The Key Security Technologies in Lustre

sbuisson@whamcloud.com

emoly@whamcloud.com

Oct. 2020

Challenges and Motivations

- More customers using Lustre as permanent data repository and not only for scratch
- Organizations forced to comply with new standards, rules, methods, etc.
- High Performance file system inserted into the ”Enterprise” workflow requires sophisticated security configuration
- New paradigms: technologies designed and developed with enhanced security in mind

Different Security Requirements

- User/node authentication
  - Only authenticated users have access
  - Only authenticated nodes are part of Lustre
- Access control
  - DAC (Discretionary Access Control)
  - MAC (Mandatory Access Control)
- Multi-tenancy
  - Provides isolated namespaces from a single file system
  - Limited namespace exposed to clients
- Encryption
  - Wire Encryption (Network)
  - Data Encryption (Logical and Physical)
- Audit

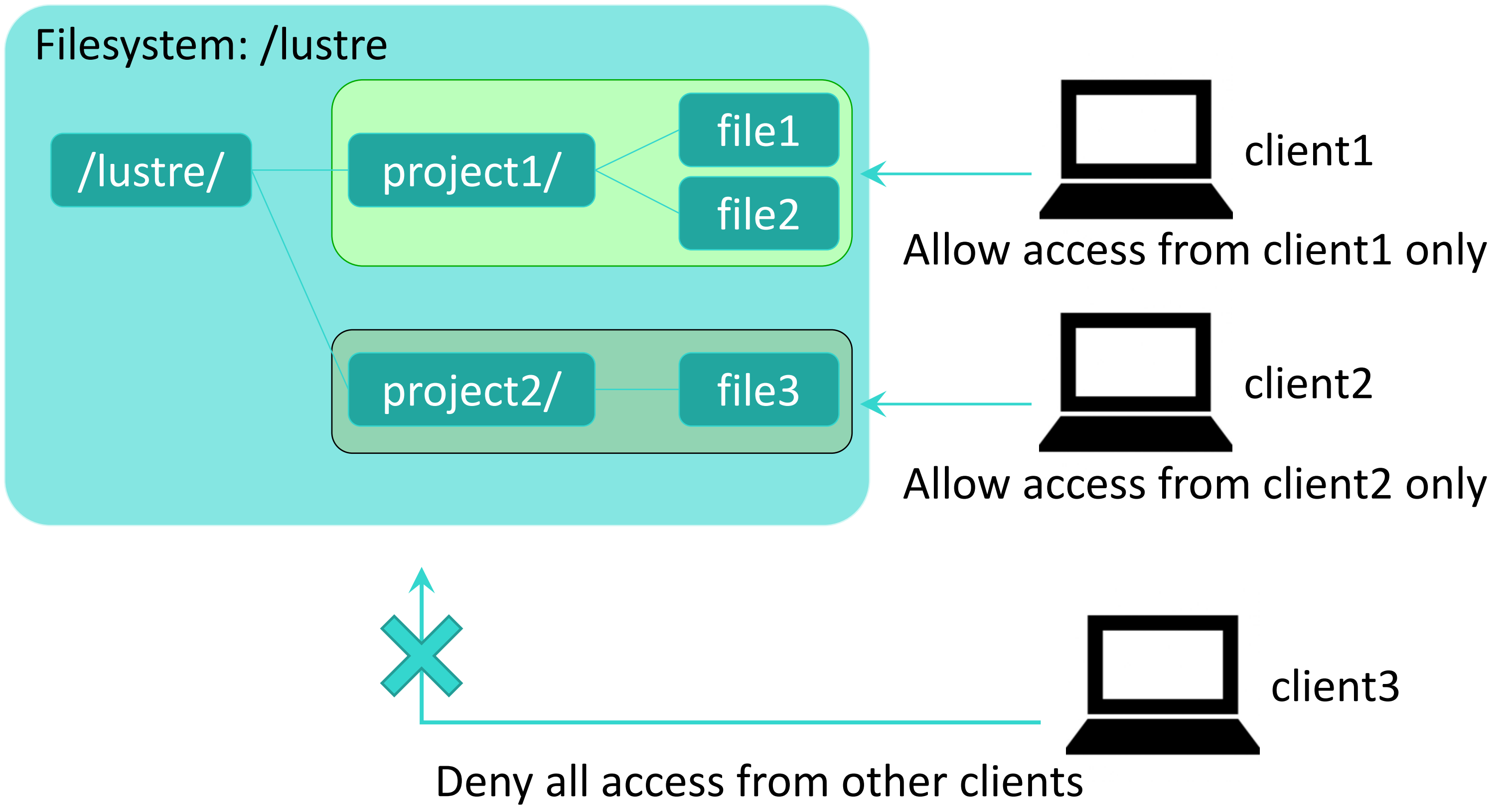
What We Have with EXAScaler

- User/node authentication
  - Kerberos authentication
  - Shared-Secret Key (SSK) authentication
- Access control
  - Discretionary Access Control
  - Targeted & MLS policies on client side
  - SELinux status checking
- Multi-tenancy
  - Lustre client in container or VM, subdir mount
- Encryption
  - On the wire with Kerberos
  - On the wire with SSK
  - Directly at the Lustre client level
- Audit
  - Changelogs-based Lustre Audit with specific Changelogs consumer

Multi-Tenancy

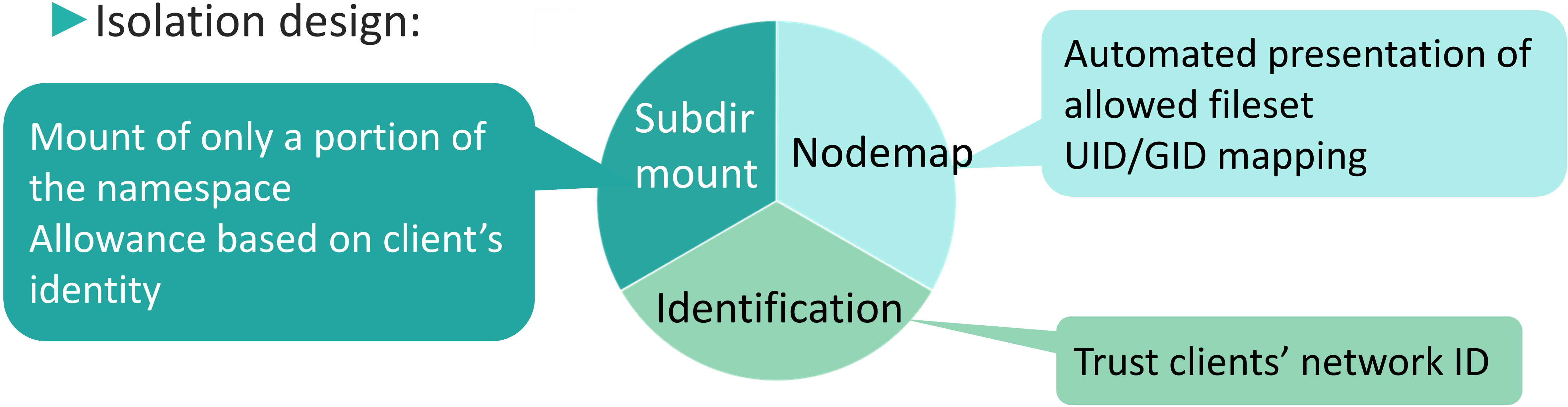
- Rough Idea and Concept
- Implementation
  - Method A
  - Method B
  - Method C
- Performance

Multi-tenancy: Rough Idea



Multi-Tenancy: Concept

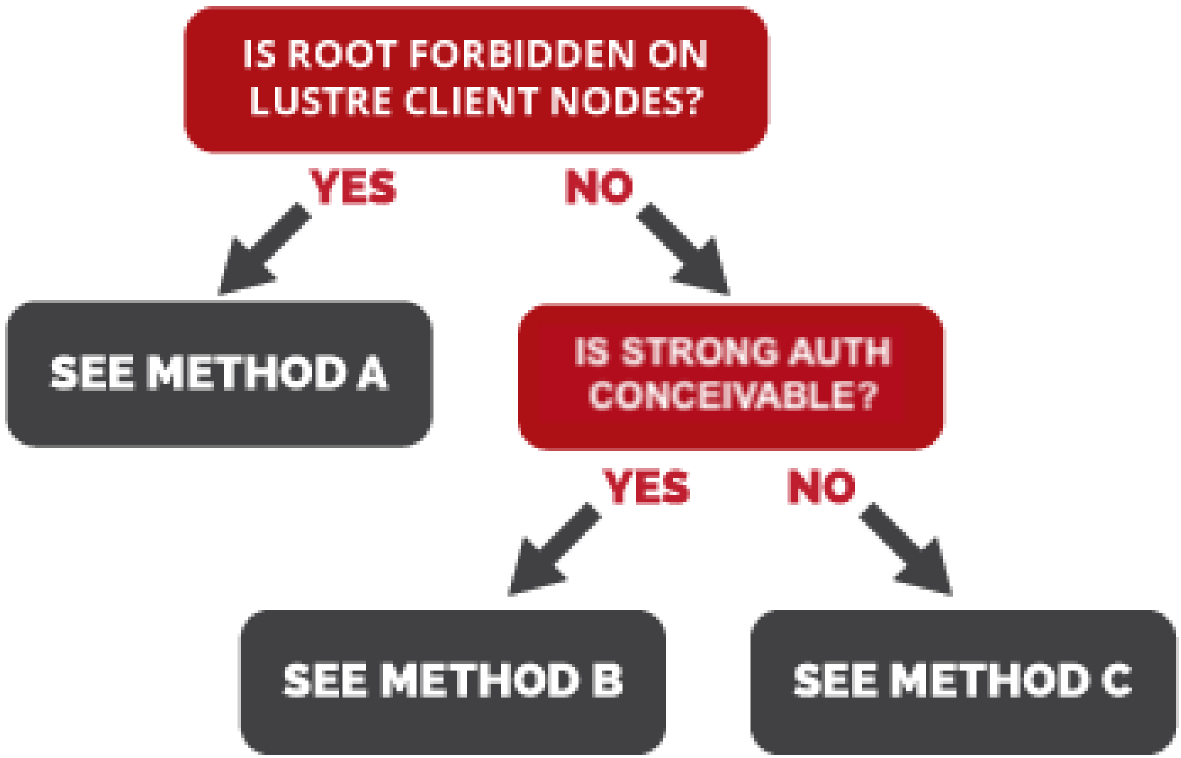
- Isolation design:



- Isolation enables Multi-tenancy:
  - different populations of users on the same file systems
  - isolation of these different populations of users
- Available from Lustre 2.10 / EXAScaler 4

Multi-tenancy: How to Implement

- Narrows down to
  - ability to properly identify the client nodes used by a tenant
  - trust those identities



Multi-tenancy: Method A

Whamcloud

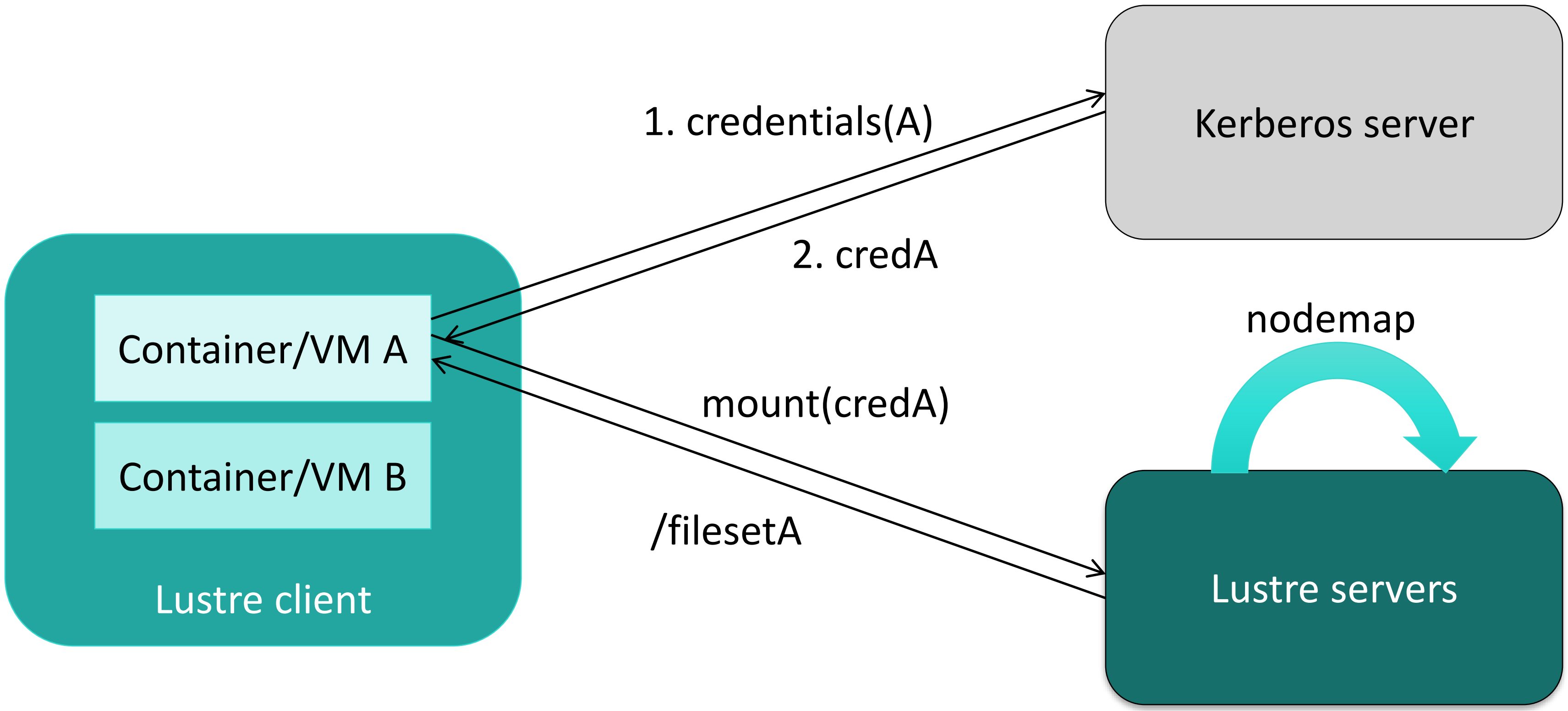
- Users cannot be root
  - clients's NIDs can be trusted.
  - multi-tenancy guaranteed by subdirectory mount and nodemap
  - groups of clients assigned to each tenant can change over time
    - needs to update tenants definitions in nodemaps.

Multi-tenancy: Method B

▶ If Root is Possible on Clients

- are Lustre clients running inside VMs or containers?
  - advantage: dynamically assign NIDs to clients used by tenants
  - drawback: malicious user may use root privileges to change Lustre client NIDs
- make use of strong authentication
  - Kerberos - if already in place at customer site
  - Shared-Secret Key is Lustre-specific alternative, much easier to implement
- how does it work?
  - maliciously modified client NID will not match client’s key
    - installed in VM or container by sec admin
  - Lustre servers will refuse connection

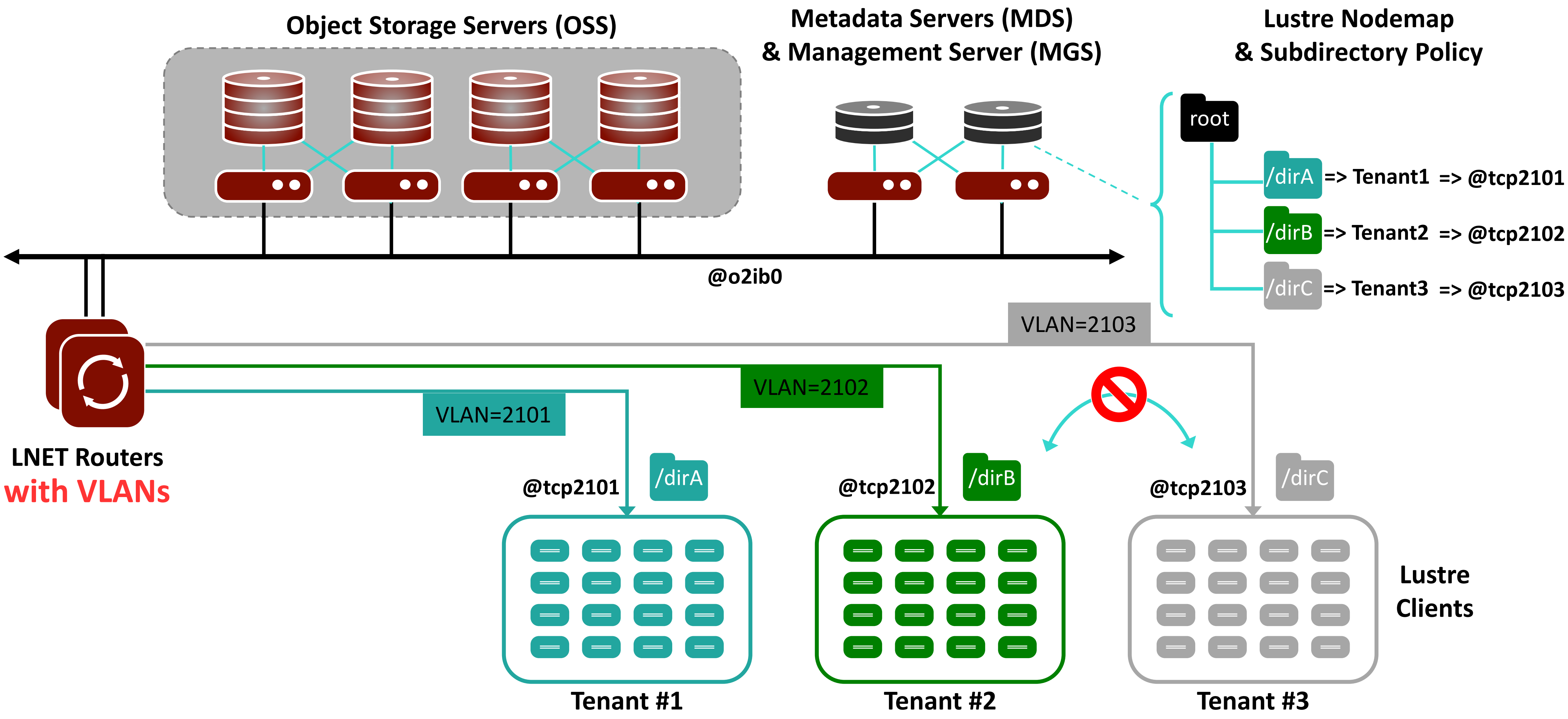
Multi-tenancy: Method B



Multi-tenancy: Method C

- When strong authentication is not an option…
  - not implemented on-site for user authentication
    - too difficult to start using Kerberos authentication with Lustre
  - not adapted to application workflows
    - too complex to deploy credentials for VMs or Containers
- Make use of Lustre routers
  - on the path between Lustre clients and servers
  - inaccessible to users
  - one LNet network per tenant
  - VLANs or IB partitions if shared client network

Multi-tenancy: Method C – as implemented at Uppsala U.



Multi-tenancy: Method C – as implemented at Uppsala U.

- Idea to achieve multi-tenancy: LNet routers
  - 1 tenant == 1 LNet network
    - 1 LNet == 1 nodemap entry
    - 1 LNet == 1 routing rule to reach servers from clients
- But users can be root inside VMs or containers
  - to prevent tenant impersonation (”NID spoofing”):
    - tenant A == VLAN A on client’s host
    - router A == Tag A on network interface

Multi-tenancy: Performance Impact

- No performance penalty incurred by isolation itself
  - tenancy arbitration done at client mount time
  - and for every metadata access if UID/GID mapping is in use
    - but no impact thanks to nodemap caching on server side
- Performance penalty may come from method used to trust clients NIDs
  - Kerberos
  - Shared-Secret Key
  - LNet routers

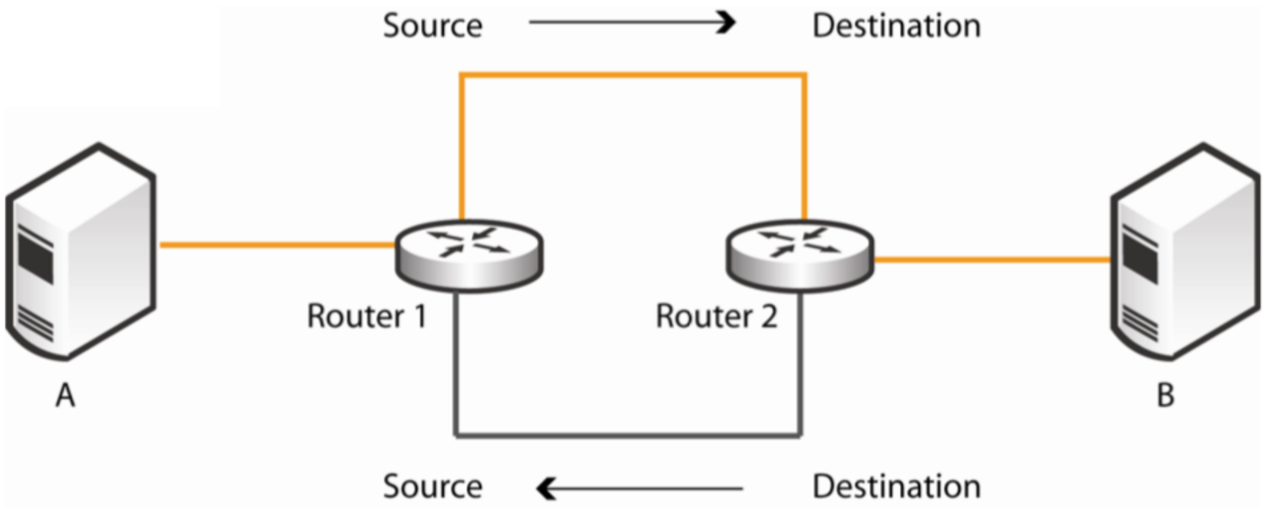
Multi-tenancy

Whamcloud

- Taking Lustre Isolation a step further
  - ability to isolate users from the same population
    - prevent users from accessing others’ data
    - flexibly adjust access capabilities
    - but still share the same file system root
- Add SELinux MLS to enforce data confidentiality

Multi-tenancy: asymmetrical route detection

- Asymmetrical route



https://support.riverbed.com/bin/support/download?did=84

- could be the clue of hostile clients injecting data to the servers

- Purpose is to drop asymmetrical route messages

- Available with Lustre 2.12.1 / EXAScaler 5

Encryption

- On the wire
- Data at REST

Encryption – On the Wire

- Objective
  - protect data transfers between nodes
    - ‘Man-in-the-middle’ attacks
- Encryption over the network with Kerberos krb5p or SSK skpi flavors
  - for communications between Lustre clients and servers
  - data encrypted on emitter’s side before sending
  - data decrypted on recipient’s side upon receival
- Performance impact
  - example with Kerberos krb5p:
    - bandwidth: 80% loss
    - metadata: 40% loss
- Available from EXAScaler 3 (Krb) / EXAScaler 4 (SSK)

Encryption – Data at REST

- **Encryption at disk level with secured disks**
  - protect against storage theft
  - encryption key is managed thanks to Cryptsoft solutions
- **Encryption on top of Lustre: gocryptfs**
  - provide privacy at user level
  - encryption/decryption happens on client nodes
  - bandwidth: 70-80% loss
  - Available from EXAScaler 4
- **Encryption at Lustre client level – based on fscrypt API**
  - protect against storage theft and network snooping
  - data is encrypted before being sent to server and decrypted upon receival from servers
  - bandwidth: ~30% loss in write, ~20% loss in read
  - *Available in 2.14 for content encryption, 2.15/2.16 for name encryption*

Lustre Audit Facility

- Changelog
- laudit

Lustre Audit Facility

Whamcloud

- Objective
  - provide records of all Lustre access

- Use Lustre changelogs
  - log activity on MDTs
  - record file system namespace & metadata events
    - with UID:GID and NID info
  - record *even failed access attempts*
  - limit duplicate `open()` and `close()` events
  - restrict nodes from which activity is recorded

- Available from Lustre 2.11 / EXAScaler 4

Lustre Audit Facility

Whamcloud

- Changelogs space consumption evaluation

	# changelog entries	changelog size	Performance impact
After 10 000 files created	30000	3755824	-15%
After 10 000 files read	50000	6096448	-15%
After 10 000 files removed	60000	7461440	-5%

- Rule of thumb: provision 125 B / entry on MDT

Lustre Audit HOWTO

- All Changelog record types must be enabled, to be able to record events such as OPEN, ATIME, GETXATTR and DENIED OPEN

- Enable all changelog entry types:

```
# lctl set_param mdd.<fsname>-*.changelog_mask=ALL
```

- Then, just register a Changelogs user:

```
# lctl --device <fsname>-<MDT number> changelog_register
```

- Control which Lustre client nodes can trigger the recording of file system access events to the Changelogs

```
# lctl nodemap_modify --name <nodmap_name> \
  --property audit_mode --value=<0,1>
```

Lustre Audit Facility

- Objective
  - provide audit facility
  - reserved to privileged user (root)

- Exploit Lustre changelogs
  - create dedicated Changelogs consumer
    - laudit: consume Changelogs and store audit info into local directory (flat files)
      - data organized for easy tracking of UID:GID and FID activities
  - create tool to query audit logs
    - laudit-report: search flat files, query options

- Available from EXAScaler 4

Lustre Audit Facility HOWTO

- From a special Lustre client, process to consume Changelog entries in background

```
# laudit -d laudit.conf
```

  - fetches Changelog entries
    - free space on Lustre metadata target
  - stores relevant audit logs
    - flat files, no complicated database schema

- Query audit logs
  - file history

```
# laudit-report -f /lustre/fileA -a ‘2019.01.01’ -b ‘2019.03.31’ laudit.conf
```

  - user history

```
# laudit-report -u 500:500 -a ‘2019.01.31 08:00' -b ‘2019.01.31 09:00’ laudit.conf
```

Whamcloud

Thank You!

sbuisson@whamcloud.com
emoly@whamcloud.com

DDN STORAGE