Google

Managed Lustre: Subdirectory (Mount)GSSIAM

DI WANG

Agenda

Problem Context

Limitations of Nodemap and traditional GSS security in a cloud environment.

GSSIAM Protocol

Hybrid RPC Model, Authentication Handshake, and Mount Process Flow.

Core Architecture

Dynamic Sandboxing (ProjID Protection) and Asynchronous Policy Refresh (MDT0-Leader Model).

User Interface

Client-side mount options and server-side administrative control.

Cloud Security & Access Governance Model

Zero-Trust Architecture

Identity-Based Access

Secure machine and service identification for cloud-native workloads.

Contextual Evaluation

Dynamic assessment of user, device, and environmental context for every request.

Advanced Access Governance

- IAM Conditions: Granular, attribute-based control for file system access.
- Resource-Based Policies: Permission evaluation at the directory level.

SAIF Framework

Defense-in-Depth

Why other GSS Security Flavors not fit

GSSIAM Context

GSSIAM bridges modern centralized IAM (OAuth, Cloud IAM) with high-performance parallel file systems.

The Performance Gap: Why other flavors fail

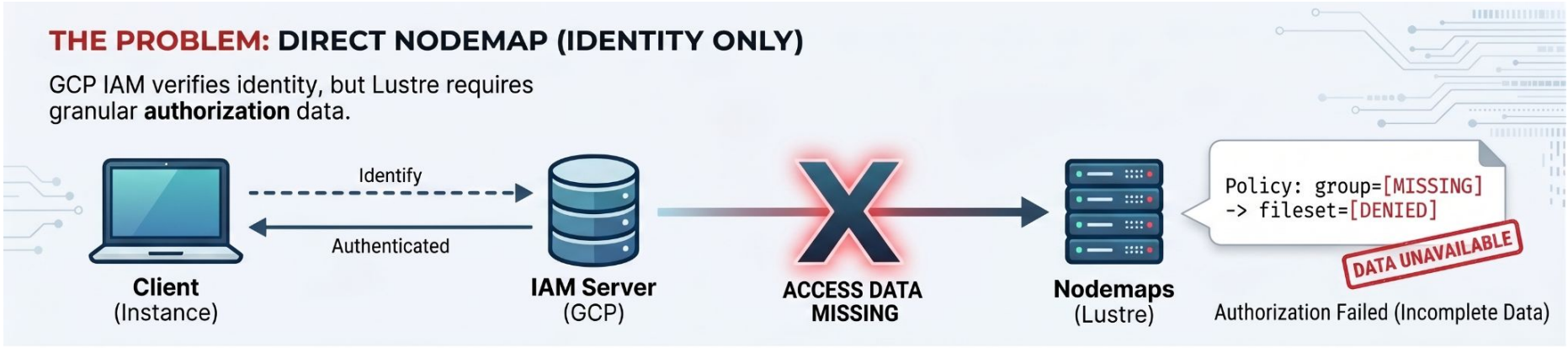
Traditional GSS (Kerberos)

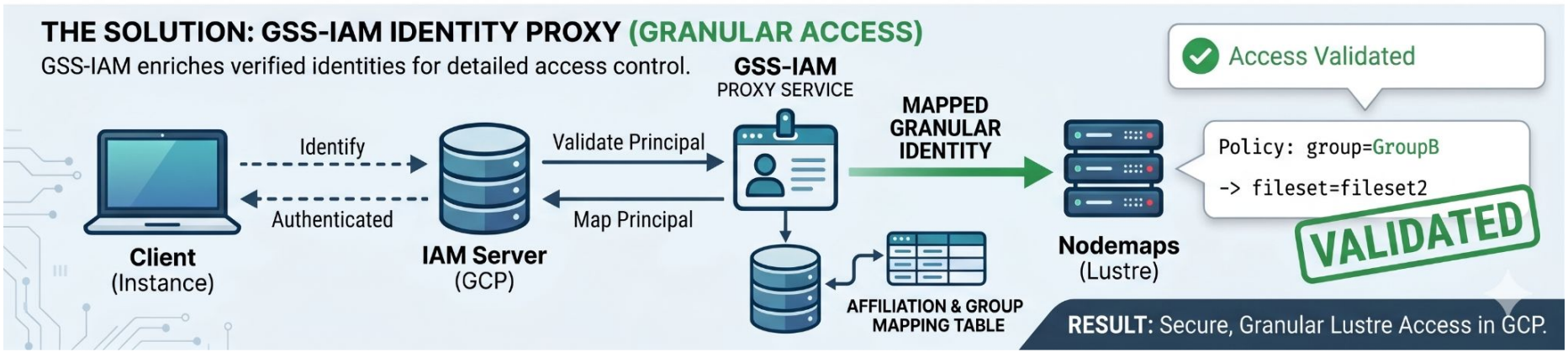
High overhead from per-packet encryption, which significantly impacts both metadata and bulk data transfer performance.

Standard GSS (Null/Plain)

Offers bare-metal performance but lacks any authentication, making it unusable for secure environments.

Why Nodemap Cannot Be Used





What is GSSIAM

GSSIAM (GSS Identity and Access Management)

A specialized security flavor in Lustre designed for high-performance, identity-based authentication and fine-grained access control.

Hybrid RPC Model

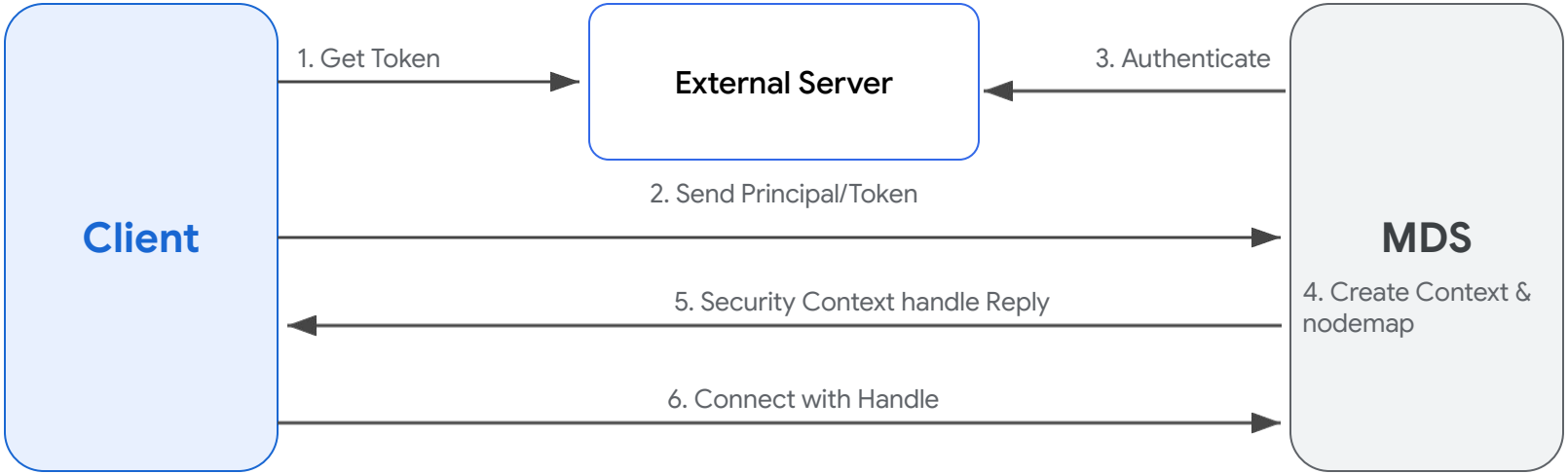
- **Authentication Phase:** Uses standard GSS (Generic Security Services) handshakes to verify a client's identity (IAM token). This occurs once during connection/context establishment.
- **Data Phase:** Once authenticated, subsequent RPCs use Null framing (no signing/encryption per RPC). This avoids the significant CPU overhead of standard GSS flavors (like krb5i or krb5p) while maintaining security by "pinning" the verified identity to the connection.

GSSIAM Handshake Process

1. Client Initialization
Client obtains token and initiates mount by principal identity.

2 & 3. Server Authentication
Principal and token sent to GSSIAM external server for validation.

4, 5 & 6. Context & Reply
MDS creates context, returns reply, and client connects with handle.



Mount Process Flow

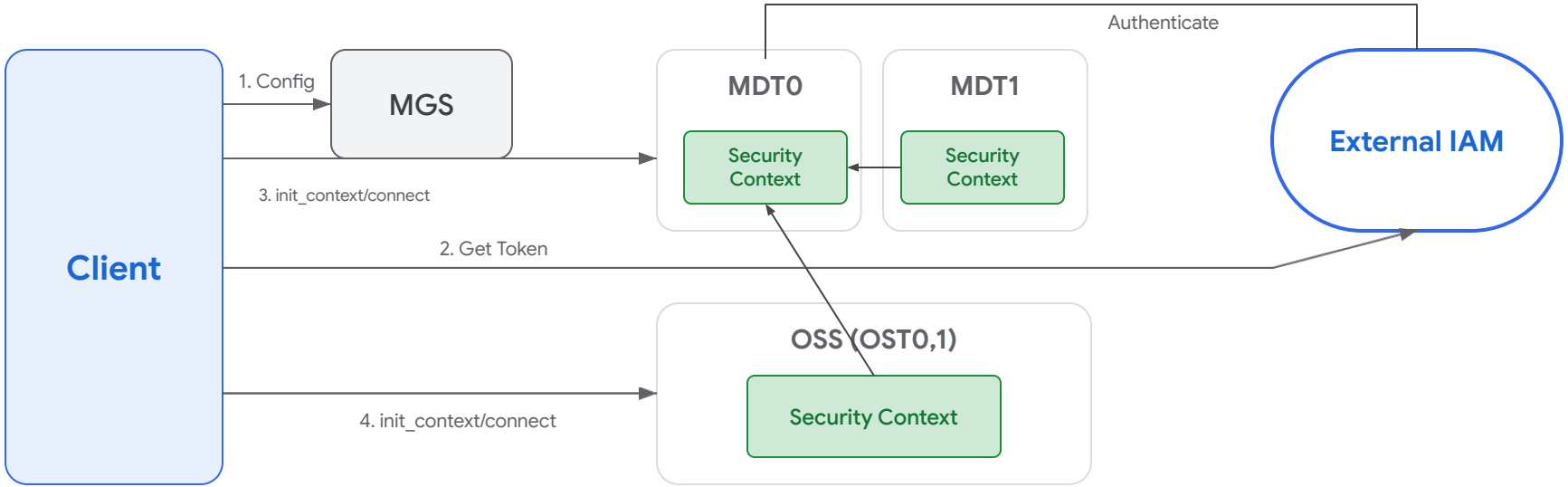
1. Configuration
Config Log security flavor=gssiam

2. IAM Auth
Get Token from External IAM

3. Connect MDT0
init_context/connect with token/principal

4. Process Flow
Authenticate and context establishment

5. Other targets
Connect other MDTs and OSTs



The "Invisible Fence" for multiple Tenancy

The Feature

GSSIAM locks an authenticated user to a specific Lustre Project ID at the session level.

Zero-Trust Sandboxing

The server ignores client-side claims. It forces a rule: you only touch files matching your assigned Project ID.

Solves "Local Root" Risks

Perfect for K8s and Docker. Local root power cannot bypass this server-side security fence.

```
// Server-Side Enforcement
if (user_projid != file_projid) {
   return -ECHRNG; // Access Denied
}
```

Asynchronous Policy Refresh

1. Centralized Heartbeat

MDT0 acts as the single "Leader" for IAM authentication to minimize external load.

2. Policy Propagation

Other servers pull the policy from MDT0 via internal protocols.

3. Efficient Enforcement

Lazy consistency ensures cluster-wide security without the overhead of per-RPC checks.

💡 Architectural Highlight: MDT0-Leader Model

- **Leader-Based Refresh:** Only MDT0 performs background heartbeats to the IAM server.
- **Automatic pull:**All other servers pull the security context from MDT0.

Summary: Centralized authentication maintains high performance and tight IAM integration.

GSSIAM User Interface

Client-Side: Simple & Transparent

For the end-user, GSSIAM is designed to be almost invisible. The only difference is how they provide their identity during a mount.

- **Mount Command:** Users provide their IAM token via the `user_principal` mount option.
- Set upcall binary: Set upcall binary to get the token from external GSSIAM server

```
lctl set_param sptlrpc.gssiam.gssiam_upcall=your_external_binary
mount -t lustre -o user_principal=<user_principal> ...
```

GSSIAM Server Interface

Server-Side: Powerful Management

Administrators use standard `lctl` commands to configure and monitor the GSSIAM environment.

Configuration

- **Enable GSSIAM Flavor:** Set the security policy for the filesystem or specific targets.
- **Set authorize binary:** Set authorize binary in user space to authenticate the token.
- **Enable nodemap:** Enable nodemap for creating dynamic nodemap for each user principal and subdirectory.

```
lctl conf_param <fsname>.<target>.<parameter>=value
lctl set_param sptlrpc.gssiam.gssiam_auth=/usr/sbin/l_gssiam_auth
```

Community Work

LU-19921: Lustre GSS IAM Integration

This initiative focuses on integrating Identity and Access Management (IAM) with Lustre's Generic Security Service (GSS).

- All relevant patches and progress are documented in the ticket.
- Projected for inclusion in a future 2.18 version.

Reference Link:
https://wiki.whamcloud.com/display/PUB/Lustre+mount+IAM+integration+HLD

Thank You!

Questions & Discussion