# A Gaussian Majorization for the Estimation of the Web Services Execution Time

F.Lelli
*INFN Laboratori Nazionali di Legnaro*

## INTRODUCTION

Monitor and control operations demand deep interaction between users and devices, while they require the adoption of high interoperable solutions that only SOA-based Web Services can offer. When the access is performed via internet using Web Services calls, the remote invocation time becomes crucial in order to understand if a service can be controlled properly, or the delays introduced by the wire and the serialization/deserialization process are unacceptable. We present and validate methodologies and algorithms, based on a $2^k$ factorial analysis and a Gaussian approximation of previous service execution times, which enable the estimation of a generic remote method execution time.

## REMOTE METHOD EXECUTION TIME EXTIMATION

A remote method invocation can be split into 7 crucial parts. During the interval *t1-t0* the client serializes the invocation input in SOAP format and sends it during the interval *t2-t1*. The remote peer receives the serialized message at time *t2* and starts the deserialization process that ends at time *t3*. During the interval *t4-t3* the remote method is executed and the output is produced. As last operation, the remote service serializes the output in SOAP format during the interval *t5-t4* and starts sending it. The invoker receives the serialized message at time *t6* and during the interval *t7-t6* starts deserializing the output message in the proper data structure.
A remote method invocation represents a time interval during which the method will be executed with a given percentile. In other words, if *h(x)* represents the probability distribution of the statistic variable *x* that describes the service invocation execution time, its integral represents the minimum time interval [0< Y♯ that can ensure the execution time with a given probability $\overline{X}$. Since we are trying to estimate a possible deadline, an upper bound estimation of *Y* is still an acceptable value while a lower bound cannot be considered. In other words, we can accept to try to predict a 95th percentile, when *de facto* the real number of deadline miss is 99th percentile. We cannot accept the opposite behaviour. An additional remark is that the probability function must take into account client and server load, the input and output type and size etc.

Finally we need to point out that the exact probability distribution *h(x)* depends on many factors and it is quite difficult to estimate, so we could try to give an upper bound like the following:

$$P(x < \overline{X}) \le P(k < \overline{X})$$

Where the probability distribution of $k$ is described by a Gaussian ($N(\mu,\sigma)$) distribution larger than the $x$ distribution, with a given average (Avg or $\mu$) and standard deviation (sDev or $\sigma$)

$$N(\mu,\sigma) \quad \mu = f(\underline{x}) \quad \sigma = g(\underline{x})$$

We can note that $\mu$ and $\sigma$ are deterministic functions of the key mentioned factors (marked as $\underline{x}$). If we succeed in the estimation of the average and the standard deviation of the presented Gaussian, we will be able to provide an upper bound of the remote method execution time, noting that:

$$Y = P(x < \overline{X}) \le q_{\overline{X}} = g(\underline{x})\Phi_{\overline{X}} + f(\underline{x})$$

Where $\Phi_{\overline{X}}$ represents the percentile of order $\overline{X}$ of a Gaussian distribution with Avg=0 and sDev=1. Considering that both f(x) and g(x) must be determined in an empiric way, we can not ignore the experimental error due to this estimation, in particular

$$\mu = f_e(\underline{x}) \pm \varepsilon_\mu \quad \sigma = g_e(\underline{x}) \pm \varepsilon_\sigma$$

Again, considering that we want to estimate an upper bound of a time, only positive errors must be considered. In addition, the errors must follow a conservative approach because we want predict worst case functions, as explained at the beginning of this section.
In conclusion the final dead line estimation will be:

$$Y_e = g_e(\underline{x})\Phi_{\overline{X}} + \varepsilon_\sigma\Phi_{\overline{X}} + f_e(\underline{x}) + \varepsilon_\mu$$

If the influence of some factors $\underline{x}$ cannot be considered under the client control (i.e. the server load), we need to substitute the worst case scenario into the just above equation. As final remark we need to consider factors like the algorithm of the method and the key input that change the algorithm behaviour. Assuming that the remote algorithm execution time is *Ra(x)* and that it will be provided by programmers [1], the final estimation of the critical intervals will be:

$$t7 - t0 = T_{ex} = g_{ex}(\underline{x})\Phi_{\overline{X}} + \varepsilon_{\mu_{ex}}\Phi_{\overline{X}} + f_{ex}(\underline{x}) + \varepsilon_{\sigma_{ex}} + Ra(\underline{x})$$
$$t3 - t0 = T_{ow} = g_{ow}(\underline{x})\Phi_{\overline{X}} + \varepsilon_{\mu_{ow}}\Phi_{\overline{X}} + f_{ow}(\underline{x}) + \varepsilon_{\sigma_{ow}}$$
$$t4 - t0 = T_{el} = g_{ow}(\underline{x})\Phi_{\overline{X}} + \varepsilon_{\mu_{ow}}\Phi_{\overline{X}} + f_{ow}(\underline{x}) + \varepsilon_{\sigma_{ow}} + Ra(\underline{x})$$

Of course, if Ra*(x)* would be known with a given error, it will need to be taken into account too [1].

The mentioned functions have been estimated using a methodology based on the $2^k$ factorial analysis [2] according to a set of collected experimental data.

## ON VALIDATING THE PROPOSED METHODOLOGY

We validated the methodology with an additional set of tests in order to use different samples from the one used for building the deadline estimators. Clients continuously and randomly choose the input and the output of the remote method and then request the service. In addition, in order to validate the methodology in the worst case scenario, by looking the residual of the function [2], we will assume that the client CPU Load is always larger than 80%.

In all tests conducted, client and server were running on a Dual Xeon 2.40GHz, 1.5GB RAM machines running the CERN Scientific Linux 3.0.4 Operating System, with Kernel 2.4.21-27.0.2.EL.cernsmp and Java 1.4.2-08-b03. The machines were interconnected to each other by a 100 MB switched Ethernet. Finally on top of this hardware and software layers we installed Tomcat 5.0.28 and Axis 1.4 as Web Service provider.

With the developed methodology we are capable to tune the deadline estimation taking into account the remote CPU usage factor. By the way, in a real scenario, it is impossible to maintain the server CPU usage constant for the entire remote method invocation. We consider the situation where the server CPU usage from different processes is always 80%. In addition we analyzed the behaviors of predictors in the case where the server is completely empty, in order to understand how much this wrong assumption impacts in this limit scenario. By the way, we need to point out that we could remove this assumption in case of an advanced reservation. With these assumptions, we performed 5000 remote method invocation with random input and output, in a scenario where only one client was invoking a loaded and unloaded server. We also repeated the previous tests with a variable number of clients (2, 4, and 6) connected to the server. Figures 1 and 2 report the results. The "total95" and "total975" curves represent the dead line estimation for a remote method execution using a direct computation of the linear regression with a quantile of 95 and 97.5 respectively. The "comp95" and "comp975" curves show the deadline estimations using the interval decomposition technique. The “OneWay” curves have similar meaning but they refer to the prediction of *t3-t0*. As expected, in both cases less than the 5% and 2.5% of dead line misses were experienced, because of the upper bound that we were forced to consider in the dead line functions estimations. In addition Figure 3 reports a test snapshot for the Remote Method Execution (*t7-t0*) in which 3 clients are concurrently requesting the service. We can note that the remote execution time prediction follows the “real” curve and that direct estimations are more accurate because of the minor experimental error.

We can conclude that the proposed methodology provides an upper bound of a remote method execution time. As we can note from Figures 1 and 2, the deadline estimation works only if the server is not overloaded, i.e. if the number of clients is three or less. According to [3], 3 concurrent clients that try to use the remote service as fast as they can, bring the server to the maximum request throughput that it can handle. With the testbed used for building this validation we succeeded to handle more that 100 clients that perform parallel requests to the service, only if the total service throughput is less than the maximum of server capability [3].

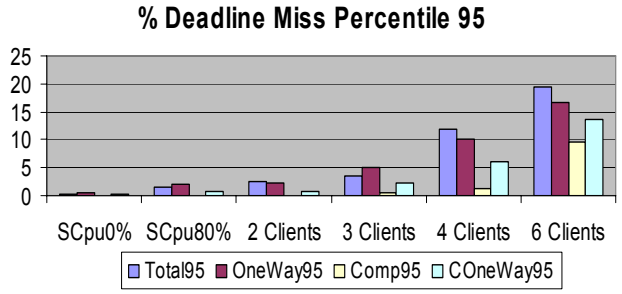


FIG. 1. *% of Deadline with expected Percentile 95*

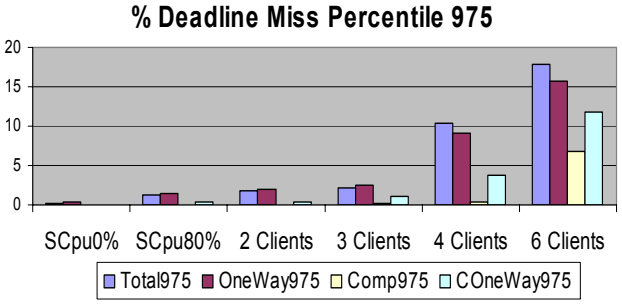


FIG. 2. *% of Deadline with expected Percentile 975*

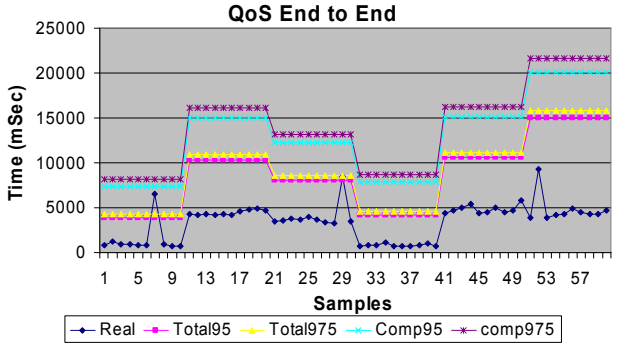


FIG. 3. *a Test snapshot with 3 clients, estimation of t7-t0*


[1] G. Buttazzo, G. Lipari, L. Abeni, M. Caccamo. Soft Real-Time Systems: Predictability vs. Effciency. Springer, 2005.

[2] D. C. Montgomery. Design and Analysis of Experiments 5th edition. John Wiley & Sons Inc, December 2004.

[3] F. Lelli, et all. Improving the Performance of XML Based Technologies by Caching and Reusing Information. in proc of International Conference of Web Services (ICWS06), IEEE Computer Society volume 1:pag 689–700, september 2006.