lot faster to use bulk SQL statement instead of individual API calls. As a result, the plugin’s current version works only with the database structure of Elgg 1.7.2. If there will be any changes to the database in future Elgg releases, they will automatically break this plugin’s functionality. If you try to run this plugin with a previous or future Elgg version with a different database structure, I can’t even predict the mess that you’ll end up in.

So how does it work? Instead of individual insert statements, all inserts are buffered, the default buffer size is 1,000 records. That proved to be safe during my testing, as it never exceeded the default MySQL allowed packet length. So if you decide to create 50,000 users for your site, this will be executed as 50 insert statements, each consisting 1000 user records. To be more precise, it will be executed as 50 inserts into the entities table, 50 inserts into the users_entity table and 50 inserts into the metadata table (with each of those consisting 4 entries for any single user).

If you feel very adventurous, you can try to modify the buffer size in the actions/generate.php file. Search for the “$insertBufferSize” variable, and increase or decrease it according to your needs. Larger buffer size will significantly speed up user generation – however you’ll most likely have to increase the MySQL max_allowed_packet value (search for that in your “my.ini” file).

Performance

For your information, here are a couple of configurations and their corresponding runtimes. All tests were done on my laptop (Intel duo core, 1.8 GHz, 1 GB RAM, Windows XP SP2, Apache 2.0.58, MySQL 5.0.45, PHP 5.2.6). You can hopefully estimate your expected runtimes based on these information.

First, the default configuration:

Number of users	50
Create icons	Yes
Number of friends	5
Number of messages	5
Number of blog posts	1
Number of comments on blog posts	1
Number of uploaded files	2
Number of comments on uploaded files	2
Total runtime for user generation	20 sec
Total runtime for user deletion (with slow deletion)	2 sec

A medium-sized site with no icons and no uploaded content:

Number of users	20,000
Create icons	No
Number of friends	50
Number of messages	50
Number of blog posts	1
Number of comments on blog posts	1
Number of uploaded files	0
Number of comments on uploaded files	0
Total runtime for user generation	30 min
Total runtime for user deletion (with slow deletion)	1 hour

Just a bunch of users:

Number of users	1,000,000
Create icons	No
Number of friends	0
Number of messages	0
Number of blog posts	0
Number of comments on blog posts	0
Number of uploaded files	0
Number of comments on uploaded files	0
Total runtime for user generation	50 min
Total runtime for user deletion (with slow deletion)	55 min

Troubleshooting and FAQ

Q: I get a “hu_skawa_genusers is a misconfigured plugin” error after enabling this plugin. What do I do?

A: This is most probably because you did not modify the directory ownership and/or rights for the plugin. Check the Installation section of this manual.

Q: I started the user generation/deletion process, and after a while I got a blank (white) screen in my browser. What happened?

A: The script exceeded the maximum execution time defined by max_execution_time in your php.ini configuration file. The other possibility is that the PHP script ran out of memory (tried to allocate more than allowed by php memory_limit variable) At this point, you’ll unfortunately have a database with broken integrity, as the script probably terminated in the middle of some insert statements. Drop the whole Elgg database, recreate it, and either set the max_execution_time variable to zero in your php.ini file, or provide a much lower number for users, friend relationships, etc. Also, increase the php memory_limit to 128M.

Q: I entered 100,000 users and 1,000 average friend relationships, then started the user generation. Hours later the script is still running. What do I do now?

A: Well, nothing to do, really. With the numbers you provided, we are talking about roughly 100,000,000 friend relationships that need to be inserted into the database. Even with the bulk database insert statements, this can be as long as a week. Please read the “Direct database manipulation” and the “Performance” section of this manual.

If you’re unable to wait until the script finishes its job, here is what you do: shut down the MySQL and Apache processes on your server, and restart them. Drop the whole Elgg database, recreate it, do again everything that is required by the Installation section, and start your user generation with smaller parameters.

Q: JMeter application can’t load or run the test file created by your plugin.

A: Sure it can. Make sure that you use correct JMeter version (2.3.2 or higher) and that you tried to load the correct test file (elggloadtest.jmx). There is another test file in the same directory, called test_template.jmx, that might be loaded but no good will come from trying to execute it.

Q: There are a number of requests that failed during the execution of the JMeter test.

A: Yes, that is a tricky one, as hundreds of reasons can cause certain test elements to fail. Network connection problems, very high server load, invalid upload file paths, and so on. The best I can say is to follow what is written in the “Executing the JMeter test / Local testing” section, there are some tips on how to pinpoint the problems causing failed requests.

Known bugs

- Rendering problems with progress report in IE 6: it has flickering images for the progress bars, and also breaks the layout slightly.

To-do list

❖ Currently I have no plans to extend the plugins functionality, however I'll try to fix reported bugs and upgrade to future Elgg versions within reasonable timeframes.