Benchmarking Cloud Performance for Service Level Agreement Parameters

Lee Gillam, Bin Li, and John O'Loughlin
Department of Computing, University of Surrey, Guildford, United Kingdom
{ L.Gillam, B.Li, J.O'Loughlin}@surrey.ac.uk

Abstract: Infrastructure as a Service (IaaS) Clouds offer capabilities for the high-availability of a wide range of systems, from individual virtual machines to large-scale High Performance Computing (HPC) systems. But it is argued that the widespread uptake for such systems will only happen if Cloud providers, or brokers, are able to offer bilateral Service Level Agreements (SLAs). In this paper, we discuss how to measure and use quality of service (QoS) information to be able to predict availability, quantify risk, and consider liability in case of failure. We demonstrate through this work that there is a pressing need for such an understanding and explore a set of benchmarks that offers an interesting characterisation of resource performance variability which can be quite significant. We subsequently identify how such information might be used both directly by a user and indirectly via a Cloud Broker in the automatic construction and management of SLAs which reference certain kinds of financial portfolios.

Keywords: Cloud Computing, Service Level Agreement (SLA), Benchmarking, Cloud economics.

1 INTRODUCTION

Cloud systems offer a valuable alternative in provisioning for the IT needs of any organisation. Some argue that they are advantageous for businesses who are attempting to understand and manage costs, increase flexibility, and reduce the physical and environmental footprint of infrastructures. The idea that organisations need to procure hardware in order to offer computational services to themselves is undergoing yet another challenge - instead of purchasing and maintaining hardware and software with inherently high startup costs, organizations and individuals can rent a variety of (essentially, shared) computer systems for various periods of time. Whilst renting externally hosted systems is not new, under the Cloud banner much more is now rentable – from actual hardware, to virtualized hardware, through hosted programming environments, to hosted software and including various kinds of data storage and yet other services – and, importantly, rental periods can be much more flexible. In the past, hosted systems have implied extended rental periods, from months through years; in the

Cloud era, providers can bill down below the hour (indeed Rackspace to, down to three decimal places), although lower prices are the norm when commitments last months or years at a time.

The provision of systems by others brings further potential gains: organisations need not be so concerned about the upgrade cycle or about disposal of ageing systems – better systems may become readily available at the same price in the near future and the organisation need only be concerned with migration to this system whilst the Cloud provider takes care of the underlying upgrading or disposal. Furthermore, a failure in the underlying system becomes the provider's problem – although there may be consequential impact to the user organisation, which to some extent is the concern of this paper.

Some would argue about the features that really differentiate Cloud from previous hosted/rented/outsourced approaches. Although this is not our concern here, it is likely the agility that offers the difference – the speed at which it is possible to scale up or scale out, combined with the duration for which the resources need to be procured. This means that longer term needs can be readily separated from shorter, and different arrangements made in order to support both in the most financially advantageous manner instead of, say, provisioning always for the peaks in utilization. Further, it will be advantageous where the incorporated cost of the rented system is cheaper on the whole than the total cost of internal ownership. Where it is more expensive, the potential benefit in flexibility can offset this. Dialling up or down capacity to meet demand, rapid repurposing, or near-immediate upgrading (of the software by the provider, or of the operating system or even the scale of the virtualized hardware) are just some of these flexibilities.

However, the idea that Cloud is somehow generically cheaper than any owned infrastructure is variously open to challenge, and certainly there are many examples, amongst others, of high performance computing (HPC) researchers showing that Clouds cannot (yet) provide HPC capabilities of highly-optimized systems that are closely-coupled via low latency networks (U.S. Department of Energy, 2011) and perhaps they can not yet even achieve high performance message throughput comparable to systems from some 4 years previous (Gillam et al., 2012; Walker, 2008). However, absent access to such systems, or the finances with which to procure them, Cloud-based HPC can be the very next best thing and present a particularly compelling case where time is of the essence – and this extends to situations where queueing time on an existing HPC system may be an inhibitor. At present, cost and flexibility only extends so far. An end user would not yet be able to run, say, a specific computational workload with fine grained control over computational requirements at a given time, to readily obtain the best price against this across a number of Cloud providers, or to easily manage the risk of the computational workload failing, being able to re-run within a limited time without suffering a cost-of-cure, and subsequently being able to identify the liability for such a failure and, if relevant, appropriate compensation for any missed opportunity or consequential loss. For some applications, if the opportunity is lost potentially so is the customer, or perhaps

the investment opportunity; the former case could relate to increased latency on serving out webpages or processing orders, whilst the latter would feature more substantially in high frequency trading applications where even microseconds of latency can make a difference.

Clouds need to support measurement, validation and specification, and one way to achieve this in agreeement with the user of the Cloud system is to specify commitments in Service Level Agreements (SLA). Until proper assurances of availability, reliability, and, perhaps, liability are formulated into these SLAs, there is likely to be an upper limit on the number of potential end-users. But this is not exactly a new proposition: in the recent past a number of Grid Economics researchers suggested that *Grids* could only attain acceptance by a larger audience of commercial end-users when binding SLAs were provided (e.g. Leff et al., 2003; Jeffery, 2004); meanwhile, the notion of Grids – at least in the business sphere – seems long gone.

Given the need for substantial computation, the financial sector could be a key market for Clouds for their pricing models and portfolio risk management applications. Certainly such applications can fit with a need for 'burst' capability. However, the commercial adoption reported of commoditized public infrastructures by the financial sector seems limited, with such organizations appearing to prefer to maintain bespoke infrastructures or to convert these to private (internal) Clouds – which some would argue are often simply the virtualized version of these bespoke infrastructures. Lack of uptake of public Clouds may suggest that either those organizations are concerned with maximizing their existing sunken investments, or that the lack of assurances of availability, reliability, security, and liability amongst other likely concerns, is a hindrance.

In our work, we are aiming towards building features more coherently into SLAs that relate at least to reliability and liability. Potential failure on an SLA relating to the performance of a specific system or application could occur for a number of reasons, but we generally consider these to either encompass performance variability in the network or on the systems being connected to across the network, or due to failure of a part of, or the whole of, the underlying infrastructure of the system. A saturated network connection, or a server running in a degraded manner, would fit with the former; server failure behind a load balancer, or in a cluster, would exemplify the latter.

We are working towards building application-specific SLAs that directly relate performance to cost, and so would offer a Cloud Broker the potential to present offers of SLAs for negotiation on the basis of price, performance, and liability – price can readily incorporate performance and liability, but transparency is likely to be beneficial to the end user. There are various reasons why such an application-specific SLA should be machine-readable. Firstly, this means they can be offered automatically, taking account of the present loading of systems and performance across the existing SLAs. Secondly, human interpretation, negotiation, and enforcement can be clumsy and would introduce additional latency likely leading to offers not being accepted in time; thirdly, the

Broker would be able to facilitate autonomic approaches to reduce the impact of failure and promote higher utilization.

Following Kenyon and Cheliotis (2002), we are working to construct risk-assessed and potentially risk-balanced portfolios of compute resources that could support a different formulation of the Cloud Economy. In portfolio management, it is important to reduce exposure to risk by ensuring that not all items in the portfolio are likely to move identically to the same underlying stimulus – hence avoiding, for example, composing a portfolio where all items in it came from the same industry sector. Our initial work in financial risk was geared towards gaining such an understanding of risk and its analysis within increasingly complex financial products and markets. Ironically, many of these complex financial products and markets have largely now disappeared as they encompassed underlying risks which could be readily hidden in their construction.

Related work on computational risk assessment using financial models to lead to financial valuations of risk often needs to assume that the underlying price is variable. However, Cloud prices seem quite fixed and stable over long periods with relatively few exceptions (e.g. the spot prices of Amazon, which are not particularly volatile but show instantaneous extremes of price movement that could be quite difficult to model and therefore predict with any degree of confidence). Recent work (Rogers and Cliff, 2012) has suggested that a Cloud broker could obtain resources at such fixed prices over extended periods (the example given uses Amazon's reserved instances), and profit from re-selling at shorter intervals provided that sufficient numbers of buyers were retained in the market. Such a broker would then be in a position to introduce variable pricing on the basis of true demand, but would not yet be able to address issues of performance or liability.

Our approach would make price variable in line with performance to begin with, and so price volatilities would follow performance volatilities over time. If resources are already obtained at fixed prices, the broker needs to assess the capability that can be offered to the user, and set up a pricing regime that varies with that capability. Additionally, there is a need to factor in the effect of failure of the underlying resource, which it is difficult to account for in traditional financial risk models since its occurrence is, effectively, the final event in price data. This means that there needs to be a way to determine the likelihood of such an event (in finance, the company being declared bankrupt) occurring at a given point in time. The analogous Cloud risk, then, should address the probability of a default on the SLA due to the failure or potential for failure or one or more of the underlying resources. The ability to account for that risk is then formulated into an offer price which includes it – essentially, building in the costs of an insurance policy over and above the costs of obtaining the resources.

In this paper, we present a way to incorporate performance information (QoS) into SLAs as an initial and vital step towards modelling risk such that it can be factored into pricing that includes such a risk. In section 2, we discuss SLAs at large, and machine-readable SLAs in particular, to identify the fit. In section 3,

we present results of benchmarks that show the extent of variability present in Cloud resource performance – our use of benchmarks is a means to an end in bringing price variability into the risk models, not an end in itself. Section 4 addresses price-based variability in performance, and section 5 offers suggestions for bringing risk assessment and SLAs together. We conclude the paper, and offer a few future work suggestions. An extended discussion of the benchmarks and results, and a visualization of these benchmark results, is also available[1].

2 CLOUD SERVICE LEVEL AGREEMENTS

2.1 Service Level Agreement (SLA)

A Service Level Agreement (SLA) acts as a contract between a service provider and a consumer (end user), possibly negotiated through a broker. The SLA should clarify the relationship between, and especially the obligations on, the parties to the contract. For the consumer, it should clarify performance and price, and describe penalties for under-performance or failure (essentially, liabilities). Sturm et al. (2000) have highlighted the components for a common SLA: purpose; parties; validity period; scope; restrictions; service level objectives; service level indicators; penalties; optional services; exclusions and administration.

At a broad level, Service Level Agreements can cover such organisational matters as how promptly a telephone call will be answered by a human operator with, say, a small penalty paid by the service provider to the service consumer if a "four rings" maximum is not met. In computing in general, and in Cloud Computing in particular, discussions of SLAs are often concerned with an overall level of service availability. Service availability may be expressed in terms of availability during different time periods, for example requiring 99.9% availability during core time, and 85% during non-core time (Bose et al, 2011). However, recent discussion extends this to embedding security requirements such as encryption, data location, compliance standards into SLAs in natural language, and also into machine readable SLAs using deontic logic (Meland et al, 2012), and even embedding protection from denial of service attacks by including attack name, description and detection using metrics for various types of DDoS attacks (Rak and Ficco, 2012).

Various public Cloud providers have service terms and/or service level agreements, provided on an "as-is" basis, but assurances to the consumer may not be particularly favourable. For instance, at the time of writing, the following clauses could be found in specific SLAs readily available on the web: "Google and partners do not warrant that (i) Google services will meet your requirements, (ii) Google services will be uninterrupted, timely, secure, or error-free, (iii) the results ... will be accurate or reliable, ..."; "Amazon services have no liability to you for any unauthorized access or use, corruption, deletion, destruction or loss

[1] A report of the Fair Benchmarking for Cloud Computing project can be found via http://tinyurl.com/FairBenchReport, and a data visualization can be found via http://tinyurl.com/FairBenchViz.

of Your Content.". Such an SLA is *non-negotiable*, and largely written to protect the provider. Negotiation over SLAs may be a possibility, but in mainstream provision this is likely to become a long and drawn out process which requires a longer term commitment prior to obtaining the resources required – not really geared towards Cloud provision so much as datacentre rental - and yet still be at a relatively generic level.

Presently, SLA clauses related to liability are likely to be few and limited to outright failure at best. In the event, service credits may be offered – rather than a refund – and the consumer can either stick with the present provider or go elsewhere. Standard SLAs for Cloud providers tend to fit such a description and it is not yet apparent that the situation will change in the near future.

Amazon Web Services (AWS) began offering an SLA in 2008, although some customers had already lost application data because of an outage in October 2007. Amazon CTO, Werner Vogels is often cited as saying, "Everything fails all the time. We lose whole datacentres! Those things happen." However, Vogels also assures us "let us worry about those things, not you as a start-up. Focus on your ideas.". The consumers of Cloud should heed the former warning, and architect applications to cope with failures. However, such architecting – for example to increase the level of redundancy of systems used or to replicate systems and data across geographies - is inherently going to increase the costs involved but still offers no assurances over performance.

As highlighted in Table 1, just for AWS, things do indeed fail. It becomes apparent that even a slight SLA offers some form of compensation (compare 2011 to 2007), but does this really make up for the consequences of such failures, and concomitant costs?

Table 1 AWS outages.

Disrupted Services	Cause	Description
EC2 [beta] (Oct, 2007)	Management software	Customer instances terminated and application data lost. *No SLA yet.*
S3 (Feb, 2008)	Exceeded authentication service capacity	Disruption to S3 requests, and lack of information about service. *Service health dashboard developed.*
EC2 (June, 2009) (July, 2009)	Electrical storm, lightning strike	Offline for over five hours; instances terminated. Loss of one of the 4 U.S. east availability zones. *Customers request more transparent information.*
EC2, EBS, RDS (Apr, 2011)	Network fault and connection failure	High profile outage in one of four U.S. East availability zones; servers re-replicating data volumes, causing data storm. *Affected customer receives more service credits than stated in the SLA.*

To obtain Service Credits for AWS, availability must drop below 99.95% based on "the percentage of 5 minute periods during the Service Year in which Amazon EC2 was in the state of "Region Unavailable"". It is unclear, here, whether such periods are cumulative, so two separate periods of 4 minutes might count as one period, or whether neither would count for anything. In addition, "Region Unavailable" is some way different to the lack of availability of a given resource within a region, and different still to degraded performance. Such conditions will impact on the performance of the supported application, and also have both a direct cost implication if trying to keep a level of performance, or an indirect one if opportunity or business is lost as a consequence.

Ideally, commercial systems would readily provide more than merely "best effort" or "commercially reasonable effort" over some long period, and would be monitored to assure this transparently to the consumer. Such SLAs should have negotiable characteristics at the point of purchase, and here we consider that such negotiation could readily be mediated through machine-readable SLAs. We discuss such machine-readable SLAs in the next section.

2.2 Machine-readable SLAs

The Cloud Computing Use Cases group has discussed Cloud SLAs and emphasised the importance of service level management, system redundancy and maintenance, security and privacy, and monitoring, as well as machine-readability. In addition, they describe use of SLAs in relation to Cloud brokers. We anticipate machine-readability of SLAs becoming increasingly important in supporting large numbers of enquiries for Cloud resources in a fast moving market. For these machine readable SLAs, there are several possibilities, amongst which are the Web Service Agreement (WS-Agreement) developed by Open Grid Forum (OGF) and described by Andrix et al. (2007), and Web Service Level Agreement (WSLA), introduced by IBM in 2003. WS-Agreement appears to be a preference amongst those working on machine-readable approaches, and has been our preference also.

WS-Agreement follows related contractual principles, allowing for the specification of the entities involved in the agreement, the work to be undertaken, and the conditions that relate to the performance of the contract. Initially, WS-Agreement consists of two sections: the Context, which defines properties of the agreement (i.e. name, date, parties of agreement); and the Terms, which are divided into Service Description Terms (SDTs) and Guarantee Terms (GTs). SDTs are used to identify the work to be done, describing, for example, the platform upon which the work is to be done, the software involved, and the set of expected arguments and input/output resources. GTs provide assurance between provider and requester on QoS, and should include the price of the service and, ideally, the probability of, and penalty for, failure. In addition, the Service Negotiation and Acquisition Protocol (SNAP, Czajkowski et al., 2002) defines a message exchange protocol between end-user and provider for negotiating an SLA. It supports the following: resource acquisition, task submission and task/resource binding.

Related work in the AssessGrid project (Kerstin et. al, 2007) uses WS-Agreement in the negotiation of contracts between entities. This relies on the creation of a probability of failure (PoF), which influences price and penalty (liability). The end-user compares SLA offers and chooses providers from a ranked list: the end-user has to evaluate the combination and balance between price, penalty and PoF. Such an approach gears readily towards Cloud Brokerage, in which multiple providers are contracted by the Broker, and it is up to the Broker to evaluate and factor in the PoF to the SLA. It is possible, then, that such a Broker may make a range of different offers which appear to have the same composition by providers but may vary for a number of reasons, not least because of actual performance and the PoF. For PoF, we may consider partial and complete failure – where partial may be a factor of underperformance of one or more resources, or complete failure of some resources, within a portfolio of such resources.

As well as PoF and liability, a machine-readable SLA should also address, at least, service availability, performance and autonomics:

Service Availability: This is typically considered to denote responsiveness to user requests. In most cases, it is represented as a ratio of the expected service uptime to downtime during a specific period. It usually appears as a number of nines - five 9s refers to 99.999% availability, meaning that the system or service is expected to be unresponsive for less than 6 minutes a year. An AppNeta study on the State of Cloud Based Services, available as one of their white papers, found that of the 40 largest Cloud providers the suggested average Cloud service availability in 2010 was 99.948%, equivalent to 273 minutes of downtime per year. Google (99.9% monthly) and Azure (99.9%, 99.95% monthly) reportedly failed to meet their overall SLA, while AWS EC2 (99.95% yearly) met their SLA but S3 (99.9% monthly) fell below. Of course, such figures could be cited differently – and the Amazon 5 minutes approach referred to earlier could be applied in a five 9s situation where only outages of 1 minute or more are counted. For us, availability can be somewhat difference to performance; the latter could vary substantially whilst availability is maintained – put another way, contactable but impossibly slow means that it is still available.

Performance: According to a survey from IDC in 2009 (Gens, 2009), the performance of a service is the third major concern following security and availability. Websites such as CloudSleuth and CloudHarmony offer some information about performance of various aspects of Cloud provisions, however there appear to be just one or two data samples per benchmark per provider, and so in-depth performance information is not available, and further elements of performance such as provisioning, booting, upgrading, and so on, are not offered. Performance consideration is vital since it becomes possible to pay for expected higher performance yet receive lower – and not to know this unless performance is being accurately monitored over time. The responsibility for that monitoring then comes into question – will a Cloud provider accept the consumer's monitoring where the latter shows a problem but the former does not?

Autonomics: A Broker's system may need to adapt to changes in the setup of the underlying provider resources in order to continue to satisfy one or more SLAs. Maintenance and recovery are just two aspects of autonomics such that partial or complete failure is recoverable with a smaller liability than would be possible otherwise. Large numbers of machine-readable SLAs will necessitate an autonomic approach in order to optimize utilization – and therefore profitability – for the Broker. Indeed, if the SLA can be used as a means to capture and therefore manage information about risk of extant agreements, at worst the Broker may decide to make no offers at a given time as may prove harmful to this set.

Since PoF should be grounded in Performance, and indeed can offer a better basis for presenting Service Availability, in the remainder of this paper we focus primarily on capturing information about performance. Performance variability of virtualized hardware is going to have an impact on Cloud applications, and the stated cost of the resource, typically focussed on by others in relation to Cloud Economics, is but a distraction from this – the performance achieved at the price paid is of greater importance and we will see that it can have greater variability: low performance at the same price is almost certainly going to be undesirable for the consumer, but cannot as yet be assured against by the provider.

To measure performance variability, we investigate a small set of benchmarks that allow us to compare performance both within Cloud instances (of the same type) of a few Cloud Infrastructure providers, and across them. The results should offer room to reconsider price variability as performance variability, and look at risk as a more dynamic problem potentially suited for assessment using risk analysis approaches from computational finance.

3 PERFORMANCE BENCHMARKS

The problem of (lack of) performance metrics in SLAs has been considered as a semantic one (Amato et al, 2012), with a solution proposed of mapping low level system metrics to high level SLA statements (Emeakaroha et al, 2010). Unforunately, the metrics that were proposed are insufficient to capture application performance and, with the exception of network bandwidth, no QoS metrics (storage read/write, memory bandwidth etc) are actually considered. In part to address this, we selected benchmarks as would address QoS metrics to be run on Linux distributions in 4 Cloud infrastructures (AWS, Rackspace, IBM, and private Openstack) to obtain measurements for CPU, memory bandwidth, and storage performance. We also undertook measurements of network performance for connectivity to and from providers, particularly of interest if migrating data but also in relation to so-called Big Data, and to assess present performance in relation to HPC type activities. Literature on Cloud benchmarking often reports CPU, Disk IO, Memory, network, and so on, usually involving AWS. Many such results reveal performance with respect to the benchmark code for AWS and possibly contrast it with a second system – though not always. It becomes the task of an interested researcher, then, to try to assemble disparate distributed tests, potentially with numerous different

parameter settings – some revealed, some not – into a readily understandable form of comparison. Many may balk at the need for such efforts simply in order to obtain a sense of what might be reasonably suitable for their purposes. What we want to see, quickly, is whether there is an outright "better" performance, or whether providers' performance varies – likely due to the underlying physical resources and variability with provision through the hypervisor.

We have tested several AWS regions, two Rackspace regions, the IBM SmartCloud, and a Surrey installation of OpenStack.

In this paper, we present results for:

1. STREAM, a standard synthetic benchmark for the measurement of memory bandwidth
2. Bonnie++, a Disk IO performance benchmark suite that uses a series of simple tests for read and write speeds, file seeks, and metadata operation;
3. LINPACK, which measures the floating point operations per second (flop/s) for a set of linear equations.

3.1 Benchmark Results

As shown in Table 2 and Figure 1 and for STREAM copy, there are significant performance variations among providers and regions. The average of STREAM copy in AWS is about 5GB/s across selected regions with 2 Linux distributions. The newest region (Dec, 2011) in AWS, Sao Paulo, has a peak at 6GB/s with least variance. The highest number is obtained in Surrey's Openstack at almost 8GB/s, but with the largest variance. Results in Rackspace look stable in both regions, though there is no indication of being able to 'burst out' in respect to this benchmark. The variance shown in Table 2 and Figure 1 suggests potential issues either with variability in the underlying hardware, contention on the same physical system, or variability through the hypervisor. It also suggests that other applications as might make demands of a related nature would suffer from differential performance on instances that are of the same type.

Table 2 STREAM (copy) Performance Results (MB/s)

	AWS US-East Ubuntu	AWS US-East RedHat	AWS US-West Ubuntu	AWS SA Ubuntu	IBM US RedHat	RS UK Ubuntu	RS US Ubuntu	OS Surrey Ubuntu
Min	2358.35	3219.70	4724.18	5853.16	4711.38	3383.53	4084.93	1961.12
Avg	4090.44	4181.46	4770.21	5980.01	5499.05	4121.16	4168.87	3758.68
Max	4796.16	4579.81	4851.99	6051.84	5743.85	4305.71	4282.85	7980.98
SD	888.21	584.32	38.14	68.61	317.52	295.44	62.55	1994.46

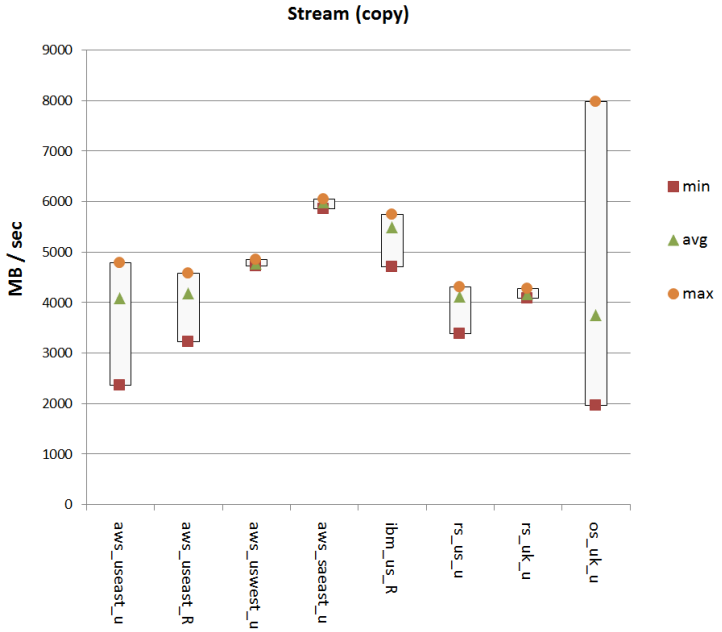

Figure 1 STREAM (copy) benchmark across four infrastructures. Labels indicate provider (e.g. aws for Amazon), region (e.g. useast is Amazon's US East) and distribution (u for Ubuntu, R for RHEL).

Table 3 bonnie++ (sequential create) Performance Results

	AWS US-East Ubuntu	AWS US-East RedHat	AWS US-West Ubuntu	AWS SA Ubuntu	IBM US RedHat	RS UK Ubuntu		OS Surrey Ubuntu
Min	9267	9569	9320	4275	4047	18529		535
Avg	12730	15974	18157.3	18009.2	13347.3	21660.71		11993.67
Max	14159	20260	23182	25620	18009	23985		29663
SD	1387.06	3557.55	4622.99	7014.06	4607.67	1857.57		12818.43

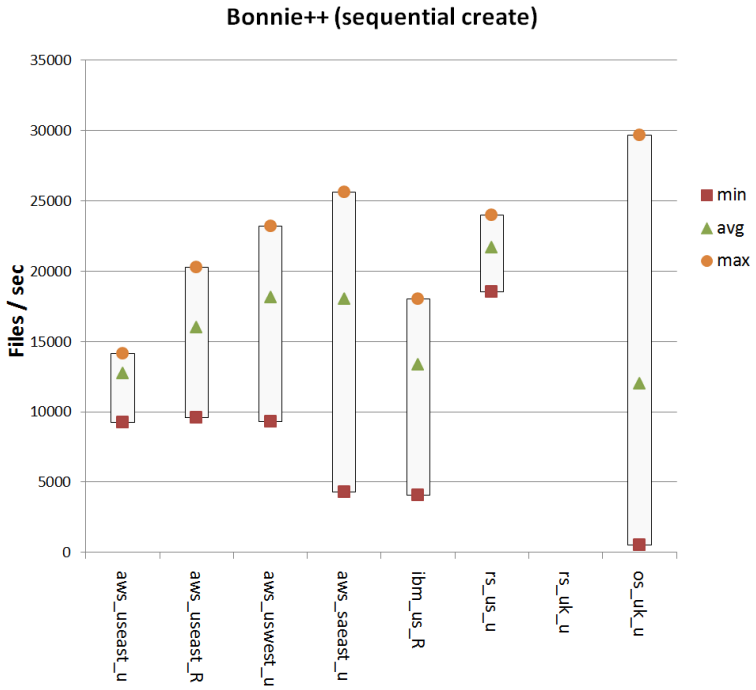

Figure 2 Bonnie++ (sequential create) benchmark across four infrastructures.

Table 3 and Figure 2 show results for Bonnie++ for sequential creation of files per second (we could not get results for this from Rackspace UK for some unknown reason). Our Openstack instances again show high performance (peak is almost 30k files/second) but with high variance. The EC2 regions show differing degrees of variance, mostly with similar lows but quite different highs.

We obtained LINPACK from the Intel website, and since it is available pre-compiled, we can run it using the defaults given which test problem size and leading dimensions from 1000 to 45000. However, Rackspace use AMD CPUs, so although it would possible to configure LINPACK for use here, we decided against this at the time. Results for Rackspace are therefore absent from Figure 3 and Table 4, and also for AWS US east region because we assumed these would be reasonably comparable with other regions.

AWS instances produce largely similar results, without significant variance. Our OpenStack Cloud again suffers in performance – perhaps a reflection on the age of the servers used in this setup.

Table 4 Linpack (intel) Performance Results

		AWS US-East RedHat	AWS US-West Ubuntu	AWS SA Ubuntu	IBM US RedHat			OS Surrey Ubuntu
Min		11.09	10.81	14.40	11.63			2.20

Avg		11.82	11.14	14.53	12.09			4.13
Max		13.53	11.40	14.58	12.25			7.03
SD		0.98	0.17	0.06	0.18			1.77

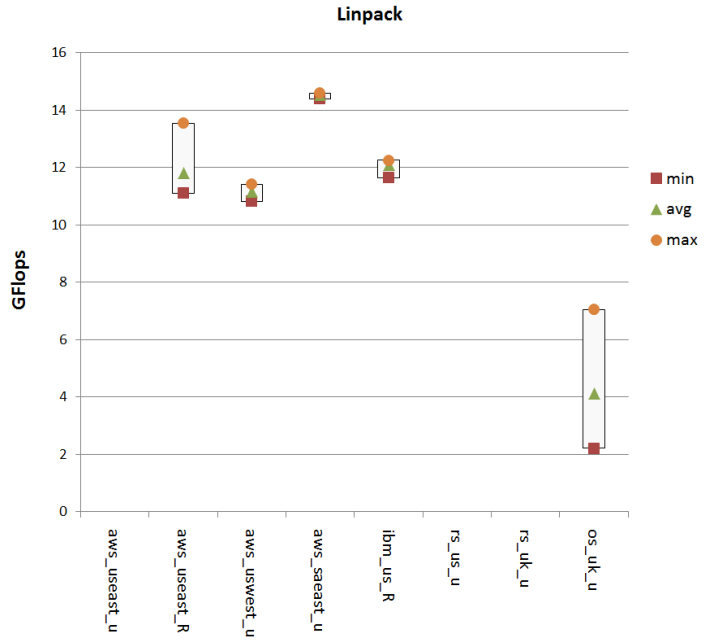

Figure 3 LINPACK (25000 tests) benchmark across tested infrastructures.

In contrast to EC2, we have both knowledge of and control of our private Cloud infrastructure, so we can readily assess the impact of sizing and loading and each machine instance can run its own STREAM, so any impacts due to contention should become apparent. The approach outlined here might be helpful in right-sizing a private Cloud, avoiding under- or over- provisioning.

We provisioned up to 128 m1.tiny (512MB, 1 vCPU, 5GB storage) instances simultaneously running STREAM on one Openstack compute node. The purpose of this substantial load is to determine the impact on the underlying physical system in the face of additional loading.

Table 5 and Figure 4 indicate total memory bandwidth consumed. With only one instance provisioned, there is plenty of room for further utilization, but as the number of instances increases the bandwidth available to each drops. A maximum is seen at 4 instances, with significant lows at 8 or 16 instances but otherwise a general degradation as numbers increase. The significant lows are interesting, since we'd probably want to configure a scheduler to try to avoid such effects.

Table 5 STREAM (copy) Performance Results (MB/s) in OpenStack

Underlying: compute04 Intel(R) Xeon(R) CPU E5540 @ 2.53GHz 8 cores, 2 chips, 4 cores/chip, 2 threads/core; Max Memory Bandwidth 25.6 GB/s; 32GB DDR3 1066 MHz mem; number of instances simultaneous running

stream N=5000000				
	Min	Max	Avg	Bandwidth Total
1	7673.09	7673.09	7673.09	7673.09
2	6974.19	13948.37	10461.28	20922.56
4	5283.30	21133.19	13208.24	52832.97
8	1817.38	14539.08	8178.23	65425.85
16	998.74	15979.82	8489.28	135828.48
32	586.01	18752.44	9669.22	309415.18
64	285.05	18243.21	9264.13	592904.26
128	127.12	16271.06	8199.09	1049483.63

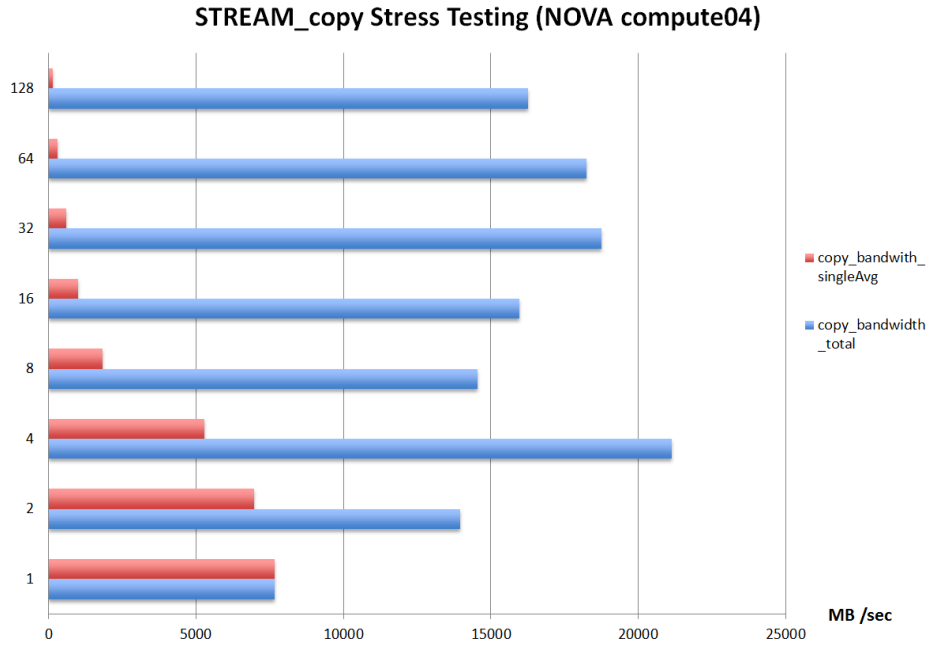

Figure 4 STREAM (copy) benchmark stress testing on Nova compute04, showing diminishing bandwidth per machine instance as the number of instances increases, and variability in overall bandwidth.

4 PERFORMANCE AND COST DISCUSSION

We have seen that there can be a reasonable extent of variation amongst instances from the same provider for these benchmarks, and the range of results is more informative than simply selecting a specific best or average result. Applications run on such systems will also be impacted by such variation, and yet it is a matter hardly addressed in Cloud systems. Performance variation is a question of Quality of Service (QoS), and service level agreements (SLAs) tend only to offer compensation when entire services have outages, not when performance dips. The performance question is, at present, a value-for-money question. But the question may come down to whether we were simply lucky or not in our resource requests. Variation may be more significant for smaller machine types as more can be put onto the same physical machine – larger types

may be more closely aligned with the physical resource leaving no room for resource sharing. Potentially, we might see more than double the performance of one Cloud instance in contrast to another of the same type – and such considerations are likely to be of interest if we were introducing, for example, load balancing or attempting any kind of predictive scheduling. Also, for the most part, we are not directly able to make comparisons across many benchmarks and providers since the existing literature is usually geared to making one or two comparisons, and since benchmarks are often considered in relative isolation – as here, though only because the large number of results obtained becomes unwieldy.

It should be apparent, however, that a Cloud Broker could – at a minimum - re-price such resources according to performance and offer performance-specific resources to others. However, the current absence of this leads to a need for the Cloud Broker to understand and manage the risk of degradation in performance or outright failure of some or all of the provided resources.

5 PROCESSES AT RISK

5.1 Cloud Monitoring

Cloud providers may offer some (typically graph-based) monitoring capabilities. Principal amongst these is Amazon's CloudWatch, which enables AWS users to set alarms for various metrics such as CPUUtilization (as a percentage), DiskReadBytes, DiskWriteBytes, NetworkIn, and NetworkOut, amongst others. The benchmarks we have explored are highly related to this set of metrics, and so it is immediately possible to consider how this would inform the setting of such alarms – although there would still be some effort needed in obtaining likely performance values per machine instance to begin with.

An alarm can be set when one of these metrics is above or below a given value for longer than a specified period of time (in minutes). At present, unless an AutoScaling policy has been created, alarms will be sent by email. Further work is needed to build a better approach to using such alarms.

5.2 Building SLAs

We have investigated the use of WS-Agreement in the automation of management of SLAs. WS-Agreement was adopted by various Grid research projects (Seidel et.al., 2007) and is becoming popular for Cloud SLAs also. WS-Agreement offers a predefined agreement templates model which describes the unique structure of general agreements, but can be used for specific requirements. In WS-Agreement, SLA creation occurs in three steps: (i) consumer receives an SLA template from the provider which specifies the offers that consumer is willing to accept; (ii) consumer make the choice either to accept an offer or reject offers by the provider; (iii) agreement negotiation may then take place in order to address further details of the agreement.

Development of WS-Agreement is now addressing better renegotiation and interoperability solutions (Ludwig et.al., 2006). Based on the latest WS-Agreement specification (1.0.0), the implementation of WS-Agreement for Java

framework (WSAG4J) helpfully separates out static resource properties – such as quantity of memory, numbers of CPUs, and so forth – from the dynamic resource properties – typically, limited to response times (Figure 5) but with scope to incorporate QoS metrics that can vary over the agreement lifetime. As demonstrated by performance variations in this paper, dynamic characteristics are rife, and support for continuous monitoring of performance and how this may affect the SLA is of particular importance in any SLA framework.

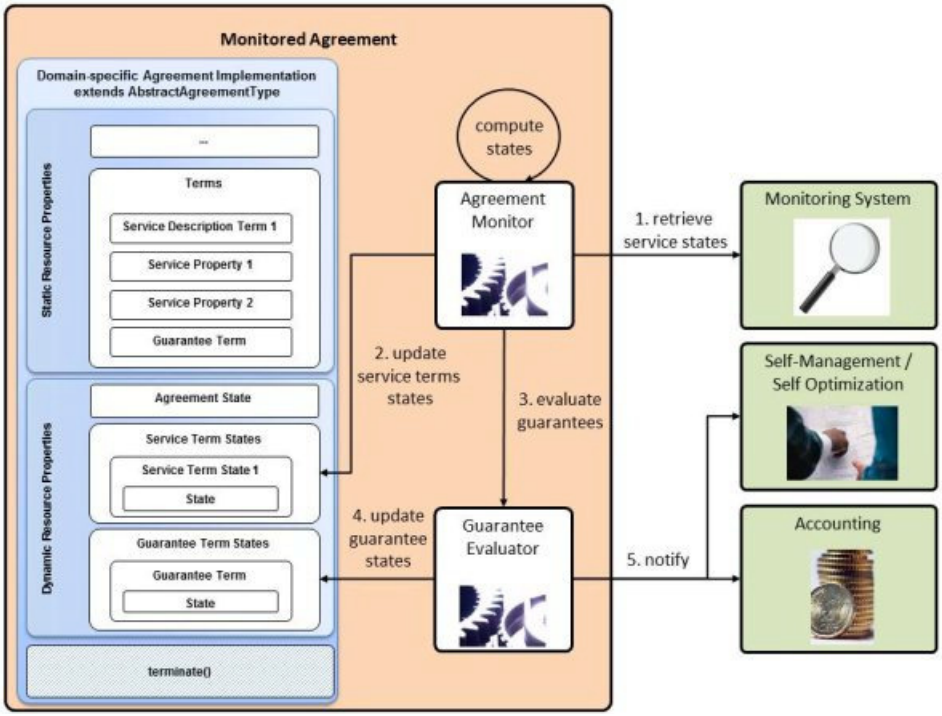

Figure 5 OGF Agreement Monitoring (source: WSAG4J, Agreement monitoring).

Introducing a Cloud Broker adds an element of complexity, but this may be beneficial. If users demand detailed SLAs but Cloud providers do not offer them, there is a clear purpose for the Broker if they can interpret/interrogate the resources in order to produce and manage such SLAs. ServiceQoS offers suggestions for how to add QoS parameters into SLAs. The principal example is through a Key Performance Indicator (KPI) Target (wsag:KPITarget) as a Service Level Objective (wsag:ServiceLevelObjective), and relates to Response Time (wsag:KPIName) (Figure 6). Examples elsewhere use Availability, and a threshold (e.g. gte 98.5, to indicate greater than or equal to 98.5%).

Cloud providers and frameworks supporting this on a practical level are as yet not apparent, leaving the direct use of QoS parameters in SLAs for negotiation via Cloud Brokers very much on our future trajectory.

```
<wsag:ServiceProperties
            wsag:Name="AvailabilityProperties"
```

```
             wsag:ServiceName="GPS0001">
     <wsag:Variables>
             <wsag:Variable
                   wsag:Name="ResponseTime"
                   wsag:Metric="metric:Duration">

<wsag:Location>qos:ResponseTime</wsag:Location>
           </wsag:Variable>
            </wsag:Variables>
</wsag:ServiceProperties>

 <wsag:GuaranteeTerm
            Name="FastReaction"
Obligated="ServiceProvider">
….
<wsag:ServiceLevelObjective>
  <wsag:KPITarget>
    <wsag:KPIName>FastResponseTime</wsag:KPIName>
      <wsag:Target>
      //Variable/@Name="ResponseTime" LOWERTHAN 800 ms
      </wsag:Target>
    </wsag:KPITarget>
 </wsag:ServiceLevelObjective>

 <wsag:BusinessValueList>
   <wsag:Importance>3</wsag:Importance>
   <wsag:Penalty>
     <wsag:AssesmentInterval>
       <wsag:TimeInterval>1 month</wsag:TimeInterval>
     </wsag:AssesmentInterval>
     <wsag:ValueUnit>EUR</wsag:ValueUnit>
     <wsag:ValueExpr>25</wsag:ValueExpr>
   </wsag:Penalty>
 <wsag:Preference>
                  .........
</wsag:Preference>
</wsag:BusinessValueList>
</wsag:GuaranteeTerm>
             .........
```

Figure 6. An example WS-Agreement Template (source: http://serviceqos.wikispaces.com/)

5.3 Collateralized Debt Obligations (CDOs) and Cloud SLAs

In financial analysis there are various techniques that are used to measure risk in order that it might be quantified and also diversified within a portfolio to ensure that a specific event has a reduced impact on the portfolio as a whole. Previously, Kenyon and Cheliotis (2002) have identified the similarity between selection of Grid (computation) resources and construction of financial portfolios. In our explorations of financial instruments and portfolios, we have found credit derivatives, and in particular collateralized debt obligations (CDOs), to offer an

interest analogy which also offers potential to quantify the (computational) portfolio risk as a price.

A CDO is a structured transaction that involves a special purpose vehicle (SPV) in order to sell credit protection on a range of underlying assets (see, for example, Tavakoli, 2008. p.1-6). The underlying assets may be either synthetic or cash. The synthetic CDO consists of credit default swaps (CDS), typically including fixed-income assets bonds and loans. The cash CDO consists of a cash asset portfolio. A CDO, and other kinds of structured investments, allows institutions to sell off debt to release capital. A CDO is priced and associated to measurements of riskiness that can be protected (for example, insured against the default on a particular loan). Protection is offered against specific risk-identified chunks of the CDO, called tranches. To obtain protection in each class, a premium is paid depending on the risk, reported in basis points, which acts like an insurance policy.

Consider, for example, a typical CDO that comprises four tranches of securities: senior debt, mezzanine debt, subordinate debt and equity. Each tranche is identified as having seniority relative to those below it; lower tranches are expected to take losses first up to specified proportions, protecting those more senior within the portfolio. The most senior tranche is rated triple-A, with a number of other possible ratings such as BB reflecting higher risk below this; the lowest tranche, equity, is unrated. The lowest rated should have the highest returns, but incorporates the highest risk. The senior tranche is protected by the subordinated tranches, and the equity tranche (first-loss tranche or "toxic waste") is most vulnerable, and requires higher compensation for the higher risk.

Correlation is used to describe diversification of CDOs; that is, the combined risk amongst names within CDO's tranches. A *default* correlation measures the likelihood that if one name within a tranche fails another will fail also. However, incorrect assumptions of correlation could lead to inaccurate predictions of quality of a CDO. Structuring means that a few high risk names - with potentially high returns – can be subsumed amongst a much larger number of low risk names while retaining a low risk on the CDO overall.

This notion of *default*, and the correlation of default, is initially of interest. The price of a Cloud resource will only likely drop to zero if the Cloud provider fails. However, there is a possibility of a Cloud resource *performance* dropping to zero. In addition, as we have seen, there is a risk that the performance drops below a particular threshold, at which point it presents a greater risk to the satisfaction of the SLA. To offer a portfolio of resources, each of which has an understood performance rating that can be used to assess its risk and which can change dynamically, suggests that the SLA itself should be rated such that it can be appropriately priced. Our explorations, therefore, have involved evaluating the applicability of principles involved with CDOs to Cloud resources, and this has a strategic fit with autonomous Clouds (Li, Gillam and O'Loughlin 2010, Li and Gillam 2009a&b) although there is much more work to be done in this direction.

6 CONCLUSIONS

In this paper we have considered the automatic construction of Service Level Agreements (SLAs) that would incorporate expectations over quality of service (QoS) by reference to benchmarks. Such SLAs may lead to future markets for Cloud Computing and offer opportunities for Cloud Brokers. The CDO model offers some useful pointers relating to tranches, handling failures, and offering up a notion of insurance. At large scales, this could also lead to a market in the resulting derivatives. We have undertaken a large number of benchmark experiments, across many regions of AWS, in Rackspace UK and US, in various datacenters of IBM's SmartCloud, and also in an OpenStack installation at Surrey. A number of different benchmarks have been used, and a large number of different machine types for each provider have been tested. It is not possible to present the results of findings from all of these benchmark runs within a paper of this length, and in subsequent work we intend to extend the breadth and depth of our benchmarking to obtain further distributions over time in which it may be possible to obtain more accurate values for variance and identify trends and other such features.

In terms of other future work, the existence of AWS CloudWatch and the ability to create alarms suggests at minimum the results we have obtained can be readily fed into a fairly basic set of SLAs and this will subsequently offer a useful baseline for the treatment of benchmark data. The emergence of KPIs in WS-Agreement also offers an opportunity in this direction, depending on the drive put into its future development, although benefits are likely less immediate; a common definition of metrics and their use will certainly be beneficial. Finally, our use of CDOs in risk assessment shows promise, and further experiments which are aimed at demonstrating this approach are currently under way.

7 ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support of the UK's Engineering and Physical Sciences Council (EPSRC: EP/I034408/1), the UK's Department of Business, Innovation and Skills (BIS) Knowledge Transfer Partnerships scheme and CDO2 Ltd (KTP 1739), and Amazon's AWS in Education grants.

8 REFERENCES

Amato, A., Liccardo, L., Rak, M. and Venticinque, S. (2012), SLA Negotiation and Brokering for Sky Computing. In Proc. 2nd International Conference on Cloud Computing and Services Science, CLOSER 2012, Porto, Portugal, 18-21 April.

Andrix A., Czajkowski K., Dan A., Keay K., et al. (2007), *Web Services Agreement Specification (WS-Agreement), Grid Resource Allocation Agreement Protocol (GRAAP)* WG, http://forge.gridforum.org/sf/projects/graap-wg

Bose, S., Pasala, A., Ramanujam A, D., Murthy, S. and Malaiyandisamy, G. (2011), SLA Management in Cloud Computing: A Service Provider's Perspective. In Buyya, Broberg and Goscinski (eds.), Cloud Computing: Principles and Paradigms, John Wiley & Sons, Inc.

Czajkowski, K., Foster, I., Kesselman, C., Sander, V., Tuecke, S. (2002), SNAP: A Protocol for Negotiation of Service Level Agreements and Coordinated Resource Management in Distributed Systems, submission to Job Scheduling Strategies for Parallel Processing Conference (JSSPP), April 30.

Emeakaroha, V.C., Brandic, I., Michael, M., and Dustdar, S. (2010). Low level Metrics to High level SLAs - LoM2HiS framework: Bridging the gap between monitored metrics and SLA parameters in cloud environments. Information Systems Journal, 2:48–54.

Gens F. (2009), *New IDC IT Cloud Services Survey: Top Benefits and Challenges, IDC research*, http://blogs.idc.com/ie/?p=730 (Jun 2011)

Gillam, L., Li, B., O.Loughlin, J. and Singh Tomar, A.P. (2012) Fair Benchmarking for Cloud Computing Systems, Technical Report, University of Surrey, UK, http://tinyurl.com/FairBenchReport

Li, B., Gillam, L., and O'Loughlin, J. (2010) Towards Application-Specific Service Level Agreements: Experiments in Clouds and Grids, In Antonopoulos and Gillam (Eds.), Cloud Computing: Principles, Systems and Applications. Springer-Verlag.

Li, B., and Gillam, L. (2009a), Towards Job-specific Service Level Agreements in the Cloud, Cloud-based Services and Applications, in 5th IEEE e-Science International Conference, Oxford, UK.

Li, B., and Gillam, L. (2009b), Grid Service Level Agreements using Financial Risk Analysis Techniques, In Antonopoulos, Exarchakos, Li and Liotta (Eds.), Handbook of Research on P2P and Grid Systems for Service-Oriented Computing: Models, Methodologies and Applications. IGI Global.

Ludwig H., Nakata T., Waldrich O., Wieder P., and Ziegler W., (2006) Reliable Orchestration of Resources using WS-Agreement. In LNCS 4208:753- 762, Springer.

Jeffery (edt.), K. (2004). Next Generation Grids 2: Requirements and Options for European Grids Research 2005-2010 and Beyond.

Kenyon, C. and Cheliotis., G. (2002). Architecture requirements for commercializing grid resources. In 11th IEEE International Symposium on High Performance Distributed Computing (HPDC'02).

Kerstin, V., Karim, D., Iain, G. and James, P. (2007). AssessGrid, Economic Issues Underlying Risk Awareness in Grids, LNCS, Springer Berlin / Heidelberg.

Leff, A., Rayfield T. J., Dias, M. D. (2003). Service-Level Agreements and Commercial Grids, IEEE Internet Computing, vol. 7, no. 4, pp. 44-50.

P Meland, P.H., Bernsmed, K., Jaatun, M.G. and Undheim, A. (2012), Expressing Cloud Security Requirements in Deontic Contract Languages. In Proc. 2nd International Conference on Cloud Computing and Services Science, CLOSER 2012, Porto, Portugal, 18-21 April.

Rak, M. and Ficco, M. (2012), Intrustion Tolerance as a Service – A SLA-based Solution. In Proc. 2nd International Conference on Cloud Computing and Services Science, CLOSER 2012, Porto, Portugal, 18-21 April.

Rogers O and Cliff D (2012), A financial brokerage model for cloud computing. Journal of Cloud Computing: Advances, Systems and Applications 2012, 1:2 (12 April 2012)

Seidel J., Waldrich O., and Yahyapour R., (2007) Using SLA for resource management and scheduling - a survey. Technical report, Institute on Resource Management and Scheduling, CoreGRID - Network of Excellence.

Sturm R, Morris W, Jander M. (2000), Foundation of Service Level Management. SAMS publication.