Synergy2Cloud: Introducing Cross-Sharing of Application Experiences Into the Cloud Management Cycle

Florin Dinu   T. S. Eugene Ng
Rice University; Houston, TX

Abstract

Automatically managing collocated cloud applications for improved performance is a hard problem due to the unprecedented scale and the dynamics of the multiplexed cloud environment. Compounding the problem, today's approaches to cloud application management are too limited in the way they acquire information. Monitoring performed by the operator is too low level and application agnostic while monitoring performed by applications in isolation is too restricted. In this paper [1] we propose sharing of application experiences as a cloud and application management building block. To achieve this level of sharing, the current state-of-the-art towards increased isolation among cloud applications needs to be re-thought. Focusing on isolation overlooks and potentially impedes the substantial benefits obtainable through sharing application experiences. We explore the benefits, challenges and incentives associated with sharing application experiences and argue why sharing is a winning proposition for both operators and applications. We also propose a web portal *synergy2cloud.com* that we hope will serve as a stepping stone for making cross-sharing cloud application experiences a reality.

1 Introduction

The case for sharing experiences in the cloud
Multiplexed, public cloud environments are popular today for economical reasons. They allow cloud operators to maximize infrastructure utilization by collocating applications belonging to multiple tenants. The prevalence of virtualization technologies further encourages collocation by facilitating the use of state-of-the-art hardware that offers unprecedented parallelism.

However, managing the multiplexed cloud environment and the applications in order to achieve the best application performance is challenging. The task is already made difficult by the highly dynamic cloud environment, its unprecedented scale and the large number of hosted applications. Unfortunately, today's approaches to cloud application management are also limited in the way they acquire information. The operator either performs monitoring in isolation [12] or shares only some basic infrastructure performance data with the applications [1]. The expressiveness of the operator's data is limited by the overhead of monitoring and it is typically low-level, application agnostic data. Applications are also limited in their self-management capabilities by the operator's tenant isolation mechanisms which only allow a highly abstracted view of the environment.

Our position is that multiplexed cloud environments give applications a unique opportunity to cross-share their experiences for potential improvements in performance, security and scalability. *Experiences* is a catch-all term that refers to any information that an application can collect as part of its execution and can be leveraged by other applications. Instead of preventing experience sharing as is the case today, the cloud operator should arbitrate, facilitate and even participate in the exchanges. We argue that this is a win-win situation for both parties: the performance of both the applications in particular and that of the cloud in general is improved. To briefly exemplify the benefits of cross-sharing application experiences, consider compute node failures which are common occurrences in the cloud [3]. Typically, failure detection is performed separately by each cloud application. The unfortunate consequence is that an application can spend an unnecessarily long amount of time detecting node failures that were already detected by other applications. This translates into unnecessarily inflated job running times and consequently extra-cost incurred by the user. Sharing such failure information allows applications to learn from other application's recent experi-

[1] This research was sponsored by NSF CAREER Award CNS-0448546, NeTS FIND CNS-0721990, NeTS CNS- 1018807, by an Alfred P. Sloan Research Fellowship, an IBM Faculty Award, and by Microsoft Corp. Views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of NSF, the Alfred P. Sloan Foundation, IBM Corp., Microsoft Corp., or the U.S. government.

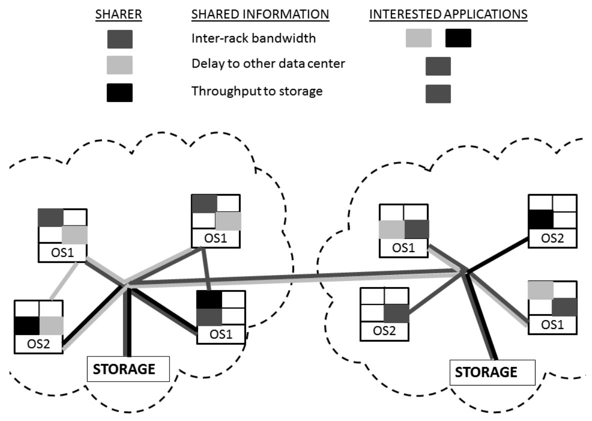

Figure 1: **Three similar applications are collocated. They apportion inter-rack network, inter-data center links, compute nodes and access to dedicated storage. Thus, sharing their experiences benefits the other applications.**

ences and respond to the failure faster. As a result, the overall efficiency of the cloud is also improved.

A primary concern when sharing application experiences is whether information shared by one application is applicable to another. The danger is that even small differences among applications can significantly influence perceived performance [20]. Fortunately, the cloud presents us with a unique opportunity. Many cloud applications are built on top of a few general purpose large scale computing frameworks (e.g. MapReduce) which provide the basic functionality of interacting with the environment and managing running jobs. Additionally, the cloud computing environment (i.e. OS, hardware) is completely known to the operator and is usually limited in diversity (e.g. small number of OS instances). Consequently, our position is that many cloud applications have substantial functionality in common thus making sharing experiences in the cloud a promising direction. Figure 1 depicts a sample cloud environment where sharing experiences can be beneficial for applications.

Importantly, operator monitoring alone is not as powerful as sharing application experiences. First, for the shared information to be meaningful, the design, configuration and implementation of the measuring component needs to be similar, if not identical. While this may be the case for two MapReduce applications the operator cannot provide different implementations for its measuring infrastructure to suit every application. Second, it is hard for an operator to provide framework specific information (e.g. throughput from dedicated storage to multiple concurrent readers). Third, any large operator monitoring effort uses valuable resources and potentially degrades user perceived performance.

Hurdles to overcome for introducing sharing
Sharing application experiences exists today in only a rudimentary form, totally insufficient for dynamically informing applications at runtime. Today, a cloud tenant configures his application through a series of trial and error runs and then measures the performance of the application and/or that of the infrastructure. Subsequently, the tenant shares the accumulated knowledge on the web [4, 19, 2] or writes a research article [14, 18]. The interested reader then learns from this past experience and applies it to his own cloud applications. This approach, however, is vastly insufficient for today's highly dynamic cloud environment where performance data can quickly become outdated and applications can scale to many different sizes. We argue that cross-sharing of application experiences should also be automatic and should occur at the timescale of the application's execution and not at that of a human being's curiosity to learn or willingness to impart experience.

To enable cross-sharing of application experiences the current trend towards providing increased isolation for collocated cloud applications needs to be re-thought. This trend hurts applications by severely restricting the possibility for legitimate information sharing. Applications either cannot communicate at all or the information they can exchange abstracts too many environmental details to make sense in the context of another application.

The drive towards isolation is a strong one. Isolation is important because it can improve application performance and security. For example, isolation allows predictable network performance. Performance predictability stimulates cloud adoption and enables cloud operators to offer SLAs. Concerning security, malicious applications could be found exploiting side-channels to detect collocation and obtain access to a collocated application's private data [16]; applications can negatively influence each other by unilaterally using multiplexed resources such as processor caches [15]. Consequently, the current trend is to develop ironclad isolation technologies at every level from network [17, 10] to compute-node resources such as disk, CPU and memory [15]. From an isolation centric point of view, the ideal scenario is that each application is provided with the illusion of a private data center [10]. Compared to this pro-isolation trend, the problem of cross-sharing application experiences in the cloud has remained largely unexplored by the research community.

This paper articulates the benefits (§2), incentives (§3) and research challenges (§4) associated with cross-sharing application experiences. We also briefly present initial ideas towards making sharing in the cloud real. We hope this work will inform the direction of the research on isolation mechanisms in order to encourage legitimate information sharing while still protecting applications from each other. We also propose a web portal *synergy2cloud.com* that we hope will provide the community initial information useful for making cross-sharing cloud application experiences a reality.

2 Benefits: What to Share and Why

In this section we explore the benefits obtainable from sharing application experiences. We give concrete examples of shareable information and argue for its usefulness. This paper focuses on public, multiplexed cloud environments. Their diversified tenant base as well as the client-provider relationship makes them more complex and more challenging. Nevertheless, our findings can also be applied to private clouds that allow applications to coexist. This is often the case today [11].

Any information measured and subsequently shared by cloud applications is not absolute information but rather the cloud and the application's performance seen through the lens of the application's implementation and design. Therefore, the danger is that information shared by one application may not be representative for others because of differences among applications. Take for example two identical cloud applications, the sole difference being the use of delayed ACK. This can lead to substantial performance differences even for basic performance information such as latency and bandwidth [20]. In the general case it is very difficult to establish the relevance of shared information for unrelated applications. Fortunately, recall that the cloud provides a unique opportunity: many cloud applications have significant functionality in common because they are built on top of a few large scale computing frameworks and are deployed in environments that have limited diversity (e.g. small number of OS instances).

Nevertheless, there exists a trade-off: the more complex or application specific the information is, the harder it is to assess the similarity of the sharing applications. Moreover, the amount of additional information necessary to verify similarity increases dramatically. For example, if configuration information is shared, details about job parameters and workload are needed to assess the relevance of the configuration for another application. With the following examples we argue that the cloud allows reasonably complex information to be shared.

2.1 Examples of Shareable Information

Performance Information
Understanding the performance of the environment is a first step towards ensuring good application performance. Following are a few examples:

- Latencies in accessing local or dedicated storage
- Data transfer throughput inside a data center or between operator's data centers.
- Local compute node usage counters (e.g. CPU load, memory use)

Sharing performance information can benefit scheduling. If there shared information of poor throughput from the dedicated storage, a scheduler may choose to postpone the execution of network-intensive operations for a suitable later time when shared information describes good throughput. Importantly, applications can obtain this benefit without paying the monetary and performance costs of detecting poor performance themselves. As another example consider a heterogeneous environment. Sharing performance information can guide the decisions on which resources to use without actually trying to run the application on all types of resources.

Scalability Information
Scaling decisions can also benefit from sharing. One example is determining how fast the application can scale out. This can be achieved by sharing the time to spawn a number of VMs of a certain type. Consider the example of an application that suddenly needs to scale out because of an exceptional situation such as a flash crowd. There is no time for it to perform trial and error work. Moreover, the application's own past may be momentarily irrelevant since the sudden scale out may require exceptional resources to which the application is not accustomed under typical use. For example, if the VM type typically used by the application is in short supply at the moment of the flash crowd, any type of VM may be acceptable as long as it can offer some processing power. Alternatively, the application may choose to trade-off some processing power for scaling speed, thus choosing a VM type that can be allocated faster when requested in large numbers.

Failure Information
Failure detection, oftentimes a lengthy and involved process in the cloud, can significantly benefit from sharing application experiences. In experiments involving injecting single TaskTracker failures in Hadoop, studies show tasks often stall for 480s even though the failure was already detected and reacted upon by other tasks [7]. Failure information can be shared live as failures are detected. This can greatly cut down failure detection time for applications that cannot quickly detect the failures themselves because they have few vantage points.

Straggler Information
Oftentimes compute nodes in the cloud do not completely fail but rather continue to run at dreadful performance levels. This can lead to straggler tasks which can significantly delay job completion [21, 5]. Sharing straggler information is useful for scheduling and speculative

execution decisions. An application would ideally avoid under-performing nodes for spawning new tasks. More importantly, existing stragglers should not be re-executed on under-performing nodes.

Security Information
Sharing application experiences also helps application security. Applications could share emerging signs of attacks against compute nodes. This may help the other applications in avoiding the compute nodes that are vulnerable or more likely to be attacked. Moreover, the operator can use this information to attempt to localize the source of the attack.

3 Incentives

Our position is that sharing application experiences is a win-win situation for both the applications and the operator. Following we describe incentives for each party to adopt sharing.

Incentives for the Operator
First, if applications perform better this is an economic advantage for the operator. This attracts more users to switch to the operator's cloud thus bringing additional revenue for the operator. Part of this increased revenue will likely translate into infrastructure upgrades thus further increasing the benefit to the applications.

Second, the operator can view the applications as a huge monitoring platform and can use the information obtained from them for managing or improving the infrastructure. This is a monitoring platform of far greater size, complexity and information throughput than the operator can ever assemble alone because of the danger of interfering with running applications and because of the large demands on the infrastructure. Moreover, because the cloud can host many applications concurrently, shared data is expected to be always fresh and have strong statistical significance.

Third, the operator can even monetize sharing by providing access to it for a small fee, which the application owner would be happy to pay if there is a good chance of performance improvement.

Fourth, the operator cannot stop application level sharing because many applications need to be able to legitimately talk to a third party. Allowing the applications to come up with inefficient and cumbersome methods of sharing can prove to be detrimental to the application performance and transitively to the cloud's performance. Instead, for the operator it is better to actively provide a sharing platform so it can retain a degree of influence over what is being shared and in what form. For example, the operator can hide potentially sensitive information (e.g. topological details) while still providing applications with the relative location information useful for sharing. The operator can also anonymize the shared information.

Incentives for Applications
A first question is why would applications participate in sharing information. If sharing leads to performance improvements, this is an important incentive in itself. In addition, the burden that sharing puts on an application is minimal. There is much shareable information that can be obtained as part of an application's typical run.

Furthermore, while one application can do some amount of measurement and trial and error work on its own it is impractical for it to reach the amount of information that can be offered by the rest of the collocated applications. This is important especially when decisions need to be made quickly, for example when an application needs to quickly scale to accommodate a flash crowd event. It has no time to perform measurements, but may obtain useful information from the other applications.

Another incentive is that sharing enables applications to receive information about resources not yet in their possession. This, for example, can be used to intelligently guide the scaling process. Applications can also receive shared information about events they have not yet encountered. For example, one application can share the failures it encountered and this may help other applications be proactive about failure detection and recovery.

Even more, one application can benefit from the information shared by a second application which is simply better suited to discover certain events because of properties like greater scale. For example, one application may discover network hotspots faster because its scale requires it to deploy tasks in multiple racks.

A second question is what are the incentives for a participating application to behave properly and not share false information or try to game the system. Assuming the operator can anonymize the information, a misbehaving application cannot direct its maliciousness at any specific target. Additionally, the misbehaving application cannot be sure its information will be used at all. Even more, as we shall discuss in (§4), the operator should be able to verify part of the shared information.

4 Challenges

We next outline a number of challenges and point out several opportunities for future work. We view operator involvement as central in addressing these challenges.

Increasing Information Expressiveness
In order to share more expressive information, a first challenge to overcome is the limiting effect of increased isolation. The result of increased isolation is that the

shared information may sometimes only be meaningful to a specific application. This is the case with quotas. If an application has reached its bandwidth quota it does not necessarily mean another one will have the same quotas. As an intermediary, the operator can identify when quotas are reached and can differentiate this from application level bottlenecks.

A second challenge is the limited knowledge applications have about the infrastructure (e.g. topology). Today, an application usually observes a highly abstracted view of the physical infrastructure which may be only relevant to itself (e.g. separate IP address spaces). Consider an application A sharing the information that the path between location D1 and D2 has been overloaded for minutes. Unknowing to A which uses a specific path between D1 and D2, there is multipath connectivity between D1 and D2. From A's experience another application may incorrectly believe that all paths between D1 and D2 are overloaded. The future research challenge is providing means for the operator to offer an abstracted view of the environment that encodes relative positioning and connectivity information while still safeguarding sensitive details.

A third challenge involves granularity. It should be possible to break down or combine shared information in order to satisfy the needs of applications that are interested in different granularity levels. The operator can provide such conversions.

Ensuring Information Authenticity
It is challenging to identify fake information. Seemingly fake shared information may be the result of incorrect parameter settings, bad application design (e.g. prone to incast collapse) or contention. Ideally, the system would not flag this information as malicious. In fact, such information could be useful to applications having similar design flaws to the sharer.

To mitigate the chance of fake shared information, future work could enforce verifiable experience exchanges. While the operator cannot verify all shared information it is in the position to verify ownership claims. An application should not be allowed to advertise information about resources it does not even hold. If verification cannot be driectly performed as it is the case for performance data, statistical approaches could potentially be used to cull obvious outliers or consensus protocols could be used by multiple applications to detect and reject malicious applications. Applications should also have the option to filter the set of sources from which experiences are received. In this manner, trust groups can be formed by applications that belong to mutually-trusting users.

Assessing Similarity Among Applications
Ideally sharing should occur between very similar applications. However, assessing similarity is challenging. Different applications versions or modified application code could still share substantial functionality. Conversely, the same application could behave differently on different hardware. Future work should consider quantifying similarity on a per-component basis. This would be similar to obtaining hashes of the specific code objects responsible for a particular measurement. On top of this the operator can add an identification mechanism for the underlying hardware and software platform.

Design
Additional challenges are related to system design. Future work may consider providing sharing using middleware libraries, thus minimizing code changes. This library could at least ensure that applications transmit their experiences unmodified by leveraging the fact that many cloud applications routinely log detailed performance data. In order to design a scalable and efficient lookup mechanism for the shared data, future work on this topic may benefit from the rich body of past research work devoted to publish-subscribe systems [8].

5 System Functionality Description

In this section we present a short description of the building blocks and the flow of shared information in our envisioned sharing system.

- Applications publish information and tag it with an identifier that describes the application components involved in measuring the information.
- An operator-managed publish-subscribe system connects sharing and receiving applications.
- The operator anonymizes the data and converts it to an abstracted view of the environment that preserves relative positioning.
- The operator adds an identifier describing the hardware and OS.
- If possible, the operator verifies the information.
- The operator performs conversion of granularity if requested by any receiving application.
- Subscribing applications receive the information and decide on its use. The operator uses the information for his own management needs.

6 Discussion

Isolation and Sharing
Throughout this paper we argued that isolation should not unnecessarily limit sharing. We see both isolation

and sharing as part of the foundation of an application's performance and scalability. As such, the way forward is for the two to complement and not exclude each other. Isolation ensures that application performance is maintained at reasonable levels, while sharing brings an extra-boost by providing supplementary information. Similarly, isolation ensures a level of predictability for scaling an application while sharing provides the extra-boost with information about where to scale. Therefore, it makes sense to combine the advantages of sharing and isolation, to obtain the best of both worlds.

Active Collaboration
The type of sharing described so far can be deemed as passive collaboration because one application receives information published by another. One can also envision active collaboration in the cloud: an application asks for the involvement of another. For example, an application's failure detection algorithm may require that lack of connectivity be reported by three vantage points, but it currently has only two such vantage points running. It may prove considerably faster to ask another application to attempt a connection on its behalf. Future work may consider defining a set of substitutable operations that application A1 can do on behalf of A2. An example of such substitutable operations are TCP connection attempts.

Related Work
Sharing of information can already be encountered in different forms. At end-hosts, flows can learn from each other and share information about the state of congestion along common network paths [6]. At the level of large enterprise networks, route redistribution allows routers to exchange routes between different routing domains [13]. Web caches can cooperate and serve each other's misses, thus reducing overall bandwidth usage [9].

The uniqueness of sharing experiences in the cloud stems from the richness of the shareable information. Performance, scalability and failure information can all be shared and in different granularities. Also, the cloud environment is more dynamic and this makes it challenging to always use fresh information. Lastly, the impact of sharing in the cloud is significant. Both the operator and cloud applications win from sharing.

7 Conclusion

We hope that our discussion on sharing application experiences reveals the richness of this research direction and the promising outlook for improving the performance, scalability and security of cloud applications. With this paper we also hope to call attention to the downside of the direction towards strong application isolation which data center design is currently embracing. While we acknowledge the importance of isolation mechanisms we argue that isolation and the sharing of experiences need not restrict each other but can actually complement each other for the benefit of both cloud operators and applications.

References

[1] Amazon CloudWatch. http://aws.amazon.com/cloudwatch/.

[2] Cloud Harmony. http://cloudharmony.com/.

[3] Failure Rates in Google Data Centers. http://www.datacenterknowledge.com/archives/2008/05/30/failure-rates-in-google-data-centers/.

[4] Rackspace Cloud Servers versus Amazon EC2: Performance Analysis. http://www.thebitsource.com/featured-posts/rackspace-cloud-servers-versus-amazon-ec2-performance-analysis/.

[5] G. Ananthanarayanan, S. Kandula, A. Greenberg, I. Stoica, Y. Lu, B. Saha, and E. Harris. Reining in the outliers in map-reduce clusters using mantri. In *OSDI*, 2010.

[6] H. Balakrishnan, H. Rahul, and S. Seshan. An integrated congestion management architecture for internet hosts. In *SIGCOMM*, 1999.

[7] F. Dinu and T. S. E. Ng. Hadoop's overload tolerant design exacerbates failure detection and recovery. In *NETDB*, 2011.

[8] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35, June 2003.

[9] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary cache: A scalable wide-area web cache sharing protocol. In *IEEE/ACM Transactions on Networking*, 1998.

[10] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang. SecondNet: a data center network virtualization architecture with bandwidth guarantees. In *CONEXT*, 2010.

[11] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, and I. Stoica. Mesos: A platform for fine-grained resource sharing in the data center. In *NSDI*, 2011.

[12] M. Isard. Autopilot: Automatic data center management. In *Operating Systems Review*, 2007.

[13] F. Le, G. Xie, D. Pei, J. Wang, and H. Zhang. Shedding light on the glue logic of the internet routing architecture. In *SIGCOMM*, 2008.

[14] A. Li, X. Wang, S. Kandula, and M. Zhang. CloudCmp: Comparing public cloud providers. In *IMC*, 2010.

[15] H. Raj, R. Nathuji, A. Singh, and P. England. Resource management for isolation enhanced cloud services. In *CCSW*, 2009.

[16] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds. In *CCS*, 2009.

[17] A. Shieh, S. Kandula, A. Greenberg, C. Kim, and B. Saha. Sharing the data center network. In *NSDI*, 2011.

[18] G. Wang and T. S. E. Ng. The impact of virtualization on network performance of amazon EC2 data center. In *INFOCOM*, 2010.

[19] J. S. Ward. A performance comparison of clouds: Amazon EC2 and ubuntu enterprise cloud. SICSA DemoFEST, 2009.

[20] M. Yu, A. Greenberg, D. Maltz, J. Rexford, L. Yuan, S. Kandula, and C. Kim. Profiling network performance for multi-tier data center applications. In *NSDI*, 2011.

[21] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, and I. Stoica. Improving MapReduce performance in heterogeneous environments. In *OSDI*, 2008.