Appendix: Artifact Description/Artifact Evaluation

Artifact Description (AD)

1 Overview of Contributions and Artifacts

The contributions of this work includes a literature review of scratch and archival storage separation. It provides analysis of anecdotal evidence and perspectives from practitioners, implementers and architects that leads to the proliferation of separate and non-unified scratch and archival storage infrastructures. It then provides experiments and results that outline the reasons the two storage modalities can be combined with declarative methods, to achieve a sustainable outcome, contributing a new view, that runs counter to traditionally held views. The artifacts include computational outputs of IOR runs under different policy conditions, hardware specifications, configuration, compute node information and the code (scripts, binaries) to generate the declarative policy and scheduler integrations.

1.1 Paper's Main Contributions

- C_1 An analysis of why scratch and archival filesystems have traditionally been separated.
- C_2 An experimental demonstration of the baseline performance of a scratch array.
- C_3 An experimental demonstration of the performance impact of naively combining scratch and archival storage.
- C_4 A proposal for a harmonized system.
- C_5 The design of a declarative data management policy to control data placement.
- C_6 A performance evaluation showing that our harmonized system performs on par with a dedicated scratch system.

1.2 Computational Artifacts

There are multiple computational artifacts related to this paper. These are IOR runs conducted under different declarative policy conditions, as follows.

- A_1 https://git.new/6xnAv8r - *Baseline Scratch Performance, No Declarative Policy*
- A_2 https://git.new/3M1SNtB - *Archival-HEAVY Declarative Policy Performance*
- A_3 https://git.new/qmgrBDd - *Archival-MOD Declarative Policy Performance*

Artifact ID	Contributions Supported	Related Paper Elements
A_1	C_2	Table 2 Figure 4
A_2	C_3	Tables 2 and 3 Figure 5
A_3	C_6	Tables 2 and 3 Figure 7

2 Artifact Identification

The following details Artifact Identification for items A_1 - A_3

2.1 Computational Artifact A_1

This artifact is the baseline scratch performance of the disk storage system in the experimental environment. It is an unmodified scratch platform that does not have any HSM, ILM or declarative policy running on it. It is a ground state for the work, with no modifiers or experimental changes applied.

Relation To Contributions

As per C_2, this artifact relates to the contribution as it is required to establish the ground state performance with no impact or potential characteristic changes. It is the "control" of the experiment.

Expected Results

A baseline characterised storage array with IOR results exhibiting a generally faster read IO performance and read throughput performance response compared to writes. There should be no unusual observed behaviour or unique IO patterns beyond a traditional parallel filesystem.

Expected Reproduction Time (in Minutes)

This expected reproduction time will take 15 hours.

- Artifact Setup will take 25 minutes.
- Artifact Execution will take 12 hours.
- Artifact Analysis will take 2.75 hours.

Artifact Setup (incl. Inputs)

Hardware. A disk storage device

- Media: 24 * 7.68TiB NVMe drives for metadata and 1020 * 18TiB NL-SAS drives for data.
- 16 * 200Gbps InfiniBand ports from ESS to data fabric.
- The entire IBM Storage Scale ESS Disk System definition and parameters can be found here: https://git.new/hu67ewK

Access nodes -CES-NFS nodes

- 4 * Dell R6525 nodes
- Network: 1 * 200Gbps HDR InfiniBand port transporting NFS protocol to the GPFS ESS based storage mounts and to compute nodes over RDMA.
- CPU: 64 * AMD EPYC 9124 16-Core Processors - hyperthreading (HT) enabled.
- Memory: 396GB DDR5 memory.

A HSM Data Storage System

- An HPE Zero-Watt Storage Spin Down Disk array consisting of 4 data mover servers and 2,232 NL-SAS HDDs.
- 10 * HPDL380 Cassandra DB Compute Nodes each with 512GB of DDR5 memory.
- 5 * Cassandra DB nodes use AMD EPYC 7543P 32-Core CPUs and 5 * use Intel Xeon Gold 6226 CPUs.
- Network: 2 * 200Gbps HDR InfiniBand ports transporting CassandraDB traffic and filesystem reflection table chunks.
- Tape library: 2 * IBM TS4500 tape libraries with 4 * dedicated grippers in each library, 6000 slots of capacity in each library.

- Tape Library drive array: 11 * IBM TS1160 tape devices in each of the two physical libraries.
- Tape Library network: Each tape drive is connected via 16Gbit/sec FC (Fibre Channel) connections.
- The entire HPE DMF7 System definition and parameters can be found here: https://git.new/4kbw0za

Compute nodes to generate workload

- 10 nodes of (2 * 48) core AMD Genoa EPYC CPUs with 1.5TB of DDR5 memory, running Rocky 8.10 OS.
- 1 * 200Gbps HDR InfiniBand port transporting NFS protocol to the GPFS ESS based storage mounts.

Software.

- An IBM ESS Storage Scale System, GPFS v6.1.9.3 running RedHat Enterprise Linux 8.10.
- Ganesha-NFS HA Cluster mode, Storage Scale, GPFS 6.1.9.3, Running RedHat Enterprise Linux 8.10.
- A HPE DMF7 HSM system (7.10) that migrates data from the ESS GPFS filesystem based on policy.
- IOR 4.1.0 from https://github.com/hpc/ior
- SLURM 24.11
- EasyBuild foss/2023a was used for GCC/GNU toolchains.

Datasets / Inputs. There are no external data sets associated with input for this experiment. IOR generates them based upon the flags and parameters mentioned in this AD as follows.

https://git.new/Ci0k2ry

Installation and Deployment.

- IOR must be compiled to use OpenMPI 4.1.5 with IOR is compiled as version 4.1.0 current from the stable branch of the GitHub Repository.
- GPFS (StorageScale) binaries installed on the ESS cluster are version 6.1.9.3, available from IBM under commercial terms.
- Matching GPFS client binaries must be installed on each participating compute node.
- GPFS binary produced "gpl-bin" must be shipped to each participating compute node.
- GPFS cluster must be formed with quorum of 5 nodes on remote mount cluster inside compute node complex.
- GPFS CES-NFS cluster must be exporting ganesha-NFS over IB to each participating compute node.
- HPE DMF7 binaries are installed on the DMF7 worker nodes, mgmt cluster and OpenVault Library servers, available from HPE under commercial terms.
- SLURM must be compiled with pmi2/pmix4 with awareness of the OFED stack/UCX integrations.

Artifact Execution

(1) This is T_1: IOR runs with the slurm submission script on the baseline environment as follows https://git.new/sglJJZi
(2) This is T_2: IOR generates the dataset that will characterize the storage device and produce an IO pattern. This task is dependent on T_1 running after submission to the SLURM queue.
(3) This is T_3: IOR run completes in a period of time. This task is dependent upon T_2's completion.
(4) This is T_4: Data is extracted for analysis. There is no additional interaction. This task is dependent upon T_3's completion.
(5) This is T_5: Data is graphed using Python using MatplotLib, Seaborn and ggplot. This task is dependent upon T_4's completion.

$$T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4 \rightarrow T_5$$

The inputs for the IOR arguments are provided here and they are chosen specifically to characterise a wide range of I/O patterns in both IOPS (io/s) and throughput (MiB/sec). https://git.new/sglJJZi There are 10 iterations for each pattern performed to control for variance and establish a standard deviation measurement. Each node distributes 6 MPI ranks making a total of 64 nodes at 384 MPI processes.

Artifact Analysis (incl. Outputs)

Outputs are found here:

A_1 https://git.new/6xnAv8r - *Baseline Scratch Performance, No Declarative Policy*

Artifacts are unremarkable. They show faster reads than writes and are consistent with an RDMA connected NFS mounted client with a parallel filesystem with the hardware layout found as per this IBM ESS hardware platform. There are no unusual outliers in the results.

2.2 Computational Artifact A_2

This artifact is the ARCHIVAL-HEAVY policy performance of the disk storage system in the experimental environment. It is a policy designed to fully load up the potential I/O interactions of the archival environment. It is the first adjusted experimental parameter.

Relation To Contributions

As per C_3, this artifact relates to the contribution as it is required to establish the impact of a naive HSM policy upon overall storage performance.

Expected Results

An exemplar that characterizes the impact of a naive aggressive HSM policy on the storage array with IOR results exhibiting a generally slower read and write IO performance and read throughput performance compared to the artifacts in C_2. There should be a degradation of storage performance associated with this experimental run.

Expected Reproduction Time (in Minutes)

All items remain the same as A_1 apart from IOR run time, which will take 20 hours.

Hardware. All items remain the same as A_1, but this introduces the use of TS1160 tape drives as I/O consumers due to HSM policy.

Software. All items remain the same as A_1, but this introduces DMF7 HSM policy changes to naively write I/O to tape on a 15 second flush interval.

Datasets / Inputs. There are no external data sets associated with input for this experiment. IOR generates them based upon the flags and parameters mentioned in this AD as follows.

https://git.new/Ci0k2ry

Installation and Deployment. All items remain the same as per A_1, but DMF7 policy changes for the ARCHIVAL-HEAVY policy:

https://git.new/FRwmAtm

Artifact Execution

The workflow is identical as per A_1, but it modifies the DMF7 policy before T_1, such that there is a new step:

(1) This is T_1: DMF7 policy is modifed to enable ARCHIVAL-HEAVY policy.
(2) This is T_2: IOR runs with the slurm submission script on the baseline environment as follows https://git.new/sglJJZi. This task is dependent on T_1 running before the SLURM submission.
(3) This is T_3: IOR generates the dataset that will characterize the storage device and produce an IO pattern. This task is dependent on T_2 running after submission to the SLURM queue.
(4) This is T_4: IOR run completes in a period of time. This task is dependent upon T_3's completion.
(5) This is T_5: Data is extracted for analysis. There is no additional interaction. This task is dependent upon T_4's completion.
(6) This is T_6:Data is graphed using Python using MatplotLib, Seaborn and ggplot. This task is dependent upon T_5's completion.

$T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4 \rightarrow T_5 \rightarrow T_6$

All other aspects of the workload and parameters remain unchanged.

Artifact Analysis (incl. Outputs)

Outputs are found here:

A_2 https://git.new/3M1SNtB - *Archival-HEAVY Declarative Policy Performance*

Artifacts demonstrate a statistically significant difference. Across all results, reads and writes are approximately 35% slower than the baseline run found in A_1.

2.3 Computational Artifact A_3

This artifact is the ARCHIVAL-MOD policy performance of the disk storage system in the experimental environment. It is a policy designed to exemplify safely combining archival and scratch storage. It is the second adjusted experimental parameter.

Relation To Contributions

As per C_6, this artifact relates to the contribution as it is required to establish the ability to control for mixing archival storage and scratch with a declarative HSM policy, demonstrating no performance degradation.

Expected Results

An exemplar that characterizes the impact of declarative filtering HSM policy on the storage array with IOR results exhibiting a similar read and write IO performance and read throughput performance compared to the artifacts in C_2. There should be a very small, degradation of storage performance associated with this experimental run.

Expected Reproduction Time (in Minutes)

All items remain the same as A_1.

Hardware. All items remain the same as A_1, but this introduces the use of TS1160 tape drives as I/O consumers due to HSM policy, that appear unused as the policy filters this interaction.

Software. All items remain the same as A_1, but this introduces DMF7 HSM policy changes to filter I/O, preventing certain paths on the basis of directory structures labelled with xattrs.

Datasets / Inputs. There are no external data sets associated with input for this experiment. IOR generates them based upon the flags and parameters mentioned in this AD as follows.

https://git.new/Ci0k2ry

Installation and Deployment. All items remain the same as per A_1, but DMF7 policy changes for the ARCHIVAL-MOD policy:

https://git.new/FRwmAtm

Artifact Execution

The workflow is identical as per A_1, but it modifies the DMF7 policy before T_1, such that there is a new step:

(1) This is T_1: DMF7 policy is modifed to enable ARCHIVAL-MOD policy.
(2) This is T_2: IOR runs with the slurm submission script on the baseline environment as follows https://git.new/sglJJZi. This task is dependent on T_1 running before the SLURM submission.
(3) This is T_3: IOR generates the dataset that will characterize the storage device and produce an IO pattern. This task is dependent on T_2 running after submission to the SLURM queue.
(4) This is T_4: IOR run completes in a period of time. This task is dependent upon T_3's completion.
(5) This is T_5: Data is extracted for analysis. There is no additional interaction. This task is dependent upon T_4's completion.

(6) This is T_6:Data is graphed using Python using MatplotLib, Seaborn and ggplot. This task is dependent upon T_5's completion.

$T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4 \rightarrow T_5 \rightarrow T_6$

All other aspects of the workload and parameters remain unchanged.

Artifact Analysis (incl. Outputs)

Outputs are found here:

A_3 https://git.new/qmgrBDd - *Archival-MOD Declarative Policy Performance*

Artifacts demonstrate a statistically insignificant difference compared to A_1 but a statistically significant difference compared to A_2. Across all results, reads and writes are approximately 1.5% slower than the baseline run found in A_1.