QuickSilver: A Distributed Policy Engine for Lustre  (An Update)

Chris Brumgard
brumgardcd@ornl.gov
LUG 2023
Problem (As a reminder)

• At ORNL, filesystems are becoming increasingly complex to accommodate the needs for faster storage as well as larger storage but at the same time $$$ matters.
  – Tiers
    • Users are terrible at managing the spatial and temporal placement of their data.
  – Unified namespace
    • Tricks them into thinking that their storage is equally accessible.
      – They will forget to purge and migrate.

• Beyond just tiering, admins want an easy and reliable way to implement different policies for different users/groups.
  – Telemetry and querying.
Enter PoliMOR

• A Distributed, Extensible, and Automated Policy Engine for Lustre.

• Purposes:
  – Migration, Purging, Data collection, and telemetry

• What do we mean by Distributed?
  – Agent-Oriented Microservices
  – Distributed Messaging queue
  – Fault Tolerance & Scalable
Enter PoliMOR

• A Distributed, Extensible, and Automated Policy Engine for Lustre.

• What do we mean by Extensible?
  – New agents can be added for functionality.

• What do we mean by Automated?
  – No intervention by the users or admins.
  – Define a set of invariants within a policy to be maintained.
PoliMOR diagram
Performance Test Setup

• Testbed
  – 11 nodes AMD EPYC 7351 16-core processor, 126 GB of RAM.
  – 6 OSSes and 4 MDS.
  – EDR Infiniband.

• ~10 million files spread across 100 project directories.
  – 80% on the capacity tier.
  – 20% on the performance tier.
  – Random timestamping for migration and purging.
  – File size used bimodal distribution found on production file system.

• 3 Scan agents, 2 policy agents, 3 purge agents, and 3 migration agents.
Scan Agent Characteristics

![Graph showing Scan Agent Characteristics](image)

- **Graph 1**: 
  - X-axis: Hours
  - Y-axis: # Files (millions)
  - Lines represent:
    - Host
    - rage7
    - rage1
    - rage4

- **Graph 2**: 
  - X-axis: Hours
  - Y-axis: %CPU
  - Legend:
    - Ifs
    - scan_ag+
Policy Agent Characteristics

![Diagram showing CPU usage over time]

- %CPU
- Hours
- COMMAND
- policy_+
- Open slide master to edit
Purge Agent Characteristics

- **# Files vs Hours**
  - Legend:
    - host
    - rage8
    - rage9
    - rage10

- **%CPU vs Hours**
  - COMMAND
    - purge_a+
Migration Agent Characteristics

- Host usage for different migration agents (rAge8, rAge9, rAge10)
- File count and gigabytes transferred over time
- CPU usage over time for different processes (lfs, migrati+)

Open slide master to edit
Impact on MDS and OSS

- Load vs. Hours for hosts: nebul27-crius-mdt2, nebul28-crius-mdt3, nebul25-crius-mdt0, nebul26-crius-mdt1

- Load vs. Hours for hosts: cirus-oss6, cirus-oss5, cirus-oss4, cirus-oss1, cirus-oss2, cirus-oss3
Scalability Results

• Used 8 nodes of the testbed.
• Deployed 1 to 64 agents across the nodes.
• Tested each agent type in isolation.
Scalability Results

- **Number of files scanned**
  - Direct 'lfs find'
  - Scan agent

- **Time (sec) taken to process 1 Million files**

- **Time (sec) taken to purge 3000 files**

- **Time (sec) taken to migrate 3000 files**
Future work

• Still in active development.
  – Reducing the scan work.
  – Productionizing.
  – Currently deploying to Orion.

• More complex policies.
  – Decomposing complex actions into simpler rules.

• Non-Lustre agents.
  – HPSS
  – Edge
Acknowledgments

• Thanks to my fellow developers Anjus George, Ketan Maheshwari, Rick Mohr, James Simmons.

• This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.