On the Response Time of Large-scale Composite Web Services

Michael Scharf

Institute of Communication Networks and Computer Engineering (IKR)
University of Stuttgart, Pfaffenwaldring 47, 70569 Stuttgart, Germany
Email: scharf@ikr.uni-stuttgart.de

Abstract: This paper studies the response time of a web service middleware that decomposes requests into sub-queries to different servers and then merges the results. We present a queuing model for such a fork-join system and an exact analysis for exponential server response times. We also provide accurate approximations for heavy-tailed server response times, which are a common effect in the Internet. Heavy-tailed distributions are critical since they may cause very long middleware response times, in particular in large-scale systems with many servers being involved. We show that in this case the performance can be significantly improved if the middleware does not have to wait for a few slow servers, i. e., if the merged result does not need to be absolutely complete. We discuss different choices to implement such a mechanism and quantify their impact on the middleware response time.

Keywords: Performance evaluation, service platforms, database federation, scalability

1 INTRODUCTION

Many upcoming web services are based on information stored in distributed databases that are offered by different service providers. A middleware can provide uniform access to these multiple data sources. In order to handle a request, such a middleware must query different service providers in parallel and integrate their responses into a single result. In the context of web services, this mechanism is an example of service *composition* [1]. The same principle is known as data *merging*, *mediation*, or *federation* in distributed databases. Service composition is based on the *fork-join* computing paradigm [2]: A request, or generally speaking a job, is distributed over several service units and can only be finished when all of them have completed processing. Thus, the overall response time includes a synchronization delay that is determined by the slowest service unit.

In this paper, we study the response time of a centralized middleware component performing large-scale composition of web services. Our work is similar to a recent study [1] that analyzes the effects of exponential response times based on an earlier work in [3]. Unlike [1, 3], we explicitly consider the effort of joining results. Detailed queuing models for distributed databases have also been studied extensively, see e. g. [4] and references therein. However, most existing work only considers constant or exponential service times. As will be shown later, measurements in the WWW and in e-commerce systems have observed heavy-tailed server response time distributions. From this we conclude that this effect may also be common in large-scale heterogeneous web services.

While it is intuitively clear that heavy-tailed response times can cause long response times in fork-join systems, we are not aware of a detailed discussion of this effect and of possible remedies. We argue that there are web services not requiring an exhaustive service composition, i. e., a reasonably complete result is sufficient. We suggest that the performance of such services can be improved by

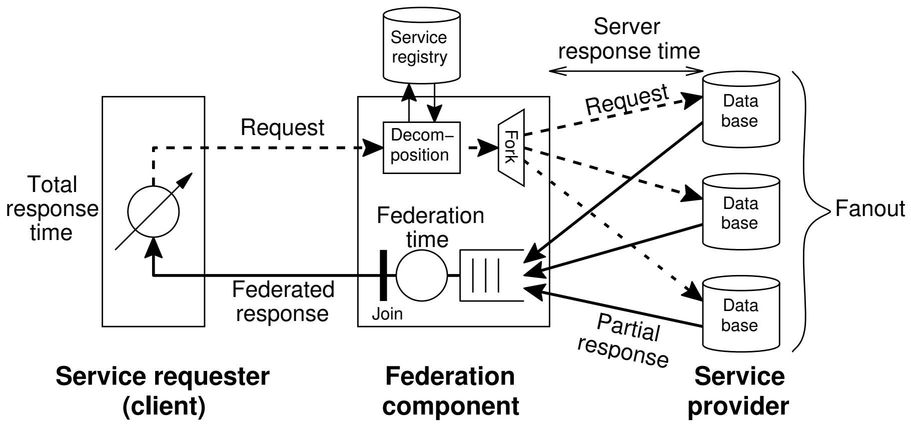

Figure 1: Data processing in a composite web service platform

considering only results from fast servers. Alternatively, timeouts can be used to avoid unnecessarily long waiting times. However, since in both cases the query result can be incomplete, the impact of setting a threshold or a timeout duration must be well understood. We present the methodology for analyzing the performance of such mechanisms and apply it to simple but realistic examples.

The rest of the paper is structured as follows: In Section 2, we model a system offering composite web services and discuss the main sources of delay. We present an exact and an approximative analysis for exponential and Pareto distributed response times in Sections 3 and 4, respectively. In Section 5, we study strategies to improve the response time. Finally, Section 6 concludes the paper.

2 SYSTEM MODEL

2.1 Architecture of a web services platform

Our research is motivated by the vision that future location-based services will be based on large-scale models of the real world, which are provided by many different providers and thus have to be federated by a web service middleware [5]. The system architecture of such a service platform is shown in Fig. 1: A service requester, e. g., a mobile terminal, can perform requests to a middleware. This so-called *federation component* provides uniform access to information offered by different service providers. To announce their services, the providers publish descriptions to a service registry.

As discussed in [5], such systems are unlikely to be realized by a single database if third-party content providers are involved. In the example of location-based services, this could mean that there are various providers offering information, e. g., about points of interest. The data is stored in databases and can be accessed by XML-based protocols. When a request arrives, the federation component determines with help of the service registry which servers store corresponding data. Then, the request is decomposed into sub-queries to each server (*fork*). Thereafter, the partial results of the sub-queries are collected and integrated into a single consistent result (*join*), which is sent back to the client.

Currently, such service platforms do not exist at large scale, but this may change in the future, e. g., because of the increasing availability of sensor information. This raises the question how scalable such an architecture is, in particular, if the number of content providers increases and thus many servers have to be queried. We label the number of servers invoked in parallel as the *fanout factor* N. The main performance metric of such a system is the response time T of the federation component, which is composed of two main sources of delay: First, the response times R_i ($1 \leq i \leq N$) of the databases, which also include processing and transmission delays. The performance of databases depends on many different factors and is hard to model. As a consequence, we characterize the database response time by its distribution function only.

The second main source of delays is the federation component. Each incoming response has to be processed and merged to the overall response. For complex data structures this join operation can require significant processing. We model the composition by a single queue with service time C_i. The federation component can only reply to the client's request if all partial responses have been received and processed. We assume that the join processing can start when the first partial response has arrived. Note that another solution would be to wait until all partial responses have been received.

This model focuses on one transaction only and thus does not incorporate resource contention caused by many clients to be served in parallel. We assume these effects to be modeled by the distribution functions. For simplicity, we do not consider analyzing the request and querying the service registry. Modeling the federation process by a single queue also neglects that this could partially be done in parallel. Splitting the problem to different service units may reduce the response time compared to a sequential processing [2]. Alternatively, such a system could also be studied by modeling each of the components by M/G/1-queues, as it is done e. g. in [4]. However, the response time of fork-join queuing networks is difficult to obtain even for exponential service times [2].

2.2 Federation response time

In the following, we assume that all servers have the same statistical characteristics: The response times R_i of the servers are independent and identically distributed with a common cumulative distribution function $F_R(t)$, the density function $f_R(t)$, and the mean r. This assumption is reasonable in rather homogeneous systems with more or less equally loaded servers. In principle, the model presented in the previous section could also be studied for different distributions for each R_i. Then, the federation response time is likely to be dominated by the slowest server. Such an example is analyzed for exponential response times in [1, 3]. However, the general case of different server characteristics is difficult to handle analytically and therefore left for further studies.

As explained, the federation component has to wait for all N servers. The maximum of the waiting (or synchronization) time is thus given by $\max(R_1, \ldots, R_N)$. Since the distribution function of the maximum R_{max} of N i. i. d. random variables is $F_{R_{\text{max}}}(t) = \prod_{i=1}^{N} F_R(t) = F_R(t)^N$, it can easily be seen that the mean waiting time for all servers is

$$W = \mathrm{E}[R_{\text{max}}] = N \int_0^\infty t\, F_R(t)^{N-1}\, f_R(t)\, \mathrm{d}t. \qquad (1)$$

For simplicity, we also assume the processing delays C_i of the partial responses in the federation component to be i. i. d. with distribution $F_C(t)$ and mean c. Since the total processing time is $C_{\text{sum}} = \sum_{i=1}^{N} C_i$, the mean the federation processing time would be

$$F = \mathrm{E}[C_{\text{sum}}] = N \cdot c \qquad (2)$$

if all responses arrived in the middleware at once. In order to normalize the values, we introduce $\kappa = \frac{r}{c}$, which can be interpreted as the relative "speed" of the federation component compared to the servers. With κ, the federation processing time can be expressed as $F = \frac{N}{\kappa} \cdot r$.

Given the fact that the join operation can start when the first partial responses have arrived, the following inequality must hold for the total response time T:

$$\max(W, F) \le T \le W + F. \qquad (3)$$

The extreme case $T_{\text{max}} = W + F$ would occur if the federation component first had to wait for all partial responses. The exact value of T depends on the distributions $F_R(t)$ and $F_C(t)$. As will be shown in the following sections, a quite accurate approximation is

$$T \approx \hat{T} = \max\left(W + c, w + F\right), \tag{4}$$

where $w = \text{E}[R_{\text{min}}] = N \int_0^\infty t\, \left(1 - F_R(t)\right)^{N-1} f_R(t)\, \text{d}t$ refers to the minimum of the N server response times. This estimation can be motivated by two extreme cases: For a quite fast federation processing, i. e., $\kappa \gg N$, the queue is empty most of the time. Therefore, the last response arriving after W can immediately be processed, resulting in a mean total delay of $W + c$. In contrast, if $\kappa \ll N$, the federation is rather slow and thus all requests get queued. As the first response arrives after w, the mean federation response time is $w + F$.

The federation response time T is likely to be larger than the mean server response time. We define a *federation slowdown factor* $S = \frac{T}{r}$ that quantifies the increase of the response time compared to an average single server. In the following, we study it for two examples: The exponential and the Pareto distribution. However, the methodology can be applied to other distributions, too.

3 EXPONENTIAL RESPONSE TIMES

In this section, we calculate the federation response time T_{exp} analytically and numerically for the case that R and C are exponentially distributed: $F_R(t) = 1 - \text{e}^{-\lambda t}$ and $F_C(t) = 1 - \text{e}^{-\nu t}$ for $t \geq 0$. Thus, the mean server response time is $r = \frac{1}{\lambda}$, and the mean federation processing time is $c = \frac{1}{\nu} = \frac{r}{\kappa}$.

3.1 Analytical model

The mean of the sum of N exponential distributions is $\frac{1}{\lambda} H_N$, where $H_N = \sum_{i=1}^{N} \frac{1}{i}$ is the N-th harmonic number (see e. g. [1]). The waiting time is thus

$$W_{\text{exp}} = r\, H_N = r \sum_{i=1}^{N} \frac{1}{i}. \tag{5}$$

Since the N-th harmonic number can be approximated by $H_N \approx \ln N + \gamma + \frac{1}{2\,N} - \frac{1}{12\,N^2} + \dots$, the waiting time increases logarithmically with N. $\gamma \approx 0.5772$ is the Euler-Mascheroni constant.

T_{exp} can be determined by using a continuous time Markov chain with the methodology that has been used in [3]. We define the Markov chain as follows: The state (i, j) indicates that i responses are queued in the federation, and j servers are still to respond. As illustrated in Fig. 2, the transition rate from state (i, j) to $(i+1, j-1)$ is then $j\,\lambda$. Also, transitions from (i, j) to $(i-1, j)$ occur with rate ν. From this follow the flow equilibrium equations for $0 \leq i \leq N$, $0 \leq j \leq N - i$, and $(i, j) \neq (0, 0)$

$$(a_1 + \kappa\, a_2)\, \pi(i, j) = a_3\, \pi(i-1, j+1) + \kappa\, a_4\, \pi(i+1, j) \tag{6}$$

with the special cases $N\, \pi(0, N) = \kappa\, \pi(1, 0)$ and $\pi(0, 0) = 0$, and the coefficients

$$a_1 = j \qquad a_2 = \begin{cases} 1 & i > 0, \\ 0 & \text{else} \end{cases}, \qquad a_3 = \begin{cases} j+1 & i > 0 \\ 0 & \text{else} \end{cases}, \qquad a_4 = \begin{cases} 1 & i + j < N \\ 0 & \text{else} \end{cases}.$$

A steady-state exists if we assume that a new request arrives immediately after a response has been completed. By further considering $\sum_{i=0}^{N} \sum_{j=0}^{N-i} \pi(i, j) = 1$, we can solve for the steady-state probabilities $\pi(i, j)$ by recursion, starting from $\pi(0, N)$. Since in every cycle there is only one federation

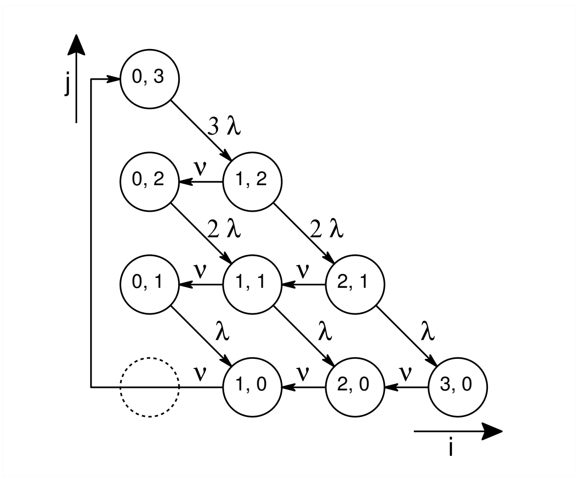

Figure 2: Markov chain for the case $N = 3$

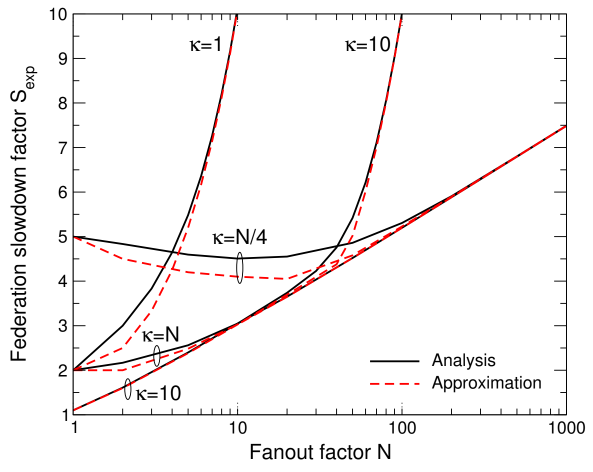

Figure 3: Expon. distr. resp. times with diff. κ

process in execution, the average response time can be obtained by applying Little's law (see [3]):

$$T_{\text{exp}} = \frac{1}{\nu \cdot \pi(1,0)} = \frac{\sum_{i=0}^{N} \sum_{j=0}^{N-i} \pi(i,j)}{\nu \cdot \pi(1,0)}. \tag{7}$$

In the latter expression, one can assume an arbitrary start value $\pi(0, N) \neq 0$. Some results for small fanout factors N are

$$\begin{aligned} N = 1: &\quad T_{\text{exp}} = r\left(1 + \tfrac{1}{\kappa}\right), \\ N = 2: &\quad T_{\text{exp}} = r\left(\tfrac{3}{2} + \tfrac{2}{\kappa} - \tfrac{1}{1+\kappa}\right), \\ N = 3: &\quad T_{\text{exp}} = r\left(\tfrac{11}{6} + \tfrac{3}{\kappa} - \tfrac{5}{1+\kappa} + \tfrac{3}{2+\kappa} + \tfrac{2}{(1+\kappa)^2}\right). \end{aligned}$$

For larger N, it is more efficient to solve (7) numerically, or to use the approximation given by (4):

$$T_{\text{exp}} \approx \hat{T}_{\text{exp}} = r \cdot \max\left(H_N + \tfrac{1}{\kappa}, \tfrac{1}{N} + \tfrac{N}{\kappa}\right). \tag{8}$$

3.2 Numerical results

Figure 3 shows the slowdown factor $S_{\text{exp}} = T_{\text{exp}}/r$ as a function of the fanout factor N. Both the exact value and the estimated value according to (8) are plotted for different choices of κ. The approximation is rather accurate, except for the example of $\kappa = N/4$, which lies between the two extreme cases that have been used to derive the approximation. Also note that the response time may even slightly decrease for larger N. In general, the federation "speed" κ must be scaled with the fanout factor N because otherwise the federation component becomes a bottleneck. Figure 3 also illustrates that there is a lower bound for T_{exp} given by H_N, which increases logarithmically with N.

3.3 Evaluation and discussion

The analysis in the previous section reveals that a large fanout increases the response time because of synchronization delays. However, in reality the response time of databases is unlikely to be exponentially distributed. For instance, the response time must always be larger than a minimum $k > 0$, which could be incorporated in the model by using a shifted negative-exponential distribution for R:

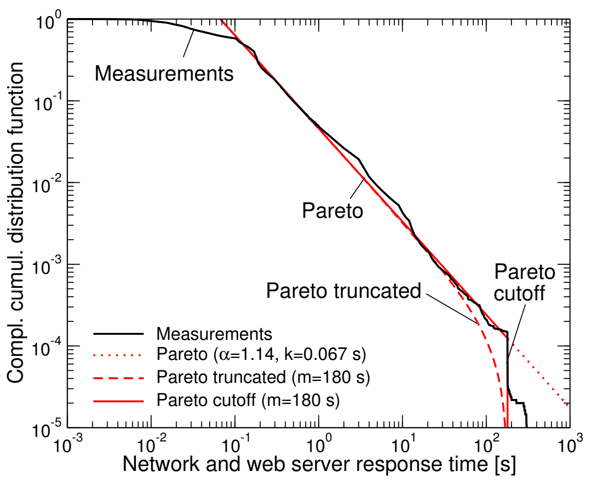

Figure 4: Measured web server response time

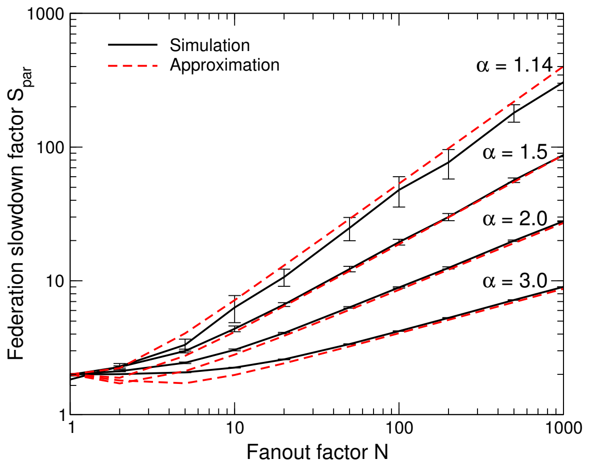

Figure 5: Pareto distr. with different tail factors α

$F_R(t) = 1 - \exp\left(-\frac{x-k}{\lambda}\right)$ for $x \geq k$. This means that k has to be added to T_{exp}. Another choice could be normal distributions: If there are a number of different tasks to be performed, each with a random delay, the total delay is likely to asymptotically have a normal distribution [6].

4 HEAVY-TAILED RESPONSE TIMES

In this section we show that the federation of different servers is much more challenging if server response times are heavy-tailed, i. e., if single values are likely to be orders of magnitudes higher than the mean. Motivated by measurement results, we use the Pareto distribution function

$$F_R(t) = \begin{cases} 0 & t \leq k \\ 1 - \left(\frac{k}{t}\right)^\alpha & t > k \end{cases}, \tag{9}$$

which has an infinite variance for $\alpha < 2$ and is then heavy-tailed.

4.1 Measurement results for server response times

There is some empirical evidence for heavy-tailed response time distributions in real systems, in particular, if they are heterogeneous. For instance, the response time of web servers, including network delay, is typically heavy-tailed. This is illustrated by Fig. 4, which presents measurement results obtained from monitoring HTTP traffic[1]. The measurements show that the web server response time, i. e., the delay between the first TCP segment of the request and the first TCP segment of the response, can be described by a Pareto distribution, at least up to a response time of about 180 s. Better models using a *cutoff* or *truncation* will be discussed in Section 5.2. The best-fit parameterization for the distribution tail is $\alpha = 1.14$ and $k = 0.067$ s, corresponding to a mean of $r = k\frac{\alpha}{\alpha-1} = 0.55$ s.

A similar study [7] reports that file transmission times in the WWW are Pareto distributed with shape factor $\alpha \approx 1.2$, up to several hundreds of seconds. A possible explanation is that the file sizes are heavy-tailed. Recent measurements on e-commerce traffic [8] reveal heavy-tailed server response times with $\alpha \approx 1.55$, which is attributed to the burstiness of arriving requests. We conclude from this that heavy-tailed response times should be considered in the dimensioning of web service platforms.

[1]The measurements have been performed in a student dormitory access network at the University of Stuttgart in November 2000 and include about 60 million IP-packets of more than 200 end users.

4.2 Approximations

For Pareto distributed server response times R_i, the waiting time W_{par} can be obtained explicitly from the maximum of N random variables, too. Even more generally, the i-th moment of the j-th *order statistic* (see Section 5.1) of N i. i. d. Pareto random variables is known to be [9]:

$$\mu_{i,j} = \frac{\Gamma(N+1)\,\Gamma(N-j+1-\frac{i}{\alpha})}{\Gamma(N+1-\frac{i}{\alpha})\,\Gamma(N-j+1)}\,k^i. \tag{10}$$

Here, we consider the mean value of the maximum, i. e., $\mu_{1,N}$. Eq. (10) can be simplified by approximating the gamma function $\Gamma(x)$ as follows: $\Gamma(x+1) = x! \approx \left(\frac{x}{e}\right)^x \sqrt{2\,\pi\,x}$ for $x \gg 1$. Thus, $\frac{\Gamma(N+1)}{\Gamma(N+1-\frac{1}{\alpha})} \approx N^{1/\alpha}$ for $N \gg 1$. Also, $\Gamma(x) \approx \frac{1}{x} + \gamma(x-1)$ for $x \in (0,1]$. Applying this to (10) gives

$$W_{\text{par}} \approx \hat{W}_{\text{par}} = r \cdot N^{1/\alpha}\left(1 - \gamma\,(\alpha-1)\,\alpha^{-2}\right). \tag{11}$$

By neglecting $\gamma\,(\alpha-1)\,\alpha^{-2}$, this expression can be further simplified to $\hat{W}_{\text{par}} \approx r \cdot N^{1/\alpha}$. Since W_{par} scales with $N^{1/\alpha}$, the tail shape α has a very significant impact: If α is close to 1, the mean waiting time can be significantly larger than r, even for a small number of servers N.

The mean value of the minimum can be obtained from (10) for $j=1$: $w_{\text{par}} = \frac{N}{N-1/\alpha}\,k$. Inserting in (4) yields to the following approximation for T_{par}:

$$T_{\text{par}} \approx \hat{T}_{\text{par}} = \max\left(\hat{W}_{\text{par}} + \tfrac{r}{\kappa}, \tfrac{r\,(\alpha-1)}{\alpha-N^{-1}} + \tfrac{r\,N}{\kappa}\right). \tag{12}$$

4.3 Numerical results

The federation response time T_{par} can be obtained by simulation of the model shown in Fig. 1. Unless explicitly otherwise stated, we assume in the following $\alpha = 1.14$, $\kappa = N$, and exponential distributed federation processing times C. Other distributions for C, such as a constant value, do not have a significant impact on the results.

From Fig. 5 follows that the approximation $\hat{T}_{\text{par}}$ given by (12) is quite accurate. Due to long synchronization delays, the federation slowdown factor $S_{\text{par}} = T_{\text{par}}/r$ varies over several orders of magnitude, in particular for $\alpha = 1.14$. For instance, a fanout factor of about 20 is sufficient to increase the federation response time by one order, i. e., $T_{\text{par}} > 6\,\text{s}$. The larger α is and the faster the distribution tail drops, the less dominant is the impact of the synchronization. Nevertheless, we conclude from Fig. 5 that in a system with α close to 1 it is hardly feasible to perform large-scale service composition, if the response time is not improved by other means.

5 IMPROVING RESPONSE TIME BY INCOMPLETE RESULTS

In this section we propose and analyze two strategies to reduce the response time of large-scale composite web services. We focus on solutions that do not require speeding up the servers since this might not be an option in a multi-provider scenario. In the first approach, the federation component does not wait for slow servers, i. e., it only considers responses of a part of the databases. Obviously, it depends on the offered service whether such an incomplete result is acceptable. In the example of location-based services, it might be sufficient to provide information about *most* points of interest in a certain area. Thus, if some results can only be obtained by waiting for responses from very slow servers, it might be better to construct the federated response earlier. Alternatively, long response times can be addressed at the servers by a monitoring component that sends an error message if no result can be provided within a certain time. After this timeout, the federation component does not have to wait for this server any more. The two approaches differ in that the first mechanism is deployed in the middleware, while the latter one must be implemented in each server.

5.1 Restriction to fast servers

There are two ways to implement a limitation to fast servers in a federation component: First, the middleware could set a single timeout M and send the response based on the information that has been received before the timeout expires. This limits the federation response time to $T \leq M$. Alternatively, it could suppose the federated response to be complete if a predefined number of responses have been received and processed. Then, the response time depends on the portion of servers q which must have answered in order to get a sufficient result.

The performance of the latter mechanism can be studied by the theory of *order statistics* [10]: If $R_1, \ldots, R_N$ are an i. i. d. sample from a distribution with density $f_R(t)$, and if they are ordered from lowest to highest value so that $R_{(1)} \leq \cdots \leq R_{(N)}$, the variable $R_{(j)}$ is called the j-th order statistic. The sample maximum is a special case with $j = N$. The density function of the j-th order statistic is

$$f_{R_{(j)}}(t) = N \binom{N-1}{j-1} f_R(t)\, F_R(t)^{j-1} \, (1 - F_R(t))^{N-j} \quad \text{for} \quad 1 \leq j \leq N. \tag{13}$$

The mean waiting time W' of the q percent fastest servers out of N is thus the mean of the order statistic for $j = \lceil q \cdot N \rceil$. Note that at least one server can be omitted only if $q \leq 1 - N^{-1}$.

It can be shown that the asymptotic distribution of quantiles of order statistics is normal when N and j tend to infinity but the ratio $q = j/N$ remains constant [10]. The quantile mean Ω is given by

$$F_R(\Omega) = q, \quad \text{i. e.,} \quad \Omega = F_R^{-1}(q). \tag{14}$$

As a consequence, independent of the distribution $F_R(t)$, the mean response time of a certain quantile is finite even for $N \to \infty$. Ω is also a quite good approximation for W' as long as $q \leq 1 - N^{-1}$.

For the Pareto distribution, we can use $\Omega = k\,(1-q)^{-1/\alpha}$ to get an approximation for W'_{par}:

$$W'_{\text{par}} \approx \hat{W}'_{\text{par}} = \begin{cases} k\,(1-q)^{-1/\alpha} & q \leq 1 - N^{-1} \\ r \cdot N^{1/\alpha}\,(1 - \gamma\,(\alpha-1)/\alpha^2) & \text{else} \end{cases}, \tag{15}$$

Therein, the result from (11) is used if q is too large and thus the federation component must wait for all servers. Since only $\lceil q\,N \rceil$ responses have to be processed, eq. (12) can be modified as follows:

$$T'_{\text{par}} \approx \hat{T}'_{\text{par}} = \max\left(\hat{W}'_{\text{par}} + \tfrac{r}{\kappa}, \tfrac{r\,(\alpha-1)}{\alpha - N^{-1}} + \tfrac{r\,\lceil q\,N \rceil}{\kappa}\right). \tag{16}$$

Figure 6 illustrates the effect of incomplete results: For any $q < 1$, the response time T'_{par} approaches a finite limit even if the fanout is very large. The steps are due to the condition that $j = \lceil q \cdot N \rceil$ must be integer. For $\alpha = 1.14$, a limitation to $q = 95\,\%$ significantly improves the response time (for $N \geq 20$). Also, the response time T'_{par} is quite close to the approximated $\hat{T}'_{\text{par}}$.

5.2 Server-based timeouts

Another way to reduce response times are server timeouts after the time m. A fixed timeout duration results in a *cutoff* of the response time distribution function. Such an effect can be observed at

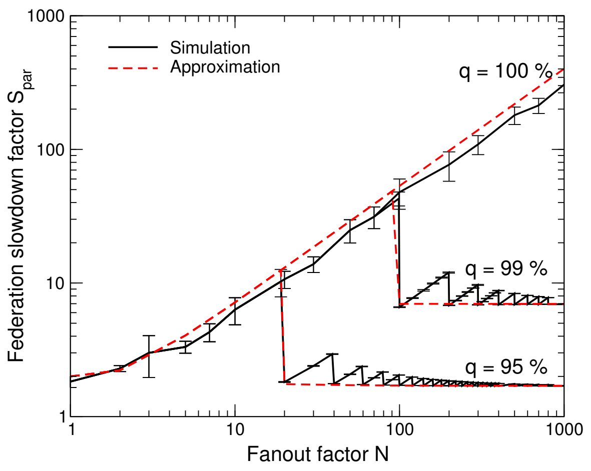

Figure 6: Effect of restriction to fast servers

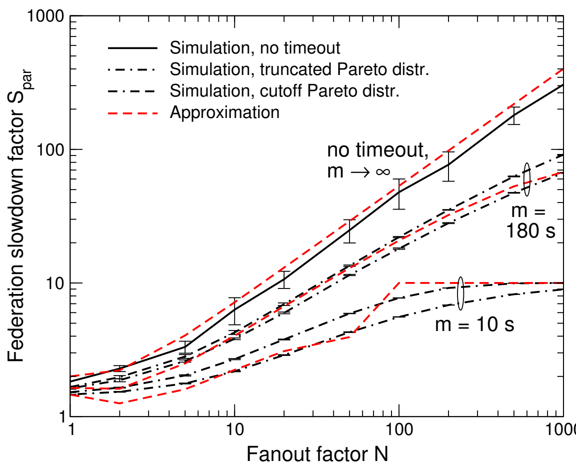

Figure 7: Effect of timeouts in the servers

$t = 180$ s in the measurement of HTTP traffic in Fig. 4. Thus, timeouts can be modeled by

$$F''_R(t) = \begin{cases} G(t) & t \leq m \\ 1 & t > m \end{cases}, \tag{17}$$

with $G(t) = F_R(t)$. The probability that the server timeout does not expire is $q = F_R(m)$. For analytical studies, it can also be beneficial to consider a *truncation* of the distribution with $G(t) = F_R(t)/F_R(m)$, because this avoids the Dirac impulse in the density function. Furthermore, timeout durations could be calculated dynamically; then $F''_R(t)$ would depend on the timeout implementation. In general, the ratio of successfull responses is given by

$$q = 1 - \int_0^\infty \frac{F''_R(t) - F_R(t)}{1 - F_R(t)} f_R(t)\,\mathrm{d}t. \tag{18}$$

Once again, we will study Pareto distributions as an example. The i-th moment of the j-th order statistic of N truncated Pareto distributions can be expressed by a hypergeometric function [9]:

$$\mu''_{i,j} = k^i\,{}_2\mathrm{F}_1\left(\tfrac{i}{\alpha},\, j;\, N+1;\, 1 - \left(\tfrac{k}{m}\right)^\alpha\right). \tag{19}$$

By using a series expansion of ${}_2\mathrm{F}_1(a,\, b;\, c;\, z)$ around $z = 1$, which is valid for a rather long timeout duration m, and applying again some approximations for the gamma function, (19) can be used to obtain the following formula for the maximum waiting time:

$$W''_{\mathrm{par}} \approx \hat{W}''_{\mathrm{par}} = \begin{cases} r\left(N^{1/\alpha} - N\left(\frac{k}{m}\right)^{\alpha-1}\right) & \text{for } N < \alpha^{-\alpha/(\alpha-1)}\left(\frac{m}{k}\right)^\alpha \\ m & \text{else} \end{cases}. \tag{20}$$

For a shorter timeout, i. e., when m is of the order of W_{par}, it is obvious that W''_{par} is rather close to m. Then, eq. (20) corresponds directly to (15). $\hat{T}''_{\mathrm{par}}$ can be obtained by inserting (20) in (16).

The simulation results in Fig. 7 show that even a rather large timeout m such as 180 s can significantly improve the federation response time T''_{par} because of the missing tail of the distribution. Then, T''_{par} is quite well approximated by $\hat{T}''_{\mathrm{par}}$. A smaller timeout, e. g., $m = 10$ s, limits the federation response time to this value, but the approximation $W''_{\mathrm{par}} \approx m$ is only asymptotically correct. Furthermore, the difference between the *cutoff* and *truncation* modeling is more significant for smaller m. Nonetheless, $\hat{T}''_{\mathrm{par}}$ can give a good estimation how timeouts improve the federation response time.

5.3 Other alternatives

There are also other possibilities to address the problems discussed in this paper, but they require more detailed system models. For example, the number of servers to be queried could be limited by a threshold: Only if the result of the first query is not sufficient, the scope will be extended to further servers. Also, caching of results could improve performance if this avoids queries to slower servers.

6 CONCLUSION

Composite web services are based on fork-join operations. Since their completion time is dominated by the slowest service unit, the tail of the server response time distribution significantly affects the performance of a web service middleware. This paper shows at the example of exponential and Pareto distributed response times that this may be a problem for large-scale service composition. We provide simple and accurate approximations to quantify the impact on the response time of systems where all servers have the same characteristics. A possible solution could be to consider only results from faster servers, which implies a trade-off between performance and response completeness. We show that omitting a few slow servers (such as 5 %) can significantly improve the response time. Our results can be used to derive dimensioning guidelines for large-scale composite web service platforms.

Acknowledgments

This work is funded by the German Research Foundation (DFG) through the Center of Excellence (SFB) 627. The author would like to thank Bernd Gloss, Marc Necker, Andreas Reifert and Gert Willmann for many fruitful discussions and Johannes Färber for providing the measurement results used in Fig. 4.

References

1. D. A. Menascé, “Response-time analysis of composite web services,” *IEEE Internet Computing*, vol. 8, no. 1, pp. 90–92, 2004.
2. R. Nelson and A. N. Tantawi, “Approximate analysis of fork/join synchronization in parallel queues,” *IEEE Trans. on Computers*, vol. 37, no. 6, pp. 739–743, 1988.
3. D. A. Menascé, D. Saha, S. C. da Silva Porto, V. A. F. Almeida, and S. K. Tripathi, “Static and dynamic processor scheduling disciplines in heterogeneous parallel architectures,” *Journ. of Parallel and Distr. Comp.*, vol. 28, pp. 1–18, 1995.
4. B. Cahoon, K. S. McKinley, and Z. Lu, “Evaluating the performance of distributed architectures for information retrieval using a variety of workloads,” *ACM Trans. Inf. Syst.*, vol. 18, no. 1, pp. 1–43, 2000.
5. D. Nicklas, M. Großmann, T. Schwarz, S. Volz, and B. Mitschang, “A model-based, open architecture for mobile, spatially aware applications,” in *Proc. 7th Intl. Symp. on Spatial and Temporal Databases*, Redondo Beach, CA, USA, 2001.
6. V. S. Adve and M. K. Vernon, “The influence of random delays on parallel execution times,” *SIGMETRICS Perform. Eval. Rev.*, vol. 21, no. 1, pp. 61–73, 1993.
7. M. E. Crovella and A. Bestavros, “Self-similarity in World Wide Web traffic: Evidence and possible causes,” *IEEE/ACM Trans. Netw.*, vol. 5, no. 6, pp. 835–846, 1997.
8. U. Vallamsetty, K. Kant, and P. Mohapatra, “Characterization of e-commerce traffic,” *Electronic Commerce Research*, vol. 3, no. 1-2, pp. 167–192, 2003.
9. N. L. Johnson, S. Kotz, and N. Balakrishnan, *Continuous Univariate Distributions*, 2nd ed., John Wiley&Sons, 1994, vol. 1.
10. M. G. Kendall and A. Stuart, *The Advanced Theory of Statistics*, 2nd ed., Charles Griffin, 1963, vol. 1.