Using Proxies to Accelerate Cloud Applications

Jon Weissman and Siddharth Ramakrishnan
Department of Computer Science and Engineering
University of Minnesota, Twin Cities

Abstract

A rich cloud ecosystem is unfolding with clouds emerging to provide platforms and services of many shapes and sizes. We speculate that future network applications may wish to utilize and synthesize capabilities from multiple clouds. The problem is this may entail significant data communication that derives from the client-server paradigm imposed by most clouds. To address this bottleneck, we propose a cloud proxy network that allows optimized data-centric operations to be performed at strategic network locations. We show the potential of this architecture for accelerating cloud applications.

1 Introduction

The explosion of data produced by simulations, network-connected instruments and sensors, as well as social and political data harvested from the Web (images, movies, blogs, etc.), is ushering in a new era of data-intensive computing and data-driven scientific inquiry. For example, projects in Geoscience, Astronomy, and High Energy Physics are routinely producing terabytes of data [12, 15] that may hold answers to deep scientific questions. This vast sea of available data provides new opportunities for a large class of distributed applications to solve new problems by utilizing multiple sources of data. Representative data-oriented application classes include distributed data mining, distributed workflows, Web 2.0 Mashups, to name a few.

To meet the data storage and computing demands of such data-rich applications, a parallel trend is emerging that is commonly referred to as cloud computing [7]. Clouds currently represent a diverse landscape containing resource providers, application hosting platforms, and service providers. In this paper, we use the term cloud broadly to refer to a large-scale out-sourced data, compute, or service resource, made available to end-users and applications. Many clouds today are optimized for very specific functionality. For example, the IBM/Google MapReduce cloud is specialized for MapReduce computations, while the satellite Earth Imagery cloud (Google Earth) [10] is specialized for providing geographical data. This trend toward specialization is likely to continue, which means that many of the data-oriented applications described above must increasingly interact with and utilize multiple specialized stand-alone clouds. Today, the end-user application is responsible for interacting with these different cloud services individually, downloading or uploading data, performing any needed intermediate computations, and generating a final result. This can lead to significant performance bottlenecks. The current cloud interaction paradigm is client-server (e.g. Web Services or http) which forces all output data to flow back to the client even if it is intermediate in the end-to-end application. The problem is even more pronounced if the end-user application is resource constrained, e.g., if the end-user is using a thin-client such as a PDA, or if the network path to the end-user application is poor.

In this paper, we propose a novel solution to address the bottlenecks inherent in the current cloud interaction paradigm: the provision of a *cloud proxy network* that allows optimized data-centric operations to be performed at strategic network locations. In our model, proxies may take on several data-centric roles: they can interact with cloud services, route data to each other, cache data for later use, and invoke compute-intensive data operators for intermediate processing. The power of our proxy approach lies in the multiplicity of their roles and interconnection. In this paper, we show the potential of proxies for accelerating network applications by exploiting network diversity.

2 Related Work

Our project is related to work in a number of areas: data-intensive programming systems, cloud computing, network computing, Web 2.0, volunteer systems, and proxy architectures. Many emerging cloud systems such as Amazon EC2/S3 [4] and Google Apps [9] provide resources to third-party applications. There are many data clouds, e.g. Google Earth [10] and Sloan Digital Sky Survey [15] that serve data to end-users. These systems are optimized for use within a single cloud or data center, however, a challenge is transferring the underlying data and the results to/from these clouds.

Volunteer edge computing and data sharing systems are best exemplified by Grid and P2P systems including, Bittorrent [3], Globus [5], BOINC [1], and @home projects [14]. These systems provide the ability to tap into donated resources such as CPU capacity or network bandwidth, but their focus has been on single applications vs. service hosting.

Internet programming systems that support content mashup and state sharing, include Yahoo Pipes [16], GME [8], and LiveMesh [13], which can be used to connect applications and/or distributed data.

Our approach is novel in that proxies may assume a diverse set of roles unlike other systems in which network nodes either compute, route, serve data, or invoke services. Enabling proxies to assume multiple roles is key to the performance and reliability of distributed data-intensive multi-cloud applications.

3 Proxy Architecture/Application Model

System Model: The system model consists of a set of cloud services logically connected by a proxy network, and has three principle entities: (1) cloud services, (2) proxy network, and (3) application initiator (Figure 1). *Cloud services* (S_i) may represent a broad array of hosted services from large scale Internet ones (e.g. Google Maps), to scientific services (e.g., Sloan Digital Sky Survey), to platform services (e.g. EC2/S3). In this paper, we use the term cloud broadly to refer to a large-scale out-sourced data, compute, or service resource, made available to end-users and applications. We can categorize clouds into different types based on the kind of resource they provide. A *data cloud* is a large-scale data source, such as data sensors that may be producing raw data, as well as data servers that may be storing or archiving large databases or third-party data. Examples could be Sloan Digital Sky Survey [15] or Amazon S3 [4]. A *compute cloud* provides computation resources for data-dependent computing. An example could be Amazon EC2 [4]. A *service cloud* provides specialized services that can be used by other applications. Google Maps [11] can be an example.

The proxy network consists of a large number of logically connected edge nodes that may assume a rich set of data roles to boost the performance and reliability of distributed data-intensive applications, including:

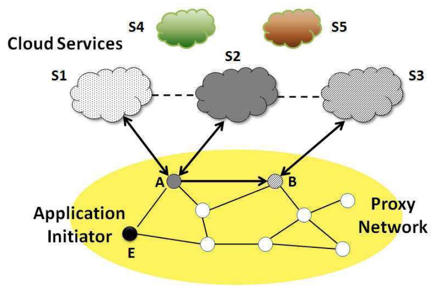


Figure 1: System Model: S_i are the cloud services, nodes A and B are proxies, and E is an initiator for an application that uses clouds S_1, S_2 and S_3. Solid arcs represent actual proxy-to-proxy and proxy-cloud interactions, and dotted lines represent logical cloud-to-cloud interactions.

• *Cloud service interaction:* A proxy may act as a client to a cloud service. This role allows a proxy with better network connectivity to access one or more cloud services. For example, a proxy may have very high bandwidth to/from a cloud service relative to the end-user.

• *Computing:* A proxy may carry out computations on data via a set of data operators. This role allows a proxy to filter, compress, merge, mine, and transform (i.e. act as a shim) data. We envision a set of well-defined *data operators* C_1, C_2, ... C_k, where C_i: $D_{in} \rightarrow D_{out}$, that is, C_i maps an input data into an output data.

• *Caching:* A proxy may efficiently store and serve data to other nearby proxies that may consume the data later on. Proxies can also cache intermediate results from a cloud interaction that may be reused again.

• *Routing:* A proxy may route data to another proxy as part of an application workflow. This role allows a proxy to efficiently send data to another proxy for additional processing, caching, cloud service interactions, etc. This role is particularly important if the application is interacting with multiple clouds which are all widely distributed, and there may be no single proxy that can efficiently orchestrate all of these interactions.

The real power of proxies lies in the combination of these roles. As a simple example, suppose the end-user is on a PDA behind a poor network connection. We can use a proxy with high bandwidth to interact with a set of service clouds to fetch a large amount of output data, and then use its computing power to process the data.

Finally, the *application initiator* is a node (E in Figure 1) in the proxy network which acts on behalf of an application, e.g., an end-user machine, job-scheduler, etc. The initiator acts as an application control point and the place where resource allocation decisions are ultimately made. Note that the end-user may or may not be located on the initiator, and could instead be using the initiator as a control point in the proxy network. There can be many initiators within the proxy network, and a node can act as a proxy or initiator at different times. In this paper, we evaluate the merits of the proxy network that is neutral to its deployment model.

Application Model: The proxy network operates on behalf of the application to accelerate its performance. Each entity in the system model has specific strengths that we wish to exploit on behalf of distributed applications: (i) the application initiator is best for holding private data and is the logical place for performing application control, (ii) the proxy network is best for providing edge resources at diverse network locations, and (iii) the cloud is best for providing critical data, maintaining shared state, and strong performance guarantees. In particular, the proxy network provides scalable resources that can be coupled to clouds or other proxies by exploiting low latency and high bandwidth connections, relative to the application initiator.

A distributed application is a coupling of these three entities. For example, Figure 1 depicts a scenario where an application invokes two cloud services S_1 and S_2, process their outputs via operators (e.g filter and merge), and then uses this intermediate data as input to another service S_3 which produces the final output. Two proxies A and B have been selected to accelerate this application. Here, A invokes S_1 and S_2 in parallel, processes the output, and routes the data to B, which invokes S_3. A and B would be chosen for their network performance to S_1 and S_2, and S_3, respectively, relative to E.

The proxy network is driven by three distributed data-intensive application scenarios that can reap substantial benefits from our approach:

• *Distributed Internet Workflows:* Data-intensive workflows are an important class of distributed applications. In prior work [2], we have demonstrated the acceleration potential of proxies for an astronomy application by routing data directly between the application services on high bandwidth paths, avoiding the initiator which was running the workflow engine.

• *Distributed Data Mining:* Distributed data mining enables data from different sources to be combined and analyzed to discover correlations. The proxy network can be used to accelerate this type of application. For example, in Figure 1, proxy A could be performing data mining on data retrieved from S_1 and S_2.

• *Mashups:* Mashups have a similar structure to distributed data mining applications - the only difference is that the proxy operator is typically a merge rather than a data mining operation.

4 Evaluation

4.1 Network Microbenchmarks

We have performed an extensive evaluation of the proxy network concept using nodes across PlanetLab. We have focused on the network benefits afforded by proxies in their roles as cloud service clients and routers in this evaluation. The first question we posed was to what extent do we see network diversity (latency, bandwidth) from the perspective of proxy nodes to/from cloud services deployed today. This would determine whether there are opportunities for placing proxies between end-clients and cloud services. The results indicate that despite plentiful bandwidth in the network core, nodes do in fact exhibit network diversity with respect to different cloud services. We chose a wide-area collection of international PlanetLab nodes and a selection of 35 different commercial web services (i.e. our cloud services). In each case, the PlanetLab proxy recorded the latency (measured as the time to fetch a very small file) to each service, (Figure 2). We show a random sample of 12 nodes and each point represents the average of at least 5 trials. The complete set of proxies is 48 nodes and we also plot the best, worst, and median observed values for comparison. Looking vertically, the results indicate great diversity in measured latency suggesting that some proxies have far better connectivity than others. In addition, there appear to be a rich set of options with many proxy nodes lying below the median line (if we assume the application initiator or end-client lies at the median). We also see that the benefit for a specific proxy may vary with the specific cloud service. For example, *planetlab1.if1.uio.no* (the right facing triangle) typically appears well below the median line, but in a few cases is above it. (e.g. at web service 26). We have observed the same broad patterns for download speed. The next question we asked was: is the diversity sufficient to provide an end-to-end performance benefit? Since the use of a proxy introduces an additional network hop (end-client-to-proxy-to-cloud service, and back), will any real end-to-end benefits be seen? To help answer this question we examined traceroute data for the same set of 48 nodes and looked at all of the paths among them (48x47). The analysis looked at all paths of the form A-B, and checked whether any for intermediate node C, the path (A-C, C-B) was cheaper than A-B. Out of 1600 total paths (48*47 minus the traceroute failures), 1106 paths (or 70%) were improved by the addition of a single intermediate proxy, and a majority (≈70%) were improved by over 20% when compared with the default path.

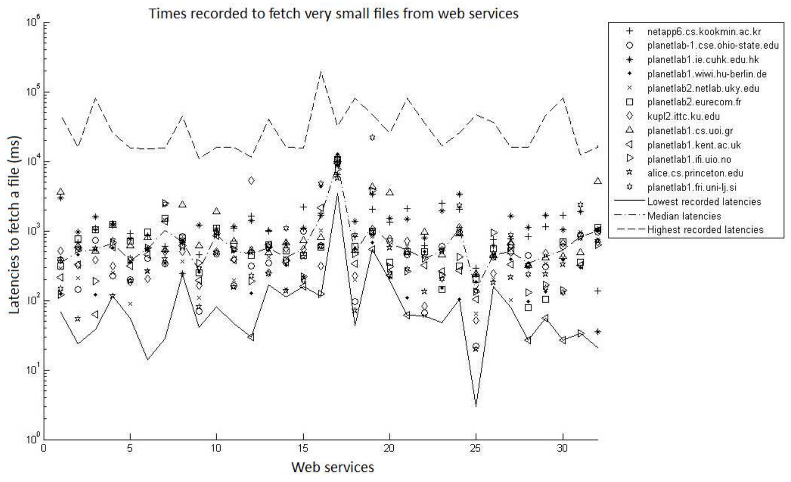
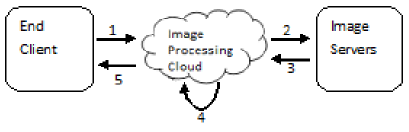
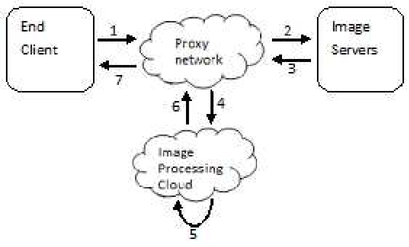


Figure 2: Latency from proxies to different web services. Each proxy is represented by a different shape.



(a) Basic Workflow



(b) Proxy Workflow

Figure 3: Application Workflow

4.2 Application Scenario

We now look at a specific application scenario: an image transformation cloud service (Figure 3(a)). In this example, the images to be processed are stored on an image server separate from the end-user location. The end-user selects an image for transformation by a contacting the cloud service (step 1). This cloud service fetches the image from the image server (steps 2/3), performs the transformation (step 4), and returns the adjusted image to the end-user (step 5). Based on triangle inqeuality violations [6] in the network, it is possible that the insertion of a proxy can improve the end-to-end performance. The reason is that it may have better network connectivity to the image server and/or image processing cloud, or both (Figure 3(b)). In the experimental setup, the image transformation performed was a resizing that reduced the image to a quarter of its size from 228KB to 48KB. We deployed this code as service on a varying set of nodes. We do the same for the image server and the end-user. All nodes are on PlanetLab. We present some representative results - all data is the average of at least five trials. We fix the end-client and the image server varying the location of the image processing servers and the proxies (Figure 4). The same result pattern is seen for many other end-client, image processing, node combination, and we present one such graph. Along the X-axis, we vary the locations of the image processing servers. Each vertical point shows the performance obtained when using a different intermediate proxy (over 40 proxies are shown). The circled points indicate when a direct connection without a proxy achieves the best performance. However, proxies can improve performance in many cases (the trail of proxies that lie beneath the * mark). In this graph and many others, at about 70% of the image processing server locations, proxies can accelerate performance. In some cases, it is only a few proxies (image server 4 and 6), and in others, many proxies (image server 5, 9, 11). The explanation for why the proxies improve performance even with the addition of the extra hop is explained by the network diversity results (Figure 2) and the triangle inequality violations in the network.

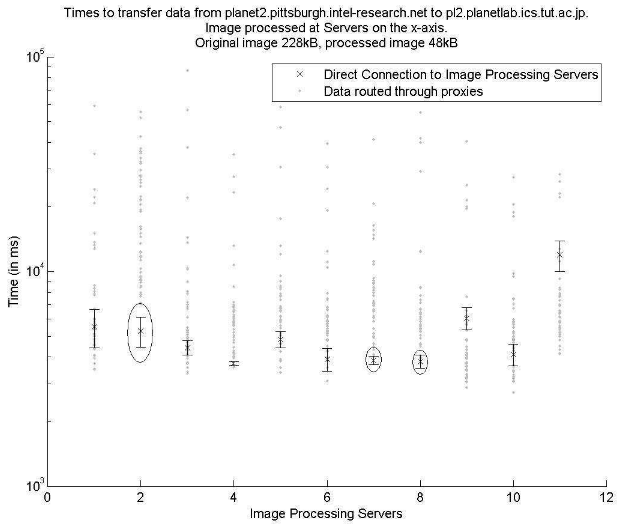


Figure 4: Proxy Acceleration: Pittsburgh End-client, Japan Image server. Each proxy is represented by a light dot and the direct path with error bars by *

5 Acknowledgments

The authors would like to acknowledge Abhishek Chandra for his input and discussion on this work.

6 Conclusion

In this paper we have introduced the concept of a cloud proxy network to accelerate applications spanning multiple specialized clouds. We described the broad features of the proxy network and presented specific evidence of its potential to improve the network performance of cloud applications by exploiting its network diversity. Our future work consists of demonstrating the full power of the proxy nodes as they take on a diverse set of roles including caching and computation. We also plan to explore techniques that can identify suitable proxies automatically based on application needs.

References

[1] D. P. Anderson. BOINC: A System for Public-Resource Compting and Storage. In *Proceedings of the 5th ACM/IEEE International Workshop on Grid Computing*, 2004.

[2] A. Barker, J. Weissman, and J. Hemert. Eliminating The Middleman: Peer-to-Peer Dataflow. In *Proceedings of HPDC*, 2008.

[3] B. Cohen. Incentives build robustness in BitTorrent. In *Proceedings of the First Workshop on the Economics of Peer-to-Peer Systems*, June 2003.

[4] Ec2/s3. `http://aws.amazon.com`.

[5] I. Foster et al. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. In *Proceedings of the Global Grid Forum*, June 2002.

[6] Stefan Savage et al. The end-to-end effects of internet path selection. In *ACM SIGCOMM*, Aug. 1999.

[7] D. Gannon. The Computational Data Center. A Science Cloud, 2008. `http://www.extreme.indiana.edu/~gannon/science-data-center.pdf`.

[8] Google mashup editor. `http://code.google.com/gme/`.

[9] Google apps. `http://www.google.com/apps`.

[10] Google earth. `http://earth.google.com`.

[11] Google maps. `http://maps.google.com`.

[12] LHC. `http://www.uslhc.us/`.

[13] Live mesh. `http://www.mesh.com`.

[14] D. Molnar. The SETI@Home problem. *ACM Crossroads*, September 2000.

[15] Sdss. `http://www.sdss.org/`.

[16] Yahoo pipes. `http://pipes.yahoo.com/`.