Cloud Computing for Teaching and Learning:

Strategies for Design and Implementation

Lee Chao

University of Houston–Victoria, USA

Managing Director:	Lindsay Johnston
Senior Editorial Director:	Heather A. Probst
Book Production Manager:	Sean Woznicki
Development Manager:	Joel Gamon
Development Editor:	Hannah Abelbeck
Acquisitions Editor:	Erika Gallagher
Typesetter:	Jen McHugh
Cover Design:	Nick Newcomer, Lisandro Gonzalez

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: <http://www.igi-global.com>

Copyright © 2012 by IGI Global. All rights reserved. No part of this publication may be reproduced, stored or distributed in any form or by any means, electronic or mechanical, including photocopying, without written permission from the publisher. Product or company names used in this set are for identification purposes only. Inclusion of the names of the products or companies does not indicate a claim of ownership by IGI Global of the trademark or registered trademark.

Library of Congress Cataloging-in-Publication Data

Cloud computing for teaching and learning: strategies for design and implementation / Lee Chao, editor.

p. cm.

Includes bibliographical references and index.

Summary: "This book provides the latest information about cloud development and cloud applications in teaching and learning, including empirical research findings in these areas for professionals and researchers working in the field of e-learning who want to implement teaching and learning with cloud computing"--Provided by publisher.

ISBN 978-1-4666-0957-0 (hardcover) -- ISBN 978-1-4666-0958-7 (ebook) -- ISBN 978-1-4666-0959-4 (print & perpetual access) 1. Cloud computing. 2. Web-based instruction. 3. Computer-assisted instruction. I. Chao, Lee.

QA76.9.C58.C585 2012

004.6782--dc23

2011047365

British Cataloguing in Publication Data

A Cataloguing in Publication record for this book is available from the British Library.

All work contributed to this book is new, previously-unpublished material. The views expressed in this book are those of the authors, but not necessarily of the publisher.

Chapter 6

Teaching Clouds: Lessons Taught and Lessons Learnt

Lee Gillam

University of Surrey, UK

Bin Li

University of Surrey, UK

John O'Loughlin

University of Surrey, UK

ABSTRACT

In this chapter, the authors discuss the scope, content, and technical challenges offered up in the construction and delivery of a 10 week long Cloud Computing module that combines discussions of the principles and key characteristics of Cloud Computing with a series of practical exercises and an implementation-based coursework. The authors present an overview of the core of this module, which starts from the Software, Platform, and Infrastructure (SPI) model and builds from this around SOAP and REST, Hadoop, related paradigms such as Grids and Peer-to-Peer (P2P) computing, and the all-important Service Level Agreement (SLA). The chapter further describes the practical exercises undertaken in lab-based sessions, and the nature of the assessment. It concludes with a brief discussion of the lessons learned to date through this delivery.

INTRODUCTION

The emergence of Cloud Computing as a topic of mass market interest is beginning to be matched by the emergence of the subject in its own right in Higher Education. In 2009, numerous departments of Computer Science, Informatics, and so forth

could readily claim to be teaching students about myriad aspects of the subject that they would deem inherently relevant to Cloud, from data structures and algorithms, through object oriented programming, to information retrieval. However, there was relatively little by way of teaching that was geared entirely to Cloud Computing per se. Our view was that it would be necessary to begin by

DOI: 10.4018/978-1-4666-0957-0.ch006

considering fundamental definitional questions for the subject, and the very discussion of what Cloud Computing was all about would lead to development of an appropriate understanding. We constructed and delivered just such a module in early 2010 to a cohort of Masters students, geared towards developing Cloud Computing literacy for the future IT professionals who, we expected, would be very much entering into a marketplace in which Cloud is becoming a key component of the IT landscape. At the second iteration of this teaching, it is apparent that the word Cloud in relation to Computing is beginning to become part of everyday parlance, with industry, government, and other sectors at various stages of Cloud adoption. This is allied to an increasingly clear demand from industry for those with Cloud knowledge and skills evidenced through job advertisements – and such demand is only likely to increase in the near term.

Our Cloud Computing module is delivered over 10 weeks, with 2 hours of lectures in each, and 2 hours of guided hands-on during the first 6 weeks; the remaining 4 weeks of hands-on is geared to providing assistance to students with their assessments. We have also featured guest lectures from Amazon, IBM, and Imagination Group to provide an industrial flavour.

In this Chapter, we discuss the scope, content, and technical challenges offered up in the construction and delivery of this Cloud Computing module. Since the subject – and the technology, by and large – is still in relative infancy despite the increases apparent in uptake, it is expected that the scope, content, and technical challenges will all vary significantly over the coming years; furthermore, the principles of Cloud Computing are likely to find themselves more deeply ingrained into other aspects of programme delivery as the subject emerges such that Cloud might become more clearly signposted throughout programmes, as well as potentially offering the technological vehicle upon which to develop such programmes.

OUTLINE OF THE MODULE: LECTURES

To introduce both the principles and practical applications of the Cloud, the important first step for us was to establish the set of seed definitions and distinctions of Cloud from which the subject should grow. It is not necessary to have the perfect set of definitions or defining characteristics – indeed, discussion of fuzziness of definition offers substantial opportunity for discussion based around attempting to interpret such definitions and determine their application to things which might, or might not, be Clouds. Such a set of definitions enables us at least to appraise offerings which are labeled “Cloud” and determine whether they fit such a label. In certain cases, the rebranding of extant products may not necessarily result in a good fit, especially if it is merely cosmetic – for example, adding ‘cloud’ to product names.

Our starting set of definitions comes from Mell and Grance (2011) at the U.S. National Institute of Science and Technology (NIST). The NIST definitions necessitate time spent dealing with the expansion of their defining characteristics (emphasized below in bold):

“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. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models”.¹

The five characteristics introduce notions that Cloud technologies should be widely accessible using a range of networked devices, self-service, and metered such that usage can be assessed and likely charged for. From the Cloud user’s

perspective, they should be readily able to scale their systems up and down as needed; for this to be possible, the provider must be able to offer sufficient resources – their systems must appear elastic. The provider should offer access to a pool of computational resources which may be used by multiple users simultaneously (multi-tenant) through partitioning at application level or through system virtualization of various kinds, and are highly likely to be recycled across customers over time. The Cloud user may not know where the relevant resource is, physically, but should always be able to contact it.

Introducing the three familiar service models, referred to elsewhere as SPI [Software, Platform, and Infrastructure as a Service, or SaaS, PaaS and IaaS], with a particular focus on what the Cloud user is able to control – application configuration for SaaS, application development but not the development environment for PaaS, and the illusion of having full control over a physical system and being able to define the environment from server resources upwards for IaaS.

The four deployment models of Public, Private, Hybrid and Community are mentioned in more critical terms initially. We note in particular that ‘Public’ is typically used to denote the offerings by companies, as would contrast with government-supported resources such as a public healthcare system or public library which are typically intended to be free at the point of use. A ‘Private’ Cloud may also be something offered by a ‘Public’ provider where systems are set aside for a specific user. A degree of additional confusion is brought about by adding Hybrid and Community to this mix: a Hybrid could blend Public, Private and Community, and a Community may be composed of Hybrids. We refine such notions later, rather than attempt to clarify from the start.

Given the characteristics and models, we are subsequently able to relate names of Cloud providers and assess the degree of satisfaction of these characteristics; some providers emerge better from such an appraisal than others.

Software as a Service (SaaS)

Miller (2009) offers a catalogue of SaaS offerings, some of which have ceased to exist since the date of publication, demonstrating a specific risk of becoming dependent on a particular SaaS vendor (“vendor lock-in”) which can be mitigated by planning an exit strategy from the start.

Discussion of SaaS begins by situating it with respect to the heritage of what may now be deemed Cloud, in mainframes and application service providers. It also provides ground for suggesting a set of characteristics of SaaS, and identifying potential advantages and some key disadvantages of migrating to SaaS from both a business software perspective and from a software developer perspective. Our key SaaS examples are Google Mail and Apps and Salesforce, against which we test our characteristics, and from Miller we condense a list of examples of applications which might be useful when considering how to replace specific on-premises software (Table 1). Each SaaS will likely have different terms and conditions, pricing, support, and so on. So, the savvy Cloud user must undertake a robust comparative review, making best use of free trial periods where they are available, and being very careful about the terms and conditions of use.

Platform as a Service (PaaS)

Similar to our discussion of SaaS, we begin by offering a set of characteristics of PaaS. Key amongst these is that the Cloud user is more likely to be a software developer, and that they will be constrained to how they develop that software by what the vendor limits them to. There can be a trade-off here between the highly beneficial features of the vendor’s platform, such as not having to worry about so-called “heavy-lifting” in having to set up a comparable automatically scalable infrastructure, and the kinds of restrictions it may impose. In some ways, PaaS offers the greatest dangers of vendor lock-in: in SaaS,

Table 1. A selection of SaaS, collected from Miller (2009)

Application	Examples
Calendar	Google, Yahoo, Windows Live, CalendarHub, Hunt Calendars
Schedules	Diarised, Windows Live Events, AppointmentQuest
Planning / Task Management	Bla-bla List, Hiveminder, HiTask, Zoho Planner
Event Management	Conference.com, RegOnline, Event Wax
Project Management	BaseCamp, Project Drive, Zoho Projects, onProject
Web Databases	Zoho Creator / Zoho DB & Reports, QuickBase, Lazybase
Bookmarking	BlinkList, Clipmarks, del.icio.us, Tagseasy
Photo Editing	FotoFlexer, Preloadr, Snipshot
Photo Sharing	dotPhoto, Flickr, Photobucket, Picasa Web Albums
Desktops	ajaxWindows, eyeOS, g.ho.st, YouOS
Web Conferencing	Genesys Meeting Center, WebEx, Zoho Meeting
Groupware	Contact Office, Project Spaces, teamspace
Blogs and Wikis	Blogger, TypePad, wikihost.org, Wikispaces, Zoho Wiki

an exit strategy might require the transfer, and potential transformation, of large quantities of data across providers, always assuming that there is an alternative provider. For PaaS, there might be no alternative provider, and this introduces the risk that either the entire hosting environment of the PaaS provider has to be replicated, which might negate a benefit of moving to PaaS in the first place, or the application has to be rewritten to an alternative provider – with concomitant costs. Furthermore, the PaaS user is at the mercy of the provider in relation to changes to the underlying system, the approach(es) offered for data storage, and so on. It will be for the PaaS user to ascertain the value of the gain against the potential for loss.

Our key PaaS examples are Google App Engine, Microsoft's Azure, and Force.com. We discuss Force.com in brief, using it as an example of turning the development platform for SaaS into a product in its own right, and to suggest how the opening up of systems can itself lead to additional business opportunities.

For the former, Severance (2009) provides valuable insights into how the platform works such that applications might migrate across Google data centers, picking up on a key Cloud characteristic,

and can scale automatically based on demand and migrate closer to that demand. We look at the structure of relatively simple Python-based examples as a means to explain the sequence of actions relating to such applications, including the use of handlers and pushing data into page templates. Further, on the basis of applications we have built and hosted at appspot.com, we are able to show the dashboard features relating to usage measurements, the underlying storage with its bespoke query language, and various other application management features. Setting up Google App Engine is a relatively straightforward task, and it is quite possible to have applications being hosted in a very short time.

Following on from Google App Engine, we discuss the Azure platform. Here, we draw on findings of a dissertation undertaken in a previous year by one of our Masters students, discussed in part in Gillam, Cooke and Skinner (2009), and in particular the need to formulate developed applications using .NET and by splitting function across web and worker roles, and the kinds of storage on offer. In contrast to Google App Engine, at the time of writing the setup efforts required to begin to use Azure are relatively substantial and time

consuming, and would have necessitated at least an upgrade in operating system across a lab even to get started with installing the other necessary packages, which includes Visual Studio, .NET, SQL Server or another Microsoft database, and potentially various specified fixes. If an application were already close to such a platform, it might be a worthwhile endeavour. For the quantity of time we are able to allocate to PaaS, it is slightly disproportionate.

Whilst we have suggested Azure falls under PaaS, the NIST definition of PaaS contains the notion that the PaaS user may be able to control the application hosting environment. This is not especially the case for Google App Engine, but evident with Azure in that differential pricing exists for compute instances – the PaaS user here needs to think about the performance of the application, and provision infrastructure accordingly. This provides an easy lead in to IaaS.

Infrastructure as a Service (IaaS)

We complete our coverage of the deployment models with IaaS. Here, Reese (2009) becomes a valuable resource in helping us to establish the key characteristics of IaaS, in relation to those of Cloud in general. Reese also helps us to explain how IaaS can be positioned in relation to rental and ownership models, or “internal IT” and “managed services” to use his terms. Where Reese suggests that Cloud is in competition to these, we suggest that it is more likely to be complementary – based on the nature of the work being done by such systems, and constraining factors such as regulatory or legislative requirements, these three approaches may be used together in various combinations. Reese’s contrast of characteristics across these three models is highly informative, and in part offers a rationale for selecting one in preference to the others.

This discussion of IaaS leads directly into the Amazon Web Services (AWS) offerings, and has subsequently led onto an elaboration of how

Eucalyptus works, insights into which are much easier to demonstrate. The availability and nature of components of AWS is described, focusing primarily on EC2 and the different kinds of storage available – S3, EBS, and instance (ephemeral) and how it is possible to setup and start-up instances more rapidly if EBS is used for the boot disk – and covering the necessity of firewall configurations, since instances will be otherwise inaccessible, and the credentials required to access these resources. It is then possible to show how a multi-tier application could be setup across multiple instances, with restricted communications across them offering a greater challenge to those attempting to break into such a system to obtain the raw data. Focus on Eucalyptus helps to clarify the need for virtualization, distinguish between full virtualization and paravirtualization, and elaborate the virtualization of networking within this system. This hopefully leads to a deeper technical understanding of the principles underlying IaaS offerings, particularly AWS.

SOAP, REST, CRUD, and JSON

Having used SPI to introduce various Cloud technologies, and along the way to identify where the other defining characteristics are met, and in discussing IaaS having suggested how systems might be interlinked, familiar Cloud interfaces are described. With SOAP and REST, or REST-like protocols commonly used in Clouds, we cover the four parts of SOAP messages and how REST offers a lightweight approach to achieving similar goals but may not achieve a similar level of robustness. As one might expect, CRUD is introduced in relation to REST, and this also helps in relating to how REST might be used for indirect manipulation of databases. We briefly contrast XML, intended to be *machine-readable* but into which people will largely embed dictionary-like human-readable labels, with JSON which provides representation without such an overhead of tags.

The combination here means that a range of examples can be related, which includes Google Search returning JSON, multiple interactions in system set up in EC2 through both SOAP and a REST-like API, and Azure Storage configuration. Furthermore, it is possible to suggest how new applications can be constructed using such interfaces, such that a new Cloud service might reuse several of these existing Cloud services in combination across the different providers – for example, to use EC2 with Azure Storage and Google Search to explore how one might offer text analytics services.

Hadoop/MapReduce

Having broadly addressed the nature of Cloud storage in relation to IaaS, it is trivial to suggest how to put the computation close to the data if the data are already available in Cloud storage. AWS hosts various EBS snapshots, so it is readily possible to start instances in EC2, mount EBS volumes that contain a copy of the snapshot, and begin working on the data. We can relate the operation of Hadoop to this, since the principle of putting computation near data is a cornerstone of this readily scalable system. HDFS offers a way to create redundant data storage across multiple machines, to which programs could also be migrated – a combination which offers both a data and compute cluster. With Hadoop, it is possible to use either HDFS or S3, which already deals with data redundancy, and MapReduce is used as the programming framework. We briefly mention how MapReduce supports parallel execution up to a point, how it is possible to use Hadoop's Pig Latin to express MapReduce workflow at a much higher level, and the kinds of tasks for which using MapReduce is likely to be suitable – typically, analytical tasks with large data volumes, rather than transactional tasks.

Related Paradigms Including Grids and Peers

In AWS, one service offering of additional academic interest is high performance computing (HPC) where the instances will be started in close physical proximity and have high-speed networking amongst them: a knowledgeable AWS user can set up a large cluster in a relatively short time, run the required analysis, and shut the whole system down at will.

We characterize HPC in relation to a background of distributed and parallel computing and in contrast with high throughput computing (HTC). Having already covered Hadoop, we can build out discussion of clusters and introduce scheduling systems such as Condor and Sun Grid Engine (now Oracle Grid Engine); more importantly, we key into both Grid Computing, and can relate this along the lines of cross-institutional clusters, and Peer to Peer (P2P) computing which we can relate to a data cluster.

It is interesting to pick up on the promises of the Grid, and to question whether this promise was delivered upon. Consider, for example:

“Grid infrastructure will provide us with the ability to dynamically link together resources as an ensemble to support the execution of large-scale, resource-intensive, and distributed applications.”
[Berman, Fox and Hey, 2003]

A number in the Grid community would argue that Clouds are simply Grids in disguise. However, if we consider the SPI model then Grids are closest to PaaS offerings, since the users typically have no control over what is available and may have to adopt specific standard versions of software in order to interact with it, and occasionally offering up (scientific) SaaS where it is only possible to adjust parameters of known applications. Furthermore, it is difficult to justify Grids as satisfying characteristics such as self-service – particularly when verifying identity may require a physical

journey, which may encompass some distance, to show your passport to another human being before you are let anywhere near the system. Once you have access, you can use clusters that are hosted at a number of organizations *if they are available*, and so the characteristic of elastic may also be difficult to satisfy.

An interesting negative example of industrial Grid computing is offered up by the network.com offering from Sun (now Oracle), which gradually faded away due to a lack of customer uptake. Sun's \$1 per CPU hour, launched 2 years after AWS in 2004, tied users to Sun's operating systems and hardware. AWS, amongst other IaaS offerings, suggests that flexibility is an important characteristic of Cloud Computing.

In relating P2P with Cloud, we can transition from centralized P2P systems – and offer a ready analogy to HDFS – through structured and unstructured decentralized P2P systems, to hybrids involving super-peers. Within this, we offer recent examples from literature that bridge Cloud and P2P relating to head-node redundancy for HDFS (Marozzo, Talia and Trunfio 2010) and provisioning and load balancing (Ranjan et al, 2010).

Service Level Agreements

For many, the Service Level Agreement (SLA) is the key to use of Cloud Computing systems. Through a module covering ethics, professionalism and law, students should already have some appreciation of contract terms and basic provisions, and we discuss the relevant terms in off-the-shelf SLAs from specific Cloud providers, identifying in particular the difference between *commercially reasonable efforts* and *best efforts*.

While it is possible to gain a broad understanding of SLAs, to offer more of a practical consideration we focus towards task-specific machine readable SLAs and, in particular, our own research in this direction (Li and Gillam 2009; Li and Gillam 2010). We look at some of the Service Description Terms and Guarantee Terms of WS-Agreement, and how the ability to capture

offers across a number of Cloud providers offers opportunities for Cloud Brokers. Subsequently, we suggest how the adoption of certain kinds of analysis of financial data offers potential for producing such offers and predicting the performance and likely penalty incurred by failures in the underlying systems/providers.

Further Topics

In rounding out our discussions of Clouds, there are a variety of topics which attract less attention but which are no less important. We introduce a few of these here, though this is not intended to be comprehensive.

Cloud Economics

Business Week's 10 laws of Cloud Economics² provide interesting grounds for discussion of a range of issues, including availability as a factor of multiples of data centers and the difficulties of moving an entire data center. The scale of new data centers, approach to populating them, and costs involved suggest substantial sunk investments by the providers, which can only really be recouped when utilization is high. For data center providers, it is in their interests to ensure a high ratio between the energy put into a data center and the computational service provision – the so-called Power Usage Effectiveness or Efficiency (PUE).

Economics is also important in relation to the cost of the work being undertaken in the Cloud; the less efficiently it is done, or when resources are simply left running when they need not be, the more expensive it will be. The sudden awareness of expenditure can have an interesting effect on students – particularly when charges can be directly applied to their own credit cards. Such exposure should lead to interesting consequences in terms of future software development, not only in raw performance, but also in the costs of using certain protocols and interchange languages which add large amounts of markup.

Green Clouds

We refer to the Greenpeace report “Make IT Green: Cloud Computing and its Contribution to Climate Change”³ which discusses the need to consider PUE in combination with how green the energy is. Certain data center providers are criticized for needing to rely on power stations which principally use fossil fuels rather than renewables. In using Clouds, we are contributing to the energy use in the country of the provider which offers potentially interesting considerations when thinking about carbon credits and trading. As a side note, we also criticize the Greenpeace report for being computationally inefficient—it is a 7MB download comprising a mere 12 pages, with heavy use of colour, graphics and large amounts of whitespace also making for expensive printing should anybody decide to read it offline.

Cloud Security

An oft-cited reason for not going to the Cloud is that it might somehow be insecure. This can be countered in three ways: (i) the security offered by providers such as Amazon is usually rather more robust than that offered by organizations for whom it is not an inherent part of their revenue stream; (ii) the personnel employed by providers to ensure systems remain secure must be very good at their jobs because of the importance of security to the Cloud providers; (iii) all systems that are network-attached have the potential to suffer from the same attacks, and if you’re not able to be more secure than those providers, your own systems are likely at greater risk.

We refer to the AWS whitepaper on security best practices⁴, and offer a brief summary along the lines that most will be thinking about protecting data in transit, protecting data at rest, and protecting the credentials and keys. A significant issue emerges here: the data can largely only be processed in unencrypted form, so at some point the data will need to exist in unencrypted form in a

system. As well as discussing hardened operating systems (bastion hosts), Reese offers discussion of data separation such that certain kinds of data could readily exist in clear but when taken alone are insufficient to cause a problem. It may be desirable still to encrypt some part of the data, but without access to the lookup table that associates it with other data, it is not likely to be useful. Reese offers credit card data as an example. We also consider a financial data provider who may wish to host data in the Cloud, who could store the list of company names entirely separately from a derivative of the share prices such that it would be necessary to capture two sets of separate data, capture the associating lookup table, and know the means to reverse the derivation. Encryption could be added for good measure.

Cloud Use Cases

Finally, we can discuss the various Use Cases presented in the Cloud Computing Use Cases White Paper⁵, and also Research Use Case Scenarios⁶, and we have covered a significant majority, though not necessarily in substantial detail in certain places, of the Cloud Stack and the Cloud definitions are once more borne out in a more informed appreciation.

OUTLINE OF THE MODULE: HANDS-ON

The six weeks of hands-on sessions are geared towards getting students started with understanding and using Cloud systems and deploying applications. Lab sessions introduce students to a Platform as a Service, a public Infrastructure as a Service, a private Infrastructure as a Service, and – in effect - a Distributed Database as a Service. An extensive workbook is provided each way, in some cases based on existing tutorials, and in other cases crafted ab initio. The six weeks cover the following topics:

1. Getting started with Google App Engine—setting up appspot.com endpoints, building and testing Python applications in the local sandbox, and deploying these to appspot.com. This helps to introduce the kinds of monitoring being undertaken by providers.
2. Getting started with AWS—obtaining credentials, setting up security groups and firewall rules, and starting up and configuring initial EC2 instances. This helps to get going with the idea of building systems from the ground up using third party infrastructures.
3. Configuring a complex two-tier web application in AWS using two different regions, for example using a webserver such as Websphere in US-East, and having a database such as DB2 running in EU-West. This is helpful in bringing into focus the need to consider latency and resilience within systems.
4. Configuring instances and rebundling into AMIs in order to reuse. This is geared towards both interacting with EC2 using the command line interface (CLI), and to being able to configure and “save” entire systems for subsequent reuse such that costs are reduced substantially whilst systems are not running. The costs of configuration in 3, above, are then also apparent.
5. Setting up and running instances using Eucalyptus. With a similar CLI to EC2, this reinforces the use of such commands, and offers insights into how to make use of such a system within an organization. It also introduces the challenge of dealing with Hybrid Cloud systems, particularly when there is a firewall in the way which the user has no control over configurations for.
6. Using Hadoop and Amazon MapReduce in AWS. Text analysis (word frequency count) examples are used here, and this demonstrates how to use S3 in place of

HDFS for MapReduce tasks. The principles of MapReduce are covered in relation to storage elsewhere.

By the end of these six weeks, the speed with which it is possible for these students to start building systems using any of these technologies should be readily apparent, and this is also necessary in order that the students will subsequently be able to use their ability to do this to undertake the assessment.

ASSESSMENT

Assessment is undertaken entirely on the basis of coursework, hence the coursework needs to present a reasonable degree of challenge. The coursework set comprises three parts: (i) a system proposal; (ii) a description of the final system; (iii) a presentation from the student in which they cover what they achieved and what they did not as well as demonstrating the result. The proposal is used formatively to see how students are approaching the problem and whether they understand the nature of the problem they are trying to tackle. Assessing the proposal usually prevents a decent proportion of students from continuing with approaches which are rather more complex than necessary, or from taking the wrong approach to developing the analytical component required.

The software that needs to be produced entails the integrated, though loosely-coupled, use of three kinds of systems: a public Platform as a Service, a public Infrastructure as a Service, and a private Infrastructure as a Service to provide for a specified – if simplified - Software as a Service. Students are required to use Google App Engine, AWS and Eucalyptus to develop software that can be readily scaled according to a specified ratio of public to private instances. The assessment necessitates use of a RESTful approach and the

implementation of a simple demand-driven approach to scaling, bringing together some of the principles and practices. Due to issues that arise with coursework descriptions appearing on sites with requests for them to be completed for certain sums of money, the actual description of this coursework is omitted, but relates to the analysis of certain kinds of financial information.

CONCLUSION

Whilst initially developing this module, there were relatively few potential course texts to choose from. By its second run, candidate course texts are rather more readily available, with our own edited Springer volume (Antonopoulos and Gillam 2010) currently being used as a source for reference material in various places.

With the second run of this module nearing completion at the time of writing, and with some 46 students having participated, there is little doubt that the subject of Cloud Computing blends excitement and novelty with challenge and complexity, and some students end up being stretched rather further than others. The majority of students are incredibly positive about what Cloud offers, and the way in which it shifts their thinking to incorporate more than a traditional ownership model of IT. The ability to start and run a number of reasonably powerful servers within mere minutes, and from there to be able to build out scalable applications, and to be able to think beyond typical limits of extant owned systems, offers a minimal demonstration of the possible; this quickly becomes second nature for the students.

Running such a technology-dependent module is not always straightforward. The Eucalyptus system has not proven particularly robust, and queries are not often answered. Hence, we are looking to migrate our Private Cloud to OpenStack which offers a similar interface but has a variety of differences in the underlying implementation. The relative recency of OpenStack, however,

leaves plenty of room for the emergence of new problems. Additionally, we have encountered versioning issues when dealing with the two-tier web application – in the first year, IBM had to rebuild one of the images on which this was based, which took about a week; more recently, we have had to update our hands-on explanations when a minor version increment was accompanied by a complete change in how the system was configured. These are the kinds of technology problems that are likely to frustrate some users, be they in academia or in industry, and so the discussion of such issues with the students brings them more directly into this, and offers more opportunities for learning than might exist when it “just works”.

ACKNOWLEDGMENT

This work has been supported in part by the EPSRC and JISC (EP/I034408/1), and we gratefully acknowledge Amazon Web Services for the supporting grants for teaching and research, and specifically Kurt Messersmith, who has offered support over and above this, and Iain Gavin for his regular guest lecture appearance. We are also grateful to Simon Baker, IBM, and Matt Ballantine, previously of Imagination Group and now with Microsoft, for their guest lecture contributions.

REFERENCES

- Antonopoulos, N., & Gillam, L. (Eds.). (2010). *Cloud computing: Principles, systems and applications*. Springer-Verlag.
- Gillam, L., Cooke, N., & Skinner, J. (2009). *Towards executable acceptable use policies (execAUPs) for email clouds*. IEEE e-Science 2009 Workshop on Cloud-based Services and Applications.
- Hammond, M., Hawtin, R., Gillam, L., & Oppenheim, C. (2010). *Cloud computing for research*. UK: JISC.

Li, B., & Gillam, L. (2009). *Towards job-specific service level agreements in the cloud*. IEEE e-Science 2009 Workshop on Cloud-based Services and Applications.

Li, B., & Gillam, L. (2010). Towards application-specific service level agreements: Experiments in clouds and grids. In Antonopoulos, N., & Gillam, L. (Eds.), *Cloud computing: Principles, systems and applications*. Springer-Verlag. doi:10.1007/978-1-84996-241-4_21

Marozzo, F., Talia, D., & Trunfio, P. (2010). A peer-to-peer framework for supporting MapReduce applications in dynamic cloud environments. In Antonopoulos, N., & Gillam, L. (Eds.), *Cloud computing: Principles, systems and applications*. doi:10.1007/978-1-84996-241-4_7

Mell, P., & Grance, T. (2011) *The NIST definition of cloud computing*. NIST Special Publication 800-145 (Draft). Retrieved from http://csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf

Miller, M. (2009). *Cloud computing: Web-based applications that change the way you work and collaborate online*. Que Publishing.

Ranjan, R., Zhao, L., Wu, X., Liu, A., Quiroz, A., & Parashar, M. (2010). Peer-to-peer cloud provisioning: Service discovery and load-balancing. In Antonopoulos, N., & Gillam, L. (Eds.), *Cloud computing: Principles, systems and applications*. doi:10.1007/978-1-84996-241-4_12

Reese, G. (2009). *Cloud application architectures: Building applications and infrastructure in the cloud: Transactional systems for EC2 and beyond*. O'Reilly Media, Inc.

Severance, C. (2009). *Using Google App engine*. O'Reilly Media, Inc.

ADDITIONAL READING

Armbrust, M., & Fox, A. Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., ... Zaharia, M. (2009). *Above the clouds: A Berkeley view of cloud computing*. Technical Report EECS-2009-28, EECS Department, University of California, Berkeley. Retrieved from <http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html>

Bailey, M. (2009). *The economics of virtualization: Moving toward an application based cost model*. IDC. Retrieved from <http://www.vmware.com/files/pdf/Virtualization-application-based-cost-model-WP-EN.pdf>

Baun, C., & Kunze, M. (2009). Building a private cloud with eucalyptus. *Proceedings of the 2009 5th IEEE International Conference on e-Science Workshops*, Oxford, UK.

Buyya, R. (2002). *Economic-based distributed resource management and scheduling for grid computing*. PhD thesis, Monash University, Melbourne, Australia.

Buyya, R., Abramson, D., & Giddy, J. (2000). An economy driven resource management architecture for global computational power grids. *Proceedings of the 2000 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA2000)* (pp. 517–525). Las Vegas, NV: CSREA Press.

Buyya, R., Abramson, D., & Venugopal, S. (2005). The Grid economy. In *Special Issue of the Proceedings of the IEEE on Grid Computing*. Los Alamitos, CA: IEEE Press.

Buyya, R., Giddy, J., & Abramson, D. (2001). *A case for economy grid architecture for service-oriented grid computing*. In 10th IEEE International Heterogeneous Computing Workshop, San Francisco, California, USA.

- Buyya, R., Yeo, C. S., & Venugopal, S. (2008). *Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities*. CoRR (abs/0808.3558)
- Cheliotis, G., Kenyon, C., & Buyya, R. (2005). 10 lessons from finance for commercial sharing of IT resources. In Subramaniam, R., & Goodman, B. D. (Eds.), *Peer-to-peer computing: The evolution of a disruptive technology*. IIRM Press.
- Foster, I., Kesselman, C., Nick, J., & Tuecke, S. (2002). Grid services for distributed system integration. *Computer*, 35(6), 37–46. doi:10.1109/MC.2002.1009167
- Foster, I., Kesselman, C., & Tuecke, S. (2001). The anatomy of the grid: Enabling scalable virtual organizations. *The International Journal of Supercomputer Applications*, 15(3), 200–222. doi:10.1177/109434200101500302
- Geelan, J. (2009) *Twenty-one experts define cloud computing*. Cloud Expo. Retrieved from <http://cloudcomputing.sys-con.com/node/612375>
- Gens, F. (2009) *New IDC IT cloud services survey: Top benefits and challenges*. IDC research. Retrieved from <http://blogs.idc.com/ie/?p=730>
- Germano, G., & Engel, M. (2006). *City@home: Monte Carlo derivative pricing distributed on networked computers*. Grid Technology for Financial Modelling and Simulation.
- Gray, J. (2003). *Distributed computing economics*. Microsoft Research Technical Report: MSRTR-2003-24. Presented in Microsoft VC Summit 2004, Silicon Valey, April 2004.
- Greenberg, A., Hamilton, J., Maltz, D. A., & Patel, P. (2009). The cost of cloud: Research problems in data centre networks. *ACM SIGCOMM Computer Communication Review*, 39(1). <http://ccr.sigcomm.org/drupal/files/p68-v39n1o-greenberg.pdf> (Jun 2011)
- Harms, R., & Yamartino, M. (2010). *The economics of the cloud*. Microsoft. Retrieved from <http://www.microsoft.com/presspass/presskits/cloud/docs/The-Economics-of-the-Cloud.pdf>
- Kenyon, C., & Cheliotis, G. (2002). *Architecture requirements for commercializing grid resources*. In 11th IEEE International Symposium on High Performance Distributed Computing (HPDC'02)
- Kenyon, C., & Cheliotis, G. (2003). Grid resource commercialization: Economic engineering and delivery scenarios. In Nabrzyski, J., Schopf, J. M., & Weglarz, J. (Eds.), *Grid resource management: State of the art and research issues*.
- Koomey, J., Brill, K., Turner, P., et al. (2008). *A simple model for determining the true total cost of ownership for data centres*. Uptime Institute. Retrieved from http://www.missioncriticalmagazine.com/Articles/White_Papers/BNP_GUID_9-5-2006_A_10000000000000199550
- Mansouri-Samani, M. (1992). *Monitoring distributed systems (a survey)*. Tech. Rep. DOC92/23, Imperial College, London, UK. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.45.4660&rank=2>
- McFedries, P. (2008, August). The cloud is the computer. *IEEE Spectrum Online*. Retrieved from <http://www.spectrum.ieee.org/aug08/6490>
- Mell, P., & Grance, T. (2009). *Effectively and securely using the cloud computing paradigm*. NIST Information Technology Laboratory.
- Rappa, M. (2004). The utility business model and the future of computing services. *IBM Systems Journal*, 43(1), 32–42. doi:10.1147/sj.431.0032
- Sosinky, B. (2011). *Cloud computing bible*. John Wiley and Sons.

KEY TERMS AND DEFINITIONS

Infrastructure as a Service (IaaS): One of the models for delivery of services in the Cloud, in this case focused on offering a virtual system that gives the appearance of acting as would an equivalent physical system, and upon which it is possible to develop a number of different platforms (see PaaS).

Platform as a Service (PaaS): One of the models for delivery of services in the Cloud, in this case focused on a specific set of mechanisms that allows for other software to be developed but which constrains the developer to use a specific approach or set of software tools in development.

Private Cloud: Typically, an organization-internal virtualized hardware system that provides similar function and management capabilities (e.g. for billing) as a Public Cloud would offer.

Public Cloud: The Cloud services (see SPI) typically offered by third party commercial companies such as Amazon, Google, Microsoft, Rackspace, and others, and usually associated with some kind of price plan.

Service Level Agreement (SLA): A set of terms and conditions, which may be written for human consumption or computationally formulated for machine consumption, that specifies how a service will operate and the set of actions a service provider will undertake in the event that they cannot satisfy that specification.

Software as a Service (SaaS): One of the models for delivery of services in the Cloud, in this case focused on software.

SPI [model]: The three most commonly referred to models of delivery of services in the Cloud, namely Software, Platform, and Infrastructure (as a Service). See: SaaS, PaaS, IaaS.

ENDNOTES

- ¹ http://csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf (Jun 2011)
- ² http://www.businessweek.com/technology/content/sep2008/tc2008095_942690.htm (Jun 2011)
- ³ <http://www.greenpeace.org/raw/content/usa/press-center/reports4/make-it-green-cloud-computing.pdf> (Jun 2011)
- ⁴ http://awsmedia.s3.amazonaws.com/Whitepaper_Security_Best_Practices_2010.pdf (Jun 2011)
- ⁵ http://opencloudmanifesto.org/Cloud_Computing_Use_Cases_Whitepaper-4_0.pdf (Jun 2011)
- ⁶ http://www.jisc.ac.uk/media/documents/programmes/research_infrastructure/cc421d007-1.0%20cloud_computing_for_research_final_report.pdf