LibreSocial Report – Patrick Szewior

Inhaltsverzeichnis

Done Tasks....2
- IP-Discovery in TestPlugin....2
- RMI-Exceptions during connection attempts from Test-slaves to master....2
- Changed storage paths of LibreSocial-storage and Monitoring-database....2
- Changed verious String comparisons....2
- Snapshots for Monitoring database....3
- Partitioning of Monitoring databases....3
- Changed tree-splitting-factor of Monitoring....3
- Delay between TestCases....3
- Random delay before first TestCase....4
- Changed Splitsize of DistributedDataSctructures (DDS)....4
- Increased Heap-Size of JVM....4
- Combine more rounds in monitoring....4
- Increased number of replicates in network....4
- Adjusted hash-calculation during registration and login....5
- Tested different routing configurations....5

Known Bugs and Problems....6
- Compile problems in eclipse....6
- Logout not working....6
- Login User not found....6
- Messaging Plugin / Chat Plugin....6
- Exceptions during decryption of DDS....7
- Loading https URL might cause crash....7
- NAT / Firewall / Port Forwarding....7
- Crashes when using Monitoring....7

Configuration for AlphaTest....9
- Hardware Requirements....9
- Software Requirements....9
- Network interfaces configurations....9
- Setting up first bootstrap node....10
- Setting up further bootstrap nodes....10

Done Tasks

This chapter containes the performed changes and improvement by Patrick Szewior on LibreSocial.

IP-Discovery in TestPlugin

The IP-discovery in TestPlugin had to be changed. When testing LibreSocial on heterogeneous Systems a bug occurred where some instances of LibreSocial did not discover their own nodes IP address correctly. For instance the master node did not realize that it is the master because the discovered IP was different from the specified IP in the config. It was caused by the implementation of IP discovery where all IP's of the node were read in and then the standard loopback address 127.0.0.1 was removed and then the first IP in the list was taken as the local IP. But on some systems, especially Debian and Ubuntu ones, there is more than just one loopback address, all starting with 127.X.X.X. So the discovery was changed in that way, that carefully all possible loopback addresses are removed from the list, before it is returned.

RMI-Exceptions during connection attempts from Test-slaves to master

While slaves tried to connect to master-node, an exception was thrown by RMI. It was caused by a missing java system property. Without this parameter set, a RMI is not working properly. Luckily it is possible to set the property java.rmi.server.hostname during runtime to the correct IP dynamically in code.

Changed storage paths of LibreSocial-storage and Monitoring-database

During first tests we experienced strange crashes of Monitoring and LibreSocial itself. Those crashes occurred 12 hours after the first instance was started. During those tests the storage and databases were stored in /home/<user>. On the used virtual machines the path /home was stored on a NFS. The reason could not be determined precisely but we are pretty sure that it has something to do with the authentication by Kerberos tickets which probably expire after 12 hours which leads to a short absence of storage and causes LibreSocial to crash. Therefore we changed the storage path to a relative path inside the binaries folder and stored the binaries on a physical disc.

Changed verious String comparisons

Changed various string comparisons from "==" to ".equals". "==" checks if both objects point to the same object and therefore same memory locations. Equals should only check if the value is the same. If equals is not overwritten, it ends up in Object.equals which

literally does nothing else as “==” comparing the two objects. But theoretically it would be possible to create two objects with same value but different object references. It seems that the JVM does recognize same values and points them to the same object. But its probably not a good idea to count on that fact and therefore we should use “.equals” correctly to avoid problems in the future with newer or other Java implementations.

Snapshots for Monitoring database

Due to the fact that LibreSocial still sometimes crashes we experienced loss of monitoring data during automated tests because the databases were corrupt. To avoid this and give us also the possibility of taking a look into the monitoring while the long-term tests are still running, we implemented a snapshot-mechanism. It saves a copy of the monitoring database hourly. So the possible loss of data is minimized to a maximum of one hour depending on configuration. This is still a lot of data, but still better than loosing a complete week of testing.

Partitioning of Monitoring databases

By default all monitored data was stored in a single database and then plotted completely. We added the possibility of storing one-hour parts of the monitoring data. That makes it really easy to get detailed plots of a certain hour from the test and gives a further kind of redundancy of storing the monitoring data.

Changed tree-splitting-factor of Monitoring

The used monitoring is based on SkyEye.KOM and organizes itself in a tree structure. Every child passes its data to the parental node and the parental nodes pass their data to their children. Every of this steps takes a certain amount of time we have to wait. By default the tree-splitting-factor was 2 which resulted in a binary tree. That minimizes the load on every node, but results in a very deep tree and therefore high latency because a message takes a lot of time to travel from the lowest child to the root. This caused some metrics to arrive later at the root that other metrics that in fact referred to earlier actions and created sometimes confusing plots. By increasing the tree-splitting-factor to 5 we get a much flatter tree and this way we shortened the latency and can now be pretty sure that newer metrics do not arrive earlier in the root than older ones.

Delay between TestCases

In the TesPlans the time between the execution can be specified. But it only refers to a block of TestCases. The Time between the execution of the same TesCase multiple times is hard-coded in source code and set to a very short period of time (some milliseconds) That caused a massive overload in bigger tests, probably because new TestCases were started while the older ones were still running. We increased this value to about one minute (some randomness is included) to get more realistic and smooth scenario.

Random delay before first TestCase

When slaves were advised to load TestPlan they started immediately with the execution of the first TestCase. In bigger scenarios that caused problems with the performance of the servers and isn’t realistic at all. So we added a random delay before the execution of the first TestCase to get a more realistic and smoother load in the network. Now also the relations between actions in different metrics can be seen clearer.

Changed Splitsize of DistributedDataSctructures (DDS)

LibreSocial uses DDS for some Plugins to store data in the network and distribute the load across multiple nodes. Because of some strange crashes during long-term-test we decided to set this splitsize to infinite to force the nodes to store a complete structure if responsible for. Because we had further problems with the virtual machines during testing, we are not 100 percent sure if the DDS really caused those problems. At least for testing purposes in rather small networks the DDS can surely be stored on single nodes. When a rather stable test-configuration will be found, it would be interesting to test this scenario with different DDS configs. DDS split-size was hardcoded in source code. We decided to make it configurable in config file.

Increased Heap-Size of JVM

When testing scenarios with high load we experienced a huge use of RAM and thererfore Memoryof JVM. To avoid LibreSocial to crash from OutOfMemoryExceptions we increased the JVM-Heap-Size to 2GB per instance. Be aware of the fact, that the JVM sometimes demands a big enough swap partition on the system! We experienced the JVM refusing to start if there was no swap partition like on most OpenVZ or Virtuozzo virtual machines even if there was enough RAM.

Combine more rounds in monitoring

In Monitoring we sometimes experience unwanted and unrealistic peaks in some measurements. Therefore we should always combine more than one round and calculate the median to get a realistic measurement. Due to bigger tests such peaks are more likely to show, so we increased the number of combined rounds to 5 like it was supposed in some earlier work on LibreSocial.

Increased number of replicates in network

When we started first alphatests we experienced random crashes of LibreSocial which sometimes were not even caused by LibreSocial but the JVM itself. With a rasther small number of Bootstrap nodes that are always online and a small number of replicates like 3 it was very likely to loose data if those nodes crashed. Be aware of the fact that the number of strong nodes that are online most of the time and have many available resources should

always be greater than the number of replicates. Otherwise the replication becomes useless and the network may slow down.

Adjusted hash-calculation during registration and login

During Alphatest we experienced a bug that forces accounts registered from a Linux binary to be used from Linux and the same for Windows. There was some inconsistency of creating hashes during registration and login so we adjusted that to be the same but unfortunately that did not bring the effect we hoped for.

Tested different routing configurations

We tested different configurations for routing. Currently we use parallel and iterative routing with a parallellity of 3. This seems to be a pretty stable and fast working configuration (at least it is the most stable and reliable configuration we found). Configurations with less parallelity or even just iterative routing slowed down a lot after a few days and made the network crash.

Known Bugs and Problems

Compile problems in eclipse

We experienced a very strange situation during compiling LibreSocial in eclipse. Sometimes eclipse destroys its own parameters and produces corrupt binaries. Even if it says that the code was compiled successfully the binary is faulty and produces unpredictable behavior like some Plugins suddenly not working that previously worked perfectly fine. This seems to be a problem with eclipse. Using a fresh install of eclipse with exactly the same configuration and source code solves the problem. This problem is very annoying due to the fact that it occurs randomly and the only way to proof is to recompile with fresh eclipse install.

Logout not working

Currently the Logout from GUI is not working properly. Many exceptions are thrown and the GUI gets blocked. In headless tests with TestPlugin it seems to work – at least thats what the monitoring says.

Login User not found

If user was registered on a Linux machine it is not possible to login from a windows system and the other way around. The user is stored as an item and therefore an ItemNotFoundException is thrown. We suspected a problem with the generation of hashes or the conversion of strings but at least in our opinion it seems to be coded correctly by strictly using utf-8. If it was a problem with calculation of hashed then probably no items from other users on different OS would be found but that works perfectly fine. Here some further research has to be done.

Messaging Plugin / Chat Plugin

Messaging plugin sometimes shows strange behavior and we expect it to correlate with eclipse compile error. When the receiver of a message is offline the messages do not reach him. There are two types of exception thrown in those cases. One occurs on the senders site and says that LibreSocial was not able to find the DDS of receiver and therefore could not deliver the message. The other one on the receiver site says that he could not decrypt the messages on DDS or those messages were null. Due to lack of time we were not able to make a further test with a freshly compiled version to be sure that it does not correlate with eclipse compile error.

Exceptions during decryption of DDS

In some cases a error lets the whole LibreSocial instance crash. This happens during or right before the executions of decryption of DDS. It leads to a crash of the whole JVM and it seems that this happens during some kind of JIT-Compiling event.

Loading https URL might cause crash

TestPlugin offeres the possibility to load TestPlan from web url. When using https this might end up with an Exceptions when the server is using SSL-certificates that are signed by StartCOM or StartSSL because they are not accepted as secure from JVM.

NAT / Firewall / Port Forwarding

A lot of work has been done in this fields but it seems to be still not working sufficiently. There are two main problems. First one is that public servers that have a public routable IP and an additional private IP in the network of the service provider LibreSocial always assumes that this server is behind a firewall and NAT even though it has direct connections to the internet. That leads to the point that this server is seen as week and is not reachable properly – it is reachable but routing shows strange behavior. Sometimes foreign nodes are not able to connect to this server or the private IP is used to route which is not wanted. If two LibreSocial users have the same private local subnet at home LibreSocial sometimes to route message directly to private IP. The other problem is that setting up port forwarding manually in the GUI does not work properly in this case. Even if you tell LibreSocial its external IP and port and check that ports are open it still uses private IP and user external port 0 which complete destroys routing. The problem is that private IP s cannot be completely forbidden in network due to the fact of local testing at home or at work in small networks. A solution has to be found to enable both. Maybe it would be a possibility to do a private subnet check and checking the external address as well. If both are equal then it would be possible to route directly through the private network. That would be probably a good extension for pastry network.

Crashes when using Monitoring

When using monitoring during testing we experienced strange crashes of the network during to crashes of multiple instances. The crashes happened more often if more istances were used. We found out that the load on our systems was significantly lower when we shut off monitoring. We assume that the problem is in monitoring. Monitoring uses multiple hashmaps that seem to be never cleaned up during monitoring even though the data was written to database already. Although hashmaps can theoretically grow infinite without loosing performance the number of elements should be limited through the size of integers because size of hashmap is an integer. It might be that this causes problems because bigger network means more measurement points per second and in total so the hashmap is running full earlier. A possible approach would be to repeat such big tests with

monitoring turned on and log terminal output of all instances to disk and then check for any thrown exceptions. The problem is that LibreSocial is writing a lot to stdout so just piping output of LibreSocial to file causes very very large files within a few hours. Maybe a script could be written that scans the output for Exceptions and just stores the relevant output to disk.

Configuration for AlphaTest

For AlphaTesting we set up a network of 8-10 bootstrap nodes (instances). All instances were running on 1-2 physical servers. For bootstraps we used Linux machines with Debian and Ubuntu. Due to unresolved problems we turned off monitoring in LibreSocial.

Hardware Requirements

The following Hardware was used to run up to 10 instances in parallel:

HP Moonshot Server X2150

- AMD Opteron X2150 Quadcore @ 1,5 GHz
- 8 GB DDR3 RAM
- 32 GB SSD

Software Requirements

Because there is currently no option to setup a network automatically without using the TestPlugin, we needed to do this manually by connecting the instances via GUI.

We require

- A minimalistic desktop environment like lxde-core
- A vncserver
- xfonts-basic to avoid problems with vnc on servers
- firefox
- Java should not be needed because it is packed in the binaries

Network interfaces configurations

LibreSocial has still some problems with correct discovery of own network connection state. That means, that if LibreSocial detects a private IP address on the interface to be used it assumes that it is behind a firewall or NAT and therefore cannot be a “strong node” even if the same interface is assigned a public routable IP. The option to tell LibreSocial before connecting that Portforwarding is set up correctly and it has a public IP is often ignored. Sometimes it even works to connect multiple instances but LibreSocial tries to route through the internal private addresses and therefore the network looses many packets. This espacially happens if multiple virtual interfaces are configured like eth0:1 and eth0:2. The only working and reliable way to configure IP correctly was to set up a dummy interface eth1 and assign the additional public IP to it. After this we had to enable forwarding in kernel. When starting LibreSocial we needed to set Interface manually to

eth1 and NOT auto. This forced LibreSocial to discover the correct public IP and gave us the possibility of AlphaTesting. Creation of dummy interfaces is described well here: http://unix.stackexchange.com/a/152334

Setting up first bootstrap node

1. Copy binary to your server to a dedicated directory that is owned and accessible by a dedicated user for each instance of LibreSocial. That user might be called ls01. The directory should be stored on a physical disc. NFS causes many problems. Make sure all file permissions ans owner are correctly and recursively set to ls01.
2. Start a vncserver as ls01.
3. Log in to vnc and start LibreSocial. Firefox should be opened automatically. Give LibreSocial a few seconds to come up correctly before clicking anything. You can monitor CPU usage with htop to see when it has finished booting.
4. Create a new Network
5. Set the interface to the correct interface manually. Use an available port number.
6. Register as a dummy user. We preferred to call them “zboot” to make clear their bootstraps but to avoid them being in front of all friend proposals.
7. If the user registered correctly, click on Network Information at the bottom of the page. Check if External Address is set correctly to public IP and port! If port is 0 you probably did not configured your network interfaces correctly or used the wrong interface! Internal address should be also set to the exact same address as external.

Setting up further bootstrap nodes

Perform steps 1 to 3 as above with other user for each instance ls02, ls02 … . Then continue as following

1. Search for the network by entering the public IP and port of any of the previous instances. If it does not find a network then try again. If that does not work try another port. Maybe the previous instance crashed in the mean time.

Continue with steps 5 to 7 as above.