# Modeling Variations in Load Intensity over Time

Jóakim v. Kistowski
Karlsruhe Institute of Technology
Am Fasanengarten 5
76131 Karlsruhe, Germany
joakim.kistowski@student.kit.edu

Nikolas Herbst
Karlsruhe Institute of Technology
Am Fasanengarten 5
76131 Karlsruhe, Germany
herbst@kit.edu

Samuel Kounev
Karlsruhe Institute of Technology
Am Fasanengarten 5
76131 Karlsruhe, Germany
kounev@kit.edu

## ABSTRACT

Today's software systems are expected to deliver reliable performance under highly variable load intensities while at the same time making efficient use of dynamically allocated resources. Conventional benchmarking frameworks provide limited support for emulating such highly variable and dynamic load profiles and workload scenarios. Industrial benchmarks typically use workloads with constant or stepwise increasing load intensity, or they simply replay recorded workload traces. Based on this observation, we identify the need for means allowing flexible definition of load profiles and address this by introducing two meta-models at different abstraction levels. At the lower abstraction level, the *Descartes Load Intensity Meta-Model* (DLIM) offers a structured and accessible way of describing the load intensity over time by editing and combining mathematical functions. The *High-Level Descartes Load Intensity Meta-Model* (HLDLIM) allows the description of load variations using few defined parameters that characterize the seasonal patterns, trends, bursts and noise parts. We demonstrate that both meta-models are capable of capturing real-world load profiles with acceptable accuracy through comparison with a real life trace.

## Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: Performance of Systems—*Modeling Techniques*

## General Terms

Benchmarking, Workload, Modeling

## Keywords

Load Intensity Variation, Load Profile, Open Workloads, Meta-Modeling, Transformation, Model Extraction

## 1. INTRODUCTION

Today's cloud and web-based IT services need to handle huge numbers of concurrent users. Customers access services independently of one another and expect reliable quality-of-service under highly variable and dynamic load intensities. In this context, any knowledge about a service's load intensity profile is becoming a crucial information for managing the underlying IT resource landscape.

Load profiles with large amounts of concurrent users are typically strongly influenced by human habits, trends, and events. This includes strong deterministic factors such as time of the day, day of the week, common working hours and planned events.

Common benchmarking frameworks such as Faban[1], Rain [2], and JMeter[2] allow job injection rates to be configured either to constant values, stepwise increasing rates (e.g., for stress tests), or rates based on recorded workload traces. A flexible, intuitive way of defining, interchanging and modifying load profiles is to the best of our knowledge not available yet. For the purpose of benchmarking and comparing the performance of cloud-based service offerings in a realistic manner, a reference load profile or a set of load profiles should become a defined or even standardized configuration input.

Our work aims at closing the gap between highly dynamic real-world load intensity profiles and the currently insufficient flexibility in handling variable load profiles in the benchmarking and dynamic system management domains. Specifically, we create load intensity models for two major use-cases: Firstly, we support the creation of custom load intensity behaviors, specifically designed to illicit certain system behavior. Secondly, we realize a (partly) automatic approximation of existing load intensity traces in form of a model instance. We propose two meta-models at different abstraction level: The *High-Level Descartes Load Intensity Meta-Model* (HLDLIM) allows the description of load variations using few defined parameters that characterize the seasonal patterns, trends as well as bursts and noise parts. At the lower abstraction level, the *Descartes Load Intensity Meta-Model* (DLIM) offers a structured and accessible way of describing load intensity profiles over time by editing and combining mathematical functions. We are working on a transformation from HLDLIM to DLIM model instances for the support of further model refinements and automatic DLIM calibration features that we envision for future work. For the handling and instantiation of the proposed load intensity models, we provide the LIMBO toolkit[3] [9]. We demonstrate by using an example real life trace that

[1]Faban `http://faban.org`
[2]JMeter `http://jmeter.apache.org`
[3]LIMBO `http://www.descartes-research.net/tools/`


*LT* '14 Dublin, Ireland

both meta-models are capable of capturing real-world load profiles with acceptable accuracy.

# 2. FOUNDATIONS

Both DLIM and HLDLIM models capture load intensity of open workloads by specifying, e.g., user, request, or job arrival rates.

## 2.1 Open and Closed Workloads

Schroeder et al.[8] define open and closed workloads as follows:

**Closed workload:** new job arrivals are only triggered by job completions

**Open workload:** new jobs arrive independently of job completions

Our meta-models do not make any assumptions about the completion times of the work units. Additionally, we assume that users in a typical cloud environment are completely unaware of one another and access a software service without having any knowledge of other users' behavior. Furthermore, work units induce statistically indistinguishable resource demands on the underlying hardware. Accordingly, DLIM or HLDLIM instances describe the intensity profile of an open workload.

## 2.2 Load Intensity

In this work *load intensity* is the function describing *arrival rates* of workload units over time. We define the *arrival rate* $r(t)$ at time $t$ as follows:

$$r(t) = R'(t)$$
$$\text{with } R(t) = |\{u_{t0} | t0 \leq t\}|$$

where $R(t)$ being the amount of all *work units* $u_{t0}$, with their respective *arrival time* $t0$, that have arrived up until time $t$.

# 3. RELATED WORK

Several models with the purpose to describe and generate workloads with an variable intensity have been proposed so far. However, these models differ from our approach in two key aspects: They are either purely statistical or they the key emphasis is put on the behavior and structure of the actual units of work they dispatch (or both). Our model tries to exclude both of these aspects by solely modeling the variations in load intensity in a deterministic fashion. Thus, we group related work in at least one of the following three categories:

**User behavior models**: Hoorn et al.[10], Roy et al.[6], and Beich et al.[2] create workload models that capture the behavior and tasks triggered by certain types of users. Zakay et al.[12] partition workloads into users types and then sample workload traces for each type in order to describe user behavior. This has a direct impact on the work type itself and differs greatly from our approach of only modeling user arrival rates. Models of this kind can be easily combined with DLIM, since they model what the user does after his/her arrival as modeled by DLIM.

**Workload modeling** with a focus on work units and resource demands: These models focus on the actual unit of work and model it in a way that estimates a specific resource use. Casale et al.[3] do so with a focus on burstiness, while Barford et al.[1] put focus on file distribution and popularity.

**Statistical models**: These models try to extract workload intensity into a few statistical numbers that describe the workload. Feitelson[4] creates a statistical model for parallel job schedulers, while Menascé et al.[5] analyze workloads on the Business, Session, Function, and HTTP Request layer. These approaches differ from our deterministic approach as they do not capture the load intensity behavior changes over time.

# 4. DESCARTES LOAD INTENSITY MODEL

The Descartes Load Intensity Model (DLIM) describes request arrival rates over time and is visualized in Fig. 1. Specifically, the model is aimed at describing the abstract variations of work unit arrival rates into characteristic load intensity behaviors. DLIM offers a way to define a piece-wise mathematical function, which is to be used for the approximation of variable arrival rates, with support for (partial) periodicity, flexibility for easy adaptations and incorporation of unplanned events, composability of DLIM instances.

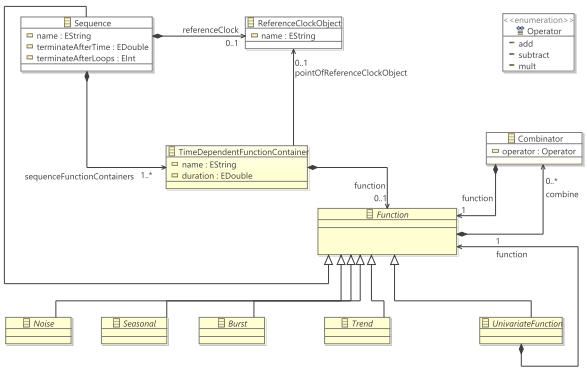


**Figure 1: The Descartes Load Intensity Meta-Model without the child implementations of the abstract *Noise*, *Burst*, *Seasonal*, and *Trend*.**

## 4.1 Sequence

The DLIM root element is the *Sequence*. The Sequence holds an ordered List of *TimeDependentFunctionContainers*, which describe the basic arrival rate functions (see section 4.2). The *TimeDependentFunctionContainers* are executed in sequence. This execution repeats as many times as indicated by the *terminateAfterLoops* attribute. Alternatively the sequence repeats for the time indicated by the *terminateAfterTime* attribute. If both are set, we calculate the final duration by

$$finalDuration = min(terminateAfterLoops * loopDuration, terminateAfterTime).$$

Infinite sequences are not allowed in order to guarantee termination.

## 4.2 TimeDependentFunctionContainer

The *TimeDependentFunctionContainer* contains functions that describe the arrival rate on a basic mathematical level. It contains one instance of a child of the abstract class *Function* (note that *Sequence* also inherits from *Function*). Any *Function* can be combined with other *Functions* using a *Combinator*, which results in a *TimeDependentFunctionContainer* carrying an entire tree of functions. The *TimeDependentFunctionContainer* describes its arrival rates for a

set *duration*, after which the next *TimeDependentFunctionContainer* in the parent *Sequence's* list is executed.

### 4.3 Function

The *Function* is the abstract parent class to mathematical functions contained within the *TimeDependentFunctionContainer*. A number of non-abstract children are provided that can be used as an *Function*. The most notable concrete *Function* is the *Sequence*, which effectively means that any *TimeDependentFunctionContainer* may hold a *Sequence* in its *Function* tree. The other concrete *Functions* fall into one of the following categories (each represented by their abstract super-class):

**Seasonal:** Functions that describe seasonal, recurring patterns (such as *sin*).

**Bursts:** Functions that describe singular bursts that reach a certain *peak* value at a *peakTime* and then return to the *base* value, from which they started.

**Noise:** Functions that describe random noise.

**Trends:** Functions that describe monotonic trends. These Functions are interpolated *Functions*, described by their *start* and *end* values.

A *Function* holds a list of *Combinators*. A *Combinator* allows the combination of this *Function*'s arrival rates with the arrival rates generated by other concurrently running *Functions*. The *Combinator* contains operators such as $+, *$.

## 5. HIGH-LEVEL DLIM

While DLIM offers a convenient way of structuring and ordering functions for the description of load intensity variations, it offers little abstracted knowledge about those variations. HLDLIM addresses this issue by describing load intensity variations with a limited number of parameters. These parameters can then be used to quickly define and characterize a load intensity model. Inspired by the time-series decomposition approach in BFAST [11], it is split into a *Seasonal* and *Trend* part. Additionally, it features a *Burst* and *Noise* part

### 5.1 Seasonal Part

The *Seasonal* part describes the underlying dummy function that repeats after every seasonal duration (e.g. every day in a month long load intensity description). HLDLIM describes it using the following parameters:

**Period:** The duration of a single iteration of the seasonal function.

**Number of Peaks:** The amount of arrival rate peaks within a single iteration of the seasonal function.

**Base Arrival Rate Level:** The arrival rate at the beginning and end of a seasonal function iteration.

**Base Arrival Rate Level between Peaks:** The local arrival rate minimum between two peaks.

**First Peak Arrival Rate** and **Last Peak Arrival Rate**. All other peak arrival rates are derived from *First Peak Arrival Rate* and *Last Peak Arrival Rate* using linear interpolation.

**Interval with Peaks:** The time interval during which the peaks are defined. This interval is centered around $Period/2$.

**Seasonal Shape:** The shape of the function interpolating between peaks and base levels. The Shape can be any DLIM *Trend Function*.

### 5.2 Trend Part

The *Trend* part describes the overarching function that describes the change in load intensity variation over several seasonal periods. In contrast to BFAST, the HLDLIM *Trend* can be either additive or multiplicative. The *Trend* part is defined using the following parameters:

**Number of Seasonal Periods within one Trend:** The duration of each *Trend* segment. It must be a positive integer number. As a result, the *Trend* segments' duration is always a multiple of the *Seasonal Period*.

**Trend Shape:** The mathematical function used to describe the *Trend* segments. The shape can be any DLIM *Trend Function*.

**Operator:** The operator with which the *Trend* is applied to the *Seasonal* part.

**List of Seasonal Arrival Rate Peaks:** The arrival rate at the beginning and end of the *Trend* segments. The *Trend* segment functions are defined so that they always begin and end at the largest *Seasonal Peak*. As a result, the values contained in this list define the resulting maximum peak after *Trend* application at the corresponding point in time. The point in time at which each arrival rate in this list is defined is always the time of the largest peak in a *Seasonal* iteration.

### 5.3 Burst Part

HLDLIM allows the definition of recurring bursts, which are added onto the existing arrival rate behavior. The *Burst* part is thus defined using the following parameters:

**First Burst Offset:** The time at which the first burst peaks.

**Inter Burst Period:** The time between two peaks.

**Burst Peak Arrival Rate:** The arrival rate added by the burst at its peak time.

**Burst Width:** The time interval in which a burst is executed. The peak takes place at $BurstWidth/2$.

### 5.4 Noise Part

HLDLIM allows the addition of a uniform distributed noise function defined by its **Minimum Noise Rate** and **Maximum Noise Rate**. Other Noise distributions can easily be added to DLIM instances.

## 6. EVALUATION

For our preliminary model accuracy evaluation we use the page request trace of the German Wikipedia. We extract DLIM and HLDLIM instances from the trace using a best-effort approach. We then compare curve similarity based on an error metric defined on the basis of the Dynamic Time Warping algorithm (in our case, the fDTW implementation [7]). We normalize the DTW distance by dividing it by the number of samples and the maximum arrival rate. We extract the DLIM instance by modeling the days within the Wikipedia trace as a *seasonal* dummy function. This function approximates the user request rate within one day. It is lowest at night and peaks right before and after lunch break. Next, we multiply a trend onto the *seasonal* function. This trend interpolates between the highest daily peaks and tries to fit *seasonal* function onto the trace's arrival rates. Additional major deviations between model and trace are modeled as additive bursts. Then, we fit the extracted DLIM instance into the HLDLIM parameters. For this we use the

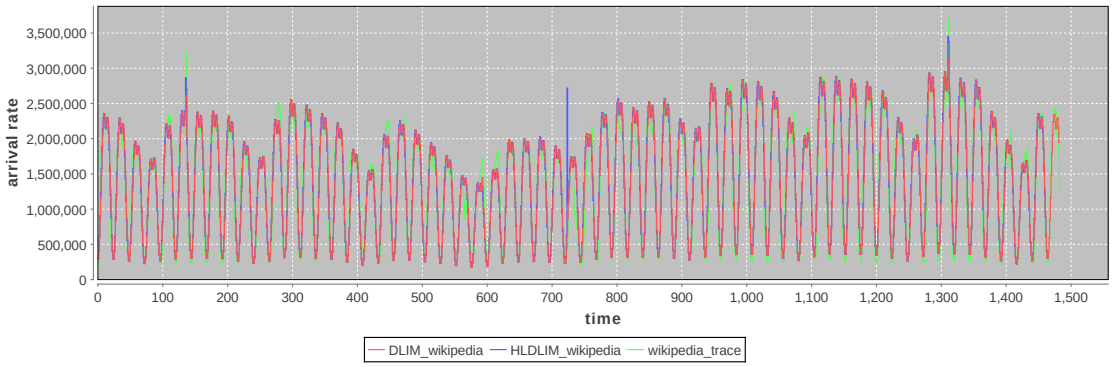


**Figure 2: The arrival rates of the Wikipedia request trace and the extracted model instances.**

median values of the respective properties in the DLIM instance. The *Inner Base Arrival Rate* is the median arrival rate of the arrival rate between the extracted peaks, while the *Inter Burst Period* is the median wait time between two bursts. The errors between the model instances and the original trace are as follows:

| | mean error | median error | DTW based curve difference |
|---|---|---|---|
| **DLIM** | 16.08% | 8.95% | 0.0222 |
| **HLDLIM** | 16.25% | 9.64% | 0.0224 |

Overall, the extracted DLIM instance seems to fit the trace's arrival rates with acceptable accuracy. All major error are caused by days which deviate from the median pattern (which is extracted as our *seasonal* dummy). The upside of the *seasonal* dummy approach however is increased model instance utility. Amongst other things, it simplifies the extraction of HLDLIM parameters.

On the other hand, while the accuracy of the HLDLIM instance is remarkably close to the DLIM instance, we have one particular observation: The Wikipedia trace does not mesh well with recurring bursts. The burst at time-stamp 724 is by far the biggest deviation from the original trace.

## 7. CONCLUSIONS

This Paper addresses the need for modeling variable load profiles for custom benchmarking or trace approximation. With DLIM and HLDLIM we introduce two load intensity models that address this need. DLIM offers a way of expressing load intensity profiles using piece-wise mathematical functions, while HLDLIM offers a concise way of expressing a profile's characteristics. Future work entails the creation of model-to-model transformations between these two models, as well as automatic and reproducible model-from-trace extraction processes. We also envision additional future use-cases for load intensity models, such as load intensity forecasting.

## 8. REFERENCES

[1] P. Barford and M. Crovella. Generating representative web workloads for network and server performance evaluation. In *Proceedings of the 1998 ACM SIGMETRICS*, pages 151–160, New York, NY, USA, 1998. ACM.

[2] A. Beitch, B. Liu, T. Yung, R. Griffith, A. Fox, and D. A. Patterson. Rain: A workload generation toolkit for cloud computing applications. Technical Report UCB/EECS-2010-14, EECS Department, University of California, Berkeley, Feb 2010.

[3] G. Casale, A. Kalbasi, D. Krishnamurthy, and J. Rolia. Burn: Enabling workload burstiness in customized service benchmarks. *IEEE Transactions on Software Engineering*, 38(4):778–793, 2012.

[4] D. Feitelson. Workload modeling for performance evaluation. In M. Calzarossa and S. Tucci, editors, *Performance Evaluation of Complex Systems: Techniques and Tools*, volume 2459 of *Lecture Notes in Computer Science*, pages 114–141. Springer Berlin Heidelberg, 2002.

[5] D. A. Menascé, V. A. F. Almeida, R. Riedi, F. Ribeiro, R. Fonseca, and W. Meira, Jr. A hierarchical and multiscale approach to analyze e-business workloads. *Perform. Eval.*, 54(1):33–57, Sept. 2003.

[6] S. Roy, T. Begin, and P. Goncalves. A complete framework for modelling and generating workload volatility of a vod system. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International*, pages 1168–1174, 2013.

[7] S. Salvador and P. Chan. Toward accurate dynamic time warping in linear time and space. *Intell. Data Anal.*, 11(5):561–580, Oct. 2007.

[8] B. Schroeder, A. Wierman, and M. Harchol-Balter. Open versus closed: a cautionary tale. In *Proceedings of the 3rd conference on Networked Systems Design & Implementation - Volume 3*, NSDI'06, pages 18–18, Berkeley, CA, USA, 2006. USENIX Association.

[9] J. v. Kistowski. *Modeling the Variation of Load Intensities*. Karlsruhe Institute of Technology, Karlsruhe, Germany, April 2014. to appear.

[10] A. van Hoorn, M. Rohr, and W. Hasselbring. Generating probabilistic and intensity-varying workload for web-based software systems. In *Proceedings of the SIPEW'08*, pages 124–143, Berlin, Heidelberg, 2008. Springer-Verlag.

[11] J. Verbesselt, R. Hyndman, G. Newnham, and D. Culvenor. Detecting trend and seasonal changes in satellite image time series. *Remote Sensing of Environment*, 114(1):106 – 115, 2010.

[12] N. Zakay and D. G. Feitelson. Workload resampling for performance evaluation of parallel job schedulers. In *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, ICPE '13, pages 149–160, New York, NY, USA, 2013. ACM.