# ADDING CLOUD PERFORMANCE TO SERVICE LEVEL AGREEMENTS

Lee Gillam, Bin Li and John O'Loughlin
*Department of Computing, University of Surrey, Guildford, U.K.*

Keywords: Cloud Computing, Cloud Brokerage, Service Level Agreement (SLA), Quality of Service (QoS), Cloud Benchmarking, Cloud Economics.

Abstract: To some the next iteration of Grid and utility computing, Clouds offer capabilities for the high-availability of a wide range of systems. But it is argued that such systems will only attain acceptance by a larger audience of commercial end-users if binding Service Level Agreements (SLAs) are provided. 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 explore a set of benchmarks that offer both an interesting characterisation of resource performance variability, and identify how such information might be used both directly by a user and indirectly via a Cloud Broker in the automatic construction of SLAs.



## 1 INTRODUCTION

Use of Clouds has become a key consideration for businesses seeking to manage costs transparently, increase flexibility, and reduce the physical and environmental footprint of infrastructures. Instead of purchasing and maintaining hardware and software, organizations and individuals can rent a variety of (essentially, shared) computer systems. Under the Cloud banner, much is rentable – from actual hardware, to virtualized hardware, through hosted programming environments, to hosted software. Rental periods are variable, though not necessarily flexible, and providers may even bill for actual use below the hour (indeed, down to three decimal places), or for months and years in advance. Some would argue about which of these really differentiates Cloud from previous hosted/rented/ outsourced approaches, although that is not our concern here.

The idea that Cloud is somehow generically cheaper than any owned infrastructure is variously open to challenge, and certainly there are many examples 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. We would contend that the price differential is at present more likely to be advantageous for Cloud where utilization has significant variability over time – alleviating the costs of continuous provision just to meet certain peak loads. Further, it will be only be advantageous where the incorporated cost of the rented system is cheaper on the whole than the total cost of internal ownership (labour, maintenance, energy, and so on) plus the cost of overcoming reluctance of various kinds (due to sunk investments, internal politics, entrenched familiarity, and other organisational inhibitors). Where Cloud actually ends up being more expensive, the potential benefit in flexibility can offset this. Dialling up or down capacity in the right quantities 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 the offer additional value.

At present, however, cost and flexibility only extends so far. An end user would not be able to run, say, a specific computational workload with fine grained control over computational performance at a given time, to readily obtain the best price or a set of comparable and re-rankable price offers against this, or to easily manage the risk of it failing and being able to re-run within a limited time without suffering a high cost-of-cure. For some applications, a particular result may be required at a specific time – and beyond that, the opportunity is lost and potentially so is the customer or perhaps the

Gillam L., Li B. and O'Loughlin J..
ADDING CLOUD PERFORMANCE TO SERVICE LEVEL AGREEMENTS.
DOI: 10.5220/0003974406210630
In *Proceedings of the 2nd International Conference on Cloud Computing and Services Science* (CloudSecGov-2012), pages 621-630
ISBN: 978-989-8565-05-1

investment opportunity; the former case would relate to high latency on serving out webpages during short popular periods, whilst the latter would feature in high frequency trading applications where microseconds of latency makes a difference.

Clouds need to emphasize measurement, validation and specification – and should do so against commitments in Service Level Agreement (SLA). Without assurances of availability, reliability, and, perhaps, liability, there is likely a limit to the numbers of potential end-users. This is hardly a new consideration: 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); the notion of Grids in the business sphere seems long gone now.

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 at specific times of the day. However, the limited commercial adoption reported of such commoditized infrastructures by the financial sector, appearing instead to maintain bespoke internal infrastructures, may suggest either that those organizations are remaining quiet of the subject to avoid media concern over the use of such systems, or are concerned with maximizing their existing infrastructure investment, or that the lack of assurances of availability, reliability, security, and liability is a hindrance.

In our work, we are aiming towards building liability more coherently into SLAs. We consider that there is potential for failure on the SLA due either to performance variability, or due to failure of a part of, or the whole of, the underlying infrastructure. We are working towards an application-specific SLA framework which would offer a Cloud Broker the potential to present offers of SLAs for negotiation on the basis of price, performance, and liability – price can incorporate the last two, but transparency is likely to be beneficial to the end user.

The application-specific SLA should necessarily be machine-readable, rather than requiring clumsy human interpretation, negotiation, and enforcement, and the Broker should be able to facilitate autonomic approaches to reduce the impact of failure and promote higher utilization.

Following Kenyon and Cheliotis (2002), we are constructing risk-balanced portfolios of compute resources that could support a different formulation of the Cloud Economy. Our initial work in financial risk was geared towards greater understanding of risk and its analysis within increasingly complex financial products and markets, though many of these products and markets have largely now disappeared.

Related work on computational risk assessment using financial models appears to treat the underlying price as 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 certain extremes of variance which could be quite difficult to model). Our approach makes price variable in line with performance, so price volatilities will follow performance volatilities over time. Additionally, we need to factor in the effect of failure of the underlying resource, which it is difficult to account for in traditional risk models since it is a very rare, and often unseen, event in price data. The Cloud risk, then, involves the probability of a partial or complete loss of service, and the ability to account for that risk somehow in an offer price which includes it – essentially, building in an insurance policy.

In this paper, we present our view of how to incorporate QoS into SLAs and to model the risk such that it can be factored into pricing. 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 variability in Cloud resource performance – our use of benchmarks is a means to an end, not an end in itself. Section 4 addresses price-based variability in performance, and section 5 offers suggestions for bringing risk assessment and SLA together. We conclude the paper, and offer a few future work suggestions.

# 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 high level, Service Level Agreements can cover such organisational matters as how promptly a telephone call will be answered by a human operator. In computing in general, and in Cloud Computing in particular, such SLAs are largely concerned with an overall level of service availability and, indeed, may detail a number of aspects of the service for which there is "disagreement". 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 of Your Content.".

Such an SLA is *non-negotiable*, and typically written to protect and favour the provider. Negotiable SLAs may also be available, but negotiation in mainstream provision is likely to be a drawn out process, require a longer term commitment – not really geared towards Cloud provision so much as datacentre rental - and yet still be at a relatively high level.

Presently, SLA clauses related to liability are likely to be few and limited to outright failure. In the event, service credits may be offered – rather than a refund – and the consumer can either stick with it or go elsewhere. Standard SLAs for Cloud providers tend to fit such a description.

AWS only began offering a general SLA in 2008, although some customers had already lost application data through an outage in Oct 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." As highlighted in Table 1, things do indeed fail. However, there remains a knock-on effect. But it also becomes apparent that even a slight SLA offers some form of compensation (compare 2011 to 2007). The specific clause for Service Credits for AWS suggests that 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 periods of 4 minutes might count as one period or might not count at all. In addition, "Region Unavailable" is some way different to the lack of availability of a given resource within a region. Also, such a clause is not applicable to a badly running instance. Such conditions can impact on performance of a supported application and so also have a cost implication. Ideally, commercial systems would provide more than merely "best effort" or "commercially reasonable effort" over some long period, and would be monitored and assure this on behalf of the consumer. Furthermore, such SLAs should have negotiable characteristics at the point of purchase with negotiation 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 considered the development of Cloud SLAs and emphasised the importance of service level management, system redundancy and maintenance,

Table 1: A few AWS outages, and the emergence of use of the SLA.

| 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.* |

security and privacy, and monitoring, as well as machine-readability. In addition, they describe use of SLAs in relation to Cloud brokers. Machine-readability will be important in supporting large numbers of enquiries in a fast moving market, and two frameworks address machine-readable SLA specification and monitoring: 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. Through WS-Agreement, an entity can construct an SLA as a machine-readable and formal contract. The content specified by WS-Agreement builds on SOA (driven by agreement), where the service requirement can be achieved dynamically. 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 will vary 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 denotes 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. However, we do not consider availability to be the same as performance, which could vary substantially whilst availability is maintained – put another way, contactable but impossibly slow.

**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.

**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 the SLA. Maintenance and recovery are just two aspects of such 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.

Since PoF should be grounded in Performance, and should offer a better basis for presenting Service Availability, in the remainder of this paper we focus primarily on performance. Performance variability of virtualized hardware will have an impact on Cloud applications, and we posit that the stated cost of the resource, typically focussed on by others in relation to Cloud Economics, is but a distraction – the performance for that cost is of greater importance and has greater variability: lower performance at the same cost is undesirable but cannot as yet be assured against. To measure performance variability, we investigate a small set of benchmarks that allow us to compare performance both within Cloud instances 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 on a more dynamic problem.

# 3 PERFORMANCE BENCHMARKS

We selected benchmarks to run on Linux distributions in 4 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, and to assess present performance in relation to HPC type activities. Literature on Cloud benchmarking already reports CPU, Disk IO, Memory, network, and so on, usually involving AWS. Often, 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 due to the underlying physical resources.

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

In this paper, we will 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 Figure 1 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

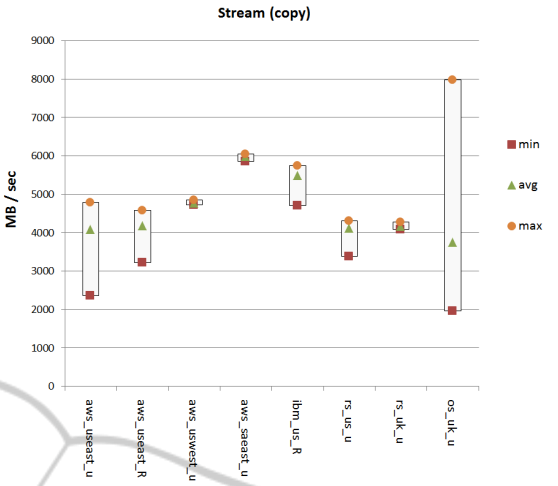


Figure 1: STREAM (copy) benchmark across four infrastructures. Labels indicate provider (e.g. aws for Amazon), region (e.g. useast is Amazon's US East) and distribution (u for Ubuntu, R for RHEL).

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 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.

Figure 2 shows 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 also for AWS US east region because we assumed these would be reasonably comparable with other regions.

AWS instances produce largely similar results,

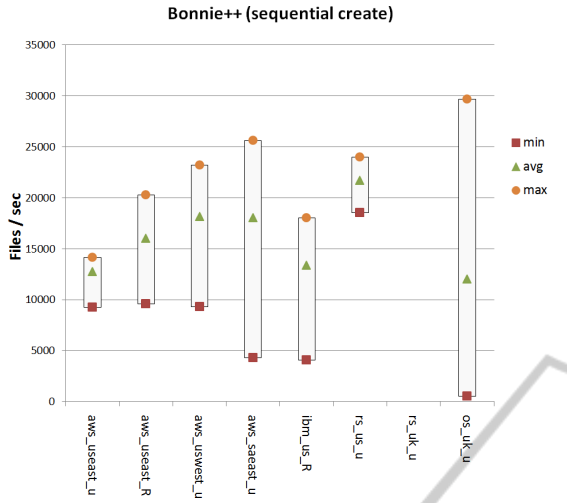


Figure 2: Bonnie++ (sequential create) benchmark across four infrastructures.

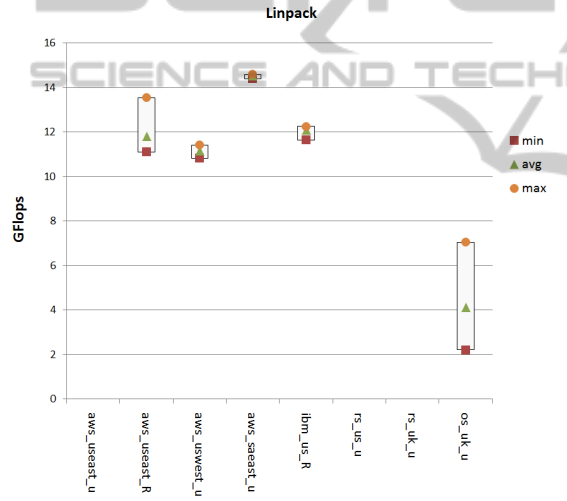


Figure 3: LINPACK (25000 tests) benchmark across tested infrastructures.

without significant variance. Our OpenStack Cloud again suffers in performance – perhaps a reflection on the age of the servers used in this setup.

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.

Figure 4 below indicates 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.

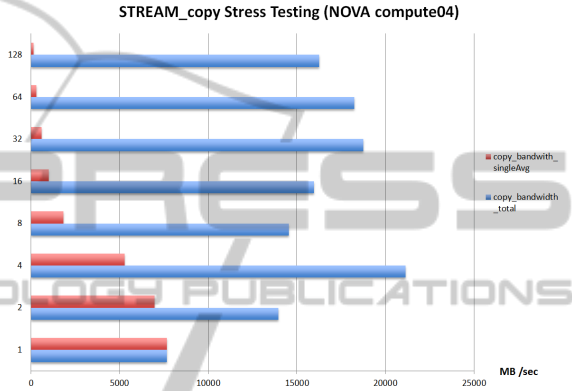


Figure 4: STREAM (copy) benchmark stress testing on Nova compute04, showing diminishing bandwidth per machine instance as the number of instances increases, and variability in overall bandwidth.

## 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. The latest WS-Agreement specification (1.0.0) helpfully separates out the static resource properties – such as amount of memory, numbers of CPUs, and so forth

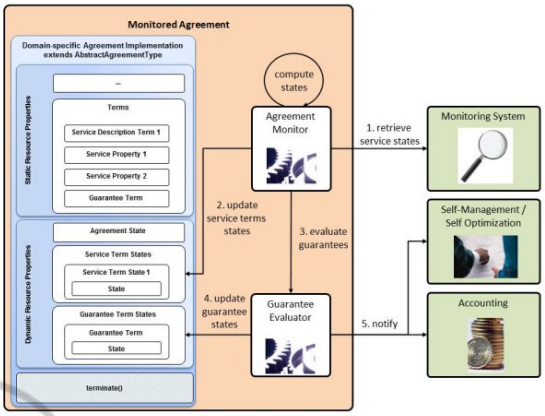


Figure 5: OGF Agreement Monitoring (source: WSAG4J, Agreement monitoring).

– from the dynamic resource properties – typically, limited to response times (Figure 5). The dynamic properties are those that can vary (continuously) over the agreement lifetime.

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.

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%).

```
<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/).

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.

## 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.

# 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.

# REFERENCES

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

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.

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

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.

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.

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

Tavakoli, M. J. (2008). Structured Finance and Collateralized Debt Obligations: New Developments in Cash and Synthetic Securitization, 2nd ed, John Wiley & Sons.