INSERT REALLY COOL PHD THESIS TITLE HERE

by

Andrew J. Younge

Submitted to the faculty of

Indiana University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

Indiana University

May 2014

INDIANA UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a dissertation submitted by

Andrew J. Younge

This dissertation has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Geoffrey C. Fox, PhD, Chair
Date	TBD
Date	TBD
Date	Committee Member C

INDIANA UNIVERSITY

As chair of the candidate's graduate committee, I have read the dissertation of Andrew J. Younge in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

Geoffrey C. Fox, PhD
Chair, Graduate Committee

Accepted for the Department

Department Chair Name, Chair
Department of Computer Science

Accepted for the College

Dean Name, Associate Dean
Golisano College of Computing and Information Sciences

ABSTRACT

INSERT REALLY COOL PHD THESIS TITLE HERE

Andrew J. Younge

Department of Computer Science

Doctor of Philosophy

TBD

ACKNOWLEDGMENTS

TBD

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

For years visionaries in computer science have predicted the advent of utility-based computing. This concept dates back to John McCarthy's vision stated at the MIT centennial celebrations in 1961.

> "If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility... The computer utility could become the basis of a new and important industry."

Only recently has the hardware and software become available to support the concept of utility computing on a large scale.

1.2 Motivation

TBD

Chapter 2

Related Research

In order to accurately depict the research presented in this article, the topics within Cloud computing are reviewed

2.1 Cloud Computing

Cloud computing is one of the most explosively expanding technologies in the computing industry today. However it is important to understand where it came from, in order to figure out where it will be heading in the future. While there is no clear cut evolutionary path to Clouds, many believe the concepts originate from two specific areas: Grid Computing and Web 2.0.

Grid computing [**?**, **?**], in its practical form, represents the concept of connecting two or more spatially and administratively diverse clusters or supercomputers together in a federating manner. The term "the Grid" was coined in the mid 1990's to represent a large distributed systems infrastructure for advanced scientific and engineering computing problems. Grids aim to enable applications to harness the full potential of resources through coordinated and controlled resource sharing by scalable

virtual organizations. While not all of these concepts carry over to the Cloud, the control, federation, and dynamic sharing of resources is conceptually the same as in the Grid. This is outlined by [**?**], as Grids and Clouds are compared at an abstract level and many concepts are remarkably similar. From a scientific perspective, the goals of Clouds and Grids are also similar. Both systems attempt to provide large amounts of computing power by leveraging a multitude of sites running diverse applications concurrently in symphony. The only significant differences between Grids and Clouds exist in the implementation details, and the reproductions of them, as outlined later in this section.

The other major component, Web 2.0, is also a relatively new concept in the history of Computer Science. The term Web 2.0 was originally coined in 1999 in a futuristic prediction by Dracy DiNucci [**?**]: "The Web we know now, which loads into a browser window in essentially static screenfulls, is only an embryo of the Web to come. The first glimmerings of Web 2.0 are beginning to appear, and we are just starting to see how that embryo might develop. The Web will be understood not as screenfuls of text and graphics but as a transport mechanism, the ether through which interactivity happens. It will [...] appear on your computer screen, [...] on your TV set [...] your car dashboard [...] your cell phone [...] hand-held game machines [...] maybe even your microwave oven." Her vision began to form, as illustrated in 2004 by the O'Riley Web 2.0 conference, and since then the term has been a pivotal buzz word among the internet. While many definitions have been provided, Web 2.0 really represents the transition from static HTML to harnessing the Internet and the Web as a platform in of itself.

Web 2.0 provides multiple levels of application services to users across the Internet. In essence, the web becomes an application suite for users. Data is outsourced to wherever it is wanted, and the users have total control over what they interact

with, and spread accordingly. This requires extensive, dynamic and scalable hosting resources for these applications. This demand provides the user-base for much of the commercial Cloud computing industry today. Web 2.0 software requires abstracted resources to be allocated and relinquished on the fly, depending on the Web's traffic and service usage at each site. Furthermore, Web 2.0 brought Web Services standards [**?**] and the Service Oriented Architecture (SOA) [**?**] which outline the interaction between users and cyberinfrastructure. In summary, Web 2.0 defined the interaction standards and user base, and Grid computing defined the underlying infrastructure capabilities.

A Cloud computing implementation typically enables users to migrate their data and computation to a remote location with minimal impact on system performance [**?**]. This provides a number of benefits which could not otherwise be realized. These benefits include:

- *Scalable* - Clouds are designed to deliver as much computing power as any user needs. While in practice the underlying infrastructure is not infinite, the cloud resources are projected to ease the developer's dependence on any specific hardware.

- *Quality of Service (QoS)* - Unlike standard data centers and advanced computing resources, a well-designed Cloud can project a much higher QoS than traditionally possible. This is due to the lack of dependence on specific hardware, so any physical machine failures can be mitigated without the prerequisite user awareness.

- *Specialized Environment* - Within a Cloud, the user can utilize customized tools and services to meet their needs. This can be to utilize the latest library, toolkit, or to support legacy code within new infrastructure.

- *Cost Effective* - Users finds only the hardware required for each project. This reduces the risk for institutions potentially want build a scalable system, thus providing greater flexibility, since the user is only paying for needed infrastructure while maintaining the option to increase services as needed in the future.

- *Simplified Interface* - Whether using a specific application, a set of tools or Web services, Clouds provide access to a potentially vast amount of computing resources in an easy and user-centric way. We have investigated such an interface within Grid systems through the use of the Cyberaide project [**?**, **?**].

Many of the features noted above define what Cloud computing can be from a user perspective. However, Cloud computing in its physical form has many different meanings and forms. Since Clouds are defined by the services they provide and not by applications, an integrated as-a-service paradigm has been defined to illustrate the various levels within a typical Cloud, as in Figure **??**.

- *Clients* - A client interacts with a Cloud through a predefined, thin layer of abstraction. This layer is responsible for communicating the user requests and displaying data returned in a way that is simple and intuitive for the user. Examples include a Web Browser or a thin client application.

- *Software-as-a-Service (SaaS)* - A framework for providing applications or software deployed on the Internet packaged as a unique service for users to consume. By doing so, the burden of running a local application directly on the client's machine is removed. Instead all the application logic and data is managed centrally and to the user through a browser or thin client. Examples include Google Docs, Facebook, or Pandora.

- *Platform-as-a-Service (PaaS)* - A framework for providing a unique computing

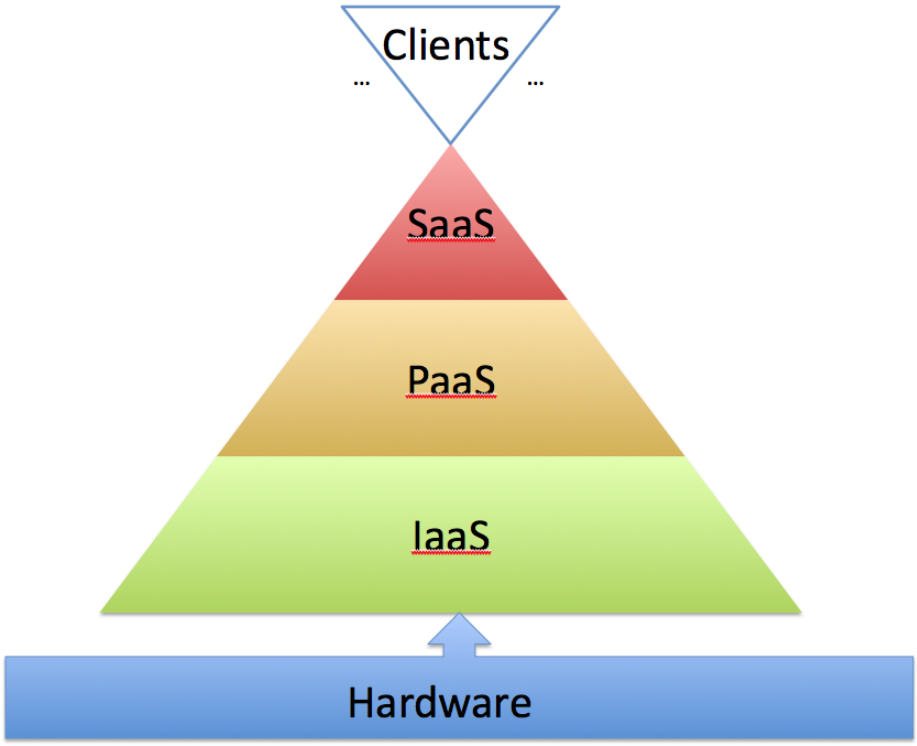

Figure 2.1 View of the Layers within a Cloud Infrastructure

platform or software stack for applications and services to be developed on. The goal of PaaS is to alleviate many of the burdens of developing complex, scalable software by proving a programming paradigm and tools that make service development and integration a tractable task for many. Examples include Microsoft Azure and Google App Engine.

- *Infrastructure-as-a-Service (IaaS)* - A framework for providing entire computing resources through a service. This typically represents virtualized Operating Systems, thereby masking the underlying complexity details of the physical infrastructure. This allows users to rent or buy computing resources on demand for their own use without needing to operate or manage physical infrastructure. Examples include Amazon EC2, Eucalyptus, and Nimbus.

- *Physical Hardware* - The underlying set of physical machines and IT equipment

that host the various levels of service. These are typically managed at a large scale using virtualization technologies which provide the QoS users expect. This is the basis for all computing infrastructure.

When all of these layers are combined, a dynamic software stack is created to focus on large scale deployment of services to users.

2.1.1 Virtualization

There are a number of underlying technologies, services, and infrastructure-level configurations that make Cloud computing possible. One of the most important technologies is the use of virtualization [**?**, **?**]. Virtualization is a way to abstract the hardware and system resources from a operating system. This is typically performed within a Cloud environment across a large set of servers using a Hypervisor or Virtual Machine Monitor (VMM) which lies in between the hardware and the Operating System (OS). From here, one or more virtualized OSs can be started concurrently as seen in Figure **??**, leading to one of the key advantages of Cloud computing. This, along with the advent of multi-core processing capabilities, allows for a consolidation of resources within any data center. It is the Cloud's job to exploit this capability to its maximum potential while still maintaining a given QoS.

Virtualization is not specific to Cloud computing. IBM originally pioneered the concept in the 1960's with the M44/44X systems. It has only recently been reintroduced for general use on x86 platforms. Today there are a number of Clouds that offer IaaS. The Amazon Elastic Compute Cloud (EC2) [**?**], is probably the most popular of which and is used extensively in the IT industry. Eucalyptus [**?**] is becoming popular in both the scientific and industry communities. It provides the same interface as EC2 and allows users to build an EC2-like cloud using their own internal resources.

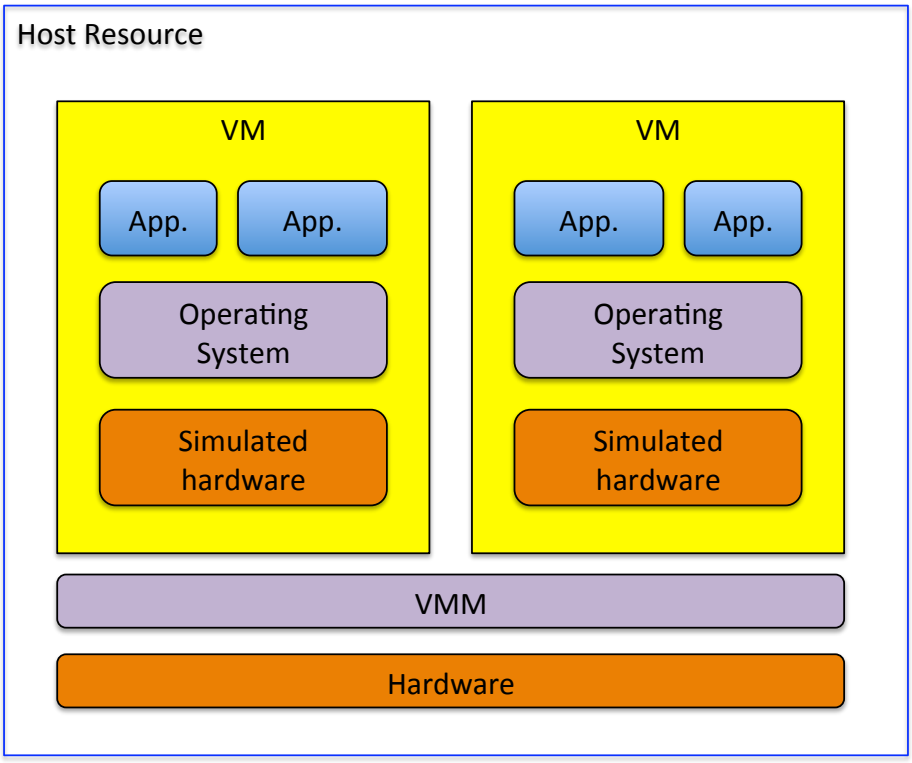

Figure 2.2 Virtual Machine Abstraction

Other scientific Cloud specific projects exist such as OpenNebula [**?**], In-VIGO [**?**], and Cluster-on-Demand [**?**]. They provide their own interpretation of private Cloud services within a data center. Using a Cloud deployment overlaid on a Grid computing system has been explored by the Nimbus project [**?**] with the Globus Toolkit [**?**]. All of these clouds leverage the power of virtualization to create an enhanced data center. The virtualization technique of choice for these Open platforms has typically been the Xen hypervisor, however more recently VMWare and the Kernel-based Virtual Machine (KVM) have become commonplace.

2.1.2 Workload Scheduling

While virtualization provides many key advancements, this technology alone is not sufficient. Rather, a collective scheduling and management for virtual machines is

required to piece together a working Cloud. Let us consider a typical usage for a Cloud data center that is used in part to provide computational power for the Large Hadron Collider at CERN [**?**], a global collaboration from more than 2000 scientists of 182 institutes in 38 nations. Such a system would have a small number of experiments to run. Each experiment would require a very large number of jobs to complete the computation needed for the analysis. Examples of such experiments are the ATLAS [**?**] and CMS [**?**] projects, which (combined) require Petaflops of computing power on a daily basis. Each job of an experiment is unique, but the application runs are often the same.

Therefore, virtual machines are deployed to execute incoming jobs. There is a file server which provides virtual machine templates. All typical jobs are preconfigured in virtual machine templates. When a job arrives at the head node of the cluster, a correspondent virtual machine is dynamically started on acertain compute node within the cluster to execute the job (see Figure BLAH).

While this is an abstract solution, it is important to keep in mind that these virtual machines create an overhead when compared to running on "bare metal." Current research estimates this the overhead for CPU bound operations at 1 to 15% depending on the hypervisor, however more detailed studies are needed to better understand this overhead. While the hypervisor introduces overhead, so does the actual VM image being used. Therefore, it is clear that slimming down the images could yield an increase in overall system efficiency. This provides the motivation for the minimal Virtual Machine image design discussed in Section **??**.

Chapter 3

Conclusion

3.1 Summary

TBD

3.2 Future Work

TBD

Appendix A

Appendix

This section provides the formal description for TBD.

A.1 AppendixA

A.1.1 Appendix A1