Simulating Place/Transition Nets by a Distributed, Web Based, Stateless Service

Jan Henrik Röwekamp, Matthias Feldmann, Daniel Moldt, Michael Simon

University of Hamburg, Faculty of Mathematics, Informatics and Natural Sciences, Department of Informatics, http://www.informatik.uni-hamburg.de/TGI/

Abstract Executing Petri nets in a distributed manner is, in general, a non-trivial task. To achieve portability of the simulation, as a precondition for scalability, it is desirable to keep the majority of the simulation components stateless. This paper presents a first approach to model distributed Place/Transition nets (P/T nets) using stateless services and only a (stateful) database. The presented theory has been implemented in a web API based service. First tests look promising.

Keywords: P/T Nets, Distributed Systems, Scalability, Stateless Architecture, Tools, Container, Docker

Introduction

The astonishing growth of today's systems allows for widespread use of computational resources. The resulting complexity of distributed systems keeps rising.

Excellent formalisms to model this kind of systems are Petri nets and the diverse formalisms built on top of them. Petri net simulators, however, usually only focus on local execution of nets and less on distribution or scalability issues.

In this work, the basic idea of a Petri net simulator focused on distribution and scalable architecture is presented. It is important to note, that a distributed simulation of Petri nets is of interest, not the simulation of distributed Petri nets.

Related Work

Chiola and Ferscha studied distributed timed Petri nets in 1993 [1]. In the case of reference nets, some results for distributed simulation are available as a plug-in for Renew [3], using virtual machines and finally some ideas that form the basis of the simulator presented here [2].

Concept

To cope with the critical parts of a distributed Petri net simulation focus is put on the net partitioning, firing uniqueness and atomic token consumption. In regards to data storage transition firings and token consumption are both write operations. Write operations usually tend to be problematic in a concurrent environment.

Transition Firing and Token Consumption

As actively simulating transitions in a distributed fashion imposes severe difficulties in regards to the uniqueness of firing and/or self-concurrency, the presented approach focuses on simulating active *tokens*. Tokens, however, may also occur as passive entities. Actively simulated tokens will be referred to as *main tokens*. Places are not modeled, but tokens and transitions hold them implicitly.

The simulation is organized in multiple infinite loops. A node selects a random token as the main token and loads all tokens of all presets of all attached transitions into memory. The node then iterates through all transitions and tries to fire a random active transition. Firing is done by issuing a transaction to the underlying data storage, that assures uniqueness.

If conflicts occur (two or more nodes handle the same token), the transaction will succeed, that is committed into the database first.

The Simulator Core and Technical Aspects

The simulator consists of an evaluation unit layer and a database service. The database service holds the current state of the net, while the evaluation layer only temporarily copies data, that is required to compute transition firings. As the evaluation layer is stateless and only communicates to the database, it is easily scalable. The database layer is harder to scale, but several well-engineered solutions exist and can be utilized.

The evaluation layer was implemented using Java Spring, with later application to colored nets and reference nets in mind, where transition firings and binding searches become more advanced in the order of magnitudes.

As scalability is one of the proposed main features, the simulator is designed to be used with container and PaaS solutions in mind, especially Docker and Kubernetes. Kubernetes offers excellent means to keep an application robust and scale it up to several replicas. Automatic scaling solutions (based on CPU load) also exist.

Next things to address are the incorporation of existing tools and formats (like PNML) and the evaluation of graph databases (like Neo4j), as well as the transition to higher level Petri net formalisms.

References

1. Chiola, G., Ferscha, A.: Distributed simulation of Petri nets. IEEE Parallel Distrib. Technol. **1**(3), 33–50 (Aug 1993)
2. Röwekamp, J.H., Moldt, D., Feldmann, M.: Investigation of containerizing distributed Petri net simulations. In: Applications and Theory of Petri Nets and Concurrency. pp. 133–142 (2018)
3. Simon, M., Moldt, D.: Extending Renew's algorithms for distributed simulation. In: Cabac, L., Kristensen, L.M., Rölke, H. (eds.) PNSE'16. CEUR Workshop Proceedings, vol. 1591, pp. 173–192. CEUR-WS.org (2016)