Agenzia nazionale per le nuove tecnologie,
l'energia e lo sviluppo economico sostenibile

Kerberos5, Slurm and OpenAFS integration on ENEAGRID infrastructure

OpenAFS Workshop, 11 June 2024

M. Fois*, F. Pascarella, G. Guarnieri, G. Santomauro, F. Palombi, F. Adragna, F. Ambrosino, D. De Chiara, A. Funel, G. Cascone, F. Bianchi, L. Acampora, M. De Rosa, N. Fonso, G. Bracco, F. Iannone, G. Ponti.

Presentation Outline

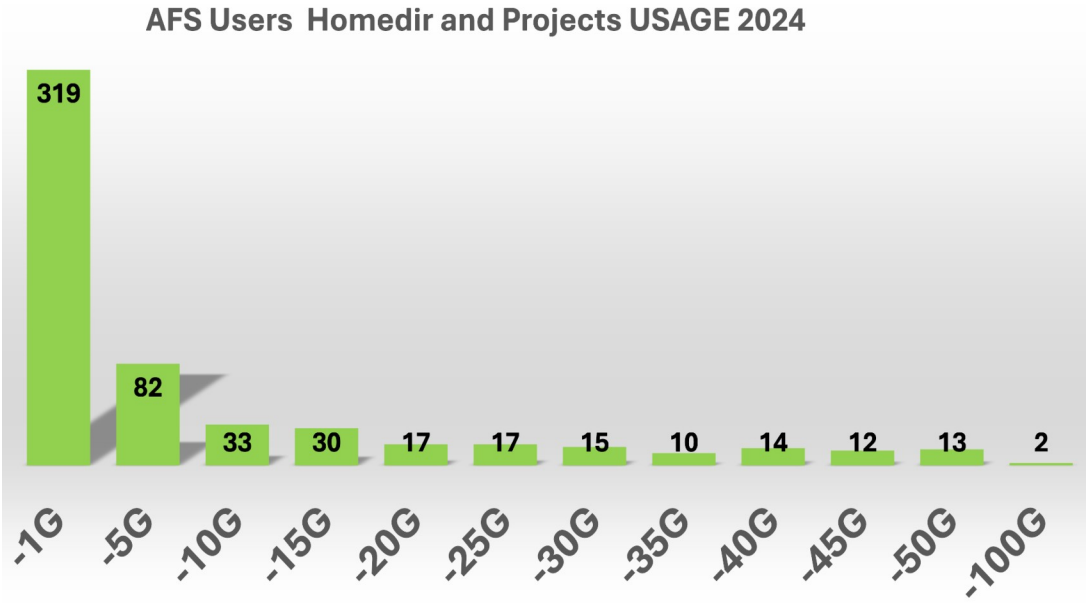
- **ENEAGRID**
  - AFS Cell Report and Clusters Overview
  - CRESCO7-XCRESCO HPC Clusters
- **Slurm-Kerberos-AFS Integration**
  - Kerberos Authentication
  - Problem: Kerberos Authentication on Compute Nodes
  - Slurm SPANK Plug-ins
  - Auks: Overview and Setup
  - Integration with Slurm: Auks SPANK Plug-in
  - Problem: OpenAFS Tokens on Compute Nodes
  - Auks SPANK Plug-in Patch
  - Conclusions

ENEAGRID

- Multisite integrated computational infrastructure developed by **ENEA.**
- 12 sites in Italy, 6 with relevant ICT resources and 3 with **HPC clusters**. Connections by **GARR**.
- Portici (**POR**) is the main HPC site, followed by Frascati (**FRA**) and Casaccia (**CAS**).
- ~4PF total computational power, divided in 6 clusters.
  - **OpenAFS** user homes and shared software.
  - **GPFS/LUSTRE** high performance filesystems.
  - **Kerberos5** authentication (7 virtualized KDCs).
  - **LSF/Slurm** job schedulers.

ENEAGRID: AFS Cell Report

- **enea.it** cell status (2024):
  - ~4000 volumes, ~23.5 TB.
  - 6 virtualized DB servers, 10 file servers (SuperMicro Storage Server with RAID6).
  - CentOS 7.9 (planned upgrade to AlmaLinux 9.2).
  - OpenAFS **1.8.11** (up from 1.6.24).
  - Fully **decentralized backup** strategy: each main site backup its volumes + Frascati-Casaccia dumpfiles collected in Portici and saved in GPFS.



ENEAGRID: Clusters Overview

Cluster Name	Cores/Nodes	Peak Rate	CPU	RAM/Node	Net	HPF	OS	Site
CRESCO6	20832/434	1.4 PF	INTEL SKL 8160	192 GB	OPA	GPFS 3.8 PB	Centos 7.3	POR
CRESCO7*	6912/144	0.5 PF	INTEL SKL 8160	192 GB	IB EDR	LUSTRE 1PB	AlmaLinux 9.2	POR
CRESCO4F	1024/64	20 TF	INTEL E52670	64 GB	IB QDR	GPFS 48 TB	Centos 7.8	FRA
CRESCO4C	512/32	10 TF	INTEL E52670	64 GB	IB QDR	GPFS 33 TB	Centos 7.8	CAS
CRESCO5F	320/11	~100 TF	AMD EPYC 7313 A100/H100	256 GB	IB EDR	NFS 134 TB	RockyLinux 8.5	FRA
XCRESCO*	21292/60	~2 PF	IBM Power9 V100	256 GB	IB EDR	NFS 134 TB	AlmaLinux 9.2	FRA

*new clusters in 2024

CRESCO7-XCRESCO: New Software Stack

- Testbed for the upcoming **CRESCO8** cluster (~10PF, expected in late summer 2024).
- Operating System: AlmaLinux 9.2 (replaces CentOS 7).
- Filesystems:
  - OpenAFS 1.8.11 (user homes, project data and shared software).
  - LUSTRE 2.14.0 (CRESCO7), NFSv4 (XCRESCO) (high performance scratch areas).
- Authentication: Kerberos V5 MIT 1.15.1.
- Job Scheduler: Slurm 23.11.5 (replaces LSF).
- Completely **open source** stack!

CRESCO7: Front

CRESCO7: Back

Kerberos Authentication

- Network Authentication protocol for un-trusted environments with simmetric encryption, developed by the MIT.
- Provides mutual authentication (authenticates both sides of a communication).
  - Based on a trusted 3rd party Key Distribution Center (**KDC**), which assigns time stamped encrypted tickets (**TGT**) to clients after authentication.
- Enables Single-Sign-On (**SSO**) in un-trusted environments.
  - Initial credential acquisition enable access to kerberized services and nodes throughout the environment.
  - Better network security (less passwords to write down), improved user experience (workstation to cluster nodes seamless access).
- Implemented in ENEAGRID AFS since 1998.

Problem: Kerberos Authentication on Compute Nodes

- Kerberos tickets are “**short lived**” (but they can be renewed).
- On HPC clusters, users tipically login into a frontend node and submit their jobs to a **scheduler**, which puts them in a queue and executes them when resources are available.
- Moreover, parallel jobs are executed on many different compute nodes.
- Tickets must be **forwarded** to the compute nodes used when the job starts, ensuring kerberized execution where the user is not directly involved.
- Tickets must be **renewed** if the job stays in queue or takes a long time to finish.
- This requires **integration** between Kerberos and the job scheduler.
- LSF scheduler (used on most ENEAGRID clusters) provides native Kerberos integration.
- What about Slurm?

SLURM SPANK Plug-ins

- Slurm doesn't integrate natively with Kerberos. Implements instead a **plug-in architecture** with the so called **SPANK** API (https://slurm.schedmd.com/spank.html).
- Provides hooks to perform various actions at different stages of jobs life cycles.
- Plug-ins are shared objects (.so) written in C, loaded at runtime by Slurm during job execution.
- Implemented plug-in functions are executed at the corresponding job stage, some examples:
  - *int slurm_spank_init (spank_t spank, int ac, char *argv[])* (called when job starts)
  - *int slurm_spank_user_init (spank_t spank, int ac, char *argv[])* (called after privilege drop)
  - *int slurm_spank_exit (spank_t spank, int ac, char *argv[])* (called when job is done)
- Helper functions *spank_getenv, spank_setenv, spank_unsetenv* can view and modify job's environment.

Auks

- **Auks** is a distributed credential delegation system. It provides:
  - Remote cache of Kerberos credentials, used to pull or push granted tickets.
  - Regular renewal of cached tickets.
  - Kerberized service, to ensure authentication and privacy of exchanges.
  - **Slurm SPANK plug-in** (*auks.so*), to use all these tools within Slurm.
- Can be easily integrated in other applications (C API + command line client).
- Developed by Matthieu Hautreux and CEA-HPC, fully **open source**.
  - https://github.com/cea-hpc/auks
- Suggested by the official Slurm documentation.
  - https://slurm.schedmd.com/related_software.html
  - https://slurm.schedmd.com/slurm_ug_2012/auks-tutorial.pdf

Auks: Components Overview

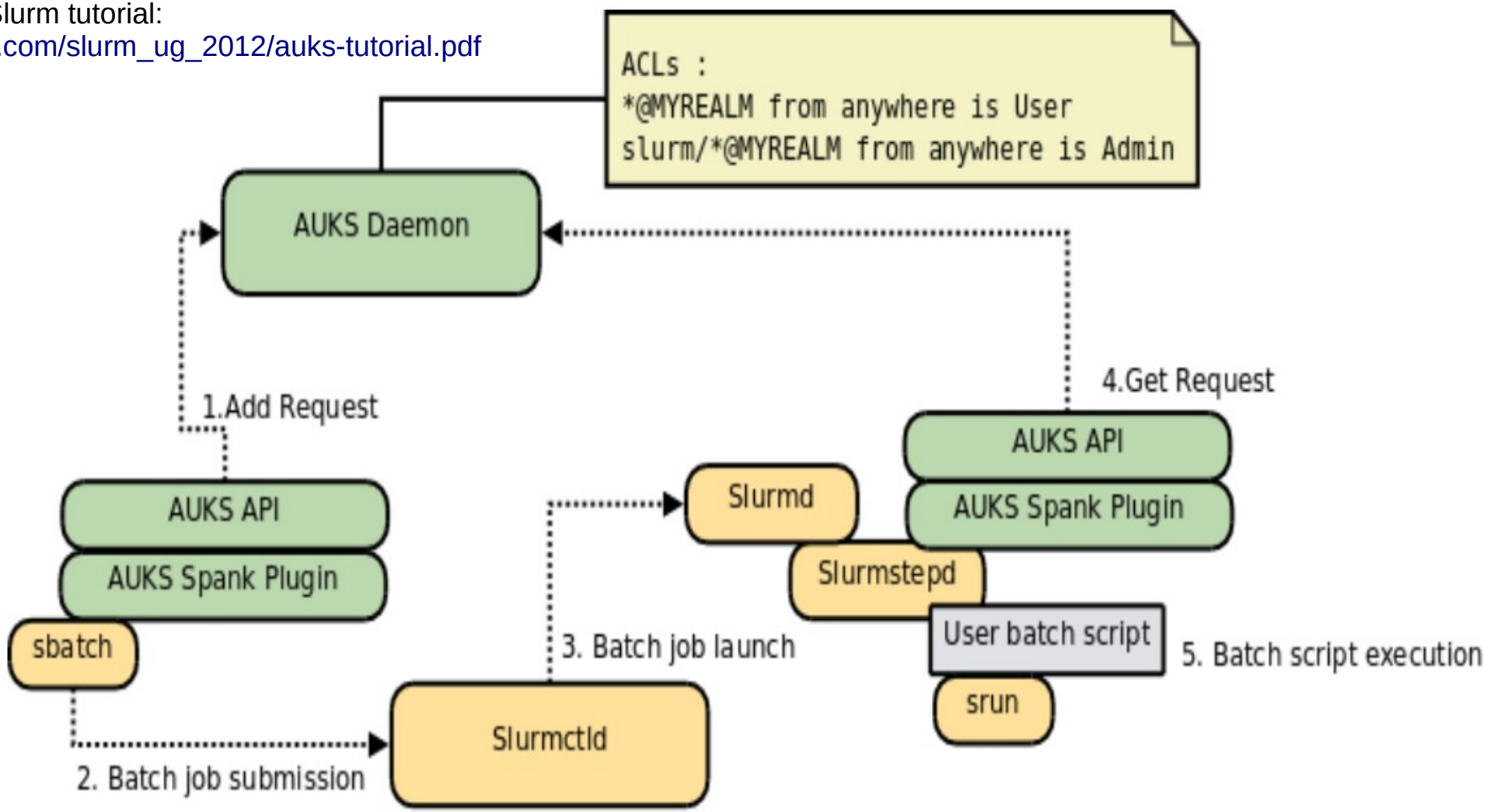
- **auksd,** main auks daemon, multi-thread server written in C**.**
  - Receives add(store in cache)/get(retrieve from cache)/remove(delete cache) TGT requests from clients.
  - Provides a kerberized service, authenticates client requests.
  - Stores TGTs in a cache directory, one per user.
- **auksdrenewer**, daemon implementing TGT renewal mechanism.
  - Separate component due to thread safety issues in Kerberos libraries.
- **aukspriv**, daemon ensuring credentials cache is accessible to:
  - SLURM, for proper credential get action during job execution (via Auks SPANK plugin).
  - auksdrenewer, for proper renew logic.
- **auks**, command line client using Auks API to request add/get/remove TGT to auksd.

Auks: Setup

- Example Scenario:
  - 1 management node (where the auks credentials cache is managed).
  - 1 frontend node (where users login and submit their jobs to Slurm).
  - 1/N compute nodes (where Slurm executes the jobs on users behalf).
- On the management node:
  - auksd, auksdrenewer, aukspriv daemons.
- On the frontend node:
  - auks cli client, Slurm SPANK plug-in.
- On compute nodes:
  - auks cli client, Slurm SPANK plug-in, aukspriv daemon.

Integration with SLURM: Auks's SPANK Plug-in

Taken from the Auks-Slurm tutorial:
https://slurm.schedmd.com/slurm_ug_2012/auks-tutorial.pdf



Problem: OpenAFS Tokens on Compute Nodes

- OpenAFS grants user access by issuing **tokens**, with the command **aklog**.
- If a Kerberos ticket is available, aklog can create a token based on the information in the ticket by contacting the AFS DB servers.
- When a job is spawned on the compute nodes by Slurm, Auks SPANK plug-in ensures the user's Kerberos credentials are correctly forwarded and periodically renewed, but…
- Token generation still needs a call to aklog on every node (otherwise the job processes cannot access AFS).

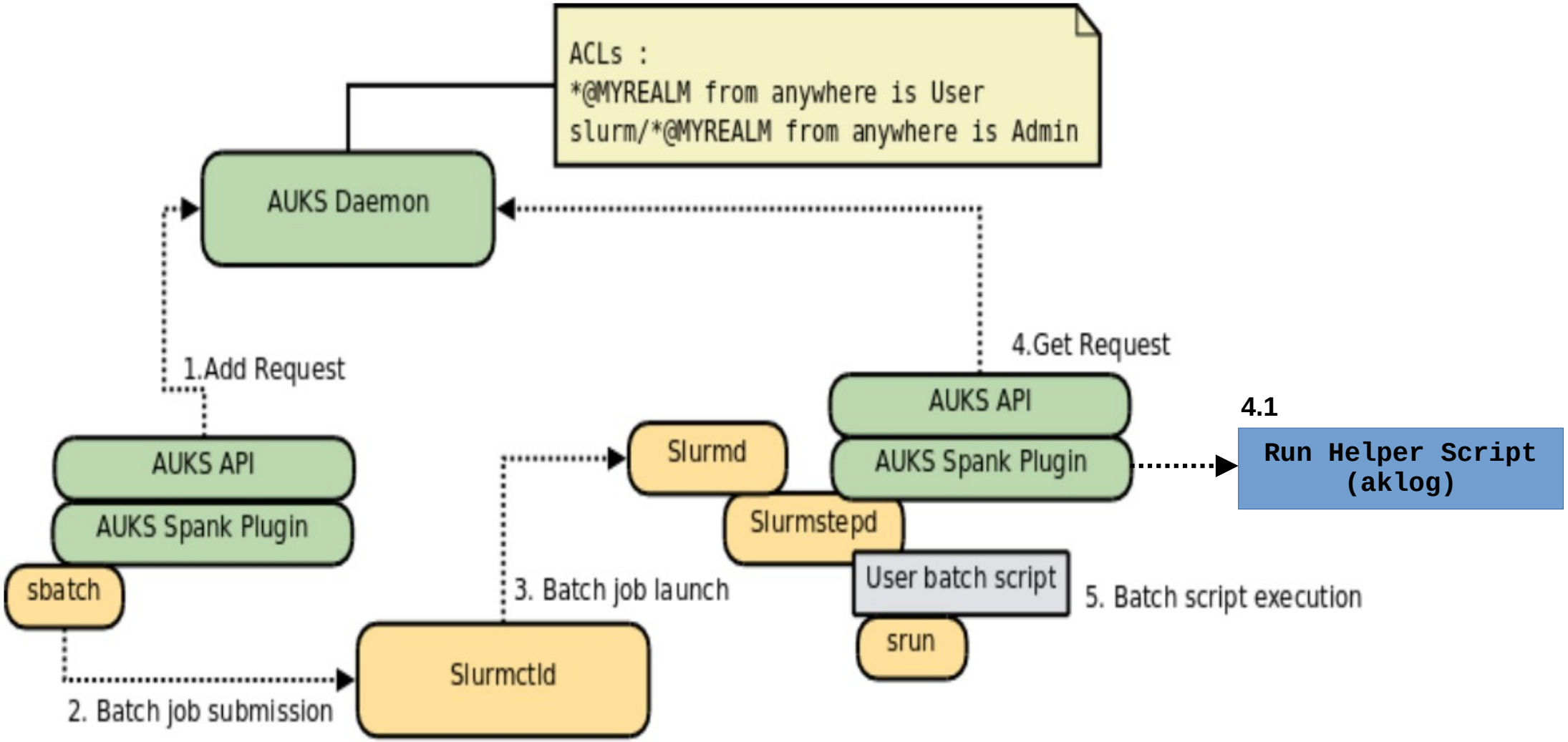
Auks's SPANK Plugin Patch

- Pending pull request on Auks repository: “Add support for a HelperScript option”.
  - https://github.com/cea-hpc/auks/pull/11
- Allows running a user defined script (e.g. aklog) after Kerberos credentials get and renew.
- In use in production environment at CERN since February 2022 (plenty of testing already).

```
135 + \fBHelperScript\fR
136 + optional path to an executable to run during renewal and first cred get operations.
137 + This executable will have the corresponding KRB5CCNAME environment variable set
138 + to enable kerberos integration with other systems (e.g. to run aklog for AFS).
```

```
@@ -634,6 +634,9 @@ spank_auks_remote_init (spank_t sp, int ac, char *av[])
634             if ( fstatus != 0 )
635                     xerror("unable to set KRB5CCNAME env var");
636
637 +           /* fire helper script for the first time */
638 +           auks_api_run_helper(engine.helper_script, auks_credcache, uid, gid);
639 +
```

Auks's SPANK Plug-in Patch



Tokens Protection

- Potential Issue:
  - When a user submits multiple jobs, some of them could be scheduled by Slurm to run on the same compute node at the same time.
  - Kerberos credentials retrieved by auks are specific for every job, but the AFS token will be **shared**.
  - If one of the jobs destroys its token (e.g. by executing **unlog**) every other user's job on the node loses access to AFS!
- Workaround:
  - Set **pagsh** as the shell of the Slurm job scripts and run aklog inside the script.
  - This way the token generated is tied to the PAG shell and unique for the job.

Conclusions

- Setting aklog as HelperScript with the patch allows Slurm jobs processes access to AFS during execution: **full Kerberos-Slurm-OpenAFS integration!** But...
- **Not merged yet!** Some use cases not well covered...
- Main issue: **job's stdout/stderr files on AFS**.
  - Slurm throws an I/O error when attempting to create job's output/error files on AFS (stage prior to job execution).
- Workarounds:
  - Write output/error files on another filesystem (e.g. LUSTRE/GPFS, limits users freedom).
  - **Patch Slurm** to make an explicit aklog call before creating these files (adopted solution).
- Future work: continue **testing** and **improving** the patch, hopefully merged soon.

Thank You!

matteo.fois@enea.it