Federating Advanced Cyberinfrastructures with Autonomic Capabilities

Javier Diaz-Montes, Ivan Rodero, Mengsong Zou, and Manish Parashar

Abstract Cloud computing has emerged as a dominant paradigm that has been widely adopted by enterprises. Clouds provide on-demand access to computing utilities, an abstraction of unlimited computing resources, and support for on-demand scale up, scale down and scale out. Clouds are also rapidly joining high performance computing system, clusters and grids as viable platforms for scientific exploration and discovery. Furthermore, dynamically federated Cloud-of-Clouds infrastructure can support heterogeneous and highly dynamic applications requirements by composing appropriate (public and/or private) cloud services and capabilities. As a result, providing scalable and robust mechanisms to federate distributed infrastructures and handle application workflows, that can effectively utilize them, is critical. In this chapter, we present a federation model to support the dynamic federation of resources and autonomic management mechanisms that coordinate multiple workflows to use resources based on objectives. We demonstrate the effectiveness of the proposed framework and autonomic mechanisms through the discussion of an experimental evaluation of illustrative use case application scenarios, and from these experiences, we discuss that such a federation model can support new types of application formulations.

1 Introduction

Cloud computing is revolutionizing the enterprise world, much as the Internet did not so long ago. Clouds are fundamentally changing how enterprises think about IT infrastructure, both internally and externally, by providing on-demand access to always-on computing utilities, an abstraction of unlimited resources, a potential for scale-up, scale-down and scale-out as needed, and for IT outsourcing and automation. Clouds also provide a usage-based payment model where users

J. Diaz-Montes (✉) • I. Rodero • M. Zou • M. Parashar
Rutgers Discovery Informatics Institute, NSF Cloud and Autonomic Computing Center, Department of Electrical and Computer Engineering, Rutgers University, 96 Frelinghuysen Road, Piscataway, NJ 08854, USA
e-mail: javidiaz@rdi2.rutgers.edu; irodero@rutgers.edu; mengsong.zou@rutgers.edu; parashar@rutgers.edu

X. Li, J. Qiu (eds.), *Cloud Computing for Data-Intensive Applications*,
DOI 10.1007/978-1-4939-1905-5_9

essentially "rent" virtual resources and pay for what they use. Underlying these cloud services are typically consolidated and virtualized data centers that exploit economies of scale to provide attractive cost–benefit ratios. In spite of being in its early stages, cloud computing is already reshaping IT world. In fact, according to The Wall Street Journal, four out of five businesses are moving or planning to move some of their business functions to cloud services.

At the same time that cloud computing is redefining IT, extreme data and compute scales are transforming science and engineering research by enabling new paradigms and practices—those that are fundamentally information/data-driven and collaborative. Cloud abstractions and infrastructures are rapidly becoming part of the overall research cyberinfrastructure, providing viable platforms for scientific exploration and discovery. It is expected that cloud services will join more traditional research cyberinfrastructure components—such as high performance computing (HPC) system, clusters and grids, as part of Cyberinfrastructure Framework for Twenty-First Century Science and Engineering (CIF21) in supporting scientific exploration and discovery. Analogous to their role in enterprise IT, clouds can enable the outsourcing of many of the mundane and tedious aspects research and education, such as deploying, configuring and managing infrastructure, and enable scientists to focus on the science. Computational and Data-enabled Science and Engineering (CDS&E) applications enabled by an advanced cyberinfrastructure (ACI) are providing unprecedented opportunities for understanding and managing natural and engineered systems, and offering unique insights into complex problems and, in addition to support traditional enterprise data analytics services (e.g., those based on MapReduce). For example, clouds can provide a platform for applications when local infrastructure is not available or supplement existing platforms to provide additional capacity or complementary capabilities to meet heterogeneous or dynamic needs [20, 42, 66]. Clouds can also serve as accelerators, or provide resilience to scientific workflows by moving the execution of the workflow on alternative or fewer resources when a failure occurs. The simplicity of the cloud abstraction can alleviate some of the problems scientific applications face in current HPC environments. The analysis of high-dimensional parameter spaces, uncertainty quantification by stochastic sampling, or statistical significance assessment through resampling, are just few examples of a broad class of problems that are becoming increasingly important in a wide range of application domains. These applications can be generally described as many task computing applications [48] and can benefit from the easy access to on-demand elastic customizable resources and the ability to easily scale up, down or out [49]. Clearly, realizing these benefits requires the development of appropriate application platforms and software stacks.

In this chapter, we present a model to support the dynamic federation of resources and the coordinated execution of application workflows on such federated environments. These resources can be of different types of infrastructure including traditional HPC clusters, supercomputers, grids, and clouds. Additionally, the federation provides autonomic scheduling mechanisms that create an abstraction with cloud-like capabilities to elastically provision the resources based on user and application policies and requirements. We discuss the requirements to enable

our federation model followed by the description of our federation model and mechanisms. In contrast to previous work such as [52] or [10], which propose models to federate and combine clouds with local resources for cloudbursting, this chapter focuses on providing abstractions to seamlessly federate and provision on-demand a wider range of resources such as high-end systems that are not typically exposed in federated systems or grids. A key aspect of our federation model is the autonomic management and optimization of application execution through cross-layer application/infrastructure adaptation. To demonstrate the effectiveness of the federation model, mechanisms and autonomic management policies we present an experimental evaluation with relevant usage scenarios of the proposed framework, including: (1) medical image research, which aims at achieving extended capacity, (2) molecular dynamics simulations using asynchronous replica exchange, which provides adaptivity and elasticity at the application-level, and (3) data analytics workflow based on clustering, which focuses on adaptation to achieve user objectives and requirements. From these use case applications, we discuss ongoing work towards enabling such a federation model to support new types of application formulations such as adaptive workflows where dynamic provisioning and federation is essential to respond to non-deterministic behaviors.

2 State of the Art

This section collects different research efforts aimed to federate resources in the context of grid and cloud computing as well as standards to ease the interoperability among infrastructures. These efforts are mainly focused on providing an infrastructure to compute large scale scientific applications.

2.1 Federating Computational Grids

In the late 1990s, grid computing [16] emerged as the model to support large scientific collaborations by providing their computational resources and the structure behind them. The core concept of grid computing defines an architecture to support shared access to resources provided by members of virtual organizations (VO) [17] that are formed by collaborative data centers and institutions. Some examples of grids are Open Science Grid (OSG) [88] in US, GridX1 in Canada [1], Naregi in Japan [83], APACGrid in Australia [12], Garuda in India [50], Grid'5000 in France [6,78], DAS-3 in the Netherlands [73], D-Grid in Germany [75], e-Science in UK [21], and EGI (following EGEE and DataGrid research efforts) in Europe [76]. Note that the majority of grids result from regional initiatives. However, large dedicated grids have been also built to serve as scientific instruments, such as XSEDE [91] in the US, DEISA [74], HPC-Europa [41] and PRACE [89] in the EU, OSG in US, EGI in Europe, and Grid'5000 in France.

As science was pushing new limits in terms of levels of computation and data and collaboration between scientists from multiple scientific domains across the globe, there was a need for interoperability among different grid systems to create large grid environments that would allow users to access resources of various VOs transparently [54]. Among the grid federation efforts we can find InterGrid [9] along with the work by Assuncao et al. [3] that promotes interlinking different grid systems through economic-based peering agreements to enable inter-grid resource sharing, Gridway [67] through its grid gateways [22] along the work by Leal et al. [35] that proposed a decentralized model for scheduling on federated grids to improve makespan and resource performance, LAGrid meta-scheduling [5, 55, 59] that promotes interlinking different grid systems through peering agreements to enable inter-Grid resource sharing, Koala [39] with the use of delegated matchmaking [24] to obtain the matched resources from one of the peer Koala instances, VIOLA [61] that implements grid interoperability via WS-Agreement [2] and provides co-allocation of multiple resources based on reservations, Grid Meta-Brokering Service (GMBS) [28, 29] proposes an architecture for grid interoperability based on high level abstractions to describe the broker's capabilities and properties using a specific language [30–32, 57], the work by Elmroth et al. [13] that presents a grid resource brokering service based on grid standards, Guim et al. [56] studied scheduling techniques for multi-site grid environments, and within EGEE, efforts to enable interoperability between gLite and UNICORE [14] systems [38, 51].

2.2 Federation in Cloud Computing

Cloud computing has emerged as a dominant paradigm that has been widely adopted by enterprises. Clouds provide on-demand access to computing utilities, an abstraction of unlimited computing resources, and support for on-demand scale up, scale down and scale out. Furthermore, dynamically federated "cloud-of-clouds" infrastructure can support heterogeneous and highly dynamic applications requirements by composing appropriate (public and/or private) cloud services and capabilities. At the same time that cloud computing is redefining IT, it is rapidly joining high-performance computing system, clusters and grids as viable platforms for scientific exploration and discovery. Current cloud platforms can provide effective platforms for certain classes of applications, for example high-throughput computing (HTC) applications. There have been several early projects that have reported successful deployments of applications on existing clouds [11, 18, 27, 68]. Additionally, there are efforts exploring other usage modes [43] and to combine clouds, such as Amazon EC2 [71], with integrated computing infrastructures. Villegas et al. [69] proposed a composition of cloud providers as an integrated (or federated) cloud environment in a layered service model. Assuncao et al. [10] described an approach of extending a local cluster to cloud resources using different scheduling strategies. Along the same lines, Ostermann et al. [42] extended a grid workflow application development and computing infrastructure to include cloud

resources, and experimented with Austrian Grid and an academic cloud installation of Eucalyptus using a scientific workflow application. Similarly, Vazquez et al. [66] proposed architecture for an elastic grid infrastructure using the GridWay meta-scheduler, and extended grid resources to Globus Nimbus; Vockler et al. [70] used Pegasus and Condor to execute an astronomy workflow on virtual machine resources drawn from multiple cloud infrastructures based on FutureGrid, NERSC's Magellan cloud and Amazon EC2; Gorton et al. [20] designed a workflow infrastructure for Systems Biology Knowledgebase (Kbase) and built a prototype using Amazon EC2 and NERSC's Magellan cloud; and Bittencourt et al. [4] proposed an infrastructure to manage the execution of service workflows in the hybrid system, composed of the union of a grid and a cloud.

Given the growing popularity of virtualization, many commercial products and research projects, such as OpenNebula [62, 86], OpenStack [87], Nimbus [84], Eucalyptus [40, 77], IBM Smart Cloud [80], Amazon EC2, and VMware vCloud Connector are being developed to dynamically overlay physical resources with virtual machines. Analogously, Riteau et al. [52] proposed a computing model where resources from multiple cloud providers are leveraged to create large-scale distributed virtual clusters. They used resources from two experimental testbeds, FutureGrid in the United States and Grid'5000 in France. In [8], Celesti et al. proposed a cross-federation model based on using a customized cloud manager component placeable inside the cloud architectures. Other example is the Resevoir [53] that aims at contributing to best practices with a cloud and federation architecture. In general, these efforts are intended to extend the benefits of virtualization from a single resource to a pool of resources, decoupling the VM not only from the physical infrastructure but also from the physical location.

2.3 Interoperability Standardization Activities

There are several projects with the goal enabling the interoperability of federated infrastructures. The Open Middleware Infrastructure Institute for Europe (OMII-Europe) aims to significantly influence the adoption and development of open standards that facilitate interoperability between gLite and UNICORE such as OGSA Basic Execution Service (BES) or Job Submission Description Language (JSDL). The Grid Scheduling Architecture Research Group (GSA-RG) of Open Grid Forum (OGF) is currently working on enabling grid scheduler interaction. They are working to define a common protocol and interface among schedulers enabling inter-grid resource usage, using standard tools (e.g., JSDL, OGSA, WS-Agreement). However, the group is paying more attention to agreements. They proposed the Scheduling Description Language (SDL) to allow specification of scheduling policies based on broker scheduling objectives/capabilities (e.g., time constraints, job dependencies, scheduling objectives, preferences). The Grid Interoperation Now Community Group (GIN-CG) of the OGF also addresses the problem of grid interoperability driving and verifying interoperation strategies. They

are more focused on infrastructure with five sub-groups: information services, job submission, data movement, authorization, and applications. Aligned with GIN-CG, the OGF Production Grid Infrastructure Working Group (PGI-WG) aims to formulate a well-defined set of profiles and additional specifications. Some recommendations of these initiatives have been considered in existing work which has identified standardization as a key element towards interoperability [15].

There are also two main activities of the OGF for job management: SAGA [19] and DRMAA (Distributed Resource Management Application API) [65]. SAGA provides a set of interfaces used as the application programming model for developing applications for execution in grid environments. DRMAA defines a set of generalized interfaces that applications used to interact with distributed resource management middleware. Both SAGA and DRMAA focus on applications.

Finally, there are other interoperability activities focused in the context of the cloud. We have Siena [90], Open Cloud Computing Interface (OCCI) [85], under the OGF umbrella, that aim at defining standards for cloud interoperability. There is an IEEE Intercloud WG Working Group [81] that is working in standards such as Standard for Intercloud Interoperability and Federation (SIIF) [82].

3 Federation Model to Aggregate Distributed Resources

The federation model that we propose is aimed to orchestrate geographically distributed resources using cloud-like capabilities and abstractions. Our proposed federation model is different from the existing ones, presented in Sects. 2.1 and 2.2, in the sense that we provide a platform to access federated resources using cloud-like capabilities such as on-demand provisioning, dynamic aggregation or cloudbursting. Moreover, we are able to federate various kind of resources (HPC, cloud, and grid) and enable autonomic computing features such as objective-driven workflow execution to efficiently compute large scale problems.

3.1 Requirements

In order to design a federation model to support large scientific and engineering problems, it is imperative to clearly define the characteristics that the resulting system should provide. Having a well determined set of necessary and sufficient requirements simplifies the design process by focusing on the essential functionality. Thus, we used our past and present collaborations with domain scientists to identify key requirements our solution should offer in order to be easy to use and flexible. Next, we describe each one of these requirements.

- **Scalability and Extended Capacity:** Due to the computational requirements of modern scientific applications, it becomes necessary to scale across

geographicallydistributed resources. This is because oftentimes a single resource is not sufficient to execute a given scientific workload (e.g. because the resource is of limited scale, or it mismatches application requirements).

- **Interoperability:** While the scalability and extended capacity requirement ensures that diverse resources can be incorporated into the federation, the interoperability requirement guarantees that the federation will be able to interact with these resources. Specifically, the federation must offer mechanisms to interface with common platforms such as personal supercomputers, MPI and MapReduce clusters, massively parallel and shared memory supercomputers, and clouds. At the same time, it must be open such that new platforms can be added in the future.
- **Capability:** By having heterogeneous resources as part of the federation, we can take advantage of their particular characteristics and optimize the resource allocation. In our model tasks and resource allocation can be achieved via a push model with central scheduler, or a pull model where resources obtain tasks via attribute-based queries. Thus, the federation must be aware of the capabilities of each resource to allow optimal usage.
- **Elasticity and On-Demand Access:** The important factor affecting applicability of the federation is its ability to scale up/down or out as needed. For many practical workloads it is difficult to predict computational and storage requirements. Moreover, many applications are dynamic in the sense of convergence, and hence provide no guarantees on the cost of execution. Consequently, the federation must be able to aggregate or drop resources seamlessly. What is important, the resulting elasticity makes the infrastructure resilient and hence improves its ability to sustain computational throughput.
- **Self-discovery:** Having the right monitoring mechanisms in place is important to ensure that the federation provides a realistic view of resources, taking into account their variability over time. Here multiple factors should be taken into account including availability, load, failure-rate, etc. The ability to self-discover strongly affects how the federation manages the offered services and optimizes resources allocation.
- **Democratization:** Users of the federation may have access to a larger number of resources or to specific resources, which enables them to tackle more important scientific challenges. This requires the capability of sharing resources and more importantly controlling their usage to ensure a fair use among all users.

3.2 Federation Architecture

The federation is designed to be dynamically shaped as it is created in a collaborative way, where each site talk with each other to identify themselves, negotiate the terms of adhesion, discover available resources, and advertise their own resources and capabilities. In this way, a federated management space is created on runtime and sites can join and leave at any point. Users can access the federation from any site, see Fig. 1.

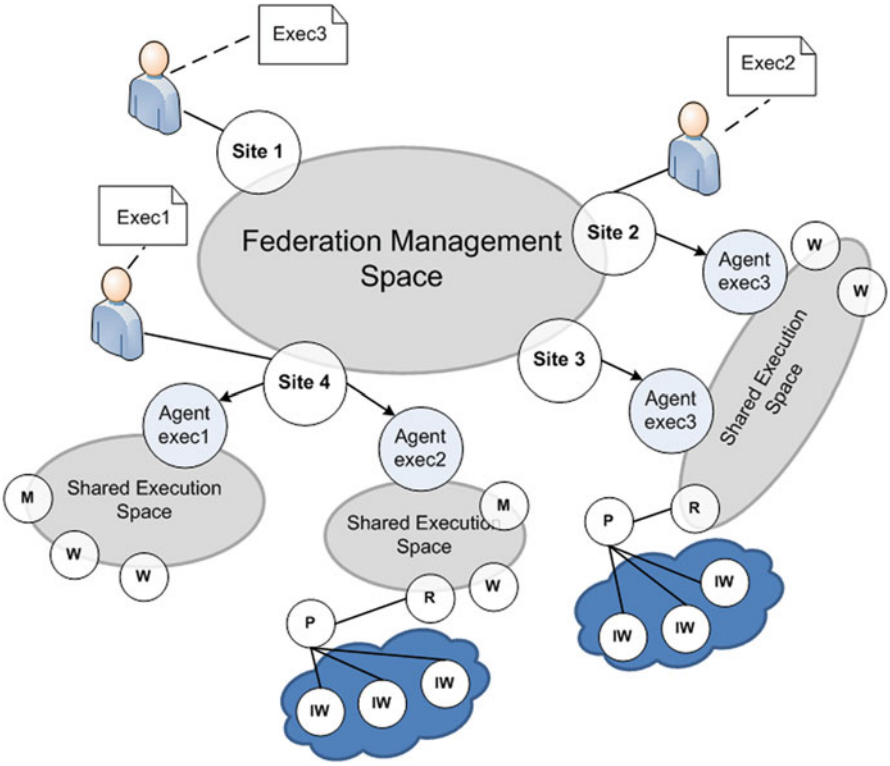

Fig. 1 Federation architecture. Here (M) denotes a master, (W) is a worker, (IW) an isolated worker, (P) a proxy, and (R) is a request handler

The federation model is based on the Comet [36] coordination spaces concept. These Comet spaces are used to coordinate the different aspects of the federation. In particular, we have decided to use two kind of spaces in the federation. First, we have a single federated management space used to create the actual federation and orchestrate the different resources. This space is used to interchange any operational message for discovering resources, announcing changes in a site, routing users' request to the appropriate sites, or initiating negotiations to create ad-hoc execution spaces. On the other hand, we can have multiple shared execution spaces that are created on demand to satisfy computing needs of the users. Execution spaces can be created in the context of a single site to provision local resources and cloudburst to public clouds or external HPC systems. Moreover, they can be used to create a private sub-federation across several sites. This case can be useful when several sites have some common interest and they decide to jointly target certain type of tasks as a specialized community.

As shown in Fig. 1, each shared execution space is controlled by an agent that creates such space and coordinates the resources for the execution of a particular

set of tasks. Agents can act as master of the execution or delegate this duty to a dedicated master (M) when some specific functionality is required. Moreover, agents deploy workers to actually compute the tasks. These workers can be in a trusted network and be part of the shared execution space, or they can be part of external resources such as a public cloud and therefore in a non-trusted network. The first type of workers are called secure workers (W) and can pull tasks directly from the space. Meanwhile, the second type of workers are called isolated workers (IW) and cannot interact directly with the shared space. Instead, they have to interact with a proxy (P) and a request handler (R) to be able to pull tasks from the space.

A key aspect of this federation is the autonomic management and optimization (of multiple objectives, including performance, energy, cost, and reliability) of application execution through cross-layer application/infrastructure adaptations. It is essential to be able to adapt to the application's behavior as well as system configuration, which can change at run time, using the notion of elasticity at the application and workflow levels. Hence, the federated infrastructure increases the opportunities to provision appropriate resources for given workflows based on user objectives or policies and different resource classes can be mixed to achieve the user objectives. Resources scale up/down/out based on the dynamic workflow and the given policies. A user objective can be to accelerate application runtime within a given budget constraint, to complete the application in a time constraint, or to select better resources matching to the application type, such as computation-intensive and data-intensive. Furthermore, application requirements and resource status may change, for example, due to workload surges, system failures or emergency system maintenance, and as a result, it is necessary to adapt the provisioning to match these changes in resource and application workload.

3.3 CometCloud

Our federation model is built on top of CometCloud [33, 72] and the concepts that CometCloud is based on. CometCloud is an autonomic computing engine based on the Comet [36] decentralized coordination substrate, and supports highly heterogeneous and dynamic cloud/grid/HPC infrastructures, enabling the integration of public/private clouds and autonomic cloudbursts, i.e., dynamic scale-out to clouds to address extreme requirements such as heterogeneous and dynamics workloads, and spikes in demands.

Conceptually, CometCloud is composed of a programming layer, service layer, and infrastructure layer. The infrastructure layer uses the Chord self-organizing overlay [63] and the Squid [60] information discovery to create a scalable content-based coordination space for wide-area and a content-based routing substrate, respectively. The routing engine supports flexible content-based routing and complex querying using partial keywords, wildcards, or ranges. It also guarantees that all peer nodes with data elements that match a query/message will be located. The service layer provides a range of services to support autonomics at the programming

and application level. This layer supports a Linda-like [7] tuple space coordination model, and provides a virtual shared-space abstraction as well as associative access primitives. Dynamically constructed transient spaces are also supported to allow applications to explicitly exploit context locality to improve system performance. Asynchronous (publish/subscribe) messaging and event services are also provided by this layer. The programming layer provides the basic functionality for application development and management. It supports a range of paradigms including the master/worker/BOT. Masters generate tasks and workers consume them. Masters and workers can communicate via virtual shared space or using a direct connection. Scheduling and monitoring of tasks are supported by the application framework. The task consistency service handles lost/failed tasks.

3.4 Autonomic Management

The autonomic management capabilities are provided by the autonomic manager, which is responsible for managing workflows, estimating runtime and scheduling tasks at the beginning of every stage based on the resource view provided by the agents. At each stage, the adaptivity manager monitors tasks runtimes through results, handles the changes of application workloads and resource availability, and adapts resource provisioning if required. Figure 2 shows the architecture of the autonomic management framework. We detail the different components of the autonomic manager below.

Workflow Manager The workflow manager is responsible for coordinating the execution of the overall application workflow, based on user's objectives and status of the infrastructure.

Runtime Estimator The runtime estimator estimates computational runtime and cost of each task. This estimate can be obtained through a computational complexity model or through quick, representative benchmarks. Since performance is strongly affected by the underlying infrastructure (clouds, HPC, or grids) it is more effective to use benchmarks to obtain runtime and cost estimates.

Autonomic Scheduler It uses the information provided by the estimator modules to determine the initial hybrid mix HPC/grids/cloud resources based on user/system-defined objectives, policies and constraints. The autonomic scheduler also profiles the tasks to allow agents to get the most suitable ones for their resources. The scheduler is dynamic and it can update both the allocations and the scheduling policy at runtime.

Elasticity Manager The status of the resources or the performance of the application can change over time and differ from the initial estimation. Thus, the elasticity manager is responsible for preventing the violation of the objectives and policies by elastically adapting the resources allocated to each workflow.

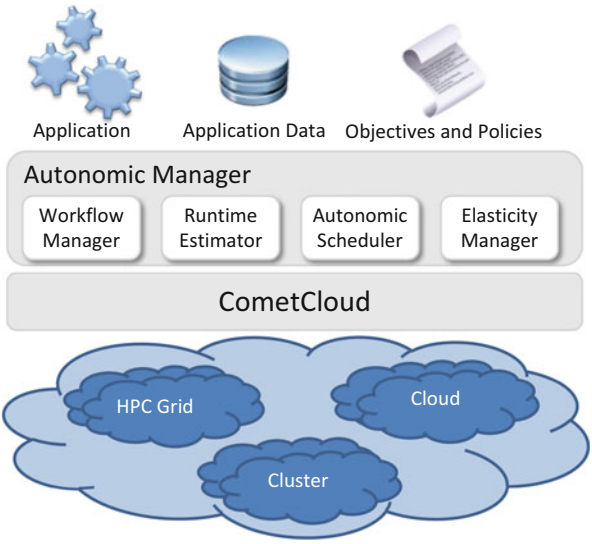

Fig. 2 Architectural overview of the autonomic manager framework. CometCloud creates a cloud abstraction where all types of infrastructures are viewed as elastic clouds that can provision resources on demand

The autonomic manager takes advantage of the cloud abstraction provided by CometCloud to seamlessly interact with any kind of infrastructures. Each infrastructure has specific properties that define the characteristics of its resources. This information is essential to allow the autonomic manager to dynamically federate resources. Since it uses CometCloud, it inherits the support for the master/worker, MapReduce and workflow programming models. Nevertheless, applications are usually described as workflows. Typically, the workflow programming model considers a workflow with multiple stages, where stages should be executed in an order, each stage can run a different application or the same application with different length of tasks, computational requirements, and data.

3.5 Enabling Autonomics

The essence of the autonomic manager resides in the user objectives and policies. They are used to drive the execution of the workflow by provisioning the appropriated number and type of resources. The allocated resources can vary over time, to make sure the application requirements are respected, if a deviation over the estimate execution plan is detected. Deviations on the plan occur due to unexpected failures, performance fluctuation, queue wait time variation, etc. In this Section we present several use cases that represent typical scenarios from the user's perspective and how to achieve them using autonomic techniques.

User Objectives Currently, the autonomic manager supports three main objectives namely acceleration, conservation, and resilience. Nonetheless, new objectives such as energy-efficiency can be easily integrated.

Acceleration Cloud infrastructures provide large amount of resources that can perfectly be used to execute certain scientific applications. Thus, they can be used to boost the execution of the applications by dramatically increasing the number of allocated resources and hence reducing the overall computational time.

Conservation HPC resources are essential to compute many scientific applications. However, the access to this type of resources is very limited and their use is typically controlled by awards. Therefore, optimizing the use of those resources is very important. The idea of this use case is to use clouds to conserve HPC allocations. For example, we could use the cloud to do an initial exploration of the application domain and migrate to the HPC resources only those tasks that progress as expected. This could be done considering runtime and budget constraints.

Resilience This use case investigates how clouds can be used to handle unexpected situations such as an unanticipated HPC/grid downtime, inadequate allocations, unanticipated queue delays or failures of working nodes. Additional cloud resources can be requested to alleviate the impact of the unexpected situations and meet user objectives.

Scheduling Policies To achieve the above user objectives, several policies can be defined. Two of the most representative policies are described as follows.

Deadline The scheduling decision is to select the fastest resource class for each task and to decide the number of nodes per resource class based on the deadline. If the deadline can be achieved with a single node, then only one node will be allocated. When an application needs to be completed as soon as possible, regardless of cost and budget, the largest useful number of nodes is allocated.

Budget When a budget is enforced on the application, the number of allocatable nodes is restricted by the budget. If the budget is violated with the fastest resource class, then the next fastest and cheaper resource class is selected until the expected cost falls within the budget limit.

4 Application Scenarios

This section provides a comprehensive discussion of different representative use case applications and experiences to illustrate the effectiveness of our proposed federation architecture, mechanisms and autonomic strategies. Specifically we discuss different Computational and Data-Enabled Science and Engineering (CDS&E) applications and an enterprise business data analytics workflow. Although the result of these applications is not the goal of this chapter, we believe the discussion of these experiences are useful to define next steps towards advanced cyberinfrastructure and clouds federation for different usage modes [43].

4.1 CDS&E Applications

Two different CDS&E applications are discussed below, namely medical image research and molecular dynamics simulations using asynchronous replica exchange. Both application use cases use federated advanced cyberinfrastructure in combination with clouds, however, they represent different usage modes.

Medical Image Research includes both medical image registration and content-based image retrieval. In the former, we focused on autonomically balancing completion time and cost using federated resources including private data centers, grids and private clouds [34]. In the latter, we use the proposed federation mechanisms to achieve extended capacity to respond to its large computational power requirements, which is described as follows.

Content-based image retrieval (CBIR) has been one of the most active research areas in a wide spectrum of image-related fields over the last few decades. In pathology, hematology already contains a large number of tools to automatically count blood cells. To classify abnormal white blood cells and compare diagnosis between a new case and cases with similar abnormalities is an interesting application. In this application use case we focus on CBIR on digitized peripheral blood smear specimens using low-level morphological features. Specifically, we use CometCloud to execute CBIR in federated heterogenous advanced cyberinfrastructure and cloud resources with the goal of reducing the completion time (i.e., provide answers within minutes or hours rather than weeks). The CBIR code was ported from Matlab to Java as a native CometCloud application to avoid licensing constrains in non-proprietary resources and to enable future implementations of the application on specialized hardware (e.g., accelerators). Since the most computation expensive part is searching query patches within each database image, we chose to use master/worker programming model, thus each image within the database was assigned to a worker. The implementation using the master/worker programming model is shown as Fig. 3. A master and a number of workers (one per physical core) form an overlay at runtime and synchronize using a tuple space (execution space). The master generates tasks (one for each image or subset of images to be processed) and then the workers pull the tasks and process the associated images simultaneously. In order to improve scalability and fault tolerance, workers store intermediate results on disk rather than returning the results back to the master using the comet space. When the workers finish, the intermediate results are consolidated (which represents a small part of the overall execution).

In order to obtain extended capacity, we federated a cluster at Rutgers (a Dell Power Edge system with 256 cores in 8-core nodes) with distributed cyberinfrastructure from NSF Extreme Science and Engineering Discovery Environment (XSEDE), NSF FutureGrid, the National Energy Research Scientific Computing Center (NERSC) and public clouds (Amazon EC2). Specifically, we used Ranger (Sun constellation with 62,976 cores in 16-core nodes) and Lonestar (with 22,656 cores in 12-core nodes) from XSEDE, Hotel (an IBM iDataPlex system with 672 cores in 8-core nodes) from FutureGrid, Hopper (a Cray XE6 system with

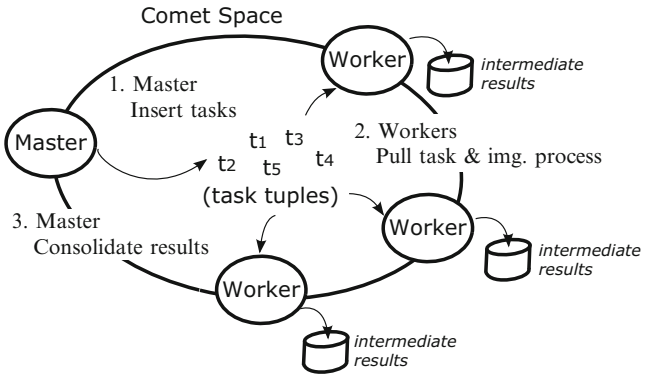

Fig. 3 Master/worker framework implementation in CometCloud

153,216 cores in 24-core nodes) from NERSC, and medium instances from Amazon EC2. The former resources were used through startup awards and the later in pay-as-you-go basis. We used advanced cyberinfrastructure systems "opportunistically" by using short waiting time queues, i.e., queues with limitations such as reduced runtime and number of queued/running jobs. Data transfer was overlapped with computation.

Figure 4 shows the completion time of CBIR algorithm over the database of 925 images for the 50 different configurations (in minutes, using logarithmic scale) and Fig. 5 shows the average throughput (i.e., processed images per minute) that a single node of each of the different platforms can achieve. Completion time for the federated scenario was obtained with real executions while for sequential and local cluster scenarios completion time is an estimation based on the actual execution of the subset of configurations, due to the limitations of very long executions.

The results show that CBIR is dramatically speeded up when using the (dedicated) dell cluster at Rutgers with respect to using a single node (from around two weeks of computation to 12 h). However, using federated infrastructure (i.e., much

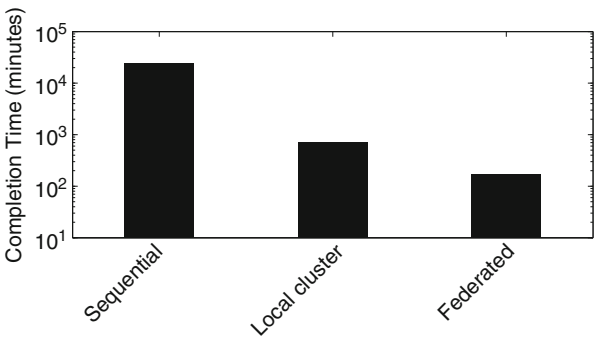

Fig. 4 Completion time using different configurations

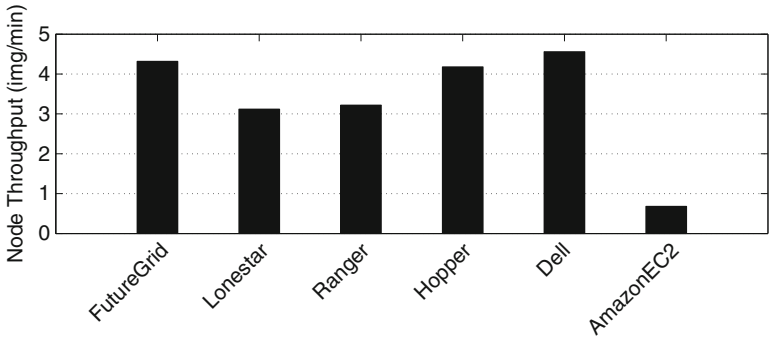

Fig. 5 Average node throughput for different platforms

more resources but not under own control) provides much shorter completion time (about 170 min). In our experiments the jobs used a single node (i.e., up to 24 cores) to run a set of 100 images, however, if we used a smaller set of images per job the penalty due to the queuing times would be higher. In case of Amazon EC2 a job processes a smaller set of images because the nodes have smaller core count. Figure 5 also shows the impact of queuing time on the average throughput in the HPC systems (i.e., Lonestar, Ranger and Hopper). Furthermore, Fig. 6 shows the throughput (i.e., number of processed images per minute) during a time interval of 3 h of the Dell cluster at Rutgers and Hopper, respectively. The Dell cluster shows more stable throughput behavior than Hopper, whose throughput presents spikes over time. Although Hopper nodes are more powerful, the queuing times and the limitation of the number of concurrent running jobs penalizes significantly the throughput.

The proposed federated system presents many opportunities and challenges in the context of medical image research such as exploiting heterogeneous federated resources from the point of view of their capabilities. For example, the Matlab incarnation of CBIR can be run when licenses are available or an incarnation for accelerators (e.g., GPU or Intel MIC) can be run when resources with accelerators are available.

Molecular Dynamics Simulations Using Asynchronous Replica Exchange Replica exchange [23,64] is a powerful sampling algorithm that preserves canonical distributions and allows for efficient crossing of high-energy barriers that separate thermodynamically stable states. In this algorithm, several copies or replicas, of the system of interest are simulated in parallel at different temperatures using "walkers". These walkers occasionally swap temperatures and other parameters to allow them to bypass enthalpies barriers by moving to a higher temperature. The replica exchange algorithm has several advantages over formulations based on constant temperature, and has the potential for significantly impacting the fields of structural biology and drug design—specifically, the problems of structure based drug design and the study of the molecular basis of human diseases associated

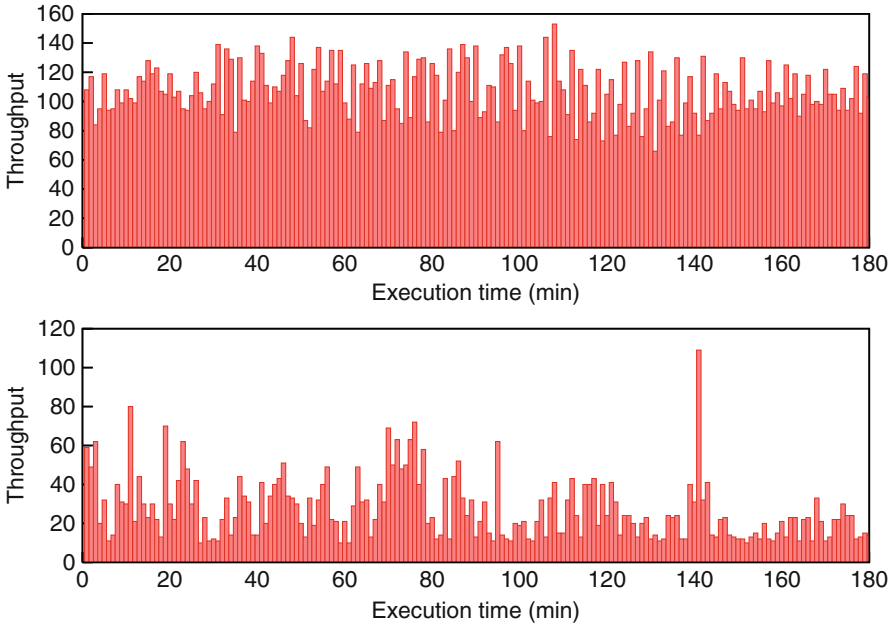

Fig. 6 Throughput (images/minute) for the Dell cluster on *top* and the Hopper supercomputer at the *bottom*

with protein misfolding. Traditional parallel implementations of replica exchange target either tightly coupled parallel systems or relatively small clusters. However, an asynchronous formulation of the replica exchange algorithm was designed and implemented [37, 79], which was proven to be effective, efficient and suitable for grid- based systems. Additionally, it is possible to reformulate the workflow to better utilize the ACI as described below.

Typically molecular dynamics simulations are very static in terms of execution models—the simulations go from start to finish irrespective of whether replicas are progressing towards correct folding. The ability to bias a trajectory by identifying pathways that are progressing towards a fully folded structure and those that are diverging away is an exciting direction for replica exchange formulations. The quality of the protein structure can be monitored by comparing the progress of each replica using secondary structure prediction methods and the radius of gyration. For example, in ubiquitin folding simulation replicas with large radius of gyration would be considered for termination because ubiquitin is a globular protein with a small radius of gyration. However, replicas exhibiting short radius of gyration would remain in the simulation due to the close resemblance to the completely folded ubiquitin protein. In addition to killing diverging replicas, the described application formulation can also spawn new replicas if they are making progress towards correct folding. As a result, the entire simulation would follow a sequence where radius of gyration and secondary structure prediction information will be used to terminate

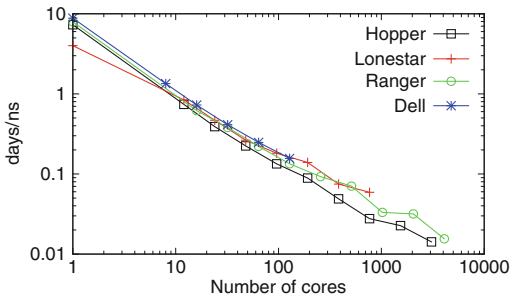

Fig. 7 NAMD scaling in different environments

some replicas, cause a conformation to spawn new replicas across a temperature range, and modulate the probability of exchanging to nearby temperatures. By utilizing such a formulation we can dynamically adapt the molecular dynamics simulations, bias trajectories to find pathways towards correct folding and, in doing so, accelerate scientific discovery.

The use of CometCloud provides the opportunity to run simulations on dynamically federated large-scale distributed resources. Thus, what we see is a hierarchical formulation that provides adaptivity and elasticity at the application-level, through asynchronous replica exchange, and at the infrastructure-level, through CometCloud. By utilizing CometCloud's capabilities and its cloud computing abstractions, we can run asynchronous NAMD [44] replica exchange on a federated, distributed environment. From an application perspective, the amount of time a simulation takes is proportional to the size of the protein or system and the desired length of the trajectory. However, by using CometCloud and asynchronous replica exchange scientists can explore the folding of very large proteins and run trajectories at microsecond or potentially even millisecond scale. This larger scale of science also gives rise to interesting scenarios at the CometCloud layer. For example, if we find that the initial allocation of resources is not enough then CometCloud can dynamically federate other distributed sites in order to obtain more resources. Conversely, in the context of protein folding, CometCloud can dynamically kill replicas if it finds that the protein structures being generated by the replicas do not progress towards the known structure or show predicted secondary structure features. By eliminating non-converging replicas we can ensure that CPU cycles are not wasted and speed-up the application.

In order for large-scale simulations to be effective the asynchronous formulation must show good scaling on this heterogeneous distributed environment. Performance evaluation is a necessary tool for understanding the limitations of the various environments provided by CometCloud. This is especially true for commodity and virtualized resources—such as those provided by FutureGrid and Amazon EC2. Thus, the performance of the entire ensemble of simulations depends on the slowest platform. In this case, the slowest platforms correspond to FutureGrid and Amazon EC2 where NAMD replica exchange is deployed on virtual machines. In terms of the simulation, the downward slope shows that the simulation time (in days) for a

nanosecond trajectory is decreasing—meaning faster simulations. From Fig. 7 we can conclude that all environments tested exhibit good scalability and consistently report faster simulation times each time the processor count is doubled. More importantly, these results provide justification for a federated architecture where all machines can run simulations in parallel. The close grouping of simulations on large-scale HPC sites (i.e., Ranger, Hopper, Lonestar) also show that distributing replicas across these environments might also be feasible. Combining the fact of the low performance of virtualized environments to run tightly couple simulations [18, 25] and NAMD scaling on HPC infrastructure, we find that there is sufficient motivation for the formulation described above.

4.2 Enterprise Business Data Analytics

Current enterprise business data analytics workflows combine different techniques in their stages such as MapReduce-like applications that aggregate large amounts of data from different sources for business intelligence with clustering techniques. For example, the output of a topic-based text analysis approach such as Latent Dirichlet Allocation (LDA) is represented in a multi-dimensional information space, which includes different topics, information categories, etc. These data points in the multi-dimensional information space can be clustered using Distributed Online Clustering (DOC) to search results and correlate them with known data sources, and allow visualizing and interpreting the results interactively through a GUI. The specific solution in this application use case is a federated hybrid cloud for handling "big data" through DOC.

DOC is a clustering algorithm that targets networked systems in which individual components can monitor their operational status or actions, represent them using sets of globally known attributes, and periodically publish this status or interactions as semantic events that contain a sequence of attribute-value pairs. The algorithm specification, along with details about its implementation and robustness to failures, were the subject of previous publications [47]. Other applications of DOC have been also studied in the context of autonomic resource provisioning [45, 58] and autonomic policy adaptation [46] Here, we explain the main characteristics of the algorithm, and refer the reader to the cited publications for further details.

In DOC, each of the events to be clustered is represented as a point in a multidimensional space, each dimension in this space, referred to as an information space, corresponds to one of the event attributes, and the location of a point within the space is determined by the values for each of its attributes. It is assumed that the range of values of each attribute is an ordered set. For each set, a distance function can be defined in order to measure the similarity between points (i.e., similarity is inversely proportional to distance). This definition is straightforward for quantitative attributes, and can be applied to non-quantitative attributes as well with an appropriate encoding. The notion of similarity based on distance in each dimension extends to the multidimensional information space, for which a

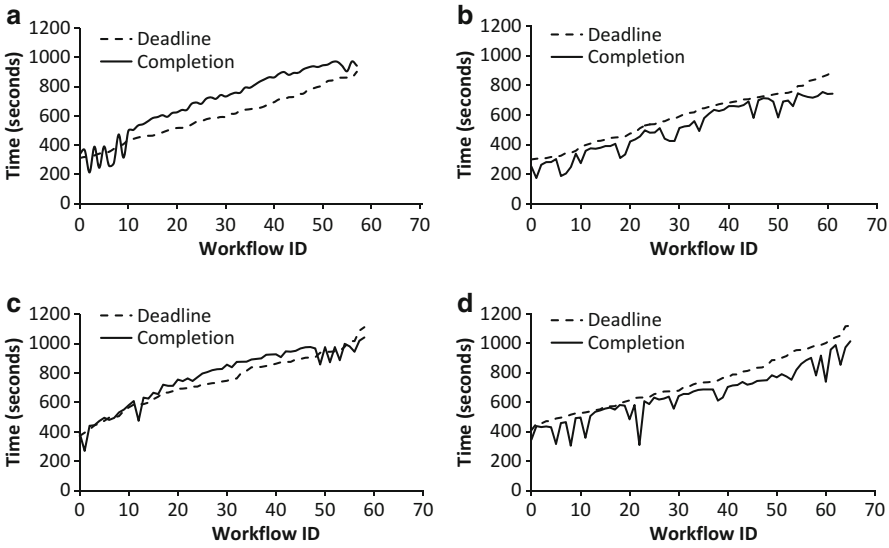

Fig. 8 Deadline and completion time of each workflow with and without cloudburst. (**a**) Only Rutgers (No Cloudburst), deadline 300 s. (**b**) Rutgers+EC2 (CloudBurst), deadline 300 s. (**c**) Only Rutgers (No Cloudburst), deadline 420 s. (**d**) Rutgers+EC2 (CloudBurst), deadline 420 s

distance function can also be defined in terms of the uni-dimensional distances. Conceptually, a cluster is a set of points for which mutual distances are relatively smaller than the distances to other points in the space [26]. However, the approach for cluster detection described in this chapter is not based primarily on evaluating distances between points, but rather on evaluating the relative density of points within the information space. In this case, point similarity is directly proportional to point density.

The approach used for the evaluation of point density, and thus for the detection of clusters and outliers, is dividing the information space into regions and to observe the number of points within each region. If the total number of points in the information space is known, then a baseline density for a uniform distribution of points can be calculated and used to estimate an expected number of points per region. Clusters are recognized within a region if the region has a relatively larger point count than this expected value. Conversely, if the point count is smaller than expected, then these points are potential outliers. However, clusters may cross region boundaries and this must be taken into account when verifying potential outliers. The approach described above lends itself to a decentralized implementation because each region can be assigned to a particular processing node. Nodes can then analyze the points within their region and communicate with nodes responsible for adjoining regions in order to deal with boundary conditions.

As part of this application scenario, we evaluated the autonomic manager by showing how to achieve user objectives such as time constraint and deadline using

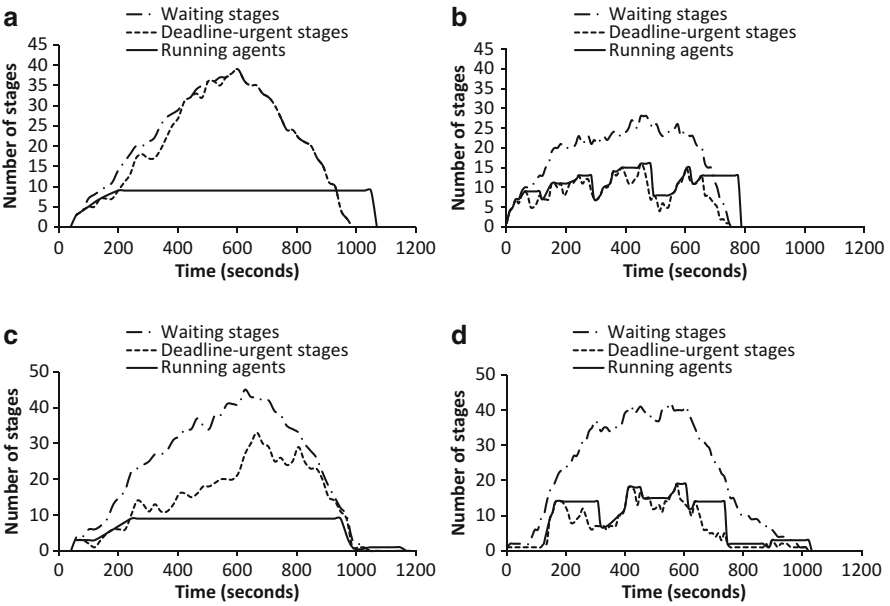

Fig. 9 Number of stages waiting to be executed and number of allocated agents. The required agents are provisioned on-demand by cloudburst regardless of the local resources limitations. (**a**) Only Rutgers (No Cloudburst), deadline 300 s. (**b**) Rutgers+EC2 (CloudBurst), deadline 300 s. (**c**) Only Rutgers (No Cloudburst), deadline 420 s. (**d**) Rutgers+EC2 (CloudBurst), deadline 420 s

cloudbursts to a public cloud when the local resources are limited. We have used Rutgers cluster as a local resource class with 27 nodes where each node has 8 cores, 6 GB memory, 146 GB storage and 1 GB Ethernet connection. For a public cloud, we used Amazon EC2, c1.medium instance type. We are going to use workflows of the DOC application with three different stages each, with different parameters and input files. Therefore, each stage of the workflow has a different execution time. From the point of view of the autonomic manager, each stage is a task and is executed by a single agent. We decided that each agent uses two workers to execute a stage. Hence, the maximum number of agents that can be allocated in the Rutgers cluster is nine because each agent involves three machines (one for the agent and two for the workers). Moreover, each agent can only execute one stage at a time, which means that if there are multiple workflows submitted in a short time, then their stages should wait in the space for some time until they are selected. Therefore, the autonomic manager has to autonomically scale-up/down agents to adapt the provisioned resources to the workload.

User objectives can be set for each stage of the workflow separately as each stage can run a different application with different constraints and the length of computation or the amount of required resources can vary among stages even for the same application. In this experiment, we set a deadline for each stage and we have used shortest deadline first serve (SDFS) policy for task selection. Hence,

agents sequentially select stages which have the shortest remaining time to deadline. The number of agents is related to the number of stages which should immediately start to meet the deadline constraint. Thus, the autonomic scheduler starts agents to execute urgent stages. The scheduler tries to allocate resources from the cluster at Rutgers and only if the local cluster does not have enough resources it allocates resources from Amazon EC2 (cloud burst).

The average interval of workflow submission has been set to 10 s during the first 600 s of execution. We show two set of experiments, fixing the deadline for each stage of the workflow to 100 and 140 s, respectively. Since each workflow has a three stages, the deadline for each workflow is 300 and 420 s, respectively. Results are shown in Figs. 8 and 9. Specifically, Fig. 8 shows the deadline and completion time of each workflow for 300 and 420 s. Note that the deadline and completion times of each workflow are relative to the time it was submitted. We can observe that when we executed the workflows using Rutgers resources only (without cloudburst), around a 90 % of the workflow violated deadline constraints, even for a large deadline. However, when we enabled cloudburst to EC2, all workflow were able to meet the deadline constraints by allocating as many EC2 instances as required on-demand.

On the other hand, Fig. 9 shows the number of waiting stages and the number of allocated agents over time. It also shows the deadline-urgent stages, which are those waiting stages that need to be executed immediately to have a chance to meet their deadlines. We can observe in Fig. 9a, c, that local resources were not able to provide the computational power needed to guarantee the deadline constrains. It caused that the waiting time of each stage and the number of deadline-urgent stages to be increased, and therefore deadlines were eventually violated. However, Fig. 9b, d shows that when we enabled cloudburst, the autonomic manager was able to dynamically scale up and down the number of allocated agents to satisfy the demand of deadline-urgent stages. Scaling up the number of allocated agents was immediately done when needed. However, we delayed the deallocation of agents to avoid too much fluctuation. Therefore, by using the autonomic manager, all the workflow stages were able to meet the deadline constraints.

5 Lessons Learned

The different use cases presented in previous sections clearly demonstrate feasibility and capability of an elastic, dynamically federated infrastructure. These use cases have shown how it is possible to use the autonomic capabilities of our framework for different objectives, including acceleration and conservation.

Oftentimes, a single resource is not sufficient to execute a given scientific workload (e.g. because it is of limited scale, or it mismatches application requirements). Although the majority of researchers with large computational demands have access to multiple infrastructures, such as HPC, computational grids, and clouds, taking advantage of the collective power of these systems is not trivial. Our results

show how a federated framework can help to aggregate the computational power (i.e. capacity) of geographically distributed resources and offer them in an elastic way to the users. In the CBIR use case (Sect. 4.1), we shown how the execution can be dramatically sped up, from weeks to minutes. One important element that contributed to the success of this experiment, was the ability of the federation to scale across institutional and geographic boundaries.

As discussed above, it is clear from the state of the art that cloud platforms can effectively support certain classes for CDS&E applications, such as for example, high throughput computing (HTC) applications. However, many other existing CDS&E application formulations are not directly suited for cloud platforms. As a result, it is essential to explore alternate formulations of these applications that could potentially benefit from cloud services. This idea has been demonstrated in the replica exchange use case (Sect. 4.1) where an asynchronous implementation allowed us to take advantage of the capabilities offered by different resources. Therefore, having highly heterogeneous resources as a part of the federation, it is crucial to take advantage of their particular characteristics and optimize resources allocation. This is synergistic with the concept of autonomic computing. In particular, in the replica exchange case we used clouds to complement HPC resources, which allowed us to save HPC allocations.

Finally, one important aspect of clouds is the ability of adapting the resources to the demands of applications and users. In the majority of cases predicting computational and storage requirements is extremely difficult. Therefore, scaling up/down or out as needed becomes essential for dynamic workloads. Our results show that elasticity allows to adapt the number of provisioned resources to the demands. In this way, it is possible to meet the deadline for different applications while utilizing just the appropriated number of resources. This concept is shown in the business data analytics application, Sect. 4.2. Additionally, the elasticity can also be used to make the infrastructure resilient to changes in the federation. Consequently, the federation is able to better sustain computational throughput.

Conclusions

We have presented a federation model that enables the orchestration of hybrid distributed infrastructures and the coordinated execution of application workflows on such federated environments. We experimentally investigated, from an application's perspective, possible usage modes for integrating HPC and clouds as well as how autonomic computing can support these modes. In particular, we used three use case scenarios to highlight different aspects of the federation. First, we showed how medical image research applications can benefit from the federation of distributed resources and their aggregated computational power. Then, we exploited the principles of adaptivity and elasticity at the application level, through asynchronous replica exchange,

(continued)

and at the infrastructure level, through CometCloud, in the context of a molecular dynamics application. We specifically argue how clouds can be beneficial to quickly explore the application domain space saving the HPC allocations to compute only those replicas that were identified as relevant during the exploration. Finally, we performed a deadline objective-driven workflow execution to further study the behavior of the autonomic manager. The workflow was based on a decentralized online clustering application and the results showed autonomic manager is able to achieve deadline constraint by provisioning resources on demand (cloudburst).

Our ongoing work includes the exploration of new scientific application scenarios that require the coordinated used of distributed hybrid infrastructures and supporting new types of application formulations such as adaptive workflows where dynamic provisioning and federation is essential to respond to non-deterministic behaviors. Moreover, we are also working in enabling new cloud-like paradigms to provide a platform where scientist only need to change the application driver to benefit from an existing federation infrastructure. Finally, we would also like to evaluate new ways to manage the different sites of the federation. Currently, it is based on a pull mode and we believe that other mechanisms such as publish/subscribe would bring us many more interesting use cases.

Acknowledgements The research presented in this work is supported in part by US National Science Foundation (NSF) via grants numbers OCI 1310283, OCI 1339036, DMS 1228203 and IIP 0758566, by the Director, Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy through the Scientific Discovery through Advanced Computing (SciDAC) Institute of Scalable Data Management, Analysis and Visualization (SDAV) under award number DE-SC0007455, the Advanced Scientific Computing Research and Fusion Energy Sciences Partnership for Edge Physics Simulations (EPSI) under award number DE-FG02-06ER54857, the ExaCT Combustion Co-Design Center via subcontract number 4000110839 from UT Battelle, and by an IBM Faculty Award. We used resources provided by: XSEDE NSF OCI-1053575, FutureGrid NSF OCI-0910812, and NERSC Center DOE DE-AC02-05CH11231. The research and was conducted as part of the NSF Cloud and Autonomic Computing (CAC) Center at Rutgers University and the Rutgers Discovery Informatics Institute (RDI2). We would also like to acknowledge Hyunjoo Kim, Moustafa AbdelBaky, and Aditya Devarakonda for their contributions to the CometCloud project.

References

1. A. Agarwal, M. Ahmed, A. Berman, B. L. Caron, et al. GridX1: A Canadian computational grid. *Future Gener. Comput. Syst.*, 23:680–687, June 2007.
2. A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Web Services Agreement Specification (WS-Agreement), GFD-R-P.107. Technical report, GRAAP WG, Open Grid Forum, March 2007.

3. M. D. Assuncao and R. Buyya. Performance analysis of allocation policies for interGrid resource provisioning. *Information and Software Technology*, 51:42–55, January 2009.
4. L. F. Bittencourt, C. R. Senna, and E. R. M. Madeira. Enabling execution of service workflows in grid/cloud hybrid systems. In *Network Operations and Management Symp. Workshop*, pages 343–349, 2010.
5. N. Bobroff, L. Fong, S. Kalayci, Y. Liu, J. C. Martinez, I. Rodero, S. M. Sadjadi, and D. Villegas. Enabling interoperability among meta-schedulers. In *IEEE CCGrid*, pages 306–315, 2008.
6. R. Bolze, F. Cappello, E. Caron, M. Dayde, et al. Grid'5000: a large scale and highly reconfigurable experimental Grid testbed. *International Journal of High Performance Computing Applications*, 20:481–494, November 2006.
7. N. Carriero and D. Gelernter. Linda in context. *Commun. ACM*, 32(4):444–458, 1989.
8. A. Celesti, F. Tusa, M. Villari, and A. Puliafito. How to enhance cloud architectures to enable cross-federation. In *IEEE CLOUD*, pages 337–34, 2010.
9. M. D. de Assuncao, R. Buyya, and S. Venugopal. Intergrid: a case for internetworking islands of grids. *Concurrency Computat. Pract. and Exper.*, 20(8):997–1024, 2008.
10. M. D. de Assuncao, A. di Costanzo, and R. Buyya. Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters. In *ACM HPDC*, pages 141–150, 2009.
11. E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good. The cost of doing science on the cloud: the montage example. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, SC '08, pages 50:1–50:12, Piscataway, NJ, USA, 2008. IEEE Press.
12. T. Dunning and R. Nandkumar. International cyberinfrastructure: activities around the globe. *Cyberinfrastructure Technology Watch Quarterly*, 2:2–4, February 2006.
13. E. Elmroth and J. Tordsson. A standards-based grid resource brokering service supporting advance reservations, coallocation, and cross-grid interoperability. *Concurr. Comput. : Pract. Exper.*, 21(18):2298–2335, Dec. 2009.
14. D. Erwin and D. Snelling. UNICORE: A Grid Computing Environment. In *International Euro-Par Conference on Parallel Processing*, pages 825–834, Manchester, UK, August 2001.
15. L. Field, E. Laure, and M. W. Schulz. Grid deployment experiences: Grid interoperation. *J. Grid Comput.*, 7(3):287–296, 2009.
16. I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan-Kauffman, 1999.
17. I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Perfomance Computing Applications*, 15(3):200–222, 2001.
18. G. Fox and D. Gannon. Cloud Programming Paradigms for Technical Computing Applications. Technical report, Indiana University, 2012.
19. T. Goodale, S. Jha, T. Kielmann, A. Merzky, J. Shalf, and C. Smith. A Simple API for Grid Applications (SAGA), GWD-R.72. Technical report, SAGA-CORE Working Group, Open Grid Forum, September 2006.
20. I. Gorton, Y. Liu, and J. Yin. Exploring architecture options for a federated, cloud-based system biology knowledgebase. In *IEEE Intl. Conf. on Cloud Computing Technology and Science*, pages 218–225, 2010.
21. T. Hey and A. Trefethen. The UK e-Science Core Programme and the Grid. *Future Gener. Comput. Syst.*, 18:1017–1031, 2002.
22. E. Huedo, R. Montero, and I. Llorente. A recursive architecture for hierarchical grid resource management. *Future Gener. Comput. Syst.*, 25:401–405, April 2009.
23. K. Hukushima and K. Nemoto. Exchange Monte Carlo method and application to spin glass simulations. *J. Phys. Soc. Jpn.*, 65:1604–1608, 1996.
24. A. Iosup, D. Epema, T. Tannenbaum, M. Farrelle, and M. Livny. Inter-Operable Grids through Delegated MatchMaking. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC07)*, pages 13:1–13:12, Reno, Nevada, November 2007.

25. A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema. Performance analysis of cloud computing services for many-tasks scientific computing. *IEEE Trans. Parallel Distrib. Syst.*, 22(6):931–945, 2011.
26. A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1998.
27. K. Keahey and T. Freeman. Science clouds: Early experiences in cloud computing for scientific applications. In *Cloud Computing and Its Applications (CCA-08)*, October 2008.
28. A. Kertesz and P. Kacsuk. Grid Meta-Broker Architecture: Towards an Interoperable Grid Resource Brokering Service. In *CoreGRID Workshop on Grid Middleware in conjunction with Euro-Par, LNCS 4375*, pages 112–116, Desden, Germany, 2008.
29. A. Kertész and P. Kacsuk. Gmbs: A new middleware service for making grids interoperable. *Future Gener. Comput. Syst.*, 26(4):542–553, Apr. 2010.
30. A. Kertesz, I. Rodero, and F. Guim. Bpdl: A data model for grid resource broker capabilities. Technical Report TR-0074, Institute on Resource Management and Scheduling, CoreGRID - Network of Excellence, March 2007.
31. A. Kertesz, I. Rodero, and F. Guim. Meta-Brokering Solutions for Expanding Grid Middleware Limitations. In *Workshop on Secure, Trusted, Manageable and Controllable Grid Services (SGS) in conjunction with International Euro-Par Conference on Parallel Processing*, Gran Canaria, Spain, July 2008.
32. A. Kertžsz, I. Rodero, F. Guim, A. Kertžsz, I. Rodero, and F. Guim. A data model for grid resource broker capabilities. In *Grid Middleware and Services*, pages 39–52, 2008.
33. H. Kim, Y. E. Khamra, I. Rodero, S. Jha, and M. Parashar. Autonomic management of application workflows on hybrid computing infrastructure. *Sci. Program.*, 19(2–3):75–89, 2011.
34. H. Kim, M. Parashar, D. J. Foran, and L. Yang. Investigating the use of autonomic cloudbursts for high-throughput medical image registration. In *IEEE/ACM GRID*, pages 34–41, 2009.
35. K. Leal, E. Huedo, and I. M. Llorente. A decentralized model for scheduling independent tasks in federated grids. *Future Gener. Comput. Syst.*, 25(8):840–852, 2009.
36. Z. Li and M. Parashar. A computational infrastructure for grid-based asynchronous parallel applications. In *HPDC*, pages 229–230, 2007.
37. Z. Li and M. Parashar. Grid-based asynchronous replica exchange. In *IEEE/ACM GRID*, pages 201–208, 2007.
38. M. Marzolla, P. Andreetto, V. Venturi, A. Ferraro, et al. Open Standards-Based Interoperability of Job Submission and Management Interfaces across the Grid Middleware Platforms gLite and UNICORE. In *IEEE International Conference on e-Science and Grid Computing*, pages 592–601, Bangalore, India, December 2007.
39. H. Mohamed and D. Epema. KOALA: a Co-allocating Grid Scheduler. *Concurrency and Computation: Practice & Experience*, 20:1851–1876, November 2008.
40. D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The eucalyptus open-source cloud-computing system. In *IEEE/ACM CCGRID*, pages 124–131, 2009.
41. A. Oleksiak, A. Tullo, P. Graham, T. Kuczynski, J. Nabrzyski, D. Szejnfeld, and T. Sloan. HPC-Europa: Towards Uniform Access to European HPC Infrastructures. In *IEEE/ACM International Workshop on Grid Computing*, pages 308–311, November 2005.
42. S. Ostermann, R. Prodan, and T. Fahringer. Extending grids with cloud resource management for scientific computing. In *IEEE/ACM Grid*, pages 42–49, 2009.
43. M. Parashar, M. AbdelBaky, I. Rodero, and A. Devarakonda. Cloud Paradigm and Practices for CDS&E. Technical report, Cloud and Autonomic Computing Center, Rutgers Univ., 2012.
44. J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. V. Kal, and K. Schulten. Scalable molecular dynamics with NAMD. *J. of Computational Chem.*, pages 1781–1802, 2005.
45. A. Quiroz, H. Kim, M. Parashar, N. Gnanasambandam, and N. Sharma. Towards autonomic workload provisioning for enterprise grids and clouds. In *IEEE/ACM GRID*, 2009.
46. A. Quiroz, M. Parashar, N. Gnanasambandam, and N. Sharma. Autonomic policy adaptation using decentralized online clustering. In *ICAC*, pages 151–160, 2010.

47. A. Quiroz, M. Parashar, N. Gnanasambandam, and N. Sharma. Design and evaluation of decentralized online clustering. *TAAS*, 7(3):34, 2012.
48. I. Raicu, I. Foster, and Y. Zhao. Many-task computing for grids and supercomputers. In *Proc. Workshop on Many-Task Computing on Grids and Supercomputers*, pages 1–11, 2008.
49. I. Raicu, Z. Zhang, M. Wilde, I. Foster, P. Beckman, K. Iskra, and B. Clifford. Towards loosely. coupled programming on petascale systems. In *IEEE/ACM Supercomputing*, 2008.
50. N. Ram and S. Ramakrishran. International cyberinfrastructure: activities around the globe. *Cyberinfrastructure Technology Watch Quarterly*, 2:15–19, February 2006.
51. M. Riedel, A. Memon, M. Memon, D. Mallmann, et al. Improving e-Science with Interoperability of the e-Infrastructures EGEE and DEISA. In *International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 225–231, Opatija, Croatia, May 2008.
52. P. Riteau, M. Tsugawa, A. Matsunaga, J. Fortes, and K. Keahey. Large-scale cloud computing research: Sky computing on futuregrid and grid'5000. In *ERCIM News*, 2010.
53. B. Rochwerger, D. Breitgand, E. Levy, A. Galis, et al. The reservoir model and architecture for open federated cloud computing. *IBM Journal of Research and Development*, 53, 2009.
54. I. Rodero, F. Guim, J. Corbalan, L. Fong, Y. Liu, and S. Sadjadi. Looking for an Evolution of Grid Scheduling: Meta-brokering. *Grid Middleware and Services: Challenges and Solutions*, pages 105–119, August 2008.
55. I. Rodero, F. Guim, J. Corbalan, L. Fong, and S. Sadjadi. Broker Selection Strategies in Interoperable Grid Systems. *Future Gener. Comput. Syst.*, 26(1):72–86, January 2010.
56. I. Rodero, F. Guim, J. Corbalan, and A. Goyeneche. The grid backfilling: a multi-site scheduling architecture with data mining prediction techniques. In *Grid Middleware and Services*, pages 137–152, 2008.
57. I. Rodero, F. Guim, J. Corbalan, and J. Labarta. How the JSDL can Exploit the Parallelism? In *IEEE International Symposium on Cluster Computing and the Grid (CCGrid)*, pages 275–282, Singapore, May 2006.
58. I. Rodero, J. Jaramillo, A. Quiroz, M. Parashar, F. Guim, and S. Poole. Energy-efficient application-aware online provisioning for virtualized clouds and data centers. In *Green Computing Conf.*, pages 31–45, 2010.
59. I. Rodero, D. Villegas, N. Bobroff, Y. Liu, L. Fong, and S. M. Sadjadi. Enabling interoperability among grid meta-schedulers. *J. Grid Comput.*, 11(2):311–336, 2013.
60. C. Schmidt and M. Parashar. Squid: Enabling search in dht-based systems. *J. Parallel Distrib. Comput.*, 68(7):962–975, 2008.
61. J. Seidel, O. Waldrich, W. Ziegler, P. Wieder, and R. Yahyapour. Using SLA for Resource Management and Scheduling - a Survey, TR-0096. Technical report, Institute on Resource Management and Scheduling, 2007.
62. B. Sotomayor, R. Montero, I. Llorente, and I. Foster. Virtual infrastructure management in private and hybrid clouds. *IEEE Internet Computing*, 13:14–22, 2009.
63. I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for internet applications. In *ACM SIGCOMM*, pages 149–160, 2001.
64. R. Swendsen and J. Wang. Replica Monte Carlo simulation of spin-glasses. *Physical Review Letters*, 57:2607–2609, 1986.
65. P. Troger, H. Rajic, A. Haas, and P. Domagalski. Standardization of an API for Distributed Resource Management Systems. In *Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid*, pages 619–626, Washington, DC, USA, 2007.
66. C. Vazquez, E. Huedo, R. Montero, and I. Llorente. Dynamic provision of computing resources from grid infrastructures and cloud providers. In *Grid and Pervasive Computing Conf.*, pages 113–120, 2009.
67. T. Vazquez, E. Huedo, R. Montero, and I. Lorente. Evaluation of a Utility Computing Model Based on the Federation of Grid Infrastructures. In *International Euro-Par Conference on Parallel Processing*, pages 372–381, Rennes, France, August 2007.

68. C. Vecchiola, S. Pandey, and R. Buyya. High-performance cloud computing: A view of scientific applications. In *Proceedings of the 2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks*, ISPAN '09, pages 4–16, Washington, DC, USA, 2009. IEEE Computer Society.
69. D. Villegas, N. Bobroff, I. Rodero, J. Delgado, et al. Cloud federation in a layered service model. *J. Comput. Syst. Sci.*, 78(5):1330–1344, 2012.
70. J.-S. Vockler, G. Juve, and M. R. Ewa Deelman and. Experiences using cloud computing for a scientific workflow application. In *2nd Workshop on Scientific Cloud Computing in conjunction with ACM HPDC*, pages 402–412, 2011.
71. Amazon EC2. http://aws.amazon.com/ec2/.
72. CometCloud Project. http://www.cometcloud.org.
73. DAS-3 Project. http://www.cs.vu.nl/das.
74. DEISA Project. http://www.deisa.eu.
75. D-Grid Project. http://www.d-grid.de.
76. EGI Europe. http://www.egi.eu.
77. Eucalyptus. http://open.eucalyptus.com/.
78. Grid' 5000 Project. https://www.grid5000.fr.
79. R. Zhang, M. Parashar, and E. Gallichio. Salsa: Scalable asynchronous replica exchange for parallel molecular dynamics applications. In *ICPP*, pages 127–134, 2006.
80. IBM Smart Cloud. http://www.ibm.com/cloud-computing/us/en/.
81. IEEE Intercloud WG (ICWG) Working Group. http://standards.ieee.org/develop/wg/ICWG-2302_WG.html.
82. IEEE Standard for Intercloud Interoperability and Federation. http://standards.ieee.org/develop/project/2302.html.
83. Naregi Project, http://www.naregi.org.
84. Nimbus Project. http://www.nimbusproject.org.
85. Open Cloud Computing Interface (OCCI). http://occi-wg.org/.
86. OpenNebula. http://www.opennebula.org/.
87. OpenStack. http://openstack.org/.
88. Open Science Grid. https://www.opensciencegrid.org/.
89. PRACE Project. http://www.prace-ri.eu.
90. Siena Initiative. http://www.sienainitiative.eu.
91. XSEDE Project. https://www.xsede.org/.