Tulsa: A Tool for Transforming UML to Layered Queueing Networks for Performance Analysis of Data Intensive Applications

Chen Li[1], Taghreed Altamimi[2], Mana Hassanzadeh Zargari[2], Giuliano Casale[1], and Dorina Petriu[2]

[1] Imperial College London, London SW7 2AZ, U.K.,
{chen.li1, g.casale}@imperial.ac.uk
[2] Carleton University, Ottawa K1S 5B6, Canada,
{taghreedaltamimi, manazargar, petriu}@sce.carleton.ca

Abstract. Motivated by the problem of detecting software performance anti-patterns in data-intensive applications (DIAs), we present a tool, Tulsa, for transforming software architecture models specified through UML into Layered Queueing Networks (LQNs), which are analytical performance models used to capture contention across multiple software layers. In particular, we generalize an existing transformation based on the Epsilon framework to generate LQNs from UML models annotated with the DICE profile, which extends UML to modelling DIAs based on technologies such as Apache Storm.

1 Introduction

The objective of our research is to design tools for iteratively enhancing the quality of data-intensive applications (DIAs) that leverage Big Data technologies hosted in private or public clouds. We consider that the DIAs are developed in a DevOps process, where the developers obtain runtime monitoring information, especially performance metrics, and reflect them back into design time models to reason about system performance improvements. In order to achieve that, a performance model needs to be generated from the DIA architecture model and runtime information. In this work, we use the Unified Modeling Language (UML) to specify the software architecture at the design stage. The architecture model characteristics (see Section 3) and its performance attributes are mainly captured by DICE profile [2], a recently proposed UML profile to annotate technology specific aspects of Storm, Hadoop and Spark into UML diagrams. DICE profile extends the standard MARTE profile[5], so it inherits the MARTE stereotypes for non-functional properties and performance attributes [4]. The specific problems we consider in this paper is how to annotate the runtime performance measurements in the UML model, and how to transform the UML model into the performance model for subsequent performance analysis.

This work is partially supported by the European Commission grant no. 644869, DICE

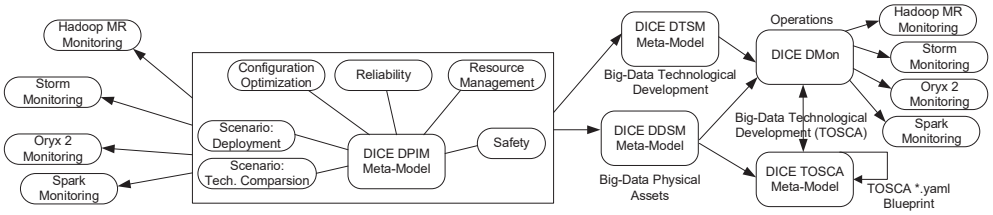

Fig. 1. High level abstract view of DICE profile

Several approaches have been proposed for generating performance models, such as queueing networks [1], stochastic Petri nets [3] and layered queueing networks (LQNs) [6] from architecture models. While these studies remain relevant, the advent of Big Data has popularized technologies such as Apache Storm and Hadoop in the implementation of DIAs. However, there is a shortage of methods for specifying UML models for these DIAs and automatically deriving performance models.

In this paper, we focus on Storm applications and transform the corresponding UML model into a performance LQN model. There are three reasons for choosing LQNs. First, a Storm topology may be seen as a network of buffers and processing elements that exchange messages, so it is quite natural to map them into a queueing network model. Second, the core elements of LQN models are semantically similar to the corresponding elements of UML activity and deployment diagrams. Third, LQN solvers such as LINE[1] or LQNS[2] are available to provide analytical methods to solve the LQN model. This paper proposes a new tool called Tulsa, which leverages DICE profile as a better way of annotating DIA UML models and transforms them to LQN models. Our work extends an existing UML+MARTE-to-LQN transformation based on the Epsilon framework[3] to leverage specific stereotypes of the DICE profile in the generation of LQN models [6].

2 DICE Profile

The DICE profile expresses some familiar model-driven architecture concepts for DIAs. In particular, the DICE profile offers three new models, called DICE Platform Independent Model (DPIM), DICE Technology Specific Model (DTSM), and DICE Deployment Specific Model (DDSM) [2][4]. Fig.1 shows the high level abstract view of DICE profile. DPIM provides an abstract specification of the DIA architecture, allowing the inclusion of computation nodes and storage nodes. At this abstraction layer, DPIMs help the developer to define a high-level topology, the main services exposed by the DIA and their QoS requirements. The DTSM layer is a refinement of the DPIM layer that encompasses technological decisions. For example, data processing needs are detailed in a DTSM through configuration requirements for appropriate Big Data technologies, such

[1] http://line-solver.sourceforge.net/
[2] https://github.com/layeredqueuing/V5/tree/master/lqns
[3] http://www.eclipse.org/epsilon/

as Hadoop. Lastly, DDSM enables the designer to specify deployment decisions on cloud infrastructures. In the DICE framework, such decisions can be subsequently translated into a concrete deployment blueprint based on TOSCA [7]. Our tool mainly focuses on the DTSM and DDSM layers, which are appropriate for performance evaluation.

3 Model Transformation

Comparing with our previous work in [6], Tulsa not only implements model transformation for general distributed systems, but also supports Storm applications by leveraging DICE profile. The underpinning scripts are mainly written in Epsilon Object Language (EOL) and Epsilon Transformation Language (ETL). Tulsa supports complex workflow which is assembled by a set of ANT tasks.

3.1 Model Mapping

The source model accepted by Tulsa considers two types of UML diagrams: deployment and activity diagram. The deployment diagram specifies the system configuration, e.g., indicating the functional components, assigning key attributes and defining constraints, and the activity diagram defines the behavior of the system. Table 1 shows the corresponding mapping from UML model annotated with DICE and MARTE stereotypes to LQN model.

Table 1. Model Mapping: from UML+DICE+MARTE to LQN Element

UML Model Element	DICE + MARTE Stereotype	LQN Model Element
model	None	lqnmodel
Deployment Diagram		
Device	GaExecHost, DdsmVMsCluster	processor
ExecutionEnvironment	DdsmStormCluster	None
Artifact	Scheduler, DdsmBigDataJob,StormSpout,StormBolt	task
Activity Diagram		
AcceptEventAction	GaStep	entry
InitialNode	GaWorkloadEvent	entry
OpaqueAction,CallOperationAction,SendSignalAction	GaStep	activity
DecisionNode,MergeNode,JoinNode,ForkNode	None	precedence
ControlFlow	StormStreamStep	precedence,synch-call,asynch-call

DICE UML model uses *Device* to stand for a VM cluster or a single server. DDSM provides a stereotype ≪*DdsmVMsCluster*≫ to capture characteristics of the VM cluster, e.g., *instances* tag means the number of single server in the cluster. *ExecutionEnvironment* represents the platform which is deployed on the VM. DDSM provides a stereotype ≪*DdsmStormCluster*≫ to annotate the Storm platform. The related single servers are nested in this *ExecutionEnvironment*. Tulsa transforms a single server (i.e., *Device*), which provides services to Storm platform, to a *Processor* in LQN model. An *Artifact* is used or produced by a software development process or deployment and operation of a system, e.g., software component. An *Artifact* can be transformed into a *Task* which stands for the software component in LQN model. In Storm applications, there are two types of *Artifact* called *Spout* and *Bolt*. DTSM defines stereotypes ≪*StormSpout*≫ and ≪*StormBolt*≫ for them respectively. These stereotypes provide tags for specifying the level of parallelism and the execution time, e.g., *parallelism* (i.e., specifying the number of threads).

Due to space limitations, we only describes some core elements and stereotypes which are mainly considered for Storm applications. More details on the elements with MARTE stereotypes can be found in [6].

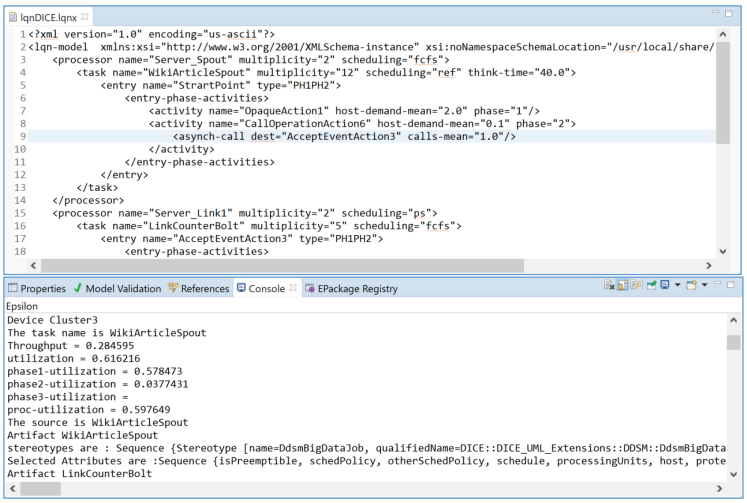

Fig. 2. Tool screenshots: fragment of the generated LQN model and performance results produced by the LQN solver

Tulsa is available at https://github.com/dice-project/DICE-Enhancement-APR. Pre-requirements of running the Tulsa are to install JDK 8, Eclipse 4.6.1, the DICE Profile[4] and the Epsilon framework. Fig. 2 shows screenshots of the results views of the Tulsa.

4 Conclusion

In this paper we have presented Tulsa, a tool for transforming a UML model annotated with DICE profile into a LQN model. Tulsa is a part of the DICE project, whose objective is to define a quality-driven framework for developing data-intensive applications that leverage Big Data technologies hosted in private or public clouds. Further development of the tool is expected to support other DIAs such as Hadoop and Spark.

References

1. Dubois, D.J., et al., Model-driven application refactoring to minimize deployment costs in preemptible cloud resources, In: CLOUD'16, IEEE Press, USA (2016)
2. Casale, G., Ardagna, D., Artac, M., et al. DICE: quality-driven development of data-intensive cloud applications, In: MiSE'15, IEEE Press, 78–83 (2015)
3. Merseguer, J., Campos, J., Software performance modeling using uml and petri nets, Performance Tools and Applications to Networked Systems, 2965, 265–289 (2004)
4. D2.1 Design and quality abstractions, http://www.dice-h2020.eu/deliverables/
5. UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems, Version 1.1, Object Management Group (2011)
6. Altamimi, T., Zargari, M.H., Petriu, D., Performance analysis roundtrip: automatic generation of performance models and results feedback using cross-model trace links, In: CASCON'16, Toronto, Canada, ACM Press (2016)
7. D2.3 Deployment abstractions, http://www.dice-h2020.eu/deliverables/

[4] https://github.com/dice-project/DICE-Profiles