CLAudit: Planetary-Scale Cloud Latency Auditing Platform

Ondrej Tomanek
Czech Technical University in Prague
ondrej.tomanek@fel.cvut.cz

Lukas Kencl
Czech Technical University in Prague
lukas.kencl@fel.cvut.cz

***Abstract*—Latency is an important, yet often underestimated aspect of the nascent Cloud-Computing scenario. A cloud service based on processing in remote datacenters may exhibit latency and jitter which may be a compound result of many various components of the remote computation and intermediate communication. Our broad vision is to design and develop tools to monitor, model and optimize the global cloud-service latency. To this end, we introduce CLAudit, a prototype planetary-scale cloud-latency auditing platform. It utilizes the experimental PlanetLab network to place globally distributed probes that periodically measure cloud-service latency at various layers of the communication stack. We present CLAudit architecture in detail and show initial test-measurements of the Microsoft Windows Azure cloud service, demonstrating the platform's practical usefulness by showcasing a few discovered anomalous results in the cloud-service latency measurements.**

I. Introduction

The inherent delay in computer-communication networks has been a long-standing problem, even accelerated nowadays by the boom in distributed applications related to the Internet and Cloud Computing growth. The key problem is the great demands of the new applications, both for computing resources and for the quality of the communications network. While the more obvious problem of computing-resources allocation is being deeply investigated, research in improving network-quality parameters, throughput apart, appears not to get as much attention. As a consequence, end-users' time and patience is often the price, and deployment of latency-sensitive applications (e.g. games) in the general cloud has been slow, often favoring proprietary solutions. Cloud-computing customers — end users and tenants — thus still stand in wait of satisfactory answers to their application-performance requirements.

Cloud-service latency may play a role in many kinds of applications. Starting with the simple web-based solutions all the way to collaborative applications, ranging from documents sharing across voice and video telepresence to haptic-operated distributed games or interactive shared 3D worlds. There can be orders of magnitude differences between their latency requirements, ranging from units of seconds to units of milliseconds. Low latency demands are not limited to those applications, but also to the practices and designs inherent to the Cloud- Computing infrastructure, such as replication, task distribution, sharing, synchronization, offload or rapid scaling. Operation with stringent latency requirements is thus only possible after extensive latency-based optimizations.

Due to the diverse Internet growth trends and increasingly complex designs of the Cloud-services infrastructure, latency keeps becoming next-to-impossible to express using analytical methods. Indications of complicated latency representation and treatment appeared already during the golden-era of Internet research. Promising end-to-end efforts (e.g. DiffServ QoS) were either not deployed at large scale or remained on paper only. With the emergence of Cloud Computing, this topic is open again and the latency problem has become even more pressing because of the new traffic patterns and complex use cases. Understanding origins and symptoms of latency is a prerequisite to tackling it, however, even rigorously monitoring latency is non-trivial in the complex Cloud environment. The planetary scale of Cloud services demands planetary scale of latency handling as well.

Due to the generally local nature of latency-reduction tools, global end-to-end optimization, taking all current Cloud state information into account, might be neither effective, nor feasible, without a globally informed view. We thus target *global* latency monitoring and modelling. As near real-time monitoring would be needed for employing network-feedback, new approaches delivering rapid latency information need to be sought. To this end, we have assembled CLAudit, a planetary-scale Cloud Latency Auditing platform. CLAudit addresses both the latency-monitor architecture design as well as the problem of systematic planetary-scale probe placements. As presented in this work, CLAudit and its prototype implementation stand out for the following combination of features:

- *Globally-distributed client nodes deployment.* Thanks to PlanetLab use, we select when and where the measurements occur. Also, Planetlab presents us with a set of homogeneous clients.
- *Datacenter point-of-view measurements.* The platform provides options for measuring communication latency of interoperating servers using probes hosted in datacenters.
- *Various ISO/OSI layers are measured.* CLAudit thus reveals protocol-level behavioral differences and benchmarks the obtained results. For example, TCP/SQL storage (on-premise and remote) is included in our testbed that simulates a multi-tier WEB 2.0 application.
- *Wide range of results application.* Gathered data and inferred concrete or general results can serve not just for the latency model design, but also for general Cloud-service provider benchmarking, reports and other applications useful to both the Cloud-Computing public and other computer networking communities.

978-1-4799-0568-3/13/$31.00

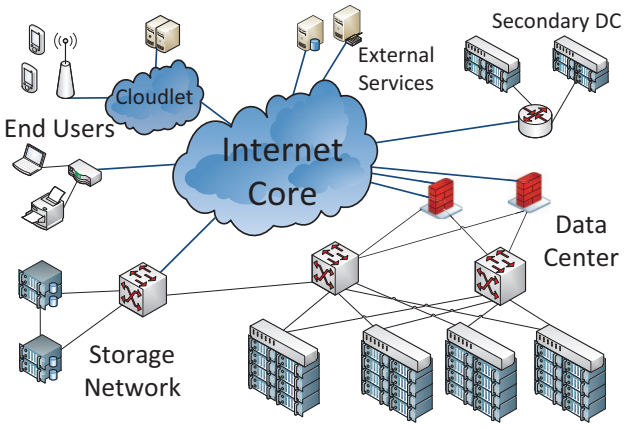


Fig. 1. *Cloud Computing system.* Example of various elements and components involved in Cloud Computing. Elements and components can also be viewed as latency-contributing sources.

II. Problem Statement

By Cloud Computing we understand the model for enabling ubiquitous, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service-provider interaction [1]. The Cloud, or the Cloud-Computing system, is the set of elements and components taking part in a particular Cloud Computing scenario (see Fig. 1). By Cloud elements we understand its elementary parts (e.g. end-user SOHO router or datacenter edge-firewall uplink) and by Cloud component the set of logically grouped elements (e.g. Internet core or Storage network). By Cloud service we understand the set of Cloud service-provider offerings contracted to tenants; and by Cloud application the tenant's application deployed on top of a particular Cloud service and possibly serving end-users via clients. Both the tenants and end users are Cloud customers.

A communication session is the information exchange inside the Cloud system (in this paper it concerns the client-to-application interaction). Such a session often consists of a master session and a number of slave sessions, necessary for completing the master session. This scenario is widespread because of the distributed nature of Cloud Computing (e.g. DNS queries, OAuth queries or SQL queries nested inside a web application). Our unit of interest is the master session, whose end-to-end latency is directly perceived by the end-user. By Cloud latency we thus understand the aggregate delay of Cloud elements involved in the entire communication session, with jitter being the latency variation [2]. By element delay we understand the reciprocal of its processing speed (e.g. wire propagation or request processing). The involvement of Cloud elements in a particular communication session is dictated by the Cloud-system state and the Cloud application in use.

The problem we address is how to rigorously measure Cloud latency. To an observer, a remote datacenter appears as a black-box, and various tricks are necessary to separate different possible latency causes. We propose how to extract the Cloud-latency information by carefully planning the architecture and deployment of CLAudit. Another problem is correct results interpretation, including the various deviations occurring due to the dynamics of the Cloud-Computing infrastructure.

III. Related work

Majority of the past latency-tackling efforts came from the Internet traffic-engineering domain. Results exist in the form of theoretical concepts, commercial solutions, RFCs and standards. Theoretical foundation of latency engineering was given by Queuing theory [3], [4] and Network calculus [5], [6]. Specific solutions such as guaranteed service networks or end-to-end jitter bounds were discussed in [7] or [8].

Commercial and research testbed implementations focus on reducing latency in certain part of the Cloud. WAN acceleration heuristics [9] or careful carrier selection algorithms are representatives of WAN-focused optimization. CDNs and Cloudlet research also belong to this category [10], [11]. TCP optimizations [12], [13], [14] have been proposed, adjusting TCP behavior within the datacenter network. Predictable datacenter performance is becoming a hot topic in general [15].

Miscellaneous Cloud measurements and analyses have already been conducted. Deriving traffic characteristics of flows inside a datacenter was the focus of [16] or [17]. Specific measurements concerning Cloud performance include, among others, [18] and [19]. End-user-perceived Cloud application performance measurement options and results were discussed for example in [20], [21] and [22]. A common goal of all these efforts is standardization of Cloud Computing and definition of the relevant metrics [23].

IV. Solution architecture

A. Platform overview

CLAudit is designed to be generally applicable and consists of four major platform components (see Fig. 2):

1) Clients: Clients play the role of real end users of the Cloud applications and work with identical protocols so that the measurement results correspond to what end users really perceive when using Cloud applications. Clients nodes are geographically distributed throughout the whole world so that various end user perspectives can be evaluated. Clients nodes run the same OS and use the same versions of software packages to conduct measurements. Such synchronization provides homogeneity, which, to some extent, eliminates the influence of end-user difference on the data. Clients are centrally instructed to send probes against applications hosted on Application servers to obtain protocol delays from respective RTTs. Probes exist for various ISO/OSI layer protocols and as such the Clients obtain different perspectives on target application, which allows aids to better reasoning in subsequent evaluation phases. Client nodes are deployed in triplets for the purposes of redundancy and anomaly-discovery verification.

2) Application servers: Application servers are client-request serving machines deployed in the Cloud data centers. There is no difference between a CLAudit Application server and a regular server - servers are provisioned exactly the same way as in the production environment and various real applications are being deployed and executed on these servers. The server has no knowledge of whether is it serving requests from end user clients or a CLAudit Clients. Application servers are geographically distributed across a global set of data centers so that we retain the property of global Cloud application in our measurements and enable evaluation of the distributed applications.

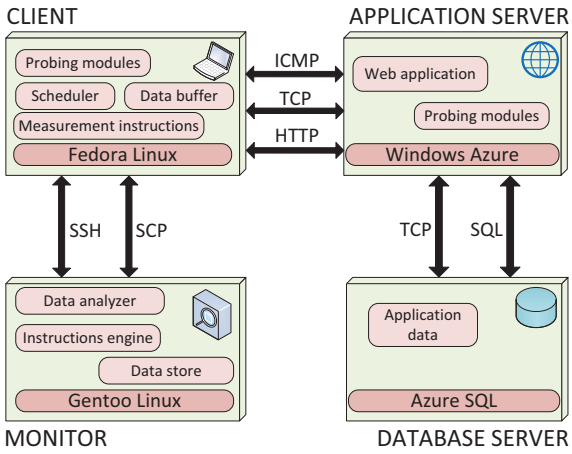


Fig. 2. *CLAudit prototype component model.* Four major components along with respective execution environments hosting key artifacts and modules for inter-component protocol interactions.

3) Auxiliary servers: The only optional component of CLAudit. Auxiliary servers are involved only in cases when Application server does need additional external information to compose the client reply. Auxiliary servers are thus being queried by the Application servers and this process increases the overall user-perceived application latency. Examples of such Auxiliary servers are authentication services or data storage (e.g. SQL database service).

4) Monitor: Latency Monitor is a single *central* entity, deployed in a redundant fashion for the purpose of a smooth failover. Its roles include instructing Clients about measurements, collecting the measured data, publishing the data and analyzing the data.

B. Prototype implementation

We have assembled the platform prototype using readily available tools and platforms (as shown in Fig. 2). We have utilized the PlanetLab [24] global research network for worldwide client nodes distribution. All client nodes are Planetlab i386 nodes running Fedora Core 8 and are provisioned with modified open-source software packages for probe-generation and RTT-measurement purposes. Microsoft Windows Azure [25] serves as the platform for hosting Application servers and Windows Azure SQL hosts the database.

The most important role of the clients is to gather data using probing modules, i.e. mechanisms for generating and sending various protocol messages as when using a real Cloud application. Probes are being sent from each client to each Application server (in our prototype, these are HTTP probes using the `httping` utility [26], TCP probes using the modified `tcping` utility [27] and ICMP probes using the `traceroute` utility). This modular architecture allows for easy extension with new probe types supported by various Cloud applications. After the Application server processes the protocol request, it replies with a relevant response packet back to the client. This request-response nature of many existing protocols allows to measure RTTs and derive latency.

A scheduler running on each client manages the measurements (e.g. tunes the probing intervals). Measured RTTs are stored locally along with timestamps. All clients have their clock synchronized via the Network Time Protocol (NTP) to enable comparison of the asynchronously measured data.

The measured Cloud application is a web application hosted on the service-provider's platform and running in an execution container. Beside this genuine Cloud-deployed application there are also probing modules on the web application-server side. Their purpose is to measure the slave-session latency when the web server and the Auxiliary server exchange information. The prototype probing modules are SQL probes implemented using the *PHP PDO API* and TCP probes implemented using the *raw TCP socket PHP API*. We choose these types of probes as, firstly, we seek user-experience improvement and as such we imagine various combinations of web-application deployment scenarios and database uses so that, for example, user-perceived latency of a multi-tier WEB2.0 application can be measured. Secondly, we benchmark the application-layer protocols against TCP RTTs, as discussed in section VI. Fig. 3 depicts an example of a sequence of interactions perceived by a user via the accumulated latency of individual protocol exchanges and request-processing times.

The Auxiliary server used in the CLAudit prototype is an ordinary SQL database server. We have used the availability of Microsoft Azure SQL platform and deployed the databases to various Azure datacenter locations. Each deployed database is being queried by every deployed web application and no platform-specific logic resides on the database servers.

The Latency Monitor interacts with the Clients only via the SCP/SSH channels. It stores the data downloaded from the Clients for backup and analysis. The data analyzer running on the Monitor interprets the data. There is also an instructions engine on the Monitor that instructs clients to adjust their measurements if needed.

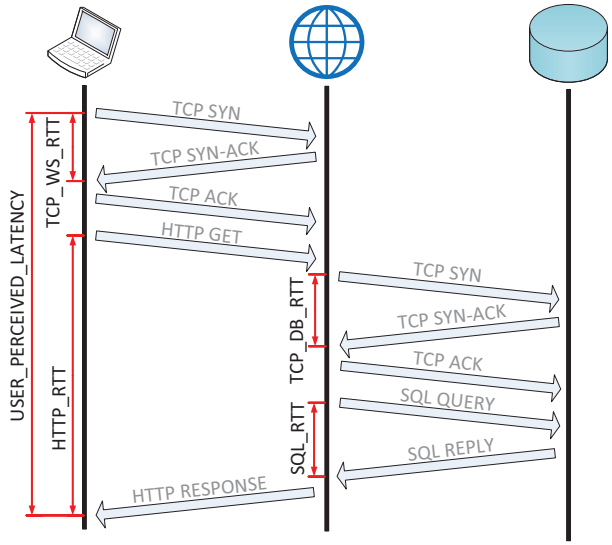


Fig. 3. *Selected CLAudit-measured variables.* Sequence diagram of the client-web-server-database interactions along with three selected elementary round-trip time variables and two composite variables, as measured by CLAudit.

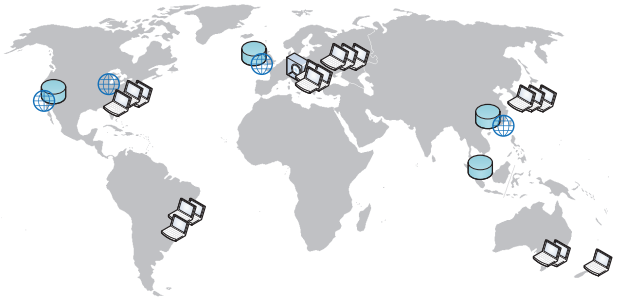


Fig. 4. *Map of CLAudit prototype deployment throughout the world*. Central monitor is represented with magnifier. Client nodes are represented with laptops, web servers with globe icons and database servers with cylinder icons. Client nodes are deployed in a redundant fashion in each region.

C. Prototype deployment

The CLAudit prototype deployment is depicted in Fig. 4. All the component instances are distributed worldwide, but not uniformly. Client nodes are deployed in representative locations on each continent apart from Africa. They reside in Washington, D.C. (USA), Belo Horizonte (BRA), Prague (CZE), Moscow (RUS), Osaka (JAP) and Melbourne (AUS). These client nodes exist in pairs, for backup purposes in case of PlanetLab maintenance or failure event. Measurements are conducted simultaneously from both nodes. In situations when one node is not operational or due to an arbitrary cause cannot measure, results from the other node are used. Other use cases for the secondary nodes include comparing measurement results and reasoning about anomalies.

Furthermore, third client nodes are deployed at nearby, but independent PlanetLab sites, because of the little performance guarantees of PlanetLab, so as to prevent the influence of campus outages or other local problems. Third nodes are located in Philadelphia (USA), Sao Paulo (BRA), Brno (CZE), Moscow (RUS), Nagoya (JAP) and Hamilton (NZL).

Application and database servers are also distributed globally and hosted in the Microsoft Windows Azure datacenters. Placement of the servers is driven by various popular application designs (e.g. geo-redundancy, storage distribution or load balancing). Various application-deployment combinations are thus evaluated, including the web server and the database server in the same datacenter, on the same continent or at the other side of the world. Web servers reside in West US, North US, North Europe and East Asia and database servers reside in West US, North Europe, South Asia and East Asia. The testing web application is deployed in the Azure Web sites container and is simple enough so that this dynamically generated web page can fit inside a single HTTP response packet.

The Azure SQL databases are linked to the web servers and serve their requests on demand. They do not interact with the clients directly, but only via the web servers. The servers again run in real datacenters and host real databases, which makes them indistinguishable from real Cloud deployment. Usage of databases in the prototype implementation is restricted to a single-packet SQL query answered by a single-packet SQL response containing numeric data (in this paper we are working with the SQL as a pure application layer ISO/OSI protocol).

The Latency Monitor is hosted on premises of the R&D Centre for Mobile Applications (RDC) at the Czech Technical University in Prague. It is a single i686 server node running Gentoo Linux OS. RDC also hosts a publicly accessible web page referencing the measured data and making it available to the general public. The project can be found at **http://www.rdc.cz/en/projects/CloudComputing/CLAudit**.

V. Testbed

A. Overview

We have launched the CLAudit platform prototype against the Microsoft Windows Azure cloud-computing platform and collected raw latency-related data over one month, resulting in a dataset of almost 400 MB in size. The presented measurement results concern one particular week from this period. The measurement goals were as follows:

1) *Evaluate user-perceived latency.* We have collected data from two application-layer TCP-based protocols (HTTP and SQL) with all the necessary underlying protocol overhead that adds up to the overall user-perceived latency.
2) *Benchmark latency.* We have set up two benchmarks and compared the measured data against them.
3) *Detect and monitor anomalies in Cloud behavior.* We have compared behavior at various protocol layers. Periods of normal behavior and anomalies are observed, using different perspectives, with results compared to filter out noise and aid detection.

B. Benchmarks

The two benchmarks, serving as bases for our comparisons, are the 2-way light propagation delay and TCP exchange delay. The light propagation delay (see Table I) is calculated as twice the great circle distance between two points, divided by the speed of light. This is a theoretical lower propagation-delay bound that cannot be superseded. Approximations inferred from this calculation are often used in practice, but their accuracy depends on many other conditions and thus we rather compare experimentally collected data against this bound.

The second benchmark is the median RTT of the initial packet exchange in the TCP-handshake process (SYN to SYN-ACK message exchange). This bound can be considered a

TABLE I. Lower one-way propagation delay bound (signal propagating at speed of light over the great circle distance between the two endpoints)

	W-US	N-US	N-EU	E-AS
BRA	35	27	3	59
USA	13	3	2	44
CZE	31	24	0.5	29
RUS	32	27	1	24
JAP	29	35	32	1
AUS	42	52	58	25

(a) Client to web server (ms)

	W-US	S-AS	N-EU	E-AS
W-US	0	45	27	37
N-US	10	50	20	42
N-EU	27	37	0	33
E-AS	37	1	33	0

(b) Web server to database server (ms)

realistic, lower two-way delay bound that in theory should not be superseded by any upper layer protocol RTT. However, in reality such situation can happen, not just due to the dynamic shared nature of Cloud Computing systems. An ICMP request-reply RTT cannot be used in our prototype due to protocol restrictions policy.

C. Preprocessing

Several issues had to be addressed for the measurements to be objective. We thus have dealt, where appropriate, with caches, warmed-up connections, pooled database connections, protocol versions, CDNs and ICMP-protocol message filtering.

HTTP is a popular protocol suitable for caching. As such it can reduce the number of unnecessary round trips and the overall amount of the HTTP data transferred. This unfortunately blurs our measurements in cases when we expect certain number of round trips to occur. Thus, client caching capabilities were turned off where appropriate in our measurements. A similar problem concerns HTTP version 1.1, which uses persistent TCP connection and thus prevents subsequent TCP handshakes to happen, which is not acceptable for us in certain situations. Microsoft's large CDN infrastructure can present end users with an illusion of the original resource or application being in close proximity, which again is not acceptable in cases when we look for data of the original Mega-datacenter RTT.

Web applications like to maintain database connection pools. These hold the TCP channel up, and following the first successful connection there are just SQL messages exchanged over the channel. Because we gather, beside others, raw periodic TCP data of the communication between the web server and the database, we force a TCP handshake to occur, by not taking the advantage of connection pools. Finally, SQL data caching on clients was prevented by always requesting random data from the database.

D. Setup

A scheduler running on the clients is instructed to read the instructions file every three minutes. In the instructions file there are series of loops within which the batch of five probes of each type is being sequentially sent to all targets. Every client thus records five RTTs for each valid type-target combination. The recorded quintuple of RTTs is then sorted in ascending order and the minimum and median values are recorded in a CSV file along with a timestamp, to deliver a latency time-serie (the purpose of keeping the median value is to smooth out RTT time-series and to filter outliers). The rest of the values in the quintuple are discarded and never used in subsequent processes. All RTTs are measured in milliseconds with the three additional decimal places, effectively making this microsecond-precision measurement. Due to impreciseness injected by excessive time multiplexing on Planetlab virtualized nodes we count with only millisecond precision. Failed round trips (or RTTs over 5 seconds) are represented with a 0 ms value and as such they can be easily identified in the resulting dataset. This setup implies that only persistent problems spanning tens of samples can be reasoned about.

Selected measured variables (or types), concerning the example interaction in Fig. 3, include:

1) *TCP_WS_RTT*. Time between the client sending a TCP SYN packet to a web server and receiving the TCP SYN-ACK packet back from the web server.
2) *HTTP_RTT*. Time between the client sending a simplest HTTP request packet to a web server and receiving the simplest HTTP response packet back from the web server.
3) *ICMP_RTT*. Time between the client sending an ICMP Echo Request packet to a web server and receiving the ICMP Time Exceeded packet back from the fixed farthest ICMP-reachable network hop in the target datacenter.
4) *TCP_DB_RTT*. Time between a web server sending a TCP SYN packet to a database server and receiving the TCP SYN-ACK packet back from the database server.
5) *SQL_RTT*. Time between a web server sending a simple packet containing an SQL query to a database server and receiving a packet containing the SQL response back from the database server.
6) *USER_PERCEIVED_LATENCY*. Total time between the client sending a TCP SYN packet to a web server and receiving the HTTP response packet back from the web server.

Note that database interaction is mandatory in the TCP_DB_RTT and SQL_RTT types, optionally involved in the HTTP_RTT and USER_PERCEIVED_LATENCY types and not present in the TCP_WS_RTT and ICMP_RTT types. Also note that data from the TCP_WS_RTT, HTTP_RTT, TCP_DB_RTT and SQL_RTT types represent the measured lower bound of the respective protocol RTT. Data from the USER_PERCEIVED_LATENCY type represent the user-perceived latency during a web-browsing scenario, with a possible intermediate query to a database.

Circa three minutes thus separate each recorded sample and every time-serie contains 480 median samples per day in the interval from April 29th 0:00 UTC to May 5th 0:00 UTC. Median and minimum time-series are stored in parallel. Eighteen clients record this data from their own perspective. There are four target web servers and four database servers, which produces a number of possible combinations of probe-type targets.

VI. Measurements

A. Examples

This section presents two examples of the CLAudit-gathered data. The data represent ordinary Cloud behavior without notable anomalies. Ordinary behavior is viewed from two different perspectives derived from the data.

1) Example 1: The first example focuses on the TCP_WS_RTT dataset subset (TCP RTTs between the clients and web servers gathered by extracting RTT from the initial packet exchange in the TCP handshake process). The target web server was located in the North US datacenter. Each client gathered 3360 data samples (resulting from 7 days-long measurement with single sample recorded every three minutes). Recorded samples are median RTT values from 3360 batches of TCP handshake quintuples sequentially being initiated by every client. Results from four representative

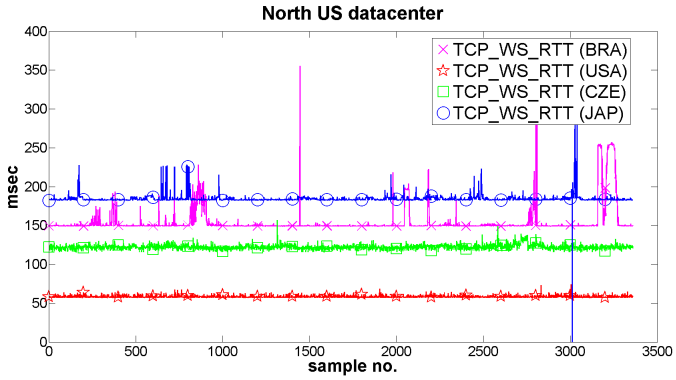


Fig. 5. *TCP_WS_RTT example*. A single week TCP RTT data time-series as recorded by clients handshaking with North US datacenter web server. The selected clients in Brasil, USA, Czech Republic and Japan recorded ordinary Cloud network behavior.

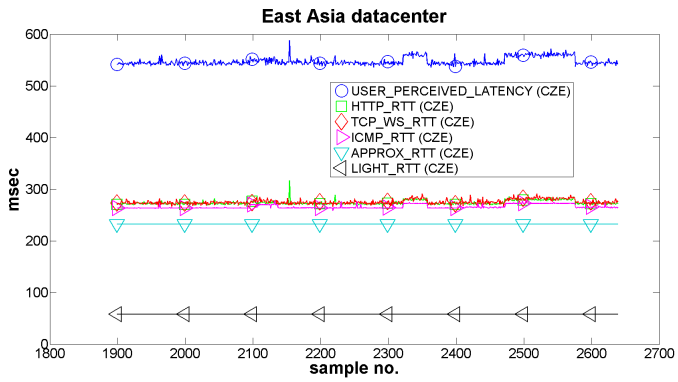


Fig. 6. *All time-series relevant for single web server as the target*. The data were collected by the Czech client communicating with the web server in the East Asia datacenter. Speed-of-light-derived variables have been precalculated. Also visualized is the obvious latency-level hierarchy among variables.

clients are presented on Fig. 5 as timely-ordered samples of RTTs recorded close to the respective time.

The US-client time-series seems very calm globally. It experiences low jitter and small number of negligible spikes. Also the firm lower RTT bound deserves attention. The most distant client, located in the Czech Republic, unsurprisingly experiences increased oscillations compared to other time-series.

The measurement thus shows more or less ordinary behavior with clients from farther locations and inferior infrastructures experiencing more negative effects and impaired behavior of the Cloud compared to clients in closer datacenter-proximity or clients from more developed infrastructures. Common denominator of all these time-series is the weekend effect, where there are little or no spikes, caused usually by high traffic load in the client-hosting campus network. All the time-series thus remain more or less calm with the exception of local effects. Datacenter-related effects did not occur.

2) Example 2: The second example reflects the dataset subset gathered using all the probe types relevant to a single web server as the target. CLAudit thus measured RTTs at various layers of the communication stack between the client and the Cloud web server. The target web server was located in the East Asia datacenter. A single client at a Czech Republic PlanetLab node gathered 740 data samples (resulting from almost 2 days-long measurement with single sample recorded every three minutes). Recorded samples are median RTT values from 740 batches of TCP, ICMP and HTTP packet quintuples sequentially sent to the web server. Aside from these we have also let full www interaction run and recorded user-perceived latency using the same procedure as with particular ISO/OSI protocols. This latency corresponds to what an end user perceives when opening the webpage for the first time. Finally we have included the lower propagation delay bound and frequently used the RTT approximation as four times the lower propagation delay bound. The plot represents timely-ordered samples of a particular round trip time measured in the corresponding time instance (See Fig. 6).

From the bottom of the Fig. 6 there are the Light propagation round trip (LIGHT_RTT variable), which in case of Prague and Hong Kong is 58 milliseconds, and an approximation of the real protocol RTT, inferred from the LIGHT_RTT (APPROX_RTT variable). These two constant values are calculated in advance and CLAudit takes them in as benchmarks. ICMP RTTs to the selected farthest ICMP-reachable datacenter hop usually float at the RTT level, slightly lower than TCP_WS_RTT and HTTP_RTT, because the packets did not have to traverse all the way through the datacenter to reach the web server. TCP_WS_RTT and HTTP_RTT behavior is virtually identical except for occasional spikes in HTTP due to the increased protocol sensitivity compared to TCP. The USER_PERCEIVED_LATENCY variable was measured independently from TCP_WS_RTT and HTTP_RTT, but nevertheless it very much looks like a sum of these two, because exactly one round trip of TCP followed by another of HTTP makes up the user-perceived latency.

B. Anomalies

Aside from Cloud-system behavior monitoring, CLAudit is also capable of revealing certain anomalies and providing background for their explanation. In this section we outline several such anomalies discovered during our measurements, analyze them and reason about them. Such anomalies are not unusual, but also not necessarily inevitable. We believe that intelligent monitoring and analysis can help decrease the occurrence of anomalies and contribute to making the Cloud more predictable to customers.

1) HTTP spike in North US: This anomaly concerns the datacenter in North US, particularly the web server that is hosting our Cloud web application (Azure Web site in Azure terminology). See Fig. 7. The incident began on Thursday after 6 p.m. UTC. It took almost 50 minutes for the Web server to fully recover and continue serving clients with regular latency.

The incident hit clients in Brasil, Czech Republic, Japan and Australia. The US and Russian clients followed in just a few minutes. HTTP latency increased by an order of magnitude for all clients — RTTs previously ranging from 50 to 500 milliseconds increased to over 3 seconds for over 30 minutes. RTTs started to decrease simultaneously after peaking and returned to a regular level after a circa 30-minutes-long recovery period. During this time there was absolutely no reflection of such an incident in the TCP RTTs targeting

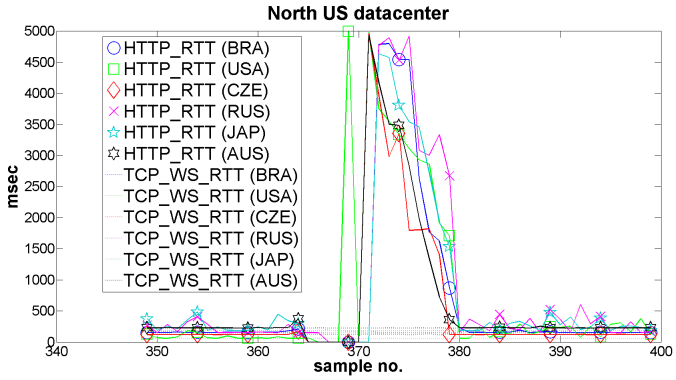


Fig. 7. *HTTP RTT spike and subsequent recovery*. Order of magnitude increase of HTTP RTT followed by a slow recovery as perceived by all CLAudit clients worldwide. The TCP RTTs targeting the identical web server remained stable, as shown in the bottom.

the same web server, effectively pointing to a web server issue as the only possible cause. Mechanisms such as TCP splitting or interception can prevent TCP connection to reach the same target as HTTP, but from the point of view of the clients the Transport layer performance did not change and Application layer performance degraded heavily. Interestingly, this anomaly occurred almost simultaneously also in the North Europe datacenter, which very much narrows the scope of the possible causes. These include VM migrations among datacenters or various types of attacks.

2) Path issue in West US: The second presented anomaly concerns the West US datacenter (see Fig. 8b). The combination of TCP, HTTP and ICMP data proved the problem‘s wide span and thus it simultaneously affected multiple Cloud applications. The anomaly was detected by CLAudit on Saturday after 12 p.m. UTC and completely disappeared on Saturday before 9 p.m. UTC. The data concerning globally-perceived ICMP issue are plotted on Fig. 8a and the Cloud applications degradation as perceived by US client is depicted in Fig. 8b.

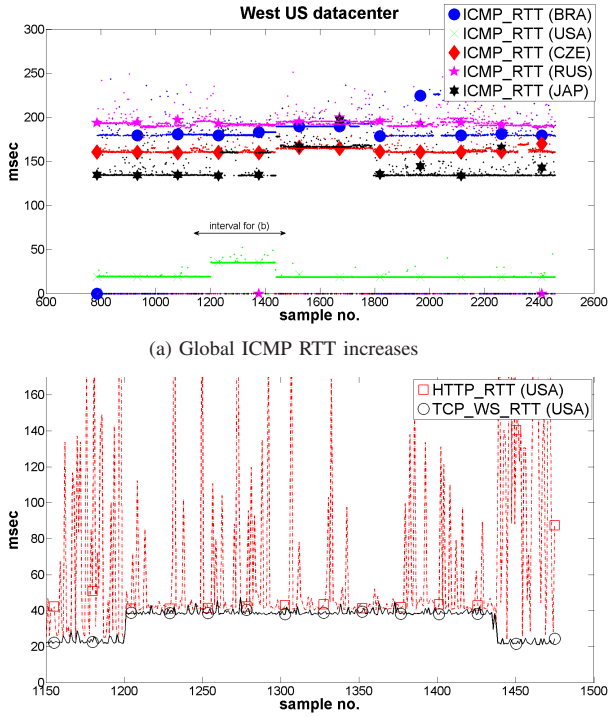
The anomaly was first perceived by the US client as persistent, almost doubled-up ICMP RTT (and transitively also TCP and HTTP RTT increase). This state suddenly disappeared at around 12 a.m. UTC on Sunday. Shortly after its disappearance the situation repeated for the other clients (Brasil, Czech Republic, Russia and Japan). The ICMP RTT increase was more intensive at the Japanese and Brazilian clients. The Czech and Russian client experienced lower, but still recognizable ICMP RTT increase. The situation suddenly disappeared for all the four nodes before Sunday 9 p.m. UTC and never repeated throughout the week to such extent.

Geographical client distribution proved no correlation to certain time of day of the anomaly occurrence. Symptoms point to routing manipulations and traffic diverts, possibly related to the weekend period. Whether intentional or not, this anomaly persistently degrades user-perceived latency for certain number of applications. Thus the end users of weekend-demand-peaking applications have their experience more impaired.

3) Periodic behavior of a multi-tier Cloud application: The last presented anomaly deals with the user-perceived latency of a multi-tier application. The Web server is located in the East Asia datacenter and the database server is located in the North Europe datacenter. Clients thus perceive latency of a geographically-partitioned application. Utilized data come from the USER_PERCEIVED_LATENCY dataset variable, which combines two TCP round trips, one HTTP round trip and one SQL round trip (see Fig. 3). The measured data exhibit predictable periodic characteristics, persisting throughout the whole week. The periodicity from Thursday circa 12 a.m. UTC to Friday circa 11 p.m. UTC is depicted in detail in Fig. 9a. The latency is toggling circa every 145 minutes between the stable period and the period of high jitter and latency value.

It is important to note that there is a significant gap residing in between the latency values. Japanese client-perceived latency hotspots are shown in a histogram on Fig. 9b. The cloud-service latency concentrates itself around the 725ms and 2200ms values. The rest of the values concentrates slightly above the respective values and is forming a sort of tail for both. The conclusion is that the end-user can expect two very different service qualities in very regular periodic fashinon.

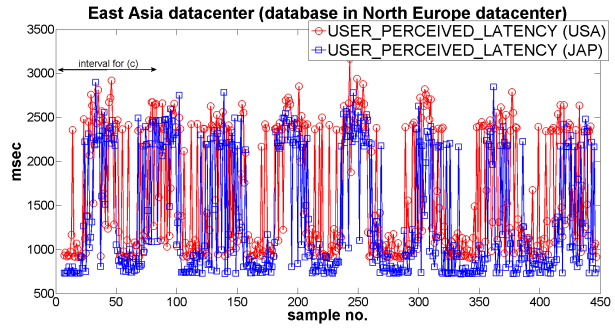
To a certain extent, this periodic effect has been found in *all other* datacenter deployment combinations of a multi-tier application. Deeper analysis in Fig. 9c reveals that the high-jitter periods are being introduced by the interaction of HTTP and SQL. The difference between HTTP RTTs, when either including or excluding the database interaction, proves a correct operation of the HTTP layer. Neither the TCP or SQL RTTs to the database server alone experienced any signs of fluctuation. The observed periodic effect could have a dramatic impact on a multi-tier application latency. For the moment we have been unable to determine its origin.



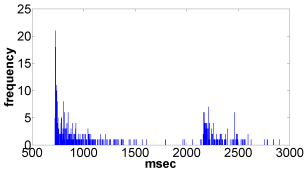
(a) Global ICMP RTT increases

(b) Services degradation from US viewpoint

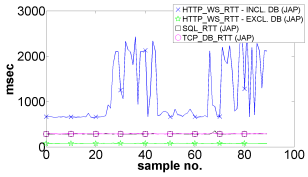
Fig. 8. *Path issues related to the West US datacenter*. Persistent ICMP RTT increase in West US datacenter perceived by the US client and followed by simultaneous persistent ICMP RTT increase perceived by four other clients. The data suggest routing manipulation as possible explanation.



(a) Observed latency periodicity anomaly (US and Japanese Client interacting with web server in East Asia using HTTP).



(b) Latency histogram (JAP)



(c) RTTs of relevant protocols

Fig. 9. *User-perceived latency periodicity anomaly*. Alternating increases and decreases in user-perceived latency of communication with a multi-tiered web application as observed during an ordinary work day. The measured latency switches between two levels.

VII. Conclusion

In this work we have presented CLAudit - a planetary-scale Cloud Latency Auditing platform. It is able to rigorously measure client-perceived latency of the deployed Cloud applications. The measurements occur at various network layers, which allows to detect anomalous behavior of the Cloud and benchmark the latency. We have tested the platform using a prototype deployment and demonstrated CLAudit's ability to detect both the normal and anomalous behavior of the Cloud. Using data analysis we were able to narrow the problem scope of the anomalies found. We hope that the platform will become a valuable tool serving not only the research community, but also the Cloud customers and service providers. Ongoing development and enhancement of the platform should broaden the scope of the covered Cloud-Computing platform and scenarios and thus increase its popularity among stakeholders.

There exists wide range of possibilities for CLAudit prototype extensions and enhancements. We would like to create close-to-real representative client node deployment and also employ mobile clients. The set of probe types can be extended by new services or different protocol versions. Automated statistical analysis of the Cloud-latency data should enable formalizing behavioral analysis and anomaly detection. When considering possible deployments, this essentially centralized prototype architecture does not scale easily for a large number of installations and is rather meant as a single point of reference for a large user community. Perhaps a more intelligently distributed and collaborative architecture would be needed for mass deployment. Ultimately, we target to deliver real-time latency-quality feedback so that faster Cloud-service optimization can be carried out and user experience is potentially minimally impaired. Philosophy-changing emerging paradigms like SDN could be employed to ensure rapid reorganization of the Cloud service. Finally, our efforts with Cloud measurements lead to a broader vision of evaluating the severity of latency-contributing sources, setting up various Cloud-latency metrics, creating a formal latency model and algorithmic latency optimization. Furthermore, new cloud networking protocols may emerge, addressing the various applications latency needs.

Acknowledgements

The authors kindly thank CESNET z.s.p.o for providing access to the PlanetLab infrastructure and Microsoft corporation for providing the Windows Azure Academic license. This work was supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS13/139/OHK3/2T/13.

References

[1] P. Mell and T. Grance, "The NIST definition of Cloud Computing," 2011, NIST Special Publication 800-145. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[2] C. Demichelis and P. Chimento, "RFC 3393: IP packet delay variation metric for IP performance metrics (IPPM)," *IETF, November*, 2002.

[3] R. Jain, *The Art of Computer Systems Performance Analysis*. John Wiley & Sons Chichester, 1991, vol. 182.

[4] J.-Y. Le Boudec, *Performance Evaluation Of Computer And Communication Systems*. Epfl Pr, 2011.

[5] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer, 2001, vol. 2050.

[6] Y. Jiang and Y. Liu, *Stochastic Network Calculus*. Springer, 2008.

[7] J. C. Bennett *et al.*, "Delay jitter bounds and packet scale rate guarantee for expedited forwarding," *IEEE/ACM Trans. Netw.*, vol. 10, no. 4, pp. 529–540, 2002.

[8] J.-Y. Le Boudec, "Application of network calculus to guaranteed service networks," *IEEE Trans. Inf. Theory*, vol. 44, no. 3, pp. 1087–1096, 1998.

[9] S. R. Smoot and N. K. Tan, *Private Cloud Computing: Consolidation, Virtualization, and Service-Oriented Infrastructure*. Morgan Kaufmann, 2011.

[10] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, 2009.

[11] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Cloudlets: Bringing the cloud to the mobile user," in *Proceedings of the third ACM workshop on Mobile cloud computing and services*, 2012, pp. 29–36.

[12] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowstron, "Better never than late: Meeting deadlines in datacenter networks," in *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4. ACM, 2011, pp. 50–61.

[13] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware datacenter TCP (D2TCP)," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 115–126, 2012.

[14] C. Lee, K. Jang, and S. Moon, "Reviving delay-based TCP for datacenters," in *SIGCOMM Comput. Commun. Rev.* ACM, 2012, pp. 111–112.

[15] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," in *SIGCOMM Comput. Commun. Rev.* ACM, 2011.

[16] S. Kandula *et al.*, "The nature of datacenter traffic: Measurements & analysis," in *Proceedings of the ACM IMC*, 2009, pp. 202–208.

[17] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the ACM IMC*, 2010, pp. 267–280.

[18] Y. A. Wang, C. Huang, J. Li, and K. W. Ross, "Estimating the performance of hypothetical cloud service deployments: A measurement-based approach," in *INFOCOM, Proceedings IEEE*, 2011, pp. 2372–2380.

[19] A. Pathak *et al.*, "Measuring and evaluating TCP splitting for cloud services," in *PAM, Proceedings*. Springer, 2010, pp. 41–50.

[20] M. Dhawan *et al.*, "Fathom: A browser-based network measurement platform," in *Proceedings of the ACM IMC*, 2012, pp. 73–86.

[21] M. Marshak and H. Levy, "Evaluating web user perceived latency using server side measurements," *Computer Communications*, vol. 26, p. 2003, 2003.

[22] J. Danihelka and L. Kencl, "Collaborative 3D environments over Windows Azure," in *Mobile Cloud, Proceedings IEEE*, 2013.

[23] A. Li, X. Yang, S. Kandula, and M. Zhang, "Cloudcmp: Comparing public cloud providers," in *Proceedings of the ACM IMC*, 2010, pp. 1–14.

[24] *Planetlab*. [Online]. Available: http://www.planet-lab.org/

[25] *Windows Azure*. [Online]. Available: http://www.windowsazure.com/en-us/

[26] *httping*. [Online]. Available: http://www.vanheusden.com/httping/

[27] *tcping*. [Online]. Available: http://www.linuxco.de/tcping/tcping.html