Newcastle
University

COMPUTING
SCIENCE

The Impact of Consistency on System Latency in Fault Tolerant Internet Computing

Olga Tarasyuk, Anatoliy Gorbenko, Alexander Romanovsky and
Vyacheslav Kharchenko

TECHNICAL REPORT SERIES

No. CS-TR-1459 March 2015

TECHNICAL REPORT SERIES

No. CS-TR-1459 March, 2015

The Impact of Consistency on System Latency in Fault Tolerant Internet Computing

O. Tarasyuk, A. Gorbenko, A. Romanovsky, V. Kharchenko

Abstract

The paper discusses our practical experience and theoretical results in investigating the impact of consistency on latency in distributed fault tolerant systems built over the Internet. Trade-offs between consistency, availability and latency are examined, as well as the role of application timeout that essentially determines the interplay between system availability and performance. The paper presents experimental results of measuring response time for replicated service-oriented systems that provide different consistency levels: ONE, ALL and QUORUM. These results clearly show that improvements in system consistency increase system latency. A set of novel analytical models is proposed that would enable quantified response time prediction depending on the level of consistency provided by a replicated system.

Printed and published by Newcastle University,
Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.

Bibliographical details

TARASYUK, O., GORBENKO, A., ROMANOVSKY, A., KHARCHENKO, V.
The Impact of Consistency on System Latency in Fault Tolerant Internet Computing
[By] O. Tarasyuk, A. Gorbenko, A. Romanovsky, V. Kharchenko
Newcastle upon Tyne: Newcastle University: Computing Science, 2015.

(Newcastle University, Computing Science, Technical Report Series, No. CS-TR-1459)

Added entries

NEWCASTLE UNIVERSITY
Computing Science. Technical Report Series. CS-TR-1459

Abstract

The paper discusses our practical experience and theoretical results in investigating the impact of consistency on latency in distributed fault tolerant systems built over the Internet. Trade-offs between consistency, availability and latency are examined, as well as the role of application timeout that essentially determines the interplay between system availability and performance. The paper presents experimental results of measuring response time for replicated service-oriented systems that provide different consistency levels: ONE, ALL and QUORUM. These results clearly show that improvements in system consistency increase system latency. A set of novel analytical models is proposed that would enable quantified response time prediction depending on the level of consistency provided by a replicated system.

About the authors

Olga Tarasyuk graduated in computer science in 2001 and received the PhD degree from the National Aerospace University, Kharkiv, Ukraine in 2004. She is an Associate Professor at the Department of Computer Systems and Networks of the National Aerospace University in Kharkiv (Ukraine). Her research interests span many aspects of software development, verification, dependability assessment and prediction.

Dr. Anatoliy Gorbenko received a PhD degree in Computer Science from National Aerospace University, Kharkiv, Ukraine in 2005. He is an Associate Professor at the Department of Computer Systems and Networks of the National Aerospace University in Kharkiv (Ukraine). There he co-coordinates the DESSERT (Dependable Systems, Services and Technologies) research group. His main research interests centre on assessment and ensuring dependability and fault tolerance in computer systems and service-oriented architectures. Dr. Gorbenko is a member of EASST (European Association of Software Science and Technology).

Alexander (Sascha) Romanovsky is a Professor in the Centre for Software and Reliability, Newcastle University. His main research interests are system dependability, fault tolerance, software architectures, exception handling, error recovery, system structuring and verification of fault tolerance. He received a PhD degree in Computer Science from St. Petersburg State Technical University and has worked as a visiting researcher at ABB Ltd Computer Architecture Lab Research Center, Switzerland and at Istituto di Elaborazione della Informazione, CNR, Pisa, Italy. In 1993 he became a postdoctoral fellow in Newcastle University, and worked on the ESPRIT projects on Predictable Dependable Computing Systems (PDCS), Design for Validation (DeVa) and on UK-funded projects on the Diversity, both in Safety Critical Software using Off-the-Shelf components. He was a member of the executive board of EU Dependable Systems of Systems (DSoS) Project, and between 2004 and 2012 headed projects on the development of a Rigorous Open Development Environment for Complex Systems (RODIN), and latterly was coordinator of the major FP7 Integrated Project on Industrial Deployment of System Engineering Methods Providing High Dependability and Productivity (DEPLOY). He now leads work on fault tolerance in Systems of Systems within the COMPASS project and is Principal Investigator of Newcastle's Platform Grant on Trustworthy Ambient Systems.

Prof. Vyacheslav Kharchenko is a Professor and heads of the Computer Systems and Networks Department at the National Airspace University, Ukraine. He is a member of IEEE and also a senior research investigator in the field of safety-related software at the State Science-Technical Centre of Nuclear and Radiation Safety (Ukraine). He has published nearly 200 scientific papers, reports and book chapters, more than 500 inventions and is the coauthor or editor of 28 books. Prof Kharchenko has been a head of the DESSERT International Conference (http://www.stc-dessert.com) in 2006-2010. His research interests include critical computing, dependable and safety-related I&C systems, multi-version design technologies, software and FPGA-based systems verification and expert analysis.

Suggested keywords
INTERNET COMPUTING
FAULT-TOLERANCE
CONSISTENCY
LATENCY
RESPONSE TIME
MODELLING

The Impact of Consistency on System Latency in Fault Tolerant Internet Computing

Olga Tarasyuk[1], Anatoliy Gorbenko[1], Alexander Romanovsky[2],
Vyacheslav Kharchenko[1]

[1]Department of Computer Systems and Networks
National Aerospace University, Kharkiv, Ukraine
`{O.Tarasyuk, A.Gorbenko}@csn.khai.edu, V.Kharchenko@khai.edu`
[2]School of Computing Science, Newcastle University, Newcastle upon Tyne, UK
`Alexander.Romanovsky@ncl.ac.uk`

Abstract. The paper discusses our practical experience and theoretical results in investigating the impact of consistency on latency in distributed fault tolerant systems built over the Internet. Trade-offs between consistency, availability and latency are examined, as well as the role of application timeout that essentially determines the interplay between system availability and performance. The paper presents experimental results of measuring response time for replicated service-oriented systems that provide different consistency levels: ONE, ALL and QUORUM. These results clearly show that improvements in system consistency increase system latency. A set of novel analytical models is proposed that would enable quantified response time prediction depending on the level of consistency provided by a replicated system.

Keywords. Internet computing, fault-tolerance, consistency, latency, response time, modelling

1 Introduction

Distributed computing has become an industrial trend, indispensable in dealing with enormous data growth. High availability requirements for many modern Internet applications require the use of system redundancy and data replication. However, basic fault tolerant solutions like *N*-modular, hot- and cold-spare redundancy usually assume a synchronous communication between replicas, which means that every message is delivered within a fixed and known amount of time [1]. This is a reasonable simplification for the local-area systems whose components are compactly located, for instance, within a single data centre.

However, this assumption does not appear to be relevant for the wide-area systems, in which replicas are deployed over the Internet and their updates cannot be propagated immediately, which makes it difficult to guarantee consistency.

The Internet and, more generally, the wide-area networked systems are characterized by a high level of uncertainty, which makes it hard to guarantee that a client will receive a response from the service within a finite time. We have previously shown that there is a significant uncertainty of response time in service-oriented systems invoked over the Internet [2, 3]. Besides, in our experience and as described in other studies [4–6], failures occur regularly on the Internet, clouds and in scale-out data centre networks. When developers apply replication and other fault tolerant techniques for the Internet- and cloud-based systems, they need to understand the time overheads and have to care about delays and their uncertainty. Similarly, providing consistency among replicas is a major issue in distributed fault-tolerant computing.

This paper will examine, both in experimental and theoretical terms, how different fault-tolerance solutions implemented over the Internet affect system latency depending on the level of consistency provided. Inspired by the experimental results obtained we propose analytical models that describe response time probability density functions. These models are applicable to basic fault-tolerance solutions, such as *N*-modular and hot-spare redundancy [7]. They are important for understanding the trade-offs between system consistency, availability and latency, as identified by the CAP theorem [8]. They allow systems developers to predict system response time depending on the chosen fault-tolerance technique and/or the selected consistency level.

The rest of the paper is organized as follows. In Section 2 we discuss the impact of the CAP theorem [8] on distributed fault-tolerant systems and examine the trade-offs between system consistency, availability and latency. Section 3 summarises results of experimental response time measurements for testbed fault-tolerant systems that have three replicas distributed over the Internet and support different consistency levels. The probabilistic models introduced in Section 4 define the relation between system response time and the consistency level provided. Section 5 evaluates the accuracy of the proposed analytical models by applying them in practice and comparing their results with our experimental data. Finally, some practical lessons learnt from our experimental and theoretical work are summarised in Section 6.

2 Understanding Trade-offs Between Consistency, Availability and Latency in Distributed Fault-Tolerant Systems

The CAP theorem [8], first appeared in 1998-1999, defines a trade-off between system availability, consistency and partition tolerance and states that the only two of the three properties can be preserved at once in distributed replicated systems. Gilbert and Lynch [9] consider the CAP theorem as a particular case of a more general trade-off between consistency and availability in unreliable distributed systems propagating updates eventually over time.

Partition property, system availability and latency are tightly connected. A replicated fault-tolerant system becomes partitioned when some of its part does not respond until timeout due to arbitrary message loss, delay or replica failure. System availability can be interpreted as a probability that each client request eventually receives a response. Though, in many real systems a response that is too late (i.e. beyond the application timeout) is treated as a failure. High latency is undesirable effect for many interactive web applications. In [12] the authors showed that if a response time increases even as small as 100 ms it dramatically reduces the probability that a customer will continue to use the system.

Strong consistency can be achieved if only all system replicas are available. Failing to receive responses from some of the replicas within the specified timeout causes a partition of the replicated systems. Thus, a partition can be considered as a time bound on replica's response time. When the system detects a partition it has to decide whether to return a possibly inconsistent response to a client or to reply an exception message worsening system availability. A slow network connection, slow responding replica or wrong timeout settings can falsely cause a decision that the system is partitioned.

For the distributed fault-tolerant systems the designers cannot forfeit partitions happened due to network failures, message losses, hacker attacks and components crashes and, hence, have to choose between availability and consistency. One of these two properties has to be sacrificed. If system developers intentionally decided to give up consistency they also can improve system response time by returning the fastest response to a client without waiting until timeout for other replica responses, though this policy increases a probability of providing inconsistent result. Besides, timeout settings are also important. If the timeout is less than the typical response time, a system will likely enter a partition mode more often [10].

It is important to remember that all these three properties are not binary. For example, modern distributed database systems, e.g. Cassandra [13], can provide a discrete set of different consistency levels for each particular read or write request. Response time can theoretically vary between zero and infinity. Though, in practice it is restricted from the right by the application timeout and from the left by some minimal affordable time higher than zero. The availability is measured as usual between 0% and 100%.

Nowadays, the architects of distributed database management systems and large-scale web applications like Facebook, Twitter, etc. often decide to relax consistency requirements by introducing asynchronous data updates in favour of system availability and response time. But the most promising approach is to balance these properties. For instance, the Cassandra NoSQL DDBS introduces a tunable replication factor and an adjustable consistency model so that a customer can choose a particular level of consistency with regards to the desired system latency.

The CAP theorem helps the developers to understand the system trade-offs between consistency and availability/latency [11]. Though there are no methods available that allow trading-off consistency against availability and latency in a quantitative way. Apart from the qualitative statement following from the CAP that "better consistency worsens system availability and latency" developers do not have quantitative models helping to estimate system response time corresponding to the chosen consistency level and to precisely trade-off between them.

Our interpretation of the CAP theorem and the trade-offs resulting from the CAP is depicted on Fig. 1. Application timeout can be considered as a bound between system availability and performance (in term of latency or response time) [14]. Thus, system designers should be able to set up timeouts according to the desired system response time also keeping in mind a choice between consistency and availability.

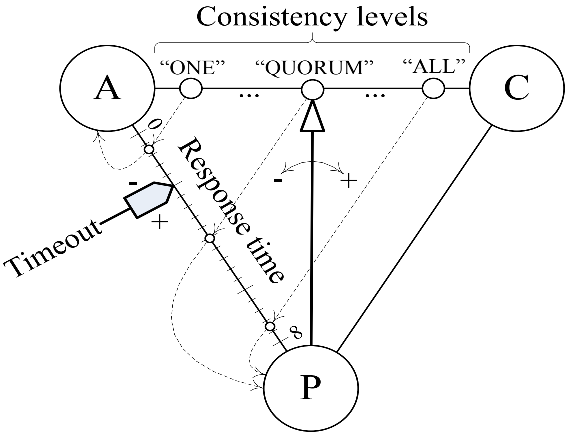


Fig. 1. The CAP trade-offs.

In the following sections we discuss our practical experience on measuring latency of fault-tolerant service-oriented system depending on the provided consistency level and also introduce analytical models predicting system response time.

3 Experimental Investigation of the CAP Impact on Fault-Tolerant Service-Oriented Systems

3.1 Description of the Testbed Architecture

To investigate the CAP impact on fault-tolerant distributed systems we have developed a testbed service-oriented system composed out of the three replicated web services (see Fig. 2).

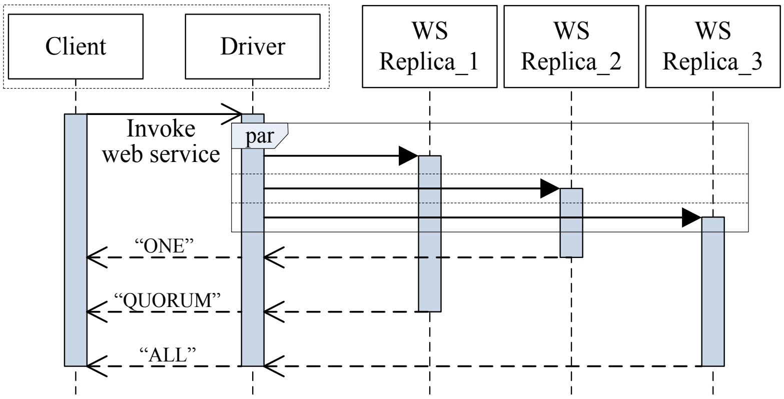


Fig. 2. Fault-tolerant service-oriented system.

A testbed web service was written in Java and its replicas uploaded to Amazon Elastic Beanstalk and were deployed in the three different location domains: (i) US West (Oregon); (ii) South America (Sao Paulo) and Asia Pacific (Tokyo). Each web service replica performs a heavy-computational arithmetic calculation such as finding the *n* digit of Pi when *n* is a large number and returns the result to the driver. The driver is responsible for invoking each of the replicated web services, waiting for the web services to complete their execution and return response, and, finally, implementing a particular fault-tolerant scheme upon the obtained results.

AWS SDK for Java was used to connect web service replicas on Amazon EC2 from clients (driver) programming code that helps to take the complexity out of coding by providing Java APIs for AWS services.

In our study we investigated the three basic fault-tolerant patterns for web services [15] corresponding to different consistency levels (ONE, ALL, QUORUM). In all cases the driver simultaneously forwards client's request to all replicated web services. The consistency level determines the number of replicas which must return a response to the driver before it sends an adjudicated result to the client application:

- ONE (*hot-spare redundancy*) – when the FASTEST response is received the driver forwards it to the client. This is the weakest consistency level though it guarantees the minimal latency;
- ALL (*N-modular redundancy*) – the driver must wait until ALL replicas return their responses. In this case the response time is constrained by the slowest replica though the strongest consistency is provided;
- QUORUM – the driver must wait for the responses from a QUORUM of replica web services. It provides a compromise between the ONE and ALL options trading off latency versus consistency. The quorum is calculated as: (*amount_of_replicas* / 2) + 1, rounded down to an integer value. As far as in our experiments we use the replication factor of 3, the quorum is 2.

The driver also implements a timeout mechanism aimed to protect clients from endless waiting in case of network or web-services failures or cloud outages.

3.2 Response Time Measurement

The driver was implemented as part of the Java client software. The client software was run at a host in the Newcastle University (UK) corporate network. It invoked replica web services several thousand times in a loop using the driver as a proxy.

For the particular client's request we measured the response time of the each web service replica and also times when the driver produces responses corresponding to different consistency levels. The delay induced by the driver itself was negligible in our experiments.

The measurement results obtained for the first 100 invocations are presented in Figs. 3 and 4. Table 1 summarizes basic statistical characteristics of the measured data whereas probability density series (*pds*) of system and replicas response times are depicted in Figs. 5 and 6.

As expected, when the system is configured to provide consistency level ONE its latency in average is less than the average response time of the fastest replica. Average system latency in case it provides consistency level ALL is larger than the average response time of the slowest replica. System latency associated with consistency level QUORUM is in the middle.

However, our main observation is that it is hardly possible to make an accurate prediction of the average system latency corresponding to the certain consistency level when the only common statistical measures of replicas response time (i.e. minimal, maximal and average values and standard deviation) are known.

This finding resulting from our massive experiments and also confirmed by other researches [16] show that it is extremely difficult to predict the timing characteristics of various types of wide-area distributed systems, including fault-tolerant SOAs, distributed databases and file systems (e.g. Cassandra, GFS, HDFS), parallel processing systems (e.g. Hadoop Map-Reduce).

In the next section we propose a probabilistic modelling approach that addresses this problem. It relies on using probability density functions (PDF) of replica response times to predict system latency at different consistency levels.

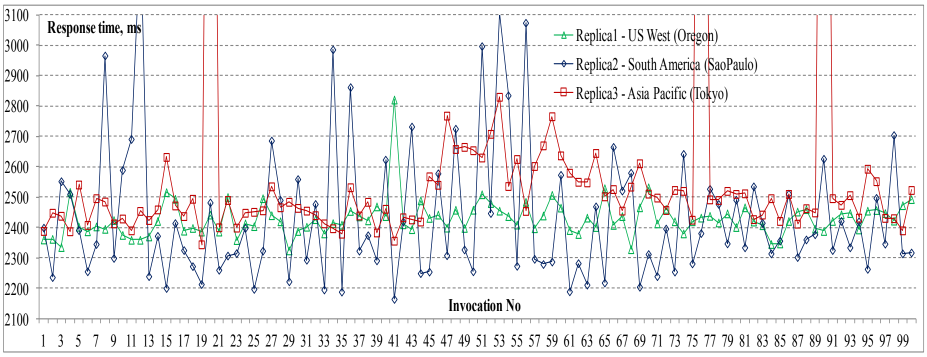


Fig. 3. Response time of different web service replicas.

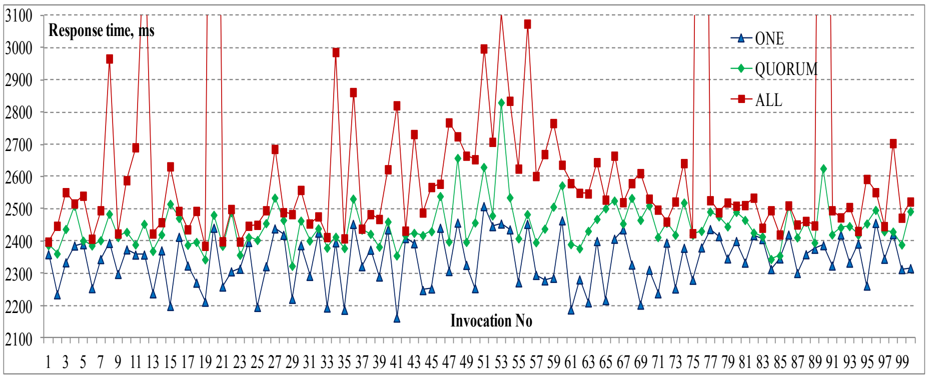


Fig. 4. System response time corresponding to different consistency levels.

Table 1. Response time statistics.

Response	Replica1	Replica2	Replica3	System consistency level		
Time, ms	(Oregon)	(Sao Paulo)	(Tokyo)	ONE	QUORUM	ALL
Minimal	2324	2164	2344	2164	2324	2386
Average	2428	2434	2588	2342	2449	2660
Maximal	2821	3371	5573	2509	2830	5573
Std. deviation	60	228	522	80	72	529

4 Probabilistic Models of System Response Time for Different Consistency Levels

We propose a set of probabilistic models that allow us to build a combined probability density function of system response time by taking into account provided consistency level and incorporating response time probability density functions for each replica.

When the system is configured to provide consistency level ALL, the probability of returning response to the client at time *t* is equal to the probability that one of the replicas (e.g. the first one) returns its response exactly at time *t*, i.e. $g_1(t)$ while two other replicas return their responses not later than *t* (by time *t*), i.e. $\int_0^t g_2(t) = G_2(t)$ and $\int_0^t g_3(t) = G_3(t)$.

So far as we have three replicas, all three possible combinations have to be accounted. As a result, the probability density function of the system response time for consistency level ALL can be defined as following:

$$f_{ALL}(t) = g_1(t)G_2(t)G_3(t) + g_2(t)G_1(t)G_3(t) + g_3(t)G_1(t)G_2(t). \tag{1}$$

where $g_1(t)$, $g_2(t)$ and $g_3(t)$ – are response time probability density functions of the first, second and third replicas respectively; $G_1(t)$, $G_2(t)$ and $G_3(t)$ – are response time cumulative distribution functions of the first, second and third replicas respectively.

When the system is configured to provide consistency level ONE, the probability of returning a response to the client at time *t* is equal to the probability that if only one of the replicas (e.g. the first one) returns its response exactly at time *t*, i.e. $g_1(t)$, while two other replicas return their responses at the same time or later on, i.e. $\int_t^\infty g_2(t) = 1 - G_2(t)$ and $\int_t^\infty g_3(t) = 1 - G_3(t)$.

Keeping in mind three possible combinations we can deduce the probability density function of the system response time for consistency level ALL as:

$$\begin{aligned} f_{ONE}(t) = {} & g_1(t)(1 - G_2(t))(1 - G_3(t)) + g_2(t)(1 - G_1(t))(1 - G_3(t)) + \\ & + g_3(t)(1 - G_1(t))(1 - G_2(t)). \end{aligned} \tag{2}$$

Deducing the response time probability density function for the QUORUM consistency level is based on a combination of the previous two cases.

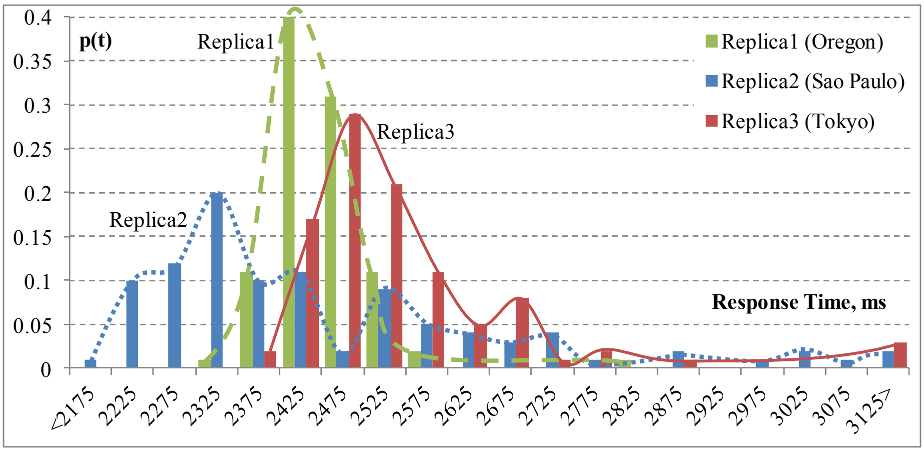


Fig. 5. Probability density series of replicas response times.

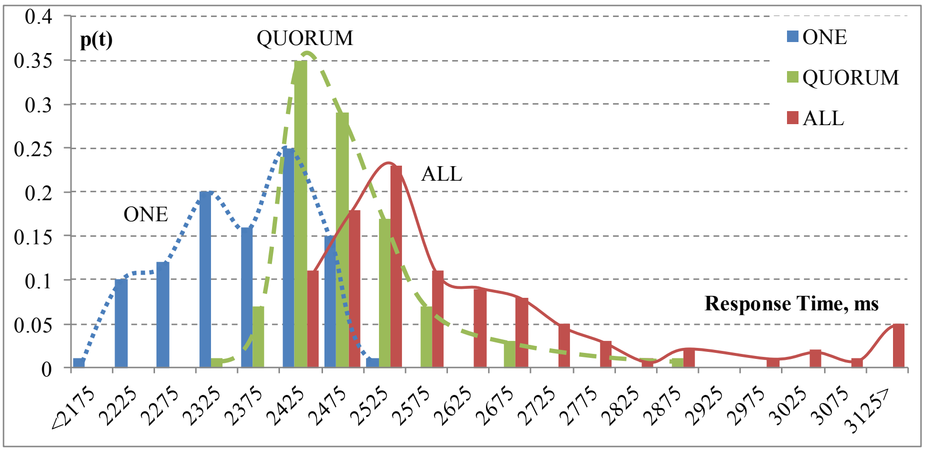


Fig. 6. Probability density series of system response time for different consistency levels.

The probability of returning response to the client at time *t* is equal to the probability that one of the replicas returns its response exactly at time *t*; one of the two remained replicas returns its response by time *t* and another one responds at time *t* or later on. Taking into account all possible combinations the probability density function of the system response time for consistency level QUORUM can be deduced as:

$$\begin{aligned} f_{QUORUM}(t) = {} & \left(g_1(t)G_2(t) + g_2(t)G_1(t)\right)\left(1 - G_3(t)\right) + \\ & + \left(g_1(t)G_3(t) + g_3(t)G_1(t)\right)\left(1 - G_2(t)\right) + \\ & + \left(g_2(t)G_3(t) + g_3(t)G_2(t)\right)\left(1 - G_1(t)\right). \end{aligned} \tag{3}$$

Using similar reasoning it is possible to deduce response time probability density functions of a system composed of n replicas:

$$f_{ALL}(t) = \sum_{i=1}^{n} \left(\frac{g_i(t)}{G_i(t)} \cdot \prod_{j=1}^{n} G_j(t) \right). \tag{4}$$

$$f_{ONE}(t) = \sum_{i=1}^{n} \left(\frac{g_i(t)}{1 - G_i(t)} \cdot \prod_{j=1}^{n} \left(1 - G_j(t)\right) \right). \tag{5}$$

It is not possible to build a general form of the probability density function of the system response time for consistency level QUORUM. However, the general reasoning is as following. The composed probability density function should be presented as a sum of m items, where m is a number of k-combinations of n (k is a number of replicas constituting a quorum). Each of the m items is a product of two factors. The first one defines the probability that a particular combination of k replicas return their responses by time t. Another factor defines the probability that the remaining (n–k) replicas return their responses after t.

5 Models Validity

In this section we check the validity and accuracy of the proposed models by comparing their prediction with the experimental data presented in Section 3. This check includes the following four steps:

- finding out theoretical distribution laws that can accurately approximate the measured replica response times;
- applying proposed mathematical models (1), (2) and (3) to deduce probability density functions of the system response time for different consistency levels;
- estimating replica and system average response times using the theoretical probability distribution functions;
- comparing the theoretical and experimental values of replica and system average response times.

5.1 Finding Theoretical Distribution Laws of Replica Response Times

Theoretical distribution laws approximating replica response times can be found in a way described in [2]. It is based on performing a series of hypotheses checks in the Matlab numeric computing environment. The techniques of hypothesis testing consist of the two basic procedures. First, the values of distribution parameters are estimated by analysing an experimental sample. Second, the null hypothesis that experimental data has a particular distribution with certain parameters should be tested.

To perform hypothesis testing itself we used the `kstest` function: `[h, p] = kstest(t, cdf)`, conducting the Kolmogorov-Smirnov test to compare the distribution of t with the hypothesized distribution defined by matrix *cdf*.

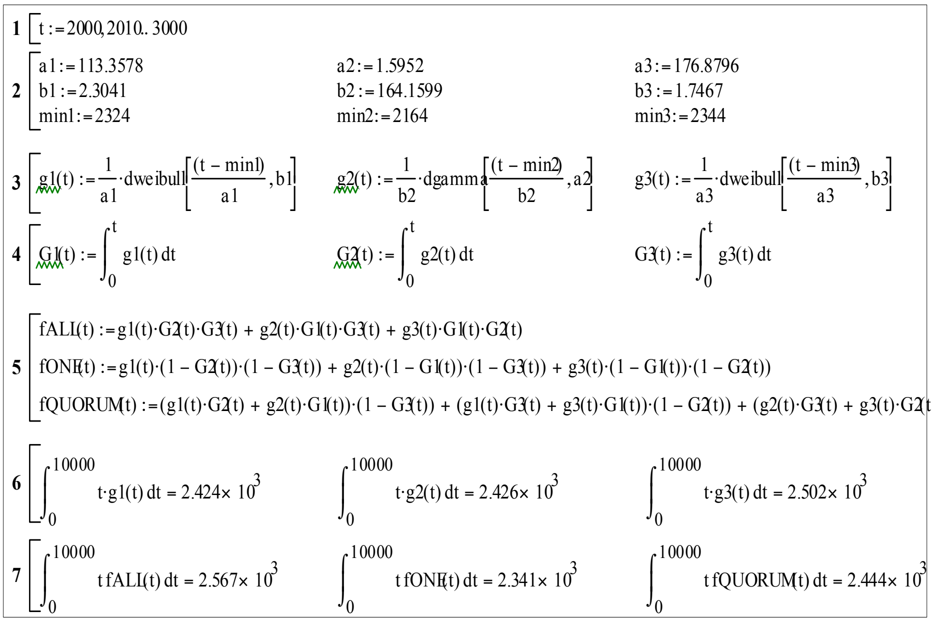
The null hypothesis for the Kolmogorov-Smirnov test is that t has a distribution defined by *cdf*. The alternative hypothesis is that x does not have that distribution. Result h is equal to '1' if we can reject the hypothesis, or '0' if we cannot. The function also returns the p-value which is the probability that x does not contradict the null hypothesis. We reject the hypothesis if the test is significant at the 5% level (if p-value is less than 0.05). The p-value returned by `kstest` was used to estimate the goodness-of-fit of the hypothesis. As a result of hypothesis testing we found out that the *Weibull* distribution fits well the response time of the first (Oregon) and the third (Tokyo) replicas. The response time of the second replica (Sao Paulo) can be accurately approximated by the *Gamma* distribution.

5.2 Deducing Probability Density Functions of the System Response Time

Mathcad has been used at the second stage of our investigation to deduce theoretical distributions of system response times for different consistency levels. It also allows to estimate average system latency and to plot probability density functions. Mathcad worksheet is shown in Fig. 7. It includes seven modelling steps.

At the 1st step we define abscissa axis t and its dimension in milliseconds. Secondly, we set up parameters of replicas response time distribution functions estimated in Matlab and also their shifts on the abscissa axis (i.e. minimal response time values).

At the 3rd and 4th steps the replica response time probability density functions $g_1(t)$, $g_2(t)$, $g_3(t)$ and the corresponding cumulative distribution functions $G_1(t)$, $G_2(t)$, $G_3(t)$ are defined using Mathcad library functions `dweibull` and `dgamma`.



1 $t := 2000, 2010 .. 3000$

2 $a1 := 113.3578$ $\quad a2 := 1.5952$ $\quad a3 := 176.8796$

$b1 := 2.3041$ $\quad b2 := 164.1599$ $\quad b3 := 1.7467$

$min1 := 2324$ $\quad min2 := 2164$ $\quad min3 := 2344$

3 $g1(t) := \frac{1}{a1} \cdot dweibull\left[\frac{(t - min1)}{a1}, b1\right]$ $\quad g2(t) := \frac{1}{b2} \cdot dgamma\left[\frac{(t - min2)}{b2}, a2\right]$ $\quad g3(t) := \frac{1}{a3} \cdot dweibull\left[\frac{(t - min3)}{a3}, b3\right]$

4 $G1(t) := \int_0^t g1(t)\,dt$ $\quad G2(t) := \int_0^t g2(t)\,dt$ $\quad G3(t) := \int_0^t g3(t)\,dt$

5 $fALL(t) := g1(t) \cdot G2(t) \cdot G3(t) + g2(t) \cdot G1(t) \cdot G3(t) + g3(t) \cdot G1(t) \cdot G2(t)$

$fONE(t) := g1(t) \cdot (1 - G2(t)) \cdot (1 - G3(t)) + g2(t) \cdot (1 - G1(t)) \cdot (1 - G3(t)) + g3(t) \cdot (1 - G1(t)) \cdot (1 - G2(t))$

$fQUORUM(t) := (g1(t) \cdot G2(t) + g2(t) \cdot G1(t)) \cdot (1 - G3(t)) + (g1(t) \cdot G3(t) + g3(t) \cdot G1(t)) \cdot (1 - G2(t)) + (g2(t) \cdot G3(t) + g3(t) \cdot G2(t$

6 $\int_0^{10000} t \cdot g1(t)\,dt = 2.424 \times 10^3$ $\quad \int_0^{10000} t \cdot g2(t)\,dt = 2.426 \times 10^3$ $\quad \int_0^{10000} t \cdot g3(t)\,dt = 2.502 \times 10^3$

7 $\int_0^{10000} t\, fALL(t)\,dt = 2.567 \times 10^3$ $\quad \int_0^{10000} t\, fONE(t)\,dt = 2.341 \times 10^3$ $\quad \int_0^{10000} t\, fQUORUM(t)\,dt = 2.444 \times 10^3$

Fig. 7. Mathcad's worksheet.

At the 5th step we define probability density functions of the system response time corresponding to different consistency levels by combining replicas *pdf* and *cdf* according to the proposed equations (1), (2) and (3). Probability distribution functions of replicas and system response times are shown in Figs. 8 and 9. The bulk of the values of probability density function $f_{ALL}(t)$ is shifted to the right on the abscissa axis as it was expected. The shapes of the $f_{ONE}(t)$ and $f_{QUORUM}(t)$ probability density functions are also in line with the reasonable expectations and experimentally obtained probability density series (see Fig. 6). Finally, at steps 6 and 7 we estimate the system and replicas average response time by integrating their theoretical probability distribution functions.

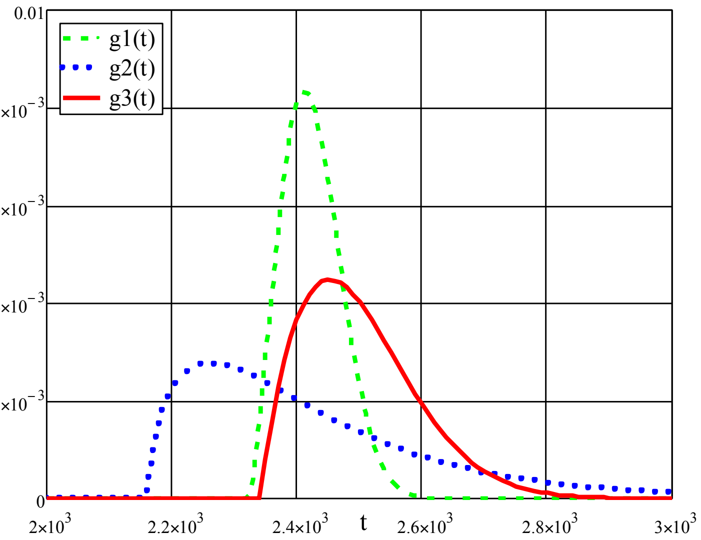


Fig. 8. Probability density functions of replicas response times.

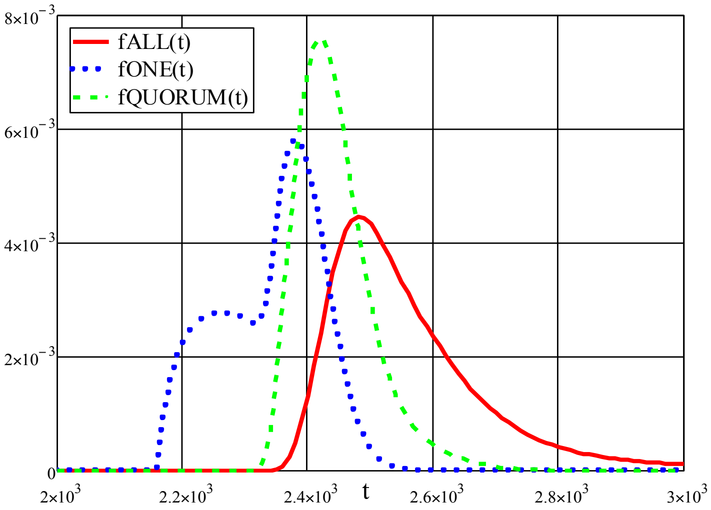


Fig. 9. Probability density functions of system response time for different consistency levels.

5.3 Accuracy of Mathematical Modelling

Table 2 shows the deviation between the average values of the system and replicas response time estimated practically (see Table 1) and theoretically with the help of the obtained probability distribution functions. These results confirm the significant closeness between actual and modelled timing characteristics. To be sure that not only the average value can be accurately predicted we compare theoretical system probability density functions (see Fig. 9) and practically obtained probability density series (Fig. 6). With this purpose we estimated experimental and theoretical probabilities that system latency at different consistency levels is less than the specified time.

Table 2. Accuracy of mathematical modelling.

	Replica1 (Oregon)	Replica2 (Sao Paulo)	Replica3 (Tokyo)	System consistency level ONE	 QUORUM	 ALL
Approximating theoretical distributions and their parameters						
distribution	Weibull	Gamma	Weibull			
alpha	113.3578	1.5952	176.8796			
beta	2.3041	164.1599	1.7467			
x-shift	2324	2164	2344			
Average response time, ms						
measured	2428	2434	2588	2342	2449	2660
modelled	2424	2426	2502	2341	2444	2567
Deviation, %	0.18	0.34	3.32	0.03	0.19	3.51

Table 3. Deviation between theoretical system *pdf* and *pds* obtained experimentally.

Time, ms	Probability that system latency is less than the specified time pds	ONE pdf	 dev.,%	 pds	QUORUM pdf	 dev.,%	 pds	ALL pdf	 dev.,%
2175	0.01	0.009	10.00	0	0	-	0	0	-
2225	0.11	0.116	5.45	0	0	-	0	0	-
2275	0.23	0.252	9.57	0	0	-	0	0	-
2325	0.43	0.385	10.47	0.01	0	-	0	0	-
2375	0.59	0.596	1.02	0.08	0.097	21.25	0	0.003	-
2425	0.84	0.858	2.14	0.43	0.434	0.93	0.11	0.073	33.64
2475	0.99	0.975	1.52	0.72	0.752	4.44	0.29	0.263	9.31
2525	1	0.998	0.20	0.89	0.903	1.46	0.52	0.476	8.46
2575	1	1	0	0.96	0.961	0.10	0.63	0.643	2.06
2625	1	1	0	0.96	0.984	2.50	0.72	0.761	5.69
2675	1	1	0	0.99	0.994	0.40	0.8	0.841	5.13
2725	1	1	0	0.99	0.998	0.81	0.85	0.892	4.94
2775	1	1	0	0.99	0.999	0.91	0.88	0.924	5.00
2825	1	1	0	0.99	1	1.01	0.89	0.945	6.18
2875	1	1	0	1	1	0	0.91	0.959	5.38
2925	1	1	0	1	1	0	0.91	0.969	6.48
2975	1	1	0	1	1	0	0.92	0.977	6.20
3025	1	1	0	1	1	0	0.94	0.982	4.47
3075	1	1	0	1	1	0	0.95	0.987	3.89
Average deviation, %			2.12			2.25			7.12

The results of this comparison (Table 3) show a close approximation of the experimental data by the proposed analytical models, especially for the consistency levels ONE and QUORUM. The probabilistic model of the system response time for consistency level ALL gives slightly optimistic prediction, though the average deviation from the experimental data is only 7% – that is close enough.

6 Conclusion and Lessons Learnt

When employing fault-tolerance techniques over the Internet and clouds, engineers should deal with delays, their uncertainty, timeouts, adjudication of asynchronous replies from replicas, and other specific issues involved in global distributed systems. The overall aim of this work was to study consistency impact on system latency in fault tolerant Internet computing.

Our experimental results clearly showed that better system consistency worsens system latency. This finding confirms one of the generally adopted qualitative implications of the CAP theorem [8]. However, system developers have not had any mathematical tools to help them to accurately predict response time of the large-scale replicated systems so far. Estimation of the system worst-case execution time still remains a common practice for many applications (e.g. embedded computer systems, server fault-tolerance solutions, like STRATUS, etc.). However this approach is no longer a viable solution for the wide-area service-oriented systems which components can be distributed all over the Internet. In our previous works [2–4] we demonstrated that unpredictable extreme delays exceeding the value of ten average response times could happen in such system quite often. In the paper we propose a set of novel analytical models providing a *quantitative basis* for the system response time prediction depending on the consistency level provided to (or requested by) clients. The models allow us to derive probability distribution function of the system response time corresponding to the particular consistency level (ONE, ALL or QUORUM) by incorporating probability density functions of replica response times.

Validity of the proposed models was verified against the experimental data we reported in Section 3. It was demonstrated that the proposed models provide significant accuracy of the system average response time prediction, especially in case of ONE and QUORUM consistency levels. The proposed models provide a mathematical foundation for predicting latency of distributed fault and intrusion-tolerance techniques working over the Internet. The models take into account the probabilistic uncertainty of replicas response time and the required consistency level.

The practical application of our work is in allowing practitioners to predict system performance, and in offering them a crucial support for the optimal time-out set-up and for understanding a trade-off between system consistency and latency. Trading off the system consistency against latency in runtime requires the knowledge of probability density functions (and the parameter values) that accurately approximate replicas response time. System developers can get this kind of statistical characteristics from the previous experience or based on the testing results. Besides, probability density functions can be (re-)estimated more accurately at runtime or during the trial usage after the system and all its replicas are deployed.

7 Acknowledgements

We are grateful to Aad van Moorsel for his feedback on the earlier version of this work and Batyrkhan Omarov for his help with running some of the experiments.

8 References

1. Lee, P.A., Anderson, T.: Fault Tolerance. Principles and Practice. Springer-Verlag (1990)
2. Gorbenko, A. et al.: Real Distribution of Response Time Instability in Service-Oriented Architecture. In: 29th IEEE Int'l Symp. Reliable Distributed Systems, pp. 92–99 (2010).
3. Gorbenko, A. et al.: Exploring Uncertainty of Delays as a Factor in End-to-End Cloud Response Time. In: 9th European Dependable Computing Conference, pp. 185–190 (2012).
4. Chen, Y. et al.: Measuring and Dealing with the Uncertainty of the SOA Solutions. In: Performance and Dependability in Service Computing: Concepts, Techniques and Research Directions, V. Cardellini et al., eds., IGI Global, pp. 265–294 (2011)
5. Potharaju, R., Jain, N.: When the Network Crumbles: An Empirical Study of Cloud Network Failures and their Impact on Services. In: 4th ACM Symposium on Cloud Computing (SoCC) (2013)
6. Scott, C., Choffnes, D.R., Cunha, I., etc. LIFEGUARD: practical repair of persistent route failures. In: ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication, 395-406 (2012)
7. Avizienis, A., Laprie, J.-C., Randell, B., Landwehr, C.: Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Transactions on Dependable and Secure Computing. 1(1), 11–33 (2004)
8. Brewer, E.: Towards Robust Distributed Systems. In: 19th Ann. ACM Symposium on Principles of Distributed Computing (PODC'00), 7–10 (2000).
9. Gilbert, S., Lynch, N.: Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. ACM SIGACT News. 33(2), 51-59 (2002)
10. Gorbenko, A., Romanovsky, A., Tarasyuk, O., Kharchenko. V. Dependability of Service-Oriented Computing: Time-Probabilistic Failure Modelling. In Avgeriou, Paris (Ed.). Software Engineering for Resilient Systems. Lecture Notes in Computer Science, Volume 7527/2012, 121–133 (2012)
11. Abadi, D.J.: Consistency Tradeoffs in Modern Distributed Database System Design. IEEE Computer. 45(2), 37–42 (2012)
12. Brutlag, J.: Speed Matters for Google Web Search. Google, http://services.google.com/fh/files/blogs/google_delayexp.pdf (2009)
13. Lakshman, A., Malik, P.: Cassandra: a decentralized structured storage system. ACM SIGOPS Operating Systems Review. 44(2), 35–40 (2010)
14. Gorbenko, A., Romanovsky, A.: Time-Outing Internet Services. IEEE Security & Privacy. 11(2), 68–71 (2013)
15. Gorbenko, A., Kharchenko, V., Romanovsky, A.: Using Inherent Service Redundancy and Diversity to Ensure Web Services Dependability. In: Methods, Models and Tools for Fault Tolerance, LNCS 5454, Springer, 324–341 (2009)
16. Rao, J., Shekita, E.J., Tata, S.: Using Paxos to Build a Scalable, Consistent, and Highly Available Datastore. In: VLDB Endowment, 243–254 (2011)