Collaborative Ranking of Grid-enabled Workflow Service Providers

Domenico Laforenza  
ISTI – CNR  
Via G. Moruzzi, 1  
Pisa, Italy  
d.laforenza@isti.cnr.it

Franco Maria Nardini  
ISTI – CNR  
Via G. Moruzzi, 1  
Pisa, Italy  
f.nardini@isti.cnr.it

Fabrizio Silvestri  
ISTI – CNR  
Via G. Moruzzi, 1  
Pisa, Italy  
f.silvestri@isti.cnr.it

ABSTRACT
Service Oriented Architecture (SOA) and Grid computing are very hot research topics, nowadays. While Grid computing is aimed at sharing dynamically heterogeneous resources, SOAs is a meta-architectural style that enable business flexibility in an interoperable way. There is a growing consensus that SOA(s) and Grid(s) might be beneficial to each other. In Grid-based SOAs a central role is played by tools for the publishing/discovering of services (resources).

This work presents SPRanker (Service Provider Ranker): a service discovery tool that is able to retrieve providers from partially specified service descriptions. It ranks providers on the basis of an Information Retrieval-based score formula that takes into account judgments expressed collaboratively by past service users.

Categories and Subject Descriptors
C.4 [Performance of Systems]: Measurement Techniques; D.2.10 [Software Engineering]: Design

General Terms
Performance, Design, Algorithms.

Keywords
SLA, QoS, SOA, Grid, Information Retrieval, Vector Space Model, Collaborative Ranking, Service Discovery, Service Ranking.

1. INTRODUCTION
Modern approaches to QoS within SOAs adopt Service Level Agreements (SLAs) as a way to define constraints customers and providers must satisfy. To this extent, SLAs represent the way out to the use of best effort as the strategy for selecting service providers.

Our ranked discovery service implements a novel ranking schema based on solid Information Retrieval theory, namely the Vector Space Model by considering historical information on expired SLAs: the ranking score is, in fact, based on the assumption that performance (in terms of QoS) of providers should be evaluated collaboratively considering users’ feedback.

The Vector Space Model [2] [3] [1] consists in representing an object as a vector in \( \mathbb{R}^n \) where each dimension corresponds to a separate term. If a term occurs in the object, its value in the corresponding vector entry is non-zero.

SLAs are modeled as vectors in \( \mathbb{R}^n \), where \( n \) is the number of possible SLA terms. To keep the model as simple as possible we consider only unit vectors. The normalization is done in a way that all the vector coordinates will range between 0 and 1/n.

Queries in SPRanker are called Query-SLA. Two definitions follows:

**SLA-Vector:** An SLA-vector is defined as a unit vector \( \vec{s}_p \otimes T = (s_1, \ldots, s_n) \) representing a successfully ended SLA issued by provider \( p \) of service \( S \) at time \( T \). Each \( s_i, i \in [1..n] \), is the value associated to term \( T_i \) of the SLA template. For example, \( \vec{s}_p \otimes T = (2, 1TB, 0.04\$) \) could represent an SLA of a service provided at time \( T \) running on a 2-way SMP with 1TB free disk space, and at the cost of 0.04\$.

**Query-SLA:** A Query-SLA is a unit vector \( \vec{q}_q \) in \( \mathbb{R}^n \), where each \( q_i, i \in [1..n] \) can assume a reference value for term \( T_i \) of the SLA template. \( \circ \) meaning that we do not want to take into account the \( i \)-th term, and \( \bullet \) meaning that the \( i \)-th term may assume any value between 0 and 1/n.

Let \( \vec{q}_q = (q_1, \ldots, q_n) \) be a Query-SLA. Let \( S \) be the set of SLA-vectors, and \( \vec{r}_p \otimes T = (s_1, \ldots, s_n) \) representing an SLA successfully issued at time \( T \), by provider \( p \), on service \( S \).

We define a similarity function (1) to include provider, and service name and to take into account the SLA issue time. We define \( \text{sim} = 0 \) if either the provider, or the service name differs. Otherwise:

\[
\text{sim} (\vec{q}_q, \vec{r}_p \otimes T) = \Delta \tau^{-1} \sum_{i=1}^{n} q_i s_i \tag{1}
\]

where \( \Delta \tau \) is a (not necessarily linear) function in \([0, 1]\) weighting the elapsed time since when the SLA was issued, i.e. \( T \). In our SPRanker implementation we set \( \Delta \tau^{-1} = e^{-\tau} \), where \( \tau \) is the number of elapsed days.

**Request for provider’s performance.** Queries of the form \( \vec{q}_q = (\bullet, \ldots, \bullet) \) are for seeking information on provider \( p \) regardless of the services they provide. This kind of queries
are solved computing the following formula:

$$score(\vec{q}_p^*) = \frac{1}{|A|} \sum_{(S,T) \in A} sim(\vec{q}_p^*, \vec{s}_p^* @ T)$$

where $A = \{(S, T) | S \in S, T \in Time\}$ is the set of all possible pairs: service name, issue time. The closer the score value to 1 the higher the importance of the provider.

Request for best performing providers of a service.

Queries of the form $\vec{q}_p^*$ seeks a list of providers offering service $S$ ordered by similarity with the query. The algorithm follows:

```
Input: $\vec{q}_p^*$
Output: A list of service provider ranked according to the similarity score

1. foreach $p$ in $P$ (the list of all providers) do
2.   foreach time $T$, which $S$ has been provided by $p$ do
3.     $s = sim(\vec{q}_p^*, \vec{s}_p^* @ T)$;
4.     rankedProviderList[$p$] += $s$;
5. endforeach
6. endforeach
7. sort rankedProviderList according to scores;
8. return the providers according to the sorted scores;
```

Figure 1: Ranking of service providers according to users’ queries.

2. SPRANKER ARCHITECTURE

The architecture of SPRanker is composed by three modules: the Gatherer, the Indexer, and the Query Server.

The Gatherer aims at collecting data from (positively) expired SLAs. We only consider positively expired SLAs because we want to discriminate between good and bad service provisioning, and because we want to enforce satisfied customers to not incur in unreal bad judgements from malicious partners (either clients, or customers) willing to lower a provider’s score. The Gatherer can act in two different modes: push-based, and pull-based.

When in push-based modality, the gatherer receives SLAs directly from providers and customers. Pull-based mode, instead, is used to periodically poll known providers for up-to-date information.

The Indexer is used to transform SLAs collected by Gatherer into a machine readable format.

The Query Server has been implemented as a Web Service. It offers two distinct methods. One for each kind of query SPRanker answers.

3. EXPERIMENTS

Due to the lack of real-world test data, conducting experiments on service discovery mechanisms is a very difficult task. Therefore, we were forced to opt for the use of a synthetic SLA generator providing successful transactions between customers and providers with randomly assigned parameters.

We created a SLA repository referring to a single service supplied by five providers. The experimental phase consists of two different steps. In the first step we generated eight SLAs belonging to the five different providers. By using a Query SLA we scored each provider according to Algorithm 1. To evaluate the impact of the SLAs on the global score, we add SLAs to the repository of the four providers which did not receive the maximum score in the first evaluation round. Then we compare the score variations with the first round maximum score.

The plot in Figure 2 shows the score obtained by each provider by varying the number of SLAs for the four providers ($\text{Provider}2, \ldots, \text{Provider}5$). $\text{Provider}1$ lose the first position only after five SLAs were added to the initial repository. We show that in order to obtain a higher score is not sufficient to have a higher number of SLAs. Providers must in fact, have a higher number of SLAs recently submitted, and having a high similarity with the Query SLA.

4. CONCLUSIONS

In this paper we have introduced a service discovery tool that can be used to rank providers with respect to a Query SLA that, according to their historical performance, evaluate the similarity between a Query SLA and the providers.

5. ACKNOWLEDGMENTS

This research has been partially funded by the IST FP7 European Project S-Cube Grant Agreement no. 215483 and by the italian FIRB INtegrated SYstem for EMERgency (INSEME).

6. REFERENCES

