Experimental Analysis of the Correlation of HTTP GET Invocations

Philipp Reinecke[1], Aad P. A. van Moorsel[2], and Katinka Wolter[1]

[1] Humboldt-Universität zu Berlin
Institut für Informatik
Berlin, Germany
{preineck,wolter}@informatik.hu-berlin.de

[2] University of Newcastle upon Tyne
School of Computing Science
Newcastle upon Tyne, United Kingdom
aad.vanmoorsel@ncl.ac.uk

Abstract. In this paper we experimentally investigate if optimal retry times can be determined based on models that assume independence of successive tries. We do this using data obtained for HTTP GET. This data provides application-perceived timing characteristics for the various phases of web page download, including response times for TCP connection set-ups and individual object downloads. The data consists of pairs of consecutive downloads for over one thousand randomly chosen URLs. Our analysis shows that correlation exists for normally completed invocations, but is remarkably low for relatively slow downloads. This implies that for typical situations in which retries are applied, models relying on the independence assumption are appropriate.

1 Introduction

When a computing job or task does not complete in a reasonable time, it makes common sense to retry it. Examples are plentiful: clicking the browser refresh button, retry of TCP connection attempts at expiration of the retransmission timer, reboot of machines if jobs do not complete ('rejuvenation'), preemptive restarting of a randomised algorithm using a different seed, etc.

In concrete terms, a retry makes sense if a new try takes less time to complete than the ongoing attempt would have taken. If we assume that the completion times of consecutive tries are independent and identically distributed and that no time penalty is incurred when issuing a retry, it can be shown that retries improve the overall mean completion time when the completion time distribution is of a particular type, most notably heavy-tailed, bi-modal or log-normal [9]. Since we know from existing experimental work (e.g., [5]) that Internet response times fit such distributions, Internet applications potentially respond positively to retries.

However, the above reasoning assumes independent identically distributed tries, an assumption that may not necessarily hold. Hence, models that determine optimal retry times based on the independence assumption [2, 4, 5, 9] may not necessarily be appropriate either. To analyse if and to what extend the independence assumption holds for Internet applications, we conducted experiments and collected data for consecutive HTTP GET invocations. This paper reports on the analyses of the correlation characteristics of this data. To the best of our knowledge such data capturing and analysis has not been done before (compare for instance the surveyed experimental work in [3]). In earlier work [6], we analysed HTTP GET invocations, but did not possess the detailed data for subsequent requests to the same URL we use in this paper.[1]

Our experiments capture data for many consecutive downloads from individual URLs, always executed in pairs. This allows us to investigate correlation between consecutive attempts as well as to determine the distribution of completion time. We collected data for various phases of the download of a web page, distinguishing TCP connection set-up, time until the first data has been received, intermediate 'stalling' times and the overall download time of an object. We are interested in these detailed metrics because they potentially provide clues on when to initiate a retry. We only obtained data for metrics that are visible at the application level, since we are interested in investigating retries that could be introduced in an Internet application such as a browser or a software agent.

We gained the following insights from the analysis in this paper. On the one hand, if we consider all samples, the correlation between subsequent tries is surprisingly low, irrespective of the considered phase in a web page download. On the other hand, if we only consider attempts that complete 'fast' (that is, within a small deviation of the average), the correlation is considerable. We conclude from this that the independence assumption is reasonable for model-based optimisation of retry times, *provided one limits retries to times at which completions can be considered slower than normal*. Clearly, this corresponds to the case in which one would want to consider retries to begin with. We also study the correlation between different phases of the same download, and conclude that these typically exhibit very low correlation. This makes it difficult to make use of knowledge of earlier phases of the download to predict the remaining download time (as has been utilised in [7] for retries of database queries).

We now first describe our experiment set-up and its underlying system model, as well as the metrics we observed. We present the collected data in Sect. 3, followed by the statistical analysis.

[1] The data used in [6] contains completion times for TCP connection set-up, images, objects and complete web pages for three experiments using 56,000 randomly selected URLs. However, it lacks data for subsequent tries over long periods of time. We have made available on the web [1] both the data set used in [6] and the one used in this paper.

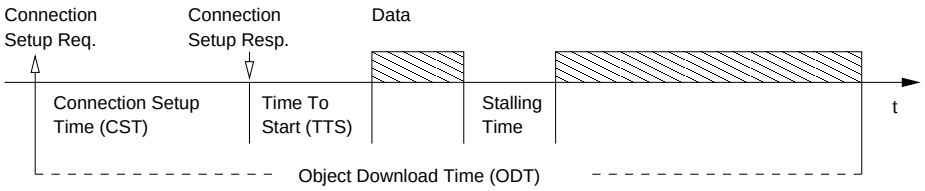

Fig. 1. Metrics and measurement points during the download task.

2 Experiment Design and Execution

Figure 1 depicts response times of HTTP GET invocations as an application perceives it. The figure shows the various phases during the download of a single object (such as a page or an image) and denotes the metrics used in our experiments.

We call the overall time spent downloading the Object Download Time (ODT), which is comprised of:

- Connection Set-up Time (CST): the time for the TCP connection set-up
- Time To Start (TTS): the time between sending the GET request and receiving the first data
- the time consumed by actual data transfer (we do not collect this time explicitly)
- Longest Stalling Time (LST): the length of possible stalling periods during the download, of which we record the longest

These metrics allow us to assess the performance of a download in three distinct phases:

Connection Set-up Time: In TCP, connection set-up is accomplished by a three-way-handshake and error handling procedures involving the Retransmission Timeout (RTO) timer. From the application's point of view, this task corresponds to calling a `connect()` function, and waiting for this call to return the connection. The application has no means to infer the state of the connection set-up, and thus the performance of this whole process is described by a single metric, namely its length.

Time To Start: The client proceeds by sending its request over the connection. The network transports the request data to the server, which generates an answer and sends it back to the client. At the transport layer, the connection's receive and send windows are adjusted to accommodate for network characteristics ('slow start'). At the application level a new server instance may have to be started on the server, and content has to be prepared and sent out. As with connection set-up, the intricacies of this phase remain hidden from the client application, since it cannot observe whether its request actually reached the server, was processed, nor whether a reply was

sent back. Instead, it encounters a period of inactivity between sending the request and receiving the first chunk of data.

Longest Stalling Time: During the download, various factors (e.g., network or server congestion) may lead to periods of temporary stalling. In these periods, no new data is available to the application when it polls the socket (done at 20 ms intervals, see below). We always track the longest such interval. If there was no stalling period, LST will be zero.

Object Download Time: Together, CST, TTS, LST, other stalling intervals, and the length of periods in which data arrives add up to the Object Download Time. That is, ODT is the time it takes from initiating the download until the object is available to the application.

2.1 Experiments

To conduct our experiments we implemented a Java client that repeatedly downloads web pages from a set of hosts and measures the above-mentioned performance indicators. The client issues `GET / HTTP/1.0` requests to a web server, and the ODT metric then corresponds to the time it takes to download the web server's root document (e.g., `index.html`) without any images or other objects that might otherwise be included.

To obtain a sample, the client chooses a URL at random. We randomise the order in which we visit servers to decrease the likelihood of introducing substantial dependencies between ordered requests to different machines. The Java client then downloads the host's root document twice in a row, with a 20 ms pause in between. The pairs of samples thus obtained allow us to study correlation between subsequent requests. All our samples have a 20 ms granularity, because the client polls a TCP socket for new data at 20 ms intervals.

The initial list of URLs was determined in two passes. First, to create a reasonably random set of live URLs, we fed words from a large word list into the Google search engine and extracted the links (the first 100) from the results. In earlier experiments we used all of the resulting 56,000 URLs. In the current experiment, we reduced the list by randomly drawing 2000 entries for detailed investigation. As failing hosts may stall the experiments, any URL that produces fatal errors or exceeded certain time thresholds (24 s for CST, 60 s for TTS, and 1200 s for ODT) was automatically removed from the list. Due to this mechanism, our list shrank, and the data presented in the next section stems therefore from 725 unique URLs, each yielding a large number of samples (at least 1000).

The experiments were run on two Linux PCs connected to the Internet using 768kbit ADSL dial-up with the same ISP. On the first, a 2.0 GHz PC, the experiment ran for 22 days, on the second, a 1.6 GHz PC, we collected samples for 10 days. The experiments faced three interruptions (interrupted dial-up connection, etc.), and we accounted for these interruptions in our analysis in such a way that they did not influence the results.

Table 1. Overview of data set characteristics for the 725 URLs analysed. All times are given in milliseconds.

	CST_1	CST_2	TTS_1	TTS_2	LST_1	LST_2	ODT_1	ODT_2
First data set (22.09.2004, 6:44 – 14.10.2004, 15:48; 816397 samples)								
mean	178.9	180.4	311.2	295.6	62.9	62.3	721.7	704.7
median	160	160	180	180	0	0	440	440
std. deviation	301.9	229.6	1663.5	1627.9	404.9	427.5	1918.6	2280.8
CoV	1.687	1.273	5.345	5.507	6.437	6.862	2.658	3.236
minimum	0	20	40	0	0	0	0	40
maximum	23700	24000	58440	60020	197480	309000	317700	1201880
Second data set (07.10.2004, 11:36 – 17.10.2004, 20:04; 212805 samples)								
mean	174.6	173.4	293.9	279.0	63.7	62.4	1382.4	1347.2
median	160	160	180	180	0	0	840	840
std. deviation	320.3	250.4	1576.6	1545.0	264.3	201.9	3547.126	6242.9
CoV	1.834	1.444	5.364	5.537	4.15	3.235	2.566	4.634
minimum	20	20	20	0	0	0	320	340
maximum	21440	24000	57880	60020	57420	26760	274940	2420360

3 Results

Table 1 gives an overview of the results for the metrics defined in Sect. 2. The first and second trial are denoted by subscripts 1 and 2, respectively. Note that for Time To Start and Object Download Time the second attempt is typically faster than the first. Most likely the second request benefits from work already done for the previous one, e.g., the server might re-use the server instance that handled the first attempt and could also serve cached data. However, this effect is not very large, and close to negligible if we consider the median (at 20 ms granularity).

It is worth mentioning that all sampled metrics have a coefficient of variation (CoV, which is defined as the standard deviation divided by the expectation of the considered metric) greater than one, indicating high variability. This holds especially true for TTS and the Longest Stalling Time. The high variability of TTS can most likely be explained by the dependence of TTS on network conditions, the server, the client's operating system state, etc. The high CoV value for LST may be explained by the fact that it is a maximum value and is therefore rather unpredictable.

In interpreting our results it is important to understand the working of TCP's retry mechanism. During connection set-up, TCP initiates a retry when the RTO times out. RTO first expires after 3000 ms and doubles after every expiration, i.e., RTO expires and TCP re-attempts to set up a connection 3, 9, 21, 45 and

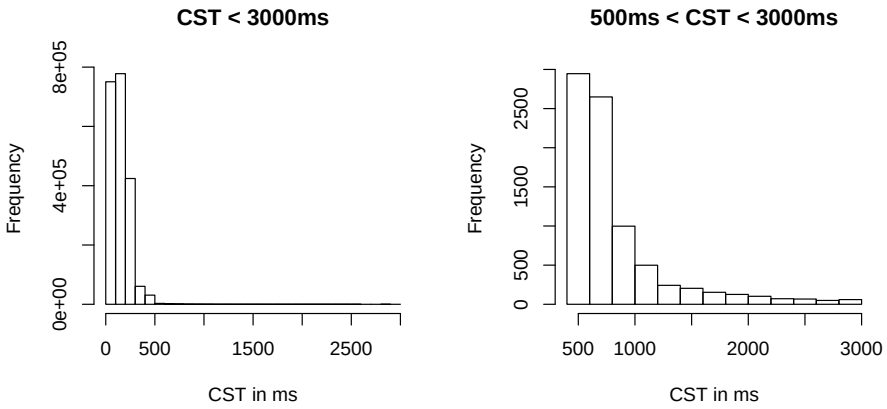

Fig. 2. Histograms for all CST < 3000 from both sets.

93 seconds after the initial attempt. The consequence of this mechanism on the download time can be observed very well in Fig. 3, as we explain in Sect. 3.1.

We note that in our experiments about 0.3 percent of all TCP connection set-up attempts experienced one or more RTO expiries, i.e., CST > 3000 ms (this is not directly visible in the table). In earlier experiments [6] we found that the IP-level failure percentage for TCP connection set-up is close to 0.6 percent. The difference can probably be explained by our method of selecting URLs, which tends to favour less failure-prone hosts.

Figure 2 illustrates the distribution of CSTs for attempts without RTO expiration, with the right side graph zooming in on the left. The histogram in Fig. 2 suggests that for non-failed connection set-ups, CSTs for random URLs have quite a high variance and frequently exhibit a relatively long completion time. This may be interesting in its own right, but also bodes well for issuing restarts, as was also found in [5, 8]. The question remains, however, whether consecutive connection set-up times are independent, an issue we will study now.

3.1 Correlation

Various models that are used to determine the optimal retry time rely on the assumption that subsequent tries exhibit independent and identically distributed (iid) completion times. If the iid assumption is true, then correlation will be zero. In other words, if correlation is high, we conclude that the independence assumption is not valid. Therefore we now analyse the correlation characteristics of our data. The analysis presented here is based solely on data from the first data set, but we found that both sets lead to similar conclusions regarding correlation.

A first visual indication of the degree to which consecutive downloads are correlated is given by the scatter plots in Fig. 3 and Fig. 4, for the CST and ODT

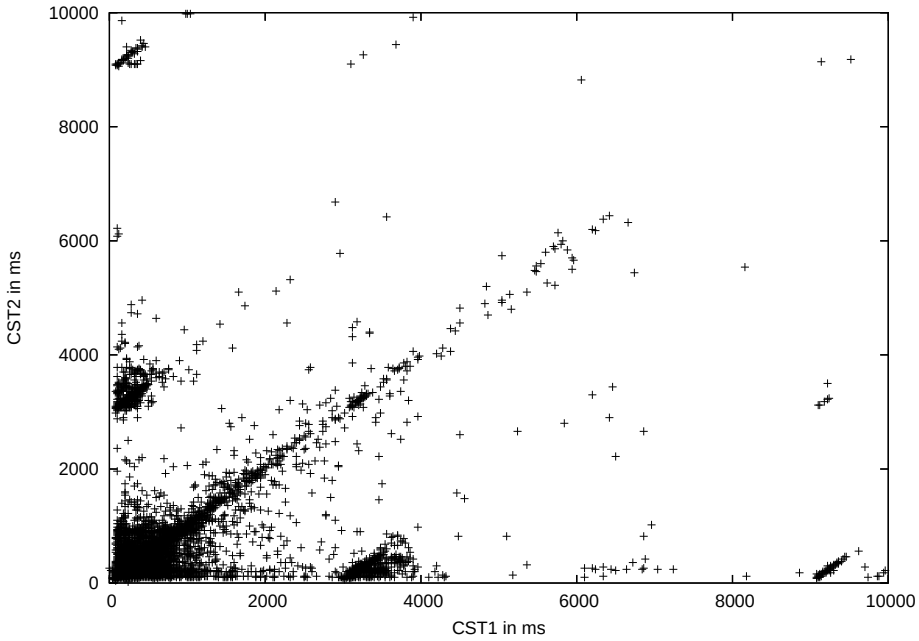

Fig. 3. Scatter plot for CST1 and CST2. TCP's RTO values are clearly visible in the clusters in the off-diagonals around 3000ms and 9000ms on each axis.

metric, respectively. Scatter plots of consecutive attempts show the correlation between the first and second attempt by plotting the duration of the former against that of the latter. For points close to the diagonal the first and the second download experienced roughly equal completion times, and therefore indicate strong correlation. Points far off the diagonal signal low correlation. Data points below the diagonal of the scatter plot could have benefitted from a retry, since the second attempt would have taken less time than the first. Points above the diagonal would not have benefitted from a retry.

Figure 3 depicts connection set-up times. The importance of TCP's RTO timeout values shows quite clearly through clusters of samples just above 3000 ms and 9000 ms near both axes. Since these two clusters are far from the diagonal, they indicate low correlation between consecutive connection set-ups. However, note that these clusters (indicating low correlation) correspond to cases in which one of the two connection set-up attempts experienced failures on the IP level. In cases where both connection set-ups succeed without an RTO expiry (that is, both CST_1 and CST_2 are below 3000 ms), the samples more strongly gravitate towards the diagonal, indicating substantial correlation.

The clusters caused by the RTO time out values are still faintly visible in the scatter plot for ODT_1 and ODT_2 (Fig. 4). However, this picture is much more diffuse, as could already have been expected based on the coefficient of variation

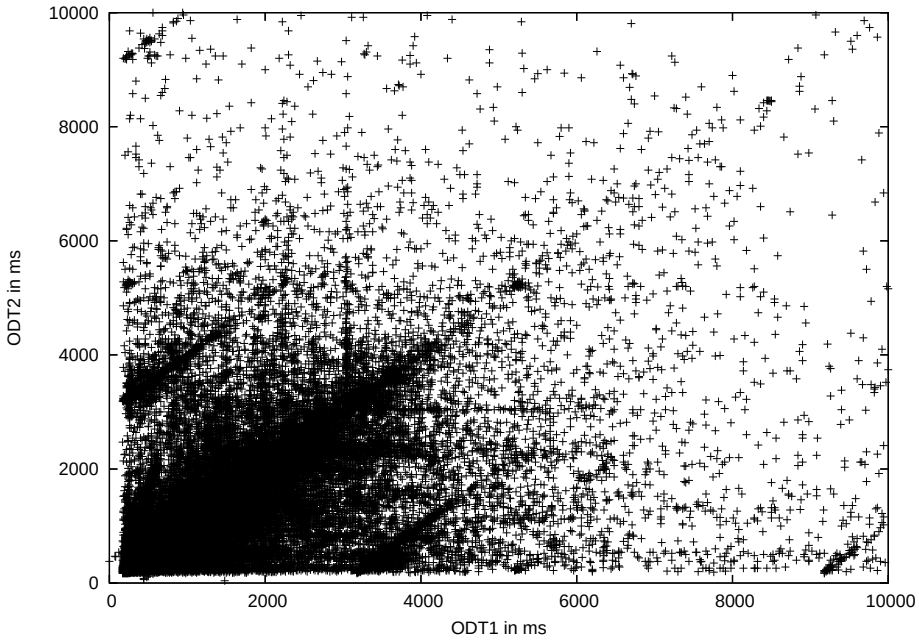

Fig. 4. Scatter plot for ODT1 and ODT2.

values in (Tab. 1) (the ODT coefficient of variation is double that for CST). Together, these two observations suggest that, while delays introduced by the RTO mechanism may have a strong influence on the ODT, the additional factors affecting this metric can change the pattern of observed ODTs considerably.

Although they provide a strong visual insight into the amount of correlation, the scatter plots do not objectively quantify the degree to which observations are actually correlated for a given set of URLs. To this end, we study the distribution of correlation coefficients per URL, as shown in the histograms in Fig. 5 until Fig. 7. That is, we split the data set by URLs and for every URL treat the (thousand or more) observations for each metric M from the first and the second attempt as resulting from two random variables M_1 and M_2. We then compute $Cor(M_1, M_2)$ for each URL and display the distribution of $Cor(M_1, M_2)$ over all URLs. Figure 5, for instance, shows that if we consider CST, close to 300 URLs have correlation in the range between 0.0 and 0.1.

In our discussion of the scatter plots, we already touched upon the difference in correlation of 'failed' attempts (i.e., failed on the IP level) and 'fast' attempts. This issue can be studied quantitatively by comparing the left and right hand bar chart in Fig. 5. The left hand side, where we consider all attempts, shows low correlation, whereas the right hand side indicates high correlation if we consider only pairs that were successful (CST below 3000 ms) in both tries. This indicates that the independence assumption is only valid in the 'failed' case, not for the

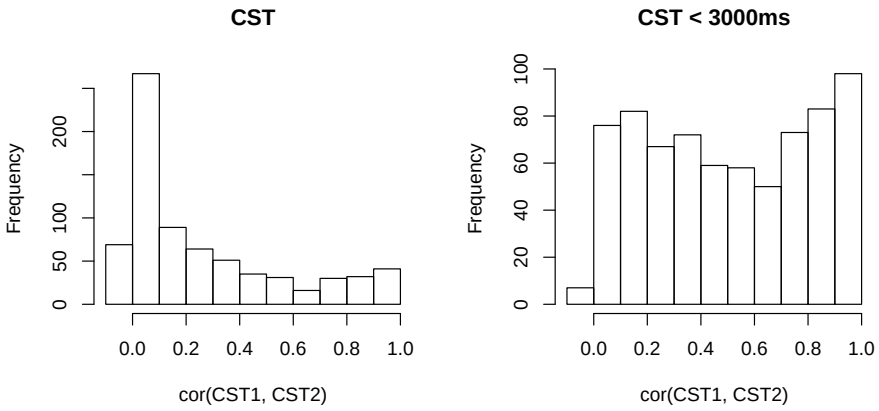

Fig. 5. Histograms of correlation coefficients for (CST_1, CST_2).

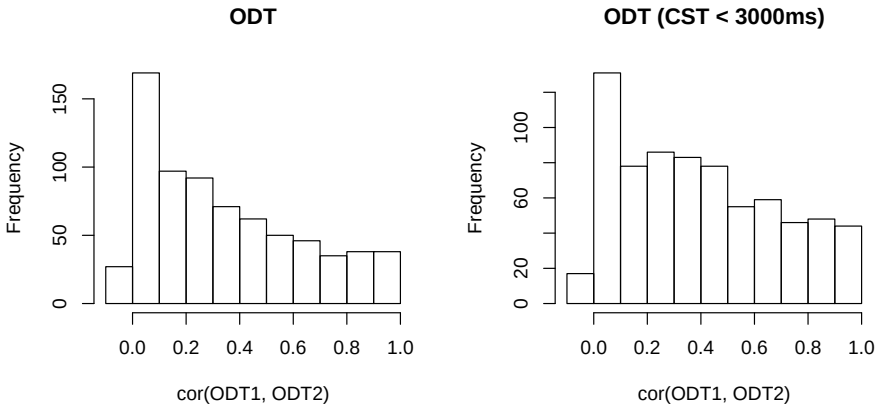

Fig. 6. Histograms of correlation coefficients for (ODT_1, ODT_2).

common situation of normally succeeding attempts. So, even though successful completion times may be distributed according to a distribution that is amenable to retries (log-normal, heavy-tail), this does not imply that retries do indeed pay off because the independence assumption is probably invalid. However, when one considers higher retry times, retries at those times are not highly correlated, and models based on the independence assumption are likely to be valid.

This difference between correlation for 'failed' and successful attempts can also be observed for ODTs in Fig. 6, although here there remains a larger portion of URLs with low correlation. The latter may find its explanation in the low

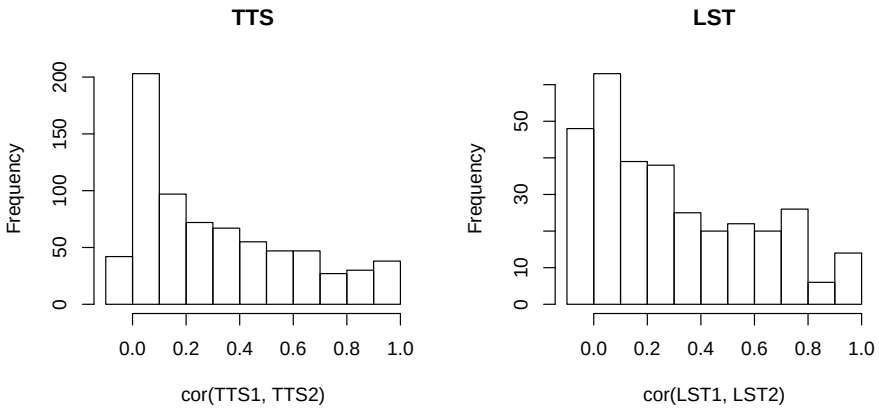

Fig. 7. Histograms of correlation coefficients for (TTS_1, TTS_2) and (LST_1, LST_2).

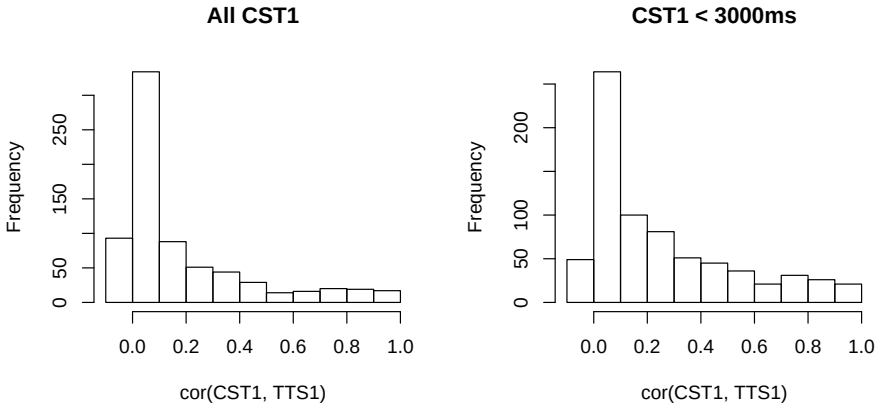

Fig. 8. Histogram of correlation coefficients for (CST_1, TTS_1).

correlation for TTS and LST, as seen in Fig. 7. The time-to-start values of the first and the second attempt seem often, but not always, uncorrelated, as can be seen in Fig. 7 on the left, while the longest stalling time often is long (or short) again upon the second try, if it was long (or short) before, as can be seen in Fig. 7 on the right.

Cross-Measure Correlation A final point of interest is the correlation between various phases of the download. If correlation exists, samples for one met-

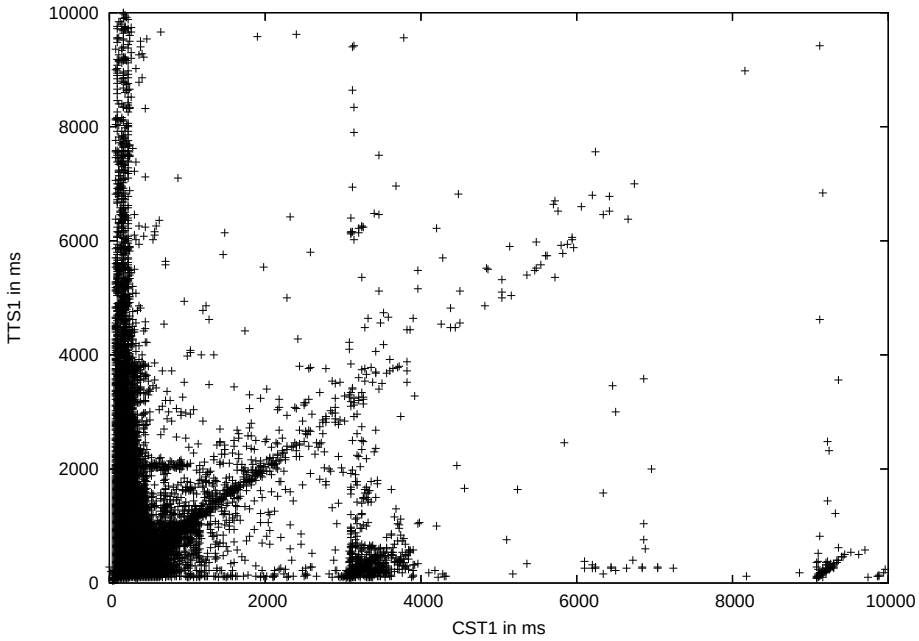

Fig. 9. Scatter plot for CST1 and TTS1, zoomed in.

ric M may be used to predict the completion time of a subsequent metric M' within the same download (along the lines of [7]).

Figure 8 is just one example for the likelihood of such correlation, namely that between connection set-up time and time to start. It shows that correlation is relatively low, and similar behaviour exists between the other phases. This makes it difficult to exploit possible inter-phase dependence for computing retry times. Figure 9 illustrates the fact further. In this scatter plot we observe that there is no obvious relation between CST and TTS; TTS may take on any value when the connection set-up was very fast, and vice versa. The cluster for values of TTS close to zero at CST values of 3000 ms and at 9000 ms signify that retries of connection set-ups do not make large TTS more likely. The diagonal, indicating positive correlation, is pronounced only for small values of both metrics, and, in particular, for CST$<$ 3000. This latter observation indicates that inter-phase dependencies may be exploitable only in the absence of packet loss.

4 Conclusions

In this paper we have analysed empirical data sampled using HTTP GET. We have investigated correlation patterns in the data and found high correlation when using data from connections that were successfully set up straight away,

i.e., with Connection Set-up Times of less than 3000 ms. For these, not only the corresponding Connection Set-up Times are correlated, but also the Object Download Times. However, when considering data from connections that 'failed' (i.e., Connection Set-up Time greater than 3000 ms), we found very little correlation. The consequence of our finding is that models that rely on the independence of successive tries will not likely be useful to determine retry times for fast tries (even if their distribution is amenable to retries). However, models based on the independence assumption *are* appropriate when one wants to determine optimal retry times for slower or 'failed' attempts. Since retries are most relevant in this latter situation, this validates the use of optimisation models that rely on the independence assumption.

References

1. The data discussed in this paper is available from the web site http://homepages.cs.ncl.ac.uk/aad.vanmoorsel/data.
2. H. Alt, L. Guibas, K. Mehlhorn, R. Karp and A. Wigderson, "A Method for Obtaining Randomized Algorithms with Small Tail Probabilities," *Algorithmica,* Vol. 16, Nr. 4/5, pp. 543–547, 1996.
3. F. Donelson Smith, F. Hernandez Campos, K. Jeffay and D. Ott "What TCP/IP Protocol Headers Can tell Us About The Web," *SIGMETRICS*, pp. 245–256, 2001.
4. M. Luby, A. Sinclair and D. Zuckerman, "Optimal Speedup of Las Vegas Algorithms," *Israel Symposium on Theory of Computing Systems,* pp. 128–133, 1993.
5. S. M. Maurer and B. A. Huberman, "Restart strategies and Internet congestion," in *Journal of Economic Dynamics and Control,* vol. 25, pp. 641–654, 2001.
6. P. Reinecke, A. van Moorsel and K. Wolter, "A Measurement Study of the Interplay between Application Level Restart and Transport Protocol," in *Springer Verlag Lecture Notes in Computer Science 3335, International Service Availability Symposium: Revised Selected Papers,* M. Malek, M. Reitenspiess and J. Kaiser (Eds.), pp. 86–100, 2005.
7. Y. Ruan, E. Horvitz and H. Kautz, "Restart Policies with Dependence among Runs: A Dynamic Programming Approach," in *Proceedings of the Eight International Conference on Principles and Practice of Constraint Programming,* Ithaca, NY, 2002.
8. M. Schroeder and L. Boro, "Does the Restart Method Work? Preliminary Results on Efficiency Improvements for Interactions of Web-Agents," in *Proceedings of the Workshop on Infrastructure for Agents, MAS, and Scalable MAS at the Conference Autonomous Agents 2001,* T. Wagner and O. Rana (Eds.), Springer Verlag, 2001.
9. A. van Moorsel and K. Wolter, "Analysis and Algorithms for Restart," *Proceedings of International Conference on Quantitative Evaluation of Systems,* Twente, Netherlands, pp. 195–204, 2004.