

# Summary Paper of *Don't Hold My Data Hostage – A Case For Client Protocol Redesign*

Authors of Original Paper: Mark Raasveldt and Hannes Mühleisen

Hannes Pohnke  
pohnke@tu-berlin.de  
TU Berlin  
Berlin, Germany

## ABSTRACT

Traditionally, database query processing and ML tasks are executed on separate, dedicated systems, but the current trend goes towards integrated data analysis pipelines that combine both tasks. In state of the art systems, orchestration of those two tasks still is inefficient due to expensive data transfer and missed global optimization potential. The paper we are summarizing, "Don't Hold My Data Hostage – A Case For Client Protocol Redesign" (DHMDH) by Mark Raasveldt and Hannes Mühleisen, addresses this problem by investigating the high cost of transferring large data from databases to the client programs, which can be much more time consuming than the actual query execution. The authors explore and analyse current serialization methods, that are used in database systems and identify their inefficiencies through various experiments. They also introduce a new columnar serialization method that can significantly enhance data transfer performance. By improving the data transfer, this approach could be a step towards efficiently combining database and machine learning systems.

## KEYWORDS

Databases, Client Protocols, Data Export

### ACM Reference Format:

Hannes Pohnke. 2018. Summary Paper of *Don't Hold My Data Hostage – A Case For Client Protocol Redesign*: Authors of Original Paper: Mark Raasveldt and Hannes Mühleisen. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 1 page. <https://doi.org/XXXXXXX.XXXXXXX>

## 1 INTRODUCTION

Having to transfer a lot of data from a database to a client program is a standard procedure when conducting complex machine learning (ML) tasks. The problem is that this transfer process is very slow, especially because database servers are typically not located on the same system as the ML workloads.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).  
*Conference acronym 'XX, June 03–05, 2018, Woodstock, NY*

© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.  
ACM ISBN 978-1-4503-XXXX-X/18/06  
<https://doi.org/XXXXXXX.XXXXXXX>

When the DHMDH paper was introduced in 2017 most research in this field, including the authors [? ], focused on performing computations within the database, avoiding the need for data transfer. Since then a lot of approaches for optimizing the transfer arose, which can be broadly categorized in server-side and client-side optimizations [? ]. Raasveldt and Mühleisen chose a client-side approach, as they already realized in 2017 that biggest and easiest to implement optimization potential lies in addressing the overhead of (de)serialization of the data. They use the term of result set serialization (RSS), which refers to converting data into a format suitable for transfer and show its significant impact on system performance by using a standard SQL query, sending it through an ODBC connector and then fetching the results for different data management systems.

Back in the day popular ML tools, including RapidMiner, Weka, R's ML packages, and Python-based toolkits like SciKit-Learn and TensorFlow, only supported bulk data transfer from databases [? ]. Therefore, users had to manually load data that was already in a table format into these tools. Because of the mentioned inefficiencies this often resulted in smaller data samples being used, which is generally a bad thing in ML. This paper examines existing serialization formats, analyzing their performance across different data sets and network scenarios. It explores the design space of serialization formats through experiments, highlighting different trade-offs in client protocol design.

Key contributions include:

- Benchmarking current RSS methods, to identify data transfer inefficiencies.
- Investigating techniques for creating efficient serialization methods.
- Proposing a new column-based serialization method with significant performance improvements.

## 2 SUMMARY

Fazit nochmal labern was impact ist und wie sich anscheinend in database solutions durchsetzen aus mehreren argumenten wenn das tatsächlich der fall ist? kp ich glaube sogar nicht so wie sich connector x anhört

## 3 ANALYSIS