Open Source Cloud Computing Tools: A Case Study with a Weather Application

Melvin Greer
Senior Research Engineer, Advanced Technology Office
Lockheed Martin
melvin.greer@lmco.com

Dr. Manuel Rodriguez-Martinez, Dr. Jaime Seguel
ECE Department
University of Puerto Rico, Mayagüez
{manuel.rodriguez7, jaime.seguel}@upr.edu

***Abstract*— Cloud Computing is a promising paradigm designed to harness the power of networks of computers and communications in a more cost effective way. Clouds provide elastic capacity to serve a wide and constantly expanding range of information processing needs, including government, military, business and education. The Cloud Computing paradigm is maturing rapidly and is being considered for adoption in government and business platforms. Open source systems refer to software systems whose source code is available, allowing for immediate incorporation of improvements and adaptations of the system by its users. This paper reports on an evaluation of open source development tools for Cloud Computing. The main tools examined are Eucalyptus, Apache Hadoop, and the Django-Python stack. These tools were used at different layers in the construction of a notional application for managing weather data. The results of our experience are reported in terms of a capability matrix that grades nine different aspects associated with the use of these tools in the development and deployment of applications in Open Source Cloud Computing environments.**

Keywords-Cloud Computing; Open source clouds;

I. INTRODUCTION

Cloud Computing is an emerging Computing technology that is rapidly consolidating itself as the next big step in the development and deployment of an increasing number of distributed applications [1, 2]. Since the large-scale endorsement issued by the Government of the United States in its FY 2010 budget [3], several agencies have launched Cloud Computing initiatives and conduct actual system developments intended to improve the efficiency and effectiveness of their information processing needs. The list of agencies includes Health and Human Services, Center for Disease Control (CDC), NASA, Navy's Next Generation Enterprise Network (NGEN), and Defense Information Systems Agency (DISA). This movement constitutes a significant departure from the usual information processing infrastructure and culture of federal agencies, business and industry, which is normally based on proprietary solutions deployed on their premises. As result, Cloud Computing is positioned at the center of an interesting and novel market niche. Ultimately, the proponents of Cloud Computing envision a planetary vision of IT services, where each enterprise will be a node connected to a cloud of services.

The US Government has also endorsed the use of open source software tools. Open source refers to any software system whose source code is made available for use or modification by third-party developers. Thus, unlike centralized, proprietary software development models, open source offers practical accessibility to the source code, allowing for immediate and concurrent incorporation of different approaches, and eventually, the branching of the system into customized variants.

Merging these two trends leads to a new breed of tools: Open Source Cloud Computing Tools (OSCCT). These tools provide a free, customizable infrastructure to deploy Clouds for any type of application domain. In this paper, we report on an evaluation of open source development tools for Cloud Computing. This evaluation was made in the context of building a notional application for managing weather data using OSCCT. The main tools examined in our efforts were Eucalyptus, Apache Hadoop, and Django-Python. These tools were used at different layers in the construction of our notional application. The results are reported in terms of a capability matrix that grades ten different aspects associated with the use of these tools for application development and deployment. We found that Hadoop and the Django-Python stack are the easiest tools to use for application development. But, our analysis also indicates that all these tools need to greatly improve their capabilities for configuration and maintenance if they are to increase their user base.

The rest of this paper is organized as follows. Section II describes the motivation and vision for OSCCT. Section III describes our notional application. The evaluation of the OSCCT is found in section IV. Our recommendations are presented in section V. Related works and technologies are found in section VI. Summary and conclusions are found in section VII.

II. MOTIVATION

Consider the development of an application for managing weather information. Such application feeds from a set of heterogeneous, distributed data sources including satellite images, GIS maps, and measurement stations for temperature, wind speed, precipitation, and atmospheric pressure. The application might need to collect historical data and present the user with aggregates from various levels of detail to summarize conditions during days, weeks, or months.

Figure 1 shows the typical architecture used to integrate all these sources, using a data warehouse environment as the integration tool. Major challenges for developers in these environments include a) how to write the software to connect to sources, b) how extract the data and map it to a global schema, and c) how to test new features without disturbing the production systems already in place.

Cloud Computing provides a new tools to tackle some of these problems since the developers of new applications can receive a pre-configure virtual machine with sample data, and sample code on how to build the connectors to securely reach the data sources. The same can be said about examples to map the data to alternative schemas. Rather than spending time learning (sometimes struggling!) how to configure the new technologies, developers are given a turnkey solution that lets them focus on building their new features, instead of learning the details on how to connect to each particular site. Prototype applications can then be deployed on different Clouds to make extensive testing, without purchasing any new hardware. Thus, developers can perform experiments to tune system parameters in a controlled environment before deploying their system to the actual production facility. This capability can reduce application development time and produce more reliable software systems.

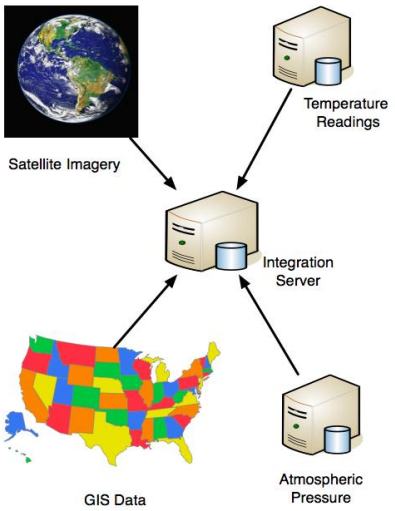

Figure 1. Weather application

A. Cloud Computing Vision

The weather application is emblematic of the new possibilities that Cloud Computing brings to the IT industry and how it changes our way of developing software. Galen Gruman, InfoWorld Executive Editor, describes Cloud Computing as "*A way to increase capacity or add capabilities on the fly without investing in new infrastructure, training new personnel, or licensing new software.*" Eric Knorr, InfoWorld Editor in Chief, envisions world–wide Cloud Computing based services when expressing "*The idea of loosely coupled services running on an agile, scalable infrastructure should eventually make every enterprise a node in the cloud.*"

This planetary vision is shared by Dan Farber, Editor in Chief of CNET News, who has been quoted saying *"We are at the beginning of the age of planetary computing. Billions of people will be wirelessly interconnected, and the only way to achieve that kind of massive scale usage is by massive scale, brutally efficient cloud-based infrastructure."*

According to the National Institute of Standards and Technology (NIST), Cloud Computing is "…*a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."* [4]. The NIST definition identifies five essential characteristics of a Cloud Computing system: *On – demand self – service, broad network access, resource pooling, rapid elasticity and measured service.*

Clouds can be classified based on the service model offered to the users. Three major service models are commonly used: *Software as a Service (SaaS), Platform as a Service (PaaS),* and *Infrastructure as a Service (IaaS).* In SaaS, a specific application is deployed on a cloud. Typically, the GUI of the application is provided to the users to be run from their web browser or desktop. The rest of the application is run from the cloud, and the user has no notion of servers, or specific host locations. The best example of this scheme is Google's GMail. In PaaS, a software application stack is made available to rapid application development and deployment. Windows Azure and Google's App Engine fall in this category. Finally, IaaS provides users with a collection of general purpose computing nodes. Amazon's Elastic Computing Cloud (EC2) technologies fall in this category.

In addition to their service model, clouds are also classified in terms of four deployment models: *Private cloud, Public cloud, Community cloud,* and *Hybrid cloud.* A private cloud is one deployed using the available infrastructure held by an enterprise on its premises. Access to this cloud is restricted to users inside the enterprise. A public cloud is available for use by the general public. Typically, these clouds follow a pay-per-use scheme where users pay for the amount of time they use the resources offered by the cloud. Amazon's EC2 operates this way. Community clouds are private clouds deployed on the premises of two or more enterprises. The goal is to share resources in a more controlled manner compared with public clouds. Hybrid clouds combine both private clouds and public clouds, getting the best features each has to offer.

B. The Impact of Open Source Software

Open source software is usually developed as a public collaboration and is often made freely available. Open Source is indeed a certification mark owned by the Open Source Initiative (OSI). Software that is intended to be freely shared, modified, and redistributed by others may use the Open Source trademark provided that the distribution terms conform to the OSI's Open Source Definition. The main elements in this definition are:

1. The software must be redistributed without restriction.
2. The source code must be made available.
3. The license can require improved versions of the software to carry a different name or version from the original software.

The US Federal Government has endorsed the use of open source software as well. A Memorandum issued in October 2009 by the Pentagon Chief Information Officer to the chief administrators of several dependencies of the Department of Defense reads: "*To effectively achieve its missions, the Department of Defense must develop and update its software-based capabilities faster than ever, to anticipate new threats and respond to continuously changing requirements. The use of Open Source Software (OSS) can provide advantages in this regard.*" The same document emphasizes aspects of open source that may represent a competitive advantage in the software market. Among them are:

"(i) The continuous and broad peer-review enabled by publicly available source code supports software reliability and security efforts through the identification and elimination of defects that might otherwise go unrecognized by a more limited core development team.

(ii) The unrestricted ability to modify software source code enables the Department to respond more rapidly to changing situations, missions, and future threats.

(iii) Reliance on a particular software developer or vendor due to proprietary restrictions may be reduced by the use of OSS, which can be operated and maintained by multiple vendors, thus reducing barriers to entry and exit.

(v) Since OSS typically does not have a per-seat licensing cost, it can provide a cost advantage in situations where many copies of the software may be required, and can mitigate risk of cost growth due to licensing in situations where the total number of users may not be known in advance."

The integration of open source components is a distinctive characteristic of NASA's Nebula project, a Cloud Computing pilot under development at NASA Ames Research Center. According to the project's Website "*Nebula integrates a set of open-source components into a seamless, self-service platform, providing high-capacity computing, storage and network connectivity using a virtualized, scalable approach to achieve cost and energy efficiencies*." The Nebula Cloud Computing system seeks to foster collaborations between scientists and researchers, and serve as an education tool. The system provides high-speed data connections and APIs used by commercial Cloud providers. Nebula is not only being built with open source tools but it is itself an open-source project.

C. Why Open Source Cloud Computing Disrupts Software Development?

Open-source software (OSS) is a common thread permeating Cloud Computing. It is being used to build the Cloud (such as the Linux foundation for Google, salesforce.com and Amazon cloud services); it is being built on top of cloud services (for example, Eucalyptus); and proprietary software used to deliver cloud services is increasingly being moved into open source (including Facebook Open Platform and Hadoop). The relationship that is being forged between OSS and cloud computing is not a matter of coincidence; it is a matter of symbiosis. While massive data centers built on low-cost commodity hardware and virtualized operating environments provide the technical foundations for global class cloud-based services, open-source software enables global-class service-based business models. Cloud computing and open source share a common connection — when a layer of software can be abstracted, they both act as agents to fundamentally alter traditional economic assumptions. Cloud computing does this through lower service costs, greater resource sharing, greater economies of scale, greater levels of architectural standardization and process optimization, and the ability to modify the usage of those resources much more quickly than with traditional software. Open-source software does this by eliminating the ability of any one provider to exploit the terms and conditions of an end user license agreement linked to proprietary code — a problem that increases the more ubiquitous a product becomes.

In this way, cloud computing and open-source software are synergistically bound in a virtuous circle. Cloud computing directly benefits from greater use of OSS, because it eliminates critical dependencies that can impact service delivery. And the more that cloud computing utilizes OSS, the more mature and diverse the open-source stack will become. The more mature and diverse the stack, the more cloud can expand. Another important dynamic of open-source software is that is conducive to service-based business models — a function of the fact that it is very difficult to commercialize code with a fluid ownership structure. Therefore, unlike software governed under an end user license agreement, open-source software enables cloud providers to tie component Service-Level Agreements (SLAs) to their own customer-facing SLAs, which allows for risk distribution.

III. Notional Application for Open Source Clouds

The notional application provides users with information about current weather conditions at various geographic locations. The application combines a Web component, and Hadoop. The web component provides the user-interface and databases to manage common operations and keep system metadata. Hadoop provides the capabilities to run data analysis jobs on demand. Hive, a Hadoop-based data warehouse, provides storage for large data sets. The web

view of the application, which is based on the Google Maps API, is shown in Figure 2. The user can click on a location in the map to its view World coordinates, WOEID, and a link to see the details of the weather conditions at that location. The WOEID acronym, stands for World On Earth ID, and can be used to obtain current weather forecast for a given location from Yahoo!'s Weather service. User can save these locations into a local database and later retrieve them to compare data values. Stored locations are represented as customized markers in the map.

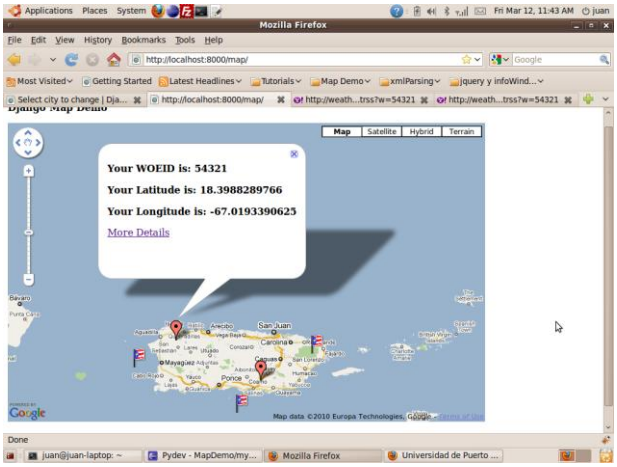

Figure 2. Web Interface for Notional Application

Historic data such as average temperatures, rainfall, and humidity may be shown as other options. These data are obtained from different, geographically distributed data repositories and stored inside the Hive data warehouse. We plan to add a monitoring feature to the system. This feature will enable users to register points whose weather data and forecasts are to be periodically retrieved from the data sources, and stored in the warehouse.

A. *Detailed Application Architecture*

Figure 3 depicts the overall multi-cloud architecture of the system, consisting of three interconnected clouds. Eucalyptus provides a cloud infrastructure with Linux-based virtual hosts to run all servers. On top of this platform cloud, we use Hadoop to build a cloud for large-scale analysis using the MapReduce [5] programming model. Finally, the Djang-Python stack provides another cloud for the web-based application.

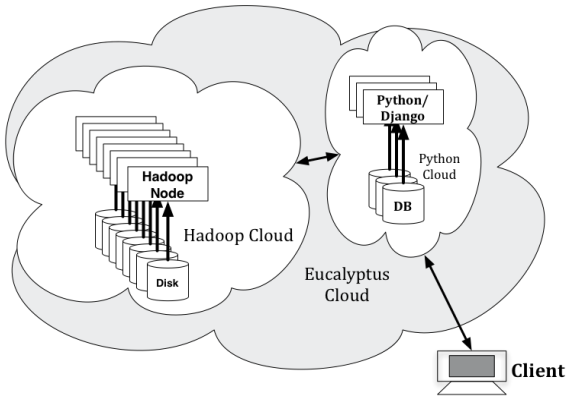

Figure 3. Multi-cloud Application Architecture

Programmatically, the application follows the model-view-controller (MVC) design pattern. The components are organized as follows:

- View – as mentioned before, the user interface is built on top of Google's Map and Javascript. It presents users with a map and a set of markers. Each marker represents a point for which measurement of current weather conditions are available.

- Controller – the controller is built as a Python module running insider Django.

- Model – the model keeps the information about the weather measurements. For each available point, the latitude, longitude, temperature, conditions, and time of measurements are kept. The data was extracted from the Yahoo! Weather page, and from other weather data repositories available on the Internet. Part of the model is kept by Django, namely, the metadata about system collections and users. This model supports three data stores: a) SQLlite – an embedded database library, b) PostgresSQL – a relational database engine, and c) Google' Python Datastore for App engine. The rest of the model is kept as relational data under the Hive data warehouse

IV. EVALUATION OF CLOUD TECHNOLOGIES

In this section we present an evaluation of all these open source tools in light of our experience building the notional application.

A. *Open Source Tools Used*

Eucalyptus – Provides a solution for IaaS, with an API compatible with Amazon's EC2. We installed Ubuntu Linux 9.10, server edition, which comes with a Eucalyptus 1.6 pre-installed. We build a cloud on top of six multi-core server machines, using the private network option. One machine was set up to be the cloud and cluster controller, while the remaining ones were set as node controllers. We also attempted, unsuccessfully, to configure the system with the network operating on MANAGED-NOVLAN Mode. Our

goal with the latter effort was to provide direct access to our cloud from the outside world. Although Eucalyptus comes with Ubuntu Linux, we found that its configuration was tricky, and the instructions should include more examples to understand how to set the multitude of parameters on different scenarios.

Hadoop – Provides an infrastructure for PaaS, specific for parallel processing. We installed Hadoop 0.20 on a six-machine cluster, and on the Eucalyptus cloud. We completed the tutorials for Hadoop and Hive that are available from the Hadoop site, implementing and running their sample applications. Since we had experience working with parallel computing and databases, we found these tutorials relatively easy to complete, and we were ready to write applications in a matter of two days. To further practice our skills with Hive, we ported to it a database-oriented application that processes 1GB worth of data from the TPC-H Decision support benchmark.

Django-Python – Provides an infrastructure for PaaS, specific for web application development. We installed Python 2.6.4 on an Ubuntu Linux virtualized machine. We had no previous experience with Python, but we were able to pickup the language from its on-line tutorial in one day. We then added to this machine Django 1.0, a Web application framework for Python. Django is the Web framework chosen by NASA for the Nebula Project. We completed the Django tutorials in a matter of hours and build an initial weather forecast information application using Django. We deployed the application on the Linux virtual machine as well as in Google's Application Engine. The latter comes with a pre-installed version of Django. We found it much simpler and elegant than either PHP or Java Servlets/JSP.

B. Cloud Technology Capability Matrix

We developed a capability matrix to judge the strengths and weaknesses of these open source cloud technologies. The matrix contained ten different criteria that have impact on the development of a user base for any of these cloud technologies. These criteria were: Management Tools, Development Tools, Node Extensibility, Use of Standards, Security, Reliability, Learning Curve, Scalability, Cost of Ownership, and Support. Table I briefly describes each criterion.

Criterion	Description
Management Tools	Tools to deploy, configure, and maintain the system
Development Tools	Tools to build new applications or features
Node Extensibility	Ability to add new nodes without re-initialization
Use of Standards	Use of TCP/IP, SSH, etc.
Security	Built-in security as opposed to use of 3rd party patches.
Reliability	Resilience to failures
Learning Curve	Time to learn technology
Scalability	Capacity to grow without degraded performance
Cost of Ownership	Investments needed for usage
Support	Availability of 3rd party support

TABLE I. CAPABILITY MATRIX CRITERIA

We score each criterion on a scale of three (3) to zero (0). A score of three indicates that the particular tool provides adequate features to fulfill the criterion. In contrast, a score of zero indicates lack of adequate feature support.

C. Results of the Evaluations

Due to lack of space we do not include the capability matrices for all the technologies but rather discuss how each technology fared on each criterion.

1) Management Tools

Eucalyptus (Score: 2) - We used version 1.6, which comes pre-installed on Ubuntu 9.10 server edition. However, we found that the system must be configured by hand, using configuration files and command-line instructions. Once the system is in operation a Web-based interface becomes available to change some of the parameters in the cloud.

Hadoop (Score: 0) - The system must be configured by hand, with a combination of XML files and text-based configuration files. There are no graphical tools to manage the cloud.

Django-Python (Score: 2) - Creating a Web-application, and configuring its back-end database must be done with command-line tools and Python modules. Once this is done, the application can be managed with an auto-generated Web interface. In addition, if the schema of the application used with the back-end database is changed, then the database configuration tools must be re-run to synchronize the Python application with its database.

2) Development Tools

Eucalyptus (Score: 0) - The source code for Eucalyptus is available and can be extended to add new features. However, we could not find any tools to aid in this effort other than importing the project into the Eclipse IDE or NetBeans.

Hadoop (Score: 3) - There is an Eclipse plug-in that helps in the development and debugging of Hadoop-based applications. This plug-in also helps the developer in the deployment of the application into the cloud. However, the cloud must be configured by hand. Moreover, it appears that active development of this plug-in has been halted.

Django-Python (Score: 3) - The PyDev plug-in can be added to Eclipse and it enables development and deployment of Python applications, including Django-based Web applications.

3) Node Extensibility

Eucalyptus (Score: 3) - Requires manual registration of new hosts, but these become available for use right away.

Hadoop (Score: 3) - Accepts new hosts easily and without any service disruptions.

Django-Python (Score: 0) – Applications needs reconfiguration if new hosts are added. A workaround could be to use a configuration database and add software to control how the system distributes load and requests.

4) Use of Standards

All three tools receive a score of 3 since they all rely on standards for their operation. These include:

- TCP/IP for networking
- SSH for data encryption
- SQL for query specification
- Standard Java (JDK 1.6)
- Python (version 2.6) for application development
- Apache tools for web application development

5) Security

All three tools get a score of 3 since they rely on encryption, and password protected user-accounts to protect and control access to the cloud. The modules that implement these features come built-in with each system.

6) Reliability

Eucalytpus (Score: 2) – Failed nodes are detected but the user becomes aware of such failures.

Hadoop (Score: 3) – The user is shielded from failures at nodes since the Hadoop job controller reschedules the tasks assigned to failed nodes to other available nodes in the cloud.

Django-Python (Score: 0) – Administrators need to handle failures by hand, users become aware of these, and the cloud might need to be restarted.

7) Learning Curve

Eucalyptus (Score: 1) – The system is not well documented and the guidelines provided are not explicit enough or do not necessarily work. For implementing our notional application we had to engage in news forums and track down error by error. It is clear that better technical support will be needed for using Eucalyptus. For production environments, 3rd party support services must be seriously considered.

Hadoop (Score: 2) - The most complex part might be understanding the MapReduce paradigm of computation. Although the paradigm is conceptually simple, its effective use in applications may require some experimentation and detailed knowledge of the inner workings of the Hadoop system. Nonetheless, reading the online tutorials and completing all exercises should allow the users to create a few simple applications in a couple of days.

Django-Python (Score: 3) - Among all three technologies, Django and Python were the easiest to learn. Python's tutorials are well organized and documented, and can be done in one day. After the tutorial, the user is prepared to begin writing programs. Django is also supported by well-written tutorials that provide all the basic information for developers to quickly create production quality applications.

8) Scalability

Due to lack of sufficiently large cluster to make a large cloud we could not evaluate this criterion. However, Hadoop has been deployed on large sites such as Facebook, and Yahoo, where the system has been made to scale to thousand of hosts and petabytes of data. Likewise, Google's Apps Engine uses Python for application development, so a Python-based system could be made to scale to large deployments as well.

9) Cost of Ownership

Eucalyptus (Score: 2) - Eucalyptus requires new, multi-core machines with lots of memory to maximize its benefits.

Hadoop (Score: 2) – Requires a cluster to be effective, although the nodes in such cluster can be cheap, off-the-shelf machines.

Django-Python (Score: 3) – Can be run on any modest machine, and it has been ported to Linux, Mac OS X, and MS Windows.

10) Support

All three tools get a score of 3 since there are 3rd party companies that provide support for them. These include Cloudera, Eucalyptus Systems, and Revolution Systems.

The final score for the tools is:

1. Eucalyptus – 18
2. Hadoop – 22
3. Django-Python - 20

V. RECOMMENDATIONS

When developing new applications based on these open source tools, the developers should follow these recommended practices:

1. If your application will be based on Django-Python, first try to make it work on a private cluster but do not tie in any notion of a server or IP address that the user can be aware of.
2. If your application from (1) needs a cloud infrastructure, consider whether a private or public cloud would do the job. If a private cloud is needed, try to make your application run first on a system like Amazon's EC2, and then move it to Eucalyptus.
3. Eucalyptus can interoperate with EC2, so it is ideal for a hybrid cloud deployment. Applications that experience seasonal peaks can benefit from this since new nodes can be leased as needed.
4. Hadoop does not provide real-time response for the application, as it is designed as a batch processing system. An application built with Hadoop, and intended for processing large amounts of data should be implemented in such a way that the user is informed when a job has been started, and sent an email notification when the output is ready.
5. The documentation for all these tools almost invariably assumes an Unix-like environment. Think hard and long before attempting a different environment since this will most probably involve additional time and resources for experimentation

VI. Related Work

The work in [6] provides an architectural overview of typical cloud technologies. Eucalyptus is rooted on a research effort with the goal of building an open source IaaS framework [7]. Similarly, Hadoop comes as an open source implementation of the MapReduce framework developed at Google [5] for PaaS. Recent work from the database research community [8] has shown that Hadoop is very slow when compared with parallel databases, but it is better in terms of fault-tolerance. HadoopDB [9] has emerged as a hybrid solution that uses a relational database engine as computing node in a Hadoop/Hive installation. The work in [10] surveyed several virtualization technologies for cloud computing. The work in [11] studied a methodology for combining traditional cluster-based services in a cloud environment. For many research issues associated with Cloud computing the user is referred to [12].

VII. Summary and Conclusions

Cloud Computing is a promising paradigm designed to harness the power of networks of computers and communications in a more cost effective way. In this paper, we reported on an evaluation of open source development tools for Cloud Computing. This evaluation was made in the context of building a notional application for managing weather data using OSCCT. The main tools examined in our efforts were Eucalyptus, Apache Hadoop, Django-Python. These tools were used at different layers in the construction of our notional application. We evaluated all three technologies based on a capability matrix used to characterize several important features. Hadoop obtained 22 points, Django-Python received 20, and Eucalyptus obtained 18. Django-Python and Hadoop are simpler to configure and use. However, the ability for Eucalyptus to interoperate with Amazon EC2 makes it an important tool to use in any real application where seasonal peaks require additional computational power.

Acknowledgements

This work was completed using the Lockheed Martin Corporation Thundercloud™ cloud computing design pattern.

References

[1] R. H. Katz, “Tech titans building boom,” IEEE Spectrum, vol. 46, no. 2, pp. 40–54, February 2009.

[2] D. A. Patterson. “Technical Perspective: The Data Center is the Computer”. Communications of the ACM, 51(1):105–105, 2008.

[3] J. Nicholas Hoover, “Federal Budget Lays Out Government Cloud Computing Plans”, Information Week, May 12, 2009. URL: http://www.informationweek.com/news/government/enterprise-architecture/showArticle.jhtml?articleID=217400505. Access date: September 22, 2009.

[4] Peter Mell and Tim Grance, “The NIST definition of Cloud Computing” http://csrc.nist.gov/groups/sns/cloud-computing/ . Access date: December 7, 2009.

[5] J. Dean and S. Ghemawat., “Mapreduce: Simplified data processing on large clusters,” in Proc. of 2004 OSDI, San Francisco, CA, USA, 2004, pp. 137–150.

[6] A. Lenk, et. al, “What’s Inside the Cloud? An Architectural Map of the Cloud Landscape”, in Proc. ICSE Workshop on Software Engineering Challenges of Cloud ComputingVancouver, Canada, 2009, pp. 23-31.

[7] D. Nurmi, et.al., “The Eucalyptus Open-source Cloud-computing System”, in Proc. 9th IEEE/ACM International Symposium on Cluster Computing and the GRID, Shangai, China, 2009, pp. 124-131.

[8] A. Pavlo, et.al., “A comparison of approaches to large-scale data analysis,” in Proc. 2009 SIGMOD, Providence, RI, USA, 2009.

[9] A. Abouzeid, et.al, "HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads", In Proceedings of VLDB 2009, Lyon, France.

[10] D. Cerbelaud, S. Garg, and J. Huylebroeck, “Opening The Clouds: Qualitative Overview of the State-of-the-art Open Source VM-based Cloud Management Platforms”, in Proc. Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware, Urbana, Illinois, 2009.

[11] R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, “Elastic Management of Cluster-based Services in the Cloud”, in Proc. 1st workshop on Automated Control for Datacenters and Clouds, Barcelona, Spain, 2009, pp. 19-24.

[12] B. Hayes, Cloud computing, Communications of the ACM, v.51 n.7, July 2008