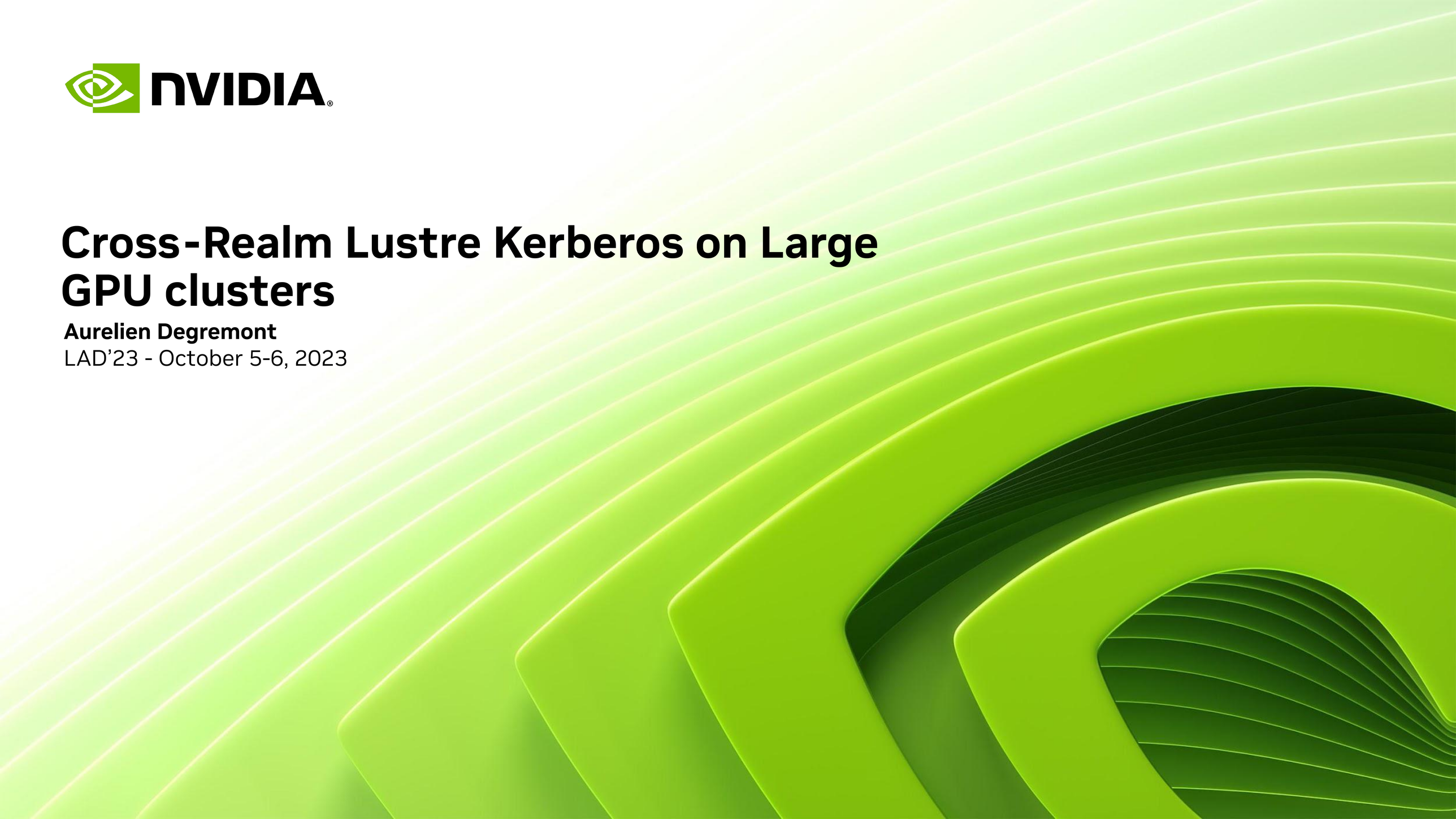


Cross-Realm Lustre Kerberos on Large GPU clusters

Aurelien Degremont

LAD'23 - October 5-6, 2023

Reference platform: the NVIDIA SuperPod

- NVIDIA cluster solution for AI/ML workload
- Scales to hundreds of DGX nodes, made of
  - 2 Xeon CPUs
  - 8 NVIDIA GPU H100
  - 2 TiB of RAM
- Connected with NVLink / Infiniband NDR
- Using Lustre filesystem via DDN Exascaler as one of the available storage solutions

Support Active Directory on SuperPOD

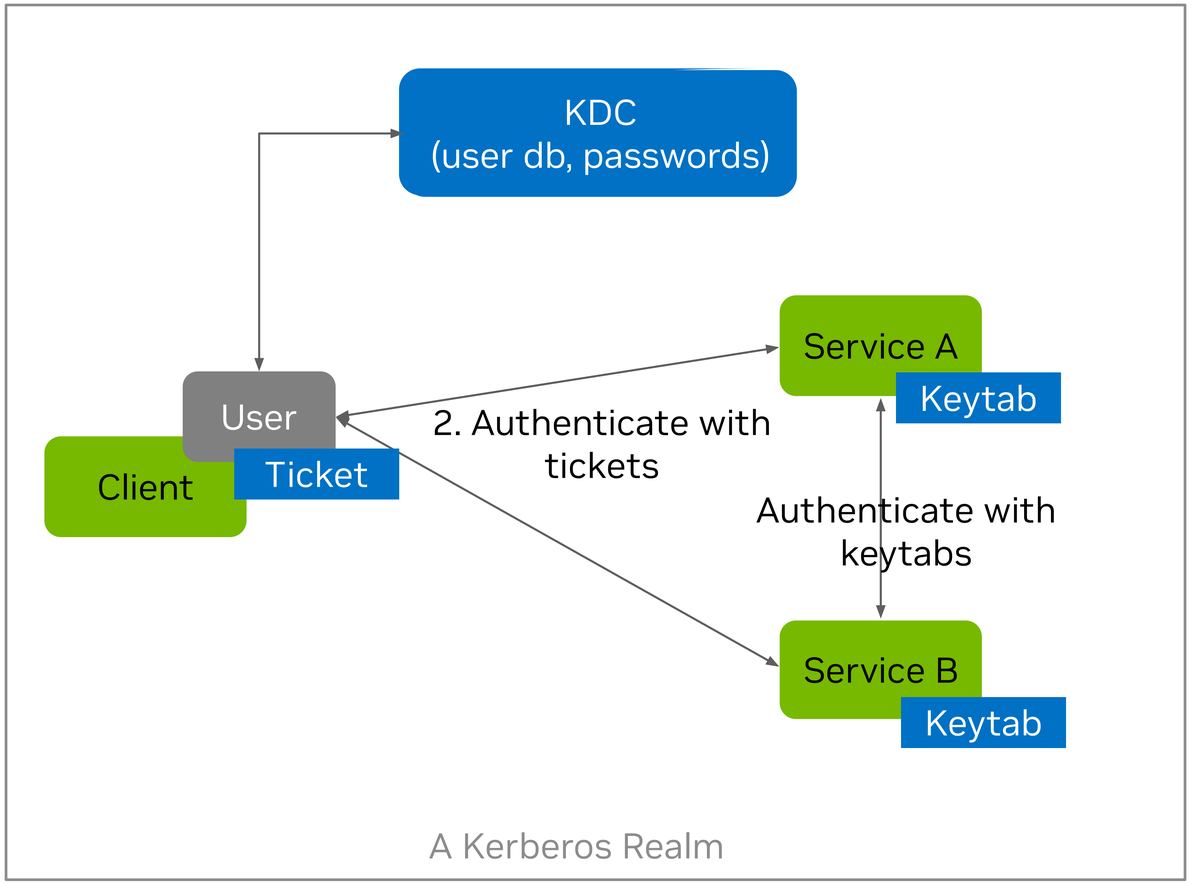
- As large GPU deployments are more and more common in enterprise deployments, there are needs for more security and integration.
- Asks for Microsoft Active Directory support are more common since it is a de facto standard in enterprise environment
- AD gives some benefits like:
  - Managing accounts and passwords
  - Centralizing the user management in one place
- Kerberos is the default Linux available protocol behind Active Directory
- So supporting AD means enabling Kerberos

- Objective here is not to limit that to login nodes but supporting it everywhere, both login and compute nodes, up to Lustre!
  - Improve the security posture, end-to-end

What Is Kerberos?

How does it work?

- Well-established authentication protocol
- Symmetric-key cryptography based
- Authentication is centralized in a KDC
- Kerberos manages:
  - Identities (*principals*) like users or services
  - Credentials (password or keytabs)
  - Domains, named *Realms*.
- Benefits
  - Keep user and password in one place
  - Services can authenticate each others
  - Single Sign on capability
  - Federated environment (cross-realm)
- Supported by Lustre



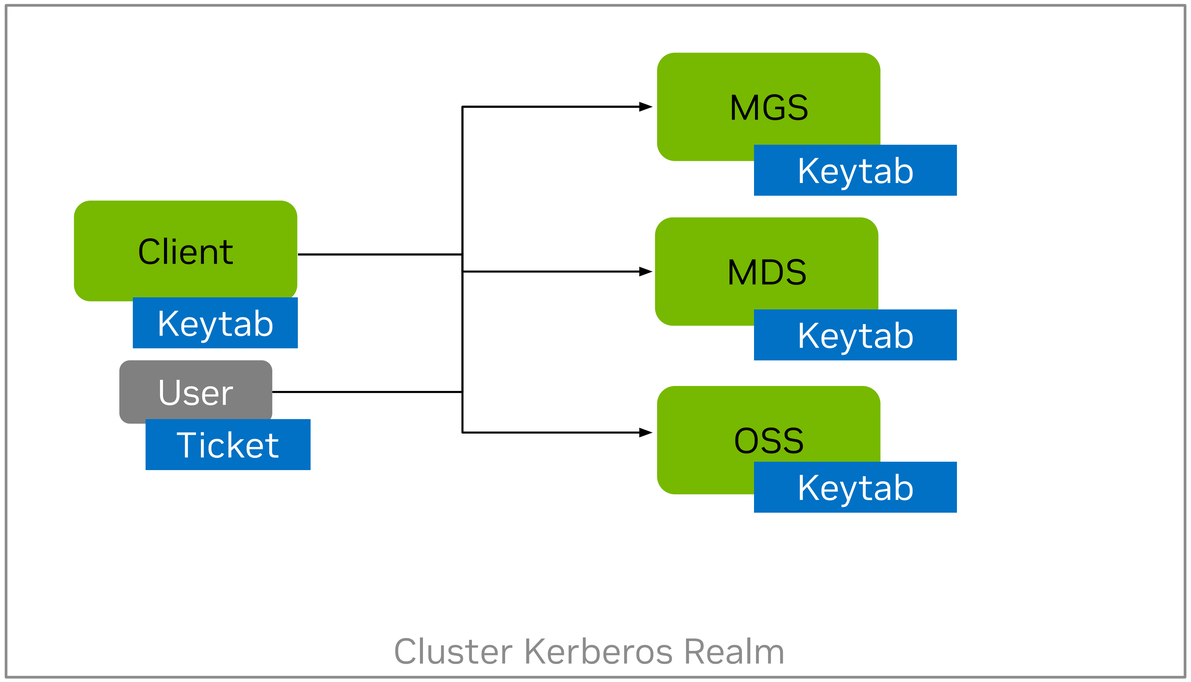
Lustre and Kerberos

Why Kerberos?

- Lustre trusts root anywhere on the network
  - No client validation
  - Servers trust provided UID/GID
  - Anybody becoming root on client can access all data
- Use Kerberos to mitigate it

- Each service and host has its own keytabs
  - Clients too
  - Servers and clients can authenticate each other
  - MDS validates users using their ticket

- Each Lustre component needs a Kerberos principal
  - MGS: lustre_mgs@REALM
  - MDS: lustre_mds@REALM
  - OSS: lustre_oss@REALM
  - Clients: lustre_root@REALM

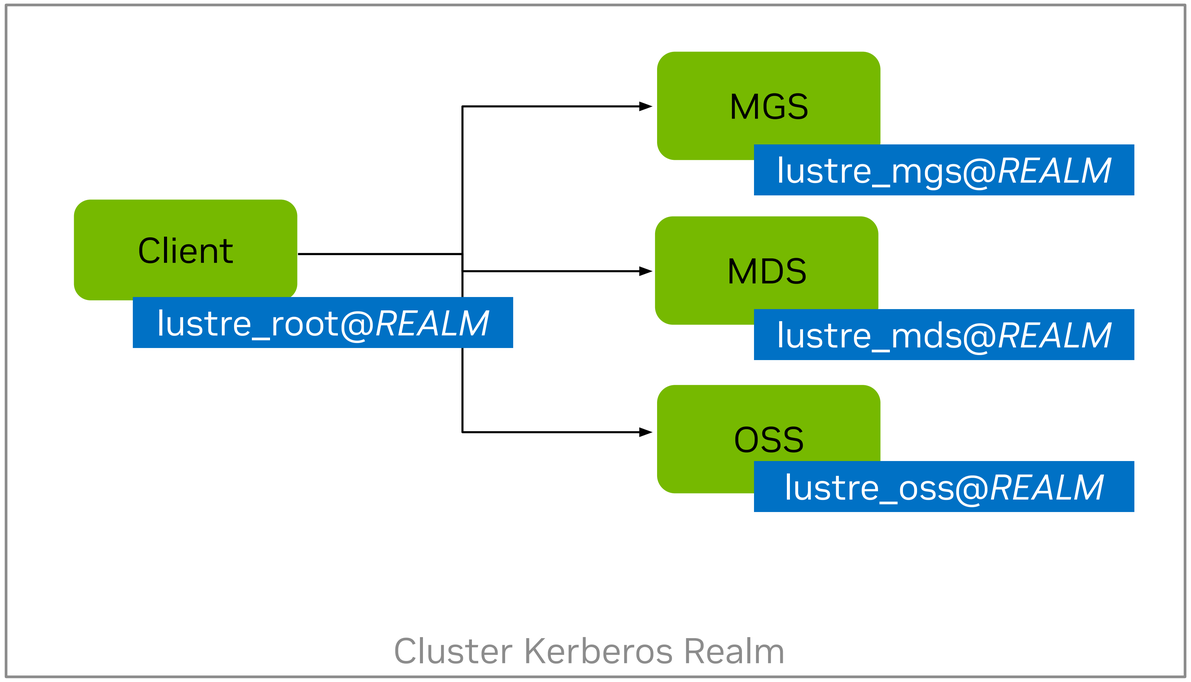


Lustre and Kerberos

How is Lustre working with Kerberos?

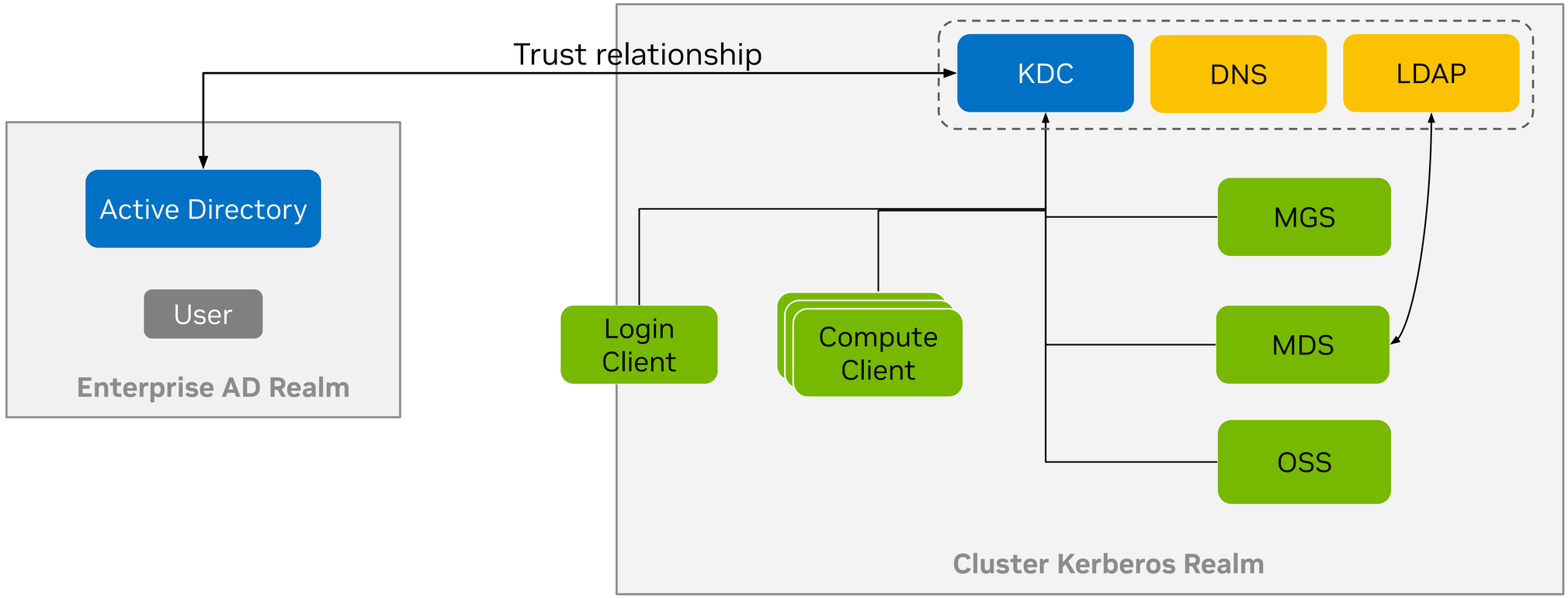
- Then, Kerberos should be enable at each connection level, with different flavors:
  - `null` (default) no kerberos
  - `krb5n` authentication only
  - `krb5a` +header message integrity
  - `krb5i` +bulk data integrity
  - `krb5p` +message privacy (encrypted)

- We only focused on `krb5n` as other modes have a too large performance impact.



Test architecture

- We first ran a test deployment, that helped us identify some limitations.
- We tried to improve security and ease of use



Improvement examples

Better credential caching and lookup (LU-16646)

- **Credentials are cached on client and server side**
  - Lustre originally cached credentials in memory (MEMORY:) or in fixed location on disk (FILE:/tmp/krb5cc_*)

  - Changed to rely on current system configuration (`default_ccache_name`, in krb5.conf)
  - Enable caching credential, on server-side, in Kernel keyring.

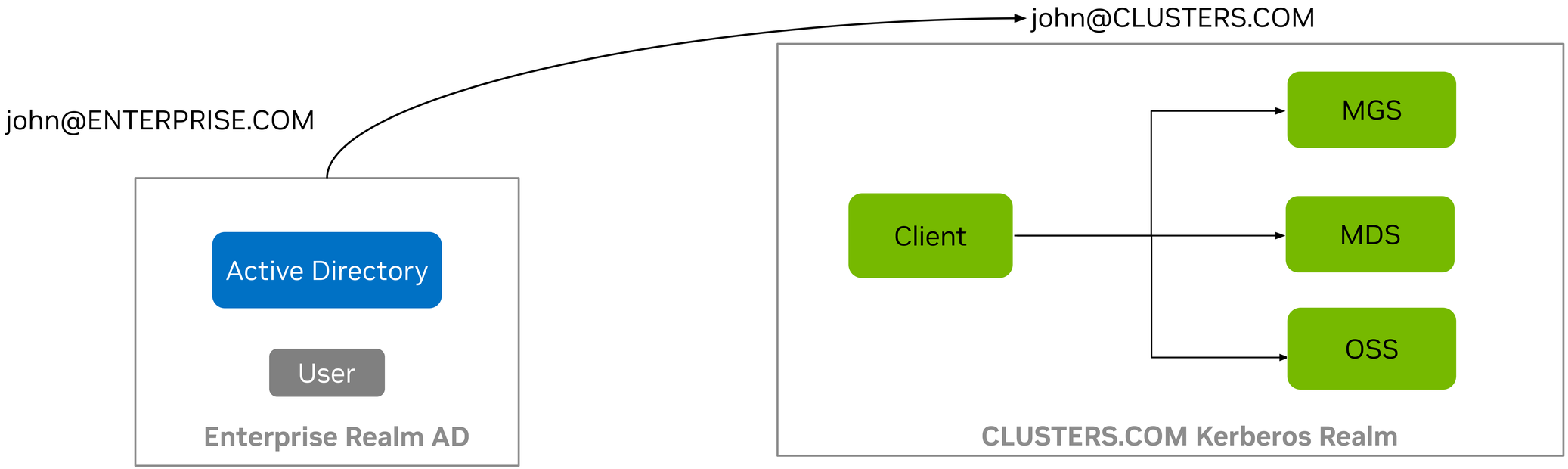
- **Credentials are searched for in hard-coded FILE:/tmp/krb5cc_<uid>**
  - They were guessable, not compatible with the security policy
  - Not where SSH GSS store them

  - Changed to rely on system current configuration (`default_ccache_name`, in krb5.conf)
  - Fallback to `/tmp/*krb5*` for backward-compatibility and to have it similar to NFS behavior
  - Enable looking for credential, on client-side, in Kernel keyring.

Improvement examples

Better support for cross-realm setup (LU-16630, LU-17023)

- Lustre needs to map the Kerberos identity to a local identity. It was doing simply dropping the realm name if it was matching the host realm name, but did not what to do if it was a different one.
- Historically, Lustre only supports a manual mapping with `/etc/lustre/idmap.conf`, which could be cumbersome to maintain, especially if you user list is large
- Switch to standard mapping mechanism `auth_to_local`, in `krb5.conf`
  - Support regexp patterns, file map, etc.
- Add an option to force a different realm than the default one from `krb5.conf` to help setup with using multiple realms.



Improvement examples

And more...

- Failover tuning with a failover group of 4 servers or more
  - Kerberos mode introduces an additional connection retry in case of failover
  - Double connections, plus trying each of the 4 servers successively in the worst case could be longer than recovery timeout
  - Solution is to increase the minimum recovery window to 500 sec.
- Support standard Kerberos host principal (LU-16758)
  - Now supports host/<hostname>@<REALM>, like SSH or NFS.
- Some limitations with supported cryptographic algorithms
- Multiple small bug fixes or improvements
  - Client LBUGs (LU-16532, LU-12896)
  - Logs verbosity (LU-16829)
  - Module unloading (LU-16888)

Using FreeIPA

The large token challenge (LU-17015)

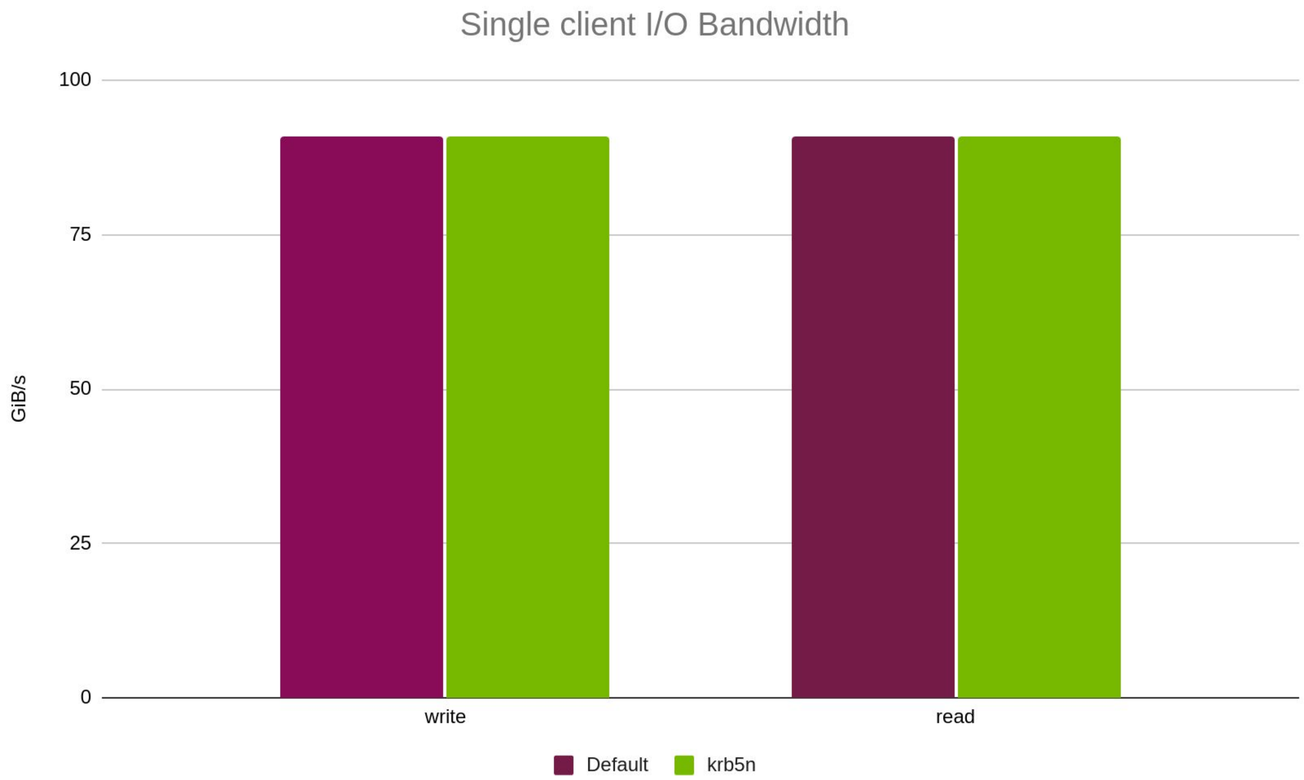
- As the integration efforts with AD can easily get more and more difficult, it is easy to end up building a complex infrastructure, easiest is to move to FreeIPA
- **FreeIPA** is integrated solution for identity/access management.
  - Kind of “Kerberos + LDAP + everything you need” in one place.
  - It is targeting easy AD integration
- Working well… too well.
- Switching to FreeIPA infrastructure ended up having much larger Kerberos tickets, crashing Lustre! (LU-17015)
  - AD is issuing tickets with PAC data (authorization data) which is making them much larger (few hundreds to several thousands bytes).
  - Lustre is relying on old SUNRPC implementation for key cache management (GSS).
  - NFS had the same issue in the past and ended up switching to a totally new implementation (gssproxy).
  - Lustre is reusing the already existing identity upcall cache instead, but this requires lots of adaption.
    - Client is fixed
    - Server patches are getting finalized

How is it performing?

Bandwidth

- Single client I/O bandwidth
  - No impact for large streaming I/O

- Test environment
  - Client
    - 2x Xeon 8480C 56c
    - 2 TB RAM
    - 2x NDR 400 Gb/s
  - Servers
    - 12x DDN Exascaler AI400X2
    - 24x MDTs
    - 96x OSTs



How is it performing?

Metadata

- Single client Metadata performance
- Performance impact is limited
  - -5% average impact
  - Except -23% for directory stats

Conclusion

- Kerberos is deployed on a large Lustre filesystem
- Supporting on a centralized Active Directory with Lustre
- Performance impact is minimal

- Interesting follow ups could be to enable Kerberos for all Lustre communications and test higher security levels

Thank you

- I’d like to thank
  - *Jonathan Calmels* at NVIDIA for all the efforts he put into our Kerberos deployment.
  - Whamcloud team members for the support and fast patch delivery: *Sébastien Buisson*, *Andreas Dilger*, *Peter Jones*


NVIDIA
Questions?