



JMEOS: A Java binding of the MEOS spatiotemporal library

As part of the defense of **Mareghni Nidhal's master thesis**,
under the supervision of **Pr. Esteban Zimányi**





Table of contents

01

Introduction

02

Requirements

03

Function Wrapper

04

Types
Implementations

05

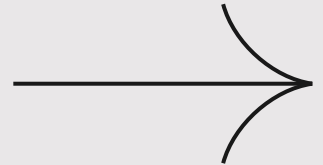
Testing and analysis

06

Use case example

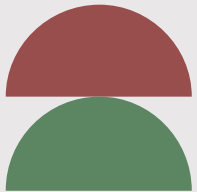
07

Conclusion



01

Introduction



1.1 Context

- Increase in public transport: 376 millions journey in 2022 for STIB.
- Increase of maritime transportation over the world.
- Global warming implies optimization for costs and efficiency.
- How to optimize ? -> New techniques to store and process such "spatiotemporal" data.

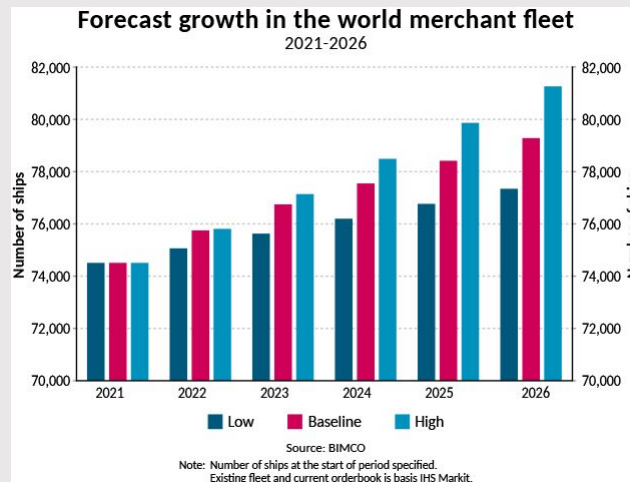
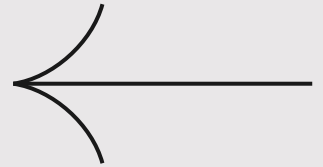


Figure 1:
<https://safety4sea.com/world-fleet-to-increase-during-the-next-five-years/>
Safety4Sea, 25 June 2021.



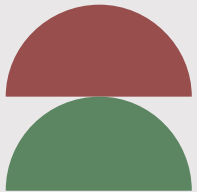
1.2 Motivation

- MobilityDB : expressive MOD, based on PostGreSQL and PostGIS
2300 functions, 40 extended spatiotemporal types.
- MEOS: C library linked to MobilityDB.
- Limitations -> need of DB, SQL language.
- Wider manipulation of spatiotemporal types for new languages.
- PyMEOS: Python binding of MEOS.
- JMEOS: Development on Java, equivalent of PyMEOS.



02

Requirements



2.1 Requirements

Linux and Maven

- Ubuntu 22.04: MobilityDB only for Linux
- MobilityDB and MEOS
- 5 main dependencies: jnr-ffi, jts, guava, maven-plugin and junit

CFFI

- JNR-FFI
- Link between MEOS C code and JAVA
- Simple native access
- Better performance than other FFI
- Automatic conversions of types

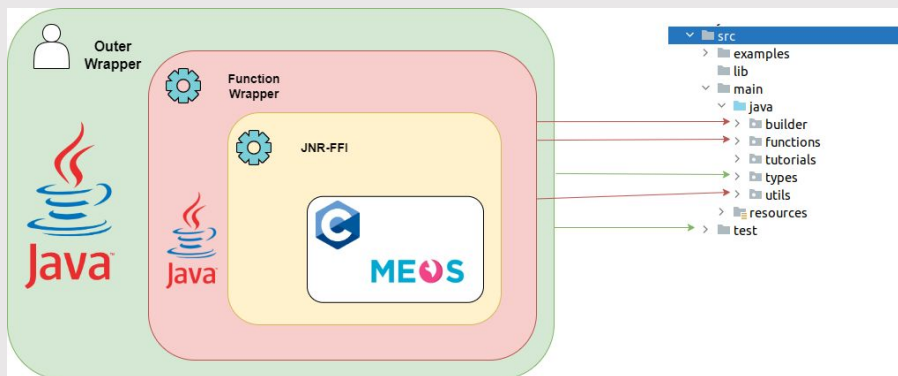
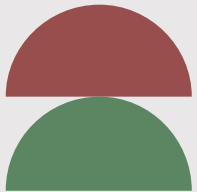
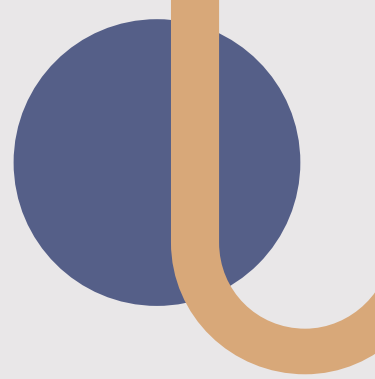


Figure 2: Package wrapper-structure of JMEOS

03

Functions wrapper



3.1 Functions extraction

- Filter the original input file with regex
- Extract functions signatures
- Extract structure types
- Ensure code maintainability
- Ease the functions wrapper process
- Output: meos_functions.h and meos_types.h

C Type	JNR-FFI/Java Type
Timestamp	LocalDateTime
TimestampTz	OffsetDateTime
*	Pointer
*char	String
**char	Pointer
bool	boolean
float	float
double	double
void	void
int	int
short	short
long	long
int8	byte
int16	short
int32	int
int64	long

Figure 3: Types conversions between C and JNR-FFI/Java

3.2 Functions generation

1. Generation of functions signature:
 - Parse meos_functions.h containing signatures
 - Converts C types in Java types
 - Stored in StringBuilder
2. Generation of interface:
 - Create JNR-FFI interface of MEOS
 - Load MEOS library
 - Stored in StringBuilder
3. Generation of the body functions:
 - Combine precedent generations
 - Create return statements of functions
 - Stored in functions.java

```
@SuppressWarnings("unused")
public static Pointer lwpoint_make(int srid, int hasz, int hasm, Pointer p) {
    return MeosLibrary.meos.lwpoint_make(srid, hasz, hasm, p);
}

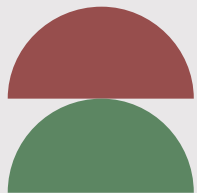
@SuppressWarnings("unused")
public static Pointer lwgeom_from_gserialized(Pointer g) {
    return MeosLibrary.meos.lwgeom_from_gserialized(g);
}

@SuppressWarnings("unused")
public static Pointer gserialized_from_lwgeom(Pointer geom, Pointer size) {
    return MeosLibrary.meos.gserialized_from_lwgeom(geom, size);
}
```

Figure 4: Part of functions.java outputted class

04

Types Implementations





4.1 Boxes

- Boxes: TBox or STBox
 - TBox encapsulates the range of values:
 - `TBOXINT XT([0, 10],[2023-06-01, 2023-06-05])`
 - STBox represent the spatial and temporal extents of an object:
 - `STBOX ZT(((1, 1, 1),(2, 2, 2)), [2019-09-01, 2019-09-02])`
 - Both types inherits from Boxes interface
- 8 Constructors: String, Pointer, Period
 - Conversions and output operations: `to_period`, `toString`
 - Topological operations: `overlaps`, `contains`
 - Position operations: `is_left`, `is_right`
 - Set operations: `intersection`, `union`
 - Comparisons operations: `equals`, `greater`
 - Transformations operations: `get_space`, `expand`

4.2 Abstract collections

- Set \approx one dimension array without duplicates:
 - TimestampSet, IntSet, GeoSet, ...
 - Span \approx range types, not empty or infinite:
 - Period, IntSpan, FloatSpan, ...
 - SpanSet \approx multiranges, same constraints:
 - PeriodSet, FloatSpanSet, ...
- Constructor:String and Pointer but how ?
 - Topological operations: is_contained_in, overlaps
 - Position operations: is_left, is_right
 - Distance operations: distance
 - Comparisons operations: lessthan, notequal

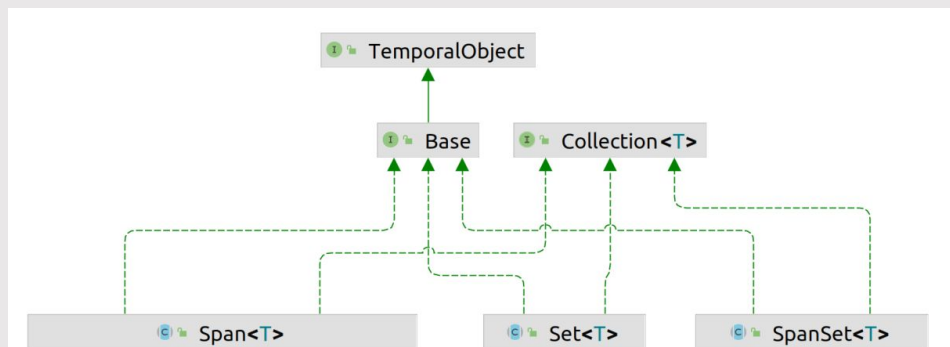


Figure 5: Base types class structure

4.3 Number

- FloatSet/IntSet: set of float/int values
 - FloatSet ("{1 , 3 , 56}")
- FloatSpan/IntSpan: range of float/int values
 - IntSpan ("[7 , 10]")
- FloatSpanSet/IntSpanSet: set of FloatSpan/IntSpan
 - IntSpanSet ("[[8, 10], [11,12]]")

- Constructors: strings and/or values with boolean bounds
- Conversions operations: to_span, to_spanset
- Accessors operations: start/end element
- Transformations operations: shift,scale

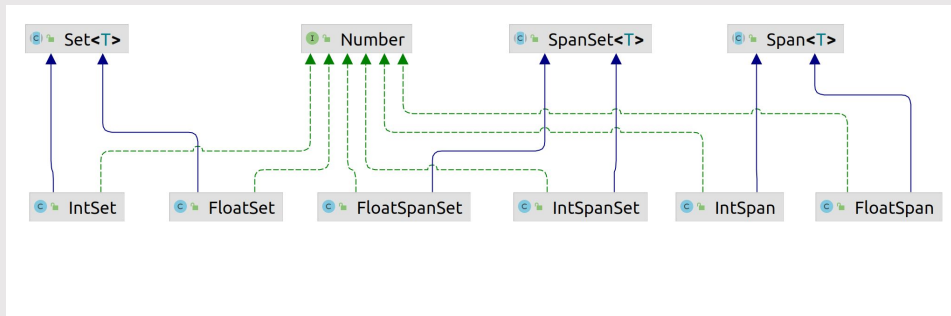



Figure 6: Number package class structure



4.4 Time



- Period is bounded time interval, Span of LocalDateTime:
 - Period ("(2019 -09 -08 00:00:00+00, 2019 -09 -10 00:00:00+00)")
 - PeriodSet is a set of Period, SpanSet of LocalDateTime
 - TimestampSet is a set of LocalDateTime:
 - TimestampSet ("{2019 -09 -01 00:00:00+0 , 2019 -09 -02 00:00:00+0 , 2019 -09 -03 00:00:00+0}")
 - Constructors: strings, set of Periods or set of Timestamp
 - Accessors operations: lower, upper
 - Topological operations: is_same
 - Position operations: is_after, is_before
- 

4.5 Temporal Types

- Capture evolution of base type over time:
 - TBool, TText, TInt, TGeomPoint, ...
- Abstract class with 3 sub-types:
 - TInstant: $v@t$
 - TSequence : $[v_0@t_0, v_1@t_1, v_2@t_2, v_3@t_3]$
 - TSequenceSet: $[v_0@t_0, v_1@t_1], [v_2@t_2, v_3@t_3]$
- Temporal class contains accessors, modifications, transformations, restrictions and other operations
- Inheritance + Interface implementation of base type.

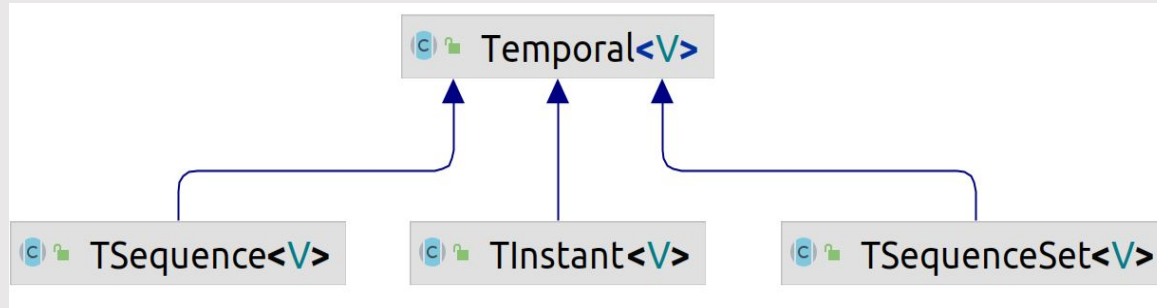



Figure 7: Temporal package structure



4.6 TBool, TText, TInt, TFloat



- TBool: true@2020-01-01 08:00:00+00
 - TText: "Open"@2020-01-01 09:00:00+00
 - TInt and TFloat: 5@2020-01-01 10:00:00+00
 - Behavior and methods should be inherited, but multiples ones not possible in java -> interfaces
 - Instantiation uses same sketch as collections
 - Specific methods: conversions, mathematical, accessors, ...
- 

4.7 TGeomPoint and TGeogPoint

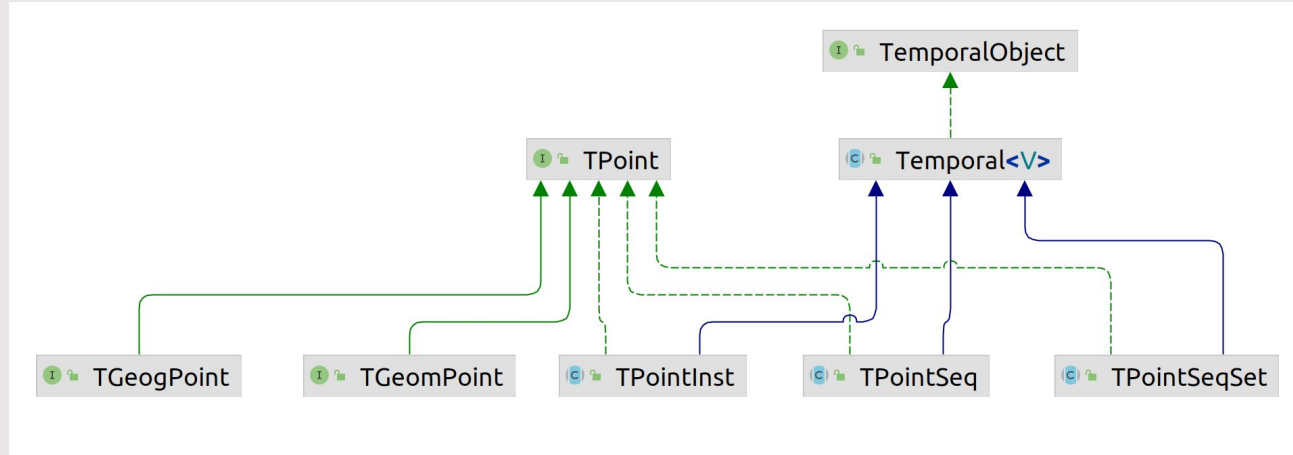
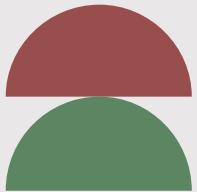
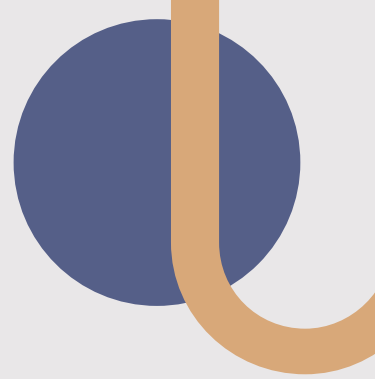


Figure 8: Temporal Point classes structure

- TGeomPoint, geometrical point: `TGeomPointInst("Point (1.5 1.5) @2019 -09 -01")`
- TGeogPoint, geographical point:
`TGeogPointInst("Point (1.5 1.5) @2019 -09 -01")`

05

Testing and analysis





5.1 Unit Tests and Deployment



- > 1,100 unit test
- Parameterized tests avoid duplication
- Ensure individual components works
- Enhance maintainability of the code
- JAR file through Maven
- Problem: MEOS depends on machine
- Solution -> Dockerization
- Docker image based on debian with all dependencies

5.2 Code Analysis

- Detects syntax, complexity, security risks early.
- Lowers bug-fixing costs in development cycle.
- Complements dynamic testing, focuses on code syntax.
- Identifies security flaws like SQL injections, buffer overflows.
- Promotes coding standards, enhances code readability.

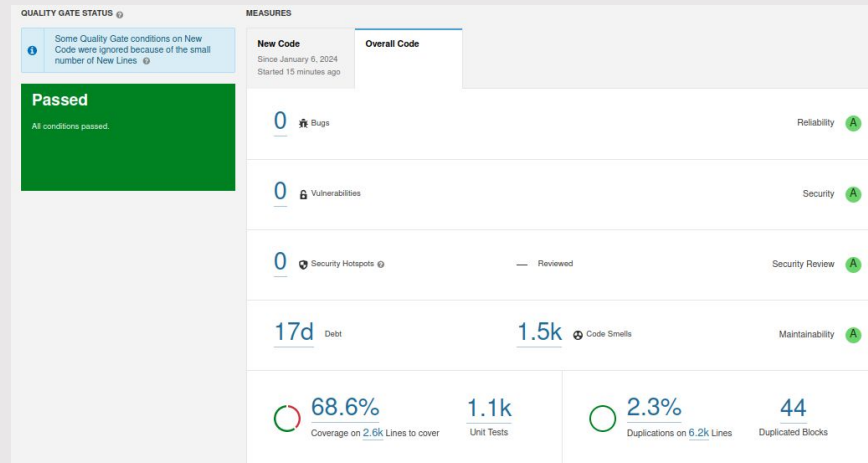
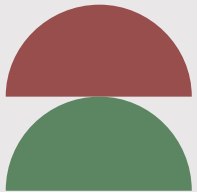
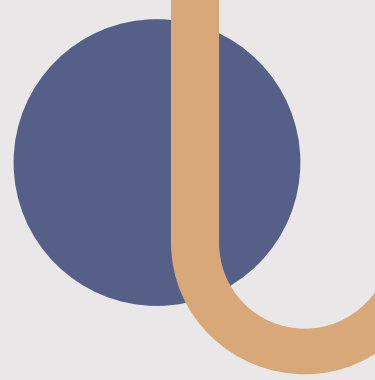


Figure 9: SonarQube analysis results on JMEOS

06

Use case examples



6.1 Motivation and files

- Important evaluate nesting of all types/functions
- Complete unit tests
- Real-world applicability of JMEOS
- Multiple examples files: hello_world, read_ais, disassemble_berlinMOD
- Allows evaluation of performance and comparisons with PyMEOS/MEOS.

6.2 Description of examples

- Hello_World: Creation of geometrical point and output in a mfjson format through Temporal method
- Read_AIS: Process AIS data using by converting CSV file records into temporal instances and sets.
- Simplify: Simplify berlinmod demonstrates CSV-based synthetic trip data processing and spatiotemporal simplification.

```
Vehicle: 5, Date: 2020-06-02, Seq: 1, No. of instants: 1, No. of instants DP: 91, No. of instants SED: 566
Vehicle: 5, Date: 2020-06-02, Seq: 2, No. of instants: 1, No. of instants DP: 81, No. of instants SED: 526
Vehicle: 5, Date: 2020-06-02, Seq: 3, No. of instants: 1, No. of instants DP: 16, No. of instants SED: 83
Vehicle: 5, Date: 2020-06-02, Seq: 4, No. of instants: 1, No. of instants DP: 15, No. of instants SED: 84
Vehicle: 5, Date: 2020-06-03, Seq: 1, No. of instants: 1, No. of instants DP: 91, No. of instants SED: 540
Vehicle: 5, Date: 2020-06-03, Seq: 2, No. of instants: 1, No. of instants DP: 81, No. of instants SED: 536
Vehicle: 5, Date: 2020-06-03, Seq: 3, No. of instants: 1, No. of instants DP: 17, No. of instants SED: 59
Vehicle: 5, Date: 2020-06-03, Seq: 4, No. of instants: 1, No. of instants DP: 18, No. of instants SED: 73
Vehicle: 5, Date: 2020-06-04, Seq: 1, No. of instants: 1, No. of instants DP: 91, No. of instants SED: 555
Vehicle: 5, Date: 2020-06-04, Seq: 2, No. of instants: 1, No. of instants DP: 81, No. of instants SED: 531
Vehicle: 5, Date: 2020-06-04, Seq: 3, No. of instants: 1, No. of instants DP: 23, No. of instants SED: 117
Vehicle: 5, Date: 2020-06-04, Seq: 4, No. of instants: 1, No. of instants DP: 117, No. of instants SED: 675
Vehicle: 5, Date: 2020-06-04, Seq: 5, No. of instants: 1, No. of instants DP: 57, No. of instants SED: 328
Vehicle: 5, Date: 2020-06-04, Seq: 6, No. of instants: 1, No. of instants DP: 16, No. of instants SED: 64
Time taken:
0.909373079
```

Figure 10: Simplify_berlinmod example output

6.3 Benchmark of read_ais file

- Machine used:
 - Dataset: AIS data from dma.dk
 - 3 scales with 0.2, 0.5 and 1.
 - Basic scale 1 = 1 millions lines.

Processor	Intel Core i3-6100U, 2.30 GHz
Cores/Threads	2 cores, 4 threads
Cache	3 MB
Graphics	Intel HD Graphics 520
Memory	8 GB RAM, DDR4, 2133 MHz
Storage	256 GB SSD

2021-01-08 00:00:00	265513270	57.059	12.272388	0
2021-01-08 00:00:01	219027804	55.94244	11.866278	0
2021-01-08 00:00:01	265513270	57.059	12.272388	0
...

6.4 Results

- Averaged over 5 iterations
- JMEOS - MEOS - PyMEOS
- JMEOS 35-50% faster than PyMEOS. MEOS still 6x better.

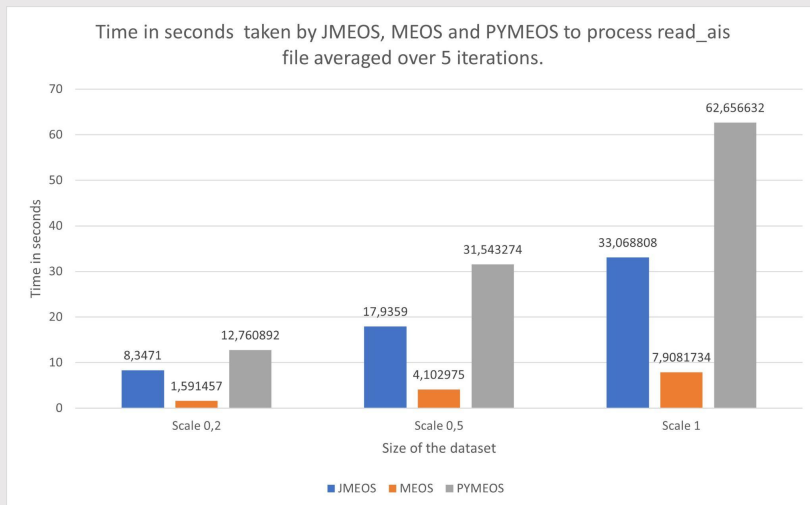


Figure 11: Time efficiency of JMEOS/MEOS/PyMEOS on read_ais file

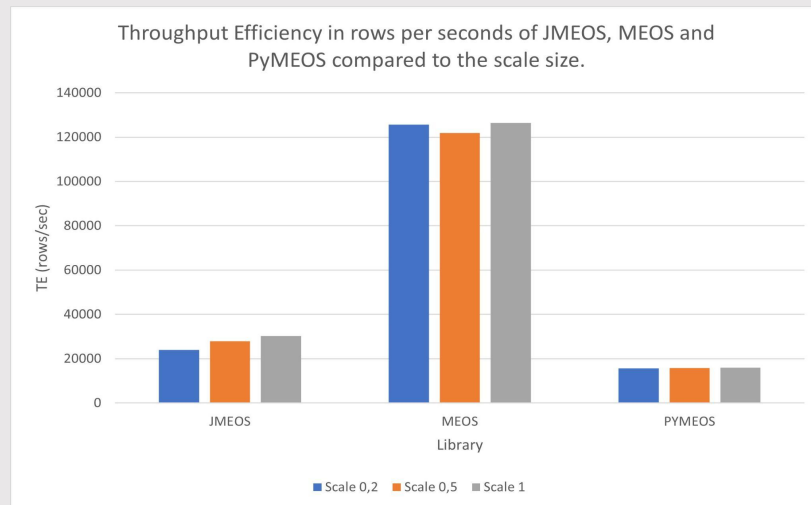
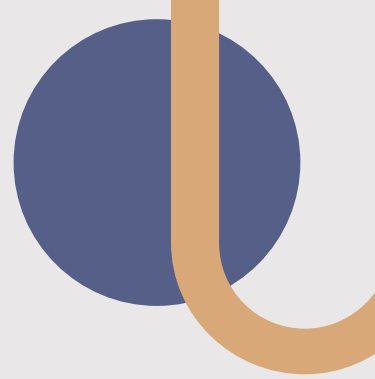
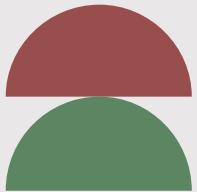


Figure 12: Throughput efficiency of JMEOS/MEOS/PyMEOS on read_ais file

07



Future work and conclusion



7.1 Future Work

Error Handling

Improve JMEOS error handling and debuggability.

New Tests

Achieve complete test coverage in JMEOS.

Last Methods

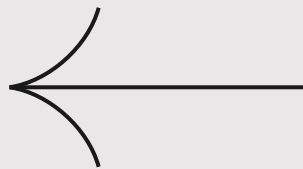
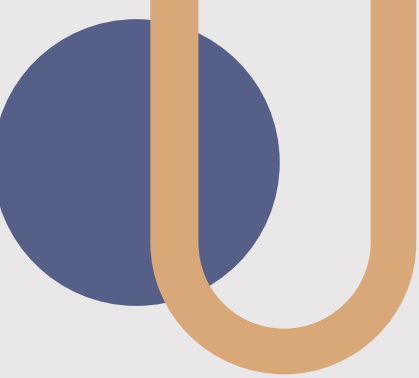
Implement remaining JMEOS methods to achieve 100% binding.

New examples

Add real-world JMEOS usage examples.

New bindings

Develop additional MEOS language bindings.



Thanks !

Do you have any questions ?

