Closer to the Cloud - A Case for Emulating Cloud Dynamics by Controlling the Environment

Ashiwan Sivakumar
School of Electrical and Computer Engineering
Purdue University
Email: asivakum@purdue.edu

Shankaranarayanan P N
School of Electrical and Computer Engineering
Purdue University
Email: spuzhava@purdue.edu

Sanjay Rao
School of Electrical and Computer Engineering
Purdue University
Email: sanjay@purdue.edu

***Abstract*—Cloud computing offers a wide range of benefits including potential cost-savings by leasing resources from cloud service providers and improved user experience by Geo-distributing applications [17]. Cloud service providers typically share the platform and cloud resources across multiple users for better utilization. They lease resources on-demand to users in three broad flavors, namely, Infrastructure as service (IaaS)[1], Platform as Service (PaaS)[9] and Software as Service (SaaS) [8]. These shared resources make the cloud environment highly dynamic and often induce substantial performance variation in the cloud services [21], [22], [14]. A critical step in developing cloud based applications is the ability to test these solutions on a controlled but cloud-like environment. Also, it is important for commercial product vendors to profile the application performance in the presence of such cloud dynamics before a production deployment. While many testbeds [13], [12], [15], [18] provide the abstraction of cloud services via virtualization, control over the cloud environment is seldom available for systems that require precise emulation of the cloud dynamics. In this work, we emphasize the importance and need for testbeds that can provide repeatable and precise control over the environment for developing such latency sensitive applications. We substantiate our claim by presenting the case study of a multi-tier enterprise application called *DayTrader* [2] deployed on a large scale experimental testbed *GENI* [7]. We show using various controlled experiments on *DayTrader* and *GENI*, the importance and utility of controlling the environment (e.g., network and link parameters) in the testbeds to emulate cloud dynamics. We show the importance of emulating cloud dynamics to enable research on adaptive systems that help applications deal with performance variation in cloud services. We present our experience with performing controlled experiments on *GENI* and briefly discuss possible enhancements in such testbeds that can open up new avenues of research in cloud computing.**

I. Introduction

Cloud computing has undoubtedly become one of the most promising technological innovations of the last decade and has received great attention from both industry and academia. On the one hand, large players in the IT industry have realized the economic potential of cloud computing and have themselves become cloud service providers, on the other hand, enterprises benefit from the elasticity, scalability, Geo-diversity and the pay-as-you-go policy of the cloud services which make them naturally suitable for the deployment of large scale applications. Cloud computing also helps reduce the operational and maintenance cost of hosting an application by sharing resources among multiple clients.

However, shared cloud infrastructure presents many unique challenges to the application developers, the most important of which is dealing with failures and performance variations in the cloud services. Recent works [14], [21], [22] have explored such dynamics in the cloud services using extensive measurement studies on commercial cloud platforms [1], [9], [8]. Therefore it is very important to test the applications on a cloud-like environment and profile their performance in the presence of various cloud dynamics. Also, dealing with performance fluctuations is still largely an unsolved research problem. A critical requirement for developing solutions to such problems is the ability to evaluate the effectiveness of the solution using controlled experiments in testbeds that provide a cloud-like abstraction.

Cloud testbeds like Open Cirrus and Eucalyptus [13], [15] provide an excellent platform for evaluation of research prototypes through techniques like virtualization that can provide applications with a cloud-like interface. They allow emulation of various scenarios that are both undesirable and hard to reproduce on commercial platforms. These testbeds are also federated across multiple data-centers similar to commercial cloud platforms, where experimenters can lease resources. While providing an infrastructure and interface similar to cloud platforms is an essential first step that enables research in cloud systems, many of these testbeds do not expose control of the environment in which the cloud services are hosted.

In this work, we bring out the importance of control over the environment of the testbed for enabling new avenues of research in cloud systems. We make use of insights gained from our measurement study on commercial cloud platforms to show the presence of performance variations in cloud services due to shared infrastructure. We present the case study of a multi-tier enterprise application, *DayTrader* deployed on a federated, large-scale *GENI* [7] testbed. Specifically, we show that it is critical to obtain control of the testbed environment for profiling the application performance using controlled and precise experiments. We present a few experiments that profile the application performance by varying the environment parameters such as network latency and intra-data center bandwidth. We also highlight the importance of a controllable environment in the research and development of new cloud

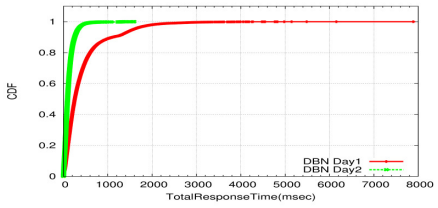
systems. As an example, we show how *GENI* can help evaluate the responsiveness of *Dealer* [19], a system that we are building to help applications adapt to short-term fluctuations in performance of cloud services [14], [21], [22]. We show using our experiments the utility of such testbeds in creating a cloud environment where it is possible to accurately replay the cloud dynamics using traces obtained from commercial cloud systems.

The rest of the paper is organized as follows. In section §II, we show the presence of performance fluctuations in the cloud environment using a measurement study on commercial cloud platforms. We then consider single and multi data-center deployments of *DayTrader* on *GENI* as case study to discuss the importance and need for emulating cloud dynamics on testbeds. Our results show that it is possible to accurately control environmental parameters like network latency which helps profile and tune the application performance. Section §V briefly talks about the importance of our approach in developing research systems. Finally, we discuss our experience with *GENI* as a testbed for conducting controlled experiments. We also discuss the enhancements that need to be effected on such testbeds which can potentially open up new avenues of research on cloud platforms.

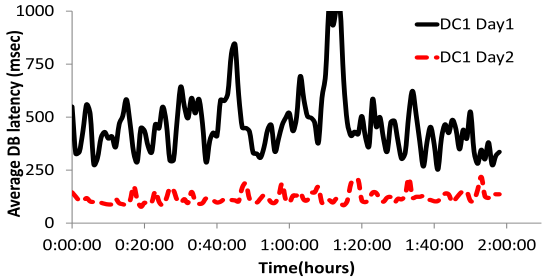
II. Motivation

A. Performance fluctuations in cloud services

Cloud services are susceptible to performance problems that can last anywhere from few seconds to few days. These problems can occur due to a number of reasons like hardware failures, network misconfiguration, increased user load or due to performance degradation of the shared cloud infrastructure. Moreover, these performance issues can affect individual VMs or even an entire data-center. These short-term and long-term variabilities in the performance of the cloud environment have severe impact on the application performance. As discussed in section §I, many researchers have studied performance problems in the cloud [14], [21], [22]. We have also observed short-term and long-term anomalies in the application performance in our experiments on real clouds. For instance, fig.1(a) outlines the performance of a commercial cloud database service deployed at a data-center in the northern part of US. The graph shows the CDF of the average response time of the database along the Y-Axis and the time in ms along the X-Axis. The data represented in the graph is collected over a duration of about four hours across two consecutive days. From the figure, we observe that the performance of the component varies drastically across the two days. Fig.1(b) shows a snapshot of the time series graph of the database latency across the two days corresponding to Fig.1(a). It can be observed from the graph that the DB showed a considerably poor performance on Day1 due to a load issue with the entire data-center while the same DB on Day 2 showed a better and stable performance. Also, the curve for Day1 shows that there is significant fluctuation in the performance of the cloud DB service.



(a) CDF of the database latency in the US North data-center of Windows Azure over a duration of four hours for two days.



(b) Time-series showing the database latency in the US North data-center of Windows Azure over a 2 hour period across two days.

Fig. 1. Database component performance issue in *Microsoft Azure*

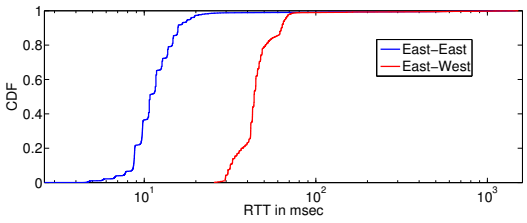


Fig. 2. CDF showing intra and inter-data center latency in *Amazon AWS*

Figure 2 shows the CDF of the inter and intra data-center round trip times measured over a period of 6 hours for a TCP stream between two micro instances of *Amazon AWS*. The graph shows that there is substantial performance variation in the measured RTT even in such simple experiments. These observations from our measurement study emphasizes the fact that cloud environment is highly dynamic. This also implies that it is important for the applications deployed on the cloud to be monitored continuously for ensuring stable performance.

B. Need for emulating cloud dynamics

Commercial and enterprise applications have stringent latency requirements and measuring the impact of such cloud dynamics on the application is a challenge. Application developers typically pilot test their application and profile its performance under realistic conditions before committing to the cloud. This often requires a federated deployment of these multi-tier applications and load testing at scale. While evaluating the prototype in commercial cloud platforms with

experiments conducted in the wild are insightful, it is very hard to reproduce failure and test scenarios accurately and repeatably on real cloud platforms due to two reasons. First, cloud providers often do not provide the required degree of control to emulate a certain scenario. Second, the shared cloud platforms have inherent performance fluctuations as we saw in the previous section. Therefore, it becomes hard for the developers to distinguish and isolate infrastructure issues from that of the application design. For instance, a short spike in the application response time could be due to increase in network latency, user load, queuing at the application server or due to the shared cloud platform. Profiling and fine tuning the application performance requires isolation of resources, accuracy and repeatability of a scenario. Testbeds which provide resource isolation and a controlled environment are therefore critical to evaluate and profile applications under emulated cloud dynamics. Often, it is also desirable to accurately replay performance traces obtained from real cloud services on the testbeds to profile the performance of the various design or deployment choices of an application. For instance, the application can be deployed in a single or multi data-center environment on a testbed and its performance with the two deployment strategies can be analyzed. Also, one could emulate the various dynamics observed in real clouds like variations in network delay, packet loss or bandwidth to study the application behavior in such scenarios.

C. Cloud testbeds for emulation

There is a wide-spread need to control environment parameters like delay, bandwidth, fault domains etc which is not possible in a shared and distributed infrastructure. We need testbeds that not only give a view of the layers above the infrastructure like PaaS and SaaS, but also those that are hidden away from the applications to emulate the cloud dynamics. Currently cloud research is limited to organizations which have access to private clusters that provide control over the resources. Researchers are developing shared testbeds like Open Cirrus, ProtoGENI, Planetlab, Emulab [13], [7], [10], [6] which can be opened up to a larger community for research purposes. One such initiative is *GENI*, that serves as an attractive, easy-to-use testbed to emulate real cloud dynamics, study application response time variations and evaluate systems which adapt to such cloud dynamics.

III. Impact of Cloud Dynamics on Application Performance

In this section we introduce a multi-tier enterprise application *DayTrader* which we deploy on the *GENI* testbed. We begin by describing the architecture of *DayTrader* (a latency sensitive enterprise application) that is used to study the impact of cloud dynamics on application performance. We present the deployment of *DayTrader* on *GENI* testbed and discuss how *GENI* helps in emulating cloud dynamics. We discuss the aspects of *GENI* that help gain control over the environment to emulate cloud dynamics.

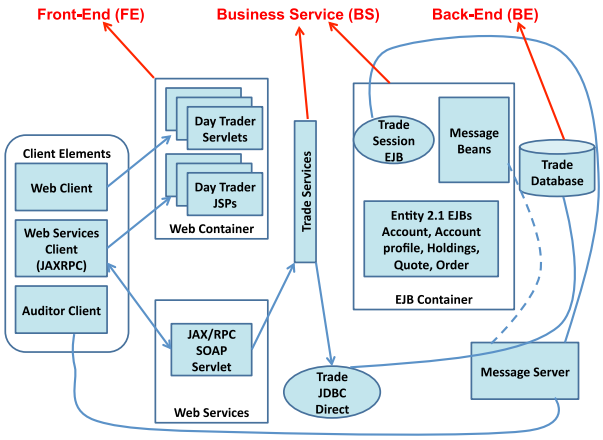


Fig. 3. *DayTrader* Application Architecture

A. Day Trader application

Figure 3 shows the architecture of *DayTrader*. It is a benchmark application developed to evaluate the performance of J2EE framework and inter-operability across multiple components in a service-oriented architecture(SOA)[2]. Like any other enterprise application, *DayTrader* follows the well-established MVC architecture with a front-end component (FE) that serves transactions from the users and presents the response as a web page, a business service component (BS) which is invoked by the front-end to perform internal trading operations like buying and selling stocks, a back-end database component (BE) which holds the user data including account, quotes and holdings. The figure shows the different complex interactions among various components. The front-end contacts the database directly through a message server to asynchronously process buy and sell stock operations. It can also contact the database through a business service- marked as TradeServices in the figure, either directly or through SOAP servlet. SOAP is a client-server protocol for communication across components deployed as web services through XML document based messages. We can see many such component interaction patterns in the diagram, potentially leading to many paths a user request can take.

Each component is deployed as instances running on application servers (Geronimo). The components are decoupled from each other and can scale dynamically based on user load. The application can be deployed within a single data-center or across multiple data-centers with one or more instances for each component. This way the application can guarantee reliability and load balancing across many servers. We can install our own load balancing mechanism or use the load balancer offered by the cloud provider. User experience in such applications can be improved by deploying the application instances closer to the clients.

B. GENI testbed

1) Changing network parameters: *GENI* is a testbed that provides a controllable and repeatable environment for conducting experiments. It provides the ability to control the

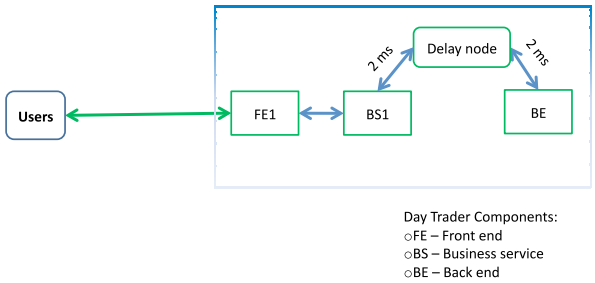


Fig. 4. Single data-center deployment testbed

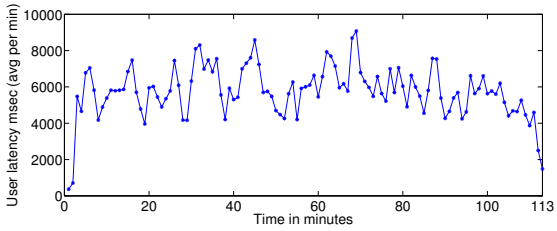


Fig. 5. A plot showing the observed application latency on replaying the trace shown in figure 1(b)

network parameters like link latency through delay nodes that we use to emulate cloud dynamics which was discussed in section §II. Though introducing delay in a link can be done in any testbed using an application queue to store and forward all packets, we require a more accurate and repeatable environment which is challenging to simulate. *GENI* provides precise control over the network parameters that can be maintained consistent for an experiment.The experimenters can also choose from various queuing options like naive TailDrop and WRED in order to simulate network congestion. The link bandwidth can be modified to induce change in the transmission delay or introduce packet losses.

2) Federated sites: Most of the existing testbeds [13], [10] provide the ability to use nodes from data-centers distributed geographically, but requires an account to be created on each of the sites. Also the resource guarantee policy vary across these sites which makes it hard to perform coordinated experiments using multiple federations. For instance, we have observed in our experiments that PlanetLab nodes are flaky and offers little guarantee of consistent performance. *GENI* integrates various federation of testbeds under a single framework so that experimenters can add nodes from different sites into their slice and access can be gained with just a single account on one of the federations. Moreover in *GENI* slices the network latency experienced by ping between nodes within a federation is negligible where as ping between nodes in different federations is predominantly Internet delay. This characteristic is consistent to what can be observed from commercial cloud platforms. Based on these observations, we believe that *GENI* would be a useful testbed for emulating Geo-distributed data-centers similar to commercial cloud platforms.

IV. Case Study: Profiling application performance

In this section, we present a few experiments which study the impact of variation in network latency on the application performance. We deployed *DayTrader* on the *GENI* testbed in two configurations, (i) All components within a single federation and (ii) Components spread across two federations. In all our experiments user workloads (HTTP requests) were generated using load testing scripts like grinder running on remote sites. We use workloads from the well-known DaCapo benchmark suite [4] to initiate user sessions that follow Poisson arrival. The workload consists of a mix of user operations like account, quote, buy and sell stock, register, update etc each resulting in a variable amount of response data depending on the user and operation.Our experiments show that *GENI* provides precise control and repeatable environment for emulating cloud dynamics. We also discuss some of the challenges involved in replaying traces obtained from a commercial cloud in a multi data-center testbed.

A. Single data-center deployment

Figure 4 shows our deployment of *DayTrader* within a single federation (Utah) of the *GENI* testbed. The front-end, business service and back-end components are deployed as web services on the *GENI* nodes. Since all components are located within the same federation, they are configured to communicate among themselves using the LAN interface. We emulate a change in the network latency with the help of a delay node [5] that is provided as part of the *GENI* infrastructure. As shown in the figure, the delay node is introduced in the link between the business service and back-end components to emulate a change in network latency between the two components.

1) Replaying traces from commercial cloud platforms: In this experiment, we emulated the database performance problem observed in a real cloud which we had described in section §II. We replayed the trace of the DB component latency corresponding to figure 1(b) and measured the impact of the latency variation on the response time of the application. The delay on the link between the business service and back-end components was changed every minute in accordance to the trace data. Figure 5 shows the end-to-end application latency corresponding to the input trace. The end-to-end application latency is the sum of the processing delay at each application component along the request path together with the communication delay between the components. The application response time more or less follows the input trace but appears scaled up by a factor. This is because each user request typically translates to multiple calls between the application components as shown in figure 3. For example, a single user request at the FE to fetch a user home page information could result in multiple calls made to the BS with each call fetching different blocks of information like user account, stock holdings, last 5 transactions, etc,. The business service in turn may run multiple database queries for each request that it receives from the FE. An interesting fallout of such

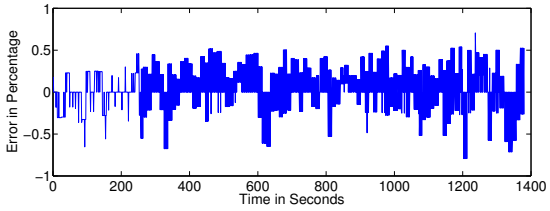


Fig. 6. A plot showing the residual error percentage between the observed link delay and the input trace.

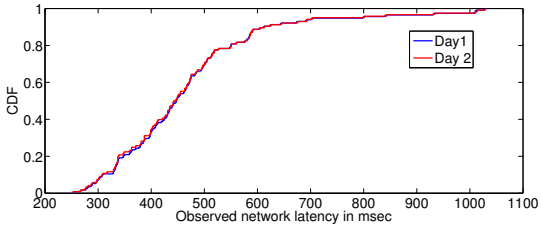


Fig. 7. CDF showing the observed network latencies across Day 1 and Day 2.

interaction is that an increase in the latency in the BS-BE link has a cumulative impact on the user perceived response time of the application.

Replaying such traces might involve coarse adjustments to the delay parameters via a delay node. But this helps in emulating real cloud dynamics and studying the behavior of various components of the application when subjected to short-term and long-term performance fluctuations. The observations made from the above experiment about the domino effect due to a performance problem perceived by the user is an example for the benefit that we get by replaying traces from real cloud. We would be able to understand the impact of such performance fluctuations on an application and improve the design so that it can adapt in the best way possible. Only controlled environment provided by open testbeds like *GENI* helps in emulating such real cloud dynamics.

2) Accuracy and repeatability of GENI *experiments:* As part of this experiment we study the accuracy of *GENI* testbeds by analyzing the error percentage between the observed BS-BE link delay obtained through ICMP and the input trace 1(b) replayed on the link. To measure the accuracy of the trace being replayed, we plot the deviation of the measured latency values from the input trace values (expected values). Figure 6 shows a snapshot of the above plot obtained for a period of 20 minutes. The Y-Axis shows the error percentage (% deviation) and the X-Axis shows the time in seconds. The graph shows that the error percentage is less than 1% at all times in our experiment. This shows that environmental parameters like network latency can be accurately emulated in *GENI* testbeds.

We also plot the measured BS-BE link latencies corresponding to the input trace along the X-axis and the CDF along the Y-axis for two days in figure 7. We can see that the measured values are almost the same on both the days with very negligible variation. A cloud experimenter would require

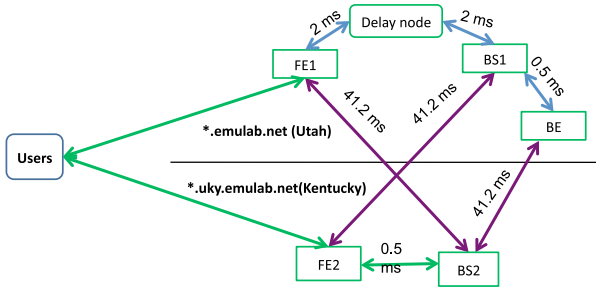


Fig. 8. Multi data-center deployment testbed

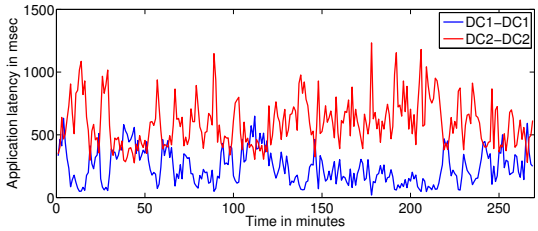


Fig. 9. Time series showing the user latency in DC1 and DC2 without any delay

such high degree of repeatability in the test environment for running controlled experiments. For this trend to be repeated across days, testbeds like *GENI* should provide consistent resource guarantees to the experimenter.

B. Multi data-center deployment

The multi data-center deployment of *DayTrader* on *GENI* is shown in figure 8. The setup has 3 nodes from Utah hosting one instance of the front-end, business service and back-end components and 2 nodes from Kentucky running an instance of the front-end and business service. The instances internal to the data-center communicate through the LAN interface while the instances running on different data-centers communicate using the stitched tunnel interface. All the components in Utah (DC1) are internal while the business service running in Kentucky (DC2) should communicate with the back-end deployed in DC1 through Internet. We also have a delay node in the link between the front-end and business service components in DC1 to emulate cloud dynamics and study the application behavior. We present the results from our experiments with different delay values set between the FE and BS in Utah.

In the first experiment we do not have any explicit delay set between the front-end and the business service components. Figure 9 is a time series showing the end-to-end application latency for requests served in DC1 and DC2. We plot the user latency in m sec along the Y-axis and the time in minutes along the X-axis. We can see that the requests served by DC2 experience more latency than those served entirely in DC1. As described in §IV-A1 each user transaction has multiple

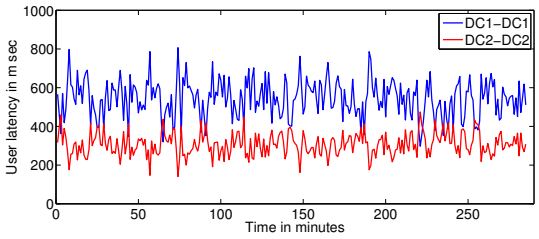


Fig. 10. Time Series showing the user latency in DC1 and DC2 with a delay of 250 ms between FE1 and BS1.

calls between each component internally and the number of inter-component calls is maximum between BS and BE. All database queries from BS2 to BE1 incur the cross data-center communication cost and hence user experience is better in DC1 than in DC2.

In another experiment we introduce a delay of about 250 ms between FE1 and BS1. We use the same user workload and traffic pattern as the previous experiment. Figure 10 shows the time series graph of the end-to-end application latency for both data-centers. As expected, we can see that the response time in DC2 is better than that in DC1 due to the increased FE1-BS1 link delay. We observe from the results that faster transactions affect the response times of slower requests. This can be seen from the DC2 application response times in both the experiments. Database has limited server connections and hence faster requests load the server thereby affecting the requests from the other data-center. This gives us the intuition that database component is the bottleneck for scalability in *DayTrader* and similar applications. What-if analyses of this kind help profile application performance under emulated cloud dynamics.

In our experiments with the multi data-center testbed we also wanted to replay traces obtained from real clouds similar to those done in a single data-center deployment. This requires modifying the link parameters dynamically during the experiment run. We found that in a multi data-center deployment *GENI* does not provide the capability to modify link delay dynamically. Hence we use ProtoGENI [11], one of the federations of *GENI* to emulate multi data-center setup with delay nodes for changing network parameters on the fly. The inter data-center links are emulated using delay nodes with a large delay value.

The ideal test scenario would be to have nodes from different federations like Utah and Kentucky in the same slice created using the ProtoGENI interface and add delay nodes to introduce a problem between system components. The delay parameter would be changed periodically based on traces from real cloud experiments while traffic flows and the application behavior would be studied. Changing the delay parameter should be oblivious to the experiment run which requires the change to take effect without reloading the slice. Nevertheless the introduction of delay nodes in *GENI* has been a great advantage towards making it a controlled environment for future cloud research.

V. Case Study: Evaluating Adaptive Systems

A. An adaptive system

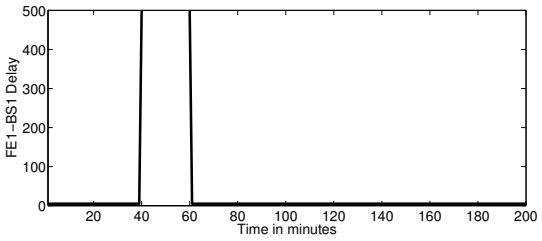
As discussed in section §II, the cloud environment is highly variable and the problem could be isolated to individual components within a data-center. Hence unlike the current redirection mechanisms in data-centers which abandon a whole deployment, it's more efficient to use other components within the same data-center which aren't faulty. This way a system can be built which can adapt to short-term variability across components in the same data-center. *Dealer* is one such system that adapts to the ever-changing cloud dynamics [19]. *Dealer* makes use of the link and component latencies measured using a monitoring system to predict path that would have the least latency. It is important to note that *Dealer* can optimally suggest paths which may involve components from different data-centers.

Dealer needs the latencies across all inter and intra data-center links and the computation latencies from every component through which the user request can travel. This introduces the necessity to run a monitoring system along with the deployed application to report the latency values to *Dealer*. This monitoring system should calculate and report the metrics not only from individual VMs or physical nodes, but also from the environment. In the initial stages of development, running these systems on real cloud hinders the efficient comparison of the algorithms given the problem with performance fluctuations. Evaluating such adaptive systems by emulating cloud dynamics is possible only in testbeds that not only provide control over the VMs but also on the environment. Since *GENI* provides the ability to control the network parameters in a repeatable manner, it can be used as a testbed to evaluate such adaptive systems.

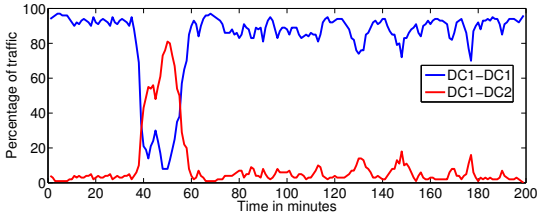
B. Evaluation results

In this section we present the results of an experiment that was conducted to evaluate the adaptive system. We use a testbed similar to that in figure 8 except that all nodes are within the same federation (Utah). Due to the challenges involved in changing link delay dynamically on a multi-data center setup that was discussed in section §IV-B, we emulate cross data-center links using delay nodes with round trip delay set to 41.2 ms. This value is obtained from measurements done on the tunnel links between Utah and Kentucky nodes.

In this experiment we increase the delay in the link between FE1 and BS1 from 5ms to 500ms, retain the value for about 20 minutes and then bring it back to 5 ms. We evaluate the adaptive system by observing the response to such standard input reference waveforms like step-up and step-down. The input waveforms are actually delay values set in the FE1-BS1 link and the system response is the path chosen for future user transactions. The results are shown in figure 11(b). The link delay was stepped up at time 40 and the value was held steady for about 20 minutes. We observe that *Dealer* redirects the transactions from FE1 to BS2 within a few seconds. Similarly the link delay was stepped down at time 60 and it



(a) A step-up and a step-down input reference waveforms given to the adaptive system.



(b) A plot showing the path taken by requests when the control inputs are injected. This is the output response of the system to standard step inputs.

Fig. 11. Evaluating an adaptive system under cloud dynamics

was observed that the request path changes from BS2 to BS1. This experiment shows that the system is agile to performance fluctuations within a data-center and it redirects the traffic to a replica of the faulty component in another data-center. Therefore we see that testbeds like *GENI* would be critical for evaluating adaptive systems.

VI. Conclusions and Future Directions

In this paper, we argue that it is critical to get control over the testbed environment for enabling new avenues of research in cloud systems. We show the presence of performance fluctuations in cloud services through a measurement study on commercial cloud platforms and argue that it is important to emulate the cloud dynamics on testbeds. We show the importance and utility of control over testbed environment for cloud developers to emulate cloud dynamics and profile their application before a production deployment. As a case study, we present a few experiments that profile a multi-tier enterprise application *DayTrader* [2] by emulating cloud dynamics using a federated, large-scale testbed *GENI*. Our experiments show that *GENI* provides an accurate and repeatable environment for running controlled experiments. We also highlight the importance of a controlled environment in the research and development of new cloud systems through the case study of *Dealer* [19], a system that helps applications adapt to short-term performance variations in cloud services.

Finally we share our experience with conducting controlled experiments on *GENI*. *GENI* provides a highly programmable interface and flexible APIs to design innovative experiments. *GENI* integrates multiple individual federations into a unified large-scale testbed which is useful in profiling applications under different deployment scenarios. Introduction of features such as delay node in *GENI* is an important first step towards providing a controllable environment. A future direction for such testbeds would be to provide dynamic control of network parameters that can be used to replay more traces obtained from cloud experiments. Also cloud application developers would use non-controllable platforms in *GENI* like Planet-lab [10] to emulate geo-distributed clients that generate traffic in their controlled experiments. Emulating cloud features like data-stores (For eg Blobs, Queues, CloudStore, BigTable [1], [9], [3], [16], [20]) would be important for enabling new avenues of research in cloud systems. We believe that by expanding its horizons *GENI* has the potential to become a testbed that enables advanced research in cloud systems.

References

[1] Amazon Web Services. http://aws.amazon.com.
[2] Apache DayTrader Benchmark Sample. http://cwiki.apache.org/GMOxDOC20/daytrader.html.
[3] CloudStore. http://kosmosfs.sourceforge.net.
[4] DaCapo DayTrader Workloads. http://www.dacapobench.org/daytrader.html.
[5] Emulab Delay nodes. http://users.emulab.net/trac/emulab/wiki/BridgeNodes.
[6] Emulab: Network Emulation Testbed. http://emulab.net.
[7] Global Environment for Network Innovation. http://www.geni.net/.
[8] Google App Engine. http://code.google.com/appengine.
[9] Microsoft Windows Azure. http://www.microsoft.com/windowsazure/.
[10] Planetlab : An Open platform for developing, deploying and accessing planetary-scale services. http://planet-lab.org.
[11] ProtoGENI: The Control Framework for GENI Cluster C. http://www.protogeni.net/trac/protogeni.
[12] T. Anderson, L. Peterson, S. Shenker, and J. Turner. Overcoming the internet impasse through virtualization. *Computer*, 38(4):34–41, 2005.
[13] A. Avetisyan, R. Campbell, I. Gupta, M. Heath, S. Ko, G. Ganger, M. Kozuch, D. O'Hallaron, M. Kunze, T. Kwan, et al. Open cirrus: A global cloud computing testbed. *Computer*, 43(4):35–43, 2010.
[14] B. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking cloud serving systems with ycsb. In *Proc. of the ACM symposium on Cloud computing (SoCC)*, 2010.
[15] D. N. et al. The eucalyptus open-source cloud-computing system. In *Proc. of Cloud Computing and Its Applications [online], Chicago, Illinois*, Oct 2008.
[16] F. et al. Bigtable: A distributed storage system for structured data. In *Seventh Symposium on Operating System Design and Implementation (OSDI)*, Nov 2006.
[17] M. A. et al. Above the Clouds: A Berkeley View of Cloud Computing. Technical Report UCB-EECS-2009-28, EECS, University of California at Berkeley, Feb 2009.
[18] R. G. et al. The Open Cloud Testbed: A Wide Area Testbed for Cloud Computing Utilizing High Performance Network Services. Technical Report arXiv:0907.4810v1, CS, Cornell University, Jul 2009.
[19] M. Hajjat, S. Narayanan, D. Maltz, S. Rao, and K. Sripanidkulchai. Dealer: Dynamic Request Splitting for Performance-Sensitive Applications in Multi-cloud Environments. Technical Report TR-ECE-11-10, Electrical and Computer Engineering, Purdue University, Apr 2011.
[20] A. Lakshman and P. Malik. Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, 44(2):35–40, 2010.
[21] A. Li, X. Yang, S. Kandula, and M. Zhang. Cloudcmp: comparing public cloud providers. In *Proceedings of the 10th annual conference on Internet measurement*, pages 1–14. ACM, 2010.
[22] G. Wang and T. Ng. The impact of virtualization on network performance of amazon ec2 data center. In *Proc. of the IEEE Infocom*, 2010.