Virtualization Aware Access Control for Multitenant Filesystems

Giorgos Kappes, Andromachi Hatzieleftheriou,
Stergios V. Anastasiadis

Department of Computer Science and Engineering
University of Ioannina, Greece

IEEE 30th International Conference on Massive Storage Systems and Technology (MSST 2014)

Storage Consolidation

Benefits

- Efficient usage of storage resources
- Sharing support
- Increased manageability
- Reduced cost

Block-level storage

- Direct virtual disk access through block interface

File-level storage

- Direct filesystem sharing through file interface

Storage Multitenancy

Goal

- Storage infrastructure shared among different tenants

Requirements

- **Scalability:** Support enormous number of end users
- **Isolation:** Isolate the user identities and access control of different tenants
- **Sharing:** Flexible data sharing within or between tenants
- **Compatibility:** Compatibility with existing applications
- **Manageability:** Flexible resource management

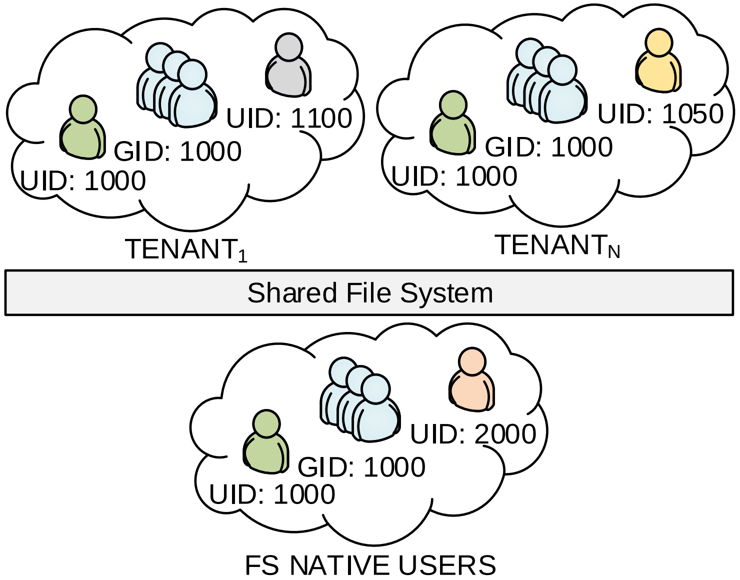
Research focus

- Efficient and secure multitenancy in VM filesystems

Motivation

Problem of multitenancy

- Shared FS namespace
- Crosstalk between tenants
- Complicated security



Native multitenancy at the filesystem level

- Clean way to isolate multiple tenants
- Shared hardware, operating system, fileservers
- Configurable isolation, sharing, performance, manageability

Prior approaches

Centralized

- The principals' identities of all tenants centrally maintained
  – Poor scalability, isolation and manageability

Peer-to-peer

- The principals of each tenant managed locally
- Tenants communicate to publicize their principals' identities
  – Overhead to periodically synchronize the tenants

Mapping

- Local principal IDs mapped to global unique IDs
  – Mapping overhead, sharing complications, security violations

The Dike Approach

Hierarchical identification and authentication

- The tenants manage their principals
- The provider manages the tenants

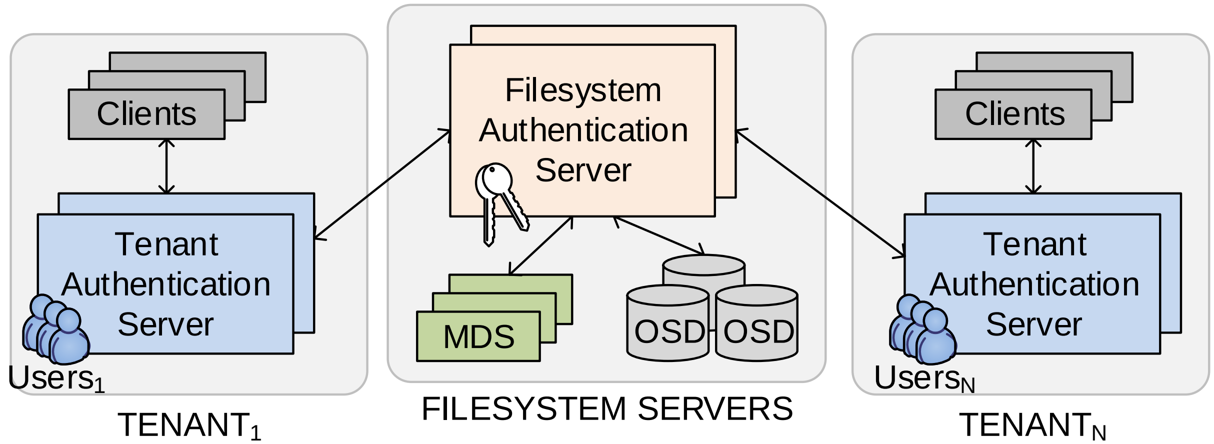
Native multitenant authorization

- Separate ACLs per tenant and provider
- Namespace isolation through filesystem views

Efficient permission management and storage

- Shared common permissions
- Inheritance of permissions

Identification



Principals

- Tenant principals: Use/manage tenant resources
- Native FS principals: Manage the FS

Tenant Authentication Server (TAS)

- Certifies local clients and principals

Filesystem Authentication Server (FAS)

- Certifies filesystem services, tenants, native principals

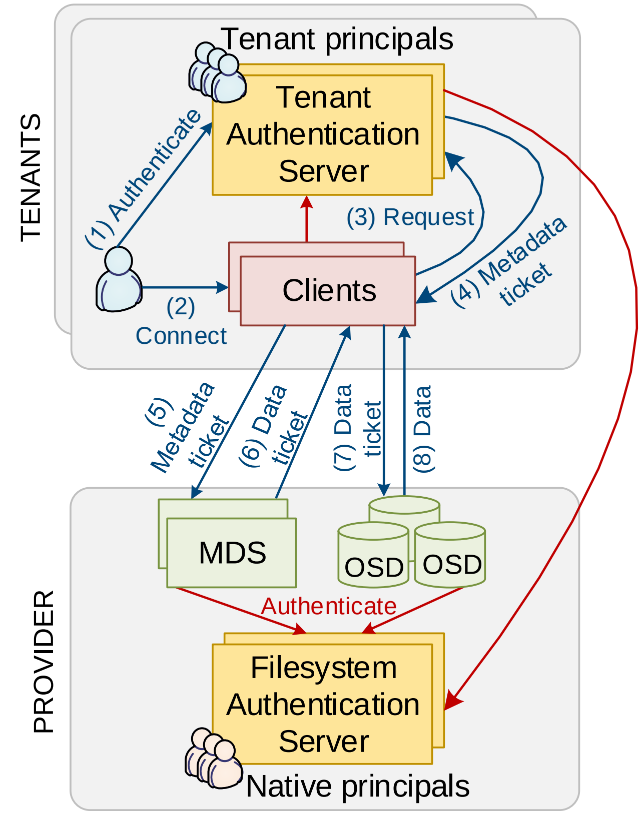
Authentication

Metadata ticket

- Securely specifies tenant principal
- Provides access to MDS

Data ticket

- Securely specifies tenant principal and permissions
- Provides access to OSDs

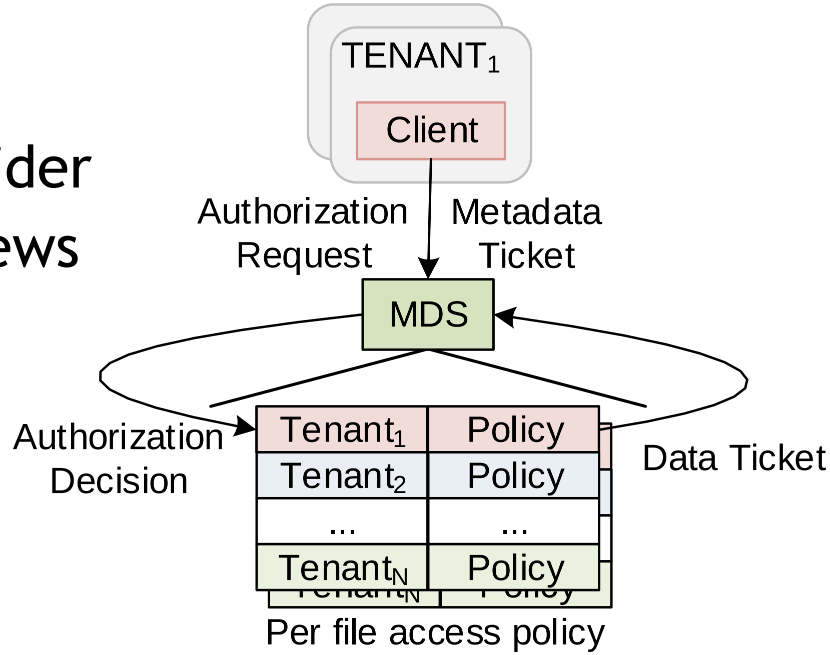


Steps		
	(1) Principal authenticated by TAS (2) Principal requests FS access (3) Client contacts TAS (4) Client receives Metadata ticket	(5) Client contacts MDS (6) MDS issues Data ticket (7) Client contacts OSD (8) Client accesses data

Authorization

Access control isolation

- Separate ACLs per tenant, provider
- Metadata accessible through views



Filesystem view

- Used by native principals to manage tenants

Tenant view

- Used by tenants to access or manage tenant resources

File sharing

- Private to a principal
- Shared across principals of one or more tenants

Common Permissions

Goal

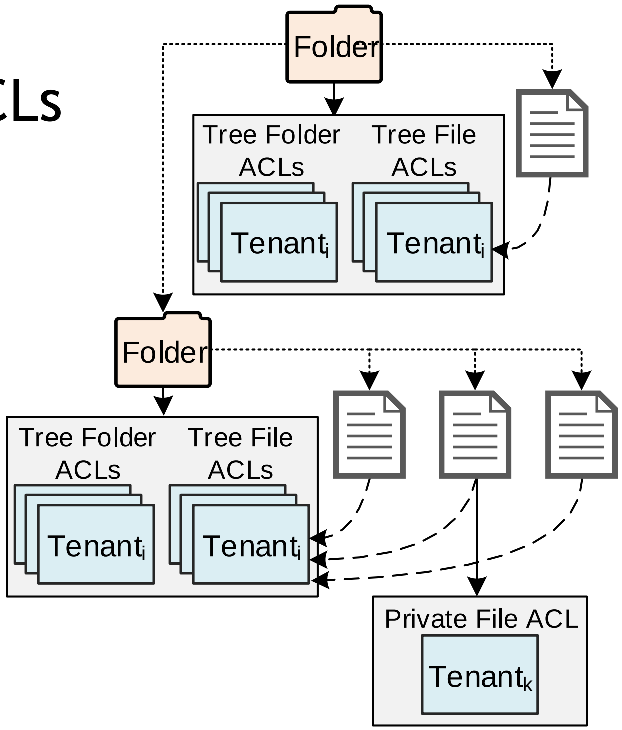
- Reduce filesystem load by reducing ACLs

Per tenant permission inheritance

- Permissions can be inherited to child files/folders

Per tenant common permissions

- Child files can share parent's ACL



ACLs

- **Tree folder ACLs:** Folder permissions
- **Tree file ACLs:** Shared child files permissions
- **Private file ACLs:** Child file permissions explicitly set by user

Security Analysis

Captured credential

- Fresh tamperproof credentials cannot be forged

Compromised tenant principal account

- Compromised tenant view is isolated
- Attack limited to principal's private or shared files
- Cross-tenant policy violation is prohibited

Attack by revoked tenant

- Restricted through deleted tenant view
- Tenant cannot access other views

Compromised provider administrator account

- Handle via good practices (e.g., restricted remote access)

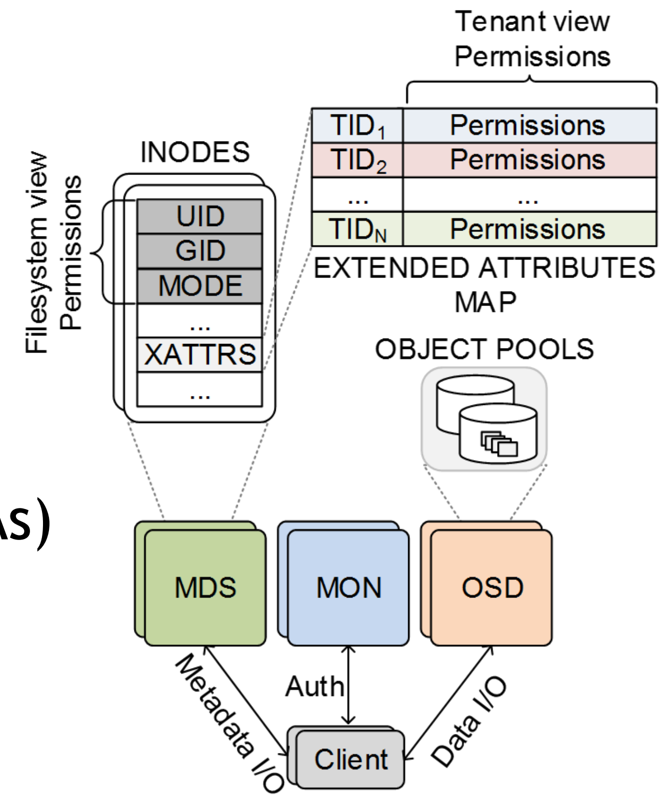
Prototype

Session

- Tenant identified by client
- Session limited to one tenant

Permission storage

- **Tenant view:** Extended Attributes (EAs)
- **Filesystem view:** Regular fields
- EAs with tenant permissions not directly accessed by clients



Implemented on CephFS

Capabilities

- Include principal and tenant identifiers
- Sent to clients with tenant file access

Experimentation Environment

Configuration: AWS EC2 Instances

- **m1.xlarge:** x3, 4 VCPU, 15 GB RAM, Linux 3.9.3
- **t1.micro:** x32, 1 VCPU, 615 MB RAM, Linux 3.9.3

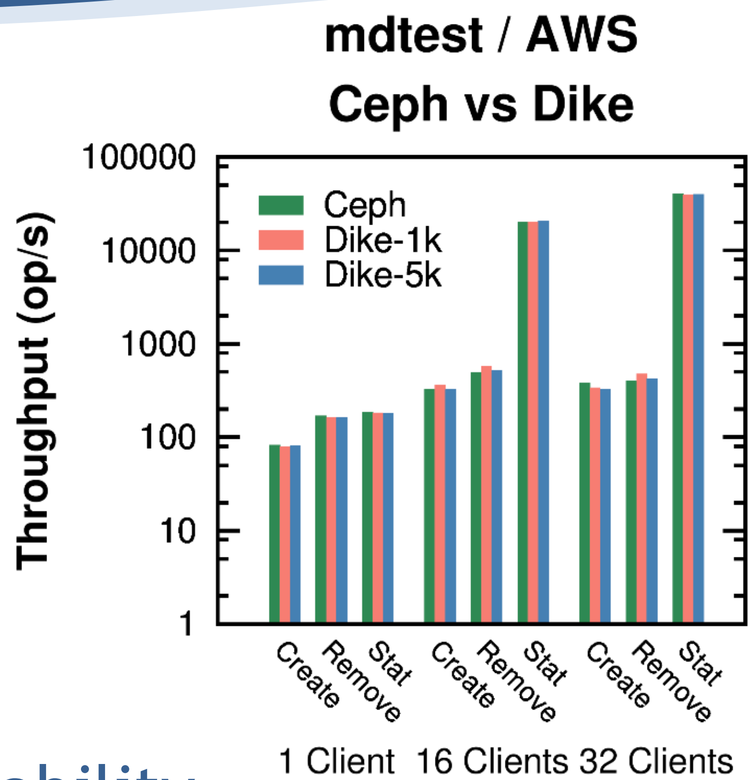
Filesystem configuration

- **Ceph/Dike:** m1.xlarge, 1xOSD+MON, 1xOSD+MDS, 1xOSD
- **Gluster/Heka:** m1.xlarge, 3 fileservers
- Replication factor 3

Microbenchmark

- mdtest
- 48000 created files and folders

Results



Dike Client scalability

- 1 → 32 clients: Similar to Ceph

Dike Tenant scalability

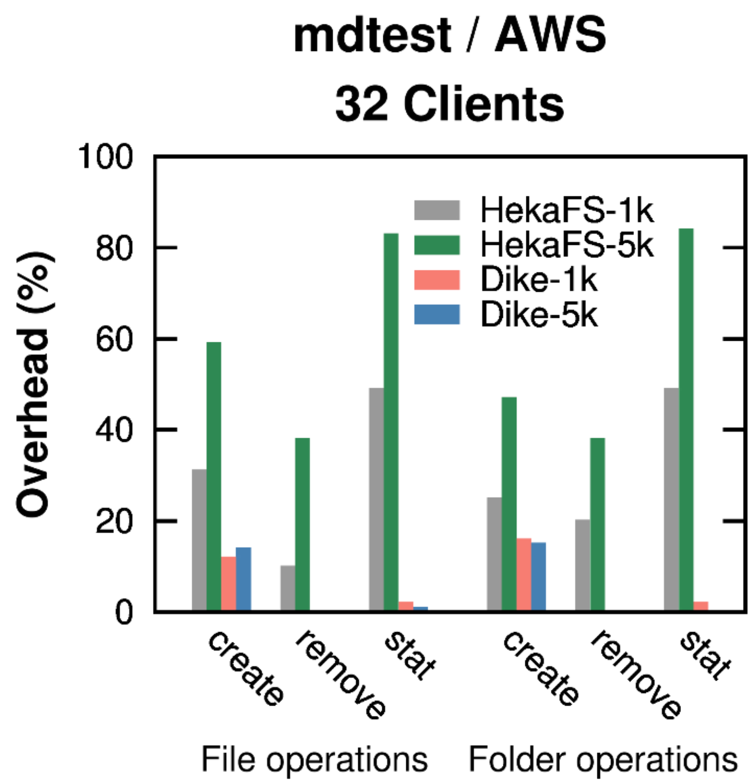
- 1k → 5k tenants: 2% extra overhead

Results

Dike native multitenancy

Limited overhead

Scalable to thousands tenants



Dike limited overhead

- 1k tenants overhead: up to 14%
- 5k tenants overhead: up to 16%

ID mapping multitenancy too costly

- 1k tenants overhead: up to 49%
- 5k tenants overhead: up to 84%

Conclusions

Native filesystem multitenancy with sharing support

- Hierarchical identification scheme
- Namespace isolation: Per tenant and provider ACLs
- Per tenant common permissions and inheritance

Performance and security analysis

- Limited multitenancy overhead up to 16%
- Dike scalable to several thousand tenants
- Tenant principals not able to violate cross-tenant policy

Future work

- I/O intensive application experimentation
- Weaker trust assumptions

Backup

Comparison of Interfaces

Benefits	Block Interface	File Interface
Compatibility	✓	
Isolation	✓	✓
Sharing		✓
Consistency, Performance		✓
Disaster recovery, migration	✓	✓
Thin Provisioning	✓	✓
Searchability		✓
Snapshoting, Versioning	✓	✓

Dike compared to Ceph

Pool-level multitenancy

- Objects organized in per tenant pools
- No support for sharing files among tenants
- **In Dike tenants can securely share the same pool**

Centralized Identity management

- Keystone integration
- Poor scalability, isolation and manageability
- **Dike: Hierarchical identity management scheme**

ACLs

- Earlier versions of Ceph support Posix ACLs
- Single ACL for all tenants leads to poor isolation
- **Dike: Separate ACLs per tenant and provider**