Opportunities and Challenges in Running Scientific Workflows on the Cloud

Yong Zhao
School of Computer Science and Engineering
Univ. of Electronic and Science Technology of China
Chengdu, China
yongzh04@gmail.com

Xubo Fei
Department of Computer Science
Wayne State University
Detroit, USA
xubo@wayne.edu

Ioan Raicu
Department of Computer Science
Illinois Institute of Technology
Chicago, USA
iraicu@iit.edu

Shiyong Lu
Department of Computer Science
Wayne State University
Detroit, USA
shiyong@wayne.edu

Abstract— **Cloud computing is gaining tremendous momentum in both academia and industry. The application of Cloud computing, however, has mostly focused on Web applications and business applications; while the recognition of using Cloud computing to support large-scale workflows, especially data-intensive scientific workflows on the Cloud is still largely overlooked. We coin the term "Cloud Workflow", to refer to the specification, execution, provenance tracking of large-scale scientific workflows, as well as the management of data and computing resources to enable the execution of scientific workflows on the Cloud. In this paper, we analyze why there has been such a gap between the two technologies, and what it means to bring Cloud and workflow together; we then present the key challenges in running Cloud workflow, and discuss the research opportunities in realizing workflows on the Cloud.**

Cloud computing; Scientific Workflow; Cloud workflow; Data Intensive Computing

I. INTRODUCTION

Governments, research institutes, and industry leaders are rushing to adopt Cloud Computing to solve their ever-increasing computing and storage problems arising in the Internet age. There has been a burgeoning of Cloud platforms and applications in both academia and industry. Only in a few years after Amazon released its Elastic Computing Cloud (EC2) and Simple Storage Service (S3) to the public, Google released App Engine, IBM unveiled "Blue Cloud" [1]; and Microsoft also rolled out the Azure Services Platform [2]. There are also quite a few open source Cloud computing platforms such as Hadoop, Eucalyptus [19], and Nimbus [13].

We define Cloud computing as a large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet [4].

There are a couple of major benefits and advantages that are driving the widespread adoption of the Cloud computing paradigm:

1) Development based on an abstract computing model: most Cloud computing platforms hide the complexity of the Cloud by providing an abstract computing model; 2) Scalability on demand: once an application is deployed onto the Cloud, the application can be automatically made scalable by provisioning the resources in the Cloud on demand, and the Cloud takes care of scaling out and in, and load balancing; 3) Better resource utilization: Cloud platforms can coordinate resource utilization according to resource demand of the applications hosted in the Cloud; and 4) Cost saving: Cloud users are charged based on their resource usage in the Cloud, they only pay for what they use, and if their applications get optimized, that will be reflected into a lowered cost immediately.

Such Cloud platforms, however, have mostly been applied to Web applications and business applications, there is a missing link that is to manage and run workflow applications, especially data-intensive scientific workflows on the Cloud. The current state of workflow organization on the Cloud has been either 1) static predefined pipelines based on batch style scripts or graphs based on the MapReduce [9] programming model; 2) ad hoc mash-up's that are connected together with, again, scripts that parse the output of one web application and feed into another.

Although several scientific workflow management systems (SWFMSs) have been successfully applied over a number of execution environments (local hosts, clusters/grids, and supercomputers), Cloud computing provides a paradigm-shifting utility-oriented computing model in terms of the unprecedented size of datacenter-level resource pool and the on-demand resource provisioning mechanism, enabling scientific workflow solutions capable of addressing peta-scale scientific problems.

We coin the term "Cloud Workflow", to refer to the specification, execution, and provenance tracking of scientific workflows, as well as the management of data and computing resources to enable the running of scientific workflows on the Cloud. In the rest of this paper, we discuss what it means to bring Cloud and workflow together; present the key challenges in supporting Cloud workflows, and identify key research opportunities in realizing workflows on the Cloud.

II. OPPORTUNITIES

We have seen the success of the Internet and Web largely due to the incentive of being connected and the TCP/IP protocol that makes such connectivity possible. While the incentive of providing computing as a utility has long been envisioned, the underlying technology that makes it possible has finally come [5]. The illusion of infinite computing resources that is provided by Cloud Computing on demand to end users is fascinating to a wide range of science and engineering applications, particularly to data and/or compute-intensive scientific workflow applications.

First, the scale of scientific problems that can be addressed by scientific workflows is now greatly increased, which was previously upbounded by the size of a dedicated resource pool with limited resource sharing extension in the form of virtual organizations.

The scale of scientific problems is reflected not only on the data sizes that scientific applications need to handle, but also on the complexities of the applications themselves. For data sizes, the scientific community is facing a "data deluge" coming from experiments, simulations, sensors, and satellites. For example, the archival data from the National Virtual Observatory for sky- and ground-based observatory is estimated to cover 40,000 square degrees of the sky and to be a few petabytes. The rate of growth of DNA databases such as GenBank [4] has been following an exponential trend, with a doubling time estimated to be 9-12 months. Data volumes are also increasing dramatically in physics, earth science, medicine, and many other disciplines. As for application complexity, a protein simulation problem [27] involves running many instances of a structure prediction simulation, each with different random initial conditions. The simulation uses an "iterative fixing" algorithm that performs multiple rounds, each involving many parallel Monte Carlo simulated annealing models of molecular moves with energy minimization. Given a couple of proteins and parameter options, the simulation can easily scale up to 100,000 rounds. Similar analyses in other disciplines also need to explore a large parameter space, and expect a fast turn-around time. Cloud platforms can offer vast amount of storage space as well as computing resources for such applications, allowing scientific discoveries to be carried out in an unprecedented scale.

Second, the on-demand resource allocation mechanism in Cloud has a number of advantages over the traditional cluster/Grid environments for scientific workflows:

a) It will improve resource utilization. Workflows usually have multiple stages, where the number of resources required for the stages may vary a lot (for instance, scatter and gather is a common pattern observed in scientific workflows where nodes tend to expand at the scatter stage, and then merge at the gather stage). Cloud-based workflow applications can get resources allocated accordingly with the number of nodes at each stage, instead of reserving a fixed number of resources.

b) It can change the experience of end users for improved responsiveness. Cloud workflows can scale out and in dynamically, resulting a fast turn-around time for end users.

c) It could also enable a new generation of scientific workflows - collaborative scientific workflows [18], in which user interaction and collaboration patterns are first-class entities for scientific workflow management. User interaction and collaboration intensive scientific workflows have been difficult to implement in a Grid environment as it is more suitable for batch-based scientific workflows.

Third, Cloud computing provides a much larger room for the trade-off between performance and cost. The spectrum of resource investment now ranges from dedicated private resources, a hybrid resource pool combining local resource and remote clouds, and a full outsourcing of computing and storage to public Clouds. Cloud Computing not only provides the potential of solving larger-scale scientific problems, but also brings the opportunity to improve the performance/cost ratio. Although the optimization of this ratio and a flexible (semi-)automatic trade-off mechanism still remain as challenging problems; see a recent case study in this direction [15].

III. CHALLENGES

Despite the advantages and opportunities we can seek in Cloud computing for scientific workflows, there are many major obstacles to the adaptation and running of scientific workflows on the Cloud, we identify a few of them below:

Architectural challenges: in our scientific workflow system reference architecture, we define an SWFMS to have four layers – operational layer, task management layer, workflow management layer, and presentation layer. To engineer an SWFMS into Clouds, it may not be as simple as to replace the operational layer with a Cloud infrastructure and that is the end of the integration story. We may need to take a bottom-up approach and evaluate the requirements, looking at integration problems at the other three layers as well, to address compatibility and impedance problems that may be introduced by different Cloud providers and implementations.

Integration challenges: many of the immediate challenges of running scientific workflows on the Cloud are to integrate scientific workflow systems with Cloud infrastructure and resources. In most cases, we will need to change the way an SWFMS acquires resources, dispatches tasks, monitors the progress of those tasks, tracks provenance information, and how it deals with errors and exceptions in the Cloud.

Computing challenges: for scientific workflows to leverage large scale computing resources in the Cloud, there are challenges such as resource requirements and provisioning, virtualization, fault tolerance, and smart-reruns.

Data management challenges: running workflows in the Cloud has to deal with data moving in and out of the Cloud, large scale data storage within the Cloud, and the exploration of data locality, data and computation co-location issues for efficiency purpose, and the tracking of data provenance in order to understand and reuse workflows.

Language challenges: so far in the Cloud, MapReduce has been the "only" widely adopted computing model, and there are a number of variations of languages based on this model for task specification in the Cloud. Examples are Sawzall [20], DryadLINQ [29], etc. However, a workflow specification requires far more functionality and flexibility than MapReduce can provide, and the implicit semantics incurred by a workflow specification goes far more than just the "map" and "reduce" operations, for instance, the mapping of computation to compute node and data partitions, runtime optimization, retry on error, smart re-run, etc. The specification and the corresponding implementation of the specification would carry about all the computing, data management challenges associated with interpreting and executing the specification.

Last but not the least, is service management challenges, as Clouds are mostly built on top of service oriented architecture, and SWFMSs are also shifting from conventional applications to service invocations. We need to deal with service discovery, large input and output handling, data services, and all the other challenges that we are facing in migrating applications into a service world.

A. *Architectural challenge*

Based on a comprehensive study of the workflow literature from an architectural perspective and our own experience from the development of the VIEW system [10][16][17] and Swift [31], we identify the following seven key architectural requirements for an SWFMS: (R1) User interface customizability and user interaction support; (R2) Reproducibility support; (R3) Heterogeneous and distributed services and software tools integration; (R4) Heterogeneous and distributed data product management; (R5) High-end computing support; (R6) Workflow monitoring and failure handling; and (R7) Interoperability.

A reference architecture for SWFMSs is proposed in [16] and an SOA-based instantiation is first implemented in the VIEW system. As shown in Figure 2, the reference architecture consists of four logical layers, seven major functional subsystems, and six interfaces. The first layer is the Operational Layer, which consists of a wide range of heterogeneous and distributed data sources, software tools, services, and their operational environments, including high-end computing environments. The second layer is called the Task Management Layer. This layer consists of three subsystems: Data Product Management, Provenance Management, and Task Management. The third layer, called the Workflow Management Layer, consists of Workflow Engine and Workflow Monitoring. Finally, the fourth layer – the Presentation Layer, consists of the Workflow Design subsystem and the Presentation and Visualization subsystem. A detailed description of the architecture is available in [20].

We argue that the above reference architecture is still valid for a Cloud-enabled SWFMS. Such validity has been achieved by the layered approach of the reference architecture and the main design principle behind the architecture: with the fast advancement of underlying computing technology, upper layers of the reference architecture should not be disturbed. The reference architecture provides the guidance for designing a concrete solution for a particular SWFMS. Here, we consider four possible solutions for deploying the proposed reference architecture in a Cloud computing environment:

- Operational-Layer-in-the-Cloud. In this solution, only the Operational Layer lies in the Cloud with an SWFMS running out of the Cloud. An SWFMS can now leverage Cloud applications as another type of task components. Cloud-based applications can take advantage of high scalability provided by the Cloud and large resource capacity provisioned by data centers. This solution also relieves a user from the concern of vendor lock-in due to the relative ease of using alternative Cloud platforms for running Cloud applications. However, the SWFMS itself cannot benefit from the scalability offered by the Cloud.
- Task-Management-Layer-in-the-Cloud. In this solution, both the Operational Layer and the Task Management Layer will be deployed in the Cloud. In contrast to traditional deployment strategies, Data Product Management, Provenance Management, and Task Management can now leverage the high scalability provided by the Cloud. For Task Management, rather than accommodating the user's request based on a batch-based scheduling system, all or most ready tasks can now be immediately deployed over Cloud computing nodes and executed instead of waiting in a job queue for the availability of resources. One limitation of this solution is that the economic cost associated with the storage of provenance and data products in the Cloud. Moreover, although task scheduling and management can benefit from the scalability offered by the Cloud, workflow scheduling and management do not since the workflow engine runs outside of the Cloud.

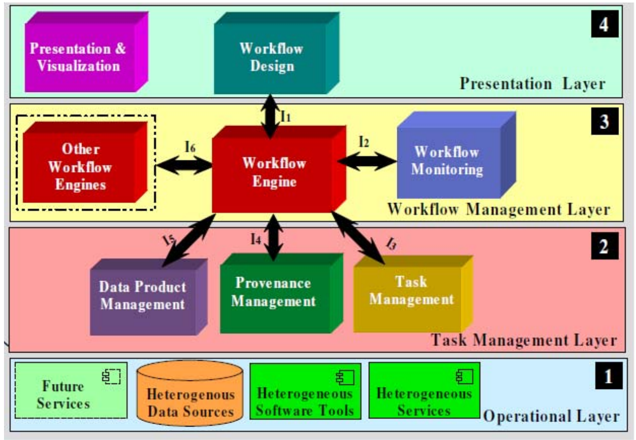

Figure 2 A reference architecture for SWFMSs [16].

- Workflow-Management-Layer-in-the-Cloud. In this solution, the Operational Layer, the Task Management Layer, and the Workflow Management Layer are deployed in the Cloud with the Presentation Layer deployed at a client machine. This solution provides a good balance between system performance and usability: the management of computation, data, and storage and other resources are all encapsulated in the Cloud, while the Presentation Layer remains at the Client to support the key architectural requirement of user interface customizability and user interaction support [20]. Such a solution is also most suitable for a scientific workflow application system in which *ad hoc* domain-specific requirements are constantly evolving, demanding frequent changes to the Presentation Layer for that domain. In this solution, both workflow and task management can benefit from the scalability offered by the Cloud, but the downside is that they become more dependent on the Cloud platform over which they run.

- All-in-the-Cloud. In this solution, a whole SWFMS is deployed inside the Cloud and accessible via a Web browser. A distinct feature of this solution is that no software installation is needed for a scientist and the SWFMS can fully take advantage of all the services provided in a Cloud infrastructure. Moreover, the cloud-based SWFMS can provide highly scalable scientific workflows and task management as services, providing one kind of Software-as-a-Service (SaaS). One concern the user might have is the economic cost associated with the necessity of using Cloud on a daily basis, the dependency on the availability and reliability of the Cloud, as well as the risk associated with vendor lock-in.

As we described, each of the above solutions has its cons and pros. In practice, a hybrid approach might also be desirable, in which for each layer, one subsystem or a piece of the subsystem is deployed in the Cloud, while the rest is deployed outside of the Cloud. For each solution, a refined microarchitecture for each layer and subsystem desires further research. We are currently experimenting with the last two solutions in the context of the VIEW system. Our research results will be presented in a future publication.

B. *Integration challenge*

Many of the immediate challenges to running scientific workflows on the Cloud are to integrate scientific workflow systems with Cloud infrastructure and resources. As we have discussed in the previous section, the degree of integration also depends on how we choose to deploy an SWFMS into Clouds. While we certainly cannot cover all aspects of the integration problems that we could encounter in the "all-in-the-cloud" approach, we strive to identify some practical ones and discuss possible solutions to them.

Applications, services, and tools integration: In the operational-layer-in-the-Cloud approach, we treat applications, services, and tools hosted in the Cloud as task units in a workflow, the scheduling and management of a workflow are mostly outside the Cloud, where these task units are invoked as they are scheduled to execute. A majority of the mashup sites (such as those that leverage Google's map service) take this approach, and they use *ad hoc* scripts and programs to glue the services together. An early exploration of the Taverna [14] workflow engine and gRAVI services in the caBIG project [25] can also be thought as an example of integrating an off-the-shelf workflow engine with Cloud/Grid services.

Once we decide to get task dispatching and scheduling into the Cloud, resource provisioning becomes the next issue to resolve. Although conceptually Cloud offers uncapped resources, and a workflow can request as much resource as it requires, these all come with a cost, and presume that the workflow engine can talk directly with the allocated resource s(which is usually not true without tweaking the configuration of the workflow engine). Taking these two factors into consideration, some existing solutions such as Nimbus would acquire a certain number of virtual machines, and assemble them as a virtual cluster, onto which existing cluster management systems such as PBS can be deployed and used as job submission/execution service that a workflow engine can directly interact with.

Debugging, monitoring, and provenance tracking for a workflow can be even more difficult in the Cloud, since compute resources are usually dynamically assigned and based on virtual machine instances, the environment that a task is executed on could be destroyed right after the task is finished, and assigned to a complete different user and task. Some Clouds also support task migration where tasks can be migrated to another virtual machine if there is problem with the node that the task is running on.

Porting an SWFMS into the Cloud can also be a concern, which would usually involve wrapping up an SWFMS as a Cloud service. To fully explore the capability and scalability of the Cloud, a workflow engine may need to be re-engineered to interact directly with various Cloud services such as storage, resource allocation, task scheduling, monitoring, etc. At the client side, either a complete Web-based user interface needs to be developed to allow users to specify and interact with the SWFMS, or a thin off-the-Cloud client needs to be developed to interact with the SWFMS Cloud service.

C. *Language challenge*

So far in the Cloud, MapReduce has been the "only" widely adopted computing model, and there are a number of variations of languages based on this model for task specification in the Cloud. MapReduce provides a very simple programming model and powerful runtime system for the processing of large datasets. The programming model is based on just two key functions: "map" and "reduce," borrowed from functional languages. The runtime system automatically partitions input data and schedules the execution of programs in a large cluster of commodity machines. Sawzall [20] further simplifies the program specification and task parallelization. It is an interpreted language that builds on MapReduce and separates the filtering and aggregation phases. Microsoft has developed the Cosmos distributed storage system and dryad processing framework, and offers DryadLINQ [29] and SCOPE [7] as declarative programming model on top of the storage and computing infrastructure. DryadLINQ uses the object oriented LINQ query syntax where SCOPE provides basic operators similar to those of SQL such as Select, Join, Aggregation, etc., both translate the abstract specification into a detailed execution plan.

While MapReduce and its variations provide certain data flow support, they all require application logic to be re-written to follow the map-reduce-merge programming model. We call this kind of workflow organization the "White-Box" approach, as users need to fully understand the applications and port the applications before they can leverage the parallel computing infrastructure. Moreover, the data being processed also need to be stored in partitioned fashion, such as in GFS, or HDFS, so that the partitions can be operated in parallel.

SwiftScript [30][31], on the other hand, serves as a general purpose coordination language, where existing applications can be invoked without modification. We call this the "Black-Box" approach, in which we focus more on the input data and output data of each computing node, and the flow of the data. Of course, some approaches will cross the edge of being white or black, as some form of modification or adaptation to the applications will be needed.

SwiftScript provides foreach and iterate operators that correspond to the Map function in MapReduce, which basically iterates over an array of data and performs a certain operation on each of the data element in the array. For functionalities similar to the Reduce and Combine operations introduced above, it will have to rely on specific applications that perform such operations. SwiftScript also uses implicit parallelism: iterations are mapped into parallel operations automatically, and independent tasks are scheduled to run in parallel. The advantage to SwiftScript style workflows is that the organization of applications and data can be more flexible, and the execution of workflows can be scheduled to run on a single box, or onto Grids and Clouds, as it does not need to port existing applications, and does not rely on specific data partitioning.

Mash-up's and ad hoc scripts (Java Script, PHP, Python etc.) have become key technologies for developing Web applications that dynamically integrate multiple data or service sources. They are essentially data integration approaches, because they take the outputs from one service/application, transform them and feed into another. Google App Engine uses a modified Python runtime and chooses Python scripting language for Web application development. Clouds such as Amazon Web Services and Microsoft's Azure Services Platform have generally adopted Web Services APIs where users access, configure and program Cloud services using pre-defined APIs, and HTTP, SOAP are the common protocols chosen for such services.

For Cloud workflow coordination, no matter what forms of language we adopt, such as APIs and scripts provided by the MapReduce computing model, or scripting languages like SwiftScript, or service based business workflow like BPEL, they all need to address the following challenges:

- Handle the mapping from input and output data into logical structures to facilitate data integration and logical operations on data.
- Support large-scale parallelism via either implicit parallelism, or explicit declaratives such as Parallel Foreach.
- Support data partitioning and task partitioning. Considering the scale of computation and data processing, data and tasks need to be efficiently partitioned and scheduled onto a large number of compute/storage nodes; and processed in parallel to improve system throughput and efficiency.
- Require a scalable, reliable, and efficient runtime system that can support Cloud-scale task scheduling and dispatching, provide error recovery and fault tolerance under all kinds of hardware and service failures, and utilize a large pool of Cloud resources efficiently.

D. *Computing challenge*

Although Clouds can potentially offer unlimited resources to SWFMSs, managing large-scale of computing resources is not a trivial task. As we have mentioned in the integration challenge section, workflow systems may not be able to talk to Cloud resources directly, they may still need to go through middleware services such as Nimbus and Falkon that handle resource provisioning and task dispatching. Things can be even more complicated if we take into consideration issues such as workflow resource requirements, data dependencies, Cloud virtualization, etc. Before we dive into details, let's take a look at how Amazon's Elastic MapReduce service [3] handles a workflow (well, to be more precise, a data flow):

Amazon Elastic MapReduce creates data processing job flows that are executed in the Hadoop platform on the web-scale infrastructure of Amazon EC2. The service automatically launches and configures the number and type of Amazon EC2 instances specified by customers. It then kicks off a Hadoop implementation of the MapReduce programming model, which loads large amounts of user input data from Amazon S3 and then subdivides the data for parallel processing on Amazon EC2 instances. As processing completes, data are re-combined and reduced into a final solution, and the results deposited back into Amazon S3. Users can configure, manipulate, and monitor job flows through web service APIs or via the AWS Management Console.

Essentially, it is the user's responsibility to specify the type of resources (chosen out of a few pre-configured EC2 instance types), and the number of resources. Data is copied in and out of the S3 storage service, and the user is able to monitor the status of the job flow. Different stages of a workflow may require different types of resources, and Cloud virtualization can configure Virtual Machines differently to meet such requirements, but to what extent (i.e. how much granularity) and how flexible it can be would be hard to decide. Amazon only offers a few EC2 instance types coarsely categorized as small, medium, and large, and they are charged differently according to the computing power they provide.

Traditional SWFMSs also place special emphasis on fault tolerance and smart reruns. A workflow may involve a large number of computations and the whole process can be lengthy, so typically a SWFMS will try to automatically recover when non-fatal errors happen (by using mechanisms such as retry on error, re-schedule computation to a different resource, etc.). Also, in the case the workflow has to be stopped, detailed execution information will be logged, and the next time the workflow is re-started, it will be able to pick up from where it was stopped, this is called smart-rerun. In a Cloud environment, the scale of a workflow can be much larger, and more components (such as VMs) can be involved, some extra measures need to be taken to support such features.

E. *Data management challenge*

As scientific applications become more data intensive, the management of data resources and dataflow between the storage and compute resources is becoming the main bottleneck. Analyzing, visualizing, and disseminating these large data sets have become a major challenge and data intensive computing is now considered as the "fourth paradigm" [12] in scientific discovery after theoretical, experimental, and computational science. Within a Cloud, data management is as important as, and sometimes, even more critical than compute resource management. As we have mentioned before, in some Clouds, the nodes responsible for data storage are separated from computation nodes, while some others may require them to be collocated. From a workflow perspective, we care more about the following aspects of data management within a Cloud: data locality, where computation can be scheduled to leverage data dependencies among tasks; collective data management, where we can get high aggregated data throughput; and provenance and metadata management.

Data Locality: As CPU cycles become cheaper and data sets double in size every year, the main challenge for efficient scaling of applications is the location of the data relative to the available computational resources – moving data repeatedly to distant CPUs is expensive and inefficient. There are large differences in IO speeds from local disk storage to wide area networks, which can drastically affect application performance. To achieve good scalability at Internet scales for Clouds, Grids, and their applications, data need to be distributed over many computers, and computations should be steered towards the best place to execute in order to minimize communication costs. Google's MapReduce system runs on top of the Google File System, within which data is loaded, partitioned into chunks, and each chunk replicated. Thus data processing is collocated with data storage: when a file needs to be processed, the job scheduler consults a storage metadata service to get the host node for each chunk, and then schedules a "map" process on that node, so that data locality is exploited efficiently.

Combining compute and data management: What is even more critical is the combination of the compute and data resource management, which leverages data locality in access patterns to minimize the amount of data movement and improve end-application performance and scalability [24]. Attempting to address storage and computational problems separately forces much data movement between computational and storage resources, which will not scale to tomorrow's exascale datasets and millions of nodes, and will yield significant underutilization of the raw resources.

Provenance: Provenance refers to the derivation history of a data product, including all the data sources, intermediate data products, and the procedures that were applied to produce the data product. Provenance information is vital in understanding, discovering, validating and sharing a certain data product as well as the applications and programs used to derive it [11]. In some disciplines such as finance and medicine, it is also mandatory to provide what is called an "audit trail" for auditing purpose. Provenance is still an unexplored area in Cloud environments, in which we need to deal with even more challenging issues such as tracking data production across different service providers (with different platform visibility and access policies) and across different software and hardware abstraction layers within one provider. In addition, the scalability of Clouds would require much more scalable provenance systems that can handle the storage and querying of potentially millions of tasks. Also secure access of provenance information, which is largely missing in existing provenance systems, would be much needed in Clouds due to its multi-tenant nature.

F. *Service management challenge*

By talking about service management, we refer to both the engineering of the components of an SWFMS as services, and the orchestration and invocation of services from an SWFMS. While the emergence of SOA as an architectural paradigm provides many benefits for distributed computing, where service abstraction, loose coupling, discoverability and interoperability are some key advantages specifically for the engineering and development of an SWFMS. As a matter of fact, many disciplines (especially in life science) have adopted service implementation, and the Taverna and LEAD [21] workflow systems deal with service workflows explicitly. There are thousands of services developed and available for the myExperiment project, and the LEAD system has developed a tool to wrap and convert ordinary science applications into services.

Orchestrating and invoking services via an SWFMS within the Cloud poses some unique challenges in addition to commonly observed ones such as service description, discovery, and composition: Firstly, managing the large number of service instances would be an issue, each service instance needs to be properly deployed and configured, and for service invocations with state transitions, this would become more tricky as to when and where to instantiate and destroy the instances. Secondly, for services involving large volume of input and output data, data movements across different service instances (and ultimately, the underlying compute and storage instances) will be critical for throughput and performance considerations. In many cases, data services may need to be involved to manage such data movements and possibly data caching as out-of-band operations to the service invocations, passing data references or data service calls instead of embedding the actual data in the invocations. For a workflow that needs to call out to public available services (such as in the case of a mash-up application), the SWFMS also needs to handle security, interoperability, and data transformation issues.

IV. RESEARCH DIRECTIONS

As have been identified in the previous section, there are a variety of challenges in getting workflows to run in the Cloud. However, those are also key areas to which we can put our research efforts and make breakthroughs and advancement towards Cloud based workflows. We want to put our emphasis on the workflow reference architecture and direct research interests towards implementing the key components in the different layers of the architecture, and also putting interoperability and reusability as top priority. There are many existing SWFMS's, but it is difficult to make them interact with each other, due to the lack of clear definition of responsibilities and interfaces. However, transitioning into the Cloud gives the opportunity to engineer and implement the various key components of an SWFMS, preferably by different people with different specialties, and make them work together. By implementing the building blocks in the reference architecture, we can also leverage existing Cloud technologies, such as monitoring, data management, resource provisioning, etc.

However, we are not advocating building a Cloud based SWFMS from ground up. Middleware technologies that can bridge existing workflow systems with the Cloud would seem more cost effective. For instance, virtual cluster technologies such as the ones provided by Nimbus and Falkon, give workflow systems the familiar environment of a cluster, to which they can dispatch tasks to, with minimal adaptation. We believe that there will be a burgeoning of middleware development in the areas of resource management, monitoring, messaging for Clouds that can be used as extensions to existing SWFMSs.

Many task computing (MTC) [22] has been defined to distinguish from traditional high performance computing (HPC) and high throughput computing (HTC), in the emphasis of using large number of computing resources over short periods of time to accomplish many computational tasks (i.e. including both dependent and independent tasks), where primary metrics are measured in seconds (e.g. FLOPS, tasks/sec, MB/s), as opposed to operations (e.g. jobs) per month, while MTC has primarily been applied in Grids and supercomputers, in the Cloud, it would be equally or even more critical, since a large Cloud workflow could involve the execution of millions of tasks, each taking a short time to finish. Methods for improving resource utilization, scheduling efficiency, and IO rates will benefit both Cloud service providers and end users.

Scripting can be an interesting and powerful direction too. We have seen the applicability of simple shell scripts [26], SCOPE and Swiftscript to large scale computing problems on large scale computing resources. Scripting has the advantage of being concise and flexible, yet powerful when combined with parallel semantics and logical operations. We expect to see scripting languages that have a mixture of these semantics, combining the coordination of applications and services (e.g. Swift), the general Map-Reduce-Merge Cloud computing model, and its relational flavored extensions (e.g. SCOPE), and things beyond.

As the Cloud is usually associated with cost, and there are many ways to configure, procure resources and execute tasks, it is to our nature to analyze the cost for computation and resource utilization, and to estimate and optimize the return on investment. Such optimization will again be more challenging in Clouds than in traditional cluster and Grid environments, but it will be more rewarding too.

Provenance in Cloud can adopt the SOA model as this would make provenance less coupled with an SWFWS than it currently does. The development of a standard, open and universal representation and query provenance model is underway by the Open Provenance Model initiative (http://openprovenance.org). Scalability would be top concern for implementation of such a model.

Security has been identified as one of the main concerns for the adoption and success of the Cloud [4] and is the first major service that needs to be provided by a Cloud provider. For example, Microsoft Azure Cloud Platform offers access control as a primary service of the .NET Services. Although much research has been done on workflow security, security for Cloud-based SWFMSs is still preliminary, of which we shed some lights on the following three aspects:

Access control. Access control concerns about which principals have the privileges to access which resources [8][28]. In a Cloud-based SWFMS, the resources include Cloud services, SWFMS services and products such as scientific workflows, tasks, provenance, data products, and other artifacts. Due to the dynamic nature (artifacts can be produced constantly) and the large-scale data, metadata, and service sharing nature of the Cloud, access control is a challenging but important research problem.

Information flow control. Information flow control concerns about to whom a piece of information can be passed on. Since a scientific workflow might orchestrate a large number of distributed services, data, and applications, particularly in a large-scale Cloud environment, the mechanism that controls mission-critical information and intellectual property (e.g., secrete parameters used to run a scientific workflow) not being propagated to an unauthorized user is worth looking into.

Secure electronic transaction protocol. Cloud Computing is one kind of utility computing based on the pay-as-you-go pricing model. A secure electronic transaction protocol is to ensure goods atomicity – a user is charged if and only if a service or resource is used by a user and the charge should be no more and no less. To prevent the abuse of Cloud accounts and double or wrong charges by a Cloud provider, further research might be needed to ensure the security of Cloud-based transaction protocols.

V. CONCLUSIONS

As more and more customers and applications migrate into Cloud, the requirement to have workflow systems to manage the ever more complex task dependencies, and to handle issues such as large parameter space exploration, smart reruns, and provenance tracking will become more urgent. As it stands now, mash-up's and MapReduce style task management have been acting in place of a workflow system in the Cloud. Cloud needs the more structured and mature workflow technologies, and vice versa, as Cloud offers unprecedented scalability to workflow systems, and could potentially change the way we perceive and conduct scientific experiments. The scale and complexity of the science problems that can be handled can be greatly increased on the Cloud, and the on-demand nature of resource allocation on the Cloud will also help improve resource utilization and user experience. In this paper, we discuss the opportunities and challenges in bringing workflow systems into the Cloud, with a focus on scientific workflow management systems; we also identify key research directions in realizing scientific workflows in Cloud environments. The key challenges span from fundamental challenges such as architecture challenge, integration challenge, to computing and data management challenges in the middle, and upper layer language challenges. Nevertheless, the challenges are also great opportunities for us to look into and tackle the problems and issues in the way, towards running scientific workflows on the Cloud.

REFERENCES

[1] http://www-03.ibm.com/press/us/en/pressrelease/22613.wss

[2] http://www.microsoft.com/azure/default.mspx

[3] "Introduction to Amazon elastic MapReduce," available from http://awsmedia.s3.amazonaws.com/pdf/introduction-to-amazon-elastic-mapreduce.pdf.

[4] I. Foster, Y. Zhao, I. Raicu, S. Lu. "*Cloud Computing and Grid Computing 360-Degree Compared*", IEEE Grid Computing Environments (GCE08) 2008, co-located with IEEE/ACM Supercomputing 2008.

[5] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica and M. Zaharia, *Above the Clouds: A Berkeley View of Cloud Computing*, EECS Department, University of California, Berkeley, Technical Report No. UCB/EECS-2009-28, February 10, 2009.

[6] http://www.psc.edu/general/software/packages/genbank/,2010

[7] R. Chaiken, B. Jenkins, P.-Å. Larson, B. Ramsey, D. Shakib, S. Weaver, and J. Zhou, *SCOPE: Easy and Efficient Parallel Processing of Massive Data Sets.* in Proc. of the 2008 VDLB Conference (VLDB'08).

[8] A. Chebotko, S. Lu, S. Chang, F. Fotouhi, and P. Yang, "Secure Abstraction Views for Scientific Workflow Provenance Querying", IEEE Transactions on Services Computing, 3(4), pp.322-337, 2010.

[9] Jeffrey Dean, Sanjay Ghemawat: MapReduce*: simplified data processing on large clusters. OSDI* 2004: 137-149.

[10] Xubo Fei, Shiyong Lu, and Cui Lin: A MapReduce-Enabled Scientific Workflow Composition Framework, ICWS 2009: 663-670.

[11] Foster, I., Voeckler, J., Wilde, M. and Zhao, Y., *Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation,* In 14th Conference on Scientific and Statistical Database Management, 2002.

[12] *The Fourth Paradigm: Data-Intensive Scientific Discovery*, Edited by Tony Hey, Stewart Tansley, and Kristin Tolle. Microsoft Research.

[13] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, J. Good, *"On the Use of Cloud Computing for Scientific Workflows"*, 3rd International Workshop on Scientific Workflows and Business Workflow Standards in e-Science (SWBES), 10 December 2008 in Indianapolis, Indiana, USA

[14] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. Pocock, P. Li, and T. Oinn, "*Taverna: a tool for building and running workflows of services.*," *Nucleic Acids Research*, vol. 34, iss. Web Server issue, pp. 729-732, 2006.

[15] Keahey K., T. Freeman. *Science Clouds: Early Experiences in Cloud Computing for Scientific Applications,* Cloud Computing and Its Applications 2008 (CCA-08), Chicago, IL. October 2008

[16] C. Lin, S. Lu, X. Fei, A. Chebotko, D. Pai, Z. Lai, F. Fotouhi, and J. Hua, *"A Reference Architecture for Scientific Workflow Management Systems and the VIEW SOA Solution",* IEEE Transactions on Services Computing (TSC), 2(1), pp.79-92, 2009.

[17] *Cui Lin, Shiyon*g Lu, Zhaoqiang Lai, Artem Chebotko, Xubo Fei, Jing Hua, and Farshad Fotouhi, *"Service-Oriented Architecture for VIEW: a Visual Scientific Workflow Management System",* In Proc. of the IEEE 2008 International Conference on Services Computing (SCC), Honolulu, Hawaii, USA, July 2008, pp.335-342.

[18] Shiyong Lu and Jia Zhang. "Collaborative Scientific Workflows", IEEE International Conference on Web Services (**ICWS**), pp.527-534, Los Angeles, CA, 2009.

[19] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, D. Zagorodnov, *The Eucalyptus Open-source Cloud-computing System*, in Proceedings of 9th IEEE International Symposium on Cluster Computing and the Grid, Shanghai, China.

[20] Rob Pike, Sean Dorward, Robert Griesemer, Sean Quinlan: Interpreting the data: Parallel analysis with Sawzall. Scientific Programming 13(4): 277-298 (2005).

[21] Plale, B., D. Gannon, J. Brotzge, K. Droegemeier, J. Kurose, D. McLaughlin, R. Wilhelmson, S. Graves, M. Ramamurthy, R.D. Clark, S. Yalda, D.A. Reed, E. Joseph, V. Chandrasekar, *CASA and LEAD: Adaptive Cyberinfrastructure for Real-Time Multiscale Weather Forecasting*, Computer special issue on System-Level Science, IEEE Computer Science Press, Vol. 39, No. 11, pp. 56-63. Nov 2006.

[22] Ioan Raicu, Ian Foster, Yong Zhao. "*Many-Task Computing for Grids and Supercomputers*", IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS08), 2008, co-located with IEEE/ACM Supercomputing 2008.

[23] Ioan Raicu, Yong Zhao, Catalin Dumitrescu, Ian Foster, Mike Wilde. "*Falkon: a Fast and Light-weight tasK executiON framework*, IEEE/ACM SuperComputing 2007.

[24] Ioan Raicu, Yong Zhao, Ian Foster, Alex Szalay. "*Accelerating Large-scale Data Exploration through Data Diffusion*", International Workshop on Data-Aware Distributed Computing 2008, co-locate with ACM/IEEE International Symposium High Performance Distributed Computing (HPDC) 2008.

[25] W. Tan, K. Chard, D. Sulakhe, R. K. Madduri, I. T. Foster, S. S.-Reyes, C. A. Goble*: Scientific Workflows as Services in caGrid: A Taverna and gRAVI Approach.* ICWS 2009: 413-420.

[26] Edward Walker, Weijia Xu, Vinoth Chandar, *Composing and executing parallel data-flow graphs with shell pipes*, Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science 2009, Portland, Oregon, November 16 - 16, 2009.

[27] M. Wilde, I. Foster, K. Iskra, P. Beckman, Z. Zhang, Allan Espinosa, Mihael Hategan, Ben Clifford, Ioan Raicu. "*Parallel Scripting for Applications at the Petascale and Beyond*", IEEE Computer Nov. 2009 Special Issue on Extreme Scale Computing, 2009.

[28] Zijiang Yang, Shiyong Lu, Ping Yang: Itinerary-Based Access Control for Mobile Tasks in Scientific Workflows. Ubisafe: 506-511

[29] Y. Yu, M. Isard, D. Fetterly, M. Budiu, U. Erlingsson, P. K. Gunda, and J. Currey, DryadLINQ: A System for General-Purpose Distributed Data-Parallel Computing Using a High-Level Language, Symposium on Operating System Design and Implementation (OSDI), San Diego, CA, 2008.

[30] Y. Zhao, J. Dobson, I. Foster, L. Moreau, M. Wilde, *A Notation and System for Expressing and Executing Cleanly Typed Workflows on Messy Scientific Data,* SIGMOD Record, Volume 34, Number 3, September 2005

[31] Y. Zhao, M. Hategan, B. Clifford, I. Foster, G. v. Laszewski, I. Raicu, T. Stef-Praun, M. Wilde. "*Swift: Fast, Reliable, Loosely Coupled Parallel Computation*", IEEE Workshop on Scientific Workflows 2007.