Software-Defined Federated Cyber-Infrastructure for Science and Engineering

Javier Diaz-Montes
Rutgers Discovery Informatics Institute
Rutgers University, USA
javidiaz@rdi2.rutgers.edu

Moustafa Abdelbaky
Rutgers Discovery Informatics Institute
Rutgers University, USA
moustafa@cac.rutgers.edu

Mengsong Zou
Rutgers Discovery Informatics Institute
Rutgers University, USA
mengsongzou@gmail.com

Manish Parashar
Rutgers Discovery Informatics Institute
Rutgers University, USA
parashar@rutgers.edu

ABSTRACT

Scientific applications are moving towards dynamic and data-driven workflows with changing resource requirements. Oftentimes, these workflows combine dynamically changing resource requirements and very large computational and throughput demands. Federated computing has been explored in various contexts and has been demonstrated to be an attractive and viable model for effectively harnessing the power offered by distributed resources. However, resources themselves may also exhibit dynamic behavior with varying capabilities, capacities, and availability over time. Hence, being able to provision an appropriate computational environment at the right time is a challenging issue that cannot be solved using classic federation models where a user is presented with a fixed set of resources. In this paper we argue that an elastic execution infrastructure based on the dynamic federation of resources can help with varying application requirements. In particular, we propose a federation architecture that combines cloud abstractions with ideas from software-defined environments to offer users an integral solution for solving large scale problems in dynamically federated cyber-infrastructure.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems

Keywords

Software-defined systems, Autonomic computing, Cloud Federation

BigSystem 2014, June 23, 2014, Vancouver, BC, Canada.
Copyright 2014 ACM 978-1-4503-2909-5/14/06 ...$15.00.
http://dx.doi.org/10.1145/2609441.2609641.

1. INTRODUCTION

In recent years the idea of using resource federations to solve complex computational problems has emerged as a promising alternative to traditional HPC [1, 4, 7, 8, 10, 5]. Two drastically different grid computing strategies have been proposed: volunteer computing, which harvests donated, idle cycles from numerous distributed workstations (e.g., BOINC [3]), and HPC grid computing, which offers a monolithic access to powerful HPC resources shared by a virtual organization (e.g., EGEE [12] or Open Science Grid [13]) However, both of these concepts have their limitations: volunteer computing is well suited for processing lightweight independent tasks in an opportunistic way, but fails when presented with traditional parallel computations. Grid computing, on the other hand, lacks the flexibility of aggregating resources on demand (without complex infrastructure reconfiguration). These limitations motivate new approaches, among them – cloud federations, which combines ideas from these two computational models with those underlying clouds (e.g., elasticity or on-demand provisioning). Cloud federations are being explored as a means to extend as-a-service models to virtualized data-centers federations. Clouds offer on-demand access to computing utilities, an abstraction of unlimited computing resources, customizable environments, and a pay-as-you-go business model. They have a potential for scaling-up, scaling-down and scaling-out as needed, and for IT outsourcing and automation. By federating private and public clouds with local data centers and HPC systems, it is possible to create hybrid cloud infrastructures. This creates novel marketplaces where users can take advantage of different types of resources, quality of service (QoS), geographical locations, and pricing models.

In our previous work [5], we presented a federation model that enables the dynamic creation of federated "Cloud-of-Clouds" by empowering users with simple mechanisms to quickly federate resources without third-party intervention. One important feature of our federation model is the ability to scale across institutional and geographic boundaries. Oftentimes, a single resource is not sufficient to execute a given scientific workload (e.g., because it is of limited scale, it does not match application requirements, or it does not have enough storage capacity). Therefore, scaling up/down or out as needed and to most appropriate resources, becomes essential for dynamic workloads.

Despite the success of our federated model in solving a representative large-scale computational engineering problem [5], we identified the lack of automation in the federation process as a key issue that could potentially decrease the usability of the federation model and impact the efficient usage of resources. In this paper, we consider two scenarios in which having the means to automatically customize the federations can alleviate this issue. The first scenario involves dealing with highly heterogeneous applications and dynamic federations of resources, such as in the case of the environment presented in [5]. During this particular experiment, the user had to actively interact with the federation in order to take an operational decision. The user had to actively monitor a number of things, such as: the progress of the execution, to decide if more resources were needed; the status of the resources, to decide whether the resource was ready to be federated and to perform computation; the amount of allocation units, to federate out a resource after a certain number of core/hours were spent; or the failure rate of each resource, to temporarily disable those resources that were experiencing problems. The second scenario is when a user wishes to have a customized version of the federation for different projects or situations. Although our federation model allows them to take advantage of the collective power these systems offer, it is the autonomic scheduler that decides which resources to use and when, based on available resources and user's objectives. However, users cannot automatically customize the view that the autonomic scheduler has of the federation under different circumstances, for example, when needing to automatically limit the usage of a particular resource for each one of the different projects based on users' preferences. As a result, a fully customizable and autonomous federation with such added flexibility is highly desirable.

In this paper we propose using software-defined systems techniques with the purpose of providing users with the capability to programmatically define what their federations should be like under different circumstances. Our goal is to create a nimble and dynamically programmable environment that automatically evolves over time, adapting to changes in both the federated infrastructure and the application requirements. On the one hand, software-defined environments allow users to describe their expectations or objectives at the resource level in a systematic way, which in turn drives automation of the infrastructure. On the other hand, our existing autonomic scheduling capabilities [6] allow users to define objectives and policies from the application perspective, which drives the provisioning of resources – from those available in the federated infrastructure – to satisfy application requirements.

The rest of the paper is organized as follows. In Section 2 we present a motivating scenario. Next, in Section 3 we present the vision of our federation model and the conclusions in Section 5.

2. MOTIVATING USE CASE

We have selected one of our recent large-scale experiments as a driving use case to design a software-defined federation model [5]. In this experiment we faced two main challenges: heterogeneity of application requirements, and dynamic availability of heterogeneous resources.

Application Requirement: The engineering problem in this experiment was understanding fluid flow in microdevices [2]. This problem is a representative use case of a broad class of problems, which includes the analysis of high-dimensional parameter spaces, uncertainty quantification by stochastic sampling, or statistical significance assessment through resampling. These "ensemble" applications usually consist of a set of heterogeneous computationally intensive, and independent or loosely coupled tasks, and can easily consume millions of core-hours on any state-of-the-art HPC resource. While many of these problems are conveniently parallel, their collective complexity exceeds computational time and throughput that average user can obtain from a single computational center.

For the particular case of the fluid flow in microdevices, the end-user developed a parallel, finite element and MPI-based Navier-Stokes equation solver, which can be used to simulate flows in a microchannel with an embedded pillar obstacle. Here, the microchannel with the pillar is a building block that implements a fluid transformation. For a given combination of microchannel height, pillar location and diameter, and Reynolds number (4 variables), the solver captures both qualitative and quantitative characteristics of flow. In order to reveal how the input parameters interplay, and how they impact flow, the end-user seeks to construct a phase diagram of possible flow behaviors. In addition, the end-user would like to create a library of single pillar transformations to enable analysis of sequences of pillars. This amounts to interrogating the resulting 4D parameter space, in which a single point is equivalent to a parallel Navier-Stokes simulation with a specific configuration. This problem is challenging for several reasons. The search space consists of tens of thousands of points, individual simulations are highly heterogeneous, and the computational cost is very difficult to estimate a priori as it depends on the resolution and mesh density required. Specifically, the computational cost of each point can be from 100 to 100,000 core-hours, even when executed on a state-of-the-art HPC cluster. Consequently, scheduling and coordination of the execution cannot be performed manually, and a single system cannot support it. Finally, because the non-linear solver is iterative, it may fail to converge for some combinations of input parameters, in which case fault-tolerance mechanisms should be engaged.

In this particular case, the user's objective was to maximize capacity, hence the application used all the resources available in the federation at any given time. In other cases the objective function to select resources could be more complex, when taking into account factors such as locality, specific capabilities, and cost. This leads us to the second challenge – how to determine which resources are part of the federation.

Resource availability: In order to solve the engineering problem described above, we used an elastic federation model that could dynamically evolve in terms of size and capabilities [5]. During the progress of the experiment, we opportunistically federated resources as they became available and released those resources that experienced problems or ran out of allocations (see in Figure 1). Specifically, we federated 10 different resources, provided by six institutions from three countries. The experiment lasted 16 days and executed a total of 12,845 tasks. Together, all tasks consumed 2,897,390 core/hours, and generated 398 GB of the output data. Thanks to the extreme flexibility of our federation model, we were able to sustain computational performance. Figure 1 shows that most of the time anywhere between 5 and 25 MPI-simulations were running, despite multiple idle periods scattered across the majority of the machines. These idle periods were caused by common factors, such as for example, hardware failures and long waiting times in system queues.

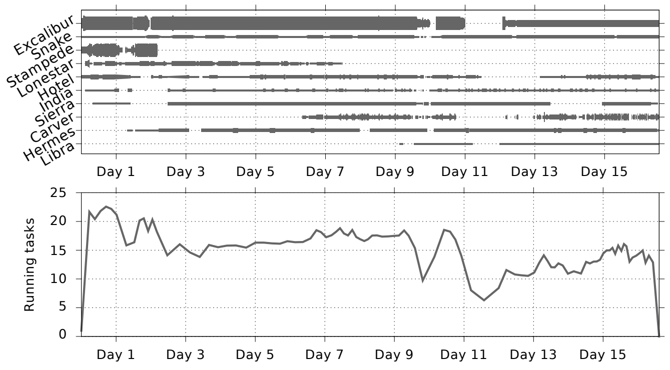

Figure 1: Summary of the experiment. Top: Utilization of different computational resources. Line thickness is proportional to the number of tasks being executed at given point of time. Gaps correspond to idle time, e.g. due to machine maintenance. Bottom: The total number of running tasks at given point of time.

As mentioned earlier, the process of federating resources was "manually" achieved by having the user actively monitoring the progress of the experiment. The user was in charge of identifying which resources were having problems as well as when they were ready to compute again; considering allocation limits; choosing the appropriate queue to maximize throughput in each particular resource; ruling out resources that were heavily loaded and could not provide the minimum required throughput; and evaluating the actual capacity of each resource. Of course, this way of "manually" shaping the federation over time is not the model that we want scientists to adopt, as it requires strong expertise in the computational domain as well as a considerable time commitment to monitor the experiment. However, this experiment clearly demonstrated the feasibility and capability of an elastic, dynamically federated cyber-infrastructure (CI). Hence, this challenge translates into defining mechanisms to allow users to describe rules and constraints that automatically drive the federation process.

3. SOFTWARE-DEFINED FEDERATION APPROACH

In this paper we propose a software-defined federation approach to empower users with the ability to independently control application execution and the federation process through relevant policies and constraints. The resulting federation acts as a living entity autonomously adapting itself to changes in the environment as well as to the application requirements.

The architecture of our software-defined federation is presented in Figure 2. In this architecture we consider two main inputs that independently describe resources and applications. We envision users using some kind of declarative language to define resource availability as well as policies and constraints to regulate their use. This approach completely abstracts users from low level details allowing them to focus on what they want rather than how to obtain it. For example, users may prefer to use a certain type of resources over others (e.g., HPC versus clouds or "free" HPC systems versus the allocation-based ones); they may want to reserve certain resources for specific projects; they may be only allowed to use certain resources at certain times of the day; they may want to use cloud resources only if they are within a desired price range; they may want to specify how to react to unexpected changes in the resource availability or performance; or they may want to only use resources within the US or Europe due to the laws regulating data movement across borders. The evaluation of these constraints provides a set of available resources at the time. We envision that by allowing users to define this kind of constraints in a formal and abstract manner, they can achieve unprecedented control over the resources, which can improve the efficiency of resource use and maximize the application's outcome.

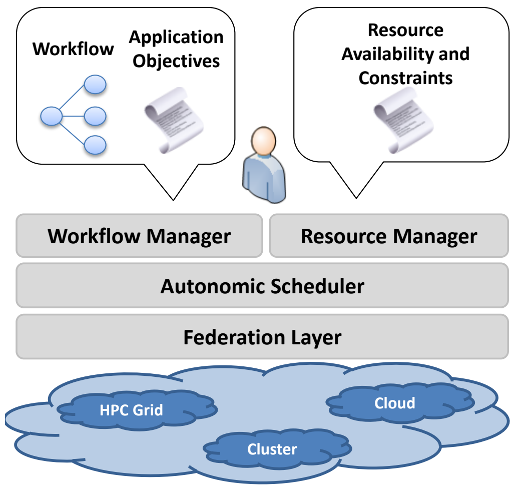

Figure 2: Software-defined Federation Architecture

The other important input is the application workflow and user's objectives. Many scientists rely on workflows to automate the different processes involved in their applications. These workflows translate a scientific problem into a semi-automated computational solution. A workflow could be described using an XML document that contains the properties of the comprising stages. Each stage defines an application to use; the location of the the input and output data; the dependencies with other stages; and the scheduling policy that will drive the execution [6].

Once the user has introduced application and resource information, the autonomic scheduler can decide which resources, from those marked as available, are the most suitable to execute the workflow. Scheduling decisions may be revised to adapt the federation capacity and capabilities to changes in the resource availability and application needs. After the scheduling decision is made, the autonomic scheduler dynamically allocates the required resources using the mechanisms provided by the federation layer. We plan to build on top of our CometCloud-based federation model [5, 9, 6] to leverage all the low level mechanisms such as, interoperability, resilience, and fault tolerance. This federation layer exposes resources using cloud-like abstractions and mechanisms that facilitate the execution of applications across the resources. The federation is dynamically created in a collaborative way, where sites "talk" to each other to identify themselves, negotiate the terms of adhesion, discover available resources, and advertise their own resources and capabilities. Sites can join and leave at any point. Notably, this requires a minimal configuration at each site that amounts to specifying the available resources, a queuing system or a type of cloud, and credentials. As a part of the adhesion negotiation, sites may have to verify their identities using security mechanisms such as X.509 certificates, or public/private key authentication.

4. TOWARDS A SOFTWARE-DEFINED CYBER-INFRASTRUCTURE PLATFORM FOR SCIENCE

To deliver a environment where users can easily integrate their applications as well as programmatically specify their desired federated resources, we are developing a Platforms as a Service (PaaS). Providing a PaaS is a way of decoupling applications from the underlying infrastructure, unrestricting these applications while offering a unified vision of the resources. Specifically, we are developing a platform that enables users to build, deploy and run applications on a hybrid software-defined cyber-infrastructure (CI). Such a platform should also provide the necessary mechanisms, exported in a semantically meaningful way, to allow users to take advantage of Cloud attributes such as elasticity. For example, elastic scale up, down or out may be defined in terms of application-specific aspects such as resolution or error thresholds.

Three main components are needed to achieve such a PaaS for Science and Engineering: (1) an API for building new applications or application workflows; (2) a descriptive language and mechanisms to create customized views of the federation that satisfy users' preferences as well as resource constraints (e.g., availability, access, etc.); and (3) a scalable middleware that masks the underlying hardware from users, while exporting Cloud abstractions such as elasticity and on-demand access in a semantically meaningful way.

We build on top of the CometCloud framework [11]. Here, CometCloud provides basic functionality, such as, autonomic capabilities, workflow management, fault tolerance mechanisms and transparent access to cloud, grid, and HPC infrastructures. Currently, Cometcloud is able to process a workflow description and autonomously orchestrate the execution of such a workflow by elastically composing appropriate cloud services and capabilities to ensure that the user's objectives are met. Moreover, CometCloud defines a flexible API that supports several common programming paradigms such as, e.g. Master/Worker, Map/Reduce and Bag-of-Tasks.

Since components (1) and (3) of our PaaS are covered by CometCloud, we are currently focusing on the component (2). This component has to interface with CometCloud to make sure that the autonomic scheduler takes into consideration both application requirements and resources constraints.

5. CONCLUSIONS

In this paper we introduced a federation model that combines the elasticity of clouds with the flexibility of software-defined systems to create a federation cyber-infrastructure (CI) on demand to tackle large-scale scientific and engineering problems. We foresee that a software-defined CI can enable a dramatic simplification in the resource management and usage. As in the case of clouds that revolutionized the way resources are offered to users, we envision software-defined environment as a game changer in the way federations are built nowadays, especially, if we combine it with other approaches such as software-defined networks to customize and optimize the communication channels.

Acknowledgments: The research presented in this work is supported in part by NSF via grants numbers OCI 1339036, OCI 1310283, DMS 1228203, IIP 0758566 and by an IBM OCR and Faculty Award.

6. REFERENCES

[1] G. Allen and D. Katz. Computational science, infrastructure and interdisciplinary research on university campuses: Experiences and lessons from the center for computation & technology. Technical Report CCT-TR-2010-1, Louisiana State University, 2010.

[2] H. Amini, E. Sollier, M. Masaeli, et al. Engineering fluid flow using sequenced microstructures. *Nature Communications*, 2013.

[3] D. P. Anderson. Boinc: A system for public-resource computing and storage. In *5th IEEE/ACM International Workshop on Grid Computing*, 2004.

[4] F. Berman, G. Fox, and A. Hey. *Grid Computing: Making the Global Infrastructure a Reality.* John Wiley & Sons, 2003.

[5] J. Diaz-Montes, Y. Xie, I. Rodero, and et. al. Exploring the use of elastic resource federations for enabling large-scale scientific workflows. In *Proc. of Workshop on Many-Task Computing on Clouds, Grids, and Supercomputers (MTAGS)*, 2013.

[6] J. Diaz-Montes, M. Zou, R. Singh, S. Tao, and M. Parashar. Data-driven workflows in multi-cloud marketplaces. In *IEEE Cloud 2014*, Accepted-2014.

[7] G. Garzoglio, T. Levshina, M. Rynge, et al. Supporting shared resource usage for a diverse user community: the OSG experience and lessons learned. *J. of Physics: Conf. Series*, 396, 2012.

[8] M. Parashar and C. Lee. Special issue on grid computing. *Proceedings of the IEEE*, 93(3), 2005.

[9] I. Petri, T. Beach, M. Zou, and et. al. Exploring models and mechanisms for exchanging resources in a federated cloud. In *Intl. Conf. on cloud engineering (IC2E)*, 2014.

[10] P. Riteau, M. Tsugawa, A. Matsunaga, et al. Large-scale cloud computing research: Sky computing on FutureGrid and Grid'5000. In *ERCIM News*, 2010.

[11] CometCloud Project. http://www.cometcloud.org/.

[12] EGEE. http://www.eu-egee.org/.

[13] Open Science Grid. https://www.opensciencegrid.org/.