Financial Software as a Service – A Paradigm for Risk Modelling and Analytics

Muthu Ramachandran and Victor Chang
School of Computing, Creative Technologies and Engineering,
Leeds Metropolitan University,
Leeds LS6 3QS UK
Email: M.Ramachandran@leedsmet.ac.uk & V.I.Chang@leedsmet.ac.uk

Abstract

Software as a service as one of the cloud delivery models that supports fine-grained components. Financial applications demand better performance and accuracy in a cloud than the traditional computing platforms. Therefore, designing financial software as a service (FSaaS) requires an engineering and systematic approach. This paper has proposed an integrated service-oriented architecture and a SaaS component model for financial domain that provides the required scalability, flexibility and customisation. We have also demonstrated the design and customisation of service component interfaces to a financial simulation so that it provides automatic prediction models for investors to know accurate results for buy and sale prices. Therefore, large-scaled simulations can be achieved within a matter of 13.5 second for outlier removal and within 9 seconds for high-performance risk computation on the Cloud. We show the holistic and complete approach of illustrating the system design of FSaaS, showing the two major algorithms and the results of experiments of running these two algorithms. We provide plans to integrate new and existing services with FSaaS.

Keywords: Software as a Service (SaaS); Financial Software as a Service (FSaaS); Financial Clouds; Monte Carlo Methods; Monte Carlo Simulations; Black Scholes Model; Financial Software as a Service (FSaaS); Variance-Gamma Processes (VPG); MATLAB SaaS applications on Clouds; programming methods for Cloud Computing; enterprise portability for Clouds.

1. Introduction

Global economic downturn caused by the financial sector is an interdisciplinary research problem which requires that experts from different sectors work altogether. The problem itself is complex and involved with a number of different causes. Firstly, Lord Turner, Chair of the UK Financial Service Authority (FSA), is quoted as follows: "The problem, he said, was that the banks' mathematical models assumed a 'normal' or 'Gaussian' distribution of events, represented by the bell curve, which dangerously underestimated the risk of something going seriously wrong." (Financial Times, June 2009). Secondly, there were reports of a lack of regulations on financial practices. Remedies have been proposed by several governments to improve on this (Financial Times, 2010; City A.M, 2010). Thirdly, there was the "Madness of Mortgage Lenders" as identified in a study conducted by Hamnett (2009) whereby uncontrolled lending to those who could not afford to repay, that led to a housing bubble and subsequent collapse. Hamnett (2009) concluded that irresponsible mortgage lending was a key factor in the collapse of Lehman Brothers and a number of banks which seemed to trigger the global financial crisis. Fourthly, MacKenzie and Spears (2010) conducted interviews and in-depth study in this subject and concluded that the cause of the problem was due to the ease of adopting an easy-to-use mathematical formula, Gaussian

Copula, in which the traders have been misused and abused the formula for massive investment. MacKenzie and Spears (2010) asserted further that the founder of Gaussian Copula, Dr David X Li, was related to the cause of the financial crisis. Their argument was that if he knew the formula has limitations, he should not promote it even remedies and warnings were done later on.

Therefore, identifying a solution to any financial crisis requires a holistic approach to problem solving and accurate prediction model for the financial crisis. This involves accurate mathematical simulation models which are discussed in this paper and have been used in practice for large-scaled financial simulations. The aim is to make all these calculations as accurate as possible, while considering and using a number of reliable formulas to check that results are consistent with each other. Financial services should be transparent and its activities such as risk modelling and analysis should follow a more scientific and rigorous steps in ensuring the accuracy, performance, security, usability and scalability can be achieved. In our previous work, we demonstrate that the use of Financial Clouds and Financial Software as a Service (FSaaS) can meet those objectives (Chang et al., 2011 a; 2014 a; 2014 b). An alternative solution is to deploy Business Intelligence as a Service to model pricing and risk which require real time and vigorous approaches to compute values of prices with their associated risks required for the investment and decision-making process.

As discussed in the last paragraph, four major factors contributed to complexity that caused global downturn. An alternative to allow experts of different disciplines working together is to have a platform such as Cloud Computing, which can offer innovative approaches for risk analysis, and knowledge sharing in a community-oriented culture (Feiman and Cearley, 2009). Cloud resources can be used to improve accuracy of risk analysis, financial modelling and knowledge sharing in an open and professional platform (Buyya et al. 2009; Martson et al, 2010; Chang et al, 2011 a; 2014 a; Chang 2014 a). To support this concept, there are demonstrations presented by authors to confirm the added values of Cloud adoption, particularly the finance sector and organizations that deploy business intelligence. Benefits include the improvement in efficiency, collaboration, revenue, cost-savings and service rating in healthcare, finance and education sectors as a result of Cloud adoption (Buyya et al. 2009; Martson et al, 2010; Chang et al. 2013; 2014 a; Chang 2014 a; 2014 b). The extended rationale for providing added value for finance is as follows: Clouds provide a common platform on which to run different modelling and simulations based on Gaussian and non-Gaussian models. The Clouds then offer the distributed high-performing resources for experts in different areas within and outside financial services to study and review the modelling together, including models using Monte Carlo Methods and Black Scholes Model. Complex risk simulations can be presented in the form of visualisation, so that any unexploited area due to the lack of understanding about risk can be presented to the stakeholders and investors with ease. The Clouds allow regulations to be undertaken transparently in parallel with results in risk modelling and visualisation while establishing and strengthening security and policy within the Clouds resources.

There are different types of clouds in the market for different applications and services, with the lack of Cloud Computing services for finance. It is apparent from the relative lack of existing literature on the subject that there has been little academic research into Financial Clouds (FC). Based on literatures and expert reviews (Chang et al 2010 a; 2010 b; 2011 a; Chang 2014 a), we have identified several reasons for this. Firstly, a majority of financial practices are closed-source, since this relates to the way they make profits and business opportunities, and sharing this type of information will be undesirable within a business context (Bryan T, 2009). Secondly, human decision makers can overrule any computing analysis of risks and even introduce excessive risk taking that can result in adverse effects as illustrated between 2008 and 2009 (Flouris and Ylimaz, 2010). Thirdly, despite advanced technologies being introduced, many financial practices still use desktop-oriented tools such as Excel and VBA together with desktop based statistical software such as SAS. A few use Grid technologies, and of among them, not all will use Clouds (Chen, Chee, Huang, Jin, Tseng, Wang and Wong, 2010, by interviews).

We propose Financial Software as a Service (FSaaS) as a holistic approach to provide services to calculate risks, inform the stakeholders about the rapid change in the risk and deal with issues related to four factors causing financial crisis as discussed above. In order to demonstrate our approach, this paper introduces important concepts, presents the service prototype and explains the FSaaS solution from the system design to implementation, as well as the background theories and interpretations of analysis. The breakdown of this paper is as follows. Section 2 explains the system design and relevant information for the FSaaS. Section 3 presents the financial models behind FSaaS, the theories, two major algorithms involved in the development of FSaaS, as well as experiments and results of running these two major algorithms. Section 4 describes two topics for discussion. Section 5 sums up this paper with future work planned.

2. Background for Financial Software as a Service

Financial applications require on-demand services that are offered by cloud computing with cost-benefits. Capegemini Analysis report (2011) emphasises cloud services adoption by financial services institutions is expected to increase with an IT spending of US$21.9 billion in 2012.Therefore, financial domain has begun to reap these bene-fits with emerging financial SaaS such as FinancialForce which is developed by SalesForce, NetSuite, Intacct, and Oracle's financial SaaS. NetSuite (2014) claims to have helped companies increase their revenues by 95% by moving to financial SaaS. The cloud technology analysis reports such as by Accenture (2011) & Capegemini (2011) on financial technology trends and high performance computing predicts:

- Leveraging technology to address new & change in regulations
- Reliable and globally harmonised financial systems
- Add value through strategic applications
- Harvest benefits from technology
- Cost-savings and usage-based billing (pay-per-use basis)
- Business continuity, and business agility and focus
- Green IT

Accenture (2011) report also claims SaaS to be a simple, efficient, engaging, accessible, clear structure, intuitive, and supportive. With keeping this set of requirements as design criteria, this paper has designed a SaaS component model and a service architecture supporting required flexibility, scalability, and allowing change in environment. Cloud technology offers financial institutions with more flexible business models that lower operational costs. However, the key to success is based on choosing the right cloud services (SaaS) that provides higher ROI and matches to their core business value. Therefore more emphasis and investment should be given for business process modelling to analyse and simulate the business process and expected business performances before implementing financial cloud services (SaaS). Our earlier work (Ramachandran 2012; 2013) in this area has provided a clear process for developing cloud services systematically.

The next main challenge is to identify and analyse types of cloud service models that fit financial domains (Banking, Investment, Taxes, Real-Estate, Insurances, etc). The types of cloud services proposed by Capegemini analysis report (2011) includes:

1. Business Process-as-a-Service (BPaaS). This type of cloud service delivery model is also known as on-demand business software as a service and is used for standard business processes such as billing, payroll, or human resources. BPaaS could also combine all the other service models such as SaaS, PaaS, and IaaS to provide the service orchestration and choreography with business

process expertise. This type of services laid on the business layer of the cloud architecture and is discussed in a later section as shown in Figure 2.

2. Software-as-a-Service (SaaS). This type of cloud service delivery model is also known as on-demand software as a service and is to provide software applications as a service along with relevant data and is hosted centrally by the cloud service providers. A cloud service provider houses the business software and related data, and users access the software and data via their web browser. Types of software that can be delivered this way include office applications (email and documentation), virtual desktop, games and entertainment applications, accounting, customer relationship management (CRM), enterprise resource planning (ERP), invoicing, human resource management (HRM), content management, and service desk management. This type of services laid on the SaaS layer of the cloud architecture.
3. Platform-as-a-Service (PaaS). This type of cloud service delivery model is also known as on-demand computing platform as a service and is to provide a complete solution stack and is hosted centrally by the cloud service providers. A solution stack, in computing, is a set of software subsystems or components needed to perform a task without further external dependencies. An example of a solution stack for a Linux based includes LAMP software bundle (a Linux operating systems, Apache (the web server), MySQL or MariaDB (the database management systems), Perl, PHP, or Python (scripting languages). Another example of a Linux based solution stack is OpenStack which includes Linux – OpenStack controller nodes run exclusively on Linux and OpenStack – providing an infrastructure as a service (IaaS). The other well known Linux-based solution stacks include LYME, GLASS, LEAP, Ganeti, XAMPP a cross-platform {X (operating system), Apache (web server), MySQL or MariaDB (database), PHP (programming language), Perl (programming language)}, MAMP, WAMP, WISA, and MEAN, etc. PaaS offerings facilitate the deployment of applications without the cost and complexity of buying and managing the underlying hardware and software and provisioning hosting capabilities. A cloud service provider offers a complete platform for application, interface, and database development, storage, and testing, execution and runtime environment, database, web servers, and development tools. This allows businesses to streamline the development, maintenance and support of custom applications, lowering IT costs and minimizing the need for hardware, software, and hosting environments. This type of services laid on the SaaS layer of the cloud architecture.
4. Infrastructure-as-a-Service (IaaS). This type of cloud service delivery model is also known as on-demand infrastructure as a service and is to provide software infrastructures, virtual machines, networked devices, platforms, required SaaS and PaaS along with relevant data and is hosted centrally by the cloud service providers. Rather than purchasing servers, software, data centre space or network equipment, this cloud model allows businesses to buy those resources as a fully outsourced service. This type of services laid on the SaaS layer of the cloud architecture and includes virtual machines, servers, storages, load balancers, networked devices, and SaaS, PaaS.

The integrated service-oriented architecture and SaaS component have been designed and implemented for FSaaS in this paper. The aim is to provide required scalability, flexibility and customisation that are at the heart of a financial SaaS. We demonstrate the system design and the implementations. Examples include the results of running software that can compute call and put prices. Results of simulations also support the fact that all computations can be completed within seconds. To illustrate our key messages, the breakdown of this paper is as follows. This paper also presents the overall view about SOA approach an FSaaS application and discusses the service component model. This paper also shows the results of running FSaaS and large-scaled simulations in seconds. The final section sums up issues covered and provides conclusion.

2.1 SOA APPROACH FOR FSaaS APPLICATIONS

As demonstrated in our previous work (Chang et al., 2011 a; 2014 b), FSaaS applications require open, flexible, interoperable, collaborative and integrated architecture to provide services to integrate various stakeholders such as citizens and financial services (e-shares, e-stockbrokerage, e-fund-management, QoS, FSaaS cloud simulation services, e-health, e-tax, e-national security, e-pension, e-payment, e-education and training, e-work and employment, e-funding, etc), business organisations and vendors (e-procurement), and government agencies (both inter and intra governments). Therefore, designing architecture for FSaaS applications pose a design challenge. Software architectural design needs to be verified, assessed and evaluated for its quality before its development. There are well known generic design criteria such as flexibility, maintainability, testability, reusability, etc. The key to achieving good architectural quality for the system being developed is to extract major characteristics of this system from various sources such as non-functional requirements, customer requirements, existing systems, experts, research literatures (Baresi, Di Nitto, and Ghezzi 2006; Cartwright and Doernenburg 2006; Erl 2005; Open Group 2009; Sellami and Jmaiel 2013 Ramachandran 2012-13) and so on. This paper identifies such characteristics from various perspectives supporting emerging technologies which are shown in Figure 1. These FSaaS application characteristics are the backbone for developing service-oriented architecture and components.

Figure 1 Characteristics of FSaaS Applications

To further expand on concepts in Figure 1, FSaaS applications need to work and interface with multi-platform and multi-vendor applications and therefore this needs to be designed for interoperability (Taher et al., 2012). Multi-channel delivery of services has been recommended to deliver FSaaS services which is interoperable, data integrity, flexible, scalable, multi-portal, 3rd party application integration, open standard applications, secured, resource-managed, SLA, available, high performance, accurate, and

collaborative, through various and emerging communication channels such as mobile phones, digital TV, cloud services, call centres, kiosks, emails, PCs, teleconferencing, e-gov apps, web, and webinars - some of these are not been used by any FSaaS at present. Each characteristic shown in Figure 1 represents not only the application attributes and it also provides a set of key design criteria and quality attributes for developing architecture that support emerging technologies for FSaaS. For example, the design criteria of interoperability are essential for FSaaS applications as they are multi-platform and multi-channel based (Taher et al., 2012). Interoperability is supported in the service-oriented architecture design for FSaaS applications by means of a service bus and loose coupling of the other components which is shown in Figure 2.

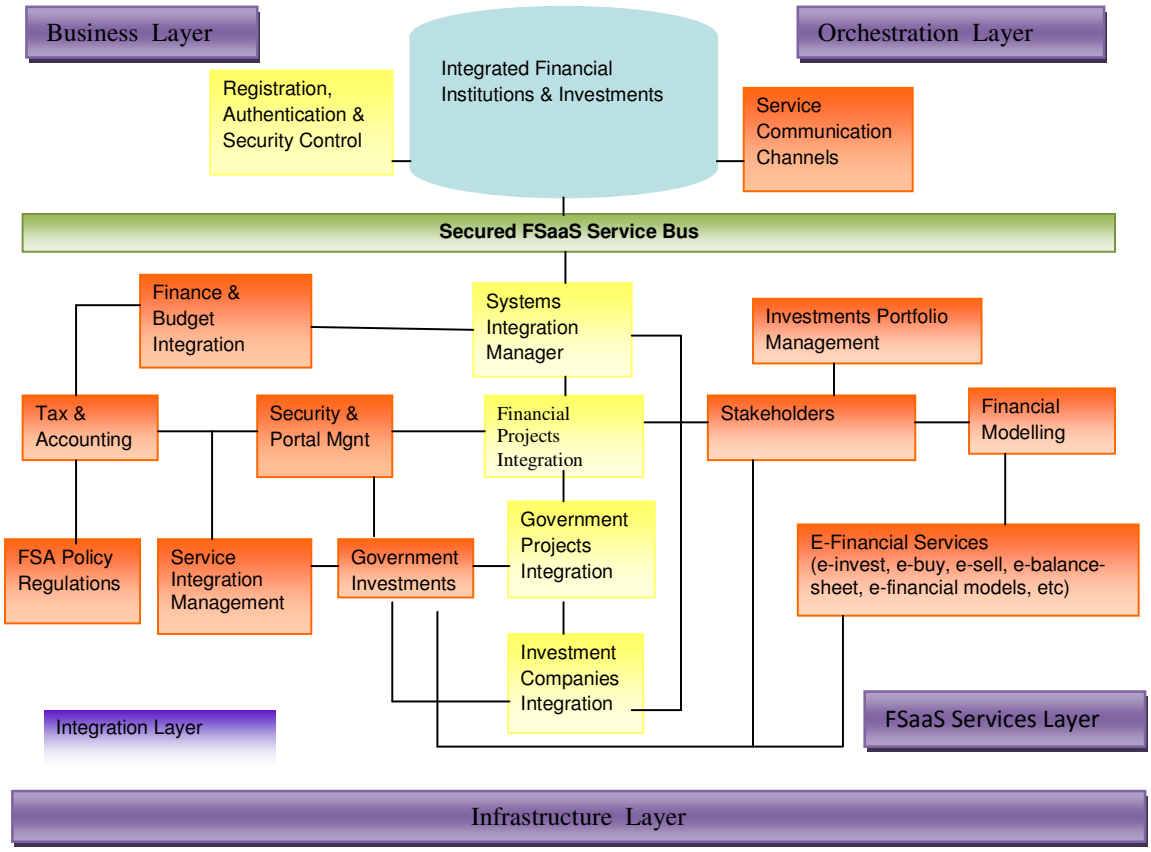

Figure 2 Service-Oriented Architecture for FSaaS

FSaaS is a framework for designing and implementation of Could Computing solutions. The emphasis is to show how FSaaS can help to address portability in Cloud Computing implementations in Finance domain. Portability involves migrating entire applications from desktops to clouds and between different Clouds in a way which is transparent to users so they may continue to work as if still using their familiar systems. Reviews for several financial models are studied, where Monte Carlo Methods (MCM) and Black Scholes Model (BSM) are chosen to demonstrate portability between desktops and clouds. A special technique in MCM, Variance-Gamma Process, is used for error corrections while performing analysis of good quality. Coding algorithm for MCM and BSM written in MATLAB are explained. Simulations for MCM and BSM are performed on different types of Clouds. Benchmark and experimental results are presented and discussed, together with implications for banking and ways to

track risks in order to improve accuracy. We have used a conceptual Financial Cloud platform to explain how this fits into the FSaaS, as well as Financial Software as a Service (FSaaS). Our objective is to demonstrate portability, speed, accuracy and reliability of applications in the clouds, while demonstrating portability for FSaaS. The modelling and simulation methods mentioned in this section are handled within the service-oriented architecture for FSaaS by the sub-system block labelled as FSaaS services layer in our model shown in Figure 2.

These services can be made available as stand alone, integrated, componentised, web based service component, composite service (a set of interconnected services), virtualised services (cloud based), and dynamically re-configurable services. The architecture presented in this paper is then based after a critical review and analysis of a number of existing architectures for FSaaS applications. As mentioned before, the SOA based architecture consists of four distinct levels of abstraction layers which are connected and communicated by messages through a core communication channel known as a service bus or a central bus. These layers are: 1) a business layer with a dedicated set of services, 2) an orchestration layer with a set of services where new services can be composed, 3) an FSaaS layer that supports integration of services, government departments and local governments, and 4) an e-business layer that supports new businesses and integration of data. The SOA based architecture for FSaaS services, then ensures that it achieves the expected service-oriented design factors such as customisation, cost-effectiveness, availability, etc.

Referring to Figure 2, at the business and orchestration layers provide high level service composition based on new business perspective and policies (both political and economical factors). Mostly, the customisation and the new business needs arise from these two key variables. The sub-systems such as registration control, security control, integrated services for FSaaS applications control, and communications channels help to achieve customisation at a higher level of abstraction without affecting underlying business logic services. These are communicated and connected to layers below using a concept of service bus known as FSaaS secured service bus. The layer below the business layer provides services for various FSaaS departments, and external suppliers (E-Business layer). The financial modelling subsystem provides link to FSaaS service components and handles various simulation models described in the last part of this paper.

2.2 SERVICE COMPONENT MODEL FOR FSaaS APPLICATIONS

Designing cloud services based on software components allows maximum flexibility, scalability, and elasticity and are required for a cloud service. More importantly, service components allow reuse, customisation, and service composition on-the-fly. Some of the main reasons for the emergence of service components is for customisation through interfaces, supporting reuse through extensibility by applying building-block concepts, and interoperability, for distributed cloud components. Service level components should support communication and exchange of messages to different systems and services on-the-fly, and therefore, componentising services will satisfy those criteria. Web services and SOA have been well established in the past few years with new technologies and architectures supporting service-oriented paradigms explicitly in the process, and they have also been proven to be a good design model. This section discusses service component models for cloud services and also discusses the design rationale. Cloud applications development should primarily focus upon user perspective, their risks, and the design and architectural models should reflect user needs and their risks.

Component models and their architecture provide a framework for system composition and integration. A generic component model that is presented in this article provides a unique concept of two distinct set of services: provide and requires. Software components are the basic unit of artefact that supports service composition with the cloud computing architecture and its environment. However, each development paradigm and application demands customisable and extendable component architectures that suit the needs of their applications. Each Web service component interfaces are mapped onto different ports within architectural layers to the request for services and offer services as, and when,

required at run-time. Service component architecture (SCA) is a set of specifications which describes a model for building applications and systems using a service-oriented architecture (SCA 2014). SCA extends the approaches on software components and builds on open standards such as Web services. This section proposes an approach to developing FSaaS applications which is based on developing a financial service component which has provided required customization extensibility. As stated before, FSaaS applications require open, flexible, interoperable, collaborative and integrated architecture to provide services. These services can be made available as stand alone, integrated, componentised, web based service component, composite service (a set of interconnected services), virtualised services (cloud based), and dynamically re-configurable services. This vision is similar to the Open Group's Service Integration Maturity Model (OSIMM) 2013. The OSIMM provides:

- A process roadmap for attaining key practices with metrics
- Seven levels of maturity to improve
- A quantitative model for assessing current practices and to improve with recommended practices

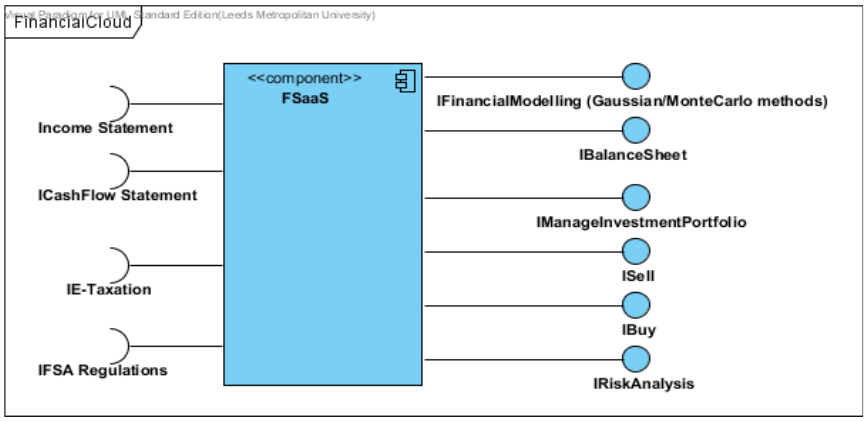

Figure 3 FSaaS Service Component Model

The aim is to map business requirements onto a service component that can be designed and implemented. A service component can be defined as the one that configures a service implementation. A service component model (UML based service model) is shown in Figure 3 which reflects the service component design principle with a number of plug-in type interfaces that allows it to connect other service components. As shown in Figure 3, a design of software as a service component model for financial applications (FSaaS) where it shows two types of services such as provide (lollypop notation) and require services (arc notation and the naming convention starts with "I", a version-naming convention used for FSaaS). Lollypop notation is used to compose and connect services in the component model. For example, FSaaS services include Income statement, ICashFlow statement, Ie-taxation, IFSA regulations). IFSA refers to providing interface service integration for Financial Authority regulations which any investment service providers should adhere to and is flexible for change in their regular change thus providing required scalability and flexibility of FSaaS characteristics discussed earlier.

Similarly, providers service interfaces are Ifinancial modelling, IBalancesheet as a service which is automatically produced from income and cashflow statements. The other provider interfaces include ImanageInvestment Portfolio, Isell, IBuy, and IRiskAnalysis services. In order to achieve the recommended process and key practices, we need to have interoperability, reconciliations and resiliency between systems, where interface linkages are appropriate to other services such as those of emerging technologies interfaces (e.g. Ie-taxation linked to E-Gov applications, and IFSA regulation service linked to FSA). Similarly, such reconciliations services must be automated for the sake of cost-efficiency. To

develop an integrated FSaaS service-oriented architecture system, it is critical that service analysts and software engineers identify and address the adoption challenges when implementing e-government services. Research questions for FSaaS adoption challenges are listed below:

- How to adopt a new process and to identify the scope of the business services to be developed and supported?
- How data integrity will be achieved and secured?
- How data will be protected when needed to support management decisions?
- How systems will fit together to support service choreography and orchestration layers?
- How systems will fit together to support the enabling emerging technologies (system architecture needs to be service-oriented thereby any new emerging technologies in the future should easily be integrated more cost-effectively than with traditional system architectures)?
- How data and services will be safeguarded and secured to ensure the integrity and re-configurability of service operations and personal data (information assurance)?
- How new services can be composed and re-configured at run-time?

The service component modelling and design provides a systematic approach to building cloud service components to allow on-the-fly configuration, to discover new business services, and to be able to connect and disconnect service compositions. Service composition is one of the key principles of service design which can't be achieved without a component-based approach. The design principle of component interface allows service flexibility, elasticity, and scalability. A service composition is defined as the development of customised services by discovering, integrating, and executing existing services. Design of service composition is not only to consume services and also to provide services. Cloud service orchestration layer and its principle can also be addressed and achieved using service composition when services are designed as components based on the model as shown here.

Service composition and orchestration allows service level reuse to happen. Service reuse is a notion of designing services as generic as possible to be reused in another service invocation. Designing services for reuse is based on SOA design principles:

- Loose coupling is to limit dependency between service consumers and service providers. This can be achieved by service interface design which has been part of a service component model as discussed.
- Autonomy is the key principle that enables service reuse. This can be achieved by designing services that can manage their own resources as database and legacies and to maintain by themselves without depending on other services. Service autonomy facilitates service adoption, scalability, QoS, SLA, and virtualisation.
- Statelessness is the property of a service to have a context but it will not have any intermediary state waiting for an event or a call-back.
- Granularity has been a prominent design principle of reuse. A large granularity of service component which is self-autonomous can yield higher level of service reuse through service composition. However, a balance must be struck when designing service components and interfaces.
- Composability is the process by which services are combined and integrated to provide a comprehensive and composite service. This principle is also the key to achieving cloud orchestration. A composite service consists of an aggregation of services that can produce another reusable service (s).

- Discoverability is an important means of mandating service time (design time reuse and runtime discoverability) notion when designing service components so that component can be called on when required. Service component interface concept allows components to be discovered and connected.

Designing reusable services can save cost as it has been a well-known benefit of reuse. Cost reduction is one of the key aspects of cloud computing which aim to reduce cost for consumers by allowing pay-per -use cost model. The design rationale and service component model discussed in this section will help to improve cloud service reuse experiences.

This paper has addressed some of these issues such as achieving business process using BPMN for identifying FSaaS component services, management decision are integrated with service component interfaces as governance, SOA architecture model has been designed to illustrate other issues. Currently, we are working on developing a WSDL to link to relevant MATLB code or any other simulation services as shown in Table 1 (discussed in the following section).

3. Demonstrations and simulations for FSaaS

This section demonstrates how FSaaS can be used and explains the code and results during and after using FSaaS services. This mainly involves Monte Carlo Methods (MCM) and Black Scholes Model (BSM). Chang et al. (2011 a) describe how to use MCM and BSM for financial computation. They demonstrate the use of risk analysis and financial modelling on Clouds based on MATLAB and Mathematica, which offer benefits such as performance, accuracy and integration with security. This includes the selection of Linear Square Method (LSM) that can compute 100,000 simulations in one go, which takes between 4 to 25 seconds depending on the number of time steps. It can also work with IBM Fined-Grained Security Model (IBM 2011), and it can provide a safer environment for FSaaS on Clouds. IBM fine-grained security model is part of the IBM Business Monitor tool which provides automated support for filtering security data, controlling security settings by authorised personal and monitoring security activities (*security filtering features*). It allows users and developers customise filtering based on regions. It also offers users and developers to add new and delete security rules. It permits users and developers to control the display of data metrics through *object filtering features*.

3.1 Financial Models

Many financial variables pose several challenges to our economy and our day-to-day lives. These financial variables include sale price for a business, cost of living, profits for investors, asset returns in international stock, and bond markets or interest rates in the liquidity market. All these pose unconditional modalities and time changing volatility. Therefore, we need a predictable solution based on mathematical accuracy. In this context, we have been successful in using Bayesian prediction models, linear models, nonlinear models, Gaussian and non-Gaussian models, and other models in practice. Gaussian-based mathematical models have been frequently used in financial modelling (Birge and Massart, 2001). As pointed out by the FSA, many banks' mathematical models assumed normal (Gaussian) distribution as an expected outcome, and might underestimate the risk for something going wrong. According to Hutchinson (2010), "The Gaussian model is too optimistic about market stability, because it uses an unrealistically high number for the key variable, the exponential rate of decay, known to its friends as alpha". To address this, other non-Gaussian financial models need to be investigated and demonstrated for how financial SaaS can be successfully calculated and executed on Clouds. Based on the various studies (Feiman and Cearley, 2009; Hull 2009), one model for pricing and one model for risk analysis should be selected respectively. A number of methods for calculating prices include Monte Carlo Methods (MCM), Capital Asset Pricing Models and Binomial Model. MCM is often used in stochastic and probabilistic financial models, and provides data for investors' decision-making (Hull, 2009) and is our choice for MCM for pricing. On the other hand, methods such as Fourier series, stochastic volatility and Black

Scholes Model (BSM) are more appropriate for volatility. As a main stream option, BSM is selected for risk analysis in this paper as BSM has finite differential equations to approximate derivatives.

3.1.1 Monte Carlo Methods in Theory

A Monte Carlo simulation approach have been used to approximate and solve the optimisation problem and also to evaluate the portfolio risk-level in presence of parameter estimation errors or mis-specified tails behaviour. Monte Carlo Simulation (MCS), originated from mathematical Monte Carlo Methods, which is a computational technique used to calculate risk analysis and the probability of an event or investment to happen. MCS is based on probability distributions, so that uncertain variables can be described and simulated with controlled variables (Hull 2009; Waters 2008). Many models have been used to describe financial time series that satisfies stochastic differential equation (a continuous time diffusion behaviour) given below. Originated from Physics, Brownian Motions follow underlying random variables can influence the Black-Scholes models, where the stock price becomes

$$dS = \mu S\,dt + \sigma S\,dW_t. \tag{1}$$

where W is Brownian—the dW term here stands in for any and all sources of uncertainty in the price history of the stock. The time intervals are divided into M units of length δt from time 0 to T in a sampling path, and the Brownian motion over the interval dt are approximated by a single normal variable of mean 0 and variance δt, and leading to

$$S(k\delta t) = S(0)\exp\left(\sum_{i=1}^{k}\left[\left(\mu - \frac{\sigma^2}{2}\right)\delta t + \sigma c_i\sqrt{\delta t}\right]\right) \tag{2}$$

for each k between 1 and M, and each is a draw from a standard normal distribution. If a derivative H pays the average value of S between 0 and T then a sample path ω corresponds to a set $\{c_1, \ldots, c_M\}$ and hence,

$$H(\omega) = \frac{1}{M+1}\sum_{k=0}^{M} S(k\delta t). \tag{3}$$

The Monte Carlo value of this derivative is obtained by generating N lots of M normal variables, creating N sample paths and so N values of H, and then taking the average. The error has order

$\epsilon = \mathcal{O}\left(N^{-1/2}\right)$ convergence in standard deviation based on the central limit theorem.

Reibero and Webber (2002) demonstrate improved calculation techniques based on Monte Carlo Methods (MCM) on top of the Variance-Gamma (VG) Process, which has been a subject of studies by researchers (Carr et al., 2002; Reibero, Webber, 2002). They explain stratified sampling method and how to stratify VG bridge. They have benchmarked the methods with European options (a finance model). However, Reibero and Webber do not provide any details of hardware and software environments used for benchmarking, and there is no information for their coding algorithm. To demonstrate Variance-Gamma, we have opted for Asian options with 10,000 MCM simulations. They perform this experiment using a desktop environment, two private clouds and one Amazon EC2 public cloud as the proof of concept and benchmarking. The purpose of demonstration is not about the finance model; either European or Asian, but it is about presenting a systematic and logical proof of concepts. In another paper published by Reibero and Webber (2004), they explain that there is simulation bias in MCM for financial options including VG process.

3.1.2 Monte Carlo Methods for Variance Gamma Process

Reibero and Webber (2002) demonstrate improved calculation techniques based on Monte Carlo Methods (MCM) on top of the Variance-Gamma (VG) Process, which has been a subject of studies by researchers (Carr et al., 2002; Reibero, Webber, 2002). They explain stratified sampling method and how to stratify VG bridge. They have benchmarked the methods with European options (a finance model). However, Reibero and Webber do not provide any details of hardware and software environments used for benchmarking, and there is no information for their coding algorithm. To demonstrate Variance-Gamma, we have opted for Asian options with 10,000 MCM simulations. They perform this experiment using a desktop environment, two private clouds and one Amazon EC2 public cloud as the proof of concept and benchmarking. Details are described in Section 4. The purpose of demonstration is not about the finance model; either European or Asian, but it is about presenting a systematic and logical proof of concepts. In another paper published by Reibero and Webber (2004), they explain that there is simulation bias in MCM for financial options including VG process.

3.1.3 Monte Carlo Methods for Least Squared Method

Least Square Method (LSM) provides a direct method for problem solving, is appropriate for large problems and it lends itself to rapid calculation in the Cloud because its computation can be divided into sections which can be calculated independently (Moreno and Navas, 2001; Chang et al., 2011; Longstaff and Schwartz, 2011). Robust algorithms have been developed which estimate best values efficiently and precisely using LSM in combination with MCS which are popular and versatile (Moreno and Navas, 2001; Choudhury et al., 2008; Longstaff and Schwartz, 2011). The theory is as follows.

Consider a data set $(x_1,y_1), (x_2, y_2),....,(x_n, y_n)$ with the fitting curve $f(x)$ has the deviation $d_1, d_1, ... , d_n$ caused by each data point, the least square method produces the best fitting curve with the property as follows

$$MinimumLeastSquareError(\Pi) = d_1^{\,2} + d_2^{\,2} + ... + d_{n-1}^{\,2} + d_n^{\,2} = \sum_{i=1}^{n} d_i^{\,2} = \sum_{i=1}^{n} [y_i - f(x_i)]^2 \qquad (4)$$

The least squares line method uses an equation $f(x) = a + bx$ which is a line graph and describes the trend of the raw data set $(x_1,y_1), (x_2, y_2),....,(x_n, y_n)$. The n should be greater or equal to 2 ($n \geq 2$)in order to find the unknowns a and b. So the equation for the least square line is

$$\Pi = d_1^{\,2} + d_2^{\,2} + ... + d_{n-1}^{\,2} + d_n^{\,2} = \sum_{i=1}^{n} d_i^{\,2} = \sum_{i=1}^{n} [y_i - (a + bx_i)]^2 \qquad (5)$$

The least squares line method uses an equation $f(x) = a + bx + cx^2$ which is a parabola graph. The n should be greater or equal to 3 ($n \geq 3$)in order to find the unknowns a, b, *and* c. When you get the first derivatives of $\prod$ in parabola, you will have

$$\Pi = d_1^{\,2} + d_2^{\,2} + ... + d_{n-1}^{\,2} + d_n^{\,2} = \sum_{i=1}^{n} d_i^{\,2} = \sum_{i=1}^{n} [y_i - (\sum_{i=1}^{n} a + \sum_{i=1}^{n} bx_i + \sum_{i=1}^{n} cx^2)]^2 \qquad (6)$$

LSM has been mathematically proven and allows advanced calculations of complex systems. $f(x) = a + bx + cx^2$ is the equation for LSM. LSM provides a direct method for problem solving, and is extremely useful for linear regressions. LSM simulates and performs calculations by linear regression, which attempt to fit to the parabolic function to get a precise approximation to the actual values closely. LSM computation can either use data or mathematical predictive modelling (no data).

3.1.4 Black Scholes Model (BSM)

The BSM is commonly used for financial markets and derivatives calculations. It is also an extension from Brownian motion. The BSM formula calculates call and put prices of European options (a financial model) (Hull, 2009). The value of a call option for the BSM is

$$C(S,t) = SN(d_1) - Ke^{-r(T-t)}N(d_2) \tag{7}$$

where $d_1 = \frac{\ln(\frac{S}{K}) + (r + \frac{\sigma^2}{2})(T-t)}{\sigma\sqrt{T-t}}$ and $d_2 = d_1 - \sigma\sqrt{T-t}$.

The price for the put option is

$$P(S,t) = Ke^{-r(T-t)} - S + (SN(d_1) - Ke^{-r(T-t)}N(d_2)) = Ke^{-r(T-t)} - S + C(S,t). \tag{8}$$

For both formulas (Hull, 2009; Lee et al., 2010),

N(•) is the cumulative distribution function of the standard normal distribution

T - t is the time to maturity

S is the spot price of the underlying asset

K is the strike price

r is the risk free rate

σ is the volatility in the log-returns of the underlying

3.2 The first major algorithm for outlier removal

Section 3.1 presents the use of Monte Carlo Methods (MCM) including the two different techniques in Variance Gamma Process (VGP) and Least Square Method (LSM), as well as Black Scholes Model (BSM). VGP is used to remove outliers and LSM is deployed to optimise performance while the main code in MCM is aimed to calculate risk (Chang et al., 2011 a; 2014 b). BSM can be used to quantify risk yet the strength is on providing results in visualisations to make unexploited areas in risk analysis being exposed as identified in our previous work (Chang et al., 2011 a; Chang 2014 a). All these methods are essential to the development of FSaaS in producing accurate results with a good performance. Another challenge has arisen in organisational adoption of Cloud Computing. While other Cloud services and researchers do not release core code algorithm, it is not easy to check whether their proposed work can be reproduced in another environment. The ability to reproduce similar results is important for Science and also organisational Cloud adoption, or adopters and investors will not be able to support plans of providing Cloud Computing services such as FSaaS or other types of applications.

3.2.1 Preliminary setup and test

This section describes how the preliminary step to setup and test results with MCM simulations. Mathematical models such as MCM are used in Risk Management area, where they are used to simulate the risk of exposures to various types of operational risks. Monte Carlo Simulations (MCS) in Commonwealth Bank Australia are written in Fortran and C#. Running such simulations may take several hours or over a day due to the large scale simulations and complexity involved (Chang et al. 2011 a; Chang 2014 a). The results may be needed by the bank for the quarterly reporting period. MCM is suitable to calculate best prices for buy and sell, and provides data for investors' decision-making (Chang et al. 2011 a; Chang 2014 a). MATLAB is used due to its ease of use with relatively good speed. While the volatility is known and provided, prices for buy and sale can be calculated. Part of the code

(fareastmc.m) to is used to present formulas in MCM and demonstrate coding algorithm presented in Table 1.Variables are explained in the FSaaS demonstrations (Chang et al., 2011).

Table 1: Coding algorithm in Monte Carlo in MATLAB for best buy/sell prices

```
dt=T/(NSteps-1);
vsqrdt=sigma*dt^0.5;
drift=(r-(sigma^2)/2)*dt;
x=randn(NSimulations,NSteps);
Smat=zeros(NSimulations,NSteps);
Smat(:,1)=S;
for i=2:NSteps,
  Smat(:,i)=Smat(:,i-1).*exp(drift+vsqrdt*x(:,i));
end
```

The following demonstrates running the code and the calculated prices. Call prices are to buy and put prices are for sale. The program calculates the lower limit, the MCPrice value and the upper limit for each buy and sale category.

> fareastmc (this is the name of the MATLAB file)
[LowerLimit MCPrice UpperLimit]
Call Prices: [4.196694 4.248468 4.300242]
Put Prices: [7.610519 7.666090 7.721662]

3.2.2 Core Code in Variance Gamma Process (VGP)

This section presents the core code in Variance Gamma Process (VGP), a technique in MCM to removal outlier to ensure that the quality of calculations is as accurate as possible. The following shows the initial part of the code, where key figures such as maturity, volatility and risk free rate are given in Table 2.

Table 2: The first part of coding algorithm for Variance-Gamma Processes

```
S=100; %underlying price
K=101; %strike
T=0.5; %maturity
sigma=0.12136; %volatility for VG model
r=0.1; %risk free rate
VG_nu=.3; %nu for VG model
VG_theta=-0.1436; %theta of VG model
nsimulations=10000; % no. of MC simulations
k=4; %2^k: the no. of resets, Asian option
nsimulations=(floor(nsimulations^.5))^2;
tmpdim=nsimulations^0.5;
omega=(1/VG_nu)*( log(1-VG_theta*VG_nu-sigma*sigma*VG_nu/2) );
```

In MATLAB, a function, gamrnd, is to calculate variance gamma model, presented in Table 3.

Table 3: The second part of coding algorithm for Variance-Gamma Processes

```
thmean=T;
thvar=VG_nu*T;
theta=thmean/thvar;
alpha=thmean*theta;
G(:,n+1)=gamrnd(alpha,theta,nsimulations,1);
subplot(5,1,1);
subplot(2,2,1);
hist(G(:,n+1),100);
title('original gamma vars');
```

The third part of the code is to calculate stratified gamma sampling presented in Table 4. This includes replicate and tile array (repmat function) and returns the inverse of the gamma cumulative distribution function (cdf) (gaminv function).

Table 4: The third part of coding algorithm showing stratified gamma sampling

```
vvec=rand(nsimulations,1);
midxvec=1:tmpdim;
midxvec=midxvec';
midxvec=repmat(midxvec,tmpdim,1);
uvec=(midxvec-1+vvec)/tmpdim;
uvec2=gaminv(uvec,alpha,theta);
subplot(2,2,2);
hist(uvec2,100);
title('stratified gamma vars');
```

The fourth part of code in Table 5 is to calculate random variable from the normal distribution.

Table 5: The fourth part of coding algorithm

```
X(:,n+1)=normrnd(VG_theta*G(:,n+1),sigma*sigma*G(:,n+1));
subplot(2,2,3);
hist(X(:,n+1),100);
title('original normal vars');
```

This fifth part of code in Table 6 is to calculate stratifying random variables from normal distribution. This includes replicate and tile array (repmat function), reshape the array (reshape function) and computes the inverse of the normal norminv cdf (norminv function).

Table 6: The fifth part of the coding algorithm

```
G(:,n+1)=uvec2;
midxvec=1:tmpdim;
midxvec=repmat(midxvec,tmpdim,1);
midxvec=reshape(midxvec,tmpdim*tmpdim,1);
vvec=rand(nsimulations,1);
uvec=(midxvec-1+vvec)/tmpdim;
X(:,n+1)=norminv(uvec,VG_theta*G(:,n+1),sigma*sigma*G(:,n+1));
subplot(2,2,4);
hist(X(:,n+1),100);
```

Additional formulations in Table 7 are added to calculate the best calling price based on MCM.

Table 7: The sixth part of the coding algorithm

```
Tvec=0:1/(2^k):1;
Tvec=T*Tvec;
Tmat=repmat(Tvec,nsimulations,1);
Smat=exp( log(S)+r*Tmat+omega*Tmat+X );
Avgvec=mean(Smat(:,2:2^k+1),2);
payoffvec=max(Avgvec-K,0);
mc_callprice=exp(-r*T)*mean(payoffvec)
return;
```

3.2.3 Results and experiments with the outlier removal in VGP

This section presents results and experiments with the outlier removal in the use of VGP method. Outliers are common while during computation and they do not fall into the expected results. To achieve a high level of accuracy, outliers need to be removed. Figure 4 is a good example. Two screenshots on the left half with “original gamma variables” show that financial indexes have outliers. In other words, they are not smooth and outliers need to be filtered out. The other two screenshots on the right half with “stratified gamma variables” means that all outliers are removed and the results are smooth and usable for predictive modelling or other financial computation. Removal of outliers can ensure a high quality of data analysis can be offered to investors. How removal of outliers relates to the architecture is as follows. It ensures numerical computations are as accurate as possible and do not provide results that may make investments or decisions that incur in the loss of money and reputation.

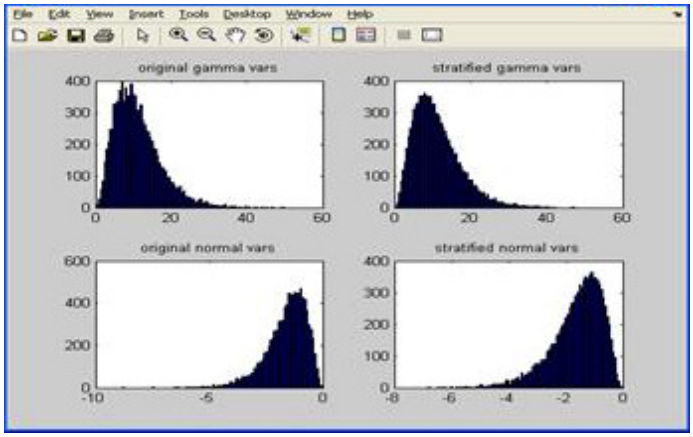

Figure 4: Removal of outliers

We then perform large-scale simulations to show that all computations can be achieved within seconds. The hardware infrastructure is as follows. The desktop has 2.67 GHz Intel Xeon Quad Core and 4 GB of memory (800 MHz) with installed. Two Amazon EC2 public clouds are used. The first virtual server is a 64-bit Ubuntu 8.04 with large resource instance of dual core CPU, with 2.33 GHz speed and 7.5GB of memory. The second virtual server is Ubuntu 7.04 with small resource of 1 CPU with 2.33 GHz speed and 1.5 GB of memory. There are two private clouds set up. The first private cloud is hosted on a Windows virtual server, which is created by a VMware Server on top of a rack server, and its network is in a network translated and secure domain. The virtual server has 2 cores of 2.67 GHz and 4GB of memory at 800 MHz. The second private cloud is a 64-bit Windows server installed on a rack, with 2.8GHz Quad Core Xeon, 16 GB of memory. Our FSaaS can work on both Octave or MATLAB. The experiment began for running the Octave 3.2.4 on desktop, private cloud and public cloud and started one at a time due to the licensing issue. The execution time is considered as a benchmark to test the service performance on the Cloud. Table 8 summarises the execution time, which involves running FSaaS simulations in four different platforms. It took less than 1 second more to run simulations in the public cloud with small instances. This is due to their low CPU and memory requirements resulting in longer completion time. Another factor is the distance between the different types of the clouds and the place where experiments were undertaken. The nearer the physicals distance was, the quicker to get tasks completed even though all the network speed remained at 1 Gbps. The key advantage of designing FSaaS as a service component is any change in the financial models which can be changed without any ripple effect to the entire FSaaS architecture.

Table 8: Timing benchmark to run FSaaS code on Octave 3.2.4

Number of simulations and time taken (sec)	5,000	10,000	15,000
Desktop	11.08	11.92	12.71
Public cloud (large instance)	11.95	12.30	13.15
Private cloud (default VMs)	11.31	12.13	12.90
Private cloud (large VMs)	9.63	10.51	11.48

3.3 The second major algorithm for high-performance computation of risk

The Least Square Methods (LSM) offers high-performance computation to calculate risk and pricing in FSaaS with two main reasons. Firstly, LSM provides a quick execution time, more than 50% compared with VGP illustrated in our previous work (Chang et al., 2011 a). Secondly, it allows the number of simulations to be pushed to 100,000 in one single attempt to show that all FSaaS service request can be completed once, before encountering issues such as stability and performance.

3.3.1 Core Code in Least Square Method (LSM)

This section describes the coding algorithm for the Least Square Method. Table 9 shows the initial part of the code, where key figures such as maturity, volatility and risk free rate are given. This allows us to calculate and track call prices if variations for maturity, risk free rate and volatility change. Similarly, we can modify our code to track volatility for risk analysis if other variables are changed. Both American price and European price methods are commonly used in Monte Carlo Simulations (Hull, 2009). It is an added value to calculate both prices in one go, and so both options are included in our code.

Table 9: The first part of coding algorithm for LSM

```
S=100; %underlying price
X=100; %strike
T=1; %maturity
r=0.04; %risk free rate
dividend=0;
v=0.2; % volatility
nsimulations=10000; % No of simulations, which can be updated
nsteps=10;    % 10 steps are taken. Can be changed to 50, 100, 150 and 200 steps.
CallPutFlag="p";
%%%%%%%%%%%%%%%%%%%%%%%%%
%AnalyAmerPrice=BjerkPrice(CallPutFlag,S,X,r,dividend,v,T)
r=r-dividend; %risk free rate is unchanged
%AnalyEquropeanPrice=BlackScholesPrice(CallPutFlag,S,X,T,r,v)
if CallPutFlag=="c",
        z=1;
else
        z=-1;
end;
```

Table 10: The second part of coding algorithm for the LSM

```
smat=zeros(nsimulations,nsteps);
CC=zeros(nsimulations,nsteps); %cash flow from continuation
CE=zeros(nsimulations,nsteps); %cash flow from exercise
EF=zeros(nsimulations,nsteps); %Exercise flag
dt=T/(nsteps-1);
smat(:,1)=S;
 drift=(r-v^2/2)*dt;
qrdt=v*dt^0.5;
for i=1:nsimulations,
   st=S;
   curtime=0;
   for k=2:nsteps,
      curtime=curtime+dt;
      st=st*exp(drift+qrdt*randn);
      smat(i,k)=st;
   end
end
```

The next step involves defining the three important variables for both American and European options, which include cash flow from continuation (CC), cash flow from exercise (CE) and exercise flag

(EF), shown in Table 10. The 'for' loop is used for starting the LSM process. Table 11 shows how the three variables CC, CE and EF are updated.

Table 11: The third part of coding algorithm for the LSM

```
CC=smat*0; %cash flow from continuation
CE=smat*0; %cash flow from continuation
EF=smat*0; %Exercise flag
st=smat(:,nsteps);
CE(:,nsteps)=max(z*(st-X),0);
CC(:,nsteps)=CE(:,nsteps);
EF(:,nsteps)=(CE(:,nsteps)>0);

paramat=zeros(3,nsteps); %coefficient of basis functions
```

Table 12 shows the main body of LSM calculations. The 'regrmat' function is used to perform regression of continuation value. This value is calculated, and fed into the 'ols' function, which is a built-in function offered by open-source Octave to calculate ordinary LSM estimation. The p value is the outcome of the 'ols' function, which is then used to determine final values of CC, EF and CE. In MATLAB 2009, the equivalent function is 'lscov' for the LSM. However, ols is not supported in the latest MATLAB such as 2013 versions.

Table 12: The fourth part of coding algorithm for the LSM

```
for k=nsteps-1:-1:2,
   st=smat(:,k);
   CE(:,k)=max(z*(st-X),0);

   %Only the positive payoff points are input for regression
   idx=find(CE(:,k)>0);
   Xvec=smat(idx,k);
   Yvec=CC(idx,k+1)*exp(-r*dt);
   % Use regression - Regress discounted continuation value at the
   % next time step to S variables at current time step
   regrmat=[ones(size(Xvec,1),1),Xvec,Xvec.^2];

   p=ols(Yvec,regrmat); %p = lscov(Yvec, regrmat) for MATLAB
CC(idx,k)=p(1)+p(2)*Xvec+p(3)*Xvec.^2;
   %If exercise value is more than continuation value, then
   %choose to exercise
   EF(idx,k)=CE(idx,k) > CC(idx,k);
   EF(find(EF(:,k)),k+1:nsteps)=0;
   paramat(:,k)=p;
   idx=find(EF(:,k) == 0);
   %No need to store regressed value of CC for next use
   CC(idx,k)=CC(idx,k+1)*exp(-r*dt);
   idx=find(EF(:,k) == 1);
   CC(idx,k)=CE(idx,k);
end
```

Table 13 shows the last part of the algorithm for the LSM. EF, calculated in Table 12 is used to decide values of an important variable 'payoff_sum', which is then used to calculate the best price for American and European options.

Table 13: The fifth part of coding algorithm for the LSM

```
payoff_sum=0;
for i=1:nsteps,
   idx=find(EF(:,i) == 1);
   st=smat(idx,i);
   payoffvec=exp(-r*(i-1)*dt)*max(z*(st-X),0);
   payoff_sum=payoff_sum+sum(payoffvec);
end

MCAmericanPrice=payoff_sum/nsimulations

st=smat(:,nsteps);
payoffvec=exp(-r*(nsteps-1)*dt)*max(z*(st-X),0);
payoff_sum=sum(payoffvec);
MCEurpeanPrice=payoff_sum/nsimulations
```

Upon running the MATLAB application, 'lsm', it calculates the best pricing values for American and European options in the Cloud. The following shows the outcome of executing LSM code.

> lsm

MCAmericanPrice = 6.3168

MCEuropeanPrice = 5.9421

3.3.2 Experiments and results on high-performing risk and price calculation in LSM

This section describes the experiments involved and results associated with LSM. In order to have a fair result compared to VGP, the same environment presented in Section 3.2.3 is used. MATLAB 2009 is used in the private clouds due to the restrictions of licence. Not every private cloud resource can be installed with the latest version of MATLAB. To make a fair comparison consistent throughout the experiments, MATLAB is used and compared between desktops and private clouds, as the licence is not available for the public clouds.

Time step is a unit in LSM to specific how many loops the code has been running to optimise accuracy. For example, if a computer program is to run 10 times to calculate the value of pi versus the same program to run 100 times, the one with 100 times of will have a closer result to the actual value since the values for standard deviations become smaller. However, the only limitation is that the program running 100 times takes more time than the one running only 10 times. This concept is the same for LSM. For the purpose of demonstration, a time step of 10 and 50 are used for comparisons. Figure 5 shows the results for time step is equal to 10 and Figure 6 shows the one for time step is equal to 50. The MATLAB code can run up to 100,000 simulations in one go to support its high-performance capacity. All simulations can be completed in 4 seconds shown in Figure 5 and less than 9 seconds shown in Figure 6. Results also show that firstly, the execution time and number of simulations are proportional to each other. It means the higher the number of simulations to be computed, the longer the execution time on desktop and the private cloud. It is not so obvious to identify linear relationship with a lower time step involved. This is likely because that execution time is so quick to complete that the range of errors and uncertainties are higher. When the time steps increases to 50, it is easier to identify the linear relationship indicated by LSM. Secondly, MATLAB 2009 offers quick execution time for the portability to Cloud,

and the significant time reduction is experienced. However, the licensing issue still prevents from a large scale of adoptions to different Clouds.

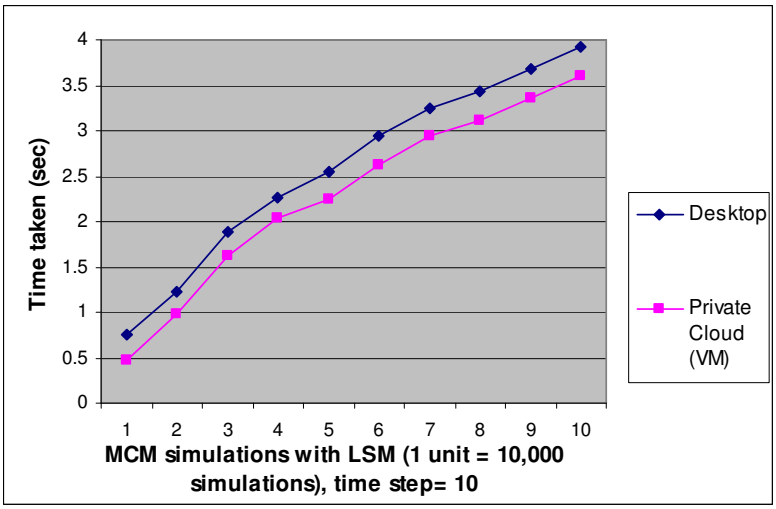

Figure 5: MATLAB timing benchmark for time step = 10

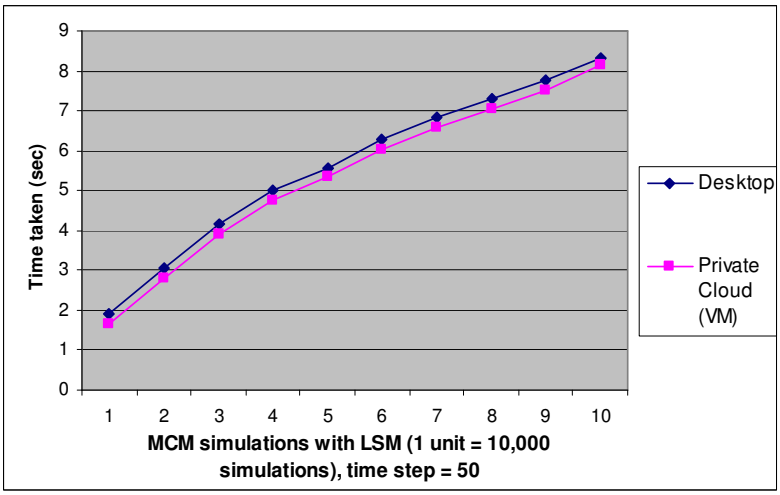

Figure 6: MATLAB timing benchmark for time step = 50

3.3.3 Other work: Risk Visualisation with Black Scholes Model (BSM)

This work has been running independently on its own, mainly to separate licensing issue experienced due to the use of MATLAB. Additionally, risk in visualisation has been providing useful results for scientists, stakeholders and investors in our previous demonstrations, in which large scale simulations have been performed to show that a large number of risk values such as 500,000 simulations can be computed and presented in not more than 25 seconds (Chang et al, 2014 b). This part of work should be developed on its own with FSaaS. The current focus on FSaaS is on numerical computation that

removes outliers and provides high-performing risk computation that can complete 100,000 simulations in 4 seconds in a default time step equal to 10, or 9 seconds for a time step of 50.

4. Discussion

Two topics of discussion with regard to FSaaS are presented as follows.

4.1 Our approach and contribution

Our paper presents a holistic approach for the FSaaS system design, algorithm and implementation. We explain the development of FSaaS from the phase of design to the implementation. Results in the experiments support that our FSaaS has a good performance. It deals with risk modelling and analysis while undertaking computation of the most suitable price options for investors. Our work presents a non-Gaussian model which can calculate risks and prices, and removes outliers to improve the accuracy and use optimisation by LSM to enhance the performance. Results can be computed within seconds. Our approach supports reproducibility, which is important for Science and organisational Cloud adoption, since it explains the syntax of the code in details rather than the theorectical algorithms which are often complex, hard to understand and difficult to be reused in a different environment. Our work has been used and adopted by a few organisations as illustrated in our previous work (Chang et al., 2011 a; 2014 c; Chang 2014 a; 2014 b).

4.2 Key factors for our approach and contribution

Chang (2014 a) explains the six success factors for Cloud Computing services and applications by illustrating the use of literature review, expert reviews, experiments and software demonstrations. These factors include usability, performance, computational accuracy, security, portability and scalability. While the current state of the art is focused on accuracy and performance to ensure that FSaaS can offer an alternative for risk modelling and analysis, future work will be focused on demonstrations by fulfilling other criteria. We can ensure that FSaaS can be easy to use, allow scalability, being able to work on different platforms such as smart phones and provide a secure environment. We will have more use cases to show that results computed by FSaaS can be as accurate as possible while completing all computational requests within seconds.

5. Conclusion and Future Work

Financial applications demand better performance and accuracy in a cloud than the traditional computing platforms. Therefore designing financial software as a service (FSaaS) requires systematic approach to engineering systems. This paper has demonstrated a systematic approach to developing large scaled financial applications with current technology such as SOA which supports SaaS features to be integrated with cloud solutions. This paper has also demonstrated the design and customisation of service component interfaces to the financial simulation so that it provides automatic prediction models for investors to know accurate results in buy and sale prices. Therefore, large-scale simulations for outlier removal can be delivered within a matter of 13.15 seconds and the large scale simulations for high-performance risk computation can be achieved within 9 seconds as demonstrated in our experiments. We show the holistic and complete approach of illustrating the system design of FSaaS, showing the two major algorithms and the results of experiments of running these two algorithms. While other research papers do not show the comprehensive design algorithm and implementation to support the reproducibility, an important aspect for Science and organisational adoption of Cloud Computing, we demonstrate our proof-of-concepts. We will continue to develop and improve on other services and integrate together with FSaaS applications.

References

Abdallah, M.A., Lei, Z.M., Li, X., Greenwold N., Nakajima, S.T., Jauniaux, E., Rao, C.V. (2004), Human Fetal nongonadal tissues contain human chorionic gonadotropin/ luteinizing hormone receptors. J. Clin. Endocrinol. Metab. 89, 952-956.

Accenture (2011) http://www.slideshare.net/fullscreen/ramblingman/accenture-financial-saa-s-external-presentation-final/3

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz R H, Kownwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M. (2009), Above the Clouds: A Berkeley View of Cloud computing. Technical Report Number UCB/EECS-2009-28, UC Berkeley, February.

Baresi, L., Di Nitto, E., and Ghezzi, C (2006) Towards open-world software: issues and challenges, Computer, Special Issue on Software Engineering: the past and the future, October 2006.

Beaty, K., Kochut, A. and Shaikh, H. (2009), Desktop to Cloud Transformation Planning, IEEE International Symposium on Parallel and Distributed Processing", May 23-May 29, Rome, Italy.

Birge, L. and Massart, P. (2001), Gaussian model selection, Journal of the European Mathematical Society, Vol. 3, No. 3, page 203-268, April.

Bryant, T.(2009), Mutuality 2.0, Working Paper, Leeds Metropolitan University and selected paper for Guardian, July.

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J. and Brandic, I. (2009), Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, Journal of Future Generation Computer Systems, Volume 25, Issue 6, June, Pages 559-616.

Campegemini (2011) Cloud Computing in Banking, Capegemini Analysis 2011, http://www.uk.capgemini.com/resource-file-access/resource/pdf/Cloud_Computing_in_Banking.pdf

Carr, P., Geman, H., Madan, D., and Yor, M. (2002), The fine structure of asset returns: An empirical investigation, Journal of Business, 75 (2), 305-332.

Cartwright I and Doernenburg E (2006) Time to jump on the SOA bandwagon, IT Now, British Computer Society, May 2006

Chang, V (2014 a) Business Intelligence as Service in the Cloud, Future Generation Computer Systems, (2014).

Chang, V. (2014 b) An Introductory Approach to Risk Visualization as a Service. Open Journal of Cloud Computing, 1, (1), 1-9.

Chang, V., Wills, G. and De Roure, D. (2010 a), A Review of Cloud Business Models and Sustainability. In: IEEE Cloud 2010, the third International Conference on Cloud Computing, 5-10 July, Miami, Florida, USA. pp. 43-50.

Chang, V., Wills, G. and De Roure, D. (2010 b), Towards Financial Cloud Framework: Modelling and Benchmarking of Financial Assets in Public and Private Clouds, IEEE Cloud 2010, the third International Conference on Cloud Computing, 5-10 July, Miami, Florida, USA.

Chang, V., Li, C. S., De Roure, D., Wills, G., Walters, R. and Chee, C. (2011 a) The Financial Clouds Review. International Journal of Cloud Applications and Computing, 1 (2). pp. 41-63. ISSN 2156-1834, eISSN 2156-1826.

Chang, V., Walters, R. J., & Wills, G. (2013). The development that leads to the Cloud Computing Business Framework. International Journal of Information Management, 33(3), 524-538.

Chang, V., Walters, R.J. and Wills, G. (2014 a) Review of Cloud Computing and existing Frameworks for Cloud adoption. In, Advances in Cloud Computing Research. , Nova Publishers.

Chang, V., Walters, R.J. and Wills, G. (2014 b) Financial Clouds and modelling offered by Cloud Computing Adoption Framework. In, Advances in Cloud Computing Research. , Nova Publishers.

Chang, V., Walters, R.J. and Wills, G. (2014 c) Monte Carlo risk assessment as a service in the cloud. International Journal of Business Integration and Management, 1-16. (In Press).

Chen, X., Chee, C., Huang, E., Jin, L., Tseng, R., Wang, C. and Wong, C. (2010), banking professionals, were asked in details for what software they use for financial work, by interviews.

City A.M, ongoing and several special editions about regulation of financial practices, 2010.

Feiman, J. and Cearley D W (2009), Economics of the Cloud: Business Value Assessments, Gartner RAS Core Research White Paper, September.

Financial Times (2009), Interview with Lord Turner, Chair of Financial Services Authority, June.

Financial Times (2010), special editions about regulation of financial practices.

Flouris, T. and Yilmaz, A. K. (2010), The Risk Management Framework to Strategic Human Resource Management, International Research Journal of Finance and Economics.

Groves, D (2005) Successfully planning for SOA, BEA Systems Worldwide, 11 Sept 2005

Hamnett, C. (2009), The Madness of Mortgage Lenders: Housing finance and the financial crisis, Working Paper, King's College London UK.

Helbig, J (2007) Creating business value through flexible IT architecture, Special Issue on Service-oriented Computing, IEEE Computer, V.40, No.11, November 2007.

Hohpe, G (2002) Stairway to Heaven, Software Development, May 2002

Hull, J. C. (2009), Options, Futures, and Other Derivatives, Seventh Edition, Pearson, Prentice Hall.

Hutchinson, M. (2010), A Finer Formula for Assessing Risk, Special column, Business Section, New York Times, May.

IBM (2011) IBM Business Monitor-Fine-Grained Security Model, http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/topic/com.ibm.iea.ibmonitor/ibmonitor/7.5/Administration/BPM_BusinessMonitor_FineGrainedSecurity.pdf

Jiang, T.J., Liu, D., Liu, K.M., Yang, W.Z., Tan, P.T., Hsieh, M.J., Chang, T.H. and Hsu, W.L. (2006), OpenVanilla – A Non-Intrusive Plug-In Framework of Text Services, ACM Journal, Computing Research Repository, 2006.

Li, C. S. (2010), Cloud Computing in an Outcome Centric World, keynote in IEEE Cloud 2010 conference, July 5-10, Miami, Florida, USA.

Longstaff, F.A., and Schwartz, E.S., Valuing American Options by Simulations: A Simple Least-Squares Approach. Review of Financial Studies, 14, 1, 113–147, 2001.

Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., & Ghalsasi, A. (2011). Cloud computing—The business perspective. Decision Support Systems, 51(1), 176-189.

Moreno, M. and Navas, J.F., On the Robustness of Least-Square Monte Carlo (LSM) for Pricing American Derivatives, Journal of Economic Literature Classification, April 2001.

Nano, O. and Zisman, A. (2007) Realizing service-centric software systems, Special issue on SoC, IEEE Software, November/December 2007.

NetSuite (2014) http://www.netsuite.co.uk/portal/uk/seo-landing-page/accounting-2/main.shtml?gclid=CLK9k5q-37sCFTHLtAodikoAzw, accessed 2014

Open Group (2009) OSIMM, Retrieved Oct 2013, from https://www2.opengroup.org/ogsys/jsp/publications/PublicationDetails.jsp?publicationid=12450

Ramachandran, M (2012) Software Security Engineering: Design and Applications, Nova Science Publishers, New York, USA, 2012. ISBN: 978-1-61470-128-6, https://www.novapublishers.com/catalog/product_info.php?products_id=26331

Ramachandran, M (2013) Business Requirements Engineering for Developing Cloud Computing Services, Springer, Software Engineering Frameworks for Cloud Computing Paradigm, Mahmmood, Z and Saeed, S (eds.), http://www.springer.com/computer/communication+networks/book/978-1-4471-5030-5

Rebeiro, C. and Webber, N. (2002), Valuing Path Dependent Options in the Variance-Gamma Model by Monte Carlo with a Gamma Bridge, Working Paper, No.02-04, Warwick Business School, September.

Ribeiro, C. and Webber, N. (2004), Valuing path-dependent options in the variance-gamma model by Monte Carlo with a gamma bridge, The Journal of Computational Finance 7 (2):81–100.

Schubert, H., Jeffery, K. and Neidecker-Lutz, B. (2010), The Future for Cloud Computing: Opportunities for European Cloud Computing Beyond 2010, Expert Group report, public version 1.0, January.

SCA (2014) Service Component Architecture (SCA), http://www.oasis-opencsa.org/sca, accessed 5th July 2014

Sellami, M. and Jmaiel, M (2013) A Secured Service-Oriented Architecture for FSaaS in Tunisia, www.redcad.org/PDFs/C33.pdf, accessed on 15th September.

Sovereign, B. (2005), Microsoft Office XP/2003 Executable Content Security Risks and Countermeasures, The US Government Working Paper, National Security Agency, May.

Taher, Y., Nguyen, D. K., Lelli, F., van den Heuvel, W. J., & Papazoglou, M. (2012). On engineering cloud applications-state of the art, shortcomings analysis, and approach. Scalable Computing: Practice and Experience, 13(3).

Waters, D. (2008), Quantitative Methods for Business, Fourth Edition, Prentice Hall, Financial Times.