Volume:04 Issue:02 Pages: 1534-1543 *(2012) ISSN : 0975-0290.*

Combinatorial Optimization of QoS Service Architecture

Akash K Singh, PhD
IBM Corporation, Sacramento, USA
Email : akashs@us.ibm.com

ABSTRACT

In this era of Cloud Computing and Service Architecture, it is expected that web services will proliferate the current market and business will expose their web service's as B2B, C2B, E2E and C2C service componets, many web services offer the similar services, and the clients application will demand more value added services and informative services rather than those offered by single entity ,or isolated web services. Synthesis of Service Semantics with high quality is prominent and in high demand, Internet based applications share webservices and offer services as data pipelines and social computing. The client applications are experiencing cost and quality intensive competition to select service portfolio and create semantics service composition, among numerous possible plans, that satisfy their quality-of-service (QoS) requirements. Typical QoS attributes associated with a semantic fabric services are the consumption cost and time, availability of services, dynamic runtime service execution rate, service and entity reputation, and web trends of service frequency and performance payload support. This paper describes the service composition architecture and webservice composition algorithm based on combinatorial and Poisson distribution. As mathematical and implementation framework of the service algorithm maintains the high standard of service QoS composition in a business portfolio of service category and assists the business entity and client application to have better service selection and QoS of request and response.

Keywords- Web service composition; Quality-of-service (QoS); QoS-oriented composition algorithm; Combinatorial Optimization and Statistical Analysis; Web service composition architecture

Date of Submission : July 05, 2012 Date of Acceptance : August 11, 2012

I. INTRODUCTION

Cloud computing and Service-Oriented Architecture (SOA) provides a flexible systems framework for service composition and QoS attributes. Using standard-based protocols (such as SOAP and WSDL), composite services can be constructed by integrating systems atomic services or business functions that are independent. Algorithms are required to perform service selection with various QoS levels as per the contractual and context based performance attributes. This paper describes the Service Architecture to facilitate the best selection of QoS-based services and optimal path for execution. The objective of service selection is to maximize an application-specific business and resource consumption functions under precise QoS constraints. The combinatorial model defines the problem as a multi-dimension statistical distribution. Runtime dynamic selection of Web services is important for building modular and de-coupled service oriented applications. An abstract layer is proposed to describe the required services at design-time, and maps service offerings that are located at runtime. With the growing number of Web services that provide the same functionality but differ in quality parameters (e.g., availability, response time), a decision needs to be made on which services should be selected such that the user's end-to-end QoS requirements are satisfied. Although very efficient, local selection strategy fails short in handling global QoS requirements. Solutions based on global optimization, on the other hand, can handle global constraints, but their poor performance renders them inappropriate for applications with dynamic and realtime requirements. In this article we address this problem and propose a hybrid solution that combines global optimization with local selection techniques to benefit from the advantages of both worlds. The proposed solution consists of two steps: first, we use mixed integer programming (MIP) to find the optimal decomposition of global QoS constraints into local constraints. Second, we use distributed local selection to find the best Web services that satisfy these local constraints. The results of experimental evaluation indicate that our approach significantly outperforms existing solutions in terms of computation time while achieving close-to-optimal results. QoS-aware service composition is a key requirement in Service Oriented Computing (SOC) since it enables fulfilling complex user tasks while meeting Quality of Service (QoS) constraints. A challenging issue towards this purpose is the selection of the best set of services to compose, meeting global QoS constraints imposed by the user, which is known to be a NP-hard problem. This challenge becomes even more relevant when it is considered in the context of dynamic service environments. Indeed, two specific issues arise. First, required tasks are fulfilled on the y, thus the time available for services' selection and composition is limited. Second, service compositions have to be adaptive so that they can cope with changing conditions of the environment. In this paper, we present an efficient service selection algorithm that provides the appropriate ground for QoS-aware composition in dynamic service environments. Our algorithm is formed as a guided heuristic. The paper also presents a set of experiments conducted to evaluate the efficiency of our algorithm, which shows its timeliness and optimality.

With the development of the semantic web (Mao, 2010) and Web Ontology Language (OWL) (Martin et al., 2007; Hasany et al., 2010), we hope to utilize the semantic web technology to integrate intelligently all kinds of web services, thus producing semantic web services. The automatic composition of semantic web services is one of the key technologies in integrating web services and it is much concerned by many researchers form the beginning. The methods for web service composition are dependent on specific framework of web services in a certain degree. The typical framework of web services is OWL-S which is based on OWL and OWL is on the basis of description logic (Horrocks et al. 2003). Description logic is effective in representing and reasoning static knowledge. However, description logic cannot represent and reason dynamic knowledge such as behaviors and services. So, Dynamic Description Logic (DDL) (Shi et al., 2004) is presented by combining description logic ALC, dynamic logic and action theory. The description ability of DDL is over logic and system based on proposition language in describing actions. The problems about reasoning are all decidable in reasoning actions. All the problems about reasoning actions can be transformed to the satisfiable problems. So, they can be reasoned based on the decision algorithm in absence of information. In sum, DDL is more applicable for describing semantic web by combining knowledge of static field based on description logic and action knowledge of dynamic field.

At present, the service composition is mainly the following several methods:

- The service composition methods based on type matching of input and output parameters

A solution based on the DAML-S (Paolucci et al., 2002) is proposed. The overlap relationship of the types of parameters is studied and the partial matching algorithm is proposed (Li and Horrocks, 2003). For large number of services, a synthetic testbed (Constantinescu et al., 2004) is proposed that can be used for simulating large deployments of services and also for generating service composition problems.

- The service composition methods based on artificial intelligence planning

The pioneer of such approaches (McIlraith and Tran, 2002) takes web services as the actions of AI planning. Therefore, web service composition process is the process of generating planning. The adopted planning algorithm is improved based on logic program Golog. The service composition based on HTN (Hierarchical Task Network) planning algorithm (Sirin et al., 2004) is proposed.

As the description logic can effectively express and reason knowledge of static domain and cannot handle the dynamic knowledge such as actions and services, dynamic description logic DDL (Shi et al., 2004) is put forward and the service description method based on DDL (Shi and Chang, 2008) is proposed. And then the service composition algorithm based on DDL (Peng et al., 2008) is presented. But because the algorithm needs to enumerate all composition sequences, efficiency is difficult to be guaranteed.

In this study, DDL is taken as service description framework. For services which are performed in sequence, a fast and effective service composition algorithm is presented which takes full advantage of

DDL describing dynamic run of services. This method utilizes the partial order relationship between services dividing the service composition into two stages: on the first stage, the partial order diagram is constructed on the registration servicer; on the second stage fast service composition is implemented based on the partial order diagram to meet the requirement of users. To achieve the goal, First inclusion operation on sets of DDL is extended, the partial order relationship between services is defined based on extended inclusion operation and the satisfiability problem of DDL formula is transformed to operations between sets. Second, the fast composition algorithm is proposed based on DDL and the correctness and time complexity is analyzed. Finally, the algorithm of generating the partial order diagram between services is proposed and the time complexity of the algorithm is analyzed. By analysis and verification, the method of service composition designed in this study is implemented in linear time complexity greatly reducing response time of the system.

A. Semantics and Service Fabric DDLs

A Semantic Service Oriented Architecture (SSOA) is an architecture that allows for scalable and controlled Enterprise Application Integration solutions.[1] SSOA describes a sophisticated approach to enterprise-scale IT infrastructure. It leverages rich, machine-interpretable descriptions of data, services, and processes to enable software agents to autonomously interact to perform critical mission functions..

Web Semantics is a knowledge-intensive and intelligent service Web. These service areas include: knowledge technologies, ontology, agents, databases and the semantic grid, obviously disciplines like information retrieval, language technology, human-computer interaction and knowledge discovery are of major relevance as well. All aspects of the Semantic Web development are covered in this paper. This paper presents theories, methods and experiments from different business areas in order to deliver innovative semantic methods and applications.

Some definitions related to Semantics Services are given as follow:

Definition 1: An atomic Service action with request and response http arguments is the form:

$$P[C_j^{(n)} = k] = \frac{j^{-k}}{k!} \sum_{l=0}^{[n/j]-k} (-1)^l \frac{j^{-l}}{l!}$$

where, k is a atomic action, and has a finite sequence of all individual variables in n and k interface of the atomic action of events, P is a set of preconditions & postcondition which must be satisfied before a service request/ response action are performed.

Definition 2: Formulas in Abstract Service Layer is defined and generated as follows:

$$S_r = (n-rj)!1(rj \le n) \times \frac{n^{[rj]}}{j^r r!} \frac{1}{n!} = 1(rj \le n) \frac{1}{j^r r!}$$

where, j is an individual service component , r is an entity relationship and n is an action of events.

Definition 3: An action of Service descriptor is defined as follows:

$$\frac{\theta d}{n} \exp\left\{ \sum_{i \ge 1} [\log(1 + i^{-1}\theta d) - i^{-1}\theta d] \right\}$$

where, θ is an atomic action.

The service description in OWL-S and the transformation method from OWL-S description to DDL description are given in the following.

$$\sum_{r \ge 0} P[T_{0b} = r] \left\{ 1 - \frac{P[T_{bn} = n-r]}{P[T_{0n} = n]} \right\}_+$$

In OWL, the function of an service is described by Input, Output, Precondition and Effect (IOPE) which is denoted by S = {I, O, P, E}. Let S.I, S.O, S.P and S.E be respectively input, output, precondition formula set and effect formula set of service S. If α(V1,…, Vn) = (P, E) is an atomic action of DDL, S can be described as α = S, where the action name is service name. P = S.I ,S.P is the premise formula of actions which is the union of input formula set and precondition formula set of service S; E = S.O and S.E is the result formula of actions which is the union of output formula set and effect formula set of service S.

$$\left| \ d_{TV}(L(\tilde{C}[1,b]), L(\tilde{Z}[1,b])) - (n+1)^{-1}\sum_{r\geq 0} P[T_{0b}=r]\left\{\sum_{s\geq 0} P[T_{0b}=s](s-r)(1-\theta)\right\}_{+} \ \right|$$

II. METHODOLOGY

A. Service Composition

The service-oriented computing environment and supported infrastructure serves the high quality services and Orchestrated and standardized Web service technologies provide a promising solution for the integration of business applications that are using different technologies, this build the platform where software can be exposed as a service and business domain can be exposed as a services and there is no need to redevelop same business function in other projects, this is purely new value-added services that can leverage legacy application and infrastructure services. As the Organization are acquired and merged there is high demand to use each other IT & Business Capabilities so there is growing interest in the development of ad-hoc service composition in the areas of business domains such as Healthcare, Telecommunication, Insurance and Financial systems, as well as in web & game based rich multimedia applications. As we experience the web service growth that provides more or less same functionality but differ in performance attributes, quality service attributes, the service composition comes under play to aid the criteria of service selection and consumptions which are highly critical for business domain support model and adhere to non-functional SLA agreements and business portfolio contracts.

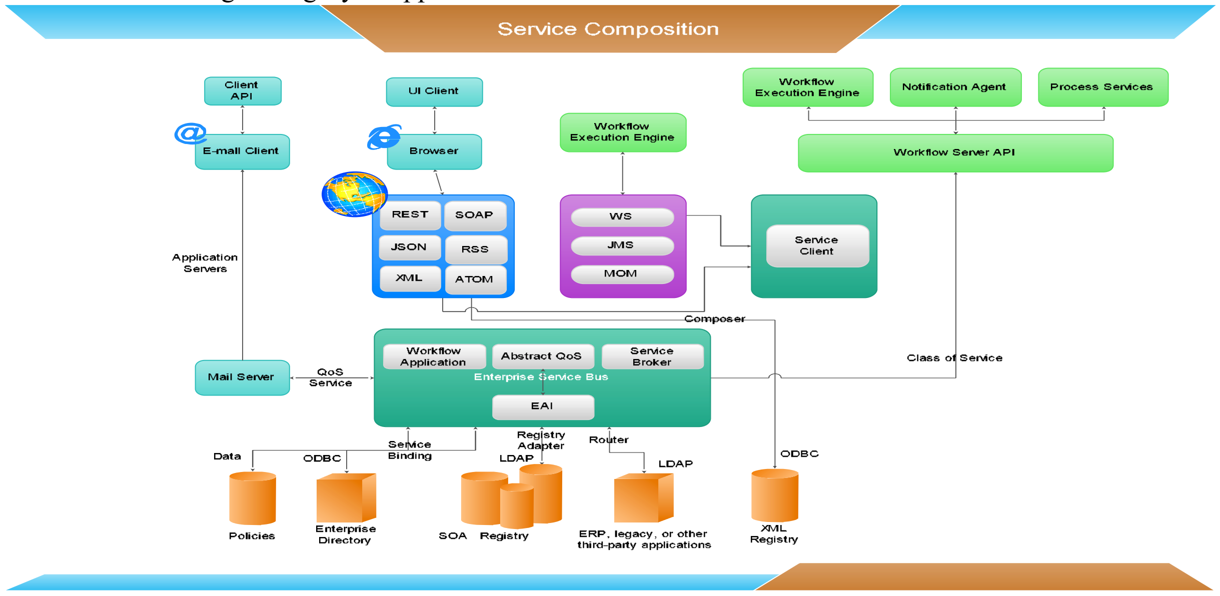

Fig. 1. Architecture overview of the orchestration of web service and service fabric composition.

The business feature provider defines the composite service model and defines the field structures as per the client application requirement to suite the best business outcome. Once the service is defined we can use workflow language such as Webservice business extraction language (WS-BPEL) or YAWL Workflows or OWL-S to model the abstract service layer for business composite services. UDDI is being used to provide the service registry that is used to locate the webservices and provide the semantic services for functional use. Based on the outcome of service selection or business or nonfunctional candidate services are categorized for each business feature, context aware QOS service selection and consumption is based under the selection criteria and context aware QoS and service contracts. Lets take business webservices that are defined by $f(\vec{x})$ and $g(\vec{x})$ functions on subset of $\mathbb{R}^n$.

We say $f(\vec{x})$ is $O(g(\vec{x}))$ as $\vec{x}\to\infty$ if and only if $\exists C \exists M > 0$ such that $|f(\vec{x})| \leq C|g(\vec{x})|$ for all $\vec{x}$, for example below statement $f(n,m) = n^2 + m^3 + O(n,m)$ as $n,m\to\infty$ asserts that there exist constants C and M such that

$\forall n, m > M : |g(n,m)| \le C(n+m),$

where $g(n,m)$ is defined by

$f(n,m) = n^2 + m^3 + g(n,m)$

Note that this definition allows all of the service attributes of $\vec{x}$ to increase to infinity. In particular the statement

$f(n,m) = O(n^m)$ as $n, m \to \infty$

(i.e., $(\exists C \exists M \forall n \forall m ...)$) is quite different from

$\forall m : f(n,m) = O(n^m)$ as $n, m \to \infty$

(i.e., $(\forall m \exists C \exists M \forall n ...)$)

Above illustration express the service attributes that can be n- times large in production environment which can be supported by high scale current available infrastructure but there is always a limit in computational systems to accept the payload and provide the performance results.

B. *QoS Contraints and Distribution of service component in Extended Enterprise Architecture*

End users specify their QoS requirements, such as Average request and response time, throughput/bandwidth, maximum computational and resource usage cost. Based on above said QoS criteria users consume candidate services but internally services are orchestrated wrt business features and tasks oriented workflow and each service component maintains the stack of context and contract aware QoS with calling functions/ services. Selecting the optimal service from the service registry for each component function is nontrival as the matrix of service composition could be very large. This request and response service scenario need to be solved better available solution, in this paper this problem is looked from the point of combinatorial optimization that consists of finding an optimal service from a finite set of available service objects in the service registry. As the dataset could be very large and exhaustive search might not be feasible to perform and could lead into performance barrier. It operates on the service business function domain of those optimization problems ,in which the set of services of adequate service QoS is discrete or can be reduced to discrete, and this will make it easier to search best possible service.

C. ***Mathematical framerwork of Service Orchestration***

Below listed mathematical proof explains the distribution of services and best possible service solution wrt contractual QoS Constraints.

The marginal distribution of services cycle counts provides a formula for the service component joint distribution of the cycle counts C_j^n, we find the service component distribution of C_j^n using a combinatorial approach combined with the inclusion-exclusion formula.

Lemma 1.1. For $1 \le j \le n$,

$$P[C_j^{(n)} = k] = \frac{j^{-k}}{k!} \sum_{l=0}^{[n/j]-k} (-1)^l \frac{j^{-l}}{l!} \qquad (1.1)$$

Proof. Consider the available registered Services in UDDI Registry set I of all possible cycles of length j, formed with business component elements chosen from set of available service components $\{1, 2, ...n\}$, so that $|I| = n^{[j]/j}$. For each $\alpha \in I$, consider the "property" G_α of having α; that is, G_α is the set of permutations $\pi \in S_n$ such that α is one of the component cycles of π. We then have $|G_\alpha| = (n-j)!$, since the elements of $\{1, 2, ..., n\}$ not in α must be permuted among themselves. To use the inclusion-exclusion formula we need to calculate the term S_r, which is the sum of the probabilities of the r -fold intersection of properties, summing over all sets of r distinct properties. There are two cases to consider. If the r properties are indexed by r cycles having no business elements in common, then the intersection specifies how rj elements are moved by the permutation, and there are $(n-rj)!1(rj \le n)$ permutations in the intersection. There are $n^{[rj]}/(j^r r!)$ such intersections. For the other case, some two distinct properties name some element in common, so no permutation can have both these properties, and the r -fold intersection is empty. Thus

$$S_r = (n-rj)!1(rj \le n) \times \frac{n^{[rj]}}{j^r r!} \frac{1}{n!} = 1(rj \le n) \frac{1}{j^r r!}$$

Finally, the inclusion-exclusion service composition series for the number of permutations having exactly k properties is

$$\sum_{l \ge 0} (-1)^l \binom{k+l}{l} S_{k+l,}$$

Which simplifies to (1.1) Returning to the original hat-check problem, we substitute j=1 in (1.1) to obtain the distribution of the number of fixed points of a random permutation. For $k = 0,1,\dots,n,$

$$P[C_1^{(n)} = k] = \frac{1}{k!}\sum_{l=0}^{n-k}(-1)^l\frac{1}{l!}, \qquad (1.2)$$

and the moments of $C_1^{(n)}$ follow from (1.2) with $j = 1$. In particular, for $n \ge 2$, the mean and variance of $C_1^{(n)}$ are both equal to 1. The joint distribution of webservices $(C_1^{(n)},\dots,C_b^{(n)})$ for any $1 \le b \le n$ has an expression similar to (1.2); this too can be derived by inclusion-exclusion. For any $c = (c_1,\dots,c_b) \in \mathbb{Z}_+^b$ with $m = \sum ic_i$,

$$P[(C_1^{(n)},\dots,C_b^{(n)}) = c]$$

$$= \left\{\prod_{i=1}^{b}\left(\frac{1}{i}\right)^{c_i}\frac{1}{c_i!}\right\}\sum_{\substack{l \ge 0 \text{ with} \\ \sum il_i \le n-m}}(-1)^{l_1+\dots+l_b}\prod_{i=1}^{b}\left(\frac{1}{i}\right)^{l_i}\frac{1}{l_i!} \qquad (1.3)$$

The service workflow joint moments of the first b counts $C_1^{(n)},\dots,C_b^{(n)}$ can be obtained directly from (1.2) and (1.3) by setting $m_{b+1} = \dots = m_n = 0$

The limit distribution of service cycle counts follows immediately from Lemma 1.1 that for each fixed j, as $n \to \infty$,

$$P[C_j^{(n)} = k] \to \frac{j^{-k}}{k!}e^{-1/j}, \quad k = 0,1,2,\dots,$$

So that $C_j^{(n)}$ converges in distribution to a random variable Z_j having a Poisson distribution with mean $1/j$; we use the notation $C_j^{(n)} \to_d Z_j$ where $Z_j \sim P_o(1/j)$ to describe this. Infact, the limit random variables are independent.

Above listed theorem explicit defines the best distribution of service components when the payload is large and also to adhere with QoS constraints and provide optimal service performance.

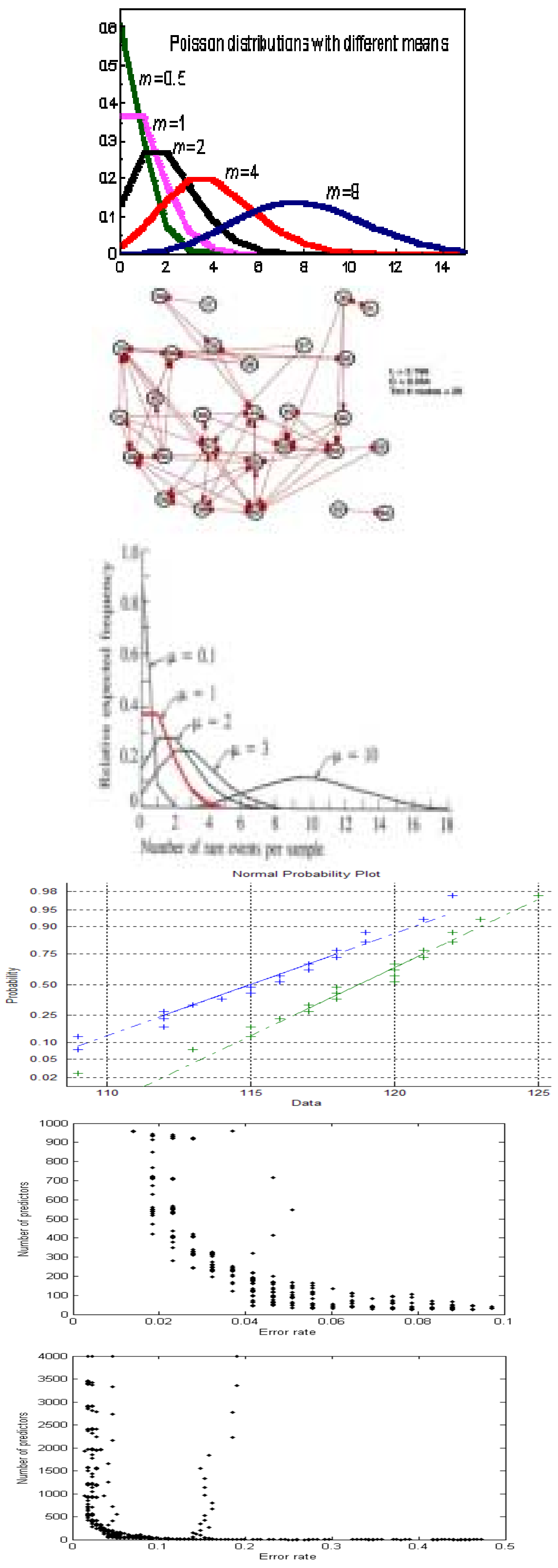

Fig. 2. Illustrates the service orchestration and composition in enterprise architecture and poisson

distribution and error rate is approximation is calculated.

Service Localization and Service Selection Strategy. The service local and global selection approach is especially useful for enterprise wide distributed system environments where central service bus maintains QoS administration and management and groups of candidate Web services are managed by distributed service hubs and stack of composite service portfolio. The objective function is to select optimal service from available group of service candidates and the services are independent in cluster of service offering hub. Using a Combinatorial resource attributes and features, the values of the different QoS criteria are mapped to a specified resource attribute and the service with optimal capability is selected. This approach is very efficient in terms of CPU cycle and resource usage, as the payload and response time complexity of the local optimization approach is O(l), where l is the number of service candidates in each group. Even if the approach is useful in decentralized environments, local selection strategy is not suitable for QoS-based service composition, with end-to-end constraints (e.g., maximum total price), since such global constraints cannot be verified locally.

Global Optimization. The global optimization approach was put forward as a solution to the QoS-aware service composition problem [Zeng et al. 2003, 2004; Ardagna and Pernici 2005; Ardagna and Pernici 2007; Kritikos and Plexousakis 2009]. This approach aims at solving the problem on the composite service level. The work of Zeng et al. [2003, 2004] focuses on dynamic and quality-driven selection of services. The authors use global planning to find the best service components for the composition. They use mixed integer programming techniques [Nemhauser and Wolsey 1988] (MIP) to find the optimal selection of component services. Similar to this approach Ardagna et al. [2005, 2007] extend the linear programming model to include local constraints. In their model, global constraints are specified by the end-user on the composition level, while local constraints can be specified by the designer of the composition on the component services' level. Unlike this approach, our proposed solution decomposes all end-users' global constraints into local constraints. Our solution can also easily handle local constraints given by the designer of the composition. Another difference between the two approaches is that, in our approach, mixed-integer programming is applied for the decomposition of the constraints and not for the selection of the services. As we discuss later in Section 4 and Section 5, the number of random variables in our model is much smaller than the number of random variables in the other approach, which makes our model more efficient in terms of computation time. Kritikos and Plexousakis [2009] claim that mixed-integer programming should be used as a matchmaking technique instead of constrained programming (CP) and provide experimental results proving it. Zhai et al. [2009] propose a solution for repairing failed service compositions by replacing the failed services only and reconfiguring the composition in a way that still meets the users' end-to-end QoS requirements. The reconfiguration of the composition and the suggestion of new services is based on MIP. Generally, MIP methods are very effective when the size of the problem is small. However, these methods suffer from poor scalability due to the exponential time complexity of the applied search algorithms [Maros 2003]. Already in larger enterprises and even more in open service infrastructures with a few thousand services, the response time for a service composition request could already be out of the real-time requirements.

Heuristic Solutions. As discussed earlier, the problem of QoS-aware service selection can be modeled as a multi-dimensional multiple-choice knapsack problem (MMKP). In the MMKP problem, a set of groups of items, where each item has a profit value and consumes some resources, exist. The goal of this problem is to select exactly one item from each group such that the total profit value is maximized under some constraints on total resource consumption. The groups and items in this problem correspond to the service classes and the candidate services in theWeb service scenario, respectively. The profit value of an item corresponds to the utility value of a Web service and the constraints on the resource consumption correspond to the QoS constraints. There exist a number of heuristics in the literature for solving the knapsack problem in general and the MMKP variant of this problem in particular. In Khan [1998] a heuristic HEU for solving the MMKP was presented. HEU uses a measurement called aggregate resource consumption to decide which item from each group should be upgraded in each round of selection. In Akbar et al. [2001] a modified version of HEU, M-HEU called was presented, where a preprocessing step to find a feasible solution and a postprocessing step to improve the total value of the solution with one upgrade (i.e., item selection that increases the total profit value) followed by one or more downgrades (i.e., item selection that decreases the total profit value) were added. In Akbar et al. [2006] the authors propose another heuristic, C-HEU, for solving the MMKP problem and evaluating its performance and optimality against several heuristics, including the M-HEU algorithm. The results of their evaluation show that C-HEU outperforms MHEU in terms of computation time. However, the experiments also show that M-HEU produces the nearest to the optimal solution among all the heuristics,while the optimality of C-HEU decreases as the number of items in each group increases. Furthermore, the results show that the C-HEU algorithm performs better in systems where the objective value to be maximized (i.e., the utility value in the Web service scenario) is not proportional to the resource requirements (i.e., the QoS values of Web services). Since the utility value of a given Web service is proportional to the QoS level of the service, the C-HEU algorithm is not applicable to the QoS-aware service selection problem. A modified version of the M-HEU algorithm, called WS-HEU, designed for the QoSaware service selection problem was proposed in Yu et al. [2007]. The authors propose two models for the QoS-based service composition problem: (1) a combinatorial model and (2) a graph model. A heuristic algorithm is introduced for each model: the WSHEU algorithm for the combinatorial model and the MCSP-K for the graph model. The time complexity of WS-HEU is polynomial, whereas the complexity of MCSP-K is exponential. Despite the significant improvement of these algorithms compared to exact solutions, both algorithms do not scale with respect to an increasing number of Web services and remain out of the real-time requirements. In our experimental evaluation, which we present in Section 5.2, we compare our hybrid approach against the WS-HEU algorithm. The results indicate the the hybrid approach outperforms the WS-HEU. Moreover, the WS-HEU algorithm is not suitable for the distributed setting of Web services. This is due to the fact that WS-HEU (following the originalM-HEU algorithm) starts with a preprocessing step for finding an initial service combination that satisfies all constraints but is not necessarily the best solution, and improves this solution in several rounds of upgrades and downgrades of one of the selected component services. Applying this algorithm in a distributed setting where the QoS data of the different service classes is managed by distributed service brokers would cause very high communication costs among the brokers to find the best composition. The hybrid approach, we propose in this article solves the composition problem more efficiently and fits the distributed environment of Web services well.

D. *Constraints and Criteria for Quality of Services*

In our study we consider quantitative nonfunctional properties of Web services, which can be used to describe the quality criteria of a Web service [Zeng et al. 2003; Liu et al. 2004]. These can include generic QoS attributes like response time, availability, price, reputation and so on, as well as domain-specific QoS attributes like bandwidth for multimedia Web services as long as these attributes can be quantified and represented by real numbers.We use the vector Qs = {q1(s), . . . , qr(s)} to represent the r QoS attributes of service s, where the function qi(s) determines the value of the ith quality attribute of s. The values of these QoS attributes can either be collected from service providers directly (e.g., price), recorded from previous execution monitoring (e.g., response time) or from user feedbacks (e.g., reputation) [Liu et al. 2004]. The set of QoS attributes can be divided into two subsets: positive and negative QoS attributes. The values of positive attributes need to be maximized (e.g., throughput and availability), whereas the values of negative attributes need to be minimized (e.g., price and response time). For the sake of simplicity, in this article we consider only negative ares (positive attributes can be easily transformed into negative overs by multiplying their values by −1).

E. *QoS Computation of Composite Services*

In our previous work [Alrifai and Risse 2009] we focused on sequential compositions. In the present work, we extend the QoS computation model to support nonsequential compositions. More specifically, in this study we consider the following four elementary composition constructs, which can be used for building more complex compositions.

(1) Sequential. A sequence of services {s1, . . . , sn} are executed in a strict sequential order one after another.

(2) Loop. A block of one or more services is executed repeatedly up to a maximum number of k executions. The aggregated QoS values of a loop construct is computed based on the worst-case scenario, where the number of iterations equals k.

(3) Parallel (and split/and join). Multiple services {s1, . . . , sn} are executed concurrently and merged synchronization.

(4) Conditional (exclusive split/exclusive join). A set of services {s1, . . . , sn} are associated with a logical

condition, which is evaluated at runtime and, based on its outcome, one service is executed. The estimated QoS values of a conditional construct are the worst values of the services {s1, . . . , sn}. For example, the estimated execution price of the conditional construct is computed as the price of the most expensive service among the services {s1, . . . , sn}. The QoS vector for a composite service CS with CS = {s1, . . . , sn} is defined as QCS = {q1(CS), . . . , qr (CS)}, where qi(CS) is the estimated end-to-end value of the ith QoS attribute. The value of qi(CS) is computed by aggregating the QoS values of the component services {s1, . . . , sn}. Depending on the QoS attribute and the composition pattern, there can be three different types of aggregation relations: (1) summation, (2) product or (3) minimum/maximum relations. Table I shows examples of such aggregation functions. In this example, we consider four different QoS attributes.

—Response Time: is the average execution time of the service and is measured by the time between sending a request and receiving a response.

—Price: is the amount of money the requester has to pay for using the service.

—Availability: is the probability that the service is accessible. This is usually measured by the percentage of the service up-time in a given period. The aggregated availability value of a composition is measured by the probability that all composed services are available at execution time, which is usually computed by the product of the individual probabilities.

—Throughput: is the number of requests the service can process per second. The overall throughput of a composition is then determined by the lowest throughput value of the composed services.

The aggregation function of each of these attributes is shown for each of the four composition constructs mentioned above. Notice that in the conditional construct, only one branch is executed at runtime, which is not known a priori. Therefore, we consider the worst-case scenario for estimating the QoS value of the conditional construct. For example, the estimated response time (or price) of a conditional construct that consists of n branches (such as the one shown in Figure 2) is the maximum response time (or price) among the n component services, that is, maxnj =1 q(sj). Similarly, for the availability (or throughput) attribute, we use the minimum value among the nservices,that is, minnj =1 q(sj).

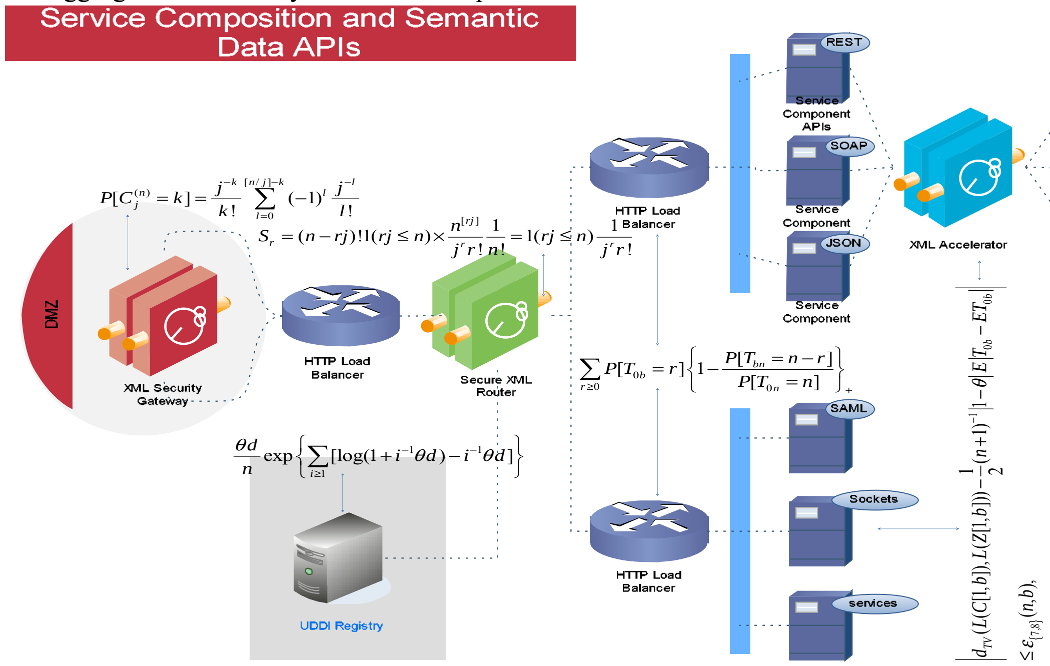
Service Composition and Semantic Data APIs
DMZ
XML Security Gateway
HTTP Load Balancer
Secure XML Router
HTTP Load Balancer
HTTP Load Balancer
UDDI Registry
REST
Service Component APIs
SOAP
Service Component
JSON
Service Component
XML Accelerator
ATOM
Service Component APIs
WS
Service Component
MOM
Service Component
SAML
Sockets
services
$P[C_j^{(n)}=k]=\frac{j^{-k}}{k!}\sum_{l=0}^{[n/j]-k}(-1)^l\frac{j^{-l}}{l!}$
$S_r=(n-rj)!1(rj\le n)\times\frac{n^{[rj]}}{j^r r!}\frac{1}{n!}=1(rj\le n)\frac{1}{j^r r!}$
$\sum_{r\ge 0}P[T_{0b}=r]\left\{1-\frac{P[T_{bn}=n-r]}{P[T_{0n}=n]}\right\}_+$
$\frac{\theta d}{n}\exp\left\{\sum_{i\ge 1}[\log(1+i^{-1}\theta d)-i^{-1}\theta d]\right\}$
$\left|d_{TV}(L(C[1,b]),L(Z[1,b]))-\frac{1}{2}(n+1)^{-1}|1-\theta|E|T_{0b}-ET_{0b}|\right|\le\varepsilon_{\{7,8\}}(n,b),$

Fig. 1. Architecture overview of the orchestration of web service and service fabric composition.

III. REFERENCES

[1]. S. Abiteboul, R. Hull, and V. Vianu. Foundations of Databases. Addison-Wesley, 995.

[2]. S. Abiteboul and V. Vianu. Datalog extensions for database queries and updates. J. Comput. Syst. Sci., 43(1):62–124, 1991.

[3]. A. Acciarri, D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, M. Palmieri, and R. Rosati. QuOnto: Querying ontologies. In Proc. AAAI-2005, pp. 1670–1671, 2005.

[4]. H. Andr´eka, I. N´emeti, and J. van Benthem. Modal languages and bounded fragments of predicate logic. J. Phil. Log., 27(3):217–274, 1998.

[5]. R. Angles and C. Gutierrez. The expressive power of SPARQL. In Proc. ISWC- 2008, pp. 114–129, 2008.

[6]. G. Antoniou. Non-monotonic rule systems on top of ontology layers. In Proc. ISWC-2002, pp. 394–398, 2002.

[7]. K. R. Apt, H. Blair, and A. Walker. Towards a theory of declarative knowledge. In J. Minker, editor, Foundations of Deductive Databases and Logic Programming, pp. 89–148. Morgan Kaufmann, 1988.

[8]. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. DL-Lite in the light of first-order logic. In Proc. AAAI-2007, pp. 361–366, 2007. 72

[9]. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The DL-Lite family and relations. J. Artif. Intell. Res., 36:1–69, 2009.

[10]. F. Baader. Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In Proc. IJCAI-2003, pp. 319–324, 2003.

[11]. F. Baader, S. Brandt, and C. Lutz. Pushing the EL envelope. In Proc. IJCAI-2005, pp. 364–369, 2005.

[12]. F. Baader, C. Lutz, and B. Suntisrivaraporn. CEL — A polynomial-time reasoned for life science ontologies. In Proc. IJCAR-2006, pp. 287–291, 2006.

[13]. J.-F. Baget, M. Lecl`ere, M.-L. Mugnier, and E. Salvat. On rules with existential variables: Walking the decidability line. Artif. Intell., 175(9/10):1620–1654, 2011.

[14]. J.-F. Baget, M.-L. Mugnier, S. Rudolph, and M. Thomazo. Walking the complexity lines for generalized guarded existential rules. In Proc. IJCAI-2011, pp. 712–717, 2011.

[15]. V. Barany, G. Gottlob, and M. Otto. Querying the guarded fragment. In Proc. LICS- 10, pp. 1–10, 2010. Full paper available from the authors.

[16]. C. Beeri and M. Y. Vardi. The implication problem for data dependencies. In Proc. ICALP-1981, pp. 73–85, 1981.

[17]. C. Beeri and M. Y. Vardi. A proof procedure for data dependencies. J. ACM, 31(4):718–741, 1984.

[18]. B. Bishop, A. Kiryakov, D. Ognyanoff, I. Peikov, Z. Tashev and R. Velkov. OWLIM: A family of scalable semantic repositories. J. of Web Semantics, 2(1):33–42, 2011.

[19]. D. Brickley and R. V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema, 2004. W3C Recommendation (10 Feb. 2004). http://www.w3.org/TR/rdf-schema/.

[20]. J. D. Bruijn, T. Eiter, A. Polleres, and H. Tompits. Embedding non-ground logic programs into autoepistemic logic for knowledge base combination. In Proc. IJCAI-2007, pp. 304–309, 2007.

[21]. L. Cabibbo. The expressive power of stratified logic programs with value invention.Inf. Comput., 147(1):22–56, 1998.

[22]. A. Cal`ı, D. Calvanese, G. De Giacomo, and M. Lenzerini. Accessing data integration systems through conceptual schemas. In Proc. ER-2001, pp. 270–284, 2001.

[23]. A. Cal`ı, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. In Proc. KR-2008, pp. 70–80, 2008.

[24]. A. Cal`ı, G. Gottlob, and T. Lukasiewicz. A general Datalog-based framework for tractable query answering over ontologies. In Proc. PODS-2009, pp. 77–86, 2009.

[25]. A. Cal`ı, G. Gottlob, and A. Pieris. Tractable query answering over conceptual schemata. In Proc. ER-2009, pp. 175–190, 2009.

[26]. A. Cal`ı, G. Gottlob, and A. Pieris. Advanced processing for ontological queries. Proc. VLDB-10, 3(1):554–565, 2010.

[27]. A. Cal`ı, G. Gottlob, and A. Pieris. Query answering under expressive Entity-Relationship schemata. In Proc. ER-10, pp. 347–361, 2010.