A year in LizardFS development
What happened in 2017 and some basic Roadmap for 2018
2017 INTRODUCED
MANY NEW FEATURES
2017 introduced many new features

- NFS 3,4,4.1 and pNFS via Ganesha NFS
- Introduction of ACL support with ACL in memory dedup
- New task engine
- Hadoop plugin (mostly done)
- New C-Client library
2017 introduced many new features II

- Read-ahead caching
- Secondary group support
- Recursive remove
- New documentation
- New Platforms
  - FreeBSD, Fedora
2017 ALSO INTRODUCED MANY CHANGES
2017 also introduced many changes
- ACL support extended to OS/X clients
- Windows port now fully functioning on Windows Linux Subsystem (except for signaling)
- Option to avoid same-ip chunkserver replication added
2017 also introduced many changes

- Chunk server load awareness
- New minimal goal configuration option
- Change to semantic versioning system
- Added correct-only flag to file repair
- New directory entry cache for faster lookups
- New whole path lookup function
2017 IMPORTANT FIXES
2017 most important fixes

- Many CPU hogs tamed in master and chunk server
- AVX support fixed
- Global Locks fixed
- Fixed dangling nodes in defective files list
- LizardFS should now compile properly on ARM systems
- Fixes for bugs in some libJudy implementations
  - More are being still fixed
- Fixed issues with reporting defective files
- Fixed request size in read cache for empty results
LIZARDFS ACL SUPPORT
LizardFS ACL Support

- Based on RichACL standard
- Translated on demand to Windows, NFS or POSIX
- Users, groups and secondary groups are taken from whatever the client supports.
- ACLs are available on:
  - Windows, Linux and MacOS/X clients
  - FreeBSD lacks ACL support in the fuse library so we have no support for ACL on FreeBSD clients yet.
LizardFS ACL Support II

- ACL deduplication
  - Many files have the same ACLs
  - Occupies much less memory
- ACL translation
  - Automatic translation to posix
  - Automatic translation to OS/X ACLs
  - Automatic translation to NFSv4 ACLs
- Full translation tables will show up in the directory RSN :)
TASK ENGINE AND
RECURSIVE REMOVE SUPPORT
Task engine and recursive remove support

- Many user complained about the metadata servers slowing down to a grinding hold on large recursive operations, like recursive remove or creation of large snapshots.
- Reason was the immediate execution of these tasks
- Additionally there was no way to see what tasks are being executed in the metadata servers
Task engine and recursive remove support II

- A task management system was added to the master
- Lizardfs-admin got additional commands to list tasks or, optionally, also stop them
- Jobs are being analyzed and if recognized as to big for an atomic execution, split into smaller tasks.
- Tasks split is done so that the first job it relatively large and subsequent ones smaller
Task engine and recursive remove support III

- For now the task engine is utilized by the snapshot creation and deletion tools and “lizardfs recursive-remove”
- The snapshot tools got two new options:
  - -s to set the size of the “initial batch”
  - -l to ignore changes to the src when creating a snapshot
- “lizardfs recursive-remove” was added, which splits the recursive removal of directories into a range of tasks, making sure that the metadata servers never get overloaded by recursive jobs
READ AHEAD CACHING
Read ahead caching

- Small sequential reads create too many requests to the backend
  - High amount of small network operations
  - High amount of backend operations
- Dynamic read caching accumulates those requests
  - Smaller amount of large operations
  - Lower IOPS requirements
READ AHEAD CACHING IN LIZARDFS
Read ahead caching in LizardFS

- Cache expiration time adjustable
- Maximum windows size adjustable
  - Cache will grow up to this max size per descriptor if needed
- Read ahead cache is set per mount point (option to the mount command)
DOCUMENTATION
The documentation project was started beginning of 2017.
Most installation and configuration work is documented.
Now work is starting on documenting LizardFS management.
Next the development guide will be done.
We are looking for help with all parts, but especially with short entries for the cookbook and the FAQ.
If you would like to help, contact me directly.
GANESHA NFS
Ganesha NFS

- Our core customers urgently required highly available, distributed NFS
- We tested many solutions, none really were supporting all NFS features including pNFS
- Ganesha provides NFS 3,4 and 4.1 and pNFS
- Ganesha already had a record with gluster and others
First tests from LizardFS mount points were not very promising
  - Slow, crashy, unreliable
Then we tried the direct approach, using a modified c-client library to create a FSAL module for ganesha. - works.
We are looking for user feedback, please go ahead and test !!!
LIZARDFS
HADOOP PLUGIN
LizardFS Hadoop Plugin

- Skytechnology had a lot of requests for the usage of LizardFS as a storage for Big Data Projects
- End of 2016 first efforts to implement HDFS on top of a LizardFS cluster
  - No Hadoop lab available
  - Very complex to set up
  - Does not really utilize LizardFS backend features
LizardFS Hadoop Plugin II

- Next implementation used the hadoop c-api
  - Lacked many functions required for Java implementation
  - Added another layer of complexity
- While developing the c-api version the LizardFS c-client lib was born
  - Direct connector now possible
  - Far less complex than the other implementations
  - Much faster than all former versions
LizardFS Hadoop Plugin III

- Ganesha NFS development changed the LizardFS client lib
  - Hadoop plugin needed to be partly rewritten again to fit the reborn client lib
- Multiple obstacles during development
  - Weird errors like “FileNotFoundException is not an instance of FileNotFoundException”
  - LizardFS client lib was still in development
  - No real hadoop users inside the dev team
LIZARDFS HADOOP PLUGIN
CURRENT STATE
LizardFS Hadoop Plugin - current state

- Current Plugin based on released LizardFS client lib
- Feature complete HDFS implementation
- Documentation is being prepared
- Should show up in our Github repo soon
- Looking for help with QA
- Looking for testers for this alpha version
  - Especially ones with real world testing potential

Skytechnology sp. z o.o., ul. Miłobędzka 35, 02-634 Warszawa, www.skytechnology.pl
C-CLIENT LIBRARY / API
C-Client Library / API

- A native way to connect to LizardFS was required
  - For the GaneshaNFS plugin
  - For the Hadoop plugin
  - For other projects
- Started with hadoop and was moved to a more general approach when NFS came around
- Available on all platforms
- Examples included
C-Client Library / API II

- Exposes functions to access the following features:
  - Managing locks
  - Connection management
  - Linking, unlinking, opening, writing and reading objects
  - Getting information about chunk servers
  - Managing ACLs
- All functions are shown in the examples file
  - src/data/liblizardfs-client-example.c
In the main source code archive
2018 ROADMAP
2018 Roadmap

- Open Version of uRAFT HA
- Release the Hadoop Plugin to the Public
- LizardFS-NG - Codename “Agama”
  - More on that on separate slides
- Move Windows platform testing to Windows Subsystem for Linux
  - Mostly ready
  - We can even run chunk server and master here
  - Only signal interpretation can’t be tested
2018 ROADMAP
LIZARDFS “AGAMA”
2018 Roadmap - LizardFS “Agama”

- Heavy architectural changes, mostly focused on performance.
- 2018 should bring the new client which can also work with traditional LizardFS and implement a lot of the changes that are scheduled for metadata and chunk servers.
LIZARDDFS “AGAMA”
GENERAL CHANGES
LizardFS “Agama” - general changes

- Event driven architecture
- Asynchronous I/O implemented with the asio library
  - [https://think-async.com/](https://think-async.com/)
  - Asio implements c++ pre standard on async I/O
- Implemented mostly in user space, avoiding usage of kernel caches etc.
  - Avoids Meltdown/Spectre slowdowns
  - Minimal Kernel Calls
LizardFS “Agama” - general changes II

- New tracing subsystem with Unique Identifiers to be able to correlate debugging between all components (metadata, chunk server, client ...)
  - 64bit Identifiers for transactions
  - Every 1024th transaction is saved with transaction id and can so be synchronized between components
- New cross servers network and I/O monitoring system for autotuning
  - Many timeouts will be automatically adjusted by this information
  - Auto adjustment to fast drives like SSD or NvME
LIZARDFS “AGAMA”
NEW CLIENT
LizardFS “Agama” - new client

- Backward compatible to 3.12
- No Kernel caching
- Own userspace network subsystem
- Full write versioning
  - Allows multiple clients to write to the same chunk without the risk of overwrites or the requirement of absolute locks
  - Safeguards against some rare EC problems
  - Avoids parity mismatch after crash in case of differences in metadata between different metadata servers.
  - Allows for much faster writes, since there are far less wait states involved
LIZARDFS “AGAMA”
NEW CHUNK SERVER
LizardFS “Agama” - new chunk server

- Event driven and aio taken from client
- Simplified architecture
- Userspace Network I/O
- Kernel space transactions limited to a bare minimum
LIZARDFS “AGAMA”
NEW METADATA SERVER
LizardFS “Agama” - new metadata server

- Basics taken from new client (eventing, aio)
- New fully consistent/coherent write algorithms allowing for a distributed metadata setup across multiple servers
  - Distributed custom key/value store
  - RAM cache of hot data
- Plans include a network of active metadata servers
LizardFS “Agama” - new metadata server II

- Utilising new write versioning from clients for faster data distribution
- Automatic timeout adjustment based on new network and I/O monitoring system.
  - Limit amount of “cryptic” configuration
  - Adapts to different environments by adjusting to different storage media and different network environments automatically
URAF - AN ADVANCED FAILOVER SYSTEM
uRaft - an advanced failover system

- Used in LizardFS since 4 years for commercial clients only
- Based on the “raft” consensus algorithm by Diego Ongaro and John Ousterhout.
- Requires at least 2 nodes and a quorum node
- Minimal settings required to run
- Sub second switchover times
- We use it for metadata server HA and Ganesha metadata node HA
uRaft - an advanced failover system II

- Floating IP model
- Fast election process
- Works with LizardFS and GaneshaNFS
- Very simple to set up
  - Handful of lines of config only
  - Config mostly identical on all uRaft servers
- Very low resource consumption
- Will be open source from now !!
- Information on the algorithm can be found here:
  - https://raft.github.io/
# Configuration for node1:

URAFT_NODE_ADDRESS = 192.168.0.1  # ip of first node
URAFT_NODE_ADDRESS = node2       # hostname of second node
URAFT_NODE_ADDRESS = node3:99427  # hostname and custom port of third node
URAFT_ID = 0                      # URAFT_ID for this node
URAFT_FLOATING_IP = 192.168.0.100  # Shared (floating) ip address for this cluster
URAFT_FLOATING_NETMASK = 255.255.255.0  # Netmask for the floating ip
URAFT_FLOATING_IFACE = eth1       # Network interface for the floating ip on this node
uRaft - requirements

- uRaft requires a minimum of 3 nodes
  - Usually 3 master servers
- One node can be setup as a so called “quorum node”, using very little resources and not being an active member but just a voting node
  - Can be piggybacked on a chunk server for example.
- Than only 2 master servers required plus 1 quorum node
- All nodes need to be in the same subnet for the IP failover to work
Where to get more information ...

- Github repository - https://github.com/lizardfs/lizardfs
- Documentation - http://docs.lizardfs.com
- Mailing list -
  https://sourceforge.net/p/lizardfs/mailman/lizardfs-users/
- Irc channel - #lizardfs on the freenode irc network
- Forum - http://www.lizardfs.org/forum
- Community website - http://www.lizardfs.org
- Commercial website - http://www.lizardfs.com/
Questions?
- regarding how things work?
- regarding implementation?
- why we did it?
- other questions?
More Questions?

For more answers just grab one of us during the session breaks . . .

Or come visit our stand in Building K Level 1.
THANK YOU

michał bielicki
community relations manager
m.bielicki@lizardfs.org
http://www.lizardfs.org

Skytechnology sp. z o.o., ul. Miłobędzka 35, 02-634 Warszawa, www.skytechnology.pl