Managing Hot Metadata for Scientific Workflows on Multisite Clouds

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

Managing Hot Metadata for Scientific Workflows on Multisite Clouds

Luis Pineda-Morales*, Ji Liu*, Alexandru Costan†, Esther Pacitti‡,
Gabriel Antoniu§, Patrick Valduriez§, and Marta Mattoso¶

*Microsoft Research - Inria Joint Centre, France
{luis.pineda-morales, ji.liu}@inria.fr

§Inria, France
{gabriel.antoniu, patrick.valduriez}@inria.fr

†IRISA / INSA Rennes, France
alexandru.costan@irisa.fr

‡LIRMM, France
esther.pacitti@lirmm.fr

¶COPPE / UFRJ, Brazil
marta@cos.ufrj.br

***Abstract*—Large-scale scientific applications are often expressed as workflows that help defining data dependencies between their different components. Several such workflows 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 this paper, we present a hybrid decentralized/distributed model for handling *hot metadata* (frequently accessed metadata) in *multisite* architectures. We couple our model with a scientific workflow management system (SWfMS) to validate and tune its applicability to different real-life scientific scenarios. We show that efficient management of hot metadata improves the performance of SWfMS, reducing the workflow execution time up to 50% for highly parallel jobs and avoiding unnecessary *cold* metadata operations.**

***Index Terms*—hot metadata, metadata management, multisite clouds, scientific workflows, geo-distributed applications.**

I. 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 [1], 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 [2]). 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 [3], Amazon EC2 [4], and Google Cloud [5].

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 [6]. 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 [7] and Taverna [8]; some of them already support multisite execution [9].

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 workflow'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. While it has been proven that efficient metadata handling plays a key role in performance [10], [11], 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 III, 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 [7], [12], [13]. 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.

This paper presents the following contributions:

- Based on the notion of *hot metadata*, we introduce an architecture for optimizing the access and ensuring the

availability of hot metadata in a multisite cloud environment (Section IV).
- We develop a prototype by coupling our proposed scheme with a state of the art multisite workflow execution engine, namely Chiron [14] (Section V).
- We demonstrate that efficient management of hot metadata improves the performance of SWfMS, reducing the execution time of a workflow by 1) enabling timely data provisioning and 2) avoiding unnecessary *cold* metadata handling (Section VI).

II. 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.

A. Why Centralized Metadata Management is an Issue?

Workflow management systems handle more than file-specific metadata; running the workflow 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 workflow, 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 [15], [16]. With many tasks' runtime in the order of milliseconds, the load of parallel metadata operations becomes very heavy, and handling it in a centralized fashion represents a serious performance bottleneck.

B. 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 [17]. In either case, multisite clouds are progressively being used for executing large-scale scientific workflows.

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 [10]. In our previous work [18], 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*.

C. What is "Hot" Metadata?

The term *hot data* refers to data that need to be frequently accessed [19]. Hot data are usually critical for the application and must be placed in a fast and easy-to-query storage [20]. 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 workflow. 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 job metadata enables the SWfMS to search and generate executable tasks. During the execution, the status and the execution site of the 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 at each site.

File metadata that we consider as "hot" for a workflow 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 or permissions are not critical for the execution and thus regarded as cold metadata.

D. 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 Workflow (Fig. 1) as described in section VI-B, 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

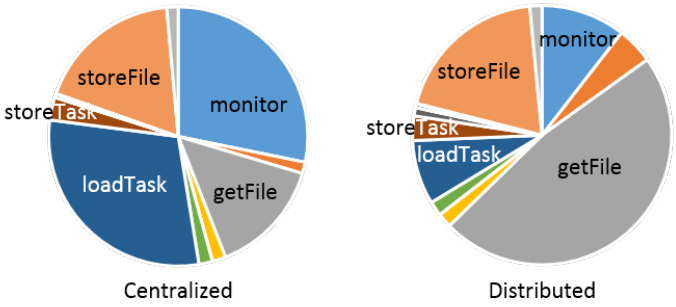


Fig. 1: Relative frequency of metadata operations in Montage.

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 workflows 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 adequate choice of hot metadata, and which is not time-bounded, is the *number of metadata operations performed*. In our experimental evaluation (Section VI) 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 workflow processing.

III. Design Principles

Three key choices set up the foundation of our architecture:

Two-Layer Multisite Workflow 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 IV we provide a detailed description of such a system architecture.

Adaptive Placement for Hot Metadata. Job dependencies in a workflow form common structures (*e.g.* pipeline, data distribution and data aggregation) [21]. SWfMS 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 workflows will yield different scheduling patterns. In order to take advantage of these scheduling optimizations, we must also adapt the workflow'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 workflow'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 [22]. Under these circumstances it is unreasonable to aim for a system with a fully consistent state in all of its components at a given moment without strongly compromising the performance of the application. Workflow semantics allow us the flexibility to opt for an eventually consistent system: a workflow execution unit (task) processes one or several specific pieces of data; such unit will begin its execution only when all the pieces it needs are available in the metadata storage; however, the rest of units continue executing independently. Thus, with a reasonable delay due to the higher latency propagation, the system is guaranteed to be eventually consistent.

IV. Architecture

In previous work we explored different strategies for workflow-driven multisite metadata management, with a focus on file metadata [18]. Our study indicated that a hybrid approach combining decentralized metadata and replication suits better the needs of large-scale multisite workflow execution. It also showed that the right strategy to apply depends on the workflow structure. In this section, we elaborate on top of such observations into two fundamental lines. (1) We present an architecture for multisite cloud workflow 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.

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

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.

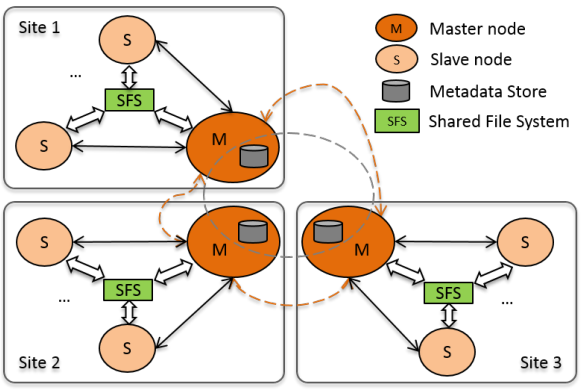


Fig. 2: Multisite SWf execution architecture w/ decentralized metadata. Dotted lines represent inter-site interactions.

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 which execute the workflow 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 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 II-C, 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 3). It separates hot and cold metadata, favoring the propagation of hot metadata and thus alleviates congestion during metadata-intensive periods. The storage location of the hot metadata is then selected based on some metadata management strategy, as developed below.

Decentralized Hot Metadata Management Strategies. We consider three different alternatives for decentralized metadata management (explored in previous work [18]). Here, we study their application to *hot* metadata. They all include a metadata server in each of the datacenters where execution nodes are deployed. They differ in the way hot metadata is stored and replicated. We briefly recall their specificities 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 workflow 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.

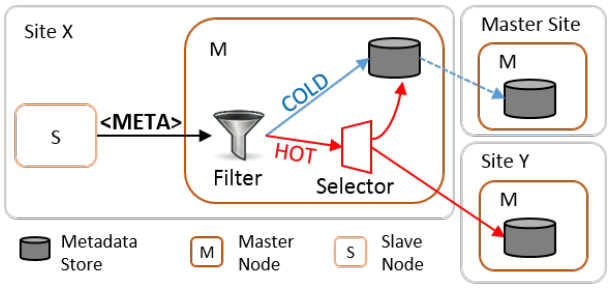


Fig. 3: The hot metadata filtering component.

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.

V. Implementation: DMM-Chiron

In order to validate our architecture, we have developed a prototype multisite SWfMS that implements hot metadata handling. It provides support for *decentralized* metadata management, with a distinction between hot and cold metadata. We denote our prototype by Decentralized-Metadata Multisite Chiron (DMM-Chiron).

A. Baseline: Multisite Chiron

This work builds on Multisite Chiron [9], a SWfMS specifically designed for multisite clouds. Its layered architecture is presented in Figure 4; 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.

Scheduling is done in two phases: the *Multisite Task Scheduler* at the coordinator site schedules each task to a site, following the random OLB (Opportunistic Load Balancing) algorithm used in [9]. While the *Single Site Task Scheduler* applies the default dynamic FAF (First Activity First) approach used by Chiron [14] to schedule tasks to computing nodes. 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. The file transfer between two different sites is performed by the *Multisite File Transfer* module. The *Multisite Message Communication* module of the master node at each site is in

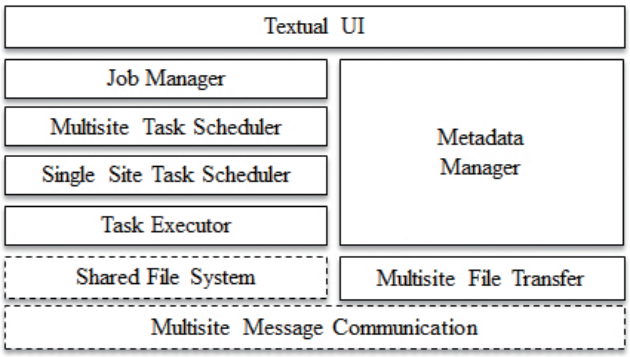


Fig. 4: Layered architecture of Multisite Chiron [9].

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

B. Combining Multisite and Hot Metadata Management

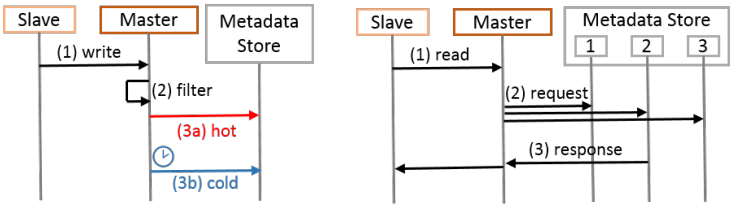
To implement and evaluate our approach to decentralized metadata management, we further extended Multisite Chiron by adding multisite metadata protocols. We mainly modified two modules as described in the next sections: the *Job Manager* and the *Metadata Manager*.

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 IV): 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.

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 workflow tasks.

Metadata Write As shown in figure 5a, 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 IV (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 5b 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. During the execution, DMM-Chiron gathers all the task metadata stored at each site to verify if the execution of a job is finished.



(a) Write (b) Read

Fig. 5: Metadata Protocols.

VI. 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.

A. Experimental Setup

DMM-Chiron was deployed on the Microsoft Azure cloud [3] 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 [23].

B. 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 [24]. 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. [15].

BuzzFlow is a data-intensive SWf that searches for trends and measures correlations in scientific publications [25]. It analyses data collected from bibliography databases such as DBLP or PubMed. Buzz is composed of thirteen jobs.

C. Different Strategies for Different Workflow Structures

Our hypothesis is that no single decentralized strategy can fit all workflow 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 workflow 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 workflows.

Figure 6 shows the average execution time for the Montage workflow 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.

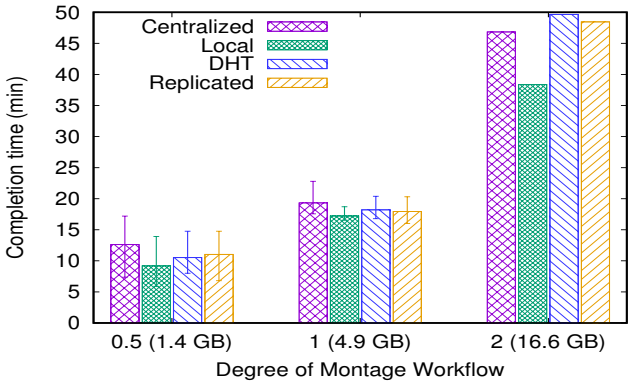


Fig. 6: Montage execution time for different strategies and degrees. Avg. intermediate data shown in parenthesis.

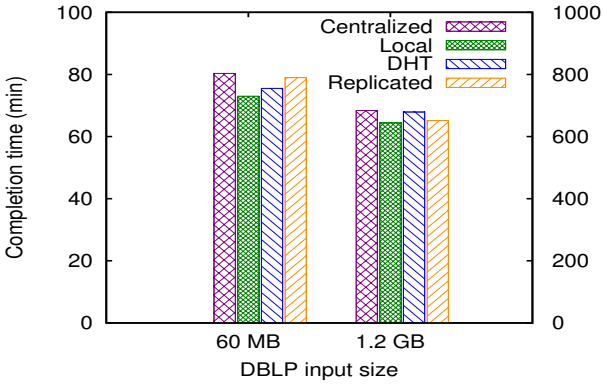


Fig. 7: Buzz workflow execution time. Left Y-axis scale corresponds to 60 MB execution, right Y-axis to 1.2 GB.

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. 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 execution 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 workflow, which is rather data intensive, with two DBLP database dumps of 60 MB and 1.2 GB. The results are shown in Figure 7; note that the left and right Y-axes differ by one order of magnitude. 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 workflow, 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 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 (if small) come at no additional cost for the user.

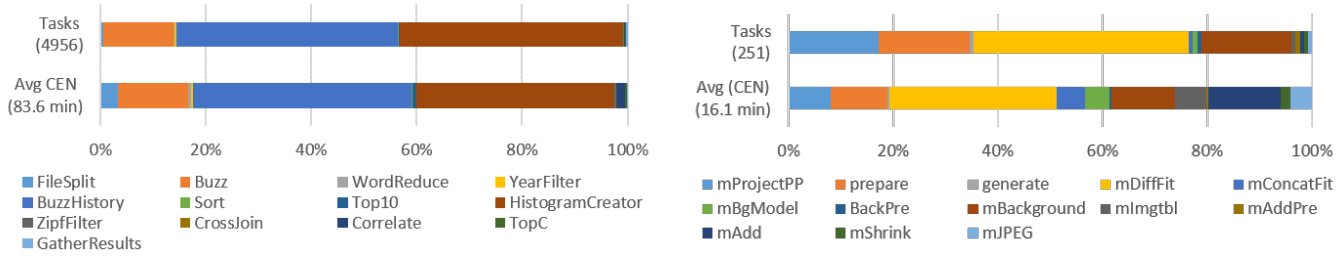
Actual monetary savings Our longest experiment (Buzz 1.2 GB) runs in the order of hundreds of minutes. With today's scientific experiments running at this scale and beyond, a gain of 10% actually implies savings of hours of cloud computing resources.

D. 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 8 shows a breakdown of Buzz and Montage workflows 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 workflow 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 9 and 10 we look into the execution time of the multi-task jobs of Buzz and Montage, respectively. Figure 9 corresponds to Buzz SWf with 60 MB input data. 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).



(a) Buzz

(b) Montage

Fig. 8: Workflow per-job breakdown. Very small jobs are enhanced for visibility.

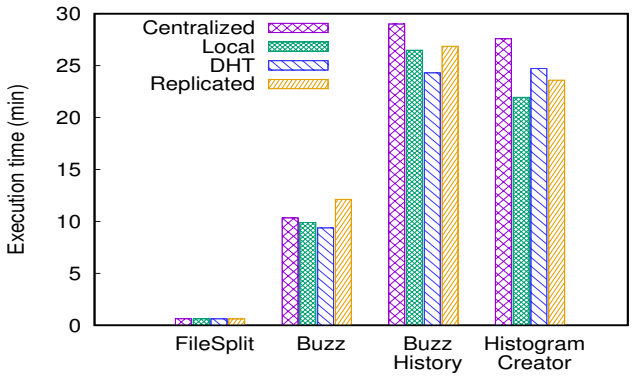


Fig. 9: Execution time of multi-task jobs on the Buzz workflow with 60 MB input data.

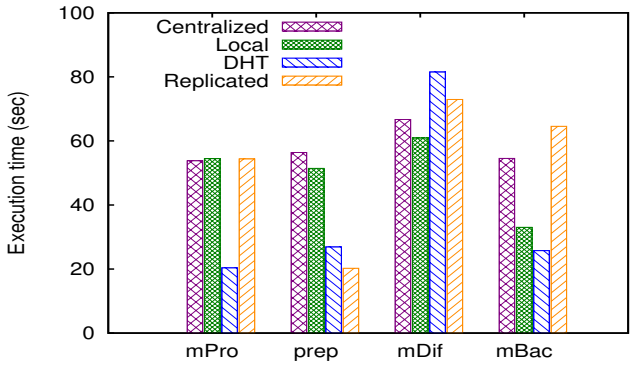


Fig. 10: Execution time of multi-task jobs on the Montage workflow of 0.5 degree.

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 7) since there are still a number of workloads executed sequentially, which have not been optimized by the current release of DMM-Chiron.

Correspondingly, Figure 10 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).

VII. Related Work

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 Taverna [8], Pegasus [7] or Chiron [14] 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 [26], [27]. A lightweight alternative to databases is indexing the metadata; although most indexing techniques [28], [29] are designed for data rather than metadata. Even the dedicated index-based metadata schemes [30] use a centralized index and are not adequate for large-scale workflows, nor can they scale to multisite deployments.

Distributed approaches. Some workflow systems opt to rely on distributed file-systems that partition the metadata and store it at each node (*e.g.* [31], [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 [33] uses full pathnames as key in the underlying HBase [34] store. Lustre [35] hashes the tail of the filename and the ID of the parent directory. Similar hashing schemes are used by [36], [37], [38] with a low memory footprint, granting access to data in almost constant time. FusionFS [39] 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 workflows that run on supercomputers. However, they are unable to meet the practical requirements of workflows executed on clouds. Similarly to us, CalvinFS [10] uses hash-partitioned key-value metadata across geo-distributed datacenters to handle small files, yet it does not account for workflow semantics.

Hybrid approaches. More recently, Zhao *et al.* [41] pro-

posed using both a distributed hash table (FusionFS [39]) and a centralized database (SPADE [42]) 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.

VIII. Conclusion

In this paper we introduced the concept of hot metadata for scientific workflows 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. 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. Coupled with a scientific workflow engine, our strategies showed an improvement of up to 28% for the whole workflow's completion time and 50% for specific highly-parallel jobs, compared to state-of-the-art centralized solutions, at no additional cost.

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. We are also looking at the possibility of adding data location awareness in order to minimize the impact of large intermediate data transfers. 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] "Resource Quotas - Google Cloud Platform," https://cloud.google.com/compute/docs/resource-quotas.
[2] "Alice Collaboration," http://aliceinfo.cern.ch/general/index.html.
[3] "Microsoft Azure Cloud," http://www.windowsazure.com/en-us/.
[4] "Amazon Elastic Compute Cloud," https://aws.amazon.com/ec2/.
[5] "Google Cloud Platform," https://cloud.google.com/.
[6] E. Deelman, D. Gannon *et al.*, "Workflows and e-science: An overview of workflow system features and capabilities," *Future Generation Computer Systems*, vol. 25, no. 5, pp. 528–540, 2009.
[7] E. Deelman, G. Singh *et al.*, "Pegasus: A framework for mapping complex scientific workflows onto distributed systems," *Scientific Programming*, vol. 13, no. 3, pp. 219–237, 2005.
[8] K. Wolstencroft, R. Haines *et al.*, "The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud," *Nucleic Acids Research*, vol. 41, no. W1, pp. W557–W561, 2013.
[9] J. Liu, E. Pacitti *et al.*, "Scientific workflow scheduling with provenance support in multisite cloud," in *High Performance Computing for Computational Science VECPAR*, 2016.
[10] A. Thomson and D. 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.
[11] S. R. Alam, H. N. El-Harake *et al.*, "Parallel I/O and the metadata wall," in *Proc. of the 6th Workshop on Parallel Data Storage*, ser. PDSW '11. New York, NY, USA: ACM, 2011, pp. 13–18.
[12] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 29–43, Oct. 2003.
[13] F. Schmuck and R. Haskin, "GPFS: A shared-disk file system for large computing clusters," in *Proc. of the 1st USENIX Conference on File and Storage Technologies*, ser. FAST '02, Berkeley, CA, USA, 2002.
[14] E. Ogasawara, J. Dias *et al.*, "An algebraic approach for data-centric scientific workflows," *Proc. of VLDB Endowment*, vol. 4, no. 12, pp. 1328–1339, 2011.
[15] G. Juve, A. Chervenak *et al.*, "Characterizing and profiling scientific workflows," *FGCS*, vol. 29, no. 3, pp. 682 – 692, 2013.
[16] E. Deelman, S. Callaghan *et al.*, "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.
[17] "USGS ANSS - Advanced National Seismic System," http://earthquake.usgs.gov/monitoring/anss/.
[18] 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*, Sept 2015, pp. 294–303.
[19] J. J. Levandoski, P.-A. Larson, and R. Stoica, "Identifying hot and cold data in main-memory databases," in *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*. IEEE, 2013, pp. 26–37.
[20] D. Gibson. (2012) Is Your Big Data Hot, Warm, or Cold? [Online]. Available: http://www.ibmbigdatahub.com/blog/your-big-data-hot-warm-or-cold
[21] S. Bharathi, A. Chervenak *et al.*, "Characterization of scientific workflows," in *Workshop on WFs in Support of Large-Scale Science*, 2008.
[22] "Azure Speed Test," http://www.azurespeed.com/.
[23] "Microsoft Azure Service Bus," https://azure.microsoft.com/en-us/services/service-bus/.
[24] E. Deelman, G. Singh *et al.*, "The cost of doing science on the cloud: The montage example," in *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, ser. SC '08, 2008, pp. 50:1–50:12.
[25] J. Dias, E. Ogasawara *et al.*, "Algebraic dataflows for big data analysis," in *Big Data, 2013 IEEE Intl. Conf. on*, 2013, pp. 150–155.
[26] M. Stonebraker, S. Madden *et al.*, "The end of an architectural era: Time for a complete rewrite," in *Proc. of the 33rd Intl. Conf. on Very Large Data Bases*, ser. VLDB '07, pp. 1150–1160.
[27] 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*, April 2005, pp. 2–11.
[28] J. Wang, S. Wu *et al.*, "Indexing multi-dimensional data in a cloud system," in *Proc. of the 2010 ACM SIGMOD Intl. Conf. on Management of Data*, 2010, pp. 591–602.
[29] S. Wu, D. Jiang *et al.*, "Efficient b-tree based indexing for cloud data processing," *Proc. VLDB Endow.*, vol. 3, no. 1-2, pp. 1207–1218, 2010.
[30] A. W. Leung, M. Shao *et al.*, "Spyglass: Fast, scalable metadata search for large-scale storage systems." in *FAST*, vol. 9, 2009, pp. 153–166.
[31] A. Gehani, M. Kim, and T. Malik, "Efficient querying of distributed provenance stores," in *ACM Int. Symposium on High Performance Distributed Computing HPDC*, 2010, pp. 613–621.
[32] T. Malik, L. Nistor, and A. Gehani, "Tracking and sketching distributed data provenance," in *Sixth Int. Conf. on e-Science*, 2010, pp. 190–197.
[33] "Giraffa," https://code.google.com/a/apache-extras.org/p/giraffa/.
[34] "Apache HBase," http://hbase.apache.org.
[35] "Lustre - OpenSFS," http://lustre.org/.
[36] P. F. Corbett and D. G. Feitelson, "The vesta parallel file system," *ACM Trans. Comput. Syst.*, vol. 14, no. 3, pp. 225–264, Aug. 1996.
[37] E. L. Miller and R. H. Katz, "RAMA: An easy-to-use, high-performance parallel file system," *Parallel Computing*, vol. 23, no. 4, pp. 419–446.
[38] S. A. Brandt, E. L. Miller *et al.*, "Efficient metadata management in large distributed storage systems," in *Proc. 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies*, 2003.
[39] D. Zhao, Z. Zhang *et al.*, "Fusionfs: Toward supporting data-intensive scientific applications on extreme-scale high-performance computing systems," in *2014 IEEE Intl. Conf. on Big Data*, Oct 2014.
[40] R. Souza, V. Silva *et al.*, "Parallel execution of workflows driven by a distributed database management system," in *ACM/IEEE Conference on Supercomputing, Poster*, 2015.
[41] D. Zhao, C. Shou *et al.*, "Distributed data provenance for large-scale data-intensive computing," in *CLUSTER*, 2013, pp. 1–8.
[42] M. J. Zaki, "Spade: An efficient algorithm for mining frequent sequences," *Machine Learning*, vol. 42, no. 1, pp. 31–60.