Exploring Uncertainty of Delays as a Factor in End-to-End Cloud Response Time

Anatoliy Gorbenko, Vyacheslav Kharchenko,
Seyran Mamutov, Olga Tarasyuk
Department of Computer Systems and Networks
National Aerospace University
Kharkiv, Ukraine
{A.Gorbenko, S.Mamutov, O.Tarasyuk}
@csac.khai.edu, V.Kharchenko@khai.edu

Alexander Romanovsky
School of Computing Science
Newcastle University
Newcastle-upon-Tyne, UK
Alexander.Romanovsky@newcastle.ac.uk

***Abstract*—This paper reports our practical experience in benchmarking a cloud-based web-service and investigates instability of its performance and the delays induced by the communication medium when measured from multiply client locations. We compare the performance of MS Azure, GoGrid and an in-house server running the same benchmark web service and analyse how the client and service implementation technologies affect this performance. The uncertainty discovered in the network delay reduces the overall performance and dependability of cloud computing provisioning and requires specific resilience techniques.**

Keywords – cloud computing; web services; performance; response time; benchmarking, dependability

I. INTRODUCTION

Services provisioning, including *software as a service* (SaaS), *Platform as a Service* (PaaS), *Infrastructure as a Service* (IaaS) and, more generally, *cloud computing*, is becoming an important element of IT for any industry, organisation and individual [1]. Cloud computing is an emergent but still not fully-understood technology, supporting the pay-as-you-go paradigm for delivering computing as a service. One of the main stumbling blocks in making service provisioning ubiquitous is the potential lack of dependability of the services, and the ability of customers to justifiably trust in the claimed performance and dependability of services. Most CFOs of the mid-sized UK firms are still unwilling to place their IT infrastructure in the cloud because of fears over data security, downtime and loss of control [2]. They are also still not fully aware of the benefits offered by cloud computing. Choosing the right cloud provider can be a very complicated for customers facing a variety of cloud technologies, providers, datacenter locations, computational options, SLAs and pricing policies.

This is why independent measurement of performance and other cloud quality characteristics is important and can help make these choices easier for customers. For example, according to the recent CloudHarm [1] IOP (Input/Output Performance) benchmarking report, the GoGrid 4GB RAM computing instance coming at a far lower cost was over 50% faster than the equivalent virtual server at Amazon. However, the results provided do not describe cloud performance from the client's point of view, taking into account the fact that clients perceive the combined performance of the media and the service running on a particular cloud computing instance as the service performance.

Invoking services from clouds may be more effective on average, but less predictable and more uncertain in particular for both service clients and providers. This uncertainty can affect usability, performance and dependability of services deployed in clouds. We have previously shown [3] that significant *uncertainty* of response time exists in service-oriented systems invoked over the Internet and that failures occur regularly. This uncertainty exhibits itself through the unpredictable response times of Internet messages and data transfers, the difficulty to diagnose the root cause of service failures, the inability to see beyond the interfaces of a service, unknown common mode failures, etc.

For example, our earlier extensive experiments with bioinformatics web services, like BASIS and BLAST [3, 4] show that the response time varies a lot because of various unpredictable factors like Internet congestions and failures, service overloads, etc. In particular, the BASIS WS response time changes from 300 ms to 120000 ms, 22% of the requests have the response time at least twice greater than the observed minimal value and 3% of requests have the response time 20 times larger. We believe it is impossible to build fast and dependable SOA without dealing with such phenomena.

In this paper we focus on measuring the uncertainty of two main delays contributing to the end-to-end response time of the benchmark service deployed in clouds: (i) network delay and (ii) cloud processing time. Besides, we are interested in understanding whether the performance of cloud-based web service is more certain than the one of in-house web services. The paper reports results of performance benchmarking for the PaaS Microsoft Azure cloud platform and their comparison with the IaaS GoGrid cloud provider and with an in-house server of similar hardware characteristics. We also investigated how the service- and client-side implementation technologies affect performance and its uncertainty.

[1] www.cloudharmony.com

Benchmarking performance and robustness under uncertainty of cloud services will provide the internal and external users with a support to help them to make informed decisions to either run their applications on company's own site or to use clouds or a combination of both.

There have been several studies on benchmarking and experimental measurements of performance, security and dependability of SOA, web services and clouds (e.g. [5-8]). These studies and the studies conducted in the CloudSleuth (www.cloudsleuth.net) and CloudHarmony projects (cloudharmony.com) aim at analysing cloud performance by measuring the end-to-end response time for various cloud providers and locations. Even though this work is important for measuring performance of various cloud providers, it neither addresses the *uncertainty challenge* nor allows distinguishing between different types of delays contributing to the overall response time.

The rest of the paper is organized as follows. In the next section we briefly describe the benchmarking technique used and provide details of the experimental settings. Section III presents the results of benchmarking delays contributing to the end-to-end cloud response time, compares the performance of MS Azure, GoGrid and an in-house server running the same benchmark web service and also analyses whether the client and service implementation technologies affect the performance. Finally, some practical lessons learnt from our experimental work are summarized in section IV.

II. Method and Experimental Settings

Our research focus is on investigating how the Internet affects the performance of the cloud-based services from the user's perspective. There are significant differences between the measurement techniques used in the work we are reporting now and the techniques discussed in Section 1. The main one is that in our experiments for each request we recorded four time-stamps: T1, T2, T3 and T4 (see Fig. 1), instead of recording only T1 and T4 that are typically measured (e.g. in [9, 10]). T1 and T2 are the times when a user sends its request to the benchmark web service and when he receives a response. T2 and T3 are the times when a user's request arrives at the benchmark web service and when a response is sent back. This allowed us to separately measure the two main delays contributing to the end-to-end response time (RT): the request processing time (RPT) by a benchmark web service deployed in cloud and the network (the Internet delay) round trip time (RTT), i.e. RT = RPT + RTT. As a benchmark to be deployed in cloud and on a in-house server we used a web service sorting a reverse-sorted three-dimensional array (NxNxN) of integers and returning back to client data of M Kbytes. In our experiments N was set up to 50 and M was set up to 100.

A Java-based application called Web Services Dependability Assessment Tool (WSsDAT) which is aimed at evaluating the performance and dependability of Web Services [11] was used to test our cloud benchmark from remote hosts. The tool supports various methods of performance and dependability testing by acting as a client invoking the remote services by its URI.

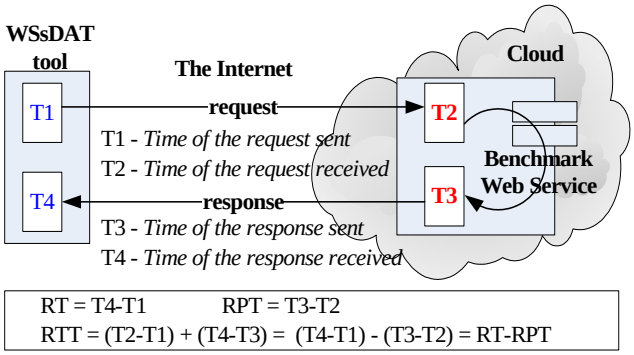

Figure 1. Response time measurement technique.

It enables the users to monitor a remote application by collecting the following reliability characteristics: (i) availability; (ii) response time; (iii) faults and exceptions.

To ensure a comprehensive assessment our tests were run from 17 end-user locations in USA, Canada and the UK and were conducted every minute during one week. The precision of the timing observations was one millisecond.

III. Benchmarking Results

A. Analysis of Delays Contributing to the End-to-End Response Time

The summary of statistical data analysis of response time and its contributing delays is presented in the Tables I-III. It includes client locations and IP addresses (the last octet of each IP address was hidden due to privacy policy), minimal, average and maximal values of the delays and also a standard deviation. A coefficient of variation (CV) that is a ratio between the delay standard deviation and its average value is taken as the measure of uncertainty.

An average request processing time (RPT) of the benchmark web service aggregated by all clients was about 705 ms.

TABLE I. MS Azure End-to-End Response Time

Client's location		Client's IP address	Response Time (RT), ms				
			Min	Avg	Max	Std Dev	CV, %
Canada	Ottawa	216.151.172.x	1484	2007	3917	255	12.7
	Burnaby	64.151.226.x	1841	2329	287448	7648	328.4
UK	Newcastle	10.8.146.x	859	1577	4016	480	30.4
	Newcastle	10.8.151.x	890	1498	3438	423	28.2
	Durham	213.175.197.x	814	1922	47740	2132	110.9
USA	New York	69.72.183.x	1331	2018	7145	575	28.5
	Chicago	209.188.85.x	1133	2104	11729	875	41.6
	Peyton	64.64.0.x	1289	1961	4637	350	17.9
	Lansing	67.225.254.x	902	2051	4483	396	19.3
		67.225.254.x	1421	2024	3992	365	18.0
		67.227.193.x	1441	2068	19365	1004	48.5
		67.227.216.x	6123	6911	11520	366	5.3
		67.227.216.x	1420	2016	3770	344	17.1
	Secaucus	204.14.93.x	1262	1983	47056	1499	75.6
		208.87.24.x	1216	1991	46457	1271	63.8
		208.87.25.x	1212	2030	59475	2232	110.0
		64.20.37.x	1216	2275	132498	5135	225.7

TABLE II. MS AZURE REQUEST PROCESSING TIME

Client's location		Client's IP address	Request Processing Time (RPT), ms				
			Min	Avg	Max	Std Dev	CV, %
Canada	Ottawa	216.151.172.x	610	733	1140	92.35	12.6
	Burnaby	64.151.226.x	609	805	1124	77.47	9.6
UK	Newcastle	10.8.146.x	609	681	969	63.05	9.3
	Newcastle	10.8.151.x	609	764	1063	98.91	13.0
	Durham	213.175.197.x	609	679	1016	75.65	11.1
USA	New York	69.72.183.x	609	679	1234	71.68	10.6
	Chicago	209.188.85.x	609	677	1000	67.65	10.0
	Peyton	64.64.0.x	610	675	1000	66.4	9.8
	Lansing	67.225.254.x	609	677	1047	69.85	10.3
		67.225.254.x	609	685	1031	76.08	11.1
		67.227.193.x	609	705	1235	84.34	12.0
		67.227.216.x	609	782	1109	91.38	11.7
		67.227.216.x	609	707	1047	88.17	12.5
	Secaucus	204.14.93.x	624	704	1032	83.9	11.9
		208.87.24.x	609	673	1015	63.67	9.5
		208.87.25.x	609	677	1047	70.18	10.4
		64.20.37.x	609	680	953	68.36	10.0

TABLE III. MS AZURE NETWORK ROUND TRIP TIME

Client's location		Client's IP address	Network Round Trip Time (RTT), ms				
			Min	Avg	Max	Std Dev	CV, %
Canada	Ottawa	216.151.172.x	805	1274	3292	288.6	22.7
	Burnaby	64.151.226.x	1129	1524	286526	7645	501.7
UK	Newcastle	10.8.146.x	218	896	3376	501.3	56.0
	Newcastle	10.8.151.x	172	734	2813	482.9	65.8
	Durham	213.175.197.x	108	1243	46991	2137	171.9
USA	New York	69.72.183.x	680	1339	6489	582	43.5
	Chicago	209.188.85.x	398	1428	10838	869.9	60.9
	Peyton	64.64.0.x	632	1286	3996	368.4	28.7
	Lansing	67.225.254.x	230	1374	3858	411.9	30.0
		67.225.254.x	777	1339	3367	385.1	28.8
		67.227.193.x	790	1363	18505	1003	73.6
		67.227.216.x	5404	6129	10895	400.6	6.5
		67.227.216.x	768	1309	3080	372.7	28.5
	Secaucus	204.14.93.x	590	1280	46212	1504	117.5
		208.87.24.x	560	1318	45644	1271	96.4
		208.87.25.x	558	1352	58600	2229	164.8
		64.20.37.x	583	1595	131623	5130	321.6

A deviation from this value of individual RPT estimations is 4.3% in average. This is an evidence of a very concerted estimation of the Request Processing Time for all clients provided by our measurement technique.

At the same time, as it can be seen from Table I, different clients perceive the performance (end-to-end response time) of the same benchmark Web Service differently mainly due to significant differences in network delays (RTT). The network delays varied significantly among different clients and even among different requests of an individual client (see Table III). From time to time all clients have been faced with delays that were extremely high. Some of them were even more than one hundred times bigger than the average response time and one hundred times bigger than ones minimal value.

To try to understand the nature of the network delay uncertainty we traced the routes between all clients and benchmark web service deployed in Microsoft Windows Azure Data Center located in Dublin (UK). A number of intermediate routers varied from 8 (for UK clients in Durham and Newcastle) up to 20 (for Lansing clients). However, the most surprising finding was that all requests from all clients (even from those located in the UK) to Dublin Data Center were sent via Amsterdam or London entry point to Microsoft corporate subnetwork back to USA into the Microsoft Data Center in Redmond (!) and only from here they were finally routed to the Microsoft Dublin Data Center (see Fig. 2). Such strange behaviour witnesses for the general problem of optimal routing in the Internet and clouds or for special routing policies applied by Microsoft for its cloud users. Unfortunately we were not able to trace the backward route and check whether it was more optimal or not. An average coefficient of variation of the request processing time took 9.8%, of the network delay – 95.7%, of the end-to-end response time – 62.2%. Thus, in our experiments the network delay makes the major contribution to the end-to-end response time and its uncertainty.

B. *Comparision Between Performance of MS Azure, GoGrid and in-house Hostings*

Customers moving their Windows applications from an in-house server to clouds have two basic options to choose from.

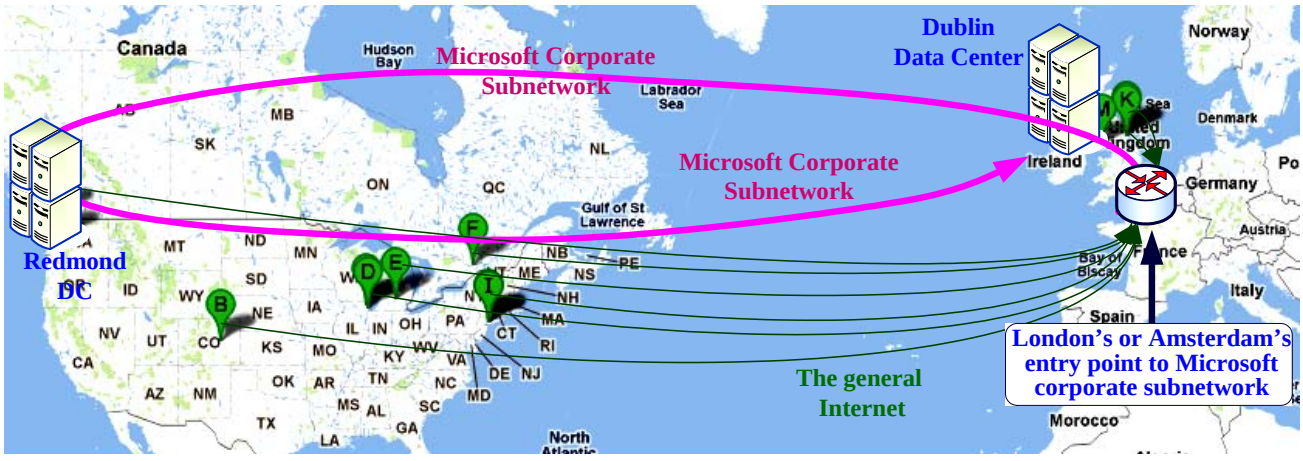

Figure 2. Routing between Windows Azure clients and Dublin Data Center.

They can either use Windows Azure that is a PaaS cloud computing service provided by Microsoft or select a Windows instance from one of numerous IaaS cloud computing services provided by Amazon, GoGrid, IBM SmartCloud, Cloud.com, etc. In our work we focused on comparing performance of initial computing instances provided by MS Azure and GoGrid cloud computing services, and an in-house service. Besides, we wanted to understand how service implementation affects its performance.

Fig. 3 shows snippets of performance trends (request processing time statistics) for our benchmark Web Service implemented using different Windows web service frameworks (WCF and ASMX) and deployed in Azure (supporting only WCF implementation), GoGrid and in-house.

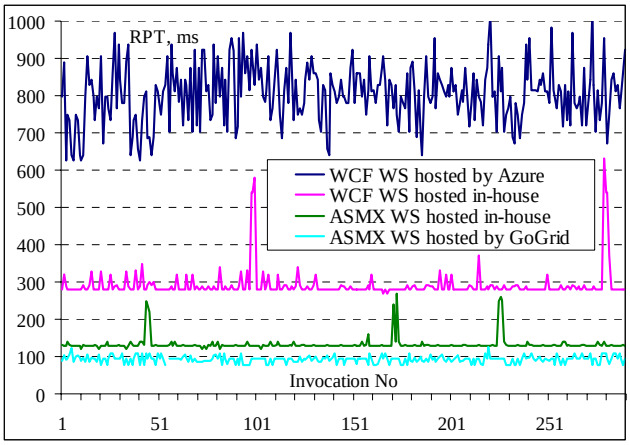

Figure 3. Request processing time statistics depending on benchmark location and implementation features.

Table IV presents characteristics of the different deployment environments used in our experiment. Azure and GoGrid use different configurations of their initial computing instances complicating their comparative analysis.

TABLE IV. DEPLOYMENT ENVIRONMENT SPECIFICATIONS

VM Instance	MS Windows Azure (Introductory Special)	GoGrid Cloud Server (512MB RAM Server)	In-house Server
Virtualisation technology	Windows Azure Hypervisor	Xen	-
CPU	1 Core 1.6 GHz	Intel Xeon E5455 1 Core 3.0 GHz	Intel Pentium T2370 1 Core 1.73 GHz
RAM	1.75 GB	512 MB	2 GB
Storage	225 GB	30 GB	60 GB
OS	Windows Azure 64-bit	Windows 2008 Server 32 bit	Windows XP SP3 32 bit
Middleware	IIS 7.0	IIS 7.0	IIS 7.0
Data center location	Europe (Dublin)	West Coast DC, San Francisco (USA)	-

Thus, Windows Azure provides larger memory capacity but less performance of the CPU. At the same time the characteristics of our in-house server were close to those provided by Azure.

Our the most surprising finding was that performance of Windows Azure which provides a native deployment environment for Windows applications was more than two times slower than performance of an equivalent in-house server (see Table V). Besides, Windows Azure has the largest standard deviation of the request processing time whereas the ASMX Web Service deployed on an in-house server has the smallest instability (i.e. the coefficient of variation). The GoGrid cloud computing service having slightly bigger instability provides performance even better than one of in-house server.

We also could see that the new Windows communication framework (i.e. WCF) degrades the performance of the ASMX Web Service by half.

Another interesting point to mention is a difference in RPT variation patterns of cloud-based Web Service and that hosted in-house. The request processing time of Windows Azure and GoGrid benchmarks varies randomly up and down from the some mean level. The request processing time of the in-house benchmarks has s shape of a line with spikes appeared from time to time. These spikes can be caused by periodic operating system processes like page swapping or garbage collection. Cloud delays seem to be less predictable because of shared hardware resources and virtualization used at the bottom software layer.

TABLE V. REQUEST PROCESSING TIME STATISTICAL SUMMARY

Benchmark WS location/technology	Min, ms	Avg, ms	Max, ms	Std.Dev, ms	CV, %
Azure/WCF	609	764	1063	98.909	13.00
in-House/WCF	270	292.38	631	41.629	14.24
in-House/ASMX	120	132.56	270	9.171	6.92
GoGrid/ASMX	78	92.839	125	9.307	10.03

C. Performance Dependence on Client Implementation Technology

The main results of our experiments were obtained by using the WSsDAT monitoring tool as a client-side application [11]. This tool is developed in Java. However, at the beginning of our work we also tried a Microsoft testbed client developed in C#. In our experiments we noticed that there is a difference between the end-to-end response times experienced by clients implemented in Java and C# while they were invoking the same cloud-based benchmark web service.

Whereas they observed pretty much the same request processing times, their network round trip times differed significantly. One-day RTT curves for Newcastle Java and C# clients are depicted on Fig, 4 and 5 respectively. They have the same shape but different variation patterns that were resulted in different forms of their probability density series (see Fig. 4b and 5b).

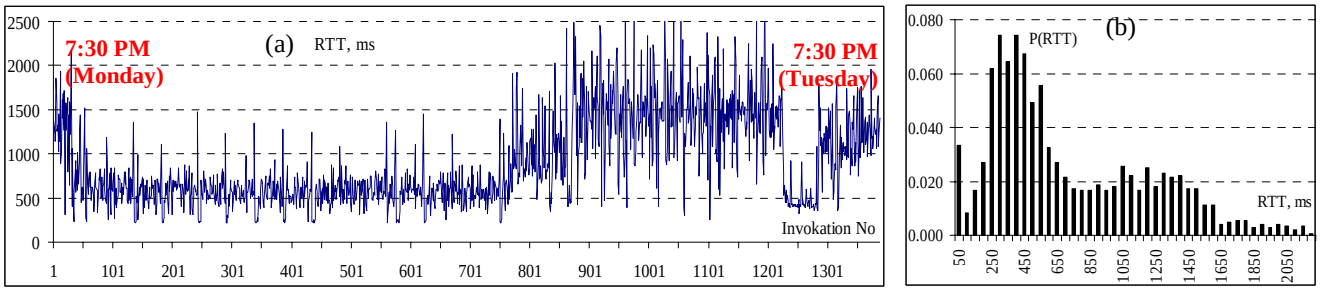

Figure 4. Network Round Trip Time statistic (a) and its probability density series (b) for Java client.

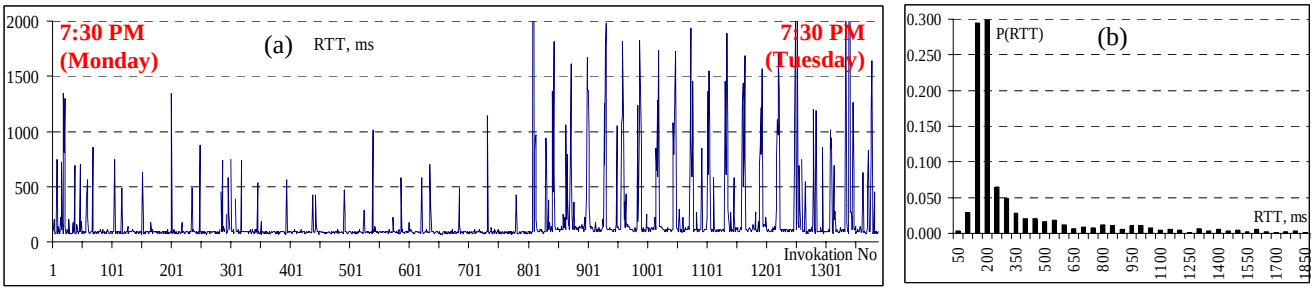

Figure 5. Network Round Trip Time statistic (a) and its probability density series (b) for Windows C# client

The general shape of RTT was caused by a hour-of-a-day dependency experienced by our clients located in Newcastle. At night (from 8 p.m. till 8 a.m.) and during rush hour (5-6 p.m.) they observed a reduced network delay that in average was more than twice as less then daily network delay At the same time the rest of our clients did not experience such clear dependency of their RTTs on hour-of-a-day.

To understand the differences in RTT variation patterns we performed a low-level analysis of network packets (their sequence and content) sent between clients and the benchmark web service. The Wireshark network packets and protocols analyzer (www.wireshark.org) was used.

It was ascertained that a Java client and the Internet Information Server (IIS 7.0) at Windows Azure implement the HTTP 1.1 specification differently. Fig. 6 shows the HTTP headers of the requests sent by Java and C# clients and the responses received from the WCF service.

It can be seen that a Java client always starts by asking an application server to keep the TCP connection alive though it is not necessary. All HTTP 1.1 connections unlike HTTP 1.0 are already considered persistent unless declared otherwise [12]. However, Microsoft IIS does not include a keep-alive header field in its response as suggested in case of receiving a keep-alive field in client's request header. As a result, the Java client finalizes the TCP connection after each invoke. Establishing a new TCP connection for every HTTP request dramatically increases an average response time and its instability.

At the same time, the Windows C# client always assumes a persistent connection that allows it to eliminate time overheads on finalizing the current TCP connections and establishing a new one. However, approximately after every thirty requests the IIS forces a C# client to finalize its persistent connection.

Java client HTTP request header:

```
POST /Service1.svc HTTP/1.1
SOAPAction: "http://tempuri.org/IService1/Calculate"
Accept: text/xml, …
Content-Type: text/xml; charset=utf-8
User-Agent: Java/1.6.0_13
Host: basisapp.cloudapp.net
Connection: keep-alive
Content-Length: 245
…
```

Microsoft C# client HTTP request header:

```
POST /Service1.svc HTTP/1.1
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://tempuri.org/IService1/Calculate"
Host: basisapp.cloudapp.net
Content-Length: 159
Expect: 100-continue
…
```

IIS 7.0 HTTP response header:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Server: Microsoft-IIS/7.0
X-Powered-By: ASP.NET
Date: Tue, 30 Nov 2010 10:06:31 GMT
Content-Length: 103520
…
```

Figure 6. HTTP headers of client requests and service responce.

IV. Conclusion and Lessons Learnt

The Internet instability significantly affects response time of services deployed in clouds. Because of network congestions and packet loses the response time could

increase in an order. Accidental and sharp increase of the response time typically occurs due to short-term network congestions causing packet losses and multiple retransmissions.

The Internet behaviour is also subjected to long-term congestions and depends on hour-of-a-day. Because of these, different clients have their own view on Web Service performance and dependability. Objective data might be obtained either by aggregating clients' experience and/or by having internal access to the Web Service operational statistics.

Network delay uncertainty can be caused by many factors which are not always evident. They include

– client's and Web Service implementation technologies and operating environment;

– Internet connections used and congestions happened;

– not optimal routes in the Internet and an internal networks of cloud computing providers.

Thus, a low-level analysis is needed to understand RTT variations. At the same time, in our experiments the cloud RPT has a much smaller variation than RTT. However, a priori it is hard to predict performance of the cloud-based web services and to choose a better cloud provider.

Benchmarking seems to be an essential means which:

– should allow the potential customers of the cloud technologies to get confidence in it and can help in choosing the cloud provider, technology and computational option;

– should help in evaluating the new and the existing technologies and to understand the bottlenecks;

– will allow the clients to evaluate their specific settings and improve them;

– could lead to extending the existing enactment engines with the advanced monitoring/prediction/fault tolerance features (in the simplest way we should be able to set the timeouts dynamically);

– will help in defining the ways to make service provisioning technologies like cloud computing, SOA and Web Services more certain, trustworthy and dependable.

Development of standard cloud performance benchmarks for different application domains (WS-, cloud-, problem-oriented) similar to TPC-W is of a great demand for both cloud computing providers and their customers.

We can also conclude that the instability of the response time depends on the quality of the network connection, the length of the network route and number of the intermediate routers. The QoS of the cloud-based Web Services cannot be ensured without guaranteeing the network QoS, especially in the case of using the Internet as a communication medium for the global service-oriented architecture.

During our experiments some of the clients caught a number of exceptions caused by various problems due to timing errors, network failures or cloud congestion. However, most of the time the root cause was really difficult to understand from the exception messages reported to client (for instance: *"Failed to read a response: javax.xml.bind.UnmarshalException - with linked exception: [javax.xml.stream.XMLStreamException: ParseError at [row,col]:[1,25896] Message: Connection reset]"*).

This work shows that from the perspectives of different clients the same Web Service deployed in the clouds and on private provider's server typically have different availability, performance and reliability characteristics. It is our strong belief that the community needs to develop advanced benchmarking techniques that allow us to (i) distinguish between different delays contributing to the end-to-end response time, (ii) support distributed benchmarking from different locations, (iii) aggregate benchmarking results gathered by multiple clients; (iv) capture short- and long-time performance trends, its uncertainty and failure statistical laws.

ACKNOWLEDGMENT

Alexander Romanovsky is partially supported by TrAmS-2 and DEPLOY projects.

REFERENCES

[1] R. Buyya, J. Broberg, A. Goscinski (Eds.), Cloud computing: principles and paradigms. John Wiley & Sons, 2011.

[2] A. Savvas, "UK CFOs fear the cloud," Computerworld UK, 2010, http://www.networkworld.com/news/2010/120410-uk-cfos-fear-the.html.

[3] Y. Chen, A. Romanovsky, A. Gorbenko, V. Kharchenko, S. Mamutov, O. Tarasyuk, "Benchmarking Dependability of a System Biology Application," Proc. 14th IEEE Int. Conference on Engineering of Complex Computer Systems (ICECCS 09), 2009, pp. 146 – 153.

[4] A. Gorbenko, V. Kharchenko, O. Tarasyuk, Y. Chen, A. Romanovsky, "The Threat of Uncertainty in Service-Oriented Architecture," Proc. Int. Workshop on Software Engineering for Resilient Systems, ACM DL, 2008.

[5] J. Duraes, M. Vieira, and H. Madeira. "Dependability Benchmarking of Web-Servers," in M. Heisel et al. (Eds.): SAFECOMP'2004, LNCS 3219, 2004, pp. 297–310.

[6] M. Vieira, N. Laranjeiro, H. Madeira. "Assessing Robustness of Web-services Infrastructures," Proc. Int. Conf. On Dependable Systems and Networks (DSN'2007), 2007.

[7] A. Li, X. Yang, S. Kandula, M. Zhang, "CloudCmp: Comparing Public Cloud Providers," Proc. 10th Conference on Internet Measurement (IMC 10), 2010, pp. 1-14.

[8] W. Sobel, S. Subramanyam, A. Sucharitakul, J. Nguyen, H. Wong, S. Patil, A. Fox, D. Patterson, "Cloudstone: Multi-Platform, Multi-Language Benchmark and Measurement Tools for Web 2.0," Proc. 1st Workshop on Cloud Computing (CCA'2008), 2008.

[9] B.F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking Cloud Serving Systems with YCSB," Proc. ACM Symposium on Cloud Computing, 2010.

[10] E. Walker, "Benchmarking amazon EC2 for high-performance scientific computing," USENIX LOGIN, Vol. 33, No. 5. 2008.

[11] P. Li, Y. Chen, A. Romanovsky. "Measuring the Dependability of Web Services for Use in e-Science Experiments," in D. Penkler, M. Reitenspiess, and F. Tam (Eds.): ISAS 2006, LNCS 4328, 2006, pp.. 193-205.

[12] RFC 2616. Hypertext Transfer Protocol - HTTP/1.1, http://tools.ietf.org/html/rfc2616