Bachelor Thesis

Integration of MongoDB in PM4Py for Preprocessing Event Data and Discover Process Models

Tom-Hendrik Hülsmann

Agenda

- Introduction
- Approach
- Evaluation
- Conclusion

Introduction

"Integration of a database system into PM4Py"

PM4Py

- Open Source Process Mining Library
- Developed by the *Chair of Process and Data Science* at RWTH Aachen and the *Fraunhofer FIT process mining group*
- First release in December 2018
- Focus on performance and integration with other Data Science tools

PM4Py [1]

Goals

- Extend PM4Py by integrating a database component using MongoDB
- Python library acting as a connector
- Storing event logs
- Perform basic preprocessing and process mining operations on the database

MongoDB

- Document based database system
- Popular in the industry
- Open Source *(Server Side Public License)*
- Build for large amounts of data, highly scalable

[2]

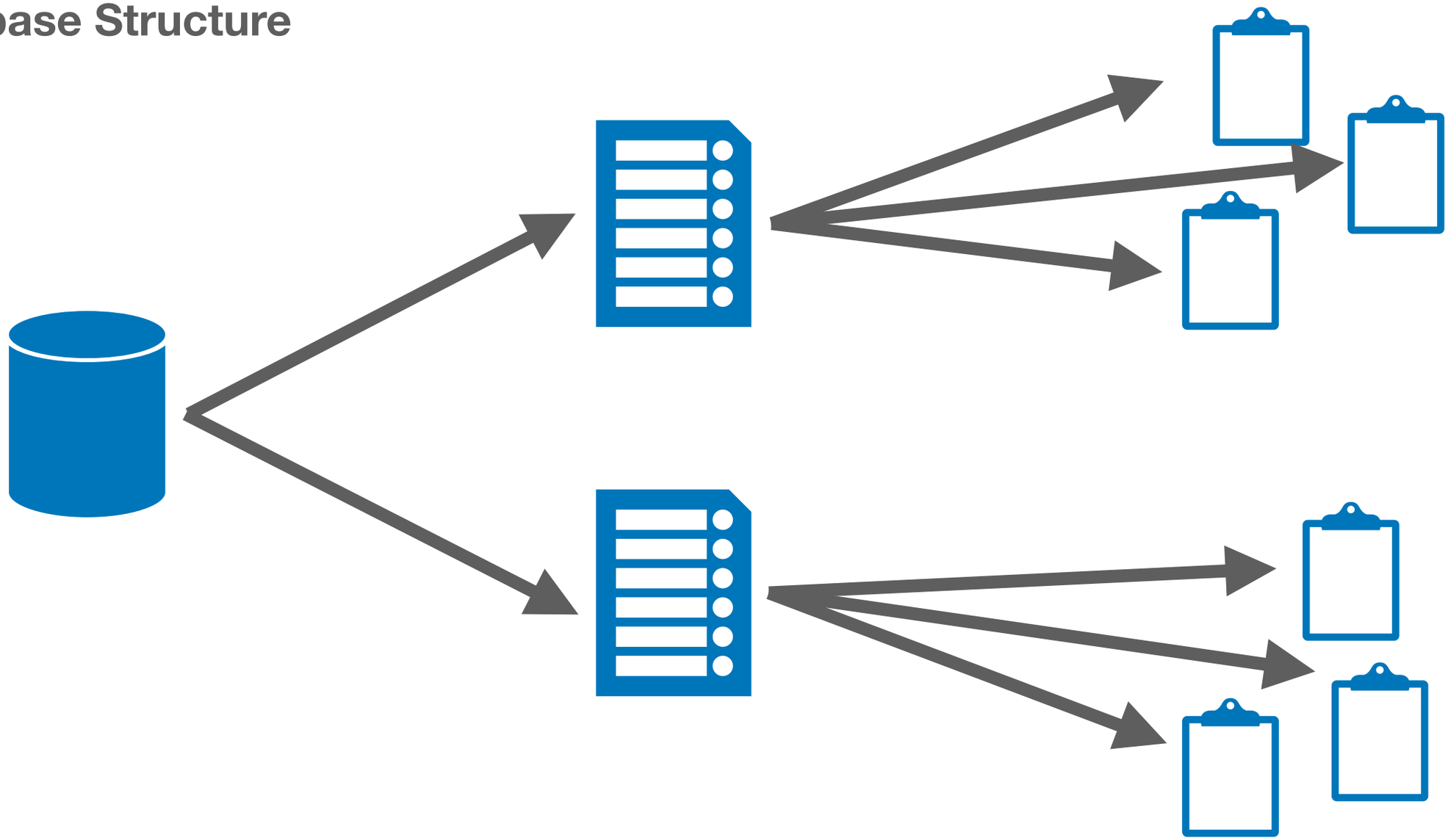
MongoDB

Advantages

- Integration into existing systems
- Performance
- Ease of use
- First-party tool support for Python (PyMongo)

MongoDB

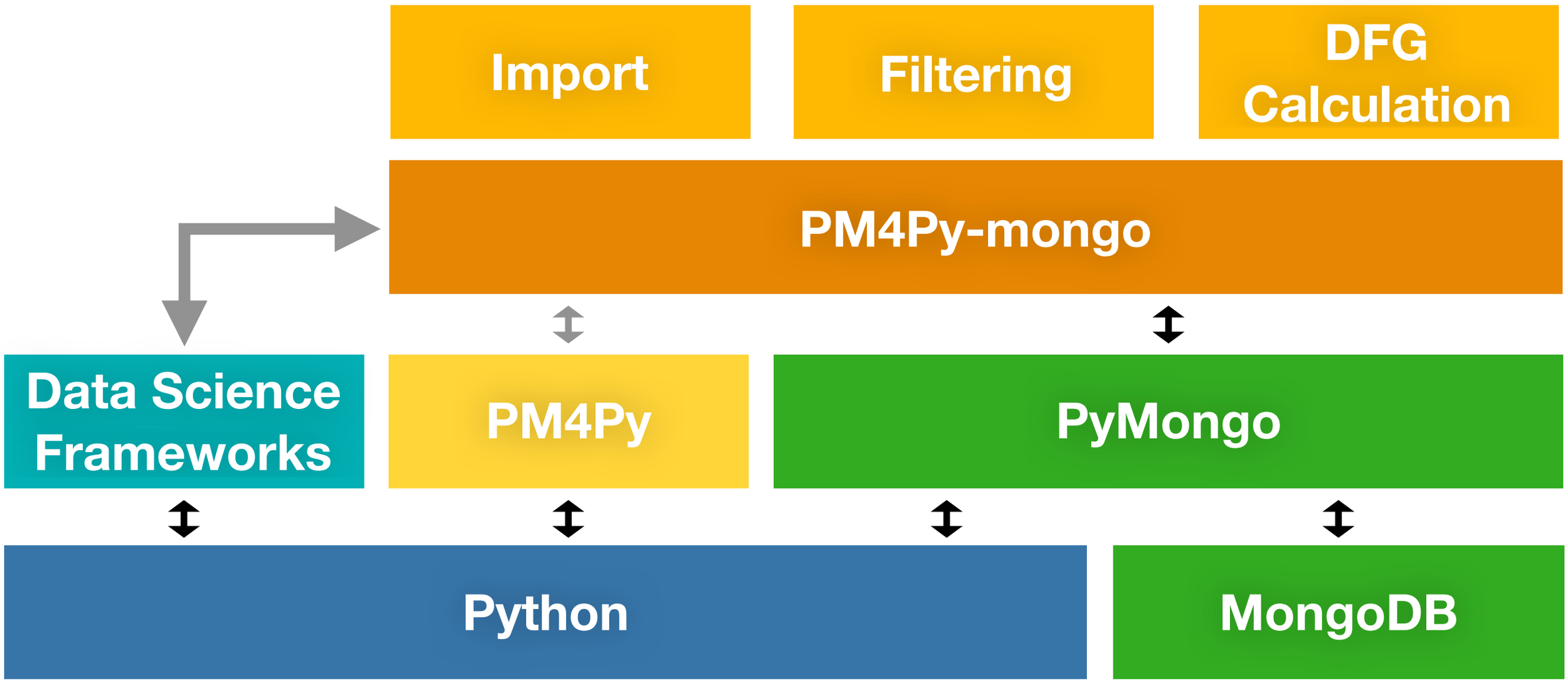
Database Structure



Database **Collections** **Documents**

Approach

Structure


Import
Filtering
DFG Calculation
PM4Py-mongo
Data Science Frameworks
PM4Py
PyMongo
Python
MongoDB

Import

CSV File

Order No.	Status	Product No.	Time
84578	Order Received	9091	2019-15-02 08:12:52
84590	Order Cancelled	4056	2019-15-02 08:23:43
84578	In Processing	9091	2019-15-02 09:05:11
…	…	…	…

Database


```
{
    "_id": ObjectId(DNW293ndnJWKNKdlw),
    "case": "84578",
    "activity": "Order Reeceived",
    "product_no": 9091,
    "timestamp": 2019-15-02 08:12:52
},
{
    "_id": ObjectId(KWNDKWO0293smwkS),
    "case": "84590",
    "activity": "Order Cancelled",
    "product_no": 4056,
    "timestamp": 2019-15-02 08:23:43
},
{
    "_id": ObjectId(POAnW029MW90wM),
    "case": "84578",
    "activity": "In Processing",
    "product_no": 9091,
    "timestamp": 2019-15-02 09:05:11
},

...
```

Import

- Import a given CSV file into the database
- Either create a new database, add to an existing database or overwrite existing database
- Uses Python CSV reader
- Adds events to the database in batches (different approaches were considered)

Filtering

- Event level
  - Attribute values
  - Attribute ranges
  - Time range
- Case level
  - Start / end activity
  - Performance range
  - Attribute values
  - Time range (intersection)
  - Variants

Filtering

- Offers common filtering operations
  - On event level
  - On case level
- Filtering results are stored in a new collection
- Implemented using *aggregation pipelines*

DFG Calculation

- Calculate frequency-based DFG
- Calculate performance-based DFG
  - average, minimum, maximum
- Computation using *Map Reduce* and aggregation pipelines

DFG Calculation

- Results are stored as documents in the database

```
{
   "_id":   {"n1": "received", "n2": "processed"},
   "value": 15
}
```

- Export DFG edges as a dictionary for use in PM4Py

```
("received", "processed") : 15
```

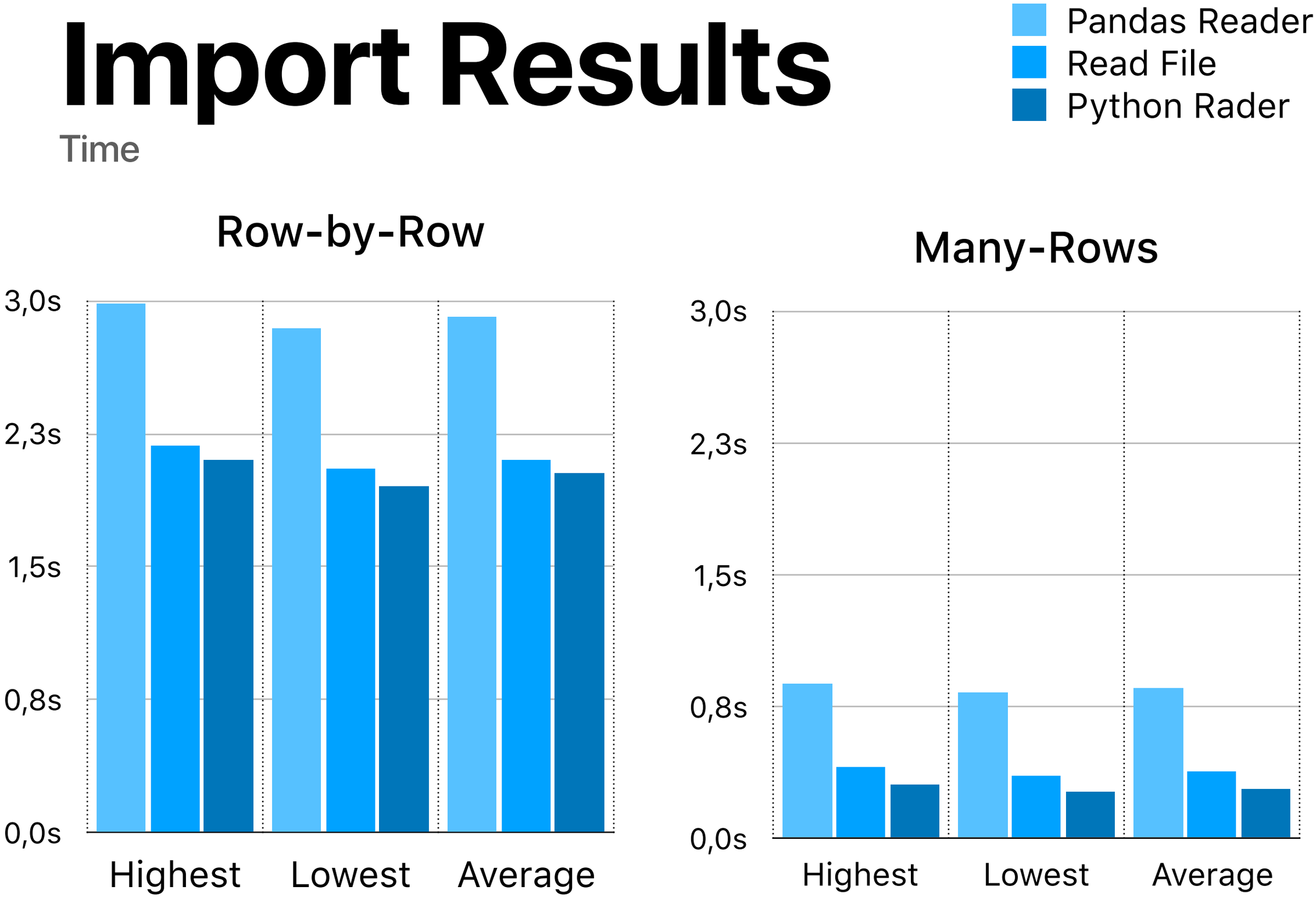
Evaluation

Import Test Case

- Different Approaches
  - Row-by-Row
  - Many-by-Many
- *Receipt* Dataset
  - 8500 events, 1400 cases
  - 2.4 MB
- 25 consecutive imports

Import Results

Time



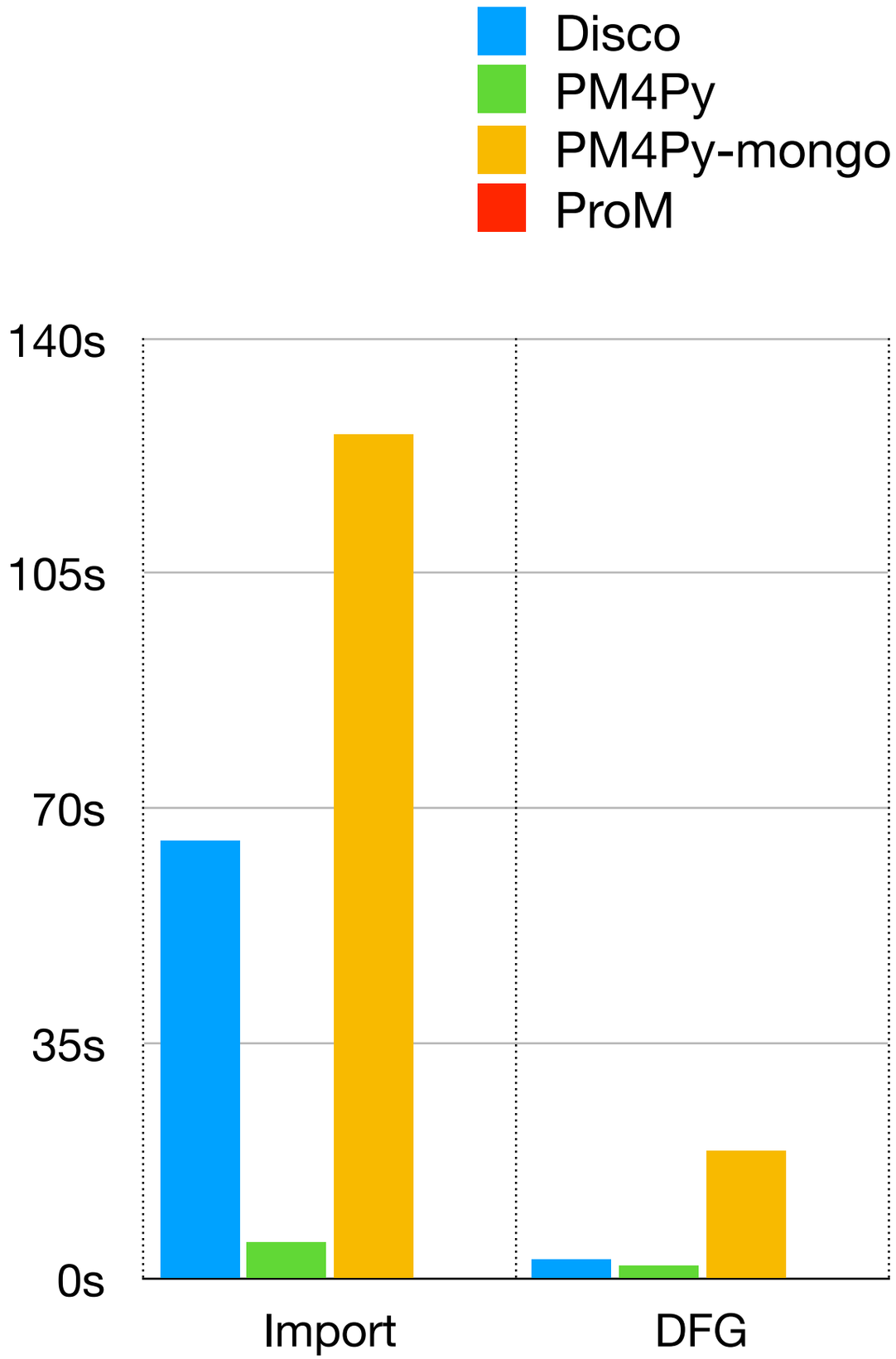
Datasets

- *bpic2019* Dataset (Large Log)
  - > 1.5 million events, > 250,000 cases
  - 527.5 MB
- *roadtraffic* Dataset (Medium Log)
  - > 500,000 events, > 150,000 cases
  - 47.9 MB

Test Case I

Large Log - DFG

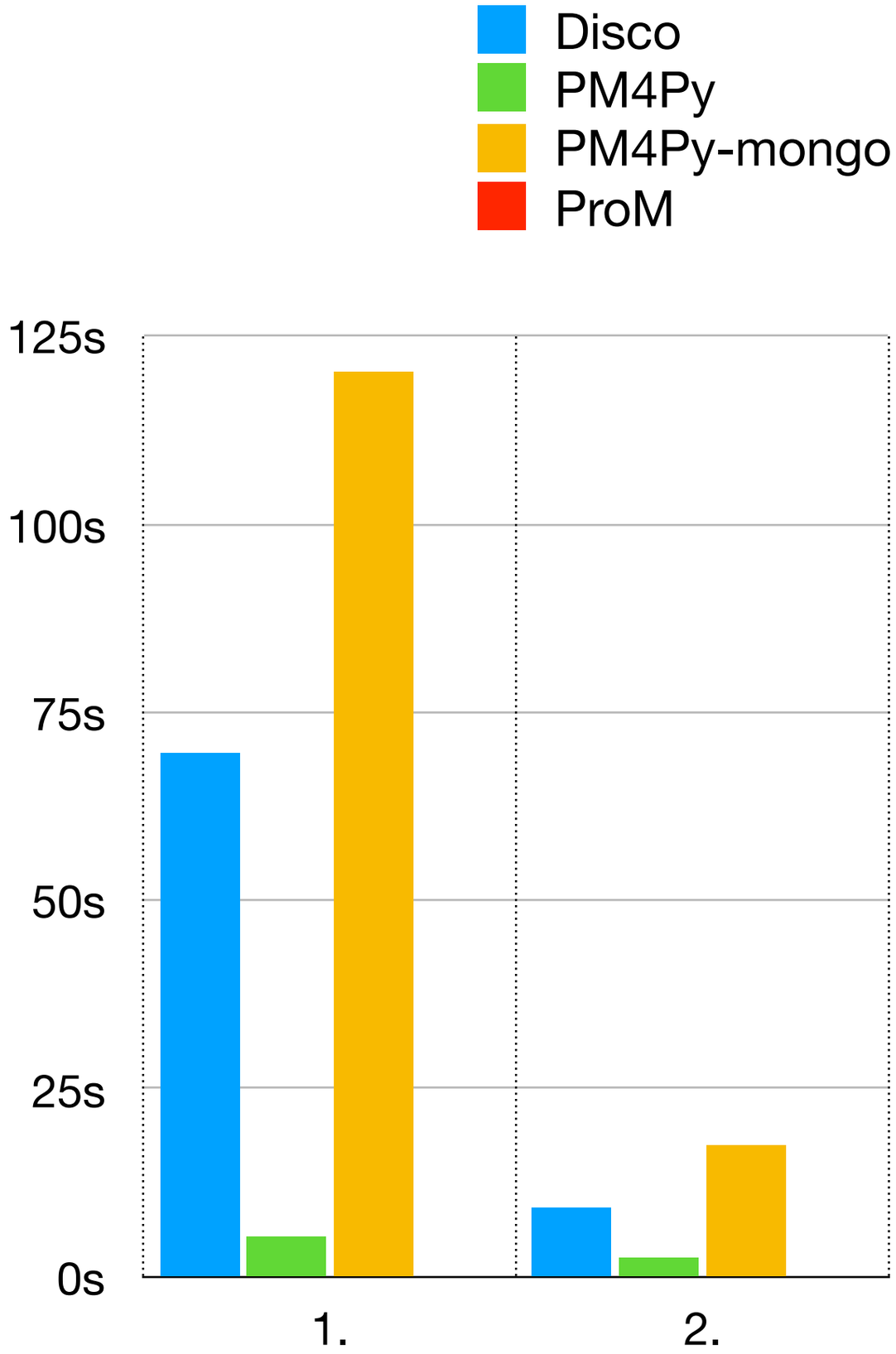
- DFG calculation performance.
- Task
  - *"Given the bpic2019 dataset, calculate the frequency based DFG."*
- ProM crashed during import.



Test Case II

Large Log - Complex Filtering

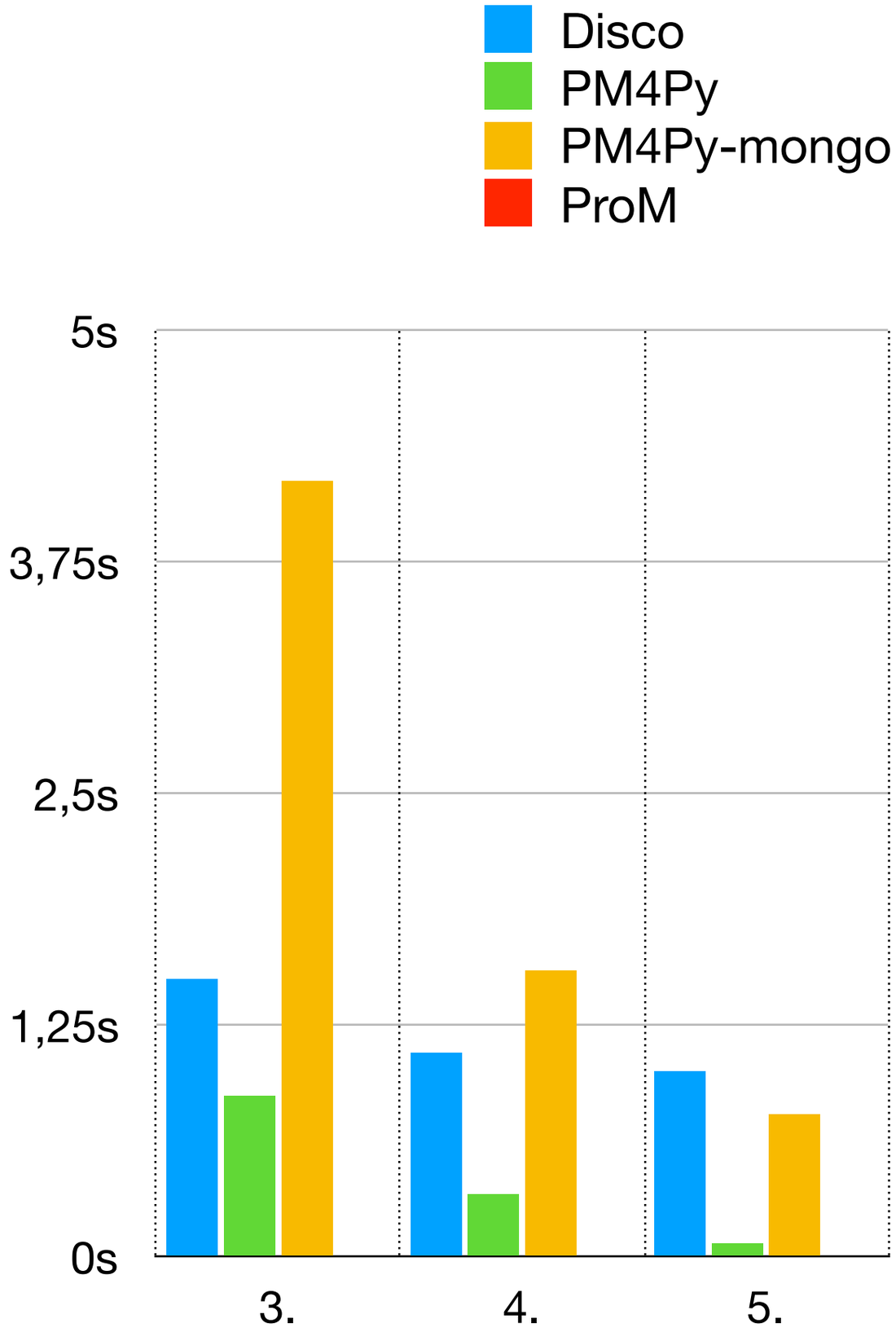
- Tasks
    1. Import
    2. End activity filter
    3. Performance filter
    4. Frequent variants filter
    5. Performance-based DFG
    6. Discover process model
- ProM crashed during import.



Test Case II

Large Log - Complex Filtering

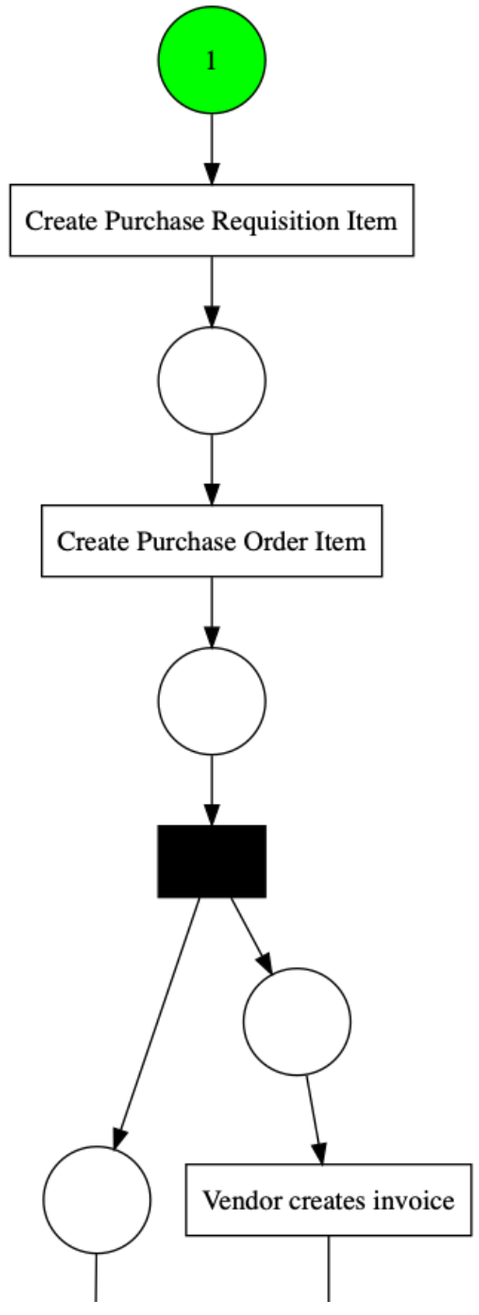
- Tasks
    1. Import
    2. End activity filter
    3. Performance filter
    4. Frequent variants filter
    5. Performance-based DFG
    6. Discover process model
- ProM crashed during import.



Test Case II

Large Log - Complex Filtering

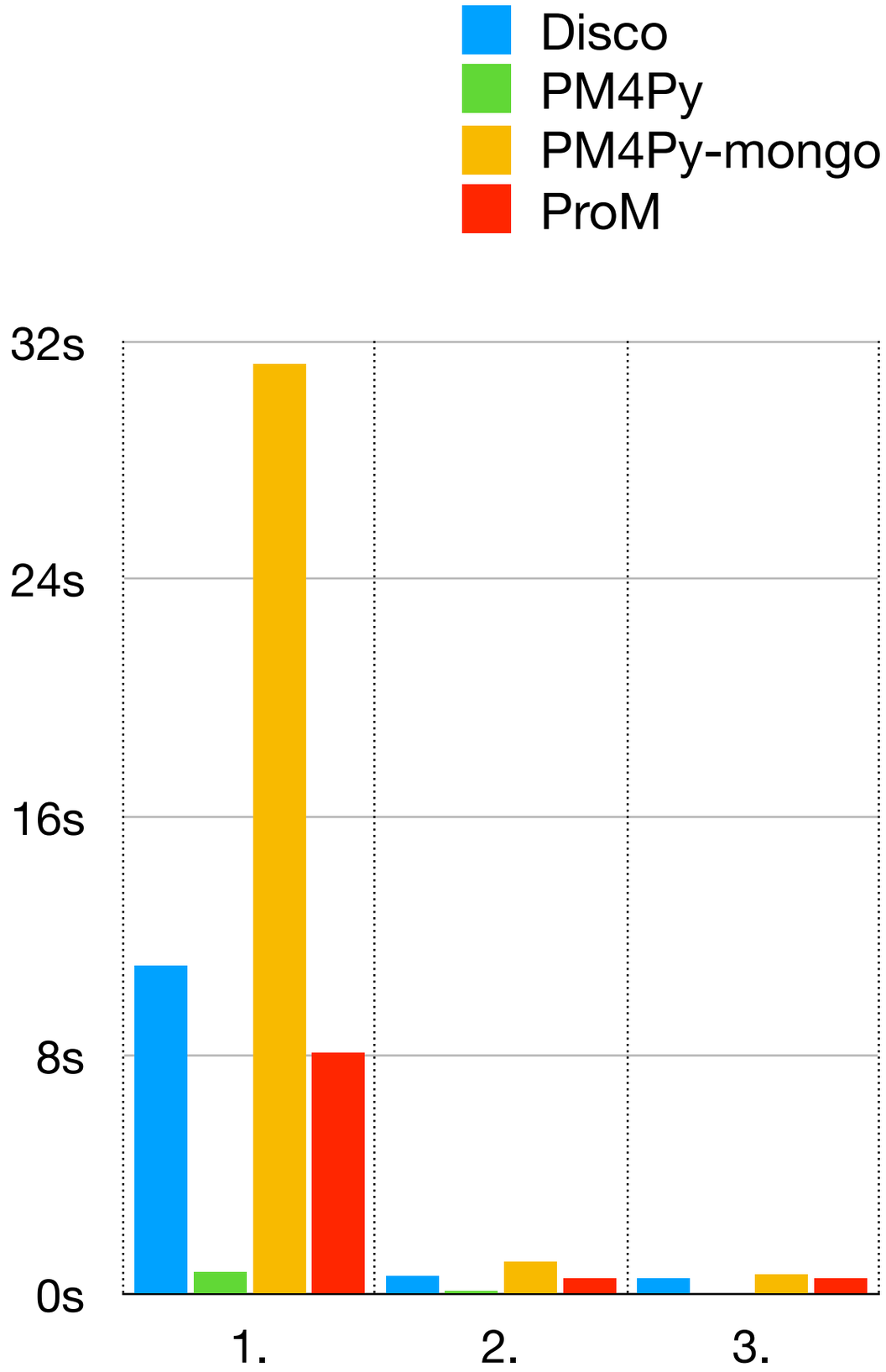
- Tasks
    1. Import
    2. End activity filter
    3. Performance filter
    4. Frequent variants filter
    5. Performance-based DFG
    6. Discover process model
- ProM crashed during import.



Test Case III

Medium Log - Simple Filtering

- Tasks
  1. Import
  2. Resource filter
  3. Performance-based DFG

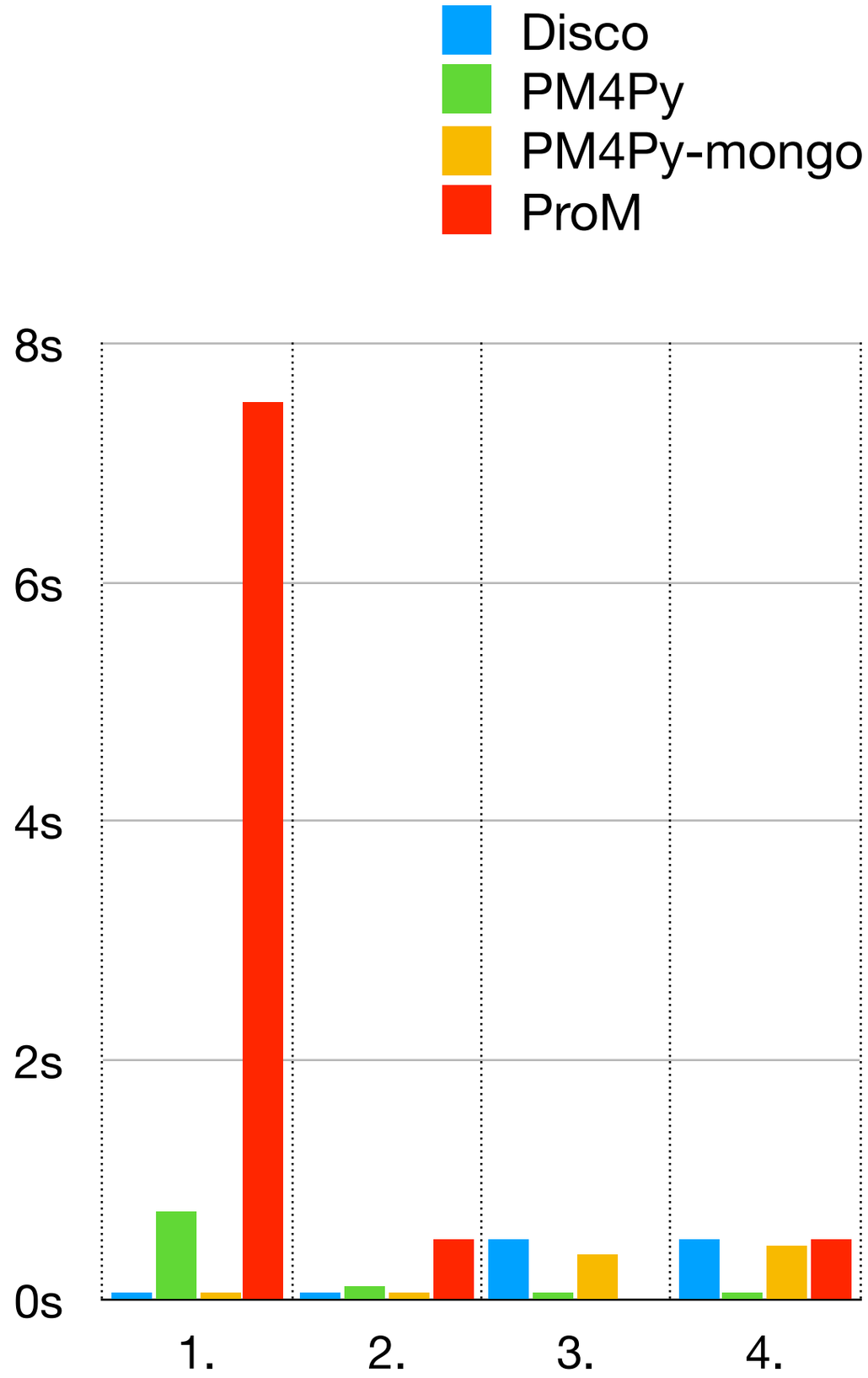


Test Case IV

Medium Log - Reuse

- Reuse results from Test Case III if possible

- Tasks
  1. Import
  2. Resource filter
  3. Time range filter
  4. Frequency-based DFG



Conclusion

Features

- Offers the most common data handling / preprocessing capabilities
  - Import
  - Event level filters
  - Case level filters
- Can compute basic process mining structure (DFG)
- Can be used with PM4Py to enable a full process mining pipeline

Limitations

- In most areas worse performance
- Limited feature set
- Only preprocessing and DFG calculation
- No conformance checking, model enhancement, ...

Advantages

- Can be integrated directly into the database
- Enables a faster / more frequent analysis
- Results are stored in the database and can be reused
- Low workload on the client
- Expandability / integration with other data science tools

Outlook

- Promising results regarding reuse and integration into database systems
- Further development is advised
- Performance improvements through the use of database features such as sharding and indexing
- Increase capabilities

Questions

[1] http://pm4py.org

[2] https://www.mongodb.com

PM4Py-mongo repository:
https://git.rwth-aachen.de/tom.huelsmann/pm4py-x-mongodb