Toward a Solution for the Cloud Account Delegation Problem

Bradley Simmons, Mark Shtern, Marin Litoiu
York University
Toronto, ON, Canada

Michael Smit
Dalhousie University
Halifax, NS, Canada

Abstract

Cloud account delegation refers to the situation in which a cloud account owner (delegator) allows one or more second parties (i.e., delegates) to acquire and use cloud infrastructure resources from the owner's account at the expense of the owner. This situation can exist in research groups, cloud testbeds, university courses, software development teams, and in general in cases where pooled resources are managed as software-defined infrastructure. The delegator assumes the risk of inefficient, wasted, or abused resources, and must rely on delegates, who are often not experienced cloud users, using the virtual infrastructure effectively and responsibly. This paper introduces the cloud account delegation problem, including three primary categories of risk, introduces a solution outline, and identifies research challenges.

1 Introduction

Organizations are adopting cloud infrastructure [1] for development, production, research, and testing purposes; the model emerging in leading cloud providers (like Amazon Web Services, AWS) and multi-cloud solutions [2, 3] is to provide organizations with a single account; the organization may then delegate access to its users. Adam Smith presciently addressed the cloud account delegation problem when he suggested we cannot expect that managers of other people's money will "watch over it with the same anxious vigilance with which [they would] watch over their own" [4].

An example familiar to academic researchers is the use of public cloud resources for teaching and/or research. The delegator (professor) provides access to delegates (research assistants, students) in support of an objective (research project, assignment). Delegates, unfamiliar with the cloud and unaccustomed to associating virtual infrastructure with costs, happily run experiments, launch instances, over-provision, try new features, and leave unused resources running. The professor receives a monthly bill, and is responsible for paying that bill (even recipients of cloud education grants[1] must provide a credit card to cover spending beyond the amount of the grant). With some variation, this problem exists for other delegator/delegate pairs: manager and dev/test team, for example. This problem is described in more detail in §2.

The act of delegation itself is typically unavoidable: cloud resources can and should be used when they will help users achieve their objectives. The focus must be on solutions that enable cloud use while providing a mix of incentivizing, assisting, and ensuring that resources are used responsibly. We propose a solution in the form of a cloud broker middleware service (e.g., as in [5, 6]) through which the delegates interact with the cloud to acquire resources, and which allows for the specification (by the delegator) and enforcement of various constraints and objectives to help delegates manage their usage in a manner consistent with the delegator's preferences. This solution, which without loss of generality is targeted at the research/teaching example above, is described in §3.

2 The Delegation Problem

While there is a substantial body of research on managing cloud-based applications (i.e., inter-

[1] http://aws.amazon.com/grants/

dependent collections of cloud resources; see [7] for an overview), the management of disparate cloud resources launched by multiple users for varied purposes has received less attention. Tang et al. [8] identified a similar issue and proposed hierarchical policies, but otherwise researchers have done little to address the delegation issue. Current multi-user single-bill accounts provide some policy options which are not intuitive and require the delegator to understand a complex policy language[2], but do not yet support per-user resource utilization tracking, do not address the efficiency of resources, and are not intended to limit the spending of a user. We expect per-user resource monitoring to eventually be provided by cloud providers, and until then assume the use of a third-party monitoring tool with this feature (e.g., [9]).

We have categorized cloud account delegation risks that can lead to unnecessary financial costs into three main categories: i) abuse of resources, ii) user inattention and negligence, and iii) complexity of public cloud pricing policies.

The **abuse of resources** is not necessarily malicious; it can be any situation in which a delegate acquires more resources than they actually require for accomplishing a specific task. It might be launching a very large (and expensive) instance "just in case". There are also malicious types of abuse; for example, a user limited to launching one virtual machine instance (VMI) can repeatedly launch and terminate one VMI; the typical cloud provider charges the hourly rate for each portion of an hour.

Referring to **user inattention and negligence** is not intended to be pejorative; rather, it recognizes that delagatees see virtual infrastructure abstractly, never see a monthly bill, and in any case are not incentivized to be careful with the account budget. Even the best-intentioned user can forget about running VMIs, or not notice a compromised machine causing $200/day in bandwidth charges.

The **complexity of public cloud pricing** can result in misunderstandings and errors – even for experienced cloud users – and waste an organization's cloud budget. While this is true for single-user accounts as well, accidental charges are an expensive but effective lesson. With many delegated accounts, this expensive lesson may be less effective, and may be experienced by many users. User inattention exacerbates this category of risks.

[2]For example, AWS has Identity Access Management, IAM.

To address these risks (to varying degrees), there is a spectrum of options, ranging from a focus on enforcing a policy by constraining and limiting the delegates, to providing tools and assistance to delegates to help them be more conscious of potential waste. The placement on the spectrum is influenced by many factors, including the objectives, risk tolerance, and budget of the delegator. To return to the teaching/research example, in a course, where all user resources can be estimated *a priori*, the professor might prefer to use a more proscribed approach, limiting resources by type, size, and duration (e.g. a "small" VMI for at most 10 hours). In a research group, good intentions and more expertise may be assumed, and precise predictions of resource requirements are difficult or impossible; thus, the researcher may prefer an approach that is more flexible, permitting students to use the cloud as they wish, but offering a type of *assisted garbage collection* to identify potential waste.

3 Proposed Solution

We propose a broker middleware conceptually based on the facade [10] design pattern that acts as an intermediary between the delegate and the acquisition of cloud resources. This broker (which can be implemented as a web application built on any modern web framework) can impose limits on the acquisition of resources, and provide feedback to users to help limit resource waste. The user interface adopts units of time as more intuitive measures of usage and limits to protect delegates from the complexities of public cloud pricing.

Fig. 1 provides an overview of the solution. The delegator provides the broker with information about their desired constraints and objectives, and the permissible delegates. The Delegation Risk Identification, Evaluation, and Mitigation (DRIEM) component uses this information to make decisions on behalf of the delegator. Authenticated delegates acquire resources through the cloud facade (a RESTful API or GUI consistent with the cloud provider). The broker mediates the acquisition of resources; for example, a basic access-control limit can be used to specify caps on the total number of VMIs per user, or to limit the VMIs used to those under a certain hourly price, or to limit the number of VMIs launched per hour. If approved, resources are acquired (potentially using cloud abstraction layers, e.g. [11]). Once acquired,

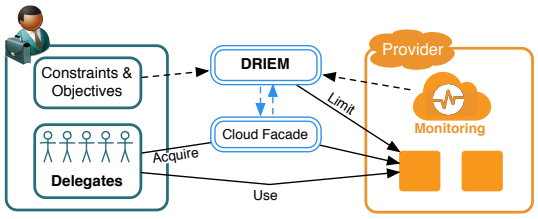

Figure 1: Broker middleware mediates the acquisition of cloud resources and uses real-time monitoring to enforce limits. Solid arrows: control; dashed arrows: information flow.

resources may be used directly by the delegates.

The DRIEM component also provides the ability to manage and enforce run-time resource limits. We envision two (possibly parallel) modes of operation: a quota system, and a system we call *graduated resource limitation* (GRL).

The **quota system**, much like other quota systems for shared resources, can ensure that a delegator's budget is never exceeded. The delegator specifies hard limits on delegate spending (either universal or per-user); DRIEM conservatively translates these limits into hourly limits for communication and enforcement. Delegates are warned when their usage reaches various percentages of the defined limit, and access is curtailed and resources released after the full quota is reached. For example, a user launching an `m1.large` instance may be told they may use that resource for 20 hours.

One enhancement to the quota system is a "delayed resource release", where a user is told a resource is unavailable, but it remains available for recovery for a limited time. Many resources are charged per hour, and it costs nothing extra to not terminate the resource until the end of the hour. The low price of storage means that files could be retained until the delegator approves their deletion.

The **graduated resource limitation system** assigns two key values to each cloud resource: a *resource lifetime* and a *decay rate*. The remaining lifetime declines based on the decay rate; the decay rate changes based on the state of the resource. A "normal" decay rate would be one minute of real time consumes one minute of lifetime, but can be modified to incentivize (or disincentive) behaviours. For example, a resource left idle might "decay" more quickly; a resource being used within a typical usage profile, more slowly. A resource temporarily suspended might not decay at all. Once the resource lifetime expires, the resource is "garbage collected"; however, with the permission of the delegator, a delegate can instead reset the resource lifetime.

This approach moves beyond merely enforcing limits, and instead provides feedback and incentives to delegates to use resources effectively. This is more efficient from both a budgetary and an environmental / "green" perspective. This general idea is familiar to users: consider a laptop or a smartphone. The estimated run time is based on the capacity of the battery and the current consumption of the battery; the state of the device determines the decay rate. A device playing streaming HD video has a high decay rate; a device in sleep mode has a very low decay rate; a device being charged actually has an increasing rate.

The management of the decay rate is crucial to this idea, and to the behavior of the broker. To choose a complex example, consider VMIs. In practice, the cost of a VMI includes the VMI itself, but also network use, storage, IO operations, and other optional charges. To ease the cognitive burden on users, we create an aggregate cost based on a statistical understanding of typical network bandwidth usage (both in and out), volume storage and object storage, and consider VMIs on that basis. Periodically – the GRL system operates with a configurable *granularity* period – the system examines each virtual machine and its utilization, and assigns a state based on its behavior. VMIs within this statistical profile are in the `in-profile` state; otherwise, they may be `under-profile` (might be idle) or `over-profile` (might be compromised, malicious, or a mistake). `paused` is a special state for stopped VMIs. Each of these states has an associated decay rate, which can be faster or slower depending on the behavior the delegator wishes to incentivize; choosing a decay rate may be policy-driven [12]. For example, by setting the decay rate to nil for `paused` and `in-profile` VMIs, and setting it high for `under-profile` and `over-profile` VMIs, the delegator can ensure resources being used normally will never be released, but that idle resources and potentially mis-used resources will be released quickly.

The GRL system also permits higher-order abstractions than VMIs. If we consider our teaching/research example, some assignments/experiments (generally, *tasks*) require that a set of VMIs be utilized simultaneously. To increase usability, the

GRL system allows lifetimes and decay rates to be applied to sets of VMI (and sets of sets, etc.). At the end of each granularity period, the system determines the shortest remaining duration for each VMI in the set, and that becomes the duration for the task being considered. This greedy approach could be substituted for a more complex algorithm.

Finally, GRL permits changes of ownership within the delegated environment. Returning to our teaching/research example, if Jennifer creates an image and graduates two years later, she (or the researcher) is responsible for transferring ownership of this image to Paul. In this two-phase process, Jennifer must give ownership of the resource to Paul and Paul must accept ownership of the image. Paul is then responsible for managing that resource within its lifetime and within his quota.

4 Conclusion

This paper has three main contributions: we have i) introduced and articulated the cloud account delegation problem; ii) described three main categories of risk associated with this problem; and iii) proposed a cloud broker middleware that will mitigate these risks and allow for clouds to be used by delegates subject to a delegator's constraints and objectives. A quota system avoids budget overruns and a graduated resource limitation system ensures resources are used more efficiently while educating students, providing them with tools and information to help them develop and improve their cloud usage practices. The context of teaching and research was used as a motivating example.

The problem definition and solution outline suggest many research challenges: expressing usage policies; capturing constraints and objectives; usability testing; translating budgets and objectives to policies; using Big Data analytics to extract "good" usage profiles from past data; estimating expected savings to justify investment in this solution; proof-of-concept implementation; applying the general idea in practice, with non-VMI cloud resources across multiple providers; and the general challenge of better managing heterogeneous, uncoupled collections of cloud resources with the same ownership but different uses.

Acknowledgements

This work was supported in part by an AWS in Education Grant award, and by IBM Centres for Advanced Studies (CAS) and the Ontario Research Fund under the Connected Vehicles and Smart Transportation (CVST) project.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] R. Buyya, R. Ranjan, and R. N. Calheiros, "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services," in *ICA3PP (1)*, 2010, pp. 13–31.

[3] M. Shtern, B. Simmons, M. Smit, and M. Litoiu, "Navigating the clouds with a MAP," in *13th IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2013, pp. 464–470.

[4] A. Smith, *An inquiry into the nature and causes of the Wealth of Nations.* T. Nelson and Sons, 1887.

[5] P. Pawluk, B. Simmons, M. Smit, M. Litoiu, and S. Mankovski, "Introducing STRATOS: A cloud broker service," in *2012 IEEE 5th International Conference on Cloud Computing (CLOUD)*, 2012, pp. 891–898.

[6] N. Grozev and R. Buyya, "Inter-cloud architectures and application brokering: taxonomy and survey," *Software: Practice and Experience*, 2012.

[7] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *Journal of Network and Systems Management*, pp. 1–53, 2014.

[8] C. Tang, C. Perng, and S. A. Baset, "Self-service financial control and organizational governance in cloud," in *Proceedings of the 8th International Conference on Network and Service Management*, 2013, pp. 229–232.

[9] M. Smit, B. Simmons, and M. Litoiu, "Distributed, application-level monitoring of heterogeneous clouds using stream processing," *Future Generation Computer Systems*, vol. 29, no. 8, pp. 2103–2114, 2013.

[10] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.

[11] M. Smit, M. Shtern, B. Simmons, and M. Litoiu, "Supporting application development with structured queries in the cloud," in *New Ideas and Emerging Results (NIER), Proc. of the Intl. Conference on Software Engineering (ICSE)*, 2013, pp. 1213–1216.

[12] M. Sloman, "Policy driven management for distributed systems," *Journal of network and Systems Management*, vol. 2, no. 4, pp. 333–360, 1994.