Network File System Version 4 C. Lever
Internet-Draft Oracle
Intended status: Informational February 3, 2020
Expires: August 6, 2020

Network File System Version 4 Requirements for Computational Storage
draft-cel-nfsv4-comp-stor-reqs-02

Abstract

This document proposes an architecture to support Computational Storage using Network File System version 4 (NFS) files.

Table of Contents

1. Introduction 2
2. Computational Storage in Operation 3
2.1. Service Discovery 3
2.2. Service Configuration 3
2.3. Service Operation 4
3. Security Considerations 4
4. IANA Considerations 5
5. References 5
5.1. Normative References 5
5.2. Informative References 5
Acknowledgments 6
Author's Address 6

1. Introduction

In traditional computing architectures, stored data is dormant. Computational storage brings computing power closer to data storage to leverage a high-bandwidth link between the compute resource and data-at-rest, or to reduce interrupt or data bandwidth needed between storage and host. Reducing the movement of large data objects lowers power consumption and increases opportunities for parallelism.

There are already several pervasive usage scenarios suited to computation offloaded to storage:

Search: Examples include SQL offload, a machine learning inference engine co-located with its dataset, or performing a "find" operation without pulling an entire filesystem's data to a client.

Data Transformation: Examples include compression, transcoding, and encryption.

Data Management: This might be a control plane that permits administrative actions such as instantiating a transfer to cold storage, integrity measurement (scrubbing), or creating a snapshot of a particular file.

In some cases, computational storage is a computational service that is available as a direct offload for a host CPU. The source and sink data both reside in the host's memory. For NFS, however, the mission of computational storage techniques is to reduce network utilization between an NFS server and its clients. Here, the source and sink are files on NFS servers. The operation of the computational service can be entirely invisible to applications running on NFS clients.

NFSv4.2 [RFC7862] already applies this approach -- features new to NFSv4.2 include copy offload and file initialization (ALLOCATE), both of which are intended to prevent extra data round-trips between clients and server.

Computational storage is an emerging technology already offered by several companies, including Samsung and HPE. A suitable introduction appears in [TORA]. The purpose of the current document is to provide a framework for discussing and reasoning about computational storage relative to the NFS protocol and typical NFS deployments.

2. Computational Storage in Operation

For various reasons, we do not want to require changes to the NFS protocol to expose computational storage resources. Instead, an NFS server host can advertise RPC programs that allow NFS clients to recognize and configure the NFS server's computational services. The services operate on data stored on that server.

We begin by defining the term Computational Storage Service (CSS) to mean a network service that performs computation on data where the service and the data it operates upon are tightly associated with a storage target.

2.1. Service Discovery

Typically a CSS configuration facility registers with the NFS server's rpcbind service [RFC1833] to advertise its listening port and RPC program number. Administrative clients or users then contact this service to configure it for use.

A CSS that has no administrative interface must also advertise its presence on the NFS server via this mechanism.

2.2. Service Configuration

Computational Storage Services have varying degrees of configurability. A so-called Fixed Computational Storage Service provides one or a few specific pre-determined functions (e.g., encryption).

A Programmable Computational Storage Service is a more general-purpose service that must be provided with a program before the CSS becomes usable (e.g., an operating system image or an FPGA bit file).

A configuration program exposes the parameters of a specific CSS via RPC. Such configuration might include the selection of encryption

algorithms or keys, or the specification of regular expressions or prepared SQL statements. The input dataset or a destination for results might also be specified.

The primary class of input and output parameters for configuration programs are objects (e.g., files and directories) that exist in a filesystem shared via NFS. When they are local, a CSS can reference such objects by filehandle and optionally a range of bytes. A CSS references a remote object using either an NFS URI (defined in Section 2.8.1 of [RFC7532]) or a tuple consisting of a network address and a filehandle.

2.3. Service Operation

There are two alternative modes of operation:

Transparent: Once configured, a CSS's operation occurs behind NFS READ and WRITE operations, and is not directly visible to NFS clients. For instance, an NFS server might perform data reduction (e.g., deduplication) or encryption-at-rest without exposing these transformations to clients.

Verbal: Clients use a separate RPC protocol to initiate requests or capture results when the results are expected to be small or are not appropriate for storing into a file. This mode of operation is useful for invoking search operations over large datasets where the results might be a small set of filehandles with byte ranges.

Serialization might be necessary to prevent an offload agent from colliding with accesses by standard NFS clients. A client might open the input file or hold a delegation for this purpose.

Alternatively, the NFS protocol might provide no serialization. Applications themselves would be responsible for maintaining the integrity of the input datasets during offloaded operations.

3. Security Considerations

NFS storage is typically deployed on open networks rather than in environments with restricted access, such as a PCIe bus or a dedicated storage fabric. In such open environments, administrators must focus extra attention on security. In particular:

- Remote access to configuration and computational results must be authenticated and authorized. The ONC RPC protocol itself [RFC5531] has such authentication mechanisms, including mechanisms that use cryptography [RFC7861].

- There must be a mechanism for authorizing offload agents to access file data on behalf of authenticated users.

- A trust relationship must exist between clients and servers. For example, how would clients be certain that the server has actually encrypted a file's content?

- NFS servers must schedule the use of Computational Storage Services fairly to prevent denial-of-service.

4. IANA Considerations

This document has no IANA actions.

5. References

5.1. Normative References

[RFC1833] Srinivasan, R., "Binding Protocols for ONC RPC Version 2", RFC 1833, DOI 10.17487/RFC1833, August 1995, <https://www.rfc-editor.org/info/rfc1833>.

[RFC5531] Thurlow, R., "RPC: Remote Procedure Call Protocol Specification Version 2", RFC 5531, DOI 10.17487/RFC5531, May 2009, <https://www.rfc-editor.org/info/rfc5531>.

[RFC7532] Lentini, J., Tewari, R., and C. Lever, Ed., "Namespace Database (NSDB) Protocol for Federated File Systems", RFC 7532, DOI 10.17487/RFC7532, March 2015, <https://www.rfc-editor.org/info/rfc7532>.

5.2. Informative References

[RFC7861] Adamson, A. and N. Williams, "Remote Procedure Call (RPC) Security Version 3", RFC 7861, DOI 10.17487/RFC7861, November 2016, <https://www.rfc-editor.org/info/rfc7861>.

[RFC7862] Haynes, T., "Network File System (NFS) Version 4 Minor Version 2 Protocol", RFC 7862, DOI 10.17487/RFC7862, November 2016, <https://www.rfc-editor.org/info/rfc7862>.

[TORA] Torabzadehkashi, M., Rezaei, S., HeydariGorji, A., Bobarshad, H., Alves, V., and N. Bagherzadeh, "Computational storage: an efficient and scalable platform for big data and HPC applications", Journal of Big Data 6, 100, DOI 10.1186/s40537-019-0265-5, November 2019.

Acknowledgments

The author is grateful to Bill Baker, Greg Marsden, and Jim Williams of Oracle, Glenn Watkins of HPE, and Stephen Bates of Eideticom for their input and support of this work.

Special thanks go to Transport Area Director Magnus Westerlund, NFSV4 Working Group Chairs David Noveck, Brian Pawlowski, and Spencer Shepler, and NFSV4 Working Group Secretary Thomas Haynes for their support.

Author's Address

Charles Lever
Oracle Corporation
United States of America

Email: chuck.lever@oracle.com