A FRAMEWORK FOR ENABLING HIGH-END HIGH PERFORMANCE COMPUTING RESOURCES AS A SERVICE

BY MOUSTAFA ABDELBAKY

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Manish Parashar

and approved by

_____

_____

_____

New Brunswick, New Jersey

May, 2012

ABSTRACT OF THE THESIS

A Framework for Enabling High-End High Performance Computing Resources as a Service

by Moustafa AbdelBaky

Thesis Director: Manish Parashar

In an era of data explosion and analysis, researchers across the globe are trying to convert massive quantities of complex data into useful knowledge using Computational and Data-Enabled Science and Engineering (CDS&E) applications. CDS&E applications are gaining traction as an important dimension of Science, Technology, Engineering, and Mathematics research. These applications require powerful processors with fast interconnects, extreme large scale, and elastic resources. While high-end High Performance Computing (HPC) resources provide the necessary requirements, the complexity of such systems has grown exponentially. Furthermore, due to their high cost, limited availability, and high demand, the queue wait times to run applications on these systems is in the order of months. All of the above challenges prevent their adoption as a mainstream solution.

On the other hand, Cloud computing is emerging as a dominant computing paradigm that offers many advantages. Consequently, early adopters have looked

into using Clouds to solve the HPC model challenges. Initially, CDS&E applications were run on commodity Clouds, but this was found to be appropriate only for certain classes of applications. Other approaches explored complementing HPC resources with Clouds but failed to address all challenges in the HPC environment. Cloud providers also tried to provide HPC as a Cloud using small HPC clusters connected to form a larger Cloud but were hindered by small scale and limited performance. These approaches fall short of providing the high performance necessary for CDS&E applications.

In this document, we propose a new approach to achieve the notion of HPC as a Service. This approach targets existing high-end HPC resources and investigates how a Cloud abstraction can be applied to provide a simple interface and support real-world applications. In particular, the application of Clouds to supercomputers is discussed, tested, and validated on an IBM Blue Gene/P supercomputer. The proposed framework transforms Blue Gene/P into an elastic cloud by bridging multiple systems to create HPC federated Clouds, supporting dynamic provisioning and efficient utilization, and maximizing ease-of-use through an *as a Service* abstraction. In order to effectively illustrate the benefits of such a concept, the proposed framework is demonstrated using a real-world ensemble application.

Acknowledgements

The author would like to acknowledge the following personnel and organizations for their tremendous support.

Rutgers University

- *Dr. Manish Parashar* for his guidance, feedback, help and continuous support throughout the program
- *Dr. Hyunjoo Kim* for her feedback on CometCloud
- *Dr. Ivan Rodero* for his feedback on the taxonomy for High Performance Computing and Clouds
- *Melissa Romanus* for her help proofreading this thesis

IBM T.J. Watson Research Center

- *Dr. Kirk E. Jordan* for his advise and thesis revision through the IBM internship program
- *Dr. Hani Jamjoom* for his help with Deep Cloud, and his feedback on the thesis
- *Vipin Sachdeva* for his feedback on the iPad application
- *Dr. Zon-Yin Shae* for his help with Deep Cloud
- *Dr. James Sexton* for the support of his group through the IBM Ph.D. fellowship program

- *Dr. Fred Mintzer* for his help with the system administration of the Blue Gene Watson system in Yorktown Heights, NY, USA

The University of Texas at Austin

- *Dr. Gergina Pencheva* for her help with the oil reservoir history matching application
- *Dr. Reza Tavakoli* for his help with the ensemble application
- *Dr. Mary F. Wheeler* for letting us use the ensemble application and the oil reservoir matching application

King Abdullah University of Science & Technology

- *Dr. Dinesh K Kaushik, Andrew C Winfer, Richard Orme, and Dr. David Keys* for giving us access to the Blue Gene Shaheen system in Thuwal, Saudi Arabia

Dedication

To my parents, to whom I owe my existence, to my siblings who've always believed in me, to my best friend who's always been there for me, to my lucky charm who's always stood by me, to my manager who's always supported me, and last but not least to my advisor who's always guided and helped me.

Table of Contents

List of Tables

List of Illustrations

Chapter 1
Introduction

1.1 Motivation

Computational and Data-Enabled Science and Engineering (CDS&E) applications are becoming increasingly important in understanding complex processes in multiple domains including Aerospace, Automobile, Business Analytics, Entertainment, Finance, Manufacturing, Oil & Gas, and Pharmaceuticals. As scientists and engineers try to gain a greater knowledge of the complex natural, engineered, and social systems, modeling and large-scale simulations of these applications becomes a critical issue, especially in reducing risks and supporting quantified decision making.

CDS&E applications use parallel programming techniques to take advantage of High Performance Computing (HPC) resources (e.g. supercomputers and large clusters) in order to solve advanced computational and data-intensive problems that cannot be solved otherwise. Due to their growing adoption in various fields, the demand for efficient systems that can run such complex applications and tools that can facilitate their development and deployment have become a necessity.

In the remainder of this document the following terms will be interchanged: high performance computing and HPC. The terms "HPC as a Cloud" and "HPC as a Service" will also be interchanged. The terms Computational and Data-Enabled Science and Engineering (CDS&E) applications and scientific applications will also be interchanged.

1.2 Problem Description

Advances in computing technologies have enabled the creation of large parallel computing systems at a relatively inexpensive cost. These high-end systems are specifically designed to run CDS&E applications with high efficiency. Their massive scale, compute power, and fast interconnects among processors has allowed these applications to run faster. However, with the increasing size of these systems, their complexity has grown as well, requiring relatively low-level user involvement and expert knowledge of such systems in order to achieve the desired performance and efficiency. As a result, at the highest-end, only a few "hero" users are able to effectively use these cutting edge systems.

Furthermore, these high-end systems only support static allocations and do not typically support elastic provisioning and dynamic scalability. Consequently, users have to manually stop the application, change the resource allocation, and resubmit their application to adjust the resources as needed. This required user interaction is a detriment to the wide adoption of such systems, due to their complexity. Finally, while these systems are more available and cheaper than before, they are still quite expensive and thus access to these systems can be an issue.

1.3 Problem Statement

As the importance of CDS&E applications grows, the efficient development and deployment of such applications becomes a major issue. While the requirements of CDS&E applications are well served by high-end supercomputing systems that provide the necessary scales and compute/communication capabilities, accessing and running applications on these systems remains a tedious task, which has impacted their adoption as a mainstream solution for CDS&E applications.

1.4 Research Overview

This research explores how a cloud abstraction can be used effectively to support real-world CDS&E applications. Clouds provide simple *as a Service* access to elastic resources and can address some of the challenges faced by scientists using the current high-end resources. Specifically, this research explores the concept of an elastic HPC as a Cloud, the software infrastructure needed to support such an abstraction, and how the abstraction can be used by CDS&E applications.

In addition, this work discusses the development of a prototype framework that transforms the IBM Blue Gene/P supercomputer into an elastic cloud, supporting dynamic provisioning and efficient utilization while maximizing ease-of-use through the *as a service* abstraction. The use of this framework has been demonstrated on an ensemble oil reservoir modeling application. This application performs history matching using an Ensemble Kalman Filter workflow on an elastic HPC as a Cloud infrastructure. This infrastructure consists of geographically distributed Blue Gene/P systems. The design of the aforementioned framework and application are also described in detail.

1.5 Research Contributions

The contributions of this work are:

- Exposing high-end supercomputing resources /it as a Service
- Enabling elastic and dynamic allocation of high-end supercomputing resources
- Providing ease of use in deploying and running CDS&E applications on complex high-end systems
- Prototyping a framework and showcasing a proof of concept that can be

generalized to provide HPC as a Service in all three layers of Cloud computing (Infrastructure as a Service, Platform as a Service, and Software as a Service)

1.6 Research Impact

HPC as a Service can support CDS&E applications in multiple ways. It can provide a platform for applications when local infrastructure is not available. It can also supplement existing platforms to provide additional capacity or complementary capabilities to meet heterogeneous or dynamic needs. For example, Clouds can serve as accelerators or provide resilience to scientific workflows by moving the execution of the workflow onto alternative or fewer resources when a failure occurs.

The simplicity of the Cloud abstraction can alleviate some of the problems that scientific applications face in the current HPC environment. For instance, the increasingly important and growing Many Task Computing (MTC) applications can benefit from the abstraction of elastic and/or readily accessible resources, ease of use, and the ability to easily scale up, down or out. Finally, Cloud computing can not only help scientists address today's problems more effectively, but also allow them to explore new ways of formulating their application using the abstraction of on-demand access to elastic resources.

1.7 Thesis Overview

This document is organized as follows: Chapter 2 will present the background and related work to this research, focusing on CDS&E applications, their classification, the current challenges in HPC environments, Cloud computing, and its benefits, the benefits of creating HPC Clouds, and a taxonomy of the current

related work in this area. Chapter 3 will cover the design approach and architecture of the proposed framework and the necessary steps in achieving HPC as a Service. Chapter 4 presents the detailed implementation of the prototype framework. Chapter 5 illustrates the usefulness of such framework by presenting a real-world CDS&E application. The overall scenario was demonstrated at the IEEE SCALE2011 Challenge in Orange County, California, and was awarded the first place. Chapter 6 provides the conclusion and outlines the future work and research agenda.

Chapter 2
Background and Related Work

2.1 CDS&E Applications: Classifications and Requirements

The wide range of CDS&E applications [13] have been broadly classified, based on their requirements and execution behaviors, into 3 classes: high performance computing (HPC) [20], high throughput computing (HTC), and many task computing (MTC) [37].

HPC applications are tightly coupled with large amounts of inter-processor communication and typically require large amounts of computing power for short periods of time. In addition to the large scale, fast interconnects among processors are needed. Message Passing Interface (MPI) applications are an example of this class.

On the other hand, HTC applications are usually loosely coupled, where communication between processors is limited or non-existent. Thus, fast interconnects are not required. While HTC applications also require large amounts of computing power, they typically have much longer execution times than HPC applications (months or years, rather than hours or days). MapReduce is an example of this class of applications.

Finally, MTC applications are emerging to close the gap between HPC and HTC applications [38]. MTC applications can consist of loosely coupled tasks, where each of these tasks is a tightly coupled application. In addition, each of these tasks are characterized by running for relatively short periods of time (i.e. seconds or minutes, rather than hours, days or months). Ensemble applications

[12] are an example of this class. MTC applications require fast interconnects, can potentially scale to extremely large scales (Peta & Exa scale), and often involve dynamic workflows that require elastic resources.

2.2 Shortcomings of Current HPC Resources

CDS&E applications require powerful hardware with high-speed interconnects, extremely large scale, and elastic resources, which should be allocated programmatically. While high-end HPC resources (e.g. supercomputers) provide the necessary performance, scale, and fast communication, these systems have a few setbacks that prevent their adoption as a mainstream solution. These setbacks can be summarized as follows:

Complexity: Current supercomputer systems are often complicated to use. They require low-level involvement to manage, allocate, and run applications. Scientists using these systems often spend valuable time administering the system, when this time could be spent on making contributions to their field of study. In addition, supercomputer systems provided by different manufacturers lack a uniform interface. As a result, scientists and engineers must take the time to acquire new system-level knowledge each time they use a different machine.

Lack of Elasticity: Due to the large demand and high cost of operation, these systems must be fully utilized in order for the system owners to offset the total cost of operation (TCO). As a result, the systems are provisioned to ensure their full utilization, which in turn leads to static allocations of these resources. Scientists and engineers have to tailor their applications to fit these resources, which often results in their developed algorithms being constrained by the system they are running on. Furthermore, these static allocations prevent scientists and engineers from running dynamic workflows that require elastic resources; for example, ensemble applications workflows require a changing number of resources

depending on the required resolution of the solution and the convergence rate of the workflow. In the current environment, this is not feasible due to such rigid allocations.

Queues: Due to the system owners' need for 'maximum utilization' of hardware, queuing mechanisms have been implemented on the various systems. A queuing system accepts submission of jobs from all users on the system and then executes them one after the other based on the available machine resources. Due to the large demand for these resources and their scarcity, queue wait times to run applications on such systems, while successfully guaranteed 100% utilization, have grown to the order of months, which in return has limited the access to such resources, and constrained scientists and engineers from truly utilizing these systems. For example, scientists and engineers can never experiment with what-if scenarios due to the long queue times. Furthermore, small bugs in applications (common and sometimes unavoidable in large codes) can result in the early termination of the application. After fixing such bugs, users often resubmit their jobs to the queues and end up having to wait for long periods of time for their re-execution.

Price: While advances in computer architecture have brought about large systems that are relatively less expensive than they were a few years ago, these high-end systems are still quite expensive. Consequently, access to such systems is limited to large universities and national laboratories that can afford them, while researchers in smaller and mid-sized organizations are often left out from using them and are limited to smaller HPC Clusters.

While current state of the art high-end HPC systems are well suited to run CDS&E applications, these systems do not suit all needs of such applications making the current HPC model far from ideal due to the fact that it is complex, expensive, and not widely available. As such, the current HPC model cannot serve as a mainstream solution.

2.3 Cloud Computing

2.3.1 A Dominant Computing Paradigm

Cloud computing [14] has emerged as a dominant paradigm that has been widely adopted by enterprises. 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. They provide 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 the opportunity for IT outsourcing and automation. 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. Finally, Clouds provide a usage-based payment model where users essentially "rent" virtual resources and pay for what they use.

Consolidated and virtualized data centers typically underlie these Cloud services, exploiting economies of scale to provide attractive cost-benefit ratios. Although it is still in its early stages, Cloud computing is already reshaping the 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. A recent report by Gartner estimates that Cloud services will be a $150 billion industry by 2015 [6].

2.3.2 Definitions

Cloud computing aims to make the vision of 'computing as a utility' a reality. In general, a Cloud can be defined as a scalable set of network enabled on-demand IT services with associated Quality of Service (QoS) guarantees [42]. As a utility,

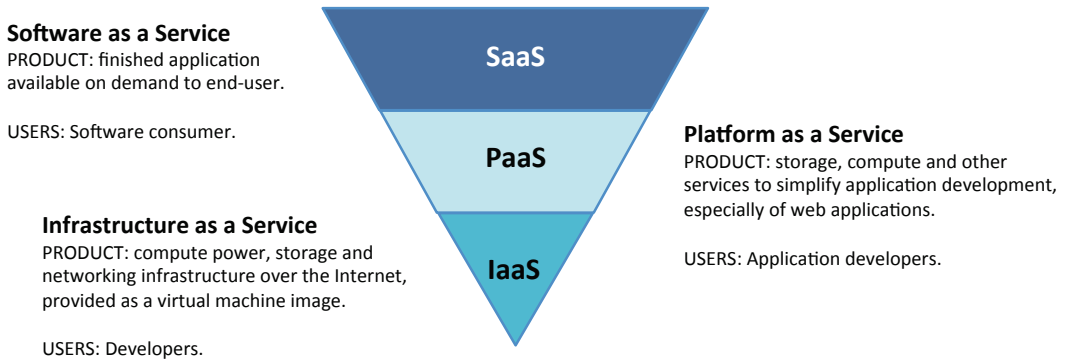


Figure 2.1: Cloud layered architecture (Source: The case for Cloud computing by Robert Grossman [27])

a Cloud can be accessed in a simple and pervasive way. Cloud computing offers elasticity to applications by providing on-demand access to resources over the Internet through a pay-per-use pricing model.

2.3.3 Layers

Cloud computing provides a layered set of service abstractions (software, middleware, and infrastructure) as shown in Figure 2.1.

At the lowest level, Infrastructure as a Service (IaaS) provides raw infrastructure as a service, where users can rent bare virtual machines to increase capacity and customize them as needed. Platform as a Service (PaaS) provides the next abstraction layer of Cloud computing, where users can rent Cloud stacks that they can use to develop and deploy their customized Cloud services. Finally, Software as a Service (SaaS) provides the highest level Cloud abstraction, where users can rent and use complete software solutions that are hosted in the Cloud.

Cloud services thus represent a new paradigm for computing that is drastically impacting price/performance behavior and trade-offs for a wide range of applications and IT services. From a provider perspective, Cloud abstractions support building larger data centers that exploit unprecedented economies of scale [4] and

lead to consolidation, better utilization, and efficiencies. From a users' perspective, Clouds largely eliminate the startup costs associated with setting up a data center, provide cost associativity benefits (i.e. 1000 computers for 1 hour has the same price as 1 computer for 1000 hours), and provide a resilient infrastructure that can be easily configured to the users' preference. Furthermore, hybrid Clouds that federate different Clouds' services can match complex applications requirements, allowing such applications to be deployed easily and effectively.

2.4 HPC and Clouds

2.4.1 Motivations for HPC as a Cloud

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. CDS&E applications are providing unprecedented opportunities for understanding and managing natural and engineered systems, and providing unique insights into complex problems. Recognizing this data and compute driven transformation of science and engineering in the 21st century, the National Science Foundation (NSF) is rethinking the national Cyberinfrastructure as part of CIF21 [10].

Analogous to the role of Cloud computing in enterprise IT, HPC as a Cloud can enable the outsourcing of many tasks related to the undesirable 'scientists as system administrators' effect, such as deploying, configuring and managing infrastructure, and enable scientists to focus on their field of study. HPC as a Cloud and the associated standardization can also improve productivity, facilitate the sharing of research results, and enable the reproducibility of associated computations. HPC as a Cloud can also democratize access to computational and data resources (by providing access to researchers who do not have adequate

local infrastructure), which has been shown to significantly improve research productivity [9]. In fact, a recent survey of DoE users conducted by the Magellan team found that the top motivations for users plugging into the Cloud was ease of access to computing resources (cited by 79%), the ability to control software environments (59%), and the ability to share the setup of software and experiments with peers (52%) [5].

However, it is also critical to look beyond the benefits of outsourcing and understand application formulations and usage modes that are meaningful in a HPC as a Cloud infrastructure. The effective utilization of emerging data and compute intensive application workflows is one such example. Additionally, it is important to think about how this abstraction can enable new practices in science and engineering. The simplicity, pervasiveness, and power of the Cloud abstraction can also potentially alleviate some of the problems CDS&E applications face in the current HPC environments. For example, MTC applications, which are growing in importance, can benefit from an abstraction of elastic and/or readily accessible resources and the ability to easily scale up, down or out.

2.5 Related Work

2.5.1 Current Landscape

Clouds are rapidly joining high-performance computing systems, clusters, and Grids as viable platforms for scientific exploration and discovery. There have been several early experiments aimed at using Clouds to support scientific applications [42]. These include:

- *HPC in the Cloud:* Focused on the outsourcing of entire applications by researchers to current public and/or private Cloud platforms.

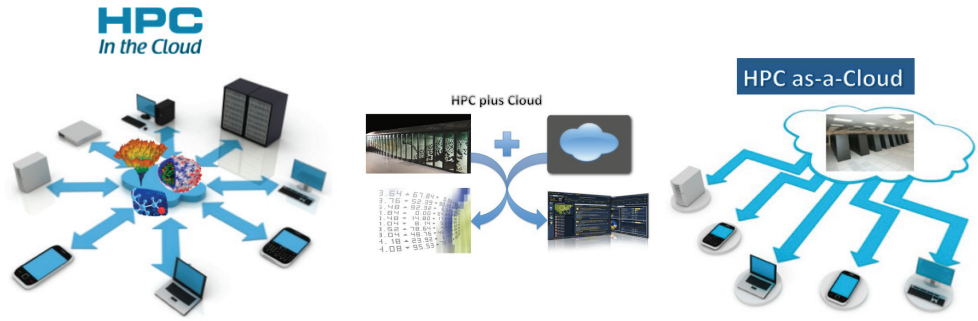


Figure 2.2: Hybrid HPC/Grid + Cloud usage modes for supporting real-world science and engineering applications

- *HPC plus Cloud:* Focused on exploring scenarios where Clouds can complement HPC/Grid resources with Cloud services to support science and engineering application workflows, for example, to support heterogeneous requirements, unexpected spikes in demand, etc.

- *HPC as a Cloud:* Focused on exposing HPC/Grid resources using elastic on-demand Cloud abstractions, aiming to combine the flexibility of Cloud models with the performance of HPC systems.

2.5.2 Existing Approaches and Research Challenges

HPC in the Cloud

Current Cloud platforms can provide effective platforms for certain classes for CDS&E applications, such as HTC applications. There have been several early projects that have reported successful deployments of applications on existing Clouds [18, 21, 30, 31]. Running these applications typically involved using virtualized commodity-based hardware, which is provisioned on-demand by commercial Cloud providers, such as Amazon EC2 or Microsoft Azure.

A recent technical report by G. Fox & D. Gannon has extensively studied running HPC applications in the Cloud [24]. According to this study, commodity

Clouds work effectively only for certain classes of HPC applications. Examples of these applications are embarrassingly parallel applications that analyze independent data or spawn independent simulations that integrate distributed sensor data, science gateways and portals, or data analytics that can use MapReduce-like applications. In addition, research work by Fox et al. [23] and Juve et al. [29] show that different variants of MapReduce computations do well on current Cloud platforms, such as iterative MapReduce. In general, HPC applications with minimal synchronization and minimal communications requirements, small I/O requirements, and modest scales are well suited for current Cloud platforms and can be successfully outsourced to Clouds. In cases where existing application formulations are not directly suited for available Cloud platforms, alternate formulations need to be explored before these applications can benefit from Cloud services. For example, the asynchronous replica exchange [49] formulation is a novel, decentralized, asynchronous and resilient formulation of the replica exchange algorithm for simulating the structure, function, folding, and dynamics of proteins, and has been successfully deployed on existing Clouds.

Research Challenges: Several aspects of current commercial Clouds continue to limit the more general deployment of applications onto Cloud platforms. These aspects include the capabilities and heterogeneity of the typical underlying hardware and the lack of high-speed interconnects to support data exchanges required by these applications. In these cases, if Clouds are the platform of choice (possibly due to availability), then alternate formulations must be explored, such as in the case of the replica exchange example above. The cost and relative performance can still be a concern [18], even in the case of suitable CDS&E applications. For example, the Magellan report cites [46] that Cloud services were found to be 7 to 13 times more expensive. Finally, HPC applications based on loose or bulk synchronizations cannot take advantage of the scalability, elasticity, and fault-tolerance due to the high latency in commercial Clouds. For example, Iosup et

al. [28] evaluated the performance of Amazon EC2 for scientific workloads and found that the reliability and the performance of EC2 are low. However, the performance and reliability of virtualized resources depends heavily on the underlying hardware and network. Younge et al. [47] compared the performance of various virtualization technologies against bare-metal on a high performance cluster and InfiniBand network provided by FutureGrid. The result was that the performance on a virtualized HPC environment is close to that of bare-metal in this case.

HPC plus Cloud

Running CDS&E applications on more traditional HPC resources and bursting onto a Cloud when there is a spike in the demand for computing capacity, or a specific capability is needed to meet heterogeneous applications requirements (also known as Cloud bursting), is an attractive approach for heterogeneous and dynamic application workflows. This holds true especially as CDS&E applications are increasingly becoming end-to-end workflows consisting of coupled simulations and integrated I/O and data analytics pipelines. A hybrid Cloud plus traditional HPC infrastructure can also support new and potentially more effective usage modes and enable new application formulations that use Clouds to achieve acceleration, resilience, or more appropriate cost/power/performance trade-offs. Existing efforts that have explored such hybrid infrastructure include InterGrid [19] and meta-schedulers [16], which interconnect different grids. There are also efforts to include Clouds, such as Amazon EC2, into integrated computing infrastructures. Buyya et al.[17] described an approach of extending a local cluster to Cloud resources using different scheduling strategies, while Ostermann et al. [36] 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. [41] proposed architecture for an elastic grid infrastructure using the GridWay meta-scheduler and extended grid resources to Globus Nimbus. Parashar et al. have also explored such a hybrid-computing infrastructure, which integrated clusters with Clouds [32] and Grids with Clouds [33] and enabled on demand autonomic Cloudbursts using CometCloud [34].

Research Challenges: The federation of traditional HPC/Grid/Cluster resources with elastic Cloud resources provides a hybrid infrastructure with the ability to dynamically add (or remove) capacity as well as capability. While such hybrid infrastructure can support new and potentially more effective usage modes, and even enable new application formulations, its inherent heterogeneity and dynamism presents significant challenges. For example, the different resource classes available in such a hybrid infrastructure can vary significantly in their usage costs, configuration, performance, availability, runtime behaviors and the guarantees for quality of service provided. On the other hand, on the application side, one is often faced with dynamic requirements and constraints. As a result, provisioning and scheduling an appropriate mix of resources for these applications requires considering appropriate cost/performance trade-offs. Furthermore, these requirements/constraints may change, due to a failure or an application event requiring dynamic system and/or application adaptations. As manually monitoring these dynamic behaviors and requirements and enforcing adaptation can quickly become unfeasible, autonomic management approaches become crucial for such workflows. A challenge also exists in programming the framework required for specifying workflow structures as well as requirements and constraints.

HPC as a Cloud

Due to the limitations of commodity Clouds in serving general HPC applications, Cloud providers realized the need to provide Cloud solutions that are built

specifically for HPC applications (e.g. hardware with faster processors and interconnects). Some providers have even provided non-virtualized hardware in order to provide the bare-bone performance that these types of applications require. This is commonly referred to as "HPC as a Cloud" (i.e. running HPC applications on HPC resources that are exposed as on-demand resources using Cloud computing abstractions, in order to take advantage of the Cloud model without sacrificing the HPC performance that scientific applications require). The current approach uses small HPC clusters that can be connected together to form a large Cloud. These HPC clusters can be virtualized or non-virtualized to provide better performance. Amazon EC2 Compute Cluster, Azure, Adaptive Systems, and Platform HPC Computing are commercial providers that use the virtualized approach, while SGI and Penguin Computing on Demand are commercial providers using the physical approach. In addition, Goscinski et al. [26] proposed and validated a similar academic approach for providing HPC as a Cloud using small clusters, known as HPCynergy. While HPCynergy is a more effective approach than commercial offerings, the scale of HPCynergy is relatively small (384 cores) especially for realistic science and engineering applications. Finally, in an early approach, Barcena et al. [15]have tried to create a grid-supercomputing environment, however their approach was specific to a certain problem and a certain system. Furthermore, Wilde et al. [45] proposed a parallel scripting framework to easily deploy MTC applications on supercomputing resources. However, their approach does not solve other HPC challenges such as elastic resources or long queues.

Table 2.1: Summary of Current HPC as a Cloud approaches

Small HPC clusters that are connected together	
Physical Solutions	Virtualized Solutions
e.g. Penguin on Demand, Silicon Graphics, HPCynergy	e.g. EC2 Compute Cluster, Azure, Adaptive Systems, CloudCycle

Research Challenges: HPC as a Cloud has a clear potential to alleviate some of the problems scientific applications face in the current HPC environment, and significantly improve the usability and productivity of traditional HPC overall. However, the current HPC as Cloud offerings have several limitations.

- Scale & Performance: The scale of current commercial HPC as a Cloud is very small (128 processors on Amazon HPC instance vs. 294,912 processors on an IBM Blue Gene/P), and therefore cannot accommodate the requirements of CDS&E applications. In addition, although the performance of these Cloud services is better than commodity Clouds, they are still outperformed by local HPC clusters that are specifically designed to support these applications [48]. Finally, spending more (i.e. scaling beyond a physical cluster) does not guarantee better performance, and performance can in fact get worse (due to communication overhead), contradicting one of the main premises of Cloud computing.

- Provisioning: In HPC as a Cloud, provisioning refers to building a virtual layer to easily configure and access HPC resources in a "Cloud-like" manner. Building this virtual layer can be very challenging. For instance, the layer should be general enough that it can encompass different HPC systems and architectures. However it should also be specific enough to take full advantage of the underlying resources.

- Elasticity and On-demand Access: Elasticity by definition is the abstraction for providing on-demand access to resources and the ability to scale up, down or out as needed. However, most HPC resources currently use a batch-queue system for running applications, which in turn prevents on-demand access to resources. Much research is needed in order to provide elasticity on top of batch-queue systems, and support on-demand access to these systems as needed.

- Programmability: While most web applications running on commodity Clouds are agnostic of the underlying hardware, scientific applications are often customized to the underlying hardware. For instance, maintaining data locality when running scientific applications can minimize communication overhead. Therefore, building scientific applications that are agnostic to the underlying hardware without compromising performance is essential. Finally, exposing Cloud abstractions provided by HPC resources (e.g. elasticity, data locality, and ease of use) to scientific applications is also critical.

2.5.3 Summary & Conclusion

As previously discussed, the initial approaches to providing HPC as a service were aimed at deploying scientific applications on current cloud platforms built on commodity hardware. This approach is usually referred to as *HPC in the Cloud*. While this approach is effective for "suitable" applications, e.g. applications that are loosely synchronized and have minimal communication requirements, it is not effective for more tightly coupled and computationally demanding applications. Reasons for this include the limited capabilities and power of the typical underlying hardware and its non-homogeneity, the lack of high-speed interconnects to support data exchanges required by these applications, as well as the physical distance between machines. Another approach for taking advantage of cloud computing for HPC is using hybrid infrastructure that combines HPC resources and public clouds, i.e., *HPC plus Clouds* or *Cloud bursting*. Cloud bursting provides some advantages to HPC applications such as acceleration, conservation, and resilience, and can support hybrid HPC workflows (e.g. HPC + Data Analytics). However, even with these approaches, the issues related to the provisioning and use of high-end HPC resources remain. While the *HPC as a Cloud* approach is more effective and a better alternative for CDS&E applications, current offerings and research fall short of providing all of the benefits of Cloud computing. In

addition, these offerings at current scale are still very small, their performance is still worse than standalone HPC systems, and they have limited QoS. A summary of the approaches and their advantages is presented in Table 2.2.

Table 2.2: Summary of HPC and Cloud Landscape

	Advantages	**Disadvantages**	**Typical Applications**	**Sample Systems**
HPC in the Cloud	Easy to use, infinite resources	Low performance, expensive	E/P, L/S, A, HTC	Amazon EC2, Microsoft Azure
HPC as a Cloud	Elasticity, on demand access	Small scale, limited QoS, medium performance	S, M, HPC, MTC	EC2CC, SGI, POD
HPC plus Cloud	Acceleration, Conservation, Resilience	Different prices, performance, and availability of resources	HPC + Analytics / Visualization	CometCloud, InterGrid, GridWay

E/P	Embarrassingly Parallel	M	Metaproblems
L/S	Loosely Synchronous	HPC	High Performance Computing
S	Synchronous	HTC	High Throughput Computing
A	Asynchronous	MTC	Many Task Computing

HPC+Clouds: An Attractive Solution or an Elusive Approach?

At this point, it seems like Cloud computing is an elusive approach, which can only partially improve the flexibility of HPC resources, and the cost for such improvement is a degradation in performance. While current research would argue that Cloud computing is ineffective in solving the problems of High Performance Computing, we argue that this is not the case. Instead, we show that Cloud computing can effectively improve the HPC model, without sacrificing the necessary performance for such applications. This research proposes an alternate approach that uses large high-end HPC systems, which can be provisioned as Clouds.

In the remainder of this document, we explore how a cloud abstraction can be effectively used to provide a simple interface for current HPC resources and

support real-world applications. In particular, the benefits of the cloud paradigm, such as ease of use and dynamic allocation, and their application to supercomputers. Specifically, these benefits are discussed, tested, and validated on the supercomputer, IBM Blue Gene/P. The underlying framework essentially transforms the Blue Gene/P supercomputer into an elastic cloud, bridging multiple systems to create HPC federated Clouds and supporting dynamic provisioning and efficient utilization while maximizing ease-of-use through an as a Service abstraction.

In order to effectively illustrate the benefits of such concept, the proposed framework is utilized on a real-world ensemble application, an oil-reservoir data assimilation and history matching application, which consists of multiple instances of reservoir realizations that are run simultaneously and the results are filtered through an Ensemble Kalman Filter (EnKF). The choice of ensemble applications was due to the fact that these applications represent the increasingly important class of MTC applications. In addition, MTC applications require extreme large scale, high performance, fast interconnects, and elastic resources. Finally, MTC applications have a dynamic workflow, thus, it is an ideal example of validating the importance of a Cloud abstraction in HPC environments.

Chapter 3
Framework Design and Architecture

3.1 Design Approach

As discussed in the previous chapter, past approaches to providing HPC as a Service were (1) run HPC applications on commodity Cloud hardware (e.g. Amazon EC2), (2) run a mix of HPC and commodity Cloud resources, and (3) create small scale 'HPC as a Cloud' offerings. Each of these approaches failed to realize that Cloud computing and High Performance Computing resources are different. These attempts have shown that the Cloud abstraction, or at least its benefits, must be applied in a different way to achieve HPC as a Cloud. We propose a method for successfully implementing HPC as a Service as follows:

1. Select an existing HPC hardware that represents similar architecture to most other HPC hardwares, in order to apply the Cloud abstractions to it.
2. Compare this HPC resource to current commodity Cloud resources
3. Differentiate between the Cloud definition, model, abstraction, and benefits
4. Investigate how these layers and functions can be applied to existing HPC resources
5. Analyze the Cloud layers and their corresponding functions
6. Build corresponding HPC as a Cloud layers that apply the Cloud concepts and benefits without sacrificing the high performance and also taking into consideration the limitations (e.g. scarcity) of these HPC resources

In the remainder of this chapter, these design steps will be discussed in details. In Chapter 4, the implementation of a prototype framework that follows these design goals will be presented. The framework will then be evaluated and validated in Chapter 5. Finally, Chapter 6 outlines the future research agenda to generalize the concept of HPC as a Cloud.

3.2 HPC Resource Selection

Supercomputers provide state-of-the-art hardware. They scale much larger than regular HPC clusters (hundreds of thousands of nodes vs. thousands of nodes). When compared to HPC clusters of the same size/performance, supercomputers have faster interconnects coupled with homogeneous and uniform hardware. They are also cheaper and utilize less power by comparison, but have the longest queues, rigid and non-elastic resources, and a complex usage model. Therefore, supercomputers would benefit more than a similar sized HPC cluster from the Cloud abstractions, and thus, they were chosen over HPC clusters to highlight the benefits of such abstractions.

3.3 A Comparison Between HPC and Commodity Cloud Resources

Table 3.1 summarizes the difference between HPC and commodity Cloud resources. This table shows that Cloud resources and HPC resources are different if not almost opposite to one another.

3.4 Cloud Computing Concepts

The rise in the widespread adoption of Cloud computing has caused the definition of Clouds to become ambiguous. As a result, the Cloud computing paradigm

Table 3.1: Comparison between HPC and commodity Cloud Resources

	Cloud Resources	**HPC Resources**
Availability	Instant, on-demand	Queue systems
Cost	Cheap	Expensive
Elasticity	Dynamic	Rigid
Payment Model	Flexible	Rigid
Performance	Low performance	High performance
QoS Guarantee	Focused on availability and up time	Focused on performance system utilization
Resource Sharing	Isolation through virtualization	Collaboration and fair share
Resources	'Infinite' or abundant	Finite and scarce
Scalability	Nodes, sites, and hardware	Nodes
Security	Through isolation	Through secure credentials verification
Software	Domain independent, e.g. web applications	Domain dependent, e.g. CDS&E applications
Usability	Easy to use and manage	Hard to use and manage
Users	Enterprises, IT, and startups	Government, national labs, research institutes, and Academia

started to represent many technologies and take on different meanings. Therefore, it is essential to clarify the difference between the Cloud definition, business model, abstraction, and benefits. This will help to distinguish between the 'as a Service' model as a business model and as a utility abstraction. The Cloud computing concepts can be summarized as follows:

Definition: In general, a Cloud can be defined as a set of network enabled on-demand IT services, scalable and with associated QoS guarantees, which can be accessed in a simple and pervasive way as a utility [42].

Business Model: Usually refers to a business model, where Cloud providers with large virtualized data-centers provide different solutions (infrastructure, platform, and software) as a Service for a fee. Nonetheless, this business model should not define or confine Cloud computing to its mere implementation. For instance,

virtualization ensures efficient utilization of the underlying resources and provides better consolidation and security measurements. Therefore, virtualization is widely used by Cloud computing service providers (e.g. Amazon or Microsoft), but it is not an essential component of Cloud computing.

Abstraction: Usually refers to abstractions such as infinite resources, scalability, elasticity, instant on-demand access, ease of use, cost associativity, and pay per use payment model

Benefits: Usually refers to benefits of the previous concepts such as economies of scale, effective cost-benefit ratios, cheap resources, efficient resource utilization and consolidation, eliminating startup costs, fault tolerance, and resilience.

3.5 Applying Cloud Concepts to HPC Resources

Table 3.2 summarizes which of these concepts should be applied to the HPC resources, which concepts should not be applied, and which concepts are not applicable.

3.6 Cloud Computing Layers

In addition to the previous Cloud concepts, one must also differentiate between the different Cloud computing layers and their corresponding functions. Table 3.3 summarizes these layers and their functionality.

3.7 HPC as a Cloud Corresponding Layers

Looking back at Cloud computing and its three layers (IaaS, PaaS, and SaaS), it was clear that similar layers are necessary to convert a HPC system to a Cloud. Starting at the bottom layer and moving up the stack, three similar layers were also formulated. These three layers of HPC Cloud computing provide the concepts

Table 3.2: Application of Cloud Concepts to HPC Resources

Concept	**Application to HPC Resources**
Business model	No
Cheap Resources	No
Cost Associativity	Yes
Ease of Use	Yes
Economies of Scale	N/A
Efficient Resource Utilization & Consolidation	Yes
Elasticity	Yes
Eliminate Startup Costs	Yes
Fault Tolerance & Resilience	Yes
Infinite Resources	N/A
Infrastructure, Platform, and Software as a Service	Yes
Instant/On-demand Access	Yes
Pay Per Use	Yes
QoS Guarantees	Yes
Scalability	Yes
Virtualization	No

No = Should not be applied, Yes = Should be applied, N/A = Not applicable

and benefits of Clouds, while maintaining the performance, and considering the limitations of such HPC resources. Table 3.4 summarizes these layers and their functionality.

Table 3.3: Summary of Cloud Computing Layers

Cloud Layer	**Definition**	**Functions**	**Users**	**Examples**
Infrastructure as a Service	Raw of virtualized infrastructure that users can rent and customize as needed	Most control of resources (hardware and software). Abstraction of unlimited and elastic resources through a pervasive interface	System developers and administrators	Amazon EC2 [1]
Platform as a Service	Users can rent Cloud stacks that can be used to develop and deploy their customized Cloud services	Simplify application development and deployment. Autonomous scalability of the application based on the workload.	Application developers	Google App Engine [7]
Software as a Service	Users can rent and use complete software solutions, which are hosted in the Cloud	Complete applications available on-demand to end-user	Software users	Gmail [8]

Table 3.4: Cloud Corresponding Layers to HPC

HPC as a Cloud Layer	**Definition**	**Functions**	**Users**	**Examples**
Infrastructure as a Service	users can rent and customize HPC resources	Abstraction of unlimited and elastic resources, can be accessed instantly through a pervasive interface	Workflow developers who require direct control of HPC resources	customize the number of nodes to run a particular job during runtime autonomically
Platform as a Service	Users can rent Cloud stacks that can be used to develop and deploy their customized Cloud services	Simplify application development and deployment. Autonomous scalability of the application based on workload	Scientist developing CDS&E Application	Provide EnKF as a Service, build domain specific ensemble applications, which can be deployed autonomously
Software as a Service	Users can rent and use complete software solutions, which are hosted in the Cloud.	Complete CDS&E applications available on-demand to end-user	Software users such as scientists, researchers, and field engineers	Providing an oil reservoir history matching application, which can help field engineers make informed decisions in real time. Researchers can utilize software written by other scientists to run experiments using their own data, and allow for data and result sharing

Chapter 4
Prototype Framework Implementation

The proposed framework mainly integrates two key components: Deep Cloud and CometCloud. In addition, new tools have been developed to complete the framework. The overall architecture is presented in Figure 4.1.

4.1 IBM Blue Gene/P

Blue Gene/P [40] is the second generation of the IBM Blue Gene Systems. Blue Gene/P is designed to run continuously at 1 PFLOPS. The system consists of compute nodes, I/O nodes, front-end nodes, a service node and a file server. Each compute node (compute card) consists of 4 cores (processors). 32 compute nodes are put on a node card along with up to 2 I/O nodes. 32 node cards form a rack, and up to 72 racks can be connected together. These compute nodes can be partitioned into partitions of size 32, 64, 128, 256, 512, etc. A 32-node partition can run up to 128 processors. Finally, a 3D torus network connects all compute nodes in the system, and a collective network connects all compute and I/O nodes. Figures 4.2, 4.3, and 4.4 show an overview of the Blue Gene/P system, its architecture, and general configuration.

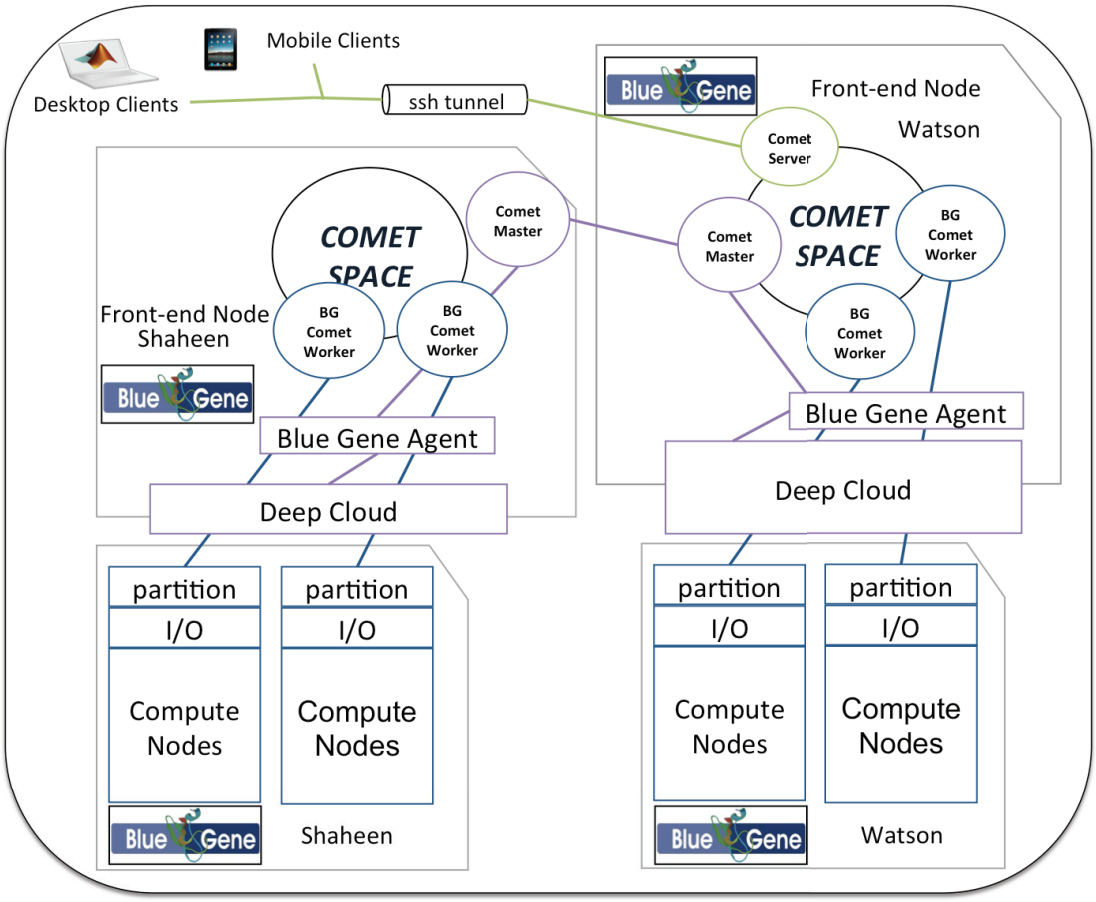


Figure 4.1: Overall Architecture of Proposed Framework

Figure 4.2: Blue Gene/P System at Rutgers University

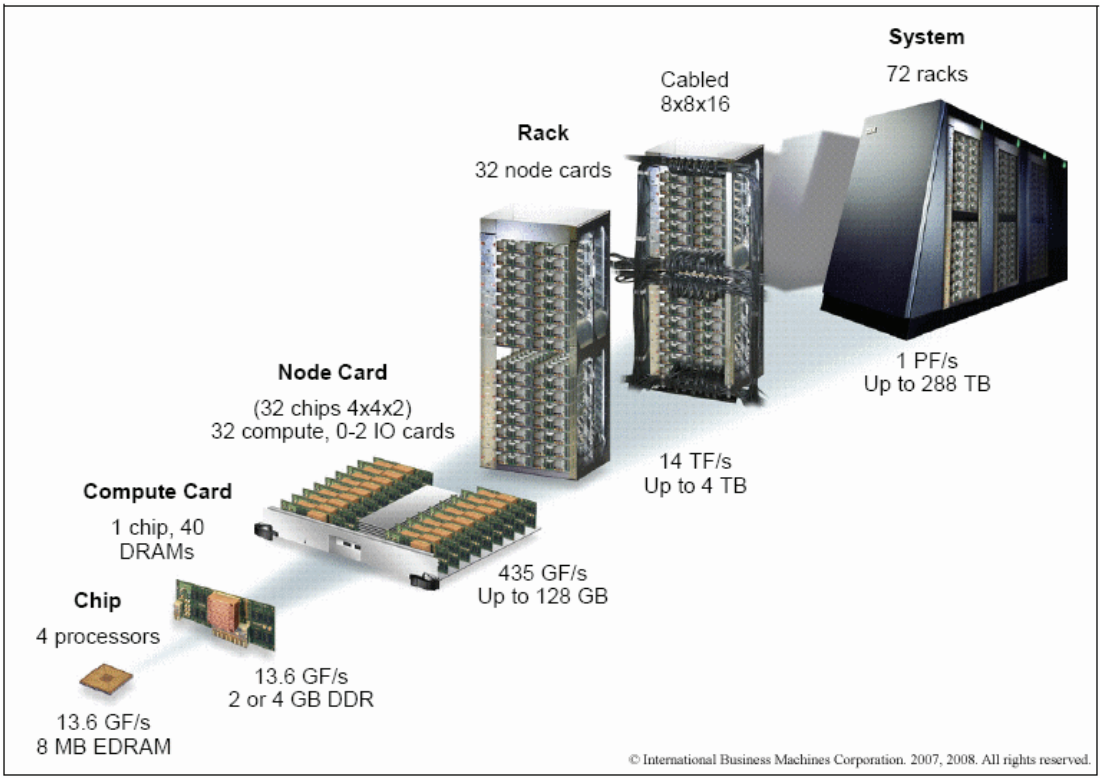


Figure 4.3: Blue Gene/P Architecture [40]

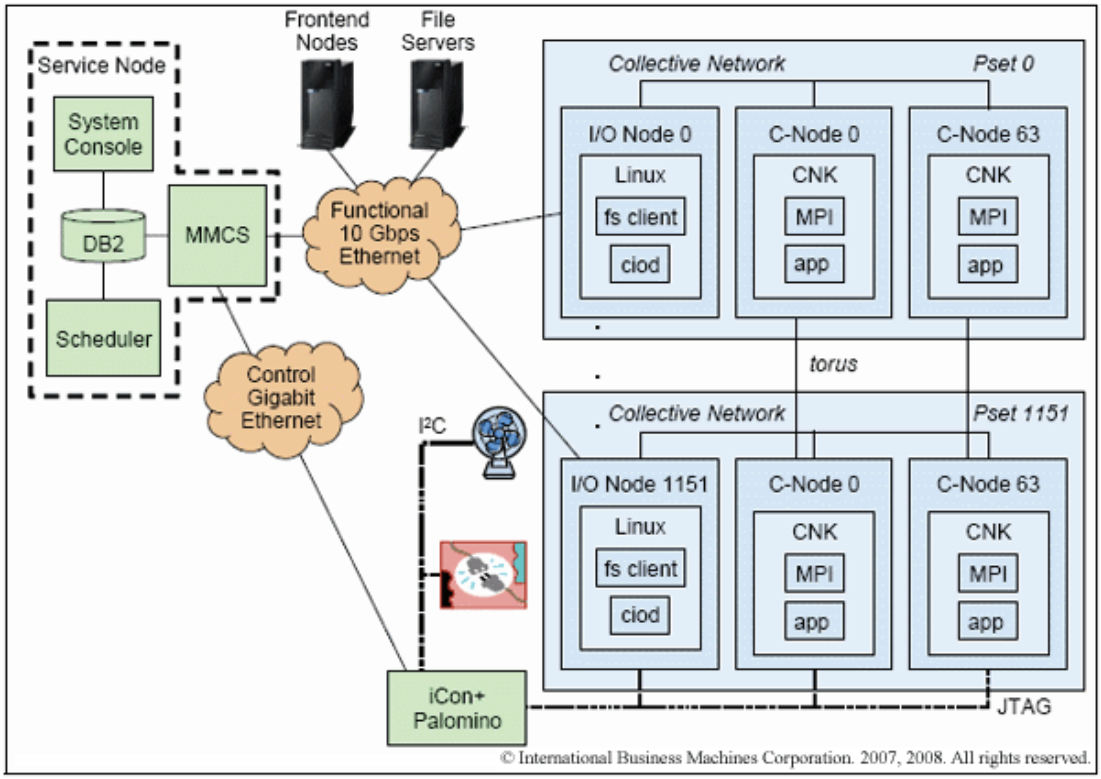


Figure 4.4: Blue Gene/P General Configuration [40]

4.2 Infrastructure as a Service on Blue Gene/P using Deep Cloud

In contrast to today's Infrastructure as a Service (IaaS) clouds (which provide on demand access to 'infinite' resources on a pay-per-use model), current supercomputers continue to rely on batch processing of jobs. While batch processing allows the scheduler to better utilize the underlying HPC resources, if not carefully managed, the queuing delay can result in excessive wait times, often lasting weeks. Fundamentally, there are two gaps in today's HPC resource model (as compared to the one offered by IaaS clouds). The first is the lack of on-demand, instantaneous access to HPC resources. For example, today's HPC resources cannot handle interactive workloads, and the HPC schedulers cannot handle tasks requiring elastic scaling of their resource needs. While some schedulers do allow on-demand access, the underlying support is typically an afterthought rather than a first order design principle. The second gap in the resource model is the awareness of resource limitations. A typical HPC job must not only specify the amount of required resources, but must also be aware of the resource limitations of the underlying HPC systems. In contrast, Cloud users are given the illusion of infinite capacity which allows them to get the resources that they need when they need them and only pay for what they use.

To address these two gaps (on demand access and illusion of infinite resources), Deep Cloud [39] was implemented. At its core, Deep Cloud is a reservation-based system backed by a dynamic pricing engine. It is currently being developed at IBM's T.J. Watson Research Center for managing supercomputer resources to (1) provide users with an abstraction of unlimited resources and (2) maximize users' satisfaction by shaping their demand. The dynamic pricing engine in Deep Cloud follows the same design principle as pricing models implemented across a number of industries (e.g., hotel, transportation, and airline industries), where

manipulating the price can control the demand of finite resources. The engine exposes the availability of resources to users, allowing them to decide when to run their jobs based on different price points. By carefully picking the price for different configurations, the engine flattens the users' demand curve. Intuitively, the engine adapts the price such that the price of using the system is highest at times where demand is also high. As a result, users with limited budget and flexible execution times will run their jobs when the system is lightly loaded, allowing users with larger budgets and pressing needs to meet their deadlines.

In addition to future reservations, Deep Cloud allows for on-demand allocation of resources, similar to the spot market on Amazon EC2. The pricing for on-demand access is dependent on the system utilization, availability, and the current demand for resources. Overall, the ability of users to decide when to run their jobs maximizes their satisfaction, because they can plan ahead for their resource allocations. This is especially true when compared to the traditional HPC model that queues users' jobs without providing any explicit runtime guarantees. Similar to mainstream cloud providers, Deep Cloud allows Blue Gene/P resources to be programmatically reserved and instantiated. It also supports on-demand provisioning and repartitioning of the resources. Deep Cloud can run (albeit in a reduced functionality mode) on top of existing batch systems to support future reservations. If proper access is given, it can bypass the scheduler's job queues and communicate directly with the resource database to allow instant allocation of resources. Figure 4.5 shows the architecture of Deep Cloud.

4.3 Platform as a Service on Blue Gene/P using CometCloud

The PaaS layer is the next layer that provides a complete solution stack. PaaS masks the underlying hardware from the user thus providing ease of use and

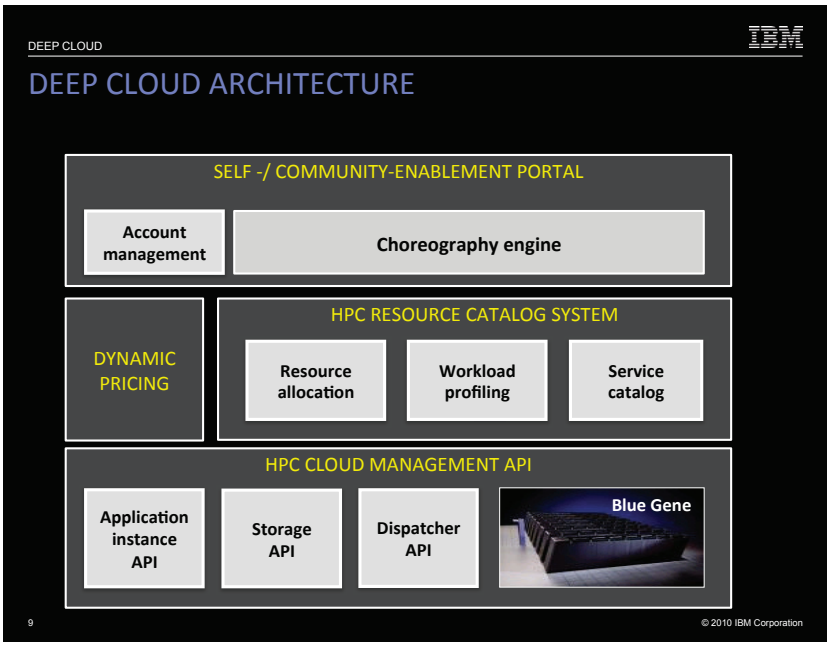


Figure 4.5: Deep Cloud Overall Architecture [39]

autonomic elasticity. PaaS also can allow for multiple cloud federation, which would allow scientists to take advantages of bridging multiple supercomputer systems, without worrying about the physical infrastructure. The CometCloud framework, currently being developed at Rutgers University, was used to provide the PaaS layer.

4.3.1 Overview of CometCloud

CometCloud [34, 3] is an autonomic computing engine that enables dynamic and on-demand federation of Clouds and grids as well as the deployment and robust execution of applications on these federated environments. It supports highly heterogeneous and dynamic Cloud/grid infrastructures, enabling the integration of public/private Clouds and autonomic Cloudbursts, i.e. dynamic scale-out to Clouds to address dynamic requirements for capabilities, heterogeneous and dynamics workloads, spikes in demands, and other extreme requirements. The CometCloud programming layer provides a platform for application development

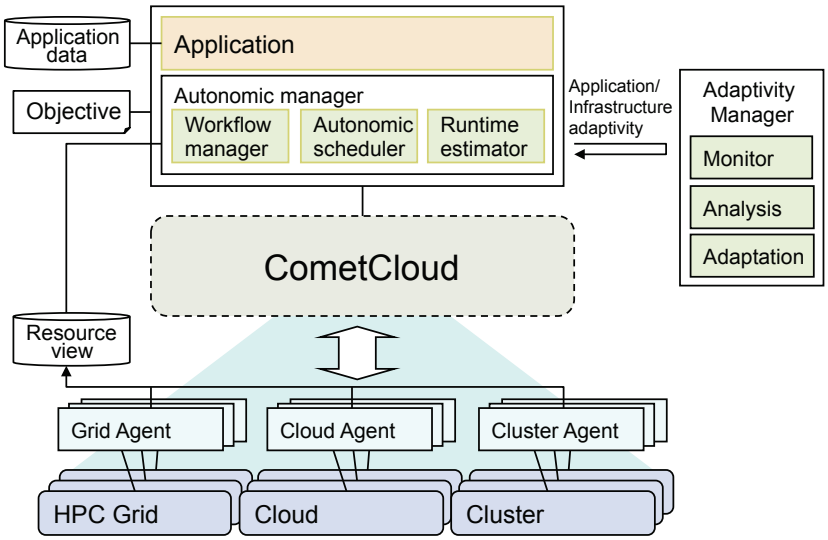


Figure 4.6: Architectural Overview of the Autonomic Application Management Framework and Supported Platforms

and management. It supports a range of paradigms including MapReduce, Workflow, and Master/Worker/BOT. The CometCloud autonomic management layer also enables the autonomic management of application workflows as well as the federated infrastructure and ensures that application objectives and constraints are satisfied. The autonomic management framework can provision the appropriate mix of HPC/Grid and public/private Cloud resources based on application requirements and constraints, monitor system/application state (e.g., workload, availability, delays) and adapt the application and/or the resources (e.g., change algorithms used or re-provision resources) to respond to changing applications requirements or system state. A schematic overview of the CometCloud-based autonomic application management framework for enabling hybrid HPC platform usage modes is presented in Figure 4.6.

4.3.2 CometCloud on Blue Gene/P

We used the Master/Worker programming framework from CometCloud. The main components are a Comet Server, Comet Master, Comet Workers, and the

Comet Space. The Server is responsible for handling requests from the user and starting the corresponding Master. The Master is responsible for the execution of the overall workflow and for guaranteeing the execution of all tasks in the Comet Space. Usually, each task is an instance of the traditional HPC application, with different parameters, such as input, number of processors, mode of operation, etc. A set of these tasks is inserted into the Comet Space. Workers then connect to the Space, pull a task, and execute it on a given partition. The pull model guarantees load balancing, since not all tasks have the same runtime. In case a task fails due to a system failure, the Master reinserts that task back into the Space to be reprocessed, providing a layer of fault tolerance as well.

The Master is also responsible for adjusting the size of the allocations according to the workload in the Comet Space, thus providing the programmatic elasticity. This is achieved through a Blue Gene Agent. The Blue Gene Agent is the link between CometCloud and Deep Cloud. The Comet Master communicates with the Agent to adjust the Blue Gene/P allocations. The Agent also takes into consideration user policy, such as budget and time to completion, and adjusts resources accordingly. The Agent communicates with the Deep Cloud API to programmatically allocate or release resources, and get prices or availability. Overall, the Blue Gene Agent addresses the elasticity of HPC resources from a user perspective as well as from a programmatic perspective. The Comet Space can be shared across multiple front-end nodes of multiple Blue Gene/P systems to provide a federated cloud of supercomputing resources, see Figure 4.7.

A synchronization step can be performed between the different systems before a new set of tasks is processed. Workers running on each system execute tasks separately and then return the results back to their respective front-end. CometCloud also provides loose code coupling, where the Comet Workers are responsible for executing the parallel instances of the traditional HPC application, and the Comet Master can couple their results if necessary.

Figure 4.7: Federated HPC Clouds

4.4 Software as a Service on Blue Gene/P

In addition to the features provided by Deep Cloud and CometCloud, the framework introduces a few other features, mainly, thin client access, interactive monitoring and steering, programmability and ease of use. These features are discussed in more details below.

4.4.1 Easy Access & Interactive Monitoring

Thin clients: Thin clients, such as an Apple iPad [2], can connect to the framework and launch a simulation by adjusting a few parameters. Accessing such a large supercomputer configuration from an iPad to easily run a complex application makes supercomputing much more accessible to scientists and engineers. Mobile computing is on the rise, and allowing field engineers to tap into such system as easily as possible will allow these engineers to run live simulations that can influence their design or decisions and increase their productivity. Using this simple interface, users can setup an experiment, adjust budget or time to completion, monitor and steer the workflow, and get interactive results. Figure 4.8 shows an iPad interface used for running supercomputing experiments.

DISCOVER: In addition to the thin client access, the proposed framework utilizes DISCOVER. DISCOVER [35] is an attempt to develop a generic framework that will enable interactive steering of scientific applications and also allow

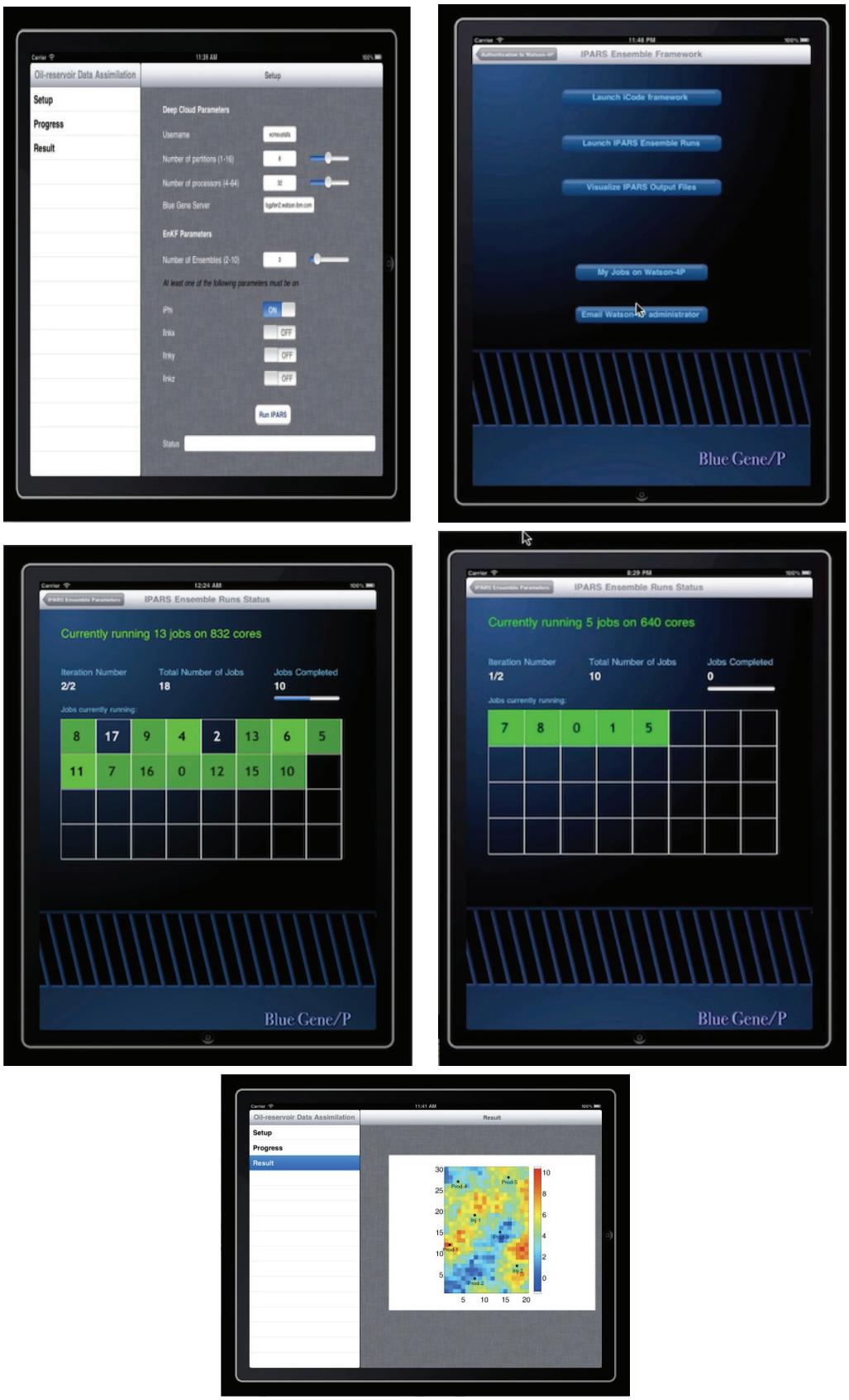


Figure 4.8: Apple iPad Interface

for collaborative visualization of data sets generated by such simulations. The goal of DISCOVER is to develop a computational infrastructure that will have the potential of transforming large-scale distributed scientific and engineering simulations into interactive and collaborative ones where numerous scientists, geographically distributed, will monitor, analyze, and steer scientific computations.

Scientific simulations play an increasingly critical role in all areas of science and engineering. The simulations of expensive scientific problems are necessary to validate and justify the investment to be made. Interactive computational steering helps increase the performance of simulations for scientists as it allows them to drive the simulation process by interacting with their data. In addition, collaboration between scientists situated in geographically distributed locations would significantly increase the understanding of the complex phenomenon underlying these simulations.

DISCOVER is currently being developed at Rutgers University. Figure 4.9 shows the DISCOVER Portal. DISCOVER aims to build a framework that can be used to integrate a suite of applications spanning multiple disciplines including:

- Numerical relativity and relativistic astrophysics
- Subsurface modeling and oil reservoir simulations
- Dynamics response of materials
- Turbulence modeling, shock- density inhomogeneous interaction
- Seismic whole-earth modeling

4.4.2 Programmability & Ease of Use

The proposed framework also masks the complexity of the supercomputer and provides scientists with an easy to program interface. This allows scientists to

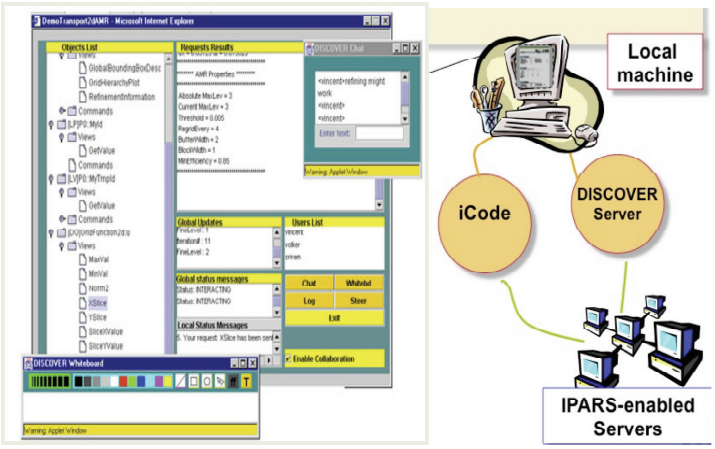


Figure 4.9: DISCOVER Portal

focus on their research and spend less time on nonscientific activities such as learning the system itself or manually allocating and adjusting resources and executing the workflow. The proposed framework was designed to integrate new applications easily and expose these applications to engineers through thin clients as Software as a Service. Integrating new applications can be done using an XML file that defines the roles of the Comet Master and Workers. Another XML file exposes the policy to be used by the Blue Gene Agent for adjusting resources. Scientists can also develop their own Comet Master and Workers if they desire. An API as well as an IDE plugin are provided to allow users to integrate or even write new applications using the proposed framework. Figure 4.10 shows the overall architecture of the proposed API while Figure 4.11 shows an example of the XML file.

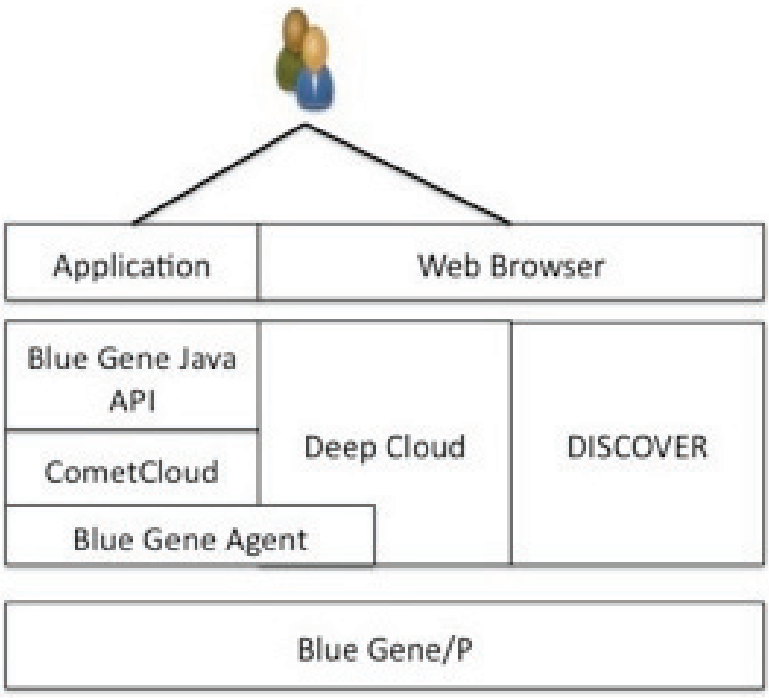


Figure 4.10: Proposed API

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Application>
    <Task>
        <Executable>    ipars   </Executable>
        <Location>  /gpfs/DDNgpfs3/xcmoustafa/ipars </Location>
        <TaskType> HPC  </TaskType>
        <TaskExecution> CN  </TaskExecution>
    </Task>
    <Task>
        <Executable>    initialize   </Executable>
        <Location>  /gpfs/DDNgpfs3/xcmoustafa/init </Location>
        <TaskType> Single  </TaskType>
        <TaskExecution> FEN  </TaskExecution>
    </Task>
    <Task>
        <Executable>    EnKF   </Executable>
        <Location>  /gpfs/DDNgpfs3/xcmoustafa/enkf </Location>
        <TaskType> HTC  </TaskType>
        <TaskExecution> CN  </TaskExecution>
    </Task>
</Application>
```

Figure 4.11: XML Integration

Chapter 5
Prototype Framework Evaluation

5.1 Background

5.1.1 Ensemble Applications

Ensemble applications are an example of the MTC class of applications. Ensemble applications [12] represent a significant class of applications that can effectively utilize high-end Petascale and eventually Exascale systems. For instance, ensemble applications can help achieve an optimal Enhanced Oil Recovery (EOR) production strategy. Such a strategy could pay for several hundred thousand cores of a large computer system. In real time reservoir management, data from various sources, including sensors in the field, are being assimilated to maximize oil production, which leads to an Instrumented Oil Field of the Future, see Figure 5.1. The instrumented oil field through ensemble data assimilation allows for better reservoir management decisions by reducing risk leading to smarter oil production, i.e. producing more oil with greater confidence and less investment.

Ensemble applications explore large parameter spaces in order to simulate multi-scale and multiphase models and minimize uncertainty associated with a single simulation. This is achieved by running hundreds to thousands of realizations simultaneously. Every time new observed data is generated, it is integrated into each of the models to update the realizations for the next step. Each realization can be a traditional parallel HPC application that requires a varying number of processors and fast communication among processors. In addition, typically a

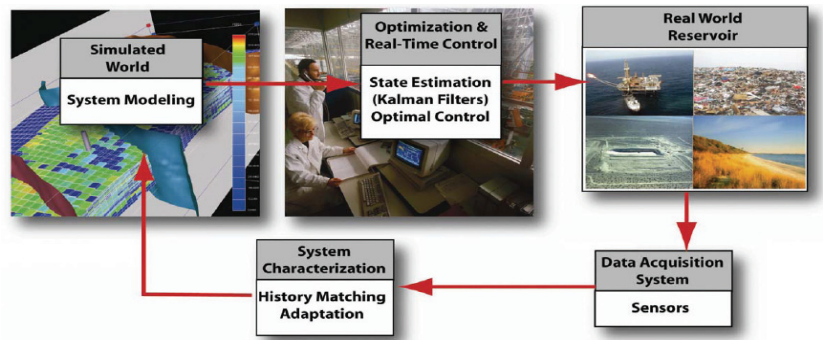


Figure 5.1: Instrumented Oil Field of the Future

large and varying number of realizations, also referred to as 'ensemble members', are also required to achieve acceptable accuracy, which in turn requires a very large and dynamic pool of HPC resources. In addition to elasticity from an application perspective, elasticity from a user perspective (cost vs. turnaround time) would also be desirable. Furthermore, usually the code for the traditional HPC application and the ensemble filter are written by different groups of scientists or at different times, therefore, easy code coupling between the filter and application is also useful. Finally, in addition to the extremely large scale, the elastic resources, and the easy code coupling required by ensemble applications, these applications would benefit from an easy to use workflow engine that guarantees the execution of the ensemble workflow.

5.1.2 Integrated Parallel Accurate Reservoir Simulator

Integrated Parallel Accurate Reservoir Simulator (IPARS) [43, 44] provides a framework and a growing number of physical models suitable for research and practical applications. Both oil reservoirs and aquifers can be simulated using either black oil or compositional equation-of-state fluid models. IPARS can solve problems involving several million grid elements in parallel. It can handle multiple fault blocks with unaligned grids and problems that involve different physical models in various regions of the reservoir.

5.1.3 Ensemble Kalman Filter

The Ensemble Kalman Filter (EnKF) [22] is a Monte Carlo formulation of the Kalman filter, which consists of a forecast step and an assimilation (update) step. The forecast step is equivalent to running the reservoir simulation (IPARS) for a suite of different parameters for the reservoir model (a set of realizations) independently to predict data at the next data assimilation time step. Based on difference between the observed and predicted data, in each assimilation step the Kalman update equation is used to update the model parameters (permeabilities, porosities) and simulation dynamic variables (pressure, saturation, etc). The construction of the Kalman gain matrix requires collecting a very large state vector from each ensemble member. This process is continuously repeated in time as new dynamic data (production data) becomes available.

The EnKF workflow (see Figure 5.2) presents an interesting use case due to the heterogeneous computational requirements of the individual ensemble members as well as the dynamic nature of the overall workflow. Ensemble Kalman filters are recursive filters that can be used to handle large, noisy data; the data in this case are the results and parameters from ensembles of reservoir models that are sent through the filter to obtain the "true state" of the data. Since the reservoir model varies from one ensemble to another, the runtime characteristics of the ensemble simulation are irregular and hard to predict. Furthermore, during execution, if real historical data is available, all the data from the different ensembles at that simulation time must be compared to the actual production data before the simulations are allowed to proceed. This translates into a global synchronization point for all ensemble-members in any given stage, which can present challenges. Figure 5.2 illustrates the variability between stages of a typical ensemble Kalman filter based simulation. The end-to-end application consists of several stages, and in general, at each stage, the number of models generated varies in size and duration.

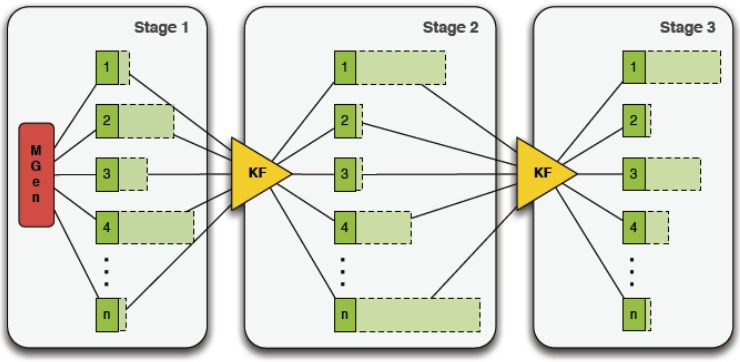


Figure 5.2: Stages of a typical ensemble Kalman filter based simulation

5.2 Experimental Setup

The overall application scenario is presented in Figure 5.3. The workflow involves multiple stages. Each stage consists of many instances of IPARS running simultaneously, a black box, and a computationally intensive oil-reservoir history matching application. The results of each stage are filtered through an Ensemble Kalman Filter. Each IPARS realization requires a varying number of processors and fast communication among these processors. The number of stages and number of ensemble members per stage are dynamic and depend on the specific problem and the desired level of accuracy.

CometCloud is responsible for orchestrating the execution of the overall workflow, i.e. running the IPARS instances and integrating their results with the EnKF. Once the set of ensemble members associated with a stage have completed execution, the CometCloud workflow engine runs the EnKF step to process the results produced by these instances and generate the set of ensemble members for the next stage. The Blue Gene Agent then dynamically adjusts resources (scales up, down, or out) to accommodate the updated set of ensemble members.

Deep Cloud is responsible for the physical allocation of resources required to execute these tasks. The Blue Gene Agent communicates with the Deep Cloud to obtain information about the current available resources. Using this information,

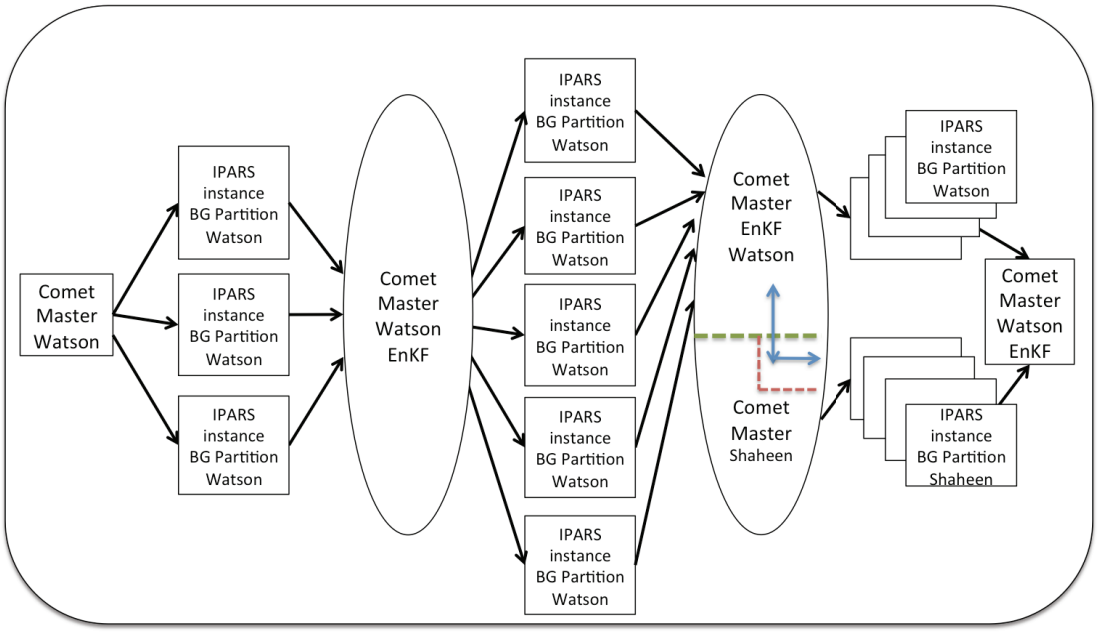


Figure 5.3: Overall Workflow

the Agent requests allocation of required Blue Gene/P partitions to Deep Cloud and integrates them into the CometCloud federated cloud. The partitions, which are no longer required, are deallocated. The entire process is repeated until the application objectives, i.e. all the available observed data is integrated, and then all resources are released and final updated model parameters and predicted data from ensemble members returned to the user.

5.2.1 Computational Example

The presented scenario models a two-phase flow (oil and water) in a three-dimensional horizontal reservoir with a 50x50x4 uniform grid. The model parameters are horizontal permeability and porosity. There are two water injection wells with constant injection rate, and seven producers constrained by constant liquid production rates. The injectors and five of the producers start operating at the beginning of the simulation time while the other two producers start production after 1000 days. The observation data to be assimilated in this demonstration include oil production rate, flowing bottom-hole pressure, and water-oil

ratio recorded every 100 days. The historical data are available for 3000 days and the total simulation time is 4500 days.

The experimental setup follows the application scenario described in Figure 5.3. The experiment starts by running the reservoir simulator forward with 10 initial ensemble members, each one requiring 128 processors. The experiment starts using 5 partitions (640 processors). Figure 5.4 shows the overall flowchart of the framework. Figure 5.4a shows the first step, where using an iPad (see Figure 4.8), a user submits a request to run the experiment with the given parameters. A Comet Server receives the request and starts an IPARS Comet Master. Based on the given information, the Master inserts 10 tasks into the Comet Space. The Master notifies the Blue Gene Agent of the size of the Space, and the number of partitions given by the user. The Agent, based on the size of the Space, and the resources provided by the user, requests 5 partitions from Deep Cloud. Deep Cloud starts up 5 32-node partitions. The agent then starts 5 Comet Workers, each mapped to a partition. The Workers pull tasks (IPARS instances) from the Comet Space and execute them on the corresponding partitions. Once a task is complete, the location of the results is sent back to the Master. The Master runs an EnKF step to generate the next set of inputs.

In step 2, Figure 5.4b, the user has the option to increase the number of partitions to 10 to achieve a faster turnaround time (1,280 processors). This part of the demonstration is intended to show the simplicity of use and dynamic scale up of the framework. Using an iPad interface, the user increases the number of partitions. The Server receives the request and notifies the Comet Master. The Master relays the information to the Agent, as well as the number of tasks (10) inserted into the Space. The Agent requests from Deep Cloud 5 new partitions, and once allocated, the Agent starts 5 more Comet Workers. The 10 Workers pull tasks from the Space simultaneously and execute them on different Blue Gene/P partitions. Note that the number of partitions allocated is at most equal to the

number of tasks in the Space, which guarantees the fastest turnaround time. Thus, if any of the 10 ensemble members would crash and the filter decides not to forward such members in time, a fewer number of partitions would be allocated autonomously, with no interference from the user.

Once all ensemble members' results have been returned, the Master runs another EnKF update step to generate a new set of input. At step 3, Figure 5.4c, the filter decides that since the number of ensemble members is small, the ensemble members will collapse. This decision is made by computing the standard deviation of the updated model parameters. As a result, the filter will redo the step by dynamically increasing the number of filter ensembles to 150 members. Given that the user has requested a faster turnaround, and in order to accommodate the increase in compute demand, the framework will dynamically scale up to the full potential of the IBM Blue Gene/P at Yorktown Heights, NY (128 32-node partitions, 16,384 processors). In addition, the framework will also scale out to run dynamically on a second Blue Gene/P, in this case located in Saudi Arabia, to run the remaining 22 ensemble members (22 64-node partitions, 5,632 processors).

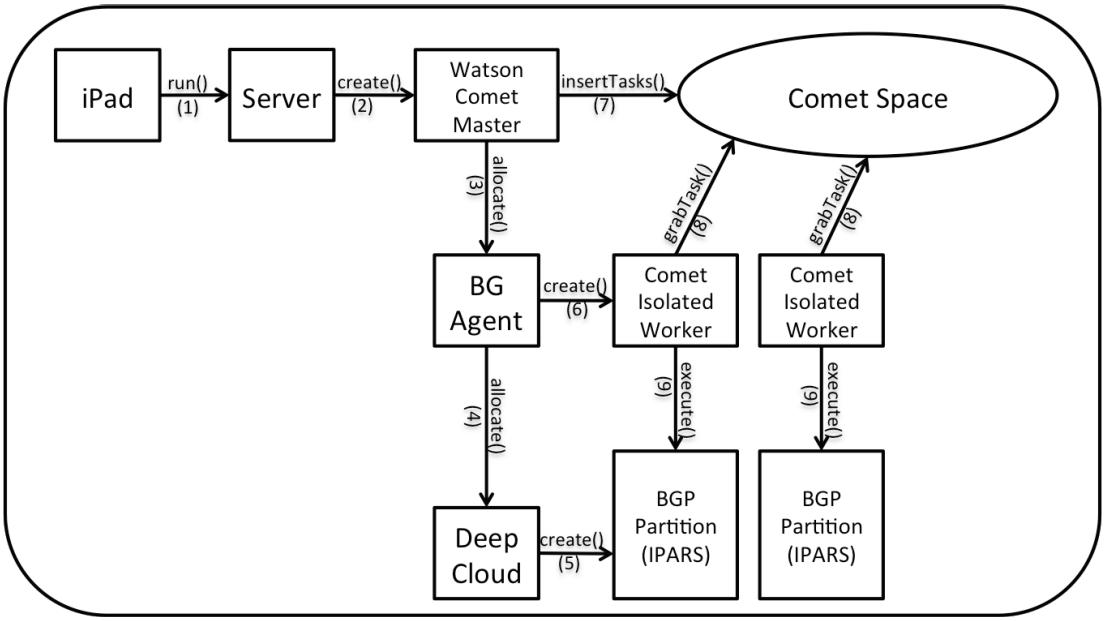
The EnKF update step requests from the original Master running at Watson in New York to increase the number of ensembles to 150 members. The Master requests 128 partitions from the Blue Gene Agent on the Blue Gene/P at Watson (Watson4P) and inserts 128 tasks into the Comet Space on Watson. The Master also connects to Shaheen, a Blue Gene/P system in Saudi Arabia at KAUST, starts a second Comet Master there, and asks it to run the remaining 22 tasks. Even though the front-end nodes of the two systems could share the same Comet Space, meaning any of the systems can run any of the 150 tasks, the Space is separated to reduce data transfers between the two systems. Otherwise, the input and output of all 150 ensemble members would have to be synchronized across the two systems, because they do not share a common file system. In this case,

there is no need to replicate the data between the two systems, but future work will address the data transfer more effectively to allow for a better load balance, task distribution, and execution among systems.

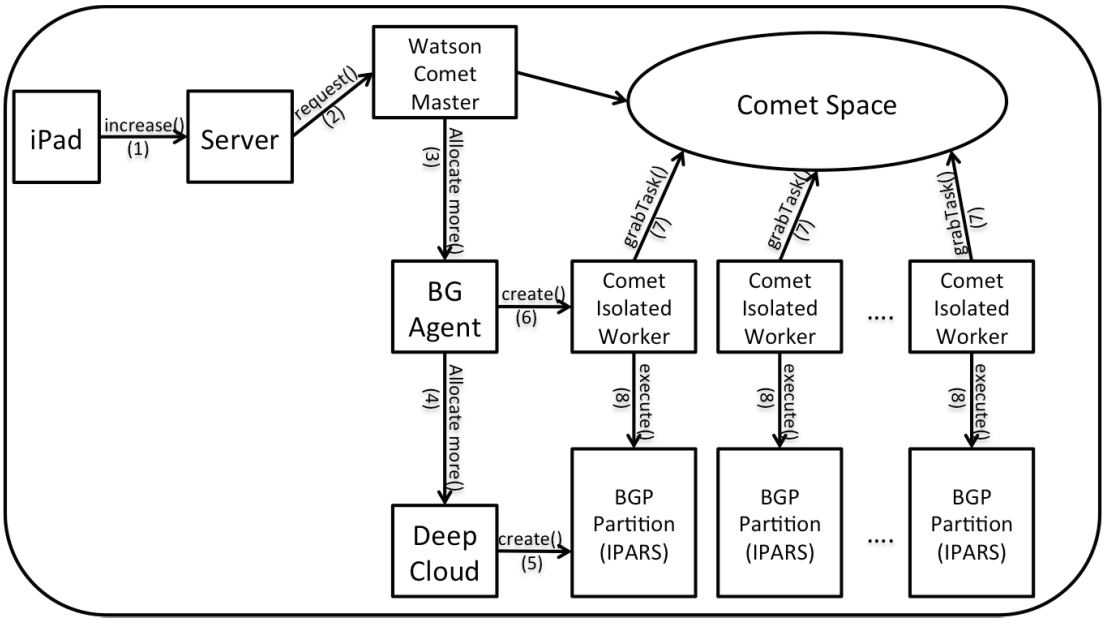
The Shaheen Master is only responsible for executing the 22 remaining tasks, so it requests from the Blue Gene Agent running on Shaheen to allocate 22 Workers. On Shaheen, jobs are run through a batch queue system. Therefore, using Deep Cloud, a reservation ID is assigned to the jobs executed by the Comet Workers. Deep Cloud then communicates with the queue to execute these tasks. The reservation ID translates to a higher priority in the queuing system, and guarantees the execution of the jobs on the Shaheen partitions with no delay. Once the jobs are executed on both Watson and Shaheen, as in Figure 5.4d, the Shaheen Master gathers and compresses information from the 22 ensemble members and sends this information to the Watson Master. In essence, the data that is being transferred between Shaheen and Watson consists of one EnKF state vector per each of the 22 ensemble members, and contains model parameters, primary variables and predicted data. Then, the Watson Master runs an EnKF update step and generates input (updated state vectors) for the 128 ensembles running on Watson, as well a compressed file with the updated state vectors that gets sent back to the Shaheen Master. Once the Shaheen Master receives this file, the updated input files for the remaining 22 ensembles are generated. The entire process is repeated until all the observed data is integrated, and then all resources are released and final results are returned to the user.

5.3 Comparison to Existing HPC as a Service Approaches

We mentioned earlier that the three main approaches for integrating HPC and Cloud computing are HPC in the Cloud, HPC plus Cloud, and HPC as a Cloud. Our proposed framework falls under the HPC as a Cloud approach (see Figure

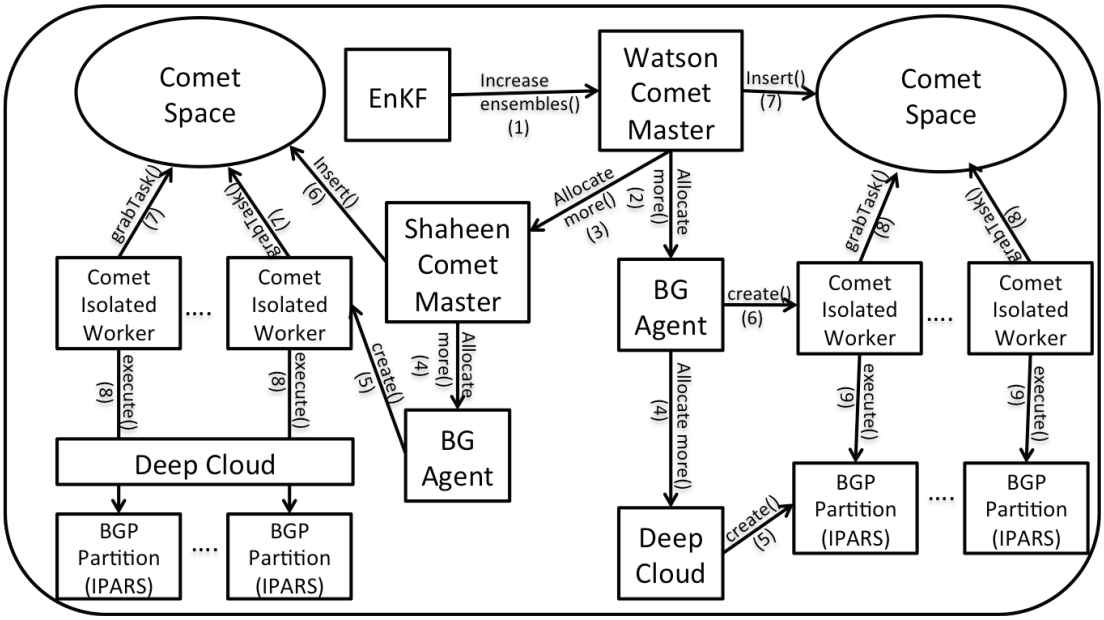


(a) Step 1

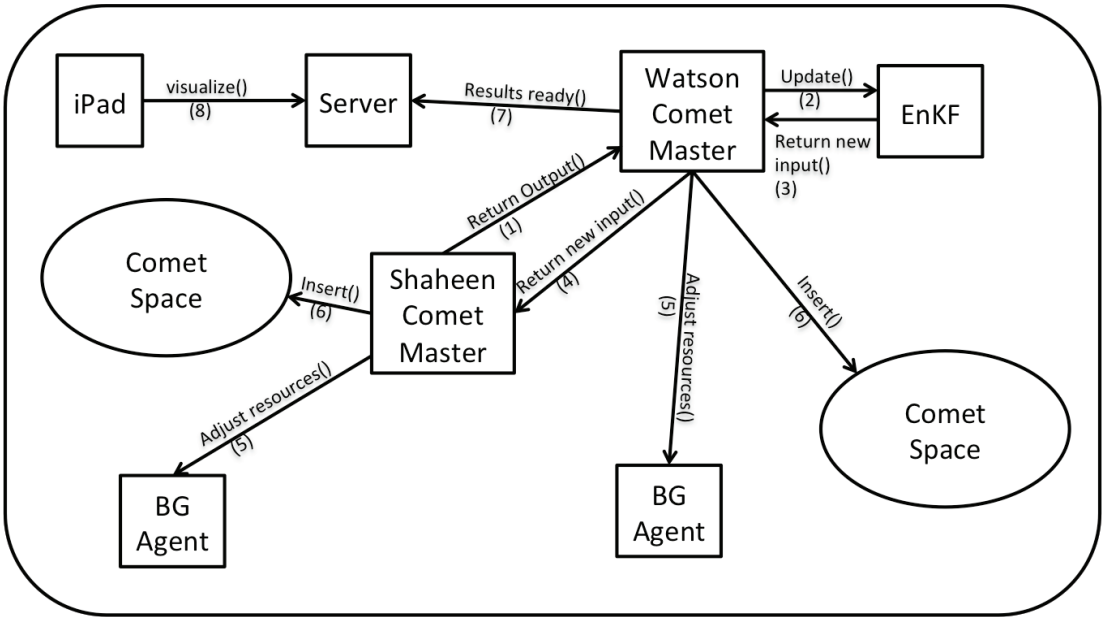


(b) Step 2

Figure 5.4: Flowchart describing steps in Framework



(c) Step 3



(d) Step 3 continued

Figure 5.4: Flowchart describing steps in Framework cont'd

5.5). Consequently, our proposed framework will be compared to other approaches for providing HPC as a Cloud. However, in this section, we briefly discuss the performance of the alternative approaches, i.e. HPC in the Cloud and HPC plus Cloud.

5.3.1 *HPC in the Cloud*

Extensive literature has already evaluated the performance of HPC in the Cloud (see section (HPC in the Cloud related work)). In summary, the performance of commodity Clouds for running tightly coupled CDS&E applications turned out to be very low.

5.3.2 *HPC plus Cloud*

While this approach provides better performance by integrating a mix of HPC and Cloud resources (see section (HPC plus Cloud related work). This approach is oriented towards hybrid workflows (e.g., CDS&E & Analytics applications). Therefore, the performance of this approach is irrelevant to our evaluation. Nonetheless, we should point out that while this approach provides some advantages over using HPC resources only (e.g. resilience or acceleration), this approach does not address some of the challenges of the HPC model such as ease of use or instant on-demand access, which is addressed by our approach.

5.3.3 *HPC as a Cloud*

Virtualized Solutions

- Azure Scheduler: does not provide HPC as a Cloud, but an auxiliary scheduler to clusters running HPC server 2008 instead.

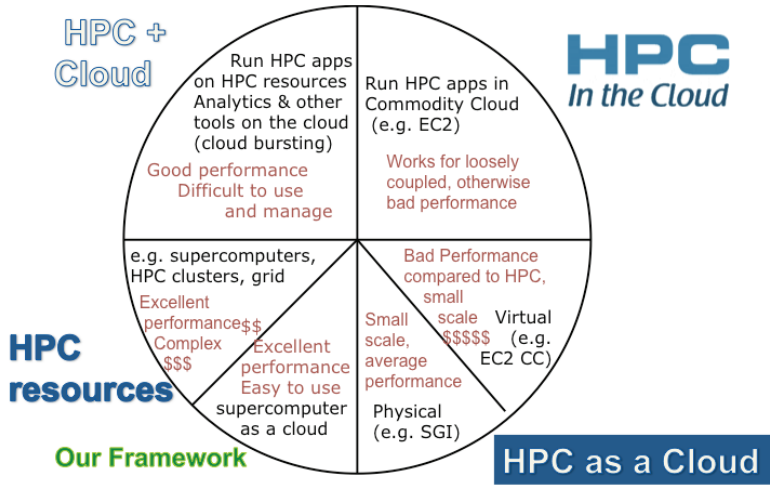


Figure 5.5: Landscape of Current HPC and Cloud Approaches

- Adaptive Systems (Moab Adaptive HPC Suite): limited QoS, low performance
- CloudCycle: resells Amazon EC2 Compute Cluster
- Amazon EC2 Compute Cluster:

  Scale: The Amazon EC2 Compute Cluster can scale up to 128 nodes

  Performance: Zhai, Y. et. al compared the performance of the Amazon EC2CC to traditional HPC Clusters and concluded that Amazon's performance is less than that of local HPC clusters of similar configurations [48]

Physical Solutions

- Silicon Graphics Cyclone: hardware specifications are extremely vague and no performance guarantees
- Penguin Computing on Demand (POD): provides remotely accessible cluster not a cloud

5.4 Evaluation Criteria

The presented scenario and experiment were used to test and validate that high performance computing can be provided as a Service. The proposed HPC as a Service framework combines the high performance of HPC resources with the flexibility and advantages of Cloud computing. A list of evaluation measurements are listed below. This list follows the analysis from Table 3.2: Application of Cloud Concepts to HPC Resources. Each of these measurements is discussed in more details, accompanied by experimental results, qualitative or quantitative analysis, and comparison to existing efforts.

- **High Performance Computing Measurements**

  Hardware Performance

  Scale

- **Cloud Computing Measurements**

  Cost associativity

  Ease of use

  Efficient Resource Utilization & Consolidation

  Elasticity

  Eliminate startup costs

  Fault Tolerance & Resilience

  Infrastructure, Platform and Software as a Service

  Instant/On-Demand access

5.5 Evaluation

In this section, we will evaluate the proposed framework and compare our results to existing HPC as a Cloud approaches.

5.5.1 High Performance Computing Measurements

One of the main goals of the framework is to provide the Cloud abstraction without sacrificing the high performance required by CDS&E applications, thus, our first evaluation criteria was to measure such performance and compare it to current solutions. We used two performance measures: hardware performance and scalability.

- *Hardware Performance*: Measured by comparing the hardware for Blue Gene/P and current offerings of HPC as a Cloud. The main factors of such comparison are: speed of processors and interconnects.
- *Scalability*: Realistic CDS&E applications require extremely large scale; therefore, the underlying hardware must be of large scale to accommodate such applications.

Performance Evaluation of HPC as a Cloud Approaches Table 5.1 summarizes the different approaches for providing HPC as a Cloud (including ours).

Table 5.1: Approaches for Providing HPC as a Cloud

HPC as a Cloud Resource	**Virtualized**	**Performance**	**Scale**
Our Proposed Framework	No	High Performance	22k
Amazon EC2CC	Yes	Medium Performance	128
Microsoft Azure	Yes	N/A	N/A
Adaptive Solutions	Yes	Low Performance	N/A
HPCynergy	No	High Performance	384
Silicon Graphics Cyclone	No	Medium Performance	N/A
Penguin on Demand	No	Medium Performance	N/A

5.5.2 Cloud Computing Measurements

- Cost associativity: Conceptually speaking, the pay per use model on top of Blue Gene/P provides cost associativity (i.e. 1 BG/P partition for 10

hours is similar to 10 partitions for 1 hour). However, realistically, due to the scarcity of the resources and the dynamic pricing model, we cannot guarantee cost associativity.

- Ease of use: Clouds provide an easy to use abstraction; therefore, in our approach we have ensured such requirements in all three levels. For example, users can programmatically allocate Blue Gene/P resources using Deep Cloud. Moreover, users can develop and deploy applications easily using CometCloud. Finally users, can run, monitor and interact with the system using thin clients and the DISCOVER portal. All of these layers mask the complexity of the systems by outsourcing the mundane tasks of system administration.

- Efficient Resource Utilization & Consolidation: The framework ensures efficient utilization of resources, by allocating and deallocating resources based on the workflow requirement.

- Elasticity: The framework can scale up/down/out based on user objectives and policies, as well as dynamic workflow requirements. In our experiment, we demonstrated such concept by scaling up to gain better around time, then scale up and out to meet the application requirements. Eliminate Startup Costs: Providing HPC as a Service democratizes access to high-end HPC resources, allowing small and mid-sized labs and research organization to take advantage of such systems.

- Fault Tolerance & Resilience: The framework provides complete fault tolerance, and guarantees the execution of the workflow. In case of a hardware failure, the framework is intelligent enough to move the corresponding work to a different partition without pausing or restarting the workflow, providing a much-needed layer of resilience. Infrastructure, Platform, and Software

as a Service: The framework provides the three layers of Cloud computing. The framework allows the decoupling the science from the underlying infrastructure (at the Platform as a Service layer). As a result, CDS&E application can now be driven by science, instead of resource availability. Furthermore, providing HPC Software as a Service opens new opportunities for CDS&E applications (e.g. better research collaboration, in field simulations to improve decision making, data and result sharing)

- Instant/On-demand access: While instant access cannot be fully guaranteed, i.e. due to the scarcity of high-end HPC resources, providing a reservation based model is a better model that ensures effective resource utilization, and increases the user satisfaction by eliminating the uncertainty of job execution (e.g. compared to a batch queue system). For example, a researcher knows exactly when their job will run on such systems, and thus they would be able to plan their day better. In addition, if the researcher is time constrained, they can pay more to get faster turnaround time, while those researcher on a constrained budget, can run their jobs, at idle times, when the resources are cheaper to use.

5.5.3 Experimental Results

This section explains the computational results from the ensemble application as well as some metrics to measure the overhead of the proposed framework. We first present a small example using the parallel reservoir simulator IPARS. This test case models two-phase flow (oil and water) in a two-dimensional horizontal reservoir with a 20x30 uniform grid. The parameters that we estimate are permeability and porosity fields with true values and well locations as shown in Figure 5.6.

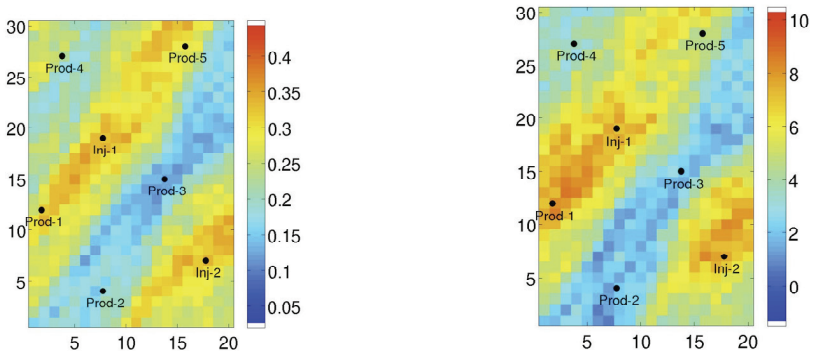


Figure 5.6: Permeability and Porosity Fields

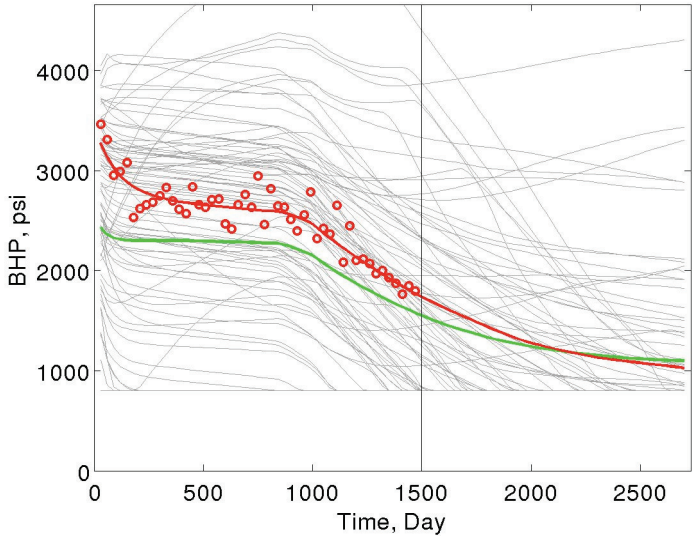


Figure 5.7: Prior Uncertainties of the Predicted Data

There are two water injection wells located in the middle of high permeability and porosity regions with constant water injection rate, and five producers constrained by total production rates.

The observation data to be assimilated in this example include oil production rate (qoil), flowing bottom hole pressure (BHP), and water-oil ratio (WOR) recorded every 30 days. The historical data are available for 1500 days. The prior uncertainties of the predicted data at three wells compared to the observed data are shown in Figure 5.7.

These results were obtained by running the reservoir simulator forward up to

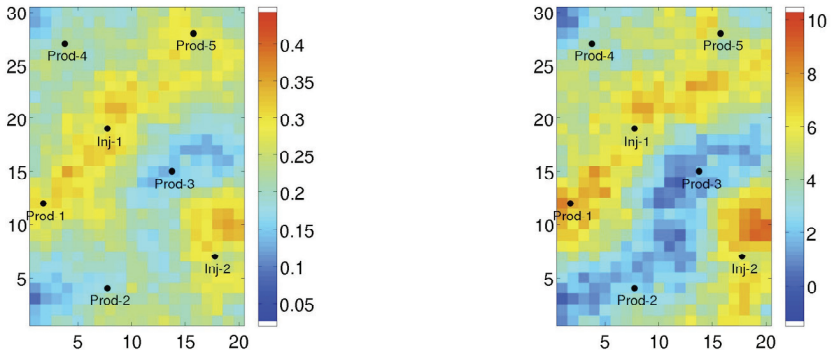


Figure 5.8: Estimated Permeability and Porosity Fields

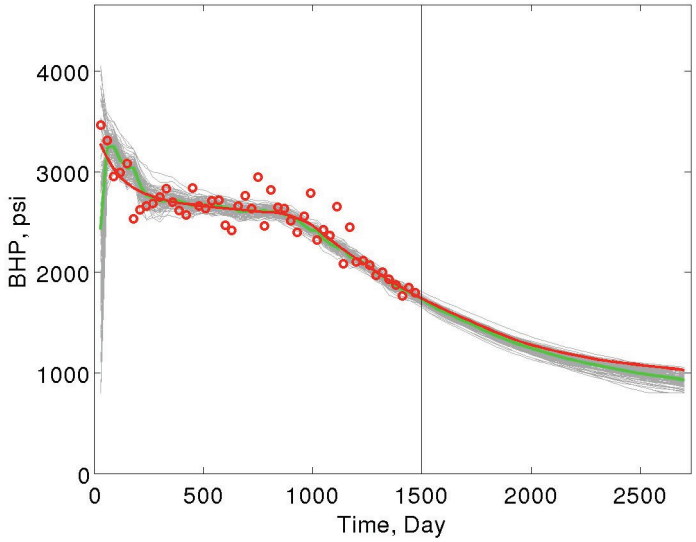


Figure 5.9: BHP Data during Data Assimilation and Future Forecast

end time with 100 initial ensemble members. As can be seen from Figure 5.7, there is a large uncertainty in the ensemble predictions represented by the wide spread of the ensemble members around the truth (red curve).

Figure 5.8 shows the estimated parameters (permeability and porosity fields) after assimilating all the observation data. Note that we have captured much of the essential features of the truth. The BHP data matches during data assimilation and future forecast for one of the producers is shown in Figure 5.9.

The vertical line denotes the end of the data assimilation period. This data suggests that a good BHP match is obtained and the variances of the prediction

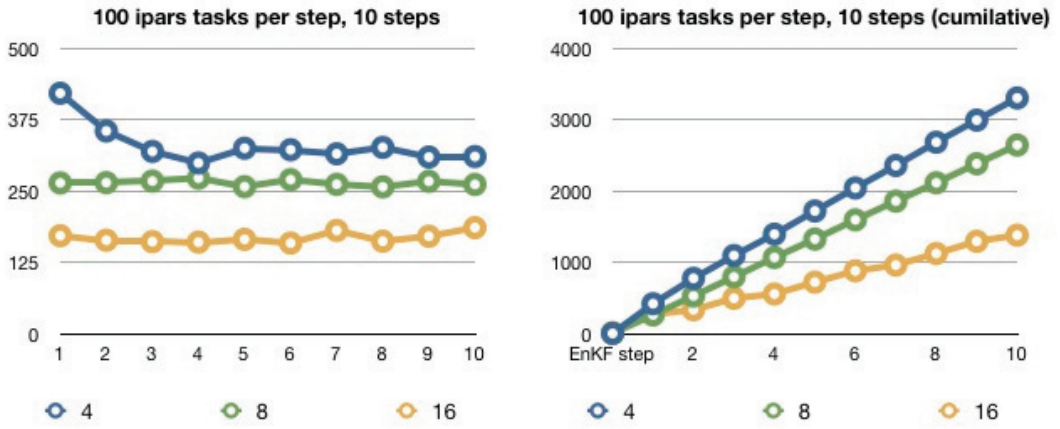


Figure 5.10: Average and Cumulative Time over 10 EnKF Steps using 100 IPARS tasks

are reduced compared to those obtained from initial ensemble members (Figure 5.7).

In addition to the results provided by the oil-reservoir application, we performed some performance evaluation of the framework to measure the overhead of running the application when using the proposed framework. Figure 5.10 shows the detailed results of this analysis.

Average time to allocate a Blue Gene/P partition (including boot and start up) is 102.8 seconds.

To measure the turnaround time, we ran the same experiment, using 100 ensemble members and 10 EnKF steps, while doubling the number of allocated partitions (4, 8, and 16 partitions).

Using 100 ensemble members (realizations) and 10 EnKF steps, the average time (sec) and cumulative (sec) per ensemble member is shown in 5.10.

Using 100 empty tasks (10 secs each) and 10 steps, the average time (sec) and cumulative (sec) per step is shown in 5.11.

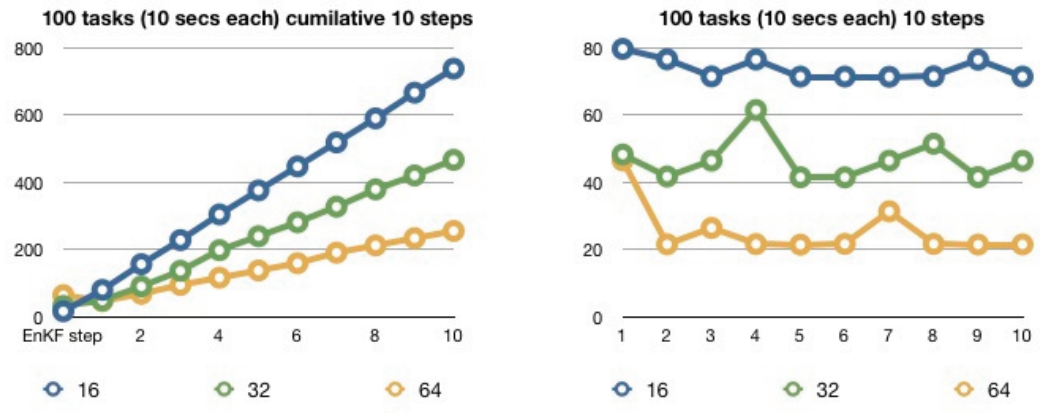


Figure 5.11: Average and Cumulative Time over 10 EnKF Steps using 10 second tasks

5.6 Evaluation Summary

In this experiment, we explored how a Cloud abstraction can be effectively used to provide a simple interface for current HPC resources and support real-world applications. In particular, we experimentally validated the benefits of the Cloud paradigm, such as ease of use and dynamic allocation, and their application to supercomputers using IBM Blue Gene/P. The CometCloud-based framework essentially transformed Blue Gene/P into an elastic Cloud, bridged multiple Blue Gene/P systems to create a larger HPC federated Cloud, and supported dynamic provisioning. We used the framework for an oil-reservoir data assimilation and history matching application, which consisted of the EnKF workflow with multiple reservoir instances. The exercise demonstrated the ease-of-use of the elastic as-a-service Cloud abstraction, and its effectiveness in improving utilization. This experiment was demonstrated at the 4th IEEE SCALE Challenge [11], and was awarded the first place. During the experiment, Blue Gene/P resources varied from 640 to 22,016 processors, spanning across two Blue Gene systems in two different continents. The objective of the challenge was to highlight and showcase a real-world problem using computing that scales. The contest focused on end-to-end problem solving using concepts, technologies and architectures that

facilitate scaling. The framework can support a broad range of applications, as the coupling between the CDS&E applications and the framework is loose. In addition, the ease of integrating new applications makes scientists and researchers more inclined to use this framework.

Chapter 6
Conclusion

6.1 Thesis Summary

Clouds are becoming an important part of production computational environments and are joining high-performance computing systems, clusters, and Grids as viable platforms for supporting Computational and Data-enabled Science and Engineering. The goal of this document was to explore how a cloud abstraction could be applied to provide an interface for current HPC resources and support real-world science and engineering applications. Moreover, this document discussed the design, implementation, and validation of such an abstraction. The proposed framework transformed the Blue Gene/P supercomputer into a federated elastic cloud. During experimental evaluation, the Blue Gene/P resources varied from 640 to 22,016 processors, spanning across two Blue Gene systems in two different continents.

The previous research and the research presented in this document help to show that there are real benefits in using Clouds and Cloud computing abstractions to support CDS&E in order to simplify the deployment of applications and the management of their execution, improve their efficiency, effectiveness and/or productivity, and provide more attractive cost/performance ratios, thus, allowing scientists and engineers to spend their time more effectively (see Figure 6.1). Furthermore, Clouds and Cloud computing abstractions can support new classes of algorithms and enable new applications formulations, which can potentially revolutionize CDS&E research and education.

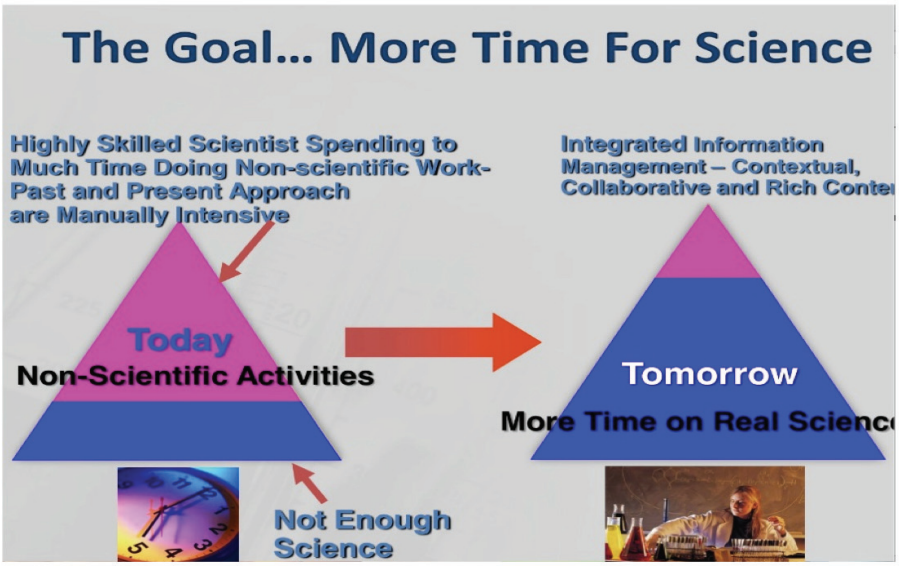


Figure 6.1: The Future of Scientific Computing [25]

6.2 Future Work for the Proposed Framework

The proposed framework was an initial proof of concept. More work is needed to make the framework more robust and secure, to allow for other HPC systems to join the federated cloud, to handle communication among the different clouds more effectively, and to generalize the solution to adapt to other HPC workflows. These goals are discussed in more details below.

6.2.1 Provisioning other HPC Resources

It is necessary to investigate how to provision other HPC resources other than the IBM Blue Gene/P such as Open Science Grid. One of the challenges in doing so is to understand how to provide on-demand access to such resources. On Blue Gene/P, Deep Cloud was utilized, which is not available for other computing resources. This may require more research in order to integrate such systems into the federated HPC as a Cloud.

6.2.2 Communication across different clouds

Communication among each cloud is optimized due to the fast interconnects within the system. However, bridging multiple systems that are geographically distributed poses a problem for data communication. An initial approach to such problem is to distribute the tasks among the different clouds in a way that would require minimum or no communication among the system, aside from synchronization between the workflow steps. However, more work is needed to solve the problem in general and to optimize the communication between the different clouds.

6.2.3 Generalize the solution

As mentioned earlier, the presented framework was a proof of concept. More work is needed to generalize the problem and to accommodate other HPC workflows. An initial approach to such a challenge is to start by generalizing the framework to support any EnKF workflow, thus, providing EnKF as a service. Once the workflow is generalized for ensemble workflow, more work will be needed to adapt the solution to other HPC workflows.

6.2.4 Providing a complete stack

We have successfully provided an Infrastructure as a Service for HPC resources, e.g. providing scientists with the state-of-the-art HPC resources, which can be accessed in a programmatic manner. This framework can be extended by providing HPC at the Platform as a Service level – for example, by providing EnKF as a service. Even more work is needed to generalize the top layer, Software as a Service for HPC. HPC in the form of SaaS will not only allow scientists to easily run scientific simulations, but it will also enable field engineers, doctors, and many others to take advantage of these applications as well. Using a thin

client that they can carry anywhere, these users can run workflows and what-if scenarios in order to obtain insightful information that can help them, in the field or with a diagnosis at the bedside. The HPC SaaS delivers the power of a supercomputer, and a scientific application to the palm of the many users in different fields. In addition, with the growing number of domains that are adopting HPC applications in order to understand the complex processes of their domain (e.g. Aerospace, Automobile, Business Analytics, Entertainment, Finance, Manufacturing, Oil & Gas, Pharmaceuticals, etc.), we believe that providing HPC as a Service will transform science: the way it is being developed and the way it is being used. HPC as a Service will even allow for more collaboration among scientists in geographically distributed locations, with centralized data and application that are running on the Cloud.

6.3 Research Agenda for CDS&E and Clouds

6.3.1 Algorithms and Application Formulations for Clouds

A key attribute of Clouds is on-demand access to elastic resources i.e., applications programmatically access more or less resources as they evolve, to meet changing needs. Such a capability can have a significant impact on how algorithms are developed and applications are formulated. For example, the execution of an application no longer has to constrain itself to a fixed set of resources that are available at runtime and can grow or shrink its resource set based on the demands of the science the science can drive the scale and type of resource involved, based on the levels of refinement required to resolve a solution feature, or the number of ensembles that need to be run to quantify the uncertainty in a solution, or the type of online analytics services that need to be dynamically composed into the application workflow. In addition to the usage modes and application scenarios discussed earlier in this documents, understanding how CDS&E applications can

effectively utilize Clouds and Cloud abstractions as part of the a hybrid cyber-infrastructure, to enable new practices and levels of scientific insights remains a research challenge. Discovering what application formulations and usage modes are meaningful in a hybrid and federated Cloud infrastructure is essential. Finally, developing application formulations that are agnostic to the underlying hardware but can exploit on-demand federated hybrid resources without compromising performance is also critical.

6.3.2 Programming Systems and Abstractions

One of the key research challenges is developing appropriate programming abstractions and language extensions that can enable CDS&E application to simply and effectively take advantage of the elastic access to resources and services during application formulation. Furthermore, it may be necessary to define constraints (for example, budgets, data privacy, performance, etc.) to regulate the elasticity, and the programming abstractions my provide support for expressing these constraints so that they can be enforced during execution. Similarly, such annotations can also define possible adaptations, which could then be used to increase performance, manageability and overall robustness of the application. For example, dynamically increase the assigned resources in order to increase the resolution of a simulation under certain convergence constraints, modify convergence goals to avoid failure or guarantee completion time. The Cloud service models can also lead to interesting services specialized to CDS&E that provide entire applications or applications kernels as a service (i.e., SaaS). Furthermore, and arguably more interestingly, it can also export specialized platforms for science as a services, which encapsulate elasticity and abstract of the underlying hybrid cyber infrastructure. In return, the scientists are only required to provide, core kernels, meaningful parameters, and basic configurations. For example, the proposed research is exploring the idea of EnKF-as-service.

6.3.3 Middleware stacks, management policies, economic models

Middleware services will need to support the new CDS&E applications formulations and services enabled. A key research aspect will be the autonomic management and optimization of application execution through cross-layer application/infrastructure adaptations. It will be essential for the middleware services 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. Furthermore, appropriate services are necessary to be able to provision different types of resources on demand. For example, in order to integrate NSF funded cyberinfrastructure such as XSEDE, Open Science Grid and FutureGrid along with commercial Clouds such as Amazon EC2 or Microsoft Azure, autonomic provisioning and scheduling techniques, including Cloud bursting will be necessary to support these usage modes. Finally, monitoring, online data analytics for proactive application/resource management and adaptation techniques will be essential as the scale and complexity of both the applications and hybrid infrastructure grows.

Acknowledgment of Previous Publications

Portions of this thesis were inspired by an article previously published by the author. The full citation is as follows.

M. AbdelBaky, H. Kim, M. Parashar, K.E. Jordan, H. Jamjoom, V. Sachdeva, Z.Y. Shae, J. Sexton, G. Pencheva, R. Tavakoli, and M.F. Wheeler, "Scalable Ensemble based Oil Reservoir Simulations using Blue Gene/P as-a-Service," White paper, SCALE 2011, the 4th IEEE International Scalable Computing Challenge was held on May 26th 2011 in conjunction with the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing at Newport Beach CA, USA.

The following non-published work of the author was also used to contribute to parts of this thesis.

M. AbdelBaky, H. Kim, M. Parashar, K.E. Jordan, H. Jamjoom, V. Sachdeva, Z.Y. Shae, G. Pencheva, R. Tavakoli, and M.F. Wheeler, "Scalable Ensemble based Oil Reservoir Simulations using HPC as a Cloud."

M. AbdelBaky, I. Rodero, M. Parashar, "Cloud Paradigms and Practices for CDS&E."

References

[1] Amazon elastic compute cloud (amazon ec2). http://aws.amazon.com/ec2/.

[2] Apple ipad. http://www.apple.com/ipad/.

[3] Cometcloud. http://cometcloud.org.

[4] The economics of the cloud. http://economics.uchicago.edu/pdf/Harms_110111.pdf.

[5] Evaluating cloud computing for hpc applications, nersc brown bag, october 2010. http://www.nersc.gov/assets/Events/MagellanNERSCLunchTalk.pdf.

[6] Forecast: Public cloud services, worldwide and regions, industry sectors, 2009- 2014, gartner research, 2009-2014. http://www.gartner.com/DisplayDocument?id=1378513.

[7] Google app engine. https://appengine.google.com.

[8] Google mail (gmail). https://mail.google.com/mail/x/.

[9] Ibm research doubles its productivity with cloud computing, case study. http://www-935.ibm.com/services/in/cio/pdf/oic03013usen.pdf.

[10] National science foundation, cyber infrastructure framework for 21st century science and engineering (cif21). http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=504730.

[11] Scale 2011, the 4th ieee international scalable computing challenge, the 11th ieee/acm international symposium on cluster, cloud and grid computing at newport beach ca, usa, may 2012. http://www.ics.uci.edu/~ccgrid11/SCALE\%20Challenge.html.

[12] S. Aanonsen, G. Nvdal, D. Oliver, A. Reynolds, and B. Valls. The ensemble kalman filter in reservoir engineering–a review. *SPE Journal*, 14(3):393–412, 2009.

[13] G. Andrews. Drowning in the data deluge.

[14] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[15] I. Barcena, J. Becerra, C. Fernández, et al. A grid supercomputing environment for high demand computational applications. *Boletín de RedIRIS*, (66-67), 2003.

[16] N. Bobroff, L. Fong, S. Kalayci, Y. Liu, J. Martinez, I. Rodero, S. Sadjadi, and D. Villegas. Enabling interoperability among meta-schedulers. In *Cluster Computing and the Grid, 2008. CCGRID'08. 8th IEEE International Symposium on*, pages 306–315. IEEE, 2008.

[17] M. De Assunção, A. Di Costanzo, and R. Buyya. Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters. In *Proceedings of the 18th ACM international symposium on High performance distributed computing*, pages 141–150. ACM, 2009.

[18] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good. The cost of doing science on the cloud: the montage example. In *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, pages 1–12. Ieee, 2008.

[19] M. Dias de Assunção, R. Buyya, and S. Venugopal. Intergrid: A case for internetworking islands of grids. *Concurrency and Computation: Practice and Experience*, 20(8):997–1024, 2008.

[20] K. Dowd. *High performance computing*. O'Reilly & Associates, Inc., 1993.

[21] C. Evangelinos and C. Hill. Cloud computing for parallel scientific hpc applications: Feasibility of running coupled atmosphere-ocean climate models on amazons ec2. *ratio*, 2(2.40):2–34, 2008.

[22] G. Evensen. The ensemble kalman filter: Theoretical formulation and practical implementation. *Ocean dynamics*, 53(4):343–367, 2003.

[23] G. Fox. Data intensive applications on clouds. In *Proceedings of the second international workshop on Data intensive computing in the clouds*, pages 1–2. ACM, 2011.

[24] G. Fox and D. Gannon. Cloud programming paradigms for technical computing applications. Technical report, Indiana University, Microsoft, 2011.

[25] F. Gagliardi. Hpc opportunities and challenges in e-science. *Lecture Notes in Computer Science*, 5101:18–19, 2008.

[26] A. Goscinski, M. Brock, and P. Church. High performance computing clouds. In *Cloud Computing: Methodology, Systems, and Applications*, chapter 11, pages 221–260. CRC Press, 2011.

[27] R. Grossman. The case for cloud computing. *IT professional*, 11(2):23–27, 2009.

[28] A. Iosup, S. Ostermann, M. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema. Performance analysis of cloud computing services for many-tasks scientific computing. *Parallel and Distributed Systems, IEEE Transactions on*, 22(6):931–945, 2011.

[29] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B. Berman, and P. Maechling. Scientific workflow applications on amazon ec2. In *E-Science Workshops, 2009 5th IEEE International Conference on*, pages 59–66. Ieee, 2009.

[30] Y. Kang and G. Fox. Performance evaluation of mapreduce applications on cloud computing environment, futuregrid. *Grid and Distributed Computing*, pages 77–86, 2011.

[31] K. Keahey, R. Figueiredo, J. Fortes, T. Freeman, and M. Tsugawa. Science clouds: Early experiences in cloud computing for scientific applications. *Cloud computing and applications*, 2008, 2008.

[32] H. Kim, S. Chaudhari, M. Parashar, and C. Marty. Online risk analytics on the cloud. In *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*, pages 484–489. IEEE, 2009.

[33] H. Kim, Y. el Khamra, S. Jha, and M. Parashar. An autonomic approach to integrated hpc grid and cloud usage. In *e-Science, 2009. e-Science'09. Fifth IEEE International Conference on*, pages 366–373. IEEE, 2009.

[34] H. Kim and M. Parashar. Cometcloud: An autonomic cloud engine. In *Cloud Computing: Principles and Paradigms*, chapter 10, pages 275–297. Wiley Online Library, 2011.

[35] V. Mann, V. Matossian, R. Muralidhar, and M. Parashar. Discover: An environment for web-based interaction and steering of high-performance scientific applications. *Concurrency and Computation: Practice and Experience*, 13(8-9):737–754, 2001.

[36] S. Ostermann, R. Prodan, and T. Fahringer. Extending grids with cloud resource management for scientific computing. In *Grid Computing, 2009 10th IEEE/ACM International Conference on*, pages 42–49. Ieee, 2009.

[37] I. Raicu, I. Foster, and Y. Zhao. Many-task computing for grids and supercomputers. In *Many-Task Computing on Grids and Supercomputers, 2008. MTAGS 2008. Workshop on*, pages 1–11. Ieee, 2008.

[38] I. Raicu, Z. Zhang, M. Wilde, I. Foster, P. Beckman, K. Iskra, and B. Clifford. Toward loosely coupled programming on petascale systems. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, page 22. IEEE Press, 2008.

[39] Z. Shae, H. Jamjoom, H. Qu, M. Podlaseck, Y. Ruan, and A. Sheopuri. On the design of a deep computing service cloud. *IBM Research Report*, 2010.

[40] I. B. G. Team. Overview of the ibm blue gene/p project. *IBM Journal of Research and Development*, 52(1/2):199–220, 2008.

[41] C. Vázquez, E. Huedo, R. Montero, and I. Llorente. Dynamic provision of computing resources from grid infrastructures and cloud providers. In *Grid and Pervasive Computing Conference, 2009. GPC'09. Workshops at the*, pages 113–120. IEEE, 2009.

[42] L. Wang, J. Tao, M. Kunze, A. Castellanos, D. Kramer, and W. Karl. Scientific cloud computing: Early definition and experience. In *High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on*, pages 825–830. Ieee, 2008.

[43] J. Wheeler. Ipars simulator framework users guide, 1998.

[44] M. Wheeler. Advanced techniques and algorithms for reservoir simulation ii: the multiblock approach in the integrated parallel accurate reservoir simulator (ipars). *Institute for Mathematics and Its Applications*, 131:9, 2002.

[45] M. Wilde, I. Raicu, A. Espinosa, Z. Zhang, B. Clifford, M. Hategan, S. Kenny, K. Iskra, P. Beckman, and I. Foster. Extreme-scale scripting: Opportunities

for large task-parallel applications on petascale computers. In *Journal of Physics: Conference Series*, volume 180, page 012046. IOP Publishing, 2009.

[46] K. Yelick, S. Coghlan, R. S. Draney, B. Canon, and et al. The magellan report on cloud computing for science. U.S. Department of Energy Office of Advanced Scientific Computing Research (ASCR), December 2011.

[47] A. Younge, R. Henschel, J. Brown, G. von Laszewski, J. Qiu, and G. Fox. Analysis of virtualization technologies for high performance computing environments. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 9–16. IEEE, 2011.

[48] Y. Zhai, M. Liu, J. Zhai, X. Ma, and W. Chen. Cloud versus in-house cluster: evaluating amazon cluster compute instances for running mpi applications. In *State of the Practice Reports*, page 11. ACM, 2011.

[49] L. Zhang, M. Parashar, E. Gallicchio, and R. Levy. Salsa: Scalable asynchronous replica exchange for parallel molecular dynamics applications. In *Parallel Processing, 2006. ICPP 2006. International Conference on*, pages 127–134. IEEE, 2006.

Bibliography

[1] G. Alosio, M. Cafaro, C. Kesselman, and R. Williams. Web access to supercomputing. *Computing in Science & Engineering*, 3(6):66–72, 2001.

[2] J. Appavoo, V. Uhlig, and A. Waterland. Project kittyhawk: building a global-scale computer: Blue gene/p as a generic computing platform. *ACM SIGOPS Operating Systems Review*, 42(1):77–84, 2008.

[3] J. Appavoo, V. Uhlig, A. Waterland, B. Rosenburg, D. Da Silva, and J. Moreira. Kittyhawk: Enabling cooperation and competition in a global, shared computational system. *IBM Journal of Research and Development*, 53(4):9–1, 2009.

[4] J. Appavoo, A. Waterland, D. Da Silva, V. Uhlig, B. Rosenburg, E. Van Hensbergen, J. Stoess, R. Wisniewski, and U. Steinberg. Providing a cloud network infrastructure on a supercomputer. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pages 385–394. ACM, 2010.

[5] S. Benkner, I. Brandic, G. Engelbrecht, and R. Schmidt. Vge-a service-oriented grid environment for on-demand supercomputing. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, pages 11–18. IEEE Computer Society, 2004.

[6] P. Bientinesi, R. Iakymchuk, and J. Napper. Hpc on competitive cloud resources. *Handbook of Cloud Computing*, pages 493–516, 2010.

[7] J. Brandt, A. Gentile, J. Mayo, P. Pebay, D. Roe, D. Thompson, and M. Wong. Resource monitoring and management with ovis to enable hpc in cloud computing environments. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–8. IEEE, 2009.

[8] M. Brock and A. Goscinski. Enhancing cloud computing environments using a cluster as a service. *Cloud Computing*, pages 193–220.

[9] M. Brock and A. Goscinski. Attributed publication and selection for web service-based distributed systems. In *Services-I, 2009 World Conference on*, pages 732–739. IEEE, 2009.

[10] M. Brock and A. Goscinski. Dynamic wsdl for supporting autonomic computing. *Autonomic Computing and Networking*, pages 105–130, 2009.

[11] M. Brock and A. Goscinski. Offering clusters from clouds using wsdl and stateful web services. In *Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific*, pages 233–238. IEEE, 2009.

[12] M. Brock and A. Goscinski. Grids vs. clouds. In *Future Information Technology (FutureTech), 2010 5th International Conference on*, pages 1–6. IEEE, 2010.

[13] M. Brock and A. Goscinski. A technology to expose a cluster as a service in a cloud. In *Proceedings of the Eighth Australasian Symposium on Parallel and Distributed Computing-Volume 107*, pages 3–12. Australian Computer Society, Inc., 2010.

[14] M. Brock and A. Goscinski. Toward a framework for cloud security. *Algorithms and architectures for parallel processing*, pages 254–263, 2010.

[15] M. Brock and A. Goscinski. Toward ease of discovery, selection and use

of clusters within a cloud. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 289–296. IEEE, 2010.

[16] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing. In *High Performance Distributed Computing, 2001. Proceedings. 10th IEEE International Symposium on*, pages 181–194. IEEE, 2001.

[17] T. Dornemann, E. Juhnke, and B. Freisleben. On-demand resource provisioning for bpel workflows using amazon's elastic compute cloud. In *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*, pages 140–147. Ieee, 2009.

[18] J. Ekanayake and G. Fox. High performance parallel computing with clouds and cloud technologies. *Cloud Computing*, pages 20–38, 2010.

[19] Y. El-Khamra, H. Kim, S. Jha, and M. Parashar. Exploring the performance fluctuations of hpc workloads on clouds. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 383–387. IEEE, 2010.

[20] I. Foster, Y. Zhao, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE'08*, pages 1–10. Ieee, 2008.

[21] T. Freeman and K. Keahey. Flying low: Simple leases with workspace pilot. *Euro-Par 2008–Parallel Processing*, pages 499–509, 2008.

[22] A. Goscinski and M. Brock. Toward dynamic and attribute based publication, discovery and selection for cloud computing. *Future generation computer systems*, 26(7):947–970, 2010.

[23] A. Goscinski and M. Brock. Toward higher level abstractions for cloud computing. *International Journal of Cloud Computing*, 1(1):37–57, 2011.

[24] T. Gunarathne, B. Zhang, T. Wu, and J. Qiu. Portable parallel programming on cloud and hpc: Scientific applications of twister4azure. In *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, pages 97–104. IEEE, 2011.

[25] H. Kim, Y. el Khamra, S. Jha, and M. Parashar. Exploring application and infrastructure adaptation on hybrid grid-cloud infrastructure. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pages 402–412. ACM, 2010.

[26] H. Kim, Y. El-Khamra, I. Rodero, S. Jha, and M. Parashar. Autonomic management of application workflows on hybrid computing infrastructure. *Scientific Programming*, 19(2):75–89, 2011.

[27] H. Kim, M. Parashar, D. Foran, and L. Yang. Investigating the use of autonomic cloudbursts for high-throughput medical image registration. In *Grid Computing, 2009 10th IEEE/ACM International Conference on*, pages 34–41. IEEE, 2009.

[28] E. Mancini, M. Rak, and U. Villano. Perfcloud: Grid services for performance-oriented development of cloud computing applications. In *Enabling Technologies: Infrastructures for Collaborative Enterprises, 2009. WETICE'09. 18th IEEE International Workshops on*, pages 201–206. IEEE, 2009.

[29] P. Martinaitis, C. Patten, and A. Wendelborn. Remote interaction and scheduling aspects of cloud based streams. In *E-Science Workshops, 2009 5th IEEE International Conference on*, pages 39–47. IEEE, 2009.

[30] J. Napper and P. Bientinesi. Can cloud computing reach the top500? In *Proceedings of the combined workshops on UnConventional high performance computing workshop plus memory access workshop*, pages 17–20. ACM, 2009.

[31] L. Nie and Z. Xu. An adaptive scheduling mechanism for elastic grid computing. In *Semantics, Knowledge and Grid, 2009. SKG 2009. Fifth International Conference on*, pages 184–191. IEEE, 2009.

[32] A. Ozer and C. Ozturan. An auction based mathematical model and heuristics for resource co-allocation problem in grids and clouds. In *Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control, 2009. ICSCCW 2009. Fifth International Conference on*, pages 1–4. IEEE, 2009.

[33] S. Pallickara, M. Pierce, Q. Dong, and C. Kong. Enabling large scale scientific computations for expressed sequence tag sequencing over grid and cloud computing clusters. In *PPAM 2009 EIGHTH INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING AND APPLIED MATHEMATICS Wroclaw, Poland*, 2009.

[34] J. Rehr, F. Vila, J. Gardner, L. Svec, and M. Prange. Scientific computing in the cloud. *Computing in Science & Engineering*, 12(3):34–43, 2010.

[35] M. Siddiqui, A. Villazon, J. Hofer, and T. Fahringer. Glare: A grid activity registration, deployment and provisioning framework. In *Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference*, pages 52–52. IEEE, 2005.

[36] Y. Simmhan, C. van Ingen, G. Subramanian, and J. Li. Bridging the gap between desktop and the cloud for escience applications. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 474–481. Ieee, 2010.

[37] T. Sterling and D. Stark. A high-performance computing forecast: Partly cloudy. *Computing in Science & Engineering*, 11(4):42–49, 2009.

[38] D. Villegas, I. Rodero, A. Devarakonda, Y. Liu, N. Bobroff, L. Fong, S. Sadjadi, and M. Parashar. Leveraging cloud federation in a layered service stack model. Journal of Computer and System Sciences, Special Issue on Cloud Computing, in press 2012.

[39] M. Vouk. Cloud computing–issues, research and implementations. *Journal of Computing and Information Technology*, 16(4):235–246, 2004.

[40] J. Wheeler and M. Wheeler. Ipars users manual. Technical report, Tech Report CSM, ICES, The University of Texas at Austin, Austin, TX, 2000.