Hot Topic: Enabling Cross-Layer Optimizations in Storage Systems with Custom Metadata

Elizeu Santos-Neto [1], Samer Al-Kiswany [1], Nazareno Andrade [2,1]
Sathish Gopalakrishnan [1], Matei Ripeanu [1]

[1] University of British Columbia
4075 - 2332 Main Mall
Vancouver, BC, Canada
{elizeus, samera, sathish, matei}@ece.ubc.ca

[2] Universidade Federal de Campina Grande
Av. Aprígio Veloso, 882
Campina Grande, PB, Brazil
nazareno@lsd.ufcg.edu.br

ABSTRACT

Today, several data-storage systems allow applications to create and manage custom metadata to improve data search and navigability in large-scale storage systems.

Our thesis is that, besides improving search and navigability, custom metadata can also serve as a two-way communication mechanism between applications and the storage layer to enable cross-layer optimizations in a uniform, application-independent and incremental fashion.

Categories and Subject Descriptors

D.4.3 [**Operating Systems**]: File Systems Management – *distributed file systems.* H.3.4 [**Information Storage and Retrieval**]: Systems and Software – d*istributed systems, Performance evaluation (efficiency and effectiveness).*

General Terms

Performance, Design, Standardization.

Keywords

Cross-layer optimization, distributed storage systems, custom metadata.

1. INTRODUCTION

Reinsel et al. [16] estimate that 160 ExaBytes of data were created, stored and replicated on digital media in 2006. Custom metadata has been used as a way to help applications cope with this information overload [1-4, 7, 11, 12]. The benefits of incorporating custom metadata features in storage systems include improved search and navigation [2, 7, 19]. Essentially, all these benefits are realized by using metadata to implicitly *communicate among applications* that use the same data.

Our thesis is that besides allowing communication among software on the application layer, *custom metadata can be used as a bidirectional communication channel between applications and the storage systems to enable cross-layer optimizations* that are hindered today by an ossified file-system interface. Possible cross-layer optimizations include:

- Applications can provide hints to the storage system about their future behaviour – future data use (e.g., co-usage), ideal data placement, or predicted data lifetime (i.e. temporary scratch files, or persistent results) – which can all be used to optimize the performance of the storage layer.
- The storage system can use metadata as a mechanism to expose, in a standard way, the key attributes of the data items stored. For example, a distributed storage system can provide information about files' location, thus enabling data-aware scheduling.
- Metadata can be used to express application-driven quality of service (QoS) requirements (reliability, availability, throughput, or security/privacy) at the data-item level.

Exposing information between different system layers implies tradeoffs between performance and transparency. The use of metadata as an information exposure mechanism between storage system layers enables experimentation within the performance/transparency tradeoffs space.

The rest of this paper elaborates on the argument that custom metadata can benefit storage systems by enabling cross-layer optimization and focuses on the following questions: *What are the potential cross-layer optimization scenarios enabled by custom metadata? How has custom metadata been used in the past? What lessons should we learn from the past to improve the design future systems?*

2. CROSS-LAYER OPTIMIZATIONS IN STORAGE SYSTEMS

Cross layer optimizations have proved essential in a number of computing systems [20-22]. For example, in wireless networks, the interaction between the low reliability of the data transmission channel and TCP congestion control mechanisms leads to inadequate performance. Using cross-layer mechanisms to convey channel capabilities to the upper network layers enables optimized transport layer operation [22]. HTTP caching is a second example: the application layer provides information, through HTML caching directives, for optimizing the behaviour of lower layers (i.e., the caching and data transfer mechanisms).

We posit that storage systems, just like communication systems, can benefit from *cross-layer optimizations*. Such optimizations for storage systems can be enabled using custom metadata as the *communication mechanism* between applications and the storage system layer and can unlock sizable efficiency gains. On one side, applications can use a specialized metadata interface to convey hints about their data usage patterns to the storage system. These hints can be used by the storage system to optimize its performance. On the other side, the storage system can use

HPDC'08, June 23–27, 2008, Boston, Massachusetts, USA.

metadata to provide applications with information only available at the storage level (e.g., data placement, caching status, status of a replication process, inconsistency details). The rest of the section discusses scenarios where this two-way communication mechanism provides benefits.

Application to storage system communication. Applications may convey hints about their requirements to the storage system such as performance, QoS, or replica management directives. We discuss a number of such scenarios below.

Performance: Recent studies [10, 14] have revealed particular characteristics of data usage in scientific applications: *files are often used in groups* of tens to hundreds of files. Metadata can enable applications to explicitly *communicate* file dependencies and co-usage information. This information can be used to improve storage system performance through optimized caching, prefetching, and/or data placement decisions.

A workflow-aware data storage system can, for instance, use the fact that files are annotated as 'temporary' (e.g., between workflow stages) to make data placement decisions (e.g., cache temporary files in fast storage, perhaps with limited reliability) or decisions about data lifetime (e.g., automatically purge old temporary files, left behind by a buggy workflow or by a workflow runtime engine crash).

QoS requirements. Different data items can have different, application-driven QoS requirements (e.g., access performance, availability or durability, and, possibly, security and privacy). A storage system that is aware of these requirements can optimize using individual items' QoS requirements rather than pessimistically provision for the most demanding QoS level. Metadata can be used to express these requirements.

Versioning. Applications can use metadata to indicate modifications to files. This will allow a file system to manage versions efficiently. Currently, versioning uses a combination of timestamps and data comparisons. However file timestamps can be altered without any change to data; metadata can reflect this and avoid an additional copy of such files when an incremental backup is performed. Similarly, metadata could be used to indicate if changes between versions are major or minor. This information could be used, when bandwidth is limited, to decide whether the latest version of a file must be obtained or a locally cached version, even if slightly outdated, can be used.

Consistency requirements. Applications can use metadata to inform the storage system about their data consistency requirements [17]. Making the choice of the consistency requirements flexible allows the application to manage the tradeoffs between performance and consistency, while enabling the storage system to allocate more resources to the data items that have stringent requirements.

Storage system to application communication. Communication across layers can occur in the reverse direction as well: from the storage layer to the application. For example, effective scheduling decisions for data-intensive applications do need to take into account data location information, as previous studies show [5, 17] . Ad-hoc solutions employed by existing scientific workflow runtime engines (e.g., Falkon [15] and [13]) to track data location could well be replaced by a uniform metadata interface exposed by the storage system.

Additional storage-level information (e.g., replication count, information about possible inconsistencies between replicas, properties and status of the storage device) could be useful when making application level-decisions as well (e.g., scheduling, data-loss risk evaluation).

3. PAST USES OF CUSTOM METADATA

File systems that support custom metadata – in addition to traditional metadata (i.e., creation and modification time, file size, etc.) – date back to the Semantic File System (SFS)[7]. SFS aims to reduce the complexity of writing data processing applications. For example, writing a program that manipulates large quantities of data spread across multiple files is much easier if the relevant information is aggregated into one file (or a single directory). To this end, SFS allows the user to aggregate the result of extracting characteristics from collections of files (e.g., the lines include a particular string) as a virtual directory, which is the abstraction used to provide flexible views over data.

Similar to SFS, Metafs [1], Haystack [2], The Linking File System (LiFS) [4] and faceted search [11] extend the traditional file system interface with a metadata interface that allows applications to create arbitrary metadata. These efforts use different approaches for providing applications the functionality of annotating files with arbitrary <key,value> pairs and/or to express relationships among files.

A more recent use of custom metadata is present in Grid systems. Globus Tool Kit, for instance, offer a separate metadata service (e.g., the Metadata Catalog Service [19]) as an independent service. The metadata augments data objects with application-specific descriptions. These descriptions are typically structured using a community-standard schema to ease search for and grouping of data objects.

Finally, Graffiti [12] is a middleware that allows tagging and sharing of tags between desktop file systems. As other systems which aim at providing a metadata interface, it supports tags and links between files, but focuses on sharing-related issues.

All solutions presented above essentially use metadata to communicate between applications. Their main focus is on providing better search, navigability and data organization capabilities on the application layer with data produced by the applications themselves; the storage system does not produce or makes use of the metadata.

Apart from these, some other less common benefits are achieved through metadata in similar systems:

- *Support for application-level consistency through application-defined dependencies among files.* For example, to detect whether library updates will break applications due to version mismatches [4].
- *Support for data provenance*. Richer metadata can be used to keep track of data provenance, for example the source site for downloaded files or a workflow definition for derived computational data [18].
- *Simplified application development*. Email clients, multimedia players, or desktop search engines often maintain custom metadata per file. In traditional file systems, these applications maintain this metadata at the application level. However, these solutions are not standardized, which may preclude their reuse across applications and platforms.Having the richer metadata

mechanism in the file system will simplify these applications development and enable communication across application.

4. DISCUSSION

This section considers the design of a storage system that supports custom metadata and discusses the opportunities and challenges brought by metadata use to support cross-layer communication.

4.1 Design Considerations

Traditional file-system design, is an 'hourglass' design, similar to that of the network stack (Figure 1). In the network stack, the IP layer works as the 'neck' of the hourglass that enables communication between different link- and transport-layer implementations as long as they communicate via IP datagrams.

In storage systems, the traditional (and after decades of use, convenient) POSIX file system API performs a similar role: enables transparent cooperation between applications and storage systems. However, unlike IP, which is provisioned with optional fields in its original design, the traditional file system API does not provide any mechanism for unanticipated cross-layer communication. In the IP case, the optional fields in the IPv4 header and the extension headers in IPv6 can be leveraged by different layers to exchange information unanticipated by system designers. In fact, IP options fields are used by transport protocols to provide hints to lower layers in wireless environments [8, 9]. The traditional file-system interface, on the other hand, limits applications to an ossified list of metadata attributes and value ranges. We argue that as in the IP example, the file system API should include an extensible mechanism for communication between different layers in the application stack.

Two other design lessons relevant to storage system design can be borrowed from the design of the network stack: First, applications and storage infrastructure should consider metadata as hints rather than hard directives, that is metadata might or might not be used at the other layers of the system, and, second, the addition of metadata should not impact on the efficiency of applications or users not using it. If a system does not use the metadata existing in the system, its performance should not be affected or there will be a disincentive for the metadata interface to be deployed. IPv4 optional fields affect the size of the datagram header. This has the undesirable side effect of demanding more resources from routers to process datagrams which optional fields. IP routers are then usually configured to handle IP packets with/without these fields differently, dropping those which use options more frequently [6].

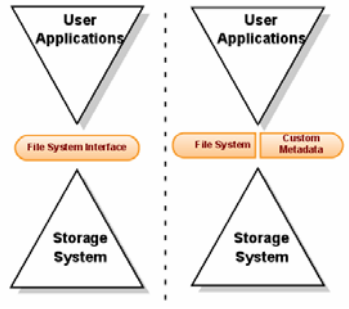

Figure 1: The traditional file system interface view (left) and the file system interface plus custom metadata features (right). The file system interface represents the confluence point between the user-level applications and the storage services.

4.2 Opportunities

An application-agnostic cross-communication layer: A custom metadata interface, as simple as the possibility of annotating files with arbitrary *<key,value>* pairs, can be exploited as an application-independent communication channel between applications and the storage layer.

Incremental transition path: Applications and the storage layer can enable an incremental adoption process exploiting custom metadata opportunistically (i.e., treat it as hints rather than hard directives).

New opportunities for usage-based optimization mechanisms: We believe that the usage-based optimization scenarios presented in the previous section are only the tip of the iceberg in terms of possible application-informed optimizations in storage systems. Thus far, innovation in this area has been stymied by the complexity of developing ad-hoc solutions to expose application-level information and to manage this information consistently within the storage system. Standard metadata interfaces and integrated management will lower the cost of exploration and lead to innovative uses of information available only to users/applications to optimize the storage system.

4.3 Challenges

Standardization is required to make progress: Low coupling between applications and storage services through metadata requires new conventions/standards for expressing application-level knowledge/requirements as metadata.

Cross-layer communication and the optimizations enabled should not break the separation of concerns among different layers: We should be careful about cross-layer design. Layering helps manage complexity by separating concerns. Therefore, it is necessary to devise mechanisms that limit the interference one layer may cause on others even though, as we argue, there are benefits in allowing information cross between layers.

Conflicting metadata: Metadata management should have mechanisms to detect and resolve conflicting metadata. For example, an application could mark a file as temporary, while another application could request three remote replicas for the same file. The policies associated with metadata management should decide in case of conflict: they may deny conflicting annotations, or the storage service may serve the strongest requirement.

Access control mechanisms may need to be enhanced to regulate access to metadata at fine granularity, especially in shared infrastructures.

Implementation complexity and scalability: Past experience suggests that, while the scalability of the metadata subsystem is crucial, its design and implementation are non-trivial endeavours. As an example, the choice to integrate metadata management directly within the file system or to decouple metadata management by building a separate service affects the implementation significantly and requires better analysis. We note that Data Grid architectures focus on decoupling, to the extent possible, all services that support storage (e.g., MCS [19]); this approach eases modifications to metadata management but may have greater performance penalties.

Policy vs. Mechanism: The cross-layer communication enables the application to request preferential treatment for certain data

objects. However, the mechanism alone is not sufficient to prevent a "tragedy of the commons", where all applications demand the highest QoS possible leading the system to the regime equivalent to the absence of differentiated QoS. Policies are required to prevent applications from "abusing" the cross layer communication mechanism. These policies may take the form of admission control or system-wide QoS optimization.

Incentive-compatible adoption: If the benefit a user receives from deploying a mechanism is realized only when the mechanism is already widely adopted, it might be difficult to cross the chasm between early adopters and mainstream users. For example, drawing another example from the IP protocol, Fonseca et al. [6] show that packets with IP options (which maps to custom metadata in storage systems) tend to have higher drop rate in comparison to packets without IP options. If a particular site/ISP that does not have applications that benefit from IP options, it does not have any incentive to process such packets. This, in turn, works as a disincentive for developers, who do not use of IP options and avoid poor service. Custom metadata mechanisms should try to provide benefits to system operators and applications even if they are not adopted by the entire user base.

5. SUMMARY

To date extensive custom metadata has been exclusively used to improve the usability (e.g., search, navigability) of large-scale storage systems. We take a complementary view and argue that native support for custom metadata is a key information exposure mechanism required to enable cross-layer optimizations in storage systems. In this paper, we present usage scenarios that highlight the potential benefits and challenges of exploiting custom metadata as communication mechanism between application and storage layers. Moreover, we review previous efforts in designing support for custom metadata features and propose considerations in conceiving future systems.

6. REFERENCES

1. *MetaFS*, http://metafs.sourceforge.net/. 2008.
2. Adar, E., D. Karger, and L. Stein. *Haystack: Per-User Information Environments*. in *International conference on Information and knowledge management*. 1999.
3. Ames, S., et al. *Richer file system metadata using links and attributes*. in *the 22nd IEEE / 13th NASA Goddard Conference on Mass Storage Systems and Technologies*. 2005.
4. Ames, S., et al. *LiFS: An attribute-rich file system for storage class memories*. in *23rd IEEE / 14th NASA Goddard Conference on Mass Storage Systems and Technologies*. 2006.
5. Casanova, H., et al., *Heuristics for Scheduling Parameter Sweep applications in Grid environments*, in *Heterogeneous Computing Workshop (HCW)*. 2000.
6. Fonseca, R., et al., *IP Options are not an option*, in *Technical Report No. UCB/EECS-2005-24*. 2005.
7. Gifford, D.K., et al. *Semantic file systems*. in *CM SIGOPS Operating Systems*. 1991.
8. Gurtov, A. and S. Floyd, *Modeling wireless links for transport protocols.* ACM SIGCOMM Computer Communication Review, 2004. **34**(2): p. 85 - 96.
9. Gurtov, A. and R. Ludwig, *Lifetime packet discard for efficient real-time transport over cellular links.* ACM SIGMOBILE Mobile Computing and Communications Review, 2003. **7**(4): p. 32 - 45.
10. Iamnitchi, A., S. Doraimani, and G. Garzoglio. *Filecules in High-Energy Physics: Characteristics and Impact on Resource Management*. in *HPDC 2006*. 2006. France.
11. Koren, J., et al. *Searching and Navigating Petabyte Scale File Systems Based on Facets*. in *Workshops of International Conference for High Performance Computing, Networking, Storage, and Analysis (SC '07)*. 2004. Reno, NV.
12. Maltzahn, C., et al. *Graffiti: A Framework for Testing Collaborative Distributed Metadata*. in *Informatics 21*. 2007.
13. Mandal, A., et al. *Scheduling Strategies for Mapping Application Workflows onto the Grid*. in *IEEE International Symposium on High Performance Distributed Computing*. 2005.
14. Otoo, A.E., D. Rotem, and A. Romosan. *Optimal File-Bundle Caching Algorithms for Data-Grids*. in *Supercomputing*. 2004.
15. Raicu, I., et al. *Falkon: a Fast and Light-weight tasK executiON framework*. in *SuperComputing*. 2007.
16. Reinsel, D., *A Forecast of Worldwide Information Growth Through 2010*, in *IDC White Paper*. 2007.
17. Santos-Neto, E., et al. *Exploiting replication and data reuse to efficiently schedule data-intensive applications on grids*. in *Workshop on Job Scheduling Strategies for Parallel Processing*. 2004.
18. Simmhan, Y.L., B. Plale, and D. Gannon. *A survey of data provenance in e-science*. in *ACM SIGMOD Record*. 2005.
19. Singh, G., et al. *A Metadata Catalog Service for Data Intensive Applications*. in *Supercomputing*. 2003.
20. Skraba, P., et al. *Cross-layer optimization for high density sensor networks: Distributed passive routing*. in *Ad-Hoc, Mobile, and Wireless Networks (ADHOC-NOW)*. 2004. Berlin, Germany.
21. Wang, J., M. Venkatachalam, and Y. Fang, *System Architecture and Cross-Layer Optimization of Video Broadcast over WiMAX*. IEEE Journal on Selected Areas in Communications, 2007. **25**(4).
22. Warrier, A., L. Le, and I. Rhee. *Cross-layer Optimization Made Practical*. in *IEEE International Conference of Broadband Communications, Networks, and Systems (Broadnets)*. 2007.