Latency–based Benchmarking of Cloud Service Providers

Vojtech Uhlir, Ondrej Tomanek, Lukas Kencl
Dept. of Telecommunications Engineering
Faculty of Electrical Engineering
Czech Technical University in Prague
{vojtech.uhlir, ondrej.tomanek, lukas.kencl} [at] fel.cvut.cz

ABSTRACT

With the ever-increasing trend of migration of applications to the Cloud environment, there is a growing need to thoroughly evaluate quality of the Cloud service itself, before deciding upon a hosting provider. Benchmarking the Cloud services is difficult though, due to the complex nature of the Cloud Computing setup and the diversity of locations, of applications and of their specific service requirements. However, such comparison may be crucial for decision making and for troubleshooting of services offered by the intermediate businesses - the so-called Cloud tenants. Existing cross–sectional studies and benchmarking methodologies provide only a shallow comparison of Cloud services, whereas state-of-the-art tooling for specific comparisons of application-performance parameters, such as for example latency, is insufficient. In this work, we propose a novel methodology for benchmarking of Cloud-service providers, which is based on latency measurements collected via active probing, and can be tailored to specific application needs. Furthermore, we demonstrate its applicability on a practical longitudinal study of real measurements of two major Cloud-service providers – Amazon and Microsoft.

1. INTRODUCTION

With Cloud service provider (CSP) offerings homogenizing, competitive differentiation starts to take place at the service quality level. However, CSPs only reveal insufficient amount of technical information about their service, leaving tenants indecisive. But, in fact, there are big differences in service quality among CSPs and even among single–CSP's datacenters (DCs), as was shown previously [17, 18].

One technique that enables decision support for selection of CSP and Cloud resources is benchmarking. The most accurate benchmarking method will always involve a cost–prohibitive trial deployment of the actual application. Cross–sectional studies and quick shallow benchmarking of Cloud resources via a test suite may be used instead, but pose a number of limitations: shallow nature provides a little or no explanation; restrictions imposed by CSP often lead to invalid comparisons and a short measurement timeframe leads to inaccuracies. Also, they often cannot answer questions related to global distributed applications.

In this paper, we propose a benchmarking methodology, based on latency measurements collected via active probing of Cloud resources at multiple layers of the network protocol stack. We focus on latency as it is among key performance parameters for the vast majority of applications. We then show an application of the Latency-based benchmarking on a case study of two major CSPs (Amazon and Microsoft) with similarly–located datacenters. The dataset is available to the research community and the general public [2].

The Latency–based benchmarking allows for:

- Comparison using diverse application requirements;
- Identifying a best-fit CSP/DC for a global client base;
- Identifying a best-fit back-end for a given front-end;
- Intra-DC, inter-DC or Internet networks applicability;
- Usage of in–house collected or third-party data;
- Performance estimations without actual deployments;
- Updatability of the results as new data get collected;
- Use as enhancement to existing benchmarking suites.

The remainder of this paper is organized as follows: § 2 provides an overview of the related work, § 3 introduces the formal terminology in use, § 4 describes the dataset and § 5 its preprocessing. § 6 describes the actual benchmarking methodology. § 7 presents a longitudinal study of the two major CSPs. We conclude and discuss implications in § 8.

2. RELATED WORK

Data transfer bottlenecks [10], performance unpredictability [5] and latency [15] are among major obstacles to Cloud growth. Benchmarking helps to reveal these, as was shown through its many use cases [9].

A number of cross–sectional (also called *breadth and snapshot*) CSP benchmarking tools considering network performance have been introduced previously: *CloudCmp* comparator [13] measures computation, storage and network resources, the latter using a TCP throughput and end-to-end response times. *Smart CloudBench* framework [6] deploys a transactional web–services benchmarking suite and, by measuring response times and recording error codes, estimates the cost–to–performance ratio. Custom–tailored benchmarking suites for testing CSPs can be created using [12]. In contrast, our network–latency–oriented approach works with weeks-to-months of collected RTTs (round-trip times) of relevant ISO/OSI layers. We do not consider throughput, to keep monetary costs low and avoid overloading networks.

UCC '16 December 06-09, 2016, Shanghai, China

ACM ISBN 978-1-4503-4616-0/16/12.
DOI: http://dx.doi.org/10.1145/2996890.3007870

For Cloud service status verification, various online dashboards exist, either run by a CSP itself (Microsoft [4] or Amazon [1]) or a third–party (*CloudHarmony* [3]). These do not offer in–depth comparisons.

Using the benchmarking output, providers can be ranked using multi–criteria decision–making techniques (as done by *CloudGenius* [14]) or utility theory and preference policies (as done by *CloudBroker* [7]).

A number of (often PlanetLab–based [29]) tools for obtaining suitable RTT measurements for latency–based benchmarking exist, namely *CLAudit* [17], *Fathom* [8] or *FlowPing* [19]. Data transformations, which the Latency–based benchmarking applies, are described in [16]. Our adaptations of metrics, used to describe application requirements, are based on [11].

3. TERMINOLOGY

A dataset file X contains $|X|$ single measurements x_m. A measurement x_m is described by the following:

- Measured protocol $l \in L$ (e.g. TCP);
- Internet Vantage Point location $v \in V$ (e.g. Planetlab host in Prague);
- Cloud front-end resource location $f \in F$ (e.g. Web server in Dublin DC);
- Cloud back-end resource location $b \in B$ (e.g. SQL database server in Singapore DC);
- Cloud Service Provider $p \in P$ (e.g. Microsoft).

Parameters v and b are mutually exclusive, reflecting that a measurement can only be conducted between v and f or f and b (i.e. Internet VP cannot directly reach a back–end resource). Using the above terminology, an input dataset for the Latency–based benchmarking is described as follows:

$$X^{L,V,F,B,P} = \{x_m^{l,v,f,b,p}\},\ m \in [1, 2, \ldots, |X|]$$

4. DATASET

The input dataset [2] was collected during the timeframe from January 10th 2016 to March 19th 2016, using *CLAudit* [17] – a system for collecting and evaluating multidimensional measurements. By *measurements* we mean RTTs of individual protocol exchanges. By *multidimensional* we mean measurements capable of being looked at from a point of view of Vantage Points, Data Centers and/or protocol layers. CLAudit consists of components, which reflect typical Cloud Computing application setups – i.e. Internet–connected client devices and DC–hosted front–end and back–end servers. RTTs of several protocol layers are recorded via active probing between every *(v, f)* and *(f, b)* pairs. A measurement iteration of every v and f consists of a 5 requests sent and up to 5 respective responses received, repeating every 4 minutes. RTTs are measured in milliseconds, rounded up to a nearest integer. All measurement types have a reasonable 10 seconds timeout set. The CLAudit deployment, also depicted in Fig. 1, is as follows:

- L = {TCP, HTTP, SQL} - Protocols used inside the dynamic web–based application that have round trips involved (TCP and HTTP between client web browsers and front–end web server; TCP and SQL between front–end web server and back–end SQL database)
- V = {AU, CZ, JP, US} - Internet Vantage Points, representing a global client base (Australia, Czech Republic, Japan, USA)
- F = {WUS, DUB, SING, EUS} - AWS and Azure front-end web server DC locations (Bay Area, Dublin, Singapore, Virginia)
- B = {WUS, DUB, SING, TOK} - AWS and Azure back-end SQL database DC locations (Bay Area, Dublin, Singapore, Tokyo)
- P = {P1, P2}. CSPs being compared. The names are obfuscated, as we only intend to demonstrate the Latency–based benchmarking methodology.

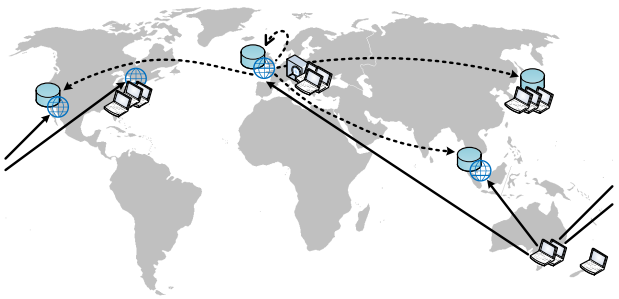


Figure 1: *CLAudit measurement platform.* Part of the CLAudit infrastructure used for Latency–based benchmarking demonstration. VPs are represented by laptops, front-end servers by globe icons and back-end servers by cylinder icons. Benchmarking computation is done at a central server, represented by lens icon. Continuous and dashed curves depict VP to front-end and front-end to back-end measurements, respectively (only measurements triggered by Australian VP and Dublin front–end are shown).

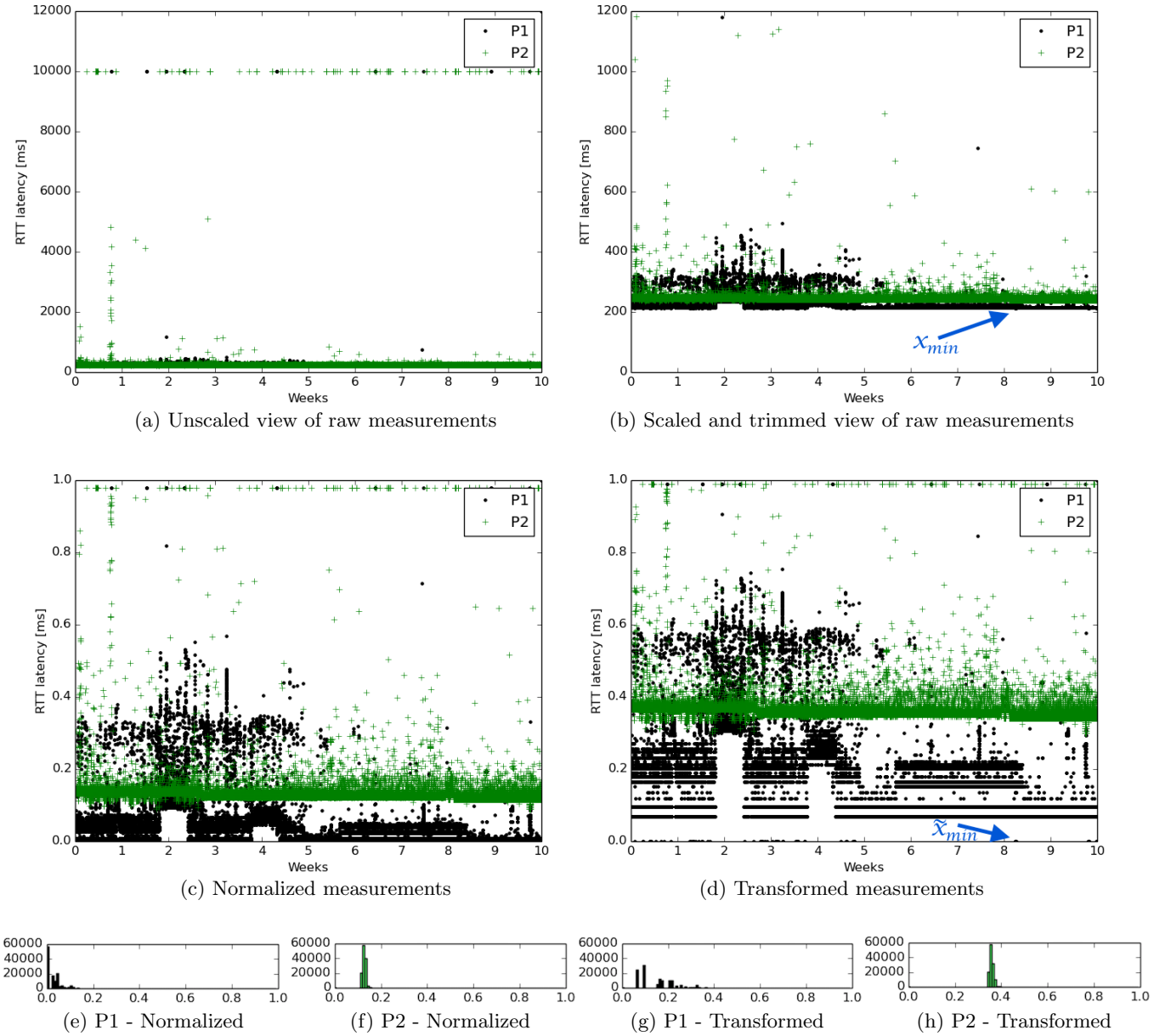
5. MEASUREMENTS PREPROCESSING

Recorded latency values (Figs. 2a and 2b) largely correspond to the geographical distances between the locations, resulting in varying scale that makes any direct comparisons impossible. Thus, we normalize the values to a new standard range (Fig. 2c) using minimum RTT, i.e. a minimum recorded latency value across all providers:

$$x_{min}^{l,v,f,b} = \min_{m,p}\{x_m^{l,v,f,b,p}\},\ m \in [1, 2, \ldots, |X|]\,, p \in P$$

Minimum–latency–based normalization results in highly–clustered measurements with different positions of central points across CSPs, locations and even protocol layers. However, for the purpose of the subsequent benchmarking calculations, the values need to be spread out such that big latency values represent one end of the interval and small values the other end. The square root transformation has the desired properties [16], as shown on transformation outcome in Fig. 2d. Such a transformation changes the distribution of measurements, but, when applied consistently, it does not affect validity of the methodology. The entire preprocessing formula is as follows:

$$\widetilde{x}_m^{l,v,f,b,p} = \sqrt{1 - \frac{x_{min}^{l,v,f,b}}{x_m^{l,v,f,b,p}}},\ m \in [1, 2, \ldots, |X|]$$



(a) Unscaled view of raw measurements

(b) Scaled and trimmed view of raw measurements

(c) Normalized measurements

(d) Transformed measurements

(e) P1 - Normalized

(f) P2 - Normalized

(g) P1 - Transformed

(h) P2 - Transformed

Figure 2: ***Latency measurements preprocessing for Latency–based benchmarking.*** **Values get normalized using minimal recorded value x_{min} and transformed using square root, as shown in respective steps (a-d) and histograms (e-h). This example shows RTTs of SQL request–response interactions between Dublin DC front–end server and Tokyo DC back–end server of both CSPs (P1 and P2), as recorded over 10 weeks.**

6. BENCHMARKING METHODOLOGY

The task of the benchmark is to report how well different systems perform under the given constraints. In practice, benchmarks are used to guide decisions about the most economical provisioning strategy as well as to gain insights into performance bottlenecks [9]. The specific need for benchmarking is given by concrete use cases such as financial trading or gaming systems. Whereas the former mandates very low latency with no tails, the latter allows for jitter and latency values below a certain psychological threshold. Such requirements are expressed via metrics, which are used throughout the benchmarking process (summarized in Figure 3).

A measurement source $s \in S$ and a destination $d \in D$ can both be any of the v, f, b (i.e. benchmarking works both upstream and downstream), but not all combinations are valid (as noted in § 3). For simplicity, we assume in the remainder of the paper that d is a CSP DC (i.e. f or b).

6.1 Selection of metrics

Given the transformed measurements with desired characteristics (§ 5), a metric value can be calculated. A metric $m \in M$ expresses a latency–related application requirement

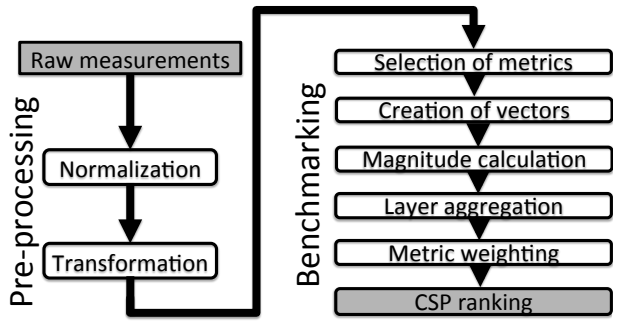


Figure 3: ***End–to–end process of transforming raw measurements into Cloud Service provider rank.***

such as low or stable latency. Arbitrary set of metrics M' can be used that reflects application requirements well, like descriptive–statistics one [11] used throughout this paper. The following example shows the Sample arithmetic mean $\overline{X} \in M'$, used to express requirements related to the latency average:

$$\overline{X}^{l,v,f,b,p} = \frac{1}{m} \sum_{m=1}^{|X|} \widetilde{x}_m^{l,v,f,b,p}$$

6.2 Metric vectors

Given $|M'|$ metrics of interest, the next step is a creation of $|M'| \cdot |P| \cdot |L|$ vectors $\vec{v}$ in a $|S|$-dimensional space, where $|S|$ is a number of measurement sources s (Fig. 4 provides an example).

Vector component is a metric value, calculated per source location s_i for provider p's destination d using protocol l:

$$\vec{v}_{\overline{X}}^{l,d,p} = (\overline{X}^{l,d,p,s_1}, \overline{X}^{l,d,p,s_2}, .., \overline{X}^{l,d,p,s_{|S|}})$$

Here we calculated a vector consisting of $|S|$ Sample arithmetic means of latency of protocol l between $|S|$ sources and provider p's destination d.

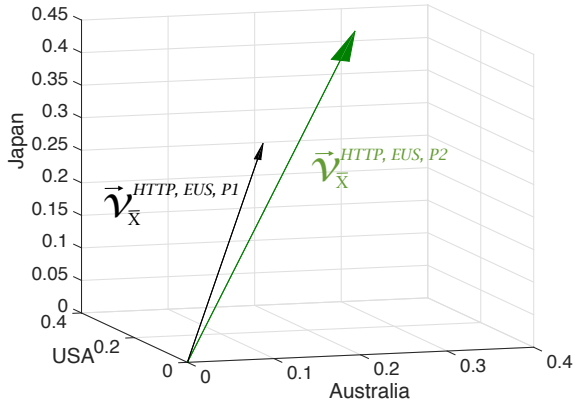


Figure 4: ***Example metric vectors of providers P1 and P2 in a 3-dimensional space.*** **Sample arithmetic means of HTTP latency measurements of similarly–located datacenter** EUS **from 3 vantage points (Australia, USA, Japan) were used. The magnitude of P1's vector is smaller and, as per our methodology, benchmarking favors P1 in the subsequent comparisons.**

A magnitude of such vector, calculated as Euclidean norm, summarizes the performance under the actual metric over all measurement source locations S:

$$\|\vec{v}_{\overline{X}}^{l,d,p}\| = \sqrt{(\overline{X}^{l,d,p,s_1})^2 + (\overline{X}^{l,d,p,s_2})^2 + .. + (\overline{X}^{l,d,p,s_{|S|}})^2}$$

The metrics, under which CSP performs well, have a vector magnitude close to 0. This also occurs in a case of co-located resources, where latency often approaches 0 ms (e.g. front-end and back-end in the same DC). Thus, co-located deployment usually dominates the comparison, which is desirable.

Note that an outstanding source does not influence the resulting CSP ranking, as it hurts all CSPs the same way. Thus, there is no need to limit the set of considered sources only to clients of target application's interest.

6.3 Protocol layer aggregation

The benchmarked application is usually built on top of the standard network protocol stack. Different protocol layers take turns in issuing round trips between application endpoints. As such, every involved protocol $l \in L'$ is essential and needs to perform well. To reflect this, we aggregate magnitudes of the per–protocol–layer vectors using the following multiplication, which ensures that all involved protocols perform well.

$$\|\vec{v}_m^{d,p}\| = \prod_{l \in L'} \|\vec{v}_m^{l,d,p}\|$$

This approach can be extended to prioritize or penalize some layers, which certain use cases may require.

6.4 Metric weighting

Depending on tenant and application needs, some metrics are of a higher priority than other. Weight $w_i \in \langle 1, MAX \rangle$ is proportional to the metric's influence on application performance - i.e. weight 1 is assigned to a metric that stands for application requirement, which, if not satisfied, does not impact performance significantly; and weight MAX is assigned to a metric having a strong impact. Given metrics of interest M', the $|M'|$ weights $w_i \in \langle 1; MAX \rangle, i \in [1, 2, \ldots, |M'|]$ are assigned and normalized as follows:

$$\widehat{w}_i = \frac{w_i}{\sum_{i=1}^{|M'|} w_i}$$

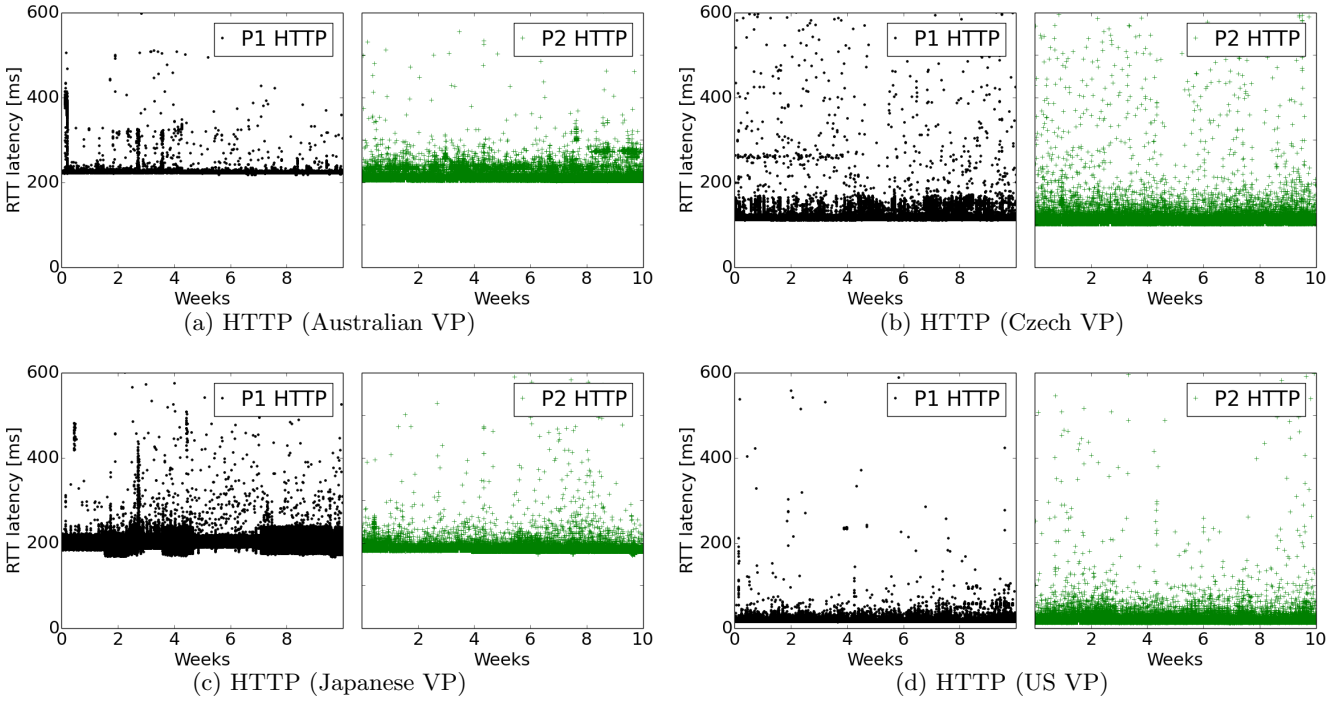
6.5 CSP ranking

CSPs are ranked by a following weighted sum of vector magnitudes:

$$f^{d,p} = \widehat{w}_1 \|\vec{v}_{m1}^{d,p}\| + \widehat{w}_2 \|\vec{v}_{m2}^{d,p}\| + \cdots + \widehat{w}_{|M'|} \|\vec{v}_{m|M'|}^{d,p}\|$$

The recommended CSPs are then the ones ranked low:

$$f^{d,p_1} \leq f^{d,p_2} \leq .. \leq f^{d,p_p}$$



(a) HTTP (Australian VP)
(b) HTTP (Czech VP)
(c) HTTP (Japanese VP)
(d) HTTP (US VP)

Figure 5: ***Case study input data - HTTP RTT latencies to Virginia DC as observed by four VPs.*** **Provider P2's DC displays a lower median latency and provider P1's DC a more–stable latency, according to aggregated observations of all VPs. These observations are quantified in various stages of benchmarking process.**

7. CASE STUDY

7.1 Benchmark calculation example

A hypothetical tenant wants to migrate an application to public Cloud environment. The application is a latency–sensitive TCP/HTTP web container serving only static web pages (previously used in [13]). The tenant considers the public CSPs P1 and P2 for deployment on the US East Coast. Nature of the application requires low-to-moderate latency and preferably a stable latency in a sense of the following weights: $median = 5 = MAX$, $standard\ deviation = 1$, $coefficient\ of\ variation = 1$. Comparable provider prices are assumed. We use the dataset from § 4, Fig. 5 shows the input HTTP measurements (TCP measurements are alike).

After normalizing and transforming measurements, benchmarking proceeds according to the steps described in § 6. Tab. 1 shows magnitudes of the selected metric vectors. P2 has >30% lower median metric–vector magnitude for both involved protocols, suggesting that it provides lower latency across the entire protocol stack. In the case of both standard deviation and coefficient of variance, P1 scored notably better and, as such, provides more stable latency. We can visualize the particular comparisons as two points in a 4-dimensional vector space, analogically to Fig. 4.

P	L	med	std	cov
P1	HTTP	0.771	0.122	0.302
	TCP	0.997	0.092	0.230
P2	HTTP	0.518	0.195	0.774
	TCP	0.726	0.130	0.836

Table 1: Calculated metric vector magnitudes $\|\vec{v}\|$

Next, we aggregate both layers via multiplication (§ 6.3) and calculate the weighted sum using the following weights:

$$\widehat{w}_1 = \frac{5}{7}, \widehat{w}_2 = \frac{1}{7}, \widehat{w}_3 = \frac{1}{7}$$

The weights reflect the primary need for low latency and the secondary need for stable latency. Weights are plugged into the weighted sum formula:

$$f^{EUS,p} = \widehat{w}_1 \|\vec{v}_{med}^{\,EUS,p}\| + \widehat{w}_2 \|\vec{v}_{std}^{\,EUS,p}\| + \widehat{w}_3 \|\vec{v}_{cov}^{\,EUS,p}\|$$

Tab. 2 shows, that CSP P2 is a recommended provider for hosting the latency-sensitive web application in East US region. Importantly, CSP P2 had a sufficiently lower latency and thus scored better overall, despite the more stable–latency environment at CSP P1. In Fig. 5, a lower median latency at P2 can be observed through order of tens milliseconds of difference at all VPs. Higher latency–stability at P1 is caused mainly by Australian and Czech VP's observations at both layers. In contrast, Japanese VP experienced a higher–stability with P2. US VP's observations were least significant, owing to a physical proximity of this VP to the DCs under consideration.

The amount of measurements needed for accurate CSP benchmarking depends on a particular CSP's stability. In the case of the two major CSPs given our timeframe, the error in resulting CSP ranks was $\leq 4.1\%$ at 3 weeks and $\leq 2.2\%$ at 6 weeks of measurements.

P	f^{EUS}
P1	0.56
P2	0.36

Table 2: Calculated CSP rank, recommending P2

7.2 Overall results

By calculating vector magnitudes for multiple representative metrics capturing latency stability (e.g. Standard deviation, Variance, Coefficient of variation) and latency amount (e.g. Mean, Median) across our 10–week dataset (§ 4), we can give an overview of the latency behavior of major public–CSP DCs:

The biggest differences in both mean and median front–end latency between the providers were observed at US datacenters. The highest latency variability and deviations were observed at Singapore DC. P2 also has an excess variance in HTTP latency at Virginia DC.

In the case of inter–DC network, big differences in latency deviation and relative variance were observed when connecting to Dublin DC (TCP and SQL latency) and Singapore DC (TCP latency). At Bay Area DCs of both providers, there was a big disagreement between SQL and TCP average latency and also the latency variability.

There are many more uses for the calculated vector magnitudes, like assigning arbitrary weights to metrics and observing changing benchmark results or benchmarking an application distributed across multiple DCs and locating clients that such a deployment would serve best.

8. CONCLUSION

In this work we have presented a methodology for, and a practical example of, Cloud-Service benchmarking, using Cloud-service latency measurements across multiple dimensions. To our best knowledge, no similar longitudinal latency–based benchmarking methods exist at the moment, unfortunately preventing a comparison to other solutions.

This practical study clearly shows that selecting the best fitting Cloud Service Provider is not straightforward, and detailed application quality-of-service requirements must be known to obtain a clear picture, as different services outperform others in different aspects or locations.

The possible applicability of such benchmarking methodology is widespread: selecting a service provider, monitoring its performance, determining a workload split among providers, possibly including real-time adjustments, or serving as input to dynamic auctioning or pricing of Cloud Services.

Clearly, improvement of the methodology is possible - for example by integrating other measurable parameters such as throughput, computing or storage performance, or by evaluating more sophisticated applications. Furthermore, despite the practical application demonstrated, a more elaborate test, considering price differences and evaluating the benefit of having selected a particular deployment based on a recommendation by the benchmarking suite, would be needed to fully confirm applicability of this methodology. We gladly invite the Cloud-computing research community to use our publicly available measurements data at **http://claudit.feld.cvut.cz** to verify some of these findings or to engage in improving the benchmarking methodology themselves. As a future work, we plan on including price per performance unit in our considerations.

Acknowledgments

This work was supported by the Grant Agency of the Czech Technical University in Prague,
grant No. SGS15/153/OHK3/2T/13.

9. REFERENCES

[1] AWS Service Health Dashboard. http://status.aws.amazon.com/.
[2] CLAudit. http://claudit.feld.cvut.cz.
[3] Cloud Harmony. http://cloudharmony.com/.
[4] Microsoft Azure status. http://status.azure.com.
[5] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. Towards Predictable Datacenter Networks. In *ACM SIGCOMM Computer Communication Review*, volume 41. ACM, 2011.
[6] M. B. Chhetri, S. Chichin, Q. B. Vo, and R. Kowalczyk. Smart CloudBench - Automated Performance Benchmarking of the Cloud. *IEEE CLOUD2013*, 2013.
[7] M. B. Chhetri, Q. B. Vo, R. Kowalczyk, and C. L. Do. Cloud Broker: Helping You Buy Better. In *International Conference on Web Information Systems Engineering*. Springer, 2011.
[8] M. Dhawan, J. Samuel, R. Teixeira, C. Kreibich, M. Allman, N. Weaver, and V. Paxson. Fathom: A Browser-based Network Measurement Platform. In *ACM IMC*, 2012.
[9] E. Folkerts, A. Alexandrov, K. Sachs, A. Iosup, V. Markl, and C. Tosun. Benchmarking in the Cloud: What it Should, Can, and Cannot be. In *Technology Conference on Performance Evaluation and Benchmarking*, pages 173–188. Springer, 2012.
[10] A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica. Above The Clouds: A Berkeley View of Cloud Computing. *UCB/EECS*, 2009.
[11] R. Jain. *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, 1991.
[12] A. Lenk, M. Menzel, J. Lipsky, S. Tai, and P. Offermann. What Are You Paying For? Performance Benchmarking for Infrastructure-as-a-Service Offerings. In *Cloud Computing (CLOUD), 2011 IEEE International Conference*, pages 484–491. IEEE, 2011.
[13] A. Li, X. Yang, S. Kandula, and M. Zhang. CloudCmp: Comparing Public Cloud Providers. *ACM IMC*, 2010.
[14] M. Menzel and R. Ranjan. CloudGenius: Decision Support for Web Server Cloud Migration. In *Proceedings of the 21st international conference on World Wide Web*, pages 979–988. ACM, 2012.
[15] R. Minnear. *Latency: The Achilles Heel of Cloud Computing*, 2011. Cloud Computing Journal.
[16] J. Osborne. Notes on the Use of Data Transformations. *Practical Assessment, Research and Evaluation*, 9(1):42–50, 2005.
[17] O. Tomanek and L. Kencl. Claudit: Planetary-Scale Cloud Latency Auditing Platform. In *Cloud Networking (CloudNet)*. IEEE, 2013.
[18] O. Tomanek, P. Mulinka, and L. Kencl. Multidimensional Cloud Latency Monitoring and Evaluation. *Computer Networks*, 2016.
[19] O. Vondrous, P. Macejko, and Z. Kocur. FlowPing-The New Tool for Throughput and Stress Testing. *Advances in Electrical and Electronic Engineering*, 13(5):516, 2015.