Efficient Scheduling of Scientific Workflows using Hot Metadata in a Multisite Cloud

Ji Liu, Luis Pineda-Morales, Esther Pacitti, Alexandru Costan, Patrick Valduriez, Gabriel Antoniu, Marta Mattoso

Ordonnancement Efficace de Workflows Scientifiques en exploitant les Métadonnées Chaudes dans un Cloud Multisite

Résumé

Les applications scientifiques à grande échelle sont souvent exprimées sous forme de workflows scientifiques (SWfs) qui aident à définir les jobs de traitement des données et les dépendances entre les activités des jobs. Certains SWfs nécessitent une très grande quantité de stockage et de calcul, ce qui peut être obtenu en exploitant plusieurs data centers dans un cloud. Dans ce contexte, la gestion des métadonnées et l'ordonnancement des tâches entre différents data centers deviennent critiques pour l'exécution efficace de SWf. Dans cet article, nous proposons une architecture et un modèle distribués hybrides, en utilisant les métadonnées chaudes (fréquemment consultées) pour l'ordonnancement efficace de SWf dans un cloud multisite. Nous utilisons notre modèle dans un système de gestion de workflows scientifiques (SWfMS) pour valider et régler son applicabilité à différents workflows scientifiques réels avec différents algorithmes d'ordonnancement. Nous montrons que la combinaison d'une gestion efficace des métadonnées chaudes et des algorithmes d'ordonnancement améliore les performances du SWfMS. En évitant les opérations inutiles de métadonnées froides, le temps d'exécution des jobs qui s'exécutent en parallèle est réduit jusqu'à 64,1% et celui de l'ensemble des workflows scientifiques jusqu'à 37,5%.

Efficient Scheduling of Scientific Workflows using Hot Metadata in a Multisite Cloud

Ji Liu
Microsoft Research - Inria Joint Centre, Inria, LIRMM and University of Montpellier
ji.liu@inria.fr

Luis Pineda
Microsoft Research - Inria Joint Centre, Inria and IRISA / INSA Rennes
luis.pineda-morales@inria.fr

Esther Pacitti
LIRMM, University of Montpellier and Inria
esther.pacitti@lirmm.fr

Alexandru Costan
IRISA / INSA Rennes
alexandru.costan@irisa.fr

Patrick Valduriez
Inria, LIRMM and University of Montpellier
patrick.valduriez@inria.fr

Gabriel Antoniu
Inria
gabriel.antoniu@inria.fr

Marta Mattoso
COPPE / UFRJ
marta@cos.ufrj.br

ABSTRACT

Large-scale scientific applications are often expressed as scientific workflows (SWfs) that help defining data processing jobs and dependencies between jobs' activities. Several SWfs have huge storage and computation requirements, and so they need to be processed in multiple (cloud-federated) datacenters. It has been shown that efficient metadata handling plays a key role in the performance of computing systems. However, most of this evidence concern only single-site, HPC systems to date. In addition, the efficient scheduling of tasks among different data centers is critical to the SWf execution. In this paper, we present a hybrid distributed model and architecture, using *hot metadata* (frequently accessed metadata) for efficient SWf scheduling in a *multisite* cloud. We couple our model with a scientific workflow management system (SWfMS) to validate and tune its applicability to different real-life scientific workflows with different scheduling algorithms. We show that the combination of efficient management of hot metadata and scheduling algorithms improves the performance of SWfMS, reducing the execution time of highly parallel jobs up to 64.1% and that of the whole scientific workflows up to 37.5%, by avoiding unnecessary *cold* metadata operations.

KEYWORDS

hot metadata, metadata management, multisite clouds, scientific workflows, geo-distributed applications.

ACM Reference format:
Ji Liu, Luis Pineda, Esther Pacitti, Alexandru Costan, Patrick Valduriez, Gabriel Antoniu, and Marta Mattoso. 2017. Efficient Scheduling of Scientific Workflows using Hot Metadata in a Multisite Cloud. In *Proceedings of BDA, Nancy, France, November 2017,* 12 pages.
https://doi.org/10.475/123_4

BDA, November 2017, Nancy, France

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.475/123_4

1 INTRODUCTION

Many large-scale scientific applications now process amounts of data reaching the order of Petabytes; as the size of the data increases, so do the requirements for computing resources. Clouds stand out as convenient infrastructures for handling such applications, for they offer the possibility to lease resources at a large scale and relatively low cost. Very often, requirements of data-intensive scientific applications exceed the capabilities of a single cloud datacenter (site), either because the site imposes usage limits for fairness and security [10], or simply because the dataset is too large. Also, the application data are often physically stored in different geographic locations, because they are sourced from different experiments, sensing devices or laboratories (*e.g.* the well known ALICE LHC Collaboration spans over 37 countries [1]). Hence multiple datacenters are needed in order to guarantee both that enough resources are available and that data are processed as close to its source as possible. All popular public clouds today account for a range of geo-distributed datacenters, *e.g.* Microsoft Azure [8], Amazon EC2 [2], and Google Cloud [6].

A large number of data-intensive distributed applications are expressed as Scientific Workflows (SWf). A SWf is an assembly of scientific data processing activities with data dependencies between them [16]. The application is modeled as a graph, in which vertices represent processing jobs, and edges their dependencies. Such a structure provides a clear view of the application flow and facilitates the execution of the application in a geo-distributed environment. Currently, many Scientific Workflow Management Systems (SWfMS) are publicly available, *e.g.* Pegasus [17] and Swift [45]; some of them already support multisite execution [29], [30], [31].

Metadata have a critical impact on the efficiency of SWfMS; they provide a global view of data location and enable task tracking

during the execution. Some SWf metadata even need to be persisted to allow traceability and reproducibility of the SWf's jobs, these are part of the so called *provenance* data. Most notably, we assert that some metadata are more frequently accessed than others (e.g. the status of tasks in execution in a multisite SWf is queried more often than a job's creation date). We denote such metadata by *hot metadata* and argue that it should be handled in a specific, more quickly accessible way than the rest of the metadata. Metadata are typically queried to get a global view of data location or to keep track of the tasks execution; they should be readily available to the system at any given time. While it has been proven that efficient metadata handling plays a key role in performance [12, 43], little research has targeted this issue in *multisite clouds*.

On multisite infrastructures, inter-site network latency is much higher than intra-site latency. This aspect must stay at the core of the design of a multisite metadata management system. As we explain in Section 4, several design principles have to be taken into account. Moreover, in most data processing systems (should they be distributed), metadata are typically stored, managed and queried at some centralized server (or set of servers) located at a specific site [17, 22, 39]. However, in a multisite setting, with high-latency inter-site networks and large amounts of concurrent metadata operations, centralized approaches are not an optimal solution.

In addition, in order to enable SWf execution in a multisite cloud with distributed input data within a reasonable time, the execution of the tasks of each job should be efficiently scheduled to a corresponding site. Then, the multisite scheduling process is to use scheduling algorithms to decide at which site to execute the tasks to achieve a given objective, *e.g.* reducing execution time. During the scheduling process, the metadata should also be provisioned to the scheduler to make smart decisions.

A centralized strategy is easy to implement and works well with a few concurrent queries. However, this strategy will be inefficient if the number of queries is high or the bandwidth is low. Furthermore, the input data of a SWf may be distributed at different sites so the tasks of one job may need to be executed at different sites. In this case, a centralized strategy for metadata management will incur additional communications. In this paper, we take a different approach based on the distribution of hot metadata in order to increase the locality of access by the different computing nodes. Inspired by [37], we adapt two distributed metadata management strategies, *i.e.* Distributed Hash Table (DHT) and replication method (Rep), for hot metadata management. DHT stores the hot metadata at the site corresponding to its hash value and Rep stores the hot metadata to the sites where it is generated and to the site corresponding to its hash value. We propose a local storage based hot metadata management strategy, *i.e.* LOC, which stores the hot metadata at the site where it is generated. We take the centralized method as a baseline to show the advantage of the distributed methods.

A major new contribution of this paper is to demonstrate the pertinence of using hot metadata for SWf execution in a multisite cloud. In the context of multisite SWfs, where thousands of tasks are executed across sites, separating hot metadata improves the SWf execution time, yet at no additional cost. In addition, during the SWf execution, some hot metadata may become cold or vice-versa. We show that metadata monitoring and dynamic hot or cold metadata classification can address this dynamic variation of the "temperature" of metadata. We also analyse the limitations of our approaches.

The paper makes the following contributions:

- Based on the notion of *hot metadata*, we introduce a distributed architecture, adapt two strategies to hot metadata management, and propose a local storage based hot metadata management strategy for optimizing the access to hot metadata and ensuring their availability in a multisite cloud environment (Section 5).
- We develop a prototype by coupling our proposed architecture and strategies with a state of the art multisite SWf execution engine, namely Chiron [36], using an RDBMS to manage hot metadata (Section 6).
- We combine the hot metadata management strategies and three scheduling algorithms, *i.e.* OLB, MCT and DIM, by enabling the hot metadata management provisioning for multisite scheduling process (Section 6).
- We demonstrate that efficient management of hot metadata improves the performance of SWfMS, reducing the execution time of a SWf by 1) enabling timely data provisioning and 2) avoiding unnecessary *cold* metadata handling (Section 7). We also show the advantages of decentralized hot metadata management strategies with different scheduling algorithms (Section 7).
- We discuss the related issues raised by the necessity of hot metadata management and dynamic hot metadata (Section 8).

This paper is organized as follows. Section 2 introduces the related work. Section 3 presents the SWf model and the hot metadata and discusses the issues to manage the hot metadata. Section 4 introduces the design principles. Section 5 gives the architecture of the SWf engine and explains our adapted and proposed hot metadata management strategies. Section 6 gives the implementation of our proposed approaches. Section 7 presents our experimental results and Section 8 analyzes the implemented system in terms of the necessity of hot metadata management, dynamic hot metadata and limitations. Finally, Section 9 concludes.

2 RELATED WORK

Most of the existing SWf engines focus on the optimizations brought to the *data* management layer. For instance active research on data placement techniques try to strategically manage placement of data before or during the execution of a SWf. In multisite environments, such solutions privilege starting the SWf only after gathering all the needed data in a shared-disk file system at one data center, which is time consuming. Less attention was given to the *metadata*, often considered a second class citizen in the SWf life cycle. In addition, scheduling is important for SWf execution in the multisite cloud environment. However, few attention was paid to the combination of hot metadata management strategies and different scheduling algorithms.

Centralized approaches. Metadata is usually handled by means of centralized registries implemented on top of relational databases,

that only hold static information about data locations. Systems like Pegasus [17], Swift [45] or Chiron [36] leverage such schemes, typically involving a single server that processes all the requests. In case of increased client concurrency or high I/O pressure, however, the single metadata server can quickly become a performance bottleneck. Also, the workloads involving many small files, which translate into heavy metadata accesses, are penalized by the overheads from transactions and locking [41, 42]. A lightweight alternative to databases is indexing the metadata; although most indexing techniques [44, 46] are designed for data rather than metadata. Even the dedicated index-based metadata schemes [25] use a centralized index and are not adequate for large-scale SWfs, nor can they scale to multisite deployments.

Distributed approaches. Some SWf systems opt to rely on distributed file-systems that partition the metadata and store it at each node (*e.g.* [21], [32]), in a shared-nothing architecture, as a first step towards complete geographical distribution. Hashing is the most common technique for uniform partitioning: it consists of assigning metadata to nodes based on a hash of a file identifier. Giraffa [5] uses full pathnames as key in the underlying HBase [3] store. Lustre [7] hashes the tail of the filename and the ID of the parent directory. Similar hashing schemes are used by [14, 15, 34] with a low memory footprint, granting access to data in almost constant time. FusionFS [49] implements a distributed metadata management based on DHTs as well. Chiron itself has a version with distributed control using an in-memory distributed DBMS[40]. All these systems are well suited for single-cluster deployments or SWfs that run on supercomputers. However, they are unable to meet the practical requirements of SWfs executed on clouds. Similarly to us, CalvinFS [43] uses hash-partitioned key-value metadata across geo-distributed datacenters to handle small files, yet it does not account for SWf semantics.

Hybrid approaches. More recently, Zhao *et al.* [48] proposed using both a distributed hash table (FusionFS [49]) and a centralized database (SPADE [47]) to manage the metadata. Similarly to us, their metadata model includes both file operations and provenance information. However, they do not make the distinction between hot and cold metadata, and they mainly target single site clusters.

Most of the previous work on metadata management comes from the HPC world, with solutions relying on low-latency networks for message passing and tiered cluster deployments that separate compute and storage nodes. On the other hand, cloud computing seems very different from HPC: high latency networks connect the datacenters, a much lower degree of (per-object) concurrency, a more specialized storage interface provided to applications, which are explicitly aware of the anticipated workloads and access patterns. An important difference to past work is our focus on a whole SWf application and its interaction with the cloud infrastructure. Because their target use-cases and interface semantics differ, parallel file systems cannot be used out-of-the-box in the cloud and are often considered to be mutually inappropriate. Instead, we borrow ideas from the HPC community and put them in place leveraging the SWf semantics and the cloud services publicly available.

Multisite scheduling. There are many algorithms to schedule tasks in multiple sites. In this paper, we use three existing scheduling algorithms, *i.e.* OLB (Opportunistic Load Balancing), MCT (Minimum Completion Time) and DIM (Data-Intensive Multisite task scheduling) [29]. OLB and MCT are basic algorithms, which are widely used in diverse SWf execution environments. DIM is the best in our context since it is designed with a centralized metadata management strategy, *i.e.* all the metadata is stored and managed at a single site, for SWf execution in a geographically distributed multisite cloud environment. Although we propose decentralized hot metadata management strategies, the cold metadata is always stored in a centralized database, which fits well with the assumption of DIM. OLB randomly selects a site for a task while MCT schedules a task to the site that can finish the execution first with the consideration of the time to transfer intermediate data. DIM first schedules the tasks to the site where the input data is located. Then, it manages the workload at each site in order to achieve load balancing and reduce the overall execution. In order to reduce the time to transfer intermediate data, both MCT and DIM are data location aware, *i.e.* they schedule many tasks to where the input data is. Since it may take much time to transfer intermediate data between sites, MCT has better performance than OLB. In addition, since it may also take much time to transfer metadata, DIM outperforms both MCT and OLB. In this paper, we also evaluate the performance of different hot metadata management strategies in combination with different scheduling algorithms.

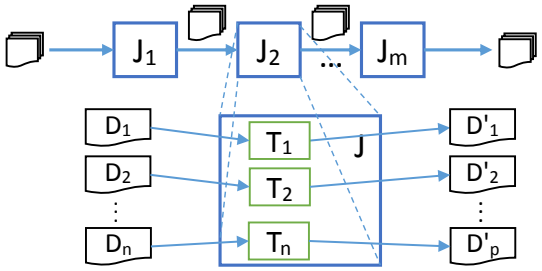


Figure 1: SWf diagram depicting the relation between *jobs* (J), *tasks* (T) and *data chunks* (D).

3 THE CORE OF OUR APPROACH: HOT METADATA

Metadata management significantly impacts the performance of computing systems dealing with thousands or millions of individual files. This is recurrently the case of SWfs. In this section, we present the SWf model, and discuss the limitations of centralized metadata management and scalability in a multisite cloud. Then, we introduce hot metadata and the challenges for hot metadata management.

3.1 SWf Model

A SWf is modeled as a graph, in which vertices represent data processing jobs and edges represent dependencies between them (Figure 1). A *job* (J) is a description of a piece of work that forms a logical step within a SWf representation. Since SWf jobs may process multiple data chunks, one job can actually correspond to several executable tasks for different parts of input data during

execution. A *task* (T) is the representation of a job within a one-time execution of this job, which processes a data chunk (D) [27], *i.e.* there is a task for each unit of data to be processed.

3.2 Why Centralized Metadata Management is an Issue?

SWfMSs handle more than file-specific metadata; running the SWf itself generates a significant amount of execution-specific metadata, *e.g.* scheduling metadata (*i.e.* which task is executed where) and data-to-task mappings. Most of today's SWfMS handle metadata in a centralized way. File-specific metadata is stored in a centralized server, either own-managed or through an underlying file system, while execution-specific metadata is normally kept in the execution's master entity.

Controlling and combining all these sorts of metadata translate into a critical workload as scientific datasets get larger. The CyberShake SWf [18], for instance, runs more than 800,000 tasks, handling an equal number of individual data pieces, processing and aggregating over 80,000 input files (which translates into 200 TB of data read), and requiring all of these files to be tracked and annotated with metadata [18, 24]. Tasks' runtime is in the order of milliseconds, *e.g.*, in a Montage SWf of 0.5 degree (see Section 7.2), out of 80 tasks, 36 tasks execute under 500 milliseconds. With many tasks, the load of parallel metadata operations becomes very heavy, and handling it in a centralized fashion represents a serious performance bottleneck.

3.3 Multisite Clouds: How to Scale?

Often enough, scientific data are so huge and widespread that they can not be processed/stored in a single cloud datacenter. On the one hand, the data size or the computing requirements might exceed the capacity of the site or the limits imposed by a cloud provider. On the other hand, data might be widely distributed, and due to their size it is more efficient to process them closer to where they reside than to bring them together; for instance, the US Earthquake Hazard Program monitors more than 7,000 sensors systems across the country reporting to the minute [11]. In either case, multisite clouds are progressively being used for executing large-scale SWfs.

Managing metadata in a centralized way for such scenarios is not appropriate. On top of the congestion generated by concurrent metadata operations, remote inter-site operations cause severe delays in the execution. To address this issue, some approaches propose the use of decentralized metadata servers [43]. In our previous work [37], we also implemented a decentralized management architecture that proved to handle metadata up to twice as fast as a centralized solution. In this paper we make one step further.

Our focus is on the metadata access frequency, particularly on identifying fractions of metadata that do not require multiple updates. The goal is to enable a more efficient decentralized metadata management, reducing the number of inter-site metadata operations by favoring the operations on frequently accessed metadata, which we call *Hot Metadata*.

3.4 What is "Hot" Metadata?

The term *hot data* refers to data that need to be frequently accessed [26]. Hot data are usually critical for the application and must be placed in a fast and easy-to-query storage [23]. We apply this concept to the context of SWf management and we define **hot metadata** as the metadata that is frequently accessed during the execution of a SWf. Conversely, less frequently accessed metadata will be denoted *cold metadata*. We distinguish two types of hot metadata: *task metadata* and *file metadata*.

Task metadata is the metadata for the execution of tasks, which is composed of the command, parameters, start time, end time, status and execution site. Hot task metadata enables the SWfMS to search and generate executable tasks. During the execution, the status and the execution site of tasks are queried many times by each site to search new tasks to execute and to determine if a job is finished. In addition, the status of a task may be updated several times. As a result, it is important to get this metadata quickly.

File metadata that we consider as "hot" for the SWf execution are those relative to the size, location and possible replicas of a given piece of data. Knowledge of file hot metadata allows the SWfMS to place the data close to the corresponding task, or vice-versa. This is especially relevant in multisite settings: timely availability of the file metadata would permit to move data *before* they are needed, hence reducing the impact of low-speed inter-site networks. In general, other metadata such as file ownership are not critical for the execution and thus regarded as cold metadata.

3.5 What are the Challenges for Hot Metadata Management?

There are a number of implications in order to effectively apply the concept of *hot metadata* to real systems. At this stage of our research, we apply simple yet efficient solutions to these challenges.

How to decide which metadata are hot? We have empirically chosen the aforementioned task and file metadata as *hot*, since they have statistically proven to be more frequently accessed by the SWfMS we use: A sample execution of 1-degree Montage SWf (Figure 2) as described in section 7.2, running 820 jobs and 57K metadata operations reveals that in a centralized execution, 32.6% of them are file metadata operations (storeFile, getFile) and 32.4% are task metadata operations (loadTask, storeTask); whereas in a distributed run, up to 67% are file operations, and task operations represent 11%. The rest correspond mostly to monitoring and node/site related operations.
However, a particular SWf might actually use other metadata more often. Since SWfs are typically defined in structured formats (*e.g.* XML files), another way to account for user-defined hot metadata would be to add a property to each job definition where the user could specify which metadata they consider as *hot*. The next item in our research agenda is to implement an environment that will allow for both user-defined and dynamically-identified hot metadata (by running training executions).

How to assess that such choice of hot metadata is right? Evaluating the efficacy of choosing hot metadata is not trivial. Metadata is much smaller than the application's data and handling it over networks with fluctuating throughput may produce inconsistent results in terms of execution time. Nevertheless, an indicator of the improvement brought by an

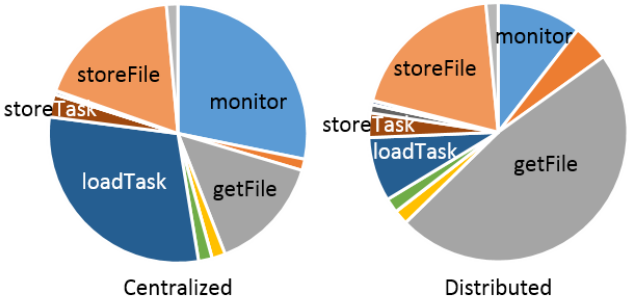


Figure 2: Relative frequency of metadata operations in Montage.

adequate choice of hot metadata, and which is not time-bounded, is the *number of metadata operations performed*. In our experimental evaluation (Section 7) we present results in terms of both execution time and number of tasks performing such operations.

The next section describes how the concept of *hot metadata* translates into architectural design choices for efficient multisite SWf processing.

4 DESIGN PRINCIPLES

Three key choices set up the foundation of our architecture:

Two-Layer Multisite SWf Management. We propose to use a two-layer multisite system: (1) The lower *intra-site* layer operates as current single-site SWfMS: a site composed of several computing nodes and a common file system, one of such nodes acts as master and coordinates communication and task execution. (2) An additional higher *inter-site* layer coordinates the interactions at site-level through a master/slave architecture (one site being the master site). The master node in each site is in charge of synchronization and data transfers. In Section 5 we provide a detailed description of such a system architecture.

Adaptive Placement for Hot Metadata. Job dependencies in a SWf form common structures (*e.g.* pipeline, data distribution and data aggregation) [13]. SWfMSs usually take into account these dependencies to schedule the job execution in a convenient way to minimize data movements (*e.g.* job co-location). Accordingly, different SWfs will yield different scheduling patterns. In order to take advantage of these scheduling optimizations, we must also adapt the SWf's metadata storage scheme. However, maintaining an updated version of all metadata across a multisite environment consumes a significant amount of communication time, incurring also monetary costs. To reduce this impact, we will evaluate different storage strategies for hot metadata during the SWf's execution, while keeping cold metadata stored locally and synchronizing such cold metadata only during the execution of the job. In the next section we recall our decentralized adaptive strategies.

Eventual Consistency for High-latency Communication. While cloud datacenters are normally interconnected by high-speed infrastructure, the latency is ultimately bounded by the physical distance between sites and communication time might reach the order of seconds [4]. Under these circumstances it is unreasonable to aim for a system with a fully consistent state in all of its components

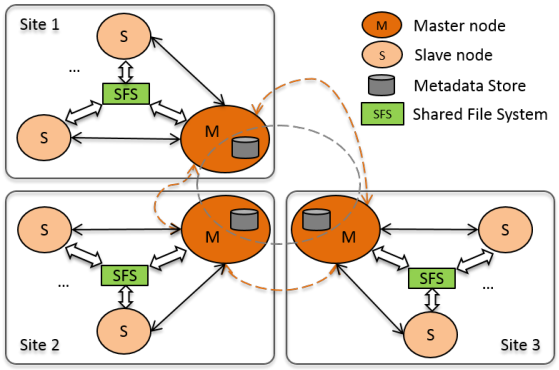


Figure 3: Multisite SWf execution architecture w/ decentralized metadata. Dotted lines represent inter-site interactions.

at a given moment without strongly compromising the performance of the application. SWf semantics allow us the flexibility to opt for an eventually consistent system: a task processes one or several specific pieces of data; such task will begin its execution only when all the pieces it needs are available in the metadata storage; however, the rest of tasks continue executing independently. Thus, with a reasonable delay due to the higher latency propagation, the system is guaranteed to be eventually consistent.

5 ARCHITECTURE

In previous work we explored different strategies for SWf-driven multisite metadata management, with a focus on file metadata [37]. Our study indicated that a hybrid approach combining decentralized metadata and replication suits better the needs of large-scale multisite SWf execution. It also showed that the right strategy to apply depends on the SWf structure. In this section, we elaborate on top of such observations into two fundamental lines. (1) We present an architecture for multisite cloud SWf processing which features decentralized metadata management. (2) We enrich this architecture with a component specifically dedicated to the management of *hot metadata* across multiple sites. (3) We adapt two metadata management strategies to hot metadata management and propose a local storage-based hot metadata management strategy.

Two-level Multisite Architecture. In accordance with our design principles, the basis for our SWf engine is a 2-level multisite architecture, as shown in Figure 3.

(1) At the inter-site level, all communication and synchronization is handled through a set of master nodes (M), one per site. One site acts as a global coordinator (master site) and is in charge of scheduling jobs/tasks to each site. Every master node holds a *metadata store* which is part of the global metadata storage and is directly accessible to all other master nodes.
(2) At the intra-site level, our system preserves the typical master/slave scheme widely-used today on single-site SWfMS: the master node schedules and coordinates a group of slave nodes that execute the SWf tasks. All nodes within a site are connected to a shared file system to access data resources. *Metadata updates* are propagated to other sites through the

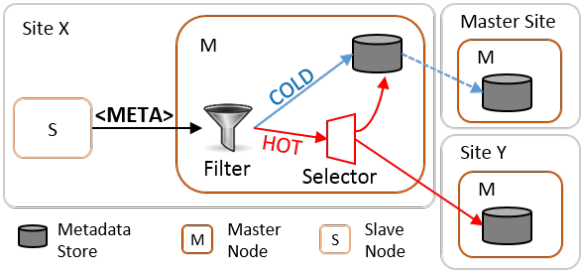


Figure 4: The hot metadata filtering component.

master node, which classifies hot and cold metadata as explained below.

Separate Management of Hot and Cold Metadata. Following our characterization of *hot metadata* from Section 3.4, we incorporate an intermediate component which filters out cold metadata operations. This model ensures that: a) hot metadata operations are managed with high priority over the network, and b) cold metadata updates are propagated only during periods of low network congestion.

The filter is located in the master node of each site (Figure 4). It separates hot and cold metadata, favoring the propagation of hot metadata and thus alleviating congestion during metadata-intensive periods. The storage location of the hot metadata is then selected based on some metadata management strategies, as developed below.

Decentralized Hot Metadata Management Strategies. We consider two different alternatives for decentralized metadata management (explored in previous work [37]). Here, we study their application to *hot* metadata and propose a local storage based hot metadata management strategy, *i.e.* LOC. They all include a metadata server in each of the datacenters where execution nodes are deployed. The two strategies differ in the way hot metadata is stored and replicated. We explain the specificities of the three strategies below.

Local without replication (LOC) Every new hot metadata entry is stored at the site where it has been created. For read operations, metadata is queried at each site and the site that stores the data will give the response. If no reply is received within a time threshold, the request is resent. This strategy will typically benefit pipeline-like SWf structures, where consecutive tasks are usually co-located at the same site.

Hashed without replication (DHT) Hot metadata is queried and updated following the principle of a distributed hash table (DHT). The site location of a metadata entry will be determined by a simple hash function applied to its key attribute, *file-name* in case of file metadata, and *task-id* for task metadata. We assume that the impact of inter-site updates will be compensated by the linear complexity of read operations.

Hashed with local replication (REP) We combine the two previous strategies by keeping both a local record of the hot metadata and a hashed copy. Intuitively, this would reduce the number of inter-site reading requests. We expect this hybrid approach to highlight the trade-offs between metadata locality and DHT linear operations.

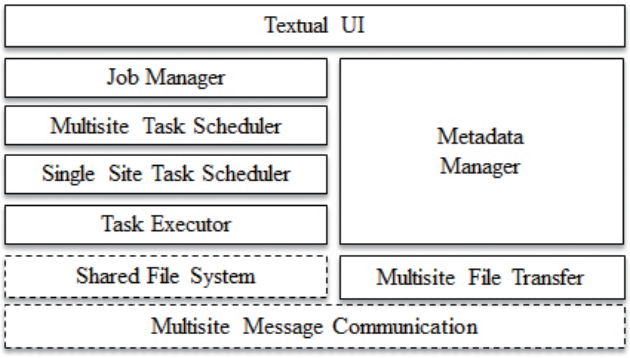


Figure 5: Layered architecture of Multisite Chiron [29].

6 IMPLEMENTATION: DMM-CHIRON

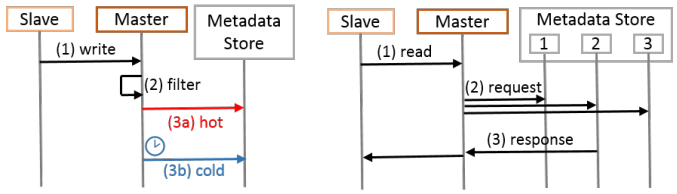
In order to validate our architecture, we developed a multisite SWfMS prototype that implements hot metadata. It provides support for *decentralized* metadata management, with a distinction between hot and cold metadata. We call our prototype Decentralized-Metadata Multisite Chiron (DMM-Chiron). In this section, we present the baseline system, *i.e.* multisite Chiron, and DMM-Chiron, including the hot metadata management strategies implementation in a multisite cloud and the techniques of provisioning hot metadata for multisite scheduling.

6.1 Baseline: Multisite Chiron

This work builds on Multisite Chiron [29], a SWfMS specifically designed for multisite clouds. Its layered architecture is presented in Figure 5; it is composed of nine modules. Multisite Chiron exploits a textual UI to interact with users. The SWf is analyzed by the *Job Manager* to identify executable activities, *i.e.* unexecuted jobs, for which the input data is ready. The same module generates the executable tasks at the beginning of job execution at the coordinator site.

Scheduling is done in two phases: the *Multisite Task Scheduler* at the coordinator site schedules each task to a site, following one of three scheduling algorithms, *i.e.* OLB, MCT and DIM. The users of Multisite Chiron can choose the multisite scheduling algorithm. While the *Single Site Task Scheduler* applies the default dynamic FAF (First job First) approach used by Chiron [36] to schedule tasks to computing nodes. Some jobs (query jobs) can be expressed as queries, which exploit the DBMS to be executed. In order to synchronize the execution, the query jobs are executed at a master site. It is worth to clarify that optimizations to the scheduling algorithms are out of the scope of this paper.

Afterwards, it is the *Task Executor* at each computing node which runs the tasks. Along the execution, metadata is handled by the *Metadata Manager* at the master site. Since the metadata structure is well defined, we use a relational database, namely PostgreSQL, to store it. All data (input, intermediate and output) are stored in a *Shared File System* at each site in order to reduce time. The location of the output data has no impact on the OLB scheduling but it has impact on the MCT or DIM scheduling. The file transfer between two different sites is performed by the *Multisite File Transfer* module. The *Multisite Message Communication* module of the master node



(a) Write (b) Read

Figure 6: Metadata Protocols.

at each site is in charge of synchronization through a master/slave architecture while the *Multisite File Transfer* module exploits a peer-to-peer model for data transfers.

6.2 Combining Multisite and Hot Metadata Management

To implement and evaluate our approach to decentralized metadata management, we further extended Multisite Chiron by adding multisite metadata protocols. We mainly modified two modules as described in the next sections: the *Job Manager* and the *Metadata Manager*.

From Single- to Multisite Job Manager. The Job Manager is the process that verifies if the execution of a job is finished, in order to launch the next jobs. Originally, this verification was done on the metadata stored at the coordinator site. In DMM-Chiron we implement an optimization to each of the hot metadata management strategies (Section 5): for LOC, the local DMM-Chiron instance verifies only the tasks scheduled at that site and the coordinator site confirms that the execution of a job is finished when all the sites finish their corresponding tasks. For DHT and REP, the master DMM-Chiron instance of the coordinator site checks each task of the job.

RDBMS & NoSQL. Since metadata has a model [35], which is equivalent to a data structure, it is convenient to store the metadata in a structured database. A data item represents a set of values corresponding to different attributes as defined in the metadata model. An RDBMS is very efficient to query structured data by comparing the value of different attributes of each data item in a structured database. Thus, we choose the PostgreSQL RDBMS to manage the hot metadata.

Introducing Protocols for Multisite Hot Metadata. The following protocols illustrate our system's *metadata operations*. We recall that metadata operations are triggered by the slave nodes at each site, which are the actual executors of the SWf tasks.

Metadata Write As shown in figure 6a, a new metadata record is passed on from the slave to the master node at each site (1). Upon reception, the master filters the record as either hot or cold (2). The hot metadata is assigned by the master node to the metadata storage pool at the corresponding site(s) according to one metadata strategy, cf. Section 5 (3a). Created cold metadata is kept locally and propagated asynchronously to the coordinator site during the execution of the job (3b).

Metadata Read Each master node has access to the entire pool of metadata stores so that it can get hot metadata from any site. Figure 6b shows the process. When a read operation is requested by a slave (1), a master node sends a request to each metadata store (2) and it processes the response that come first (3), provided such response is not an empty set. This mechanism ensures that the master node gets the required metadata in the shortest time without waiting for other late responses. During the execution, DMM-Chiron gathers all the task metadata stored at each site to verify if the execution of a job is finished.

6.3 Hot Metadata Provisioning for Multisite Scheduling

In order to efficiently schedule the tasks at different sites, the scheduling process of MCT and DIM needs to get the information about where the input data of each task is located. This information is available in the hot metadata, *i.e.* file metadata. Thus, the hot metadata is provisioned to the multisite task scheduler. Since the job manager generates tasks for job execution at the coordinator site, the file metadata is also available at the coordinator. Thus, we directly load the file metadata at the coordinator site for multisite scheduling. In addition, we use the same technique by loading the file metadata in the memory once and then use the data for the whole scheduling process in order to reduce scheduling as explained in [28].

7 EXPERIMENTAL EVALUATION

Along the following experiments we compare our results to a multisite SWfMS with centralized metadata management, which we recall being the state-of-the-art configuration. We use Multisite Chiron as an example of such architecture. In this section, we present the experimental setup and the use cases. Then, we explain the experimental results by executing real life SWfs in a multisite cloud while using different hot metadata management strategies, *i.e.* the centralized strategies & LOC, DHT and REP. Finally, we evaluate the performance of different hot metadata management strategies using different scheduling algorithms.

7.1 Experimental Setup

DMM-Chiron was deployed on the Microsoft Azure cloud [8] using a total of 27 nodes of A4 standard virtual machines (8 cores, 14 GB memory). The VMs were evenly distributed among three datacenters: West Europe (WEU, Netherlands), North Europe (NEU, Ireland) and Central US (CUS, Iowa). Control messages between master nodes are delivered through the Azure Bus [9].

7.2 Use Cases

Montage is a toolkit created by the NASA/IPAC Infrared Science Archive and used to generate custom mosaics of the sky from a set of images [19]. Additional input for the workflow includes the desired region of the sky, as well as the size of the mosaic in terms of square degrees. We model the Montage SWf using the proposal of Juve *et al.* [24].

BuzzFlow is a data-intensive SWf that searches for trends and measures correlations in scientific publications [20]. It analyses data collected from bibliography databases such as DBLP or PubMed. Buzz is composed of thirteen jobs.

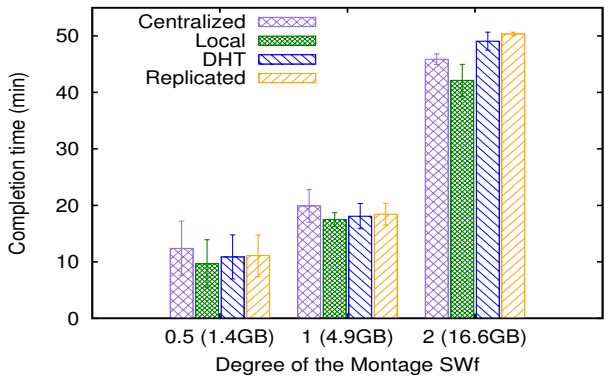


Figure 7: Montage completion time for different strategies and degrees. Avg. intermediate data shown in parenthesis.

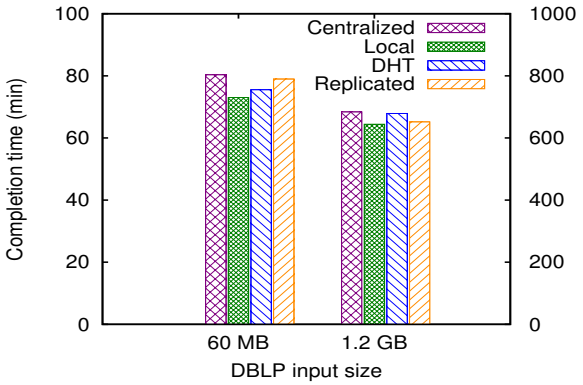


Figure 8: Buzz SWf completion time. Left Y-axis scale corresponds to 60 MB execution, right Y-axis to 1.2 GB.

7.3 Different Strategies for Different SWf Structures

Our hypothesis is that no single decentralized strategy can well fit all SWf structures: a highly parallel task would exhibit different metadata access patterns than a concurrent data gathering task. Thus, the improvements brought to one type of SWf by either of the strategies might turn to be detrimental for another. To evaluate this hypothesis, we ran several combinations of our strategies with the featured SWfs and the OLB scheduling algorithm.

Figure 7 shows the average completion time for the Montage SWf generating 0.5-, 1-, and 2-degree mosaics of the sky, using in all the cases a 5.5 GB image database distributed across the three datacenters. With a larger degree, a larger volume of intermediate data is handled and a mosaic of higher resolution is produced. Table 1 summarizes the volumes of intermediate data generated per execution.

	0.5-degree	1-degree	2-degree
CEN	1.4 (0.5, 0.5, 0.4)	4.9 (1.7, 1.5, 1.7)	17.1 (6.0, 6.4, 4.7)
LOC	1.3 (0.7, 0.2, 0.4)	4.8 (2.1, 1.0, 1.7)	16.2 (8.4, 4.6, 3.2)
DHT	1.5 (0.6, 0.6, 0.4)	4.9 (1.9, 1.3, 1.8)	16.6 (5.4, 4.9, 6.2)
REP	1.4 (0.5, 0.5, 0.4)	4.9 (1.5, 1.9, 1.5)	16.8 (6.6, 3.8, 6.4)

Table 1: Intermediate data in GB for different degrees of Montage executions. Per-site breakdown is expressed as: Aggregated (size WEU, size NEU, size CUS).

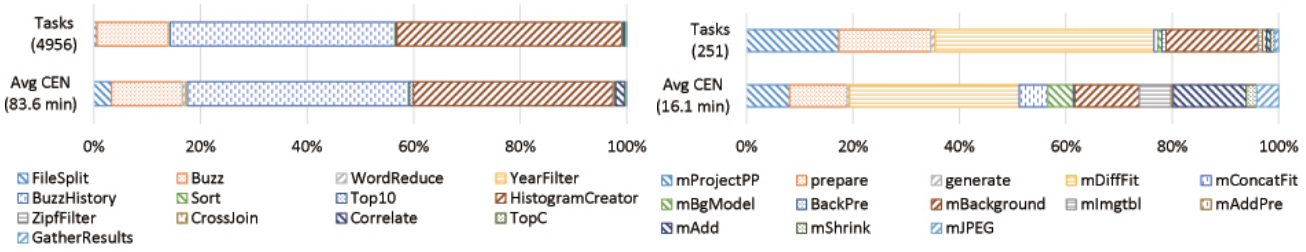
In the chart we note in the first place a clear time gain of up to 28% by using a local distribution strategy instead of a centralized one, for all the degrees. This result was expected since the hot metadata is now managed in parallel by three instances instead of one, and it is only the cold metadata that is forwarded to the coordinator site for scheduling purposes (and used at most one time).

We observe that for mosaics of degree 1 and under, the use of distributed hashed storage also outperforms the centralized version. Since the metadata is distributed at different sites when using DHT, the read operations are also processed at different sites. This distribution can achieve better load balancing for the read operations, which is faster compared with the centralized method [37]. However, we note a performance degradation in the hashed strategies, starting at 1-degree and getting more evident at 2-degree. We attribute this to the fact that there is a larger number of long-distance hot metadata operations compared to the centralized approach: with hashed strategies, 1 out of 3 operations are carried out on average between CUS and NEU. In the centralized approach, NEU only performs operations in the WEU site, thus such long latency operations are reduced. We also associate this performance drop with the size of intermediate data being handled by the system: while we try to minimize inter-site data transfers, with larger volumes of data such transfers affect the completion time up to a certain degree and independently of the metadata management scheme. We conclude that while the DHT method might seem efficient due to linear read and write operations, it is not well suited for geo-distributed executions, which favor locality and penalize remote operations.

In a similar experiment, we validated DMM-Chiron using the Buzz SWf, which is rather data intensive, with two DBLP database dumps of 60 MB and 1.2 GB. The results are shown in Figure 8; note that the left and right Y-axes differ by one order of magnitude. This is why it seems that the maximum gain is closer to a constant gain without considering the different scale of axes. But the real maximum gain is different and increases according to the overall execution time of SWfs. We observe again that DMM-Chiron brings a general improvement in the completion time with respect to the centralized implementation: 10% for LOC in the 60 MB dataset and 6% for 1.2 GB, while for DHT and REP the time improvement was of less than 5%.

In order to better understand the performance improvements brought by DMM-Chiron, and also to identify the reason of the low runtime gain for the Buzz SWf, we evaluated Montage and Buzz in a per-job granularity. The results are presented in the next section. Albeit the time gains perceived in the experiments might not seem significant at first glance, two important aspects must be taken into consideration:

Optimization at no cost Our proposed solutions are implemented using exactly the same number of resources as their counterpart centralized approaches: the decentralized metadata



(a) Buzz (b) Montage

Figure 9: SWf per-job breakdown. Very small jobs are enhanced for visibility.

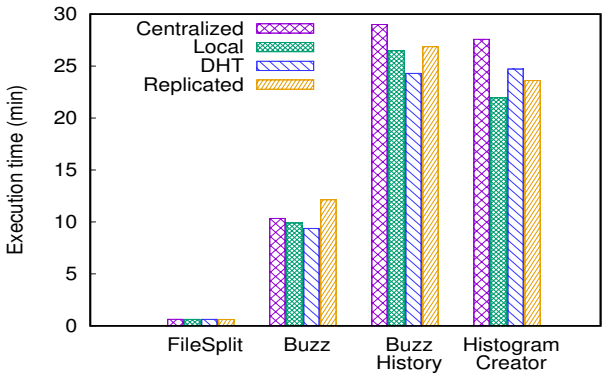


Figure 10: Execution time of multi-task jobs on the Buzz SWf with 60 MB input data.

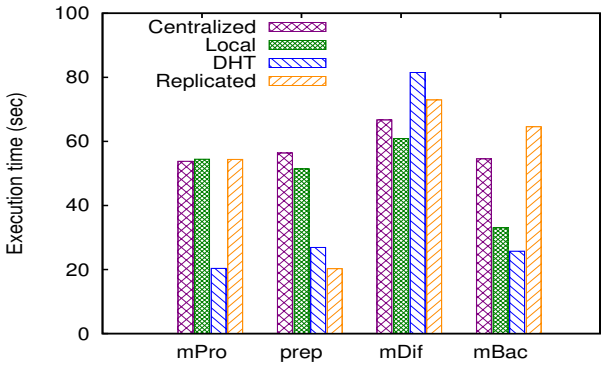


Figure 11: Execution time of multi-task jobs on the Montage SWf of 0.5 degree.

stores are deployed within the master nodes of each site and the control messages are sent through the same existing channels. This means that such gains come at no additional cost for the user.

Actual monetary savings Our longest experiment (Buzz 1.2 GB) runs in the order of hundreds of minutes. With today's scientific experiments running at this scale and beyond, a gain of 10% actually implies savings of hours of cloud computing resources.

7.4 Zoom on Multi-task Jobs

We call a job *multi-task* when its execution consists of more than a single task. In DMM-Chiron, the various tasks of such jobs are evenly distributed to the available sites and thus can be executed in parallel. We argue that it is precisely in these kind of jobs that DMM-Chiron yields its best performance.

Figure 9 shows a breakdown of Buzz and Montage SWfs with the proportional size of each of their jobs from two different perspectives: tasks count and average execution time. Our goal is to characterize the most *relevant* jobs in each SWf by number of tasks and confirm their relevance by looking at their relative execution time. In Buzz, we notice that both metrics are highly dominated by three jobs: Buzz (676 tasks), BuzzHistory (2134) and HistogramCreator (2134), while the rest are so small that they are barely noticeable. FileSplit comes fourth in terms of execution time and it is indeed the only remaining multi-task job (3 tasks). Likewise, we identify for Montage the only four multi-task jobs: mProject (45 tasks), prepare (45), mDiff (107) and mBackground (45).

In Figures 10 and 11 we look into the execution time of the multi-task jobs of Buzz and Montage, respectively. In Figure 10, we observe that except for one case, namely Buzz job with REP, the decentralized strategies outperform considerably the baseline (up to 20.3% for LOC, 16.2% for DHT and 14.4% for REP). In the case of FileSplit, we argue that the execution time is too short and the number of tasks too small to reveal a clear improvement. However, the other three jobs confirm that DMM-Chiron performs better for highly parallel jobs. It is important to note that these gains are much larger than those of the overall completion time (Figure 8) since there are still a number of workloads executed sequentially, which have not been optimized by the current release of DMM-Chiron.

Correspondingly, Figure 11 shows the execution of each multi-task job for the Montage SWf of 0.5 degree. The figure reveals that, on average, hot metadata distribution substantially improves centralized management in most cases (up to 39.5% for LOC, 52.8% for DHT and 64.1% for REP). However, we notice some unexpected peaks and drops specifically in the hashed approaches. After a number of executions, we believe that such cases are due to common network latency variations of the cloud environment added to the fact that the execution time for the jobs is rather short (in the order of seconds).

7.5 Different Strategies for Different scheduling algorithms

There is no single decentralized strategy that can fit all scheduling algorithms: some scheduling algorithms may be optimized for a certain metadata management strategy. Thus, the improvements

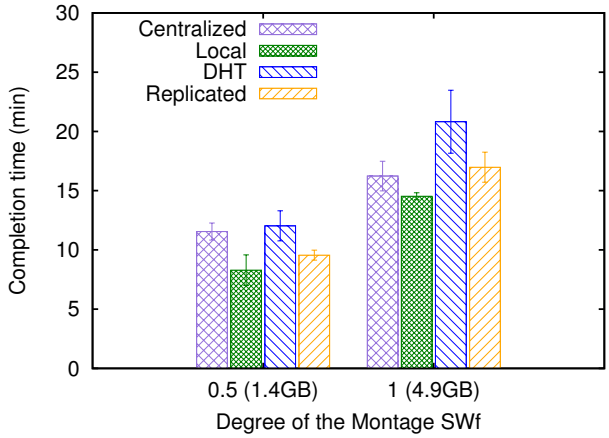


Figure 12: The completion time of the Montage SWf with MCT for different strategies and degrees. Avg. intermediate data shown in parenthesis.

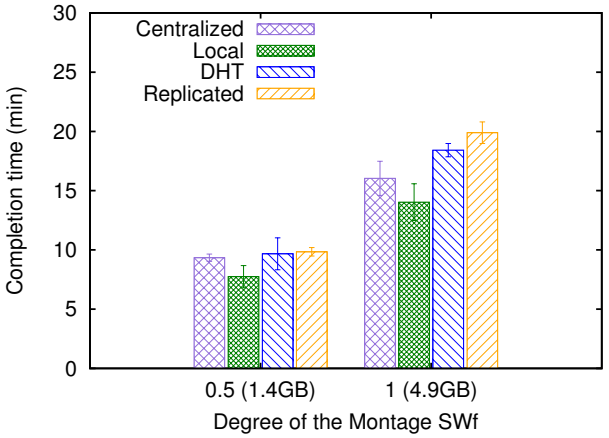


Figure 13: The completion time of the Montage SWf with DIM for different strategies and degrees. Avg. intermediate data shown in parenthesis.

brought to one scheduling algorithm by either of the strategies might turn to be not so good or even detrimental for another. To better understand this relationship between strategy and scheduling algorithm, we ran several combinations of our strategies with Montage and two other scheduling algorithms, *i.e.* MCT and DIM. We use the same configuration except for the scheduling algorithm as explained in Section 7.3.

Figures 12 and 13 show that LOC is always better (up to 28.2%) than the centralized strategy in terms of overall SWf completion time with MCT and DIM, which perform better than OLB. However, the centralized strategy outperforms DHT and REP when using MCT and DIM scheduling algorithms. DMM-Chiron may gather the data to the master site (WEU) for some jobs (query jobs), which exploit the RDBMS to process the data. The input data of the following jobs of the query is centralized at the master site. In addition, MCT and DIM are data location aware and schedule most of the tasks where the input data is. In this case, the centralized strategy stores the hot metadata at the master site, which is read many times by the same site during execution, while DHT and REP distribute the hot metadata to other sites. As a result, the centralized strategy outperforms DHT and REP. However, in this case, similar to the centralized strategy, the LOC strategy always stores the metadata at the site where the metadata is produced, *i.e.* the master site. As a result, LOC can always outperform the centralized strategy in terms of overall SWf completion time. Figures 14 and 15 show that LOC is generally better (up to 31.1% for MCT and 33.7% for DIM) than the centralized strategy while DHT and REP may be worse than the centralized strategy in terms of execution time of multi-task jobs. In addition, the combination of LOC and DIM can reduce the overall SWf execution time up to 37.5%, which is much better than the best result (28%) presented in [38], of the execution time of Montage compared with the combination of the centralized strategy and OLB.

Since DIM is optimized for the centralized metadata management strategy, it is reasonable that the centralized strategy outperforms DHT and REP and that the gain brought by LOC is less obvious compared with MCT. The LOC strategy only stores the hot metadata at the site where it is produced. The hot metadata is read many times by the same site during the execution of the tasks. Thus, LOC can always reduce the time to manage metadata. Thus, LOC always outperforms the centralized strategy even when the DIM scheduling algorithm is used.

8 DISCUSSION

In this section, we analyze DMM-Chiron from three perspectives. First, we address the question of whether *hot metadata* management is actually necessary. Then, we discuss the issues about dynamic hot metadata. Afterwards, we consider some technical limitations of our system.

8.1 Is "regular" metadata management really a problem?

Along this paper, we do not argue that handling metadata as a whole (*i.e.* without distinction between hot and cold) is inefficient. Actually, as presented in Section 2, many of today's file systems do not make any distinction when processing metadata, and some have very good performance. However, such systems are rather general purpose and deployed in a single site. We proved that in the context of multisite SWfs, where thousands of tasks are executed across datacenters, separating hot metadata improves the SWf execution time, yet at no additional cost.

8.2 Towards Dynamic Hot Metadata

Scientific applications evolve during execution. This means that at a given point, some data might no longer be as relevant as they were initially; in other words, hot data become cold, or vice-versa. In the case of hot to cold data, SWfMSs might remove them from the fast-access storage or even delete them; conversely, data that become relevant can be promoted to fast storage. Some frameworks assess the data "temperature" *offline*, *i.e.* they perform a later analysis on a frequency-of-access log to avoid overhead during the operation [26], however, this method is only useful when there are subsequent runs. More interestingly for us, *online* approaches maintain a rank

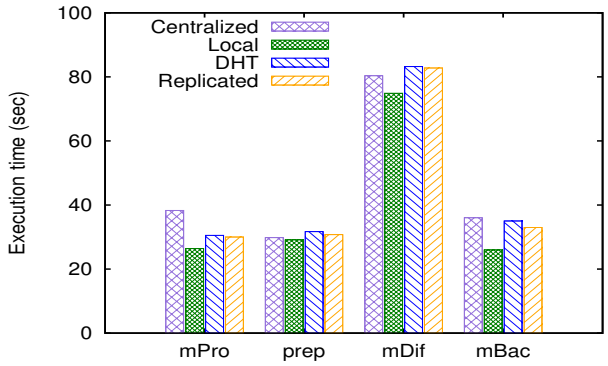


Figure 14: Execution time of multi-task jobs on the Montage SWf of 0.5 degree with MCT.

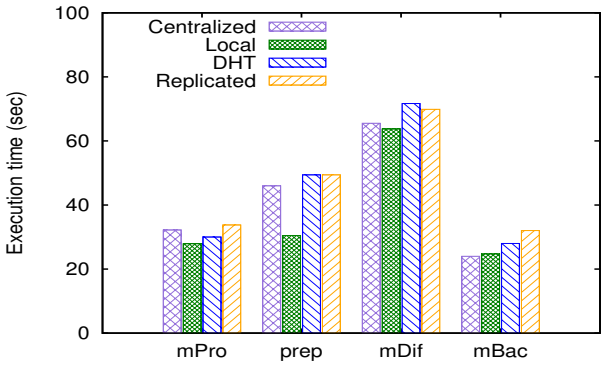


Figure 15: Execution time of multi-task jobs on the Montage SWf of 0.5 degree with DIM.

on the frequency of access to the data alongside the execution, for example in adaptive replacement cache [33]. This phenomenon certainly occurs also at the metadata level; so, how could we handle these "temperature" changes in a multisite SWfMS? We look into the two situations.

Promoting Cold to Hot Metadata. User-defined hot metadata as discussed above would not allow metadata to be dynamically promoted, since an XML SWf definition file is rather static. However, we can build on this idea and integrate the described online ranking: given a SWf defined through an XML file (or any other definition language), a list of metadata attributes could be passed to the execution engine in the same file; then, the engine would monitor the access frequency of each of such attributes and periodically produce a ranking to verify whether an attribute is required more often, and thus *promote* it to hot metadata. The maximum number of attributes allowed as *hot* metadata could be also dynamically calculated according to the aggregated size of the metadata stores.

Downgrading Hot to Cold Metadata. Less frequently accessed metadata could also be identified using the same attribute ranking approach as above. Upon identification, degrading some metadata to *cold* would also require a collection mechanism that ensures that metadata previously considered hot are removed from fast-access storage. Moreover, this action should take place during not-congested periods, or at the end of the execution so that it does not incur overhead. Taking one step further, we can consider an algorithm that determines the probability that metadata could become hot again later in the execution based on historical data; such metadata could be left in the storage, preventing I/O congestion.

To avoid interfering with the SWf scheduling and execution processes, these scenarios should ideally be implemented transparently within the metadata storage system. We consider them as a potential extension to our work, but we do not implement them.

8.3 Limitations

Our focus in this paper was on handling metadata in a smart distributed way so that this improves job/task execution time when processing a large number of data pieces. While our techniques show an improvement with respect to a centralized management, we also notice that when the scale of the SWf and the size of data become larger, there is a degradation in the performance of DMM-Chiron when using OLB (see Figure 7) due to the increase of intermediate data transfers. To mitigate this degradation, the in-memory caching techniques can be used in order to reduce the time to read or write data. In addition, we can also use DIM to reduce inter-site data transfer. The gain of LOC is even more obvious when we process bigger data set using DIM (see Figure 13) compared with the centralized method.

In addition, we have only considered the case of a homogeneous multisite environment, where each site has the same amount of VMs of the same type. While this configuration is often utilized, in several cases multisite clouds are heterogeneous (different number of resources, different VMs sizes). A next step in this path would be to account for these variations in order to balance the hot metadata load according to the site's computing capacity.

9 CONCLUSION

In this paper, we introduced the concept of hot metadata for SWfs running in large, geographically distributed and highly dynamic environments. Based on it, we designed a hybrid decentralized and distributed model for handling metadata in multisite clouds. The model includes a distributed architecture and two adapted and one proposed hot metadata management strategies. The two adapted strategies are DHT and REP and our proposed strategy is LOC, which stores the hot metadata at the site where it is generated. We choose a RDBMS to manage the metadata and coupled the architectures and hot metadata management strategies with a SWf engine. We also enable three scheduling algorithms, *i.e.* OLB, MCT and DIM, to perform multisite scheduling by provisioning hot metadata. Our proposal is able to optimize the access to and ensure the availability of hot metadata, while effectively hiding the inter-site network latencies and remaining non-intrusive and easy to deploy. We validated metadata management strategies by executing real life SWfs with different scheduling algorithms in a multisite cloud. Our experimental results showed an improvement of up to 37.5% for the whole SWf's execution time and 64.1% for specific highly-parallel jobs, compared to state-of-the-art centralized and OLB solutions, at no additional cost. In addition, our experiments show that, although

no single decentralized strategy can well fit all SWf structure, our proposed hot metadata strategy, *i.e.* LOC, always outperforms other strategies with different SWf structures and scheduling algorithms in terms of overall SWf execution time.

Encouraged by these results, we plan to broaden the scope of our work and consider the impact of heterogeneous multisite environments on the hot metadata strategies. Another interesting direction to explore is integrating real-time monitoring information about the executed jobs in order to dynamically balance the hot metadata load according to each site's live capacity and performance.

ACKNOWLEDGMENT

This work is supported by the MSR - Inria Joint Centre, the ANR OverFlow project and partially performed in the context of the Computational Biology Institute. The experiments were carried out using the Azure infrastructure provided by Microsoft in the framework of the Z-CloudFlow project. Luis is partially funded by CONACyT, Mexico. Ji is partially funded by EU H2020 Programme, MCTI/RNP-Brazil, CNPq, FAPERJ, and Inria (MUSIC project).

REFERENCES

[1] Alice Collaboration. http://aliceinfo.cern.ch/general/index.html.
[2] Amazon Elastic Compute Cloud. https://aws.amazon.com/ec2/.
[3] Apache HBase. http://hbase.apache.org.
[4] Azure Speed Test. http://www.azurespeed.com/.
[5] Giraffa. https://code.google.com/a/apache-extras.org/p/giraffa/.
[6] Google Cloud Platform. https://cloud.google.com/.
[7] Lustre - OpenSFS. http://lustre.org/.
[8] Microsoft Azure Cloud. http://www.windowsazure.com/en-us/.
[9] Microsoft Azure Service Bus. https://azure.microsoft.com/en-us/services/service-bus/.
[10] Resource Quotas - Google Cloud Platform. https://cloud.google.com/compute/docs/resource-quotas.
[11] USGS ANSS - Advanced National Seismic System. http://earthquake.usgs.gov/monitoring/anss/.
[12] Sadaf R. Alam, Hussein N. El-Harake, Kristopher Howard, Neil Stringfellow, and Fabio Verzelloni. Parallel I/O and the metadata wall. In *Proc. of the 6th Workshop on Parallel Data Storage*, PDSW '11, pages 13–18, New York, NY, USA, 2011. ACM.
[13] Shishir Bharathi, Ann Chervenak, Ewa Deelman, Gaurang Mehta, Mei-Hui Su, and Karan Vahi. Characterization of scientific workflows. In *Workshop on WFs in Support of Large-Scale Science*, 2008.
[14] S. A. Brandt, E. L. Miller, D. D. E. Long, and Lan Xue. Efficient metadata management in large distributed storage systems. In *Proc. 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies*, 2003.
[15] Peter F. Corbett and Dror G. Feitelson. The vesta parallel file system. *ACM Trans. Comput. Syst.*, 14(3):225–264, August 1996.
[16] E. Deelman, D. Gannon, M. Shields, and I. Taylor. Workflows and e-science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, 25(5):528–540, 2009.
[17] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3):219–237, 2005.
[18] Ewa Deelman, Scott Callaghan, Edward Field, Hunter Francoeur, Robert Graves, Nitin Gupta, Vipin Gupta, Thomas H. Jordan, Carl Kesselman, Philip Maechling, John Mehringer, Gaurang Mehta, David Okaya, Karan Vahi, and Li Zhao. Managing large-scale workflow execution from resource provisioning to provenance tracking: The cybershake example. In *IEEE Intl. Conf. on e-Science and Grid Computing*, 2006.
[19] Ewa Deelman, Gurmeet Singh, Miron Livny, Bruce Berriman, and John Good. The cost of doing science on the cloud: The montage example. In *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, SC '08, pages 50:1–50:12, 2008.
[20] J. Dias, E. Ogasawara, D. De Oliveira, F. Porto, P. Valduriez, and M. Mattoso. Algebraic dataflows for big data analysis. In *Big Data, 2013 IEEE Intl. Conf. on*, pages 150–155, 2013.
[21] A. Gehani, M. Kim, and T. Malik. Efficient querying of distributed provenance stores. In *ACM Int. Symposium on High Performance Distributed Computing HPDC*, pages 613–621, 2010.
[22] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. *SIGOPS Oper. Syst. Rev.*, 37(5):29–43, October 2003.
[23] Dan Gibson. Is Your Big Data Hot, Warm, or Cold?, 2012.
[24] Gideon Juve, Ann Chervenak, Ewa Deelman, Shishir Bharathi, Gaurang Mehta, and Karan Vahi. Characterizing and profiling scientific workflows. *FGCS*, 29(3):682 – 692, 2013.
[25] Andrew W Leung, Minglong Shao, Timothy Bisson, Shankar Pasupathy, and Ethan L Miller. Spyglass: Fast, scalable metadata search for large-scale storage systems. In *FAST*, volume 9, pages 153–166, 2009.
[26] Justin J Levandoski, P-A Larson, and Radu Stoica. Identifying hot and cold data in main-memory databases. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 26–37. IEEE, 2013.
[27] J. Liu, E. Pacitti, P. Valduriez, and M. Mattoso. A survey of data-intensive scientific workflow management. *Journal of Grid Computing*, 2015.
[28] J. Liu, E. Pacitti, P. Valduriez, and M. Mattoso. Scientific workflow scheduling with provenance data in a multisite cloud. *Transactions on Large-Scale Data- and Knowledge-Centered Systems*, 2016. to appear.
[29] J. Liu, E. Pacitti, P. Valduriez, and M. Mattoso. Scientific workflow scheduling with provenance support in multisite cloud. In *High Performance Computing for Computational Science VECPAR*, 2016.
[30] J. Liu, E. Pacitti, P. Valduriez, D. De Oliveira, and M. Mattoso. Multi-objective scheduling of scientific workflows in multisite clouds. *Future Generation Computer Systems*, 63:76–95, 2016.
[31] J. Liu, V. S. Sousa, E. Pacitti, P. Valduriez, and M. Mattoso. Scientific workflow partitioning in multisite cloud. In *Euro-Par 2014: Parallel Processing Workshops - Euro-Par 2014 Int. Workshops*, pages 105–116, 2014.
[32] T. Malik, L. Nistor, and A. Gehani. Tracking and sketching distributed data provenance. In *Sixth Int. Conf. on e-Science*, pages 190–197, 2010.
[33] Nimrod Megiddo and Dharmendra S Modha. Arc: A self-tuning, low overhead replacement cache. In *FAST - USENIX Conference on File and Storage Technologies*, volume 3, pages 115–130, 2003.
[34] Ethan L. Miller and Randy H. Katz. RAMA: An easy-to-use, high-performance parallel file system. *Parallel Computing*, 23(4):419–446.
[35] E. S. Ogasawara, J. Dias, V. Silva, F. S. Chirigati, D. de Oliveira, F. Porto, P. Valduriez, and M. Mattoso. Chiron: a parallel engine for algebraic scientific workflows. *Concurrency and Computation: Practice and Experience*, 25(16):2327–2341, 2013.
[36] Eduardo Ogasawara, Jonas Dias, Fabio Porto, Patrick Valduriez, and Marta Mattoso. An algebraic approach for data-centric scientific workflows. *Proc. of VLDB Endowment*, 4(12):1328–1339, 2011.
[37] L. Pineda-Morales, A. Costan, and G. Antoniu. Towards multi-site metadata management for geographically distributed cloud workflows. In *IEEE Intl. Conf. on Cluster Computing*, pages 294–303, Sept 2015.
[38] L. Pineda-Morales, J. Liu, A. Costan, E. Pacitti, G. Antoniu, P. Valduriez, and M. Mattoso. Managing hot metadata for scientific workflows on multisite clouds. In *IEEE Int. Conf. on Big Data*, pages 390–397, 2016.
[39] Frank Schmuck and Roger Haskin. GPFS: A shared-disk file system for large computing clusters. In *Proc. of the 1st USENIX Conference on File and Storage Technologies*, FAST '02, Berkeley, CA, USA, 2002.
[40] R. Souza, V. Silva, Daniel Oliveira, Patrick Valduriez, A.A.B. Lima, and Marta Mattoso. Parallel execution of workflows driven by a distributed database management system. In *ACM/IEEE Conference on Supercomputing, Poster*, 2015.
[41] M. Stonebraker and U. Cetintemel. "one size fits all": an idea whose time has come and gone. In *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, pages 2–11, April 2005.
[42] Michael Stonebraker, Samuel Madden, Daniel J. Abadi, and Nabil Hachem. The end of an architectural era: Time for a complete rewrite. In *Proc. of the 33rd Intl. Conf. on Very Large Data Bases*, VLDB '07, pages 1150–1160.
[43] Alexander Thomson and Daniel J Abadi. CalvinFS: consistent wan replication and scalable metadata management for distributed file systems. In *Proc. of the 13th USENIX Conf. on File and Storage Technologies*, 2015.
[44] Jinbao Wang, Sai Wu, Hong Gao, Jianzhong Li, and Beng Chin Ooi. Indexing multi-dimensional data in a cloud system. In *Proc. of the 2010 ACM SIGMOD Intl. Conf. on Management of Data*, pages 591–602, 2010.
[45] J. M. Wozniak, T. G. Armstrong, M. Wilde, D. S. Katz, E. L. Lusk, and I. T. Foster. Swift/t: scalable data flow programming for many-task applications. In *ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 309–310, 2013.
[46] Sai Wu, Dawei Jiang, Beng Chin Ooi, and Kun-Lung Wu. Efficient b-tree based indexing for cloud data processing. *Proc. VLDB Endow.*, 3(1-2):1207–1218, 2010.
[47] Mohammed J. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1):31–60.
[48] D. Zhao, C. Shou, T. Maliky, and I. Raicu. Distributed data provenance for large-scale data-intensive computing. In *CLUSTER*, pages 1–8, 2013.
[49] D. Zhao, Z. Zhang, X. Zhou, T. Li, K. Wang, D. Kimpe, P. Carns, R. Ross, and I. Raicu. Fusionfs: Toward supporting data-intensive scientific applications on extreme-scale high-performance computing systems. In *2014 IEEE Intl. Conf. on Big Data*, Oct 2014.