Summer 2021 Internship Report

Click here for the final presentation slide deck.

Main feature (parent):
Issue #24546 - Add the ability to use CVMFS on sites that do not provide local installations

Issue #	Status	Subject
24637	Resolved	Integrate CVMFS feature with GlideinWMS code base
25981	Resolved	Cleanup script not executing correctly at glidein termination/expiration
26090	Resolved	Supporting big files in the glideinwms repository
26093	New	Error reporting for cleanup scripts
26094	Work in progress	Automate the creation of platform-specific cvmfs distribution
26095	New	Allow glideins to use cvmfsexec mode 3
26096	New	Reorganize/Refactor CVMFS setup that happens through glideins
26153	New	Version-based download/build of cvmfsexec distribution tar files

Status info of tickets with respect to #24546 (summer 2021)

Issue # (hyperlink to Redmine)	Branch name (hyperlinked to Redmine)	Subject
24637	v37/24637, v37/24637_1	Integrate CVMFS feature with GlideinWMS code base
25981	v37/25981, v37/25981_v2	Cleanup script not executing correctly at glidein termination/expiration
26090	v37/26090, v37/26126	Supporting big files in the glideinwms repository
26093	-	Error reporting for cleanup scripts
26094	v37/26094	Automate the creation of platform-specific cvmfs distribution
26095	-	Allow glideins to use cvmfsexec mode 3
26096	-	Reorganize/Refactor CVMFS setup that happens through glideins
26153	-	Version-based download/build of cvmfsexec distribution tar files

Tickets directly related to #24546 and their corresponding code branches

Issue # (hyperlinked to Redmine)	Subject
24636	Incorporate code review feedback into scripts
24635	Prepare separate scripts for mount and unmount of CVMFS
24634	CVMFS test results matrix for different platforms
24511	Install GlideinWMS framework on Fermicloud
24502	Complete orientation and training for the GlideinWMS project

Previous tickets related to #24546 (summer 2020)

GWMS documentation written), mainly to capture the status and have some starting material that we could use for a future paper

The worldwide distributed computing infrastructure consists of a diverse set of computing resources (“sites”). Some of these resources, especially those used for high-performance computing (HPC), may not provide the CernVM File System. The CernVM File System (CVMFS) is a distributed file system used by scientists in high-energy physics collaborations to distribute their software libraries and experiments’ data. The primary goal of the internship project was to find a solution that would make CVMFS available on HPC sites when a local installation of CVMFS was not available. Additionally, we also wanted our solution to minimize the effort of site administrators to install CVMFS locally on the nodes.

We designed and developed a feature in GlideinWMS that enables CVMFS to be provisioned on demand. GlideinWMS This is achieved by extending the current functionality of the glidein (pilot job) to install CVMFS on the worker node when not locally available. By leveraging unprivileged user namespaces, FUSE interface and a provisioning tool, the glidein makes CVMFS available using the most reliable option for a given worker node. This enables scientists to run analyses and simulations everywhere as GlideinWMS ensures that CVMFS is always available on all computing resources, including the ones where it is not natively available.

For provisioning CVMFS, we use cvmfsexec. cvmfsexec is an open-source package that supports unprivileged CVMFS. CVMFS can be mounted as an unprivileged user without having to be installed by a system administrator. Using unprivileged user namespaces and FUSE

framework, cvmfsexec can mount CVMFS in unprivileged mode in four different ways (https://www.github.com/cvmfs/cvmfsexec):

1. (mode 1) using mountrepo/umountrepo
2. (modes 2 and 3) using cvmfsexec
3. (mode 4) using singcvmfs

Using cvmfsexec, distributions with CVMFS software and configuration can be created for multiple platforms based on various sources from which latest cvmfs and configuration rpms are to be downloaded. Further customizations are possible as cvmfsexec allows additional CVMFS configuration settings before creating a distribution file. Further, the package also allows creation of self-contained distributions and distributions for mounting CVMFS inside a container. Our mutual collaboration with one of the developers which led to direct interactions and resulted in software improvements. Having direct access to the developer helped understand the software and, more importantly, to request new features with quick implementation.

The Process

- **Summer 2020**

  1. **Feasibility Study**

     At first, a feasibility study was conducted to understand the mounting and unmounting behavior with cvmfsexec. We tested on various platforms and validated against expected behavior by cloning the cvmfsexec repository. Following were the system parameters which formed the basis for determining feasibility:

     - operating system distribution
     - operating system version
     - unprivileged user namespaces available/supported
     - unprivileged user namespaces enabled
     - FUSE installed, fuser mount available
     - user in fuse group

     (testing matrix figure from the presentation)

  2. **Prototyping**

     a. **Design:**

System checks

- Platform (rhel6, rhel7, rhel8, centos, other)
- Kernel (2.x, 3.x, 4.x, other)
- Unprivileged user namespaces supported/available
- Unprivileged user namespaces enabled
- FUSE installed and fusermount available
- User in fuse group

CVMFS detection and mounting

- Check for local installation of CVMFS
- Mount CVMFS using cvmfsexec package if not locally available

Unmount previously mounted CVMFS when glidein terminates/expires

b. Implementation:

- Three bash-compliant shell scripts
  a. **`cvmfs_helper_funcs.sh`**
  b. **`cvmfs_mount.sh`**
  c. **`cvmfs_umount.sh`**
- `INFO/WARN/ERROR` messages to improve output/error/debug messages
- Inline documentation to aid code readability
- Code modularization, standard logging mechanism
- Addition of unit tests using BATS to ensure code quality

3. Working Feature

(1) Configured a **custom script** using the parameters in the glidein config file
- **cvmfs_setup.sh** — imports helper functions and invokes the CVMFS mount script

(2) Manually created **tarball** containing auxiliary files
- **cvmfs_utils.tar.gz** — contains
- **cvmfs_distros.tar.gz**: Utilities to mount/unmount CVMFS: platform- and architecture-specific distributions
- **The three scripts**: (a) helper functions, (b) mount script and (c) unmount script

- Added (1) and (2) to the Factory configuration file (via *<files>*) to ship to the glidein-customized node

- Large number of distributions created for various combinations of platform- and architecture-specifications
    - Extra level of granularity with osg and egi configuration repositories for CVMFS

- **Summer 2021**

1. Integration with GlideinWMS codebase

- Added the custom script and the tarball to the default list of uploads
- Added three attributes to the Factory config:
    - CVMFS_SRC — enables selection of CVMFS repos based on the source, i.e. config repository (`osg`, `egi` or `default`)
    - GLIDEIN_CVMFS — for better error handling behavior in case of errors encountered during mounting of CVMFS (`required`, `preferred`, `optional` or `never`)
    - GLIDEIN_USE_CVMFSEXEC — whether the tarball should be unpacked (`1`) or not (`0`)
- Patch fix for correct execution of cleanup script at glidein termination/expiration (#25981)
- Incorporated logic for un-mounting CVMFS with additional logic for locally installed vs. glidein-based CVMFS
- Use of error_gen.sh for reporting success and failure messages during the the execution flow
- Feature is being released in GlideinWMS v3.7.5

2. Code and workflow optimizations

a. Supporting big files in GlideinWMS

- Tarball with utilities and helper scripts for CVMFS provisioning — BIG!
    - Need for alternative solution to store big files
- Considered Git-LFS and Git-Annex which are possible ways of implementing version control for large files
    - Our big file requires no versioning — the latest version is what would be used
- Developed symbolic link-based solution (by Marco Mambelli)
    1. Added a new directory *bigfiles* to the codebase; no content tracking
    2. Hosted the tarball on the glideinwms website
    3. Added scripts to upload and download the files to *bigfiles*
    4. Used symbolic links to access the downloaded “big” file

b. Dynamic creation/selection of cvmfsexec platform-specific distribution

- `cvmfs_utils.tar.gz = cvmfs_distros.tar.gz + a few scripts`
    - `cvmfs_distros.tar.gz` — static tar file containing multiple cvmfsexec distribution files (corresponding to a CVMFS source, system platform and architecture)
    - `cvmfs_utils.tar.gz` file transported to the glidein — inefficient as only one distribution file is needed for customizing the worker node
- generate_cvmfsexec_distros.sh — automatically generate all possible combinations of CVMFS source and platform-specific distribution files (packaged as individual tar files)
    - Invoked at the time of reconfig/upgrade (more dynamic)
- Modified GlideinWMS code to add the generated distributions (as tar files) to the default list of uploads at the time of factory reconfiguration/upgrade
- cvmfsexec_platform_select.sh — automatically selects the appropriate distribution tar file based on the specifics of the worker node
    - Invoked during worker node customization by the glidein
    - Reduces the number of distributions that are shipped as artifacts (ONE versus many)

c. Code reorganization

- Tarball only contains the scripts at this point; no necessity to package the scripts as a tarball
- Modularize the entire code for CVMFS provisioning such that one script serves as both an executable and a source file
    - Instead of having separate scripts with helper functions, functions for mount and un-mounting of CVMFS

Tarball with cvmfsexec package is downloaded separately; only the scripts need to be downloaded by the glidein; so there is no need to package them
Distro-tarball is independent of the auxiliary tarball with the dynamic creation/selection.

- **Next Steps**
    - Enable glideins to use mode 3 of the cvmfsexec tool (#26095)
        - cvmfsexec command can handle clean unmounting of CVMFS repositories; even when the processes are hard-killed (kill -9)

    - cvmfsexec spawns a sub-process in a namespace unshared from the parent process
        - Find a way to make the glidein configuration variables visible inside the new namespace
- Version-based selective updates for cvmfsexec distribution files (#26153)
    - How to determine if re-downloading and rebuilding the cvmfsexec distribution files is needed on subsequent reconfig/upgrade of the factory
- Error reporting for cleanup scripts (#26093)
    - No reporting information found in client logs in the Factory from the cleanup procedure
- Test on RHEL8 and SuSE platforms

References

- GlideinWMS Documentation
- https://glideinwms.fnal.gov/presentations/intro/GlideinWMS.pdf
- CernVM File System Docs
- cvmfsexec - https://www.github.com/cvmfs/cvmfsexec
- Unprivileged User Namespaces - https://lwn.net/Articles/532593/
- FUSE - https://www.kernel.org/doc/html/latest/filesystems/fuse.html
- Project codebase - https://www.github.com/namrathaurs/gwms-cvmfs

Acknowledgements

This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.