# LibReDE: A Library for Resource Demand Estimation

[Demonstration Paper]

Simon Spinner
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
simon.spinner@kit.edu

Giuliano Casale
Imperial College London
London, UK
g.casale@imperial.ac.uk

Xiaoyun Zhu
VMware, Inc.
Palo Alto, US
xzhu@vmware.com

Samuel Kounev
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
kounev@kit.edu

## ABSTRACT

When creating a performance model, it is necessary to quantify the amount of resources consumed by an application serving individual requests. In distributed enterprise systems, these resource demands usually cannot be observed directly, their estimation is a major challenge. Different statistical approaches to resource demand estimation based on monitoring data have been proposed, e.g., using linear regression or Kalman filtering techniques. In this paper, we present LibReDE, a library of ready-to-use implementations of approaches to resource demand estimation that can be used for online and offline analysis. It is the first publicly available tool for this task and aims at supporting performance engineers during performance model construction. The library enables the quick comparison of the estimation accuracy of different approaches in a given context and thus helps to select an optimal one.

## 1. INTRODUCTION

A resource demand is the time a unit of work (e.g., request or transaction) spends obtaining service from a resource (e.g., CPU or hard disk) in a system. Resource demands are input parameters of widely used stochastic performance formalisms (e.g., Queueing Networks or Queueing Petri Nets). In order to obtain accurate performance predictions for a system, a performance engineer needs to determine representative values for the resource demands during performance model construction.

State-of-the-art monitoring tools can only provide aggregate resource usage statistics on a system- or per-process-level. However, in many applications one process may serve requests of different types with varying resource requirements (due to different computations, caching, etc.). As a result, the resource demands usually cannot be observed. Instead, we need to estimate resource demands based on the available aggregate monitoring data. Different approaches to resource demand estimation using statistical techniques, e.g., linear regression [4], Kalman filtering [6, 5], or non-linear optimization [3, 2], have been proposed. These approaches differ in their expected input measurements and their robustness to data anomalies. Furthermore, the approaches need to be parameterized correctly to yield good results. The selection of a suitable estimation approach and the optimization of its parameters usually requires experiments to validate the resulting resource demands.

Given that there are no publicly available implementations of estimation approaches, a performance engineer is currently forced to implement estimation approaches on his own. This is a time-consuming and error-prone task. In this paper, we present LibReDE, a library supporting performance engineers to determine resource demands by providing a set of ready-to-use implementations of estimation approaches. Based on the actual system and the available monitoring data, the estimation library can automatically determine a set of candidate estimation approaches and execute them. A performance engineer can then validate the resulting resource demand estimates and select the approach that yields the best results. Furthermore, the library also provides a framework that can be used as a basis by developers of estimation approaches. Through reuse, the effort for adapting existing estimation approaches or for implementing new ones, can be significantly reduced.

## 2. USAGE AND FEATURES

The library can be either executed as a standalone program or integrated in other programs as a Java library. The standalone program can be either called through a console interface or through a Matlab function. It is designed for offline analyses where the measurement traces are available beforehand. If executed on the console, the user must provide the monitoring data as comma-separated values stored in a file. If executed within Matlab, it is possible to pass Matlab arrays directly to the library without conversion. The shared library offers the same functionality through a


*ICPE'14,* March 22–26, 2014, Dublin, Ireland.
ACM 978-1-4503-2733-6/14/03.
http://dx.doi.org/10.1145/2568088.2576093.

Java API. Additionally, it contains functions to continuously add new measurement data to an estimator and update the estimates recursively over time. This makes the library also applicable for online analyses, e.g., in a self-adaptive system that uses a performance model for planning reconfigurations.

The library currently provides implementations of seven estimation approaches: (a) a least squares regression approach based on the Utilization Law [4], (b) a Kalman filter estimator based on observed average response times and utilization [6], (c) a Kalman filter estimator based on the Utilization Law [5], (d) a non-linear optimization algorithm minimizing the difference between observed and calculated average response times [3], (e) a non-linear optimization algorithm relying on utilization and average response time observations [2], (f) an approach based on the Service Demand Law apportioning the aggregate utilization based on the observed response times [1], and (g) an approach that approximates the demands with the observed response times (applicable in low load scenarios) [1]. The user can select one or more estimation approaches to execute. If none is specified, the library will automatically select a set of applicable approaches based on the structure of the estimation problem and the available monitoring data. All selected approaches are then executed to estimate the resource demands. The user gets an overview of the estimates from each approach.

The monitoring data must be available as time series data with associated timestamps for each sample. The library can work on time series with individual events (e.g., arrival times and response times of individual requests) or on fixed sampled time-aggregated data (e.g., average response times or average throughput). If the input data consists of time series with individual events, the library automatically computes the required time-aggregated data.

The behavior of the estimation approaches can be controlled by a set of parameters. The *step size* parameter determines the sampling interval over which the input data is aggregated. If the estimator is executed recursively, it also determines the interval in which the estimates are updated. The *estimation window* parameter defines a sliding window controlling how many samples are included when updating the estimates. A *start time* and *end time* for the estimation can be specified. Additionally, there may be parameters specific to a certain estimation approach.

# 3. LIBRARY DESIGN

Figure 1 shows the major components of LibReDE. The *estimation approach* instantiates concrete instances of an estimation algorithm, an observation model, and a state model. It then triggers and coordinates the estimation procedure. The *estimation algorithm* component implements the underlying statistical technique of an estimation approach (e.g., non-negative least squares regression or non-linear optimization). The *state model* component encodes apriori knowledge about the resource demands as state constraints and contains functions to determine initial estimates. The *observation model* component defines the relationship between the observed system metrics and the hidden resource demands. The state model and the observation model both access the *monitoring repository* and the *workload description* components. The monitoring repository stores the user-provided monitoring data and provides functions to query and aggregate this data. The workload description component provides information about the system services

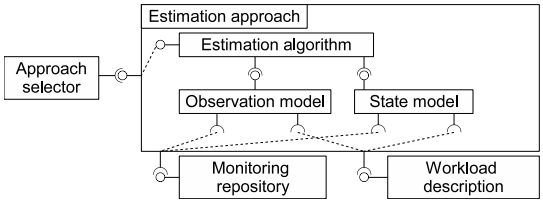


**Figure 1: UML component diagram**

and resources (including scheduling strategies and number of CPU cores). Finally, the *approach selector* instantiates all available estimation approaches and executes a subset verified to be applicable to a given estimation problem.

The library design enables a high level of reuse between different estimation approaches. The same estimation algorithms, observation models or state models can be shared between different estimation approaches. Thus the component-based structure facilitates the implementation of new estimation approaches.

# 4. CONCLUSION AND FUTURE WORK

In this paper, we presented a library for resource demand estimation. The library provides ready-to-use implementations of estimation approaches and can be used as a framework to implement new approaches. The source code of the library is open-source and publicly available[1]. The future development of the library will focus on integrating additional estimation approaches and supporting automatic cross-validation for the estimated resource demands. Furthermore, we also plan to implement techniques to automatically search for optimal parameter values (e.g., the step size) for an estimation approach.

# 5. REFERENCES

[1] F. Brosig, N. Huber, and S. Kounev. Automated extraction of architecture-level performance models of distributed component-based systems. In *26th IEEE/ACM Intl. Conf. On Automated Software Engineering*, Nov 2011.

[2] Z. Liu, L. Wynter, C. Xia, and F. Zhang. Parameter inference of queueing models for it systems using end-to-end measurements. *Perf. Eval.*, 63(1):36–60, 2006.

[3] D. Menascé. Computing missing service demand parameters for performance models. In *Proc. of the 2008 Computer Measurement Group (CMG) Conference*, pages 241–248, 2008.

[4] J. Rolia and V. Vetland. Parameter estimation for performance models of distributed application systems. In *Proc. of the 1995 conf. of the Centre for Advanced Studies on Collaborative research*, page 54. IBM Press, 1995.

[5] W. Wang, X. Huang, X. Qin, W. Zhang, J. Wei, and H. Zhong. Application-level cpu consumption estimation: Towards performance isolation of multi-tenancy web applications. In *2012 IEEE 5th Intl. Conf. on Cloud Computing*, pages 439–446, Jun 2012.

[6] T. Zheng, M. Woodside, and M. Litoiu. Performance model estimation and tracking using optimal filters. *IEEE Trans. on Soft. Eng.*, 34(3):391–406, 2008.

[1] http://www.descartes-research.net/tools