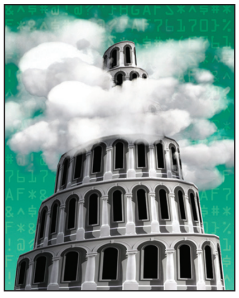
Editor: George Pallis • *gpallis@cs.ucy.ac.cy*

CometCloud

Enabling Software-Defined Federations for End-to-End Application Workflows



Javier Diaz-Montes, Moustafa AbdelBaky, Mengsong Zou, and Manish Parashar • *Rutgers University*

Emerging applications, from big science to the Internet of Things, increasingly involve dynamic and data-driven end-to-end workflows with large and often heterogeneous requirements. CometCloud aims to provide infrastructure and programming support for enabling such workflows via flexible, software-defined synthesis of custom cyberinfrastructure through the autonomic, on-demand federation of geographically distributed compute and data resources.

Emerging applications increasingly involve dynamic and data-driven end-to-end workflows. For example, end-to-end science and engineering applications integrate data sources (such as monitoring, observations, and experiments) with computational modeling, analytics, and visualization. These workflows typically have large and often heterogeneous requirements, and necessitate platforms that dynamically combine resources across systems and data centers — for example, to aggregate capacity and capabilities. Furthermore, emerging pervasive computational ecosystems, powered by the Internet of Things (IoT), and the resulting proliferation of data sources could enable a new class of application workflows that can fundamentally transform our ability to manage and optimize our lives and environment. However, these applications will once again require that we seamlessly and opportunistically combine pervasive digital data sources and computational power.

The CometCloud project at the Rutgers Discovery Informatics Institute (RDI2) aims to provide infrastructure and programming support for such end-to-end application workflows. CometCloud enables flexible, software-defined synthesis of custom cyberinfrastructure through the autonomic, on-demand federation of geographically distributed compute and data resources. The CometCloud team has been working closely with scientists and engineers from different domains to understand application workflows and their requirements, and to explore appropriate usage modes and abstractions. This has led CometCloud to expose a software-defined federated cyberinfrastructure using cloud abstractions to support various programming paradigms and application requirements.

Here, we present CometCloud's architecture and design, and employ sample use cases to illustrate how CometCloud supports dynamic application workflows.

CometCloud Overview

CometCloud is an autonomic framework designed to enable highly heterogeneous, dynamically federated computing and data platforms that can support end-to-end application workflows with diverse and changing requirements.[1] This occurs through autonomic, on-demand federation of geographically distributed compute and data resources as applications need them, and by exposing the federation using elastic cloud abstractions and science-as-a-service platforms. Consequently, CometCloud can create a nimble and programmable environment that autonomously evolves over time, adapting to changes in infrastructure and application requirements. Figure 1a shows an overview of the CometCloud architecture, including its three key layers: infrastructure/federation, autonomic management, and programming/interface.

The infrastructure/federation layer manages dynamic resource federation and provides

 1089-7801/15/$31.00 © 2015 IEEE Published by the IEEE Computer Society

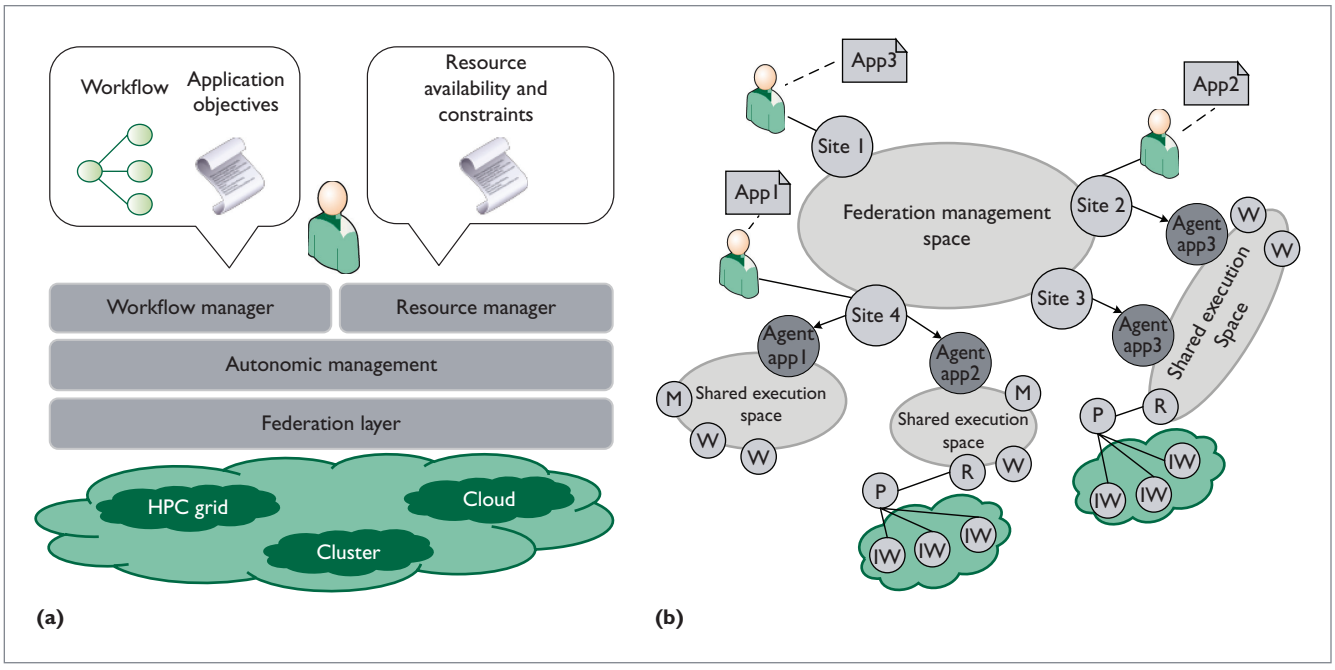


Figure 1. CometCloud. (a) The CometCloud architecture comprises three key layers: infrastructure/federation, autonomic management, and programming/interface. (b) The CometCloud coordination model is used to orchestrate different aspects of the federation.

essential services. Several key components make up this layer. The first is an information lookup system built on a content-based distributed hash-table (DHT) based on a structured peer-to-peer overlay. This system is used for information discovery (for instance, to locate resources using their attributes). It maintains content locality and guarantees that content-based information queries, specified using keywords and wildcards, are satisfied with bound costs (in terms of the number of hops required to find the data).

The second is a scalable, decentralized, shared coordination space, built on top of the DHT,[2] called CometSpace, which all resources in the federation can access associatively. CometSpace provides tuple-space-like abstraction for coordination and messaging, and enables coordination in the federation model (Figure 1b). Specifically, we define two types of coordination spaces. First, a single-management space spans across all resource sites, creating and orchestrating the federation. Second, multiple shared execution spaces are created on-demand during application workflow executions to satisfy computational or data needs. Execution spaces can be created within a single resource site, or can burst to others, such as public clouds or external high-performance computing (HPC) systems.

CometCloud federation is created dynamically and collaboratively; resources or sites can join or leave at any point, identify themselves (using security mechanisms such as public/private keys), negotiate the federation terms, discover available resources, and advertise their own resources and capabilities.[3]

Next, the autonomic management layer lets users and applications define objectives and policies that drive resource provisioning and application workflow execution while satisfying user constraints (such as budgets and deadlines) and application requirements (types of resources). The autonomic mechanisms in place not only provision the right resources when needed, but also monitor the execution's progress and adapt it to prevent violations of established agreements.[4]

Finally, the programming/interface layer provides interfaces to independently describe application workflows and resources. Currently, we can describe application workflows, using XML documents, as a set of stages defining input and output data, dependencies to other stages, and scheduling policies, and possibly annotated with specific objectives and policies.

The CometCloud resource interface lets providers and users declaratively specify resource availability as well as policies and constraints to regulate their use. For example, a provider can offer resources only at certain times of the day; users might require certain resource capabilities or capacities, or specific connectivity; users might prefer certain resource types over others (such as HPC versus clouds), or want to use cloud resources only if they're within a desired price range; or users

might want to only use resources within a specific geographic region due to data movement regulations. These constraints collectively define the set of resources that are federated at any instant in time and that the application can use.

CometCloud Use Case Scenarios

We next present use cases to illustrate how CometCloud and its federation model can be used to enable real-world applications.

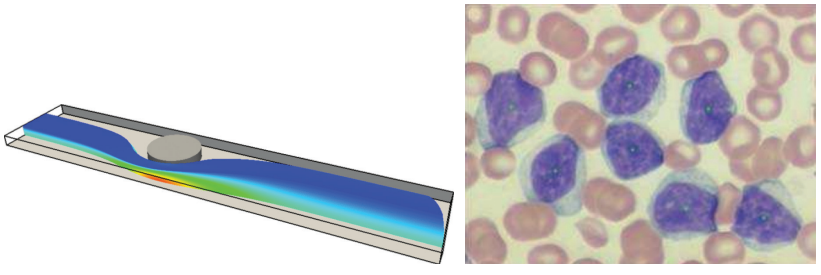


Figure 2. CometCloud use cases for large-scale science and engineering. We can see sample results for (a) the fluid flow in microchannel[3] and (b) cancer detection use cases.

Large-Scale Science and Engineering

The complexity of many science and engineering problems requires computational capacity exceeding what an average user can expect from a single computational center. The analysis of high-dimensional parameter spaces or uncertainty quantification by stochastic sampling are just two examples of a broad class of problems that are becoming increasingly important in a wide range of application domains. Although we can view many of these problems as a set of independent tasks, their collective complexity easily requires millions of core hours on any state-of-the-art supercomputer, and throughput that a single multi-user queuing system can't sustain. We've used CometCloud to enable a range of applications in this area,[5] and highlight two specific use cases here.

In the first use case, we explored using aggregated HPC resources to solve a large-scale engineering problem. We showed how CometCloud can help build a computational federation that's elastic, resilient, scalable, and easy to use by end users. We focused on understanding fluid flow in microscale devices. During the experiment, we opportunistically federated resources as they became available and released any that experienced problems or ran out of allocations. We federated 10 resources across six institutions from three countries. This experiment lasted 16 days and executed 12,845 tasks. It consumed 2,897,390 core hours and generated 398 Gbytes of data. Our federation model's extreme flexibility enabled a sustained performance (see Figure 2a).[3]

In the second use case, we explored how to enable data-driven applications on top of national cyberinfrastructure to efficiently support state-of-the-art analytics across distributed image databases. We focused on content-based histopathology image retrieval (CBIR), in which datasets can be geographically distributed. In this experiment, we created a hybrid federation of HPC and cloud resources to perform parallel searching of content-wise similar images in several datasets. We showed not only the feasibility but also a significant reduction in the overall computational time, which ensures our proposed solution's practical utility for near real-time medical diagnosis (see Figure 2b).[6]

Smart Infrastructure

The rise of Internet-connected instrumentation and the extraordinary growth of digital data sources have led to an unprecedented amount of data that usually can't be predetermined. Due to data and resource's heterogeneous and distributed nature, we must rethink how we store, process, and analyze them to provide insight in a timely manner. Examples include the wide variety of sensor-network-based applications, in which sensors interface with real-world artifacts and must respond to unpredictable physical phenomena. In this context, we discuss two specific CometCloud examples.

The first use case targets data analysis from electricity meters to support smart (power) grids, such as electric vehicle charging stations. This scenario generates multiple datastreams in which raw data is continuously transmitted at variable and unpredictable rates. We combined CometCloud with a reference net (a particular type of Petri net) based interpreter to support simultaneous datastream processing and enable on-demand scaling up or down of heterogeneous computational resources to ensure promised quality of service (throughput) for each datastream (see Figure 3a). CometCloud lets us easily extend this model to, for example, exploit geolocation and provisioning resources at the closest datacenter.[7]

The second use case is in the context of smart buildings, where sensors and actuators control the building's energy consumption. Energy optimization requires the real-time use of sensor data, with several parameters needing optimization based on

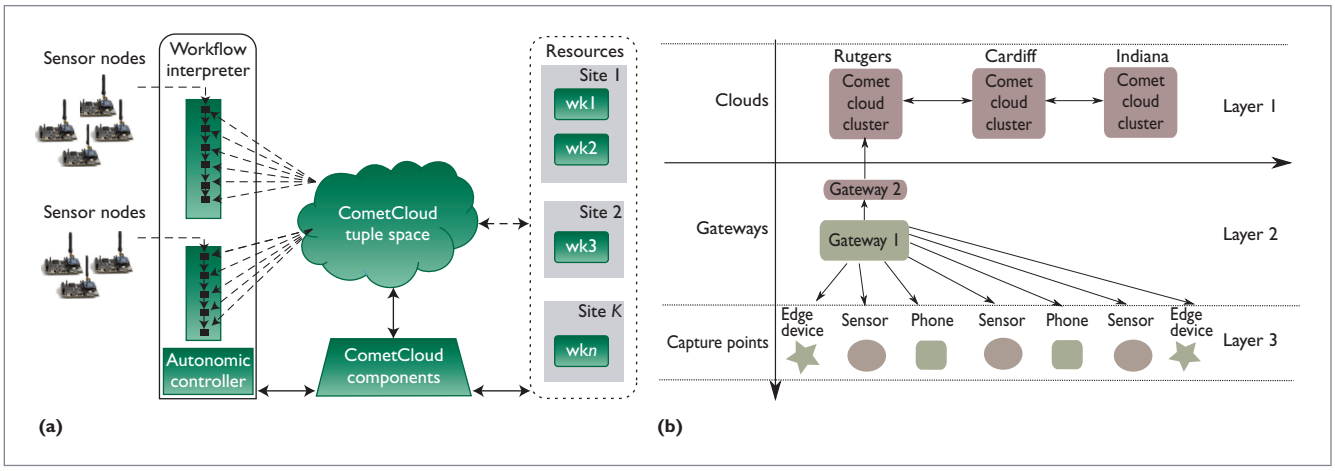


Figure 3. CometCloud use cases in the smart infrastructure domain. We can see the architectures for the (a) streaming data[7] and (b) building analytics[8] use cases.

a particular building representation. Because sensors can provide readings in 15- to 30-minute intervals, any simulation or optimization must be carried out with a similar interval. We've developed a multilayer cloud infrastructure that exploits the available computation at the cloud's edge by distributing processing over sensing nodes, multiple intermediate or gateways nodes, and cloud resources. Using this infrastructure, we explored questions such as where processing should be carried out, what processing should be undertaken centrally versus at an edge node, and how processing can be distributed across multiple datacenter locations to achieve QoS and cost targets.[8]

Analogous to how clouds have revolutionized the way we acquire and use IT resources, we believe that software-defined environments can fundamentally affect how resources (compute, data, and networking) are federated and customized. These environments will be especially important in an era where digital data sources proliferate (including geographically distributed sensors, mobile devices, and instrumented infrastructure) and nontrivial computational power is ubiquitous. Dynamic, data-driven applications can potentially transform our ability to understand and manage our lives and our environment — we can envision data-driven and information-rich pervasive computational ecosystems that seamlessly and opportunistically federate these data and computing power to model, manage, control, adapt, and optimize virtually any realizable subsystem of interest.

We're currently exploring usage modes where data is processed and insights gleaned close to the source rather than necessarily transporting it to remote data processing resources. Specifically, we're looking at application formulations where data is processed in situ (at the edge devices) and in transit (along the data path), transforming real-time data into knowledge that can drive critical decisions.

Acknowledgments

This work is supported in part by the US National Science Foundation under OCI-1339036, OCI-1310283, OCI-1441376, and IIP-0758566, and by IBM via Open Collaboration Research (OCR) and Faculty awards.

References

1. H. Kim and M. Parashar, "CometCloud: An Autonomic Cloud Engine," *Cloud Computing: Principles and Paradigms*, Wiley, 2011, pp. 275–297.
2. Z. Li and M. Parashar, "Comet: A Scalable Coordination Space for Decentralized Distributed Environments," *Proc. Int'l Workshop on Hot Topics in Peer-to-Peer Systems*, 2005, pp. 104–111.
3. J. Diaz-Montes et al., "Federated Computing for the Masses — Aggregating Resources to Tackle Large-Scale Engineering Problems," *Computing in Science & Eng.*, vol. 16, no. 4, 2014, pp. 62–72.
4. J. Diaz-Montes et al., "Data-Driven Workflows in Multi-Cloud Marketplaces," *Proc. IEEE Int'l Conf. Cloud Computing*, 2014, pp. 168–175.
5. M. Parashar et al., "Cloud Paradigms and Practices for Computational and Data-Enabled Science and Engineering," *Computing in Science & Eng.*, vol. 15, no. 4, 2013, pp. 10–18.
6. X. Qi et al., "Content-Based Histopathology Image Retrieval using CometCloud," *BMC Bioinformatics*, vol. 15, no. 287, 2014, pp. 1–17.
7. R. Tolosana-Calasanz et al., "Extending CometCloud to Process Dynamic Data Streams on Heterogeneous Infrastructures," *Proc. 2014 Int'l Conf. Cloud and Autonomic Computing* (CAC 2014), 2014, pp. 196–205.
8. I. Petri et al., "In-Transit Data Analysis and Distribution in a Multi-Cloud Environment using CometCloud," *Proc. Int'l Workshop Energy Management for Sustainable Internet of Things and Cloud Computing*, 2014, pp. 1–6.

Javier Diaz-Montes is an assistant research professor at Rutgers University and a member of the Rutgers Discovery Informatics Institute (RDI2). His research interests are in parallel and distributed computing and include autonomic computing, cloud computing, virtualization, and scheduling. Diaz-Montes received a PhD in computer science from the Universidad de Castilla-La Mancha, Spain. Contact him at javidiaz@rdi2.rutgers.edu.

Moustafa AbdelBaky is a PhD candidate in the Electrical and Computer Engineering Department at Rutgers University, a member of RDI2 and the NSF Cloud and Autonomic Computing Center at Rutgers, and a research intern at IBM T.J. Watson Research Center. AbdelBaky received the IBM PhD fellowship three years in a row, and is a member of IEEE and SIAM. Contact him at moustafa.a@rutgers.edu.

Mengsong Zou is a PhD student in the Computer Science Department at Rutgers University, and a member of RDI2. His research interests lie in parallel and distributed computing, cloud computing, and scientific workflow management. Zou received an MS in computer science from Huazhong University of Science and Technology, China. Contact him at mengsongzou@gmail.com.

Manish Parashar is a professor in the Department of Computer Science at Rutgers University; the founding director of RDI2), the NSF Cloud and Autonomic Computing Center at Rutgers, and the Applied Software Systems Laboratory, and is associate director of the Rutgers Center for Information Assurance. Parashar is an IBM Faculty Award, Tewkesbury Fellowship, and Enrico Fermi Scholarship recipient. He's a fellow of AAAS and IEEE/IEEE Computer Society, and a senior member of ACM. Contact him at parashar@rutgers.edu.