服务合成的性能预测

谈华芳[1] Ratakonda Krishna C.[2] 朱 俊[1] 苏 辉[1]

(1.IBM 中国研究院；2.IBM 华生研究中心)

Service Composition Performance Predication

Huafang Tan (IBM China Research Lab, Beijing 100094)

Ratakonda Krishna C. (IBM Waston Research Center)

Jun Zhu, Hui Su (IBM China Research Lab, Beijing 100094)

Abstract: One benefit of SOA is from service composition which combines existing services to form a new valued-added service in the form of business process. This also creates the need to ensure that the performance of those composite services meets the business requirements. This paper proposes a method for evaluating the performance of composite services. We automatically transform a composite service modeled by BPEL, into a performance model based on Layered Queuing Networks (LQN) and then leverage existing LQN solvers to predict their performance. The inputs to our transformation algorithm are an XML file which contains a service composition model complying with the BPEL4WS specification, a related performance profile and a service topology. The output of the model is the corresponding LQN model which can be directly analyzed using existing LQN solvers.

Key words: BPEL; LQN; transformation

1 Introduction

Service-oriented architecture(SOA)[1] is an evolution of distributed computing based on the request/reply design paradigm for synchronous and asynchronous applications. An application's business logic or individual functions are modularized and presented as services for consumer/client applications. What's key to these services is their loosely coupled nature; i.e., the service interface is independent of the implementation. Application developers or system integrators can build applications by composing one or more services into composite services without knowing the underlying implementations of individual services. Currently a variety of approaches to service composition are emerging. BPEL4WS (BPEL)[2] is the most popular one.

In carrying out this composition task showed by Fig.1, the critical step is to look up right services to make an abstract process executable. Typically the integrator has either no direct control over these factors themselves or cannot easily predict them from the available information. Thus, the focus is typically on ensuring that the functional requirements are met. But it is also very important to understand and be able to predict performance factors such as average request-response time. Given that non-functional requirements play a key role in the usability and scalability of an application, it is highly desirable to clearly understand the performance implications of a given service composition and to be able to reason about them during design to help integrator select right services and make deployment decisions.

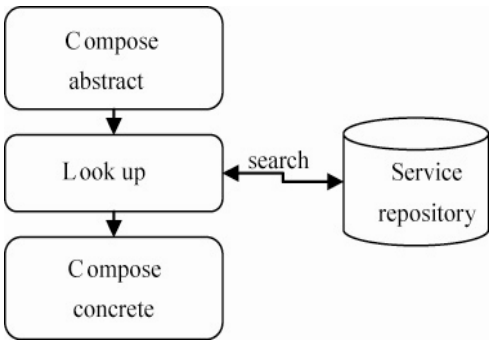

Fig.1　Service composition task

To predicate the performance of a composition service, firstly we need build a performance model and then use the analysis and simulation method to get the predicated performance metrics. There are two approaches for performance modeling based on Layered Queueing Network(LQN)[3] and historical data analysis. LQN is a popular performance prediction model and has been used in study the performance of distributed software systems[4,5] which amply demonstrate its potential. There are many researchers work on how to analyze or simulate it to identify the performance metrics such as response time, throughput and utilization according the LQN model. Comparing with historical performance data analysis, it doesn't need historical data. So it is adapt to a new designed composite service.

This paper provides a method to automatically generate an LQN performance model from the service composition model described by a BPEL program and the related performance profile–thus enabling the architect to combine the activities of design specification and performance modeling. We believe and in practice observed that designing keeping the SOA performance in perspective from the design onset helps improve the design process itself. This ensures the architect makes the right design choices and does not need to redesign/ re-architect during the testing phase necessitating expensive post-development fixes.

The rest of the paper is organized as follows. The related background introductions about LQN model and BPEL are presented in Section 2. Section 3 gives an overview of the proposed method. The transformation of BPEL to LQN is discussed in Section 4. A case study of a loan approval process is given in Section 5. The related work and conclusion are separately presented in Section 6 and Section 7.

2　Background

2.1 LQN model

LQN model is a widely used technique for predicting the performance of computing system. It was developed as an extension to the QN model to handle complex interactions among various software and hardware in client-server distributed environment. An LQN model uses terms such as task, host processor, entry, call and demand. It is represented as an acyclic graph whose nodes are the tasks which represent software entities and hardware devices, and the arcs denote calls. A task has one or more entries which represent operations performed by the task. A related host processor is linked with a task to model the physical entity that carries out the operations. Calls are requests for service from one entry to an entry of another task. Demand is the average amount of host processing time and average number of calls for service operations required to complete an entry. Detailed description of the sequence of operations, when a task accepts a request at an entry, can be defined by describing activities with a precedence graph. Activities are connected together to form a directed graph which represents one or more execution scenarios. Execution may branch into parallel concurrent threads of control which may or may not execute in parallel on the target system. Execution may also choose randomly between different paths[3]. This semantic is consistent with BPEL.

There are six types of the connection between activities supported by the extension of LQN, Connecting, And-Fork, And-join, Or-Join, Or-Fork and Repetition. An example with an activity is given in Fig.5. It is consistent with the relationship among activities in BPEL.

The parameters of an LQN model are as follows:

①customer (client) classes and their associated populations or arrival rates,

②for each phase (activity) of a software task entry: average execution time,

③for each phase (activity) making a request to a device: average service time at the device, and average

number of visits,

④for each phase (activity) making a request to another task entry: average number of visits,

⑤for each request arc: average communication delay,

⑥for each software and hardware server: scheduling discipline.

2.2 BPEL

The business process execution language for web services(BPEL4WS)[2] represents the uniting of two previously competing standards: the web services flow language(WSFL)from IBM and Microsoft's XLANG. Like WSFL and XLANG, BPEL4WS has been designed to compose web services.

In BPEL4WS, the service providers and client are defined as business partner. The service is invoked by invoke activity. Besides invoke activity, the basic activities include assignments, receiving requests, replying to requests, waiting for a durance of time and empty. These basic activities are combined into structured activities using ordinary sequential control flow constructs like sequencing, switch constructs, and while loops. Concurrency is provided by the flow construct. The synchronization between concurrent activities is achieved by using links. Each link has a source activity and a target activity. If there is a link from one activity to another, then the target activity can only start once the source activity has completed. With each link a transition condition which is a Boolean expression is associated. Each activity has a join condition. The join condition consists of incoming links of the activity combined by Boolean operators. Once all the source activities corresponding to the incoming links of an activity have completed, the join condition of the activity is evaluated. If the join condition evaluates to true, then the activity is started. Otherwise, the activity will never start.

3 An Overview of Our Service Composition Performance Evaluation Framework

Our service composition performance evaluation framework is showed by Fig.2. We process XML files produced by current service composition tools (BPEL editors), which obviously do not support performance profile and do not include service topology information. Therefore, we attach those related information by hand. Then our transformation component produces the LQN model to the solver which generates the predication result. The integrator may change the service composition according the analysis result. So we give a feedback from the result to service composition (represented with gray arrow). But currently, we do not implement it.

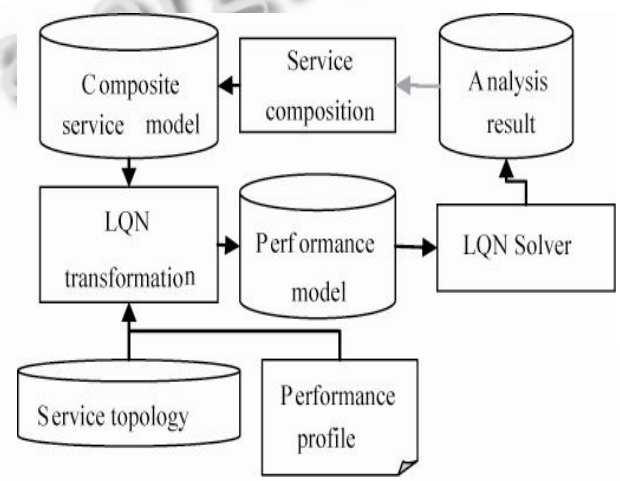

Fig.2 A component view of service composition performance evaluation framework

The transformation traverses the BPEL program to build the structure of the performance model. The related business partners and the process self are constructed as tasks for the model. The operations for the service and process are constructed as the entries for the related tasks. The activities in the process are transferred to the corresponding activities for the process task. And the related call is generated between a specific invoke activity and the service.

The BPEL performance profile provides facility for specifying workload characteristics and execution parameters which are used by transformation algorithm to define the visit ratio and execution time of the related activity.

The visit ratio is determinate by the workload characteristics which includes the concurrent request number and the request type mix. We use the stochastic

information which indicates the probability of transitions being fired at runtime to represent the request type distribution for the switch, pick, link etc.transition and use the average number of the execution of the repeat part to represent the while transition.

The execution time is directly given by the related activity execution time in the performance profile during transformation. The service topology is important and it gives us information about the linkage between hardware device(processor) and the related task during transformation.

4 LQN Generation

The inputs of the LQN generation are the BPEL program, the related performance profile and the service topology. The output is an LQN model which can be analyzed by the existing LQN solvers.

The algorithm walks through the nodes of a BPEL program and follows the listing rules to translate it to a corresponding LQN model. The main steps of the algorithm are:

4.1 The algorithm

① Generate the LQN model structure

a) Determine LQN tasks from the business partner link definition (the composed services).

b) Build the linkage between tasks and hardware devices according the service topology information.

② Generate the LQN details on entries, activities from the BPEL

a) Add entries for the corresponding task. Each operation in the service definition is mapped to an entry. The Receive and the Pick with OnMessage have a corresponding entry associated with the process task.

b) Add activities within an entry. According the LQN definition, when a task accepts a request at an entry, the detailed description of the sequence of operations can be defined by describing activities with a precedence graph. So the activities followed with this request are transformed to the related activities of the LQN model according the semantic of BPEL elements.

③ Traverse the LQN elements, compute their parameters and write out the model file. The parameters for each activity are service times and visit ratios. They are all given by the BPEL performance profile.

4.2 Traversing the BPEL program

One important part of the transformation is traversing the BPEL program.

There are two steps for it:

① Parse BPEL program to a graph or tree. The model generator of the Eclipse Modeling Framework (EMF)[6] is used here to generate a hierarchy of Java classes from the XSD specification of BPEL4WS. These classes represent the abstract syntax of the language and parse the BPEL program to a set of instances of those classes.

② Visit each node in a suitable order. When visiting a node, a corresponding rule is applied to implement transformation according the type of the node. A guider is used here to make sure the order of the visiting. To separate the action definition from the AST classes, a visitor pattern can be exploited which supports defining external methods for different types of nodes.

4.3 Transformation rule

For each type of activity in BPEL, we have a mapping rule to transform it to the corresponding activity in LQN model.

① Basic activity is directly mapped to an activity in the target LQN model except the invoke activity. For the invoke activity, besides a corresponding activity, a call is generated to the related service task entry.

② Structural activities: Sequence, while, pick, switch, flow.

a) The sequence is directly mapping to a sequence activity.

b) The while is mapping to a loop. If the repeat part is a sequence, the rest of the sequence is expanded one by one. If it is a complex structure of the activities, a separate pseudo-task is added. And the nested activity is added to the task.

c) The Switch and Pick is mapping to OR-fork and OR-join.

d) Flow is mapped to a pair of AND-fork and AND-join. If an activity is a target of link, it has a join condition, and there is an OR-fork added as a precedence of it and a corresponding OR-Join is added as its

subsequence showed by Fig.3. The visit ratio for each branch is given by the distribution of the value for the condition expression. If the activity is act as a source of more than one links, one AND-fork is added as a subsequence of the activity. If the activity is act as a target of more than one links, one AND-join is added as a subsequence of the activity. If the activity in a flow does not act as a source of any one link, one sequence is added between AND-fork for the flow and the activity. If the activity in a flow does not act as a target of any link, one sequence is added between the activity and the AND-fork for the flow.

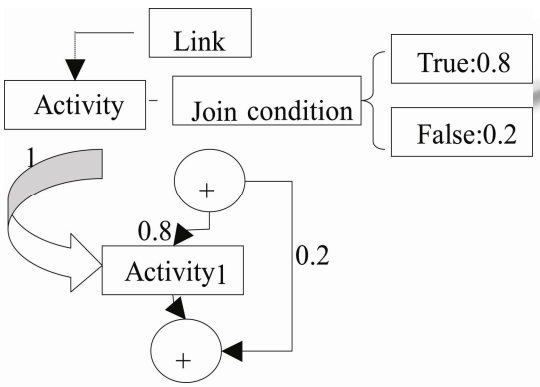

Fig.3 Activity with undetermined join condition expression transformation

4.4 The parameters of LQN elements

The visitor ratio for each kind of transition can be directly attached to the corresponding edge of the model excepting link. To calculate the visit ratio for the target activity of more than one link, we look each value distribution for the link condition expression as a (0, 1) distribution which is given by the corresponding transition probability in performance profile. According join condition expression, the conditional probability for the join condition is calculated based on value distribution of each individual link and the corresponding visit ratio for the target activity can be acquired.

During design time, the values of the parameters are given by the designer. After the process is deployed, the monitoring and measurement can be applied to provide feedback to the performance profile for the process.

5 Case study

This section gives the result of the BPEL program to LQN transformation algorithm applied to a LoanApproval Process. The generated LQN model is solved under the different request number at the certain request mix distribution with an existing LQN analytic solver [7]. The purpose of this paper is to present the proposed BPEL to LQN transformation, so no performance analysis results are presented here.

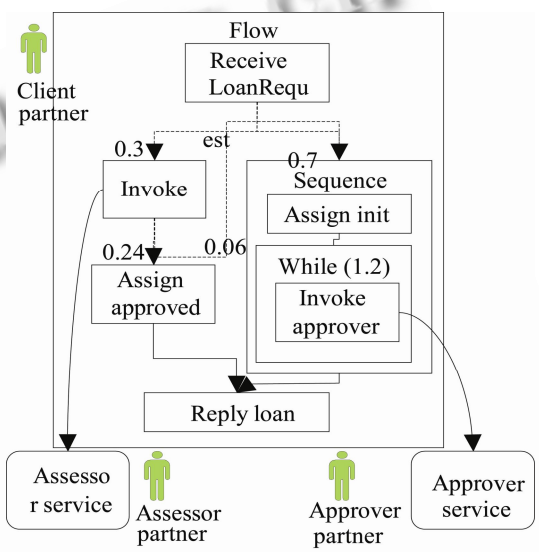

Fig.4 Loan approval process

The LoanApproval process is showed by Fig.4. Two services are composed by it. One is Assessor and another is Approver. If the request amount is lower than 1,000, the Assessor service is invoked to do the evaluation. If the amount is higher than 1,000 or the Assessor service reply “risk high”, the approver service is invoked to get the approval. When the first invocation to the approver service is failed, it will be invoked again. But the invocation number for this service is no more than two. If the Assessor service reply “risk low” or the approver service reply “approved”, the loan request is approved, otherwise is failed /rejected. The parameters for the request mix are annotated to each transition with value. In this process, the transitions and related values are listed by Table 1.

Table 1　Transactions and the related values for the LoanApproval proces

Transaction description	Probability for the transition being fired
Link1:The loan amount is lower than 1,000	0.3
Link2: The loan amount is high than 1,000	0.7
Link3: The assessor return risk low	0.24
Link4: The assessor return risk high	0.06
While: the number of the repeat part execution	1.2

The rectangle denotes the activity, and the rectangle with rounded corner denotes the services are composed by this process. The dashed arrow line denotes the link with condition.

The LQN model automatically generated for this process by our transformation algorithm is given in Fig.5. The nomenclature is adopted from Chu et. al.[8]. The entry for the LoanApproval process is "loanApproval" and the "reply loan" is a reply activity. When it is finished, it sends a reply to the requester that initiated the execution of the entry. Invoke assessor and Invoke approver separately makes one synchronous call to entry requestAssess and requestApproval.

The model was then given as input to the analytical LQN solver and the results for the Mean Client Response Time for the different values of the number of clients are obtained.

6　Related Work

To make sure a composed service in a form of process is efficient in terms of its service time, its ability to handle higher loads, the integrator should select the appropriate services that have operational metrics (such as service time, load capacity) during design stage. There are two research directly use the service composition model as performance model to predicate the performance. Mathematical methods have been used by Cardoso, Miller et al.[9] to analyze and estimate the overall QoS of a process. But it is not flexible enough to handle the variable workload. Another alternative for estimating the QoS of a process is to utilize simulation analysis[10]. Simulation can play an important role in evaluating the quality of a process, before its actual execution.

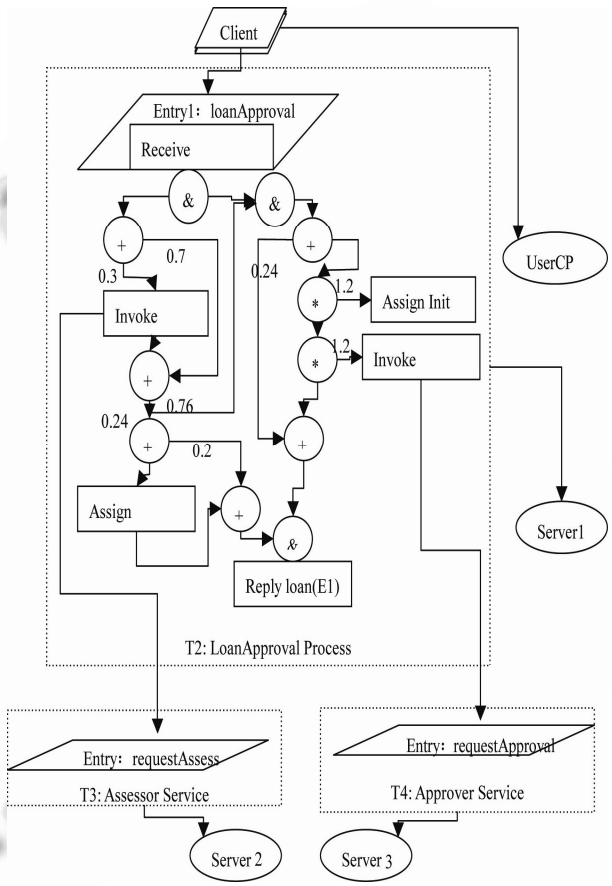

Fig.5　LQN model automatically generated for Loan-Approval proces

The advantage of our method to do performance evaluation is that it can leverage already existing LQN research efforts to do performance evaluation of a process by automatically transformation from a BPEL program with performance annotation into an LQN model. So it can use some features from LQN to make evaluation more flexible, such as change each service scheduling discipline.

There are also some related works which transfer a UML model with performance annotation into an LQN

performance model based on graph-grammar[11,12]. This approach is similar with it, but our target is service composition model, not UML model.

7 Conclusions

SOA brings a good benefit of reuse by composing services. But it also brings a challenge to do performance analysis. Performance estimation can play an important role in evaluating the performance before its actual execution. The innovation contribution of this paper is a method to automatically transform a composite service to an LQN model to leverage already existing research to do performance estimation in design stage and make sure the consistence between design specification and performance modeling.

One kind of extension is to give more useful feedback to service integrators when they do service composition according the analysis result, such as which service is the bottleneck. And another kind of extension is to let current BPELWS specification contain performance profile and service deployment information to make performance predication more smoothly.

References

1 SOA definition http://www.javaworld.com/ javaworld/ jw-06-2005/jw-0613-soa.html.

2 BPEL4WS. http://www.oasis-open.org.

3 Franks R G. Performance analysis of distributed server systems[Ph.D.dissertation].Carleton University, Ottawa,Ontario,Canada, 1999.

4 Hrischuk C, Rolia J, Woodside CM. Automated generation of software performance model using an objectoriented prototype. International Workshop on Modelling and Simulation. Analysis, Simulation of Computer and Telecommunication Systems(MASCOTS'95), 1995:399－409.

5 Liu TK, Kumaran S, Luo ZW. Layered Queueing Models for Enterprise Java Beans Applications, IBM Research Report, 2001.

6 Budinsky F, Steinberg D, Merks E, Ellersick R, Grose TJ. Eclipse Modeling Framework. The Eclipse Series. AddisonWesley, 2003.

7 LQN Solver. http://www.sce.carleton.ca/rads/ek-rads-etc/software.html.

8 Chu WW, Sit CM, Leung KK. Task response time for real-time distributed systems with resource contention. IEEE Transactions on Software Engineering, 1991,17 (10):1076－1092.

9 Cardoso J, Amit PS, A. John M, et al. Modeling quality of service for workflows and Web service processes. Web Semantics Journal: Science, Services and Agents on the World Wide Web Journal, 2004,1(3):281－308.

10 Cardoso J, Sheth A, Miller J. Workow Quality of Service. Proceedings of the International Conf. on Enterprise Integration and Modeling Technology and International Enterprise Modeling, 2002.

11 Petriu DC, Shen H. Applying the UML Performance Profile: Graph Grammar-Based Derivation of LQN Models from UML Specifications. Computer Performance Evaluation / TOOLS, 2002:159－177.

12 Amer H, Petriu DC. Software Performance Evaluation: Graph Grammar-based Transformation of UML Design Models into Performance Models. submitted for publication, 2002,33.