4CeeD: Real-Time Data Acquisition and Analysis Framework for Material-related Cyber-Physical Environments

Phuong Nguyen, Steven Konstanty, Todd Nicholson, Thomas O'brien, Aaron Schwartz-Duval, Timothy Spila, Klara Nahrstedt, Roy H. Campbell, Indranil Gupta, Michael Chan, Kenton McHenry, Normand Paquin
University of Illinois at Urbana-Champaign
{*pvnguye2,stevek,tcnichol,tobrien3,asschwa2,tspila,klara,rhc,indy,mgc,mchenry,paquin*}*@illinois.edu*

***Abstract*—In this paper, we propose a data acquisition and analysis framework for materials-to-devices processes, named 4CeeD, that focuses on the immense potential of capturing, accurately curating, correlating, and coordinating materials-to-devices digital data in a real-time and trusted manner before fully archiving and publishing them for wide access and sharing. In particular, 4CeeD consists of: (i) a curation service for collecting data from experimental instruments, curating, and wrapping of data with extensive metadata in real-time and in a trusted manner, (ii) a cloudlet for caching collected data from curation service and coordinating data transfer with the back-end, and (iii) a cloud-based coordination service for storing data, extracting meta-data, analyzing and finding correlations among the data. Our evaluation results show that our proposed approach is able to help researchers significantly save time and cost spent on experiments, and is efficient in dealing with high-volume and fast-changing workload of heterogeneous types of experimental data.**

I. Introduction

Studies suggest that it typically takes 20 years to go from the discovery of new materials to fabrication of new and next-generation devices based on the new materials [1]. This cycle must be shortened, and it will require a major transformation in how we collect digital data about materials and how we make the digital data available to computational tools for developing new materials and fabricating new devices and to the research community as a whole.

However, the development of computational tools for materials engineering has lagged behind the development of such tools in other engineering fields because of the complexity and sheer variety of materials and physical phenomena that must be captured [2][3]. In particular, the current state of data capture and storage in materials and semiconductor domains often involves a lot of manual processes that leads to poor documentation of results. For example, data transfer between research lab and office is often done via "sneaker-net" techniques using flash-drives or email. During such process, no data file conversion is available, which hinders researchers from previewing the results early and making timely decision during lab sessions. In addition, it is common that only "best" results and images are kept for publishing, although what is "best" is determined by a narrow, specific scientific objective. "Imperfect" data and/or data of secondary importance to the researchers recording the data, perhaps containing vital information that could accelerate the use of a new material for a device application, may simply never be captured (or only be logged manually in an unsearchable way) for fellow researchers to leverage or for a device engineer to search for. As a result, other researchers may end up repeating the very sample experiments over and over to retrieve these results.

While other related efforts, such as NanoHub [6], SEAD [8], or DataUp [12], have been focusing on making existing datasets more accessible and shareable, our focus and approach in this paper shifts to the immense potential of capturing, accurately curating, correlating, and coordinating materials-to-devices digital data in a real-time and trusted manner before fully archiving and publishing them for wide access and sharing. In particular, we develop a data acquisition and analysis framework for materials-to-devices processes, named 4CeeD. To the best of our knowledge, we are the first to build such a framework in the space of materials science and device fabrication to cut the time and cost of the materials-to-devices processes.

In this paper, we present the overview of our proposed multi-tier 4CeeD framework, and the design and implementation of its main components. Specifically, at the user tier, the 4CeeD's curation service will perform nimble and adaptive data collection from experimental instruments, data curation, and wrapping of data with extensive metadata in real-time and in a trusted manner. At the intermediate tier, cloudlet can cache the data from curation service, and coordinate data transfer with the cloud-based back-end system. At the cloud tier, the 4CeeD's coordination service will filter data, perform extraction of meta-data, analyze and find correlations among the data to identify dependency relations between materials and device fabrication processes.

We have implemented and deployed the 4CeeD system at the University of Illinois at Urbana-Champaign (UIUC) with test users from two of UIUC's major research labs that share research facilities: Micro and Nano Technology Laboratory (MNTL) and Materials Research Laboratory (MRL). The primary feedback from the test users indicate that the system has helped them significantly reduce time and cost spent on experiments. In addition, our evaluation on the

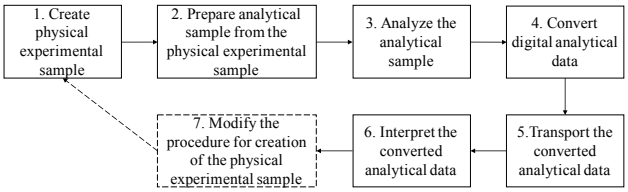


Figure 1: Experiment flow for material research.

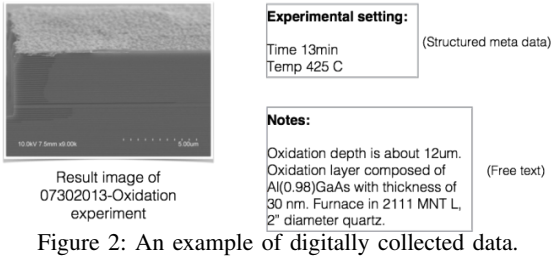


Figure 2: An example of digitally collected data.

system scalability show that the 4CeeD system is efficient in dealing with high-volume and fast-changing workload of heterogeneous types of experimental data.

The paper is organized as follows. In Section II, we provide some background information about the target environment of the proposed system and some insights from our user study as the motivations for the proposed solution. We present the architectural overview of the 4CeeD system in Section III. In Section IV and V, we respectively describe the design and implementation of curation and coordination services – the two main components of the 4CeeD system. We show some evaluation results in Section VI and related work in Section VII. We conclude the paper and present some future directions in Section VIII.

II. Background & User Study

To better understand the target environment of our proposed solution, we provide in this section some background information on the materials, semiconductor experiments, and analytical instruments used in Materials Science research. In addition, we present some insights from our user study to shed light on the user requirements and expectations.

A. Background

Figure 1 shows a typical experiment flow in material research. In the first step, researchers create physical experimental samples, either in their labs or in shared fabrication facilities. These physical samples can range from microelectronic devices, to biological samples, to nanoparticles. Once physical samples are created, they must be prepared for analysis (Step 2). For example, with analysis using Scanning Electron Microscopes (SEM), the preparation usually involves cutting the sample into a size which can be placed under the microscope and attaching it to a SEM sample holder. The result of such preparation is called analytical sample. The actual analysis of analytical sample happens using analytic tools (Step 3), including SEM and other electron, scanning probe, or optical microscopies could be performed, as well as x-ray, ion, electron, and optical scattering experiments.

The results of the analysis in Step 3 are the digital footprints of the physical experimental samples. These digital data can vary in format, depending on the type of analytic instrument used. For example, the output of SEM microscopes (Figure 2) consist of: (i) digital images of analytical sample that are stored in standard image format (e.g., .TIF, .GIF., or .JPEG), (ii) instrument specific information and meta-data (e.g., temperature, pressure, accelerating voltage, detector used, etc.) that are stored in a text file, and (iii) unstructured notes by researchers about the experimental or analytical results. On the other hand, output data from the TEM microscopes is in proprietary data format (i.e., DM3) that contains both image data and instrument specific meta-data. In such the case of proprietary data format, it might require another step to convert the results of analytic tools (Step 4). The researchers must then transport the converted files to their personal workstation (Step 5 - which often uses a "sneakernet" of USB thumb drives) for follow-up interpretation (Step 6). If the interpretation result is negative, further modifications might be needed for the procedure to create physical experimental sample (Step 7), which causes repetitions of the process until the desired criteria are satisfied.

While new analytic techniques have allowed for a surge of nanomaterials research publications and related innovative products, the time between discovery of new materials and application in semiconductor fabrication processes is at a relative stagnation, taking several years between an incepted material design and its commercial usage. This slow process can be attributed largely in part to communication of research, or rather the lack-there-of, specifically pertaining to nanomaterial analysis tools. Most often negative results from these nanomaterial analysis tools are not published, the transportation of the collected data is often insecure, and the resulting data files are often propriety causing inherent loss of data through file conversion in order to work up the data for publication quality figures.

In order to accelerate the experimental process, it is necessary to have an expedient mean to capture and process the digital data (i.e., output of Step 3) in real-time and in trusted manner before archiving, further analysis and visualization for more efficient interpretation of the results. Such a distributed real-time and trusted framework would greatly reduce the time, security and data loss risks of the manual efforts involved in the Step 4, 5, and 6 of the experimental process. In addition, a networked platform that provides authorized access to archived experimental data would help close the communication gap between researchers and prevent unnecessary repetitions of the experimental process caused by the lack of information in the

literature.

B. User Study

To further verify the necessity and practicality of such framework, we undertake a user study by surveying 52 users of MNTL and MRL labs in the University of Illinois.

The results from the survey show that the majority of users utilizes a "sneakernet" method to transport data from the lab. Specifically, 96% of the users use USB thumb drive to transport data from the experimental session to their office for further analysis, and 66% of them feel they have enough time during the session to upload the data if such a data acquisition tool exists.

While survey results encouragingly show that nearly 80% of users are interested in using such a framework for data acquisition, analysis, and a distributed platform for archiving and sharing data, they also point out some challenges in building such a framework. First, the scale of data generated during lab session tends to be different from user to user, as shown in Figure 3a. In addition to the large number of users from multiple research labs who might work concurrently during peak hours, the system infrastructure should be scalable and capable of dealing with varying workloads. Second, researchers use a wide variety of instruments for their experiments. While Figure 3b shows the most popular instruments (i.e., SEM and TEM), the long tail (i.e., "Other") consists of a very diverse set of instruments. As a result, the framework should be designed to support analyzing heterogeneous types of data generated from different types of instruments. On the other hand, by knowing the most popular types of instruments being used, we can put more focus on those types in designing evaluation and targeting potential users. Third, as digital data is collected for wide access and sharing, users might want to perform search over shared repository of experimental data. The objectives can be to update/correct any missing meta-data/setting, erroneous information from user's uploaded experimental data, to learn from others' successful or failed experiments, or simply to look for related experiments for reference purposes. Our survey shows that users want to search over the data using a variety of structured information, such as instrument types, experimental types and settings, beside traditional keyword-based search (Figure 3c).

III. 4CeeD Architecture Overview

In this section, we present the architectural overview of our proposed 4CeeD system for real-time capture, curation, coordination, collaboration, and distribution of scientific material-related data.

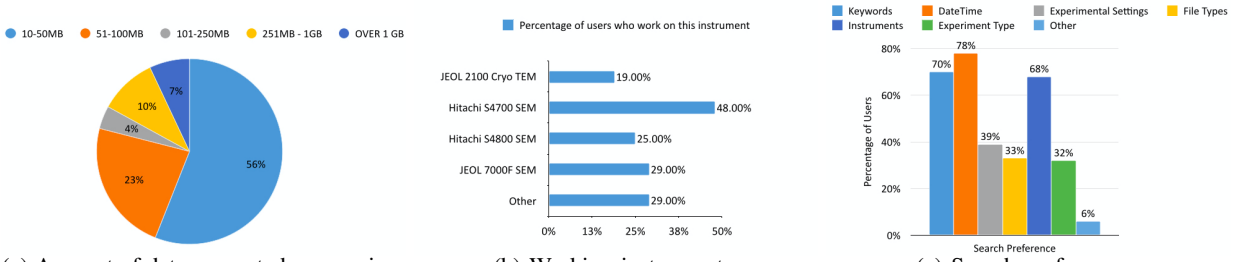
From the user study and survey results, we learned that one of the major challenges for 4CeeD is to be able to capture the data as it is generated in real-time during the experimental session, transfer it to the cloud-based system for curation, archiving, analysis, and after-session viewing, editing, or sharing. In addition, the cloud-based system needs to be scalable, to deal with large amount of input data, and be flexible, to support processing heterogeneous types of experimental data. To address these challenges, we propose a 3-tier architecture of the 4CeeD system (Figure 4) that consists of three services: curation, cloudlet, and coordination.

Curation service (instrument and user tier): The data curation service consists of two user-facing components: uploader and curator. With *uploader*, users can upload raw data generated from materials-making/characterization instruments (e.g., a microscopies) and device fabrication instruments via a user interface during experimental session. We will assume user-in-the-loop concept during the data input step, because all of the materials and device fabrication instruments are supervised and controlled by experimenting users. Often, we want users to enter process-related data, notes regarding experimentation with new materials, reasoning on why a certain physical component was added or removed, and so forth. After the experimental session, using *curator* tool, users will also have the capability to annotate, add tags, remove erroneous data captured because of wrong instrument settings and/or configurations. We will describe the design and implementation of curation components in details in Section IV.

Cloudlet (intermediate tier): Since concurrently streaming data from uploaders directly to the cloud might cause traffic congestion and overload for cloud tier during peak hours, we propose to have an intermediate caching edge server, called cloudlet, deployed at each research lab. Cloudlet will coordinate with the coordination tier (to be described) to schedule data transfer and perform certain processing tasks on the data, in order to avoid traffic congestion, reduce unnecessary data to be sent to the coordinator, and offload computation from the cloud. In addition, cloudlet is particularly important if scientific instruments are connected to PCs running old and unsupported OS (e.g., Windows XP) whose software cannot be patched with important security protections. In case the scientific instruments are connected to secure and updated PCs, and there is only a small number of instruments from a lab get connected to the back-end cloud, the cloudlet layer is optional.

Coordination service (cloud tier): This is the centralized cloud infrastructure for storing and processing collected data. To support heterogeneous types of workflow-based data processing tasks, we design 4CeeDs coordination service based on topic-based publish/subscribe model. In addition, to support scalability, we design a dynamic resource management mechanism based on explicitly modeling performance of the cloud-based pub/sub system. We will describe the design and implementation of the coordination tier in details in Section V.

For the first-phase development of our proposed architecture, we have designed and implemented the curation and



(a) Amount of data generated per session. (b) Working instruments. (c) Search preferences.

Figure 3: Survey responses on the data size and experimental instruments.

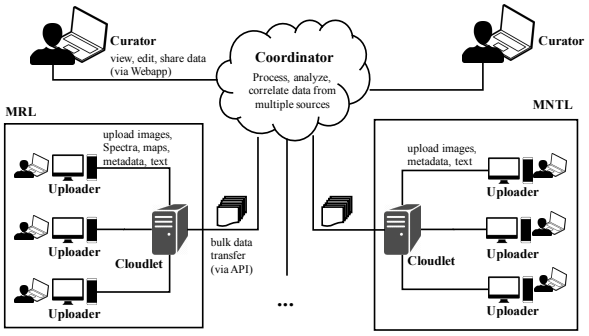


Figure 4: Overview of 4CeeD system.

coordination tiers that communicate directly with each other. We leave the design of cloudlet and its interaction with other tiers as a future work.

IV. 4CeeD's Curation Service

A. Requirements and Design Methodology

Since the primary purpose of data curation service (uploader in particular) is to help users save time at the microscopes by easily uploading raw experimental data to the cloud repository, such a tool should require minimal interactions with users. In addition, since the targeted users are non-IT people, the interface should be intuitive and simple to use.

Another requirement for curation service is to support various types of input from different types of experiments and by different users. Our discussions with users show that users tend to have different ways to organize their experimental data. For example, one might organize his/her data by experiment dates, while the other might use instruments or types of experiments. As a result, the curation service should provide an extendable data model for inputting data to support diverse use cases.

As user study suggested, the curation service should also support users the ability to search through shared data repository by efficient filtering of structured and meta data.

In terms of the compatibility, since the PCs attached to instruments can run different versions of operating systems (including older, unsupported OSes like Windows XP), the curation tools should be platform independent.

B. Implementation

To realize the above requirements, we first propose an extendable data model by expanding the one used by Clowder[1] to support nested structures necessary to mimic hierarchical folders which represent step by step scientific experiments and device fabrication processes. Specifically, the data model hierarchy includes three main concepts: nested collection, datasets, and files. At the lowest level, files represent experimental result data, such as images, text, or proprietary files. A dataset is a grouping of files that have metadata capturing the preparation information of the experimental sample. A collection is a way for users to organize their datasets (e.g., each collection represents experiment data for a day, or done by a particular instrument). The nested structure of collections provides users the flexibility to describe their own data organization.

The uploader is implemented as web-based app (hence to be platform-independent) with its interface divided into 3 simple dependent steps following the nested data model (Figure 5). The uploader requires users to log-in using their own credentials every time using the application. In the first step, user creates or selects a collection or sub-collection from his/her own set of nested collections visualized by a tree-based structure. After selecting (or creating) a collection, in step two, user can create or select a dataset. Beside having users manually enter meta-data, we support metadata templates (i.e., each template correspond to a collection of meta-data fields) to allow users to quickly and accurately record any metadata associated with the dataset. In the third step, users can drag and drop multiple raw experimental files to the dataset selected/created in Step 2 to submit. Additional file-level metadata can also be added in the third step.

Similarly, the web-based curator also provides users access to their data via nested data model in Figure 6a. In particular, users can browse, view, edit their uploaded experimental data by collections, datasets, or files. Users can see all data processing tasks done on the raw experimental

[1]Clowder - https://clowder.ncsa.illinois.edu

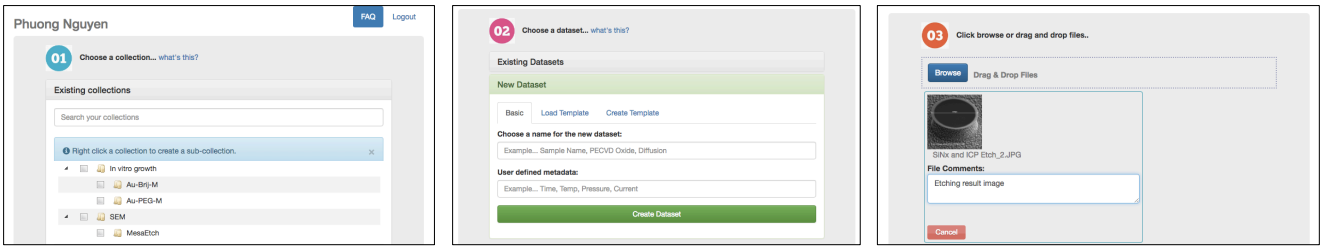
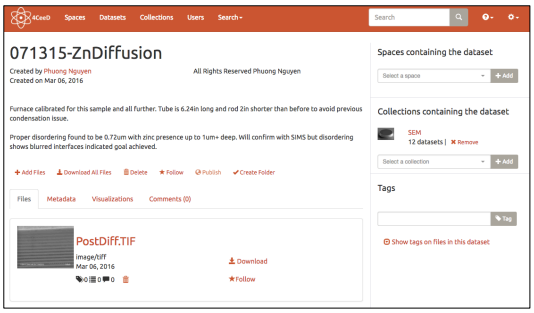
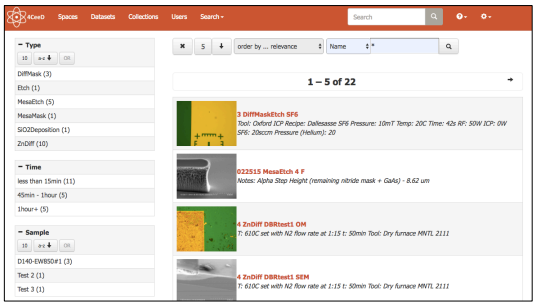


Figure 5: 4CeeD uploader's simple 3-step interface.



(a) View/edit uploaded experimental data.



(b) Faceted search across shared data repository.

Figure 6: 4CeeD's curator interfaces.

data. Examples of the tasks include extracting instrument-specific meta-data and image from DM3 file, generating thumbnail previews for microscopy images, and classifying experimental data into appropriate types (e.g., diffusion, oxidation, etc.) or outcomes (i.e., success or failure). Each type of experimental data requires different set of data processing tasks to be applied, which can be configured on the coordination service (to describe in Section V). Each set of tasks are often in form of workflow or Directed Acyclic Graph (DAG) of tasks, and corresponds to a type of *a data processing job*. In addition, we provide an "e-commerce style" search (i.e., similar to that of e-commerce sites like Amazon, Newegg) over shared data repository of experiments (Figure 6b). Users can easily and efficiently search through a large amount of experimental data by combining traditional keyword-based search and structured data filtering (or faceted search). The structured data used in filtering can be instrument-specific meta-data, experiment-related settings, or analytical results of data processing tasks (e.g., success/failure classification).

V. 4CeeD's Coordination Service

A. Requirements and Design Methodology

Since the experimental data uploaded to coordination service can be of various types and formats, one of the main requirements for 4CeeD's coordination service is to able to support processing heterogeneous types of data processing jobs, each corresponds to a type of uploaded data. To deal with job heterogeneity, we design the coordination's architecture based on topic-based publish/subscribe model that supports execution of heterogeneous workflows. We leverage the message passing mechanism in pub/sub system, in which different components in the system can post events (in form of messages) and react to those posted by other components, to give applications the flexibility to decide the logic of how to react to events and what chain of the steps needed to process an event. Our proposed design separates *control plane*, which manages resources, user permissions, and all the execution logic of workflows (i.e., task dependencies), and *compute plane*, which focuses on actual processing of workflow's tasks, and thus, allows scalable and flexible implementation of heterogeneous workflows/jobs. We describe our proposed architecture in details in Section V-B.

Since a large number of users might work concurrently during the peak hours, the coordination service needs to be scalable. In addition, since real-world applications often have variable and sometimes bursty loads, static and rule-based resource provisioning strategies are not suitable. To address these challenges, we leverage the elasticity of coordination service's cloud infrastructure to design a dynamic resource management approach for the pub/sub-based coordination. Our proposed solution also supports different resource provisioning strategies to satisfy different objectives set by users, such as quality of service (e.g., response time, utilization), or cost of resources. We describe in details our proposed resource management approach in Section V-C.

B. Publish Subscribe-based Coordination System

Our proposed system architecture for the pub/sub-based coordination service is presented in Figure 7. The system consists of three main components: front-end, control plane, and compute plane.

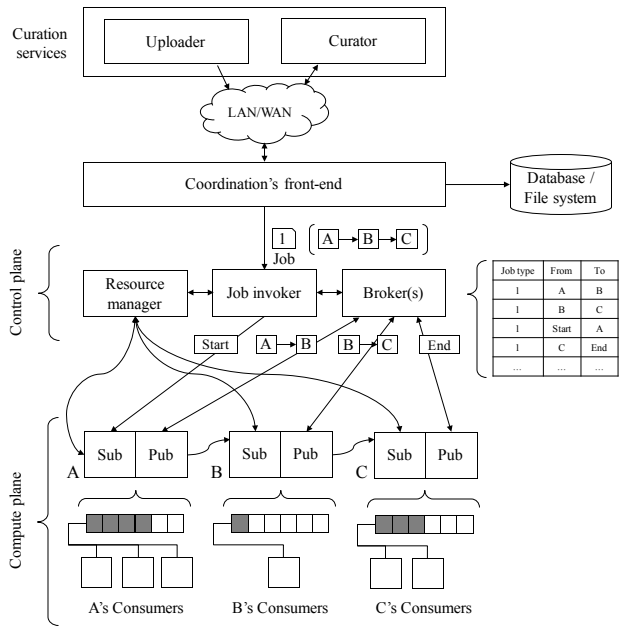


Figure 7: Cloud-based pub/sub system architecture.

Front-end is the cloud entry point for all incoming requests (e.g., requests for processing raw experimental data uploaded by users, or curation requests). Any input data that come with the requests are stored into a database or file system (which will then be accessible when requests are processed). Front-end translates the incoming request into a *job profile* that includes time-stamp, job ID, job type, user information, and any references to its input data stored in database/file system.

Control plane manages resources and handles all the execution logic of jobs. When the *Job invoker* receives the job profile from front-end, it first checks to see if the user who submitted the request has appropriate permission on the data that the job accesses to. Since each job type corresponds to a workflow of tasks, job invoker will ask the *brokers* which task of the job should be processed first. The brokers maintain a mapping table that includes all the task dependencies of all the job types that system supports. Particularly, given a job type and a current task (i.e., "From" field), the brokers will return what is the next task (i.e., "To" field) to be processed for a job. The Job invoker forward the job profile to the appropriate component in compute plane where the first task of the job will be processed.

Another main component of the control plane is *resource manger* that monitors the performance of processing components in the compute plane to make resource provisioning decisions. We describe the design and implementation of resource manager in details in Section V-C.

Compute plane is in charge of actual processing of tasks of a job. It consists of a "peer-to-peer"-like network of processing components, which are commonly abstracted as *topics* in pub/sub system, each is responsible for processing a particular type of task (e.g., extracting meta-data from DM3, generating previews of experimental images, classifying experimental data). Each topic operates both as a subscriber and a publisher. As a *subscriber*, a topic maintains a *message queue* for requests of the task type it is in charge for and a set of *consumers* that subscribes to the queue to process the requests (the number of consumers per topic can be adjusted dynamically and is the topic of study for resource management). Each consumer of a topic also acts as a *publisher*. After a task is processed, the consumer will ask control plane's brokers for the subsequent task(s) of the job and then, forward the job request to the appropriate topic(s).

In the Figure 7 shows an example of a flow of a job through the system. The request of a type-1 job consists of three tasks $A \rightarrow B \rightarrow C$ that are executed consecutively. After verifying the permissions of user who submitted the request, job invoker asks the brokers and send the request to the appropriate topic in charge of the first task of the job (i.e., task A). The job request is processed by one of the consumers of topic A, and then, after the A's consumer consults the brokers, the request is forwarded to the next topic (i.e., topic B). Similar procedure applies for the transition from task B to C. The processing of the job request ends when task C's consumer is notified that task C is the last task of a type-1 job.

C. Coordination Service's Resource Manager

In the following, we will describe our proposed design and implementation of an efficient resource manager for 4CeeD's coordination service.

Resource manager, which is part of the control plane, collects various statistics of the system in real-time, such as job request arrival rates, actual job response time, and decides whether to perform rescheduling of resources based on monitoring information (e.g., when system's average response time is greater than a certain predefined threshold). If a rescheduling is needed, resource manager executes appropriate resource allocation algorithm and produces a new allocation of resources over topics (i.e., how many consumers are needed for each topic).

We use a 3-step approach to design the resource allocation algorithm for 4CeeD's coordination service:

- Step 1 (**Optimization**): Formulate resource management as optimization problems.
- Step 2 (**Modeling**): Formally model the performance metrics of the system (e.g., response time, utilization).
- Step 3 (**Solution**): Efficiently solve the optimization problems to find optimal resource allocation strategies.

In the following, for the paper's completeness, we will briefly describe each step. More details can be found in our previous work [9].

Before describing each step, we define some notations used in this section. Let us consider a cloud-based pub/sub system that consists of J topics (i.e., supports processing

J tasks) and accepts requests for N types of jobs, each job corresponds to a workflow of tasks supported by the pub/sub system. In terms of computational parameters, for each topic j ($1 \leq j \leq J$), there are m_j (uniform) consumers subscribe to its message queue. The numbers of consumers over topics $\mathbf{m} = (m_1, m_2, ..., m_J)$ (which can be dynamically provisioned by exploiting the elasticity of the cloud infrastructure) are the main variables to measure performance of the elastic pub/sub system.

The system performance metrics, we use *work-in-progress*, denoted as WIP, as the performance metric for time (since response time is linearly related to the number of job requests being in the system, by Little's law). Particularly, $WIP(\mathbf{m}) = \sum_{j=1}^{J} \nu_j L_j(m_j)$, with ν_j and $L_j(m_j)$ are respectively the value of a job request (assumed to be given) and the number of job requests in progress at topic j. In terms of the resource cost, the total resource cost depends on the number of provisioned consumers per topic and is define as $F(\mathbf{m}) = \sum_{j=1}^{J} F_j(m_j)$, where $F_j(m_j)$ is the cost of allocating m_j consumers at topic j.

Optimization. The resource management problem for cloud-based pub/sub system can be formulated as optimization problems using different objective functions to allow flexible selection of resource provisioning strategies. In particular, for the first optimization problem, the objective is to minimize system's overall response time, or appropriately the WIP metric:

Problem Definition 1: (Minimal Time Resource Allocation) Given a cloud-based pub/sub system that supports N types of job and J different tasks, and a cost budget $\mathcal{M}$, find an optimal allocation $\mathbf{m}$ of consumers to topics to minimize system's work-in-progress WIP:

$$\underset{\mathbf{m}}{\text{argmin}} \quad WIP(\mathbf{m}) = \sum_{j=1}^{J} \nu_j L_j(m_j)$$

$$\text{subject to} \quad \sum_{j=1}^{J} F_j(m_j) = \mathcal{M}$$

For the second optimization problem, the objective is to minimize the total resource cost of allocating consumers across topics:

Problem Definition 2: (Minimal Cost Resource Allocation) Given a cloud-based pub/sub system that supports N types of job and J different tasks, and a WIP (or time) constraint $\mathcal{T}$, find an optimal allocation $\mathbf{m}$ of consumers to topics to minimize system's total resource cost $F(\mathbf{m})$:

$$\underset{\mathbf{m}}{\text{argmin}} \quad F(\mathbf{m}) = \sum_{j=1}^{J} F_j(m_j)$$

$$\text{subject to} \quad \sum_{j=1}^{J} \nu_j L_j(m_j) \leq \mathcal{T}$$

Modeling. In order to solve the above problems, it is important to obtain the formulation for the performance metric WIP. In particular, from the system architecture description in Section V-B, it is intuitive to model each topic as a *network of queues*, each queue corresponds to a topic's message queue. Besides, as job requests can be of different job types and they arrive then leave the system as they are finished, the queuing network model of the system is categorized as *multi-class* and *open*. To make the system model more realistic, we consider job request arrival rates and processing time at each topic both follow general distributions. As a result, we can model each topic as a *GI/G/m* queue and the elastic pub/sub system as a *Generalized Multiple-class Jackson OQN*.

With this modeling, we are able to obtain the approximation of performance measure of individual topic $L_j(m_j)$, and of the whole system WIP as $WIP(\mathbf{m}) = \sum_{j=1}^{J} \nu_j L_j(m_j)$.

Solution. Dynamic resource allocation for the system require more efficient solutions for Problem 1 and 2. By realizing the convex property of the objective functions (i.e., WIP in particular), we propose greedy strategies that not only efficiently solve the optimization problems, but also provide the optimal solutions. In particular, the Algorithm 1 starts with initializing each topic with one consumer, and then, the algorithm greedily finds the topic with the largest *benefit* if being allocated an additional consumer. For Problem 1, the allocation benefit is defined to be proportional to the decrease of the number of work-in-progress job requests (i.e., $\nu_j[L_j(m_j^{i-1}) - L_j(m_j^{i-1} + 1)]$). For Problem 2, the benefit is defined to be inversely proportional to the increase in resource cost (i.e., $F_j(m_j^{i-1}+1) - F_j(m_j^{i-1})$) and directly proportional to the decrease of the number of work-in-progress job requests (i.e., $\nu_j[L_j(m_j^{i-1}) - L_j(m_j^{i-1} + 1)]$). The greedy algorithm ends when it reaches the optimization constraint (cost budget $\mathcal{M}$ in Problem 1, and response time constraint $\mathcal{T}$ in Problem 2).

Algorithm 1 Greedy Resource Allocation

```
1: procedure GREEDYRESALLOC
2:     Initial allocation m^0: m_j^0 = 1, ∀1 ≤ j ≤ J
3:     Initialize iteration count i = 1
4:     while The optimization constraint is satisfied do
5:         Find the topic j* that maximizes the allocation benefit
6:         Add one consumer to most benefit topic m_{j*}^i = m_{j*}^{i-1} + 1
7:         Update iteration count i = i + 1
8:     Return allocation solution m^i
```

VI. Evaluation

A. 4CeeD Implementation Details

We implemented 4CeeD's uploader as a lightweight Web application using PHP programming language. Curators communicate directly with 4CeeD's cloud-based coordination service via front-end APIs, which are based on Clowder's API implementation. 4CeeD's curator is implemented based on Clowder with added features including nested collections data model, structured data-based search (or

faceted search). We use Elasticsearch[2] as the search indexing server for faceted search.

We implemented 4CeeD coordination service's proposed cloud-based elastic pub/sub system using RabbitMQ[3] as the message queue engine and Docker[4] container technology as the implementation platform for consumers (for better isolation and server consolidation). Particularly, each consumer is implemented and encapsulated into a Docker image and subscribes to a RabbitMQ's message queue of appropriate topic. We deployed coordinator on a cluster of three servers, each server is equipped with an Intel Xeon quad core processor (1.2Ghz for each core) and 16GB of RAM. We use Kubernetes[5] as the Docker container ostrastration engine for the cluster and each topic's consumer set is abstracted as a Kubernetes' ReplicationController. The resource manager (resource allocator in particular) interacts directly with Kubernetes to dynamically scale the size of ReplicationController (i.e., number of consumers) of each topic. All system components are implemented using Python programming language.

B. Curator

To evaluate curator, we ask our beta testers for their opinion about the tool after a few months of use. In particular, we ask users about how easy it is to use the tool and how much time would they be able to save using the curator during experimental sessions.

In terms of the ease of use, the user statements show that "the curator application is simple", "the steps in usage are pretty clear", and tools "allows them to utilize their preferred organizational strategy" without any instructions.

In terms of the time saving, an average SEM user states that every time using the SEM, he spend about 15% of the time exporting and transferring the images to a server. In addition to this time, he also need to spend another 15% of the time analyzing the images since he does not have a way to view the proprietary image format and I does not want to lose all the SEM metadata after exporting resulted file to a .TIF or .JPEG image format for after-session viewing. So, in total, for an hour long reservation, he loses about 15 to 20 minutes depending on the type of experiment. This time savings can also be translated into cost savings. Particularly, each hour in the clean room normally costs $15 and the SEM costs another $10 per hour. In addition to the labor cost, it can go up to $75 per hour. Therefore, the time spent on exporting and moving files would costs $25 to $30 each hour.

Being able to use the curator during experiment sessions allows users effectively save the time spent copying files and transport them to office for after-session interpretation. In addition, the previews of experimental files help users save time converting between different preview image formats. More importanly, since all metadata are also captured, users would know all configurations of the instruments (e.g., SEM camera settings at which the image was taken), and thus can easily try the same measurement again in the future.

[2]Elasticsearch - https://www.elastic.co

[3]RabbitMQ - https://www.rabbitmq.com

[4]Docker - https://www.docker.com

[5]Kubernetes - http://kubernetes.io

Task	Description	μ_j	cs^2_j
A	Unpacking digital microscope output files (e.g., DM3, HDF5)	4.2	0.33
B	Extracting and analyzing metadata from input file	3.7	0.5
C	Extracting and analyzing image from input file	6.7	0.4
D	Classify the input file into appropriate experiment type and predict if the experiment is successful or not	5.1	0.5

(a) Supported types of task.

Job type	Format	ca^2_i
1	A → B → D	0.33
2	A → B, A → C, B → D	0.5
3	C → D	0.25

(b) Supported types of job.

Figure 8: Tasks and jobs supported by the system.

C. Coordinator

In this section, we evaluate the effectiveness of our proposed coordinator compared to baseline in the two resource management tasks defined in Section V.

1) Evaluation Settings:

Case study: We take the application of executing scientific computing workflows as the case study. Particularly, the system supports analyzing experimental data generated by digital microscopes (which are usually in forms of DM3, or HDF5 files). Four types of task are supported, which correspond to the steps needed to process input data (Figure 8a). Depending on the input data, the system can support three different types of job, each job consists of all or a subset of supported tasks (Figure 8b).

Parameter settings: The processing rates of tasks are given in Figure 8a. The squared coefficient variance (scv) of job arrival rates are given in Figure 8b, while the expected arrival rates of each job type (i.e., λ_i) are varied during the evaluation to represent changing workload. Please note that the time unit we use for rates (i.e., processing time rate μ_j and job arrival rate λ_i) is *per minute*. To simplify the computation, we use a uniform resource cost function, i.e., $F_j(m_j) = m_j, \forall j$, and consider the job requests as equally important, i.e., $\nu_j = 1, \forall j$[6].

We compare our resource management algorithms, named `MinTime` (greedy algorithm for Problem 1) and `MinCost` (greedy algorithm for Problem 2), with random resource allocation approach, named `Random`. In `Random`, for each iteration, a topic is randomly chosen to be allocated an additional consumer. To evaluate the performance of different algorithms, we initially allocate one consumer to

[6]Please note that $F_j(m_j)$ and ν_j can be chosen in any form so that $WIP(\mathbf{m})$ and $F(\mathbf{m})$ maintain their non-increasing and non-decreasing convex properties.

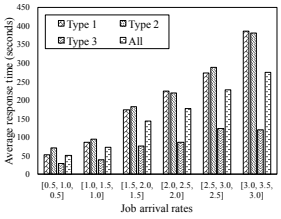
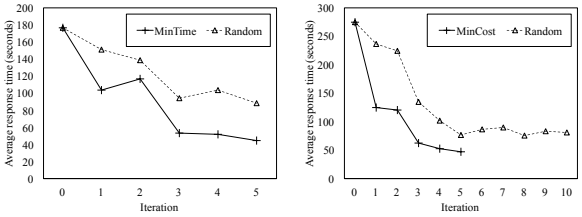


Figure 9: Average response time when incoming workload increase.



(a) Minimize time resource allocation comparison.

(b) Minimize cost resource allocation comparison.

Figure 10: Optimization tasks comparison.

each topic: $\mathbf{m} = (1, 1, 1, 1)$. Then, after each iteration (i.e., after a consumer is allocated to a topic), we measure the average response time of each type of job, as well as the average of all jobs. An algorithm is considered better if it achieves lower average response time after a given number of iterations (in case of minimal time allocation), or requires less iterations to reach a predefined response time threshold (in case of minimal cost allocation).

2) Varying workload: We first evaluate the performance of the pub/sub system by varying the input workload. In this experiment, we fix the number of consumer at each topic to be 1 (i.e., $\mathbf{m} = (1, 1, 1, 1)$) and increase the arrival rates of different job types. The results in Figure 9 show that, as expected, when the arrival rates increase, the average response time of the system (averaging over each individual job type as well as over all job types) increases. More importantly, in Figure 9, we also observe that the increases in the average response time of different job types are different. For example, job type 3 seems to be less affected by the increase of the arrival rates, compared with job type 1 and type 2. This suggests that, when provisioning the number of consumers at each topic, one should consider the differences in the sensitivity of different job types to the changing workload. This insight is also consistent with our motivation in designing the greedy resource allocation strategies (Section V), in which, we give higher provisioning priority to topic whose provisioning gives largest benefit.

3) Optimization Tasks: For the Minimal Time Resource Allocation task, given a workload $\{\lambda_i\} = (2.0, 2.5, 2.0)$ and a cost constraint $\mathcal{M} = 5$, we perform resource allocation using `MinTime` and `Random`. Figure 10a shows that, when two algorithms reach the cost constraint (i.e., after 5 iterations), `MinTime` outperforms `Random` by achieving a lower average response time of all types of job. On the other hand, `Random` could not achieve optimal result due to its randomization in selecting topics for provisioning.

Figure 11: Dynamic provisioning to deal with bursty workload.

For the Minimal Cost Resource Allocation task, given a workload $\{\lambda_i\} = (3.0, 3.5, 3.0)$ and a response time constraint of 50 seconds: $\mathcal{T} = 50$, we perform resource allocation using `MinCost` and `Random` until the system average response time of all types of job smaller than or equal $\mathcal{T}$. The result in Figure 10b shows that `MinCost` only needs 5 additional consumers to satisfy the response time constraint, while `Random` struggles in bringing down the response time to below $\mathcal{T}$, even after 10 iterations.

The results in both optimization tasks help confirm the effectiveness of using greedy strategy in selecting the topics for resource provisioning that maximize the overall benefit.

4) Efficient Dynamic Provisioning: We evaluate the efficiency of our proposed resource management solution when dealing with changing workload. Particularly, we simulate a bursty workload that consists of 100 job request for each type of job. The first 20% of the requests arrive with rates $\{\lambda_i\} = (0.5, 1.0, 0.5)$ and the remaining 80% of the requests abnormally arrive with rates multiple times higher $\{\lambda_i\} = (3.0, 3.5, 3.0)$. At the beginning of the test, each topic has one consumer: $\mathbf{m} = (1, 1, 1, 1)$. Our resource manager is configured to run during the test using `MinTime` algorithm and cost constraint $\mathcal{M} = 5$.

The response time statistics of all requests during the test period (Figure 11) show that the resource manager observes the increase in the average response time of the system and quickly provisions the resources, i.e., decide new allocation $\mathbf{m} = (2, 2, 1, 4)$, to bring the average response time back to the level before bursty load happens. The whole process from observing the increasing response time, generating new rescheduling strategy, to re-scaling the system is efficient, and thus, the bursty load only affects a small portion of requests (about 15% of requests) during a short amount of time.

VII. Related Work

The related efforts in scientific data management and cyberinfrastructure have been focusing on making existing datasets more accessible and shareable [6][8][12][13], and distributed cyberinfrastructure frameworks that incorporate cloud technologies [5][4][15]. On the other hand, our focus in this paper shifts to capturing, accurately curating, corre-

lating, and coordinating materials-to-devices digital data in a real-time and trusted manner **before** fully archiving and publishing them for wide access and sharing. As a result, our effort is complement to those other efforts, and we could effectively leverage existing solutions to solve our problems.

Publish-subscribe system [18][17], with its wide range of applications, has been a large topic of study. There have been a lot of efforts recently [7][10][16] to deploy pub/sub system in the cloud environment to take advantage of the elasticity of the cloud. For example, Gascon et al [10] propose a cloud resource provisioning strategy for pub/sub system based on monitoring the incoming workload, Setty et al. [11] study the resource cost-effective deployment problem of pub/sub system with known workload. Our proposed cloud-based 4CeeD's coordination service in this paper leverages pub/sub model to support scalable execution of heterogeneous workflows.

Most of the efforts on resource management for cloud-based systems have been on batch processing [19], interactive big data analytics systems [20], or synchronous data stream processing [14]. In our previous work [9], we focus on resource management for real-time asynchronous pub/sub system that can support multiple types of jobs. Our proposed resource allocation strategies can be used with other more generic cloud resource management solutions, such as YARN [22] and Mesos [21] that help allocate available computational resources to applications.

VIII. Conclusions and Future Work

In conclusions, in this paper, we have presented design and implementation of 4CeeD, a data acquisition and analysis framework for materials-to-devices processes. 4CeeD supports capturing, curating, correlating, and coordinating materials-to-devices digital data in a real-time and trusted manner before fully archiving and publishing data. The evaluation results show that our curation service helps users speed up about a third of the time spent at digital microscopes, and avoid using widely popular "sneakernet" method to transport data that limits data capacity and poses security concerns. At cloud level, 4CeeD's coordination service supports scalable execution of heterogeneous workflows, where tasks can be written by users and plugged-in to the coordination system via containerization.

In terms of future work, we would like to incorporate cloudlet into the current implementation to help orchestrate data transfer between multiple sides and the cloud to avoid traffic congestion. We will also investigate how to perform off-load computational tasks from the cloud to cloudlet to support applications that require low-latency and fast responses, as well as to prevent unnecessary data transferred to the cloud. In addition, we would also like to increase the number of instruments supported by 4Ceed framework and expand 4CeeD's user base.

Acknowledgment

This research was funded by the National Science Foundation NSF ACI 1443013. The opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the view of the National Science Foundation.

References

[1] Holdren, J.P. *Materials genome initiative for global competitiveness*. National Science and Technology Council OSTP 2011, Washington, USA.

[2] Apelian, D. et al. *Accelerating technology transition: bridging the valley of death for materials and processes in defense systems*. National Materials Advisory Board, NAE, 2004.

[3] Oden, J.T. et al. *Simulation-Based Engineering Science: Revolutionizing Engineering Science through Simulation*. Report of the National Science Foundation Blue Ribbon Panel on Simulation-Based Engineering Science, 2006.

[4] Padhy, S. et al. *Brown Dog: Leveraging everything towards autocuration*. In Big Data 2015.

[5] McLennan, M., Kennell, R. *HUBzero: a platform for dissemination and collaboration in computational science and engineering*. Computing in Science & Engineering 2010, 12(2), pp.48-53.

[6] Klimeck, G. et al. *nanohub.org: Advancing education and research in nanotechnology*. Computing in Science & Engineering 2008, 10(5), pp.17-23.

[7] Li, M., Fan, Y., Kim, M. K., Chen, H., and Lei, H.. *A scalable and elastic publish/subscribe service*. In Proceedings of IPDPS 2011, pp. 1254-1265.

[8] Plale, B. et al. *SEAD virtual archive: Building a federation of institutional repositories for long-term data preservation in sustainability science*. International Journal of Digital Curation 2013, 8(2), pp.172-180.

[9] Nguyen, P. and Nahrstedt, K. *Resource Management for Elastic Publish Subscribe Systems: A Performance Modeling-based Approach*. In CLOUD 2016.

[10] Gascon-Samson, J. et al. *Dynamoth: A Scalable Pub/Sub Middleware for Latency-Constrained Applications in the Cloud*. In Proceedings of ICDCS 2015, pp. 486-496.

[11] Setty, V. et al. *Cost-effective resource allocation for deploying pub/sub on cloud*. In Proceedings of ICDCS 2014, pp. 555-566. IEEE, 2014.

[12] Strasser, C. et al. *DataUp: A tool to help researchers describe and share tabular data*. In F1000Research, 3.

[13] Szalay, A.S. et al. *The SDSS skyserver: public access to the sloan digital sky server data*. In SIGMOD 2002.

[14] Fu, T. Z. J. et al. *DRS: Dynamic Resource Scheduling for Real-Time Analytics over Fast Streams*. In Proceedings of ICDCS 2015.

[15] Mayernik, M. et al. *The data conservancy instance: Infrastructure and organizational services for research data curation.* D-Lib Magazine 2012, 18(9), p.2.

[16] Tran, N.-L., Skhiri, S., and Zimnyi, E. *Eqs: An elastic and scalable message queue for the cloud*. In Proc. of CloudCom 201, pp. 391-398.

[17] Jacobsen, H.A. et al. *The PADRES Publish/Subscribe System*. Principles and Applications of Distributed Event-Based Systems 2010, 164, p.205.

[18] Eugster, P.T., Felber, P.A., Guerraoui, R. and Kermarrec, A.M., 2003. The many faces of publish/subscribe. *ACM Computing Surveys (CSUR)*, 35(2), pp.114-131.

[19] Zaharia, M. et al. *Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling*. In Proceedings of the 5th European conference on Computer systems 2010, pp. 265-278.

[20] Melnik, S. et al. *Dremel: interactive analysis of web-scale datasets*. Proceedings of the VLDB Endowment 2010, 3(1-2), 330-339.

[21] Hindman, B. et al. *Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center*. In Proceedings of NSDI 2011, Vol. 11, pp. 22-22.

[22] Vavilapalli, V.K. et al. *Apache hadoop yarn: Yet another resource negotiator*. In Proceedings of the 4th annual Symposium on Cloud Computing 2013 (p. 5).