ARTICLE IN PRESS

Future Generation Computer Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A cross-layer optimized storage system for workflow applications

Samer Al-Kiswany [a,*], Lauro B. Costa [b], Hao Yang [c], Emalayan Vairavanathan [d], Matei Ripeanu [c]

[a] University of Waterloo, Canada
[b] Google Inc., United States
[c] University of British Columbia, Canada
[d] NetApp Inc., Canada

ARTICLE INFO

Article history:
Received 30 April 2016
Received in revised form
18 February 2017
Accepted 24 February 2017
Available online xxxx

Keywords:
Cross layer optimizations
Distributed file systems
Workflow management
Batch processing systems

ABSTRACT

This paper proposes using file system custom metadata as a bidirectional communication channel between applications and the storage middleware. This channel can be used to pass hints that enable cross-layer optimizations, an option hindered today by the ossified file-system interface. We study this approach in the context of storage system support for large-scale workflow execution systems: Our workflow-optimized storage system (WOSS), exploits application hints to provide per-file optimized operations, and exposes data location to enable location-aware scheduling. We argue that an incremental adoption path for adopting cross-layer optimizations in storage exists, present the system architecture for a workflow-optimized storage system and its integration with a workflow runtime engine, and evaluate this approach using synthetic and real applications over multiple success metrics (application runtime, generated network stress, and energy). Our performance evaluation demonstrates that this design brings sizeable performance gains. On a large scale cluster (100 nodes), compared to two production class distributed storage systems (Ceph and GlusterFS), WOSS achieves up to 6× better performance for the synthetic benchmarks and 20–40% better application-level performance gain for real applications.

1. Introduction

Custom metadata features (a.k.a., 'tagging') have seen increased adoption in systems that support the storage, management, and analysis of 'big-data'. However, the benefits expected are all essentially realized at the application level either by using metadata to present richer or differently organized information to users (e.g., enabling better search and navigability [1,2]) or by implicitly communicating among applications that use the same data items (e.g., to support provenance, or inter-application coordination).

Our thesis *is that*, besides the above uses, *custom metadata can be used as a bidirectional communication channel between applications and the storage system* and thus become the key enabler for cross-layer optimizations that, today, are hindered by an ossified file-system interface.

This communication channel is *bidirectional* as the cross-layer optimizations enabled are based on information passed in both directions across the storage system interface (i.e., application to storage and storage to application). Possible cross-layer optimizations include:

- (*top-down*) Applications can use metadata to provide hints to the storage system about their future behavior, such as: per-file access patterns, ideal data placement (e.g., co-usage), predicted file lifetime (i.e., temporary files vs. persistent results), access locality in a distributed setting, desired file replication level, or desired quality of service. These hints can be used to optimize the storage layer.
- (*bottom-up*) The storage system can use metadata as a mechanism to expose key attributes of the data items stored. For example, a distributed storage system can provide information about data location, thus enabling location-aware scheduling.

The approach we propose has four interrelated advantages: it uses an application-agnostic mechanism, it is incremental, it offers

* Correspondence to: David R. Cheriton School of Computer Science, University of Waterloo, 200 University Avenue West, Waterloo, ON, Canada N2L 3G1.
E-mail addresses: alkiswany@uwaterloo.ca (S. Al-Kiswany), lbeltrao@gmail.com (L.B. Costa), haoy@ece.ubc.ca (H. Yang), emalayan@netapp.com (E. Vairavanathan), matei@ece.ubc.ca (M. Ripeanu).

http://dx.doi.org/10.1016/j.future.2017.02.038
0167-739X/

a low cost for experimentation, and it focuses the research community effort on a single storage system prototype, saving considerable development and maintenance effort dedicated, nowadays, to multiple storage systems each targeting a specific workload (e.g., HDFS and PVFS [3]). First, the communication mechanism we propose: simply annotating files with arbitrary ⟨*key*, *value*⟩ pairs, is application-agnostic. Second, our approach enables evolving applications and storage-systems independently while maintaining the current interface (e.g., POSIX), and offers an incremental transition path for legacy applications and storage-systems: A legacy application will still work without changes (yet will not see performance gains) when deployed over a new storage system that supports cross-layer optimizations. Similarly a legacy storage will still support applications that attempt to convey optimization hints, yet it will not offer performance benefits. As storage and applications incrementally add support for passing and reacting to optimization hints, the overall system will see increasing gains. Finally, exposing information between different system layers implies tradeoffs between performance and transparency. To date, these tradeoffs have been scarcely explored. We posit that a flexible encoding (key/value pairs) as the information passing mechanism offers the flexibility to enable low-cost experimentation within this tradeoff space.

The approach we propose falls in the category of 'guided mechanisms' (i.e., solutions for applications to influence data placement, layout, and lifecycle), the focus of other projects as well. In effect, the wide range (and incompatibility) of past solutions proposed in the storage area in the past two decades (and incorporated to some degree by production systems — pNFS, PVFS [3], GPFS [4], Lustre, and other research projects [5–7]), only highlights that *adopting a unifying abstraction is an area of high potential impact*. The novelty of this paper comes from the "elegant simplicity" of the solution we propose. First, unlike past work, we maintain the existing API (predominantly POSIX compatible), and, within this API, we propose using the existing extended file attributes as a flexible, application-agnostic mechanism to pass hints across the application/storage divide. Second, and equally importantly, we propose an extensible storage system architecture that can be extended with application specific optimizations.

We demonstrate our approach by building a POSIX-compatible storage system to efficiently support one application domain: scientific workflows (an application domain detailed in Section 2 and Fig. 1). We chose this domain as this community has to support a large set of legacy applications (developed using the POSIX API). Our storage system is instantiated on-the fly to aggregate the resources of the computing nodes allocated to a batch application (e.g., disks, SSDs, and memory) and offers a shared file-system abstraction with two key features. First, it optimizes the data layout (e.g., file and block placement, file co-placement) to efficiently support the workflow data access patterns (as hinted by the application). Second, the storage system uses custom metadata to expose data location information so that the workflow runtime engine can make location-aware scheduling decisions. These two features are key to efficiently support workflow applications as their generated data access patterns are irregular and application-dependent.

Contributions. This project demonstrates that it is feasible to have a POSIX compatible storage system that can be yet optimized for each application (or application mix) even if the application has a different access pattern for different files. The key contributions of this work are:

- *We propose a new approach* that uses custom metadata to enable cross-layer optimizations between applications and the storage system. Further, we argue that this approach can be adopted incrementally. This suggests an evolution path for co-designing POSIX-compatible file-systems together with the middleware ecosystem they coexist such that performance efficiencies are not lost and flexibility is preserved, a key concern to support legacy applications.
- *We present an extensible storage system architecture* that supports cross-layer optimizations. We demonstrate the viability of this approach through a storage system prototype optimized for workflow applications (dubbed WOSS, Fig. 1). WOSS supports application-informed data placement based on per-file hints, and exposes data location to enable location-aware task scheduling. Importantly, we demonstrate that it is possible to achieve our goals, with only minor changes to the workflow scheduler, and without changing the application code or tasking the developer to annotate their code to reveal the data usage patterns.
- *We demonstrate, using synthetic benchmarks as well as three real-world workflows, that this design brings sizeable performance gains*. On a large scale cluster (100 nodes), compared to two production class distributed storage systems (*Ceph* [8] and *GlusterFS* [9]), WOSS achieves up to 6× higher performance for the synthetic benchmarks and 20%–40% application-level performance gain for real applications.

Organization of this paper. The final section of this paper includes a detailed design discussion and design guidelines, discusses the limitations of this approach, and elaborates on the argument that custom metadata can benefit *generic* storage systems by enabling cross-layer optimizations (Section 5). Before that, we present the context (Section 2), the design (Section 3) and evaluation (Section 4) of a first storage system we designed in this style: *the workflow-optimized storage system (WOSS)*.

2. Background and related work

This section starts by briefly setting up the context: the target application domain and the usage scenario. It then continues with a summary of data access patterns of workflow applications (Section 2.1) and a survey of related work on alleviating the storage bottleneck (Section 2.2.

The application domain: workflow applications. Metaapplications that assemble complex processing workflows using existing applications as their building blocks are increasingly popular in the science domain [10–13]. A popular approach to support these workflows is *the many-task approach* [14], through which the workflow assembles a set of independent processes that communicate through intermediary files stored on a shared POSIX file-system (e.g., Montage workflow detailed in Fig. 15). The dependency between the different executables in a workflow are expressed in scripts or special domain-specific languages [15].

The workflow scheduler schedules the workflow tasks on a set of compute nodes allocated exclusively to the workflow (Fig. 1). The scheduler submits the tasks only when all its input files are available in the storage system. Workflow tasks access the storage system through the Linux file system API.

Three main advantages make most workflow runtime engines adopt this approach: simplicity, direct support for legacy applications and support for fault-tolerance. First, a shared file-system approach simplifies workflow development, deployment and debugging: essentially workflows can be developed on a workstation then deployed on a large machine without changing the environment. Moreover, a shared file-system simplifies workflow debugging as intermediate computation state can be easily inspected at runtime and, if needed, collected for debugging or performance profiling. Second, a shared file-system supports the legacy applications that form the individual workflow stages as these generally use the POSIX API. Finally, this approach simplifies fault-tolerance:

Please cite this article in press as: S. Al-Kiswany, et al., A cross-layer optimized storage system for workflow applications, Future Generation Computer Systems (2017), http://dx.doi.org/10.1016/j.future.2017.02.038