Quantifying Uncertainty for Preemptive Resource Provisioning in the Cloud

Marin Aranitasi
Polytechnic University of Tirana,Faculty of Information Technology, Department Fundamentals of Informatics
Tirana, Albania
Email:maranitasi@fti.edu.al

Benjamin Byholm, Mats Neovius
ÅboAkademi University, Faculty of Science and Engineering, Department of Information Technologies
Turku, Finland
Email: bbyholm@abo.fimneovius@abo.fi

Abstract—**To satisfy quality of service requirements in a cost-efficient manner, cloud service providers would benefit from providing a means for quantifying the level of operational uncertainty within their systems. This uncertainty arises due to the dynamic nature of the cloud. Since tasks requiring various amounts of resources may enter and leave the system at any time, systems plagued by high volatility are challenging in preemptive resource provisioning. In this paper, we present a general method based on Dempster-Shafer theory that enables quantifying the level of operational uncertainty in an entire cloud system or parts thereof. In addition to the standard quality metrics, we propose monitoring of system calls tocapture historical behavior of virtual machines as an input tothe general method. Knowing the level of operationaluncertainty enables greater accuracy in online resourceprovisioning by quantifying the volatility of thedeployedsystem.**

Keywords- Cloud uncertainty ; Resource provisioning; System calls;.

I. INTRODUCTION

Day-to-day operations in cloud computing are plagued with uncertainty. Service providers must quickly decide which resource management actions to take based on incomplete information. Typical challenges according to the National Institute of Standards and Technology (NIST) include timely provisioning and release of resources [1]. NIST's definition of cloud computing lists three service models: software, platform and infrastructure, cf. Fig 1. This article applies to the first two models, since SaaS is not concerned with Virtual Machines (VM). Further, NIST defines cloud computing as follows: "cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." [1].

The NIST definition gives five essential characteristics of cloud computing: on-demand self-service, broad network access, resource pooling, rapid elasticity or expansion, and measured service. Rapid elasticity or expansion combined with measured service is necessary to handle the volatility in service demand in a cost-efficient manner. However, the high amount of uncertainty makes cost-efficiency difficult to achieve in practice. By quantifying this level of operational uncertainty in a cloud system, the parameters to utilize the resources more optimally are available to the service providers.

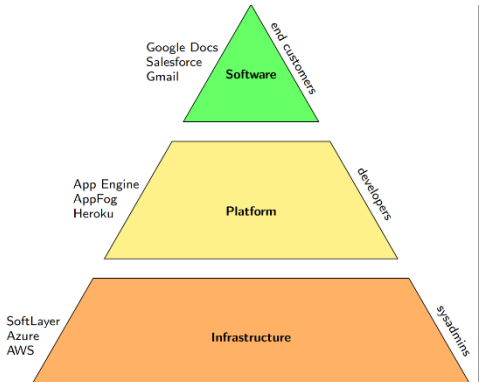

Figure 1. Cloud services

A reactive system will always lag behind the now, which is why predictive resource provisioning is desirable. However, prediction introduces more error, and thus more uncertainty, into the system [2]. Traditional performance models for capacity planning do not work for volatile cloud systems, as they require a priori knowledge of the system. Automated empirical derivation of performance models is possible [3], but does not scale in highly dynamic environments [2].

In cloud systems, there are different types of uncertainty. Table 1 outlines those related to resource provisioning. The two facets of uncertainty experienced by the customer relate to the uncertainty on information (accuracy, timeliness etc.) and on service (overloading, clogging). However, as the reasons are highly coupled, where one leads
to the other, customers experience the combined effects as unreliable, slow service. In this paper, we quantify the total uncertainty daunting the customer from the service provider's perspective.

As such, we define uncertainty as the amount of incomplete information, neither supported nor rejected by the evidence. For example, while a colorblind person can tell whether a light is on or off, they may not be able to tell

if the light is red or green. The available evidence only confirms or denies that the light is on.

Table 1: Cloud computing parameters and main sources of their uncertainty [4]

Source of Uncertainty / **Parameters**	Data	Virtualization	Jobs Arrival	Resource availability	Elasticity	Elastic provisioning
Effective performance		x	x	x	x	x
Available memory	x		x	x	x	x
Available storage	x			x		x
Resource Capacity		x	x		x	x
Network Capacity	x					x

We use historical data to form predictions of the future. In a stable system, the level of uncertainty goes towards zero, indicating that the system's resource demand is fully predictable, which does not apply to On-Line Transaction Processing systems. For this, we use Dempster-Shafer theory to quantify the level of uncertainty in cloud systems, mentioned as a possibility by Tchernykh. al [5]. We derive this from quantified uncertainty through monitoring VMs and aggregating results in the cloud hypervisor. Hence, the VM stakeholder gains information about anomalies [6, 7], while the cloud provider gains a quantified level of uncertainty, facilitating cost-efficient resource provisioning. In addition to the standard quality metrics, we also propose monitoring of system calls to capture historical behavior of virtual machines as an input to the general method.

The service provider seeks to optimize supply with respect to demand while adhering to the Service Level Agreement (SLA). The smallest VM available for provisioning determines the quantization of the scaling operations. Over-allocation means an opportunity cost, under-allocation means lost revenue. The scope of this paper is quantification of uncertainty; hence, we will not go into decision-making.

II. BACKGROUND AND MOTIVATION

Research on uncertainty in computing are plentiful with examples in computational biology, decision-making, statistical model checkers and as a foundation for logics. These domains frequently assume readily quantified uncertainty derived from repeated tests in a laboratory environment by the manufacturer, e.g. a sensor's uncertainty. The problem in software construction is then calleddependability [8] or resilience [9]. In dependability fault prevention, tolerance, removal and forecasting are technologies for resilience, i.e. they view the same problem from a slightly different angle. The obvious solution in theseis redundancy that fundamentally raise the probability of satisfiability. However, redundancy through overdimensioning is not financially sustainable. This merely indicate the extent to which assuming that the event will conform to a given contract. Parallel to this, the domain of statistical model checkers do quantify stochastic systems by a level of reliance [10]. These require an assumed probability distribution in terms of an automaton [11] to, for example, construct a Markov Chain on which to run simulations. The final level of reliance must then exceed the given safety integrity level as defined in the requirements.

Another track of quantifying uncertainty is that of fuzzy logics. Fuzzy logics approach the problem with the assumption that the input is inherently inaccurate, e.g. a linguistically vague proposition [12] such as tall. As this is not the case in computerized systems, where events measured or of logical nature with parameterized levels, the logic is motivated to be closer to probabilities than fuzzy sets. That is, as we can measure an uptime of a system in percentage, we lose information if we fuzzyfy this crisp value to, for example, "very stable".

Uncertainty specifically in the context of cloud computing has been defined as the "difference between the available knowledge and the complete knowledge" [5]. The definition faces by the challenges of defining complete knowledge. We claim the "complete knowledge" to be impossible to quantify or parametrize and hence, to be unknown voiding the level of available knowledge as well.

In addition to cost, performance and reliability, Trenzet al. [24]identified privacy, security and availability as major sources of uncertainty.These may have devastating consequences, such as in the case of data breaches on Sony PlayStation [13] and Dropbox [14] or in case of social attacks. However, as these consider uncertainty as an attribute of reliability often being a probability, the uncertainty considered in this paper is very different.

The motivation of this paper is merely in quantifying uncertainty in that the resource demand of a cloud system is constant. From this, the level of uncertainty indicate the volatility in this demand and indicates thereby the level underutilization from the rule of thumb, i.e. the tradeoff between the SLA and revenue.

III. HOW WE MIGHT APPROACH THESE ISSUES

The model quantifying uncertainty relies on performance metrics and system calls. We obtain the performance metrics from the VMs and system call traces from an instrumented hypervisor. The model's output is then a level of certainty of consistent behavior; or conversely, the level of behavioral uncertainty. Letting the certainty count for normality, the model also defines this and hence, using it for detecting anomalies is possible. This normality will define the way the system works in a state of behavioral certainty, with any anomaly indicating a situation calling for further attention. Also by the monitoring of the hypercalls we can define the resource usage form the different VMs. This will help in further analysis of the resource provisioning in order to

maintain the QoS for the client but also not to stress or overload the physical machines where the VMs resides.

A. *Cloud architecture*

A hypervisor is software that exists outside of a guest operating system to intercept the commands sent to the computer hardware. The term "hypervisor" comes from the different levels of an operating systems kernel; it performs actions with more authority than the "supervisor" level, hence, *hyper*-visor.Popular hypervisors used in industry include XEN and KVM. They have some differences but at the end, they provide the same services: allowing multiple guest VMs to share the system resources. However, the VMs have to transfer control to the hypervisor to execute sensitive and privileged instructions on the HW after which the control is returned to the VM. Hence, the hypercalls are very similar to system calls in normal operating systems. Hypercalls, as system calls, differ depending if the HW

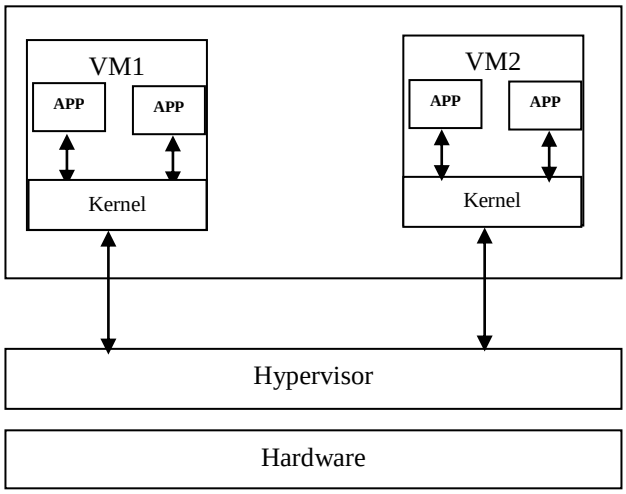

Figure 2. Cloud architecture

architecture [15].

B. *System calls*

Asystem call is an atomic request in a Unix-like operating system made via a software interrupt by an active process for a service performed by the kernel [16].System calls are a

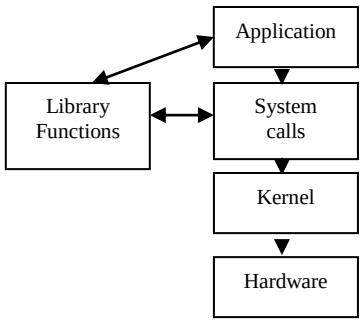

Figure 3. System calls

direct entry point into the kernel through which programs request services from the kernel.Developers gain access to the system calls through an application-programming interface (API). The API functions invoke the system calls. This is illustrated in Figure 1. By using the API, certain benefits can be gained:

- Portability: as long a system supports an API, any program using that API can compile and run.
- Ease of Use: using the API can be significantly easier than using the actual system call.

The system calls are plentiful and vary between operating systems, with Linux kernel having 300+ system calls and Windows 7 having close to 700. These can be categorized to five different categories [17]:

1. **Process control**is a running program that needs to be able to stop execution either normally or abnormally. When execution is stopped abnormallytypically, a dump of the memory is taken to be examined by a debugger.
2. The **file management** system callsinclude create(), delete(), read(), write(), reposition(), or close(). In addition, there is a need to determine the file attributes – get and set file attribute. Often the OS provides an API to make these system calls.
3. The **device management** process requires several resources to execute, if these resources are available, they will be granted and control returned to the user process. These resources are also thought of as devices. Some are physical, such as a video card, and others are logical, such as a file.User programs *request* the device, and when finished they *release* the device. Similar to files, we can *read*, *write*, and *reposition* the device.
4. The **information management**system call exists for transferring information between the user program and the operating system. An example of this is *time*, or *date*. The OS also keeps information about all its processes and provides system calls to report this information.
5. The **communication**system call exists in two models of interprocess communication, the message-passing model and the shared memory model.
   - Message passing uses a common mailbox to pass messages between processes.
   - Shared memory use certain system calls to create and gain access to regions of memory owned by other processes. The two processes exchange information by reading and writing in the shared data.

For our model, we monitor 2 categories of system calls:

1. File management.
2. Communication

For the file management *F*category we propose to monitor read (), write (), delete () and create () system calls and for the *C*communication category we propose to monitor accept (), socket (), connect () system calls. Thus, the set of system / hyper calls $X = F \cup C = \{\mathrm{r}, \mathrm{w}, \mathrm{d}, \mathrm{cr}, \mathrm{a}, \mathrm{s}, \mathrm{co}\}$. We will denote this system / hypercallshereafter merely as *calls*. We chose to use these in the model because Cloud Security Alliance report[18]note thesecalls to be frequentin the threats captured in the cloud systems, and because these are used when applicationsrequest resources from the VM. Thus, by monitoring these we can analyze the resource usage by the VMs and use this analysis for quantifying uncertainty.

C. *System and hyper call patterns*

A call is by nature executed in an atomic manner and they are exclusive. This means that a call is run one at a time from the beginning until the end without interrupts omitting race conditions. Moreover, the set of monitored X is exhaustive. Given this, the log of calls serves as the history of the cloud system. Analyzing this history provides evidence for the normal behavior of the system, i.e. the certainty of continuation. This certainty outlines the justifiable level to which assuming the cloud system to continue to operate as before. In the special case of a stable call pattern such as a reoccurring ping, the certainty tends towards full certainty providing a very concrete normal state of the system.

In addition to stable systems, discovering behavioral patternsprovides a basis for increased certainty on the upcoming resource demand. These patterns would extend the definition of normality over some domain. These domains vary and may depend on external events, such as being context dependent. Whatever the underlying reasons, to quantify uncertainty on the calls, we teach the model by the distribution of the monitored calls*X*. If any change in the distribution occurs, for example due to migration of VMs or a change in the utilization of the VM, this is typically detected by a momentary increasein the level of uncertainty. Gradually, however, depending on the consistency of calls, the model will adjust to the new normal.

D. *Theory of evidence*

On the problem and domain outlined in this paper, we propose to use Dempster-Shafer theory, aka, evidence theory. The evidence theory is a generalization of Bayesian theory of subjective probabilities on a set of exclusive and exhaustive events X. The powerset 2^X denotes all combinations of calls, realistically enabling comparing any category of callsto discover new domain specific patterns; in this paper file management and communication categories. The mass *m* is the level of certainty on a set of events where $m : 2^X \rightarrow [0,1]$, $m(\emptyset) = 0$ and $\sum_{2^X} m = 1$. On this, the belief*bel*of a subset of outcomes $A \subseteq X$ is $bel(A) = \sum_{x \subseteq A} m(x)$ and plausibility *pl* is $pl(A) = \sum_{x \cap A \neq \emptyset} m(x)$ as for the possibilityof this outcome. This implies that $bel < pl$ whenever $m(X) \neq 0$and$A \subset X$. The semantics of this is that the difference between *bel* and *pl* denote the quantified uncertainty. The complement of a set of events Adenoted $\bar{A}$is the evidence against this proposition, i.e. $pl(A) = 1 - bel(\bar{A})$.Thus, with respect to DS-theory, *bel* indicates the certainty in favour of a proposition and 1-*pl*the certainty against this proposition and thereby, *pl* - *bel* is the uncertainty.

E. *Quantifying uncertainty by calls*

Having the Dempster-Shafer theory of evidence as a solid mathematical foundation, we derive the values for each call $x \in X$ from the log. Here each call is a piece of evidence defined as an *experience* denoted *Exp*.Inspired by Krukow's [19] and Teacy et al. [20] and continuing related work [21, 22, 23, 7, 25], an *Exp*is defined as a four tuple $(\delta, \epsilon, \zeta, \eta)$where δ is the subject system's and application's identification, ϵ the timestamp, ζ as a subset of calls and η a score $\in \{0, 1\}$. An example *Exp* is $(vm_1, time, w, 1)$ indicating that a *write* call from a service at *time*originating from $vm_1$was recognised.The history of *Exp*is a set of such tuples, i.e. $\{(\delta, \epsilon, \zeta \subseteq X, \eta)\}$. Adding a new experience is straightforward: $\{(\delta, \epsilon, \zeta \in X, \eta)\} \cup (\delta, time, w, 1)$.

A projection on the history of experiences is $Exp(\delta, t, x \subseteq X) = \{\eta\}$ where*t* may define a timespan. Whenever recent experiences weigh more,decay d_{ϵ_m} at ϵ_m is defined by a function $\lambda \in [0,1]$ with the semantics of higher indicates lower decay with 1 indicating no decay and 0 vacuous experiences. Thus, $d_{\epsilon_m}\left(Exp(\delta, \epsilon_i, \zeta \subseteq X)\right) = \{(\lambda^{\epsilon_m - \epsilon_i} * \eta)\}$ is a set of experiences given by the projection over a timespan.To aggregate the set of decayed score called an abstraction, we apply simple summation, i.e. $Abs_{\epsilon_m}\left(Exp(\delta, \epsilon_i, \zeta \subseteq X)\right) = \sum_{d_{\epsilon_m} Exp(\delta, \epsilon_n, \zeta \subseteq X)} \eta$ providing the $bel(\zeta)$

To quantify the uncertainty a non-informative priori weight *W*is defined. Let *W*= 2.Acquiring the evidence against a proposition $\bar{\zeta}$ is similar $Abs_{\epsilon_m}\left(Exp(\delta, \epsilon_i, \bar{\zeta} \subseteq X\right.$providing$bel(\zeta)$, hereafter called $dis(\zeta)$ as for disbelief. Then, the uncertainty *u*is defined as the relation of the non-informative priori weight with respect to the evidences, i.e. $\frac{W}{bel(\zeta) + dis(\zeta) + W}$. Replacing the dividend by $bel(\zeta)$ and $dis(\zeta)$ gives the normalised *b* and *d*. Having a tuple *b*, *d*, *u*maps this method to Subjective Logic and enables illustration in abarycentric coordinate system [26, 27].

IV. CONCLUSION

We presented a model to quantify uncertainty for preemptive resource provisioning in the cloud. Over-provisioning constitutes an opportunity cost, while under-provisioning constitutes lost revenue. Service providers can make more informed decisions when provisioning their cloud systems by accounting for their inherent uncertainty. Our model is based on Dempster-Shafer theory and quantifies the level of

uncertainty, that which is neither confirmed or denied by available evidence. We use historical data of performance normal behavior, we can reliably measure the level of uncertainty. In the future, we intend to validate the proposed method by implementing it and testing on realistic data.

REFERENCES

[1] US national institute of standards and technology.http://csrc.nist.gov/, 2016, accessed 15.1.2017

[2] A. Ashraf, B. Byholm, I. Porres, "Prediction-based VM provisioning and admission control for multi-tier web applications", Journal of Cloud ComputingAdvances, Systems and Applications, 5:15, 2016.

[3] B. Urgaonkar, P. Shenoy, T. Roscoe,"Resource overbooking and application profiling in a shared internet hosting platform", ACM Trans. Intern. Tech. 9, 1, Article 1, February 2009.

[4] J.Ramirez, A.Tchernukh, R. Yahyapour, U. Schwiegelshohn, A.Quezada, J.Gonzales, A.Hirales "Job allocation Strategies with user Run Time Estimates for Online Shceduling in Hierachal Grids." Journal of Grid Computing, 9:95-116, Springer. February 2011

[5] A. Tchernykh, U. Schwiegelsohn, V. Alexandrov, E.-G. Talbi, Towards understanding uncertainty in cloud computing resource provisioning, Proc. Comput. Sci., 51, pp. 1772–1781, 2015

[6] Audun Jøsang. "*Subjective Logic: A Formalism for Reasoning Under Uncertainty*", ISBN 978-3-319-42337-1, Springer Verlag, 2016

[7] M. Aranitasi, M. Neovius, "Anomaly Detection in Cloud Based Application using System Calls" CLOUD COMPUTING, The Eighth International Conference on Cloud Computing, GRIDs, and Virtualization, 2017.

[8] Jean-Claude Laprie, From dependability to resilience, in: IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2008.

[9] A. Legay, B. Delahaye, S. Bensalem,"Statistical model checking: an overview. In: Runtime verification (RV)", LNCS, vol 6418. Springer, Berlin, pp 122–135, 2010

[10] P. Bulychev, A. David, K. G. Larsen, A. Legay, M. Mikucionis, D. B. Poulsen. "*Checking & Distributing Statistical Model Checking*." 4th NASA Formal Methods Symposium, pages 449-463, LNCS 7226, Springer, 2012

[11] A. Jøsang, "A logic for uncertain probabilities," *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 9, no. 3, pp. 279-311., 2001

[12] H. Liu, "Software Performance and Scalability: A Quantitative Approach", Wiley Publishing, ISBN: 0470462531, 2009

[13] E. Petterson, "Sony to pay as much as $8 million to settle data breach case", Bloomberg Technology, 2015https://www.bloomberg.com/news/articles/2015-10-20/sony-to-pay-as-much-as-8-million-to-settle-data-breach-claims accesed on 13.01.2017.

[14] S. Yin, "Dropbox accounts were accessible by anyone fo four hours on Sunday" PCMag UK, 2011.http://uk.pcmag.com/storage-devices-reviews/9092/news/dropbox-accounts-were-accessible-by-anyone-for-four-hours-onaccesed on 13.01.2017.

[15] N.Pitropakis, C. Lyvas, C. Lambrinoudakis, "The grater the power, the more dangerous the abuse: Facing malicious insiders in the cloud" The eighth International conference on Cloud Computing, GRIDs and Virtualization, Athens 2017

[16] Linux Information Project "System call definition" 2016 http://www.linfo.org/system_call.html accessed on 13.01.2017.

[17] Kansas State University. http://faculty.salina.k-state.edu/tim/ossg/Introduction/sys_calls.html, accesed on 29.03.2017

[18] Cloud security alliance, "Cloud Computing top threats in 2016".https://downloads.cloudsecurityalliance.org/assets/research/top-threats/Treacherous-12_Cloud-Computing_Top-Threats.pdfaccesed on 13.01.2017

[19] K. Krukow, "Towards a theory of trust for the global ubiquitous computer," PhD thesis, University of Aarhus, Denmark., 2006.

[20] W. Teacy,J. Patel, N. Jennings, and M. Luck, "TRAVOS: Trust and Reputation in the Context of Inaccurate Information Sources," Autonomous Agents and Multi-Agent Systems, vol. 12, no. 2, pp. 183-198. , 2006.

[21] M. Neovius, "Trustworthy Context Dependency in Ubiquitous Systems," TUCS dissertations nr. 151. PhD thesis, Turku, Finland, 2012.

[22] M. Neovius,M. Stocker,M. Rönkkö, and L. Petre, "Trustworthiness Modelling on Continuous Environmental Measurement," in Proc. of the 7th Int. Conf. on Environmental Modelling and Software, 2014.

[23] M. Neovius, "Adaptive Experience-Based Composition of Continuously Changing Quality of Context," in Int. Conf. on Adaptive and Self-Adaptive Systems and Applications, 2015

[24] M.Trenz, J.C. Huntgeburth, D.Veit "The role of Uncertainty in Cloud Computing Continuance: Antecedents, Mitigators, and Consequences, ECIS, 147. 2013

[25] M. Neovius, B. Duncan, "Anomaly Detection for Soft Security in Cloud Based Auditing of Accounting Systems", To appear in the 7th International Conference on Cloud Computing and Services Science, 2017.

[26] A. Jøsang, "A logic for uncertain probabilities," *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 9, no. 3, pp. 279-311., 2001

[27] A. Jøsang. "*Subjective Logic: A Formalism for Reasoning Under Uncertainty*", ISBN 978-3-319-42337-1, Springer Verlag, 2016