Towards a Smart, Internet-Scale Cache Service for Data Intensive Scientific Applications

Yubo Qin, Anthony Simonet, Philip E. Davis, Azita Nouri, Zhe Wang
Manish Parashar, Ivan Rodero
{yubo.qin;anthony.simonet;philip.e.davis;azita.nouri;jay.wang;parashar;irodero}@rutgers.edu
Rutgers Discovery Informatics Institute
Piscataway, New Jersey, USA

ABSTRACT

Data and services provided by shared facilities, such as large-scale observing facilities, have become important enablers of scientific insights and discoveries across many science and engineering disciplines. Ensuring satisfactory quality of service can be challenging for facilities, due to their remote locations and to the distributed nature of the instruments, observatories, and users, as well as the rapid growth of data volumes and rates.

This research explores how knowledge of the facilities usage patterns, coupled with emerging cyberinfrastructures can be leveraged to improve their performance, usability, and scientific impact. We propose a framework with a smart, internet-scale cache augmented with prefetching and data placement strategies to improve data delivery performance for scientific facilities. Our evaluations, which are based on the NSF Ocean Observatories Initiative, demonstrate that our framework is able to predict user requests and reduce data movements by more than 56% across networks.

CCS CONCEPTS

• **Computer systems organization** → *Cloud computing*; *Client-server architectures*; • **Information systems** → *Data exchange*.

KEYWORDS

Cyberinfrastructure, Virtual Data Collaboratory, Distributed Data Sharing, Distributed Facilities, Prefetching, Data Repository

ACM Reference Format:
Yubo Qin, Anthony Simonet, Philip E. Davis, Azita Nouri, Zhe Wang and Manish Parashar, Ivan Rodero. 2019. Towards a Smart, Internet-Scale Cache Service for Data Intensive Scientific Applications. In *10th Workshop on Scientific Cloud Computing (ScienceCloud'19), June 25, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3322795.3331464

1 INTRODUCTION

Data and services provided by large-scale sensor networks, instruments and observatories, coupled with the computation power provided by Advanced Cyberinfrastructures (ACI), are increasingly driving scientific insights and discoveries across many science and engineering disciplines. As a result, the quality of service provided by these facilities and the associated cyberinfrastructures is an important concern and has a direct impact on the ability of researchers to effectively use these facilities. However, ensuring the performance of data delivery can be challenging due a to the remote location and distributed nature of many of these instruments and observatories. This data delivery challenge has been exacerbated by the rapid growth of the data volume and variety being requested, as well as size of the user communities consuming the data.

While application-level solutions have been proposed to reduce the amount of data movement in geo-distributed data analytics [4, 8, 15] (and despite the general trend of bringing the application code to the data) the *pull-based* data access interface that most observatory repositories provide is becoming obsolete. Modern dynamic, event-based and data-driven applications require *push-based* Application Programming Interfaces, such as Pub/Sub, to react to natural phenomena in near real-time [16]. Without a push-based interface being available, the burden of optimizing data access and movements across geo-distributed application components falls upon application developers. Cyberinfrastructure (CI) users have to work around these limitations at the application level, repeatedly polling data from the repositories and incurring long transfer times due to the heterogeneity of the Wide-Area Networks (WAN) and of the ever-increasing number of instruments and growing data volume. Eventually, this lack of support from the CIs will impact both the science impact of users and the quality of service of the facilities.

This research addresses these challenges and explores how information about facilities' access patterns, coupled with emerging cyberinfrastructure solutions, can be used to improve the performance, usability and scientific impact of data and services provided by these facilities. In order to understand the usage patterns of large facilities we have conducted an analysis inspired by the National Science Foundation (NSF) Ocean Observatory Initiative (OOI) [18] and identified several classes of users and usage patterns (e.g., interactive vs. recurrent automated requests). We also study the cyberinfrastructure requirements of these user classes and usage patterns.

In particular, we exploit the Virtual Data Collaboratory (VDC) platform [14] and propose to build up a smart internet-scale cache framework. Through analysis of user access logs from the OOI repository, we identify user access patterns and correlations among user requests. Based on these discoveries, we design a prefetching model and a data placement strategy that can potentially benefit most of the large scale observatories. To demonstrate this, we evaluate our solution against OOI access logs and show that such a cache framework could significantly improve the performance of data retrieval for OOI users.

The main contributions of this paper are summarized as follows:

- We present the results of an analysis based on the NSF OOI, and identify request patterns and correlations between them.

- We propose a data prefetching model and cache data placement strategies, as well as an internet-scale cache service framework.
- We evaluate the model with log data extrapolated from the NSF OOI data repository and demonstrate the performance improvement for user data retrieval requests.

The remainder of the paper is structured as follows. Section 2 presents the motivation for building a distributed data cache framework and related works. Section 3 analyzes the user access patterns contained in OOI access logs. Section 4 describes the system design and model. We provide an experimental evaluation of the system performance in Section 5 through simulation. Finally, we conclude the article and outline ongoing and future research activities.

2 BACKGROUND AND RELATED WORK

This section discusses data exchange techniques in existing geo-distributed applications, scientific workflows and observatories to draw motivations for this work.

2.1 Geo-distributed applications

Many scientific experiments are distributed in nature. Their distribution is sometimes due to the distance between the facilities that generate data and those that store it; sometimes it is due to researchers collaborating remotely; in many cases, it results from both. This distribution creates various challenges.

For researchers, geo-distribution leads to difficulties in accomplishing relatively mundane tasks such as transferring and sharing data sets, even when advanced networking technologies are available. Unlike the Large Hadron Collider (LHC) [5] or the Human Genome Project [6], which can afford a dedicated cyberinfrastructure and a well-tuned software stack, for most of the geo-distributed applications, the cost of retrieving data from remote repositories is non-trivial and hard to avoid: repositories typically are not able to provide co-located computing resource. Therefore, these applications have to transfer data to a remote computing facility.

Reducing the cost of geo-distributed applications' data retrieval from the repositories has not been holistically studied yet, although several research works have proposed solutions to specific challenges. At the application level, research has been done on data analytics [15], machine learning [4, 8] and Big Data processing [9] in an attempt to alleviate the overhead of data exchange and communication. Jiang et al. [10] explored caching the datasets shared in common by geo-distributed applications to reduce network traffic. At a systems level, Liu et al. [12] studied the bottleneck of wide area data transfer and exploited reinforcement learning [13] to continuously find the ideal configuration for Data Transfer Nodes (DTN) and thus optimize networks for scientific applications.

However, these works only addressed the applications' intermediate data exchange or optimized the data transfers, they did not seek to alleviate the application's initial data retrieval cost and user wait time. We believe that an optimal solution should combine both comprehensive studies of applications and optimization at the system level.

2.2 Scientific workflows based on observatory data

2.2.1 Observatory data. It refers to data related to the observation of phenomena. Observatory data is typically produced by scientific instruments and sensor networks and is hosted and distributed by dedicated data repositories.

Observatory data has several defining characteristics. First, observatory data is immutable; most records are instantaneous values of an observation of a certain nature at a given location, and the complete history of these values form a time series. Second, observatory data is reused by different applications for different research purposes; these two properties mean that most observatories operate in a *write once, read many* storage mode. Third, observatory data is frequently spatiotemporal data. Every spatiotemporal data point has a spatial attribute that indicates the location where the observation was made, often in the form of a *(longitude, latitude)* couple. A timestamp attribute indicates when the observation was made.

Li et al. have shown in [11] that about 80% of tile access patterns for geospatial data have a locality feature, meaning that knowledge of past queries can be used to improve the service of future queries. The present work is partly based on the expectation that similar locality features can be found in the spatiotemporal data managed by scientific observatories, and that these features can be leveraged to offer advanced caching mechanisms, including prefetching.

2.2.2 Scientific workflows. They typically use observatory data for two purposes. The first one is monitoring the occurrence of a certain kind of event, which requires continuously examining the freshest data available in short time intervals. The second purpose is investigating the evolution of some natural phenomenon. Workflows of this kind retrieve many data objects at once which span long periods of time.

These two purposes for access usually correspond to two data retrieval methods and have different access patterns. For compatibility, most observatories offer HTTP interfaces (either through a machine-to-machine interface such as a REST API, or through an interactive portal). The API-based workflows are usually programmed to periodically request a set of data objects for a fixed time range, which makes their request patterns consistent and easy to predict. On the other hand, the workflows using interactive interfaces usually involve a person and thus create access patterns that are difficult to predict based on previous queries.

Understanding the access patterns of workflows would be useful in improving the synchronous and *pull-based* transfer methods used by most existing observatories, which limit opportunities for the servers to optimize data delivery.

2.2.3 Observatory data retrieval correlation. Observatory data is mainly used for scientific research purposes. The data retrieval behavior is thus not random and there is a correlation between requests in two aspects, *i*) data type, and *ii*) spatiotemporal property.

First, the observatory scientific workflows have their own research purposes. The types of data requested by the workflows used for research in a given discipline follow scientific laws. For example, a weather forecast workflow requires data about temperature, wind speed, humidity, etc; air quality research programs require

atmospheric metrics (carbon dioxide, ozone, etc.) traffic data, and so on. This establishes correlations in data access, according to the types of workflows accessing the data. Second, the observatory data is spatiotemporal data. The application data retrieval also follows rules in terms of locality and time series.

By learning data access patterns, it is feasible to improve user data retrieval performance by developing a *push-based* and asynchronous data delivery framework.

3 OBSERVATORY DATA ACCESS PATTERN

This section presents a study of the access logs of the OOI from the month of November 2018, in order to classify users according to their access patterns.

3.1 Mining correlation

3.1.1 Log data attributes. Each request includes a data object name, instrument name, and requested time range. We model the user access pattern by mining the correlation among these three attributes individually.

3.1.2 Data object. The data object name represents the instrument measurement data, often as a time series. For example, the "CTD" instrument measures conductivity, temperature and depth. Achieving scientific insight from instrument data will generally require the examination of several inter-related data objects, and so we hypothesize correlation among user-requested data objects.

3.1.3 Instrument locality. OOI instruments are naturally distributed into groups. As shown on the OOI website [2], they are clustered into five groups, which are distributed in the Pacific and Atlantic Ocean. Each group is distant from the others and represents a research area. We hypothesize that a user's requests are aggregated to one or few clusters and there is additionally spatial correlation among its requests across the instruments within a cluster.

3.1.4 Time range. The time range indicates the start and end time point of the data in which the user is interested. Because the observatory instruments have fixed data sampling frequency, the time range is linearly proportional to the data size of the response.

3.1.5 Correlate user request pattern. To test this hypothesis, we extract two attributes from the user requests, data object and instrument location and plotted them. Figure 1 is an example with 3 selected users. The y-axis is the enumerated data stream name. The x-axis is the instrument location ID which is sorted by the instrument clusters as mentioned in Section 3.1.3, so that adjacent number correspond to instruments which are close to each other. For example, the instrument location ID from #0 to #31 are instruments that are located in the North Pacific Ocean cluster.

By plotting requests user-by-user, we made several observations. First, there is a correlation among data objects and instrument locations. A user (e.g., User #1) would successively request the same set of data objects from multiple locations. Second, users share similar access patterns, such as User #1 and User #2, User #1 and User #3. These observations provide feasibility for request prediction and data cache.

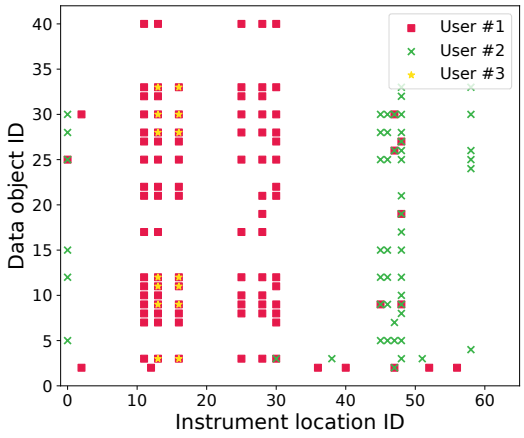


Figure 1: User request pattern analysis. The x-axis is the instrument location and the y-axis is the data object ID. The visible line patterns suggest that correlations localities and data objects exist.

3.2 User classification

Users can be classified into two categories according to their data retrieval methods, *interactive users* and *Machine-to-Machine (M2M) users* (programs). Through analyzing one month of OOI log data, we found that 42.7% users are programs. These programs generated 99.9% of the requests and 98.8% of the total volume of data movement from OOI.

Programs can be further classified into real-time users and non-real-time users according to their request interval. We noticed that some users queried data at very high frequency (e.g. few seconds) and thus classified them as the real-time users.

Because programs' access and request patterns are typically consistent and easy to recognize, it is feasible to design a data prefetching model that improves the response time for their requests. In other words, we can build a preemptive *push-based* data delivery framework instead of the passive *pull-based* data delivery pattern and therefore improve the user data retrieval performance.

4 SMART, INTERNET-SCALE CACHE SERVICE FRAMEWORK

4.1 System infrastructure

4.1.1 Cyberinfrastructure platform. The target infrastructure is the Virtual Data Collaboratory (VDC) platform [14]. The VDC uses the Science DMZ model, a concept adopted by several other regional and national cyberinfrastructures, such as the Pacific Research Platform (PRP) and the National Research Platform (NRP) [17]. The Science DMZ model is a network of DTNs that act as access points for the CIs and other institutions to connect to other institutions on a dedicated Wide-Area Network. In our design, all participating institutions are connected through a DTN, meaning all requests and data transfers initiated by users have to go through at least one of these fast I/O nodes. This architecture gives the DTNs the

opportunity to transparently cache data, without incurring any additional performance cost.

It is important to note that while all DTNs follow the same Flash I/O Network Appliance (FIONA) standard proposed by the Prism@UCSD project [1], there is still a lot of heterogeneity in the network and between the nodes. First, it is impossible to guarantee the same quality (throughput and latency) of service for all the links at the scale of a country like the United States. This can, for example, be observed on the PRP dashboard [3] that usually shows important variations between network links. Second, nodes are purchased, configured and installed by different institutions and over long time frames. While they have to implement the specification, they can still vary enough in terms of hardware (CPU, RAM, and storage) and software to impact the framework performance.

This important heterogeneity along with the network topology must be taken into account when designing our cache framework and its prefetching and placement strategies in particular, in order to deliver the best possible performance.

4.2 Prediction model

We train our prediction model on user requests from observatories. Based on this information, we represent a log dataset as a sequence $R = \langle r_1, r_2, r_3, \ldots, r_n \rangle$. Each request tuple r_i includes the name of the data object (or stream) d; the location of the instrument l; and the time range t. Equation 1 develops a requests sequence R in which $D_n = \langle d_1, d_2, \ldots, d_n \rangle$ is the sequence of data streams, $L_n = \langle l_1, l_2, \ldots, l_n \rangle$ is the sequence of instrument locations and $T_n = \langle t_1, t_2, \ldots, t_n \rangle$ is the sequence of time ranges.

$$\begin{aligned} R &= \langle (d_1, l_1, t_1), (d_2, l_2, t_2), \ldots, (d_n, l_n, t_n) \rangle \\ &= D_n, L_n, T_n \end{aligned} \tag{1}$$

We observed that OOI users using the M2M interface (i.e., programs) usually have more consistent request patterns compared to interactive users' (i.e., humans) that are thus far more difficult to predict. In this work, we design a hybrid prediction model that combines a historical record-based prediction method and an associative rule mining method to model and predict both categories of user requests.

4.2.1 Historical record-based prediction model. Through the analysis of the OOI requests, we found that some users have consistent patterns in their requests. These patterns, that are easily recognizable in the information of the requests, can take different forms: *i*) requests are occurring at a fixed interval (e.g., every 4 hours); *ii*) requests are for the same set of data objects; or *iii*) requests are for a fixed-size sliding window. These requests can be easily recognized in the historical record. As the framework reads through the requests, it defines a *pattern repeat threshold* ξ_r. Once a user's access pattern repeats over the threshold ξ_r, the system marks it as predictable and starts prefetching data in anticipation of future requests.

The access interval of real-time programs is too short for prefetching data, so we categorize them as *unprefetchable*. These users make up 2.7% of total users, and the remaining 54.6% automated users are possibly prefetchable.

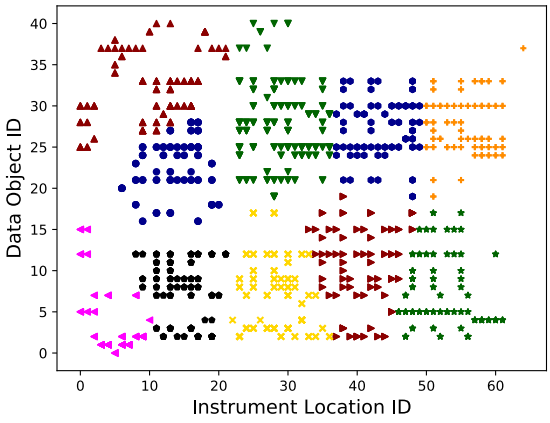


Figure 2: Clusters of users produced by K-Means with $K = 10$ for the OOI logs datasets. Clusters are represented by identical symbols and colors and contain users that share similar features in terms of instrument location and data interest.

4.2.2 Associative rule mining prediction model. As discussed in Section 2, observatory data has spatiotemporal attributes, and there are correlations among user requests. We use the association rule mining FP-Growth algorithm [7] to build up the prediction model, as seen in other works [11, 19, 20]. FP-Growth will find an association among past requests to predict future requests based on two attributes, the data object D_n and instrument location L_n. The model is constructed as follows:

a) FP-tree construction: the frequent-pattern tree (FP-tree) is a compact structure that stores quantitative information about frequent patterns in a database. The algorithm first scans the training dataset and counts the number of times a data point $d_i \in D_i, l_i \in L_i$ appears in the sequence of requests. This number is called *support*. Then, the algorithm finds the frequency 1-itemsets by comparing the itemset *support* with a predefined *support threshold* δ_s. Finally, it rescans the dataset and constructs the FP-tree.

b) FP-Growth: based on the FP-tree that contains association rules between itemsets and their corresponding *confidence* value ϕ_j as in $(d_i, d_{i+1}, d_{i+2}, \ldots, d_{i+m-1}) \rightarrow \langle d_{i+m}, \phi_j \rangle$, the algorithm filters out association rules which have a *confidence* lower than the *confidence threshold* ϕ_c, in order to form the complete set of frequent patterns.

Selecting appropriate threshold values for *support* δ_s and *confidence* ϕ_c is important to the model performance. We will determine these values through experimentation in Section 5.

Once the model has predicted the data object that will be requested next, the system places a copy of the object in the user's local DTN cache. This copy has the same time range as the user's last request for a different object.

4.3 Virtual groups

In a geographically distributed scenario, the placement of cached data is expected to have a significant performance impact due to variations in the network, as discussed earlier. Our objective here is thus to place cached data as close as possible to potential users.

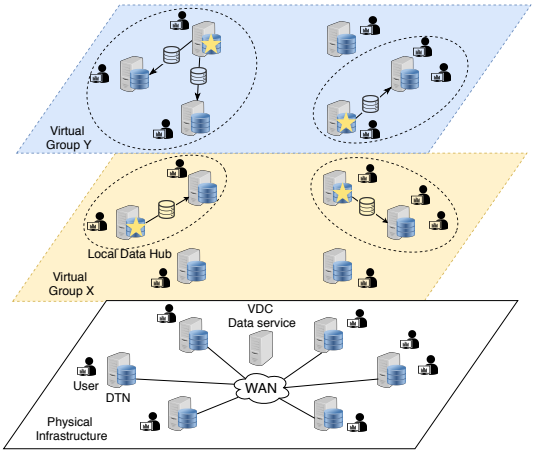


Figure 3: Virtual groups and Local Data Hub (star). The layered architecture is inspired by [10]. On top of the physical infrastructure, users can belong to several groups (layers) that correspond to the clusters found by the K-Means algorithm. Based on the distance between users, each virtual group may be split into several sub-groups (represented by circles).

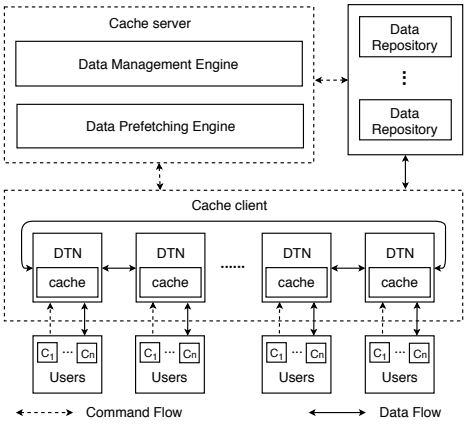


Figure 4: Cache system framework design.

To determine the optimal location for cached data, we rely on users' access patterns. We define a *virtual group* as a group of users that share common interests in data objects and are geographically close to each other. In order to find users with similar interest, we run the K-Means algorithm on the historical records and partition the requests into K clusters. If a user requested data objects that belong to different clusters, then the user belongs to as many different virtual groups. Figure 2 shows the clustering result of the OOI logs. The logs that we currently have do not contain any location information; in order to group users that are geographically close, we assume that their location information can be interpreted from a user's IP address and that we know which DTN each user accesses.

Figure 3 shows the concept of the virtual group. At the bottom is the physical infrastructure composed of DTNs that acts as the users' access points to the network. Every past request made by a user maps to a cluster. Each user is put in virtual groups depending on which clusters they have accessed in the past, as shown in the upper two layers. On the figure, *Virtual Group X* and *Virtual Group Y* are two clusters as output by K-Means. Each virtual group can be further partitioned based on the relative distance between users (circles).

The clustering algorithm is executed at a pre-defined frequency (e.g. daily), which allows the system to adapt when new users join the network or when existing users start accessing data objects that belong to a cluster they have not accessed in the past.

4.4 Local data hub

Selecting an optimal DTN within each virtual group for storing the cache data is critical for performance. In each group, we call this DTN the *local data hub*. From the perspective of the DTN, there are two parameters affecting the cache performance: the network throughput and the resource utilization (e.g., storage, CPU). From the perspective of the user, if their request frequency is high, it is better to choose the DTN they are connected to as the local data hub, in order to improve locality.

For each DTN in a virtual group, the local data hub selection algorithm accumulates the weighted values of the above three factors: throughput, resource utilization, and the frequency of user requests.

Equation 2 illustrates the local data hub selection approach. We assume that virtual group G has n DTNs $\{v_1, v_2, ..., v_n\}$. Within this group, P_{ij} represents the network throughput from v_i to v_j and θ_p is the weight for throughput. U_i represents the DTN device resource utilization (e.g., storage and CPU) with weight θ_u. F_i represents the frequency of request that comes from the virtual group members that are connected to this DTN, and has weight θ_f. As the data hub, we select the DTN V_{dh} that maximizes the sum of the three values.

$$V_{dh} = max(\theta_p \sum_{j \neq i}^{n} P_{ij} + \theta_u U_i + \theta_f F_i),\ 0 < i, j \leq n \qquad (2)$$

The local data hub selection algorithm is running at a defined interval. Once the local data hub changes, the previous data hub keeps the data that is already cached and its user-defined eviction policy, while new data will be cached to the new data hub to keep the reconfiguration cost minimal.

4.5 System design

The framework features a server-client architecture, as shown in Figure 4. The cache server hosts the Data Management and the Data Prefetching engines that run the prediction and the clustering algorithms discussed earlier in this section. The *Data Management Engine* acts as a proxy to serve queries made by the users to the data repositories. The cache server also maintains the cache index and dynamically manages the virtual groups and the local data hubs. The *Data Prefetching Engine* learns the user access pattern and manages prefetching transfers from the repositories to the correct DTN. In case of a cache miss (i.e. the object that is being requested by a user is not in the local cache), the Prefetching Engine decides

whether the object should be downloaded by the local cache from a remote DTN or from the repository by comparing the transfer costs of each. The prefetching model is periodically updated with a pre-defined frequency, such as hourly or daily.

The cache client is running on the DTN network, close to the user and is in charge of processing user requests. If the requested data is not in the local DTN cache, it will pass the command to the cache server. The *Data Prefetching Engine* records the request and the *Data Management Engine* provides a data download link, either from a local DTN cache or the original data repository according to the data transfer cost. Once the user receives the data, the cache client will cache them and evict old cache data if necessary.

It is noteworthy that with this architecture, the flow of commands goes only from the cache server to the cache clients, and that the flow of data goes directly from the repositories to the cache client.

5 PERFORMANCE EVALUATION

This section describes the experimental setup and results. Our experiments are based on simulations of a distributed DTN network similar to the VDC combined with PRP. Our experiments evaluate the performance of the Smart Internet-Scale Cache framework using the OOI logs.

5.1 Experimental setup

5.1.1 OOI data requests. The anonymized data log was obtained extrapolating OOI requests from the month of November 2018. This dataset contains over 17 million valid user pull records. As user identities are anonymized, we cannot interpret user location information and have to randomly distribute them to the DTN network, which limits our ability to cluster users.

Because the logs do not contain the amount of data transferred to satisfy a request, we have to infer this from the instrument sampling frequency and time range. We define the *data density* as the size of a data object that an instrument generates every second, in MB per second. Then, for each request, we can estimate the data size through multiplying the density by the request time range in seconds. The data density of each data object was calculated through experiment. We retrieved 3 hours data for each of them and dividing the total downloaded size by 10,800.

5.1.2 Simulate VDC DTN network. To simulate a realistic DTN network, we selected eight 40GB-DTNs from the PRP web dashboard and recorded their instant throughput and resource utilization on Feb. 13, 2019 at 15:39:00. These eight DTNs are located in eight distinct institutes in California. These nodes are distributed over a large geographic area and their network throughput is heterogeneous, which is representative of an Internet-Scale DTN network.

We select one DTN as the access point to the OOI data repository and randomly distribute all the OOI users to the other seven DTNs. The DTN throughput value and resource utilization is only used in selecting the local data hub; we leave the computation of data transfer times as future work, as we don't account for network contention in this work.

5.1.3 Evaluation metric. From the cache point of view, data is stored in a 3-tier hierarchy at the top of which is the data repository, followed by the remote DTN caches, and the local DTN cache, in the order of increasing data transfer speed to a user. Hence, our cache strategy aims at moving data to the user's connected DTN, in order to allow the user applications to get as much data as possible from the local DTN.

The experiments compute two metrics for evaluation: the *cache hit rate* and the volume of *data movement*. Further, they are divided in *cache hit rate at remote DTNs*, *cache hit rate at local DTN*, *data movement from remote DTN* and *data movement from local DTN*. Improving performances for users means maximizing the *cache hit rate at local DTN* and *data movement from local DTN*.
For cache storage capacity, we chose values that are typically observed on DTNs that implement the FIONA specification [1]: 128GB, 256GB, 512GB, 1TB and 2TB.

5.2 Evaluation of the prediction model parameters

This experiment aims at determining the optimal parameters for the prediction model.

5.2.1 Historical record-based prediction model. A historical record-based prediction mode is usually appropriate for programs, except for real-time programs. Because these users query the freshest data at a very high frequency, there is no sufficient time to prefetch data for them. To determine if a user is *prefetchable*, we set the *pattern repeat threshold* ξ_r to 3 and the *access interval threshold* ξ_a to 60 seconds. This means that a user is considered *prefetchable* if accesses the same objects in the same sequence at least three times, with a pause of at least 60 seconds between each sequence. We found these values to be sufficient to identify the repeated patterns while excluding real-time users, and these values can be tuned in the simulator.

5.2.2 Associative rule mining-based prediction model. The FP-growth model has two major parameters, the *support threshold* δ_s and the *confidence threshold* ϕ_c. δ_s indicates how frequently the itemset appears in the dataset and ϕ_c specifies how often the rule has been found to be true. There are trade-offs in selecting these values; higher δ_s values will result in selecting fewer itemset candidates and higher ϕ_c will reduce the size of the ruleset. We vary these two parameters and calculate the model's *precision* and *recall* values.

In this experiment we initially train the prefetching model with one week of OOI requests and retrain the model every day with the freshest data. We then compute the average precision and recall for each user at the end of the month. Figure 5 plots the average precision (top) and recall (bottom) of all the users with different support and confidence threshold combinations. Based on these results the values with the best trade-off seem to be δ_s=30 and ϕ_c=0.5, as these values maximize the recall while maintaining satisfying precision. We fix these values and use them for the rest of the experiments.
Our results are however limited by the small size of our dataset, since 37.3% of users made only a single request during that month, and our initial training dataset is only one week. This experiment still demonstrates the feasibility of prefetching for observatory data, and we hope to improve these results by using larger datasets in the future.

The next section evaluates the two prefetching models as part of the whole Smart Cache framework.

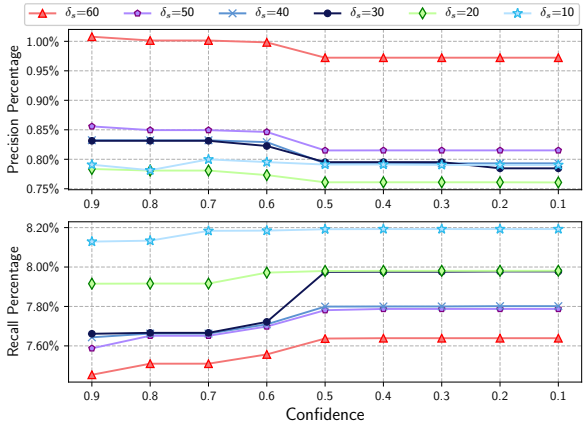


Figure 5: FP-Growth modle performance evaluation with different *support* δ_s and *confidence* ϕ_c threshold combinations. It illustrates the *Precision* (on top) and the *Recall* (on bottom) result.

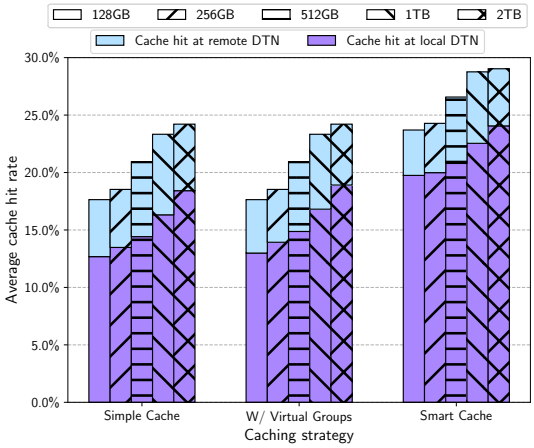
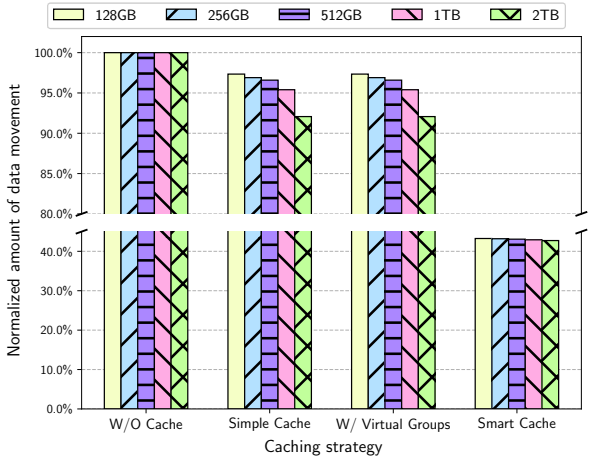
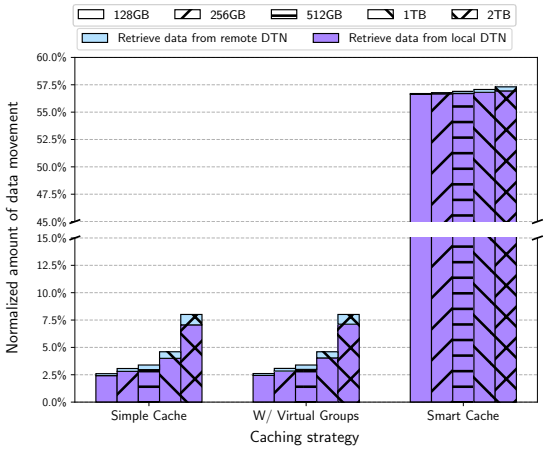


Figure 6: This is the cache hit rate comparison among the three cache strategies. It also traces the cache hit rate at remote DTN and user local DTN.



(a) Data transfer from the remote repository.



(b) Data transfer from the DTN caches.

Figure 7: Volume of data transfer from the OOI data repository and from the DTN caches under the four scenarios.

5.3 Evaluation of the Smart Cache framework

This experiment replays the transfers contained in the OOI logs and compares the performance of the cache under different strategies: no cache, a simple cache without optimization, a simple cache with virtual group data placement optimization and the Smart Cache.

5.3.1 Evaluation scenarios.
No cache: a scenario in which users retrieve data directly from the data repository. This scenario is the baseline for all subsequent experiments and the denominator for result normalizations.

Simple cache: adds a cache layer without any optimization such as prefetching or data placement. We evaluate a simple Least Recently Used (LRU) cache eviction mechanism to determine the impact of data re-use.

Simple cache with Virtual Groups: cache with data placement strategies, virtual groups, and local data hub. This scenario should result in a higher cache hit rate at the local DTN and in improved data retrieval performance thanks to data locality.

Smart cache: the smart cache incorporating the associative rule-based prefetching model, the virtual groups and the data placement strategies.

5.3.2 Results. Figure 6 shows the cache hit rate for the three cache strategies, with varying storage space on the DTNs. We observe that the simple cache alone with the smallest storage space (128GB) achieves 17.6% of cache hit rate. This first result shows that there exists some degree of data reuse to exploit.

Our second scenario reveals that our data placement strategies, virtual group and local data hub alone do not improve the cache hit rates significantly. Compared with the simple cache, the cache hit rate at the local DTN is on average 0.44% greater. It should however be noted that the total volume of data movement is on the scale of TB and therefore even a 1% saving is on the scale of tens of GB.

Finally, we find that the Smart Cache can achieve higher hit rate on local DTNs under restricted storage space. By adding the prefetching mechanism, the smart cache can achieve 19.8% of hit rate on the local DTN with 128GB storage, which is better then the 1TB cache space with virtual groups and placement strategies alone. Also, as the cache size increases, the cache hit rate does not significantly change, which means the smart cache can achieve relative high cache hit within a restricted cache space.

Figure 7a plots the amount of data transferred from the repositories in the four scenarios, normalized against the baseline (no cache, i.e. all data is fetched directly from the repository). Simply increasing the storage space of the DTNs results in a reasonable reduction of the volume of data that users need to retrieve from the remote repository. The Simple Cache and Simple Cache with Virtual Groups scenarios still result in over 92% of data being transferred from the data repository with 2TB of storage. However, diving into the details of cache hits, Figure 7b shows that most cache data was retrieved from the local DTN. In all cases, the reduction in data movement from varying cache space from 128GB to 2Tb is less than 5%. This seem to mean that simply increasing the cache storage space without changing the passive *pull-based* cannot improve the performance of data delivery significantly.

In contrast, the Smart Cache uses a prefetching mechanism to implement a preemptive *push-based* data delivery model. This largely reduces the amount of data transferred (to less than 43.3% of the no-cache scenario under 128GB storage space) from the repository, and makes the local DTN serve more than 56.6% of the data. Considering that 54.6% users are *prefetchable*, our smart cache framework has the potential to significantly improve the data retrieval performance for the users of scientific observatories.

6 CONCLUSION AND FUTURE WORK

In the context of increasing investments in large-scale sensor networks, instruments and observatories, this work studies how leveraging knowledge of past data retrieval requests made to scientific facilities can help improve the user experience of future requests. We have presented a smart, internet-scale cache with prefetching, driven by data analytics and machine learning techniques, able to predict future requests and preemptively place data close to the users that are more likely to request it in the future. We have evaluated our cache framework via simulation using a data set obtained from Ocean Observatory Initiative (OOI) requests containing more than 17 million transfer records. Evaluations show promising results with 57% reduction of data transfers from the OOI repository compared to the current solution.

In the future, we plan to test our cache framework against more data sets from different scientific facilities. One particular challenge that we are eager to address is evaluating the placement model with actual location data, which is currently challenging due to privacy constraints. In the meantime, we are working on a first prototype implementation that will eventually be deployed on the Virtual Data Collaboratory network.

ACKNOWLEDGEMENTS

This work is supported in part by National Science Foundation via grants numbers OAC 1640834, OAC 1835692, and OCE 1745246. This research was conducted as part of the Rutgers Discovery Informatics Institute (RDI^2).

REFERENCES

[1] FIONA - Flash I/O Network Appliance. Retrieved 2019-04-09 from https://fasterdata.es.net/science-dmz/DTN/fiona-flash-i-o-network-appliance/
[2] Ocean Observatories Initiative. Retrieved 2019-03-25 from https://oceanobservatories.org/
[3] PRP Web Dashboard. Retrieved 2019-02-13 from https://perfsonar.nautilus.optiputer.net/maddash-webui/index.cgi?dashboard=NautilusMesh
[4] Ignacio Cano, Markus Weimer, Dhruv Mahajan, Carlo Curino, and Giovanni Matteo Fumarola. 2016. Towards geo-distributed machine learning. *arXiv preprint arXiv:1603.09035* (2016).
[5] ATLAS Collaboration et al. 2008. The ATLAS experiment at the CERN large hadron collider.
[6] Francis S Collins, Michael Morgan, and Aristides Patrinos. 2003. The Human Genome Project: lessons from large-scale biology. *Science* 300, 5617 (2003), 286–290.
[7] Jiawei Han, Jian Pei, and Yiwen Yin. 2000. Mining frequent patterns without candidate generation. In *ACM sigmod record*, Vol. 29. ACM, 1–12.
[8] Kevin Hsieh, Aaron Harlap, Nandita Vijaykumar, Dimitris Konomis, Gregory R Ganger, Phillip B Gibbons, and Onur Mutlu. 2017. Gaia: Geo-Distributed Machine Learning Approaching {LAN} Speeds. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*. 629–647.
[9] Anca Iordache, Christine Morin, Nikos Parlavantzas, Eugen Feller, and Pierre Riteau. 2013. Resilin: Elastic MapReduce over Multiple Clouds. In *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid 2013)*. 261–268. https://doi.org/10.1109/CCGrid.2013.48
[10] Fan Jiang, Claris Castillo, and Stan Ahalt. 2018. Cachalot: A network-aware, cooperative cache network for geo-distributed, data-intensive applications. In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 1–9.
[11] Rui Li, Wei Feng, Huayi Wu, and Qunying Huang. 2017. A replication strategy for a distributed high-speed caching system based on spatiotemporal access patterns of geospatial data. *Computers, Environment and Urban Systems* 61 (2017), 163–171.
[12] Zhengchun Liu, Prasanna Balaprakash, Rajkumar Kettimuthu, and Ian Foster. 2017. Explaining wide area data transfer performance. In *Proceedings of the 26th International Symposium on High-Performance Parallel and Distributed Computing*. ACM, 167–178.
[13] Zhengchun Liu, Rajkumar Kettimuthu, Ian Foster, and Peter H Beckman. 2018. Toward a smart data transfer node. *Future Generation Computer Systems* 89 (2018), 10–18.
[14] Manish Parashar, Vasant Honavar, Anthony Simonet, Ivan Rodero, Forough Ghahramani, Grace Agnew, and Ron Jantz. 2019. The Virtual Data Collaboratory: a Regional Cyberinfrastructure for Collaborative Data-Driven Research. *Computing in Science Engineering* (2019).
[15] Qifan Pu, Ganesh Ananthanarayanan, Peter Bodik, Srikanth Kandula, Aditya Akella, Paramvir Bahl, and Ion Stoica. 2015. Low latency geo-distributed data analytics. In *ACM SIGCOMM Computer Communication Review*, Vol. 45. ACM, 421–434.
[16] Ivan Rodero and Manish Parashar. 2019. Data Cyber-Infrastructure for End-to-end Science: Experiences from the NSF Ocean Observatories Initiative. *Computing in Science Engineering* (2019).
[17] Larry Smarr, Camille Crittenden, Thomas DeFanti, John Graham, Dmitry Mishin, Richard Moore, Philip Papadopoulos, and Frank Würthwein. 2018. The Pacific Research Platform: Making High-Speed Networking a Reality for the Scientist. In *Proceedings of the Practice and Experience on Advanced Research Computing*. ACM, 29.
[18] Leslie M Smith, John A Barth, Deborah S Kelley, Al Plueddemann, Ivan Rodero, Greg A Ulses, Michael F Vardaro, and Robert Weller. 2018. The Ocean Observatories Initiative. *Oceanography* 31, 1 (2018), 16–35.
[19] Lian Xiong, Zhengquan Xu, Hao Wang, Shan Jia, and Li Zhu. 2016. Prefetching scheme for massive spatiotemporal data in a smart city. *International Journal of Distributed Sensor Networks* 12, 1 (2016), 4127358.
[20] Lian Xiong, Liu Yang, Yang Tao, Juan Xu, and Lun Zhao. 2018. Replication Strategy for Spatiotemporal Data Based on Distributed Caching System. *Sensors* 18, 1 (2018), 222.