HPS

Abdellah Omar Adolf
Supervisor: Patrick Höhn

Emerging Trends in Parallel File System

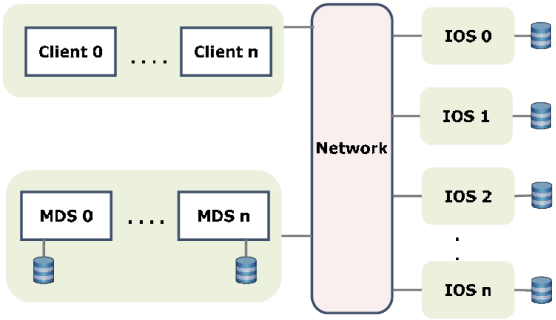
Table of contents

Table of Contents

Definition

A kind of distributed file system that allows data to be stored across multiple networked servers and coordinate input/output operations between clients and storage nodes.

- Often uses a metadata server to store information about the data, such as the file name, location and owner.



Architecture of Parallel File System.
Image source: Shameem and Shaji, “A Methodological Survey on Load Balancing Techniques in Cloud Computing”

Function

- Breaks up a dataset and distributes, or stripes, the blocks to multiple storage drives.
  - The block data can be located in local and/or remote servers.

Function

- Breaks up a dataset and distributes, or stripes, the blocks to multiple storage drives.
  - The block data can be located in local and/or remote servers.
- Reads and writes data to distributed storage devices using multiple I/O paths concurrently.
  - The coordinated use of multiple I/O paths can provide significant performance benefit.

Function

- Breaks up a dataset and distributes, or stripes, the blocks to multiple storage drives.
  - The block data can be located in local and/or remote servers.
- Reads and writes data to distributed storage devices using multiple I/O paths concurrently.
  - The coordinated use of multiple I/O paths can provide significant performance benefit.
- Uses a global namespace to facilitate data access.

Function

- Breaks up a dataset and distributes, or stripes, the blocks to multiple storage drives.
  - The block data can be located in local and/or remote servers.
- Reads and writes data to distributed storage devices using multiple I/O paths concurrently.
  - The coordinated use of multiple I/O paths can provide significant performance benefit.
- Uses a global namespace to facilitate data access.
- Is ideal for HPC workloads that require coordinated I/O and support for large volumes of shared data.

Some of the Major PFS in HPC

- General Parallel File System (GPFS).
- Lustre.
- Parallel Virtual File System.
- Expand 1.0
- BeeGFS

Boito, Pallez, and Teylo, “The role of storage target allocation in applications’ I/O performance with BeeGFS”

Table of Contents

Motivation

- Data growth beyond PB and to EB-scale.
  - Millions of directories and billions of files.
  - Commensurable processing resources needed.
- Metadata also growths proportionally.

Motivation

- Data growth beyond PB and to EB-scale.
  - Millions of directories and billions of files.
  - Commensurable processing resources needed.
- Metadata also growths proportionally.
- In large scale HPC enviroments:
  - Metadata account for up to 80% of total file system operations, Dai et al., “The state of the art of metadata managements in large-scale distributed file systems—Scalability, performance and availability”.

Motivation

- Data growth beyond PB and to EB-scale.
  - Millions of directories and billions of files.
  - Commensurable processing resources needed.
- Metadata also growths proportionally.
- In large scale HPC enviroments:
  - Metadata account for up to 80% of total file system operations, Dai et al., “The state of the art of metadata managements in large-scale distributed file systems—Scalability, performance and availability”.
- Performance bottlenecks in terms of I/O operations.

Motivation

- Data growth beyond PB and to EB-scale.
  - Millions of directories and billions of files.
  - Commensurable processing resources needed.
- Metadata also growths proportionally.
- In large scale HPC enviroments:
  - Metadata account for up to 80% of total file system operations, Dai et al., “The state of the art of metadata managements in large-scale distributed file systems—Scalability, performance and availability”.
- Performance bottlenecks in terms of I/O operations.
- Metadata management efficiency crucial in improving system performance.

Need for emerging trends

File systems need to adopt to these challenges:

- to enhance performance, scalability, efficiency and availability.

Garcia-Carballeira et al., “A new Ad-Hoc parallel file system for HPC environments based on the Expand parallel file system”.

Table of Contents

Selected Emerging Trends

Two emerging metadata management will be presented in this talk.

- AdaM: Adaptive Metadata Management scheme based on Deep Reinforcement Learning.
- Duplex: A scalable metadata service based on full-path indexing.

Table of Contents

AdaM

Insight

- Traditional file systems use centralised strategy with one Metadata Server (MDS).

AdaM

Insight

- Traditional file systems use centralised strategy with one Metadata Server (MDS).
- Simple design but suffers heavy concurrent access and single point of failure.

AdaM

Insight

- Traditional file systems use centralised strategy with one Metadata Server (MDS).
- Simple design but suffers heavy concurrent access and single point of failure.
  - Efficiency, availability and reliability problems.

AdaM

Insight

- Traditional file systems use centralised strategy with one Metadata Server (MDS).
- Simple design but suffers heavy concurrent access and single point of failure.
  - Efficiency, availability and reliability problems.
- Allocation of metadata nodes across multiple MDSs can help to solve address single MDS problem.

AdaM

Insight

- The ultimate goal is to distribute metadata nodes in the namespace tree such that:

AdaM

Insight

- The ultimate goal is to distribute metadata nodes in the namespace tree such that:
  - there is workload balance among MDSs.

AdaM

Insight

- The ultimate goal is to distribute metadata nodes in the namespace tree such that:
  - there is workload balance among MDSs.
  - there is metadata node locality (spatial locality) preservation.

AdaM

Insight

- The ultimate goal is to distribute metadata nodes in the namespace tree such that:
  - there is workload balance among MDSs.
  - there is metadata node locality (spatial locality) preservation.
- However, designing such schemes can be challenging.

AdaM

Insight

- The ultimate goal is to distribute metadata nodes in the namespace tree such that:
  - there is workload balance among MDSs.
  - there is metadata node locality (spatial locality) preservation.
- However, designing such schemes can be challenging.
  - file access patterns are uncertain and time-varying.

AdaM

Insight

- The ultimate goal is to distribute metadata nodes in the namespace tree such that:
  - there is workload balance among MDSs.
  - there is metadata node locality (spatial locality) preservation.
- However, designing such schemes can be challenging.
  - file access patterns are uncertain and time-varying.
  - POSIX standard can present conflict.

AdaM

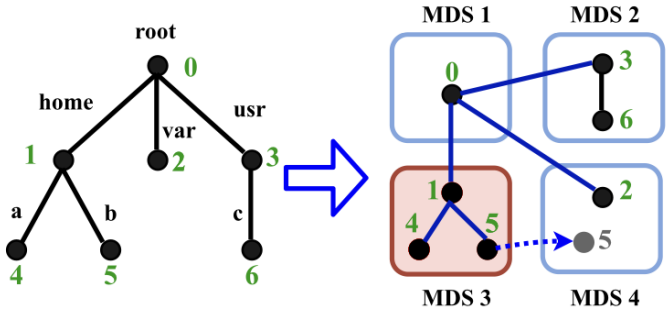


Figure: Partition the entire namespace tree, distribute among four MDS's. Then, the subtree "/home" in MDS3 becomes a hot spot.

Image sources: Huang et al., "An Adaptive Metadata Management Scheme Based on Deep Reinforcement Learning for Large-Scale Distributed File Systems"

AdaM

- AdaM is an "adaptive fine-grained metadata management scheme".

AdaM

- AdaM is an "adaptive fine-grained metadata management scheme".
- migrates “hot” metadata nodes to different MDSs by learning from:

AdaM

- AdaM is an "adaptive fine-grained metadata management scheme".
- migrates “hot” metadata nodes to different MDSs by learning from:
  - access pattern,

AdaM

- AdaM is an "adaptive fine-grained metadata management scheme".
- migrates “hot” metadata nodes to different MDSs by learning from:
  - access pattern,
  - the structure of namespace tree,

AdaM

- AdaM is an "adaptive fine-grained metadata management scheme".
- migrates “hot” metadata nodes to different MDSs by learning from:
  - access pattern,
  - the structure of namespace tree,
  - current distribution of nodes on MDSs.

AdaM

- Automatically does migration among servers.
- Ensures load balance.
- Preserves metadata locality.

AdaM Design Architecture

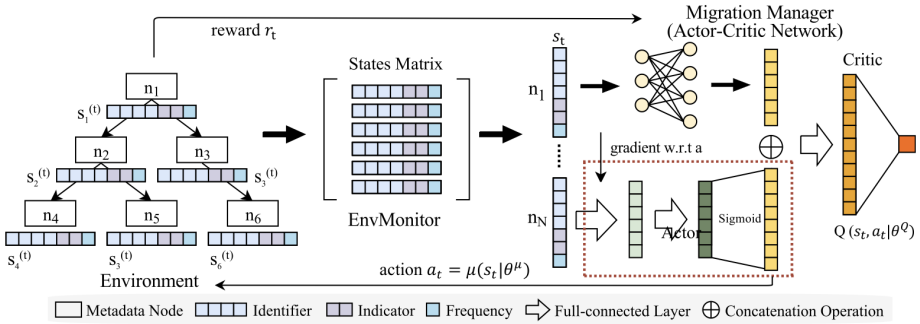


Figure: "A graphical illustration of AdaM.

Image source: Huang et al., "An Adaptive Metadata Management Scheme Based on Deep Reinforcement Learning for Large-Scale Distributed File Systems"

Development of Metadata Management Techniques over the years

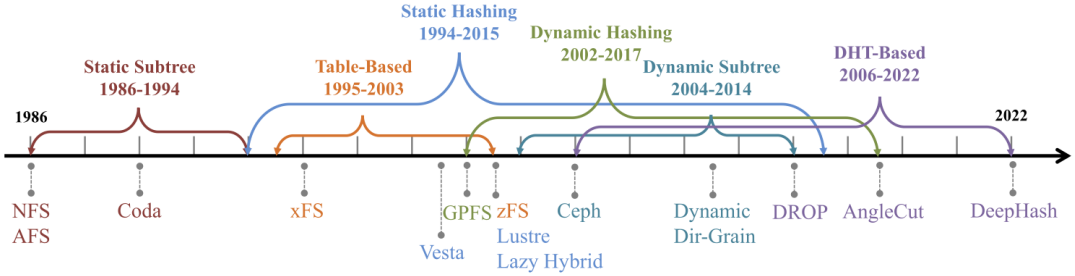


Figure: Timeline of milestones in the development of metadata management techniques.

Image source: Huang et al., “An Adaptive Metadata Management Scheme Based on Deep Reinforcement Learning for Large-Scale Distributed File Systems”

Table of Contents

Why Duplex?

- Traditional file systems use a hierarchical tree-based structure to organize metadata.

Why Duplex?

- Traditional file systems use a hierarchical tree-based structure to organize metadata.
- In this setup, directories act as branch nodes, while files are represented as leaf nodes.

Why Duplex?

- Traditional file systems use a hierarchical tree-based structure to organize metadata.
- In this setup, directories act as branch nodes, while files are represented as leaf nodes.
- For example, GoogleFS manages the entire directory tree within a dedicated Metadata Server (MDS).

Why Duplex?

- Traditional file systems use a hierarchical tree-based structure to organize metadata.
- In this setup, directories act as branch nodes, while files are represented as leaf nodes.
- For example, GoogleFS manages the entire directory tree within a dedicated Metadata Server (MDS).
- However, this centralized approach can lead to:

Why Duplex?

- Traditional file systems use a hierarchical tree-based structure to organize metadata.
- In this setup, directories act as branch nodes, while files are represented as leaf nodes.
- For example, GoogleFS manages the entire directory tree within a dedicated Metadata Server (MDS).
- However, this centralized approach can lead to:
  - performance bottlenecks

Why Duplex?

- Traditional file systems use a hierarchical tree-based structure to organize metadata.
- In this setup, directories act as branch nodes, while files are represented as leaf nodes.
- For example, GoogleFS manages the entire directory tree within a dedicated Metadata Server (MDS).
- However, this centralized approach can lead to:
  - performance bottlenecks
  - system overload due to intensive concurrent access.

Why Duplex?

- Traditional file systems use a hierarchical tree-based structure to organize metadata.
- In this setup, directories act as branch nodes, while files are represented as leaf nodes.
- For example, GoogleFS manages the entire directory tree within a dedicated Metadata Server (MDS).
- However, this centralized approach can lead to:
  - performance bottlenecks
  - system overload due to intensive concurrent access.
  - single point of failure (reliability)

Why Duplex?

- Traditional file systems use a hierarchical tree-based structure to organize metadata.
- In this setup, directories act as branch nodes, while files are represented as leaf nodes.
- For example, GoogleFS manages the entire directory tree within a dedicated Metadata Server (MDS).
- However, this centralized approach can lead to:
  - performance bottlenecks
  - system overload due to intensive concurrent access.
  - single point of failure (reliability)
  - scalability challenges.

Duplex

Simple Scalability improvement approach

- Some File Systems use multiple servers.

Duplex

Simple Scalability improvement approach

- Some File Systems use multiple servers.
- Each subtree is managed by a distinct server.

Duplex

Simple Scalability improvement approach

- Some File Systems use multiple servers.
- Each subtree is managed by a distinct server.
- However, this method can result in workload imbalances among MDSs,

Duplex

Simple Scalability improvement approach

- Some File Systems use multiple servers.
- Each subtree is managed by a distinct server.
- However, this method can result in workload imbalances among MDSs,
  - especially as workloads change dynamically.

Duplex

Simple Scalability improvement approach

- Some File Systems use multiple servers.
- Each subtree is managed by a distinct server.
- However, this method can result in workload imbalances among MDSs,
  - especially as workloads change dynamically.
- Access hotspots at the root directory.

Duplex

Simple Scalability improvement approach

- Some File Systems use multiple servers.
- Each subtree is managed by a distinct server.
- However, this method can result in workload imbalances among MDSs,
  - especially as workloads change dynamically.
- Access hotspots at the root directory.
- Other approach involves flattened metadata management.

Duplex

Design and Architecture

Duplex adheres to three fundamental architecture to optimise the indexing mechanism metadata management:

- Dual access path that support POSIX-style permission verification:
  - fast access path featuring a centralized permission server (PMS) for low latency access.
  - Slow path with high throughput.
- Tree-based permission merging algorithm to reduce the PMS's space footprint.
- flattened metadata management based on double consistent hashing.
  - This enables low-latency access to deep files and super directories.

Duplex

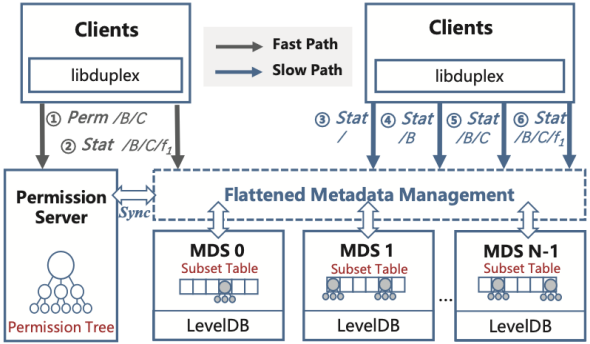


Figure: Duplex Architecture.

Image source: Dong et al., “Low-Latency and Scalable Full-path Indexing Metadata Service for Distributed File Systems”

Duplex

How the permission server works on the namespace?

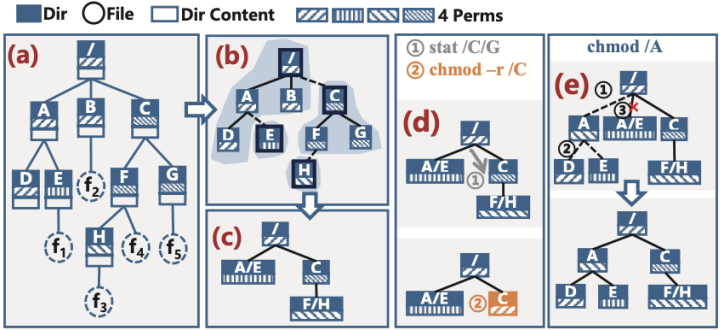


Figure: (a) The original directory tree; (b) The initial permission tree generated from (a); (c) The merged permission tree from (b); (d) A lookup and a recursive update to (c); (e) A non-recursive update to (c).

Image source: Dong et al., “Low-Latency and Scalable Full-path Indexing Metadata Service for Distributed File Systems”

Duplex

Evaluations, Dong et al., “Low-Latency and Scalable Full-path Indexing Metadata Service for Distributed File Systems”

- Duplex shows significant improvement .

Duplex

Evaluations, Dong et al., “Low-Latency and Scalable Full-path Indexing Metadata Service for Distributed File Systems”

- Duplex shows significant improvement .
- Metadata-intensive benchmarks.
  Significantly reduces:

Duplex

Evaluations, Dong et al., “Low-Latency and Scalable Full-path Indexing Metadata Service for Distributed File Systems”

- Duplex shows significant improvement .
- Metadata-intensive benchmarks.
  Significantly reduces:
  - The average lookup latency by up to 84%

Duplex

Evaluations, Dong et al., “Low-Latency and Scalable Full-path Indexing Metadata Service for Distributed File Systems”

- Duplex shows significant improvement .
- Metadata-intensive benchmarks.
  Significantly reduces:
  - The average lookup latency by up to 84%
- Enhancement in lookup IOPS:

Duplex

Evaluations, Dong et al., “Low-Latency and Scalable Full-path Indexing Metadata Service for Distributed File Systems”

- Duplex shows significant improvement .
- Metadata-intensive benchmarks.
  Significantly reduces:
  - The average lookup latency by up to 84%
- Enhancement in lookup IOPS:
  - Up to 7.6 times against CephFS

Duplex

Evaluations, Dong et al., “Low-Latency and Scalable Full-path Indexing Metadata Service for Distributed File Systems”

- Duplex shows significant improvement .
- Metadata-intensive benchmarks.
  Significantly reduces:
  - The average lookup latency by up to 84%
- Enhancement in lookup IOPS:
  - Up to 7.6 times against CephFS
  - Up to 2.3 times when compared to BeeGFS.

Table of Contents

In this talk, two emerging trends for metadata management have been presented:

First, an Adaptive Metadata Management Scheme (AdaM) for migrating hot metadata nodes to different MDSs.

- AdaM has two main components:
  - Environment Monitor
  - Migration Manager.
- Environment Monitor collects node information on the namespace tree.
  - Access pattern
  - Structure of namespace tree
  - Current distribution of nodes on MDSs
- Migration manager uses this information to migrate hot metadata nodes across MDSs to ensure load balance while preserving node locality.

Second, Duplex for fast permission verification to reduce high and unstable latency to super directories.
Duplex has three key designs:

- fast access path with centralized permission server for quick permission verification.
- permission merging algorithm to avoid overburdening the permission server.
- flattened metadata management based on DCH for low latency access to super directories.

With that I look forward to your questions.

Boito, Francieli, Guillaume Pallez, and Luan Teylo. “The role of storage target allocation in applications’ I/O performance with BeeGFS”. In: *2022 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE. 2022, pp. 267–277.

Dai, Hao et al. “The state of the art of metadata managements in large-scale distributed file systems—Scalability, performance and availability”. In: *IEEE Transactions on Parallel and Distributed Systems* 33.12 (2022), pp. 3850–3869.

Dong, Chao et al. “Low-Latency and Scalable Full-path Indexing Metadata Service for Distributed File Systems”. In: *2023 IEEE 41st International Conference on Computer Design (ICCD)*. IEEE. 2023, pp. 283–290.

Garcia-Carballeira, Felix et al. “A new Ad-Hoc parallel file system for HPC environments based on the Expand parallel file system”. In: *2023 22nd International Symposium on Parallel and Distributed Computing (ISPDC)*. IEEE. 2023, pp. 69–76.

Huang, Xiuqi et al. “An Adaptive Metadata Management Scheme Based on Deep Reinforcement Learning for Large-Scale Distributed File Systems”. In: *IEEE/ACM Transactions on Networking* (2023).

Shameem, P. and R.S. Shaji. “A Methodological Survey on Load Balancing Techniques in Cloud Computing”. In: *Asian Journal of Information Technology* 12 (Oct. 2013), pp. 160–169. DOI: 10.3923/ajit.2013.160.169.