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 present 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 novel services: a curation service for collecting data from microscopes and fabrication instruments, curating, and wrapping of data with extensive metadata in real-time and in a trusted manner, and a cloud-based coordination service for storing data, extracting meta-data, analyzing and finding correlations among the data. Our evaluation results show that our novel cloud framework can 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 [3]. 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.

Figure 1 shows the current state of data capture and storage in materials and semiconductor fabrication domains. In the first step, researchers create physical experimental samples, e.g., microelectronic devices, biological samples, or nanoparticles, in their labs or fabrication facilities. Once physical samples are created, they must be prepared into analytical sample for analysis (in 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. The actual analysis of analytical sample happens in Step 3 using analytical tools such as SEM, optical microscopies, or X-Ray. 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 analytical instrument used. For example, the SEM output (Figure 2) consists of digital images (in standard image format like

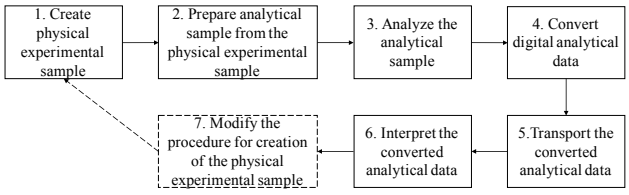


Figure 1: Experiment flow of material research.

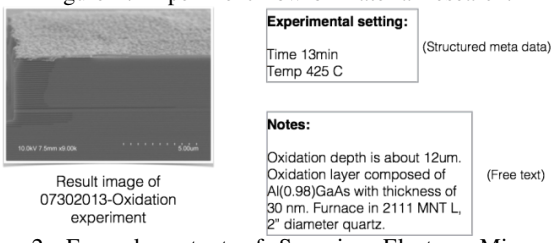


Figure 2: Example output of Scanning Electron Microscopes (SEM).

.TIF, .GIF., or .JPEG), instrument specific meta-data (e.g., temperature, pressure, accelerating voltage, detector used, etc.), and free-text notes by users; while the Transmission Electron Microscopy (TEM) output is in a proprietary format (i.e., DM3 file) that contains both image data and instrument specific meta-data. In case of proprietary data format, it might require another step to convert the results of analytical tools to formats that can be opened on researchers' computer (Step 4). The researchers must then transport the converted files to their personal workstation (Step 5) for the 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 repetition of the experiment flow until the desired material properties are satisfied.

There are several issues with the above process that contribute to the long cycle from discovery of new materials to fabrication of new devices. First, in terms of *data transfer*, transferring files between materials research lab and researchers' office is often done via "sneaker-net" techniques using flash-drives or emails. During such process, no data conversion is available, which hinders researchers from previewing the results early and making timely decision at the microscope during lab sessions. Second, in terms of *data management and processing*, researchers often store data in

their local hard drive or on cloud-based storage services, such as Box or Google Drive. However, these storage mechanisms do not provide any assistance in organizing and processing data to support efficient search, curation, coordination and management of data. Thus, it often takes a lot of manual effort from researchers to organize their data and log important information about the experiments as they are being interpreted and curated, which leads to poor documentation of results. Third, in terms of *data access and sharing*, despite their strong dependency, materials science and semiconductor fabrication areas have never been digitally connected and the only linkage until now is through published results in publications. However, it is common that only "best" results and data are kept for publishing, while "imperfect" data and/or data of secondary importance to the researchers recording the data, which perhaps contain vital information that could accelerate the use of a new material for a semiconductor device in the future, may simply be discarded or not be shared with fellow researchers to leverage or for a device engineer to search for. Thus, other researchers may end up repeating the very same experiments over and over again to obtain these results.

While other related efforts, such as NanoHub [6], SEAD [9], or DataUp [14], have been focusing on making existing datasets more accessible and shareable, our approach to accelerate the experimental process is to provide an expedient mean to capture, transfer, and process the digital data in real-time and in trusted manner before archiving, further analysis, visualization for more efficient interpretation and sharing of the experimental results. In this paper, we present the design and validation of our 4CeeD[1] framework and its main components. At the user tier, 4CeeD's curation service will perform nimble and adaptive data collection from instruments by wrapping of data with extensive meta-data in real-time and in a trusted manner. In addition, this service also provides advanced data management, curation, and sharing of the collected data after they have been processed by the back-end service. At the cloud tier, the 4CeeD's coordination service will filter data, perform extraction of meta-data from microscope images, analyze and find correlations among the data to identify new dependency relations between materials and device fabrication processes. With these services, 4CeeD aims to greatly reduce time, security and data loss risks of the manual efforts involved in transferring, storing, and managing data. In addition, by making archived experimental data available via authorized access, 4CeeD helps to close the communication gap between researchers and prevent unnecessary repetitions of the experimental process caused by the lack of information in the literature.

We have implemented and deployed the 4CeeD framework 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). Preliminary 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 system scalability shows that the 4CeeD system is efficient in dealing with high-volume and fast-changing workload of heterogeneous types of experimental data.

[1]4CeeD stands for **C**apture, **C**urate, **C**oordinate, **C**orrelate, and **D**istribute material-related experimental data.

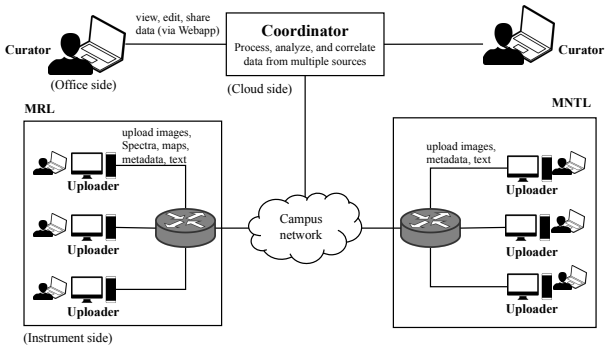


Figure 3: Overview of 4CeeD system.

In summary, our contributions are as follow:

- We present *the first* real-time data acquisition and analysis framework that helps digitally connect material science and semiconductor fabrication areas.
- We design and validate a novel curation service that enables nimble, comprehensive data collection and smart data management capabilities.
- We design and validate a novel scalable cloud-based system architecture that supports processing of heterogeneous multi-modal material-related data.
- We have implemented and deployed the system in a real environment with encouraging evaluation results.

The remaining of the paper is organized as follows. We present architectural overview of the 4CeeD system in Section II. In Section III and IV, we respectively describe the design and validation of curation and coordination services – the two main components of the 4CeeD system. We show some evaluation results in Section V and related work in Section VI. We conclude the paper and present some future directions in Section VII.

II. 4CeeD Architecture Overview

In this section, we present the architectural overview of the 4CeeD framework for real-time capture, curation, coordination, collaboration, and distribution of scientific material-related data. The framework consists of two main services: *curation* and *coordination* services (Figure 3).

Curation service (instrument and user tier): The data curation service consists of two components: uploader and curator. With *uploader*, we design a new simple, user-in-the-loop interface for uploading raw data generated from materials-making/characterization instruments (e.g., microscopes) and

device fabrication instruments during lab sessions. The user involvement in the data input step is because all 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. With *curator*, we design a novel interface that allows users to annotate, add tags, remove erroneous captured data (e.g., because of wrong instrument settings and/or configurations) after lab sessions. Underneath uploader and curator, we design an *extensible data model* for heterogeneous and multi-modal material-related data (i.e., combination of multimedia data like images, structured and unstructured data, text, tags, etc.). ***Coordination service (cloud tier):*** This is the centralized cloud infrastructure for storing and processing uploaded data. To support heterogeneous types of workflow-based data processing jobs, we design 4CeeDs coordination service based on *a novel micro-service based architecture* and leverage the topic-based publish/subscribe (or pub/sub for short) middleware to connect different services. To support scalability and handle variable and bursty workload, we design a *novel dynamic resource management* mechanism based on explicit performance modeling of the micro-service based system.

We will describe the design and validation of curation and coordination services in details in Section III and IV respectively. We will start with briefly discussing requirements for each service[2], and follow-up by our design and validation.

III. 4CeeD's Curation Service

A. Requirements

Since the primary purpose of data curation service 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 (**R1: *Intuitive and simple user interface***).

Another important requirement for curation service is to support various types of input data from different types of experiments and instruments. In addition, our discussions with users show that users tend to have different ways to organize their own data. For example, one might organize his/her data by experiment dates, while the other might use instruments or types of experiments. Thus, the curation service should provide an extendable and flexible data model for inputting data to support diverse data types and use cases (**R2: *Extendable and flexible data model***).

To support data discovery, the curation service should also support users the ability to search through shared data repository by efficient filtering of structured and meta data (**R3: *Efficient search over uploaded 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 (**R4: *Platform independent applications***).

[2]For more detailed discussion on requirements, please refer to our extensive user study and survey[2].

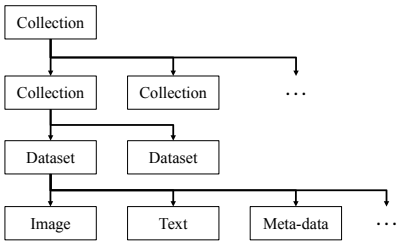


Figure 4: Data model.

B. Design of Data Management and Curation Service

Data Model: To realize the above requirements, we design a novel extendable data model for material- and semiconductor-related scientific experiments (Figure 4). The model is based on 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 collections*, *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 and 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 specific instrument). The nested structure of collections provides users the flexibility to describe their own data organization.

While the concepts of collections, datasets, and files provide a "vertical" organization of data, we use the concept of *spaces* for "horizontal" organization to support sharing of data. A space is a set of collections and datasets that are shared among multiple users. Different levels of user permissions (e.g., owner, editor, and viewer) can be configured for each space to enable flexible collaboration between users.

The uploaded data and associated metadata are stored in a document-oriented database (i.e., we use MongoDB). Our reasons for using document database is two-fold. First, the schema-free property of document-oriented database offers better support for heterogeneous types of experimental data uploaded by users. Second, different from relational databases that normally store data in separate tables (and thus data about a single experiment might spread across several tables), document databases store all information for a given experiment as a single object (i.e., document) in the database. Thus, this makes document databases a more desirable choice for storing material-related data, since

the uploaded experiment data are subjected to continuous change and curation by users after the data are acquired and stored in the database.

Uploader: The uploader provides an interface that consists of 3 simple dependent steps following the nested data model. In the first step, user creates or selects a collection or sub-collection from his/her own set of existing nested collections (stored on the cloud) visualized by a tree-based structure. After selecting (or creating) a collection, in step 2, user can create or select an existing dataset. Under dataset, users can manually enter meta-data associated with the experiment, or use provided meta-data templates (i.e., each template correspond to a collection of meta-data fields) for faster and more accurate recording of meta-data. In the third step, users can drag and drop multiple raw experimental files generated from the instruments to the dataset selected/created in step 2 to submit to the cloud. Additional file-level meta-data can also be added in the third step.

Curator: The curator provides users ability to browse, view, edit their uploaded data at the office side, using the nested data model. Especially, as the raw uploaded data has been processed by the coordination service, users can see results of all data processing tasks done on the raw data. Examples of the tasks include extracting instrument-specific meta-data and image from DM3 file, generating 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 a different set of data processing tasks, which are expressed in form of a *workflow* or Directed Acyclic Graph (DAG) of tasks, to be applied. Each workflow or task graph corresponds to a type of *a data processing job*[3], which can be configured on the coordination service (to describe in Section IV). In addition, within the curation service, we provide an "e-commerce style" search (i.e., similar to search feature on e-commerce sites like Amazon, Newegg) over shared data repository of experiments. 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 the results of data processing tasks (e.g., outcome classification).

Both uploader and curator can be accessed as web-based applications (hence are platform-independent) and both require authentication to access, curate and share data. Curator is developed based on the Clowder open-source project[4]. The communication between uploader, curator and the cloud-based system is via HTTPS protocol to ensure security.

[3]From here, we will use *workflow* and *job* interchangeably as each job type corresponds to a specific workflow.

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

IV. 4CeeD's Coordination Service

A. Requirements

As mentioned in the previous section, 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 heterogeneous types of data processing jobs, each job corresponds to a type of uploaded data and is modeled as a workflow or a DAG task graph (**R5: *Support heterogeneous workflows***).

The coordination service also needs to be scalable to support many users working concurrently during the peak hours. For example, the MRL lab in UIUC has more than 30 instruments that are used by hundreds of users and produce data of varied formats and sizes. In addition, as shown in our user and instrument study [2], the real workload generated from instruments in the labs is often variable and bursty (especially during experimental sessions). Thus, static and rule-based resource provisioning strategies are not suitable (**R6: *Scalability and dynamic provisioning***).

In the remaining of this section, we describe the main components of our 4CeeD architecture and the algorithms and design decisions to address the above challenges.

B. Micro-service based Coordination System Design

While the traditional workflow systems often have a monolithic design, which models workflows as tightly coupled set of tasks, and thus, is inefficient in dealing with heterogeneous workflows, we design the coordination service based on a micro service-based architecture. Specifically, each task is modeled as a micro-service, or processing component (PC for short), that handles requests for a type of task. We use topic-based publish/subscribe system as the middleware to connect different PCs. Specifically, we leverage the message passing mechanism in pub/sub system, in which different PCs in the system can post events (in form of messages) and react to those posted by other components, to enable the flexibility in designing the logic of how to react to events (e.g., job processing requests) and what chain of the steps, or workflow, needed to process an event.

The 4CeeD coordination service is presented in Figure 5. The system consists of three main components: front-end, control plane, and compute plane.

Front-end receives incoming requests from curation services (e.g., requests for processing raw experimental data from uploaders, or curating requests from curators) and store input data that come with the requests into a database or file system (which will then be accessible when requests are processed). Front-end translates the incoming request into a *job request* that includes arrival time, job ID, job type, user information, and any references to its input data stored in database.

Control plane manages resources and handles all the execution logic of jobs. The *Job invoker* analyzes incoming

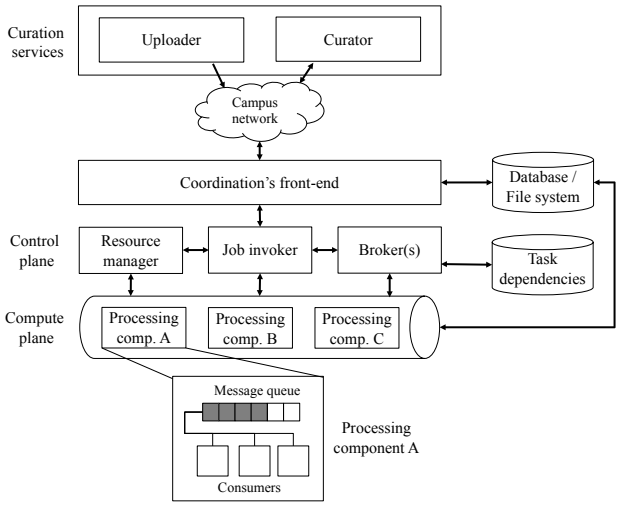


Figure 5: Micro-service based Coordination System.

Job type	From	To
1	A	B
1	B	C
1	Start	A
1	C	End
3	Start	C
3	D	End
...	...	...

Table I: Example of task dependency table.

job request and checks to see if the user who submitted the request has appropriate permission on the data that the job needs to access. 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 task dependency database that includes all dependencies between tasks of all the job types. Particularly, given a job type and a current task, the brokers will return what is the next task(s) to be processed. The Job invoker forward the job request to the appropriate PC(s) in compute plane where the first task(s) of the job will be processed.

Compute plane consists of a set of PCs, each handles requests of a type of tasks (e.g., extracting meta-data from DM3, generating previews of experimental images, classifying experimental data). The PCs are connected by a publish-subscribe middleware, in which, each PC corresponds to a *topic*[5] in pub/sub system and has its own *message queue* that stores requests for that topic. The actual processing work is done by a set of *consumers* that subscribe to the message queue of the PC. Besides being a subscriber, each consumer in a PC also acts as a *publisher*. After a consumer processes a task in a job request, it will ask brokers for information about the subsequent task(s) of the job and then, forward the job request to the appropriate PC(s).

Table I shows an example of a task dependency table that maintains the dependencies between tasks of job type 1 & 3. From these dependencies, given a request of job type 1, the job invoker will first send the request to the PC A (i.e., the first task of job type 1). After being processed by one of the consumers of PC A, the consumer will consult the brokers about the following task(s), and then, forward the request the next PC (in this case, 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 informed by the brokers that task C is the last task of a job type 1.

Our design separates control plane, which manages resources, user permissions, and all 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 (i.e., **R4**). In addition, our micro-service based architecture also enables continuous deployment of individual task (i.e., a task can be updated without affecting other tasks in the same workflow), and fine-grained resource scheduling at task level (to describe in the next section), and flexible workflow composition (i.e., since the workflow definitions are separated from the PCs, we can easily update a workflow without restarting its PCs).

C. Coordination Service's Resource Manager Design

The more consumers subscribe to the message queue of a PC, the more requests the PC can process in parallel. Hence, the processing capacity of the coordination service depends on configuration of the number of consumers across all PCs. However, deciding such configuration is non-trivial and often requires manual effort from users, which makes the system difficult to scale when dealing with variable and bursty workload. In addition, since each job corresponds to a workflow of multiple tasks and different job types can share common tasks, it is even more challenging to derive optimal configuration when dealing with heterogeneous types of jobs. For example, if we have job type 1 & 3 that share PC C (Table I), the provision of PC C should consider the fact that it has to process requests for both types of jobs. In addition, it is not clear how to balance the provisioning for PCs A and B of type 1. If we over-provision A & B, it could make the downstream PC C overloaded and negatively affect the response time of requests of job type 3 (since its upstream PC C becomes the bottleneck).

In the following, we will describe our design of an efficient resource manager for 4CeeD coordination service that dynamically scale the number of consumers for each PC to meet different objectives set by users, such as average job response time, total resource cost.

Let us assume that our 4CeeD coordination service supports processing J types of tasks and accepts requests for N types of jobs, each job corresponds to a workflow of tasks. For the PC associated with task type j ($1 \leq j \leq J$), there are m_j (uniform) consumers which subscribe to its message queue. The configuration of the numbers of consumers over

[5]From now on, we will use *processing component* (or PC), *topic*, and *task* interchangeably as they refer to the same notion.

all PCs is denoted as $\mathbf{m} = (m_1, m_2, ..., m_J)$. The resource manager collects various performance measurements of the system in real-time, such as job request arrival rates, job response time, task processing time, and decides whether to reconfigure the number consumers over PCs (i.e., updating $\mathbf{m}$) based on user defined constraints (e.g., when system's average response time is greater than a certain predefined threshold). If a reconfiguration is needed, resource manager will execute appropriate resource allocation algorithm and produces a new allocation of consumers over PCs.

The resource cost is measured as the total cost of allocating consumers over all PCs, denoted as $F(\mathbf{m})$. In this paper, we assume that the cost of allocating a consumer is the same for all tasks, and thus, the total resource cost is equivalent to the total number of consumers: $F(\mathbf{m}) = \sum_{j=1}^{J} m_j$.

To quantify job processing time, we use *work-in-progress*, denoted as WIP, that captures the total number of job requests currently in message queues of all PCs as the representative metric for response time. This is because, by the Little's law [1], the number of work-in-progress requests is proportional to the average time a request spends in the system, and thus proportional to response time. To formulate WIP, we leverage the result of our previous work [11] on performance modeling of elastic pub/sub systems. In particular, we model the compute plane as a network of queues with each queue corresponds to a task's message queue. If we assume general distribution of job request arrival rates to the system (denoted as $\{\Lambda_i\}, 1 \leq i \leq N$) and processing time at each PC (denoted as $\{M_j\}, 1 \leq j \leq J$), each PC becomes a *GI/G/m* queue and the compute plane becomes a *Generalized Multiple-class Jackson Open Queuing Network*. Thus, we can obtain the formulation of WIP of the j-th WIP_j [11], and the WIP of the whole system as $WIP(\mathbf{m}) = \sum_{j=1}^{J} WIP_j$.

However, as the compute plane is modeled as a set of independent micro-service based PCs, computing WIP_j requires decomposing $\{\Lambda_i\}(1 \leq i \leq N)$ to the aggregated arrival rate distribution of job requests at *each individual PC*: $\{\hat{\Lambda}_j\}, (1 \leq j \leq J)$. In this paper, we propose an algorithm, named `DECOMPOSE`, based on the parametric decomposition method, proposed by Vliet et al. [8]. The procedure is illustrated in Figure 6. It starts with loading and validating workflow descriptions, (which are stored in edge-list format) of jobs that system supports to make sure that they are valid DAGs. After that, it merges all workflows into an aggregated graph, and then sorts the vertices in the aggregated graph (each vertex corresponds to a type of task or PC) based on breadth-first search (BFS) ordering. Obtaining such an order of PCs can be done offline or periodically each time workflows are updated. Using this order, the procedure then performs parametric decomposition on each individual PC (i.e., the BFS order is to ensure that a PC is decomposed only after its precedent PCs have been decomposed). The parametric decomposition step is performed online as it takes

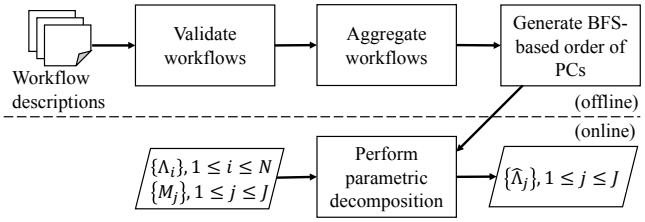


Figure 6: `DECOMPOSE` algorithm.

real-time job arrival rates & processing time distributions to compute the aggregated arrival rate distribution at each PC: $\{\hat{\Lambda}_j\}(1 \leq j \leq J)$.

After decomposition, WIP, from a function of $\{\hat{\Lambda}_j\}, \{M_j\}$, and $\mathbf{m}$, becomes the function of $\mathbf{m}$ only. Thus, the resource allocation problem becomes finding $\mathbf{m}$ that minimizes system's work-in-progress $WIP(\mathbf{m})$ while satisfying a cost constraint of the total resource cost $F(\mathbf{m})$.

To perform resource allocation **efficiently**, we propose a *greedy elastic scaling algorithm*, denoted as `GRESMAN`, to dynamically find $\mathbf{m}$. In particular, the Algorithm 1 starts with the current configuration of consumers over PCs, and then, greedily finds the PC with the largest *local benefit* if being allocated one additional consumer, denoted as $\Delta(m_j^i, m_j^i+1)$. The notion of local benefit of a PC is defined to be proportional to the decrease of the number of work-in-progress job requests of that PC. The most beneficial PC is added to a queue $\mathcal{A}$ that maintains an ordered list of PCs being provisioned. The reason of having a queue $\mathcal{A}$ is that order of allocation of consumers also affects system performance. As requests travel through the system following task dependencies, non-careful allocation order of consumers can cause bottleneck at a PC if it is allocated with more consumer after its precedent PCs. The algorithm ends when either system's WIP is under a threshold $\mathcal{T}$ (i.e., represent time constraint), or the total resource cost $F(\mathbf{m})$ reaches a certain budget $\mathcal{C}$.

Algorithm 1 Dynamic Greedy Elastic Scaling Algorithm (GRESMAN)

1: **procedure** `GRESMAN`
2: Define $\mathbf{m}^0$ as the current configuration of consumers
3: Initialize allocation plan $\mathcal{A} = []$
4: Initialize iteration count $i = 1$
5: $\{\hat{\Lambda}_j\} =$ `DECOMPOSE`$(\{\Lambda_j\}, \{M_j\}, \mathbf{m}^0)$ ▷ Initial decomposition
6: **while** $WIP(\mathbf{m}^0) > \mathcal{T}$ and $F(\mathbf{m}) < \mathcal{C}$ **do**
7: $\hat{j} = argmax_{1 \leq j \leq J} \Delta(m_j^i, m_j^i + 1)$ ▷ Find the most beneficial PC
8: $m_{\hat{j}}^i = m_{\hat{j}}^i + 1$ ▷ Add one consumer to that PC
9: $\mathcal{A}.append(\hat{j})$ ▷ Update allocation plan
10: $\{\hat{\Lambda}_j\} =$ `DECOMPOSE`$(\{\Lambda_j\}, \{M_j\}, \mathbf{m}^i)$ ▷ Update $\{\hat{\Lambda}_j\}$
11: $i = i + 1$ ▷ Update iteration count
12: Return $\mathcal{A}$, $\mathbf{m}^i$

In terms of complexity, the online component of the `DECOMPOSE` procedure requires to iterate over all vertices and edges in the aggregated graph. Therefore, the complexity

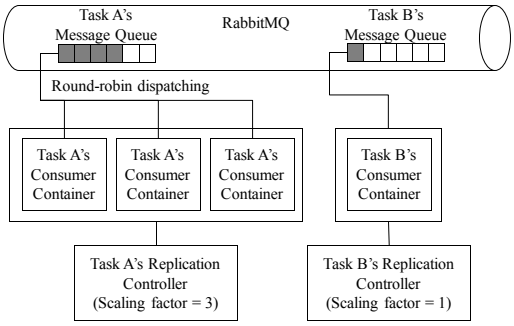


Figure 7: Compute plane implementation.

of DECOMPOSE is $O(|V| + |E|)$, with V and E being the set of vertices (i.e., PCs and $|V| = J$) and edges (i.e., and interactions between PCs) in the aggregated graph. With GRESMAN, finding the most beneficial PC (Line 7) and calculating WIP (Line 6 – executed once for each iteration) both require to iterate over all PCs. The number of iterations of the while loop depends on the convergence of WIP to under the time constraint $\mathcal{T}$, or the total resource cost (which equals the total number of consumers) reaches the budget constraint. For simplicity, if we assume that the cost constraint is met first, then the complexity of GRESMAN is $O(\mathcal{C} \cdot (|V| + |E|))$.

D. Design Validation

We implement 4CeeD's coordination front-end by leveraging Clowder implementation with added features including nested collections data model, structured data-based search (or faceted search). The uploaders and curators communicate with coordination service via a set of RESTful APIs that support uploading and curating data.

For the compute plane, we use RabbitMQ[6] as the message queue engine and container technology (Docker[7] in particular) as the deployment platform for consumers (Figure 7). Specifically, each consumer is implemented and encapsulated into a Docker container and subscribes to RabbitMQ's message queue of a task. Deploying consumers as containers not only provides better isolation and server consolidation for data processing tasks, but also allows us to easily scale the number of consumers that subscribe to a task by adding or removing consumer container of the task. In our implementation, we use Kubernetes[8] as the Docker container orchestration engine for our back-end cluster and abstract the set of consumers of each task as a Kubernetes' replication controller, which ensures that a specified number, which is defined as scaling factor, of container "replicas" are running at any one time. The number of consumers per task m_j can be updated by simply changing the scaling factor of the appropriate replication controller. After that, Kubernetes will handle the rest, from creation (or removal) of consumer's containers, to restarting crashed containers and migrating containers to other available nodes if a node in the cluster crashes.

In terms of work sharing among consumers in a topic, we employ round-robin dispatching mechanism for each topic's message queue so that multiple requests can be processed in parallel and each consumer will receive a fair share of requests.

In terms of fault tolerance, to make sure a request never gets lost (e.g., because a consumer crashes, or a consumer is removed from a message queue while processing a request due to resource allocation decision by resource manager), we employ a message acknowledgment mechanism between message queue and its consumers. Specifically, a request is still cached in message queue even after is has been dispatched to a consumer, and an acknowledgement is sent back from the consumer to tell message queue that the request has been received, processed and that the message queue is free to delete it.

V. Evaluation

We have implemented and deployed the 4CeeD system at the University of Illinois at Urbana-Champaign (UIUC). We currently have 27 test users from two of UIUC major laboratories: Micro and Nano Technology Lab (MNTL) and Material Research Lab (MRL). We open-sourced 4CeeD and make it available online. For more details about the features and implementation of 4CeeD' services, please visit our project website [9]. In the following, we present some evaluation results on 4CeeD's curation and coordination services.

A. Curation Services

To evaluate curation services, we ask our beta testers for their opinion about the tool after a few months of use. For example, 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 (i.e., how much can 4CeeD help accelerate material research).

In terms of the ease of use, the users' responses show that the curator's data curation steps are simple and intuitive, and the data model is flexible enough to allow them to organize data using their own preferred organizational strategy.

In terms of the time saving, our results show that an average SEM user spends about 15% of the time exporting and transferring the images to a server. In addition to this time, he/she also need to spend another 15% of the time analyzing the images since he/she does not have a way to view the proprietary image format and 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 lab session reservation, an average

[6]RabbitMQ - https://www.rabbitmq.com
[7]Docker - https://www.docker.com
[8]Kubernetes - http://kubernetes.io
[9]4CeeD Project Website: https://4ceed.github.io

Task	Description
A	Unpacking digital microscope output files (e.g., DM3, HDF5)
B	Extracting and analyzing metadata from input file
C	Extracting and analyzing image from input file
D	Classify the input file into appropriate experiment type and predict if the experiment is successful or not

(a) Supported types of task.

Job type	Format
1	A → B → D
2	A → B, A → C, B → D, C → D
3	C → D

(b) Supported types of job.

Figure 8: Material data processing workflows & tasks.

SEM user 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. Combined with the labor cost, the total cost 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 curation services during lab 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 captured, users know all configurations of the instruments (e.g., SEM camera settings at which the image of the new material was taken), and thus can easily try the same measurement again in the future.

B. Coordination Service

We evaluate the performance of the coordination service in handling variable and bursty workload of heterogeneous types of workflows.

1) Evaluation Settings: In terms of the workflow set, we use the material data processing workflows (Figure 8), or MDP for short. Particularly, these workflows support processing experimental data formats generated by digital microscopes, such as DM3, HDF5. Four types of tasks are supported, which correspond to the steps needed to process input data (Figure 8a), and each workflow consists of all or a subset of supported tasks (Figure 8b). In addition, to further evaluate the ability of the system to support more complex workflows, we also evaluate with LIGO Inspiral Analysis workflows [7] that analyze data from the coalescing of compact binary systems such as binary neutron stars and black holes. In particular, we use 3 complex sub-workflows from LIGO: CAT, Full, and Injection, which consist of 7 types of tasks. For more information about the LIGO workflows, please refer to [7].

In terms of the workload, as shown in our user study and survey [2], the workload of data generated from instruments in material-related research tends to be very bursty (i.e., abnormal high rate of requests during experiment sessions). Therefore, in our evaluation, we emulate the bursty workload situation by abnormally increasing the arrival rates of requests to the system up to 10 times compared with the normal rates, and measure the effectiveness of our resource allocation strategy.

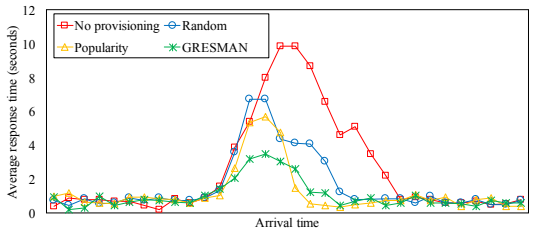
We deployed the coordination service on a cluster of three servers, each server is equipped with an Intel Xeon quad core processor (1.2Ghz for each core) and 16GB of RAM.

2) Effectiveness comparison of resource allocation strategies: In this evaluation, we take the bursty workload situation described earlier and compare the effectiveness of our `GRESMAN` (Algorithm 1), with baseline resource allocation strategies: `Random` (which randomly assign consumers to PCs), and `Popularity` (which assign consumers to PCs proportional to the popularity of the task, or the number of workflows a task belongs to). The main metric for comparison is the average response time over all types of workflows.

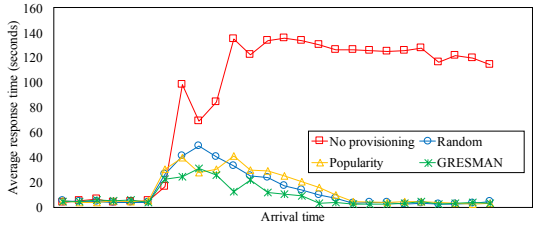
We initially allocate one consumer to each topic in MDP and LIGO workflows, except Inspiral & TrigBank tasks for their high popularity in LIGO's workflows, and seek to minimize average job response time given a resource cost constraint of 10 and 60 additional consumers for MDP and LIGO workflows respectively. Resource allocation is kicked off when the resource manager observes that average job response times exceed a certain threshold (2 seconds for MDP, and 10 seconds for LIGO).

From the results in Figure 9, we can easily see the impact of the abnormal change in the number of arrival requests to the response time without provisioning in both workflow sets (the red line). The impact is more significant in LIGO case, since it is a more complex set of workflows with higher number of tasks and interactions between tasks. The results also show that, in both workflow set, our `GRESMAN` algorithm is more effective than the baselines in dealing with bursty workload situation, and the `Popularity` approach performs better than `Random`. That is because our approach can accurately capture the workload situation of each individual PC and allocates additional consumers to the PCs that are most beneficial in bringing down the response time. On the other hand, random allocation does not consider the workload situation of individual PC when allocating additional consumers, and `Popularity` approach relies on a simple heuristic based on the popularity of PCs to decide its allocation.

3) Scalability: To evaluate the scalability of the coordination service, we vary the arrival rates of job requests and measure the number of consumers that need to be provisioned (the allocation of consumers over PCs is generated by `GRESMAN` algorithm) so that the average response time of



(a) MDP workflows.



(b) LIGO workflows.

Figure 9: Effectiveness comparison of resource allocation approaches in bursty workload situation.

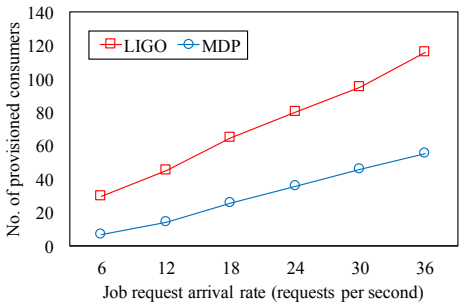
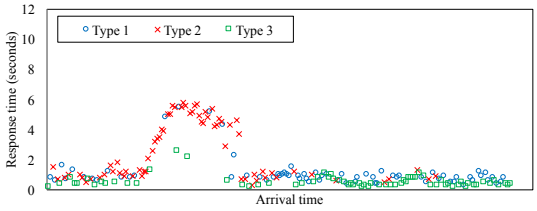
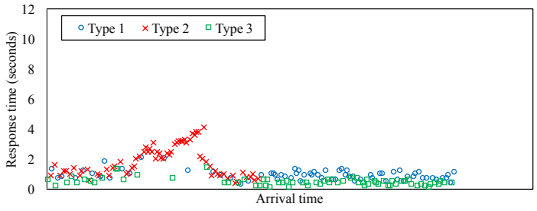


Figure 10: Scalability of coordination service by varying arrival rates of job requests.



(a) Effect of resource provisioning using GRESMAN on different job types.



(b) Carefully ordering provisioning of resources further improves provisioning effectiveness.

Figure 11: Impact of resource provisioning order in handling bursty workload with MDP workflows.

the system is kept under a certain threshold. We use both sets of workflows for this evaluation and set the response time threshold to 2 seconds for MDP and 10 seconds for LIGO workflows.

The result in Figure 10 shows that, as the arrival rate increases[10], the number of consumers that coordination service need to provision must increase to maintain the quality of service (i.e., overall response time under a certain threshold). However, we also see that, in both sets of workflows, the required number of consumers is almost linear to the job request arrival rates. This is acceptable and allow us to further scale the coordination service to adapt to a high number of incoming requests. In our evaluation, with a modest setup of three-node cluster (described earlier), we can provision up to 150 consumers, each running as a Docker container, using Kubernetes without any performance issue.

4) Impact of resource order on provisioning overhead: As we have seen in Figure 9, it still takes some time for resource provisioning to take full effect and put the response time back to normal. One of the main reasons is due to the overhead of starting up provisioned consumers. This overhead is difficult to avoid without advance knowledge of arrival workload. Another reason, as we find out in our experiments, is due to the provisioning order of resources. Let us again take the bursty workload situation with MDP set in previous section as an example. In this case, our GRESMAN algorithm decides new optimal provisioning strategy as $\mathbf{m} = (m_A, m_B, m_C, m_D) = (2, 3, 2, 7)$. If we simply provision the PCs in the sequential order, e.g., first start with A, then B, C, and D, it might put D, which is the one needs additional consumers the most, under more stress, because other PCs have their consumers provisioned earlier and start sending more requests to D. To reduce the effect of the second overhead, we can carefully order the provisioning of topics. In particular, we leverage the allocation plan $\mathcal{A}$ obtained from the GRESMAN algorithm (Algorithm 1), which specifies the provisioning order additional consumers to PCs that is most beneficial in bringing down the overall response time. Figure 11b shows the results of the carefully ordered provisioning, compared with the results by using sequential provisioning order (i.e., A, B, C, D) in Figure 11a, when they both use the same optimal allocation $\mathbf{m} = (2, 3, 2, 7)$ generated by GRESMAN algorithm.

[10]The request arrival rates we used in the evaluation are significantly larger than the actual rate we observe in real lab environment as shown in [2] (measured at hour scale).

VI. Related Work

The related efforts in scientific data management and cyberinfrastructure have been focusing on making existing datasets more accessible and shareable (e.g., DataUp [14], SkyServer [15]), toward long-terms preservation (e.g., SEAD [9]). Other efforts focus on providing easy access and collaboration to distributed cyberinfrastructure that incorporate cloud and grid technologies, such as HubZero [5], NanoHub [6] (for nanotechnology simulations), BrownDob [4], Data Conservancy Instance [16] (for cloud-based data

curation). Our focus, on the other hand, shifts to 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. Thus, our effort is complement to those other efforts, and we could effectively leverage results of existing solutions (e.g., data preservation tools for long-term storage of data, data curation tools for curating data of after it has been captured and stored). In addition, our solution is *the first* one to digitally connect material science and semiconductor fabrication via data. Although 4CeeD is designed for materials-related data capturing and processing, it can be extended to use in other domains where the nested data model and workflow-based data processing mechanism apply.

Scientific workflow systems [10][19] have traditionally employed a monolithic design. Each workflow is described by a high-level workflow specification of tightly coupled tasks and is translated into a workflow execution plan to run on a computation infrastructure with little coordination with other workflows that might share common tasks. Thus, they are not efficient in dealing with heterogeneous workflows. In this paper, we present a micro-service based approach to improve the flexibility of workflow composition and execution, as well as enable fine-grained scheduling at task level considering task sharing across different workflows.

Publish-subscribe system [18], with its wide range of applications, has been a large topic of study. Our coordination service uses publish/subscribe middleware to connect different processing components. Different from related work on resource management of elastic pub/sub systems [12][17] [13], we leverage our previous result [11] on theoretical modeling of the performance of elastic pub/sub system to design efficient resource allocation algorithms to support scalable heterogeneous workflows execution (i.e., `GRESMAN` and `DECOMPOSE` procedures). Our proposed workload-aware resource allocation strategies can be used in complement with other more generic cloud resource management solutions, such as YARN [21] and Mesos [20] that mainly focus on allocating available computational resources to applications.

VII. Conclusions and Future Work

In conclusion, in this paper, we have presented design and implementation of 4CeeD, a data acquisition and analysis framework for materials-to-devices processes that supports capturing, curating, correlating, and coordinating materials-to-devices digital data in a real-time and trusted manner before fully archiving and publishing data.

In terms of future work, we would like to incorporate an edge device, or cloudlet, into the future design to help shaping data transfer between multiple instruments sides of different and mismatched network capabilities and the cloud to avoid traffic congestion, time-outs, data losses, and reduce end-to-end delays. 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.

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] Little's Law https://en.wikipedia.org/wiki/Little%27s_law Accessed November 07, 2016.

[2] User Study and Survey on Material-related Experiments http://hdl.handle.net/2142/94738 Accessed November 30, 2016.

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

[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] Juve, G. et al. *Characterizing and profiling scientific workflows*. Future Generation Computer Systems, 29(3) 2013, 682-692.

[8] Van Vliet, M. et al. (1991). *Machine allocation algorithms for job shop manufacturing*. Journal of Intelligent Manufacturing.

[9] 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.

[10] Liu, J. et al. (2015). *A survey of data-intensive scientific workflow management*. Journal of Grid Computing, 13(4), 457-493.

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

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

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

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

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

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

[17] Tran, N.-L., Skhiri, S., and Zimnyi, E. *Eqs: An elastic and scalable message queue for the cloud*. In Proc. of CloudCom 2011.

[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] Yu, J., & Buyya, R. (2005). *A taxonomy of scientific workflow systems for grid computing*. ACM Sigmod Record, 34(3), 44-49.

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

[21] 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).